EXPLORING POWER RELIABILITY
TRADEOFFS IN ON-CHIP NETWORKS

A Dissertation in
Computer Science and Engineering
by
Aditya Yanamandra

© 2010 Aditya Yanamandra

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2010
The dissertation of Aditya Yanamandra was read and approved\(^1\) by the following:

Mary Jane Irwin  
Evan Pugh Professor of Computer Science and Engineering  
A. Robert Noll Chair in Engineering  
Dissertation Co-Advisor  
Co-Chair of Committee

Vijaykrishnan Narayanan  
Professor of Computer Science and Engineering  
Dissertation Co-Advisor  
Co-Chair of Committee

Mahmut Taylan Kandemir  
Professor of Computer Science and Engineering

Padma Raghavan  
Professor of Computer Science and Engineering  
Director of Institute of CyberScience

Vittal Prabhu  
Professor of Industrial and Manufacturing Engineering

Raj Acharya  
Department Head of Computer Science and Engineering

\(^1\)Signatures on file in the Graduate School.
Abstract

The past decade has seen a shift in the arms race of the microprocessor industry from increasing the clock frequency to increasing the number of cores on chip. Power and performance considerations have motivated the adoption of chip multiprocessors (CMPs). As a result, there is an emphasis on the performance of the on-chip interconnection network that connect these cores. Bus based designs dominated the interconnection network in the industry so far. They are early indications of buses being replaced with a packet based Network-on-Chip (NoC) due to the scalability issues of on-chip buses.

Reliability concerns such as soft errors, process variation and premature transistor failure threaten future technology scaling. NoCs are beset by the same challenges as well. The distributed nature of the NoC provides for a different approach for solving these issues. In this dissertation, we address several phenomena that cause errors in the Network on Chip domain. We also address the reliability impact of several techniques proposed in the literature to improve the performance and power of the NoC.

NoC power consumption has been identified as a chief limiter to adoption in the industry. Providing reliability costs area and power on the chip. For example, error correction codes (ECC) are employed to mitigate the effect of soft errors on NoC. Such power requirements place an additional burden on the chip designers. In this dissertation, we propose techniques to reduce the power required to maintain iso-reliable systems. Reducing the power required to maintain reliable NoCs is an important factor in achieving the goal of highly reliable low-power NoCs. Techniques proposed in this dissertation aim to achieve such a system.
# Table of Contents

List of Tables ........................................................................................................ viii
List of Figures ......................................................................................................... ix
Acknowledgments .................................................................................................. xii
Chapter 1. Introduction ......................................................................................... 1

Chapter 2. Optimizing Power and Performance for Reliable On-Chip Network . 8
  2.1 Introduction .................................................................................................. 8
  2.2 Preliminaries ............................................................................................... 11
    2.2.1 Support for Error Correction ............................................................... 11
  2.3 Power Reliability Trade-off ........................................................................ 12
    2.3.1 Modeling Residual Error Rate with Error Correction ..................... 12
    2.3.2 Modeling Residual Error Rate with Architectural Vulnerability Factor . 13
    2.3.3 Power Equations ................................................................................ 14
  2.4 Methodology ............................................................................................... 15
    2.4.1 Static Single AVF (SAVF) Throttling ................................................ 16
    2.4.2 Multiple AVF Domains (MAVF) ....................................................... 16
  2.5 Bounding AVF via throttling ...................................................................... 18
  2.6 Experimental Setup .................................................................................... 19
  2.7 Static SAVF Throttling results .................................................................... 20
    2.7.1 Power Savings .................................................................................... 20
    2.7.2 Throughput Loss ................................................................................ 22
  2.8 Multiple AVF Domains .............................................................................. 23
    2.8.1 Static MAVF ...................................................................................... 23
    2.8.2 Dynamic MAVF ................................................................................ 24
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.9 Application Results</td>
<td>24</td>
</tr>
<tr>
<td>2.10 Related Work</td>
<td>25</td>
</tr>
<tr>
<td>2.11 Conclusion</td>
<td>26</td>
</tr>
<tr>
<td>Chapter 3. Reliability Aware Performance and Power Optimization in DVFS-based On-chip Networks</td>
<td>28</td>
</tr>
<tr>
<td>3.1 Introduction</td>
<td>28</td>
</tr>
<tr>
<td>3.2 Network Architecture</td>
<td>30</td>
</tr>
<tr>
<td>3.2.1 Base Router Architecture</td>
<td>30</td>
</tr>
<tr>
<td>3.2.2 Support for Error Correction</td>
<td>31</td>
</tr>
<tr>
<td>3.3 Implementation of DVFS on NoCs</td>
<td>31</td>
</tr>
<tr>
<td>3.4 Reliability Considerations under Dynamic Variations of Voltage</td>
<td>33</td>
</tr>
<tr>
<td>3.4.1 Error Rate Variation</td>
<td>33</td>
</tr>
<tr>
<td>3.4.2 Error Correction Power</td>
<td>35</td>
</tr>
<tr>
<td>3.5 Experimental Setup</td>
<td>37</td>
</tr>
<tr>
<td>3.6 Experimental Results</td>
<td>38</td>
</tr>
<tr>
<td>3.6.1 Performance Benefits</td>
<td>38</td>
</tr>
<tr>
<td>3.6.2 Power Benefits</td>
<td>39</td>
</tr>
<tr>
<td>3.6.2.1 Experiments with Threshold Levels</td>
<td>40</td>
</tr>
<tr>
<td>3.6.2.2 Varying Network Size</td>
<td>41</td>
</tr>
<tr>
<td>3.6.2.3 Varying Traffic Patterns</td>
<td>42</td>
</tr>
<tr>
<td>3.6.2.4 Varying Injection Rate</td>
<td>42</td>
</tr>
<tr>
<td>3.7 Application Results</td>
<td>43</td>
</tr>
<tr>
<td>3.8 Related Work</td>
<td>43</td>
</tr>
<tr>
<td>3.9 Conclusion</td>
<td>45</td>
</tr>
<tr>
<td>Chapter 4. In-Network Caching for Chip Multiprocessors</td>
<td>47</td>
</tr>
<tr>
<td>4.1 Introduction</td>
<td>47</td>
</tr>
<tr>
<td>4.2 Motivation and Background</td>
<td>49</td>
</tr>
<tr>
<td>4.2.1 In Network Cache Coherence</td>
<td>50</td>
</tr>
<tr>
<td>4.3 In-Network Data Caching</td>
<td>52</td>
</tr>
</tbody>
</table>
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>On chip regulator power consumption</td>
<td>32</td>
</tr>
<tr>
<td>3.2</td>
<td>ECC scheme power, delay and area</td>
<td>35</td>
</tr>
<tr>
<td>4.1</td>
<td>Simulation setup</td>
<td>62</td>
</tr>
<tr>
<td>5.1</td>
<td>Cache Configurations</td>
<td>78</td>
</tr>
<tr>
<td>5.2</td>
<td>DVFS Table</td>
<td>87</td>
</tr>
</tbody>
</table>
List of Figures

1.1 MIPS per Watt ........................................... 2
1.2 ITRS 2005 Interconnect ................................... 3
1.3 Base Router Architecture ................................. 4

2.1 Router pipeline with error correction support ............ 11
2.2 Average buffer utilization ................................ 16
2.3 Throttling in the router .................................. 19
2.4 Power savings .............................................. 20
2.5 Throughput loss ........................................... 21
2.6 MAVF results .............................................. 23
2.7 Application results ....................................... 25

3.1 Router pipeline ............................................ 29
3.2 Residual word error rate at varying supply voltages and error protection 34
3.3 (a)Percentage of power expended for ECC to the total network power and (b) 16-way CMP layout where MC is the memory controller .... 36
3.4 Average latency (per flit) improvement with DVFS ............ 39
3.5 Savings in EC power for various threshold levels ............ 41
3.6 Impact of network size on savings in EC power .............. 42
3.7 Impact of traffic patterns on power savings ................ 43
3.8 Impact of load rate on savings in EC power ................ 44
3.9 Power saved for multi-core benchmarks ........................ 45

4.1 Ratio of the number of private to shared addresses ............ 49
4.2 Ratio of the accesses to private and shared addresses ......... 49
4.3 Basic In-network cache coherence - Read operations ........... 50
4.4 Basic In-network cache coherence - Write operations ........... 51
4.5 Proposed In-network cache coherence - Read operations .......... 53
4.6 Proposed router microarchitecture .......................... 56
4.7 Area overhead - with TreeCache ........................................... 58
4.8 Area overhead - without TreeCache .................................... 58
4.9 Zonal caching implementation ............................................. 60
4.10 Latency improvement using ideal caching. ............................ 63
4.11 Latency improvement using independent caching. ................... 63
4.12 Zonal vs Independent 4-entry ............................................ 65
4.13 Normalized Zonal vs Independent 16-entry .......................... 65

5.1 Switching activity in unidirectional and bidirectional links ........ 73
5.2 Peak Temperature .............................................................. 75
5.3 Router to Core MTTF comparison ....................................... 76
5.4 CMP layout ................................................................. 76
5.5 Experimental setup flow .................................................... 77
5.6 Mesh - 32nm ................................................................. 79
5.7 Mesh - 45nm ................................................................. 79
5.8 Mesh - All links ............................................................. 80
5.9 Torus - 32nm ................................................................. 80
5.10 Torus - 45nm ............................................................... 81
5.11 Unidirectional and bidirectional links .................................. 81
5.12 Adaptive Bidirectional Links performance ............................ 83
5.13 Bidirectional MTTF 32nm ................................................. 83
5.14 Bidirectional MTTF 45nm ............................................... 84
5.15 DVFS Performance .......................................................... 85
5.16 DVFS MTTF - 32nm ....................................................... 86
5.17 DVFS MTTF - 45nm ....................................................... 86
5.18 AARP Performance degradation Unidirectional ....................... 92
5.19 AARP Performance degradation Bidirectional ......................... 92
5.20 AARP Performance degradation DVFS ................................ 93
5.21 AARP MTTF evaluation for links ....................................... 94
5.22 AARP MTTF evaluation for adaptive bidirectional links .......... 95
5.23 AARP MTTF evaluation for NoC with DVFS .......................... 96
6.1 Variations in Leakage Power of a Single Buffer Slot (128 bits) . . . . . . 99
6.2 Generic FIFO structure . . . . . . . . . . . . . . . . . . . . . . . . . . . 100
6.3 IntelliBuffer Structure . . . . . . . . . . . . . . . . . . . . . . . . . . . 101
6.4 Leakage Power Savings using IntelliBuffer. . . . . . . . . . . . . . . . . . 102
6.5 (a) ViChar Architecture , (b) Limitations of Existing FIFO Buffers and 
ViChaR’s Solution and (c) Slot Availability Tracker . . . . . . . . . . . 104
6.6 Leakage Power Savings in ViChar using IntelliBuffer . . . . . . . . . . 106
Acknowledgments

I am grateful to my advisors Dr. Vijaykrishnan Narayanan and Dr. Mary Jane Irwin for always challenging me and getting the best out of me during my PhD. Both of them have been great role models and mentors to me and I hope to represent them well wherever life may lead me. I had a great time working at my PhD which all in thanks to them. I was lucky enough to work with Dr. Mahmut Kandemir and Dr. Padma Raghavan during my formative years of PhD research. Their constant guidance through my first years of research made me realize that doctoral studies was an experience to be enjoyed and not a degree to be attained.

I was lucky to have a wonderful set of seniors and lab mates through the years of my stay at MDL. I wish I could thank each one of you in person someday. In particular, I would like to thank Dr. Sri Hari Krishna Narayanan for his help, friendship, and being the senior that every junior needs to keep things in perspective. Dr. Ramkrishnan Krishnan, your goofiness and enthusiasm for work made staying and working in the lab late in the night fun for everyone around you. Thanks to everyone who I ever worked with for you were all sounding boards that shaped my thoughts and inspired me to work harder.

It would be impossible to thank all my friends at Penn State. However, I would like to mention Dr. Shiva Chaitanya and Guru (my former roommates), Sai Prashanth(my cubicle mate), Dr. Rajaraman Ramanarayan, Dr. Sharoda Paul, Sriram, and Srikar. You made life at Penn State incredibly enjoyable. Basketball was an integral part of my life at PSU. Special mention to the members of the “baskigumbal”. I will surely miss lacing it up with you folks.

The three semesters I spent interning at Intel, MA were one of the highlights of my graduate life. Thanks to Shubu Mukhejee Paul Racunas, Arijit Biswas and the others at SPEARS group for their mentorship. The knowledge I obtained from interacting with you was far more than the technical content of my projects. Shubu, you encouraged me to aim higher and helped boost my self-confidence. The conversations we had and the
lessons I learnt from you invaluable. I made many friends during my internships and I would like to thank them all for making them so memorable. I spent a good part of my free time in the summer and fall of 2009 with Mohit Gambhir and Dr. Angshuman Parashar talking work, photography and playing Mass Effect. They are some of the best memories I have from the past five years and I will cherish them for a lifetime. Above all, I would like to thank mom, dad, Sailu and Soumya for your support through the ups and downs.
To,

Mom, Dad and Soumya
Chapter 1

Introduction

The push for faster computation led to large processors running at higher and higher frequencies. Some of this speed was achieved by deeply pipelining the processor. In turn, heavy speculation was employed to keep these pipelines full. While processors were being marketed for their frequency, architects saw obvious problems with this strategy. The main concern was the performance per watt achieved by these processors. This metric kept steadily decreasing with increasing processor complexity as shown in Figure 1.1. Power and performance considerations led to a shift in the philosophy of microprocessor industry. Chip-multi processors (CMP) were introduced in the early part of this (2000’s) decade to yield better performance per watt (67). The idea was to slow down the processor and regain the performance by means of parallelism. Multiple cores would be integrated on a single die and stitched together using a flexible and scalable communication medium.

In the year 2001, IBM released its commercial dual-core CPU, the IBM Power 4. In 2005, Sun came up with the Sun UltraSPARC T1 (codenamed Niagara) which was a 8-core CPU. Intel soon followed suit with its dual core and core duo processors. The current generation of commercial Intel Core i7 (Nehalem architecture) is a quad core on a single die and its 8-core version is expected soon (as of January 2010). In addition, Intel show cased the 80 core prototype (81) in September 2006, named Polaris, promising to usher in the “Era of the Tera”. We have now entered a phase of irreversible march towards an increase in the number on chip processing elements (PEs). It is predicted that by 2011 more than 90% of the commercial chips sold would be Chip Multi Processors. Projecting into the future, we can imagine a chip to have hundreds of simple cores. Communication between such large number of PEs is still an unresolved piece of the puzzle.
Further, the global interconnect is unable to keep pace with the rapid scaling of transistors. Figure 1.2 from ITRS shows the trend for the global interconnect. As the process technology node scales, the difference between the gate delay and the global delay is increasing. Thus, the future microprocessors will be communication centric and the interconnect delay will dominate their performance.

Buses were traditionally used to connect on-chip components. While buses that are broadcast based integrate well with on-chip coherence protocols and are simple to implement, they are not a scalable solution for future multicores. Buses are prone to crosstalk loss, signal integrity, and limited frequency. Further, buses were found to be a power inefficient solution for communicating between multiple cores on the die. The global interconnect delay would make it prohibitive to connect large number of PEs. Signaling an entire bus for each communication even when the destined communicating pair are adjacent was found to be expensive. A segmented bus was proposed to overcome this limitation ((14)). However, buses alone are not an adequate solution to the bandwidth demands and parallelism of modern multi-cores. Packet-switched networks were proposed as a communication solution on chip that can provide the required concurrency.
The network is comprised of several routers connected together by inter-router links of length equal to one or a few PEs. Such an architecture gave birth to the tiled CMP and simplified the layout of the network. Packet switched network enabled smarter flow control and higher throughput. In the following subsection, we provide the microarchitectural details of a typical network on chip (NoC) router.

**Router Microarchitecture**

We give a brief overview of an on-chip router architecture used as a baseline in most of this work. The generic router is a pipelined architecture with 5 stages namely Routing Computation (RC), Virtual channel Allocation (VA), Switch Allocation (SA), Switch Traversal (ST) and Link Traversal (LT) (24; 70) as shown in the figure 1.3.

The main components of the router are one RC unit per VC or physical port, input buffers (for a input buffered router) with one buffer for every physical port shared among the VCs, a NxN crossbar switch where N is the number of input/output ports, and the arbiters. In a mesh network, each router has 5 input and output ports/channels.
corresponding to the four cardinal directions and one injection/ejection port i.e. the local processing element (PE).

The RC unit is responsible for directing the incoming flit to the appropriate output physical channel. This is based on the destination information and based on the routing protocol (deterministic or adaptive). The RC stage is done only for the header flit. The rest of the flits follow the header flit in case of wormhole switching. (A packet consists of a header, zero or more body flits and a tail flit. A flit is the smallest unit of data that can be transmitted across the network.) The routing function could also resolve the virtual channel (R→PV) in which case the VA stage is simpler.

In the VA stage, the header flit arbitrates for a free virtual channel (VC) in the next router using (P*V V:1 (VA1) and P*V P*V:1 (VA2) where P is number of physical channels (PC) and V is the number of VCs per PC). In VA1, an each input virtual channel (P*V input virtual channels) choose an output VC. VA2 arbitrates the winners
of VA1 and chooses one winner for each output VC. If the routing function resolves the input-output virtual channel mapping then the VA1 stage can be omitted.

In the SA stage, every flit that won VA can arbitrate for the crossbar switch and only one flit can pass through an input port and output port combination in one cycle. The SA consists of two stages of arbiters (P V:1 and P P:1 arbiters). SA1 chooses one VC from each PC using P V:1 arbiters and the SA2 then chooses one input PC to be connected to an output PC using P P:1 arbiters. In ST stage the winners of SA traverse the crossbar in one cycle to reach the inter-router links. The LT stage accounts for the inter-router link traversal and writing the flit into the input buffer of the next router. Thus, RC and VA are done only for the header flit. SA, ST, LT are performed for all flits. Consequently, the header flit passes through 5 stages of pipeline while the body and tail flits pass through only 3 pipeline stages.

Enhancements to this basic router architecture have been proposed. The simplest of them is using look-ahead routing wherein the route is pre-computed before it arrives at the router and the current router computes the route to be taken by the packet after the next router. This results in a four cycle router with RC and VA being done in parallel, followed by SA, ST and LT stages. These router optimizations lead to performance benefits. In addition to performance, network power consumption is also a very important constraint for designing the NoC. With the process technology scaling, new design challenges such as process variations, soft errors and aging phenomena have to be considered during design in addition to the power and performance considerations.

The ever shrinking CMOS technology has made routers vulnerable to radiation induced soft errors. The raw error rate per device in a bulk CMOS process is projected to remain roughly constant (59). Successive processor technologies typically increase the number of devices on chip. This means that the susceptibility to soft errors is steadily increasing. To reverse this trend, expensive solutions such as SOI, or redundancy have to be employed. In the context of on-chip networks, to protect from data corruption due to soft errors, error correction (EC) schemes have to be employed. This consumes power and places an additional burden on the power budget. Chapter 2 and Chapter 3 address
this additional power and propose solutions to minimize the power cost of providing reliability.

Chapter 2 saves on the EC power for applications that don’t require the EC to be turned on all the time. Some applications can tolerate few errors such as applications in the RMS suite. We provide a framework by which the EC can be turned off partly while maintaining a desired reliability levels. Chapter 3 addresses the error rate in a DVFS based system. When power-saving techniques such as DVFS is used, the operating voltage and frequency are changed dynamically. At lower voltages, the probability of errors increase and thus, higher error protection levels are required for keeping the system error-free. However, these higher error protection schemes are more expensive and thus, they should be used wisely. We provide a reconfigurable framework of EC in DVFS based systems in order to minimize the cost of providing EC. Depending on the operating voltage, least-cost EC code that meets the reliability requirements is chosen. Apart from power and reliability tradeoffs, we also propose network architecture enhancement that leads to significant performance benefits. The idea is based on a simple observation that the quick access to small amount of frequently used shared data can lead to lucrative performance benefits. Effective communication of on-chip data is a critical factor for determining the performance of future chip multiprocessors. The shared data maybe tens of cycles away from the accessing nodes and this leaves vast rooms for optimizations in hardware, operating systems as well as the parallel programming models. Chapter 4 discusses an in-network caching mechanism to keep shared data close to the destination PEs. By caching data in the routers, the number of hops required to access performance critical data is reduced thus improving the performance of the system. Chapter 5 consider the permanent aging effects in NoCs. We motivate the unique nature of Electromigration, an interconnect failure, in the NoCs and propose models for unidirectional and bidirectional wires and evaluate the MTTF on NoC links. Finally, we propose Aging Aware Routing Protocols (AARP), a suite of routing techniques to mitigate the aging phenomena on the NoCs. In Chapter 6, we study the effects of process variation on the leakage power on NoC buffers. The NoC power consumption is dominated by the buffers and the leakage power is a major contributor. We propose IntelliBuffer, an intelligent
way to order and use the NoC buffers, in the order of their leakage power. IntelliBuffer reduces the leakage power of the buffers which in turn reduces the overall power of the NoC. Chapter 7 concludes the thesis.
Chapter 2

Optimizing Power and Performance for Reliable On-Chip Network

2.1 Introduction

Network-on-chip (NoC) has emerged as a solution for the interconnection fabric in Chip Multi Processor (CMP) market (81). With increasing number of cores being integrated on a single die, the NoC has a significant impact on the system performance, power and reliability. The low supply voltage and the miniature size of state-of-art CMOS transistors have made them more susceptible to transient errors caused by environmental impacts. Thus, protecting the communication fabric against these transient errors is required for maintaining the reliability of the system. Transient faults due to soft errors (SE) have become a major concern for designers in current and future technologies (3). Radiation-induced soft errors result from high-energy neutrons and alpha particles causing a single-event upset (SEU) that may alter the state of the system. These bit-flips, typically caused in the sequential logic elements such as the router buffers, memories, and registers may result in program malfunctioning. The router buffers in NoC are typically implemented as SRAM cells (45). SRAM based designs are popular for their speed and density considerations. However, they are vulnerable to soft errors (72).

Prior work on NoC reliability has shown that error correction schemes are much more power efficient than retransmission schemes. The power expended in error correction can be as high as 20% of the network power (61) in the current technology. Orion (83) power models confirmed that at 90nm, 22mW of power is expended by a single flit to traverse a mesh router and error correction power would be 4.5mW. ITRS predicts that in future technologies, reliability will become more critical and catering for the reliability requirements calls for scalable power and performance costs. In this
work, we explore techniques for achieving power-performance trade-offs at iso-reliability for the network fabric.

Modern systems are designed to work for a given target reliability. For example, IBM targets 1000 years system Mean Time Before Failure (MTBF) for undetectable errors and 25 years for detectable errors for its Power4 processor (16). Applications have varying reliability demands from the underlying hardware. This is based on the implicit fault tolerance of the application and the end-user requirements. Some applications, such as recognition benchmarks from the RMS suite (27) can tolerate errors in their data fields. Thus, the desired reliability guarantees can be achieved even if the error protection is switched off for a fraction of the time. The possible power savings by this are dependent on the application error tolerance.

Not all SEUs surface as program errors. For instance, a bit flip in a buffer that currently holds invalid data does not cause any observable error in the system. Architectural Vulnerability Factor (AVF) is defined as the probability that a transient error causes an observable error in the output (59). The AVF of most array structures, such as the router input buffers, are dependent on their utilization (59). By throttling the buffer utilization, we limit the AVF of the router which in turn bounds the vulnerability of the network to soft errors. Throttling enables us to save power consumed by the reliability component while providing guarantees on the soft error vulnerability of the network. The power savings vary depending on the AVF guarantees required by the application and increase for decreasing AVF guarantees. However, bounding AVF comes at the cost of loss in throughput of the network. We consider the impact on throughput based on the AVF bound, injection rate and different traffic patterns. It is important to note that we study the power-performance trade-offs in the network while maintaining constant reliability. For our analysis, we consider a packet switched network with virtual channel flow control, wormhole switching and deterministic routing.

Buffer utilization (and hence the AVF) of a router varies across the network based on the router location and the traffic pattern. Consequently, setting the same AVF bound across all the routers critically affects some routers, thus causing high throughput
losses. As a result, an optimal setting of the AVF bound can yield good performance-power trade offs. While the optimal choice of AVF settings is dependent on application characteristics, in this work, we provide alternate schemes where we divide the chip into domains and use different AVF bounds in different domains.

Our main contributions include

1. Analytical model for the target reliability as a function of the AVF and the percentage of time error protection is enabled. To the best of our knowledge, this is the first work to propose an analytical model for the vulnerability of the network-on-chip.

2. Applications can tolerate some failures. Exploiting this tolerance, we study the impact on throughput and power while providing reliability guarantees. We perform a thorough analysis and evaluate the impact of various network parameters and traffic patterns.

3. Three schemes, namely, Static Single AVF (Static SAVF), Static and Dynamic Multiple AVF (MAVF) domains are proposed and the power performance trade-offs are evaluated.

We explore both static and dynamic multiple AVF schemes and perform a comprehensive evaluation of the techniques over various network parameters. We conduct our analysis for both synthetic traffic and real application traces. We conclude that while, at low injection rates Static MAVF scheme suffices, at higher injection rates Dynamic MAVF scheme is required to achieve good power savings at small throughput losses. For application traffic, and a 10% error tolerance, Dynamic MAVF can result in up to 44% power savings for a mere 3% loss in throughput. The rest of the chapter is organized as follows. A brief background is presented in Section 2.2. The analytical modeling for residual error rate under AVF bounding is explained in Section 2.3. In Section 2.4 and 2.5, we discuss our methodology and architectural details respectively. Section 2.6 gives the details of the experimental setup followed by results in Section 2.7 and 2.8. Related work is presented in Section 2.10 and we conclude in Section 2.11.
2.2 Preliminaries

In this section, we give a brief overview of the baseline router architecture. Base router is a pipelined architecture with 4-stages namely Routing Computation (RC), Virtual channel Allocation (VA), Switch Allocation (SA), Switch Traversal (ST) and Link Traversal (LT). The RC and the VA stage run in parallel as shown in Figure 2.1. Only the header flits go through the RC and VA stages. SA, ST and LT stages are performed for all flits. The main components of the router are input buffers, RC unit, arbiters for VA and SA, and crossbar.

2.2.1 Support for Error Correction

In this study, we consider radiation induced transient errors in the routers. Such errors manifest as single bit upsets. Park (68) showed that hop-by-hop error correction is more effective than end-to-end correction. Hence we adopt a Single Error Correction (SEC) scheme at every hop in the network. An encoder and decoder pair are needed for providing error protection at the granularity of a flit in every router. The encoding and decoding take a cycle each (Sec 2.6). All flits are encoded and decoded at every hop.

We discuss the modifications to the 4-stage router pipeline required for supporting the protection schemes. The pipeline is shown in Figure 2.1. The decoding is done in parallel with the RC and VA stage for the header flit and in parallel to the SA stage for the body flits. If there is an error in the incoming flit, then it is corrected in this cycle and the work done in the parallel stages is discarded. In the RC and VA stages, routing information such as the output port and the virtual channel (VC) id are generated and
stored locally in the router. The RC stage routing information is generated within one cycle. The VA stage however, can span multiple cycles because it involves arbitration. Similarly, some of router optimizations, write information into the header packet. We protect the RC stage generated information and VA stage output independently using Triple Modular Redundancy (TMR). This RC and VA stage output are typically a small number of bits and thus, the overhead of using TMR is minimal. Thus, all routing information is protected at all times. In addition, the header packet is protected through the ECC circuitry at all times to avoid misrouting the packet.

2.3 Power Reliability Trade-off

In this section, we present a theoretical model for providing reliability guarantees while reducing power required for the protection schemes. In order to bound the reliability in the presence of an ECC mechanism, we need to be able to bound the undetected error rate. We first model the undetected error rate agnostic of the architecture and later include the architectural masking impacts by using AVF metric. The basic power model for the ECC component is also discussed.

2.3.1 Modeling Residual Error Rate with Error Correction

The ECC mechanisms can detect/correct a subset of errors. The fraction of errors that can be detected is called the coverage of the ECC. The undetected (or residual) error rate ($\lambda_{res}$) is related to the raw error rate ($\lambda_{raw}$) in the presence of an ECC with a coverage of ‘c’ by Equation 2.1

$$\lambda_{res} = (1 - c)\lambda_{raw}$$  \hspace{1cm} (2.1)

Let the duty period for the error correction represented as $\rho$ be defined as the fraction of time the error correction mechanism is enabled. As not all errors need to be corrected, depending on the target residual error rate, we can disable the error correction circuitry and save power. This results in a decrease in the effective coverage of the ECC by a
factor of $\rho$ and thus the residual error rate is given by,

$$\lambda_{res} = (1 - c \star \rho) \lambda_{raw}$$  \hspace{1cm} (2.2)

Let ‘$k$’ be the ratio of residual error rate to the raw error rate. $k$ is a metric for error tolerance. A higher $k$ would indicate tolerance to higher number of errors and vice versa.

$$k = \frac{\lambda_{res}}{\lambda_{raw}}$$  \hspace{1cm} (2.3)

From Equations 2.2 and 2.3 we get,

$$\rho = \frac{1 - k}{c}$$  \hspace{1cm} (2.4)

The raw error rate is dependent on the operating conditions, and environmental conditions. For example, the raw error rate is higher at high altitudes. Thus, the raw error rate can vary during the lifetime of a single application. The residual error rate is, however application dependent. Thus, $k$ can vary during the lifetime of an application. The radiation induced errors in buffers typically result in single bit errors. Further, using interleaving techniques can reduce multi bit errors to single bit errors (58). Thus, Single Error Correcting code (SEC) is sufficient to give a full coverage for the errors. Assuming a SEU model for errors and a SEC code, we get $c = 1$. Substituting in Equation 2.4 we get,

$$\rho = 1 - k.$$  \hspace{1cm} (2.5)

We establish a relation between the duty period of the ECC (fraction ECC is turned on) and the error tolerance of the application. Thus, for higher $k$, the fraction of time we need to keep the ECC enabled is smaller.

### 2.3.2 Modeling Residual Error Rate with Architectural Vulnerability Factor

In the previous section, we quantified the undetected error rate without taking the architectural impacts into consideration. Traditionally, the effective error rate due to soft errors, has been related to the raw error rate and the Architectural Vulnerability
Factor (AVF) by the Equation 2.6.

$$\lambda_{eff} = AVF \cdot \lambda_{raw} \quad (2.6)$$

The residual error rate is given by equation 2.7 (similar argument as made in Equation 2.2).

$$\lambda_{res} = (1 - c \cdot \rho) \lambda_{eff} \quad (2.7)$$

Thus, from equations 2.6, 2.7 we get,

$$\lambda_{res} = (1 - c \cdot \rho) \cdot AVF \cdot \lambda_{raw} \quad (2.8)$$

where c is the coverage of the ECC.

From the definition of $k$, equation 2.8 can be written as

$$\rho = \frac{1}{c} \cdot (1 - \frac{k}{AVF}) \quad (2.9)$$

For errors modeled as a SEU and with an SEC error correction, we get

$$\rho = (1 - \frac{k}{AVF}) \quad (2.10)$$

2.3.3 Power Equations

The basic power consumed by the reliability component of the network is given by

$$Total Power_{basic} = Encoding Power + Decoding Power$$

$$Encoding Power = \frac{N_h \cdot N_{hops} \cdot Power_{encoder}}{num\_cycles}$$
Decoding Power = \frac{N_f \cdot N_{hops} \cdot Power_{decoder}}{num\_cycles} \tag{2.11}

where \( N_f \) is the number of flits injected into the system, \( N_{hops} \) is the average number of hops by a flit in the system, \( Power_{encoder} \) and \( Power_{decoder} \) are the power of the encoder and the decoder respectively.

The power consumed by the turning on and off of ECC is given by

\[
Total\ Power = \rho \cdot (\text{Encoding Power} + \text{Decoding Power})
\tag{2.12}
\]

Thus, the percentage of power saved is

\[
Power\ saved = (1 - \rho)\ % \tag{2.13}
\]

2.4 Methodology

The AVF of components is typically derated by several factors such as validity and dynamically dead instructions. An online AVF mechanism cannot take into account derating factors such as dynamically dead instructions as these are difficult to establish at run-time, especially in the network. Utilization is typically a good approximation in such cases. In our work, this translates to the buffer utilization being used as an approximation for the AVF of the buffers. Buffer utilization takes into account validity of entries and is already measured in network routers for optimizations like congestion control (30). Thus, we approximate the AVF of the router to be the buffer occupancy. Note that this value is a conservative estimate of the AVF. Thus, throttling the router input buffer occupancy is an effective way to bound the total AVF.

If the AVF of the input buffers is bounded by a threshold value (\( AVF_{thresh} \)), then by Equation 2.10, the \( \rho \) value is also bound (from \( \rho_{thresh} = 1 - k/AVF_{thresh} \)). Noting that if the \( AVF_{thresh} \) decreases then the \( \rho_{thresh} \) decreases, and from Equation 2.13, we conclude that the power savings can be further increased by decreasing the buffer utilization threshold of the routers. However, limiting the buffer utilization will impact
the overall throughput of the system negatively. In the forthcoming section, we present various strategies to smartly throttle the network to achieve power savings by trading off some network throughput while maintaining the desired $\lambda_{res}$ and thus providing reliability guarantees.

2.4.1 Static Single AVF (SAVF) Throttling

For a given $AVF_{thresh}$, we set the $AVF_{thresh}$ of every router to the same bound value. This is a simple approach in which each router gets throttled beyond the same maximum buffer utilization. Throttling involves stopping the flits from entering a router. These flits arrive either from a local PE in the case of a new injection or from an adjacent router. Also, every router performs ECC with the same duty period of $\rho_{thresh}$. The demand for buffers is not uniform across the routers due to deterministic routing protocol. This motivates our next approach of utilizing different threshold values for different routers as explained below.

2.4.2 Multiple AVF Domains (MAVF)

Buffer utilization in the network is non-uniform. It is usually determined by the traffic pattern, routing algorithm and the topology of the network. For most cases the
buffer utilization of the central routers is higher than the edge routers. Figure 2.2 shows the average buffer utilization across a 4X4 network for Uniform Random traffic. The figure shows that the buffer utilization is higher in the central routers. Consequently, using a single global AVF bound can adversely affect the throughput. Thus, we propose multiple AVF (MAVF) domains wherein the inner routers have a higher AVF bound than the outer routers. These bounds can be either set statically (Static MAVF) or changed dynamically (Dynamic MAVF).

**Static MAVF**

In the static approach, the inner routers are given higher threshold than the outer routers. The key point to note is that the overall AVF of the network is still maintained at $AVF_{threshold}$. Thus, the routers are still protected at the same duty cycle ‘$\rho$’ as the corresponding ‘Static SAVF’ scheme with the same threshold AVF. The goal of this approach is to maintain the same power savings while improving the overall performance of the network.

**Ideal Static Throttling**

In order to understand the maximum benefits that can be obtained by the static MAVF scheme, we implement an ‘Ideal Static’ bounding approach. New packets are throttled at each node if the buffer utilization of the entire network crosses the threshold. Thus, there is no strict bound for the buffer utilization of any one router. The advantage that ‘Ideal Static’ approach provides is two-fold. Firstly, lack of strict bound on individual routers enables the valid buffers to be distributed based on the required utilization. Secondly, the inter-router movement of flits is never hindered.

**Dynamic MAVF**

In the dynamic approach, every router is allowed to change its local $AVF$ and $\rho$ values dynamically. To enable this, the target residual errors of the network are divided equally among the routers (e.g. in a 16 node system, each router is targeted for $\lambda_{res}/16$ of the errors). With this target, every router can independently change it’s AVF bound
(no global communication) and $\rho$ value to meet the target reliability. Initially, all routers have the same AVF bound. During run time, every router tracks the total number of flits that have been stalled due to its AVF bound for a window of time. If the number of such flits is higher than a threshold value, the router increases its AVF and the corresponding $\rho$ value to maintain the same overall $k$. As a result of changing the $\rho$, the obtained power savings decrease from the ‘Static SAVF’ and the ‘Static MAVF’ case. The main advantage of this scheme over the static schemes is that the throughput loss is lower.

### 2.5 Bounding AVF via throttling

Figure 2.3 shows our implementation for bounding the buffer utilization. Each router monitors its buffer utilization in every cycle and if its utilization exceeds the set AVF threshold (NR in the figure), it signals to all its neighboring routers to stop transmitting traffic to it. On receiving this throttle signal in the following cycle, each of the neighboring routers (CR in the figure) flag the flits in the SA stage that are contending for the switch port leading to the susceptible router(s) (NR). All the flagged requests are blocked for this cycle by disabling the appropriate arbiters in the SA1 stage. These requests can contend for the switch port in the next cycle subject to the new buffer utilization in that cycle.

We discuss the implementation details for the Multiple AVF domain scheme. In the case of Static MAVF, no extra hardware support is needed. Each router stores an independent threshold value for comparison. Dynamic MAVF requires the ability to keep track of the number of flits rejected due to the AVF bounding. We use a 16-bit counter in every router for doing this. The power overhead of the counter is taken into account for all our results. Whenever a throttle signal is sent to the neighbors, a actualThrottle signal is received back in the next cycle which indicates if there was really a packet that was ready to be transmitted but blocked due to the throttle signal. We count the number of occurrences of the actualThrottle signal and if this count exceeds a given rejectThreshold then the AVF and $\rho$ are adjusted dynamically to bring down this count. The counter is reset after a window size of cycles. This scenario is also shown in Figure 2.3.
2.6 Experimental Setup

In the base case for all our experiments, ECC is provided 100% of the time. Further, in the base case, there is no throttling of buffers. We use an in-house network-on-chip cycle accurate simulator and run our simulations for 150,000 packets. The NoC router implemented is a state-of-art low latency pipeline which adopts deterministic XY routing, input buffering, virtual channel flow control and wormhole switching. We assume that our router and the ECC components run at 1GHz. The encoder and decoder were synthesized using RTL and TSMC Synopsys Design Compiler at 90nm. The dynamic and leakage power obtained for the 32 bit encoder are 0.4mW and 55nW respectively and that of the 32 bit decoder are 4mW and 493nW respectively. The power of the router components were obtained from Orion (83). Note that we model the power consumed by the extra bits for the encoded data in the router components. Also, the number of extra bits grows very slowly for increasing flit size. For the synthetic traffic results, a 4x4 network was used. For the application results, we use the PARSEC benchmark suite (10). The simulation setup used Virtutech Simics (53) running a 64 core CMP with Linux kernel 2.6.15. The network is an 8 X 8 fabric with each node consisting
2.7 Static SAVF Throttling results

In this section, we evaluate the power and throughput trade-offs of our ‘Static SAVF’ bounding scheme for different parameters. Note that in all these experiments, a single $AVF_{threshold}$ value is statically set for every router.

2.7.1 Power Savings

The percentage of power saved is dependent on the value of ‘$k$’ and ‘$AVF$’ (Equation 2.10, 2.13). Consequently, the percentage of power saved is independent of network size, traffic pattern and load rate. Further, as the header packet is protected at all times, the length of the packet is a factor in determining the percentage power saved. The experiments were conducted on a 4 X 4 network and injected packets followed a Uniform Random (UR) pattern at a load rate of 0.10 packets/node/cycle.
Effect of different AVF thresholds

We experiment with different threshold values (each experiment has a single AVF value for all routers) and the percentage of power savings (over the base case of no throttling) are summarized in Figure 2.4(a). Point B in the figure, represents the case when $AVF_{\text{threshold}}$ is equal to the $k$ value. In this scenario, maximum power savings are obtained as all errors can be tolerated by the system and the ECC apparatus is completely disabled for the body flits. Consequently, decreasing the $AVF_{\text{threshold}}$ will only result in lower throughput with no gain in power. At Point A there is no throttling ($AVF_{\text{threshold}}$ is one). Hence, at Point A all the power savings obtained are due to application’s error tolerance. As shown in the Figure 2.4(a), power savings can be increased from 18% when there is no AVF bounding to 72% by setting a threshold on buffer utilization.

Effect of different $k$ values

We sweep the value of error tolerance ($k$) to cover all possible scenarios. For this experiment, the $AVF_{\text{threshold}}$ is kept constant at 0.35. Figure 2.4(b) shows that the percentage power savings grow linearly until the AVF value becomes equal to the $k$ value (at Point C). Beyond Point C, the power savings peak off. Note that the maximum power savings are less than 100%. The remaining power is always consumed for the header flits which are protected even when the $\rho$ value is zero.
Effect of the message length

Figure 2.4(c) shows the power savings for differing values of the message length. Experiments were performed for message lengths from two to eight flits. Each of these has only one flit (the header flit) that is protected even when the $\rho$ value is zero. Thus, the power savings and the peak value of the savings increase with message length.

2.7.2 Throughput Loss

The power savings that are obtained due to throttling come at the expense of throughput loss. In this section, we quantify the throughput loss for various traffic patterns and load rates.

Traffic Patterns

Figure 2.5(a) shows the effect of traffic pattern on the throughput loss. We experiment with Uniform Random (UR), Bit Compliment (BC), Tornado (TN) and Transpose (TP) traffic patterns (24). BC has the highest loss of throughput and TN has the least loss. The load rate is fixed at 0.10 packets/node/cycle for this experiment.

Load Rate

Figure 2.5(b) shows the effect of increasing load rate and $AVF_{threshold}$ on throughput loss for UR traffic pattern. As seen from the figure, the loss of throughput is extremely large for low values of threshold under high load rates. This implies that when a highly reliable system is desired at large load rates, a static SAVF scheme is insufficient due to the loss of throughput. This motivates our Multiple AVF domains techniques (Section 2.8).

Link Throughput

Figure 2.5(c) shows that the link throughput for the base case and for two different AVF bounds for UR traffic. From the figure, we observe that the loss in link throughput increases with the load rate. In addition, lowering the AVF bound leads to decrease in link throughput.
In summary, the static SAVF method incurs a large throughput loss under tight AVF bounds and high load rates. Multiple AVF (MAVF) solutions were proposed in Section 2.4 to decrease the loss in throughput while still obtaining the power savings. The results are presented in the following section.

2.8 Multiple AVF Domains

In this section, we present the experimental results for the Multiple AVF Domain schemes as described in Section 2.4. We experiment with a 4X4 network and an AVF bound of 0.15 for varying load rates. For the SAVF case, the threshold of all the routers was set to 0.15.

2.8.1 Static MAVF

In the Static MAVF experiment, we set the inner four routers to a threshold of 0.30 and the outer twelve routers to a threshold of 0.10. This leads to an overall vulnerability of 0.15 (same as the Static SAVF case). There are several combinations for the AVF bounds in inner and outer routers that achieve an overall AVF bound of 0.15. Among all of them, the combination chosen above is the most beneficial.

Figure 2.6(a) shows the result of the experiment. Under ‘Ideal Static’ throttling, the impact on the throughput is lower especially for high load rates. The ‘Static MAVF’ always performs better than the Static SAVF and approaches the performance of Ideal Static for low injection rates.
2.8.2 Dynamic MAVF

In the Dynamic MAVF scenario, all the routers start with an AVF threshold of 0.15. Each router counts the number of flits that it has rejected as a result of AVF bounding. The AVF bound is increased in small steps of 0.05 whenever the number of rejected flits crosses the reject threshold in a given time window (Section 2.5). The upper bound of the AVF is set as 0.50. Figure 2.6(a) also shows the loss in throughput for the Dynamic MAVF case.

Dynamically adapting AVF throttling performs better than the Ideal Static experiments. The reason being in Dynamic MAVF, the overall AVF is not bound to a constant value as in all the static cases. Increasing the overall system AVF bound, when throughput losses are high, is the chief reason for Dynamic MAVF to outperform all the static solutions. However, the increase in overall AVF, reduces the power saved by Dynamic MAVF. The results for the normalized power savings compared to the static case are shown in Figure 2.6(b). Note that the power saved in all static cases is the same. Results have been normalized to the static case in order to make them independent of the target residual error rate. As observed from the results, Dynamic MAVF is not a good solution when seeking aggressive power savings.

Dynamic MAVF changes the overall AVF of the router buffers. Figure 2.6(c) shows the fraction of total number of valid buffers (the AVF of the router buffers) during the simulation of Dynamic MAVF for UR traffic at a load rate of 0.35. The figure shows a steady increase in the system AVF from 0.15 and it levels off at a value of 0.31.

2.9 Application Results

In this section, we present the results of our experiments for the PARSEC benchmark suite. For the sake of brevity, we select the kernels canneal, dedup, streamcluster and the applications fluid and x264. Other benchmarks from the suite followed the same trend. The network throughput loss are shown in Figure 2.7(a). The experiments for the ‘Static SAVF’ and the ‘Ideal Static’ are performed with an AVF bound of 0.15. ‘Static MAVF’ has an AVF bound of 0.30 for the inner most 16 routers and 0.10 for the other 48
router. This is equivalent to a system with an overall AVF bound of 0.15. In ‘Dynamic MAVF’, the AVF bound was initialized to a value of 0.15 for every router. During the execution, each router adjusts the AVF bound and $\rho$ value independently.

The maximum throughput loss is observed for the dedup application and is 9.8% and 2.3% for Static SAVF and the Dynamic MAVF respectively. The normalized power savings for the Dynamic MAVF is shown in Figure 2.7(b). For each application, the power saved is normalized with the power saved in the corresponding static scheme. The power saved is greater than 92% of the maximum possible (static case) for all the benchmarks.

When the $k$ value is 0.1, the actual power saved in the Dynamic MAVF scheme is 44%. Thus, for the load rates seen for these applications, Dynamic MAVF tunes itself to provide high power savings with minimal loss in network throughput. On an average, for all benchmarks, the power saved reaches up to 92% of the maximum possible savings while incurring less than 2.3% of throughput loss.

\subsection*{2.10 Related Work}

Bertozzi et al. (8) discussed error protection in buses with a residual error rate. An adaptive error protection scheme for energy efficient buses was presented in Li (51). Routers in NoC being distributed resources provide a different challenge. Jantsch et
al. (82) gave the power analysis of link level protection in network on chip and conclude that error correction schemes have a lower power consumption than retransmission schemes. Despite their energy efficiency, the total energy consumption for encoding and decoding every flit at each hop can be as high as 20% of the power consumed by it for traversing the router (61) and our power models confirmed this. Thus, we explore power saving techniques in this work. Architectural Vulnerability Factor (AVF) was introduced by Mukherjee et al. (59).

Fick et al. (29) proposed a router micro-architecture that deals with hard faults in the network such as link failures. Koibuchi et al. (46) also proposed a lightweight router micro-architecture to deal with hard failures by providing backup paths in the network. Peh et al. (50) describe the implementation of an architecture-level early-stage design space exploration tool that incorporates the effect of process and temperature variation for Network-on-chips (NoC). None of these works considered power-performance trade-offs in providing reliability. In this work, we model reliability in the presence of transient errors. Park (68) addressed the impact of different error-correction schemes in Network-on-chip in the presence of soft errors. They provide protection at all times. In this work, we propose to save power by exploiting the inherent error tolerance of applications while still providing constant reliability. Murali et. al. (61) discussed the impact of different error correction schemes on power. Their results were chosen as our basic error protection scheme.

2.11 Conclusion

In this chapter, we present an analytical model for studying the trade-offs between the performance and power while providing reliability guarantees for on-chip networks. Two key observations that led to our proposed power saving techniques are: (a) Some applications have inherent fault tolerance and thus the ECC schemes can be relaxed by enabling them for a fraction of the time and (b) The buffer utilization is a good approximation for the AVF of router buffers. We bound the AVF by throttling the traffic into the router to save power while guaranteeing reliability at a marginal throughput loss. In order to reduce the penalty incurred in terms of throughput, we use multiple AVF
domains. The proposed dynamic multiple AVF domains technique saves up to 44% of the reliability power at minimal cost of less than 3% loss of network throughput for applications.
Chapter 3

Reliability Aware Performance and Power Optimization in DVFS-based On-chip Networks

3.1 Introduction

Advancements in semiconductor technology have lead to diminutive feature sizes for a transistor. This has lead to a dramatic increase in the overall number of transistors available on a modern chip. To take advantage of the ever increasing transistor budget, there has been a paradigm shift towards having multiple processors on a chip (62). Chip multiprocessors (CMPs) have found a niche in both the embedded markets and as well as the mainstream laptop and desktop computers. Network on Chip (NoC) has been suggested as a solution to the exacerbating global wire delay problem in newer technology generations and for a scalable number of cores. Currently, commercially designed NoC topologies include ring (Intel’s Larrabee), mesh (Tilera) and clustered networks. The on-chip network plays a vital role in the performance, and power consumption of the system.

Reliability has been identified as one of the key limiters to future transistor scaling. Process variation, wear-out and transient errors are the major contributors to faults in chips. Process variations are posing a big challenge for the semi-conductor industry in terms of the yield. In addition, dynamic variations due to non-uniformity of workload activity across the chip can also cause erroneous scenarios. The on-chip network which is the basic medium of communication among the components is a distributed resource and thus will play a major role in determining the overall system reliability. The impact of process and temperature variations on the on-chip interconnect were studied in (50; 65).

Recent work done by Kim et.al (44) demonstrate the use of on-chip voltage regulators for a per-core Dynamic Voltage Frequency Scaling (DVFS) in a CMP. With technologies such as this, DVFS can become an integral part of the on-chip network as
well. DVFS can also be applied to routers for saving power as well as for controlling congestion i.e. increasing the throughput of the system. Under congestion, techniques such as throttling the upstream routers and/or increasing the frequency of the congested router can help controlling the congestion. For using such a variable frequency system, micro-architectural changes such as using a dual clock I/O buffer for the input buffers would be required.

![Router pipeline](image)

Fig. 3.1 Router pipeline

In this chapter, we deal with transient errors induced due to voltage fluctuations and temperature variations for an on-chip interconnection network. In addition, power and performance optimizations such as per-router DVFS that are reliability-agnostic will result in significant variation in the error rate. The transmitted data in the NoC can be protected using error correcting codes (ECC) such as Single Error Correcting (SEC), double error correcting (DEC) codes and Triple error correction (TEC). The ECC schemes differ in their protection capabilities, power consumption and the speeds at which they operate. Better coverage schemes have higher power consumption and have a higher latency penalty and thus these must be used wisely. Prior work does similar analysis in the context of buses which is a centralized communication media (51). On-chip interconnect poses different challenges as it is a distributed system. In addition, this work distinguishes itself as the first to identify the variation of error rate due to non-uniformity in voltage profile that result from using tweaks such as DVFS. In this work, we provide a comprehensive evaluation of the overheads involved in supporting dynamically reconfigurable hybrid ECC in the presence of per-router DVFS.
In the following sections, we motivate the necessity to accommodate differing levels of error correction capabilities in the NoC to tackle the voltage and temperature variations. In order to facilitate tunable levels of reliability, we need to accommodate for multiple types of encoder/decoder combinations that increases the area overhead of ECC from 1.8% to 3.2% of the total network area. We monitor the instantaneous vulnerability to transient errors at the granularity of an individual router and choose the ECC scheme capable of meeting the reliability target. With a small area penalty, we obtain upto 55% of savings in the power expended for error protection for synthetic traffic and up to 8% savings for applications. Further, we also decrease the latency penalty of reliability mechanisms. These latency improvements along with the latency benefits obtained due to per-router DVFS, result in upto 35% in average flit latency.

3.2 Network Architecture

3.2.1 Base Router Architecture

In this section, we outline the details of the underlying network architecture. Every network node is a processing core along with its private L1 and shared L2 bank. Every node has a router which handles the communication for it and the routers are connected through inter-router links. A simple mesh is used as the network topology. Thus, the router has five input/output ports, four of them are used to communicate with its neighbors in the four cardinal directions and one port for the processing element (PE). One or more nodes in the network have an extra injection/ejection port attached to the memory controller.

A flit is the smallest fraction of data that can be transmitted through the network. Every packet consists of a header flit and/or multiple body and tail flits. The base router is a pipelined structure with 5 stages namely Routing Computation (RC), Virtual channel Allocation (VA), Switch Allocation (SA), Switch Traversal (ST) and Link Traversal (LT) as shown in Figure 3.1. Only the header flits go through the RC and VA stages. SA, ST and LT stages are common to all flit types. The main router components are the RC unit, input buffer, arbiters for VA and SA stage and the crossbar switch as shown in
the Figure 1.3. The RC unit computes the output port based on the packet destination. The input buffer is composed of multiple virtual channels. A virtual channel is allocated to a packet at a time. The VA unit arbitrates for a virtual channel in the input buffer of the next router. The winners of VA compete for the switch in the SA stage. Finally, the winners of the SA stage traverse the crossbar switch and the corresponding link to reach the next router.

3.2.2 Support for Error Correction

Protection from data errors in a NoC has been studied previously in literature. Srinivasan (61) et al. and Park et al. (68) investigate the various options and compare end-to-end correction or hop-by-hop correction. Error correction can be done at the message level or the flit level. The above mentioned works conclude that flit level hop-by-hop correction is the best option in terms of performance and power overheads. Thus, we use hop-by-hop error correction as the basis for our error correction scheme.

Consequently, for providing error correction every router is augmented with an encoder and a decoder. Every flit is encoded at its source and decoded at every hop to make sure the data remains correct. As part of the inter-router link traversal, the flit is written into the next router’s buffer. The decoding of the flit is done in parallel with the RC stage (if it is a header flit) and checked for errors and corrected if any. If it is not a header flit then the decoding is done in parallel with the SA stage. If the router is running at its base frequency then our synthesis analysis shows that the decoding delay is less than a cycle. More details are discussed in section 3.4.2. The extra state information calculated during RC and VA stages etc., for the header flit are protected by triple modular redundancy as this is a very small amount of data that is store locally in the router. The architecture for the reconfigurable ECC is discussed in Section 3.5.

3.3 Implementation of DVFS on NoCs

We build a CMP framework in which every router performs DVFS independently in order to gain network performance. We increase the frequency of congested routers which reduces the amount of time spent by packets in the network. In order to achieve
Table 3.1 On chip regulator power consumption

<table>
<thead>
<tr>
<th>Output Voltage(V)</th>
<th>Power(mW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.75</td>
<td>34.1</td>
</tr>
<tr>
<td>0.9</td>
<td>41.5</td>
</tr>
<tr>
<td>1.0</td>
<td>52.3</td>
</tr>
</tbody>
</table>

per-router DVFS, each router should be supplied with different voltages and frequencies from which one can be selected. Instead of distributing multiple supply voltages across the network, we can use an on-chip voltage regulator for every router. We use the 2-step voltage regulator configuration as proposed by Kim’s work. An off-chip regulator performs the initial step down from 3.7V to 1.8V followed by multiple on-chip voltage regulators where each of them steps it down from 1.8V to 1.2V. This approach amortizes the degradation in conversion efficiency of employing only off chip regulators. A multi-phase buck converter that can provide three voltage and a programmable voltage controlled ring oscillator provides the three frequency levels (38). The on-chip regulators operate at 125MHz switching frequency and provide voltage transitions from 1V to 0.75V. Though we can support more levels of voltage and frequency, we use three levels one for each ECC scheme as it suffices to demonstrate our goal and the benefits of reconfigurable ECC schemes. The voltage settling time for every 100mV change is 13ns. The power consumption (at activity factor of 0.5) of our regulator with conversion efficiencies similar to that in (44) is given in Table 3.3. The area overhead in a 6x6 mesh for 36 on-chip regulators is 2.25mm² which is around 25% of the area of all routers in the network. This is based on our router area of 0.245mm² and power of 0.2W at 65nm technology and 1.3GHz based on our synthesized design.

In our DVFS scheme, we use buffer utilization as the indicator for changing the voltage. The thresholds for transition were chosen to accommodate for the voltage settling time. We perform our DVFS at a coarse window of time as we expect that the buffer utilization takes some time to stabilize and to prevent unnecessary overhead due to transitions. A time window of 500 cycles is chosen for this work.
3.4 Reliability Considerations under Dynamic Variations of Voltage

In order to evaluate our reconfigurable scheme, we consider use of three error correction schemes of different strengths based on the BCH codes.

• Single error correction and double error detection (SECDED) is the most popular error coding scheme for single errors. For a code length of 127 bits, (127, 120) Hamming code is used.

• Double Error Correction detects and corrects up to two errors. If there are more than two errors then it can erroneously indicate two or less errors. We use (7,2) BCH code (n = 127 and data = 113) for this.

• Triple Error Correction (TEC) detects and corrects up to three errors. BCH code of (7,3) is used which produces a code-length of 127 bits for 106 data bits. It cannot detect more than three errors.

In the following subsections, we establish the motivation for our reconfigurable ECC schemes. We present three main reasons in support of this additional flexibility. The first reason is that the error rates increase exponentially with decrease in voltage. Consequently, at higher voltages and higher speeds, lower protection schemes can be adopted. The second reason is that the amount of power expended to protect the network at all times with the highest protection capability scheme is significant and thus wasteful if we can do better than that. Finally, in the presence of DVFS when the routers are being operated at a higher frequency, the more powerful ECC scheme will require multiple cycles to execute and consequently may lead to performance degradation.

3.4.1 Error Rate Variation

Though DVFS is attractive for optimizing performance per watt, it results in a non-uniform profile of voltage and frequency on the chip. It could also exacerbate the non-uniform temperature profiles. These variations in voltage lead to a change in the reliability of the router.
To understand this further, we model the errors due to noise in voltage that was presented in (32). In addition to the noise in supply voltage, we also vary the supply voltage. Let \( \epsilon \) be the probability that an error occurs either when a transfer occurs across a wire or a data bit is stored in the buffer. Let \( V_{sw} \) be the supply voltage. Let the noise voltage \( (V_N) \) have a variance of \( \sigma_N^2 \). Then we get,

\[
\epsilon = Q\left(\frac{V_{sw}}{2\sigma_N}\right)
\]

where \( Q(x) \) is the Gaussian noise pulse

\[
Q(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-\frac{y^2}{2}} dy
\]

From Equation 3.1, we conclude that decreasing the voltage of a router in the network will increase the probability of an error in the data passing through the router.

\[ \text{(a) } \sigma = 0.1 \]
\[ \text{(b) } \sigma = 0.15 \]

Fig. 3.2 Residual word error rate at varying supply voltages and error protection

It is to be noted that not all bit errors result in program errors. Every application has an inherent tolerance to errors. Thus, it is sufficient to keep the error rate below the tolerance limit of the application. Figure 3.2(a) and 3.2(b) show that as the supply voltage is increased, in order to maintain a constant residual error rate, a less powerful
Table 3.2 ECC scheme power, delay and area

<table>
<thead>
<tr>
<th></th>
<th>SEC @ 1V</th>
<th>DEC @ 0.9V</th>
<th>TEC @ 0.75V</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encoder</td>
<td>Decoder</td>
<td>Encoder</td>
<td>Decoder</td>
</tr>
<tr>
<td>Power (mW)</td>
<td>0.25</td>
<td>2.76</td>
<td>1.517</td>
</tr>
<tr>
<td></td>
<td>3.65</td>
<td>2.036</td>
<td>5.83</td>
</tr>
<tr>
<td>Delay (ns)</td>
<td>0.25</td>
<td>0.27</td>
<td>0.25</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>0.26</td>
<td>0.65</td>
</tr>
<tr>
<td>Area (µm²)</td>
<td>164.1</td>
<td>1158</td>
<td>271.7</td>
</tr>
<tr>
<td></td>
<td>1813.6</td>
<td>343.1</td>
<td>4204.9</td>
</tr>
<tr>
<td>Normalized Power</td>
<td>1</td>
<td>10.84</td>
<td>5.96</td>
</tr>
<tr>
<td></td>
<td>14.36</td>
<td>8</td>
<td>22.93</td>
</tr>
</tbody>
</table>

ECC scheme suffices. We find that for a supply noise of 0.2 or more the residual word error rate is extremely high (of the order of 0.9) and no protection scheme will help. Thus, circuit designers will need to do design tricks to keep the noise at tangible levels. We assume a noise margin of 0.15 for this work and we believe that it is in the ballpark for state-of-art circuits. The tolerance limit is a characteristic of the application and varies for different applications. We statically chose the target residual error rate to be at 0.0085 for this work. This can be a user input depending on how much reliability is required. Tightening this tolerance limit reduces the opportunities for optimizations and relaxing this limit will give rise to more scope for improvements.

Figure 3.2(b) shows that between voltages V1 and V2 TEC is essential to keep the error rate below the target residual error rate. For voltages between V2 and V3, TEC is not required as it overprotects and DEC is sufficient to meet the target residual error rate. Beyond V3 voltage, SEC can keep the error rate at the desired level. Thus, we chose voltages (V1, V2, V3) for our current experiments.

3.4.2 Error Correction Power

Table 3.4.2 shows the latency, power and area of each type of encoder and decoder. These were obtained from RTL synthesis of BCH encoder and decoders using Synopsys design compiler at 90nm technology and scaled down to 65nm. The corresponding router frequencies at these voltages chosen were 1.3GHz, 1.56GHz, and 1.73GHz respectively. The power numbers shown in Table 3.4.2 are scaled to these frequencies and voltages.
chosen in the previous section. The last row shows the power of each component normalized to the SEC encoder. From the table, it is clear that the most powerful TEC scheme is also the most expensive.

In order to maintain a target residual error rate in the network with no adaptability to dynamic variations, we would need to use the most powerful protection scheme all the time. The total power expended to provide TEC protection at all times in every router in the network can be as high as 15% of the total network power and is larger for bigger network sizes. Figure 3.3(a) shows the percentage of power expended by ECC to the total network power for various injection rates. This motivates us to look into the ECC power as a potential opportunity to save power. In addition in the presence of DVFS, if a router’s frequency is boosted up then the TEC scheme takes multiple cycles to execute and can eat into the performance benefits obtained due to the higher frequency. For the highest frequency in the system (1.73 GHz), from Table 3.4.2 we can see that both DEC and TEC will require multiple cycles for decoding. Consequently, providing TEC protection at all times is not the best option.

In summary, we identify that the reliability at a given instant of time is varying across the NoC due to non-uniform voltages. We propose to have a more power and
performance optimal approach by having differing levels of correction capabilities and switching between them dynamically along with the voltage domains.

3.5 Experimental Setup

We used an in-house cycle-accurate on-chip network simulator that models the 5-cycle ECC router shown in Figure 2.1. We used wormhole switching, virtual channel flow control and deterministic routing algorithm. We experiment with several synthetic traffic patterns namely uniform random (UR), transpose (TP), bit complement (BC) and tornado (TN) (24). We experiment with varying number of node configurations: 4X4(16), 6X6(36), 8X8(64) and 10X10(100).

We also present applications results of benchmarks chosen from SPLASH2 and SPECOMP suites. A trace-based simulation with Simics (53) as front-end were collected for a 16-node CMP as shown in the figure 5.4(b). The full system simulation therein was running on Solaris 9 operating system. Simics was made aware of the NoC layout with a 5 cycle router hop. Traces from this were fed into our NoC simulator to observe the characteristics of applications.

We assume the numbers of VCs as two per PC with a buffer depth of four. The injected packet size is 512 data bits. The flit size (the channel width) is 129 bits with 127 code bits (106 data and 21 maximum ECC bits) and 2 bits to indicate the type of ECC. The number of ECC bits used is 21 bits for TEC, 14 bits for DEC and 7 bits for SEC. Even though, more data can be sent with DEC and SEC, we cannot dynamically repartition the data in the flits. Wormhole switching prevents us from waiting to form the entire packet in a router. Thus, we pad the rest of the data bits in DEC and SEC. At the destination node, when we eject the flit to the PE, depending on the ECC type indicated by the 2 bits in each flit, we drop the extra pad and restore the original data.

Reconfigurable ECC Support

In the case of the reconfigurable scheme, when a new flit arriving and the current router have different levels of ECC, then in addition to decoding and making sure the data received is correct, we also need to encode this data with the current router ECC
scheme. Thus, the critical path for ECC now includes decoder delay of the arrived flit and the encoder delay of the new scheme of the current router. As the encoder delay is very small, this does not incur additional cycle penalty. If the ECC schemes of current flit and arriving flit are the same then no encoding is required. Thus, we incur an extra encoder power during transitions. This extra cost during transitions is small compared to the benefits obtained by saving decoder power. The total area overhead of supporting the 3 schemes increases from 1.85% to 3.24% of the network area.

DVFS of routers was performed at voltage and frequency levels of 1.3GHz (0.75 V), 1.56GHz (0.90 V) and 1.73GHz (1.0 V). In the reconfigurable ECC scenario, TEC maps to 0.75V domain, DEC to 0.90V domain and SEC to 1V domain. In the non-reconfigurable case however, the TEC needs to operate at frequencies of 1.56GHz, and 1.73GHz for higher voltage domains and thus will always require two cycles to finish its operation.

3.6 Experimental Results

As explained earlier, the reconfigurable reliability fabric provides us with opportunities to obtain both performance and power benefits in the reliability component. The performance benefits are obtained because the operation of TEC and DEC requires multiple cycles at higher frequencies. In this section, we capture the performance benefits of switching to a lower level of protection which is faster while still satisfying the reliability specifications.

3.6.1 Performance Benefits

For a non-reconfigurable fabric, the highest level of ECC protection (TEC) should be enabled at all voltage domains. However, when the router is working at the high voltage (and thus high frequency) domain, the EC stage will require two cycles. Thus, whenever the router is operating at these voltages due to the DVFS technique, the router pipeline increases from 5 stages to 6 stages. We model this extra bubble in our simulator at the higher voltage domains. This effects the overall performance improvement attained
Fig. 3.4 Average latency (per flit) improvement with DVFS by the DVFS technique. Figure 3.4 shows the drop in performance for UR and TP traffic for a various network sizes at an injection rate of 0.20 flits/node/cycle.

From the figure, we also see a consistent drop in performance improvement for the non-reconfigurable scenario for both UR and TP traffic. Thus, having reconfigurability is critical to realize the full potential of performance benefits that can be achieved by DVFS technique in CMPs. The performance improvement decreases with increasing network size. The reconfigurable fabric provides about 91% improvement over the non-reconfigurable fabric for UR traffic at 16 nodes and about 44% for TP pattern.

3.6.2 Power Benefits

The power saved by switching to lower level of ECC varies with several factors including threshold levels, network size, load rate and traffic pattern. We explore each of these in the following sections. The base case in all these experiments is the network using TEC at all voltage levels. The power consumed by the TEC scheme at a given voltage is given by Equation 3.2. Thus, the total power consumed in the base case is the power consumed at each of the voltages as given by Equation 3.3.

\[
Power_{\text{base-case}}(V) = Power_{\text{TEC-encoder}}(V) \cdot N_{\text{enc}}(V) + Power_{\text{TEC-decoder}}(V) \cdot N_{\text{dec}}(V)
\]  

(3.2)
where $Power_{\text{base-case}}(V)$ is the power in the base case at a voltage of ‘V’, $Power_{\text{TEC-encoder}}(V)$ and $Power_{\text{TEC-decoder}}(V)$ are the encoder and decoder power of TEC at voltage ‘V’, $N_{\text{enc}}(V)$ and $N_{\text{dec}}(V)$ are the number of encodes and decodes that take place at voltage ‘V’.

\[
Power_{\text{base-case}} = Power_{\text{base-case}}(V_1) + Power_{\text{base-case}}(V_2) + Power_{\text{base-case}}(V_3) \quad (3.3)
\]

where $V_1$, $V_2$ and $V_3$ are the three voltage domains.

\[
Power_{\text{reconfig}} = Power_{\text{TEC-encoder}}(V_1) \times N_{\text{TEC-enc}}(V_1) + \\
\sum_{V = V1, V2, V3} Power_{\text{TEC-decoder}}(V) \times N_{\text{TEC-dec}}(V) + \\
Power_{\text{DEC-encoder}}(V_2) \times N_{\text{DEC-enc}}(V_2) + \\
\sum_{V = V1, V2, V3} Power_{\text{DEC-decoder}}(V) \times N_{\text{DEC-dec}}(V) + \\
Power_{\text{SEC-encoder}}(V_3) \times N_{\text{SEC-enc}}(V_3) + \\
\sum_{V = V1, V2, V3} Power_{\text{SEC-decoder}}(V) \times N_{\text{SEC-dec}}(V) \quad (3.4)
\]

where the variables are similar to Equation 3.2. Note that while encoding is done only in the corresponding voltage levels (e.g. TEC encoding is always done in $V_1$), decoding can be done at any voltage level (e.g. TEC decoding can occur in a router with voltage $V_3$). The variables used in these equations are collected from simulation to estimate the power consumed in each case.

3.6.2.1 Experiments with Threshold Levels

In our framework, the routers transition into higher voltage and frequency domains based on certain threshold buffer utilization levels. These pre-determined levels are critical for determining the network performance improvement. In this section, we estimate the power reduction caused in the reliability component due to the presence of
reconfigurable EC component. Figure 3.5 shows the power reduction for three different threshold domains

1. **T1** Thresh-050-075: Switch to 0.9V at buffer utilization of 50% and 1V at buffer utilization of 75%

2. **T2** Thresh-025-050: Switch to 0.9V at buffer utilization of 25% and 1V at buffer utilization of 50%

3. **T3** Thresh-025-030: Switch to 0.9V at buffer utilization of 25% and 1V at buffer utilization of 30%

These threshold levels are ordered in the increasing order of aggressive scaling. Thus, T3 is more aggressive scaling compared to T2, which is turn more aggressive than T1. Figure 3.5 shows that the power saved increases with increased aggression in thresholds. For each domain, the power saved also increases with an increase in load rate per node. For the T3 domain, the power saved increases from 2% at a load rate of 0.10 flits/node/cycle to about 55% at a load rate of 0.35 flits/node/cycle.

### 3.6.2.2 Varying Network Size

Figure 3.6 shows the percentage power saved in the reliability component as a function of the network size. Experiments were performed at a constant load rate of 0.20 flits/node/cycle. As shown earlier, the power consumed by the reliability component
increases with node size. Figure 3.6 shows that the best of the savings are also obtained with an increase in network size. Thus, we believe reconfigurability in the reliability component is critical for reliable CMP fabrics of the future.

3.6.2.3 Varying Traffic Patterns

Traffic patterns determine the congestion in the network and the buffer utilization across the NoC. Thus, the opportunity to perform DVFS varies with the injected traffic pattern. Figure 3.7 shows the percentage of power saved in the reliability component as a function of the injected traffic pattern. Transpose (TP) has the highest savings of 28% and Tornado (TN) has the least savings of 11%. A network size of 64 nodes with an injection rate of 0.2 flits/node/cycle was chosen for this experiment.

3.6.2.4 Varying Injection Rate

Injection rate is one of the key determining factors for buffer utilization in the NoC routers. Higher injection rates lead to larger buffer occupancy and thus will impact our DVFS technique. An increase in injection rate will cause the routers to transition to higher voltage levels and thus lead to improved performance of the congested network. This gives us an opportunity to decrease the ECC scheme to be used and with it the power consumed by the reliability component.
Figure 3.8 shows the effect load rate on the power savings. The experiment was performed for a network size of 64 nodes with UR traffic. The power savings steadily increase with the injection rate. At an injection rate of 0.35 flits/node/cycle, the percentage savings of the reliability component are as high as 56%.

3.7 Application Results

In this section, we experiment with benchmarks from the SPECOMP (6) suite and the SPLASH2 suite (84) for power saved in those applications due to reconfigurable ECC. The threshold values for buffer utilization were set at 0.25 and 0.30. Figure 3.9 shows the results of this experiment. The power savings are highest for ‘barnes’ and ‘radix’ from the SPLASH2 suite of about 8%. We anticipate these benefits to increase with the ability to simulate larger configurations of processor nodes (as observed from Figure 3.6) and emerging applications with larger memory footprints and higher injection ratio.

3.8 Related Work

Buses, a traditional solution to multi processor communication, are infeasible in the presence of large number of cores as they don’t scale well. Network-on-Chip has emerged as the de-facto solution to address this problem (23). Managing on chip data
effectively is critical for the performance of multiprocessors and has lead to a lot of recent research in this area.

Commercial designs such as MIT RAW (80) and ALPHA 21364 network (57) show similar trends of 36% power consumption. Thus, reducing power consumption of NoC is one of the major challenges facing network design. Power-aware design of the router micro-architecture such as segmented crossbar, write-through buffer have been investigated (83). Also, throttling a flit traversal in a router (75) has been shown as a means to manage peak power constraints in off-chip networks. Prior works have proposed DVFS for links (74; 39) to manage power in off-chip networks. Dynamic voltage and frequency scaling (DVFS) techniques are being used in state-of-art chip-multi processors (CMPs) for saving power. Several works have proposed DVFS in processors (17; 85; 73) for power management. With increasing number of cores on the chip, the network connecting the components on-chip is expected to consume as much as 40W to 60W (14; 1).

Managing the reliability of the NoC was studied by (61; 8) . They consider various methods by which on-chip reliability can be achieved and evaluate the tradeoffs. The tradeoff points include error correction/error detection and retransmission, end-to-end/hop-by-hop, message level/flit level protection. They conclude that hop-by-hop flit level error correction was best suited for reliability of the NoC. The reliability of data passing through the buses was investigated in (51). This work uses the same model of reliability based on supply voltage and variation in the voltage. They evaluate the
Fig. 3.9 Power saved for multi-core benchmarks

variation in supply voltage ($\sigma_V$) only. Our study maintains a constant supply voltage variation and studies the effect of varying supply voltage on residual word error rate. Thus, our work differs in both the medium of communication and the cause for the data corruption.

Our work investigates the power benefits that can be obtained from the reliability component under an iso-reliability scenario where the residual error rate changes due to changing voltage. The voltage is changed to tweak the performance of the system. In (7), the authors investigate giving iso-reliability by changing the voltage domain to appropriate levels whenever the residual error rate changes. Thus, they propose changing voltages as a solution to changing error rate. In our work however, voltage change is the cause for changing error rates. Our solution is to provide a reconfigurable fabric of ECC to provide the required reliability at minimum energy consumption.

3.9 Conclusion

In this chapter, we propose a dynamically reconfigurable reliability module which changes its coverage based on the current operating conditions. We model multiple errors and various multi-error correction codes. We provide an adaptive DVFS framework for a Network-on-chip that optimizes performance and power under a given reliability budget with a reduced power overhead. The savings in power expended in reliability can be as high as 55% for synthetic traffic and up to 8% for applications. In addition the adaptive
scheme also reduces latency overhead due to reliability. We incur an additional area overhead penalty of 3.2% (increases from 1.8%) to accommodate for multiple encoders and decoders. We conclude that an adaptive error correction schemes can be used to achieve balance in reliability, power and performance tradeoffs.
Chapter 4

In-Network Caching for Chip Multiprocessors

4.1 Introduction

Effective management of shared data is critical to the performance of emerging multi-core architectures. Consequently, the use of shared L2 cache architectures has emerged as a popular trend to facilitate data sharing. Maintaining cache coherence is a critical need in such shared structures and is commonly enforced using either snoop-based or directory based protocols. In the broadcast-based snoop protocol, a requesting node broadcasts a snoop request to all nodes to find whether they currently cache the data or not. In contrast, the directory-based protocol is more scalable with number of cores as they eliminate the broadcast with directed messages according to the directory information. The directory-based protocol manages which node caches data in a directory corresponding to the home node at the requested address. Whenever a data is requested by a processor, the shared directory entry is used to identify the state of requested data as well as the current location of the shared data. Consequently, the data request involves three steps: (1) Communication from the requesting node to the home node of the directory entry; (2) In case a shared copy exists, a message is sent from the home node to the location of remote node caching the data. In case, no copy exists, the next level cache or memory is accessed and data is returned to requesting node skipping step 3; (3) a communication packet forwarding the data from the remote node with shared data to the requesting node. However, this protocol of the shared directory suffers from the latency overhead of having to send the data request in the order of the requesting node, the home node and the caching node.

Due to the communication-centric nature of such shared data accesses, the underlying communication fabric and protocols play an important role in determining the data
access latency. There have been several efforts at reducing the overheads of directory-based protocols and on the design of low-latency communication fabrics. In this work, we focus on the design of an efficient packet-based on chip communication fabric to reduce the number of hops required in servicing a data request. Particularly, we focus on caching data along the routers of the multi-hop communication fabric to facilitate faster response to data accesses. Most closely related to our effort is the effort to migrate the entire data cache on to the network layer by Mirzahi et al. (56). In order to circumvent the resource challenges of migrating large amount of data into the network router in an on-chip network, Eisley et al. (28). proposed caching only the coherence protocol information rather than the actual data. Our work identifies that caching a small fraction of shared data within the router in addition to caching protocol information can provide significant performance gains.

The key contributions of this chapter are

- Analyzing the memory access patterns of a set of applications from SpecOMP (6) suite and identifying that a small number of shared addresses dominate the accesses to memory systems and reducing access latencies to these shared locations has a significant impact on overall memory latency.

- Exploration of three different in-network data caching strategies to reduce overall latency that indicates that memory access latency can be reduced by upto 38% and on average by 27% across the applications.

- Design of the router micro-architecture to support in-network data caching to demonstrate the feasibility of incorporating them in on-chip networks.

The rest of this chapter is organized as follows. The next section analyzes the memory access behavior of the SpecOMP (6) benchmark suite and provides an overview of In-network cache coherence. Section 4.3 describes the modifications to the protocol and the microarchitectural changes in the router architecture required to enable and implement In-network caching. Section 4.4 explores our proposed In-network caching policies. Section 4.5 details the experimental setup for this work. Section 4.6 provides quantitative
evaluation of the in-network caching schemes along with the area overhead of the design. Section 4.7 discusses the related work. We conclude this work in Section 4.8.

4.2 Motivation and Background

Access to shared data is one of the critical factors that impact the performance of parallel algorithms. To understand this better we profile the SpecOMP benchmark for statistics on the number of addresses and the number of accesses to them. We classify these statistics as private and shared. We define a private address as one which is accessed by only one processor and is thus private to a single processor. A shared address is one which is accessed by at least two processors. Figure 4.1 shows the distribution of private and shared addresses for the profiled applications. The results indicate that the number of distinct shared addresses accessed is a small fraction of all addresses accessed. In addition to this observation, we noticed that the number of accesses to the shared addresses accounted for a large fraction (71% on average) of all memory accesses (See Figure 4.2). These two observations indicate that reducing the access latency to a small fraction of shared address locations may have a significant impact of overall latency. This motivates our effort to cache the shared data in a small data store of on-chip routers.
4.2.1 In Network Cache Coherence

Our technique is built on top of the in-network cache coherence (28) work that reduces the hop count for data accesses by caching coherence information within the routers. Hence, we provide a brief overview of this protocol (readers are referred to (28) for details.) The key to the in-network cache coherence work is a virtual tree of links that connects all sharers of the data with the home node. The location of the home node is determined statically based on the address of the data as in the case of traditional directory-based protocols (6).

The node that caches the shared data first serves as the root of the virtual tree and links are maintained at each router in the path between the home node and the sharing nodes. These virtual trees enable new read requests to identify the location of
Write operations for the basic In-network Cache coherence (MSI) protocol. Write
operations for the In Network Cache coherence (MSI) protocol. (a) A read request sent to
the home node intersects an existing virtual tree causing teardown message propagation
(b) Acknowledge messages from the leaf node (c) Acknowledge message causing teardown
and the write request message in flight (d) The new virtual tree with the requester as the
root.

the shared data from the coherence information on the routers instead of having to first
traverse to the home node.

Read Operation

When a node (R1) issues a read request, the packet is transmitted towards the
home node (H) as shown in Figure 4.3(a). There two possible scenarios in handling this
request. In the first case, the data is not cached in any other node and no outgoing virtual
link exists from the home node. In this case, H loads the data from the secondary cache
or memory and responds to the requester with data. When the response packet traverses
from the home node towards the requester (R1), the virtual links in each intermediate
router are set to point towards the requesting node. Figure 4.3(b) illustrates the virtual
tree formed as a result of such an occurrence.

In the second case, the requested data is already cached by another node. In
Figure 4.3(c) when R2 is requesting data for an address, R1 has the latest copy of the
data and there is tree with the values. (A request from R2 in Figure 4.3(c)). There are two sub-cases based on whether the path between the requester and the home node (H) intersects with an already formed virtual tree. In the first sub-case, when the request does not intersect with a virtual tree, the request reaches the home node. This is followed by a traversal through the virtual links towards the root node. At each hop along the virtual tree, the local node checks for a cached copy of the shared data. If the line is present in any of the intermediate nodes on the path, the request is satisfied and a response packet to the requester is formed. If not, the response packet originates from the root node. The response packet modifies the virtual tree on its way to the requester. Figure 4.3(d) depicts the augmented virtual tree. The second sub-case is similar to the first sub-case, except that the request is routed towards the root of the virtual tree when the request intersects with the virtual tree link at the node I. Consequently, it reduces the traversal all the way to the home node.

**Write Operation**

When a node requests for the write operation, a packet is transmitted towards the home node (H) for the address . In case it intersects a tree for the same address in some router along the way, that tree needs to be torn down before the current request is granted. In Figure 4.4(a) the write request from R2 intersects a tree on its way to the H. Then, the current node issues teardown messages for that address and sends it in all directions of the tree. Once the teardown message reaches a leaf in the directed tree, it invalidates the line for the entry in it’s tree cache. Further, it sends an acknowledgment up the tree, as shown in Figure 4.4(b). As a result of the mechanism, all the acknowledgments head towards H as shown in Figure 4.4(c). Once H becomes the leaf of the tree (every path out of it is invalidated) it then grants the request for R2. This results in forming a tree to R2 as shown in 4.4(d).

### 4.3 In-Network Data Caching

This section describes the necessary augmentations to the in-network cache coherence protocol to support the proposed data caching technique (Figure 4.5). We also
show the micro-architectural modifications required to support the new functionalities along with a brief description of the implementation.

4.3.1 Protocol Modifications

Fig. 4.5 Read operations for the proposed In Network Caching (MSI) protocol. (a) A read request sent to the home node (b) A read reply message forms the virtual tree rooted at the requester; The cache line is stored at an intermediate router (c) A second read request intersects the virtual tree at node with cache line in the router data store (d) The second requester obtains the data from the data store and becomes part of the virtual tree

**Reads:** Changes occur in the behavior of both the read reply message and the read request message. The data from a read reply message passing through an intermediate node towards the requester, can potentially be cached in the router's data store depending on several factors, including availability of free lines and the caching policy. Any read request message will intersect the virtual tree for the cache line either at the home node or at an intermediate node. In the basic version of the protocol, along the path of a tree, each node is checked to see if it has a local copy of the data. In our scheme, along with a check at the node, a *simultaneous check is made in the data store of the router.* If no copy exists, the read request message is routed towards the root as usual. Once a copy of the data is found either in the intermediate node’s cache or the data store of its
router, a read reply message is generated. A hit in the data store can reduce the number of hops needed to put the reply in the network and the hops the reply needs to take to its destination.

**Writes:** When a write request intersects the virtual tree of the cache line for which the write has been requested, it causes teardown messages to propagate towards the leaf nodes in the tree. At each intermediate node, in addition to the basic actions described in Section 4.2.1, any copy of the cache line in the data store of the router is marked as invalid. The write latency to the caches is not directly changed by our protocol. Any change in the write latency will be from the new traffic in the network which can arise from the altered read-response traffic.

**Teardown:** As soon as a teardown message is seen in the router, a check is made to see if the teardown address is a match in the router’s cache. In case there is a hit, the address is invalidated immediately. The rest of the protocol remains the same as in the In-Network Cache Coherence approach.

### 4.3.2 Router Microarchitecture

The proposed router architecture (shown in Figure 4.6(a)) has ports to access its four neighbors and it’s local processing element (PE). The proposed router pipeline that supports the modified protocol is shown in the Figure 4.6(c). It consists of the Route Compute (RC) stage, a Virtual Channel Allocation (VA) stage, a Switch Allocation (SA) stage, a Switch Traversal and a Link Traversal stage. When a packet enters the pipeline, the RC unit uses X-Y routing to route the packet.

The traditional router pipeline was enhanced with a *Tree Cache (TC)* stage to enable In Network Cache Coherence (28). The Tree Cache stage maintains the virtual links that form the tree. These virtual links are used to redirect requests for data and hence help reduce the hops taken by the request packet before it reaches a copy of the data.

This work adds another unit called the *Router Local Cache* (RLC) to the pipeline to be used for data storing. The structure of the RLC is as shown in Figure 4.6 (b). The main components of the RLC are *Data Store, Invalidation lines to the VC* and
the Evict Buffer. This unit implements policies to store cache lines observed at the router into the data store during a read reply message. It also implements policies that define when the data in the data store is to be invalidated when it observes a write request or a teardown message. Finally, it also recognizes packets containing read requests for cache lines stored in it and responds appropriately.

4.3.2.1 Data Store

The Data Store is the main component of the RLC. It is a set of entries where each entry contains a cache line address and a copy of the latest value for that address along with a valid bit as shown in Figure 4.6(b). It is a five-ported structure handling upto five packets in a single cycle. The address of the cache line contained in each packet that passes through the RLC is checked for a match with the addresses contained in the Data Store. The Data Store consists of two main structures, the Hit Lines and the Dedicated VC which are described below.

Hit Lines: Every read reply packet contains a cache line that can be stored in the data store. However, to maintain coherence, the address of the line should either be previously present in the TC or be inserted in the current cycle in the TC before data is cached in the data store. In some scenarios, such as a conflict in the TC which requires special handling, the read reply packet waits in the VC until the one of the TC lines (and hence and entire tree) is invalidated. The data of such packets are not stored as they are not a part of the TC. The communication of whether the address of a packet is hit in the TC is accomplished by using ‘hit lines’. Hit lines are wires, one for each port of the RLC, that are used to check for the presence in the TC of the packet entering through the corresponding port.

Dedicated VC: A hit for a read request packet in the data store creates a response packet for the data from the Data Store’s entry for that address. If the original packet that experienced the hit is from the PE, then the reply is immediately ejected to the PE. Any contention for the PE port is handled in a manner similar to the Early ejection (43). For packets from the remaining ports, a VC is dedicated in the input buffers for the PE as shown in Figure 4.6(b). Dedicating a VC ensures that the new packet that is sent
Fig. 4.6 Proposed router microarchitecture. (a) The router consists of $P$ input ports consisting of $V$ virtual channels and $P$ output ports connected to a crossbar. Packets pass through the router pipeline between the virtual channels and the switch. (b) The proposed Router Local Cache stage that manages the data in the Data Store in the router (c) The router pipeline. The Router Local Cache stage occurs in parallel with the packet routing and TC stage.

out does not have to search for a free VC every time when there is a hit in the Router Cache. This does not adversely effect the packets from the PE as the injection rates for applications are quite low. However, there are two limitations to this method. The number of packets that can be sent out as a response every cycle are now limited by the number of dedicated VCs (one in our case). Note that, out of the four directional physical channel packets entering a router, there can be potentially four hits in the Data Store. Since there is only one VC into which responses go, three of the hits are not responded to from this data store. Notice that, while not responding from the Data Store for all hits that occur decreases the performance benefits that can be acquired from our technique, it does not cause an error in the request procedure. The request packets that are not responded from the data store, are forwarded as usual along the tree towards the root. They can be responded either by the RLC unit of a different router or ultimately from the root as in the base protocol. The second limitation to this method has to do with
the dedicated VC being occupied at the time of a hit in the Data Store. In that case, the
current RLC can’t service the request packet. This again effects the performance gains
achievable from our work. In our simulations, we found the occurrence of these cases to
low enough not to affect the actual performance gains. Our experimental setup models
these two limitations. A hit in the data store for addresses contained in teardown and
write request packets results in the data stored in the Data Store entry for that address
being invalidated. This is accomplished by resetting the valid bit of the entry.

4.3.2.2 Invalidation Lines

When a packet enters the router, it is placed in a VC. From the next cycle, this
packet arbitrates for the virtual channels of the next router. If this packet is a read
request that is responded to from the Data Store, it needs to be restricted from further
propagation. This is achieved by invalidating the VC. This invalidation is accomplished
using the invalidation lines shown in Figure 4.6(b). There is one invalidate line for each
VC of the router, which can set or reset the valid bit of each of these VCs.

4.3.2.3 Evict Buffer

The condition that needs to be met to maintain cache coherence is as follows “If
a line is valid in the RLC, it should be valid in the TC as well”. The evict buffer ensures
that this condition is held. If at a given instance, the Data Store contains an entry for an
cache line that is present in the TC, any state changes (invalidations) for the lines in the
TC need to be reflected in the entries of the Data Store as well. A change in state for a
line for a TC can occur in two ways 1) through an external packet or, 2) through internal
protocol transitions. In the case of teardown and write request messages, a conservative
approach is adopted and the cache line in the Data Store is immediately invalidated.
This is accomplished through the appropriate ‘valid bit’ entry. Internal state changes
include evictions of the TC line due to conflicts. A line that is evicted from the TC due
to a conflict must be evicted in the next cycle from the data store to maintain coherence.
In addition to this, to maintain coherency, we modify eviction policy. This is handled
in conjunction with a modified replacement (eviction policy) which ensures that when a
line is being evicted from the TC, another packet does not have a hit to the same line. However, in the corner case, all the ways of a TC cache can be a hit to the incoming packets. In such cases, the packet requesting the eviction, backs off for a cycle before trying to evict line again. To invalidate the evicted addresses in the following cycle, all the addresses which are evicted in the TC in the current cycle are collected into an Evict Buffer as shown in Figure 4.6(b). These addresses are invalidated from the RLC in the following cycle. This way we ensure that the correctness condition stated above is satisfied.

Finally, the set of packets that are to be processed in the data store in a given cycle include the potential five new packets and a potential 5 new evictions (all packets from the preceding cycle evict). The Data Store is a five-ported structure and can thus handle only five packets in a given cycle. Not handling any of the packets might lead to a coherence violation in the Data Store. We handle this situation of more packets to process than available ports by simply invalidating the entire Data Store. Our experiments revealed that the occurrence of these events is quite rare due to the low injection rates of applications and thus doesn’t affect the performance of our system.
4.3.3 Implementation

The RLC stage was designed in HDL and was synthesized at the 90nm technology. From this, we saw that the TC stage was still the frequency determining stage for all sizes of data store considered in this work. Thus, our work doesn’t reduce the frequency of the baseline router. The increase in area of the router is shown in Figures 4.7, 4.8. Figure 4.7 shows the increase in area for different sizes of data store compared to the baseline router. In this instance, the area overhead is less that 2% for our largest sized data store. The TreeCache (TC) stage replaces the directories in the cache subsystem. Thus, we also compare the impact on area for a baseline router without the TC stage. Figure 4.8 shows the percentage increase in area compared to the baseline router without a TC. Here the area overheads are higher and vary from 2.5% to 29%.

4.4 In Network Caching Policies

In this section, we detail the caching policies explored in this work. When a node is replying to a read request, it forms a ‘read reply’ message and sends it to the network. A read reply message always follows the tree back to the request node. In case it arrives at a node without a tree, the node is added to the tree structure before forwarding the packet. When a new read reply packet enters the router, a decision needs to be made on saving the data from the packet in the Data Store. We experiment with three methods of making this decision namely Ideal Caching, Independent Caching and Zonal caching.

4.4.1 Ideal Caching

This scheme demonstrates the best case performance that can be obtained from storing data in the routers. We model ideal caching by simulating an infinite sized data store. Thus, the decision made in every router is to store data from all the ‘read reply’ packets. Though impractical to be actually implemented, this experiment gives us the upper limit on the amount of savings possible through in-network caching. This also helps us understand the effectiveness of the more practical schemes that follow. In the ideal caching, every node in the tree cache for a shared data has the data as well. Thus,
Fig. 4.9 Zones in a mesh: The implementation of Zonal Caching using zonal bits. Zonal bits indicate whether a cache line is currently cached within a particular zone. The zonal bits of the message show that it is cached in Zones 1 and 2 and not cached in Zones 3 and 4. Hence, it may be cached at the highlighted node.

for a read request packet, intersecting a tree for the requested address is equivalent to finding the data.

4.4.2 Independent Caching

In our next caching policy, we look at an easy to implement caching scheme. In this scheme, the router has a fixed size for the local store. In independent caching, every router decides what it wants to keep in the additional storage independently. Thus, the decision is made by each router without any global knowledge. Each router looks at the data passing through it and decides whether to cache it or not. This policy is implemented in the following way. The router tries to cache all the data that passes through it. If the router’s data store is already full, we replace the least recently used line. However, we do not evict a line unless it’s last access has been before a threshold number
of cycles (50 cycles) to the current cycle. This is done to prevent evicts from the router’s data store frequently. The biggest drawback with this scheme is that it under-utilizes the local store capacity. For instance, if a data packet passes in a particular direction, then it has the potential of being cached in every router on the path. However, given the fact that the number of shared addresses is few, this caching scheme is still effective.

4.4.3 Zonal Caching

The drawback of the independent caching scheme is that it might cache multiple copies of the same data in the network. The best way to use the available capacity of the router’s cache is to ensure that there is just a single copy of this data in the network. However, trying to reduce the number of copies also conflicts with the goal of reducing the number of hops to service the request.

Zonal caching aims to compromise between these two conflicting goals. This scheme aims to increase the effective data store capacity over the Independent caching scheme while not restricting the number of copies to one. Multiple copies are allowed in the network as long as they are distant from each other. To realize such a scheme, some global state information is required on where the copies of the data are present in the data stores of the network. Carrying explicit information on the location of the data in the router cache space gives the most information to make such decisions. However, it is not scalable.

As a more scalable approach, we divide the mesh of routers into a grid and aim to place one copy in each of the zone in the grid (Figure 4.9). A four bit vector is added to the header of a packet to capture this information. Each bit corresponds to the presence of a packet in a zone. If (say) the bit for zone 1 is turned on, that means that one of the routers in the zone 1 has a copy of the data in its local store. In our router, when a read reply packet is cached in the data store, the packet header is modified to turn on the current zone bit in the RLC stage. In the Figure 4.9, when a read reply packet enters the highlighted node in zone 4, it carries the information that it has been cached in zone 1 and zone 2 but not in zone 3 and zone 4. Since, the node seeing this packet is in zone 4, it decides it can cache this data and tries to cache it. Note that this
logic could be implemented with a single bit but the scheme here illustrates the point better. In order for a router not to cache the data from it’s own node we propose an optimization. Packets injected from the current node are not candidates for being added into the router’s data store. The replacement scheme is as described in the independent caching. The zonal caching can lead to increase in content for the cache space along the edges of the router as compared to an interior router. This scenario is not an issue in our experimental setup as all the routers except the corner routers are along the edge of a zone.

### 4.5 Experimental Setup

We use Simics (53), a full system simulator to model a 16-core CMP architecture. The system is modeled as an in-order core for the SPARC ISA running the Solaris 9 operating system. The cache configuration for the simulated system is shown in Table 4.1.

The SpecOMP (6) benchmark suite was run on this system and traces collected from their runs for addresses accessed. Each benchmark was marked with an initialization phase until it enters the main program. Caches were warmed up for 500 million instructions (executed per processor) at this stage. Traces were collected for the next 1 billion instructions (per processor). Our network on chip simulator modeled a 4 X 4 mesh with a 5 stage pipeline for the router. The simulator modeled the ‘In-network cache coherence protocol’ and the in network caching proposed in this work. The traces

**Table 4.1 Simulation setup**

<table>
<thead>
<tr>
<th></th>
<th>Number of cores</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1 I-Cache per core</td>
<td>capacity</td>
<td>16KB</td>
</tr>
<tr>
<td></td>
<td>assoc</td>
<td>1 way</td>
</tr>
<tr>
<td></td>
<td>access cycle</td>
<td>1 cycle</td>
</tr>
<tr>
<td>L1 D-Cache per core</td>
<td>capacity</td>
<td>16KB</td>
</tr>
<tr>
<td></td>
<td>assoc</td>
<td>2 way</td>
</tr>
<tr>
<td></td>
<td>access cycle</td>
<td>1 cycle</td>
</tr>
<tr>
<td>Unified L2 per core</td>
<td>capacity</td>
<td>256KB</td>
</tr>
<tr>
<td></td>
<td>assoc</td>
<td>8 way</td>
</tr>
<tr>
<td></td>
<td>access cycle</td>
<td>6 cycles</td>
</tr>
</tbody>
</table>
obtained for each benchmark were fed into the network simulator and the average latencies were observed. For all our results, the base case is a system with ‘In Network Cache coherence’.

4.6 Experimental Results

The In-Network caching protocol proposed in this work is beneficial to the read latencies of accessing the memory. The write latencies are not directly modified by the protocol changes. However, the change in the traffic in the system changes them. We usually saw a very minor improvements for all of our experiments. The number we report in this work is the overall memory access latency for both reads and writes to the caches. Figure 4.10 shows the percentage improvement of memory access latency for the applications in the ideal caching scenario. The maximum performance improvement is shown by apsi (38%) and the least is shown by art (17%). On average the benefit shown by the benchmarks is 27%. It can be observed that there is a clear relation between the benefits shown by each benchmark and the number of accesses to shared lines. For example, apsi has the largest ratio of accesses to shared lines to accesses to private lines. Therefore, it benefits the most from the caching of shared lines. At the other end of the spectrum, the benchmarks art and wupwise have the two smallest values for the ratio of
accesses to shared lines to accesses to private lines. Correspondingly, they also show the two lowest improvements when Ideal caching is employed.

Figure 4.11 shows the percentage improvement for each benchmark for four different local store sizes when Independent caching is employed over the base case when no In network caching is employed. The trend that can be observed is similar to that of the Ideal caching case. The benchmark *apsi* shows the most improvement in general and the most improvement when the number of entries is increased. The benchmark *wupwise* and *art* show the least improvement in general and the least relative improvement when the local store size is increased. The reasons for this are the same as the Ideal caching case. A more interesting comparison can be made when the improvements of the benchmarks *mgrid* and *swim* are studied. Both benchmarks access shared data more than 90% of the time. However, *swim* shows a disproportionately large performance improvement compared to *mgrid*. The reason behind this is that the percentage of shared data in *swim* is very small compared to *mgrid*. Therefore, an increase in the size of the local store allows *swim* to cache a greater portion of its shared working set compared to *mgrid*. Therefore, *swim* has a larger improvement compared to *mgrid*.

Figure 4.12 shows the benefits of Zonal caching and Independent caching for a local store of 4 entries. Clearly Zonal caching is more beneficial for all benchmarks compared to Independent caching. The average improvement of the zonal caching over the independent caching is 17%. The largest benefits over Independent Caching are shown by *apsi* and the lowest by *wupwise* and *art* for the reasons elaborated earlier. From the results shown in this figure, it is clear that Zonal caching should be the preferred approach to caching shared data.

To contrast the zonal and the independent schemes further, Figure 4.13 presents the percentage improvement of Zonal caching with a 4-entry local store normalized to the percentage improvement obtained by Independent caching with a 16-entry local store. We see that on average, Zonal caching with a 4-entry local store provides 97% of the performance improvements of Independent caching with 16-entry local store. Therefore, we can conclude that using Zonal caching with a 4-entry local store is as good as using
Fig. 4.12 Latency improvement using Zonal and Independent caching with a 4-entry data store.

Independent caching with a 16-entry local which in turn means that Zonal caching effectively provides a four-fold increase in local store capacity compared to Independent caching.

4.7 Related Work

Power and performance consideration have motivated the use to chip multiprocessors (67). As buses are not scalable, packet switched networks were proposed for on chip communication (23). The basic router design is a five stage pipeline design is explained in (70). This was further improved in several works including (60; 9). Recent research in the field of on-chip networks is surveyed in (12; 36). Kim et al. (42) aim to reduce the number of hops for packets in the network by using high radix networks. Kumar et al. (47) reduce the number of hops by skipping some routers along the way. Our work takes an alternate approach by bringing the required data closer to the requester.

Liu et al. (52) also identify that the amount of shared data is low and the access to them frequent. As the number of concurrent accesses to a data block by the same processor is low, it is important that the data be placed where it is close to the processors that access it. They implement a fast center cell cache that resides in the center of the processor array. However, their work requires that the center cell be placed between
all the processors in a 4-core CMP and is thus not scalable. In contrast, this work considers a distributed cache placed in the routers of the networks and the experiments are provided for a tiled 16-core system and is scalable to networks beyond this size.

Eisley et al. (28) propose (see section 4.2.1) to embed cache coherence information into the routers in the NoC itself. This allows data requests to be routed to the processor whose cache contains the data. Our work builds upon (28) by adding a small data store the routers that stores data being transported. These enhanced routers will be able to service requests for data that they hold which reduces the read latency times. Other works that optimize the network for the cache coherence protocol include (11; 78).

Mizrahi et al. (56) implemented a cache coherence in network and also migrated data in the network. Their network is a multi stage interconnection network implemented as a tree. Our work focuses on a direct mesh of routers. None of the detailed pipeline proposals are presented in their work. Their evaluation scheme assumes a fixed number of writable variables and no particular benchmark unlike our work. Iyer et al. (37) suggest a similar technique of putting the caches in network switches. However, their implementation is for multiprocessor systems connected by generic network and not a solution for network-on-chip architecture. Since adding cache to a network switch doesn’t effect the overall frequency of a system, their works lacks the implementation detail presented in our work.

4.8 Conclusion

This chapter presented a novel approach to improving the effective performance of an NoC. Based on the premise that the amount of shared data in parallel applications is small and that the proportion of accesses to them are large, this work proposed to store shared data in the network fabric. The proposed approach involved adding a data store to the routers of the NoC. The work presented a coherence protocol and a router microarchitecture to support the proposed scheme. Three caching policies were examined and the results show a significant decrease in the network latency of the benchmark applications.
Chapter 5

Aging in On-Chip Networks

5.1 Introduction

Power and performance considerations have motivated the use of chip multiproces-
sors (CMPs). Performance per watt has transformed the microprocessor industry
from a race for higher gigahertz to the multicore era (3). All major semiconductor
companies now offer multicore designs for their products. Instead of making power hun-
gry complex cores, the focus has shifted to the building a large number of “simpler”
cores. The performance of a modern CMP, lies not only in the performance of the core,
but also the performance of the communication medium that connects these cores. To
this end, packet based router designs have been proposed and shown to be scalable
for a large number of processing elements (24). They overcome the limitations and re-
place the traditional bus based designs especially with larger number of cores (24). The
network-on-chip (NoC) is thus a critical component of modern CMPs. NoC performance
is particularly important for multi threaded applications as it has an effect on access to
the shared data (88).

Reliability concerns are one of the major limiting factor for transistor scaling (3).
Soft errors, process variation and premature transistor aging are among the major con-
cerns for future technology nodes (3). Aging in transistors reduces the useful lifetime of
systems and is a major challenge for the future technology generations. NoC routers are
no exception to this. In this work, we address the aging phenomena, on the Network on
chip routers. Specifically, we asses the impact of Negative Bias Temperature Instability
(NBTI), Time-Dependent Dielectric Breakdown (TDDB), Thermal Cycling (TC) and
Electromigration (EM). All aging effects have implications on future system designs by
causing dynamic, in the field permanent faults. NBTI, TDDB and TC are ailments that
effect all the logic on the chip including the cores and routers. Solutions proposed for
tackling these aging effects (76) at the core are equally applicable to the routers. While these aging effects still need to be addressed, the fundamental nature of the problem remains the same as these earlier works.

Electromigration, however is an interconnect failure, that takes effect only on the longer interconnects of the system (79). This includes the power grids, the clock tree networks and the links between the routers of a NoC. Interconnects that connect logic in the cores typically don’t reach these lengths. Thus, electromigration presents a unique problem in the context of NoC links. Traditional solutions to address EM, such as employing thicker wires and vias work better for resources such as power grids. The increased area overhead of such solutions to NoC links would result in placing additional constraints on the processor design and leads to lower density of wires and weaker designs. For instance increasing the width of the NoC link, can lead to lower bandwidth between the router to compensate for the increased area, increasing the serialization latency of the network packets. Thus, it is important to explore other solutions to the interconnect problem. EM is parasitic in nature to the lifetime of a link and system designers need to address this phenomena at all levels. In this work, provide solutions to tackle EM at the micro-architectural level.

The performance of the NoC is critical to the system performance. Several recent efforts have aimed to achieve a high-throughput, low-latency router and NoC (13; 28; 41; 48; 60; 69). One such optimization is the use of bidirectional links to adapt the varying bandwidth needs to the routers (21). NoC links can be unidirectional or bidirectional. Unidirectional links have the fixed channel width in each direction where as bidirectional links will have the combined capacity of these links and can provide the desired bandwidth for each router. Bidirectional links are advantageous, as the bandwidth of the each direction is configurable depending on the demand of the application in a particular phase. This configurability can lead to increased performance of the network, which directly leads to improvements in the application performance. Cho et al. (21; 49) shows the benefits of having such bidirectional links.

Per-router DVFS is another technique that helps the performance of the NoC. Two such works in the recent literature are presented in (86; 55). Increasing the voltage
of a particular router along with the link results in higher frequency of that router linearly (24). This can help decrease the congestion in the network and improve the overall performance of the system. The instantaneous buffer utilization of the router is a good metric to detect the congestion in a router (55; 31). Thus, an increase in the buffer occupancy can signal the need to increase the frequency of the router and process the packets therein quickly. This leads to a better performance in the network.

EM is a reversible phenomenon when current passes in the opposite direction. Traditional unidirectional links heal themselves as they pass opposite current on every switch of charge ($0 \rightarrow 1$ and $1 \rightarrow 0$) in the wire (79). Bidirectional links, on the other hand can cause multiple charges in the same direction even on a switch in the activity of the wire. We explore this in detail in Section 5.2.4.1. The accumulated charge leads to more vulnerable links. Similarly, increasing the voltage in DVFS can alleviate congestion and improve network performance. However, increasing the voltage also aggravates the fatigue caused by EM. This presents another scenario where the time to failure is being traded off for performance. While bidirectional links and DVFS schemes are certainly useful from a performance point of view, this work assesses their impact on the reliability of the NoC links. Aging is a a long term phenomena that can be solved with redundancy on the chip. Doubling the number of links of the NoC and using it replace the failed links doubles the lifetime of links on the chip. Thus, any solution proposed to fight aging without redundancy should be simple and not effect the overall performance of the system significantly.

To mitigate the impact of aging, we propose Aging Aware Routing Policies (AARP). AARP is a suite of adaptive routing techniques aimed to reduce the impact of EM in NoC links. The switching activity of links increases under higher load on the link. Adversarial traffic on a link, such as high usage rate of the inner links in uniform random traffic, can lead to lower lifetime values for the links. The non-uniform usage can lead to scenarios where certain links fail much faster than the rest of the links. AARP is designed to balance the activity of wires across the routing topology. This increases the lifetime of the frequently used links by sacrificing some life from the infrequently used lines.
The chief contributions of this work are:

- Characterizing and evaluating the MTTF values of links in NoCs.
- Modeling the impact of NBTI on the bidirectional link of a NoC.
- Measuring the decreased reliability of using bidirectional links and DVFS in NOCs.
- Proposing and evaluating an AARP and evaluating the impact on the MTTF of NoC links.

The rest of this chapter is organized as follows. Section 5.2 models the MTTF values for the various failure mechanisms considered in this work. In Section 5.2.4, we propose a method to model MTTF due to EM under AC and DC stress, which in turn gives us models for unidirectional and bidirectional links. Section 5.4 describes the experimental setup used for all our experiments. Section 5.5 presents the MTTF of links and the worsening of the problem with performance optimizations and congestion management. In Section 5.6 we describe the AARP algorithms and evaluate its impact on the performance of the network. Section 5.7 evaluates the MTTF of links under various AARP schemes. The related work is presented in Section 5.8 and we conclude our work in Section 5.9.

5.2 Modeling Mean Time to Failure

In this section, we present the model of the mean time to failure by permanent faults in modern devices. Specifically, we consider Negative Bias Temperature Instability (NBTI), Time-Dependent Dielectric Breakdown (TDDB), Thermal Cycling (TC) and Electromigration (EM). For NBTI, TDDB and TC we present the models discussed by Srinivasan et al. (77; 76). We present detailed models for Electromigration with focus on modeling the effect on AC and DC stress (79).

5.2.1 Negative Bias Temperature Instability (NBTI)

NBTI is the increase in the threshold voltage, and the resulting degradation of the mobility of charge and drain current of P-FETs (71) in devices. It is created due to
the interface traps and oxide charge by a negative gate bias at an elevated temperature. The increase in threshold voltage leads to degradation and eventually to the failures in the transistor. The MTTF equation for NBTI, Eq 5.1, is based of the work of Zafar et al. (89).

\[
MTTF_{NBTI} \propto \left[ \ln\left( \frac{A}{1 + 2e^{BkT}} \right) - \ln\left( \frac{A}{1 + 2e^{BkT} - C} \right) \right] \cdot \frac{T}{e^{BkT}}^{\beta}
\]

(5.1)

where T is the temperature, ‘k’ is the Boltzmann constant, \(A = 1.6328\), \(B = 0.07377\), \(C = 0.01\), \(D = -0.06852\) and \(\beta = 0.3\)

5.2.2 Time-Dependent Dielectric Breakdown (TDDB)

Successive generations of technology make the gate oxide thinner. However, the voltage across the dielectric doesn’t scale equally. Thus device scaling stresses the gate dielectric. TDDB is the phenomena of wearing down of the gate dielectric with time. It leads to the formation of a conducting path in the dielectric preventing any flow of charge from the source to the drain. The MTTF equation due to TDDB is as shown in Equation 5.2.

\[
MTTF_{TDDB} \propto \frac{1}{V} e^{\left( a - bT \right) + \frac{X + Y + ZT}{kT}}
\]

(5.2)

where T is the temperature, ‘k’ is the Boltzmann constant, \(a = 78\), \(b = -0.081\), \(X = 0.759\text{ev}\), \(Y = -66.8\text{evK}\) and \(Z = -8.37 \times 10^{-4}\text{ev/K}\)

5.2.3 Thermal Cycling (TC)

The operation of devices takes it through several thermal cycles. Switching on, switching off, going into sleep mode are some examples of when systems go through temperature cycles. These changes lead to accumulation of fatigue and eventually to permanent failure of devices. The failure due to TC is guided by the Equation 5.3

\[
MTTF_{TC} \propto \left( \frac{T}{T_{ambient}} - T_{ambient} \right)^q
\]

(5.3)

where T is the temperature, \(T_{ambient}\) is the ambient temperature (298K) and \(q = 2.35\)
5.2.4 Electromigration Modeling

Current in a metal wire running for a substantial period of time can cause the transport of metal ions in the direction of the current flow. Metal wires contain impurities such as vacancies, grain boundaries. These imperfections cause the electron flow of the current to pull the metal ions along with them (79). In modern VLSI circuits, during a lifetime of a wire, charge accumulation can result in the formation of shorts or voids. These both effects are undesirable and can lead to the failure of the entire system. EM occurs under both DC stress and AC stress. These stresses have differing levels of TTF as shown in Equations 5.4 5.6 The Time to Failure of a link under DC \( (TTF_{DC}) \) stress is usually characterized by the Black Equation shown in Equation 5.4

\[
TTF_{DC} \propto J^{-n} e^{\frac{E_a}{kT}}
\]  

(5.4)

where A and n are empirical constants, \( E_a \) is the activation energy, k is the Boltzmann’s constant, T is the temperature in Kelvin and J is the current density. The current density \( J \) is given by

\[
J = \frac{C.V_{DD}}{W.H} \cdot f \cdot p
\]  

(5.5)

where \( C \) is the capacitance, \( V_{DD} \) is the supply voltage, \( W \) and \( H \) are the width and height of the link respectively, \( f \) is the clock frequency and \( p \) is the switching probability of the wire. AC stress helps the self-healing in the links. The alternating charge flow directions help in reducing the charge accumulation in any particular direction. It’s time to failure is typically larger than that of DC stress. The Time to Failure of AC stress is given by (79).

\[
TTF_{AC} = 2f \times TTF_{DC}^2
\]  

(5.6)

5.2.4.1 Modeling Unidirectional and Bidirectional links

A unidirectional wire changes it charge flow for every switch in the current. This is shown in Figure 5.1 for the unidirectional link. Thus, there is no net switching in a particular direction. This motivates the use to AC stress for the unidirectional wire.
Fig. 5.1 Switching activity in unidirectional and bidirectional links

Thus,

$$TTF_{unidirectional} = TTF_{AC}$$  \hfill (5.7)$$

A bidirectional wire does not always balance out the charge flow in each direction. Instead, there can be a net switching in a particular direction. Figure 2 illustrates these scenarios. Note that a flow of ‘1’ from left to right and ‘0’ from right to left are the same (both are treated as ‘1’ from left to right in Figure 5.1). The stress in the bidirectional situation can be modeled as a combination of both AC stress and DC stress. Since the inverse of TTF’s are additive, we get

$$\frac{1}{TTF_{bidirectional}} = \frac{1}{TTF_{AC}} + \frac{1}{TTF_{DC}}$$ \hfill (5.8)$$

We evaluate the AC and DC components of the stress by computing the respective switching factors as shown in Figure 5.1.
5.2.4.2 EM variation with voltage

Increasing the voltage also leads to an increase in the temperature of the physical device. This leads to a decrease in MTTF. In this work, we conservatively estimate that the temperature doesn’t increase with an increase in voltage. Thus,

Equation 5.6 can be rewritten as

\[ TTF \propto f \times \frac{1}{V^2 f^2} \]  

(5.9)

We scale the voltage and the frequency in a linear fashion as shown in Table 5.2. Thus,

\[ TTF \propto \frac{1}{V^3} \]  

(5.10)

Section 5.5 shows the decrease in MTTF of the links due to DVFS.

5.3 Router vs Core MTTF

While EM is specific to longer links, aging artifacts such as NBTI, TDDB and TC which effect the logic in the router also effect the cores on a CMP. The main factor determining the MTTF values of the failures governed by Equations 5.1, 5.2 and 5.3 is the “temperature”. Figure 5.2 shows the peak temperature of the router and the core across a suite of applications. Section 5.4 describes the experimental setup for these results. We note that the core temperatures are consistently higher than the router temperatures across all applications. At an average, the peak temperature of the core is 22 degrees higher than that of the router. This represents about 40% of the peak router temperature.

Not surprisingly, this is reflected in the MTTF values of these failures. Figure 5.3 shows the ratio MTTF values for the router compared to the MTTF value of the core for NBTI, TDDB and TC. The router MTTF values for these failure models are much higher than the core. At an average, the router lifetime is at least twice of the core lifetime for each of these failure models.
Fig. 5.2 Peak Temperature

The results from Figure 5.3 are not sufficient to conclude that the NBTI, TDDC and TC are not a concern for the routers. However, we note that the solutions proposed for these failure models in literature for the processor structures (76), where the degree of the problem is higher, should be sufficient for the routers as well. This leaves us with tackling the EM in the links of a NoC.

5.4 Experimental Setup

We run our experiments for applications from the SpecOMP suite (6). Network traffic traces are collected for these applications by running them on Virtutech Simics (53). The system simulated had 16 cores with private-L1 cache of 128KB (both I-cache and D-cache) and shared L2 cache (16MB). The L2 cache was split into 1MB slices resulting in a tiled CMP fabric as shown in Figure 5.4. The cache configurations have been detailed in Table 5.1. The NoC properties are detailed in Table 2.

The traces collected from Simics were fed to an in-house network on chip simulator. The power numbers were obtained by running the applications on Wisconsin
Fig. 5.3 Router to Core MTTF comparison

Fig. 5.4 CMP layout
GEMS simulator (54) with WATTCH (18). These power numbers were then used to generate the temperature profile of the network using the HotSpot tool set (35). This experimental flow is shown in Figure 5.4. All the results were generated for a technology node of 32nm using the Predictive Technology Models (PTM) (2).

5.5 Estimating Mean Time To Failure by EM induced fatigue

5.5.1 MTTF values for unidirectional links

We first present the MTTF values for unidirectional links under normal operation in a mesh-based NoC. Figure 5.5.1 and 5.5.1 show the MTTF values across applications from the SpecOMP suite (6) at 32 nm and 45 nm technology nodes. A MTTF value greater than “30” is equivalent to a system with infinite age in our scenario. Note the MTTF value of wupwise is greater than this threshold and is hence not really suspectable to EM induced failures. In fact, we find throughout this work that the wupwise benchmark has MTTF value of greater than 30 years. However, we leave it in our results to indicate that some benchmarks, due to their low network usage, are not affected by EM.
Table 5.1 Cache Configurations

<table>
<thead>
<tr>
<th>Cache</th>
<th>Size</th>
<th>Assoc</th>
</tr>
</thead>
<tbody>
<tr>
<td>IL1</td>
<td>128 KB</td>
<td>1</td>
</tr>
<tr>
<td>DL1</td>
<td>128 KB</td>
<td>2</td>
</tr>
<tr>
<td>UL2</td>
<td>16 MB</td>
<td>16</td>
</tr>
</tbody>
</table>

This helps us maintain the averages so that the EM problem is not overstated. Our results indicate that the inner links have consistently lesser expected age than the outer ones in a mesh topology. The mesh is an asymmetric network and its inner links are stressed more than the outer links. At the 32nm technology node, the average MTTF of the inner link is 15.5 years and the outer link is 18.9 years. Thus, the inner links fatigue faster than the outer links by 22%. For 32nm technology, the benchmark *equake* has the lowest MTTF value for the inner link at 7 years. At 45nm the average MTTF for the inner link is 28 years. Thus, one technology step is decreases the lifetime of the link by 45% (or 10.5 years).

We also consider at the impact of aging for the torus topology. Figure 5.5.1 and 5.5.1 show the MTTF value for a torus topology of the NoC at 32nm and 45 nm technologies respectively. At 32nm, the average lifetime of the inner links is, 17.6 years and the outer links is 17.8 years. Comparing with results for mesh MTTF, we see that the torus topology fatigues slower than the mesh topology. Further, the torus topology doesn’t stress the inner part of a NoC due to its symmetric nature. Thus, the links age faster than the outer links by only 1.4% compared to 22% for the mesh. In fact, the MTTF of outer links is higher than the MTTF of inner links for *equake* benchmark as shown in Figures 5.5.1 and 5.5.1.

5.5.2 Interaction of reliability with performance optimization

Traditional designs for the links in the on-chip interconnect are a pair of unidirectional wires connecting adjacent routers. Cho (21) makes a case for bidirectional links between the routers. Figure 5.5.1 shows the equivalent systems. The advantage of a bidirectional link is that it can change the bandwidth of each direction depending on the current need.
Fig. 5.6 Mesh - 32nm

Fig. 5.7 Mesh - 45nm
Fig. 5.8 Mesh - All links

Fig. 5.9 Torus - 32nm
Fig. 5.10 Torus - 45nm

Fig. 5.11 Unidirectional and bidirectional links
In this work, we use the work by Cho et al. (21) to configure the bandwidth requirements in each direction. The microarchitectural modifications required to have adaptive bidirectional links work in the NoCs are described in their work. We provide a brief description of the mechanism here. Bidirectional links contain a **bandwidth arbiter** which decides the direction of the bidirectional links connecting adjacent routers. Each router indicates to a bandwidth manager its expected use of the bandwidth by sending its pressure. The allocator can configure the direction every cycle. Pressure is approximated as the outstanding packets that the router has queued. In our work, we simplify the possible bandwidth configuration by the manager. The smallest unit or each direction is half the bandwidth of the bi-directional links. Thus, the only possible configurations that the bandwidth manager can set of the bidirectional links are, all the bandwidth in one direction (which constitutes two different configurations depending on the direction), or half the bandwidth in each direction (like a unidirectional links system).

There is a direct correspondence between the request for a particular bandwidth and the type of packet the router has to send. A “request” packet (read request and write request) needs half the bandwidth. A response packet carries the data along with the address field and is hence two flits and requires the entire bandwidth.

The performance results for the adaptive bandwidth configuration are shown in Figure 5.5.2. The average packet latency for the *equake* decreases by 27%. Across all the benchmarks, the average decrease in the packet latency is 18%. Thus, bidirectional links is a useful tool for the architects to improve the performance of the NoC and in turn improves the system performance. However, bidirectional links have shorter life spans than unidirectional links. Figure 5.5.2, 5.5.2 shows the MTTF value of inner and outer bidirectional links for the mesh topology at 32nm and 45nm respectively. At an average, the MTTF of bidirectional links at 32nm is 10 years and at 45 nm is 12 years. This represents a 35% decrease in the MTTF value of the inner links and 34% decrease in MTTF for the outer links.
Fig. 5.12 Adaptive Bidirectional Links performance

Fig. 5.13 Bidirectional MTTF 32nm
5.5.3 Interaction of reliability with congestion control

Dynamic Voltage Frequency Scaling (DVFS) has been proposed to improve the power and performance of systems. Similarly, DVFS techniques improve the performance of the network. By controlling the frequency of the router, architects can alleviate congestions and lead to decreased packet latency in the network. Decreasing the voltage can lead to desirable power savings when the network is sparsely loaded. Recent works that have looked into DVFS of the network include works by Yanamandra et al. (86) and Mishra et al. (55). We propose a simple NoC DVFS schema, adapted from (86). Here, we briefly describe the algorithm used.

The challenge in doing a per router DVFS is to determine when to increase the voltage and frequency of an individual router. While running all the routers at the highest voltage might lead to a better performance, it is not ideal in terms of the performance per watt. Buffer utilization has been identified as the metric that can help answer this question. Both the DVFS works mentioned above use domains of buffer utilization to determine the frequency of a particular router. In this work, we limit our voltage
domain to three levels. Table 5.2 shows the buffer utilization domains that we use and the corresponding voltage levels. While our approach is much simpler than the work proposed in (55), it suffices for showcasing the adverse effects DVFS. The switching between a voltage domain takes place at the granularity of a ‘1000’ cycles. Doing it at a rate higher than that will not amortize the cost of switching between these domains (86).

Figure 5.5.3 shows the performance results of our DVFS schema. The average message latency decreases by 13.34% for the ammp benchmark. Across the benchmarks we see an average decrease of 12% in the packet latency in the network.

\[
V_{avg} = V_1 \cdot t_1 + V_2 \cdot t_2 + V_3 \cdot t_3
\]  

(5.11)
Fig. 5.16 DVFS MTTF - 32nm

Fig. 5.17 DVFS MTTF - 45nm
Table 5.2 DVFS Table

<table>
<thead>
<tr>
<th>Buffer Utilizations</th>
<th>Voltage</th>
<th>Freq</th>
</tr>
</thead>
<tbody>
<tr>
<td>$BU &lt; 0.15$</td>
<td>0.9 V</td>
<td>2 GHz</td>
</tr>
<tr>
<td>$0.15 &lt; BU &lt; 0.40$</td>
<td>1.0 V</td>
<td>2.2 GHz</td>
</tr>
<tr>
<td>$BU &gt; 0.40$</td>
<td>1.1 V</td>
<td>2.4 GHz</td>
</tr>
</tbody>
</table>

where $V_1, V_2, V_3$ are the three voltages shown in Table 5.2 and $t_1$, $t_2$ and $t_3$ are the fraction of time that the links have these voltages respectively. The average frequency can be calculated by a similar equation.

Outer routers are loaded lightly and usually don’t reach the higher voltage domains in DVFS. In our experiments, we found that the MTTF value of the outer links (Node 0-1 connection) increases by less than 1%. Figure 5.5.3, 5.5.3 show the MTTF of the inner links at 32nm and 45nm respectively. apsi is the worst effected by the DVFS with a decrease in MTTF value by about 12 years. At an average, the MTTF value is decreased by about 6 years which represents a decrease by about 39% from the baseline system of NoC with unidirectional links and no DVFS.

While adaptive bidirectional links and DVFS are still attractive solution for the performance and power considerations of the system, there is clearly a need for a solution to arrest the loss of lifetime. In summary, adaptive bidirectional links decrease the MTTF of links by 35% and the DVFS in NoC decreases the MTTF value by 39%. In Section 5.7, we evaluate the proposed AARP methodology and detail how it helps increase the MTTF value for not only the baseline unidirectional NoC but also with adaptive bidirectional links and NoC under DVFS.

5.6 Aging-Aware Routing Policies (AARP)

Routing is one of the key elements that architects can control to reduce the activity of links in NoCs. Link activity is not the same as the link ‘switching’ activity, which is the variable in the MTTF equations from Section 5.2.4. For example, passing the same value in successive cycles on a link contributes to activity but not switching activity as illustrated in Figure 5.1. However, counting the switching activity is very expensive in a distributed system such as a NoC. Switching activity is a per wire metric
unlike switching activity which is the same for all wires in a link. Using these counters is not scalable with an increase in number of routers in the network. In this work, we use activity to approximate the switching activity aging and our evaluation prove that it is indeed an effective technique. In fact, using activity in the aging equation instead of switching activity, gives a conservative estimate on the MTTF of the links (as switching activity is always less than activity). Apart from being a low-overhead solution, link activity is a metric already evaluated in several systems. It is traditionally a metric of interest for performance considerations and has applications in congestion aware routing.

AARP is a adaptive routing strategy at solving the EM in links. Based on link activity, the flow of traffic is diverted away from the “highly susceptible” links. This helps reduce the stress on links that age quickly. This redirection is guided by a threshold in the activity count per cycle of the link, crossing which makes the link be considered susceptible. This redirection can sometimes lead to degradation of performance. Thus, AARP sacrifices some performance for improve the lifetime of the link.

Adaptive routing techniques are common technique to avoid congestion in NoCs. Employing adaptive routing for aging purposes, introduces a new flavor to the problem. However, note that they are not the same. Links throughput can be very high even when there is no congestion in the system. Further, aggressive routing techniques to avoid aging “hotspots” can be detrimental in the long run as the packet passes through several more links than the originally intended. Congestion control techniques typically don’t sacrifice performance. In fact, rerouting the packet away from a detected congestion improves performance (31). AARPs rerouting might introduce aging issues into previously non-problematic links in the NoC (outer links that are usually underutilized). To study this, we propose three techniques within AARP namely, AARP-MinADP, AARP-FullADP, AARP-SmrtADP.

As indicative of the names, the difference between the techniques in the degree of adaptability in the routing algorithm of the NoC router.

AARP-MinADP uses the minimally adaptive routing to address the aging issue. The routing algorithm is as follows. The X or the Y direction (if both are possible) is chosen based on the link with the lesser activity. However, it chooses only from one of
Algorithm 1 AARP-MinADP Algorithm

1: AARP-MinADP
2: if \( \text{cur}_x = \text{dest}_x \) then
3: \( \text{Go towards dest in Y dir} \)
4: else \( \{ \text{cur}_y = \text{dest}_y \} \)
5: \( \text{Go towards dest in X dir} \)
6: else
7: if \( \text{activity}_x < \text{activity}_y \) then
8: \( \text{Go towards dest in X} \)
9: else
10: \( \text{Go towards dest Y} \)
11: end if
12: end if

these and doesn’t go away from the destination. If the link activity (link utilization per cycle) goes above a certain threshold, the links are configured as unidirectional wires as in the base case for a certain period of time until their activity becomes lower. The pseudo code for AARP-MinADP is presented in Algorithm 5.6.

Algorithm 2 AARP-FullADP Algorithm

1: AARP-FullADP
2: dir := Min activity direction
3: if \( \text{dir} = \text{away from dest} \) then
4: With prob: 0.66 ( \( \text{dir} := \text{towards dest} \) ) \{ /*switch back to current direction regardless*/ \}
5: end if
6: if \( \text{dir} = \text{towards dest} \& \text{activity}_x < \text{activity}_y \) then
7: \( \text{Go towards dest in X dir} \)
8: else \( \{ \text{dir} = \text{towards dest} \& \text{activity}_y < \text{activity}_x \} \)
9: \( \text{Go towards dest in Y dir} \)
10: else \( \{ \text{dir} = \text{away from dest} \& \text{activity}_w < \text{activity}_z \} \)
11: \( \text{Go away from dest in X dir} \)
12: else
13: \( \text{Go away from dest in Y dir} \)
14: end if

AARP-FullADP uses a fully adaptive routing for a packet. The pseudo code for this is presented in Algorithm 5.6. At every instance, a packet is routed to go in one of the four cardinal directions (or the number of directions it can move in for edge or corner routers). The idea is to route the packet along the least active path. This is done probabilistically. The packet goes in the correct direction (towards the destination) with a two-thirds probability even if the activity is lowest in a path away from the destination. This is done in order to avoid starvation that can result from frequently going away
from the destination. To further arrest the starvation, packet age is tracked and once it exceeds a threshold age in the network, its routing is switched to XY routing (24) regardless of the aging. A dedicated escape virtual channel is used to break the possible deadlocks (24).

**Algorithm 3 AARP-SmrtADP Algorithm**

1: AARP-SmrtADP Algo
2: if packet took a hop away from dest then
3: route around the problematic link /*The stressed link info is carried by the packet*/
4: end if
5: if \( cur_x = dest_x \) then
6: if link Y not stressed then
7: Go in correct Y dir
8: else
9: Go away from dest in X
10: end if
11: else \( \{ cur_y = dest_y \} \)
12: if Link X not stressed then
13: Go in correct X dir
14: else
15: Go away from dest in Y
16: end if
17: else
18: if \( activity_x < activity_y \) then
19: if link X not stressed then
20: Go in correct X dir
21: else
22: Go away from dest in X /*X is stressed ⇒ Y is stressed even more*/
23: end if
24: else \( \{ activity_y < activity_x \} \)
25: if link Y not stressed then
26: Go in correct Y dir
27: else
28: Go away from dest in Y /*Y is stressed ⇒ X is stressed even more*/
29: end if
30: end if
31: end if

**AARP-SmrtADP** balances out the disadvantages of the previous two schemes. It doesn’t restrict itself to just a minimally adaptive path. It’s adaptive routing algorithm is not as aggressive as the full adaptive routing. Instead of choosing links with the lowest activity every cycle, a packet deviates from the MinADP path only if the difference in activity is higher than the predefined threshold. This avoids the frequent hops of packets away from its destination. We set the threshold in the difference of switching activity to ‘1000’. Thus, the SmrtADP algorithm routes a packet away from the destination only if detects that the link(s) that leads the packet towards the destination are highly stressed. Further, it remembers the link which caused a step away from the destination.
and avoids it in further hops. Like the AARP-FullADP schema, it uses age counter to prevent starvation and has a dedicated escape channel to resolve deadlocks. The pseudo code for AARP-SmrtADP is presented in Algorithm 5.6.

5.6.1 AARP for bidirectional links and DVFS

AARP schemes are independent of the nature of links and can help arrest the aging in bidirectional links as well. Bidirectional links however fatigue faster than unidirectional links. Figures 5.5.1, 5.5.2 show that the bidirectional nature reduces the MTTF of NoC links. We can improve the lifetime of these links by configuring their use as unidirectional whenever they are stressed heavily. This leads to a loss in the performance improvements afforded by these configurable links. So, it must be done only when the link is highly stressed. However, we get an improved MTTF for links of the NoC. Links at a higher voltage present a similar quandary as bidirectional links. They fatigue faster than the baseline links. Figure 5.5.1, 5.5.3 quantify this notion. AARP decreases the fatigue in these links by restricting the voltage increase whenever a link reaches extreme fatigue. We consider a link to be highly stressed when its activity is higher than the other links of the corresponding router by a threshold. Through experiments, we see that a value of ‘3000’ for the threshold gives suitable results. Our results are not sensitive to this threshold as long as it is in the same order.

5.6.2 AARP Performance degradation

The AARP suite trades off performance for lifetime. In this section, we evaluate the decrease in performance that results in the use of AARP. We evaluate the impact of AARP on MTTF results in Section 5.7.

Figure 5.6.2 shows the results for the performance of the AARP suite on our baseline system of unidirectional links. The latency of each of the AARP-MinADP suffers almost no loss in performance. The performance of AARP-MinADP is within 3% of the unidirectional links. AARP-SmrtADP performs equally well with an average loss of performance of 4%. AARP-FullADP performs the worst of the lot, but the
Fig. 5.18 AARP Performance degradation Unidirectional

Fig. 5.19 AARP Performance degradation Bidirectional
performance loss is within 7%. Thus, we conclude that the entire AARP suite performs well for NoC with unidirectional links.

Figure 5.6.2 shows the performance of the AARP suite on a system with adaptive bidirectional links described in Section 5.5.2. AARP-MinADP performs the best again with an average decrease in the performance by 6%. AARP-SmrtADP and AARP-FullADP suffer a performance loss of 7% and 13% respectively. In particular, *equake* suffers a loss in performance by 21% when the routing is performed using the AARP-FullADP.

In Figure 5.6.2, the performance results of AARP suite is shown on a DVFS based NoC described in Section 5.5.3. MinADP, SmrtADP and FullADP cause an average performance loss of 5%, 6% and 9%.

We conclude that AARP-MinADP performs very close to the base systems in all systems with AARP-SmrtADP being a close second. The AARP-FullADP suffers a significant loss in performance especially in a adaptive bidirectional system with a 13% loss in performance.
In this section, we evaluate the MTTF values of links under our proposed aging schemas. Figure 5.21 shows the results for experimental evaluation of the AARP’s suite of aging aware routing on NoC with unidirectional links. AARP-SmrtADP is the most efficient at increasing the MTTF value of the link for both the inner and the outer links. At an average, the AARP-SmrtADP improves the MTTF value of the inner links by 60.4% and improves the value of outer links by 30.1%. AARP-MinADP doesn’t perform as well. It improves the MTTF of inner links by 15.8% and the outer links by 12.4%. AARP-FullADP presents an interesting case with varying results for the inner and outer links. It improves the age of the inner links by 30.6%, which is better than the savings using AARP-MinADP. However, it degrades the age of the outer links by about 7.6%. FullADP attains the benefits for the inner links by redirecting most of the traffic to the outer links. Thus it improves the age of the inner links at the cost of the outer links. Inner links fail faster than the outer links and so we conclude that the overall behavior is still beneficial compared to the AARP-MinADP.
Figure 5.6.2 shows the performance of AARP on MTTF of adaptive bidirectional links. AARP-SmrtADP improves the MTTF value of inner links by 31% and outer links by 20%. AARP-FullADP increases the MTTF of inner links by 16% and decreases the MTTF of outer links by 4%. AARP-MinADP improves the MTTF of both the inner and outer links by 8%. AARP performs better on the bidirectional links than unidirectional links. This is because, AARP controls the adaptive bandwidth controller. Under extreme stress, AARP configures the links as unidirectional which increases the MTTF. Note that despite AARP, the MTTF of bidirectional links is lower than their unidirectional counterparts.

In Figure 5.6.2 we evaluate the performance of AARP on a DVFS NoC. The MTTF value increases by 16% for inner links and 12% for outer links for the AARP-SmrtADP methodology. AARP-MinADP increases the MTTF by 6% and 7% respectively for the inner and outer links. AARP-FullADP improves the MTTF of inner links by 10%. As with the previous experiments, AARP-FullADP reduces the MTTF of outer links. In this experiment, we found the decrease to be 5%.
Thus, we conclude that our AARP-SmrtADP consistently improves the MTTF values of NoC links under various scenarios while effecting performance minimally.

5.8 Related Work

Soft errors, process variations and premature transistor aging are the main challenges for the reliability of current and future technologies (3). The problem of soft errors in on-chip networks was addressed in (87; 61). Nicopoulos et al. (63; 64) address the challenges of process variations in NoCs.

EM does not take effect for wires that are smaller than the Blech length (79). Typically, these wires are present between various gates at the logic level. Power grids, clock trees, buses and router links are some examples of interconnects that the EM affects. Abella (4) presents a microarchitects view of EM. The modeling of EM for links under AC and DC stress was presented in (79). The effect of EM for bidirectional links in multicores for bus based interconnects was presented in (5; 25). In particular, Refuel (5) alleviates some of the DC stress by artificially introducing packets that increase the AC
stress but contain the maximum DC stress. Refuel, by its nature needs to maintain counters for each wire in the link. This works well for bus based solutions on the chip. However, Network-on-chip fabric provides a distributed framework. The number of wires increases with an increase in the number of cores. Some layouts have banks of L2 as separate processing elements on chip thus increasing the size of the network. For example, a 16 node system will have a total of 3072 wires and a 64 node system will have a total of 8064 wires. (64-bit bandwidth per direction). Refuel will need as many counters in its implementation. Further, Refuel attacks only one component of the EM lifetime, namely the switching activity. AARP techniques, however reduce the activity of links in turn reducing the peak temperature of the links. Thus, AARP effectively attacks all the factors that effect the lifetime of a NoC link. Our solution uses the technique of avoiding highly susceptible areas to mitigate the aging effects in general and EM in particular.

5.9 Conclusion

Electromigration is an important limiter for NoC adoption at future technology nodes. In this work, we evaluate the MTTF of NoC links under the influence of EM at 32nm and 45 technology nodes for benchmarks from the SpecOMP suite. Further, we study the impact of performance enhancing techniques such as, adaptive bidirectional links and DVFS and study their impact on the lifetime of NoC links. After showing the degraded value of MTTF, we propose AARP, a suite of aging aware routing techniques. Of these techniques, AARP-SmrtADP improves the MTTF value by 20% for bidirectional links and 16% for links in DVFS NoC at the performance degradation of 4% and 6% respectively.s In conclusion, we show that AARP is a simple and an effective technique to deal with the EM on links.
Chapter 6

Variation-Aware Low-Power Buffer Design

6.1 Introduction

Aggressive scaling of transistor sizes and heavy on-chip integration have led to manufacturing uncertainties, termed as Process Variation (PV). PV is observed due to random effects, like Random Dopant Fluctuations (RDF), and systematic spatially-correlated effects, like dose, focus, and overlay variations (19). PV impacts various device characteristics, such as effective gate length, oxide thickness, and transistor threshold voltages (15). The exponential dependency of leakage power on the threshold voltage typically leads to large variations in leakage power consumption. It has been shown that leakage current can vary up to 20x in some systems (15). Also, the degree of variation increases with decreasing transistor sizes, thus aggravating this problem in future technology generations.

In this work, we propose an intelligent FIFO buffer architecture, aptly named IntelliBuffer, to address the impact of PV on leakage power. In particular, we consider the buffers in a Network-on-Chip router.

The poor scaling of wire delays when compared to gate delays has made communication on a chip very expensive (33). Packet-based networks have been proposed in place of traditional bus-based designs to alleviate this issue (22). In this work, we study a router-based Network-on-Chip architecture and focus our attention on the FIFO buffers.

In the second half of this chapter, we present a case study to assess the effectiveness of the IntelliBuffer scheme on an existing NoC buffer architecture. Specifically, we incorporate our proposed intelligent buffering structure into the dynamic Virtual Channel Regulator (ViChaR), a state-of-the-art NoC buffer implementation introduced
ViChaR employs a unified buffer structure, as opposed to the discrete and statistically partitioned VC buffers in conventional NoC routers. It was demonstrated in (66) that ViChaR’s efficiency can lead to substantial dynamic power consumption savings. In our case study, we present the necessary modifications required to augment ViChaR with IntelliBuffer in order to demonstrate improvements in leakage power consumption. Our subsequent experimental results corroborate the effectiveness of IntelliBuffer in yielding significant leakage savings within ViChaR.

The rest of the chapter is organized as follows: Section 2 analyzes the impact of PV on the NoC input buffers, Section 3 presents our proposed IntelliBuffer architecture, and Section 4 incorporates our design in an existing buffer implementation. Section 5 concludes the chapter.

6.2 Impact of Process Variation on NoC Input Buffers

Buffers are the chief contributors of leakage energy in a NoC router. Studies have shown that about 64% of the total router leakage power comes from the buffers. (20). To avoid the address decoding/encoding latencies of traditional memories and the access latencies associated with global bitlines/wordlines, router buffers are usually implemented as small registers (34). The buffers are accessed in the router pipeline only when the data
Fig. 6.2 Generic FIFO structure

enters or exits the router. The delays involved in the control path of the pipeline and the wait time during contention for a physical channel contributes to the time that the data spends in a NoC buffer. Deterministic routing algorithms traditionally employed in these systems cause imbalanced traffic thus leading to high contention in some of the physical channels. As a result, the amount of time data spends in the buffer increases. Thus, the average leakage power consumption of the buffers becomes important when considering the design of NoC routers. This observation motivates us to focus our attention to the power impact of process variation in the buffers, and particularly leakage power considering its dominance (20).

We analyze the variations in leakage power for both the 90 nm and 32 nm technology nodes using the Predictive Technology Model (2). A 128-bit register file was designed to test the variation impact on the leakage power consumption due to different parametric variations. We model PV in the registers due to the random variation effects. The model assumes parametric variations with total standard deviations of 5% on effective gate length (Leff) and 10% on threshold voltage (Vth) (33). Monte Carlo statistical analysis using the normal distributions of the parameters was performed in HSPICE.

Fig. 6.1 demonstrates the variations in the total leakage power consumption of a single 128-bit buffer slot at the 90 nm technology node. As expected, the buffer leakage
power follows a log-normal distribution due to the exponential dependence of leakage power consumption on the process parameters. The minimum and the maximum values differ by close to 4X, as observable from the figure. The leakage distribution, when fitted to a normal distribution, demonstrated a 90% variance around the nominal leakage power consumption.

6.3 IntelliBuffer: A Leakage-Aware Elastic Buffer Structure

The input port of a router has a dedicated k-flit FIFO buffer for each of its v virtual channels. These are implemented as a parallel FIFO, so that a flit need not traverse all slots in a pipelined manner before exiting the buffer. This is achieved by maintaining read/write pointers for the buffer in a FIFO order. Fig. 6.2 shows a generic implementation of the read/write pointers using two wrap-around shift registers, each of depth k (k=4). The read and write pointers keep shifting by at most a single slot position per clock cycle in the same direction. Thus, the buffers are utilized in a sequential order.

It is precisely this sequential behavior of FIFO buffers that renders them susceptible to process-variation induced leakage anomalies. Leaky slots cannot be differentiated from other slots, because of the rigid sequential access pattern. Thus, on an average, leaky slots are used as much as the other slots. To tackle this issue, we propose a modification to the pointer logic of the FIFO buffers to minimize leakage-power. We call the
IntelliBuffer differs from a conventional FIFO buffer in two ways. First, its read/write pointers do not blindly follow a sequential order. The slots are classified in advance based on their leakage characteristics. Then, the write pointer always tries to direct incoming flits to the least leaky slots, based on this leakage pre-classification. Second, all unused slots are supply-gated using sleep transistors to minimize leakage power consumption. To avoid performance penalties related to wake-up latency, we utilize the write pointer to power up the next empty slot as soon as the current slot starts being written to (40). This look-ahead power-up policy ensures that an empty slot is always active in anticipation of the next incoming flit. The combination of these two attributes creates a leakage-aware elastic buffer, with active slots “expanding” and “contracting” based on usage (i.e. incoming traffic). Most importantly, though, this expansion/contraction is not done arbitrarily; the leakiest slots are always the last to be used, ensuring that, under light traffic load, they are almost always switched off.

The IntelliBuffer architecture is shown in Fig. 6.3. The slot IDs are pre-classified in order of “leakiness” in the very compact Leakage Classification Register (LCR). The write pointer logic always chooses the first available slot which is highest in the leakiness list (i.e. least leaky). As slots are being written to, the slot IDs are written into a
small FIFO buffer (the Slot ID List). The head of this list forms the read pointer of the main IntelliBuffer flit FIFO, which holds all incoming flits. Conventional FIFO buffers are dominated by the datapath delay (i.e. reading and writing large flits into the slots). Thus, the pointer logic latency is masked by the datapath latency. Based on this realization, the additional control logic of IntelliBuffer was architected to operate in parallel with the buffer’s datapath; the control logic latency is completely hidden by the datapath latency. Synthesized results validate our assertion that IntelliBuffer is as fast as a conventional buffer structure. In 90 nm technology, the critical path delay of the new control logic is 0.24 ns, which is smaller than the buffer datapath delay of 0.35 ns. Additionally, because the overall buffer structure is dominated by the large flit slots (the Flit FIFO in Fig. 6.2), our additions to the control logic inflict a negligible 4% area and power overhead over the conventional FIFO buffer. The power increase is compensated by the much larger savings extracted from ordering the use of the main buffer slots.

Fig. 6.4 illustrates the leakage power savings generated by the IntelliBuffer architecture, as opposed to a structure which simply supply-gates empty buffer slots (40), for 90 nm and 32 nm technologies. IntelliBuffer achieves substantial benefits of around 24%, on average, at 90 nm, and savings of around 28% at 32 nm. Note that the parametric variations at 32 nm are expected to be even higher, which would accentuate the savings due to IntelliBuffer. Furthermore, even though the difference in percentage savings between 90 nm and 32 nm is small, the absolute savings are much greater, since the 32 nm node is significantly leakier. The Leakage Classification Register (LCR) is populated off-line at router startup using Built-in self-test (BIST) techniques (26).

6.4 IntelliBuffer in ViChar

6.4.1 Preamble

In order to further assess the efficacy of the IntelliBuffer scheme, we employ a state-of-the-art dynamic NoC buffer structure, recently introduced in (66). This all-encompassing dynamic buffer manager is known as ViChaR (a Virtual Channel Regulator). ViChaR is characterized by its two fundamental operating principles: (1) a
unified buffer, instead of the partitioned approach used in generic NoC routers, and (2) a unified control logic module, which coordinates the functionality of all virtual channels within a single input/output port. As such, ViChaR operates at the granularity of one input/output port; i.e., a typical NoC router would require five ViChaR components (North, East, South, West, PE). ViChaR’s premise of operation is dynamically dispensing buffer resources, based on prevailing network traffic. This includes both dynamic VC allocations and dynamic VC depth allocation. These two goals are achieved through the use of the aforementioned structures:

1. The Unified Buffer Structure (UBS) replaces the constrained approach of the individual and statically-partitioned FIFO buffers of conventional NoC routers. Where the latter typically employ one FIFO buffer of fixed depth for each virtual channel within an input port, ViChaR’s UBS utilizes a logically uniform buffer structure, where resources are not pre-assigned to specific VCs. This principle is illustrated in Figure 6.5(a). If we assume that a conventional NoC router has \( v \) virtual channels, each with a depth of \( k \) flits, then ViChaR unifies all \( vk \) flit slots into one unified structure, as shown in the figure. Hence, UBS can provide the NoC router with two enormous advantages:

   (a) A variable number of VCs
   (b) A variable buffer depth for each VC.

2. The Unified Control Logic (UCL) is a novel table-based component comprised of several sub-modules, each operating independently and in parallel. Therefore, the
control path can complete its operation in a single clock cycle, which is an essential attribute in the severely resource-constrained NoC environment. Furthermore, it is the centralized nature of the control logic that provides ViChaR with the information required to dynamically dispense buffer resources to optimally accommodate network traffic at any given time. Figure 6.5(a) depicts the difference between the centralized UCL and the distributed, independent control units for each VC in a conventional NoC router.

The benefits offered by the flexible and dynamic operation of ViChaR are significantly reduced network latency (25%), with a minimal power overhead (2%) (66). Most importantly, though, ViChaR can achieve the same performance as a conventionally buffered router by using 50% less buffer space. This result is of paramount importance, since it allows for the halving of buffer size with no effect on performance. This translates into net area and dynamic power benefits of around 30% and 34%, respectively, over the entire NoC router (66).

ViChaR’s performance benefits are a direct consequence of its highly efficient buffer utilization, which allows flits to percolate through the router faster (i.e., with less blocking). Figure 6.5(b) illustrates how ViChaR alleviates a serious drawback of existing NoC buffers: under-utilization. In statically-assigned VC buffers, if only part of a packet occupies a buffer at a given time, the empty slots of that buffer cannot be reassigned to a new packet to avoid packet mixing. This trait of FIFO buffers can lead to under-utilization of the buffers. ViChaR, on the other hand, does not suffer from this issue, as the unified buffer dynamically varies both the number of VCs and each VC’s depth according to incoming traffic needs. It is important to note that, in order to enable fine flow control granularity, ViChaR assigns at most one packet to each VC. Further, systems that allow multiple packets to share the same VC are susceptible to Head-of-Line (HoL) blocking, where a blocked packet impedes the progress of another packet which happens to use the same VC. Hence, a vk-flit ViChaR structure (Figure 6.5(a)) can support at least v VCs (with each VC occupying the maximum of k flits) and at most vk VCs (with each VC occupying the minimum of 1 flit) at any given, and anything in-between these two extremes. In general, this attribute leads to fewer but deeper VCs under light
traffic, and more but shallower VCs under heavy traffic. It has already been established that ViChaR can lead to smaller buffer requirements without affecting performance, thus yielding substantial dynamic power benefits. However, leakage power consumption still remains unaddressed. It is precisely this concern that we aim to tackle in this case study, by augmenting the ViChaR architecture with our IntelliBuffer scheme to lower leakage power consumption.

In the next sub-section, we present the main components of the ViChaR architecture, and identify the module that needs modifications in order to accommodate IntelliBuffer.

6.4.2 ViChaR Component Analysis

The main components comprising the ViChaR architecture are the UBS and the UCL. The unified buffer (UBS) is controlled by the unified control module (UCL) through the Arriving and Departing Flit Pointers Logic. The UCL can be decomposed into five main sub-modules: (1) The VC Control Table, (2) the Arriving/Departing Flit Pointers Logic, (3) the Token (VC) Dispenser, (4) the VC Availability Tracker, and (5) the Slot Availability Tracker.

Following is a brief description of each of the UCL sub-components, leading to the proposed augmentation of IntelliBuffer to the ViChaR architecture:
1. VC Control Table - This is a compact table that holds the slot IDs of all flits currently residing in the buffers. The table is organized by VC number, with each VC containing at most a single packet. The table can be configured to either assume constant packet size, or variable-sized packets. The VCs can include non-consecutive buffer slots to allow full-flexibility in buffer utilization.

2. Arriving/Departing Flit Pointers Logic - This module directly controls the Input and Output MUXes/ DEMUXes of the UBS, and its operation is coordinated by the VC Control Table module. When a flit departs, its location in the VC Control Table is invalidated. When a new flit arrives, the pointer logic guides the flit to the appropriate slot in the UBS, based on the flit’s VC ID and information from the Slot Availability Tracker.

3. Token (VC) Dispenser - The Token (VC) Dispenser is responsible for dispensing free VCs to requesting packets. VCs can be viewed as tokens; they are granted to new packets and then returned to the dispenser upon release. The decision to grant a new VC or not is made based on information provided by the VC Availability Tracker. The Dispenser grants new VCs on a First-Come-First-Served (FCFS) basis.

4. VC Availability Tracker - The VC Availability Tracker holds a log of all unused VCs in the VC Control Table. The VC Availability Tracker provides this information to the Token (VC) Dispenser, which, in turn, dynamically assigns VCs to new incoming packets accordingly.

5. Slot Availability Tracker - The Slot Availability Tracker keeps a log of all unused UBS flit slots. When a new flit arrives, it is stored into a slot indicated by the Slot Availability Tracker. It consists of a small table, as shown in Figure 6.5(c). Each row of the table corresponds to one buffer slot. For each entry in the table, one bit indicates that the VC/Slot is available (logic 1) or occupied (logic 0). The Next Available Slot Pointer (Figure 6.5(c)) points to the top-most available entry.

Clearly, the slot availability tracker is the module that must be modified in order to accommodate IntelliBuffer as it decides which buffer slot is to be used next. To
incorporate IntelliBuffer into the ViChar architecture, the slot IDs should not follow a blind sequential descending order. Instead, they are now classified based on their leakiness, and stored in the Leakage Classification Register (Figure 6.3). The write pointer directs incoming flits to the least leaky slots. All unused slots are then supply-gated to minimize leakage power consumption, as described earlier.

To evaluate the benefits afforded to ViChaR by IntelliBuffer, we modified the cycle-accurate simulator from (66) to include the detailed architectural implementation of IntelliBuffer. Further, we back-annotated the synthesized power consumption numbers of the IntelliBuffer structure into the simulator. This allowed for a detailed comparison between the ViChaR and the ViChaR+IntelliBuffer implementations. The results of these experiments at 90nm technology are summarized in Figure 6.6. Results for three different UBS sizes (8, 12, and 16 flit slots per input port) at various network traffic injection rates are shown in the figure. Evidently, incorporating IntelliBuffer into ViChaR results in 21% leakage power savings on an average. The percentage savings are higher when the number of buffers are higher. Further, the higher the number of buffers the lower the percentage utilization for a given injection rate. This increases the opportunity to keep the most leaky buffer slots shut down for longer periods of time. Also, the percentage savings in the power consumption decrease with increased injection rate. With an increase in the injection rate the opportunity to save from the more leaky slots decreases. For low injection rates, the power savings can be as high as 30%. These results clearly demonstrate the efficacy of IntelliBuffer, as applied to an existing buffer architecture.

6.5 Conclusions and Future Work

Process variations in a chip are the result of manufacturing imperfections. PV can lead to substantially degraded performance and violate power budgets. In this work, we propose an intelligent FIFO buffer design, called IntelliBuffer, which can lead to average power savings of 24% for the buffers in a Network-on-Chip router. In our experiments we considered only static variations. However, temperature and voltage fluctuations can
lead to dynamic variations that cannot be captured statically. As future work, we intend to address these issues.
Chapter 7

Conclusion and Future work

Conclusion

The quest for a thousand-core CMP will be incomplete without a low-power reliable on-chip network that stitches these cores together. Several recent works have focused on optimizing the router micro architecture, exploring network topologies for a many-core system, and customizing the network to suit the on-chip traffic patterns. Along with these design improvements, it is critical to provide resources for guaranteeing reliability of the communication system. These additional resources elevate the stress on the already power constrained on-chip network design. Consequently, studying the power and reliability tradeoffs of the distributed and shared on-chip network is an interesting problem. To the best of our knowledge, we believe that this is the first work to deal with the power and reliability tradeoffs in NoCs. In this work, we tackle various sources of faults such as radiation induced soft errors, process variation induced supply noise as well as dynamic operating condition variations. The primary goal of this work is to minimize the power and performance cost implications of providing reliability guarantees in the NoC.

In Chapter 2, we exploit the error tolerance of some applications and selectively turn off the ECC protection for a fraction of the time in order to save power. In order to provide reliability guarantees, the vulnerability is bounded leading to loss in throughput. Static and dynamic schemes are proposed to improve the power and performance tradeoffs for given reliability guarantee. This work models single event upsets (SEUs).

In Chapter 3, we account for multiple bit errors due to noise voltage. Further, variations in operating conditions lead to higher probability of errors. An example of a technique that results in operating condition changes is Dynamic Voltage and Frequency Scaling (DVFS). DVFS is used in modern processors for power and performance tradeoffs.
We study the adverse effect of DVFS on reliability and provide a reconfigurable EC as a solution to tackle the problem.

Chapter 4 motivates intelligent design for NoCs to improve the power of the system. NoCs are an ideal place to determine which data is used frequently. Such profiling information can be used to provide quick access to these frequently accessed data. We provide a router microarchitecture enhancement to achieve this by introducing the ‘Router Local Cache’ unit into the router pipeline. The results showed lucrative improvements in performance for modest area overhead.

The past decade has seen extensive research for improving the lifetime of processors. This was done by either providing device level optimizations for reducing the aging, or by augmenting processors to tolerate wearout, or by improving the accuracy of detecting the aged components. The network on chip circuitry is not an exception to these aging phenomena. In Chapter 5, we deal with the aging phenomena in NoCs. Specifically, we motivate the problem of Electromigration (EM), on the NoC links. We evaluate the mean time to failure (MTTF) of NoC links due to EM. Adaptive bidirectional links (21) and DVFS in NoCs (86; 55) are techniques used to improve the NoC performance. However, they are detrimental to the lifetime of an NoC. To improve the MTTF of NoC links we propose, Aging Aware Routing Policies (AARP), a suite of routing techniques that detect stressed links and reroute the traffic to improve their lifetime. The results show that AARP is an attractive technique even in the presence of lifetime decreasing techniques such as the use to bidirectional links and DVFS.

Chapter 6 studies the effect process variation (PV) on the leakage power on NoC buffers. NoC buffers are a dominant power consumer of the NoC router. Process variation can negatively effect the leakage power of storage structures. To fight this trend, we propose IntelliBuffer, an process variation aware buffer management technique to reduce the leakage power consumption of the buffers. Our results indicate the effectiveness of this technique in mitigating NoC buffer leakage. This technique is generic and can be applied to other FIFO buffer on the chip as well.
Lifetime reliability of devices has some key questions that need to be addressed as the field matures. A metric to study the tradeoffs of lifetime versus achievable performance during the lifetime is one such metric. Currently, there is no standard way to compare a system with a certain lifetime to another system which is slower and has more lifetime. The answer to question is likely to vary with the domains and needs to be addressed in future research. Reliability is a problem whose importance is going to increase with the coming generations of technology. Emerging technologies alleviate some of the concerns but still haven’t matured to be used by the microprocessor industry at large. This motivates the continued efforts to provide highly reliable devices and systems.
Bibliography


Vita

Aditya Yanamandra

Aditya Yanamandra was born in India on September 11, 1983. He received his Bachelor of Technology degree in Computer Science and Engineering from the Indian Institute of Technology, Madras, in 2005. He joined the PhD program in Computer Science and Engineering at the Pennsylvania State University, University Park in August 2005. He worked as a teaching assistant from August 2005 to December 2006. Since then, he has been as a research assistant by the program. He has worked as an intern at Intel, MA in Summer 2008 and Summer,Fall 2009. He has served as a reviewer for several conference and journals such as MICRO, ISCA, HPCA, ICCAD, ASPLOS, PACT, DAC, ISVLSI, TCAD and TVLSI. His research interests include reliability of systems, Networks-on-Chip, and performance impact of scratchpad memories. He is a student member of the ACM.