On-chip Communication in the Many-core Era

DIETI - Department of Electrical Engineering and Information Technology
University of Naples Federico II

A dissertation submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Author:
Edoardo Fusella

PhD Supervisor:
Prof. Antonino Mazzeo

PhD Co-Supervisor:
Prof. Alessandro Cilardo

PhD Coordinator:
Prof. Francesco Garofalo

March 2015

This dissertation is dedicated to my future daughter, with a hope that she would one day realise that education is a weapon to fight ignorance and poverty and a key to open doors for turning dreams into reality.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

Edoardo Fusella
March 2015

Acknowledgements

First of all, I would like to acknowledge my supervisor, Prof. Antonino Mazzeo, who made my PhD possible. He trusted me and gave me the opportunity to be involved in his teaching activity.

Then, I really thank Prof. Alessandro Cilardo, my co-supervisor, for his understanding, support and help, which it has been decisive for me to accomplish this thesis. I still remember how I felt at the beginning of my PhD. During the first months I was completely lost. I didn't know what to do or what I was going to do. He helped me to step forward in a moment of disorientation, he offered me ideas and a lot of work. His kindness and professionalism are unique, and I will never end to thank him for the opportunity that he gave me and for everything he taught to me. Since we met, he has made the effort to get the best out of me.

Ing. Luca Gallo and I have worked side by side during the beginning of my PhD. Together we solved many problems, we published some papers and we attended several conferences. His support has been invaluable in a time when it all seems to be difficult.

Then, I would like to gratefully and sincerely thank Prof. José Flich for giving me the chance to come to the Universitat Politècnica de València. I have been extremely lucky to have a supervisor who cared so much about my work, and who responded to my questions and queries so promptly. I really enjoyed working with him and I hope to go on collaborating with him.

Finally, I wish also to acknowledge all the members of the Department of Electrical Engineering and Information Technologies of the University of Naples Federico II and of the Parallel Arquitectures Group of the Universitat Politècnica de València for creating a pleasant and cheerful work atmosphere.

Abstract

Electronic system design is being revolutionized by the widespread adoption of the multi- and many-core paradigms. As the number of elements on a single chip and their performance continue to increase, the communication architecture is gradually becoming the key ingredient determining various trade-offs between costs and performance: on-chip interconnects provide a vital facility for highly parallel systems, particularly in data intensive applications, where the choice of the underlying communication architecture, tailored on the particular application requirements, is critical to the whole architecture.

In that respect, the first part of this thesis goes through the main options available for building different on-chip communication architectures, focusing on the design automation of structured communication architectures based on crossbars and shared buses connected through bridges. An automated methodology for optimizing many-core interconnect architectures, based on the application communication requirements, is presented. The proposed methodology turns the description of the application communication requirements into an on-chip synthesizable interconnection structure satisfying given area constraints. In addition, it could take into account possible dependencies between tasks in order to co-optimize the communication scheduling and interconnect synthesis.

However, it is increasingly challenging for an electrical interconnect to meet power constraints since the power dissipation of electrical on-chip networks poorly scales with performance leading to a growing energy cost. Silicon Photonics seems to be able to empower ultra-high bandwidth with low power consumption: nanophotonic waveguides (the photonic equivalent of a wire) can achieve bandwidths in the Tb/s range, while photonic signaling consumes less power than electrical interconnects since the energy consumption necessary to send a message optically is independent of the bitrate and the distance between the two end-points with no need for power consuming repeaters, regenerators or buffers. However, designing an optical on-chip network requires addressing several challenges that have no equivalent in the electronic domain.

The photonics inability to perform inflight buffering and logic suggests the use of hybrid architectures made up of a photonic circuit-switched and an electronic packet-switched networks. In this regard, in the second part of this thesis, we first propose a new power-aware path-setup protocol able to put allocated circuits on a stand-by state, rapidly recovering them when needed. Then, a novel hybrid Optical-Electronic NoC named H^2ONoC is presented. Thanks to a hybrid optical topology, it is possible to achieve higher bandwidth values and constant energy dissipation regardless of the network and traffic size. Compared to previously proposed architectures, H^2ONoC exhibits higher throughput, lower latency, and improved energy efficiency with heavy traffic.

Therefore the main contribution of the thesis is twofold: engineering as the integration of the proposed design methods into tools to automate the interconnect design steps has direct applicability for designing multi- and many-core systems, and scientific since advanced and original communication architectures exploiting silicon photonics are presented.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Trends in electronic system design

The continued scaling of CMOS technology has led to double the number of on-chip components (i.e. transistors) every two years. Until the early 2000s, this growing chip capacity has been used to build larger microprocessor cores operating at higher frequencies. However, the traditional single processor frequency scaling paradigm introduces a power issue: increases in frequency thus increase the amount of power dissipated. On these grounds, the semiconductor industry has moved from the single processor frequency scaling entering the era of parallelization. Electronic system design is being revolutionized by the widespread adoption of the many-core paradigm. As an example, an NVIDIA GK110 GPU has more than 7 billion transistors enabling the integration of 2880 single- and 960 double-precision CUDA cores in a single chip [4]. In a few years, we will be able to build many-core systems with thousands of small cores as well as large memory elements (MEs) such as shared and local caches [22].

According to its inner architecture, a many-core system can be classified into two categories: Chip MultiProcessor (CMP) and MultiProcessor System-on-Chip (MPSoC). CMPs are usually symmetric systems made up of several homogeneous cores each with its own private memory/cache. A task could be assigned to any core and may also be moved around from a core to another at will. This CMP architecture seems reasonable for general purpose processing to run different applications that prevent from customizing the architecture at design time. Differently, MPSoCs consist of general purpose cores coupled with processing elements with specific functionalities reflecting the need of the expected application domain, e.g. graphics engines, as well as some memory and I/O elements. MPSoCs are well suited for the embedded domain where the system architecture is tailored on the needs of a specific application.

The main advantage of many-core systems is that they exhibit better performance and energy efficiency compared to traditional systems as well as they ensure linear performance improvement with complexity and power [22]. In addition there are some other benefits such as: 1) energy can be saved by turning off the unused cores or by using different voltages and frequencies for each core; 2) the processing load can be split across different cores in order to distribute and lower the die temperatures. However, highly parallel computing systems have to face serious communication challenges. In a many-core system, the on-chip communication architecture (OCA) provides a vital facility, enabling the computation to be distributed among the different PEs and data to be scattered across the MEs. In particular, the choice of the underlying OCA in such systems is a critical design step since it affects the entire inter-component data traffic and impacts the overall system performance and cost [112]: the inter-core communication could easily become a bottleneck leading to the saturation of performance increase with the number of processing elements. Not surprisingly, the industry and the academia are continuously introducing new architectures and components dictating the evolution of on-chip communication. Furthermore, to meet the tight time-to-market constraints and efficiently handle the design complexity, we also need suitable computer-aided design (CAD) tools supporting the automation of these tasks.

1.2 On-chip communication

Shared buses are the traditional and simplest on-chip communication architecture, consisting of a set of shared parallel wires connected to all components in the system. At any given time, only one PE can drive the bus. This limits the achievable concurrency of the system, which makes shared buses non-scalable and unsuitable for highly parallel applications. Advanced shared buses appeared during the last years have introduced several specific features, such as separation of address/control and data phases, pipelined operations, burst-based, split and out-of-order transactions, etc. [120]. A few examples include AMBA™ (AHB/APB) [1] by ARM® and CoreConnect™ (PLB/OPB) [61] by IBM®. In addition, several research works have focused on improving some features of the shared bus architecture itself, such as the arbitration schemes [66, 79, 86, 127, 129]. However, single shared buses remain a major performance bottleneck for the majority of highly concurrent applications due to their inherent lack of parallelism.

To overcome the limits of shared-buses, hierarchical architectures consisting of several buses interconnected through bridge components were introduced [120]. In addition to improving the potential bandwidth, this approach introduces a new dimension in the design space, the definition of the topology of the interconnect, which becomes the main parameter affecting the overall performance of the communication architecture. In fact, based on the positions of the bridges, we can essentially build any topology. As an example, a ring can be built as a group of consecutive bus segments, which can operate in parallel, connected through bridges, which may be mono- or bi-directional. To take this new design dimension into account, several newly introduced design methodologies include automated topology synthesis as a central step for improving performance.

Crossbars, also called bus matrices, are another major design alternative. They consist of a multi-layered communication architecture with multiple buses operating in parallel connecting multiple inputs to multiple outputs in a matrix-like scheme. A crossbar can be full or partial depending on the required connectivity between inputs and outputs. If a crossbar with M inputs and N outputs is full, it is essentially made of a matrix with $M \times N$ cross-points or places where the buses intersect. Partial crossbars require of course less buses. The adoption of crossbar interconnects in MPSoC communication architectures has recently been accelerated by the support for crossbar solutions introduced by many commercial technologies, such as AMBA™ AXI™ [8] by ARM® and the STbus™ [2] by STMicroelectronics®. A crossbar overcomes most of the limitations of the buses, although this benefit usually comes at non-negligible area and power costs [89, 131]. As a consequence, several design methods to reduce area [58, 68, 69, 105, 122, 123, 163], power [63, 103, 107], or both [57, 65, 67, 164] were proposed. These approaches mainly optimize the interconnect for the communication patterns of a given application. Unfortunately, as the number of IP cores connected to a crossbar grows, the increased complexity of the interconnect circuit may easily result in lower clock frequencies, especially for ASIC implementations. To address this issue, the cascaded crossbar paradigm, consisting of crossbars connected with each other in a cascaded manner, was proposed by some of the above works [63, 65, 67–69, 107, 163, 164].

Shared buses, crossbars, and any combination of them, such as hierarchical buses and cascaded crossbars, possibly connected to each other in a heterogeneous scheme, are collectively referred to here as bus-based on-chip interconnect architectures. The major alternative to bus-based interconnects is provided by *networks-on-chip* (NoCs) [14, 34, 51]. In a NoC, packet switched network fabrics are used to transfer data between on-chip components. Bringing the packet-based communication paradigm to the on-chip domain, NoCs require the fundamental building blocks used in traditional multi-computer systems, i.e., switches, also called routers in this context, Network Interfaces (NIs), also called network adapters, and links [36]. NoC can be customized by placing these components according to a specific topology and configuring their features in terms of buffer depth, arbitration policy, flow control mechanism, etc. A large body of research works deal with NoC customization and design space exploration. Refers to [10, 132, 133] for an overview of the current state of the art on NoCs.

1.3 Thesis contribution

This thesis focuses on two different aspects of the on-chip communication architecture. First, methods for the automatic design space exploration of bus-based interconnects, satisfying given performance and cost constraints, are investigated. Then, new hybrid photonic-electronic NoCs are designed in order to propose alternative and innovative solution able to overcome the physical limitations of electronic interconnects. Note that computation and communication are decoupled from each other: this work completely neglects the design of the computation side while deeply explores the communication architecture. On one hand, this thesis brings a strong engineering contribution as the proposed design methods are oriented to solve real problems related to MPSoC design. In particular, the integration of the presented algorithms into tools to automate the interconnect design steps has direct applicability for designing MPSoCs. On the other hand, the thesis has made a scientific contribution since advanced and original communication architectures exploiting silicon photonics are presented. Designing photonic communication architectures requires to face several challenges and to deal with metrics that have no electronic equivalent. In this thesis these aspects are extensively discussed and handled bring new contributions to the research community.

1.4 Thesis overview

This thesis is organized in four main parts. First, an introductory section illustrates the state of the art of the on-chip communication design addressing both the analysis and synthesis aspects. After that, the second section of the thesis contains the main contributions in the area of the design automation of the communication architecture. The third section introduces the photonic communication and the chip-scale photonic interconnects with their distinctive properties. Finally, the fourth section addresses the design of hybrid photonic-electronic NoCs and introduces new architecture exploiting silicon photonics.

Section I (Chapter 2) The essential aim of this chapter is to fill this gap by presenting an extensive review of state-of-the-art design automation techniques for application-specific on-chip interconnects. The chapter goes through the main options available for building different on-chip interconnect topologies, discussing the details of hierarchical buses, crossbars, and cascaded crossbars as well as the approaches that can be adopted to formalize the description of such topologies and the related parameters of interest. Then, the chapter surveys the most relevant techniques proposed in the literature to analyze a given interconnect solution, i.e. quantify parameters such as latency, bandwidth, area cost, power consumption, operating frequency, followed by an in-depth review of the main approaches for interconnect synthesis, including several advanced aspects such as co-synthesis of memory and communication architectures, joint scheduling and interconnect synthesis, floorplanning, dynamic configuration, multi-path communication.

Section II (Chapters 3, 4) These chapters propose automated methodologies to search the interconnect design space, avoiding a manual and time consuming try-and-error processes. The methodology presented in Chapter 3 turns the description of the application communication requirements into an on-chip synthesizable interconnection structure satisfying given area constraints. Chapter 4 improves the approach proposed in Chapter 3 by taking into account possible dependencies between communication tasks, which further constrains the design space making the identification of a feasible solution more challenging. Targeted at FPGA technologies, these approaches combine crossbars and shared buses, connected through bridges, yielding a scalable, efficient structure. The resulting architecture improves the level of communication parallelism that can be exploited, while keeping area requirements low. The chapters thoroughly describe the formalisms and the methodologies used to derive such optimized heterogeneous topologies. They also discuss some case-studies emphasizing the impact of the proposed approaches and highlighting the essential differences with a few other solutions presented in the technical literature.

Section III (Chapter 5) This chapter first explains the unique and distinctive properties of the silicon optics technology and then provides a broadband overview of the opportunities and challenges posed at the architecture level by this new interconnect paradigm. In fact, silicon Photonics introduces many new design tradeoffs, having no electronic equivalent, involving aspects such as wavelength selectivity, physical constraints, and spectral parallelism. The text analyses a number of such tradeoffs involving relevant photonic features and compares different design choices. For instance, nanophotonic waveguides can achieve ultra-high bandwidths by exploiting wavelength division multiplexing (WDM), but the use of wavelength selectivity to implement bandwidth aggregation prevents the exploitation of wavelength arbitration and routing, introducing non-trivial design dilemmas. As highlighted throughout the paper, getting a cross-cutting understanding of these trade-offs is essential for harnessing the full potential of on-chip Photonics.

Section IV (Chapters 6, 7) These chapters propose some novel hybrid photonic-electronic NoC architectures aiming to improve the performance and to reduce the cost compared with previous approaches proposed in the literature. In particular, Chapter 6 presents a new power-aware path-setup protocol able to put allocated circuits on a stand-by state, rapidly recovering them when needed. Then it compares in terms of performance and energy consumption some path-setup architectural solutions that differ from each other in the routing algorithm, the path-setup protocol and the deadlock avoidance technique. This comparison could help manycore system designers to select the most appropriate path-setup architecture according to traffic characteristics and network size. Differently, Chapter 7 proposes a novel hybrid Optical-Electronic NoC named H^2ONoC. H^2ONoC exploits hybrid topologies in the photonic layer enabling reduced insertion loss and hence improving the actual bandwidth offered by the interconnect. An analytical model describing the performance of the hybrid topology, used to characterized the worst-case insertion loss is presented. In addition the chapter addresses all the relevant aspects of the H^2 ONoC architecture, including the routing algorithm, the flow control mechanism, the deadlock management policy, the path-setup protocol, the electronic router and the photonic switches microarchitectures. Finally, the experimental setup and some comparisons with a few purely electronic and hybrid reference NoCs are shown.

Chapter 2

Design Methodologies for Bus-based Interconnects: Background

To exploit the MPSoC potential benefits to the fullest, suitable design methodologies are required for addressing two different facets of system design [72]. First, it is essential to properly map the application's computation requirements to a set of PEs like CPUs, DSPs, application specific cores, etc. Second, it is equally necessary to map the system's communication requirements onto an optimized communication architecture possibly tailored on the specific application. The separation between communication and computation is key [72]. This feature enables the system designer to explore the communication architecture independent of processing element selection and mapping. In particular, the choice of the underlying communication architecture in data-intensive applications, tailored on the particular application requirements, is a critical design step since the amount of communication among functional blocks critically determines the global performance [153]. This chapter focuses on the design automation of a broad class of communication architectures, here referred to as *bus-based on-chip interconnects*. While this class includes usual shared buses and full crossbars, suffering from either limited performance or poor scalability, it also embraces complex scalable high-performance interconnects with customizable topologies such as hierarchical architectures, rings, cascaded crossbars as well as ad-hoc heterogeneous networks made up of a combination of shared buses and crossbars. The essential aim of this chapter is to fill this gap by presenting an extensive review of state-of-the-art design automation techniques for application-specific on-chip interconnects. The chapter will go through the main options available for building different on-chip interconnect topologies, discussing the details of hierarchical buses, crossbars, and cascaded crossbars (Section 2.1). The treatment will first focus on the approaches that can be adopted to formalize the description of such topologies and the related parameters of interest (Section 2.2). Then, the chapter will survey the most relevant techniques proposed in the literature to *analyze* a given interconnect solution, i.e. quantify physical parameters such as latency, bandwidth, area cost, power consumption, operating frequency, for a fixed interconnect, by either analytical methods, simulation, or hybrid approaches (Section 2.3). Subsequently, the chapter will present and in-depth review of the main approaches available for interconnect *synthesis*, where the interconnect topology is the outcome of an automated design process aimed at a given objective function under a set of constraints, involving a certain combination of the above physical parameters (Section 2.4). Table 2.1 contains a synoptic overview of the approaches surveyed in the chapter, listing for each the topologies, the type of evaluation adopted, the combination of physical parameters chosen as objective functions and constraints, as well as other relevant features.

Table 2.1 Review of design methodologies for bus-based on-chip interconnect.

Reference	**Topologies**(1)	**Evaluation type**	**Objective function**(2)	**Constraints**(2)	**Other features**
[11]	P2P	Analytical	$L \wedge Pw \wedge A$	$L \wedge A \wedge$ Fanout	Joint Scheduling
[30]	Heter. Arch	Analytical	L	A	
[31]	Heter. Arch	Analytical	L	A	Joint Scheduling
[41]	Hier. bus	Analytical	T	L	Floorplanning
[49]	Casc. cross	n/a	n/a	n/a	
[58]	Crossbar	n/a	A	L	Dynamic Configuration
[57]	Crossbar	n/a	$Pw \wedge A$	L	Dynamic Configuration
[62]	Hier. bus	Analytical	E	L	Mem-Comm Cosynthesis
[63]	Casc. cross	Simulation	Pw	$L \wedge Bw \wedge f \wedge A$	
[65]	Casc. cross	Hybrid	$E \wedge A$	L	
[68]	Casc. cross	Analytical	$f \vee A$	$L \wedge Bw \wedge (A \vee f)$	
[69]	Casc. cross	Analytical	$f \vee A$	$L \wedge Bw \wedge (A \vee f)$	
[67]	Casc. cross	Analytical	$Pw \vee A$	$L \wedge Bw$	Floorplanning
[73]	Hier. bus	Hybrid	L		
[78]	Hier. bus	Hybrid	L		
[81]	Hier. bus	Hybrid	L		
[98]	Hier. bus	Simulation	$L \wedge A$		Floorplanning
[105]	Crossbar	Simulation	A	$L \wedge Bw$	
[103]	Crossbar	Simulation	Pw	$L \wedge Bw$	Floorplanning
[107]	Casc. cross	Analytical	Pw	T	
[118]	Hier. bus	Analytical	$E \wedge A$		
[119]	Hier. bus	Analytical	E	Bw	Mem-Comm Cosynthesis
[124]	Hier. bus	Simulation	$f \wedge A$	$T \wedge$ Wire Delay	Floorplanning
[123]	Crossbar	Hybrid	A	T	
[122]	Crossbar	Hybrid	A	$T \wedge A_{memory}$	Mem-Comm Cosynthesis
[125]	Crossbar	Hybrid	$E \wedge A$	$T \wedge$ Wire Length	Floorplanning
[130]	Hier. bus	Simulation	L		
[134]	Hier. bus	n/a	L		Dynamic Configuration
[140]	Hier. bus	Analytical	Pw		Mem-Comm Cosynthesis
[144]	Hier. bus	Analytical	Custom Metrics	Physical Layout	
[159]	Hier. bus	Analytical	E	L	Joint Scheduling
[163]	Casc. cross	Analytical	A	$L \wedge Bw$	
[164]	Casc. cross	Analytical	$Pw \wedge A$	$L \wedge Bw \wedge f$	Floorplanning

(1) Hier. Bus=hierarchical bus, Casc. Cross.=cascaded crossbar, Arch=heterogeneous bus and crossbar architecture
(2) L=Latency, Bw=Bandwidth, T=Throughput, Pw=Power, E=Energy, A=Area, f=frequency

2.1 Topologies for on-chip bus-based interconnects

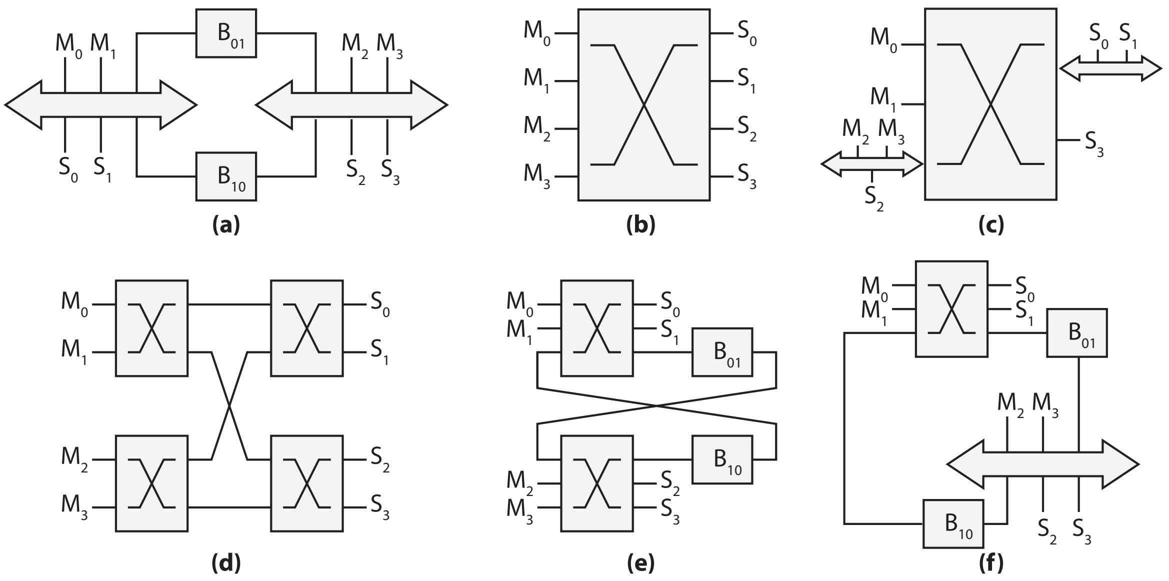


Fig. 2.1 A few examples of topologies. (a) A hierarchical bus. (b) A single crossbar. (c) A crossbar-based architecture with slaves and masters grouped in shared buses. (d) A cascaded crossbar. (e) A clustered crossbar architecture. (f) A clustered heterogeneous bus and crossbar architecture. (*M*: master, *S*: slave, *B*: bridge)

2.1.1 Hierarchical buses

As shown in Figure 2.1(a), the traditional single bus can be extended to a hierarchical bus scheme by bus partitioning (also called bus splitting). In essence, by scattering communication traffic over multiple buses, it is possible to reduce bus contention and improve concurrency [73]. PEs and MEs connected to the same buses form *domains*. Communication interactions in different domains can take place concurrently as long as they do not refer resources residing outside the domain (hereafter we call this kind of parallelism *global parallelism*). Unfortunately, in addition to increasing the available global parallelism, using more buses also causes power and performance penalties because of the hardware overhead caused by the additional bridges, which should be taken into account while solving the partitioning problem [140]. Furthermore, transactions across bridges involve additional time overhead, i.e. longer communication latencies, and make buses inaccessible for other communication interactions during the transfer. Bus splitting algorithms thus usually cluster the IP cores based on the communication profile [78, 140, 141]. Grouping frequently communicating elements in the same local domains is in fact necessary to minimize the communication through the bridges across different buses (*global* or *inter-domain* traffic). In addition, cores attached to different domains should communicate through as few bridges as possible and congestion of individual buses should be taken into account and minimized [41]. The problems of selecting an optimal mapping of PEs to buses and the problem of choosing the best topology are deeply interrelated: a poor mapping leads to degraded performance even on a good topology, while, likewise, a judicious mapping on a bad topology is useless. Bus segments can operate with different protocols and frequencies as well as have different properties, e.g. bus width. This allows IP cores to be placed at the appropriate level in the hierarchy according to the performance level they require: low-performance cores can be placed on low-performance buses so that they do not interfere with higher performance components. Of course, bridges need to support the necessary logic to accurately handle the inter-domain communication between different bus types. According to the positions of bridges, any arbitrary topology can be built. Often, interconnects are based on a chain of bus segments [41, 73, 78], possibly wrapped around to form a ring topology [85], although tree like structures [62, 124] are also frequently adopted.

One of the main differences between a shared bus and a hierarchy of buses is the total bandwidth they provide: since a single bus can be used by only one PE at any time, the bandwidth available to each PE decreases significantly as the bus size increases. Hierarchical bus architectures overcome this restriction only if local domains can work concurrently. As a consequence, these topologies are well suited for applications where clusters of PEs exhibit highly localized communication patterns with a small topological degree of communication. In other words, hierarchical buses target applications where it is easy to find small clusters of highly communicating cores. Figure 2.1(a) shows a hierarchical bus structure made of two buses. In order to make the inter-domain communication feasible, two bridges are placed, leading to a ring topology.

2.1.2 Crossbar-based architectures

The crossbar topology (also called bus matrix) is a multi-layered communication architecture with multiple buses connecting multiple inputs to multiple outputs in a matrix scheme. Figure 2.2 shows the internal architecture of the crossbar in Figure 2.1(b). The input stage is equipped with buffers in order to handle interrupted bursts as well as register and store incoming transfers if receiving slaves cannot accept them immediately. The decode stage generates select signals for the appropriate slaves. Unlike traditional shared bus architectures, arbitration is not centralized, but rather it is distributed, with every slave having its own arbiter. Buses can operate concurrently as long as they do not refer the same resources (we will call this type of parallelism *local parallelism*). A crossbar connecting every input with

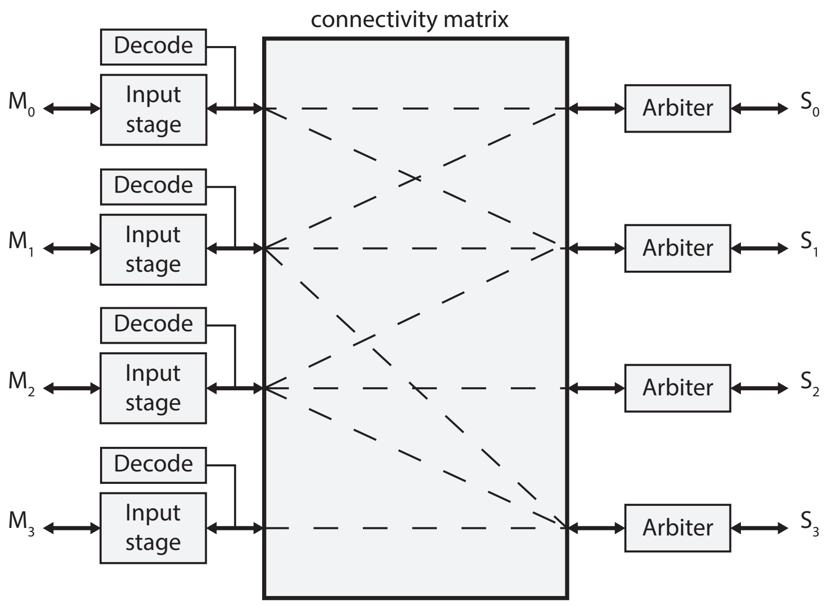


Fig. 2.2 The internal architecture of the partial crossbar in Figure 2.1(b).

every output is called a *full crossbar*. However, based on the actual connectivity required by the application, we can specify a connectivity matrix $z(c)$ indicating only a subset of actually connected input/output pairs. The resulting architecture is called a *partial crossbar*. Since on-chip interconnects are not constrained by a limited off-chip pin count, commercial solutions such as AMBA™ AXI™ by ARM® [8], introduced separated address/data channels in order to increase performance. In such architectures, different topologies may be specified for the two channels: a Multiple-Address Multiple-Data (MAMD) consists of separate data and address crossbars, while a Shared-Address Multiple-Data (SAMD) combines crossbars for the data channels and shared buses for write and read address channels. In fact, in most systems, the address channel bandwidth requirement is significantly less than the data channel. Such systems can achieve a good balance between system performance and interconnect complexity by using a shared address bus with multiple data buses enabling parallel data transfer.

The main advantage of a crossbar is that any parallel application can be mapped to a physical interconnect exhibiting the necessary parallelism. In addition a crossbar is inherently a single-hop latency interconnect. These benefits come at non-negligible area and power costs [89, 131]: the area cost grows quadratically with the number of ports [30] and, consequently, the power consumption has a similar trend [57]. Consequently, single crossbars do not scale well with the number of IP cores. To mitigate this drawback, it is possible to group slaves on shared buses as long as performance constraints are met [122, 123]. In addition, transactions that involve slaves accessed exclusively by a single master do not necessarily have to go through the crossbar: a shared bus can be placed on a input port of the crossbar in order to group these slaves and the corresponding master. Since these shared buses connect at most one master, they do not require additional arbitration components. Such a structure has fewer channels, which reduces the crossbar area in terms of wires and arbiter components and simplifies the design of decoders, reducing in turn the resulting power consumption. A further improvement can be achieved by also grouping masters [103, 105]. Obviously, the resulting structure should closely match the traffic characteristics and performance requirements of the application. By relying on this simplification, the crossbar size can be further reduced. Both approaches need efficient algorithms able to cluster master and slave cores and reduce congestion. They are motivated by the observation that communication patterns of different applications can be effectively handled by different logical topologies as in most cases applications require only a small portion of all-to-all communication [57, 70, 148]. Figure 2.1(c) displays a crossbar-based architecture enhanced as described above in order to reduce the interconnect size. Note that slave S_2 is placed on the same bus of master M_2 since M_2 is the only master involved in the communication with S_2. Using two shared buses to group slaves and masters allows reducing the size of the crossbar from 4×4 to 3×2.

When designing the topology, it is critical to consider the effect of the physical parameters, such as the wire delays. To achieve timing closure of the design, the inherent wiring complexity of alternative topologies should be considered during the synthesis phase. Taking the wiring complexity into account in the early stages of the design cycle will lead to a better and more scalable communication architecture. As the system size grows, shared bus size cannot increase indefinitely and, hence, the central bus matrix may become prohibitively large. As a consequence, the logic depth of the crossbar increases and so do the wires. Since the delay of a wire grows quadratically with its length [54], the added delay will inevitably lower the clock frequency of the bus matrix. Partitioning the wires in segments with repeaters in between [137] leads the total wire delay to become linear with the total wire length. Unfortunately, inserting repeaters increase the cost of the communication architecture in terms of area and power consumption [75]. In order to design high frequency, power efficient crossbar-based architectures under stringent area constraints, the cascaded crossbar paradigm was introduced [63, 65, 67–69, 107, 163, 164]. As exemplified by Figure 2.1(d), a cascaded crossbar topology consists of multiple small crossbars connected to each other without bridges in a cascaded scheme such that each master can access each connected slave through a multi-hop path. A bus pipeline stage, called register slice, can be inserted either between the masters and the crossbar or between the crossbar and the slaves as well as between crossbar pairs for pipelining the interconnect where the critical path is too long, so that the timing can meet the given requirements. Of course, this reduced logic delay comes at the cost of additional latency cycles.

In addition, during the design space exploration of crossbar-based interconnects, it is necessary to determine the connectivity matrix of each crossbar. Using fully connected crossbars limits the area efficiency and the achievable performance due to many unused paths. On the other hand, eliminating the unnecessary connections after the synthesis step only marginally improves the resulting architecture, and does not enlarge the design space [69]. Improved solutions can be reached by considering the partial connection of crossbars simultaneously when determining the topology [67].

2.1.3 Clustered heterogeneous bus and crossbar architecture

The above topologies only consist of either buses or only crossbars. Using shared buses to group slaves or masters at the boundaries of a cascaded crossbar is the only case where buses and crossbars coexist in the same topology. Likewise, different crossbars are used in the same topology only in a cascaded fashion. Clustered heterogeneous topologies (Figure 2.1(e) and Figure 2.1(f)) overcome this limit allowing different components, such as shared buses and crossbars, to coexist in a network [30, 31]. As in hierarchical bus topologies, PEs and MEs connected to the same bus or crossbar form local domains. Communication interactions in different domains can take place concurrently, while local domains may be implemented by inherently concurrent architectures. This enables heterogeneous networks to take advantage of both inter-domain and the local parallelism. Communication across different domains is made possible by means of bridge components involving additional overhead. Usually a single domain is implemented by a single bus or crossbar, but in principle any of the above topologies can be used. Choosing the size of the local domains, in terms of how many IP cores are connected, involves a tradeoff between the two types of parallelism: a few large local domains lead to more local parallelism, while smaller local domains result in increased global parallelism. Each local domain must be connected to the external interconnect via at least one bridge. Each unidirectional bridge requires an additional master port and a slave port, and hence increases the size and cost of the interconnect. However, this may be balanced by the reduced costs enabled by decomposing the interconnect. Clustering IPs according to their communication requirements is critical in order to have a balanced interconnect. The capability of exploiting spatial locality in the communication patterns is key: placing the nodes that communicate more frequently closer to each other allows minimizing the traffic between different domains and matching the localized traffic patterns induced by a given application [116]. This allows an efficient use of the interconnect leading to a higher potential bandwidth as well as a lower energy consumption [115]. Figure 2.1(e) shows a two-domain architecture where each domain is implemented with a crossbar. Local parallelism is enabled by the two crossbars being able to operate concurrently, while different transactions are allowed to take place simultaneously in each crossbar (domain parallelism). Figure 2.1(f) depicts a similar architecture exhibiting less concurrency. The local domain C_1, previously implemented with a crossbar, is now realized as a shared bus.

2.2 Applications and Architectures Description

In this section we introduce some concepts and terminology related to the formal description of the target application and the communication architecture, which is necessary for the analysis and the automated design of optimized interconnects. We assume that the target application is decomposed into a set of computation tasks, via static analysis or simulation, that are already mapped to a set of processing elements (PEs). We refer to a memory mapped communication scenario, where the MEs and I/O devices are mapped to slave nodes, while the PEs, such as CPUs and DMAs, are mapped to master nodes, sharing the same address space within the system. A master can initiate a communication transaction, whereas slaves merely respond to the transactions initiated by masters.

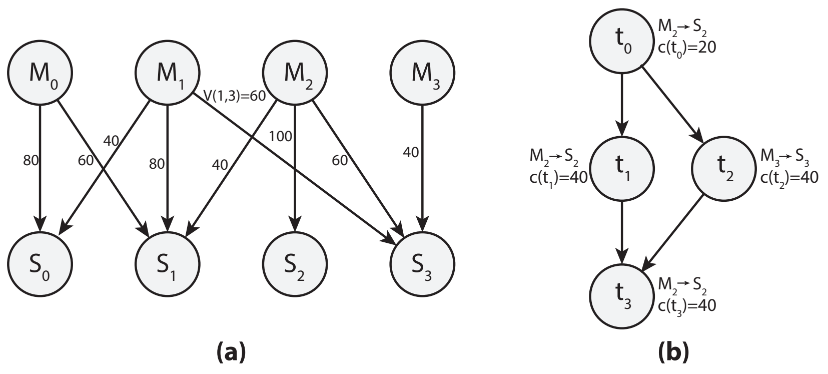

M_0
M_1
M_2
M_3
V(1,3)=60
80
60
40
80
40
100
60
40
S_0
S_1
S_2
S_3
(a)
t_0
$M_2 \rightarrow S_2$ c(t_0)=20
t_1
$M_2 \rightarrow S_2$ c(t_1)=40
t_2
$M_3 \rightarrow S_3$ c(t_2)=40
t_3
$M_2 \rightarrow S_2$ c(t_3)=40
(b)

Fig. 2.3 A few examples (M: master, S: slave, t: task.) (a) A *Communication Graph* (CG). (b) A *Dependency Graph* (DG).

Definition 2.2.1. *A* Communication Graph $CG = G(V_m, V_s, E)$ *is a directed graph, where* $v_m \in V_m$ *and* $v_s \in V_s$ *are, respectively, a master node and a slave node, and* $e_{m,s} \in E$ *is the edge between tasks* v_m *and* v_s *characterizing the communication between them.*

Each edge $e_{m,s}$ can have several attributes expressing application-specific information (e.g., communication volume $V(e_{m,s})$ between vertices v_m and v_s) and design constraints (e.g., latency requirements $L(e_{m,s})$ or communication bandwidth $B(e_{m,s})$). Figure 2.3(a) contains a CG with four masters and four slaves, where the communication volume, here expressed in MB, is specified for each master/slave pair. Concepts similar to the *CG* are defined in most papers addressing the design space exploration for on-chip bus-based interconnects. For instance, solutions based on labeling the edges with the communication volume are adopted in [30, 32, 41, 140, 144], although the graph is called with different names, e.g. *Communication-Connectivity Graph* (CCG), *Core Graph* (CG), and *Communication Graph* (CG). [59] introduces the required bandwidth constraint on each edge. Similarly, [78] presents a *Communication Analysis Graph* (CAG) where the information on an edge includes several statistical properties of the communication transactions involving the two connected cores, such as the number of transactions, the distribution of their sizes (in terms of mean, variance etc.), the critical path information, the number of transactions with zero slack (critical transactions), etc. [123–125] uses a *Communication Throughput Graph* (CTG) where each edge is weighted with a throughput constraint. Last, [67, 68, 163, 164] introduces the *communication trace graph* (CTG), where both the bandwidth requirement and latency constraint are included on each edge. Later on, [69] also includes the required frequency lower bound.

Since we refer to an application-driven design, the CG is the main data structure necessary to guide the whole design process. We assume that the communication traffic between a master and a slave is made up of tasks, which are non-preemptive atomic entities with an arbitrary load, i.e. the amount of transmitted bytes, although two different tasks can have the same master/slave pair. This allows modeling any traffic pattern. Traffic within the same interconnect is called local, while traffic between master-slave pair mapped to different interconnects is called global.

Definition 2.2.2. *A* Dependency Graph $DG = G(V_t, E)$ *(sometimes called* task graph*) is a directed acyclic graph, where $v_i \in V_t$ is a communication task and $e_{v_i,v_j} \in E$ is the edge between $v_i \in V_t$ and $v_j \in V_t$, representing the dependencies, i.e. precedence relationships, between the two tasks.*

Since the graph is acyclic, it establishes a partial order relationship ($\leq$) on its vertices, where the relationship $v_i \leq v_j$ occurs when there exists a directed path from v_i to v_j. Notice that the inputs and outputs of a node do not convey data, but they only express communication dependency constraints. A node with no parents is called *source*, while a node with no children is called *sink*. Here, the communication cost (e.g. the communication volume of a task) is represented by the weight of a node, denoted $c(v_i)$, while the weight on an edge is representative of the computation cost, denoted $w(v_i, v_j)$. This cost represents the delay, due to the computation, taking place between two consecutive communication tasks. Figure 2.3(b) shows a part of a DG with four tasks where, for each task, the master/slave pair and the amount of traffic they exchange are specified. Notice that the communication volumes between master M_2 and slave S_2 and between master M_3 and slave S_3 are respectively made up of three and one tasks. Concepts equivalent to the above definition of DG are present in [11, 31, 159] and are often adopted when scheduling problems are addressed.

While the above concepts are essential in that they formalize the input of the design space exploration methodology, it is equally necessary to describe formally the output communication architecture. We chose the following formalism as it is general enough to describe the full spectrum of bus-based on-chip interconnect architectures.

Definition 2.2.3. *A bus-based on-chip communication architecture (OCA) can be uniquely described by a pair $Arch(T(Sb,C,B),\Omega(V))$, where*

1. *the labeled graph $T(Sb,C,B)$ represents the network topology. The shared buses, crossbars, and bridges are given by the sets Sb, C, and B respectively;*
2. *$\Omega : V_m \cup V_s \rightarrow Sb \cup C$ is a function mapping each vertex $v \in V_m \cup V_s$ in the CG to a bus in Sb or a crossbar in C.*

Taking as example the topology of Figure 2.1(a), the Sb set contains two 3×3 buses, the C set is empty, while the B set contains the two unidirectional bridges between the two buses. The Ω function is responsible for the assignment of the masters and slaves in the topology. In this example, it maps masters M_0, M_1, and slaves S_0, S_1 to the first bus while the remaining PEs and MEs are mapped to the second bus. For each crossbar $c \in C$, $z(c)$ specifies the inner cross-points connectivity. As an example, considering the 4×4 crossbar in Figure 2.2, we have $z(c) = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$, which means that master M_0 is connected to slaves S_0, S_1, master M_1 is connected to slaves S_0, S_1, S_2, and so on. Notice that different portions of the interconnect may be connected via bridges or just wires. While wires directly connect all master interface signals of an interconnect portion to all slave interface signals of another portion without logic or storage capabilities, a bridge provides decoupling facilities, letting the two interconnect portions operate concurrently. Wires can connect only interconnect portions with the same properties such as protocol or frequency. On the contrary, bridges usually have the required logic to provide protocol or frequency conversion, adaptability to different data widths as well as some advanced features such as burst or split transaction support.

2.3 Quantitative analysis of an interconnect

This section will first recapitulate the issues related to the quantification of the most relevant design metrics, i.e. parameters like latency, bandwidth, area cost, power consumption, operating frequency, for a given interconnect solution, followed by a review of the main evaluation approaches, including analytical methods, simulation, and hybrid methods.

2.3.1 Design metrics

An optimal architecture is defined under a set of constraints while minimizing/maximizing one or more objective functions. In order for such a process to take place it is essential to define significant quantitative design metrics and rely on suitable techniques to evaluate or estimate them. Table 2.2 provides a few examples of design metrics, thoroughly illustrated in the following.

Table 2.2 Examples of design metrics. The comparisons refer to the topologies in Figure 2.1 and the CG in the Figure 2.3(a)

Topology	**Latency** $(Mcycles)$	**Bandwidth Utilization**	**Dynamic Energy** (kJ)	**Area** $(LUTs)$
Fig. 2.1(a)	306.70	0.92	336.12	784
Fig. 2.1(b)	170.39	0.70	806.66	2406
Fig. 2.1(c)	262.14	0.60	747.30	1621
Fig. 2.1(d)	222.82	0.70	519.49	5600
Fig. 2.1(e)	176.94	0.53	588.71	3686
Fig. 2.1(f)	271.97	0.76	618.52	2234

Notes: the comparison was done using AXI-based interconnects on a Xilinx Zynq™-7000 [7] FPGA.

2.3.1.1 Transport latency

Transport latency is defined as the time (in clock cycles) between the occurrence of a transaction request from a master to a slave and the moment when the response is received. In the remainder of this chapter, we will simply refer to this time interval as latency. When a transaction occurs from a master node v_m to a slave node v_s, it goes through a path $p \in p_{m,s}$, consisting of a set of bridges B and local domains D. The latency, denoted by $L(p)$, is given by

$$L(p) = \sum_{b \in B} W_b + \sum_{d \in D} \left(W_{arbit} + \left\lceil \frac{v(e_{m,s})}{Bw} \right\rceil \right) \tag{2.1}$$

where W_b is the latency necessary to go through bridge b, W_{arbit} is the arbitration latency needed to grant a request, $v(e_{m,s})$ and Bw are respectively the communication volume between vertices v_m and v_s and the channel bandwidth of domain d. Pipelined interconnects incur, furthermore, additional delay cycles. Their use must thus be limited when we aim at minimizing the latency cost [164]. In case of intra-domain communication, since $B = \emptyset$ and D contains only a single bus or crossbar, Equation 2.1 can be simplified. In many cases, the interconnect contains a single centralized arbiter that sequentially grants a request at a time, while data transmission can be parallelized. Consequently, the arbitration latency W_{arbit} grows as the number of crossbar ports is increased [59]. In addition, the arbitration latency includes the time overhead caused by congestion. The impact of congestion depends on: 1) the total amount of pending requests, 2) the priorities of these requests, 3) the volume of the requests, and 4) the arbitration policy. This additional overhead W^c_{arbit}, sometimes called congestion-incurred latency [63], can indeed be modeled by an exponential dependence on the total amount of data traffic through the interconnect, as follows [60, 63, 143]:

$$W^c_{arbit} \propto e^{\alpha_{m,s}+\beta_{m,s}} \tag{2.2}$$

where $\alpha_{m,s}$ denotes the base traffic utilization from master port m to slave port s and $\beta_{m,s}$ denotes the interfering traffic utilization from the other master ports to slave port s. $\alpha_{m,s}$ and $\beta_{m,s}$ are defined as follows:

$$\alpha_{m,s} = \frac{v_{m,s}}{f \times width} \qquad \beta_{m,s} = \frac{\sum\limits_{l=1,l\neq m}^{n} v_{l,s}}{f \times width} \tag{2.3}$$

where $v_{m,s}$ is the communication volume between master port m and slave port s, f is the clock frequency, $width$ is the data bus width, and n is the total number of master ports of the interconnect. In order to reduce W_{arbit}, some architectures can perform bus transactions without waiting for the bus access to be granted by the arbiter [90].

According to the transaction type, we can distinguish write and read latencies. In case of read latency, a master needs to wait for the read data to be received from a slave after generating the read address. Therefore, read latency is defined as the time from the generation of the address to the arrival of the data. Instead, write latency does not consider the time necessary for a slave to complete the transaction, as it only refers to the time between the generation of the address to the acknowledgement. Frequently, the latency L is defined as the average of read and write transaction latencies:

$$L = \frac{\sum l_r + \sum l_w}{n_r + n_w} \tag{2.4}$$

where $\sum l_r$ and $\sum l_w$ are respectively the sums of all read and write transaction latencies, while n_r and n_w are respectively the total number of read and write transactions.

Since evaluating accurately the latency during the design space exploration is a non-trivial task, the hop count is commonly used to approximate the latency, as it is deemed proportional to the number of bridges in a path or to the cascading depth [41, 67, 68]. Obviously, this metric ignores the arbitration delays and network contention. Reducing the amount of overlap of traffic is the main approach to minimizing the latency [63, 103, 105]. As an example, in [30, 32], the general Equation 2.1 is evaluated for a design implemented on a Xilinx Zynq™-7000 [7] chip in case of burst traffic with no congestion, using custom AXI-based masters, slaves, and domains with $d_w = 32$ bits, $b_l = 16$ beats[1] and $Bw = 32$ bits/clock cycle, where d_w, b_l, and Bw are respectively the channel data width, the burst length, and the channel bandwidth. Results show that $W_{arbit} = 4\left(\frac{v(e_{m,s})}{d_w \cdot b_l}\right)$, $W_{slave} = 3$, and $W_b = 10\left(\frac{v(e_{m,s})}{d_w \cdot b_l}\right)$, where the burst length refers to the number of transactions in a single burst. As a consequence

$$L(p) = \left(\frac{52 + 10h_p}{64}\right) v(e_{m,s}) \tag{2.5}$$

where h_p is the hop count of the path $p \in p_{m,s}$. Based on this equation, Table 2.2 provides the total latency for the CG in Figure 2.3(a) and the different topologies presented in Figure 2.1. As expected, the hierarchical bus exhibits the worst latency while the single crossbar is the best solution since it is intrinsically a zero-hop architecture. The crossbar-based architecture with slaves and masters grouped in shared buses is highly constrained by the shared buses, leading to a 35% overhead. On the contrary, the cascaded crossbar introduces some latency cycles proportional to the cascaded depth. Since the cascaded crossbar does not use bridges, this additional latency (23%) is due to register slice crossing. Finally, the two clustered bus and crossbar architectures exhibit respectively a 4% and a 37% overhead compared to the single crossbar. The difference depends on the different implementation of the local domain C_1.

2.3.1.2 Bandwidth

The bandwidth of an interconnect represents the amount of data transfer per unit time, while the *Bandwidth Utilization* (BwU) is defined as the percentage of available ideal bandwidth being used to actually transfer data, given by

$$BwU = \frac{Bw_{used}}{Bw_{ideal}} \tag{2.6}$$

where Bw_{used} and Bw_{ideal} are, respectively, the actually used bandwidth and the available ideal bandwidth. A higher value means that more data are transferred within a period of

[1]A beat is an individual data transfer within an AXI burst [8].

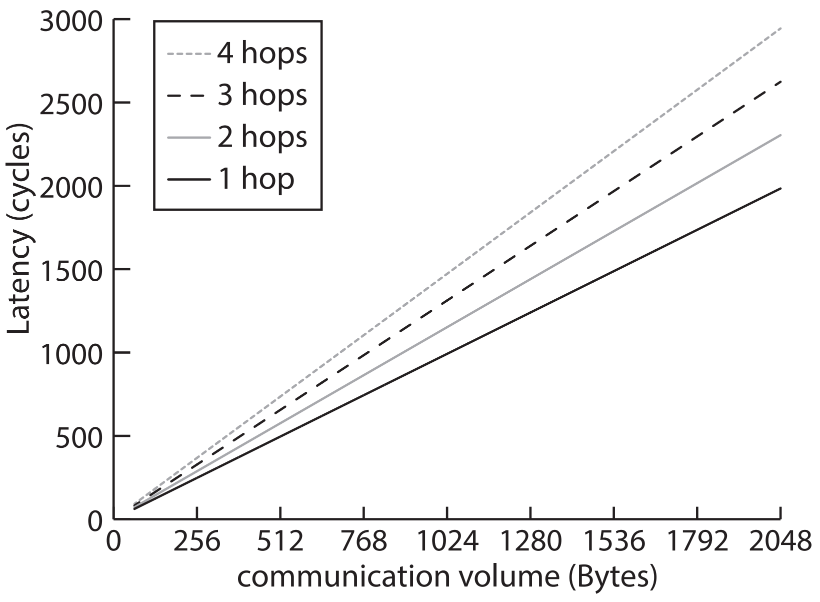


Fig. 2.4 The latency for varying message sizes and crossed domains.

time implying a shorter transaction latency as well as more efficient utilization of the interconnect. On the contrary, a waste of bandwidth means that the communication load on the interconnect is not balanced, which suggests reducing communication overlapping to obtain better performance. Notice that the Bandwidth Utilization is strictly dependent on the arbitration scheme. As an example, TDMA-based architectures can provide bandwidth guarantees for each component by appropriately assigning slots in the timing wheel as well as proportional bandwidth allocation. The total bandwidth of a domain d, e.g. a bus or a crossbar within an interconnect is given by the sum of the single bandwidths provided by each channel. The channel bandwidth in turn is closely related to its operating frequency $f(d)$ and data width $d_w(d)$. The ideal bandwidth of an interconnect $d \in Sb \cup C$ is given by $Bw_{ideal}(d) = d_w(d) \cdot f(d)$. As an example, a shared bus with $Dw(d) = 32$ bits operating at 100 MHz provides an ideal total bandwidth of 800 MB/s with the read and write bandwidths being 400 MB/s each. Table 2.2 provides the Bandwidth Utilization of the AXI-based implementation of the topologies in Figure 2.1 with $d_w = 32$ bits.

2.3.1.3 Power and energy consumption

As technology feature sizes are increasingly reduced, energy efficiency of devices is not scaling with the integration capacity [50]. In the context of MPSoC design, future devices are likely to be seriously power limited. For instance, at 22 nm, 21% of a fixed-size chip must be powered off, while at 8 nm, this rate grows to more than 50% [44]. The energy consumption due to the communication facilities has been often ignored in the past due to the limited contribution of the wires compared to the logic. Instead, in current and future devices, the communication architecture is gradually becoming one of the primary energy bottlenecks, contributing up to 50% of the total energy consumption in an integrated circuit.

When a transaction t occurs from a master node v_m to a slave node v_s, it goes through a path $p \in p_{m,s}$, consisting of a set of bridges B and domains D. The power (denoted by $Pw(t)$) is given by

$$Pw(t) = \sum_{b \in B} Pw(b) + \sum_{d \in D} Pw(d) \tag{2.7}$$

where $Pw(d)$ is the energy consumed in domain d, and $Pw(b)$ is the energy consumed in bridge b. The total power consumption of an interconnect is the sum of different contributions due to dynamic effects and leakage. The former are caused by the charging and discharging of wire capacitances. The latter is caused by the subthreshold leakage current and reverse-biased diode junction leakage current. To quantify the dynamic power consumption we can rely on the following relationship [160]

$$Pw_{dyn}(t) = \frac{1}{2} n_{trans} C f V_{dd}^2 \tag{2.8}$$

where n_{trans}, C, f, V_{dd} are respectively the average toggle rate (switching activity), the total capacitance of the interconnect, the frequency of the interconnect, the supply voltage. The frequency f can be expressed as

$$f = \frac{(V_{dd} - V_{th})^{\alpha}}{k V_{dd}} \tag{2.9}$$

where V_{th} is the threshold voltage, k is a constant for a given technology process, and $1 < \alpha \leq 2$. [164] and [67] rely on a more detailed model describing the dynamic power consumption as two components. The first is proportional to the signal switching activity while the second is non-proportional (e.g. the power consumption of the clock tree). On the other hand, the leakage power consumption can be expressed as [160]

$$Pw_{leak}(t) = V_{dd} I_{sub} + |V_{bs}| (I_j + I_b) \tag{2.10}$$

where I_{sub}, I_j, I_b,V_{bs} are respectively the subthreshold leakage current, the drain-body junction leakage current, the source-body junction leakage current, the body bias voltage. I_{sub} is given by

$$I_{sub} = I_s \left(\frac{w}{l}\right) e^{\frac{-V_{th}}{nV_T}} \tag{2.11}$$

where I_s and n are technology parameters, w and l are device geometries, and V_T is the thermal voltage. Similar models can be found in [62, 63, 67, 118, 164]. Table 2.2 provides the dynamic energy consumption of the CG in Figure 2.3(a) on a AXI based implementation of the communication architectures presented in Figure 2.1. The data were collected using the XPower [6] tool from Xilinx and refer to the Zynq-7000 family [7]. In terms of power consumption there is a big gap between shared buses and crossbars: as an example, a 4×4 crossbar consumes about $4.2\times$ more power than a shared bus of the same size. However, when evaluating the energy dissipation this difference is mitigated by the worst performance of shared buses. Moreover, according to Equation (2.8), dynamic energy depends on the total capacitance of the interconnect. Since this value is strictly dependent on the size of the interconnect, larger crossbars and buses consume more power than smaller interconnects: as an example a 4×4 crossbar consumes about twice the dynamic power of a 4×4 bus.

Because of the importance of the energetic aspects, a large part of the available design methodologies aim to minimize the energy consumption of the communication architecture. One way of achieving low power consumption consists in providing different frequencies to the various interconnect domains. This creates an opportunity of sharply cutting down on the power consumption of low bandwidth modules clustered together, by reducing the frequency of the bus they are connected to [141]. [57, 103] address the observation that custom crossbars reduce the power consumption when compared with general-purpose crossbars. [63] tries to reduce the power consumption of cascaded crossbar interconnects by exploiting the locality principle in order to transmit data traffic with high bandwidth requirements through as few crossbar switches as possible. [164] and [67] propose a model for the power consumed by a single port of an interconnect. Then, the area/power trade-off of crossbars with a different amount of ports is analyzed by means of the power consumed by its ports. In addition, [67] tries to avoid the pipeline insertions since they are responsible for a significant portion of the whole energy consumed. Some works deal with the concurrent minimization of both the energy consumption due to the communication and the memory [62, 119, 140]. These approaches explore the trade-off between using an increasing amount of memory blocks and the resulting communication complexity. Refers to Section 2.4.1 for a detailed analysis. A further optimization consists in adding switches on each channel so that the activity propagates only to the necessary channels, thus decreasing the capacitive load of the interconnect [119]. In addition, the energy consumption can be reduced by shutting down the unused channels via switches [159]. Differently, it is possible to use the dynamic supply voltage scaling (DVS) technique to save energy [118]. This technique requires a driver, which initiates the data transfer, capable to scale the voltage of each communication task dynamically. Each voltage conversion from V_i to V_j comes at an energy and latency cost, which are respectively $E_{conv} = C_r(V_i - V_j)^2$, where C_r is the capacitance of the power rail and $L_{conv} = \alpha \left|V_i - V_j\right|$ where α is a constant [95]. Clearly, the saving in terms of energy reduction must overcome the switching overhead.

2.3.1.4 Area cost

To evaluate the feasibility of an interconnect architecture, its silicon area requirements must be carefully taken into account. It is thus of paramount importance to determine the amount of relative area a potential interconnect solution consumes. The area cost of a shared bus grows linearly with the number of its ports, while in case of a crossbar the cost grows quadratically. While the area of an ASIC interconnect is normally evaluated in terms of mm^2, in case of FPGAs, interconnect components are usually dominated by the number of Look-Up Tables (LUTs). When considering storage elements such as queues or register slices, it is also necessary to take into account the number of Flip-Flops (FFs). As an example, the area cost of a 4×4 crossbar for ASICs and FPGAs is $0.28mm^2$ and 2406 LUTs, respectively, for a 65nm ASIC process and a Xilinx Zynq 7020 FPGA technology. In [30, 32] the following equations were obtained for calculating the area cost in terms of LUTs for $n \times m$ AXI interconnect IP cores by Xilinx on the Zynq™-7000 [7] FPGA chip.

$$A_{crossbar}\; n \times m = 101n + 60nm + 42m + 874 \tag{2.12}$$

$$A_{bus}\; n \times m = 80n + 18.75m + 95.5 \tag{2.13}$$

Notice that the above equations provide only a baseline cost, because the cost may grow in case of use of advanced features. As an example, a register slice for a single channel requires around 50 LUTs.

Table 2.2 provides the area requirements of the AXI-based implementation of the topologies in Figure 2.1 with $d_w = 32$ bits based on the above equations.

2.3.1.5 Wire delay and frequency

Wire delay is defined as the amount of time (in seconds) it takes for the head of the signal to travel from the sender to the receiver. Note that, unlike latency, the wire delay depends only on physical parameters, i.e. the wire length and the wave propagation speed. The importance of the wire delay is related to its linear dependence on the operating clock frequency: large delays reduce the frequency and hence the achievable performance. Obviously, in case of interconnect components, the wire delay depends on the total number of ports of

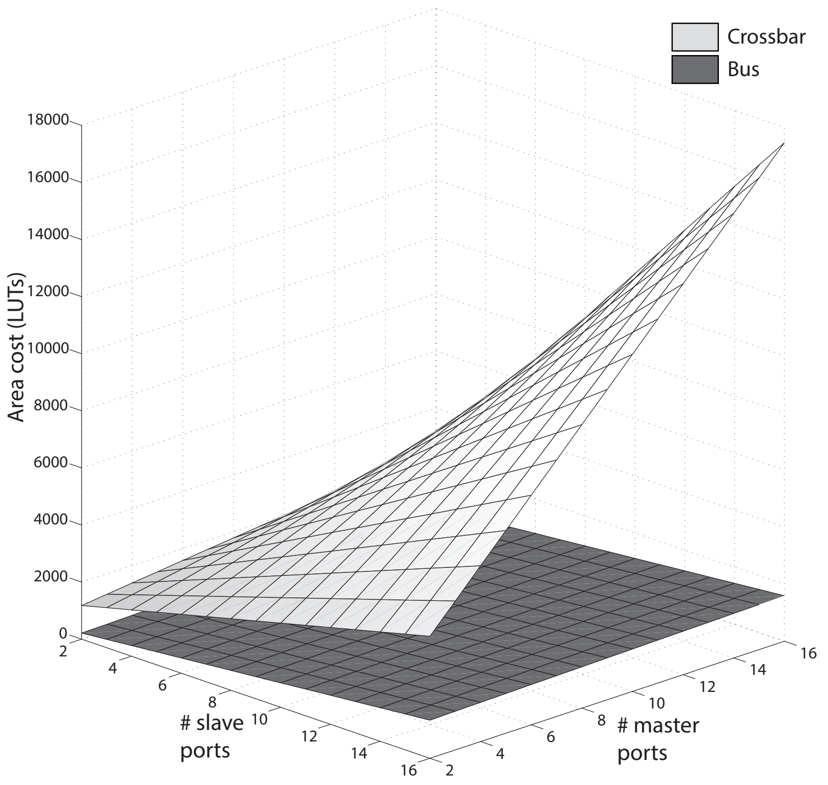


Fig. 2.5 The area cost for bus and crossbar interconnect of varying port numbers.

the interconnect, which is one of the reasons motivating the adoption of cascaded crossbar topologies. As an example, in a Samsung 90nm low-voltage ASIC process, the maximum wire delay in the 4×4 full crossbar of Figure 2.1(a) is 4 ns, while each of the small 2×2 crossbars of Figure 2.1(d) incurs a delay of 3.3 ns.

2.3.2 Evaluation approaches

Any design exploration methodology must rely on a suitable evaluation procedure to be invoked whenever we consider a potential interconnect solution and need to estimate its costs and performance levels. This is typically performed at the electronic system level (ESL) before the implementation (or register transfer level, RTL) model is created. The RTL modeling abstraction has two main drawbacks: it is prohibitively time consuming for a large design space and a considerable effort is required for making changes due to the amount and complexity of the design details [120]. Raising the modeling abstraction level is crucial to overcome the RTL limitations. In this way it is possible to efficiently face an extremely large design space allowing the evaluation of each design point in a much shorter span of time. Obviously, speed and accuracy are key to an efficient evaluation methodology. Since communication is dynamic and unpredictable, simulation-based methods are able to provide an accurate estimation at a non-negligible computational cost. Simulation requires models of the components and their communication at different levels of abstraction. Different communication abstractions provide a further trade-off between simulation time and accuracy. However, these techniques still require the simulation of the whole system. As a result, simulative methods explore only a few design options and cannot be used for exploring large design spaces. On the other hand, analytic approaches take advantage of static models that capture the system performance and cost as a function of certain parameters. They are usually overly pessimistic or ignore dynamic effects such as bus contention, but are several order of magnitude faster than simulative approaches. A third class of estimation techniques take advantage of both the above approaches to speed up communication architecture performance estimation while generating accurate results. A review of analytical and simulative evaluation methods is presented respectively in Section 2.3.2.1 and Section 2.3.2.2, while Section 2.3.2.3 discusses hybrid models that jointly use static estimation and dynamic simulation.

2.3.2.1 Analytical models

A number of methodologies look at the exact quantitative evaluation of the interconnect cost and performance. Several works [30, 31, 63, 65, 67–69, 103, 163] use a characterization technique that involves the RTL code generation and synthesis of different components (crossbars, buses, bridges, etc.). Then, area/timing/power information are obtained from the RTL synthesis results. This is computationally feasible for the characterization of a small set of components, since hundreds of synthesis can still be parallelized and carried out in a reasonable amount of time with a high-performance machine. As the evaluation space grows, however, this approach becomes impractical and it is necessary to limit the synthesis to a subset of the whole exploration space. Based on the achieved values, analytical models can be built using regression analysis techniques [103]. Different levels of accuracy are reached by exploring the trade-off between accuracy and number of synthesis. Since any configuration of an interconnect can be assembled with a set of input and output ports and the corresponding connections, most of these models provide area or power costs as a function of the number of the input/output interconnect ports. Likewise, memory-aware methodologies also require performance and cost estimation of the memory architecture. The majority of the approaches relies on well-known models such as the CACTI model [83, 106] that include information on integrated cache and memory access time, cycle time, and area as well as leakage and dynamic power.

Since communication patterns are dynamic, it is challenging to statically estimate the performance in an accurate way. Latency is often evaluated using the hop-count concept [41, 67–69], which often turns out to be far from a precise estimate. In order to obtain more precise data, [30, 31] analyze the time necessary to go through each communication components and derive a cycle-accurate model for crossing each path in an interconnect architecture. Obviously, these models are dependent both on the used IP cores and arbitration policies and still ignore contention. On the other hand, a time-division multiple-access (TDMA) communication system makes the interconnect subsystem fully predictable and, hence, it simplifies the performance estimation since it provides channel throughput guarantees [62].

[41, 118] add statistical methods to improve the estimation accuracy. [41] estimates contention making a few simplifying assumptions: a communication task is scheduled according to a uniform distribution between its *as soon as possible* (ASAP) and *as late as possible* (ALAP) schedule times. Then, contention between two tasks depends on their schedule overlapping. [118] assumes that the volume can be modeled for each communication task as a random variable with a known probability distribution, and the arrival rate and inter-task arrival time as a Poisson distribution. Furthermore, the statistical parameters of voltage are estimated using an analytical method and the above probability distribution function. Unlike the previous works, [74] proposes a static performance-estimation technique based on a queuing analysis assuming that the memory traces and the schedule information are given. In this model there are N processing elements PE_i, $0 \leq i < N$, that are regarded as customers, competing for the use of a single bus that is regarded as a server. The model aims at estimating the congestion-incurred latency for each processing element by building a steady-state transition diagram. Each processing element issues memory requests with a given request rate λ_i, computed as the ratio between the memory access count and the scheduled latency in absence of contention. On the contrary, the bus access rate, considering contention, is denoted by θ_i. The mean service rate of a server for the request from PE_i is denoted by μ_i and its mean service time is the inverse of the service rate, i.e., $1/\mu_i$. Then

$$\theta_i = (1 - k_i - u_i)\lambda_i \tag{2.14}$$

where k_i is the expected number of requests waiting for the bus and $u_i = \theta_i/\mu_i$ is the bus utilization factor. Finally, the congestion-incurred latency of the stalled request r is calculated as

$$W^c_{arbit}(r) = k_i/\theta_i \tag{2.15}$$

This model assumes that the arrival process of the communication tasks can be described as a Poisson process. This assumption is actually common in network performance analysis [90, 111, 117]. However, while this may be realistic in case of non-bursty traffic, typical real-world scenarios are characterized by bursty traffic which is better described by self-similar processes [146].

2.3.2.2 Simulation-based approaches

Simulation-based methodologies capture the communication architecture behavior at different levels of abstraction. Typically, if a model considers more properties, it is more accurate in estimating the performance, but also slower to simulate. Usually an ESL simulation uses high-level languages, such as SystemC [84], that are at least an order of magnitude faster than the hardware description language (i.e. Verilog or VHDL) used during RTL simulations.

[124] uses a technique proposed in [121] consisting in simulating the whole system using a SystemC description of the components with a fast transaction-based, bus cycle-accurate modeling abstraction. [105] and [103] simulate the traffic patterns in windows: the entire simulation period is divided into a number of fixed-sized windows; within each window, the application is simulated in order to verify the communication requirements. Unlike the previous works, [98] first generates a dependency graph using the partially ordered sequences of all memory accesses derived from the memory access traces. Then, latency information are calculated by performing discrete event simulation[2]. In order to simulate the whole system on both its hardware and software levels, hardware-software co-simulation was introduced supporting the SystemC simulation in combination with instruction set simulators (ISSs) or software emulators [154]. [130] includes in the solution space exploration such a technique. To improve the speed of the co-simulation with multiple ISS instances, [162] proposes trace-driven virtual synchronization, which greatly reduces the synchronization overhead between ISS instances. This technique is used in some hybrid approaches. Another category of performance analysis techniques that accounts for on-chip communication includes trace-based techniques.

Concerning power consumption, since dynamic power depends on the switching activity, several works [32, 57, 63, 107, 125] rely on gate-level simulation using ad-hoc tools such as XPower [6] by Xilinx® or PrimeTime [5] by Synopsys®.

[2]A discrete-event simulation models the operation of a system as a discrete sequence of events. Each event takes place in a certain time instant and causes a change of state in the system. Between consecutive events, no change in the system is assumed to occur and hence the simulation can directly jump from one event to the next [12].

2.3.2.3 Hybrid methodologies

In order to exploit the strengths of both analytical and simulative techniques, heterogeneous approaches, that jointly perform static and dynamic analysis, have been proposed. The design methodologies presented in [73, 81] break down the estimation procedure in two steps. In the first step, the static performance estimation technique based on the queuing model [74] is used to quickly perform an exhaustive evaluation and prune the evaluation space drastically. The second step then uses the trace-driven simulation presented in [162] to accurately evaluate each design point in the reduced design space. On the other hand, [78] uses the approach proposed in [77] where some static analysis is used to group the traces and a trace-driven simulation is applied with the trace groups. The aim is to reduce the time complexity of trace-driven simulation via an early static analysis. Similar to [74], the methodology in [65] consists in pruning the design space efficiently by two static analysis techniques reducing the use of simulation. The two analytical techniques are bandwidth analysis and memory contention analysis. The purpose of bandwidth analysis is to compute the lower bound of memory bandwidth requirements considering the task scheduling information, while memory contention analysis aims to estimate the average latency of the memory accesses while considering resource contention. Any architecture candidates that fail to meet the bandwidth and latency requirements are excluded from the subsequent time-consuming cycle-level simulation. Finally, several works [122, 123, 125] perform static analysis and rank the results from the best case solution to the worst. Then, starting from the best case solution, the simulative technique proposed in [121] is applied to each design point until a solution that meets all the constraints is found.

2.4 Synthesizing an Interconnect

The synthesis flow aims to generate an optimal communication architecture, formally described by $Arch(T(Sb,C,B),\Omega(V))$, for a certain application, while satisfying given design constraints. It is necessary to define both the topology $T(Sb,C,B)$ and the mapping $\Omega(V)$ of the cores it contains. In that respect, every interconnect core must be characterized, the position of each bridge must be defined, and the PEs and MEs must be connected to the associated interconnect domains. To generate the optimal architecture configuration, the synthesis flow takes the application requirements and constraints given in the input CG as well as an analysis methodology necessary to evaluate the costs and performance levels of the resulting communication architecture. The synthesis process becomes more challenging in case of multi-dimensional design constraints such as performance, power, and area cost. These constraints are often conflicting. For instance, large data widths and high frequencies increase the bandwidth and hence the performance, but also the power consumption and area cost. It is therefore essential to optimize the interconnect while balancing different and contrasting goals.

Since the manual exploration of a potentially huge design space is infeasible even for small systems, there is a number of algorithmic approaches that can be used to automate the design space exploration. Many types of heuristics, iterative improvement, probabilistic, integer linear and nonlinear programming approaches, and other types of algorithms can be adopted for this task. Mathematical programming is widely used in the area of operation research, electrical engineering, control engineering, etc. Its main goal is to reach the globally optimal solution for an objective function $f(x_0,x_1,...,x_{n-1})$ under a set of m constraints $c_j(x_0,x_1,...,x_{n-1}) \leq k_j$ with $0 \leq j \leq m-1$ and variable bounds $a_i \leq x_i \leq b_i$ with $0 \leq i \leq n-1$. As an example, x_i can be a decision variable indicating whether a master or a slave is connected to an interconnect domain or indicating whether an interconnect is a bus or a crossbar. The objective function can be a design metric such as the power consumption of the whole communication architecture or a combination of different metrics such as a linear and weighted combination of power and frequency. The constraints can belong to different classes: some are related to the design metrics such as a latency constraint or a bandwidth constraint; some are related to the architecture definition, for instance a master or a slave must be connected to one and only one crossbar or the number of masters connected to a crossbar must be greater than two; some are optional constraints determining additional features such as the maximum number of paths between a master-slave pair.

When the objective function and the constraints are linear functions of the variables, then the problem is called *linear programming* (LP) as opposed to *nonlinear programming* (NLP). In case all x_i are binary integer variables, a LP problem is called *integer linear programming* (ILP). If only a subset of them are binary integer variables, the problem is called *mixed integer linear programming* (MILP). Due to the availability of various solution methods, many approaches [62, 63, 68, 105] employ MILP to deal with the problem of communication architecture synthesis. However, the MILP approach can be applied to relatively small problems since it does not scale well. Indeed, its complexity grows exponentially with the problem size, i.e., the number of masters and slaves in the interconnect [69]. In addition, when the objective function or some constraints are defined according to some probabilistic statement, the problem is called *stochastic linear/nonlinear programming*. [118] model the synthesis problem as stochastic NLP. In particular, two variables indicating respectively the number of data bits to transfer and the duration of the transaction are handled as normally distributed variables, while the voltage is statistically estimated based on the above variables. Solving a stochastic NLP problem is known to be an NP-complete problem. However, [118] relies on an efficient convex nonlinear optimization algorithm, proposed in [108], able to solve the problem with a polynomial time complexity.

Unlike the contributions above, several works in the literature adopt fast and efficient heuristics, that normally scale better for larger problems. Numerous heuristic approaches are based on *simulated annealing* (SA) [41, 98, 144, 163], *genetic algorithms* (GAs) [11, 140, 164], *quantum-inspired evolutionary algorithms* (QEAs) [65], *first-fit* heuristics for bin packing [103], *greedy-based* approaches [62, 78], or custom methodologies [69, 119]. SA and GA techniques are capable of both broad search (exploration) and local search (exploitation) of a search space. They are often preferred over gradient search methods because they avoid local minima, and do not require a smooth search space. The main problem of implementing either the simulated annealing or the genetic algorithm flows lies in defining a suitable encoding of the states and implementing a valid transition function. In fact, since an arbitrarily chosen interconnect is not necessarily a feasible point in the design space, applying random transformations may not work. In [163, 164] the problem is addressed using a representation technique called traffic group encoding (TGE). Starting from a CG, a Traffic Group (TG) is defined as an unordered set of edges from the CG, TG $\subset$ E, while a TGE is defined as an ordered set of traffic groups.

$$TGE = \{TG_i | TG_i \subset E, 1 \leq i \leq n\} \tag{2.16}$$

Traffic groups correspond to interconnect elements, such as shared buses or crossbars, while their order represents the direction of traffic flow, and hence the position of bridges and wires. For example, the topology of Figure 2.1(d) is generated by the following TGE:

$$\begin{gathered} TGE = \{(M_0 \to S_0), (M_0 \to S_1), (M_1 \to S_0), (M_1 \to S_1), (M_1 \to S_3)\}, \\ \{(M_2 \to S_1), (M_2 \to S_2), (M_2 \to S_3), (M_3 \to S_3)\}, \\ \{(M_0 \to S_0), (M_0 \to S_1), (M_1 \to S_0), (M_1 \to S_1), (M_2 \to S_1)\}, \\ \{(M_2 \to S_2), (M_1 \to S_3), (M_2 \to S_3), (M_3 \to S_3)\} \end{gathered}$$

The main property of this encoding method is that a TGE always leads to an

$Arch(T(Sb,C,B,L),\Omega(V))$ that satisfies the CG.

QEA [52] inherits the main features of evolutionary algorithms but uses quantum bits (Q-bits) to stochastically represent individuals instead of using the binary representation of genes as in a genetic algorithm. In addition, a few approaches use branch-and-bound techniques to break down the complexity of the problem [107, 122, 123, 125].

Although communication interconnect synthesis is a stand-alone process, additional opportunities lie in combining it with different design steps. Co-synthesizing the memory and the communication architecture, embodying the communication task scheduling in the problem, and jointly floorplanning and synthesizing the interconnect may lead to improved customized solutions.

2.4.1 Co-synthesis of memory and interconnect architectures

The memory architecture dictates most of the traffic in a MPSoC, which in turn influences the performance of the interconnect architecture. Designing a highly parallel architecture without taking care of the on-chip memory architecture can in fact easily result in degraded performance. The task of synthesizing a memory architecture consists in memory partitioning, configuration of memory modules, and mapping of data to modules. Often, the memory architecture and the communication subsystem are defined sequentially as separate steps [35, 155]. While greatly reducing the complexity of the design space exploration, such an approach can lead to suboptimal solutions and miss good design points [76]. There are, however, a few methodologies that simultaneously consider the communication architecture and the on-chip memory architecture. Notice that both the power efficiency and the area cost of a design are deeply affected by the selection of the memory architecture and the interconnect infrastructure [71, 96, 114]. The solutions for the co-synthesis of memory and interconnect architectures target one of these two aspects. In particular, in order to generate a power-efficient configuration, it is necessary to implement the memory architecture via small distributed memory modules since the distributed memory paradigm ensures less energy consumption than a centralized structure [15, 26]. Confining the majority of the accesses to smaller size memory blocks reduces the energy per access of each data element. At the same time, the integration of a large number of memory modules on a SoC increments the logic for communication handling and bank address decoding. This tradeoff limits the amount of the memory partitions and their sizes. Some approaches merge the topology selection with the partitioning and mapping of memory to the interconnect during synthesis [119, 140]: the required parallelism in memory accesses is translated into a minimum number of memory modules that the architecture must have along with a minimum number of buses needed to access the memory blocks. In addition, the memory blocks are mapped in the topology by taking advantage of spatial locality, i.e. mapping memory blocks to physical locations on the chip that are close to the cores that access them, while reducing the access conflicts. Communication locality reduces the average distance traveled by communication tasks, which minimizes power and increases performance [18]. A further optimization consists in replacing on-chip caches with scratch pad memories (SPMs) [62]. SPMs are more power efficient than caches because of the absence of tag memory, tag comparators, and hardware that enforce cache coherence. It is only necessary to add DMA engines so as to prefetch the data on these SPMs. The above approaches rely on exploiting the finer grain control due to the mapping of smaller memory chunks to different bus trunks at the bus partitioning stage in order to cluster frequently accessed data onto a smaller memory. On the contrary, the methodology presented in [122] relies on minimizing the area cost by applying a few operations such as merging several memory blocks into one, changing a memory from shared to private, or adding out-of-order (OO) buffers in order to reduce the number of ports of the memory modules. In addition, reducing the amount of memory blocks leads to a smaller communication architecture, which saves further area resources. Clearly, all the above techniques require the application to have regular memory access patterns that can be statically analyzed and predicted at compile time.

2.4.2 Task scheduling and interconnect synthesis

The scheduling of a dependency graph determines for each communication task $v_i \in V_t$ its precise start time $s(v_i)$. The *latency* of the schedule is the time required to execute the whole schedule or, equivalently, the difference between the completion time of the sink vertex and the start time of the source. If two or more sources/sinks are available, the source/sink with the smallest/largest start time is considered. Since a feasible solution must satisfy dependency constraints, the start time of a communication task is at least as large as the start time of each of its predecessors plus their *execution delays* $d(v_i)$. An execution delay represents the time required to execute a communication task in a given architecture. Therefore a schedule must satisfy the following relations:

$$s(v_i) \geq s(v_j) + d(v_j) \;\; \forall v_i, v_j : e_{v_i,v_j} \in E \tag{2.17}$$

Two noticeable scheduling policies are the as soon as possible, or ASAP, and the as late as possible, or ALAP, policies enforcing respectively the least and the maximum start time allowed by the dependencies. The difference between the ALAP and ASAP times of the same communication task is called slack. It measures the degree of freedom available for scheduling communication tasks. The problem of communication scheduling and interconnect synthesis are deeply interrelated. The schedule determines the precise start time of each communication task. The start times must satisfy the DG dependencies, which limits the amount of parallelism of the communication, because any pair of communication tasks involved in a direct dependency, or a chain of dependencies, may not execute concurrently. Scheduling determines the concurrency of the resulting interconnect implementation, and therefore it affects its performance and its cost. Similarly, the maximum number of concurrent communication tasks at any step of the schedule is bounded by the interconnection topology. The final architecture must be able of handling the parallelism owned by a certain schedule. The traditional scheduling problem is known to be NP-complete [47]. The problem of scheduling communication tasks onto a communication architecture is more complicated because the task execution delays are dependent on the topology in terms of number of hops.

Based on these observations some works [65, 73, 81, 119] make use of the schedule information known a priori in order to better tailor the interconnect on the application characteristics. [119] extracts the interconnect access pattern from the schedule and uses this information to determine several conflict free-communication paths. On the contrary, [65, 73, 81] take advantage of a schedule-aware performance estimation technique presented in [74] in order to improve the accuracy of the performance estimation. This approach consists in splitting the schedule into several constant time slots according to the access pattern. Then, for each time slot, a queueing analysis is performed. In addition, in [65] a lower bandwidth bound for each communication channel is obtained via a bandwidth analysis that uses the static scheduling information. Differently, [118] exploits the slack to reduce energy by simultaneously scaling the voltage during communication tasks with a non-zero slack. The basic idea is to introduce a bus driver that is capable to scale the voltage in order to reduce the energy consumption while spreading the task in more time slots. Notice that the execution delay of a task t depends on the supply voltage V according to [95]

$$d(t) = k\frac{V}{(V - V_{th})^{\alpha}} \tag{2.18}$$

where V_{th} is the threshold voltage, k is a technology dependent constant, and $1.4 < \alpha \leq 2$ is the saturation velocity.

Although the above approaches are schedule-aware, they do not implement joint scheduling and interconnect synthesis. On the contrary, [11, 31, 159] concurrently perform a scheduling phase during the interconnect synthesis in order to achieve different goals. [11] tries to reduce the hop count resulting in a lower latency and lower power. [159] explores the scheduling solution space to find a schedule that avoids bus contention and maximizes the amount of buses that can be switched off when unused in order to minimize the energy consumption. Finally, [31] uses a deterministic control flow in order to schedule all communication tasks in advance avoiding access conflicts, enabling the application of techniques for temporal merging [33] since communication tasks that do not happen at the same time can share the same resources. In addition, considering a static schedule may lead to further optimizations: less expensive communication architectures can be achieved by using buses without any arbitration scheme and statically avoiding access conflicts [48]. This technique can be extended to handle non-deterministic control flows with a hardware overhead to handle the required synchronization mechanisms.

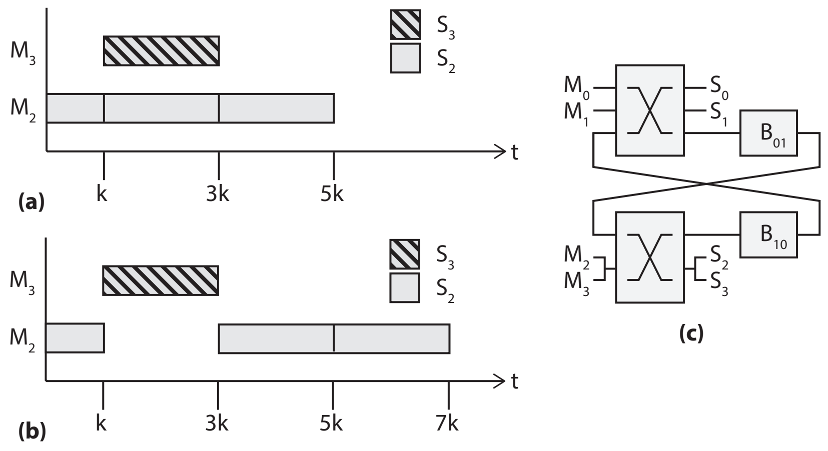


Fig. 2.6 The impact of joint scheduling and interconnect synthesis. (a) The ASAP schedule of the DG in Figure 2.3(b). (b) A different schedule with less concurrency for the same application. (c) The derived topology

The example in Figure 2.6 demonstrates some benefits enabled by joint scheduling and interconnect synthesis. Figure 2.6(a) depicts an ASAP schedule that is compatible with the topology in Figure 2.1(e). On the contrary, Figure 2.6(b) shows a different schedule that is compatible with the topology in Figure 2.6(c) without producing congestion, while the schedule (a) is incompatible with that topology. The advantage lies in the possibility of reducing the size of the crossbar, since tasks involving masters M_2 and M_3, and tasks involving slaves S_2 and S_3 will never overlap and hence they are compatible and can share the same crossbar port. Of course, this advantage comes at a latency cost since the schedule latency is increased by 2 time units.

2.4.3 Floorplanning and interconnect synthesis

Floorplanning is an essential step in electronic design. The usual floorplanning problem can be stated as follows [152]. Let $R = \{r_1, r_2, ..., r_n\}$ be a set of rectangular modules whose respective width and height are denoted by w_i and h_i, $1 \le i \le n$. The position of each module is identified by (x_i, y_i). A floorplan F is an assignment of (x_i, y_i) for each r_i, $1 \le i \le n$, such that there is no overlapping between any two different modules. The floorplan give us information about the area cost and wire length: the area is the bounding box of the design while the wire length can be calculated by using the half-perimeter of the minimum bounding box containing all terminals of a wire [24]. The goal of floorplanning in MPSoC design is to determine the physical location of the cores and communication components while optimizing predefined architecture features such as the total area cost (i.e., the minimum bounding rectangle of F) or wire length (i.e., the sum of all interconnection lengths). A detailed study of general floorplanning methods is of course beyond the scope of this survey. Refers to [138] for a review of floorplanning techniques. Concerning the communication architecture, the problem can be reduced to the following formulation: Given a communication architecture $Arch(T(Sb,C,B,L),\ \Omega(V))$ and the sizes of the cores mapped on it, find a set of physical locations which minimizes an objective function under a set of design/application constraints.

Traditional floorplanning tools, such as Parquet [9], are usually employed after the communication architecture is synthesized at the system level. A drawback of this approach consists in detecting timing violations on the wires only at floorplanning time. This can be mitigated by inserting register slices on the interconnect to meet cycle time constraints. However, such interconnect pipelining may not always be compatible with performance constraints (e.g. a latency). Including the floorplanning in an earlier step of the design methodology allows detecting and eliminating such violations as early as possible in the design flow at the system level [125]. It is even possible to drive the synthesis through a more accurate estimation of the performance and costs such as area, latency, and power [94]. Some approaches [41, 67, 98, 103, 124, 125, 164] address these observations by supporting the floorplanning step within the synthesis phase: the floorplanner is invoked whenever the communication architecture is updated in order to measure the feasibility of the resulting layout and its performance.

2.4.4 Exploiting multi-path communication

Some of the topologies that can be built with bus-based on-chip interconnects introduce the possibility of multiple concurrent paths across different local domains. In hierarchical bus topologies or cascaded crossbars as well as heterogeneous architectures, by statically setting bridge addresses, it is possible to split the traffic across two end-domains along different paths. Multiple paths introduce flexibility in the network topology, as they create a further opportunity for balancing the load across the interconnect, as well as concurrency in the inter-domain communications (inter-domain parallelism). As an example, Figure 2.7 demonstrates the benefits of exploiting multiple paths. Figure 2.7(a) shows some communication requirements of an example application, while Figure 2.7(c) depicts a possible topology. Depending on the bridge address ranges, two different paths can be used between clusters 0 and 2. For instance, the communication between M_{00} and S_{20} can go through B_{02}

and the communication between M_{01} and S_{21} can follow a multi-hop path through B_{01} and B_{12}. Figure 2.7(b) and Figure 2.7(d) contain two possible schedules obtained without and with exploiting multipaths. Bridge configurations enabling parallel communications lead here to an improvement in terms of communication overhead roughly equal to 33%.

In general, multiple paths can improve the performance of the communication between: 1) One master and one slave in different domains (one-to-one), 2) One master and two or more slaves in different domains (one-to-many), 3) Two or more masters and one slave in different domains (many-to-one), and 4) Two or more masters and two or more slaves in different domains (many-to-many). The problem of deriving suitable multipaths given certain communication requirements can be formulated as a multi-commodity flow (MCF) problem [17, 55, 104]. In case of one-to-one communication, the traffic between a master $v_m \in V_m$ and a slave $v_s \in V_s$ (i.e., the edge $e_{m,s} \in E$) is treated as a flow of multi-commodity. Assuming there are n paths between v_m and v_s, it is possible to apply techniques of traffic splitting in order to split traffic across n paths and handle the traffic on each path as a single commodity flow with a given demand $d^i_{m,s} > 0, i = 1,2,...,n$. Regarding one-to-many, many-to-one, and many-to-many interactions, the communication between each master-slave pair (i.e., each edge $e_{m,s} \in E$) is also treated as a flow of single commodity with a given demand $d_{m,s} > 0$. Let $p_{m,s}$ be the set of all paths between v_m and v_s and let $\mathbf{p} := \cup_{m,s} p_{m,s}$, $f(p)$ denotes the amount of flow sent along path p for every $p \in \mathbf{p}$. Assume there are k commodities, the MCF formulation is then:

$$\text{Min}: \sum_{i=1}^{k} \sum_{p \in {p_{m,s}}^i} c(p) \cdot f(p) \tag{2.19}$$

$$\forall 1 \leq i \leq k : \sum_{p \in {p_{m,s}}^i} f(p) \geq d_{m,s} \tag{2.20}$$

$$\forall p \in \mathbf{p} : f(p) \geq 0 \tag{2.21}$$

$c(p)$ is the cost for communicating across a path p (e.g., in terms of latency, power consumption, et cetera). The objective is to minimize the whole cost for inter-domain communication, which is the sum of all the communication flows times the corresponding cost over all the paths (Equation (2.19)). Through constraint (2.20), we make sure that communication demand for each master-slave pair is satisfied. Additional application- or design-specific constraints (e.g. bandwidth, latency etc.) may be included too. Notice that two paths cannot share the same resources, such as a shared bus or a crossbar port, in order to be concurrent.

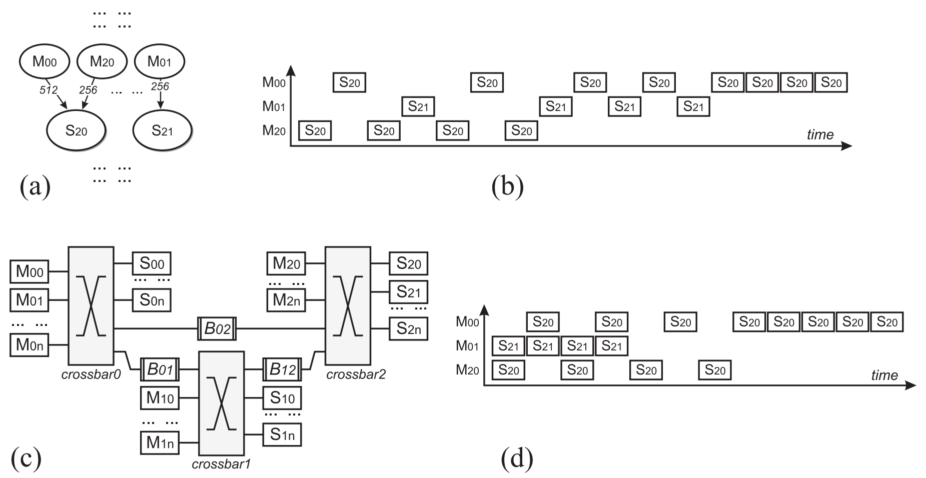


Fig. 2.7 The impact of multiple paths. (*M*: master, *S*: slave, *B*: bridge.) (a) Some communication requirements of an example application. (b) Schedule with no multiple paths. (c) The communication architecture implementation. (d) Schedule with multiple paths.

2.4.5 An FPGA-specific feature: dynamic configuration

Since the communication requirements between PEs can considerably change over time, performance can be enhanced and costs can be reduced by resorting to interconnects that dynamically fulfill the communication needs. Current FPGAs fully support partial reconfiguration, which enables them to reconfigure only a portion of their resources [93]. This operation is allowed at run-time: several modules are compiled and stored as bitstreams at synthesis time and are loaded at run-time. The latency overhead for the reconfiguration depends on the target technology and on the corresponding partial bitstream size. Techniques and methodologies to handle the partial reconfiguration are outside the scope of this chapter. We refer the reader to [87] for pointers to recent research and developments.

When dynamic reconfiguration is exploited, the reconfiguration overhead may be a bottleneck: the performance gain introduced by a communication architecture loaded at run-time must be greater than the reconfiguration overhead. Even for small bitstreams this overhead cannot be neglected. Moreover, dynamic reconfiguration requires additional hardware resources resulting in a trade-off between area cost and performance. [58] and [57] propose an FPGA-based design technique for a communication architecture based on the general concept of *on-demand* reconfigurable interconnect [147]. The aim of these approaches is to provide, for each given application, an interconnect that exhibits a physical topology perfectly matching the logical topology, instantly and dynamically switching to other topologies to adaptively meet the communication traffic changes.

Interestingly, there are a few approaches to interconnect reconfiguration targeting ASICs. FlexBus [134] introduces two hardware mechanisms for ASICs to dynamically control both the communication architecture topology and the mapping of IP cores to domains: *bridge by-pass* temporarily merges two or more bus segments into a single shared bus, while *component remapping* dynamically switches the mapping of some slaves between two buses at specific times.

In spite of the above approaches, either based on FPGAs or just relying on interconnect configuration mechanisms, effectively deciding at runtime how to reconfigure the interconnect, based on dynamic application requirements, is a challenging issue that leaves room for further investigation.

2.5 Summary

On-chip interconnects are essential to the performance of highly parallel MultiProcessor Systems-on-Chip. In particular, this chapter presented an extensive review of design automation techniques for on-chip *bus-based* interconnects. The large body of research in this area confirmed the fundamental importance of such application-driven design methodologies, particularly in data-intensive applications where the choice of the underlying communication architecture, tailored on specific requirements, is critical to the global performance. The work went through the main options available for building different on-chip interconnect topologies, i.e. hierarchical buses, crossbars, cascaded crossbars etc. Then it surveyed the most relevant techniques in the literature to analyze a given interconnect solution and reviewed the main approaches available for interconnect synthesis, including several advanced aspects such as co-synthesis of memory and communication architectures, joint scheduling and interconnect synthesis, floorplanning, dynamic configuration, multi-path communication.

Chapter 3

Automated Synthesis of Heterogeneous Interconnect Topologies

A methodology to automatically generate an on-chip synthesizable interconnection structure in a memory-mapped communication environment, satisfying given area constraints, is proposed in this chapter. Specifically, the approach combines crossbars and shared buses, connected through bridges, in a hierarchical topology inherently supporting multiple communication paths, yielding a scalable structure and enabling efficient communication patterns. The resulting architecture improves the level of communication parallelism that can be exploited, while keeping area requirements low, as proven by a couple of case-studies presented at the end of the chapter.

3.1 Problem definition and methodology overview

This chapter presents a novel approach to automatically building hierarchical interconnection structures, minimizing the communication overhead and maximizing the degree of parallelism that can be achieved by concurrent communication interactions. In that respect, two key aspects are the capability of exploiting spatial locality in the communication patterns and the support for heterogeneous, hierarchical topologies enabling concurrent communication. The first aspect essentially consists in attempting to place the nodes that communicate more frequently closer to each other, minimizing the traffic between communicating elements and matching the localized traffic patterns induced by a given application. The communication parallelism is made possible by defining *local domains*, i.e. subsets of nodes in the interconnect that can directly communicate with each other, independent of other domains where different communication interactions can take place concurrently. Local domains are implemented as either crossbars or buses. By appropriately controlling the mapping of the address spaces in each bridge, furthermore, multiple physical paths can be realized between pairs of communicating elements in different domains. Multiple paths introduce flexibility in the network topology as they create a further opportunity for balancing the load across different channels.

A fundamental performance/cost trade-off addressed by the proposed methodology is the choice of the actual implementation for local domains. These can be built as either shared buses or crossbars. The envisioned approach chooses buses for those situations where we do not strictly need a full-crossbar network. The methodology defines the local domain and builds the actual technology mapping in an iterative fashion, minimizing the communication overhead and maximizing the communication parallelism under given area constraints.

Precisely, the interconnect synthesis problem is stated as follows. Given

- a communication graph $CG = G(V_m, V_s, E)$, as defined in chapter 2.2, where each edge $e_{m,s}$ has an attribute, $V(e_{m,s})$, representing the communication volume in terms of the amount of bytes to be transferred between master v_m and slave v_s;
- area constraints;

find:

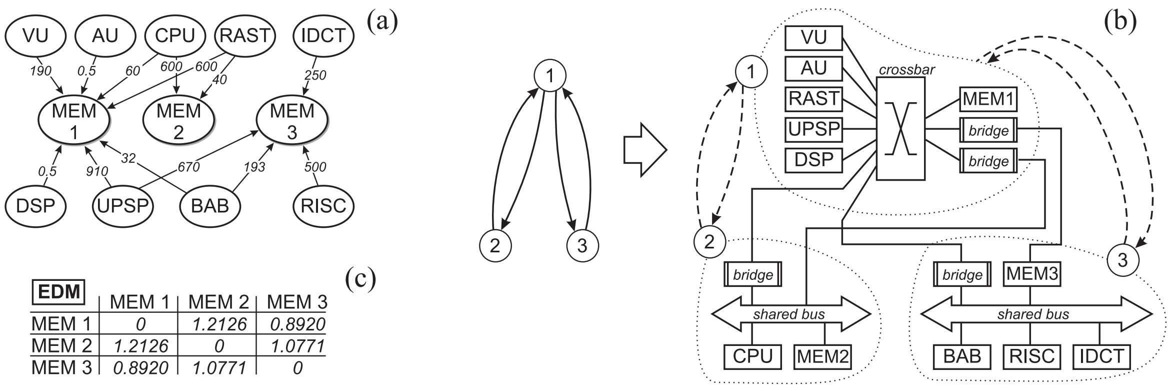
- a bus-based on-chip communication architecture $Arch(T(Sb, C, B), \Omega(V))$ specification based on a heterogeneous bus/crossbar architecture, as defined in chapter 2.2, minimizing the target cost function.

As an example, consider Figure 3.1. Part (a) contains an example of a Communication Graph with 9 masters and 3 slaves. The numbers on the arcs are representative of the $V(e_{m,s})$ attribute, here expressed in MB. When describing a computing architecture, an element of the V_m set corresponds to a microprocessor or a DMA controller or, more generally, to any master port of a unit in the system. The elements of V_s, on the other hand, are representative of the slave ports on the interconnect. They can be either ordinary peripherals or manually optimized hardware components used for application-specific purposes. Given a CG, a communication architecture description can be generated, expressing a specification directly synthesizable to an interconnect. A $Arch(T(Sb, C, B), \Omega(V))$ is made up of local domains interconnected by means of bridges. A local domain can be either a bus or a crossbar where some masters and slaves are involved in communication tasks. Communication on a global scale between different local domains can be done via a proper configuration of bridge address ranges. In the crossbar case, the output will include a connection matrix that fully specifies the connections between every master port and every slave port, possibly defining sparse crossbars. Figure 3.1.(b) gives an example of a $Arch(T(Sb,C,B),\Omega(V))$ derived from the CG in Figure 3.1.(a).

A $Arch(T(Sb,C,B),\Omega(V))$ implementation must meet interconnect area constraints. Formally, call: $N_c^{n\times m}$ the number of $n\times m$ *crossbars*, $A_c^{n\times m}$ their on-chip area, $N_{sb}^{n\times m}$ the number of $n\times m$ *shared buses*, $A_{sb}^{n\times m}$ their area, N_{br} the number of unidirectional *bridges*, and A_{br} their area:

$$\sum_m\sum_n \left[N_c^{n\times m}\cdot A_c^{n\times m}+N_{sb}^{n\times m}\cdot A_{sb}^{n\times m}\right]+N_{br}\cdot A_{br}\leq AREA$$

where $AREA$ denotes the constraint on the used hardware resources. Figure 1.(c) shows an example of an *Euclidean Distance Matrix (EDM)*, a symmetric square matrix showing in each position (i,j) the affinity between slaves s_i and s_j. If s_i and s_j exhibit a similar behavior according to the quantity of exchanged traffic with masters, their affinity tends to be high. This data structure is critical for the selection of local domains. Refers to Section 3.2 for more details.



EDM	MEM 1	MEM 2	MEM 3
MEM 1	*0*	*1.2126*	*0.8920*
MEM 2	*1.2126*	*0*	*1.0771*
MEM 3	*0.8920*	*1.0771*	*0*

Fig. 3.1 A few examples (a) A *Communication Graph* (CG). (b) A *Communication Architecture* $Arch(T(Sb,C,B),\Omega(V))$. (c) A *Euclidean Distance Matrix* (EDM).

The proposed topology synthesis flow consists of two phases:

1. Communication elements clustering;

2. Inter- and intra-cluster topology definition;

The first phase generates *local domains*, i.e. subsets of nodes in the interconnect that can directly communicate with each other. This step is performed by clustering the masters and slaves so as to exploit spatial locality maximizing the local traffic inside each domain in order to match the localized traffic patterns induced by a given application. Local domains will be implemented as either crossbars or buses. In the second phase, the clusters are connected in order to make all inter-cluster communications feasible by means of bridges. By properly setting the mapping of the address spaces in each bridge, furthermore, multiple physical paths between different domains can be realized. Multiple paths introduce flexibility in the network topology as they create a further opportunity for balancing the load across the interconnect. In addition, we need to figure out how a single cluster will be implemented. This phase also covers the definition of the topology for each cluster. The identification of the final implementation for the local domains is driven by the area constraints.

3.1.1 Assumptions

This chapter first specifies the assumptions we made for this interconnection design problem and then offers an overview on the proposed methodology. First, we target a memory mapped communication architecture with all interconnect channels sharing the same properties (protocol, channel width, clock frequency, etc.). The routing is deterministic and defined by statically setting bridge addresses. Next, we assume that the communication traffic is made up of *communication tasks*, i.e. non-preemptive atomic entities with an arbitrary load (amount of transmitted bytes) transferred in a burst mode. We do not take into account data dependencies, i.e. we assume that all nodes are always ready to transmit their remaining amount of data. This is true for pipelined data-flow shared-memory systems, where each node always reads data from memories and writes them back after elaboration. When relaxing this assumptions our approach still works fine leading to a softly over-provisioned communication infrastructure.

3.2 Communication elements clustering

The clustering is one of the main steps involved in the interconnect architecture definition. In fact, the quality of the clusters heavily affects both the area occupation and the degree of communication parallelism that the final interconnect can exploit. This phase takes place in two steps: 1) hierarchical slave clustering, and 2) master assignment to the slave clusters.

The first step performs an agglomerative hierarchical clustering in order to fix the number of local domains and determine the partitioning of the slave nodes. To cope with this problem, slaves are represented in a Euclidean n-space. For each slave v_s, we build an array containing N_m elements, the total number of masters, where each array element i represents the fraction of the total traffic from/to slave v_s involving master i. Then, in order to decide which clusters should be combined, the Euclidean distance is used as a measure of dissimilarity between slaves. The *Euclidean Distance Matrix* (EDM), containing the Euclidean distance for each pair of slaves, is built. Notice that $EDM(h,k) \leq \sqrt{|V_m|}\ \forall h,k$.

Figure 3.1.(c) shows the EDM matrix for the CG in Figure 3.1.(a). The EDM is the input of the clustering algorithm, described below, which returns a number of clusters based on the given area constraints.

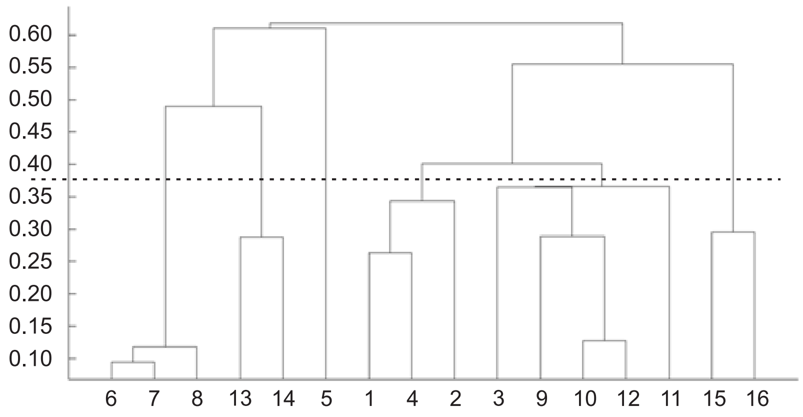


Fig. 3.2 An example of agglomerative hierarchical clustering of slave nodes. Slaves are on the x-axis, while the Euclidean distance is on the y-axis.

Figure 3.2 shows an example of agglomerative hierarchical clustering of slave nodes based on the EDM. The dotted line, corresponding to a particular value of the Euclidean distance Δ, determines the groups of slaves that will be part of the same cluster. The algorithm starts from an initial value Δ_0 for Δ, reaching a final value Δ_n. At the beginning, the algorithm verifies if the clustering has reached an intra-cluster Euclidean distance Δ_0. Once this is reached, the algorithm verifies if that clustering would be feasible under the given area constraints in the worst-case conditions (i.e. assuming all possible bridges between clusters and intra-cluster communication based on full crossbar). If the cluster is feasible, the algorithm goes on using $2\Delta_0$ as the limiting value. There are two possible outcomes: the algorithm either finishes without saturating the area, in which case only one local domain will be used maximizing communication concurrency in the case of a full crossbar, or two values $i * \Delta_0$ and $j * \Delta_0$ ($i < j$) are found where the latter is infeasible and the former is feasible. The above steps are then repeated once more using $\Delta = \Delta_0/2$. The algorithm always stops if the condition $\Delta = \Delta_n/2$ is reached. Acceptable values for Δ_0 and Δ_n are $0.2 \cdot \sqrt{|V_m|}$ and $0.02 \cdot \sqrt{|V_m|}$. Notice that the working frequency of a crossbar/bus is not heavily dependent on the number of its ports, while the area overhead tends to grow rapidly when adding ports. As a consequence, without area constraints, the clustering iterations leads to a single crossbar, which is consistent with our goals because a single large crossbar guarantees the lowest possible communication overhead. With increasingly stringent area constraints, we obtain a growing number of smaller clusters.

The second step assigns the masters to the clusters with which they exchange most data in order to keep as much communication as possible within single clusters (intra-cluster communication) and minimize the communication through bridges. Figure 3.3 shows an example. Masters M_1, M_6, M_8, M_9 are assigned to the first cluster, M_3 to the second, and so on.

	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12
Cluster 1	$\boxed{5}$	0	0	0	25	$\boxed{55}$	230	$\boxed{100}$	$\boxed{180}$	20	5	0
Cluster 2	0	0	$\boxed{600}$	0	0	0	0	0	0	300	0	0
Cluster 3	0	0	40	$\boxed{700}$	300	0	0	0	0	190	$\boxed{50}$	0
Cluster 4	0	$\boxed{5}$	0	0	0	0	0	0	0	5	0	5
Cluster 5	0	0	0	0	$\boxed{715}$	5	$\boxed{550}$	20	0	140	0	$\boxed{90}$
Cluster 6	0	0	20	20	30	0	0	0	0	$\boxed{320}$	30	0

Fig. 3.3 An example of master assignment. Values are expressed in MB.

At the end of this phase, all masters and slaves are divided in local domains whose internal and external topologies are however still to be defined.

3.3 Inter- and intra-cluster topology definition

The aim of this phase is to determine both the inter- and the intra-cluster topology. It is not possible to address the two problems separately because they are deeply inter-related due to area occupation constraints. In order to increase communication parallelism we have to maximize intra-cluster communication while keeping inter-cluster communication as low as possible and satisfying the given area constraints. Hence, we resort to an approach that is optimum for the definition of the intra-cluster topology and best-effort for the inter-cluster topology, under given area constraints. We proceed as follows:

1. Define a basic inter-cluster topology, making all inter-cluster communications feasible
2. Define the topology of each cluster
3. As long as area constraints are met, add bridge components in order to exploit Inter-cluster parallelism

For step 1), we define the number and the position of the bridges by solving an optimum branching problem. We rely on a well-known algorithm (i.e. Edmonds' algorithm), setting the weights on the arcs to the inverse of the communication requirements between clusters. In other words, we prioritize the arcs having higher communication requirements, i.e. we place bridges between clusters that exchange more data.

Concerning step 2), for each local domain we need to define its topology. We use a greedy heuristic approach: we order the clusters in a descending order according to the total traffic that they have to deal with. Starting from the first one in this ordered list (i.e., a list of clusters) we try to associate a crossbar to that cluster; that crossbar can be sparse depending on the required connectivity between inputs and outputs. The possibility of considering sparse crossbars during the topology determination process, rather than at a later step, can greatly benefit area requirements. The cluster list is then evaluated. The available area is the one expressed by the area constraint minus the area required by the bridges created during the previous step. The criteria for assigning a crossbar or a bus are as follows:

1. Compute the area of the necessary sparse crossbars.

2. If the area of the necessary crossbars is less than the available area minus the area of a shared bus multiplied by the number of clusters for which the topology has not been defined yet, then assign the crossbar; otherwise assign a shared bus.

3. Go to the next cluster in the list and start from step a).

The idea behind this heuristic is that we want to maximize communication parallelism in the clusters where the demand for traffic is higher while still meeting area constraints.

Step 3) deals with the enhancement of the inter-cluster topology. Keeping in mind that Step 2) has decreased the overall available area, the greedy heuristic for this step proceeds as follows:

1. Build the set (call it *S*) of all the possible bridges between all clusters that were not added during Step 1).

2. Sort the list in descending order according to the traffic demand (i.e., the amount of bytes to be transferred between each pair of clusters).

3. For each element in *S*:

   (a) Increment the available area with the cost of two crossbars (or buses, depending on the chosen topology).

   (b) Subtract from the available area the cost of a bridge and the cost of two crossbars (or buses), where the size of the crossbars/buses to be connected also takes into account the port for the new bridge. If the area constraint is still met, add this bridge to the inter-cluster topology.

Each local domain can have more than one bridge connected to it, and each bridge can forward data to other bridges in the network. This enables traffic from the same cluster (and, in principle, from the same communicating element) to reach another cluster (or another communicating element) following different and, possibly, multi-hop paths. In order to correctly exploit inter-cluster path parallelism, the definition of the slave and bridge address windows is key, as it determines the actual routing of data through the network, possibly letting the same slave exchange data through different paths depending on the addresses of the transferred data. In that respect, the bridge address ranges are defined by solving a path balancing problem [37]. This approach allows us to configure bridge address ranges in such a way as to balance the traffic over different links, effectively exploiting parallel communication paths available in the topology. Section 2.4.4 exemplifies the benefits obtained from inter-cluster multi-path communication.

3.4 Experiments and Case Studies

3.4.1 Experimental Setup

We carried out the experiments on a prototyping FPGA board, namely a ZedBoard by Avnet Design Services for the Xilinx Zynq™-7000 [7]. The communication architecture synthesis flow uses the Xilinx AXI components compliant with the AMBA® AXI version 4 specification from ARM [8]. We built accurate analytical models for the evaluation of the area cost, the latency and the power consumption, obtained by interpolating extensive RTL synthesis results. From the data collected, we extrapolated the following equations, valid for the Zynq-7000 family, used to calculate the area information of AXI-based interconnects:

$$A_c^{n\times m} = 101n + 60nm + 42m + 874 \tag{3.1}$$

$$A_{sb}^{n\times m} = 80n + 18.75m + 95.5 \tag{3.2}$$

We considered burst-based transactions with only the start address issued and the payload transferred in a single burst that can comprise multiple beats[1]. Concerning the bandwidth estimation, we considered the bandwidth of a channel as the maximum rate at which a master can receive/send data from/to a slave. Due to the burst-based communication, we considered that address arbitration does not impact that rate significantly[2]. To compute the execution

[1]A beat is an individual data transfer within an AXI burst.

[2]According to the component documentation, arbitration latencies typically do not impact data throughput when transactions average at least three data beats.

time of a communication task, we proceeded as follows: intra-cluster communication tasks require 52 clock cycles every 64 bytes of data. In case of inter-cluster communication 10 additional clock cycles are required for each traversed bridge. Obviously, these numbers are dependent both on the architecture and the actual IP cores used. Concerning the static power consumption, we considered the power from transistor leakage on all connected voltage rails and the circuits required for the FPGA to operate normally. Obviously this power is independent of the user design and, consequently, it is affected only by voltage and temperature. Unlike static power, dynamic power is the power of the user design, i.e. it depends on the input data pattern and the design internal activity. We calculated this power by means of simulation results based on the average switching rate of every signal in the interconnect.

3.4.2 Results

We tested the method for a real-world application and a synthetic one. The real-world application (AppI) is an MPEG4 decoder with 9 masters and 3 slaves derived from [139], while the synthetic one (AppII) consists of 12 masters and 16 slaves and is representative of an application with a medium/high communication workload. The characteristics of the benchmarks are summarized in Table 3.1.

	VU	AU	CPU	RAST	IDCT	RISC	BAB	UPSP	DSP
MEM1	190	0.5	60	600	0	0	32	910	0.5
MEM2	0	0	600	40	0	0	0	0	0
MEM3	0	0	0	0	250	500	193	670	0

	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}	M_{12}
S_1	5	0	0	0	5	15	100	0	80	0	0	0
S_2	0	0	0	0	20	10	30	60	50	10	5	0
S_3	0	0	0	0	25	5	10	20	0	20	0	0
S_4	0	0	0	0	0	30	100	40	50	10	0	0
S_5	0	5	0	0	0	0	0	0	0	5	0	5
S_6	0	0	300	0	0	0	0	0	0	150	0	0
S_7	0	0	150	0	0	0	0	0	0	100	0	0
S_8	0	0	150	0	0	0	0	0	0	50	0	0
S_9	0	0	0	0	90	0	100	0	0	60	0	30
S_{10}	0	0	0	0	200	0	290	0	0	0	0	45
S_{11}	0	0	0	0	350	0	100	0	0	60	0	10
S_{12}	0	0	0	0	50	0	50	0	0	0	0	5
S_{13}	0	0	5	0	0	0	0	0	0	15	5	0
S_{14}	0	0	15	20	30	0	0	0	0	315	25	0
S_{15}	0	0	40	300	220	0	0	0	0	80	25	0
S_{16}	0	0	0	400	80	0	0	0	0	110	25	0

Table 3.1 Characteristics of the case-study applications

For each application, we applied our method and synthesized the resulting architecture in a MPSoC. AppI is a small size application with only 3 slaves. Its CG is shown in Figure 3.1.(a). In the graph, the circles denote the processing or memory cores and are annotated by their respective types. The edges denote the communication between cores, and are annotated by the global amount of data in MB for aggregate read/write operations. Figure 3.1.(c) shows the AppI Euclidean Distance Matrix (EDM). As previously mentioned, the Euclidean Distance between two slaves is bounded by $\sqrt{|V_m|}$, where $|V_m|$ is the cardinality of V_m. In this application $|V_m| = 9$, so the Euclidean Distance ranges here between 0 and 3. The suggested value of $\Delta_0 = 0.6$ is never reached, so our approach finds 3 clusters, one for each slave. This is justified by the limited affinity between the slaves. Notice that the limited scale of the problem does not create any significant opportunity for optimizing the communication in this case. The area available for the interconnection architecture was less than 4000 LUTs. The topology obtained consists of a 7×3 crossbar, a 2×2 bus, and a 4×2 bus for cluster 1, 2, and 3, respectively, with a total area of 3713 LUTs. Figure 3.1.(b) shows the $Arch(T(Sb,C,B),\Omega(V))$ graph and its corresponding topology implementation. The circles denote the clusters while the edges denote the inter-cluster communication. To evaluate the overall impact of our method, we compared it with a previous method based on MILP [68]. The MILP approach can operate with two different objectives: maximize the operating frequency (Objective 1) or minimize the total crossbar area (Objective 2). To perform a timing and area comparison, we applied the MILP approach and we implemented on FPGA the interconnect architectures obtained. We used the Xilinx LogiCORE IP AXI Timer (v1.03.a) to have timing information. Notice that the time to execute a memory transfer depends on the hop-count in terms of number of bridges to traverse. The goal of our methodology is to minimize the inter-cluster traffic with the aim of maximizing the parallelism of the communication while minimizing the multi-hop communication. In that respect, a useful metric for quantifying this behavior was introduced in [116]: the *localization factor* L_f, i.e. the ratio of the local traffic to the total traffic. For example, if $L_f = 0.6$, then 60% of the traffic generated by an IP occurs within its cluster while the rest of the traffic involves the remaining parts of the SoC. For the AppI application, we calculated the localization factor for the topology obtained with our approach and with the MILP approach. We found $L_f = 0.80$ with our approach, while $L_f = 0.39$ with MILP. As a result, we have measured a 17% reduction in the total communication time compared with the communication architecture obtained by optimizing the first objective, while a reduction of 10% in the total area occupied compared with the topology obtained by optimizing the second objective.

AppII, on the other hand, is a medium/large-size application with a significant number of masters and slaves. Our approach found 6 clusters. Figure 3.2 shows the hierarchical clustering tree for this application. The dotted line represents the stop value where the algorithm converges. A larger value means having larger crossbars/buses with a considerable growth in the area occupation and a higher communication efficiency. The extreme solution is an unbounded clustering leading to a single crossbar. This solution is very expensive in terms of area occupation but offers the best parallelism achievable with only intra-cluster traffic and $L_f = 1$. At the opposite end, the extreme solution is the one with one cluster for every slave. This solution is reached with a $\Delta_0 < \min(EDM(h,k))$ and expresses the highest level of inter-cluster traffic. Figure 3.3 shows how masters are assigned to clusters for AppII. The procedure maximizes the intra-cluster communications by assigning a master to the cluster with which it communicates most. Due to the area constraints, the topology obtained consists of three crossbars, one 6×8, one 4×3, and one 2×4, as well as three shared buses, one 5×5, one 2×4, and one 2×1 bus.

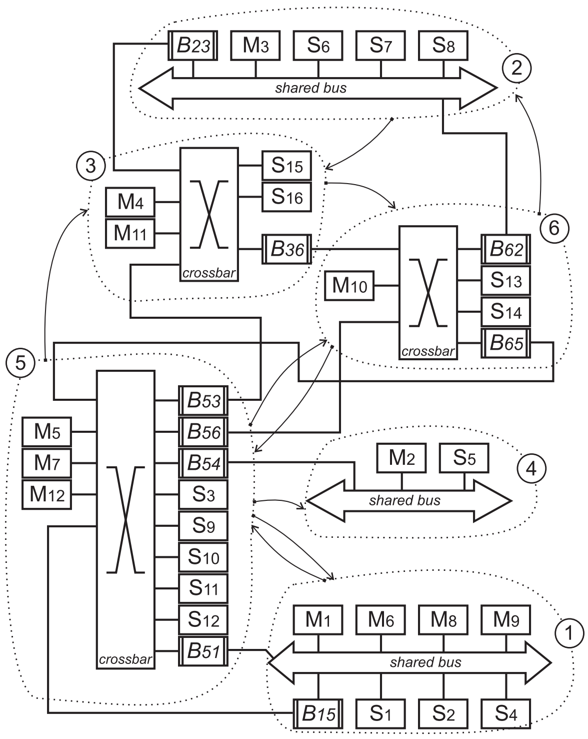


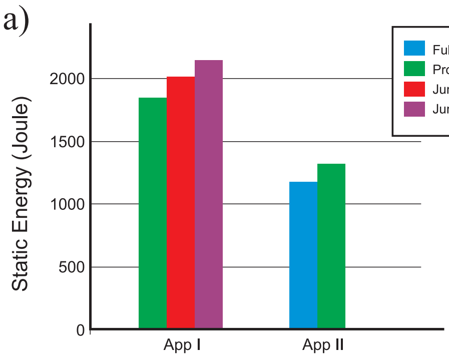
Fig. 3.4 The communication architecture description for the second case-study application (AppII). (M: master, S: slave, B: bridge.)

Figure 3.4 shows the $Arch(T(Sb,C,B),\Omega(V))$, where each region enclosed by a dotted line represents a cluster. The figure also emphasizes a few multiple paths between some clusters. Communication between clusters 5 and 6 can go through bridge B_{56} or follow the multi-hop path going through B_{53} and B_{36}. Similarly, communication between clusters 6 and 3 can go through bridges B_{62} B_{23} or B_{65} B_{53}. For this application the total area of the interconnection architecture is 9720 LUTs, while a 12×16 full crossbar occupies around 14200 LUTs, 46% more than our solution. Concerning traffic locality, our topology has a $L_f = 0.7$, causing our solution behave slightly worse in terms of total communication time, although it needs only 10% more time to complete compared to a full crossbar solution, which of course reach the best possible L_f.

We also evaluated the overall energy consumption for each case-study application, taking into account the power consumption and the overall execution time incurred by the communication tasks. Figure 3.5 shows the main results in terms of both static and dynamic energy (based on the Xilinx XPower power estimation tool), referring to a full crossbar implementation, a previous literature solution [68], and the presented approach. Static energy consumption, as expected, is directly proportional to the execution time. Notice that the results provided by XPower in terms of static power refer to the whole chip and may vary for other devices. Nevertheless, they are indicative of the energy saving enabled by improved overall execution times. The dynamic energy consumption, on the other hand, does depend on the specific structure of the implemented design. The proposed method generates interconnect architectures consuming on average 5% less dynamic energy than full crossbar implementations and respectively 10% and 30% less dynamic energy than [68] for Objective 1 and Objective 2. To interpret these results, recall that dynamic power can be modelled as $P_{dynamic} = {V_{DD}}^2 \cdot \sum_{n \in nets} (C_n \times f_n)$ where V_{DD} is the supply voltage, while C_n and f_n are, respectively, the capacitance and the average toggle rate (switching activity) of a net n. The number of nets, in the case of crossbar solutions, grows quadratically with the number of ports. As a consequence, a large crossbar tends to consume more dynamic power per port than an interconnect made up of small crossbars or buses. This explains the advantage in terms of energy consumption of our approach over the full crossbar implementation, in spite the fact that the full-crossbar implementation has an overall execution time 10% lower than our approach. The improvement over [68], on the other hand, is due to the advantage in terms of either area or execution time for both objectives. In fact, compared to [68], Objective 1, our solution leads to architectures with similar dynamic power but better execution time, while, compared to [68], Objective 2, it leads to architectures with similar execution time but better dynamic power.

3.5 Summary

This chapter presents an automated design methodology for the synthesis of complex heterogeneous on-chip interconnects made of crossbars, buses, and bridges starting from the specification of the application requirements. This solution is based on a greedy algorithm to cluster communicating elements in local domains and a method to balance the load across the bridges. The algorithm aims at maximizing both intra- and inter-cluster communication parallelism, respectively by concentrating traffic within local domains and by creating

a)


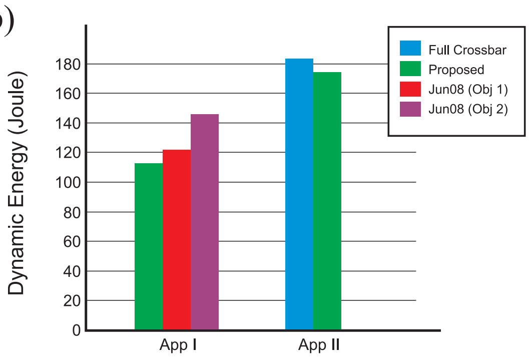
b)


Fig. 3.5 Energy consumption comparison vs. [68] (denoted as Jun08 in the charts) and a full crossbar solution. (a) Static energy (b) Dynamic energy.

parallel, multi-hop inter-cluster communication paths. Experimental results show that this approach can synthesize designs made of dozens of IP cores with a small communication overhead and low area and power requirements, exhibiting encouraging improvements over previous proposals in the literature.

Chapter 4

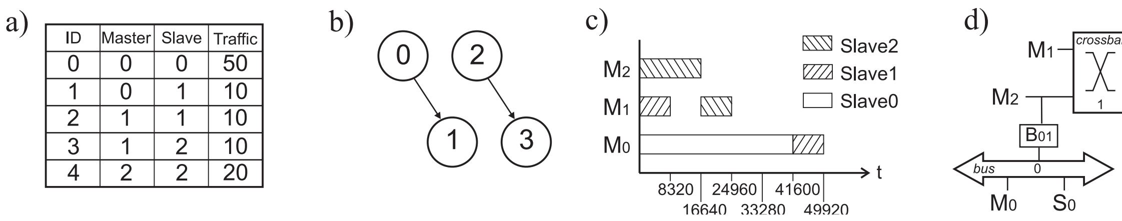
Joint Communication Scheduling and Interconnect Synthesis

Based on the observation in Chapter 2.4.2, in this chapter, we propose a complete methodology supporting a joint communication scheduling/interconnect synthesis optimization. By jointly solving a scheduling and interconnect synthesis problem, the methodology turns the description of the application communication requirements, including data dependencies, into an on-chip synthesizable interconnection structure along with a communication schedule satisfying given area constraints. As in the previous chapter, the interconnect architecture description generated by the methodology is based on a combination of cascaded crossbars and shared buses, connected through bridges. This chapter thoroughly describes the formalisms and the methodology used to derive such optimized heterogeneous topologies. It also discusses some case-studies emphasizing the impact of the proposed approach and highlighting the essential differences with a few other solutions presented in the technical literature.

4.1 Problem Definition

This chapter proposes an application-specific on-chip network topology synthesis and communication scheduling method. It takes as input the information on the communication tasks and their dependency relationships, and generates the specification of an on-chip interconnect along with the communication task schedule. In the following we give some useful definitions as well as a simple example.

Definition 4.1.1. *A* Task List *(TL) is a list, made up of n_{task}* communication tasks*, indexed by a unique* taskID *where each entry t_i contains the* master *and the* slave *involved in the*



ID	Master	Slave	Traffic
0	0	0	50
1	0	1	10
2	1	1	10
3	1	2	10
4	2	2	20

Fig. 4.1 A few examples ("M" : master, "S" : slave, "B" : bridge.) (a) A *Task List* (TL). (b) A *Dependency Graph* (DG). (c) A *Communication Schedule* (CS). (d) A *Communication Architecture Description* (OCA).

communication and a cost c_i *(i.e. the data traffic) in terms of number of bytes to be transferred.*

The communication traffic between a master and a slave is made up of communication tasks which are non-preemptive atomic entities with an arbitrary load, i.e. the amount of transmitted bytes, transferred in a burst mode. Notice that two different tasks can have the same master/slave pair. This allows modeling any traffic pattern.

Definition 4.1.2. *A* Communication Schedule *(CS) is defined as a vector indexed by a task identifier containing in each entry* s_i *the* start time *of communication task* t_i *in clock cycles. The* latency *of the CS is the number of cycles to execute the entire schedule, or equivalently, the difference in start time of the sink and source vertices. If two or more sources/sinks are available, the source/sink with the smallest/biggest start time is used. Since a feasible solution must satisfy data dependency constraints, the start time of a communication task is at least as large as the start time of each of its predecessors plus their* execution delay d_i. *An execution delay is an integer representing the amount of clock cycles required to execute a communication task in a given architecture. Therefore a schedule must satisfy the following relations:*

$$s_i \geq s_j + d_j \quad \forall i, j : (v_j, v_i) \in E \tag{4.1}$$

It is of course desirable to find the minimum latency *schedule that can be run on a physical topology under given area constraints.*

As an example, consider Figure 4.1. Part a) contains a TL with five tasks where, for each task, the master/slave pair and the amount of traffic they exchange are specified. The DG (Figure 4.1.b) expresses the existing dependency relationships between the tasks. In this case, there are dependency constraints between t_0 and t_1 and between t_2 and t_3. In Figure 4.1.c, a possible scheduling solution is depicted. It is the as-soon-as-possible (ASAP) schedule for the given DAG and TL. Notice that although t_3 and t_4 do not have any data dependency, their execution is serialized due to a slave incompatibility: two different tasks can not simultaneously access the same slave. In the same way, accessing simultaneously the same bridge in a multi-hop communication would lead to a tasks incompatibility. Figure 4.1.d depicts the OCA made of two clusters (C_0 and C_1) and one bridge allowing task t_1 to be performed. C_0 is implemented by a shared bus and C_1 by a crossbar. This topology exhibits the required parallelism to run the above schedule. This is obtained by taking advantage of both global parallelism, because there are two clusters running concurrently, and local parallelism, because cluster C_1 is implemented as a 2×2 crossbar.

4.1.1 Objectives

The interconnect synthesis problem is stated as follows. Given

- a communication Task List (TL) containing, for each task, the master and the slave involved and the amount of bytes to be transferred;
- a Dependency Graph (DG) describing the possible inter-task precedence relationships, i.e. data dependency constraints;
- area constraints;

find:

- a On-chip Communication Architecture (OCA) specification based on a heterogeneous bus/crossbar architecture minimizing the target cost function;
- a minimum-latency communication task schedule (CS) compatible with the identified architecture.

The problem of communication scheduling and synthesizable topology definition are deeply interrelated. The schedule identifies the precise start time of each communication task. The start times must satisfy the DG dependencies, which limits the amount of parallelism of the communication, because any pair of communication tasks involved in a direct dependency, or a chain of dependencies, may not execute concurrently. Determining the concurrency of the topology implementation, scheduling affects the resulting performance and cost. Similarly, the maximum number of concurrent communication tasks at any step of the schedule is bounded by the interconnection topology. The final architecture must be able of handling the parallelism offered by a certain schedule.

The cost of the interconnect in terms of area can be upper bounded to satisfy some design requirements. When resource constraints are imposed, the number of concurrent communication tasks whose execution can overlap in time is limited by the parallelism of the topology. Tight bounds on the interconnect area cause serialized communication. As a limiting case, a scheduled sequencing graph may be such that all communication tasks are executed in a linear sequence. This is indeed the case when only a single bus is available to execute all communication tasks. On the contrary, the unconstrained solution is a crossbar with a suitable number of connections enabling the maximum number of concurrent communication tasks according to DG dependencies. Area/latency trade-off points can be derived as the solutions to different constrained scheduling problems. This methodology jointly considers them to derive a heterogeneous interconnection topology satisfying both area and dependency constraints.

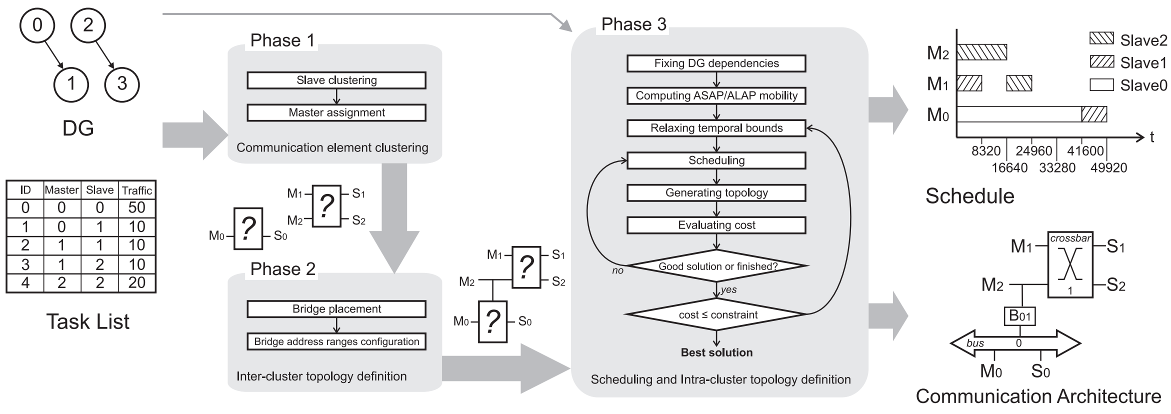
4.1.2 Assumptions

Following are the main assumptions we made for the interconnection design problem:

- First, we target a memory mapped communication scenario where the memory blocks and I/O devices are mapped to slave nodes while the initiators, such as CPUs and DMAs, are mapped to master nodes sharing the same address space in the system. IP cores that have both roles, i.e. initiator and target, are mapped to both a master node and a slave node, handled like the other nodes.
- All the interconnect channels use the same protocol and channel width.
- The routing is deterministic and defined by statically setting bridge addresses.
- We neglect the overhead due to bus contention when computing latencies.
- A pre-characterized library of crossbars and buses is ready for various configurations (e.g., different numbers of masters and slaves). We built an accurate area cost model, obtained through interpolation of extensive area characterizations using RTL synthesis. We will explain the proposed method and show the experimental results for AXI-based interconnects. However, the proposed method can also be applied to crossbar-based interconnects implementing other protocols.

4.1.3 Overview of the proposed method

This section presents a novel approach to automatically building highly parallel interconnection structures, minimizing the communication overhead and maximizing the degree of parallelism that can be achieved by concurrent communication interactions. The approach also deals with dependency constraints expressed using the DG. The proposed topology synthesis flow, shown in Figure 4.2, consists of three phases:



ID	Master	Slave	Traffic
0	0	0	50
1	0	1	10
2	1	1	10
3	1	2	10
4	2	2	20

Fig. 4.2 Proposed interconnect synthesis flow

1. Communication elements clustering
2. Inter-cluster topology definition
3. Scheduling and intra-cluster topology definition

The first step performed (Phase 1) is the clustering of communicating elements in local domains. The capability of exploiting spatial locality in the communication patterns is here key, as we attempt to place the nodes that communicate more frequently closer to each other, minimizing the traffic between communicating elements and matching the localized traffic patterns induced by a given application.

In Phase 2 the clusters are connected in order to make all inter-cluster communications feasible by means of bridges. By properly setting the mapping of the address spaces in each bridge, furthermore, multiple physical paths between different domains can be realized. Multiple paths introduce flexibility in the network topology, as they create a further opportunity for balancing the load across the interconnect, as well as concurrency in inter-cluster communication (inter-cluster parallelism).

Finally, we need to figure out how single clusters will be implemented (Phase 3). This step is performed jointly with the communication scheduling: an iterative procedure finds an optimal communication tasks schedule (in terms of latency) and a global topology containing enough resources to execute the found schedule. The identification of the final implementation for the local domains is driven by the area constraints. The chosen implementation must ensure a degree of intra-cluster parallelism compatible with the identified schedule. The definition of an effective method generating the intra-cluster architecture given a global scheduling solution is essential.

Notice that the above approach targets a single application. In case different applications are anticipated to share the same interconnect architecture, the methodology can still be adopted by identifying a clustering that averages the characteristics of all applications, similar to cluster ensemble techniques [142]. While the first step of this approach is presented in Chapter 3.2, the second and third steps covered by this methodology are explained in the following sections.

4.2 Inter-cluster Topology Definition

As already mentioned, both inter-cluster (i.e. the O set of the ST) and intra-cluster topologies (i.e. the I function of the ST) need to be defined. The aim of this phase is to determine the global inter-cluster architecture: the O set defines the actual links between the clusters (from an implementation viewpoint, it determines the number and the positions of the bridges between crossbars and buses). The intuition behind the proposed algorithm is that, in order to have low-latency communication, we have to maximize intra-cluster communication while keeping inter-cluster communication as low as possible satisfying given area constraints. Hence, we resort to an approach that is biased towards optimizing the I function more than the O set. As shown in Figure 4.2, this is achieved in two steps.

1) Populate the O set, making all inter-cluster communications feasible.

2) Define bridge addresses by solving a path balancing problem.

For step 1), we solve an optimum branching problem taking as nodes the C set. We rely on a well-known algorithm (i.e. Edmonds' algorithm [42]). The important clue is that we set the weights on the arcs to the inverse of the communication requirements between clusters. In other words, we prioritize arcs having higher communication requirements.

Concerning step 2), we resorted to the approach in [37] that is capable of solving a path balancing problem with a complexity of $O(n \log n)$. This approach allows us to configure bridge addresses in a balanced way without overloading a reduced number of links. Furthermore, this possibly enables the use of parallel communication paths between masters and slaves in different clusters, enabling the exploitation of inter-cluster parallelism. Section 2.4.4 shows the benefits obtained from inter-cluster multi-path communication.

4.3 Scheduling and intra-cluster topology definition

This section presents the approach to the concurrent definition of the intra-cluster architecture and the scheduling solution. This is the most complex phase of the methodology since it concurrently involves the design of the intra-cluster interconnections and the communication scheduling. The whole procedure, consisting of several steps, is depicted in Figure 4.2, Phase 3.

At the beginning, the DG is fixed in order to take into account and preemptively solve any potential structural conflict due to slave and bridge accesses. This is because the original DG, taken as input by the methodology, only expresses dependency relationships. The resulting DG will comply with the following rule: *all tasks using the same slave or bridge must be connected by a single path.* This means that there will be a partial order relation for slave and bridge accesses. This is achieved by prioritizing tasks that, in an ASAP schedule, start first. This must be computed only once by the methodology and is represented by the first step. In addition, in this step the execution delay d_i of each communication task t_i is derived in order to solve equations (4.1). Notice that d_i does not only depend on the cost of the task but also on the interconnect topology: inter-cluster communications require additional clock cycles due to bridge crossing. These values are strictly dependent on the technology and can be computed only after the definition of the global topology.

After fixing the DG, the second step computes the ASAP and ALAP schedules and the mobility values. Then, the iterative phase takes place. First, a temporal bound, subsequently relaxed at each iteration of the outer loop, is fixed (third step). Since we are optimizing the global execution time, we start with the minimum temporal bound, i.e. the latency of the ASAP schedule. In fact, the ASAP schedule finds the optimum execution time in an unconstrained problem [99]. At each iteration of the inner loop we calculate the optimal schedule, in terms of area occupied by an interconnection architecture able to run that communication schedule, under that temporal bound (fourth, fifth, and sixth steps). The temporal bound is relaxed until a solution, satisfying the area constraints, is found. A key decision here is how much the temporal bound should be relaxed. These steps rely on an algorithm to find the synthesizable architecture with the minimum degree of parallelism allowing a certain schedule to be run (fifth step). Then, the cost of the whole architecture is quantified by using a suitable area model. The last step checks if the area of the architecture found meets the given constraints. If this is not the case, we go back to the third step where, in addition to relaxing the temporal bound, the mobility values are recomputed accordingly. In the following, several essential aspects related to the above flow are described in more detail.

4.3.0.1 Relaxing the temporal bound

The granularity of the temporal bound, relaxed at each repetition of the third step above, is critical to guaranteeing a comprehensive exploration of the available design choices. Of course, relaxing the bound by a single clock cycle at each step would be infeasible. We chose to relax the bound by the minimum time allowing a task to eliminate at least one overlap in the schedule. The intuition behind this choice is that less overlapping leads to less concurrency, and hence the architecture will take less area. As an example, in Figure 4.3 the temporal bound is relaxed by two time units. Had the temporal bound been shifted by only one time unit, task t_1 would have still overlapped with task t_2.

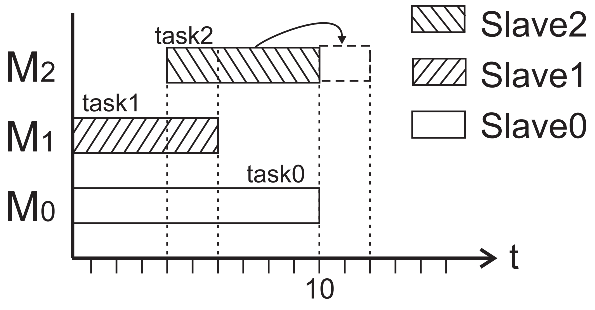


Fig. 4.3 Example of temporal bound relaxing

4.3.0.2 Scheduling Algorithms

The scheduling algorithm is responsible for determining a schedule compatible with the lowest-area synthesizable communication architecture satisfying a given temporal bound and all the constraints expressed by the modified DG (including slaves and bridges conflicts). Table 4.1 summarizes the characteristics of the different scheduling algorithms evaluated.

Table 4.1 Scheduling Algorithms

Name	Abbreviation	Solution	Type
Genetic Algorithm	GA	Approximate	Evolutionary
Priority-based List Scheduling PBLS	PBLS	Approximate	Greedy
Randomized Priority-based List Scheduling	R-PBLS	Approximate	Greedy
Random search	Rand	Approximate	Random Search
Exhaustive Exploration	EX	Exact	Exhaustive Search
Smart-Exhaustive Exploration	Smart-EX	Exact	Exhaustive Search

Genetic Algorithm (GA) The scheduling problem can be solved by general iterative methods, like genetic algorithms [23] that starts from a random initial population and then mutate and combine individuals in an iterative fashion. For implementing a scheduling algorithm relying on a genetic approach, we used the Opt4J Java-based modular framework [92]. In order to obtain only acceptable solutions at each step, we created a new *genotype* to encode the solutions. The scheduling genotype is a class containing a vector corresponding to the scheduling vector itself, indexed by the task identifier and containing in each entry the starting clock cycle of the corresponding task. The *phenotype* is simply the scheduling vector contained in the corresponding genotype. The evaluator of the phenotype is described in the subsequent paragraph and is the same for all other algorithms. The *fitness function* returns the minimum area required by a communication architecture exhibiting enough parallelism to run the identified scheduling. Due to the custom implementation of a personalized genotype, phenotype, and evaluator, a custom optimizator operator was also implemented to drive the generation of the new population. The initial population is generated starting with the ASAP solution by moving a random number of tasks by a random quantity not larger than the mobility. This guarantees a good diversity at the chromosome level with a relatively small population size. During the movements, all data and structural dependency relationships must be preserved. When generating the new population, we do not perform crossover but we only rely on mutation. Specifically, we substitute the worst L elements with L new individuals mutating the remaining ones with probability p. Mutation takes place in the same way as random generation but starting from the parent's schedule and not from the ASAP schedule. After the last iteration, the best individual is chosen. L, p, the initial population size, and the number of iterations are configurable.

Priority-based List Scheduling This implementation of the priority-based list approach (PBLS) tries, at each step, to make the best move as possible within a list of admitted moves, i.e. the moves satisfying the data-dependency constraints. The list is ordered according to the area cost incurred by the topologies associated with any potential move. The procedure to derive a topology, given a schedule, is explained in Section 4.3.0.3. Its behavior is radically different than the genetic approach. It does not admit uphill moves (causing a temporary cost increase), hence the probability of sticking at a local minimum solution tends to be high. The starting point must be a valid solution. At each step, the algorithm evaluates all neighboring solutions by analyzing what happens by moving each task by the minimum quantity to eliminate an overlap. The list is comprised of all neighboring solutions, then sorted according to the benefits achieved by taking that move. Only the best solution is chosen. When, at a certain step, there are no moves improving the solution, the algorithm takes randomly one of the possible moves leading to an equivalent cost solution and goes on. The number of consecutive random moves is bounded by a value configurable by the user. This value must be accurately tuned, keeping in mind that the solution space may be very smooth and have several equivalent neighboring solutions. When this condition persists, the identified solution is likely to be a local minimum. The search process exits when the maximum number of iterations is reached or when there are only uphill moves available. The algorithm does not perform well, in general, when the design space is irregular. Since it does not admit uphill moves, this approach is likely to yield the optimal solution only if we can take an initial scheduling that is already quite close to the optimum, otherwise it gets stuck at a local minimum (a point not having better neighboring solutions).

Randomized Priority-based List Scheduling This is, essentially an optimized version of the previous algorithm that tries to avoid getting stuck at a local minimum. This algorithm tries to explore a certain number of regions of attraction according to how many iterations are performed. The algorithm is very similar to its conventional counterpart, but when it arrives to a local minimum (a point not having better neighboring solutions) it records the solution and generates another random starting point in the hope of falling in a different region of attraction. The procedure exits as soon as a maximum number of iterations (or a maximum number of local minima) is found. Figure 4.4 shows the behaviour of the algorithm when finding local minima. The initial solution is represented by point 0. Then, the algorithm goes through the solution space until it finds the local optimum represented by point 2. Unlike the non-randomized version, it does not get stuck at this point. As an example, it may jump to point 3 with a random movement.

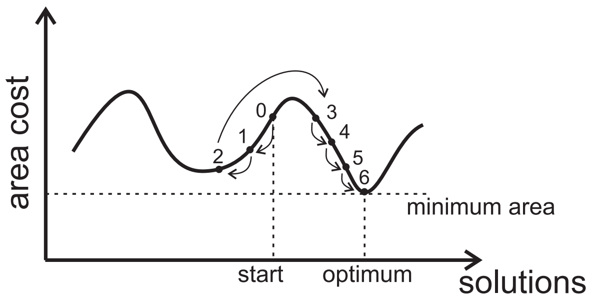


Fig. 4.4 Behaviour of the randomized list scheduling algorithm

Random search, Exhaustive exploration, and Smart-Exhaustive Exploration In order to extend the comparisons, some general problem-solving techniques were implemented: a random search and two different exhaustive search approaches.

The Smart-Exhaustive exploration is an optimized version of the standard exhaustive approach. The optimization consists in relying on an adaptive granularity when moving tasks. Specifically, the quantity by which a task must be moved is automatically computed in order to eliminate at least an overlapping in the current schedule configuration. Obviously, whenever a task is moved, all tasks dependent on it must be moved as well to preserve dependency and structural constraints. Albeit this optimization largely reduces complexity, exhaustive approaches are still too complex and can only be used for small-sized problems.

4.3.0.3 Evaluation of scheduling solutions

All the evaluated algorithms rely on a procedure to evaluate a communication task schedule. In that respect, we need to find the lowest cost architecture that exhibits enough parallelism to accommodate the identified scheduling. This process is responsible for the quality of the local parallelism. Concerning the derivation of valid intra-cluster topologies, we rely on compatibility graphs [99]. Compatibility graphs are usually adopted in binding problems, to figure out whether two scheduled operations can share a hardware resource. In this methodology, compatibility graphs are used to determine the number of master and slave ports needed by local interconnects.

Definition 4.3.1. *Given a Communication Scheduling, two tasks t_i and t_j are* compatible *if and only if they do not overlap in time.*

Definition 4.3.2. *Two masters (slaves), including bridge master (slave) ports, are* compatible *if and only if all the tasks in which they are involved are compatible and they belong to the same cluster.*

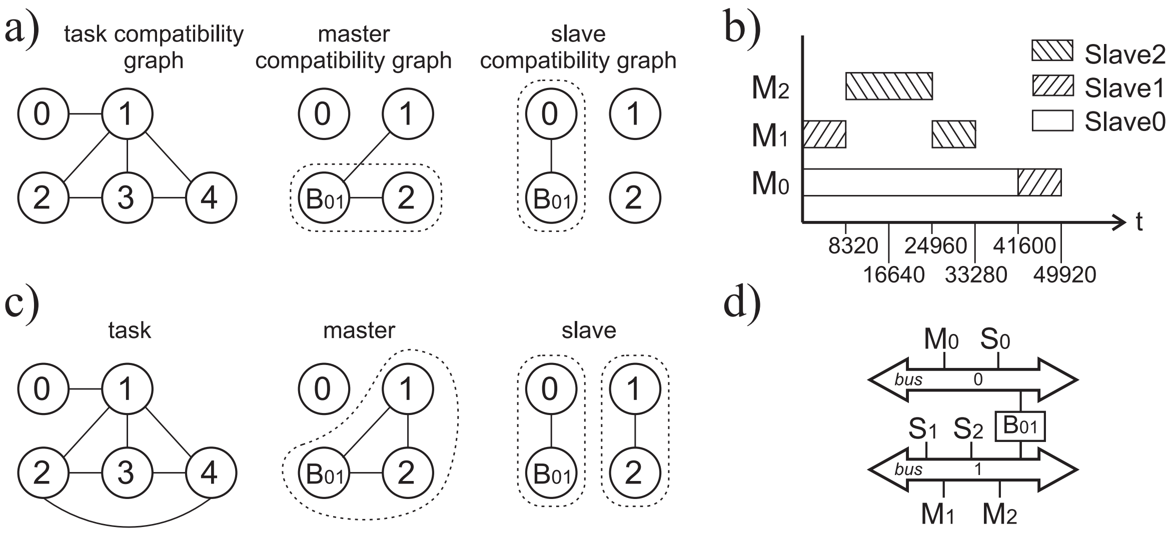


Fig. 4.5 Deriving a topology from a given schedule. (a) Compatibility graphs for the schedule in Figure 4.1. (b) An enhanced schedule, with less concurrency, for the same application. (c) Its compatibility graphs. (d) The derived topology.

From the definitions given above, the construction of compatibility graphs for communication tasks and then for masters and slaves is straightforward. First, a communication task compatibility graph $CG_t(V,E_t)$ is built. The vertex set $V = \{v_i : i = 0,1,...,n_{task}-1\}$ is in one-to-one correspondence with the set of communication tasks, while the edge set $E_t = \{(v_i,v_j) : i,j = 0,1,...,n_{task}-1\}$ denotes compatibility between the above tasks: if two tasks do not overlap, then an arc between their two vertices is placed. Then, a master $CG_m(V_m,E_m)$ and a slave $CG_s(V_s,E_s)$ compatibility graph are built. The vertex sets $V_m = \{v_i : i = 0,1,...,n_{master} - 1\}$ and $V_s = \{v_i : i = 0,1,...,n_{slave} - 1\}$ are in one-to-one correspondence with the set of masters and slaves, while the edge sets $E_{master} = \{(v_i,v_j) : i,j = 0,1,...,n_{master} - 1\}$ and $E_{slave} = \{(v_i,v_j) : i,j = 0,1,...,n_{slave} - 1\}$ denote compatibility between the masters and slaves. In order to build the master (slave) compatibility graph, we start from a completely connected graph and we delete arcs progressively as follows. For each pair of non-compatible tasks, we delete the arc between the corresponding master (slave) vertices. Starting from those compatibility graphs, for each connected component we solve a clique-partitioning problem [102] that identifies the minimum number of cliques[1]. Then, we assign a master (slave) port to each clique in its cluster. If for a cluster a single clique is identified, both in the master and the slave compatibility graphs, then a single shared bus is enough because all tasks are compatible, hence sequential. Otherwise a crossbar is necessary. If there are no clique in a cluster, then a full crossbar will be used, otherwise the number of concurrent channels will be dependent on the number of cliques. Figure 4.5 shows the differences between compatibility graphs for two distinct schedules. Figure 4.5a depicts the compatibility graphs for the schedule in Figure 4.1c. Masters M_1 and M_2 and slaves S_1 and S_2 are incompatible with each other. The derived topology, consisting of a crossbar for the cluster C_1 and a shared bus for C_0, is shown in Figure 4.1d. Notice that masters M_2 and B_{01} are compatible and, hence, they share the same crossbar port. Figure 4.5b shows a different schedule that removes the above incompatibility (Figure 4.5c). The solution of the clique-partitioning problem is highlighted by the circles. This leads to a less parallel and less expensive architecture, shown in Figure 4.5d.

4.4 Experiments and Case Studies

4.4.1 Experimental Setup

For the experiments, we used a prototyping FPGA board, namely a ZedBoard by Avnet Design Services for the Xilinx Zynq™-7000 [7]. The communication architecture synthesis flow uses the Xilinx AXI components compliant with the AMBA® AXI version 4 specification from ARM. The components we used for generating the system architecture include:

- Xilinx LogiCORE IP AXI Interconnect (v1.06.a)
- Custom AXI to AXI bridge

[1]In graph theory, a clique in an undirected graph is a subset of its vertices such that every two vertices in the subset are connected by an edge [91].

- Xilinx LogiCORE IP AXI to AXI Connector (v1.00.a)

The Xilinx AXI Interconnect may be configured as a bus or a crossbar. All data channels have the same bus width of 32 bits. Albeit the number of total crossbars, buses, and bridges provides an indication of the cost of the overall communication architecture, in order to target physical devices, such as FPGAs, we need a measure related to the technology. Synthesized FPGA designs are normally evaluated in terms of Look-Up Tables (LUTs) and Flip-Flops (FFs). Interconnect components, in particular, are usually dominated by the number of LUTs. Hence, to explore the design space efficiently, we need a technique to estimate how many LUTs are taken by a topology (without resorting to an actual synthesis). This measure clearly depends on the technology, because the internal architectures of FPGA chips can vary considerably from one family to the other.

We built accurate analytical models for the evaluation of the area cost, the latency, and the power consumption, obtained by interpolating extensive RTL synthesis results. The area model of the AXI Interconnect was obtained by synthesizing the AXI Interconnect IP core for a subset of all the possible configurations with 1 to 16 master ports and 1 to 16 slave ports with a step of 3, a data bus width of 32 bits, and the two available address bus modes (shared bus or SAMD[2]) using the Xilinx PlanAhead Design Tool. From the data collected, we extrapolated the following equations, valid for the Zynq-7000 family [7], used to calculate the area information of any interconnects with 1 to 16 master ports or 1 to 16 slave ports (16 masters/slaves is the maximum value supported by AXI Interconnect IP core).

$$A_{crossbar}\, n \times m = 101n + 60nm + 42m + 874 \tag{4.2}$$

$$A_{bus}\, n \times m = 80n + 18.75m + 95.5 \tag{4.3}$$

We considered burst-based transactions with only the start address issued and the payload transferred in a single burst that can comprise multiple beats[3] Concerning the bandwidth estimation, we considered the bandwidth of a channel as the maximum rate at which a master can receive/send (r/w) data from/to a slave. Due to the burst-based communication, we considered that address arbitration does not impact that rate[4]. To compute the execution

[2]Shared Address buses and Multiple Data buses: in most systems, the address channel bandwidth requirement is significantly less than the data channel bandwidth requirement. Such systems can achieve a good balance between system performance and interconnect complexity by using a shared address bus with multiple data buses to enable parallel data transfer [8]

[3]A beat is an individual data transfer within an AXI burst [8].

[4]Arbitration latencies typically do not impact data throughput when transactions average at least three data beats [3]

delay d_i of each communication task t_i , we used the following equation:

$$d_i = (\frac{52+10h_i}{64}) * c_i \tag{4.4}$$

where h_i is the *hop count* defined as the number of bridges crossed by a communication task t_i and c_i is the *computation cost* defined in Section 4.1. Intra-cluster communication tasks require 52 clock cycles every 64 bytes of data sent: 4 clock cycles for the initial access latency due to the phases of arbitration and handshaking on the address channels, and 3 clock cycles for each transfer of a single beat on the data channel. In case of inter-cluster communication 10 additional clock cycles are required for each traversed bridge. Obviously, these numbers are dependent both on the architecture and the used IP cores. Concerning the static power consumption, we considered the power from transistor leakage on all connected voltage rails and the circuits required for the FPGA to operate normally. Obviously this power is independent of the user design and, consequently, is affected only by voltage and temperature. Unlike static power, dynamic power is the power of the user design, i.e. it depends on the input data pattern and the design internal activity. We calculated this power by means of simulation results based on the average switching rate of every signal in the interconnect. Concerning the software framework, including the implementation of all the above scheduling algorithms, the optimization tool is implemented in Java 7. For the genetic algorithm we rely on the modular framework Opt4J [92].

4.4.2 Overview of the experiments

In order to validate the clustering choices and the cost of the corresponding communication architecture, and to demonstrate the impact of scheduling on the resulting topology, we tested the method for six synthetic benchmarks as well as a real-world application. The benchmarks were obtained using the TGFF package [38], removing possible concurrent tasks with the same master, while the application is a Canny edge detection algorithm [25] whose TL and DG are taken from [128]. Their characteristics are summarized in Table 4.2. The number of tasks in the experiments ranges from 13 to 88, while the number of masters and slaves ranges, respectively, from 5 to 16 and from 5 to 20. We chose benchmarks with a various number of masters and slaves to evaluate the effectiveness and the scalability of the proposed method for larger systems. Table 4.2 also gives the number of clusters $n_{cluster}$ found after the Communication Element Clustering phase. In addition, the table contains an index representing the completeness of the DG. It summarizes how much the design space exploration is constrained by data dependencies. The index has been derived as the ratio between the number of arcs in the DG and $(n_{task}^2 + n_{task})/2$, the maximum possible number of arcs with no cycles. The last column of the table contains the localization factor[5]. Notice that the localization factor depends only on the mapping of communication elements within local domains and hence is schedule-independent.

Table 4.2 Benchmarks Characteristics

	n_{task}	n_{master}	n_{slave}	$n_{cluster}$	**DG completeness**	**Localization factor**
App-I	36	9	6	3	0.0570	0.9460
Bench-I	13	5	5	2	0.2200	0.9697
Bench-II	25	7	8	3	0.1267	0.9178
Bench-III	31	8	10	4	0.1275	0.9267
Bench-IV	43	11	14	5	0.0731	0.8261
Bench-V	62	12	16	6	0.1100	0.7087
Bench-VI	88	16	20	8	0.1350	0.7935

We first carried out a set of experiments to evaluate the benefits of using the exploration techniques presented in this chapter and to analyze the area/latency trade-off. Area/latency trade-off points can be derived as the solutions to different constrained scheduling problems. The methodology was applied to each benchmark and application with different area constraints. The results obtained on Bench-III are discussed in Subsection 4.4.3. Here we will also give some general remarks.

In a second set of experiments, a stringent area constraint, ranging between 30% and 70% of the area of a full crossbar implementation, was fixed. Then, we compared this approach to a previously adopted method [30], used under the same area constraints, as well as to a Full Crossbar, a Hierarchical Bus, a single Shared Bus and a Network-on-Chip. Furthermore, in order to appreciate the overall impact of the scheduling algorithm on the whole methodology, we also analyzed and compared the different scheduling algorithms and the resulting solutions. In addition, we give some general remarks about power consumption. Finally, we focused on the area saving by applying the proposed methodology and the method in [30] under different area constraints able to obtain the same latency. Notice that [30] does not perform the scheduling step but only relies on aggregated parameters (i.e. the total amount of traffic exchanged between master/slave pairs) to automate the interconnection design. These results are presented in Subsection 4.4.4.

4.4.3 Exploring area/latency trade-offs

As shown by Table 4.2, the number of clusters generated depends on the complexity of the application as well as on the area constraint. For the most complex benchmark (Bench-VI), 8 clusters are derived by this procedure, while only two are required for the most simple

[5]The localization factor is a metric introduced in [115], expressing the ratio between the local traffic and the total traffic

benchmark (Bench-I). Figure 4.6 shows an example of clustering algorithm applied to

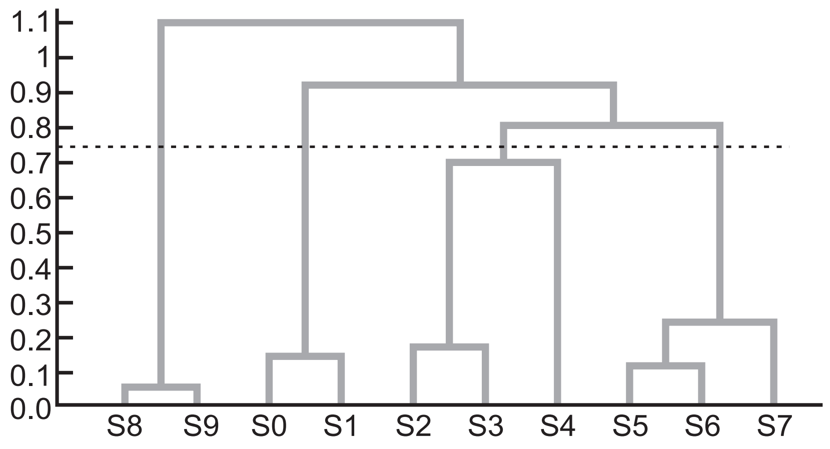


Fig. 4.6 An example of slave clustering. Slave nodes are on the x-axis, while the Euclidean Distance is on the y-axis.

Bench-III (of medium complexity) where, due to the imposed area constraints, the maximum possible value of inter-cluster Euclidean distance is 0.75, giving an outcome of 4 clusters. This result is obtained with an area constraint of 4000 LUTs. Above this value, the number of clusters starts decreasing. Furthermore, Figure 4.7 shows the schedules and the corresponding topologies obtained from the same benchmark with area constraints of 2700 LUTs and 4000 LUTs using the randomized priority-based list scheduling.

Communication tasks involving the same bridges or slaves, or exhibiting dependency relationships, never overlap. Concerning unrelated tasks, if they do not overlap, their communication interactions are serialized, so that they can share the same resources and a single communication link can be synthesized. On the other hand, when two tasks do overlap, multiple resources must be synthesized. Four clusters are connected by means of bridges that allow the traffic to be exchanged according to the requirements expressed by the Task List. The location of the bridges depends on the solution of the optimum branching problem that tries to place the clusters that exchange more data as near as possible to each other (determining less hops to go through). Notice that there are two parallel communication paths between masters inside cluster C_1 and slaves inside cluster C_0: the first goes through bridge B_{10}, while the second goes through the multi-hop path consisting of bridges B_{13} and B_{30}. The difference between the two architectures lies in the intra-cluster architectures. With 4000 LUTs we have a more parallel intra-cluster architecture achieving a gain of roughly 32% in execution time (around 11.75 Mega Clock-cycles against 8.10) due to the presence of the crossbar in cluster C_1. Instead, with an area constraint of 2700 LUT, all tasks involving M_2, M_3 and M_4 as well as S_2, S_3, S_4 are executed in a linear sequence, leading to a single bus executing all communication tasks in cluster C_1. It is important to highlight that, in this

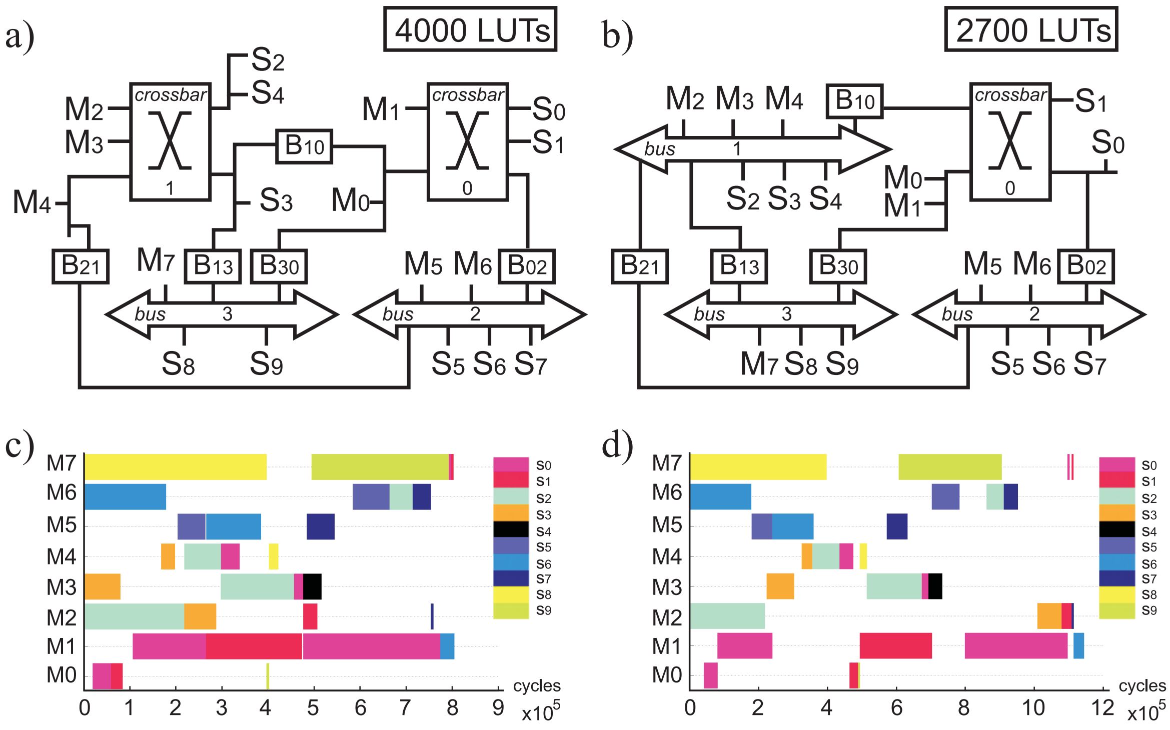


Fig. 4.7 Communication architecture descriptions and their schedule found for Bench-III with the Randomized Priority-based List Scheduling and two different Area constraints. (a) The *on-chip communication architecture* (OCA) description obtained with an area constraint of 4000 LUTs. (b) The OCA obtained with an area constraint of 2700 LUTs. (c) The *Communication Scheduling* (CS) obtained with an area constraint of 4000 LUTs. (d) The CS obtained with an area constraint of 2700 LUTs.

case, the two area constraints lead to a difference only on the intra-cluster parallelism, while the global and the inter-cluster parallelism does not change. For Bench-III we achieved a localization factor of approximately 0.92. It indicates a highly localized traffic and hence improved opportunities for global parallelism, resulting in a lower overall execution latency. Localization factors for each benchmark and application are shown in the last column of Table 4.2.

4.4.4 Comparisons with existing methods for various scheduling algorithms

To illustrate the benefits of this approach, a comparison with a previously adopted method [30] is presented. Figure 4.8 summarizes the results obtained by applying the two methodologies to all the case-studies under a fixed area constraint. We compared them also with a full crossbar, a hierarchical bus, a shared bus, and a NoC implementation. Concerning the hierarchical bus, we implemented each local domain found after the clustering step with a single shared bus and we keep the inter-cluster topology unchanged. On the other hand, a NoC implementation is more customizable than a bus-based interconnect. We considered a basic NoC implementation with a 2-D mesh topology, an oblivious minimum-path routing, and a wormhole flow control. The network channel width and flit size was set to 8 bits. Each packet in the network contains 64 body flits and 1 header flit which carries the address information. The FIFO buffers at the output ports of the router have a depth of 32 flits. The router takes 4 cycles to process the header flit [82]. After the virtual channel is acquired by the header flit, the remaining flits follow the header flit in a pipelined fashion. Furthermore, the network interface overhead due to packetization/depacketization was set to 2 cycles [19]. Then, the mapping of cores to their cross-points was done at a high level of abstraction (based on the traffic between cores) exploiting the traffic locality such as to reduce the number of router in a path. Refers to [104] for more details. Concerning the proposed approach, all the scheduling algorithms discussed in Subsection 4.3.0.2 were used, except the too slow exhaustive searches.

For each experiment, the full crossbar implementation is the unconstrained solution and, hence, it exhibits the minimum possible latency at the price of a highly oversized area, while the other points are Pareto-optimal points. For Bench-VII we were not able to synthesize a single Full Crossbar and a single Shared Bus due to the size of the benchmark[6]. Obviously, this approach shows different behaviors according to the scheduling algorithm used. The R-PBLS algorithm is able to obtain a latency that on average is only 11% larger compared to the latency obtained with a full crossbar implementation, while, due to the imposed constraints, the area ranges between 30% and 70% of the area of a full crossbar implementation. The communication architectures found with the PBLS and GA show a similar trend with an average latency overhead of respectively 16% and 15% compared with the full-crossbar implementation. This happens mainly because of the adopted clustering technique. For the small fraction of inter-cluster communication, the small degradations due to the extra clock cycles taken by bridge crossing (10 cycles every 64 bytes for each bridge) are not appreciable in the figure (scheduling involves millions of clock cycles). This means that our iterative method is capable of achieving roughly the same latency as a full crossbar despite stricter area constraints. Using the approach in [30] we have a performance degradation instead: our approach with R-PBLS scheduling found solutions that on the average lead to a latency reduction of about 33%. In fact, ignoring dependency relationships may result in a crossbar even when it is not strictly necessary as well as buses in cases where some communication

[6]The AXI IP Core can be configured to comprise maximum 16 Slave Interfaces (SI) and 16 Master Interfaces (MI) [3].

tasks can be parallelized. In other words, it may cause the serialization of tasks on the critical path, increasing the execution time. The solution points corresponding to Shared Buses, as expected, are associated with the minimum area and the largest latency (on the average 6.5× less area than a Full Crossbar with a latency overhead of about 2.6×) followed by the Hierarchical buses (on the average 4.4× less area than a Full Crossbar with a latency overhead of about 36%). The NoC implementation exhibits a different behavior. In order to keep the router cost down, the channel width was set to only 8 bits. This leads to a performance degradation. In addition we considered a 4 cycle overhead to process the header flit and a 2 cycle overhead to handle the Packetization/Depacketization. As a consequence, the NoC solutions show on the average a latency that is about 14% bigger than our approach solutions. Furthermore, there is a considerable gap between the areas of the two approaches: the NoC implementations occupy about twice the area of our approach implementations for a quarter of the channel width (32 bits against 8 bits). This is a well-known drawback of soft overlay NoCs compared to the hard implementations [56]. As an example, a 4 × 4 Hermes [101] NoC implementation with 1, 2 and 4 virtual channels occupies respectively 11511, 22036 and 50962 LUTs on a Xilinx XC2V6000 FPGA device [97].

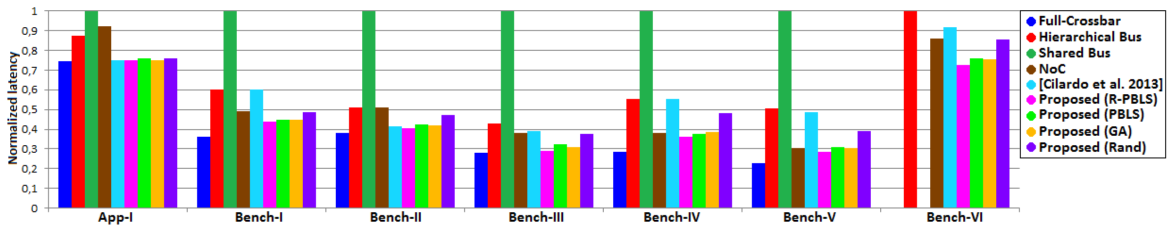


Fig. 4.8 Latency comparison. The proposed approach and [30] are used under the same area constraints

To better explain these results, we provide Table 4.3. Element (i, j) in the table represents the average percentage of time in which the communication links are used by application i using the approach j. A low value means wasted area due to low channel usage. A value equal to 1 means a communication architecture always working as in a single shared bus. Notice that the approach in [30] (as well as the use of a single crossbar architecture) may lead to an underutilized network because it does not consider dependency relationships, which of course are very likely to be found in any real application.

In addition, Table 4.4 shows the runtime of the scheduling algorithms to reach the above solutions. Obviously, the results show a dependence on the application size in terms of number of tasks. The Smart-Ex is computationally infeasible while the PBLS and hence the R-PBLS scale poorly due to their inherent complexity. On the other hand, the GA scales more smoothly as the number of tasks increases.

Table 4.3 Utilization of communication channels

	Full Crossbar	**Hier. Bus**	**Shared Bus**	**[Cilardo et al. 2013]**	**Proposed (R-PBLS)**	**Proposed (PBLS)**	**Proposed (GA)**	**Proposed (Rand)**
App-I	0.2072	0.3896	1	0.2737	0.3410	0.3410	0.3361	0.3361
Bench-I	0.5500	0.8657	1	0.5771	0.7877	0.7747	0.7716	0.7157
Bench-II	0.4383	0.7282	1	0.4484	0.5546	0.5240	0.5320	0.4754
Bench-III	0.5053	0.6452	1	0.4760	0.5496	0.4925	0.5119	0.4214
Bench-IV	0.3913	0.4500	1	0.3214	0.4872	0.4699	0.4587	0.3688
Bench-V	0.4013	0.4925	1	0.3075	0.4782	0.4399	0.4885	0.3470
Bench-VI	0.4501	0.5952	1	0.4120	0.5218	0.5062	0.5061	0.4440

Table 4.4 The runtime (*ms*) of the proposed scheduling algorithms

	Proposed (R-PBLS)	**Proposed (PBLS)**	**Proposed (GA)**	**Proposed (Rand)**	**Proposed (Smart-Ex)**
App-I	8550	250	2190	520	
Bench-I	1500	80	1610	310	523200
Bench-II	8770	240	2020	450	
Bench-III	22750	300	2110	680	
Bench-IV	87149	720	2430	1030	
Bench-V	300401	2000	3450	2060	
Bench-VI	685030	2550	3760	2300	

We also evaluated the overall energy consumption for each case-study, taking into account the power consumption (based on the Xilinx XPower power estimation tool) and the overall execution time incurred by the communication tasks. Static energy consumption, as expected, is directly proportional to the schedule latency since the results provided by XPower in terms of static power refer to the whole chip and, hence, are design-indipendent. The dynamic energy consumption, on the other hand, does depend on the specific structure of the implemented design. Figure 4.9 shows the main results in terms of dynamic energy, referring to a full crossbar, a hierarchical bus and a single bus implementations as well as a previous literature solution [30] and the presented approach. As regards our approach, there are no significant differences between the architectures found with different scheduling algorithms. The proposed method generates interconnect architectures consuming on average 28% less dynamic energy than full crossbar implementations and 36% less dynamic energy than [30]. Compared with a hierarchical bus and a single shared bus there is an higher energy consumption of respectively 15% and 77%. To understand these results, recall that dynamic power can be modelled as $P_{dynamic} = V_{DD}{}^2 \cdot \sum\limits_{n \in nets} (C_n \times f_n)$ where V_{DD} is the supply voltage, while C_n and f_n are, respectively, the capacitance and the average toggle rate (switching activity) of a net n. The number of nets, in the case of crossbar solutions, grows quadratically with the number of ports. As a consequence, a large crossbar tends to consume more dynamic power per port than an interconnect made up of small crossbars. This explains the advantage of both our approach and [30] over full crossbar implementations in terms of energy consumption. The improvement over [30], on the other hand, is due to lower latencies achieved by the communication scheduling step, as shown in Figure 4.8. In one specific case (Bench-V) [30] is worse than the full crossbar implementation, whereas our approach still performs better because of reduced-size components and improved latency.

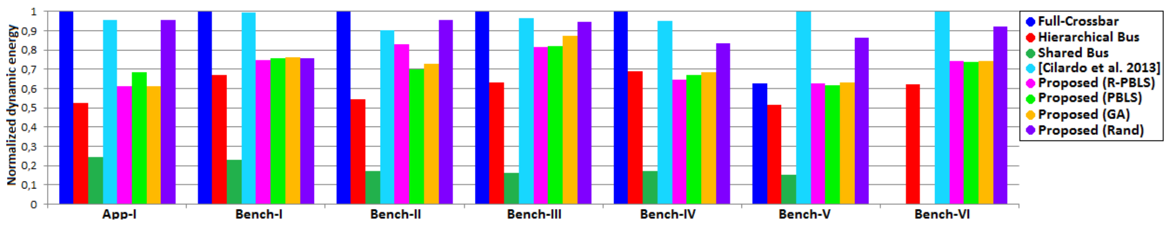


Fig. 4.9 Dynamic energy consumption comparison.

Finally, we explored the design space by varying the area constraints in the two methodologies until two architectures yielding the same latency were found. As shown in Figure 4.10, our methodology obtains an average area reduction of 43% compared to [30] under the same latency.

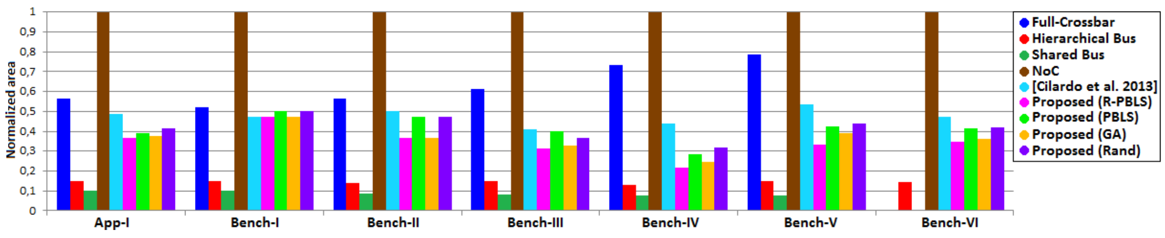


Fig. 4.10 Area comparison of interconnects yielding the the same latency

4.5 Summary

In this chapter, we presented an automated design methodology for the synthesis of complex on-chip interconnects. This approach is based on a heterogeneous topology made of crossbars, buses, and bridges. The methodology can generate a synthesizable interconnection network starting from the specification of the application requirements, including dependency relationships. The approach is based on heuristic iterative optimization algorithms. The algorithms aim at getting low-cost concurrent communication architectures by maximizing both intra- and inter-cluster communication parallelism as well as global parallelism, respectively by concentrating traffic within different local domains and by creating parallel, multi-hop inter-cluster communication paths. In particular, a greedy algorithm for clustering processing elements was introduced, capable of exploiting traffic spatial locality and creating several parallel paths among clusters. The algorithm is used in conjunction with a method for balancing the load across the bridges that connect the local domains. In addition, a novel approach to combined communication scheduling and interconnect generation is defined. Several scheduling algorithms, required for the definition of intra-cluster topologies, were analyzed and compared. Finally, a framework for the experimental evaluation of architectural solutions in terms of area/latency trade-offs is introduced. Experimental results show that this approach can synthesize designs made of dozens of cores exploiting different level of parallelisms, exhibiting encouraging improvements over previous proposals in the literature.

Chapter 5

Photonic Network-on-Chip Design

Moore's Law has enabled a restless grow in transistor integration for decades, pushing the semiconductor industry to a shift from the single-core era to a multicore paradigm. Indeed, large-scale multicore –possibly, manycore– architectures are the only solution that can stand the end of Dennard scaling and effectively face the *power wall*, today seriously limiting further technology scaling. In a large-scale multicore scenario, an energy-efficient on-chip communication fabric is the key ingredient ensuring performance scalability. While traditional electronic interconnects are constrained by physical limitations in terms of power dissipation, latency, and bandwidth [113], silicon Photonics [16] appears a promising path to energy-efficient ultra-high bandwidth on-chip communication. Nanophotonic waveguides, the photonic counterpart of a wire, can in fact achieve bandwidths in the order of terabits per second by exploiting wavelength division multiplexing (WDM), while photonic signaling consumes less power than electrical interconnects. In particular, Photonics ensures bit-rate transparency and low loss in optical waveguides, meaning that the energy consumption necessary to send a message optically is independent of the bitrate and the distance between the two end-points. Turning these potential benefits into an energy-efficient communication architecture is the main challenge faced by photonic *networks on chip* (NoCs) design.

5.1 Photonic technology

Figure 5.1 shows a high-level diagram of a basic photonic interconnect system. The lack of photonic processing and memory elements combined with the impracticality of implementing traditional store-and-forward NoCs without performing an electronic-optical (E-O) and optical-electronic (O-E) conversion in each node, inherently require an electronic layer driving the optical communication. The photonic layer is made up of four components: a laser source, a waveguide, a modulator, and a photodetector. Basically, a laser generates

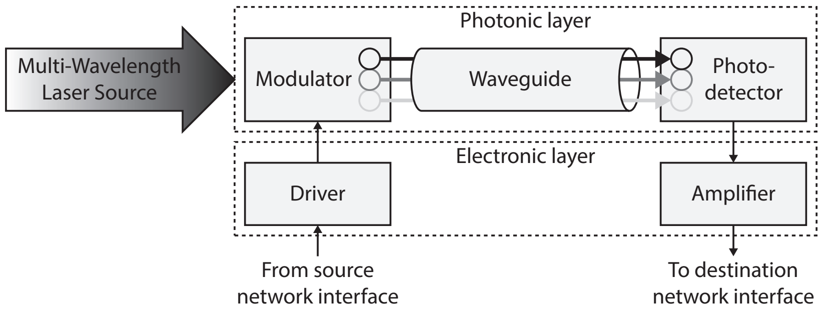

Photonic layer
Multi-Wavelength Laser Source
Modulator
Waveguide
Photo-detector
Electronic layer
Driver
Amplifier
From source network interface
To destination network interface

Fig. 5.1 A basic on-chip optical interconnect data path.

light at specific target wavelengths, which is then encoded with electronic data and transmitted through the waveguide to the photodetector, that translates the incoming optical signal into the electrical domain. Different photonic NoC configurations can be implemented as extensions of this basic system. Below we briefly analyze the building blocks of photonic architectures.

Laser source The laser source can be integrated as an on-chip device or, alternatively, it can be located outside and coupled to the chip through optical fibers. On-chip laser sources are affected by lower light emission efficiency, whereas bringing the light to the chip from the outside may incur a considerable power loss. Traditionally, light at each required wavelength is emitted by a different single-frequency laser matching a WDM channel. Recent advances have enabled broadband lasers to generate multi-wavelength light.

Silicon waveguide Waveguides are the photonic counterpart of electronic wires. Unlike wires, a single waveguide is able to carry multiple data by exploiting WDM. Optical signals are affected by power attenuation as they propagate along the waveguide.

Microring resonator A microring resonator is a waveguide forming a closed loop and having an own resonance frequency that depends on the material and the design choices. Usually, it is placed near a waveguide. When an optical signal injected into the waveguide match the wavelength of the resonance frequency, then it is coupled into the ring and steered to the *drop* port. Otherwise, the signal propagates to the *through* port. Being able to selectively add/drop a single wavelength to/from a multi-wavelength optical signal, microring resonators are used to implement filters, multiplexers, and demultiplexers. In addition, a microring can be powered on by injecting an electrical current into the p/n active regions surrounding it or by changing its temperature. It is thus possible to dynamically provide switching or modulating facilities. Figure 5.2 shows two implementations of a 2×2

photonic switching element (PSE). In a PSE, microring resonators are added at waveguide crossings to provide switching facilities.

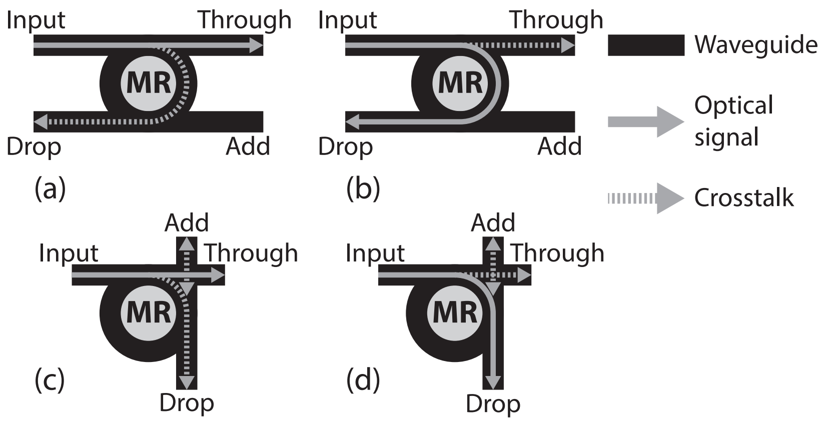


Fig. 5.2 (a) Parallel PSE in OFF state. (b) Parallel PSE in ON state. (c) Crossing PSE in OFF state. (d) Crossing PSE in ON state

Based on the basic 2×2 PSEs, complex photonic switches can be implemented. The inner architecture of photonic switches impact the performance and determines the feasibility of a photonic NoC. A photonic switch should exhibit a non-blocking behaviour and should be optimized to reduce both crosstalk and insertion loss being aware of the routing protocol. In that respect its physical layout could be optimized to reduce the number of waveguide crossings, bends, and the waveguide length. In addition it is possible to take advantage of the routing strategy. For instance, dimension-order routing results in at most a single turn between a source and a destination. Consequently, using straight default paths[1] leads to cross just a single microring resonator in ON resonance. Finally, a further optimization consists in prohibiting certain paths, such as the U turns or the turns from the Y to the X dimension in case of XY routing. As a consequence, researchers have proposed several switch architectures, shown in Figure 5.3.

Coupler Couplers enable the physical interfacing between off- and on-chip devices. An optical fiber and a silicon waveguide are connected to each other by means of a coupler. Off-chip laser sources are a representative example. Couplers can be classified in vertical and lateral. Lateral couplers have the capability of multiwavelength coupling, while vertical couplers have a reduced size and hence there are no restrictions in placing them.

[1]A default path is the path that the signal takes when all the rings are placed in a off resonance state.

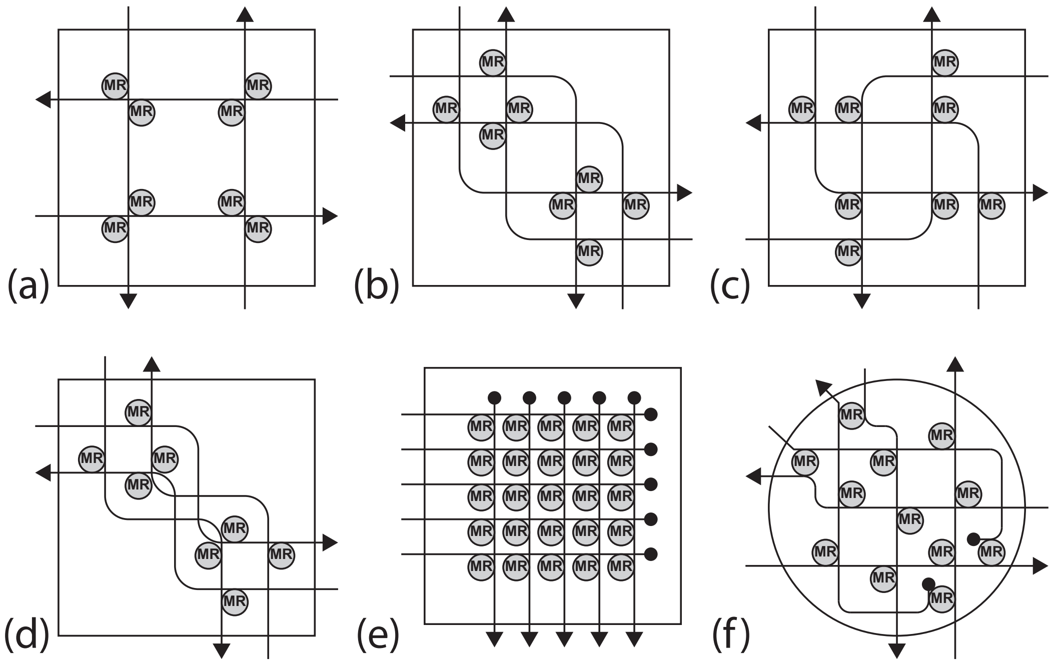


Fig. 5.3 (a) The 4 × 4 blocking PSE with straight default paths [136]. (b) The 4 × 4 non-blocking PSE [Wang et al.]. (c) The 4 × 4 non-blocking PSE optimized for insertion loss [27]. (d) The 4 × 4 non-blocking PSE with straight default paths [27]. (e) The non-blocking crossbar PSE. (f) The 5 × 5 blocking PSE optimized for insertion loss [64].

Photodetector A photodetector is necessary in order to ensure the proper conversion between the optical and electronic domains. It absorbs a single wavelength-light and generates an electronic data signal.

5.2 Wavelength selectivity

On-chip optical communication can exploit an extremely large spectral bandwidth of hundreds of nanometers. Spectrally multiplexing different optical signals into a single waveguide results in parallel wavelength channels: optical data streams are allowed to coexist on the same waveguide allocated at mutually exclusive frequencies to avoid interference. This technique is referred as Wavelength-Division Multiplexing (WDM). The resulting wavelength selectivity can be used with a twofold purpose: bandwidth aggregation is achieved by concurrently routing all the wavelength channels from a source node to a destination node, while routing facilities can be obtained by statically or dynamically assigning a wavelength to a node or to a couple of nodes. This opens up the first design dilemma:

exploit wavelength selectivity for bandwidth *aggregation or as a routing facility?*

Current approaches Figure 5.4 shows two basic photonic network configurations. In Figure 5.4.a, the wavelength selectivity is used with a bandwidth aggregation purpose: nodes are allowed to send and receive optical data streams using all the wavelength channels available. Concerning the implementation requirements, for each node, N_λ photonic microring resonators are required for both modulating and filtering purposes, where N_λ is the total number of wavelengths available. In addition, we need to take into account the microring resonators required for providing the switching facilities. In principle, a switch operating on a WDM signal requires a single ring resonator for each wavelength channel. However, microrings may support multiwavelength routing when designed with a small *free spectral range*[2] (FSR). However, the minimum achievable FSR is bounded by physical constraints such as the maximum ring diameter. Silicon broadband comb switches were demonstrated that can simultaneously switch up to 20 wavelength channels [20].

On the contrary, Figure 5.4.b shows a possible implementation of a wavelength routed network. A different wavelength is assigned to each node. A source node is able to send data on all the other wavelength channels: when it needs to send data to a destination, it uses the destination wavelength channel. Here a network with N_{nodes} nodes requires, for each node, $N_{nodes} - 1$ microring resonators as modulators and a single microring as filter.

Key insights Exploiting wavelength selectivity for bandwidth aggregation purposes provides the easiest solution for reaching ultra-high bandwidth levels, but side aspects like physical constraints and routing choices, discussed later in this chapter, must be carefully taken into account. On the contrary, wavelength routing can enable extremely short latencies since, with no conflicts, the propagation delay is simply the time of flight at the speed of light. The downside is that scaling the network size is limited by the maximum number of wavelengths available: at least N wavelengths are required for a network with N nodes. Choosing between these two possibilities is a non-trivial task. In principle, wavelength-routed networks suit the needs of QoS-sensitive applications due to their low latency and good predictability. Differently, networks with a high bandwidth aggregation are more suitable for data-intensive applications requiring large memory capacities. Usually, wavelength selectivity is used only for one of the two purposes, resulting in a lack of spectral parallelism or routing. However, in case of small-size networks, one can also exploit a subset of the to-

[2]The free spectral range of a ring resonator is the spacing between different wavelengths that resonate with the ring and is inversely proportional to the circumference of the ring.

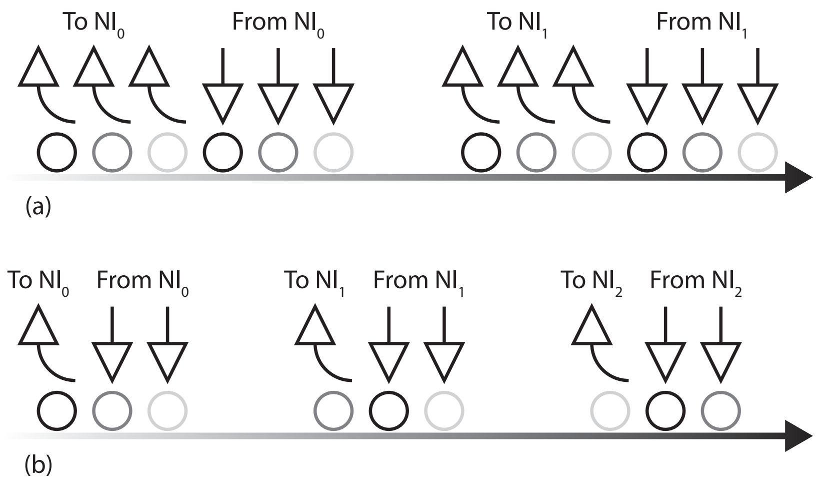


Fig. 5.4 Two basic architectures exploiting wavelength selectivity to implement: a) Bandwidth aggregation b) Routing facilities. The arrow is a waveguide, while the triangles are respectively drivers and receivers. The circles are microring resonators and each color identifies a different resonance frequency.

tal number of wavelengths for bandwidth aggregation, while the remaining wavelengths provide routing diversity.

5.3 Dealing with physical constraints

In a typical electronic design flow, the architectural level can be mostly addressed independently of the physical level relying on well-established abstraction techniques which isolate the different layers. On the other hand, Photonics introduces new electro-magnetic effects, having no electronic equivalent, that potentially impact the optical NoC architecture to a large extent. This leads to the following design dilemma:

how to abstract away physical constraints
in architecture-level design?

Below we analyze the impact of two major electro-magnetic effects, power loss and crosstalk, involved in the photonic architecture design.

Power loss

Photonic signals are subject to power attenuation: in addition to the propagation loss due to the waveguide, every other device inserted along the path introduces an additional loss usually referred to as insertion loss.

- *Propagation loss*: Propagation loss depends on several aspects including waveguide dimensions and material properties. Crystalline silicon waveguides with a thickness and a width of respectively 200-250 and 450-500 nm generate at least a propagation loss of 1-2 dB/cm [149, 157]. More exotic materials could be employed to reduce this loss up to an order of magnitude [40]. However these waveguides are not suitable for photonic NoC architectures due to the wider dimension and higher bending loss [110].

- *Bend loss*: A bend of the waveguide introduces an insertion loss proportional to the bending radius that has been measured to be 0.005 dB/90^o [157]. Generally, the insertion loss caused by waveguide bends is negligible compared to the other sources of loss.

- *Crossing loss*: A crossing happens whenever two waveguides intersect each others leading to an insertion loss between 0.05 dB and 0.15 dB [21]. Waveguide crossings are necessary due to the planar nature of the topologies achievable on silicon.

- *Drop and through losses*: A microring resonator produces a power loss that depends on its state: usually in the ON state the insertion loss (drop loss) ranges between 0.5 dB and 1.5 dB while in the OFF state (through loss) it is negligible [80]. Recently, a tradeoff between the drop and through losses was demonstrated [110]: the microring resonator can be re-engineered in order to reduce the drop loss and increase the through loss.

- *Coupler losses*: The loss due to the coupler is the attenuation that an optical signal is subject to when it is injected into the chip. Both vertical and lateral couplers generate at least a loss of 1-1.5 dB [39].

Power loss is one of the major limitations when designing a photonic NoC. First of all, the power of an optical signal must be above a certain threshold when arriving at the photodetectors in order to ensure a proper detection. As a consequence, the power injected into the chip must be higher than the photodetector sensitivity plus the worst case power loss. However, the total power cannot exceed a certain threshold due to the nonlinearities of the silicon material. In case of multiwavelength signals, this problem is exacerbated since these considerations must apply to each individual wavelength channel. In case of equal power for each wavelength channel, the above constraints yield the following inequality [28]:

$$P_{Budget}^{dB} \geq IL_{wc}^{dB} + 10\log_{10} N_{\lambda} \tag{5.1}$$

where P_{Budget}^{dB} is the optical power budget, representing the difference between the upper limit in transmission power and the photodetector sensitivity, IL_{wc} is the worst case insertion loss, and N_λ is the total number of wavelength channels. Since the power loss is highly dependent on the routing technique and the network topology, the above inequality points out the deep interplay between an electro-magnetic effect and the system-level network design.

Crosstalk

Crosstalk is caused by an unfavorable coupling between optical signals and can be generally classified in inter-message and intra-message crosstalk.

- In multihop photonic NoCs, two different optical signals can induce crosstalk noise to each other when reaching simultaneously a waveguide crossing or a photonic switch. In case of perfect coupling between two waveguides, optical signals propagate entirely with no reflection and with no crosstalk. However, ideal crossing is unfeasible and hence a small amount of optical power switches into the coupled waveguide. Figure 5.2 shows how crosstalk propagates for both the parallel and the crossing PSE in ON or OFF resonance. We refer to this crosstalk as inter-message crosstalk.

- In WDM photonic NoCs, different optical signals are allowed to concurrently travel on the same waveguide. To handle the wavelength selectivity appropriately, filtering actions are required. In case of perfect filtering, a single wavelength channel is totally separated from the entire multi-wavelength signal. However, ideal filtering is unfeasible and hence a small amount of optical power is added to other wavelength channels. We refer to this crosstalk as intra-message crosstalk.

Current approaches To contain power loss, many approaches presented in the literature introduce insertion-loss aware design techniques. Previous works proposed switches (in a multi-hop communication the whole power loss depends on the power loss of each single switch) and topologies (how the nodes are arranged impacts the lengths and insertion loss of the paths) exhibiting reduced insertion loss [29] as well as techniques optimizing the physical layouts [45] (the physical layout is responsible for the occurrence of waveguide crossings). Straight waveguides are preferred over bends and waveguide crossings are implemented only if essential. In addition, the worst case number of waveguide crossings is commonly minimized in order to reduce both power loss and inter-message crosstalk at the network level. Notice that, although some approaches aiming to reduce the crosstalk were proposed, only a few deal with architecture-level optimizations.

Key insights Unlike traditional electronic design, silicon photonic design is highly affected by physical constraints that must be exposed at the system level. Consequently, a holistic perspective that takes into account each aspects of the design is mandatory for photonic networks.

While a large body of works aim to reduce the power loss, most of them do not perform a crosstalk noise analysis. Crosstalk noise should be the main target when designing a photonic NoC architecture, since a high crosstalk noise may easily result in an inoperable architecture. A good crosstalk analysis points out the critical behavior of crosstalk noise when scaling the network size: for instance, in both a 8×8 mesh-based ONoC and a 12×12 torus-based ONoC the noise power is bigger than the signal power [109, 158]. Waveguide crossings and microring resonators are the main source of both power loss and inter-message crosstalk. A common mistake consists in believing that reducing the worst case number of waveguide crossings and microring resonators results in less power loss and crosstalk. This is not always the case since the power loss depends only on the path where the optical signal propagates, while the inter-message crosstalk depends on the other signals and the paths they take at a given time. As a result, reducing the worst case power loss is not the right design strategy if it simultaneously increases the inter-message crosstalk. In addition, a tradeoff arises when considering both power loss and crosstalk: lower power loss values lead to higher wavelength parallelism, which results in higher intra-message crosstalk. As a consequence, reducing the power loss for exploiting more spectral parallelism could be useless if simultaneously the crosstalk noise increases.

5.4 Routing domains

The routing technique should be carefully designed as it impacts all the electronic traditional metrics, i.e. latency (crossing a hop involves a latency overhead), throughput (balancing the traffic across the network reduces the congestion), power (crossing a hop involves an energy overhead) as well as on the photonic electro-magnetic metrics, i.e. power loss (crossing a hop involves a power attenuation) and crosstalk (different paths are subject to different crosstalk noises). Unlike traditional electronics that just rely on the spatial diversity, photonics can exploit both the space and wavelength domains to achieve concurrency in communication, opening up the following design dilemma:

routing in the space domain
or the wavelength domain?

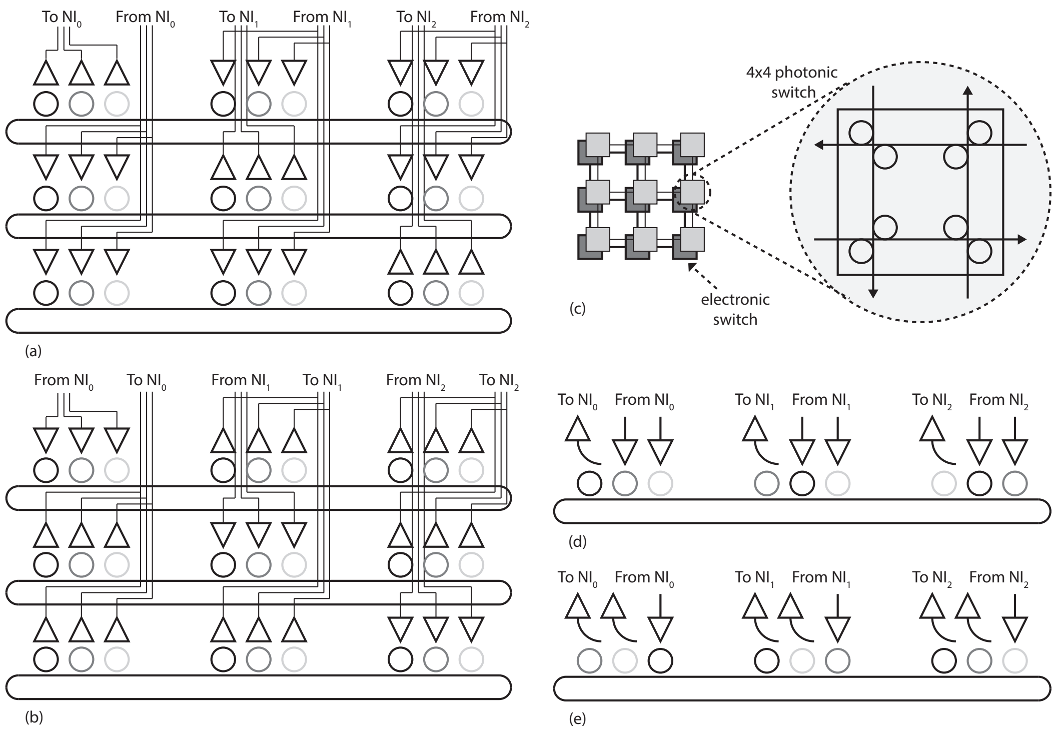


Fig. 5.5 Architectures supporting spatial routing: a) MWSR b) SWMR c) hybrid electronic-photonic circuit-switched network. Architectures supporting wavelength routing: d) source-routed bus e) destination-routed bus

Current approaches Routing in the space domain is achieved by means of different waveguides. Based on this spatial diversity, two possible architectures can be used: multi-write single-read (MWSR) (Figure 5.5.a) and dual single-write multi-read (SWMR) buses (Figure 5.5.b) or circuit-switched networks (Figure 5.5.c). Circuit-switching requires combining electronic and photonic technologies to build a hybrid network made up of two sub-networks: an electronic packet-switched network for handling control messages and a photonic circuit-switched network for data messages [136]. Topologically, circuit-switched optical NoCs are often based on standard direct topologies, such as torus schemes, because of their inherent simplicity and regularity [100, 135, 136, 161]. The implementation requirements are highly dependent from the topology and the inner architecture of the photonic switches used. Differently, in a MWSR bus, a waveguide is assigned to each destination node [145]. Waveguides usually are wrapped around in a ring-based fashion. A source node is able to send data on all the waveguides, while a destination can read data only on its own waveguide: when a source needs to send data to a destination, it uses the destination waveguide. In the same way, in a SWMR bus (Fig. 5.5b), a waveguide is assigned to each source node: a source can send data only on its own waveguide while all the destination nodes can read data on all the waveguides. In order to implement a network with N_{nodes} nodes and N_λ wavelengths, N_{nodes} waveguides and $N_{nodes} \times N_\lambda$ microring resonators are required.

On the contrary, routing in the wavelength domain is achieved by means of different wavelengths: in a network with N_{nodes} nodes, at least N_{nodes} wavelengths are required. Figure 5.5.d and Figure 5.5.e show two implementations of a wavelength routed network: the source-routed bus and its dual called destination-routed bus. In the source-routed bus, a different wavelength is assigned to each node. A source node is able to send data on all the other wavelength channels: when it needs to send data to a destination, it uses the destination wavelength channel. The destination-routed bus is the dual architecture. As in the source-routed bus, a different wavelength is assigned to each node but when a source needs to send data to a destination it uses its own wavelength channel. A source-routed bus with N_{nodes} nodes requires, for each node, $N_{nodes} - 1$ microring resonators as modulators and a single microring as filter, while the destination-routed bus requires a single microring resonator as modulator and $N_{nodes} - 1$ microrings as filters. In addition, by combining source-routed and destination-routed buses, one can assign a dedicated wavelength to each couple source-destination. This leads to wavelength crossbar architectures. Obviously, $N_{nodes} \times N_{nodes} - 1$ wavelengths and $N_{nodes} - 1$ microring resonators are required as both modulators and filters.

Key insights Optical circuit switching has many benefits. Once an optical path is established, the communication latency and the power consumption are independent of the distance between the two end points. This method requires a path-setup mechanism to allocate on-demand the necessary resources, i.e. the ring resonators along the path, to establish the end-to-end optical path. Obviously, the main drawback is the path-setup overhead that could lead to high latencies and power consumptions in case of small size messages. This issue could be partially solved via a selective transmission policy that sends electronically messages smaller than a certain threshold [46].

Concerning the wavelength domain, MWSR and SWMR buses can ideally achieve low latency, high bandwidth, and small power consumption. However, scaling the network size is limited by the number of waveguides (one for each node) and microring resonators. In addition the MWSR bus requires an arbitration step when more nodes try to communicate with the same destination, since the destination waveguide is shared between all the source nodes. In the SWMR bus the situation is worse because every destination must be arbitrated in order to proper access to the right source bus for every communication transaction.

Likewise, the two wavelength routed architectures introduce a design tradeoff: the source-routed bus takes advantage of a simplified arbitration mechanism required to avoid multiple sources to concurrently send data to a common destination, while the destination-routed bus requires an arbitration step to define the destination node of each communication interaction. At the same time, the destination-routed bus can be implemented with microring resonators having the ability to be in two states, while the microring resonators used for implementing the filters in a source-routed bus require three states: disabled, '0' and '1' bit transmissions. Consequently, the network complexity shifts from the arbitration circuit to the filter architecture. In addition, the network size is highly constrained by the maximum number of wavelength available.

Choosing between the two routing domains is a critical task. As discussed earlier, using a routing on the wavelength domain prevents exploiting the wavelengths selectivity for bandwidth aggregation purpose but with no conflicts the propagation delay is simply the time of flight at the speed of light. At the same time, the implementation requirements are usually lower compared to the alternative architectures, although scaling the network size is limited by the maximum number of wavelengths available. Small-size networks where small messages with random communication patterns are exchanged could take advantage of wavelength-routed architectures. Differently, routing in the space domain allows higher bandwidth aggregation. MWSR and SWMR buses have lower latency compared to wavelength-routed networks since the higher bandwidth results in less serialization latency. However, these architectures exhibit worst scalability issues and higher implementation costs. Circuit-switched networks are ideally more scalable, but we need to contain the crosstalk and insertion loss. The high latency and power consumption due to the path-setup overhead may be neglected in case of large data streams with static communication patterns.

5.5 A cross-cutting view of the different design challenges

As pointed in the above sections, the photonic NoC design requires to face three main "design dilemmas". First, the *wavelength selectivity* provided by WDM can be exploit either for bandwidth aggregation or as a routing facility, opening up a non-trivial design trade-off for the definition of the interconnect architecture. Second, *physical constraints* become a first-class issue for the design of photonic interconnects. Unlike the well-isolated abstraction levels available for traditional electronic design, which clearly separate the physical layer from the logic domain, silicon Photonics depends to a large extent on physical constraints that deeply impact the architectural level. Third, *spectral parallelism* –a distinctive property of Photonics– introduces a new domain in the routing design space inherently affecting the definition of the architecture. Getting a cross-cutting understanding of these trade-offs is essential for harnessing the full potential of on-chip Photonics.

Photonics allows wavelength selectivity that could be used for either bandwidth aggregation or routing facilities, while exploiting an increased spectral parallelism results in higher intra-message crosstalk, regardless of how wavelength selectivity is used. Bandwidth aggregation prevents wavelength routing, thereby forcing the space domain choice. At the same time, both intra-message crosstalk and power loss limit the spectral parallelism, since the maximum number of wavelength channels is bound by the worst case power loss and the usability of the architecture depends on the noise power. In addition, the inter-message crosstalk and power loss are prohibitive for multi-hop topologies with spatial routing.

In particular, the choice of the routing technique is so critical that the whole NoC architecture needs to change according to the routing domain. In case of space domain, one can exploit wavelength selectivity for bandwidth aggregation. Circuit switching is one of the most prominent approaches and the scalability of circuit-switched architectures essentially depends on the worst case power loss and inter-message crosstalk. Likewise, for wavelength routing the maximum number of nodes in the network is constrained by the maximum number of wavelengths and hence it ultimately depends on physical effects, i.e. the intra-message crosstalk.

5.6 Summary

Photonic on-chip networks are essential to the overcome the physical limitations of electronic interconnects. In particular, this chapter introduces the basics of silicon photonics and shows its theoretical benefits. However, although photonic interconnects are a promising answer to the requirements of low-power on-chip communication, the deep interplay of wavelength selectivity, physical constraints, and routing domain issues is very likely to impact their design and affect their practical applicability because of a number of non trivial design trade-offs. As highlighted by this chapter, getting a cross-cutting understanding of these trade-offs is an essential step for harnessing the full potential of on-chip Photonics in future computing scenarios.

Chapter 6

A Path-Setup Architecture for Exploiting Hybrid Photonic-Electronic NoCs

As previously seen, photonics is not suitable for implementing traditional store-and-forward NoCs due to its inability to store and process data without an optical-electronic-optical conversion. A promising solution consists in combining electronic and photonic technologies in order to build a hybrid network made up of two subnetworks: an electronic packet-switched network (ENoC) for handling control and short size messages and an optical circuit-switched network (ONoC) for burst messages [136]. Circuit switching requires a path-setup protocol to allocate the required resources to send a message optically. Obviously, the path-setup implies an overhead in terms of performance and costs that could be amortized sending large data messages: the setup latency is much longer than the transmission latency and the setup power consumption is extremely higher than the transmission power consumption. In addition, when two or more transmissions try to allocate the same resource, a conflict arises. Conflicts could be handled in two different ways: messages are sent using the ENoC or it is possible to try again to setup the path. Sending a large message electronically instead of optically may lead to a latency and power degradation of several orders of magnitude. As a consequence, reducing the number of large messages sent electronically is a must. Note that, once a path is setup, there are no additional costs or consumption in the electronic network. In the meanwhile, keeping an unutilized optical path set results in keeping consuming power in the optical layer and incrementing the possibility to have conflicts on the electronic network. This tradeoff raises the question of when allocate the resources and for how long for the purpose of power saving. In order to achieve photonics benefits, it is necessary to design the control network as well as the path-setup protocol so as to minimize the path-setup overhead while maximizing the number of concurrent optical paths and reducing the possibility of having conflicts. In this chapter, we propose and compare, in terms of performance and energy consumption, some path-setup architectural solutions that differ from each other in the routing algorithm, the path-setup protocol, the deadlock avoidance technique, and some implementation choices such as the number of virtual channels used. Based on this study, we propose a new power-aware path-setup protocol that is able to reduce the path-setup latency and the power consumption due to its ability to put allocated circuit on a stand-by state.

6.1 Architecture overview

The communication architecture is made up of two different layers: an electronic packet-switched network (ENoC) where short and control messages are transmitted, and a optical circuit-switched network (ONoC) for large messages or bursty traffic. Each core is connected to a smart network interface (NI) owning the necessary logic to perform selective transmission. Basically, when the head flit of a message is injected in the NI, it is buffered according to the message size: messages shorter than a predefined value are stored in a buffer and then directly send via ENoC while larger or bursty messages use a different buffer. Buffers are implemented as FIFO queues. When a burst message arrives to the head of the queue, the path-setup phase begins and, if successful, the message is sent via the ONoC. On the contrary, the electronic network is used, causing not negligible performance and power degradation. As in traditional NoCs, the basic building block of the ENoC is the electronic router. Figure 6.1(a) shows the router block diagram. Since this router is designed to handle just short and control messages, it does not require the support for large messages and, hence, it is optimized for latency and not for throughput. As a consequence, we implemented a non-pipelined router able to route in a single clock cycle a flit[1] from an input port to an output port as well as to make the necessary path-setup decisions. The network is non-interfering, meaning that two virtual networks are used in order to prevent data packets from blocking control packets. This is essential since the setup procedure must be as fast as possible in order to prevent sending large messages through the ENoC. The two networks are identical from a topological point of view, meaning that each electronic router is coupled with a *photonic switching element* (PSE) that is the basic building block of the ONoC. Figure 6.1(b) shows the five ports PSE considered in our study, called *Crux*, that was first presented by [161]. As for the electronic router, there are five bidirectional ports: the four ports corresponding to the cardinal directions and the local port connected to the O/E and E/O conversion modules. This is a blocking switch, meaning that under certain conditions

[1]A flit is the smallest unit of information recognized by the flow control method that is equal to the link width.

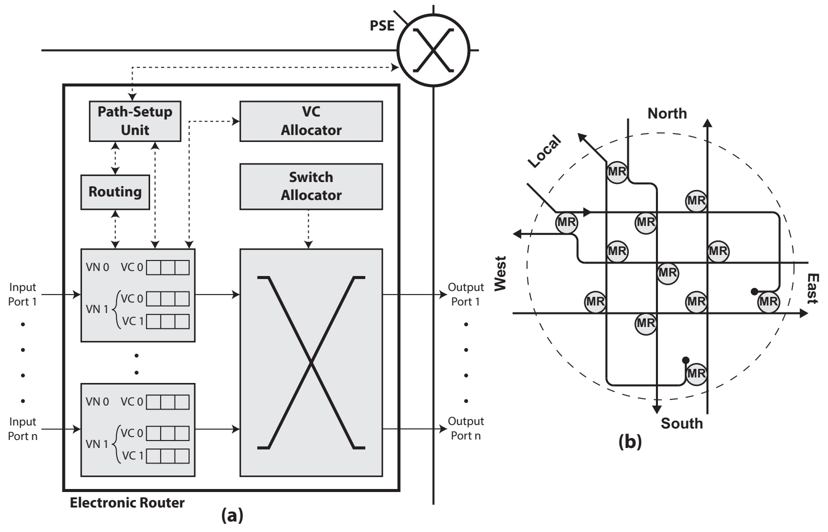


Fig. 6.1 The two building blocks of the Hybrid NoC (a) The electronic router (b)The Crux photonic switching element.

it is not possible to use simultaneously the same port as input and output (as an example the two paths *west* $\rightarrow$ *north* and *south* $\rightarrow$ *west* are mutually exclusive). Each PSE is controlled by a *Path-Setup Unit* (PSU) located in the corresponding electronic router whose role is to ensure that all the ring resonators are always set properly for a data transmission. A detailed description of the path-setup protocols and of the PSU is given in Section 6.2.

6.1.1 Optical loss and bandwidth model

Traditionally, electronic networks make use of several distinct wires in order to enhance the parallelism of a communication. Differently, by exploiting Wavelength-Division Multiplexing (WDM) each wavelength can carry different data in a single waveguide. However, in order to achieve this benefit, it is necessary to deal with the insertion loss (*IL*), which affects photonic signals when they are propagated through a waveguide. The worst case insertion loss limits the total number of wavelengths N_λ, and hence the available bandwidth, according to the inequality (5.1). In this chapter, the worst case insertion loss is evaluated according to

$$IL^{dB}_{wc} = IL^{dB}_{prop} + IL^{dB}_{cross} + IL^{dB}_{bend} + IL^{dB}_{drop} + IL^{dB}_{pass} \tag{6.1}$$

where

- $IL^{dB}_{prop} = P_{prop} \times d_{max}$ is the loss that a signal is affected by when it propagates in a straight waveguide with a length equal to d_{max}
- $IL^{dB}_{cross} = P_{cross} \times n_{cross}$ is the loss due to crossing other waveguides

- $IL^{dB}_{bend} = P_{bend} \times n_{bend}$ is the loss due to waveguide bends
- $IL^{dB}_{drop} = P_{drop} \times n_{drop}$ is the loss due to dropping into a ring
- $IL^{dB}_{pass} = P_{pass} \times n_{pass}$ is the loss due to passing by a ring

with P_{prop}, P_{cross}, P_{bend}, P_{drop}, P_{pass} the loss occurring in a single operation and n_{cross}, n_{bend}, n_{drop}, n_{pass} the number of occurrences in the worst case scenario. Concerning the Crux PSE, the maximum number of waveguide crossings and passings by a ring are all equal to four, while the maximum number of bends is three. Differently, the maximum number of droppings depends on the routing strategy. Note that the switch was designed in order to take advantage of dimension-order routing (DOR) where there is at most a single turn along the path between a source and a destination. As a consequence, straight default paths[2] are implemented and hence, using DOR, there are three droppings in the worst case scenario regardless of the network size: one for injection, one for ejection, and one for the turn. Differently, in case of other routing strategies, this value grows proportionally with the maximum number of turns and hence with the network size. Table 6.1 shows the insertion loss for single operations used for the experimental evaluation.

Once the maximum number of exploitable wavelengths is defined, it is possible to evaluate the maximum achievable bandwidth as

$$Bw = clk \times N_{\lambda} \tag{6.2}$$

where *clk* is the optical clock speed that is conservatively assumed to be 10 Ghz. Note that, as explained in Section 5.3, the maximum number of wavelength channels could be limited by electro-magnetic effects such as the inter-channel crosstalk. In addition, it is necessary to equip the microrings with the capability of multiwavlength routing and hence they must be designed with a small *free spectral range*. In this chapter we neglected both such electro-magnetic effects and physical constraints since the main focus is the design of the electronic path-setup architecture. For instance, in case of a 4×4 mesh using DOR routing, it is possible to exploit up to 221 wavelengths providing a theoretical bandwidth of 240 GB/s.

6.1.2 Energy model

In order to evaluate the energy consumption of the whole network it is necessary to consider the consumption of both the ENoC and the ONoC. Regarding the electronic power consumption, we rely on the method presented in [43] previously used in a similar work [136]. The

[2] A default path is the path that the signal takes when all the rings are placed in a off resonance state.

Table 6.1 Optical parameters

Parameter	**Value**	**Ref.**
Clock Frequency	10 Ghz	[13]
Power Budget	30 dB	[16]
Propagation Loss in Silicon	1.5 dB/cm	[157]
Waveguide Crossing	0.05 dB	[21]
Waveguide Bend	0.005 dB/90°	[157]
Dropping into a Ring	0.5 dB	[80]
Passing by a Ring	0.005 dB	[80]
Power PSE^{ON}	10 mW	[156]

method is based on the assumption that at each hop it is necessary to perform the following operations: 1) reading from the buffer; 2) taking routing and arbitration decisions; 3) crossing the inner switch; 4) going through the link, and 5) writing to a buffer. As a consequence the energy consumed by sending a message end-to-end is equal to the sum of these energy values times the number of hops.

$$E_{message} = E_{hop} \times N_{hops} \tag{6.3}$$

where N_{hops} is the number of hops taken by a message and E_{hop} is the energy necessary to cross a single hop.

$$E_{hop} = (E_{link} \times d_{link}) + E_{buffer} + E_{crossbar} + E_{static} \tag{6.4}$$

where d_{link} is the link length between two adjacent routers. Table 6.2 reports the values of the energy consumed in these operations (E_{buffer} is the sum of the two components due to the reading and the writing, while the energy due to routing and arbitration is neglected) as evaluated in [136] using the ORION simulator [151]. Note that ORION may overstate the buffer power consumption of a 40% [53]. However, the impact on the results is limited to a maximum of 4% of the total power consumption, since the buffer power consumption consists of the 10% of the whole electronic consumption.

Table 6.2 Energy consumpion for an electronic hop crossing

Parameter	**Value**
E_{link}	0.34 pJ/mm/bit
E_{buffer}	0.12 pJ/bit
$E_{crossbar}$	0.36 pJ/bit
E_{static}	0.35 pJ/bit

On the contrary, photonic signaling benefits from the two proprieties of *bit-rate transparency* and *low loss in optical waveguides*, meaning that the energy consumption necessary to transmit a message in the ONoC is independent of the bit-rate and the distance between the two end points. The power consumption of a PSE depends on its state: in the OFF state the power is negligible, while in the ON state it consumes about 10 mW [156]. This means that the energy that a message consumes depends on how long the PSEs are in the ON state regardless of whether they are used or not. As a consequence it is necessary to avoid leaving some PSEs in the ON state. As we shall see in Section 6.2, this is one of the tasks of the path setup unit.

6.2 Path-setup protocol for hybrid NoCs

The path-setup protocol is responsible for allocating all the required resources in order to guarantee that the optical communication is feasible. A basic protocol requires four types of control messages: *path-setup*, *path-ack*, *path-nack*, and *path-teardown*. When a long message or a burst arrives to the head of the buffer queue in the NI, a path-setup message is injected into the first router. The aim of this type of message is to reach the destination node. In case of success, a path-ack message is generated and sent back to the source of the communication. On the contrary, when a path-setup message reaches a router and it is unable to take a output port due to conflicts with other optical paths, a path-nack is sent back to the source. When a path-ack reaches its destination, the end-to-end optical path is ready for the communication and hence the data message is optically sent. When the optical communication ends, the resources previously allocated are released by sending a path-teardown message through the ENoC. When a path-nack arrives to a NI, it is possible either to try again the setup process or to send the message via the ENoC. In our case, we adopted a mixed policy: we try to setup the optical path for a certain amount of time after which the message is sent electronically. In this way, the time that a message can stay in the queue is bounded, avoiding blocking the subsequent messages but, at the same time, messages are not sent via the ENoC when there is a sporadic conflict. In order to implement a distributed protocol, we added in each router a Path-Setup Unit (PSU) whose functions are: 1) configure the switching functions of the corresponding PSE; 2) keep track of the PSE state; 3) manage path-setup conflicts; 4) implement a routing policy for control messages. Each PSU owns an internal PSE Connectivity Table (PCT) and a Counter Array (CA) whose structures are shown respectively in Figure 6.2(a) and (b). Note that the size of these data structures is constant for the NoC design, regardless the network size, ensuring scalability. The PCT contains for each output port the fields *state* and *input port*. In the basic path-setup protocol, there are three states: *Unused*, *Reserved*, and *Allocated*. Figure 6.2(c) shows the finite-state transition diagram for the basic path-setup protocol. At the beginning, the state is set to unused meaning that the path is free for further reservations. When a path-setup message arrives, the state changes to *reserved* and the input port from which the message

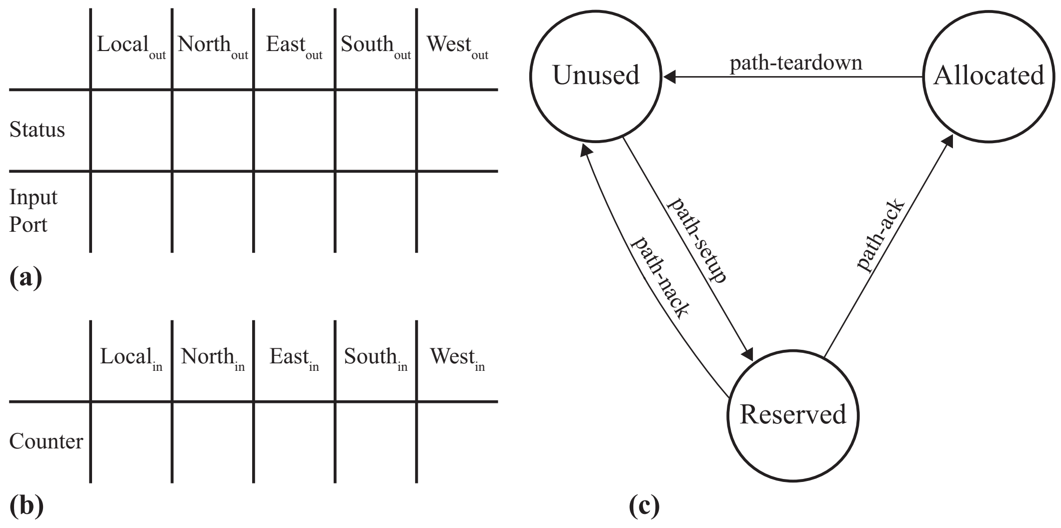


Fig. 6.2 The data structures located inside the Path-Setup Unit for the basic path-setup protocol. (a) The PSE Connectivity Table. (b) The Counter Array. (c) The finite-state transition diagram for the state of a output port in the PSE.

arrived is stored in the associated entry. In the reserved state, when a path-nack message arrives, the state comes back to *unused*, otherwise, in case of path-ack, the state changes to *allocated*. An output port in the *allocated* state implies that the necessary ring resonators are set in order to establish an optical path between the input and output ports. Hence, the PSE is ready to be used and as a consequence it begins to consume energy. When the optical communication ends, the source sends a path-teardown message in order to set the state to *unused*. A conflict arises when a message cannot be routed since all the required output ports are reserved or allocated. In that case, a path-nack message is generated and sent back to the previous router. Depending on the routing algorithm, it is possible that a path-setup message needs to be routed to more than a single output port. Assuming that the path-setup message is sent through k output ports with $k > 1$, two resulting scenarios are possible: the router will receive k path-nack messages or one path-ack and $k-1$ path-nack messages. In both cases, we need a synchronization mechanism in order to avoid leaving junk messages in the network: for each input port, a counter keeps track of how many messages coming from that input port were sent. When a path-ack or path-nack message arrives, the corresponding counter is decremented. Only when the counter is equal to 0, the path-ack or path-nack message is sent back to the previous router on the path. In this way, only one message arrives to the source NI and there are no junk messages left in the network.

6.2.1 Routing

Data messages in the ENoC are routed using XY dimension-order routing. Of course, different routing algorithms can be used, but, as our goal is to investigate on the control and photonic networks, this choice provides a simple electronic infrastructure that is enough to guarantee basic connectivity. Using the same algorithm for data and control messages could lead to poor performance: the four turns allowed by XY routing do not permit any adaptiveness in routing. In this chapter we analyzed three different routing algorithms to route path-setup messages: dimension order routing, minimal path flooding, and non-minimal path flooding. With the minimal path flooding, each router checks if it is the target router. If not, the path-setup message is sent to the neighbors that are closer to the target router. Each router repeats the process or, in case of a conflict, generates a path-nack message. Non-minimal path flooding removes the closeness restriction by allowing for each path a fixed number of non-minimal hops. Flooding requires message replication while using non-minimal routing leads to explore paths otherwise inaccessible. Both these features have the drawback of potentially increasing the power consumption as well as the network congestion perturbing the ENoC performance. At the same time, there are more possibilities to find a free path, thus avoiding multiple floodings. These choices introduce the possibility of a tradeoff between the breadth and depth of the path exploration process and the number of times the exploration is performed. On the contrary, in the photonic layer, the latency is independent of the distance between the two end-points, while using PSE with straight default paths the energy consumption depends only on the number of turns. As a consequence paths with more hops than others not necessarily lead to higher power consumption. Switching from DOR to flooding routing algorithm introduces the deadlock issue. As a consequence, the routing algorithm must be designed so as to prohibit just enough turns to break all of the cycles in the network. The drawback is that prohibiting some turns reduces the adaptiveness of the algorithm. In order to overcome this limitation, virtual channels could be used. When there is an invalid turn, the routing algorithm forces the packet to change the virtual channel, breaking the dependency cycle. If a message cannot cross a router due to a forbidden turn and the lack of available virtual channels, then a path-nack message is generated and sent back.

6.2.2 Path-setup protocol with standby

As previously described, using flooding routing algorithms not only increases the chances of finding free paths but also the number of messages in the network and hence the possibility of congestion. Our path-setup protocol with standby allows using flooding for routing path-setup messages, and in the meanwhile it reduces the number of messages in the network in order to overcome the above issues as well as to reduce the probability of conflicts in the routers. Implementing this protocol requires two new messages, i.e. *path-standby* and *path-wakeup* as well as two new states, i.e. *standby* and *woken up*. In addition, the PCT is extended by adding for each output port the field *Source Node* and *k* fields *Destination*, where *k* is the maximum number of paths, crossing this port, which can be simultaneously in the standby state. Note that, the maximum number of standby paths in all the network is equal to the number of destination nodes. Assuming standby paths uniformly distributed over the source nodes, *k* may be evaluated as $\lceil N_{dst}/N_{src} \rceil$. Figure 6.3(a) shows the modified PCT. Basically, the idea is to exploit the deterministic nature of burst communication, where a source node communicates with a destination node for a while before switching to a new destination. When a path-ack message reaches a router, in addition to the usual steps, the *ID* of the source and destination nodes are stored in the corresponding fields. Destination node fields are managed with a Least Recently Used policy in order to keep track of the recently used optical connection. When the optical communication finishes, the path is put in *standby* state by sending a path-standby message instead of tearing it down. A standby state means that the ring resonators are in OFF mode but the path is reserved for a further use without the necessity to perform again costly flooding operations. The source node that plans to use a path on standby must send a path-wakeup message and wait for a path-ack message. In case of receiving a path-nack message it has to try again sending a new path-wakeup. Path-nack messages are generated in case of conflicts. A new flooding is performed only after a path-teardown message is received or in case it is necessary to communicate with a new destination. Paths in *idle* state can be used only by the source node that allocated it for the first time. At the same time, in order to avoid source nodes keeping unutilized or standby paths, which prevent other sources in establishing optical circuits, a path-setup, not able to reach its destination, is allowed to tear down standby circuits allocated by other sources. The path-teardown crosses the network up to the source node of the standby circuit freeing in the PCT the entry related to that path. Since the tear-down message could be sent from any node along the path and not from the destination node, it is possible that a remaining piece of circuit keeps being in the standby state. In this case, this junk piece of circuit will never be used again and will teared down when necessary. Conflicts arise only when all the required output ports are reserved, allocated or woken up and hence are waiting for path-ack and path-nack messages. As a consequence, path-teardown messages will release the resources related to all the paths that are in the standby status and cross the output port where the conflict arises. In case of concurrent path-wakeup and path-teardown messages related to the same circuit, the path-wakeup will reach a router where the circuit has already been

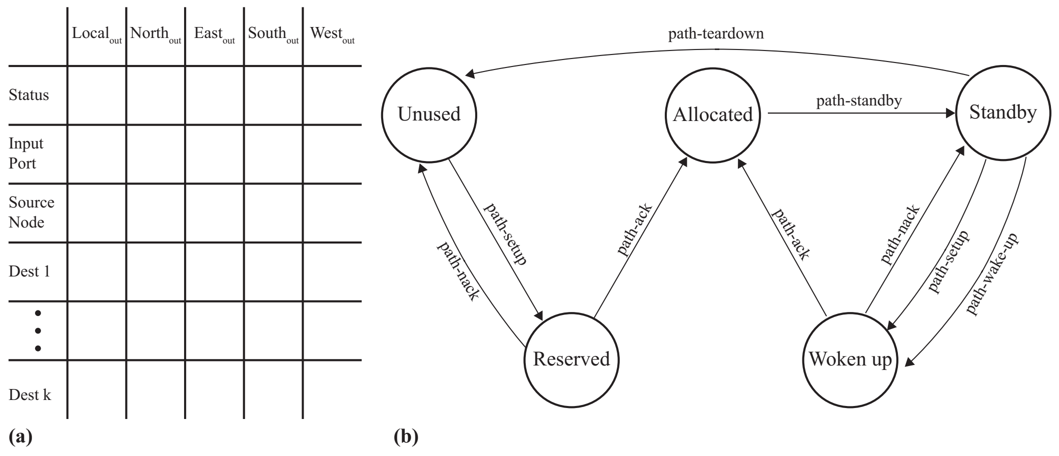


Fig. 6.3 The data structures located inside the Path-Setup Unit for the path-setup protocol with standby. (a) The PSE Connectivity Table (b) The finite-state transition diagram for the state of a output port in the PSE

teared down. At this time, a path-nack message is generated and sent to the source node. Then a new flooding will occur, since, in the meanwhile, the path-teardown message will have reached the source node. This protocol allows the network to reach a steady state where there are only a few floodings passing through a transitional state with a higher number of floodings. Figure 6.3(b) shows the finite-state transition diagram considering the new two states.

6.3 Comparison and analysis

The simulation environment is obtained by extensively modifying gMemNoCsim that is an in-house event-driven cycle-accurate NoC simulator. The optical bandwidth, the number of usable wavelengths as well as the energy consumption are evaluated by integrating the models in Section 6.1.1 and 6.1.2 into the simulator. The main simulation parameters are summarized in Table 6.3.

Non-bursty traffic is injected at a rate of 0.2 flits/cycle/NI in order to keep the ENoC busy. All the considerations in this section concern burst and control traffic. Bursty traffic is described by means of three parameters: the burst message size, the burst injection rate and the percentage of nodes that are involved in a burst transfer as sources or destinations. These nodes are randomly selected and during the simulation are changed in order to provide a fair scenario independent of the possible mapping choices. Each one is characterized by a low, medium, or high value. This traffic characterization is suitable for describing scientific or multimedia applications where communication patterns are far away from the

Table 6.3 Simulation parameters

Parameter			Value
Clock Frequency (Ghz)			1
Message Size (Bytes)			8
Injection Rate			0.2
Burst Message Size (Bytes)		Small	256
		Medium	1024
		Large	4096
Burst Injection Rate	mesh 4×4	Low	$1/80$
		Medium	$1/60$
		High	$1/40$
	mesh 8×8	Low	$1/300$
		Medium	$1/250$
		High	$1/200$
Percentage of source and destination nodes of burst traffic		Low	20% src 20% dst
		Medium	40% src 40% dst
		High	40% src 60% dst
Flit Size (Bytes)			4
Buffer Size (Flits)			6

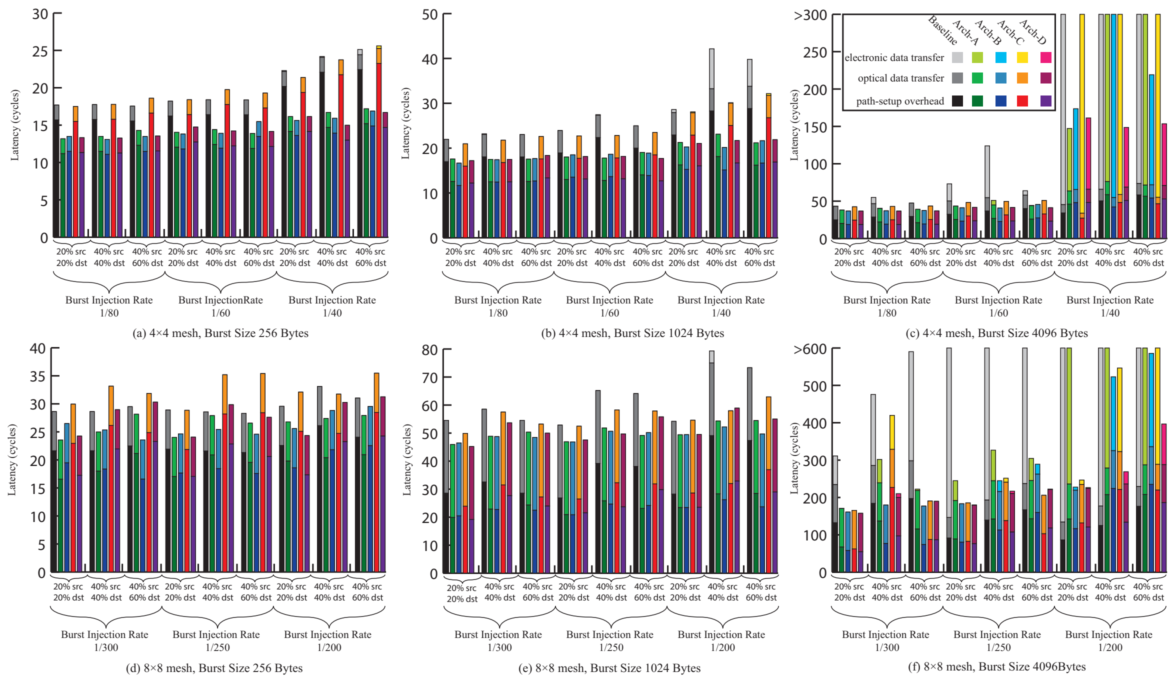
full connectivity [70]: applications that scale most efficiently to large numbers of IP Cores tend to depend on point-to-point communication patterns where the average topological degree of communication[3] ranges between three and seven distinct destinations [148]. In addition, this kind of applications may have to face challenging heavy traffic with large size messages. For instance, the $H.264$ video decoder and the sparse matrix solver applications, included in the realistic NoC traffic benchmark suite presented in [88], are respectively characterized by an average message size of 1280 and 820 bytes. Simulations are performed for 4×4 and 8×8 meshes as well as for each combination of the three values characterizing the bursty traffic. Note that 8×8 meshes, that are subject to a higher power loss than 4×4 meshes, have a smaller bandwidth. As a consequence the injection rate values for 8×8 meshes are smaller.

Table 6.4 Characteristics of the architectures analyzed

Architecture	Path-setup msg Routing	#VNs	#VCs	Stand-by optical paths
Baseline	XY	2	1	NO
Arch - A	Flooding (minimal)	2	1	NO
Arch - B	Flooding (minimal)	2	2	NO
Arch - C	Flooding (non-minimal)	2	2	NO
Arch - D	Flooding (minimal)	2	2	YES

Table 6.4 shows the characteristics of the five architectural solutions analyzed. The basic path-setup protocol is used in all the architectures excluding Arch-D, where the path-setup protocol with standby is implemented. In the baseline architecture, XY dimension-order routing is used. In order to provide a growing level of adaptiveness in routing, Arch-A and

[3]The Topological Degree of Communication is defined as the number of destinations that a given processing element must reach

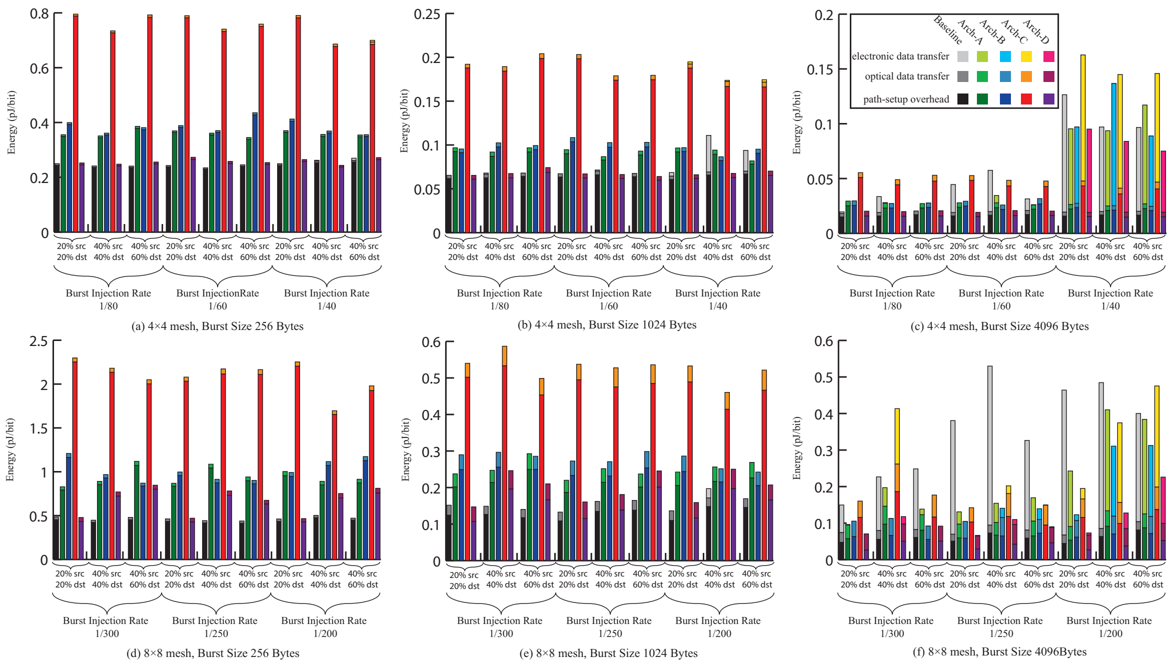


(a) 4×4 mesh, Burst Size 256 Bytes
(b) 4×4 mesh, Burst Size 1024 Bytes
(c) 4×4 mesh, Burst Size 4096 Bytes
(d) 8×8 mesh, Burst Size 256 Bytes
(e) 8×8 mesh, Burst Size 1024 Bytes
(f) 8×8 mesh, Burst Size 4096Bytes

Fig. 6.4 Average burst message latency

Arch-B are equipped with minimal flooding and in Arch-B a single forbidden turn is allowed by exploiting two virtual channels. Even Arch-C is fitted with two virtual channels and in addition can take advantage of non minimal paths with a single non minimal hop in each path. Arch-D improves Arch-B by adding the standby version of the path-setup protocol.

Figure 6.4 shows the average latency for each architecture when varying the simulation parameters values. The latency is divided into three parts: the latency due to electronic and optical data transfers and the latency due to the path-setup overhead. It is evaluated as the average of the latencies of all the burst messages that reach their destination via the ENoC as well as the ONoC. The ONoC latency includes the path-setup latency. The baseline architecture and Arch-C perform adequately for low values of the burst parameters, but when the burst size, injection rate, and percentage of nodes are increased, the latency grows faster compared to the other architectures. In the worst case, when the burst size and injection rate are very high, the network gets congested. This is definitely true for the 4×4 mesh where Arch-C cannot take advantage of the non-minimal flooding, since routing the messages to all the output ports of a router has the drawback of reserving a huge amount of resources for a small network. As a consequence, the number of path setups that can be carried out simultaneously drastically decreases. However, in the 8×8 mesh with higher values of the burst parameters, Arch-C performs better than the baseline due to the higher possibility of finding free paths. In the 4×4 mesh, Arch-A and Arch-B are performing

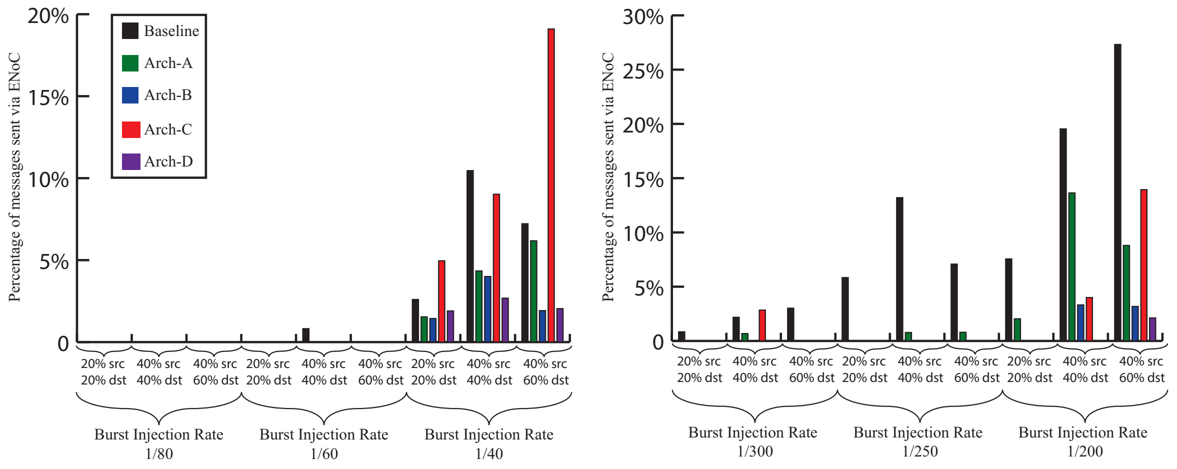


(a) 4×4 mesh, Burst Size 256 Bytes (b) 4×4 mesh, Burst Size 1024 Bytes (c) 4×4 mesh, Burst Size 4096 Bytes (d) 8×8 mesh, Burst Size 256 Bytes (e) 8×8 mesh, Burst Size 1024 Bytes (f) 8×8 mesh, Burst Size 4096Bytes

Fig. 6.5 Average burst message energy per bit consumption

between 30% and 60% better than the baseline, depending on the size of the messages and the injection rate. Basically, when these two parameters grow, the gap increases. Differently, by implementing Arch-B in an 8×8 mesh, the ability of safely crossing forbidden turns leads to an average reduction of latency of about 27% with a maximum of 60% for the high burst size. All these architectures perform poorly in case of a high number of nodes sending large-sized messages with a high injection rate. When these values are high, the network gets congested. On the contrary, Arch-D performs better than all the other architectures by exploiting the path-setup protocol with standby state: the network is never congested and only in case of the 8×8 mesh and all the burst values high, the latency begin slightly too high. In other cases, when Arch-B suffers high latencies, Arch-D is able to reach a reduction of about 50%.

Figure 6.5 shows the average energy per bit consumption of the burst messages. The energy is divided into three parts: the energy for sending via ENoC, via ONoC, and the energy due to the path-setup overhead. The energy due to the path-setup is the main component in case of small and medium sized messages: more than 95% of the whole energy in case of small size messages and more than 80% for medium size messages using the baseline architecture. When the burst size is large and the injection rate high, a number of messages are sent via the ENoC due to the congestion on the control network. This causes the increment of the ENoC energy that can become the main component even in case of a few messages. The general behavior is that Arch-C performs very poorly due to the huge amount of flooding messages sent, followed by Arch-B and Arch-A. The more messages are sent during the path-setup phase, the more power is consumed. The baseline consumes least energy since no message replication is performed. This trend does not occur in case of large messages. The path-setup overhead is independent of the size of the message and hence it is not relevant. Concerning the baseline and the Arch-C architecture, a high injection rate leads to a congestion on the control network and hence to a huge increase of the energy consumed in the ENoC. This is true for each architecture but Arch-B is able to route less messages in the ENoC. Different considerations apply to Arch-D. Its ability to put an optical circuit in a standby state allows reducing the number of floodings and hence power consumption. Compared to the baseline architecture, which does not perform floodings, it reaches the same values for small-sized messages, while for larger messages it achieves more than 30% reduction in case of a 4×4 mesh and 50% in case of a 8×8 mesh. To better explain these results, the percentage of failures in establishing the optical circuit and hence the percentage of burst messages sent via the ENoC is shown in Fig. 6.6. The graphs concern only messages with a size of 4096 bytes since, for lower sizes, the percentage is close to zero. As expected, an high percentage of failures is strictly related to high latency and power consumption.



(a) 4×4 mesh, Burst Size 4096 Bytes

(b) 8×8 mesh, Burst Size 4096Bytes

Fig. 6.6 Percentage of messages sent via the ENoC

6.4 Summary

In this chapter we have shown the importance of using an adequate path-setup protocol, routing strategy as well as control network to enable efficient use of the photonic resources in a hybrid photonic-electronic network. We have analyzed the performance and energy consumption of five different path-setup architectures. The results show that, in case of non-challenging traffic, a standard architecture with XY routing performs reasonably well. When the message sizes and the injection rates increase, the number of conflicts in establishing the circuit grows leading to performance/power degradations. In such a case, the latency could be reduced up to 60% by properly flooding the path-setup messages instead of using dimension order routing. However sending a higher number of control messages leads to increase the power consumption. In order to address this issue, we propose an energy-aware path-setup protocol able to put an optical circuit in a standby state. Exploiting this feature, it is possible to reduce the power consumption up to a 50%.

Chapter 7

H^{2}ONoC: a Hybrid Optical-Electronic NoC based on Hybrid Topology

Photonic networks proposed in the literature can be mainly divided into two classes, passive or active according to the type of the Microring Resonator (MR) used. Usually passive networks are wavelength-routed networks, while active networks are circuit-switched networks indicating the use of wavelength selectivity in order to implement respectively routing facilities or bandwidth aggregation. As seen in Chapter 5, the characteristics of these two classes induce various tradeoffs: wavelength-routed networks lead to low-latency efficient networks that do not require any arbitration while circuit-switched networks lead to scalable networks where high-bandwidth channels are obtained by multiplexing data onto many parallel wavelengths. Despite their benefits, active networks are affected by a high insertion loss, i.e. the power loss an element induces when it is inserted in an optical path, which limits the total number of wavelengths and hence the available bandwidth. By taking into account the impact of the topological choices on the insertion loss, it is possible to carefully design the communication architecture fully exploiting the extremely high bandwidth density offered by WDM. From the topological point of view, hybrid photonic-electronic active NoCs usually employ a 2D regular topology such as meshes and tori [29, 100, 135, 136, 161]. Due to their internal structure, direct topologies suffer from a high number of waveguide crossings.

In this chapter, we propose a novel hybrid optical-electronic NoC, called H^{2}ONoC. The hybrid topology used by H^{2}ONoC for the photonic layer relies on both direct and indirect schemes at different levels of the interconnect, as a solution enabling reduced insertion loss, hence maximizing the number of available wavelengths and the resulting actual bandwidth. This chapter addresses all the relevant aspects of the H^{2}ONoC architecture, including the routing algorithm, the flow control mechanism, the path-setup protocol, the electronic router and the photonic switches microarchitectures.

7.1 H^{2}ONoC Architecture

7.1.1 Overview

This chapter introduces a novel solution for active networks. Unlike existing solutions described in the previous chapters, the definition of the architecture is driven by the minimization of the insertion loss, leading to a twofold advantage: maximizing the number of usable wavelengths, and hence the bandwidth, and reducing the optical power injected into the chip. The proposed communication infrastructure relies on a hybrid architecture made up of two different layers: an electronic packet-switched network (ENoC) where small-size and control messages are transmitted, and an optical circuit-switched network (ONoC) for large size messages. H^{2}ONoC targets CMPs with a regular structure. The chip is divided in tiles and each tile consists of a processing element (PE), a memory element (ME), an electronic router (ER), and a photonic switch element (PSE). The ER and the PSE provide switching facilities respectively in the ENoC and ONoC and are coupled to each other. Inside each electronic router there is the required logic to drive the PSE facilities. From a topological point of view the two networks are different. The ENoC is based on a *concentrated mesh* where each router services a PE and a ME. Differently, the ONoC rely on a *hybrid topology* that requires 4×4 PSEs as well as high radix PSEs. Fig. 7.1 illustrates an example of the H^{2}ONoC architecture. The tiles are arranged in a 2D 8×6 grid. In each tile, an electronic router connects the adjacent tiles as well as a single PE and ME. At the same time, each photonic switch connects a PE and a ME as well as two high-radix switches. The high-radix switches provide connectivity between photonic switches in the same row or column.

Each core is connected to a smart network interface (NI) owning the necessary logic to perform selective transmission. Basically, when the head flit of a message is injected in the NI, it is buffered according to the size of the message: messages smaller than a certain value are stored in a buffer and then directly sent via ENoC, while larger messages use a different buffer. Buffers are implemented with FIFO queues. When a large-size message arrives to the head of the queue, the path-setup phase begins and, when it ends, the message is sent via the ONoC. The ENoC is non-interfering, meaning that two virtual networks are used in order to prevent data packets from blocking control packets. This is essential since the setup procedure must be as fast as possible in order to reduce the latency of large-size messages.

7.1.2 ONoC based on a hybrid topology

Hybrid topologies [126] combine features of both direct and indirect topologies in order to take benefits from both: in a $n-$dimensional hybrid topology, the nodes are organized

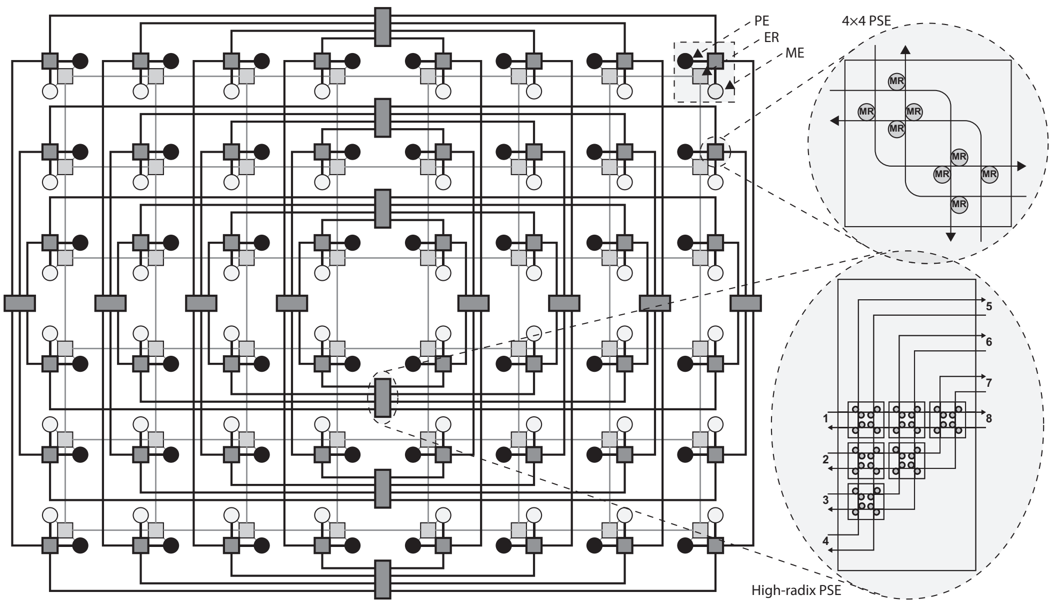


Fig. 7.1 The H^{2}ONoC architecture. The black and white circles are respectively processing and memory elements. The dark squares are photonic switch elements while the light squares are electronic routers. The dark rectangles are high-radix PSEs. The thick lines are bidirectional optical channels made up of two waveguides, while the thin lines are electronic links. The inner architecture of a PSE and a high-radix PSE are shaded and enlarged.

in a $n-$dimensional space as in direct topologies but, instead of using exclusive links between adjacent nodes, the connectivity in each dimension is achieved by small indirect networks. Bringing the hybrid topology to the on-chip photonic domain is a nontrivial task since photonics introduces new design challenges that have no electronic equivalent. H^{2}ONoC implements the optical network by means of a 2D planar hybrid topology with a gateway concentration of 2 cores. If k_i is the number of switches in the ith dimension, the number of tiles, consisting of PE/ME pairs, is given by $k_0 \times k_1$. The basic building block of the optical interconnect is the four ports photonic switch (see the scheme highlighted in Fig. 7.1), first presented in [Wang et al.]. In particular, the photonic switch is designed using only eight 1×2 PSEs which are made up of a single waveguide crossing and ring resonator. This switch has a twofold advantage compared to the other switches proposed in the literature: it is a non-blocking switch with a reasonable cost and insertion loss and it implements non-straight default paths[1]. Section 7.1.3 explains why this is essential in hybrid topologies for ONoCs. On the contrary, the indirect subnetworks are implemented by means of single high-radix switches that are based on the basic switch proposed in [135]. The basic switch consists of four 2×2 PSEs, each containing a waveguide crossing and two microring res-

[1]A default path is the path that the signal takes when all the rings are placed in a off resonance state.

onators enabling switching facilities. Each high-radix switch requires a number of basic switches equals to $\frac{n}{8}(n-2)$, where n is the number of bidirectional ports required. Compared to a multi-stage implementation, the high-radix switch has a higher system cost but leads to better performance, since multi-stage networks need to take switching decisions at each stage. Notice that the basic photonic switch and, hence, the high-radix switch are blocking in the sense that it is not always possible to use simultaneously the same port as input and output (as an example, the two paths south→east and east→north in the basic switch are mutually exclusive). Even if using blocking switches is potentially a drawback, the design complexity and, hence, the switch internal insertion loss grow exponentially with the switch size. This is the reason why we neglected this possibility.

Fig. 7.1 highlights the internal architecture of an eight-ports high-radix switch. The high-radix switches connect every switch in the same row or column. There are k_0 high-radix switches with k_1 input/output ports providing connectivity in the y dimension and k_1 radix switches with k_0 input/output ports providing connectivity in the x dimension. The high-radix switches and the gateway switches are labeled in the following way. The topology is divided in four quadrants such as in a Cartesian coordinate system where the high-radix switches act as axes. The high-radix switches are labeled by two values: the first indicates the dimension in which they provide the connectivity while the second the position in that dimension. The switches are labeled by a pair where each element gives the position in $i-$th dimension. In the following we will use (x,y) with $|x| \leq k_1/2$ and $|y| \leq k_0/2$ to indicate a switch and $[dim, pos_{dim}]$ with $dim = x,y$, $|pos_x| \leq k_1/2$ and $|pos_y| \leq k_0/2$ to indicate an indirect network. Since each photonic switch is included within a tile along with a PE, a ME, and an ER, we use the same labelling system for any element of the tile.

7.1.3 Routing

When a message is transmitted between two tiles, we need to decide which route to take. Small-size data messages sent via the ENoC make use of XY dimension order routing since its logic is easy to implement and requires little area cost. Differently, large-size messages are sent via a hybrid-topology ONoC. Unlike direct networks, in a hybrid topology the hop count is independent of the nodes distance. Considering a hypothetical communication between two cores that are connected respectively to switches sw_i and sw_j whose coordinates are (x_i, y_i) and (x_j, y_j). A minimal path consists of crossing two switches and one indirect network in case that sw_i and sw_j share a dimension $(x_i = x_j \vee y_i = y_j)$: the packet is first sent to the indirect network $[x, x_i]$ ($[y, y_i]$), and then directly to the destination switch. Otherwise, in case of $x_i \neq x_j \wedge y_i \neq y_j$, a path is made of three switches and two indirect networks: the packet is first sent to indirect network $[x, x_i]$ ($[y, y_i]$) along the Y axis (X axis),

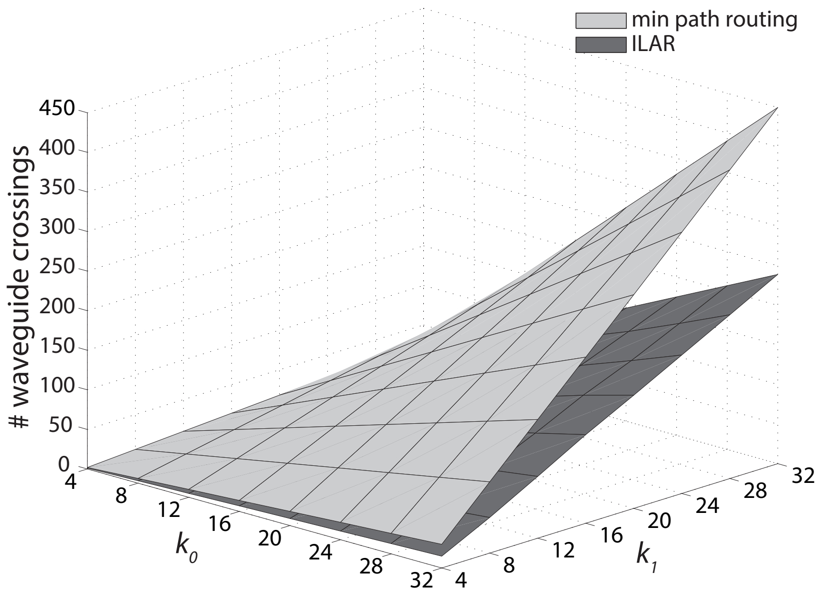


Fig. 7.2 The worst case number of waveguide crossings between two tiles with minimal path routing and ILAR

then to switch (x_i,y_j) $((x_j,y_i))$, then to indirect network $[y,y_j]$ $([x,x_j])$, and finally to the switch connected to the destination. In the electronic domain minimal paths are preferred since the performance and power consumption grow with the hop count. Differently, in photonics, latency and power consumption are independent of the distance traveled by a message. A key feature of a routing algorithm for hybrid topology consists of reducing the worst case insertion loss by avoiding using paths that cross a high number of waveguides, one of the main source of power loss. Here, we propose a routing strategy, called *Insertion Loss Aware Routing* (ILAR), that relaxes the minimal-path constraint and allows a path to cross up to three indirect networks. ILAR avoids worst case minimal paths with a large number of waveguide crossings and chooses paths that, although not minimal, cross a lower number of waveguides. Basically, in a communication between (x_i,y_i) and (x_j,y_j), if the minimal-paths cross a number of waveguides higher than a certain threshold, a packet is first sent from (x_i,y_i) to an intermediate node (x_{tmp},y_{tmp}) and then from (x_{tmp},y_{tmp}) to (x_j,y_j) using a minimal path. The intermediate node is the one between (x_0,y_{tmp}) and (x_{tmp},y_0) with $|x_0| < |x_{tmp}| \leq \frac{k_1}{2}$ and $|y_0| < |y_{tmp}| \leq \frac{k_0}{2}$ that crosses the least number of waveguides. Figure 7.2 shows that the worst case number of waveguide crossings is effectively reduced through Insertion Loss-Aware Routing. The reduction consists of about a 50% regardless of the size of the network. Details of the insertion loss and the impact of the waveguide crossing loss on the total loss are given in Section 7.2. Notice that, regardless of the routing algorithms for data messages in the ENoC and ONoC, the control messages must be routed so that they can reach every tile where a photonic switch must be set. For the inner nature of hybrid topologies, communications between switches located in different rows and columns are not allowed without going through a switch in the same row or column. As a consequence, every photonic switch that must be set can be reached via straight paths. Once the photonic switch is reached and the required resources allocated, if the message is located in the destination tile, it is sent to a local port, otherwise a single turn is necessary to reach the next switch.

7.1.4 Path-setup

In order to use the optical network it is first necessary to establish a photonic path. The path-setup protocol is responsible for allocating all the required resources in order to guarantee that the optical communication is feasible. The protocol requires four types of control messages: *path-setup*, *path-ack*, *path-nack*, and *path-teardown*. When a large-size message arrives to the head of the buffer queue in the NI, a path-setup message is created and sent to the first router. The message is made up of a single flit and contains only control information, i.e. the destination address and some additional routing details such as the address of a possible intermediate node. The aim of this kind of message is to reach the destination node. In case of success, a path-ack is generated and sent back to the source of the communication. On the contrary, when a path-setup message is blocked due to a conflict, a path-nack is sent back to the source. When a path-ack reaches its destination, the end-to-end optical path is ready for the communication and hence the data message is optically sent. When the optical communication ends, the resources previously allocated are released by sending a path-teardown message. When a path-nack arrives to a NI, we need to try again to establish an optical path.

H^{2}ONoC implements a distributed path-setup protocol by means of a Path-Setup Unit (PSU) that is embedded in each electronic router. The PSU functions are:

- to configure the switching functions of the PSE located in the same tile
- to configure the switching functions of the high-radix switches connected to the PSE located in the same tile
- to keep track of the PSE state
- to manage path-setup conflicts
- to implement a routing policy for control messages.

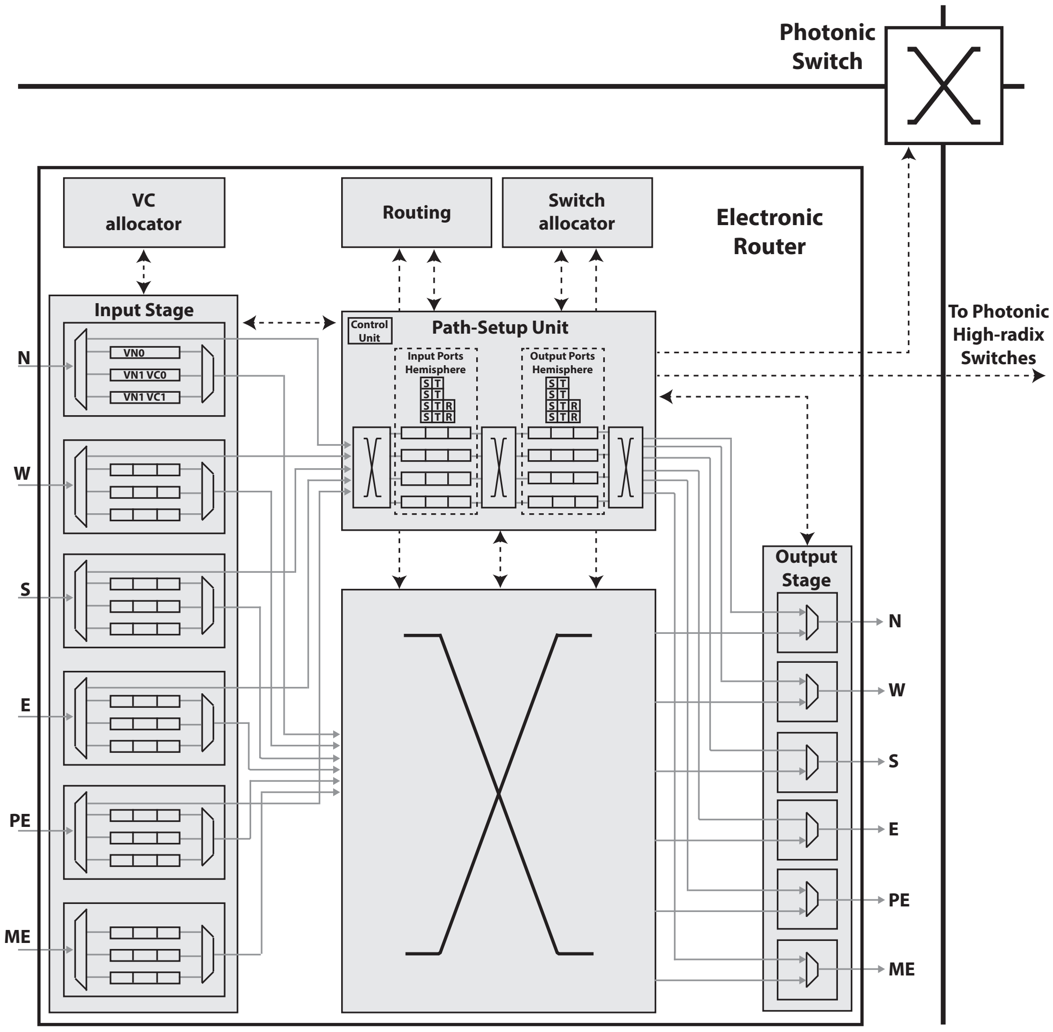


Fig. 7.3 The electronic router architecture and its path-setup unit

The PSU consists of two hemispheres necessary respectively for allocating the input and output ports of the 4×4 photonic switches as well as the output and input ports of the high-radix switch connected with it. Each hemisphere contains, for each of the four ports, a FIFO queue and a status array. The status array is made up of three fields: S, T, and R. The field S indicates the status of the port. There are three states: *Unused*, *Reserved*, and *Allocated*. The field T is a timer used to keep trace of the number of cycle a path-setup message stay at the head of the queue. In addition the status array of the ports connected to a high-radix switch contains a field R, used to allocate the ports of the high-radix switch. Note that the size of these data structures is constant for the NoC design, regardless the network size, preserving scalability. Fig. 7.3 shows the microarchitecture of an electronic router including its PSU.

At the beginning, the FIFO queues are empty and the states are set to unused meaning that the paths are free for further reservations. When a path-setup message arrives at the input stage of the router, it is necessary to check if the photonic resources must be allocated or not. A path-setup message must be processed by the path-setup unit only in the tiles where path turns take place. This is a direct consequence of using a hybrid topology. As an example, considering a communication between a PE and a ME in the two tiles (x_i, y_i) and (x_j, y_j) with $x_i \neq x_j \wedge y_i \neq y_j$. Assuming that the optical circuit to allocate is the following: $(x_i, y_i) \rightarrow [x, x_i] \rightarrow (x_i, y_j) \rightarrow [y, y_j] \rightarrow (x_j, y_j)$. As a consequence, the path-setup message must reach the three electronic routers (x_i, y_i), (x_i, y_j), and (x_j, y_j) where it is necessary to allocate the resources. In all these routers a single turn takes place. In such routers, the path-setup message is not allowed to follow the standard data path and is routed in the *Input Ports Hemisphere* of the PSU. The message is buffered according to the required input port of the photonic switch. Notice that the photonic and electronic ports are different. When the message reach the first position of the buffer the relative field of the status array are filled: the status changes to Reserved, the timer is initialized, and information necessary to allocate the high radix switch are stored in the *R* field. A message is routed to the *Output Ports Hemisphere* only when it arrives at the head of the queue. In the same way, the messages are buffered according to the photonic output port required. When a message reaches the head of this buffer all the fields of the status array are filled as in the other hemisphere. At this time the resources necessary to allocate the optical circuit are reserved and the message can be routed to the next hop. When a port is in the reserved state and a path-nack message arrives, the state goes back to unused, otherwise, in case of path-ack, the state changes to allocated. An output port in the allocated state implies that the necessary ring resonators are set in order to establish an optical path between the input and output port. Hence, the PSE is ready to be used and as a consequence it begins consuming energy due to the ring resonators in the ON resonance state. When the optical communication ends, the source sends a path-teardown in order to set the state to Unused and release the head of the buffers.

Fig. 7.4 shows an easy example. The source PE and the destination ME are located respectively in the tiles $(3,2)$ and $(1,4)$. The path-setup message goes trough the path $(3,2) \rightarrow (3,3) \rightarrow (3,4) \rightarrow (2,4) \rightarrow (1,4)$. Along this path only in three routers $(3,2)$, $(3,4)$, and $(1,4)$ the setup procedure takes place. When the message arrives to the path setup unit of router $(3,2)$, it is buffered first in the input buffer of the West port and then in the output buffer of the North port since these are the two photonic ports required in the optical circuit. The timers are initialized in order to be able to generate a path-nack message after an appropriate period of time without receiving a path-ack message. In addition the *R* entry corresponding to the output north port is filled with the requested output of the high radix photonic switch. The same steps are performed in routers $(3,4)$ and $(1,4)$. When the path-

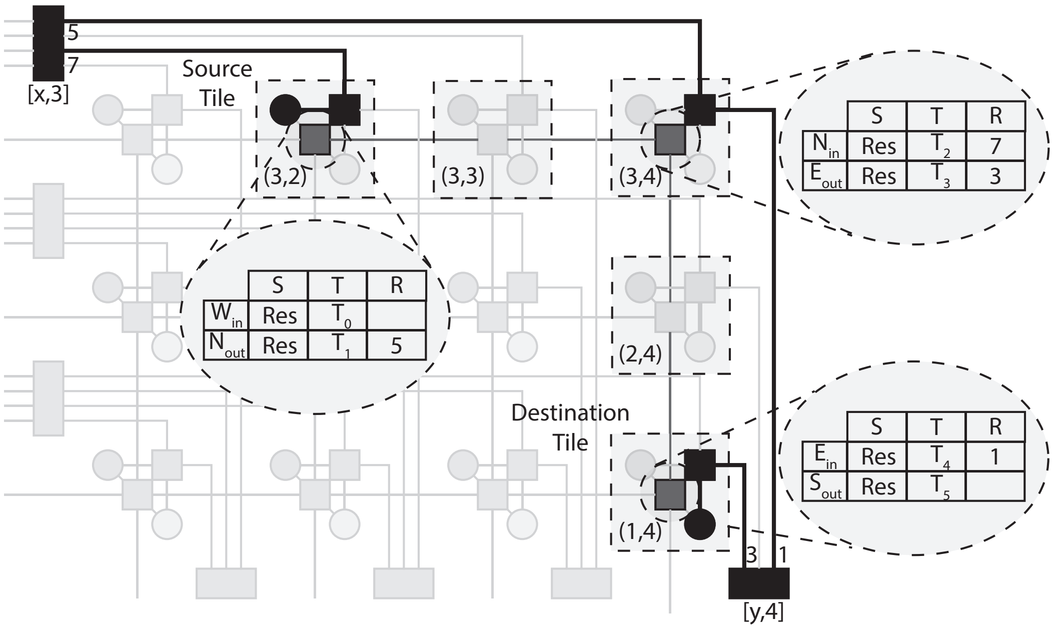


Fig. 7.4 An example of a path-setup execution. The two black circles are the source PE and the destination ME. The thin lines and the light squares are respectively the electronic links and routers used during the path-setup procedure. The thick lines, the black squares, and the black rectangles are respectively the photonic links, 4×4 switches, and high-radix switches used during the optical communication. The data structures of the path-setup unit used during the path-setup procedure are shaded and enlarged.

setup message arrives at the destination routers, a path-ack message is generated and sent back to the source router via the same route. This message will update all the entries in the path-setup units changing the status to allocated.

7.1.5 Flow Control and Deadlock Management Policy

H^{2}ONoC rely on a credit-based flow control: a counter is used to keep track of the amount of free buffer space and dedicated feedback wires are used to notify every sent message. Thanks to this inexpensive mechanism, buffer overflows are avoided. At the same time the ILAR routing is no deadlock free and therefore the routing algorithm is designed so as to prohibit just enough turns to break all of the cycles in the network. Fortunately, due to its nature, ILAR routing allows at most two turns in every path. As a consequence prohibiting just a single turn is enough to guarantee a deadlock-free algorithm. H^{2}ONoC uses two virtual channels in order to overcome this limitation. When there is an invalid turn, the routing algorithm forces the packet to change the virtual channel, breaking the dependency cycle.

7.2 Performance and Energy and Models

7.2.1 Optical Loss and Bandwidth Model

The worst case insertion loss is given by

$$IL_{wc}^{dB} = IL_{mod}^{dB} + IL_{detect}^{dB} + IL_{coup}^{dB} + IL_{top}^{dB} \tag{7.1}$$

where

- IL_{mod}^{dB} is the loss due to the electro-optic modulator
- IL_{detect}^{dB} is the loss due to the photodetector
- IL_{coup}^{dB} is the loss due to the couplers used to interface with the off-chip components such as off-chip laser sources.
- $IL_{top}^{dB} = IL_{prop}^{dB} + IL_{mod}^{dB} + IL_{bend}^{dB} + IL_{drop}^{dB} + IL_{pass}^{dB}$ is the sum of all the losses affecting a signal due to topological choices:
- $IL_{prop}^{dB} = P_{prop} \times d_{max}$ is the loss affecting a signal when it propagates in a straight waveguide with a length equal to d_{max}
- $IL_{cross}^{dB} = P_{cross} \times n_{cross}$ is the loss due to crossing other waveguides
- $IL_{bend}^{dB} = P_{bend} \times n_{bend}$ is the loss due to waveguide bends
- $IL_{drop}^{dB} = P_{drop} \times n_{drop}$ is the loss due to dropping into a ring
- $IL_{pass}^{dB} = P_{pass} \times n_{pass}$ is the loss due to passing by a ring

with P_{prop}, P_{cross}, P_{bend}, P_{drop}, P_{pass} being the loss occurring in a single operation and n_{cross}, n_{bend}, n_{drop}, n_{pass} the number of occurrences in the worst case scenario. H^{2}ONoC is designed in an insertion-loss aware fashion ensuring reduced insertion loss for minimum/medium size CMPs. ILAR routing guarantees that an optical circuit crosses at most four 4×4 photonic switches and three high-radix switches. The number of ring dropping occurrences in the worst case is equal to five: two in the source and destination 4×4 switches and one in each high-radix switch. The other two 4×4 switches are dropping-free since all the photonic switches are designed in order to ensure default paths between the two ports connected to the high-radix switches. This value does not depend on the network size, ensuring scalability. Differently, the number of waveguide crossings inside the 4×4 switch is equal to five in case of default paths and four in case of non-default paths, while inside an $n \times n$ high-radix switch it is equal to $n-2$. By adding all the components, including the number of crossings in the waveguides outside the switches, it is possible to evaluate the total worst case number of waveguide crossings that is equal to $\frac{k_0 k_1}{2} + 10$, with k_0 and k_1 equal to the number of tiles along the x and y dimension. The analytical computations behind these equations are shown in the following subsection. Since the insertion losses due to bending waveguides and passing by off-resonance rings are some orders of magnitude smaller, these components are neglected. Table 7.1 shows some insertion loss parameters for single operations that are close to currently realizable values.

Once the maximum number of exploitable wavelengths is defined, it is possible to evaluate the maximum achievable bandwidth as

$$Bw = clk \times N_\lambda \tag{7.2}$$

where clk is the optical clock speed.

Table 7.1 Insertion loss parameters

Parameter	**Value**	**Ref.**
Modulator	0.5 dB	[13]
Photodetector	0.1 dB	[13]
Coupler	1 dB	[13]
Propagation Loss in Silicon	1.5 dB/cm	[157]
Waveguide Crossing	0.05 dB	[21]
Waveguide Bend	0.005 dB/90°	[157]
Dropping into a Ring	0.5 dB	[80]
Passing by a Ring	0.005 dB	[80]

7.2.1.1 A formal study on the waveguide crossings of hybrid topologies

This section provides a formal study on the worst case insertion loss in terms of number of ring drops and waveguide crossings since these are the most significant components of power attenuation. Ring drops may happen in the gateway switches (their number is referred to as D_{sw} here) or in the switches within the indirect subnetworks (D_{in}). On the contrary, the waveguide crossings may occur within the switches (C_{sw}) or outside them. Concerning the crossings outside the switches, they are located inside the indirect subnetworks (C_{in}) and between a switch and an indirect subnetwork (C_{sw-in}).

In each 4×4 photonic switch, there are 8 microring resonators allowing each pair of ports to be connected and 10 waveguide crossings. The default paths do not require a drop into a ring and need to go through five waveguide crossings. On the contrary, the other paths require a single drop into a ring and 1 or 4 waveguide crossings. As a consequence,

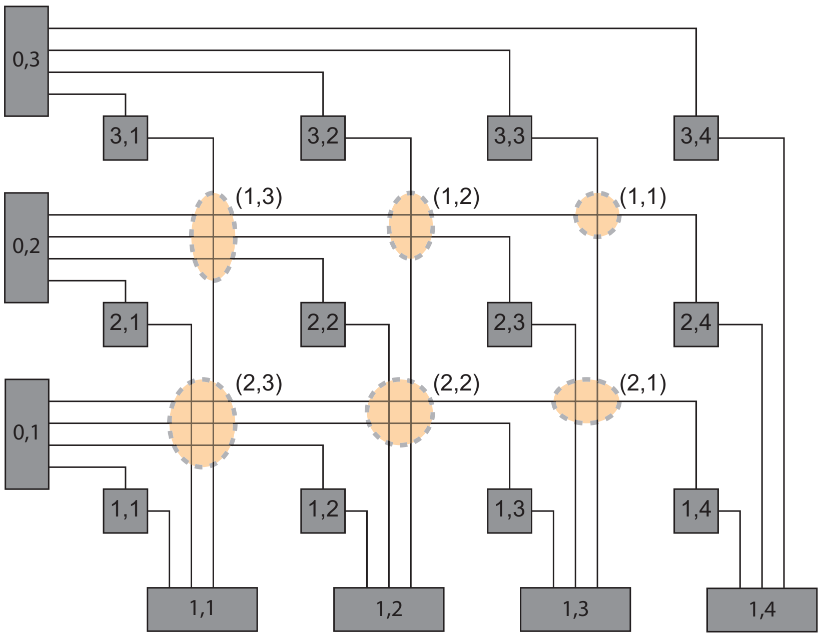


Fig. 7.5 The waveguides crossings hops for the hybrid topology in Figure 7.1

the worst case insertion loss for a single switch is 1 drop and 4 crossings.

$$D_{sw}^{wc} = 1 \qquad C_{sw}^{wc} = 4 \tag{7.3}$$

Concerning indirect subnetworks, they exhibit a different behavior according to the radix of the switch used. Here, we will consider only a single-stage implementation of the indirect subnetworks with a single high-radix switch. In a single high radix switch, the number of ring resonators is $k_i(k_i - 2)$. The worst case number of drops into a ring is directly proportional to the number of stages of the networks since at each stage a switching decision is taken. As a consequence

$$D_{in}^{wc} = 1 \tag{7.4}$$

Concerning the crossings, using high radix switches tends to concentrate the crossings within the switches rather than in the waveguides between them. As a result, the worst case number of waveguide crossings depends on the number of ports according to

$$C_{in}^{wc} = k_i - 2 \tag{7.5}$$

Concerning the waveguide crossings between gateway switches and indirect subnetworks, we introduce a new concept called *waveguide crossings hop* (H_{cross}). Usually, in the networking terminology, a hop is defined as an intermediate device through which data pass when moving from a source to a destination. A waveguide crossings hop, instead, is made of a certain number of waveguide crossings located at the intersection between two indirect subnetworks, one running along the x-dimension and the other along the y-dimension. Figure 7.5 shows the six waveguide crossings hops present in the positive quadrant of the hybrid topology in Figure 7.1. Each waveguide crossing hop may be uniquely identified by its axis coordinates. In addition, it is associated with two weights (w_0, w_1) corresponding to the number of waveguides that a signal needs to cross when moving along the x- and y-dimension. The total number of waveguides crossings of a single waveguide crossings hop is $w_0 \cdot w_1$. The weights of a hop can be calculated based on its position (x_0, y_0) as follows: $w_0 = k_1/2 - |x_0|$ and $w_1 = k_0/2 - |y_0|$. Since there are $(k_0/2-1)(k_1/2-1)$ waveguide crossing hops in each of the four quadrants, the total number of waveguide crossings over all the hops is

$$4\sum_{x=1}^{k_0/2}\sum_{y=1}^{k_1/2}\left(\frac{k_1}{2}-x\right)\left(\frac{k_0}{2}-y\right)=\frac{k_0k_1}{4}\left(\frac{k_0}{2}-1\right)\left(\frac{k_1}{2}-1\right) \tag{7.6}$$

The number of waveguide crossing hops between a gateway switch (x_{sw}, y_{sw}) and an indirect network $[dim, pos]$ along the x and y dimension are respectively $(|y_{sw}|-1)$ and $(|x_{sw}|-1)$. As a result, the number of waveguide crossings between an indirect network and a gateway switch is

$$C_{sw-in} = \begin{cases} (|y_{sw}|-1)\left(\frac{k_1}{2}-|x_{sw}|\right) & \text{if } dim = 0, \\ (|x_{sw}|-1)\left(\frac{k_0}{2}-|y_{sw}|\right) & \text{if } dim = 1. \end{cases} \tag{7.7}$$

The worst case number of waveguide crossings between a gateway switch and an indirect subnetwork is calculated by finding the maximum value of the function in Equation 7.7. Since $|x_{sw}|^{max} = k_1/2$, $|y_{sw}|^{max} = k_0/2$ and $|x_{sw}|^{min} = |y_{sw}|^{min} = 1$, then

$$C^{wc}_{sw-in} = \left(\frac{k_0}{2}-1\right)\left(\frac{k_1}{2}-1\right) \tag{7.8}$$

In a path between two switches (x_0, y_0) and (x_1, y_1) with $x_0 \neq x_1 \wedge y_0 \neq y_1$ using XY routing it is necessary to sum the four components $(x_0, y_0) \rightarrow [0, x_0]$, $[0, x_0] \rightarrow (x_0, y_1)$, $(x_0, y_1) \rightarrow [1, y_1]$ and $[1, y_1] \rightarrow (x_1, y_1)$. As a result, the number of waveguide crossings in all the waveguide crossings hops between two gateway switches is

$$\begin{aligned} C_{sw-sw}^{XY} &= (|y_0|-1)\left(\frac{k_1}{2}-|x_0|\right)+(|y_1|-1)\left(\frac{k_1}{2}-|x_0|\right) \\ &+(|x_0|-1)\left(\frac{k_0}{2}-|y_1|\right)+(|x_1|-1)\left(\frac{k_0}{2}-|y_1|\right) \\ &=(|y_0|+|y_1|-2)\left(\frac{k_1}{2}-|x_0|\right) \\ &+(|x_0|+|x_1|-2)\left(\frac{k_0}{2}-|y_1|\right) \end{aligned} \tag{7.9}$$

In the same way, in case of YX routing

$$\begin{aligned} C_{sw-sw}^{YX} &= (|y_0|+|y_1|-2)\left(\frac{k_1}{2}-|x_1|\right) \\ &+(|x_0|+|x_1|-2)\left(\frac{k_0}{2}-|y_0|\right) \end{aligned} \tag{7.10}$$

In Equation 7.9 and Equation 7.10 the first addend provides the number of waveguide crossings when moving in the x-dimension while the second in the y-dimension. In case of $x_0 = x_1$ ($y_0 = y_1$) the equation is reduced by removing the respective addend as follows.

$$C_{sw-sw} = \begin{cases} (|y_0|+|y_1|-2)\left(\frac{k_1}{2}-|x_0|\right) & \text{if } x_0 = x_1, \\ (|x_0|+|x_1|-2)\left(\frac{k_0}{2}-|y_0|\right) & \text{if } y_0 = y_1. \end{cases} \tag{7.11}$$

The worst case number of waveguide crossings between two endpoint gateway switches is calculated by finding the maximum value of the functions in Equation 7.9, Equation 7.10, or Equation 7.11.

$$C_{sw-sw}^{wc} = \frac{k_0 k_1}{2} - k_0 - k_1 + 2 \tag{7.12}$$

The worst case paths are the following:

$$\text{XY routing:}\begin{cases}\left(\pm x_i,\pm\frac{k_0}{2}\right)\rightarrow\left(\pm\frac{k_1}{2},\pm 1\right) & \forall x_i\neq\frac{k_1}{2},\\ \left(\pm 1,\pm\frac{k_0}{2}\right)\rightarrow\left(\pm\frac{k_1}{2},\pm y_i\right) & \forall y_i\neq\frac{k_0}{2}.\end{cases}$$

$$\text{YX routing:}\begin{cases}\left(\pm\frac{k_1}{2},\pm y_i\right)\rightarrow\left(\pm 1,\pm\frac{k_0}{2}\right) & \forall y_i\neq\frac{k_0}{2},\\ \left(\pm\frac{k_1}{2},\pm 1\right)\rightarrow\left(\pm x_i,\pm\frac{k_0}{2}\right) & \forall x_i\neq\frac{k_1}{2}.\end{cases}$$

$$\text{X routing:}\left(\pm 1,\pm\frac{k_0}{2}\right)\rightarrow\left(\pm 1,\mp\frac{k_0}{2}\right)$$

$$\text{Y routing:}\left(\pm\frac{k_1}{2},\pm 1\right)\rightarrow\left(\mp\frac{k_1}{2},\pm 1\right)$$

By summing up all the partial waveguide crossings, we can obtain the worst case number of waveguide crossings between two cores

$$C^{wc}=C^{wc}_{sw-sw}+\sum_{dim}C^{wc}_{in}+\sum_{sw}C^{wc}_{sw}=\frac{k_0k_1}{2}+10 \tag{7.13}$$

Since C^{wc} is a sum of several terms and the one with the largest growth rate is C^{wc}_{sw-sw}, which depends on the path, the routing algorithm directly impacts the total insertion loss. As an example, consider the two possible paths between switches $(1,4)$, $(3,1)$ in Fig. 7.5(a). By using XY routing $C_{sw-sw}=12$, while in case of YX routing $C_{sw-sw}=0$. More generally, when $C^{XY}_{sw-sw}>C^{YX}_{sw-sw}$ the YX routing provides less insertion loss than XY routing. This happens when

$$(x_0y_1-x_1y_0)+(x_1-x_0)+(y_0-y_1)<0 \tag{7.14}$$

Switching between XY and YX will reduce the C^{wc}_{sw-sw} between two switches (x_0,y_0) and (x_1,y_1) with $x_0\neq x_1\wedge y_0\neq y_1$ as follows

$$C^{wc}{}_{\substack{sw-sw\\ x_0\neq x_1\wedge y_0\neq y_1}}=\frac{k_0k_1}{2}-\frac{3}{2}\max(k_0,k_i)-2\min(k_0,k_1)+7 \tag{7.15}$$

The worst case paths are

$$\begin{cases}\left(\pm 1,\pm\frac{\min(k_0,k_1)}{2}-1\right)\leftrightarrow\left(\pm 2,\pm\frac{\min(k_0,k_1)}{2}\right)\text{if } k_0\leq k_1\\ \left(\pm\frac{\min(k_0,k_1)}{2}-1,\pm 1\right)\leftrightarrow\left(\pm\frac{\min(k_0,k_1)}{2},\pm 2\right)\text{if } k_0\geq k_1\end{cases}$$

This reduction does not apply to the case $x_0=x_1\vee y_0=y_1$, and hence the worst case is still unchanged.

In order to reduce C^{wc}_{sw-sw} for every possible case, it is necessary to use a non-minimal routing. This is the reason why we propose the *Insertion Loss Aware Routing* (ILAR) routing strategy that, allowing path crossing up to three indirect subnetworks, reduce C^{wc}_{sw-sw} of around 50% regardless of the size of the network.

7.2.2 Energy Model

In order to evaluate the energy consumption of the whole network it is necessary to consider the consumption of both the ENoC and the ONoC. Regarding the electronic power consumption, we rely on the method presented in [43] previously used in a similar work [136]. The method is based on the assumption that at each hop it is necessary to perform the following operations: 1) reading from the buffer; 2) taking routing and arbitration decisions; 3) crossing the inner switch; 4) going through the link; and 5) writing to a buffer. As a consequence the energy consumed by sending a message end-to-end is equal to the sum of these energies times the number of hops.

$$E_{message} = E_{hop} \times N_{hops} \tag{7.16}$$

where N_{hops} is the number of hops passed through by a message and E_{hop} is the energy necessary to cross a single hop.

$$E_{hop} = (E_{link} \times d_{link}) + E_{buffer} + E_{crossbar} + E_{static} \tag{7.17}$$

Table 7.2 reports the values of the energy consumed in these operations (E_{buffer} is the sum of the two components due to reading and writing, while the energy due to routing and arbitration is neglected) as evaluated in [136].

Table 7.2 Energy consumpion for an electronic hop crossing

Parameter	Value
E_{link}	0.34 pJ/mm/bit
E_{buffer}	0.12 pJ/bit
$E_{crossbar}$	0.36 pJ/bit
E_{static}	0.35 pJ/bit

On the contrary, photonic signaling benefits from the two proprieties of *bit-rate transparency* and *low loss in optical waveguides*, meaning that the energy consumption necessary to transmit a message in the ONoC is independent of the bit-rate and the distance between the two end points. The power consumption of a PSE depends on its state: in the OFF state the power is negligible, while in the ON state it consumes about 10 mW [156]. This means that the energy that a message consumes depends on how long the PSEs are in the ON state regardless of whether they are used or not. As a consequence, one should avoid leaving PSEs in their ON state.

7.3 Experimental Evaluation

7.3.1 Simulation Setup

The simulation environment is obtained by extending and modifying an in-house event-driven cycle-accurate NoC simulator. The optical bandwidth, the number of usable wavelengths as well as the energy consumption are evaluated by integrating the models in Section 7.2 into the simulator. The main simulation parameters are summarized in Table 7.3. We targeted a 32 nm process technology, and we assumed a $400mm^2$ CMP die area. Regarding the ENoC, all the data channels have the same bus width of 32 bits, equal to a single flit. The routers are configured to have a buffer depth of 6 flits. The electronic components in the network are clocked at 1.0 GHz, while the optical clock speed is conservatively set to 10 Ghz as in [13]. According to [16], a 30 dB optical power budget and a 10 Gb/s fixed modulation rate per wavelength are assumed. In addition the laser sources are assumed to be off-chip and, as a consequence, their power consumption is neglected.

Table 7.3 Simulation parameters

Parameter	Value
Chip size (mm^2)	20×20
Flit Size (Bytes)	4
Buffer Size (Flits)	6
Electronic Clock Frequency (Ghz)	1
Optical Clock Frequency (Ghz)	10
Optical Power Budget (dB)	30
Modulation rate (Gb/s)	10

In order to test our architecture and the achievable performance/power consumption gains, we first compared H^2ONoC first with an electronic mesh-based NoC and then with two previously proposed hybrid photonic-electronic active NoCs. The comparison was done by means of three synthetic benchmarks where messages are generated according to a uniform distribution. The three benchmarks differ in the message size distributions, as shown in Fig. 7.6. The distributions were chosen in order to represent different realistic traffic workloads. Bench I is characterized by a communication pattern with a large number of small-size messages normally distributed around 8 bytes with a small variance and a few large-size messages normally distributed around 2 kB with a high variance. Differently, in Bench II, the messages have an average medium size of 256 Bytes and smaller size messages are generated with a higher probability. Bench III is the opposite of Bench II since the

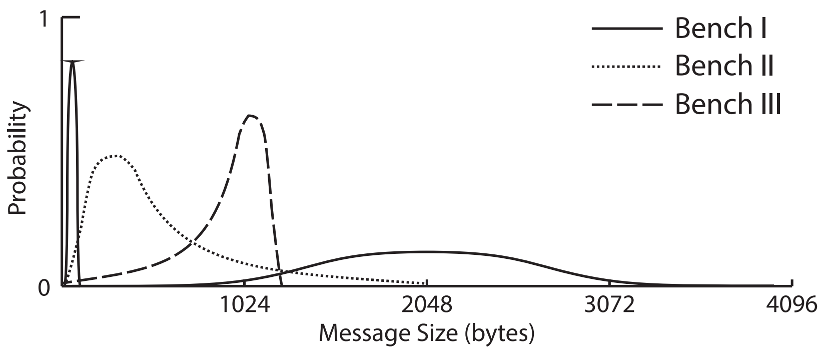


Fig. 7.6 The message sizes distributions for the three benchmarks.

average size is around 1 kB and large-size messages are generated with a higher probability compared to small-size messages.

7.3.2 Analyzing the Path-setup overhead

Optical circuit switching has many benefits. Once an optical path is established, the communication latency and the power consumption are independent of the distance between the two end points. In addition the wavelengths selectivity can be used with a bandwidth aggregation purpose. However, the path-setup protocol introduces an overhead in terms of both latency and power consumption that must be carefully evaluated when deciding to send a message optically: using the ONoC to send messages with a size smaller than a certain threshold leads to a power waste and a latency increase. Fig. 7.7 shows the path-setup contribution to the latency and power consumption for sending optically a message for various message sizes. The path-setup overhead is constant, independently of the message size, while the latency and power consumption due to the optical signaling increase with the message size. As an example, the latency and power overhead for sending a 256 bytes message optically are respectively more than 90% and 85%, while in case of 64 kB message size the overhead drops to respectively 18% and 9%.

7.3.3 Comparisons with Electronic Mesh

We first carried out a set of experiments to evaluate the benefits of using the H^{2}ONoC architecture compared to a traditional electronic mesh-based network-on-chip (ENoC). For the comparison we rely on a state-of-the-art ENoC with a link width of a single flit, a buffer depth of six flits, and a three-stage pipelined router. The results in terms of latency and energy consumption for an 8×8 CMP in absence of congestion are summarized in Fig. 7.8. As expected, the latency of the ENoC is extremely dependent on the message

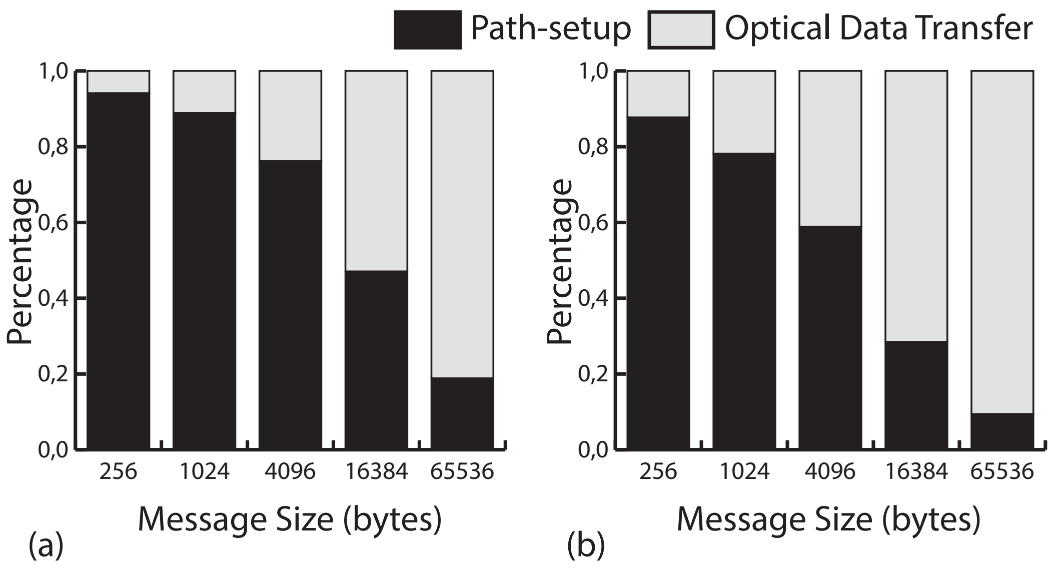


Fig. 7.7 The path-setup overhead for sending optically a message in term of (a) latency and (b) power consumption for varying message sizes.

size and hence sending large-size messages on the ENoC will eventually lead to congesting the network. Differently, the latency of H^2ONoC is almost message size insensitive. The main source of latency depends on the path-setup overhead that is constant regardless of the message size. Similar considerations can be made for the energy consumption: optical signaling provides a large bandwidth, enabling a considerable performance per watt. The power consumption is made up of two components, the first due to the path-setup overhead while the second depends on the number of ring resonators in the ON state. In H^2ONoC, both the two components are not affected by the message size and the distance between the two end points. Obviously, this is not true for the ENoC leading to extremely high energy consumption for large-size messages.

7.3.4 Comparisons with Hybrid Optical-Electronic Mesh and Torus NoC

In this section, a comparison with two previously proposed hybrid photonic-electronic active architectures is performed. The two NoCs are the torus-based hybrid NoC presented in [161] and the mesh-based hybrid NoC presented in [100]. In order to provide a fair comparison, the two architectures were implemented in our simulator according to the description provided in the respective papers. Fig. 7.9 shows for each topology the maximum possible network-level insertion loss for varying the CMP size. The insertion loss is represented as the breakdown of the three main components due to the propagation on the

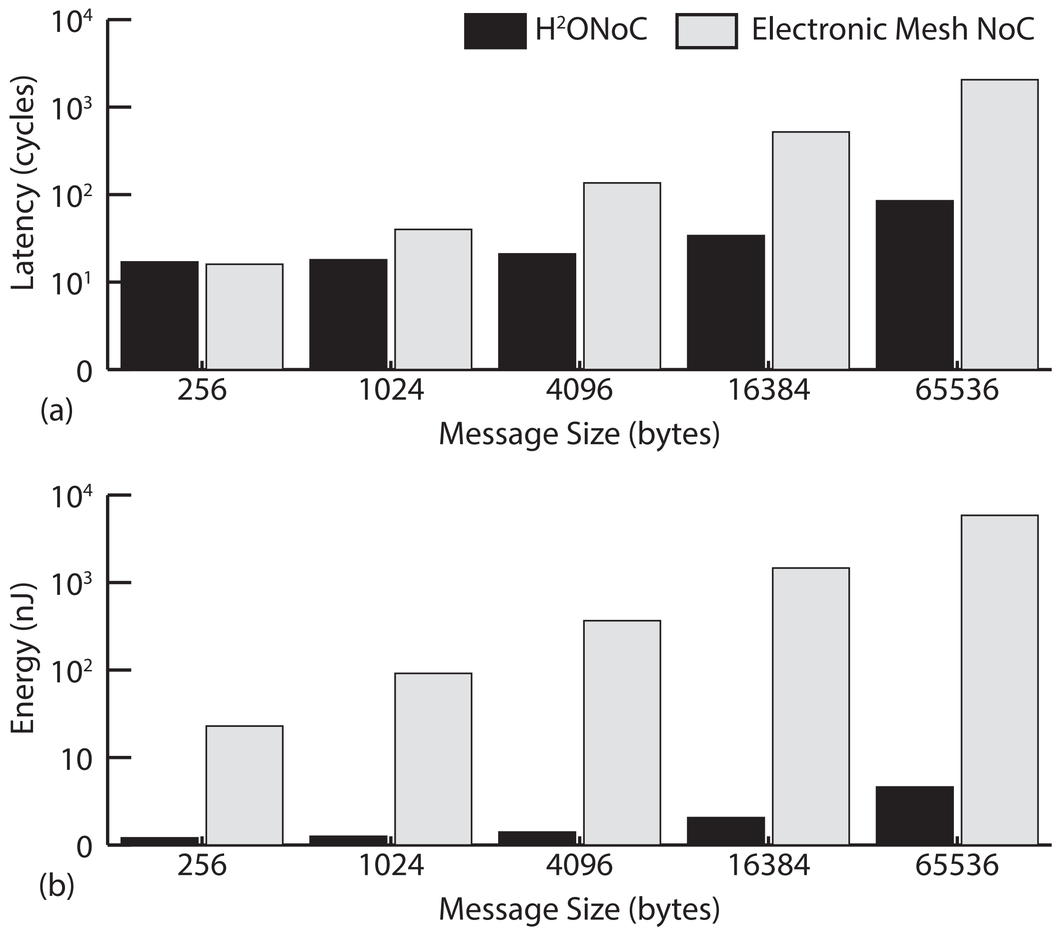


Fig. 7.8 A comparison with an electronic mesh-based NoC in term of (a) latency and (b) energy per message consumption.

waveguide, the waveguide crossing, and the ring dropping. These graphs are evaluated using the values in Table 7.1. The main component is the propagation loss that depends on the length of the longest optical circuit. Both [136] and [100] rely on dimension order routing while H^{2}ONoC uses ILAR. For these topologies with these routing strategies, the longest path is equal to the semiperimeter of the chip size. As a consequence, the insertion loss is the same for the three topologies. The number of occurrences of dropping into a ring is equal to three for the mesh and torus topologies considering an implementation of the photonic switches with straight default paths. This leads to a loss of 1.5 dB. Differently, as explained in Section 7.2, the worst case number of droppings into a ring with H^{2}ONoC is five and hence the loss is 2.5 dB. Concerning the insertion loss due to the waveguide crossings, the torus topology outperform the mesh topology of an average 15%, while H^{2}ONoC has an average of 90% less loss compared to the torus. This difference results in a bandwidth gain.

Fig. 7.10 shows a latency comparison for the 8×8 and 12×12 CMPs for the dif-

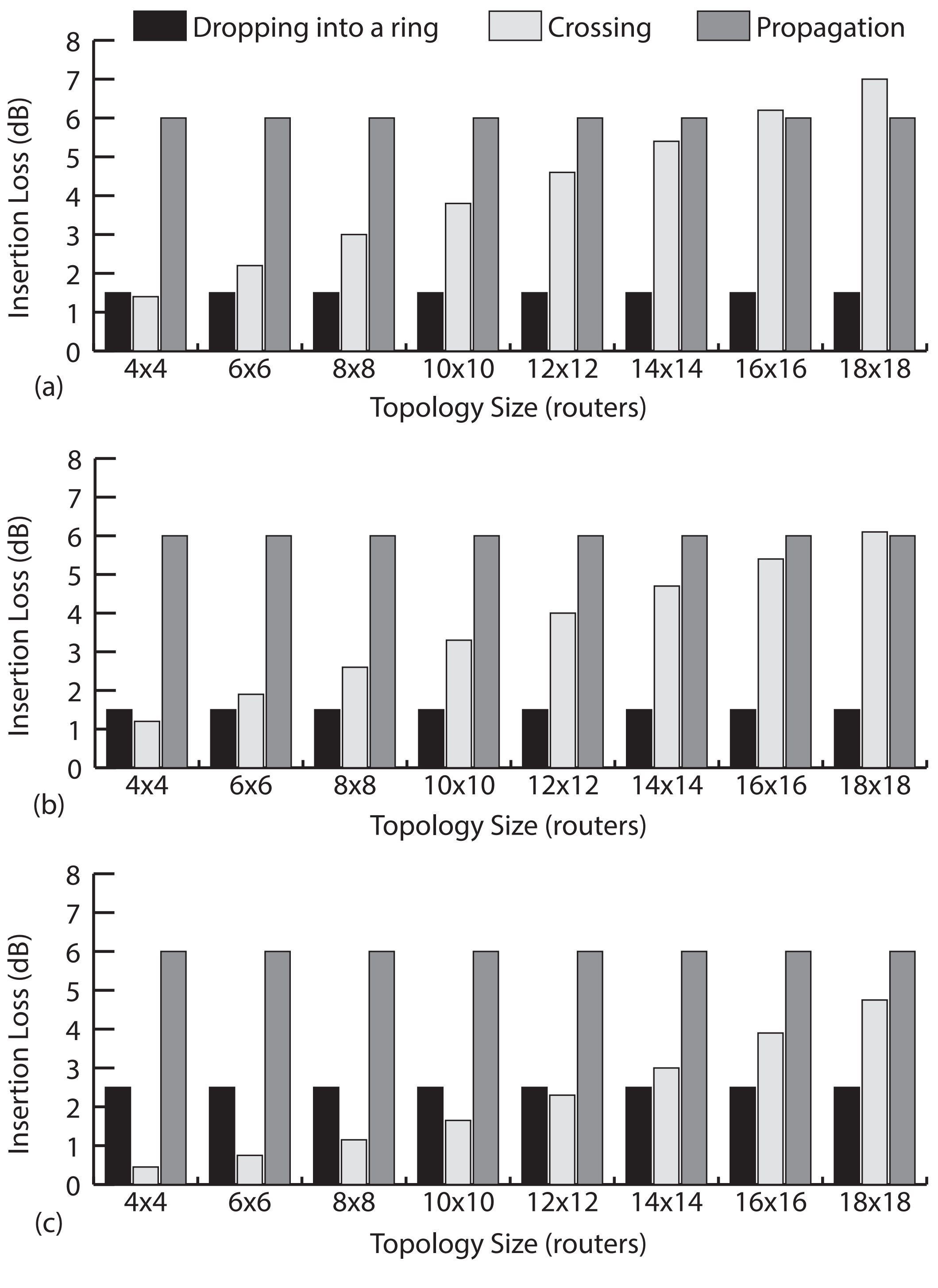


Fig. 7.9 The worst case network-level insertion loss. (a) Mesh Topology. (b) Torus Topology. (c) Hybrid Topology.

ferent benchmarks. For the first benchmark, H^{2}ONoC achieves better results compared to the other architectures thanks to the hybrid topology that helps when sending the large-size

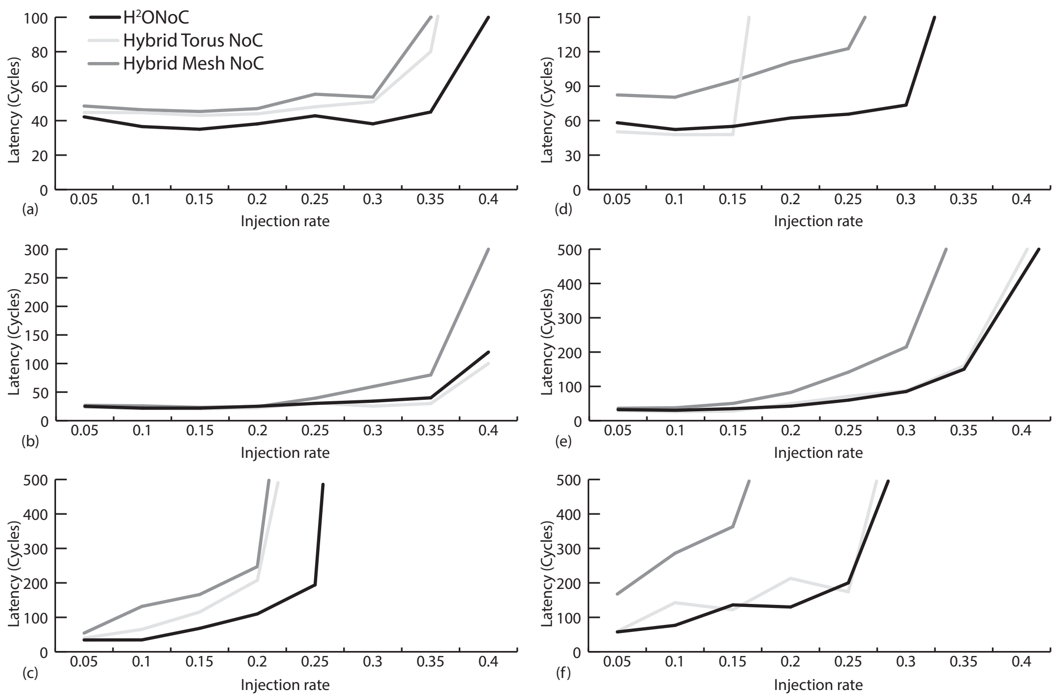


Fig. 7.10 Latency comparison between H^{2}ONoC, the Hybrid Torus NoC [136], and the Hybrid Mesh NoC [100] for two different CMP sizes 8×8 (a) (b) (c) and 12×12(d) (e) (f) for the three benchmarks (a) (d) Bench I (b) (e) Bench II (c) (f) Bench III.

messages. The difference with the torus and mesh NoCs in case of a 8×8 CMP are on the average respectively 18% and 27%. These values increase for the 12×12 CMP. Concerning the second benchmark, which has smaller size messages, H^{2}ONoC performs similar to the torus NoC but both H^{2}ONoC and the torus NoC exhibit a 20% advantage over the mesh NoC. For some injection rates the torus NoC outperforms H^{2}ONoC. This is because with this benchmark the average size of the messages is quite small and hence the reduced diameter of the torus topology may be beneficial in order to reduce the path-setup overhead that is the main components of the latency for messages of this size. Concerning the third benchmark, H^{2}ONoC performs better than both of the alternative architectures thanks to the hybrid topology. Obviously, as the injection rate increases, the network begins saturating and the latency grows rapidly. In case of bigger message sizes, such as in Bench III, this happens with a smaller value of the injection rate. In general, H^{2}ONoC is able to keep the latency under acceptable values for every benchmark providing better results in case of traffic workloads with large size messages. Fig. 7.11 shows a throughput comparison. The throughput and latency trends are similar to each other. Under a specific threshold of the injection rates, i.e. about 0.15, the three architectures provide the same throughput, while for high values of the injection rate, H^{2}ONoC performs better than the alternative architec-

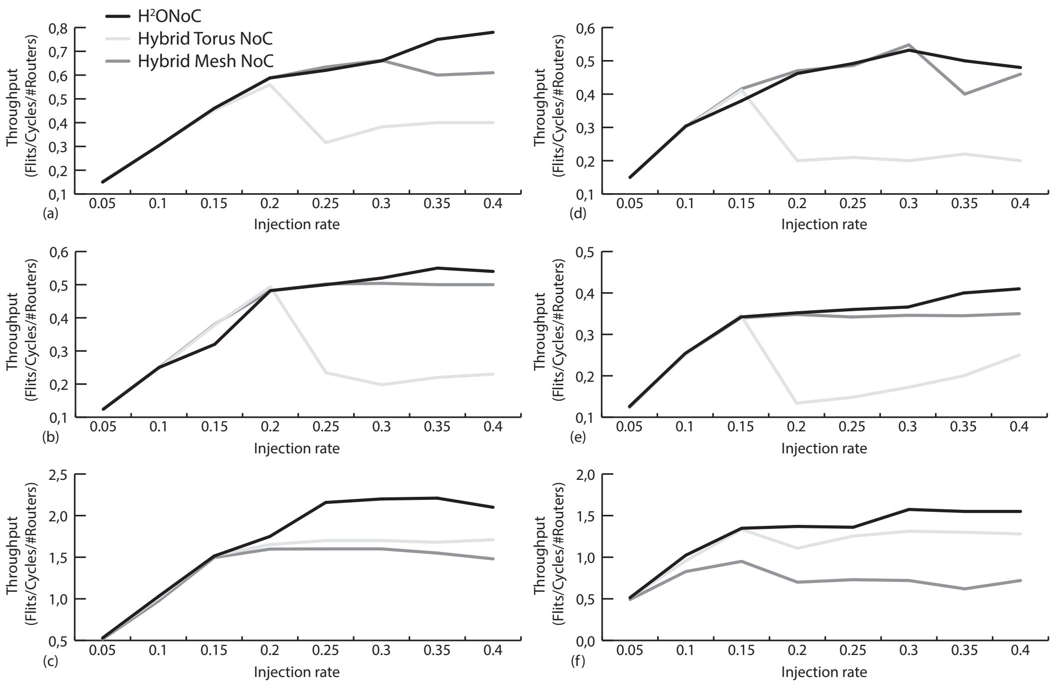


Fig. 7.11 Throughput comparison between H^{2}ONoC, the Hybrid Torus NoC [136], and the Hybrid Mesh NoC [100] for two different CMP sizes 8×8 (a) (b) (c) and 12×12(d) (e) (f) for the three benchmarks (a) (d) Bench I (b) (e) Bench II (c) (f) Bench III.

tures. The throughput of the mesh architecture begins saturating very early while the torus architecture performs similar to H^{2}ONoC. For very high injection rates (more than 0.35) H^{2}ONoC has a slight advantage thanks to the higher bandwidth of the photonic path. As with the latency, the throughput begins saturating for high injection rates leading to congestion. Finally, Fig. 7.12 evaluates the energy consumption for sending a single bit optically. The consumption is divided in two components: the path-setup electronic consumption and the energy consumed by the photonic elements along the optical path. The photonic power consumption is quite constant regardless of the message size, while the path-setup power overhead is higher in the second benchmark due to the low average message sizes. For the first benchmark, the values are the same for the three architectures. When using the second benchmark, the torus have an advantage caused by its wrap-around links that reduce the network diameter, useful in case of small messages since the path-setup overhead has a major impact. Differently, using Bench III, H^{2}ONoC outperforms the other two architectures since it performs better in case of large-size messages.

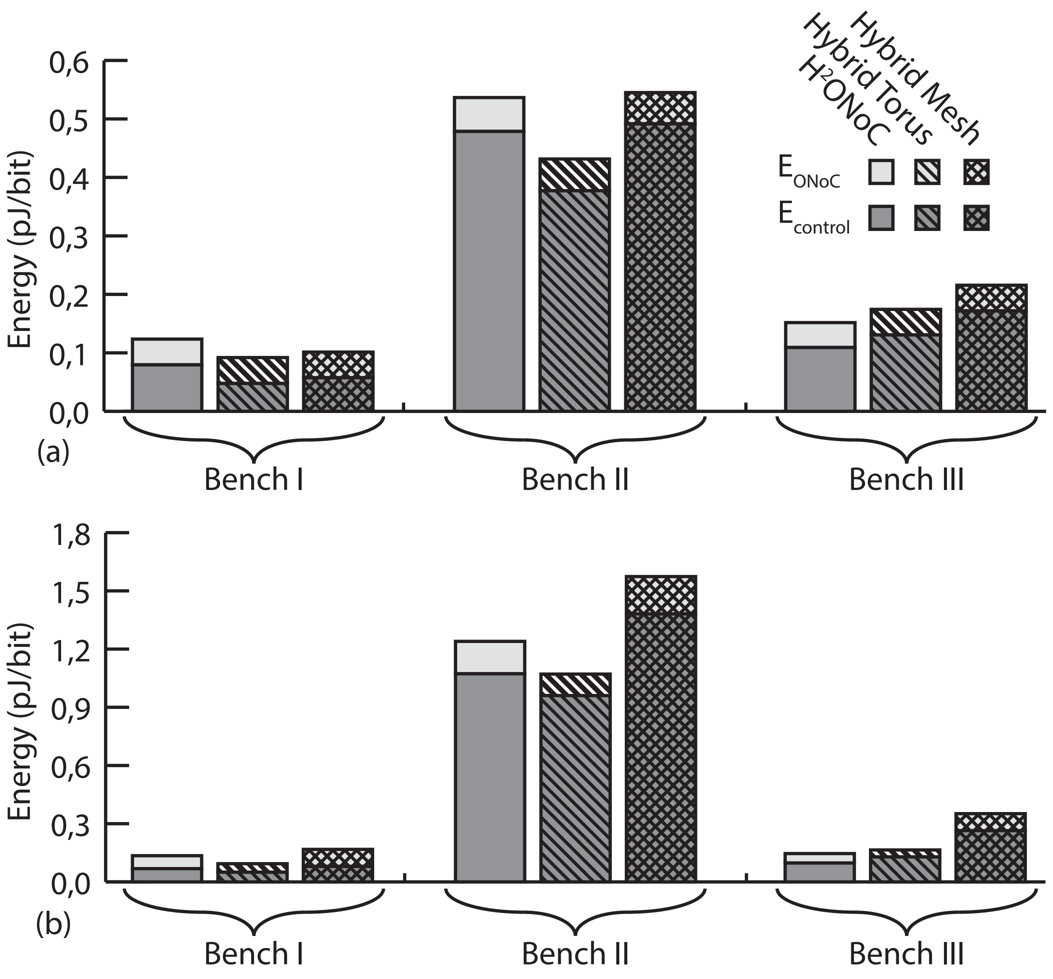


Fig. 7.12 Average energy efficiency comparison for two different CMP sizes 8×8 (a) and 12×12 (b) for the three benchmarks. The breakdown of the energy consumption for the electronic and optical components is shown.

7.4 Summary

In this chapter, we proposed H^{2}ONoC, a hybrid network-on-chip architecture that exploits a combination of electrical and optical signaling to face future CMP needs in terms of performance and energy efficiency. An electronic packet-switched NoC is used for handling control and short size messages and an optical circuit-switched network for large-size messages. Hybrid topologies are used in the photonic layer as a solution enabling reduced power loss. H^{2}ONoC exploits the high wavelength selectivity provided by photonic signaling in order to implement ultra-high bandwidth aggregation. The integration of classical electronics with silicon photonics introduces new design challenges. All the main features of the architecture are analyzed and presented in this chapter.

Compared to an all-electrical concentrated mesh topology, H^2ONoC improves by several orders of magnitude the performance as well as the energy efficiency of large-message communication. Compared to previously proposed architectures, H^2ONoC demonstrates higher throughput, lower latency, and improved energy efficiency with heavy traffic.

Chapter 8

Conclusion

First of all, this thesis has presented an extensive review of state-of-the-art design automation techniques for application-specific electronic on-chip interconnects: Chapter 2 surveyed the most relevant techniques in the literature to analyze a given interconnect solution and reviewed the main approaches available for interconnect synthesis, including several advanced aspects such as co-synthesis of memory and communication architectures, joint scheduling and interconnect synthesis, floorplanning, dynamic configuration, multipath communication.

Then, an automated design methodology, for the synthesis of complex electronic on-chip communication architectures, is presented. The approach combines crossbars and shared buses, connected through bridges, in a hierarchical topology inherently supporting multiple communication paths, yielding a scalable structure and enabling efficient communication patterns. In addition, the above design methodology, presented in Chapter 3, is improved in Chapter 4 by taking into account possible dependencies between tasks. The enhanced approach concurrently defines the structure of the interconnect and the communication task scheduling in order to better exploit the achievable parallelism. Experimental results show that these approaches can synthesize designs made of dozens of IP cores with a small communication overhead and low area and power requirements, exhibiting encouraging improvements over previous proposals in the literature.

While the first half of this thesis targets electronic interconnects, the second half addresses silicon photonics, one of the most prominent emerging technologies for on-chip communication. In that respect, this thesis, first, introduces the basics of silicon photonics and shows its theoretical benefits and then explores a number of non trivial design dilemmas affecting its practical applicability. Chapter 5 provides a cross-cutting understanding of these design challenges, that is an essential step for harnessing the full potential of on-chip Photonics in future computing scenarios.

Then, Chapter 6 proposes and compares, in terms of performance and energy consumption, some path-setup architectural solutions for hybrid photonic-electronic on-chip network that differ from each other in the routing algorithm, the path-setup protocol, the deadlock avoidance technique, and some implementation choices such as the number of virtual channels used. Based on this study, a new power-aware path-setup protocol, that is able to reduce the path-setup latency and the power consumption due to its ability to put allocated circuit on a stand-by state, is proposed.

Finally, Chapter 7 presents H^2ONoC, a hybrid network-on-chip architecture that exploits a combination of electrical and optical signaling to face future CMP needs in terms of performance and energy efficiency. An electronic packet-switched NoC is used for handling control and short size messages and an optical circuit-switched network for large-size messages. Hybrid topologies are used in the photonic layer as a solution enabling reduced power loss. H^2ONoC exploits the high wavelength selectivity provided by photonic signaling in order to implement ultra-high bandwidth aggregation. Compared to an all-electrical concentrated mesh topology, H^2ONoC improves by several orders of magnitude the performance as well as the energy efficiency of large-message communication. Compared to previously proposed architectures, H^2ONoC demonstrates higher throughput, lower latency, and improved energy efficiency with heavy traffic.

References

[1] (1999). *AMBA Specification (Rev 2.0)*. ARM.

[2] (2007). *STBus Communication System: Concepts And Definitions*. STMicroelectronics.

[3] (2012a). *LogiCORE IP AXI Interconnect (v1.06.a)*. Xilinx.

[4] (2012). Nvidia's next generation cuda™ compute architecture: Kepler™ gk110. White Paper.

[5] (2012). *PrimeTime Golden Timing Signoff Solution and Environment*. Synopsys.

[6] (2012b). *XPower Estimator User Guide*. Xilinx.

[7] (2012). *Zynq-7000 All Programmable SoC Overview*. Xilinx.

[8] (2013). *AMBA AXI and ACE Protocol Specification*. ARM.

[9] Adya, S. N. and Markov, I. L. (2003). Fixed-outline floorplanning: Enabling hierarchical design. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 11(6):1120–1135.

[10] Atienza, D., Angiolini, F., Murali, S., Pullini, A., Benini, L., and De Micheli, G. (2008). Network-on-chip design and synthesis outlook. *INTEGRATION, the VLSI journal*, 41(3):340–359.

[11] Bambha, N. K. and Bhattacharyya, S. S. (2005). Joint application mapping/interconnect synthesis techniques for embedded chip-scale multiprocessors. *Parallel and Distributed Systems, IEEE Transactions on*, 16(2):99–112.

[12] Banks, J., Carson, J., and Nelson, B. (2000). *DM Nicol, Discrete-Event System Simulation*. Prentice Hall.

[13] Batten, C., Joshi, A., Orcutt, J., Khilo, A., Moss, B., Holzwarth, C., Popovic, M., Li, H., Smith, H. I., Hoyt, J., et al. (2008). Building manycore processor-to-dram networks with monolithic silicon photonics. In *High Performance Interconnects, 2008. HOTI'08. 16th IEEE Symposium on*, pages 21–30. IEEE.

[14] Benini, L. and De Micheli, G. (2002). Networks on chips: A new soc paradigm. *Computer*, 35(1):70–78.

[15] Benini, L., Macchiarulo, L., Macii, A., and Poncino, M. (2002). Layout-driven memory synthesis for embedded systems-on-chip. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 10(2):96–105.

[16] Bergman, K., Carloni, L. P., Biberman, A., Chan, J., and Hendry, G. (2013). *Photonic Network-on-Chip Design*. Springer.

[17] Bertozzi, D., Jalabert, A., Murali, S., Tamhankar, R., Stergiou, S., Benini, L., and De Micheli, G. (2005). Noc synthesis flow for customized domain specific multiprocessor systems-on-chip. *Parallel and Distributed Systems, IEEE Transactions on*, 16(2):113–129.

[18] Bezerra, G. B., Forrest, S., and Zarkesh-Ha, P. (2011). Reducing energy and increasing performance with traffic optimization in many-core systems. In *Proceedings of the System Level Interconnect Prediction Workshop*, page 3. IEEE Press.

[19] Bhojwani, P. and Mahapatra, R. (2003). Interfacing cores with on-chip packet-switched networks. In *VLSI Design, 2003. Proceedings. 16th International Conference on*, pages 382–387. IEEE.

[20] Biberman, A., Lee, B. G., Bergman, K., Dong, P., and Lipson, M. (2008). Demonstration of all-optical multi-wavelength message routing for silicon photonic networks. In *Optical Fiber Communication Conference*, page OTuF6. Optical Society of America.

[21] Bogaerts, W., Dumon, P., Thourhout, D. V., and Baets, R. (2007). Low-loss, low-cross-talk crossings for silicon-on-insulator nanophotonic waveguides. *Optics letters*, 32(19):2801–2803.

[22] Borkar, S. (2007). Thousand core chips: a technology perspective. In *Proceedings of the 44th annual Design Automation Conference*, pages 746–749. ACM.

[23] Burjorjee, K. M. (2013). Explaining optimization in genetic algorithms with uniform crossover. In *Proceedings of the twelfth workshop on Foundations of genetic algorithms XII*, FOGA XII '13, pages 37–50, New York, NY, USA. ACM.

[24] Caldwell, A. E., Kahng, A. B., Mantik, S., Markov, I. L., and Zelikovsky, A. (1999). On wirelength estimations for row-based placement. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 18(9):1265–1278.

[25] Canny, J. (1986). A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):679–698.

[26] Catthoor, F., Greef, E. d., and Suytack, S. (1998). *Custom memory management methodology: Exploration of memory organisation for embedded multimedia system design*. Kluwer Academic Publishers.

[27] Chan, J., Biberman, A., Lee, B. G., and Bergman, K. (2008). Insertion loss analysis in a photonic interconnection network for on-chip and off-chip communications. *IEEE Lasers and Electro-Optics Society (LEOS)*.

[28] Chan, J., Hendry, G., Bergman, K., and Carloni, L. P. (2011). Physical-layer modeling and system-level design of chip-scale photonic interconnection networks. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 30(10):1507–1520.

[29] Chan, J., Hendry, G., Biberman, A., and Bergman, K. (2010). Architectural exploration of chip-scale photonic interconnection network designs using physical-layer analysis. *Journal of Lightwave Technology*, 28(9):1305–1315.

[30] Cilardo, A., Fusella, E., Gallo, L., and Mazzeo, A. (2013). Automated synthesis of fpga-based heterogeneous interconnect topologies. In *Field Programmable Logic and Applications (FPL), 2013 23rd International Conference on*, pages 1–8. IEEE.

[31] Cilardo, A., Fusella, E., Gallo, L., and Mazzeo, A. (2014a). Joint communication scheduling and interconnect synthesis for fpga-based many-core systems. In *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014*, pages 1–4. IEEE.

[32] Cilardo, A., Fusella, E., Gallo, L., Mazzeo, A., and Mazzocca, N. (2014b). Automated design space exploration for fpga-based heterogeneous interconnects. *Design Automation for Embedded Systems*, pages 1–14.

[33] Cong, J., Huang, Y., and Yuan, B. (2011). Atree-based topology synthesis for on-chip network. In *Computer-Aided Design (ICCAD), 2011 IEEE/ACM International Conference on*, pages 651–658. IEEE.

[34] Dally, W. J. and Towles, B. (2001). Route packets, not wires: On-chip interconnection networks. In *Design Automation Conference, 2001. Proceedings*, pages 684–689. IEEE.

[35] Daveau, J.-M., Ismail, T. B., and Jerraya, A. A. (1995). Synthesis of system-level communication by an allocation-based approach. In *Proceedings of the 8th international symposium on System synthesis*, pages 150–155. ACM.

[36] De Micheli, G. and Benini, L. (2006). *Networks on chips: technology and tools*. Academic Press.

[37] Devanur, N. R. and Feige, U. (2011). An o (n log n) algorithm for a load balancing problem on paths. In *Algorithms and Data Structures*, pages 326–337. Springer.

[38] Dick, R. P., Rhodes, D. L., and Wolf, W. (1998). Tgff: task graphs for free. In *Proceedings of the 6th international workshop on Hardware/software codesign*, pages 97–101. IEEE Computer Society.

[39] Doany, F. E., Lee, B. G., Assefa, S., Green, W. M., Yang, M., Schow, C. L., Jahnes, C. V., Zhang, S., Singer, J., Kopp, V. I., et al. (2011). Multichannel high-bandwidth coupling of ultradense silicon photonic waveguide array to standard-pitch fiber array. *Lightwave Technology, Journal of*, 29(4):475–482.

[40] Dong, P., Qian, W., Liao, S., Liang, H., Kung, C.-C., Feng, N.-N., Shafiiha, R., Fong, J., Feng, D., Krishnamoorthy, A. V., et al. (2010). Low loss silicon waveguides for application of optical interconnects. In *Proc. IEEE Photon. Soc. Summer Topical Meeting Ser*, pages 191–192.

[41] Drinic, M., Kirovski, D., Megerian, S., and Potkonjak, M. (2006). Latency-guided on-chip bus-network design. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 25(12):2663–2673.

[42] Edmonds, J. (1968). *Optimum branchings*. National Bureau of standards.

[43] Eisley, N. and Peh, L.-S. (2004). High-level power analysis for on-chip networks. In *Proceedings of the 2004 international conference on Compilers, architecture, and synthesis for embedded systems*, pages 104–115. ACM.

[44] Esmaeilzadeh, H., Blem, E., St Amant, R., Sankaralingam, K., and Burger, D. (2011). Dark silicon and the end of multicore scaling. In *Computer Architecture (ISCA), 2011 38th Annual International Symposium on*, pages 365–376. IEEE.

[45] Feng, K., Ye, Y., and Xu, J. (2013). A formal study on topology and floorplan characteristics of mesh and torus-based optical networks-on-chip. *Microprocessors and Microsystems*, 37(8):941–952.

[46] Fusella, E., Flich, J., Cilardo, A., and Mazzeo, A. (2015). On the design of a path-setup architecture for exploiting hybrid photonic-electronic nocs. In *Exploiting Silicon Photonics for Energy-Efficient High Performance Computing (SiPhotonics), 2015 Workshop on*, pages 9–16. IEEE.

[47] Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.

[48] GASTEIER, M. and GLESNER, M. (1999). Bus-based communication synthesis on system level. *ACM Transactions on Design Automation of Electronic Systems*, 4(1):1–11.

[49] Goren, O. and Netanel, Y. (2006). High performance on-chip interconnect system supporting fast soc generation. In *VLSI Design, Automation and Test, 2006 International Symposium on*, pages 1–4. IEEE.

[50] Groups, T. I. T. W. (2005). International technology roadmap for semiconductors (itrs).

[51] Guerrier, P. and Greiner, A. (2000). A generic architecture for on-chip packet-switched interconnections. In *Proceedings of the conference on Design, automation and test in Europe*, pages 250–256. ACM.

[52] Han, K.-H. and Kim, J.-H. (2002). Quantum-inspired evolutionary algorithm for a class of combinatorial optimization. *Evolutionary Computation, IEEE Transactions on*, 6(6):580–593.

[53] Hayenga, M., Johnson, D. R., and Lipasti, M. (2010). Pitfalls of orion-based simulation. *ORION*, 35:40–000.

[54] Ho, R., Mai, K. W., and Horowitz, M. A. (2001). The future of wires. *Proceedings of the IEEE*, 89(4):490–504.

[55] Hu, Y., Zhu, Y., Chen, H., Graham, R., and Cheng, C.-K. (2006). Communication latency aware low power noc synthesis. In *Proceedings of the 43rd annual Design Automation Conference*, pages 574–579. ACM.

[56] Hur, J. Y. (2011). *Customizing and hardwiring on-chip interconnects in FPGAs*. PhD dissertation, TU Delft.

[57] Hur, J. Y., Stefanov, T., Wong, S., and Goossens, K. (2012). Customisation of on-chip network interconnects and experiments in field-programmable gate arrays. *IET computers & digital techniques*, 6(1):59–68.

[58] Hur, J. Y., Stefanov, T., Wong, S., and Vassiliadis, S. (2007). Systematic customization of on-chip crossbar interconnects. In *Reconfigurable computing: architectures, tools and applications*, pages 61–72. Springer.

[59] Hur, J. Y., Wong, S., and Stefanov, T. (2010). Design trade-offs in customized on-chip crossbar schedulers. *Journal of Signal Processing Systems*, 58(1):69–85.

[60] Iancu, C. C. and Strohmaier, E. (2007). Optimizing communication overlap for high-speed networks. In *Proceedings of the 12th ACM SIGPLAN symposium on Principles and practice of parallel programming*, pages 35–45. ACM.

[61] IBM (2012). Coreconnect interconnect standard.

[62] Issenin, I., Brockmeyer, E., Durinck, B., and Dutt, N. D. (2008). Data-reuse-driven energy-aware cosynthesis of scratch pad memory and hierarchical bus-based communication architecture for multiprocessor streaming applications. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 27(8):1439–1452.

[63] Jang, Y., Kim, J., and Kyung, C.-M. (2010). Topology synthesis for low power cascaded crossbar switches. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 29(12):2041–2045.

[64] Ji, R., Xu, J., Yang, L., et al. (2013). Five-port optical router based on microring switches for photonic networks-on-chip. *IEEE Photon. Technol. Lett*, 25(5):492–495.

[65] Joo, Y.-P., Kim, S., and Ha, S. (2012). Efficient hierarchical bus-matrix architecture exploration of processor pool-based mpsoc. *Design Automation for Embedded Systems*, 16(4):293–317.

[66] Jun, M., Bang, K., Lee, H.-J., Chang, N., and Chung, E.-Y. (2007). Slack-based bus arbitration scheme for soft real-time constrained embedded systems. In *Design Automation Conference, 2007. ASP-DAC'07. Asia and South Pacific*, pages 159–164. IEEE.

[67] Jun, M., Woo, D., and Chung, E.-Y. (2012). Partial connection-aware topology synthesis for on-chip cascaded crossbar network. *Computers, IEEE Transactions on*, 61(1):73–86.

[68] Jun, M., Yoo, S., and Chung, E.-Y. (2008). Mixed integer linear programming-based optimal topology synthesis of cascaded crossbar switches. In *Design Automation Conference, 2008. ASPDAC 2008. Asia and South Pacific*, pages 583–588. IEEE.

[69] Jun, M., Yoo, S., and Chung, E.-Y. (2009). Topology synthesis of cascaded crossbar switches. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 28(6):926–930.

[70] Kamil, S., Oliker, L., Pinar, A., and Shalf, J. (2010). Communication requirements and interconnect optimization for high-end scientific applications. *Parallel and Distributed Systems, IEEE Transactions on*, 21(2):188–202.

[71] Keitel-Schulz, D. and Wehn, N. (2001). Embedded dram development: Technology, physical design, and application issues. *IEEE Design & Test*, 18(3):7–15.

[72] Keutzer, K., Newton, A. R., Rabaey, J. M., and Sangiovanni-Vincentelli, A. (2000). System-level design: Orthogonalization of concerns and platform-based design. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 19(12):1523–1543.

[73] Kim, S. and Ha, S. (2006). Efficient exploration of bus-based system-on-chip architectures. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 14(7):681–692.

[74] Kim, S., Im, C., and Ha, S. (2005). Schedule-aware performance estimation of communication architecture for efficient design space exploration. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 13(5):539–552.

[75] Kırman, N., Kırman, M., Dokania, R. K., Martinez, J. F., Apsel, A. B., Watkins, M. A., and Albonesi, D. H. (2007). On-chip optical technology in future bus-based multicore designs. *IEEE micro*, 27(1):56–66.

[76] Knudsen, P. V. and Madsen, J. (1999). Integrating communication protocol selection with hardware/software codesign. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 18(8):1077–1095.

[77] Lahiri, K., Raghunathan, A., and Dey, S. (2001). System-level performance analysis for designing on-chip communication architectures. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 20(6):768–783.

[78] Lahiri, K., Raghunathan, A., and Dey, S. (2004). Design space exploration for optimizing on-chip communication architectures. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 23(6):952–961.

[79] Lahiri, K., Raghunathan, A., and Lakshminarayana, G. (2006). The lotterybus on-chip communication architecture. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 14(6):596–608.

[80] Lee, B. G., Biberman, A., Dong, P., Lipson, M., and Bergman, K. (2008). All-optical comb switch for multiwavelength message routing in silicon photonic networks. *Photonics Technology Letters, IEEE*, 20(10):767–769.

[81] Lee, C., Kim, S., and Ha, S. (2010). A systematic design space exploration of mpsoc based on synchronous data flow specification. *Journal of Signal Processing Systems*, 58(2):193–213.

[82] Lee, H. G., Chang, N., Ogras, U. Y., and Marculescu, R. (2007). On-chip communication architecture exploration: A quantitative evaluation of point-to-point, bus, and network-on-chip approaches. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 12(3):23.

[83] Li, S., Chen, K., Ahn, J. H., Brockman, J. B., and Jouppi, N. P. (2011). Cacti-p: Architecture-level modeling for sram-based structures with advanced leakage reduction techniques. In *Computer-Aided Design (ICCAD), 2011 IEEE/ACM International Conference on*, pages 694–701. IEEE.

[84] Liao, T. G. S., Martin, G., Swan, S., and Grötker, T. (2002). *System design with SystemC*. Springer.

[85] Liljeberg, P., Plosila, J., and Isoaho, J. (2003). Self-timed ring architecture for soc applications. In *SOC Conference, 2003. Proceedings. IEEE International [Systems-on-Chip]*, pages 359–362. IEEE.

[86] Lin, B.-C., Lee, G.-W., Huang, J.-D., and Jou, J.-Y. (2007). A precise bandwidth control arbitration algorithm for hard real-time soc buses. In *Design Automation Conference, 2007. ASP-DAC'07. Asia and South Pacific*, pages 165–170. IEEE.

[87] Liu, M., Lu, Z., Kuehn, W., and Jantsch, A. (2012). A survey of fpga dynamic reconfiguration design methodology and applications. *International Journal of Embedded and Real-Time Communication Systems (IJERTCS)*, 3(2):23–39.

[88] Liu, W., Xu, J., Wu, X., Ye, Y., Wang, X., Zhang, W., Nikdast, M., and Wang, Z. (2011). A noc traffic suite based on real applications. In *VLSI (ISVLSI), 2011 IEEE Computer Society Annual Symposium on*, pages 66–71. IEEE.

[89] Loghi, M., Angiolini, F., Bertozzi, D., Benini, L., and Zafalon, R. (2004). Analyzing on-chip communication in a mpsoc environment. In *Proceedings of the conference on Design, automation and test in Europe-Volume 2*, page 20752. IEEE Computer Society.

[90] Lu, R., Cao, A., and Koh, C.-K. (2007). Samba-bus: a high performance bus architecture for system-on-chips. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 15(1):69–79.

[91] Luce, R. D. and Perry, A. D. (1949). A method of matrix analysis of group structure. *Psychometrika*, 14(2):95–116.

[92] Lukasiewycz, M., Glaß, M., Reimann, F., and Teich, J. (2011). Opt4j: a modular framework for meta-heuristic optimization. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, GECCO '11, pages 1723–1730, New York, NY, USA. ACM.

[93] Lysaght, P., Blodget, B., Mason, J., Young, J., and Bridgford, B. (2006). Invited paper: Enhanced architectures, design methodologies and cad tools for dynamic reconfiguration of xilinx fpgas. In *Field Programmable Logic and Applications, 2006. FPL'06. International Conference on*, pages 1–6. IEEE.

[94] Marculescu, R., Ogras, U. Y., Peh, L.-S., Jerger, N. E., and Hoskote, Y. (2009). Outstanding research problems in noc design: system, microarchitecture, and circuit perspectives. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 28(1):3–21.

[95] Martin, S. M., Flautner, K., Mudge, T., and Blaauw, D. (2002). Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads. In *Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design*, pages 721–725. ACM.

[96] Meftali, S., Gharsalli, F., Rousseau, F., and Jerraya, A. A. (2001). An optimal memory allocation for application-specific multiprocessor system-on-chip. In *Proceedings of the 14th international symposium on Systems synthesis*, pages 19–24. ACM.

[97] Mello, A., Tedesco, L., Calazans, N., and Moraes, F. (2005). Virtual channels in networks on chip: implementation and evaluation on hermes noc. In *Proceedings of the 18th annual symposium on Integrated circuits and system design*, pages 178–183. ACM.

[98] Meyer, B. H. and Thomas, D. E. (2007). Simultaneous synthesis of buses, data mapping and memory allocation for mpsoc. In *Proceedings of the 5th IEEE/ACM international conference on Hardware/software codesign and system synthesis*, pages 3–8. ACM.

[99] Micheli, G. D. (1994). *Synthesis and Optimization of Digital Circuits*. McGraw-Hill Higher Education, 1st edition.

[100] Mo, K. H., Ye, Y., Wu, X., Zhang, W., Liu, W., and Xu, J. (2010). A hierarchical hybrid optical-electronic network-on-chip. In *VLSI (ISVLSI), 2010 IEEE Computer Society Annual Symposium on*, pages 327–332. IEEE.

[101] Moraes, F., Calazans, N., Mello, A., Möller, L., and Ost, L. (2004). Hermes: an infrastructure for low area overhead packet-switching networks on chip. *INTEGRATION, the VLSI journal*, 38(1):69–93.

[102] Mujuni, E. and Rosamond, F. (2008). Parameterized complexity of the clique partition problem. In *Proceedings of the fourteenth symposium on Computing: the Australasian theory - Volume 77*, CATS '08, pages 75–78, Darlinghurst, Australia, Australia. Australian Computer Society, Inc.

[103] Murali, S., Benini, L., and De Micheli, G. (2007). An application-specific design methodology for on-chip crossbar generation. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 26(7):1283–1296.

[104] Murali, S. and De Micheli, G. (2004). Bandwidth-constrained mapping of cores onto noc architectures. In *Proceedings of the conference on Design, automation and test in Europe-Volume 2*, page 20896. IEEE Computer Society.

[105] Murali, S. and De Micheli, G. (2005). An application-specific design methodology for stbus crossbar generation. In *Design, Automation and Test in Europe, 2005. Proceedings*, pages 1176–1181. IEEE.

[106] Muralimanohar, N., Balasubramonian, R., and Jouppi, N. P. (2009). Cacti 6.0: A tool to model large caches. *HP Laboratories*.

[107] Na, S., Yang, S., and Kyung, C.-M. (2009). Low-power bus architecture composition for amba axi. *Journal of Semiconductor Technology and Science*, 9(2):1.

[108] Nesterov, Y., Nemirovskii, A., and Ye, Y. (1994). *Interior-point polynomial algorithms in convex programming*, volume 13. SIAM.

[109] Nikdast, M., Xu, J., Wu, X., Zhang, W., Ye, Y., Wang, X., Wang, Z., and Wang, Z. (2014). Systematic analysis of crosstalk noise in folded-torus-based optical networks-on-chip. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 33(3):437–450.

[110] Nikolova, D., Rumley, S., Calhoun, D., Li, Q., Hendry, R., Samadi, P., and Bergman, K. (2015). Scaling silicon photonic switch fabrics for data center interconnection networks. *Optics Express*, 23(2):1159–1175.

[111] Ogras, U. Y., Bogdan, P., and Marculescu, R. (2010). An analytical approach for network-on-chip performance analysis. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 29(12):2001–2013.

[112] Ogras, U. Y., Hu, J., and Marculescu, R. (2005). Key research problems in noc design: a holistic perspective. In *Proceedings of the 3rd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, pages 69–74. ACM.

[113] Owens, J. D., Dally, W. J., Ho, R., Jayasimha, D., Keckler, S. W., and Peh, L.-S. (2007). Research challenges for on-chip interconnection networks. *IEEE micro*, 27(5):96.

[114] Panda, P. R., Dutt, N. D., and Nicolau, A. (1999). *Memory issues in embedded systems-on-chip: optimizations and exploration*. Springer.

[115] Pande, P. P., Grecu, C., Jones, M., Ivanov, A., and Saleh, R. (2005a). Effect of traffic localization on energy dissipation in noc-based interconnect. In *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*, pages 1774–1777. IEEE.

[116] Pande, P. P., Grecu, C., Jones, M., Ivanov, A., and Saleh, R. (2005b). Performance evaluation and design trade-offs for network-on-chip interconnect architectures. *Computers, IEEE Transactions on*, 54(8):1025–1040.

[117] Pandey, S. and Drechsler, R. (2008). Robust on-chip bus architecture synthesis for mpsocs under random tasks arrival. In *Proceedings of the 2008 Asia and South Pacific Design Automation Conference*, pages 601–606. IEEE Computer Society Press.

[118] Pandey, S. and Glesner, M. (2007). Simultaneous on-chip bus synthesis and voltage scaling under random on-chip data traffic. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 15(10):1111–1124.

[119] Papanikolaou, A., Koppenberger, K., Miranda, M., and Catthoor, F. (2004). Memory communication network exploration for low-power distributed memory organisations. In *Signal Processing Systems, 2004. SIPS 2004. IEEE Workshop on*, pages 176–181. IEEE.

[120] Pasricha, S. and Dutt, N. (2010). *On-chip communication architectures: system on chip interconnect*. Morgan Kaufmann.

[121] Pasricha, S., Dutt, N., and Ben-Romdhane, M. (2004). Fast exploration of bus-based on-chip communication architectures. In *Proceedings of the 2nd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, pages 242–247. ACM.

[122] Pasricha, S. and Dutt, N. D. (2007). A framework for cosynthesis of memory and communication architectures for mpsoc. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 26(3):408–420.

[123] Pasricha, S., Dutt, N. D., and Ben-Romdhane, M. (2007). Bmsyn: bus matrix communication architecture synthesis for mpsoc. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 26(8):1454–1464.

[124] Pasricha, S., Dutt, N. D., Bozorgzadeh, E., and Ben-Romdhane, M. (2006). Fabsyn: Floorplan-aware bus architecture synthesis. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 14(3):241–253.

[125] Pasricha, S., Park, Y.-H., Kurdahi, F. J., and Dutt, N. (2010). Capps: A framework for power–performance tradeoffs in bus-matrix-based on-chip communication architecture synthesis. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 18(2):209–221.

[126] Penaranda, R., Gomez, C., Gomez, M. E., Lopez, P., and Duato, J. (2012). A new family of hybrid topologies for large-scale interconnection networks. In *Network Computing and Applications (NCA), 2012 11th IEEE International Symposium on*, pages 220–227. IEEE.

[127] Peng, H.-K. and Lin, Y.-L. (2010). An optimal warning-zone-length assignment algorithm for real-time and multiple-qos on-chip bus arbitration. *ACM Transactions on Embedded Computing Systems (TECS)*, 9(4):35.

[128] Pham-Quoc, C., Al-Ars, Z., and Bertels, K. (2012). A heuristic-based communication-aware hardware optimization approach in heterogeneous multicore systems. In *Reconfigurable Computing and FPGAs (ReConFig), 2012 International Conference on*, pages 1–6. IEEE.

[129] Pyoun, C. H., Lin, C. H., Kim, H. S., and Chong, J. W. (2003). The efficient bus arbitration scheme in soc environment. In *System-on-Chip for Real-Time Applications, 2003. Proceedings. The 3rd IEEE International Workshop on*, pages 311–315. IEEE.

[130] Ryu, K. K. and Mooney, V. J. (2004). Automated bus generation for multiprocessor soc design. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 23(11):1531–1549.

[131] Ryu, K. K., Shin, E., and Mooney, V. J. (2001). A comparison of five different multiprocessor soc bus architectures. In *Digital Systems Design, 2001. Proceedings. Euromicro Symposium on*, pages 202–209. IEEE.

[132] Salminen, E., Kulmala, A., and Hamalainen, T. D. (2007). On network-on-chip comparison. In *Digital System Design Architectures, Methods and Tools, 2007. DSD 2007. 10th Euromicro Conference on*, pages 503–510. IEEE.

[133] Salminen, E., Kulmala, A., and Hamalainen, T. D. (2008). Survey of network-on-chip proposals. *white paper, OCP-IP*, pages 1–13.

[134] Sekar, K., Lahiri, K., Raghunathan, A., and Dey, S. (2008). Dynamically configurable bus topologies for high-performance on-chip communication. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 16(10):1413–1426.

[135] Shacham, A., Bergman, K., and Carloni, L. P. (2007). On the design of a photonic network-on-chip. In *Proceedings of the First International Symposium on Networks-on-Chip*, pages 53–64. IEEE Computer Society.

[136] Shacham, A., Bergman, K., and Carloni, L. P. (2008). Photonic networks-on-chip for future generations of chip multiprocessors. *Computers, IEEE Transactions on*, 57(9):1246–1260.

[137] Shah, H., Shiu, P., Bell, B., Aldredge, M., Sopory, N., and Davis, J. (2002). Repeater insertion and wire sizing optimization for throughput-centric vlsi global interconnects. In *Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design*, pages 280–284. ACM.

[138] Sherwani, N. A. (1995). *Algorithms for VLSI physical design automation*. Kluwer academic publishers.

[139] Srinivasan, K. and Chatha, K. S. (2006). A low complexity heuristic for design of custom network-on-chip architectures. In *Proceedings of the conference on Design, automation and test in Europe: Proceedings*, pages 130–135. European Design and Automation Association.

[140] Srinivasan, S., Angiolini, F., Ruggiero, M., Benini, L., and Vijaykrishnan, N. (2005a). Simultaneous memory and bus partitioning for soc architectures. In *SOC Conference, 2005. Proceedings. IEEE International*, pages 125–128. IEEE.

[141] Srinivasan, S., Li, L., and Vijaykrishnan, N. (2005b). Simultaneous partitioning and frequency assignment for on-chip bus architectures. In *Proceedings of the Conference on Design, Automation and Test in Europe - Volume 1*, DATE '05, pages 218–223, Washington, DC, USA. IEEE Computer Society.

[142] Strehl, A. and Ghosh, J. (2003). Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *The Journal of Machine Learning Research*, 3:583–617.

[143] Suh, J. and Yoo, H.-J. (2004). Arbitration latency analysis of the shared channel architecture for high performance multi-master soc. In *Advanced System Integrated Circuits 2004. Proceedings of 2004 IEEE Asia-Pacific Conference on*, pages 388–391. IEEE.

[144] Thepayasuwan, N. and Doboli, A. (2004). Layout conscious bus architecture synthesis for deep submicron systems on chip. In *Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings*, volume 1, pages 108–113. IEEE.

[145] Vantrease, D., Schreiber, R., Monchiero, M., McLaren, M., Jouppi, N. P., Fiorentino, M., Davis, A., Binkert, N., Beausoleil, R. G., and Ahn, J. H. (2008). Corona: System implications of emerging nanophotonic technology. In *ACM SIGARCH Computer Architecture News*, volume 36, pages 153–164. IEEE Computer Society.

[146] Varatkar, G. and Marculescu, R. (2002). Traffic analysis for on-chip networks design of multimedia applications. In *Design Automation Conference, 2002. Proceedings. 39th*, pages 795–800. IEEE.

[147] Vassiliadis, S. and Sourdis, I. (2007). {FLUX} interconnection networks on demand. *Journal of Systems Architecture*, 53(10):777 – 793. Embedded Computer Systems: Architectures, Modeling, and Simulation.

[148] Vetter, J. S. and Mueller, F. (2003). Communication characteristics of large-scale scientific applications for contemporary cluster architectures. *Journal of Parallel and Distributed Computing*, 63(9):853–865.

[149] Vlasov, Y. and McNab, S. (2004). Losses in single-mode silicon-on-insulator strip waveguides and bends. *Optics express*, 12(8):1622–1631.

[Wang et al.] Wang, H., Lee, B. G., Shacham, A., and Bergman, K. On the design of a 4× 4 nonblocking nanophotonic switch for photonic networks on chip.

[151] Wang, H.-S., Zhu, X., Peh, L.-S., and Malik, S. (2002). Orion: a power-performance simulator for interconnection networks. In *Microarchitecture, 2002.(MICRO-35). Proceedings. 35th Annual IEEE/ACM International Symposium on*, pages 294–305. IEEE.

[152] Wang, L.-T., Chang, Y.-W., and Cheng, K.-T. T. (2009). *Electronic design automation: synthesis, verification, and test*. Morgan Kaufmann.

[153] Wolf, W. (2004). The future of multiprocessor systems-on-chips. In *Proceedings of the 41st annual Design Automation Conference*, pages 681–685. ACM.

[154] Wolf, W. H. (1994). Hardware-software co-design of embedded systems [and prolog]. *Proceedings of the IEEE*, 82(7):967–989.

[155] Wuytack, S., Catthoor, F., De Jong, G., and De Man, H. J. (1999). Minimizing the required memory bandwidth in vlsi system realizations. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 7(4):433–441.

[156] Xia, F., Rooks, M., Sekaric, L., and Vlasov, Y. (2007a). Ultra-compact high order ring resonator filters using submicron silicon photonic wires for on-chip optical interconnects. *Optics express*, 15(19):11934–11941.

[157] Xia, F., Sekaric, L., and Vlasov, Y. (2007b). Ultracompact optical buffers on a silicon chip. *Nature Photonics*, 1:65–71.

[158] Xie, Y., Nikdast, M., Xu, J., Wu, X., Zhang, W., Ye, Y., Wang, X., Wang, Z., and Liu, W. (2013). Formal worst-case analysis of crosstalk noise in mesh-based optical networks-on-chip. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 21(10):1823–1836.

[159] Xu, C. Q., Xue, C. J., He, Y., and Sha, E. H. (2010). Energy efficient joint scheduling and multi-core interconnect design. In *Proceedings of the 2010 Asia and South Pacific Design Automation Conference*, pages 879–884. IEEE Press.

[160] Yan, L., Luo, J., and Jha, N. K. (2005). Joint dynamic voltage scaling and adaptive body biasing for heterogeneous distributed real-time embedded systems. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 24(7):1030–1041.

[161] Ye, Y., Xu, J., Wu, X., Zhang, W., Liu, W., and Nikdast, M. (2012). A torus-based hierarchical optical-electronic network-on-chip for multiprocessor system-on-chip. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 8(1):5.

[162] Yi, Y., Kim, D., and Ha, S. (2007). Fast and accurate cosimulation of mpsoc using trace-driven virtual synchronization. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 26(12):2186–2200.

[163] Yoo, J., Lee, D., Yoo, S., and Choi, K. (2007). Communication architecture synthesis of cascaded bus matrix. In *Design Automation Conference, 2007. ASP-DAC'07. Asia and South Pacific*, pages 171–177. IEEE.

[164] Yoo, J., Yoo, S., and Choi, K. (2009). Topology/floorplan/pipeline co-design of cascaded crossbar bus. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 17(8):1034–1047.