The Pennsylvania State University
The Graduate School

A COMPREHENSIVE APPROACH TO DESIGN NETWORK-ON-CHIP ARCHITECTURES FOR SOC/MULTICORE SYSTEMS

A Thesis in
Computer Science and Engineering
by
Jongman Kim

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

May 2007

The thesis of Jongman Kim was reviewed and approved* by the following:

Chita R. Das
Professor of Computer Science and Engineering
Thesis Advisor, Chair of Committee

Padma Raghavan
Professor of Computer Science and Engineering

Vijaykrishnan Narayanan
Associate Professor of Computer Science and Engineering
Graduate Officer

W. Kenneth Jenkins
Professor of Electrical Engineering
Department Head

Raj Acharya
Professor of Computer Science and Engineering
Department Head

*Signatures are on file in the Graduate School.

Abstract

With the advent of deep sub-micron technology, System-on-Chip (SOC) architectures are becoming possible for a range of applications. However, as single chip systems become a reality, ingenious solutions are needed for many arising architectural issues.One such issue is the design of an on-chip interconnect to facilitate communications among different IP blocks. The integration of multiple cores on a single die has signaled the beginning of a communication-centric design philosophy rather than a computationally centered one. Further, the introduction of nano-scale technology has emphasized the importance of a communication-conscious design where global wiring delays do not scale down as fast as gate delays in newer technologies. Therefore, on-chip interconnections are expected to be a major hurdle in the design of embedded SoC architectures and high-performance multicore architectures alike. While there is a large body of literature on traditional multiprocessor architectures, the design and analysis of an on-chip communication infrastructure is inherently more complex because of its resource constraints, floor planning, and technology scaling artifacts. Consequently, to resolve the growing concerns of on-chip communication behavior, new architectural and technological solutions are being vigorously investigated. Among the architectural trends, use of on-chip packet-based communication networks, known as Networks-on-Chip (NoC), have been gaining wide acceptance due to their scalability. These NoCs have been deployed in current commercialized products, including a recently announced 80-core TERAFLOP processor [1]. While NoC research has made significant progress, there is still the lack of a generic design methodology which encompasses issues in performance, scalability, power, and reliability in a cohesive fashion.

This research consists of five parts. First, the design and analysis of NoC micro-architectures has been explored, where a comprehensive platform for evaluating the performance and energy consumption behavior has been developed. This framework allows for investigating the scalability issues of these NoC architectures as well as provides a means by which different system configurations, wiring layouts, switching mechanisms, routing algorithms and micro-architectural designs could be evaluated. The rapidly increasing use of SoC architectures has accentuated the need for efficient on-chip communication infrastructures. Thus, as the second part of this work, we proposed one type of solution, in which a low-latency on-chip router architecture supporting adaptive path-sensitive mechanisms, was shown to be able to minimize average packet latency by intelligent path selection and reduced switching activities. Third, we have developed an analytical model. Another fundamental aspect of NoC design is the ability to precisely and efficiently provide analysis of an NoC's performance, fault-tolerance, and energy behavior. We developed a queuing-theory-based analytical model for NoC architectures which performed latency and power analysis at the granularity of individual hardware sub-modules, resulting in an increase of the models accuracy. The model developed here quantified the overall power consumption by capturing the utilization of different component and their corresponding energy consumptions. By integrating performance, power, and reliability models, the analytical model was further able to evaluate multi-objective tradeoffs. Fourth, as a comprehensive design paradigm, we proposed a novel, fine grained modular router architecture utilizing a Row-Column Decoupled (RoCo) design. We explored the SoC/NoC design space for reliable and predicatively high-performing architectures, and developed suitable fault-tolerant techniques to handle permanent hard faults. This architecture comprised of a powerful amalgam of novel techniques, all of which work in unison to produce a very efficient and fault-tolerant interconnection system. The development of concepts such as early ejection, guided flit queuing, the mirroring effect, and hardware recycling provided huge benefits to the router's operation and help create a very efficient and resilient system. In the last chapter, we explored the design of a 3D, crossbar-style, NoC for upcoming 3D VLSI technology as emerging chip multiprocessor systems. This exploration provided insight into the tradeoffs between circuit complexity and performance using real commercial and scientific benchmarks in the simulation testbed.

This research helps to further understand the role of NoC design as an integral part of the future SoC/multicore architectures by investigating and developing new micro-architectural solutions. This thesis is useful for making experimental and theoretical advances in understanding the interplay between performance, energy, and reliability in deep-submicron designs, providing development of comprehensive design analysis models/tools.

Table of Contents

List of Figures

List of Tables

Acknowledgments

During the first couple of years of my graduate study, I became discouraged and fell into despair because I was confused and had little direction. During this time I came to the Department of Computer Science and Engineering where I met a very caring and inspirational mentor, Dr. Das, whose grace changed my life. I am very grateful and indebted to my thesis advisor, Dr. Das, for the unending priceless guidance, patience, and encouragement he has given me during my time here at Penn State. I would like to sincerely thank him for his warm-hearted advice and support.

I am also grateful and indebted to Dr. Vijaykrishnan Narayanan, for inspiration and enlightening discussions on a wide variety of topics. His valuable guidance was very much appreciated especially during the preparation of conference papers. I am grateful to my Ph.D. committee members, Dr. Padma Raghavan and Dr. Kenneth Jenkins, for their insightful comments on my work. I also thank Dr. Lee, an advisee of Dr. Raghavan, for giving his time to discuss scientific applications and measurements. I would like to express my heartfelt thanks to Ms. Vicki Keller, Ms. Trina L Miller, Ms. Connie Jackson, Ms. Karen Alise Corl, Ms. Kelley A Williams, Ms. Helen DeFurio, and Ms. Barbara A Einfalt for their hard work and support as department staff.

I would like to thank my co-workers Chrysostomos Nicopoulos, Dongkook Park, and Reetuparna Das for helping me develop and implement research ideas. I would also like to express my gratitude to Michael DeBole and Seung Woo Son in the reviewing and editing of this thesis. I appreciate all the wonderful memories I have of my graduate study and research pursuit in Pennsylvania State University. I would also like to recognize my colleagues in our research group (The High Performance Computing Laboratory), Deniz Ersoz, Gunwoo Nam, Younghyun Oh, Pushkar Patankar, Seunghwan Lim, and Asit Kumar Mishra. Penn State has also given me many fond memories studying with my Korean colleagues Dr. Jinhyung Park, Dr. Sunho Lim, Dr. Ingyu Lee, Jungsub Kim, Heesook Choi, Jaesung Shin, Sungmin Bae, Boram Lee, Jaehyun Lim, Insoo Kim, Youngjae Kim, Giltae Song, Byungwon On, and Yoonchan Jhi.

I am thankful to God for giving me the opportunity to have this experience and be surrounded by many wonderful people. I am very thankful for being able to wake up everyday and be in the company of someone else. I would like to express my most sincere gratitude to my family and lovely wife, Minseon Gu, for her endless support and prayers.

Chapter 1

Introduction

Design and analysis of System-on-Chip (SoC) architectures, incorporating hundreds of functional units to solve real-world problems, is an emerging and exciting research field. Based on the International Technology Roadmap for Semiconductors (ITRS) [2], by the end of the decade, SoCs using 45/32nm technology will dominate the future semiconductor products [3] because they will have hundreds of cores, consisting of billions of transistors, and be able to incorporate multiple technologies.

A fundamental issue in the design of multicore and System-on-Chip (SoC) architectures with numerous homogeneous and heterogeneous functional blocks is the design of the on-chip communication fabric [4, 5, 6]. It is projected that on-chip interconnects will be the prominent bottleneck in terms of performance, energy consumption, and reliability, as technology scales down further into nanoscale regime [2]. This is primarily because the scaling of wires increases resistance, wiring delays, and energy consumption, while tighter spacing affects signal integrity and, as a result, reliability [7, 8]. Therefore, design of scalable, high performance, reliable, and energy efficient on-chip interconnects is crucial to the success of the multicore/SoC design paradigm, and has become a recent research thrust.

Although interconnection network design has matured over the years with respect to multiprocessor architectures, [9, 10, 11, 12, 13, 14, 15], connecting functional or Intellectual Property (IP) blocks/cores in the same die presents a distinct set of problems that needs ingenious solutions. The first problem is in designing a high bandwidth network with a limited silicon budget. In fact, these are conflicting requirements since the increased number of IP blocks reduces the available real estate for the network, and puts constraints on the available high performance design choices. This issue leads to further complexity in that an on-chip network cannot be designed in isolation; rather co-design of the complete system is essential for best possible results. Second, on-chip interconnects will be vulnerable to several types of errors such as process variation, crosstalk, electromagnetic interference (EMI), and radiation induced soft errors [5, 16]. These effects are the artifacts of deep sub-micron technology, and thus, reliable communication will be another important issue in terms of design and verification. Third, given that on-chip communication consumes a significant portion of the chip's power budget (about 40%) [17, 18], designing energy-efficient multicores/SoCs becomes a critical challenge. Low-power techniques should become one of the fundamental components of the design philosophy. Satisfying these requirements needs a design space exploration involving three parameters– performance, reliability and energy efficiency.

Researchers have examined area-constrained design alternates [19, 20], power-efficient and thermal-aware systems [21, 22, 23] and fault-tolerant mechanisms [24, 25, 26, 27, 28, 29, 30]. However, to my knowledge, none of the past research has taken a *comprehensive approach* encompassing the interplay of the above system-level requirements in the nanoscale regime. Moreover, how the NoCs can benefit from emerging technologies such as 3D chip design is not yet clear. The design of NoCs is thus believed to be in its infancy.

This research aims at developing a comprehensive framework for designing high-performance, reliable, energy-efficient, on-chip, networks considering a multi-dimensional design space and technology constraints. The primary intellectual foci of this research is the investigation into different design tradeoffs in the context of the three evaluation parameters discussed above. Previous work has primarily focused on the design of the interconnect in isolation, whereas in this research, a comprehensive approach is taken to develop a framework which incorporates the entire SoC/multicore system. This is done through theoretical analysis, detailed simulation, and experimental validations at the VLSI design level.

This research consists of five parts. The first part starts with the building a general architectural design and simulation testbed along with associated system structures, performance and power models. This includes the development of a micro-architectural framework to drive the subsequent research with a comprehensive platform to understand the interplay between applications, system architecture and on-chip interconnects. The framework has been used for exploring the NoC design space for high and predictable performance, and has served as a platform for validating various optimizations that have been proposed. We used the workload characteristics of several potential applications. The applications include synthetic scientific [9], self-similar web traffic [31] , multimedia [32, 33], and a cache simulation environment running real commercial and scientific benchmarks. In addition, such designs have been implemented in structural Register-Transfer Level (RTL) Verilog and then synthesized in Synopsys Design Compiler using TSMC cell library.

In the second part of this thesis, we proposed a low-latency router architecture, so-called, a Path-Sensitive Design, suitable for on-chip networks which utilizes an adaptive routing algorithm. The architecture uses a novel path selection scheme resulting in a 2-stage switching operation with a decomposed crossbar switch. We showed that under non-uniform traffic this Path- Sensitive Architecture reduces the overall network latency by a significant factor. Due to the low latency of the Path-Sensitive Architecture, congestion information about each router's neighboring routers is swiftly updated, resulting in almost real-time updates.

Next, we developed an analytical model in performance, energy and reliability for evaluating the entire NoC system. While researchers have examined the area-constraint design alternatives, energy models or fault-tolerance issues individually, a systematic design methodology encompassing the interplay of performance, fault-tolerance and energy constraints is yet to evolve. Such a design methodology is impeded in part due to the lack of efficient techniques to quickly explore alternatives from a large design space. Towards this end, we proposed a queuing-theory-based tool quantifying the performance and energy behavior of on-chip networks. Although wormhole switched, traditional off-chip networks have been analyzed extensively in the literature, analytical models for NoCs considering detailed architectural artifacts are almost nonexistent. In addition, this model is different from that he computes the average delay due to path contention, virtual channel and crossbar switch arbitration using a queuing theory approach, which can capture the blocking phenomena of wormhole switching quite accurately.

As the forth part of this research, we integrated the performance, energy and reliability models to develop a design paradigm that provided a comprehensive framework for exploring high-performance, fault-tolerant, and energy-efficient SoC/muliticore NoC architectures. We explored various types of router faults in the SoC/NoC design space, and developed suitable fault-tolerant mechanisms to combat several types of hard faults in permanent errors. With these, towards the goal of designing low-latency, energy-efficient and reliable on-chip communication networks, we proposed a novel fine-grained modular router architecture. The proposed architecture employs decoupled parallel arbiters and uses smaller crossbars for row and column connections to reduce output port contention probabilities as compared to existing designs. Furthermore, the router employs a new switch allocation technique known as "Mirroring Effect" to reduce arbitration depth and increase concurrency. In addition, the modular design permits graceful degradation of the network in the event of permanent faults and also helps to reduce the dynamic power consumption. Furthermore, we proposed a combined measure, PEF (Performance, Energy consumption, and Fault-tolerance = (Latency*Energy)/(Completion Probability)). Evaluation using the combined metric indicates that the proposed architecture provides 35-50 % overall improvement compared to existing router architectures.

3D technology is envisioned to provide a performance-rich, area- and energy-efficient, and temperature-aware design space for multicore/SoC architectures. In this context, the on-chip interconnect in a 3D setting will play a crucial role in optimizing the performance, area, energy and thermal behaviors. In the last part of this research, we have explored several design options for 3D NoCs, specifically focusing on the inter-strata communication. Three possible designs that include a simple bus for the vertical connection, a symmetric 3D hop-by-hop topology, and a true 3D crossbar architecture are investigated. The proposed 3D architecture, called the 3D DimDe router, supports two vertical interconnects to achieve a balance between the path diversity and high bandwidth offered by a full 3D crossbar and the simplicity of a bus. We have investigated the detailed microarchitectural implications of the design, which include the feasibility of the inter-strata vertical wire layout, arbitration mechanism, and virtual channel support for providing deadlock-free routing.

This research could be of great value to computer architects in the near future. The upcoming designs which use SoC/multicore architectures will need high performance and energy-efficient on-chip networks, which must provide seamless and trouble-free operation throughout the chip's life. By applying a combination of several techniques explored in this research, it will possible to design an NoC architecture satisfying the critical objective functions.

The thesis is organized as follows. Chapter 2 discusses the challenges of SoC architectures based on a general purpose multicore system with a communication-centric approach. Prior work and design issues related to NoC architecture is represented in Chapter 3. This is followed by the path-sensitive design in Chapters 4. Chapter 5 is dedicated to building up an analytical model for performance, power and reliability with a queuing-theory-based technique. Then, a fine grained modular design and a 3-D architecture are described in Chapters 6 and 7, respectively. Finally, the thesis is summarized in chapter 8 along with future research directions.

Chapter 2

Design Challenges of System-on-Chip (SoC) Architecture

SoCs/multicores have begun to be used in an increasingly diverse set of applications such as supercomputers, portable multimedia products, smart homes, health and medical instruments, intelligent buildings, mobile computing environments, and automobiles. This chapter is devoted to introducing a generic architecture model for System-on-Chip (SoC) designs.

The semiconductor industry predicts that, while manufacturing complex integrated circuits will be feasible at least down to 45 nm technology scale, the cost of developing and implementing a comprehensive design will be spectacularly high [2]. Trying to overcoming the problems of design complexity and the ever increasing time-to-market (TTM) has led to the introduction of the SoC design concept. One of the most important issues in SoC applications is design and verification efficiency. SoC designs should be able to provide integrated solutions for increasing circuit complexity at the optimum production cost, while reducing the TTM. Moving from the ASIC design flow to the SoC design methodology should be more structured at a higher level of abstraction, and should emphasize design and system level reuse, and construction of sub-systems through the integration of functional blocks.

Further, for developing an economically feasible SoC design flow, platform based design (PBD) [34] methodologies have been proposed to integrate intellectual property (IP) blocks, the reuse of IP cores, a hardware/software co-design methodology, a compatible organization, and a standard interface. Platform-based design is based on flexible design templates that provide an efficient methodology for SoC/multicore processor design. These templates can be extended and adapted to a diverse set of applications by reconfiguring, revising or programming some components. Other methodologies are already being promoted for industry-wide use by organizations such as the SPIRIT (Structure for Packaging, Integrating and Re-using IP within Tool-flows) [35] Consortium. The main characteristics of these design strategies are their modularity, flexibility, and scalability for the applications in which they are used. Thus, they can bring forth plug-and-play style design environments using pre-designed blocks. This approach is appealing in both development and manufacturing domains in terms of performance, functionality and product-volume constraints.

2.1 SoC Development and Tiled Architecture

In this section, I present a brief description on a generic architecture with a tile based design template. The architecture contains arrays of processing tiles [4, 17, 36], where the processes of an application can be mapped onto the tiles for efficient execution. Each tile, organized in a rectangular geometry, can be composed of a processing element (PE), its local cache, a controller and an interfacing logic. Each PE can be an independent processor core. DSP, FPGA, ADC, ASIC, GPU or a memory block can be used either as a homogeneous or a heterogeneous module as shown in Figure 2.1.

This design template can support efficient design flow and reconfiguration by incorporating a variety of functional blocks, and structured/hierarchical partitionings of the blocks using on-chip communication infrastructures.

Consequently, embedded SoC and multicore architecture design methodologies are observed from three aspects: System Architecture, Modularization and Design Reuse, and Communication Infrastructure.

System Architecture: Depending on the applications, the whole system is configured and partitioned into a number of features. For each tile, several character-

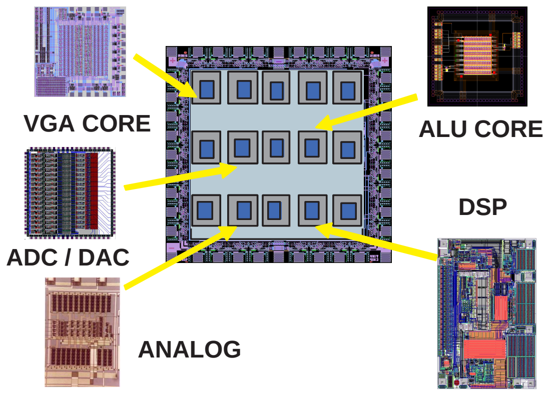


Figure 2.1. A Tiled Architecture

istics such as the type of the PE and IP functional blocks, memory configurations, and I/O controls, should be determined beforehand. Further, understanding the capabilities of the selected IPs and the environment in which they will be used becomes important when deciding on a decoupled interconnect architecture, because they constrain the design space and the level at which abstractions can be made. The method of composition or assembly of the basic modules or IP blocks to construct this tile-based architecture impacts not only the design complexity and verification, but flexibility and scalability. The method chosen must be capable of extending an SoC design to the next level of abstraction by fast configuration and verification of re-usable sub-components. Task-processor mapping policies/algorithms and floor planning have a co-relation with the geometry and topological issues as well.

In addition, in a general purpose multicore system, if the system can be reconfigured in several ways, the system has more flexibility and improved performance by the distribution of computation among available PEs. For example, the TRIPS [36] system has a mechanism that the processing core and on-chip memory are allowed to be reconfigured and combined in diverse modes for instruction, data, and thread-level parallelism, all of which helps in enhancing the overall performance. The system can be adapted from simple control blocks to a fully distributed parallel machine.

On the other hand, recent embedded applications necessitate high performance processors integrating fast and low-power caches. As a consequence, new design techniques for the on-chip memory hierarchy has been pursued. Utilizing the memory subsystem and data relocation, the NUCA (Non-Uniform Cache Architecture) caches [37, 38, 39] have been proposed to overcome the bottlenecks resulting from growing wire delays in multicore architectures. NUCA caches are large L2 caches, organized in sub-banks. Each sub-bank can be accessed independently, with an access time depending on its physical distance from the cache controller. Flexible placement with fine-grained reusable cache banks is led by design modularization and an on-chip communication infrastructure.

Modularization and Design Reuse: This is based on a hierarchical approach, which proceeds by partitioning a system into modules and requires compatibility and consistency. Proper system partitioning allows independence between the design of different modules. The decomposition is generally guided by structuring rules aimed at hiding local design decisions in such a way that only the interface of each module is visible. This kind of methodology is also called “a modular design”, and it consists of sound design rules in terms of timing constraints, hierarchical design, and floor-planning. The overall modular approach optimizes the insertion of reusable components and stand-alone design within the circuit.

Design for reuse concerns all additional activities that have to be performed to generate an easy-to-use and flexible module. By adapting an IP and system level reuse methodology, the whole system can be seen as a set of blocks designed with different methodologies and for different purposes. A diverse set of components and modules can be ported and reused from previous designs, depending on the application specification.

Communication Infrastructure: If chip designers and system architects can effectively build an on-chip communication infrastructure for connecting many discrete building blocks it would enable plug-and-play IP reuse and would scale with new generations of process technology. The modular design approach and component reuse methodology show the need to address the complex requirements in SoCs. Therefore, the design of the SoC communication infrastructure, also called the on-chip network, facilitates the development and employment of reusable system components. Since wire delays for crossing the die has started to reach tens of clock cycles, a network module, using multiple packets and high-speed links, is becoming a viable solution. This structured communication infrastructure yields reduced die size and costs with higher wire efficiency. The components of an on-chip network, such as the switching fabric, link circuitry, buffer and control logic, and the interface, which are designed to be compatible with heterogeneous and homogeneous Processing Elements (PEs), should be interoperable and reusable.

2.2 Network-Driven Computing: The Path from SoC to NoC

As a result of the increasing degree of integration on a silicon die, the SoC design paradigm is seen as a way to design communication architectures for varying an exceedingly high number of pre-designed, computational, and storage blocks. This communication-centric SoC is a new design paradigm, which is suitable for many applications; more thread level parallelism (TLP) with less focus on instruction level parallelism (ILP) and light weight parallel processing agents.

2.2.1 Interconnects Dominate

With the growing complexity of System-on-Chip (SoC) architectures, the on-chip interconnects are becoming a critical bottleneck in meeting the performance and power consumption budgets of the chip design. The ICCAD 2004 Keynote Speaker[40] emphasized the need for an interconnect centric design by illustrating that in a 65nm chip, up to 77% of the delay will be due to interconnects.

As the transmission line size shrinks and the number of connected IP blocks increases the corresponding capacitances and resistances of wires increases. This negatively impacts the propagation delay, the achievable clock cycle, and may result in ad hoc interconnections that could suffer from poor utilization because of expensive wiring tracks of higher metal layers. This can also lead to superfluous silicon budget consumption with redundant link circuitry, more power dissipation and less controllable performance. Figure 2.2 shows the occupancy of the interconnect at each layer in a chip's cross section, where an upper layer takes double the space of a lower layer. In addition to serious performance bottlenecks, on-chip interconnect architectures can take up to 50% of the silicon budget as well [41]. In

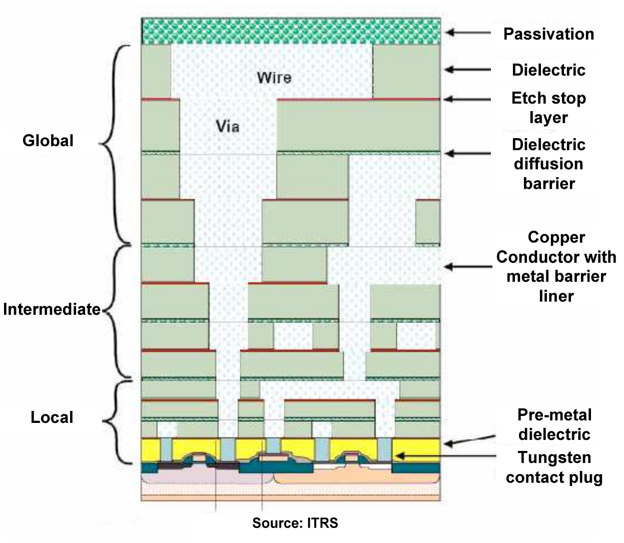


Figure 2.2. Wiring Hierarchy in a Chip Cross Section [2]

the forthcoming technologies, ad hoc wire routing in global interconnects will be replaced by packet-based switching with self-synchronous IPs that communicate with one another through a network-centric architecture. These interconnection networks will address the following issues.

Global Wire Scaling Problem: One of the major problems associated with ultra-deep submicron processes arises from non-scalable global wire delays. Global wires carry signals across a chip, but these wires typically do not scale in length as technology scales. Though gate delays scale down with technology, global wiring delays typically increase exponentially, or at best linearly, by inserting repeaters. Even after repeater insertion, the delay may exceed the limit of one clock cycle. It is estimated that non-repeated wires with practical constraints result in delays of about 20 clock cycles across the chip in the 45 nm technology node, as shown in Figure 2.3.

Predictability: The capability to make early decisions based on the expected performance of the final implementation is very important in a communication design to avoid time-consuming design iterations. The shift towards deep-sub-micron integration makes this aspect even more critical, where the propagation delay varies with physical parameters and electrical properties. Furthermore, the increasing ratio of the delay of long wires with respect to gate delay and the dependence of the propagation delay make it increasingly hard to have the desired

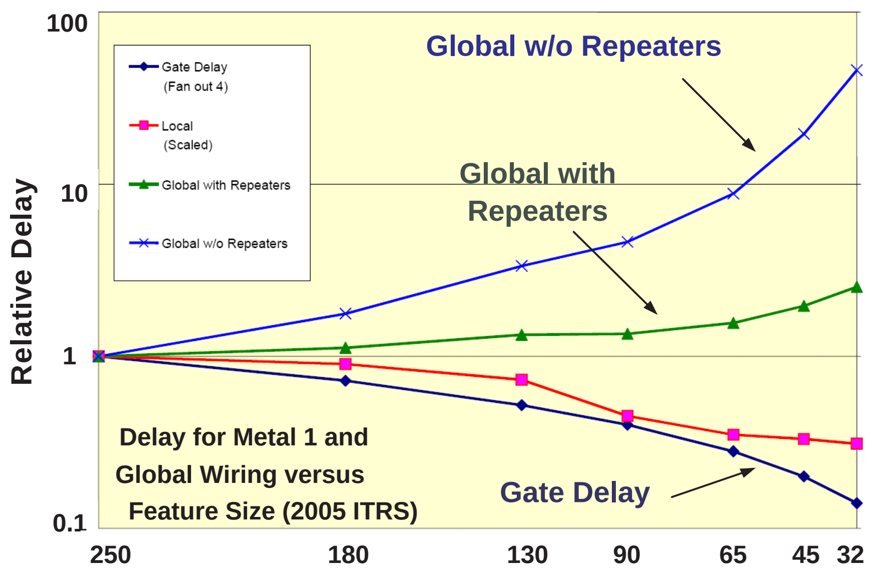


Figure 2.3. Delay for Metal 1 and Global Wiring vs Feature Size [2]

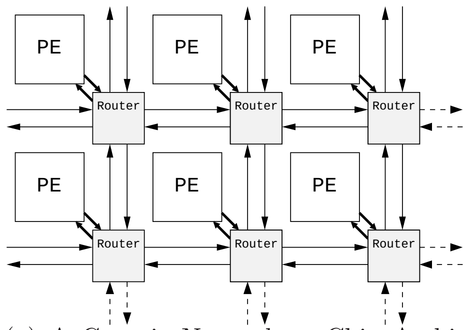
system functionality.

Thus, a chip-wide communication-centric approach should facilitate the design of scalable and energy-efficient SoC architectures, which is not limited by design size or number of cores. The trend toward fine grain parallelism also increases the demand on the underlying SoC communication infrastructure.

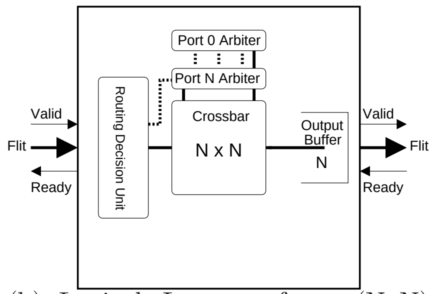
A communication-centric approach has been introduced to overcome key problems in the integration of multiple homogeneous/heterogeneous IP blocks in complex SoCs. This strategy also allows for the processing elements and its network interface controller (NIC) to be able to be decoupled from the communication fabric. Consequently, Network-on-chip (NoC) architectures are expected to be crucial for implementing complex and function-rich chips in platform based design. As chip complexity continues to increase, a systematic approach is required to effectively transport and manage on-chip traffic, optimize wire utilization, and allow designs to scale down in size, decreasing complexity and component reuse without compromising performance, power consumption, and reliability.

2.3 Architectural Framework

Most recent research has focused on switch-based direct and indirect networks. A generic framework for the NoC architectures explored in this work is shown in Figure 2.4 (a) and is characterized by the number of processing elements (PEs) in the array, the bandwidth of the links between the routing nodes, the topology of the network, and the mechanism for packet forwarding. Each PE is attached to a router/switch that is used for communicating with the other PEs. The routers are connected in a regular pattern (using metal interconnects in the on-chip network), defining the network topology. Figure 2.4 (a) depicts a 2D mesh, which is most commonly used in recent studies. Communication between the PEs occurs by exchanging packets of information that traverse through the routing nodes and the links. Each packet, in turn, is composed of several flits. The packet forwarding mechanism determines how data moves in the network. In general, wormhole switching [4] and store-and-forward/packet switching techniques are considered for packet forwarding in the network. In the latter approach all the flits of a



(a) A Generic Network-on-Chip Architecture



(b) Logical Layout of an (NxN) Router Architecture. The Valid and Ready signals are part of the inter-node control signals, and indicate the availability of a packet, and the availability of buffer space for it.

Figure 2.4. System-on-Chip Architecture with a 2-D Interconnect

packet reach a routing node before they are forwarded to the next routing node. In the case of wormhole routing, the flits can move as long as the first flit of the packet is not blocked. It is also possible to use either a deterministic or adaptive routing algorithm for data transfer. Communication handshaking between nodes is performed using additional control wires when the packet transmission system used prevents any packet loss in the network since congestion/backpressure information is also carried. The key component in the NoC is the router/switch as shown in Figure 2.4 (b). Our initial logic for the router design is implemented in 90nm CMOS techonology.

On-chip design provides a number of ways to exploit new routing techniques and architectural innovations. These issues need to be thoroughly examined in order to provide novel solutions. Although a few researchers have investigated the NoC design space, a systematic, system wide design paradigm is yet to evolve. In this thesis, I explore the NoC design space and develop efficient and scalable NoC architectures. Our initial logic for the router design is implemented in 90nm CMOS techonology.

Chapter 3

Related Work in Designing NoC Architectures

As stated earlier, NoC design for SoC/multicore architectures comes with a different flavor because of the area, energy and reliability constraints in deep sub-micron domain. In this chapter, we describe prior work related to NoC design issues. Work related to this research primarily falls under three categories: scalable and distributed NoC architectures, power and energy optimization, and reliable communication. Prior research in each of these areas is summarized in the following subsections.

3.1 Scalable and Distributed Architectures

Three types of on-chip networks have been proposed by several researchers. These are bus-based designs, centralized switch-based designs and distributed switch-based direct/indirect networks.

The simplest communication medium is a shared bus, and it has been commerically adopted [42, 43, 44, 45, 46, 47]. A few researchers have proposed several contention alleviation techniques for increased concurrency in a shared bus: the ring-based bus (circular EIB) [42], The Lottery bus [48], the Octagon architecture [49], the TDMA bus, [50, 51] and the CDMA bus [52]. These designs work efficiently for connecting a smaller number of IP blocks (typically less than 10 cores on-die) [5, 53]. However, a shared bus cannot meet the demand of scalable systems because of the high wire load and slow signal propagation in deep sub-micron design[7].

In a centralized switch design, all of the on-chip computing elements are connected through a central crossbar or its variant [54, 55, 56, 57]. As a result, communication latency is limited to a single hop. Such designs have been experimented with in [58, 59, 23, 60]. However, a central router design is also not a scalable solution, and furthermore, the design of energy-efficient crossbars is a nontrivial task.

Therefore, interest in distributed, router-based, on-chip networks has rapidly gained momentum for designing scalable SoC/multicore architectures , and analysis and optimization of these multicore architectures have garnered great attention [5, 4, 61, 62, 63, 64, 65, 66, 67, 21, 68, 69, 70, 71]. Such networks are ideal for component reuse, design modularity, plug-and-play, scalability, and they avoid the uncertainty of global wire delay perspectives. The large body of literature on several aspects of NoC architectures shows the current interest in this area. Using a direct network such as a mesh, the IP blocks/PEs are placed as regular tiles and the tiles are interconnected through the switches. For example, the MIT Raw and the TRIPS machines use a 2-D mesh architecture [17, 36]. Both packet switching [72, 69, 32] and wormhole switching techniques [73, 21, 74, 75, 76, 77] have been used in the design of these networks. Starting from simplistic routers with deterministic routing algorithms [4, 17], research has evolved to fine-grain pipelined routers with Virtual Channels (VCs), efficient arbitration schemes, and performance enhancement techniques to achieve high-bandwidth [78, 76, 79, 80, 75]. In addition, some researchers have designed SoC networks for specific applications such as multimedia [32, 33], DSP [67, 81], and scientific applications [82, 83, 17, 36, 68]. For example, it was shown in [83] that it is possible to design better optimized on-chip networks than meshes or tori by utilizing the temporal and spatial communication properties of well behaved applications. A few researchers have analyzed other topologies for the on-chip interconnect [84, 85, 67, 86].

Since area, energy efficiency and fault-tolerance are essential in the NoC domain, a handful of recent designs have analyzed these issues. Area-constrained design alternates are examined in [41, 87, 88, 20], power-efficient models in [89, 90, 21], and fault-tolerant mechanisms in [30].

However, none of the prior studies have considered a comprehensive approach to the design and analysis of on-chip interconnects encompassing the three design metrics - performance, power, and reliability. Furthermore, technology scaling allows for the opportunity to exploit new flow control mechanisms, routing techniques and architectural innovations such as 3D networks. These issues need careful investigation in order to provide novel solutions. My research addresses these issues.

3.2 Power and Energy Optimization

With the current trend of shrinking feature size, future multiprocessor/SoC architectures are likely to require an increasing amount of power. In many cases this increased power requirement can limit the feasibility of design alternatives, ultimately reducing the overall design space. Since the on-chip network consumes a significant fraction of the power budget, energy-driven interconnect design plays an important role in the future of multiprocessor/SoC architectures. Energy and power characterization for on-chip communications can be summarized under three broad categories: physical-layer power model, network-architecture-level power and energy characterization, and system-level energy conservation.

At the physical level, researchers have focused on device power optimization using low voltage swing techniques and link pipelining for on-chip signals that are propagated across the interconnect [91, 92]. This low voltage swing adversely effects the interconnect's immunity to noise. Thus, adaptive and differential signaling were introduced as alternatives [93, 94, 95, 46] along with appropriate encoding schemes [25, 96].

At the network level, researchers have developed energy models for analyzing the power consumption of routers/switches [90, 62, 97, 21, 59, 45, 23], which in turn will enable energy efficient implementations of communication architectures in NoCs. Researchers [41, 98, 4] have explored the tradeoffs between performance and power consumption in various NoC configurations. Peh et al. [21, 23, 90] have proposed an energy model for NoCs, and suggested router energy optimizations based on resource utilization using segmented switch, write-through buffer and cutthrough crossbar design techniques. Their results show up to 44% power saving without any performance penalty. In addition, it is shown that topologies like express cube [99] are energy efficient because of lower network diameter. Their energy model, although quite accurate, only captures dynamic power consumption based on flit transmission. It does not take into account leakage energy, which will be a significant concern with shrinking feature size. Furthermore, the accuracy of the model needs to be verified for a wide range of design alternatives with realistic traffic patterns.

At the system level, primarily two techniques, called Dynamic Voltage Scaling (DVS) [22, 100, 101, 18] and Dynamic Link Shutdown (DLS)[102, 103, 104], have been investigated. With the DVS scheme, the link frequency and voltage are regulated based on link utilization. The DLS scheme, on the other hand, shuts down a link if its utilization is below a certain threshold. Both these techniques are effective in minimizing the link power dissipation. However, the impact of such techniques on the reliability of the network is not clear.

With higher power densities, the exponentially rising thermal dissipation is becoming a serious issue in microprocessor design [105, 106, 107]. Power-efficient and temperature-aware NoC design has been proposed in [108], where a thermal modeling and simulation framework was incorporated into a distributed throttling scheme for thermal efficiency. Irwin et al. proposed thermal-aware IP virtualization and placement techniques for NoC architectures in [109]. However, a comprehensive approach to designing thermal-aware multicore systems, is still missing.

In contrast, our objective in this research is to design and implement a comprehensive energy estimation tool that considers all different components of an NoC and is amenable to evaluating the influence of different power optimization techniques such as switching activities, buffer management, and flow control including leakage power. Furthermore, we propose developed a high level analytical power model for guiding future designs and analyze the impact of such designs on reliability.

3.3 Reliable On-Chip Communication

Information transfer is increasingly becoming unreliable in on-chip networks due to data corruption, buffer overflow, and control logic and synchronization errors [30]. These are artifacts of crosstalk, substrate noise, clock skew, electromagnetic interference, material aging, process variation, and (cosmic) soft errors. In fact, recent studies predict that the soft error rate (SER) chip of logic circuits will become comparable to the SER of unprotected memory elements by 2011 [110]. For example, data corruption or buffer overflow could occur if the required signals are not received in a timely fashion or are corrupted during transmission. Continued power supply reduction, high clock frequencies, and reduced feature sizes further exacerbate the error resiliency of NoCs.

Prior efforts for reliable communication can be summarized in two categories: transient error control schemes [111, 112, 16, 113, 114, 25, 115], and stochastic communication models [116, 30, 29]. A transmission error control mechanism typically uses a forward error correcting (FEC) scheme [25, 16], or retransmission strategies [117, 118, 16, 24]. The tradeoffs between the two policies (FEC and ARQ) should be carefully investigated for designing reliable and low power NoCs. Researchers [119, 97, 111] have explored hybrid techniques such as hop-by-hop header error correction (HEC) and end-to-end data retransmission, but a robust design methodology for integrating these techniques in a design needs further exploration.

Stochastic models rely on alternate paths in the event of a switch or link fault. A load-balanced routing [120] technique and a hybrid deflection routing scheme [121] have been introduced in this domain.

Reliable SoCs should protect on-chip communication against both transient and permanent failures. [116] has proposed a fault model for CMP switch architectures and analyzed the reliability versus area tradeoffs in providing online repair and recovery capabilities. Das et al. [119, 97] presented a comprehensive approach against transient and permanent faults in both the router and links.

Recently, process variation has emerged as an obstacle in meeting time-to-market and product-volume requirements in platform-based design (PBD) methodologies. An adaptive FPGA architecture [122] has been developed to minimize timing variations and to provide a better yield. Along these lines, an NoC process variation model should be developed to aid designers during the exploration phase.

Dependability studies of various multiprocessor interconnects have been reported in the literature [123, 124, 125, 126, 127, 128, 129]. However, these models are not directly applicable to NoCs. Reliability analysis of NoCs should capture several NoC-specific design intricacies such as very wide channels and elaborate router designs.

To my knowledge, none of the prior research has considered all of the above issues in exploring the NoC design space. At best, researchers have examined the area-constraint design alternates [19, 20], energy models [21, 22, 23] or fault-tolerance issues [24, 25, 26, 27, 28, 29, 30] in isolation. A systematic design methodology encompassing the interplay of various constraints has yet to evolve. The motivation of this research is to explore the NoC design space considering the technology constraints. The intellectual foci of the research are in understanding the intricacy of the entire design space and investigating the design tradeoffs in optimizing performance, energy and reliability parameters. This has been done through theoretical analysis, detailed simulation, and experimental validations at the VLSI design level.

Chapter 4

A Path-Sensitive NoC Router Architecture

In this chapter, I propose a low latency, on-chip, router architecture supporting adaptivity using a path-sensitive mechanism [80]. This mechanism has been shown to minimize average packet latency by intelligent path selection and leads to a reduction in switching activity.

4.1 Introduction

Early NoC designs used dimension order routing due to its simplicity and deadlock avoidance. However, adaptive routing algorithms provide adaptivity to various traffic patterns for handling congestion as it evolves in a network. The challenge in using adaptive routing in NoC designs is to limit the overhead in implementing such a complicated design.

In this chapter we present a low-latency, two-stage, router architecture suitable for NoC designs. The decomposed router architecture, called the *path-sensitive router*, utilizes look-ahead routing when selecting the next route. The router is called path-sensitive because, based on the destination address, it selects one of the four possible quadrants (NE, NW, SE, and SW) and routes the packet to the corresponding VCs, assigned for that quadrant. The router architecture uses a speculative strategy based on lookahead information obtained from neighboring routers for providing routing adaptation. A key aspect of the proposed design is a low latency feature that makes the lookahead information more representative than possible in many existing router architectures with higher latencies. Further, the router employs a preselection mechanism for the output channels that helps to reduce the complexity of the crossbar switch design. In addition, based on this partitioned VCs, we use a decomposed crossbar that needs only half the size (connects) of a full crossbar, thereby reducing packet conflict probability in the crossbar. The router can support both deterministic and adaptive routing.

We evaluated the proposed router architecture by using it in a 2D mesh and performing cycle-accurate simulation of the entire NoC design using various workloads. The experimental results reveal that the proposed architecture results in lower latency than when using a router with a deeper pipeline. This is due to the low-latency routers to provide more current congestion information then is possible with high latency routers. We also demonstrate that adaptability provides better performance when compared to deterministic routing for various workloads. We then used a router design, laid out, to evaluate our design from an energy standpoint, obtaining both dynamic and leakage energy consumption of the router. Our results indicate that for non-uniform traffic, our adaptive routing algorithm consumes less energy than the dimension order routing due to the decrease in the overall network latency. Evaluation of our architecture in a 2-D mesh using various traffic patterns shows that it can provide better performance (lower latency) and energy conservation compared to a conventional lookahead router.

4.2 Proposed Router Model

Minimization of message latency by optimizing the intra-node delay and utilizing organized wiring layout with regular topologies has been targeted in NoC designs. The proposed router, designed with this objective, consists of a two-stage pipelined model with look ahead routing and speculative path selection [9, 130]. In this section, we present a customized router architecture for on-chip networks that can support deterministic, and adaptive routing in 2-D networks.

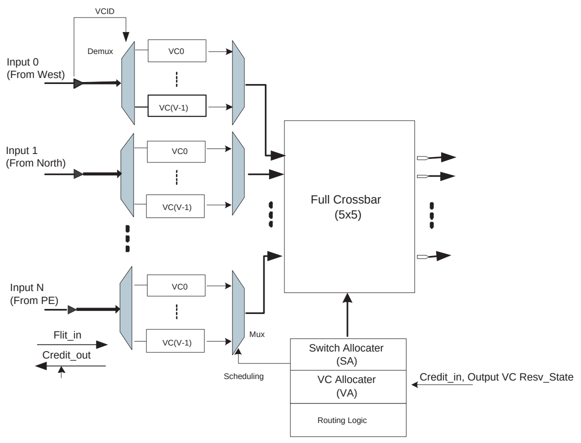


Figure 4.1. A 2-Stage Virtual Channel Look-ahead Router

4.2.1 NoC Router Architecture

A typical state-of-the-art, wormhole-switched, virtual channel (VC) flow control router consists of four major modules: routing control (RC), VC allocation (VA), switch allocation (SA), and switch transfer (ST). In addition, it may have an extra stage at each input port for buffering and synchronization of arriving flits. A generic virtual-channel-based wormhole router is shown in Figure 4.1. Incoming packets are placed into the the input buffers according to their corresponding virtual channel identification (VCID). The RC assigns a set of valid out-going VCs upon which the packet can be routed. The VA arbitrates between competing packets requesting the same output VC, and allocates an unused output VC to a winning packet. Each packet keeps state information for the availability of buffer space at their assigned output VC. Provided flits are selected to be sent, and next router's buffer space is available, input VCs request the necessary crossbar connections to their valid output channels. The SA module tracks and matches these requests to output ports on a cycle-by-cycle basis. Flits traverse the crossbar switch toward their destined nodes.

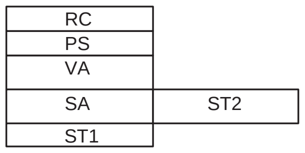
Pipelining the router architecture can significantly improve performance by increasing throughput much like the pipeline of a microprocessor, thereby reducing the average latency. As described in [130], there are certain critical components that are best kept intact within a pipeline stage. These atomic modules represent the finest granularity at which efficient pipelining can occur. The RC, VA, SA and crossbar represent the fundamental modules within an NoC router. By employing clever techniques such as Speculative Switch Allocation [130] and Look-ahead Routing [?], we have been able to break some of these interdependencies by parallelizing operations, thus shortening the router's critical path. This has led to 2-stage router implementations.

4.2.2 A Path-Sensitive Router Architecture

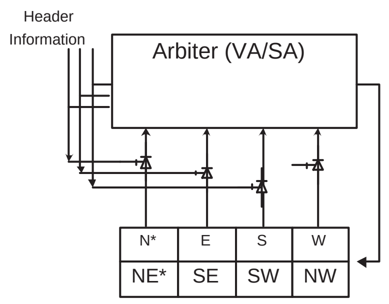
Figure 4.2(a) illustrates the logical modules of our two-stage pipelined router incorporating look ahead routing and speculative allocation. The first stage of the router performs look ahead routing decision for the next hop, pre-selection of an optimal channel for the incoming packet (header), VA and SA in parallel. The actual flit transfer is essentially split in two "stages", the preliminary semi-switch traversal through the VC selection (ST1) and the decomposed crossbar traversal (ST2).

In contrast to the prior router architectures, the proposed model incorporates a pre-selection (PS) unit as shown in Figure 4.2 (b). The PS unit uses the current switch state and network status information to decide a physical channel (PC) from among the possible paths, computed during the previous stage look-ahead decision. Thus, when a header flit arrives at a router (stage *i*), the RC unit decides a possible set of output PCs for the next hop (*i+1*). The possible paths depend on the destination tag and the routing algorithm employed. Instead of using a routing table for path selection, we use a hardware control that computes a 3-bit direction vector, called Virtual Channel ID (VCID), (based on the four destination quadrants (NE, NW, SE, and SW) and four directions (N, E, S and W)), which can be sent along with the header, for deciding an optimal path using the PS unit in the next hop. The VA and SA are then performed concurrently for the selected output, before the the transfer of flits across the crossbar.

The detailed design of the router is shown in Figure 4.3. We describe its functionality with respect to a 2-D mesh. The router has five inputs, marked for convenience, from the four directions and one from the local PE. It has four sets of VCs, called path sets; one set for possible traversal in each of the four quadrants; NE, SE, NW, SW. Each path set has three groups of VCs to hold flits from possible directions from the previous router.



RC (Routing Computation), PS (Pre-selection), VA (Virtual Channel Allocation), SA (Switch Allocation), ST1 (Switch Traversal to Path Candidate), ST2 (Crossbar Switch Traversal)
(a) Router Pipeline



*NE (North East Quadrant), N (Output port for North)
(b) Direction Vector Table in PS

Figure 4.2. The Two-Stage Pipelined Router

This grouping is customized for a 2-D interconnect. For example, a flit will traverse in the NE quadrant only if it is coming from the west, south or the PE itself. Similarly, it will traverse the NW quadrant if the entry point is from the south, east or the local PE. Thus, based on this grouping, we have 3 VCs per PC. Note that it is possible to provide more VCs per group if there is adequate on-chip buffer and the VA selects one of the VCs in a group.

The MUX in each path set selects one of the VCs for crossbar arbitration. In the mean time, the PS generates the pre-selection enable signals to the arbiter based on the credit update, and congestion status information of the neighboring routers. The arbiter, in turn, handles the crossbar allocation.

Another novelty of this router is that since we are using topology tailored routing, we use a 4×4 decomposed crossbar with half the connections of a full crossbar as shown in Figure **??** (b). Usually a 5×5 crossbar is used for 2-D networks with one of the ports assigned to the local PE. The decomposed crossbar offers two advantages for on-chip design. First, it needs less silicon and second, it consumes less energy compared to a full crossbar. In addition, because of a lower number of connections, the output contention probability is reduced. This, in turn, should help in reducing the mis-speculation of the VA and SA stages.

Path-Sensitive Decoder: In our proposed architecture, the input decoders (DEMUXes) undertake a more significant role than in a generic router for supporting routing adaptivity utilizing look-ahead information in selecting an output path. In conventional architectures, the input decoders can only distribute incoming flits

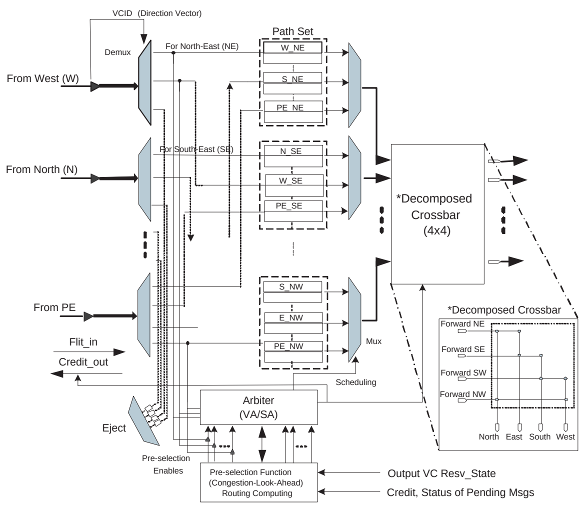
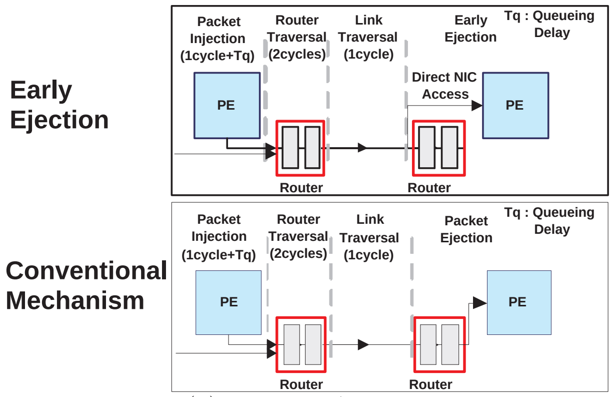


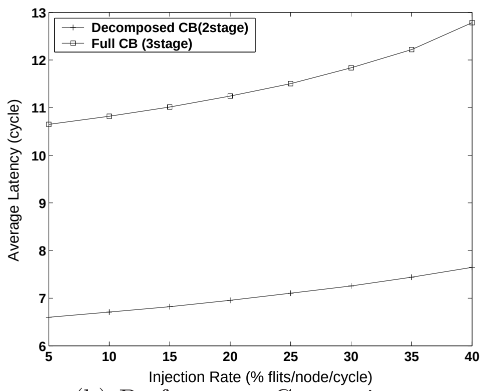
Figure 4.3. Proposed Path-Sensitive Router

to the VC buffers of a single port set (see Figure **??** (a)). In the Path-Sensitive architecture however, the input decoders can distribute flits to multiple path sets. This mechanism amounts to a preliminary switching operation without an additional clock cycle and any extra control logic. Furthermore, it significantly alleviates contention later on in the crossbar by pre-arranging incoming flits according to their desired output port. The area consumed by the router is dominated by the buffers. Therefore, while this reconfiguration increases wiring complexity, the wires have plenty of space to be routed above the buffers in upper layers of the chip, thus imposing minimal overhead.

Early ejection: A flit destined for the local PE does not traverse the crossbar, but instead, is ejected immediately upon arrival (hence the term "Early Ejection"). This mechanism utilizes the look-ahead routing information to detect if the incoming flit is destined for the local PE and accordingly ejects it after the DEMUX as shown in Figure **??** (b). This early ejection saves two cycles at the destination node by avoiding switch allocation and switch traversal (Figure 4.4 (a)). Also, it reduces the input load for each of the crossbars' input ports. This provides a significant advantage for nearest-neighbor traffic and can take advantage of NoC mapping, which places frequently communicating PEs close to each other [131]. We simulated the router architecture employing Early Ejection for nearest-neighbor traffic in order to investigate the impact on average latency. Figure 4.4 (b) illustrates that early ejection and intra-router latency reduction jointly decrease the overall network latency.



(a) Latency Analysis



(b) Performance Comparison

Figure 4.4. Early Ejection Mechanism and its Advantage for Nearest-Neighbor Traffic

4.2.3 Pre-selection Function

The pre-selection function, as outlined earlier, is responsible for selecting the channel for a packet. It maintains a direction vector table for the path selection and works one cycle ahead of the flit arrival. The direction vector table, as shown in Figure 4.2 (b), selects the optimal path for each of the four quadrant path sets. As an example, if a flit is in the W_NE VC, the PS decides whether it will use the N or E direction and inputs this enable signal to the arbiter. The PS logic uses the congestion look-ahead information and crossbar status to determine the best path, and also immediately updates the credit priorities into the arbitration. The congestion look-ahead scheme provides adaptive routing decisions with the best VC (or set of VCs) and path selection from the corresponding candidates based on the congestion information of neighboring nodes.

An adaptive routing algorithm either needs a routing table or hardware logic to provide alternate paths. Our proposed router can support deadlock-free fully adaptive routing for 2-D mesh and torus networks using hardware logic. Note that a flit can use any minimal path in one of the four quadrants, as discussed in the router model in Section 4.2.2. The final selection of an optimal channel is done by the PS module. It can be easily proved that the adaptive routing is deadlock free for a 2-D mesh because the four path sets, shown in Figure 4.3 are not involved in any cyclic dependency. Whenever two flits from two quadrants are selected to traverse in the same direction (for example a flit from NE and a flit from SE contend for an east channel), they use separate VC's in the NE and SE path sets, thereby avoiding channel dependency.

Although an adaptive routing algorithm helps in achieving better performance, specifically under non-uniform traffic patterns, the underlying path selection function has a direct impact on performance. Most adaptive routing algorithms take little account of temporal traffic variation and other possible traffic interferences.

The proposed adaptive routing algorithm utilizes look-ahead congestion detection for the next router's output links using a credit-based system. The PS module keeps track of credits for each neighboring router on the candidate paths. Neighboring routers send credits indicating congestion (VC state) with the amount of free buffer space available for each VC. In addition, time between successive transmissions to neighboring routers is kept minimal due to the low latency of our architecture, capturing traffic congestion in short spurts and reacting accordingly. This helps in capturing congestion fluctuation and picking the right channel from amongst the available candidate channels.

4.2.4 Hardware Implementation

The router architecture was implemented in structural Register-Transfer Level (RTL) Verilog and then synthesized in Synopsys Design Compiler using the TSMC 90 nm cell library. The resulting design operates at a supply voltage of 1 V and a clock speed of 500 MHz. Both dynamic and leakage power estimates were extracted from the synthesized router implementation, using a 50% switching activity. These power numbers were then imported into our cycle-accurate network simulator and used to accurately portray the power profile of the entire on-chip network. While some researchers have modeled power consumption based solely on hop traversals per flit/packet [30], we also account for individual router component utilizations to precisely estimate the dynamic and leakage envelope of each router throughout simulation. This allows for a more realistic estimation when handling different flit types. For example, a header flit requires processing by the routing, preselection, and virtual channel allocation units in the router, while a data or tail flit does not. This attribute affects the different component utilizations during the simulation and, as a result, the power consumption is accurately reflected in the power estimation model.

4.3 Performance Evaluation

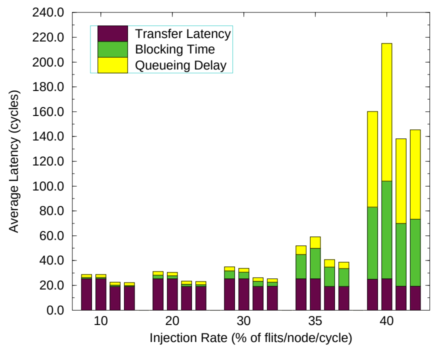
In this section, simulation-based performance evaluation of our architecture in terms of network latency and energy consumption under various traffic patterns is presented.

We evaluated our architecture using two 8x8 networks, one using 2D torus and the other using 2D mesh topologies. We use 4 flits per packet and wormhole routing, with 4-flit deep buffers per VC. We use 3 VCs per PC for the 2D mesh and 6 VCs per PC for the 2D torus (as required by our algorithm). Every simulation initiates a warm-up phase of 20,000 packets. 1,200,000 packets were injected and the simulation was conducted until all the packets are received at the destination nodes. We then log the simulation results and gather network performance metrics based on our simulations for various traffic patterns. For packet injection rates, we used three workload traces: self similar web traffic, uniform and MPEG-2 video multimedia traces. We simulated each trace using four different traffic permutations: normal-random, transpose, tornado and bit-complement [9].

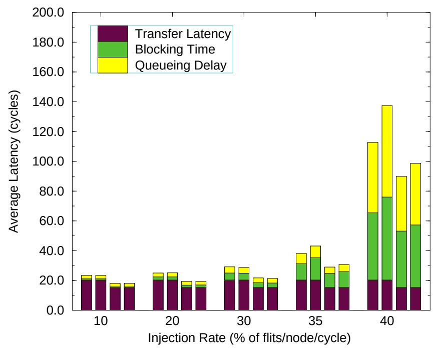
4.3.1 Simulation Results

First we compare our proposed 2-stage router architecture to a 3-stage pipelined router in order to evaluate the average network latency. In the 3-stage router, we implement both a dimension-order (DOR) algorithm as well as our adaptive routing algorithm. We separate the crossbar arbitration from the routing decision stage resulting in three stages in the router pipeline when using our adaptive algorithm. A direct effect of this is a full crossbar switch instead of the decomposed crossbar our 2-stage architecture employs. We compare both routers (2-stage vs. 3-stage) using both DOR and our adaptive algorithm.

We partition the average latency in three parts: contention free transfer latency, blocking time, and queuing delay in the source node. In both zero load latency, and congestion avoidance, the 2-stage router performs better than the 3 stage



(a) 2D Mesh Network



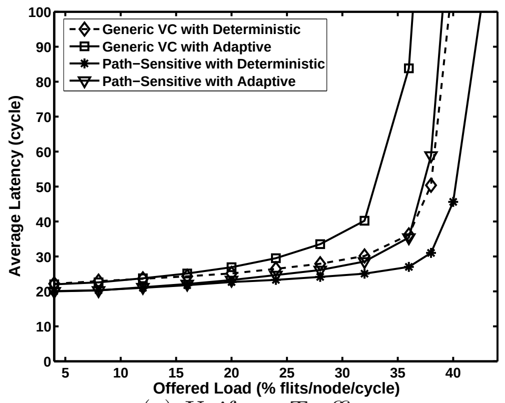
(b) 2D Torus Network

Figure 4.5. Comparison of 2-stage vs. 3-stage routers utilizing both dimension-order and adaptive routing algorithms. [For each injection rate, from left to right, columns correspond as follows: 3-Stage DOR, 3-Stage Adaptive, 2-Stage DOR, 2-Stage Adaptive]

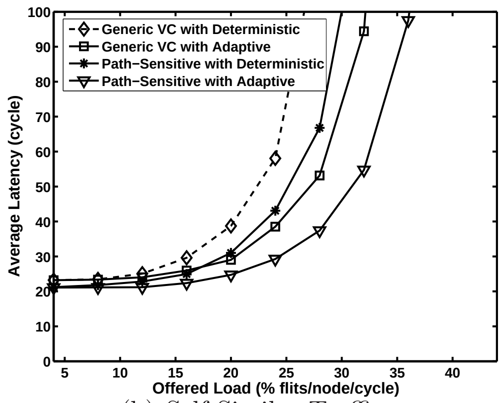
router. Applying the 2-stage system to a generic router, the proposed architecture outperforms by upto 28% under uniform traffic in 4.6 (a). Figure 4.5 also shows the impact of adaptivity, indicating poor performance when compared to DOR under uniform random traffic. Spatial traffic evenness benefits load balancing in DOR, even if we improve adaptivity.

In the presence of non-uniform traffic, however, the adaptive algorithm outperforms the DOR algorithm as shown in Figures 4.6 (b), 4.7, and 4.8 for transpose (TP), self-similar(SS) and Multimedia(MM) traffic patterns. As bursty traffic is more representative of a practical on-chip communication environment, we expect our adaptive algorithm to be very useful.

The increased hardware complexity in implementing the adaptive routing algorithm results in a higher power consumption than when implementing the simpler, DOR algorithm. However, the latency reduction when using adaptive algorithm for non-uniform, more practical traffic environments, more than compensates for this power increase, reducing the overall energy. Figure 4.9(a)shows the overall energy consumption for uniform and transpose traffic when comparing adaptive routing vs. DOR. In Figure 4.9(b), we also see that adaptive routing is beneficial when we consider the Energy-Delay Product for non-uniform traffic.

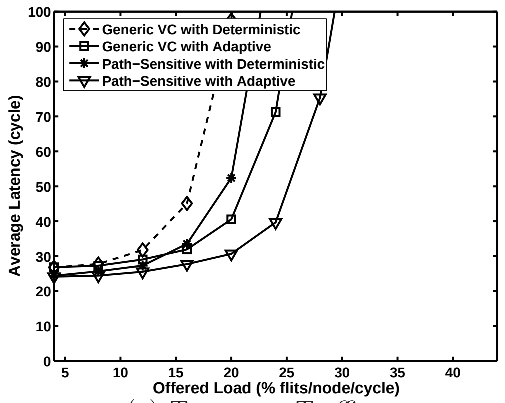


(a) Uniform Traffic

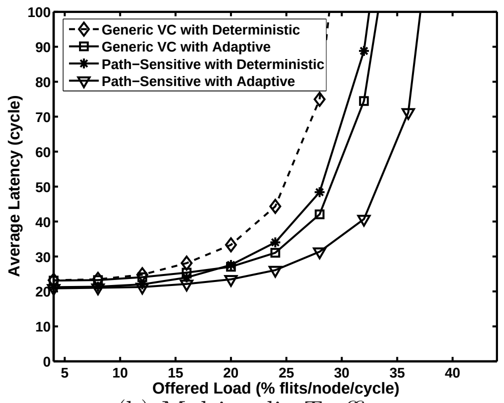


(b) Self-Similar Traffic

Figure 4.6. Performance Comparison



(a) Transpose Traffic

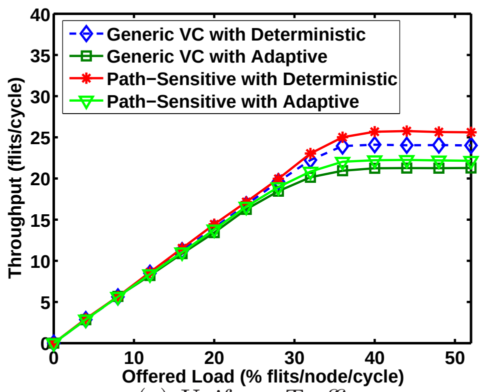


(b) Multimedia Traffic

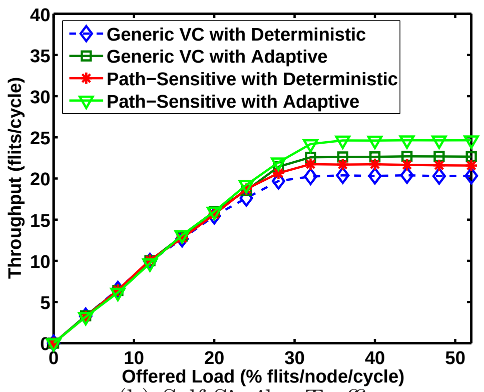
Figure 4.7. Average Latency under Non Uniform Traffic

4.4 Conclusions

We proposed a low-latency router architecture suitable for on-chip networks which utilizes an adaptive routing algorithm. The architecture uses a novel path selection scheme resulting in a 2-stage switching operation with a decomposed crossbar switch. We evaluate our architecture using various traffic patterns and we show that under non-uniform traffic our architecture reduces the overall network latency by a significant factor. Due to the low latency of our architecture congestion information about each router's neighboring routers is swiftly updated, resulting in almost real-time updates. We also show that the energy consumption when using adaptive routing is also reduced due to the reduction in the network la-

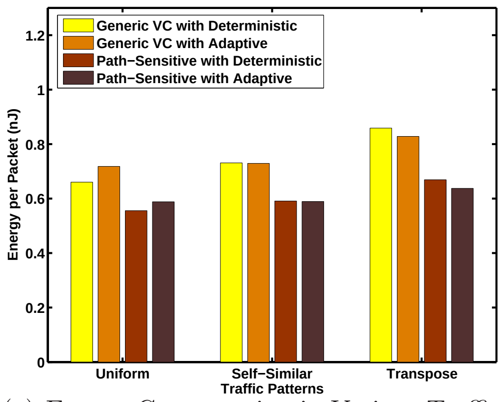


(a) Uniform Traffic

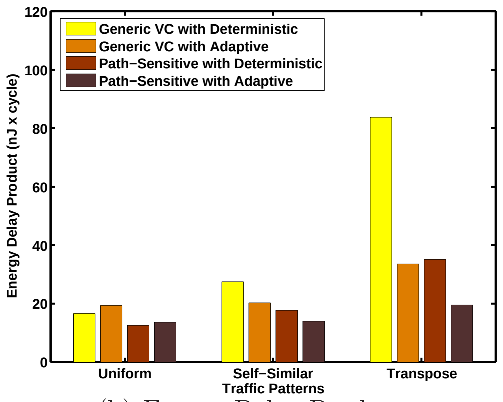


(b) Self-Similar Traffic

Figure 4.8. Throughput with Uniform and Self-Similar Traffic



(a) Energy Consumption in Various Traffic Patterns



(b) Energy Delay Product

Figure 4.9. Energy Consumption

tency. In addition, our improvements result in better performance in the presence of nearest-neighbor traffic, a particular benefit for NoC designs which emphasize spatial locality.

Chapter 5

An Analytical Model for Performance, Power and Reliability

The previous chapter discussed a low-latency model based on path-sensitive mechanisms, suitable for supporting adaptive routing in tiled SoC architectures. In this chapter, a queuing-theory-based model is proposed for evaluating the performance and energy behavior of on-chip networks. Then the model is used to demonstrate the effectiveness of our proposed Path-Sensitive Router. The performance (average latency) and energy consumption results from the analytical model are validated with those obtained from a cycle-accurate simulator.

While researchers have examined the area-constraint design alternatives [19, 20], energy models [22] or fault-tolerance issues [26, 25, 29, 27, 30, 28] individually, a systematic design methodology encompassing the interplay of performance, fault-tolerance and energy constraints is yet to evolve. Such a design methodology is impeded, in part, due to the lack of efficient techniques to quickly explore alternatives from a large design space. While most prior on-chip interconnect analyses are based on time consuming simulation models, in order to provide fast performance estimates during the design cycle, we have developed a queuing-theory-based model for quantifying the performance and energy behavior of on-chip networks.

Although wormhole switched, traditional off-chip networks have been analyzed extensively in the literature [132, 133], analytical models for NoCs considering detailed architectural artifacts are almost nonexistent. To our knowledge, the first analytic techniques [90] appeared recently. Our model is different from [90] in that we compute the average delay due to path contention, virtual channel and crossbar switch arbitration using a queuing theory approach, which we believe, can capture the blocking phenomena of wormhole switching quite accurately. We first present the model for performance analysis for a generic wormhole switched router. The model is then used to estimate the power consumption by estimating the utilization of the router components and multiplying them with component level power profiles, obtained from actual circuit-level synthesis. Comparison with simulation results indicate that the proposed analytical model is quite accurate and can be used as an efficient design tool. An extension of the proposed analytical model is also shown to demonstrate the utility of the model within the domain of fault-tolerance to capture the impact of error detection and retransmission mechanisms.

Our transient fault resilience enhancement covers both link errors and logic (component) errors in a router. For the link errors, we analyze five possible retransmission mechanisms, and argue that a separate header error checking based retransmission is a better choice than conventional End-to-End (E2E) and Hop-by-Hop (HBH) techniques because packet misrouting can be alleviated by identifying header bit errors. Then, we propose several architectural and algorithmic safeguards to our two-stage router architecture to protect against six types of intra-router soft faults without inducing prohibitive area, power and latency overheads. To the best of our knowledge, this is the first attempt to account for and protect against internal router soft faults in on-chip networks.

The chapter is organized as follows. Sections 5.1 and 5.2 present our analytical model for latency and power consumption analysis. Section 5.3 describes the reliability issues resulting from link errors and transient faults within the router components and proposes several protection methods. The conclusions of this study are drawn in the last section.

5.1 Performance Analysis of NoC Architectures

In this section, we present an analytical model for computing the average latency in a 2-D mesh network. The average network latency consists of two parts. The first part is the actual message transfer time. The second part is due to blocking, which is mostly caused by the conflicts at a VA, contention at a SA and limited buffer size. The actual transmission time with a P-stage pipelined router is $(P-1+M)$ cycles for an M-flit packet. In order to compute the second part of the network latency, let us define B_j as the average blocking length (in number of flits) seen by each incoming flit at the input and arbitration stages of a router j. B_j captures flit blocking in a pipelined virtual-channel router. Then, the effective length of the packet becomes $(M + B_j)$ flits. Let W_j be the average waiting time required to transfer one flit to the next slot in the router queue. Thus, the network latency for a single router (T_j) is

$$T_j = (M + B_j)W_j + P - 1. \tag{5.1}$$

Let T_{link} be the delay through the link connecting adjacent routers. We compute the the latency (T) for an M-flit packet to traverse through a P pipelined router for a given *path* as

$$T = \sum_{j \in path} ((B_j W_j + P) + T_{link\_j}) + ((B_{dest} + M)W_{dest} + P - 1). \tag{5.2}$$

The first term in Equation 5.2 represents the time spent at intermediate hops, and the second term denotes the time at the ejection node. Note that this does not include the queuing delay outside the router. For simplicity, using the average blocking length (B), the average waiting time (W) and the average distance (H hops) on a given *path*, and assuming $T_{link} = onecycle$, the total latency can be approximated to

$$T \approx (BW + P + 1)H + ((B_{dest} + M)W_{dest} + P - 1). \tag{5.3}$$

Now we will derive the blocking length (B and B_{dest}) and waiting time (W and W_{dest}) in terms of the contention probability (P_{con}) and the buffer-full probability (P_{block}), so that we can determine the total latency.

5.1.1 Contention Probability

The router model assumes an $(N+1) \times (N+1)$ router with V virtual channels per physical channel and a deterministic routing algorithm. Our network model is characterized on a flit basis, which is appropriate for virtual channel router architectures. We assume that flit arrivals at the N inputs from neighbors and at the local input from the PE are governed by independent and identical Poisson processes. Let the probability of a flit arriving at an input virtual queue per cycle be P_c. Note that it is the normalized arrival rate [134]. Let the flit injection probability into each virtual channel queue in any given cycle from a PE be P_{pe}. The flits are assumed to travel H hops on the average. Let the incoming flits have equal probability $1/N$ of being addressed to any given output at V VCs. Thus, we can compute the P_c from P_{pe} as follows:

$$P_c = \frac{P_{pe}H}{NV}. \tag{5.4}$$

The total contention probability (P_{con}) consists of the VA conflict probability (P_{con_va}) and the SA contention probability (P_{con_sa}). They are independent of each other in a speculative virtual channel router, and thus P_{con} is given by

$$P_{con} = 1 - (1 - P_{con\_va})(1 - P_{con\_sa}). \tag{5.5}$$

First, we analyze the VA conflict (P_{con_va}), which has two parts, P_{ready_busy} and P_{con_idle}. The first part is the probability (P_{ready_busy}) that the candidate output VCs are already used by other packets. P_{ready_busy} is the probability there is at least one header flit destined to a free output VC, multiplied by the effective packet length, since the VC will be reserved for the entire packet transmission. Thus, P_{ready_busy} in a given cycle is given by

$$P_{ready\_busy} = \left(1 - \left(1 - \frac{P'_h}{RV}\right)^{NV} \cdot \left(1 - \frac{P'_{peh}}{RV}\right)\right)(M+B). \tag{5.6}$$

We derive P_{ready_busy} and P_{con_idle} as a function of the probability of a header flit arriving in an arbitrary time slot (P_h), which is calculated as P_c/M. When k header flits are destined to a particular output VC, one of the k header flits is granted an output VC, while the rest $k-1$ headers are blocked at the respective input queues. These $k-1$ blocked headers can be regarded as independent header flit arrivals, and thus the average probability of an incoming header flit at each VC queue can be approximated to P'_h as follows:

$$P'_h = P_h(1 + P_{con\_va} + P^2_{con\_va} + ...) = \frac{P_h}{1 - P_{con\_va}}. \tag{5.7}$$

The P'_{peh} term can be similarly obtained by computing the header flit injection probability P_{peh} from a local PE. The R term in Equation 5.6 is the number of the output ports that can be addressed by routing algorithms. For example, a packet placed into the east-side input queues may be routed to 4 output ports including the ejection channel with a fine-granularity, while the south channel inputs may have 2 output candidates with coarse granularity using the Dimension Ordered Routing (XY Routing) algorithm in a two dimensional network. On the other hand, a packet incoming to any input may be destined to 4 output ports using an adaptive routing algorithm.

Next, we compute P_{con_idle}. In the steady state, $(1 - P_{ready\_busy})$ represents the probability that a particular output VC is idle and available. The probability that a header flit at the head of a queue can be routed to a particular VC is $\frac{P'_h}{R} \cdot \frac{1}{v}$, where v represents the number of unused VCs per output port, which is computed by $(1 - P_{ready\_busy})V$. Thus, we can compute the probability, $P'_h(j)$ such that j headers out of $NV - 1$ buffers are addressed to a free output VC, as

$$P'_h(j) = \begin{pmatrix} NV - 1 \\ j \end{pmatrix} \left(\frac{P'_h}{R}\frac{1}{v}\right)^j \cdot \left(1 - \frac{P'_h}{R}\frac{1}{v}\right)^{NV-1-j}. \tag{5.8}$$

Assuming random arbitration, the contention probability (P_{con_idle}) can be calculated using Equations 5.8 and 5.9. When a header flit of a particular input is assumed to request a output VC, the first term in Equation 5.9 represents the probability that more than two headers (one comes from the request of this particular input, others ($j >= 1$) from different inputs and no header from the local PE) are destined toward that output VC. In this case, the probability that a header fails to be granted a VC is $j/j + 1$, since (j) headers among $j + 1$ headers should wait for the next arbitration. The second term denotes the probability that a header from the local PE is addressed to the same output VC. We consider the incoming effective header probability (P'_{peh}) at the injection link from the PE as

$\frac{P_{pe}/M}{1-P_{con\_va}}$. Thus, the contention probability, P_{con_idle} can be estimated as

$$P_{con\_idle} = \sum_{j=1}^{NV-1} \frac{j}{j+1} \Big\{ P'_h(j) \left(1 - \frac{P'_{peh}}{R}\frac{1}{v}\right) + P'_h(j-1) \left(\frac{P'_{peh}}{R}\frac{1}{v}\right) \Big\}. \tag{5.9}$$

Thus, we have P_{con_va} as a combination of P_{ready_busy} and P_{con_idle}, which are recursively correlated. In the steady state, the VA conflict probability can be evaluated as

$$P_{con\_va} = \left(\frac{P'_h}{R}\right) \Big\{ (P_{ready\_busy})^V + \sum_{v=1}^{V} \begin{pmatrix} V \\ v \end{pmatrix} (P_{ready\_busy})^{V-v} \cdot (1 - P_{ready\_busy})^v \frac{1}{v} (P_{con\_idle}) \Big\}. \tag{5.10}$$

Another contention exists at the SA module. The contention probability, P_{con_sa}, consists of two parts. One is the input port contention probability ($P_{con\_sa\_in}$), and the other is the output port contention probability ($P_{con\_sa\_out}$). We can compute the SA contention probability as follows:

$$P_{con\_sa} = 1 - (1 - P_{con\_sa\_in})(1 - P_{con\_sa\_out}). \tag{5.11}$$

$P_{con\_sa\_in}$ results from the sharing of an input port by V virtual channels at a V input arbitration, which is given by

$$P_{con\_sa\_in} = \sum_{i=2}^{V} \frac{i-1}{i} \begin{pmatrix} V \\ i \end{pmatrix} P'^{i}_c (1 - P'_c)^{V-i}. \tag{5.12}$$

The output port contention probability is analyzed similar to P_{con_idle}, but it is different in that the allocation process is made on every flit including the data flits. In order to compute the output port contention probability, $P_{con\_sa\_out}$, let us define the incoming flit probability at the input of a physical channel, P_p, as $(1 - (1 - P'_c)^V)$, where $P'_c = \frac{P_c}{1-P_{con}}$. The effective physical channel utilization, P'_p, is also recursively computed as $P'_p = P_p/(1 - P_{con\_sa})$. Similar to Equation 5.8, the probability that k flits compete for a particular output port, assuming that each flit has an equal probability $1/R$ of being destined to any output, is

$$P'_p(k) = \begin{pmatrix} N-1 \\ k \end{pmatrix} \left(\frac{P'_p}{R}\right)^k \left(1 - \frac{P'_p}{R}\right)^{N-1-k}. \tag{5.13}$$

Similar to Equation 5.9, the first term in Equation 5.14 indicates the probability that more than 2 flits are destined to a particular output with no flit coming from the local PE. The second part takes into account the probability of an incoming flit from the local PE, P'_{pep}. The $k/k+1$ term indicates that only one of the competing flits is granted the output port, while the rest have to be included in the contention probability calculation. Thus, the output port contention probability is

$$P_{con\_sa\_out} = \left(\frac{P'_p}{R}\right) \sum_{k=1}^{N-1} \frac{k}{k+1} \Big\{ P'_p(k) \left(1 - \frac{P'_{pep}}{R}\right) + P'_p(k-1)\frac{P'_{pep}}{R} \Big\}. \tag{5.14}$$

By integrating the virtual channel conflict probability (P_{con_va}) and the switch allocation contention probability (P_{con_sa}), we can determine the total contention probability, P_{con} as a function of P_c.

Figure 5.1 illustrates the trend of contention probability for flits at a row (x-dimension) input port at a distributed single router for various network sizes and input load with uniform traffic, when DOR is used. The effective flit arrival probability at an input virtual queue, P_c, is determined by the local injection probability (P_{pe}) and the average distance (H hops), where the average distance is associated with network size in uniform traffic. The flit must travel H hops on the average along the path between source and destination. The effective flit arrival probability increases as the network size scales up, which leads to high contention probability.

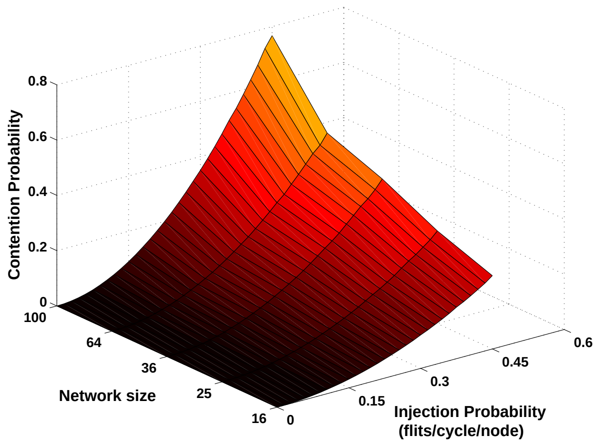


Figure 5.1. Contention Probability for Various Network Sizes and Input Load

5.1.2 Buffer Design Influence of Contention Probability

Figure 5.2 shows the contention probabilities of row (x-dimension) and column (y-dimension) input ports with Dimension-Order Routing (X-Y routing) in an 8x8 network. The contention probability of the row (the east and the west) input is higher than that of the column (the north and the south) input. The Switch Allocation (SA) has more influence on the overall contention probability than the Virtual Channel Allocation (VA) module.

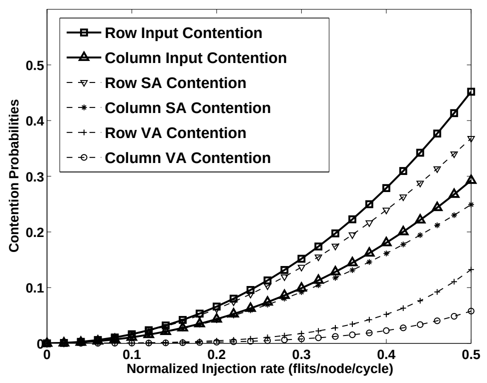


Figure 5.2. Contention Probabilities

With X-Y routing, a packet must first travel all possible hops in the X direction before moving to the Y direction. Therefore, any message moving in the Y direction can never return to the X direction. This means that while it is possible for all of the other ports to attempt to send a message to the north port in a given cycle, it is possible for only two ports (the west port, and the port attached to the PE) to send a message to the east direction in a given cycle. This behavior can be exploited by allocating more buffer resources in the east and the west input ports, where congestion is more likely to occur, than to the north and the south input queue. Table 5.1 summarizes the performance advantages in a 2D mesh topology due to different buffer allocatus at 35% injection rate.

Buffers	1NS,1EW	2NS,1EW	1NS,2EW	2NS,2EW
Normalized Latency	1	0.88	0.79	0.71

Table 5.1. Influence of Buffer Allocation on Packet Latency

While the two mixed buffer size configurations ((2NS,1EW) and (1NS,2EW)) have the same area and power requirement, the configuration with an additional buffer in the the east and west input queue has a 14% smaller latency. This buffer adjustment helps to reduce the buffer blocking probability, and improve performance. This scheme increases switch and link utilization.

5.1.3 Modeling Finite Size Buffer in NoCs

We should also consider the probability of a full buffer in NoCs, since the VCs can hold a fixed number of flits. The buffer unavailability probability (P_{block}) can be estimated from the probability $P_{hold}(r)$ that r flits are waiting in a queue and the traffic intensity (ρ), which are estimated by using a discrete-time state transition diagram as shown in Figure 5.3.

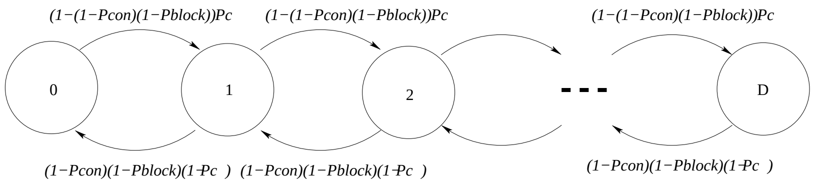


Figure 5.3. State Transition Diagram for a Finite Buffer

From Figure 5.3, we can also obtain $P_{hold}(r)$ as

$$\begin{aligned} P_{hold}(r) &= P_{hold}(r-1)(1-(1-P_{con})(1-P_{block}))P_c \\ &\quad +P_{hold}(r)(1-(1-P_{con})(1-P_{block}))(1-P_c) \\ &\quad +P_{hold}(r+1)(1-P_{con})(1-P_{block})(1-P_c) \\ &\quad +P_{hold}(r)(1-P_{con})(1-P_{block})P_c \end{aligned} \tag{5.15}$$

As shown in Figure 5.3, $P_{hold}(r)$ is related to P_{block} and is recursively obtained by computing P_{block} and the finite buffer size. For a D-flit buffer depth,

$$P_{block} = P_{hold}(r >= D). \tag{5.16}$$

We can also obtain the average number of flits in the queue, B, which is used in Equation 5.3, as

$$B = \sum_{r=0}^{D} r P_{hold}(r). \tag{5.17}$$

The average waiting time, W, can be estimated from the steady-state traffic intensity under uniform traffic density as follows:

$$W = 1 + \sum_{k=1}^{D} k P_B^k (1 - P_B), \ where \ P_B = 1 - (1 - P_{con})(1 - P_{block}). \tag{5.18}$$

In order to estimate the blocking length (B_{dest}) and the waiting time (W_{dest}) per flit at the input of the destination node, we can consider only the contention probability, P_{con} in Equation 5.5 without considering the buffer full probability, P_{block}, at the next node. Thus, we can get $P_{hold\_dest}(r)$ using Equation 5.15 as follows.

$$\begin{aligned} P_{hold\_dest}(r) &= P_{hold\_dest}(r-1) P_{con} P_c + P_{hold\_dest}(r) P_{con} (1 - P_c) \\ &\quad + P_{hold\_dest}(r+1)(1 - P_{con})(1 - P_c) + P_{hold\_dest}(r)(1 - P_{con}) P_c. \end{aligned} \tag{5.19}$$

Equations 5.17 and 5.18 can be changed to compute B_{dest} and W_{dest} as

$$B_{dest} = \sum_{r=0}^{D} r P_{hold\_dest}(r), \tag{5.20}$$

$$and \ \ W_{dest} = 1 + \sum_{k=1}^{D} k P_{con}^k (1 - P_{con}). \tag{5.21}$$

Now, we have all the parameters to estimate the average latency using Equation 5.3.

5.1.4 Modeling of the Path-Sensitive Router

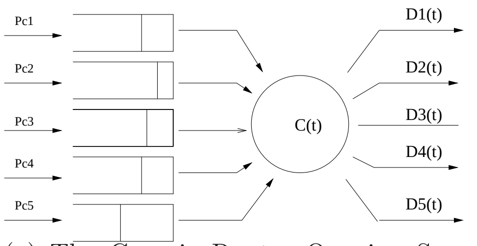
We now extend the model to analyze our path-sensitive router. In a generic architecture, flits arriving with the probabilities P_{c1} through P_{c5} are enqueued at the respective input port as shown in Figure 5.4(a), whereas in the path-sensitive router, the input probabilities are P_α, P_β, P_γ and P_δ (Figure 5.4(b)). Under the same traffic condition, the total incoming traffic per router ($\sum_{i=1}^{5} P_{ci}$) is larger than the total traffic ($\sum_{j=\alpha}^{\delta} P_j$) in the path-sensitive model by the flit ejection ($\cong$

injection) probability (P_{pe}). The lower arrival probability due to early ejection, influences both the average contention probability and blocking length at a higher workload. On the other hand, at low load, the average latency is dominated by the critical path in the pipeline stage. The flits traverse $H-1$ hops on the average in the path-sensitive model compared to H hops in the generic architecture. Thus, H in Equation 5.3 is replaced by $H-1$, and Equation 5.3 is changed to

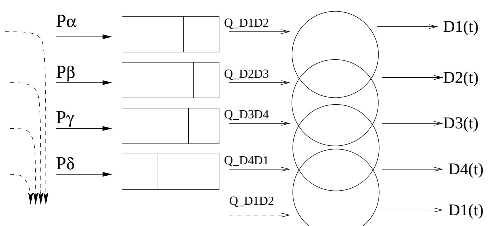
$$T_{ps} \approx (B_{ps}W_{ps} + P + 1)(H-1) + 1 + ((B_{ps\_dest} + M)W_{ps\_dest} + P - 1) \qquad (5.22)$$

The average blocking length (B_{ps}) and waiting time (W_{ps}) per flit are modified from the generic B and W, because of the changes in the VA contention probability (P_{con_idle}) and the SA output port contention probability ($P_{con\_sa\_out}$). The number of competing inputs for a particular output is reduced by half ($N+1$ to $\lfloor(N+1)/2\rfloor$), since the path-sensitive architecture sends incoming flits to a path candidate queue via the DEMUX according to the routing algorithm and direction vector ID as shown in Figure 5.4 (b). Thus, we can rewrite the probability ($P'_h(j)$) in Equation 5.8 that j headers are destined to an unused output VC as:

$$P'_{ps\_h}(j) = \begin{pmatrix} \lfloor(N+1)/2\rfloor V - 1 \\ j \end{pmatrix} \left(\frac{P'_{ps\_h}}{R_{ps}}\frac{1}{v}\right)^{j} \left(1 - \frac{P'_{ps\_h}}{R_{ps}}\frac{1}{v}\right)^{(\lfloor(N+1)/2\rfloor V - 1 - j)} . \qquad (5.23)$$



(a) The Generic Router Queuing System



(b) The Path-Sensitive Queuing Model

Figure 5.4. Queuing Systems of the Generic and the Path-Sensitive Router Models

There is no separate injection input port to access the crossbar, and injected flits are evenly spread out across other input ports. Note that the amount of early ejection is the same as injection. Thus, we can remove the effect of the incoming flits at the injection link from the PE in Equation 5.9. Thus, $P_{ps\_con\_idle}$ is given by

$$P_{ps\_con\_idle} = \sum_{j=1}^{\lfloor (N+1)/2 \rfloor V-1} \frac{j}{j+1} P'_{ps\_h}(j). \tag{5.24}$$

Similarly, Equations 5.13 and 5.14 are modified as follows, for the path-sensitive router as

$$P'_{ps\_p}(k) = \begin{pmatrix} \lfloor (N+1)/2 \rfloor - 1 \\ k \end{pmatrix} \left(\frac{P'_{ps\_p}}{R_{ps}} \right)^{k} \left(1 - \frac{P'_{ps\_p}}{R_{ps}} \right)^{(\lfloor (N+1)/2 \rfloor - 1 - k)}, \tag{5.25}$$

$$and \;\; P_{ps\_con\_sa\_out} = \left(\frac{P'_{ps\_p}}{R_{ps}} \right) \sum_{k=1}^{\lfloor (N+1)/2 \rfloor - 1} \left(\frac{k}{k+1} \right) P'_{ps\_p}(k). \tag{5.26}$$

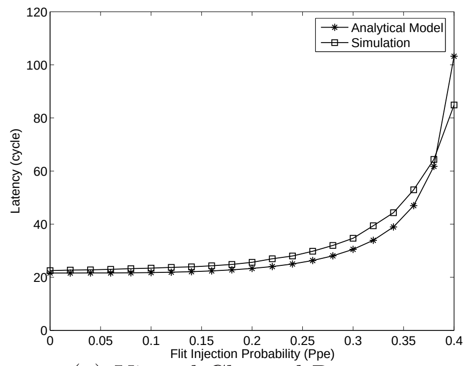
The number of competing input queues is reduced to $\lfloor (N+1)/2 \rfloor$, while the flit arrival probability for a particular output increases by $\frac{N}{N-1}$ in the example of Figure 5.4 (b). Thus, we have a lower contention probability, which results in lower average queuing length (B_{ps}) and waiting time (W_{ps}).

Figure 5.5 shows the comparison between our analytical model and a cycle-accurate simulator in terms of average packet latency. As we described in section 4.3, the model is based on an 8x8 mesh network with 4-flit packets, 3VCs per physical channel and a 4-flit buffer depth, where a deterministic routing algorithm is used. It is clear that the simulation and analytical values are in close agreement validating the correctness and accuracy of our mathematical model. In addition, the results show that the path-sensitive router performs slightly better than the generic router.

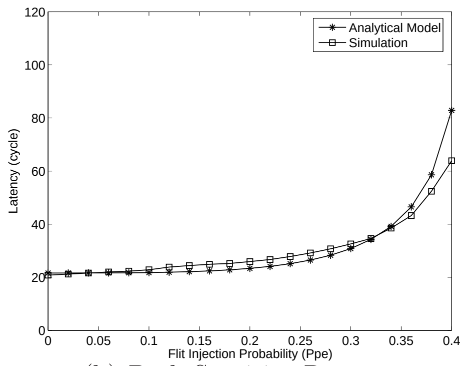
5.2 Power Model

We now extend our analytical model to include power calculations at the resolution of individual router components. For this, we will use the buffering delay and contention probability parameters from Section 5.1. The power dissipated in an NoC can be decomposed as follows:

$$P_R = \sum_{r \in PE} \left(P_{r\_buf} + P_{r\_arbiter} + P_{r\_crossbar} + P_{r\_link} \right), \tag{5.27}$$



(a) Virtual Channel Router



(b) Path-Sensitive Router

Figure 5.5. Performance Comparison of the two router models with Analytical Model and Simulation

where P_{r_buf} is the average buffer power consumption including both dynamic and static power, $P_{r\_arbiter}$ is the average power consumption in the routing computation, VA and SA modules, $P_{r\_crossbar}$ is the average crossbar traversal power consumption, and P_{r_link} is the average link power consumption between neighboring routers. P_{r_buf} can be estimated using the average flit arrival probability (P_c) in Equation 5.4 at each VC and average queue length (B) in Equation 5.17 as

$$P_{r\_buf} = (P_c(P_{wrt\_flit} + P_{rd\_flit}) + BP_{leak\_flit})(N+1)V, \tag{5.28}$$

where P_{wrt_flit}, P_{rd_flit} and P_{leak_flit} are the power consumptions for the write and read operations per flit, and the leakage power dissipation per flit. These power numbers are obtained from the synthesized design, as explained in section 2.2.3. In order to estimate the crossbar and link power consumption, we use the flit arrival probability (P_p) at a single physical link, which is computed as $(1-(1-P_c')^V)$ in section 5.1.1. $P_{crossbar\_flit}$ is the power dissipation of a flit traversal through each output port, and P_{link_flit} is flit power consumption per link. In a 2D mesh, the router at each edge has $N-1$ links and the four routers at the corners have $N-2$ links. Otherwise, they have maximum N connections per router. Similarly, the number of crossbar output ports varies from $N-1$ to $N+1$ including the ejection channel. Thus, $P_{r\_crossbar}$ and P_{r_link} are given by

$$P_{r\_crossbar} = P_p P_{crossbar\_flit} Max(N+1, N, N-1), \tag{5.29}$$

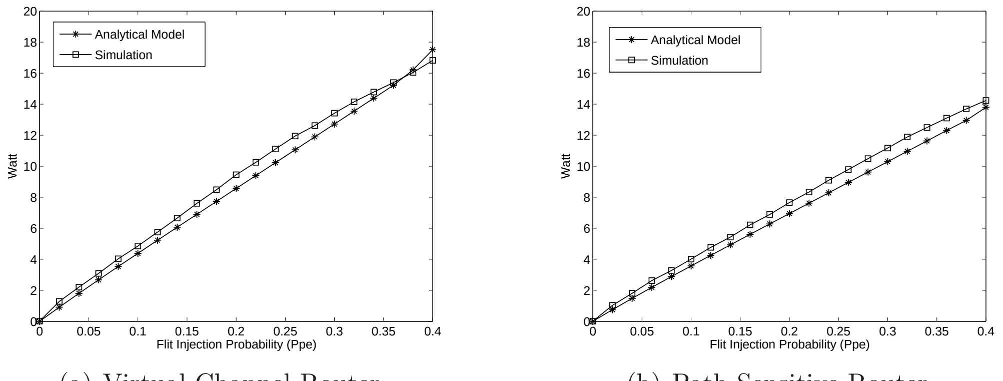
$$and \;\; P_{r\_link} = P_p P_{link\_flit} Max(N, N-1, N-2). \tag{5.30}$$

For computing $P_{r\_arbiter}$, we consider the probability of three separate activities and the power consumption for each activity. The first term is the probability that there is at least one header flit arrival (P'_h) to one of the (N+1)V VCs and the corresponding power consumption $P_{r\_arb\_va}$. The second term represents the probability that there is at least one data flit that needs SA operation (P'_p) and the corresponding power consumption $P_{r\_arb\_sa}$. The last term is the probability of a header arrival (P_h) to one of the VCs and the corresponding power consumption ($P_{r\_rt\_hflit}$). Thus, $P_{r\_arbiter}$ is estimated as

$$P_{r\_arbiter} = (1-(1-P'_h)^{(N+1)V})P_{r\_arb\_va} + (1-(1-P'_p)^{(N+1)})P_{r\_arb\_sa} + P_h P_{r\_rt\_hflit}(N+1)V. \tag{5.31}$$

The three elements (P'_h, P'_p and P_h) are combined here to illustrate power consumption in the above equation. The three power consumption parameters ($P_{r\_arb\_va}$, $P_{r\_arb\_sa}$, and $P_{r\_rt\_hflit}$) are obtained from the synthesized design. The leakage power is assumed to be negligible in logic and link propagation.

Figure 5.6 compares results from the analytical model and a cycle-accurate simulation. The simulation environment is the same as in Section 4.2.3. The results from the proposed analytical model match closely the experimental ones.



(a) Virtual Channel Router (b) Path-Sensitive Router

Figure 5.6. Total Power Consumption in Analytical Model and Simulation (8x8 Mesh)

5.3 Communication Reliability

The possible soft faults that could afflict a network architecture can be grouped in two main categories: link errors that occur during the traversal of flits from router to router, and router errors that occur within the router hardware components. Link errors are caused mainly by channel disturbances such as cross-talk, coupling noise and transient faults [135]. They have so far been considered the dominant source of network infrastructure errors and error detecting and correcting codes are being used extensively in on-chip communication links as a form of protection against link errors [136, 115]. Our proposed architecture employs both error detecting (in the form of Cyclic Redundancy Check [CRC]) and error correcting (in the form of Single-Error-Correction-Double-Error-Detection [SECDED]) codes to combat this problem.

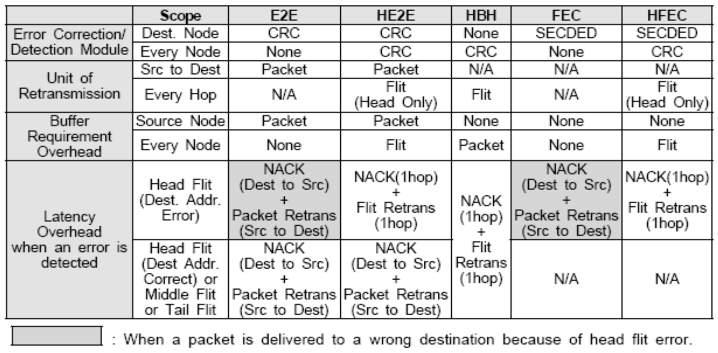
Existing models [137, 115], however, fail to capture the effects of transient errors (e.g. soft errors) occurring within a router. Transient faults are rapidly becoming a force to be reckoned with in the deep sub-micron era. In fact, the susceptibility of circuits to such errors increases exponentially with technology scaling [138]. It is imperative that modern designs effectively account for these events. Soft errors within the router would escape the error detecting/correcting blanket because they do not actually corrupt the data, but, instead, cause erroneous behavior in the functionality of the routing process. Our proposed router architecture and routing algorithm work in concert to protect against a multitude of such faults. To the best of our knowledge, this is the first attempt to account for and protect against router soft faults by utilizing both architectural and algorithmic traits.

5.3.1 Link Errors

To handle link errors, we simulated 5 different retransmission/error correction schemes as shown in Table 5.2. Three of them, End-to-End (E2E), Hop-by-Hop (HBH), and Forward Error Correction (FEC), are widely used techniques in traditional networks, while Header E2E (HE2E) and Header FEC (HFEC) are extensions of E2E and FEC schemes, respectively, adopting hop-by-hop head-flit error checking. In these two schemes, each intermediate router checks the header flits for possible errors and if an error is detected, the correct head flit is retransmitted from the previous node. These schemes are different from HBH in that the head error-checking logic checks only part of the head flit, not the whole flit, and, thus, we can use smaller and faster error-checking logic. The portion of the head flit that needs to be checked is the one which contains the source and destination addresses, thus ensuring that a packet is not misdirected to a wrong destination. Delivery to a wrong destination is a problem that both E2E and FEC schemes suffer from when the destination address is corrupted. In Table 5.2, we summarize the modules used for each scheme, how much data should be retransmitted when an error is detected, and the overhead in terms of both area and latency. Error checking and correction are assumed not to be in the critical path, causing no latency overhead. This is because error detection/correction can be performed in parallel with other router or Network Interface Controller (NIC) operations.

The CRC and SECDED units were implemented in 90nm technology as separate modules, analyzed in terms of dynamic and leakage power consumption and the numbers imported into our simulator. Depending on the retransmission scheme employed (i.e. End-To-End or Hop-By-Hop) the router architecture changes accordingly. The overhead retransmission control logic and buffer size and implementation are markedly different. In ETE schemes, the number of flit buffers required is much larger than HBH, because a router must keep a copy of each flit transmitted for as long as it takes to receive possible NACK (Negative Acknowledgement) from the receiver which could be far away. The buffers, however, can be disabled when not in use to save leakage power. In HBH schemes, the number of flit buffers required is much smaller, but the buffer is cyclic since the number of cycles between acknowledgements from neighboring routers is known precisely. This implies that the buffers cannot be disabled, since they rotate contents every clock cycle. These architectural differences are vital in correctly estimating power consumption in various retransmission schemes. Our hierarchical hardware implementation and analysis of all these components individually allows our model to account for these subtleties, thus substantially improving our projected results.

We also tracked buffer utilization under these schemes to identify the trends in buffer usage. The average buffer utilization at a fixed flit arrival probability of 0.35 was measured. In FEC, HFEC and HBH, only about 10% of a single-flit buffer is utilized on an average, as shown in Figure 5.7 (a). But in E2E and HE2E,

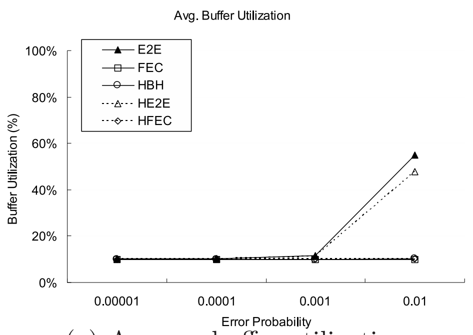


	Scope	E2E	HE2E	HBH	FEC	HFEC
Error Correction/ Detection Module	Dest. Node	CRC	CRC	None	SECDED	SECDED
	Every Node	None	CRC	CRC	None	CRC
Unit of Retransmission	Src to Dest	Packet	Packet	N/A	N/A	N/A
	Every Hop	N/A	Flit (Head Only)	Flit	N/A	Flit (Head Only)
Buffer Requirement Overhead	Source Node	Packet	Packet	None	None	None
	Every Node	None	Flit	Packet	None	Flit
Latency Overhead when an error is detected	Head Flit (Dest. Addr. Error)	NACK (Dest to Src) + Packet Retrans (Src to Dest)	NACK(1hop) + Flit Retrans (1hop)	NACK (1hop) + Flit Retrans (1hop)	NACK (Dest to Src) + Packet Retrans (Src to Dest)	NACK(1hop) + Flit Retrans (1hop)
	Head Flit (Dest Addr. Correct) or Middle Flit or Tail Flit	NACK (Dest to Src) + Packet Retrans (Src to Dest)	NACK (Dest to Src) + Packet Retrans (Src to Dest)		N/A	N/A

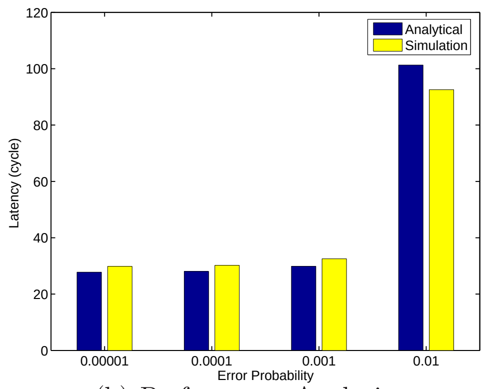
▨ : When a packet is delivered to a wrong destination because of head flit error.

Table 5.2. Retransmission Schemes

buffer utilization increases abruptly as the error probability increases, since the retransmissions from the source to the destination inject additional messages into the network, hence increasing the overall network traffic. The HE2E scheme, being able to fix some of the head flit errors using HBH head flit retransmission, shows less buffer utilization than E2E at higher error probabilities.



(a) Average buffer utilization



(b) Performance Analysis

Figure 5.7. Buffer utilization & Performance Analysis under Error Detection and Retransmission

5.3.2 Modeling End-to-End Retransmission

Our proposed mathematical model can capture any reliability-related parameters to account for both latency and power overhead imposed by any of the five link error control schemes. We demonstrate the utility of our analytical model by analyzing the behavior of an end-to-end (E2E) error detection and retransmission scheme, as an example.

The end-to-end retransmission technique significantly alters the traffic load parameter in our analytical model. Retransmitted packets from error detection at the destination node increase the effective traffic load, which increases contention and reduces buffer availability. These changes can be reflected mathematically. As discussed in section 5.1.1, the zero-error latency (T) (Equation 5.3) is estimated as a function of the flit arrival probability (P_c). Assuming a switch-to-switch error probability, P_{error}, we can determine the end-to-end reliability latency as follows:

$$T_{ete}(P_c) = T(P_c'') + T_{ret}(P_c'')(1 - (1 - P_{error})^H), \tag{5.32}$$

where H is the average number of hops between source and destination, and P_c'' (the modified traffic load) is given by

$$P_c'' = \frac{P_c}{(1 - (1 - P_{error})^H)}. \tag{5.33}$$

T_{ret} includes the round-trip time of message retransmission and the re-send request control signal ($NACK$) delay, where the $NACK$ signal is assumed to be error free for simplicity. Thus, the total latency is given as

$$T_{ret}(P_c'') = T_{ete}(P_c) + T_{NACK}(P_c''). \tag{5.34}$$

The control signal delay (T_{NACK}) is the network latency for one single flit transmission. Using Equation(5.3) in Section 5.1.1, we can compute (T_{NACK}) as

$$\begin{aligned} T_{NACK}(P_c'') &\approx (B(P_c'')W(P_c'') + P + 1)H \\ &\quad + B_{dest}(P_c'')W_{dest}(P_c'') + P - 1. \end{aligned} \tag{5.35}$$

End-to-end error detection and retransmission schemes inflict a significant performance penalty to the network operation under bursty error conditions and high error probabilities. Figure 5.7 (b) compares the results of our analytical model to those of a cycle-accurate simulator. Clearly, results from our mathematical model closely match the experimental results.

5.3.3 Router Logic Errors

Our router implementation (shown in Figure 4.3) consists of six major components, each susceptible to transient faults: the routing unit, the VA, the SA, the crossbar, the retransmission buffers (included in the CRC and SECDED units) and the valid/ready handshaking signals (used between neighboring routers). Following is a detailed description of possible faults and proposed solutions for all major router components. Additionally, the latency and power overhead of our proposed solutions/safeguards are also listed. All the results are summarized in Table 5.3.

Type	Location	Symptom	Solution	Overhead	
				Latency	Power
Head Flit Only	Routing Logic (RT)	- **Message is misdirected to a wrong path:** - This will not cause data corruption since VA, SA performed based on this information. - But the message may be directed to a **blocked path** (either hardware fault [router/link outage] or network edge in various topologies). - Flit misdirected to a **non-blocked path**: - In deterministic routing, it can cause deadlock. - Certain turn schemes prohibited by the routing algorithm - In adaptive routing, not critical (but delay will occur).	- **Current node routing:** Error caught by SA; re-route message. - **Look-ahead routing:** Error reported to the previous router for routing repetition. - Can only be detected in deterministic routing by receiving router. **(Case 1)**	2 cycles / 3 cycles (NACK + routing + retrans.) / 3 cycles (NACK + routing + retrans.)	1 routing stage / Retrans. power. / Retrans. power.
	Virtual Channel Allocator (VA)	- **Can assign invalid VC:** - For 3VCs, we need 2bits for VCID. If MSB is toggled, it will change 1 (01) to 3 (11) which is not available. - Thus, message can be lost. - **Can assign unavailable VC (reserved or full or empty)** - If reserved: two messages will be mixed. - If full: it will overwrite channel buffer. - If empty: it will be overwritten later on	- Use Hamming code (parity checks) to correct single-bit errors in virtual channel ID. **(Case 2)**	None (Masked within stage 1 of the router operation)	Additional power from additional bits (minimal), and Hamming code parity checks (also minimal, one XOR gate per check bit)
All Flit Types	MUX/ DEMUX	- VC DEMUX (1:N)*: Same problem as VA above. *(input : output) - VC MUX (N:1): Same problem as SA below.	- See each section.	▨	▨
	Switch Arbiter (SA)	- **No flits are sent to the crossbar (N:0):** - This would cause the loss of all the flits moved out of the FIFO buffers and into the pipeline buffers in stage 1 of the arbitration.	- Send flits moved out of the FIFO buffers to both the pipeline latches and the retrans. buffers. **(Case 3a)**	2 cycles (Error signal + retrans.)	Retrans. power.
		- **Non-head flits might be directed to a path different from the header flit.**	- Append Message ID to all data flits. **(Case 3b)**	2 cycles (NACK + retrans.)	MsgID check logic + retrans. power.
		- **Can direct several flits to the same output port (N:1)**	- Corrupt flit detected by error detection unit. Retransmit all related flits. **(Case 3c)**	2 cycles (NACK + retrans.)	Retrans. power.
		- **A flit can be sent to several output ports (1:N multicast)**	- Utilize existing routing table entries. **(Case 3d)**	2 cycles (NACK + retrans.)	Retrans. power.
	Retrans. Buffer	- **Endless retransmission loop since original data itself is corrupt.**	- Buffer duplication **(Case 4)**	None	Additional buffer power.
	Crossbar (XB)	- Single-bit upsets to traversing flits.	- Error Detection / Correction **(Case 5)**	▨	▨
	Link(LK)	- Traditional transient fault case	- Error Detection / Correction	▨	▨
	Hand-shaking signals	- Could upset the operation of the network through erroneous Valid/Ready control signals.	- Triple Module Redundancy (TMR) **(Case 6)**	None	Power for additional redundant lines, but negligible.

Table 5.3. Logic Errors

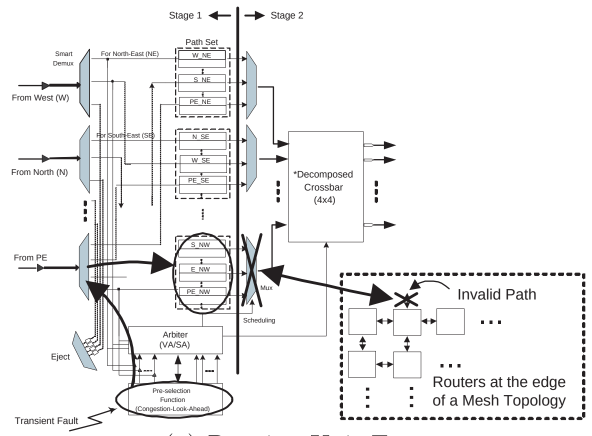
Case 1 - Routing Unit Protection: A transient fault in the routing unit logic could cause a flit to be misdirected. This will not cause any data corruption, since the subsequent virtual channel allocation and switch arbitration would be performed based on the misdirection. The erroneous direction, however, may be blocked, either because of a link outage, or a network edge in various topologies (see Figure 5.8 (a)). The proposed solution to such errors depends on the routing algorithm: in current-node routing schemes (i.e. the routing decision for the current router is taken at the current router), the misdirection will be caught by the switch arbiter which can inform the routing unit to repeat the routing procedure. This will incur a two-cycle delay. In look-ahead routing (i.e. the routing decision for the current router was taken at the previous router), the error will also be caught by the switch arbiter and reported to the previous router through a NACK, thus incurring a 3 cycle delay.

Misdirection to a functional path, however, will not be caught by the switch arbiter. It could potentially cause deadlock in deterministic routing algorithms. In such algorithms, however, the error can be detected in the router that receives the misdirected flit. A NACK to the sending router would then fix the problem within 3 clock cycles (NACK + re-routing + retransmission). In adaptive routing schemes the error cannot be detected. However, in such schemes a misdirection fault is not fatal; it would merely delay the flit traversal.

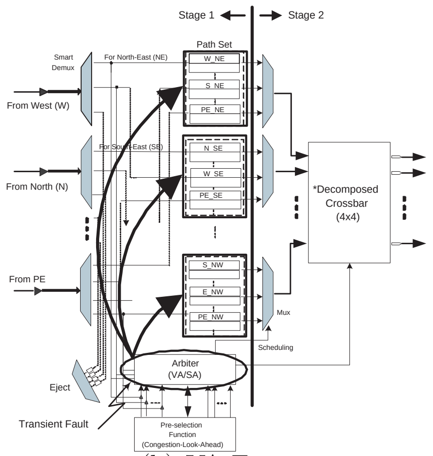
Figure 5.8(a) illustrates an example effect of a soft error in the routing unit. As explained above, such an error could direct a flit to a blocked path. For example, a router at the top edge of a mesh topology has a blocked output link to the north. A transient fault in the routing unit could direct a flit coming from the local processing element (or any input link) to be placed in the blocked path to the north. This error will be caught by the switch arbiter which knows that the path is blocked.

Case 2 - Virtual Channel Allocator (VA) Protection: A soft error in the VA unit could lead to an invalid virtual channel flit assignment. A bit flip in the virtual channel ID could give rise to either a non-existent virtual channel or a different existing one. This channel could be reserved or full. In the former case, two different messages would be mixed, and in the latter case an existing flit would be overwritten. Both problems would lead to flit/packet loss. If the erroneous virtual channel is unreserved and empty, then the flit will eventually be overwritten when a new, correct packet is assigned to that channel. The proposed solution involves the use of a Hamming code within the virtual channel ID to correct all single-bit errors. The overhead for such a code is minimal because of the small length of the virtual channel ID. In our proposed architecture there are 3 virtual channels per physical path set, thereby requiring a 2-bit virtual channel ID. For a 2, 3, or 4-bit word, a single-error-correcting Hamming code requires 3 check bits. A 3-bit overhead is inconsequential compared to the protection and flexibility that such a code offers; with those 3 check bits, the virtual channels per physical path can be increased to any number up to 16 without any impact on the virtual channel ID protection. The area and power overhead is also minimal, since the code requires only one XOR gate per check bit. There is no latency incurred since the Hamming check is masked within stage 1 of the router operation (along with routing and virtual channel and switch arbitrations).

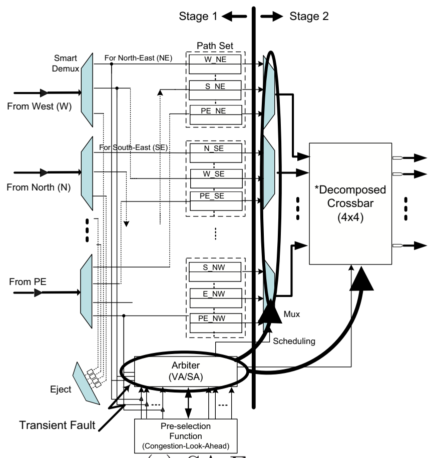
In the block diagram of our proposed architecture shown in Figure 5.8 (b), the VA unit is combined with the switch arbiter; however, their operations are distinct. Just like the routing process, the virtual channel allocation is done entirely in stage 1 of the router operation. Thus, a fault would immediately cause an erroneous result, i.e. an invalid virtual channel assignment within a path set, as indicated by the arrows (see Figure 5.8 (b)).



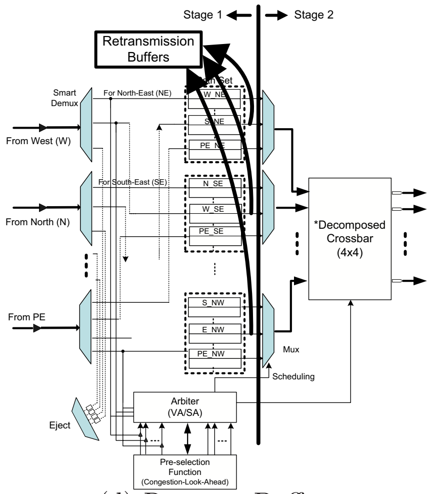
(a) Routing Unit Error



(b) VA Error



(c) SA Error



(d) Retrans. Buffers

Figure 5.8. Routing Unit, VA and SA Errors

Case 3 - Switch Arbiter (SA) Protection: A transient fault in the switch arbiter could give rise to more complex errors than the previous cases because the control signals span both stages of the router operation (see large arrows in Figure 5.8 (c)). The control and scheduling signals for the crossbar traversal are generated during stage 1, but used in stage 2. Therefore, the switch arbiter instructs the FIFO virtual channel buffers to shift one position in stage 1, and the flits latched in the pipeline buffers are sent to the crossbar in stage 2. Thus, a soft error in the SA could give rise to several packet-loss problems:

(a) A soft error in the control signals might prevent the flits from traversing the crossbar in stage 2. This would cause the loss of all the flits moved out of the FIFO buffers and into the pipeline buffers in stage 1 of the arbitration. To avoid this fatal situation, the retransmission buffer and an additional flag bit could be used for recovery. The flits moved out of the FIFO buffers should be sent to both the pipeline latches and the retransmission buffers (see Figure 5.8 (d)). The retransmission buffers in our proposed architecture are large enough to keep four consecutive flits sent out on each output link. Additionally, whenever valid data is moved out of the FIFO buffers and into the pipeline latches a ”Data Valid” bit is set. This bit is subsequently ANDed with the control signal of the switch arbiter in stage 2. If the control signal was erroneously reset to a zero state, i.e. no crossbar traversal for any flit, then the mistake will be caught. A signal is sent to the switch arbiter which can retransmit the flits from the retransmission buffer. The additional latency is two clock cycles (Error signal + retransmission).

(b) If a data flit is mistakenly sent to a direction different from the header flit, it would cause flit/packet loss. We propose using a very compact header in all data flits which would include a message ID. Additionally, each router would have a simple message ID checker unit. If the message ID check reveals that the incoming flit ID is not in the router's routing table, then a NACK is sent to the sending router which would retransmit the flit to the correct router from its retransmission buffer. The incurred overhead comes from the message ID logic which is very simple and compact in area. The header overhead in the data flits is also minimal since only the message ID is stored. The additional latency is two clock cycles (NACK + retransmission) on an error.

(c) The error could cause the arbiter to direct two flits to the same output.

This will lead to a corrupt flit which will be detected by the error detection code in the next router. A NACK will be sent and the correct flits retransmitted from the retransmission buffer. This error recovery process will incur two cycles (NACK + retransmission) latency overhead.

(d) The error could cause the arbiter to send a flit to multiple outputs (multicasting). If the flit is a data flit, the error will be taken care of by case 3(b) above at the next router. If, however, the flit is a header flit then one needs to consider two possibilities:

(i) If there is no existing message in the erroneous path, then the wrong header flit will be considered as the beginning of a new packet at the next router. Thus, a virtual channel will mistakenly be reserved. The way to tackle this is to utilize the existing routing table entries. When a correct header comes in on the same path later on, the router will detect that the path is already reserved through its InputVC-OutputVC pair in its routing table. It can, therefore, request a retransmission of the last header flit. If it receives the same flit again it will realize that the channel was erroneously reserved and release the path to the correct header flit. (ii) If a correct header flit has already used the current path, then the next router will request a retransmit, as in case 3(d)(i) above. The previous router will not resend the wrong header flit to the same router again, so the receiving router can discard the erroneous header flit from its buffer without disrupting the correctly reserved path.

The incurred overhead in both cases is a single flit retransmission from the previous router, i.e. a two-cycle latency and minimal power overhead.

Case 4 - Retransmission Buffer Protection: A soft error in the retransmission buffer would yield an endless retransmission loop since the original data itself is now corrupt. The way to avoid this incidence is to use duplicate retransmission buffers. This will double the buffer area overhead and power. However, the number of retransmission flit buffers in on-chip routers is very small (four per output link in the case of our proposed two-stage router), so doubling the area and power of this component will not be prohibitive.

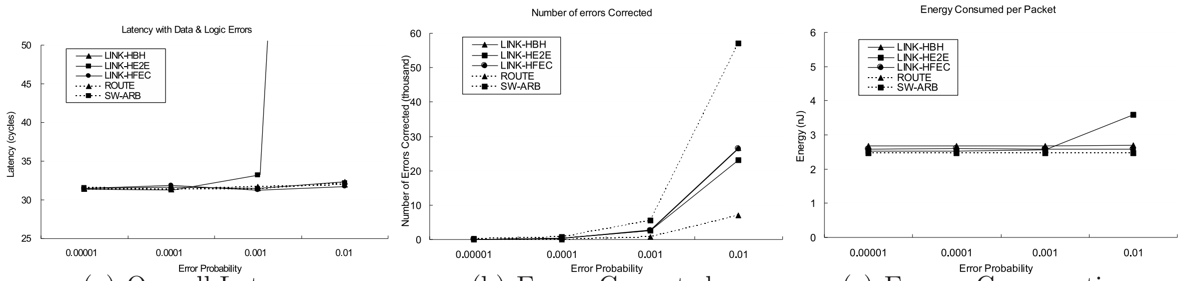
Case 5 - Crossbar Protection: A transient fault within the crossbar would produce single-bit upsets, not entire flits being misdirected as in the switch arbiter case. Single-bit upsets are taken care of by the error detection and correction unit employed within each router, thus eliminating the problem.

Case 6 - Valid/Ready Handshaking Signal Protection: Every router has several valid/ready handshaking signal lines with neighboring routers to facilitate proper functionality and synchronization. For example, each output link has a ”Data Valid” output line and a ”Data Ready” input line to/from the adjacent router. Transient faults on these lines could severely hamper the operation of the network. Therefore, Triple Module Redundancy is proposed by which three lines and voting are used, instead of one, to ensure protection against soft errors. There is an area and power overhead increase, but the area occupied by these lines is negligible compared to the area of the other router components.

5.3.4 Simulation Results

The experimental setup described in section 2 was used. Single-bit errors were uniformly injected in the network, both as link errors and as logic errors within the router architecture. The definition of error probability is slightly different depending on the error model. For link errors, it is defined as the probability of flit error during link traversal. For routing and switch arbitration logic errors, it is defined as the probability that a flit is mishandled by the logic as a result of single-event upsets.

Figure 5.9 (a) shows the overall latency of each scheme for a wide range of error probabilities. In all schemes, except HE2E, the latency overhead incurred on the overall latency as a result of error detection/correction was minimal, because all the schemes typically require only 1-3 cycles, as shown in Tables 5.2 and 5.3. However, in the HE2E scheme, if a message has errors in the flit portion where error checking is not performed at intermediate routers, then the whole packet should be retransmitted, and these additional flits significantly increase the overall message injection rate of the network, and increase the average message latency. The proposed architectural improvements within the router components (i.e. routing logic (ROUTE) and switch arbiter logic (SW-ARB) curves) inflict no significant latency increase, as predicted. Figure 5.9 (b) shows the number of detected and corrected errors under each error protection scheme. Without error detection/correction, the message will either be corrupted or lost, and, thus, no errors will be corrected at all. By using these schemes, errors can be corrected



(a) Overall Latency (b) Errors Corrected (c) Energy Consumption

Figure 5.9. Simulation Results

either through retransmission or error-correction. Since errors in the routing logic only affect head flits, our routing logic protection scheme has the least number of errors detected/corrected. Both switch arbiter errors and link errors can corrupt all flits. Since most flits go through switch arbitration several times at each node as network congestion increases, while they only traverse the link once, errors in the switch arbiter are more frequently detected/corrected than link errors. This trend justifies the inclusion of our proposed router-error safeguards in on-chip network architectures, since, as mentioned before, soft error faults within the routers will continue to increase exponentially as technology scales down. As shown in Figure 5.9(b), as error probabilities increase the number of errors detected and corrected by our switch arbiter reliability measures rises abruptly. All those errors would otherwise escape and cause severe packet/flit losses. Figure 5.9 (c) shows the energy consumption per packet. In the case of HE2E, source to destination retransmission incurs significant energy consumption, while the other schemes have minimal energy overhead, since either they are not involved in retransmission, or are involved in hop-by-hop retransmission which has minimal power overhead. Our proposed router-logic protection measures do not cause any significant increase in energy consumption.

Even though our reliability measures provide a blanket of protection against both link and router logic errors, the protection is limited to single-bit errors. Temporal multi-bit errors (i.e. two independent transient faults affecting the same flit at different times), and spatial multi-bit errors (i.e. one fault causing two different errors at different locations) are not tackled. However, the probability of such events is still considered extremely low.

5.4 Conclusion

The rapidly increasing use of SoC architectures has accentuated the need for efficient on-chip communication infrastructures. The resource-constrained nature of this system needs precise and efficient analysis of their performance, fault-tolerance and energy behaviors. To the best of my knowledge, we propose the first queuing-theory-based analytical model for a 2D mesh network, which performs latency and power analysis at the granularity of individual router sub-modules for increased accuracy. Simulation results validate the correctness and accuracy of the model for a conventional router. The model is then used to demonstrate it's effectiveness in analyzing a novel path-sensitive router architecture that can minimize average packet latency by intelligent path selection and reduced switching activities. Furthermore, the analytic model is used in quantifying the overall power consumption by capturing the utilization of different components and their corresponding energy consumption.

The study then focuses on the fault-tolerance aspects of on-chip interconnects by analyzing two types of faults: link errors and router logic errors. We explore five types of retransmission techniques to combat link errors and conclude that by providing a separate error coding technique for the header flits, the packet misrouting probability is significantly reduced, thereby providing better fault-tolerance. We then extend our analytical model to evaluate the network under the End-to-End (E2E) protection scheme to demonstrate the utility of the model in analyzing all three parameters: performance, energy and fault-tolerance. We additionally propose a series of transient fault protection techniques to tackle soft errors within the router's individual components. Our safeguards are shown to incur no significant area, latency, or power overhead to the network.

Chapter 6

A Fine-Grained Modular Router Design

In the previous chapter, we looked at how to build up analytical performance/power model for Network-on-Chip (NoC) architectures. This chapter proposes a novel fine-grained modular router architecture [74], towards the comprehensive approach of designing low-latency, energy-efficient and reliable on-chip communication networks. The proposed architecture employs decoupled parallel arbiters and uses smaller crossbars for row and column connections to reduce output port contention probabilities as compared to existing designs. Furthermore, the router employs a new switch allocation technique known as "Mirroring Effect" to reduce arbitration depth and increase concurrency. In addition, the modular design permits graceful degradation of the network in the event of permanent faults and also helps to reduce the dynamic power consumption.

6.1 Introduction

In this chapter, we present the design of a modular wormhole-switched router architecture considering the performance, energy, and fault-tolerance issues in a cohesive manner. The salient features of the proposed router that makes it distinct compared to other contemporary designs are the following: (i) For enhancing performance, we cleverly exploit the virtual channel allocation (VA) and switch allocation (SA) units in minimizing the delay due to resource contention. (ii) For energy conservation, we focus on use of smaller crossbars, simplified arbiter circuits, and other design tricks such as early ejection and mirrored arbitration. (iii) For enhancing fault-tolerance, we rely on a modular design such that failure of a router component can be tolerated by allowing the switch to operate in a degraded mode. In this context, we propose several techniques to handle hardware faults in different components of the router.

The proposed Row-Column (RoCo) Decoupled Router enhances performance by reducing the contention probability. This is achieved be splitting the router operation into two distinct and independent modules. The modules are each responsible for handling traffic in a single dimension (X-dimension and Y-dimension). This decoupling permits the use of smaller and simpler components with reduced logic depth. Each module requires a compact 2x2 crossbar, as opposed to the bigger (5x5) monolithic crossbar used in conventional architectures. Furthermore, the proposed router uses a novel switch arbitration scheme, known as the Mirroring Effect. This mechanism requires fewer global arbiters and maximizes crossbar utilization by providing optimal matching between inputs and outputs. Contention in the crossbar is further reduced through the use of a preliminary path-sensitive buffering process, known as Guided Flit Queuing. Finally, the proposed architecture exploits the look-ahead information in arriving packets to eject flits destined to the local processing element (PE), thus, eliminating unnecessary passage through the stages of the local router (Early Ejection).

The RoCo router possesses inherent fault-tolerant attributes. Its decoupled operation allows for partial functionality in the event of a hard failure. Having two operationally independent modules implies that one module can continue to provide service in one dimension even if the other module is blocked due to a permanent failure. This alleviates contention around the faulty node, which has a profound effect on network latency. Additionally, the proposed architecture employs a Hardware Recycling Mechanism, which uses resource sharing to circumvent hard failures in various intra-router components. We present a comprehensive router fault model, and propose several safeguards to our RoCo router architecture to protect against several types of intra-router hard faults. These measures induce minimal area, power and latency overheads. The novelty in this approach is that operation of the router can continue in the presence of a faulty component through resource sharing.

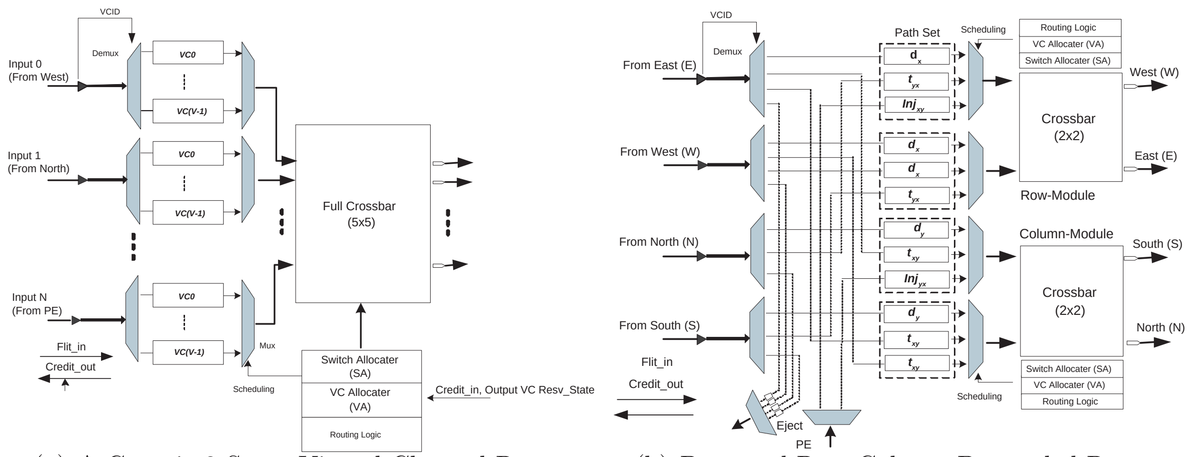
This chapter is organized as follows. First, we present the proposed router architecture in Section 6.2, the fault-tolerant analysis in Section 6.3 and evaluation results in Section 6.4. Finally, concluding remarks are made in Section 6.5.

6.2 The Proposed Row-Column Decoupled Router

Figure 6.1 (a) illustrates the architecture of a generic 5-port 2-stage NoC router employing virtual channel flow control and wormhole switching [22]. The five ports correspond to the four cardinal directions and the connection to the local Processing Element (PE). The router consists of six major components: the Routing Computation unit (RC), the Virtual Channel Allocator (VA), the Switch Allocator (SA), the virtual channel buffers, and the crossbar. It employs a pipelined design with speculative path selection to improve performance. Instead of relying on a unified architecture with a monolithic crossbar, the proposed router consists of dual compact crossbars arranged in Row and Column Path Sets. The novelty of the proposed architecture is that it can support deterministic, XY-YX and adaptive routing in a 2-D mesh network under fault-free and faulty environments. It will be shown that this dual-set architecture is very efficient in accurately capturing temporal traffic fluctuations through accurate speculative path selection.

6.2.1 Row-Column Switch

This section introduces the Row-Column Decoupled (RoCo) Router, which is partitioned into two small identical switching fabrics. Figure 6.1 (b) depicts the major components of the new two-stage, pipelined router architecture. The first stage is responsible for look-ahead routing, virtual channel allocation (VA) and speculative switch allocation (SA); all three operations are performed in parallel. The second stage is responsible for crossbar traversal. The functionality of the router is described in this work with respect to a 2D mesh interconnect. The router has five inputs, marked for convenience as inputs from the four directions (North, East, South and West) and one from the local PE. It has two sets of crossbars, called Row-Module (East-West) and Column-Module (North-South). The router is divided into two distinct, independent modules, each responsible for possible

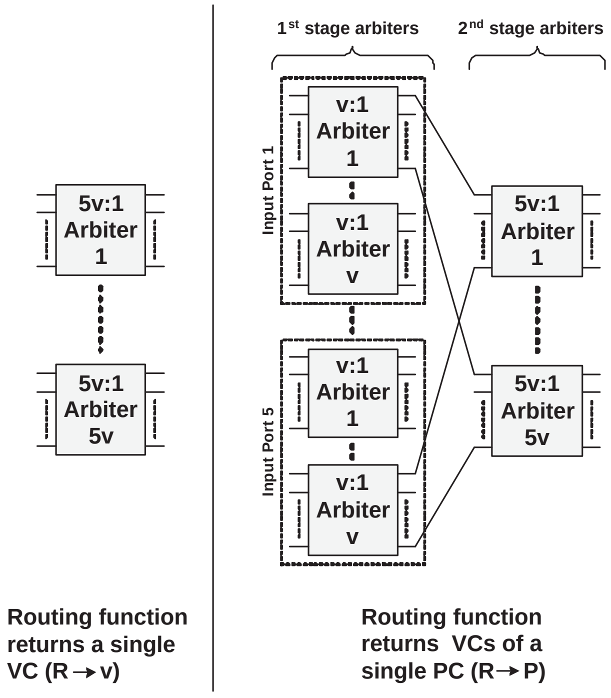


(a) A Generic 2-Stage Virtual Channel Router (b) Proposed Row-Column Decoupled Router

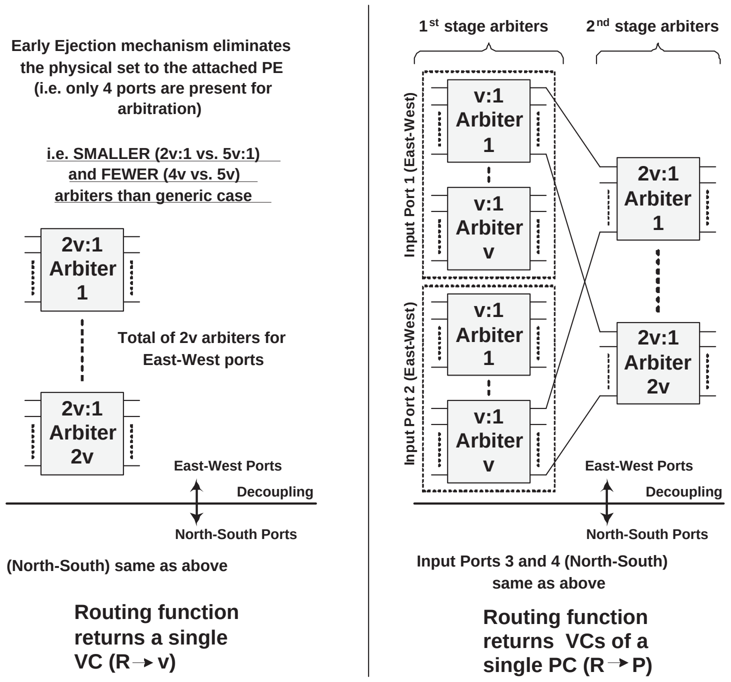
Figure 6.1. On-chip Router Architectures

traversal in the corresponding crossbar connections; i.e. in the East-West direction, or the North-South direction. Each port of the crossbar module has a set of three Virtual Channels (VCs) to hold arriving flits from neighboring routers or the local PE. These sets are aptly named Path Sets, since all flits within such a set travel in the same physical direction. In order for an incoming header flit to pass through the DEMUX and be placed into the buffer corresponding to its output path, the header flit should know its route before departing the previous node. To remove routing from the router's critical path, the Routing Computation (RC) can be performed one step ahead. By employing this Look-Ahead Routing scheme [139], the flit is guided to the appropriate buffer by the DEMUX.

Based on the required output port, the header flit requests a valid output VC. The virtual allocation unit, VA, arbitrates between all packets requesting access to the same VCs and decides on winners. Figure 6.2 compares the complexity of the VA unit of a generic 5-port (North, East, South, West, PE) router and the proposed RoCo router. The use of Early Ejection allows the RoCo router to eliminate the PE path set. (Early Ejection is analyzed later on in this sub-section.) In this comparison, we assume v VCs per input port for both the generic and RoCo architectures. Figure 6.2 compares two cases: one, where the routing function returns a single virtual channel ($R => v$), and one, where the routing function returns a single physical channel ($R => p$). Clearly, the RoCo router requires fewer and smaller arbiters in both cases. This attribute significantly reduces the



(a) The VA in a Generic 5-port NoC Router [130]


Early Ejection mechanism eliminates the physical set to the attached PE (i.e. only 4 ports are present for arbitration)
i.e. SMALLER (2v:1 vs. 5v:1) and FEWER (4v vs. 5v) arbiters than generic case
2v:1 Arbiter 1
2v:1 Arbiter 2v
Total of 2v arbiters for East-West ports
East-West Ports
Decoupling
North-South Ports
(North-South) same as above
Routing function returns a single VC (R→v)
1st stage arbiters
2nd stage arbiters
Input Port 1 (East-West)
Input Port 2 (East-West)
v:1 Arbiter 1
v:1 Arbiter v
Input Ports 3 and 4 (North-South) same as above
Routing function returns VCs of a single PC (R→P)

(b) The VA in the proposed RoCo Decoupled Router

Figure 6.2. Virtual Channel Allocator (VA) Comparison

complexity of the arbitration process, since smaller and fewer arbiters imply less contention and reduced arbitration depth. On the contrary, increased complexity causes the VA in a generic architecture to require multiple iterative arbitrations before satisfying all pending requests [22].

The RoCo router partitions the operation of the VA in two smaller independent modules, thus substantially reducing size and complexity.

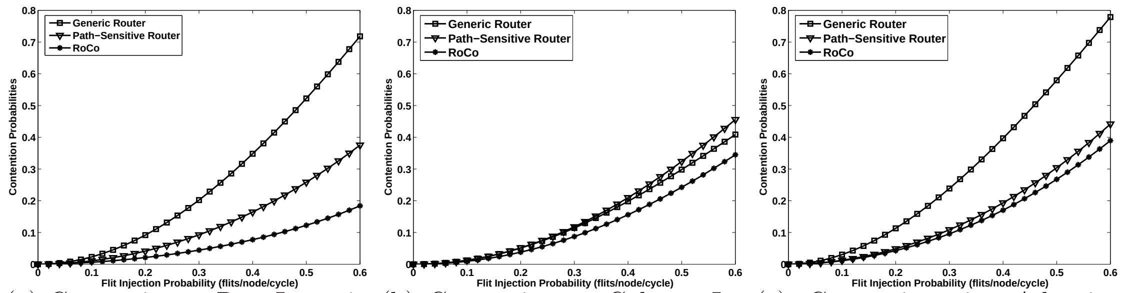
In the proposed architecture, look-ahead routing decides the valid outgoing channels of packets based on their output paths (Row-Module or Column-Module) and on whether or not they are continuing along the same dimension.

Input Port	Row-Module		Column-Module	
	Port 1	Port 2	Port 1	Port 2
Adaptive Routing	$d_x\ t_{yx}\ Inj_{xy}$	$d_x\ d_x\ t_{yx}$	$d_y\ t_{xy}\ Inj_{yx}$	$d_y\ t_{xy}\ t_{xy}$
XY-YX Routing	$d_x\ t_{yx}\ Inj_{xy}$	$d_x\ d_x\ t_{yx}$	$d_y\ t_{xy}\ Inj_{yx}$	$d_y\ d_y\ t_{xy}$
XY Routing	$d_x\ d_x\ Inj_{xy}$	$d_x\ d_x\ Inj_{xy}$	$d_y\ t_{xy}\ Inj_{yx}$	$d_y\ d_y\ t_{xy}$

Table 6.1. VC Buffer Configuration for the three supported routing algorithms

In Figure 6.1 (b), VCs marked d_x (d_y) hold flits which continue traversal in their current X (Y) dimension, i.e. East or West (North or South). VCs marked t_{xy} (t_{yx}) hold flits which switch from the X to the Y dimension (Y to X). For example, a flit traversing the network from the east toward the north or the south will arrive at the t_{xy} VC of the first input port in the Column-Module. If the flit is to continue traversal to the west, it is buffered in the d_x VC. That is, d_x and d_y VCs are used for on-going flits along the same dimension, while t_{xy} and t_{yx} are used for changing from the Row-Module to the Column-Module and the other way around. A flit coming from a local PE which is first addressed to the X-dimension, such as the east or the west outputs, is buffered in the Inj_{xy} VC of the Row-Module, while a flit first addressed to the Y-dimension is queued into the Inj_{yx} VC of the Column-Module. Depending on the type of routing algorithm used in the network, the number and configuration of the VC buffers changes accordingly. A deadlock free deterministic routing algorithm, such as the XY routing, requires a minimum of 8 VCs for correct functionality (2 d_x, 2 d_y, 2 t_{xy}, 1 Inj_{xy} and 1 Inj_{yx} for source-destination pairs which lie in the same column). To provide support for deadlock-free XY-YX routing, two additional d_x VCs are required. Finally, to provide support for deadlock-free adaptive routing, two more t_{xy} VCs are needed, giving a total number of 12 VCs, as shown in Figure 6.1 (b).

These VCs are grouped into 4 path sets, each containing 3 VCs. When the router is used with deterministic or XY-YX routing (which can operate with less than 12 VCs), the extra VCs are re-assigned in such a way as to improve performance by reducing Head-Of-Line blocking (HOL). For example, XY routing gives rise to asymmetric utilization of the router; HOL in the X-dimension happens more frequently than in the Y-dimension, and the injection channel Inj_{xy} is much more frequently used than Inj_{yx}, as a result of the routing scheme. This behavior was validated in our simulations, and is described in Section 6.2.2 and shown in Figure 6.3. Moreover, since flits are ejected after traversing the Y-dimension in XY routing, the traffic in the Row-Module of our architecture would be higher than the Column-Module assuming uniform communication traffic. To account for this unbalanced traffic distribution, two additional d_x VCs are assigned to the extra buffers available in the router. Similarly, all 12 VCs present in the router are assigned differently, according to the routing algorithm used. The VC buffer configurations for the three supported routing algorithm types are summarized in Table 6.1.



(a) Contention at Row Input in XY Routing (b) Contention at Column Input in XY Routing (c) Contention in Adaptive Routing

Figure 6.3. Contention Probabilities

Upon successful VC allocation and provided buffer space is available in the downstream router, a flit requests access to the crossbar by undergoing Switch Arbitration (SA). The SA arbitrates between all VCs requesting access to the crossbar and grants permission to the winning flits. The latter are then able to traverse the crossbar and are forwarded to the respective output links. Switch arbitration works in two stages; stage 1 requires a v-input arbiter for each input port (since there are v VCs per port). Stage 2 arbitrates between the winners from each input port and requires P P-input arbiters, where P is the number of physical ports. In order to improve the crossbar switching speed, the SA should use a simple algorithm and reduced logic depth. To that extend, the proposed architecture splits the SA module in two smaller modules, each responsible for a small 2x2 crossbar. The reduced number of crossbar ports reduces the complexity of the SA modules, which function independently from each other. Operation of the SA modules is described in detail in Section 6.2.3.

Deadlock Freedom: Deadlock is avoided by eliminating cyclic dependencies in the network fabric through the use of disjoint virtual channels used by flits when changing dimension (i.e. making a turn). Adding extra VCs is a technique commonly used to provide deadlock freedom in adaptive routing [10, 9]. A generic 5-port router requires at least 10 VCs (5x2) to ensure deadlock freedom in adaptive routing in a 2D mesh. As previously mentioned, the proposed RoCo router requires a minimum of 12 VCs, as shown in Figure 6.1 (b), to support deadlock-free deterministic, XY-YX and adaptive routing. The two d_x VCs in the second path set of the Row-Module provide a deadlock free path in the East-West direction during a potential deadlock. The location of the two VCs need not be in the second path set. Interchanging the locations of the VCs in the two path sets of the Row-Module would still yield the same effect. The two t_{xy} VCs in the second path set of the Column-Module are used to ensure deadlock-free routing in case of a chained cyclic dependency. The first t_{xy} VC of the Column-Module is used for turning from the east to the south direction, and the second t_{xy} VC is used for turning from the east to the north direction. Once again, the location of these VCs may be interchanged between the two path sets of the Column-Module with the same effect.

Early Ejection: A flit destined for the local PE, does not traverse the crossbar, but, instead, it is ejected immediately upon arrival, as represented in Chapter 4 [80]. This mechanism utilizes the look-ahead routing information to detect if the incoming flit is destined for the local PE and accordingly ejects it after the DEMUX. The previously proposed Path-Sensitive Router also uses this technique. This early ejection saves two cycles at the destination node by avoiding switch allocation and switch traversal. Also, it reduces the input load for each crossbar input port. This provides a significant advantage in terms of nearest-neighbor traffic, and can take advantage of NoC mapping, which places frequently communicating PEs close to each other [131]. Most research in the NoC realm so far has assumed uniform spatial distribution of traffic patterns for evaluating parallel systems. However, this is not very realistic in SoC environments, because different functions will be mapped to different parts of the SoC and thus, the traffic may exhibit highly localized patterns [131].

Modular Router and Guided Flit Queuing: The fact that this router architecture uses topology-tailored routing, it allows for the use of two compact 2x2 decentralized crossbars. This fine-grained modular organization provides two distinct modular entities, each operated independently and in parallel. This leads to high-speed switching, less circuit complexity and smaller logic depth in the arbiter and control blocks. In addition, the smaller crossbars enjoy a reduced power consumption as compared to a big unified crossbar, because of their smaller number of connections. At the same time, wiring resources are more efficiently utilized because of the increased parallelism afforded by the smaller crossbars. In our proposed architecture, the input decoders (DEMUXes) undertake a more significant role than in a generic router. In the latter, the input decoders can only distribute incoming flits into the VC buffers of a single port set (see Figure 6.1 (a)). In the RoCo architecture, however, the input decoders can distribute flits to multiple path sets. This mechanism amounts to a preliminary switching operation, which we call “Guided Flit Queuing”, and significantly alleviates contention later on in the crossbar by pre-arranging incoming flits according to their desired output path dimension (X or Y).

6.2.2 Blocking Delay

Network latency consists of actual transfer time and blocking delay. The blocking delay is heavily influenced by the switch allocation strategy and the traffic pattern at run time, while the actual transfer time is determined by the floor-plan and the topology of the network at design time. Given that transfer time is defined by the physical design, we address the other component of network latency, i.e. blocking delay due to contention. Contention is a result of the two arbitration processes occurring within the router: virtual channel allocation and crossbar passage (input port service scheduling and output port allocation).

Figure 6.3 shows the comparison of the input contention probabilities in three different architectures (Generic, Path-Sensitive and RoCo) in an 8x8 network with uniform traffic pattern. In Dimension-Order Routing (XY routing), the flits of the row input are involved in more severe output conflicts than the column input, because of the nature of the routing algorithm (i.e. X first, Y next). Thus, contention at the row input is higher than in the column input as shown in Figures 6.3 (a) and (b). Adaptive routing is useful for avoiding local congestion, but it does not reduce the contention probability unless efficient allocation techniques are employed. In fact, adaptive routing may have poor performance with uniform traffic, as explained in [9]. It is evident in Figure 6.3 that the generic router suffers from high contention probability, which inevitably leads to high Head-Of-Line (HOL) blocking. The RoCo router has the least contention probability. Furthermore, the RoCo router significantly outperforms the other two architectures in terms of non-blocking probability (i.e. when each output port has one input connection; we call this maximal matching between input and output ports). The non-blocking probabilities for three router architectures are shown in Table 6.2. Assuming that each input flit has an equal probability $1/(N-1)$ of accessing one of the $(N-1)$ output ports in an $N \times N$ crossbar, the number of cases in which non-blocking maximal matching, $F(N)$, occurs is computed as

$$F(N) = N! - \sum_{j=1}^{N} \begin{pmatrix} N \\ j \end{pmatrix} F(N-j), \ where\ N >= 3,\ F(1) = 0\ and\ F(2) = 1. \qquad (6.1)$$

In the Path-Sensitive router proposed earlier [80], arriving flits are grouped in sets depending on their destination quadrant (North-East, North-West, etc.). In that architecture, two inputs from each quadrant path set request one output port. For example, flits in the two quadrants NE and NW may compete for the north output channel. In a similar fashion, two input ports also compete for one output in the RoCo router. However, RoCo uses parallel and independent crossbars, while the Path-Sensitive router has chained dependence between requests. Thus, only 2 cases out of 2^4 matches are non-blocking in the Path-Sensitive Router, while 2 cases out of 2^2 are non-blocking in each module of the RoCo router, where the operation of two modules is independent of each other. The RoCo router is almost six times more likely to achieve maximal matching than a generic router (25% to 4.3%), and two times more likely than the Path-Sensitive router (25% to 12.5%). This implies that the RoCo design is more likely to provide non-blocking connections.

Router Designs	Generic	Path-Sensitive	RoCo
Non-Blocking	$0.043 = \left(\frac{F(N)}{(N-1)^N}\right)$, $N = 5$	$0.125 = \left(\frac{2}{2^4}\right)$	$0.25 = (1 - 0.5)(1 - 0.5)$

Table 6.2. Non-Blocking Probabilities in three different Router Architectures

6.2.3 Concurrency Control for High-Contention Environments

Maximal matches become a more realistic target for a high-speed on-chip router for resolving HOL blocking. However, maximal matches can also be quite expensive. In this section, we consider maximal matching and concurrency control techniques that are applicable to high-contention environments. Further, we introduce the "Mirroring Effect", a new switching allocation scheme which provides maximal matching in the RoCo router. The Mirroring Effect is a simple algorithm that finds the maximum number of matches between inputs and outputs, customized to the small 2x2 crossbar of each module (Row-Module and Column-Module). The two sets of disjoint pair-wise switch allocations are illustrated in Figure 6.4. The algorithm is based on the rationale that maximal matching is achieved only when the switch allocation results of the two input ports of a single module are mirror images of each other. This realization allows the RoCo implementation to perform global arbitration in only one of the two input ports of each module, and the result is mirrored in the other port. This is illustrated on the right-hand side of Figure 6.4. For example, if a flit in the top input port is to be forwarded to the West direction, then the bottom port should forward a flit to the East direction to ensure full utilization of the crossbar. Hence, the bottom input port grants access to a flit which wants to continue traversal in the East direction. This scheme constitutes a simple and concurrent global arbitration mechanism, as compared to complex hierarchical arbitrations and Parallel Iterative Matching (PIM) [9].

The proposed mechanism requires two arbiters per input port for the first (local) stage of arbitration, as opposed to just one in the generic case. The two arbiters are required to ensure maximal matching by providing the winning requests

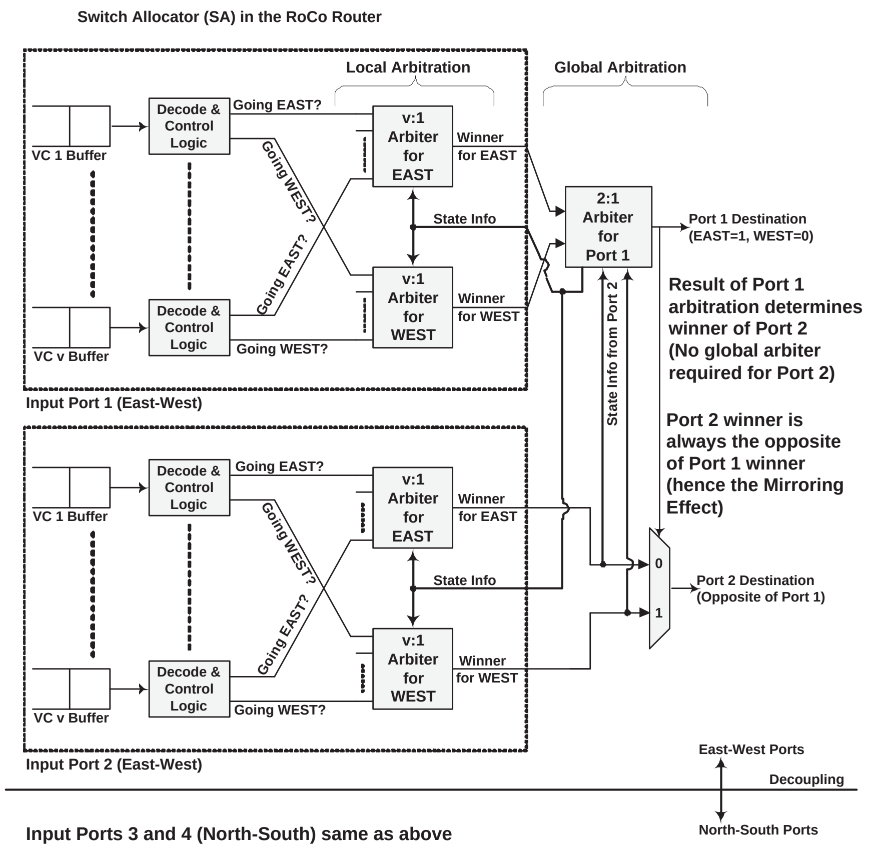

Switch Allocator (SA) in the RoCo Router
Local Arbitration
Global Arbitration
VC 1 Buffer
VC v Buffer
Decode & Control Logic
Going EAST?
Going WEST?
v:1 Arbiter for EAST
v:1 Arbiter for WEST
Winner for EAST
Winner for WEST
State Info
2:1 Arbiter for Port 1
Port 1 Destination (EAST=1, WEST=0)
Result of Port 1 arbitration determines winner of Port 2 (No global arbiter required for Port 2)
Port 2 winner is always the opposite of Port 1 winner (hence the Mirroring Effect)
State Info from Port 2
Input Port 1 (East-West)
Input Port 2 (East-West)
0
1
Port 2 Destination (Opposite of Port 1)
East-West Ports
Decoupling
North-South Ports
Input Ports 3 and 4 (North-South) same as above

Figure 6.4. The Proposed Mirror Allocator

for both directions (East-West or North-South). However, this small overhead is compensated by the fact that only one arbiter is required per module (because of the Mirroring Effect) in the second (global) arbitration stage (see Figure 6.4). The mirror arbiter is ideal for a high-throughput switch, because it resolves HOL blocking, eliminates iterative arbitrations, and reduces the inefficiency of local arbitration. Local arbitration in generic routers first picks a local input request which is then pitted against the winning requests from the other input ports in the global arbitration stage. However, local arbitrations are oblivious of each other, which often leads to unnecessary contention in the global arbitration stage. The Mirroring Effect mechanism, on the other hand, ensures that the most efficient choice is made for each of the two input ports in a module, thus maximizing crossbar utilization.

6.2.4 Flexible and Reusable On-Chip Communication

The routing logic, virtual channel arbitration, switch allocation and switching hardware are all partitioned into two separate and independent modules (row and column sets). This decoupling allows for partial operation in case some component within the router malfunctions or suffers a hard failure. In generic router architectures, a hard failure causes the entire node to be taken off-line, since the operation of the router is unified between all components. In the RoCo router, however, the two disjoint modules function independently. Should a component fail, only the affected module is isolated, with full operation in the remaining module still possible. This would allow the afflicted router to still handle network traffic, albeit in limited directions. The fault-tolerance advantages afforded by the RoCo router are analyzed in the following section.

6.3 Fault-Tolerance through Hardware Recycling

Utilizing the properties of the proposed modular router, a new "Hardware Recycling" mechanism is introduced to ensure fault-tolerant operation of the router. In this section, we explore various possible failure modes within an NoC router, and propose detailed recovery schemes with minimum area and power cost. In conjunction with the proposed solutions, inclusion of fault-tolerant algorithms and adaptive routing would guarantee graceful degradation of performance and packet completion probability in the event of a hard fault.

6.3.1 Fault Model

A typical NoC router architecture consists of a number of fundamental component modules. In existing implementations, these modules operate in a unified environment in which the system can function as a whole only if each and every component within the router is fully functional. This attribute of the router can prove detrimental to the on-chip network in the event of hard faults; if a hard fault incapacitates the entire node, the area surrounding the node will suffer from excessive network congestion. To support better fault-tolerance, a system should be able to continue partial operation even if one of its components has sustained a catastrophic failure. Our proposed RoCo router architecture possesses some inherent fault-tolerance due to its decoupled design, which splits operation in two distinct parallel phases. This additional operational granularity can be utilized to allow replacement of a faulty component by another one, thus allowing partial operation of the router instead of a complete breakdown. The substitution of defective elements by healthy ones elsewhere in the system provides a kind of virtual recycling bin, where functional components can be reused in other parts of the implementation should the need arise. Our proposed scheme avoids the more traditional approach in fault-tolerance, which resorts to replication of resources. Silicon real-estate and energy are at a premium in on-chip applications, thus necessitating the efficient re-use of existing resources.

The proposed router implementation consists of six major components, shown in Table 6.3, each susceptible to permanent faults: the Routing Computation Unit (RC), the Virtual Channel Allocator (VA), the Switch Allocator (SA), the MUXes and DEMUXes which control the flit flow through the virtual channel buffers, the VC buffers, and the crossbar. These components can be classified into two categories, based on their operational regime: (a) per-packet components, and (b) per-flit components. Per-packet components (i.e. the RC and VA) are only used to process the header flit of a new incoming packet. The subsequent flits simply follow the wormhole created by the header flit. Per-flit components (i.e. the remaining components) are used to process every single flit passing through the router. Clearly, since the per-packet based components are driven only by the header flit, their utilization is relatively low compared to the flit-by-flit operation of per-flit components; the latter are fully utilized in non-blocked operation. Thus, packet-based resources can be shared during their unloaded periods. Even though the VA operates only on header flits, in generic NoC router architectures it is almost always busy because it iteratively allocates virtual channels to previously unsuccessful requests. On the contrary, in the proposed RoCo architecture, the VA arbitrates between significantly fewer requests, as a direct result of the decoupled operation of the entire system. This implies that fewer iterations are needed to satisfy all pending requests for a virtual channel, thus leaving the VA unloaded for significantly more cycles than in other architectures. Hence, the VA can be effectively used elsewhere during its idle time.

We further sub-divide the fundamental router components into two classes: message-centric and router-centric. A message-centric component requires a single individual packet as its input, and does not exhibit any inter-dependencies with other incoming messages. The Routing Computation Unit (RC) and the virtual channel buffers are such examples; they operate on a single message (i.e. packet) and their operation does not require state information from other components within the router. On the other hand, router-centric components require inputs from several pending messages in order to execute their function. The VA and SA are such examples; they arbitrate between all messages requesting passage through the router, and their functionality requires state information from the buffers and adjacent routers.

Finally, it is important to note that the operation of the router consists of a critical pathway and non-critical control logic. The datapath of the router (i.e. guided passage of a flit and switch traversal) constitutes the critical pathway; it consists of buffers, decoders, multiplexers and the crossbar. It should be noted that even though the VC buffers lie in the critical datapath, they may or may not be classified as critical, depending on the presence or not of a bypass path. If bypass paths are employed in the buffers for performance optimization, then the VC buffers can be classified as non-critical because of the redundancy supplied by the extra path, as explained later on in the section. Otherwise, the buffers are classified as critical. The operation of the control logic - comprised mostly of the arbiters of the VA and SA - lies in a non-critical pathway. Table 6.3 illustrates the fault classifications of the router components.

Fault Type	Per-flit Operation		Per-packet Operation	
	Critical Pathway	Non-Critical Pathway	Critical Pathway	Non-Critical Pathway
Message Centric	MUX/DEMUX Buffer (w/o bypass path)	Buffer (with bypass path)	–	RC
Router Centric	Crossbar	SA	–	VA

Table 6.3. Component Fault Classification

Each router node is assumed to be able to detect a faulty component through the use of simple control signals. The novelty in our approach lies in the reaction of the router to a hard failure. If a faulty component belongs to a message-centric and non-critical region, the failure can be bypassed instead of resorting to blocking of the whole router module (Row-Module or Column-Module). We can still partially use the router module with the faulty component. If the faulty block lies on the critical path (i.e. datapath), or if it is a router-centric component (i.e. requires state from multiple sources/elements), the permanent failure cannot be bypassed. In this case, the module is isolated and the router remains partially operational through the use of the other parallel module in our proposed scheme. Operational state is tracked by neighboring routers through the use of simple handshaking signals. Should a hard fault be detected by the neighbors, corrective actions are initiated to minimize the influence of the component crash to the entire network. The recovery schemes proposed for each component failure are outlined below:

Routing Computation Unit (RC) Failure: A hard fault in the routing unit logic could cause all flits to be forwarded in the same direction, or, in a more severe case, completely halt the generation of routing signals. The misdirection will not cause any data corruption, but it could lead to deadlock in deterministic routing algorithms. As soon as a failure in the RC unit is detected, it is broadcast to the adjacent routers. After knowing the Routing Computation Unit's (RC) hard fault, the adjacent nodes can now substitute for the faulty RC unit by performing double routing as shown in Figure 6.5. Neighboring nodes sending flits to the faulty router need not worry, because their look-ahead routing will ensure that data arriving at the faulty node has already been taken care of. The problem affects the nodes receiving data from the faulty router; flits arriving at those nodes have not undergone look-ahead routing, because of the faulty RC unit in the previous router. Therefore, nodes receiving flits from the faulty router must first conduct Current-Node Routing on those flits and then proceed to Look-Ahead Routing. This will incur a latency penalty, but it is negligible compared to the latency penalty inflicted on the network by moving the entire faulty node offline. This way, the fault-afflicted node can still serve the on-chip network with the rest of its hardware, which is still fully functional. This scheme avoids the entire blocking of the faulty node, which would generate a potential traffic bottleneck in that region. The overhead involved is minimal and comes only from the few additional control signals; no additional resources are required.

Buffer Failure: In a typical wormhole router, when a flit enters an input port,

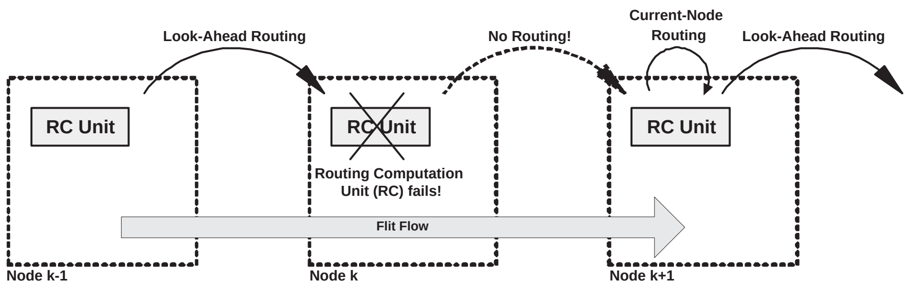


Figure 6.5. Double Routing Mechanism in the event of RC failure

it is written to an input buffer queue, and makes a request for switch traversal when it reaches the head of the queue. When the request is granted, the flit is read out of the input buffer, traverses the crossbar switch fabric and makes its way through the link to the next hop. Bypassing the input buffer when the buffer is empty is a common optimization for performance; the flit heads straight to switch arbitration, and if it succeeds, the flit gets sent directly to the crossbar switch, circumventing the input buffers. This bypass path connecting the router input port with the crossbar input port can be also be utilized in the event of buffer failure within a node. Virtual buffer management and switch allocation can still be performed in the current node, but buffer storage is offloaded to the previous node. As soon as a flit stored in the previous node wins the switch arbitration in the current node, it can use the bypass path to circumvent the faulty buffer and proceed to the healthy crossbar of the current node. In essence, data is physically stored in another router, but virtually queued and arbitrated in a different node through control signals between neighboring routers, as shown in Figure 6.6. This "Virtual Queuing" mechanism permits partial utilization of a partially faulty node. There is a small latency penalty involving the round-trip delays of the handshaking signals, but it does avert the complete isolation of the faulty node. In terms of area cost, Virtual Queuing incurs minimal overhead, since no additional resource is required. The only overhead comes from the addition of a few more control signals.

Virtual Channel Arbiter (VA) Failure: Hard faults in this router-centric component are hardly recoverable by simply sharing of router resources. The operation of the VA cannot be offloaded to surrounding nodes, since its operation requires state information from several sources and it exhibits inter-dependencies

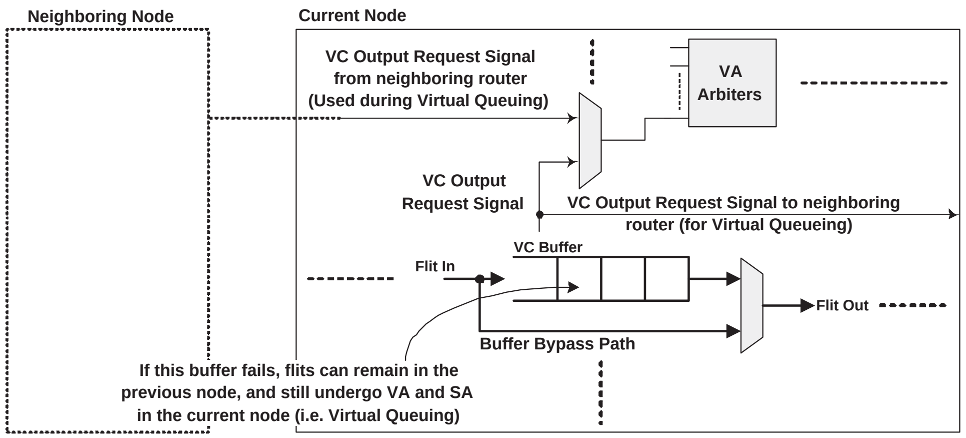

Neighboring Node
Current Node
VC Output Request Signal from neighboring router (Used during Virtual Queuing)
VA Arbiters
VC Output Request Signal
VC Output Request Signal to neighboring router (for Virtual Queueing)
VC Buffer
Flit In
Flit Out
Buffer Bypass Path
If this buffer fails, flits can remain in the previous node, and still undergo VA and SA in the current node (i.e. Virtual Queuing)

Figure 6.6. The Proposed Virtual Queueing Mechanism

with other router elements and downstream nodes. Offloading such operations would require excessive transfer of state info. Faults in the VA can be bypassed only through resource replication, which is costly in terms of area and power overhead. The other option is to offload the arbitrations to the Switch Arbiter hardware, which contains identical arbiter modules. Nevertheless, this is infeasible because the SA is a per-flit component, meaning that it operates on all flits on a cycle-by-cycle basis. Hence, since it is fully utilized, its operation cannot be preempted. The only choice, therefore, is to disable the whole router module in the event of a hard failure in the VA. However, whereas in generic architectures that would mean complete isolation of the entire node, in the proposed architecture only one of the two independent modules needs to be disabled (the one containing the faulty component). The other module may continue to work, providing the node with partial functionality. The router can report its partial failure to its neighbors, which can redirect the corresponding traffic using adaptive routing.

Switch Arbiter (SA) Failure: Despite it being a router-centric component, the SA can still be saved in the event of a hard failure in one of its components. Its operation cannot be transferred to neighboring routers because of the excessive transfer of state information required by such an endeavor. The solution proposed is much simpler and relies on the fact that the SA uses identical hardware with the VA, which is a per-packet component. Per-packet implies lower utilization, as explained in the beginning of this sub-section. This lends itself nicely to sharing of resources. By including a small number of compact 2-to-1 multiplexers at the input of some of the VA's arbiters, the SA can offload its operation to the VA, as shown in Figure 6.7. Given that the VA is operational only for header flit processing, its arbiters can be used by the SA when they are idle. Performance, of course, is degraded because of the sharing of resources, but it is still a preferable alternative to the complete shutdown of the module (Row-Module or Column-Module). The area and power overhead imposed by the MUXes is minimal as compared to the area and power budget of the entire router.

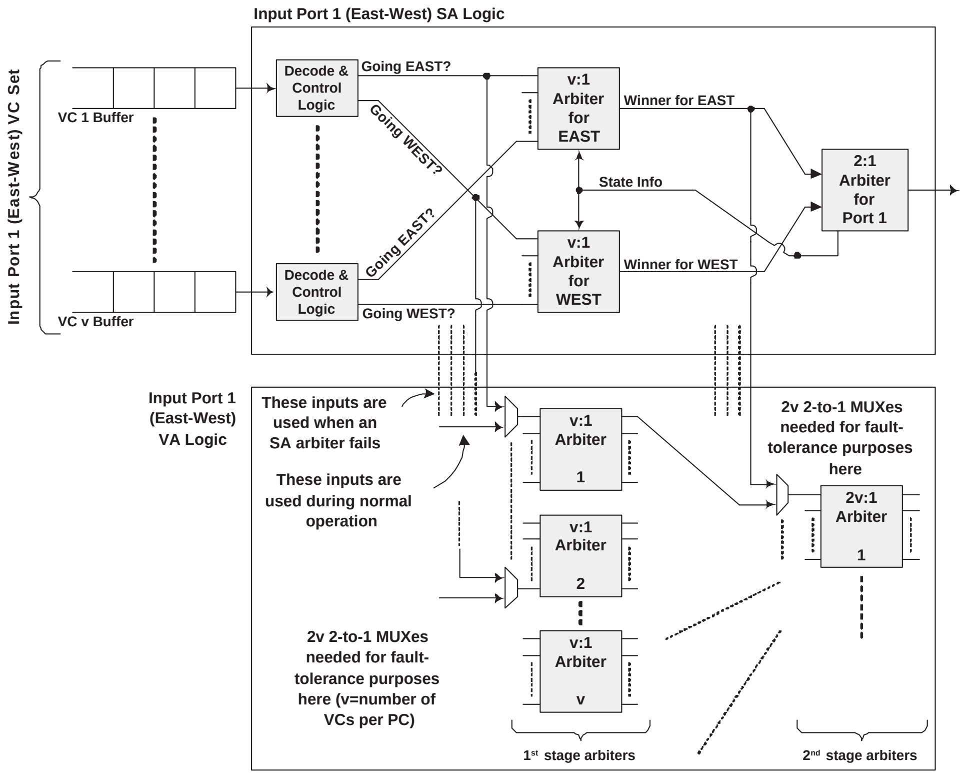

Input Port 1 (East-West) SA Logic
Input Port 1 (East-West) VC Set
VC 1 Buffer
VC v Buffer
Decode & Control Logic
Decode & Control Logic
Going EAST?
Going WEST?
Going EAST?
Going WEST?
v:1 Arbiter for EAST
v:1 Arbiter for WEST
Winner for EAST
Winner for WEST
State Info
2:1 Arbiter for Port 1
Input Port 1 (East-West) VA Logic
These inputs are used when an SA arbiter fails
These inputs are used during normal operation
v:1 Arbiter 1
v:1 Arbiter 2
v:1 Arbiter v
2v 2-to-1 MUXes needed for fault-tolerance purposes here (v=number of VCs per PC)
2v 2-to-1 MUXes needed for fault-tolerance purposes here
2v:1 Arbiter 1
1st stage arbiters
2nd stage arbiters

Figure 6.7. Switch Allocator Fault-Tolerance through Resource Sharing

Crossbar and MUX/DEMUX Faults: In the proposed RoCo router architecture, a decoder (DEMUX) is used to guide a flit into a group of path-sensitive queues, and a multiplexer (MUX) is used to direct a winning flit to the crossbar input. Therefore, the MUXes and DEMUXes all lie on the critical pathway of the router. A hard failure in one of these critical components can severely hamper the datapath progression. Once again, bypassing the datapath would imply replication of resources, which is not desirable. Hence, if any of these modules is rendered useless, the corresponding router module is blocked while the other, sound module keeps operating. The partial operation of the other module reduces congestion around the faulty node, giving more alternative choices to the surrounding routers as compared to a conventional router architecture.

6.3.2 Recycling Mechanism

Once a permanent failure is detected, the "Recycling Mechanism" diagnoses the degree of malfunction according to the aforementioned fault types and internal dependencies. Based on pre-classified information on the possible impact of each component defect, it decides if the faulty router module may continue to function by bypassing the faulty component through re-use of neighboring resources. If not, then the affected router module is isolated and partial operation continues using the remaining healthy module. It is important to note that the proposed recycling mechanism is not a monolithic hardware block; such an approach would raise the issue of a hard failure occurring within the block, which compromises the effectiveness of the block's existence. Instead, the mechanism takes a more distributed form, interspersed between the router's components. Hence, a failure in any component is treated independently by local resources in a decentralized manner. Furthermore, as described in the previous sub-section, the recycling mechanism consists mostly of control signals with minimal hardware overhead. The health status of each router is reported to all its neighboring nodes. The Recycling Mechanism operates in four distinct phases: (1) detect, (2) diagnose, (3) replace and/or isolate, and (4) notify. Thus, our flexible yet compact fault-tolerant mechanism minimizes the influence of a malfunctioning component on the performance of the on-chip network.

6.4 Performance Evaluation

In this section, we present simulation-based performance evaluation of our architecture, a generic router architecture and the Path-Sensitive architecture [80], in terms of network latency, energy consumption and fault-tolerance under various traffic patterns. We describe our experimental methodology, and detail the procedure followed in the evaluation of these architectures.

6.4.1 The Performance, Energy and Fault-Tolerance (PEF) Metric

Traditional performance metrics used in NoC analysis, such as the Energy-Delay Product (EDP) and Power-Delay Product (PDP), focus on the two fundamental notions of latency (i.e. performance) and energy/power consumption. These metrics, however, do not capture the importance of reliability and its relation to both performance and power. Given that reliability is becoming a major concern in deep sub-micron technologies, it is imperative that evaluation of modern systems accounts for such issues. To address this need, we propose a composite metric in this work, which unifies all three components: latency, energy, and fault-tolerance. Before introducing the new metric, we present three related definitions.

Network Latency: This is defined as the average number of cycles taken for end-to-end packet traversal, i.e. from source to destination. It is the baseline parameter which determines the performance of a specific architecture.

Energy Consumption per Packet: This is divided into two components: dynamic and leakage energy consumption. Both are defined as the total dynamic (or leakage) energy consumed in the network fabric over a time period divided by the total number of packets delivered during that period. Leakage power captures the effect of blocking delay, which translates into buffer static energy consumption. Dynamic power captures the effect of high contention within the router, which increases energy consumption due to excessive iterative operation of the SA and the VA units.

Packet Completion Probability. This is defined as the number of received messages divided by the total number of injected messages in the on-chip network. Deterministic XY routing cannot provide alternate paths in the event of a hard failure, because of its lack of routing flexibility. XY-YX routing can provide some routing flexibility to avoid a faulty node, but it is still limited by a fairly rigid routing policy. Minimal adaptive routing offers more flexibility in providing alternate paths in faulty environments. It should be noted, however, that fault-tolerant non-minimal routing algorithms may adversely affect average latency, while improving the packet completion probability. This, in turn, may lead to higher energy consumption due to increased traffic in the network.

The inter-dependence between speed, power and fault-tolerance highlights the importance of a metric which can identify the best tradeoffs between these three competing traits. Hence, we introduce the Performance, Energy and Fault-tolerance (PEF) metric, as a comprehensive parameter that reflects the correlation between the three desired design goals. We define PEF as $\frac{(Average\ Latency)\times(Energy\ per\ Packet)}{Packet\ Completion\ Probability}$, i.e. PEF=$\frac{Energy-Delay-Product}{Packet\ Completion\ Probability}$. In a fault-free network, $Packet\,Completion\,Probability$ = 1; thus, PEF becomes equal to EDP. Hence, PEF integrates reliability into EDP, thus providing a more complete evaluation metric.

6.4.2 Performance Results

The performance of the proposed RoCo router was analyzed and compared to two other existing router architectures (generic router, Path-Sensitive router of [80]) using the aforementioned cycle-accurate simulator. All architectures under investigation were evaluated using an 8x8 2D Mesh network. Wormhole routing based on virtual-channel flow control was employed in all cases. In the generic router architecture, 3VCs per port were assumed, with a 4-flit deep buffer per VC; for a 5-port router, this configuration gives a total buffer capacity of 60 flits per router. To ensure fairness, since both our proposed RoCo architecture and the Path-Sensitive router have 4 ports instead of 5, we assumed 3VCs per port in both implementations, each with a 5-flit deep buffer; this gives a total buffer capacity of 60 flits per router, similar to the generic case.

Several experiments were conducted to evaluate the performance of all architectures under various traffic patterns and three different routing algorithms. We used uniform and transpose [9] traffic, and two synthesized workload traces: self-similar web traffic [31] and MPEG-2 video multimedia traces [140] in three different routing algorithms: DOR (XY routing), oblivious XY-YX routing, and minimal adaptive routing schemes. Average network latency and power consumption were recorded for all experiments. Furthermore, several experiments were conducted to evaluate performance in faulty environments. A number of router faults (both Message-Centric and Router-Centric, as explained in Section 6.3.1) were randomly injected in the network infrastructure and the packet completion probability was analyzed. The traffic injection rate was 30%. The latency, energy and fault-tolerance results were subsequently integrated into the PEF metric of Section 6.4.3

to reflect the combined measure.

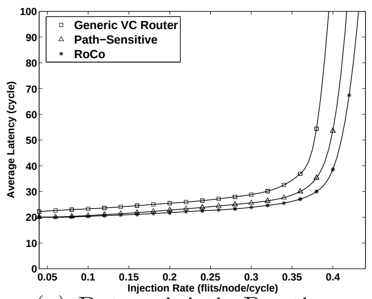
The latency results of all three architectures for various traffic patterns are illustrated in Figures 6.8 through 6.11. Clearly, the proposed RoCo router outperforms both the generic and Path-Sensitive routers in all traffic patterns and routing algorithms. With deterministic routing, the RoCo router reduces average latency by up to 35% compared to the generic router and by about 7% compared to the Path-Sensitive router. With XY-YX routing, these numbers become 38% and 10% respectively. Finally, in adaptive routing, latency reduces by up to 40% compared to the generic router, and about 4% compared to the Path-Sensitive router. The decoupling of the architecture into two distinct and functionally independent modules significantly reduces contention probability within the router. This effect manifests itself in lower average latency within the network. Furthermore, the use of the novel Mirroring Effect in switch arbitration increases crossbar utilization and reduces blocking.

Figure 6.14 compares the energy efficiency conservation behavior of the three different router architectures at 30% injection rate. The energy per packet is about 20% lower in the RoCo router, as compared to the generic router architecture, and about 6% compared to the Path-Sensitive router. This is a consequence of the simpler crossbars, smaller VA and SA units, and shorter logic depth. Therefore, the benefits afforded by the RoCo router are two-fold: reduced average network latency and lower energy consumed per packet. This is testament to the fact that a more streamlined architecture can benefit both performance and power consumption.

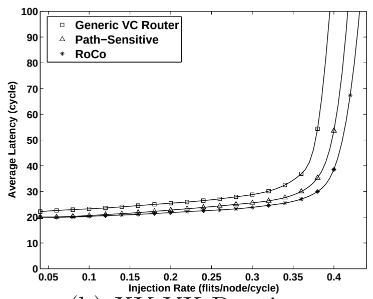
Figures 6.12 and 6.13 illustrate the packet completion probabilities of the three router architectures when operating in faulty environments with 1, 2 and 4 random network faults. Figure 6.12 concentrates on Router-Centric faults. These are critical faults which cause the entire node to be blocked in the generic and Path-Sensitive cases. In the RoCo architecture, however, such faults only cause one of the two modules (Row-Module or Column-Module) to be blocked, thus, allowing for partial operation of the faulty router. Completion probability is consistently higher in the RoCo router in Figure 6.12. As the number of faults increases from 1 to 4, the advantage of the proposed router becomes more obvious. The RoCo router provides up to 70% improvement in packet completion probability for different fault patterns with deterministic routing. The improvement drops to about 7% when adaptive routing is used. In both XY-YX and adaptive routing, the results are close because the routing algorithms provide alternate paths for all three architectures. However, this metric alone does not reflect the fact that even though completion probability is high in the generic and Path-Sensitive cases, the latency penalty incurred by excessive congestion around the faulty nodes is very high. This result will be captured later on in the PEF metric.

Figure 6.13 focuses on Message-Centric faults, which are not critical. In the generic and Path-Sensitive routers, such faults would still cause the entire node to be blocked. However, in the RoCo router, such faults are remedied by the Recycling Mechanism of Section 4, which bypasses faults through resource sharing. The oblivious routing schemes, i.e. deterministic and XY-YX, suffer more in the presence of faults, because of their rigid routing policies. These results indicate that the newly proposed Recycling Mechanism improves completion probability considerably without any significant router area overhead. Furthermore, even during critical Router-Centric faults, partial operation of the router can still serve network traffic in one dimension (through the use of a single healthy module), thus, alleviating congestion around the faulty node. This is achieved without any additional overhead. The results indicate that the RoCo router can achieve packet completion probabilities in oblivious routing that are close to those of adaptive routing schemes. This is of profound importance, since it indicates that the RoCo router provides uniform fault-tolerance under all routing algorithms. Through the recycling of faulty components and resource sharing, our proposed architecture degrades gracefully in faulty environments.

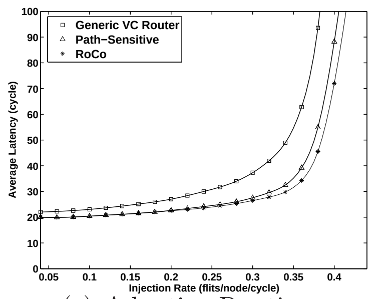
Figure 6.15 shows the combined measure, PEF results for all three router architectures. The bars use the scale on the left-hand axis, while the curves use the scale on the right-hand axis. This metric can successfully capture the subtle fact that despite high completion probabilities in adaptive routing, the generic and Path-Sensitive routers suffer from high latency due to congestion created around the faulty nodes. The RoCo router, on the other hand, has significantly lower latency numbers, due to graceful degradation and the novel hardware recycling mechanism. Taking into consideration performance, energy consumption, and fault-tolerance in the integrated PEF metric, the RoCo router turns out to be the clear winner compared to the other two architectures. It provides almost 50% improvement compared to generic router and 35% improvement compared to the Path-Sensitive router.



(a) Deterministic Routing

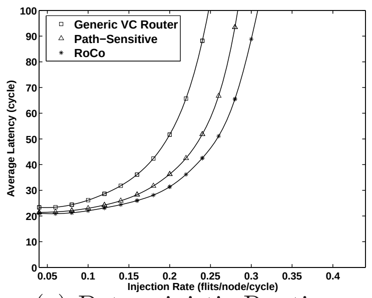


(b) XY-YX Routing

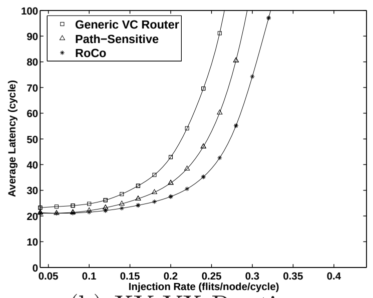


(c) Adaptive Routing

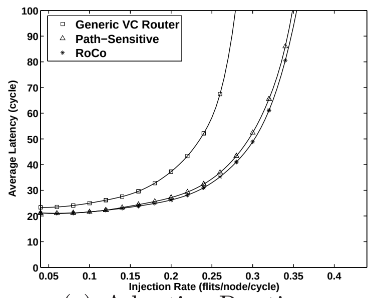
Figure 6.8. Uniform Random Traffic



(a) Deterministic Routing



(b) XY-YX Routing

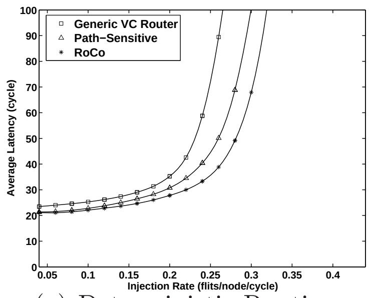


(c) Adaptive Routing

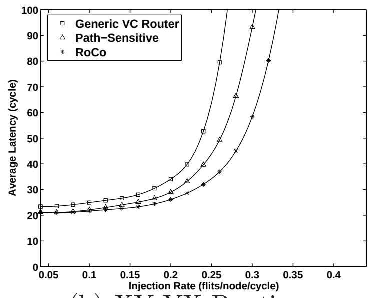
Figure 6.9. Self-Similar Traffic

6.5 Conclusions

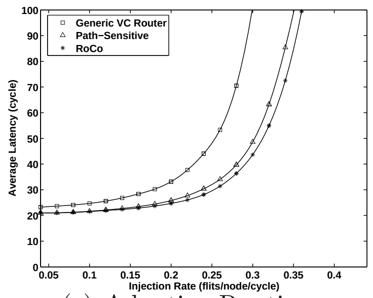
In this chapter, we have presented a new router architecture, called Row-Column Decoupled Router, suitable for on-chip interconnects. The uniqueness of the proposed router is that it considers the three desirable objective functions: performance, energy and fault-tolerance, in exploring the design space. The proposed two-stage wormhole-switched RoCo router has a number of features that make it distinct compared to the earlier designs. First, it uses two smaller 2×2 crossbars instead of a larger 5×5 crossbar that is traditionally used for 2D mesh networks.



(a) Deterministic Routing

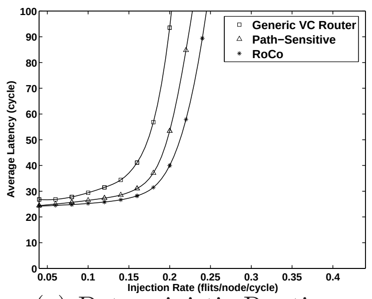


(b) XY-YX Routing

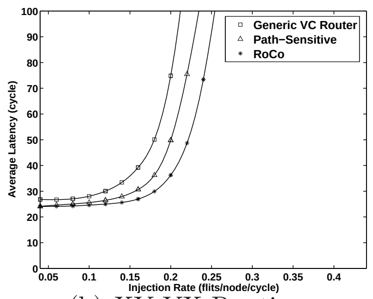


(c) Adaptive Routing

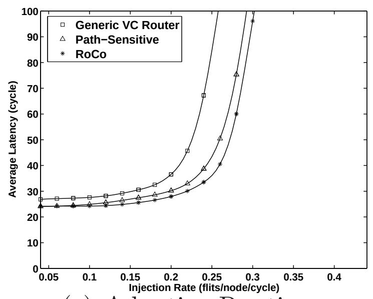
Figure 6.10. Multimedia Traffic



(a) Deterministic Routing



(b) XY-YX Routing

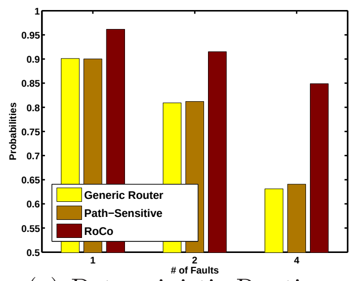


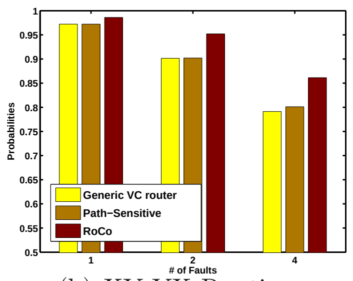
(c) Adaptive Routing

Figure 6.11. Transpose Traffic

Second, it uses a path-sensitive buffering scheme, where, the virtual channels are divided into four sets to support dedicated row and column routing in the two crossbars. These two features along with early ejection, mirrored allocation, look-ahead routing and speculative path selection, help in reducing the contention. Third, unlike most earlier designs, we show how deterministic (XY) routing, XY-YX routing and adaptive routing can be supported in this architecture. Fourth, because of the modular design, we show how different types of faults such as VA, SA, and crossbar failures can be handled with graceful degradation, thereby providing better fault-tolerance compared to earlier designs. In addition, while all prior NoC studies have analyzed at best two of the three parameters, such as energy-delay product, we introduce a comprehensive parameter, called PEF, for analyzing the performance, energy and fault-tolerance attributes of NoC architectures.

A flit-level, cycle-accurate simulator along with a detailed energy model for 90 nm synthesis were used to analyze the three objective functions using a variety


(a) Deterministic Routing


(b) XY-YX Routing

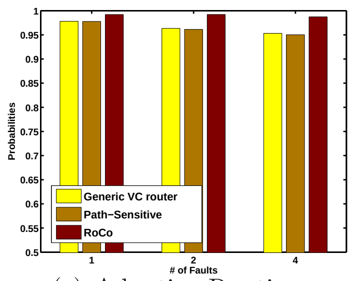
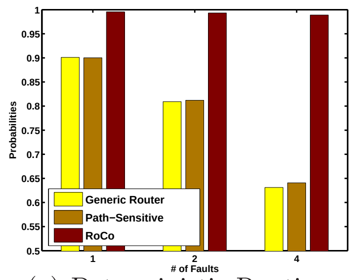
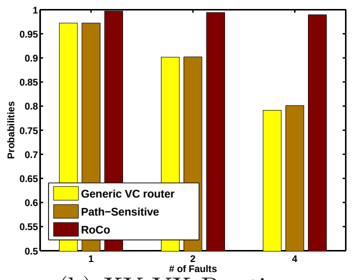

(c) Adaptive Routing

Figure 6.12. Packet Completion Probabilities under Various Faults within Router-Centric or Critical Pathway Components


(a) Deterministic Routing


(b) XY-YX Routing

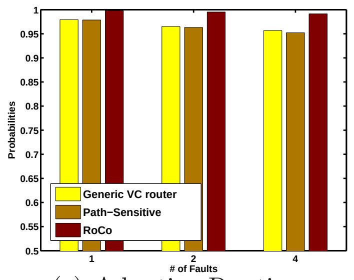

(c) Adaptive Routing

Figure 6.13. Packet Completion Probabilities under Various Faults within Message-Centric and Non-Critical Components

of traffic patterns. Our performance analysis with an 8×8 network shows that the proposed router can reduce the average network latency up to 40% compared to a generic 2-stage router and by 10% compared to the path-sensitive router. This is a significant improvement considering the fact that the 2-stage speculative router design is a quite aggressive architecture. In terms of energy consumption per packet, the proposed RoCo design outperformed the 2-stage router and Path-Sensitive router by 20% and 6%, respectively. The packet completion probability is improved by about 70% with deterministic routing. Evaluation with the composite performance, energy and fault-tolerance parameter (PEF) indicates that our architecture provides 50% and 35% better results compared to the generic and Path-Sensitive models, respectively.

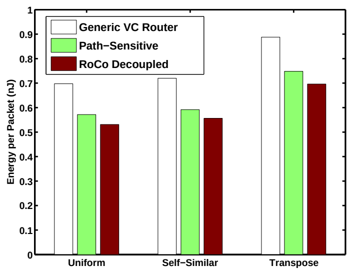
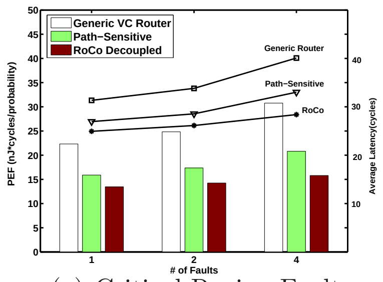
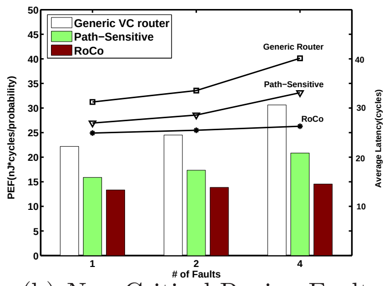


Figure 6.14. Energy per Packet



(a) Critical Region Fault



(b) Non-Critical Region Fault

Figure 6.15. Performance-Energy-Fault (PEF) Product

Chapter 7

A 3-D Router Architecture

In this chapter, we explore the design of a 3-D, crossbar-style, NoC for upcoming 3-D VLSI technology. This exploration provided insight into the tradeoffs between circuit complexity and performance using real commercial and scientific benchmarks in the simulation testbed.

7.1 Introduction

3D chip technology promises to reduce interconnect delays by stacking multiple layers on top of each other, and by providing shorter vertical connections [141]. 3D technology has matured and demystified some of the concerns on thermal viability and reliability of inter-wafer vias. In addition, it promises to enable integration of heterogeneous technologies on the same chip − such as having layers of memory stacked on top of processor cores − and is even attractive for placing analog and digital components on the same chip, as this avoids common substrate noise problems. Interconnect architecture design across the layers in a 3D architecture requires careful attention for the components on different layers to communicate effectively. Furthermore, there is a need for an integrated approach to interconnect design in the 2D planes and the vertical direction. Currently, there exists no systematic effort at exploring the interconnect architecture for 3D chips. Recently, researchers have started examining some tradeoffs, such as the influence of bandwidth variation of inter-layer interconnects between processor and memory subsystems [142], and combining vertical interconnects with an NoC fabric for chip multiprocessor caches [143].

In this chapter, we investigate various architectural options for 3D NoC design. Interconnect design in 3D chips imposes new constraints and opportunities compared to that of 2D NoC design. There is an inherent asymmetry in the delays in a 3D architecture between the fast vertical interconnects and the horizontal interconnects that connect neighboring cores, due to differences in wire lengths (few tens of μm in the vertical direction as compared to few thousand μm in the horizontal direction). Consequently, extending a traditional NoC fabric to the third dimension by simply adding routers at each layer (called the Symmetric NoC in this work, due to the symmetry of routing in all directions) is not a good option, as router latencies may dominate the fast vertical interconnect. Hence, we explore two alternate options; a 3D NoC-bus hybrid structure and a true 3D router fabric for the vertical, inter-strata interconnect. A key challenge with 3D NoC routers is limiting the arbitration complexity due to the large path diversity resulting from the additional interconnects in the third dimension. Vertical interconnects also

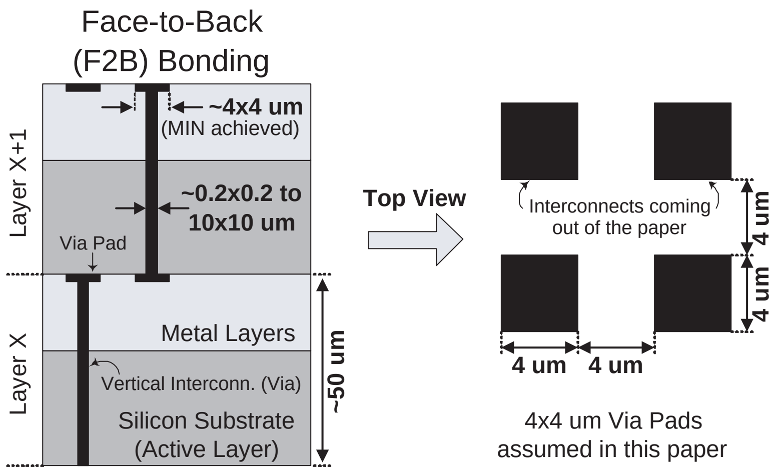


Figure 7.1. Face-to-Back (F2B) Bonding and the Assumed Vertical Via Layout in this Paper

impose a larger area overhead than corresponding horizontal wires due to the requirement for bonding pads, and can compete with device area as the inter-strata vias punch through the wafer when Face-to-Back (F2B) bonding (see Figure 7.1) is used. Therefore, the desired number of vertical interconnects used in the 3D router architecture needs to be investigated.

In exploring these tradeoffs in a 3D router design, we developed a new 3D NoC router architecture that we call the 3D Dimensionally-Decomposed (DimDe) Router. The name is a direct corollary of the fact that communication flow through the DimDe router is classified according to the three axes in Euclidean space: X (corresponding to East-West intra-layer traffic), Y (corresponding to North-South intra-layer traffic), and Z (corresponding to inter-layer traffic in the vertical dimension). The idea of decomposing traffic in two dimensions in a 2D environment was introduced in [74]. While our proposed DimDe router was inspired by the work in [74], our contribution goes well beyond the introduction of a new traffic dimension. *The DimDe router fuses the crossbars of all the routers in the same vertical "column" (i.e. same X, Y coordinate but different Z coordinate) into a unified entity which allows coordinated concurrent communication across different layers through the same crossbar.* This design amounts to a *true physical* 3D crossbar (unlike the mere stacking of 2D routers in multiple wafer layers). It is important to note that 3D topologies have long been in existence in the macro-network field (e.g. k-ary n-cube), but these rely on 2D routers connected in such a way as to form a *logical* 3D topology. However, 3D chip integration is now enabling the creation of a true physical 3D topology, where the router is itself a three-dimensional entity.

DimDe exhibits the following characteristics that make it a desirable interconnect structure for 3D designs:

(1) *DimDe supports a true 3D crossbar structure which spans all the active layers of the chip.* Irrespective of the number of layers used in the implementation, the 3D crossbar allows a single-hop connection between any two layers, treating all strata as part of a single router structure.

(2) The DimDe design-space provides options for varying the number of vertical connections from one to four to emulate anything between a segmented bus and a full crossbar. Through design space exploration, DimDe was selected to support two vertical interconnects to strike a balance between the path diversity and high bandwidth offered by a full 3D crossbar and the simplicity of a bus. Most importantly, *DimDe's partially-connected crossbar achieves performance levels similar to those of a full 3D crossbar, with substantially reduced area and power overhead and orders of magnitude lower control logic complexity.*

(3) *DimDe supports segmented vertical (i.e. inter-strata) links in the partially-connected crossbar to enable concurrent communication between the different layers of the 3D chip.* This simultaneous data transfer in the vertical dimension significantly increases the vertical bandwidth of the chip as compared to a 3D NoC-bus hybrid structure.

(4) *The DimDe design employs a hierarchical arbitration scheme for inter-strata transfers that reduces area and delay complexity*, while still efficiently enabling simultaneous data transfers. The first stage arbitrates between all requests for vertical communication from within a single layer and the second stage accommodates as many simultaneous requests from the winners of the first stage arbitration.

(5) Similar to the Row-Column (RoCo) Decoupled Router of [74], DimDe completely separates East-West and North-South intra-layer traffic through a pre-sorting operation at the input. However, inter-layer traffic cannot be completely isolated in its own module. A true 3D crossbar requires inter-layer traffic to merge with intra-layer traffic in a seamless fashion; this would allow incoming packets from different layers to continue traversal in the destination layer. DimDe facilitates this tight integration by augmenting the Row (East-West) and Column (North-South) modules with a Vertical Module which fuses with the other two. *The Vertical Module then extends to all other layers and unifies them in a single operational entity.* The Vertical Module assumes the double role of "gluing" all the layers together and blending inter- and intra-layer traffic through unidirectional connections to the Row and Column modules of all layers. It will be demonstrated that this approach dramatically reduces the 3D crossbar complexity, while still allowing concurrent communication between different layers through the switch.

We compare the proposed 3D router design to four different interconnect architectures: a 2D NoC, a 3D Symmetric NoC, a 3D NoC-Bus Hybrid, and a Full 3D Crossbar[1] implementation. To provide as comprehensive an evaluation as possible, we employed a two-pronged simulation environment: (a) a stand-alone, cycle-accurate NoC simulator running synthetic workloads, and (b) a hybrid NoC/cache simulator running a variety of commercial and scientific workloads within the context of a shared, multi-bank NUCA L2 cache in an 8-CPU Chip Multi-Processor

[1]Our interpretation of a "full" 3D crossbar is presented in Section 3.4 and subsequently formalized in Section 5.3.

(CMP) scenario. This double-faceted evaluation process ensures exposure to several traffic patterns, including request/reply memory traffic.

The proposed DimDe design consistently provides the lowest latency for different traffic patterns and it saturates at higher workloads compared to other considered architectures. Our synthetic workload results show that, for high traffic loads, the recently proposed 3D NoC-Bus Hybrid Architecture [143] exhibits the worst latency and throughput for all traffic patterns (even worse than the 2D topology), as the bus saturates first with higher workload. In terms of throughput behavior, the DimDe architecture provides 18% average improvement over the other designs, while remaining within around 3% of the Full 3D Crossbar's throughput. The real workload results indicate that DimDe provides an average improvement of 27% over the 3D Symmetric and 3D NoC-Bus Hybrid designs, and remains within 4% of the Full 3D Crossbar's performance. However, with the Energy-Delay Product (EDP) as the metric, *DimDe significantly outperforms all other designs, including the Full 3D Crossbar, by 26% on average.* Hence, when accounting for both performance and power consumption, the DimDe design is superior to all other 3D router architectures analyzed in this paper. To the best of our knowledge, this is the first systematic exploration and analysis of 3D interconnect architectures and their ramifications on overall system performance.

The rest of chapter is organized as follows. The next section discusses related work. Section 7.3 provides details of the different 3D interconnect architectures. Section 7.4 delves into the proposed DimDe architecture. Section 7.5 presents experimental results, and the conclusions are drawn in section 7.6.

7.2 Related Work

The work related to this Chapter is summarized in two sub-sections: 3D Technology and 3D Architectures.

7.2.1 3D Integration Technology

Three-dimensional integration technology [144] is an attractive option for overcoming the barriers in interconnect scaling, offering an opportunity to continue the CMOS performance trend. In a three-dimensional (3D) chip, multiple device layers are stacked together. Various 3D integration vertical interconnect technologies have been explored, including wire bonded, microbump, contactless (capacitive or inductive), and through-via vertical interconnect [144]. Through-via interconnection has the potential to offer the greatest vertical interconnect density and therefore is the most promising one among these vertical interconnect technologies. There are two different approaches to implementing through-via 3D integration: the first one involves sequential device process, in which the front-end processing (to build the device layer) is repeated on a single wafer to build multiple active device layers, before the interconnects among devices are built. The second approach processes each active device layer separately, using conventional fabrication techniques, and then stacking these multiple device layers together using wafer-bonding technology. The latter approach requires minimal changes to the manufacturing steps and is more promising; therefore, it is adopted in our proposed architecture. Wafers can be bonded Face-to-Face (F2F) or Face-to-Back (F2B). The through wafer via in F2F wafer-bonding does not go through the thick buried silicon layer and can be fabricated with smaller via sizes. However, for 3D Integrated Circuits (IC) with more than two active layers, F2B stacking provides better scalability, and, therefore, is adopted in our architecture.

Thermal considerations have been a significant concern for 3D integration [145]. However, various techniques have been developed to address thermal issues in 3D architectures such as physical design optimization through intelligent placement [146], increasing thermal conductivity of the stack through insertion of thermal vias [145], and use of novel cooling structures [147]. Further, a recent work demonstrated that the areal power density is the more important design constraint in placement of the processing cores in a 3D chip, as compared to their location in the 3D stack [148]. Consequently, thermal concern can be managed as long as components with high power density are not stacked on top of each other. Architectures that stack memory on top of processor cores, or those that rely on low-power processor cores have been demonstrated to not pose severe thermal problems [149]. In spite of all these advances, one can anticipate some increase in temperature as compared to a 2D design, and also a temperature gradient across layers. Increased temperatures increase wire resistances, and consequently the interconnect delays.

To capture this effect, we study the impact of temperature variations on the 3D interconnect delay to assess the effect on performance.

7.2.2 3D Architectures

Modern System-on-Chip (SoC) designs, such as CMPs, can benefit from 3D integration as well. For example, by placing processing memory, such as DRAM and/or L2 caches, on top of the processing core in different layers, the bandwidth between them can be significantly increased and the critical path can be shortened [150]. In this context, [143] proposed a 3D Network-in-Memory architecture and explored the challenges of managing 3D CMPs together with L2 cache design-space issues. They also proposed the use of an NoC-Bus Hybrid structure for the 3D interconnect. In this paper, we use this structure as one of the comparison points and demonstrate that our proposed architecture is superior. In [142], a CMP design with stacked memory layers is proposed. The authors show that the L2 cache can be removed due to the availability of wide low-latency inter-layer buses between the processing core layer and DRAM layers, and the area saved from this can be recycled for additional cores. Also, [151] has proposed a multi-bank uniform on-chip cache structure using 3D integration. The notion of adding specialized system analysis hardware on separate active layers stacked vertically on the processor die using 3D IC technology is explored in [152]. The modular snap-on introspective layer collects system statistics and acts like a hardware system monitor.

7.3 Three-Dimensional Network-on-Chip Architectures

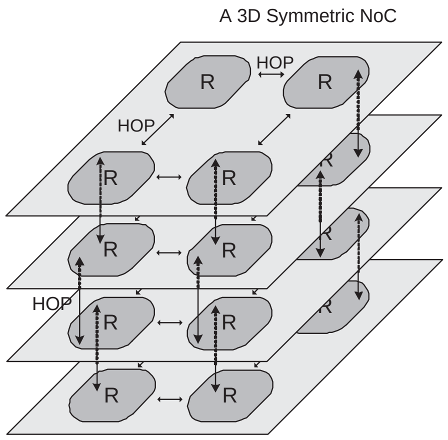
This section delves into the exploration of possible architectural frameworks for a three-dimensional NoC network. A typical 2D NoC consists of a number of Processing Elements (PE) arranged in a grid-like mesh structure, much like a Manhattan grid. The PEs are interconnected through an underlying packet-based network fabric. Each PE interfaces to a network router through a Network Interface Controller (NIC). Each router is, in turn, connected to four adjacent routers, one in each cardinal direction.

Expanding this two-dimensional paradigm into the third dimension poses interesting design challenges. Given that on-chip networks are severely constrained in terms of area and power resources, while at the same time they are expected to provide ultra-low latency, the key issue is to identify a reasonable tradeoff between these contradictory design threads. Our task in this section is precisely this: to explore the extension of a baseline 2D NoC implementation into the third dimension, while considering the aforementioned constraints.

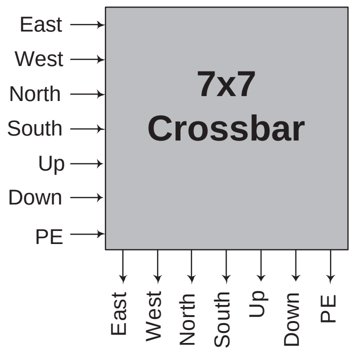
7.3.1 A 3D Symmetric NoC Architecture

The natural and simplest extension to the baseline NoC router to facilitate a 3D layout is simply adding two additional physical ports to each router; one for Up and one for Down, along with the associated buffers, arbiters (VC arbiters and Switch Arbiters), and crossbar extension. We call this architecture a 3D Symmetric NoC, since both intra- and inter-layer movement bear identical characteristics: hop-by-hop traversal, as illustrated in Figure 7.2(a). For example, moving from the bottom layer of a 4-layer chip to the top layer requires 3 network hops.

This architecture, while simple to implement, has two major inherent drawbacks: (1) It wastes the beneficial attribute of a negligible inter-wafer distance (around 50 μm per layer) in 3D chips, as shown in Figure 7.1. Since traveling in the vertical dimension is multi-hop, it takes as much time as moving within each layer. Of course, the average number of hops between a source and a destination does decrease as a result of folding a 2D design into multiple stacked layers, but inter-layer and intra-layer hops are indistinguishable. Furthermore, each flit must undergo buffering and arbitration at every hop, adding to the overall delay in moving up/down the layers. (2) The addition of two extra ports necessitates a larger 7×7 crossbar, as shown in Figure 7.2(b). Crossbars scale upward very inefficiently, as illustrated in Table 7.1. This table includes the area and power budgets of all crossbar types investigated in this paper. Details of the design and synthesis methodology are given in Section 7.5.2. Clearly, a 7×7 crossbar incurs significant area and power overhead over all other architectures. Therefore, the 3D Symmetric NoC implementation is a somewhat naive extension to the baseline 2D network.



(a) Overall View



(b) Crossbar Configuration

Figure 7.2. A 3D Symmetric NoC Network

Crossbar Type	Area	Power with 50% switching activity at 500 MHz
4×2 Crossbar (for 3D DimDe)	3039.32 μm^2	1.63 mW
5×5 Crossbar (Conventional 2D Router)	8523.65 μm^2	4.21 mW
6×6 Crossbar (3D NoC-Bus Hybrid)	11579.10 μm^2	5.06 mW
7×7 Crossbar (3D Symmetric NoC Router)	17289.22 μm^2	9.41 mW

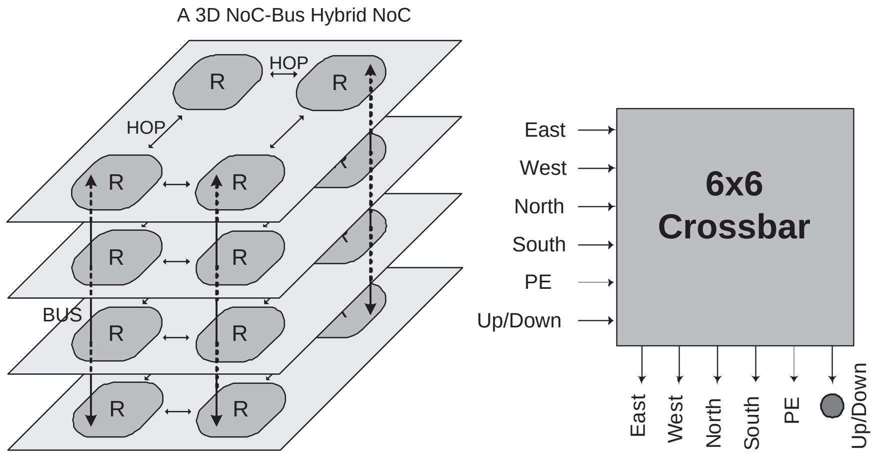
Table 7.1. Area and Power Comparisons of the Crossbar Switches Assessed in this Work

7.3.2 The 3D NoC-Bus Hybrid Architecture

The previous sub-section argues that multi-hop communication in the vertical (inter-layer) dimension is not desirable. Given the very small inter-strata distance, single-hop communication is, in fact, feasible. This realization opens the door to a very popular shared-medium interconnect, the bus. The NoC router can be hybridized with a bus link in the vertical dimension to create a 3D NoC-Bus Hybrid structure, as shown in Figure 7.3(a). This approach was first introduced in [143], where it was used in a 3D NUCA L2 Cache for CMPs. This hybrid system provides both performance and area benefits. Instead of an unwieldy 7×7 crossbar, it requires a 6×6 crossbar (Figure 7.3(b)), since the bus adds a single additional port to the generic 2D 5×5 crossbar. The additional link forms the interface between the NoC domain and the bus (vertical) domain. The bus link has its own dedicated queue, which is controlled by a central arbiter. Flits from different layers wishing to move up/down should arbitrate for access to the shared medium.

Figure 7.4 illustrates the side view of the vertical via structure. This schematic depicts the usefulness of the large via pads between the different layers; they are deliberately oversized to cope with misalignment issues during the fabrication process. Consequently, it is the large vias which ultimately limit vertical via density in 3D chips.

Despite the marked benefits over the 3D Symmetric NoC router of Section 7.3.2, the bus approach also suffers from a major drawback: it does not allow concurrent communication in the third dimension. Since the bus is a shared medium, it can only be used by a single flit at any given time. This severely increases contention and blocking probability under high network load, as will be demonstrated in Section 7.5. Therefore, while single-hop vertical communication does improve performance in terms of overall latency, inter-layer bandwidth suffers.



(a) Overall View (b) Crossbar Configuration

Figure 7.3. A 3D NoC-Bus Hybrid Architecture

7.3.3 A True 3D NoC Router

Moving beyond the previous options, we can envision a true 3D crossbar implementation, which enables seamless integration of the vertical links in the overall router operation. Figure 7.5 illustrates such a 3D crossbar layout. The vertical links are now embedded in the crossbar and extend to all layers. This implies the use of a 5×5 crossbar, since no additional physical channels need to be dedicated for

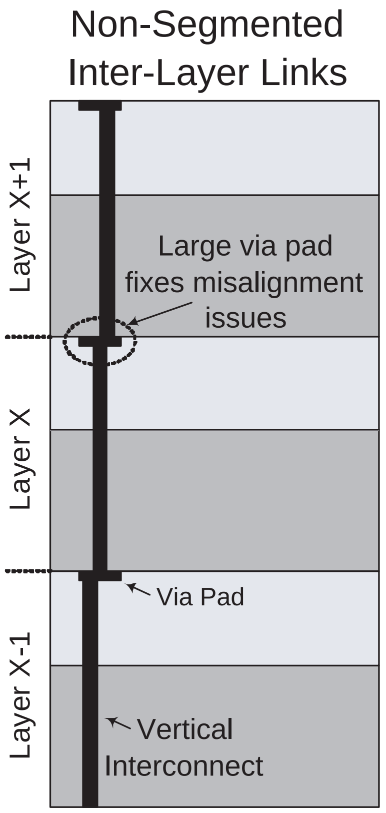

Non-Segmented Inter-Layer Links
Layer X+1
Layer X
Layer X-1
Large via pad fixes misalignment issues
Via Pad
Vertical Interconnect

Figure 7.4. Side View of the Inter-Layer Via Structure in a 3D NoC-Bus Hybrid Structure

inter-layer communication. As shown in Table 7.1, a 5×5 crossbar is significantly smaller and less power-hungry than the 6×6 crossbar of the 3D NoC-Bus Hybrid and the 7×7 crossbar of the 3D Symmetric NoC. Interconnection between the various links in a 3D crossbar would have to be provided by dedicated connection boxes at each layer. These connecting points can facilitate linkage between vertical and horizontal channels, allowing flexible flit traversal within the 3D crossbar. The internal configuration of such a Connection Box (CB) is shown in Figure 7.6(a). The horizontal pass transistor is dotted, because it is not needed in our proposed 3D crossbar implementation, which is presented in Section 4. The vertical link segmentation also affects the via layout, as illustrated in Figure 7.6(b). While this layout is more complex than that shown in Figure 7.4, the area between the offset vertical vias can still be utilized by other circuitry, as shown by the dotted ellipse in Figure 7.6(b).

Hence, the 2D crossbars of all layers are physically fused into one single three-dimensional crossbar. Multiple internal paths are present, and a traveling flit goes through a number of switching points and links between the input and output ports. Moreover, flits re-entering another layer do not go through an intermediate buffer; instead, they directly connect to the output port of the destination layer. For example, a flit can move from the western input port of layer 2 to the northern output port of layer 4 in a single hop.

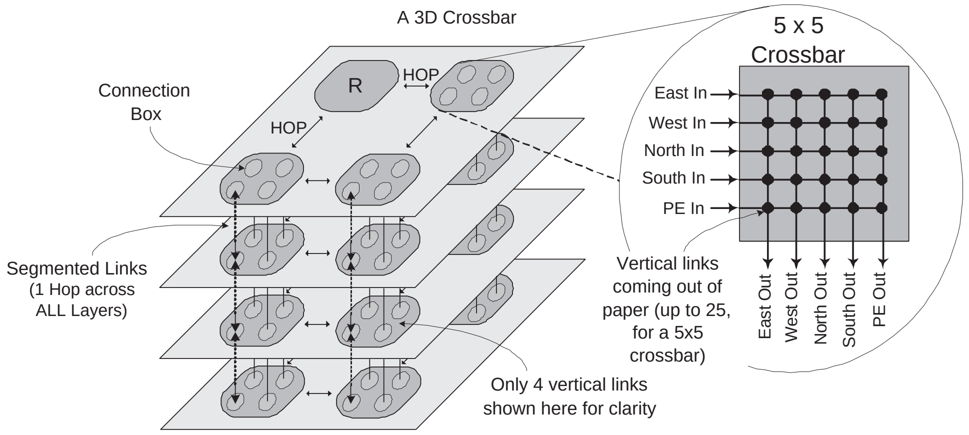
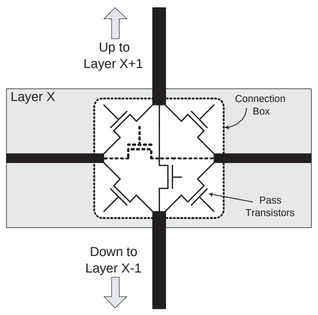
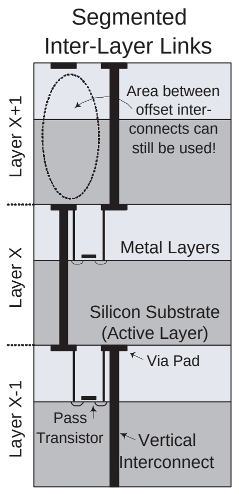

A 3D Crossbar
R
HOP
HOP
Connection Box
Segmented Links (1 Hop across ALL Layers)
5 x 5 Crossbar
East In
West In
North In
South In
PE In
East Out
West Out
North Out
South Out
PE Out
Vertical links coming out of paper (up to 25, for a 5x5 crossbar)
Only 4 vertical links shown here for clarity

Figure 7.5. NoC Routers with True 3D Crossbars



(a) Internal Details of a Connection Box (CB)


Segmented Inter-Layer Links
Layer X+1
Area between offset inter-connects can still be used!
Layer X
Metal Layers
Silicon Substrate (Active Layer)
Via Pad
Layer X-1
Pass Transistor
Vertical Interconnect

(b) Inter-layer Via Layout

Figure 7.6. Side View of the Inter-Layer Via Structure in a 3D Crossbar

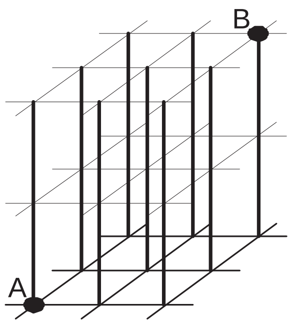


Figure 7.7. A 3D 3×3×3 Crossbar in Conceptual Form

It will be shown in Section 7.4 that adding a 128-bit vertical link, along with its associated control signals consumes only about 0.01 mm^2 of silicon real estate. However, despite this encouraging result, there is an opposite side to the coin which paints a rather bleak picture. Adding a large number of vertical links in a 3D crossbar to increase NoC connectivity results in increased path diversity. This translates into multiple possible paths between source and destination pairs. While this increased diversity may initially look like a positive attribute, it actually leads to a dramatic increase in the complexity of the central arbiter, which coordinates inter-layer communication in the 3D crossbar. The arbiter now needs to decide between a multitude of possible interconnections, and requires an excessive number of control signals to enable all these interconnections. Even if the arbiter functionality can be distributed to multiple smaller arbiters, then the coordination between these arbiters becomes complex and time-consuming. Alternatively, if dynamism is sacrificed in favor of static path assignments, the exploration space is still daunting in deciding how to efficiently assign those paths to each source-destination pair. Furthermore, a full 3D crossbar implies 25 (i.e. 5x5) Connection Boxes (see Figure 7.6(a)) per layer. A four-layer design would, therefore, require 100 CBs! Given that each CB consists of 6 transistors, the whole crossbar structure would need 600 control signals for the pass transistors alone! Such control and wiring complexity would most certainly dominate the whole operation of the NoC router. Pre-programming static control sequences for all possible input-output combinations would result in an oversize table/index; searching through such table would incur significant delays, as well as area and power overhead. The vast number of possible connections hinders the otherwise streamlined functionality of the switch. Note that the prevailing tendency in NoC router design is to minimize operational complexity in order to facilitate very short pipeline lengths and very high frequency. A full crossbar with its overwhelming control and coordination complexity poses a stark contrast to this frugal and highly efficient design methodology. Moreover, *our experimental results will show that the redundancy offered by the full connectivity is rarely utilized by real-world workloads, and is, in fact, design overkill.*

To understand the magnitude of the path diversity issue in a true 3D crossbar (as shown in Figure 7.7 for a 3×3×3 example), one can picture the 3D crossbar itself as a 3D Mesh network. For the 3D 3×3×3 crossbar of Figure 7.7, the number of minimal paths, k, between points A and B is given in [9] as

$$k = \begin{pmatrix} \Delta_x + \Delta_y + \Delta_z \\ \Delta_x \end{pmatrix} \begin{pmatrix} \Delta_y + \Delta_z \\ \Delta_y \end{pmatrix} = \frac{(\Delta_x + \Delta_y + \Delta_z)!}{\Delta_x! \Delta_y! \Delta_z!}, \tag{7.1}$$

where Δ_x, Δ_y and Δ_z are the numbers of hops separating A and B in the X, Y, and Z dimensions, respectively. In our example, $\Delta_x = \Delta_y = \Delta_z = 2$. Thus, the number of minimal paths between A and B is 90. For a 3D 4×4×4 crossbar, this number explodes to 1680. If non-minimal paths are also considered, then path diversity is practically unbounded [9].

Hence, given the tight latency and area constraints in NoC routers, vertical (inter-layer) arbitration should be kept as simple as possible. This can be achieved by using a limited amount of inter-layer links. The question is then: *how many links are enough? Our experiments in Section 5 demonstrate that anything beyond two links per 3D crossbar yields diminishing returns in terms of performance.*

7.3.4 A Partially-Connected 3D NoC Router Architecture

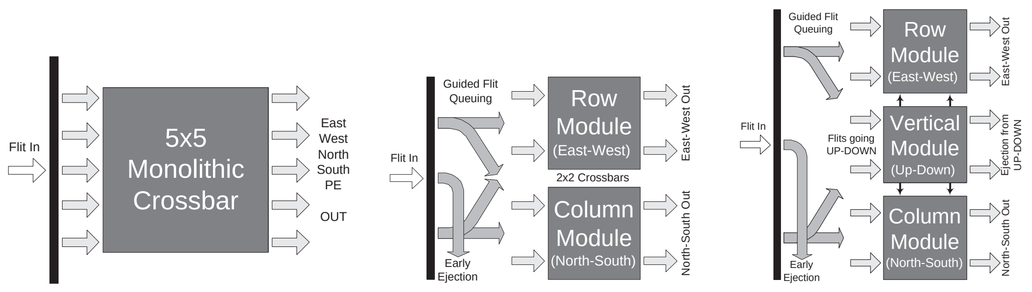
The scalability problem in vertical link arbitration highlighted in the previous sub-section dictates the use of a partially-connected 3D crossbar, i.e. a crossbar with a limited number of vertical links. The arbitration complexity can be further mitigated through the use of hierarchical arbiters. Two types of arbiters should be employed: intra-layer arbiters, which handle local requests from a single layer, and one global arbiter per vertical link to handle requests from all layers. This decoupling of arbitration policies can help parallelize tasks; while flits arbitrate locally in each layer, vertical arbitration decides on inter-layer traversal. These design directives were the fundamental drivers in our quest for a suitable 3D NoC implementation. As such, they form the cornerstones of our proposed architecture, which is described in detail in the following section.

7.4 The Proposed 3D Dimensionally-Decomposed (DimDe) NoC Router Architecture

The heart of a typical two-dimensional NoC router is a monolithic, 5×5 crossbar, as depicted abstractly in Figure 7.8(a). The five inputs/outputs correspond to the four cardinal directions and the connection from the local PE. The realization that the crossbar is a major contributor to the latency and area budgets of a router has fueled extensive research in optimized switch designs. Through the use of a preliminary switching process, known as Guided Flit Queuing [74], incoming traffic may be decomposed into two independent streams: (a) East-West traffic (i.e. packet movement in the X dimension), and (b) North-South traffic (i.e. packet movement in the Y dimension). This segregation of traffic flow allows the use of two smaller 2×2 crossbars and the isolation of the two flows in two independent router sub-modules, as shown conceptually in Figure 7.8(b). The resulting two compact modules are more area- and power-efficient, and provide better performance than the conventional monolithic approach.

Following this logic of traffic decomposition in orthogonal dimensions, we propose in this work the addition of a third information flow in the Z dimension (i.e. inter-layer communication). An additional module is now required to handle all traffic in the third dimension; this component is aptly called the Vertical Module. On the input side, packets are decomposed into the three dimensions (X, Y, and Z), and forwarded to the appropriate module. However, as previously mentioned, simply adding a third independent module cannot lead to a true 3D crossbar, because inter-layer traffic must be able to merge with intra-layer traffic upon arrival at the destination chip layer. A totally decoupled Vertical Module would force all packets arriving at a particular layer and wishing to continue traversal within that layer to be re-buffered and re-arbitrate for access to the Row/Column modules.

Hence, the Vertical Module must somehow fuse the Row and Column modules to allow movement of packets from the Vertical Module to the Row and Column Modules. An abstract view of the proposed 3D DimDe implementation is illustrated in Figure 7.8(c). The diagram clearly shows the Vertical Module linking with the Row and Column Modules. Also notice that the communication link is one-way, i.e. from the Vertical Module to the Row/Column Modules. There is no need for the Row/Column Modules to communicate with the Vertical Module, since intra-layer traffic wishing to change layer is pre-directed to the Vertical Module at the input of the router.



(a) Conventional 2D NoC Router Overview

(b) The 2D Row-Column (RoCo) Decoupled Router

(c) The Proposed 3D DimDe Router Architecture

Figure 7.8. Different NoC Router Switching Mechanisms

The streamlined nature of a dimensionally decomposed router lends itself perfectly for a 3D crossbar implementation. The simplicity and compactness of the smaller, distinct modules can be utilized to create a crossbar structure which extends into the third dimension without incurring prohibitive area and latency overhead. The high-level architectural overview of our proposed 3D DimDe router is shown in Figure 7.9. As illustrated in the figure, the gateway to different layers is facilitated by the inclusion of the third, Vertical Module.

Inter-Layer Link Length	Number of Repeaters	Delay
50 μm (Layer 1 to 2)	0	7.86 ps
100 μm (Layer 1 to 3)	0	19.05 ps
150 μm (Layer 1 to 4)	0	36.12 ps
150 μm (Layer 1 to 4)	1 (layer 3)	105.14 ps

Table 7.2. The Effect of Inter-Layer Distance on Propagation Delay

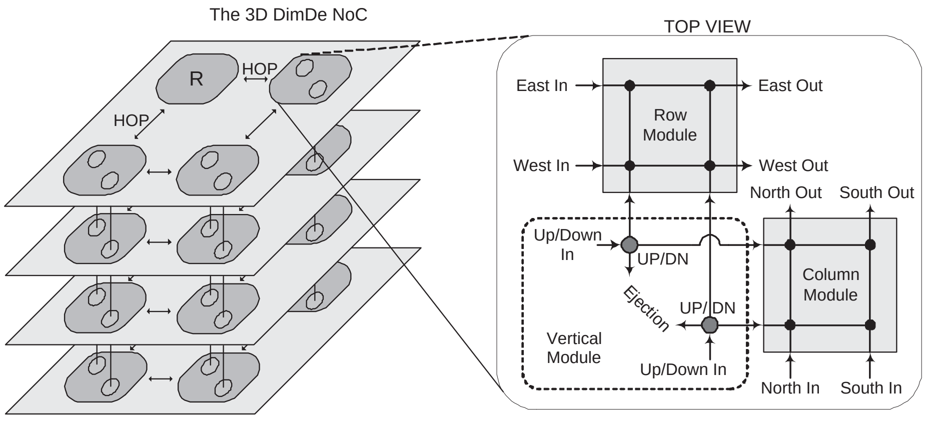


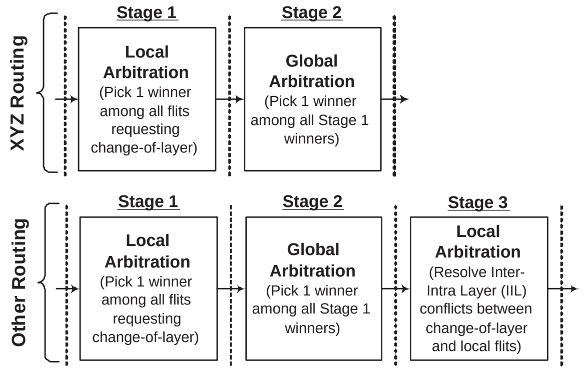
Figure 7.9. High-Level Overview of the 3D DimDe NoC Architecture

The 3D DimDe router uses vertical links which are segmented at the different device layers through the use of compact Connection Boxes (CB). Figure 7.6(a) shows a side view cross-section of such a CB. Each box consists of 5 pass transistors which can connect the vertical (inter-layer) links to the horizontal (intra-layer) links. The dotted transistor is not needed in DimDe, because the design was architected in such a way as to avoid the case where intra-layer communication needs to pass through a CB. The CB structure allows simultaneous transmission in two directions, e.g. a flit coming from layer X+1 and connecting to the left link of Layer X, and a flit coming from layer X-1 connecting to the right link of layer X (see Figure 7.6(a)). The inclusion of pass transistors in the data path adds delay and degrades the signal strength due to the associated voltage drop. However, this design decision is fully justified by the fact that inter-layer distances are, in fact, negligible. To investigate the effectiveness and integrity of this connection scheme, we laid out the physical design of the CB and simulated it in HSpice using the Predictive Technology Model (PTM) [153] at 70 nm technology and 1 V power supply. The latency results for 2, 3 and 4-layer distances are shown in Table 7.2. Evidently, even with a four-layer design (i.e. traversing four cascaded pass transistors), the delay is only 36.12 ps; this is a mere 1.8% of the 2 ns clock period (500 MHz) of the NoC router. In fact, the addition of repeaters will increase latency, because with such small wire lengths (around 50 μm per layer), the overall propagation delay is dominated by the gate delays and not the wiring delay. This effect is corroborated by the increased delay of 105.14 *ps* when using a single repeater, in Table 7.2.

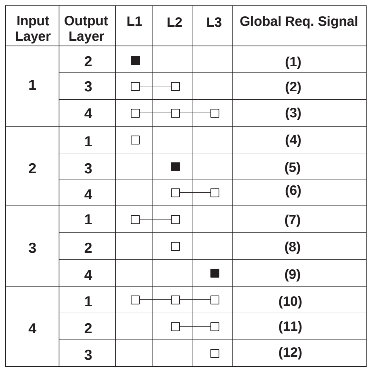
To indicate the fact that each vertical link in the proposed architecture is composed of a number of wires, we thereby refer to these links as bundles. The presence of a segmented wire bundle dictates the use of one central arbiter for each vertical bundle, which is assigned the task of controlling all traffic along the vertical link. If arbitration is carried out at a local level alone, then the benefit of concurrent communication along a single vertical bundle cannot be realized; each layer would simply be unaware of the connection requests of the other layers. Hence, a coordinating entity is required to monitor all requests for vertical transfer from all the layers and make an informed decision, which will favor simultaneous data transfer whenever possible. Concurrent communication increases the vertical bandwidth of the 3D chip. Given the resource-constrained nature of NoCs, however, the size and operational complexity of the central arbiter should be handled judiciously. The goal is not to create an overly elaborate mechanism which provides the best possible matches over several clock cycles. Our objective was to obtain reasonably intelligent matches within a single clock cycle.

To achieve this objective, we divided the arbitration for the vertical link into two stages, as shown at the top of Figure 7.10(a). The first stage is performed locally, within each layer. This stage arbitrates over all flits in a single layer which request a transfer to a different layer. Once a local winner is chosen, the local arbiter notifies the second stage of arbitration, which is performed globally. This global stage takes in all winning requests from each layer and decides on how the segmented link will be configured to accommodate the inter-layer transfer(s). The arbiter was designed in such a way as to realize the scenarios which are suitable for concurrent communication.

Figure 7.10(b) illustrates all possible requests to the global arbiter of a particular vertical bundle, assuming a 4-layer chip configuration using the deterministic XYZ routing. The designations L1, L2, and L3 indicate the different segments of the vertical bundle; L1 is the link between layers 1 and 2, L2 is the link between layers 2 and 3, and so on. As an example, let us assume that a flit in layer 1, which wants to go to layer 2, has won the local arbitration of layer 1; global request signal 1 (see Figure 7.10(b)) is asserted. Similarly, a flit in layer 2 wants to go to layer 3; global request signal 5 is asserted. Finally a flit in layer 3 wants to go to layer 4; global request signal 9 is asserted. The global arbiter is designed to recognize that the global request combination 1, 5, 9 (black boxes in Figure 7.10(b)) results in full concurrent communication between all participating layers. It will, therefore, grant all requests simultaneously. All combinations which favor simultaneous, non-overlapping communication are programmed into the global arbiter. If needed, these configurations can be given higher priority in the selection process. The arbiter can be placed on any layer, since the vertical distance to be traveled by the inter-layer control signals is negligible.


XYZ Routing
Stage 1
Local Arbitration
(Pick 1 winner among all flits requesting change-of-layer)
Stage 2
Global Arbitration
(Pick 1 winner among all Stage 1 winners)
Other Routing
Stage 1
Local Arbitration
(Pick 1 winner among all flits requesting change-of-layer)
Stage 2
Global Arbitration
(Pick 1 winner among all Stage 1 winners)
Stage 3
Local Arbitration
(Resolve Inter-Intra Layer (IIL) conflicts between change-of-layer and local flits)

(a) Vertical (Inter-Layer) Link Arbitration Stages



Input Layer	Output Layer	L1	L2	L3	Global Req. Signal
1	2	■			(1)
	3	□	□		(2)
	4	□	□	□	(3)
2	1	□			(4)
	3		■		(5)
	4		□	□	(6)
3	1	□	□		(7)
	2		□		(8)
	4			■	(9)
4	1	□	□	□	(10)
	2		□	□	(11)
	3			□	(12)

(b) 3D Global Arbitration Request Scenarios

Figure 7.10. Vertical (Inter-Layer) Link Arbitration Details

The aforementioned two arbitration stages suffice only if deterministic XYZ routing is used. In this case, a flit traveling in the vertical (i.e. Z) dimension will be ejected to the local PE upon arrival at the destination layer's router. If, however, a different routing algorithm is used, which allows flits coming from different layers to continue their traversal in the destination layer, then an additional local arbitration stage is required to handle conflicts between flits arriving from different layers and flits residing in the destination layer. The third arbitration stage, illustrated at the bottom of Figure 7.10(a), will take care of such Inter-Intra Layer (IIL) conflicts. The use of non-XYZ algorithms also complicates the request signals sent across different layers. It is no longer enough to merely indicate the destination layer; the output port designation on the destination layer also needs to be sent. IIL conflicts highlight the complexity involved in coordinating flit traversal in a 3D network environment. An example of the use of a non-XYZ routing algorithm is presented in Figure 7.11, which tracks the path of a flit traveling from Layer X to the eastern output of Layer X+1. In this case, the flit changes layer and continues traversal in a different layer.

Each vertical bundle in DimDe consists of a number of data wires (128 bits in this work), and a number of control wires to/from a central arbiter which coordinates flit movement in the vertical dimension. These control signals include: (a) Request signals from all layers to the central arbiter indicating the requested destination layer (and possibly output port, depending on the routing algorithm used), and the corresponding acknowledgement signals from the arbiter. (b) Enable signals from the arbiter to the pass transistors of the Connection Boxes of each layer spanned by the wire bundle. The total number of wires, w, in a vertical bundle is given by

$$w = \begin{cases} b + 2(n-1)^2 + 5(n-1) & , if\ XYZ\ algorithm\ is\ used \\ b + 2(n-1)^2 + 6(n-1) + 5(n-1) & , otherwise \end{cases}$$

where

$$\begin{aligned} b &= number\ of\ data\ bits/wires, \\ 2(n-1)^2 &= number\ of\ request/acknowledgement\ signals\ to/from\ the\ central\ arbiter \\ &\quad assuming\ an\ n-layer\ chip, \\ 6(n-1) &= number\ of\ additional\ signals\ sent\ to/from\ the\ arbiter\ for\ output\ port \\ &\quad designation\ (3-bit\ designation\ for\ the\ four\ possible\ output\ ports\ and \\ &\quad the\ ejection\ port)\ when\ a\ non-XYZ\ routing\ algorithm\ is\ employed, \\ 5(n-1) &= number\ of\ enable\ signals\ for\ the\ pass\ transistors\ of\ the\ CB\ of\ each\ layer. \end{aligned}$$

Assuming a 4-layer configuration ($n = 4$), XYZ routing, and 128 data bits (i.e. $b = 128$), the number of wires in a vertical bundle, w, is 161. Based on the square-like layout of Figure 7.1, the area consumed by the bundle is around 10,000 $\mu m^2 = 0.01$ mm^2. This amounts to a vertical via density of around 1.5 million individual wires per cm^2. This result illustrates the fact that increasing the number of vertical vias is, in fact, feasible in terms of area consumption by the wires themselves. However, as explained in Section 7.3.4, adding extra vertical bundles in the 3D crossbar is prohibitive in terms of arbitration complexity; the area, power and latency increases incurred by a highly-complex arbitration scheme negate any advantages provided by the increased number of inter-layer bundles. Furthermore, it will be demonstrated later on that increasing the number of inter-layer bundles yields rapidly diminishing returns in terms of performance gain under both synthetic and real workloads.

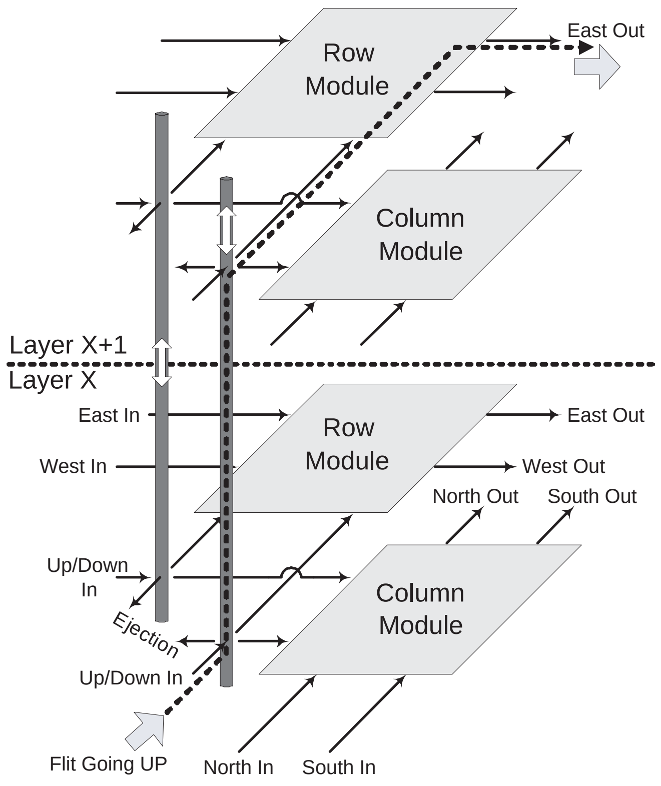


Figure 7.11. An Example of a Non-XYZ Routing Algorithm

A detailed view of the proposed 3D DimDe architecture is shown in Figure 7.12. DimDe employs Guided Flit Queuing [74] to guide incoming flits to an appropriate Path Set (PS). Guided Flit Queuing is a preliminary switching operation at the input of the router which utilizes the look-ahead routing information present in incoming header flits. This information denotes the requested output path; thus, incoming traffic can be decomposed into the X, Y, and Z dimensions. The Vertical Module adds two extra path sets to the 2D implementation. One path set is used by incoming flits from the East-West (intra-layer) dimension, and the other for flits from the North-South dimension. Just like Guided Flit Queuing, the Early Ejection Mechanism [74] uses the look-ahead routing information to identify

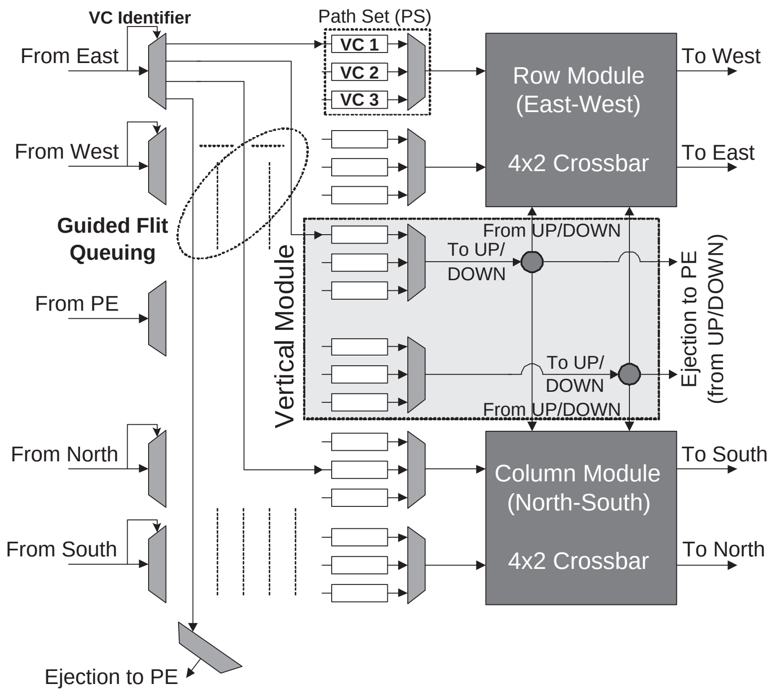


Figure 7.12. Architectural Detail of the Proposed 3D DimDe NoC Router

packets which need to be ejected to the local PE. This enables such flits to bypass the destination router and be directly ejected to the NIC. The Vertical Module consists of two bidirectional vertical bundles, one for each of the two path sets. Note that the number of vertical bundles can be varied from four to one. Each vertical link has one input connection and three output connections on each layer. The input connection comes from the associated path set's MUX (see dark box in the middle of Figure 7.12). The three output connections are as follows: (1) One connection to the Row Module Crossbar for flits which arrive from other layers and need to continue traversal in the East-West dimension of the current layer. (2) One connection to the Column Module crossbar for flits which need to continue in the North-South dimension of the current layer. (3) One connection for ejection to the Network Interface Controller (NIC) of the local PE. This configuration implies that the Row and Module crossbars need to grow in size from 2×2 in the 2D case to 4×2 in DimDe to accommodate the two additional connections from the two vertical links. Despite this increase in size, two 4×2 crossbars are still substantially smaller than a single monolithic 6×6 or 7×7 crossbar, as illustrated in Table 7.1. Once again, it is precisely for this reason that we chose to use this architecture in our 3D NoC implementation.

The Vertical Module of the proposed DimDe router uses two Path Sets to group the available Virtual Channels. As shown in Figure 7.13, the DimDe router requires 5 VCs for correct functionality under a deterministic, deadlock-free algorithm: one VC for injection from each of the four incoming directions, and one for injection from the local PE. The sixth VC can be used as a drain channel for deadlock recovery under adaptive routing algorithms. Moreover, depending on the algorithm used, additional VCs can be added to the two Vertical Module path sets to ensure deadlock freedom. These drain VCs need to operate on deadlock-free algorithms to guarantee deadlock breakup [154]. In this work, we concentrated on deterministic XYZ and ZXY algorithms as a proof of concept of the proposed architecture. Since these algorithms are inherently deadlock-free, the sixth VC buffer was used as an additional injection VC from the local PE.

As previously explained in Section 7.2.2, thermal issues are of utmost importance in 3D chips. Stacking several active layers with minimal distance in-between favors the creation of hotspots. From a 3D NoC perspective, it was important to investigate the effect of high temperature on the propagation delay of the signals on the vertical (inter-layer) interconnects. To that extend, the propagation delay between the layers was modeled as an RC ladder (Figure 7.14(b)) to accurately capture the distributed resistance, capacitance, and temperature variations along the inter-strata vias. The resistance of metals is affected by temperature, and it was modeled using equations from [155]. Assuming a T_{Layer1} temperature of 85 °C and a fixed linear temperature gradient between each layer, the propagation delay of these vias was simulated in HSpice with the required temperature annotations. Even in the worst case of a 10 °C temperature increase per layer for 8 layers, the total propagation delay from the lowest to the highest layer was only 0.11 ps and, therefore, considered inconsequential for our work. The results of the thermal analyses are summarized in Figure 7.14(a).

7.5 Performance Evaluation

In this section, we present simulation-based performance evaluation of our architecture, a generic 2D router architecture, a 3D Symmetric NoC design, the 3D

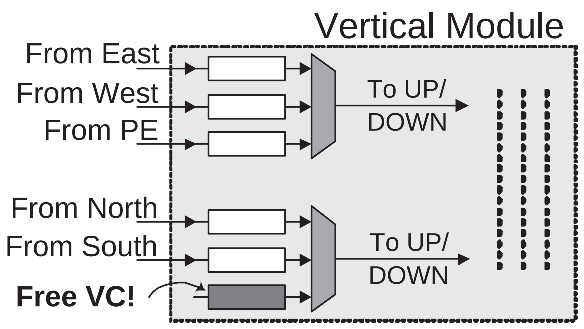
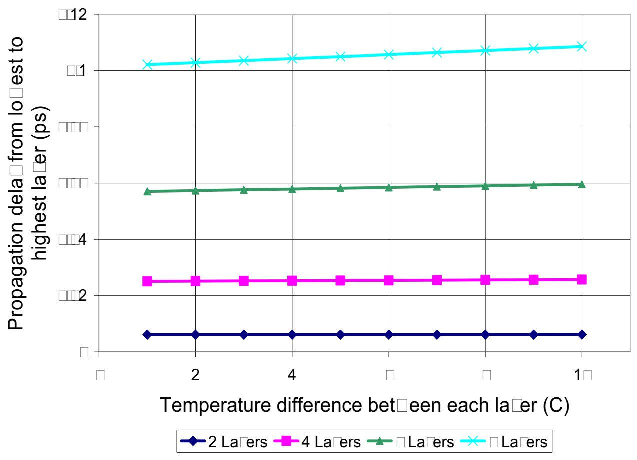
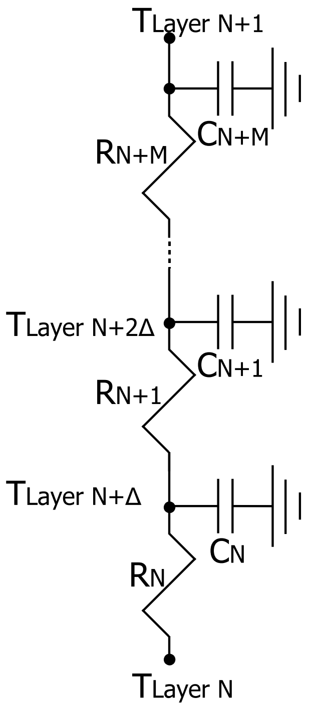


Figure 7.13. Virtual Channel Assignments in the Vertical Module of DimDe



(a) Inter-Layer Propagation Delay vs. Temperature



(b) Modeling of Temperature Effect on Propagation Delay

Figure 7.14. Thermal Effects on Inter-Layer Propagation Delay

NoC-Bus Hybrid architecture, and the Full 3D Crossbar implementation, in terms of network latency, throughput and power consumption under various traffic patterns. Our experimental methodology is followed by the experimental results.

7.5.1 Simulation Platform

A double-faceted evaluation environment was implemented in order to conduct a detailed evaluation of the router architectures analyzed in this paper: (a) A cycle-accurate stand-alone 3D NoC simulator was developed, which accurately models the routers, the interconnection links and vertical pillars, as well as all the architectural features of the various NoC architectures under investigation. The simulator was built by augmenting an existing 2D NoC simulator and models each individual component within the router architecture, allowing for detailed analysis of component utilizations and flit flow through the network. The activity factor of each component is used for analyzing power consumption within the network. In addition to the network-specific parameters, our simulator accepts hardware parameters such as power consumption (dynamic and leakage) for each component and overall clock frequency. This leg of the simulation process examines the behavior of all the architectures under synthetic workloads.

(b) To provide a more diversified simulation environment, we also implemented a detailed trace-driven cycle-accurate hybrid NoC/cache simulator for CMP architectures. The memory hierarchy implemented is governed by a two-level directory cache coherence protocol. Each core has a private write-back L1 cache (split L1 I and D cache, 64 KB, 2-way, 3-cycle access). The L2 cache is shared among all cores and split into banks (32 banks, 512 KB each for a total of 16 MB, 6-cycle bank access). An underlying NoC model connects the L2 banks. The L1/L2 block size is 64 B. Our coherence model includes a MESI-based protocol with distributed directories, with each L2 bank maintaining its own local directory. The simulated memory hierarchy mimics SNUCA [38]. The sets are statically placed in the banks depending on the low order bits of the address tags. The network timing model simulates all kinds of messages: invalidates, requests, replies, write-backs, and acknowledgements. The interconnect model is the same as (a) above. The off-chip memory is a 4 GB DRAM with a 260-cycle access time.

Detailed instruction traces of four commercial server workloads were used: (1) TPC-C [156], a database benchmark for online transaction processing (OLTP), (2) SAP [157], a sales and distribution benchmark, and (3) SJBB [158] and (4) SJAS [159], two Java-based server benchmarks. The traces – collected from multiprocessor server configurations at Intel Corporation – were then run through our NoC/cache hybrid simulator to measure network statistics. Additionally, a second set of memory traces was generated by executing programs from SPLASH [160], a suite of parallel scientific benchmarks, on the Simics full system simulator [161]. Specifically, barnes, ocean, water-nsquared (wns), water-spatial (wsp), lu, and radiosity (rad) were used. The baseline configuration is the Solaris 9 Operating system running on eight UltraSPARC III cores. Benchmarks execute 16 parallel threads. Again, the number of banks for the L2 shared cache is 32. Thus, 32 nodes are present in the NoC network, 8 of which are also CPU nodes.

The proposed components of the 3D router architectures, and a generic two-stage 5-port router architecture were implemented in structural Register-Transfer Level (RTL) Verilog and then synthesized in Synopsys Design Compiler using a TSMC 90 *nm* standard cell library. The vertical interconnects were modeled as 2D wires with equivalent resistance and capacitance. The resulting designs operate at a supply voltage of 1 V and a clock speed of 500 MHz. Both dynamic and leakage power estimates were extracted from the synthesized router implementation. These power numbers were then imported into our cycle-accurate simulation environment and used to trace the power profile of the entire on-chip network.

7.5.2 Performance Results

The proposed 3D DimDe design was compared against four other router architectures (2D NoC, 3D Symmetric NoC, 3D NoC-Bus Hybrid, and a Full 3D Crossbar configuration) using our cycle-accurate simulation environment. Our definition of a "full" 3D crossbar implies that all connection points inside the 2D 5$\times$5 crossbar (i.e. 25 links) extend into the third (i.e. vertical) dimension. In both simulation phases, two deterministic routing algorithms (XYZ routing and ZXY routing) were used to measure the average network latency, throughput, and power consumption in all experiments. For the synthetic workload simulation phase (described in part (a) of Section 7.5.1), all architectures under investigation were evaluated using a regular mesh network with 64 nodes. In the 3D designs, 4 layers were used, each with 16 nodes (4x4). Wormhole routing based on virtual-channel flow control was employed in all cases. To ensure fairness, all architectures under test had 3 VCs per input port, and a total buffer space of 80 flits per node. Each simulation consists of two phases: a warm-up phase of 20,000 packet injections, followed by the main phase which injects a further one million packets. Each packet consists of four 128-bit flits. The simulation terminates when all packets are received at the destination nodes. Uniform, matrix-transpose (dimension reversal) [162] and self-similar traffic patterns were used.

For the real workload simulation phase (described in part (b) of Section 7.5.1), the 32 L2 cache banks (nodes) were "folded" into 4 layers, with each layer holding 8 banks (4x2). The 8 CPUs were also split into 2 CPUs/layer. The commercial workloads were simulated for 10,000 transactions per thread, whereas the scientific workloads were simulated for 100 million instructions per core upon commencement of the parallel phase of the code. Data messages were 5-flit packets (64 B cache-line plus network overhead), while control messages were single-flit packets.

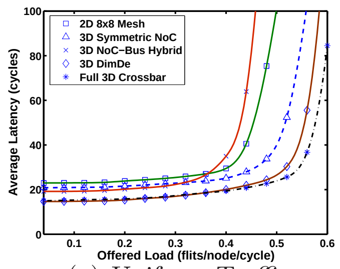
The 3D DimDe architecture design exploration provided different options for the number of pillars in the Vertical Module. Since we are using 2×2 crossbars as the basic building blocks, four pillars would provide a complete crossbar connection, while a single pillar would provide a segmented bus connection. As previously mentioned, the caveat is that more vertical pillars offer more path diversity and complicate the arbiter design. Hence, the number of pillars should be decided based on the performance, energy and area tradeoffs. Our experiments (not shown here due to space considerations) suggest that the two-pillar DimDe design provides the best compromise in terms of performance, area and energy behavior. Therefore, in the rest of the evaluations, we use the two-pillar DimDe architecture (as shown in Figure 7.12). Moreover, to validate our assertion that anything more than two vertical bundles would yield diminishing returns, we will compare our design to a full 3D crossbar configuration (Section 7.3.4) with 25 vertical bundles. In all experiments, *the Full 3D crossbar was assumed to complete its configuration in a single clock cycle. Therefore, all results for this Full 3D Crossbar design will be very optimistic. However, despite discounting the complexity of the control and arbitration logic of the Full 3D crossbar, the proposed DimDe router will still be able to achieve comparable performance.*

The latency and throughput results of all five architectures for various synthetic traffic patterns (i.e. phase 1 of our simulation experiments) are illustrated in Figures 7.15 through 7.17. It can be observed that the proposed DimDe design consistently remains within 5% (on average) of the ideal Full 3D Crossbar's performance, while providing much lower latency and saturating at much higher workloads than the remaining architectures. Compared to the DimDe design, the 3D Symmetric topology suffers from the additional router delay at each inter-layer hop. At low loads (e.g., up to 20% for all traffic patterns with XYZ routing), the NoC-Bus Hybrid provides lower latency compared to the 3D Symmetric NoC as it benefits from the single hop vertical communication. As the load increases, the NoC-Bus Hybrid Architecture exhibits the worst latency and throughput for all traffic patterns (even worse than the 2D topology) as the bus saturates first with higher workload. Consequently, the 3D NoC-Bus Hybrid may be suitable only for 3D architectures where the traffic is mostly confined to the 2D strata and the load on the vertical links is sparse. Clearly, *the proposed 3D DimDe router outperforms the other three designs in all traffic patterns, and it achieves performance very close to that of a full crossbar, using only two (instead of 25) vertical bundles. This soundly resonates our assertions that a full 3D crossbar is design overkill in terms of performance enhancement*.

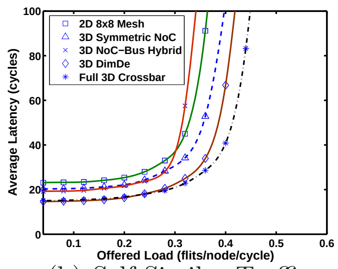
Figure 7.16 shows the latency results with ZXY routing. As ZXY routing incurs a two-cycle arbitration delay (see Figure 7.10(a)), in contrast to the single-cycle arbitration of XYZ routing, both the 3D DimDe and Full 3D Crossbar routers incur higher latency compared to the XYZ routing. In terms of the throughput behavior (Figure 7.17), the DimDe architecture provides 18% average improvement over the other designs, while remaining within around 3% of the Full 3D Crossbar's throughput.

Figures 7.18 and 7.19 show the results of phase 2 of our simulation experiments, i.e. real commercial and scientific workloads in an 8-CPU CMP environment. Figure 7.18 depicts the average network latency for the four 3D architectures under test. We do not show the 2D results, since the significantly larger hop count in the 2D case naturally leads to substantially worse results compared to all 3D architectures. Clearly, the proposed 3D DimDe design outperforms all designs except the Full 3D Crossbar. DimDe provides an average improvement of 27% over the 3D Symmetric and 3D NoC-Bus Hybrid designs, and remains within 4% of the Full 3D Crossbar's performance. However, a more complete picture is painted in Figure 7.19, which compares the Energy-Delay Product (EDP) of all the architectures.

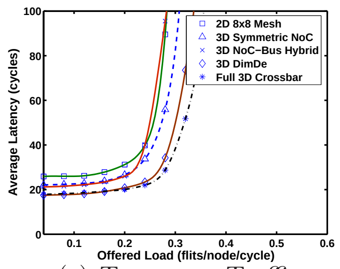
This metric is, in fact, more meaningful since it accounts for both performance and power consumption. Here, the efficiency of the proposed 3D DimDe design shines through. DimDe significantly outperforms all other designs, including the ideal Full 3D Crossbar, by 26% on average. These results are a testament to the holistic efficiency of the DimDe architecture. *Through the decomposition of incoming traffic into smaller components, the use of a simple, partially-connected 3D crossbar, and reduced arbitration complexity, DimDe can outperform even the optimistic results of a full (i.e. 25 vertical bundles) crossbar structure. This result is of profound significance, because it shows that increasing inter-layer links arbitrarily increases design complexity and overhead without tangible performance benefits.*



(a) Uniform Traffic

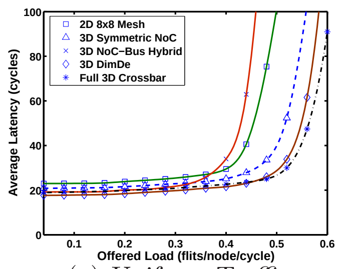


(b) Self-Similar Traffic

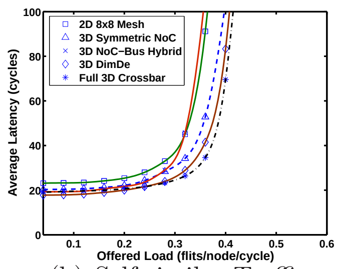


(c) Transpose Traffic

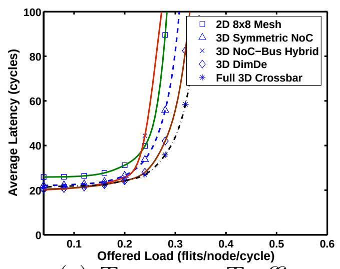
Figure 7.15. Average Latency with various Synthetic Traffic Patterns (XYZ routing)



(a) Uniform Traffic

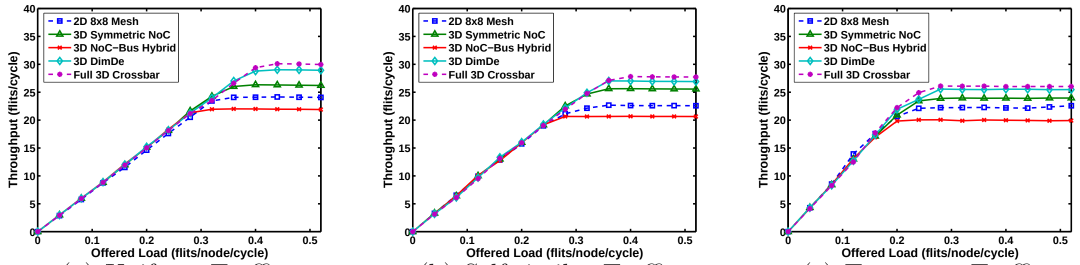


(b) Self-similar Traffic



(c) Transpose Traffic

Figure 7.16. Average Latency with various Synthetic Traffic Patterns (ZXY Routing)



(a) Uniform Traffic (b) Self-similar Traffic (c) Transpose Traffic

Figure 7.17. Throughput with various Synthetic Traffic Patterns (XYZ Routing)

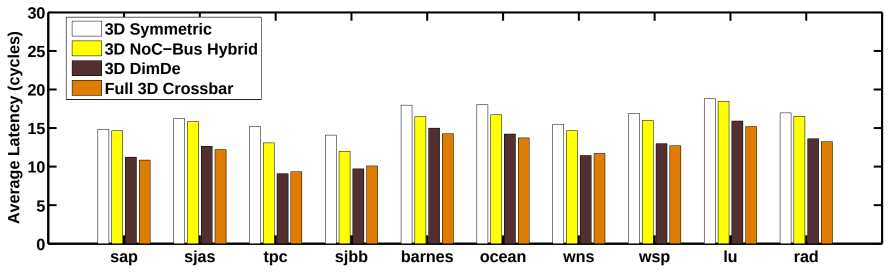


Figure 7.18. Average Latency with various Commercial and Scientific Workloads

7.6 Conclusions

3D technology is envisioned to provide a performance-rich, area- and energy-efficient, and temperature-aware design space for multicore/SoC architectures. In this context, the on-chip interconnect in a 3D setting will play a crucial role in optimizing the performance, area, energy and thermal behaviors. In this Chap-

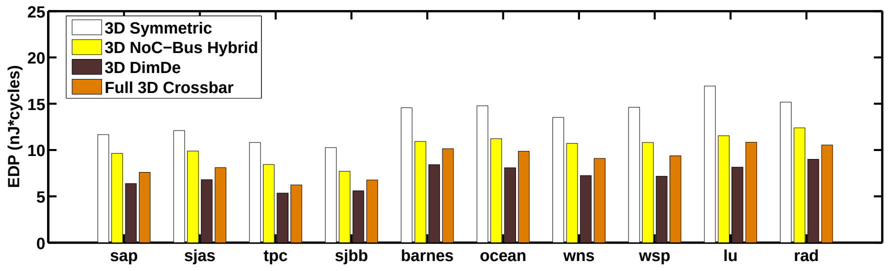


Figure 7.19. Energy-Delay Product (EDP) with various Commercial and Scientific Workloads

ter, we have explored several design options for 3D NoCs, specifically focusing on the inter-strata communication. Three possible designs that include a simple bus for the vertical connection, a symmetric 3D hop-by-hop topology, and a true 3D crossbar architecture are investigated. The proposed 3D architecture, called the 3D DimDe router, supports two vertical interconnects to achieve a balance between the path diversity and high bandwidth offered by a full 3D crossbar and the simplicity of a bus. DimDe supports a true 3D crossbar structure spanning all layers of the chip and fusing them into a single router entity. We have investigated the detailed micro-architectural implications of the design, which include the feasibility of the inter-strata vertical wire layout, arbitration mechanism, and virtual channel support for providing deadlock-free routing. The design has been implemented in structural Verilog and synthesized using a TSMC 90 *nm* standard cell library to analyze the area, energy, and thermal behaviors. It has been shown that it is possible to implement a hierarchical two-stage vertical arbitration mechanism.

To ensure a comprehensive evaluation environment, we utilized a double-faceted simulation process to expose all designs to several traffic patterns, including request/reply memory traffic. Phase 1 of the simulation used a stand-alone, cycle-accurate NoC simulator running synthetic workloads, while Phase 2 used a hybrid NoC/cache simulator running a variety of commercial and scientific workloads within the context of a multi-bank NUCA L2 cache in an 8-CPU CMP environment. In both cases, the proposed DimDe design was demonstrated to offer average latency and throughput improvements of more than 20% over the other 3D architectures, while remaining within 5% of the full 3D crossbar performance. More importantly, the DimDe architecture outperforms all other designs, including the full 3D crossbar, by an average of 26% in terms of the Energy-Delay Product (EDP). *One of the most important contributions of this work is the clear indication that arbitrarily adding vertical links in a 3D NoC router yields diminishing returns in terms of performance, while increasing control and arbitration complexity.*

3D integration presents the interconnect designer with several new challenges. In the future, we plan to investigate the design of a pipelined arbitration scheme to support adaptive routing within the context of fault tolerance and load balancing.

Chapter 8

Conclusions

With the advent of deep-submicron technology, single chip system design is becoming a reality. However, with the transistor counts reaching into the billions and chip densities increasing, novel solutions are needed for arising architectural issues. One key area that will effect the performance, design, verification, and manufacturing of future System-on-Chip (SoC) or multicore architectures is the interconnection network. This research was aimed at developing a comprehensive design paradigm for exploring the on-chip interconnect design consisting of five components important to NoC architectures.

8.1 Summary of Contribution

In the first part of this research we have summarized SoC design challenges and analyzed NoC design issues and alternatives at different layers of abstraction. Because of the ever increasing circuit complexity, the generic architecture needs to adapt into a flexible and parallel platform. As a joint project the designs were implemented in structural Register-Transfer Level (RTL) Verilog and then synthesized in Synopsys Design Compiler using a TSMC 90 nm and 70nm cell library. Current trends in system design methodology are moving towards co-design of mixed hardware/software system targeting multiprocessor System-on-Chip. Thus, modularity, flexibility and scalability are required to have an efficient multiprocessor design flow, which are feasible and applicable to large application fields. The most common way to achieve modularity is to separate the sub-system communications from the behavior when partitioning a system. Therefore, on-chip communication requirement is becoming a dominant factor, shifting the paradigm from computation centric to communication centric design (in terms of area, performance, power and reliability). Most prior studies have not considered a comprehensive approach to the design and analysis of on-chip interconnects encompassing the three design metrics - performance, power, and reliability.

The rapidly increasing use of SoC architectures has also accentuated the need for efficient on-chip communication infrastructures since they require ultra-low latency and an overall enhancement of performance. In the second work of our research, we proposed one type of solution, in which a low-latency on-chip router architecture supporting adaptive path-sensitive mechanisms, was shown to be able to minimize average packet latency by intelligent path selection and reduced switching activities. This architecture reduced the size of the control logic, decreasing the number of connections and switching points necessary with the conventional switch fabric. We have shown that this technique allows for significant energy savings in a chip's tight power budget. Furthermore, we evaluated how network latencies influence on energy consumption of NoC architectures implemented in deep sub-micron technologies. We have accounted for individual router component utilizations to precisely estimate the dynamic and leakage envelope of each router throughout the simulation. This allows for a more realistic estimation when handling different flit types. For example, a header flit requires processing by the routing, computation, and virtual channel allocation units, while a data or tail flit do not. These attributes are accurately reflected in my power estimation model since they all effect different component utilizations during the simulation and, as a result, the energy consumption.

While the design space remains large, leaving many design alternatives unexplored, the ability to precisely and efficiently provide analysis of an NoC's performance, fault-tolerance, and energy behavior becomes another fundamental aspect of NoC design. While most prior on-chip interconnect analyses are based on time-consuming simulation models, in order to provide fast performance estimates during the design cycle, we developed an accurate queuing-theory-based analytical model for capturing performance and energy behavior at the granularity of individual on-chip network sub-modules, as our third part of this thesis. The model developed here quantified the overall power consumption by capturing the utilization of different components and their corresponding energy consumptions. By integrating performance, power, and reliability models the analytical model was further able to evaluate multi-objective tradeoffs. In addition, current network design will not be able to provide reliable communication due to the increased role of crashes and failures. We have explored error detection and correction mechanisms that provide different energy-reliability- performance tradeoffs and extend our model to evaluate the on-chip network in the presence of these error protection schemes. Our reliability exploration culminates with the introduction of an array of transient fault protection techniques, both architectural and algorithmic, to tackle reliability issues within the router's individual hardware components. We proposed a complete solution safeguarding against both the traditional link faults and internal router upsets, without incurring any significant latency, area and power overhead.

In the fourth part of our research, we proposed a novel fine-grained modular router architecture, called the Row-Column (RoCo) Decoupled Router, towards more comprehensive framework of designing low-latency, energy-efficient and reliable on-chip communication networks. The proposed RoCo router has a number of features that make it distinct compared to the earlier designs. First, it uses two smaller 2×2 crossbars instead of a larger 5×5 crossbar that is traditionally used for 2D mesh networks. Second, it uses a path-sensitive buffering scheme, where, the virtual channels are divided into four sets to support dedicated row and column routing in the two crossbars. These two features along with early ejection, mirrored allocation, look-ahead routing and speculative path selection help in reducing the contention. Third, unlike most earlier designs, we showed how deterministic (XY) routing, XY-YX routing and adaptive routing can be supported in this architecture. Fourth, because of the modular design, we demonstrated how different types of faults such as VA, SA, and crossbar failures can be handled with graceful degradation, thereby providing better fault-tolerance compared to earlier designs. The novelty of this approach is that operation of the router can continue in the presence of a faulty component through resource sharing. The substitution of defective elements by healthy ones elsewhere in the system provides a kind of virtual recycling bin, where functional components can be reused in other parts of the implementation should the need arise. Our proposed scheme avoids the more traditional approach in fault-tolerance, which resorts to replication of resources. Silicon real-estate and energy are at a premium in on-chip applications, thus necessitating the efficient re-use of existing resources. All these techniques were used to create a very efficient and resilient system, while incurring no significant area, latency, or power overhead. In addition, while all prior NoC studies have analyzed at best two of the three parameters, such as energy-delay product, we introduced a comprehensive parameter, called PEF, for analyzing the performance, energy and fault-tolerance attributes of NoC architectures. The inter-dependence between speed, power and fault-tolerance highlights the importance of a metric which can identify the best tradeoffs between these three competing traits. Hence, we proposed the Performance, Energy and Fault-tolerance (PEF) metric, as a comprehensive parameter that reflects the correlation between the three desired design goals. In this work, we could develop a design paradigm that provided a comprehensive framework for exploring high-performance, reliable, and energy-efficient SoC/muliticore NoC architectures.

In the final work, we explored the design of a three dimensional (3D) crossbar-style, NoC for upcoming 3D VLSI technology. Much like multi-storey buildings in densely packed metropolises, three-dimensional (3D) chip structures are envisioned as a viable solution to skyrocketing transistor densities and burgeoning die sizes in multi-core architectures. Partitioning a larger die into smaller segments and then stacking them in a 3D fashion can significantly reduce latency and energy consumption. Such benefits emanate from the notion that inter-wafer distances are negligible compared to intra-wafer distances. This attribute substantially reduces global wiring length in 3D chips. The work in this thesis integrated the NoC design into a 3D setting. While NoCs have been studied extensively in the 2D realm, the microarchitectural ramifications of moving into the third dimension have yet to be fully explored. We presented a detailed exploration of inter-strata communication architectures in 3D NoCs. Three design options are investigated; a simple bus-based inter-wafer connection, a hop-by-hop standard 3D design, and a full 3D crossbar implementation. This exploration provided great insight into the balance between path diversity and bandwidth benefits using real commercial and scientific benchmarks in the simulation testbed. In this context, we proposed a novel partially-connected 3D crossbar structure, the 3D Dimensionally-Decomposed (DimDe) Router, which provides a good tradeoff between circuit complexity and performance benefits.

To conclude, this research considered the three objective functions, performance, energy, and fault-tolerance, to lead an investigation into the NoC design space. As a result, we were able to develop a variety of novel techniques that will enable computer architects to be able to design scalable, high-performance, energy-efficient, and reliable, on-chip communication networks for future SoC/multicore systems.

8.2 Future Research Directions

Design and analysis of System-on-Chip (SoC) architectures incorporating hundreds of functional units to solve real-world problems is an emerging and exciting research field. This research is likely to have a significant influence on the design of next generation multicore/SoC architectures. The results from this work help in the advancement of research areas within multicore computing which is expected to be the main design paradigm for future high performance architectures.

Real-time processing of video and audio streams are computationally demanding and require a high quality of service. SoC/multicore architectures are an ideal solution for these types of tasks, especially in mobile computing environments. We plan to explore a set of dedicated processors for handling mission critical tasks including next generation wireless Internet traffic, encryption/decryption, real time scheduling, and high definition video. We will analyze parallel processing and stream processing, and will develop a global hierarchy and local pipeline organization as follows: (i) Partition the overall functionality into several parallel tasks. (ii) Performance characterization of inter-processor communications. (iii) Pipelining multimedia platforms through the use of sub-connections; the performance of mobile multimedia SoC depends on efficiently streaming large amounts of variable data through the devices, and the data processing is pipelined by the dedicated processors.

Three-dimensional integrated circuits (3D ICs) are attractive options for overcoming the barriers in interconnect scaling, offering an opportunity to continue the CMOS performance trend. Design decisions of 3D interconnects is closely intertwined with the design of the architecture. I plan to explore the influence of alternate design styles for stacking different components (processors, memory blocks, etc...) on top of each other and then completely redesign individual components for multiple layers to improve communication traffic across multiple layers and within a single layer.

Bibliography

[1] Intel (2006) "Intel Develops Tera-Scale Research Chips," Http://www.intel.com/pressroom/archive/releases/20060926corp_b.htm.

[2] "International Technology Roadmap for Semiconductors," Http://public.itrs.net/.

[3] Edenfeld, D., A. B. Kahng, M. Rodgers, and Y. Zorian (2004) "2003 Technology Roadmap for SeSmiconductors," *IEEE Computer*, **37**(1), pp. 47–56.

[4] Dally, W. J. and B. Towles (2001) "Route Packets, Not Wires: On-Chip Interconnection Networks," in *Proceedings of the 38th Design Automation Conference*.

[5] Benini, L. and G. D. Micheli (2002) "Networks on Chips: A New SoC Paradigm," *IEEE Computer*, **35**(1), pp. 70–78.

[6] Benini, L. and G. D. Micheli. (2002) "Networks on Chip: A New Paradigm for Systems on Chip Design," in *Proceedings of Design Automation and Test Conference in Europe*.

[7] Ho, R., K. W. Mai, and M. A. Horowitz (2001) "The Future of Wires," in *Proceedings of the IEEE*, pp. 490–504.

[8] Cao, Y., C. Hu, A. B. Kahng, S. Muddu, D. Stroobandt, and D. Sylvester (2000) "Effects of Global Interconnect Optimizations on Performance Estimation of Deep Submicron Designs," in *International Conference on Computer-Aided Design*, pp. 56–61.

[9] Dally, W. J. and B. Towles (2003) *Principles and Practices of Interconnection Networks*, Morgan Kaufmann.

[10] Bhuyan, L. N. and X. Zhang (eds.) (1994) *Mutliprocessor Performance Measurement and Evaluation*, IEEE Computer Socitey Press, (Tutorial).

[11] Bhuyan, L. N., Q. Yang, and D. P. Agrawal (1989) "Performance of Multiprocessor Interconnection Networks," *IEEE Computer*, pp. 25–37.

[12] Tzeng, N.-F. (1991) "Alleviating the Impact of Tree Saturation in Multistage Interconnection Networks," *Journal of Parallel and Distributed Computing*, pp. 107–117.

[13] Varma, A. and C. S. Raghavendra (1993) *Interconnection Networks for Multiprocessors and Multicomputers*, IEEE Computer Society Press.

[14] Feng, T. Y. (1981) " A Survey of Interconnection Networks," *IEEE Computer*, **14**.

[15] Raghunathan, V., M. B. Srivastava, and R. K. Gupta (2003) "Energy-Aware System Design: A Survey of Techniques for Energy Efficient On-Chip Communication," in *Proceedings of the 40th conference on Design automation*, pp. 900–905.

[16] Taylor, M. B., J. Kim, J. Miller, D. Wentzlaff, F. Ghodrat, B. Greenwald, H. Hoffmann, P. Johnson, J.-W. Lee, W. Lee, A. Ma, A. Saraf, M. Seneski, N. Shnidman, V. Strumpen, M. Frank, S. Amarasinghe, and A. Agarwal (2002) "The Raw Microprocessor: A Computational Fabric for Software Circuits and General Purpose Programs," *IEEE Micro*, **22**(2), pp. 25–35.

[17] Simunic, T. and S. Boyd (2002) "Managing Power Consumption in Networks on Chips," in *Proceedings of Design, Automation and Test in Europe Conference.*

[18] Huh, J., S. W. Keckler, and D. Burger. (2001) "Exploring the Design Space of Future CMPs," in *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques (PACT).*

[19] Moritz, C. A., D. Yeung, and A. Agarwal (2001) "SimpleFit: A Framework for Analyzing Design Trade-Offs in Raw Architectures," *IEEE TPDS*, **12**(7), pp. 730–742.

[20] Wang, H.-S., L.-S. Peh, and S. Malik (2003) "Power-Driven Design of Router Microarchitectures in On-Chip Networks," in *Proceedings of the 36th MICRO.*

[21] Shang, L., L.-S. Peh, and N. K. Jha (2003) "Dynamic Voltage Scaling with Links for Power Optimization of Interconnection Networks," in *Proc. HPCA.*

[22] Wang, H.-S., L.-S. Peh, and S. Malik (2002) "A Power Model for Routers: Modeling Alpha 21364 and InfiniBand Routers," in *Proc. Hot Interconnect 10.*

[23] Lettieri, P., C. Fragouli, and M. B. Srivastava (1997) "Low Power Error Control for Wireless Links," in *Proceedings of MOBICOM*, pp. 139–150.

[24] Bertozzi, D., L. Benini, and G. D. Micheli. (2002) "Low Power Error Resilient Encoding for On-chip Data Buses," in *Proceedings of Design Automation and Test Conference in Europe.*

[25] Anghel, L. and M. Nicolaidis. (2000) "Cost Reduction and Evaluation of Temporary Faults Detecting Technique," in *DATE 2000*, pp. 591–598.

[26] Krstic, A., Y. M. Jiang, and K. T. Cheng. (2001) "Pattern Generation for Delay Testing and Dynamic Timing Analysis Considering Power-Supply Noise Effects," *IEEE Transactions on CAD*, **20**(3), pp. 416–425.

[27] Shepard, K. L. and V. Narayanan. (1996) "Noise in Deep Submicron Digital Design," in *IEEE/ACM ICCAD-96*, pp. 524–531.

[28] Dumitras, T., S. Kerner, and R. Marculescu (2003) "Towards On-chip Fault-Tolerant Communication," in *Proc. Asia & South Pacific Design Automation Conf.(ASP-DAC).*

[29] Marculescu, R. (2003) "Networks-On-Chip: The Quest for On-Chip Fault-Tolerant Communication," in *Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI'03).*

[30] Barford, P. and M. Crovella (1998) "Generating representative Web workloads for network and server performance evaluation," in *SIGMETRICS '98/PERFORMANCE '98: Proceedings of the 1998 ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*, ACM Press, New York, NY, USA, pp. 151–160.

[31] Varatkar., G. (2002) "Traffic Analysis for On-Chip Networks Design of Multimedia Applications," in *Design Automation Conference.*

[32] Goossens, K., J. van Meerbergen, A. Peeters, and P.Wielage. (2002) "Networks on Silicon: Combining Best-Effort and Guaranteed Services," in *Proceedings of Design Automation and Test Conference in Europe*, pp. 423–425.

[33] Smith, G. (2004) "Platform based design: does it answer the entire SoC challenge?" in *DAC '04: Proceedings of the 41st annual conference on Design automation*, pp. 407–407.

[34] SPIRIT Consortium, "http://www.spiritconsortium.org," .

[35] Sankaralingam, K., R. Nagarajan, H. Liu, C. Kim, J. Huh, D. Burger, S. W. Keckler, and C. R. Moore (2003) "Exploiting ILP, TLP, and DLP with The Polymorphous TRIPS Architecture," in *Proceedings of the 30th Annual International Symposium on Computer Architecture*, pp. 422–433.

[36] Kim, C., D. Burger, and S. W. Keckler (2002) "An adaptive, non-uniform cache structure for wire-delay dominated on-chip caches," in *ASPLOS-X: Proceedings of the 10th international conference on Architectural support for programming languages and operating systems*, pp. 211–222.

[37] Beckmann, B. M. and D. A. Wood (2004) "Managing Wire Delay in Large Chip-Multiprocessor Caches," in *MICRO 37: Proceedings of the 37th annual IEEE/ACM International Symposium on Microarchitecture*, pp. 319–330.

[38] Chishti, Z., M. D. Powell, and T. N. Vijaykumar (2003) "Distance Associativity for High-Performance Energy-Efficient Non-Uniform Cache Architectures," in *MICRO 36: Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture*, p. 55.

[39] Rickert, P. (2004), "Problems or Opportunities? Beyond the 90nm Frontier," ICCAD - Keynote Address.

[40] Kumar, R., V. Zyuban, and D. M. Tullsen (2005) "Interconnections in Multi-Core Architectures: Understanding Mechanisms, Overheads and Scaling," in *ISCA '05: Proceedings of the 32nd Annual International Symposium on Computer Architecture*, IEEE Computer Society, Washington, DC, USA, pp. 408–419.

[41] Kahle, J. A., M. N. Day, H. P. Hofstee, C. R. Johns, T. R. Maeurer, and D. Shippy (2005) "Introduction to the cell multiprocessor," *IBM J. Res. Dev.*, **49**(4/5), pp. 589–604.

[42] Sonics, Incorporated, "http://www.sonicsinc.com," .

[43] Peterson, W. (1999) "Design Philosophy of the Wishbone SoC Architecture," in *http://www.silicore.net/wishbone.htm*.

[44] Hsieh, C.-T. and M. Pedram. (2002) "Architectural Energy Optimization by Bus Splitting," *IEEE Transactions on CAD*, **21**(4), pp. 408–414.

[45] Flynn, D. (1997) "AMBA: Enabling Reusable On-Chip Design," in *IEEE Micro*, pp. 20–27.

[46] (2000) "IBM CoreConnect Information Web Site," in *http://www.chips.ibm.com/products/powerpc/cores*.

[47] Lahari, K., A. Raghunathan, and G. Laskhminarayana. (2001) "Lotterybus: A New High-performance Communication Architecture for System-on-Chip Designs," in *Design Automation Conference*.

[48] Karim, F., A. Nguyen, S. Dey, and R. Rao. (2001) "On-chip Communication Architecture for OC-768 Network Processors," in *Design Automation Conference*.

[49] Wingard, D. (2001) “MicroNetwork-Based Integration for SOCs,” in *Design Automation Conference*, pp. 673–677.

[50] van Meeuwen, T., A. Vandecappelle, A. van Zelst, F. Catthoor, and D. Verkest. (2001) “System-level Interconnect Architectures Exploration for Custum Memory Organizations,” in *International Symposium on System Synthesis*, pp. 13–18.

[51] Yoshimura, R., T. B. Keat, T. Ogawa, S. Hatanaka, and T. Matsuoka (2000) “DS-CDMA Wired Bus with Simple Interconnection Topology for Parallel Processing System LSIs,” in *IEEE ISSCC*, pp. 370–371.

[52] Richardson, T. D., C. Nicopoulos, D. Park, V. Narayanan, Y. Xie, C. Das, and V. Degalahal (2006) “A Hybrid SoC Interconnect with Dynamic TDMA-Based Transaction-Less Buses and On-Chip Networks,” in *VLSID ’06: Proceedings of the 19th International Conference on VLSI Design held j ointly with 5th International Conference on Embedded Systems Design*, pp. 657–664.

[53] Davis, J. D., C. Fu, and J. Laudon (2005) “The RASE (Rapid, Accurate Simulation Environment) for chip multiprocessors,” *SIGARCH Comput. Archit. News*, **33**(4), pp. 14–23.

[54] Barroso, L. A., K. Gharachorloo, R. McNamara, A. Nowatzyk, S. Qadeer, B. Sano, S. Smith, R. Stets, and B. Verghese (2000) “Piranha: a scalable architecture based on single-chip multiprocessing,” in *ISCA ’00: Proceedings of the 27th annual international symposium on Computer architecture*, pp. 282–293.

[55] Hammond, L., B. A. Nayfeh, and K. Olukotun (1997) “A Single-Chip Multiprocessor,” *Computer*, **30**(9), pp. 79–85.

[56] Chang, J., S. Ravi, and A. Raghunathan. (2002) “FLEXBAR: A Crossbar Switching Fabric with Improved Performance and Utilization,” in *Proceedings of IEEE CICC*, pp. 405–408.

[57] et al., H. C. (1999) *Surviving teh SoC Revoution: A Guide to Platform-Based Design*, Kluwer Academic Publishers.

[58] Ye, T. T., L. Benini, and G. D. Micheli (2002) “Analysis of Power Consumption on Switch Fabrics in Network Routers,” in *Design Automation Conference, DAC; Proceedings 2002*, pp. 524–529.

[59] Wang, H., X. Zhu, L.-S. Peh, and S. Malik (2002) “Orion: A Power-Performance Simulator for Interconnection Networks,” in *ACM/IEEE MICRO*.

[60] Ogras, U. Y. and R. Marculescu (2006) “Prediction-based flow control for network-on-chip traffic,” in *DAC ’06: Proceedings of the 43rd annual conference on Design automation*, pp. 839–844.

[61] Kim, J., D. Park, T. Theocharides, N. Vijaykrishnan, and C. R. Das (2005) “A Low Latency Router Supporting Adaptivity for On-Chip Router,” in *42nd Design Automation Conference (DAC)*.

[62] Liang, J., S. Swaminathan, and R. Tessier (2000) “aSOC: A Scalable, Single-Chip Communications Architecture,” in *the IEEE International Conference on Parallel Architectures and Compilation Techniques*, pp. 524–529.

[63] Kumar, S., A. Jantsch, J. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, and A. Hemani (2002) “A Network on Chip Architecture and Design Methodology,” in *Proc. IEEE Computer Society Annual Symposium on VLSI*, pp. 105–112.

[64] Hu, J. and R. Marculescu. (2003) "Exploiting the Routing Flexibility for Energy/Performance Aware Mapping of Regular NoC Architectures," in *Proc. Design, Automation and Test in Europe Conference*.

[65] Ye, T. T., L. Benini, and G. D. Micheli (2003) "Packetized On-Chip Interconnect Communication Analysis for MPSoC," in *Proc. Design Automation and Test in Europe*, pp. 344–349.

[66] Zhang, H., M. Wan, V. George, and J. Rabaey (1999) "Interconnect Architecture Exploration for Low-Energy Reconfigurable Single-Chip DSPs," in *Proceedings of the WVLSI*.

[67] Baker, J. M., S. B. Jr., M. Bucciero, B. Gold, and R. Mahajan (2002) "SCMP: A Single-Chip Message Passing Parallel Computer," in *Parallel and Distributed Processing Techniques and Applications (PDPTA'02)*, pp. 1485–1491.

[68] Adriahantenaina, A., H. Charlery, A. Greiner, L. Moriiez, and C. A. Zeferino. (2003) "SPIN: A Scalable, Packet Switched, On-chip Micro-network," in *Proceedings of the DATE*.

[69] Sgroi, M., M. Sheets, A. Mihal, K. Keutzer, S. Malik, J. Rabaey, and A. Sangiovanni-Vincentelli. (2001) "Addressing The System-on-a-Chip Interconnect Woes through Communication-based Design," in *Proceedings of Design Automation Conference*, pp. 667–672.

[70] Day, J. D. and H. Zimmermann (1983) "The OSI Reference Model," *Proc. of the IEEE*, **71**(3), pp. 1334–1340.

[71] Guerrier, P. and A. Grenier "A Generic Architecture for On-Chip Packet-Switched Interconnections," in *Proc. IEEE Design Automation and Test in Europe (DATE 2000)*, pp. 250–256.

[72] Marescaux, T., A. B. D. Verkest, S. Vernalde, and R. Lauwereins (2002) "Interconnection Networks Enable Fine-Grain Dynamic Multitasking on FPGAs," in *FPL*, p. LNCS 2438.

[73] Kim, J., C. Nicopoulos, D. Park, V. Narayanan, M. S. Yousif, and C. R. Das (2006) "A Gracefully Degrading and Energy-Efficient Modular Router Architecture for On-Chip Networks," in *33rd International Symposium on Computer Architecture (ISCA)*, pp. 4–15.

[74] Nicopoulos, C., D. Park, J. Kim, V. Narayanan, M. S. Yousif, and C. R. Das (2006) "ViChaR: A Dynamic Virtual Channel Regulator for Network-on-Chip Routers," in *Proc. of 39th International Symposium on Microarchitecture (MICRO) - to appear*.

[75] Jalabert, A., S. Murali, L. Benini, and G. D. Micheli (2004) "xPipes Compiler: a tool for instantiating application specific networks on chip," in *Proc. of DATE*, pp. 884–889.

[76] Rijpkema, E., K. Goossens, A. Radulescu, J. van Meerbergen, P.Wielage, and E. Waterlander. (2003) "Trade offs in the Design of a Router with Both Guaranteed and Best-effort Services for Networks on Chip," in *Design Automation and Test Conference in Europe*.

[77] Dielissen, J. and et. al. (2003) "Concepts and Implementation of the Philips Network-on-Chip," in *IP-Based SOC Design*.

[78] Mullins, R. and et. al. (2004) "Low-Latency Virtual-Channel Routers for On-Chip Networks," in *Proc. of the 31st ISCA*, IEEE Computer Society, p. 188.

[79] Kim, J., D. Park, T. Theocharides, N. Vijaykrishnan, and C. R. Das (2005) "A Low Latency Router Supporting Adaptivity for On-Chip Router," in *42nd DAC 2005*.

[80] Williams, J., N. Heintze, and B. Ackland (2002), "Communication Mechanisms for Parallel DSP Systems on Chip," .

[81] A. Jaleel, M. M. and B. Jacob (2006) "Last level cache (LLC) performance of data mining workloads on a CMP - a case study of parallel bioinformatics workloads," in *HPCA '06: Proceedings of the 12th International Symposium on High-Performance Co mputer Architecture*, pp. 88–98.

[82] Ho, W. H. and T. M. Pinkston (2003) "A Methodology for Designing Efficient On-Chip Interconnects on Well-Behaved Communication Patterns ," in *The Ninth International Symposium on High-Performance Computer Architecture (HPCA'03)*, p. 377.

[83] et al., S. K. (2002) "A Network on Chip Architecture and Design Methodology," in *Proc. Symposium on VLSI*, pp. 117–124.

[84] et al., A. H. (2000) "Network on a Chip: An Architecture for Billion Transistor Era," in *Proc. of the IEEE NorChip Conference*.

[85] Brinkmann, A., J.-C. Niemann, I. Hehemann, D. Langen, and M. P. andU. Ruckert (2002) "On-Chip Interconnects for Next Generation System-on-Chips," in *In Proc. of the 15th Annual IEEE International ASIC/SOC Conference*.

[86] Grecu, C. and M. Jones (2005) "Performance Evaluation and Design Trade-Offs for Network-on-Chip Interconnect Architectures," *IEEE Trans. Comput.*, **54**(8), pp. 1025–1040, student Member-Partha Pratim Pande and Senior Member-Andre Ivanov and Senior Member-Resve Saleh.

[87] Ivanov, A. and G. D. Micheli (2005) " Guest Editors Introduction: The Network-on-Chip Paradigm in Practice and Research," *IEEE Design and Test of Computers*, **22**(5), pp. 399–403.

[88] Wang, H., L.-S. Peh, and S. Malik (2005) "A Technology-Aware and Energy-Oriented Topology Exploration for On-Chip Networks," in *DATE '05: Proceedings of the conference on Design, Automation and Test in Europe*, IEEE Computer Society, Washington, DC, USA, pp. 1238–1243.

[89] Eisley, N. and L.-S. Peh (2004) "High-level power analysis for on-chip networks," in *CASES '04: Proceedings of the 2004 international conference on Compilers, architecture, and synthesis for embedded systems*, ACM Press, New York, NY, USA, pp. 104–115.

[90] Svensson, C. (2001) "Optimum Voltage Swing on On-Chip and Off-Chip Interconnect," *IEEE Journal of Solid-State Circuits*, **36**(7), pp. 1108–1112.

[91] (2000) "Low-Swing On-Chip Signaling Techniques: Effectiveness and Robustness," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, **8**(3), pp. 264–272.

[92] Dally, W. and J. W. Poulton (1998) *Digital Systems Engineering*, Cambridge University Press.

[93] Worm, F., P. Ienne, P. Thiran, and G. De Micheli (2002) "An Adaptive Low-Power Transmission Scheme for On-Chip Networks," in *Proceedings of The 15th International Symposium on System Synthesis*, pp. 92–100.

[94] ANSI/TIA/EIA-644-1995 Telecommunications Industry Association (1996), "Electrical Characteristics of Low Voltage Differentual Signaling (LVDS) Interface Circuits," .

[95] Sotiriadis, P. P. and A. Chandrakasan (2000) "Low Power Bus Coding Techniques Considering Inter-wire Capacitances," in *Proceedings of the IEEE Custom Integrated Circuit Conf.*

[96] Kim, J., D. Park, C. Nicopoulos, N. Vijaykrishnan, and C. R. Das (2005) "Design and analysis of an NoC architecture from performance, reliability and energy perspective," in *ANCS '05: Proceedings of the 2005 symposium on Architecture for networking and communications system s*, pp. 173–182.

[97] Kim, J. S., M. B. Taylor, J. Miller, and D. Wentzlaff (2003) "Energy characterization of a tiled architecture processor with on-chip networks," in *ISLPED '03: Proceedings of the 2003 international symposium on Low power electronics and design*, ACM Press, New York, NY, USA, pp. 424–427.

[98] Dally, W. J. (1991) "Express Cubes: Improving the Performance of k-ary n-cube Interconnection Networks," *IEEE Trans. Comput.*, **40**(9), pp. 1016–1023.

[99] Shang, L., L.-S. Peh, and N. K. Jha (2002) "Power-Efficient Interconnection Networks: Dynamic Voltage Scaling with Links," in *Computer Architecture Letters*, vol. 1.

[100] Simunic, T. and S. Boyd (2002) "Managing Power Consumption in Networks on Chip," in *Proceedings of the conference on Design, automation and test in Europe*.

[101] Kim, E. J., K. H. Yum, G. M. Link, C. R. Das, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin (2003) "Energy Optimization Techniques in Cluster Interconnects," in *International Symposium on Low Power Electronics and Design (ISLPED'03)*.

[102] Soteriou, V. and L.-S. Peh (2003) "Dynamic Power Management for Power Optimization of Interconnection Networks Using On/Off Links," in *Proceedings of the 11th Symposium on High Performance Interconnects (Hot Interconnects)*.

[103] Irani, S., S. Shukla, and R. Gupta (2003) "Algorithms for Power Savings," in *Proc. ACM SODA*, pp. 37–46.

[104] Brooks, D. and M. Martonosi (2001) "Dynamic Thermal Management for High-Performance Microprocessors," in *HPCA*, pp. 171–.
URL `citeseer.ist.psu.edu/brooks01dynamic.html`

[105] Skadron, K., M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan (2003) "Temperature-aware microarchitecture," in *ISCA '03: Proceedings of the 30th annual international symposium on Computer architecture*, ACM Press, New York, NY, USA, pp. 2–13.

[106] Donald, J. and M. Martonosi (2006) "Techniques for Multicore Thermal Management: Classification and New Exploration," in *ISCA '06: Proceedings of the 33rd annual international symposium on Computer Architecture*, IEEE Computer Society, Washington, DC, USA, pp. 78–88.

[107] Shang, L., L.-S. Peh, A. Kumar, and N. K. Jha (2004) "Thermal Modeling, Characterization and Management of On-Chip Networks," in *Proc. of the 37th MICRO*.

[108] Hung, W., C. Addo-Quaye, T. Theocharides, Y. Xie, N. Vijaykrishnan, and M. J. Irwin (2004) "Thermal-Aware IP Virtualization and Placement for Networks-on-Chip Architecture," in *ICCD '04: Proceedings of the IEEE International Conference on Computer Design (ICCD'04)*, IEEE Computer Society, Washington, DC, USA, pp. 430–437.

[109] Shivakumar, P., M. Kistler, S. W. Keckler, D. Burger, and L. Alvisi (2002) "Modeling the Effect of Technology Trends on the Soft Error Rate of Combinational Logic," in *DSN '02: Proceedings of the 2002 International Conference on Dependable Systems and Networks*, IEEE Computer Society, Washington, DC, USA, pp. 389–398.

[110] Murali, S., T. Theocharides, N. Vijaykrishnan, M. J. Irwin, L. Benini, and G. D. Micheli (2005) "Analysis of Error Recovery Schemes for Networks on Chips," *IEEE Des. Test*, **22**(5), pp. 434–442.

[111] Anghel, L. and M. Nicolaidis (2000) "Cost reduction and evaluation of temporary faults detecting technique," in *DATE '00: Proceedings of the conference on Design, automation and test in Europe*, pp. 591–598.

[112] Shepard, K. L. and V. Narayanan (1996) "Noise in deep submicron digital design," in *ICCAD '96: Proceedings of the 1996 IEEE/ACM international conference on Computer-aided design*, pp. 524–531.

[113] Liou, J. J., A. Krsti&#263;, Y. M. Jiang, and K. T. Cheng (2000) "Path selection and pattern generation for dynamic timing analysis considering power supply noise effects," in *ICCAD '00: Proceedings of the 2000 IEEE/ACM international conference on Computer-aided design*, pp. 493–497.

[114] Zimmer, H. and A. Jantsch (2003) "A Fault Model Notation and Error-Control Scheme for Switch-to-Switch Buses in a Network-on-Chip," in *Proceedings of the 1st IEEE/ACM/IFIP international conference on Hardware/software codesign & system synthesis*, pp. 188–193.

[115] Constantinides, K., S. Plaza, J. Blome, B. Zhang, V. Bertacco, S. Mahlke, T. Austin, and M. Orshansky (2006) "BulletProof: A Defect-Tolerant CMP Switch Architecture," in *In proceedings of the 12th International Symposium on High-Performance Computer Architecture (HPCA-12)*.

[116] Lin, S., D. J. C. Jr., and M. J. Miller (1984) "Automatic-Repeat-Request Error-Control Schemes," *IEEE Communications Magazine*.

[117] Kermode, R. G. (1998) "Scoped Hybrid Automatic Repeat reQuest with Forward Error Correction (SHARQFEC)," in *Proceedings of the ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication*, pp. 278–289.

[118] Park, D., C. Nicopoulos, J. Kim, N. Vijaykrishnan, and C. R. Das (2006) "Exploring Fault-Tolerant Network-on-Chip Architectures," in *DSN '06: Proceedings of the International Conference on Dependable Systems and Networks (DSN'06)*, pp. 93–104.

[119] Nilsson, E., M. Millberg, J. Oberg, and A. Jantsch. (2003) "Load Distribution with the Proximity Congestion Awareness in a Network on Chip," in *Proc. Design Automation and Test in Europe*.

[120] "Nostrum Chip in NOCARC Project," in *http://www.imit.kth.se/info/FOFU/NOC/*.

[121] Nabaa, G., N. Azizi, and F. N. Najm (2006) "An adaptive FPGA architecture with process variation compensation and reduced leakage," in *DAC '06: Proceedings of the 43rd annual conference on Design automation*, pp. 624–629.

[122] Das, C. R. and J. Kim (1992) "A Unified Task-Based Dependability Model for Hypercube Computers," *IEEE Trans. on Parallel and Distributed Systems*, **3**(5), pp. 312–324.

[123] Goseva-Popstojanova, K. and K. S. Trivedi (2000) " Stochastic Modeling Formalisms for Dependability, Performance and Performability," *Performance Evaluation*.

[124] Das, C. R., P. Mohapatra, L. Tien, and L. N. Bhuyan (1993) "An Availability Model for MIN-Based Multiprocessors," *IEEE Trans. on Parallel and Distributed Systems*, **4**(10), pp. 1118–1129.

[125] Mohapatra, P. and C. R. Das (1995) "On Dependability Evaluation of Mesh Connected Systems," *IEEE Trans. on Computers*, **44**(9), pp. 1073–1084.

[126] Kim, J. and C. R. Das (1991) "On Subcube Dependability in a Hypercube," in *Proc. ACM SIGMETRICS Conference on Measurements and Modeling of Computer Systems*, pp. 111–119.

[127] Hunter, S. W., T. Philip, and K. S. Trivedi (1997) "Combined Performance and Availability Analysis of a Switched Network Applications," in *ICC*, pp. 241–245.

[128] Blake, J. T. and K. S. Trivedi (1989) "Reliability Analysis of Interconnection Networks Using Hierarchical Decomposition," *IEEE Trans. on Reliability*, pp. 111–120.

[129] Peh, L.-S. and W. J. Dally (2001) "A Delay Model and Speculative Architecture for Pipelined Routers," in *Proceedings of the 7th International Symposium on High-Performance Computer Architecture*.

[130] Hu, J. and R. Marculescu (2003) "Energy-Aware Mapping for Tile-based NoC Architectures Under Performance Constraints," in *Proc. of ASPDAC*.

[131] Sarbazi-Azad, H., M. Ould-Khaoua, and L. M. Mackenzie (2002) "A Performance Model of Adaptive Wormhole Routing in k-Ary n-Cubes in the Presence of Digit-Reversal Traffic," *J. Supercomput.*, **22**(2), pp. 139–159.

[132] Jung, E., K. K. H. Yum, and C. R. Das (2001) "Calculation of Deadline Missing Probability in a QoS Capable Cluster Interconnect," in *NCA '01: Proceedings of the IEEE International Symposium on Network Computing and Applications (NCA'01)*, IEEE Computer Society, Washington, DC, USA, p. 36.

[133] Agarwal, A. (1991) "Limits on Interconnection Network," *IEEE Transaction on Parallel and Distibuted Systems*, **2**(4).

[134] Sridhara, S. R. and N. R. Shanbhag (2004) "Coding for system-on-chip networks: a unified framework," in *Design Automation Conference*, pp. 103–106.

[135] Vellanki, P., N. Banerjee, and K. Chatha (2004) "Quality-of-Service and Error Control Techniques for Network-on-Chip Architectures," in *Proceedings of the Great Lakes Symposium on VLSI*.

[136] Bertozzi, D., L. Benini, and G. D. Micheli (2002) "Low power error resilient encoding for on-chip data buses," in *Proceedings of 2002 DATE*.

[137] Shivakumar, P., M. Kistler, S. Keckler, D. Burger, and L. Alvisi (2002) "Modeling the Effect of Technology Trends on the Soft Error Rate of Combinational Logic," in *Proceedings of the International Conference on Dependable Systems and Networks*.

[138] Vaidya, A. S., A. Sivasubramaniam, and C. R. Das (1997) "Performance benefits of virtual channels and adaptive routing: an application-driven study," in *ICS '97: Proceedings of the 11th international conference on Supercomputing*, ACM Press, New York, NY, USA, pp. 140–147.

[139] Caminero, B., F. J. Quiles, J. Duato, D. S. Love, and S. Yalamanchili (1999) "Performance Evaluation of the Multimedia Router with MPEG-2 Video Traffic," in *CANPC '99: Proceedings of the Third International Workshop on Network-Based Parallel Computing*, Springer-Verlag, London, UK, pp. 62–76.

[140] Zeng, A. Y., J. J. Lü, K. Rose, and R. J. Gutmann (2005) "First-Order Performance Prediction of Cache Memory with Wafer-Level 3D Integration." *IEEE Design & Test of Computers*, **22**(6), pp. 548–555.

[141] Kgil, T., S. D'Souza, A. Saidi, N. Binkert, R. Dreslinski, S. Reinhardt, K. Flautner, and T. Mudge (2006) "PICOSERVER: Using 3D Stacking Technology to Enable a Compact Energy Efficient Chip Multiprocessor," in *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-XII)*.

[142] Li, F., C. Nicopoulos, T. Richardson, Y. Xie, V. Narayanan, and M. Kandemir (2006) "Design and Management of 3D Chip Multiprocessors Using Network-in-Memory," in *33rd International Symposium on Computer Architecture (ISCA)*, pp. 130–141.

[143] Davis, W. R., J. Wilson, S. Mick, J. Xu, H. Hua, C. Mineo, A. M. Sule, M. Steer, and P. D. Franzon (2005) "Demystifying 3D ICs: The Pros and Cons of Going Vertical," *IEEE Design & Test of Computers*, **22**(6), pp. 498–510.

[144] Cong, J. and Y. Zhang (2005) "Thermal via planning for 3-D ICs," in *ICCAD '05: Proceedings of the 2005 IEEE/ACM International conference on Computer-aided design*, IEEE Computer Society, Washington, DC, USA, pp. 745–752.

[145] Goplen, B. and S. Sapatnekar (2003) "Efficient thermal placement of standard cells in 3D ICs using a force directed approach," in *International Conference on Computer Aided Design (ICCAD)*, pp. 86–89.

[146] Dang, B., P. Joseph, M. Bakir, T. Spencer, P. Kohl, and J. Meindl (2005) "Wafer-level microfluidic cooling interconnects for GSI," in *Proceedings of the IEEE 2005 International Interconnect Technology Conference*, pp. 180–182.

[147] Hung, W.-L., G. Link, Y. Xie, N. Vijaykrishnan, and M. J. Irwin (2006) "Interconnect and Thermal-aware Floorplanning for 3D Microprocessors," in *7th International Symposium on Quality Electronic Design (ISQED)*, pp. 98–104.

[148] Black, B., M. M. Annavaram, E. Brekelbaum, J. DeVale, L. Jiang, G. H. Loh, D. McCauley, P. Morrow, D. W. Nelson, D. Pantuso, P. Reed, J. Rupley, S. Shankar, J. P. Shen, and C. Webb "Die Stacking (3D) Microarchitecture," *To appear in the proceedings of the 39th annual IEEE/ACM International Symposium on Microarchitecture*, 2006.

[149] Liu, C. C., I. Ganusov, M. Burtscher, and S. Tiwari (2005) "Bridging the Processor-Memory Performance Gapwith 3D IC Technology," *IEEE Design & Test of Computers*, **22**(6), pp. 556–564.

[150] Puttaswamy, K. and G. H. Loh (2005) "Implementing Caches in a 3D Technology for High Performance Processors," in *ICCD '05: Proceedings of the 2005 International Conference on Computer Design*, IEEE Computer Society, Washington, DC, USA, pp. 525–532.

[151] Mysore, S., B. Agrawal, N. Srivastava, S.-C. Lin, K. Banerjee, and T. Sherwood (2006) "Introspective 3D chips," in *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*.

[152] Duato, J. (1993) "A New Theory of Deadlock-Free Adaptive Routing in Wormhole Networks," *IEEE Trans. on Parallel and Distributed Systems*, **4**(12), pp. 1320–1331.

[153] Chandrakasan, A., W. J. Bowhill, and F. Fox (2001) *Design of High-Performance Microprocessor Circuits*, IEEE Press.

[154] "TPC-C Design Document," Http://www.tpc.org/tpcc/.

[155] "SAP Sales and Distribution Benchmark," Http://www.sap.com/solutions/benchmark/index.epx.

[156] "SPECjbb2005 Java Business Benchmark," Http://www.spec.org/jbb2005.

[157] "SPECjAppServer Java Application Server Benchmark," Http://www.spec.org/jAppServer.

[158] Woo, S. C., M. Ohara, E. Torrie, J. P. Singh, and A. Gupta (1995) "The SPLASH-2 programs: characterization and methodological considerations," in *ISCA '95: Proceedings of the 22nd annual international symposium on Computer architecture*, pp. 24–36.

[159] Magnusson, P. S., M. Christensson, J. Eskilson, D. Forsgren, G. Hallberg, J. Hogberg, F. Larsson, A. Moestedt, and B. Werner (2002) "Simics: A Full System Simulation Platform," *Computer*, **35**(2), pp. 50–58.

[160] Sarbazi-Azad, H., M. Ould-Khaoua, and L. M. Mackenzie (2002) "Analytical modelling of wormhole-routed k-ary n-cubes in the presence of matrix-transpose traffic," *J. Parallel Distrib. Comput*, **62**(4), pp. 605–621.

Vita

Jongman Kim

Jongman Kim is a Ph.D. candidate in the Department of Computer Science and Engineering, the Pennsylvania State University, University Park. He received his B.S. degree from the Department of Electrical Engineering, Seoul National University, Seoul, Korea, in 1990. Since 1993, after discharging military service, he had been employed in LG Electronics and involved in other professional work. He received M.S. degree from the Department of Electrical Engineering, the Pennsylvania State University, University Park, in 2001. In 2002, he enrolled in the Ph. D. program in the Department of Computer Science and Engineering at the Pennsylvania State University. He is a member of IEEE. He served as a technical referee for numerous journals and conferences including IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Computer, Int'l Symposium on High-Performance Computer Architecture, Int'l Conference on Parallel and Distributed Processing Symposium, Int'l Conference on Parallel Processing, and European Conference on Parallel Computing.