5-2015

Design of Reliable and Secure Network-On-Chip Architectures

Dean Michael B Ancajas
Utah State University

DESIGN OF RELIABLE AND SECURE NETWORK-ON-CHIP ARCHITECTURES

by

Dean Michael B Ancajas

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

of

DOCTOR OF PHILOSOPHY

in

Electrical Engineering

Approved:

Dr. Koushik Chakraborty Major Professor	Dr. Sanghamitra Roy Committee Member
Dr. Todd Moon Committee Member	Dr. Chris Winstead Committee Member
Dr. Dan Watson Committee Member	Dr. Mark R. McLellan Vice President for Research and Dean of the School of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2015

Abstract

Design of Reliable and Secure Network-on-Chip Architectures

by

Dean Michael B Ancajas, Doctor of Philosophy

Utah State University, 2015

Major Professor: Dr. Koushik Chakraborty
Department: Electrical and Computer Engineering

Network-on-Chips (NoCs) have become the standard communication platform for future massively parallel systems due to their performance, flexibility and scalability advantages. However, reliability issues brought about by scaling in the sub-20nm era threaten to undermine the benefits offered by NoCs. This dissertation demonstrates design techniques that address both reliability and security issues facing modern NoC architectures. The reliability and security problem is tackled at different abstraction levels using a series of schemes that combine information from the architecture-level as well as hardware-level in order to combat aging effects and meet secure design stipulations while maintaining modest power-performance overheads.

(128 pages)

Public Abstract

Design of Reliable and Secure Network-on-Chip Architectures

by

Dean Michael B Ancajas, Doctor of Philosophy

Utah State University, 2015

Major Professor: Dr. Koushik Chakraborty
Department: Electrical and Computer Engineering

The trend towards massive parallel computing has necessitated the need for an On-Chip communication framework that can scale well with the increasing number of cores. At the same time, technology scaling has made transistors susceptible to a multitude of reliability issues. This dissertation demonstrates design techniques that address both reliability and security issues facing modern NoC architectures. The reliability and security problem is tackled at different abstraction levels using a series of schemes that combine information from the architecture-level as well as hardware-level in order to combat aging effects and meet secure design stipulations while maintaining modest power-performance overheads.

This dissertation is lovingly dedicated to my parents, Norma Ancajas and Danilo Ancajas, and to my brothers and sister: Diana, Jan and Norman. Their support, encouragement and constant love have sustained me throughout my life.

Acknowledgments

I would like to express my gratitude to the many persons who have helped me during my PhD years. To my committee chair, Dr. Koushik Chakraborty, and my co-advisor, Dr. Sanghamitra Roy, for their advice, patience, financial support and guidance throughout my entire tenure as a PhD student. They have given me the opportunity to work on various high-impact research topics and have influenced the way I think about various issues both in my professional and personal life. To all of my committee members, Dr. Todd Moon, Dr. Chris Winstead and Dr. Dan Watson, for their valuable insights and comments on this research. I also want to thank them for supporting me, despite numerous obstacles and tough situations, in obtaining a summer internship during my final year at USU.

I would also like to thank members of the Bridge Lab–those who have moved on already, those currently enjoying doing research, and those just beginning–for their companionship, support and friendship. Thank you to Kshitij for being an excellent roommate and friend; Yiding for the friendship and those late-night coding sprees to finish our global routing program; Jason for the work we did in the Exascale NoC and Fort-NoCs; Saurabh for the opportunity to work on the NBTI-aware register file design; Satyajit and Gopal for the friendship and the fun racquet ball games; Hu Chen for the fun conversations about life and research; and Rajesh for continuing the work on NoC reliability/security. I would like to acknowledge several students who have helped me one way or another: McCabe, Brennan, Manzi, Brian, Prabal, Shayan, Chidham, Shamik, Harshita, Jacob, Andrew, Niranjan, Ashish, Soojeong and Jamy.

My deepest appreciation goes to the ECE Department and all of the staff members, for giving me the opportunity to pursue my PhD studies. I am very thankful to Mary Lee Anderson and Tricia Brandenburg for the invaluable support in helping me solve various administrative matters; Trent Johnson and Scott Kimber for the prompt technical support and the condor pool maintenance; you have contributed greatly in allowing me to beat paper deadlines.

Last, but certainly not least, I would like to thank my significant other, Tanya Abaya, for being my partner in crime and for experiencing life together; my good friend Julio Martin, for the fun times and adventures we undertook in Logan; my families: the Ancajas-es, Blascos, Calabias and the Rabes for the neverending support throughout my career.

Dean Michael B Ancajas

Contents

List of Tables

List of Figures

Acronyms

AcTh	Accomplice Thread
ALU	Arithmetic Logic Unit
BC	Bit Cruising
BCCLS	Bit Cruising & Crossbar Lane Switching
BRAR	Buffered-Router Aware Routing
CAM	Content-Addressable Memory
CDF	Cumulative Distribution Function
CMP	Chip Multiprocessor
CMOS	Complimentary Metal Oxide Semiconductor
CLS	Crossbar Lane Switching
CS	Cruise Setting
C-NoC	Compromised NoC
DSENT	Design Space Exploration for Network Tool
DC	Direct Current
DS	Data Scrambling
DCM	Distributed Cycle Mode
EDPPF	Energy Delay-Product per Flit
HAAF	HCI Aging Analysis Framework
hNoC	Heterogeneous Network-on-Chip
IB	Input Buffers
IP	Intellectual Property
NoC	Network-on-Chip
EM	Electromigration
HCI	Hot-Carrier Injection
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
MPSoC	Multiprocessor SoC

NI	Network Interface
NBTI	Negative Bias Temperature Instability
NMOS	n-channel MOSFET
NObf	Node Obfuscation
OCP	Open Core Protocol
OS	Operating System
PMOS	p-channel MOSFET
PDF	Probability Distribution Function
PC	Packet Certification
PE	Processing Element
PLB	Permission Lookaside Table
RC	Route Calculation
RTL	Register Transfer Level
RIE	Routing Information Extraction
RIR	Routing Information Register
SA	Switch Allocators
SoC	System-on-Chip
SNR	Signal-to-Noise Ratio
SVP	Single Vulnerability Period
SRAM	Static Random Access Memory
TDDB	Time-Dependent Dielectric Breakdown
TSMC	Taiwan Semiconductor Manufacturing Company
VC	Virtual Channel
VPI	Verilog Procedural Interface
WF	Wearout Factor
WMS	Wearout Monitoring System

Chapter 1

Introduction

Rapid technology scaling has fueled an unprecedented growth in the semiconductor industry, transforming the face of modern society. Commodity systems have undergone a sea change from the uniprocessor era of past decades to the current multicore era. Furthermore, the prospect of many-core systems in the near future looms large. With numerous on-chip processing cores, the communication fabric in many-core systems becomes a critical design component. Network-on-Chip (NoC) architectures are widely touted as the most promising design for the communication platform for future many-core systems, primarily due to their scalability. Many-core prototypes such as the Intel 80-core, Intel Single Chip Cloud system and Tilera have already used NoCs as the backbone of communication between their on-chip processors [1, 2].

Continuous technology scaling also causes a rapid deterioration in the device-level robustness, forcing the need for fault-resilient system design. While a large body of recent works targets on-chip computing resources (processing cores), the emergence of the many-core era has raised the importance of addressing the sustainability of on-chip communication architectures such as NoCs. Future many-core systems, comprising abundant rudimentary in-order cores can lack in their individual ability to hide network latency [2]. Consequently, aging-induced performance degradation in the NoC can create a serious bottleneck, undermining system-level efficiency and reliability. With economic viability playing a central role in both personal and corporate decision making, it is imperative to ensure sustained lifetime in both NoCs and processing cores, in order to have a meaningful impact at the system level.

In this context, this dissertation addresses sustainability challenges in NoC architectures while preserving system-level efficiency. Merely optimizing for power-performance efficiency creates a fundamental tension in the NoC design spectrum. Existing state-of-the-art techniques for NoCs attempt to tackle this tension in a reactive manner by focusing solely on power-performance until a component failure occurs, after which corrective measures are triggered [3–7]. This work presents a series of proactive orthogonal approaches, simultaneously addressing the need for aging-awareness while preserving system power-performance efficiency in NoCs.

Thwarting aging-induced latency degradation in NoCs can be instrumental in designing sustainable future many-core systems. The techniques proposed in this work can expand the time span of fault-free execution in the NoC through symmetric and graceful aging degradation. This dissertation also demonstrates that under certain situations, aging resilience can be provided in NoCs without sacrificing power-performance. This intriguing property, achieved by recognizing the criticality aspects of on-chip communication, can spawn similar effective techniques in the pursuit of balancing power-performance and robustness in NoC architectures.

This dissertation also addresses a new emerging challenge in the design of NoCs: *Security*. As System-on-Chip development grows in complexity and cost, there is an increasing emphasis on employing third party IP NoC blocks to connect components for cost-reduction purposes. One of the key assumptions in trustworthy computing is that the software is running on top of a fully secure hardware. With tight time-to-market deadlines and increasing popularity of 3rd party Intellectual Property (3PIP) use, this assumption does not hold true anymore and causes current security techniques to be circumvented. In particular, a series of low overhead approaches where the system security is maintained in the presence of an information-leaking HW trojan embedded by a malicious 3PIP designer.

1.1 Contributions of This Research

The findings from this research are very promising, with one work being nominated for a Best Paper Award in *2014 IEEE International Conference in Hardware/Software Codesign and System Synthesis (CODES+ISSS)*. Several publications stemming from this dissertation can also be found in the proceedings of the *2013 & 2014 IEEE/ACM Design*

Automation Conferences (DAC), *2013 IEEE/ACM Design, Automation and Test in Europe (DATE)* and *2014 IEEE Transactions on Very Large Scale Integration (TVLSI)*.

Publications made during this period of research are listed as follows:

1.1.1 Conference Papers

- Tackling QoS-induced Aging in Exascale Systems. Dean Michael Ancajas, Koushik Chakraborty, Sanghamitra Roy. *2014 IEEE International Conference in Hardware/Software Codesign and System Synthesis (CODES+ISSS)*

- Fort-NoCs: Mitigating the Threat of a Compromised-NoC. Dean Michael Ancajas, Koushik Chakraborty, Sanghamitra Roy. *2014 IEEE/ACM Design Automation Conference.*

- HCI-Tolerant Network-On-Chip Router Microarchitecture. Dean Michael Ancajas, Koushik Chakraborty, Sanghamitra Roy. *2013 IEEE/ACM Design Automation Conference.*

- Proactive Aging Management in Heterogeneous Network-On-Chip Architectures. Dean Michael Ancajas, Koushik Chakraborty, Sanghamitra Roy. *2013 IEEE/ACM Design, Automation and Test in Europe (DATE)*

1.1.2 Journal Paper

- Wearout Resilient Network-On-Chip Architecture. Dean Michael Ancajas, Koushik Chakraborty, Sanghamitra Roy. *2014 IEEE Transactions on Very Large Scale Integration (TVLSI).*

1.2 Tackling QoS-induced Aging in Exascale Systems

This work shows that supporting QoS in an exascale NoC has profound implications on its reliability. Through experiments, it is shown that as a NoC is scaled, providing QoS dramatically reduces its Mean Time To Failure (MTTF) due to the increased power consumption and elevated thermal profile. This increased power/thermal characteristics from QoS support does not come from a performance increase, rather from a more proportionate resource management enabled through QoS support [8]. Thus, although the NoC system continues to offer an identical bandwidth, QoS support results in an accelerated wearout and a reduced lifetime.

To effectively increase the lifetime of QoS-enabled exascale NoCs, this research introduces *Dynamic Wearout Resilient Routing (DWRR)* algorithms. The DWRR algorithms balance QoS and reliability impact by extenuating the additional stress induced by enforcing QoS guarantees in the system. The proposed schemes are based on a cross-layer theme, where device-level wearout is sensed at the circuit layer, and communicated to the architecture layer to dictate the routing path selection. Overall, the lifetime is increased by aiming to retain uniform wearout among all routers in an exascale NoC. To the best of the author's knowledge, this is the first work to tackle the tension between reliability and QoS in an exascale NoC.

The contributions of this work are as follows:

- **Reliability Analysis of QoS and Scaling Effects in Exascale NoCs:** An analysis of the reliability impact of providing QoS in large scale NoCs. The experiments in Section 5.1 show that the MTTF of a 1024-node NoC can be reduced by 36.6% (two years) of its rated lifetime just by supporting QoS. In the same light, as the number of nodes in the system are increased from 64 to 1024-nodes, the MTTF can be reduced by as much as 48%. These looming problems signify a need for managing the resources in an exascale NoC appropriately.

- **DWRR Algorithms:** This work also developed two novel DWRR algorithms (Section 5.2.4). The first algorithm, fresh routing (FR), lengthens NoC lifetime by always prioritizing the least degraded path at the cost of reduced QoS. The second algorithm, latency-reclamation routing (LR), gives balanced priority to aging and QoS.

- **Top-down Evaluation Platform:** A top-down simulation platform was developed to provide a rigorous assessment of system reliability. The infrastructure spans the whole spectrum of design abstraction layers, from architectural simulation down to transistor-level analysis. In this process, several tools such as Booksim 2.0, DSENT 0.91, HotSpot 5.02, an in-house logic analysis tool and HSpice are integrated. These tools perform architectural simulation, power analysis, thermal analysis and delay degradation evaluation of major wearout mechanisms. Compared to the recently proposed state-of-the-art VICIS [9], the best routing algorithm improves QoS and long-term sustainability (Mean Time To Failure) of the system by 9% and 25%, respectively, while avoiding an increase in flit latency.

1.3 HCI Tolerant NoC Router Microarchitecture

This work proposes a NoC router microarchitecture that lessens the aging effect of Hot Carrier Injections. HCI is an unrecoverable aging phenomena [10], which affects the components due to their dependence on switching activity [11]. Due to aggressive transistor scaling, the thinner gate dielectric in CMOS transistors increases the probability of HCI degradation. In fact, HCI can account for a major component of aging in a 10-year product lifetime [12]. To the best of the author's knowledge, none of the existing works consider HCI aging mitigation in the NoC architecture.

This research performs a holistic cross-layer analysis of HCI degradation in the NoC router microarchitecture. The crossbar structure of the router microarchitecture has been the focus of this study, due to its profound significance in dictating the router frequency [13]. Combining application-level traffic profile with bit-level logic analysis, it is found that the crossbar structure is highly vulnerable to HCI aging. Due to the data communication patterns in many-core applications, it is observed that a majority of gate-level switching activities are restricted to a small portion of the entire crossbar circuit topology, resulting in a large HCI degradation. To throttle HCI aging in the crossbar, this work proposes a series of low-overhead techniques that evenly distribute the switching activity in the crossbar, without affecting the architecture-level routing latency and bandwidth.

The contributions made through this work are:

- This work developed a cross-layer framework for HCI aging analysis of an NoC router. The framework combines application traces, RTL gate-level simulation of a crossbar circuit, logic analysis, and HSPICE simulation of HCI degradation effect (Sections 4.2.2 and 4.4.3).

- An analysis of the switching activity of the crossbar, a major circuit in an NoC router using real-world applications and find that only a small group of gates account for most of the switching activity. On an average for PARSEC benchmarks, only 25% of the gates account for more than 75% of the switching activity, severely damaging some gates while leaving others unscathed (Section 4.2).

- A proposal for four schemes using low overhead techniques to evenly distribute the switching activity and minimize HCI degradation (Section 4.3). The four schemes are: *Bit Cruising* that distributes the high activity bits around the channel; *Distributed Cycle Mode* that exploits idle cycles in the NoC; *Crossbar Lane Switching* that manipulates the port in the crossbar by utilizing the virtual channels; and a combination of Bit-Cruising and Crossbar Lane Switching.

- A holistic evaluation of the proposals spanning full-system simulation down to RTL and gate-level HSPICE simulation (Section 4.5). The best schemes proposed in this work improve the switching activity distribution by up to 31% (ave: 19%). A maximum of 30% (ave: 26%) improvement in the clock cycle degradation, while the system performance degradation is reduced by up to 17.6% (ave: 11%) compared to the baseline scheme are also observed. The Energy Delay Product per Flit is improved by up to 27% (ave: 17%).

1.4 Proactive Aging Management in Heterogeneous Network-on-Chips

This research studies reliability issues inherent in power-efficient Heterogeneous Network-On-Chip architectures. In particular, the asymmetric utilization seen in NoC components, substantially exacerbate aging degradation. For example, Mishra et al. observe that routers in the central region of the mesh can have more than 2X utilization compared to those near the peripheral region [14]. Such higher utilization in some NoC components can manifest as rapid aging-induced power-performance degradation in those components, causing system-wide deterioration.

To effectively tackle aging degradation in power-efficient NoCs, this work presents a novel adaptive routing algorithm that dynamically modifies traffic flow and exploits opportunities to avoid the use of buffered routers, while minimizing system-level power-performance impact. The proposed approach incorporates a *Wearout Monitoring System (WMS)* (composed of NBTI delay sensors) in NoC components, and combines architecture-level packet criticality during routing to relieve the heavy usage in aged components. Overall, the proposed routing algorithm can substantially reduce the effects of aging degradation in *hNoCs*, thereby assuring a graceful degradation in the communication architecture of emerging systems.

This work makes the following contributions:

- **Reliability Analysis of an hNoC:** This research presents an extensive study analyzing the impact of routing policies in an *hNoC* (Section 3.1). The analysis from this work shows an alarming increase in utilization asymmetry in certain *hNoC* components (by more than 1.35×), which can cause rapid aging degradation in those components, severely affecting system-level performance characteristics. In this context, a new opportunity in reliability-driven routing in an *hNoC* was uncovered, by demonstrating that a substantial portion of the data packets routed in the network are non-critical (i.e., system performance is insensitive to their latency). Thus, utilization in centrally placed buffered routers can be reduced by minimizing non-critical packets routed through them.

- **Proactive Routing to Mitigate NoC Aging:** This work developed a novel dynamic aging-aware routing algorithm based on architecture-level criticality information and accumulated wearout information using a proposed Wearout Monitoring

System (Section 3.2). To the best of the author's knowledge, this is the first work that uses the criticality information of packets to improve the reliability of NoCs.

- **Holistic Evaluation:** A holistic evaluation of the proposed algorithm is done by integrating: (a) SPICE-level simulation to estimate the combined effect of device aging based on circuit-level utilization and process variation (Section 3.3), (b) statistical timing analysis of synthesized hardware to accurately estimate the delay distribution under aging and expected usage pattern (Section 3.3), and (c) cycle accurate architectural simulation using GARNET and GEMS toolsets with multithreaded applications on a 16-core system with an *hNoC* (Section 3.4). Compared to the aging overhead seen in a state-of-the-art *hNoC* routing scheme [15], the best scheme developed in this work shows 38%, 53% and 29% improvement in network latency, system performance and EDPPF degradation (on average), respectively (Section 3.5).

1.5 Fort-NoCs: Mitigating the Threat of a Compromised-NoC

This research on security uncovers a novel and imminent threat to an emerging computing paradigm: MPSoCs built with third party IP NoCs. It is demonstrated that a compromised NoC (C-NoC) can enable a range of security attacks with an accomplice software component. To counteract these threats, a technique called Fort-NoCs[1] has been proposed, a series of schemes that work together to provide protection from a C-NoC in an MPSoC. Fort-NoCs's foolproof protection disables covert backdoor activation, and reduces the chance of a successful side-channel attack by "clouding" the information obtained by an attacker. Compared to recently proposed techniques, Fort-NoCs offers a substantially better protection with lower overheads.

This work shows that a malicious third party vendor can provide a compromised NoC by embedding a hardware trojan within the IP block. Such a trojan can facilitate a range of possible attacks with an accomplice software. For example, an accomplice software component can establish a covert communication with the NoC to snoop on the ongoing

[1] Fort-NoCs is a word play between NoCs and Fort Knox–a security hardened structure housing US gold deposits.

data communication through the NoC, thereby stealing classified information. Many other types of attacks such as voluntary data corruption or denial of service are also possible.

The following contributions are made through this work:

- A new threat model stemming from a C-NoC design. This model can be a potent security threat as third party NoCs become more prominent in low-cost cloud computing on MPSoCs (Section 6.1).

- A detailed design of a C-NoC and evaluate its design footprint and runtime performance overhead. The rigorous analysis performed in this dissertation demonstrates that it is possible to realize this threat model with a minimal footprint, clearly showcasing the potency of the threat model that is uncovered in this work (Section 6.2).

- Fort-NoCs, a holistic layered approach to harden security on systems with a C-NoC: *Node Obfuscation* (NObf), *Packet Certification* (PC) and *Data Scrambling* (DS). The proposed techniques play complementary roles in hardware level protection by preventing the two-way communication between software and the hardware C-NoC, and introducing noise in the NoC data communication semantics with negligible overhead on performance and bandwidth of the NoC (Section 6.3).

- A thorough analysis of the proposed secure design solutions by measuring the associated power, area, and performance overheads. Compared to the recently proposed state-of-the-art NoC-MPU [16], the proposed Fort-NoCs offers compelling advantages in power-performance overheads and threat resilience from a C-NoC (Section 6.5).

Chapter 2

Literature Survey and Related Work

In this chapter, a comprehensive literature survey is done to build the foundation of this research. The relevant areas related to this study include the current state-of-the-art work on power-performance, reliability and security optimizations in NoCs. To this end, this chapter is organized into the following sections: Section 2.1 explains recent work that tackle power-performance, reliability and security issues. The succeeding sections collate the information in Section 2.1 in order to clearly delineate the contribution of this dissertation. Section 2.2 positions this research in the realm of existing work on Exascale NoCs. Section 2.3 justifies the *hNoC* reliability research against existing works. In Section 2.4, the importance of addressing Hot-Carrier Injection is explained. Lastly, Section 2.5 discusses the current landscape of security research in NoCs.

2.1 State-of-the-Art in NoC Research

This section explains the current state-of-the-art in reliability, power-performance and security research.

- **BRAR [15]**: BRAR stands for Buffered-Router Aware Routing. It is a routing technique that exploits the non-uniformity of router design in an *hNoC*. In a traditional *hNoC*, the flits can be misdirected when there is conflict in bufferless routers. BRAR decreases the probability that a flit is deflected by routing towards buffered routers. Overall, BRAR is optimized for a power-efficient *hNoCs*.

- **Yin et al. [17]**: Yin's work optimizes path routing in an *hNoC* that is used as a CPU-GPU communication platform. It is primarily motivated from the fact that GPU traffic can tolerate a slack. Since GPU traffic is usually bandwidth-sensitive, it can tolerate a small delay without affecting its overall performance. Thus, packets can be routed using non-minimal paths to be able to optimize energy-efficiency.

- **Li et al.** [**18**]: Li's work optimizes the packets' latencies by identifying two types of traffic that are in the network. There are latency-critical packets and latency-noncritical packets. The former traffic is important as there prompt delivery dictates the performance of the whole program. Subsequently, latency-noncritical packets can be delayed without affecting the performance. Li created a run-time technique which leverages data-transaction and protocol information. By letting latency-critical traffic bypass pipeline stages, they are able to accelerate a program's performance.

- **AGE-ADAP** [**19**]: AGE-ADAP created an adaptive routing algorithm to relieve NBTI stress in heavily utilized routers and links. They introduced a new metric called Traffic Threshold per Epoch (TTpE) that limits the amount of traffic that a router can have in a time interval. By using a control system to adjust the TTpE, they are able to provide graceful aging degradation in NoCs.

- **Fu et al.** [**20**]: Fu's work optimizes the NoC architecture reliability for both process variation and NBTI effects. They improve the reliability profile of virtual channel allocation logic and registers. Fu's work require an online detector to detect the transistor degradation of both logic and register circuits. Using the detected values, the degradation can be controlled to ensure uniform aging.

- **Vicis** [**9**]: Vicis is an NoC design that can tolerate the loss of many network nodes due to aging. Each router node in Vicis has a built-in self test circuit that checks hard-faults in the system and corrects them by using ECC, port swapping and crossbar bypassing. Vicis can tolerate a large amount of errors (1 in 2000 gates) and can also operate while only half of its routers are fully functional.

- **ROCO [21]:** ROCO aims to design a low-latency, energy-efficient and reliable NoC. ROCO decouples the arbitration and uses smaller crossbars for row and column decoding. By using many crossbars, ROCO is able to reduce the output port contention probabilities compared to a baseline router. The modular design of ROCO allows graceful aging degradation in the even of hard-faults. Consequently, it also has less dynamic power.

- **Park et al. [22]**: Park et al. examined the impact of transient errors in the reliability of NoCs. They proposed novel techniques to tolerate or prevent from such errors. To handle soft link errors, they introduce a retransmission scheme that detects and retransmits flits that were corrupted temporarily. They claim that the design of their retransmission scheme has lower overhead compared to other works.

- **BLESS [23]:** BLESS is the first work that extensively evaluated the feasibility of using a complete bufferless network. They argue that buffers in modern NoCs consume enormous energy, occupy a large chip area and also increase the design complexity due to the circuits needed to manage them. Routing without buffers significantly reduce energy consumption while providing the same performance at lower network loads. BLESS can be used for processor/cache networks where network utilization is low.

- **Abeyratne et al. [24]:** Abeyratne et al. conducted several experiments on asymmetric high-radix topologies to explore network designs that can support thousands of processors. Asymmetric topologies optimize both local and global communication by matching the wire speed and the router speed. They used fast medium-radix router for local communication and a few slow high-radix routers for optimized global communication. They are able to achieve about 45% network latency improvement compared to a traditional mesh. case for on chip bufferless networks

- **Kilo-NoC [25]:** Kilo-NoC proposes an *hNoC* for scalability and QoS. They proposed a network that has few nodes providing QoS. Traffic requiring QoS are then routed towards this region before being sent to its final destination. They used elastic buffers to provide a fast communication path into and out of QoS region. By constraining QoS in a specific region with few routers, Kilo-NoC is able to provide QoS in an massive setting without severely compromising power consumption of the NoC.

- **PVC [26]:** PVC implements a Virtual Clock (VC) control flow with preemption capabilities in an on-chip network. They discuss the challenge of implementing a power and energy efficient VC flow in an on-chip network that has a smaller silicon budget compared to a traditional IP network. The preemption capability of the network allows PVC to prioritize critical traffic in order to maintain QoS in the system while requiring only a small amount of buffers compared to the traditional VC control flow. PVC employs a low-latency global acknowledgment network in order to facilitate preemption and retransmission of packets.

- **Hosseini et al. [4]:** Hosseini et al. present a new dynamic routing algorithm for NoCs serving SoCs. The routing algorithm they created has the ability to pinpoint both static and dynamic permanent failures. The algorithm is also able to distinguish between permanent failures and soft-errors by keeping track how many errors are encountered in a particular phase. The algorithm uses the acknowledgment system of a NoC to ensure that packets are correctly sent to the downstream router. In addition to reliability advantages, their algorithm can also distribute the load over the whole network by considering the stress factors in the system. The algorithm uses the queue length at each router as an indicator of network congestion.

- **Explicit Path Routing [5]:** Chaix et al. propose Explicit Path Routing (EPR), a deadlock-free adaptive routing algorithm. The goal of EPR is to limit latency degradation of packets in the NoC even under faulty conditions. Deadlock avoidance for EPR is implemented using a variant of the turn-restriction model. Essentially, some turns are prohibited depending on the path of the packet so as to break cycle dependencies. EPR also provides high performance under faulty conditions by studying the state of the network at runtime. Whenever the system is reconfiguring, an *echo* packet is sent, which finds the optimal path of a source-destination pair. Once the path of a faulty source-destination pair is determined, this path is used indefinitely.

- **BFT-NoC [6]:** Tsai et al. propose the Bidirectional Fault-Tolerant NoC technique that is capable of addressing both static and dynamic channel failures. By using bidirectional channels, a faulty router can be reached as long as one of its links is still functional. In contrast, schemes that use unidirectional links will result in unreachable nodes the moment one of its links becomes faulty. Under faultless conditions, BFT-NoC operates similarly as a traditional NoC router. However, once a fault is detected, BFT-NoC routers go through a reconfiguration phase where the channel mode (i.e. transmit or receive) is being determined based on network demand. This approach provides graceful degradation for NoCs.

- **Zhang et al. [7]:** Zhang et al. propose a reconfigurable routing algorithm that is deterministic, fault-tolerant, distributed and reconfigurable. The main idea of the algorithm is to create a path to route the packets through a cycle-free contour that surrounds a faulty router. Hence, restoring all broken paths caused by the faulty router. The algorithm uses 9 canonical contours of a mesh NoC to establish paths at runtime. Once a packet cannot find a route using the normal algorithm due to faults in the system, the routing path is changed by connecting the different contours to establish a path. Deadlock-avoidance is achieved by prohibiting specific turns in the path.

- **FARM [27]:** FARM stands for Fault-Aware Resource Management, a system-level fault-tolerant technique that uses dynamic application mapping to achieve optimal system performance while minimizing communication energy consumption. FARM's resource allocation algorithm also takes into account permanent, transient and intermittent faults in the system. The main idea of FARM is a feedback mechanism where system runtime metrics are gathered in order to determine the optimal task mapping. Based on current conditions, a multidimensional problem is solved at runtime to come up with a task mapping. FARM optimizes metrics such as manhattan distance of communicating tasks, link contention, and system fragmentation.

- **Lan et al. [28]:** Lan et al. present a study in Performance and Energy tradeoffs in reliable NoCs. The reliability issue considered in the work is that of ECC (Error Control Codes) hardware implemented in NoC routers. The work's main objective is to characterize the causes of energy consumed for fault tolerance hardware and also to provided a comparison among different ECC implementations. Note that the ECC only corrects errors present in the data plane. In contrast, errors can be also present in the control plane of the router (i.e. allocation, routing, arbitration etc ..). The ECC codes compared by the study are CRC4/8, DED, PAR, SEC, SECDED. It is found out that the retransmission of flits creates the biggest energy consumption as opposed to the different encoding and decoding HW required by ECC.

- **FoREVER [29]:** Parikh et al. propose FOREVER: Formally Enhanced Runtime Verification to ensure NoC Functional Correctness, a scheme that complements the use of formal verification and runtime verification. It addresses the limitations of formal verification by creating a runtime technique that detects errors, and recovers from them. In FoREVER, a bare bones light-weight checker network is used to alert destination nodes of incoming packets in advance. Once a bug in the HW is detected (i.e. via an undelivered packet), the original packet is sent to the destination using the alternate checker network. This provides full functional coverage as long as the checker network is error-free. FoREVER claims to recover errors within 30K cycles.

- **NoCAlert [30]:** Prodromou et al. proposes NoCAlert, an online and real-time fault detection scheme for NoCs. NoCAlert uses the concept of invariance checking. Similar to FoREVER, it uses micro-checker modules that essentially implements hardware assertions. The checker modules operate in lockstep unison with normal NoC operation, preventing the need for periodic or trigger-based self testing. NoCAlert claims 97% instantaneous detection of faults from the test vectors they have used. By using invariance checking, NoCAlert reduces the solution space of the errors that it needs to detect. This is because some of the errors in the system are harmless (i.e. a misrouted flit can still reach its destination). As such, it provides good coverage at a very low overhead.

- **StageNetSlice [31]:** Gupta et al. propose StageNetSlice, a reconfigurable microarchitecture building block for resilient CMPs. StageNet relies on a reconfigurable network of replicated processor pipeline stages to maximize the lifetime of a chip. Hence, allowing graceful degradation of the microprocessor. Each pipe stage is replicated and are stitched together using a crossbar switch. StageNetSlice efficiently handles inter pipeline dependencies in order to come up with a fast distributed network design. As errors are manifested in the pipeline throughout its lifetime, the chip is also reconfigured in order to operate in an error-free manner.

- **Siddiqua et al. [32]:** Siddiqua et al. propose Recovery Boosting, a technique that addresses NBTI degradation in SRAM cells. NBTI affects SRAM cells when a logic "0" is stored in the cell. As such, Recovery Boosting counteracts this effect by engaging PMOS devices in recovery mode (i.e. biasing to Vdd) during times of inactivity in the system. Recovery Boosting is implemented in the physical register file of a processor to enhance its reliability. To integrate Recovery Boosting in a modern processor, the control circuitry of the Register File must be modified to support Recovery mode when a register is not used. Recovery Boosting has been shown to provide over 50% static noise margin improvement for the register file.

- **Fiorin et al. [33]:** Fiorin et al. propose a scheme to enforce security of memory accesses in NoCs. The proposed idea uses data protection units (DPU), implemented in the network interfaces, to authenticate memory accesses in the system. All the DPUs are managed by a central unit called the Network Security Manager. The DPUs, configured by the NSM, serves as a firewall that checks and limit the access rights of nodes accessing data and instructions in shared memory. It divides the resources in the system (i.e. memory, IP) into secure/unsecure areas and limits accesses based on user policies. As long as the NSM is uncompromised, the system can be considered secure.

- **Diguet et al. [34]:** Diguet et al. proposed a NoC architecture that protects SoC against attacks such as: Hijacking, secret information extraction and Denial of Service. This work uses a 4-step approach towards SoC security. First, a single centralized, dedicated and secured master IP must supply all security-related configurations of the chip. Second, the network interfaces must be enhanced in order to enforce access rights restriction. Third, there must be a two separate VCs in the network in order to separate communication and security data. Fourth, holistic integration of previous steps to determine if the SoC security has been compromised.

- **Gebotys et al. [35]:** Gebotys et al. proposed a framework for security on NoC enabled technologies. Security is addressed in both the transport layer (i.e. network) and the application layer (i.e. core). For the transport layer security, each IP core has a security wrapper and a module that securely hides private keys are included in each NoC router. This model prevents unencrypted private keys from ever leaving the cores and NoC. At the core level, an obfuscation method is used to hide the correlation between a hash calculation and its power profile. This two-pronged approach creates a secure system that is immune to probing and side-channel attacks.

- **Kapoor et al. [36]:** Kapoor et al. proposed a framework that uses Authenticated Encryption and Session Keys to implement security in a NoC. Encryption and decryption of keys are implemented in the network interfaces of every IP core. Communications between secure IP cores use a predefined key while communications between secure and non-secure cores uses a dynamically generated session key. By continuously changing the key, the NoC can determine if the system has been attacked. DoS attacks are prevented using counters inside the NoC routers while unauthorized memory accesses are prevented using an access rights table.

- **SurfNoC [37]:** Wassel et al. proposed SurfNoC: a low latency and provably non-interfering approach to secure NoCs. The aim of SurfNoC is to provide security by maximizing temporal and spatial separation of different flows in the system. SurfNoC implements static isolation of flows in the network without sacrificing NoC performance. Static isolation of resources are desired to prevent sharing of information through side-channels. The key to SurfNoC's approach is to do network scheduling in wave manner. This allows multiple flows in the network to co-exist in a non-interfering nature while also avoiding the overheads of a cycle-by-cycle time multiplexing.

- **Wang and Suh [38]:** Wang and Suh proposed an efficient architecture for preventing timing-channel attacks in a network on chip. Their schemes uses a priority based arbitration and a static limit mechanism to provide one-way information-leak protection. To do this, the authors came up with a Reversed Priority with Static Limit algorithm. This algorithm assigns a higher priority to low-security traffic so that its behavior is not affected by high-security traffic. Hence, preventing a one-way information leak. The static limit is imposed for low-security applications in order to prevent it from doing a Denial-of-Service attack on the high-security application.

- **Illinois Malicious Processor [39]:** King et al. proposes many of techniques on how to design a malicious hardware such as a processor. Their main claim is that instead of creating a hardware that can support one type of attack, a flexible hardware can be created to allow powerful and general purpose attacks. The HW overhead for creating such “attack-infrastructure” are relatively small. The authors demonstrate the potency of a HW attack through a login backdoor. A circuitry inside a modern processor that can be used to easily provide an attacker complete privileged access into the system.

2.2 Reliability Considerations for Exascale Systems

Emerging systems, with hundreds of billions of transistors, are likely to have many faults, even at the point of tape-out [40]. These faults can impact both processing cores as well as NoC components, drastically reducing their functionality. Many recent NoC works target these faulty components, and outline a plethora of techniques to tolerate faults and sustain successful communication between two nodes [4–7, 27–30].

In the realm of exascale computing, most research on exascale NoCs have revolved around performance, energy efficiency, QoS and scalability. Moscibroda et al. explore the use of bufferless routers for a scalable and energy-efficient NoC network [23]. Abeyratne et al. explore scalable, asymmetrical high-radix NoC topologies for use in a Kilo-Core systems [24]. Grot et al. studied scalable QoS incorporation in exascale systems [26]. However, system resiliency is also fast becoming a first-class design constraint as designers continue to progress with Moore's law. Bhardwaj et al. explore aging aware routing in NoCs, but do not consider exascale systems and QoS implications [19, 41]. Similarly, several recent works explore el fault-tolerant routing in small-medium scale NoCs, but do not consider exascale NoCs. One of the most prominent works is VICIS by Fick et al. that provides a complete infrastructure for network recovery in the face of component failures [9]. Since most of the works in exascale focused on solving individual problems, they failed to realize that the reliability of the system is severely compromised when several of these proposed ideas are combined. To the best of the author's knowledge, this is the first work to clearly identify the reliability tension in exascale NoCs, caused by scaling and supporting QoS. The proposed novel approach aims to boost NoC sustainability without sacrificing QoS and scalability.

2.3 Reliability Design in *hNoCs*

Few works beyond the BRAR [15] investigate routing approaches in an *hNoC*. Yin et al. [17] proposed an energy efficient non-minimal path routing algorithm for a heterogeneous NoC running a CPU-GPU system by exploiting the slack provided by the bandwidth-sensitive GPU traffic. In essence, they classify criticality of flits as the source of the traffic (GPU/CPU). Li et al. present runtime techniques to reduce the overall network latency of latency-critical packets by letting them bypass the router pipeline stages, hence improving performance [18].

While previous works focus on using criticality to improve performance and power efficiency in a router-homogeneous NoC, the technique proposed in this work is the first to consider criticality in the context of reliability-driven routing in heterogeneous NoCs. The routing technique mitigates the effect of aging degradation by relieving the burden on routers that are likely to be highly degraded, while minimizing the system level impact of non-optimal flow control. Moreover, the WMS and deflection based schemes can also be used in homogeneous NoCs to add aging-aware functionality.

2.4 HCI-Aware Design of NoCs

The aggressive scaling in CMOS technology has made reliability a primary design constraint in modern computing systems. While there has been a wide scope of studies tackling different reliability issues (NBTI, TDDB, HCI) in processing elements [31, 42], there is only a limited number of works which address wear-out mitigation in the on-chip communication infrastructure of such systems. Bhardwaj et al. implemented a dynamic routing algorithm to equalize NBTI and electromigration aging across the on-chip network [19]. Fu et al. created new virtual channel allocation and routing algorithms in order to improve process variation and NBTI effects in key components of the router [20]. Park et al., Fick et al., and Kim et al. explored fault tolerant NoC architectures by decoupling modules and having redundancies in order to recover from intermittent errors in the network or provide graceful degradation [9, 21, 22].

Most of the studies mentioned above focus on recovering from intermittent errors or minimizing NBTI effect on storage elements by balancing the duty cycle. On the contrary, the technique proposed here will focus on HCI, an unrecoverable aging phenomena that affects combinational components. HCI mitigation presents a different set of challenges because of its dependence on the switching activity of transistors, as opposed to NBTI which depends only on the input bias. To the best of the author's knowledge, the proposed scheme in this work will be the first work to tackle HCI in an NoC router microarchitecture.

2.5 Landscape of Security for NoCs

A vast amount of recent works focus on hardware and software security, primarily to address a problem of growing importance. The discussion in this section is focused on NoC security, which is most relevant to the contribution of this research.

The current state-of-the-art in NoC security revolves around protecting information traveling in the network against side channel, physical and software attacks. Table 2.1 presents a high-level comparison of existing works on NoC security based on four major factors. Similar to software protection mechanisms, many existing works provide access control by monitoring the memory addresses (e.g., *Data Protection Unit (DPU)* proposed by Fiorin et al. [33], firewall from Sonics [43], access control on memory banks by Diguet et al. [34]). Some proposals aim to use encrypted data transmission over the NoC (e.g., [35,36]) or partition the NoC into separate zones based on trust [37,38]. In the industry, ARM has approached trusted computing through its TrustZone platform [44]. The idea is to establish secure and non-secure states throughout the chip. By changing the security mode of a component along with safety checks, information leaking from secure to non-secure areas are prevented.

The work presented in this dissertation explores a new threat model orthogonal to these works. First, it is assumed that the hardware trojan is embedded in the NoC itself, whereas others assume a trustworthy NoC. Second, it is demonstrated that information can be extracted from the NoC without relying on memory access (either through on-chip cache or off-chip memory). Third, it is also assumed that an accomplice software thread exists, which acts as the orchestrator of the attack. The hardware-software coalition in the proposed threat model is similar to the Illinois Malicious Processor (IMP) [39]. However, the IMP does not consider the threat of a C-NoC.

Table 2.1: Comparison of threats in NoCs.

	Trojan Loc[a]	**IS**[b]	**Prot**[c]	**TM**[d]
Fort-NoCs	NoC	NoC	NI	S/W
DPU [33]	S/W	Mem	NI	—
KeyCore [35]	S/W	Mem	NI	—
surfNoC [37]	S/W	S/W	NoC	—
AE [36]	S/W	Mem	NI	—
IMP [39]	μP	μP/Mem	—	S/W
NoC-MPU [16]	S/W	Mem	NI	—

[a] The part of the system where a *trojan* is inserted.
[b] The part of the system where the *information* is *stolen*.
[c] The part of the system where a *protection* mechanism is implemented to prevent an attack.
[d] The *triggering mechanism* in case of a hardware trojan.

Chapter 3

Reliability for Heterogeneous NoCs

This chapter discusses the technique developed in this work that addresses aging-induced degradation in *hNoCs*. The approach created is based on the combination of packet-criticality information and dynamic routing algorithms.

3.1 Motivation

In this section, the proposed scheme for mitigating aging effects in an *hNoC* is discussed by demonstrating two important circuit-architectural characteristics. First, it is shown that existing routing policies in a typical *hNoC* can lead to an alarming rise in the utilization asymmetry, exacerbating the NoC reliability challenge (Chapter 3.1.1). Second, the criticality of network packets in an NoC running multithreaded applications are analyzed to identify opportunities of improving reliability by exploiting packet non-criticality (Chapter 3.1.2).

Recently-proposed heterogeneous NoC designs exploit the non-uniform traffic across routers to proportion routing resources [14]. Zhao et al. [15] further improve performance by employing a routing algorithm that leverages the non-uniform structure of the network to move data along buffered routers to keep packets on their optimal paths. Routing through the buffered routers prevents these packets from being deflected arbitrarily by bufferless routers, which can increase their network latency. However, this routing approach overburdens the buffered routers as a majority of the flits (*flits* are the units of information flow in NoC networks) use buffered routers as a part of their pathways.

3.1.1 Utilization Asymmetry in Heterogeneous NoCs

The setup used in this study is shown in Figure 3.1(a). The *Buffered Router Aware Routing (BRAR)* routing algorithm from Zhao et al. [15] is used. Buffered routers are placed identically as shown in their work. BRAR routes flits along buffered routers to decrease the chance of packets deflected farther from their destination (Figure 3.1(b)). However, if there is output port contention, the flit is stored in the buffer and delayed for another cycle. In contrast, the routing on bufferless routers (Figure 3.1(c)) needs to deflect flits towards a different direction because once a packet loses arbitration for the switch, it will be sent out to any free port. Two out of three times, the packet will be sent out towards a non-optimal direction.

Figure 3.2 shows the increase in utilization asymmetry of the buffered routers when using the BRAR algorithm (distribution of buffered and bufferless routers are shown in Figure 3.1(a)). All 16 routers in a 4×4 NoC mesh are shown, where the numbers in each router indicate the percentage increase in utilization compared to a homogeneous NoC employing XY routing. Results shown are average across several multithreaded PARSEC benchmarks used in this study. It has been observed that the centrally placed routers in the *hNoC* show 23–82% increase in utilization, while all of the peripheral routers show a range of reduced utilization. Compared to the homogeneous NoC, this *hNoC* employing BRAR routing shows 1.35× increase in utilization asymmetry[1] which ultimately leads to a more than 2× increase in timing degradation on the buffered routers.

3.1.2 Criticality of Different Flits in NoCs

The latencies of various packets transmitted through a NoC can have varied effects on performance. Previous works have exploited this criticality to improve system performance [18,45]. This work exploits the latency criticality of packets in the proposed dynamic aging-aware routing policies. The criticality classification and the opportunity for aging-awareness routing through packet-criticality are discussed next.

[1] Utilization asymmetry is estimated by the range of utilization seen across the NoC components.

Criticality Classification: In general, precise estimation of the packet criticality at the NoC router is hard as it merely has information about source-destination and the packet type. A thorough criticality estimation may require information about the relative performance of running program threads [45, 46], detailed cache coherence transitions, and so forth. To mitigate this complexity, a low-complexity approach is employed, requiring no change in existing interfaces. This involves identifying criticality based on packet type and source-destination. Table 3.1 shows the summary of the different classifications. Using this policy, any data packet transmitted from L1 to L2 (destination) are tagged as non-critical in a shared two level cache hierarchy. A vast majority of these packets are writebacks because of cache eviction, and thus the system performance is insensitive to their network latency. Some of these packets are also a result of data sharing among on-chip cores, but they are expected to be a much smaller component due to the predominance of private data even in multithreaded programs [47].

Opportunity: Figure 3.3 shows the percentage of non-critical packets of PARSEC benchmarks averaged across all the buffered routers. An average of 49% of packets traversing through the buffered routers are non-critical and can actually take a different routing path with minimal performance degradation. Moreover, all benchmarks show substantial opportunity, ranging from 44% to 51% in these benchmarks. By redirecting non-critical traffic to the bufferless routers, the utilization of the buffered routers are minimized, thereby mitigating the aging effects in the buffered routers.

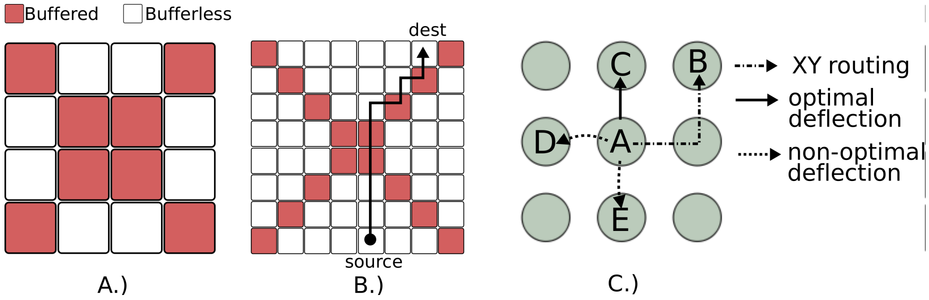


Fig. 3.1: A.) 4×4 mesh configuration B.) BRAR routing on an 8×8 mesh C.) Deflection routing on bufferless routers.

Table 3.1: Packet-criticality classification.

Data Messages		**Classification**
source	***dest***	
L1 Cache	L2 Cache	non-critical
L2 Cache	L1 Cache	critical
Memory	L2 Cache	critical
L2 Cache	Memory	non-critical

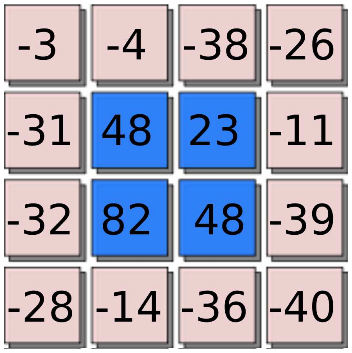
Control Messages		**Classification**
source	***dest***	
L1 Cache	L2 Cache	critical
L2 Cache	L1 Cache	critical
Memory	L2 Cache	critical
L2 Cache	Memory	critical

3.1.3 Significance for Reliability Driven Routing in Heterogeneous NoCs

The substantial presence of non-critical packets offers an intriguing opportunity for reliability aware routing in *hNoCs*, while preserving inherent advantages in power efficiency. Instead of always emphasizing the use of buffered routers, the routing algorithm can make a selective choice of routes based on packet criticality. Non-critical packets can be deflected away from the buffered routers, thereby reducing their utilization. The proposed approach in exploring such criticality-aware routing algorithms are explained in subsequent sections.

3.2 Design: Aging-Driven Routing via Packet-Criticality

This section discusses how the criticality information of packets are used to implement a deflection-based aging-driven routing algorithm. First, the Wearout Monitoring System (WMS) is explained. Then, it is discussed how WMS and criticality information are combined to implement aging-aware routing schemes.



-3	-4	-38	-26
-31	48	23	-11
-32	82	48	-39
-28	-14	-36	-40

Fig. 3.2: Percentage traffic increase of each router using BRAR (average across PARSEC benchmarks). This utilization difference leads to more than 2× divergence in NBTI-induced performance degradation.

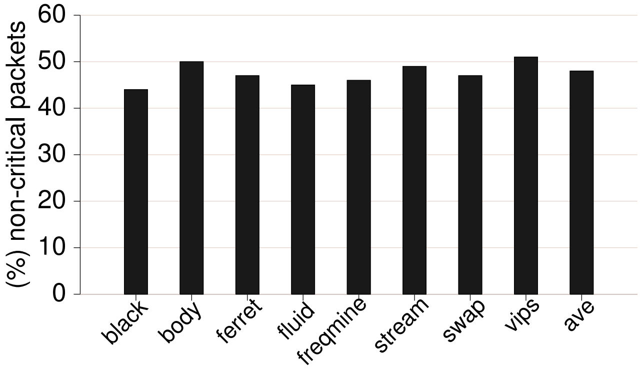


Fig. 3.3: Percentage of non-critical data packets routed through the buffered routers.

3.2.1 Wearout Monitoring System for NoC Routers

To be able to guide the aging-aware routing algorithm, the WMS profiles the extent of degradation in each router. The WMS circuit shown in Figure 3.4 augments all pipeline stages of a router. As the performance degradation of a router is dictated by the worst case delay degradation in any pipeline stage, the proposed monitoring system measures the maximum delay degradation across all paths in different pipeline stages.

Within a stage, the WMS uses a multiplexer to estimate the delay of all n paths in a combinational logic. The control unit in Figure 3.4 alters the multiplexer select signal in each cycle to choose which path to measure. Then, a series of m cascaded delay buffers (db_1, db_2, ..., db_m) sample the signal at equal time intervals. The state transition captured at the output of each delay buffer provides an estimate of the delay of the path. Finally, the comparator selects the maximum delay degradation among the n paths over a span of n cycles. The WMS measures the wearout factor (WF) as

$$WF_{router} = \max(wf_1, wf_2, ..., wf_N), \tag{3.1}$$

and

$$wf_i = \max(wf_{p1}, wf_{p2}, ..., wf_{pn}), \tag{3.2}$$

where wf_1, wf_2, ..., wf_N are the wearout factors for N stages of the router micro-architecture, and wf_{p1}, wf_{p2}, ..., wf_{pn} are the wearout factors of the n paths in a single stage i.

Implementation Overhead: Since NBTI aging is a slow process, a slow sampling period for WMS (e.g. 1 out of 10^{10} times at 1 GHz) is satisfactory [48]. Thus, the WMS component is rarely activated and is clock-gated majority of the time. Also, the WMS measurement is accessed in parallel with the pipeline stage avoiding any impact on the performance. Using a well known NoC router model [49] as a baseline, the implementation shows 1.2% and 0.047% area and power overheads using a 45nm TSMC standard cell technology targeted at 1GHz.

The propagation of WF to adjacent routers is done through the flit link network, by triggering a dedicated multiplexer to latch the WF vector. Given the sampling rate, this WF propagation has minimal effect on system performance.

3.2.2 Criticality-Driven Path Selection

The proposed criticality-driven routing incorporates two major design considerations: (a) criticality of the incoming packet; and (b) WF that dictates the current aging. A maximum threshold for deflecting non-critical packets is established, defined as DFL_{Max}. Subsequently, based on the aging degradation in a router, the deflection rate of each router is pro-rated.

Integrating Criticality in Routing: To drive the deflection logic in routing path selection, the source router adds a single-bit to store the criticality in the header flit of

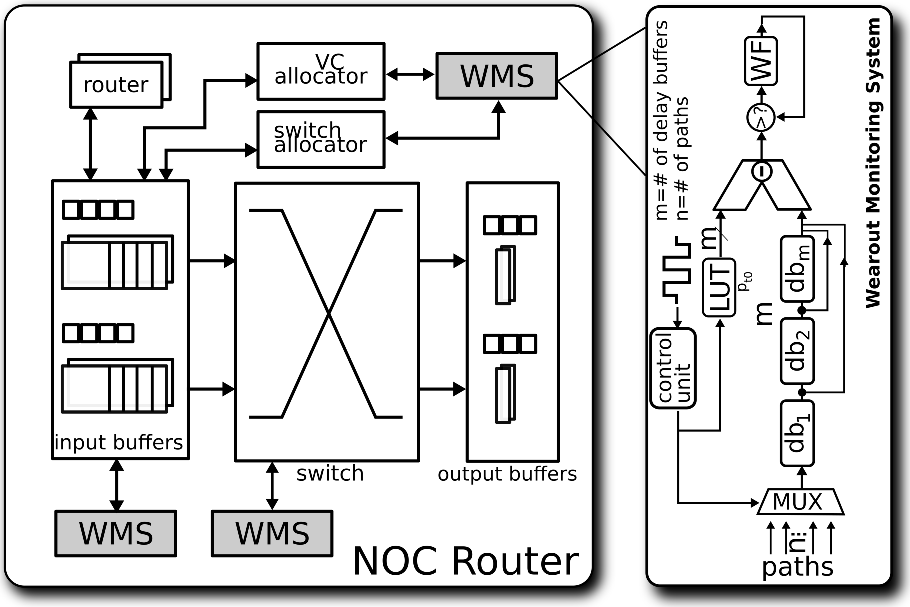


Fig. 3.4: WMS circuit. Each path delay is sampled through a buffer sequence and compared with the reference delay to calculate the WF.

every packet. All intermediate routers peek into this criticality bit to select different routing paths based on criticality.

Integrating Wearout Monitoring: Different routers can undergo different aging degradation based on their utilization history. In a given router, the WF provides its current aging degradation. Table 3.2 shows the pro-rating scheme used in this work. For example, a router with a WF of 0.8 will deflect 25% of all non-critical packets, assuming DFL_{Max} is 0.5. At every sampling interval of the WMS, the WF will be sent to adjacent routers to communicate the degradation of a particular router and a corresponding link. Each router stores the WF of four adjacent routers (North, South, East, West) in dedicated WF registers.

Deflecting Non-Critical Packets: For every incoming flit in a router, the deflection logic uses the WF and packet criticality information to determine whether the packet will be sent in the direction of the pre-established path or deflected away from the buffered router. For a bufferless router, this task is accomplished through a multiplexer and a selection logic. For a buffered router, an additional entry in the routing table is added corresponding to the possible deflection paths for each output port. For instance, an output in the North direction can be deflected to East or West if its coming from the South input. This logic is achieved by using a 4-bit XOR of the number of ports (N,S,E,W) and the ports used for input and the desired output. Since there can be multiple deflection paths, the one that has no pending flits in the output buffer is used. For ties, the first port is always used using a standard priority encoder.

3.3 Aging Model for an hNoC

In this section, an aging model for a typical *hNoC* is derived. The said aging model is used to calculate the additional delay experienced by flits due to NBTI degradation in heavily utilized routers and links. Unlike homogeneous networks, the impact of the presence/absence of buffers in some of the routers in an *hNoC* must be considered. This section first discusses the effect of NBTI aging on routers and links, after which the methodology to estimate their collective aging effect on *hNoCs* are explained.

3.3.1 NBTI Impact on Routers and Links

Router Effect

The NBTI effect causes components in an NoC to experience stress and recovery phases. The long-term effect on such components is a degradation in its threshold voltage, which

Table 3.2: WF-based deflection estimation.

Wearout Factor Range	**Scheme**
0.0 - 0.50	$\frac{1}{8} \times DFL_{Max}$
0.50 - 0.75	$\frac{1}{4} \times DFL_{Max}$
0.75 - 1.00	$\frac{1}{2} \times DFL_{Max}$
1.00 - +∞	$1 \times DFL_{Max}$

in turn worsens the delay of basic gates, leading to system failure. The change in V_{th} in an NoC router for an aging period of t seconds is

$$\Delta V_{th-router} \approx \left(\frac{n^2 K_v^2 \alpha C t_1 t}{\xi^2 t_{ox}^2 (1-\alpha)} \right)^n . \tag{3.3}$$

All associated parameters can be referred in the work of Bhardwaj et al. [50].

To translate this change in V_{th} to timing degradation for NoC *routers*, the first-order Taylor expansion is used to estimate timing increase as done by Chang [51]. The delay is modeled as a Gaussian distribution perturbed around its normal value. The $(\mu + 2\sigma)$ value of this delay is then taken from the Monte Carlo simulation results and assigned as the router delay in the architectural simulation.

Link Effect

For links, NBTI affects the threshold voltages of its repeater circuits, which leads to a higher drive resistance [52]. The additional resistance further degrades the parasitic delay of the NoC interconnects. The model for ΔV_{th} in links is taken from the work of Wang et al. [53] and is equivalent to

$$\Delta V_{th-link} = b\alpha^n t^n . \tag{3.4}$$

Parameters α and t are the switching probability and aging time in seconds, respectively, while n and b are fitting parameters [54]. The switching probability of each link is taken from the profile of the benchmarks used. Using the $\Delta V_{th-link}$, the additional delay for each NoC link is then calculated using an RC model from [52]. Subsequently, the corresponding link delays (in conjunction with the router delays) are also used in the simulator for a system-level evaluation of the NBTI effect.

3.3.2 NBTI Effect on Heterogeneous NoC

In an *hNoC*, some routers are buffered and some are bufferless. Since the NBTI effects are much more pronounced in sequential circuits, buffered routers are more susceptible to aging degradation as compared to bufferless routers. Moreover, buffered routers are positioned in the network such that a majority of the flits will traverse through them, further increasing the gap between the degradation rate of the two kinds of routers. To prove this phenomenon, an experiment is conducted to measure the increase in delay in *hNOC* routers caused by NBTI aging degradation.

The setup for this experiment uses Synopsys HSPICE, Predictive Technology Models [55], long-term NBTI degradation model [50] and static timing analysis of an *hNoC* derived from an open-source NoC Router [49]. The steps are as follows:

1. The effect of NBTI aging and process variation in basic logic gates are studied. Monte Carlo HSPICE simulations (10K sample) are run to get a statistical performance distribution of each gate.

2. The buffered and bufferless versions of an open-source NoC Router are synthesized to obtain a netlist for each router.

3. Using the netlist in Step #2, a statistical timing analysis is conducted to find various critical paths in the router, and their corresponding delay distributions using data from Step #1 (NBTI effect and process variation) and taking into account the diverse utilization among the routers.

Table 3.3 shows the result of this experiment, where the buffered routers experience 2× more degradation compared to the bufferless ones. Hence, in order to increase NoC lifetime, a routing algorithm should aim to minimize buffered router utilization by deflecting non-critical packets in order to minimize performance impact.

3.4 Methodology

This section discusses the simulation infrastructure used to evaluate aging of an NoC. Doing a lifetime simulation (5-10 years) to assess aging impact is computationally prohibitive. Instead, a methodology is created to accurately capture the long-term aging impact in NoCs while keeping a tractable simulation time. This section first discusses the NoC

Table 3.3: Delay distribution for buffered (high utilization) and bufferless (low utilization) routers in BRAR.

Router Type	μ	σ	Worst case delay[1]
Buffered	0.84	0.0304	0.931
Bufferless	0.75	0.0130	0.789

[1] worst case delay is calculated as $\mu + 3\sigma$

model with respect to how its lifetime is evaluated (Chapter 3.4.1). Then, the cross-layer framework is explained. The framework consists of several tools ranging from architectural down to circuit-level simulators in order to accurately assess long-term degradation in NoCs (Chapter 3.4.2).

3.4.1 NoC Reliability Evaluation

The evaluation of network-wide reliability encompasses several layers in the hierarchy. Evaluation starts at the level of a NoC running a real-world application and traverses different design levels (modules, gates etc..) down to the transistor level. Figure 3.5(a) shows this hierarchy along with supplemental information. Each hierarchy has the following roles:

- **Network-On-Chip:** The NoC is modeled in the context of a 16nm technology [55]. The NoC is composed of 16 nodes connected by 16 high-speed virtual channel routers running real-world traffic benchmarks. As NoC system utilization plays a large role in aging [19], all traffic patterns are accurately captured using an architectural simulator. All simulation statistics are then propagated further down the hierarchy for further evaluation.

- **Router:** The router has two equivalent models, one for the architectural simulation and a circuit-level model for per-module power evaluation. The architectural model is embedded in Booksim [56] and emulates a router with 6 virtual channels, 8 input and output ports each, 5 buffers per virtual channels, and a 3-stage speculative pipeline

(Route Calculation, Virtual Channel Allocation, Switch Traversal). Power evaluation uses DSENT [57] that models the router as a collection of modules (crossbar, buffers, allocators etc..). DSENT is configured to use the same parameters as that of the architectural model. The power information will be used for NBTI analysis.

- **Modules:** The router modules are modeled as a collection of gates. This work is interested in the accurate evaluation of HCI aging in the crossbar circuit as it is the module that dictates cycle time [13]. As such, actual bit patterns are collected during architectural simulation to accurately capture the switching activity of different modules. The switching activity will dictate the stress that a transistor has undergone [11].

- **Gates:** Gates are modeled as a collection of transistors connected together. Not all switching transitions cause HCI degradation, therefore the experiment only considers the pertinent transitions of the most susceptible circuit in the NoC to reduce simulation time [58].

- **Transistor:** At the transistor level, the aging degradation is modeled as a combination of both NBTI and HCI effects. NBTI, TDDB and EM degradation are accelerated by large temperatures [59] while HCI is manifested from excessive switching activity. The information gathered from the higher levels in the hierarchy are utilized and a threshold voltage (V_{th}) degradation is calculated using the aging models from the works of Srinivasan et al. [60] and Wang et al. [61]. Both these mechanisms cause accelerated Vth degradation, decreasing timing slack, and eventually causing failure.

3.4.2 Lifetime Simulation

Detailed lifetime reliability simulation of a NoC is a computationally intensive task because aging-induced degradations take years before manifesting as timing errors. In this work, a technique is developed to accurately approximate the lifetime aging impact of running different benchmark programs on a NoC.

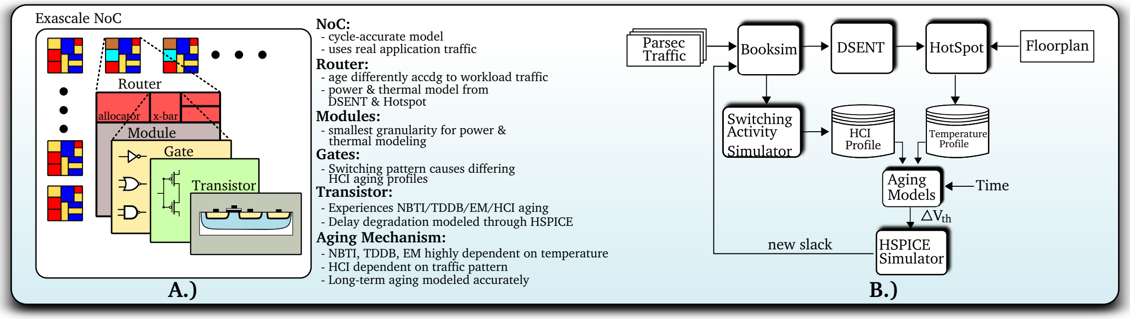

Exascale NoC
Router
allocator
x-bar
Module
Gate
Transistor
NoC:
- cycle-accurate model
- uses real application traffic
Router:
- age differently accdg to workload traffic
- power & thermal model from DSENT & Hotspot
Modules:
- smallest granularity for power & thermal modeling
Gates:
- Switching pattern causes differing HCI aging profiles
Transistor:
- Experiences NBTI/TDDB/EM/HCI aging
- Delay degradation modeled through HSPICE
Aging Mechanism:
- NBTI, TDDB, EM highly dependent on temperature
- HCI dependent on traffic pattern
- Long-term aging modeled accurately
A.)
Parsec Traffic
Booksim
DSENT
HotSpot
Floorplan
Switching Activity Simulator
HCI Profile
Temperature Profile
Aging Models
Time
ΔV_{th}
new slack
HSPICE Simulator
B.)

Fig. 3.5: Reliability-aware framework.

The simulation framework that used in this study are composed of several tools. This work uses Booksim 2.0 [56] and Netrace 0.9 [62] for accurate traffic modeling without incurring the overhead of a full-system architectural simulation. Power and Temperature modeling of different NoC modules are done using DSENT 0.91 [57] and HotSpot 5.02 [63]. An in-house switching activity analyzer is also implemented to observe HCI stress of transistors. Lastly, Synopsys HSpice is used to evaluate timing degradation of NBTI and HCI-induced aging degradation.

The reliability framework is built as a closed-loop system with each iteration equivalent to one (1) month of wall clock time. As such, a simulation of N years would take N×12 iterations. The steps involved in each iteration are as follows:

1. Netrace is integrated into Booksim in order to run real-world applications (PARSEC) in the NoC. This allows the framework to run real benchmarks without incurring the overhead of full system simulation.
2. After the Booksim run, the data dump and per-router utilization is used to evaluate NBTI, TDDB, EM and HCI V_{th} degradation. The router utilizations are fed into DSENT to obtain per-module power statistics while the data dump are used by the switching activity analyzer to analyze HCI impact.
3. DSENT's power outputs are then used as input to HotSpot 5.02 along with a minimalistic floorplan (also from DSENT) to obtain the steady state temperature profile of each router. Meanwhile, the switching activity analyzer outputs the HCI profile of the representative circuit evaluated [49]. Note that only a small circuit (crossbar)

is used as the switching activity analysis is a computationally intensive task. This approach is also used by Oboril et al. [64].

4. The aging models [60,61] take as input the current time, the switching and the temperature profile to calculate the aging-induced V_{th} degradation. The change in V_{th} is then annotated in the HSpice simulation and is used to simulate for the new slack of each router.
5. The new slack values are then annotated back to the Booksim simulator and used by the routing algorithms to calculate the new paths with minimal aging impact. The *time* variable for aging analysis is also incremented. This process is repeated until the end of lifetime of the chip (i.e. a slack threshold is reached). For performance evaluation, the delay values at the end of the lifetime are used by annotating them in the simulator.

3.5 Results

This section evaluates the reduction of performance overheads brought about by the use of the proposed criticality-based aging-aware routing algorithm. The baseline configuration used is an *hNoC* architecture that has not experienced any aging-degradation. Five different schemes are used for comparison purposes, as described next.

3.5.1 Routing Schemes

The capabilities of the proposed routing algorithm are shown by comparing it with the state-of-the-art in *hNoC* routing, the BRAR algorithm. The BRAR algorithm [15] seeks to route flits towards the buffered routers. BRAR does not model any aging-awareness and criticality of the packets. This work evaluates four different schemes within the proposed algorithm, varying in their deflection threshold for non-critical packets DFL_{Max} (see Chapter 3.2.2).

1. **S25:** At most, 25% non-critical packets can be deflected away from the buffered routers ($DFL_{Max} = 0.25$).

2. **S50:** At most, 50% non-critical packets can be deflected away from the buffered routers ($DFL_{Max} = 0.5$).

3. **S75:** Up to 75% non-critical packets can be deflected away from the buffered routers ($DFL_{Max} = 0.75$).

4. **S100:** All non-critical packets can be deflected away from the buffered routers ($DFL_{Max} = 1.0$).

3.5.2 Performance Analysis

To show the effectiveness of criticality-based aging-aware routing in mitigating aging degradation effects, this work evaluates the impact of aging (45nm, 7 years) on four performance metrics with respect to the baseline system.

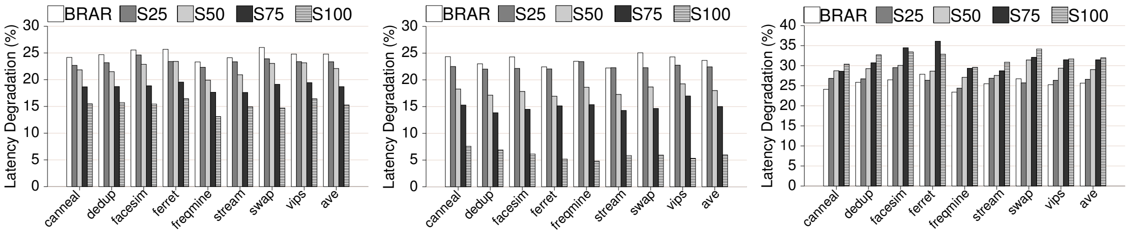
- **Overall Network Latency** - Figure 3.6(a) shows the latency degradation of all schemes. Across several PARSEC benchmarks, the schemes reduce the aging effect on latency by 6–38%. The proposed scheme achieves lower utilization in buffered routers, thwarting the aging effect substantially.

- **Critical Packet Latency** - As the schemes relieve the burden on buffered routers, the latency degradation for critical packets (Fig. 3.6(b)) decreases even further due to reduced aging in buffered routers present in their path. For example, the scheme *S100* achieves 75% improvement in critical packet latency compared to BRAR.

- **Non-Critical Packet Latency** - Subsequently, because the non-critical packets are deflected to a non-optimal path, they are showing an opposite trend of increasing latency degradation (Fig. 3.6(c)): ranging from 4–25% across different schemes. In two cases (facesim and ferret), S75 has more degradation compared to S100, as based on the intrinsic traffic patterns in these benchmarks and sporadic congestion, eagerly bypassing the buffered routers marginally improves the latency in S100.

- **Performance Comparison:** Figure 3.7(a) shows the performance degradation of different schemes under aging. Reduced utilization in the central NoC components substantially mitigates their aging-induced performance degradation, enabling superior system performance. For example, *S100* achieves an average of 53% reduction in aging impact compared to the BRAR across these benchmarks.

- **Energy Delay-Product per Flit (EDPPF):** Figure 3.7(b) shows the EDPPF degradation of all schemes. Except for S25, all of the schemes have lower EDPPF compared to BRAR. S75 and S100 show 16.5% and 29.3% EDPPF improvement, respectively. The reason S25 has worse EDPPF compared to BRAR is that its delay improvement is not enough to compensate for the increase in energy incurred by routing longer paths when packets are deflected.

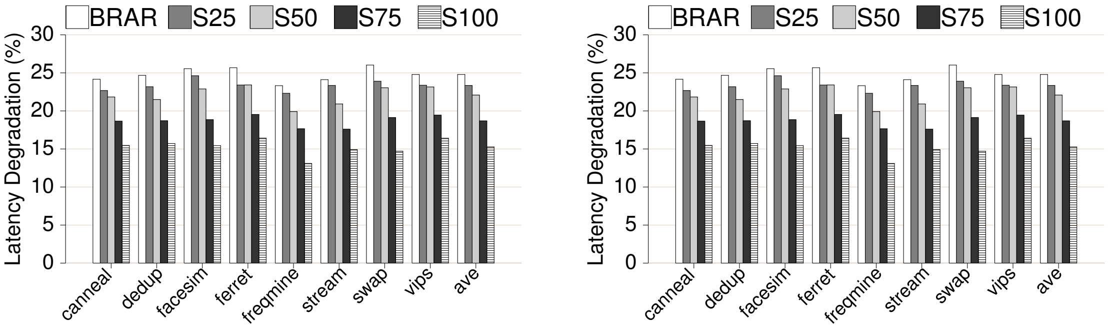


(a) All flits.

(b) Critical flits.

(c) Non-critical flits.

Fig. 3.6: Latency degradation of different types of flits in *hNoC* (lower is better).



(a) Performance degradation.

(b) EDPPF Degradation.

Fig. 3.7: Performance and EDPPF degradation (lower is better).

Chapter 4

HCI-Tolerant NoC Router Microarchitecture

This chapter discusses microarchitectural techniques created in this work to combat HCI aging-degradation in NoCs. The schemes presented operate under a common theme of manipulating the switching activity in order to level out the HCI wearout in the system.

4.1 Background

This section introduces the HCI model used in this work that correlates threshold voltage degradation with the time spent by a transistor in stress.

HCI occurs when a carrier overcomes the potential barrier between silicon and the gate oxide and leaves the channel. A portion of the carriers (hole/electrons) that leave the channel are deposited into forbidden regions in the transistor such as the gate oxide. Throughout a transistor's lifetime, these deposited carriers change the conductive properties of the transistor and ultimately lead to degradation of the threshold voltage (V_{th}), drain saturation current (I_{on}) and transconductance (Δg_m).

The HCI effect on the transistor parameters described above can be modeled as a power-law with respect to the stress time (t) [65,66]. The model for Δg_m can be referred in the work of Bravaix et al. [66]. Only the V_{th} model is discussed as the one for I_{on} is similar. The degradation in threshold voltage is described as

$$\Delta V_{th} = A \cdot t^n, \tag{4.1}$$

where A and n are technology dependent parameters. Parameter n has been widely accepted as ∼0.5 over a wide range of processes [67]. Parameter t is the time the transistor is under stress, while A is derived as

$$A = \frac{q}{C_{ox}} K \sqrt{C_{ox}(V_{GS} - V_{th}} \cdot e^{\frac{E_{ox}}{E_0}} e^{\frac{\varphi_{it}}{q\lambda E_m}}. \tag{4.2}$$

All relevant parameters in Equation 4.2 can be obtained from Wang et al. [61]. The stress time of a transistor is derived from the transition density and the pertinent transitions, since not all inputs that cause switching have a significant contribution to HCI aging [58]. A brief background of how pertinent transitions are estimated is shown in Section 4.4.3.

4.2 Motivation

This section discusses the motivation of the need for HCI-aware design of components in an NoC router. First, major reliability concerns in the datapath of an NoC router are examined. Second, the framework for holistic HCI-aging analysis of the crossbar circuit is explained in detail. Lastly, this section shows the results, which demonstrate the need for HCI-aware techniques in the design of resilient NoC routers.

4.2.1 HCI Degradation in the NoC Crossbar

Massively parallel programs running in the many-core use the NoC as an interconnect fabric due to scalability demands. Processors communicate with each other through messages sent as packets in the NoC. Since on-chip wiring is abundant, a lot of these packets that were previously sent over narrow off-chip buses now cannot fully utilize the whole channel bandwidth available. Coupled with the fact that most data sent through the network are narrow width [68], this trend leads to uneven sensitization of transistors, eventually causing unbalanced HCI degradation across the channel.

The crossbar switch is at the heart of the communication infrastructure in an NoC router, largely dictating the cycle time [13]. There are three critical reliability issues in an NoC crossbar. First, the gate-level activity in a crossbar is only concentrated in a very few bits of the channel width, due to the bit patterns being sent. This asymmetry causes unbalanced HCI degradation. Second, since most upper bit transistors do not switch and only maintain their values, they can undergo NBTI degradation. Third, since the crossbar is a wide circuit with a shallow logic depth (Table 4.1), minor delay variations caused by both HCI and NBTI will have a profound effect on its overall critical path delay.

4.2.2 Aging Analysis Framework for the NoC Crossbar

Figure 4.1 shows the methodology employed in assessing HCI degradation in the crossbar circuit. The cross-layer approach comprises system-level simulation of 16-thread parallel programs and their gate-level HCI degradation in a crossbar circuit. Since HCI depends on switching activity, the switching activity of each gate is acquired by capturing cycle-by-cycle actual data values traversing the crossbar. The setup then evaluates the overall degradation effect for each transistor in the circuit using the model discussed in Section 4.1. However, using real-world applications to assess gate-level degradation is a computationally intensive task. As such, the framework adopts several important steps to efficiently avoid long simulation times, while still providing a holistic analysis of HCI aging effect.

First, multiple sample points are picked in different phases of execution of the program. The sample points are chosen according to traffic intensity in the NoC. Each sample phase contains about 1 million flits. Second, the simulation setup is run (Section 4.4) and the traces of data traffic are taken at the specified points. Third, the data traces are fed to an Open Source RTL Verilog model of a 16-core NoC to gather cycle-by-cycle inputs in the crossbar circuit. Lastly, the novel HCI Aging Analyzer Framework (Section 4.4.3) is utilized to analyze degradation in the circuit.

4.2.3 Results

Logic Depth Analysis

Table 4.1 shows the results for the logic depth analysis performed on major circuits from NoC and processor systems. The following circuitsa are compared: crossbar switch from a NoC, the Arithmetic and Logic Unit (ALU), the memory address generator and the issue queue selector of a Fabscalar core [69]. Among all these modules, the crossbar has the shallowest logic depth that can be 10× lower than the other circuits. This characteristic makes it more susceptible to aging as there is little chance that a different signal path can hide the delay incurred by degraded transistors. Thus, it is important to implement efficient aging mitigation techniques in the crossbar data path.

HCI Degradation Results

Figure 4.2 shows the switching activity data in the crossbar circuit. The x-axis shows the percentage of gates while the y-axis shows its accumulated switching activity as a percentage of the total activity. Ideally, a 1:1 ratio between the percentile gates and the switching activity is optimal for HCI aging (i.e. a straight line with a 45° slope). However, it can be seen that on an average, only 25% of the gates account for 75% of the total switching activity. This large asymmetry leads to unbalanced HCI degradation between different parts of the circuit and can accelerate failure of NoCs before their rated lifetime.

The corresponding clock cycle degradation of a NoC router (22nm, 7 years) as a result of the unbalanced HCI degradation are also shown in Figure 4.3. From this data, *swaptions* experiences the most clock cycle degradation at 10.51%, while *canneal* has the least at 8.99%. This trend can also be verified from Figure 4.2, where swaptions (left most curve) has the most concentrated switching activity among all programs.

Both the results above show that the inherent imbalance in switching activity caused by data patterns sent over the network causes non-uniform HCI aging in the crossbar circuit. This asymmetrical aging causes some path delays to increase disproportionately and will eventually lead to premature router failure. In the succeeding sections, the proposed designs

Table 4.1: Logic depth of various modules.

Circuit	**Logic Depth**	**# of gates**
Crossbar Switch	4	5760
64-bit ALU	46	4728
Address Generator	43	491
Issue Queue Logic	33	189

are discussed, which primarily shifts the switching activity from one part of a circuit to another in order to slow down HCI degradation and balance aging impact.

4.3 Design Overview

This section discusses the proposed techniques for mitigating HCI effect in the router crossbar. The techniques aim to balance HCI degradation by distributing the switching activity. Four microarchitectural techniques are explored: *Bit Cruising (BC)*; *Distributed Cycle Mode (DCM)*; *Crossbar Lane Switching (CLS)*; and *BCCLS* that is a combination of schemes BC and CLS. Apart from DCM, all of the schemes involve minimal modifications at the front-end of the router and do not affect the critical path of the pipeline (crossbar traversal).

4.3.1 Bit Cruising (BC)

Bit Cruising swaps the different portions of the data being transmitted in the crossbar. This technique is largely motivated by two properties of the programs. First, most data traversing the NoC do not occupy the full channel width of the network because most data in the cache line are aggregated at the lower bits. In some cases, all data bits are actually zero. Recent works have also exploited this characteristic by compressing flits and sending only those that have important data [68]. Second, control requests, while being sent as a single flit also do not store information in the most significant portions of the channel as the routing information can fit in the first few bytes of the whole channel. In the setup used in this study, the control flit only utilizes 25% of the channel width, leaving the remaining 75% constant. Together, these two characteristics radically lower the switching activity in certain bits while emphasizing others.

To prevent this asymmetry in HCI degradation, the data being sent across the network must be such that the switching activity across the channel is distributed. By passing different data values each time a gate is used, it will balance the switching activity and hence also uniformly degrade all gates. This is the primary working philosophy of Bit Cruising, where highly changing bits are being *cruised* around the channel. The Bit Cruiser circuit is

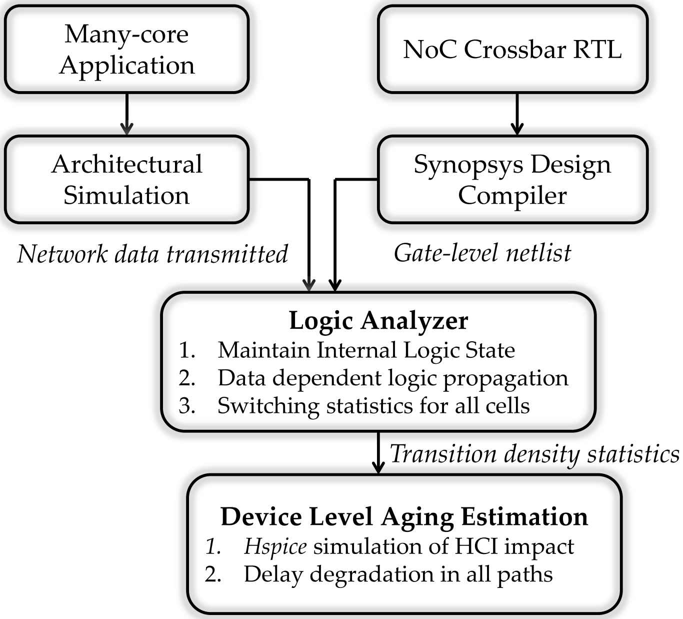

Many-core Application
NoC Crossbar RTL
Architectural Simulation
Synopsys Design Compiler
Network data transmitted
Gate-level netlist
Logic Analyzer
1. Maintain Internal Logic State
2. Data dependent logic propagation
3. Switching statistics for all cells
Transition density statistics
Device Level Aging Estimation
1. Hspice simulation of HCI impact
2. Delay degradation in all paths

Fig. 4.1: HCI aging analysis framework.

situated in the Network Interface (NI) and **does not add any overhead in the critical path of the pipeline** of an NoC. The functionality and the circuit implementation of the BC circuit are explained in detail in the next section.

Circuit Implementation of BC

Figure 4.4 shows the Bit Cruiser circuit that is responsible for cruising the bits around the channel. Bit Cruising can be implemented at different granularities. However, in this work a granularity of four is used (i.e. the whole channel is segmented into four equal parts) because the most lower quarter of the channel bits have the most activity based on the input traces. The Bit Cruiser circuit takes in as inputs the *channel bits* and a 2-bit *cruise setting*. The *cruise setting* is then used as an input to a 4-to-1 multiplexer in order to reshuffle the bits as desired.

Shown in Figure 4.5 is an example of the effect of bit cruising on channel bits. The *cs* signal in the figure represents the cruise setting. In this example, it is assumed that in each clock cycle, *cs* is increased by one.[1] The shaded circles are used to indicate one segment

[1] Note that the cruise setting can also be altered for larger time granularities.

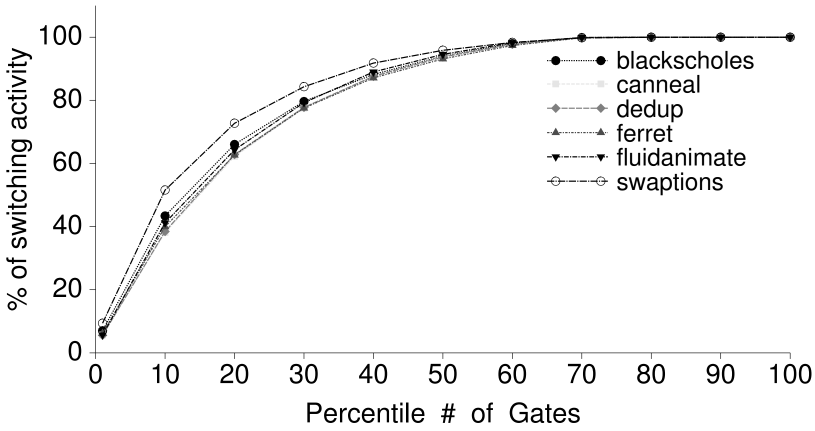


Fig. 4.2: Cumulative distribution function of the switching activity vs gate count.

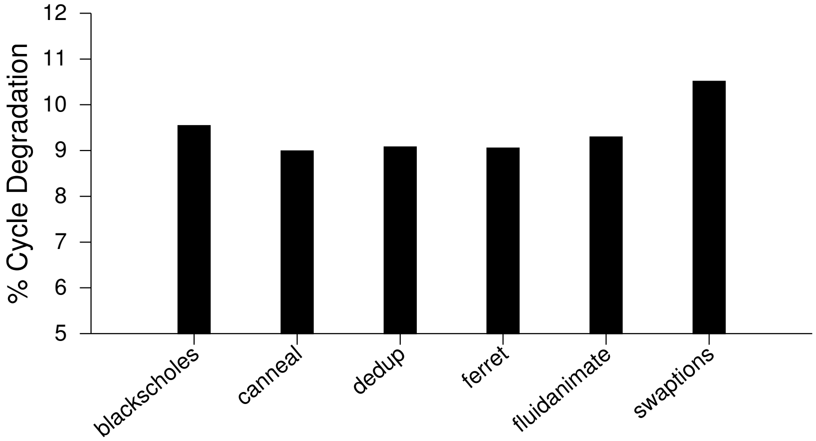


Fig. 4.3: Clock cycle degradation of a 22nm NoC router due to HCI after 7 years.

of a channel. In the figure, when *cs* is equal to zero, the BC circuit output is the same as the channel input (i.e. or when there is no BC circuit at all). When *cs*=1 the lowermost segment is transferred to the uppermost and the second lowermost segment is shifted to the former's place (direction indicated by an arrow). All other segments follow in unison.

Overhead of BC

The circuit in Figure 4.4 will be placed in the Network Interface right before sending the flit to the router of the source node. Since the bits being sent through the network are now jumbled, the router front end must be able to appropriately identify the header flit bits in order to route the circuit correctly. To this end, this work introduces a Routing

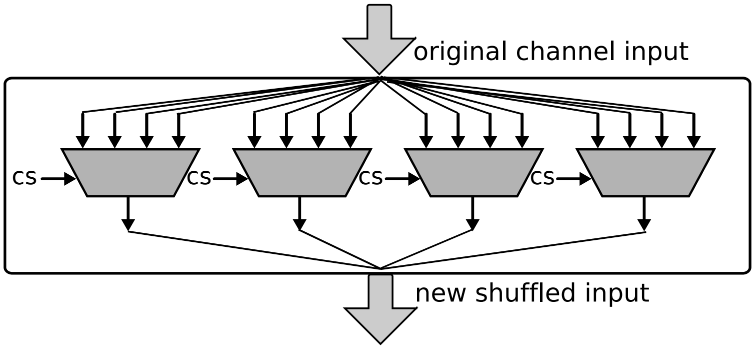


Fig. 4.4: Bit cruiser circuit.

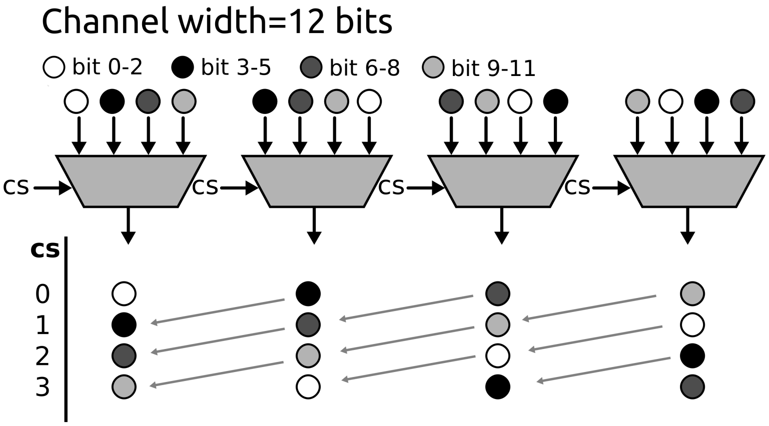


Fig. 4.5: Time lapse example of a bit cruiser circuit for a 12-bit channel. Signal cs is the cruise setting.

Information Extraction (RIE) circuit. The RIE circuit extracts the appropriate bits from the shuffled channel bits and places it in a Routing Information Register (RIR), which will be accessed by the Routing Calculation module in the succeeding pipeline stages.

Figure 4.6 shows the implementation of the RIE circuit. Every time a flit arrives and is about to be written in the virtual channel, the RIE circuit (using the cruise setting information) will determine if the flit is a head flit. If it is, the routing information is latched into the RIR. The RC module in the next pipeline stage will then use the contents of the RIR to route the flit in the corresponding VC. Since there is only going to be one packet in each virtual channel, the overhead for the RIR is minimal. The calculation of RIR's overhead in a typical modern NoC architecture is discussed shortly.

In modern NoCs, the network width is often large as on-chip wiring is abundant and bandwidth is also important. For a 128-bit flit width in an 8×8 network with an algorithmic routing algorithm that uses the number of hops in the x and y directions, the RIR will only require 3 bits in each direction for a total of 6 bits. If there are 5 flits in each buffer, then the overhead is 0.9%. For deeper flit buffers, the overhead further goes down.

4.3.2 Distribution Cycle Mode (DCM)

The Distribution Cycle Mode technique utilizes idle cycles to balance out the HCI degradation of transistors. Most real-world applications spend a considerable time waiting for information from the NoC. Moreover, cache coherence requests are self-throttling or that succeeding requests are not sent unless a reply is received [62]. As such the crossbar spends most time (average of 85% in the setup used in this study) sending no data through the crossbar. This presents us with tremendous opportunities in using reliability techniques without incurring a latency overhead. The second technique, DCM, introduces a new mode of operation for the crossbar circuit during idle phases of execution. In DC mode, the crossbar uses optimized inputs to achieve a near-uniform switching activity across the data channel.

To explain the idea behind DCM, a simple example of a 2-bit NOR-gate in DC mode is discussed. Figure 4.7 shows a 2-bit NOR gate in standard CMOS executing optimized inputs. For each set of inputs, the transition number is indicated along with an arrow in the circuit that indicates the corresponding path that the output signal has taken. For instance, transition 0: inputs A=0 and B=0, switches both the PMOS transistors while transition 1 (A=0, B=1) only switches NMOS B. The key to successfully implementing the DC Mode is to balance out the number of switching transitions that each transistor makes.

In this example, executing transitions 0-2 once gives each transistor a balanced switching count. Note that there are four possible inputs for the NOR gate but only three are needed in order to get a balanced HCI degradation. Being able to accurately determine the needed inputs, rather than exhaustively using all possible, is the key to optimal DCM operation.

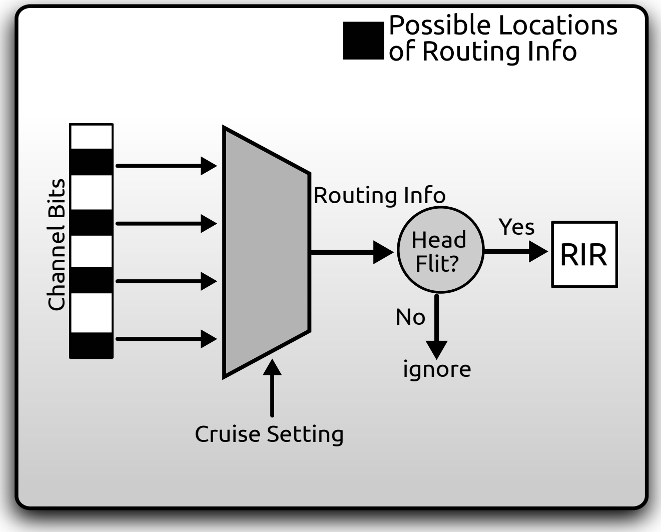


Fig. 4.6: Routing information extraction circuit.

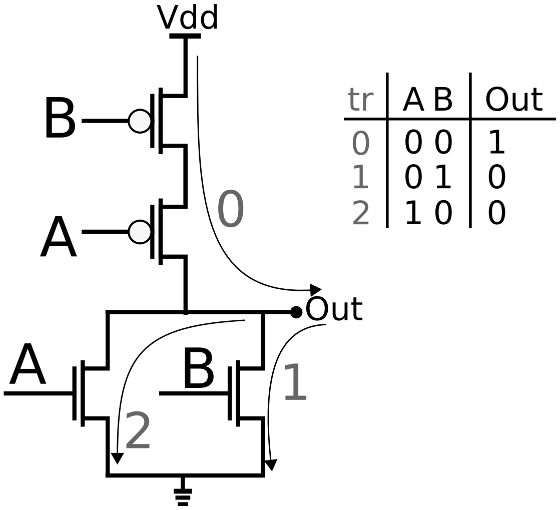
Implementing DCM in the Crossbar

Applying DCM to a big circuit such as a router crossbar poses some major challenges because optimal HCI degradation is only achieved when the inputs are carefully constructed to balance the switching activity. However, despite the enormity of the crossbar, its regular structure allows us to analyze a small subset of the circuit and use the results to optimize the whole component.

There are three key requirements to seamlessly applying DCM while maintaining the correct and unobstructed execution of the NoC Router. They are:

1. **Idle time identification** - To engage the crossbar in the Distributed Cycle Mode, idle cycles must be correctly identified or else the correct value that is supposed to be transferred during the *switch traversal* stage of an NoC is going to be overwritten. This overwriting can corrupt a running program.

2. **Identification of optimal inputs** - The optimal inputs to the crossbar circuit are derived using an offline analysis similar to the one discussed in the previous subsection. This is a one time effort that can be used throughout the lifetime of the NoC router.



tr	A B	Out
0	0 0	1
1	0 1	0
2	1 0	0

Fig. 4.7: Two-bit NOR gate showing the different transitions with respect to inputs.

3. **Feeding mechanism of customized inputs** - The crossbar must have an option of using the inputs provided by the analysis above in order distribute HCI aging in all of its transistors.

In a typical router in an NoC, the crossbar switch has multiple lanes to handle simultaneous demands of multiple inputs to multiple outputs (NORTH, SOUTH, EAST, WEST). As such, when no input port is scheduled to transfer a data to a specific output port in a particular time, that output port (or lane) is considered idle. Thus, correctly identifying the idle cycles of a crossbar depends mostly on the output of the scheduling algorithm of the switch.

The main mechanism to identify idle cycles is already present in any Switch Allocator (SA) implementation as it outputs a schedule of the switches every clock cycle. Figure 4.8 shows an NoC router along with the supplementary logic and components to identify idle cycles and implement DCM. Aside from the main DCM module that serves as the control unit for DCM operation, a lookup table and an additional multiplexer is added for the purpose of storing the optimized values and to have the ability to load them when desired, respectively.

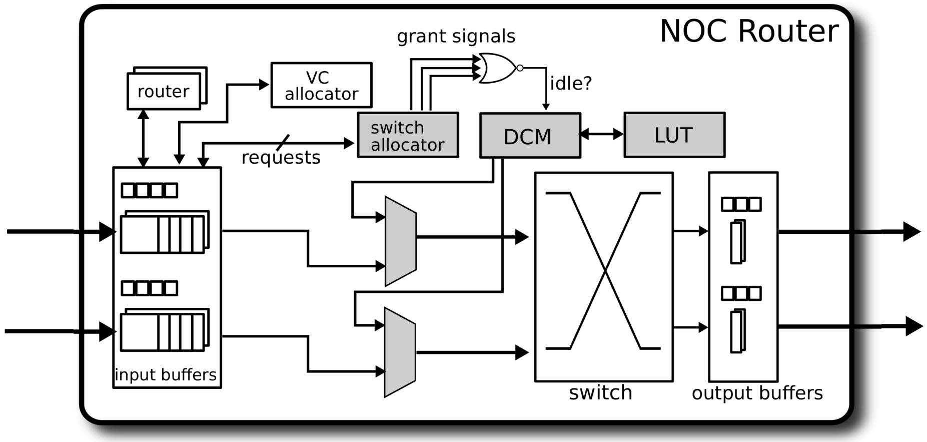


Fig. 4.8: Modified NoC router to accommodate DCM operation.

In each clock cycle, the SA takes in as input the requests of different virtual channels and input ports and gives the permission to specific input ports to use the output ports in the next cycle. If there are no contention of requests, all requests could be permitted to traverse in the crossbar the next cycle. However, if there is, it is resolved based on a scheduling priority. In other cases though, there simply are not enough requests to keep the switch/crossbar fully utilized. When this happens, the DCM module immediately senses this and queries the lookup table and instructs the multiplexer to load an HCI aging-optimized value in the subsequent cycle.

4.3.3 Crossbar Lane Switching (CLS)

The two previous techniques focused on distributing the switching activity across an entire channel of an input port to balance HCI degradation. However, another asymmetrical degradation also occurs in the crossbar lanes that are immune to techniques applied in the channel-level.

This type of asymmetric degradation arises when some input-output pairs are used more than others. This occurence is demonstrated via an example in Figure 4.11 where there are two paths (p0 and p1) that both use the same *East* output port. For instance, if path p0 is used more than p1, then the transistors along the path p0 will be sensitized more and hence, experience more HCI degradation.

The third technique, CLS, is also situated at the front-end of the router pipeline and aims to balance the usage of the crossbar lanes.[2] In the canonical router model, an input port directly forwards flits to the output ports by establishing a physical connection between the two via the crossbar switch. As such, flits coming from the same input port will always use the same crossbar lane to connect to different output ports. However, the introduction of Input Buffers (IB) and Virtual Channels (VC) in modern router architectures decouples this one-to-one association because the flits are first stored in the IB before being transmitted to the output ports. With trivial modifications in the VC allocator and the Route Calculation part of the pipeline, the router can control which crossbar lane an input port will utilize at any given time.

This new allocation and routing policy will now cause the crossbar circuit to use a different path and activation circuit, but still send the same data as if it were coming from the original input port. Thus, the scheme preserves the correctness of the flit and the route.

Similar to the Bit Cruising technique's *cruise setting*, CLS will need a *knob* input to indicate the new mapping between input ports and crossbar lanes.

CLS Implementation

Figure 4.9 shows a logical diagram for a traditional Virtual Channel (VC) flow NoC Router with two input ports and two virtual channels per input port. The virtual channels are used to handle multiple concurrent streams per input port, each waiting for its turn to use the crossbar switch, hence improving the overall bandwidth of the network. In the example, the north input port can only utilize VCs 1 and 2, while the south utilizes 3 and 4. In each clock cycle, all VCs request usage of the crossbar for the succeeding cycle. The switch allocator will then determine a winner and subsequently connect the virtual channel to the desired output buffers.

As discussed in Section 4.3.3, the lanes of the crossbar can undergo uneven degradation when certain input and output pairs are used more. CLS aims to balance this degradation by evenly distributing the paths taken by a flit. Figure 4.10 shows the necessary modifications

[2] a lane is the path taken by an input port to the output port.

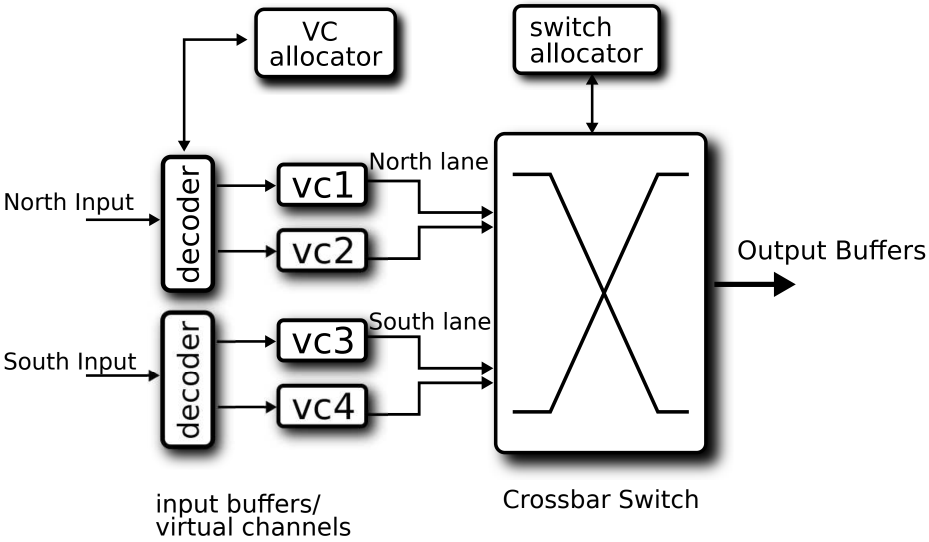


Fig. 4.9: Baseline implementation of an NoC router showing the virtual channels, input ports and the crossbar switch. Output ports and output virtual channels are not shown.

on the NoC Router to be able to implement CLS. Also, the VC allocator must be able to assign any incoming flit to any virtual channel (additional lines in the decoder).[3] As the virtual channels are implemented as SRAM arrays [49] similar to a register file in a processor, there will be no additional logic needed to access the different virtual channels. The only extra logic needed will be for the VC allocator to distribute the flits across the many virtual channels which can be accomplished by a simple counter circuit which is added to the offset of the decoding stage. The Route Calculation (RC) stage will automatically determine the route of the flit since the routing information is stored in the head flit. The RC will then send the SA the appropriate commands, preserving the correctness of the flit and its route.

The light blue line in Fig. 4.10 shows the path taken for a flit arriving at the North input and traversing the South lane of the crossbar. This is made possible by storing the flit in virtual channels 3 or 4 and then informing the SA to use the same channel as input to the crossbar. In summary, an incoming flit uses the same input port, a different virtual channel and crossbar lane, and the same output port.

[3] Note that there are many possible implementations of the Input Buffers. The overhead is analyzed with respect to an open-source RTL implementation of a modern NoC router [49].

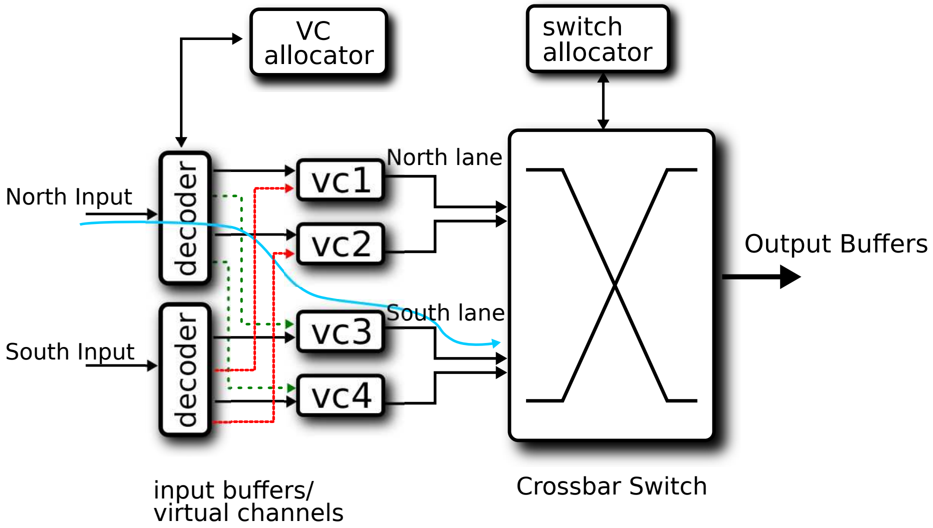


Fig. 4.10: CLS implementation. VC allocator can assign incoming flits to any virtual channel.

4.3.4 Bit Cruising and Crossbar Lane Switching (BCCLS)

The last technique is a combination of the BC and CLS schemes. BCCLS combines both the benefit of switching distribution inside a channel (BC scheme) and the distribution of activity across many channels (CLS scheme). The implementation of BCCLS comes naturally because both BC and CLS tackle different portions of the router circuit. BC reshuffles the data sent through the network while CLS effectively changes the port a flit is coming from by modifying the VC allocation and route calculation.

4.4 Methodology

This section discusses the simulation infrastructure that combines multiple tools across different abstraction layers. The methodology can be broadly classified into three categories: Architectural Setup, RTL and Switching Activity Simulation and HCI degradation analysis using SPICE.

4.4.1 Architectural Setup

The simulation setup is composed of a 16-node mesh system arranged in a 4×4 grid. Each node in the system is composed of 1 processor, 1 L1 Cache and a slice of a system-

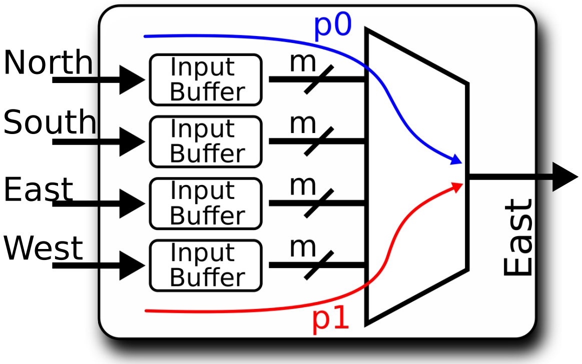


Fig. 4.11: East section of a crossbar switch. CLS works on the **inter**-lane[3] (by changing the path of the data) level while BC works only on the **intra**-lane level (by changing the bit ordering within a path).

shared L2 cache. Each router in the system has seven sets of input and output ports including the ones for the processor and caches. The flit size is configured at 16-bytes (128 bits). A single control request fits in a single flit while data flits needed to transfer a 64-byte cache line are sent in five (4 data + 1 control) consecutive flits. Each processor's L1 and L2 cache sizes are 64kB and 512kB, respectively.

4.4.2 RTL and Switching Activity Simulation

The first step in obtaining an accurate switching activity is to produce real-word data vectors from standard benchmark programs as inputs to the RTL circuits. This work uses the PARSEC [70] benchmark suite (large inputs) running on gem5 [71] to collect data traces. Data traces are collected for the four center most routers in a 16-node mesh.

After the traces are taken, a trace feeder is implemented through a Verilog VPI based functional verification framework called Teal [72]. This module allows us to easily obtain cycle-by-cycle values in any sub-module of the router such as the crossbar.

4.4.3 HCI Degradation Analysis

Using the outputs from the previous step, the logic analysis tool is then used to obtain the transition densities of each transistor (Figure 4.1). All results are post-processed to calculate V_{th} degradation and be able to simulate them in HSPICE to obtain clock cycle degradation data for all paths across different benchmarks. In all of the analysis, the 22nm [55] technology is used with an aging period of 7 years.

Pertinent Transitions

HCI affects a transistor during a switching activity. However, for a reliability evaluation of a VLSI circuit consisting of thousands of transistors operating for years (typically 7-10), accurate HCI degradation analysis using HSPICE takes too long. As such, it has been determined by Kamal et al. [58] that only certain type of transitions in a logic gate generate interface traps in its transistors. Hence, this work only calculates the HCI impact of these transitions, allowing for a practical simulation time. The pertinent transitions of the gates used in the design (INV, NAND, NOR) are listed in Table 4.2, the transitions indicated in the second column and third column induce HCI degradation for NMOS and PMOS transistors, respectively. All these transitions are simulated in order to evaluate their HCI aging impact on the logic gates. Only transitions that affect the transistor near the output node are counted as they contribute the most to HCI [58].

Table 4.2: Pertinent transitions of various gates.

GATE	**NMOS**	**PMOS**
INV	↑A	↓A
NOR	(↑A,$\overline{B}$) ($\overline{A}$,↑B)	(↓A,B)
NAND	(↑A,B)	(A,↓B) (↓A,B)

Aging Framework

In the gate level, HCI degradation is manifested during transistor switching. This work developed a tool to examine the possible HCI impact on all gates of a circuit through extensive logic analysis. The HAAF tool works by taking the input in every clock cycle and propagating the logic in a domino fashion until it reaches the output. During the course of this propagation, some gates will switch while others will not. The transitions in all clock cycles are recorded and are used to calculate the transition density of the gate. Note that all the transition events are post-processed in order to determine if they are pertinent transitions before calculating the transition densities.

Figure 4.12(a) shows a detailed example of how this analysis is done on a circuit with three gates indicated as G_0, G_1, G_2, with initial states as shown.[4] Figure 4.12(b) shows a new set of inputs being fed and denotes the specific gates that will change (highlighted in gray). G_0 and G_2 changes in this cycle while G_1 does not. The transition density (TD_g) of a gate g is calculated as

$$TD_g = \frac{\sum_{n=1}^{x} S_{gn}}{x}, \tag{4.3}$$

where x is the total number of cycles simulated and $S_{gn} = 1$ if gate g made a pertinent transition at cycle n (0 otherwise). The HAAF then uses the transition density to calculate the new V_{th} using the model in Equation 4.1. A new propagation delay t_g is then obtained for each gate g using HSPICE simulation. Note that the framework simulates t_g for all gates and not just the ones in the critical path because the critical path can change depending on the extent of degradation in different parts of the circuit. Finally, the HAAF calculates the new propagation delay (T_P) of the whole circuit as

$$T_P = max(X_0, X_1, ..., X_Y), \tag{4.4}$$

[4] initial states are a result of a previous execution

and

$$X_y = \sum_{g=1}^{H_y} t_g, \tag{4.5}$$

where Y is the set of all paths in the circuit and X_y is the total propagation delay of path y. H_y is the set of all gates in path y.

The process discussed above forms the bulk of the evaluation framework and although it is very computationally intensive, its thoroughness allows us to accurately evaluate the benefits of the architectural techniques at a circuit-level accuracy.

4.5 Results

This section presents the effectiveness of the schemes across different metrics.

4.5.1 Comparative Schemes and Evaluation Metrics

The following five schemes are compared:

- **BASE**: Baseline configuration where the system is unmodified.
- **BC**: Bit Cruising scheme, the channel is divided into four segments and a bit cruiser circuit is placed between the NI and the router.
- **DCM**: Distributed Cycle Mode technique presented in Section 4.3.2.
- **CLS**: Crossbar Lane Switching scheme discussed in Section 4.3.3.
- **BCCLS**: Combination of BC and CLS schemes.

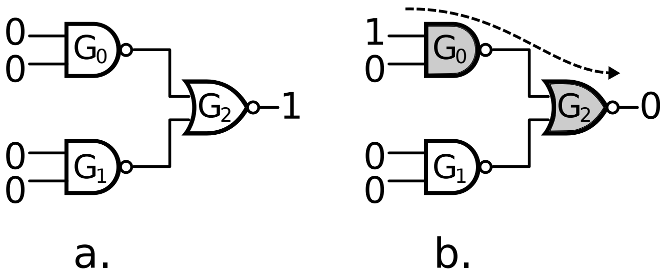


Fig. 4.12: HCI analysis.

These schemes are evaluated in terms of switching activity distribution through Cumulative Distribution Function (CDF) plots, clock frequency degradation, Energy-Delay Product Per Flit (EDPPF) and System Performance. The circuits used to facilitate all these schemes (except DCM) are added in the front-end of the pipeline without affecting the actual crossbar circuit, and as such do not incur any additional timing overhead in the critical path.[5]

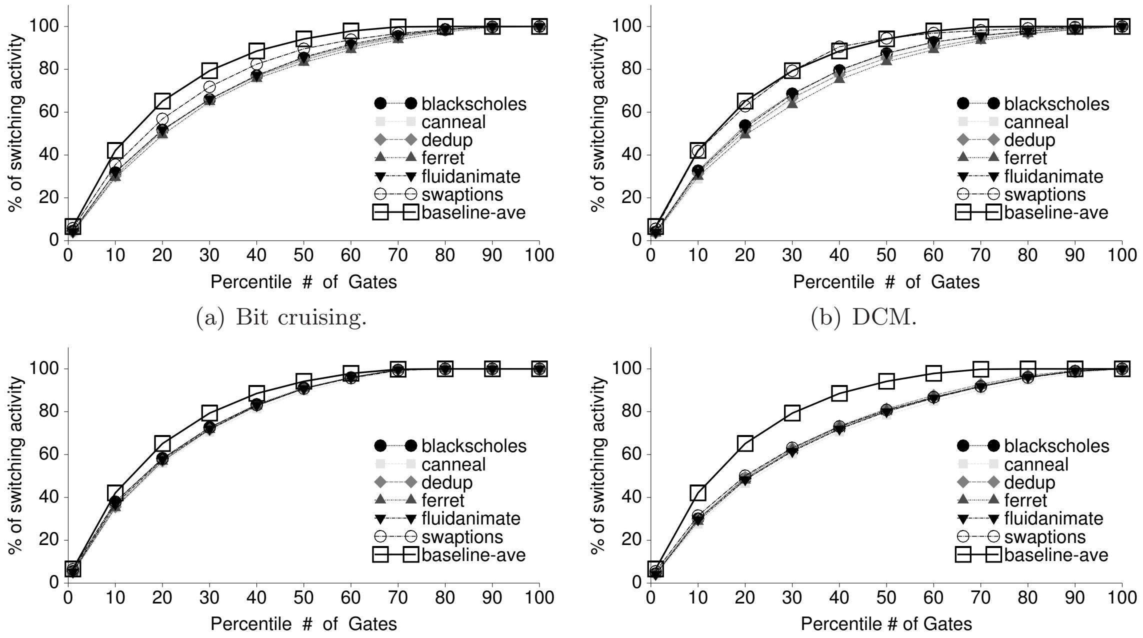
4.5.2 Switching Activity Distribution

CDF plots are shown in Figures 4.13 and 4.14 of the switching activity distribution of all schemes. The average of the baseline scheme is superimposed in each figure. All of the proposed schemes outperform the baseline by having a lower value (y-axis) at any percentile point. Hence, it is evident that the proposed schemes achieved their aim of distributing the switching activity. At an evaluation point of 20 percentile, the best performing scheme (BCCLS in Fig. 4.14) shows 31% less switching activity compared to the baseline.

4.5.3 Clock Cycle Degradation

Figure 4.15(a) shows the cycle degradation for the NoC router at the end of a 7-year aging period using the ASU 22nm predictive technology model [55] operating at 1 Ghz. On an average, the base scheme degrades the clock cycle by 9.4%. The proposed schemes improve this degradation by 20.6%, 0%, 12% and 25.5% for BC, DCM, CLS and BCCLS, respectively. Combining both BC and CLS schemes results in the least amount of clock cycle degradation while DCM provides no improvement from the baseline. As HCI is an unrecoverable degradation [10], any damage done during normal operation cannot be rectified. The difference between DCM and all other schemes is that it is reactive while the others are proactive (preventing aging beforehand). However, DCM improves other aspects of the circuit such as the EDPPF which will be discussed next.

[5] DCM's cycle degradation is taken without the timing of the additional multiplexers so as to show the timing degradation in the crossbar circuit only across all schemes.



(a) Bit cruising.

(b) DCM.

(c) Crossbar lane switching.

(d) BCCLS.

Fig. 4.13: Cumulative Distribution Graph of BC, DCM and CLS schemes. Solid line in each graph is the baseline average (lower is better).

4.5.4 Energy Delay Product Per Flit (EDPPF)

Figure 4.15(b) shows the EDPPF of all schemes. The base scheme is shown as a line at 100%. Most schemes have lower EDPPF compared to the baseline except for some outliers. For the BC scheme, *dedup* and *ferret* have larger EDPPFs while for CLS, *swaptions* has a slightly larger EDPPF than the baseline. Upon further investigation, although BC has helped achieve less degradation and a more distributed switching activity, its dynamic switching activity for benchmarks *dedup* and *ferret* are actually 63% and 30% more

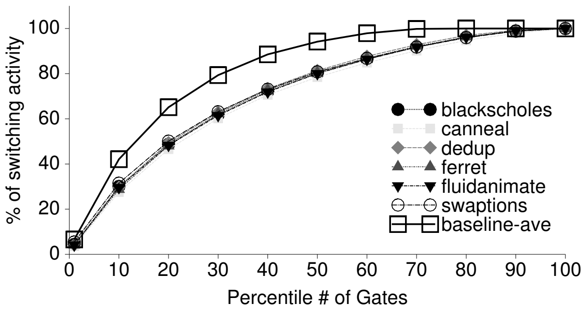


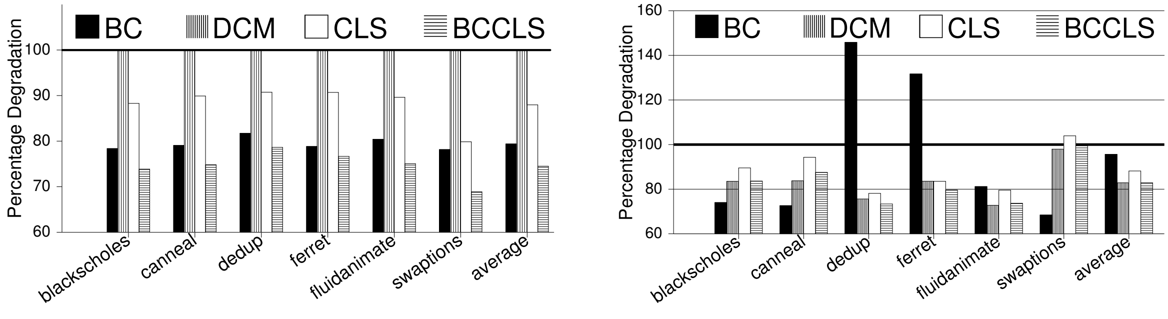
Fig. 4.14: CDF for BCCLS scheme.

compared to the average of all other programs. This unusual activity increase is due to the workload-dependent bit patterns being sent across the network. For *swaptions*, the switching activity for the benchmark is unusually high in all schemes except for BC.

Even though DCM does not provide any improvement in the clock cycle, it provides consistent reduction in EDPPF. This reduction is because optimally aged transistors have higher threshold voltages and will have lesser leakage power. Leakage power cannot be ignored in small technologies such as the one used in this work (22nm). On an average, DCM improves the EDPPF by 18% compared to the baseline.

4.5.5 System Performance

Figure 4.16 shows the overall system performance impact of all schemes relative to the baseline. DCM shows no improvement because it has the same clock degradation as the baseline. On an average, performance degradation is reduced by 9.3%, 8% and 11% for BC, CLS and BCCLS schemes. Maximum is 17.6% for the BCCLS scheme running *ferret*. Overall, the system performance improvement is less than the clock cycle degradation improvement due to the sublinear dependence of clock frequency and performance.



(a) Cycle degradation.

(b) EDPPF.

Fig. 4.15: Router Cycle time Degradation and Energy Delay Product Per Flit. Solid line indicates baseline (lower is better).

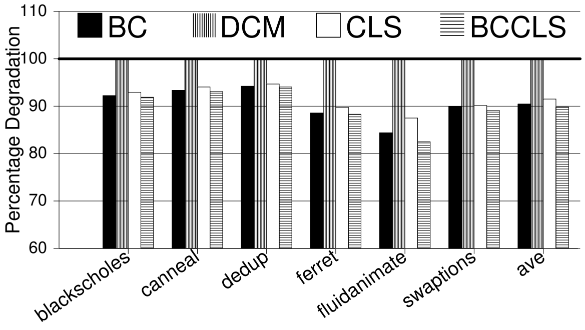


Fig. 4.16: System performance degradation (lower is better).

Chapter 5

Tackling QoS-induced Aging in Exascale NoCs

This chapter explains the work proposed on this study to address QoS-induced aging in machines with hundreds to thousands (i.e. exascale) of NoC routers. First, it is discussed how QoS support in exascale machines can be detrimental to its reliability. Then, a dynamic wearout-resilient routing algorithm is proposed, which combats aging in a scalable manner.

5.1 Motivation

This section presents a quantitative analysis of the threat posed by rapid device-level wearout on an exascale NoC design. In particular, two key sustainability challenges stemming from wearout on these systems are: (a) wearout impact of NoC scalability (Section 5.1.1); and (b) adverse interaction between QoS support and NoC lifetime (Section 5.1.2). Finally, this section briefly outlines the profound significance of the analysis on exascale NoC designs in Section 5.1.3.

5.1.1 Scalability Impact on NoC Wearout

Figure 5.1 shows the effect of scaling on the average router power. As the number of nodes in the system are increased, more power is needed to support the enormous communication bandwidth/traffic demand imposed by more nodes. For instance, at an injection rate of 0.1 flits/cycle, a router in an exascale system (1024-nodes) will have to consume about 2.2× the power compared to a router in a 64-node system. For low network loads, the power consumption difference is barely 20%. Such large power increase leads to higher power and thermal densities, degrading the system reliability.

Figure 5.2 shows the detrimental lifetime impact of wearout on exascale NoCs at 16nm, measured using the *Mean Time to Failure*[1] *(MTTF)*. The NoC is organized as a 2D mesh handling a uniform-random traffic. As the NoC is scaled from 64 to 1024 nodes, the components in the central region experience rising stress from handling a larger number of traffic flows, resulting in a significant MTTF reduction. For example, at 0.06 flits/per cycle injection, the expected lifetime of a kilo-node NoC is 48% less than a 64-node NoC. Furthermore, higher injection rates are increasingly likely on exascale NoCs running big data applications [73]. These applications with massive data footprints also show more concentrated traffic patterns (e.g., substantially more memory traffic than inter-processor communication) [74]. Collectively, these trends can play a havoc in drastically shortening the lifetime of the NoC. For example, doubling the injection rate from 0.04 to 0.08, shortens the lifespan of the 1024-node system by 45%, compared to 16% in the 64-node system.

5.1.2 QoS Support and NoC Lifetime

To uncover the conflict between wearout resilience and QoS support, the GSF QoS policy [8] is used to run a benchmark with heavy memory traffic on an NoC supporting an exascale system with distributed memory controllers. This benchmark broadly represents a big-data application, where the process-memory communication typically dwarfs inter-

[1] To estimate the MTTF, the NoC's delay-degradation is simulated at a month granularity under several major device wearout mechanisms [60, 61], until a threshold value (see Section 5.3) is reached.

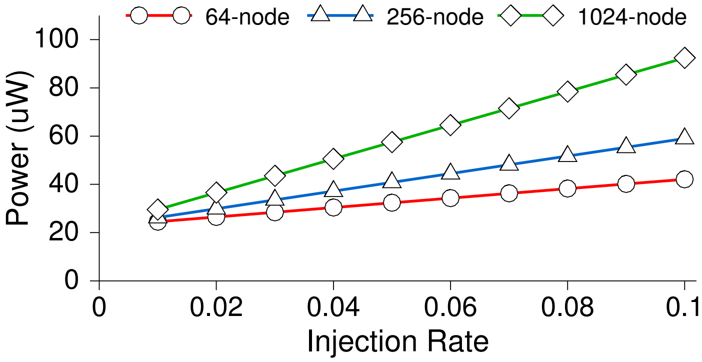


Fig. 5.1: Effect of scaling on average router power consumption (16nm node).

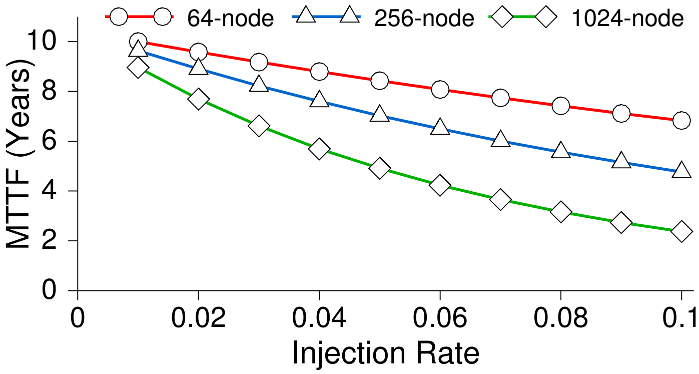


Fig. 5.2: Trend for scaling impact on MTTF (16nm node).

processor communication [74]. Now the question is how does QoS support alter the stress patterns seen in the NoC?

To understand this aspect, *one must recognize that QoS support does not alter the underlying bandwidth or performance offered by the NoC* [26]. Instead, QoS support leads to a change in how various resources are proportioned among various flow demands. For example, a fairness guarantee through QoS support leads to a more uniform distribution of bandwidth among the on-chip flows. This phenomenon, also observed by Lee et al. [8], leads to enable more flows to simultaneously communicate through the NoC, causing a dramatic increase in the power and thermal profile of the system. To explain this concept, a simple example in Figure 5.3(a) is shown, where three nodes A, B and E attempt to send flits to D. Without QoS, nodes A and B can be unfairly treated (only receiving 1/4th of the bandwidth due to contention). However, QoS support fairly distributes the $E-D$ link bandwidth between all three nodes (A,B, and E), resulting in increased power consumption from the greater network activity. Note that the total bandwidth available in the network does not change when supporting QoS.

Such increased power and thermal profile from QoS support causes accelerated wearout in the NoC devices. Figures 5.3(b) and 5.4 show the impact of QoS support on the power and MTTF of the NoCs. At larger network sizes, the difference in power consumption between supporting QoS and not is even more pronounced as there are more traffic flows that need to be supported. The experiments find that the maximum number of flows that has to be guaranteed at a certain level of QoS are 2016, 32640 and 523,776 for 64, 256 and 1024 nodes, respectively. Consequently, QoS support leads to a pronounced detrimental impact on the lifetime of an exascale system. Furthermore, it is observed that adding QoS support at 1024 nodes decreases the MTTF by almost two years or 36% of its rated lifetime.

5.1.3 Significance

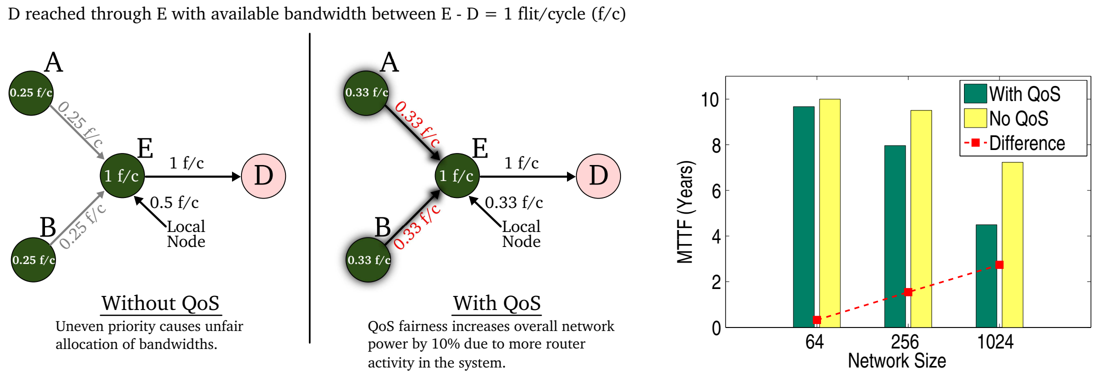
Without a reliability-aware policy in place, the QoS-policy dramatically decreases the average MTTF of an exascale system. However, for large systems, QoS is indispensable to avoid the problem shown in Figure 5.3(a). Then, a key question is how can one enable QoS support while limiting its damaging impact on the NoC lifetime? Thus, there is a need to implement reliability-aware schemes to improve MTTF, while still maintaining guarantees enforced by QoS policies. In this work, a reliability-aware routing algorithm is proposed to balance the lifetime of NoC routers by using network wide aging information to route packets.

5.2 Design of Wearout Resilient Routing in an Exascale NoC

This section describes the design of the proposed QoS-aware wearout resilient routing techniques for exascale NoCs. The proposed three-step approach spans multiple layers to manage wearout degradation in NoCs, while maintaining short term power-performance goals: 1) Sensing: The scheme senses the device-level wearout of routers and links in the circuit layer using a NoC Health Meter (Section 5.2.1); 2) Communication: The technique communicates between the circuit and the architecture layers by propagating the wearout information across the NoC (Section 5.2.2); and 3) Routing: Wearout information in the architecture layer is utilized during NoC routing to dynamically mitigate the effects of aging (Section 5.2.4).

5.2.1 NoC Health Meter (NHM)

To guide the routing algorithm, the NHM profiles the extent of degradation in each router and incoming links. The degraded delay of incoming links are measured as part of



(a) QoS effect on the network traffic.

(b) MTTF impact of QoS support.

Fig. 5.3: Conflicting goals of QoS support and sustainability: although the bandwidth offered by the NoC remains unchanged, different resource usage under QoS causes an accelerated wearout and a shortened lifetime.

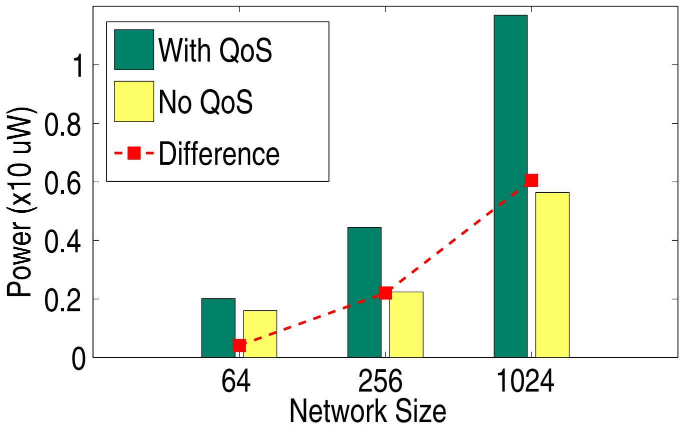


Fig. 5.4: Effect of providing QoS on the average router power consumption.

the first stage of the router (i.e. input buffers). The NHM circuit shown in Figure 5.6 augments all the pipe stages of a router. The proposed meter essentially measures the delay degradation in the combinational circuit between two pipeline registers by measuring the slack in each stage. To this end, this work uses a high resolution all-digital, self-calibrating time-to-digital converter (HR-TDC) consisting of a Vernier Chain (VChain) circuit that has a measurement resolution of 5ps [75]. After measuring the delay degradation of each stage, D_{max} is determined, which is the maximum degradation among all pipe stages.

The HR-TDC is an *in situ* delay-slack monitor consisting of a Vernier Chain circuit with an overall measurement window of 150ps, which is sufficient for timing slack measurements in 2Ghz+ systems. To accommodate process variability, the HR-TDC is calibrated post-fabrication so that each HR-TDC stage is tuned to 5ps increments. To also avoid the expensive cost of post-silicon calibration (i.e. off-chip measurements and testing), the HR-TDC allows automatic self-calibration under the control of a firmware and using only an off-chip crystal oscillator for clock generation. Fick et al. has demonstrated that a complete full self-calibration of an entire TDC implemented on a 64-bit Alpha processor can take only five minutes [75].

Using HR-TDC in NoCs

This section discusses in more details the use of High Resolution Time-to-Digital Converters in NoCs.

Using HR-TDC circuits to measure the slack or propagation delay of each pipeline stage in an NoC is important because exascale chips with thousands of nodes can experience both global and local Process-Voltage-Temperature (PVT) variability. Shown in Figure 5.5 is an in situ delay-slack self-calibrating Time-to-Digital converter from the work of Fick et al. [75]. The HR-TDC operates in three modes:

1. **Normal operation** - The HR-TDC is measuring the delay fed from the NoC Datapath. Not all paths are measured as it is expensive, instead this work only measures 30% of the top most critical paths as done by the work of Das et al. [76]. Subsequently, the data from the Time-to-Digital converter is sent to the NHM and aggregated to decide the minimum slack (max delay) amongst all pipeline stages.

2. **Reference Delay Chain (RDC) Calibration** - The HR-TDC is measuring the delay of the "Reference Delay Chain" using statistical sampling. Calibration of the RDC is needed before VChain calibration is started.

3. **Vernier Chain Calibration** - The HR-TDC is calibrating the Vernier Chain so that each stage in the chain has a delay of 5ps. A stage in the VChain is made tunable by

using eight firmware-controlled capacitor loads, with each load designed to introduce 1ps shifts in the delay.

The Vernier Chain (i.e. red portion of Figure 5.5 [75]) is responsible for measuring the slacks from the NoC data paths in each pipeline stage and converting it to a digital code. The rest of the modules in the figure (i.e. green modules) are used as a support or for self-calibrating the delays in the circuit. Immediately after fabrication, the HR-TDC is run in modes 1 and 2. At runtime, the HR-TDC is turned off. It is only turned on during boot-time sequence to measure the slacks of the pipeline stages. If it is suspected that the delay chains in the HR-TDC have changed, it can also be re-calibrated (run in modes 1 and 2) in order to maintain high accuracy in the slack measurements.

Hardware Overhead

The NHM is a low overhead circuit for measuring router delays. This work assumes that a router is faulty when its delay exceeds 20% more than the manufactured clock frequency. Overall, the implementation of the NHM on top of the open-source NoC router [49] at the 16nm technology node yields 3.2% and 1.2% overheads in area and power, respectively.

5.2.2 Propagating Delay Information and Routing Table Update

The encoded delay information is estimated and propagated through the firmware during the system boot-up, once a month. Three steps are needed to perform this function. First, all nodes estimate their own D_{max} in parallel throughout the system (Section 5.2.1). Second, the D_{max} is broadcasted through the flit link network. However, to avoid extreme flooding, the network is divided into small equally sized regions. Then, one node from each region broadcasts its D_{max} throughout the system. Third, the routing tables in each node are updated using this D_{max} information. For a 1024-node system, it is found that the D_{max} propagation of all nodes can be performed under $0.5ms$, assuming a 2GHz clock. *Doing this process at boot time avoids any runtime overhead, while negligibly affecting system boot time.*

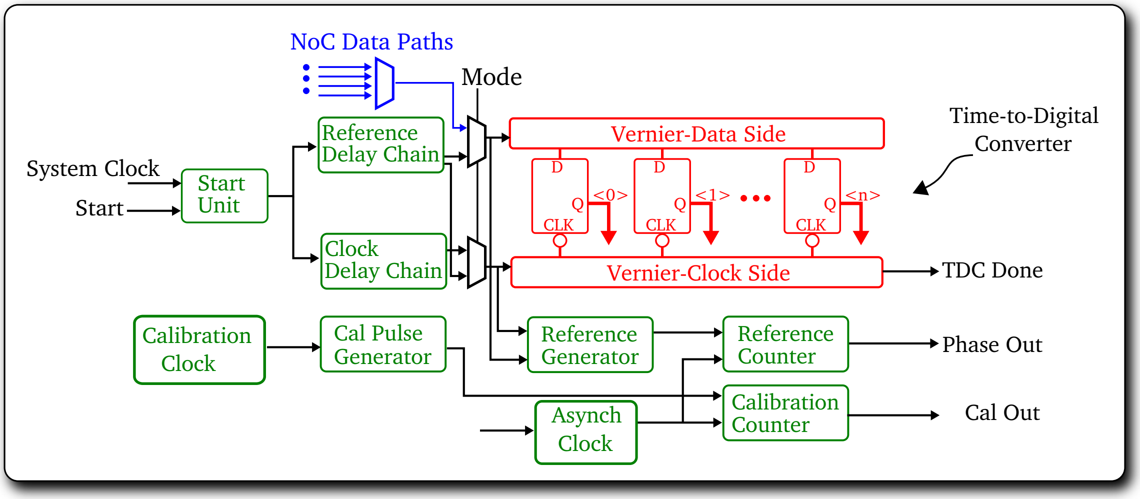


Fig. 5.5: High resolution in situ delay slack measurement from Fick et al. [75].

5.2.3 Routing Algorithm

This section discusses the light-weight and scalable routing algorithm used to route packets in order to balance the path degradation across the NoCs.

Scalable Routing

The proposed routing algorithm profiles all two-turn minimal paths of all source-destination pairs. The paths are chosen based on a particular metric such as average router degradation or maximum router degradation. Note that the path for a particular source-destination pair is only updated once per month in the boot-up time after all wearout information are propagated throughout the system. The firmware selects which path to take by writing the node address of the turning points in a routing register. Subsequently, the NoC will read this register at runtime and encode routing information in the head flit. For scalability purposes, the proposed approach uses algorithmic routing to decide which port the flit should be sent to.

Figure 5.7 shows an example of the routing algorithm in action. In this example, it is assumed that the firmware has already decided which turns to make for a flit with a source-destination of 0 and 11, respectively. The turns are made on nodes 2 and 10. Additionally,

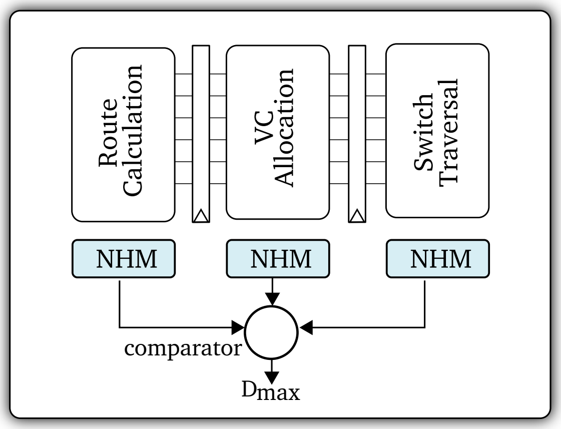


Fig. 5.6: NoC router augmented with NHM.

a single bit in the head flit is used to indicate which direction the flit should first go, X or Y direction. Whether it is up/down or left/right will be decided by the algorithmic routing based on the relative address of the source and the turning points. Once the flit hits one of the turning nodes, it is going to turn towards the direction of the destination. As such, the proposed algorithm is very scalable because no matter what the size of the exascale NoC is, the routing information stored in a flit (i.e. address of turning points) to be sent from a node to another will only grow by $log(n)$ with n being the # of nodes. All these information are shown in Figure 5.8. The total overhead is minimal at $2 \times log(n) + 1$ bits.

As discussed above, the algorithm only chooses between two-turn paths when deciding which routes to use. The reason for using this radically smaller solution space is that using three or more turns does not provide any significant benefits but will cost linear overhead on flit space requirements (i.e. m-turn path will need $m \times log(n)$ bits in the flit to route). In fact, the simulations in this work indicate that using an unlimited number of turns yields less than 4% degradation at the most, compared to two-turn paths.

Deadlock Avoidance

Routing packets using various two-turn path configurations can lead to protocol deadlock when cyclic resource dependencies exist. In the DWRR implementation, 1 Virtual

Channel (VC) is allocated in each port as an *escape channel* only to be used when avoiding a deadlock. Normally, when there is no contention, the flits will be routed on the non-escape channels. However, when all non-escape VCs from all routers are occupied for a certain period of time (100 cycles in simulations used in this work), a cyclic dependency could exist. This is possible because flits are not restricted to use the same VC ID in each hop in order to maximize the bandwidth of the network. The cyclic dependency is broken by halting further injection in the NoC and allowing in-flight flits to arrive at their destination using deterministic routing via the escape channels.

5.2.4 Applying NoC Health Meter in Dynamic Wearout Resilient Routing

Two DWRR algorithms are proposed that harness the platform provided by the NoC health meter to dampen the additional QoS-induced traffic stress in NoC routers. This work uses Duato's theory to restrict virtual channels to specific packet classes to avoid deadlocks [56]. The first algorithm is Fresh Routing (FR), which always routes the flits using the least-degraded path. This path is constructed by considering several minimal paths and comparing the average wearout information in each path. The second algorithm, Latency Reclamation routing (LR), seeks to balance congestion and reliability objectives by using dynamic runtime information when deciding a path. LR first compares the number of available credits–a metric quantifying the level of congestion in a node–of neighboring routers. If the least degraded path is congested, LR will choose the non-congested path. Let us consider a routing path with p routers, having maximum delays $D_1, D_2, ..., D_p$, respectively. Two variants of both FR and LR are studied:

- FR_{Avg} - This scheme uses the average wearout of all routers in a path to select the *least-aged path.* ($D_{path} = avg(D_1, D_2, ..., D_p)$).

- FR_{Max} - This variant of the FR algorithm selects the least-aged path using the maximum router wearout of each path. ($D_{path} = max(D_1, D_2, ..., D_p)$). This scheme seeks to limit the wearout of the most degraded router at any time interval.

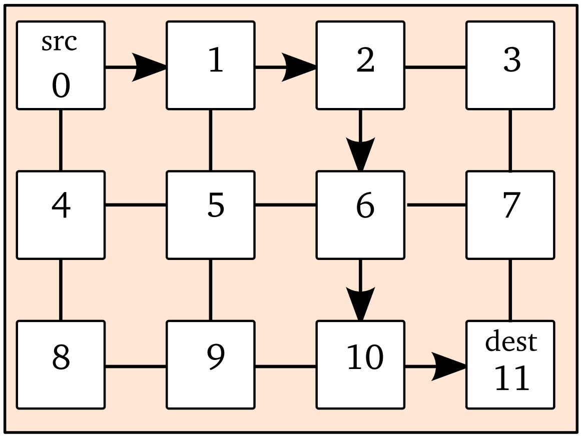


Fig. 5.7: Two-turn path routing.

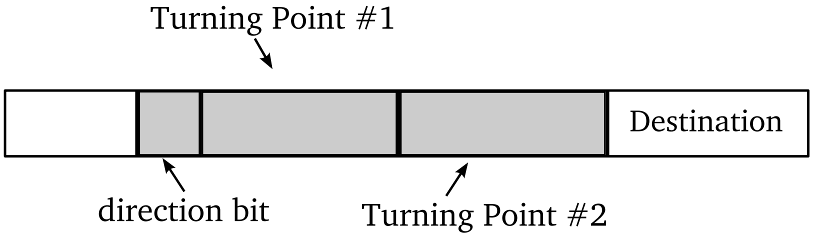


Fig. 5.8: Head flit: unused fields can be used to store additional algorithmic routing information.

- LR_{Avg} - This scheme is similar to FR_{Avg}, selecting the least-aged path based on average. However, during congestion, it avoids queuing delay by sending flits in the direction with more credits at times, when the least-aged path is overly congested. More details about congestion-awareness are discussed in Section 5.2.4.

- LR_{Max} - This variant of the LR algorithm also allows credit-based exceptions to the least-aged path. However, like the FR_{Max} scheme, it determines the least-aged path using the maximum router delay in each path.

Congestion Awareness

The DWRR algorithms have a congestion-aware variant, LR_{ave} and LR_{max}. This work did not use a complicated congestion avoidance scheme because the overhead of providing network-wide congestion data to each node in an exascale system is very prohibitive. For instance, in an $n \times n$ mesh network, adding a point-to-point congestion network would need $2 \times n \times (n-1) \times m$ wires for an m-bit congestion resolution. Instead, this work uses the RC-1D [77] congestion aware algorithm to supplement LR_{ave} and LR_{max}. In this scheme, the available bandwidth credit in each router is propagated in a single dimension in the mesh. So DWRR dynamically estimates the level of congestion in various paths by comparing their respective bandwidth credits. When the reliability-aware path is congested (i.e., low credit), the algorithm can choose a less congested path, temporarily overriding the reliability awareness.

This dissertation acknowledges that there are other possible congestion aware algorithms, such as piggybacking schemes (GCA [78]) that do not add significant area overhead but are prone to estimation errors due to fast changing congestion information in the network. These estimation errors would only be exacerbated in an exascale setting because congestion profiles will take longer to arrive.

5.3 Methodology

This section discusses the experimental methodology used to evaluate aging of an exascale NoC. First, an explanation of the hierarchical analysis capturing different levels of modeling (Section 5.3.1) is carried out. Then, the cross-layer simulation framework is discussed, which integrates multiple tools and models to enable wearout evaluation of exascale NoCs (Section 5.3.2).

5.3.1 Hierarchical Reliability Evaluation

The evaluation of network-wide reliability encompasses several layers in the hierarchy (Figure 5.9a). The analysis starts at the level of an exascale NoC and traverse different design levels (modules, gates, etc.) down to the transistor level. Figure 5.9a shows this hierarchy. From top to bottom, the platform is composed of the following hierarchy:

- **NoC:** At the top of the hierarchy is an architectural model of an exascale NoC with 1024 nodes implementing the GSF QoS policy [8]. The analysis starts by running traffic patterns captured from an architectural simulation of real-world PARSEC [70] benchmarks applications that are widely used in CMPs.

- **Router:** The router has two equivalent models, one for the architectural simulation and a circuit-level model for per-module power evaluation. The architectural model is embedded in Booksim [56] and emulates a QoS-aware router with 8 input and 8 output ports each router, 6 virtual channels (VC) per network port, 5 flits per VC, and a 3-stage speculative pipeline (Route Calculation, Virtual Channel Allocation, Switch Traversal).

- **Module:** The next level down the hierarchy is the module level. Each router in the NoC is modeled as a collection of modules that is the finest granularity for power/thermal modeling. The power/thermal properties of a module are used to evaluate several major wearout mechanisms such as NBTI and TDDB. The analysis done also evaluates HCI aging in the crossbar, the module that dictates cycle time [13], by capturing the switching activity in different modules [58].

- **Gates:** The modules are composed of several gates connected together, while gates are composed of several transistors connected together. At this level, data traffic are collected to obtain pertinent transistor switching information for use in HCI aging analysis.

- **Transistor:** At the transistor level, aging degradation is modeled as a combination of BTI, TDDB and HCI effects. BTI and TDDB are accelerated by large temperatures [59], while HCI is manifested by the switching transitions in the transistors [58]. For HCI and TDDB, the compact model by Wang et al. [61] is used, while for BTI aging the recent model employing the trapping-detrapping of ions [79] is utilized. All these mechanisms cause accelerated V_{th} degradation, decreasing timing slack, and eventually causing failure.

5.3.2 Lifetime Simulation

The simulation framework used in this study is shown in Figure 5.9(b) and is composed of several tools. Booksim 2.0 [56] and Netrace 0.9 [62] are used for accurate traffic modeling without incurring the overhead of a full-system architectural simulation. Power and Temperature modeling of different NoC modules are done using DSENT 0.91 [57] and HotSpot 5.02 [63]. An in-house logic analysis tool is implemented to observe HCI stress of transistors. Lastly, Synopsys HSpice is used to evaluate timing degradation from the device-level wearout.

The reliability framework is built as a closed-loop system with each iteration equivalent to one (1) month of wall clock time. As such, a simulation of N years would take N×12 iterations. The evaluation starts by running application benchmarks (PARSEC) using the Booksim/Netrace setup to simulate accurate traffic patterns and NoC router utilization profiles. The data dump and per-router utilization are then used to evaluate BTI, TDDB and HCI degradation. This is achieved by feeding the utilization profiles into DSENT to obtain per-module power statistics, and subsequently use HotSpot to calculate steady state router temperature. HCI aging is evaluated using the switching activity caused by the data embedded in the traffic.

5.4 Experimental Results

This section evaluates the effectiveness of the proposed DWRR algorithms. The baseline scheme, labeled Base is an exascale system with a QoS policy similar to GSF [8]. Four schemes are evaulated: FR_{Avg}, FR_{Min}, LR_{Avg}, LR_{Min}. The details of all these schemes are discussed in Section 5.2.4. This work compares the effectiveness of these schemes with the following metrics: the *overall system MTTF*, the quality-of-service through *mean-absolute difference (MAD) of packet latency* and *coefficient of variance* and lastly, *performance.*

The proposed schemes are compared against a state-of-the-art related work in fault tolerance called VICIS. VICIS is an NoC recovery mechanism proposed by Fick et al. [9]. VICIS provides many enhancements in an NoC router. The most notable is a crossbar bypass bus that is used to transfer flits from the input port to the output port, once a

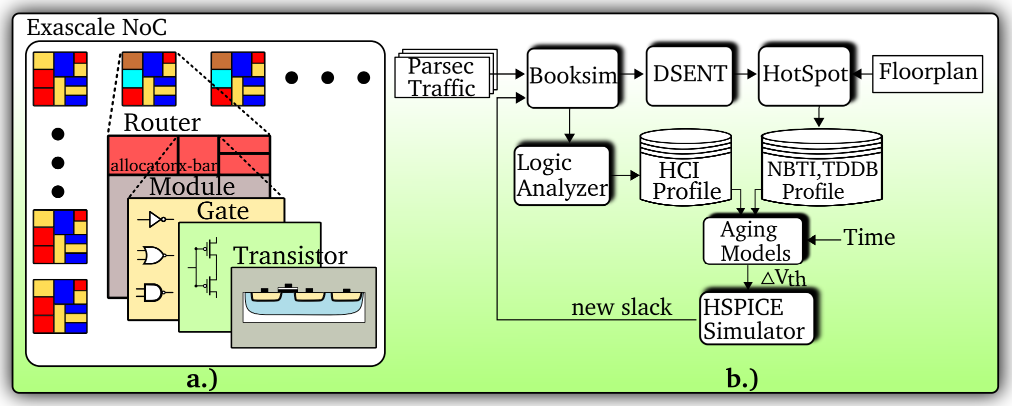


Fig. 5.9: Reliability-aware framework.

wearout-induced hard fault is manifested in the crossbar switch. Essentially, the router is reconfigured to a *bare bone* version trading off performance for correct functionality.

5.4.1 MTTF: Fault Model and Comparison

MTTF is traditionally defined as the time it takes from the start to the point of the first component failure. In NoCs, this is usually the point where a router in the network fails. However, proposals such as VICIS tolerate failures by providing hot-swappable bare bone components, thus retaining functional correctness at the cost of performance. As more components fail, the performance penalty rises till the system loses its ability to offer favorable cost-performance benefit over replacing the system with a new one. To capture this interplay while having a fair comparison of VICIS against the proposed schemes[2], in this work, a system is considered to be faulty only when 5% of the routers in the network have failed. Moreover, this work choses a 20% delay guardband for each component. Since VICIS offers fault tolerant routing, it may appear that this fault model is not fully justified. However, in reality, the experiments in this work suggest that a more relaxed component failure threshold or an aggressive guardband further degrades VICIS performance compared to DWRR. The reasons for this result are twofold:

[2] Recall that the proposed schemes have an orthogonal goal of extending fault-free and high-performance communication.

- If more component failures are allowed before considering a system to be faulty, the estimated MTTF improves both for VICIS and DWRR, but the relative improvement in DWRR is even more. On the other hand, the performance of the system suffers significantly beyond 5% failures. The simulations conducted can notice the substantial rise in latency soon after a component failure in VICIS. Moreover, as VICIS replaces faulty buffered routers with logically bufferless components, packets tend to incur more and more non-deterministic delays, further hurting the QoS support of the underlying network. Figure 5.10 shows this, where after the 3rd year, VICIS' latency has worsened compared to FR_{avg}.

- If an aggressive guardband is allowed in VICIS, it can marginally increase the operating frequency of the network. However, this slightly higher frequency does not provide any long-term performance boost. First, allowing a smaller guardband leads to faster component failures (e.g., much before the 3-year mark shown in Figure 5.10). Subsequently, VICIS starts to show poor performance compared to the relaxed guardband, due to the increased latency from the bufferless routers. Second, the MTTF is also reduced as the point of 5% failures is reached earlier. Overall, aggressive guardbanding neither helps performance or QoS, nor does it help the MTTF.

Figure 5.10 shows the network latency in *blackscholes* (one of the PARSEC benchmarks), as the system progressively degrades. The point A is the time of the first router failure, the Base scheme stops at year 3. VICIS extends the lifetime of the NoC by replacing broken routers with bare bone versions at the cost of lower performance. Meanwhile, the proposed schemes only use DWRR to extend the lifetime of the routers. Eventually, VICIS fails around 8 years (point B). FR_{avg} improves the performance by further pushing the point of the first router failure to C (from A), thereafter complemented with VICIS to promote both wearout resilience and fault-tolerance. Eventually, FR fails at 9.8 years (point D). The occasional improvement in latency is due to specific routing paths exercised in that time frame (e.g., paths with fewer degraded routers), although device-level wearout monotonically degrades the router delay characteristics.

Figure 5.11 shows the overall MTTF of the different schemes. Base has the least MTTF with an average of 5.6 years. VICIS extends the MTTF by 45% to 8.1 years. The proposed routing schemes provide an additional 25% MTTF improvement on top of VICIS, on an average. Among the proposed schemes, the minimum improvement over VICIS is at 15%, while the maximum is at 26%. FR does not consistently outperform LR in terms of MTTF because the former only cares about routing all packets through the least degraded path. This approach can be occasionally detrimental as a particular path may be overly used by all flits once it is determined to be optimal for a certain time.

5.4.2 Quality-of-Service

Quality-of-Service (QoS) is evaluated using two widely used metrics in Figures 5.12 and 5.13. Figure 5.12 shows the jitter of the programs under different comparative schemes, where jitter is measured using the Mean-Absolute-Difference (MAD) [80]. For a majority of the programs, the best schemes provide better resilience to jitter, compared to VICIS. The proposed schemes achieve this by dampening the traffic-induced stress incurred by adding QoS. The DWRR algorithms redistribute the traffic around degraded routers, hence, extending the lifetime of high performance routers. Consequently, the proposed schemes maintain the QoS in the system, while VICIS's QoS degrades due to many low-performance bare bone routers replacing the original routers. LR_{max} provides the best improvement for *swaptions* that see 30% more resilience to jitter. On an average, the improvement is about 9.3% for LR_{max}. Some benchmarks provide little or no benefit in terms of jitter resilience,

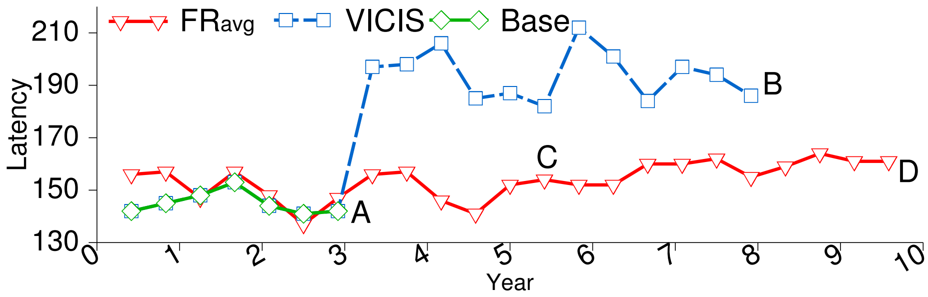


Fig. 5.10: NoC performance of blackscholes under progressive aging in various schemes.

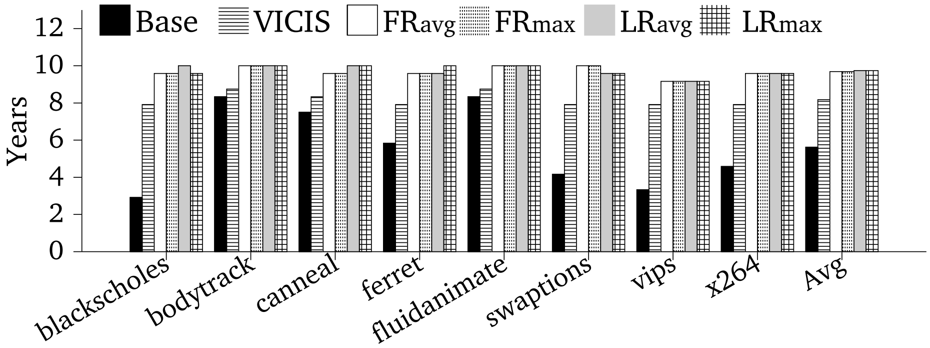


Fig. 5.11: MTTF (higher is better).

primarily due to low-levels of congestion in them, offering little opportunity to improve jitter.

Figures 5.13 and 5.14 show another QoS metric which measures the dispersion of packet latencies using Coefficient of Variation (CoV $= \frac{\sigma}{\mu}$). Figure 5.13 shows the lifetime CoV of all schemes. On an average, the FR_{max} and LR_{max} schemes improve the CoV by 7% and 16%, respectively. The biggest improvement is from *ferret* which is reduced by 70-75% when using the DWRR algorithms in this work. Three programs (canneal,swap,vips) show degradation of QoS for FR schemes due to congestion, however, the LR schemes are able to dynamically reroute traffic and improve the QoS.

Figure 5.14 is a time-lapse graph of *blackscholes* showing how VICIS provides a functional platform with less QoS when routers start to fail. From year 3 to 8, VICIS has higher CoV compared to the DWRR schemes because it uses bare bone routers and takes a longer time to route flits in the network. In contrast, the proposed DWRR schemes degrade gradually and only start to worsen their QoS towards the end of their lifetime (years 8.4-10).

5.4.3 Performance

Figure 5.15 shows the performance measured as the average flit latency. The proposed schemes perform better than VICIS by extending the fault free execution of NoC routers. In contrast, VICIS utilizes bare bone routers with lower performance as soon as it encounters a defect in the crossbar switch. The best improvement is on benchmark *vips* which is 71% using the LR_{avg} scheme. VICIS's performance on *swaptions* and *vips* is more than 2×

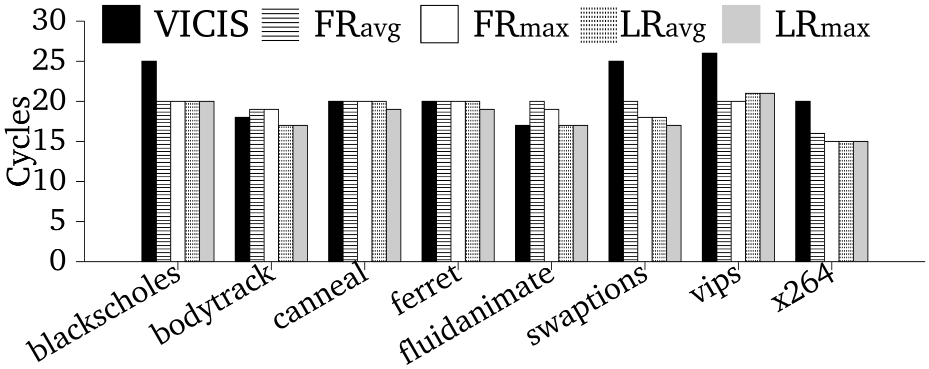


Fig. 5.12: QoS:jitter (lower is better).

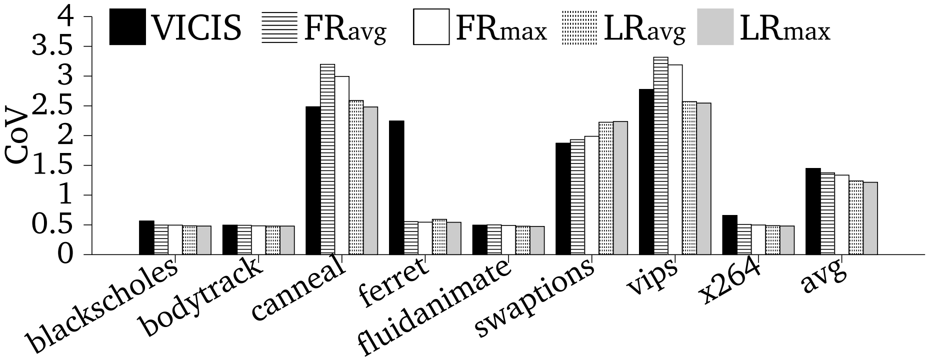


Fig. 5.13: QoS:CoV (lower is better).

slower. Upon a closer look, it is found that these performance penalties stem from particular traffic patterns in these benchmarks, which overly exercise routing paths with routers operating at a reduced functionality (bare bone version) and lacking the ability to route flits in an efficient manner. Other benchmarks show a fairly constant traffic throughout the simulation. In all programs, LR performs better compared to FR by dynamically avoiding congested paths during high traffic. On an average, FR and LR outperform VICIS by 39% and 43%, respectively.

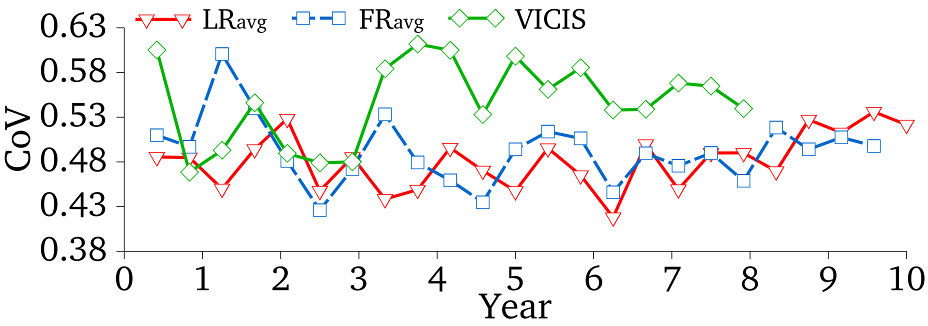


Fig. 5.14: QoS:Blackscholes CoV (lower is better).

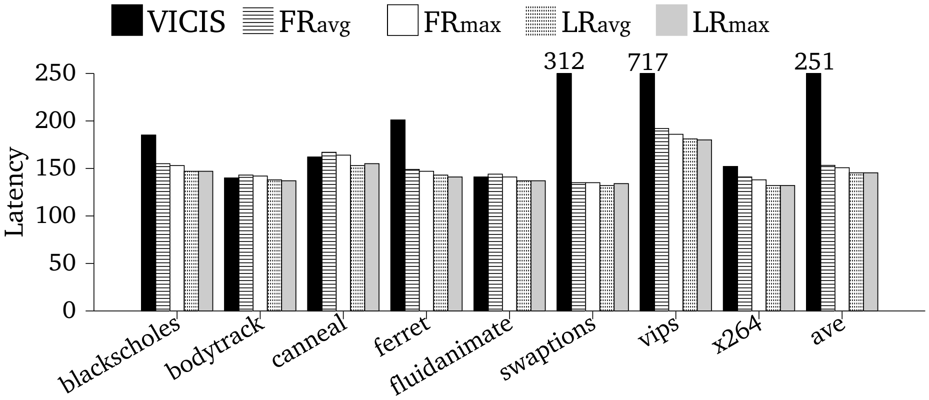


Fig. 5.15: Latency (lower is better).

Chapter 6

Fort-NoCs: Mitigating the Threat of a Compromised-NoC (C-NoC)

This chapter presents, Fort-NoCs, a series of techniques to address the threat of a C-NoC. First, this chapter starts with a detailed discussion of why a C-NoC poses a great threat in emerging architectures. Then, three proposed techniques that work in different abstraction levels are explained. Lastly, the overheads of the said techniques in terms of performance, power and area are evaluated.

6.1 Threat of a C-NoC

This section outlines the security threat posed by a compromised NoC. The discussion starts by outlining the threat model and briefly discussing the significance of this threat in modern software and hardware practices. Subsequently, the next sections explains how an attack can be implemented with a low overhead, demonstrating the high potency of this threat.

6.1.1 Threat Overview

Figure 6.1 shows a conceptual overview of one specific threat posed by a compromised NoC: information leak. The security attack must have two complementary hardware and software components. First, a hardware trojan implanted inside the NoC is responsible for facilitating an attack on the on-chip communication network. Second, an accomplice application must secretly communicate with the hardware trojan to activate it and send commands. Based on these commands, the trojan can carry out a plethora of potent attacks, such as information leak and denial of service.

Without loss of generality, the sequence of phases to realize information leaking on a compromised NoC are outlined next.

1. *Design Phase:* In the design phase, a third party NoC IP provider inserts a hardware trojan in the NoC. The trojan is able to detect and act upon commands, which are inserted in the flits communicated through the NoC. Section 6.2 shows how to insert this trojan in a standard NoC router with a minimal footprint.

2. *Activation Phase:* There are two specific steps in the activation phase. First, an accomplice thread (AcTh) is scheduled on one of the on-chip processing elements (PE). This step can be realized in a cloud computing setup where a client thread is co-scheduled with other client threads of the service provider. Second, the AcTh establishes a covert communication channel with the embedded hardware trojan in the NoC. For example, the AcTh can write a covert message on its own address space, and then cause eviction of the cache block containing that message. When a close-by NoC node receives data flits containing that covert message, it sends an acknowledgment message back to the AcTh. That NoC node subsequently sends covert messages to other NoC nodes in the network to activate them, while reporting the location of the AcTh. Once all activation sequences are received, the trojan in each NoC node will be waiting for further commands from the AcTh.

3. *Operational Phase:* During the operational phase, the AcTh sends commands to initiate a malicious activity. In the case of information steal, the AcTh may request duplication of specific on-going data communication. For example in Figure 6.1, an AcTh residing in the local PE of NoC node X could send a command to the NoC node W to leak the communication between nodes Y and Z. The hardware trojan at NoC node W will then duplicate all packets going in/out of its local PE and send it to the AcTh at the PE of NoC node X.

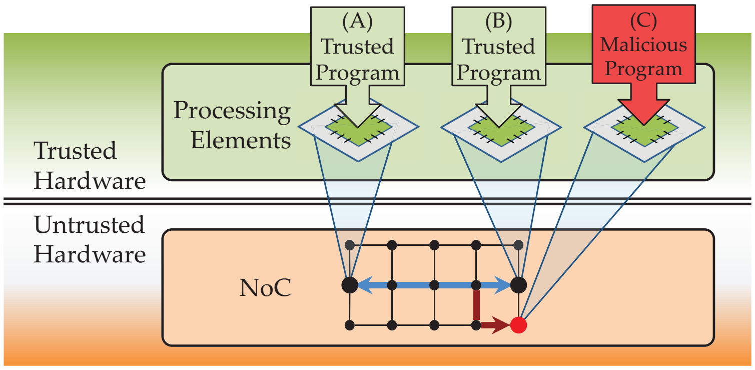


Fig. 6.1: Compromised NoC snooping data messages between programs A & B, and leaking to the accomplice program C.

4. *Tear down Phase:* In the tear down phase, the AcTh informs the compromised NoC to suspend its malicious activity. Similar to activation, this operation can be realized by sending a previously agreed upon message through a dirty cache block.

6.1.2 Threat Relevance

The setting of the security attack in this work is a cloud computing system with many users running different programs on an MPSoC. This threat will be of growing importance as MPSoCs are poised to take over general purpose processors in cloud computing hosting [81]. This work also assumes that for promoting a cost-efficient MPSoC design, the on-chip interconnect is designed with a 3rd party NoC IP. This is a very likely scenario as NoC IP blocks continue to find their way in many SoCs. In fact, iSuppli, an independent market research firm, has determined that four out of the top five Chinese fabless semiconductor OEM companies use the FlexNoC interconnect from Arteris [82]. Consequently, Arteris has achieved a three-year 1002% sales growth through IP licensing [83]. Given this trend, this dissertation has shown a unique threat model where the NoC design is compromised. With an accomplice software, such a C-NoC can engage in a wide range of malicious activities such as information extraction, denial of service or voluntary data corruption.

A key question now is can a third-party design a seemingly innocuous but malicious NoC? To answer this question, one must understand two central aspects of a C-NoC design.

First, the design footprint of such a design must be evaluated, in terms of its area and power overhead. Second, an analysis must be done of the runtime performance overhead on other applications, while the NoC is engaged in malicious activities. Ideally, a designer would like to keep these overheads low to increase the threat potency. These key issues are discussed next.

6.2 Evaluating a C-NoC

This section goes into more detail about the design of a C-NoC (Section 6.2.1), and evaluates its design footprint (Section 6.2.2) and the runtime overhead (Section 6.2.3).

6.2.1 Design Overview

Figure 6.2 shows a classic NoC 4-stage virtual-channel (VC) router pipeline. The four stages are the input buffers/route calculation, VC allocation, switch allocation and switch traversal. There are p ports with v virtual channels in each port. A trojan hardware (HW) that duplicates incoming packets from the local processing element can be inserted in each or one of the ports as shown in the figure. The trojan taps the incoming links from the network interface (NI) and watches out for covert signals from a possible accomplice thread.

The brain of a trojan is a state machine that has three major states: *inactive*, *waiting* and *leaking*, described in details in Table 6.2. There are minor states in between the major states to ensure that the major state changes are sanctioned by the AcTh. To change the state of the trojan, a coded sequence of flits are sent by the AcTh. Although, accidental major state changes are possible due to random traffic, such an occurrence has an extremely low probability. The probability that m consecutive n-bit sequences can occur when having a random traffic is described as

$$P_{change} = \left[\frac{1}{2^n}\right]^m. \tag{6.1}$$

For $m = 5$ and $n = 32$, $P_{change} \approx 10^{-50}$. Both variables can be customized by the trojan creator according to their needs.

6.2.2 Design Footprint

To evaluate the power and area overhead of adding a trojan in an NoC, the RTL of an open-source NoC Router [49] is modified. The logic for the trojan described in this section is added. It is also assumed that the router is used as part of a mesh topology with 5-input/output ports (4 cardinal directions + 1 local) and 5 virtual channels in each port. The design is then synthesized with the TSMC 45nm library using the Synopsys Design Compiler. The results for the power and area overhead are shown in Table 6.1. Adding the trojan HW yields only 4.62% and 0.28% overheads in area and power, respectively. This result demonstrates that the footprint added by the trojan HW is really low and hard to detect.

6.2.3 Runtime Overhead

Evaluating the runtime overhead of the proposed attack model is critical, as a designer would like to engage in covert activity with low overhead so that its activity remains undetected. Therefore, a rigorous analysis is performed to simulate the effect of a covert theft on the network traffic.

The experiments conducted used the gem5+garnet simulator [71] with PARSEC [70] benchmarks to simulate execution on an 8x8 mesh topology. The trojan model is then added that duplicates every packet inserted at the target node and redirects it to the AcTh. Finally, the overhead is determined using the average packet latency increase due to the additional traffic caused by duplicating packets. In a large NoC, the relative position of the target node and the malicious node, where the AcTh is scheduled, can vary substantially. Figure 6.3 shows two of the many possible theft paths. Depending on this relative position,

Table 6.1: Design footprint of the NoC trojan hardware.

Metric	**Baseline**	**with Trojan**	**Overhead**
Area (um^2)	66400	69474	4.62%
Power (mw)	705.56	707.71	0.28%

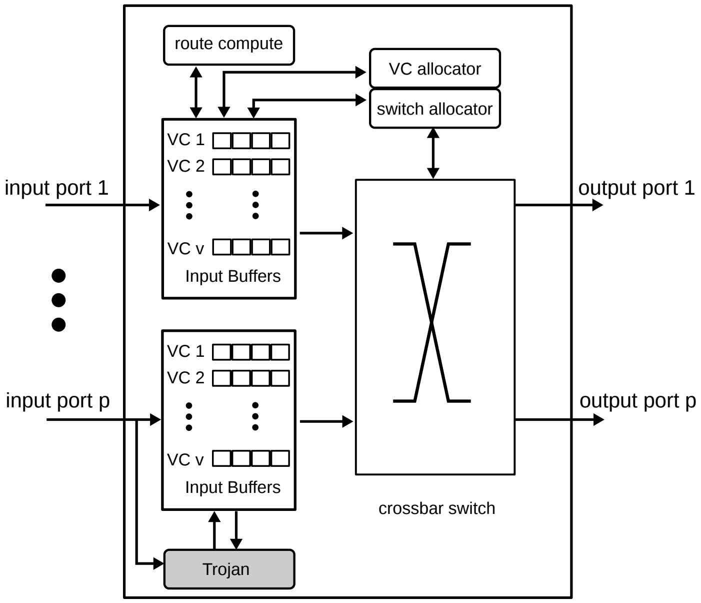


Fig. 6.2: Compromised NoC with hardware trojan.

the overall impact on network latency can change substantially. Thus, the experiments performed analyze all possible theft paths across all benchmarks to evaluate the runtime overhead.

Figure 6.4 shows the probability distribution function (PDF) of the average latency increase versus the number of possible theft paths. On an average, 83% of the theft paths have a 0% overhead, 14% have between 0-5% performance overhead and *only* 0.25% of the theft paths have an overhead of more than 10%. These results show that this type of a covert theft can occur with a very small impact on the network latency when the longer theft paths are avoided. Given this result, it is clear that a compromised NoC poses a potent threat to MPSoCs in the near future. The proposed approach in protecting against this threat model is discussed next.

6.3 Mitigating the Threat of a C-NoC

This section discusses the proposed design solutions to prevent covert data theft by a compromised NoC. The proposed solutions introduce security measures in the SoC firmware that interfaces the processing element with the network interface (NI) of the NoC. Shown

Table 6.2: States of the trojan.

State #	Description
Inactive	The trojan is inactive and is waiting for an accomplice thread to communicate.
Waiting	An accomplice thread has established communication. The trojan is waiting for further commands.
Leaking	The trojan is currently sending duplicate data to an accomplice thread.

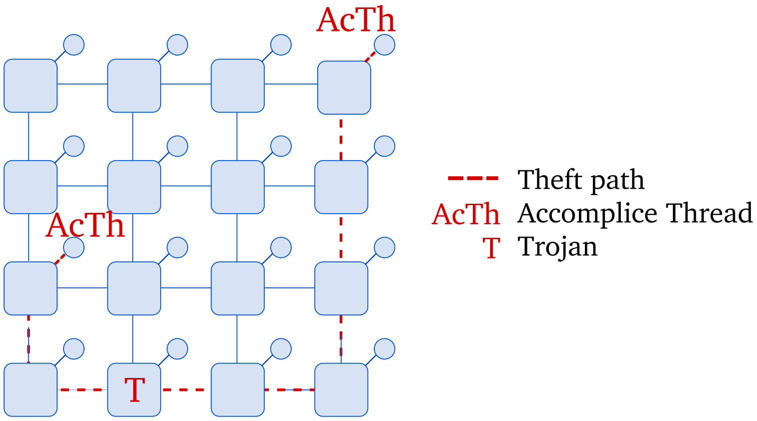


Fig. 6.3: Trojan with two possible locations of AcTh. The length of the theft path depends on where the AcTh is.

in Figure 6.5 is the design methodology used by SoC integrators. This work assumes two security levels: a *trusted* in-house design team hired by the SoC integrator and one or more *untrusted* 3rd IP vendor(s). Once the in-house design team and the IP vendors agree on what protocol to be used, the design team can then independently develop its own trusted interface to the IP NoC chip. Further, it is assumed that the NIs use the Open-Core Protocol (OCP) [84] for interfacing on-chip components. Hence, *Fort-NoCs's techniques do not depend on 3rd party vendors, including the NoC IP provider, and are protected from malicious alterations by them.*

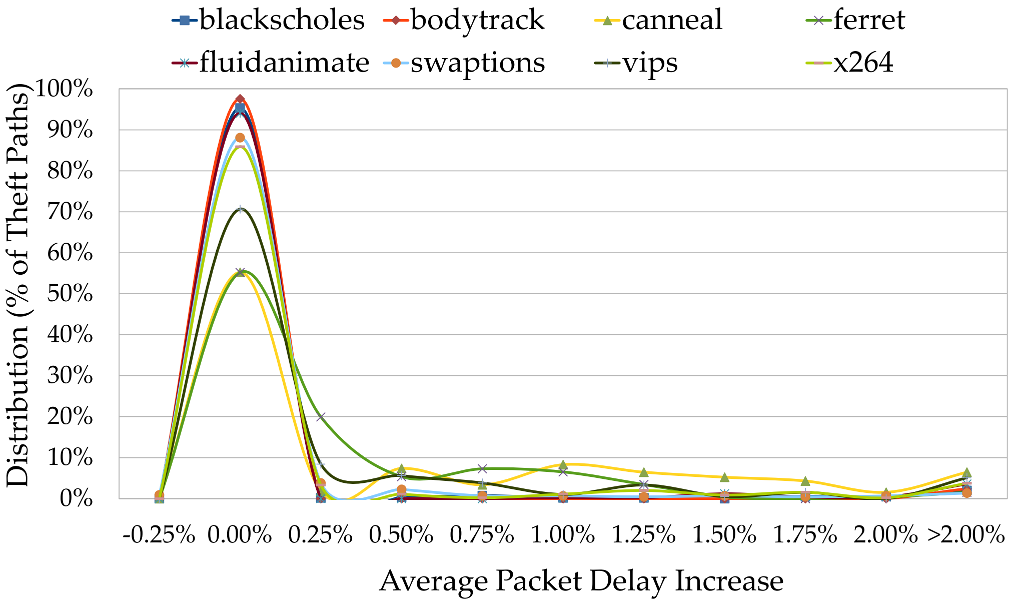


Fig. 6.4: Runtime performance overhead. The figure shows overhead distribution across all possible theft paths.

Fort-NoCs is a three-layer security system approach that provides both proactive and reactive protection against information leaking attacks. The three layers of security are outlined below:

- *Layer* 1*–Data Scrambling:* At the lowermost layer, Fort-NoCs proposes a low-overhead data scrambling technique to avoid clean data transmission over the NoC at all times (Section 6.3.1). This technique creates a stiff barrier for the backdoor activation of the trojan, as well as, dramatically reducing the efficacy of NoC data snooping.

- *Layer* 2*–Packet Certification:* The second layer creates a barrier between the NoC and the processing element through a dynamic packet certification mechanism (Section 6.3.2). Flits with invalid certificates are never forwarded to the processing element, thereby breaking the link between the software running on the processing element and the NoC hardware.

- *Layer* 3*–Node Obfuscation:* At the topmost layer, this work introduces an elegant technique to dynamically hide the communication nodes in an NoC, thereby introducing massive noise in the NoC data (Section 6.3.3). The proposed technique achieves remarkable side channel resilience with minimal performance overhead, and zero overhead on NoC bandwidth demands.

6.3.1 Layer 1: Data Scrambling (DS)

The first technique, Data Scrambling (DS), proactively prevents covert trojan backdoor activations by scrambling the data in the SoC firmware before injecting it in the NoC. Consequently, the activation key sent by the AcTh will be distorted, thereby blocking its intended communication with the trojan in the C-NoC. Apart from preventing activations, it also renders the leaked data incomprehensible to the attacker. The SoC firmware scrambles the data using a transformation technique that can only be reversed at the intended destination. There are many ways to provide low overhead hardware encryption. This work utilizes the well known XOR cipher encryption to encrypt the data before sending it in the compromised NoC. An example of how DS works and its implementation are discussed next.

Figure 6.6 shows a conceptual view of DS implementation with XOR cipher encryptors and decryptors. The sender end of the SoC firmware encrypts the data packet using a dynamic key, while the receiver end of the SoC firmware decrypts the packet using the same dynamic key. This work uses a low-overhead dynamic key for every destination node where the key is generated by the firmware at the system boot-time, and stored in a lookup

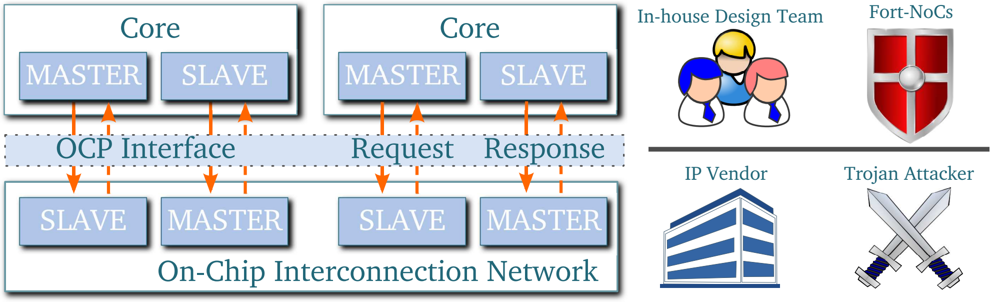


Fig. 6.5: Fort-NoCs design perspective.

table. For example, one can use the node ID itself as a key to encrypt the messages. Figure 6.7 shows an example of how the DS works with a dynamic key (destination node ID in this case). It also shows the possible decoding of the message as it is routed to different nodes, including the destination node 8 in Table 6.3. As such, the keys are never exposed to untrusted parties involved in the MPSoC development process. The configuration can be done through a firmware and can be changed anytime as desired.

6.3.2 Layer 2: Packet Certification (PC)

The goal of PC is to provide a second layer reinforcement of security, once the previous technique has been breached. Although the previous technique is effective in preventing intended activations, the trojan backdoor can still be activated accidentally, albeit at a very low probability (Section 6.2.1). Other types of triggers that cannot be prevented by DS can also be integrated in the circuit. For instance, analog thermal triggers (i.e. triggering after certain temperature is reached) and timer triggers can also be inserted by attackers [85]. This is the precise reason a multi-layer security approach is important to protect against C-NoCs.

PC provides data integrity protection by attaching an encrypted tag at the end of the packet before injecting it in the NoC. The SoC firmware creates a dynamically generated random lookup table at the boot-time that lists a 16-bit unique identifier for each node in the system. Based on the destination node of a packet, each data packet embeds a tag

Table 6.3: Possible decoding of 0xDABC at different nodes.

Dest Node #	Result	Dest Node #	Result
0	–	5	0x579B
1	0xB579	6	0xAF36
2	0x6AF3	7	0x5E6D
3	0xD5E6	8	0xBCDA
4	0xABCD		

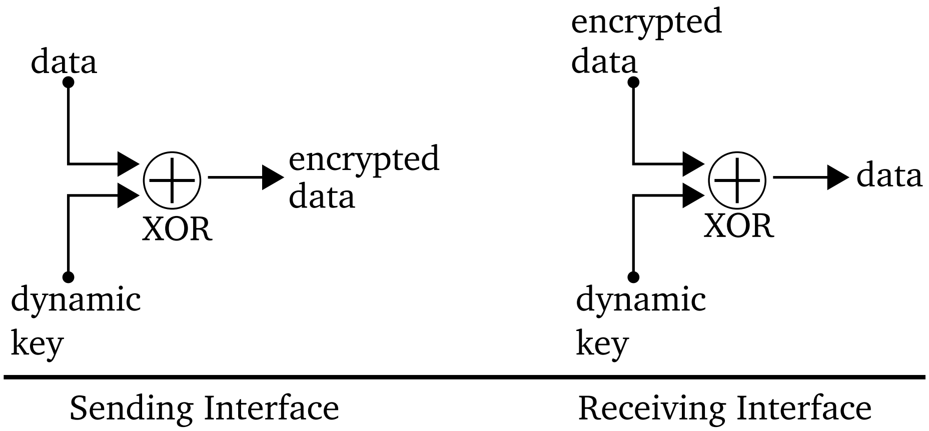


Fig. 6.6: Low-overhead XOR cipher allows encryption/decryption using dynamic keys from the firmware.

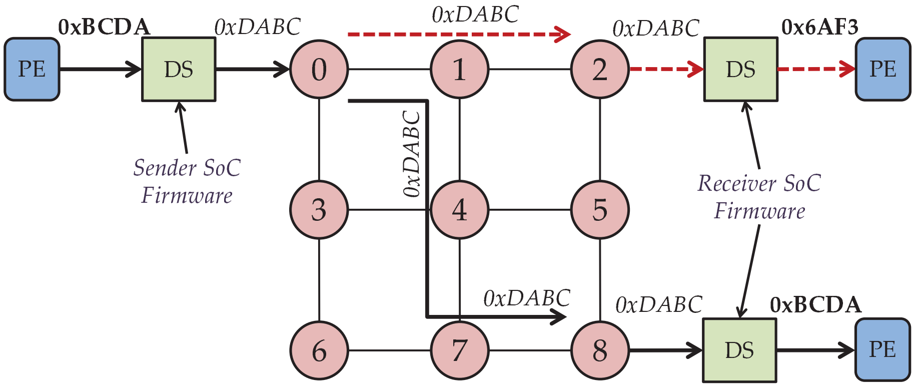


Fig. 6.7: Example for DS.

containing the *translated identifier of the destination node* from the lookup table. Before forwarding the data to the PE, the SoC firmware at the destination verifies this certificate by comparing its own local copy of the lookup table and the destination ID from the OCP header. PC relies on the fact that an on-going information leak attack will always involve sending a copy of a flit to an unintended location, where the malicious application resides. Once this is detected, the SoC firmware drops the packet, while also triggering an exception that will alert the hypervisor/OS of an ongoing security breach.

With sufficient insight and inside information, a creative attacker can outdo PC by thinking ahead and dropping the tail flits or by replacing it with a copy of the tail flits originally sent to the node of the malicious application. This can fool the verification test by the SoC firmware at the PE interface. This can be easily solved by using a post-silicon configuration to determine which part of the packet the tag should be inserted, and periodically altered at the boot-time. By configuring post-silicon, Fort-NoCs ensures that the designer is always ahead of the attacker in the "cat-and-mouse" security game.

6.3.3 Layer 3: Node Obfuscation

Node Obfuscation (NObf) is the final armament in Fort-NoCs's arsenal of security measures. One of the key aspects of security threats posed by a compromised NoC depends on identifying the target application running on a particular node on the NoC. NObf aims to obfuscate the NoC communication semantics by periodically decoupling the source-destination of a given communication. This section discusses the insight of the proposed mechanisms, implementation of the scheme, and outlines how to estimate the efficacy of this technique.

Design Insight

Wang and Suh have shown that a malicious thread running together with a hashing algorithm can analyze or narrow-down the possible values of a private key by observing its own traffic latency [38]. Pellegrini et al. also showed that private keys can be derived from observations of various faulty encrypted data [86]. These observations led us to explore a fundamentally different and complimentary approach of introducing massive noise in the NoC data communication by altering the source/destination. By mixing the data sent over any specific routing path, recovering meaningful sensitive information from an application becomes almost impossible.

Implementation

NObf is implemented in the SoC firmware by initiating a routine seamless migration of the running application to another node in the system. Of course, there are various PEs in the MPSoC, and care must be taken while performing these migrations so that matching

PEs are selected when a thread is migrated. To this end, SoC firmware maintains the type of PEs, and uses such information for performing NObf. The overhead of NObf comes from moving the architected state (processor registers) and cold cache effects. For practical intervals between two migrations, the architectural simulations show these effects to be low for r benchmarks.

Node Obfuscation is specifically implemented through the firmware micro-code utilizing the support for application migration provided by many modern ISAs [87, 88]. The micro-code uses special instructions to save the processor state (all registers) into a pre-designated memory location. Subsequently, the processor state is restored from the that memory location in the new processing element. The memory state is transferred through the cache coherence protocol of the system.

Two key problems with such application migration stem from portability: backward and forward. The backward compatibility problem arises when the application is migrated on a processing element that does not fully support all the functionality of the previous processing element. Similarly, forward compatibility can be a problem by altering error behavior sequence (e.g., previously invalid opcode correctly decoded on the new machine). To resolve these problems, the SoC firmware maintains a compatibility list for each on-chip processing element, and enables node obfuscation only between compatible nodes.

Efficacy Estimation

To estimate the efficacy of the proposed NObf, this dissertation utilizes the recent work by Kocher et al. to transform this problem into a signal-to-noise (SNR) estimation. Though noise injection to the signal does not provide theoretical security, it increases the attacker's effort of extracting secret keys [89]. Kocher et al. has shown that decreasing the SNR of the side-channel information can linearly or quadratically increase the number of inputs required for a successful side-channel analysis [90]. Hence, this work analyzes the disturbance introduced by NObf to the original data in the network. The experiments done involve computing the amount of original data (i.e. signal) communicated over a given routing path, compared to unrelated data communicated over the same path.

For better comprehension, this work analyzes the communication profile in a given scheduling quanta (Q), which can be a regular OS scheduling quanta (commonly 10ms, but can be up to 100ms for high priority tasks [91]). To estimate this SNR, Fort-NoCs must assess the maximum time a given communication, uniquely identified by the source-destination pair, is utilizing a given routing path. Without NObf, this time is the entire Q. However, NObf migrates a running application to a different node after every P time (referred as *Singular Vulnerability Period (SVP)*, thereby creating $\left\lceil \frac{Q}{P} \right\rceil$ intervals within a Q. Assuming, there are N threads in the system, a given application may occupy the same node for $\left\lceil \frac{Q}{PN} \right\rceil$ intervals, for a total time of $P \times \left\lceil \frac{Q}{PN} \right\rceil$. For all other times in the Q, the communication between the said source node to the destination node is essentially noise in the system. Thus, the SNR of the proposed NObf is derived as

$$SNR_{NObf} = \frac{P \times \left\lceil \frac{Q}{PN} \right\rceil}{Q - P \times \left\lceil \frac{Q}{PN} \right\rceil}. \tag{6.2}$$

Side-Channel Resiliency

Figure 6.8 shows the side channel resiliency, measured as SNR, of the proposed NObf as a function of the SVP (see Section 6.3.3). Fort-NoCs uses a round-robin scheduling of threads because it prolongs the time that the target thread is scheduled again on the target node compared to a random scheduler. In the said figure, a minimal SNR is desired in order to hide the actual information with other data. At the same time, this minimum SNR is ideal if it were to occur at the maximum period as much as possible so that NObf does not have to incur the overhead of rescheduling threads very often. However, designers do not have to use the minimum SNR and can set a *threshold* at a more practical period size.

A key feature of the proposed scheme is the scalability as seen from the figure: increasing thread count substantially improves SNR. For instance, in a 16-threaded application, there are several minimas. However, the optimum period is at point A because it incurs less overhead performance-wise. As the number of applications or threads on the system are increased, the optimum period gets pushed to the left (points B and C) as the system needs more and more applications to use any specific node in order to introduce more noise. For a 64-threaded application, the optimum period is 0.25ms for a 10ms quanta. However, for a 256-thread system, the optimum P is less than 0.1ms, but one may even chose 0.25ms and achieve comparable SNR to a 64-thread system.

6.4 Potential Spoofing Attacks

In the threat model proposed in this dissertation, the compromised NoC can spoof any node in order to create a dummy request of a privileged information. The proposed techniques successfully prevent these *spoofing* attacks. For PT, embedding identification information in the tail of the packet can distinguish a real request from a bogus one. Intra-node spoofing attacks also exist when running multiple threads in a PE, such as one where a PE is a Simultaneously Multithreaded Processor. When running multiple threads, an untrusted co-running program can initiate an attack and steal information from its co-scheduled thread successfully. To nullify this attack, Fort-NoCs copies the thread identification tag from the OCP header into the unused portion of the head flit. Subsequently, the NI matches these thread IDs before allowing the thread to access the NoC data.

Another successful spoofing attack can happen if the NoC trojan can appropriately decode the *tags* or the *scrambling* of the flits. However, even if an attacker is aware of the mechanisms used by Fort-NoCs, the added complexity needed to circumvent these security measures will greatly increase design overheads of the trojan, defeating its purpose due to easier detection.

6.5 Experimental Results

This section presents the area, power and performance overheads incurred by adding the proposed FortNoCs technique. This work observes a scalable side channel resilience of the proposed schemes (presented in detail in Section 6.3.3). The baseline used in this study is a traditional NoC without any security features. This work compares Fort-NoCs against state-of-the-art related works in preventing information leakage. The NoC-MPU scheme by Porquet et al. that prevents unauthorized read and memory accesses at each NI [16] is used

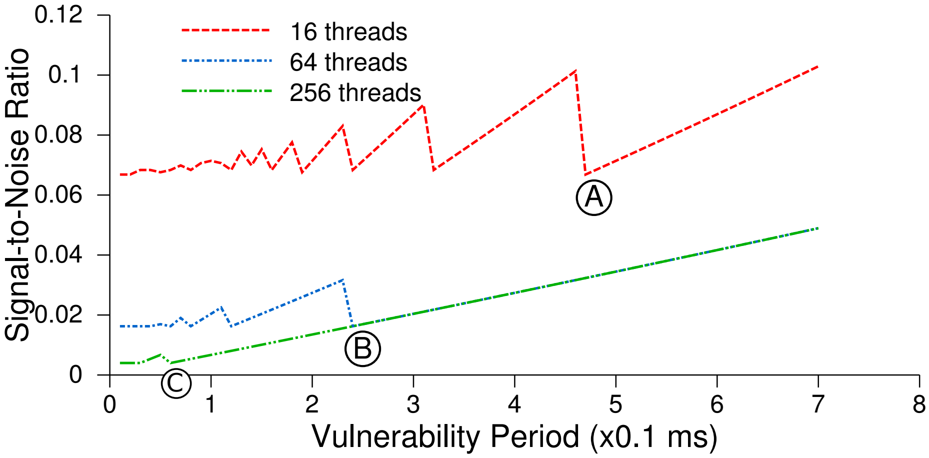


Fig. 6.8: Side-channel resilience comparison (lower is better).

as a comparative scheme. The implementation of the NoC-MPU used has a small 10-entry permission lookaside buffer (PLB) table in the NI. The PLB accesses a Permission Table (PT) in a separate on-chip memory when a miss is encountered in the PLB. The PT is not counted towards the area and power overhead. The MPU takes 1 cycle to authenticate an operation and 10-cycles to refill a PLB miss.

6.5.1 Implementation

The proposed designs are evaluated on several metrics such as power, area and performance. To obtain power and area overheads, this work uses the open-source Stanford Verilog model of a modern NoC router as the RTL baseline [49]. The trojan is inserted on top of a virtual channel router to implement the functionality described in Section 6.2. The NoC router model used in this work has a 3-stage speculative pipeline composed of RC/VC allocate, switch allocation and switch traversal. To evaluate the performance overhead, this work models DS, PC and NObf in the gem5+garnet setup [71]. These modules are interfaced during packet dispatch and reception. Real-world PARSEC benchmarks are then simulated using full-system simulation.

6.5.2 Area and Power

Table 6.4 shows the overhead of adding security features in a standard SoC OCP interface [92]. NObf is excluded from the table as it does not change the OCP interface. Fort-NoCs's schemes show lower overhead compared to the NoC-MPU. Between Fort-NoCs's two schemes, PC has about 15× lower area overhead compared to DS. The large area overhead of DS is due to the shifters needed to transform the data bits during the packetization and depacketization stages. PC has no power overhead as it needs minimal logic to attach a tag on the unused field of the head flit. NoC-MPU has a large overhead compared to Fort-NoCs because of the logic to check if each access is authorized. This involves checking each entry of a content-addressable-memory (CAM) like the PLB to see if a thread has read/write access to a memory location.

6.5.3 Performance

Fort-NoCs provides a three-layer security to a system with a compromised communication platform. As such, this work evaluates the incremental effect on the performance overhead of adding each layer of security. This trade-off analysis will allow designers to configure Fort-NoCs based on the type of security required by the application or design. For instance, high-security domains can use Fort-NoCs's three layers at the cost of higher performance penalty while low-security domains will probably need only 1 or 2 layers of security.

Figure 6.9 shows the performance overhead of Fort-NoCs. The figure shows the incremental overhead of adding each layer of protection. All Fort-NoCs techniques have low overheads. On an average, DS adds 3.8% overhead, PC adds 2%, while NObf adds 0.01%

Table 6.4: Overhead of security schemes.

Metric	**Baseline**	**PC**	**DS**	**NoC-MPU**
Area (um^2)	4917	+0.34%	+9.57%	+42.00%
Power (mw)	1.705	+0%	+5.08%	+64.29%

increase in latency. The reason NObf adds minimal overhead on an average is that some of the threads experience degradation, while some experience improvement due to application-dependent traffic patterns. This correlation between thread location and performance was also observed by Misler and Jerger [93]. Between DS and PC, PC has a smaller overhead because the size of the certificate to be checked and decrypted is much smaller compared to the size of the data flits. Combining the first two layers of security yields 5.8% overhead. Fort-NoCs's overhead (3-layers) is at 5.9%. NoC-MPU's overhead is larger (8.03%) because when an access to the PLB misses, it needs to wait for a PLB refill before authenticating the current transaction. On an average, Fort-NoCs's 3-layer security system has 26% less overhead compared to NoC-MPU.

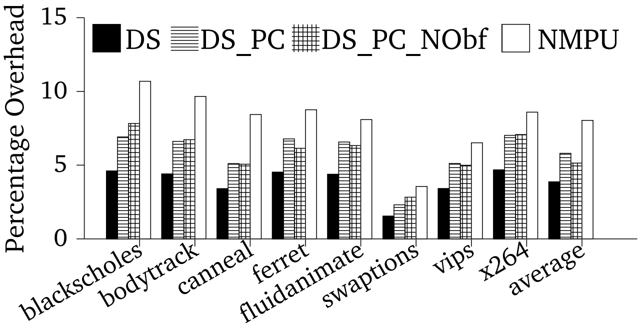


Fig. 6.9: Performance overhead.

Chapter 7

Conclusion

This dissertation has covered different design techniques to implement reliable and secure Network-On-Chip architectures. The techniques presented by this work tackle reliability and security by attacking the problem on many different abstraction levels. This holistic approach allows us to gain power-performance improvements that are otherwise hard to achieve when focusing only on one aspect of a system.

Using analysis from both architecture and RTL level, this work found out that NoC architectures running parallel programs produce communication patterns that lead to unbalanced HCI degradation through asymmetrical gate switching activity. This research exploited this property and presented four novel proposals in HCI mitigation in the crossbar circuit, a major component in an NoC router which dictates the operating frequency of the network. Overall, the proposed schemes that tackle HCI degradation distribute the switching activity, improve the clock cycle degradation, energy delay product per flit and system performance.

In the realm of exascale computing, this work also explored the tradeoffs of increasing system reliability while enforcing QoS guarantees in an NoC. Two novel DWRR algorithms were introduced, FR and LR. The proposed algorithms are guided by a low-cost NHM and a broadcast-based profiling and configuration. These proposed schemes boost the NoC MTTF on top of VICIS by 25%. Throughout an NoC's lifetime, the algorithms demonstrate an average of 9.3% QoS improvement by prolonging the lifetime of the high performance routers through efficient redistribution of QoS-induced traffic stress across the network.

Heterogeneous NoCs, a power-efficient alternative to the traditional NoC design has inherent power and resource asymmetry that leads to unbalanced aging degradation. In this context, this work also introduced a novel aging-aware routing algorithm based on architecture-level criticality information for *hNoCs* that minimizes the performance overheads caused due to NBTI using a dynamic routing algorithm. The algorithm deflects non-critical packets based on the degradation of the routers. As the buffered routers are most affected by NBTI, their minimal utilization through the routing algorithm improves latency, system performance and EDPPF by 38%, 53% and 29%, respectively, as compared to BRAR routing.

Security is also fast becoming a first-class design metric in modern chips. Systems that use NoCs are especially susceptible because a compromised central communication system can have huge implications on system security. Specifically, NoCs designed by third-party entities may contain IP blocks with hardware trojans. Such Compromised-NoCs can coordinate an attack with an accomplice software thread to pose a potent threat for the emerging computing platforms. In light of these trends, this dissertation proposed Fort-NoCs, implemented by augmenting the SoC firmware, that mitigates the threat of a C-NoC through a layered approach. Through a rigorous evaluation methodology, it has been demonstrated that Fort-NoCs prevents information leaking attacks in NoC at a modest overhead.

References

[1] Y. Hoskote, S. R. Vangal, A. Singh, N. Borkar, and S. Borkar, "A 5-ghz mesh interconnect for a teraflops processor," *IEEE Micro*, vol. 27, no. 5, pp. 51–61, 2007.

[2] D. Wentzlaff, P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Ramey, M. Mattina, C.-C. Miao, J. Brown, and A. Agarwal, "On-chip interconnection architecture of the tile processor," *Micro, IEEE*, pp. 15–31, Sept.-Oct. 2007.

[3] K. Aisopos, A. DeOrio, L.-S. Peh, and V. Bertacco, "Ariadne: Agnostic reconfiguration in a disconnected network environment," in *IEEE Parallel Architectures and Compilation Techniques*, pp. 298–309, 2011.

[4] A. Hosseini, T. Ragheb, and Y. Massoud, "A fault-aware dynamic routing algorithm for on-chip networks," in *IEEE International Symposium on Circuits and Systems*, pp. 2653–2656, 2008.

[5] F. Chaix, D. Avresky, N.-E. Zergainoh, and M. Nicolaidis, "A fault-tolerant deadlock-free adaptive routing for on chip interconnects," in *IEEE/ACM Design Automation & Test in Europe*, pp. 909–912, 2011.

[6] W.-C. Tsai, D.-Y. Zheng, S.-J. Chen, and Y. H. Hu, "A fault-tolerant noc scheme using bidirectional channel," in *IEEE/ACM Design Automation Conference*, pp. 918–923, 2011.

[7] Z. Zhang, A. Greiner, and S. Taktak, "A reconfigurable routing algorithm for a fault-tolerant 2d-mesh network-on-chip," in *IEEE/ACM Design Automation Conference*, pp. 441–446, 2008.

[8] J. Lee, M. C. Ng, and K. Asanovic, "Globally-synchronized frames for guaranteed quality-of-service in on-chip networks," in *International Symposium on Computer Architecture*, pp. 89–100, 2008.

[9] D. Fick, A. DeOrio, J. Hu, V. Bertacco, D. Blaauw, and D. Sylvester, "Vicis: a reliable network for unreliable silicon," in *IEEE/ACM Design Automation Conference*, pp. 812–817, 2009.

[10] H. Kufluoglu, *Mosfet Degradation due to NBTI and HCI and Its Implications for Reliability-Aware VLSI Design*, Ph.D. dissertation, Purdue University, West Lafayette, IN, 2007.

[11] D. Lorenz, M. Barke, and U. Schlichtmann, "Aging analysis at gate and macro cell level," in *IEEE International Conference on Computer-Aided Design*, pp. 77–84, 2010.

[12] T. Nigam, B. Parameshwaran, and G. Krause, "Accurate product lifetime predictions based on device-level measurements," in *IEEE International Reliability Physics Symposium*, pp. 634 –639, 2009.

[13] P. Kundu, “On-die interconnects for next generation cmps,” in *IEEE Workshop on On-Chip Interconnection Network*, pp. 1–26, 2006.

[14] A. K. Mishra, N. Vijaykrishnan, and C. R. Das, “A case for heterogeneous on-chip interconnects for cmps,” in *International Symposium on Computer Architecture*, pp. 389–400, 2011.

[15] H. Zhao, M. Kandemir, W. Ding, and M. J. Irwin, “Exploring heterogeneous noc design space,” in *IEEE International Conference on Computer-Aided Design*, pp. 787–793, 2011.

[16] J. Porquet, A. Greiner, and C. Schwarz, “Noc-mpu: a secure architecture for flexible co-hosting on shared memory mpsocs,” in *IEEE/ACM Design Automation & Test in Europe*, pp. 1–4, 2011.

[17] J. Yin, P. Zhou, A. Holey, S. S. Sapatnekar, and A. Zhai, “Energy-efficient non-minimal path on-chip interconnection network for heterogeneous systems,” in *ACM International Symposium on Low Power Electronic Devices*, pp. 57–62, 2012.

[18] Z. Li, J. Wu, L. Shang, R. P. Dick, and Y. Sun, “Latency criticality aware on-chip communication,” in *IEEE/ACM Design Automation & Test in Europe*, pp. 1052–1057, 2009.

[19] K. Bhardwaj, K. Chakraborty, and S. Roy, “Towards graceful aging degradation in nocs through an adaptive routing algorithm,” in *IEEE/ACM Design Automation Conference*, pp. 382–391, 2012.

[20] X. Fu, T. Li, and J. A. B. Fortes, “Architecting reliable multi-core network-on-chip for small scale processing technology,” in *IEEE Dependable Systems and Networks*, pp. 111–120, 2010.

[21] J. Kim, C. Nicopoulos, D. Park, V. Narayanan, M. Yousif, and C. Das, “A gracefully degrading and energy-efficient modular router architecture for on-chip networks,” in *International Symposium on Computer Architecture*, pp. 4–15, 2006.

[22] D. Park, C. Nicopoulos, J. Kim, N. Vijaykrishnan, and C. R. Das, “Exploring fault-tolerant network-on-chip architectures,” in *IEEE Dependable Systems and Networks*, pp. 93–104, 2006.

[23] T. Moscibroda and O. Mutlu, “A case for bufferless routing in on-chip networks,” in *International Symposium on Computer Architecture*, pp. 196–207, 2009.

[24] N. Abeyratne, R. Das, Q. Li, K. Sewell, B. Giridhar, R. Dreslinski, D. Blaauw, and T. Mudge, “Scaling towards kilo-core processors with asymmetric high radix topologies,” in *Proceedings of High Performance Computer Architecture*, pp. 89–101, 2013.

[25] B. Grot, J. Hestness, S. W. Keckler, and O. Mutlu, “Kilo-noc: a heterogeneous network-on-chip architecture for scalability and service guarantees,” in *International Symposium on Computer Architecture*, pp. 401–412, 2011.

[26] B. Grot, S. W. Keckler, and O. Mutlu, “Preemptive virtual clock: a flexible, efficient, and cost-effective qos scheme for networks-on-chip,” in *IEEE/ACM International Symposium on Microarchitecture*, pp. 268–279, 2009.

[27] C.-L. Chou and R. Marculescu, “Farm: Fault-aware resource management in noc-based multiprocessor platforms,” in *IEEE/ACM Design Automation & Test in Europe*, pp. 673–678, 2011.

[28] Y.-C. Lan, M. Chen, W.-D. Chen, S.-J. Chen, and Y.-H. Hu, “Performance-energy tradeoffs in reliable nocs,” in *Quality of Electronic Design, 2009. ISQED 2009. Quality Electronic Design*, pp. 141–146, 2009.

[29] R. Parikh and V. Bertacco, “Formally enhanced runtime verification to ensure noc functional correctness,” in *IEEE/ACM International Symposium on Microarchitecture*, pp. 410–419, 2011.

[30] A. Prodromou1, A. Panteli1, C. Nicopoulos1, and Y. Sazeides, “Nocalert: an online and real-time fault detection mechanism for network-on-chip architectures,” in *IEEE/ACM International Symposium on Microarchitecture*, pp. 60–71, 2012.

[31] S. Gupta, S. Feng, A. Ansari, J. Blome, and S. Mahlke, “Stagenetslice: a reconfigurable microarchitecture building block for resilient cmp systems,” pp. 1–10, 2008.

[32] T. Siddiqua and S. Gurumurthi, “Enhancing nbti recovery in sram arrays through recovery boosting,” *IEEE Transactions on VLSI Systems*, vol. PP, no. 99, p. 1, 2011.

[33] L. Fiorin, G. Palermo, S. Lukovic, V. Catalano, and C. Silvano, “Secure memory accesses on networks-on-chip,” *IEEE Transactions on Computers*, vol. 57, no. 9, pp. 1216–1229, 2008.

[34] J.-P. Diguet, S. Evain, R. Vaslin, G. Gogniat, and E. Juin, “Noc-centric security of reconfigurable soc,” in *NOCS*, pp. 223–232, 2007.

[35] C. H. Gebotys and R. J. Gebotys, “A framework for security on NoC technologies,” in *IEEE Computer Society Annual Symposium on VLSI*, pp. 113–117, 2003.

[36] H. K. Kapoor, G. B. Rao, S. Arshi, and G. Trivedi, “A security framework for noc using authenticated encryption and session keys,” *Circuits, Systems, and Signal Processing*, pp. 1–18, 2013.

[37] H. M. G. Wassel, Y. Gao, J. Oberg, T. Huffmire, R. Kastner, F. T. Chong, and T. Sherwood, “Surfnoc: a low latency and provably non-interfering approach to secure networks-on-chip,” in *International Symposium on Computer Architecture*, pp. 583–594, 2013.

[38] Y. Wang and G. E. Suh, “Efficient timing channel protection for on-chip networks,” in *ACM/IEEE International Symposium on Networks-on-Chip*, pp. 142–151, 2012.

[39] S. T. King, J. Tucek, A. Cozzie, C. Grier, W. Jiang, and Y. Zhou, “Designing and implementing malicious hardware.” in *Proceedings of 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*, vol. 8, pp. 1–8, 2008.

[40] S. Borkar, "Thousand core chipsa technology perspective," in *IEEE/ACM Design Automation Conference*, pp. 746–749, 2007.

[41] K. Bhardwaj, K. Chakraborty, and S. Roy, "An milp based aging aware routing algorithm for nocs," in *IEEE/ACM Design Automation & Test in Europe*, pp. 326–331, 2012.

[42] T. Siddiqua and S. Gurumurthi, "Enhancing nbti recovery in sram arrays through recovery boosting," *IEEE Transactions on VLSI Systems*, vol. 20, no. 4, pp. 616–629, 2012.

[43] Sonics Inc. (2008). SonicsMX: Smart interconnect security datasheet. [Online]. Available: http://www.sonicsinc.com

[44] T. Alves and D. Felton, "Trustzone: integrated hardware and software security," *ARM white paper*, vol. 3, no. 4, pp. 18–24, 2004.

[45] R. Das, O. Mutlu, T. Moscibroda, and C. Das, "Application-aware prioritization mechanisms for on-chip networks," in *IEEE/ACM International Symposium on Microarchitecture*, pp. 280–291, 2009.

[46] R. Das, O. Mutlu, T. Moscibroda, and C. R. Das, "Aergia: exploiting packet latency slack in on-chip networks," in *International Symposium on Computer Architecture*, pp. 106–116, 2010.

[47] J. F. Cantin, J. E. Smith, M. H. Lipasti, A. Moshovos, and B. Falsafi, "Coarse-grain coherence tracking: RegionScout and Region Coherence Arrays," *IEEE Micro*, vol. 26, no. 1, pp. 70–79, 2006.

[48] E. Karl, P. Singh, D. Blaauw, and D. Sylvester, "Compact in-situ sensors for monitoring negative-bias-temperature-instability effect and oxide degradation," in *International Solid-State Circuits Conference*, pp. 410 –623, 2008.

[49] D. Becker. (2012, August). Open source noc router rtl. [Online]. Available: https://nocs.stanford.edu/cgi-bin/trac.cgi/wiki/Resources/Router

[50] S. Bhardwaj, W. Wang, R. Vattikonda, Y. Cao, and S. Vrudhula, "Predictive modeling of the NBTI effect for reliable design," in *Proceedings of IEEE Custom Integrated Circuits Conference*, pp. 189–192, 2006.

[51] H. Chang and S. S. Sapatnekar, "Statistical timing analysis considering spatial correlations using a single pert-like traversal," in *IEEE International Conference on Computer-Aided Design*, pp. 621–626, 2003.

[52] B. Datta and W. Burleson, "Analysis and mitigation of nbti-impact on pvt variability in repeated global interconnect performance," in *ACM Great Lakes Symposium on VLSI*, pp. 341–346, 2010.

[53] W. Wang, Z. Wei, S. Yang, and Y. Cao, "An efficient method to identify critical gates under circuit aging," in *IEEE International Conference on Computer-Aided Design*, pp. 735 –740, 2007.

[54] A. Maheshwari and W. Burleson, “Current sensing techniques for global interconnects in very deep submicron (vdsm) cmos,” in *IEEE Computer Society Workshop on VLSI*, pp. 66 –70, 2001.

[55] W. Zhao and Y. Cao. (2012, June). Predictive technology model. [Online]. Available: http://ptm.asu.edu/

[56] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks.* San Francisco, CA: Morgan Kaufmann, 2004.

[57] C. Sun, C.-H. Chen, G. Kurian, L. Wei, J. Miller, A. Agarwal, L.-S. Peh, and V. Stojanovic, “Dsent - a tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling,” in *ACM/IEEE International Symposium on Networks-on-Chip*, pp. 201–210, 2012.

[58] M. Kamal, M. P. Qing Xie, A. Afzali-Kusha, and S. Safari, “An efficient reliability simulation flow for evaluating the hot carrier injection effect in cmos vlsi circuits,” in *IEEE International Conference on Computer Design*, pp. 352–357, 2012.

[59] Seyab and S. Hamdioui, “Nbti modeling in the framework of temperature variation,” in *IEEE/ACM Design Automation & Test in Europe*, pp. 283 –286, 2010.

[60] J. Srinivasan, S. Adve, P. Bose, and J. Rivers, “Lifetime reliability: toward an architectural solution,” *IEEE/ACM International Symposium on Microarchitecture*, pp. 70–80, 2005.

[61] W. Wang, V. Reddy, A. Krishnan, R. Vattikonda, S. Krishnan, and Y. Cao, “Compact modeling and simulation of circuit reliability for 65-nm cmos technology,” *IEEE Transactions on Device and Materials Reliability*, vol. 7, no. 4, pp. 509 –517, 2007.

[62] J. Hestness, B. Grot, and S. W. Keckler, “Netrace: dependency-driven trace-based network-on-chip simulation,” in *ACM/IEEE International Symposium on Networks-on-Chip*, pp. 31–36. ACM, 2010.

[63] W. Huang, M. R. Stan, K. Skadron, K. Sankaranarayanan, S. Ghosh, and S. Velusamy, “Compact thermal modeling for temperature-aware design,” in *IEEE/ACM Design Automation Conference*, pp. 878–883, 2004.

[64] F. Oboril, F. Firouzi, S. Kiamehr, and M. Tahoori, “Reducing nbti-induced processor wearout by exploiting the timing slack of instructions,” in *IEEE/ACM International Conference on Hardware/software Codesign and System Synthesis*, pp. 443–452, 2012.

[65] E. Takeda and N. Suzuki, “An empirical model for device degradation due to hot-carrier injection,” *IEEE Electron Device Letters*, vol. 4, no. 4, pp. 111 – 113, 1983.

[66] A. Bravaix, C. Guerin, V. Huard, D. Roy, J. Roux, and E. Vincent, “Hot-carrier acceleration factors for low power management in dc-ac stressed 40nm nmos node at high temperature,” in *IEEE International Reliability Physics Symposium*, pp. 531 –548, 2009.

[67] C. Bertolini, O. Heron, N. Ventroux, and F. Marc, "Relation between hci-induced performance degradation and applications in a risc processor," *IEEE On-Line Testing Symposium*, pp. 67–72, 2012.

[68] R. Das, A. Mishra, C. Nicopoulos, D. Park, V. Narayanan, R. Iyer, M. Yousif, and C. Das, "Performance and power optimization through data compression in network-on-chip architectures," in *Proceedings of High Performance Computer Architecture*, pp. 215–225, 2008.

[69] N. K. Choudhary, S. V. Wadhavkar, T. A. Shah, H. Mayukh, J. Gandhi, B. H. Dwiel, S. Navada, H. H. Najaf-abadi, and E. Rotenberg, "FabScalar: composing synthesizable rtl designs of arbitrary cores within a canonical superscalar template," in *International Symposium on Computer Architecture*, pp. 11–22, 2011.

[70] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The parsec benchmark suite: characterization and architectural implications," in *IEEE Parallel Architectures and Compilation Techniques*, pp. 72–81, 2008.

[71] N. Binkert, B. Beckmann, 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," *SIGARCH Computer Architecture News*, vol. 39, no. 2, pp. 1–7, Aug. 2011.

[72] M. Mints and R. Ekendahl, *Hardware Verification with C++: A Practitioners Handbook*, 1st ed. New York, NY: Springer, 2006.

[73] A. Basu, J. Gandhi, J. Chang, M. D. Hill, and M. M. Swift, "Efficient virtual memory for big memory servers," in *International Symposium on Computer Architecture*, pp. 237–248, 2013.

[74] P. Lotfi-Kamran, B. Grot, and B. Falsafi, "Noc-out: Microarchitecting a scale-out processor," in *IEEE/ACM International Symposium on Microarchitecture*, pp. 177–187, 2012.

[75] D. Fick, N. Liu, Z. Foo, M. Fojtik, J. sun Seo, D. Sylvester, and D. Blaauw, "In situ delay-slack monitor for high-performance processors using an all-digital self-calibrating 5ps resolution time-to-digital converter," in *International Solid-State Circuits Conference*, pp. 188–189, 2010.

[76] S. Das, C. Tokunaga, S. Pant, W.-H. Ma, S. Kalaiselvan, K. Lai, D. Bull, and D. Blaauw, "RazorII: In situ error detection and correction for PVT and SER tolerance," *Journal of Solid-State Circuits*, vol. 44, no. 1, pp. 32–48, Jan. 2009.

[77] P. Gratz, B. Grot, and S. W. Keckler, "Regional congestion awareness for load balance in networks-on-chip," in *Proceedings of High Performance Computer Architecture*, pp. 203–214, 2008.

[78] M. Ramakrishna, P. V. Gratz, and A. Sprintson, "GCA: Global congestion awareness for load balance in networks-on-chip," in *ACM/IEEE International Symposium on Networks-on-Chip*, pp. 1–8, 2013.

[79] T. Grasser, B. Kaczer, W. Goes, H. Reisinger, T. Aichinger, P. Hehenberger, P. J. Wagner, F. Schanovsky, J. Franco, M. Luque, and M. Nelhiebel, "The paradigm shift in understanding the bias temperature instability: From reaction-diffusion to switching oxide traps," *IEEE Transactions on Electron Devices*, vol. 58, no. 11, pp. 3652–3666, 2011.

[80] E. A. H. El Amir, "On uses of mean absolute deviation: decomposition, skewness and correlation coefficients," *International Journal of Statistics*, vol. 70, no. 2-3, pp. 145–164, 2012.

[81] S. Li, K. Lim, P. Faraboschi, J. Chang, P. Ranganathan, and N. P. Jouppi, "System-level integrated server architectures for scale-out datacenters," in *IEEE/ACM International Symposium on Microarchitecture*, pp. 260–271. ACM, 2011.

[82] K. Shuler. (2013, August). Majority of leading china semiconductor companies rely on arteris network-on-chip interconnect ip. [Online]. Available: www.arteris.com/China_Majority_Arteris_pr_19_august_2013

[83] ——. (2013, August). Arteris makes big gains on inc. 500 list of america's fastest-growing private companies. [Online]. Available: www.arteris.com/Inc-500-Arteris-pr-2013-august-20

[84] OCP International Partnership. (2009). Open Core Protocol Specification: Release 3.0. [Online]. Available: www.ocpip.org

[85] R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor, "Trustworthy hardware: Identifying and classifying hardware trojans," *IEEE Computer*, vol. 43, no. 10, pp. 39–46, 2010.

[86] A. Pellegrini, V. Bertacco, and T. Austin, "Fault-based attack of rsa authentication," in *IEEE/ACM Design Automation & Test in Europe*, pp. 855–860, 2010.

[87] R. Uhlig, G. Neiger, D. Rodgers, A. L. Santoni, F. C. M. Martins, A. V. Anderson, S. M. Bennett, A. Kägi, F. H. Leung, and L. Smith, "Intel virtualization technology," *Computer*, vol. 38, no. 5, pp. 48–56, 2005.

[88] Advanced Micro Devices. (2008, April). Live migration with AMD-V extended machine migration. [Online]. Available: http://developer.amd.com/wordpress/media/2012/10/43781-3.00-PUB_Live-Virtual-Machine-Migration-on-AMD-processors.pdf

[89] M. Rostami, F. Koushanfar, J. Rajendran, and R. Karri, "Hardware security: Threat models and metrics," in *IEEE International Conference on Computer-Aided Design*, pp. 109–118. ACM, 2013.

[90] P. C. Kocher, J. Jaffe, B. Jun, and P. Rohatgi, "Introduction to differential power analysis," *Journal of Cryptographic Engineering*, vol. 1, no. 1, pp. 5–27, 2011.

[91] D. Bovet and M. Cesati, *Understanding the Linux Kernel - from I/O Ports to Process Management: Covers Version 2.6 (3rd ed.).* Sebastopol, CA: O'Reilly, 2005.

[92] R. P. Gudla, “Design and implementation of clocked ocp interfaces between ip cores and on-chip network fabric,” Master’s thesis, University of Utah, Salt Lake City, UT, 2011.

[93] M. Misler and N. D. E. Jerger, “Moths: Mobile threads for on-chip networks,” *ACM Transactions in Embedded Computing Systems*, vol. 12, no. 1s, p. 56, 2013.

Vita

Dean Michael B Ancajas

Published Journal Articles

- Wearout Resilient Network-On-Chip Architecture. Dean M Ancajas, Kshitij Bhardwaj, Koushik Chakraborty and Sanghamitra Roy, *IEEE Transactions on Very Large Scale Integration 2014 (TVLSI), Accepted.*

- Exploring High Throughput Computing Paradigm in Global Routing. Yiding Han, Dean M Ancajas, Koushik Chakraborty and Sanghamitra Roy, *IEEE Transactions on Very Large Scale Integration 2012 (TVLSI), Accepted.*

Published Conference Papers

- Tackling QoS-induced Aging in Exascale Systems. Dean M Ancajas, Koushik Chakraborty, Sanghamitra Roy and Jason Allred. *Proc. IEEE Intl. Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, May 2002.

- Fort-NoCs: Mitigating the Threat of Compromised-NoC. Dean M Ancajas, Koushik Chakraborty and Sanghamitra Roy. *IEEE/ACM 51st Design Automation Conference (DAC)*. June 2014.

- HCI-Tolerant NoC Router Microarchitecture. Dean M Ancajas,James McCabe Nickerson Koushik Chakraborty and Sanghamitra Roy. *IEEE/ACM 50th Design Automation Conference (DAC)*. June 2013.

- DMR3D: Dynamic Memory Relocation for 3D-Multicore Processors. Dean M Ancajas, Koushik Chakraborty and Sanghamitra Roy. *IEEE/ACM 50th Design Automation Conference (DAC)*. June 2013.

- Efficiently Tolerating Timing Violations in Pipelined Microprocessors. Dean M Ancajas, Koushik Chakraborty, Brian Cozzens, Sanghamitra Roy and Dean M Ancajas. *IEEE/ACM 50th Design Automation Conference (DAC)*. June 2013.

- Proactive Aging Management in Heterogeneous Network-on-Chip Architectures. Dean M Ancajas, Koushik Chakraborty and Sanghamitra Roy. *IEEE/ACM Design Automation and Test in Europe* . March 2013.

- Mitigating NBTI in the Physical Register File through Stress Prediction. Saurabh Kothawade, Dean M Ancajas, Koushik Chakraborty and Sanghamitra Roy. *IEEE/ACM 30th Internation Conference on Computer Design (ICCD)*. Sep 2012.

- Exploring High Throughput Computing for Global Routing. Yiding Han, Dean M Ancajas, Koushik Chakraborty and Sanghamitra Roy. *IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. November 2011.

- Dual-Core Capability of DLX Microprocessors. Dean M Ancajas et al. . *IEEE Region 10 Conference (TENCON)*. November 2007.