Towards an Adaptive Interconnect for System-on-Chip

by Suboh Suboh

B.Sc. in Computer and Control Engineering, August 1994,
Hijjawi Faculty for Applied Engineering, Yarmouk University, Jordan

Master of Science in Computer Engineering, August 2001,
School of Engineering and Applied Science, The George Washington University, USA

A Dissertation submitted to

The Faculty of
The School of Engineering and Applied Science
of The George Washington University in partial satisfaction
of the requirements for the degree of Doctor of Philosophy

August 31, 2010

Dissertation directed by

Tarek El-Ghazawi

Professor of Engineering and Applied Science

The School of Engineering and Applied Sciences of The George Washington University certifies that Suboh Suboh has passed the Final Examination for the degree of Doctor of Philosophy as of June 21, 2010. This is the final and approved form of the dissertation

TOWARDS AN ADAPTIVE INTERCONNECT FOR SYSTEM-ON-CHIP

Suboh A. Suboh

Dissertation Research Committee:

Tarek El-Ghazawi, Professor of Engineering and Applied Science, Dissertation Director

Can E. Korman, Professor of Engineering and Applied Science, Committee Member

Shahrokh Ahmadi, Assistant Research Professor of Engineering and Applied Science, Committee Member

Man-Lap (Alex) Li, Assistant Professor of Engineering and Applied Science, Committee Member

Mohamed Bakhouya, Research Scientist, Computer Science, Université de Technologie de Belfort-Montbéliard (UTBM), France, Committee Member

Vikram K. Narayana, Postdoctoral Research Scientist, The George Washington University, Committee Member

Acknowledgment

I am thankful to many people for their help and support throughout my doctoral program. A special recognition goes to my dissertation advisor, Professor Tarek El-Ghazawi for his guidance, patience and encouragement to help me complete this work. I appreciate the long hour's professor El-Ghazawi spent with me brainstorming and discussing new ideas.

I also would like to extend my appreciation to Dr. Bakhouya, for all the good discussions, and ideas that we have worked on and implemented together; and Dr. Narayana for the guidance, help and support he provided me.

My gratitude goes to the Dissertation committee member, Professor C. Korman for his continuous help and support throughout my doctoral program, Professor Ahmadi, for the good discussions and thorough comments, Professor Li for agreeing to serve on my dissertation committee.

My appreciation also extends to my former advisor, Professor Nikitas Alexandridis for giving me the opportunity to teach classes in the ECE department and work with him on several projects such as IP-Tools LUCITE project, and for his help and support.

I am thankful to all my colleagues at the HPCL lab: Olivier Serres, Mohamed Taher, Abdullah Kayi and Dr. Saumil Merchant, for their kind help, and special thanks to Esam El-Araby and Abdullah Al-Dahlawi for all their help and support, and for being good friends.

I thank my family for their support and unwavering love.

Abstract

Towards An Adaptive Interconnect For System on Chip

Emerging System-on-Chip (SoC) platforms, such as those for mobile systems, are typically battery-powered systems that have to support a wide range of streaming applications such as video and audio. To meet the high performance requirements for these systems, SoCs integrate various hardware resources such as CPUs, DSPs, Memory and I/O peripherals. These hardware resources normally communicate using a shared medium such as the bus system. For large SoCs, the bus-based schemes become restrictive because they are non-scalable and have higher overheads that adversely impact performance and energy consumption. Network-on-Chip (NoC) architectures were therefore proposed to solve the scalability problem experienced in bus-based SoCs. They incorporate a communication infrastructure defined by a network topology, routers and switches, in order to provide a scalable and high performance interconnection between SoC resources while satisfying the constraints of embedded platforms.

This work: 1. conducts a comprehensive empirical and analytical study to characterize the impact of NoC topology on performance, as well as energy and area requirements; 2. introduces the concept of Field Adaptable NoC (FAN); and 3. proposes the use of Network Calculus as a NoC modeling technique for design space exploration.

The empirical study aimed at exploring different configurations of network on chip has been conducted. Different topologies are selected and analyzed using specific routing and switching methods to check the suitability for NoC. Additionally, since the energy and area are the primary concerns in NoC, models for energy and area are adapted and simulators are built to calculate these metrics.

The static on-chip interconnects, despite their advantages over bus based systems, have latency and throughput that depend on the topology as well as application traffic scenarios. Application-specific optimizations, needed to address this issue, require an adaptable NoC infrastructure to allow in-field customizations. A methodology for augmenting a NoC with a programmable infrastructure that allows application-specific adaptation called Field Adaptable NoC (FAN) is presented. The resulting infrastructure provides the flexibility required to maximize the performance for a given application, while adhering to the system constraints such as energy and area. To evaluate the proposed methodology of the adaptable NoC, the WK-recursive on-chip interconnect as well as the 2D Mesh network are used as case studies.

Performance analysis and evaluation of on-chip interconnect (OCI) architectures are widely based on system level simulation techniques. Due to the time-consuming nature of simulation based approaches, as well as their limitations in providing an understanding on the effects of various design parameters, analytical models for NoCs are essential. Therefore, we introduce Network Calculus for design space exploration and as a suitable analytical modeling tool for estimating and evaluating the performance of on-chip interconnects. One of the primary advantages of Network Calculus is its ability to model network traffic and service in terms of bounds, which enables worst-case analyses to be carried out. To validate this analytical method and demonstrate its applicability for NoC, different scenarios for application traffic are analyzed for different NoC configurations and compared with simulation. The results from the analyses closely match those obtained from the comprehensive empirical study.

Table of Contents

Acknowledgment ..... **iii**

Abstract ..... **iv**

Table of Contents ..... **vi**

List of Figures ..... **ix**

List of Tables ..... **xiii**

List of Acronyms ..... **xiv**

Chapter 1 - Introduction **1**

1.1 Background **1**

1.1.1 Platform-based design methodology 2

1.1.2 Evolution of Network-on-Chip 4

1.2 Motivations **6**

1.3 Problem Statement **8**

1.4 Proposed Approach **8**

1.5 Scope and Limitations **10**

1.6 Organization of the Dissertation **11**

Chapter 2 -Related Work **13**

2.1 NoC Architecture **14**

2.1.1 Topology studies 14

2.1.2 NoC router 15

2.1.3 Communication infrastructure 16

2.1.4 Adaptable NOC architecture 18

2.2 Performance Studies **19**

2.3 Energy and Area Studies **21**

2.4 Analytical Models for NoC 23

Chapter 3 - Exploring NoCs: Methodology and Performance Evaluations 25

3.1 Structures and Topological models 26

3.2 The Performance Model 28

3.3 The Application Model 30

3.3.1 Synthetic generator workload 30

3.3.2 Real application workload 31

3.4 Simulation Methodology 33

3.5 Experimental results 36

Chapter 4 Exploring NoC Architectures: Energy and Area Considerations 42

4.1 The Energy Model 43

4.2 The Area Model 44

4.3 WK-Recursive Network Overview 46

4.4 WK-Recursive Network Evaluation 49

4.4.1 Performance evaluation 49

4.4.2 The energy evaluation 52

4.4.3 The area evaluation 56

4.5 Scalability Study 57

Chapter 5 Field Adaptable NoCs (FAN) 62

5.1 Introduction to NoC Customization 62

5.2 Methodology for Customization 64

5.2.1 Network-centric infrastructure 64

5.2.2 Application-specific Adaptation 70

5.3 Methodology Evaluation and Results 73

5.3.1 Synthetic applications 75

5.3.2 Simulation of real workloads 77
Chapter 6 - Design Space Exploration of NoCs Using Network Calculus 82
6.1 Background and Motivation 83
6.2 Network Calculus: An Overview 85
6.3 Proposed Methodology for Modeling NoC Architectures 90
6.4 NoC Performance Modeling: An Example 95
6.5 Validation of the Analytical Model 100
6.6 Analytical Performance Comparison of Network Topologies 103
Chapter 7 - Conclusions 106
7.1 Summary of Contributions 107
7.2 Future Work 110
Bibliography 114

List of Figures

Figure 1-1: SoC architecture main resources 2

Figure 1-2 Platform-based design: components, architecture reference and models and the augmentation of the platform with software and HW components 3

Figure 1-3 a) NoC-Mesh topology example b) Some traffic pattern of communication between routers [5] 5

Figure 2-1: Bus-based SoC architecture and main resources 13

Figure 2-2: Example of NoC topologies [6] 15

Figure 2-3: Example of router architecture with four input/output ports in 2-D Mesh NOC [20] 16

Figure 2-4: NoC 5-layer model compared with the standard OSI 7-layers model. Two layers, namely the presentation layer and session layer, are omitted in NoC model, whereas the other 5 layers have direct correspondence with each other [25] 17

Figure 2-5: Addressing scheme for 10x10 PEs in 2D Mesh NoC in network layer [5] .. 17

Figure 2-6: Express Virtual Channels (EVC), referred to by dotted lines: 00 is a sink or source for both x and y dimensions, node 13 is source or sink for x dimension, 04 for y dimension, 10 and 02 are bypass nodes along the y dimension [27] 19

Figure 3-1: Chip layout for: (a) Mesh, (b) Ring, (c) Spidergon, (d) Butterfly Fat Tree (BFT), and (e) Fat Tree (FT) [6] 27

Figure 3-2: Times contribute to the total latency between source and destination..... 30

Figure 3-3: Methodology for simulating real application workload (left) compared with synthetic workload generator (right) 32

Figure 3-4 : Visualization of communication traffic (a) FFT and (b) MG applications... 33

Figure 3-5: Latency vs. Buffer size in Spidergon with 16 nodes, 190 Mbps injection rate 37

Figure 3-6: Loss probability vs. Buffer size in Spidergon with 16 nodes, 190 Mbps injection rate..... 37

Figure 3-7: Communication load vs. Loss probability in Spidergon with 16 nodes 38

Figure 3-8: Comparing topologies for the maximum latency metric; (a) d_ℓ=0.1ms, Bw_{max}=200 Mbps and (b) d_ℓ=1μs, Bw_{max} = 4 Gbps..... 39

Figure 3-9: Comparison of different structures for different injection rate: (a) Communication load, (b) Throughput (Mbps), and (c) Loss probability 40

Figure 4-1: The structures of (a) WK (4, 0), (b) WK (4, 1), and (c) WK(4, 2) 47

Figure 4-2: WK-Recursive Topology for K=5 for (a) level 1, (b) level 2 and (c) level 3 47

Figure 4-3: Router structure in WK(4,2)-recursive on-chip interconnect 48

Figure 4-4: Maximum latency for topologies using: (a) heavy traffic, (b) light traffic... 49

Figure 4-5: Loss probability vs. injection rate for three NoCs with 16 nodes..... 50

Figure 4-6: Throughput vs. injection rate for three topologies, # nodes=16 50

Figure 4-7: Variation of communication load vs. injection rate..... 51

Figure 4-8: Chip layout of NoC Topologies with 16 nodes: (a) WK (4,2), (b) 2D Mesh 52

Figure 4-9: Comparison of the average energy consumption using CBR traffic for NoCs with 16 nodes..... 53

Figure 4-10: Comparison of the average number of hops using CBR..... 54

Figure 4-11: Comparison of the average energy consumption using CBR under 5Mb/s (flit drop=0)..... 55

Figure 4-12: Average area comparison of Mesh, Spidergon, FT and WK by varying the buffer size: (a) flits size equal 8 bytes, and (b) flits size equal 16 bytes 56

Figure 4-13: Comparing diameters of 2D Mesh D_{Mesh} and WK-recursive network D_{WK} for varying # Nodes. Clearly, WK-recursive scales better than 2D Mesh..... 58

Figure 4-14: Comparison of average Latency of three NoCs for varying # nodes N..... 59

Figure 4-15: Comparison of energy consumption of three NoCs for varying #nodes N.. 60

Figure 4-16: Comparison of performance per Watt of three NoCs for varying # nodes N 60

Figure 5-1: Illustration of augmentation of WK-recursive topology, to obtain AWKR .. 65

Figure 5-2: Modified router design..... 67

Figure 5-3 : Direct links infrastructure, q=3 (a) WK and (b) Mesh..... 74

Figure 5-4: Latency comparison for two versions of application traffics, n=16 nodes 76

Figure 5-5: Energy comparison for two versions of application traffics 76

Figure 5-6: Two link activation options for q = 4, WK topology. Legend used is same as in Figure 5-1..... 78

Figure 5-7: Comparing two applications domain on FAN, Class A and class B: (a) Energy, (b) Latency, and (c) communication load 80

Figure 6-1: Arrival and service curves in Network Calculus with delay and backlog bounds..... 85

Figure 6-2: Illustration of (a) the concepts of backlog and virtual delay and (b) the bounds on delay and backlog based on arrival and service curves..... 87

Figure 6-3: Linearization of a network subsystem by using the service curve β. The output of the linear system, Rlin(t), is a lower bound on the output traffic flow R*(t). Figure adapted from [47]. 89

Figure 6-4: Illustration of a single traffic flow in a NoC [72] 91

Figure 6-5: The input arrival curve α(t), service curve β(t) and output arrival curve α*(t) for a NoC router. The backlog bound and delay bound are shown. 91

Figure 6-6: A router handling two input flows. 93

Figure 6-7: On-chip interconnects with data flows: (a) 2D Mesh, (b) Spidergon, (c) WK recursive 95

Figure 6-8: Notation used for denoting input, output and aggregate arrival curves 97

Figure 6-9: Expression for maximum delay and buffer requirements for the routers participating in the traffic flows depicted in Figure 6-7(a) for 2D Mesh. 99

Figure 6-10: Analytical results for (a) Spidergon and (b) WK-recursive topology - Expressions for maximum router delay and buffer requirements for traffic flows depicted in Figure 6-7 100

Figure 6-11: The maximum e2e delay, using simulation (NS) and Network Calculus (NC) for 2D Mesh 102

Figure 6-12: The buffer size evaluated using Network Calculus (NC) and measured using simulation (NS) 102

Figure 6-13: The maximum latency for the Mesh, Spidergon and WK interconnects, using Network Calculus, for 16 nodes. 104

Figure 6-14: The maximum buffer size required for three topologies using Network Calculus, 16 nodes. 104

List of Tables

Table 3-1: On-chip interconnects parameters 27

Table 3-2 : Traffic parameters for CBR, Exponential, and Pareto generators 35

Table 4-1 : Comparison of NoC with other network types 42

Table 5-1: On-chip interconnects parameters 75

Table 5-2. Comparing average #Hops for two real applications on 2D Mesh 78

Table 5-3: Average #Hops and Energy for two AWKR configurations, q=4 79

Table 6-1: Summary of the notations used for the analytical model based on Network Calculus 86

Table 6-2: Parameters used for the analytical model 101

List of Acronyms

PDA	Personal Digital Assistant
SoC	System on Chip
IP	Intellectual Property
DSP	Digital Signal Processor
I/O	Input/Output
QoS	Quality of Service
NoC	Network on Chip
FPGA	Field-Programmable Gate Array
NAS	Numerical Aerodynamic Simulation
DSE	Design Space Exploration
OCI	On-Chip Interconnects
CMOS	Complementary Metal Oxide Semiconductor
NS-2	Network Simulator ver. 2
FIFO	First In First Out
UDP	User Datagram Protocol
FT	Fat Tree
BFT	Butterfly-Fat Tree
CLICHE	Chip-level Integration of Communicating Heterogeneous Element
PE	Processing Element
CBR	Constant Bit Rate
FAN	Field Adaptable NoC
AWKR	Adaptable WK- Recursive
AMesh	Adaptable Mesh
VLSI	Very Large Scale Integration
ASIP	Application-Specific Instruction-set Processor

Chapter 1 - Introduction

The Network on Chip (NoC) paradigm has been a recent development in System-on-Chip (SoC) design, fueled by the advances in VLSI technology which has lead to very high levels of integration of hundreds of interacting components within a single chip. A background on the evolution of NoCs will be first provided here, along with a description of the limitations and the motivation for the research work described in this thesis. This will be followed by a formal statement of the research problem, an overview of the proposed methodology, as well as the scope and limitations of this work. The chapter will then end with a summary of the organization of this thesis, thereby setting the stage for a detailed description in the remaining chapters.

1.1 Background

Embedded systems have become widespread in various application domains, as evident from their use in products such as PDAs, household appliances, automobiles. This trend has been further accelerated by the advances in silicon technology, which has enabled the design of complex SoCs that incorporate many hardware blocks and Intellectual Property (IP) cores. In other words, SoCs nowadays integrate different types of processors such as CPUs and DSPs, in addition to memories and peripherals in a heterogeneous configuration, along with dynamically reconfigurable components such as embedded FPGAs within a single chip, as shown in Figure 1-1.

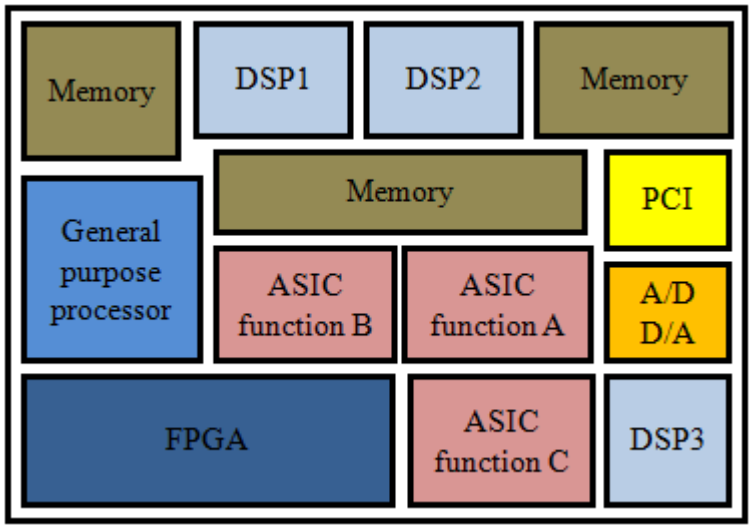


Figure 1-1: SoC architecture main resources

Due to their complexity, the design of SoCs has become very challenging. In order to manage the complexity, platform-based design methodology was introduced in the recent years.

1.1.1 Platform-based design methodology

Platform-based design is an integration-oriented design approach for developing products based on the use of platforms and compatible hardware and software components [1]. A library of qualified software and hardware components is used to derive a new product from an application specific pre-verified platform. Figure 1-2 shows the generic SoC components, a "platform reference" that integrates a selected subset of these components for an application domain using a generic architecture, and the platform customization model that is used for modifying and augmenting the software and hardware components.

Platform-based design aims to improve/reduce product cycle time and development costs by providing a standard "platform" which a system designer can readily modify to

produce several distinct and usable end products. Typically the platform is composed of a base block, an IP library, an expandable architecture and an integration environment. The designer is given the responsibility of thinking through the augmented architecture very carefully to promote flexibility so that effective re-use of IP is realized.

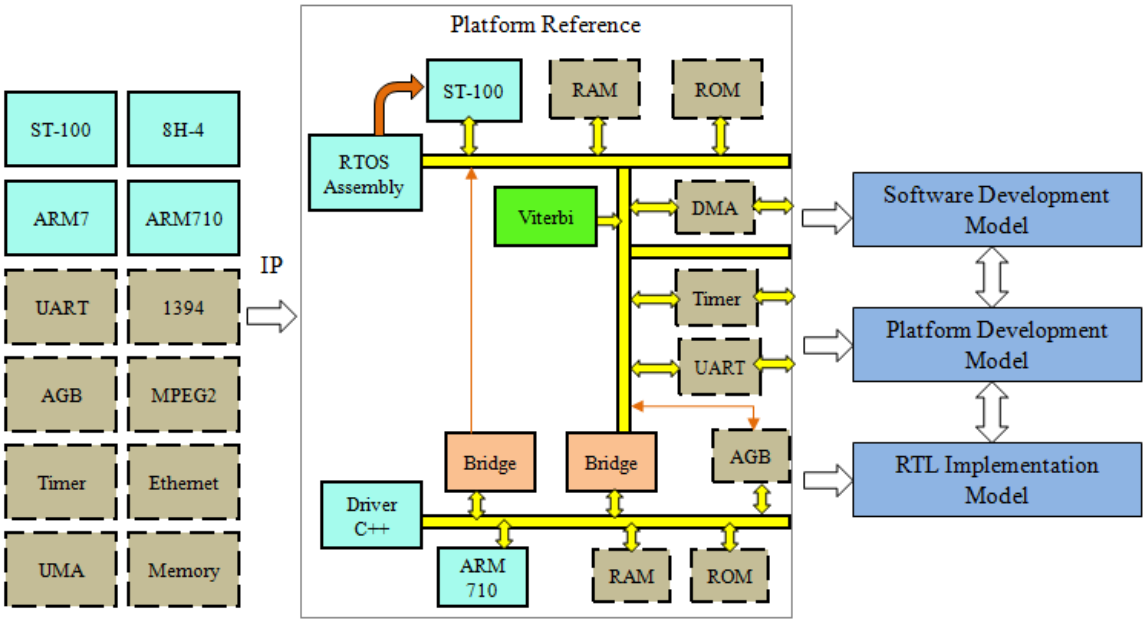


Figure 1-2 Platform-based design: components, architecture reference and models and the augmentation of the platform with software and HW components

The base blocks shown in the left with solid border line comprise the fixed portion in the platform. The fixed part is re-used and combined with different IP modules, shown in lower blocks, to produce various derivative products, whereas the IP library is a collection of "soft" and "hard" modules that should be verified. These features enable the high re-use capability of the platform-based design approach, and ultimately lead to lower development costs and faster time-to-market for products. The derivative products adopt the same processor core and communication bus, after which various IP modules can be selected from the IP library, giving rise to a very versatile design with numerous options for its final implementation.

1.1.2 Evolution of Network-on-Chip

Through a methodic approach to SoC design, the platform-based approach helps in managing the system complexity, while lowering the efforts involved in software and hardware development. However, the need for high performance is not adequately addressed by the initial model as depicted in Figure 1-2. To elaborate, the use of a generic, shared-bus scheme for integrating SoC components can adversely impact the system performance. With the emergence of new applications, such as video and audio running on mobile, battery-powered devices, the bus-based schemes become limited because they are inherently non-scalable and incur substantial overheads in performance.

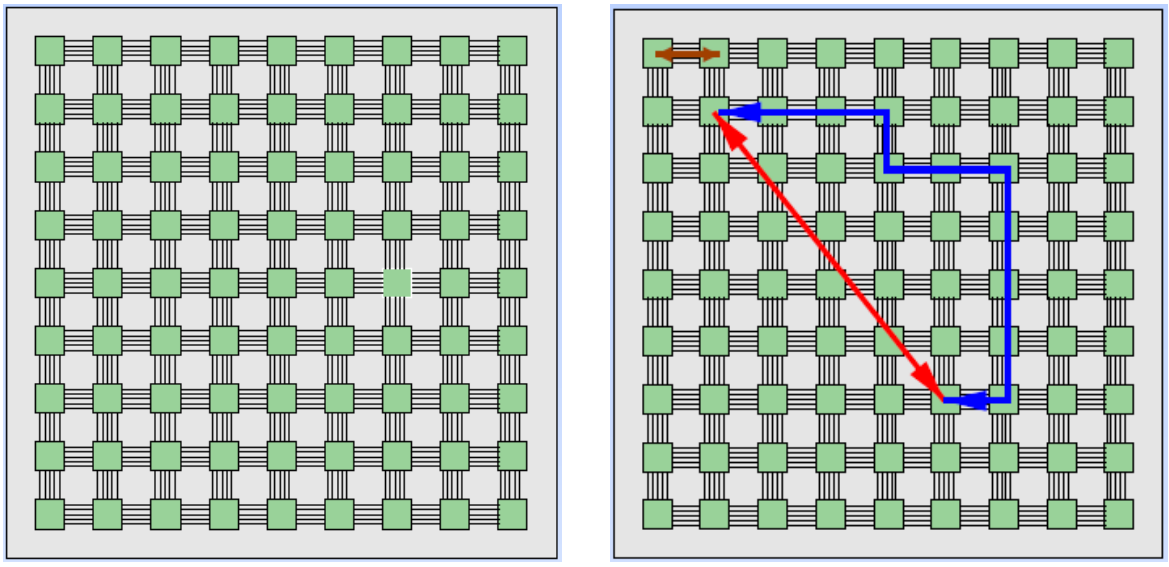
To satisfy quality of service (QoS) requirements of users for these emerging applications, and also reduce the communication overhead, the network on chip (NoC) paradigm has emerged as a solution for the interconnection problem in large SoCs, after being introduced at the beginning of this decade [2,3].

The idea behind employing NoC is to address the communication issue by separating the communication from the computation. A sophisticated communication infrastructure is better suited to handle the increasing communication requirements of large SoCs. The communication in a NoC is handled by on-chip interconnects, so scalability, reliability and the performance can be improved [4].

Extending the platform-based design methodology with the NoC paradigm introduces some additional steps into the design process. These involve (a) choice of a suitable network topology for the on-chip interconnection network, (b) mapping the applications to the hardware resources communicating through the chosen on-chip interconnect, and (c) managing the physical effects of the VLSI implementation, such as cross-talk and hot-

spots. The research described here focuses on (a), the NoC architecture design and selection; (b) and (c) are beyond the scope of this research.

NoC architecture is mainly characterized by the interconnect topology, routers that are attached to different types of resources, and the data exchange and communication between these routers using standard network layers. Figure 1-3 (a) shows large number of SoC resources connected using a Mesh topology. Each of the nodes includes a router and a computational resource. Different types of interconnection topologies are possible, and a suitable network must be chosen based on the applications that need to be mapped to the SoC. Different applications will have varying communication requirements. Figure 1-3 (b) shows one of the possible communication patterns for an application, across various computational nodes.



(a) (b)

Figure 1-3 a) NoC-Mesh topology example b) Some traffic pattern of communication between routers [5]

1.2 Motivations

Although the use of a network-on-chip offers the potential for significant performance gains, there are many challenges that need to be addressed: (a) finding a suitable choice of network topology for implementation on a single chip embedded platform with performance and energy constraints, (b) accommodating wide range of requirements for the different SoC applications and (c) establishing fast design space exploration required for designing the system that addresses (a) and (b).

(a) Network Topology

The network topology can significantly impact the performance of the NoC for different applications that run on the embedded platform. The trade-offs between performance, energy, and area makes the selection of an appropriate topology for a NoC an important issue [6]. For instance, a topology in which the node degree is high, or increases as the network size increases, can significantly improve the performance of the network. However, designing routers for such network is complex and expensive. In addition, the choice of the NoC features and parameters, based on the application traffic patterns as shown in Figure 1-3 (b), is critical for the communication architecture [2]. Thus, a comprehensive study of the various network topologies is essential, before choosing one of them for implementation on a single chip.

(b) Wide Range of Application Requirements

Another important issue in NoC selection is the ability of the platform to target a wide range of applications while maintaining efficiency. The design objectives are to

satisfy the quality of service requirements, optimize the resource usage, and minimize the energy consumption by separating communication from computation. For a single application, this may be achieved by carrying out suitable application-specific optimizations of the interconnection network. However, the static topology of the interconnection can be a major impediment when multiple applications are to be mapped to the embedded platform. To address the wide range of requirements of many applications, the NoC topology needs to be adaptable. In other words, starting from the selected network topology in (a) above, adaptable features need to be incorporated, along with mechanism(s) for in-field adaptation before the execution of any application.

(c) Fast Design Space Exploration

Meeting the challenges in (a) and (b) is effectively a design space exploration problem. More precisely, Design Space Exploration (DSE) is the process of searching the design space to find design configurations that, in addition to meeting the specification requirements, satisfy system constrains and achieve best objective functions for criteria such as performance, power, cost, and area minimization. Simulation based approaches for evaluating various points in the design space can be very time consuming, especially for large SoCs with many components. A mathematical abstraction of the network characteristics is essential in order to allow a fast evaluation of different network configurations. Rapid DSE therefore requires an analytical model of different network parameters, capturing both the characteristics of the network topology as well as the application.

1.3 Problem Statement

The efficient mapping between application domains and architecture resources is important for successful SoC design. Certain resources offer some flexibility that the designer can utilize to target application requirements while adhering to basic design requirements such as time to market.

We consider a SoC environment that includes fairly large number of hardware computation resources such as general processors, DSPs and FPGAs, in addition to the interconnection infrastructure that deals with the communications events among these resources. Given such environment, it is then required to:

- Investigate the suitability of different network structures for NoCs and propose efficient one(s)
- Propose an adaptable on-chip interconnect infrastructure that optimizes performance while meeting the system constraints such as area and power for target applications
- Develop models to estimate the most important metrics in NoC such as throughput, communication load, and energy
- Develop analytical models to accelerate the design space exploration by estimating the key metrics in NoC in early stages of the design process

1.4 Proposed Approach

To identify a suitable topology for NoC applications, several interconnection structures are empirically compared based on simulation studies. In addition to networks used in SoC designs, networks used in parallel machines and other domains are also

evaluated. Comparisons are based on critical performance metrics such as latency, throughput, and communication load using a discrete event simulator. Energy and area models for on-chip interconnect reported in the literature are adopted and used for augmenting the basic set of performance metrics, resulting in a thorough comparison of the various NoC architectures. The scalability for NoC structures is one of the main features that need to be evaluated. Scalability study is conducted by comparing different NoCs when number of nodes is increased. The detailed evaluation of the various NoC architectures leads me to propose [7,8] the use of recursive networks [9] for the on-chip interconnection network. Based on our results, the WK-recursive networks were found particularly suited for NoC implementations [10,11].

With the WK-recursive network taken as the base topology, the addition of features that would lend themselves to application-specific adaptations is explored. A general methodology is then developed for the design and use of adaptable network topologies. The adaptable features are first incorporated in the form of additional, programmable links to the underlying structure, based on reducing the average distance from a purely topological standpoint. With this augmented structure, application-specific adaptation algorithms are developed in order to statically reconfigure the network based on application requirements.

In order to obtain realistic estimates of the performance improvements achieved by using the adaptable features, the workloads and communication patterns from real-life applications are characterized. An emulation environment based on real applications running on parallel machines is developed, in order to mimic the behavior of those applications in a NoC environment.

Finally, based on the insights obtained from the simulation studies using synthetic workloads and emulation of real applications, an analytical model is developed for aiding rapid design-space exploration. The model (unlike other probabilistic techniques such as queuing theory) is based on Network Calculus, which captures the traffic patterns generated by different applications, and allows the designer to specify a system as a mathematical model and evaluate performance bounds of important parameters such as the end-to-end delay.

1.5 Scope and Limitations

A SoC architecture is usually composed of a number of computational resources that can be fixed or reconfigurable. In simulation, we assume that all hardware resources are fixed. Additionally, the computational resources are viewed as black boxes and modeled as delay elements, all the effects that come from their memories or caches if shared or distributed between cores are not considered in this research.

There is a NoC connecting all resources with bidirectional links. The NoC can be statically or dynamically adaptable. In this thesis, the target architecture is the statically adaptable NoC. As such, in-field adaptation of the NoC is considered here before the execution of an application. It is different than dynamic adaptation since the decision to turn ON and OFF some switches to activate/deactivate some links to benefit a target application within the application domain is done before running and is instrumented (embedded) into the application code. Dynamic adaptation at application run-time is beyond the scope of this research.

Workloads are considered to run long enough to capture consistent statistical characteristics. Workloads are also assumed from some application domains, where the workloads within one application domain exhibit somewhat similar characteristics.

1.6 Organization of the Dissertation

This dissertation is organized as follows. In Chapter 2, the related work in the context of SoC research and the NoC challenges is discussed with emphasis on architectures, performance and energy studies. Chapter 3 details the methodology used for evaluating NoC architectures by introducing the models for performance as well as the metrics used for comparison. The experimental setup and detailed experimental results which conclude with the best performing NoCs are given at the end of the chapter. Chapter 4 introduces the energy and area models, then the WK-recursive networks are presented and compared with the best performing NoC from the previous chapter; the energy and area metrics are used to demonstrate that WK–recursive is a suitable network topology for NoC. Chapter 5 describes the Field Adaptable NoCs methodology used for augmenting a given NoC topology in order to incorporate the features of adaptability. Using the WK-recursive topology as a case study, the chapter describes the resulting architecture termed as adaptable WK-recursive (AWKR). From the experimental results, the improvement of adapting these structures is demonstrated and compared with regular WK-recursive topology. 2D Mesh topology is also augmented using the same technique and experimental results are shown to verify the validity of the methodology for different topologies. Chapter 6 introduces Network Calculus, which is used here as an analytical technique for evaluating NoC. Network Calculus is then compared against the simulation

to show its main advantages when used for design space exploration. Chapter 7 concludes this thesis, by summarizing the main contribution of this work, and includes suggestions for future work.

Chapter 2 -Related Work

On-chip interconnect architectures, used in current SoCs to integrate hardware resources or cores, are based on shared-medium bus-based schemes such as the one depicted in Figure 2-1. Because of the growing complexity of SoCs, and high application demand for computation and communication, the bus-based schemes become limited because they are inherently non-scalable and produce a large communication overhead that affects the performance and can potentially increase the energy consumption; for instance, common bus systems limit the maximum number of nodes to 16 PEs. Different techniques are proposed to improve the bus systems either by expansion through bridges, or using different coding methods to reduce the amount of transferred bits such as in [12,13,14,15]. But these approaches have not completely solved the scalability problem. New on-chip interconnects that are flexible as well as energy efficient are required. To meet these requirements, NoCs have been proposed [3].

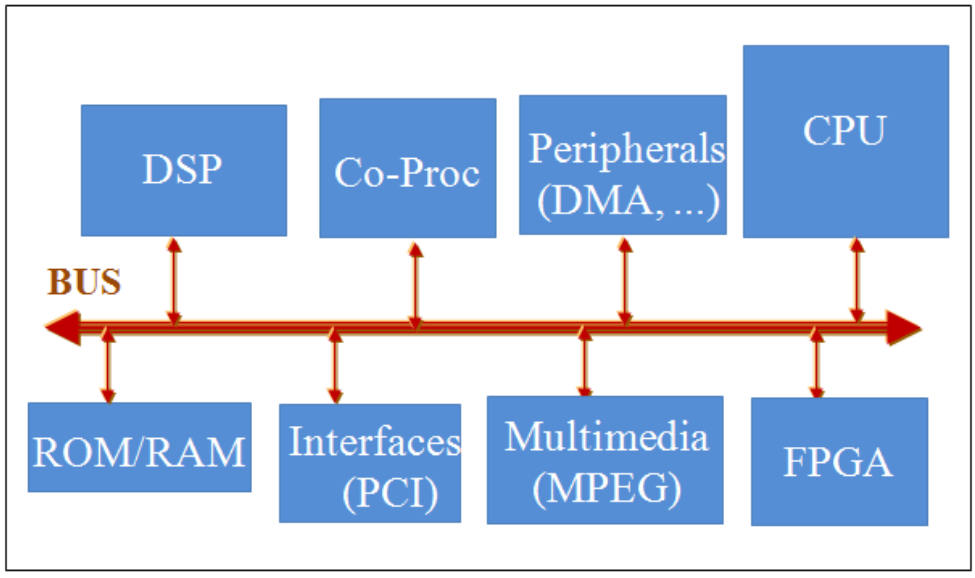


Figure 2-1: Bus-based SoC architecture and main resources

2.1 NoC Architecture

NoC architecture is mainly characterized by the topology, the switches or routers that are attached to the resources, regions which include resources that are arranged in a physical proximity in a groups which provide isolation between different types and sizes of resources, and finally by the communication techniques used to allow data exchange between these resources. In this section, we will explore the existing research work on each of these properties.

2.1.1 Topology studies

Different on-chip interconnect architectures based on packet-switching have been studied and adapted for SoCs. The on-chip interconnect architecture specifies the structure in which routers and PEs [*] are connected together. Examples of these architectures are Fat-Tree (FT) [6], 2D Mesh [15], Ring [16], Butterfly-Fat Tree (BFT) [17], Torus [3], Spidergon [18], and Octagon [19]. Figure 2-2 shows some of these on-chip interconnection networks for the particular case of 16 PEs and 16 switches or routers.

NoC architectures have been studied and compared in terms of performance metrics, resource usage and energy consumption as factors in selecting NoC attributes such as topology, routing strategy and application mapping to PEs.

[*] PE(Processing Elements) and IP (Intellectual Property) refer to hardware resources in Figure 1-1, both will be used interchangeably throughout this thesis

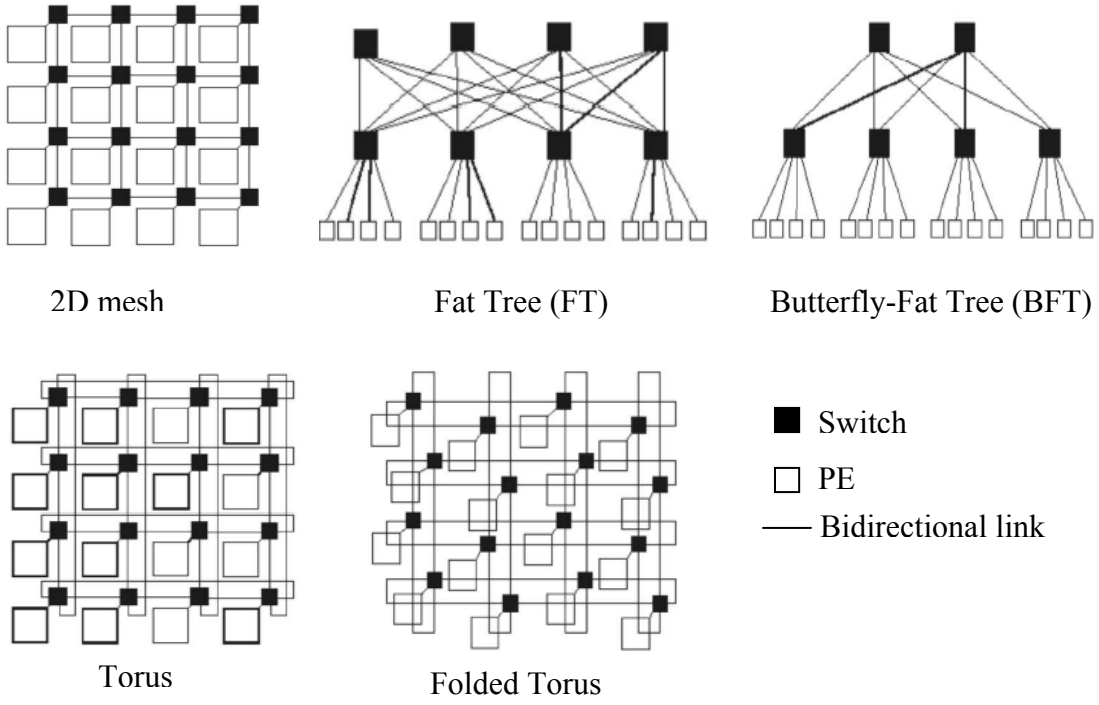


Figure 2-2: Example of NoC topologies [6]

2.1.2 NoC router

Router is the main active component of a NoC architecture. Each router is attached to a hardware resource, see Figure 2-2, and all communication activities are conducted with the involvement of routers. The design of the routers is critical since routers introduce a non-negligible area overhead and increase the average communication latency [20]. Therefore, it is important to reduce the area occupied by the router to a minimum while keeping performance and energy consumption at acceptable levels. The common logic blocks that can be found in the router are buffers, switches and arbitration logic. Figure 2-3 shows the logic circuit for an area optimized router in which authors in [20] claim to reduce the area by reducing the buffer size.

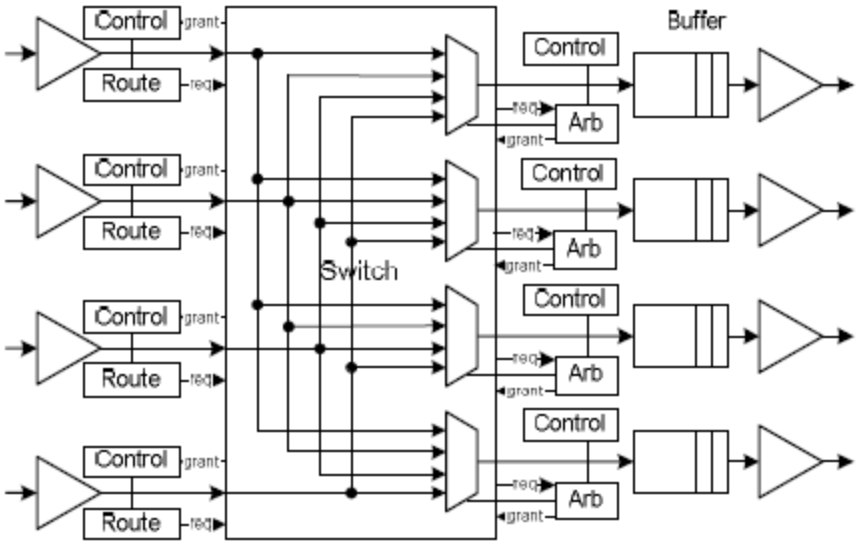


Figure 2-3: Example of router architecture with four input/output ports in 2D Mesh NOC [20]

In [21], authors presented an algorithm that minimizes the energy consumption and reduce delay by reducing the size of input buffers. On the other hand, authors in [22,23] developed different algorithms that focus on maximizing the timing performance. In [24], authors suggested three light routers and an algorithm is developed to select the required router based on the application constraints and target performance. Another approach that allows the configurability is presented in [20]. It allows customization by sharing some of its resources to improve the bandwidth. In Chapter 5 we will discuss the customized router design that is used for our adaptable NoC architecture.

2.1.3 Communication infrastructure

NoC architecture ensures acceptable levels of quality of service for the communication between the router and resources. The communication activities are performed on different layers following the standard reference model of layers. A comparison between the standard OSI 7-layer model and NOC 5 layer-model is depicted in Figure 2-4.

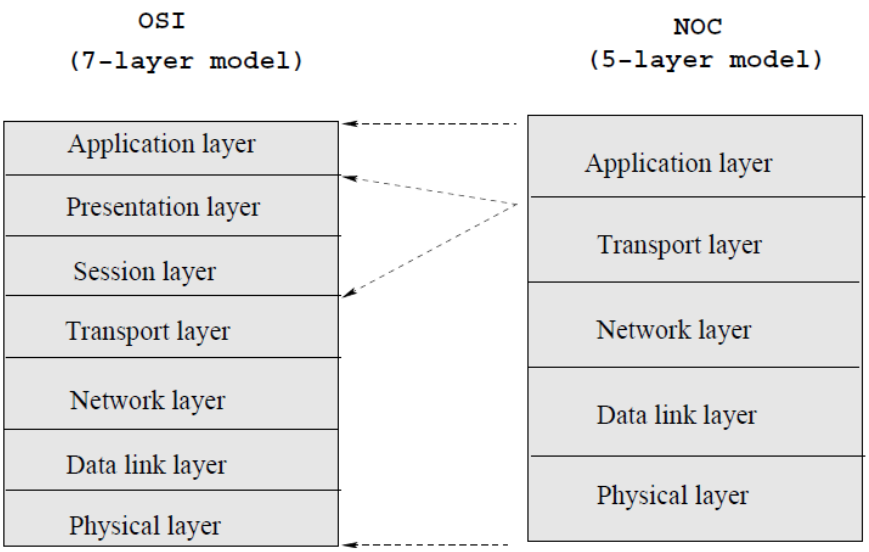


Figure 2-4: NoC 5-layer model compared with the standard OSI 7-layers model. Two layers, namely the presentation layer and session layer, are omitted in NoC model, whereas the other 5 layers have direct correspondence with each other [25]

Most research efforts focus on the physical layer and network layer. The physical layer includes parameters such as the physical distance between routers, number of lines or wires, logical circuit control and the buffer and pipelining scheme used in communication. The network layer parameters include packet/flit size, routing algorithm, and addressing scheme based on number of resources as shown in Figure 2-5.

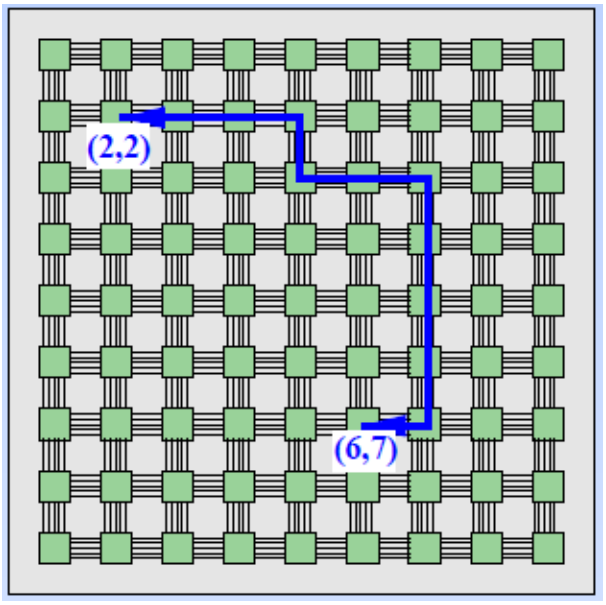


Figure 2-5: Addressing scheme for 10x10 PEs in 2D Mesh NoC in network layer [5]

In the network layer, the research focused mainly on the routing algorithms. For example, in [26] the authors developed a simulator to evaluate different routing algorithms, selecting the best for target applications.

2.1.4 Adaptable NOC architecture

Network-on-Chip is built based on state of the art packet switched network to replace non-scalable bus-based system. However, the complex router design which is placed on each hop of the network creates major overhead that reduces the aggregate throughput and increases the delay and energy. To address this issue, NoC architecture should be modified to provide very low delay and energy and increase the throughput. The solution can be achieved by following three directions that were discussed in the literature and also proposed in chapter 5.

The first approach is to search the best location and insert the minimum number of long range physical links that bypass the hops and the complex routers. The static insertion targets specific applications [57]. The drawback of this technique is the additional wires that add more area and complicate the network. Moreover, the chip design is dedicated to a single application.

In the second approach, the bandwidth of the physical link is divided into two halves, one will be available for packets that pass regular link and routers, and the second half is designated for packets that bypass the routers. The fact that no physical links are added and the simplicity of the router design represent the main benefits of this approach, subsequently, the latency and energy are reduced, but the aggregate throughput is decreased because of the bandwidth reduction [72].

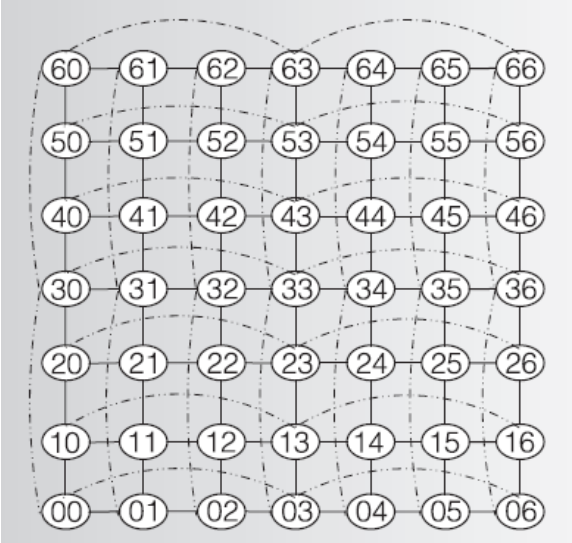


Figure 2-6: Express Virtual Channels (EVC), referred to by dotted lines: 00 is a sink or source for both x and y dimensions, node 13 is source or sink for x dimension, 04 for y dimension, 10 and 02 are bypass nodes along the y dimension [27]

The third approach that has been discussed in the literature is based on Express Virtual Channels EVC [27]. In EVC network, the packets/flits are injected into the pipeline of routing stages in the router, the stages represent the decoding of the head flit, allocation of the virtual channel or allocate switch. The head flit is decoded to determine if the remaining flits will continue through the pipeline stages or go directly to the virtual channel. Only one cycle is required to bypass the router. The virtual channels designated on X and Y dimensions as shown in Figure 2-6. A distance of K hops (in this example 3 hops) is used, where the first and the last hops of this distance are sources and sinks, while the hops in between are the bypass nodes. The drawback of this approach is the large area incurred in modifying all the routers in the NoC.

2.2 Performance Studies

A simulation-based approach was proposed in [25] using the NS-2 simulator to analyze a NoC Mesh interconnection topology. It is based on the CLICHE (Chip-Level

Integration of Communicating Heterogeneous Element) [2]. In the Mesh, the number of nodes equals to number of routers, since each router is connected to one PE node. NS-2 is used to construct the topology and generate different traffic scenarios using random generator (synthetic generator) which generates packets according to an exponential ON/OFF distribution. Packets are sent at a fixed rate during ON periods, and no packets are sent during OFF periods. Both ON and OFF periods are taken from an exponential distribution. Packets also have a constant size. Using this traffic generator, common network performance metrics such as loss probability, packet delay, throughput and communication load are analyzed against different buffer sizes and traffic injection rates.

In [6], another simulation-based approach is used to compare six common topologies, Fat Tree (FT), Mesh, Butterfly Fat Tree (BFT), Octagon, Torus and Folded-Torus in terms of several performance metrics such as message latency and throughput. Pareto traffic is adopted to simulate the application using the wormhole switching technique. Pareto traffic is identical to the Exponential ON/OFF distribution except that the ON and OFF periods are taken from a Pareto distribution in which the aggregate traffic exhibits characteristics found in streaming applications, such as high degrees of self-similarity and long range dependency [28, 29].

Other investigations using the Mesh NoC have been conducted in [30] to verify the suitability of this structure for NoC, based on tradeoffs of network parameters. Metrics such as latency and packet loss probability were presented as a function of the communication load and the buffer size, using the NS-2 simulator. In [31], the authors compared the Ring, Irregular Mesh and Spidergon topologies using a discrete-driven simulator (OMNET++) based on the wormhole switching technique. Their analysis has

shown that the NoC based on Spidergon outperforms others in average latency and throughput. The type of traffic has not been mentioned despite its prime importance in NoC. In [32], an Application Specific NoC (ASNoC) design methodology was proposed, that is, using a customized topology to fit the requirements of specific applications. In that work, the OPNET simulator is used to compare the proposed structure with a Mesh topology, using a HDTV decoder SoC as an application example; the results show that a fully customized NoC outperforms the Mesh NoC. However, this approach has clear drawbacks such as low reusability and unpredictable physical parameters.

2.3 Energy and Area Studies

NoCs are typically part of an embedded system that requires deployment in the field and where energy is very valuable. On-chip interconnects are a dominant source of energy consumption and area requirement in current SoC designs [33, 34]. Prior work includes comparison of on-chip interconnect in terms of energy and area requirements. For example, in [6], a simulation methodology was proposed to compare the performance characteristics of a variety of on-chip interconnect architectures. Energy consumption and area requirements models have been adopted because of the prime importance of these metrics in SoC evaluation and design. Pareto generators were used to model traffic generated by sources. The reported results show that on-chip interconnect architectures with a higher degree of connectivity like 2D Mesh and Octagon have greater average energy dissipation at the saturation point than the other architectures, but they provide higher throughput and lower latency on the average. A circuit level design and synthesis method to compare the silicon area requirements for those on-chip interconnect architectures are also proposed and the results obtained based on their layout showed that

FT and Octagon architectures have a considerably higher silicon area overhead due to their higher degree of connectivity.

Design methodologies and tools to assist designers in developing high performance and energy efficient NoC architectures were proposed in [35]. The objective was to optimize certain metrics such as minimizing power and increasing the performance by customizing the given NoC platform for a specific application using an energy-aware scheduling algorithm. An energy model was also proposed and validated by simulation on a regular 2D Mesh. In [36], an energy model was proposed to estimate the energy consumption of on-chip architectures. This model is based on packet and circuit switching techniques to predict the energy consumption per transported bit for the routers. A regular 2D Mesh is compared with a traditional bus architecture using a model proposed in [37,38] to estimate the energy consumption of the wires between the PEs. The reported results show that the energy consumed to transport one bit in the bus-based architecture is higher than other architectures. In addition, the energy consumption increases quickly as the number of nodes increases in comparison with other architectures, so the benefit of using an on-chip interconnect architecture was clearly noticeable for large SoCs. In addition, the energy consumption when using a packet switching technique is slightly higher compared to the circuit switching technique. A model to evaluate the silicon area of the switching components inside the switch box was presented in [39]. This model is based on the equations given in [40,41] and reported results show that the buffer memory occupies the greatest part of the switch box.

2.4 Analytical Models for NoC

On-chip interconnect architectures adopted for SoCs are characterized by different trade-offs with regard to latency, throughput, communication load, energy consumption, and silicon area requirements. Several works, such as that presented in [42], have demonstrated that there is a crucial need for system design tools and methodologies to analytically evaluate and compare NoC architectures for a given application.

Researchers have pointed out that current design methodologies are unable to provide the required frameworks and simulation methods. Despite their accuracy, they are very expensive and time consuming [42]. Techniques and tools are required to extract application communication requirements to efficiently estimate their performance for different candidate communication architectures.

Recently, there has been a great deal of interest in the development of analytical performance models for NoC design. In [34], an analytical model using queuing theory was introduced to evaluate the traffic behavior in Spidergon NoC. Simulation results to verify the model for message latency under different traffic rates and variable message lengths have been reported. In [43], a queuing-theory based model for evaluating the average latency and energy consumption of on-chip interconnects was proposed. The results from the analytical model were validated with those obtained from a cycle-accurate simulator.

In [44], a generalized router model, using a probabilistic approach, was proposed for NoC performance analysis and application mapping for an on-chip interconnect. In [45], a model using a cyclo-static dataflow graph was used for buffer dimensioning in NoC applications. The results derived from the model were compared with those extracted

from simulations. It should be noted that most analytical models consider incoming and outgoing traffic as a probability distribution (e.g., Poisson traffic). This allows designers to perform a statistical analysis on the whole system in order to evaluate the network performance metrics such as average buffer occupancy and average buffer delay. These are, however, steady-state analyses that fail to capture the dynamic characteristics of the network traffic.

In [11], a performance analysis methodology using Network Calculus was presented and used to analyze and evaluate the performance of 2D Mesh topology for on-chip interconnect. Simulations were performed and results were compared with those produced from Network Calculus based model to emphasize its usefulness for evaluating on-chip interconnects.

Chapter 3 - Exploring NoCs: Methodology and Performance Evaluations

In general, NoC is inspired by the interconnection networks in high performance parallel computers and computer network design and operation. While parallel computer architectures generally have a low degree of heterogeneity and are designed for general purpose applications, NoC connects heterogeneous processing elements (e.g., CPU, DSP, memory, etc) in a SoC that is usually tailored for a specific application. Additionally, meeting energy constraints is of prime importance in NoC since applications run on battery operated devices.

The major difference between NoC and a computer network is that NoCs have defined topologies (e.g., Mesh, Ring, Fat-tree, etc), require energy-aware switching techniques, and they should occupy a small area on the chip. Since the on-chip interconnect topology is the key to determining the main performance metrics of the NoC, basic performance, energy and area models are adapted and used to compare and report experimental results for five common NoC topologies: 2D Mesh, Ring, Spidergon, FT and BFT.

Performance metrics are explored for considered topologies. A simulation environment is developed [25] based on NS-2 simulator, and extended to build different topologies and incorporate new parameters. NS-2 simulator is an object-oriented and discrete-event driven network simulator developed at UC Berkeley and written in C++ and OTCL [37]. The objective of using NS-2 is to rapidly explore the design space by observing the performance of on-chip networks while using its facilities to vary the

network topology, transmission protocols, routing algorithms, and traffic generation (e.g., Exponential, Pareto, etc).

3.1 Structures and Topological models

In this section, parameters related to the on-chip interconnect and the traffic models used in NoC analysis are presented. Resources can be modeled as either sources or sinks, where packets are produced by source PEs and consumed by sink PEs.

Additionally, there are links that interconnect these resources. As shown in Figure 3-1, there are two types of links, router-to-PE and router-to-router. Two types of parameters are considered to model each link: link delay (d_ℓ) and maximum bandwidth or the maximum transmission speed permitted for each link ($BW_{\max}$). The expected bandwidth for the NoC communication link is in the order of Gigabits/s [49]. However, due to limitations in the resolution of the NS-2 simulator, as well motivated by power limitations in embedded systems, we choose relatively slower parameters for the NoC - 200 Mbps for the interconnection link bandwidth, and 0.1 ms for the constant link delay. We demonstrate that the use of these slower parameters, when we present the results in Section 3.5, does not result in any loss of generality and applicability to higher bandwidth and lower latency networks as the trend remains the same.

We also consider that the buffer size inside each PE is unlimited since it has enough space and speed to handle and process the arriving data, thereby preventing packets or flits* from dropping.

*Wormhole is a routing technique in which packets are divided into flits (short for "flow control digit") in order to pipeline their transmission across multiple nodes. They usually have a smaller size; more information can be found in [34]

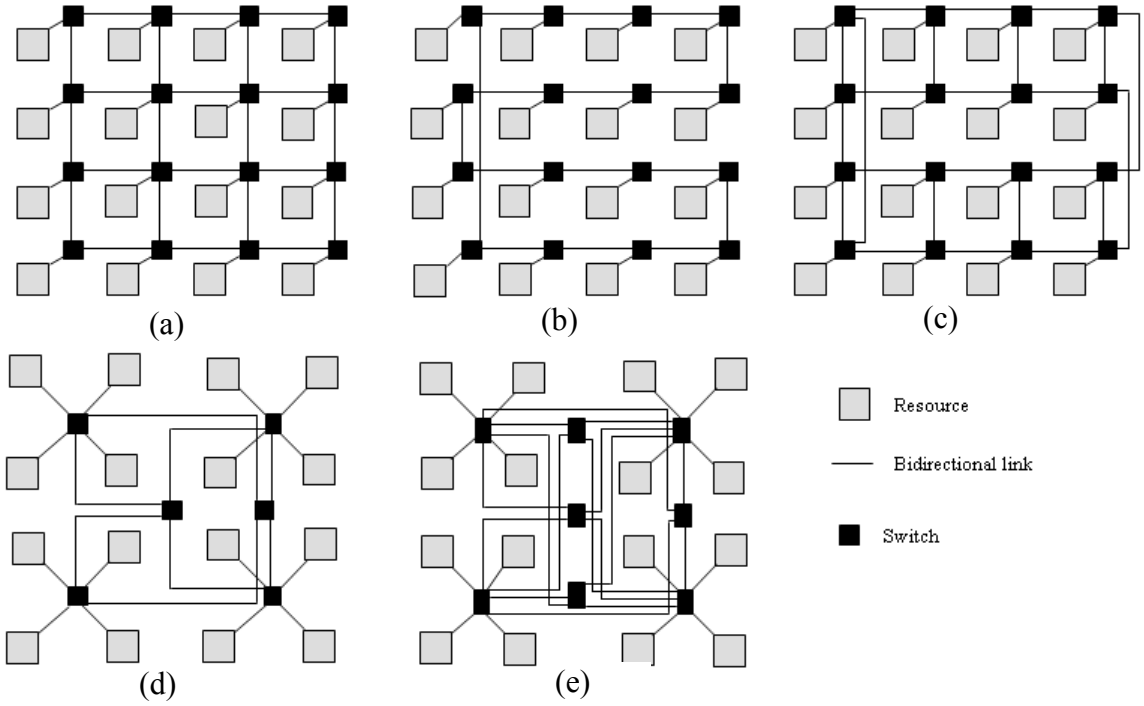


Figure 3-1: Chip layout for: (a) Mesh, (b) Ring, (c) Spidergon, (d) Butterfly Fat Tree (BFT), and (e) Fat Tree (FT) [6]

In the routers, each port has a buffer for temporary storage of received flits. When a flit arrives at a router, it must go into the buffer. The buffer type is similar to a Drop-tail queue with a FIFO queue management mechanism. In this type of queue, when the buffer capacity of the queue is exceeded, the next flit to arrive will be dropped. Table 3-1 summarizes the parameters used for the on-chip interconnects.

Table 3-1: On-chip interconnects parameters

On-Chip Interconnects (OCI)	2D Mesh, Spidergon, FT, BFT, Ring,
Number of PEs	16
Number of routers	Depends on the topology
Number of links	Depends on the topology
Link delay (d_ℓ)	0.1ms
Maximal link capacity ($Bw_{\max}$).	200Mbps
Buffer size inside PEs	Unlimited
Buffer size inside routers	2, 4, 8, 16, 32 flits
Type of the buffer size	Drop-tail queue

A wormhole switching technique should be used to minimize the dropping of flits, reduce the buffer size requirements, and to also achieve a low latency. A way to use

wormhole technique instead of the store-and-forward technique in the NS-2 simulator is described in [10]. The authors in [10] have justified the use of NS-2 by comparing a cycle accurate simulation developed using System-C with the NS-2 simulator, and proved that they have the same behavior (i.e., results obtained are in the same order of magnitude). To direct flits between routers, a deterministic routing method, in which all routes are predetermined based on shortest path, is used. Unlike dynamic routing protocols that require the implementation of a complex logic at the switch level, deterministic routing protocols are usually used because they can be implemented using simple logic [34].

3.2 The Performance Model

The main performance metrics that are used in the simulation study are communication load, throughput, flit loss probability and latency. The expressions that are used to calculate these metrics are as follows [25,50]:

Communication load

The communication load is the fraction of the total network capacity that is busy during the execution of the application. In other words, the communication load L is

$$L = \frac{N_{flits-actual}}{N_{flits-peak}} \tag{1}$$

where $N_{flits\text{-}actual}$ is the actual number of flits that have arrived over all links during the application execution and $N_{flits\text{-}peak}$ is the maximum number of flits that can possibly be transmitted over all links during the same duration. The value of $N_{flits\text{-}peak}$ is calculated based on the total time taken for executing the application, T_{total} (seconds), the maximum

bandwidth of each link $Bw_{\max}$ (bits/second), the size of each flit S_{flit} (bits/flit), and the number of bi-directional links N_l, as follows:

$$N_{flits-peak} = \left(\frac{T_{total}}{S_{flit} / Bw_{\max}} \right) N_l \tag{2}$$

Clearly, the value of L is between 0 and 1.

Throughput

The aggregate throughput for an application is a measure of the total number of bits that have arrived per second at all the PEs in the network. If T_i denotes the throughput for a single PE (PE*i*), the aggregate throughput T (bits/second) can be calculated as:

$$T = \sum_{i=1}^{Nr} T_i \tag{3}$$

where N_r is the number of resources or PEs in the network.

Flit Loss Probability

Flit loss probability p_{loss} is the probability of flits getting dropped, i.e., flits not forwarded due to a lack of buffer space in the routers. The simulator counts the number of transmitted packets and reports the number of packets that are dropped. The metric is defined as follows:

$$p_{loss} = \frac{N_{flits-dropped}}{N_{flits-sent}} \tag{4}$$

where $N_{flits\text{-}dropped}$ represents the number of flits dropped during the entire execution of the application and $N_{flits\text{-}sent}$ is the number of flits sent from all source PEs during the same duration.

Latency

Latency is defined as the time that elapses between injection of a flit into the network at the source PE and its arrival at the destination PE. It includes buffering, queuing, retransmission and propagation delays. Figure 3-2 shows the components that contribute to the total latency of the flit when it traverses the network.

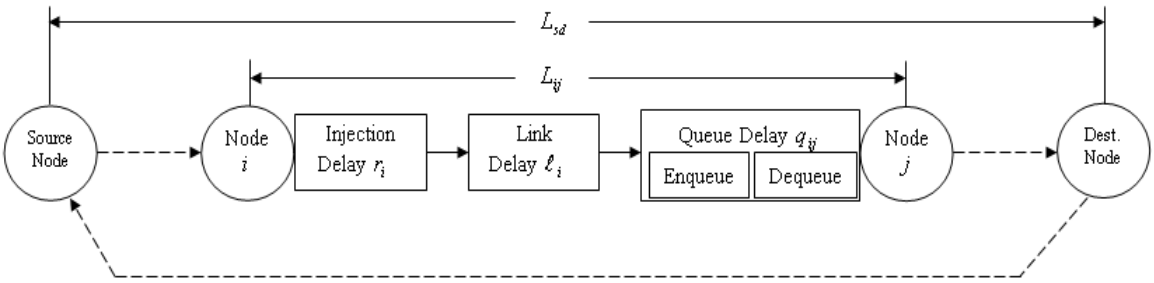


Figure 3-2: Times contribute to the total latency between source and destination

The time between two neighboring nodes is $L_{ij} = r_i + \ell_{ij} + q_{ij}$ where $j = i + 1$. The injection delay can be calculated as follows: $r_i = \frac{S_{flit}}{Bw_{\max}}$. The total delay for one flit from source node to destination node is $L_{sd} = \sum_{i=1}^{n} L_{ij}$ or $L_{sd} = n(\frac{S_{flit}}{Bw_{\max}} + \ell) + \sum_{i=1}^{n} q_{ij}$, where n is the number of nodes that the flit traverses.

3.3 The Application Model

3.3.1 Synthetic generator workload

In this study, two types of application representations are used. These are standard traffic models, commonly used in the literature to mimic certain application domains such as streaming applications, and real application representations based on an application selected and instrumented using real hardware.

For evaluating the performance using a system-level simulation methodology, we consider that the application is represented as communicating parallel processes. Each process is linked with a traffic generator that injects flits according to a given traffic pattern. In the proposed evaluation, two cases are considered to generate traffics in the system. In the first method, we assume that PEs continually send data at a constant rate to the on-chip interconnect. In this method, the CBR (Constant Bit Rate) model is used to inject flits at a deterministic rate. The second case employs probabilistic functions to model the application traffic behavior. Examples of applications exhibiting this pattern are audio and video streams [51].

3.3.2 Real application workload

There is a need to model certain applications and study their characteristics so that the NoC architecture can be adapted for specific applications. To do so, a methodology to characterize traffic for SoC applications is proposed. The methodology flow is depicted in Figure 3-3. The methodology uses benchmark applications running on a multiple node parallel machine (e.g. a cluster) and extracts the communication traffic as traces. Two real applications, FFT and MG, have been selected from the NAS benchmark set, as representative examples in this study.

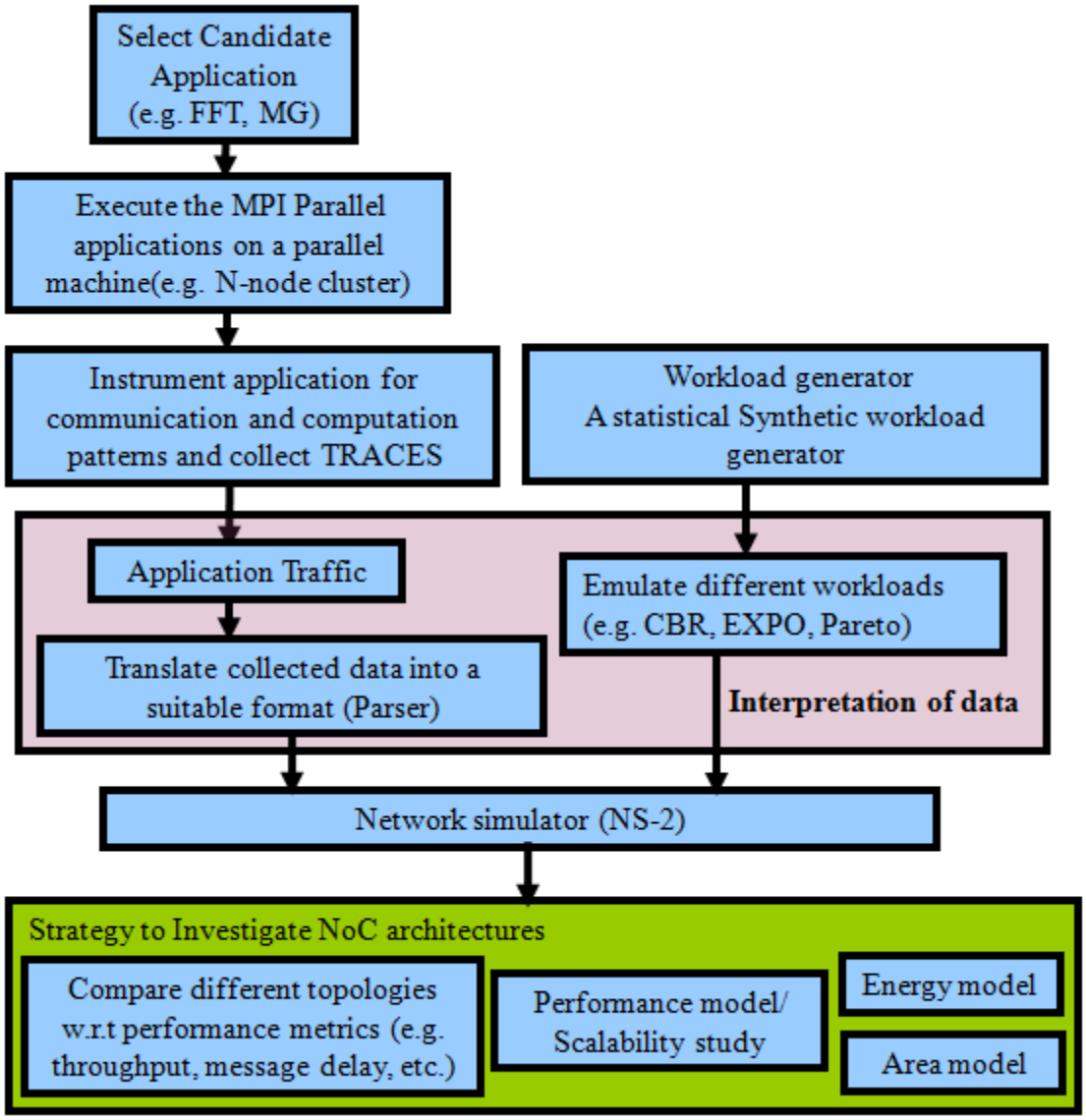
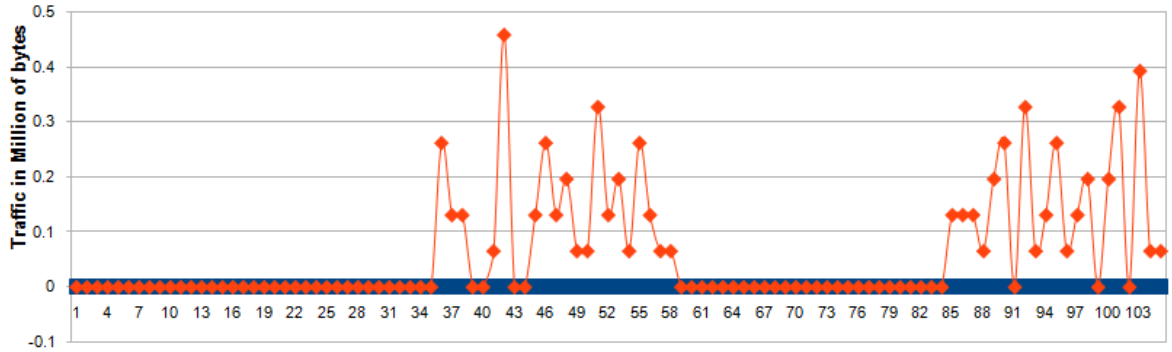
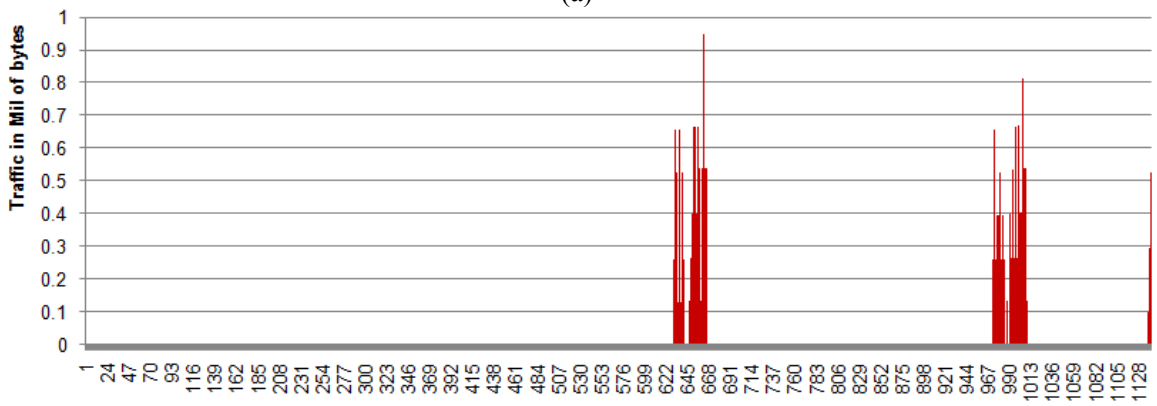


Figure 3-3: Methodology for simulating real application workload (left) compared with synthetic workload generator (right)

They represent two distinct types of application related to the communication traffic they generate. FFT is considered a communication intensive application with little computation, while MG is considered computation intensive with less communication traffic. It is possible to characterize the application traffic using exponential function as shown in [52].



(a)



(b)

Figure 3-4 : Visualization of communication traffic (a) FFT and (b) MG applications

An instrumentation tool MPICL [53] is used to instrument the communication events in these applications. Figure 3-4 depicts the shape of the communication events for both FFT and MG applications. The X axis represents the time intervals in which the communication events occur while the Y axis represents the amount of traffic generated.

The traces are collected and analyzed and statistical characteristics are extracted and made usable for the simulator. In addition, common workload specifications found in literature [52] are also used and their parameters are fed into NS-2.

3.4 Simulation Methodology

A deterministic routing protocol is used to direct packets between routers. Unlike dynamic routing protocols that require the implementation of complex logic at the router level, deterministic routing protocols are usually used because they require simple logic

for implementation [32]. A wormhole switching technique is also used to overcome the packets' drop problem, reduce the size of the buffer, and achieve a low latency.

A traffic generator that imitates applications is attached to each IP node. Three kinds of source traffic models are considered: CBR, Exponential, and Pareto traffic sources. The CBR model generates traffic at a deterministic rate. Exponential and Pareto models [51] are used as traffic generators to generate flit bursts. The exponential model is an ON/OFF model with exponentially distributed ON and OFF times. During each ON period, an exponentially distributed random number of packets are generated at a fixed injection rate, with an average OFF time of 1ms and an average ON time of 0.1s as suggested in [25]. In the Pareto model, five parameters are considered: the burst time, the idle time, the traffic rate, the size of the flits, and the shape. The burst time is the average ON time for the generator. The idle time is the average OFF time for the generator. The traffic rate is the injection rate during a period of ON time. During each ON period of the Pareto flow, flits are generated at a fixed rate, an average burst time of 0.1s and an average idle time of 1ms. The shape parameter of the Pareto distributed OFF and ON times is 1.5.

The values considered for the injection rate are 100, 130, 160, and 190 Mbps. 100Mbps represents light injection rate and 190Mbps is the heavy injection rate approaching the maximum permitted injection arte of 200Mbps. In the reported results, the flit size is 8 bytes generated according to the traffic generator used. Table 3-2 summarizes the parameters related to traffic generators.

Note that the values of the burst time and the injection rate affect the communication load, i.e., if these values are high, then more flits are generated into network in unit time.

The value of the traffic rate should be lower than the maximum bandwidth; otherwise, a large number of flits will be dropped.

Table 3-2 : Traffic parameters for CBR, Exponential, and Pareto generators

CBR	Sources	16
	Sinks	16
	Traffic pattern	Peer-to-peer
	Flit size	8 byes
	Injection rate	100, 130, 160, 190 Mbps
Exponential	Sources	16
	Sinks	16
	Traffic pattern	Peer-to-peer
	Flit size	8 byes
	Injection rate	100, 130, 160, 190 Mbps
	ON time	0.1s
	OFF time	1ms
Pareto	Sources	16
	Sinks	16
	Traffic pattern	Peer-to-peer
	Flit size	8 byes
	Injection rate	100, 130, 160, 190 Mbps
	ON time	0.1s
	OFF time	1ms

At the beginning of the simulation instance, each source starts to generate flits according to the Pareto ON/OFF distribution. In this simulation, all module nodes are traffic sources that transmit packets. However, sinks (destination) are selected according to the communication locality principle described in [25]. This principle states that 75% of the traffic takes place between neighboring resources and 25% of the traffic is uniformly distributed among the rest. More precisely, 0.75 represents the probability of one resource being one hop away from resource selected as a sink, and 0.25 represents the probability of one resource being more than one hop away from the resource selected as a sink. In this simulation, all nodes are selected to be sources and sinks in order to simulate a high traffic network load.

3.5 Experimental results

As described earlier, the two primary components of a NoC are the IP nodes and Routers. For our studies, the IP node is considered to have unlimited buffer capacity, whereas the buffer size of routers is finite. The buffer modeled in each router is a FIFO using a Drop-tail queue management mechanism, i.e., all the arriving flits are dropped whenever the FIFO is full. Furthermore, there are two types of links, router-to-IP and router-to-router. Two parameters are used for modeling both types of links: link delay (d_ℓ=0.1ms) and maximal bandwidth ($Bw_{\max}$=200Mbps). In all simulations carried out on the NS-2 simulator for the NoC comparison study, the transmission protocol UDP is adopted to transport packets from IP sources to IP destinations. This protocol is adopted because of low overhead compared to other protocols that guarantee the transmission. Static routing is used for all networks implemented using routing tables obtained from Dijkstra's all-pairs shortest path first (SPF) algorithm. The routing tables are incorporated in all the routers. To model the network traffic, each IP node is attached to a traffic generator that is randomly activated to act as a source.

The following scenarios are considered for the comparative simulations of different topologies, such as: (a) variation of latency with the buffer size, (b) loss probability for different buffer sizes, (c) variation of the loss probability with communication load and (d) throughput versus loss probability. The results are first shown for the Spidergon network as an illustrative example. Subsequently, a comparative study is presented for different networks such as Spidergon, Mesh, Ring, Fat Tree (FT) and Butterfly Fat Tree (BFT). Finally, the best results are taken and compared against the WK-recursive network.

The variation of packet transmission latency with the buffer size in each router is

shown in Figure 3-5, for the Spidergon network. The figure shows an increase in latency as the buffer size increases under heavy traffic conditions corresponding to an injection rate 190Mbps at each source. The observed trend is due to the increase in waiting time in each queue.

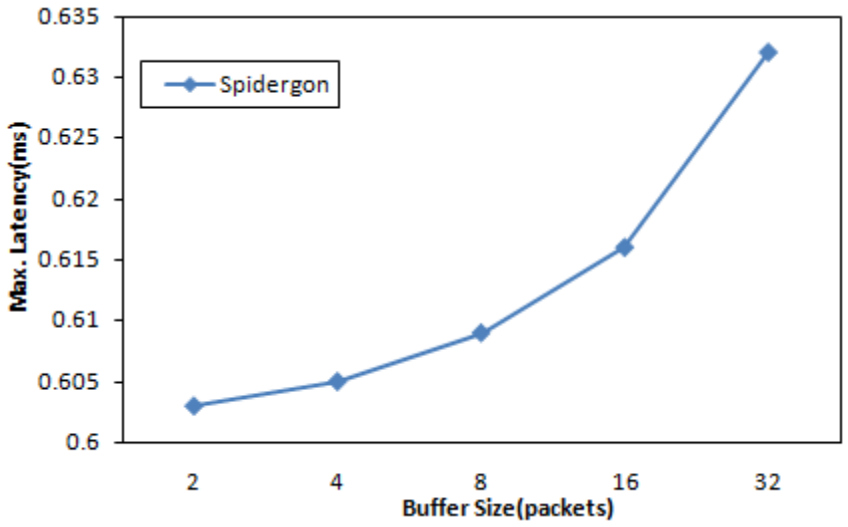


Figure 3-5: Latency vs. Buffer size in Spidergon with 16 nodes, 190 Mbps injection rate

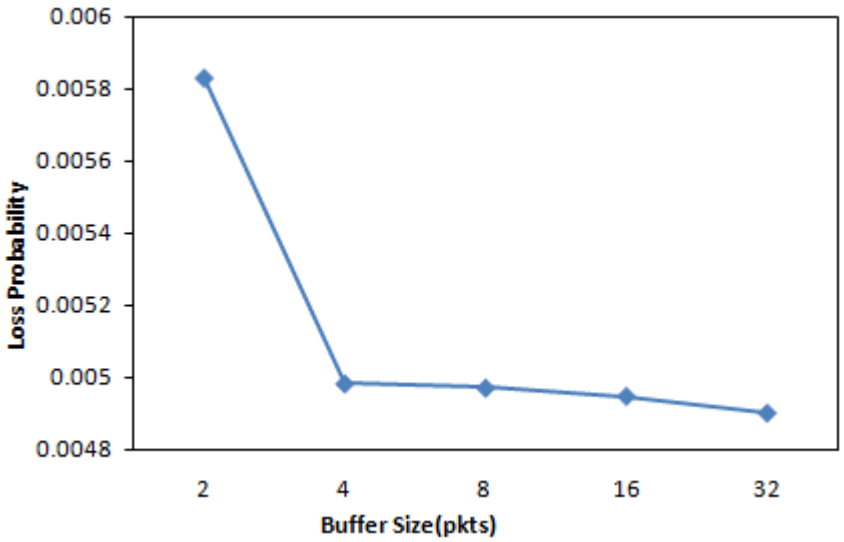


Figure 3-6: Loss probability vs. Buffer size in Spidergon with 16 nodes, 190 Mbps injection rate

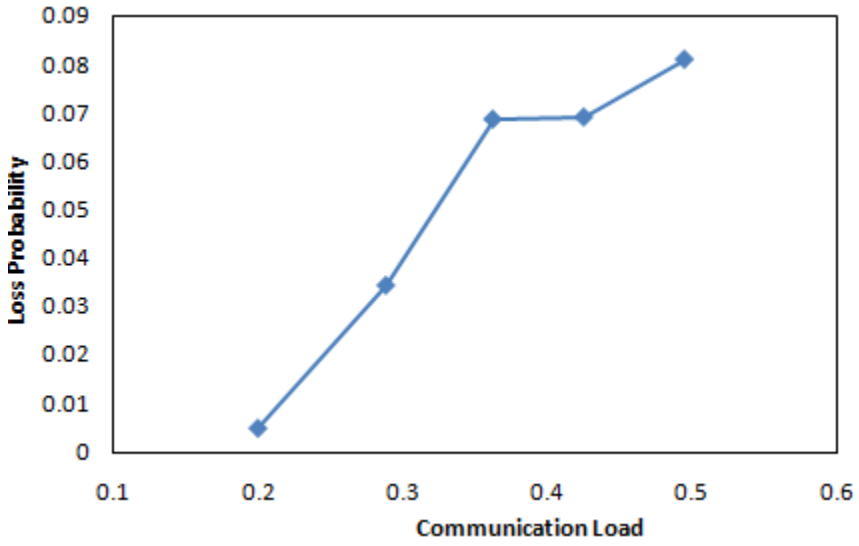
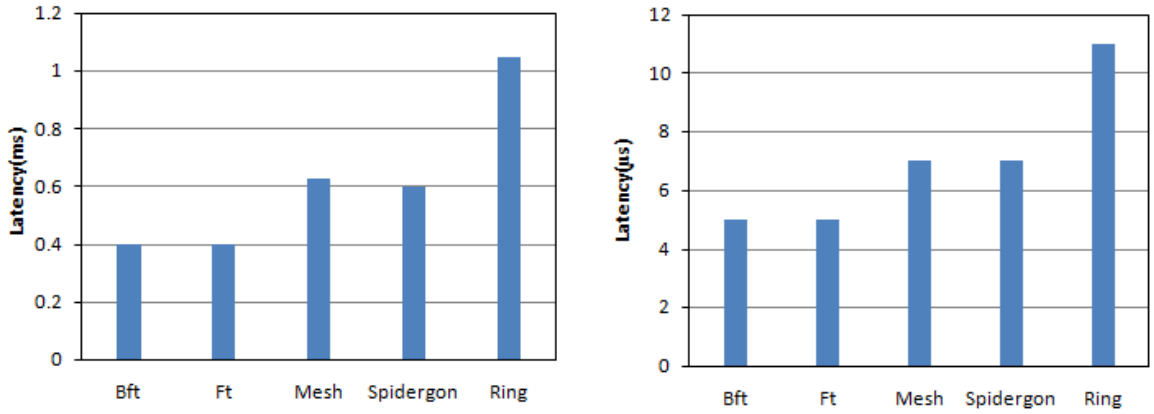


Figure 3-7: Communication load vs. Loss probability in Spidergon with 16 nodes

The loss probability for packets or flits, for different buffer sizes, is shown in Figure 3-6 for the Spidergon network. As expected, fewer packets are lost when the buffer size is increased in each router. The probability of a packet loss is high when the buffer size is relatively small (i.e. 2 packets). There is, however, a diminishing return for an increase in the buffer size beyond a certain point, here seen at a size of 4 packets.

If the buffer size is fixed, the number of packets dropped depends on the rate at which traffic is generated by the sources. Figure 3-7 shows that the likelihood of packet drop increases as the communication load increases when the buffer size is fixed (i.e. 16 packets). This is due to the fact that as the load increases, the network becomes more congested with heavy traffic. Thus, the queues tend to remain full for longer durations of time, leading to more packets being dropped.

The other topologies exhibit results similar to Spidergon and are not presented here for the sake of brevity. A comparison of the performance of different topologies is presented next.



(a) **(b)**

Figure 3-8: Comparing topologies for the maximum latency metric; (a) d_ℓ =0.1ms, Bw_{max}=200 Mbps and (b) d_ℓ =1µs, Bw_{max} = 4 Gbps

Figure 3-8 shows a comparison of the maximum latency for different NoC topologies. Since the network diameter plays a significant role in the average latency experienced by packets, the ring network exhibits the highest latency values as expected.

As described earlier in Section 3.1, the bandwidth and latency parameters were chosen for a relatively slower NoC. We also conducted the same set of experiments with bandwidth= 4Gbps and link delay = 1µs. Due to the increase in link bandwidth, we scaled up the injection rate from 190Mbps, to 3800 Mbps. The choice of link bandwidth was in fact limited by the largest possible injection rate we could simulate - using an injection rate larger than 3800 Mbps resulted in a floating point exception in the NS-2 simulator.

The results obtained using the faster set of parameters is given in Figure 3.8 (b). We observe the same general trend of the results as observed with the original set of parameters in Figure 3.8(a). However, with the faster parameters, there is a loss in accuracy in the latency results. This is attributed to the fact that the recorded time events in the output trace file of the simulator have a resolution limited to microseconds. In

other words, the minimum time duration that can be measured is 1 µs, and all time durations are integer multiples of 1 µs. When we use very high bandwidths such as 3800 Mbps, the injection delay r_i (see Section 3.2) is in the nanoseconds scale, which cannot be recorded in the simulator output file. Hence the simulation results are less accurate when we use faster values of the network link parameters. We therefore use the slower set of parameters given in Section 3.1.

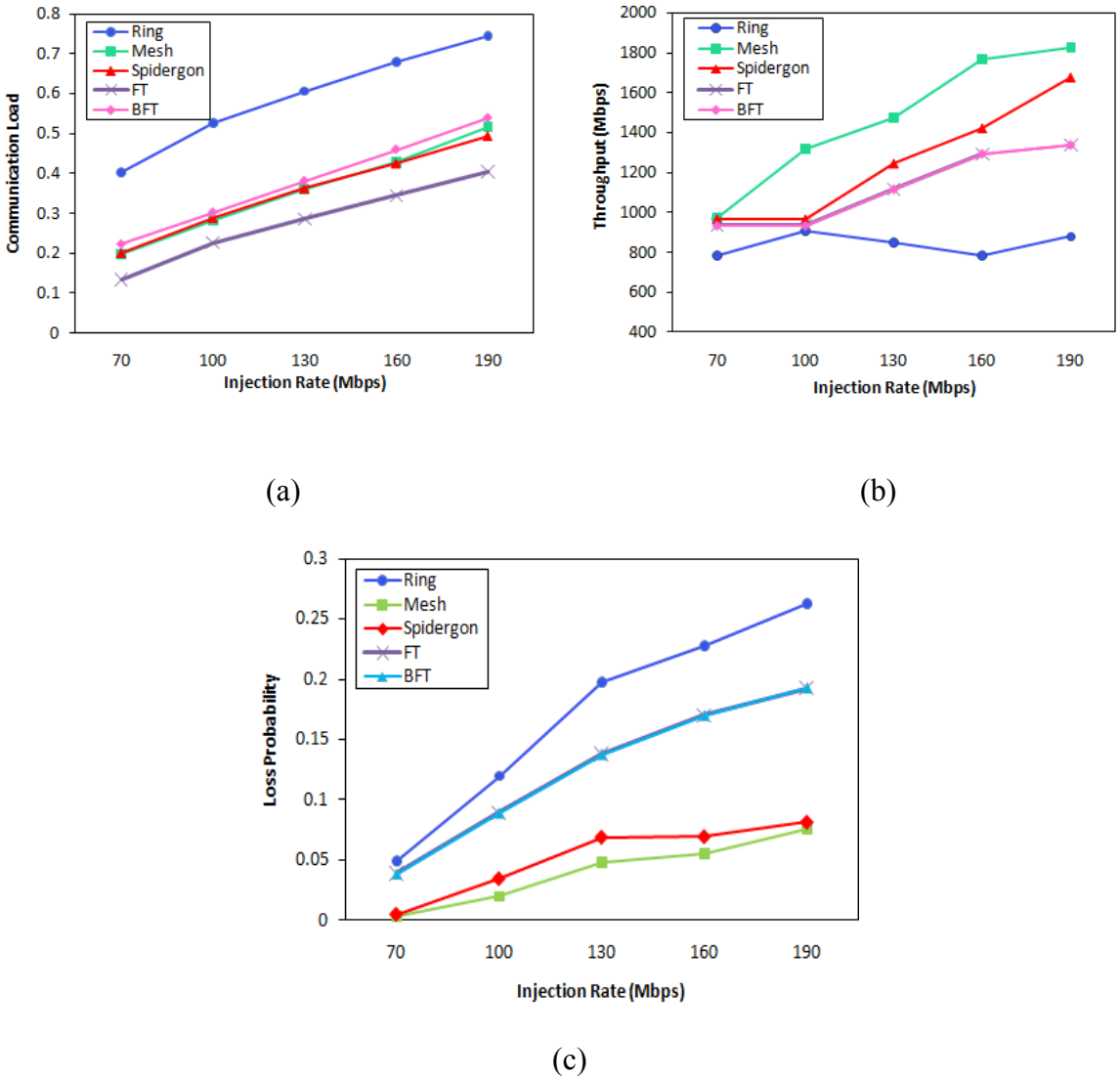


Figure 3-9: Comparison of different structures for different injection rate: (a) Communication load, (b) Throughput (Mbps), and (c) Loss probability

Figure 3-9(a) shows the variation of communication load under different traffic rates. It is observed that the FT topology outperforms the rest of the topologies for this metric. However, FT is not suitable as a NoC due to its large number of links.

Figure 3-9(b) shows the variation of throughput under different injection rates. Mesh outperforms the rest of the topologies because it has the lowest sensitivity for packet drops (sensitivity means that the slope of the curve is high for one topology compared to the other). Figure 3-9(c) depicts the variation of packet drop probability under different traffic rates. As the injection rate increases, the likelihood of packet drop increases for all topologies; however, the Mesh and Spidergon have the least tendency, while the ring is the worst. Note that FT and BFT produce similar results in latency, loss probability and throughput but FT outperforms BFT in communication load because of the additional redundant router nodes that increase the number of available links.

Chapter 4 Exploring NoC Architectures: Energy and Area Considerations

After exploring the performance metrics for common NoC architectures, other metrics such as energy and area requirements need to be investigated because of their prime importance in NoC. In regular computer networks, the energy consumed by routers is not a major concern due to the size of the network and wires, while the area requirements are not considered since the space covered by the network is usually very large compared to the size of the main network components. Table 4-1 compares NoC architectures to regular computer networks and parallel machines.

Table 4-1 : Comparison of NoC with other network types

Network on Chip (NoC)	Parallel Computing	General computer Network
Higher degree of heterogeneity	Lower degree of heterogeneity	Heterogeneous
Tailored toward Application	General purpose Applications	General purpose Applications
Important energy constraints (energy aware routing and switching techniques are needed)	Energy is not a major concern	Energy is not a major concern
Defined topology (Fixed by design)	Defined Topology (extendable)	Undefined Topology
Sophisticated routing elements (routers)	Routing is generally handled by the processor	Routers and switches

In this chapter, we will extend the evaluation of NoC architectures by introducing energy and area models. An additional topology called WK-recursive will be introduced and adapted for NoC and then compared with the best two topologies obtained from the performance comparison results, namely, Mesh and Spidergon. The energy and area models will be used for extending the evaluation of NoC architectures. Scalability for selected NoCs will also be studied because of the importance of this property in the NoC evaluation process.

4.1 The Energy Model

To calculate and evaluate the energy consumption of NoC architectures, we adapted the model proposed in [35,36], which estimates the energy consumption for the traversal of each bit. The total energy required to send one flit through the network can be decomposed into the energy consumed per traversal from an input port of a router to an output port, and energy consumed per wire distance traversed. Formally, the flit energy ($E_{i,j}$) metric is defined as the energy consumed when one flit is sent from its source, PE_i, to its destination, PE_j. It is calculated as follows:

$$E_{i,j} = hE_L + (h-1)E_R \tag{5}$$

where h is the number of links the flit crosses to travel from PE_i to PE_j, E_L is the average energy consumed during transportation of the flit between two neighboring nodes, and E_R is the average energy consumed during buffering and routing operations inside a router. The values of E_L and E_R are constants for a given on-chip interconnect and depend mainly on the router or switch architecture and the link characteristic such as the width, the length, etc. In this evaluation, we use the values already estimated in [36] in which the amount of energy required for a single bit to pass the switch or router is E_R = 0.9776 pJ/bit. The amount of energy required for a single bit to cross a link is E_L = ($0.39 + 0.12 \times \ell_W$) pJ/bit, based on the model proposed in [6] using 0.13µm CMOS technology library. Here, ℓ_W is the length of the link in *mm*. Based on equation (5), the total energy consumed by all flits can therefore be calculated as follows:

$$E = \sum_{k=1}^{Nb_{flits}} S_{flits}(h(k)E_L + (h(k)-1)E_R) \tag{6}$$

where $h(k)$ is the number of links traversed by the k^{th} flit, Nb_{flits} is the total number of flits generated, and S_{flits} is the size of each flit. Expanding equation (6) using expressions for E_L and E_R, the average values for the link length (ℓ_W^{av}), and number of hops (h_{av}), the expression for total energy consumed is:

$$E = Nb_{flits} S_{flits} [h_{av} (0.39 + 0.12 \times \ell_W^{av}) + 0.9776 (h_{av} - 1)] \quad (7)$$

The value of ℓ_W^{av} is computed based on the chip layout depicted in Figure 3-1. In all topologies, we consider that the link between each PE (resource) and its corresponding router is of length ℓ_W^r, equal to 1mm. There are 16 links of length ℓ_W^r. The length of links between neighboring routers is calculated based on the on-chip layout. All links (horizontal or vertical) between neighboring switches, at *1, 2,.., i,..., n* hops, when placed on the 2-D regular substrate are also considered to be of length $\ell_W^{s_1}$, $\ell_W^{s_2}$,…, $\ell_W^{s_i}$, …, $\ell_W^{s_n}$ respectively, where $\ell_W^{s_i} = 2 \times i\ mm$.

We can see that the energy consumption depends mainly on three parameters: application traffic (number of generated flits, Nb_{flits}), the number of hops or links h_{av} that the flits traverse, and the average link length ℓ_W^{av}. The values of Nb_{flits} and h_{av} will vary depending on the application. Three different applications were used in order to study the variation of these parameters as well as the total energy.

4.2 The Area Model

For the NoC architectures depicted in Figure 3-1, we use some high level abstractions based on the literature in order to obtain estimates for the required silicon area. There are three components that contribute towards the total area: switches/routers,

links and PEs. The switches, in turn, are composed of two main components: the buffers used for temporarily holding the flits and the logic which implements the routing algorithm. The links exist between routers or between a router and a PE; the area required for a link depends on its length [31]. For the PE resources, the area can vary based on the size of the PE components, such as DSP, FPGA and Memory. If we denote the area requirements for the switch *i*, resource *j* and link *k* by $A_R(i)$, $A_{pe}(j)$ and $A_l(k)$ respectively, the total area may be computed as:

$$A = \sum_{i=1}^{N_R} A_R(i) + \sum_{j=1}^{N_{pe}} A_{pe}(j) + \sum_{k=1}^{N_l} A_l(k) \tag{8}$$

where N_R is the number of routers, N_{pe} is the number of resources (PEs), and N_l is the number of bidirectional links. Due to unequal values of PE areas, link lengths and the areas of switches, we use average values for each of them in order to obtain an estimate for the total SoC area. Therefore:

$$A = N_R A_R + N_{pe} A_{pe} + N_l A_l \tag{9}$$

where A_R is the average router area, A_{pe} is the average resource or PE area, and A_l is the average link area. The value of switch area A_R is given by the sum of the buffer area B_{tot} and the remainder switch silicon area R_s required for the routing table and the logic that implements the routing algorithm. That is, $A_R = B_{tot} + R_s$. The buffer area B_{tot} is, in turn, estimated as:

$$B_{tot} = a_s d_g S_{flits} B_s \tag{10}$$

where B_s is the buffer size in flits, a_s is the area required for one byte, S_{flits} is the size of each flit in bytes, and d_g is the average degree of the on-chip interconnect topology (which also represents the average number of buffers inside the switch/router).

The link area A_l is given by the product of the link area per unit length, a_w, and the link length, ℓ_w^{av}. In other words, $A_l = a_w \ell_w^{av}$. Substituting for A_R and A_l in (9), the total area is:

$$A \approx N_R (R_s + a_s d_g S_{flit} B_s) + N_{pe} A_{pe} + a_w N_l \ell_w^{av} \quad (11)$$

4.3 WK-Recursive Network Overview

The WK-recursive network has received considerable attention in the parallel computing community due to its many favorable properties, such as scalability, and ease of extendibility [10].

One of the interesting properties of this structure that suits NoC is its ease of expandability; the WK-recursive network can be constructed hierarchically by grouping basic modules. More precisely, a WK-Recursive network with amplitude W and level L, denoted by WK(W,L), can be recursively constructed [10]. A WK(W,0) is a vertex or node with W free edges. A WK(W,1) is a W-vertex complete graph that is denoted by a K_W, with the degree K equal to W. Each vertex has one free edge and W-1 edges that are used for connecting to the other vertices. A WK(W,H) consists of W copies of WK(W,H-1) as supervertices and the W supervertices are connected as a K_W, where $2 \le H \le L$. By induction, WK(W,L) has W^L vertices and W free edges. Consequently, for any specified amplitude W, WK-Recursive networks can be expanded to an arbitrary

level L without any changes to the router design. The structures of WK(4,0), a WK(4,1), and WK(4,2) are depicted in Figure 4-1, and the structures of WK(5,0), a WK(5,1) and WK(5,2) are depicted in Figure 4-2.

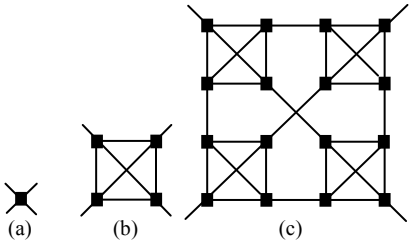


Figure 4-1: The structures of (a) WK (4, 0), (b) WK (4, 1), and (c) WK(4, 2)

More details about the WK-recursive network features and topological properties can be found in [54].

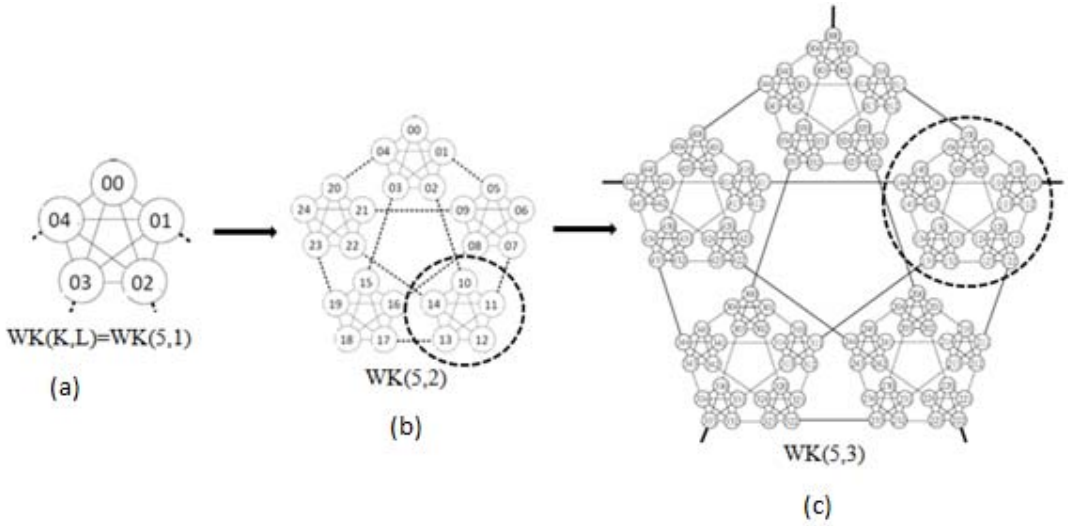


Figure 4-2: WK-Recursive Topology for K=5 for (a) level 1, (b) level 2 and (c) level 3

For the structure shown in Figure 4-1 (c), W=4 (amplitude), K = 4 (degree), and L = 2 (level), and in Figure 4-2 (c), W=5, K=5, and L=2. The fundamental property of WK is the fact that W=K always and the network is characterized by the degree for each supervertex, the number of levels L, and is represented as WK (K, L). The diameter of

the network is independent of the amplitude W and degree K, and has a value D =$2^{L} - 1$. The network structure scales well since the degree remains the same for L=2,3,4,5 … The number of levels and number of nodes are related by: $L = log_K N = log_W N$.

Other attractive properties of WK-Recursive networks include high network connectivity [55] and Self-routing property (routes based on nodes labeling scheme) [56].

Due to the above properties of the WK-recursive network, especially the ease of extendibility that suits NoC systems, we study and investigate this structure within the NoC domain. Section 4.4 provides details of this study and includes performance comparisons with a 2D Mesh of the same size, as well as other topologies.

Since the degree is the same irrespective of the number of levels, a fixed router design, such as the one shown in Figure 4-3, can be used for WK(4,L) networks. Note that the routers used in the interior of the network would use all the 5 links associated with the router, whereas the routers at the corners will use only 4 links. For all cases, the value of d_g used in Equation (10) is d_g = 5, for WK(4,L) networks.

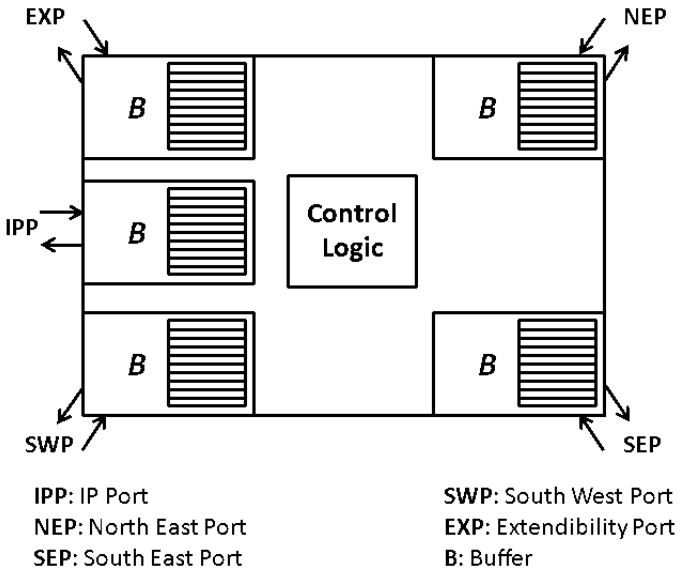


Figure 4-3: Router structure in WK(4,2)-recursive on-chip interconnect

4.4 WK-Recursive Network Evaluation

In this section, WK-recursive on-chip interconnect, WK(4,2), is compared with Spidergon and Mesh networks. Different simulation scenarios for each topology are studied based on the simulation parameters specified earlier. Different values of injection rates are considered for evaluating NoC performance metrics. The values considered are 70, 100, 130, 160, and 190Mbps. The buffer's size is fixed to 32 flits.

4.4.1 Performance evaluation

Figure 4-4 compares the maximum latency for the three topologies for the following two cases: (a) heavy traffic (190Mbps injection rate) and (b) light traffic (70Mbps injection rate). It is seen that the WK-recursive topology offers a significantly better performance, compared to the other two networks.

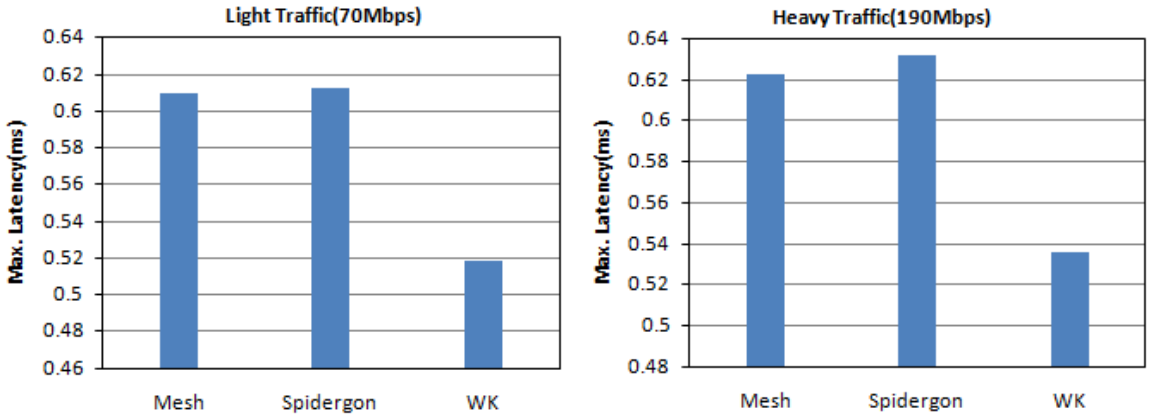


Figure 4-4: Maximum latency for topologies using: (a) heavy traffic, (b) light traffic

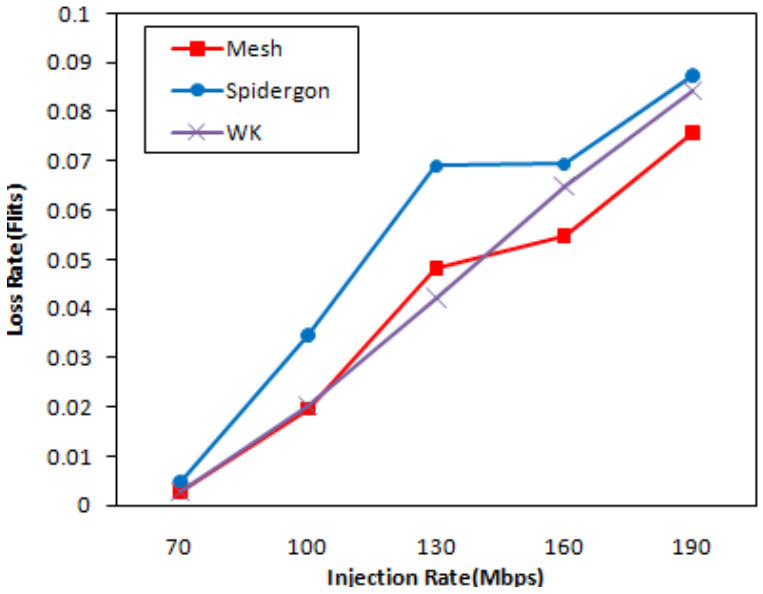


Figure 4-5: Loss probability vs. injection rate for three NoCs with 16 nodes

Figure 4-5 illustrates the variation of loss probability under different injection rates. As the injection rate increases the likelihood of dropped flits increases for all topologies; the WK-recursive topology has a low sensitivity when the injection rate is less that 160Mbps. The Mesh is less sensitive than WK, while the FT has the highest sensitivity.

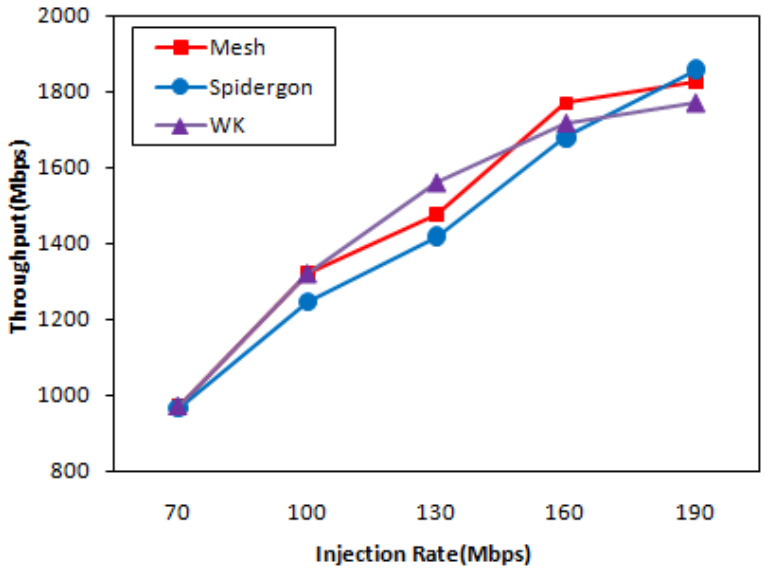


Figure 4-6: Throughput vs. injection rate for three topologies, # nodes=16

Figure 4-6 shows the variation of aggregated throughput under different injection rates. Independent of the injection rate, the throughput values are relatively similar for the three topologies.

The results depicted in Figure 4-7 show the variation of communication load under different traffic rates. Again, the WK-recursive topology exhibits very good results.

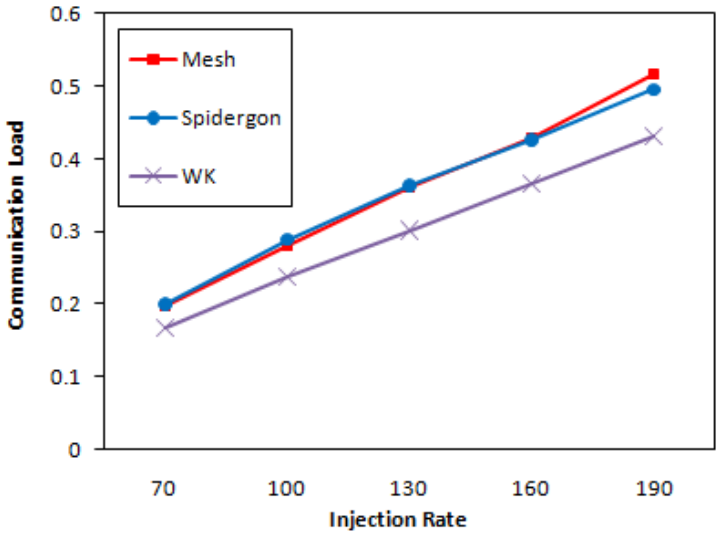


Figure 4-7: Variation of communication load vs. injection rate

Results similar to those presented here were obtained using different buffer sizes; the measured values of the performance metrics are in the same order of magnitude.

In conclusion, the WK-recursive topology compared favorably against Mesh and Spidergon topologies, because of its regular structure as well as low diameter, leading to lower latency and lower communication load. However, under heavy traffic conditions, a slight increase in the dropping of flits is observed. Other important metrics, such as energy consumption and area requirements, are evaluated in the following subsections.

4.4.2 The energy evaluation

Different on-chip architectures are simulated and the energy consumption is estimated. Similar to the studies described in the previous subsection, different values of the injection rates are considered, varying from light traffic to heavy traffic conditions (100, 130, 160, and 190Mbps). The values of the buffer sizes used in the simulations are 2, 4, 8, 16, and 32 flits. The size of each flit is considered constant in this simulation (8 bytes).

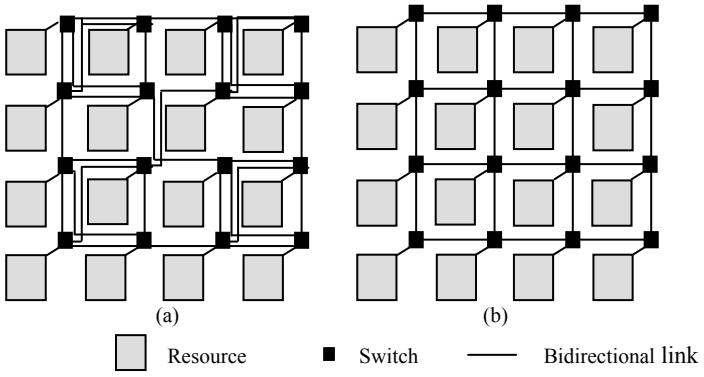


Figure 4-8: Chip layout of NoC Topologies with 16 nodes: (a) WK (4,2), (b) 2D Mesh

To evaluate the average energy consumption, the average link length ℓ_W^{av} for each of the NoCs has to be calculated based on their approximated two-dimensional chip layout as depicted in Figure 3-1. For the WK-recursive network, the two-dimensional placement shown in Figure 4-8 (a) is used. For convenience and for comparison, the Mesh topology from Figure 3-1 is repeated in Figure 4-8 (b). The Mesh has 16 links of length ℓ_W^r and 24 links of length ℓ_W^{s1}, so the average length is 1.6 mm (for the definition of the symbols, please see Section 4.1). The WK-recursive network has 16 links of length ℓ_W^r, 20 links of length ℓ_W^{s1} and 10 links of length ℓ_W^{s2}, so the average link length is 2.087 *mm*.

Based on the energy model (see equation (7)), the energy consumption depends on a variety of factors. However, the size of the flits (S_{flits}) and the average link length (ℓ_W^{av}) are constants. Therefore, the energy consumption depends mainly on the application traffic (number of generated flits) and the number of hops or links traversed by the flits, both of which can be easily ascertained from the output generated by NS-2.

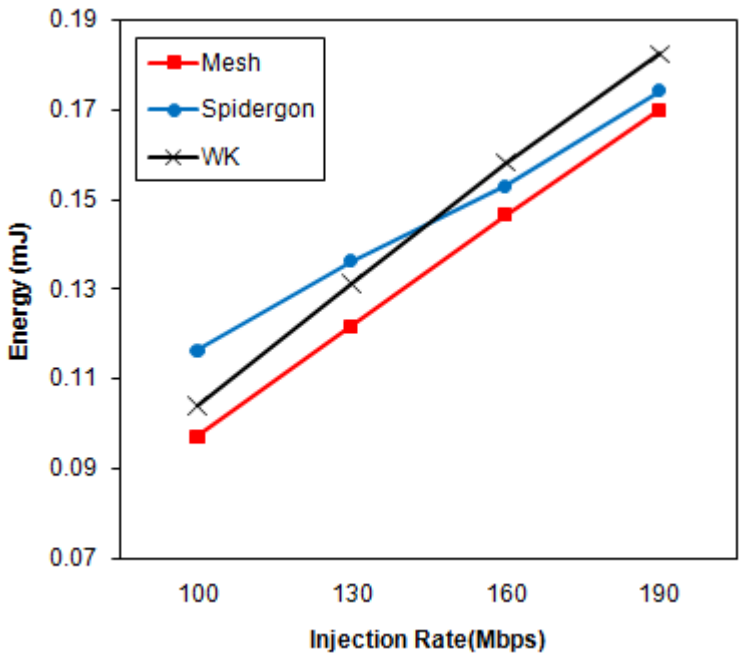


Figure 4-9: Comparison of the average energy consumption using CBR traffic for NoCs with 16 nodes

The effect of the number of flits, or equivalently, the injection rate, is first studied. Figure 4-9 compares the energy consumption of the different on-chip architectures as a function of the injection rate, when the constant bit rate (CBR) traffic model is used (the buffer size is fixed here to 16 flits). Results using other traffic models have also been obtained, but not shown here for the sake of brevity. The figure shows that the WK-recursive topology has the highest energy consumption compared to the other

architectures. In addition, the average energy consumption increases when the injection rate increases, as expected.

To explain the reasons behind the higher energy consumption for the WK-recursive topology, the number of hops for the networks needs to be compared. Figure 4-10 shows the comparison for different injection rates. It is seen that the WK-recursive topology has the highest number of hops. The larger number of hops is due to the topological characteristics of the WK-recursive architecture (such as a larger number of paths that are diameters).

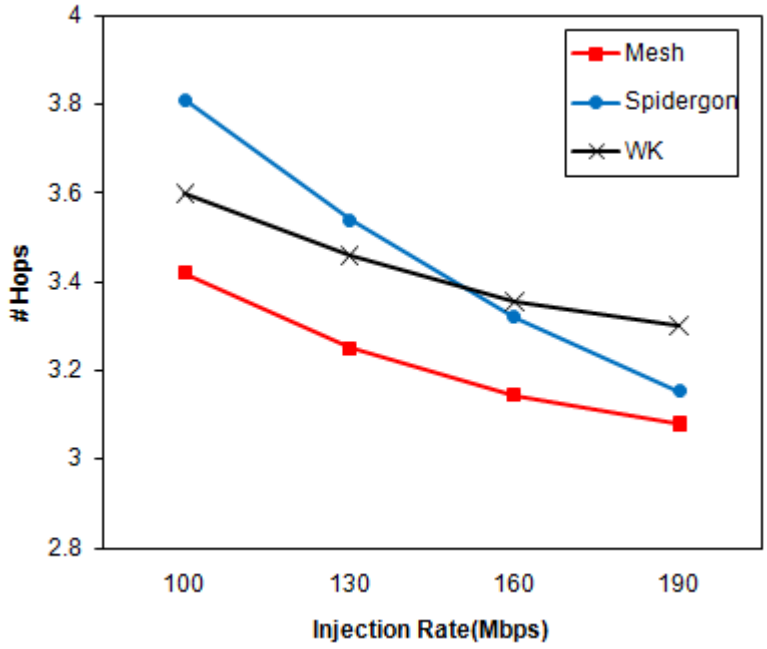


Figure 4-10: Comparison of the average number of hops using CBR

From Figure 4-10, it is observed that the average number of hops decreases with an increase in the injection rate. The reduction is due to the fact that a larger number of flits are dropped at higher injection rates, thereby reducing the number of flits that traverse the entire distance from source to destination. It may also be noted that the reduction is less for the WK-recursive architecture, since the sensitivity of the loss probability is less for

this architecture (as described earlier). This in turn leads to higher energy consumption for the WK-recursive architecture at higher injection rates. It is important to note here that although the average number of hops decreases with the injection rate, the energy consumption increases, since energy also depends on the number of flits generated. Nevertheless, the higher number of hops for the WK-recursive architecture highlights the need for direct links between heavily communicating nodes for this architecture.

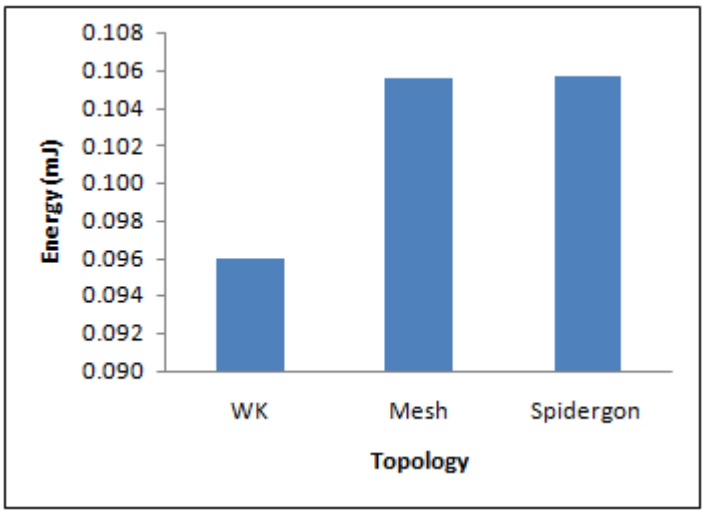


Figure 4-11: Comparison of the average energy consumption using CBR under 5Mb/s (flit drop=0)

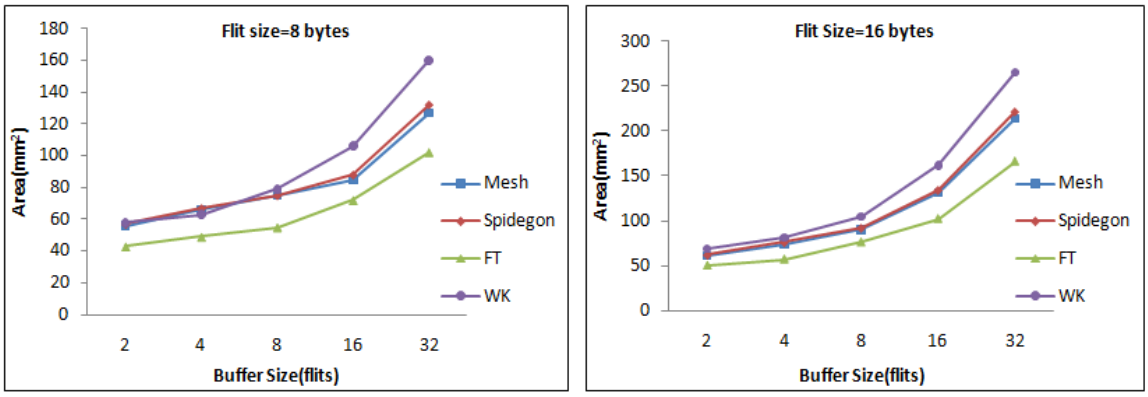
All the earlier plots compare the performance of various networks at highly overloaded cases, effectively showing us which of them saturates faster and deteriorates in performance as the injection rate is increased. All the performance results show that WK is better than both Spidergon and 2D Mesh.

For energy, it is not a fair comparison at these injection rates, because a large percentage of flits is dropped and not counted for, and the network is in not "normal/functional operation". To ensure that all flits are counted, we increase the injection rate starting from a very low value, until any further increase results in the

dropping of flits. An injection rate of around 5 Mb/s was eventually used. Figure 4-11 shows the energy results based on this injection rate, with no flits being dropped. It shows that WK consumes less energy compared to Mesh and Spidergon.

4.4.3 The area evaluation

The switches or routers are the key components of a NoC due to the buffering and routing tasks; the different architectures will therefore be compared by varying the buffers and flits sizes. Since a deterministic routing algorithm is used and the number of PEs and routers is fixed, the area required to store the routing table and other related components of the area are considered constant, $R_s = 1mm^2$ in this evaluation.



(a) (b)

Figure 4-12: Average area comparison of Mesh, Spidergon, FT and WK by varying the buffer size: (a) flits size equal 8 bytes, and (b) flits size equal 16 bytes

Figure 4-12 summarizes the area comparison of WK-recursive with different on-chip interconnect architectures by varying the buffers size, using $a_s = 0.005mm^2$, $a_w = 0.02mm^2$, $A_r = 2mm^2$, $R_s = 1mm^2$; the symbols have meanings as described in Section 3.5.

We consider that the chip size is of $20mm \times 20mm$. The value of ℓ_w^{av} for each architecture is calculated as described in section 4.4.2 based on the layout of the architectures. WK-recursive network uses more wires to achieve high throughput and low latency. As expected, when the buffer size is equal to 32 flits, it requires more area; for example, it needs 40% more than the most area efficient implementation of BFT. However, the loss probability in WK and 2D Mesh architectures is approximately 4% compared to 25% in BFT. With the buffer size equal to 2 flits in WK, we can obtain high throughput as compared with the other architectures that use a buffer size equal to 32 flits. So, the WK occupies less silicon area when the performance metrics are considered. We can see from Figure 4-12 that the buffer size significantly affects the router/switch area. Therefore, the buffer size should be carefully selected, because an unnecessarily large size increases the silicon area and a small buffer size affects the performance. The choice of buffer size depends both on the application as well as on the underlying on-chip interconnect.

4.5 Scalability Study

All previous simulation experiments were conducted on fixed NoC topologies where the number of nodes is always equals to 16 PEs. As we mentioned earlier, NoC paradigm was introduced to address the scalability problem faced in SoCs when a non-scalable bus-based system is used. It is therefore clear that in the evaluation and the NoC exploration process, the scalability should be addressed as one of the main factors.

To determine if the NoC is scalable we will evaluate the effect that increasing the size of network has on the main performance and energy metrics. The main structural and

topological characteristics of NoC are first analyzed, and then simulation study will be conducted to evaluate this effect. The three main topologies will then be compared based on energy and performance metrics.

As part of the initial analysis, we first consider the diameter of the network, which is one of the main topological parameters which clearly affects the performance and energy consumption. Observing the diameter when increasing the number of nodes gives a general insight about the performance of the topology and whether the network is scalable enough to suite NoC systems.

Figure 4-13 compares the diameter of WK-recursive network and 2D Mesh. It shows that WK-recursive is more scalable than Mesh, which we can expect to see in the performance and energy results from simulation.

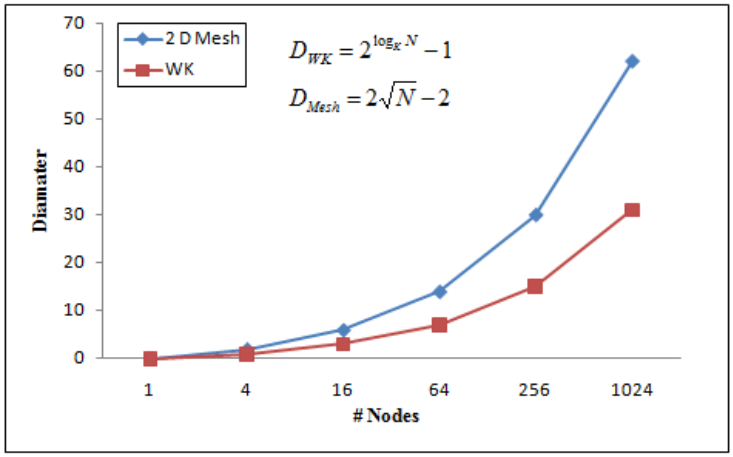


Figure 4-13: Comparing diameters of 2D Mesh D_{Mesh} and WK-recursive network D_{WK} for varying # Nodes. Clearly, WK-recursive scales better than 2D Mesh

In the second phase, experimental study based on simulation is conducted. It uses similar experimental environment with only difference in some parameters such as buffer size to accommodate the growing size of the network and prevents network from getting

overloaded or congested. NoC structure are built with size growing: 16 , 64 and 256, and the main performance metrics are reported. Figure 4-14 shows a comparison between the three NoCs when using CBR traffics as the application and observing the effect of increasing the size of the network on average latency. While the WK-recursive scales slightly better than 2D Mesh for reasons discussed earlier, the general conclusion is that WK-recursive and 2D Mesh scale relatively well compared to Spidergon.

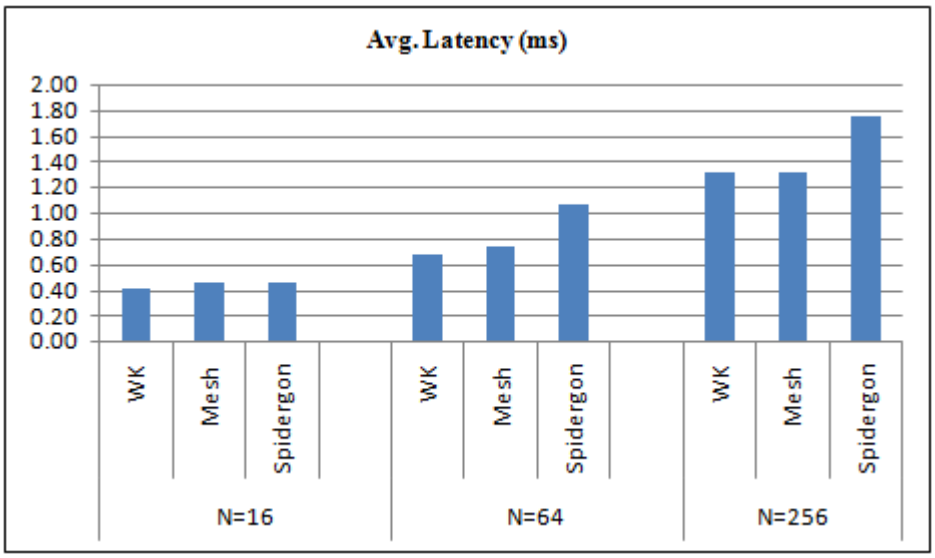


Figure 4-14: Comparison of average Latency of three NoCs for varying # nodes N

Figure 4-15 shows comparison between the three architectures when energy is observed. We can easily notice the same pattern compared to average latency, however, Spidergon clearly doesn't scale well when energy is observed.

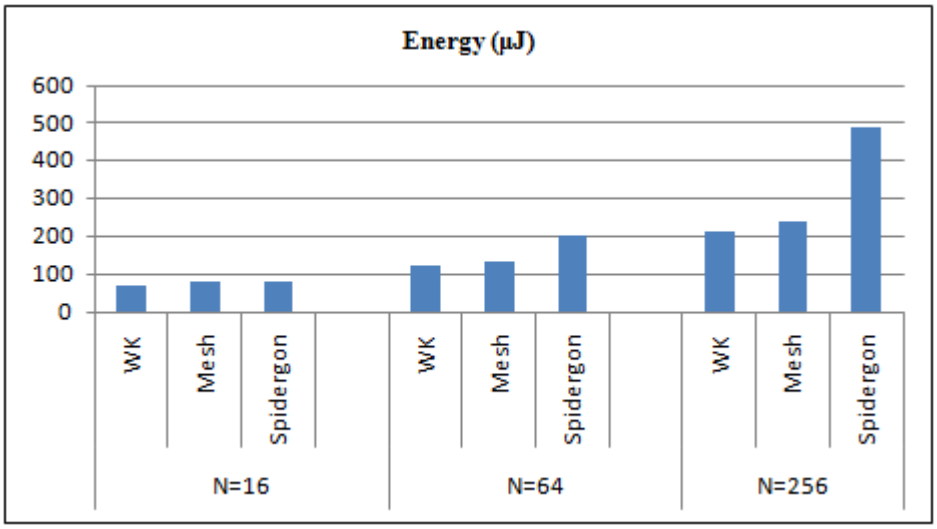


Figure 4-15: Comparison of energy consumption of three NoCs for varying #nodes N

When increasing the number of nodes from 16 to 256, the average latency almost went up three to four times in three NoCs, but the energy went up three times in WK-recursive and 2D Mesh and six times in Spidergon.

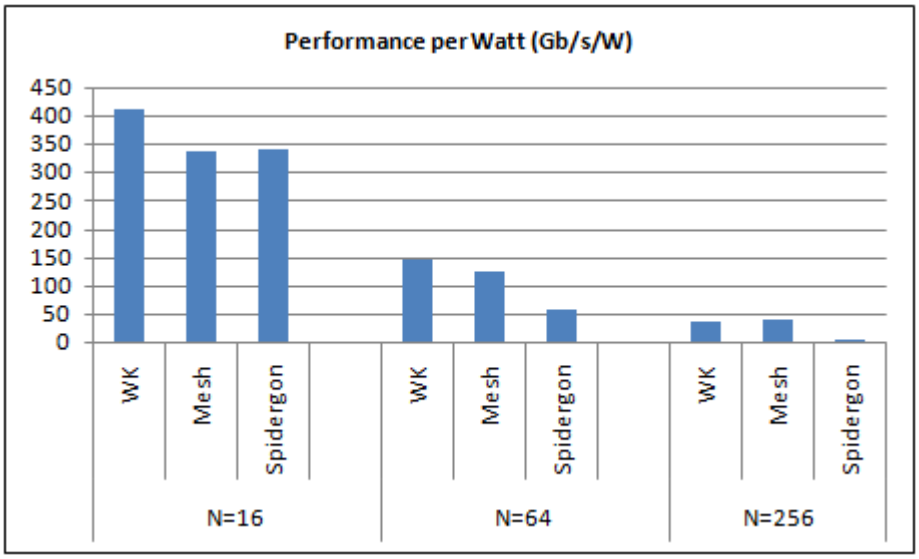


Figure 4-16: Comparison of performance per Watt of three NoCs for varying # nodes N

Figure 4-16 shows the performance per watt relative metric when to evaluate the scalability for the three NoC architectures. Again, the chart shows the same pattern, but

is more useful to show how this metric deteriorate for the Spidergon, thereby giving a clear illustration of the general conclusion.

In the scalability study, WK-recursive and 2D Mesh NoC scales well since the increasing number of nodes from 16 to 256 lead to approximately three times increase in average latency and energy. However, Spidergon doesn't scale especially for energy which is of prime importance in NoCs.

Chapter 5 Field Adaptable NoCs (FAN)

5.1 Introduction to NoC Customization

NoCs are generally built based on regular networks with fixed topologies. This is primarily due to the fact that building a fully customized network such as in [32] is time consuming and only suites a specific application, which is restrictive as it prevents reusability [57]. In addition, electrical parameters become unpredictable for fully customized networks.

While NoCs are usually built around regular networks with defined topologies, they may suffer from high latency because of long routes traversed by packets exchanged between distant, communicating nodes. In real time applications, where the timing deadlines are firm, this creates a considerable challenge [28].

To address this issue, NoC systems need to be optimized with respect to the architectural topology as well as application-specific requirements. Several studies have investigated the architectural aspect by evaluating different types of topologies and structural configurations that best fit the unique needs of NoC systems. These studies have employed the tools and techniques that are usually used in regular computer networks. Simulations of different architectural configurations are primarily used in the estimation of various performance metrics, thereby helping in the selection of a suitable NoC architecture. Based on the studies in the last chapter, one of the promising structures is the WK-recursive network [8]. It has attractive features such as scalability, self-routing and low diameter, making it suitable for NoC implementation. Therefore, it

will be considered in this chapter along with another popular and efficient network, the 2D Mesh.

Despite the advantages of having a regular network, there is a crucial need to customize the architecture for application specific communication workloads to improve the performance [28]. Techniques and tools are required to extract application communication requirements and to efficiently customize the network by physically inserting additional links. In order to optimize a NoC system to achieve the best performance for a specific application, the traffic patterns of the application need to be analyzed. This allows for identification of congestion points and areas of highly concentrated traffic, leading to solutions that reroute traffic to different locations or areas on the chip [28]. Another type of adaptation is the use of different routing and switching techniques based on these traffic patterns.

The alternative is to have an intermediate approach wherein the main system features are not completely altered. Recent research has attempted to optimize NoC architectures by adding long-range links to the base network architecture, based on the communication requirements of the application [57]. Researchers have proposed a methodology that starts with a regular on-chip interconnect architecture like the 2D Mesh and partially customizes it by inserting a few application-specific long range links. Although the application specific link insertion is meant for SoCs with Mesh topology, the technique is suitable for other architectures as well. However, extending an architecture based on the application makes the NoC application-specific, which is somewhat restrictive. For the majority of the cases when the application is chosen at deployment, an adaptable NoC

structure is desirable. We therefore propose the augmentation of NoCs with programmable links, allowing in-the-field adaptation.

5.2 Methodology for Customization

A methodology for adapting NoC topologies is presented here, resulting in a Field-Adaptable NoC (FAN). In order to achieve the required adaptability in the field, a two-phase approach is adopted. In the first phase a network-centric approach is used, which is carried out once during the design/implementation of the NoC. The base NoC structure is extended by adding links in strategically selected locations. The added links are programmable through their associated simple routing infrastructure, allowing in-field customization. In the second phase, the customization is carried out and can be based on the traffic pattern analysis of the application. Application traffic analysis is carried out offline to characterize different application needs, but the adaptation is carried out in the field depending on the application selected for execution, thereby improving performance while maintaining generality for targeting a wide range of applications.

5.2.1 Network-centric infrastructure

In order to demonstrate our FAN approach, we first consider the WK-recursive NoC. One of the main advantages of the WK-recursive interconnect is the ability for the structure to be extended without making major changes in the nodes or routers. However, sometimes there is a need to modify the structure beyond increasing the number of the processing elements. For example, consider a traffic pattern that heavily uses the route from 11 to 03, in Figure 5-1. All of the flits/packets in this example will traverse many

hops before reaching their destinations. If we introduce direct links between heavily communicating distant nodes, there is a significant potential for a performance improvement. Indeed, some studies carried out in [57] for regular Mesh networks indicate the benefits of inserting application-specific long range links.

Even though insertion of long-range links is desirable, a redesign of the NoC for every application is clearly infeasible. To enable link insertion in the field for applications that need it, a new methodology is proposed, FAN. When we apply it to WK-recursive networks, the resulting derivative is termed here as AWKR, or Adaptable WK-Recursive NoC architecture. It should be noted here that AWKR is just a case study, and the methodology is applicable for other topologies as well.

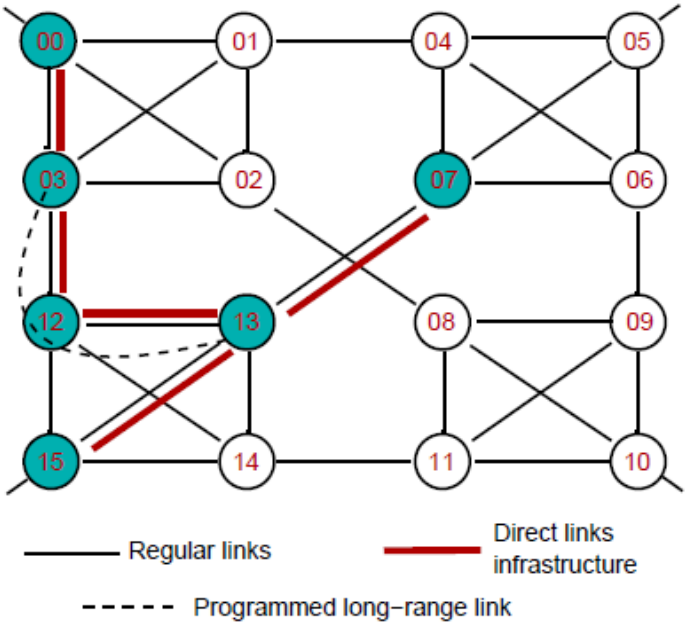


Figure 5-1: Illustration of augmentation of WK-recursive topology, to obtain AWKR

The basic idea in the proposed AWKR is to overlay a fixed network on top of the WK-recursive network, to provide the required direct links among distant nodes. The

augmented portion is not a circuit-switched network; it is instead a set of fixed/direct connections. Since application-specific needs will be different, it is desirable to provide some programmability in the fixed network: the direct links can be activated/de-activated as desired.

An example of AWKR is shown in Figure 5-1. The basic idea is to provide additional links in parallel with the existing links, as shown by the solid lines in Figure 5-1. By adding links in parallel with existing ones, the physical implementation maintains the original structure, and the regularity and predictability of electrical parameters is retained. The affected routers need to be modified. However, the modifications to the routers will not significantly add to the delay in routing decisions; this is due to the fact that these added links, termed as "direct links", form a separate, fixed network which is separate from the original WK-recursive structure. At appropriate routers, packets/flits can crossover between the fixed network and the regular network.

The programmability in the fixed network arises from the fact that multiple direct links can be statically programmed to be daisy-chained, in order to provide a long-range link. Consider the example shown in Figure 5-1. The nodes/routers numbered 00, 03, 07, 12, 13, 15 are part of the adaptable network of the AWKR. Separately considered, it is possible to create a direct, long-range path between any two pairs of nodes. For example, a long-range path between 03 and 13 is possible, by a static programming of the router at node 12. We can therefore say that a "virtual crossbar" exists between these six nodes. However, it may be noted that due to shared resources, it is impossible to have all pairs of long-range links active simultaneously.

To elaborate, consider that the long-range link between nodes 03 and 13 has been established. Under this condition, it is not possible to establish a direct link between 00 and 13, since it shares the path with the other direct link. If we chose to allow the direct path 00←→13 in addition to 03←→13, then both of them are no longer direct paths; this is due to the fact that a routing decision will have to be made at node 12, which effectively means we have added an additional hop. Note that when there is a long-range link between 03←→13, it is still possible to have a long-range link from 15←→07.

Another important fact to consider regarding the proposed infrastructure is that the long-range links are intended to be established statically, at the beginning of the application, that is, dynamic adaptation is not envisioned. The reasons for prohibiting dynamic reconfiguration will become clear in the description of the router design that follows. But as a consequence, it is necessary to choose the best possible long-range links from the programmable infrastructure that is available. An intelligent selection of the long-range links to be programmed can be carried out based on the requirements of the application, as will be described later in 'Phase Two' of the methodology.

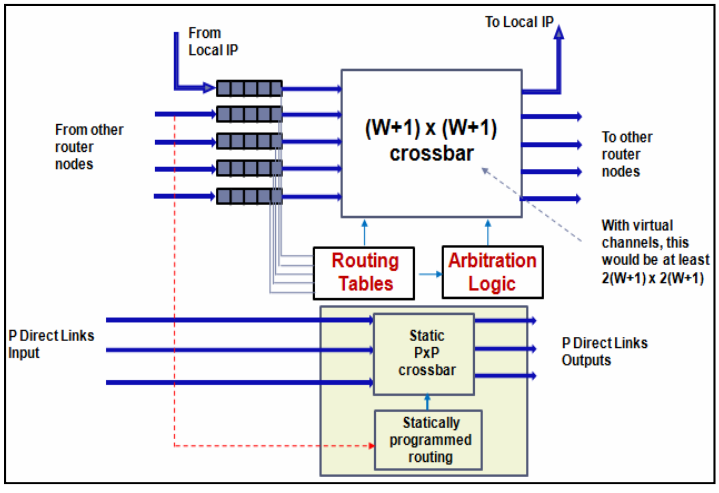


Figure 5-2: Modified router design

The routers associated with the proposed fixed network will have to be modified. An example of the original router design, prior to adding links, can be seen in the upper part of Figure 5-2, for a WK(W,L) network. It is based on the design in [20], and includes a $(W+1)\times(W+1)$ crossbar unit, routing tables, as well as an arbitration unit for cases when the same destination port is required simultaneously by flits arriving at different inputs ports.

To accommodate the additional links corresponding to the fixed network, the router is modified by adding a static $P \times P$ crossbar, where P is number of links to add, and its value depends on the designed fixed network. The modification is shown in Figure 5-2. Clearly $P \leq W$ since links are added in parallel with existing ones. Direct routes are programmed statically into the new crossbar during the initialization phase. As noted earlier, dynamic adaptation is prohibited, so the additional crossbar in the router is static. Therefore, the area overhead is very limited or negligible compared to total router size, since it does not have high-speed switching requirements (as compared to the original cross-bar, which will have much higher area requirements to support low latency routing decisions). Further, the fixed network part does not require any buffering (since connections from input to output are static), thereby further reducing the area and energy requirements. Note that there needs to be some mechanism to allow programming of the static crossbar (suggested by the dotted red line in the figure). Further, paths between the fixed network and regular network need to be provided at some of the routers; these implementation details are not shown here since they are not the focus of this work.

Given this basis for the fixed but programmable infrastructure for long-range links, it is necessary to determine the locations to add direct links to the original NoC topology.

Nodes need to be chosen from the original WK-recursive structure so that a suitable fixed network may be overlaid onto the regular network, to allow the direct connections. As discussed earlier in this section, it may not be possible to connect all the selected nodes simultaneously. In other words, a physical crossbar does not exist between the nodes selected for the fixed network; yet, a "virtual crossbar" exists, as explained earlier.

Algorithm 1: Selecting nodes to be used for constructing the augmented NoC

Input: Base network with N nodes, constraint on number of nodes (q) that should be the communication end-points for the direct links.

Output: Selection of q nodes that would give the best reduction in average distance

1.1 Build the adjacency matrix A for the base network with N nodes

1.2 $D \leftarrow \infty$;

1.3 **for** *all possible combinations of q nodes out of N nodes* **do**

1.4 Add links to A, corresponding to a "virtual crossbar" for selected q nodes;

1.5 Obtain the connectivity matrix C from A;

1.6 Compute average # hops H using C.;

1.7 **if** $H < D$ **then**

1.8 $D = H$;

1.9 Store the combination of q nodes;

1.10 **return** the stored combination of q nodes ;

The number of nodes 'q' for the fixed network needs to be decided beforehand, based on resource constraints and expected requirements of applications. Since the actual applications that will execute can be of different types, the selection of the q nodes must be carried out with the objective of providing a benefit to as many applications as possible. A metric that can provide a broad measure of performance improvement is the average number of hops (or average distance) that is a characteristic of the network. The objective, therefore, is to choose the q nodes which minimize the average distance.

For small networks, explorations of the choice of q nodes (among the available N nodes) can be done exhaustively. Algorithm 1 gives the procedure to carry out the required selection. The algorithm begins with the construction of the adjacency matrix A for the base network, the WK-recursive network in our case. In every iteration of the algorithm, one combination of q nodes is selected, and the matrix A is augmented based on the fact that a "virtual crossbar" exists between these q nodes. This is followed by a computation of the connectivity matrix C using standard techniques. The average distance of the augmented network is then computed by averaging out the values of the C matrix, i.e. [sum of all elements of C] ÷ N[N-1]. The division is by $N(N-1)$ instead of N^2, since the diagonal elements of the connectivity matrix will be zeroes.

After selecting the q nodes which give the best reduction in average distance, the direct links can be added between the routers. In doing so, direct links are added along a shortest path between pairs of the chosen q nodes.

5.2.2 Application-specific Adaptation

The augmented NoC was derived with the aim of reducing the average distance of the topology, without considering the application characteristics. It is therefore suited for a wide range of applications, thereby maintaining the generality of the architecture. However, as we have noted earlier, all possible long-range links in the augmented network cannot be activated simultaneously, due to shared resources. Although this reduces the area and energy overheads, it also calls for an intelligent selection of the links to be activated, based on the application characteristics.

To formalize the problem, let the augmented network consist of M programmable long-range links that can be activated. Then, the set of programmable links can be

denoted as $L = \{L_1, L_2, ..., L_M\}$. If we use a binary variable s_i to denote the activation state of the link L_i, then an activation set that denotes the state of all the links is $S_k = \{s_1, s_2, ..., s_M\}$. The subscript k identifies only one possible activation set, out of all possible activation sets. Since there are M possible links, the total number of possible activation sets is $P = 2^M$. In other words, one of the activation sets among $S_1, S_2, ..., S_P$ can be used for programming the augmented network. However, some of the activation sets are illegal, due to the shared resource constraint noted earlier. For example, $S_P = \{1, 1, ...1\}$ is not allowed, since it is not possible to activate all links simultaneously.

Whenever an augmented network is obtained using the methodology given in previous subsections, it is possible to immediately obtain the number of links M, the number of activation sets P, as well as the forbidden set F consisting of the indices k for which S_k is illegal. The forbidden set, for example, could be $F = \{6, 11, 13,.., P\}$. We can be sure that P is a member of the forbidden set F, for reasons just described earlier.

Based on the above formalism, the intelligent selection of links boils down to a suitable selection of the activation set S_k. For solving this optimization problem, a suitable metric needs to be chosen. The average packet latency for a given application is a good measure of the performance of the application. Indeed, it has been shown in [57] that a reduction in the average latency results in an increase in the critical traffic load, thereby improving the usefulness of the network.

The average packet latency for an application depends on the application traffic pattern – which in our case essentially consists of the number of packets injected at each node, as well as the number of hops traversed by each packet. To minimize the average

packet latency, we need to equivalently minimize the application-specific average distance D_{app} :

$$D_{app} = \frac{\sum_i \sum_j n_{ij} h_k(i,j)}{\sum_i \sum_j n_{ij}} \tag{12}$$

where n_{ij} is the number of packets sent from node i to node j, and $h_k(i,j)$ is the number of hops traversed by a packet in going from node i to node j. In our work, we use static routing tables based on Dijkstra's all-pairs shortest path first (SPF) algorithm. Therefore, $h_k(i,j)$ is constant for each packet traversing the route $i \rightarrow j$, and depends only on the activation set S_k.

Algorithm 2: Determining the links to activate in the augmented NoC

Input: Augmented network with N nodes and L possible direct links, forbidden set F, and traffic pattern $nij, i, j = 1, \ldots, N$

Output: The activations set S_c consisting of best possible link activations for given application

2.1 Compute$\{h_k(i,j), i,j = 1, \ldots, N\}\ \forall S_k, k \notin F$;
2.2 *Initialize* $c \leftarrow 0$, $D_{min} \leftarrow \infty$;
2.3 *Initialize* $L \leftarrow L1, \ldots L_M$;
2.4 **while** L is *non-empty* **do**
2.5 Fine Link $l \in L$ that gives min D_{app} using(1);
2.6 Record l in the activation set S_c;
2.7 Remove l from L
2.8 Find links in L that conflicts with l(based on F);
2.9 Remove conflicting links from L;
2.10 **return** S_c ;

A greedy algorithm for obtaining the activation set is shown in Algorithm 2. This algorithm starts with the computation of shortest path routes $h_k(i,j)$ for different flavors of the network (different indices k), as long as it is not part of the forbidden set F. Then the long-range link which yields the minimum application-specific average distance is

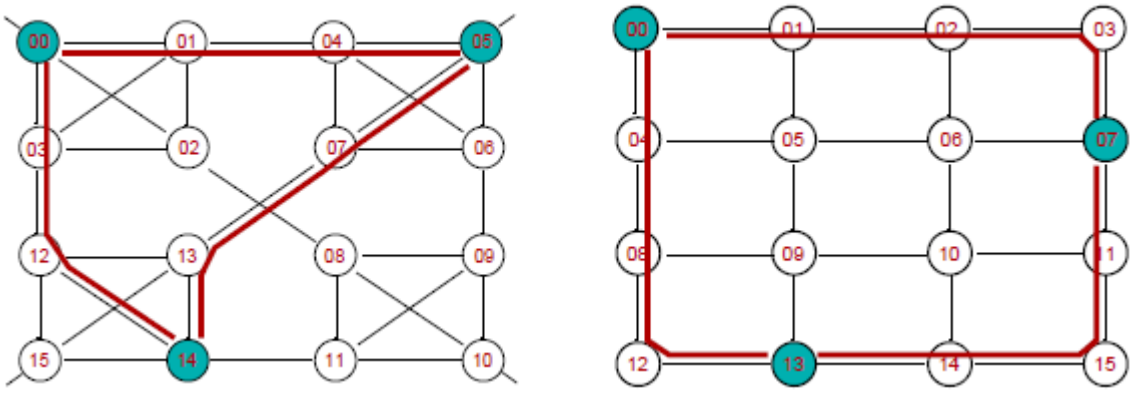
chosen. Based on this link choice *l*, one or more links that cannot be active along with *l* are removed from consideration. The process then repeats and links are greedily chosen for activation till the algorithm terminates. It should be noted that the use of an analytical estimate of the application performance (line 2.5) considerably speeds up the exhaustive search process. It would have been much slower if simulation using a network simulator (such as NS-2) was carried out for each iteration in the algorithm.

5.3 Methodology Evaluation and Results

The methodology described in the previous section is suitable for deriving an adaptable NoC for any base network. The described approach is used for building AWKR. In addition, the approach is also used here for obtaining an adaptable Mesh topology, since Mesh is the most common structure used for NoCs. Comparisons and evaluation of AWKR and Adaptable 2D Mesh (AMesh) are presented.

For the WK-recursive network, all evaluations are carried out for the WK(4,2) network topology shown in Figure 5-1. We have used values of q=3 and q=4 since the network size is small. It may be noted that the value of q is governed by the resource constraints (in terms of overhead that can be tolerated) as well as the number of nodes in the network.

Based on Algorithm 1, discussed earlier, for WK(4,2) and q=3 nodes, there are 24 possible options that can give the best reduction in average distance for the network. The huge number of choices is due to the regular, recursive structure. If the processors in the network are identical, any of the options can be chosen.



(a)WK-S1 **(b) 2D Mesh-S1**

Figure 5-3 : Direct links infrastructure, *q*=3 (a) WK and (b) Mesh

One of these options is shown in Figure 5-3(a), with the selected q nodes shown highlighted. The fixed network is constructed along the red lines overlaid on top of the base WK-recursive network as shown. Note that this example is a relatively simple case and the long-range links that are shown in red are constructed by using three links each. The number of options for activating long-range links is limited in this case. Theoretically, the average distance of the network is reduced from 2.2 to 2.03, or equivalently, 7.58 %. For q=4 nodes, there is only one option for WK(4,2), and that option uses the corner nodes. The average distance in this case is reduced from 2.2 to 1.9 or, 13.64 %. Note that improvements for WK are not large, since it is already efficient and has a low diameter. On the other hand, the Mesh network gives a reduction in average distance from 2.67 to 2.2 for q = 3 (i.e. 17.5 %), and for q = 4, the theoretical reduction is from 2.67 to 2.05, or 23.13 %. This is a significant improvement by the augmentation. One among the 4 available options for the Mesh network is shown in Figure 5-3(b) for q = 3.

Referring to the chosen network configurations in Figure 5-3 as ***S1*** and the baseline without augmentation as ***S0***, we carry out some experimental evaluations using synthetic generators and real workloads.

5.3.1 Synthetic applications

First, we study the advantages of S1 over S0, with respect to two synthetic applications on a discrete event simulator. The simulator is derived from the NS-2 simulator [51] as described in [25]. The NoC parameters used for the simulations using NS-2 are given in Table 5-1. A constant-bit rate traffic generator in the NS-2 simulator is used for the synthetic applications.

Table 5-1: On-chip interconnects parameters

On-Chip Interconnects (OCI)	WK-Recursive/2D Mesh
Number of PEs	16
Link delay (d_ℓ)	0.1ms
Maximal link capacity ($Bw_{\max}$).	800Mbps
Buffer size inside PEs	Unlimited
Buffer size inside routers	32 packets
Type of the buffer size	Drop-tail queue

Two applications are generated as follows: (a) SYN1 represents traffic that is distributed uniformly on all 16 nodes and (b) SYN2 represents traffic that is from the nodes in the top half of network to the nodes in the remaining half. An injection rate of 20 Mbps is used. All simulations use a packet size of 8 bytes.

The NoC metrics that are selected for evaluation are latency and energy. Average packet latency is obtained from the simulation trace of the NS-2 simulator by summing up the latency (in ms) incurred for each packet and then averaging it out. The energy estimates are also inferred from the traces by adopting the energy model introduced in chapter 4, and computing the energy for every packet separately and then summing them

up. The energy depends on the packet size, number of hops traversed by a packet from source to destination, as well as the physical link length in the network.

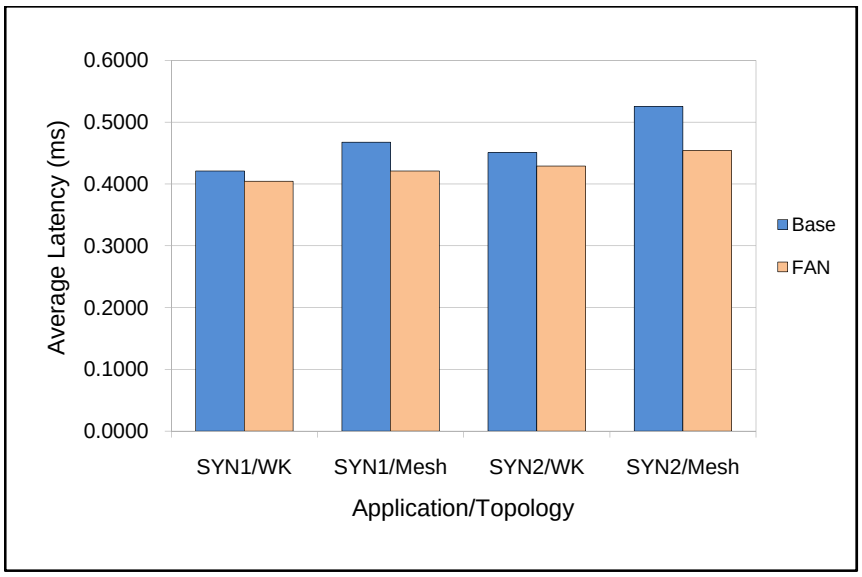


Figure 5-4: Latency comparison for two versions of application traffics, n=16 nodes

Figure 5-4 shows the latency for the two applications when simulated using the base configuration S0 (labeled as 'Base') and the adaptable configuration S1 (labeled as 'FAN'). A latency improvement of up to 5% for AWKR and up to 13:7% for AMesh are observed.

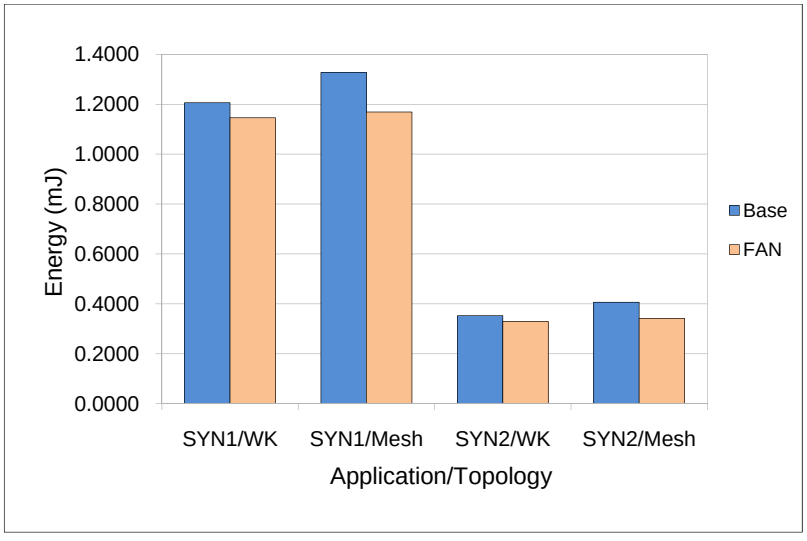


Figure 5-5: Energy comparison for two versions of application traffics

The improvement in the average number of hops for SYN1 (not shown) exactly matches the theoretical improvement in the average distance, since the traffic is uniform. For SYN2, larger improvements in average number of hops are seen (8.75% for WK, 22.12% for Mesh). This is an expected result, since SYN2 restricts communication to distant nodes belonging to the other half (top or bottom half) of the network.

Figure 5-5 shows the evaluation of the energy for both applications. It can be noticed that the improvement depends on the number of packets (higher for SYN1 since all nodes are transmitting). Mesh exhibits the largest improvement in energy, 16% for S1 compared with S0, for the application SYN2.

5.3.2 Simulation of real workloads

To demonstrate the applicability of the methodology as well as the simulation framework, we have simulated two real applications – the Fast Fourier Transform (FFT) and MultiGrid (MG) applications from the NAS parallel benchmark suite [58]. Since NS-2 simulator does not directly support execution of parallel applications, the FFT and MG applications were first run on a cluster of 16 nodes, based on Fortran and MPI. The communication traces of the applications were captured by instrumenting the code using the MPICL tool [53]. In order to feed this communication trace to NS-2, the traces were postprocessed using ParaGraph which is part of MPICL and then decomposed into individual traces for every source-destination pair. NS-2 was modified and rebuilt to ensure that the traces are taken from the beginning of the files, instead of random positions as per default behavior of NS-2. The packet size was restricted to 128 bytes, and any larger messages were broken down into 128 byte packets transmitted every 4 μs,

which is well within the maximum link bandwidth simulated (Table 5-1). The results obtained from the simulations for these two applications are given in Table 5-2, in terms of the reduction in the average number of hops traversed by the packets. Since FFT has a uniform all-to-all traffic distribution, it is seen that the reduction in average number of hops is higher. On the other hand, MG has more localized communication.

Table 5-2. Comparing average #Hops for two real applications on 2D Mesh

Application	S0 (Baseline)	S1 (Adapt)	% Reduction	
NPB-MG	1.4782	1.4067	4.84%	
NPB-FT	1.5981	1.5064	5.74%	

Improvements for WK topology were insignificant and hence not reported here. To demonstrate the true potential of our proposed methodology, we consider the AWKR network for $q = 4$. It turns out that Algorithm 1 results in only one option, the selection of the four corner nodes. The selected nodes are highlighted in Figure 5-6(a).

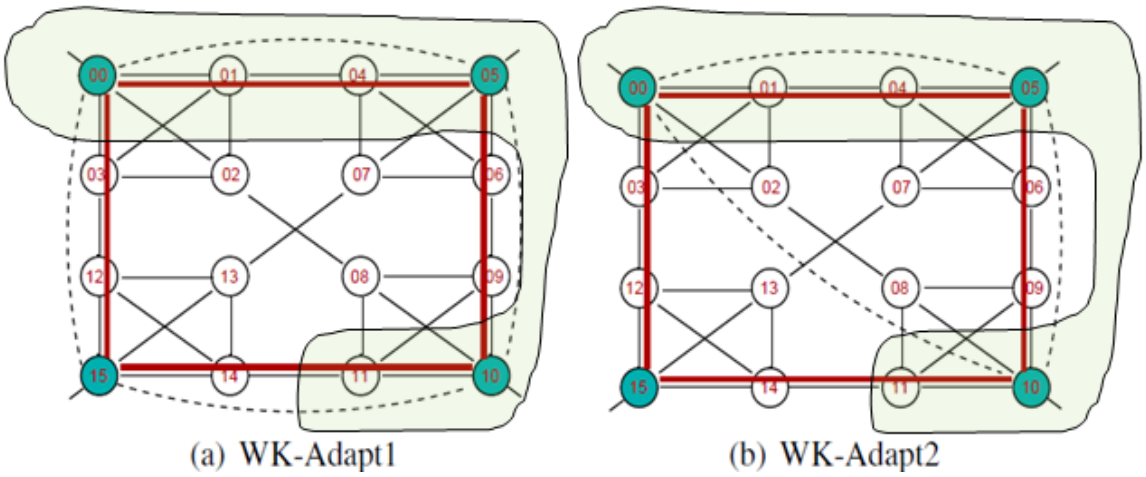


Figure 5-6: Two link activation options for q = 4, WK topology. Legend used is same as in Figure 5-1

There are several options that can be chosen for activating the long range links in the network shown. The selection of links to activate has to be carried out using Algorithm 2, based on the application. We consider a synthetic application that has nodes 00, 01, 04,

05, 10, and 11 communicating in an all-to-all fashion, each node injecting traffic at 20 Mbps, for duration of 45 ms. The communicating nodes are indicated by the bubble in Figure 5-6. Using the considered traffic pattern, Algorithm 2 selects the link activation as shown in Figure 5-6(b) – a fixed path is programmed in node 15 to allow the direct link 00←→10, in addition to the activation of other direct links 00←→05 and 05←→10. The selected active links matches the intuitively expected result, since traffic is concentrated among the 6 nodes considered. We experimentally verified it by simulating the activation suggested by Figure 5-6(a) in addition to that in Figure 5-6(b).

Table 5-3: Average #Hops and Energy for two AWKR configurations, *q*=4

Parameter	Baseline	Adapt1	% Reduction	Adapt2	% Reduction
# Hops	2.3333	1.8667	20.00 %	1.6667	28.57 %
Energy (mJ)	0.1568	0.1334	14.92 %	0.1252	20.15 %

The simulation results are shown in Table 5-3. The results show a significant improvement over the baseline, and demonstrate that Algorithm 2 activates the appropriate long-range links. The results also show that depending on the traffic pattern, significant improvements in the number of hops and energy can be achieved (28.57% in hops and 20.15% in energy for this example).

Static adaptability in FAN can be leveraged for not only a single application, but rather a class of applications that share common characteristics (application domain). A given flavor of FAN will suit one application domain over the other. To demonstrate this idea, we consider two application categories: HTTP applications and streaming applications. In HTTP applications, users require images and text; the generated traffic can be modeled by Markov On-Off arrivals, with ON and OFF periods varying according to an exponential distribution. Streaming applications such as HDTV and MPEG video

comprise large data blocks and must respect strict time intervals. The traffic generated by these types of applications is modeled with Pareto On-Off distribution. This traffic model possesses self-similarity, which is a fundamental property exhibited by the bursty traffic between on-chip modules in typical MPEG-2 video applications.

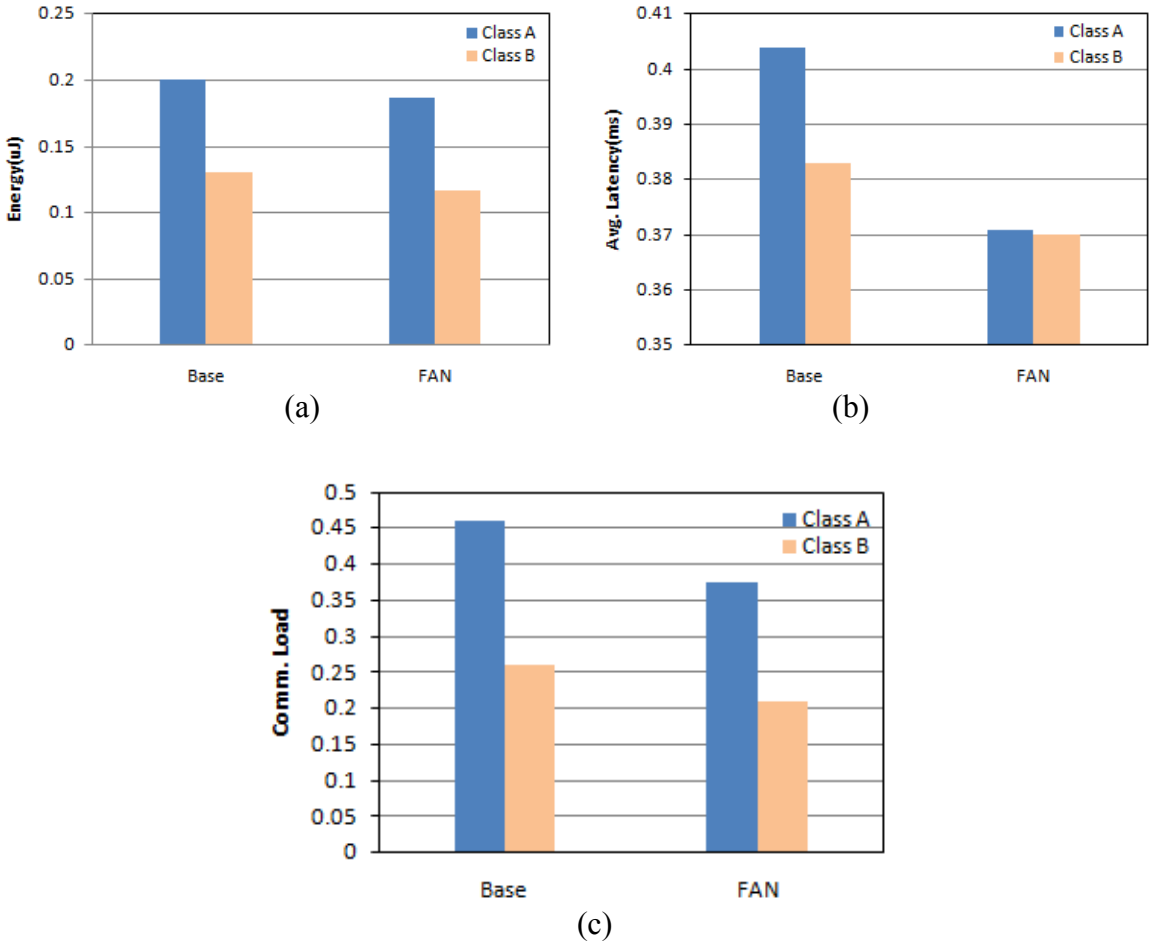


Figure 5-7: Comparing two applications domain on FAN, Class A and class B: (a) Energy, (b) Latency, and (c) communication load

To study the performance of these two application domains under a given FAN configuration, simulation experiments have been conducted with the HTTP application domain, shown here as Class A applications, and Streaming applications, shown as Class B applications. Figure 5-7 shows the two classes for latency, energy, and load. Clearly, different improvements in latency are seen for Class A and Class B, comparing the FAN

configuration S1 against the base configuration S0. An improvement of up to 20% in load for class B is observed, compared to 17% for class A applications. The results clearly demonstrate varying levels of performance achieved for different application domains, for a given FAN. This also demonstrates that further improvements in performance may be achieved by building a FAN that is customized for a particular class of applications, while at the same time retaining the adaptability in order to cater to the different applications within the application domain.

Chapter 6 - Design Space Exploration of NoCs Using Network Calculus

Performance analysis and evaluation of on-chip interconnect (OCI) architectures are widely based on system level simulation techniques. Simulation can incur very high computational costs, especially for large-scale NoCs. Furthermore, simulation-based approaches do not generally yield fundamental insights into the behavior of the network [28]. To carry out rapid design space exploration (DSE), and to allow designers to detect and locate network contention and bottlenecks in the early stages of the design process, an effective traffic analysis methodology is required. Due to the time-consuming nature of simulation-based approaches, as well as their limitations in providing an understanding of the effects of various design parameters, analytical models for NoCs are essential.

In this chapter, we introduce Network Calculus as a suitable analytical modeling tool for estimating and evaluating the performance of on-chip interconnects. This tool was originally proposed for studies involving the quality of service in the internet [47,48,59] and has later been used in many other fields such as embedded systems for network processors [60,61], switched Ethernet networks [62], sensor networks [63], and optimal routing [64].

The main objective of this chapter is introduce the use of Network Calculus as a high level modeling tool that allows the designer to carry out a systematic and efficient performance comparison of on-chip interconnect architectures.

First, the fundamentals of Network Calculus are introduced, followed by a description of the proposed methodology for modeling NoCs. Using a sample application workload for a 2D Mesh topology, performance modeling using Network Calculus is demonstrated. In order to validate the model, the results are compared with simulations using the same

application workload, thereby demonstrating the effectiveness of using Network Calculus in performance analysis and evaluation. Finally, analysis and comparison of three topologies, namely, 2D Mesh, Spidergon and WK is carried out for different traffic loads. The conclusions derived from the analytical comparison closely matches those obtained from the empirical study in the previous chapters, which helps validate the model. Network Calculus is thus shown to be a useful tool in NoC evaluation. Moreover, design parameters such as buffer size, which cannot be directly obtained using simulation, are estimated quickly using the proposed analytical approach.

6.1 Background and Motivation

On-chip interconnect architectures adopted for SoCs are characterized by trade-offs between different parameters such as latency, throughput, communication load, energy consumption, and silicon area requirements.

Several works, such as that presented in [42], have demonstrated that there is a crucial need for system design tools and methodologies to compare NoC architectures for a given application. The authors note that simulation-based methods, despite their accuracy, are very expensive and time consuming. Techniques and tools are required to extract the communication requirements of applications and to efficiently estimate the performance for different interconnection architectures.

To this end, in [34], an analytical model using queuing theory was introduced to evaluate traffic behavior in Spidergon NoC. Simulation results to verify the model for message latency under different traffic rates and message lengths were reported.

A queuing-theory-based model for evaluating the average latency and energy consumption of on-chip interconnects was proposed in [43]. The results from the

analytical model were validated with those obtained from a cycle-accurate simulator. In [65], a generalized router model was proposed using a probabilistic approach. The model was used for carrying out NoC performance analysis and for analyzing the mapping of applications on NoCs.

Probabilistic approaches, such as queuing theory, model the network elements in an equilibrium state, and provide estimates on the average latency experienced by packets and the average number of packets held in the system (a router, for example). However, steady state analyses fail to capture the dynamic characteristics of network traffic, especially when certain QoS requirements have to be met in the midst of transients. Furthermore, NoCs must be designed to handle worst-case scenarios; for example, the buffer within a router must be sized so that it can handle cases when traffic from multiple input ports exhibit burstiness.

Network Calculus models the flow of traffic, as well as the service offered by network components, in terms of bounds, thereby enabling worst-case analyses to be carried out. The mathematical framework provided by Network Calculus is useful for obtaining computational rules that bound the delay and backlogs in networks, thereby allowing the designer to estimate the worst case latency and buffer requirements. Moreover, the Network Calculus framework transforms a non-linear system, such as a queue, into a linear system allowing a simplified analysis.

In this chapter, Network Calculus is presented here as a tool for analyzing and evaluating the performance of on-chip interconnects. Analytical studies are conducted to show the potential of this technique. The studies demonstrate that Network Calculus can

be used to select a suitable topology among candidate architectures, as well as for choosing the suitable network parameters such as buffer size.

Three on-chip interconnects, 2D Mesh, Spidergon, and WK, under different workloads, are considered for the performance evaluation. Simulations are carried out for the 2D Mesh network in order to validate the results obtained from the proposed analytical model. Results using the analytical model for the three topologies are then compared in order to demonstrate the usefulness of Network Calculus for evaluating on-chip interconnection networks.

6.2 Network Calculus: An Overview

Network Calculus [47,48] is a modeling framework that allows designers to specify a system as a mathematical model and evaluate performance bounds of important metrics such as end-to-end delay. The theory is based on the abstractions of "Arrival Curve" for input and output traffic flows, and "Service Curve" for the network components. Using these abstractions, it is possible to compute the bounds on the worst-case delay and backlog. These estimates, when applied to each router in a NoC, can be used to determine the latency and buffer requirements.

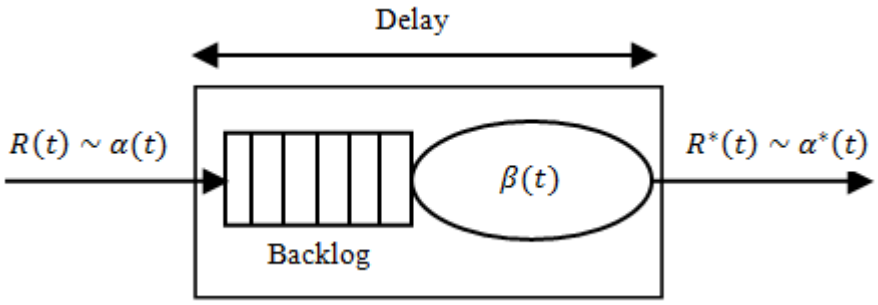


Figure 6-1: Arrival and service curves in Network Calculus with delay and backlog bounds

Table 6-1: Summary of the notations used for the analytical model based on Network Calculus

t	Time (seconds)
$R(t)$	Input traffic function, cumulative data units seen on the traffic flow in $[0, t]$.
$R^*(t)$	Output traffic function for the input $R(t)$, cumulative data units observed on the flow in $[0, t]$.
$x(t)$	*Backlog,* amount of data units that are held inside the system at time t
$d(t)$	The *virtual delay,* or the delay incurred by a flit that arrives at time t
$\alpha(t)$	Arrival curve that characterizes the input flow $R(t)$
$\alpha^*(t)$	Arrival curve that characterizes the output flow $R^*(t)$
$\beta(t)$	Service curve of the network subsystem
$(f \otimes g)(t)$	Convolution operation in Min-plus algebra, given by $inf_{0 \le s \le t} \{f(t-s) + g(s)\}$
$(f \oslash g)(t)$	Deconvolution operation in Min-plus algebra, given by $sup_{s \ge 0} \{f(t+s) - g(s)\}$
r	Traffic rate (bit/s) for a flow
b	Burst parameter (bits) for a flow
R_s	Guaranteed service rate (bits/s) offered by a network component
T	Minimum delay (seconds) experienced by a flit when serviced by a network component
k	Flit size (bits)
r	data transmission rate
D	Upper bound on the delay
B	Buffer size requirement, or upper bound on backlog
D_j	Upper bound on delay at router s_j
B_j	Buffer size requirement for router s_j
$\alpha_p(t)$	Arrival curve associated with a flow f_p at the input of a router
$\alpha_p^*(t)$	Arrival curve associated with a flow f_p at the output of a router
$\overline{\alpha}_j(t)$	Aggregate input arrival curve for all traffic arriving at router s_j

We begin with an overview of the general terminology used in a system under the Network Calculus framework. Any network is comprised of one or several components that exchange data in order to accomplish a given task. As a result of this data exchange,

there are multiple data flows that are established between source and destination nodes. All network components involved in the data exchanges are essentially subsystems that accept an input traffic flow and shape the traffic in order to generate an output flow. An illustration of a network subsystem or component is shown in Figure 6-1. The notation used in this chapter is summarized in Table 6-1.

The figure shows a subsystem 'C' with input traffic represented by the accumulated input traffic flow $R(t)$. This input traffic flows through the subsystem and eventually generates traffic at the output port, measured by the output traffic flow function $R^*(t)$. Both functions are monotonically increasing functions with respect to t. With these input and the output functions, we can derive the two quantities of interest, namely, the *backlog* and the *virtual delay* [35]. The *backlog* $x(t)$ is the amount of data units held inside the system, i.e. $x(t) = R(t) - R^*(t)$. The *virtual delay* $d(t)$ is the delay that would be experienced by a data unit arriving at time t if all units received before it are served before it; mathematically, this is given by $d(t) = inf\{\tau \geq 0 \mid R(t) = R^*(t+\tau)\}$.

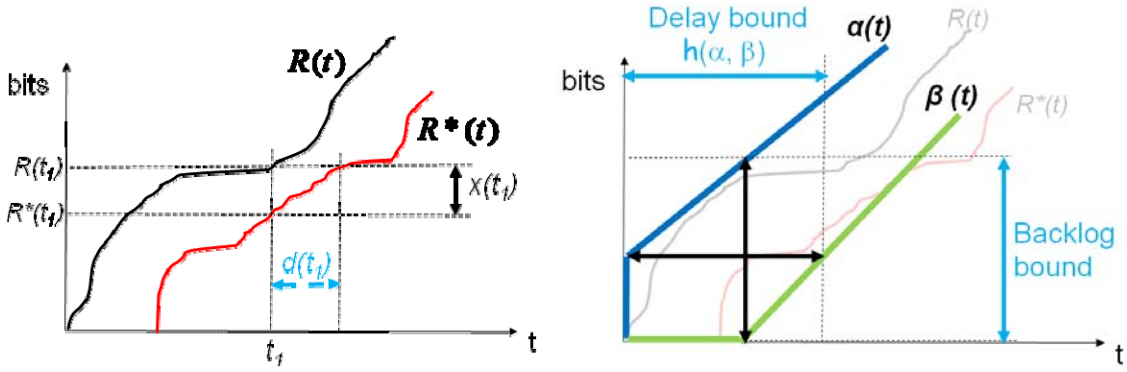


Figure 6-2: Illustration of (a) the concepts of backlog and virtual delay and (b) the bounds on delay and backlog based on arrival and service curves

The delay $d(t_1)$ and backlog $x(t_1)$ at time t_1 are illustrated in Figure 6-2(a). The values of these parameters depend on the time instant $t_1$considered. In order to obtain bounds on these parameters, the concepts of arrival and service curves are useful. Each traffic function can be characterized by an arrival curve as follows. An arrival curve $\alpha(t)$ characterizes a traffic flow $R(t)$, iff it upper-bounds the amount of data arriving through this traffic flow during any time interval $[0, t]$. More formally, given a wide-sense increasing function $\alpha(t)$) defined for $t \geq 0$, we say that a flow $R(t)$ is constrained by $\alpha(t)$ iff for all $s \leq t$: , $R(t) - R(s) \leq \alpha(t - s)$. It is also said that $R(t)$ has α as an arrival curve [35]. A similar definition holds for the arrival curve $\alpha^*(t)$ for the output $R^*(t)R^*(t)$.

The service curve $\beta(t)$ characterizes the component C under consideration. The output function $R^*(t)$ for the component C can be calculated based on the effect of the service curve $\beta(t)$ on the input function $R(t)$. We say that C offers to the flow R, a service curve β (non-decreasing function) such that $\beta(0) = 0$ iff $\forall t \geq 0$, $R^*(t) \geq$ $inf_{0 \leq s \leq t}$ $\{R(s) + \beta(t - s)\}$.

Based on the definitions of the arrival and service curves, it is possible to show that the upper bound D on the delay $d(t)$ is given by: $D = h(\alpha, \beta) = sup_{s \geq 0}$ (inf$\{\tau \geq$ $0: \alpha s \leq \beta s + \tau)$ [35]. Similarly, the backlog $x(t)$ is upper-bounded for all t such that $xt \leq$ $sup_{s \geq 0}\{\alpha(s) - \beta(s)\}$. Figure 6-2(b) illustrates the concepts of the bounds; if the arrival curve and service curve for the system are as shown, then the delay bound is the maximum horizontal distance between the two curves, and the backlog bound is the maximum vertical distance between the two curves [47].

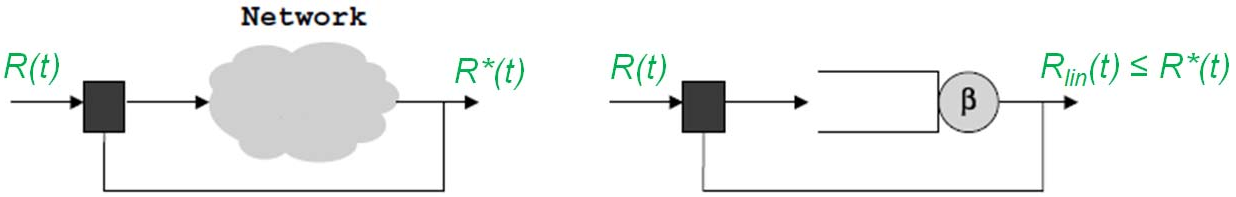


Figure 6-3: Linearization of a network subsystem by using the service curve β. The output of the linear system, $R_{lin}(t)$, is a lower bound on the output traffic flow R*(t). Figure adapted from [47].

The concepts of arrival and service curves can be compactly represented using Min-plus algebra, in which the *addition* (+) and *multiplication* (×) operators from standard algebra are replaced by the minimum (^) and addition (+) operators respectively. Based on these operators, it is possible to obtain equivalent expressions in Min-plus algebra for operations such as convolution that are defined in standard algebra. For example, given f and g monotonically increasing functions and $f(0) = g(0) = 0$, their convolution is defined as $(f \otimes g)(t) = inf_{0 \leq s \leq t} \{f(t-s) + g(s)\}$. Another operator, called the deconvolution operator, is defined by $(f \oslash g)(t) = sup_{s \geq 0} \{f(t+s) - g(s)\}$.

Based on the compact representation in Min-plus algebra, and by deriving analogies from standard linear systems theory using the convolution/deconvolution operators, it is possible to arrive at simplifications to characterize the network behavior. The first analogy with standard linear systems theory is observed in the Min-plus definition of the service curve: $\beta(t)$ must satisfy $R^* \geq R \otimes \beta$. In other words, it is possible to replace the non-linear network subsystem C by a linear system that provides a lower bound on the output traffic flow, as illustrated in Figure 6-3.

Other simplifications using Min-plus algebra are as follows. The arrival curve $\alpha(t)$ for the traffic flow $R(t)$ needs to satisfy the condition $R \leq R \otimes \alpha$. Furthermore, the arrival curve $\alpha^*(t)$ for the output $R^*(t)$ can be shown to be given by $\alpha^*(t) = \alpha(t) \oslash \beta(t)$. Another important property that can be derived based on Min-plus algebra

is that, for a cascade of two systems with service curves $\beta_1(t)$ and $\beta_2(t)$, the effective service curve is $\beta_1(t) \otimes \beta_2(t)$. There are several other important properties that simplify the computations in Network Calculus, as explained in [59,66].

6.3 Proposed Methodology for Modeling NoC Architectures

In this section, we present the use of the Network Calculus methodology for analyzing on-chip interconnects and evaluating various performance metrics. With the knowledge of the arrival and service curves for each component in the network, it will be possible to calculate performance metrics such as end-to-end delay and buffer size requirements.

6.3.1 Network Calculus Analysis for a Router with Single Flow

We consider that an application has already been mapped to a set of processing elements that comprise the NoC. Efficiently mapping an application to a set of computational components (or cores) is beyond the scope of this study. Tools and techniques such as that described in [67,68] can be used in the application mapping process.

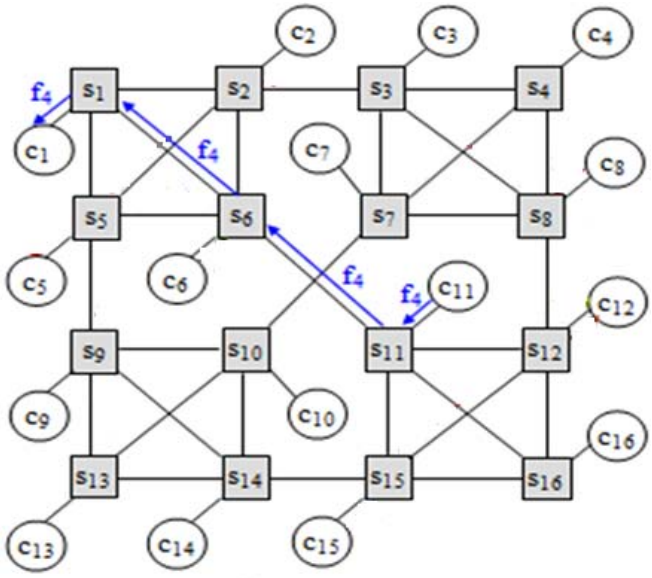


Figure 6-4: Illustration of a single traffic flow in a NoC [72]

For the purpose of analysis, all traffic in a NoC is decomposed into infinite streams of unidirectional traffic between source and destination nodes. Communication between two cores is characterized by flows that are represented by sequences of hops. For example, in Figure 6-4, the flow $f_4 = (c_{11}, s_{11}, s_6, s_1, c_1)$ represents communication from core c_{11} to the core c_1. The path traversed by the flow is statically determined, based on the network topology.

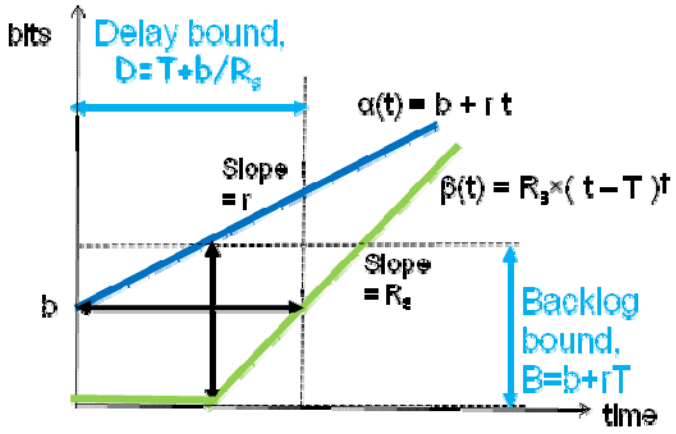


Figure 6-5: The input arrival curve α(t), service curve β(t) and output arrival curve α*(t) for a NoC router. The backlog bound and delay bound are shown.

Each router or switch along the path of the flow comprises a network component that can be analyzed using Network Calculus. Each router takes an input arrival curve and transforms it into an output arrival curve, incurring a maximum delay D and a backlog of B. Each flow, at its source, is assumed to consist of a controlled leaky bucket flow generator, with the arrival curve given by $\alpha(t) = b + rt$. This arrival curve signifies that no more than b data units can be sent at once, while maintaining a data transmission rate r in the long-term. Each router is assumed to have a service curve characterized by the rate-latency curve $\beta(t) = R_s \times (t-T)^{\dagger} = R_s \times \max(t-T, 0)$. This service curve is a standard curve as per the Internet Engineering Task Force (IETF). The arrival and service curves are depicted in Figure 6-5. These two curves are widely used in evaluating systems [48,62,63,64]. Based on the discussion in the previous section, the bounds on the delay D and backlog B for the router can be easily shown to be:

$$D = T + b/R_s \quad (1)$$

$$B = b + r\,T \quad (2)$$

The output arrival curve α* can be derived based on the fact that $\alpha^*(t) = \alpha(t) \oslash \beta(t)$. Using the expressions for $\alpha(t)$ and $\beta(t)$ and applying the deconvolution operator, as shown in Figure 6-5, we get

$$\alpha^*(t) = b + rT + rt \quad (3)$$

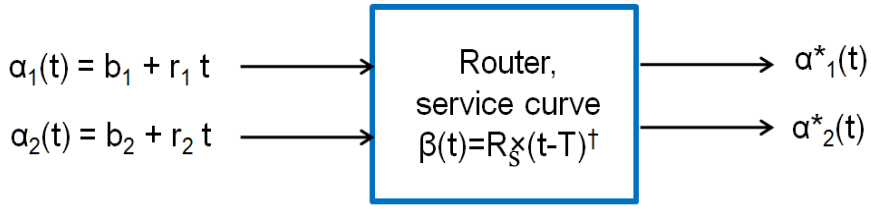


Figure 6-6: A router handling two input flows.

It is noteworthy that the same expression can be obtained based on the buffer requirement given in equation (2). Since the buffer stores $B_i = (b + rT)$ bits, it gives rise to a burst component b+rT in the output arrival curve $\alpha^*(t)$; the output data rate and the input data rate are the same, therefore the component "rt" is carried forward from the expression in $\alpha(t)$ to $\alpha^*(t)$. In other words,

$$\alpha^*(t) = (\text{router buffer size}) + rt \tag{4}$$

This simplification based on the buffer requirement will be used in deriving the output arrival curve when multiple flows are involved.

6.3.2 Router Analysis with Multiple Flows

As described, all the traffic seen in a NoC is decomposed into traffic flows. Since multiple traffic flows traverse the network, a single router can be involved in handling more than one flow. The flows can arrive at one or more input ports to the router, and leave the router through one or more output ports, depending on the source and destination cores for each of the traffic flows. Consider a router that handles two flows, as shown in Figure 6-6. The aggregate input arrival curve $\overline{\alpha}(t)$ is given by the sum of the arrival curves for the input traffic flows, that is,

$$\overline{\alpha}(t) = \alpha_1(t) + \alpha_2(t) = (b_1 + b_2) + (r_1 + r_2)\, t \tag{5}$$

The burst component of this arrival curve is (b_1+b_2). Using this burst component, the maximum delay D experienced by packets can be obtained from (4), to give:

$$D = T + (b_1 + b_2)/R_s \tag{6}$$

The effective traffic rate for the input arrival curve is (r_1 + r_2). By using (5), the maximum buffer requirement B is, therefore,

$$B = (b_1 + b_2) + (r_1 + r_2)\,\mathrm{T} \tag{7}$$

Based on the buffer requirement, we can derive the expressions for the output arrival curves using the procedure previously outlined. Assuming that the buffer is equally shared by the two input traffic flows, the output arrival curves based on (3) are therefore given by,

$$\alpha_1^*(t) = \frac{[(b_1 + b_2) + (r_1 + r_2)\,\mathrm{T}]}{2} + r_1 t$$

$$\alpha_2^*(t) = \frac{[(b_1 + b_2) + (r_1 + r_2)\,\mathrm{T}]}{2} + r_2 t$$

If there are more than two flows associated with the router, the procedure above is easily extended in order to obtain the output arrival curves. These output arrival curves will then serve as the input arrival curves to the routers that they are incident on. Each router in the NoC can thus, in turn, be analyzed for the maximum delay as well as the required buffer size. The end-to-end delay for each flow can be obtained by summing up the individual delays computed for the routers that are traversed by the flow. The maximum latency for the entire NoC can therefore be easily determined, by determining the flow that experiences the maximum end-to-end latency.

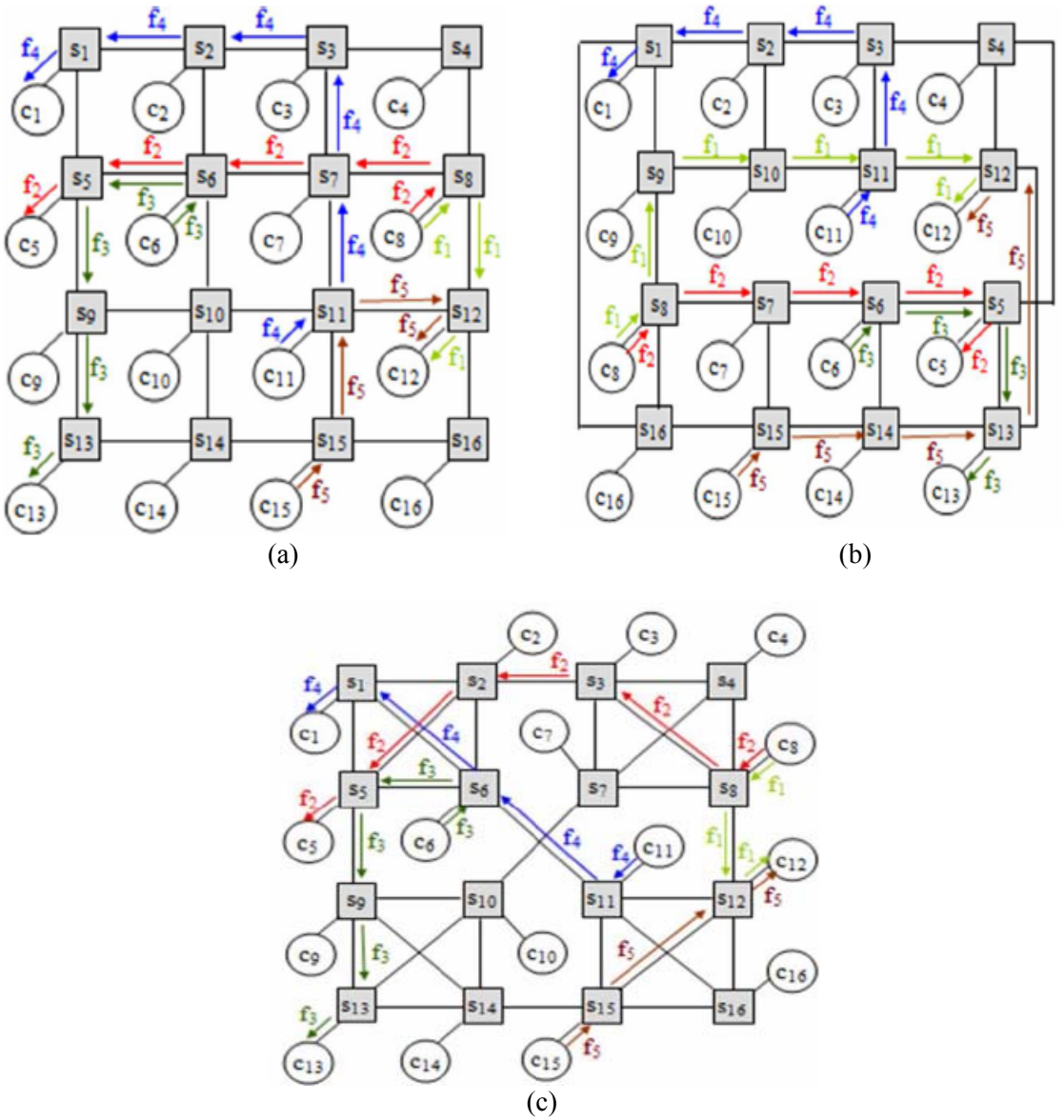


Figure 6-7: On-chip interconnects with data flows: (a) 2D Mesh, (b) Spidergon, (c) WK recursive

6.4 NoC Performance Modeling: An Example

In this section, to show the effectiveness of using Network Calculus for the performance analysis and evaluation, the on-chip interconnects shown with data flows depicted in Figure 6-7 are considered. Based on the methodology described earlier, the designer can describe each component with the arrival and output flows. In order to analyze and evaluate the performance of the whole system, we need to build the model of

the complete system. As depicted in Figure 6-7(a), there are three important elements in a NoC: IP cores (or PEs), routers (or switches), and bidirectional links. The number of routers is assumed here to be 16 and the number of cores or PEs is taken as 16. Each core can be either a source or a sink, that is, cores can either generate or consume messages. Each input port in the router has a buffer for temporary storage of information. When a flit arrives at a router, it must go into the buffer. The buffer is similar to a drop-tail queue with FIFO queue management mechanism. A simulation environment presented in [25] based on the discrete event driven simulator NS-2 [51] is used in this study in order to validate the analytical model. The 2D Mesh network with the flows shown in Figure 6-7(a) will be used as an example for the following analysis in this section.

The sources ($c_6,\ c_8,\ c_{11},\ c_{15}$), and sinks ($c_1,\ c_5,\ c_{12},\ c_{13}$) together with data flows ($f_1,\ f_2,\ f_3,\ f_4,\ f_5$) used in the analysis and evaluation study are also used in the simulation. These data flows are computed using a deterministic routing protocol to direct flits between routers. Unlike dynamic routing protocols that require the implementation of complex logic at the switch level, deterministic routing protocols require simple logic, thereby minimizing the silicon area inside the chip [64]. Our study therefore uses deterministic routing.

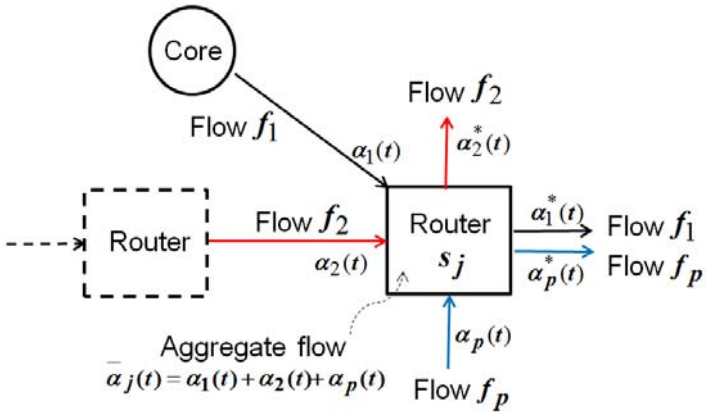


Figure 6-8: Notation used for denoting input, output and aggregate arrival curves

The cores (c_6, c_8, c_{11}, c_{15}) are randomly selected to be sources and each c_i generates a traffic flow which is constrained by an arrival curve $\alpha_i(t) = b + rt$. The cores (c_1, c_5, c_{12}, c_{13}) are considered as sinks, selected so that the following communication locality property holds: 75 percent of the traffic takes place between neighboring cores and 25 percent of the traffic is uniformly distributed among the rest. As depicted in Figure 6-7(a), every PE or core is directly connected to a router/switch. Therefore, we can describe the entire network if we know all the data flows that enter and leave every router. Using a deterministic routing protocol, five data flows are selected as follows: $f_1 = (c_8, s_8, s_{12}, c_{12})$, $f_2 = (c_8, s_8, s_7, s_6, s_5, c_5)$, $f_3 = (c_6, s_6, s_5, s_9, s_{13}, c_{13})$, $f_4 = (c_{11}, s_{11}, s_7, s_3, s_2, s_1, c_1)$, and $f_5 = (c_{15}, s_{15}, s_{11}, s_{12}, c_{12})$. We can see that for the chosen traffic pattern, c_8 is selected two times as a traffic source, corresponding to flows f_1 and f_2, while c_{12} appears as the sink for two traffic flows, f_1 and f_5.

After defining data flows and routers participating in transmitting and/or receiving data, the system is modeled by following the procedure outlined in the previous subsections. Each router is successively analyzed based on its input traffic flows, in order to determine the delay and buffer requirements. The output traffic flow from the

router then feeds other routers, which are then analyzed in the succeeding iterations. We use the following notation in the discussion that follows: at any router s_j being analyzed, input arrival curve for flow f_p is denoted by $\alpha_p(t)$ and the output arrival curve for the same flow is $\alpha_p^*(t)$. To maintain simplicity in notation, we avoid adding a second subscript j to the arrival curve notation. Nevertheless, j is used in our notation for the aggregate input arrival, denoted by $\overline{\alpha_j}(t)$ for router s_j. This notation is summarized in Figure 6-8. In addition, the delay and buffer requirement at switch s_j are denoted by D_j and B_j respectively.

For the 2D Mesh network in Figure 6-7(a), the router s_{15} is first analyzed. This router has only one input flow f_5, with the arrival curve given by $\alpha_5(t) = b + rt$. The aggregate input arrival curve to s_{15} is the same, that is, $\overline{\alpha}_{15}(t) = b + rt$. The maximum delay and the buffer requirement at this router is given by (1) and (2); therefore,

$$D_{15} = T + b/R_s$$

$$B_{15} = b + rT$$

The output arrival curve for the router s_{15} is easily derived using (7), which gives us $\alpha_5^*(t) = b + rT + rt$. The output flow f_5 from s_{15} serves as the input flow to s_{11}. The router s_{11} also has another input flow f_4 originating from core c_{11}. Since all the input flows to s_{11} are now known, it is possible to analyze the router s_{11}. For this router, the input arrival curves from the flows f_4 and f_5 are given by $\alpha_4(t) = b + rt$ and $\alpha_5(t) = b + rT + rt$, respectively. The aggregate input arrival curve for s_{11} is therefore $\overline{\alpha}_{11}(t) = 2b + rT + 2rt$. Following the procedure in Section 6.3.2, the delay and backlog at this router are given by:

$$D_{11} = T + \frac{2b+rT}{R_s} = \left(1 + \frac{r}{R_s}\right)T + 2b/R_s$$

$$B_{11} = (2b + rT) + 2rT \ = 2b + 3rT$$

Router	Delay	Buffer Req.
1	$D_1 = (\frac{17}{4}\frac{r}{R_s}+1)T + \frac{b}{R_s}$	$B_1 = \frac{21}{4}rT + b$
2	$D_2 = (\frac{13}{4}\frac{r}{R_s}+1)T + \frac{b}{R_s}$	$B_2 = \frac{19}{4}rT + b$
3	$D_3 = (\frac{9}{4}\frac{r}{R_s}+1)T + \frac{b}{R_s}$	$B_3 = \frac{13}{4}rT + b$
4	(Unused)	(Unused)
5	$D_5 = (\frac{17}{4}\frac{r}{R_s}+1)T + \frac{2b}{R_s}$	$B_5 = \frac{25}{4}rT + b$
6	$D_6 = (\frac{9}{4}\frac{r}{R_s}+1)T + \frac{2b}{R_s}$	$B_6 = \frac{17}{4}rT + 2b$
7	$D_7 = (\frac{5}{2}\frac{r}{R_s}+1)T + \frac{2b}{R_s}$	$B_7 = \frac{9}{2}rT + 2b$
8	$D_8 = T + \frac{2b}{R_s}$	$B_8 = 2rT + 2b$
9	$D_9 = (\frac{25}{8}\frac{r}{R_s}+1)T + \frac{b}{R_s}$	$B_9 = \frac{33}{8}rT + b$
10	(Unused)	(Unused)
11	$D_{11} = (\frac{r}{R_s}+1)T + \frac{2b}{R_s}$	$B_{11} = 3rT + 2b$
12	$D_{12} = (\frac{5}{2}\frac{r}{R_s}+1)T + \frac{2b}{R_s}$	$B_{12} = \frac{9}{2}rT + 2b$
13	$D_{13} = (\frac{33}{8}\frac{r}{R_s}+1)T + \frac{b}{R_s}$	$B_{13} = \frac{41}{8}rT + b$
14	(Unused)	(Unused)
15	$D_{15} = T + \frac{b}{R_s}$	$B_{15} = rT + b$
16	(Unused)	(Unused)

Figure 6-9: Expression for maximum delay and buffer requirements for the routers participating in the traffic flows depicted in Figure 6-7(a) for 2D Mesh.

As mentioned in Section 6.3.2, the flows f_4 and f_5 are assumed to share the buffer equally, that is, each flow uses a buffer of size $(2b + 3rT)/2$. The arrival curves for the output flows is therefore given by $\alpha_4^*(t) = b + 3rT/2 + rt$ and $\alpha_5^*(t) = b + 3rT/2 + rt$, by following the procedure in Section 6.3.2.

In the third iteration, the parameters for the router s_8 are calculated in the same manner. The calculation will be repeated for the other routers, in the order as follows: s_{12}, s_7, s_3, s_2, s_1, s_6, s_5, s_9 and s_{13}. The results from the analysis are given in Figure

6-9. The figure gives the delay and buffer requirements for all the routers that are involved in processing the traffic that is given in Figure 6-7(a).

Router	Delay	Buffer Req.
1	$D_1 = (\frac{9}{2}\frac{r}{R_1}+1)T+\frac{b}{R_1}$	$B_1 = \frac{11}{2}rT+b$
2	$D_2 = (\frac{7}{2}\frac{r}{R_2}+1)T+\frac{b}{R_2}$	$B_2 = \frac{9}{2}rT+b$
3	$D_3 = (\frac{5}{2}\frac{r}{R_3}+1)T+\frac{b}{R_3}$	$B_3 = \frac{7}{2}rT+b$
4	(Unused)	(Unused)
5	$D_5 = (4\frac{r}{R_5}+1)T+\frac{2b}{R_5}$	$B_5 = 6rT+2b$
6	$D_6 = (2\frac{r}{R_6}+1)T+\frac{2b}{R_6}$	$B_6 = 4rT+2b$
7	$D_7 = (\frac{r}{R_7}+1)T+\frac{b}{R_7}$	$B_7 = 2rT+b$
8	$D_8 = T+\frac{2b}{R_8}$	$B_8 = 2rT+2b$
9	$D_9 = (\frac{r}{R_9}+1)T+\frac{b}{R_9}$	$B_9 = 2rT+b$
10	$D_{10} = (2\frac{r}{R_{10}}+1)T+\frac{b}{R_{10}}$	$B_{10} = 3rT+b$
11	$D_{11} = (3\frac{r}{R_{11}}+1)T+\frac{2b}{R_{11}}$	$B_{11} = 5rT+2b$
12	$D_{12} = (6\frac{r}{R_{12}}+1)T+\frac{2b}{R_{12}}$	$B_{12} = 8rT+2b$
13	$D_{13} = (5\frac{r}{R_{13}}+1)T+\frac{2b}{R_{13}}$	$B_{13} = 7rT+2b$
14	$D_{14} = (\frac{r}{R_{14}}+1)T+\frac{b}{R_{14}}$	$B_{14} = 2rT+b$
15	$D_{15} = T+\frac{b}{R_{15}}$	$B_{15} = rT+b$
16	(Unused)	(Unused)

(a)

Router	Delay	Buffer Req.
1	$D_1 = (\frac{3}{2}\frac{r}{R_1}+1)T+\frac{b}{R_1}$	$B_1 = 2rT+b$
2	$D_2 = (2\frac{r}{R_2}+1)T+\frac{b}{R_2}$	$B_2 = \frac{5}{2}rT+b$
3	$D_3 = (\frac{r}{R_3}+1)T+\frac{b}{R_3}$	$B_3 = \frac{3}{2}rT+b$
4	(Unused)	(Unused)
5	$D_5 = (\frac{9}{2}\frac{r}{R_5}+1)T+\frac{2b}{R_5}$	$B_5 = \frac{11}{2}rT+2b$
6	$D_6 = (\frac{r}{R_6}+1)T+\frac{2b}{R_6}$	$B_6 = 3rT+2b$
7	(Unused)	(Unused)
8	$D_8 = T+\frac{2b}{R_8}$	$B_8 = 2rT+2b$
9	$D_9 = (\frac{13}{4}\frac{r}{R_9}+1)T+\frac{b}{R_9}$	$B_9 = \frac{13}{4}rT+b$
10	(Unused)	(Unused)
11	$D_{11} = T+\frac{b}{R_{11}}$	$B_{11} = 3rT+2b$
12	$D_{12} = (2\frac{r}{R_{12}}+1)T+\frac{2b}{R_{12}}$	$B_{12} = \frac{5}{2}rT+2b$
13	$D_{13} = (\frac{17}{4}\frac{r}{R_{13}}+1)T+\frac{b}{R_{13}}$	$B_{13} = \frac{17}{4}rT+b$
14	(Unused)	(Unused)
15	$D_{15} = T+\frac{b}{R_{15}}$	$B_{15} = rT+b$
16	(Unused)	(Unused)

(b)

Figure 6-10: Analytical results for (a) Spidergon and (b) WK-recursive topology - Expressions for maximum router delay and buffer requirements for traffic flows depicted in Figure 6-7.

Following the same procedure for the other two topologies, namely, Spidergon from Figure 6-7(b) and WK-recursive from Figure 6-7(c), we obtain the delay and buffer requirements as summarized in Figure 6-10.

6.5 Validation of the Analytical Model

In this section, we first compare analytical and simulation results using the same traffic pattern used earlier, which serves as an example for a target application. In particular, we analyze the delay bound experienced by data flows sent by PEs, as well as the maximum buffer size needed to store the data at input ports of routers, using both analytical modeling and simulation. The comparison is carried out for the 2D Mesh topology, with the flows depicted in Figure 6-7(a).

Table 6-2: Parameters used for the analytical model

Parameter	Value
b	64 bits
r	25, 50, 75, and 100 Mbps
R_S	200 Mbps
T	k/R_S= 0.32 µs

After deriving the performance model that captures all information required for the performance analysis (Figure 6-9), the worst case buffer requirements B_i and the data transfer delay D_i of each switch s_i can be calculated by assigning specific values to the various parameters. Since the simulation setup generates a constant bit rate (CBR) traffic, the burst parameter b in this study is taken to be one flit; that is, b=k, where k is the flit size. In our case, a flit consists of 8 bytes, therefore b=k=*64* bits. The parameter r is given by the injection rate at the source; the values considered for the injection rate r are 25, 50, 75, and 100 Mbps. These parameters are used for analyses as well as simulations using the discrete event simulator. For modeling the router service $\beta(t) = R_s \times (t-T)^{\dagger}$, the value of R_s = 200 Mbps is used based on the maximum bandwidth of each outgoing link. The parameter T specifies the fixed delay that is experienced by every flit at the router, in addition to the variable delay based on the rate of service. T it is therefore given by k/R_S, the fixed delay involved in traversing a link with bandwidth R_S. The values of these parameters are summarized in Table 6-2.

The total delay, called end-to-end delay bound D_{f_i}, of each data flow f_i (from PE source to PE sink) can be calculated by summing up the delay of each participating router. Figure 6-11 show the maximum end-to-end delays calculated using Network Calculus (NC) and network simulator (NS). In this figure, the delay increases linearly as injection rate increases. We can see that the delay obtained using Network Calculus analysis is in the same order of magnitude as the end-to-end delay obtained using simulation.

We can also see that simulation results corroborate analytical results obtained using Network Calculus, i.e., the deviation between them is less than 15 percent. In other words, Network Calculus provides a high level abstraction for the application and network behavior, allowing faster performance analysis and evaluation while producing results that are sufficiently accurate.

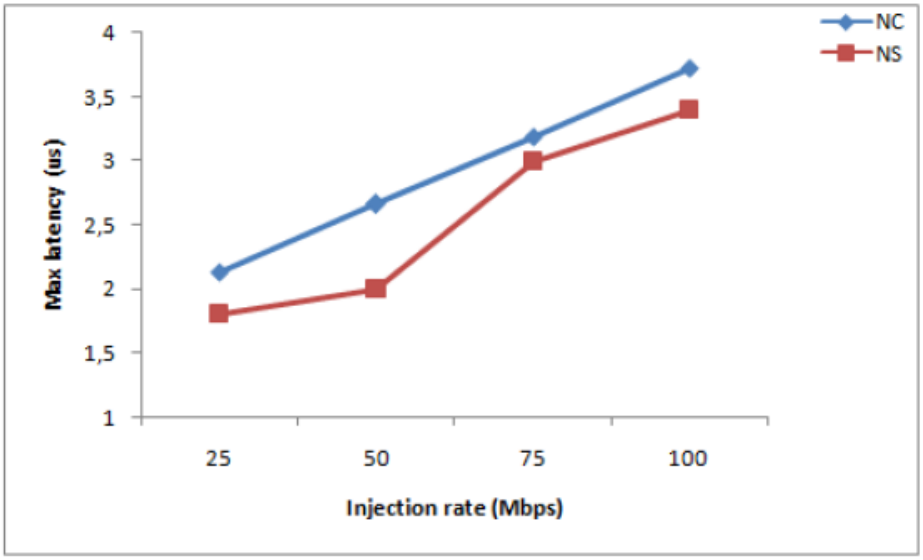


Figure 6-11: The maximum e2e delay, using simulation (NS) and Network Calculus (NC) for 2D Mesh

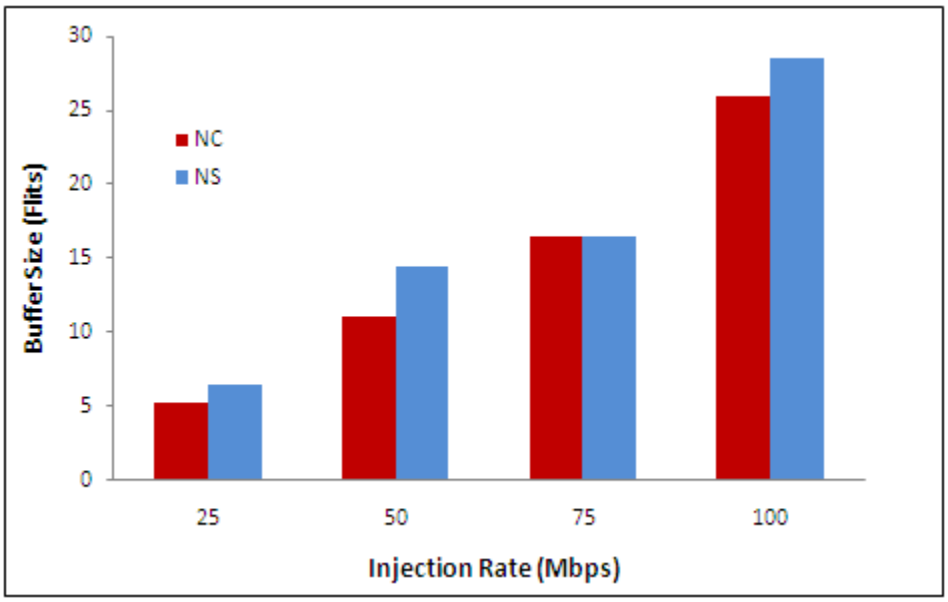


Figure 6-12: The buffer size evaluated using Network Calculus (NC) and measured using simulation (NS)

It was demonstrated in previous works that a dominant part of the NoC area is due to the buffers [69,70]. Thus, finding the minimum buffer size that still satisfies the application requirements is very important. As described earlier in this chapter, Network Calculus can be used to compute the buffer size required for each router, given a traffic pattern (that is, application requirements). Figure 6-12 shows the results obtained for the 2-D Mesh network. The buffer size results from the analytical model are rounded to the next integer value. As mentioned earlier, it is not possible to directly obtain the buffer requirements using simulation; to estimate the required buffer size, simulations have to be repeatedly carried out with increasing the buffer sizes, until there are no dropped flits observed in simulation. Figure 6-12 shows that the buffer size requirements from simulations are comparable to that obtained using the proposed analytical method. Moreover, simulation incurs very high computational cost, due to the need for multiple iterations.

6.6 Analytical Performance Comparison of Network Topologies

As described earlier, it is useful to perform traffic analysis and evaluation in the early stages of the design. The goal is to avoid bottlenecks and contentions in the network, by selecting appropriate on-chip interconnect architecture parameters. The results in the previous subsection have shown that Network Calculus is fairly accurate in estimating the performance metrics such as latency and buffer size requirements. In the rest of this section, we will use Network Calculus to evaluate these metrics for different on-chip interconnects, in order to demonstrate its usefulness as a modeling tool for comparing different interconnect architectures. Herein, the 2D Mesh interconnect with data flows as shown in Figure 6-7(a), is compared against the Spidergon in Figure 6-7(b) and WK-

recursive on-chip interconnect depicted in Figure 6-7(c). This follows along the lines of the detailed empirical analysis presented in chapter 3 for comparing the WK-recursive architecture with the 2D Mesh architecture and other structures, in terms of several performance metrics such as throughput, for a variety of load scenarios.

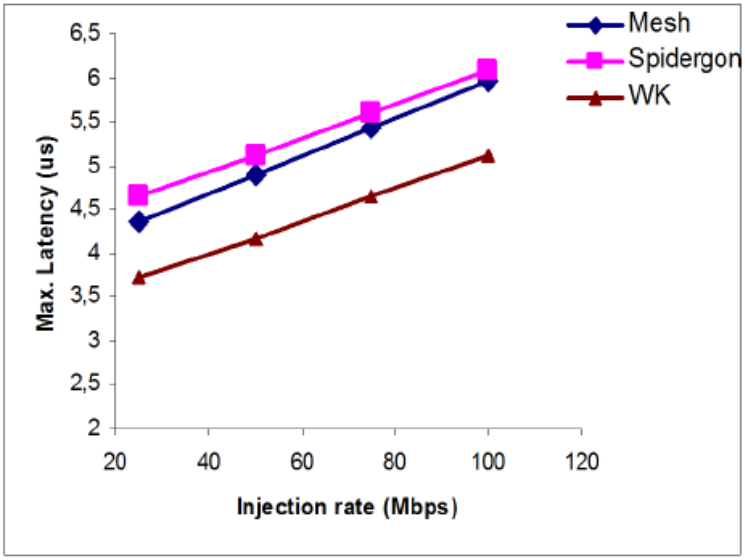


Figure 6-13: The maximum latency for the Mesh, Spidergon and WK interconnects, using Network Calculus, for 16 nodes.

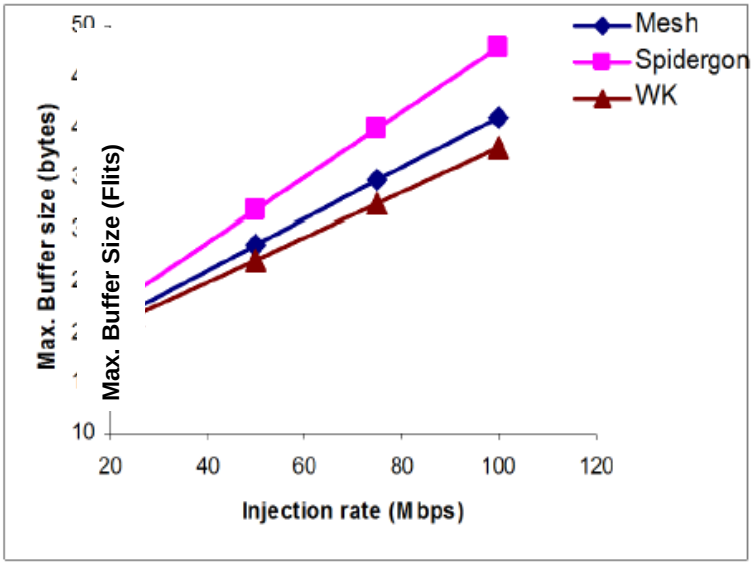


Figure 6-14: The maximum buffer size required for three topologies using Network Calculus, 16 nodes.

Figure 6-13 compares the maximum latency for the three on-chip interconnects under varying traffic load conditions, determined by the injection rate at each node. The results show that the maximum latency increases as the communication load increases (with increasing injection rate). In this figure, 2D Mesh shows a high maximum latency compared to the WK because of its higher diameter.

Figure 6-14 illustrates the maximum buffer size required based on the analytical model. As the injection rate increases, the buffer size increases because the network becomes more congested with heavy traffic. Therefore more space is required to store flits at higher injection rates. WK-recursive topology exhibits a lowest maximum buffer size requirement than the 2D Mesh. It is important to note that the analytical model directly gives the buffer requirements, without the need for repeated simulation to determine the appropriate buffer size, as explained earlier.

In this chapter, we introduced Network Calculus as a useful tool for performance analysis and evaluation of on-chip interconnects. This work demonstrates the modeling of NoC application behavior using Network Calculus, and the computation of some bounds on buffer sizes and end-end delay. The 2D regular Mesh, Spidergon, and WK-recursive on-chip interconnects were selected and some performance metrics were computed using a traffic pattern representative of an example target application. Simulations were carried out to validate the results obtained using Network Calculus. Based on the results presented in this chapter, it can be concluded that Network Calculus is a powerful tool for analysis and design space exploration. It efficiently captures the characteristics of on-chip communication traffic and therefore it is helpful in designing the on-chip interconnect architecture optimized for a target application.

Chapter 7 - Conclusions

Network-on-chip (NoC) has evolved in response to the scalability constraints imposed by bus-based Systems on-Chip (SoC). One of the primary goals of using a NoC is to have a dedicated and reliable communication infrastructure in the system which satisfies the quality of service requirements, while meeting other system constraints, namely power and area.

The network topology plays an important role in determining the NoC performance. Hence, the study, conducted in this thesis, was first focused on studying different topologies to determine which one of them best suits the NoC unique characteristics which differ from regular computer networks or parallel machines. Performance models that measure and evaluate main NoC metrics such as latency and throughput were adopted. Extensive simulation-based experiments were deployed for all architectures using a discrete-event simulator, and performance models were used for comparing these architectures based on these metrics.

Additionally, energy and area models were adopted because of their significance in NoC systems. Simulation experiments were conducted and the energy and area models were used for evaluating the chosen architectures in the performance comparison stage. WK-Recursive network and 2D Mesh topologies were found to be the most suitable structures for NoC implementations.

It was observed that while NoC architectures with regular topologies have clear advantages over bus-based systems, their latency and throughput depend on the topology as well as the application specific scenarios.

To tackle this issue, application-specific optimizations were suggested. This required an adaptable NoC infrastructure to allow in-field customization. A methodology for augmenting a NoC with a programmable infrastructure that allows application-specific adaptation was presented. Based on the developed infrastructure, an algorithm was also developed for static adaptation based on application traffic patterns. To evaluate the proposed methodology of the adaptable NoC, the WK-recursive on-chip interconnect as well as the 2D Mesh network were used as case studies. Simulations of the networks using an event-driven simulator demonstrate the usefulness of the proposed approach.

Since the simulation-based approaches normally incur very high computational costs, especially for large-scale NoCs, and do not generally yield fundamental insights on the behavior of the network when exposed to application-specific traffic patterns, an analytical model based on Network Calculus was developed. It was used as a powerful tool for NoC analysis and design space exploration which efficiently captures the characteristics of the on-chip communication traffic and therefore helps in the design of the on-chip interconnect architecture optimized for a target application.

7.1 Summary of Contributions

The main contributions presented in this dissertation include:

Empirical study for evaluating and exploring NoC architectures

An empirical study aimed at exploring different configurations of network on chip has been conducted. Five different topologies were selected and analyzed using specific routing and switching methods, simulations have been carried out to compare the metrics of performance such as packet loss, throughput, message delay and network load.

The Ring topology performs badly for most of the performance metrics. In terms of throughput, the Mesh and Spidergon outperform the rest, due to their reduced packet loss probability at high injection rates. Packet losses are exhibited when there are too many shared links, which also explains why the Mesh appears to be better compared to Spidergon, based on throughput and packet loss probability metrics. For the average latency (and therefore the average distance), the Spidergon is better than the Mesh for 16 nodes, but this is not the case for 64 and 256 nodes.

Adoption of the WK-recursive topology for NoC architectures

We proposed a NoC interconnect based on the WK recursive network. Simulation study aimed at exploring this on-chip interconnect for NoC was conducted. The study compared this topology with two NoC architectures: 2D regular Mesh and Spidergon. The selected configurations of these architectures were constructed and simulated to compare main performance metrics such as packet loss, throughput, message delay, and network load.

WK-recursive topology generally outperform the Mesh and Spidergon topologies because of its reduced diameter, leading to lower average latency and average number of hops. At injection rates well below the network saturation, no packets are dropped and therefore the reduced number of hops reflects in the lowest energy consumption exhibited by WK-recursive topology. Consequently, the performance/watt is the best for WK. At very high injection rates, the WK-recursive topology is slightly inferior to the other two topologies for some cases, due to the shared communication paths in WK between nodes belonging to different supernodes.

Energy and Area models adaptation

Since the energy and area are the primary concerns on NoC, networks models for energy and area were adapted and simulators were built to calculate these metrics. We quantitatively compared and characterized the average energy consumption of five on-chip interconnect architectures for different communication load and traffic models; the comparison of the best NoC topologies have been presented in this thesis (Chapter 4).

We investigated the effects of load and traffic rates. The results show that the traffic rates and the communication load have a direct effect on the energy consumption. In these results, WK-recursive network and Mesh produced minimum energy consumption when compared against other architectures in all load and traffic scenarios.

Field Adaptable NoC

A single, fixed network topology cannot provide the best performance for a wide range of applications. More precisely, it is observed that applications that have heavy communication between distant nodes naturally exhibit lower performance and higher energy consumption. Application-specific optimizations require an adaptable NoC infrastructure to allow in-field customizations. A methodology for designing this reconfigurable infrastructure has been presented, along with a technique to carry out in-field adaptation based on the application traffic patterns. Field Adaptable NoC (FAN) based on WK recursive network, called Adaptable WK- recursive (AWKR) network, was introduced and compared with FAN for a 2D Mesh. Simulation results were presented based on synthetic traffic generators and real application traces. It was demonstrated that

significant improvements in the data communication time and energy consumption can be achieved, depending on the traffic patterns exhibited by the application.

Analytical modeling techniques based on Network Calculus

We introduced Network Calculus as a useful tool for performance analysis and evaluation of on-chip interconnects. This work demonstrated the modeling of NoC application behavior using Network Calculus, and computed some bounds such as buffer sizes and end-to-end delay.

The 2D regular Mesh and WK on-chip interconnects were selected and some performance metrics were computed using a specific traffic pattern representative of a target application. Simulations were also carried out to show the effectiveness of using Network Calculus. The main objective was to verify the preciseness of Network Calculus compared to a simulation of the system. Our conclusion is that Network Calculus is a powerful analysis and exploration tool which efficiently captures the characteristics of on-chip communication traffic and can therefore guide the designer to select and design the on-chip interconnection architecture that is optimized for a given target application.

7.2 Future Work

In addition to our research findings and contributions, we have also identified a number of areas that merit further investigations. Among those directions that should be explored are:

Multi-resolution tools and techniques for NoC evaluation and design

There are a variety of tradeoffs between detailed simulation, high level simulation, detailed analytical modeling, and high level analytical modeling. In general, greater accuracy is achieved whenever the modeling is carried out in higher detail. To allow efficient design space exploration, it is necessary to have an integrated software tool that uses a hierarchy of models, ranging from a high level analytical modeling to a detailed simulation model. This will allow the designer to rapidly explore a large number of architectural options based on certain traffic patterns, by modeling at a coarse-grain level of details. A few potential candidates can be then chosen for simulation/modeling in greater detail, using the same traffic patterns. In addition to synthetic traffic patterns, it would be helpful if traffic traces from real applications can be integrated into the tool in all levels of the hierarchy. For example, if Network Calculus is used, then the input application traces would have to be automatically converted into their equivalent arrival curves for use in the high-level modeling component of the tool. These features will complement the designer's expertise in making correct choices in selecting the various network parameters and topology.

Improvements to the Field Adaptable NoC

The proposed Field Adaptable NoC (FAN) methodology relies on an exhaustive search of the available options for the nodes constituting the reconfigurable infrastructure. Future work must improve the methodology so that it is feasible for very large networks. One approach, for example, is the use of heuristic algorithms such as Genetic Algorithms in obtaining the infrastructure for the reconfigurable links.

The reconfigurable infrastructure proposed in this thesis is very general and may possibly hinder the advantages that can be achieved by long-range links; future work must address the possibility of developing a reconfigurable infrastructure for a particular application domain. One possible approach would be the use of our proposed analytical model in exploring and identifying reconfigurable links that would improve the performance of the targeted class of applications. The use of domain-specific FAN would be conceptually similar to an ASIP (Application-Specific Instruction-set Processor), wherein the programmable processor has its instruction set tailored to a specific application domain. FAN, however, focuses on the network. Coupling FAN with ASIP would be very powerful and an interesting topic for further research exploration.

This thesis has limited the scope of adaptivity by considering only static adaptation, that is, reconfiguration at the beginning of the application. Implications of dynamic adaptation, based on the observation of application traffic patterns at run-time, need to be evaluated as part of future work.

Improvements to the Network Calculus model

We have seen that Network Calculus is a powerful tool in modeling the performance of the network topology, based on high level abstractions of the traffic and network characteristics. However, equations that model the system are constructed manually in this thesis. To allow efficient modeling, the various iterative computations involving the modeling of each of the routers must be automated. Furthermore, the static routes for different flows must be automatically determined based on the network topology.

In this thesis, we have only considered applications which do not have flows that create cycles. In other words, the system is represented by an acyclic digraph. Future work can possibly leverage the approach proposed in [71] in order to address the problem of using Network Calculus for arbitrary graphs.

Bibliography

[1] D. Wingard, "Micronetwork-Based Integration for SoCs". Proc. The Design Automation Conference (DAC), 2001, pp. 673–677.

[2] S. Kumar et al., "A Network-on-Chip Architecture and Design Methodology". Proceedings of the International Symposium on VLSI (ISVLSI), 2002, pp. 117–124.

[3] W. J. Dally and B. Towles, "Route Packets, not Wires: "On Chip Interconnection Networks". Proc. Design Automation Conference (DAC), 2001, pp. 683-689

[4] U. Y. Ogras, J. Hu, and R. Marculescu, "Key Research Problems in NoC Design: A Holistic Perspective". Proc. CODES+ISSS, 2005.

[5] A. Jantsch., Networks on Chip - Invited Presentation at Darmstadt University of Technology, 2003. Available online at http://www.ele.kth.se/ axel/presentations/2003/Darmstadt.pdf.

[6] P. P. Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh, "Performance Evaluation and Design Tradeoffs for Network-on-Chip Interconnect Architectures". IEEE Transactions, 2005, (54), No. 8.

[7] S. Suboh, M. Bakhouya, J. Gaber, and T. El-Ghazawi, "An Interconnect Architecture for Network on Chip Systems". Kluwer Publishers, Telecommunication Systems Journal, 37: 1-3. pp. 137-144, 2008.

[8] S. Suboh, M. Bakhouya, and T. El-Ghazawi, "Simulation and Evaluation of On-Chip Interconnect Architectures: 2D Mesh, Spidergon, and WK-Recursive Network". ACM/IEEE NoCS 2008 Proceedings, pp. 205-206.

[9] G. D. Vecchia and C. Sanges, "A Recursively Scalable Network VLSI Implementation". Future Generat. Comput. Syst. 4(3) (1988) 235-243.

[10] F-J Fang, Y-R Wang, and H-L Huang, "The m-Pancycle-Connectivity of a WK-Recursive Network". Inf. Sci., 2007, Vol 177, N 24, (Elsevier Science Inc), pp 5611-5619

[11] M. Bakhouya, S. Suboh, J. Gaber, and T. El-Ghazawi, "Analytical Modeling and Evaluation of On-Chip Interconnects Using Network Calculus". Proceedings of the 3rd ACM/IEEE International Symposium on Networks-on-Chip, San Diego, CA, USA, May 10-13, 2009.

[12] D. Flynn, "AMBA: Enabling Reusable on-Chip Designs". IEEE Micro, vol. 17, July 1997, pp. 20–27.

[13] R. Hofmann and B. Drerup, Next generation CoreConnect Processor Local Bus Architecture. Proc. The Annual IEEE International ASIC/SOC Conference, 2002, pp. 221–225.

[14] www.silicore.net.

[15] S. Kumar, A. Jantsch, J-P. Soininen, M. Forsell, M. Millberg, J. Öberg, K. Tiensyrjä, and A. Hemani, A Network on Chip Architecture and Design Methodology', Proc. Int't Symp. VLSI (ISVLSI), 2002, pp. 117-124

[16] L. Bononi and N. Concer, "Simulation and Analysis of Network on Ship Architectures: Ring, Spidergon, and 2D Mesh". Proc. DATE'06, 2006

[17] P. Guerrier and A. Greiner, "A Generic Architecture for On-Chip Packet-Switched Interconnections". Proc. DATE'00, 2000,pp. 250-256

[18] M. Coppola, R. Locatelli, G. Maruccia, L. Pieralisi, and A. Scandurra, "Spidergon: a Novel On-Chip Communication Network". Proc. International Symposium on System-on-Chip, 2004

[19] F. Karim, A. Nguyen, and S. Dey, "An Interconnection Architecture for Networking Systems on Chip". IEEE Microprocessors, 2002, 22(5):36-45

[20] M. Vestias and H. Netot, "Router Design for Application Specific Network on Chip on Reconfigurable Systems". Proc of IEEE 2007.

[21] S. Murali and G. de Micheli, "Bandwidth-Constrained Mapping of Cores onto NoC Architectures". Proc. Of DATE, February 2004.

[22] T. Lei, S. Kumar, "A Two-step Genetic Algorithm for Mapping Task Graphs to a NoC Architecture". DSD 2003.

[23] T. Lei, S. Kumar, "Algorithms and Tools for NoC Based System Design". SBCCI 2003

[24] M. Kreutz, et al., "Design Space Exploration Comparing Homogeneous and Heterogeneous Network-on-Chip Architectures". SBCCI'05, September 2005

[25] Y Y.R. Sun, "Simulation and performance evaluation for networks on chip", Master's thesis, Department of Microelectronics and Information Technology, Royal Institute of Technology, Stockholm, Sweden, December 2001

[26] Gu Lee, Jin Lee, and Sin-Chong Park, "Adaptive Routing Scheme for NoC Communication Architecture". In Proc. of 2005, ICACT 2005, Volume 2, pp.1180-1184

[27] Krishna, Kumar, et al, “Noc with Near-Ideal Express Virtual Channels Using Global-line Communication “. 2008

[28] G. Varatkar and R. Marculescu, “Traffic Analysis for On-Chip Networks Design of Multimedia Applications”. Proceedings of the Design Automation Conference (DAC), 2002, pp. 795–800.

[29] D. Avresky, V. Shubranov, R. Horst, , and P. Mehra, “Performance evaluation of the ServerNet SAN under self-similar traffic”. Proceedings of the 13th International Parallel and Distributed Processing (IPPS) and 10th Symposium on Parallel and Distributed Processing (SPDP), 1999, pp. 143–147.

[30] A. Hegedus, G. M. Maggio, , and L. Kocarev, “A NS-2 Simulator Utilizing Chaotic Maps for Network-on-Chip Traffic Analysis”. Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), vol. 4, 2005, pp. 3375–3378.

[31] A. O. Balkan, G. Qu, and U. Vishkin, “A Mesh-of-Trees Interconnection Network for Single-Chip Parallel Processing”. Application-specific Systems, Architectures and Processors (ASAP'06), 2006

[32] J. Xu, W. Wolf, J. Henkel, and S. Chakradhar, “A Design Methodology for Application-Specific Networks-On-Chip”. ACM Transactions on Embedded Computing Systems, vol. 5, May 2006, pp. 263–280.

[33] V. Raghunathan, M. B.Sb Vastava, and R. K. Gupta, “A Survey of Techniques for Energy-Efficient On-Chip Communication”. In Proc.DAC’ 2003, pp. 900-905

[34] M. Moadeli, A. Shahrabi, W. Vanderbauwhede, and M. Ould-Khaoua, “An Analytical Performance Model for the Spidergon NoC”. 21st AINA Proc., pages 1014–1021, 2007.

[35] J. Hu, “Design Methodologies for Application Specific Networks-On-Chip”. PhD thesis, Carnegie Mellon University, May, 2005, http://www.gigascale.org/pubs/629.html

[36] P. T. Wolkotte, G. J. M. Smit, N. Kavaldjiev, J. E. Becker, J. Becker, “Energy Model of Networks-on-Chip and a Bus”. Proceedings of International Symposium on System-on-Chip, pp. 82 – 85, 2005.

[37] K. Banerjee and A. Mehrotra, “A Power-Optimal Repeater Insertion Methodology for Global Interconnects in Nanometer Designs”. IEEE Transactions on Electron Devices, vol. 49, no. 11, pp. 2001–2007, November 2002.

[38] P. T. Wolkotte, G. J. Smit, and J. E. Becker, “Energy-efficient NoC for Best-effort Communication”. In Proceedings of the 15th International Conference on Field Programmable Logic and Applications, Tampere, Finland, August 24-28 2005.

[39] A. Brinkmann, J. -c. Niemann, I. Hehemann, D. Langen, M. Porrmann, U. R Uckert, “On-Chip Interconnects for Next Generation System-on-Chips”. In Proc. of the 15th Annual IEEE International ASIC/SOC Conference, pp 211-215, 2002.

[40] D. Langen, A. Brinkmann, U. Ruckert, “High level Estimation of the Area and Power Consumption of On-chip Interconnects”. 13th Annual IEEE International ASIC/SOC Conference, 2000.

[41] R. Mehra and J. Rabaey, Behavioral Level Power Estimation and Exploration, Proc. First International Workshop on Low Power Design, Napa Valley, CA, pp. 197-202, April 1994.

[42] K. Lahiri, S. Dey, and A. Raghunathan, “Evaluation of the Traffic-Performance Characteristics of System-on-Chip Communication Architectures”. VLSI Design Proc., page 29, 2001.

[43] H. J. Kim, D. Park, C. Nicopoulos, V. Narayanan, and C. Das., “Design and analysis of an noc architecture from performance, reliability and energy perspective”. ACM SANCS Proc., pages 173–182, 2005.

[44] A. Hansson, M. Wiggers, A. Moonen, K. Goossens, and M. Bekooij, “Applying Dataflow Analysis to Dimension Buffers for Guaranteed Performance in Networks on Chip”. NOCS Proc., pages 211–212, 2008.

[45] A. Hansson, M. Wiggers, A. Moonen, K. Goossens, and M. Bekooij, “Applying Dataflow Analysis to Dimension Buffers for Guaranteed Performance in Networks on Chip”. NOCS Proc., pages 211–212, 2008.

[46] U. Y. Ogras, J. Hu, and R. Marculescu, “Key Research Problems in NoC Design: A Holistic Perspective”. Proc. CODES+ISSS, 2005.

[47] J.-Y. L. Boudec and P. Thiran, “Network Calculus: A Theory of Deterministic Queuing Systems for the Internet”. LNCS 2050, 2001.

[48] R. L. Cruz. A Calculus for Network Delay, Part II: Network Analysis”. IEEE Tran. on Information Theory, 37(1):132– 141, 1991

[49] D. Bertozzi and L. Benini, Xpipes, “A Network-on-chip Architecture for Gigascale System on-chip”. IEEE Circuits and Systems magazine, Second Quarter 2004

[50] S. Suboh, M. Bakhouya, S. Lopez-Buedo, and T. El-Ghazawi, “Simulation-based Approach for Evaluating Network-on-Chip Interconnect Architectures”. SPL 2008 Proceedings, pp. 75-80.

[51] Network Simulator, NS-2, http://www.isi.edu/nsnam/ns,

[52] Sucheta Chodnekar et al, "Towards a Communication Characterization Methodology for Parallel Applications”. Proceedings of the 3rd IEEE Symposium on High-Performance Computer Architecture,Page: 310 1997

[53] MPICL Tool, http://www.netlib.org/mpicl/ accessed June 2010.

[54] A. I. Mahdaly, H. T. Mouftah and N. N. Hanna, “Topological Properties of WK-Recursive Networks”. Proc of Second IEEE Workshop on Future Trends of Distributed Computing Systems, (1990) 374-380.

[55] D-R. Duh and G-H. Chen, “Topological Properties of WK-Recursive Networks”. Journal of Parallel and Distributed Computing, vol. 23, issue 3, pp. 468-474, Dec 1994

[56] L. Verdoscia, “An Adaptive Routing Algorithm for WK-Recursive Topologies”. Computing, vol. 63, issue 2, pp. 171-184, 1999.

[57] Umit Y. Ogras and Radu Marculescu, "Application-Specific Network-on-Chip Architecture Customization via Long-Range Link Insertion". ICCAD 2005

[58] NAS Parallel Benchmarks, http://www.nas.nasa.gov/Resources/Software/npb.html, accessed Dec 2009.

[59] V. Firoiu, J.-Y. L. Boudec, D. Towsley, and Z.-L. Zhang, “Theories and Models for Internet Quality of Service”. Proceedings of the IEEE, ISSN 0018-9219, vol. 99, no. 9, pp. 1565–1591, 2002.

[60] L. Thiele, S. Chakraborty, M. Gries, A. Maxiaguine, and J. Greutert, “Embedded software in network processors models and algorithms”. EMSOFT Proc., vol. LNCS 2211, pp. 416–434, 2001.

[61] E. Wandeler, L. Thiele, M. Verhoef, and P. Lieverse, “System Architecture Evaluation using Modular Performance Analysis - a Case Study”. STTT Journal, vol. 8, no. 6, pp. 649–667, 2006.

[62] J.-P. Georges, E. Rondeau, and T. Divoux, “Evaluation of Switched Ethernet in an Industrial Context by using the Network Calculus”. 4th IEEE Workshop on Factory Communication Systems, pp. 19–26, 2002.

[63] J. B. Schmitt, F. A. Zdarsky, and U. Roedig, “Sensor network calculus with multiple sinks”. IFIP Networking Proc., 2006.

[64] A. Bouillard, B. Gaujal, S. Lagrange, and E. Thierry, “Optimal Routing for End-to-End Guarantees using Network Calculus”. Performance Evaluation, vol. 64, no. 11-12, pp. 883–906, 2008.

[65] U. Y. Ogras and R. Marculescu. “Analytical Router Modeling for Networks-on-Chip Performance Analysis”. DATE Proc., pages 1–6, 2007.

[66] Le Boudec Jean-Yves, “Network Calculus Made Easy”. 1996. Technical report EPFL-DI 96/218

[67] S. J. Krolikoski, F. Schirrmeister, B. Salefski, J. Rowson, and G. Martin, “Methodology and Technology for Virtual Component Driven Hardware/ Software Co-design on the System-Level”. ISCAS 99 Proc., vol. 6, p. 456459, 1999.

[68] M. Grnewald, J.-C. Niemann, M. Porrmann, and U. Rckert, “A Mapping Strategy for Resource-Efficient Network Processing on Multiprocessor SoCs”. Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE04), 2004.

[69] U. V. A.O. Balkan, G. Qu, “A Mesh-of-Trees Interconnection Network for Single-Chip Parallel Processing”. Application-specific Systems, Architectures and Processors (ASAP), 2006.

[70] M. Bakhouya, “Evaluating the energy Consumption and the Silicon Area of On-Chip Interconnect Architectures”. Journal of Syst. Architect, no. doi:10.1016/j.sysarc.2009.07.002, 2009.

[71] D. Starobinski, M. Karpovsky, and L. Zakrevski, “Application of Network Calculus to General Topologies using Turn-Prohibition”. IEEE/ACM Transactions on Networking (TON), vol. 11, no. 3, pp. 411–421, 2003.

[72] Hsin-Chou Chi, Chia-Ming Wu, and Sung-Tze Wu, "A Switch Supporting Circuit and Packet Switching for On-Chip Networks", IEEE DDECS 2006, Page(s): 224 - 225