UNIVERSITY OF GLASGOW

Quarc: An Architecture For Efficient On-Chip Communication

by

Mahmoud Moadeli

A thesis submitted in partial fulfillment for the degree of Doctor of Philosophy

in the

Faculty of Information and Mathematical Sciences

Department of Computing Science

June 2010

Declaration of Authorship

I, Mahmoud Moadeli, declare that this thesis titled, 'Quarc: An Architecture For Efficient On-Chip Communication' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

Dedicated to my parents

UNIVERSITY OF GLASGOW

Abstract

Faculty of Information and Mathematical Sciences
Department of Computing Science

Doctor of Philosophy

by Mahmoud Moadeli

The exponential downscaling of the feature size has enforced a paradigm shift from computation-based design to communication-based design in system on chip development. Buses, the traditional communication architecture in systems on chip, are incapable of addressing the increasing bandwidth requirements of future large systems. Networks on chip have emerged as an interconnection architecture offering unique solutions to the technological and design issues related to communication in future systems on chip. The transition from buses as a shared medium to networks on chip as a segmented medium has given rise to new challenges in system on chip realm.

By leveraging the shared nature of the communication medium, buses have been highly efficient in delivering multicast communication. The segmented nature of networks, however, inhibits the multicast messages to be delivered as efficiently by networks on chip. Relying on extensive research on multicast communication in parallel computers, several network on chip architectures have offered mechanisms to perform the operation, while conforming to resource constraints of the network on chip paradigm. Multicast communication in majority of these networks on chip is implemented by establishing a connection between source and all multicast destinations before the message transmission commences. Establishing the connections incurs an overhead and, therefore, is not desirable; in particular in latency sensitive services such as cache coherence.

To address high performance multicast communication, this research presents Quarc, a novel network on chip architecture. The Quarc architecture targets an area-efficient, low power, high performance implementation. The thesis covers a detailed representation of the building blocks of the architecture, including topology, router and network interface. The cost and performance comparison of the Quarc architecture against other network on chip architectures reveals that the Quarc architecture is a highly efficient architecture. Moreover, the thesis introduces novel performance models of complex traffic patterns, including multicast and quality of service-aware communication.

I would like to thank my supervisor, Dr Wim Vanderbauwhede for his extraordinary support and guidance. Indeed, this thesis would not have been possible without relying on his deep knowledge, patience and support. I owe my deepest gratitude to Professor Mohamed Ould-khaoua and Dr. Ali Shahrabi for their advise on communication mod- eling and to Partha Maji for his advise on improving the hardware implementation. I acknowledge that in this thesis, the hardware implementation of the architecture using Verilog is mainly carried out by Partha Maji. Also, it has been an honour to work with Dr. Leif Azzopardi who opened my eyes to the interesting field of information retrieval. Special thanks go to Dr. Fernando Rodriguez and Professor Irfan Awan for their valuable comments on the thesis in the viva. I also thank all my colleagues for their comments and help, and the staff of the Department of Computing Science for their kind and friendly attitude.

I would also like to thank my mother for the unending love, protection and encourage- ment she has given me throughout my life. I owe equal gratitude to my greatest source of inspiration, my father. I feel delighted that one of their wishes by this achievement has come true. Thanks also go to my brothers and sisters for their continuous encouragement and support.

Contents

List of Figures

List of Tables

Abbreviations

BE	**B**est **E**ffort
BRCP	**B**ase **R**outing **C**onformed **P**ath
CDC	**C**lock **D**omain **C**rossing
CDG	**C**hannel **D**ependency **G**raph
DSP	**D**igital **S**ignal **P**rocessor
FPGA	**F**ield **P**rogrammable **G**ate **A**rray
GALS	**G**lobally **A**synchronous **L**ocally **S**ynchronous
GS	**G**uaranteed **S**ervice
GT	**G**uaranteed **T**hroughput
HOL	**H**ead **O**f **L**ine
ILP	**I**nteger **L**inear **P**rogramming
IP	**I**ntelectual **P**roperty
MIN	**M**ulti-stage **I**nterconnection **N**etwork
NI	**N**etwork **I**nterface
NoC	**N**etwork **o**n **C**hip
OCP	**O**pen **C**ore **P**rotocol
PEPA	**P**erformance **E**valuation **P**rocess **A**lgebra
QoS	**Q**uality **o**f **S**ervice
SoC	**S**ystem **o**n **C**hip
TDM	**T**ime **D**ivision **M**ultiplexing
VC	**V**irtual **C**hannel
VCI	**V**irtual **C**omponent **I**nterface
VOQ	**V**irtual **O**utput **Q**ueing

Part I

Preliminaries

Chapter 1

Introduction

The advances in semiconductor technologies have enabled the integration of all components of a complicated system on a relatively small chip. This concept is referred to as System-on-Chip (SoC). By the further downscaling of the feature size and at the same time a growing demand for more functionality, the number of IP modules on a single chip are increasing. Hence, SoCs with hundreds of IP cores are becoming a reality.

Exploiting the full potential of the powerful concurrent IP cores to successfully meet the demands of the applications to a large extent depends on the performance of the communication architecture. The role of communication in future SoCs is so fundamental that SoC development has been forced to a paradigm shift from traditional computation-based design to communication-based design. The interconnection architecture for future SoCs must allow a high level of task-level parallelism and offer a significant aggregated bandwidth.

Buses and point-to-point connections as traditional SoC communication mediums are being stretched to their limits. Buses are inherently non-scalable and their power consumption is proportional to the number of IP cores attached. Moreover, bus arbitration when used by several masters is not trivial. Hierarchical buses with sophisticated protocols and multiple bridges between them have emerged to offer more bandwidth (e.g. PCI Express and AMBA). In such an architecture, communication between nodes in different levels are made via several buses. Timing closure is a growing problem due to significant overhead involved in excessive check requirements. Having a simple structure and offering standard communication protocols are major advantages of buses.

Point-to-point connections as an alternative communication medium in SoC development offer ultimate flexibility, more aggregated bandwidth and more communication parallelism. In point-to-point connections, typically, computation is intertwined with communication and the wiring complexity grows exponentially with the number of IP cores. Adopting point-to-point connections in most large SoCs involves using long links to connect IP cores. To avoid signal degradation, long links must be supplied by full swing signaling which, in addition to high power consumption, can increase crosstalk and parasitic capacitance side effects on adjacent links. Those effects can easily cause uncertainty and bugs in application behavior which varies from one run to another. This type of unreliability in application behavior is extremely difficult to debug. Therefore, drawbacks of the point-to-point interconnections far outweigh any benefits for future SoCs.

Crossbar switches are another type of high performance communication architecture for SoC development. Crossbar switches with full connectivity are prohibitively expensive to be a right candidate for future large SoCs.

1.1 Networks on Chip

Networks on chip (NoC) have emerged as a natural evolution of SoC interconnects to address the design and technological challenges of communication in large SoCs [1–8]. NoCs promise to deliver the following characteristics and functionality to the SoC community:

- NoCs structure and manage wires in deep sub-micron technologies to avoid the crosstalk and parasitic side effects [1–4].
- NoCs are energy efficient and reliable. Unlike the bus-based interconnections, in a NoC-based system, IP cores do not have to constantly monitor the data on the bus. This can account for a significant power saving. Also, structuring the links leads to more reliable communication and allows driving the links by much less than the conservative full-swing signaling power. Hence, communication in a NoC-based SoC is more power-efficient and predictable [1, 3].

- NoCs allow efficient wire utilization through sharing the physical links between the communicating IP modules [3, 8, 9].
- NoCs scale better than buses and point-to-point interconnections [5, 8]. Depending on the topology of a NoC, the network can be extended by adding links and routing elements. Extending the network increases the aggregated throughput the network offers.
- NoCs decouple computation from communication through well-defined interfaces, enabling IP modules and interconnect to be designed in isolation, and to be integrated more easily [2, 10].
- NoCs offer a *Globally Asynchronous Locally Synchronous* (GALS) development paradigm by acting as an adapter between the IP modules operating at different frequencies [11, 12].

Figure 1.1 shows a NoC employed as on-chip communication architecture. The figure represents a mesh network connecting 16 IP modules. Such an architecture functions as a packet-switched network to interconnect IP modules. The main components of the communication architecture are routers, links and network interfaces (NI).

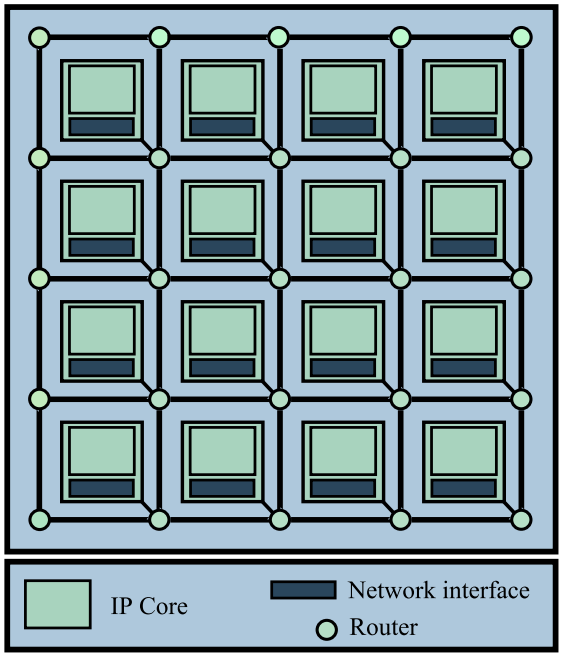


FIGURE 1.1: A NoC-based SoC.

The links and the routers implement the functionality of physical, data-link and network layers of OSI reference model [13]. The routers implement routing algorithm, switching technique and flow control mechanism. The routers also have buffers which their granularity depends on the switching technique.

The network interface functions as a glue between computation and communication by implementing the interfaces to both network and the IP cores. In most NoC architectures the network interface implements the functionality of the transport and session layers of OSI reference model.

An IP core can be any computation or storage component that has been employed in traditional SoC development, including processor, memory, FPGA, and DSP. A core may be comprised of a number of IP cores which are connecting to each other by any communication architecture including a NoC.

1.2 Thesis motivation

Despite being a relatively emerging field, Networks-on-Chip has attracted the attention of researchers in both industry and academia. The NoC paradigm has witnessed numerous NoC architectures spanning from simple proof of concept to full-fledged and versatile schemes. The proposed architectures have mainly focused on delivering high performance unicast traffic. This is a rational decision since most of the on-chip traffic is formed by unicast communication. However, to successfully replace traditional interconnect architectures, NoCs have to be able to deliver all types of communication efficiently.

The applications running on a NoC-based SoC, typically, have some performance requirements. Before adopting a particular NoC architecture, the performance requirements of the application must be matched against the NoC deliverable performance. Therefore, the NoC community must have access to tools and techniques to evaluate the performance of the NoC architectures.

The motivations behind this research were:

- To propose a novel NoC architecture to efficiently deliver unicast and multicast traffic and

- To develop analytical models of complex traffic patterns including multicast and QoS-aware communication in Networks-on-Chip.

1.2.1 A novel NoC architecture

By leveraging the advantages of a shared medium, buses have been highly efficient in performing broadcast and multicast communication. The segmented nature of a NoC, however, does not allow a message to be delivered to multi-destinations as efficiently. Due to similarities between NoCs and interconnection networks for parallel computers the NoC community has mainly relied on experiences in parallel computers to deliver multicast messages. A number of NoC architectures such as Æthereal [14] and Nostrum [15] proposed mechanism to perform multicast communication in the NoC domain. All architectures rely on setting up a connection between source and destinations before starting the message transmission.

To address the need for efficient multicast message delivery, the thesis proposes the Quarc NoC. In contrast to earlier architectures which offered multicast as an additional service to the NoC architecture, multicast communication has been the main motivator of the Quarc NoC scheme. This work present a full and detailed implementation of all components constituting the Quarc NoC.

1.2.2 Analytical modeling of NoC traffic

Analytical modeling as one of the most cost-effective methods has been widely used to evaluate the performance of interconnection networks. The most demanded performance measures of interest in an interconnection network are average message latency, throughput and latency jitters. Evaluating message latency has been paid a special attention, as many services and applications such as cache coherence and multimedia are latency sensitive. The literature has seen numerous analytical models to predict message latency in networks with variety of configurations. However, the majority of the models have been proposed for unicast traffic [19, 20], and analysis of more complicated traffic patterns including collective and QoS-aware communication have not been paid much attention. Shahrabi et al.[21] introduced a model for predicting the broadcast communication latency in a Hypercube-topology network. However, in their system model only unicast

traffic was wormhole-routed, and broadcast communication was not wormhole-routed. Also, their model was developed for architectures using *one-port* routers.

The interconnection architectures employing an *all-port* router scheme have been known for their superiority in delivering high performance multicast communication operations [122, 123]. The performance of all-port router architectures, in particular the wormhole-routed ones, has always been evaluated using simulations programs. Developing an analytical model to address such communication architectures is regarded as a step forward.

Most applications running on a system on chip must meet a minimum performance to operate successfully. To address the performance demands of the application running on a NoC-based SoC, most architectures offering guaranteed or best-effort QoS support. In particular, differentiated services-based QoS is implemented as the only QoS provisioning mechanism [9, 22, 23] or has been offered in combination with guaranteed services [24–29]. Evaluation of the performance of interconnection networks adopting differentiated services-based QoS in parallel computers and NoC paradigms have always been based on simulation or prototyping. Developing an analytical model to evaluate the performance of prioritized traffic will certainly benefit the community by leveraging the cost-efficient use of the analytical models to evaluate the performance of QoS-aware NoCs.

1.3 Thesis contributions

The contributions of the thesis cover the following aspects of NoC research: architecture, analytical modeling and performance/cost comparison.

1.3.1 Architecture

The Quarc NoC The introduction of the Quarc NoC is one of the most fundamental contributions of the thesis. The Quarc NoC is introduced as an all-port router architecture offering high performance communication at low cost. Key characteristics of the Quarc NoC include: even distribution of traffic in network, highly efficient multicast communication support and low implementation cost.

The Quarc network interface As the interface between computation and communication, the network interface plays a fundamental role in delivering the services required by a NoC-based application. This thesis presents a modular design and implementation of the network interface in the Quarc architecture.

1.3.2 Analytical modeling

Communication modeling of multicast traffic Multicast is one of the most widely used collective communication operations in interconnection networks. The thesis introduces a novel analytical model to predict the average message latency of multicast traffic in wormhole-routed interconnection networks employing all-port router scheme.

Modeling differentiated services-based QoS traffic Differentiated services-based QoS is implemented by most architectures addressing QoS. The thesis develops an analytical model to predict the average message latency of prioritized traffic.

1.3.3 Performance/cost evaluation

Performance and cost are two paramount factors in adopting a particular interconnection architecture for SoCs. To evaluate the performance and cost of the Quarc NoC, the thesis presents a comparison against two architectures.

The Quarc NoC versus the Spidergon STNoC The Quarc NoC is inspired by the Spidergon STNoC [30] and can be considered as an improvement to the architecture. To investigate the effect of the modifications on performance and cost, hardware implementations of both architectures are developed. Moreover, the performance of the two architectures has been compared in different configuration settings.

The Quarc NoC versus the mesh-based architecture Mesh is the most widely used network topology in the NoC domain. The thesis present an extensive comparison between the Quarc NoC and an architecture adopting mesh as its topology. Hardware implementations of the two architectures are developed for various configurations. Also,

an extensive performance comparison between the two architectures in various working configurations is provided.

1.4 Thesis overview

The thesis comprises twelve chapters. The chapters are organized in five parts depending on the subjects covered.

Part I: Preliminaries Part I consists of the first two chapters of the thesis. Chapter 1 presents and introduction to the thesis and Chapter 2 gives the preliminaries required for understanding the following chapters. It starts with an overview of the characteristics of a NoC including topology, routing algorithm, switching technique and buffering strategy. The chapter follows by presenting a literature overview of a number of the NoC architectures.

Part II: The Quarc NoC Part II consists of Chapters 3 and 4. Chapter 3 gives an insight into the Quarc NoC. Topology, routing algorithm, switching technique and buffering strategy of the Quarc NoC along with a hardware implementation of the Quarc router is presented in this chapter. Chapter 4 discusses the network interface in the NoC realm and provides a detailed design and implementation of the network interface in the Quarc NoC.

Part III: Cost and performance evaluation Chapters 5 and 6 form Part III of the thesis. Chapter 5 presents a cost and performance comparison between the Quarc NoC and the Spidergon STNoC. A cost and performance comparison of the Quarc NoC against a mesh-based architecture is covered in Chapter 6.

Part IV: Analytical performance modeling Part IV includes Chapters 7, 8, 9, 10 and 11. Chapter 7 presents an overview of the tools and techniques to evaluate the performance of interconnection networks and investigates their suitability in the NoC domain. Chapter 8 presents an analytical model to predict the message latency of unicast communication in wormhole-routed interconnection networks. The unicast model

presented in the chapter is used in the following chapters to evaluate the performance of more complicated traffic. Chapter 9 introduces a novel analytical model to predict the latency of multicast communication in wormhole-routed interconnection networks employing all-port router. Chapter 10 covers the mechanisms employed to offer QoS in interconnection networks and introduces an analytical model to predict the communication latency in QoS-aware wormhole-routed interconnection networks. Chapter 11 presents a model of broadcast communication in QoS-aware interconnection networks by combining the models presented in Chapters 9 and 10.

Part V: Conclusion and trend for further research The concluding remarks and the trend for future research are discussed in Chapter 12.

Chapter 2

Network on Chip Characteristics

The rich literature on the design of system-level interconnection networks can, to a large extent, be applied to NoCs. However, on-chip networks present a number of unique challenges that require solutions distinct from the tried-and-tested system-level techniques. Perhaps the biggest difference between system-level networks and NoCs is their cost structure [3]. For system-level networks, the major cost lies in the links. The optimal system-level network is one that delivers the required performance with a minimum number of links (and in particular a minimum number of long, and hence costly, links). In contrast, the on-chip links used to realize NoCs are constructed from inexpensive on-chip wires. The dominating factors in cost of a NoC are switches and buffers, and not the links. Hence, the optimal NoC is one that minimizes switch and buffer area, often at the expense of more links [3]. This difference in cost structure motivates the use of very different topologies, routing algorithms, and flow-control methods in a NoC than would be used in a system-level network.

This chapter presents the characteristics of NoCs in terms of topology, routing algorithms, switching techniques and buffering strategies. The chapter also provides an overview of a number of NoC architectures.

2.1 Topology

The topology of a NoC describes the physical interconnection structure of the network graph. Adopting a particular topology for a NoC is paramount and is typically the result

of a trade-off between cross-cutting metrics such as performance and cost. The topology of a network affects the scalability, performance, complexity of the routing elements, fault tolerance and power consumption.

An on-chip network can be direct or indirect. In direct on-chip networks each router in the architecture is connected to neighboring routers and using a network interface it is also connected to an IP core e.g. processor, memory or DSP. In indirect on-chip networks the number of routers in the architecture is larger than the number of IP cores. In such on-chip networks, a number of routers are connected only to other routers. Example of indirect networks are crossbars and multistage interconnection networks.

Similar to other interconnection networks, NoC topologies are evaluated by several metrics. Topology comparison is usually based on theoretical cost and performance measures including number of nodes, network degree, diameter, average distance, bisection width, number of edges, extendability, symmetry and routing strategy.

The NoC literature has witnessed several topology-dependent and topology-independent architectures. The topology-dependent architectures are built on top of a regular topology. The topology-dependent architectures can leverage the benefits of fully verified research on their underlying topology. The topology-independent architectures, however, can offer more customizability which is important in NoC-based SoC development.

The following sections present an overview of the most widely-used regular topologies in the NoC domain.

2.1.1 Multistage interconnection networks

Multistage interconnection networks (MINs) are a class of high performance indirect networks. In MINs the input and output devices are connected by a number of stages of crossbar switches. Omega [31] and delta [31] networks are examples of MINs. MINs were first used in telephone switching application. They are also used as high performance interconnection networks for parallel computers, e.g. Thinking Machines CM-5 [32].

MINs have been employed as NoC interconnection topologies. Fat tree [5] and butterfly fat tree [33] are examples of NoCs adopting MINs as their network topologies. Despite a rather large network cost and extendability, small locality, increased wire density (with

many long interconnection wires), and rather expensive VLSI layout, some variations of MINs can be considered for NoC realization in data parallel applications, such as multimedia processing [30].

2.1.2 Mesh and torus

Two-dimensional mesh has been regarded as the most favored regular topology for NoC architectures. This is mainly due to the regularity, scalability and ease of synthesis on chip the topology offers. Mesh also can provide an acceptable wire cost and reasonably high bandwidth.

The literature has witnessed extensive research on square mesh in a variety of applications including parallel computing and the NoC domain. In the NoC paradigm, however, it is more likely that the chip has dimensions of different sizes and presents a rectangular shape. The rectangular mesh may be considered as an appropriate candidate in such circumstances. Despite its wide use, the work on rectangular mesh has been insignificant.

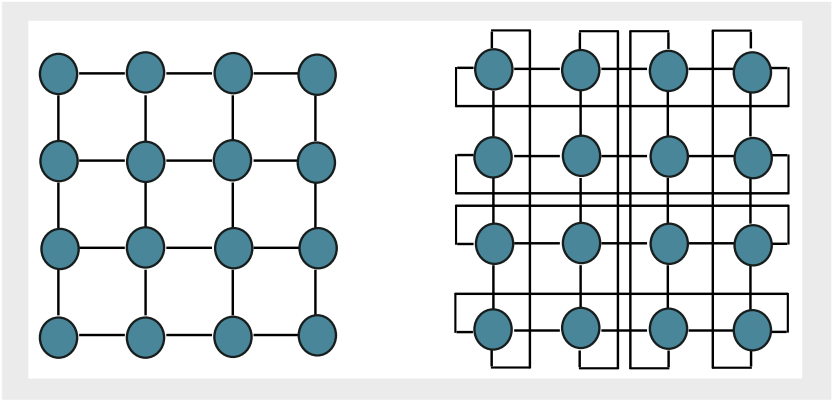
The adjacent nodes in a mesh may be connected by *unidirectional* or *bidirectional* links. The mesh topology may be formally presented as follows. Suppose that the network has $m \times n$ nodes where m and n are the number of nodes at x and y dimensions, respectively. Each node in network is identified by its position at x and y dimensions. The indices $< i, j >$ where $0 \leq i < m$, $0 \leq j < n$ are denoting the position of the node at x and y dimensions, respectively. Typically, the node at the top-left corner is assigned label $< 0, 0 >$ and the x-dimension and y-dimension indices are incremented as we move to the right and bottom, respectively. The label of each link in network may be determined by the label of the node it is connected to and the relative position (N, S, E, W) of the link to the node. For example, $< i, j, E >$ is the label of the link connecting the node $< i, j >$ to node labeled $< i + 1, j >$.

In a mesh network deadlock may be avoided by adopting the XY routing algorithm. In XY routing the packets pass the links in different dimensions according to a predefined priority.

Having a relatively high network diameter is a drawback of the mesh topology. The two-dimensional torus reduces the 2-D mesh diameter by adding wrap-around links. These

links run between $< i, n-1 >$ and $< i, 0 >$ for all $0 \leq i < n$, and between $< n-1, j >$ and $< 0, j >$, for all $0 \leq j < n$.

A major benefit of torus compared to mesh is the smaller diameter ($2 \lfloor n/2 \rfloor$ vs. $2n-2$) and larger bisection width ($2n$ vs. n). Moreover, notice that compared to a mesh, a torus has higher structural uniformity, since it is vertex- and edge-symmetric. Another advantage of torus over mesh is that unlike mesh, torus distributes traffic more evenly at network links. In mesh architectures using XY routing, the links in the middle of network are more heavily utilized compared to the links near the edges. Figure 2.1 represents 4×4 mesh and torus topologies.



(a) (b)

FIGURE 2.1: 4×4 (a) mesh and (b) torus topologies.

2.1.3 Ring and chordal rings

Rings and variations of the ring topology have been heavily researched in interconnection networks for parallel computers. Chordal rings are a variation of the ring topology. The following section presents a brief description of the family of graphs that can formally define chordal rings, followed by a discussion of rings and chordal rings.

2.1.3.1 Cayley graphs

Most constant degree topologies introduced for interconnection networks can be represented by the well known family of *Cayley* graphs [34, 35]. Cayley graphs are based on algebraic group theory. Nodes of the Cayley graph are elements of a permutation group, G. The edges of the graph are based on applying the group generator operator ($\oplus$) .

Node x connects to node y, iff $x \oplus \gamma_i = y$ for some $\gamma_i \in S$, where S is a set of group generators for G. Cayley graphs share many interesting topological properties. For example, all Cayley graphs are vertex symmetric. This rich family of graphs can be used to generate small degree, low diameter networks. Moreover, almost all Cayley graphs are Hamiltonian, and many are hierarchically recursive and optimally fault tolerant.

2.1.3.2 Circulant networks

Circulant networks (also called circulants) [30, 35] are special Cayley graphs and digraphs defined on cyclic addition groups $\mathbb{Z}_N$, where N is a positive integer greater than or equal to 3. Circulant networks are formally defined as follows. Let $G(N; s_1, s_2, \cdots, s_k)$ be the digraph with N nodes, labeled with elements from $\mathbb{Z}_N$, and each node x is adjacent to k other nodes $x + s_i$, $i = 1, 2, \cdots, k$. The graph $G(N; \pm s_1, \pm s_2, \cdots, \pm s_k)$ is the undirected version of $G(N; s_1, s_2, \cdots, s_k)$, where each node x is adjacent to $2k$ other nodes $x \pm s_i$, $i = 1, 2, \cdots, k$; naturally, $s_1, s_2, \cdots, s_k$ are called skip distances. The addition or subtraction is taken modulo N.

2.1.3.3 Rings

The ring topology is a popular topology for many communications and parallel processing applications. This is due to its structural simplicity and very efficient routing protocols. These properties contribute to low implementation cost and high transmission throughput, with high latency being a potential drawback. Even though a single node or link failure will not disconnect the nodes of an n-node ring, it has an effect on the routing strategy and tends to further increase the transmission latency. More importantly, two failures (node/node, link/link, or node/link) will almost certainly isolate some parts of network.

2.1.3.4 Chordal rings

One approach to improve the robustness, network diameter and average distance of the ring topology entails addition of skip links or chords to the ring network. The augmented ring is called a chordal ring and can be formally defined as a Cayley graph on $\mathbb{Z}_N$, that

is as circulant graphs, for which 1 and -1 belong to the set of generators. These graphs are obtained from the cycle by adding chords to each vertex in a regular manner.

Chordal rings have been studied extensively for use as communication and parallel processing networks [36–40]. Applications of chordal rings to parallel systems dates back to very early in the history of parallel processing and have continued to date [41, 42], although in some cases the interconnection structures include subtle variations and carry different names, thus making it difficult to identify the underlying chordal ring networks. The bulk of studies of chordal rings in relation to interconnection networks deal with networks of small, fixed node degrees; most commonly, 3-6 for undirected (4 being most heavily studied [43, 44]), and 2-3 for directed networks [45, 46].

2.2 Routing

The routing algorithm determines the path from source to destination. The routing algorithm for an on-chip network or in general for an interconnection network is significantly affected by the topology of network. Numerous routing algorithms have been proposed and adopted by the interconnection networks for parallel computers. However, the peculiarities of network on chip inhibits employing most of the contributions in the on-chip domain. The NoCs must implement efficient routing algorithms without using the routing tables and complex arbitration protocols, targeting small area and high frequency implementation.

Following presents a brief introduction to the most widely-used techniques to implement the routing algorithms in interconnection networks.

2.2.1 Deterministic routing

Deterministic routing is a type of distributed routing algorithm in which the path between source and destination is always the same. Due to resource constraints of the NoC development, deterministic routing algorithms are the most favored options for on-chip networks [47]. In a distributed routing algorithm, each intermediate router computes the next link to be followed. The algorithm requires only the destination address for deciding the next link. Deterministic routing algorithms mainly follow the shortest path.

Because of the prevalence of the mesh topology in the NoC domain, the most widely used type of deterministic routing is XY routing, a variation of dimensional routing.

2.2.2 Adaptive routing

Adaptive routing is another variation of distributed routing. In contrast to deterministic routing, adaptive routing allows more than one path between source and destination. In adaptive routing, the decision on the next link at each intermediate router is not based on the destination address only. The traffic information at each possible route is also taken into account. The adaptive routing algorithms may lead to non-minimal paths, which are typically undesirable and more prone to deadlock and livelock. For irregular traffic or hot spots, adaptive routers usually outperform deterministic ones [30]. However, the incurred cost for implementing those algorithms is not affordable in most NoC architectures. In addition to increasing the complexity at routers, employing adaptive routing requires enough buffer at the destination to store the packets which potentially arrive out of order.

2.2.3 Source routing

In source routing, the routing information on entire path is provided by the source node. The packet includes the *route*, i.e. an ordered list of the addresses of all intermediate nodes. On arrival to an intermediate router, the routing field is typically shifted in order to expose the relevant routing choice for the next router on its path.

Source routing generally works with discovery packets that dynamically discover routes from the source node to any required destination. So it is a very dynamic mechanism. The drawback of course is that the header size grows with the number of nodes. Another drawback is that the route discovery loads network.

Source routing represents a flexible and cheap solution for on-chip networks and has been implemented in several NoCs including Æthereal [48] Chain [49] and Spidergon [30]. Source routing is an appropriate candidate to handle potential network faults [50].

2.3 Switching

The switching techniques determine when and how internal switches are set to connect router inputs to outputs and the time at which message components may be transferred along the paths. Switching typically has a more significant impact on performance than routing and topology do [31]. An interconnection network may adopt circuit switching, store-and-forward switching, virtual cut-through or wormhole switching.

2.3.1 Circuit switching

In circuit switching, a physical path from source to destination has to be established before the message is transmitted. To establish the path a probe header is injected into the network. The header contains the destination address and other control information. As the header moves towards destination the links in the path are reserved. Once the header reaches the destination an acknowledgment is sent back to source to inform it to send data. The links will be released by source or destination after the information transmission is accomplished.

Circuit switching is appropriate for situations in which messages are long in comparison to the header and message exchange is infrequent. Otherwise, the network experiences unacceptable latency and blocking rate.

In the NoC domain variations of circuit switching are adopted to offer guaranteed services or multicast support [14, 51].

2.3.2 Store-and-forward switching

In store-and-forward switching each message is split into packets and each packet is sent to the network individually. Each packet has routing information to reach destination. At each intermediate router, after the entire packet is stored at the local buffer, the routing information is extracted from the packet and an output port is determined according to the routing algorithm.

In store-and-forward switching no reservation is required (unless QoS demands dictate otherwise). Store-and-forward switching is considered to be an appropriate choice when

messages are frequent and short. A major drawback of store-and-forward switching is the large storage requirement at intermediate routers. The situation worsen when packets are large and multiple packets must be buffered at a node. Moreover, in store-and-forward switched networks the latency is distance-sensitive. These characteristics are in contrast to the NoC demands for small buffers (at routers) and high performance in terms of latency.

2.3.3 Virtual cut-through switching

In store-and-forward switching the decision on identifying the output port is made after the entire message is placed at the local buffer of the intermediate router. Since the routing information is usually in the first few bytes of the packet, it is possible to make routing decision before the entire packet is received and send the incoming flow to output buffers if any available. Similar to store-and-forward switching, if the output port is busy the packet is stored at the buffer of the intermediate node. This modified version of store-and-forward switching was presented by Kermani et al. and is called *virtual cut-through* [52].

In the absence of blocking, the latency experienced by the header at each node is the routing latency and propagation delay through the router and along the physical links. At high network loads, in which header experience blocking at each router the latency of virtual cut-through is the same as store-and-forward switching. Virtual cut-through shares the drawback of the demand for large amount of buffers with store-and-forward switching technique.

2.3.4 Wormhole switching

Wormhole switching [53, 54] has been introduced to resolve memory issues in store-and-forward switching and virtual cut-through switching. In wormhole switching packets are split into smaller units of flow control called flits. Flits are the smallest unit of data which require synchronization between the sender and receiver before transmission. Flits are comprised of *phits*. The size of a phit equals to bandwidth of the link and is transmitted in only one clock cycle.

The header flit has the routing information and the following flits pursue it in a pipelined fashion. Each intermediate router has enough space for up to a few flits which is considerably less than buffer size required in store-and-forward switching. Usually the flits of a packet occupy several links and stay at their current link buffers, if the header flit blocks.

Holding multiple links at the same time by a worm increases the channel dependency and consequently makes network more prone to deadlock. This issue is specially important when performing collective communication operations.

2.4 Deadlock and livelock

Mutual dependency among the resources occupied by two or more packets may lead to deadlock [31]. In such situations none of the involved packets are able to move further. The possibility of experiencing deadlock is in direct relation to the number of links network must acquire to send a packet. In store-and-forward switched networks dependency exists only between adjacent nodes. In contrast, in wormhole switched networks a message can occupy several links at a time. Holding multiple links at the same time may lead to mutual dependency between the links, if routing algorithm is not deadlock free.

To resolve the deadlock problem, mutual dependencies among the network resources must be broken. The solutions to prevent and avoid deadlock may be realized by employing appropriate routing algorithm, e.g. *Dimension-ordered routing* and *Turn-model routing* [55] in mesh. Deadlock may also be avoided by employing virtual channels or virtual networks. The Spidergon scheme [30] is an example of the network topologies that breaks the channel dependencies by adding virtual channels to physical links. In the Spidergon architecture a number of virtual channels are used only as escape channels to avoid deadlock.

Livelock is similar to a deadlock, except that the states of the processes involved in the livelock constantly change with regard to one another, none progressing. Adaptive routing algorithms are more prone to livelock.

2.5 Buffering strategies

The buffering strategy determines the location of the buffers inside the router. Typically, a router may adopt an output queuing or input queuing strategy. In input queuing the buffers are at the input of the router. There is requirement for queuing at each input. To avoid contention, a scheduler decides which queues are connected to which output port. The long-standing view has been that input-queued switches exhibit poor performance due to *head of line* (HOL) blocking; if the packet/flit at the front of the queue is blocked, other packets/flits in the queue cannot be forwarded to other unused inputs. For large-degree routers, router utilization saturates at 59% of the network capacity [56].

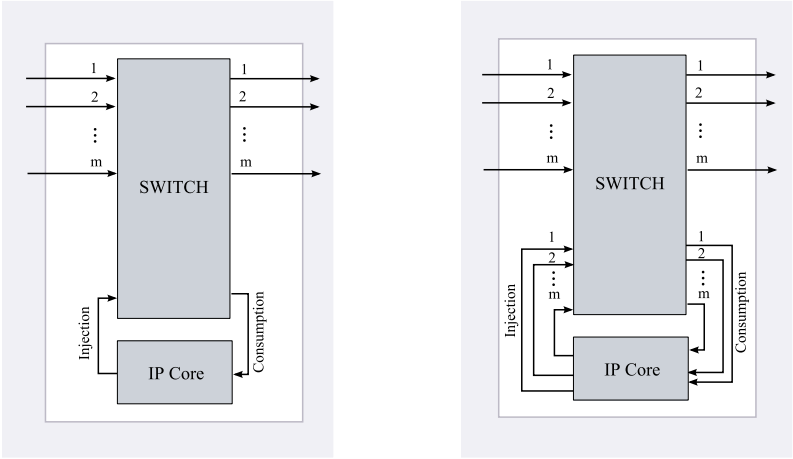
A solution to the HOL issue is to employ an output queuing buffering strategy. In output queuing the number of buffers at each output port equals to number of all inputs links. From the inputs to the outputs there is a fully connected bipartite interconnect to allow every input to write to every output. Output queuing has the best performance among the buffering strategies, however, the interconnect will make the router wire dominated and expensive already for small-degree routers.

Another version of input queuing is virtual output queuing (VOQ) [57]. VOQ combines the advantages of input queuing and output queuing. It has a switch like in input queuing and has the link utilization close to that of output queuing. VOQ has as many buffers as output queuing strategy. However, the switch can operate at a lower frequency [58].

In Networks-on-Chip, buffers are too costly in terms of area and power consumption [47]. Therefore, the VOQ and output queuing strategies are not considered as an appropriate option in the NoC realm. Moreover, since most NoC schemes employ wormhole-switching a packet typically spans several routers, and if the header flit is blocked other flits are stalled. This is an undesirable feature of the wormhole switching. To overcome this problem, most NoCs adopt input queuing along with employing virtual channels for a smooth flow control.

2.6 Router architecture

In direct interconnection networks, a router is connected to other neighboring routers through a number of external links. The router is also connected to the local node via one or more internal links. The architectures adopting only one internal link are referred to as *one-port* architectures. Increasing the number of internal links significantly improves the performance of the collective communication operations [123]. Architectures having an internal link corresponding to each external link are referred to as *all-port* architectures. The schematic of the router in a one-port and all-port router architectures are depicted in Figure 2.2.



(a) (b)

FIGURE 2.2: (a) One-port versus (b) all-port router architecture.

2.7 Collective communication operations

The dominant type of communication in an interconnection network is one-to-one. In such a communication only two processes are involved. There are however, situations where more than two processes participate in communication. This type of operation is referred to as a collective communication operation. Collective communications operations have been traditionally adopted to simplify the programming of applications for parallel computers, facilitate the implementation of efficient communication schemes on various machines, and promote the potability of applications across different architectures [31]. These communication operations are particularly useful in applications which

often require global data movement and global control in order to exchange data and synchronize the execution among processes.

The collective communication operations can be classified into three categories: one-to-many, many-to-one, and many-to-many. In one-to-many communication one process sends a message to a number of processes in the system. The source may send the same message to the destinations (multicast) or it may send different messages to different destinations (scatter). Many-to-one communications can be viewed as an inverse of one-to-many communication. One process receives the message from a number of other processes. An example of this communication is gather which allows the receiver to collect information from other processes. In many-to-many communication, all processes in a collective communication group are sender and receiver. An example of such operation is all-to-all broadcast.

The collective communication operations defined above are based on an application perspective of the operations. This type of definition is useful for studying the semantics of such operations. However, such a view may not be the most appropriate one when the focus is on the interconnection network. Therefore, it is important to separate the application perspective and system view of such operations. Since this work is more concerned about the performance evaluation of the system level and physical interconnection architecture issues, this study focuses on collective communication in terms of physical network architecture.

2.7.1 Multicast implementation

Among the collective communication services, multicast is the most frequently used operation. Multicast is adopted in control operations such as global synchronization and to signal changes (e.g. faults) in network condition or availability of the IP cores. In distributed shared-memory paradigm, multicast is used to support shared data validation and updating procedures for cache coherence protocols [118]. Moreover, multicast is used to implement other collective communication operations such as barrier-synchronization. Since these operations are typical operations in interconnection networks, a large body of research has been allocated to provide platforms offering efficient multicast operation.

The support for multicast communication may be implemented in two groups depending on whether the underlying technique requires special hardware support. In the first class, which is called software-based, each multicast operation is reduced to a sequence of exchanges of unicast messages. In software-based approach the multicast operation is implemented as an enhancement layer sitting on top of an existing reliable unicast communication service. In contrast, in hardware-based multicast implementation network is enhanced via special hardware to support multi-destination message transmission.

2.7.1.1 Software implementation

Software-based schemes, which are also referred to as unicast-based, are implemented as a sequence of unicast message exchange. Separate addressing is the most simple unicast-based scheme, in which separate messages are sent to each destination [119]. This method performs poorly not only because it wastes network bandwidth due to excessive traffic generated by only one node, but also because it involves several start-up latencies and requires excessive time (especially in a single port architectures in which a local processor may send only one message at a time).

To overcome this, efficient software-based multicast algorithms employing a divide and conquer strategy have been proposed [119–122] and used in communication libraries such as MPI and PVM. According to this approach, nodes form a tree structure which is called a multicast tree. Although any multicast algorithm can also be used, for the special case of broadcast, the design of the broadcast tree may be treated differently.

To multicast a message, a node transmits the message along a spanning tree rooted at its own location; whereby a source node sends the message directly to a subset of destinations, each of which then participate recursively (forming a tree) by re-transmitting copies of the message to the remaining destination nodes. Eventually, all nodes will receive the multicast message. The software-based implementation uses underlying unicast communication to deliver multicast messages and, therefore, does not require any additional hardware support. These schemes aim to reduce the number of start-up phases by allowing some destinations to act like the source node after receiving the message.

The important issue to consider in software-based multicast implementation is reducing the height of the tree and also reduce or eliminate the contention between the branches

of multicast trees. Umesh [119] is proposed to address the contention free multicast problem for mesh networks adopting dimension ordered-routing. The Umesh algorithm is based on the recursive doubling technique. A drawback of the Umesh algorithm is its poor performance in the presence of multiple multicast operations. To overcome this, the SPUmesh [52] algorithm was proposed. The SPUmesh algorithm involves the position of the source node in decision making on partitioning the multicast destinations.

2.7.1.2 Hardware implementation

Software-based approaches typically have limitations in delivering the required performance. Implementing the required functionality partially or fully in hardware has proved to improve performance of the collective operations. Depending on required performance, the hardware support for collective communication may be achieved by customizing the switching [53], routing, number of ports [123] or even allocating a dedicated network for collective communication operations as in the Connection Machine CM-5 [32].

Obviously, employing dedicated hardware for supporting multicast operations significantly improves the performance of the operations. A basic reason for hardware support is to eliminate the need for creation and manipulation of the message at software stacks. Moreover, hardware-based multicast can be designed such that a single multicast message utilizes a common path to cover more than a single destination. Examples of useful hardware support for multicast operations are absorb-and-forward and replication.

A message that is intended for more than one destination is referred to as a multi-destination message [98]. Most of the hardware multicast support is dedicated to offering services to multi-destination messages. The absorb-and-forward operation is a function that may be offered by a hardware router to allow a message to be concurrently forwarded and stored at the local IP core. In replication, the router has capacity to replicate the incoming flits and forward them to different directions.

Hardware-based multicast schemes can be broadly classified into *path-based* and *tree-based* schemes. In a path-based approach, the primary problem for multicasting is finding the shortest path that covers all nodes in the network [31]. After path selection, the intermediate destinations perform absorb-and-forward operation along the path.

The Hamilton path-based algorithm [124, 125] and the Base Routing Conformed Path (BRCP) approach [76, 126] are examples of path-based algorithms utilizing absorb-and-forward property at hardware layer.

In the tree-based scheme, the multicast problem is finding a Steiner tree with a minimal total length to cover all network nodes [127]. The tree operation introduces additional network resource dependencies which could lead to deadlock and is difficult to avoid if global information is not available [128]. Hence, in wormhole-routed direct networks, the tree-based multicast is usually undesirable, unless the messages are very short.

2.7.2 Multicast support in NoC

Broadcast and multicast traffic in NoC is an important research field that has not received much attention. A multicasting scheme for a circuit-switched NoC is proposed in [17]. The scheme relies on the global network state using global traffic information and is therefore not easily scalable. Multicast operation is supported by the Æthereal NoC [14]. However, Æthereal relies on a logical notion of global synchronicity which is not trivial to implement as the system scales. In [16] a multicast scheme in wormhole-switched NoCs is proposed. By this scheme, a multicast procedure consists of establishment, communication and release phases. A multicast group can request to reserve virtual channels during establishment and has priority on arbitration of link bandwidth. In Nostrum [51] the multicast service is offered by allowing multiple destinations on a virtual circuit. In Nostrum the virtual circuits are set up semi-statically, in which the route is decided at design time, but the bandwidth is variable at run-time. Performing a multicast operation in all approaches mentioned above requires setting up a connection explicitly or implicitly before the transmission starts. In XHiNoC [18] the detailed implementation of a wormhole router offering multicast communication service is presented. The XHiNoC router transfers a flit to all its destinations in parallel. The method resembles the tree-based multicast approach hence, sharing the similar drawbacks. Unlike the connection-oriented methods, in the Quarc NoC architecture there is no need for establishing a connection between source and destinations before the data transmission commences. The Quarc NoC employs a *BRCP* multicast routing algorithm and an all-port router architecture to offer high-efficient multicast operation.

2.8 Literature overview

Since the on-chip network concept emerged, the SoC community has witnessed numerous NoC architectures. Those architectures span a wide range from simple theoretical proof of concept to full-fledged and commercial NoCs. A survey of the state of the art and the majority of NoC architectures are presented in [59, 60]. This section represents a selection of on-chip communication architectures.

SPIN: In [5] Guerrier and Grenier have proposed an architecture which is called *SPIN* (Scalable, Programmable, Integrated Network). SPIN is one of the first concrete implementations of switching networks on silicon chips. SPIN uses a 4-ary fat-tree structure as network topology implemented using 8×8 routers, wormhole routing and input queuing. Fat-tree has been proved formally to be the most cost efficient for VLSI realizations [61].

In the SPIN architecture the nodes are routers and the leaves constitute the IPs. The total number of routers can grow as many as the number of IPs. Routing in SPIN is typically adaptive and distributed. Increasing the level of stages reduces contention and improves both message latency and network throughput. In SPIN each two points are linked by two 36-bits wide unidirectional links.

SPIN adopts a credit-based flow control on the paths. The overflows at destination of a path are checked at source. The receiver notifies the sender of every received data, with a dedicated feedback wire. Although the introduction of dedicated control wires is costly, it is shown that this architecture performs very well in terms of latency and throughput [111].

In SPIN, there is no limit on packet size. Each packet consists of a sequence of 4-byte flits, i.e. first flit, data flits and end of packet flit. The first flit (header) has a 1-byte address and remaining bits for special services or routing options.

Figure 2.3 shows a SPIN architecture with 16 IPs and two levels of routers. In this architecture the size of network grows as $(n \log n)/8$ and the number of switches converges to $S = 3n/4$, where n is the system size in terms of the number of functional IP cores.

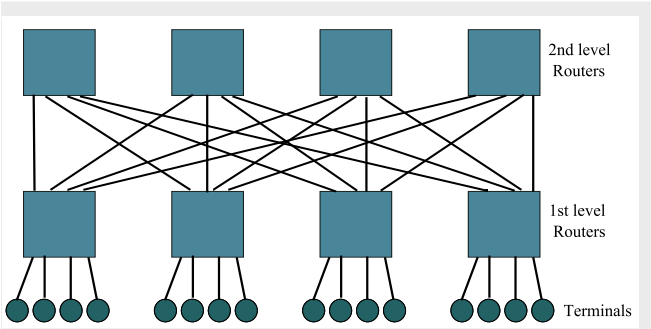


FIGURE 2.3: A SPIN architecture of 16 nodes.

The network interface in SPIN consists of two wrappers (VCI/SPIN and SPIN/VCI) compliant to Virtual Component Interface (VCI) specifications for interfacing the SPIN network with external IP core. A 32-port SPIN network was implemented in a $0.13\mu m$ CMOS process [62] with total area of $4.6mm^2$ and peak bandwidth $\sim 100Gbits/s$.

Butterfly fat tree: Pande et al. [33] proposed the butterfly fat-tree architecture. In their network, the IP cores are placed at the leaves and switches at the vertices. A pair of coordinates is used to label each node to denote the node's level and its position in the network. At the lowest level, there are N IP blocks which are connected through the switches at higher levels. Each switch has four children and two parent ports. The IP cores are connected to $N/4$ switches at first level. The number of levels depends on number of IP cores i.e. $\log_4 N$. Figure 2.4 depicts a 16 nodes butterfly fat tree architecture.

Butterfly fat-tree employs a packet based communication mechanism, adaptive shortest path routing and employs wormhole switching. A packet consists of a header flit and one or more data flits, where the number of data flits is included in the header flit.

This architecture enables simple traffic aggregation to/from a particular set of cores and regular structuring of the switches in the layout, simplifying design. In fact, butterfly fat-tree trades throughput for reducing area overhead and power efficiency, more than the SPIN architecture.

MIT RAW: The MIT RAW microprocessor network addresses the challenge of whether a future general-purpose microprocessor architecture could be built that runs a greater subset of the ASIC applications, while still running the same existing ILP-based

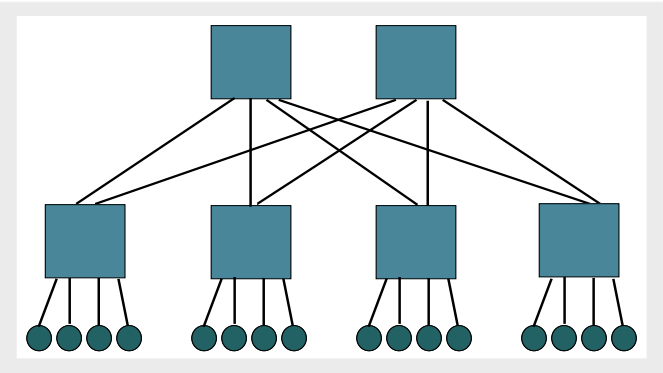


FIGURE 2.4: Butterfly fat-tree.

(Integer Linear Programming) sequential applications with reasonable performance in the face of increasing wire delays [63, 64]. The RAW architecture supports an ISA that provides a parallel interface to the gate, pin, and wiring resources of the chip through suitable high level abstractions, enabling the programmer (or compiler) to determine and implement the best allocation of resources for each possible application.

The MIT RAW design divides silicon area into 16 identical, programmable tiles. Each tile consists of an eight-stage in order single issue MIPS processor, a four-stage pipelined floating point unit, a 32-Kbyte data cache and 96-Kbytes of software-managed instruction cache. Each tile connects to north, east, south or west neighbor tiles using four full duplex 32-bits wide networks, two static and two dynamic. The static router (routes specified at compile time) is a five stage pipeline that controls two physical networks used for point-to-point scalar (operand) transport among tiles. The dynamic routers control two dynamic networks for the remaining traffic, e.g. memory, interrupt, I/O, and message passing. This implementation reduces the longest wire, ensuring scalability. Packets consist of a single header that specifies the destination tile, a user field and packet length. RAW implements fine-grain communication between replicated processing elements with local memory. Thus it is able to exploit parallelism in data parallel applications, such as multimedia processing.

Cliche: The Chip-Level Integration of Communicating Heterogeneous Elements NoC architecture (called Cliche) adopts a 2-D mesh topology with a packet switched communication protocol [7].

Each router is connected through input/output links to one external IP or processor/DSP resource and two, three or four other neighbor routers, depending on its location

in the layout. The architecture defines four layered inter-resource communication protocol (physical, data-link, network and transport layer), which must be implemented in the resource to network interface for every resource in the architecture. Connection to the external resource may require network interfaces.

Folded torus NoC: In [3] Dally discussed the need for networks on chip as an alternative to the bus-based approach. The paper adopted a folded torus as the topology of the NoC architecture. The layout provides extra wiring, reducing the hop count for any transmitted packet. However, reducing misrouted or dropped packets and avoiding deadlock (through virtual channels) require a larger buffer size.

Embedded Chip-Level Integrated Parallel SupErcomputer (Eclipse): The embedded chip-level integrated parallel supercomputer (Eclipse) [65] is a scalable, high performance computing architecture for NoCs. An Eclipse consists of multi-threaded architecture with chaining processors with dedicated instruction memory modules, highly interleaved data memory modules, and a high-capacity sparse mesh interconnection network. Because Eclipse's memory system is cache-less, it has no cache coherency problems. Eclipse's structure is homogeneous, simplifying design and making it easier to integrate into a larger SoC. Eclipse features a completely software-based design methodology to support flexibility and general-purpose operation.

Nostrum: Nostrum [15] targets low overhead in terms of hardware and energy usage in combination with tolerance against network disturbance. In the Nostrum architecture, a service of guaranteed bandwidth and latency has been implemented in addition to the existing service of best-effort packet delivery. The guaranteed performance is offered via *virtual circuits*. Virtual circuits are set up semi-statically, in which the route is decided at design time, but the bandwidth is variable at run-time. To share the bandwidth between a number of disjoint virtual circuits a variation of time division multiplexing has been exploited which is referred to as *Temporary Disjoint Networks* (TDN). Nostrum employs deflective switching to avoid congestion and require less hardware as no routing tables or input/output queues are needed.

The drawbacks of the Nostrum concept are the potential waste of bandwidth in the returning phase of the container in the loop, since the container might travel empty if

the best-effort traffic is one-way. Also, the limited granularity of bandwidth possible to subscribe to, might become a problem.

Æthereal: The Æthereal NoC is a full-fledged on-chip interconnect consisting of generic routers and network interfaces [4, 8, 66–68]. The routers use input queuing, deterministic source-based (destination tag) wormhole routing and link-level flow control. To provide time-related guarantees, such as throughput guarantees (on a finite time scale) or latency bounds, interference of other traffic must be limited and characterized. The Æthereal NoC facilitates software programming by offering a strong end-to-end QoS paradigm that provides high-level services, such as transaction ordering or throughput and latency guarantees. The QoS protocol defines traffic classes for throughput/latency guaranteed (GT) and best-effort (BE) services. While GT flits use a connection-oriented, contention-free time-division-multiplexed circuit-switching based on slot tables and appropriate packet headers, BE flits are scheduled to remaining output ports using conventional wormhole routing, input or output queuing, and round-robin arbitration.

The Æthereal network interface converts the OSI network layer of the routers to transport layer services for the connected IP core. All end-to-end connection properties are implemented by network interfaces, i.e. reordering, transaction completion and flow control. The IP cores negotiate with network interfaces to obtain connections by reserving resources, such as network interface buffers, credit counters and slots in router tables. Æthereal supports narrowcast, multicast or simple connections and shared memory-like transactions, such as read, write, acknowledged write, test and set or flush.

Hermes: The Hermes infrastructure generates wormhole-routed NoCs based on basic network components, such as routers and buffers [69]. Hermes employs different topologies, while adjusting flit and buffer size and routing algorithms. Hermes implements three layers of the OSI reference model: physical wiring interface, an explicit handshake data link protocol for transferring data reliably between routers and a network layer for packet switching. It also supports OCP, ensuring enhanced re-usability of the infrastructure and connectivity to available compliant IP cores. The main component is the Hermes router which aims at a 2-D mesh topology. It contains control logic and a set of up to 5 bidirectional ports, i.e. east, west, north and south connections to 4 neighbor

routers and local connection to an IP core. Hermes can use simple XY routing with input queue buffers to enable practical small area implementation. In addition, dynamic arbitration resolves conflicts when multiple packets arriving simultaneously at the router require the same output port.

Mango: Mango's implementation is based on clockless circuit techniques, and thus inherently supports a modular, GALS-oriented design flow. The Mango router exploits virtual channels to provide connection-oriented guaranteed service (GS), as well as connectionless best-effort (BE) routing. The architecture is highly flexible, in that support for different types of BE routing and GS arbitration can be easily plugged into the router. The services are implemented using separate physical buffers and a smart scheduling scheme called asynchronous latency guarantees [70]. Note that for this scheme, latency guarantees are not inversely dependent on bandwidth guarantees, as is the case in TDM-based scheduling. In addition, the mechanism adopted at routers makes global timing robust, since no timing assumptions are necessary between routers. Mango interfaces the asynchronous network to a clocked OCP-based standard socket through network interfaces designed using primitive routing services of the clockless network.

QNoC: QNoC [9] introduces a design process that satisfies the QoS requirements of an on-chip application at low cost. QNoC adopts a mesh topology. However, the process may find some links not useful and trim them from the NoC. The routing algorithm employed is shortest path XY routing algorithm. And the NoC employs wormhole switching. QNoC defines presence of four different services namely, signaling, real-time, read/write and block-transfer.

The routers have the capacity to accommodate a few flits of different classes. The design process starts by defining and connecting modules in an ideal network. In the next step the design is augmented with inter-module traffic. Further in the design process the assumptions about the links traffic are validated and the modules placed in such a way to minimize the system spatial density. By this stage the place and inter-module communication requirements have been determined. During the next step the modules are mapped to a NoC.

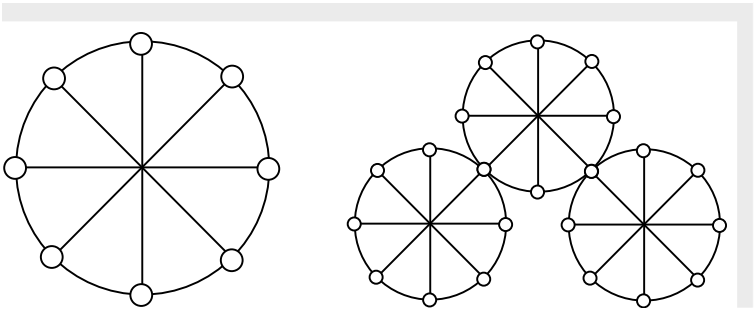
Chain NoC: The Chain NoC [49] is topology-independent, implementing GALS with asynchronous links to interconnect IP modules efficiently. Chain provides a flexible, clock-independent solution, increasing bandwidth, reducing power consumption, and resolving timing closure problems in deep submicron technology. A router implementation for Chain provides differentiated services with soft deadlines by prioritizing virtual channels [12].

×pipes and the NetChip compiler: ×pipes and the NetChip compiler provide an automated NoC design flow based on parametric network building blocks for application specific NoC architectures [71]. The ×pipes library provides switches that support reliable communication for arbitrary link pipeline depths, and an OCP-compliant network interface that connects to/from the IP cores. ×pipes supports regular and heterogeneous architectures with source-based, wormhole routing.

Based on the final SoC architecture floor plan consisting of switches, links, network interfaces and IP blocks, the ×pipes compiler automatically extracts synthesizable SystemC cycle- and signal-accurate executable specifications for all network components. Input to the ×pipes compiler is provided either through a user-specified topology file, or using the SUNMAP tool [72]. This tool automatically maps the IP cores onto the NoC topologies selected from a library, considering different NoC routing strategies, such as dimension ordered and shortest path. By utilizing floor plan information, SUNMAP can minimize area or power dissipation requirements, and maximize performance characteristics.

Octagon: Octagon, proposed by ST Microelectronics [73], is an interesting circuit-switched interconnect based on a regular point-to-point topology designed for targeting the network processor domain. The basic Octagon configuration is shown in Figure 2.5(a), which is an eight node bidirectional ring with cross connections. Thus, each processor is directly connected to two adjacent nodes and the node directly across. This basic topology has small degree (3), diameter of just two hops, and allows for a simple and efficient shortest path routing. The network provides a high concurrency, low latency on-chip communication architecture, able to meet network processing needs. It has significantly higher performance than bus-based on-chip communication, while having less wiring complexity. The ST Microelectronics Octagon has two main drawbacks

that limit flexibility, efficiency and scalability as a prospective NoC architecture: circuit switching based on centralized arbitration, and significant network extendability (8 nodes) which represents high granularity when scaling to a larger network configuration. Figure 2.5(b) demonstrates an strategy to extend the Octagon architecture [73].



(a) (b)

FIGURE 2.5: (a) The Octagon basic topology, (b) Extending the Octagon topology.

2.9 Conclusion

On-Chip networks share significant similarities with system-level interconnects. This section presented an overview of the characteristics of the interconnection networks including topology, routing algorithm, switching technique and buffering strategy. Along with the description of each feature, we showed how they suit the NoC paradigm. The chapter has concluded that regular topologies, deterministic routing algorithms, wormhole switching and input buffering are more promising in the NoC paradigm. The section also presented a brief literature review of a selection of NoC architectures.

Part II

The Quarc Network on Chip

Chapter 3

The Quarc NoC Architecture

The NoC paradigm has witnessed numerous NoC architectures with different characteristics and capabilities. The primary focus of those architectures has been on delivering efficient unicast communication. Putting the effort on implementing efficient unicast communication is a rational decision as unicast is the dominant on-chip traffic. Collective communication operations form a part of overall on-chip traffic in a variety of applications running on NoC-based applications. The current NoC architectures either do not implement hardware support for performing collective communication operations, or the implementation involves significant overhead and is therefore not efficient. In contrast to the existing NoC architectures, the collective communication operations (in particular multicast and broadcast) have been the main motivator behind proposing the Quarc NoC.

The Quarc NoC is introduced as a simple and efficient architecture. The Quarc NoC is similar to the Spidergon STNoC architecture and it preserves all features of the architecture including the wormhole switching, deterministic shortest path routing algorithm and the efficient on-chip layout. The Quarc NoC improves on the Spidergon STNoC by applying modifications to the topology, routers and the network interfaces. The modifications are mainly aimed at *i*) balancing traffic more evenly at network links and *ii*) offering efficient multicast/broadcast operations. The Quarc NoC improves on the Spidergon STNoC by applying the following modifications: *i*) doubling the across links, *ii*) enhancing the one-port router architecture to an all-port router scheme and *iii*) enabling the routers to absorb-and-forward flits simultaneously.

This chapter presents a detailed representation and implementation of the Quarc architecture. The characteristics of the network including topology, routing, switching and buffering strategies are described in details.

3.1 Topology

There is a large body of research on regular topologies for interconnection networks both in parallel computers and the NoC domain. In general, it is highly desirable to capture the traffic requirements of the application using a regular topology. On the other hand, embedded multicore applications require customizable heterogeneous communication infrastructures [30]. The customizability and heterogeneity spans a wide range of configuration settings, including the network topology. Of course, customizing the topology of a network may lead to irregular topologies, which are not desirable.

Simple topologies, such as rings, are cost-effective in terms of cost, but deliver relatively poor performance [74], especially as the number of connected cores increases. On the other hand, higher connectivity topologies, such as 2-D mesh, provide interesting theoretical metrics. However, the nature of the on-chip communication traffic, physical characteristics of the IP cores and physical attributes of the final product, typically, inhibits full exploitation of the offered features. Physical attributes of the IP cores are likely to inhibit the topology to preserve its regular structure on chip. Moreover, physical implementation limitations may also not allow or justify adopting a particular topology to be built on chip.

Being inspired by the Spidergon topology, the Quarc topology, despite subtle differences, shares a significant similarity with the Spidergon topology. At a high level of abstraction, where all implementation details are ignored, the topology of the Spidergon and the Quarc NoCs can be presented by the same graph. Due to this similarity and to present a clear comparison between the two topologies, this section presents the Spidergon topology in details, followed by a description of the Quarc topology.

3.1.1 The Spidergon STNoC Topology

The Spidergon [30] topology was proposed by ST Microelectronics to address the need for a fixed and optimized NoC topology to realize cost effective MPSoC development. The Spidergon STNoC topology is a regular topology similar to a simple bidirectional ring, except that each node has, in addition to links to its clockwise and counter-clockwise neighboring nodes, a direct bidirectional link to its diagonally opposite neighbor. Figure 3.1 shows a 16 nodes Spidergon and a potential on-chip layout.

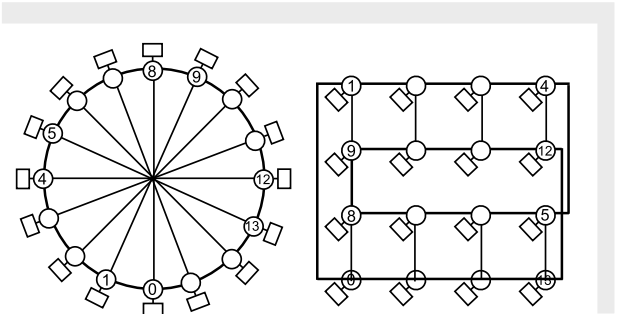


FIGURE 3.1: The Spidergon topology and the on chip layout.

Formally, the Spidergon graph belongs to the general family of undirected circulant graphs. Within this family, the Spidergon network connects an even number of nodes $N = 2n$, $n = 1, 2...$ as a vertex-symmetric $3-$circulant with $k = 2$, $s_1 = 1$ and $s_2 = (l + n)modulo(N)$. Thus, Spidergon consists of a bidirectional ring in both clockwise (right), and anti-clockwise (left) directions; in addition, for each node there is a cross connection, i.e. from node i, $0 \leq i < N$ to node $(i + n)modulo(N)$.

When it comes to implementation, the nodes of the Spidergon STNoC are connected by unidirectional links. The Spidergon topology is a variation of the ring topology and prone to deadlock. The Spidergon and the Quarc NoCs use similar approaches to avoid deadlock. The deadlock avoidance strategies are discussed later when the routing is discussed.

The key characteristics of the Spidergon topology include: good network diameter, low node degree, homogeneous building blocks (the same router to compose the entire network), vertex symmetry, low extendability granularity and simple routing scheme.

Compared to complex topologies, Spidergon offers a small number of links and simple implementation. For current, realistic NoC configurations with up to 60 nodes, the

proposed Spidergon graph has a smaller number of edges and a competitive network diameter with respect to fat-tree or 2-D mesh topologies [74].

In contrast to the existing NoC architectures which are either based on a fixed regular topology, or are topology-independent, the Spidergon topology is introduced as a *pseudo-regular* topology. Using this concept, NoC topology becomes an architectural parameter that can be configured depending on the communication patterns exhibited by the application. Thus, the Spidergon topology fills the gap between regularity and customizability.

Depending on the application traffic, the Spidergon topology can be customized and simplified. This feature allows the Spidergon to support different families of topologies. These topologies are essentially degree 2 or 3 Spidergon sub-graphs that range from rings and simple spanning trees to irregular chordal rings which are built using similar building blocks. Figure 3.2 depicts different topologies supported by the Spidergon topology. Moreover, the Spidergon STNoC allows traffic at injection and ejection links to be aggregated to utilize the links more efficiently.

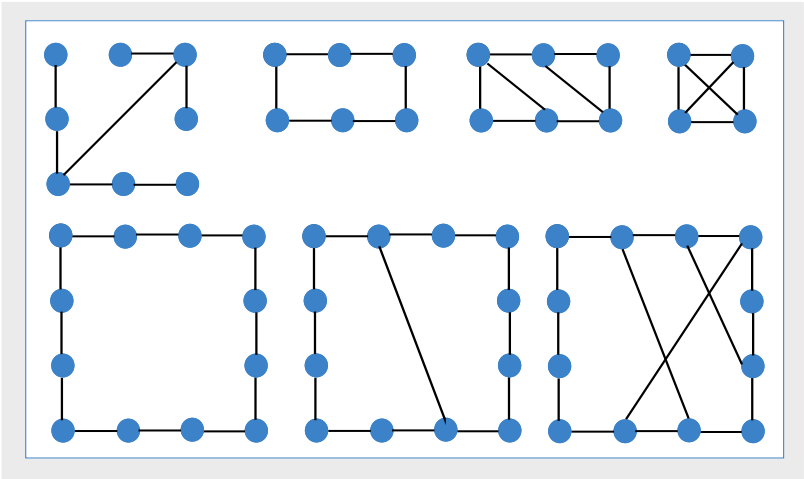


FIGURE 3.2: A number of topologies supported by the Spidergon NoC.

The Spidergon topology allows extending network as a hierarchical structure. Hierarchical network structures increase the performance, since they reduce conflicts by exploiting locality, while ensuring global all-to-all connectivity. Figure 3.3 shows two hierarchical network structures supported by the Spidergon STNoC.

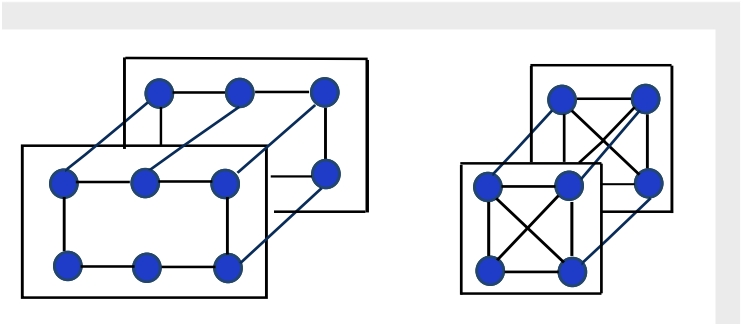


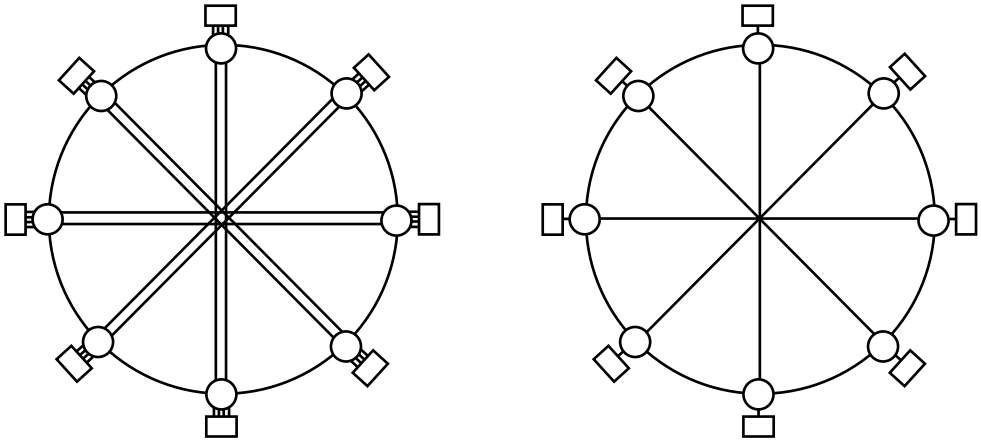
FIGURE 3.3: Hierarchical network structures supported by the Spidergon STNoC.

3.1.2 The Quarc NoC Topology

As mentioned earlier, the Spidergon allows adding or removing links between the nodes on demand. In respect to this feature, the Quarc topology can be regarded as a variation of the Spidergon. Similar to the Spidergon, the topology of the Quarc NoC can be formally represented as a undirected circulant graph connecting $N = 2n$, $n = 1, 2...$ nodes as a vertex-symmetric $3-$circulant with $k = 2$, $s_1 = 1$ and $s_2 = (l + n)mod_N$. Thus, exactly as the Spidergon, the Quarc topology consists of a bidirectional ring in both clockwise (right), and anti-clockwise (left) directions; in addition, for each node there is a cross connection, i.e. from node i, $0 \leq i < N$ to node $(i + n)mod_N$.

In the Spidergon topology, half of the nodes are accessed through the left and right links; the rest of the nodes are accessible through the across links. Therefore, the across links can become a bottleneck. On the implementation side, Quarc is distinguished from Spidergon by using two physical links to separate the access to across-left and across-right nodes. Adding this modification to the topology results in a more even distribution of traffic on links. Moreover, it leads to simpler routing algorithms and, last but not least, the effect of modification combined with the proper network interface implementation is best manifested in performing multicast and broadcast communications. The resulting topology for an 8-node Quarc and Spidergon topologies are compared in Figure 3.4.

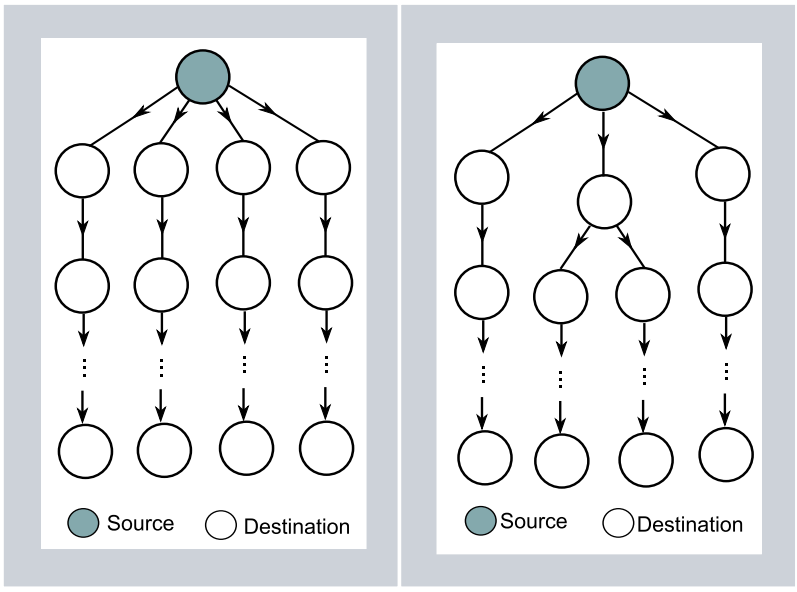
Symmetry is an important feature affecting VLSI design issues and implementation of efficient point-to-point routing algorithm. If an automorphism exists that maps any node α into another node β the graph is vertex-symmetric. A vertex-symmetric graph looks identical from any network node. Based on this definition, the Spidergon topology is vertex-symmetric. Doubling across links does not affect this property. Therefore,



(a) (b)

FIGURE 3.4: Topology of (a) Quarc and (b) Spidergon.

graph representation of the Quarc topology is vertex-symmetric. Figure 3.5 compares the view of each Quarc and Spidergon node to the network, where Across-First routing algorithm [30] is adopted. As will be shown, the difference between the two graphs which is simply a result of doubling across links has a significant impact on routing decision and particularly on the performance of multicast communication, while does not incur extra cost at the routing elements.



(a) (b)

FIGURE 3.5: The graphs representing the view of each node to the (a) Quarc and (b) Spidergon topologies.

3.2 Packet format in the Quarc NoC

The Quarc NoC uses a simple deterministic routing discipline, therefore, the packet format for unicast and collective communication is quite simple. For a Quarc NoC employing flit size of 34 bits various flit types composing a packet are depicted in Table 3.1. Bits [1 : 0] denote the flit types namely, *header*, *body* and *tail*. Bits [7 : 2] and [13 : 8] represent the source and destination addresses, respectively. Bits [30 : 14] in case of multicast represent the *bitString*. Finally, the last 3 bits of header flits represent traffic types which are shown for *unicast*, *multicast* and *broadcast*. Each packet must have the header and tail flits.

Note that due to the scalability issues of the Quarc NoC, it is assumed that the network size may be up to 64 nodes. However, larger networks may employ flits of larger size or use multi-flit headers for specifying multi-addresses for multicast operations.

Bits	[33:31]	[30:14]	[13:8]	[7:2]	[1:0]
Unicast header	unused	unused	destination address	source address	0
Broadcast header	unused	unused	destination address	source address	0
Multicast header	unused	bitstring	destination address	source address	0
Body	Payload				1
Tail	Payload				2

TABLE 3.1: Flit type formats in the Quarc NoC.

3.3 Routing algorithm

The routing algorithm for an on-chip network or in general for an interconnection network is significantly affected by the topology of the network. Numerous routing algorithms have been proposed and adopted by interconnection networks for parallel computers. However, the peculiarities of NoCs inhibit employing most of the contributions in the on-chip domain.

The following sections present unicast, multicast and broadcast routing algorithms in the Spidergon and the Quarc NoCs.

3.3.1 Unicast routing

Spidergon The Spidergon STNoC adopts a deterministic routing algorithms. The routing schemes are simple since they leverage the symmetry and simplicity of the Spidergon topology. Moreover, the relevant implementation of the on-chip routers is extremely efficient as it does not require expensive routing tables. The Spidergon STNoC uses source routing to encode routing information in the packet header at network injection points or network interfaces. Thus, routers can easily decode the path from the header.

The routing in the Spidergon STNoC is programmable. The path from source to destination is determined by a routing function executed at packet injection time. This function can be changed during run-time through software reconfiguration, fully exploiting topological path redundancy. A primary advantage of routing programmability is fault tolerance support which is foreseen to become mandatory in deep-submicron technologies.

The Spidergon STNoC employs oblivious routing [30]. This means that packet routing decisions are carried out using only local information available at each network node. When a router receives the header flit, the routing algorithm compares the network address of the current router to that of the destination router. If the two network addresses match, flits are routed to the local port of the router. Otherwise, an attempt is made to forward the flit towards a clockwise or counter-clockwise direction along the ring, an across link, or in a hierarchical way to another instance of the Spidergon topology family.

To follow the shortest path, the Spidergon STNoC may implement two algorithms. The routing algorithms are distinguished based on when to take the across link. The first algorithm, called Across-First, moves packets along the ring, in the proper direction, to reach the destination nodes. The across links are used only once at the beginning for destinations that are far away. Across-Last is another routing scheme that can be used on the Spidergon STNoC. Instead of jumping through the across link as the first hop and moving along the ring to reach the final destination, packets can first move along the clockwise or counter-clockwise directions and finally take the across link to the destination node.

Quarc Similar to the Spidergon NoC, the Quarc NoC can implement a variety of unicast routing algorithms. Depending on the position of the destination, a packet may require taking an across link in the path. Employing the Across-First routing algorithm in the the Quarc NoC can realize the routing algorithm using the minimum resources. In this approach, if the shortest path requires taking an across link, this will be the across link of the router corresponding to the source node.

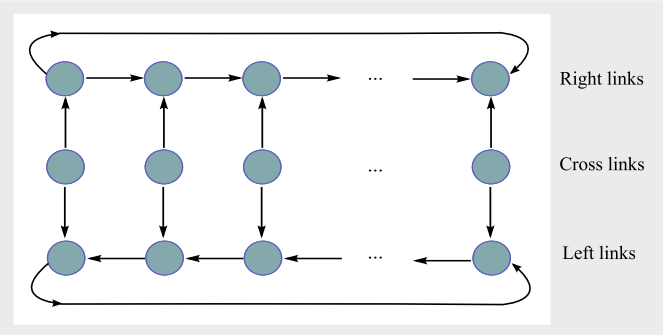
Using the Across-First routing algorithm, unicast in the Quarc NoC is quite simple: packets are either destined for the local port or forwarded to a single possible destination. Consequently, the proposed NoC switch requires no routing logic. The route is completely determined by the port in which the packet is injected at the source node. Of course, the network interface of the source IP core must make this decision and therefore calculate the quadrant as outlined above. However, calculating the quadrant in the network interface incurs a negligible cost.

3.3.2 Deadlock avoidance

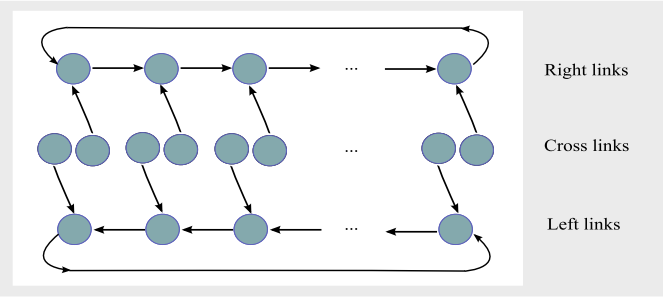
Given that the Across-First routing is adopted, the cyclic dependency graphs (CDGs) of the Across-First routing scheme in the Spidergon and the Quarc NoCs are illustrated in Figure 3.6. Due to cycles in corresponding CDGs, the Across-First routing in the Spidergon and the Quarc NoCs are not deadlock free. Both topologies are regarded as variations of chordal ring families. Thus, the cycles in both CDGs arise from dependencies in bidirectional rings. Therefore, handling deadlock in the Spidergon and the Quarc NoCs adopting Across-First routing is mandatory. This may be addressed by breaking the cycles created from the ring.

Since the topology of the Quarc NoC is considered as a relative of the Spidergon, and doubling the across links does not affect the channel dependency, the rest of this section focuses on deadlock handling in the Spidergon NoC. Any solution can be readily applied to the Quarc NoC.

As mentioned above, deadlock in Spidergon arises from bidirectional rings in the topology. Hence, low-level deadlock in the Spidergon STNoC can be avoided analogous to that in the ring topology. Adopting virtual channel is an approach to remove cyclic dependencies in rings [75]. Consequently, the Across-First routing method presented in



(a)



(b)

FIGURE 3.6: CDG of Across-First routing in the (a) Spidergon and (b) Quarc architectures.

Section 3.3.1 can be extended to the following virtual channel allocation algorithm to avoid deadlock.

In a naive approach each physical link constituting the ring is shared by two virtual channels, vc_0 and vc_1, where virtual channel vc_0 is used, when required, as an escape channel. Also, we assume that node 0 is the dateline. On entering the network, packets use vc_1 and continue using vc_1 before meeting the dateline node. Upon crossing the dateline node, the packet uses vc_0 for the rest of its journey to destination. Adopting this approach guarantees deadlock freedom. However, it performs poorly in terms of virtual channel utilization. That is because only $\frac{N}{4}$ of vc_0's in the network are ever used. $\frac{N}{4}$ is the maximum distance a packet may travel on the Spidergon ring after the dateline node.

In the above naive deadlock avoidance scheme many buffers associated to vc_0 are unused. An optimized utilization of virtual channels can improve the performance by reducing contention. For example, if a packet route does not cross the dateline, any virtual channel can be used, i.e. virtual channel assignment for these packets is unconstrained.

A simple origin-based virtual channel allocation algorithm is proposed to appropriately combine deadlock avoidance and load balance on the Spidergon STNoC [30]. This scheme is based on partitioning the Spidergon ring into two contiguous halves: nodes $0, N-1, N-2, ...N/2+1$ are called *type I*, while nodes $N/2, N/2-1, ..., 1$ are *type II* nodes. As mentioned earlier, using shortest-path deterministic routing, the maximum distance of an Across-First path on the ring is $N/4$. Thus, packets will change their partitions at most once. With origin-based virtual channel allocation, packets that do not cross the two dateline nodes ($N/2$ and 0) are assigned to any of the two virtual channels, i.e. either vc_0 or vc_1. However, packets crossing the dateline nodes, i.e. packets originating at type I nodes crossing node $N/2$, or packets originating at type II nodes crossing node 0, are initially routed on vc_0, and then upon crossing their dateline they are shifted to vc_1.

Assuming random traffic, the virtual channel assignment is fair on $2(N/4-1)$ nodes with packets equally assigned on both virtual channels. It is in favor of vc_0 by $2:1$ on $N/4$ nodes, in favor of vc_1 by $2:1$ on $N/4$ nodes, and completely unfair on only two nodes (those required for deadlock-free operation).

Designing routing schemes that are free of dependency cycles by construction is another way to avoid deadlock. This is normally achieved by restricting the possible paths in the topology. However, such an approach is typically considered as application specific and hard to adopt in a wide range of applications.

3.3.3 Broadcast routing

Spidergon Given that the Spidergon NoC does not provide hardware implementation of the functionality to perform collective communication operations, broadcast in the architecture can be handled most efficiently by unicast with a "unicast tree" algorithm depicted in Figure 3.7. The initiating node, say node 0, sends a packet to node $N/2$; nodes 0 and $N/2$ send a packet to $N/4$ and $N/2+N/4$; all 4 nodes send a packet to nodes $N/8$, $N/4+N/8$, $N/2+N/8$, $N/2+N/4+N/8$ and so on. Because this is a multi-stage process ($log_2 N$ stages) the broadcast packet needs a decrementing count field to identify the stage of the broadcast process. When a NoC router receives a broadcast packet, it must take the following decisions:

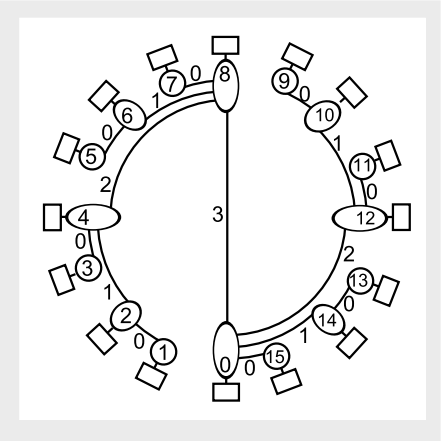


FIGURE 3.7: Broadcast in a 16 nodes Spidergon NoC.

1. Is the current node a destination node or a forwarding node? The rule for this decision is: if the distance between the source address and the node address is smaller than the value of the count field, the packet must be forwarded (on the rim). Otherwise, the packet is received by the local node.

2. Is further broadcast required? The rule for this decision is: if the count field is 0, no further broadcast is required.

3. If further broadcast is required, how many packets need to be sent? The number of packets to be sent is given by the count field of the ingress packet. Essentially, the switch decrements the count field and forwards the packet along the rim. This means that the router or network interface must buffer the packet for the duration of the broadcast and decrement the count field in the buffered packet before each transmission, until the count is 0.

The problem with this scheme (and in general with broadcast-by-unicast) is that the router or network interface requires buffer space to store a whole packet. If the required space is not available, broadcast has to be performed using the simplest and least efficient approach of sending unicast messages to each individual destination separately.

Quarc Broadcast, the key motivation behind the Quarc topology, is elegant and efficient: The Quarc NoC adopts a BRCP (Base Routing Conformed Path) [76] approach to perform multicast/broadcast communications. BRCP is a type of path-based routing in which the collective communication operations follow the same route as unicasts do. Since the base routing algorithm in the Quarc NoC is deadlock-free, adopting the BRCP

technique ensures that the broadcast operation, regardless of the number of concurrent broadcast operations, is also deadlock-free.

To perform a broadcast communication the network interface of the initiating node has to broadcast the packet on each port of the all-port router. The network interface tags the header flit of each of four packets destined to serve each branch as broadcast to distinguish it from other types of traffic. The network interface also sets the destination address of each packet as the address of the last node that the flits stream may traverse according to the base routing. Each receiving node simply checks if the destination address at the header flit matches its local address. If so, the packet is received by the local node. Otherwise, if the header flit of the packet is tagged as broadcast, the flits of the packet at the same time are received by the local node and forwarded along the rim. This is simply achieved by setting a flag on the ingress multiplexer which causes it to clone the flits. Using this algorithm, the Quarc NoC can deliver a broadcast message in only one step.

Broadcast in a Quarc NoC of 16 nodes is depicted in Figure 3.8. Assuming that Node 0 initiates a broadcast, it tags the header flits of each stream as broadcast and sets the destination address of packets as 4, 5, 11 and 12 which are the address of the last node visited on left, across-left, across-right and right rims respectively. The intermediate nodes receive and forward the broadcast flit streams, while the destination node absorbs the stream.

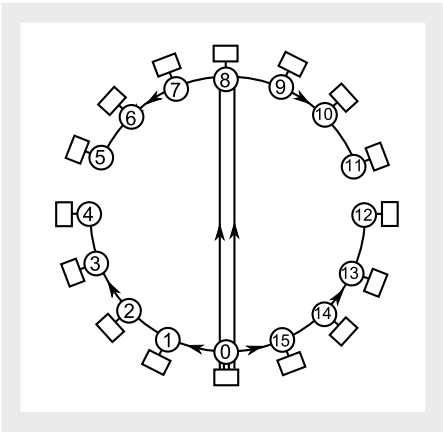


FIGURE 3.8: Broadcast in a Quarc of 16 nodes.

3.3.4 Multicast routing

Similar to broadcast, in the multicast operation, the last node to be visited must be specified as the destination address on the rim in the header flit. For broadcast all nodes in the path from source to destination are the receiving nodes. In the case of multicast the target addresses are specified in the *bitstring* field. Each bit in the *bitstring* represents a node; its hop-distance from the source node corresponds to position of the bit in the *bitstring*. The status of each bit indicates whether the visited node is a target of multicast or not. Consequently, broadcast is simply a special case of multicast where every node is a target.

3.4 Switching technique

In the NoC domain the resources are scarce and the applications have typically some performance requirements. A major challenge in the domain is therefore, to fulfill the applications' performance demands using the limited available resources. Resource constraints lead to employing those algorithms and techniques that will realize the required functionality in a more cost effective fashion. Buffers account for a significant share of the overall cost and the power consumption in NoCs. For instance, by increasing the buffer size at each input channel from 2 to 3 words, the router area of a 4×4 NoC increases by 30% or more [47]. Thus, the overall use of buffering resources has to be minimized to reduce the implementation overhead in NoCs.

The switching technique employed by a network has a direct relation to the size of the buffers at routers. Store-and-forward and cut-through switching techniques require enough buffers to store a whole packet at each intermediate router. This does not seem to be affordable by most NoC architectures. Moreover, storing the packet at each intermediate router means that the message latency is directly related to the distance between source and destination. The need for efficient communication at low cost leads to adopting wormhole-switching as the dominant switching technique in the NoC realm [3, 47].

The Quarc NoC targets an area-efficient, low power, high performance implementation. Therefore, it adopts wormhole-switching to leverage the low buffer demands and distance insensitivity feature of the the switching technique and to minimize power consumption.

3.5 Buffering strategy

The Quarc NoC adopts input queuing at routers and employs virtual channels to improve the network bandwidth and latency. To avoid deadlock, the basic Quarc architecture requires sharing each physical link by two virtual channels. The optimal number of virtual channels in the Quarc NoC depends on the application and the size of the network.

3.6 The router architecture

This section presents the router architecture of the Quarc NoC with multicast and broadcast communication support. Figure 3.9 presents a minimal architecture for use with deterministic routing, i.e. the hardware is tailored to the data-paths allowed by the deterministic routing discipline.

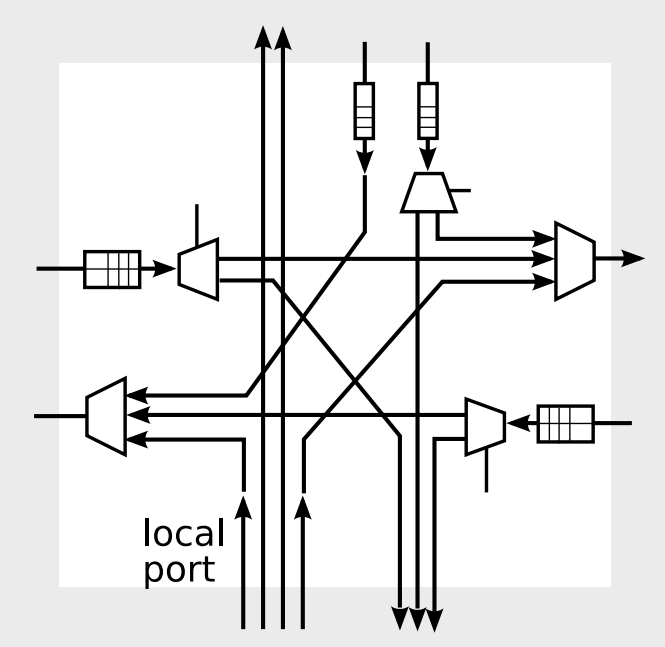


FIGURE 3.9: Minimal router architecture in the Quarc NoC.

The Quarc router implements the OSI reference model services at network, data-link and physical links. The Quarc router deals with switching, topology and routing schemes.

QoS provisions can also be implemented by the router. However, the basic Quarc router does not support QoS.

Figure 3.10 depicts a top level functional block diagram of the Quarc router for variable virtual channels. As shown in Figure 3.10, the Quarc router consists of three fundamental modules, namely, input port controller (IPC), switch, and output port controller (OPC). Any flit entering the Quarc router passes through four stages, namely, input buffering, routing, virtual channel allocation and switch allocation. We have developed an FPGA implementation of the Quarc router using Verilog targeting the *Xilinx Virtex- II Pro (XCV2P30) FPGA.*

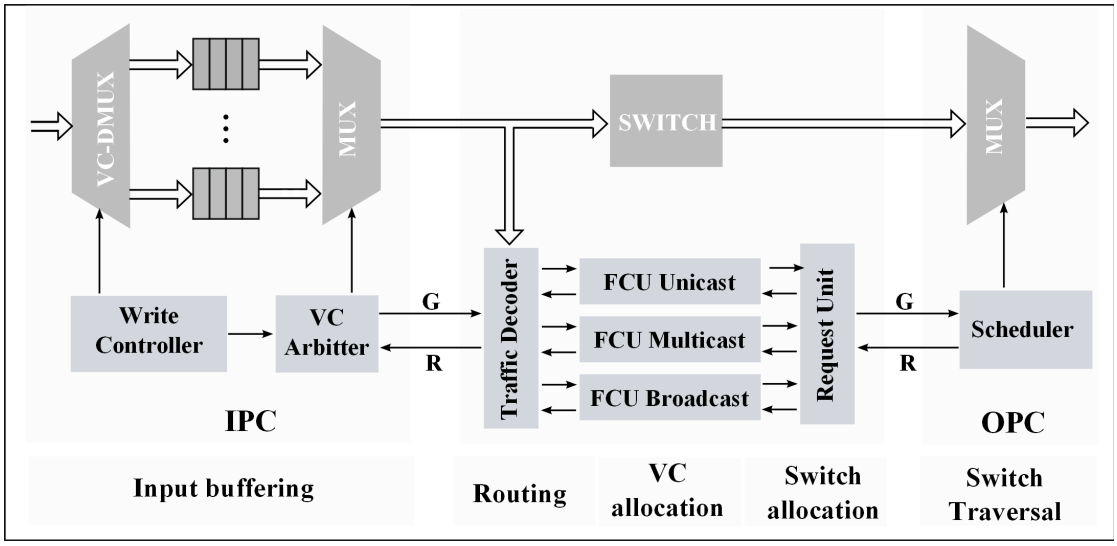


FIGURE 3.10: Functional block diagram of the Quarc router.

3.6.1 Input port controller

The IPC performs two main operations on incoming flits: de-multiplexing and buffering. A write-controller acts as the controller of the IPC. Its main job is to read the input handshake signals and enable the buffer to store the flits at appropriate time. The IPC also sends back the buffers status to the source router. The buffers in the IPC are parametrized in width and depth.

3.6.2 The switch module

The switch module consists of three main sub modules, namely, crossbar, VC arbiter and the flow control unit (FCU). The crossbar passes the flits to its destination, while

the FCU acts as its main controller. The crossbar in the Quarc architecture is very simple as a flit may be either destined to a local node or to be forwarded on the same direction on the rim. This makes the Quarc router very light-weight compared to the 2-D mesh topology, where every input can have four possible destinations. The VC arbiter arbitrates between flits received in more than one virtual channel and sends a request (R) to the FCU for routing and switch allocation. The task of the FCU is in three parts, *i*) to classify the traffic type, *ii*) to request appropriate OPC, and *iii*) to enable the crossbar to pass the flits.

The traffic decoder shown in Figure 3.10 performs the traffic classification task. There are three sub-units for three different types of traffic, namely, FCU unicast, FCU broadcast and FCU multicast. The advantage of the having three separate sub units is to have two units switched off, while only one is functioning. As a result, dynamic power consumption will be less in the FCU unit. The request unit is completely combinatorial in nature and sends appropriate request (R) signals to the OPC.

3.6.3 Output port controller

The OPC consists of two sub-modules, namely, scheduler and the multiplexer. By using suitable arbitration mechanism, the scheduler allows only one of the requests received and sends back a grant (G) signal to corresponding FCU. The FCU forwards the grant signal to the corresponding IPC and sets the crossbar to transmit the flit to the proper destination. The scheduler also controls the OPC's multiplexer by enabling the appropriate flit to flow to the next router. The scheduler also generates the appropriate handshake signals for synchronization and sends the flits to the next router.

It is important to note that the flit at the buffer of a local port is immediately consumed by the network interface or the IP core. Therefore, multicast or broadcast traffic may block only as the result of resource unavailability at the next router on the rim.

3.7 Conclusion

The chapter introduced the Quarc NoC as a simple and efficient architecture for on-chip communication. The topology, routing algorithm, switching technique and buffering

strategy in the architecture are described. A design and FPGA implementation of the Quarc router is also demonstrated in the chapter. The topological attributes of the the Quarc allows employing a simple packet format to send unicast, broadcast and multicast communication. However, this should be combined with an appropriate network interface. The Quarc network interface is covered in the next chapter.

Chapter 4

The Quarc Network Interface

Increasing design complexities and time-to-market pressures have made modular SoC development inevitable. Obviously, the most fundamental requirement for a modular SoC development is the ability to interconnect the cores with the minimum effort. Point-to-point communication standards such as OCP (Open Core Protocol) [77], VCI (Virtual Component Interface) [78] and AMBA AXI [79] were introduced to address this issue in bus-based SoC development. The communication interfaces have been successfully adopted to offer communication between an IP core and the bus, as well as between two IP cores. The existing communication protocols adopted in bus-based development can be used in the NoC realm to connect the IP cores to a NoC. However, the hop-by-hop and (possibly) packet-based nature of interconnection in NoC requires a specifically customised interface between the core and the interconnection network. In the NoC paradigm such an interface is referred to as network interface (NI).

The NI functions as a glue between computation and communication by implementing the interfaces to both the IP core and interconnection network. The IP core interface implements a standard point-to-point protocol allowing core reuse across several platforms. The most widely used core interface protocols include OCP [77], VCI [78], AMBA AXI [79]. These interfaces assume the attributes of a *socket*, which captures all signaling between the core and the system. This can offer a transaction-based model [68] of communication which is backward compatible with bus-based SoCs. Message passing and shared memory abstraction are two transaction-based programming models employed in

the NoC domain. Shared memory is easier to implement, while message passing is more scalable.

The NI core interface can be viewed as an implementation of the session layer in OSI reference model. Traditionally, the session layer represents the user's interface to the network. It determines when the transaction session is opened, how long it will be used and when to close it. Moreover, it controls the transmission of data during the session, supports security and name lookup, enabling computers to locate each other. Flow control strategies and QoS negotiations can be also implemented at this layer.

The communication services made available at the session layer must be implemented by the transport layer, in order to make the communication behaviour fit to the interconnect. The transport layer provides reliable, sequenced, and QoS-oriented data transfer. This layer provides the basic end-to-end connection.

The transport layer provides transparent transfer of data between end nodes using the services of the network layer. These services, together with those offered by the link and physical layer are implemented in the network interface part of the NI. Data packetization and routing related functions are considered as essential tasks performed by the network layer; offering a reliable links is considered as a service of the data-link layer.

Network interface design has been extensively researched for parallel computers [80–82], and computer networks [83, 84]. The designs of the interface for such networks are to optimise for performance (high throughput, low latency), and often consist of a dedicated processor, and large amount of buffering, hence prohibitively costly for the NoC paradigm. On-chip network interfaces must provide a low area overhead, because the size of the IP modules attached to NoC is relatively small.

4.1 Network interface services

To successfully fulfil its functionality as a an interface between computation and communication, the NI is expected to offer a variety of services to both sides. Scherrer et al. in [85] presented a classification of services offered by NIs. The NI services span from session and transport level services down to low-level ones such as packetization and

clock-domain crossing. However, implementation of a service depends on the application requirements and is a platform-instance-dependent decision.

4.1.1 Adaptation services

These are the basic wrapping services. Their role is to adapt the communication protocol of the component to the communication protocol of the network. Of course, the challenge here is to minimise performance loss in term of latency at minimal area/power cost.

Packetization To facilitate the communication with a packet-switched interconnection network, the NI should perform packetization on incoming messages. The packetization must be compliant with the interconnection network communication protocol and can be dynamic or static. This service selects the size of the packets based on the characteristics of the messages and/or by trading-off between performance and energy. For example, burst messages could be split into smaller packets to better fit the network characteristics. Indeed the average packet size of a packet switched network affects the performance and energy. Increasing the packet size will increase the energy dissipation in network, it will decrease the energy consumption on cache and memory. Because larger packet sizes will decrease the cache-miss rate, both cache energy consumption and memory energy consumption will be reduced [86]. In QoS-aware NoCs, the protocol adapter module equips the packets with relevant information.

The NI should also perform de-packetization of the incoming packets from NoC and send the data to the IP cores according to the communication protocol between them.

In [87] a comparison of three approaches in implementing packetization and depacketisation services are provided. The results show that the hardware implementation can be realised with much less resources and offer significantly better performance in terms of latency compared to software implementation. Implementing the functionality in hardware or software requires the IP core to offer the relevant programming or reconfigurability services. For the cores that are neither programmable nor reconfigurable, an option for interfacing with the networking logic of the tile is to utilise a wrapper. In [87] a wrapper is implemented which has the responsibility of packetizing and depacketizing the cores requests and responses. Due to implementing the wrapper in different technology than the software/hardware approach, the cost comparison was not

available. However, the analysis revealed that wrappers offer a good trade-off between cost and performance (latency). Employing wrappers is considered as the dominant core-interconnect interfacing strategy.

Clock adaptation Synchronization of the IP cores is becoming hard to achieve in future large SoCs. That is why future SoCs will be *locally synchronous and globally asynchronous*. A NoC is composed of rather simple elements, and thus they could potentially run at higher frequencies in order to decrease the latency seen by the components. For example, the SOCBus micro-network designers expect it to run at 1.2 GHz [88], which is higher than the frequency of average cores. Also, there are a number of NoC architectures which are clockless [70]. Therefore, an important role of the NI is to adapt between the IP core and network operation frequency.

Bandwidth and latency guarantees Most NoC architectures offer guaranteed or best-effort services QoS. In such networks the NI has to utilize the QoS services of NoC. This means for example to build and send the virtual circuit set-up or tear-down packets, or to allocate multiple buffering resources and design complex packet schedulers in the NI to handle traffic. Static reservation does not affect the NI.

Core interfacing The ability to develop a SoC in a plug-and-play and modular fashion is essential to reduce the design time and time-to-market. A key factor in success of such a development process is separating computation from communication, which may be best realized through a well-defined layering of different functionality of components of a system [89]. The seamless integration of components in such a development environment requires standard communication protocols between different layers. Industry has witnessed several successful protocols, such as OCP-IP [77] and AXI [79] for point-to-point communication between SoC components. Since the components (IP, bus and NoC) may expose different protocols, an adaptation between these protocols is required. The network interface can be considered as an appropriate place to implement such functionality.

4.1.2 Network services

In a packet-switched NoC, the task of the interconnection network is to deliver the packets from source to destination. The reliable communication between source and destination, however, requires services which are mainly expected to be delivered by the transport layer in computer networks. Their implementation strongly depends on the features of the underlying network.

Transactions ordering In packet switched networks employing adaptive routing algorithms, packets can potentially arrive unordered. In such circumstances, the NI must reorder the transactions before forwarding them to the IP core. This typically requires a large amount of buffering resources and that is the main reason that deterministic routing is favored in the NoC paradigm. SPIN [5] uses adaptive routing in a fat tree topology, therefore re-sequencing buffers is mandatory at the receiver's network interface.

Reliable transactions According to some studies [90] the assumption that the on-chip communication medium is an error-free medium is expected to be no longer true in future deep sub-microns technologies. Thus, depending on the reliability of the medium, the NI should implement an appropriate mechanism to deal with it (acknowledgment, error-correcting codes).

Collective communication operation Most communication in a typical SoC is between two IP cores. However, there are situations in which more than two IP cores are involved in a communication. Multicast and broadcast are two widely used samples of such operations. The NI can be regarded as an appropriate place to implement such functionality.

Flow control When a given buffering resource in the network is full, there needs to be a mechanism to stall the packet propagation and to propagate the stalling condition upstream. The flow control mechanism is in charge of regulating the flow of packets through the network, and of dealing with localized congestion. ACK/NACK and credit-based are the most widely used strategies in the NoC schemes. Æthereal [48], Nostrum

[15] and SPIN [5] are examples of the NoC architectures employing credit-based end-to-end flow control.

4.1.3 Functional services

In addition to a reliable end-to-end communication, the NI can implement a number of widely-used services such as cache coherence and security to the SoC. Of course, such functions may also be implemented by the IP module or software. However, implementing them at the NI enhances the design reuse.

Cache coherence Cache coherency can be achieved on a bus at a very low cost by means of snoop devices, leveraging the shared nature of the communication medium. Cache coherence on a network is no longer an easy task because snooping is not possible. Cache coherency is a traditional subject in the parallel processing. Due to the similarities of the domain with NoC, the community can rely on related experiences in parallel processing domain to implement efficient cache coherence services. The NI can be regarded as one of the best candidates to host the functionality.

Low power Power customization is an important issue in SoCs. The power issue becomes even more critical in presence of network-centric systems. In fact, early NoC prototypes show a significant contribution of the NoC to the system power dissipation. Beyond electrical and gate level low-power design techniques, higher level techniques are likely to achieve larger savings. To this end much work can be done at design time, but also at run time. Switching off some components and waking them up is for example a good technique to save power, and it is believed that such techniques could be implemented in the wrapper [85].

Security Currently security is not regarded as a crucial issue when it comes to communication between the IP cores on a chip. However, the NI can implement the desired authentication services, should future systems require the functionality.

4.2 A review of network interface implementations in the NoC domain

The NIs are expected to implement several essential services such as virtual channel arbitration, frequency adaptation and routing strategies. There are other tasks that maybe implemented by the IP core directly (e.g. packetization) or might need specific customization. As a consequence, several NoC architectures [30, 48] have assigned the functionality of the NI into *kernel* and *shell* modules, where the kernel module provides essential functionality, and the shell modules offer instance-specific services. The complexity of an NI depends on the services of the network and on the functionality the attached IP cores demand for. This section presents the implementation of the network interfaces in a number of NoC architectures.

Æthereal Æthereal offers a shared-memory abstraction to the connected modules [68]. Communication is performed using a transaction-based protocol, where master modules issue request messages that are executed by the slave modules, which may respond with a response message. The Æthereal NoC offers its services by means of connections. Connections allow differentiated services and guarantees offered to the attached cores. The connection can be peer-to-peer, multicast or narrowcast.The Æthereal network interface provides services at the transport layer in the OSI reference model [91]. The Æthereal network interface provides a modular NI, which can be configured at design time. This is, the number of ports and their type (i.e., configuration port, master port, or slave port), the number of connections at each port, memory allocated for the queues, the level of services per port, and the interface to the IP modules are all configurable at design (instantiation) time using an XML description. The NI allows flexible NoC configuration at run time. Each connection can be configured individually, requiring configurable NoC components (i.e., router and NI). However, instead of using a separate control interconnect to program them, the NoC architecture is used to program itself. This is performed through configuration ports using DTL-MMIO (memory-mapped IO) transactions [92]. The NoC architecture can be configured in a distributed fashion (i.e., via multiple configuration ports), or centralized (i.e., via a single port).

The communication services of the Æthereal NoC are defined to meet the following goals: *i*) decouple computation (IP modules) from communication (NoC), *ii*) provide backward

compatibility to existing bus protocols, *iii*) provide support for real-time communication, and *iv*) have a low-cost implementation.

The Æthereal NI consists of two parts, namely, the NI kernel and the NI shells. The NI kernel implements the channels, packetizes messages, and schedules them to the routers, implementing the end-to-end flow control and clock-domain crossing. The NI shells implement the connections, transaction ordering and other high level issues.

In [68] a hardware implementation of the entire NI introduces a latency overhead of between 4 and 10 cycles, pipelined to maximize throughput.

×pipes The ×pipes NI offers an OCP-compliant (Version 2.0), low overhead and high performance network interface [93]. The NI is parameterizable in both the width of the OCP fields and of ×pipes flits. This feature provides a wide range of NI deployment flexibility. The communication in ×pipes is packet-switched, with source routing and wormhole-switching.

The ×pipes NI functionality include the synchronization between OCP and ×pipes timings, the packetizing of OCP transactions into ×pipes flits and vice versa, the computation of routing information, and the buffering of flits to improve performance. In addition to the core OCP signals, the ×pipes supports the ability to perform both non-posted or posted writes (i.e. writes with or without response) and various types of burst transactions, including reads with single request and multiple responses. This allows for thorough exploration of bandwidth/latency trade-offs in the design of a system. The ×pipes NI has a low area but it supports only a single outstanding read transaction.

In a ×pipes based NoC, a master-slave device will need two NIs, an initiator and a target for operation. Each NI is additionally split in two loosely-coupled sub-modules; one for the request and one for the response channel.

MANGO In [11] an OCP compliant NI architecture for the MANGO NoC is presented. The NI enables modular, GALS type SoC design by providing synchronous, memory-mapped interfaces, based on the clockless message-passing services of the network. The flexible architecture, which mixes clocked and clockless circuits, can easily be configured for different sockets and/or networks. Unlike Æthereal and ×pipes which are

purely clocked designs, MANGO leverages the advantages of both clocked and clockless design styles by synchronizing clocked interface sockets with asynchronous NoC, thereby more efficiently address issues related to global synchronization in large-scale SoC design and at the same time enjoys the benefits of asynchronous implementation, including zero idle power and low forward latency.

The MANGO NI adopts a transaction-based communication model, which assumes communicating cores of two different types: masters and slaves. The NI provides a number of input and output network ports, each corresponding to a time-guaranteed connection oriented services or to a best-effort connection-less service. Each port may be used by several threads. The NI also implements services for threads.

4.3 The Quarc network interface

The Quarc adopts a modular approach in developing the NI to enable easy development and upgrade of the modules and to enhance the re-usability of the IP cores. Similar to Æthereal [68] and the Spidergon STNoC [30], the Quarc NI splits the services into two categories, namely, *essential* services and *instance-specific* ones. The essential services such as routing, virtual channel arbitration and flow control are provided by the NI kernel, while the instance-specific services such as packetization, guaranteed services, multicast, security, and clock-domain-crossing (CDC) are to be implemented by the NI shells.

A modular Quarc NI with instant-specific shells allows connecting IPs with any proprietary protocol to a Quarc NoC. Depending on the communication protocol between the IP and the NI, the message structure may have different formats. However, the message structure is irrelevant to the NI, as it just sees the messages as pieces of data that must be molded into the Quarc compliant flits format. Moreover, a SoC designer may choose to implement any shell module with specific functionality inside the NI. A functional block diagram of the Quarc NI is shown in Figure 4.1. The following sections describe the detailed functionality of the NI kernel and shell modules as the constituting components of the Quarc NI. A message entering the NI may take a number of NI shells, nevertheless the format of the output at the stage before the NI kernel must be Quarc NoC compliant. Since the number of shells feeding the NI kernel is instance-specific, a

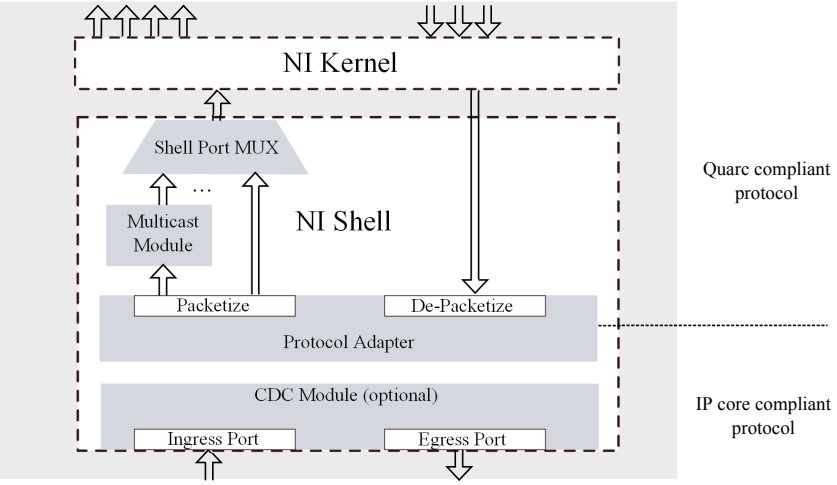


FIGURE 4.1: Functional block diagram of the Quarc NI.

shell multiplexer is employed to manage flit transfer from a number of shells to the NI kernel.

We have implemented the Quarc NI kernel, protocol adapter and multicast modules in Verilog targeting the *Xilinx Virtex- II Pro (XCV2P30) FPGA*.

4.3.1 The Quarc NI kernel

The NI kernel offers the essential services of the Quarc NI. The NI kernel implements the basic functionality to send and receive the Quarc compliant flits. It offers routing, virtual channel arbitration and flow control services. The Quarc NI kernel communicates with the NI shells via ports. The communication protocol between kernel and shells can be customized as the kernel interfaces are not exposed beyond the NI boundaries. The architecture of the Quarc NI kernel is shown in Figure 4.2.

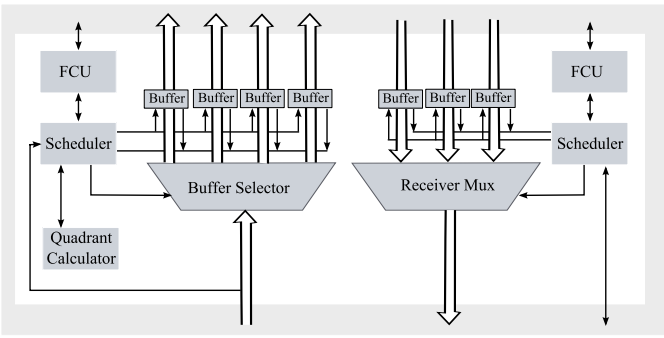


FIGURE 4.2: Functional block diagram of the Quarc NI kernel.

Upon receiving a header flit from any of the shell modules, the scheduler inside the NI kernel informs the quadrant calculator to compute the appropriate quadrant to transmit the packet. The scheduler allocates a virtual channel in the computed quadrant to the packet and updates its internal status. The information stored in the scheduler can be used to forward the remaining flits of the packet to the proper virtual channel without the need to recompute the quadrant.

The decision on a flit's transfer to the network is also made by the scheduler. The decision is based on the information provided by the FCU (Flow Control Unit) and also the scheduler service policy. The FCU manages the transfer of flits between NI and routers by using an on/off flow control mechanism.

The functionality of the NI on receipt of a flit from the router is as follows. Similar to the transmission to the network, the flow of incoming flits from the routers is controlled by the FCU. After being granted the permission, the flit is transferred from the router to the NI and is stored at an appropriate buffer inside the NI kernel. At this stage the scheduler signals sending the flit to the output port of the NI kernel via receiver multiplexer which then can be processed by either any shell modules or the IP core.

4.3.2 The Quarc NI shells

The NI kernel described in the previous section offers peer-to-peer connections. These type of connections are useful in systems involving chains of modules communicating peer-to-peer with one another. In the Quarc NI, more complicated communication such as multicast, and the instance-specific services can be plugged into the NI as the shell modules. The rest of this section present three widely used shell modules.

4.3.2.1 Protocol adapter

For the IP cores which are not customizable, the protocol adapter module serves as a wrapper for packetizing and de-packetizing the messages. The packetization unit converts the received transactions to the Quarc NoC compliant packets. The protocol adapter in the Quarc architecture may be a master or a slave depending on the implementation and actual memory-map. A master protocol adapter converts transactions from IP compliant protocol domain to packet format in the Quarc NoC. While, a slave

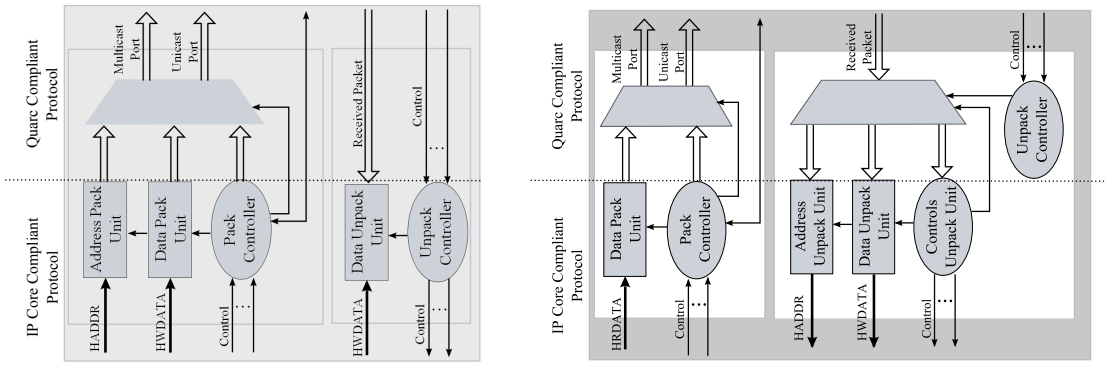


FIGURE 4.3: Functional block diagram of (*a*) a master and (*b*) a slave protocol adapter module.

protocol adapter converts packets from the NoC domain to transactions in IP compliant domain. The diagrams in Figure 4.3 demonstrate the functional block diagram of a master and a slave protocol adapter where the IP core lies in the *AMBA 3 AHB-Lite* domain. The protocol adapter can be replaced with *AMBA 3 AXI*, *OCP* or any other protocol without modifying the NI kernel. Algorithm 1 shows the steps involved in the packetization unit at the master protocol adapter.

Algorithm 1 Packetization at the master protocol adapter

Begin

1. Translation of the addresses from the received address and setting up the packet header with appropriate source and destination addresses.
2. Preparation of body or payload of the packets from the received control signals and forwarding to the multiplexer.
3. Preparation of body or payload of the packet from the received data bits and forwarding to the multiplexer.
4. Preparation of packet tail for error checking (currently not supported) and forwarding the packet to the multiplexer.

End

4.3.2.2 Multicast shell module

Multicast is an important operation in the Quarc NoC. Figure 4.4 depicts the functional block diagram of a multicast shell module. The top 4 buffers are dedicated to store the destinations of the multicast at each quadrant. The buffers corresponding to each quadrant include the address of the last destination to be visited and a bit-string which indicates the potential destinations of the multicast message at each quadrant. The

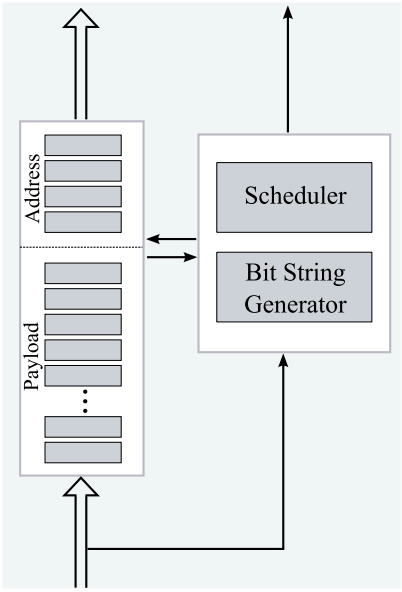


FIGURE 4.4: A schematic of the multicast module.

multicast and packet format in the Quarc NoC are described in Section 3.3 and Section 3.2, respectively. The remaining buffers store the payload.

The bit-string generator creates the bit-string associated to the destination nodes at each quadrant. The algorithm presented in Algorithm 2 is implemented by bit-string generator and is executed on arrival of each multicast destination. The bit-string is used in the header flit of the multicast messages to indicate multicast destinations at a quadrant. The scheduler in the multicast shell stores the information regarding the state of flow at each quadrant. The scheduler holds the address of the next data at the buffer to be transferred to the kernel. The relevant information regarding to each quadrant is updated on transfer of each flit from multicast module to the NI kernel.

Algorithm 2 Bit-string generation.

// The algorithm is performed by bit-string generator on a multicast destination at quadrant q.
// $mcastAdrs$: the multicast address
// $srcAdrs$: address of the source node
// $dstAdrs[q]$: indicates the last destination at quadrant q
Begin
    If $(|mcastAdrs - srcAdrs|) > |dstAdrs[q] - srcAdrs|)$
    Begin
        $dstAdrs[q] = mcastAdrs$
    End
    UpdateBitString(mcastAdrs) // in either cases the bitstring has to be updated.
End

4.3.2.3 CDC shell module

Clock adaptation is a vital service of an NI in GALS-based SoC paradigm. Clock adaptation can be realized by employing FIFOs and logic to manage flow of data between FIFOs [94, 95]. To address this demand efficiently, the Quarc NI shell offers an optional CDC (clock domain crossing) module. The CDC module can transfer transactions from IP's clock domain to NoC's clock domain and vice versa. Adopting CDC offers maximum flexibility in plug-and-play and speed-up of the platform. CDC can also be implemented using redundant memory on the IP cores. A functional block diagram of a CDC module is shown in Figure 4.5. It is important to note that the area of CDC module will depend on the number of links in the ingress and egress port of the NI, which eventually is a characteristic of the IP complaint protocol. Note that the Quarc CDC shell module is still at design stage.

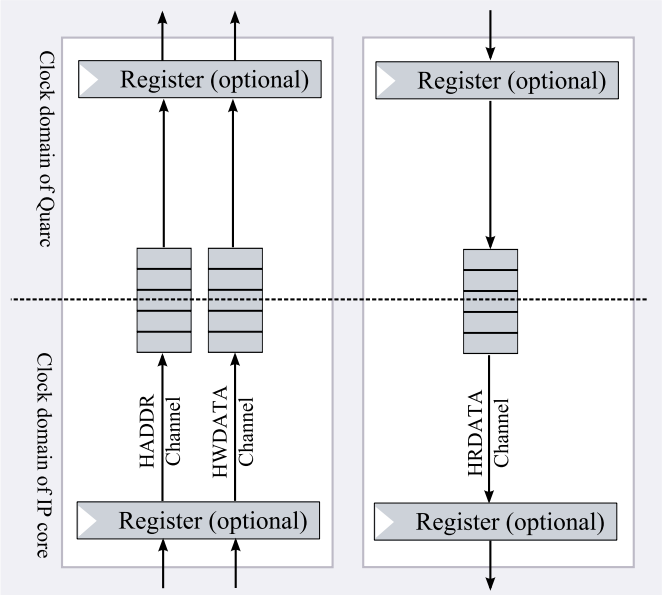


FIGURE 4.5: Functional block diagram of a CDC module.

4.4 Conclusion

The network interface functions as an interface between the IP modules and the NoC architecture. This chapter presented the services that an NI may offer. The chapter also demonstrated the design and implementation of the Quarc NI. The Quarc NI implements the essential services in the kernel module and delegates the instance-specific services to the shell modules. Following this approach a Quarc NoC may be connected to any

IP with proprietary communication protocol by adding a shell module implementing the associated communication protocol. Moreover, the NI can be extended to offer any desired functionality. The design and implementation of three shell modules offering protocol adaptation, multicast communication and clock domain crossing have also been covered in the chapter.

Part III

Cost and Performance Evaluation

Chapter 5

A Comparison Between the Quarc and the Spidergon NoCs

The Quarc NoC improves on the Spidergon STNoC. It is therefore interesting to investigate how the applied modifications affect the performance and cost of the Spidergon STNoC. This chapter presents a comparison between the cost and performance of the Quarc and the Spidergon NoCs. The link usage, router and network interface in both NoCs are considered in cost comparisons. In evaluating performance, the average message latency of unicast and broadcast traffic in different configuration settings are compared in both architectures.

5.1 Cost

The following sections present a comparison between the building blocks of the Quarc and the Spidergon NoCs.

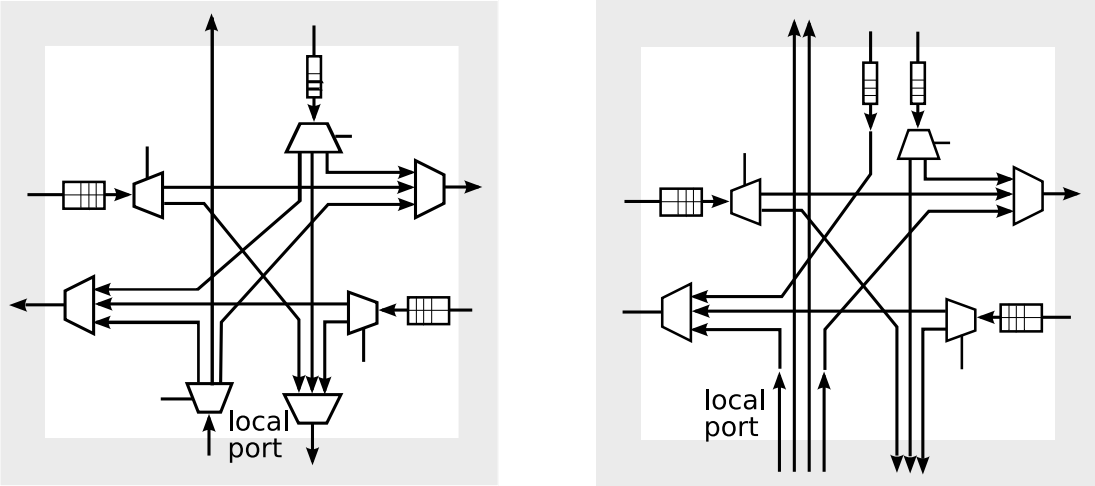
5.1.1 Links

In the NoC domain the contribution of the links to the overall cost of the NoC architecture is typically insignificant. Nevertheless, links and links management are paramount issues in deep sub-microns. For a network of N nodes, the Spidergon STNoC requires $3N$

unidirectional links. As it doubles the across links, the Quarc NoC requires $4N$ unidirectional links. Doubling the across links does not significantly affect design complexity and the cost of the Quarc NoC compared to the Spidergon STNoC.

5.1.2 Router

Routers account for a significant fraction of a NoC cost. In this section we present the router architectures of the Quarc and the Spidergon NoCs. Figure 5.1 shows simplified diagrams for a Spidergon 4×4 router with 1 injection link and 3 network links (Figure 5.1(a)) and the Quarc router (Figure 5.1(b)) with 4 injection links and 4 network links. Both diagrams show minimal architectures for use with deterministic routing, i.e. hardware is tailored to the paths allowed by the routing discipline.



(a) (b)

FIGURE 5.1: Minimal switch architectures for (a) Spidergon and (b) Quarc with deterministic routing.

The main differences are the number of local ingress ports (4 for Quarc) and the doubling of the across links. Further differences are not obvious from the figure. The Quarc router performs a true broadcast, i.e. the ingress multiplexers have a state that clones the flit; the decision logic is very simple (see Section 3.3.3). The Spidergon STNoC can only broadcast by unicast, and therefore needs a more complex logic to decide if a router needs to clone a broadcast packet; furthermore, the ingress packet is not simply cloned but the header flit needs to be rewritten. Given that the Spidergon STNoC implements the broadcast algorithm presented in Section 3.3.3, we assume that the broadcast packets at intermediate routers are stored at the network interface.

We demonstrate that the Quarc router is smaller in size and at the same time is less complex than the Spidergon router and this saving in area outweighs the overheads incurred by additional ports and the area for additional links. To present a comparison between the two architectures, we have implemented 16, 32, and 64-bits wide flit versions of both the Quarc and the Spidergon routers in Verilog targeting the *Xilinx Virtex- II Pro (XC2VP30) FPGA*. The design is optimized for area without using any BlockRAM or Distributed RAM. The silicon cost includes data path and the corresponding flow control units.

For the 32-bits version of the Quarc router the number of occupied FPGA slices is $1,453$ the corresponding version of the Spidergon router occupies $1,700$ FPGA slices. Note that the area occupied by the crossbar and flow control unit are small. This result supports the argument that the Quarc NoC does not have complex crossbar or routing logic, which saves area. A comparison of the cost analysis in terms of *slice count* for various routers employing 16, 32 and 64 bits wide flits are presented in Figure 5.2. As the graphs show, surprisingly, the Quarc router is smaller than the Spidergon router.

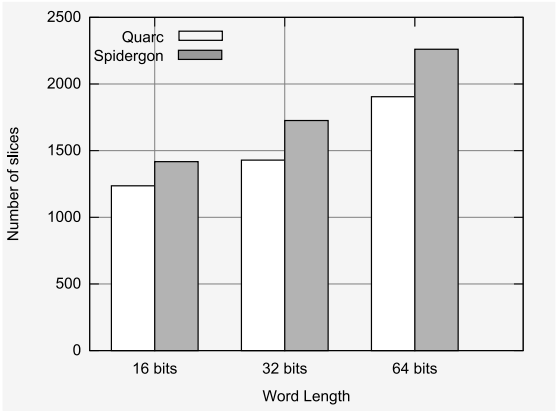


FIGURE 5.2: Cost comparison between the Quarc and the Spidergon routers.

5.1.3 The network interface

The size of a network interface is proportional to the functionality it offers and the buffer requirements. To simplify the comparison between the network interface in the Spidergon and the Quarc NoCs, we assume that the cost of implementing the logic in both architectures is almost equal. This assumption is slightly in favor of the Spidergon NoC, because while both architectures implement similar unicast functionality, broadcast in the Spidergon network interface requires more complicated logic. Therefore,

in the comparison we take into account only the buffer requirements at each network interface.

First, we investigate the buffer requirements in case of unicast communication and then consider the extra buffer requirements to perform broadcast communication. The Spidergon STNoC is a one-port router architecture; it has 1 flit buffer at injection port and 1 flit buffer at consumption port. While, the Quarc NoC is an all-port router and has 4 buffers at injection port and 3 buffers at consumption port.

In the comparison, we assume that each node in network may send broadcast communication. Given that the Spidergon STNoC implements the broadcast algorithm presented in Section 3.3.3 indicates that the network interface at each node, when acting as an intermediate node, must have enough buffer to store a whole packet. Although, in the Quarc NoC the broadcast packets are not stored at intermediate nodes, the whole packet must reside in the network interface before the broadcast operation commences. Therefore, the buffers allocated for the broadcast operation in both architectures are almost identical and slightly over a whole packet size.

According to the above analysis, it can be deduced that the Quarc network interface buffer requirement is 5 flits more than that of the Spidergon network interface. This investment in the network interface pays off in reducing the cost of routers and more importantly, as will be shown in the next section, results in significant performance gain, in particular in multicast and broadcast communication.

In scenarios where only particular nodes generate multicast traffic, the buffer requirements of the Quarc network interface is much less than that of the Spidergon network interface. Assuming that network has $4n$ nodes and only one node sends the broadcast message. The Spidergon STNoC requires $2n$ intermediate nodes to have buffers for a whole packet to implement the broadcast routing algorithm presented in Section 3.3.3. While, the Quarc needs buffer to store a whole packet only at the source network interface.

5.2 Performance comparison

The rest of the thesis employs simulation programs extensively to evaluate performance of a variety of the NoC architectures or to validate the analytical models. The following sections present the basic assumptions defined for all NoC simulators in the thesis, demonstrates the basic Quarc simulator and compares the performance of the Spidergon STNoC against the Quarc architecture.

5.2.1 The basic assumptions of the NoC simulators

The simulation programs model networks adopting different topologies e.g. mesh, Spidergon and Quarc. In those architectures network may exchange unicast, multicast, broadcast, QoS-aware traffic or a combination of these traffic types. Despite of the differences between the NoC architectures they model, a number of assumptions are common in all NoC simulators covered throughout the thesis. All NoC simulators share the following assumptions:

- The size of the queue at source node is not limited.
- The destination node has infinite buffer to store the incoming flits.
- The time consumed to perform switching logic at routers is ignored.
- Regardless of the size of the link, the propagation delay at each link is one cycle.
- The network employs wormhole switching.
- The network adopts deterministic routing.
- Unicast traffic distribution is uniform.
- Latency of a unicast message is regarded as the time from generation of the unicast message at the source node until the time when the last flit of the message is absorbed by the sink at destination.
- The multicast message latency is the time from generation of the multicast message at the source node until the time when the last flit of the message is absorbed by the sink at the last receiver of the multicast message.

- The broadcast message latency is the time from generation of the broadcast message at the source node until the time when the last flit of the message is absorbed by the sink at the last receiver of the broadcast message.
- The size of the messages are identical.
- The buffer depth at input buffer of the routers is one flit.

The rest of the thesis refer to the above assumptions as *simulators basic assumptions.*

5.2.2 The basic Quarc NoC simulator

The Quarc NoC simulator will be used to evaluate the performance of a variety of traffic types in the following chapters. Exchanging different types of traffic requires applying modifications to the architecture. For example, to offer the differentiated services-based QoS, the Quarc network interface and the Quarc router must be QoS-aware. This section demonstrates the architecture of a basic Quarc NoC simulator. The Quarc NoC simulator is a discrete-event simulator operating at flit-level. It is developed using OMNET++ [96]. The schematic of the components of each node in the basic Quarc NoC is shown in Figure 5.3. Note that all connections are via unidirectional links.

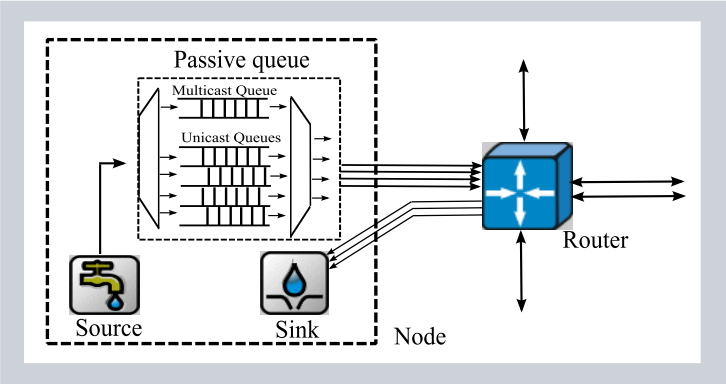


FIGURE 5.3: The schematics of a node in the basic Quarc NoC.

Source: produces messages according to a distribution process.

Passive queue: stores the messages and defines the policy to send them to the router. The passive queue is connected to the router through four injection links. The passive queue has four queues for unicast messages corresponding to each quadrant; and one

queue for multicast messages. Messages are served according to a FIFO policy. In the Quarc NoC a multicast message starts its transmission only when *i*) it is generated before all messages in the unicast queues and *ii*) all required injection links for transmission of the multicast message are free. If the multicast message is generated before all messages in the unicast queues and its transmission does not require access to one or more injection links, the unicast messages at their corresponding queues can be served.

Router: The router implements the routing algorithms presented in Section 3.3. It is connected to three neighbouring routers, a sink and a passive queue. The router is connected to the sink via three consumption links.

Sink: absorbs the messages destined to its associated node.

In the the rest of the thesis we may modify one or more components of the above architecture in order to address the required functionality; e.g. QoS and multicast.

5.2.3 Performing simulation runs

Each simulation experiment should be run until the network reaches its steady state, i.e. until a further increase in the simulated network cycles does not change the collected statistics appreciably. Typically a network exchanges various traffic types at different rates. Therefore, the number of messages generated to suffice network reaching its steady state depends on traffic distribution and hence, varies from one configuration to another. The following algorithm is proposed to guarantee that the statistics gatherings are collected at the network steady state.

The simulation run starts by generating enough messages to guarantee that each node sends on average 1000 messages (it is an initial value) of the most minority traffic type at each node. For example, if network generates unicast and multicast traffic, and multicast forms 5% of the overall traffic, each node in network generates $20,000$ messages. Therefore, a network of 64 nodes in this scenario exchanges $1,280,000$ messages. Using this approach the network simulation runs for 5 times using randomly generated seeds. To avoid the start-up transients, the statistics gatherings of the first run are inhibited and the collected statistics corresponding to runs 2 to 5 are compared against each other.

If the collected statistics at all runs are reasonably close it indicates that generated messages has been enough; averaging over the results from runs 2 to 5 will yield the measures of interest.

A significant discrepancy between the results indicates that the number of messages at each run must be increased. In such circumstances, the number of messages per node are doubled and the simulation runs are repeated. The above steps are followed until the desired accuracy is achieved. Algorithm 3 demonstrates the procedure for performing simulation runs.

Algorithm 3 Performing simulation runs

// Input: $\theta = (\varphi_1, \varphi_2, ..., \varphi_n)$, φ_i denotes the fraction of the traffic of type i , where $\sum_{i=1}^{n} \varphi_i = 1$
// Input: ε denotes the desired accuracy in variance

Step 1: message per node = 1000/minimum(θ)

Step 2: For $i = 1..5$ do

// r_i stores the results gathered from ith run of simulation
2.1 r_i = Run-Simulation(message per node, random seed)

Step 3: If $Var(r) > \varepsilon$

3.1 Double up message per node
3.2 Go to Step 2

Step 4: If $Var(r) \leq \varepsilon$

4.1 Return $\frac{1}{4}\sum_{i=2}^{5} r_i$
4.2 Exit // End of the experiment

5.2.4 The NoC simulators

To evaluate and compare the performance of the Quarc and the Spidergon NoC architectures, we have developed a discrete event simulator for both architectures. The schematic of the components of each node in the Spidergon and the Quarc NoCs are shown in Figure 5.4 and 5.3, respectively. The functionality of the Quarc NoC is as the basic Quarc NoC described in Section 5.2.2. In the Spidergon STNoC, the passive queue has two separate queues to store unicast and multicast messages. The messages

at queues are served according to a FIFO policy. Moreover, in the Spidergon STNoC, the passive queue-router and router-sink connection is via single links.

In both NoCs all nodes produce messages according to a *Poisson* distribution. The reason we have chosen Poisson traffic distributions are *i*) *Poisson* distributions typically offer a good estimate on the average performance of the measures of interest, *ii*) *Poisson* distribution has been widely used in evaluation of interconnection networks, therefore, the results can be compared against a wide range of networks evaluated using similar assumptions, *iii*) specific assumptions on traffic pattern may better capture the measures of interest for a limited number of applications, but they do not address many others.

The simulators operate on the assumptions defined in Section 5.2.1. Moreover, for the Quarc and the Spidergon NoCs the simulators implements the unicast and broadcast routing algorithms presented in Section 3.3.

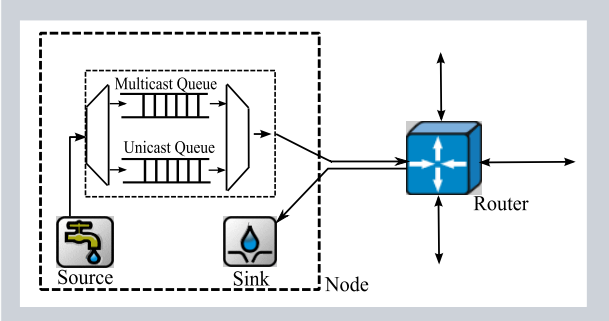


FIGURE 5.4: The schematic of a sample simulation node in the Spidergon STNoC.

5.2.5 Analysis of the simulation results

The performance of the Quarc architecture has been evaluated against the Spidergon NoC for numerous configurations by changing the network size, the message length and the rate of broadcast traffic. The simulation runs are performed according to the algorithm presented in Section 5.2.3.

In graphs, N, M and β represent the number of nodes, the message length and the rate of broadcast traffic, respectively. The horizontal axis in the figures shows the message rate per node per cycle, while the vertical axis describes the average message latency.

We have presented the comparisons in three different configuration settings. In each configuration setting, two parameters (out of three parameters, namely, the network size,

the message length and the broadcast rate) are fixed, while one parameter varies. Figure 5.5 shows the average latency experienced by unicast and broadcast traffic in the Quarc and the Spidergon NoCs in configurations where network size $N = 16$ and broadcast rate $\beta = 5\%$ are fixed, while the message length can be 8, 16 and 32 flits. As the graphs show, doubling the message length almost halves the saturation point in network. This shows that the network saturation depends on the number of flits exchanged in network and it is less sensitive to the number of flits constituting a message.

Figure 5.6 compares the simulation results for networks having 16, 32 and 64 nodes with a fixed message length of 16 flits and 10% broadcast traffic. The broadcast rate of 10% may seem unlikely in a NoC-based application, however, it can clearly serve the purpose of the performance comparison between the two NoC architectures. As it is evident from the graphs, doubling the network size brings down the network saturation point. This is due to increase in the number of packets competing for the network links.

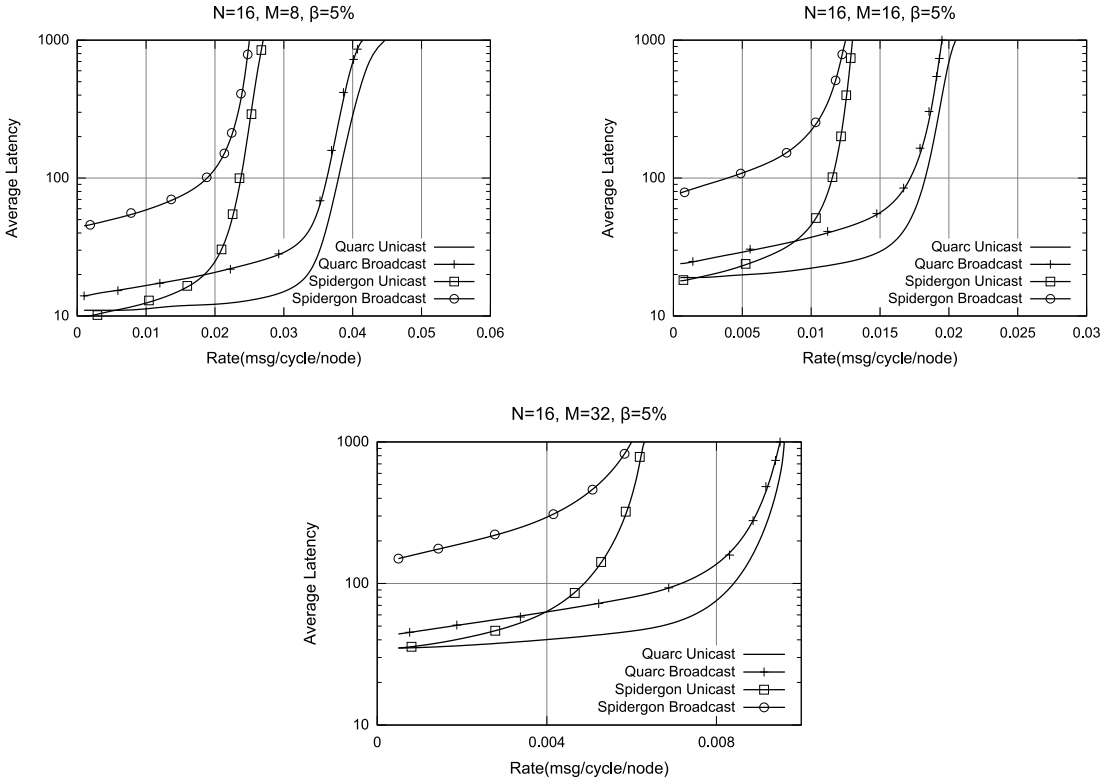


FIGURE 5.5: Comparison of the Quarc and the Spidergon NoCs for the message lengths of 8, 16 and 32 flits.

As can be seen from the figures, the Quarc NoC outperforms Spidergon over the complete range of N, M and β . The most striking performance difference is clearly observed for broadcast traffic, with almost an order of magnitude improvement on the average latency.

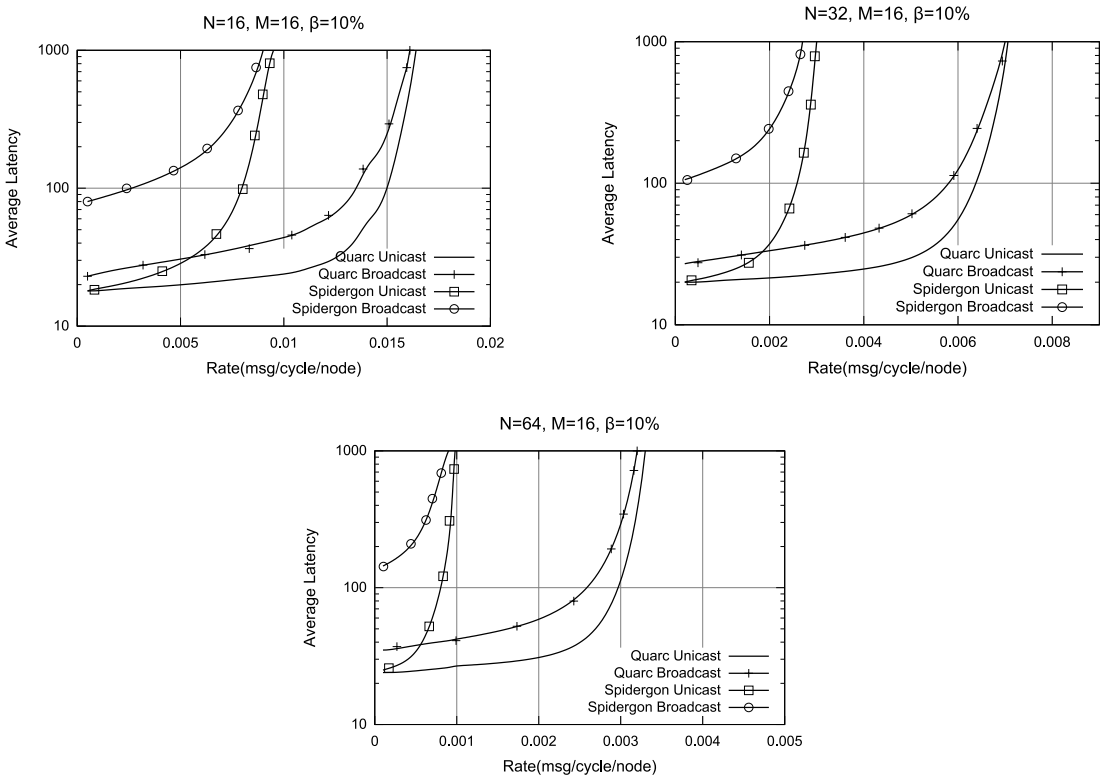


FIGURE 5.6: Comparison of the Quarc and the Spidergon NoCs of 16, 32 and 64 nodes.

Note that larger networks show the performance gain more clearly. The unicast latency is, on average, at least a factor of 2 lower.

The graphs in Figure 5.7 compare the average latency in the Quarc and the Spidergon NoCs for the configuration where the network size $N = 64$ and the message length $M = 16$ are fixed, while the broadcast rate, β, varies between 0%, 5% and 10%. The graphs reveal that the Quarc NoC is highly capable of sustaining broadcast traffic. As can be seen, the injection of broadcast traffic into the Spidergon NoC severely reduces the sustainable load in network. While, in the Quarc NoC the adverse impact of broadcast traffic on the sustainable load and on the performance of unicast is hardly appreciable.

5.3 Conclusion

The results presented in this chapter demonstrated that the architectural improvement of the Spidergon STNoC to the Quarc NoC enhances the performance significantly, while does not incur an appreciable extra cost. The simulation results have shown that the Quarc NoC outperforms Spidergon in terms of the message latency over the complete

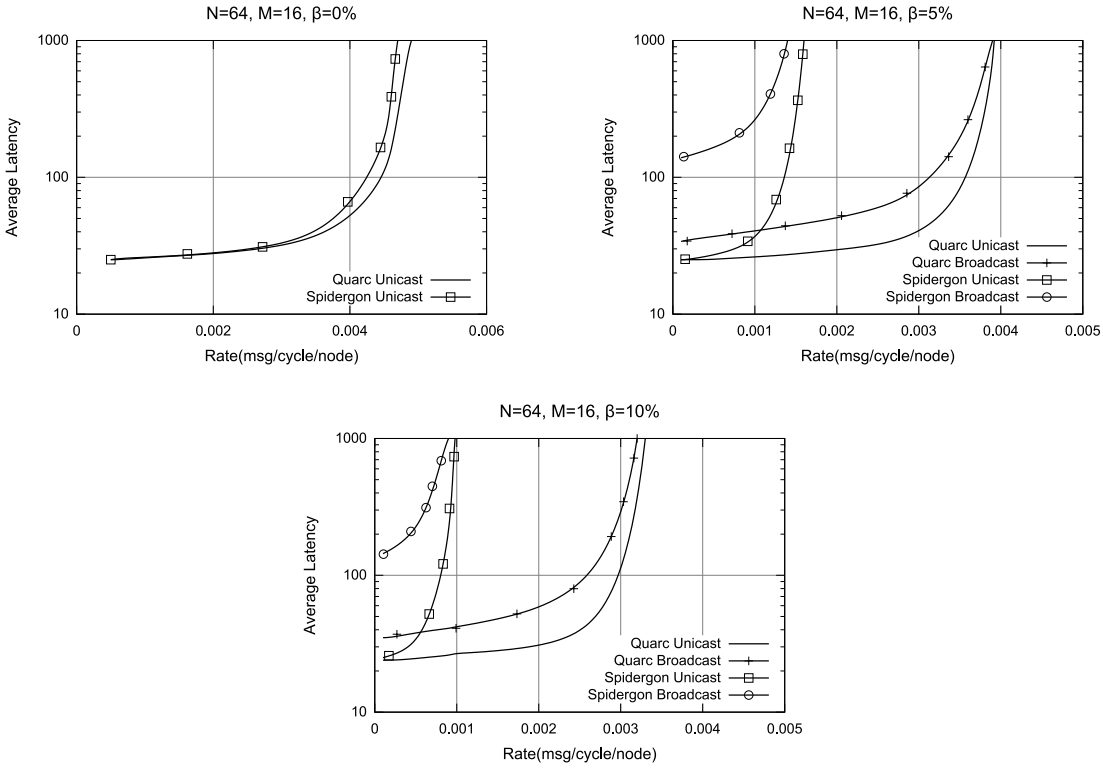


FIGURE 5.7: Comparison of Quarc and the Spidergon for broadcast rates of 0%, 5% and 10%.

range of number of nodes, the message lengths and the broadcast rate. In the absence of broadcast traffic, the Quarc NoC slightly outperforms the Spidergon STNoC. The superiority of the Quarc NoC will become evident once broadcast traffic is injected to network. Our experiments revealed that, on average, the Quarc NoC delivers unicast and broadcast traffic 2 times and 10 times faster than the Spidergon STNoC does.

Performance gain typically comes at a cost. The chapter compared the link usage, cost of the router and the network interface in both the Quarc and the Spidergon architectures. The analysis has demonstrated that by doubling the across links, the Quarc NoC requires more physical links than the Spidergon scheme does. Given that Spidergon and Quarc offer the same functionality, it is has been shown that in the extreme case, where every node may generate broadcast traffic, the Quarc network interface is slightly larger than the Spidergon network interface.

To compare the routers, an FPGA implementation of the routers in different configuration settings for both architectures has been developed. The cost analysis has shown that the Quarc router is smaller than the the Spidergon router. This is mainly due to the simpler routing algorithm adopted in the Quarc NoC.

Chapter 6

A Comparison Between the Quarc NoC and Mesh

The regular mesh is one of the most popular topologies for NoCs, used by e.g. Cliche [7], aSoC [97] and Hermes [69]. These architectures have no or limited hardware support for multicast communication. A number of topology-independent architectures such as ï¿½thereal [14] offer mechanism to perform multicast communication. However, due to resource constraints of the NoC development, they resort to techniques realizing the feature with minimum cost. Thus, neither approaches fully exploit the potential of the mesh topology to deliver multicast traffic.

In contrast, the Quarc NoC is designed for efficient multicast communication. This chapter aims at comparing the Quarc NoC against a mesh-based NoC architecture. To present a fair comparison between the two NoCs, based on the experiences in parallel computers, we implement a most efficient multicast routing algorithm for the mesh-based NoC and compare its performance and cost against the Quarc NoC.

The chapter begins by presenting the unicast and multicast routing algorithms in the mesh topology. It follows by a description of the router and the network interface in a mesh-based NoC architecture. And finally, the cost and performance comparisons are presented.

6.1 Routing

This section presents the unicast and broadcast routing algorithm in the mesh-based NoC reference model.

6.1.1 Unicast routing

We assume that mesh adopts XY routing algorithm. The XY routing algorithm is deadlock free and adheres to resource constraints of the NoC development. The network uses unidirectional links to connect router-to-router and IP core-to-router.

6.1.2 Broadcast routing

Implementing fast and scalable multicast communication for wormhole-routed interconnection networks has been extensively researched for parallel computers. The adopted approaches can be broadly classified in two categories, depending on whether the system performs collective communication merely by relying on unicast-based communication, or offers hardware supports for multi-destination messages. Umesh [98] and SPUmesh [99] are two unicast-based algorithms in which the objective is to minimize the height of the multicast tree.

The systems supporting multi-destination message passing can perform collective communication operations more efficiently by using hierarchical leader-based schemes [98, 100]. Given a multicast destination set, these schemes group the destinations in a hierarchical manner to minimize the number of unicast/multi-destination worms required to cover all destinations. The hierarchical leader-based routing algorithms presented in [98, 100] perform poorly in presence of multiple multicast. That is because several multicast sources use the same leader nodes. SQHL [99] and SCHL [99] were proposed to improve the communication performance in the presence of multiple multicast messages. The SQHL and SCHL algorithms involve position of the source node in decision making on selecting the leader nodes. In [31] it is shown that SCHL is the most efficient algorithm to perform multicast communication in a mesh-based network. In [101] Panda et. al. showed that adopting the BRCP for performing multicast communication results in more efficient resource utilization, and at the same time improves the network

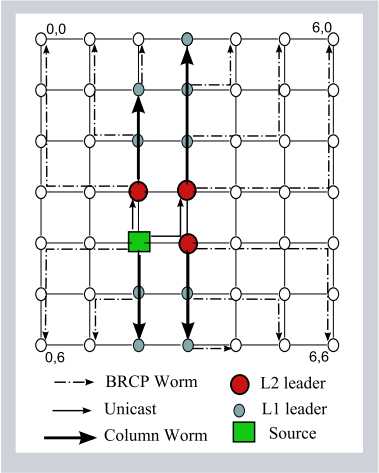


FIGURE 6.1: Broadcast in mesh.

performance. Therefore, we assume that mesh adopts a BRCP-based SCHL algorithm to perform the broadcast. The algorithm for leader selection for n-dimensional mesh is presented in [99].

In the BRCP-based SCHL algorithm employed in mesh, the BRCP model provides three different kinds of paths (R, C, and RC) on an XY routed 2D mesh. To avoid deadlock in such a network, the router must have four consumption links [101].

Figure 6.1 demonstrates a broadcast scenario for a non-bordering node in mesh. The source node sends the packets to the L_2 (Level 2) leaders using multi-destination messages. Upon receiving the broadcast packets, the L_2 leaders are responsible to send the packets to the L_1 leaders, which are on the same columns as the L_2 leaders. To do so, each L_2 leader prepares a multi-destination packet destined to the possible L_1 leaders. Each intermediate L_1 leader, between the L_2 and the last L_1 leader absorbs the flits and forward them to the next L_1 leader simultaneously. While, the last L_1 leader consumes the flits. Once the whole packet is received by an L_1 leader, it creates a multi-destination packet to serve the nodes located on a path conformed to the RC paths originating from that L_1 leader.

6.2 Implementation of a mesh-based NoC

The following sections present building blocks of a mesh-based NoC including physical links, router and the network interface.

6.2.1 Number of links

Given that the network has N nodes, the Quarc NoC and square mesh require $4N$ and $4\sqrt{N}(\sqrt{N}-1)$ unidirectional links, respectively.

6.2.2 The mesh router

Figure 6.2 presents a functional block diagram of the architecture of a non-bordering router in the mesh-based NoC. The router adopts input queuing and thus, must have buffers to store the incoming flits from neighboring routers. Non-bordering, edge and corner routers are connected to 4, 3 and 2 routers, respectively. We assume that the depth of the buffers at input queues is one flit and their quantity at each input is proportional to the number of virtual channels sharing the physical link. The router implements wormhole switching and deterministic routing. The functional block diagram of the mesh router is similar to the one described for the Quarc NoC presented in Section 3.6.

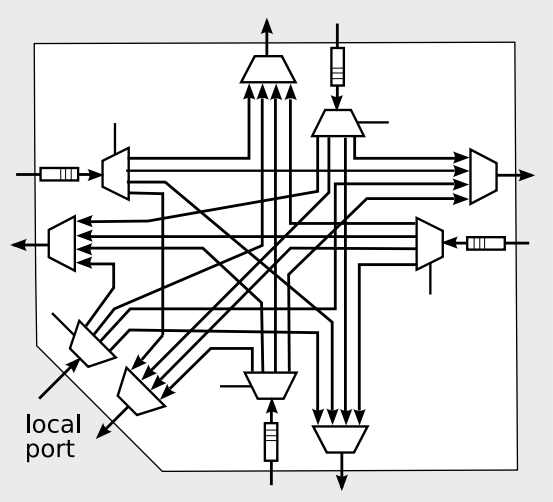


FIGURE 6.2: Schematic of the structure of a non-bordering router in mesh.

6.2.3 The mesh network interface

Figure 6.3 depicts a schematic of the network interface in the mesh-based NoC. The network interface has one buffer to store the flit at injection link and four buffers to store the incoming flits at the consumption links. The transmission controller implements the logic to route a flit from the local IP core or multicast buffer to the buffer at injection

link. The Flow control unit (FCU) at the transmission path (left FCU unit) decides when a flit to leave the buffer at injection link. Moreover, it informs the transmission control unit about the state of the buffer at injection link.

The reception controller decides whether a packet has to be sent only to the IP core or it has to be forwarded to the multicast buffer and the IP core simultaneously. The reception controller manages flows from buffers at consumption links to the IP core or/and multicast buffer. The flow control unit (FCU) at consumption path manages the flow of flits from the buffers at router to the buffer at the network interface. The multicast buffer in the diagram is to store the packet only if the IP module is a leader node.

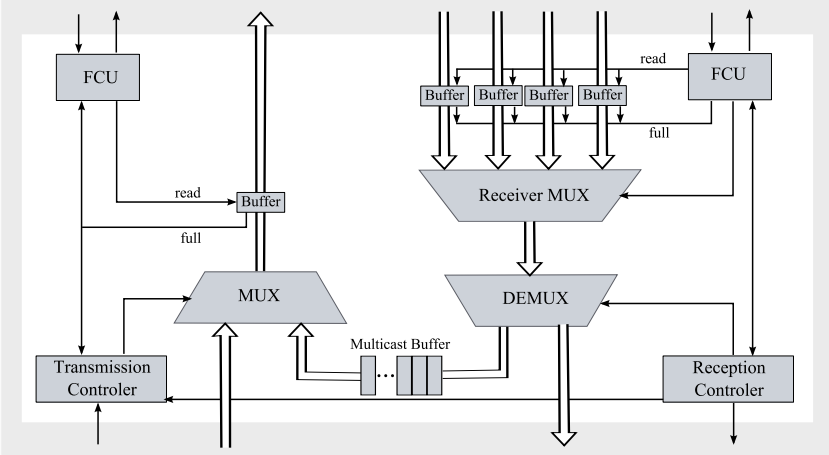


FIGURE 6.3: The network interface in the mesh-based NoC.

6.3 Cost comparison

In this section, we present a detailed comparison between the FPGA implementations of the routers and the network interfaces in the Quarc and the mesh-based architectures. The comparison is performed for architectures with various configuration settings. We assume that every tile of the NoC hosts an IP core, typically a microprocessor with local memory.

To present a comparison between the two architectures, we have implemented 16, 32, and 64-bits wide flit versions of both the Quarc and the mesh-based schemes with 2, 4, 6, 8 and 10 virtual channels in Verilog targeting the *Xilinx Virtex-II Pro FPGA*. Table 6.1 and Table 6.2 present the cost of the router and the network interface in both the Quarc

and the mesh-based architectures, respectively. At router and the network interface in both architectures, the buffers are synthesized using *Xilinx Block SelectRAM+ (BRAM)*. Note that as there is no notion of link in FPGA implementation, there is no need for comparison between the link usage in the NoC architectures.

As expected, due to having multiple injection links and logic for quadrant calculation, the network interface in the Quarc NoC is larger than in the mesh-based NoC. On the other hand, the router in the Quarc NoC is smaller than the mesh-based architecture router.

Router cost	16-bits				32-bits				64-bits			
	Quarc		Mesh		Quarc		Mesh		Quarc		Mesh	
No. of VCs	Slice	BRAM(bit)	Slice	BRAM(bit)	Slice	BRAM(bit)	Slice	BRAM(bit)	Slice	BRAM(bit)	Slice	BRAM(bit)
2	653	4(144)	1016	4(144)	832	4(272)	1327	4(272)	1188	8(528)	1944	8(528)
4	659	4(288)	1025	4(288)	841	4(544)	1340	4(544)	1204	8(1030)	1967	8(1030)
6	667	4(432)	1034	4(432)	852	4(816)	1355	4(816)	1222	8(1540)	1992	8(1540)
8	677	4(576)	1044	4(576)	865	4(1060)	1371	4(1060)	1242	8(2060)	2018	8(2060)
10	689	4(720)	1056	4(720)	880	4(1320)	1388	4(1320)	1264	8(2570)	2046	8(2570)

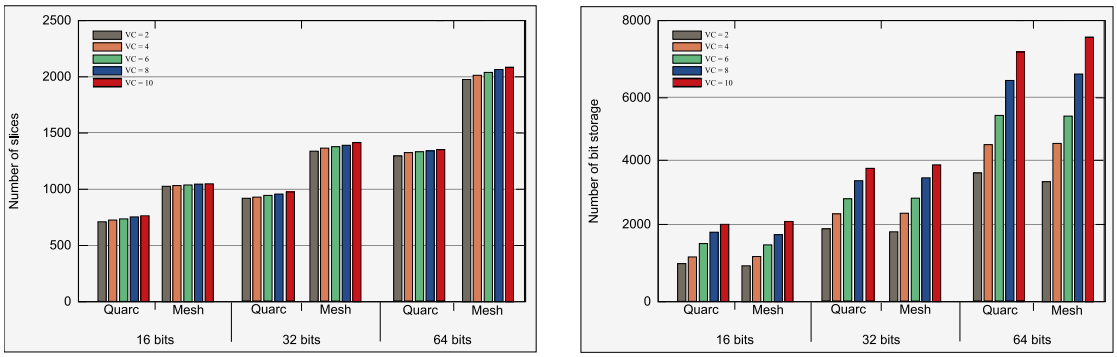
TABLE 6.1: The cost of router in the mesh-based and the Quarc architectures.

NI cost	16-bits				32-bits				64-bits			
	Quarc		Mesh		Quarc		Mesh		Quarc		Mesh	
No. of VCs	Slice	BRAM(bit)	Slice	BRAM(bit)	Slice	BRAM(bit)	Slice	BRAM(bit)	Slice	BRAM(bit)	Slice	BRAM(bit)
2	87	8(832)	36	6(756)	89	8(1568)	38	6(1428)	93	16(3040)	40	12(2772)
4	89	8(944)	38	6(900)	91	8(1776)	40	6(1700)	95	16(3440)	42	12(3300)
6	91	8(1056)	40	6(1044)	93	8(1984)	42	6(1972)	97	16(3840)	44	12(3828)
8	92	8(1168)	41	6(1188)	94	8(2192)	43	6(2244)	98	16(4240)	45	12(4356)
10	94	8(1280)	43	6(1332)	96	8(2400)	45	6(2516)	100	16(4640)	47	12(4884)

TABLE 6.2: The cost of the network interface in the mesh-based and the Quarc architectures.

A comparison between the overall cost of router and the network interface in terms of *slice count* and *bit storage count* for various configurations in two architectures are presented in Figure 6.4.

Comparing the Quarc and the mesh-based schemes in terms of the number of FPGA slices occupied reveals that in 16, 32 and 64-bits implementations, on average, the mesh-based NoC uses 41.6%, 47.8% and 54.4% more slices than its Quarc NoC counterpart. In all implementations, the Quarc NoC uses 20% more BRAMs than the mesh-based NoC does. However, surprisingly, the actual bit storage in the mesh-based NoCs employing



(a) (b)

FIGURE 6.4: A comparison between the overall cost of router and the network interface in the Quarc and the mesh-based architectures in terms of the number of (a) FPGA slices and (b) bit storage.

$VC > 6$ virtual channels is more than bit storage in the Quarc NoC. This is of course because the Quarc NoC has one consumption link less than the mesh-based NoC (given that the injection links at both architectures are shared by two virtual channels). This results clearly show that the ASIC implementation of the network interface in the Quarc and the mesh-based schemes are expected to be realized at almost the same cost.

The reason for this is that BRAMs have a minimum size regardless of the number of bits actually stored. In fact that is even true for Distributed RAMs. An ASIC implementation would have exactly as much storage as required and would therefore be much more area-efficient.

Also, note that in the Quarc network interface, the multicast buffer which accounts for a significant proportion of the component size is required only if its corresponding IP core is a potential source of multicast communication. Therefore, if the IP core is not a source of multicast communication, the size of BRAMs in the Quarc network interface will be less than the BRAMs in the mesh-based NoC.

These results show that the Quarc router does not occupy as much silicon area as the mesh-based NoC does. On the other hand, the Quarc network interface requires more memory (BRAM) to store flits than the mesh-based NoC does. This extra cost for the Quarc network interface actually improves the performance of the Quarc NoC as shown in the next section.

6.4 Performance comparison

To evaluate and compare the performance of the two architectures we have developed the Quarc and the mesh-based NoC simulators. The performance of the Quarc NoC can be derived using the basic Quarc simulator demonstrated in Section 5.2.2. The schematics of the components of a sample node in the mesh-based architecture is presented in Figure 6.5. Functionality of the components of the mesh-based NoC is identical to the ones presented in Section 5.2.2 for the Quarc NoC. The passive queue in the mesh-based NoC has a single queue to store unicast, multicast and broadcast messages.

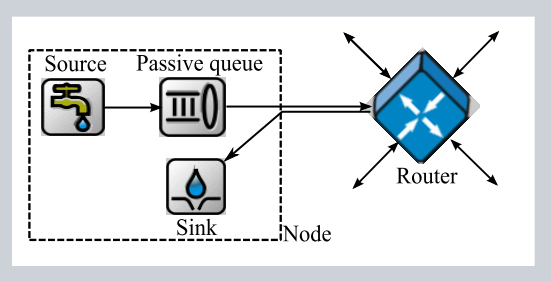


FIGURE 6.5: The schematic of a sample node in a mesh-based NoC.

Both the Quarc and the mesh-based NoC simulators operate on the basic assumptions defined in Section 5.2.1. Moreover, packets of equal size, 32-*flits,* are generated at each node according to a *Poisson* process. It is assumed that network generates both unicast and broadcast traffic, where broadcast is $\beta = 0\%$, $\beta = 10\%$ and $\beta = 20\%$ of the overall traffic generated at each node. Destinations of the unicast packets are randomly selected. The unicast and broadcast routing algorithms employed by the Quarc architecture are presented in Sections 3.3.1 and 3.3.3, respectively. The mesh-based architecture employs *XY* [54] unicast routing algorithm. To perform broadcast, mesh adopts the routing algorithms described in Section 6.1.2. Each simulation experiment runs according to the algorithm presented in Section 5.2.3.

The performance comparison is conducted for the Quarc and the mesh-based networks employing 2, 4, 6, 8 and 10 virtual channels. The performance of the Quarc architecture is evaluated against the mesh-based NoC for numerous configurations. Figure 6.6 presents the comparisons for a number of configuration settings. The horizontal axis in the graphs show the rate in terms of *flit/cycle/tile*, and the vertical axis describes the latency.

6.4.1 Message latency

Our experiments show that before approaching the saturation points, the mesh-based and the Quarc NoCs deliver the unicast packets with almost equal performance. The Quarc NoC outperforms the mesh-based NoC in smaller networks of up to around 30 nodes while, the mesh-based NoC performs better in larger networks.

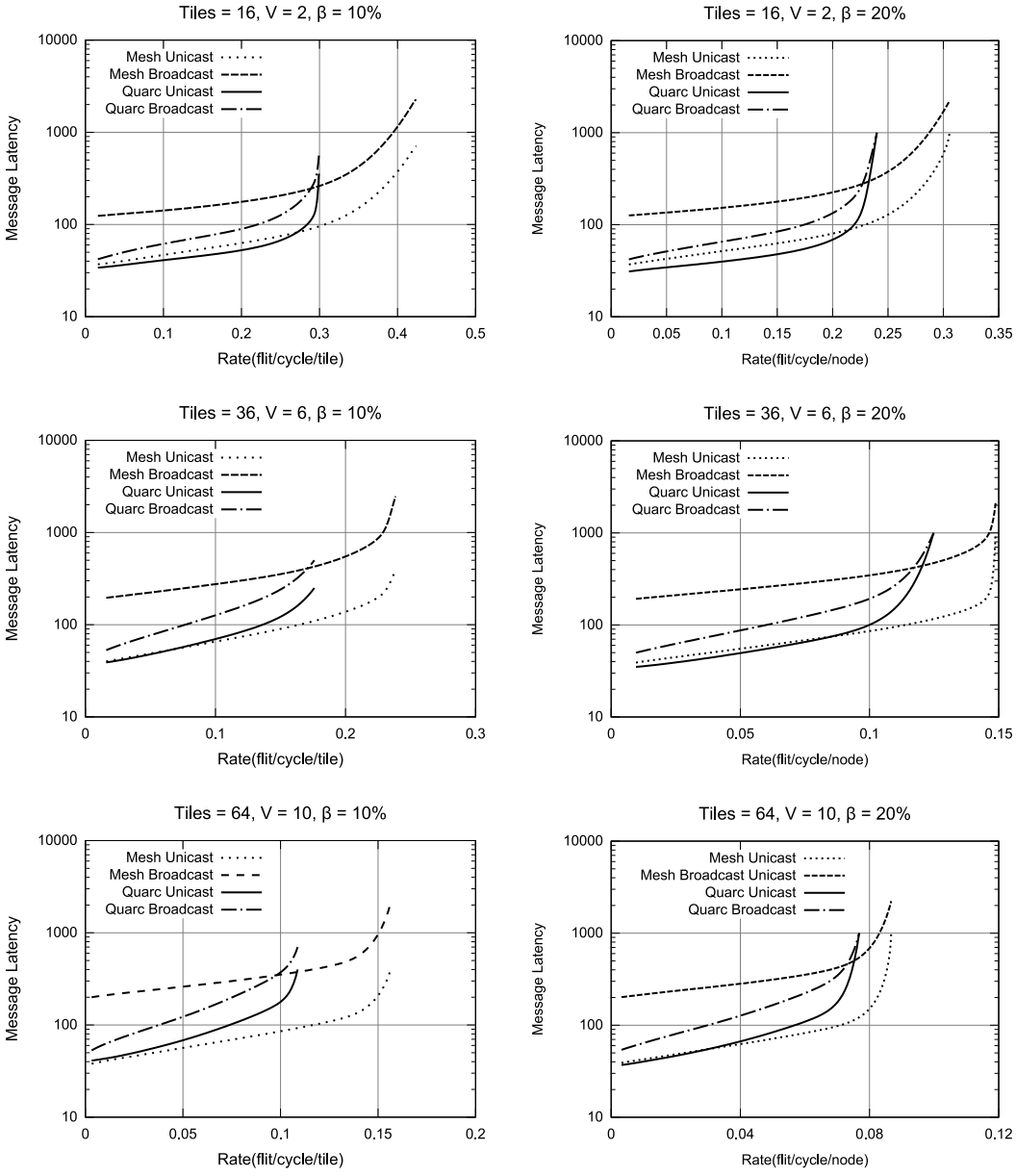


FIGURE 6.6: A comparison of unicast and broadcast in the mesh-based and the Quarc NoCs.

When it comes to broadcast, as the graphs in Figure 6.6 show, the Quarc NoC significantly outperforms the mesh-based NoC in all configuration settings before saturation

points. Our experiments demonstrate that, on average the Quarc NoC performs a broadcast operation over two times faster than the mesh-based NoC does.

6.4.2 Effect on unicast

It is generally desirable to implement the collective communication operations without sacrificing the performance of unicast communication. The conducted experiments show that injecting $\beta = 10\%$ ($\beta = 20\%$) broadcast traffic to the mesh-based and the Quarc NoCs, on average increases the unicast latency by 24.2% (72.2%) and 12.1% (28.6%), respectively. Our findings also suggest that the performance of unicast communication in smaller mesh-based networks is more adversely affected to the presence of broadcast communication, in the Quarc NoC the adverse effect of broadcast communication is much less than in the mesh-based NoC.

6.4.3 Broadcast tolerance

Parallel to observing the effect on the performance of unicast communication, we investigated the network sustainable load in response to injection of various broadcast traffic rates. Table 6.3 shows how injecting broadcast traffic affects the network sustainable load. The results reveal that injecting $\beta = 10\%$ ($\beta = 20\%$) broadcast traffic to the mesh-based and the Quarc NoCs on average decreases the network sustainable load by 51% (68.2%) and 38% (54%), respectively. These results combined with the effect on the performance of unicast communication demonstrates that the Quarc NoC is clearly more tolerant to the presence of multicast communication traffic.

6.4.4 Network saturation

From the graphs in Figure 6.6 it is evident that compared to the Quarc NoC, the mesh-based NoC can sustain more traffic before it saturates. Explaining the reason why the mesh-based NoC tolerates more traffic requires detailed traffic analysis at each link in both architectures. In particular, we are interested in stochastic properties of the first link that saturates in network. Obviously, this is the link that exchanges the highest amount of traffic. Traffic distributions at network links while nodes generate messages according to a *Poisson* process in the mesh-based and the Quarc networks can be found

in [102] and [103], respectively. The results show that the links at the middle of mesh and surrounding links in the Quarc NoC exchange the highest volume of traffic compared to other links.

Let's assume that (*i*) mesh is square, (*ii*) both mesh and the Quarc NoC have the same number of nodes, (*iii*) each node sends messages to other nodes uniformly according to a *Poisson* distribution with average rate of λ and (*iv*) network is stable. Moreover, we denote by ϕ_Q and ϕ_M the traffic rate at the links exchanging the highest volume of traffic in the Quarc NoC and the mesh-based NoC, respectively. Adopting results from [102] and [103] shows that $\phi_Q > \phi_M$, for a fixed λ. Therefore the weakest link (and consequently, network) in the Quarc NoC saturates earlier than mesh.

Broadcast rate	32-bits					
	16-bits		32-bits		64-bits	
	Quarc	Mesh	Quarc	Mesh	Quarc	Mesh
0	0.7	0.79	0.27	0.48	0.14	0.37
10	0.43	0.49	0.16	0.23	.09	0.13
20	0.33	0.35	0.18	0.14	0.06	.08

TABLE 6.3: The effect of injecting broadcast traffic on the network sustainable load.

6.4.5 The effect of adding virtual channels on network saturation

Table 6.4 shows the network saturation improvement in response to adding virtual channels to a base network. We assume that the base network employs 2 virtual channels. In the table, the first column represents the number of virtual channels sharing each physical link. The other columns show the network saturation improvement compared to the base network. Our experiments reveal that the Quarc NoC responds more efficiently to additional virtual channels.

Virtual channels	32-bits					
	16-bits		32-bits		64-bits	
	Quarc	Mesh	Quarc	Mesh	Quarc	Mesh
4	%25	%18	%26	%17	%35	%16
6	%39	%25	%47	%28	%45	%30
8	%54	%34	%57	%33	%63	%33
10	%60	%37	%68	%38	%66	%38

TABLE 6.4: The effect of increasing virtual channels on network saturation.

The reason for the different responses to additional virtual channels lies in topological attributes of the architectures. Let's assume that V virtual channels share each physical link (including injection links). In mesh, V virtual channels of a non-bordering link may receive messages from $3 \times V$ virtual channels from neighboring network links and from V virtual channels from injection link. While, V virtual channels of a surrounding link in the Quarc NoC receive messages from $2 \times V$ network link virtual channels and from V injection link virtual channels. In other word, V virtual channels of a non-bordering link in mesh and a surrounding link in the Quarc NoC are loaded by traffic from $4 \times V$ and $3 \times V$ virtual channels, respectively. This means that in presence of high loads the weakest link in mesh more quickly reaches the point where it exploits the full potential that network can gain from virtual channel multiplexing. Therefore, the mesh-based NoC responds more poorly to the addition of the virtual channels.

6.5 Conclusion

This chapter presented a comparison between a mesh-based NoC architecture and the Quarc NoC. Performing efficient multicast/broadcast is the most outstanding feature of the Quarc. Therefore, to demonstrate a fair comparison we have implemented a most performant broadcast routing algorithm for the mesh-based NoC as well.

To compare the cost, we have developed FPGA implementations of the router and the network interface in both architectures for numerous configuration settings in terms of the flit width and the number of virtual channels. The cost comparison has shown that the network interface in the mesh-based NoC can be realized using less resources compared to the Quarc network interface, while the Quarc router implementation requires less resources compared to router in the mesh-based NoC. The analysis has shown that overall cost of implementing the router and the network interface in both architectures are almost equal.

The simulation experiments have shown that in operational range (before saturation point) the Quarc outperforms the mesh-based NoCs in delivering unicast messages in small networks of up to 30 nodes. Whereas, mesh-based NoC exchanges unicast messages faster in larger networks. The Quarc NoC significantly outperforms mesh in performing broadcast communication. Moreover, the analysis of the results has demonstrated that

the Quarc NoC is more tolerant to the presence of multicast traffic and responds more effectively to the additional virtual channels.

Part IV

Analytical Performance Models of Communication in the Quarc NoC

Chapter 7

Performance Evaluation Tools and Techniques

The successful operation of a NoC-based application depends to a large extent on the performance of the underlying communication architecture. Thus, it is crucial to ensure that the network can deliver the performance requirements of the application. This can be achieved by comparing the performance demands of the application against the performance measures derived when the network operates based on widely used assumptions or benchmarks. The most widely used approaches to derive the performance measures of a communication architectures are analytical models and simulation programs. Depending on the characteristics of the system to be evaluated, accepted accuracy and cost, one technique may be given preference to others. This chapter presents the performance evaluation tools and techniques and investigates their applicability into the NoC realm.

7.1 Analytical models

Analytical models are regarded as a cost-effective method to evaluate the performance of the systems [110]. Most analytical performance models for interconnection networks are based on stochastic processes and queuing theory. This section presents a brief review of Markov models and queuing networks.

7.1.1 Markov process

Markov processes belong to a class of stochastic processes which is characterized by its limited historical dependency or memory-less property. Like other stochastic processes, a Markov process is defined as $\{X(t) : t \in \tau\}$. $X(t)$ is a random variable which represent the state of the process at time t and can assume any values from state space S. And τ is regarded as time. The dynamic behavior of a Markov process is determined by the time spent at each state and transitions between the states. Finding the steady state probability distribution of a Markov process is achieved by solving a *generator matrix* which its elements are the transition rate from one state to other potential states.

A fundamental characteristic of any Markov process is its memory-less or Markovian property. The Markovian property simply says that the whole history of a process is summarized in its current state. In other words, the holding time in the current state and the transition to the next state is independent of the times spent in the previous states and sample path taken to reach to the current state.

The memory-less view of the system is compatible with behavior of many activities in communication and computer systems. This property when combined with the simple and well-know methods of solving a Markov process is probably the reason behind the versatile and successful use of Markov processes in modeling a wide range of systems.

Markov processes can be used to derive the stationary distribution of large class of birth-death processes. They are widely used in evaluating variations of $M/M/1$ queues which are derived by changing the number of servers and buffer size of the queue [104].

The mathematical notations of Markov processes typically are not intuitive enough to give an insight into the functionality of the system they model. Moreover, as the system complexity and consequently the state space grows, it is not easy to capture all details in a diagram. To abstract the complexity of Markov models, a number of tools and languages are introduced such as SPA (stochastic process algebra) [105] and SPN (stochastic Petri nets) [106]. The primitive notations of these tools and modeling languages map to their counterpart in a Markov process. In fact, the analytical results these tools provide are mainly produced by the underlying Markov processes corresponding to the models.

The major drawback of Markov models is the size of the state space. As the size of the state space grows solving the generator matrix and deriving the performance measures is not a trivial task or even impossible with current methods and technology support. A traditional challenge in the domain has been to solve the Markov processes of large systems.

To investigate the suitability of Markov-based techniques to evaluate the performance of NoCs, we have presented a PEPA [105] model for a simple network [107]. The system consists of three nodes communicating through unidirectional links using wormhole switching. We assume having a uniform traffic in which each node chooses destinations arbitrarily, and there is an equal *Poisson* traffic between each pair of nodes in the system.

Implementing such a small system, as is shown [107], involves a large PEPA program with over $18,000$ states which prohibitively grows as the network size increases. Thus, modification, verification and debug of such models in a system consisting of tens of nodes is not unlikely to be impractical.

We selected PEPA to evaluate the performance of the simple system mentioned above. Although it offers a level of abstraction over using a Markov process directly, the model is still too complex to handle and unlikely to scale. It is not very hard to deduce that modeling the system with stochastic Petri nets (SPN) [106] would be almost as complicated as the PEPA model. This is because both PEPA and SPN eventually map to an underlying Markov process. Therefore, to handle complexity and size of the real NoC models, which are at least hundreds of times larger and more complicated than the simple system analyzed in [107], employing higher level techniques are necessary.

7.1.2 Queuing networks

Queuing theory and queuing networks have been used extensively to evaluate the performance of computer and communication systems. In particular the interest is on the product form queuing networks. In a product form queuing network, the stationary distribution of the network is the product of the distribution of each queue analyzed in isolation from the network. In other words, queuing networks provide a compositional approach to handle complexity involved in analysis of large systems consisted of smaller subsystems.

Using the already established concepts, deriving the measures of interest at each queue requires arrival and service rate at each queue. Once the characteristics of each queue is known, it is possible to represent the state of the system as a product of individual queues. Of course, such an approach is very limited in respect to the type of arrival and service time of the queues.

7.2 Simulation programs

Simulators are computer programs that mimic the operations of the real world systems. Developing simulation programs for large systems is often a non-trivial task and running the application requires powerful computer systems in operation for a long period of time. As capturing all details of the real system in a simulation program may not be feasible or cost-efficient, usually the system is modeled at a higher level of abstraction.

Simulation programs have a few advantages over analytical models. In contrast to analytical models which typically assume simple traffic distributions, in simulation the inter-event times are not restricted. Also, since there is no concerns about the state space, models can be more realistic and larger. Moreover, simulations enable studying the transient behavior of the models which is not often easily derived from analytical models. And finally, in contrast to analytical models, the measures derived from simulation programs are not only mean values, and it is possible to derive minimum and maximum of the measures of interest.

The general simulation tools may be used for evaluating NoCs. However, due to peculiarities in the field, a number of simulators have been developed for the domain. Following introduces a number of tools that might be employed for simulating NoCs.

- The On-Chip Communication Network (OCCN) [108] tool is an efficient, open-source research and development framework built on top of SystemC for specification, modeling and simulation of the on-chip communication architectures.
- OMNeT++ [96] is a public-source, component-based, modular and open-architecture simulation environment with strong graphical user interface support and an embeddable simulation kernel. Its primary use is simulation of communication networks. However, due to its generic and flexible architecture, it has been successfully used

in other areas such as IT systems, queuing networks, hardware architectures and business processes as well.

- As the dominant switching method in on-chip networks is wormhole switching a number of simulators have been developed to model wormhole switched networks. The flit level simulator proposed by Smith [109] is an example of such simulators.

7.3 A comparison between performance evaluation techniques

The performance evaluation techniques reviewed so far are the most widely used techniques in analysis of computer and communication systems. This section compare these techniques in respect to *i*) learning time, *ii*) assumptions, *iii*) size and scalability and *iv*) accuracy and reliability.

7.3.1 Learning

Most Markov-based tools provide a graphical representations of the interaction between the components of the system, this feature makes them more intuitive and easier to assimilate, in particular for small systems. Learning stochastic process algebras require more time to use efficiently. Queuing networks, if viewed at a high level of abstraction can be manipulated by simple algebra. However, understanding the underlying concept in queues and queuing networks and mastering them is time consuming. Simulation models generally require writing and debugging a complex computer program. Developing a good simulation model often requires detailed knowledge of the system being represented as well as a thorough understanding of the statistical techniques.

7.3.2 Assumptions

A fundamental assumption in Markov-based models is the memory-less property of the system. This assumption conforms to the behavior of many activities in computer and communication domain. In queuing networks, typically the product form of the queues provide the measures of interest. The assumption in such systems is that the

system is *quasi-reversible* [110]. This assumption is hardly met in reality. Simulation programs typically are able to investigate the model in almost every situation and no severe assumptions are enforced.

7.3.3 Size and scalability

The major handicap in Markov processes is the size of the model. These techniques typically require the whole state space to be manipulated by math software programs. The maximum size of the state space which these tools can handle depends on the hardware and software employed. However, it is too common to fail to handle the state space, even for the systems of a moderate size. This situation is referred to as *state space explosion*.

The size is not a major problem in the product form queuing networks because every queue is handled in isolation form the rest of the system. Thus, they are more scalable than Markov processes.

In the case of simulation, the complexity of coding and debugging is proportional to the size of the system and the desired level of accuracy. Moreover, running large simulation programs requires powerful computer systems.

7.3.4 Accuracy

Since the assumptions in the Markov-based tools are compatible with the system they model, it is rational to expect reasonable results. In queuing networks the assumptions are more severe and more unrealistic. Nevertheless, queuing networks typically provide a good trade-off between cost and performance measures they offer.

Since the simulation programs are run rather than solved, performance measures are observed rather than derived. Simulation programs offer a great deal of freedom in how measures can be defined. However, several observations with a variety of parameters should be carried out to improve the accuracy.

7.4 Conclusion

In this chapter we presented the most widely used tools and techniques adopted to evaluate the performance of computer systems and networks. The investigation of the strength and weakness of the evaluation techniques has shown that queuing networks and the simulation programs are more suitable candidates in the NoC realm. The rest of the thesis presents the analytical models along with the simulation programs to evaluate the performance of interconnection networks exchanging a variety of communication types including unicast, multicast and QoS-aware traffic. The rational behind presenting the models is mainly to show the strength and accuracy of the analytical models as a cost-effective performance evaluation tool.

Chapter 8

Modeling Unicast Communication

Unicast is the most common type of on-chip communication. In unicast communication a process sends a message to another process. The communicating processes may both reside at the same node, or they might be at different nodes. In this work we are more concerned and interested in performance of the communication architecture. Therefore, we assume that the communicating processes are at different nodes.

Due to the dominant nature of unicast communication on a chip, designers tend to employ architectures offering high performance unicast delivery. Simulation programs are the dominant tools to evaluate the performance measures (such as message latency) of the NoC architectures [74, 111]. Since NoCs are in concept similar to interconnection network for parallel computers with multiple processors, the analytical modeling techniques from the latter can be readily applied to the former. Guz et. al. [112] presented an analytical model to compute the network delay in application-specific wormhole-routed NoCs. They model the links as $M/D/1$ queues and their approximation does not capture inter-link dependencies. Therefore, as they have mentioned, the model is generally too optimistic for medium and high loads.

This chapter presents an analytical model to compute average message latency in wormhole-routed interconnection networks. The model is developed based on queuing theory and has been used to predict the unicast message latency in other interconnection networks [19, 113–115]. The reason we present the model is twofold. Firstly, to present a reliable unicast model for the Quarc NoC. And secondly, the unicast model will be used as a basis for more complicated analytical models to evaluate the performance of multicast

and QoS-aware traffic in the following chapters. The model is applied to the Quarc NoC and its predictions are validated by comparison against the results derived from a flit-level simulator developed using OMNET++ [96].

8.1 The basic assumptions of the analytical models

Developing analytical models of communication in interconnection networks requires defining a number of assumptions. These assumptions include, but are not limited to network topology, routing algorithm, switching technique and message generation functions. The rest of the thesis is dedicated to developing analytical models for various communication patterns in NoCs. The models are developed based on a number of common assumptions in addition to the instance-specific ones. The common assumptions that are shared by all analytical models in this thesis are widely used in the literature [19, 114, 116] and are defined as:

- The network adopts deterministic routing.
- The network employs wormhole switching.
- Local queues corresponding to each injection link in a given source node have infinite capacity.
- The messages are transferred to the local IP core as soon as they arrive at destinations.
- The messages are all the same size. The message size is larger than the network diameter.
- Unicast traffic is uniform.
- Propagation delay at each link, regardless of the link size, is one cycle.
- The latency of a unicast message is regarded as the time from generation of the unicast message at the source node to the time when the last flit of the message is absorbed by the sink at destination.
- The multicast message latency is the time from generation of the multicast message at the source node until the time when the last flit of the message is absorbed by the sink at the last destination of the multicast message.

- The broadcast message latency is the time from generation of the broadcast message at the source node until the time when the last flit of the message is absorbed by the sink at the last destination of the broadcast message.

8.2 The analysis method

The objective of this section is to introduce a model to evaluate the average message latency in interconnection networks employing wormhole switching. The model is developed on the following assumptions:

- The basic assumptions defined in Section 8.1.
- Nodes are generating unicast traffic independently of each other, following a *Poisson* process.
- The messages length is $\overline{msg}$ flits.

We view our network as a network of queues, where each link is modeled as an $M/G/1$ queue. For an $M/G/1$ queue the average waiting time is [117]

$$W_{M/G/1} = \frac{\lambda\rho}{2(1-\lambda x)}(1+\frac{\sigma^2}{x^2}) \tag{8.1}$$

$$\rho = \lambda x \tag{8.2}$$

where λ is the mean arrival rate, x is the mean service time and σ^2 is the variance of the service time distribution.

We define $\overline{x}_j$ as the service time at link ℓ_j. The service time is defined as the time when the link is reserved by the header flit to the time when the last flit of the message leaves the link. And, the waiting time, w_j, at link ℓ_j is the average time required for the header flit to be granted the access to the link.

During the journey towards a destination, a message passes through a number of links. Since in wormhole routing flits follow the header flit, the waiting time needs to be evaluated only for the header flit. Since $\overline{msg}$ is larger than the network diameter, the

latency may be defined as the total waiting and the service times experienced at the injection link plus a number of extra cycles that is equal to the hop distance from source to destination. Thus, for an arbitrary node, γ_j, in the network, the latency may be expressed as

$$\overline{L}_j = w_{S_j} + \overline{x}_{S_j} + \overline{D}_j - 1 \tag{8.3}$$

where w_{S_j} and $\overline{x}_{S_j}$ are the average waiting and the service time experienced at injection link. And $\overline{D}_j$ is the average hop distance between the source node, γ_j, and all destinations accessible from this node.

Averaging over all nodes yields the average message latency in network as

$$L = \frac{1}{N}\sum_j \overline{L}_j = \frac{1}{N}\sum_j (w_{S_j} + \overline{x}_{S_j} + \overline{D}_j) - 1. \tag{8.4}$$

As we modeled each link as an $M/G/1$ queue, w_{S_j} will be derived once the mean service time and its variance are known. Since in wormhole routing a message typically spans several links at each time, the service time at each link depends on the waiting time and the service time at its subsequent links. Therefore, to analyze the service time at each link, the waiting time and the service time at all possible successive links are required. Figure 8.1 can aid in analyzing the service time at an arbitrary link ℓ_i.

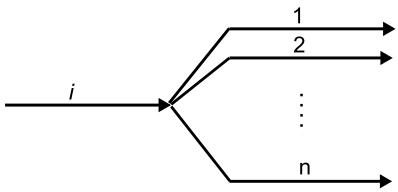


FIGURE 8.1: The service time at link ℓ_i depends on n successive links.

Figure 8.1 illustrates that traffic leaving link ℓ_i, may be injected to any of its n subsequent links, where n is a non-negative integer. We denote by $P_{i\rightarrow j}$ the probability that traffic enters link ℓ_j immediately after leaving link ℓ_i. The average service time experienced at

link ℓ_i therefore, can be expressed as

$$\overline{x}_i^{in} = \sum_j (w_j + \overline{x}_j + 1) P_{i \to j}. \tag{8.5}$$

This equation states that the service time at each link is derived once the waiting time and the service time of its successive links are known. The mean waiting time, w_j , may be approximated using an $M/G/1$ queuing model with the service time $\overline{x}_j$.

Draper and Ghosh [19] suggested that the variance of the service time at each link to be defined as

$$\sigma^2 = (x - \overline{msg})^2 \tag{8.6}$$

thus, the waiting time for an $M/G/1$ queuing system which is used for modeling wormhole routing may be approximated by

$$W_j = \frac{\lambda_j \overline{x}_j^2}{2(1 - \lambda_j \overline{x}_j)} (1 + \frac{(\overline{x}_j - \overline{msg})^2}{\overline{x}_j^2}). \tag{8.7}$$

The above waiting time, W_j, is the mean waiting time for a server in which messages arrive at rate λ_j and no message may block other messages. In wormhole-routed networks, however, a message may block other messages and therefore, W_j will not equal to w_j. In fact, when a message occupies a link, there is no competing traffic from this link to the subsequent links. Traffic at the link competes only with the incoming traffic from other links. This means that the mean w_j is less than the mean W_j. The effect of blocking is addressed by introducing a blocking probability which is defined as

$$P_{bl_{i \to j}} = 1 - \frac{\lambda_i^{in}}{\lambda_j} P_{i \to j} \tag{8.8}$$

where λ_i^{in} is the incoming traffic at ℓ_j from ℓ_i, λ_j is the total traffic rate at ℓ_j and $P_{i \to j}$ is the probability that ℓ_j is traversed immediately after leaving ℓ_i.

Finally, w_j may be obtained as the product of W_j and $P_{bl_{i \to j}}$

$$w_j = W_j . P_{bl_{i \to j}} \tag{8.9}$$

By combining Equations 8.5, 8.8 and 8.9 we obtain the service time at an intermediate link as

$$\overline{x}_i^{in} = \sum_j ((1 - \frac{\lambda_i^{in}}{\lambda_j} P_{i \to j}) W_j + \overline{x}_j + 1) P_{i \to j} \tag{8.10}$$

where, W_j is approximated using Equation 8.7 and $\overline{x}_j$ is the mean service time at link ℓ_j.

Given that the service time at the consumption link is $\overline{msg}$. Equation 8.10 can be adopted to compute the service time at all links from the consumption link at destination back to the injection link at the source node. Once the service time at the injection link is known, the average message latency experienced at the node and consequently the average latency in network is computed using Equations 8.3 and 8.4, respectively.

8.3 Traffic analysis in the Quarc NoC

Computing the average message latency in a network requires knowledge of traffic at all network links. We assume that the Quarc network adopts Across-First routing algorithm [30]. Traffic at each link is comprised of several incoming links and will be transmitted to a number of successive links. This section presents the traffic rate at each link of a Quarc NoC. As traffic distribution is slightly different, depending on whether the number of nodes is a factor of four ($N = 4x$) or only a factor of two ($N = 4x + 2$), we present them separately when required.

We denote by λ_g the traffic rate from a node to each individual destination. Depending on the destination address a message may take any of the four injection links. The traffic rate at injection links for traffic heading to right surrounding link, left surrounding link, across-right link and across-left link are denoted by $\lambda_{S_{right}}$, $\lambda_{S_{left}}$, $\lambda_{S_{\times right}}$ and $\lambda_{S_{\times left}}$, respectively, and equal to

$$\lambda_{S_{right}} = \lambda_{S_{left}} = \left\lceil \frac{N}{4} \right\rceil \lambda_g \tag{8.11}$$

$$\lambda_{S_{\times right}} = \left\lfloor \frac{N}{4} \right\rfloor \lambda_g \tag{8.12}$$

$$\lambda_{S_{\times left}} = (\left\lfloor \frac{N}{4} \right\rfloor - 1)\lambda_g. \tag{8.13}$$

The traffic rate at each surrounding link, λ_A, is comprised of traffic from three sources, *i*) across-network link, *ii*) previous link and *iii*) injection link. Traffic rates at each surrounding link is

$$\lambda_A = \begin{cases} \lceil \frac{N}{4} \rceil^2 \lambda_g & N = 4x \\ (\lfloor \frac{N}{4} \rfloor^2 + \lfloor \frac{N}{4} \rfloor + 1)\lambda_g & N = 4x + 2 \end{cases}. \tag{8.14}$$

Traffic rates at right, left and across-left ejection links are denoted by $\lambda_{e_{right}}$, $\lambda_{e_{left}}$ and $\lambda_{e_{\times left}}$, respectively, and equal to

$$\lambda_{e_{right}} = \lambda_{e_{left}} = (\frac{N}{2} - 1)\lambda_g \tag{8.15}$$

$$\lambda_{e_{\times left}} = \lambda_g \tag{8.16}$$

And finally, the traffic rate at right-cross network link, λ_{C_r}, and left-cross network link, λ_{C_l}, are equal to

$$\lambda_{C_r} = \lambda_{S_{\times right}} \tag{8.17}$$

$$\lambda_{C_l} = \lambda_{S_{\times left}} \tag{8.18}$$

8.4 Validation and analysis

To validate the unicast communication model, we apply the model to the Quarc NoC and compare the model prediction against the Quarc NoC simulator. The basic Quarc simulator presented in Section 5.2.2 is used to derive the unicast message latency. The simulator operates on the assumption defined in Section 5.2.1. Moreover, the unicast messages are generated at each node according to a *Poisson* process.

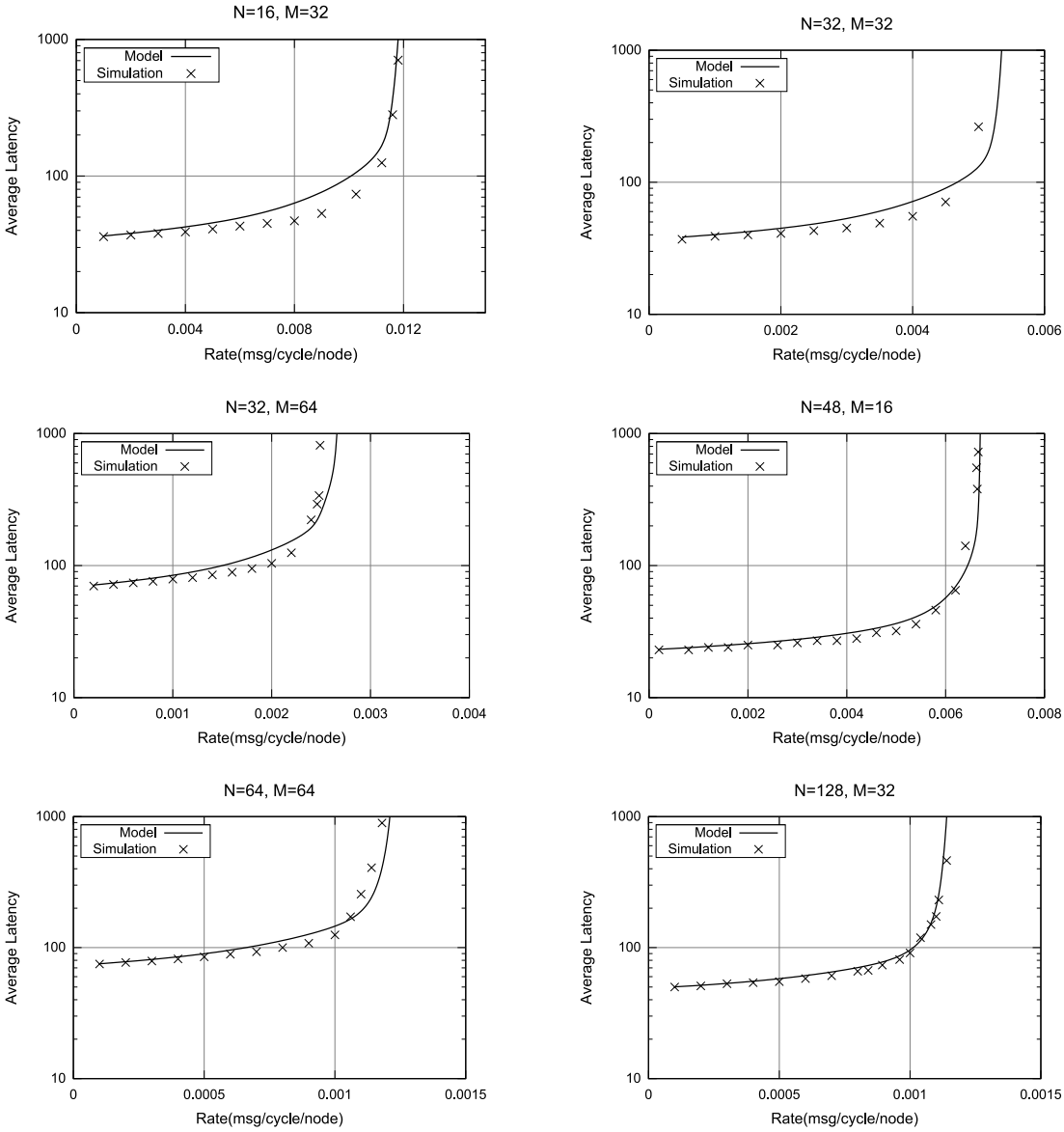


FIGURE 8.2: Analytical model against the simulation results.

The model is compared against the simulation results for numerous configurations by changing the message length and the network size. Figure 8.2 compare the simulation results against the analysis for networks of 16, 32, 64 and 128 nodes, while the message length is set to 16, 32, 48 and 64. Each simulation experiment is run according to the algorithm presented in Section 5.2.3.

In the graphs, N and M stand for the number of nodes and the message length, respectively. The horizontal axis of the graphs show the message rate per node per cycle, while, the vertical axis presents the average message latency. As can be seen from the the figures, the analytical model presents a good approximation of the network latency

in the presence of light to heavy traffic. In particular the figures reveal that the model predicts the network saturation points accurately.

8.5 Conclusion

The performance models of unicast communication have been widely reported in the literature. The chapter presented a model based on queuing theory concepts. The analytical model has been applied to the Quarc NoC and the predictions verified against the results produced by a flit-level simulator. The analytical model presented in this chapter will be employed in the following sections to model more complicated traffic. The next chapter uses the model to predict the average message latency of multicast traffic in all-port router architectures.

Chapter 9

A Performance Model of Multicast Communication

The literature has witnessed numerous analytical performance models of unicast traffic [19, 20] and analysis of unicast traffic in the presence of broadcast traffic [115] in parallel computers and the NoC domain. In [21] Shahrabi et al. introduced a model to compute the broadcast communication latency in Hypercube. However, in their system under model only unicast is wormhole-routed and broadcast communication is not wormhole-routed. Also, their model is developed for the architectures adopting *one-port* router scheme. This section presents a novel analytical model to compute the average multicast communication latency in a system adopting wormhole-routing for both unicast and multicast communication.

In an interconnection network employing a multi-port router scheme, the multicast latency is defined as the time from generation of the message at source until the time when the last flit of the multicast message is absorbed by the last destination of the multicast message among messages leaving m injection ports. The novelty of the approach introduced in this section lies in its ability to predict the average latency of multicast in networks adopting all-port or multi-port router architecture; in which there is no synchronization between messages emerging from each port. Otherwise, the model could be classified as a variation of the available analytical models of unicast communication.

9.1 An analytical model of multicast communication

The model adopts the following assumptions:

- The basic assumptions defined in Section 8.1.
- There are two types of messages in the network: multicast and unicast. A multicast message is delivered to all destinations identified in the header flit(s). A unicast message is sent to other nodes in the network with equal probability. When a message is generated at a source node, it has a finite probability $0 \leq \alpha \leq 1$ of being multicast and probability $(1 - \alpha)$ of being unicast.
- Nodes generate traffic independently of each other, according to a *Poisson* process with a mean rate of λ_g message/cycle.
- The messages length is $\overline{msg}$ flits.

In a multi-port router architecture employing deterministic routing, depending on the position of the destination node, the appropriate injection link should be taken to transmit the message through. We define S_{j_c} as the subset of the network nodes receiving the multi-destination message sent from node γ_j through injection port Φ_{j_c},

$$S_{j_c} = \{\gamma_i : (\gamma_i \in \mathscr{M}_j)\, and\, (\Re_{j \to i}\, passes\, through\ port\, \Phi_{j_c})\} \tag{9.1}$$

where

- $1 \leq c \leq m$, denotes the index of injection port Φ_{j_c} at node γ_j,
- $\mathscr{M}_j$ denotes a set of multicast destinations to be reached from node γ_j,
- $\Re_{j \to i}$ denotes a set of links whose member determine the path from node γ_j to node γ_i and,
- The multicast destination set associated with a port is disjoint from the multicast destination sets associated with other ports, i.e.

$$\bigcap_{c=1}^{m} S_{j_c} = \varnothing. \tag{9.2}$$

To multicast a message, it should be sent to disjoint sub-networks through one or more ports of the router. It can be argued that the message latency experienced by the largest sub-network can be regarded as the multicast communication latency. Although this justification sounds reasonable in most situations, the dynamic behavior of traffic in different sub-networks may easily lead to situations where, smaller sub-networks deliver the message later than larger sub-networks do. Therefore, it is desirable to find a more reliable solution based on the latencies experienced at each sub-network connected to different ports.

By adopting the analytical model explained in Chapter 8, this section presents an approach to compute the mean multicast message latency in a wormhole-routed interconnection network employing asynchronous all-port routers. It is important to note that, there is no form of synchronization between flit streams leaving different ports of a router. This means that each injection port of the router transmits the multicast messages independently of the other injection ports.

The communication latency experienced by a message is a factor of three components, namely, the message length, the number of hops and the total waiting time at intermediate links. Among those parameters, the message length and the number of hops are fixed, while the waiting time varies. The total waiting times at all intermediate links from source to destination may be any non-negative real time number. Nevertheless, its average is the total of the waiting times which may be computed using the method explained in Chapter 8.

Using the above definition, the average latency of unicast traffic at injection port Φ_{j_c} at node γ_j may be expressed as

$$\overline{L}_{j_c} = \sum_{l} w_l + \overline{msg} + \psi_{j_c} - 1 \tag{9.3}$$

where

- $\overline{L}_{j_c}$ is the average communication latency experienced by a message leaving injection port Φ_{j_c} at node γ_j.
- w_l is the waiting time experienced by the header flit of a message before being granted access to link $\ell \in \Re_{j \rightarrow dst}$ (dst denotes destination). w_l can be computed using the approach presented in Section 8.2.

- ψ_{j_c} denotes the number of hops traversed by a message in sub-network S_{j_c} originating from node γ_j.

The total waiting times at all intermediate links from source to destination, $\sum_\ell w_\ell$, may be any non-negative real time number. This average is the total of the waiting times which may be computed using the method explained in Section 8.2. Therefore, for each individual injection port, Φ_{j_c} $(1 \leq c \leq m)$, of an all-port router at node γ_j, we are able to define an exponential distribution, $E_{1,\mu_{j_c}}$, which its expected time is the total waiting times (from source to destination) experienced by the header flit of the message leaving injection port Φ_{j_c} at node γ_j.

Using the above definition, μ_{j_c} is expressed as:

$$\mu_{j_c} = \frac{1}{\sum_l w_l} \quad 1 \leq c \leq m,\ 1 \leq j \leq N. \tag{9.4}$$

By associating the waiting times at each port of the routers to independent exponentially distributed random variables and recalling the definition of the message latency, the multicast waiting time will be defined as the expected time for the occurrence of the last event among m independent exponentially distributed random variables. Which is of course, the expected total waiting time experienced by the last message delivered to its destination among m messages transmitted at injection ports of the router.

To compute the expected time of the last event, we use two properties of the exponential distributions.

- The exponential distributions are memory-less.
- The minimum of independent exponential distributions is exponentially distributed [110]. i.e.

$$P\left[min\left\{E_{1,\mu_1}, E_{1,\mu_2}\right\} > t\right] = e^{-(\mu_1+\mu_2)t}. \tag{9.5}$$

Using the above two properties, we first compute the expected time for the last event in case of only two independent exponentially distributed random variables, $E[max\left\{E_{1,\mu_1}, E_{1,\mu_2}\right\}]$,

and then generalize the method for $m \geq 2$ independent exponentially distributed random variables.

According to Equation 9.5 the expected time for occurrence of the first event between two independent exponential distributions is

$$E\left[min\left\{E_{1,\mu_1}, E_{1,\mu_2}\right\}\right] = \frac{1}{\mu_1 + \mu_2} \tag{9.6}$$

Due to the memory-less property of the exponential distributions, the expected time for the next event after the occurrence of the first event is $\frac{1}{\mu_2}$ or $\frac{1}{\mu_1}$ depending on whether the first event has been fired by E_{1,μ_1}or E_{1,μ_2}, respectively. The probability that E_{1,μ_1}or E_{1,μ_2} has been the first event, however, is related to μ_1 and μ_2. In other words, the probability that E_{1,μ_1}has been the first event is $P_{E_{1,\mu_1}} = \frac{\mu_1}{\mu_1+\mu_2}$ and the probability that E_{1,μ_2} has been the first event is $P_{E_{1,\mu_2}} = \frac{\mu_2}{\mu_1+\mu_2}$. Therefore, the expected time for the last event between two independent exponentially distributed random variables is

$$E\left[max\left\{E_{1,\mu_1}, E_{1,\mu_2}\right\}\right] = \frac{1}{\mu_1 + \mu_2} + P_{E_{1,\mu_1}} \times E[E_{1,\mu_2}] + P_{E_{1,\mu_2}} \times E[E_{1,\mu_1}]. \tag{9.7}$$

Generalizing the solution for $m \geq 2$ yields the expected time for occurrence of the last event between m independent exponentially distributed events as

$$\begin{aligned} E\left[max\left\{E_{1,\mu_1}, E_{1,\mu_2}, ..., E_{1,\mu_m}\right\}\right] = & \frac{1}{\mu_1 + \mu_2 + ... + \mu_m} + \\ & \frac{\mu_1}{\mu_1 + \mu_2 + ... + \mu_m} E\left[max\left\{E_{1,\mu_2}, E_{1,\mu_3}, ..., E_{1,\mu_m}\right\}\right] + \\ & \frac{\mu_2}{\mu_1 + \mu_2 + ... + \mu_m} E\left[max\left\{E_{1,\mu_1}, E_{1,\mu_3}, ..., E_{1,\mu_m}\right\}\right] + ... \\ & \frac{\mu_m}{\mu_1 + \mu_2 + ... + \mu_m} E\left[max\left\{E_{1,\mu_1}, E_{1,\mu_2}, ..., E_{1,\mu_{m-1}}\right\}\right]. \end{aligned} \tag{9.8}$$

Adopting the above analysis, the expected waiting time experienced by the last message (among m independent branches of a multicast message leaving node γ_j) delivered to its destinations, W_j, may be computed as

$$W_j = E\left[max\left\{E_{1,\mu_{j,1}}, E_{1,\mu_{j,2}}, ..., E_{1,\mu_{j,m}}\right\}\right]. \tag{9.9}$$

Therefore, the average multicast latency at node γ_j, $\overline{L}_j$, may be expressed as

$$\overline{L}_j = W_j + \overline{msg} + \psi_j - 1 \tag{9.10}$$

where

$$\psi_j = Max(\psi_{j_c}) \qquad 1 \leq c \leq m \tag{9.11}$$

is the maximum hops traversed among m sub-networks connected to node γ_j.

Averaging over all nodes in the network yields the average multicast message latency as

$$\overline{L} = \frac{1}{N} \sum_{j}^{N} \overline{L}_j. \tag{9.12}$$

9.2 Traffic analysis

To evaluate the service and the waiting time at each link, the detailed traffic information at the link is required. This section presents traffic at each link in the Quarc NoC adopting the assumptions defined in Section 9.1. Moreover, we assume that the Quarc NoC employs Across-First unicast routing algorithm and adopts the multicast routing algorithm presented in Section 3.3.4.

The Quarc NoC employs an all-port router scheme, depending on the multicast destinations set one or more injection links are used to transmit the message to destinations at each quadrant. After leaving the injection link, a multi-destination message at each quadrant travels towards the address specified in the destination address of the header flit. On the way to destination, it is absorbed by multicast destinations specified by bit-string in the header flit. In analysis of network traffic, as will be shown, the final destination at each rim affects traffic in the network. We denote by ψ_r, ψ_l, ψ_{cr} and ψ_{cl} the relative position of the last multicast destination at right, left, across-right and across-left rims, respectively, to a node.

We denote by λ_g the total traffic rate generated at each node. This traffic consists of unicast and multicast traffic. It is assumed that α, $0 \leq \alpha \leq 1$, represents the rate of multicast generation at each node. Therefore, the total traffic at each node may be

represented as

$$\lambda_g = (1-\alpha)\lambda_g + \alpha\lambda_g. \tag{9.13}$$

To simplify traffic analysis, we define λ_m, the average rate of multicast traffic at each node, as

$$\lambda_m = \alpha\lambda_g \tag{9.14}$$

and λ_u, the average unicast traffic rate to each individual node in the network, as

$$\lambda_u = \frac{(1-\alpha)\lambda_g}{N-1}. \tag{9.15}$$

Applying λ_u to equations presented in Section 8.3 yields unicast traffic at each link. The rate of multicast traffic at right, left, across-right and across-left injection links are denoted by $\lambda_{S_{m_r}}$, $\lambda_{S_{m_l}}$, $\lambda_{S_{m_{\times r}}}$ and $\lambda_{S_{m_{\times l}}}$, respectively. The traffic rate at those links are zero if their corresponding rim does not include any multicast destinations. Otherwise, they equal to λ_m. The rate at ejection links depends on the number of sources targeting a node as a multicast destination.

The multicast traffic rate at each surrounding link at right and left to a node are denoted by $\lambda_{A_{m_r}}$ and $\lambda_{A_{m_l}}$, respectively, and equal to

$$\lambda_{A_{m_r}} = \begin{cases} (\psi_r + \psi_{cl} - 1)\lambda_m & \psi_{cl} > 0 \\ \psi_r.\lambda_m & \psi_{cl} = 0 \end{cases} \tag{9.16}$$

and

$$\lambda_{A_{m_l}} = \begin{cases} (\psi_l + \psi_{cr} - 1)\lambda_m & \psi_{cr} > 0 \\ \psi_l.\lambda_m & \psi_{cr} = 0 \end{cases}. \tag{9.17}$$

And finally, the multicast traffic rate at each across-right and across-left links are denoted by $\lambda_{C_{m_r}}$ and $\lambda_{C_{m_l}}$, respectively, and equal to

$$\lambda_{C_{m_r}} = \begin{cases} \lambda_m & \psi_{cl} > 0 \\ 0 & \psi_{cl} = 0 \end{cases} \tag{9.18}$$

and

$$\lambda_{C_{m_l}} = \begin{cases} \lambda_m & \psi_{cr} > 0 \\ 0 & \psi_{cr} = 0 \end{cases}. \tag{9.19}$$

9.3 Validation

To validate the analysis we apply the model to the Quarc NoC and compare its predictions against the results derived from the basic Quarc simulator presented in Section 5.2.2. The simulator operates on the basic assumption defined in Section 5.2.1. Moreover, we assume that source produces the messages according to a *Poisson* distribution. A message may be unicast or multicast. Multicast destinations are selected randomly at the beginning of the simulation run and remain fixed during the experiment.

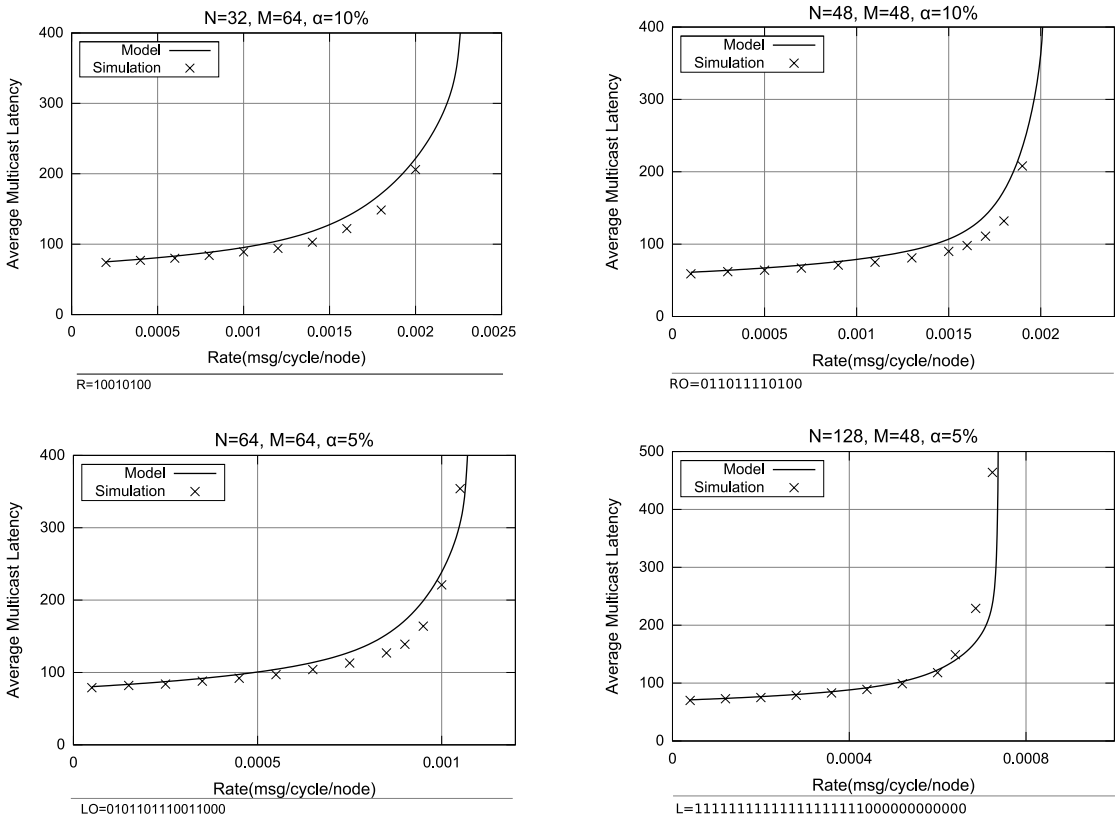


FIGURE 9.1: Model validation for localized multicast destinations.

The model predictions are compared against the simulation results for numerous configurations by changing the Quarc network size, the message length and the rate of multicast traffic. The model is validated against the simulation results for three different scenarios, *i*) multicast destination are at local proximity of each other, *ii*) unordered multicast destinations and *iii*) broadcast.

Figures 9.1, 9.2 and 9.3 compare the simulation results against the analysis for networks having 16, 32, 48, 64 or 128 nodes. The message length can be 16, 32, 48 or 64 flits. Multicast traffic comprises 3%, 5% or 10% of the overall traffic. Each simulation experiment is run according to the algorithm presented in Section 5.2.3.

In the graphs, N, M and α represent the number of nodes in the Quarc NoC, the message length and the rate of multicast traffic, respectively. L, R, LO and RO denote the *bit-strings* corresponding to multicast destinations at left, right, across-left and across-right of the node, respectively. The horizontal axis in the figures shows the message rate, while the vertical axis describes the latency.

9.3.1 Localized multicast

In the Quarc NoC placing multicast destinations at one rim can offer the most efficient approach to multicast a message. In such a scenario, multicast can be viewed as a unicast communication with the target destination being the last node to be visited. The model has been verified to predict the multicast message latency when destinations are in spatial locality of each other and can be reached by only one rim. The graphs in Figure 9.1 represent such scenarios that the destination nodes are at one single rim relative to the source node.

9.3.2 Multicast

The graphs in Figure 9.2 show the configurations in which multicast destinations are selected randomly. This is the most common type of multicast, where destinations are situated in different locations in the network.

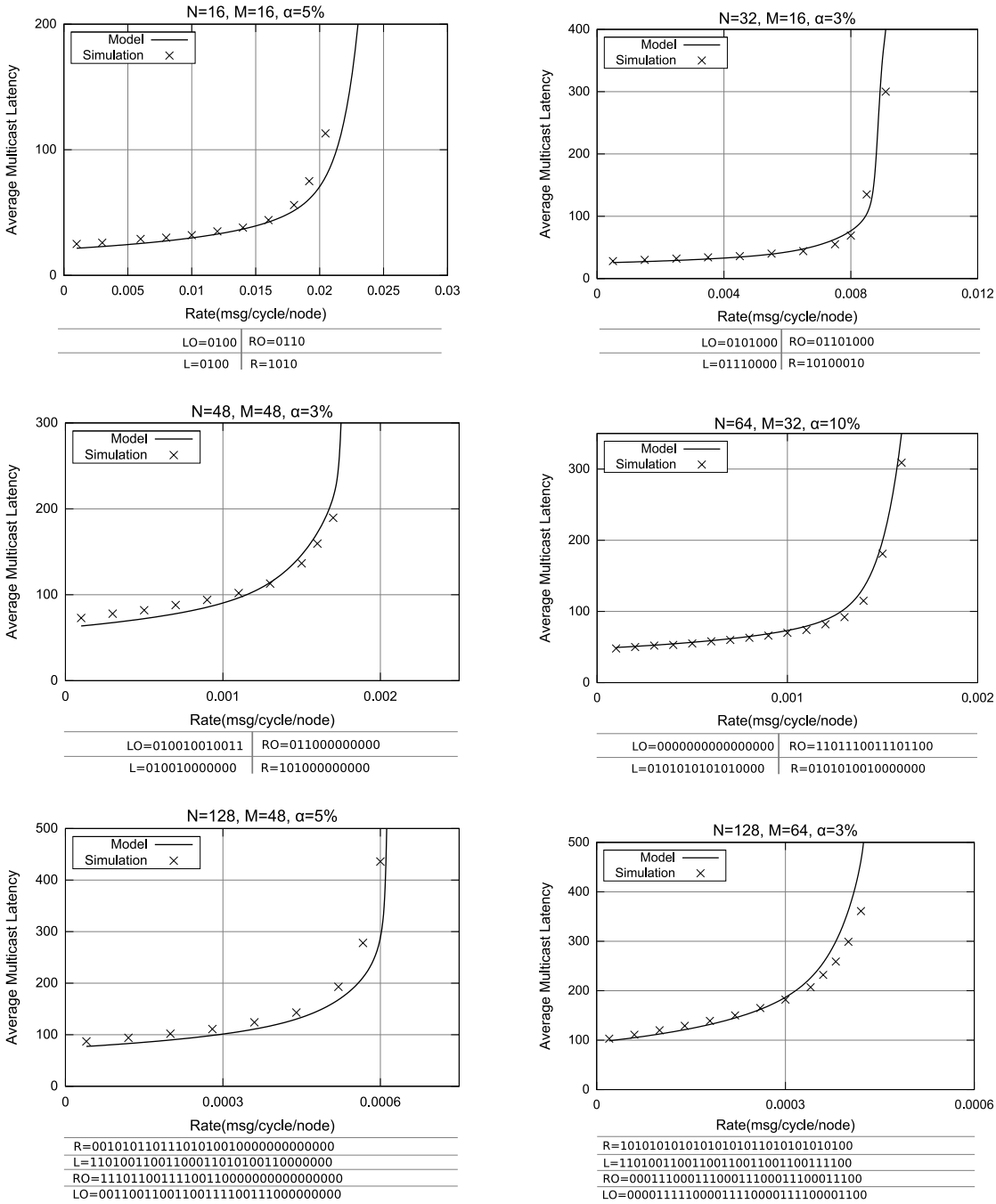


FIGURE 9.2: Model validation for random multicast destinations.

9.3.3 Broadcast

Broadcast, in which a node sends a message to all other nodes in the network, is considered as the most fundamental collective communication operation. To validate the model for the extreme case, we present the comparison of the model predictions against the simulation results for broadcast traffic. The comparisons are demonstrated in Figure

9.3. In contrast to multicast communication, broadcast communication does not require setting the bit-string in the header flit.

As can be seen from the figures, the analytical model presents a good approximation of the average latency for multicast and broadcast in a wide range of configuration settings.

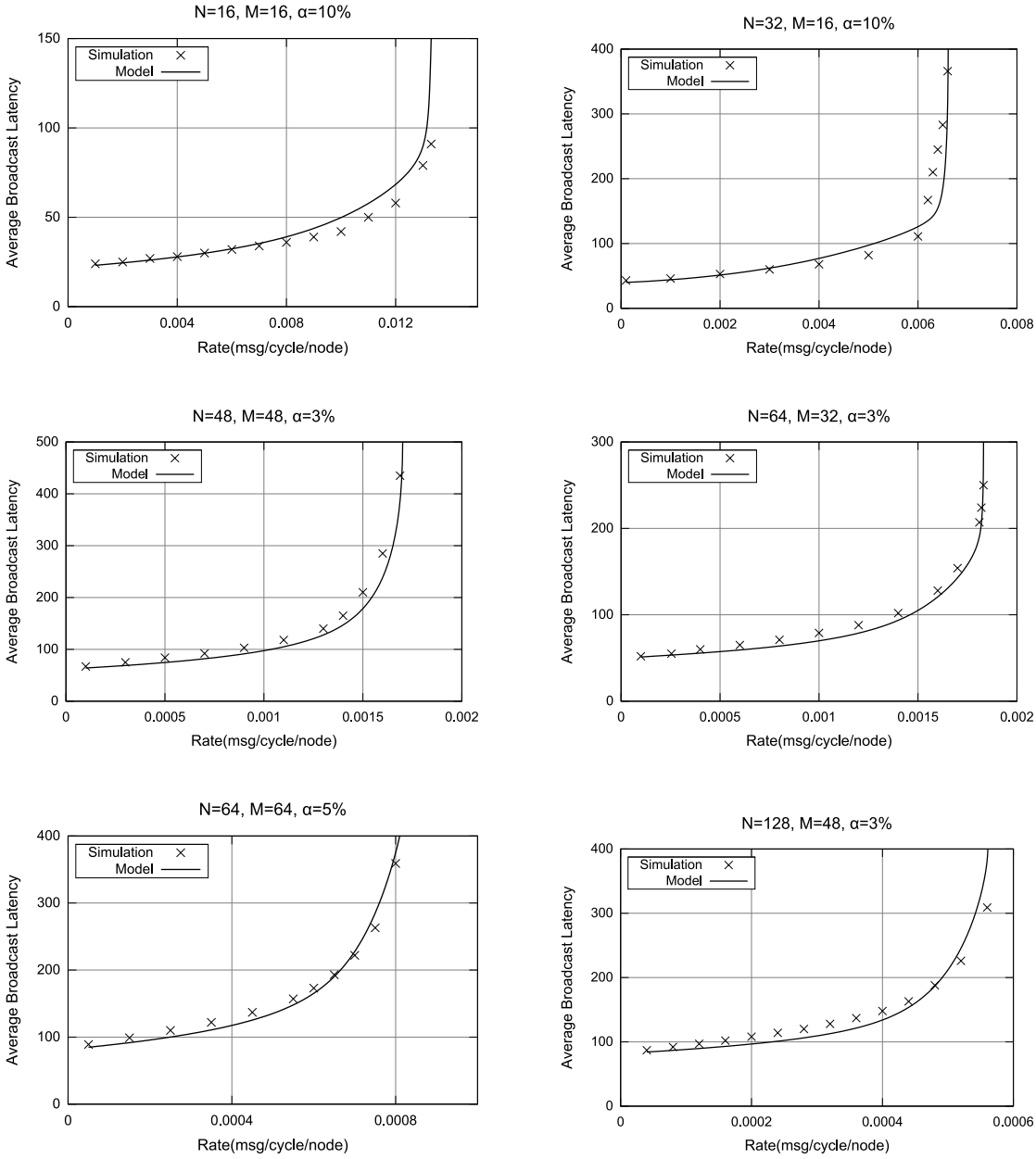


FIGURE 9.3: Model validation for broadcast traffic.

9.4 Conclusion

In this chapter we presented a new analytical model to predict the message latency of multicast communication in all-port wormhole-routed interconnection networks. The model is developed using queuing theory and stochastic properties of the exponentially distributed random variables. The model evaluates the average message latency of multicast, while network exchanges both unicast and multicast traffic. The model has been applied to the Quarc NoC in a wide range of configuration settings. The comparison between the model predictions and the results derived from a flit level simulator has verified the accuracy of the model.

Chapter 10

Modeling Differentiated Services-based Quality of Service in Wormhole-routed NoCs

Meeting a minimum quality of service (QoS) is necessary for a wide range of performance sensitive applications and system services to perform their intended tasks successfully. The performance demands of the application are typically expressed as the need for a certain bandwidth, a maximum message latency or bounds on jitter. The NoC architectures may offer QoS as best-effort service, guaranteed service or a combination of both. This chapter presents an overview of a number of architectures offering QoS support in the NoC paradigm. The chapter follows by introducing a performance model of differentiated services-based QoS traffic in wormhole-routed interconnection networks.

10.1 The QoS support in the NoC domain

In the NoC domain QoS is implemented as best-effort service and/or guaranteed service. This section presents a brief description of each approach and introduces a number of NoC architectures implementing QoS support.

10.1.1 Best-effort service

Employing best-effort services is a widely used approach to support QoS. In this method QoS is achieved by relatively prioritizing traffic in networks employing connection-less communication mechanisms. For each traffic class, the packets are treated differently at the routers (Per-Hop-Behavior) to e.g. minimize end-to-end delay, jitter or packet loss. Best-effort services typically utilize the network resources more efficiently as they do not require resource reservation. However, the major drawback of best-effort QoS is failure to offer predictability or performance guarantees.

Bolotin et al. [9] , Rostislav et al. [23] and Beigne et al. [22] exploited best-effort service in NoCs by designing the routers that support QoS based on differentiated services.

QNoC [9] introduces a design process that satisfies the QoS requirements of an on-chip application at low cost. In the architecture, traffic may belong to four classes of service namely, *signaling*, *real-time*, *read/write* and *block-transfer*. Traffic with higher priority win the competition on using the resources.

Rostislav et al. [23] designed synchronous and asynchronous routers for both single and multiple service levels and compared their performance. The routers can support prioritized best-effort service. They showed both synchronous and asynchronous routers provide almost similar throughput. Nevertheless, asynchronous routers require less resources to implement.

Beigne et al. [22] proposed a complete asynchronous NoC architecture which integrates QoS. Each physical link is shared by two virtual channels. The first one which is dedicated to real-time, low latency traffic has higher priority and can suspend the streams on the other virtual channel. While the other one is employed to exchange best-effort traffic.

10.1.2 Guaranteed service

Employing best-effort service is not appropriate in the systems demanding stringent QoS requirements. Unlike best-effort service, implementing guaranteed service can support predictability and guarantee in the performance. Reserving the resources to guarantee a level of service may be performed at design time or dynamically at run-time.

Most architecture designated to support guaranteed service are offering best-effort QoS to utilize the available bandwidth more efficiently. Implementing a hybrid of best-effort/guaranteed QoS was first pointed out by Rijpkem et al. [24].

The ï¿½thereal NoC [14] supports both best-effort and guaranteed QoS. Support for real-time communication is achieved by providing throughput, latency and jitter guarantees. In ï¿½thereal, this is implemented by configuring connections as pipelined time-division multiplexed circuits over network. Time multiplexing is only possible when the network routers have a notion of synchronicity which allows slots to be reserved consecutively in a sequence of routers [8, 67]. Throughput guarantees are given by the number of slot allocated for a connection. A slot corresponds to a given bandwidth B_i and, therefore, reserving N slots for a connection results in a total bandwidth of $N \times B_i$. The latency bound is given by the waiting time until the reserved slot arrives and the number of routers between source and destination. Since all network routers implement a common notion of time in a slot counter, link contention for GT packets is completely avoided by controlling the time they enter the network using a local slot allocation table. Notice that slot allocation can be performed statically during initialization. In recent versions, in order to save area, slot tables have been removed, while this information is provided in the GT packet header. BE traffic is scheduled to non-reserved, and reserved but unused slots with a non optimal algorithm based on parallel iterative matching. The logically separated guaranteed (GT) and best-effort (BE) routers are combined to share link and data path resources.

In the Nostrum architecture [51], a service of guaranteed bandwidth and latency has been implemented in addition to the existing service of best-effort packet delivery. The guaranteed performance is offered via virtual circuits. Virtual circuits are set up semi-statically in which the route is decided at design time, but the bandwidth is variable at run-time.

MANGO [70] offers connection-oriented guaranteed service as well as connection-less best-effort service. MANGO adopts virtual circuits to support guaranteed-services.

Sathe et al.[25] presented an architecture to provide best-effort and soft guaranteed services. In their approach, using best-effort delivery a circuit is established between source and destination and is canceled once the communication is over. Although their approach may utilize resources more efficiently as they are not statistically allocated,

the overhead to establish and cancel the connections may not be acceptable in some applications.

Liang et al. [26] introduced aSoC, a modular communication architecture for on-chip network interconnects. In their architecture communication between two nodes takes place via a point-to-point pipelined connections which can be established statically at design time or dynamically at run-time.

In the architectures exploiting a hybrid of best-effort and guaranteed QoS, the resources for guaranteed service must be reserved for the worst case scenarios. In such architectures the reserved resources may not be utilized efficiently in periods when there is no communication takes place. Andreasson et. al. [28] proposed a scheme to more efficiently use the slack-time not used by guaranteed service for improving the quality of best-effort traffic.

10.2 Communication modeling of differentiated services-based QoS

The above review of the QoS support in NoCs clearly shows that best-effort service is implemented in majority of NoCs as the main approach to support QoS or as an additional service to guaranteed service. Due to ubiquitous nature of best-effort service in the NoC architectures offering QoS, this section introduces a model to evaluate the average message latency in wormhole-routed interconnection networks offering differentiated services-based QoS. The concept of differentiated services (of which the DiffServ specification [130] is the best known example) is that traffic is divided into classes which have a different priority. In the simplest case, there are two categories of traffic, e.g. high priority and best effort.

The model is developed on the following assumptions:

- The basic assumptions defined in Section 8.1.
- Nodes are generating the messages independently and according to a *Poisson* process.

- Upon creation of a message, it is assigned a class of service c_p $(1 \leq p \leq P)$. Where a message from class c_{p+1} has priority over a message belonging to class c_p.
- The message length is $\overline{msg}$ flits.
- The network supports differentiated services-based QoS.

The network is modeled as a queuing network, where each link is modeled as a priority $M/G/1$ queue. For a priority $M/G/1$ queue with customers belonging to P classes of service the average waiting time is [131]

$$W_p = \frac{W_0}{(1-\sigma_p)(1-\sigma_{p+1})} \tag{10.1}$$

$$W_0 = \sum_{i=1}^{P} \frac{\lambda_i \overline{x_i^2}}{2\overline{x_i}} \tag{10.2}$$

$$\sigma_p = \sum_{i=p}^{P} \rho_i \tag{10.3}$$

$$\rho = \lambda x \tag{10.4}$$

where λ_i, x_i and $\overline{x_i^2}$ are the arrival rate, the service time and the second moment of the service time distribution, respectively, for customers belonging to class c_i. In the model, $M/G/1$ queues are non-preemptive and any arriving customer, regardless of its priority, is not allowed to interrupt the execution of the current customer. The average time for the execution of the current customer is called *residual time* and is denoted by W_0.

For an arbitrary node γ_j in the network, the latency experienced by a message belonging to class c_p may be expressed as

$$\overline{L}_{j_p} = w_{S_{j_p}} + \overline{x}_{S_{j_p}} + \overline{D}_j - 1 \tag{10.5}$$

where $w_{S_{j_p}}$ and $\overline{x}_{S_{j_p}}$ are the average waiting and the service time at injection link for the messages belonging to class c_p and $\overline{D}_j$ is the average distance in terms of the number of links traversed by the messages leaving node γ_j.

Averaging over all nodes yields the average latency for class c_p traffic in the network as

$$L_p = \frac{1}{N}\sum_j \overline{L}_{j_p} = \frac{1}{N}\sum_j (w_{S_{j_p}} + \overline{x}_{S_{j_p}} + \overline{D}_j) - 1 \tag{10.6}$$

As we modeled each link as an $M/G/1$ queue, $w_{S_{j_p}}$ will be derived once the mean service time and its variance are known.

Since in wormhole-routed networks a message typically spans several links at each time, the service time at each link depends on the waiting and the service time of its subsequent links. Therefore, to analyze the service time at each link the waiting time and the service time at all possible successive links are required. i.e.

$$\overline{x}_{i_p}^{in} = \sum_j (w_{j_p} + \overline{x}_{j_p} + 1) P_{i \to j} \tag{10.7}$$

where $\overline{x}_{j_p}$ and $\overline{w}_{j_p}$ represent the service time and the waiting time at link ℓ_j, respectively. And $P_{i \to j}$ denotes the probability that traffic enters link ℓ_j after leaving link ℓ_i.

To compute the average waiting time at a priority $M/G/1$ queue, the second moment of the service time is required. The second moment of the service time can be derived, once the variance of the service time is known. We define the variance as

$$Var = (\overline{x}_j - \overline{x}_{j-1})^2 \tag{10.8}$$

where link ℓ_{j-1} is taken immediately after leaving link ℓ_j.

The variance defined above may be adopted to compute the second moment of the service time which consequently can be used to derive W_0 and W_{j_p} using Equations 10.1 and 10.2 as

$$W_{j_p} = \frac{W_0}{(1-\sigma_p)(1-\sigma_{p+1})} \tag{10.9}$$

$$W_0 = \sum_{i=1}^{P} \frac{\lambda_{i_p} \overline{x_{i_p}^2}}{2\overline{x}_{i_p}}. \tag{10.10}$$

As explained in Section 8.3, in wormhole-routed networks, once a link is allocated to a message, flits of other messages are unable to use the link. This feature reduces the competing traffic for accessing the subsequent links. In other words, W_{j_p} and w_{j_p} are not equal. The situation is addressed by introducing a blocking probability as

$$P_{bl} i \rightarrow j = 1 - \frac{\lambda_i^{in}}{\lambda_j} P_{i \rightarrow j} \tag{10.11}$$

where λ_i^{in} is the incoming traffic at link ℓ_j from link ℓ_i, λ_j is the total traffic rate at link ℓ_j and $P_{i \rightarrow j}$ denotes the probability that link ℓ_j is traversed after leaving link ℓ_i.

Finally, w_{j_p} is obtained as the product of W_{j_p} and $P_{bl_{i \rightarrow j}}$ as

$$w_{j_p} = W_{j_p} . P_{bl_{i \rightarrow j}}. \tag{10.12}$$

By combining Equations 10.7 and 10.11 we obtain the service time at an intermediate link as

$$\overline{x}_{i_p}^{in} = \sum_{j} ((1 - \frac{\lambda_i^{in}}{\lambda_j} P_{i \rightarrow j}) W_{j_p} + \overline{x}_{j_p}) P_{i \rightarrow j} \tag{10.13}$$

where W_{j_p} is approximated by Equation 10.9 using $\overline{x}_{j_p}$ as the mean service time.

Given that the service time at ejection link is $\overline{msg}$ cycles, using Equation 10.13, the service time at all links from the ejection link at destination back to the injection link at source can be derived. Applying the service time at the injection link to Equation 10.5 and averaging over all nodes will yield the average latency experienced by each class of traffic in the network.

10.3 Validation

In this section we validate the analytical model by comparing its prediction against the results derived from a simulator. The basic version of the Quarc NoC presented in

Part II of the thesis does not support QoS. To verify the model, we have developed a discrete event simulator of the Quarc NoC supporting differentiated services-based QoS. The simulator which is developed using OMNET++ [96] improves on the basic Quarc simulator (Section 5.2.2) by modifying the passive queue and the router. The schematic of the components in each node is shown in Figure 10.1. The passive queue has eight queues to store the messages belonging to high and normal traffic corresponding to each quadrant. The passive queue sends the messages based on their priority and creation time. Should the high priority queue be not empty, the messages in the high priority queues are served according to a FIFO policy. Otherwise, the messages in the normal priority queue are served as a FIFO queue.

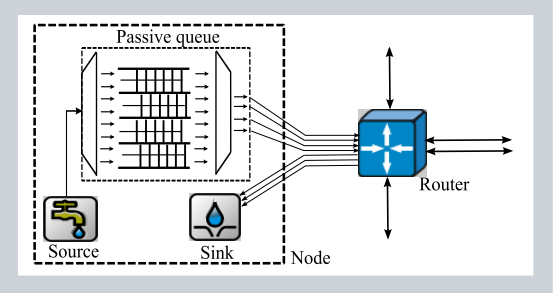


FIGURE 10.1: The components of a sample node in the QoS-aware Quarc architecture.

Similar to the assumptions defined for the model, resources are non-preemptive. While servicing a message, if other messages try to receive service, the routers record their information. After the last flit of the current message leaves the router, the router investigates the messages waiting on the recently released resource. Should the waiting messages have different priorities, the message with higher priority wins the competition and the corresponding router or passive queue is notified to start transmission. In situation where the messages have equal priority, the messages are served according to a FIFO policy. The routers serve the packets according to their relative priority. Packets with equal priorities are served as FIFO.

The simulator operates on the assumption defined in Section 5.2.1. Moreover, we assume that source produces the messages according to a *Poisson* distribution. A message generated at the source node has a finite probability $0 \leq \alpha \leq 1$ of being a high priority message and probability $(1 - \alpha)$ of being normal priority.

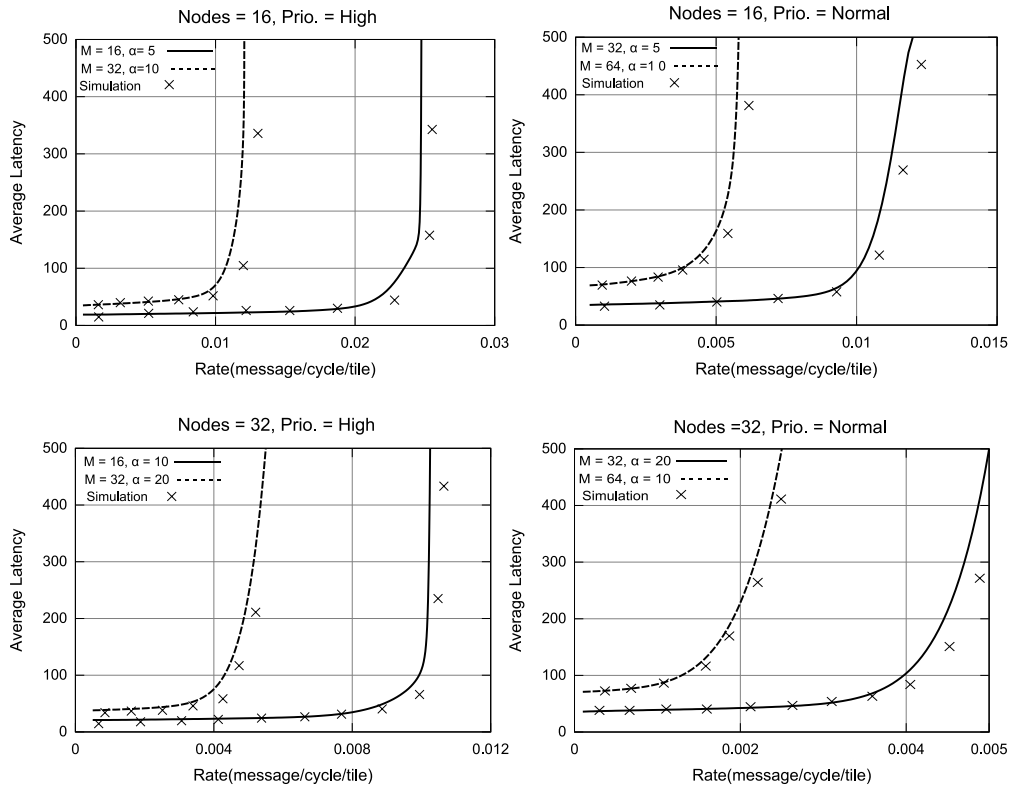


FIGURE 10.2: Validation of the model against the simulation results for networks of 16 and 32 nodes.

Traffic at each link in the model equals to traffic presented in Section 8.3. The high and normal traffic rate at each link is proportional to the rate of the high priority generated at source nodes. The model is compared against the simulation results for numerous configurations by changing the network size, the message length and the rate of high priority traffic. Figures 10.2 and 10.3 compare the simulation results against the analysis for networks having 16, 32, 64 or 128 nodes. The message length can be 16, 32 or 64 flits. And the high priority traffic, α, may comprise 5%, 10% or 20% of the overall traffic. Each simulation experiment runs according to the algorithm presented in Section 5.2.3.

In graphs M and α represent the message length and the rate of high priority traffic, respectively. The priority may be high or normal, which shows whether the graph presents the comparison for privileged or normal traffic. The horizontal axis in the figures show the message per cycle per node. While, the vertical axis describes the average message latency. As can be seen from the figures, the analytical model presents a good approximation of the network latency in a wide range of configurations.

The analysis also reveals that increasing the high priority traffic rate, α, adversely affects

the message latency of high priority traffic. That is apparently because increasing the rate of high priority traffic raises the competition between the high priority messages, which in turn reduces the effect of traffic prioritization.

Our findings also suggests that regardless of the size of network, as long as traffic load is not high, there is no appreciable difference between the variance of the message latency corresponding to high and low priority traffic. The difference between the variance of the two is widening as the network exchanges more load.

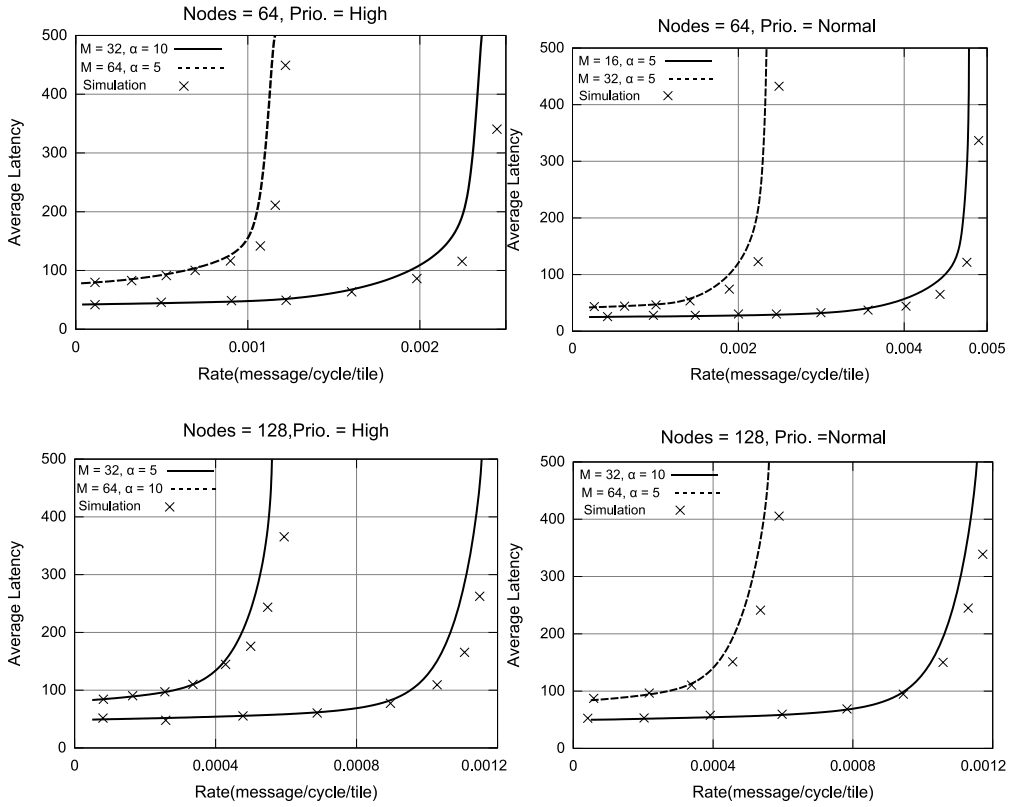


FIGURE 10.3: Validation of the model against the simulation results for networks of 64 and 128 nodes.

10.4 Conclusion

The applications running on a NoC-based SoC typically must meet a minimum performance to successfully perform their intended tasks. Employing differentiated services is a widely used approach to support QoS by relatively prioritizing traffic in the networks employing connection-less communication mechanisms. This chapter presented a novel performance model for predicting differentiated services-based QoS in wormhole-routed

interconnection networks. The comparison with the results produced by a flit-level simulator showed that the model predicts the latency of high and normal traffic with a good degree of accuracy in a wide range of configuration settings.

Chapter 11

An Analytical Model of Broadcast in QoS-Aware Wormhole-Routed NoCs

The previous two chapters presented two models for predicting message latency of multicast traffic and differentiated services-based QoS. Those two models are an improvement to earlier modeling of unicast communication in wormhole-routed interconnection networks. However, traffic at a network is typically a mix of unicast and collective communication operations, where each communication has its own attributes in terms of performance demands. Built on the models presented in the last two chapters, this chapter introduces a model to evaluate the average latency of broadcast communication in multi-port wormhole-routed interconnection networks supporting differentiated services-based QoS.

11.1 Modeling QoS-aware broadcast communication

The model adopts the following assumptions:

- The basic assumption presented in Section 8.1.
- There are two types of messages in the network: unicast and broadcast. A broadcast message is delivered to all nodes in network. A unicast message is sent to

other nodes in the network with equal probability. When a message is generated at a source node, it has a finite probability $0 \leq \beta \leq 1$ of being a broadcast message and probability $(1 - \beta)$ of being unicast.

- Nodes generate traffic independently of each other, according to a *Poisson* process.
- The network adopts multi-port routers scheme.
- The network supports differentiated services-based QoS.

The model combines the features of the previous models in Chapters 9 and 10, therefore, it shares a significant similarities with them. To avoid repeating the contents of those chapters, we only present a guideline of how to merge those two analytical models.

Similar to the analysis of multicast communication, the prediction of QoS-aware broadcast is performed at two stages. At the the first stage, the model adopts the analysis of differentiated services-based QoS presented in Chapter 10 to obtain the total of the average waiting times experienced by a message belonging to a particular class of service. This is the total of the waiting times at all intermediate hops from source to destination.

The second stage relies on the analysis presented in Section 9.1 to compute the average latency of broadcast traffic at each node, using the results of the first stage. Averaging over all nodes in the network yields the average broadcast communication latency.

11.2 Validation

To validate the analysis presented in this chapter we apply the model to a QoS-aware Quarc NoC simulator in which broadcast traffic has priority over unicast. Of course, the model is still valid if unicast is given higher priority, or both unicast and broadcast are assigned an equal priority.

To validate the analytical model we have developed a discrete event simulator of the Quarc NoC supporting differentiated services-based QoS using OMNET++ [96]. The simulator improves on the basic Quarc simulator (Section 5.2.2) by applying modifications to the passive queue and the router.

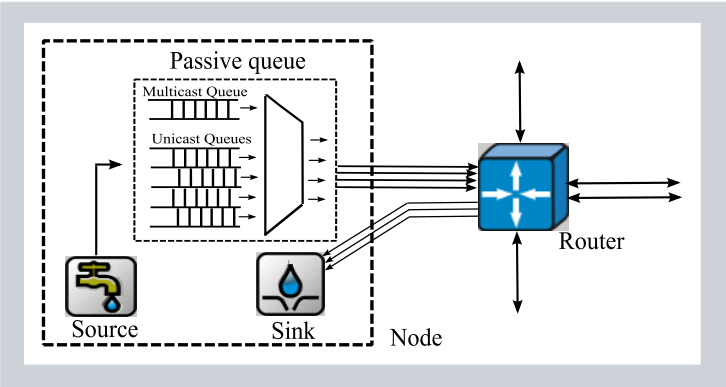


FIGURE 11.1: The components of a sample node in the architecture.

The schematic of the components of the enhanced simulator are depicted in Figure 11.1. The passive queue has a queue to store the broadcast messages and four queues corresponding to each injection link for unicast traffic. The passive queue sends messages based on their priority and creation time. If the broadcast message queue is not empty, messages in the broadcast queue are served according to a FIFO policy. Otherwise, the unicast messages at each rim are served independently as a FIFO queue. Also, the router implements differentiated services-based QoS.

The simulator operates on the assumption defined in Section 5.2.1. Moreover we assume that source produces messages according to a *Poisson* distribution.

The model is compared against the simulation results for numerous configurations by changing the network size, message length and the rate of broadcast traffic. Figure 11.2 compares the simulation results against the analysis for networks having 16, 32, 48, 64 or 128 nodes. The message length can be 16, 32 or 64 flits. And broadcast traffic may comprise 3%, 5% or 10% of the total traffic. Each simulation experiment runs according to the algorithm presented in Section 5.2.3.

In graphs, N, M and β represent the number of nodes, message length and the rate of broadcast traffic, respectively. The horizontal axis in the figures shows the message rate per cycle per node, while the vertical axis describes the average broadcast latency. As can be seen from the the figures, the analytical model predicts QoS-aware broadcast communication with a good degree of accuracy.

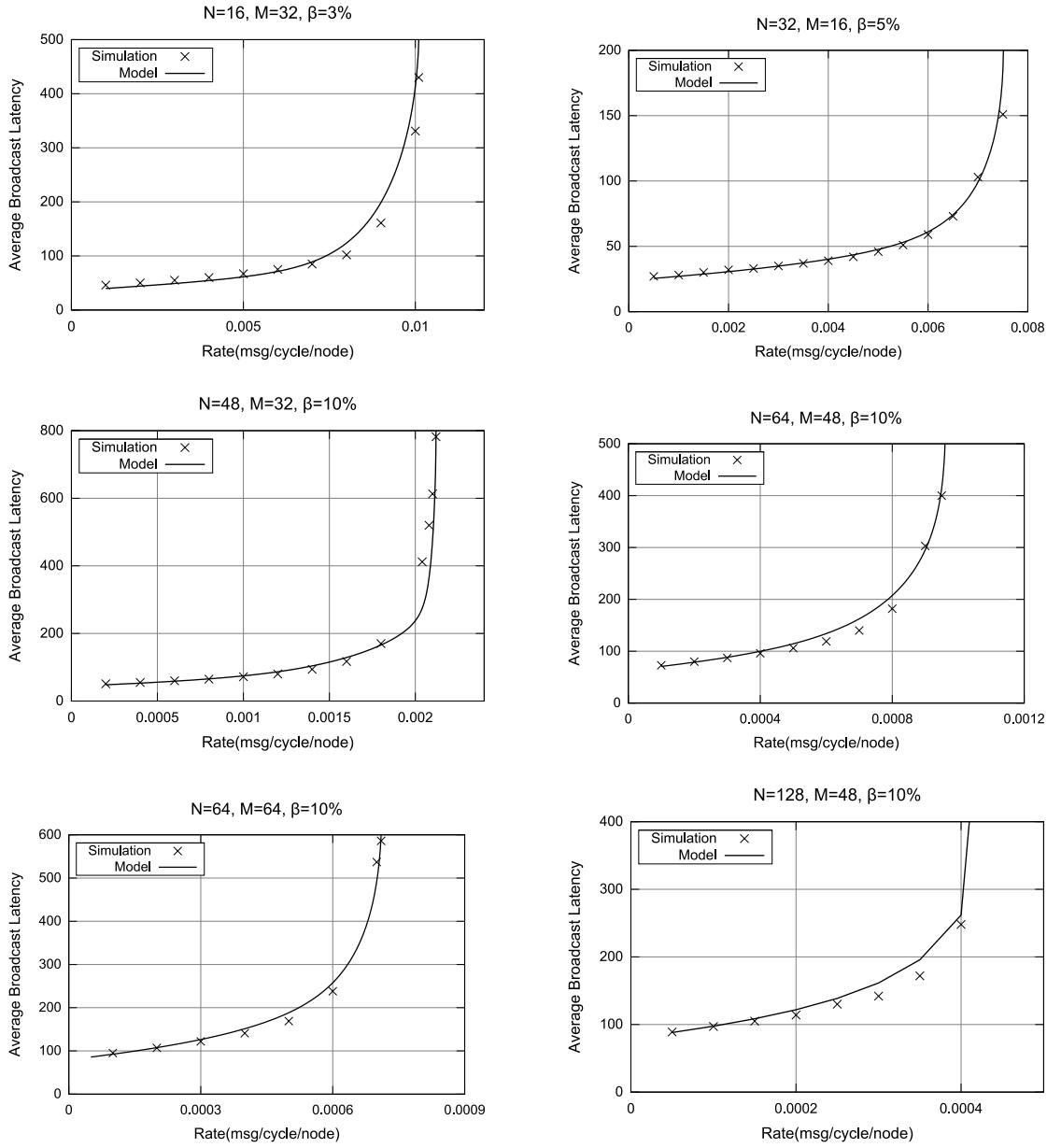


FIGURE 11.2: Comparison of the analytical model against the simulation results.

11.3 Conclusion

The chapter demonstrated that the analytical models presented in previous chapters can be combined to predict more sophisticated traffic. By adopting the models to compute multicast and QoS-aware traffic, the chapter has presented a model to predict broadcast communication in all-port wormhole-routed interconnection networks supporting differentiated services-based QoS. The model predictions have been extensively verified against the results derived from a simulator.

Part V

Conclusion and Trends for Further Research

Chapter 12

Conclusion and Future Work

The traditional SoC communication mediums such as buses and point-to-point connections are being stretched to their limits. Networks on chip (NoCs) are regarded as the most promising communication architecture addressing the design and technological challenges of communication in future large SoCs. NoCs offer a scalable, structured, power efficient and reliable communication medium.

Multicast communication, the most widely-used collective communication operation, typically forms a part of on-chip traffic. By leveraging the shared nature of the medium, buses have been highly efficient in performing such an operation in traditional SoCs. The segmented nature of NoCs, however, does not allow multicast to be as efficient as in buses. Due to similarities between NoCs and interconnection networks for parallel computers, the rich literature on the design of the platforms performing efficient multicast communication at system-level may seem to be a panacea. However, the difference between the cost structure of the system-level networks and NoCs inhibits employing the majority of the solutions in the former to be applied to the latter. As such, the NoC architectures perform poorly in delivering multicast traffic.

The novel Quarc NoC is proposed to offer a highly efficient multicast communication in the NoC realm. The thesis has presented the design and implementation of the building blocks of the Quarc NoC, compared the architecture against a number of existing architectures and proposed novel analytical models to predict the average message latency of complicated traffic patterns.

12.1 Contributions of the thesis

The thesis contributions are classified in three categories, namely, architecture, performance modeling and performance/cost comparison. The most outstanding contribution of the thesis is indeed the Quarc NoC. The Quarc NoC improves on the Spidergon STNoC; targeting an area-efficient, low power, high performance implementation. The Quarc NoC balances traffic on network links, and offers a highly efficient multicast communication. The Quarc NoC employs deterministic routing and wormhole switching. The unique topological properties of the Quarc NoC enables adopting a simple packet format for all types of traffic. Interestingly, in the Quarc NoC the path-based and tree-based graphs corresponding to multicast communication are identical.

The Quarc NoC owes its high performance to a large extent to the ability of the network interface which leverages the unique topological attributes of the Quarc architecture. The Quarc network interface is developed in a modular approach in order to allow easy upgrade and customization of the network interface. In the Quarc network interface, the essential services are performed by the kernel module, while, the instance-specific services are delegated to the shell modules. Following such a modular approach, the Quarc NoC may communicate to any IP with proprietary or emerging communication protocols. Moreover, it facilitates enriching the network interface with new functionality as desired.

To evaluate the performance and cost of the Quarc NoC, the thesis has presented a cost and performance comparison of the Quarc NoC with the Spidergon STNoC and a mesh-based NoC. The Spidergon STNoC has been selected due to the similarities between the architecture and the Quarc NoC, while, the mesh-based NoC is employed as a reference NoC because of the popularity of the topology.

The comparison between the Spidergon STNoC and the Quarc NoC revealed that the Quarc NoC outperforms the Spidergon STNoC significantly, while both architectures can be realized at almost the same cost. The simulation results have shown that the performance of the Quarc NoC is superior to that of the Spidergon architecture in all configuration settings. The most striking performance gain is achieved in the presence of multicast traffic. In the presence of both unicast and broadcast traffic, our experiments

revealed that, on average, the Quarc NoC delivers unicast and broadcast traffic 2 times and 10 times faster than the Spidergon STNoC does.

The FPGA implementation of the Quarc and the Spidergon routers demonstrated that the Quarc router is smaller than the Spidergon router. The comparison between the network interfaces in the architectures showed that in the extreme case, where every node may generate broadcast traffic, the Quarc network interface is slightly larger than the Spidergon network interface.

To compare the Quarc NoC against a mesh-based NoC architecture we have implemented a most performant broadcast routing algorithm for the mesh-based NoC. The FPGA implementation of the router and the network interface in both architectures were prepared to compare the cost of the two architectures. The implementation carried out for architectures with numerous configuration settings in terms of the flit width and the number of virtual channels. The cost comparison showed that the network interface in the mesh-based NoC can be realized using less resources compared to the Quarc network interface. However, the Quarc router implementation requires less resources compared to the router in a mesh-based NoC. Nonetheless, the overall cost of implementing the router and network interface in both architectures are almost equal.

The simulation experiments demonstrated that the Quarc NoC significantly outperforms the mesh-based NoC in performing broadcast communication, while, the Quarc and mesh-based NoCs deliver unicast messages almost with equal performance. Moreover, the analysis of the results revealed that the Quarc NoC is more tolerant to the presence of multicast traffic and responds more effectively to the additional virtual channels.

The thesis demonstrated that analytical modeling based on queuing networks serve as a fast and cost-efficient technique to evaluate performance of interconnection networks. The thesis presented novel analytical models to predict multicast and differentiated services-based QoS traffic. Both models are the first of their kinds. The multicast model computes the message latency of multicast communication in all-port wormhole-routed interconnection networks. The model has been developed using queuing theory and the stochastic properties of the exponentially distributed random variables. The model predicts the average message latency of multicast communication, while network exchanges both unicast and multicast traffic.

The analytical evaluation of differentiated services-based QoS has been performed by modeling the network as a queuing network, where each queue is an $M/G/1$ priority queue. By combining the models predicting multicast and differentiated services-based QoS, the thesis has developed an analytical model to compute the average message latency of broadcast communication in QoS-aware wormhole-routed interconnection networks.

All three models have been applied to the Quarc NoC in a wide range of configuration settings. The comparison between the model predictions and the results derived from a flit level simulator has proved that the models predict the average message latency with an excellent degree of accuracy.

12.2 Direction for future work

The thesis covered the basic elements of the Quarc NoC. There are several interesting issues and open problems that require further investigation. A selection of such problems is listed below.

In the basic version of the Quarc NoC, no provision for QoS is implemented. Our aim is to enhance the router and the network interface to meet the performance requirements of the applications statically at design time and dynamically at run-time.

In the current version of the Quarc NoC, the flow control between the adjacent routers is governed by a link-level on/off protocol; and there is no mechanism to offer end-to-end flow control. Addressing the end-to-end flow control guarantees that packets are not experiencing delays at network interface of the destination nodes, thereby, reducing the network congestion. End-to-end flow control is more crucial in wormhole-routed networks due to adverse effect of the blocked worms on the flow of other worms.

In communication centric SoC development, low power consumption is a paramount feature of the communication architecture. The Quarc NoC is targeting a low power architecture, however, this has to be measured precisely. We are interested in measuring the power consumption in the Quarc NoC and potentially improve the the power consumption of the architecture.

To serve as a successful interconnection in GALS-based SoC paradigm, the Quarc must perform clock-domain-crossing efficiently. Implementing a CDC shell module in the Quarc network interface is another objective we pursue. In particular we are interested in investigating the feasibility of merging the area required by CDC and the multicast module in order to customize the buffer utilization.

We are also interested in implementing services that are built on top of multicast communication, an outstanding examples of such services is cache coherency.

The thesis demonstrated the performance of the Quarc NoC by comparisons against the Spidergon and the mesh-based NoCs. Since the purpose of the thesis has been an introduction to the Quarc NoC, we mainly focused at network level; consequently the evaluations have been carried out based on rather theoretical traffic distributions without considering realistic application level traffic. For a future work we are interested to evaluate the performance of the Quarc NoC, where it serves as an interconnection network for SoCs exchanging real-world traffic.

Bibliography

[1] Luca Benini and Giovanni De Micheli. Powering networks on chips: energy-efficient and reliable interconnect design for socs. In *ISSS '01: Proceedings of the 14th international symposium on Systems synthesis*, pages 33–38, New York, NY, USA, 2001. ACM. ISBN 1-58113-418-5. doi: http://doi.acm.org/10.1145/500001.500009.

[2] L. Benini and G. De Micheli. Networks on chips: a new soc paradigm. *Computer*, 35(1):70–78, Jan 2002. ISSN 0018-9162. doi: 10.1109/2.976921.

[3] W. J. Dally and B. Towles. Route packets, not wires: On-chip interconnection networks. *Proceedings of Design Automation Conference (DAC)*, pages 683–689, 2001.

[4] K. Goossens, J. van Meerbergen, A. Peeters, and P. Wielage. Networks on silicon: Combining best-effort and guaranteed services. In *Proceedings of the Design, Automation and Test Conference*, pages 423–425, March 2002.

[5] A. Greiner P. Guerriert. A generic architecture for on-chip packet-switched interconnections. *Proceedings of Design Automation Conference (DAC)*, pages 683–689, 2001.

[6] Faraydon Karim, Anh Nguyen, and Sujit Dey. An interconnect architecture for networking systems on chips. *IEEE Micro*, 22(5):36–45, 2002. ISSN 0272-1732. doi: http://dx.doi.org/10.1109/MM.2002.1044298.

[7] Shashi Kumar, Axel Jantsch, Mikael Millberg, Johny Oberg, Juha-Pekka Soininen, Martti Forsell, Kari Tiensyrja, and Ahmed Hemani. A network on chip architecture and design methodology. *VLSI, IEEE Computer Society Annual Symposium on*, 0:0117, 2002. doi: http://doi.ieeecomputersociety.org/10.1109/ISVLSI.2002.1016885.

[8] E. Rijpkema, K. G. W. Goossens, A. Radulescu, J. Dielissen, J. van Meerbergen, P. Wielage, and E. Waterlander. Trade offs in the design of a router with both guaranteed and best-effort services for networks on chip. In *DATE '03: Proceedings of the conference on Design, Automation and Test in Europe*, page 10350, Washington, DC, USA, 2003. IEEE Computer Society. ISBN 0-7695-1870-2.

[9] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny. Qnoc: qos architecture and design process for network on chip. *Journal of Systems Arch*, pages 105–128, 2004.

[10] M. Sgroi, M. Sheets, A. Mihal, K. Keutzer, S. Malik, J. Rabaey, and A. Sangiovanni-Vencentelli. Addressing the system-on-a-chip interconnect woes through communication-based design. In *DAC '01: Proceedings of the 38th annual Design Automation Conference*, pages 667–672, New York, NY, USA, 2001. ACM. ISBN 1-58113-297-2. doi: http://doi.acm.org/10.1145/378239.379045.

[11] T. Bjerregaard, S. Mahadevan, R.G. Olsen, and J. Sparsoe. An ocp compliant network adapter for gals-based soc design using the mango network-on-chip. In *System-on-Chip, 2005. Proceedings. 2005 International Symposium on*, pages 171–174, Nov. 2005. doi: 10.1109/ISSOC.2005.1595670.

[12] F. Feliciian and S.B. Furber. An asynchronous on-chip network router with quality-of-service (qos) support. In *SOC Conference, 2004. Proceedings. IEEE International*, pages 274–277, Sept. 2004. doi: 10.1109/SOCC.2004.1362432.

[13] John D. Day and Hubert Zimmermann. The OSI reference model. pages 38–44, 1995.

[14] Kees Goossens, John Dielissen, and Andrei Radulescu. Aethereal network on chip:concepts, architectures, and implementations. *IEEE Design and Test of Computers*, 22(5):414–421, 2005. ISSN 0740-7475. doi: http://doi.ieeecomputersociety.org/10.1109/MDT.2005.99.

[15] Mikael Millberg, Erland Nilsson, Rikard Thid, Shashi Kumar, and Axel Jantsch. The nostrum backbone - a communication protocol stack for networks on chip. In *VLSID '04: Proceedings of the 17th International Conference on VLSI Design*, page 693, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 0-7695-2072-3.

[16] Zhonghai Lu, Bei Yin, and Axel Jantsch. Connection-oriented multicasting in wormhole-switched networks on chip. In *ISVLSI '06: Proceedings of the IEEE Computer Society Annual Symposium on Emerging VLSI Technologies and Architectures*, page 205, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2533-4. doi: http://dx.doi.org/10.1109/ISVLSI.2006.31.

[17] Jian Liu, Li-Rong Zheng, and Hannu Tenhunen. Interconnect intellectual property for network-on-chip (noc). *J. Syst. Archit.*, 50(2-3):65–79, 2004. ISSN 1383-7621. doi: http://dx.doi.org/10.1016/j.sysarc.2003.07.003.

[18] F.A. Samman, T. Hollstein, and M. Glesner. Multicast parallel pipeline router architecture for network-on-chip. In *Design, Automation and Test in Europe, 2008. DATE '08*, pages 1396–1401, March 2008. doi: 10.1109/DATE.2008.4484869.

[19] Jeffrey T. Draper and Joydeep Ghosh. A comprehensive analytical model for wormhole routing in multicomputer systems. *Journal of Parallel and Distributed Computing*, 23(2):202–214, November 1994.

[20] M. Moadeli, A. Shahrabi, W. Vanderbauwhede, and M. Ould-Khaoua. An analytical performance model for the Spidergon NoC. *21st IEEE International Conference on Advanced Information Networking and Applications*, pages 1014–1021, 2007.

[21] A. Shahrabi, L. M. Mackenzie, and M. Ould-Khaoua. A performance model of broadcast communication in wormhole-routed hypercubes. In *MASCOTS '00: Proceedings of the 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, page 98, Washington, DC, USA, 2000. IEEE Computer Society. ISBN 0-7695-0728-X.

[22] E. Beigne, F. Clermidy, P. Vivet, A. Clouard, and M. Renaudin. An asynchronous NoC architecture providing low latency service and its multi-level design framework. *11th IEEE International Symposium on Asynchronous Circuits and Systems*, pages 54–63, 2005.

[23] D. Rostislav, V. Vishnyakov, E. Friedman, and R. Ginosar. An asynchronous router for multiple service levels networks on chip. *11th IEEE International Symposium on Asynchronous Circuits and Systems*, pages 44–53, 2005.

[24] E. Rijpkema, K. Goossens, and P. Wielage. Router Architecture for Networks on Silicon. *Proceedings Of Progress , 2nd Workshop On Embedded Systems*, 2001.

[25] S. Sathe, D. Wiklund, and D. Liu. Design of a switching node (router) for on-chip networks. *Proceedings of 5th International Conference on ASIC*, 1:75 – 78, 2003.

[26] J. Liang, J. Chou, I. Swarbrick, and D. Wingard. An architecture and compiler for scalable on-chip communication. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 12:711 – 726, 2004.

[27] W. D. Weber, A. Laffely, S. Srinivasan, and R. Tessier. A quality-of-service mechanism for interconnection networks in system-on-chips. *Proceedings of Design, Automation and Test in Europe*, 2:1232– 1237, 2005.

[28] D. Andreasson and S. Kumar. Slack-time aware routing in NoC systems. *IEEE International Symposium on Circuits and Systems*, 3:2353– 2356, 2005.

[29] E. Beigne , F. Clermidy, P. Vivet, A. Clouard, and M. Renaudin. An Asynchronous NoC Architecture Providing Low Latency Service and its Multi-Level design Framework. *11th IEEE International Symposium on Asynchronous Circuits and Systems*, pages 54– 63, 2005.

[30] Marcello Coppola, Miltos D. Grammatikakis, Riccardo Locatelli, Giuseppe Maruccia, and Lorenzo Pieralisi. Design of cost-efficient interconnect processing units: Spidergon stnoc (system-on-chip design and technologies. 2008.

[31] J. Duato, S. Yalamanchili, and L. Ni. *Interconnection networks: An Engineering Approach*. Morgan Kaufmann, 2003.

[32] Charles E. Leiserson, Zahi S. Abuhamdeh, David C. Douglas, Carl R. Feynman, Mahesh N. Ganmukhi, Jeffrey V. Hill, Daniel Hillis, Bradley C. Kuszmaul, Margaret A. St. Pierre, David S. Wells, Monica C. Wong, Shaw-Wen Yang, and Robert Zak. The network architecture of the connection machine cm-5 (extended abstract). In *SPAA '92: Proceedings of the fourth annual ACM symposium on Parallel algorithms and architectures*, pages 272–285, New York, NY, USA, 1992. ACM. ISBN 0-89791-483-X. doi: http://doi.acm.org/10.1145/140901.141883.

[33] P.P. Pande, C. Grecu, A. Ivanov, and R. Saleh. Design of Switch for Network on Chip Applications. *Proceedings of Int'l Symposium on Circuit and Systems (ISCAS)*, 5:217–220, May 2003.

[34] Jonathan L. Gross and Jay Yellen. *Graph Theory and Its Applications, Second Edition (Discrete Mathematics and Its Applications)*. Chapman & Hall/CRC, 2005. ISBN 158488505X.

[35] Junming Xu. *Topological Structure and Analysis of Interconnection Networks*. Kluwer Academic Publishers, 2001.

[36] Behrooz Parhami. Chordal rings based on symmetric odd-radix number systems. In *Communications in Computing*, pages 196–199, 2005.

[37] J.-C. Bermond, Is Cnrs, Rue A. Einstein, F. Comellas, and D. F. Hsu. Distributed loop computer networks: A survey, 1995.

[38] F. K. Hwang. A complementary survey on double-loop networks. *Theor. Comput. Sci.*, 263(1-2):211–229, 2001. ISSN 0304-3975. doi: http://dx.doi.org/10.1016/S0304-3975(00)00243-7.

[39] F. K. Hwang. A survey on multi-loop networks. *Theor. Comput. Sci.*, 299(1-3): 107–121, 2003. ISSN 0304-3975. doi: http://dx.doi.org/10.1016/S0304-3975(01)00341-3.

[40] B.W. Arden and Hikyu Lee. Analysis of chordal ring network. *Computers, IEEE Transactions on*, C-30(4):291–295, April 1981. ISSN 0018-9340. doi: 10.1109/TC.1981.1675777.

[41] R. Michael Hord. *The Illiac IV: The First Supercomputer*. Computer Science Pr, 1982.

[42] Yulu Yang, Akira Funahashi, Akiya Jouraku, Hiroaki Nishi, Hideharu Amano, and Toshinori Sueyoshi. Recursive diagonal torus: An interconnection network for massively parallel computers. *IEEE Transactions on Parallel and Distributed Systems*, 12(7):701–715, 2001. ISSN 1045-9219. doi: http://doi.ieeecomputersociety.org/10.1109/71.940745.

[43] Raman Beivide, Carmen Martanez, Cruz Izu, Jaime Gutierrez, Jose-Angel Gregorio, and Jose Miguel-alonso. Chordal topologies for interconnection networks. In *Proc. 5th International Symp. High-Performance Computing*, pages 385–393. Springer-Verlag, 2003.

[44] J.-F. Fang, J.-Y. Hsiao, and C.-Y. Tang. Embedding meshes and torus networks onto degree-four chordal rings. *Computers and Digital Techniques, IEE Proceedings* -, 145(2):73–80, Mar 1998. ISSN 1350-2387.

[45] F.K. Hwang. Comments on "reliable loop topologies for large local computer networks". *Computers, IEEE Transactions on*, C-36(3):383–384, March 1987. ISSN 0018-9340. doi: 10.1109/TC.1987.1676914.

[46] C S Raghavendra and J S Silvester. A survey of multi-connected loop topologies for local computer networks. *Comput. Netw. ISDN Syst.*, 11(1):29–42, 1986. ISSN 0169-7552. doi: http://dx.doi.org/10.1016/0169-7552(86)90027-9.

[47] Umit Y. Ogras, Jingcao Hu, and Radu Marculescu. Key research problems in noc design: a holistic perspective. In *CODES+ISSS '05: Proceedings of the 3rd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, pages 69–74, New York, NY, USA, 2005. ACM. ISBN 1-59593-161-9. doi: http://doi.acm.org/10.1145/1084834.1084856.

[48] John Dielissen Andrei, Andrei Radulescu, Kees Goossens, and Edwin Rijpkema. Concepts and implementation of the philips network-on-chip. In *IP-Based SoC Design*, 2003.

[49] J. Bainbridge and S. Furber. Chain: a delay-insensitive chip area interconnect. *Micro, IEEE*, 22(5):16–23, Sep/Oct 2002. ISSN 0272-1732. doi: 10.1109/MM.2002.1044296.

[50] Young Bok Kim and Yong-Bin Kim. Fault tolerant source routing for network-on-chip. *Defect and Fault-Tolerance in VLSI Systems, IEEE International Symposium on*, 0:12–20, 2007. doi: http://doi.ieeecomputersociety.org/10.1109/DFT.2007.14.

[51] M. Millberg, E. Nilsson, R. Thid, and A. Jantsch. Guaranteed bandwidth using looped containers in temporally disjoint networks within the nostrum network on chip. *Proceedings of the Conference on Design, Automation and Test in Europe*, 2:890 – 895, 2004.

[52] Parviz Kermani and Leonard Kleinrock. Virtual cut-through: a new computer communication switching technique. *Computer Networks*, 3:267–286, 1979.

[53] William J Dally and Charles L.Seitz . The torus routing chip. *Journal of Distributed Computing*, 1986.

[54] W. J.Dally and C. L. Seitz. Deadlock-free Message Routing in Multiprocesor Interconnection Networks. *IEEE Transactions on Computers*, 1987.

[55] Christopher J. Glass, Christopher J. Glass, Lionel M. Ni, and Lionel M. Ni. The turn model for adaptive routing. In *In Proceedings of the International Symposium on Computer Architecture*, pages 278–287, 1992.

[56] M. Karol, M. Hluchyj, and S. Morgan. Input versus output queueing on a space-division packet switch. *Communications, IEEE Transactions on*, 35(12):1347–1356, Dec 1987. ISSN 0090-6778.

[57] M.K.M. Ali and M. Youssefi. The performance analysis of an input access scheme in a high-speed packet switch. In *INFOCOM '91. Proceedings. Tenth Annual Joint Conference of the IEEE Computer and Communications Societies. Networking in the 90s., IEEE*, pages 454–461 vol.2, Apr 1991. doi: 10.1109/INFCOM.1991.147539.

[58] Nicholas William McKeown. *Scheduling algorithms for input-queued cell switches*. PhD thesis, Berkeley, CA, USA, 1995.

[59] Tobias Bjerregaard and Shankar Mahadevan. A survey of research and practices of network-on-chip. *ACM Comput. Surv.*, 38(1):1, 2006. ISSN 0360-0300. doi: http://doi.acm.org/http://doi.acm.org/10.1145/1132952.1132953.

[60] Erno Salminen, Ari Kulmala, and Timo D. Hamalainen. Survey of network-on-chip proposals. In *WHITE PAPER, @ OCP-IP,*, 2008.

[61] Charles E. Leiserson. Fat-trees Universal Networks for Hardware-Efficient Supercomputing. *IEEE Transactions on Computer*, 34:892–901, 1985.

[62] Adrijean Andriahantenaina and Alain Greiner. Micro-network for soc: Implementation of a 32-port spin network. In *DATE '03: Proceedings of the conference on Design, Automation and Test in Europe*, page 11128, Washington, DC, USA, 2003. IEEE Computer Society. ISBN 0-7695-1870-2.

[63] Michael Bedford Taylor, Jason Kim, Jason Miller, David Wentzlaff, Fae Ghodrat, Ben Greenwald, Henry Hoffman, Paul Johnson, Jae-Wook Lee, Walter Lee, Albert Ma, Arvind Saraf, Mark Seneski, Nathan Shnidman, Volker Strumpen, Matt Frank, Saman Amarasinghe, and Anant Agarwal. The raw microprocessor: A computational fabric for software circuits and general-purpose programs. *IEEE Micro*, 22(2):25–35, 2002. ISSN 0272-1732. doi: http://dx.doi.org/10.1109/MM.2002.997877.

[64] Raw architecure workstation. *http://www.cag.lcs.mit.edu/raw*.

[65] Martti Forsell. A scalable high-performance computing solution for networks on chips. *IEEE Micro*, 22(5):46–55, 2002. ISSN 0272-1732. doi: http://dx.doi.org/10.1109/MM.2002.1044299.

[66] Edwin Rijpkema, Kees Goossens, and Paul Wielage. A router architecture for networks on silicon. In *Proceedings of Progress 2001, 2nd Workshop on Embedded Systems*, Veldhoven, the Netherlands, October 2001.

[67] Kees Goossens, John Dielissen, Jef van Meerbergen, Peter Poplavko, Andrei Ruadulescu, Edwin Rijpkema, Erwin Waterlander, and Paul Wielage. Guaranteeing the quality of services in networks on chip. pages 61–82, 2003.

[68] Andrei Radulescu, John Dielissen, Kees Goossens, Edwin Rijpkema, and Paul Wielage. An efficient on-chip network interface offering guaranteed services, shared-memory abstraction, and flexible network configuration. *IEEE Transactions on CAD of Integrated Circuits and Systems*, 24:4–17, 2004.

[69] Fernando Moraes, Ney Calazans, Aline Mello, Leandro Möller, and Luciano Ost. HERMES: an infrastructure for low area overhead packet-switching networks on chip. *Journal of VLSI Integration*, 38:69–93, 2004.

[70] T. Bjerregaard and J. Spars0. A router architecture for connection-oriented service guarantees in the mango clockless network-on-chip. *11th IEEE International Symposium on Asynchronous Circuits and Systems*, pages 44–53, 2005.

[71] Srinivasan Murali and Giovanni De Micheli. Bandwidth-constrained mapping of cores onto noc architectures. In *DATE '04: Proceedings of the conference on Design, automation and test in Europe*, page 20896, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 0-7695-2085-5-2.

[72] Srinivasan Murali and Giovanni De Micheli. Sunmap: a tool for automatic topology selection and generation for nocs. In *DAC '04: Proceedings of the 41st annual Design Automation Conference*, pages 914–919, New York, NY, USA, 2004. ACM. ISBN 1-58113-828-8. doi: http://doi.acm.org/10.1145/996566.996809.

[73] Faraydon Karim, Anh Nguyen, Sujit Dey, and Ramesh Rao. On-chip communication architecture for oc-768 network processors. In *DAC '01: Proceedings of the 38th annual Design Automation Conference*, pages 678–683, New York, NY, USA, 2001. ACM. ISBN 1-58113-297-2. doi: http://doi.acm.org/10.1145/378239.379047.

[74] Luciano Bononi and Nicola Concer. Simulation and analysis of network on chip architectures: ring, spidergon and 2d mesh. In *DATE '06: Proceedings of the conference on Design, automation and test in Europe*, pages 154–159, 3001 Leuven, Belgium, Belgium, 2006. European Design and Automation Association. ISBN 3-9810801-0-6.

[75] William J. Dally. Performance analysis of k-ary n-cube interconnection networks. *IEEE Trans. Comput.*, 39(6):775–785, 1990. ISSN 0018-9340. doi: http://dx.doi.org/10.1109/12.53599.

[76] Dhabaleswar K. Panda , Sanjay Singal , and Ram Kesavan . Multidestination Message Passing in Wormhole k-ary n-cube Networks with Base Routing Conformed Paths. *IEEE Transactions on Parallel and Distributed Systems*, 1995.

[77] Ocp-ip, open core protocol specification, version 2.0. *www.ocpip.org*, 2003.

[78] Virtual component interface standard. *http://www.vsi.org.*

[79] Arm, amba 3.0 axi specification. 2004.

[80] G. Buzzard, D. Jacobson, S. Marovich, and J. Wilkes. Hamlyn: A high-performance network interface with sender-based memory management. *Proceedings of Hot Interconnects*, 1995.

[81] D. J. Culler, J. P. Singh, and A. Gupta. *Parallel Computer Architecture: A Hardware/Software Approach.* Morgan Kaufmann, San Francisco, CA, 1999.

[82] Peter Steenkiste. A high-speed network interface for distributed-memory systems: architecture and applications. *ACM Trans. Comput. Syst.*, 15(1):75–109, 1997. ISSN 0734-2071. doi: http://doi.acm.org/10.1145/244764.244767.

[83] A.A. Chien, M.D. Hill, and S.S. Mukherjee. Design challenges for high-performance network interfaces. *Computer*, 31(11):42–44, Nov 1998. ISSN 0018-9162. doi: 10.1109/2.730735.

[84] Seth Copen Goldstein and Timothy Callahan. Nifdy: A low overhead, high throughput network interface. *Computer Architecture, International Symposium on*, 0:230, 1995. doi: http://doi.ieeecomputersociety.org/10.1109/ISCA.1995.524564.

[85] Antoine Scherrer, Tanguy Risset, and Antoine Fraboulet. Hardware Wrapper Classification and Requirements for On-Chip Interconnects, 2004.

[86] Giovanni De Micheli and Luca Benini. *Networks on Chips: Technology and Tools (Systems on Silicon)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2006. ISBN 0123705215.

[87] Praveen Bhojwani and Rabi Mahapatra. Interfacing cores with on-chip packet-switched networks. In *VLSID '03: Proceedings of the 16th International Conference on VLSI Design*, page 382, Washington, DC, USA, 2003. IEEE Computer Society. ISBN 0-7695-1868-0.

[88] Dake Liu, Daniel Wiklund, Erik Svensson, Olle Seger, and Sumant Sathe. Socbus: The solution of high communication bandwidth on chip and short ttm. *Real-Time and Embedded Computing Conference*, 2002.

[89] Philippe Martin. Design of a virtual component neutral network-on-chip transaction layer. In *DATE '05: Proceedings of the conference on Design, Automation and Test in Europe*, pages 336–337, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2288-2. doi: http://dx.doi.org/10.1109/DATE.2005.114.

[90] Luca Benini and Giovanni De Micheli. Powering networks on chips: energy-efficient and reliable interconnect design for socs. In *ISSS '01: Proceedings of the 14th international symposium on Systems synthesis*, pages 33–38, New York, NY, USA, 2001. ACM. ISBN 1-58113-418-5. doi: http://doi.acm.org/10.1145/500001.500009.

[91] Marshall T. Rose. *The open book: a practical perspective on OSI*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1990. ISBN 0-13-643016-3.

[92] Philips Semiconductors. Device Transaction Level (DTL) Protocol Specification. *Version 2.2*, 2002.

[93] Stergios Stergiou, Federico Angiolini, Salvatore Carta, Luigi Raffo, Davide Bertozzi, and Giovanni De Micheli. Xpipes lite: A synthesis oriented design library for networks on chips. In *In DATE*, pages 1188–1193. IEEE, 2005.

[94] Jabulani Nyathi, Souradip Sarkar, and Partha Pratim Pande. Multiple clock domain synchronization for network on chip architectures. In *SOC Conference, 2007 IEEE International*, pages 291–294, Sept. 2007. doi: 10.1109/SOCC.2007.4545477.

[95] Tiberiu Chelcea and Steven M. Nowick. Robust interfaces for mixed-timing systems. *IEEE Trans. Very Large Scale Integr. Syst.*, 12(8):857–873, 2004. ISSN 1063-8210. doi: http://dx.doi.org/10.1109/TVLSI.2004.831476.

[96] A. Varga. Omnet++. *IEEE Network Interactive, in the column Software Tools for Networking, www.omnetpp.org*, 16(4):683–689, 2002.

[97] Jian Liang, S. Swaminathan, and R. Tessier. Asoc: a scalable, single-chip communications architecture. In *Parallel Architectures and Compilation Techniques, 2000. Proceedings. International Conference on*, pages 37–46, 2000. doi: 10.1109/PACT.2000.888329.

[98] Panda, Dhabaleswar K., Singal, Sanjay, and Kesavan, Ram. Multidestination message passing in wormhole k-ary n-cube networks with base routing conformed paths. *IEEE Trans. Parallel Distrib. Syst.*, 10(1):76–96, 1999. ISSN 1045-9219. doi: http://dx.doi.org/10.1109/71.744844.

[99] Ram Kesavan and Dhabaleswar K. P. Minimizing node contention in multiple multicast on wormhole k-ary n-cube networks. In *In Proceedings of the International Conference on Parallel Processing*, pages 188–195, 1996.

[100] Xiaola Lin, Xiaola Lin, and Lionel M. Ni. Deadlock-free multicast wormhole routing in multicomputer networks. In *In Proceedings of the International Symposium on Computer Architecture*, pages 116–124, 1991.

[101] Dhabaleswar K. P, Sanjay Singal, and Ram Kesavan. Multidestination message passing in wormhole k-ary n-cube networks with base routing conformed paths, 1999.

[102] M Moadeli, A Shahrabi, and W Vanderbauwhede. Analytical modelling of communication in the rectangular mesh noc. In *ICPADS '07: Proceedings of the 13th International Conference on Parallel and Distributed Systems*, pages 1–8, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 978-1-4244-1889-3. doi: http://dx.doi.org/10.1109/ICPADS.2007.4447826.

[103] Mahmoud Moadeli, Wim Vanderbauwhede, and Ali Shahrabi. A performance model of communication in the quarc noc. In *ICPADS '08: Proceedings of the 2008 14th IEEE International Conference on Parallel and Distributed Systems*, pages 908–913, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3434-3. doi: http://dx.doi.org/10.1109/ICPADS.2008.54.

[104] Gunter Bolch, Stefan Greiner, Hermann de Meer, and Kishor Shridharbhai Trivedi. *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications*. Wiley, 1998.

[105] Jane Hillston. *A compositional approach to performance modelling*. Cambridge University Press, New York, NY, USA, 1996. ISBN 0-521-57189-8.

[106] F. Bause and P. S. Kritzinger. *Stochastic Petri Nets – An Introduction to the Theory*. Vieweg Verlag, Germany, 2002.

[107] Mahmoud Moadeli and Wim Vanderbauwhede. Towards a framework to evaluate performance of the nocs. *Workshop on Process Algebra and Stochastically Timed Activities*, 2006.

[108] Marcello Coppola, Stephane Curaba, Miltos D. Grammatikakis, Riccardo Locatelli, Giuseppe Maruccia, and Francesco Papariello. Occn: a noc modeling framework for design exploration. *J. Syst. Archit.*, 50(2-3):129–163, 2004. ISSN 1383-7621. doi: http://dx.doi.org/10.1016/j.sysarc.2003.07.002.

[109] Denvil Smith. A flit level simulator for wormhole routing. In *ACM-SE 38: Proceedings of the 38th annual on Southeast regional conference*, pages 109–116, New York, NY, USA, 2000. ACM. ISBN 1-58113-250-6. doi: http://doi.acm.org/10.1145/1127716.1127742.

[110] Randolph Nelson. *Probability, stochastic processes, and queueing theory: the mathematics of computer performance modeling*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[111] Partha Pratim Pande, Cristian Grecu, Michael Jones, Andre Ivanov, and Resve Saleh. Performance evaluation and design trade-offs for network-on-chip interconnect architectures. *IEEE Trans. Comput.*, 54(8):1025–1040, 2005. ISSN 0018-9340. doi: http://dx.doi.org/10.1109/TC.2005.134.

[112] Zvika Guz, Isask'har Walter, Evgeny Bolotin, Israel Cidon, Ran Ginosar, and Avinoam Kolodny. Network Delays and Link Capacities in Application-specificwormhole NoCs. *Journal of VLSI Design*, 2006.

[113] R Greenberg and L. Guan. Modelling and Comparison of Wormhole Routed Mesh and Torus. *Ninth IASTED intel*, 1997.

[114] H. Hashemi-Najafabadi, H. Sarbazi-Azad, and P. Rajabzadeh. An accurate performance model of fully adaptive routing in wormhole-switched two-dimensional mesh multicomputers. *Microprocessors and Microsystems*, 31(7):445–455, 2007. URL http://dblp.uni-trier.de/db/journals/mam/mam31.html#Hashemi-NajafabadiSR07.

[115] A. Shahrabi, L. M. Mackenzie, and M. Ould-Khaoua. An analytical model of wormhole-routed hypercubes under broadcast traffic. *Perform. Eval.*, 53(1):23–42, 2003. ISSN 0166-5316. doi: http://dx.doi.org/10.1016/S0166-5316(02)00204-3.

[116] M. Moadeli et al. Communication Modeling of the Spidergon NoC with Virtual Channels. In *ICPP*, 2007.

[117] L. Kleinrock. *Queueing Systems Volume I: Theory.* John Wiley and Sons, 1975.

[118] J. Protic, M. Tomasevic, and V. Milutinovic. Distributed shared memory: concepts and systems. *Parallel and Distributed Technology: Systems and Applications, IEEE*, 4(2):63–71, Summer 1996. ISSN 1063-6552. doi: 10.1109/88.494605.

[119] Philip K. McKinley, Hong Xu, Lionel M. Ni, and Abdol-Hossein Esfahanian. Unicast-based multicast communication in wormhole-routed networks. *IEEE Trans. Parallel Distrib. Syst.*, 5(12):1252–1265, 1994. ISSN 1045-9219. doi: http://dx.doi.org/10.1109/71.334899.

[120] Chi-Ming Chiang and L. M. Ni. Efficient software multicast in wormhole-routed unidirectional multistage networks. In *SPDP '95: Proceedings of the 7th IEEE Symposium on Parallel and Distributeed Processing*, page 106, Washington, DC, USA, 1995. IEEE Computer Society. ISBN 0-8186-7195-5.

[121] Ram Kesavan and Dhabaleswar K. Panda. Multiple multicast with minimized node contention on wormhole k-ary n-cube networks. *IEEE Trans. Parallel Distrib. Syst.*, 10(4):371–393, 1999. ISSN 1045-9219. doi: http://dx.doi.org/10.1109/71.762817.

[122] Yih jia Tsai and Philip K. Mckinley. An extended dominating node approach to collective communication in all-port wormhole-routed 2d meshes. In *In Proceedings of the Scalable High Performance Computing Conference*, page pp., 1994.

[123] David F. Robinson, Dan Judd, Philip K. McKinley, and Betty H. C. Cheng. Efficient multicast in all-port wormhole-routed hypercubes. *J. Parallel Distrib. Comput.*, 31(2):126–140, 1995. ISSN 0743-7315. doi: http://dx.doi.org/10.1006/jpdc.1995.1151.

[124] X. Lin , A.-H. Esfahanian , and A Burago. Adaptive Wormhole Routing in Hypercube Multicomputers. *Journal of Parallel and Distributed Computing*, pages 274–277, 1998.

[125] Xiaola Lin , Philip K. McKinley , and Abdol-Hossein Esfahanian. Adaptive multicast wormhole routing in 2d mesh multicomputers. *Parallel Architectures and Languages Europe*, 1993.

[126] Dhabaleswar K. Panda, Sanjay Singal, and Pradeep Prabhakaran. Multidestination message passing mechanism conforming to base wormhole routing scheme. In *PCRCW '94: Proceedings of the First International Workshop on Parallel Computer Routing and Communication*, pages 131–145, London, UK, 1994. Springer-Verlag. ISBN 3-540-58429-3.

[127] Ju-Young L. Park, Hyeong-Ah Choi, Natawut Nupairoj, and L.M. Ni. (r) construction of optimal multicast trees based on the parameterized communication model. *Parallel Processing, International Conference on*, 1:0180, 1996. doi: http://doi.ieeecomputersociety.org/10.1109/ICPP.1996.537159.

[128] Jenq-Shyan Yang and Chung-Ta King. Efficient tree-based multicast in wormhole-routed 2d meshes. *Parallel Architectures, Algorithms, and Networks, International Symposium on*, 0:494, 1997. ISSN 1087-4089. doi: http://doi.ieeecomputersociety.org/10.1109/ISPAN.1997.645142.

[129] M. Moadeli, W. Vanderbauwhede, and A. Shahrabi. Quarc: A novel network-on-chip architecture. In *Parallel and Distributed Systems, 2008. ICPADS '08. 14th IEEE International Conference on*, pages 705–712, Dec. 2008. doi: 10.1109/ICPADS.2008.53.

[130] RFC 3086. Definition of differentiated services per domain behaviors and rules for their specification. *IETF Request for Comments 3086*, 2, 2001.

[131] L. Kleinrock. *Queueing Systems Volume II: Computer Applications*. John Wiley and Sons, 1975.

Publications

during the course of this research

- **Journal papers**
  - M. Moadeli and W. Vanderbauwhede, An Analytical Model of Broadcast Communication in All-port QoS-Aware NoC, Journal of Interconnection networks.
  - M. Moadeli and W. Vanderbauwhede, Communication Modeling of Multicast in All-Port Wormhole-Routed NoCs, Journal of Systems and Software, 2010.
  - W. Vanderbauwhede, L. Azzopardi, M. Moadeli, FPGAs Hold Key to Greener Search, XCell Journal, 2010.
  - M. Moadeli, W. Vanderbauwhede, A. Shahrabi and P. Maji, An Analytical Performance Model for the Spidergon NoC with Virtual Channels, Journal of Systems Architectures, 2009.
  - M. Moadeli, A. Shahrabi, W. Vanderbauwhede, and M. Ould-Khaoua, An Analytical Comparison of the Spidergon and Rectangular Mesh NoCs, Journal of Interconnection Networks, Volume: 10, Issues: 1-2 (March & June 2009), pp. 167-188.
  - Mahmoud Moadeli, Ali Shahrabi, Wim Vanderbauwhede, and Mohammed Ouldkhaoua, Communication Modelling of QoS-Aware Wormhole-Routed NoCs, Journal of Interconnection Networks, Vol. 9, No. 4, pp. 409-423, December 2008.

- **Conference and worksop papers**
  - M. Moadeli and W. Vanderbauwhede, A Communication Model of Broadcast in Wormhole-Routed Networks on-Chip, IEEE, 23^{rd} International Conference on Advanced Information Networking and Applications, UK, 2009.
  - M. Moadeli, P. Maji and W. Vanderbauwhede, Quarc: a High-Efficiency Network on-Chip Architecture, IEEE, 23^{rd} International Conference on Advanced Information Networking and Applications, UK, 2009.
  - M. Moadeli and W. Vanderbauwhede, A Performance Model of Multicast Communication in Wormhole-Routed, IEEE, 8^{th} International Workshop on Performance Modeling, Evaluation and Optimization of Ubiquitous Computing and Networked Systems, Italy, 2009.

- M. Moadeli, P. Maji and W. Vanderbauwhede, Design and implementation of the Quarc Network on-Chip, IEEE, 16^{th} International Workshop on Reconfigurable Architectures, Italy, 2009.
- W. Vanderbauwhede, L. Azzopardi, M. Moadeli, FPGA-Accelerated Information Retviela: High-Efficiency Document Filtering, IEEE, International Conference on Field Programmable Logic and Applications (FPL), Czech Republic, 2009.
- L. Azzopardi, W. Vanderbauwhede, M. Moadeli, Developing Energy Efficient Filtering Systems, SIGIR, 2009.
- M. Moadeli,W. Vanderbauwhede and A. Shahrabi, Quarc: a novel Network on-Chip Architecture, IEEE, 14^{th} International Conference on Parallel and Distributed Systems, Australia, 2008.
- M. Moadeli, W. Vanderbauwhede and A. Shahrabi, A Performance Model of Communication in the Quarc NoC, IEEE, International Workshop on Performance Modeling and Analysis of Communication in Wired and Wireless Networks, Australia, 2008.
- M. Moadeli, W. Vanderbauwhede and A. Shahrabi, Modeling Differentiated Services-based QoS in Wormhole-routed NoCs, IEEE, 22^{nd} International Conference on Advanced Information Networking and Applications, Japan, 2008.
- M. Moadeli, A. Shahrabi, W. Vanderbauwhede and M. Ould-Khaoua, Communication Modeling of the Spidergon NoC with Virtual Channels, IEEE, 36^{th} International Conference on Parallel Processing (ICPP), China, 2007.
- M. Moadeli, A. Shahrabi, W. Vanderbauwhede and M. Ould-Khaoua, An Analytical Performance Model for the Spidergon NoC, IEEE, 21^{st} International Conference on Advanced Information Networking and Applications, Canada, 2007.
- M. Moadeli, A. Shahrabi and W. Vanderbauwhede, Analytical Modeling of Communication in the Rectangular Mesh, IEEE, International Workshop on Performance Meddling and Analysis of Communication in Wired and Wireless Networks, Taiwan, 2007.
- M. Moadeli, W. Vanderbauwhede, Towards a Framework to Evaluate the Performance of the NoCs, Fifth Workshop on Process Algebra and Stochastically Timed Activities, UK, 2006.

- **Publications under review**

  - M. Moadeli, P. Maji and W. Vanderbauwhede, The Quarc network interface, NoC Symposium, 2010.
  - M. Moadeli, P. Maji and W. Vanderbauwhede, A comparison between the Quarc NoC and ubiquitous mesh, NoC Symposium, 2010.