DISSERTATION

RELIABILITY-AWARE AND ENERGY-EFFICIENT SYSTEM LEVEL DESIGN FOR NETWORKS-ON-CHIP

Submitted by

Yong Zou

Department of Electrical and Computer Engineering

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Spring 2015

ABSTRACT

RELIABILITY-AWARE AND ENERGY-EFFICIENT SYSTEM LEVEL DESIGN FOR NETWORKS-ON-CHIP

With CMOS technology aggressively scaling into the ultra-deep sub-micron (UDSM) regime and application complexity growing rapidly in recent years, processors today are being driven to integrate multiple cores on a chip. Such chip multiprocessor (CMP) architectures offer unprecedented levels of computing performance for highly parallel emerging applications in the era of digital convergence. However, a major challenge facing the designers of these emerging multicore architectures is the increased likelihood of failure due to the rise in transient, permanent, and intermittent faults caused by a variety of factors that are becoming more and more prevalent with technology scaling. On-chip interconnect architectures are particularly susceptible to faults that can corrupt transmitted data or prevent it from reaching its destination. Reliability concerns in UDSM nodes have in part contributed to the shift from traditional bus-based communication fabrics to network-on-chip (NoC) architectures that provide better scalability, performance, and utilization than buses.

In this thesis, to overcome potential faults in NoCs, my research began by exploring fault-tolerant routing algorithms. Under the constraint of deadlock freedom, we make use of the inherent redundancy in NoCs due to multiple paths between packet sources and sinks and propose different fault-tolerant routing schemes to achieve much better fault tolerance capabilities than possible with traditional routing schemes. The proposed schemes also use replication opportunistically to optimize the balance between energy overhead and arrival rate.

As 3D integrated circuit (3D-IC) technology with wafer-to-wafer bonding has been recently proposed as a promising candidate for future CMPs, we also propose a fault-tolerant routing scheme for 3D NoCs which outperforms the existing popular routing schemes in terms of energy consumption, performance and reliability. To quantify reliability and provide different levels of intelligent protection, for the first time, we propose the network vulnerability factor (NVF) metric to characterize the vulnerability of NoC components to faults. NVF determines the probabilities that faults in NoC components manifest as errors in the final program output of the CMP system. With NVF aware partial protection for NoC components, almost 50% energy cost can be saved compared to the traditional approach of comprehensively protecting all NoC components. Lastly, we focus on the problem of fault-tolerant NoC design, that involves many NP-hard sub-problems such as core mapping, fault-tolerant routing, and fault-tolerant router configuration. We propose a novel design-time (RESYN) and a hybrid design and runtime (HEFT) synthesis framework to trade-off energy consumption and reliability in the NoC fabric at the system level for CMPs. Together, our research in fault-tolerant NoC routing, reliability modeling, and reliability aware NoC synthesis substantially enhances NoC reliability and energy-efficiency beyond what is possible with traditional approaches and state-of-the-art strategies from prior work.

ACKNOWLEDGEMENT

First and foremost, I would like to extend my sincere gratitude to my advisor Dr. Sudeep Pasricha who continuously inspired me and provided vast guidance towards the completion of this research. The course and research work I did under the guidance of Dr. Sudeep Pasricha truly helped me to develop expertise from silicon-to-system. His words of wisdom enabled me to learn cutting edge technologies by understanding and addressing many of the challenges faced by industry as well as the research community. By providing focus on critical and valuable research problems and practical solutions, he helped me to get visibility for my research within well-known conferences, journals and institutions. Dr. Sudeep Pasricha is dynamic, enthusiastic, energetic and full of many ideas, which truly makes him one of the greatest advisers and Professors I have known and worked with. The regular brainstorming sessions with Dr. Sudeep Pasricha provided me great insight on many technological areas. I am truly grateful for his excellent guidance and support over the past six years. Without his guidance and help, this research work was definitely not possible. I recall our brainstorming sessions for every research paper or publication that I worked on, and always felt that I was full of ideas to try solving the next set of challenges every time walking out of his office. I performed many “what-if' analysis cases” to obtain the best possible solutions. Dr. Sudeep Pasricha always kept raising the bar and always pushed me to do better in every aspect of research. I was also fortunate to attend a number of conferences and present my work in front many researchers and the engineering community. The exciting feedback and awards received during these conferences speaks for itself about the high standard that Dr. Sudeep Pasricha adheres to in all aspects of research that includes every major and minor aspect of research. If I look back, not only have I gained technical knowledge but his guidance has also helped to improve my presentation, communication, and writing skills.

I would like to take this opportunity to thank the respected members of my PhD committee, Dr. Sourajeet Roy, Dr. T. W. Chen and Dr. Wim Bohm for their insightful feedback and encouragement for my research. The feedback they provided me helped to enhance and broaden my research. I am also thankful to the many students in Dr. Pasricha's MECS lab for their support and help during my PhD: Ishan Thakkar, Yi Xiang, C Sai Vineel Reddy, Tejasi Pimpalkhute, Nishit Kapadia, Mark Oxley, Dalton Young and Srinivas Desai. Weekly team meetings managed by Dr. Sudeep Pasricha helped to institutionalize the knowledge through the presentations and discussions with all team members.

Last but not least, I would like to thank my family for their support to continue my research. My wife Eva encouraged and supported me to continue my studies and inspired me to keep going while managing multiple priorities with my family, studies, and job. This achievement was not possible without their immense support.

DEDICATION

To my parents, beautiful wife Eva, all of my friends and family

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CURRICULUM VITAE

EDUCATION

PhD. Candidate, 2014, Colorado State University, Fort Collins

Master, 2009, University of science and technology of China, China

Bachelor, 2005, Jilin University, China

INTERNSHIP

2014.6-2014.8	Synopsys, Mountain view, California
2011.5-2013.8	Broadcom corporation, Longmont, Colorado
2007.5-2009.3	Spreadtrum communication, Shanghai, China

PUBLICATIONS

BOOK CHAPTERS:

S. Pasricha, Y. Zou, "Hybrid Partially Adaptive Fault Tolerant Routing for 3D Networks-on-Chip", Embedded Systems: Hardware, Design, and Implementation, John Wiley & Sons, Inc., Hoboken, NJ, USA. doi: 10.1002/9781118468654.ch10, Nov 2012.

JOURNAL PAPERS:

J1: Y. Zou, Y. Xiang, S. Pasricha, "Characterizing Vulnerability of Network Interfaces in Embedded Chip Multiprocessors", IEEE Embedded System Letters, 4(2), Jun 2012.

J2: Y. Zou, S. Pasricha, "NARCO: Neighbor Aware Turn Model Based Fault Tolerant Routing for NoCs", IEEE Embedded System Letters, Vol. 2, No. 3, Sep 2010.

J3: M. Oxley, S. Pasricha, A. A. Maciejewski, H. J. Siegel, J. Apodaca, D. Young, L. Briceno, J. Smith, S. Bahirat, B. Khemka, A. Ramirez and Y. Zou, "Makespan and Energy Robust Stochastic

Static Resource Allocation of Bags-of-Tasks to a Heterogeneous Computing System", accepted for publication, IEEE Transactions on Parallel and Distributed Systems, 2014.

J4: D. Young, J. Apodaca, L. Briceno, J. Smith, S. Pasricha, A. Maciejewski, H. Siegel, S. Bahirat, B. Khemka, A. Ramirez and Y. Zou, "Deadline and Energy Constrained Dynamic Resource Allocation in a Heterogeneous Computing Environment", Journal of Supercomputing, 2012.

CONFERENCE PAPERS

C1: Y. Zou, S. Pasricha, "HEFT: A Hybrid System-Level Framework for Enabling Energy-Efficient Fault-Tolerance in NoC based MPSoCs", IEEE/ACM International Conference on Hardware/Software Codesign and System Synthesis (CODES-ISSS), Oct 2014.

C2: Y. Zou, S. Pasricha, "Reliability-Aware and Energy-Efficient Synthesis of NoC based MPSoCs", IEEE International Symposium on Quality Electronic Design (ISQED), Mar. 2013.

C3: Y. Zou, Y. Xiang, S. Pasricha, "Analysis of On-chip Interconnection Network Interface Reliability in Multicore Systems", IEEE International Conference on Computer Design (ICCD), Oct. 2011.

C4: S. Pasricha, Y. Zou, "A Low Overhead Fault Tolerant Routing Scheme for 3D Networks-on-Chip", IEEE International Symposium on Quality Electronic Design (ISQED 2011) , Santa Clara, CA, Mar 2011.

C5: S. Pasricha, Y. Zou, "NS-FTR: A Fault Tolerant Routing Scheme for Networks on Chip with Permanent and Runtime Intermittent Faults", IEEE/ACM Asia & South Pacific Design Automation Conference (ASPDAC), Yokohama, Japan, Jan 2011.

C6: S. Pasricha, Y. Zou, D. Connors, H. J. Siegel, "OE+IOE: A Novel Turn Model Based Fault Tolerant Routing Scheme for Networks-on-Chip", Proc. IEEE/ACM/IFIP International

Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS), pp. 85-93, 2010.

C7: D. Young, J. Apodaca, L. Briceno, J. Smith, S. Pasricha, A. Maciejewski, H. Siegel, S. Bahirat, B. Khemka, A. Ramirez and Y. Zou, "Energy-Constrained Dynamic Resource Allocation in a Heterogeneous Computing Environment", Fourth International Workshop on Parallel Programming Models and Systems Software for High-End Computing (P2S2 2011), Sep 2011.

C8: J. Apodaca, D. Young, L. Briceno, J. Smith, S. Pasricha, A. Maciejewski, H. Siegel, S. Bahirat, B. Khemka, A. Ramirez and Y. Zou, "Stochastically Robust Static Resource Allocation for Energy Minimization with a Makespan Constraint in a Heterogeneous Computing Environment", ACS/IEEE International Conference on Computer Systems and Applications (AICCSA 2011), Dec 2011.

CHAPTER 1
INTRODUCTION

This chapter will first describe the motivation of our research: why we chose NoC as the on-chip communication fabric and the need for fault-tolerant design for NoCs. Following the research motivation, a brief introduction will be given for the main research themes that we have pursued: fault-tolerant NoC routing algorithm design, 3D NoC interconnection architectures, NoC reliability modeling, and NoC system level reliability aware synthesis. Lastly, we summarize our research contributions in this thesis.

1.1 Motivation

1.1.1 Network-on-chip (NoC)

With CMOS technology aggressively scaling into the ultra-deep sub-micron (UDSM) regime and application complexity growing rapidly in recent years, processors today are being driven to integrate multiple cores on a chip. Such chip multiprocessor (CMP) architectures offer unprecedented levels of computing performance for highly parallel emerging applications in the era of digital convergence. Traditionally bus-based architectures have been used for CMP on-chip communication. Figure 1 shows a 22nm 15-core enterprise Xeon processor block diagram with a recent Intel Ivytown microarchitecture [7]. A ring bus is used for on-chip communication. Figure 2 shows the "AMBA" bus architecture which is widely used in mobile processors: A master starts the transaction by requesting access to central arbiter. An arbiter decides the priorities in case of access conflicts and allows the master to access the bus when the bus is ready. However, these bus-based communication architectures do not scale well with the increasing number of on-chip cores in terms of bandwidth, clocking frequency and power because of the contention in the single bus resource [2].

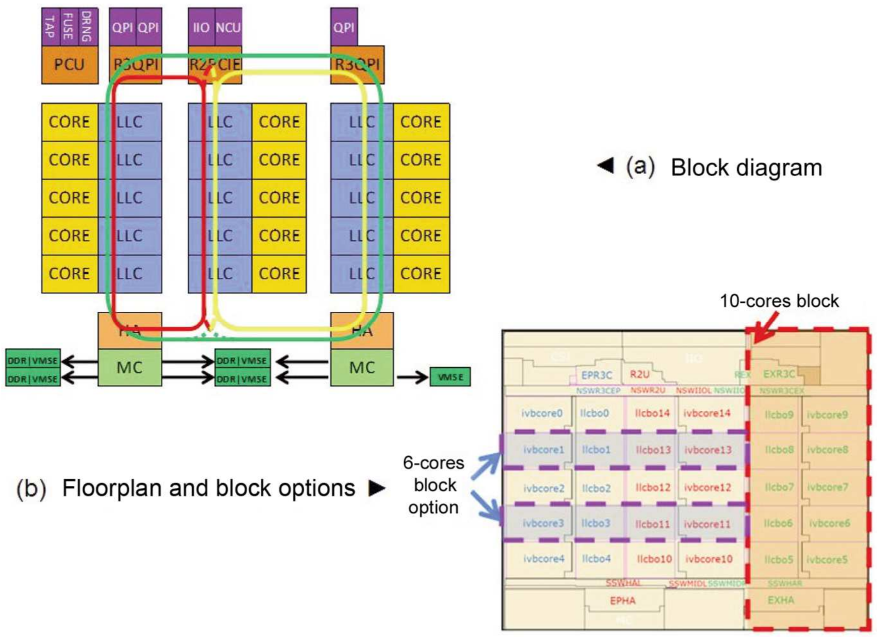


Figure 1 Block diagram and die diagram of Ivytown

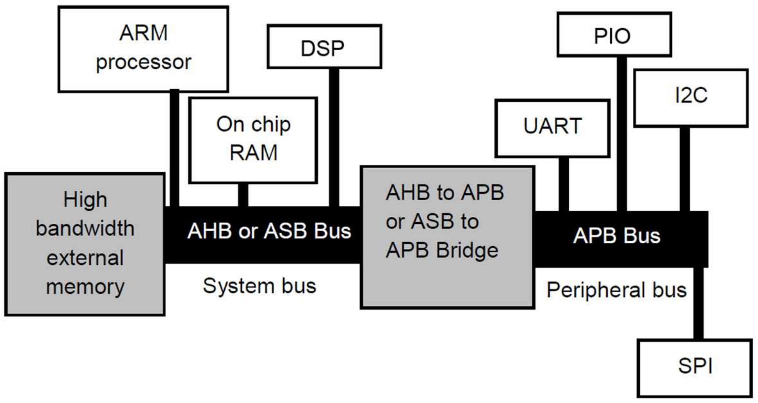
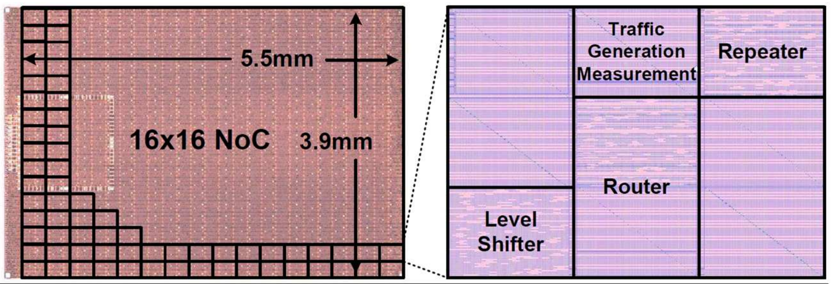


Figure 2 AMBA bus-based system architecture

Network-on-Chip (NoC) fabrics have emerged as promising on-chip communication architecture candidates for chip multiprocessors (CMPs). NoCs have been shown to provide better scalability, predictability, and performance than bus-based communication architectures [1]. Figure 3 shows an Intel 256-core processor that has a 16×16 mesh NoC, with each of the 256 nodes having independent voltages and clocks, and a transfer throughput of 20.2 terabits per second between the nodes [8]. Routers in NoC fabrics play a key role in ensuring successful delivery of packets from the source to the destination node. Figure 4 shows a block diagram of a NoC router. When a packet enters a router, it is first stored in the input buffers. Subsequently based on the destination information the route allocator (RA) will decide which output port the data will be sent to, virtual channel allocator (VA) will decide its virtual channel buffer, and switch allocator (SA) will reserve the necessary resources for the packet to be sent to the appropriately allocated output port buffer.



Process	22nm Tri-gate CMOS	Nominal Operation	0.9V, 25°C
Topology	16x16 Mesh	Throughput	20.2Tb/s
V_{DD}/Clock Domains	256	Energy Efficiency	7.0Tb/s/W
Data Bus Width	112b	Bisection Bandwidth	2.8Tb/s
Packet Size	11b Control + 32b Data	Circuit-switched Latency/Hop	407ps
Circuit Size	3b Control + 80b Data	Near-threshold Operation	430mV, 25°C
Link Length	855µm	Throughput	3.4Tb/s
Die Area	23mm^2	Peak Energy Efficiency	18.3Tb/s/W
Equivalent NoC Area	167mm^2	Ultra-low-voltage Operation	340mV, 25°C
Router Area	138µm x 109µm	Throughput	946Gb/s
Transistor Count	150M	Router Power	363µW

Figure 3 Die micrograph and measurement summary of Intel 256-node efficient network-on-chip

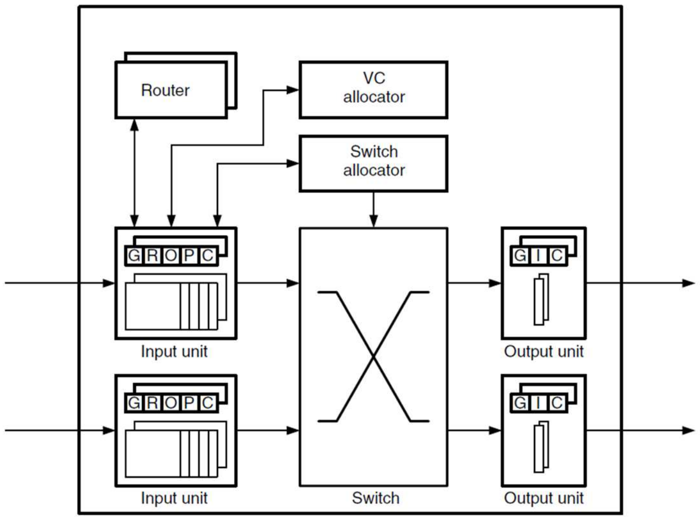


Figure 4 Virtual-channel NoC router block diagram [1]

In practice, NoC communication fabrics today still need to overcome two major challenges that are extremely relevant for emerging CMP applications. Firstly, packets switched NoCs must support low latency transfers between cores (especially for real-time applications) but are often unable to meet QoS (Quality of Service) guarantees. Using virtual circuit switching instead of packet switching allows for better QoS management, but unpredictable circuit setup delays still exist. Secondly, NoCs must enable low power data transfers. However, the large number of network interfaces, routers, links, and buffers that are part of the NoC fabric lead to a communication infrastructure that consumes a significant portion of the power from the overall CMP power budget [3][4][5]. In our research when we propose different fault-tolerant routing algorithms and reliability-aware system level synthesis solutions, minimizing communication energy and increasing NoC reliability are the two most important goals. At the same time, under the constraints of meeting the power budget and reliability goals, we provide solutions to improve the performance in terms of data delivery latency and throughput.

1.1.2 Fault tolerance in NoC

In the near future, a major challenge facing the designers of emerging multicore architectures is the increased likelihood of failure due to the rise in transient, permanent, and intermittent faults caused by a variety of factors that are becoming more and more prevalent with technology scaling. Transient faults (also called soft errors) occur when an event such as high-energy cosmic neutron particle strike, alpha particle strike due to trace uranium/thorium impurities in packages, capacitive and inductive crosstalk, or electromagnetic noise manifests itself. Such events cause the deposit or removal of enough charge to invert the state of a transistor, wire, or storage cell [9][10]. The inverted value may propagate to cause an error in program execution. These errors occur for a very short duration of time and can be hard to predict. Permanent faults occur due to manufacturing defects, or after irreversible wearout damage due to electromigration in conductors, negative bias temperature instability (NBTI), dielectric breakdown, etc. [11][12]. A third class of faults, called intermittent faults, occur frequently and irregularly for several cycles, and then disappear for a period of time [13]. These faults commonly arise due to process variations combined with variation in the operating conditions (such as voltage and temperature fluctuations).

On-chip communication architectures are particularly susceptible to faults that can corrupt transmitted data or prevent it from reaching its destination [14]. Reliability concerns in UDSM nodes have in part contributed to the shift from traditional bus-based communication fabrics to network-on-chip (NoC) architectures that provide better scalability, performance, and utilization than buses [15][16]. The inherent redundancy in NoCs due to multiple paths between packet sources and sinks can greatly improve communication fault resiliency. Several multicore chip designs are emerging that make use of NoCs as interconnection fabrics [17]-[20]. To ensure reliable data transfers in these communication fabrics, utilizing fault-tolerant design strategies is essential. Traditionally, error detection coding and retransmission has been a popular means of achieving resiliency towards transient and intermittent faults [21][22]. Alternatively, forward error correction (FEC) schemes can provide better resiliency against these faults, but usually at a higher performance and energy overhead [23]. Circuit and layout optimizations such as shield insertion and wire sizing to reduce crosstalk induced transient faults have also been proposed [24][25]. To overcome permanent faults in NoCs, fault-tolerant routing schemes are also a critical requirement and the focus of several research efforts over the last few years [27]-[45]. In the presence of intermittent or permanent faults on NoC links and routers, routing schemes can ensure error free data flit delivery by using an alternate route that is free of faults.

1.2 Fault-tolerant NoC routing

In our research, we only consider NoC with mesh topology, but all of our proposed research ideas can be applied to other NoC topologies as well. The inherent communication redundancy in NoCs due to multiple paths between packets sources and sinks can greatly improve communication fault resiliency. Figure 5 shows a 5×5 mesh NoC with each circle representing a router connected to a processing element. Each router is identified by a router id from 0 to 24. As can be seen, there are multiple paths from router 0 to router 12, so during the situation when the link between router 5 and 10 is broken, data can still be delivered from router 0 to router 12 via other routing paths.

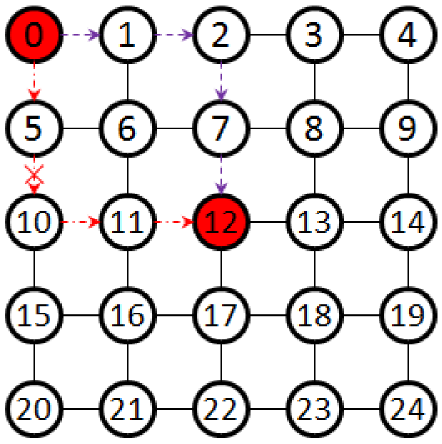


Figure 5 Routing path diversity

Ideally packets should follow the minimal path from the source to destination node, which is why the simple XY routing scheme that routes packets first in the X direction and then in the Y direction is widely used in mesh-based NoCs. While the XY routing is popular because of its simplicity and communication efficiency, it is static in nature and does not possess any adaptivity or fault resiliency to cope with situations where there are faults inside the network. Different adaptive routing algorithms have been proposed in literature to overcome this drawback. But there can be many potential problems due to the introduction of adaptive routing.

Firstly, traffic unbalancing due to adaptive routing can lead to congestion and communication performance degradation. In our routing algorithm research, when adaptively choosing a routing path to avoid errors in a link or router, we also try to balance the traffic to minimize traffic congestion. Secondly, an inevitable issue from adaptive routing is deadlock or livelock [1]. In our research we build on the turn model routing scheme because of its deadlock and livelock free properties and extend it to different situations to cope with potential permanent/intermittent/transient faults in the on-chip network. Last but not least, complex routing algorithms can introduce excessive area and energy overhead. In all of our proposed routing schemes, we set our design goal as minimizing energy, area, and power overhead while maximizing performance in terms of latency and reliability in term of successful packet delivery rate.

1.3 3D NoC interconnection

Advances in CMOS fabrication technology in recent years have led to the emergence of chip multiprocessors (CMPs) as compact and powerful computing paradigms. However, to keep up with rising application complexity, it is widely acknowledged that solutions beyond aggressive CMOS scaling into deeper nanometer technologies will be required in the near future, to improve CMP computational capabilities [6]. 3D integrated circuit (3D-IC) technology with wafer-to-wafer bonding [51][52] has been recently proposed as a promising candidate for future CMPs. In contrast to traditional 2D-IC based chips that have a single layer of active devices (processors, memories), wafer-to-wafer bonded 3D-ICs consist of multiple stacked active layers and vertical Through Silicon Vias (TSVs) to connect the devices across the layers. Multiple active layers in 3D-ICs can enable increased integration of cores within the same area footprint as traditional 2D-ICs. In addition, long global interconnects between cores can be replaced by shorter inter-layer

TSVs, improving performance and reducing on-chip power dissipation. Recent 3D-IC test chips from IBM [51], Tezzaron [52], and Intel [3] have confirmed the benefits of 3D-IC technology.

A major challenge facing the design of such highly integrated 3D-ICs in deep submicron technologies is the increased likelihood of failure due to permanent and intermittent faults caused by a variety of factors (e.g., misalignment of 3D dies, inter-layer crosstalk) that are becoming more and more prevalent. On-chip communication architectures are particularly susceptible to faults that can corrupt transmitted data or altogether prevent it from reaching its destination. Reliability concerns in sub-65nm nodes have in part contributed to the shift from traditional bus-based communication fabrics to network-on-chip (NoC) architectures that provide better scalability, predictability, and performance than buses [16][53]. The inherent redundancy in NoCs due to multiple paths between packet sources and sinks can greatly improve communication error resiliency. It is expected that 3D-ICs will employ 3D NoCs, much like todays 2D-IC based CMPs are employing 2D NoCs [3][19][54]. To ensure reliable data transfers in these communication fabrics, utilizing fault-tolerant design strategies is essential. In particular, fault-tolerant routing schemes are critical to overcome permanent and intermittent faults at design time as well as runtime. In the presence of intermittent or permanent faults on NoC links and routers, routing schemes can ensure error free data flit delivery by using an alternate route that is free of faults. As a result, fault-tolerant routing schemes for 2D NoCs have been the focus of several research efforts over the last few years [13]-[25]. However, to the best of our knowledge, no prior work has yet explored fault-tolerant routing schemes for 3D NoCs. In our research, for the first time, we explore and proposel effective techniques for fault-resilient 3D NoC routing.

1.4 NoC reliability modeling

Chip designers typically set a failure rate target for each design and strive to maximize performance subject to this constraint. To validate that a design meets the failure rate target, designers perform extensive pre- and post-silicon analysis. This analysis measures the rate at which faults occur in a system as well as whether a given fault will cause a system failure. Designers then make use of one of several well-known techniques to improve reliability in the presence of soft errors, such as triple modular redundancy (TMR) and error correction/detection coding schemes. However, these techniques come with extra overhead in terms of area, power, and even performance, and therefore must be used judiciously.

Fortunately, not all faults eventually affect the final program outcome. For example, a bit flip in an empty translation lookaside buffer (TLB) entry will not cause any effect in the program execution. Based on this observation, Mukherjee et al. [55] defined a structure‘s Architectural Vulnerability Factor (AVF) as the probability that a transient fault in the structure finally produces a visible error in the output of a program. At any point of time, a structure‘s AVF can be derived by counting all the important bits that are required for Architecturally Correct Execution (ACE) in the structure, and dividing them by the total number of bits of the structure. Such ACE analysis has been used to derive an upper bound on AVF using performance simulation for intra-processor buffers [55]-[57] and cache structures [58] in recent years.

In our research, we use the idea of AVF and extend AVF from the microprocessor domain and propose a network vulnerability factor (NVF) metric to characterize the susceptibility of NoC components. Specially, we analyze the vulnerability factor of network interface and the router in NoC so that we can quantify the reliability of these components and make protection decisions. Our studies reveal that different network interface (NI) and router buffers behave quite differently on transient faults and each buffer can have different levels of inherent fault-tolerant capabilities.

1.5 NoC system level reliability aware synthesis

As power and energy costs have become essential constraints during chip design, a systematic design methodology is required to balance reliability measures and their associated power/energy costs. This is a synthesis problem that impacts many phases of a CMP design flow, such as core-to-die mapping, NoC router configuration, NoC routing algorithm selection, etc.

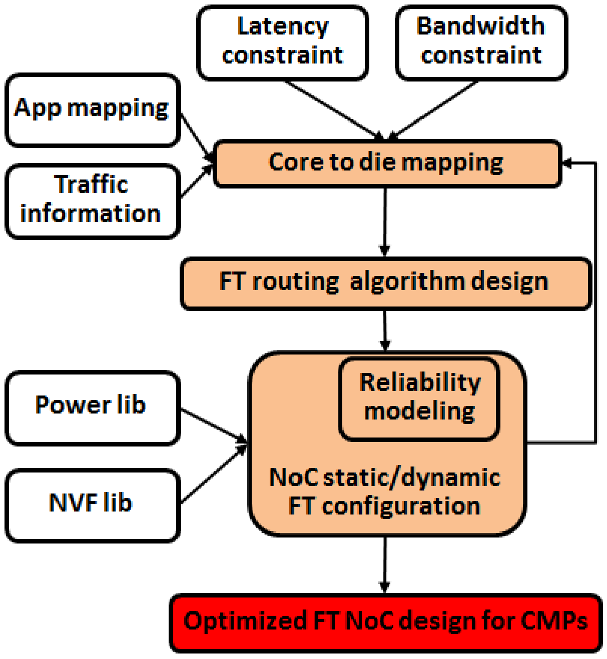


Figure 6 Reliability aware NoC synthesis framework

The ultimate goal of our research is to find a holistic fault-tolerant design for NoC communication fabrics in CMPs. As shown in Figure 6, in our research we want to find a mapping of cores to the 2D mesh-based die that meets application communication latency constraints and also configure (synthesize) the NoC fabric for the entire chip by customizing NoC routers at each node with appropriate protection mechanisms to meet a designer-specified reliability goal while minimizing energy consumption.

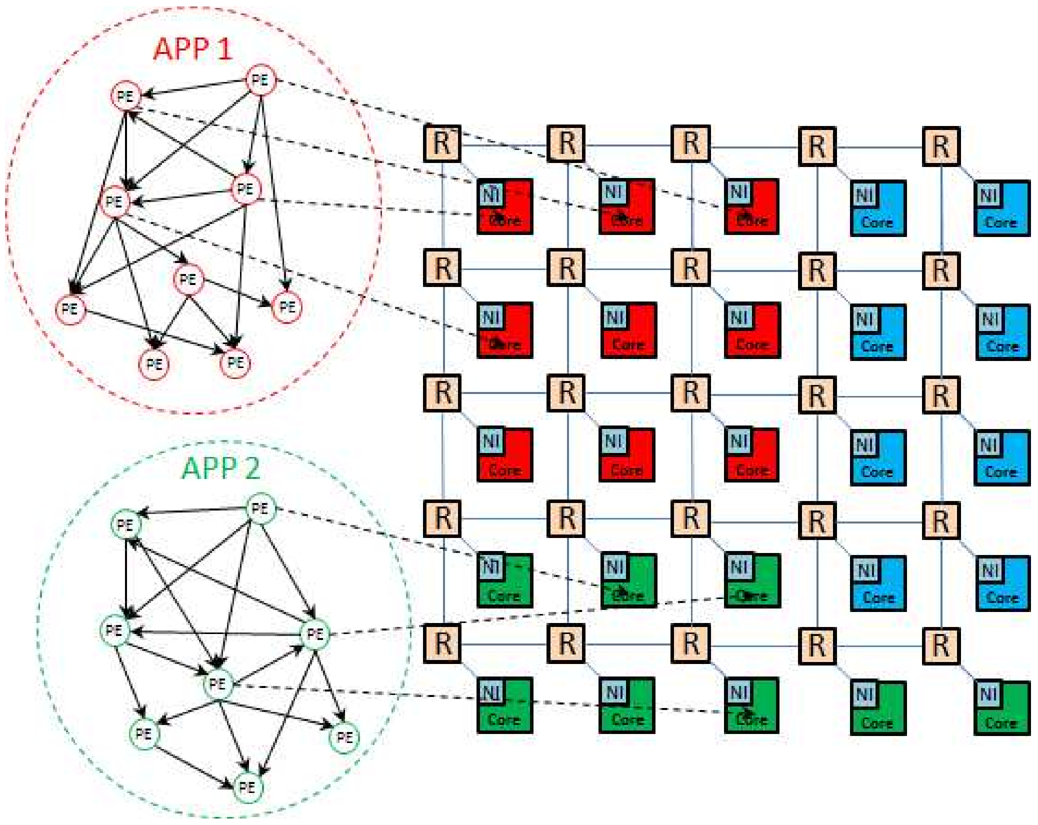


Figure 7 Core to tile mapping

Figure 7 shows an example of mapping each core in an application communication graph (ACG) to a CMP with a 2D mesh topology based NoC fabric. ACG is a directed graph, in which each vertex $C_i \in V$ represents an IP core (processor or memory), and each directed edge $comm_{ij} \in E$ represents communication dependencies between the source core C_i and destination core C_j. The volume of communication from C_i to C_j is $v(comm_{ij})$ and $v(C_i) = \sum_{j \in V}(v(comm_{ij}) + v(comm_{ji}))$. Each edge $comm_{ij}$ has a weight $B(comm_{ij})$ for application-specific communication bandwidth constraints, and a weight $L(comm_{ij})$ for latency constraints (maximum number of hops allowed between cores C_i and C_j), We assume that the designer can pre-define application zones on the mesh-based die, as shown in Figure 7, allowing multiple applications to co-execute on the regular CMP die. In general, communication occurs at an intra-application node level, i.e., only between nodes belonging to an application, and not between nodes of different applications. For a given NoC link width (e.g., 32 bits), we assume platform constraints related to the maximum bandwidth that the NoC links can support B_{link}, which is dependent on the CMOS technology node and chip power constraints. Thus, for all mapped cores utilizing a link, it is required that the total utilized bandwidth on that link $T_{link\_max} \leq B_{link}$.

The buffers in NoC routers are most susceptible to transient single event upset (SEU) errors, and we assume that errors on the relatively small footprint inter-router links are negligible. To protect NoC router buffers against SEU errors, we can employ two widely-used protection mechanisms: Hamming Error Correction Codes (HECC) and replicating components with Triple Modular Redundancy (TMR). Both mechanisms can correct SEU errors. But unlike TMR, HECC operation entails additional latency to encode and decode data. On the other hand, TMR implementations possess a larger area and power footprint than HECC. Table 1 shows the average power, latency, and area overhead trends for using either only HECC or only TMR to protect all buffers inside a NoC router, compared to a baseline NoC router without any protection. Results are obtained using an RTL model of a five staged pipelined NoC router that was enhanced with HECC and TMR protection mechanisms, and taken through logic synthesis using Synopsis tools and layout for a 45nm TSMC library.

Table 1 HECC vs. TMR overhead comparison

	HECC	**TMR**	**Integrated**
Power overhead	8%	259%	12%
Latency overhead	150%	10%	75%
Area overhead	5%	200%	7%

It is apparent that using only HECC or TMR has high associated penalties. To minimize these overheads, we chose to judiciously combine HECC and TMR in a NoC router, by protecting input FIFO buffers (that can store multiple flits) in NoC routers with HECC due to its lower area and power overhead, and protecting the smaller output NoC buffers (that only store a single flit) using TMR to minimize latency overhead. This integrated HECC/TMR configuration, shown in Figure 8 and whose overheads are shown in the last column of Table 1 ("Integrated"), results in a relatively fast packet traversal with a reasonable area and power overhead.

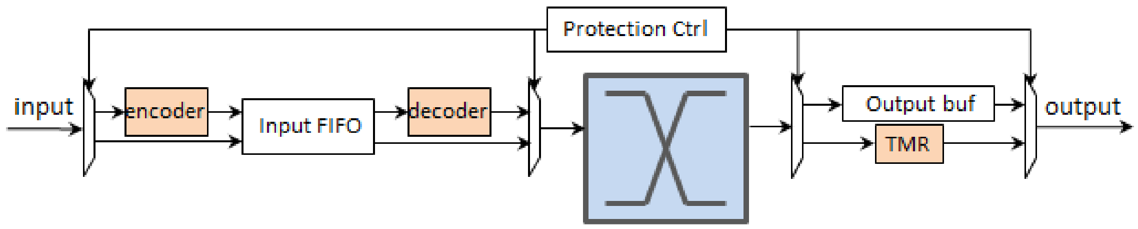


Figure 8 Router fault-tolerant configuration

But even in such a NoC router configuration, protecting all the buffers in all NoC routers on the die can lead to excessive power dissipation and latency overhead. There is an opportunity to further reduce power and latency by intelligently determining an appropriate subset of buffers for protection against SEUs in the NoC routers. To this end, we require a selective protection methodology, to best match the time-varying behavior of an application, and optimally reduce the cost of enabling fault tolerance in NoCs. Our research proposes intelligent mechanisms to perform such optimization.

1.6 Contribution

In this thesis, our contribution to the area of fault-tolerant NoC design encompasses three main topics:

- **Fault-tolerant routing algorithm design** which utilizes the path diversity in NoCs to achieve fault resilient transfers. The goal is to minimize the communication energy overhead and improving the packets' successful delivery rate, at the same time the proposed routing algorithms should be deadlock/livelock free.

- First, we proposed a novel fault-tolerant routing scheme (OE+IOE) for 2D mesh-based NoCs that can adapt to design-time and runtime permanent link faults, as well as potential intermittent faults in NoC communication architectures. Our scheme uses replication opportunistically based on the fault rate, and combines the odd-even (OE) and inverted odd-even (IOE) turn models to achieve deadlock free packet traversal. Experimental results show that our proposed OE+IOE scheme can provide better fault-tolerant ability (i.e., higher successful packet arrival rates) than traditional fault-tolerant routing schemes such as N-random walk.
- Based on OE+IOE, with more link fault information, we propose a novel low-overhead neighbor-aware turn model based fault-tolerant routing scheme (NARCO) for NoCs in embedded systems, which combines threshold based replication, a parameterizable region based neighbor awareness in routers, and the odd-even and inverted odd-even turn models. The proposed scheme enables a better packet arrival rate than state of the art of existing fault-tolerant routing schemes, while enabling a trade-off between communication reliability and energy overhead.
- By exploring more types of routing algorithms we propose a novel fault-tolerant routing scheme (NS-FTR) for NoCs that combines the North-last and South-last turn models to create a robust hybrid NoC routing scheme. The proposed scheme is shown to have a low implementation overhead and to adapt to design time and runtime faults better than the existing turn models, stochastic random walk schemes, and dual virtual channel based routing schemes such as XYX and OE+IOE.
- We then extend the hybrid routing algorithm approach for 3D NoCs and propose a novel fault-tolerant routing scheme (4NP-First) for 3D NoCs that combines the 4N-First and 4P-

First turn models to create a robust hybrid routing scheme. The proposed scheme is shown to have a low implementation overhead and adapts to design-time and runtime faults better than existing turn models, stochastic random walk schemes, and hybrid dual virtual channel based routing schemes.

- **Reliability modeling** based on our network vulnerability factor metric with which we quantify NoC reliability so that we can balance the energy and reliability. For the first time, a detailed characterization of vulnerability was performed on an AXI-based network interface (NI) architecture using full system simulation with consideration of thermal throttling, to determine the probabilities that faults in NI components manifest as errors in the final program output of the CMP. The resulting characterization can aid in the design of NI architectures possessing high reliability and low power dissipation overhead, using NVF-based opportunistic protection.
- **NoC system level synthesis** which is a holistic problem that includes the core to tile mapping, on-chip routing, and router fault tolerance design.

o First, we proposed a novel design-time framework (RESYN) to trade-off energy consumption and reliability in NoC-based CMPs. RESYN employs a nested genetic algorithm (NGA) approach to guide the mapping of cores on a die, and opportunistically determines the locations to insert fault tolerance mechanisms in NoC routers to minimize energy consumption while satisfying reliability constraints. Our experimental results show that RESYN can reduce communication energy costs by 14.5% on average compared to a fully protected NoC design, while still maintaining more than a 90% fault tolerance. RESYN also enables a comprehensive trade-off between reliability and energy-efficiency for more stringent reliability constraints, generating a Pareto set of solutions which may reveal opportunities for energy savings even for high reliability configurations; e.g., for a higher 95% reliability constraint, a notable 13% energy savings can still be achieved for some applications. Given the increasing importance of reliability in the nanometer era for NoC-based CMPs, this framework provides important perspectives that can guide the reduction of energy overheads associated with reliable NoC design.

- Second by considering NoC communication runtime information such as traffic congestion distribution and congestion patterns, we proposed a system-level framework called HEFT to trade-off energy consumption and fault-tolerance in the NoC fabric. Our hybrid framework tackled the challenge of enabling energy-efficient resilience in NoCs in two phases: at design time and at runtime. At design time, we implemented algorithms to guide the robust mapping of cores onto a die while satisfying application bandwidth and latency constraints. At runtime we devised a prediction technique to monitor and detect changes in fault susceptibility of NoC components, to intelligently balance energy consumption and reliability. Experimental results show that HEFT improves the energy/reliability ratio of synthesized solutions by 8-20%, while meeting application performance goals, when compared to prior work on reliable system-level NoC design. Given the increasing importance of reliability in the deep nanometer era for CMPs, our framework provides an important tool that can guide the reduction of energy overheads associated with reliable NoC design.

1.7 Thesis outline

The research presented in this thesis is organized as follows. Chapter 2 surveys recent literature in the domains of NoC fault-tolerant routing algorithm design, NoC reliability modeling, and NoC reliability aware synthesis. Chapter 3 lists our proposed fault-tolerant routing algorithms with the goal of minimizing energy, area overhead and at the same time optimizing the latency and reliability in terms of successful packet delivery rates. Chapter 4 describes our proposed reliability model based on the network vulnerability factor (NVF) metric for NoC communication subsystems. In Chapter 5, we propose two system-level frameworks, RESYN and HEFT, to trade-off energy consumption and fault-tolerance in the NoC fabric by exploring different core to tile mapping, routing algorithms and fault-tolerant NoC protection configuration strategies. RESYN explores an offline router protection strategy while HEFT contains both offline and online strategies for NoC synthesis. Chapter 6 summarizes the conclusion of this thesis and lists some directions for future work.

CHAPTER 2
LITERATURE SURVEY

In general our research focuses on three main topics related to fault-tolerant NoC design: fault-tolerant NoC routing, NoC reliability modeling and NoC reliability aware synthesis. This chapter discusses the advantages and drawbacks of the most influential existing research work in literature in each of these three areas.

2.1 Fault-tolerant NoC routing

On-chip interconnect architectures are particularly susceptible to faults that can corrupt transmitted data or prevent it from reaching its destination [14]. Reliability concerns in UDSM nodes have in part contributed to the shift from traditional bus-based communication fabrics to network-on-chip (NoC) architectures that provide better scalability, performance, and utilization than buses [15][16]. The inherent redundancy in NoCs due to multiple paths between packet sources and sinks can greatly improve communication fault resiliency. To overcome permanent faults in NoCs, fault-tolerant routing schemes are a critical requirement and the focus of several research efforts over the last few years [27]-[45]. In the presence of intermittent or permanent faults on NoC links and routers, routing schemes can ensure error free data flit delivery by using an alternate route that is free of faults.

NoC routing schemes can be broadly classified as either static (also called deterministic) or dynamic (also called adaptive). While static routing schemes [1][26] use fixed paths and offer no fault resiliency, dynamic (or adaptive) routing schemes [27]-[45] can alter the path between a source and its destination over time as traffic conditions and the fault distribution changes. The design of adaptive routing schemes is mainly concerned with increasing flit arrival rate, avoiding deadlock, and trying to use a minimal hop path from the source to the destination to decrease transmission energy. Unfortunately, these goals are often conflicting, requiring a complex trade-off analysis that is rarely addressed in existing literature.

More specifically, fault-tolerant routing schemes can be functionally classified into three categories: (i) stochastic, (ii) fully adaptive, and (iii) partially adaptive.

Stochastic routing algorithms provide fault tolerance through data redundancy by replicating packets multiple times and sending them over different routes [27]. For instance, the probabilistic gossip flooding scheme [28] allows a router to forward a packet to any of its neighbors with some pre-determined probability. Directed flooding refines this idea by preferring hops that bring the packet closer to its addressed destination [29]. N-random walk [29] limits the number of packet copies by allowing N replications at the source only. In the rest of the network, these copies stochastically take different routes without further replication. The major challenges with these approaches are their high energy consumption, strong likelihood of deadlock and livelock, overhead of calculating, storing, and updating probability values, and poor performance even at low traffic congestion levels.

Fully adaptive routing schemes make use of routing tables in every router or network interface (NI) to reflect the runtime state of the NoC and periodically update the tables when link or router failures occur to aid in adapting the flit path. For instance, [30] proposes using source routing, with routing tables stored at NIs instead of in the routers. But the scheme is only useful for design time faults and cannot adapt to runtime faults. In [31][32], routing tables are used in every router to determine the best path to take in the presence of faults at runtime in mesh topologies. There are several issues with these approaches that prevent their widespread use. First, these schemes require that the routing schemes be updated with global fault and route information frequently, which can take hundreds to thousands of cycles at runtime during which the network state is unstable. In addition, obtaining a global view of the system may not be practically feasible from an energy and performance point of view. Second, the cost and scalability of these schemes can make them prohibitive. As the size of the NoC grows, the table sizes also increase rapidly, increasing router (or NI) area, energy, and latency. Finally, deadlock is a strong possibility that can bring the entire system to a halt, unless high ooverhead deadlock recovery mechanisms such as escape channels or distributed timeout counters are used.

Partially adaptive routing schemes enable a limited degree of adaptivity, placing various restrictions and requirements on routing around faulty nodes, primarily to avoid deadlocks. Turn model based routing algorithms such as negative-first, north-last, and south-last [33]-[36] are examples of partially adaptive routing, where certain flit turns are forbidden to avoid deadlock. The odd-even (OE) turn model [36] classifies columns in a mesh topology as either odd or even and prevents different turns in the odd and even columns to prevent cyclic dependencies that can cause deadlocks. In [29], the XY and YX schemes are combined to achieve fault resilient transfers. In [38], routing tables are combined with a customized turn model to avoid deadlock. The work is primarily aimed at design time permanent faults. Unfortunately, the degree of adaptivity provided by these routing algorithms is highly unbalanced across the network, which in some cases results in poor performance.

A few works have proposed using convex or rectangular fault regions with turn models [39]-[45]. Routing detours around these regions can be selected so that deadlock-prone cyclic dependencies are impossible, such as by using turn models or cycle free contours. For instance, [40] builds a fault ring around faults and introduces rules to route flits around the fault region to avoid deadlock. But the scheme can only handle design time permanent faults (i.e., fault regions are built only at design time and cannot handle runtime faults) and also requires up to four VCs, which can quickly become cost ineffective as the NoC size scales. In [41], the OE turn model was used together with fault regions. However, the work was based on assumptions that no fault exists in the two columns that are adjacent to the west and east edges of the mesh. In general, fault-tolerant routing schemes that make use of fault regions are either too conservative (disabling fully functional routers to meet region shape requirements), have restrictions on the locations of the faults that can be bypassed, or cannot adapt to runtime intermittent and permanent faults.

2.2 NoC reliability modeling

A major challenge facing the designers of NoC architectures is the increased likelihood of failure due to the rise in transient faults (or soft errors). Such faults are caused by a variety of factors (e.g., high-energy cosmic neutron or alpha particle strikes, capacitive and inductive crosstalk) that are becoming more and more prevalent with the exponential growth rate of on-chip transistors, along with lower supply voltage and increasing clock frequency. Soft errors cause the deposit or removal of enough charge to invert the state of a transistor, wire, or storage cell, which can lead to catastrophic system failure [9]. Techniques to ensure fault tolerance have thus become a critical design concern in recent CMP designs. Chip designers typically set a failure rate target for each design and strive to maximize performance subject to this constraint. To validate that a design meets the failure rate target, designers perform extensive pre- and post-silicon analysis. This analysis measures the rate at which faults occur in a system as well as whether a given fault will cause a system failure. Designers then make use of one of several well-known techniques to improve reliability in the presence of soft errors, such as triple modular redundancy (TMR) and error correction/detection coding schemes. However, these techniques come with extra overhead in terms of area, power, and even performance, and therefore must be used judiciously.

Fortunately, not all faults eventually affect the final program outcome. For example, a bit flip in an empty translation lookaside buffer (TLB) entry will not cause any effect in the program execution. Based on this observation, Mukherjee et al. [55] defined a structure‘s Architectural Vulnerability Factor (AVF) as the probability that a transient fault in the structure finally produces a visible error in the output of a program. At any point of time, a structure‘s AVF can be derived via counting all the important bits that are required for Architecturally Correct Execution (ACE) in the structure, and dividing them by the total number of bits of the structure. Such ACE analysis has been used to derive an upper bound on AVF using performance simulation for intra-processor buffers [55]-[57] and cache structures [58] in recent years.

There are two main approaches to calculate AVF values: ACE analysis and Statistical Fault Injection (SFI). The former provides a (tight, if the underlying system is appropriately modeled [63]) lower bound on the reliability level of various processor structures, and has been adopted in many research works on performance models. Fu et al. [57] quantitatively characterized vulnerability phase behavior of four processor microarchitecture structures based on a system framework proposed in [56]. Zhang et al. [64] performed a similar analysis on SMT architectures. Soundararajan et al. [65] described a simple infrastructure to estimate an upper bound of the processor reorder buffer AVF, and also proposed two mechanisms (Dispatch Throttling and Selective Redundancy) to restrict the vulnerability to any limit. Sridharan at el. decomposed AVF into two parts: a software-dependent portion and a hardware-dependent portion, known as Program Vulnerability Factor (PVF) [66] and Hardware Vulnerability Factor

(HVF) [67], respectively. Biswas et al. measured AVF for L2 cache structures [58]. None of these prior works has explored vulnerability for components of NoC communication fabrics.

Several works have addressed the problem of making NoC fabrics more reliable. For instance, Kim et al. [61] explored error correction/detection and retransmission techniques to handle errors in links and router logic. More specifically for router buffers, Yu et al. [68] proposed using Hamming codes while Frantz et al. [69] proposed using TMR together with Hamming codes to address errors. Chang et al. [70] proposed using spare routers to handle NoC router failure at runtime. Pasricha et al. [59] presented a hybrid turn model based fault-tolerant routing algorithm that adapts to faults at runtime on NoC links and routers. Refan et al. [60] proposed adding spare links from each core to its neighboring router, to allow packets to bypass failed switches. None of these existing works attempt to understand the underlying vulnerability of NoC components to guide efforts that improve reliability.

In our research, for the first time, we introduce the concept of a network vulnerability factor (NVF) to quantify and model reliability and then perform a detailed NVF analysis for sub-components in a state-of-the-art AXI protocol based NI architecture, using full system simulation, to understand masking effects and determine the probabilities that faults in NI sub-components manifest as errors in the final program output in a CMP system.

2.3 NoC reliability aware synthesis

Tradition system level synthesis for NoC can involve many design sub-problems such as core-to-tile mapping, routing algorithm design, task mapping/scheduling, etc. The problem of synthesizing NoC communication architectures has been addressed extensively in literature. Some of the more comprehensive efforts also perform mapping of cores to a die, which can significantly improve communication power dissipation and latency, by optimizing the flow of traffic between cores. Here we briefly present some representative examples of prior work in the area.

In [87], Ye et al. propose a model for energy consumption of NoC routers called "bit energy model" that defines the energy consumption for one bit of data transfer through a NoC router and link. Using this bit energy model, In [82], a branch and bound algorithm is proposed to map processing cores on to a mesh NoC to satisfy bandwidth constraints and minimize total energy consumption. In [83], Hu et al. proposed a multi-commodity flow based scheme to optimize NoC power consumption by customizing the topology and packet routes. In [98], Srinivasan et al. use a slicing tree algorithm to search the core mapping space and arrive at a mapping that adheres to latency constraints with the goal of minimizing energy. In [14], Chou et al. introduce a traffic contention aware core mapping algorithm using integer linear programming. In [84], Leung et al. focus on NoC implementations with voltage islands and use a genetic algorithm (GA) to find an energy-aware voltage island partitioning and assignment solution. In [79], Ascia et al. analyze and evaluate the performance of different mapping solutions in a mesh NoC topology and solve the power-performance multi-objective mapping problem with a heuristics based on GA and branch-and-bound. In [80], Murali et al. present a greedy core-to-die mapping heuristic for a regular NoC-based MPSoC die, based on communication volumes emanating from each core. Their approach also searches different routing paths to balance NoC traffic and meet bandwidth constraints, but it does not include energy cost as an optimization goal. None of the existing techniques performs core-mapping and NoC synthesis together with reliability goals, nor do they attempt to trade-off reliability with energy.

Due to reliability becoming an increasingly important goal for NoC design, some recent efforts are considering reliability as part of an overall optimization objective. In [99], Aisopos et al. use

Monte Carlo simulations to model process variations and calculate reliability values for circuits. In [29], Pirretti et al. use a successful data delivery rate metric to quantify reliability and evaluate the fault tolerance of their NoC routing algorithms. In [96], Ababei et al. define the reliability of a NoC based on the path lengths between communicating cores on the die. They use this reliability model and use the core-to-die mapping framework from [95] to balance NoC reliability and energy costs.

In our research, we propose a novel design-time framework (RESYN) and a hybrid design/run time framework (HEFT) to trade-off energy consumption and reliability in the NoC fabric at the system level for CMPs that improves upon these prior efforts.

CHAPTER 3
FAULT-TOLERANT ROUTING ALGORITHM DESIGN

In this chapter we propose different low overhead fault-tolerant routing schemes for 2D and 3D NoC mesh topology. Our goal is to achieve better fault tolerance than existing routing algorithms in the presence of different types of faults in the NoC; at the same time we want to minimize the energy cost of routing.

3.1 OE+IOE fault-tolerant routing scheme

In this section, we present our novel low overhead fault-tolerant routing scheme (OE+IOE) that combines the odd-even (OE) and inverted odd-even (IOE) turn model based routing schemes to achieve better fault tolerance than existing fault-tolerant routing schemes. The target topology for the scheme is the 2D mesh, which is the most popular interconnection topology in NoCs today because of its layout efficiency, predictable electrical properties, and simple core addressing. The proposed scheme uses replication and dual virtual channels (VCs) at ports opportunistically to optimize the balance between energy overhead and flit arrival rate.

3.1.1 OE and inverted OE turn models

The odd-even (OE) turn model for deadlock-free routing in 2D meshes was introduced by Chiu [36]. A turn in this context refers to a 90-degree change of traveling direction for a flit. There are eight types of turns that are possible in a 2D mesh based on the traveling directions of the associated channels. To facilitate the explanation, we label the four sides of the 2D mesh as East, West, South, and North (Figure 9). Then a turn is called a NE turn if it involves a flit traveling in the North direction and attempting to turn East. Similarly, we can define the other seven types of turns, namely EN, WS, WN, SE, SW, ES, and NW turns, where E, W, S, and N indicate East, West, South, and North, respectively.

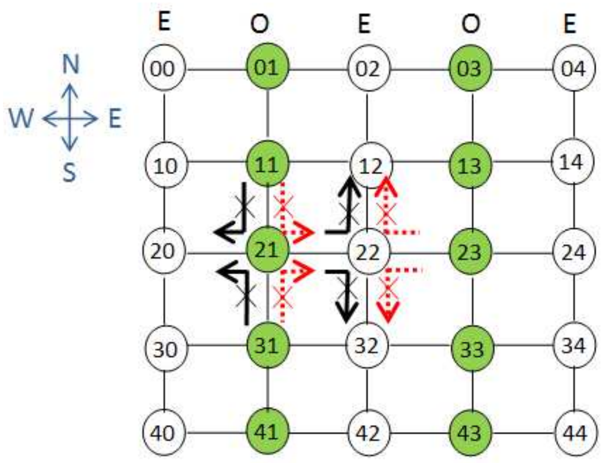


Figure 9 OE and IOE prohibited turns based on column; black arrows indicate prohibited turns in OE, while red arrows indicate prohibited turns in IOE

Deadlock in wormhole routing is caused by flits waiting on each other in a cycle. Unlike traditional turn models [1] that avoid deadlock by prohibiting certain turns, the OE turn model restricts the locations at which certain turns can occur to ensure that a circular wait does not occur. In OE turn model based routing, columns in a 2D mesh are alternately designated as odd (O) and even (E), as shown in Figure 9 for a 5×5 2D mesh. Then the following two forbidden turn rules ensure cycle- and deadlock-free routing: *(i) a packet is not allowed to take an EN or ES turn at any node located in an even column; and (ii) a packet is not allowed to take a NW or SW turn at any node located in an odd column.* A deadlock-free minimal path route can then be found for any source destination pair as described in [36], and we refer the reader to that paper for details. The prohibited turns for the OE turn model are depicted in black in Figure 9. No other restrictions need to be applied to ensure freedom from deadlock. The inverted odd-even (IOE) turn model can be understood by rotating the mesh by 180 degrees, while preserving the odd-even column designations and corresponding prohibited turns. Figure 9 shows the prohibited turns in the odd and even columns for the IOE turn model in red, which are required to ensure deadlock free operation.

3.1.2 OE+IOE routing scheme: overview

To ensure robustness against faults, redundancy is a necessary requirement, especially for environments with high fault rates and unpredictable fault distributions. As the level of redundancy is increased, system reliability improves as a general rule. However, redundancy also detrimentally impacts other design objectives such as power and performance. Therefore in practice it is important to limit redundancy to achieve a reasonable trade-off between reliability, power, and performance. Unlike directed and random probabilistic flooding algorithms that propagate multiple copies of a packet to achieve fault-tolerant routing in NoCs, the proposed OE+IOE routing algorithm sends only one redundant packet for each transmitted packet, and only if the fault rate is above a replication threshold δ. The value of δ is a designer-specified parameter that depends on the application characteristics, routing complexity, and power-reliability trade-off requirements. The original packet is sent using the odd-even (OE) turn model while the redundant packet is propagated using an inverted odd-even (IOE) turn model based routing scheme. This packet replication happens only at the source. Two separate VCs, one dedicated to the OE packets and the other for the IOE packets ensure deadlock freedom. If the fault rate is low (i.e., below δ), replication is not utilized and only the original packet is sent using the OE scheme while power/clock gating the IOE virtual channel to save power. The proposed routing algorithm prioritizes minimal paths that have higher probabilities of reaching the destination even if faults are encountered downstream. Minimal paths and the replication threshold ensure low power dissipation under a diverse set of fault rates and distributions. No restriction on the number or location of faults is assumed, but the routers must know which of their adjacent (neighbor) links/nodes are faulty, which is accomplished using minimal control signaling. The proposed routing approach can be combined with error control coding (ECC) techniques for transient fault resiliency and optimizations such as router buffer reordering [46] and router/NI backup paths [47] to create a comprehensive fault-tolerant NoC fabric. In the following subsections, we describe the implementation of the proposed OE+IOE routing scheme.

3.1.2.1 Turn restriction checks

Whenever a packet arrives at a router in the OE+IOE scheme, three scenarios are possible: (i) there is no fault in the adjacent links and the packet is routed to the output port based on the OE (or IOE) scheme that selects a minimal path to the destination, (ii) there are one or more faults on adjacent links that prevent the packet from propagating in a valid direction (i.e., an output port with a fault-free adjacent link that does not violate turn model routing rules) in which case the packet must be dropped as a back turn is not allowed, and (iii) one or more adjacent links have faults but a valid direction exists to route the packet towards, based on the OE (or IOE) rules. A packet traversing an intermediate router must choose among four output directions (N, S, W, E). However, choosing some directions may lead to a violation of a basic OE (or IOE) turn rule immediately or downstream based on the relative location of the destination. Figure 10 shows the SystemC [48] pseudocode of the turn restriction check phase for the OE turn model in our router model to aid in detecting an invalid routing direction. The reference to XY coordinates follows the convention as shown in Figure 9 where each node is labeled with its XY coordinates. Note that the pseudocode for the IOE scheme is similar with the directions inverted, and is not shown here for brevity.

First, we check whether the output port direction has a fault in its attached adjacent link (Steps 2-4), in which case this is an invalid direction. Next, we check the forbidden turn rules for the OE turn model as discussed in Section 3.1 (Steps 5-10) based on the router location (in an odd or even column), the input port of the packet, and its output port direction. If the packet is attempting a forbidden turn, then the direction is invalid. If the direction has no adjacent faults and does not violate the basic OE routing rules, we then check if the direction will lead to a turn rule violation downstream based on the location of the destination. We start by checking for the scenario when the packet is attempting to go in the East direction (Steps 11-18). If the destination is on an even column (dest_yco%2 = 0) and the current packet is on its left neighbor column but not in the same row as the destination, then the packet should not be routed to the East direction (Steps 12-14). This is because after it arrives at the destination column, it has to route in the North or South direction to reach the destination row, which however breaks the basic OE turn rules and will not be allowed. Alternatively, if the destination is to the West of the current node, then the East direction is also forbidden (Steps 15-17). This is because once the packet is routed in the East direction, it cannot go back in the West direction without violating the basic OE turn rules (if it arrives at an even column and wants to go West, it must go North or South which breaks the basic OE turn rules; if it arrives at an odd column then after turning North or South it must go West which also breaks the basic OE turn rules).

Next we check for the scenario where the destination is on the even column and the current packet is on its left neighbor column and attempting to go in a direction other than the East direction (Steps 19-27). In this case, when the current node is not on the same row as the destination, then the only choice is to go North or South towards that destination row. If the current node is on the same row as the destination, then it can only choose the East direction. The next check is for the scenario where the current node is on an odd column and the destination is to its West direction (Steps 28-29). In this case, the North and South directions are both forbidden. This is because if a packet goes in the North or South directions, it will ultimately need to go West, which is not possible without violating the basic OE turn rules. Next, we check for the scenario where the current node is on the same column as the destination, but the current node is also on the West border of the mesh or on an odd column (Steps 30-36). In this case if the packet moves North or South in a direction away from the destination, it will not be able to turn West on the West border or make a subsequent West turn when in an odd column to reach the destination without violating basic OE turn rules. Finally, we check for a back turn, which is not allowed (Steps 37-39). If all these checks pass, then the given direction is valid for sending the packet (Step 40).

Pseudocode: Turn Restrictions for OE Implementation

```
1:  RESULT   check_neighbor (UI ip_dir, UI direct, ULL dest_id) {
    //ip_dir: direction where the packet comes from
    //direct: direction we want to check
    //cur_id: current node position
    //dest_id: destination node position
    //dest_yco, dest_xco: y,x coordinates of destination node
    //cur_yco,cur_xco: y,x coordinates of current node
    //dif_yco=dest_yco - cur_yco
    //RESULT: LINK_FAULT, TURN_FAULT, BACK_TURN, OK
2:     if (check_linkfault(direct)==LINK_FAULT){
3:         return LINK_FAULT;
4:     }
5:     if ((cur_yco%2==0)&&(ip_dir==E)&&(direct==N||direct==S)){) {
6:             return TURN_FAULT;
7:     }
8:     if ((cur_yco%2==1)&&(ip_dir==N||ip_dir==S)&&(direct==W)) {
9:             return TURN_FAULT;
10:    }
11:    if (direct==E) {
12:        if ((dest_xco!=cur_xco)&&(dif_yco==1)&&(dest_yco%2==0)) {
13:            return TURN_FAULT;
14:        }
15:        if (dest_yco<=cur_yco) {
16:            return TURN_FAULT;
17:        }
18:    }
19:    if ((dif_yco==1)&&(dest_yco%2==0)) {
20:        if ((dest_xco<cur_xco)&&(direct==S)) {
21:            return TURN_FAULT;
22:        } else if ((dest_xco>cur_xco)&&(direct==N)) {
23:            return TURN_FAULT;
24:        } else if ((dest_xco==cur_xco)&&(direct==N|| direct==S|| direct==W)) {
25:          return TURN_FAULT;
26:        }
27:    }
28:    if ((dest_yco<cur_yco)&&(cur_yco%2!=0)&&((direct==N)||(direct==S)))
29:          return TURN_FAULT;
30:    if (((borderW(cur_id))||(cur_yco%2!=0))&&(cur_yco==dest_yco)) {
```

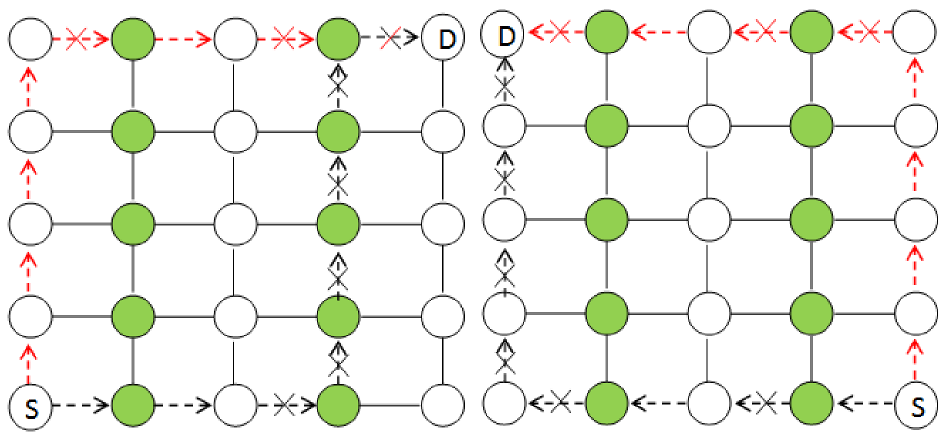
```
            if ((cur_xco<dest_xco)&&(direct==N)) {
             return TURN_FAULT;
            } else if ((cur_xco>dest_xco)&&(direct==S)) {
             return TURN_FAULT;
            }
    }
    if (ip_dir==direct) {
            return BACK_TURN;
     }
     return OK;
}
```

Figure 10 SystemC pseudocode for OE turn restrictions

The turn restriction check procedure described above is invoked in parallel for all four possible directions in an intermediate node for a header flit. We created a gate level implementation of the circuit and synthesized it to obtain power and latency overhead of the scheme. At the TSMC 65nm process technology, we observed that our implementation has low overhead, dissipating a nominal 2.03 μW of power on average, and with a critical path latency of 0.54 ns.

3.1.2.2 Prioritized valid path selection

After the turn restriction checks, it is possible for a packet to have multiple valid directions that it can follow from its current location. To ensure low energy transfers, directions in minimal paths are always given higher priority in our scheme. If, however, two valid directions in minimal paths exist to the destination from the current router, the goal is to give higher priority to the direction that allows for greater path diversity so that the packet can still route around faults downstream and reach its destination. In the OE (or IOE) turn models, path diversity is strongly dependent on the location of destination. In our proposed OE+IOE scheme, given two minimal paths to the destination, *we prioritize the North and South directions over the East and West directions*.



(a) (b)

Figure 11 Choosing minimal paths (a) destination to the north-east of source, (b) destination to the north-west of source

Consider Figure 11(a) which shows a case where the destination is to the North-East of the current (in this case source) node. Two valid minimal paths exist from source S to destination D – one to the North (red dashed line) and the other to the East (black dashed line). We use the convention of white nodes lying in the even column and green nodes lying in the odd column. If the East minimal path encounters any one of the six faults shown with an 'X', the packet will have to be dropped as it cannot reach the destination. In contrast, with a North minimal path, there are only three possible faults that will cause the packet to be dropped; faults in any of the other links can be routed around using another valid alternate path. Thus, there is a greater probability of the packet reaching its destination if the North direction is given greater priority over the East direction. Figure 11(b) shows the case where the destination is to the North-West of the source node, with the North minimal path again having higher probability of delivering the packet to the destination compared to the West minimal path. Similar arguments hold for cases when the destination is to the South-East and South-West of the source, with South minimal paths providing greater probability of successful packet arrival than if the West or East minimal paths are chosen.

3.1.2.3 Stochastic valid path selection

An alternative to using prioritized minimal paths during OE or IOE routing is to stochastically choose one of the available valid paths. In this method, every outgoing channel in a router is assigned a probability value: PN, PS, PE, and PW, where the sum of all probabilities is 1. We assume that valid paths have an equal probability of being selected. For instance, if a packet enters a router from the South port and a West turn is forbidden, then PS = 0 (back turn not allowed), PW = 0 (forbidden turn), and therefore PE = PN = 0.5. Thus the North or the East directions can be chosen with equal probability. The usefulness of this technique is in enabling the selection of non-minimal valid paths, which in some cases may lead to higher arrival rate, at the cost of higher energy consumption. We contrast the prioritized and stochastic path selection techniques in our experiments, presented in Section 4.

3.1.3 Router architecture

Figure 12 depicts our virtual channel (VC) router architecture. Packets traversing the network have a header flit with a destination ID field that is used to determine the output port at each router. The router datapath consists of the buffers and the switch. The input FIFO buffers store

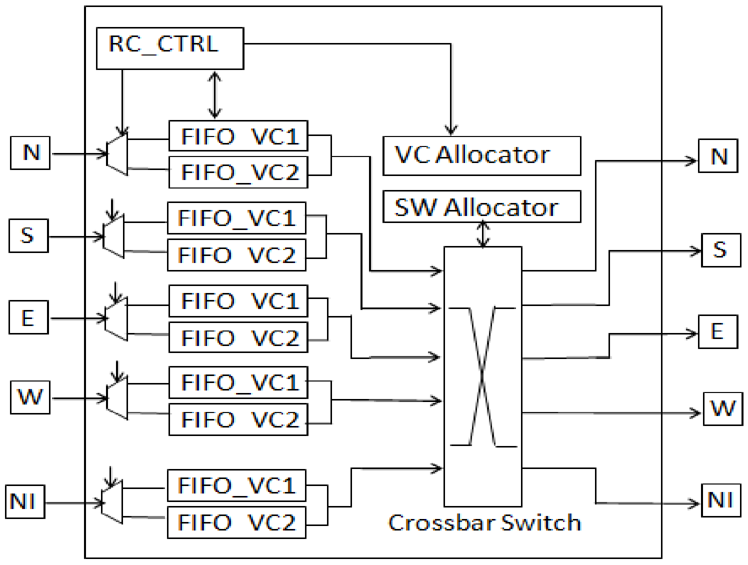


Figure 12 Router architecture

up to 16 flits waiting to be forwarded to the next hop. There are two input FIFO buffers each dedicated to a VC, with one VC for the OE routed packets and the other for the IOE routed packets. When a flit is ready to move, the switch connects an input buffer to an appropriate output channel. To control the datapath, the router contains three major control modules: a route compute control unit (RC_CTRL), a virtual-channel (VCA) allocator, and a switch allocator (SA). These control modules determine the next hop direction, the next VC, and when a switch is available for each packet/flit. The routing operation takes four phases: route computation (RC), virtual-channel allocation (VCA), switch allocation (SA), and switch traversal (ST). When a head flit (the first flit of a packet) arrives at an input channel, the router stores the flit in the buffer for the allocated VC and determines the next hop for the packet using the checks and priority rules described in the previous sections (RC phase). Given the next hop, the router then allocates a virtual channel (VCA phase). Finally, the flit competes for a switch (SA phase) if the next hop can accept the flit, and moves to the output port (ST phase). In our implementation, a packet carries its virtual channel (OE or IOE) information in a bit in the header flit, therefore the output port and the destination VC can be obtained at the end of the RC stage.

To reduce energy consumption, if the fault rate is below the replication threshold δ, the RC_CTRL unit shuts down the IOE virtual channels in the router. We assume that dedicated control signals connected to a lightweight fault detection unit (FDU) can allow estimation of the fault rate at runtime in the NoC, and the decision to change the replication status can be communicated to the nodes and implemented with a delay of a few hundred cycles. Typically, such a change in status would be a rare occurrence as permanent faults do not appear at runtime very frequently.

An important requirement for any fault-tolerant NoC fabric is a control network that can be used to send acknowledgement signals to inform the source whether the packet successfully arrived at the destination. We assume a lightweight and fault-free control network that is topologically similar to the data network for routing ACK signals from the destination to the source node on successful packet arrival, or NACK signals from an intermediate node to the source node if the packet must be dropped due to faults that make it impossible to make progress towards the destination. In our implementation, a source can re-send a packet at most twice after receiving NACK signals, before assuming that the packet can never reach its destination. While this can signal unavoidable failure for certain types of applications, other applications may still be able to operate and maintain graceful degradation. For instance, a multiple use-case CMP can shut down certain application use-cases when their associated inter-core communication paths encounter fatal faults, but there might be other use cases that can still continue to operate on fault-free paths. We assume that a higher level protocol (e.g. from the middleware or OS levels) can step in to initiate thread migration and other strategies to ensure useful operation in the presence of unavoidable system faults.

3.1.4 Experimental studies

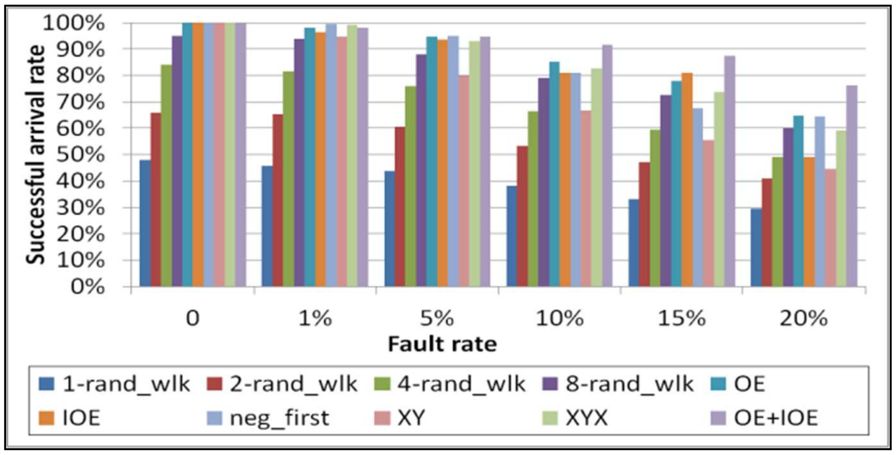
In this section we present the results of our experimental studies. Section 4.1 describes our experimental setup. Section 4.2 compares the proposed OE+IOE routing scheme with several fault-tolerant routing schemes proposed in literature. Finally, Section 4.3 contrasts variants of our proposed fault-tolerant routing scheme.

3.1.4.1 Experimental setup

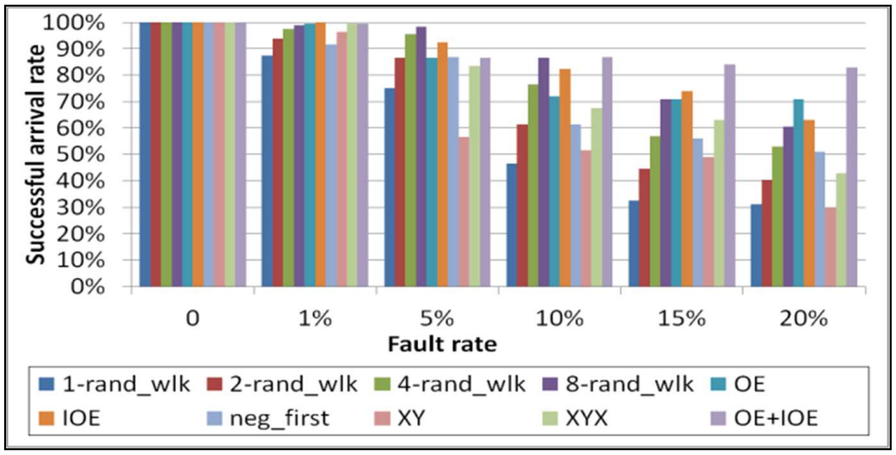
We implemented and evaluated our proposed algorithm using a SystemC based cycle-accurate 2D mesh NoC simulator that was created by extending the open source Nirgam simulator [49] to support faults and different fault-tolerant routing schemes. In addition to communication performance estimates, we were interested in NoC power and energy estimation as well. For this purpose we incorporated dynamic and leakage power models from Orion 2.0 [50] into the simulator. The NoC fabric was clocked at 1 GHz. The target implementation technology was 65nm, and this node was used to determine parameters for delay and energy consumption. Two different core complexities were considered for our analysis: 6×6 (36 cores) and 9×9 (81 cores). The stimulus for the simulations was generated for specific injection rates using three different traffic models: uniform random, transpose, and hotspot. A fixed number of flits (3000/core) were transmitted according to the specified traffic patterns in the simulations, to enable comparisons for not only successful arrival rates, but also for overall communication energy. Faults in the NoC were modeled as randomly distributed link-level hard failures that can represent design time (fabrication related) or runtime (wearout related) failures. Ten different random fault distributions were generated and analyzed for each fault rate, and for each traffic type to obtain a more accurate estimate of fault resiliency for the routing schemes. The faults can cause routers to be only partially functional, or entirely non-functional if all of the links of a router are faulty. Both of these scenarios were tested in the simulations.

3.1.4.2 Comparison with existing FT routing schemes

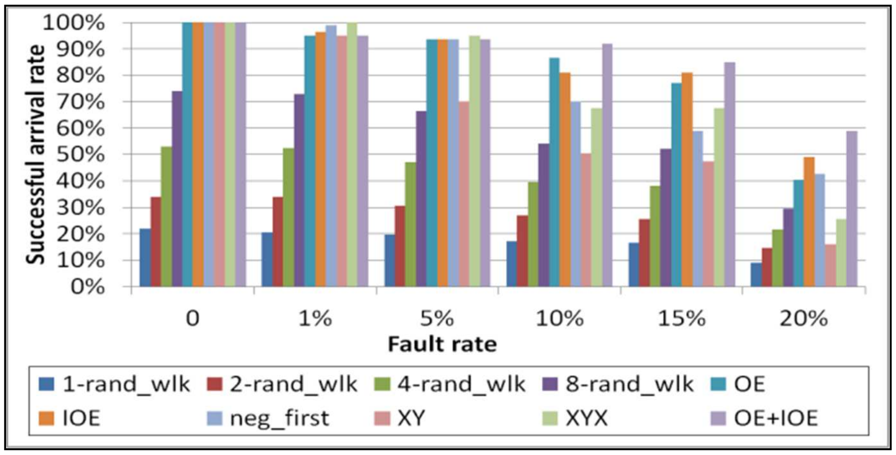
To demonstrate the effectiveness of our proposed OE+IOE fault-tolerant routing scheme for NoCs, we compared it to several existing fault-tolerant routing schemes from literature. The following schemes were considered in the comparison study: (i) XY dimension order routing [1] (although not inherently fault-tolerant, it is considered here because of its widespread use in 2D mesh NoCs), (ii) N-random walk [29] for N = 1,2,4,8, (iii) adaptive negative first turn model [34], (iv) adaptive OE turn model [36], (v) adaptive IOE turn model, and (vi) XYX [37] which combines replication with the XY and YX routing schemes on two VCs. Other fault-tolerant routing schemes such as probabilistic flooding [28] and fully adaptive table based routing [31] were considered but their results are not presented because these schemes have major practical implementation concerns- their energy consumption is an order of magnitude higher than other schemes and frequent deadlocks hamper overall performance. For our OE+IOE scheme, we used a prioritized valid path selection (Section 3.2.2) and a replication threshold value of δ = 6%, based on our extensive simulation studies that showed a good trade-off between arrival rate and energy for 8% ≥ δ ≥ 2%.



(a)



(b)

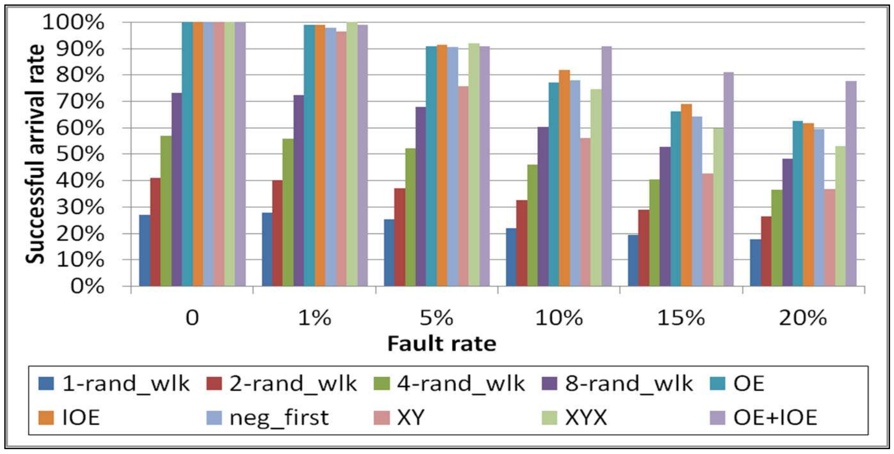


(c)

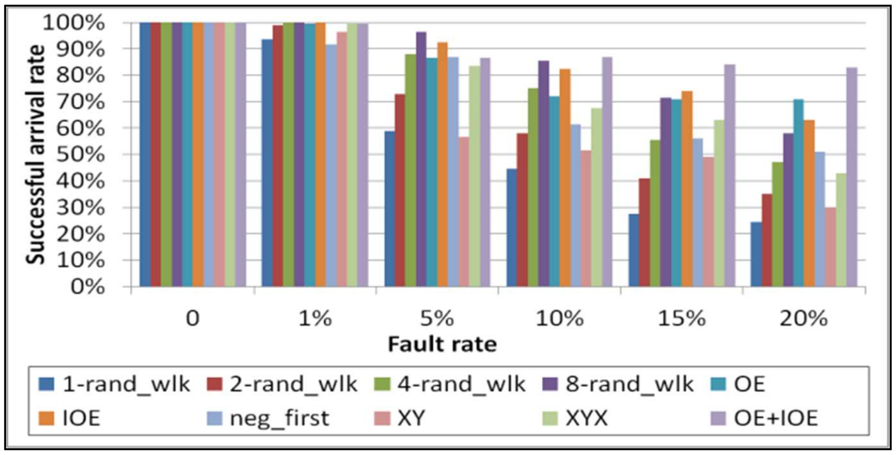
Figure 13 Packet arrival rate for 6×6 NoC (a) Uniform random traffic, (b) Hotspot traffic, (c) Transpose traffic

Figure 13(a)-(c) compares the successful packet arrival rate for various fault-tolerant routing schemes under different fault rates. Results are shown for the three traffic types, a 6×6 NoC, and a 20% injection rate (0.2 flits/node/cycle). It can be seen that the N-random walk schemes have a low arrival rate even in the absence of faults, for the uniform and transpose traffic types. This is due to frequent deadlocks during simulation which the scheme is susceptible to, causing packets to be dropped. As hotspot traffic has fewer actively transmitting cores than the other traffic types, the arrival rates for N-random walk schemes are a little better for low fault rates (as there is lower probability of deadlock) but get worse as the fault rate increases. The best arrival rate for the N-random walk scheme is achieved for a value of N=8. Out of the turn/dimension-order based routing schemes, XY not surprisingly performs worse than the other schemes due to the lack of any built in fault tolerance capability. For low fault rates, OE, IOE, and negative first have a comparable successful arrival rate, but for higher fault rates OE and IOE on average perform better than the negative first scheme. Interestingly, the dual virtual channel XYX scheme performs better than other schemes for low fault rates (~1%), but for higher fault rates, its successful arrival rates are on average lower than those for single virtual channel OE, IOE, and negative first turn model based schemes! The lack of path diversity in the XYX scheme leads to this dramatic reduction in arrival rate as the number of faults in the network increase. Our proposed OE+IOE fault-tolerant routing scheme can be seen to have a much higher successful packet arrival rate compared to the other schemes, especially for higher fault rates. The results indicate that the OE+IOE scheme can scale well with the higher fault rates that are expected to be the norm in future CMP designs. To explore how the fault-tolerant schemes fare as the size of the CMP increases, we repeated the experiments on a 9×9 NoC.

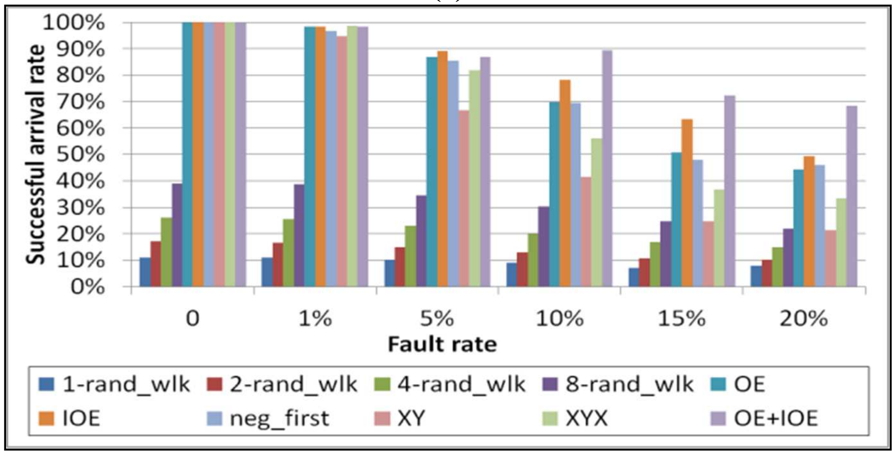
Figure **14**(a)-(c) shows the successful packet arrival rates for the same set of routing schemes under different fault rates and a 20% injection rate, for a 9×9 NoC. Once again it can be seen that the OE+IOE scheme has a greater fault tolerance than the other schemes, showing how well the scheme scales with increasing levels of core integration. Results for other injection rates are not shown here because as long as the fault distribution and source-destination pairs remain the same, changing the injection rate does not impact the successful packet arrival rate (although it does impact energy and average packet latency).



(a)



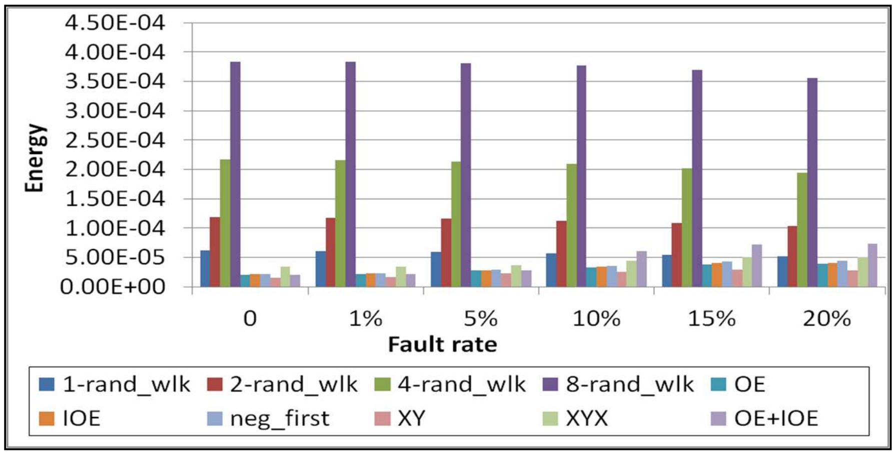
(b)



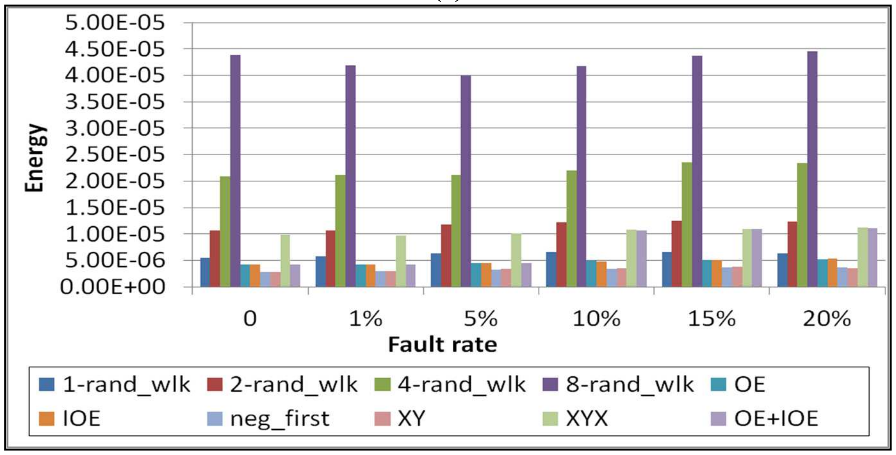
(c)

Figure 14 Packet arrival rate for 9×9 NoC (a) Uniform random traffic, (b) Hotspot traffic, (c) Transpose traffic

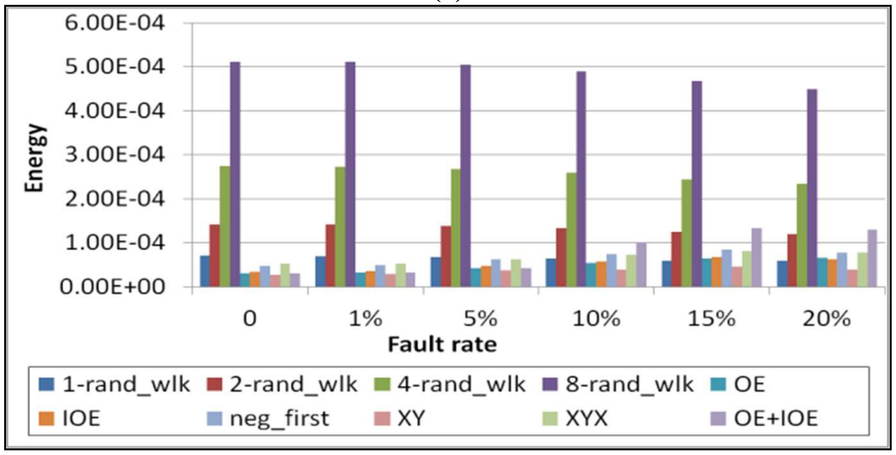
In addition to arrival rate, an increasingly important metric of concern for designers is the energy consumption in the communication network. Figure 15(a)-(c) shows the energy consumption (in Joules) for the various routing schemes under different fault rates, for a 9×9 NoC with a 20% injection rate. Results for a smaller 6×6 NoC are omitted for brevity as they follow a similar trend to the 9×9 NoC. It can be seen that the N-random walk fault-tolerant routing schemes have significantly higher energy consumption compared to the other schemes. This is due to the high level of replication with increasing values of N, as well as because of the non-minimal paths chosen to route the packets on. For single VC schemes (OE, IOE, XY, and negative first), the XY scheme has the lowest energy dissipation which explains its popularity in NoCs today. But as it is lacking in any fault tolerance, the scheme may not be a viable option for future CMPs. The negative first scheme has lower energy consumption than the OE and IOE schemes on average for hotspot traffic, and higher energy consumption on average for uniform random and transpose traffic scenarios. The XYX scheme makes use of two VCs and replication, therefore it does not come as a surprise to see that it has higher energy consumption than the single VC schemes. Our proposed OE+IOE scheme uses the replication threshold δ to trade-off energy consumption with fault tolerance. For the chosen value of δ=6%, our scheme utilizes a single VC and no replication for low fault rates (< 6%) and then switches to its dual VC mode with replication for higher fault rates. This explains the low energy consumption for low fault rates for the OE+IOE scheme in Figure 15. For higher fault rates (≥ 10%), the OE+IOE energy consumption is comparable to that of XYX for hotspot traffic, and somewhat higher than XYX for uniform random and transpose traffic. But the higher energy consumption for OE+IOE is an artifact of its much higher successful packet arrival rate. The overall energy consumption for the scheme is still significantly lower than other fault-tolerant schemes such as stochastic flooding, N-random walk and table based source or distributed routing. If lower energy is desirable, the value of the replication threshold can be increased by designers. For instance, for multimedia applications, occasionally dropped pixel data may not be perceivable by viewers, and in such scenarios the resulting lower energy consumption may lead to a much longer battery life and better overall user experience.



(a)

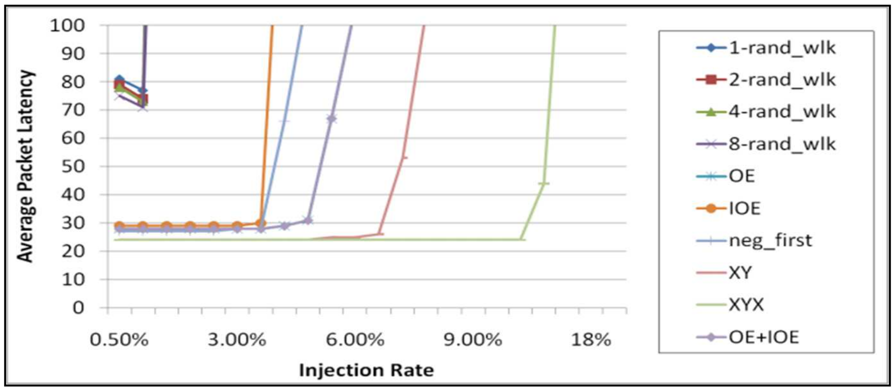


(b)

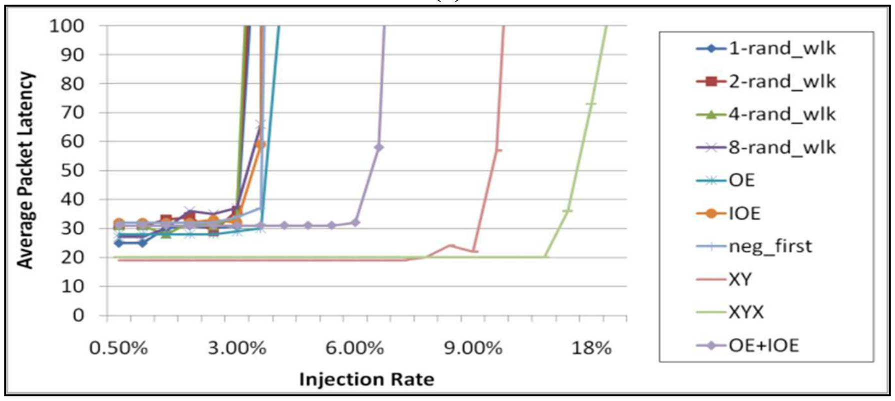


(c)

Figure 15 Communication energy for 9x9 NoC (a) Uniform random traffic, (b) Hotspot traffic, (c) Transpose traffic



(a)



(b)

Figure 16 Average packet latency for 9x9 NoC with uniform random traffic (a) 1% fault rate, (b) 20% fault rate

Figure 16(a)-(b) summarizes the average packet latency (in cycles) for the routing schemes for various injection rates under the uniform random traffic model. The results are shown for a 9×9 NoC with low fault rate (1%) and high fault rate (20%). The lowest packet latency is achieved for the XY scheme for low injection rates and the XYX scheme for higher injection rates, which are both minimal path schemes. The latency of the XYX scheme is lower than that of the XY scheme for high injection rates because the XY scheme encounters more faults at high injection rates that cause it to drop packets more frequently than the XYX scheme which in some cases still manages to deliver packets using its YX route even if the XY route has a fault in it. The OE+IOE scheme has the next lowest latency on average, which coupled with its higher successful arrival rate compared to the XY, XYX, and the rest of the schemes makes it an extremely viable alternative for fault-tolerant routing in future NoC based CMPs.

3.1.4.3 Comparison with OE+IOE variants

Next, we compare our OE+IOE scheme that uses prioritized path selection and selective replication with two of its variants: *(i)* OE+IOE with prioritized path selection and without selective replication (OE+IOE_no_δ), and *(ii)* OE+IOE with stochastic valid path selection and with selective replication (OE+IOE_S).

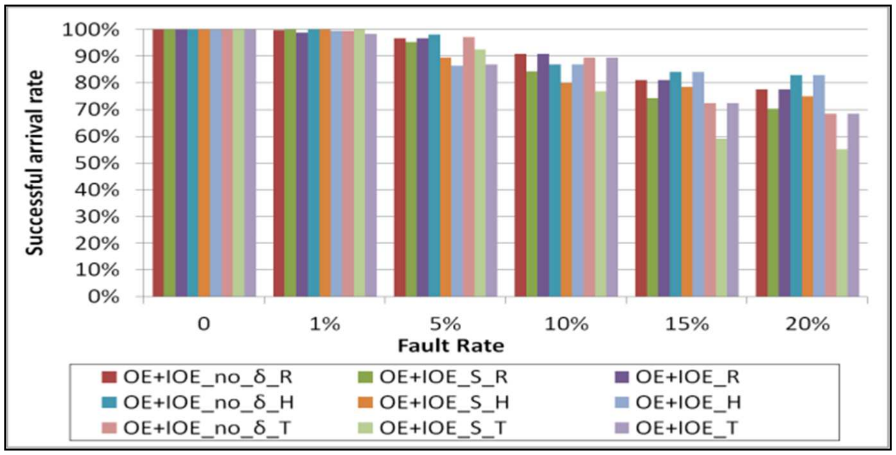


Figure 17 shows the successful packet arrival rate for our proposed scheme (OE+IOE) compared to the OE+IOE_S and OE+IOE_no_δ variants for different fault rates and the three traffic types, for a 9×9 NoC and 20% injection rate. The highest successful packet arrival rate on average is obtained for the OE+IOE_no_δ scheme which always uses replication and dual VCs. The OE+IOE scheme has a similar behavior and hence packet arrival rate compared to the OE+IOE_no_δ scheme for fault rates higher than the δ threshold. But for lower fault rates, our emphasis on lower energy consumption with the OE+IOE scheme translates into a slightly lower arrival rate than OE+IOE_no_δ. Using stochastic valid path selection in the OE+IOE_S scheme leads to higher arrival rates than the OE+IOE scheme only for low fault rates below δ. For higher fault rates, it can be clearly seen that stochastic valid path selection is not as efficient as prioritized valid path selection.

Figure 18 compares the energy consumption (in Joules) for the OE+IOE, OE+IOE_S, and OE+IOE_no_δ schemes under different fault rates for the same 9×9 NoC and traffic scenarios as above. The extremely low energy consumption of the OE+IOE scheme compared to its variants for low fault rates is a strong motivation for the existence of the replication threshold parameter. In general, the OE+IOE_S scheme employs more non-minimal paths than the other schemes which leads to longer packet latencies and higher overall energy consumption. For high fault

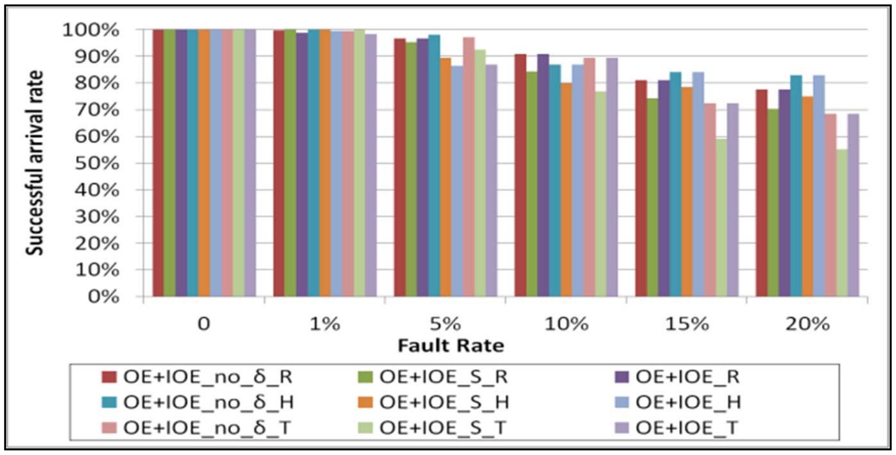


Figure 17 Packet arrival rate for 9x9 NoC for three OE+IOE variants under uniform random traffic (R), hotspot traffic (H), and transpose traffic (T)

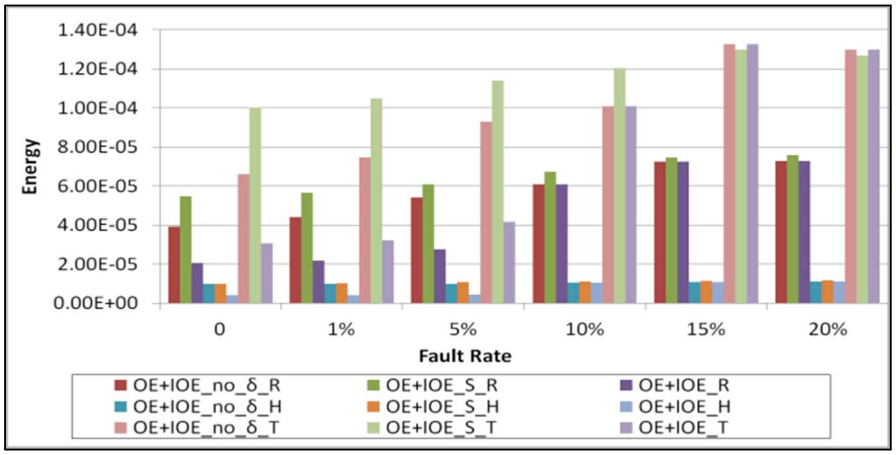


Figure 18 Communication energy for 9x9 NoC for three OE+IOE variants under uniform random traffic (R), hotspot traffic (H), and transpose traffic (T)

rates under transpose traffic, the OE+IOE_S energy consumption is lower, but only because it has a lower packet arrival rate with many more dropped packets before they can complete their routes. Thus, it can be seen that OE+IOE not only provides much higher fault tolerance compared to existing fault-tolerant routing schemes, but also enables a trade-off between reliability and energy consumption that can be tuned on a per-application basis by the designer.

3.1.5 Summary

Rapidly scaling CMOS technology and ever increasing levels of core integration on-chip multiprocessors (CMPs) have led to an increase in transient, intermittent, and permanent faults that can lead to chip failure. There is thus a critical need today to focus on fault-tolerant design techniques to overcome the detrimental impact of faults on emerging CMP designs. In this section we proposed a novel fault-tolerant routing scheme (OE+IOE) for 2D mesh NoCs that can adapt to design time and runtime permanent link faults, as well as potential intermittent faults in NoC communication architectures. Our scheme uses replication opportunistically based on fault rate, and combines the odd-even (OE) and inverted odd-even (IOE) turn models to achieve deadlock free packet traversal. Experimental results show that our proposed OE+IOE scheme can provide better fault tolerance (i.e., higher successful packet arrival rates) than traditional fault-tolerant routing schemes such as N-random walk and turn model based schemes.

3.2 NARCO neighbor aware fault-tolerant routing

OE+IOE makes use of the connecting links information to make fault-tolerant routing decision. In lots of situation, if a router can get more link fault information within the NoC by using extra control logic, the router can make better routing decision. In this section, we propose a novel low overhead neighbor aware turn model based fault-tolerant routing scheme (NARCO) for 2D mesh based NoCs. Similar to OE+IOE, NARCO utilizes threshold based replication which balances energy dissipation with the need to replicate packets to bypass permanent faults in routes. Different from OE+IOE however, NARCO uses extra control logic to monitor more links fault condition, so it can achieve even better fault tolerance than OE+IOE. Our experimental results and comparison studies indicate that the NARCO outperforms existing turn model, stochastic random walk, dual virtual channel based routing schemes that have been proposed in literature, as well as the adaptive OE+IOE routing scheme proposed in section 3.1.

3.2.1 NARCO: fault-tolerant routing overview

In this section we present an overview of the NARCO fault-tolerant routing scheme for 2D mesh NoCs. Section 3.2.1.1 presents details of the implementation and operation of the proposed NARCO routing scheme. Section 3.2.1.2 describes how neighbor awareness can be used to make more informed routing decision.

3.2.1.1 Routing algorithm implementation details

Similar to OE+IOE, NARCO transmits only one redundant packet for each transmitted packet, and only if the fault rate is above a replication threshold δ. The original packet is sent using the odd-even (OE) turn model while the redundant packet is propagated using an inverted odd-even (IOE) turn model scheme. This packet replication happens only at the source network interface. Two separate virtual channels (VCs), one dedicated to the OE packets and the other for the IOE packets ensure deadlock freedom. If the fault rate is low (i.e., below δ), replication is not utilized to save energy. Moreover, with NARCO the routers can detect more link faults from their adjacent (neighboring) links/nodes than in OE+IOE, with the help of extra light-weight control signals.

Pseudocode: Turn Restrictions for OE Implementation

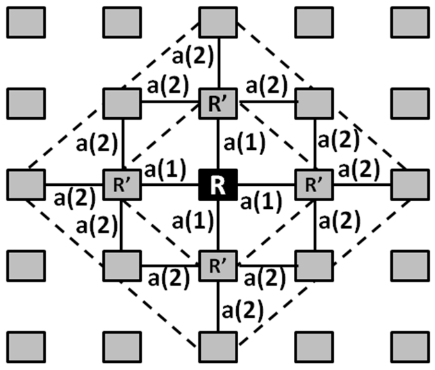
```
1:  RESULT   check_neighbor(UI ip_dir, UI direct, ULL dest_id) {
      //ip_dir: direction where the packet comes from
      //direct: direction we want to check
      //cur_id: current node position
```

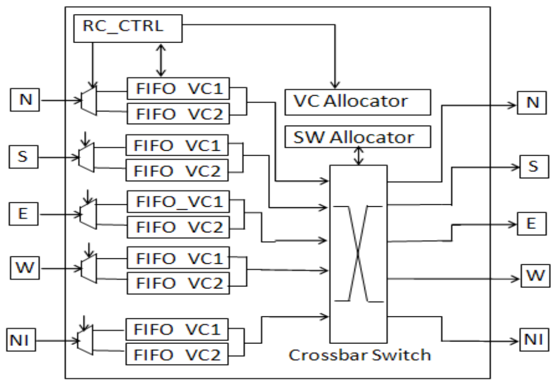
```
    //dest_id: destination node position.
    //dest_yco,dest_xco: y,x coordinates of destination node
    //cur_yco,cur_xco: y,x coordinates of current node
    //dif_yco=dest_yco-cur_yco
    //RESULT: LINK_FAULT, TURN_FAULT, BACK_TURN, OK
    if (check_linkfault(direct)==LINK_FAULT ){
            return LINK_FAULT;
    }
    if ((cur_yco%2==0)&&(ip_dir==E)&&(direct==N||direct==S)){) {
            return TURN_FAULT;
    }
    if ((cur_yco%2==1)&&(ip_dir==N||ip_dir==S)&&(direct==W)) {
            return TURN_FAULT;
    }
    if (direct==E) {
        if ((dest_xco!=cur_xco)&&(dif_yco==1)&&(dest_yco%2==0)) {
            return TURN_FAULT;
        }
        if (dest_yco<=cur_yco) {
            return TURN_FAULT;
        }
    }
    if ((dif_yco==1)&&(dest_yco%2==0)) {
        if ((dest_xco<cur_xco)&&(direct==S)) {
            return TURN_FAULT;
        } else if ((dest_xco>cur_xco)&&(direct==N)) {
            return TURN_FAULT;
        } else if ((dest_xco==cur_xco)&&(direct==N|| direct==S|| direct==W)) {
              return TURN_FAULT;
        }
    }
    if ((dest_yco<cur_yco)&&(cur_yco%2!=0)&&((direct==N)||(direct==S)))
                return TURN_FAULT;
    if (((borderW(cur_id))||(cur_yco%2!=0))&&(cur_yco==dest_yco)){
         if ((cur_xco<dest_xco)&&(direct==N)){
                return TURN_FAULT;
         } else if ((cur_xco>dest_xco)&&(direct==S)){
                return TURN_FAULT;
         }
    }
    if (ip_dir==direct){
            return BACK_TURN;
    }
            return OK;
}
```

Figure 19 SystemC pseudocode for OE turn restrictions

Figure 19 shows the SystemC pseudocode of the checks for the OE turn model in our router model to aid in detecting an invalid routing direction. The procedure is invoked in parallel for all directions in a router node. Note that the pseudocode for the IOE scheme is similar, and is not shown here for brevity. A high level overview of the pseudocode is as follows. We first check whether the output port direction has a fault in its attached adjacent link (Steps 2-4), in which case this is an invalid direction. Next, we check the restricted turn rules for the turn models as discussed in Section A (Steps 5-10) based on the router location (in an odd or even column), the input port of the packet, and its output port direction. If the packet is attempting a forbidden turn, then the direction is invalid. If the direction has no adjacent faults and does not violate the basic OE routing rules, we then check if the direction will lead to a turn rule violation downstream based on the location of the destination (Steps 11-36). Finally, we check for a back turn, which is not allowed (Steps 37-39). If all these checks pass, then the given direction is valid for packet transfer (Step 40).

After the turn restriction checks, it is possible for a packet to have multiple valid directions that it can follow from its current location. To ensure low energy transfers, directions in minimal paths are always given higher priority in our scheme. If, however, two valid directions in minimal paths exist to the destination from the current router, we give higher priority to the direction that allows for greater path diversity so that the packet can still route around faults downstream. In NARCO, given two minimal paths to the destination, *we prioritize the North and South directions over the East and West directions.*





(a) **(b)**

Figure 20 (a) Neighborhood awareness (b) Router architecture

3.2.1.2 Neighbor awareness and router architecture

The pseudocode for our implementation in the previous subsection assumed that each router knows the fault status of its adjacent links. To reduce the probability of encountering a fault downstream when forwarding a packet from a router, we incorporate neighbor awareness into our router. Figure 20(a) shows the adjacent links (i.e., adjacency = 1) for a router R with the links designated as a(1). Neighbor awareness can be achieved with a small buffer in the router for the purpose of keeping the fault status in the router's neighborhood beyond its adjacent links. For instance, a buffer of size 16 bits is sufficient to keep a fault status of links with adjacency levels 1 (a(1)) and 2 (a(2)). The size of the neighborhood is customizable, but should not be too large as a large size entails a larger buffer size and a more complex fault status signaling network that can impact overall performance and energy consumption. The fault status buffer is initialized at startup and then periodically updated every 500K cycles to account for runtime permanent faults that may (in practice infrequently) arise in the network.

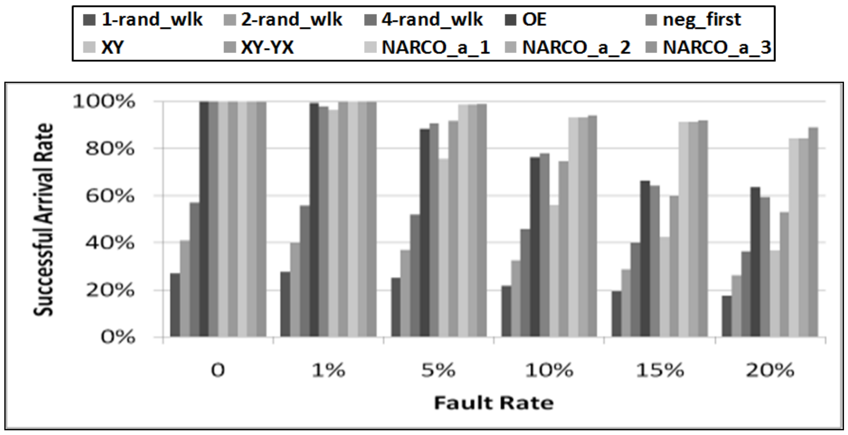
Figure 20(b) shows the architecture of our router with the fault-tolerant routing and neighbor awareness incorporated in the route compute control (RC_CTRL) unit. The input FIFO buffers store up to 16 flits waiting to be forwarded to the next hop. There are two input FIFO buffers each dedicated to a VC, with one VC for the OE routed packets and the other for the IOE routed packets. To reduce energy consumption, if the fault rate is below the replication threshold δ, the RC_CTRL unit shuts down the IOE virtual channels in the router.

3.2.2 Experimental results

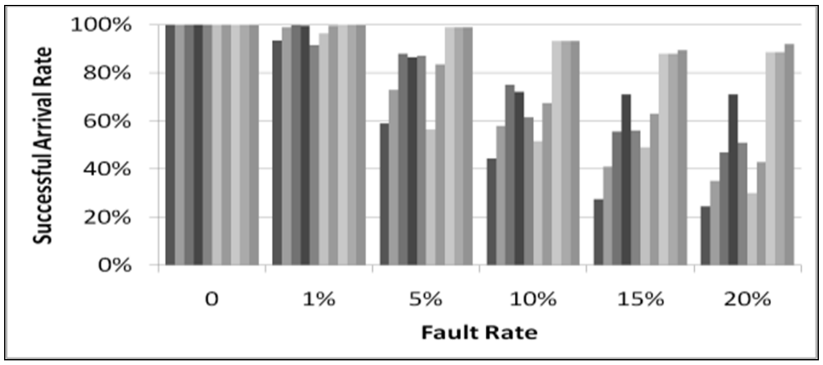
3.2.2.1 Experimental setup

We extended the open source Nirgam NoC simulator to support faults and different fault-tolerant routing schemes, and incorporated dynamic and leakage power models for NoC components. For accurate power and delay analysis of the NARCO algorithm, we developed a gate level implementation of the algorithm and included delay and power results. A 2D mesh of size 9×9 (81 cores) was considered in our analysis. The NoC fabric was clocked at a frequency of 1 GHz, with a CMOS implementation technology of 65nm. Three different traffic models were analyzed: uniform random, transpose, and hotspot. Faults in the NoC were modeled as randomly distributed link-level hard failures that can represent design time or runtime failures. Ten different random fault distributions were generated and analyzed for each fault rate, and for each traffic type.

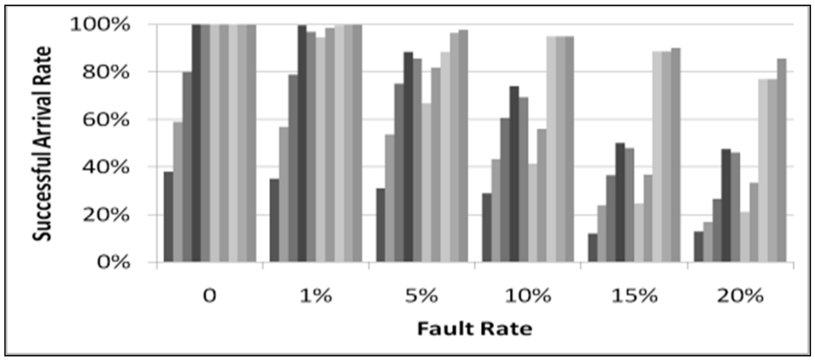
3.2.2.2 Comparison with existing FT routing schemes



(a)



(b)



(c)

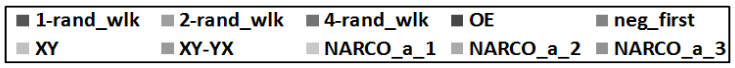
Figure 21 Packet arrival rate for 9×9 NoC (a) Uniform random traffic, (b) Hotspot traffic, (c) Transpose traffic

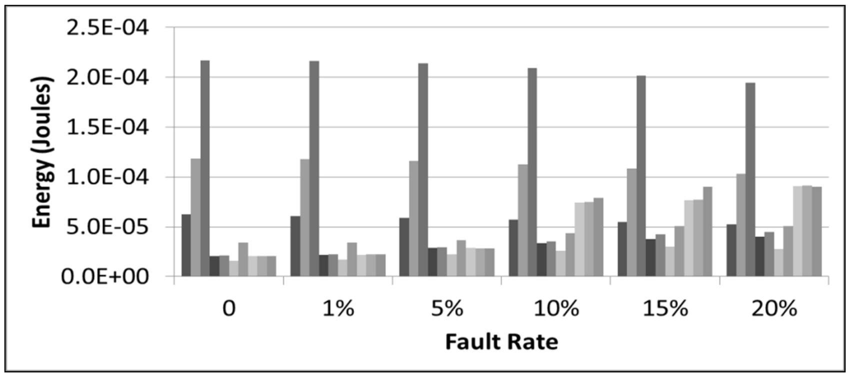
To demonstrate the effectiveness of the NARCO fault-tolerant routing scheme for NoCs, we compared it to several existing fault-tolerant routing schemes from literature. The following schemes were considered in the comparison study: *(i)* XY dimension order routing *(ii)* N-random walk for N = 1,2,4 [29] *(iii)* negative first turn model [33], *(iv)* odd even (OE) turn model [36], *(v)* XY-YX which combines replication with the XY and YX routing schemes on two VCs [37], and *(vi)* our proposed NARCO scheme with neighborhood adjacency of 1, 2, and 3. A replication threshold value of δ = 6% was chosen for the NARCO scheme, based on our extensive simulation studies that showed a good trade-off between arrival rate and energy for 8% ≥ δ ≥ 2%. Results are shown for the three traffic types and a 20% injection rate (0.2 flits/node/cycle).

Figure 21(a)-(c) compare the successful packet arrival rate for the various fault-tolerant routing schemes under different fault rates. It can be seen that the configurations of our proposed fault-tolerant routing scheme (NARCO_a_1, NARCO_a_2, NARCO_a_3) have a much higher successful packet arrival rate compared to the other schemes, especially for higher fault rates. In addition, having a higher neighbor awareness for high fault rate environments ensures a better packet arrival rate.

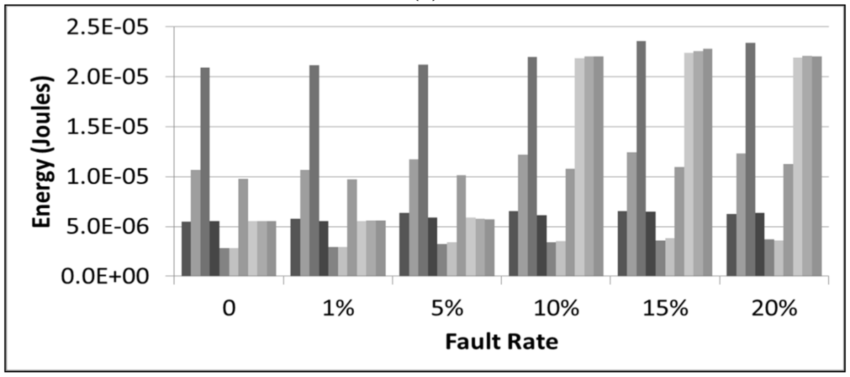
Energy consumption for fault-tolerant routing schemes is an important metric, especially in embedded systems often running on a fixed battery budget.

Figure 22(a)-(c) show the energy consumption of the NoC (in Joules) for the different routing schemes, under different fault rates. It can be seen that the N-random walk fault-tolerant routing schemes have significantly higher energy consumption compared to the other schemes. This is due to the high level of replication with increasing values of N, as well as because of the non-minimal paths chosen to route the packets on. Single VC schemes (XY, negative first, OE) have a low energy cost, but possess low fault tolerance (Figure 21) making them not very viable for future CMPs. The XY-YX scheme makes use of two VCs, so it comes as no surprise that it has higher energy consumption than the single VC schemes. The extremely low energy consumption of the NARCO configurations for low fault rates (comparable to the single VC schemes) is a strong motivation for the existence of the replication threshold parameter. For higher fault rates, the NARCO configurations have higher energy consumption than the XY-YX and single VC schemes, but this energy overhead ensures a significantly higher packet arrival rate compared to the other schemes, as can be seen from Figure 21. Thus it can be seen that the proposed NARCO fault-tolerant routing scheme not only provides a much higher fault tolerance compared to existing routing schemes, but also enables a trade-off between reliability and energy consumption that can be tuned on a per-application basis by the designer by setting the replication threshold and neighbor awareness parameters.

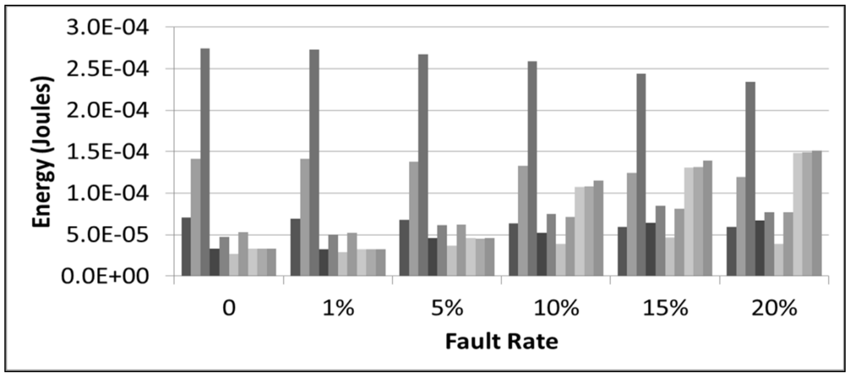




(a)



(b)



(c)

Figure 22 Communication energy for 9x9 NoC (a) Uniform random traffic, (b) Hotspot traffic, (c) Transpose traffic

3.2.3 Summary

In this section we proposed a novel low overhead neighbor aware turn model based fault-tolerant routing scheme (NARCO) for NoCs in embedded systems, which combines threshold based replication, a parameterizable region based neighbor awareness in routers, and the odd-even and inverted odd-even turn models. NARCO enables better packet arrival rate than state of the art, while enabling a trade-off between communication reliability and energy overhead.

3.3 NS-FTR fault-tolerant routing

Until now our hybrid routing algorithms have been based on the OE and IOE turn models. The reason we chose this combination is because IOE is a good complement for the OE algorithm enabling high cumulative routing path coverage. In this section, we propose a new fault-tolerant routing scheme (NS-FTR) for mesh NoCs that combines the North-last and South-last turn models together with opportunistic replication to increase flit arrival rate while optimizing power dissipation. For the first time, we explore the performance of fault-tolerant routing schemes in the presence of both runtime intermittent and design time permanent faults. Compared to our OE+IOE scheme, the proposed NS-FTR algorithm has much lower implementation overhead. Our extensive experimental results indicate that NS-FTR outperforms OE+IOE as well as other state-of-the-art NoC fault-tolerant routing schemes based on single and hybrid turn models, and stochastic random walk heuristics.

3.3.1 NS-FTR routing scheme

In this section we present an overview of our proposed fault-tolerant routing scheme (NS-FTR) for 2D mesh NoCs. Section 3.3.1.1 introduces north last (NL) and south last (SL) turn model routing. Section 3.3.1.2 presents details of the implementation and operation of the proposed routing scheme that combines the NL and SL turn models. Finally, Section 3.3.1.3 describes the router architecture and control network that supports our NS-FTR in a NoC fabric.

3.3.1.1 NL and SL turn models

The north last (NL) turn model for deadlock-free routing in large scale 2D mesh networks was first introduced by Glass et al. [33]. A turn in this context refers to a 90-degree change of traveling direction for a flit. There are eight types of turns that are possible in a 2D mesh based on the traveling directions of the associated flits. A deadlock in wormhole routing can occur because of flits waiting on each other in a cycle. In the NL turn model cycle- and deadlock- free routing is achieved by prohibiting two out of the eight possible turns, as shown in Figure 23(a). A flit in the NL turn model is initially routed in the E, W, or S directions before finally turning in the N direction, after which it cannot make further turns. In a similar manner, the south last (SL) turn model ensures deadlock free routing by prohibiting two out of the eight possible turns as shown in Figure 23(b). A flit in the SL turn model is initially routed in the E, W, or N directions before finally turning in the S direction, after which it cannot make further turns.

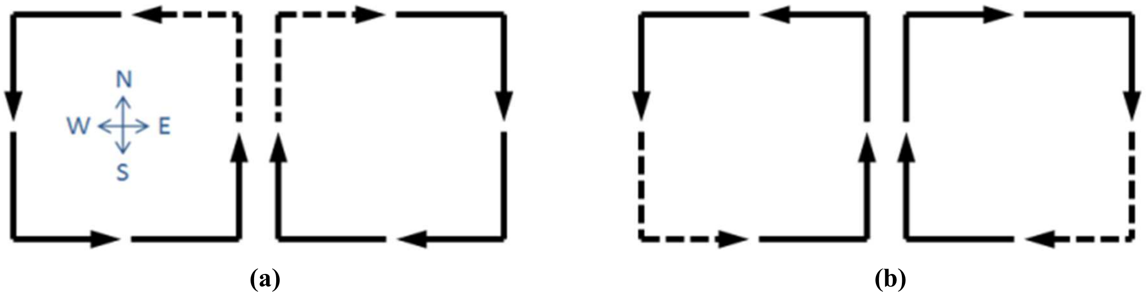


Figure 23 Turns allowed in the (a) North-last (b) South-last algorithms

3.3.1.2 NS-FTR Routing Scheme: Overview

To ensure robustness against faults, redundancy is a necessary requirement, especially for environments with high fault rates and unpredictable fault distributions. As the level of redundancy is increased, system reliability improves as a general rule. However, redundancy also detrimentally impacts other design objectives such as power and performance. Therefore in practice it is important to limit redundancy to achieve a reasonable trade-off between reliability, power, and performance. Unlike directed and random probabilistic flooding algorithms that propagate multiple copies of a packet to achieve fault-tolerant routing in NoCs, the proposed NS-FTR sends only one redundant packet for each transmitted packet, and only if the fault rate is above a replication threshold δ. The original packet is sent using the north last (NL) turn model while the redundant packet is propagated using south last (SL) turn model based routing scheme. The packet replication happens only at the source. Two separate virtual channels (VCs), one dedicated to the NL packets and the other for the SL packets ensure deadlock freedom. If the fault rate is low (i.e., below δ), replication is not utilized and only the original packet is sent using the NL scheme while power/clock gating the SL virtual channel to save power. The value of δ is a designer-specified parameter that depends on several factors such as the application characteristics, routing complexity, and power-reliability trade-off requirements.

The NS-FTR routing algorithm prioritizes minimal paths that have higher probabilities of reaching the destination even if faults are encountered downstream. Minimal paths and the replication threshold ensure low power dissipation under a diverse set of fault rates and distributions. No restriction on the number or location of faults is assumed, but the routers must know which of their adjacent (neighbor) links/nodes are faulty, which is accomplished using control signaling. The proposed routing approach can be combined with error control coding (ECC) techniques for transient fault resiliency and optimizations such as router buffer reordering [46] and router/NI backup paths [47] to create a comprehensive fault-tolerant NoC fabric. In the following subsections, we describe the implementation of the proposed NS-FTR scheme.

3.3.1.2.1 Turn Restriction Checks

Whenever a packet arrives at a router in the NS-FTR scheme, three scenarios are possible: (i) there is no fault in the adjacent links and the packet is routed to the output port based on the NL (or SL) scheme that selects a minimal path to the destination, (ii) there are one or more faults on adjacent links that prevent the packet from propagating in a valid direction (an output port with a fault- free link that does not violate turn model routing rules) in which case the packet must be dropped as a back turn is not allowed, and (iii) one or more adjacent links have faults but a valid direction exists to route the packet towards, based on the NL (or SL) rules. A packet traversing an intermediate router must choose among four output directions (N, S, W, E). However, choosing some directions may lead to a violation of a basic NL (or SL) turn rule immediately or downstream based on the relative location of the destination.

Figure 24 shows the SystemC [48] pseudocode of the turn restriction check phase for the NL turn model in our router model to aid in detecting an invalid routing direction. The input to the turn restriction check phase is a potential output port direction for a header flit in the router input buffer, and the output is an indication whether the output port direction is valid or not.

First, we check whether the output port direction has a fault in its attached adjacent link (steps 3-4) or is a back turn (steps 5-6), in which case this is an invalid direction. Next, we check for scenarios where choosing the given output port direction may lead to violations of the NL turn model rules immediately or eventually at some point downstream. In steps 7-8, we check for the scenario where the packet arrives in the same column as its destination which is located in the N direction. If the packet attempts to make a turn to E, W, or S, it will violate the basic NL turn model rules, and this is not allowed. Steps 9-12 test for two cases when the packet cannot be routed in the N direction without violating the NL turn model rules: if the current router node is already to the north of the destination code, or if the current node is not on the same column as its destination node. Finally, in steps 13-18, we check for a particular scenario in which the current node is on the southern border of the mesh, and with the packet is attempting to go in the opposite direction to where the destination is located (e.g., going W when the destination node is in the E direction; steps 14-15). As the packets can go north once they arrive at the column of their destination nodes, going in the opposite direction will eventually violate the NL turn model rules and is thus not allowed.

Pseudocode: Turn Restrictions for North-last Implementation

```
1:  RESULT   check_neighbor(UI ip_dir, UI direct, ULL cur_id, ULL dest_id)
    //ip_dir: direction where the packet comes from
    //direct: direction we want to check
    //cur_id: current node position
    //dest_id: destination node position.
    //dest_yco,dest_xco: y,x coordinates of destination node
    //cur_yco,cur_xco: y,x coordinates of current node
    //dif_yco=dest_yco-cur_yco
    //RESULT: LINK_FAULT, TURN_FAULT, BACK_TURN, OK
2:      {
3:   if (check_linkfault(direction)==LINK_FAULT)
4:      return LINK_FAULT;
5:   if (ip_dir==direct)
6:     return BACK_TURN;
7:   if(((cur_yco==dest_yco)&&(cur_xco>dest_xco))&&(direct!=N))
8:      return TURN_FAULT;
9:   if((cur_xco<dest_xco)&&(direct==N))
10:     return TURN_FAULT;
11:  if((cur_yco!=dest_yco)&&(direct==N))
12:     return TURN_FAULT;
13:  if(borderS(cur_id)) {
14:     if((dest_yco>cur_yco)&&direct==W)
15:             return TURN_FAULT;
16:     else if((dest_yco<cur_yco)&&direct==E)
17:             return TURN_FAULT;
18:  }
19:     return OK;
20: }
```

Figure 24 SystemC pseudocode for NL turn restriction checking

The turn restriction check phase described above is invoked in parallel for all four possible directions in an intermediate node for a header flit. Figure 25 shows the circuit level diagram for the NL turn restriction check phase, implemented in the route compute unit of a NoC router. The circuit is replicated at each input port of the router. At the TSMC 65nm technology node, the circuit dissipates 1.34 µW of power on average, and has a critical path latency of 0.48 ns. The pseudocode and circuit level diagrams for the SL turn model are similar and not shown here for brevity.

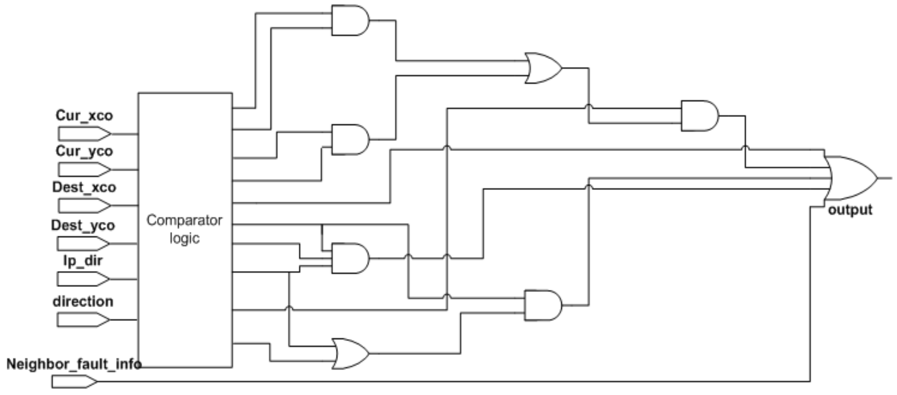
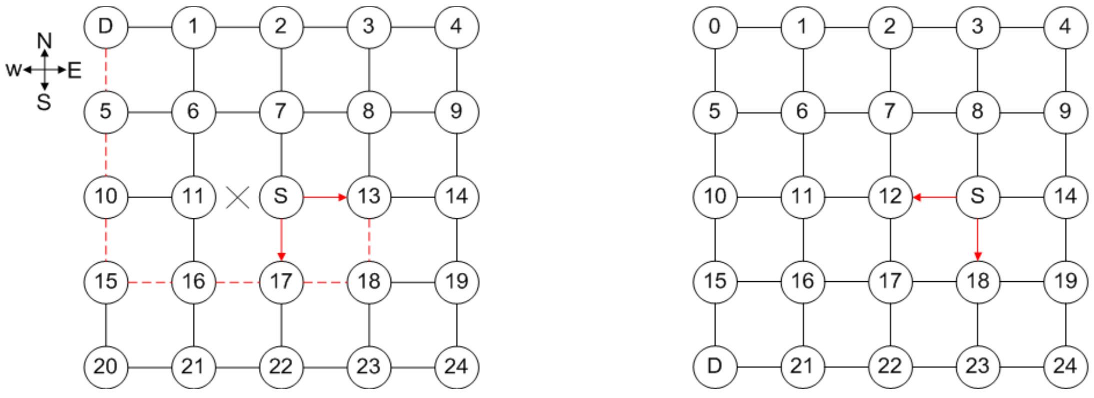


Figure 25 NL turn restriction check phase circuit diagram

In contrast, the circuit level implementation overhead for the turn restriction check phase in our previous work that combined the OE and IOE turn models was much greater, with 2.03 μW of power dissipation on average, and a critical path latency of 0.54 ns at the 65 nm node. Section IV presents results to compare the fault tolerance capabilities of these two schemes, as well as several other schemes proposed in literature.

3.3.1.2.2 Prioritized valid path selection

After the turn restriction checks, it is possible for a packet to have multiple valid directions to choose. In the NL scheme, we identify two scenarios where such a choice between valid paths exists and how NS-FTR selects among them. Again, we omit describing analogous scenarios for the SL scheme for brevity.



(a) (b)

Figure 26 Path selection scenarios for NL turn scheme

Figure 26(a) demonstrates an example of the first scenario. “S” and “D” are the source and destination nodes. With no faults, the NL minimal route would take the path S-11-10-5-D. Suppose that the link between S and node 11 has a fault. Then two choices exist: sending the packet to node 13 or 17. In this situation, we should choose node 17 as the resulting path is shorter by two hops than if node 13 was selected. We assume that a shorter path has lower probability of encountering a fault. This example can be generalized as follows: if a packet is to the SE or SW of its destination node, and the minimal path cannot be chosen due to a fault, then we give the S direction higher priority.

Figure 26(b) shows the second scenario. At source S, a packet intended for destination D can be sent along two minimal paths. We observe that if the packet is sent along the path with node 18, the only paths to D that are available are S-18-17-16-15-D, S-18-17- 16-21-D, S-18-17-22-21-D, S-18-23-22-21-D, and S-18-19-24-23- 22-21-D. In contrast, if the packet is sent along the path with node 12, many more paths exist. This greater path diversity is critical in the presence of faults, as it can ensure a better probability of packet arrival at D. Again we can generalize the example as follows: if a packet is at the NE (or NW) of the destination and if multiple valid minimal paths exist, then we prioritize the E (or W) directions.

3.3.3 Router architecture and control network

Figure 27 depicts our dual virtual channel (VC) router architecture. Packets traversing the network have a header flit with a destination ID field that is used to determine the output port at each router. The router datapath consists of the buffers and the switch. The input FIFO buffers store up to 16 flits waiting to be forwarded to the next hop. There are two input FIFO buffers each dedicated to a VC, with one VC for the NL routed packets and the other for the SL routed packets. When a flit is ready to move, the switch connects an input buffer to an appropriate output channel. To control the datapath, the router contains three major control modules: a route compute control unit (RC_CTRL), a virtual-channel (VCA) allocator, and a switch allocator (SA). These control modules determine the next hop direction, the next virtual channel, and when a switch is available for each packet/flit. The routing operation takes four phases: route computation (RC), virtual-channel allocation (VCA), switch allocation (SA), and switch traversal (ST). When a header flit (the first flit of a packet) arrives at an input channel, the router stores the flit in the buffer for the allocated VC and determines the next hop for the packet using the checks and priority rules described in the previous sections (RC phase). Given the next hop, the router then allocates a virtual channel (VCA phase). Finally, the flit competes for a switch (SA phase) if the next hop can accept the flit, and moves to the output port (ST phase). In our scheme, a packet carries its virtual channel (NL or SL) information in a bit in the header flit, therefore the output port and the destination VC can be determined early, at the end of the RC stage.

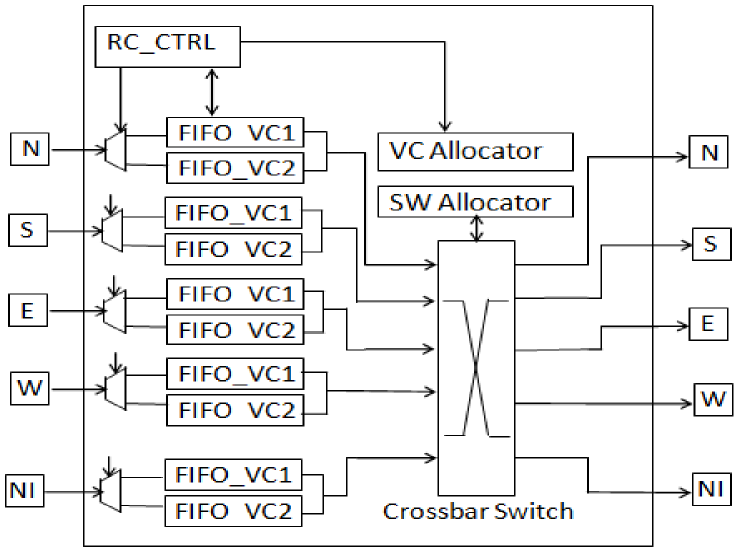


Figure 27 Router architecture

To reduce energy consumption, if the fault rate is below the replication threshold δ, the RC_CTRL unit shuts down the SL virtual channels in the router. We assume that dedicated control signals connected to a lightweight fault detection unit (FDU) can allow estimation of the fault rate at runtime in the NoC, and the decision to change the replication status (initiate or cut-off) can be communicated to the nodes and implemented with a delay of at most a few hundred cycles. Typically, such a change in status would be a rare occurrence as intermittent or permanent faults do not appear at runtime as frequently as transient faults.

An important requirement for any fault-tolerant NoC fabric is a control network that can be used to send acknowledgement signals to inform the source whether the packet successfully arrives at the destination. We assume a lightweight, fault-free control network that is topologically similar to the data network for the purpose of routing ACK signals from the destination to the source node on successful packet arrival, or NACK signals from an intermediate node to the source node if the packet must be dropped due to faults that make it impossible to make progress towards the destination. In our implementation, a source can re-send a packet at most twice after receiving NACK signals, before assuming that the packet can never reach its destination. While this can signal unavoidable failure for certain types of applications, other applications may still be able to operate and maintain graceful degradation. For instance, a multiple use-case CMP can shut down certain application use-cases when their associated inter-core communication paths encounter fatal faults, but there might be other use cases that can still continue to operate on fault-free paths. We assume that a higher level protocol (e.g. from the middleware or OS levels) can step in to initiate thread migration and other strategies to ensure useful operation in the presence of unavoidable system faults.

3.3.4 Experimental studies

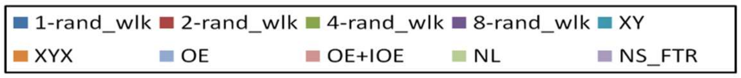
3.3.4.1 Evaluation setup

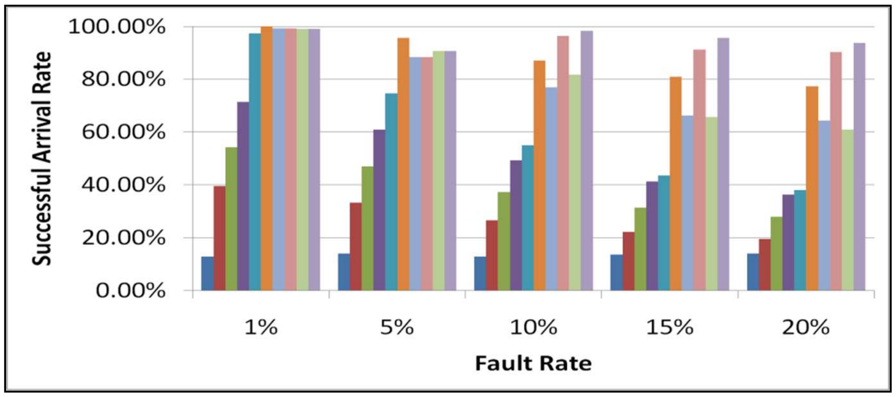
We implemented and evaluated our proposed algorithm using a SystemC based cycle-accurate 2D mesh NoC simulator that was created by extending the open source Nirgam simulator [49] to support faults and different fault-tolerant routing schemes. In addition to communication performance estimates, we were interested in NoC energy estimation as well. For this purpose we incorporated dynamic and leakage power models for standard NoC components from Orion 2.0 [50] into the simulator. We also incorporated the power overhead of the circuits in the routers (obtained after synthesis) required to realize the fault-tolerant routing algorithms. The NoC fabric was clocked at 1 GHz. The target implementation technology was 65nm, and this node was used to determine parameters for delay and energy consumption. A 9×9 (81 core) CMP with a mesh NoC fabric was considered as the base system in our experiments. The stimulus for the simulations was generated for specific injection rates using uniform random traffic models. A fixed number of flits (3000/core) were transmitted according to the specified traffic pattern in the simulations, to enable comparisons for not only successful arrival rates, but also for overall communication energy and average packet latency.

We modeled two types of faults in the NoC: *(i)* design time permanent faults that are randomly distributed link-level hard failures, and *(ii)* runtime intermittent faults, which are also randomly distributed and appear due to reasons such as periods of thermal hotspots and high crosstalk noise during system operation. For intermittent faults, we assumed a fault duration of 5000 cycles, with the fault location being randomly generated, as for permanent faults. Ten different random fault distributions were generated and analyzed for each fault rate to obtain a more accurate estimate of fault resiliency for the routing schemes. To ensure fair comparison, the randomized fault locations and durations were replicated across simulation runs for different fault-tolerant routing schemes, to study the performance of the schemes under the same fault conditions.

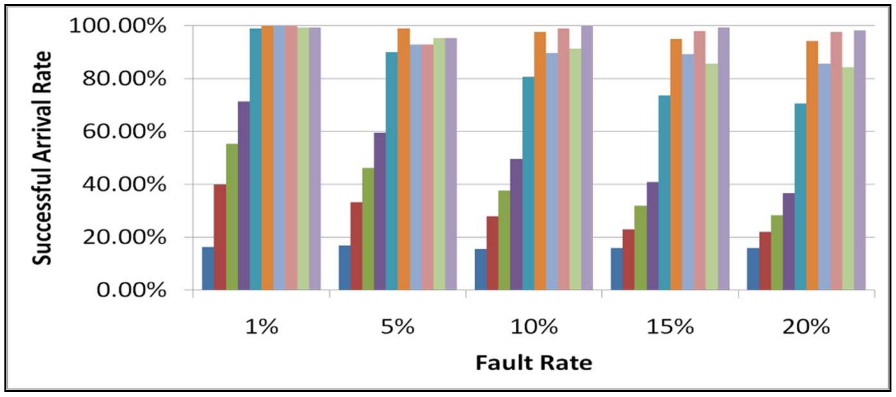
3.3.4.2 Comparison with Existing FT Routing Schemes

Our first set of experiments compared the packet arrival rates for our NS-FTR fault-tolerant routing scheme with several fault-tolerant routing schemes from literature. The following schemes were considered in the comparison study: *(i)* XY dimension order routing [1] (although not inherently fault-tolerant, it is considered here because of its widespread use in 2D mesh NoCs), *(ii)* N-random walk [29] for N = 1,2,4,8 *(iii)* adaptive OE turn model [36], *(iv)* adaptive North-last turn model [33], *(v)* adaptive OE+IOE which combines replication with the OE and IOE turn models on two VCs, and *(vi)* XYX [37] which combines replication with the XY and YX routing schemes on two VCs. Other fault-tolerant routing schemes such as probabilistic flooding [28] and fully adaptive table based routing [31] were considered but their results are not presented because these schemes have major practical implementation concerns - their energy consumptions is an order of magnitude higher than other schemes and frequent deadlocks hamper overall performance. For our NS-FTR scheme, we used a prioritized valid path selection introduced in Section III.B.2 and a replication threshold value of δ = 6%, based on our extensive simulation studies that showed a good trade-off between arrival rate and energy for 8% ≥ δ ≥ 2%.

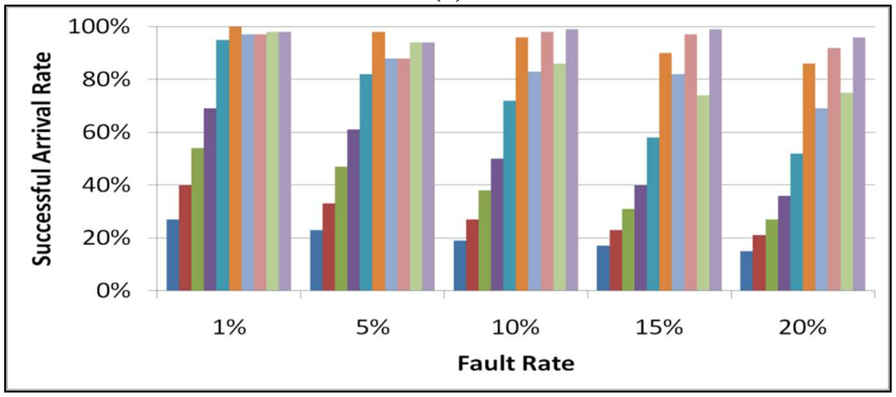




(a)



(b)



(c)

Figure 28 Packet arrival rate for 9×9 NoC with (a) permanent faults, (b) intermittent faults, (c) both permanent and intermittent faults

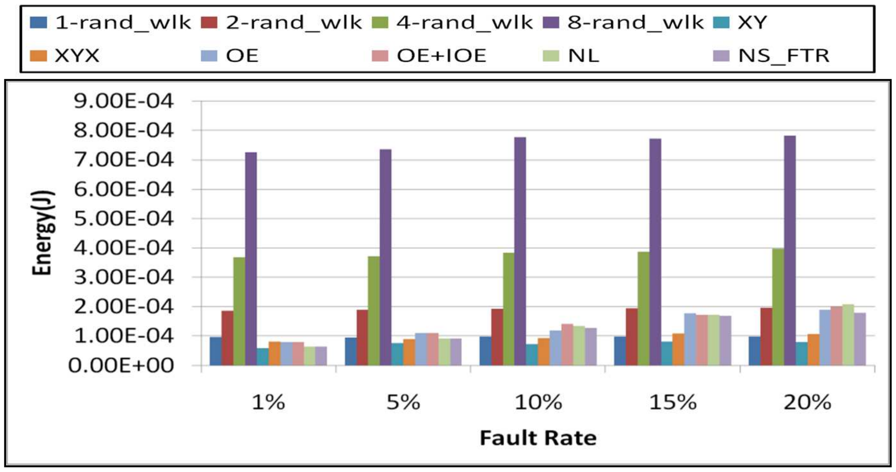
Figure 28(a)-(c) compares the successful packet arrival rate for the fault-tolerant routing schemes described above under different fault rates and types. Results are shown for a 20% injection rate (0.2 flits/node/cycle). The arrival rates are shown for permanent faults (Figure 28(a)), intermittent faults (Figure 28(b)), and an equal distribution of permanent and intermittent faults

(Figure 28(c)). Not surprisingly, arrival rates for all routing schemes in general are higher under intermittent faults than under permanent faults, as intermittent faults only last for a short period of time. On analyzing individual routing scheme performance, it can be immediately seen that the N-random walk schemes have a low arrival rate even in the absence of faults. This is due to frequent deadlocks during simulation which the scheme is susceptible to, causing packets to be dropped. The best arrival rate for the N-random walk scheme is achieved for a value of N=8. Out of the turn/dimension-order based routing schemes, XY not surprisingly performs worse than the other schemes due to the lack of any built in fault tolerance capability. The arrival rates for the NL and OE routing schemes are comparable. Among the dual VC schemes, OE+IOE outperforms XYX due to greater path diversity with the less restrictive OE and IOE turn model schemes than with XY or YX schemes. However, our NS-FTR scheme has a higher successful packet arrival rate than OE+IOE and every other scheme we compared. For fault rates under 6%, replication in NS-FTR is disabled to save energy (discussed later), which explains the slight dip in successful arrival rate for low fault rates. However, with replication enabled, NS-FTR outperforms every other scheme at low fault rates as well. NS-FTR scales particularly well under higher fault rates that are expected to be the norm in future CMP designs.

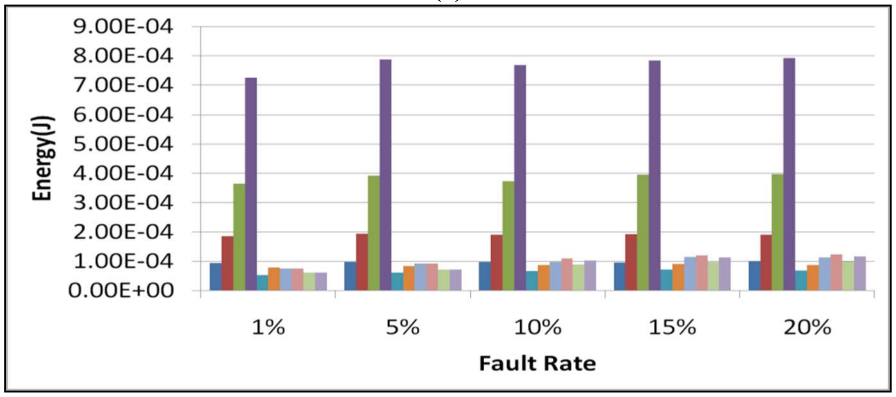
In addition to arrival rate, an increasingly important metric of concern for designers is the energy consumption in the on-chip communication network.

Figure **29** shows the energy consumption for the different routing schemes under the same fault rates and types, and injection rates as in the previous experiment. The energy consumption accounts for circuit level implementation overheads as well as acknowledgements and retransmission in the schemes. Energy consumption under intermittent faults is lower than under permanent faults as intermittent faults last for only a short amount of time, incurring lower retransmission and non-minimal path overheads in the schemes. It can be seen that the N-random walk fault-tolerant routing schemes have significantly higher energy consumption compared to the other schemes. This is due to the high level of replication with increasing values of N, as well as because of the non-minimal paths chosen to route the packets on. For single VC schemes (OE, NL, XY), the XY scheme has the lowest energy dissipation which explains its popularity in NoCs today. But as it is lacking in any fault tolerance, the scheme may not be a viable option for future CMPs.

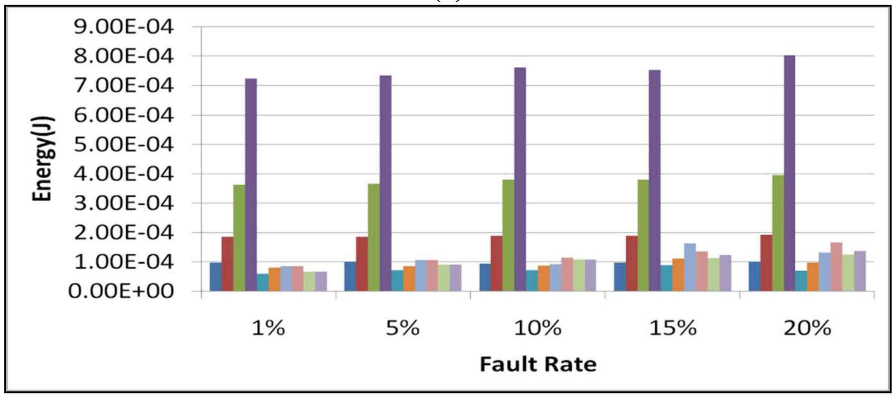
The energy consumption for the NL and OE schemes is comparable and higher than the XY scheme because they use non- minimal paths at times. But both NL and OE schemes offer higher successful packet arrival rate due to greater path diversity compared to the XY scheme, with the gap increasing especially under higher fault rates. For high fault rates, the XYX scheme has lower energy consumption than NS-FTR, OE+IOE, and even the single VC schemes (NL and



(a)



(b)



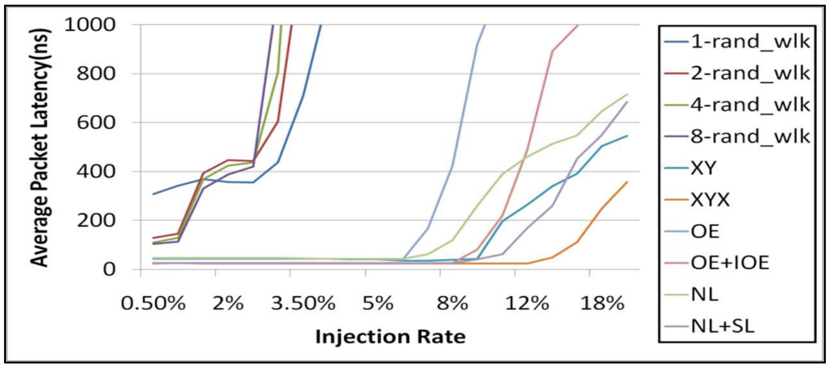
(c)

Figure 29 Communication energy for 9×9 NoC with (a) permanent faults (b) intermittent faults, (c) both permanent and intermittent faults

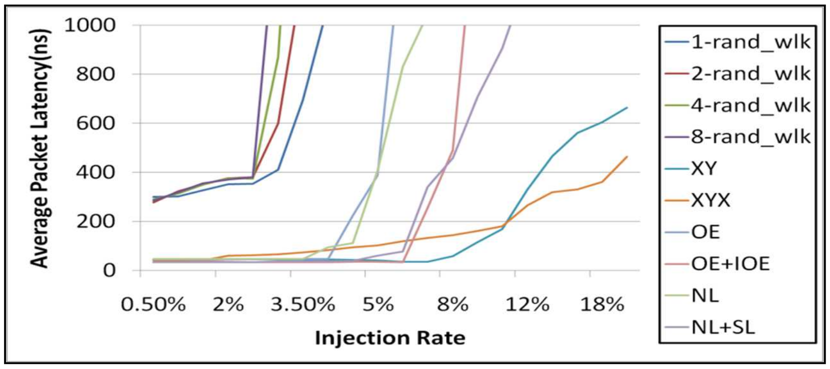
OE) because it only uses minimal paths, much like the XY scheme. However, the XYX arrival rates are lower than that for NS-FTR and OE+IOE. Under low fault rate conditions (< 6%), both NS-FTR and OE+IOE disable replication, and therefore have lower energy consumption than XYX. In some cases, such a replication threshold based mechanism can be extremely useful to trade-off energy with packet arrival rate. For instance, for multimedia applications, occasionally dropped pixel data may not be perceivable by viewers, and in such scenarios the resulting lower energy consumption may lead to a much longer battery life and better overall user experience. Conversely, if fault resiliency is paramount, the value of δ can be reduced to close to 0, to enable the highest possible successful packet arrival rate.

Figure 30(a)-(b) summarizes the average packet latency for the routing schemes for various injection rates, with a low fault rate (1% permanent faults and 1% intermittent faults; Figure 30(a)) and a high fault rate (10% permanent faults and 10% intermittent faults; Figure 30(b)). The N-random walk scheme has significantly higher packet latencies than other schemes due to its stochastic non- minimal path selection process. For low fault rates (Figure 25(a)), low injection rates show comparable packet latencies for the rest of the schemes. However, as injection rates increase, the minimal path schemes (XY, XYX) have lower average latencies. But the latency figure becomes irrelevant for the XY and XYX schemes as their packet arrival rates suffer under high fault rates (Figure 28). Among schemes that support adaptivity in path selection, the NS-FTR has the lowest packet latencies even for high injection rates.

For high fault rates (Figure 30(b)), at low injection rates, both XY and XYX schemes surprisingly have higher packet latencies than turn model schemes such as NS-FTR, OE+IOE, OE, and NL. This is because while turn model schemes such as NS_FTR or OE may not use minimal paths (as XY and XYX schemes do), they have greater adaptivity to avoid faults and thus avoid retransmission, which can increase packet latencies directly and also indirectly (e.g., by increasing congestion). At higher injection rates, XY and XYX again have lower packet latencies, but suffer from lower packet arrival rates than the NS-FTR and OE+IOE schemes. Out of the two, NS-FTR again shows lower packet latencies than OE+IOE, even as injection rates increase.



(a)



(b)

Figure 30 Average packet latency for 9×9 NoC with both permanent and intermittent faults (a) 1% fault rate, (b) 20% fault rate

In summary, from the results it is quite apparent that the proposed NS-FTR scheme possesses high fault tolerance that scales well with rising (permanent and intermittent) fault rates and injection rates. Compared to the XYX scheme, NS-FTR has much higher successful packet arrival rates (>10%) especially when fault rates are high. Compared to the OE+IOE scheme, NS-FTR in general has lower implementation overhead, higher packet arrival rates, lower energy consumption, and lower packet latencies. These results strongly motivate the use of NS-FTR in future CMP designs to ensure fault-tolerant on-chip communication.

3.3.5 Summary

In this section, we proposed a novel fault-tolerant routing scheme (NS-FTR) for NoCs that combines the North-last and South-last turn models to create a robust hybrid NoC routing scheme. The proposed scheme is shown to have a low implementation overhead and adapt to design time and runtime faults better than existing turn model, stochastic random walk, and dual virtual channel based routing schemes such as XYX and OE+IOE.

3.4 Fault-tolerant routing for 3D NoCs

Three-dimensional integrated circuits (3D-ICs) offer a significant opportunity to enhance the performance of emerging chip multiprocessors (CMPs) using high density stacked device integration and shorter through silicon via (TSV) interconnects that can alleviate some of the problems associated with interconnect scaling in sub-65nm CMOS technologies. However, network-on-chip (NoC) fabrics that will connect the cores together in 3D-ICs will increasingly be susceptible to permanent and intermittent faults, which can cause catastrophic system failure. In this section, we explore fault resilient 3D NoC routing and propose a new fault-tolerant routing scheme (4NP-First) for 3D mesh NoCs that combines the negative-first and positive-first turn models together with opportunistic replication to increase successful packet arrival rate while optimizing energy consumption. Our extensive experimental results indicate that the proposed 4NP-First fault-tolerant routing scheme has a low implementation overhead and provides better fault resiliency than existing dimension-order, turn-model, and stochastic random walk based fault-tolerant 2D NoC routing schemes that were extended to 3D NoCs.

3.4.1 Proposed 4NP-First routing scheme

3.4.1.1 3D turn models

We extend the 2D turn models proposed by Glass [33] into the third dimension in this section. The 2D west-first, negative-first, and north-last turn models are transformed into 2 negative-first (2N-First), 3 negative-first (3N-first), and 4 negative first (4N-First) turn models in 3D, respectively. For the sake of discussion and because we use it in our approach, we focus on the 4N-First model.

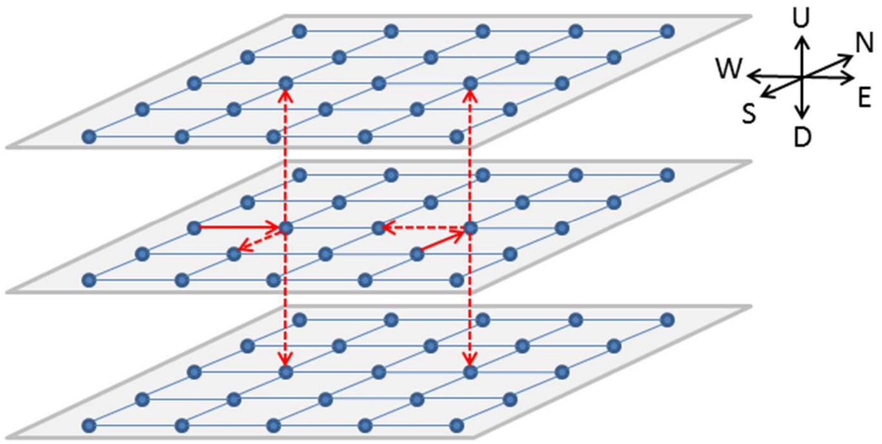


Figure 31 Forbidden turns in the 4N-First turn model

In the 4N-First turn model, two out of the possible three positive directions (N, E, U) need to be selected as the last two directions before routing a packet to its destination. Thus at the beginning of the routing phase, we can adaptively choose to route the packet along the remaining one positive, and three negative (S, W, D) directions. For our implementation, we chose the two positive last directions as N and E. Figure 31 shows the forbidden turns for the 4N-First turn model. A packet coming from the W cannot be routed in the S, U, or D directions, while a packet coming from the south direction cannot be routed in the W, U, or D directions. To summarize, the forbidden turns in the 4N-First turn model are: E-U, E-D, E-S, N-U, N-D, and N-W. These turn restrictions allow deadlock free routing with some measure of adaptivity in the 4N-First turn model. The analogous 4P-First turn model can be understood by inverting the 3D mesh by 180 degrees. In the 4P-First turn model, two out of the possible three negative directions must be selected as the last two directions before routing a packet to its destination. In our routing algorithm, we choose the two negative last directions as S and W.

3.4.1.2 4NP-First Overview

To ensure robustness against faults, redundancy is a necessary requirement, especially for environments with high fault rates and unpredictable fault distributions. As the level of redundancy is increased, system reliability improves as a general rule. However, redundancy also detrimentally impacts other design objectives such as power and performance. Therefore in practice it is important to limit redundancy to achieve a reasonable trade-off between reliability, power, and performance. Unlike directed and random probabilistic flooding algorithms that propagate multiple copies of a packet to achieve fault-tolerant routing in NoCs, the proposed 4NP-First routing scheme sends only one redundant packet for each transmitted packet, and only if the fault rate is above a replication threshold δ. The original packet is sent using the north last (4N-First) turn model while the redundant packet is propagated using south last (4P-First) turn model based routing scheme. The packet replication happens only at the source. Two separate virtual channels (VCs), one dedicated to the 4N-First packets and the other for the 4P-First packets ensure deadlock freedom. If the fault rate is low (i.e., below δ), replication is not utilized and only the original packet is sent using the 4N-First scheme while power and clock gating the 4P-First virtual channel to save power. The value of δ is a designer-specified parameter that depends on several factors such as the application characteristics, routing complexity, and power-reliability trade-off requirements.

The 4NP-First routing algorithm prioritizes minimal paths that have higher probabilities of reaching the destination even if faults are encountered downstream. Minimal paths and the replication threshold ensure low power dissipation under a diverse set of fault rates and distributions. No restriction on the number or location of faults is assumed, but the routers must know which of their adjacent (neighbor) links/nodes are faulty, which is accomplished using basic control signaling that already exists in any NoC architecture. The proposed routing approach can be combined with error control coding (ECC) techniques for transient fault resiliency and optimizations such as router buffer reordering and router/NI backup paths to create a comprehensive fault-tolerant NoC fabric. In the following subsections, we describe the implementation of the proposed 4NP-First scheme. The implementation description focuses on the 4N-First turn model. The implementation for the analogous 4P-First turn model is similar, and therefore is excluded for brevity.

3.4.1.3 Turn Restriction Checks

Whenever a packet arrives at a router in the 4NP-First scheme, three scenarios are possible: (i) there is no fault in the adjacent links and the packet is routed to the output port based on the 4N-First (or 4P-First) scheme that selects a minimal path to the destination, (ii) there are one or more faults on adjacent links that prevent the packet from propagating in a valid direction (an output port with a fault- free link that does not violate turn model routing rules) in which case the packet must be dropped as a back turn is not allowed, and (iii) one or more adjacent links have faults but a valid direction exists to route the packet towards, based on the 4N-First (or 4P-First) rules. A packet traversing an intermediate router must choose among six output directions (N, S, W, E, U, D). However, choosing some directions may lead to a violation of a basic 4N-First (or 4P-First) turn rule immediately or downstream based on the relative location of the destination.

Pseudocode: Turn Restrictions for 4N-First Implementation

```
1: RESULT   check_neighbor(UI ip_dir, UI direct, ULL cur_id, ULL dest_id) {
     //ip_dir: direction where the packet comes from
     //direct: direction we want to check
     //cur_id: current node position
     //dest_id: destination node position.
     //dest_yco,dest_xco: y,x coordinates of destination node
     //cur_yco,cur_xco: y,x coordinates of current node
     //dif_yco=dest_yco-cur_yco
     //RESULT: LINK_FAULT, TURN_FAULT, BACK_TURN, OK
2:   if (check_linkfault(direction)==LINK_FAULT)
3:       return LINK_FAULT;
4:   if (ip_dir==direct)
5:           return BACK_TURN;
6:   if(((ip_dir==S)||(ip_dir==W))&&((direct!=N)&&(direct!=E)))
7:       return TURN_FAULT;
8:   if(((dest_xco>cur_xco)&&((direct==N)||(direct==E)))
       ||((dest_yco<cur_yco)&&((direct==N)||(direct==E)))
       ||((dest_zco!=cur_zco)&&((direct==N)||(direct==E))))
9:       return TURN_FAULT;
```

```
10: if((dest_xco==cur_xco)&&(direct==N)){
11:     return TURN_FAULT;
12: if((dest_yco==cur_yco)&&(direct==E)){
13:     return TURN_FAULT;
14: if((borderD(cur_zco)&&borderS(cur_address.tile_id))
        &&(dest_zco==cur_zco)&&(dest_yco>cur_yco)
        &&(dest_xco==cur_xco)&&(direct==W))
15:     return TURN_FAULT;
16: if((borderD(cur_zco)&&borderW(cur_address.tile_id))
        &&(dest_zco==cur_zco)&&(dest_yco==cur_yco)
        &&(dest_xco<cur_xco)&&(direct==S))
17:         return TURN_FAULT;
18: return OK;
19: }
```

Figure 32 Pseudocode for 4N-First turn restriction checking

Figure 32 shows the pseudocode of the turn restriction check phase for the 4N-First turn model to aid in detecting an invalid routing direction. The input to the turn restriction check phase is a potential output port direction for a header flit currently in the router input buffer, and the output of the phase indicates whether the output port direction is valid or not.

First, we check whether the output port direction has a fault in its attached adjacent link (steps 2-3) or is a back turn (steps 4-5), in which case this is an invalid direction. Next, we check for scenarios where choosing the given output port direction may lead to violations of the 4N-First turn model rules immediately or eventually at some point downstream. In steps 6-7 we check for the basic 4N-First turn rules (Section 3.1), and prohibit a packet coming from S or W to go in any direction other than N or E. Steps 8-9 check for the following condition: if the destination is on the S, W, or another layer relative to the current node, then the packet cannot turn N or E, because once it does, it can only be routed N or E and will never reach the destination. Steps 10-13 check for two conditions. If a packet is in the same row as the destination, it should not be routed to the N direction as the packet will eventually need to be routed to the S direction that will break a basic turn model rule. Similarly, if a packet is on the same column as its destination, it should not be routed to the E direction as it eventually will have to turn W and this will break the basic turn model rule. Steps 14-17 check for two special situations. First, if a packet is on the bottom layer and on the S border, and the destination is on the same layer and to the E direction of the current node, then the packet should not be routed to the W direction. This is because the packet will eventually need to go N or U, then E, and then finally S or D, which will break the basic turn model rules. Second, if a packet is on the bottom layer and on the W border, and the destination is on the same layer and to the S direction of the current node, then the packet should not be routed in the N direction, for reasons similar to the first scenario.

3.4.1.4 Prioritized Valid Path Selection

After the turn restriction checks, it is possible for a packet to have multiple available (valid) directions to choose from along minimal paths, or if a minimal path is not available, then we may need to select among directions that lead to non-minimal paths. In such a scenario, this section answers the question: which out of the multiple available paths should be selected for the 4N-First routing algorithm to ensure maximum arrival rate?

A packet in the process of being routed using the 4N-First scheme may encounter two situations:

Case 1: The packet has already been routed along the N or E directions, and thus it can only choose the N or E directions to its destination;

Case 2: The packet has not been routed along the N or E directions and therefore can have at most four choices (S, W, U, or D directions)

For each of these cases, we perform prioritized valid path selection in accordance with two general rules: (i) directions along the minimal path are given the highest priority, and (ii) if a minimal path direction cannot be selected, we prefer a direction that has the largest next-hop node diversity number compared to other directions with the same distance to the destination. The diversity number of a node enumerates the number of available paths to the destination from that node. In the following sections, we describe specific rules for prioritized valid path selection for the two cases identified above.

3.4.1.4.1 Prioritized valid path selection for Case 1

Figure **33** shows the case with the destination (X) to the north and east of the current node (C), with each node annotated with its diversity number, determined recursively. For all minimal paths from C to X, it can be seen that the paths passing along the middle of the bounding sub-mesh have the highest node diversity at each hop, and therefore the highest probability of reaching the destination in the presence of faults along the routed path from C to X. In other words, it would be preferable to choose a path where we toggle between the N and E directions as much as possible (as in the path highlighted in red in

Figure **33**) than follow a path along the periphery (as for instance in XY routing). Thus, a path along the middle of the mesh is preferable than a path closer to the boundary.

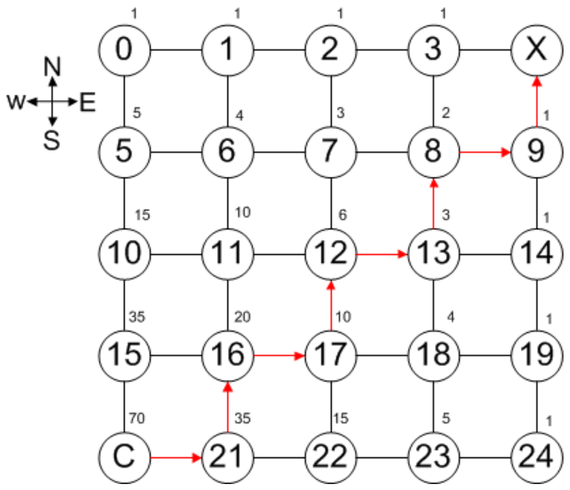


Figure 33 Node diversity and prioritized path for Case 1

3.4.1.4.2 Prioritized valid path selection for Case 2

For case 2, consider

Figure **34** with the current node C to the E and U direction relative to destination X. Each node in the mesh is annotated with its diversity number in terms of n, which is the number of layers (four in this case). It can be observed that sending the packet to the row coordinate of the destination before sending it in the D direction (depicted in the solid red line) leads to fewer path choices for the packet in case the path has a fault. In contrast, sending the packet down to the layer of the destination first before routing in the D direction (depicted in the dotted red line) is a path with greater diversity, and thus greater resiliency if faults are encountered. It should also be observed that nodes towards the middle of the mesh have greater diversity numbers – therefore paths along the middle of the mesh have a greater opportunity of arriving at the destination than paths along the periphery, which is a similar observation as what we described for Case 1 in the previous section. Thus, given a choice among S, W, U, and D directions, to ensure highest arrival rate, the packet should first be routed U or D to the same layer as the destination, and then routed in a direction along the middle of the mesh as much as possible.

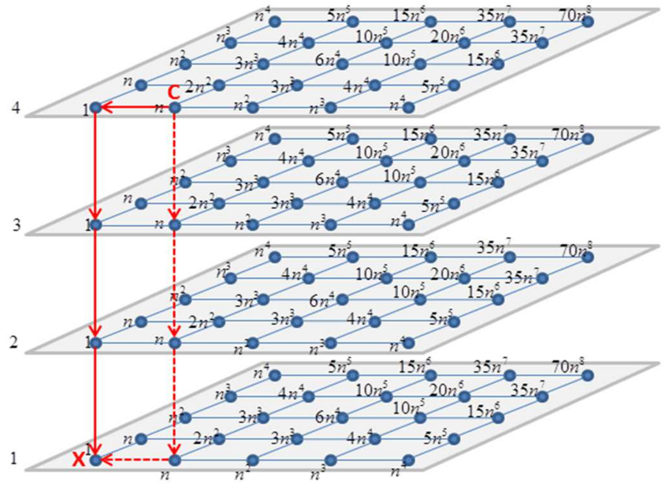


Figure 34 Node diversity and prioritized path for Case 2

3.4.2 4NP-first router implementation

Figure 35 depicts our dual virtual channel (VC) router architecture. Packets traversing the network have a header flit with a destination ID field that is used to determine the output port at each router. The router datapath consists of the buffers and the switch. The input FIFO buffers store up to 16 flits waiting to be forwarded to the next hop. There are two input FIFO buffers each dedicated to a VC, with one VC for the NL routed packets and the other for the SL routed packets. When a flit is ready to move, the switch connects an input buffer to an appropriate output channel. To control the datapath, the router contains three major control modules: a route compute control unit (RC_CTRL), a virtual-channel (VCA) allocator, and a switch allocator (SA). These control modules determine the next hop direction, the next virtual channel, and when a switch is available for each packet/flit. The routing operation takes four phases: route computation (RC), virtual-channel allocation (VCA), switch allocation (SA), and switch traversal (ST). When a header flit (the first flit of a packet) arrives at an input channel, the router stores the flit in the buffer for the allocated VC and determines the next hop for the packet using the checks and priority rules described in the previous sections (RC phase). Given the next hop, the router then allocates a virtual channel (VCA phase). Finally, the flit competes for a switch (SA phase) if the next hop can accept the flit, and moves to the output port (ST phase). In our scheme, a packet carries its virtual channel (4N-First or 4P-First) information in a bit in the header flit, therefore the output port and the destination VC can be determined early, at the end of the RC stage.

To reduce energy consumption, if the fault rate is below the replication threshold δ, the RC_CTRL unit shuts down the SL virtual channels in the router. We assume that dedicated control signals connected to a lightweight fault detection unit (FDU) can allow estimation of the fault rate at runtime in the NoC, and the decision to change the replication status (initiate or cut-off) can be communicated to the nodes and implemented with a delay of at most a few hundred cycles. Typically, such a change in status would be a rare occurrence as intermittent or permanent faults do not appear at runtime as frequently as transient faults.

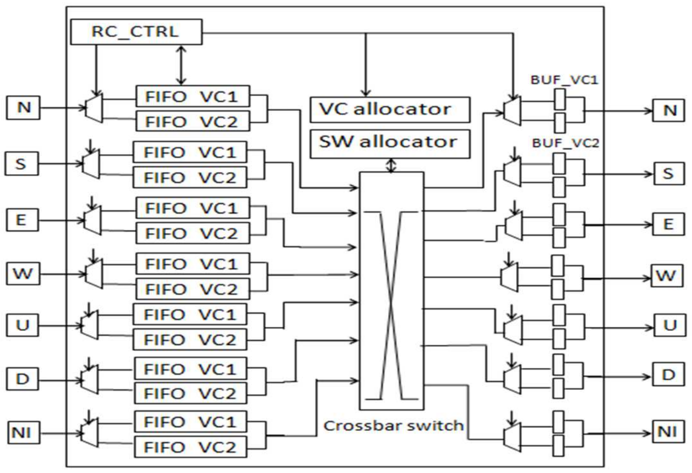


Figure 35 Router architecture

An important requirement for any fault-tolerant NoC fabric is a control network that can be used to send acknowledgement signals to inform the source whether the packet successfully arrived at the destination. We assume a lightweight and fault-free control network that is topologically similar to the data network for the purpose of routing ACK signals from the destination to the source node on successful packet arrival, or NACK signals from an intermediate node to the source node if the packet must be dropped due to faults that make it impossible to make progress towards the destination. In our implementation, a source can re-send a packet at most five times after receiving NACK signals, before assuming that the packet can never reach its destination. While this can signal unavoidable failure for certain types of applications, other applications may still be able to operate and maintain graceful degradation. For instance, a multiple use-case CMP can shut down certain application use-cases when their associated inter-core communication paths encounter fatal faults, but there might be other use cases that can still continue to operate on fault-free paths. We assume that a higher level protocol (e.g. from the middleware or OS levels) can step in to initiate thread migration and other strategies to ensure useful operation in the presence of unavoidable system faults.

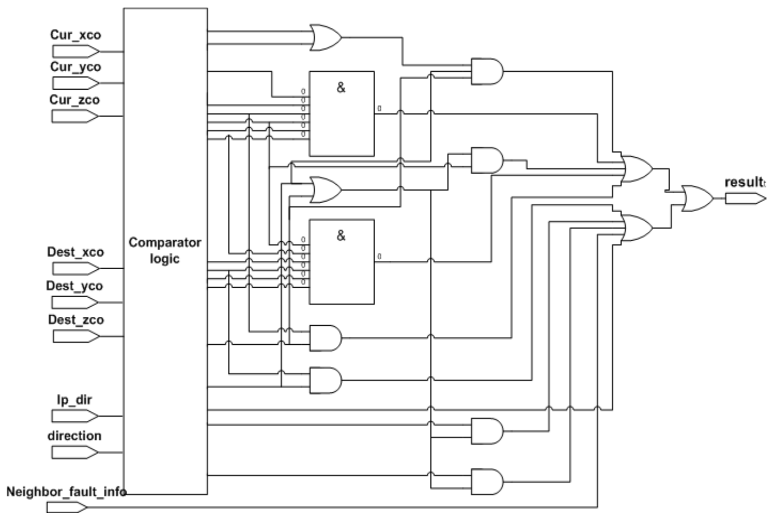


Figure 36 4N-First routing logic circuit diagram

Figure 36 shows the circuit level diagram for the 4N-First turn restriction check phase and prioritized path selection, implemented in the route compute unit of a NoC router. The circuit is replicated at each input port of the router. At the TSMC 65nm technology node, the circuit dissipates 0.98 µW of power on average, and has a critical path latency of 0.77 ns.

3.4.3 Experiments

3.4.3.1 Experimental Setup

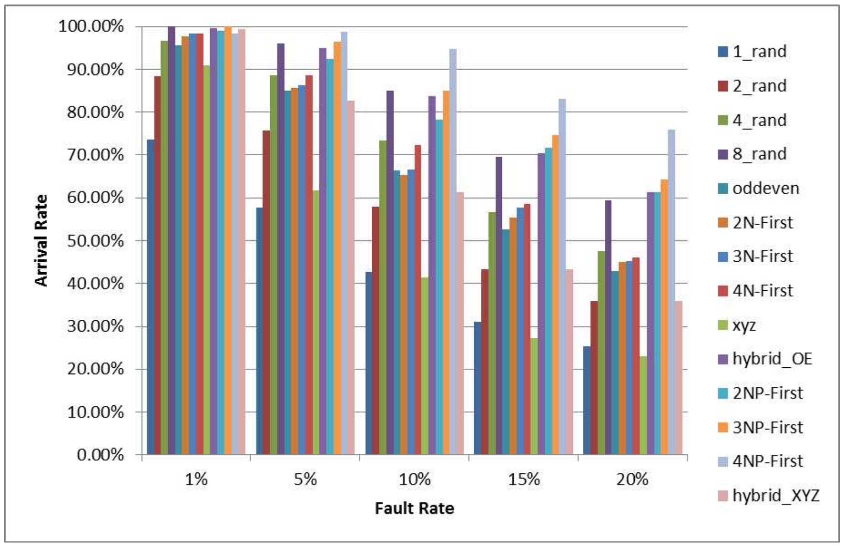
We implemented and evaluated our proposed algorithm using a SystemC [48] based cycle-accurate 3D mesh NoC simulator that was created by extending the open source 2D NoC Nirgam simulator [49] by adding support for a 3D network, faults, and various fault-tolerant routing schemes. In addition to communication performance estimates, we were interested in NoC energy estimation as well. For this purpose we incorporated dynamic and leakage power models for standard NoC components from Orion 2.0 [50] into the simulator. We also incorporated the power overhead of the circuits in the routers (obtained after synthesis) required to realize the fault-tolerant routing algorithms. The NoC fabric was clocked at 1 GHz. The target implementation technology was 65nm, and this node was used to determine parameters for delay and energy consumption. A 5×5×4 (100 cores; 25 cores/layer in 4 layers) CMP with a 3D mesh NoC fabric was considered as the base system in our experiments. The stimulus for the simulations was generated for specific injection rates using uniform random, transpose, and hotspot traffic models. A fixed number of flits (3000/core) were transmitted according to the specified traffic pattern in the simulations, to enable comparisons for not only successful arrival rates, but also for overall communication energy and average packet latency. We created a statistical fault model with random distributions for design time and runtime permanent faults across the network fabric. Ten different distributions were considered for each evaluation point to obtain a more accurate estimate of fault resiliency for the routing schemes. To ensure fair comparison, the randomized fault locations were replicated across simulation runs for different fault-tolerant routing schemes, to study the performance of the schemes under the same fault conditions.

3.4.3.2 Comparison with existing FT routing schemes

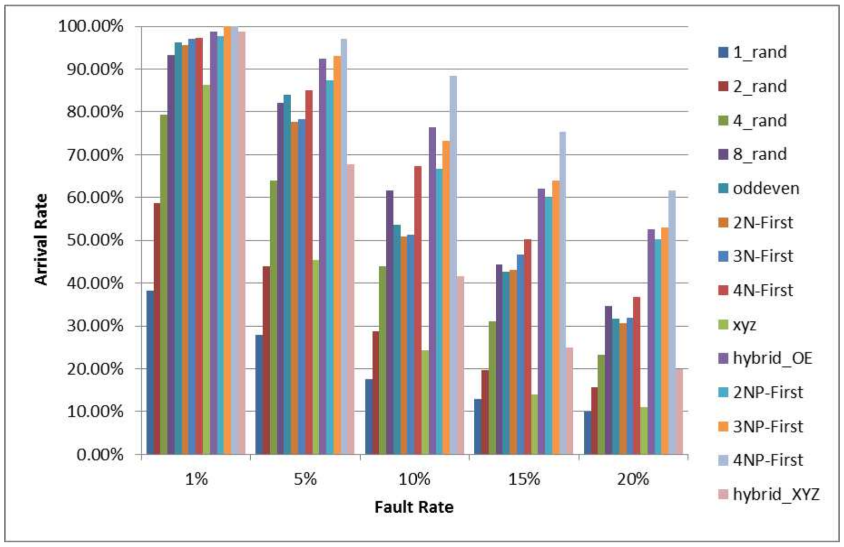
Our first set of experiments compares the successful packet arrival rates for our 4NP-First fault-tolerant routing scheme with fault-tolerant routing schemes for 2D NoCs proposed in literature that were extended to 3D NoCs, as well as variants of turn model based routing schemes that we extended to 3D NoCs. The following schemes were considered in the comparison study: *(i)* XYZ dimension order routing (although not inherently fault-tolerant, it is considered here because of its widespread use in 3D mesh NoCs proposed in literature), *(ii)* N-random walk [29] extended to 3D NoCs for N = 1,2,4,8 *(iii)* adaptive odd-even (OE) turn model [36] extended to 3D NoCs, *(iv)* 2N-First turn model, *(v)* 3N-First turn model, *(vi)* 4N-First turn model, *(vii)* hybrid XYZ (XYZ and ZYX with replication on separate VCs), *(viii)* 2NP-First (2N-First and 2P-First with replication on separate VCs), *(ix)* 3NP-First (3N-First and 3P-First with replication on separate

VCs), *(x)* 4NP-First (4N-First and 4P-First with replication on separate VCs), and *(xi)* hybrid OE+IOE (OE and inverted OE with replication on separate VCs). Note that several of these schemes have been adapted to 3D NoCs for the first time. Other fault-tolerant routing schemes such as probabilistic flooding [28] and fully adaptive table based routing [31] were considered but their results are not presented because these schemes have major practical implementation concerns – their energy consumptions is an order of magnitude higher than other schemes and frequent deadlocks hamper overall performance. For our 4NP-First scheme, we used a prioritized valid path selection introduced in Section 3.4 and a replication threshold value of $\delta = 4\%$, based on our extensive simulation studies that showed a good trade-off between arrival rate and energy for $8\% \geq \delta \geq 2\%$. Ultimately, our goal with these experiments was to explore a variety of schemes with fault tolerance capabilities, and compare the results for our proposed 4NP-First routing scheme with the other schemes.

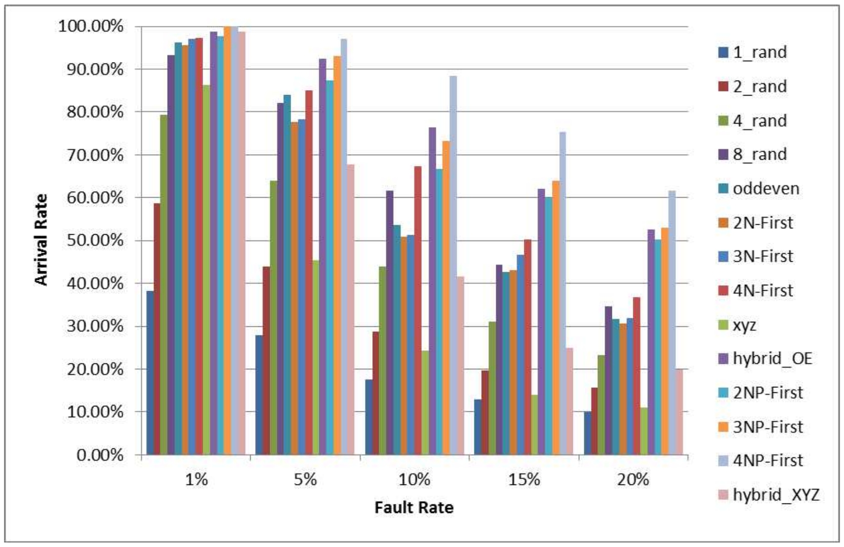
Figure 37(a)-(c) shows the successful packet arrival rate for all the routing algorithms over a spectrum of fault rates from a low 1% to a high of 20%, and with a flit injection rate of 20% (0.2 flits/node/cycle). On analyzing individual routing scheme performance, it can be immediately seen that the N-random walk schemes (1_rand, 2_rand, 4_rand, 8_rand) have a low arrival rate even for very low fault rates. This is due to frequent deadlocks during simulation which the scheme is susceptible to, causing packets to be dropped. The hybrid XYZ scheme performs well for very low fault rates, but as the fault rate rises, the lack of adaptivity inherent to the XYZ and ZYX schemes results in its arrival rate dropping rapidly compared to turn model based schemes. Overall, it can be clearly seen that the proposed 4NP-First routing algorithm has the highest arrival rate compared to other schemes across the three traffic types. The reason for the high arrival rate with the 4NP-First routing scheme is the greater path diversity due to smart path prioritization, which leads to better resiliency to faults in the network. For fault rates under 4%, replication in 4NP-First is disabled to save energy, which explains the slight dip in successful arrival rate for low fault rates. However, with replication enabled, 4NP-First outperforms every other scheme at low fault rates as well (i.e., 100% arrival rate at 1% fault rate). 4NP-First scales particularly well under higher fault rates that are expected to be the norm in future CMP designs.



(a)



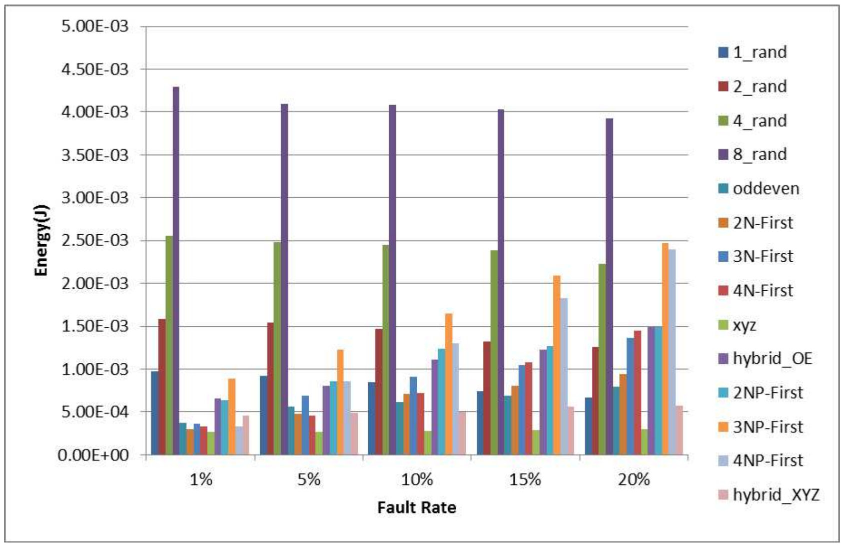
(b)



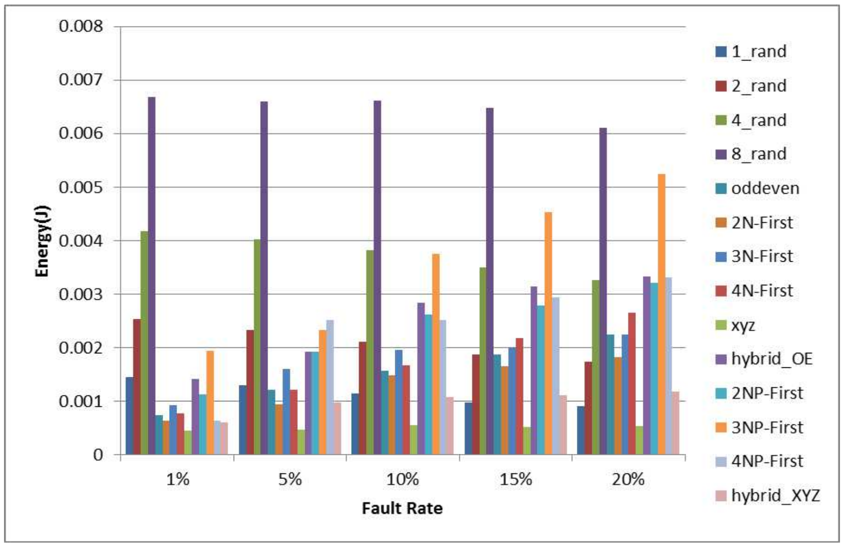
(c)

Figure 37 Packet arrival rate for 5×5×4 NoC (a) Uniform random (b) Transpose (c) Hotspot

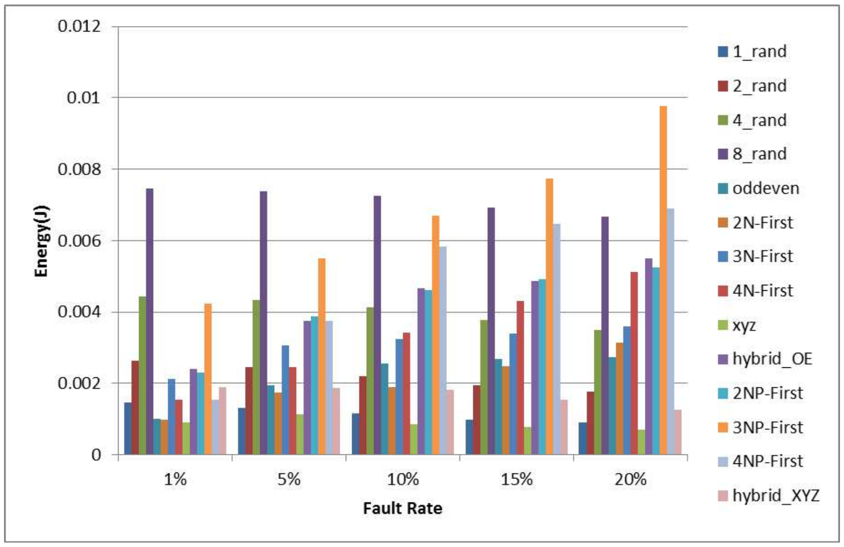
In addition to arrival rate, an increasingly important metric of concern for designers is the energy consumption in the on-chip communication network. Figure 38 shows the energy consumption for the different routing schemes under the same fault rates and types, and injection rates as in the previous experiment. The energy consumption accounts for circuit level implementation overheads as well as acknowledgements and retransmission in the schemes. It can be seen that the N-random walk fault-tolerant routing scheme has significantly higher energy consumption compared to the other schemes. This is due to the high level of replication with increasing values of N, as well as because of the non-minimal paths chosen by default to route the packets on. Among single VC schemes (oddeven, 2N-First, 3N-First, 4N-First, XYZ), the XYZ scheme has the lowest energy dissipation which explains its popularity in NoCs today. But as it is lacking in any fault tolerance, the scheme may not be a viable option for future CMPs.


5.00E-03
4.50E-03
4.00E-03
3.50E-03
3.00E-03
2.50E-03
2.00E-03
1.50E-03
1.00E-03
5.00E-04
0.00E+00
Energy(J)
1%
5%
10%
15%
20%
Fault Rate
1_rand
2_rand
4_rand
8_rand
oddeven
2N-First
3N-First
4N-First
xyz
hybrid_OE
2NP-First
3NP-First
4NP-First
hybrid_XYZ

(a)


0.008
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0
Energy(J)
1%
5%
10%
15%
20%
Fault Rate
1_rand
2_rand
4_rand
8_rand
oddeven
2N-First
3N-First
4N-First
xyz
hybrid_OE
2NP-First
3NP-First
4NP-First
hybrid_XYZ

(b)



(c)

Figure 38 Energy consumption for 5×5×4 NoC (a) Uniform random (b) Transpose (c) Hotspot

Among the rest of the (hybrid) schemes that incorporate replication, the communication energy for the hybrid XYZ scheme is the lowest. However, in light of its extremely low successful arrival rates as fault rates increase, the energy savings become irrelevant. For low fault rates below 4%, the 4NP-First scheme has the lowest energy because it disables replication to trade-off energy with fault tolerance. As the fault rate increases, initially the energy consumption for the 4NP-First scheme is competitive and generally lower than the other hybrid schemes. However, for higher fault rates, the energy consumption for 4NP-First becomes larger than all other hybrid schemes except 3NP-First. This rise in energy for 4NP-First happens primarily because the scheme uses more and more non-minimal paths as faults increase, to ensure high successful packet arrival rates. The other hybrid schemes have significantly lower packet arrival rates, because in a lot of cases, packets hit a fault soon after transmission and are dropped. Despite frequent retransmissions, the energy consumption of these hybrid schemes remains lower than that of 4NP-First. In some cases, the replication threshold based mechanism used in 4NP-First can be extremely useful to trade-off energy with packet arrival rate. For instance, for multimedia applications, occasionally dropped pixel data may not be perceivable by viewers, and in such scenarios the resulting lower energy consumption may lead to a much longer battery life and better overall user experience. Conversely, if fault resiliency is paramount, the value of δ can be reduced to close to 0, to enable the highest possible successful packet arrival rate. Given that ensuring high arrival rates is more important than saving energy in a majority of scenarios, the 4NP-First scheme is a better choice for environments with both low and high fault rates.

3.4.4 Summary

In this section, we proposed a novel fault-tolerant routing scheme (4NP-First) for 3D NoCs that combines the 4N-First and 4P-First turn models to create a robust hybrid routing scheme. The proposed scheme is shown to have a low implementation overhead and adapts to design time and runtime faults better than existing turn model, stochastic random walk, and hybrid dual virtual channel based routing schemes.

3.5 Chapter summary

In this chapter, by taking advantage of the path redundancy in the NoC mesh topology, we proposed different low-overhead hybrid fault-tolerant routing schemes for 2D and 3D NoCs. Our proposed schemes use replication opportunistically based on fault rate, and combine different turn models to achieve deadlock free packet traversal. Experimental results show that all of our proposed fault-tolerant schemes can achieve better fault tolerance (i.e., higher successful packet arrival rates) than traditional fault-tolerant routing schemes such as N-random walk and turn model based schemes.

CHAPTER 4
RELIABILITY MODELING FOR NOCS

In Networks-on-Chip (NoC), network interface (NI) components act as the glue logic to adapt communication between cores and the NoC. With ever-increasing NI complexity and technology scaling into the nanometer regime, transient single-event upsets (SEUs) have become a key design challenge not just for processing cores and caches, but also for NIs. Techniques to deal with these transient faults usually come at a power, area, and performance cost. In this chapter, we extend the concept of architectural vulnerability factor (AVF) from the microprocessor domain and propose a network vulnerability factor (NVF) to characterize the susceptibility of NoC components such as the NI to transient faults. For the first time, a detailed characterization of vulnerability is performed on a state-of-the-art AXI-based NI architecture using full system simulation, to understand masking effects and determine the probabilities that faults in NI sub-components manifest as errors in the final output. Our studies reveal that different NI buffers behave quite differently on transient faults and each buffer can have different levels of inherent fault tolerance to faults. Our analysis also considers the impact of thermal hotspot mitigation techniques such as frequency throttling on the NVF estimation.

4.1 Motivation

With the rising number of cores on the die, the complexity of the interconnection network-on-chip (NoC) fabric has been on the rise. Routers and network interface (NI) components have significant buffer resources and implementation footprint, and are increasingly susceptible to soft errors. There is thus a need to investigate the impact of soft errors on NoC resources, and ultimately, overall program execution in CMPs. Prior work on NoC reliability has been limited to studying techniques for fault-tolerant routing [59], using redundant resources [60], and employing error detection or correction coding schemes [61] to protect links and buffers. However, the susceptibility of individual NoC fabric structures to soft errors remains unexplored. For the first time, we propose to evaluate the vulnerability of a key NoC component: the network interface (NI) that provides the glue for computing cores to communicate with the NoC fabric. We first develop a sophisticated network interface architecture that enables a translation between the processor side AXI [62] communication protocol and the native packet-switched NoC data transfer protocol. We present the concept of a network vulnerability factor (NVF) and perform a detailed NVF analysis for sub-components in the state-of-the-art NI architecture, using a full system simulation with a real OS. The impact of thermal variations on NVF is also analyzed. Finally, we illustrate how knowledge of the NVF for NI components can enable low overhead NI implementations with high reliability.

4.2 Network interface architecture

In this chapter we present a brief overview of our network interface architecture that connects cores that use the popular AXI communication protocol from ARM [62] and the NoC fabric. We developed two NI architectures: a processor NI and a memory NI. The processor NI is responsible for sending write or read data requests to the memory modules. The memory NI interfaces with memory modules that can only accept write or read data requests from other cores, but cannot initiate any requests.

Figure 39 shows a block diagram of the processor NI architecture. The processor sends write or read signals to the NI through an AXI interface. For a memory write request from the processor, the NI receives AXI_W_CTL signals which contain the writing transaction ID, target address, burst size, burst type and other control information for the request. The NI accepts the request if there is sufficient available data buffer space (axiw_dat_buf) inside the NI to save the

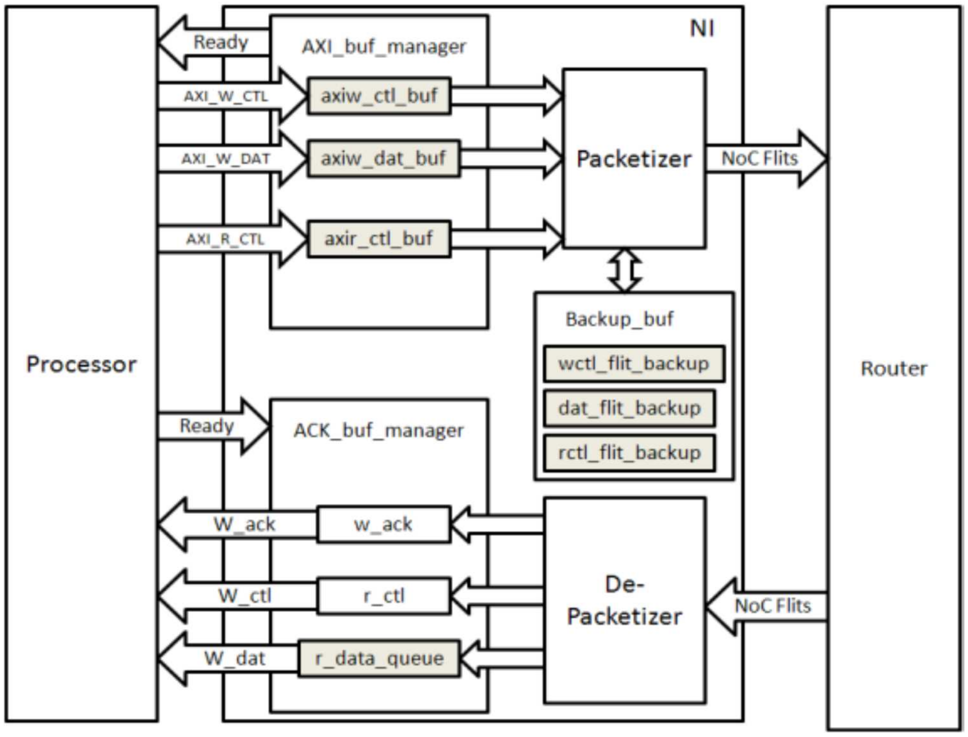


Figure 39 Processor network interface block diagram

data. After all the data for this writing transaction has been stored in the NI buffers, it is transferred to the Packetizer unit which converts the request into a NoC packet comprised of multiple flits: a header flit, body flits, and a tail flit. These flits are then injected into the NoC and hop through possibly multiple routers to reach the memory module. The NI supports end-to-end ACK/NACK flow control which requires flits to be backed up until an ACK signal is received from the destination. Read requests are sent out in a similar manner (except that the outgoing flit does not contain any data). On receiving the read data packet from the NoC, a De-packetizer unit is used to convert the packet into data and control signals in the AXI protocol. The de-packetized output is stored in a set of buffer managed by the Ack_buf_manager unit. An ACK signal is sent back to the source of the data packet, and then the received data is sent to the processor using the AXI protocol.

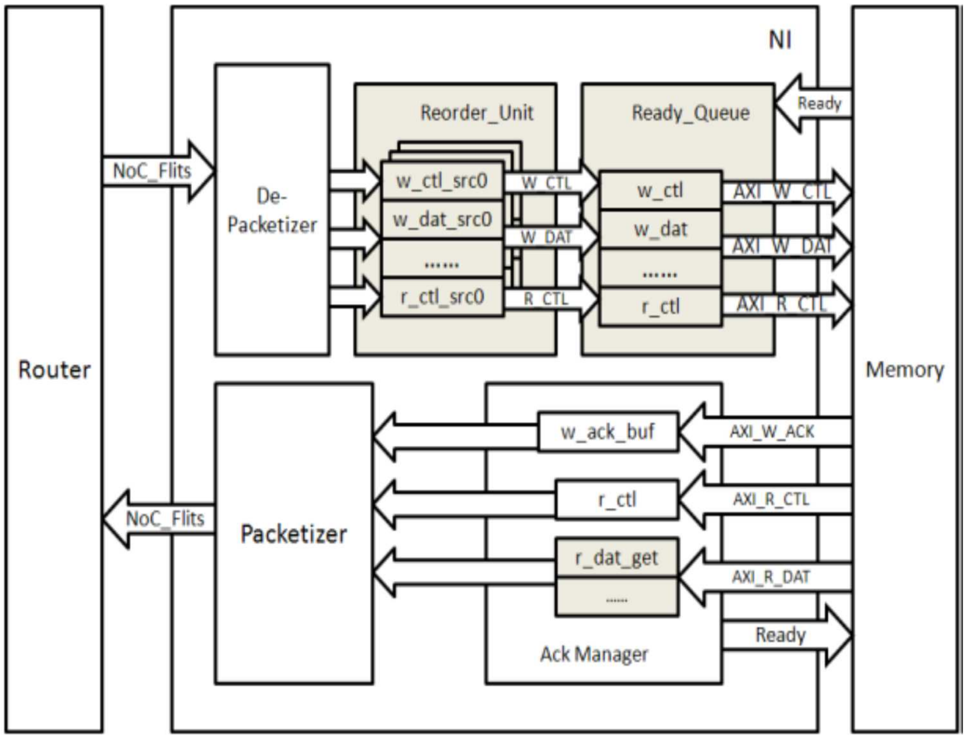


Figure 40 Memory network interface block diagram

Figure 40 shows a block diagram of the memory NI architecture. After a read or write request packet arrives at this NI, the De-packetizer unit removes NoC-specific routing information and then sends the data into a Reorder_Unit to put into order flits that may possibly have been received out-of-order. This unit has separate buffers dedicated to saving incoming packets from a cluster of cores. After all the data for a memory transaction has arrived, we move the data into the Ready_Queue, which buffers requests to be sent to memory. When the memory is ready, it accesses the queue to obtain pending requests and operates on them. If the request is to write data, an acknowledge signal is sent from the memory after the data has been written. This signal is saved in the w_ack_buf buffer. The Packetizer unit transforms the acknowledgement into a single ACK flit that is sent back to the source. For a read request, after the data has been read from memory, it is stored into the r_dat_get buffer in the NI. A round robin contention resolution scheme is used if a write acknowledge and read data simultaneously request access to the packetizer.

4.3 Network vulnerability factor (NVF)

Inspired by the AVF concept for processor and memory structures, we define a network vulnerability factor (NVF) that aims to describe the vulnerability of structures found in NoC fabrics. Intuitively, a masking effect should certainly be present in NoC routers and NIs, but to what extent is not clear. We propose to use the idea of architecturally correct execution (ACE) bits to calculate NVF for the network interface (NI) architecture, which is a key component of all NoCs. ACE bits in our case are the subsets of buffer entries that contain useful data during flit transmission. Any errors that involve these bits will cause a visible error in the final application results. In contrast, unACE bits do not affect the application results, even if soft errors cause changes to these bits. Then the NVF for a hardware component $\boldsymbol{\mathcal{H}}$ of size $\boldsymbol{S_{\mathcal{H}}}$ bits over an analysis window of N cycles can be expressed as

$$NVF = \frac{\sum_{i=1}^{N} \textit{(no.of ACE bits in } \mathcal{H} \textit{ at cycle i)}}{N \times S_{\mathcal{H}}}$$

which allows us to calculate soft error rate (SER) for $\boldsymbol{\mathcal{H}}$ as: $SER = \boldsymbol{S_{\mathcal{H}}} \times (FIT/bit) \times NVF \times TVF$, where *FIT/bit* is the Failure in Time per bit, and *TVF* is the timing vulnerability factor encapsulating the fraction of each cycle that a bit is vulnerable.

There are several reasons that can lead to the presence of unACE bits in a structure that end up making it less vulnerable to soft errors. Here we discuss the major detectable causes of unACE bits that are relevant to NI components in a NoC fabric.

4.3.1 Idle state

This is the most basic scenario: when there are no flits or data saved inside a buffer, we consider the buffer to be in an idle state. In this case, even when there are transient errors inside the buffer, as there is no critical information saved inside it at that moment, the final application result will not be affected. Buffers in the NI can be in an idle state in several scenarios. For instance, when the processor is waiting to get data from memory, then all the buffers related to a write will be idle in its NI till the data arrives. Even when data arrives, it may not completely fill up the buffers, causing the read buffers to be partially idle. Moreover, as long as the processor executes instructions in its ISA that do not involve data reads or writes from external memory, the NI buffers will remain idle. To generalize this further, applications can be characterized with a compute intensity and a memory intensity. If an application has a high compute intensity and a corresponding lower memory intensity, it spends a majority of its time executing compute (e.g., arithmetic, control) instructions instead of accessing memory, which will lead to NI buffers that are idle for longer, and thus with a higher aggregate of unACE bits.

4.3.2 Write-write

This scenario corresponds to a write-after-write event. It may so happen that a data that is propagated through the NI buffers and the NoC is overwritten before it is used. In such a case, even if the data gets corrupted, it does not impact the correctness of the program in any way. For instance, a processor may update the same location in memory twice. If there are no read requests from that location between the two writes, then the first write contributes to unACE bits in the buffers it occupied in the NI on its way to the memory (i.e., any corruption of the first write data does not affect program correctness). This scenario can happen fairly frequently with processors configured with write-through caches. In shared memory multicore systems, processors may also write data to shared memory that may never be read by other processors and is eventually overwritten (e.g., when an application thread is terminated and data meant for it in memory is no longer used, the memory space is eventually reclaimed by other application threads). The old data that is never used contributes to unACE bits in NI buffers.

4.3.3 Read mask

Often, data that is read into a processor may not be used in its entirety. For instance, consider the following assembly language code snippet:

```
And: c = a & b
     ldi r1 0xa010cfff
     ldi r3 0x000000ff
       and r1 r3 r1
```

After the "and r1 r3 r1" instruction is executed, the most significant three bytes inside r1 will be masked by r3. Thus, even if any of these three bytes in r1 gets corrupted, the output of the "and" instruction would not be affected. Therefore, the masked bits can be considered as unACE bits in the buffers of the NI that they propagate through, on the way to the processor. The example shown above is just one of the many possible cases when read masking can occur. Table 2 lists a few other instructions which may lead to partial masking of read data.

Table 2 Instructions with masking potential

Instruction	Description	Effect
and	And	c = a & b
bic	Bit Clear	c = a & ˜b
bis	Bit Set	c = a | b
ornot	Or Not	c = a | ˜b
sra	Shift Right Arithmetic	c = a >> (b % 64)
sll	Shift Left	c = a << (b % 64)
srl	Shift Right Logical	c = a >>> (b % 64)

4.3.4 AXI protocol

The AXI communication protocol is used to transfer control, address, and data from the processor to a NI. For specific NoC implementations, it is possible that some of the transferred AXI information may not be relevant, and thus the buffers used to store the unneeded information can be characterized as having unACE bits. As an example, consider the 64-bit AWADDR address signals that are buffered in the NI. If the global address space is only 48 bits, then the most significant 12 bits in the NI buffers storing the address will be unACE. Many other similar scenarios exist. For instance, if the 3-bit AWPROT protection signals and the 4-bit AWCACHE cacheable data signals are not used when communicating between certain cores, these can be considered as unACE as well, for the NI buffers.

4.3.5 Other scenarios

There may be other scenarios that can lead to unACE bits which we may not be able to detect easily due to the limitation of our experimental platform. For instance, it is possible that an application or OS initializes some memory for future use, but never uses the memory during the analyzed window of application execution. However, the memory may be used at some point, and such pathological cases may only be detectable by running simulations for large amounts of time (many weeks to months), which is not practical. Some situations involving pointer accesses and dynamically mutable data are also hard to trace, and therefore omitted from our analysis studies.

4.4 NVF estimation flow

In this section, we describe our approach to estimate NVF for NI components. We use this flow to calculate NVF for buffers shown shaded in Figure 39 and Figure 40. The buffers that were omitted in the analysis were primarily excluded because of their small size.

Figure 41 shows the estimation flow we use to calculate the NVF for each NI buffer. As a first step, we cross-compile benchmarks (SPLASH-2 [71] in our case) to execute on a modified M5 full-system simulator [72] running the Linux OS. The main output of running the benchmarks on the simulator is a set of traces that record detailed information with time stamps related to instructions executed, core ID, memory accesses, destination addresses for transfers, and so forth. We developed an instruction trace analyzer using Python scripts to process the instruction trace, extract memory access information, and detect masking scenarios (as described in section IV). These traces are fed into a modified open-source cycle-accurate NoC simulator (Nirgam [49]) that we enhanced with our NI models and more sophisticated router architectures and protocols (flow control, routing). The trace-driven simulation was then used to obtain NVF for the NI buffers being considered.

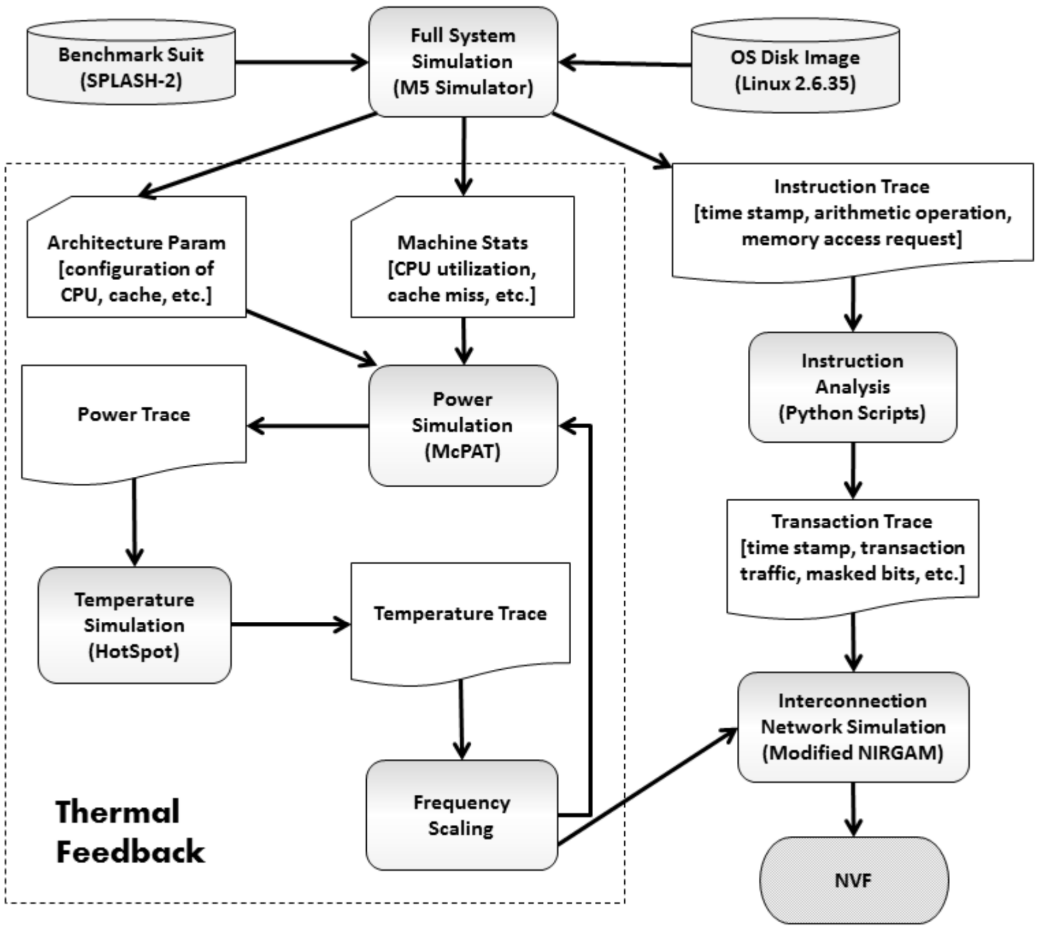

Benchmark Suit (SPLASH-2)
Full System Simulation (M5 Simulator)
OS Disk Image (Linux 2.6.35)
Architecture Param [configuration of CPU, cache, etc.]
Machine Stats [CPU utilization, cache miss, etc.]
Instruction Trace [time stamp, arithmetic operation, memory access request]
Power Trace
Power Simulation (McPAT)
Instruction Analysis (Python Scripts)
Transaction Trace [time stamp, transaction traffic, masked bits, etc.]
Temperature Simulation (HotSpot)
Temperature Trace
Interconnection Network Simulation (Modified NIRGAM)
Frequency Scaling
Thermal Feedback
NVF

Figure 41 NVF estimation flow

An important goal during this estimation process involved considering the impact of thermal variations on the estimated NVF. Thermal hotspots are an inevitable occurrence in highly integrated CMP architectures, and designers deal with them by throttling frequency of components affected by hotspots. This can however affect NVF calculation because under scaled frequency conditions, bits stay in NI buffers for longer. We wanted to incorporate the dynamics involved in such situations into our calculations. Figure 41 shows the thermal feedback component of the flow that was used to incorporate the effect of thermal variations during NVF estimation. We used the McPAT 0.8 [73] tool to process statistics generated by the M5 system simulator periodically and generate a power trace for the CMP die. This power trace was sent to Hotspot 5.0 [74] to generate an updated thermal map of the die. The thermal trace over time was used to identify points where frequency should be scaled to address hotspot scenarios. These points correspond to user-specified thresholds. If temperature exceeds an upper threshold, frequency needs to be scaled down; once the temperature falls below a lower threshold, frequency can be scaled up. This scaling information was fed into the trace driven simulation with Nirgam, to generate thermal-variation aware NVF estimates for the NI buffers.

4.5 Evaluation studies

4.5.1 Experiment setup

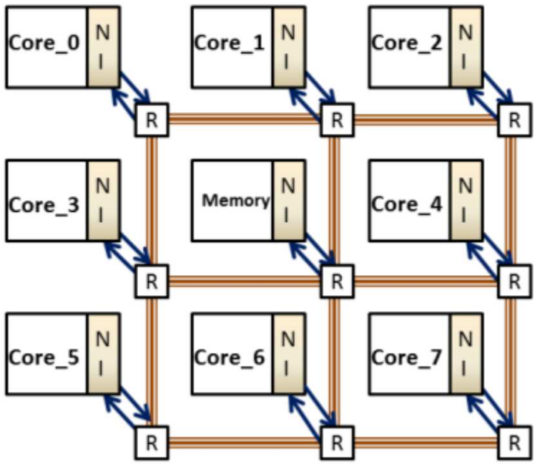


Figure 42 CMP network topology

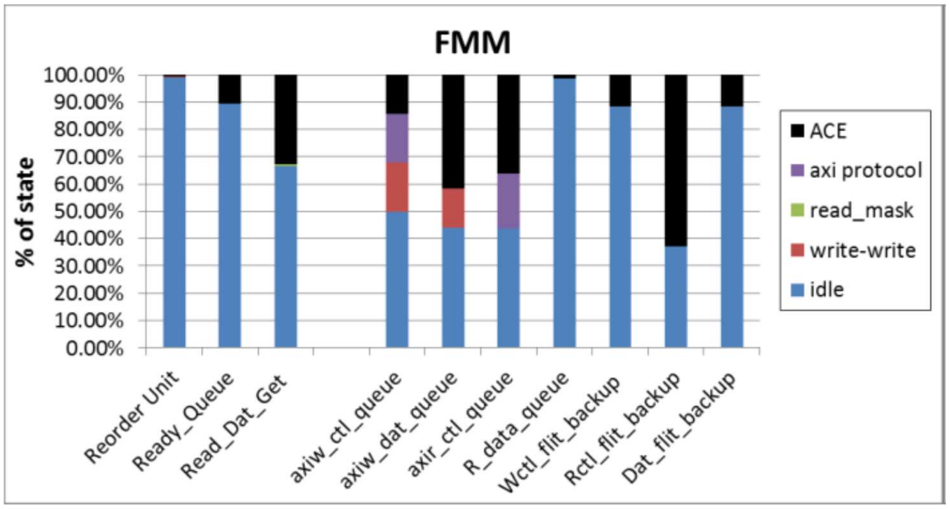
We consider a CMP platform with 8 compute cores and a shared global L2 cache memory, connected together by a 3×3 mesh network, as shown in Figure 42. The L1 instruction and data cache sizes are set to 64Kb and the L2 cache is 2048Kb. Every core runs at a baseline frequency of 2GHz. The implementation process technology is 32nm. We use an M5 full-system simulation [72] of three parallelized SPLASH-2 benchmarks (Fmm, Ocean, Water-nsquared) [71], to generate traces that are fed into a heavily modified Nirgam NoC simulator [49]. The detailed execution traces were generated for a span of 4 billion cycles, after fast-forwarding for 750 million cycles to account for an initial warm-up period (for OS boot up, etc). The Mcpat 0.8 tool [73] was invoked every 100K cycles to estimate the power dissipation trace, which was sent to the Hotspot 5.0 tool [74] to calculate a thermal map of the die, and guide the thermal-aware frequency scaling. Table 3 summarizes details such as the number of entries per buffer, bit width of each entry, and buffer functionality for the 10 NI buffers we analyzed. The 10 studied buffers comprised of 3 buffers from the memory NI and 7 buffers from the processor NI.

Table 3 Network interface buffer description

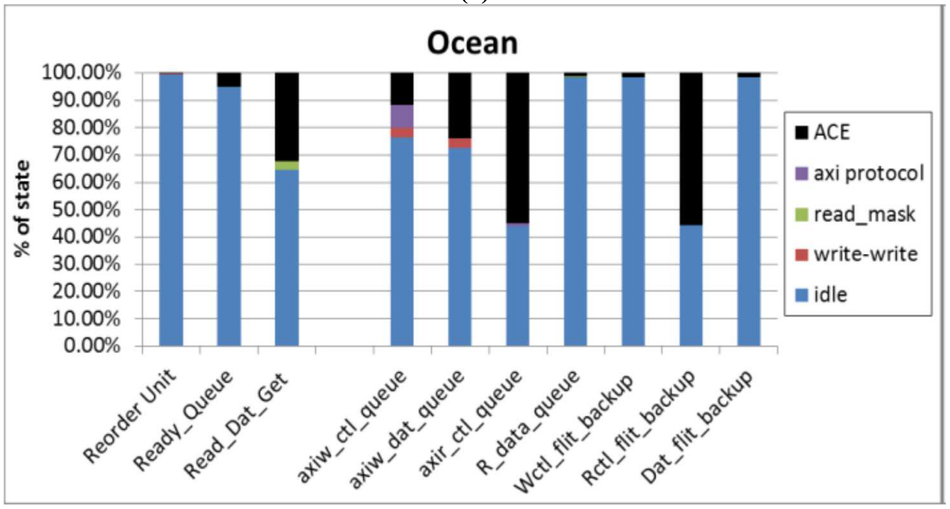
Buffer Name	Size (entries)	Width (bits)	Function
Reorder_Unit (mem)	32	32	reorder request transaction data
Ready_Queue (mem)	10	32	queue pending memory requests
Read_Dat_Get (mem)	20	32	store received read data from memory
axiw_ctl_queue (μp)	32	54	store write request AXI control signals from processors
axiw_dat_queue (μp)	32	46	store write data from processors
axir_ctl_queue (μp)	32	54	store read request AXI control signals from processors.
R_data_queue (μp)	5	32	store read data from memory
Wctl_flit_backup (μp)	10	32	backup write control signals sent from processor NI
Rctl_flit_backup (μp)	1	32	backup read control signals sent from processor NI
Dat_flit_backup (μp)	15	32	backup data sent from processor NI.

4.5.2 Thermal-aware NVF estimates for NIs

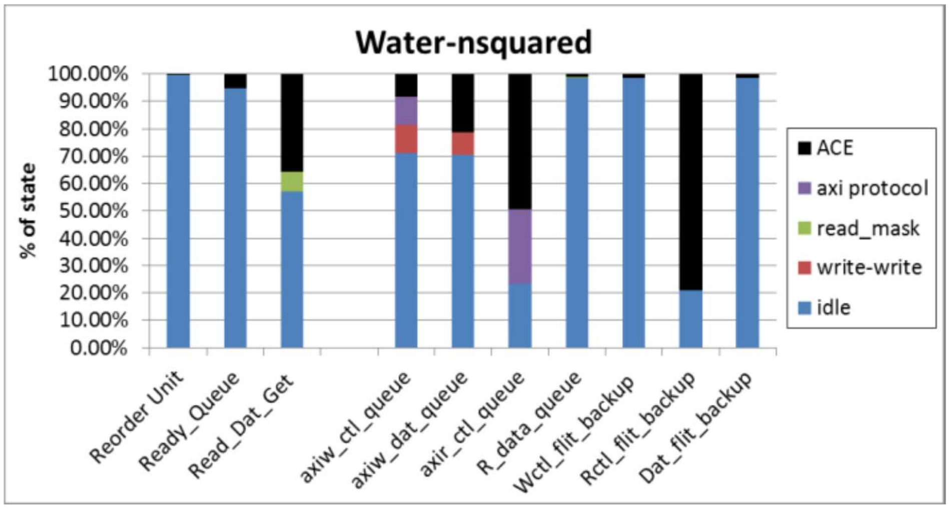
For brevity, we omit results for the baseline case in which we calculated NVF for NI buffers without considering thermal effects. Figure 43 shows the results for the more realistic scenario of thermal-aware estimation of buffer states that accounts for frequency throttling in the presence of hotspots. The first three buffers (leftmost three bars) correspond to the memory NI buffers, while the remaining seven buffers belong to the processor NI. The figures show the % of cycles that the buffers contain ACE bits and unACE bits. The NVF for each buffer is essentially the % of cycles that a buffer contains ACE bits.



(a)



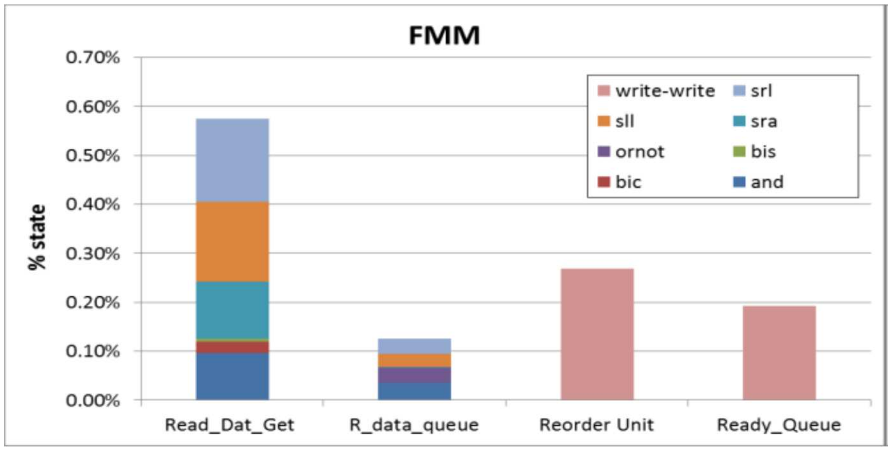
(b)



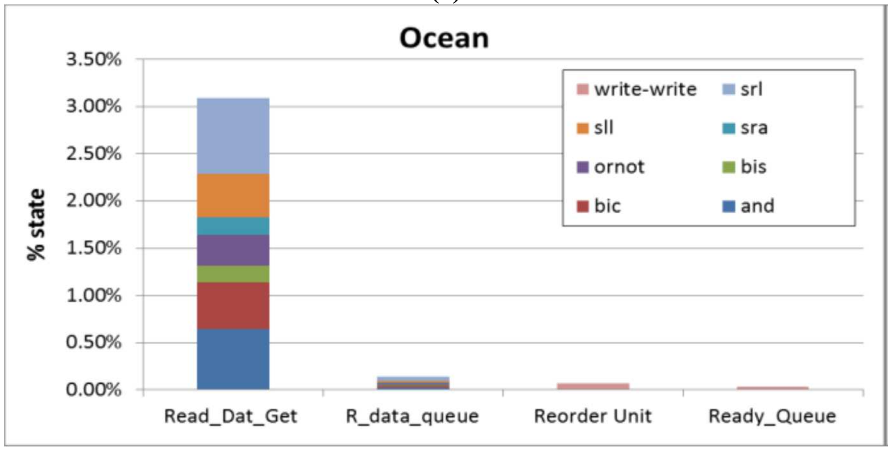
(c)

Figure 43 Breakdown of different architectural and microarchitectural states for the NI buffers with (a) FMM (b) Ocean (c) Water-Nsquared; (NVF for each buffer is the % of cycles it contains ACE bits)

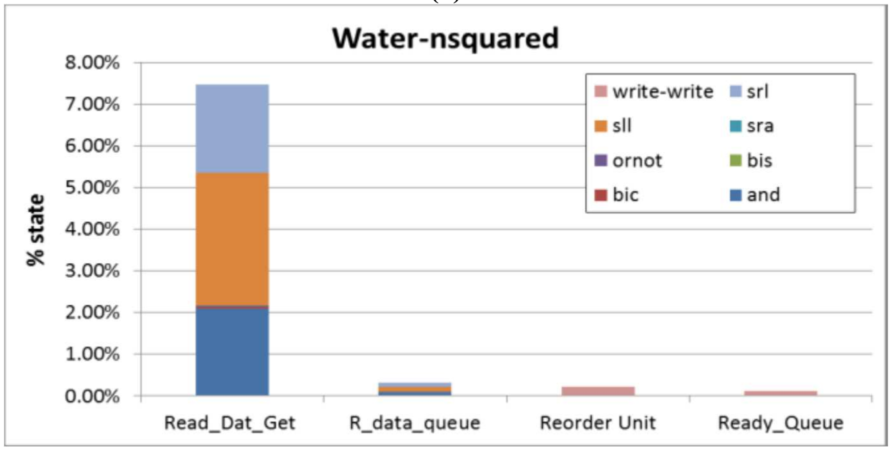
It can be seen from Figure 43 that NVF varies significantly across buffers in the NI, ranging from 0.35% up to as much as 55.75%. In almost all cases, the buffers hold unACE bits for a majority of the time, with the idle state being the biggest contributor. The relatively small idle state for the Rctl_flit_backup buffer is because the buffer ends up storing backup copies of read requests (that occur more frequently than write requests) till the read data is received by a processor from the memory, which involves a round trip latency and the memory access latency. In contrast, due to the low relative frequency of writes, the Dat_flit_backup and Wctl_flit_backup buffers remain in idle state for a majority of the time. Similarly, the Reorder_Unit did not need to reorder packets because we made use of an XY routing scheme that sends packets in order. Moreover, the processors rarely sent out transactions out of order. Thus the Reorder_Unit forwards most requests quickly to the Ready_Queue. Due to low concurrent memory accesses and therefore memory congestion in our experiments, the Ready_Queue does not need to buffer requests for very long, and therefore has a relatively high idle state.



(a)



(b)



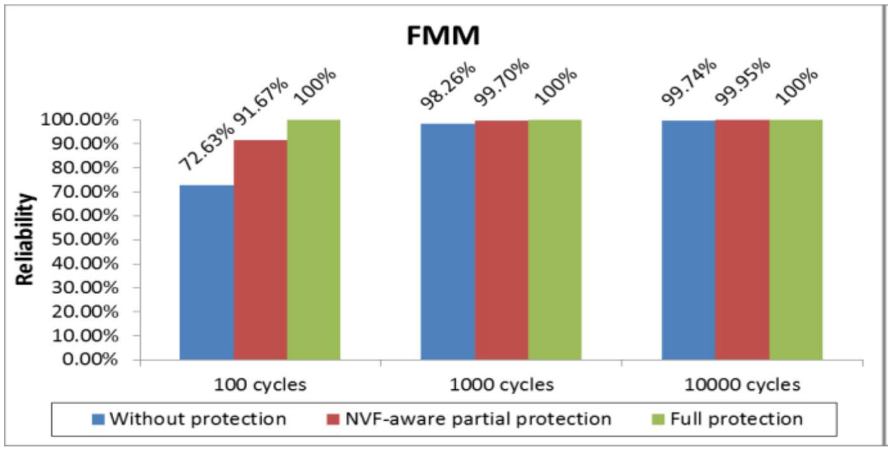
(c)

Figure 44 Breakdown of write and read masking for subset of NI buffers, for (a) FMM (b) Ocean (c) Water-Nsquared

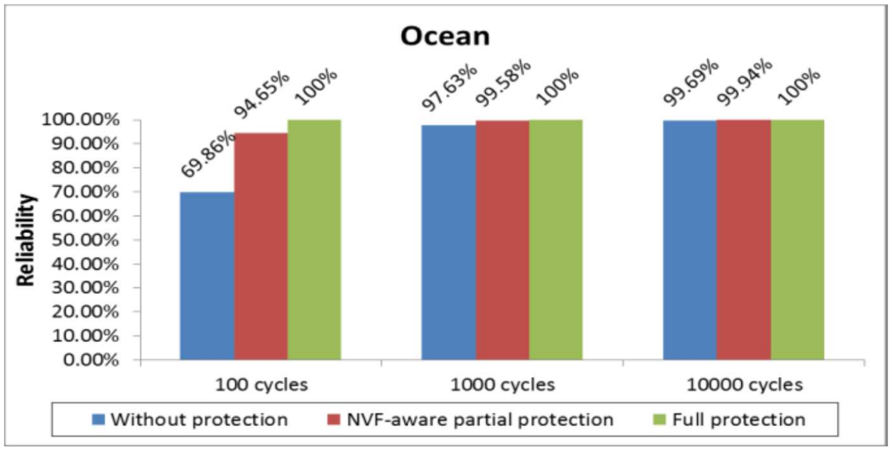
A few buffers have notable read and write masking which contributes to the unACE state. Figure 44 shows a more detailed breakdown of this masking for a subset of NI buffers with notable masking. AXI-centric buffers axiw_ctl_queue, axiw_dat_queue, axir_ctl_queue are omitted because they have easily observable write-write and AXI protocol masking from Figure 43. The read masking in Figure 44 is broken down based on the specific instructions that cause the masking, and corresponds to the set of instructions presented earlier in Table 2 (in Section IV). The masking effect is most prominent for the Read_Dat_Get buffer which holds read data from memory for long periods due to network congestion, before packetization and injection into the network. While the same masking states exist for the R_data_queue buffer in the processor NI that eventually receives the read data from memory, the buffer in general flushes the data to the processor quickly, and thus has low masked data occupation duration.

4.5.2 NI reliability

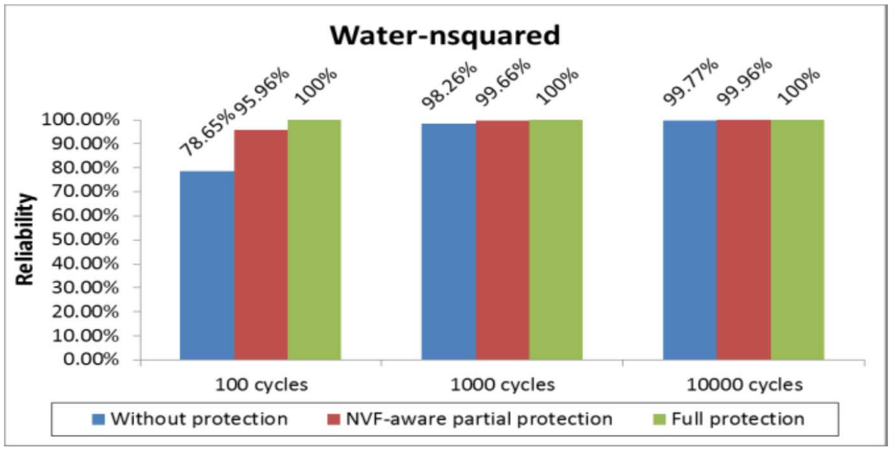
From the results presented in Figure 44 and Figure 45, most NI buffers can be seen to have a low NVF due to various masking effects. The results provide important insights on how certain NI buffers such as axir_ctl_queue and Rctl_flit_backup are more susceptible to soft errors than other buffers, because they store critical (ACE) data more frequently. This observation can help guide the design of NI architectures that must be optimized for reliable operation.



(a)



(b)



(c)

Figure 45 Reliability vs fault rate for the three NI design approaches, across the benchmarks (a) FMM (b) Ocean (c) Water-Nsquared

We conducted a study that explored the reliability and overheads of different approaches for designing NI architectures. The baseline case is a NI architecture designed without protection mechanisms such as encoding or redundancy. Such an NI architecture has low area and power overhead, but lacks resilience to transient faults. Another approach, employed in recent literature for NoC routers, is to design multiple spare routers [70], or use triple modular redundancy (TMR) for each buffer [69]. Since prior work has mainly addressed the design of reliable routers and ignored reliable NI design, we extrapolate these approaches to NI architectures, and create a fully protected scenario in which all buffers in the NI are protected using TMR. Such an architecture is expected to have very high reliability. The third and final approach to NI design is one that employs partial protection for a subset of NI buffers guided by our NVF analysis. In this approach, if a NI buffer has a NVF of 5% or more (referred to as the protection threshold), we make use of TMR to protect the buffer. For the other buffers, no protection mechanism is used. Figure 45 shows the reliability of the NI architecture designed using the three approaches discussed above. Three different error injection rates were considered, with a single soft error being randomly injected into the NI buffers every 100, 1000, or 10000 cycles. We assume such accelerated error rates in order to expose the error behavior and measure the reliability of the architectures. Reliability is measured in terms of how often the incident fault causes failure in the program – a close to 100% reliability implies that the faults do not adversely impact program behavior. It can be seen that NI architectures without any protection are particularly susceptible to faults, even with lower fault incidence rates. More importantly, the reliability of the NVF-aware partial protection approach is very close to that for a fully protected approach.

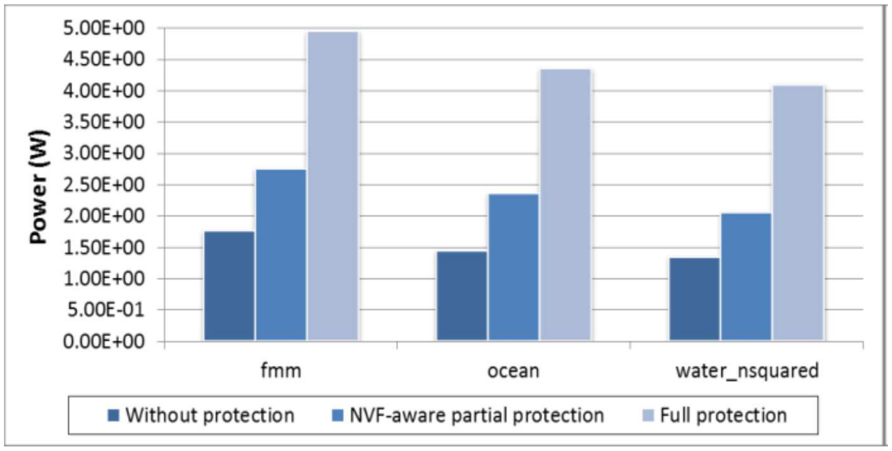


Figure 46 Average power dissipation for the three NI design approaches across fmm, ocean, and water-nsquared benchmarks

Figure 46 presents the overall power dissipation for the three NI design approaches. The results aggregate the power dissipation of all NIs in the 9 core CMP we consider. The limitation of using a fully protected NI architecture approach is obvious – the power dissipation is almost 3×

higher than for the baseline NI architecture without any protection. In contrast, the NVF-aware partially protected architecture has almost 50% lower power dissipation than the fully protected approach. The advantage of analyzing vulnerability for the components of the NI architecture is thus quite apparent. By using insights gained from application- and platform-aware NVF analysis of various NI buffer structures, intelligent partial protection mechanisms can be designed that achieve high reliability at low implementation overheads. Moreover, by varying the protection threshold in our proposed approach, a trade-off between reliability and power dissipation can be achieved: a higher threshold value than the 5% chosen in this research would allow sacrificing some reliability to achieve NI architectures with even lower power dissipation.

4.6 Chapter summary

In this chapter, for the first time, a detailed characterization of vulnerability was performed on an AXI-based network interface (NI) architecture to understand masking effects, using full system simulation and incorporating thermal-aware frequency throttling. Our research has identified dominant masking effects and sub-structures in NIs that are particularly vulnerable to faults. The insights from this chapter can enable opportunistic low power protection of NI and router architectures to meet reliability goals in emerging NoC designs.

CHAPTER 5
FAULT-TOLERANT NOC SYNTHESIS

In sub-65nm CMOS process technologies, networks-on-chip (NoC) are increasingly susceptible to transient faults (i.e., soft errors). To achieve fault tolerance, Triple Modular Redundancy (TMR) and Hamming Error Correction Codes (HECC) are often employed by designers to protect buffers used in NoC components. However, these mechanisms to achieve fault resilience introduce power dissipation overheads that can disrupt stringent chip power budgets and thermal constraints. In this chapter, we propose two system-level frameworks, RESYN and HEFT, to trade-off energy consumption and fault-tolerance in the NoC fabric by exploring different core to tile mapping, routing algorithms and fault-tolerant NoC protection configuration strategies. RESYN is a design-time synthesis framework which employs a nested evolutionary algorithm approach to guide the mapping of cores on a die, and opportunistically determine locations to insert fault tolerance mechanisms in the NoC to minimize energy while satisfying reliability constraints. To more accurately consider runtime characteristics of NoC traffic, we also propose a hybrid synthesis framework HEFT with design-time core to die mapping but a run-time fault tolerant NoC reconfiguration strategy to reduce fault protection energy overheads in NoC routers. Given the increasing importance of reliability in the nanometer era for CMPs, this chapter provides important perspectives that can guide the reduction of overheads of reliable NoC design.

5.1 Design time fault-tolerant NoC synthesis

In this section, we propose a novel NoC synthesis framework (RESYN) that, for the first time, co-optimizes reliability and energy consumption to design a robust and energy-efficient NoC fabric for CMPs. The major contributions of this section are: (i) we develop a system-level power model for components in a NoC router from gate-level models at 45nm; (ii) we employ a network vulnerability factor (NVF) metric to develop a reliability model of the NoC fabric; (iii) we extract communication traces from real-world applications using full-system simulation to accurately analyze the impact of NoC customizations on real workloads; and (iv) we solve the problem of core-to-die mapping and application-specific NoC synthesis for reliability and energy-efficiency by using a nested genetic algorithm (NGA) as part of a novel synthesis framework (RESYN). Experimental results show that RESYN can reduce energy costs by 14.5% on average, compared to a fully protected NoC, while still maintaining more than a 90% fault tolerance.

5.1.1 Motivation

To ensure reliable operation in NoC routers, designers often use encoding techniques such as Hamming Error Correction Codes (HECC) [69][76], or replicate sub-components with Triple Modular Redundancy (TMR) schemes that can tolerate transient error in one of the sub-components [77][78]. The primary emphasis is to protect input and output buffers where packets spend the majority of the time inside the router. However, such protection has an undesirable side effect: an increase in power dissipation and, consequently, energy costs. As most CMPs are already severely constrained by power ceilings and energy budgets (e.g., mobile devices run on batteries with limited energy capacity), the additional power and energy costs can increase time-to-market and significantly drive up costs. Thus the problem of designing a reliable NoC that is also energy-efficient is an important one. The problem of synthesizing (customizing) NoC fabrics at the system level based on application-specific goals has been addressed by many researchers [79]-[84], where the emphasis has been on trading-off energy-efficiency and performance. However, to date, none of these efforts have integrated reliability concerns as part of their system-level NoC synthesis frameworks.

5.1.2 Problem formulation

In this section we present the inputs, assumptions, and objectives for our system-level synthesis problem.

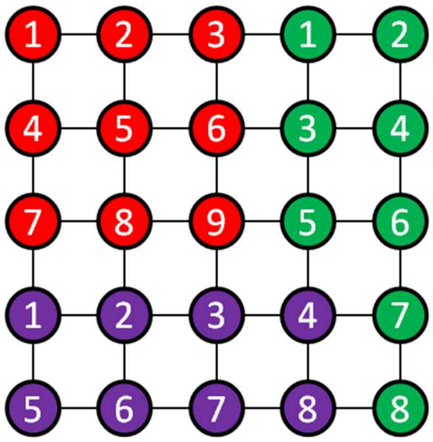


Figure 47 Example of 5x5 mesh NoC with 3 applications mapped to three unique application islands

Inputs:

- n×n mesh NoC based MPSoC platform: We assume a regular mesh NoC topology composed of a 2D regular array of identical tiles. Each tile contains a processing core (or memory) and a NoC router.
- k applications running on the MPSoC: Each application is set to run on a group of cores that constitute a well-defined island of cores in the mesh. Figure 47 shows an example of a 5×5 mesh based MPSoC on which three different applications are mapped.
- Application constraints: An application is defined by an application communication graph (ACG) *G(V,C)*, which is a directed graph, where each vertex $v_i \in V$ represents an IP core (processor or memory), and each directed arc $c_{ij} \in C$ represents the communication between source core vi and destination core v_j. Each edge c_{ij} has a weight $w(c_{ij})$ to represent communication constraints. In this section, we focus on latency constraints (in terms of hop

counts) that are application-specific.

- NoC router architecture: We consider a router architecture that contains 5 input/output ports (coming from and going to the North, South, East, West and Core directions), a crossbar switch, route-compute unit (RC), switch allocator (SA) and virtual channel allocator (VA). Figure 48 show the router architecture.

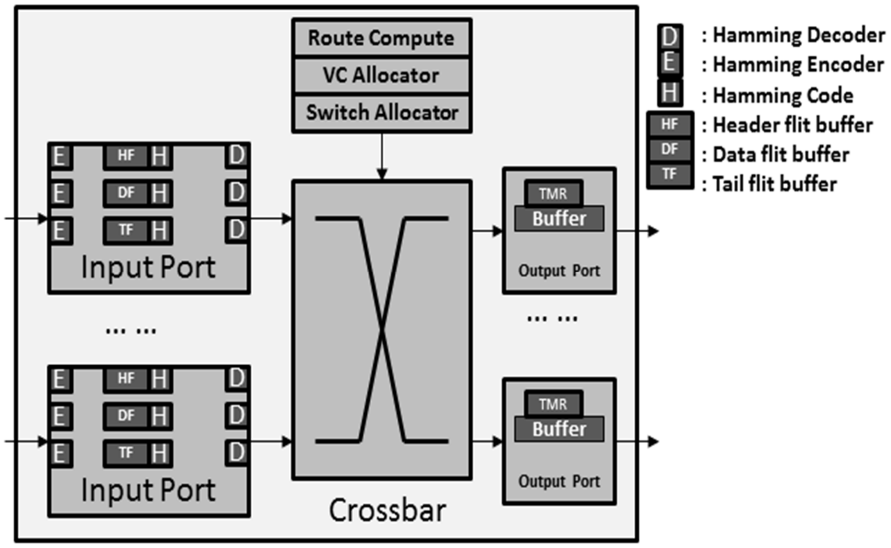


Figure 48 NoC router architecture with buffer protection

- Protection mechanisms: To enable fault tolerance in NoC routers, we employ two mechanisms: Hamming Error Correction Codes (HECC) and replicating sub-components with Triple Modular Redundancy (TMR). Both mechanisms can correct single bit errors in components. But unlike TMR, HECC operation entails additional latency to encode and decode data. TMR implementations however possess a larger area and power footprint, compared to HECC implementations.

Assumptions:

- We assume that there is no communication between applications; i.e., cores only communicate with other cores within an island, and not with cores in other islands. Within an application island, all the cores communicate primarily with one or more memory nodes (tiles) via packet-switched wormhole switching.

- We assume at most a single event upset (SEU) that affects a single bit in a NoC buffer, at any given time. NoC buffers are extremely susceptible to SEUs and therefore our primary focus is to protect these buffers from SEUs that can occur unexpectedly at runtime;
- We assume that to keep implementation overheads low while still enabling fault tolerance in NoC routers, it is preferable to employ HECC for the input FIFO port buffers (that can store multiple flits), and TMR for the smaller single flit capacity output port buffers (to keep TMR implementation costs low), as shown in Figure 48.

Problem Objective: *we propose to accomplish the following objective: (i) for each application, find a mapping of cores to the 2D mesh-based die that meets application communication latency constraints; and (ii) synthesize the NoC fabric for the entire chip by customizing NoC routers at each node with appropriate protection mechanisms (discussed above) to meet a designer-specified reliability goal while minimizing energy consumption*

In the rest of this section, we discuss how we define and calculate reliability and power dissipation in the NoC fabric.

5.1.3 Reliability model

Not all faults that occur on a chip eventually affect the final program outcome. For example, a bit flip in an empty translation lookaside buffer (TLB) entry will not affect the program execution. Based on this observation, Mukherjee et al. [85] defined a structure's Architectural Vulnerability Factor (AVF) as the probability that a transient error in the structure finally produces a visible error in the output of a program. At any point of time, a structure's AVF can be derived via counting the important bits required for Architecturally Correct Execution (ACE) in the structure, and dividing them by the total number of bits of the structure. Such ACE analysis has helped to derive an upper bound on AVF using performance simulation for processor buffers and caches in recent years [75].

Inspired by the AVF concept for processor and memory structures, we consider a network vulnerability factor (NVF) that aims to describe the vulnerability of structures found in NoC fabrics. Intuitively, a masking effect should certainly be present in NoC routers, but to what extent is not clear. We propose to use the idea of architecturally correct execution (ACE) bits to calculate NVF for the buffers in a NoC router, which are the key components of all NoCs. ACE bits in our case are the subsets of buffer entries that contain useful data during flit transmission. Any errors that involve these bits will cause a visible error in the final application results. In contrast, unACE bits do not affect the application results, even if soft errors cause changes to these bits. Then the NVF for a hardware component $\mathcal{H}$ of size $S_{\mathcal{H}}$ bits over an analysis window of N cycles can be expressed as

$$NVF = \frac{\sum_{i=1}^{N} (no.of\ ACE\ bits\ in\ \mathcal{H}\ at\ cycle\ i)}{N \times S_{\mathcal{H}}}$$

which allows us to calculate soft error rate (SER) for $\mathcal{H}$ as: SER = $S_{\mathcal{H}}$×(FIT/bit)×NVF×TVF, where FIT/bit is Failure in Time per bit, and TVF is timing vulnerability factor which represents the fraction of each cycle that a bit is vulnerable.

There are several reasons that can lead to the presence of unACE bits in a structure that end up making it less vulnerable to soft errors. The major detectable causes of unACE bits that we found to be relevant to NoC router components in a NoC fabric are: (i) Idle time: this is the most basic scenario - when there are no flits or data saved inside a buffer, we consider the buffer to be in an idle state. In this case, even when there are transient errors inside the buffer, as there is no critical information saved inside it at that moment, the final application result will not be affected; (ii)

Write-after-Write: this scenario corresponds to a write-after-write event. It may so happen that a data that is propagated through the NoC router buffers is overwritten before it is used by a processor. In such a case, even if the data gets corrupted, it does not impact the correctness of the program in any way; (iii) Read masking: often, data that is read into a processor may not be used in its entirety. For instance, during an xor operation, some bits may not affect the outcome of the operation. These bits can be considered unACE. We consider all of these factors in our NVF calculations for each buffer, in each router in the NoC.

Clearly the NVF value associated with a buffer in a particular NoC router depends on the data traffic flowing through it, which in turn depends on the manner in which cores are mapped to the die and the routing algorithm used. Thus it is vital to consider the core mapping problem together with the NoC synthesis problem, which is the approach we take. The reliability value R(i,j) of each buffer i in NoC router j, knowing its network vulnerability factor value NVF(i,j), can then be calculated as:

$$R_{(i,j)} = 1 - NVF_{(i,j)}$$

Then the overall reliability RNoC for the entire NoC can be expressed as the product of the reliability values of each buffer in each router in the NoC:

$$R_{NOC} = \prod_{\forall i, \forall j} R_{(i,j)}$$

We assume that there can be at most a single bit error in a NoC router buffer at any given time. So it is possible to correct the error using HECC or TMR. In other words, if a buffer is protected by an HECC or TMR mechanism, then its reliability value can be safely assumed to be 1. We use knowledge of communication flows through a router after a core-mapping and minimal routing step to generate NVF estimates for each NoC router buffer, which when combined can allow us to determine the reliability of the entire NoC.

An important motivation for considering the core-to-die mapping problem together with the problem of balancing energy and reliability during NoC-based MPSoC synthesis in our work is that the unACE bits inside NoC router buffers depend on the data traffic flowing through them, which in turn depends on manner in which cores are mapped to the die. As the unACE bits in NoC routers determine NoC router reliability, and consequently the reliability of the entire NoC architecture, core-to-die mapping has a direct impact on overall network reliability.

5.1.4 Power model

Many research efforts make use of the Orion 2.0 [50] tool to calculate NoC router power. However, the tool does not allow us to get power overheads of protection mechanisms. To achieve a detailed power characterization of NoC routers with protection mechanisms, we implemented a NoC router at the RTL level, and performed logic synthesis and gate-level analysis using Synopsys Design Compiler and Primetime tools [86] to obtain power dissipation for each sub component within a NoC router. We also implemented HECC and TMR mechanisms in the NoC router module, and obtained the overhead of integrating these protection mechanisms into NoC routers.

Table 4 lists the average dynamic and static power values for the various NoC router components, for the 45nm CMOS process technology. We omit the power overhead for the small tail flit input buffers for brevity. As in Orion 2.0, we assume a 0.5 switching probability to calculate the average power values. Using these power values, it is possible to calculate the energy of a flit flowing through the NoC router. We use knowledge of communication flows through a router after a core-mapping and minimal routing step together with NoC router sub-component power values to generate application-specific communication energy estimates. This energy value is obviously impacted by any protection mechanisms that are integrated in the NoC router.

Table 4 Router module power library (45nm)

	Input Header Buffer	Input Data Buffer	Input HECC Header	Input HECC Data	Output Buffer	TMR Output Buffer
Dynamic	216.8uw	1.36mw	425.65uw	1.51mw	45uw	267.55uw
Static	794nw	3.54uw	1.76uw	5.18uw	120nw	1.43uw
	Link	**Crossbar**	**SA**	**VA**	**RC**	
Dynamic	51.3uw	121uw	105uw	101uw	91.5uw	
Static	915nw	2.56uw	2.33uw	2.51uw	1.02uw	

5.1.5 RESYN synthesis framework

In this section, we present our framework for reliability-aware and energy-efficient NoC-based CMPs synthesis. Figure 49 shows the high-level synthesis flow of our RESYN framework. The inputs consist of the application communication graph (which provides communication latency constraints), a 2D-mesh NoC platform with application islands already mapped (i.e., shape of each application island has been defined, but the cores for each application within the islands are unmapped), the NoC power library which helps estimate communication energy, and the NVF library that contains application-specific NVF data for each buffer in every NoC router. The RESYN framework employs a nested genetic algorithm (NGA) approach. The outer GA performs application core mapping to the application islands for each application, with the aim of satisfying communication latency constraints. The inner GA performs NoC synthesis for the given core mapping from the outer GA, by configuring each NoC router to meet an overall reliability constraint while minimizing energy at the system-level. In the following subsections, we provide a brief overview of genetic algorithms and then describe the NGA algorithm used in the RESYN framework.

5.1.5.1 Overview of genetic algorithms

Genetic algorithms (GAs) [87] generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover. A GA involves the evolution of a population of individuals over a number of generations. Each individual of the population is assigned a fitness value whose determination is problem dependent. At each generation, individuals are selected for reproduction based on their fitness value. Such individuals are crossed to generate new individuals, and the new individuals are mutated with some probability. The objective of a GA is to find the optimal solution to a problem. However, because GAs are heuristics, the solution found is not always guaranteed to be the optimal solution. Nevertheless, experience in applying GAs to a variety of problems has shown that often the goodness of the solutions found by GAs is sufficiently high.

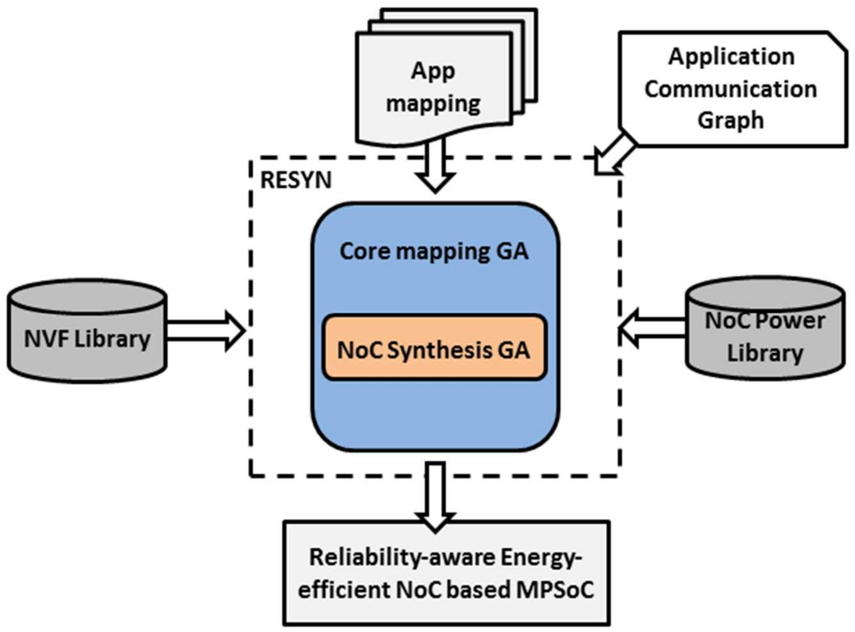

App mapping
Application Communication Graph
RESYN
Core mapping GA
NoC Synthesis GA
NVF Library
NoC Power Library
Reliability-aware Energy-efficient NoC based MPSoC

Figure 49 RESYN synthesis flow

5.1.5.2 Nested genetic algorithm (NGA)

The core mapping and NoC synthesis problems that we consider in this section are instances of constrained quadratic assignment problems which are known to be NP-hard [88]. We use a nested genetic algorithm (NGA) approach to accomplish the dual objectives of core mapping and

NoC synthesis. In our NGA implementation, the outer loop (external) GA performs core mapping, and the inner loop (internal) GA uses the generated core mapping from the external GA to perform a search on the NoC design space, to synthesize a reliability-aware and energy-efficient NoC.

Algorithm 1: External GA

Input: Application mapping, communication latency constraints, power library, NVF library

```
1: Generate_core_mapping_chromosome();
2: for generation=0 to MAX_GEN_OUTER do
3:     Internal_GA()
4:     Solution_List();
5:     Selection()
6:     Cycle_Crossover()
7:     Mutate()
8: end for
```

Output: core mapping solution, synthesized NoC architecture

Algorithm 1 shows the pseudo code for the external GA. Initially, we generate core mapping solution chromosomes (step 1). This chromosome is a core id array for all the cores on the die, and is encoded as:

$$coreid_{1,1}coreid_{1,2}\ldots coreid_{1,n(1)},\ldots,coreid_{m,1}coreid_{m,2}\ldots coreid_{m,n(m)}$$

where there are m applications (and thus application islands) on the die, with the number of cores in the ith application given by n(i). As an example, if we have a 3×3 NoC with 9 tiles, and two applications that require 4 cores (with core id's 1 to 4) and 5 cores (with core id's 5 to 9) respectively, then a mapping chromosome value of 153482976 implies that core 1 of application 1 is mapped to tile 1, core 5 of application 2 is mapped to tile 2, and so on. Step 1 randomly generates 10 different core mapping arrays that satisfy latency hop constraints. As there are many feasible core mapping solutions that satisfy latency hop constraints, the goal is to iteratively generate new core mappings (over MAX_GEN_OUTER generations) and use the internal GA to synthesize the NoC fabric to minimize a fitness function (steps 2-8). We define the fitness function for our problem as the ratio of the NoC communication energy and the NoC reliability. Thus, minimizing the fitness value entails decreasing energy and increasing reliability, moving us towards a more desirable solution.

In step 3, we call the internal GA to calculate each chromosome's fitness value. The internal GA performs NoC synthesis to generate the NoC energy (E_{NoC}) and reliability (R_{NoC}), which allows us to calculate the fitness value of a chromosome. We use the internal GA to generate the minimum fitness values of all chromosomes. We describe the internal GA in Algorithm 2, later in this section.

In Step 4, we store the non-dominated solutions (chromosomes) in a solution list which is iteratively updated. In Step 5, we perform chromosome selection. A standard proportional technique is used to randomly choose whether or not drop a chromosome and generate a new one, as described next. Suppose the fitness value of the 10 chromosomes are fit_i, i=0~9. We find the sum of the 10 fitness value (fit_{sum}) and calculate the proportion of each chromosome's fitness value within the sum ($rfit_i$, i=0~9). We then calculate a cumulative number for each chromosome ($cfit_i$) such that $cfit_0 = rfit_0$, $cfit_i = cfit_{i-1}+rfit_i$. We randomly generate ten numbers $rand_i$ between 0 and 1, with i=0~9. If $rand_0<cfit_0$, then chromosome 0 is kept. If $cfit_{i-1}<rand_i<=cfit_i$, then we keep the chromosome i. We replace chromosomes that are discarded with new randomly generated chromosomes that satisfy latency constraints.

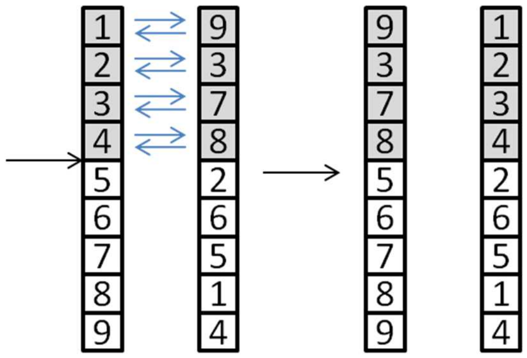


Figure 50 Traditional crossover operation

In Step 6 we perform crossover. Figure 50 shows a traditional crossover operation, where a crossover point is selected, and then all the entries above the point are switched between two parent chromosomes. In our core mapping problem, we note that after this crossover, in chromosome 1 core 9 is mapped twice and core 1 is not mapped at all, which creates an invalid solution. To avoid such a scenario, we replace the traditional crossover with the cycle crossover operator [89] which does allow us to generate valid crossover solutions.

After the cycle crossover we check whether the two new chromosomes that have been generated meet the latency constraint. If they both do, then we replace the two original chromosomes with the new ones. If both new chromosomes do not meet latency constraints, then we keep the original two chromosomes and move to the next step. If only one of the new chromosomes can meet the latency constraint, we replace one of the original chromosomes that has the higher fitness value with the new one. Finally, in Step 7 we perform mutation. For each chromosome, we randomly generate a number between 0 and 1. If the number is smaller than 0.2, then we perform a mutation on the current chromosome. This mutation is performed by randomly selecting two cores within an application island and switching them in the current chromosome. After the outer GA terminates, we obtain a final set of solutions that trade-off reliability and energy efficiency.

Algorithm 2: Internal GA

```
Input: Core mapping, NVF library, power library
1: Build_chromosome_routing_mask()
2: Generate_noc_synthesis_chromosome()
3: for generation=0 to MAX_GEN_INNER do
4:     Evaluate_fitness_value();
5:     Selection();
6:     Crossover();
7:     Mutate();
8: end for
Output: NoC configuration with minimum fitness function value
```

Algorithm 2 shows the pseudo code for the internal GA. For a given core mapping configuration, this internal GA explores various configurations of NoC routers, with the goal of finding a configuration that minimizes the fitness function value (E_{NoC}/R_{NoC}), as defined earlier. The encoding of the chromosome for the internal GA is shown below, where g_{ijk} is a gene which represents whether buffer k in port j of NoC router i is protected or not.

$$g_{110}g_{111}g_{112}g_{113} \cdots g_{ij0}g_{ij1}g_{ij2}g_{ij3} \cdots g_{(n*n)j0}g_{(n*n)j1}g_{(n*n-1)j2}g_{(n*n)j3}$$

$$g_{ijk} = \begin{cases} 0: router\ i\ port\ j\ buffer\ k\ is\ not\ protected \\ 1: router\ i\ port\ j\ buffer\ k\ is\ protected \end{cases}$$

$$k = \begin{cases} 0: input\ header\ buffer \\ 1: input\ data\ buffer \\ 2: input\ tail\ buffer \\ 3: output\ buffer \end{cases}$$

In Step 1, based on the core mapping configuration from the external GA, we establish minimal routing paths between communicating cores and build a mask array based on the routing paths. The mask is similar to the chromosome except that for the NoC router buffers that do not participate in any communication, the corresponding entry in the chromosome is set to 0. In step 2, we randomly generate 10 chromosomes (that represent solutions with varying degrees of protection across NoC router buffers on the chip), masked by the mask array created in step 1. Steps 3~8 attempt to perform traditional crossover, in addition to mutation and selection of chromosomes in a manner similar to the external GA. After MAX_GEN_INNER iterations, the best NoC configuration with the lowest fitness value is returned.

Ultimately, after the external GA finishes, we obtain a set of solutions, each with a unique mapping of application cores to their respective application islands on the die and with NoC routers uniquely configured with HECC and TMR protection mechanisms, to trade-off reliability and energy efficiency for on-chip communication.

5.1.6 Experimental results

5.1.6.1 Experimental setup

Figure 51 shows the setup we used to generate the data needed for the RESYN synthesis framework. In our experiments we considered combinations of three SPLASH-2 benchmark applications [90] mapped to a mesh of 25 (5×5) ALPHA processors, each with 64kB L1 instruction and data caches. We assume that tiles in the mesh have already been allocated to applications (i.e., application islands are assumed to be given), based on prior designer analysis, e.g., proximity of an applications' nodes to specific I/O pins. We also assume that inter-core latency constraints (in terms of hop counts) for each application are given, based on pre-computed criticality analysis of communication flows.

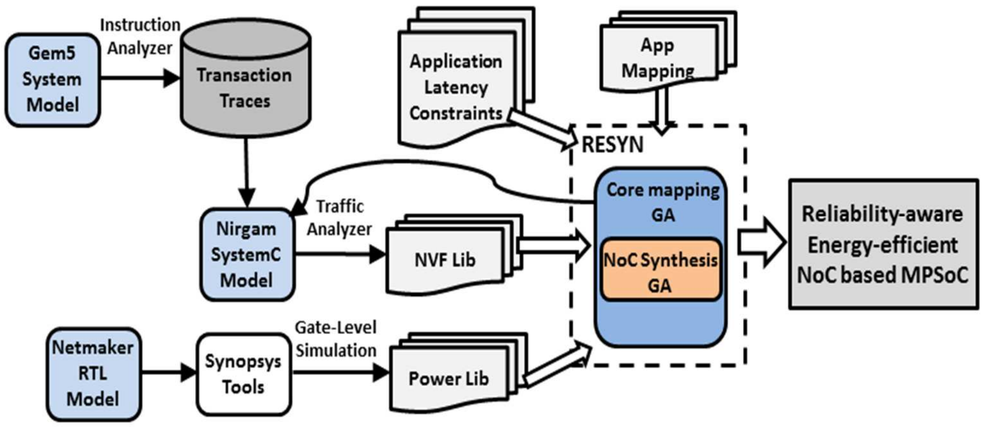


Figure 51 Experimental setup for RESYN synthesis flow

We use a modified Gem5 full system simulator [91] with the SPLASH-2 benchmark applications to generate instruction traces for each core. We parsed 50 million "memory read" and "memory write" instructions from each application, and used the Nirgam SystemC-based NoC simulator [92] for trace-driven simulation, to generate NVF values for NoC router buffers for each core mapping configuration. Every time the core mapping is changed, this simulation is run again to update the NVF library (as different core mappings cause changes in traffic flow, which in turn changes the NVF value of NoC router buffers).

We use Synopsys VCS [93] to run simulations with RTL models of the NoC built by modifying the Netmaker library [94], to get signal traces. We also perform logic synthesis using Synopsys Design Compiler, and gate level power analysis using Synopsys Primetime to generate the power library for NoC router sub-components and integrated protection mechanisms. This power data is used to calculate communication energy, once a core mapping and routing paths for communication flows are established.

5.1.6.2 RESYN comparative analysis

We compare the solutions generated by RESYN with two other design alternatives: (i) unprotected, which refers to an MPSoC with latency-aware core to die mapping but without any protection mechanisms in router architectures; and (ii) fully protected, which refers to an MPSoC with latency-aware core to die mapping, and with all NoC router buffers protected (HECC on input buffers, TMR on output buffers).

Figure 52 shows the energy costs for the three approaches, for three different combinations of SPLASH-2 benchmark applications. For RESYN, we select the solution with the minimum energy that still meets at least 90% reliability (i.e., $R_{NoC} >= 0.9$). The results are shown relative to the energy consumption of the fully protected case. It can be seen that energy consumption of the unprotected case is the least, which is not surprising, considering that this approach does not protect NoC router buffers from transient faults. Compared to the energy consumption of the fully protected case, our RESYN framework enables savings of 14.5% in communication energy costs on average.

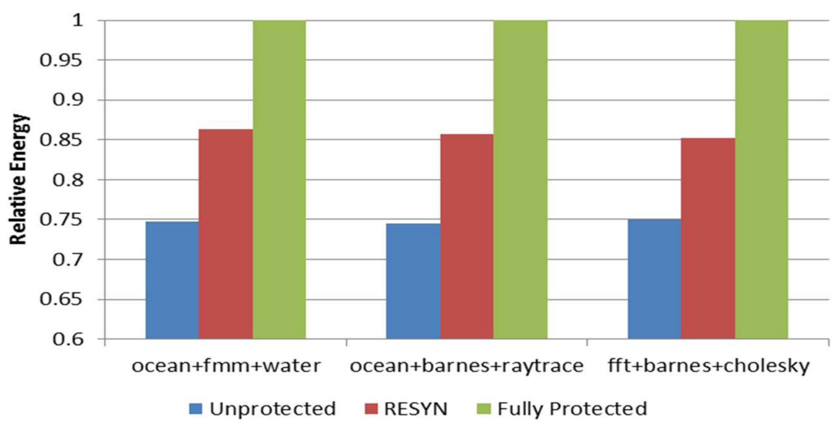


Figure 52 Energy cost comparison for RESYN with unprotected and fully protected design alternatives

Figure 53 shows the reliability values for the same solutions for which energy costs are shown in Figure 52. It can be seen that the unprotected case is not very viable, given its low reliability in the face of transient faults. The solutions generated by RESYN all have a reliability of greater than 90%. The results in Figure 53 and Figure 54 demonstrate how RESYN allows a convenient mechanism for trading off reliability with energy at the system-level, allowing designers to minimize energy given a reliability goal. Techniques such as end to end ACK/NACK flow control can be utilized to complement the NoC fabric generated by RESYN, to ensure that no error goes undetected.

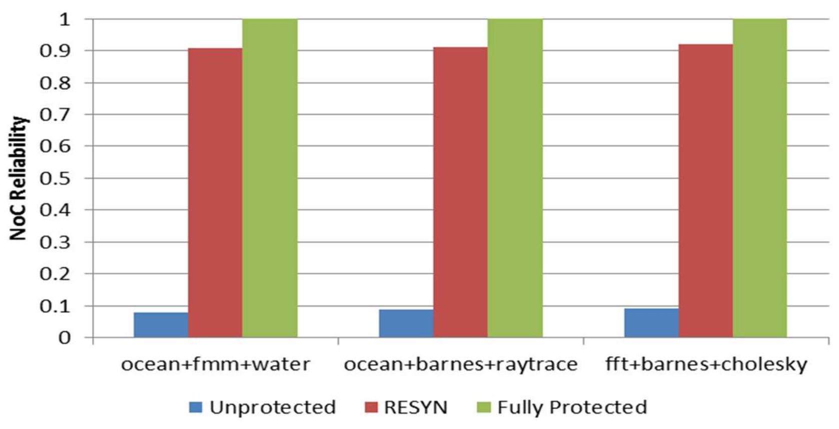
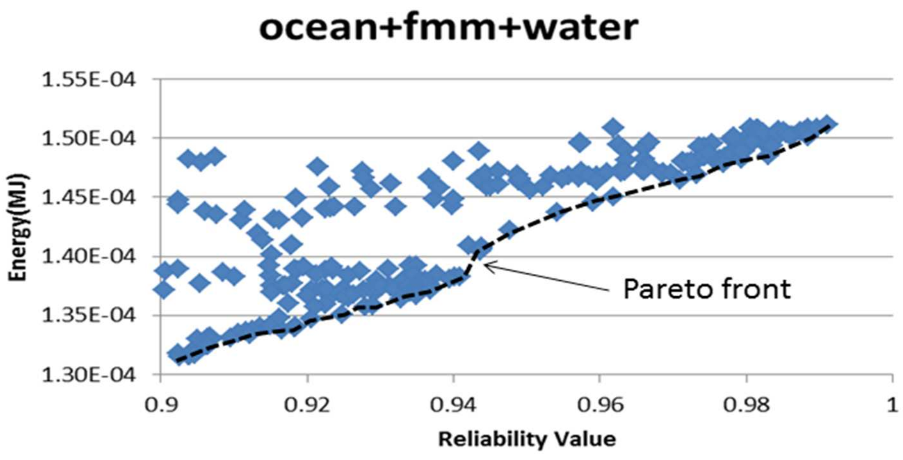
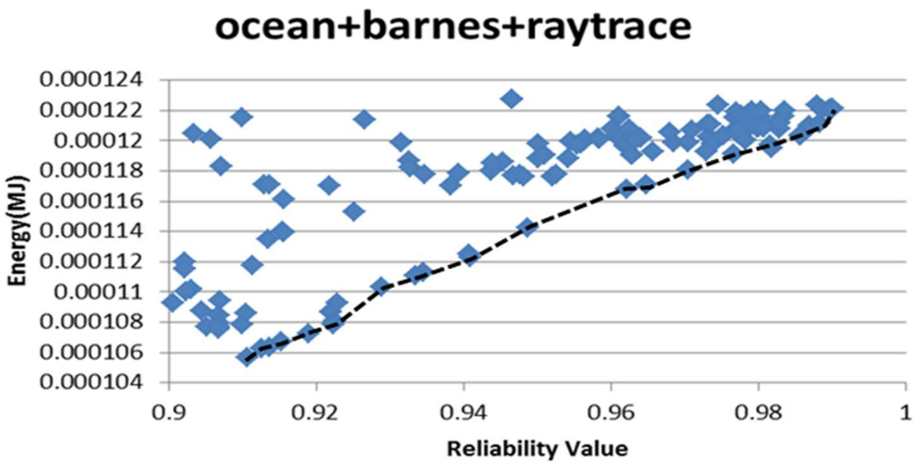


Figure 53 Reliability comparison for RESYN with unprotected and fully protected design alternatives

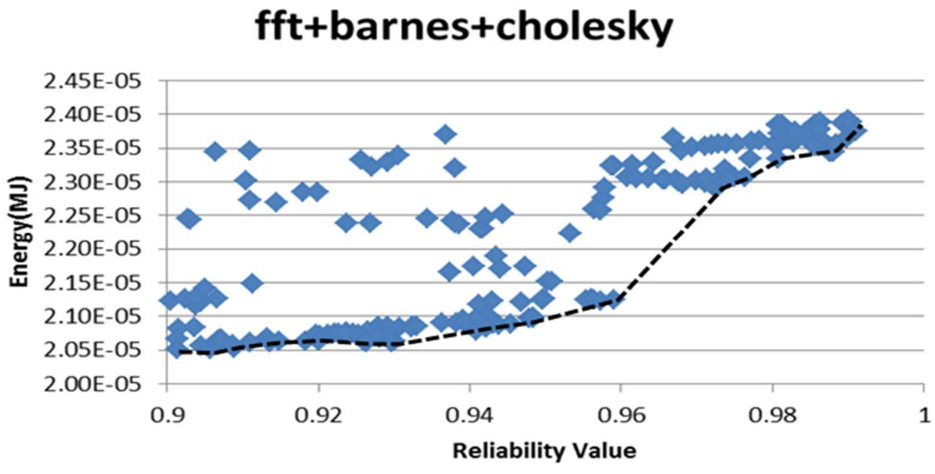
In some scenarios, a reliability goal of 90% may be too low, and designers may require the ability to trade-off reliability with energy efficiency over a broad spectrum of higher reliability values. Figure 54(a)-(c) show the solution set generated by RESYN for the three different combinations of applications that we study. The figures show the energy and reliability of several solutions whose reliability varies from 90% to 100%. Designers can then select the appropriate energy-efficient solutions along the Pareto front, for any desired value of reliability. Even for higher values of reliability (i.e., more stringent reliability constraints), it is possible to get significant energy savings. For example, for a 95% reliability constraint in fft+barnes+cholesky, RESYN generates a solution that still saves a significant 12% energy consumption over the fully protected case. RESYN thus represents an invaluable system-level tool for designers in the era of nanometer design uncertainty, providing a design space exploration environment that enables comprehensive trade-offs between reliability and energy-efficiency for NoC based MPSoC applications.



(a)



(b)



(c)

Figure 54 Generated solution set using RESYN with Pareto front for the three applications (a) ocean+fmm+water, (b) ocean+barnes+raytrace, (c) fft+barnes+cholesky

5.1.7 Summary

In this section, we proposed a novel design-time framework (RESYN) to trade-off energy consumption and reliability in NoC-based CMPs. RESYN employs a nested genetic algorithm (NGA) approach to guide the mapping of cores on a die, and opportunistically determine the locations to insert fault tolerance mechanisms in NoC routers to minimize energy consumption while satisfying reliability constraints. Our experimental results show that RESYN can reduce communication energy costs by 14.5% on average compared to a fully protected NoC design, while still maintaining more than a 90% fault tolerance. RESYN also enables a comprehensive trade-off between reliability and energy-efficiency for more stringent reliability constraints, generating a Pareto set of solutions which may reveal opportunities for energy savings even for high reliability configurations; e.g., for a higher 95% reliability constraint, a notable 13% energy savings can still be achieved for some applications.

5.2 HEFT design/run time hybrid fault-tolerant NoC synthesis

5.2.1 Motivation

As a design-time framework RESYN can trade-off energy consumption and reliability in NoC-based CMPs, but it does not accurately consider runtime characteristics of NoC traffic, such as unpredictable congestion due to traffic-bursts that impact both performance and energy. Changing NoC conditions at runtime also lead to a change in the fault vulnerability of various NoC components [97]. Thus, there is a need to complement design time NoC-based CMPs synthesis frameworks with mechanisms that can adapt to changing runtime conditions, to optimally manage the trade-off between fault tolerance, energy, and performance.

In this section, we propose a hybrid framework for synthesizing NoC-based CMPs called HEFT. This hybrid framework performs analysis and parameter tuning at design time and at runtime to aggressively manage trade-offs between energy, fault-tolerance, and performance. At design time, the framework solves the problem of mapping cores executing multi-application workloads on to a die while satisfying application latency and bandwidth constraints. At runtime, the framework utilizes a novel reliability predictor that dynamically estimates fault vulnerability of NoC router components, to efficiently manage energy overheads of enabling fault tolerance mechanisms in the NoC. Our main contributions in this section are:

- We create a NoC reliability model that is based on fault vulnerability analysis and improves upon prior models;

- We propose and explore runtime fault vulnerability predictors and adaptive protection management for NoC architectures;
- We explore the impact of two different design time core-to-die mapping techniques for enabling energy-reliability trade-offs; the techniques minimize NoC energy and support tile shutdown for dark silicon driven enhancements, respectively;
- Our experimental results on several memory-intensive and compute-intensive parallel application workloads show a notable reduction in energy and increase in reliability while satisfying application performance goals, compared to multiple prior works on reliable system-level NoC design.

5.2.2 Problem formulation

In this section, two problems that impact the performance, reliability, and energy consumption of a NoC architecture are explored. The first problem is to decide the mapping of processing cores (on which application tasks have already been mapped) on to the die. The second problem is to select mechanisms to enable fault tolerance in NoC routers that support data transfers between cores. As these two problems are closely related to each other, we need to solve both of them in a unified manner.

Figure 55 shows an example of mapping each core in an application communication graph (ACG) to an CMP with a 2D mesh topology based NoC fabric. ACG is a directed graph, in which each vertex $C_i \in V$ represents an IP core (processor or memory), and each directed edge $comm_{ij} \in E$ represents communication dependencies between the source core C_i and destination core C_j. The volume of communication from C_i to C_j is $v(comm_{ij})$ and $v(C_i) = \sum_{j \in V}(v(comm_{ij}) + v(comm_{ji}))$. Each edge $comm_{ij}$ has a weight $B(comm_{ij})$ for application-specific communication bandwidth constraints, and a weight $L(comm_{ij})$ for latency constraints (maximum number of hops allowed between cores C_i and C_j).

We assume that the designer can pre-define application zones on the mesh-based die, as shown in Figure 55, allowing multiple applications to co-execute on the regular MPSoC die. In general, communication occurs at an intra-application node level, i.e., only between nodes belonging to an application, and not between nodes of different applications. For a given NoC link width (e.g., 32 bits), we assume platform constraints related to the maximum bandwidth that the NoC links can support B_{link}, which is dependent on the CMOS technology node and chip power constraints. Thus, for all mapped cores utilizing a link, it is required that the total utilized bandwidth on that link $T_{link\_max} \leq B_{link}$.

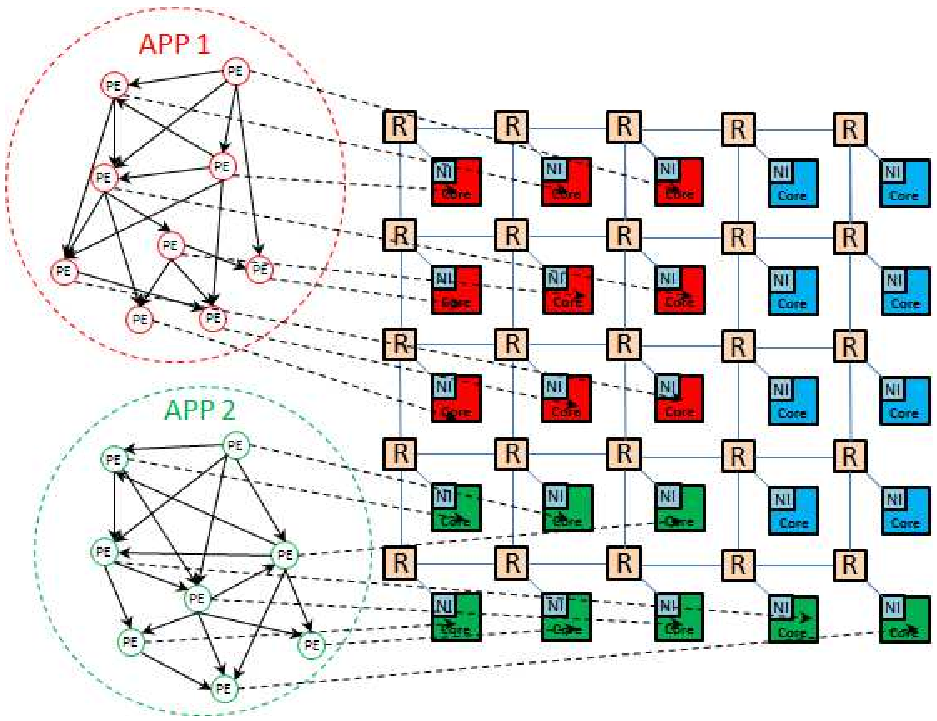


Figure 55 Multi-application core-to-die mapping example

The buffers in NoC routers are most susceptible to transient single event upset (SEU) errors, and we assume that errors on the relatively small footprint inter-router links are negligible. To protect NoC router buffers against SEU errors, we can employ two widely-used protection mechanisms:

Hamming Error Correction Codes (HECC) and replicating components with Triple Modular Redundancy (TMR). Both mechanisms can correct SEU errors. But unlike TMR, HECC operation entails additional latency to encode and decode data. On the other hand, TMR implementations possess a larger area and power footprint than HECC. Table 5 shows the average power, latency, and area overhead trends for using either only HECC or only TMR to protect all buffers inside a NoC router, compared to a baseline NoC router without any protection. Results are obtained using an RTL model of a five staged pipelined NoC router that was enhanced with HECC and TMR protection mechanisms, and taken through logic synthesis using Synopsis tools and layout for a 45nm TSMC library.

Table 5 HECC vs. TMR overhead comparison

	HECC	TMR	Integrated
Power overhead	8%	259%	12%
Latency overhead	150%	10%	75%
Area overhead	5%	200%	7%

It is apparent that using only HECC or TMR has high associated penalties. To minimize these overheads, we chose to judiciously combine HECC and TMR in a NoC router, by protecting input FIFO buffers (that can store multiple flits) in NoC routers with HECC due to its lower area and power overhead, and protecting the smaller output NoC buffers (that only store a single flit) using TMR to minimize latency overhead. This integrated HECC/TMR configuration, shown in Figure 56 and whose overheads are shown in the last column of Table 5 ("Integrated"), results in a relatively fast packet traversal with a reasonable area and power overhead. s

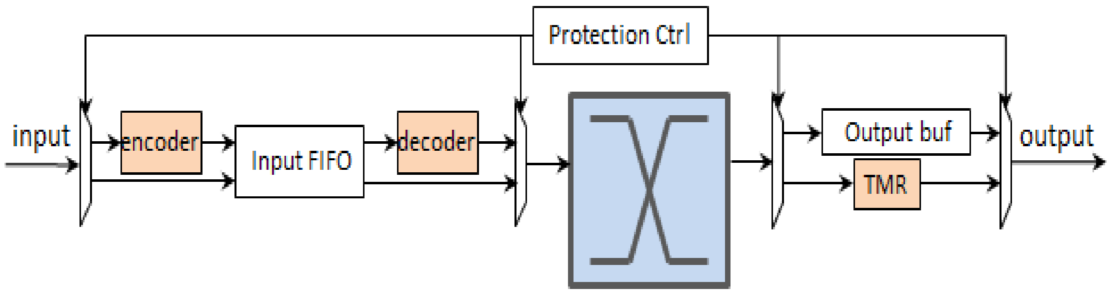


Figure 56 Input/output NoC buffer protection control logic

But even in our proposed NoC router configuration, protecting all the buffers in all NoC routers on the die can lead to excessive power dissipation and latency overhead. There is an opportunity to further reduce power and latency by intelligently determining an appropriate subset of buffers for protection against SEUs in the NoC routers. To this end, we require a dynamically selective protection methodology, to best match the time-varying behavior of an application, and optimally reduce the cost of enabling fault tolerance in NoCs. Our problem statement is summarized below:

Given: A set of applications (ACGs) with their unique bandwidth, latency and reliability constraints, a pre-defined island of cores on the die for each application, platform and technology constraints.

Find: a mapping of application cores to their pre-defined islands on the die; and an adaptive mechanism to enable energy-efficient fault tolerance in NoC routers such that all application, platform, and technology constraints are satisfied and the following problem objective is optimized:

Problem Objective: Minimize ($E_{network}/R_{network}$)

In the problem objective, $E_{network}$ is the total NoC energy consumption, which includes the energy overhead of implementing HECC and TMR in routers. $R_{network}$ is the reliability of the NoC fabric. This reliability model is discussed in the next section. By minimizing the ratio ($E_{network}/R_{network}$), we trade-off energy with reliability while simultaneously minimizing energy and maximizing reliability.

5.2.3 Reliability and power modeling

In this section, we discuss how reliability and power are modeled for the NoC architecture considered.

5.2.3.1 Reliability model

There are several scenarios that can lead to the presence of unACE bits in a NoC buffer, as discussed in section 5.1.3 which used the NVF concept as part of a reliability-aware NoC synthesis framework. However, unlike in section 5.1 where NVF was estimated at design time, in this section we attempt to estimate NVF at run time. Data masking scenarios that require comprehensive analysis (such as "read masking" and "write-after-write masking" in section 5.1.3) are impossible to detect at runtime, nevertheless two scenarios contributing to unACE bits can be detected at runtime: *(1) Idle time:* this is the most basic scenario - when there are no flits or data saved inside a buffer, we consider the buffer to be in an idle state. In this case, all the bits within the idle buffer are considered as unACE; *(2) Unused flit bits:* The flit entry in each input FIFO and output buffer of a NoC router has the same width. This assumption simplifies NoC router design and pipelined router operation with flits, which is why almost all NoC architectures adhere to it in their implementations. But as a result, there can be unused bits within a flit. Table 6 lists the unused bits (marked with an 'x') for the header and data flit types in our NoC architecture. The unused bits for a flit type are considered as unACE bits.

Table 6 unACE bits across flit types (x indicates unACE bits)

	Is_tail	Pkt_id [5:0]	Flit_id [2:0]	Src_id [4:0]	Dest_id [4:0]	Data [63:0]
Header	√	√	√	√	√	x
Data	√	x	x	x	x	√

Once the reliability of individual NoC routers has been calculated by estimating the total unACE bits, the reliability of the entire NoC fabric ($R_{network}$) can be obtained by taking the product of the reliability R_{router_i} of all N_{router} routers in the NoC fabric.

$$R_{network} = \prod_{0<i<N_{router}} R_{router_i}$$

The definition of NoC reliability in above equation is different from section 5.1 where each buffer is assigned a separate value of reliability, and the total network reliability is calculated as the product of all the buffer reliability values within the NoC. The drawback of this model from section 5.1.3 is that it gives all buffers the same importance in the final reliability value and does not consider the fact that different NoC buffers can have different sizes. For example, section 5.1.3 gives equal importance to input and output buffers when calculating network reliability. However, as the input buffer in our NoC router architecture is larger than the output buffer (input buffer has storage for multiple flits, and output port has storage only for a single flit), the network reliability value should be impacted to a greater degree by the reliability of the input buffer than the reliability of the output buffer. The current reliability model takes this factor into consideration, and is thus more accurate than section 5.1.3.

5.2.3.2 Power model

To achieve a detailed power characterization of NoC routers with protection mechanisms, we use the same library as in section 5.1.4.

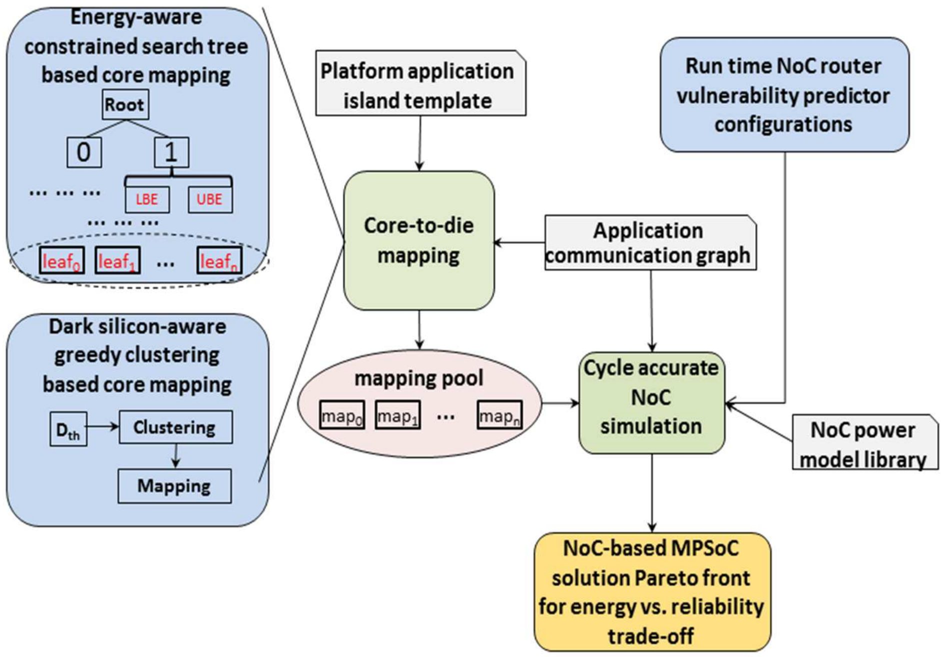

Energy-aware constrained search tree based core mapping
Root
0
1
LBE
UBE
$leaf_0$
$leaf_1$
$leaf_n$
Dark silicon-aware greedy clustering based core mapping
D_{th}
Clustering
Mapping
Platform application island template
Core-to-die mapping
Application communication graph
Run time NoC router vulnerability predictor configurations
mapping pool
map_0
map_1
map_n
Cycle accurate NoC simulation
NoC power model library
NoC-based MPSoC solution Pareto front for energy vs. reliability trade-off

Figure 57 Overview of HEFT synthesis framework

5.2.4 HEFT synthesis framework

In this section, we describe our system-level framework for enabling energy-efficient Fault-Tolerance (HEFT) in NoC-based CMPs.

Figure 57 gives an overview of the HEFT framework. We explore two different core-to-die mapping techniques to generate a mapping pool that contains mappings optimized for either tile shutdown in dark silicon die environments using a greedy clustering approach or for overall low energy consumption using a constrained search tree approach. With the generated mappings in the mapping pool, we perform cycle accurate simulation using a NoC simulator enhanced with models for measuring energy and reliability, and runtime NoC vulnerability prediction module configurations. The simulations enable an accurate estimation of energy and reliability for the generated solutions, allowing for a Pareto front of solutions that satisfy application performance constraints while trading-off energy with reliability. In the following subsections, we discuss the core-to-die mapping techniques and runtime vulnerability prediction in detail.

5.2.4.1 Core-to-die mapping

We were interested in exploring the effectiveness of two diverse strategies for core mapping towards enabling a viable trade-off between energy and reliability. These strategies are discussed next.

5.2.4.1.1 Dark silicon-aware core mapping (DSCM)

Our first core mapping strategy is designed to work in environments with dark silicon [101] constraints, where a designer may be required to shut down a subset of tiles on a die to ensure that chip-level power constraints are not violated. Shutting down a tile involves shutting down an IP core (e.g., processor, memory bank) as well as its associated NoC router. Ensuring that the shutting down of a NoC router does not end up disconnecting two communicating tiles on the die is an open problem. Our proposed mapping technique attempts to maximize the opportunities for turning off tiles to meet dark silicon power budget constraints without adversely impacting communication for the tiles that are executing workloads and communicating with each other.

```
Algorithm 3: Dark silicon-aware core mapping
Phase 1: Clustering
Input: $n_{core\_app\_i}$: total core count in application i.
        $v(comm_{ij})$, $B(comm_{ij})$, $L(comm_{ij})$: volume, bandwidth, and latency
        constraints for communication from core $C_i$ to core $C_j$, respectively
        $v(C_i) = \sum_{\forall j \text{ and } j \neq i}(v(comm_{ij}) + v(comm_{ji}))$
 1: foreach application app_i:
 2:    build an array D of $diff_{th}$ values: $diff_{th}$ represents a threshold that
       influences cluster formation; the threshold values in the array are
       unique values calculated as: $v(C_i) - v(C_j)$ $\forall i, j$ and $j \neq i$
 3:    foreach $d_{th} \in D$:
 4:       sort cores in increasing order of $v(C_i)$ and generate a core list
          $\{C_{s1}, C_{s2}, \ldots, C_{sn}\}$.
 5:       j = 0; clu = 0;
 6:       for i=1 to $n_{core\_app\_i}$:
 7:          if $v(C_{si}) - v(C_{sj}) >= diff_{th}$:
 8:             clu++, j = i;
 9:          else add core $C_{si}$ to $cluster_{clu}$;
10:       end for
Phase 2: Mapping
11:       sort clusters in decreasing order of communication volume, and sort
          cores within clusters also in decreasing order of communication
          volume to create an ordered list of cores $\{C_{m1}, C_{m2}, \ldots, C_{mn}\}$.
12:       map $C_{m1}$ randomly either to a corner or to the middle node.
13:       find an unassigned tile as the next tile i with $\min(\sum_{j \in M, i \in U}(md_{ij}))$
          that is connected with at least one of the mapped cores
14:       map the next core in the ordered list of cores to the chosen tile.
15:       repeat step 13 and 14 until all the cores are mapped.
16:       add the generated mapping solution into mapping pool $MP_{app\_i}$. if it
          satisfies $B(comm_{ij})$ and $L(comm_{ij})$ $\forall i, j$
17:    end foreach
18: end foreach
19: calculate the energy cost for different combinations of application mappings from each $MP_{app\_i}$.
20: choose $\Psi$ mappings with lowest energy as the final MP.
Output: a pool of core to tile mapping solutions
```

Algorithm 3 shows the steps of our dark silicon-aware core mapping technique. We perform mapping separately for different application islands (step 1). For each application, we generate a separate mapping pool (step 16). In the end we greedily combine different mappings from the mapping pools of different applications together into full mappings and select Ψ lowest energy mappings (step 20). These mappings are eventually explored in more detail with cycle-accurate simulations to generate accurate reliability (as well as energy) estimates which are used to create a Pareto set of solutions that trade-off energy with reliability.

For the core mapping within each application, there are two phases. The first phase involves grouping cores with similar communication volume profiles into the same cluster. To form clusters, a threshold diff_{th} is utilized, which represents the maximum communication volume difference within a cluster. We first build a threshold array D that contains all the differences of communication volume between any two cores in the application (step 2). We then use each element in D as a threshold value for each mapping loop (steps 3-17). In steps 4-10, starting from the core with the lowest communication volume with the other cores, we group cores into clusters based on diff_{th}.

The second phase involves mapping the clusters to tiles on the die. We sort the generated clusters from phase 1 in decreasing order of total communication volume of the cores inside the cluster, and then sort the cores within the clusters also in decreasing order of total communication volume of the cores to create an ordered list of cores (Step 11). Clusters with higher communication volumes and cores within them are mapped first. We randomly map the first core to a corner node or middle node of the mesh (step 12). In step 13, M represents the occupied tile set, U represents the unoccupied tile set and mdij is the Manhattan distance from an unassigned tile i to tile j. The cores are then iteratively mapped to the tiles (steps 13-15) and the full mapping solution for the application is added into its mapping pool of solutions as long as bandwidth and latency constraints are not violated, for an XY routing scheme (step 16).

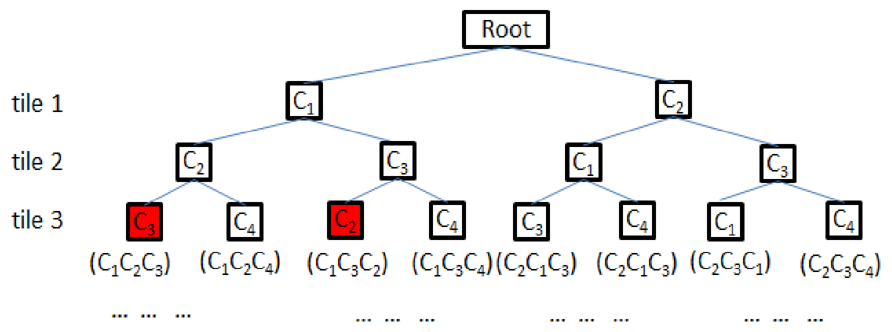


Figure 58 Constrained search tree core mapping

Thus in this manner, by clustering and mapping cores with similar communication volumes together, the routers connected to the cores with less communication volume can have more opportunities to be turned off in scenarios with stringent dark silicon power budgets, as long as there is no data transfer through them.

5.2.4.1.2 Energy-aware core mapping (EACM)

In this mapping technique, our emphasis is on discovering mappings that minimize communication energy, without any optimizations for tile shutdown scenarios. By default there can be n! different ways to map n cores on n tiles. As an efficient way to represent different mappings, we build a tree structure to save different mapping solutions. In this tree, each node contains a portion of the mapping solution. By traversing the tree from the root to a leaf node, and combing the information from intermediate nodes, a unique and complete mapping solution can be obtained. Figure 58 shows an example of this tree structure. The root represents the starting point with no cores mapped. At the first level of the tree, each node represents a partial solution with one core mapped to a tile on the mesh. Subsequent levels each add one more core to the set of mapped cores and a leaf node level represents the last remaining core being mapped to the last remaining tile on the die.

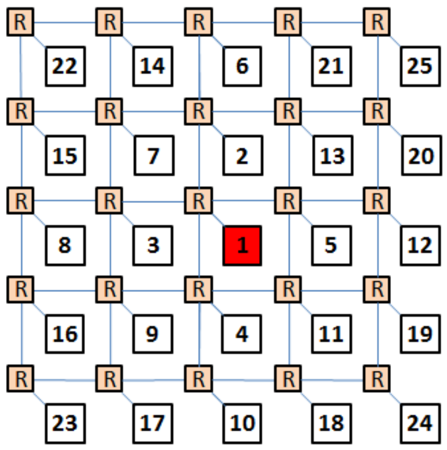


Figure 59 Tile mapping pattern

An exhaustive build of our search tree to represent all possible mapping solutions is prohibitive as it leads to solution space explosion. To limit our search, we integrate two constraining techniques as part of a constrained search tree approach. First, we limit the number of children nodes to a user defined parameter k for each node. The k different cores selected for mapping at a level have the largest value for $\sum_{j \in M}(\mathrm{v}(c_{ij}) + \mathrm{v}(c_{ji}))$. Here M is the mapped core set and i is the unmapped core under consideration. The goal is to place highly communicating closer to each other. The manner in which tiles are selected at each successive level in the tree is shown in Figure 59, for a 25 core 2D mesh NoC topology. This tile selection pattern ensures that successive cores being mapped are close to cores that were mapped earlier. Limiting the number of child nodes reduces the total number of mapping solutions that need to be evaluated from n! to $(\sum_{i \leq k}(i)) \times k^{n-k}$. But the number of total mapping solutions is still very large, e.g., for a 9 core system (n=9) and if k=4, the total number of different mapping solutions that must be evaluated is as high as 10240. Therefore we utilize a second technique to further intelligently constrain the search process by discarding highly energy inefficient partial solutions while still considering potentially interesting solutions.

```
Algorithm 4: Constrained search tree core mapping
Input: $n_{core\_app\_i}$: total core count in application i.
        $v(comm_{ij})$, $B(comm_{ij})$, $L(comm_{ij})$: volume, bandwidth, and latency
        constraints for communication from core $C_i$ to core $C_j$, respectively
        $v(C_i) = \sum_{\forall j \text{ and } j \neq i}(v(comm_{ij}) + v(comm_{ji}))$

 1: set $SE_{app\_i}$ threshold for application i as the highest communication energy over p random mapping trials
 2: foreach application app_i
 3:    foreach unoccupied tile:
 4:       foreach node in current level:
 5:          choose at most k cores with highest $\sum_{j\in M, i\in U}(v(comm_{ij}) + v(comm_{ji}))\ \forall i \in U$
$\sum_{j\in M, i\in U}(v(comm_{ij}) + v(comm_{ji}))\ \forall i \in U$ and sort these cores in decreasing order

 6:          foreach chosen core in sorted order:
 7:             generate a child node $n_c$: mapping the core onto the current tile
 8:             if $n_c$ is the leaf node of current application:
 9:                save mapping from root to $n_c$ to mapping pool $MP_{app\_i}$ if it
                   satisfies $B(comm_{ij})$ and $L(comm_{ij})$ $\forall i,j$
10:             else
11:                estimate the lowest energy (LE) of $n_c$.
12:                if $LE > SE_{app\_i}$:
13:                   mark this nodes as end node which means that the tree
                      will not be expanded further from this node.
14:          end foreach
15:       end foreach
16:    end foreach
17: end foreach
18: calculate the energy cost for different combinations of application
    mappings from each $MP_{app\_i}$.
19: choose $\Psi$ mappings with lowest energy cost as the final MP
Output: a pool of core to tile mapping solutions
```

Algorithm 4 describes our constrained search tree core mapping algorithm in detail. At the beginning (step 1), we set a standard energy SE_{app_i} parameter for each application i, by assigning it the highest value of communication energy for *p* (chosen to be 100) random full mapping trials. This parameter will be used to constrain high energy solutions later in the algorithm. Similar to the DSCM algorithm from the previous section, we generate mapping pools separately for different application islands (step 2) and in the end we greedily combine different mappings from the mapping pools of different applications together into full mappings and select Ψ lowest energy mappings (step 19). These mappings are eventually explored in more detail with cycle-accurate simulations to generate accurate reliability (as well as energy) estimates which are used to create a Pareto set of solutions that trade-off energy with reliability.

Starting from the root of the tree for an application, we generate k new nodes at each level of the tree (steps 4-7), with nodes being selected at each level from the set of unmapped cores that have the highest value of $\sum_{j\in M, i\in U}(v(comm_{ij}) + v(comm_{ji}))$. Here M is the set of mapped cores and U is the set of unmapped cores. For the degenerate initial case (M={null}), we select k cores with the highest values for $v(C_i)$. The tile at each level is selected based on the pattern shown in Figure 59. If a generated node is a leaf node, we add the corresponding full mapping (from root to leaf) into the application mapping pool (step 8-9) as long as no bandwidth and latency constraints are violated by the mapping. If the generated node is not a leaf node, we estimate the lowest energy (LE) that can be generated with this partial mapping information of the current node (steps 10-11). If LE is larger than SE_{app_i} for the node, it indicates that we are unlikely to find a mapping that has lower energy cost than SE_{app_i} which includes this node. Therefore we mark this node as an 'end node' and in the next level of the tree, we do not expand nodes marked as end nodes (step12-13).

The LE value for a node is calculated as follows:

$$LE = E_{static} + E_{LE(m,m)} + E_{LE(u,m)} + E_{LE(u,u)} \quad (3)$$

$$E_{LE(m,m)} = \sum_{\forall i,j\in M}(E_{dynamic_{ij}} + E_{dynamic\_ji}) \quad (4)$$

$$E_{LE(u,m)} = \sum_{\forall i\in U, j\in M}(E_{dynamic\_ij} + E_{dynamic\_ji}) \quad (5)$$

$$E_{LE(u,u)} = \sum_{\forall i,j\in U}(E_{dynamic\_ij} + E_{dynamic\_ji}) \quad (6)$$

M: *mapped core set*. U: *unmapped core set*

where E_{static} is the static energy of the communication subsystem, $E_{LE(m,m)}$ is the theoretically lowest dynamic energy attributed to communication between msapped cores, $E_{LE(u,m)}$ is the theoretically lowest dynamic energy attributed to communication between mapped and unmapped cores, and $E_{LE(u,u)}$ is the theoretically lowest dynamic energy attributed to communication between unmapped cores. $E_{dynamic\_ij}$ is the dynamic energy attributed to communication from core C_i to core C_j. The power model discussed in subsection 4.2 together with average estimates of packet delay through routers and hop counts along minimal path are used to calculate the energy values for communication flows between nodes. $E_{LE(m,m)}$ can be calculated easily as the required core mapping information is known. To estimate $E_{LE(u,m)}$, we randomly select unmapped cores, and greedily map each core i to an unoccupied tile that generates the lowest $\sum_{\forall j \in M}(E_{dynamic\_ij} + E_{dynamic\_ji})$. By adding the energy due to all of the newly added communication flows, $E_{LE(u,m)}$ can then be estimated. To estimate $E_{LE(u,u)}$, for each unmapped core C_i, we need to estimate a value for $E_{dynamic\_ij}$ where j represents the other unmapped cores. First, we map a core C_i with the highest value of $v(C_i)$ to a randomly chosen unmapped tile. We then sort other unmapped cores in decreasing order of their total communication volumes with core C_i, and then map the cores in the sorted order to tiles in increasing order of Manhattan distance to the tile to which C_i is mapped. After all the cores are mapped, $E_{LE(u,u)}$ is easily obtained.

5.2.4.2 Adaptive router protection

Application core to die mapping, discussed in the previous section, occurs at design time. In our framework, we complement the design time mapping mechanisms with a runtime technique to adapt protection in NoC routers. Our motivation for the runtime support stems from the observation that traffic between cores varies in its intensity and burstiness over time. The variation in traffic in turn creates variations in vulnerability in NoC routers. If we are able to somehow predict variations in vulnerability at runtime by observing traffic characteristics over time, it is possible to identify epochs of time during which vulnerability in one or more routers is very low. This information can then be used to turn off protection (e.g., disable HECC and TMR support) inside the low vulnerability NoC routers to save communication energy while still maintaining a relatively high on-chip communication reliability.

To achieve this objective, we propose a runtime reliability prediction manager (RPM) module within each NoC router.

Figure 60 shows the block diagram of the RPM module for a pair of input and output buffers along the +x direction. For each input and output buffer, a counter is used to monitor the total ACE bits encountered within a pre-defined time window interval $t_{interval}$. The counter width depends on the value of $t_{interval}$, e.g., larger $t_{interval}$ values require more counter bits to keep an accurate count for the greater traffic volumes encountered in the interval, compared to smaller $t_{interval}$ values for which fewer counter bits may suffice. The monitored information in the interval is used to turn on/off protection modules inside a NoC router at the beginning of the next time interval based on the history-based prediction from the RPM.

Figure 61 shows the state machine of the saturation counter based predictor for each buffer that guides the decision to protect that buffer. There can be n_p + 1 different states for each saturation counter with 1 unprotected state and n_p protected states. We chose only a single state for keeping a component unprotected to give more emphasis to reliability over energy savings. Transitions between states occur when a "1" or "0" signal is received, representing "protect" or "unprotect" decisions made by the saturation counter based predictors. At the end of each $t_{interval}$ period, all counters are reset and a new round of ACE bit monitoring is initiated for the next interval (
Figure **60**).

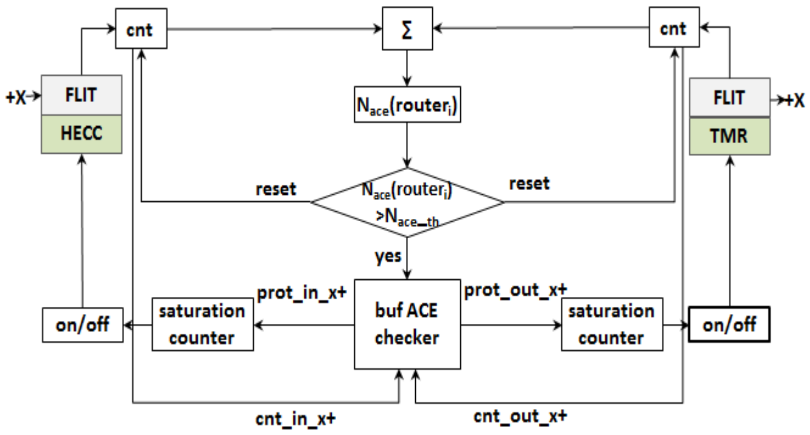


Figure 60 Block diagram of reliability prediction manager

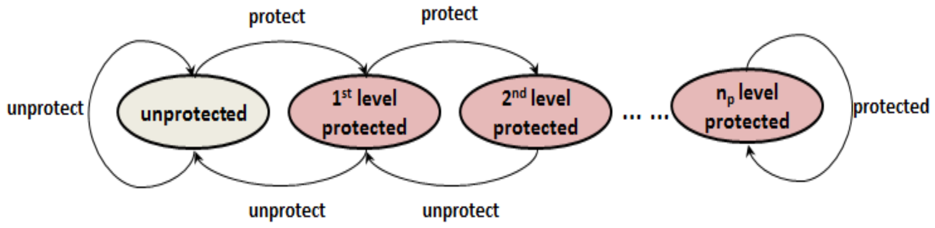


Figure 61 State machine for saturation counter based predictor

Algorithm 5: Reliability prediction manager

Input: ACE bits counter value $\mathrm{cnt}_{\mathrm{buf\_i}}$ for each input and output buffer.

```
  $N_{ace\_th\_router} = t_{interval} * \sum_{i \in ports}(n_{buf\_i}) * (1 - p_{th})$
  $N_{ace\_th\_buf\_i} = t_{interval} * n_{buf_i} * (1 - p_{th})$
1: $N_{ace}(router_i) = \sum_{i \in ports} cnt_{buf\_i}$

2: if $N_{ace}(router_i) > N_{ace\_th\_router}$:
3:     foreach buf i in the router:
4:         if $cnt_{buf\_i} >= N_{ace\_th\_buf\_i}$:
5:             $sa\_in_i = 1$
6:         else:
7:             $sa\_in_i = 0$
8:         endif
9:     end foreach
10:endif
```

Output: protection decision for each input and output buffer.

Algorithm 5 presents the details of how our runtime RPM is implemented inside a NoC router. The RPM module is invoked at the end of every $t_{interval}$ cycles, which is a user defined interval. A router ACE bits threshold value $N_{ace\_th\_router}$ is calculated and stored, based on the size of $t_{interval}$, the total number of buffer bits within the router $\sum_{i \in ports}(n_{buf\_i})$ and the user-defined reliability threshold p_{th} (a value between 0 and 1, with 1 indicating 100% reliability) for the communication subsystem. Similarly, for each buffer, we calculate and store a buffer-level ACE bit threshold value. Both of these ACE bit threshold values typically only need to be computed once at startup (or every time the size of $t_{interval}$ is varied dynamically; but this is beyond the scope of our research).

At the end of $t_{interval}$ cycles, the RPM module checks the number of ACE bits encountered over the interval by adding together values from ACE bit counters for each buffer (step 1). The sum $N_{ace}(router_i)$ is compared to $N_{ace\_th\_router}$. If $N_{ace}(router_i)$ is larger than $N_{ace\_th\_router}$, we then proceed to check each buffer's ACE bit counter cnt_{buf_i} (steps 2-3). If the ACE bit count cnt_{buf_i} for a buffer is larger than the ACE bit threshold for this buffer, we send a "protect" signal to the buffer's saturation counter, otherwise, an "unprotect" signal is sent to the saturation counter (steps 4-7). Based on the resulting new state of each saturation counter, we decide whether or not to protect the corresponding buffer (Figure 61).

5.2.5 Experimental results

5.2.5.1 Experimental setup

We modified NoC RTL models from the Netmaker library [94] with HECC and TMR protection mechanisms and RPM module support. Synopsys VCS was used to run simulations with the NoC RTL models, and with the generated signal traces, we performed logic synthesis using Synopsys Design Compiler, and gate level power analysis using Synopsys Primetime [102] to produce an activity based power library for NoC router sub-components. This power data was used to calculate communication energy, once a core mapping was established. We used a modified Gem5 full system simulator with applications from the Splash-2 benchmark suite [90] to generate instruction traces. Gem5 was configured with ALPHA processors, each with 64kB

L1 instruction and data cache, and 256K L2 caches. We considered 50 million “read” and “write” transactions out of instruction traces that varied from 1-10 billion instructions depending on the benchmark. The traces were used to enable a trace-driven simulation with a modified version of the Nirgam [49] SystemC-based cycle-accurate NoC simulator (with added support for dynamic RPM-based reliability and energy management). The trace-driven simulations allowed us to determine accurate communication energy and reliability data for our proposed HEFT framework. We performed various studies to explore different configurations of our framework and also performed comparisons with relevant prior work.

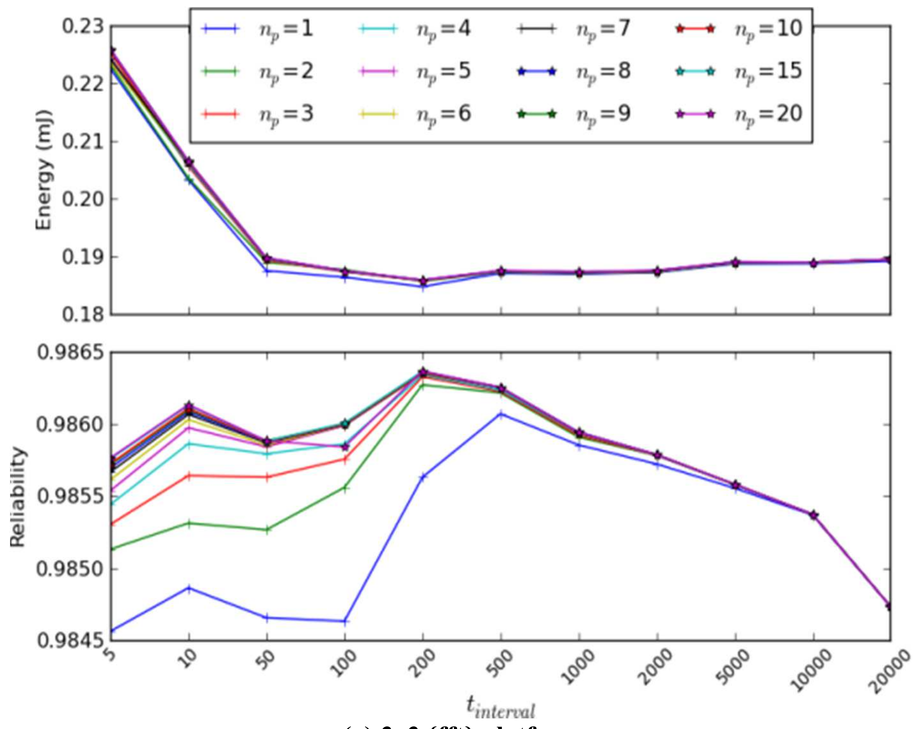
From [100], applications can in general be categorized as either compute intensive or memory intensive, based on whether an application spends more time computing than communicating with memory (compute intensive) or spends more time communicating with memory than computing (memory intensive). Thus compute intensive applications generate less communication in the network while memory intensive generate more communication traffic in the network, for the same simulation time. Table 7 shows a classification of a subset of Splash-2 benchmarks into the compute intensive and memory intensive categories. For our experimental analysis, we created single-application workloads as well as multi-application workloads that combined these applications into interesting configurations with varying compute and memory intensities. More specifically, we created the following workloads for the two platform sizes we considered: (i) 3x3 mesh: we used the compute intensive {ocean} and memory intensive {fft} single-application workloads; and (ii) 5x5 mesh: we used the {fft, barnes, raytrace} memory intensive and {ocean, fft, barnes} hybrid intensity multi-application workloads. For the 5x5 mesh workloads, we assigned 9 cores to a randomly chosen application, and 8 cores for the other applications.

Table 7 Splash-2 workloads

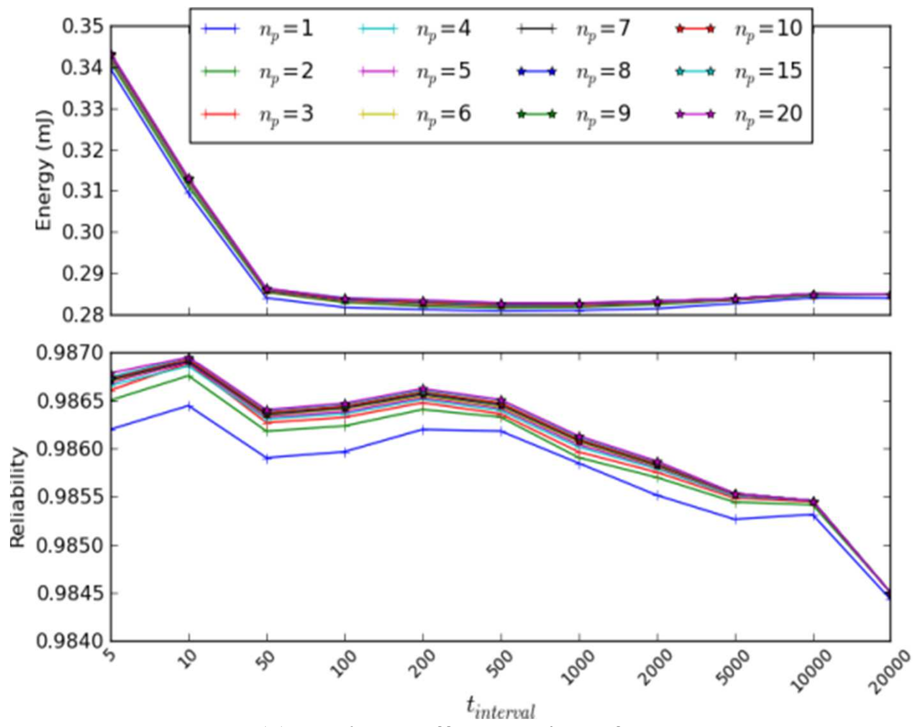
	Applications
Compute intensive	ocean, fmm, water
Memory intensive	fft, barnes, raytrace

5.2.5.2 RPM parameter sensitivity analysis

Our first set of experiments explores the parameters that impact the operation of our runtime reliability control manager (RPM) module. There are two main design parameters in the RPM that can have an impact on the reliability and energy results: the time interval length during which vulnerability is monitored ($t_{interval}$) and the number of states in the saturating counter-based predictor (n_p). Figure 62 shows energy and reliability results when varying these two parameters for the EACM mapping solution with the lowest value for ($E_{network}/R_{network}$). The parameter n_p is varied across a range from 1 to 20, and $t_{interval}$ values are varied across a range from 10 to 20000 cycles. Results are presented for two platforms: a 3x3 (9 core) platform with the {fft} workload and a larger 5x5 (25 core) platform running the {ocean, fft, barnes} multi-application workload.



(a) 3x3 {fft} platform



(b) 5x5 {ocean, fft, barnes} platform

Figure 62 RPM parameter sensitivity analysis for solutions from the EACM mapping approach

It can be observed from Figure 62 that when $t_{interval}$ is small, the overhead of frequently tabulating overall router vulnerability and more frequent decision making leads to slowdown and an increase in energy consumption, despite better tracking of changing traffic characteristics. As the $t_{interval}$ value increases, the overhead of RPM operation becomes more manageable, and the predictor allows opportunities for shutdown of HECC and TMR modules, resulting in a reduction in energy consumption. For very high values of $t_{interval}$, the predictor is not able to find opportunities to shutdown protection components as frequently, and therefore the energy consumption becomes larger due to inefficient runtime management of HECC and TMR modules. From a reliability perspective, high values of $t_{interval}$ end up prolonging the effect of mispredictions, which results in a reduction in reliability. For low to medium values of $t_{interval}$ the reliability is typically higher, with some variations in trends between the small and large platform sizes. We found that values of $t_{interval}$ between 200 to 500 cycles provide the best trade-off between prediction overhead and tracking effectiveness.

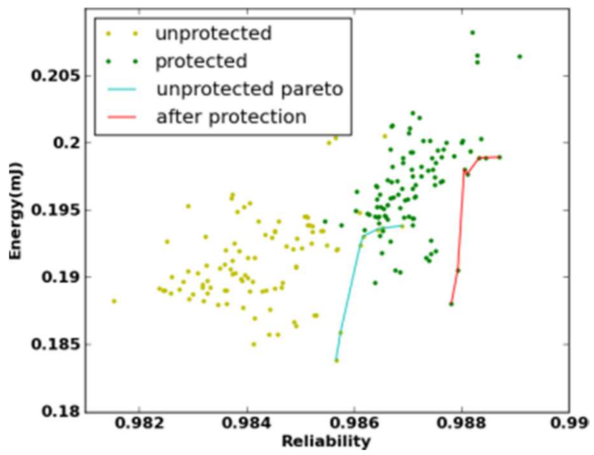
Figure 62 also shows the impact of varying n_p values. In general, the impact of a change in n_p values on energy and reliability is less pronounced than for the $t_{interval}$ parameter. For very small values of n_p (1-2) both reliability and energy are lower, due to a greater bias towards energy savings over reliability. As the value of n_p increases (3-5), reliability improves at a slight increase in energy consumption, due to the inherent bias of the predictor shifting form energy to reliability. For higher values of n_p, there is minimal impact on both reliability and energy. We found that n_p values between 2 to 4 provided a reasonable tradeoff between energy and reliability.

Based on these results, we configured our RPM module with $t_{interval} = 300$ and $n_p = 3$. This is the RPM configuration we use when evaluating our HEFT framework in the following sections.

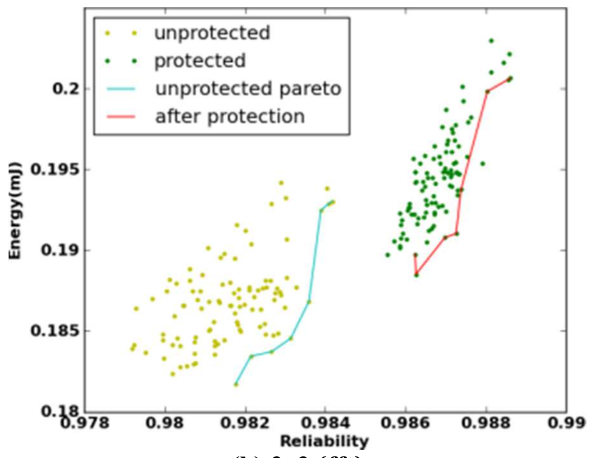
5.2.5.3 HEFT framework pareto front exploration

In this section, we explore the results generated by the HEFT framework in more detail. We were interested in understanding the value of enabling protection and RPM module based runtime management in NoC routers. Therefore we compared two scenarios: solutions generated by a core-to-die mapping framework that does not consider protection in NoC routers (i.e., a reliability un-aware synthesis framework), and solutions generated by a core-to-die mapping framework that does consider protection in NoC routers (i.e., a reliability-aware synthesis framework). For the first scenario, we only used the EACM mapping technique to generate a pool of 100 solutions, whereas for the second scenario, we additionally enabled protection and RPM-based runtime management support for the set of 100 solutions generated by EACM.

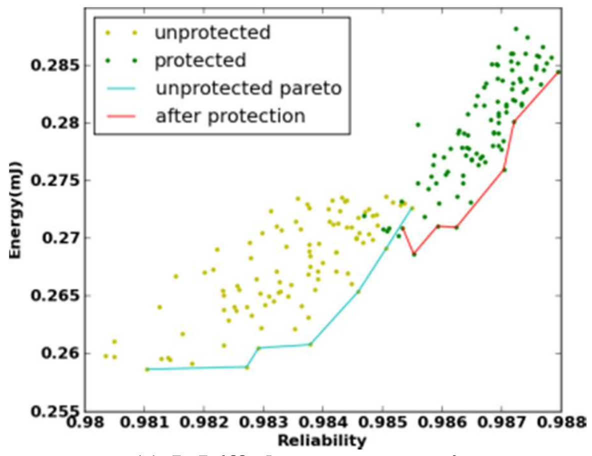
Figure 63 shows the energy and reliability results for this comparison study, for four different platform and workload combinations. The yellow dots represent the EACM-generated mapping solutions, with the blue line depicting the Pareto curve for these generated solutions. The green dots represent these same solutions after enabling protection and RPM-based runtime management, with the red line now depicting the solutions that were originally on the blue line (note: the red line is not a Pareto curve; it is simply the set of solutions that were on a Pareto front, after enabling protection support). From Figure 63, it can be clearly observed that enabling protection support boosts the overall reliability of the solution set, shifting it from the lower left quadrant up towards the upper right quadrant. There is certainly also an increase in overall energy consumption for the solutions that involve protection, which is an inevitable side-effect of increasing reliability. In general however, there is a diverse population of solutions that span a more interesting section of the design space with higher reliability guarantees, while also offering the designer flexibility to select solutions with lower energy, if lower reliability is acceptable.



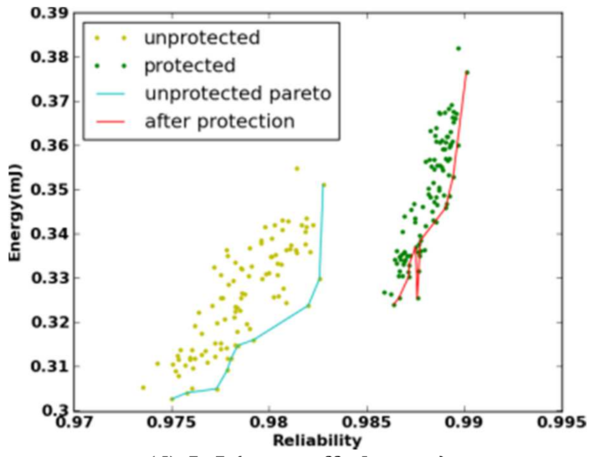
(a) 3x3 {ocean}



(b) 3x3 {fft}



(c) 5x5 {fft, barnes, raytrace}



(d) 5x5 {ocean, fft, barnes}

Figure 63 Pareto front with and without protection by HEFT

5.2.5.4 Comparison with prior work

In this section, we compare the effectiveness of our proposed HEFT framework with prior work in the area of reliability-aware NoC-based MPSoC design and synthesis. Table 8 shows the different synthesis frameworks that we evaluate and compare. In general, we compare various configurations derived from our HEFT framework with two prior works: RESYN and "xAVFs" [77]. "xAVF" [77] proposes runtime ECC protection management for all NoC router buffers, based solely on their utilization. Three techniques are proposed in that work: SAVF, S-MAVF, D-MAVF. SAVF statically sets the same utilization threshold for all routers in the NoC, such that if buffer utilization is higher than this threshold in a router, then ECC is enabled for all the buffers in that router. S-MAVF statically sets different utilization thresholds for routers in the NoC, based on design-time profiling, whereas D-MAVF dynamically changes the utilization thresholds for each router, based on runtime profiling. As the xAVF techniques do not discuss core mapping, for a fair comparison with HEFT, we employed the EACM mapping technique when obtaining results for each of these three NoC reliability management approaches.

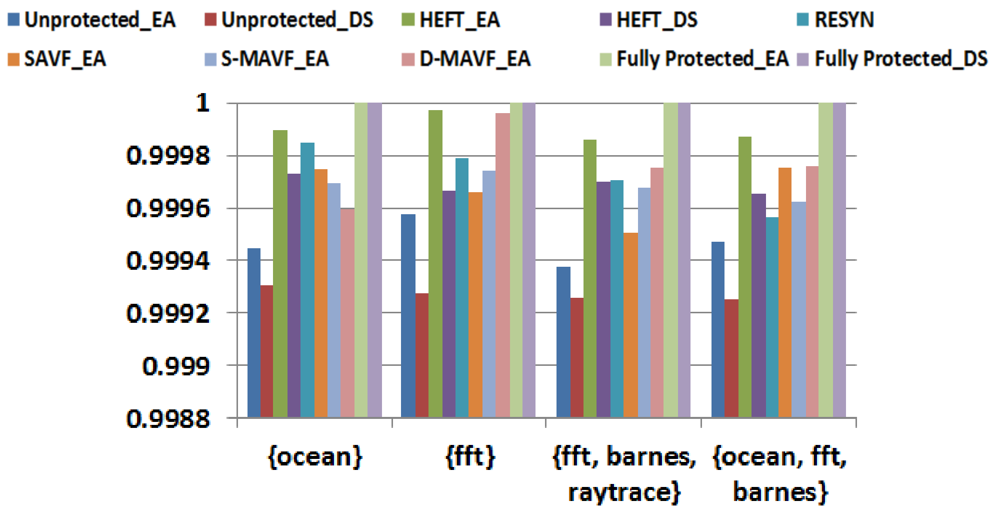
Table 8 Synthesis framework

	Mapping	**Reliability configuration**
Unprotected_EA	EACM	None
Unprotected_DS	DSCM	None
HEFT_EA	EACM	Dynamic protection
HEFT_DS	DSCM	Dynamic protection
RESYN	Genetic algorithm	GA static protection
SAVF_EA	EACM	SAVF
S-MAVF_EA	EACM	S-MAVF
D-MAVF_EA	EACM	D-MAVF
Fully Protected_EA	EACM	Full protection
Fully Protected_DS	DSCM	Full protection

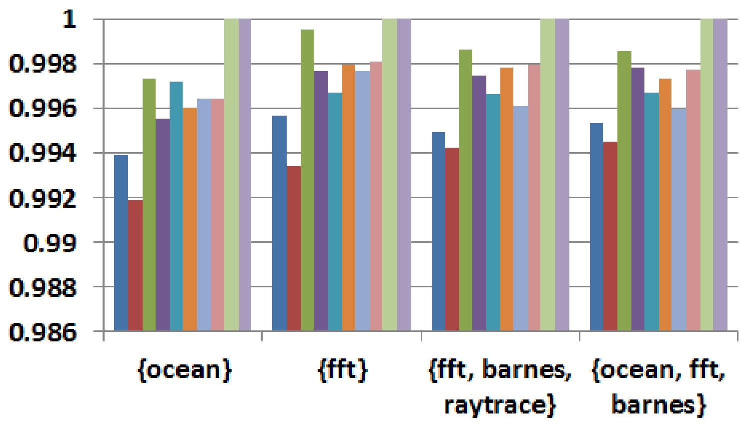
We compared these prior works with the two variants of our HEFT framework: HEFT_EA which uses the EACM mapping technique, and HEFT_DS, which uses the DSCM mapping technique. In addition, we also obtained results for Unprotected_EA and Unprotected_DS which represent a subset of our HEFT framework that use only the mapping techniques, without any

NoC router protection or RPM-based management. Finally, we also obtained results for Fully_Protected_EA and Fully_Protected_DS which are subsets of our HEFT framework that use the mapping techniques but fully protect the NoC (i.e., do not employ RPM-based runtime management).

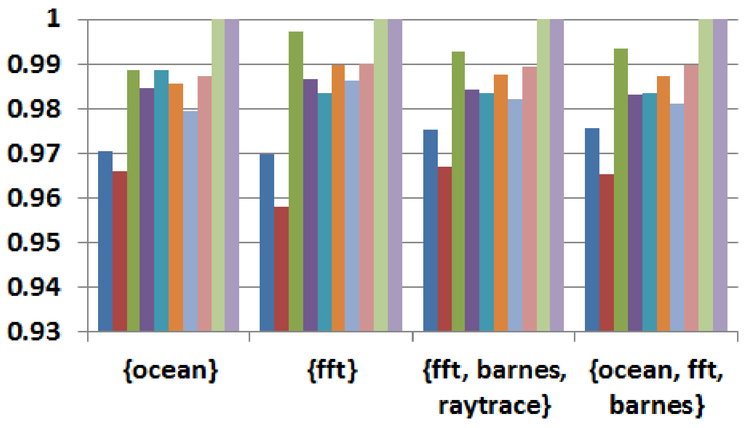
Figure 64 shows the successful packet arrival rate (along y-axes) of the generated solutions for the various frameworks listed in Table 8, for four different combinations of platforms and workloads (along x-axes), across fault rates ranging from 0.1% to 20% (Figure 64(a)-(d)). A fault rate of 20% implies that every 5 cycles a bit fault is inserted into a randomly chosen NoC router buffer. Each bar in the figures represents results averaged over ten different trials with randomized fault insertion at the appropriate fault rate. From the results, it can be observed that, not surprisingly, protecting all buffers (Fully_Protected_EA, Fully_Protected_DS) results in the highest successful packet arrival rate. However, as will be shown later, these techniques have a high energy cost that makes them less attractive than other approaches. At the other end of the spectrum, ignoring protection for NoC routers (Unprotected_EA, Unprotected_DS) results in significantly lower reliability, which becomes more pronounced as fault rates increase.



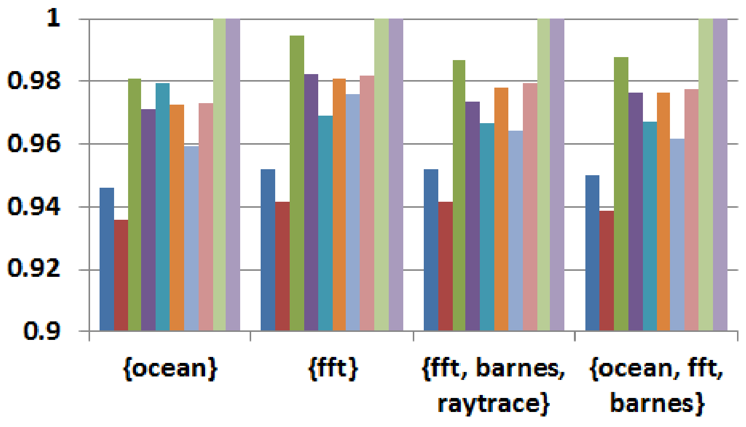
(a) Fault rate = 0.1%



(b) Fault rate = 1%



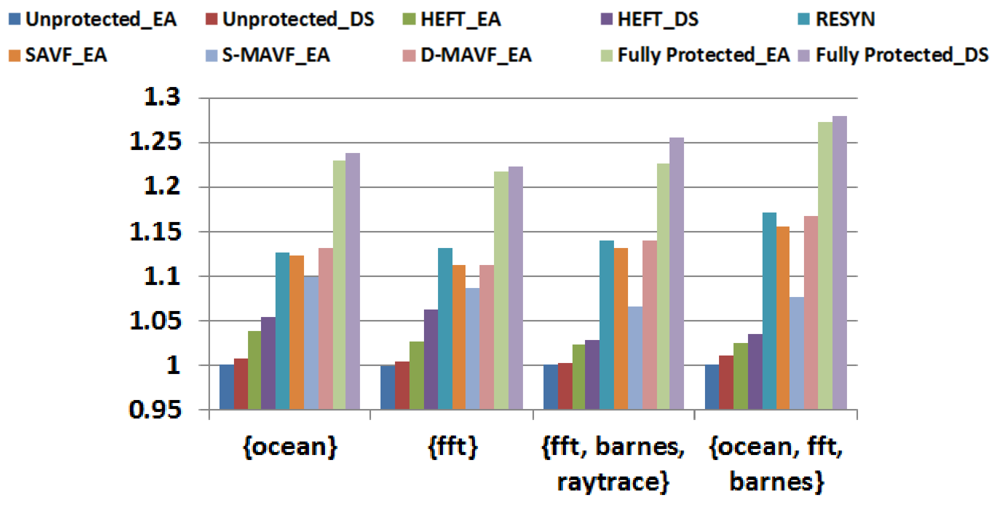
(c) Fault rate = 10%



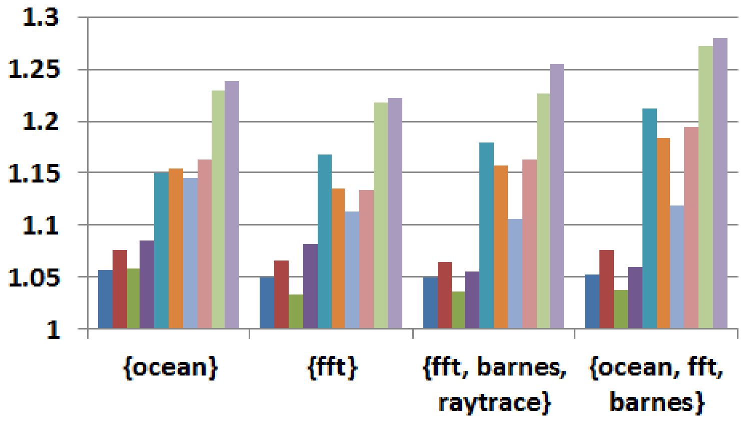
(d) Fault rate = 20%
Figure 64 Successful packet arrival rate comparison

Among the rest of the techniques, HEFT_EA can be observed to have the highest successful packet arrival rate. It turns out that optimizing for dark silicon-based shutdown with HEFT_DS is not very conducive to providing more opportunities for significantly reducing protection strengths in a subset of NoC routers. In fact, the clustering based approach used in the core-to-tile mapping in HEFT_DS detrimentally impacts traffic characteristics, resulting in reduced prediction accuracy with RPM, which in turn reduces reliability and successful packet arrival rate. Compared to RESYN, HEFT_EA can better adapt protection mechanisms dynamically to congestion and variation in traffic over time, thus avoiding scenarios where transient traffic spikes occur in NoC buffers that are left unprotected by RESYN (with this decision being made statically at design time), and cause a reduction in successful packet arrival rates in RESYN. HEFT_EA also improves upon the runtime protection mechanisms employed by SAVF, S-MAVF, and D-MAVF. The "xAVF" techniques suffer from many scenarios where either the static profiling based thresholds are too rigid (SAVF, S-MAVF) or where the dynamic prediction mechanism is unable to accurately predict buffer occupancy, compared to what the RPM is able to accomplish in our HEFT framework.

Figure 65 shows results for the energy/reliability metric for the various frameworks considered, across the four platform and workload combinations. Results are shown for two fault rates: a low 0.1% fault rate and a higher 20% fault rate. All energy values were normalized to the energy of the fully unprotected case and thus the energy values range from 0 to 1, similar to the range for reliability which is also from 0 to 1. Even though the fully protected approaches have the highest reliability, when considering energy costs these approaches become a lot less attractive than other approaches. For low fault rates, leaving the NoC unprotected seems to provide the best balance between energy and reliability. HEFT_EA has the lowest cost among all the other approaches. As fault rates become higher, it can be observed that HEFT_EA provides the lowest cost and the best balance between energy and reliability compared to not just prior work but also the unprotected approaches that suffer from a steep drop in reliability. For the higher fault rates, HEFT_EA provides 8-20% improvements over RESYN and 8-10% improvement over xAVF [77] approaches. Thus, HEFT_EA represents a scalable and effective system-level solution for balancing energy and reliability in emerging NoC-based CMPs. Our proposed framework possesses low overheads when fault rates are low and at high fault rates provides greater reliability while balancing reliability costs with energy consumption overheads.



(a) Fault rate = 0.1%



(b) Fault rate = 20%

Figure 65 Normalized energy/reliability comparison

5.2.6 Summary

In this section, we proposed a system-level framework called HEFT to trade-off energy consumption and fault-tolerance in the NoC fabric. Our hybrid framework tackled the challenge of enabling energy-efficient resilience in NoCs in two phases: at design time and at runtime. At design time, we implemented algorithms to guide the robust mapping of cores on to a die while satisfying application bandwidth and latency constraints. At runtime we devised a prediction technique to monitor and detect changes in fault susceptibility of NoC components, to intelligently balance energy consumption and reliability. Experimental results show that HEFT improves energy/reliability ratio of synthesized solutions by 8-20%, while meeting application performance goals, when compared to prior work on reliable system-level NoC design.

5.3 Chapter summary

In this chapter, we proposed two system-level frameworks, RESYN and HEFT, to trade-off energy consumption and fault-tolerance in the NoC fabric by exploring different core to tile mappings, routing algorithms, and fault-tolerant NoC protection configuration strategies. As a design-time synthesis framework, RESYN can reduce energy costs by 14.5% on average compared to a fully protected NoC, while still maintaining more than a 90% fault tolerance. By taking advantage of run time traffic congestion characteristics, HEFT can improve energy/reliability ratio of synthesized solutions by 8-20% compared with existing reliability aware synthesis frameworks. Given the increasing importance of reliability in the deep nanometer era for CMPs, our work in this chapter can guide the reduction of energy overheads associated with reliable NoC design.

CHAPTER 6
SUMMARY AND FUTURE WORK DIRECTIONS

6.1 Research summary

In this thesis, we addressed reliability challenges facing conventional electrical NoCs as CMOS technology aggressively scales into the ultra-deep sub-micron (UDSM) regime. We started by exploring different fault-tolerant routing algorithms to take advantage of the inherent redundancy available in NoC fabrics. To quantify reliability so that we can make decisions about protecting NoC routers and network interfaces, we extend the concept of architectural vulnerability factor (AVF) from the microprocessor domain and propose a network vulnerability factor (NVF) to characterize the susceptibility of NoC components. As shown in Figure 66, by utilizing our research on fault-tolerant routing algorithms and reliability modeling, we solve the NoC reliability problem at the system level by proposing a synthesis framework that contains core to tile mapping, reliability modeling, routing, and energy-efficient resilience design for NoC communication subsystems.

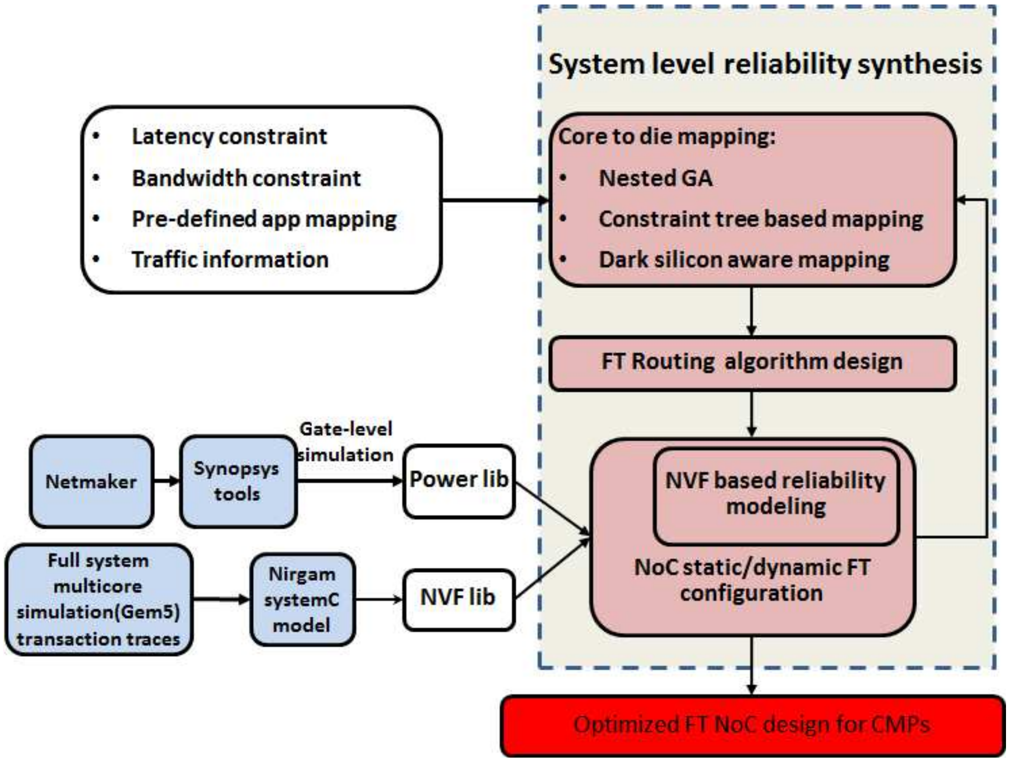

System level reliability synthesis
Latency constraint
Bandwidth constraint
Pre-defined app mapping
Traffic information
Core to die mapping:
Nested GA
Constraint tree based mapping
Dark silicon aware mapping
FT Routing algorithm design
Netmaker
Synopsys tools
Gate-level simulation
Power lib
NVF based reliability modeling
NoC static/dynamic FT configuration
Full system multicore simulation(Gem5) transaction traces
Nirgam systemC model
NVF lib
Optimized FT NoC design for CMPs

Figure 66 Research summary

Our first research contribution is in the area of fault-tolerant NoC routing algorithm design. We proposed OE+IOE which is a novel fault-tolerant routing scheme for 2D mesh NoCs that can adapt to design time and runtime permanent link faults, as well as potential intermittent faults in NoC communication architectures. Our scheme uses replication opportunistically based on fault rate, and combines the odd-even (OE) and inverted odd-even (IOE) turn models to achieve deadlock free packet traversal. Experimental results show that our proposed OE+IOE scheme can provide better fault tolerance (i.e., higher successful packet arrival rates) than traditional fault-tolerant routing schemes such as N-random walk and turn model based schemes. By extending OE+IOE, we proposed a neighbor aware, threshold-based, and replication-based fault-tolerant routing scheme (NARCO) which uses parameterizable region based neighbor awareness in routers. NARCO can achieve even better packet arrival rate than OE+IOE, while still enabling a trade-off between communication reliability and energy overhead.

Next we proposed the NS-FTR routing scheme for NoCs that combines the North-last and South-last turn models to create a robust hybrid NoC routing scheme. Based on our analysis, NS-FTR can outperform all other hybrid replication based turn model routing schemes in terms of energy overhead, performance, and fault-tolerant ability. As three-dimensional integrated circuits (3D-ICs) offer a significant opportunity to enhance the performance of emerging chip multiprocessors (CMPs) using high density stacked device integration and shorter through silicon via (TSV), we extended NS-FTR for 3D NoCs and proposed 4NP-First which is a fault-tolerant routing scheme that combines the 4N-First and 4P-First turn models together. 4NP-First is shown to have a low implementation overhead and adapts to design time and runtime faults better than existing turn model, stochastic random walk, and hybrid dual virtual channel based routing schemes.

Our second research contribution is to quantify and model reliability in NoCs so that we can enable intelligent protection configurations for NoC fabrics based on the given reliability requirements. We extended the concept of architectural vulnerability factor (AVF) from the microprocessor domain and proposed a network vulnerability factor (NVF) metric to characterize the susceptibility of NoC components such as the Network Interface (NI) to transient faults. Our studies reveal that different NI buffers behave quite differently with transient faults and each buffer can have different levels of inherent fault-tolerance. Our analysis also considered the impact of thermal hotspot mitigation techniques such as frequency throttling on the NVF estimation.

Our third contribution is the design of two system-level frameworks RESYN and HEFT to trade-off energy consumption and fault-tolerance in NoC fabrics. RESYN employs a nested genetic algorithm (NGA) approach to guide the mapping of cores on a die, and opportunistically determine the locations to insert fault tolerance mechanisms in NoC routers to minimize energy consumption while satisfying reliability constraints. Our experimental results show that RESYN can reduce communication energy costs by 14.5% on average compared to a fully protected NoC design, while still maintaining more than a 90% fault tolerance. RESYN also enables a comprehensive trade-off between reliability and energy-efficiency for more stringent reliability constraints, generating a Pareto set of solutions which may reveal opportunities for energy savings even for high reliability configurations; e.g., for a higher 95% reliability constraint, a notable 13% energy savings can still be achieved for some applications. While RESYN is a pure design time synthesis framework, our second synthesis framework HEFT tackles the challenge of enabling energy-efficient resilience in NoCs in two phases: design time core to tile mapping and runtime router fault-tolerant configuration. At design time, we implemented algorithms to guide the robust mapping of cores on to a die while satisfying application bandwidth and latency constraints. At runtime we devised a prediction technique to monitor and detect changes in fault susceptibility of NoC components, to intelligently balance energy consumption and reliability. Experimental results show that HEFT improves energy/reliability ratio of synthesized solutions by 8-20%, while meeting application performance goals, when compared to prior work on reliable system-level NoC design.

Together, our three major contributions in the area of fault-tolerant NoC design have improved upon the state-of-the-art and significantly advanced the field of reliable chip multiprocessor design.

6.2 Future research

As CMOS technology aggressively scales into the ultra-deep sub-micron (UDSM) regime and application complexity grows rapidly, there will always be a need to consider more aggressive fault-tolerant design because of phenomena such as process variations during fabrication, hardware aging effects such as NBTI, HCI, electromigration, or issues in the hardware environment such as high temperatures, EMI effects, etc. We further envision the following future work directions:

- **NoC vulnerability analysis under process variation effects and the creation of process variation aware NoC synthesis frameworks**. As silicon-based MOSFETs are scaled down to nanoscale feature size, process parameter fluctuations (process variations) play a vital and important role in matching performance and yield analysis. Process variations cause the fluctuations of gate width, length, and gate oxide thickness, which cause threshold voltage V_T instability and cause changes in drain current I_d. As a result, critical path delay cannot be guaranteed to meet circuit timing requirements. To solve this uncertainty due to process variations, traditionally, clock frequency is set to meet the critical path delay under the worst case scenario. According to [105][106], the length of the slack time for a circuit path has an impact on its vulnerability. We have already proposed NVF to quantify the reliability of buffer units in NoC routers and NIs based on their data content and buffer utilization. It would be interesting to quantify the reliability based on the timing characteristics of control logic under the impact of the process variation, and then design a framework for reliable NoC synthesis that considers these issues.
- **Cross layer reliability aware NoC synthesis**. As in [104], to design a reliable system, optimizations in multiple layers' should be co-ordinated, such as compiler layer, OS layer, architectural layer, circuit layer, etc. By combining enhancements in different layers, a more effective trade off can be made at the system level when it comes to power consumption, performance, and reliability. Until now my research has focused on the architectural and circuit levels, it would be interesting to consider fault-tolerant strategies in other layers as

well.

- **Aging aware NoC design.** Continuous technology scaling has made aging mechanisms such as Negative Bias Temperature Instability (NBTI), hot carrier injection (HCI) and electromigration key concerns in NoC design. Reliability modeling for different aging effects is necessary so that we don't end up with circuit over-protection and cause excessive overhead. Different modules in NoCs have different vulnerabilities to different aging phenomena. Based on component-specific aging effects we can intelligently enable different levels of protection across the NoC. Also fault tolerant routing schemes can be designed to have prioritized path selection to alleviate NBTI and HCI effects.
- **Reliability-aware photonic NoC design.** On-chip communication faces scalability issues even with NoCs. Optical communication is a promising solution to address the bandwidth and scalability issues for on-chip communication; however silicon photonic devices are sensitive to process and thermal variations [107]. Different protection strategies and fault tolerant routing schemes can be specially designed to exploit characteristics of photonic NoCs.

REFERENCES

[1] W. J. Dally and B. Towles, "Principles and Practices of Interconnection Networks", Morgan Kauffman, 2004.

[2] D. Pham, S. Asano, M. Bolliger, M. Day, H. Hofstee, C. Johns, J. Kahle, A. Kameyama, J. Keaty, Y. Masubuchi, M. Riley, D. Shippy, D. Stasiak, M. Suzuoki, M. Wang, J. Warnock, S. Weitzel, D. Wendel, T. Yamazaki and K. Yazawa, "The design and implementation of a first-generation CELL processor", Proceedings of IEEE International Solid State Circuits Conference (ISSCC), pp. 184-185, 2005.

[3] S. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, P. Iyer, A. Singh, T. Jacob, S. Jain, S. Venkataraman, Y. Hoskote and N. Borkar, "An 80-Tile 1.28 TFLOPS Network-on-Chip in 65 nm CMOS", Proceedings of IEEE International Solid State Circuits Conference (ISSCC), pp. 98-99, 2007.

[4] Corporation Tilera, "Tilera multicore processors", http://www.tilera.com/products/processors.

[5] J. Owens, W. Dally, R. Ho, D. Jayasimha, S. Keckler and L-S. Peh. , "Research challenges for on-chip interconnection networks", International Symposium on Microarchitecture (MICRO), pp. 96-108, 2007.

[6] ITRS Technology Working Groups, "International Technology Roadmap for Semiconductors (ITRS)", http://public.itrs.net. 2007.

[7] S. Rusu, H. Muljono, D. Ayers, S. Tam, W. Chen, A. Martin, S. G. Li, S. Vora, R. Varada and E. Wang, “Ivytown: A 22nm 15-Core Enterprise Xeon® Processor Family”, Proceedings of IEEE International Solid State Circuits Conference (ISSCC), pp.102-103, 2014.

[8] G. Chen, M. A. Anders, H. Kaul, S. K. Satpathy, S. K. Mathew, S. K. Hsu, A. Agarwal, R. K. Krishnamurthy, S. Borkar and V. De, “A 340mV-to-0.9V 20.2Tb/s Source-Synchronous Hybrid Packet/Circuit-Switched 16×16 Network-on-Chip in 22nm Tri-Gate CMOS”, Proceedings of IEEE International Solid State Circuits Conference (ISSCC), pp. 276- 277, 2014.

[9] S. S. Mukherjee, J. Emer and S. K. Reinhardt, “The soft error problem: An architectural perspective”, Proceeding of International Symposium on High-Performance Computer Architecture (HPCA), pp.243-247, 2005.

[10] E. Normand, “Single event upset at ground level”, IEEE Transactions on Nuclear Science, pp. 2742-2750, 1996.

[11] C. Constantinescu, “Trends and challenges in VLSI circuit reliability”, Proceedings of International Symposium on Microarchitecture (MICRO), pp. 14-19, 2003.

[12] S. Nassif, “Modeling and analysis of manufacturing variations”, Proceeding of IEEE Conference on Custom Integrated Circuits (CICC), pp. 223-228, 2001.

[13] C. Constantinescu, “Intermittent faults in VLSI circuits”, Proc. IEEE Workshop on Silicon Errors in Logic - System Effects (SELSE), pp. 1-6, 2007.

[14] S. Pasricha and N. Dutt, "On-Chip Communication Architectures", Morgan Kauffman, 2008.

[15] L. Benini and G. D. Micheli, "Networks on chips: a new SoC paradigm", IEEE Computer, pp. 70-78, 2002.

[16] W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks", Proceedings of Design Automation Conference (DAC), pp. 684-689, 2001.

[17] D. C. Pham, T. Aipperspach, D. Boerstler, M. Bolliger, R. Chaudhry and D. Cox, "Overview of the architecture, circuit design, and physical implementation of a first-generation cell processor", IEEE Journal of Solid-State Circuits, pp. 179-196, 2006.

[18] Intel Teraflops, "terascale/terascale_overview_paper.pdf", http://download.intel.com/research/platform/.

[19] Picochip PC102. http://www.picochip.com/highlights/pc102.

[20] S. Bell, B. Edwards, J. Amann, R. Conlin and K. Joyce, "TILE64 processor: A 64-core SoC with mesh interconnect", Proceedings of IEEE International Solid State Circuits Conference (ISSCC), pp.88-89, 2008.

[21] D. Bertozzi, L. Benini and G. De Micheli, "Error control schemes for on-chip communication links: the energy–reliability tradeoff", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), pp. 818-831, 2005.

[22] S. Murali, L. Benini, T. Theocharides, N. Vijaykrishnan, M. J. Irwin and G. D. Micheli, “Analysis of error recovery schemes for networks on chips”, IEEE Design & Test of Computers, pp. 434-442, 2005.

[23] S. Lin and D. J. Costello, “Error control coding: fundamentals and applications”, Englewood Cliffs, NJ: Prentice-Hall, 1983.

[24] D. Bertozzi, L. Benini and G. De Micheli, “Low power error resilient encoding for on-chip data buses”, Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE), pp. 102-109, 2002.

[25] M. Lajolo, “Bus guardians: an effective solution for online detection and correction of faults affecting system-on-chip buses”, IEEE Transactions on Very Large Scale Integration (TVLSI), pp.974-982, 2001.

[26] M. Dehyadgari, M. Nickray, A. Afzali-kusha and Z. Navabi, “Evaluation of pseudo adaptive XY routing using an object oriented model for NoC”, Proceedings of International Symposium on Microarchitecture (MICRO), pp. 13-15, 2005.

[27] H. Zhu, P. P. Pande and C. Grecu, “Performance evaluation of adaptive routing algorithms for achieving fault tolerance in NoC fabrics”, Proceedings of IEEE International Conf. on Application -specific Systems, Architectures and Processors (ASAP), pp. 42-47, 2007.

[28] T. Dumitras and R. Marculescu, “On-chip stochastic communication”, Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE), pp. 790-795, 2003.

[29] M. Pirretti, G. M. Link, R. R. Brooks, N. Vijaykrishnan, M. Kandemir and M. J. Irwin, "Fault tolerant algorithms for network-on-chip interconnect", Proceedings of IEEE Computer society Annual Symposium on VLSI (ISVLSI), pp. 46-51, 2004.

[30] Y. B. Kim and Y-B. Kim, "Fault tolerant source routing for network-on-chip", Proceedings of IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT), pp. 1550-5774, 2007.

[31] T. Schonwald, J. Zimmermann, O. Bringmann, and W. Rosenstiel, "Fully adaptive fault tolerant routing algorithm for network-on-chip architectures", Proceedings of Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD), pp. 527-534, 2007.

[32] T.Schonwald, O.Bringmann and W.Rosenstiel, "Region-based routing algorithm for network-on-chip architectures", Proceeding of Norchip, pp. 1-4, 2007.

[33] C. J. Glass and L. M. Ni, "The turn model for adaptive routing", Proceedings of International Symposium on Computer Architecture (ISCA), pp. 278- 287, 1992.

[34] C. J. Glass and L. M. Ni, "Fault-tolerant wormhole routing in meshes without virtual channels", IEEE Transactions on Parallel and Distributed Systems (TPDS), pp. 620-635, 1996.

[35] C. M. Cunningham and D. R. Avresky, "Fault-tolerant adaptive routing for two-dimensional meshes", Proceeding of International Symposium on High-Performance Computer Architecture (HPCA), pp. 122-131, Jan. 1995.

[36] G.-M. Chiu, "The odd-even turn model for adaptive routing", IEEE Transactions on Parallel and Distributed Systems (TPDS), pp. 729-738, 2000.

[37] A.Patooghy and S.G.Miremadi, "XYX: A power & performance efficient fault-tolerant routing algorithm for network on chip", Proceeding of International Conference on Parallel, Distributed and Nsetwork-based Processing, pp. 245-251, 2009.

[38] D.Fick, A.DeOrio, G.K.Chen,V.Bertacco, D.Sylvester and D. Blaauw, "A highly resilient routing algorithm for fault-tolerant NoCs", Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE), pp. 21-26, 2009.

[39] R.V. Boppana and S. Chalasani, "Fault-tolerant wormhole routing algorithms for mesh networks", IEEE Transactions on Computers, pp. 848-864, 1995.

[40] J. Wu, "A fault-tolerant and deadlock-free routing protocol in 2D meshes based on odd-even turn model", IEEE Transactions on Computers, pp. 1154-1169, 2003.

[41] A. Rezazadeh, M. Fathy and A. Hassanzadeh, "If-cube3: An improved fault-tolerant routing algorithm to achieve less latency in NoCs", Proceedings of IEEE International Advance Computing Conference (IACC), pp. 278-283, 2009.

[42] S.Jovanovic, C.Tanougast, S.Weber and C.Bobda, "A new deadlock-free fault-tolerant routing algorithm for NoC interconnections", Proceedings of International Conference on Field Programmable Logic and Applications (FPLA), pp. 326-331, 2009.

[43] M. Andres, P.Maurizio and F.José, “Region-Based Routing: A Mechanism to Support Efficient Routing Algorithms in NoCs”, IEEE Transactions on VLSI, pp. 356-369, 2009.

[44] J. Hu and R. Marculescu, “Dyad - smart routing for networks-on-chip”, Proceedings of Design Automation Conference (DAC), pp. 260-263, 2004.

[45] Z. Zhang, A. Greiner and S. Taktak, “A reconfigurable routing algorithm for a fault-tolerant 2D-mesh network-on-chip”, Proceedings of Design Automation Conference (DAC), pp. 441-446, 2008.

[46] W.-C. Kwon, S. Yoo, J. Um and S.-W. Jeong, “In-network reorder buffer to improve overall NoC performance while resolving the in-order requirement problem”, Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE), pp. 1058-1063, 2009.

[47] M. Koibuchi, H. Matsutani, H. Amano and T. M. Pinkston, “A lightweight fault-tolerant mechanism for network-on-chip”, Proceedings of International Symposium on Networks-on-Chip (NOCS), pp. 13-22, 2008.

[48] SystemC initiative, http://www.systemc.org.

[49] Nirgam simulator, http://nirgam.ecs.soton.ac.uk/.

[50] A. Kahng, B. Li, L.-S. Peh and K. Samadi, “ORION 2.0: A fast and accurate NoC power and area model for early-stage design space exploration”, Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE), pp. 423-428, 2009.,

[51] K. Bernstein, P. Andry, J. Cann, P. Emma, D. Greenberg, W. Haensch, M. Ignatowski, S. Koester, J. Magerlein, R. Puri and A. Young, “Interconnects

in the Third Dimension: Design Challenges for 3D ICs”, Proceedings of Design Automation Conference (DAC), 2007, pp. 562-567.

[52] R. S. Patti, “Three-Dimensional Integrated Circuits and the Future of System-on-Chip Designs”, Proceedings of IEEE, pp. 1214- 1224, 2006.

[53] L. Benini and G. De-Micheli, “Networks on Chip: A New SoC Paradigm”, Proceedings of Computer, pp. 70-71, 2002.

[54] Tilera Corporation. “TILE64™ Processor. Product Brief”, http://www.tilera.com/products/processors, 2007.

[55] S. S. Mukherjee, C. Weaver, J. Emer, S. K. Reinhardt, and T. Austin, “A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor”, Proceedings of International Symposium on Microarchitecture (MICRO), pp. 29-40, 2003.

[56] X. Fu, T. Li, and J. Fortes, “Sim-SODA: A Unified Framework for Architectural Level Software Reliability Analysis”, Proc. Workshop Modeling, Benchmarking and Simulation, pp. 234-240, 2006.

[57] X. Fu, J. Poe, T. Li, and J. Fortes, “Characterizing Microarchitecture Soft Error Vulnerability Phase Behavior”. Proc. IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), pp.147-155, 2006.

[58] A. Biswas, C. Recchia, S. S. Mukherjee, V. Ambrose, L. Chan, A. Jaleel, A. E. Papathanasiou, M. Plaster and N. Seifert, “Explaining Cache SER Anomaly Using DUE AVF Measurement”, Proceeding of International

Symposium on High-Performance Computer Architecture (HPCA), pp 1-12, 2010.

[59] S. Pasricha, Y. Zou, D. Connors and H. J. Siegel, “OE+IOE: A Novel Turn Model Based Fault Tolerant Routing Scheme for Networks-on-Chip”, Proc. IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS), pp. 85-93, 2010.

[60] F. Refan, H. Alemzadeh, S. Safari, P. Prinetto and Z. Navabi, “Reliability in Application Specific Mesh-Based NoC Architectures”, in Proceedings of IEEE International On-Line Testing Symposium, pp. 207-212, 2008.

[61] J. Kim, D. Park, C. Nicopoulos, N. Vijaykrishnan, and C. R. Das, “Design and analysis of an NoC architecture from performance, reliability and energy perspective”, Proceedings of Symposium on Architecture for networking and communications systems (ANCS), pp. 173-182, 2005.

[62] ARM, “AMBA AXI Protocol Specification v2.0”, www.arm.com, 2010.

[63] A. Biswas, P. Racunas, J. Emer, and S. Mukherjee, “Computing Accurate AVFs Using ACE Analysis on Performance Models: A Rebuttal”, Proc. Computer Architecture Letters, pp 21-24, 2008.

[64] W. Zhang, X. Fu, T. Li and J. Fortes, “An Analysis of Microarchitecture Vulnerability to Soft Errors on Simultaneous Multithreaded Architectures”, Proceedings of IEEE International Symposium on Performance Analysis of Systems & Software (ISPASS), pp. 169-178, 2007.

[65] N. Soundararajan, A. Parashar and A. Sivasubramaniam, “Mechanisms for Bounding Vulnerabilities of Processor Structures”, International Symposium on Computer Architecture (ISCA), pp. 506-515, 2007.

[66] V. Sridharan and D. R. Kaeli, “Eliminating microarchitectural dependency from architectural vulnerability”, Proceeding of International Symposium on High-Performance Computer Architecture (HPCA), pp. 117-128, 2009.

[67] V. Sridharan and D. R. Kaeli, “Using hardware vulnerability factors to enhance AVF analysis”, International Symposium on Computer Architecture (ISCA), pp. 461-472, 2010.

[68] Q. Yu, B. Zhang, Y. L and P. Ampadu, “Error Control Integration Scheme for Reliable NoC”, Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS), pp. 3893-3896, 2010.

[69] A. P. Frantz, F. L. Kastensmidt, L. Carro and É. Cota, “Dependable Network-on-Chip Router Able to Simultaneously Tolerate Soft Errors and Crosstalk”, Proceedings of. IEEE International Test Conference (ITC), pp. 1-9, 2006.

[70] Y. Chang, C. Chiu, S. Lin and C. Liu, “On the Design and Analysis of Fault Tolerant NoC Architecture Using Spare Routers”, Proceedings of Asia and South Pacific Design Automation Conference (ASPDAC), pp. 431-436, 2011.

[71] SPLASH-2, http://kbarr.net/splash2.

[72] M5 simulator system, http://www.m5sim.org.

[73] McPat 0.8, http://www.hpl.hp.com/research/mcpat/.

[74] Hotspot 5.0, http://lava.cs.virginia.edu/HotSpot/license.htm.

[75] N.J. George and C.R. Elks, “Transient fault models and AVF estimation revisited”, Dependable Systems and Networks (DSN), pp. 477-486, 2010.

[76] T. Lehtonen, P. Liljeberg and J. Plosila, “Online reconfigurable self-timed links for fault tolerant NoC”, International Conference on VLSI Design (VLSID), pp.1-13, 2007.

[77] A. Yanamandra, S. Eachempati and N. Soundararajan, “Optimizing power and performance for reliable on-chip networks”, Proceedings of Asia and South Pacific Design Automation Conference (ASPDAC), pp.431-436, 2010.

[78] K. Constantinides, S. Plaza, J.Blome and B. Zhang, “Bullet Proof: a defect-tolerant CMP switch architecture”, Proceeding of International Symposium on High-Performance Computer Architecture (HPCA), pp .5-16, 2006.

[79] G. Ascia, V. Catania and M. Palesi, “Mapping cores on network-on-chip”, International Journal of Computational Intelligence Research Research India Publications (IJCIR), pp. 109-126, 2005.

[80] S. Murali and G. D. Micheli, “Bandwidth-constrained mapping of cores onto NoC architectures”, Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE), pp. 896-901, 2004.

[81] K. Srinivasan, K.S. Chatha and G.Konjevod, “Linear programming based techniques for synthesis of network-on-chip architectures”, IEEE Conference on Computer Design (ICCD), pp.422-429, 2004.

[82] J. C. Hu and R. Marculescu, “Exploiting the routing flexibility for energy/performance aware mapping of regular noc architectures”,

Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE), pp. 688-693, 2003.

[83] Y.F. Hu, Y. Zhu, H.Y. Chen and R. Graham, “Communication latency aware low power noc synthesis”, Proceedings of Design Automation Conference (DAC), pp. 574-579, 2006.

[84] Y.F. Hu, H.Y. Chen, Y. Zhu, A.A. Chien and C.K. Cheng, “Physical synthesis of energy efficient networks on chip through topology exploration and wire style optimization”, IEEE Conference on Computer Design (ICCD), pp.111-118, 2005.

[85] S. S. Mukherjee and C. Weaver, “Systematic methodology to compute the architectural vulnerability factors for a high performance microprocessor”, Proceedings of International Symposium on Microarchitecture (MICRO), pp. 29-40, 2003.

[86] Synopsys DC and Primetime, www.synopsys.com.

[87] T.T. Ye, L. Benini and G.D. Micheli, “Analysis of power consumption on switch fabrics in network routers”, Proceedings of Design Automation Conference (DAC), pp. 524-529, 2002.

[88] M.R. Garey and D.S. Johnson, “Computers and intractability: a guide to the theory of NP-completeness”, W. H. Freeman & Co. New York, NY, USA, 1990.

[89] Synopsys, “VCS®/VCSi™ User Guide”, 2011.

[90] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh and A. Gupta, “The SPLASH-2 programs: characterization and methodological considerations”, International Symposium on Circuits and Systems (ISCAS), pp. 24-36, 1995.

[91] G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, “The gem5 simulator”, ACM SIGARCH Computer Architecture. pp. 1-7, 2011.

[92] Y. F. Hu and H. Y. Chen, “Physical synthesis of energy-efficient networks-on-chip through topology exploration and wire style optimization”, IEEE Conference on Computer Design (ICCD), pp. 111-118, 2005.

[93] J.C. Hu and R. Marculescu, “Energy-aware mapping for tile-based NoC architectures under performance constraints”, Proceedings of Asia and South Pacific Design Automation Conference (ASPDAC), pp. 233-239, 2003.

[94] Netmaker, http://www-dyn.cl.cam.ac.uk/~rdm34/wiki/.

[95] J. Hu and R. Marculescu, “Energy- and performance-aware mapping for regular NoC architectures”, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), pp. 551-562, 2005.

[96] C. Ababei, H.S. Kia andO.P. Yadav, “Energy and reliability oriented mapping for regular networks-on-chip”, Proceedings of International Symposium on Networks-on-Chip (NOCS), pp. 121-128, 2011.

[97] Y. Zou, Y. Xiang and S. Pasricha, “Characterizing Vulnerability of Network Interfaces in Embedded Chip Multiprocessors”, IEEE Embedded System Letters, pp. 41- 44, 2012.

[98] K. Srinivasan and K. S. Chatha, “A Technique for Low Energy Mapping and Routing in Network-on-Chip Architectures”, Proceedings of the 2005 International Symposium on Low Power Electronics and Design (ISLPED), pp. 387-392, 2005.

[99] K. Aisoposyx, C. O. Chenx and Li-Shiuan Peh, “Enabling System-Level Modeling of Variation-Induced Faults in Networks-on-Chips”, Proceedings of Design Automation Conference (DAC), pp. 930-935, 2011.

[100] T. Pimpalkhute and S. Pasricha, “NoC Scheduling for Improved Application-Aware and Memory-Aware Transfers in Multi-Core Systems”, IEEE International Conference on VLSI Design (VLSID), Jan. 2014.

[101] M. B. Taylor, “A Landscape of the New Dark Silicon Design Regime”, Proceedings of International Symposium on Microarchitecture (MICRO), pp. 8-19, 2013.

[102] Synopsys primetime, www.synopsys.com.

[103] Y. Zou and S. Pasricha, “NARCO: Neighbor Aware Turn Model Based Fault Tolerant Routing for NoCs”, IEEE Embedded System Letters, pp. 85-89, 2010.

[104] J. Henkel, L. Bauer and N. Dutt, “Reliable on-chip systems in the nano-era: lessons learnt and future trends”, Proceedings of the 50th Annual Design Automation Conference, pp. 1-10, 2013.

[105] H. T. Nguyen and Y.Yagil, “A systematic approach to SER estimation and solutions”, Proceeding of IEEE International Reliability Physics Symposium, pp. 60-70, 2003.

[106] N. Seifert and N. Tam, “Timing Vulnerability Factors of Sequentials”, IEEE Transactions on Device and Materials Reliability, pp. 516-522, 2004.

[107] M. Mohamed, Z. Li, Xi.Chen, L. Shang, and A. R. Mickelson, “Reliability-Aware Design Flow for Silicon Photonics On-Chip Interconnect”, IEEE Transactions on Very Large Scale Integration (TVLSI), pp. 1763-1776, 2014