Runtime Adaptive Scrubbing in Fault-Tolerant Network-on-Chips (NoC) Architectures

A thesis presented to

the faculty of

the Russ College of Engineering and Technology of Ohio University

In partial fulfillment

of the requirements for the degree

Master of Science

Travis H. Boraten

May 2014

© 2014 Travis H. Boraten. All Rights Reserved.
This thesis titled
Runtime Adaptive Scrubbing in Fault-Tolerant Network-on-Chips (NoC) Architectures

by
TRAVIS H. BORATEN

has been approved for
the School of Electrical Engineering and Computer Science
and the Russ College of Engineering and Technology by

Avinash Karanth Kodi
Assistant Professor of Electrical Engineering and Computer Science

Dennis Irwin
Dean, Russ College of Engineering and Technology
ABSTRACT

BORATEN, TRAVIS H., M.S., May 2014, Electrical Engineering

Runtime Adaptive Scrubbing in Fault-Tolerant Network-on-Chips (NoC) Architectures
(69 pp.)

Director of Thesis: Avinash Karanth Kodi

As aggressive scaling continues to push Multi-Processor System-on-Chips (MPSoCs) to new limits, complex hardware structures and stringent area and power constraints will continue to diminish reliability. Waning reliability in integrated circuits will increase the susceptibility of transient and permanent faults. There is an urgent demand for adaptive Error Correction Coding (ECC) schemes in Network-on-Chips (NoCs) to provide fault tolerance and improve overall resiliency of MPSoC architectures. The goal of adaptive ECC schemes should be to maximize power savings when faults are infrequent and increase application speedup by boosting fault coverage when faults are frequent. In this thesis, I propose Runtime Adaptive scrubbing (RAS), a novel multi-layered error correction and detection scheme with a three mode area efficient configurable encoder for encoding packets on the switch-to-switch (s2s) layer, thus preventing faults from accumulating up the network stack and onto the end-to-end (e2e) layer. As fault rates fluctuate I propose a dynamic methodology for improving fault localization and intelligently adapting fault coverage on demand to sustain graceful network degradation. RAS successfully improves network resiliency, fault localization, and fault coverage compared to traditional static switch-to-switch (s2s) schemes. Simulation results demonstrate that static switching RAS improves network speedup by 10% for Splash-2/PARSEC benchmarks on a 8 x 8 mesh network while reducing area overhead by 15% and incurring on average 6.6% power penalty. Further, my dynamic ECC scheme maintains 97.88% of performance and incurs on average 20% power penalty.
I dedicate this thesis to my grandfather, Henry E. "Hank" Boraten.

"Glad you got to see me... The pleasure was all yours"

Hank (1928-2011)
ACKNOWLEDGMENTS

I would first like thank my parents for putting three boys through college and myself through multiple degrees. Second, I would like to thank my advisor, Dr. Avinish Kodi, for encouraging me to continue my education, his relentless support, and for always challenging me in my work. Lastly, I would like to thank my graduate committee members, Dr. Kaya, Dr. Dill, and Dr. Stinaff for making my MS degree possible.
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>3</td>
</tr>
<tr>
<td>Dedication</td>
<td>4</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>5</td>
</tr>
<tr>
<td>List of Tables</td>
<td>8</td>
</tr>
<tr>
<td>List of Figures</td>
<td>9</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>11</td>
</tr>
<tr>
<td>1.1 Motivation</td>
<td>11</td>
</tr>
<tr>
<td>1.2 Fault Classification</td>
<td>14</td>
</tr>
<tr>
<td>1.2.1 NoC Router</td>
<td>15</td>
</tr>
<tr>
<td>1.2.2 Link Faults</td>
<td>18</td>
</tr>
<tr>
<td>1.2.3 Control Faults</td>
<td>18</td>
</tr>
<tr>
<td>1.3 Fault Tolerant Techniques</td>
<td>19</td>
</tr>
<tr>
<td>1.3.1 Error Correction and Detection Codes</td>
<td>19</td>
</tr>
<tr>
<td>1.3.2 Encoding Schemes</td>
<td>24</td>
</tr>
<tr>
<td>1.4 Related Work</td>
<td>26</td>
</tr>
<tr>
<td>1.4.1 Joint Crosstalk Avoidance Triple-Error-Correction Quadruple-Error-Detection</td>
<td>26</td>
</tr>
<tr>
<td>1.4.2 Dual-Layer Transient and Permanent Error Co-Management</td>
<td>27</td>
</tr>
<tr>
<td>1.4.3 ARIADNE Fault Tolerant Routing</td>
<td>28</td>
</tr>
<tr>
<td>1.5 Proposed Design</td>
<td>29</td>
</tr>
<tr>
<td>2 RAS Architecture</td>
<td>31</td>
</tr>
<tr>
<td>2.1 RAS Hybrid-Approach</td>
<td>31</td>
</tr>
<tr>
<td>2.2 RAS Micro-Architecture</td>
<td>34</td>
</tr>
<tr>
<td>2.3 Encoder and Decoder Design</td>
<td>36</td>
</tr>
<tr>
<td>2.4 Dynamic Mode Selection</td>
<td>40</td>
</tr>
<tr>
<td>3 Performance Evaluation</td>
<td>46</td>
</tr>
<tr>
<td>3.1 Static Analysis</td>
<td>46</td>
</tr>
<tr>
<td>3.1.1 Network Speedup</td>
<td>48</td>
</tr>
<tr>
<td>3.1.2 Area</td>
<td>53</td>
</tr>
<tr>
<td>3.1.3 Power</td>
<td>54</td>
</tr>
<tr>
<td>3.1.4 Reliability</td>
<td>57</td>
</tr>
<tr>
<td>3.2 Dynamic Analysis</td>
<td>58</td>
</tr>
</tbody>
</table>
3.2.1 Network Speedup and EDP ................................. 61
3.2.2 Mode Transitions ........................................... 62
3.2.3 Decoding and Retransmissions ............................. 63

4 Conclusions ....................................................... 66

References .......................................................... 67
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Table shows how SEC and SECDED hamming codes expand as encoded bit lengths increase.</td>
<td>23</td>
</tr>
<tr>
<td>3.1</td>
<td>Simulation parameters [1] for obtaining Splash-2/PARSEC traces using the SIMICS application suite.</td>
<td>47</td>
</tr>
<tr>
<td>3.2</td>
<td>Test cases for evaluating my proposed dynamic switching model.</td>
<td>59</td>
</tr>
</tbody>
</table>
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>2011 International technology roadmap for semiconductor (ITRS) SoC power consumption trends[2].</td>
<td>13</td>
</tr>
<tr>
<td>1.2</td>
<td>Runtime fault probabilities for increasing synthetic traffic loads caused by increasing temperature [3].</td>
<td>16</td>
</tr>
<tr>
<td>1.3</td>
<td>Standard NoC Router with four VCs and four buffers per VC.</td>
<td>17</td>
</tr>
<tr>
<td>1.4</td>
<td>A 3-bit binary cube illustrating the spatial difference between code words.</td>
<td>20</td>
</tr>
<tr>
<td>1.5</td>
<td>Bottom: Top level view of an encoded channel. Left: (7,4) Binary hamming encoder. Right: (7,4) Binary hamming decoder where 'd' bits are encoded with 'p' parity bits.</td>
<td>22</td>
</tr>
<tr>
<td>1.6</td>
<td>Traditional NoC encoding schemes in mesh topologies.</td>
<td>25</td>
</tr>
<tr>
<td>2.1</td>
<td>TOP: The encoding checkpoints locations for e2e, s2s, and hybrid encoding schemes. BOTTOM: High level state diagram for a multi-layered RAS encoding scheme and how network with the RAS methodology may adjust the encoding over time.</td>
<td>32</td>
</tr>
<tr>
<td>2.2</td>
<td>RAS e2e and s2s micro-architecture.</td>
<td>34</td>
</tr>
<tr>
<td>2.3</td>
<td>The hamming based JTEC and JTEC-QED encoders shown on the left provided the motivation for the RAS 3-mode configurable s2s encoder. The three figures on the right show the active components and control lines for each RAS mode. The non-active components are shown in grey.</td>
<td>37</td>
</tr>
<tr>
<td>2.4</td>
<td>Block diagram for the RAS decoder showing the active and disabled (grey) components for each mode.</td>
<td>38</td>
</tr>
<tr>
<td>2.5</td>
<td>Proposed state diagram for routers switching modes in dynamic RAS.</td>
<td>41</td>
</tr>
<tr>
<td>2.6</td>
<td>Example of RAS dynamically switching between CRC (white), RAS-W (purple), and RAS-S (red).</td>
<td>42</td>
</tr>
<tr>
<td>2.7</td>
<td>TOP: The residual flit error rate (RFER) of the compared networks as the bit error rate (BER) is increased. BOTTOM: Counter threshold needed to measure the BER for each injection rate.</td>
<td>45</td>
</tr>
<tr>
<td>3.1</td>
<td>Network speedup as the flit error injection rate increases.</td>
<td>49</td>
</tr>
<tr>
<td>3.2</td>
<td>Network speedup as link failures increase. As links fail, DUAL-Layer and RAS boost fault coverage to continue using the unreliable links while Ariadne must reroute packets around them.</td>
<td>50</td>
</tr>
<tr>
<td>3.3</td>
<td>TOP: The number of routers in each mode. BOTTOM: The amount of subnetworks formed as link failure percentage increases.</td>
<td>51</td>
</tr>
<tr>
<td>3.4</td>
<td>Router area overhead for each architecture.</td>
<td>53</td>
</tr>
<tr>
<td>3.5</td>
<td>Normalized EDP as the percentage of link failures is increased.</td>
<td>54</td>
</tr>
<tr>
<td>3.6</td>
<td>Router power consumption for each architecture.</td>
<td>55</td>
</tr>
</tbody>
</table>
3.7 We show how the BER affects the RFER of each ECC. By examining their RFER we can measure the fault tolerance of each scheme. 58
3.8 In this figure we show how we chose t1 and t2 for each test case in Table 3.2. The BER used to test each case were chosen before, in between and after each vertical line. More details on the evaluation setup for will be discussed in my thesis defense. 60
3.9 TOP: Speedup for each case as the BER increases. BOTTOM: EDP for each case as the BER increases 61
3.10 As the BER was increased we tracked the total number of mode transitions for each case. 63
3.11 As the BER was increased we tracked how each flit was decoded for each case. 65
3.12 As the BER was increased the number of retransmissions in each case was logged. 65
1 INTRODUCTION

1.1 Motivation

In the world of integrated circuits (ICs), technological advancements in nanofabrication are paramount if chip designers are expected to extract the maximum performance as theorized in Moore’s Law. Moore’s law has been the de facto rule in research and development of the semiconductor industry which predicts that the performance of future integrated chips will approximately double every two years. To meet the performance goals in each technology generation, transistor fabrication technology must continue to advance the limits of transistor sizing, speed, and overall chip density. The continuous demand for high performance computing (HPC) from integrated circuits has enabled and driven the design of densely packed chips with full system-level integration of processing elements, memory modules and peripherals into a fully embedded Multiprocessor System-on-Chip (MPSoC) architecture. In the past, designers leveraged aggressive frequency scaling and parallelism to improve performance by exploiting available chip densities. Today, research is focused on improving the scalability of MPSoC architectures and to enable integration of hundreds to thousands cores [4]. As more cores work in tandem on the chip, simultaneous memory access on the chip has also increased. To meet the high performance communication requirements needed to support multiple cores, packet based Network-on-Chip (NoC) architectures have emerged as the leading strategy to deliver scalable, modular, and adaptive networks in MPSoCs [5] [6]. Traditional point to point bus-based networks have not been able to overcome the fundamental challenges associated with wire scaling (due to high capacitance and resistance) and increased wire delay [7] [8]. Commercially, NoC-based interconnects have already been proposed and implemented in many MPSoCs such as Tilera’s 72-core architecture [9], Intel’s 80-core TeraFlops [10], and NVIDIA’s CUDA 512-core Fermi
architecture [11]. As NoCs will continue to be the backbone of future on-chip architectures, exhaustive research has expanded the field of NoCs into several directions: network topologies, routing algorithms, router micro-architectures, wireless, optical interconnects, buffered and non-buffered networks and fault tolerance.

As technology keeps pushing beyond the sub-micron region, chip designers will have to mold NoC architectures to meet energy, performance and reliability standards for overall chip requirements. Furthermore engineers must design architectures capable of scaling with each technology generation. Unfortunately as transistors scale, their performance and power consumption characteristics are scaling unfavorably due to an increase in leakage current. The power consumption of a transistor has two components, the leakage power and the switching power. Leakage power results from unwanted current flowing through narrow channels formed within the source to drain regions of a transistor while the transistor is switched off. Switching power, on the other hand is also referred to as the dynamic power is the amount of power consumed when a transistor switches from off to on state. The dynamic power consumed is dependent on frequency, voltage and capacitance as shown in the equation $P_{\text{dynamic}} = CV^2f$ where $C$ is capacitance, $V$ is voltage and $f$ is frequency. Figure 1.1 shows the international technology roadmap for semiconductors (ITRS) SoC power consumption trends. The Figure shows the breakdown between leakage and dynamic power consumption extrapolated from 2011 data [2]. Figure 1.1 shows that as the feature size decreases, overall power consumption and the percentage of the power due to leakage power scales at an alarming rate. Feature size or minimal gate-length is the term used to describe a particular technology generation. NoCs tend to consist of complex logic and power hungry buffers (memory) to store packets at multiple routers as they reach their destination. In next generation MPSoCs, the power consumed by the buffers is expected to reach 46% of overall router power [12], and forcing NoCs to consume an alarming portion
Figure 1.1: 2011 International technology roadmap for semiconductor (ITRS) SoC power consumption trends[2].

of total chip power. This is evident in Intel’s Teraflops architecture which consumes 28% [10] of total chip power.

Beyond logical, architectural and fabrication flaws and design challenges, fault tolerance in MPSoCs is also becoming a growing concern for performance critical NoCs. To properly analyze how the overall performance of a NoC can vary under a wide range of scenarios, the traffic injection rate can be adjusted to simulate a network under different traffic load and link utilization. A network experiencing high load(which implies majority of the links in the network are busy), may consume more power and raise the temperature for highly utilized components. Figure 1.2 shows how runtime faults and
router temperature behave as the injection rate of an 64-core system is stepped from zero load, low load, high load, and saturation. As the injection rate is increased along the x-axis, the warmest routers saw roughly a 10°C increase in temperature and a 2% increase in probability of fault. On an average, all routers experienced a 5°C increase in temperature and a 1% increase in probability of fault as the injection rate increases. The dynamic fault rate observed in Figure 1.2 which was caused by temperature alone, provides motivation for NoCs to incorporate fault tolerant techniques. For NoCs to effectively improve fault tolerance and prevent systematic failure, multiple approaches must be taken to handle the wide variety of faults and the locations they may appear because both will dictate the fault tolerant techniques required and whether performance is affected.

In the remaining subsections I will cover the classification of faults, why, how and what they can affect, how to mitigate them and the trade-offs of those methods, followed by related research in fault tolerant NoCs and finally my proposed design.

1.2 Fault Classification

To understand how to mitigate faults, I must first expand the classification of faults into two categories, transient and permanent. Mitigation techniques for each type and variation of fault will be discussed in one of the following subsections.

A transient fault is a fault that is temporal and is not assumed to be caused by a logical flaw or broken component. Transient faults are also called soft errors. A soft error is caused by one of the following: process variation [13], capacitive and inductive crosstalk [14], alpha particles, cosmic rays, thermal neutrons and other radiation effects [15]. Permanent faults emulate broken components, which can be caused by electro-migration stress [16], stuck-at faults [17], soft errors causing wear out, and fabrication flaws. When a soft error occurs, a soft error may have no systematic affect, cause silent data corruption (SDC), or a detected unrecoverable error (DUE). The soft error rate (SER) of a system characterizes
the rate in which silent data corruption (SDC) and detected unrecoverable errors (DUE) events occur. The SER can be sensitive to temperature as shown in Figure 1.2, but as well as altitude, where 100x increase can be seen in aerospace applications [18].

In NoCs, soft error and transient fault susceptibility is expected to increase [19], [20], [21] because of diminishing feature sizes and aggressive dynamic voltage scaling. If current trends in voltage scaling and feature sizes are to continue, further research to improve fault tolerance is needed [20]. As the primary role of NoCs is to act as the central nervous system of MPSoCs, ensuring data integrity is top most priority, followed by the goal of efficiently servicing that data. Fault tolerance in NoCs is typically broken into the protection of links between routers, and the control logic within the router micro-architecture. To limit the performance implications of recovery mechanisms after the faults manifest, it is critical for NoCs to integrate fault tolerance within the NoC infrastructure and provide graceful network degradation. If fault tolerance was handled by a higher communication layer protocol instead of the physical layer, it could lead to excessive retransmissions in a faulty NoC and would cripple future MPSoCs. Providing NoCs with fault tolerant safe guards however, requires additional hardware and overhead to supply that capability. Since NoCs already account for a significant 10-15% portion of overall chip power consumption, dynamic fault tolerant schemes should be considered to efficiently adjust fault tolerance on demand by lowering coverage and saving power when faults are infrequent, and boosting coverage when faults are common.

1.2.1 NoC Router

To service packet based communication between cores, each core is given a network interface and attached to an input and output of a router in a NoC. The architectural design of router in NoCs is referred to as the router micro-architecture. A typical NoC router is shown in Figure 1.3. In large networks packets are broken into multiple segments
called flits to increase throughput. A head flit which leads transmission contains routing information for route setup, followed by body flits containing the payload and the final flit called the tail flit used to signal the end of a packet. In the micro-architecture, critical pipeline stages control flit progression to service data requests and responses from each input to the correct output port. The typical pipeline consist of five stages: routing computation (RC), virtual channel allocation (VA), switch allocation (SA), switch traversal (ST), and link traversal (LT). During the RC stage, information from the head flit is read-
in so the packet can be sent to the correct output port. Once the output port is selected the VA and SA stages allocate a virtual channel (VC) and the desired output port taken to reach the downstream router. VCs are used to continuously buffer incoming flits and prevent deadlocks, a severe network condition capable of crippling all network traffic when contention stops flit progression. When a flit reaches the ST stage, it traverses the switch, and lastly exits the router by traversing the link in the LT stage. Without VCs, if a downstream input port is full, flit movement is blocked until contention is relieved.
1.2.2 Link Faults

For the data links between routers, transient faults are of primary concern because of the response needed for recovery. If a transient fault occurs on a link, and detection is successful, the packet in which the fault was detected must be dropped and requested for retransmission. The segmentation of packets into flits allows for retransmissions to be taken with higher granularity between routers. For detection to take place, error correction codes (ECC) and error detection codes (EDC) must be employed. In-depth information on ECC and EDC codes will be covered in the following subsection. When a fault materializes within a link, it can affect the link several ways. The fault could cause a SEU, extrapolate into multiple bit upsets, form a string of errors called burst errors, or affect adjacent wires via crosstalk. If a fault goes undetected in the payload of a packet, systematic failure may occur and or result in data corruption. If a fault were to affect the head of a flit, the flit may be routed in the wrong direction or sent to the wrong destination node and render the requesting node to signal for retransmission. In the event permanent faults develop on a link, if the applied ECC is still capable to correct the faults, the link may continue to be used, otherwise the link is considered broken and can no longer be used.

1.2.3 Control Faults

If faults materialize in the RC, VA, or SA stages of the micro-architecture pipeline, flits may arrive out of order, at the wrong destination, or in the wrong virtual channel (VC) and not eventually cause live lock and dead lock. Live lock is when a resource becomes blocked and starves components downstream from progressing. Dead lock on the other hand occurs when at least two competing packets are competing for access to the same resource and are blocking each other. Similar to the arbitration example mentioned in the motivation section, faults may have little impact and incur virtually no performance penalty. In the ST and LT stages, the faults could cause data corruption and require retransmission.
In all stages, permanent faults may impair execution and prevent normal operation. When routers can no longer correctly service requests and responses, the router is considered ‘failed’. When a router fails the attached core starves, making it no longer usable. Each failed router also affects the network performance as it alters the network topology. Since traffic must be rerouted, average hop count, latency, power and energy costs increase. Now with the understanding of the classification of faults and their affects on NoCs, evaluation of mitigation techniques and current research to improve fault tolerance will be examined.

1.3 Fault Tolerant Techniques

In this section current research in fault tolerant NoCs will be evaluated and provide further motivation for adaptive solutions.

For fault tolerant NoCs, the goal is to provide a framework for which faults may be detected, localized, corrected if possible, and then allowed to resume. If an occurrence has permanent fault ramifications, hardware must be in place to bypass and circumvent the broken hardware in order to achieve graceful network degradation and minimize single points of failure. The accuracy and granularity at which fault localization occurs, dictates the amount of hardware needed to diagnose and determine the impact a permanent fault will have on the network.

1.3.1 Error Correction and Detection Codes

Before I go further into the evaluation and analysis of ECC in NoCs, I must first properly illustrate the fundamentals of error correction.

The basic idea behind error detection and correction is that some level of reliability can be gained by adding redundancy to the transmitted message. After transmission a downstream receiver can check the additional bits for consistency and determine the validity of that message. Error correction is very similar to encryption in the sense that a message is encoded and decoded at the source and destination but for very different
purposes. Encryption’s goal is to obscure the data, while error correction is concerned with ensuring data integrity. The simplest form of ECC and EDC algorithms is parity. By appending parity bits computed from a deterministic algorithm, the parity bits can be recomputed at the destination and compared for error detection. Research in error detection codes has led to the design of repetition codes, codes that repeat bit streams across a channel in blocks, parity bits as just mentioned, checksums sums, cyclic redundancy checks (CRCs) [22] and error-correcting codes. ECCs are different from EDCs such that when an error occurs, retransmission is not needed and the already included parity data is analyzed to compute where the error(s) occurred. Research in ECCs has led to the design of many codes such as, hamming [23], hsiao[24], JTEC[25], BCH[26], reed-solomon[27], and turbo codes[28].

To differentiate between the multitude of codes that exist in the study of information theory, the hamming distance metric is used to express the signal distance when analyzing codes. Hamming distance is defined as the minimal number of bit positions that differ in value between two code words of equal length. The easiest way to illustrate this concept is by considering a 3-bit binary cube shown in Figure 1.4. The distance between each 3-bit sequence varies from sequence to sequence. An ECC with a hamming distance of 1 and
A codeword length of 3, would mean every 3-bit sequence in Figure 1.4 would be a valid codeword because there exist at least 1-bit difference between each sequence. If instead a hamming distance of 2 was required for a 3-bit codeword, only sequences that differed in 2-bits would be valid. For example, '000', '101', '011' and '110' could be one of the acceptable set.

The hamming distance of a particular coding algorithm allows insight into the maximum number of bits that can be detected and corrected. If a code has a hamming distance of $d$, any codeword that experiences $d - 1$ errors results in a distinguishable sequence that is detected beyond the set of valid codewords. If a codeword experiences more than $d - 1$ errors, the faulty sequence may go completely undetected or falsely diagnose a correctly transmitted bit(s) as incorrect. The nomenclature devised to fully characterize and properly describe a particular hamming code used across a binary channel is given in the form $(n,k,m)$, where $n = 2^r - 1$, $k = 2^r - 1 - r$, and $m = r$. and where $r$ is the number of parity bits, $k$ is the amount of data bits being encoded and $n$ is the total number of bits after encoding.

To correct errors at the receiver using binary hamming codes, the transmitted parity bits are temporarily stored and sent to the syndrome generator. A second set of parity bits is then computed from the received payload to compare against the received parity bits. After the second set of bits is computed, they are compared with the received bits in the syndrome generator. The syndrome generator has two roles, to act as a comparator, and to localize the fault location for the corrector circuit. Due to the carefully constructed sequence of XOR gates used in the parity generator, the parity bits are transmitted in an arrangement such that the location of a faulty bit can be identified from the result and swapped in the corrector. The corrector uses the output from the syndrome generator to locate the location of the faulty bit and flips the signal appropriately.
Due to the variance of algorithms and their complexity, implementation overhead eliminates many ECCs from being applicable in NoCs as encoders and decoders as these must meet tight on-chip area and timing constraints. Because NoCs experience low error rates similar to DRAM technology, hamming codes are a popular design choice [29] due to their simplicity, yet minimal fault coverage. In the bottom of Figure 1.5, a top level diagram of a generic encoded transmitter and receiver across a noisy channel is illustrated. Detailed block diagrams of the encoder and decoder can be seen on the left and right respectively. To encode 4-bits from the example in Figure 1.5, three parity bits are computed in the parity
Table 1.1: Table shows how SEC and SECDED hamming codes expand as encoded bit lengths increase.

<table>
<thead>
<tr>
<th>Number of bits</th>
<th>Number of parity bits for SEC</th>
<th>Number of parity bits for SECDED</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2 to 4</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>5 to 11</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>12 to 26</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>27 to 57</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>58 to 120</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>121 to 247</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>248 to 502</td>
<td>9</td>
<td>10</td>
</tr>
</tbody>
</table>

generator block and added transmitted alongside the data. The computation of each parity bit can be shown in the parity generation matrix table. A cell containing an ‘X’ represents the inclusion of that bit in the XOR tree for the computation of each parity bit of that row. For decoding of incoming bits, the received parity bits are sent to the syndrome generator while a second set of parity bits are computed to compare against for validity. The result of the syndrome generator then determines the correctness of the transmitted data and signals the corrector to find the location of the erroneous bit needing correction, and the result is flipped. The ordering arrangement of data and parity bit placement is key to finding the faulty bit location. The result of applying XORs to the parity bits tells us which overlapping bits were affected, even if the fault arose in one of the transmitted parity bits. For example, if parity bit p1 and p2 were flagged, the sum of the index from each bit results the index of the erroneous bit. In this case $1 + 2 = 3$, and signals that the bit indexed at location 3 needs to be flipped.
The encoder and decoder shown in Figure 1.5 can only correct and detect 1 error, but if a single overall parity bit is added, the hamming distance of each codeword can be increased by 1 and allow for single error correction double error detection (SECDED). Another advantage to hamming codes is how well they scale. This can be seen in Table 1.1. The amount of additional bits needed to encode a very large set of data is relatively low compared to the amount of data actually being encoded and applicable for technology generations. The disadvantage with on-chip implementations however is that large XOR trees are needed to compute the small number of parity bits. The trees grow much more complex and consume more power and area. Increasing bit lengths also increases the probability of more faults occurring, but the amount of total corrected and detected faults is always 1 and 2 respectively. Causing the need for research in the implementation of more robust ECCs or configurable encoders but with the challenge of finding a scalable and possibly dynamic solution.

1.3.2 Encoding Schemes

While research in fault tolerant NoCs is broad, my proposed design focuses mainly on methods used to mitigate transient faults in datapaths. For transient faults ECC and EDC codes can be implemented in a variety of schemes and strengths. Improving fault tolerance however comes at the cost of power, performance, and area trade-offs. The trade-offs between design choices are dependent on the size of the NoC, the ECC or EDC scheme, and the fault rate of the system. In end-to-end (e2e) encoding schemes, encoding and decoding takes place only at the source and destination as shown in Figure 1.6. In multi-hop networks e2e schemes have poor fault localization and produce costly retransmissions when fault rates are high. Switch-to-switch (s2s) schemes on the other hand offers a more robust solution with encoding at each router. On the downside, if fault rates are low, s2s consumes excess power because flits are encoded and decoded at every
hop. Retransmission in s2s also has less impact on the network because retransmission can be realized at each router instead of only at the destination as shown in Figure 1.6. The major drawback with s2s schemes is that the encoding cost exists in every router and that intra-router faults are not protected against since flits are decoded at the input ports and not encoded until right before link traversal. Between those stages, faults may occur during buffer write and switch traversal stages. To mitigate intra-router faults a cross code-disjoint detection (cdd) schemes as shown in Figure 1.6, can eliminate the coverage gap by adding additional encoders and decoders but at the cost of further overhead. With that known, choosing a single scheme may not provide a one size fits all solution when factors such as thermal sensitivity, voltage, and frequency scaling affect the SER at runtime. Research in hybrid schemes that can adapt to adjusting fault rates at runtime are needed with the goal of providing graceful network degradation. Adaptive schemes must also aim at efficient power
consumption by increasing coverage when faults are frequent, and reducing coverage when faults are low.

1.4 Related Work

In this section the research related to my work will be introduced and surveyed in order to shed light on current research in regards to fault tolerance in NoCs, the challenges that exist with current techniques and possible solutions for how future generations will need to adapt.

1.4.1 Joint Crosstalk Avoidance Triple-Error-Correction Quadruple-Error-Detection

As mentioned previously, SECDED hamming codes are often desired in NoC and dynamic random access memory (DRAM) implementations because of their simplicity to implement, and low overhead. The downside is their never changing ability to only correct one error and detect two at most. Fortunately the error rate in current generations does not require much stronger codes as long as the length of time between encoding and decoding data does not become large enough that faults accumulate and increase the vulnerability. With aggressive scaling, multiple-bit correction codes will become imperative as NoCs expand and increase average hop counts, forcing data to cross longer and more vulnerable paths. In efforts to improve reliability and NoC efficiency the authors in [25] developed a novel joint crosstalk avoidance code capable of supplying triple-error-correction (JTEC) and quadruple-error-detection (JTEC-QED). JTEC expands the hamming distance of a traditional (38,32) SEC hamming code of 3 to 6 by duplicating the output bits and further increasing it to 7 by adding an additional overall parity bit resulting in a JTEC (77,32). The duplication of bits serves to reduce the effect of crosstalk in adjacent wires and correct multiple bit errors. While the size of the JTEC encoder is the same as the traditional hamming encoder but with duplicated wires, the decoder requires both the original and duplicated bits to be decoded, requiring double the components and
an increase in complexity for the syndrome generator and corrector. If the SEC hamming code is swapped for a SECDED Hsiao code, an optimized hamming code, JTEC is capable of expanding to JTEC-QED, allowing for all three bit error corrections and quadruple error detection. In my proposed design the JTEC-QED encoder and decoder is used as the foundation for my proposed configurable encoder.

1.4.2 Dual-Layer Transient and Permanent Error Co-Management

With aggressive scaling and the increasing demand for high availability, NoCs must adapt and resiliently downgrade faulty links and routers to achieve graceful network degradation and limit the performance impact of oscillating fault rates. Relying on a static level of fault coverage with a one size fits all approach is overly ambitious. Excessive fault coverage creates additional overhead and wasteful power consumption. Architectures with inadequate static coverage risk immediate failure rendering full or partial chip service interruption. Employing NoCs with the flexibility to adjust fault tolerance on demand provides a scalable method to reduce the costly latency and speedup penalties that build up compared to mitigation techniques that avoid areas by rerouting and bypassing. In [30] the authors propose a Dual-Layer method for handling transient and permanent faults. Instead of rerouting around links causing faults, a configurable error correction coder offers adjustable fault tolerance for transient faults. To account for permanent faults, each link has a set of spare redundant wires unused until a dedicated history table diagnoses permanent wire damage in the network. If transient fault coverage needs to be boosted, the link encoders can switch from using a two-way interleaved SEC Hamming code to a four-way SEC Hamming code. When encoders switch to the second mode, the serialization of data must split flits in half to account for the additional bits needed for the extra error correction capability. The alternative is shutting down the link. When a link is shutdown routers must reroute packets around them, extending the average hop count which increases the
average packet latency and overall power consumption. Boosting to a stronger ECC thus minimizes the performance impact but is traded-off for additional area and power penalty required for the second ECC implementation. The trade-off however heavily depends on the complexity of the stronger ECC. The selection of ECC can also be tuned to the type of faults an architecture may be more susceptible too, such as burst errors, cross-talk, multiple bit upsets and single event upsets. The challenge is providing efficient, scalable designs to accommodate every major fault type a link may experience with a single area efficient encoder.

1.4.3 ARIADNE Fault Tolerant Routing

When multiple transistor failures inject faults at a rate that surpasses the use of ECCs employed in a NoC, fault tolerant routing algorithms must be in place to reconcile the removal of nodes and paths to enable and service demands for continued operation. Supplying fault tolerant interconnects like several mitigation fault challenges, balances improved reliability with performance and area. Achieving performance near normal operation and adhering to the same standards disallowing deadlock, livelock, and minimizing hop counts after reconfiguration is required. Reconfiguration however comes at the cost of complexity. In [31] the authors proposed Agnostic Reconfiguration in a Disconnected Network Environment (ARIADNE), a reconfigurable algorithm aimed at supporting aggressive scaling, unbounded faults and without pattern constraints in path creation. ARIADNE offers a desired solution based off of up*/down* routing and can execute on any topology. Up*/down* labels all links as either up, or down, and disallows packets from taking paths that transverse a down link followed by an up link and allowing all remaining turns. The resulting routing combination is built by broadcasting initially from a single root node and spanning to forming a tree of up and down links. If nodes
become disconnected and the network becomes partitioned, sub-networks will be formed and packet transmission is limited to only within that sub-network.

By adding fault tolerant routing algorithms to the arsenal of techniques architectures such as ARIADNE, and Vicis[32] faulty links and nodes can be efficiently avoided, but should only be considered complementary to ECC schemes because of the cost of boosting coverage with configurable encoders can minimize the penalty of costly retransmissions and increasing hop counts.

1.5 Proposed Design

As aggressive scaling continues and pushes MPSoCs to new limits, increasingly complex hardware structures and stringent area and power constraints will continue to diminish reliability and become more susceptible to the fluctuations of transient and permanent faults. For intermediary communication between cores, MPSoCs rely on NoCs as the exclusive interconnect fabric. Since NoCs act as the sole transport of data between cores, the unpredictability of fault locations makes ensuring data integrity and graceful network degradation a primary concern for further research. To prevent the likely-hood of full chip failure, NoCs must tailor adaptive ECC schemes to handle a wide array of irregular effects caused by cross-talk, process variation, multiple-bit upsets and soft errors. With adaptive ECC schemes, failure-prone paths can be optimized to maximize performance to improve resiliency and increase overall chip efficiency. When faults are infrequent, the fault coverage can be reduced to augment power savings and increased on demand. In this thesis, I propose Runtime Adaptive Scrubbing (RAS), a novel multi-layered error correction and detection scheme with an area efficient 3-mode configurable encoder that encodes packets on the end-to-end layer and optimizes fault coverage on the switch-to-switch layer in NoCs to improve application speedup and power savings. With RAS higher fault granularity and graceful network degradation can be achieved by extending the life (usage) of error-prone
links by tailoring ECC control. Simulation results suggest that scrubbing successfully improves network speedup by 10% for Splash-2/PARSEC benchmarks on a 8 x 8 mesh network while incurring on average 6.6% power penalty per flits and saving 15% in area compared to Dual-Layer [30].

The major contributions of this work are as follows:

• An adaptive multi-layer encoding scheme that improves network resiliency and graceful network degradation by improving overall fault coverage over traditional single layer ECC schemes.

• Improved NoC energy-efficiency by boosting fault coverage rather than bypassing components.

• An area efficient s2s ECC encoder, configured to detect cross-talk, burst errors, and intra-router faults across 3 different modes.

The remaining chapters of this thesis will explain my proposed design and methodology associated with Runtime Adaptive Scrubbing (RAS), the RAS micro-architecture, encoder design, dynamic mode-selection, performance evaluation, and conclusion.
2 RAS ARCHITECTURE

In this chapter, I examine the proposed Runtime Adaptive Scrubbing (RAS) architecture. This chapter discusses the proposed hybrid encoding scheme, the RAS micro-architecture, switch-to-switch (s2s) encoder and decoder design, and lastly a proposed method for dynamically choosing different modes as different links and routers experience various fault rates.

2.1 RAS Hybrid-Approach

In order to provide higher resiliency in NoCs using my proposed RAS technique, I show a high level state diagram in Figure 2.1 illustrating the general encoding states and how each layer affects different states. A network employing RAS combines e2e encoding with adjustable s2s encoding. The states shown in Figure 2.1 represent the possible s2s encoding modes for each router. To maintain minimal fault coverage all packets are initially encoded in CRC using the CRC-32 IEEE polynomial standard [33] as they enter and exit the network. The primary role of CRC-32 is to maintain baseline fault coverage as packets travel in the network, regardless of the s2s encoding they may receive along the way. When faults are detected in the CRC of a packet, counters at each router where the errors are detected are updated to estimate the error rate and localize the fault location. When the error rate at a router surpasses threshold $t_1$ as shown in Figure 2.1 the router transitions to RAS-W (weak) where a weak s2s ECC is enforced. Should a router experience an error rate beyond the threshold $t_2$, the router upgrades its s2s encoding to RAS-S (RAS-strong). If a router resides in a s2s encoding mode, all incoming traffic to that router is s2s encoded using the ECC defined for that mode. To support the additional bits needed to s2s encode packets with a stronger ECC as in RAS-S, traffic is split from 64 to 32 data bits per cycle. The leftover bits are utilized to encode the data more aggressively and upgrade the fault coverage. At any time, if the error rate should decline and diminish below threshold $t_1$
Figure 2.1: TOP: The encoding checkpoints locations for e2e, s2s, and hybrid encoding schemes.
BOTTOM: High level state diagram for a multi-layered RAS encoding scheme and how network with the RAS methodology may adjust the encoding overtime.
or $t_2$, the fault coverage can be downgraded accordingly. Lastly, NoCs should employ a third s2s mode, RAS-P (power), for two purposes; (i) to circumvent permanent faults and (ii) to provide an energy efficient ECC mode when minimal s2s coverage is required in under-utilized routers. By using RAS-P when a link has experienced a permanent fault, the link can still be used assuming the amount of faults does not exceed the strength of the ECC employed in RAS-P. Upgrading to a stronger ECC in this scenario proves much more efficient than shutting down the link and forcing packets to reroute which I will show in my performance evaluation (Chapter 3).

A foreseeable challenge with adaptive schemes such as RAS, is detecting when and where to boost the coverage. For example, if a fault is detected in the CRC of a packet, there is no information on the origin of the fault. If the injected fault is transient, periodic built-in self-test (BIST) hardware will have difficulty in finding the root of the fault. If the network environment also experiences varying fault rates, the challenge becomes significantly more important if we are to ensure graceful network degradation. In RAS, I tackle this issue by using what little information we do have and boost the coverage immediately upon detecting faults in CRC. By using a deterministic routing algorithm such as up/down routing [31], I can localize the fault’s origin to the packet route and upgrade each router to RAS-W and increment their respective counters. By upgrading to RAS-W, I increase the chance of detecting any future faults by improve locality and encoding on the s2s layer. With the additional coverage, more accurate fault rates can be determined and if the fault rate exceeds either of the thresholds ($t_1$ or $t_2$), we can upgrade accordingly. Improving the granularity in which faults are detected allows us to minimize their impact on the network. For every CRC detected fault, a costly full packet e2e retransmission is required. If coverage is boosted in the hot spot locations, fault detection on the s2s layer can be immediately corrected or signal packet retransmission at s2s level. Improper or early mode selection can also have adverse effects on the network. In my proposed encoder
design for RAS, each mode affects the network speedup and energy cost differently. In my implementation of RAS-S, I improve fault coverage at the cost of split transmission. Additional energy and latency is further required to encode each 32-bit segment of the original 64-bit flit. For now we have a basic understanding of the RAS methodology and acknowledge the new challenges with adaptive schemes bring to NoCs. Dynamic mode switching and threshold selection will be discussed in more detail later in this chapter.

2.2 RAS Micro-Architecture

In this section I provide an overview of the proposed RAS micro-architecture for fault tolerant designs in NoCs. While the s2s encoders and decoders are important components
of the RAS micro-architecture, I will discuss those in the Encoder and Decoder design section.

In Figure 2.2, a top level view of the proposed RAS micro-architecture is shown. In RAS, each processing core has a network interface connecting the processing element into the network. In a typical mesh network topology, each core is attached to its own dedicated router. Each router is then connected in a 2 dimensional grid as shown on the left in Figure 2.2 with two unidirectional links connecting each router to its adjacent routers. In my proposed design, all data must first be encoded with CRC-32 (256,224,32) before it enters the network to form a 256 bit packet. The 256 bit packet consists of 224 data bits which is appended with a 32 bit check code from my CRC e2e encoder. After the packet is encoded, the packet is broken into four equal 64 bit flits and queued for injection into the router. Once the flit wins arbitration, it is free to make their way through the current router and transverse the network.

In RAS, routers consist of a 5-stage pipeline with the following stages: decoder, input arbitration, routing and switch traversal, output arbitration and encoder. At each stage, control blocks manage the progression in every cycle. Each stage of the router is shown in Figure 2.2. In the first stage of the pipeline, flits enter into the decode stage. In the decode stage, s2s ECC decoding takes place as flits are received on the input port of the router. At which point the type of decoding that occurs is defined by the mode of the destination. To ensure flits are originally encoded with the correct s2s ECC, triple modular redundancy (TMR) control lines signal upstream routers to encode flits appropriately. Control lines in this stage also signal requests for flit retransmissions and acknowledgements of successful flit transmission. TMR controls are commonly used to ensure proper signaling and execution integrity. If a control line encounters a fault, a vote from the three redundant lines determines the correct feedback. After decoding, the next four stages follow standard router pipeline techniques providing virtual channel and
switch allocation, routing computation, and switch traversal. The last stage flits encounter is the encoder stage. In this stage, flits are encoded with my s2s encoder. The ECC that the flits are decoded with is signaled by the control lines as mentioned above, but from the downstream routers since flits are now leaving the current router. In the next section, we will provide an in-depth look at my encoder and decoder design for these stages.

2.3 Encoder and Decoder Design

In this section, we will discuss the motivation and design choices for my configurable s2s RAS encoder and decoder. The major challenge in implementing configurable encoders is the complexity required to implement them. The uniqueness and complexity of ECCs typically rules out the possibility of reusing the hardware. Therefore taking advantage of any modularity is a design goal when building configurable encoders. Performance and power trade-offs should also be minimized to reduce the impact of switching between modes, but it is important to realize the additional cost may be justified if the cost of boosting coverage by using a stronger ECC is less than the cost of rerouting flits and increasing hop counts to bypass it.

To implement an area and power efficient configurable s2s encoder we examine the previous work in multi-mode encoders such as the ones proposed in DUAL-Layer [30] and inherently modular encoders such as the ones proposed in joint crosstalk avoidance and triple-error correction (JTEC) [25] and Hsiao [24] encoders. In [30], the authors propose a DUAL-Layer method for transient and permanent faults with a two mode ECC encoder for transient faults and spare wires for permanent faults. Their ECC encoder however requires separate hardware for each mode and relies on interleaving to increase fault coverage. When operating in one mode, the hardware for the other mode is not being used. Interleaving is a common technique to improve the detection of burst errors, but the total number of faults detectable in each codeword is not increased. For example in DUAL-
Layer, if two faults occur in the same codeword, the faults will be undetected because each interleaved codeword is encoded with a single error correction (SEC) hamming code.

In Figure 2.3 the implementation of my s2s encoder is shown. The block diagrams on the left of Figure 2.3 show the joint crosstalk avoidance and triple-error correction quadruple-error detection (JTEC-QED) encoders that serve as the foundation of my configurable design. The JTEC encoder is implemented using a single hamming code encoder with the output duplicated with an additional overall parity bit. The duplicated
output in the resulting code helps to avoid crosstalk by placing similar signals in pairs. Pairing the signals reduces the chance of a wire from being completely surrounded with wires of the opposite polarity and therefore helps avoid crosstalk. The additional parity bit increases the total hamming distance of JTEC 7 allowing triple error correction. In [25], JTEC can be further expanded to JTEC-QED if the hamming encoder is swapped for an optimized Hsaio encoder. In my design, instead of duplicating the output from a single Hsaio encoder, we combine the output from two Hsaio H(39,32) SECDED encoders.
with a pair of 2 x 1 muxes, a switch to duplicate the output from a single encoder, an additional syndrome generator and a control block to manage the additional components and give us our three RAS modes. When in mode RAS-W, as shown in the top right inlet, flits are broken into two 32 bit segments and encoded separately through each H(39,32) Hsaio encoder. The switch that ties the output from the two encoders is turned off and the output from both encoded segments is interleaved to increase burst error protection. If we want to boost coverage and upgrade routers to RAS-S as shown in the middle right inlet, the data is serialized and only 32 bits of the 64 is encoded per cycle to split transmission. The 32 bits that enters, passes through both multiplexers and encoders that act in dual modular redundancy (DMR) to result in a JTEC-QED codeword. The additional syndrome generator is then enabled to compare the output from both H(39,32) encoders and ensure that faults do not manifest within the encoder itself. To further increase the internal fault coverage of routers in this mode, the syndrome generator is also used to compare the parity bits of flits as they entered the router, with the newly computed parity bits in the encoder as flits prepare to exit the router. Comparing the old parity bits allows us to detect faults injected from the previous stages in the router pipeline. Lastly, to offer a RAS-P mode, split transmission occurs just like RAS-S, but instead of both encoders executing in DMR, only one of the encoders is enabled. All remaining hardware in this mode is turned off to save power, except for the switch that allows the output from the active encoder to be duplicated. Duplication results in a JTEC-QED encoded codeword. In all modes, if faults are detected but cannot be corrected, the flit is dropped and s2s retransmission is immediately requested. However if a fault occurs within a router and is detected by the syndrome generator when in RAS-S, the fault cannot be corrected inside the encoder because the correcting hardware does not exist within the encoder. In this scenario e2e retransmission is requested. If faults manifest from one router to another in RAS-S, s2s retransmission can still be used.
In order to decode each type of ECC from my RAS encoder, we also had to design a configurable decoder to handle each RAS mode as shown in Figure 2.4. In this Figure all active components for each mode are shown, while turned off components and control signals are in grey. In all three modes both decoders are required to decode flits in JTEC-QED from RAS-S and RAS-P and 2 x H(39,32) from RAS-W. While RAS-P and RAS-S use similar hardware, the syndrome corrector for decoding a flit in JTEC-QED uses a different algorithm that relies on the syndrome from both decoded segments. When in RAS-W, the syndrome corrector is notified to correct the output from both Hsiao decoders separately. Lastly, to support the internal fault detection mechanism within routers in RAS-S, the existing parity bits are forwarded with the rest of the flit so they can be compared at the encoder. Due to the modularity of the JTEC and JTEC-QED encoders exploited to create my configurable designs proposed in RAS, my complete micro-architecture improves fault coverage and reduces area over non-reusable designs such as DUAL-Layer.

Now that the e2e and s2s layer of my proposed encoding scheme has been explained, the final section of this chapter will describe a proposed method to dynamically switch routers between each RAS mode in an effort to react to varying fault rates.

2.4 Dynamic Mode Selection

In the performance evaluation chapter we provide a first order analysis of RAS with static mode switching. For parts of the analysis involving static mode switching, the assumption is made that the most efficient mode is known. Making that assumption allows us to measure the performance and power characteristics of specific scenarios because accurately detecting when to switch modes is a separate task. Analysis of static mode switching additionally allows us to measure the upper bound for speedup and EDP. In this section, we propose a method for dynamically selecting the mode of each router in
an attempt to localize faults as they manifest and adjust fault coverage on demand using information tracked from my hybrid encoding scheme and s2s encoders.

Before we examine how to select the thresholds used to switch between modes, we must first revisit the state diagram discussed in the methodology section shown in Figure 2.1 and review the trade-offs of each s2s mode. In the methodology section we showed a basic state diagram in Figure 2.1 outlining the dynamic theme of my micro-architecture. Now that my entire architecture has been discussed we need to expand on that idea and produce a state diagram that accurately portrays all the complexities of each mode in a single state diagram. In Figure 2.5 the full state diagram for RAS is shown. Initially, all routers begin in the CRC-mode and do not encode flits with s2s ECC. Once a fault is detected within a packet on the e2e layer, a request for the retransmission is sent back to
Figure 2.6: Example of RAS dynamically switching between CRC (white), RAS-W (purple), and RAS-S (red).

the source router. As the request travels back to the source, each router the original packet traversed is immediately upgraded from CRC-mode to RAS-W. When faults are detected on the e2e layer the origin of the fault in NoC is unknown. However, the fault location is limited to the path the original packet traversed. We also do not know if the fault was transient or the beginning of a permanent fault. By upgrading routers immediately to boost fault coverage on the s2s layer, we aim to improve the localization of any future faults and prevent costly e2e retransmissions. To track faults as they manifest each router has a counter used to measure the fault rate. Every fault corrected or detected in a router is logged
in the counter. To account for faults detected on the e2e layer, we also update the counters for every router in original path of the faulty packet. However when faults are detected in the e2e layer, CRC encoding only informs us if a fault exists while the percentage of faults is unknown. Instead of incrementing counters by the number of faults in this scenario, we increment by a base value of three. Three was chosen because it is the minimal number of faults required to surpass the detection capability of the s2s layer. After a router upgrades to RAS-W, it will stay in that mode for the time interval $T_c$. $T_c$ is the number of decodes chosen before the counter is compared to the thresholds $t_1$ and $t_2$. If the counter value is less than $t_1$, the router downgrades to CRC-mode. If the counter is above $t_1$ but below $t_2$ it will stay in RAS-W. However if the counter exceeds $t_2$, the router boosts coverage and transitions to RAS-S. In the event of a permanent fault the router transitions to RAS-P to continue using its links instead of rerouting around the fault. An illustrated example of these transitions can be seen in Figure 2.6.

To quantify the counter thresholds $t_1$ and $t_2$ we use the following equation from [34] but with parameters values from my architecture.

$$Threshold = \frac{\lambda_{\text{avg}}(1 - (1 - \epsilon)^{w_f \times w_{\text{avg}}})}{T_c}$$  \hspace{1cm} (2.1)

where $\lambda_{\text{avg}}$ is the average injection rate, $\epsilon$ is the BER, $w_f$ is the number of bits in a flit and $w_{\text{avg}}$ is the average hop count. To solve for the threshold value in equation 2.1 we chose a $T_c$ of 1024, set $w_f$ to 64 bits and measured the average hop count from Splace-2/PARSEC benchmarks in a 8 x 8 mesh network to be 5.5 for each application. We then plotted the result of equation 2.1 expressed in bits as $\lambda_{\text{avg}}$ and $\epsilon$ were varied as shown in the bottom of Figure 2.7. The plot of equation 2.1 shows that as the BER is relatively low, only a few bits are needed to detect the BER as expected. As the BER increases, we need more bits to detect the BER as expected. The only remaining question is determining what BER is acceptable for each mode. To offer some conjecture on this
remaining challenge, we can look at the residual flit error rate (RFER) of each ECC and observe when the RFER begins to climb. The RFER as shown in top of Figure 2.7 is the probability a flit will contain an undetected fault and nullify the additional fault tolerance supplied in ECC scheme. Every undetected fault that evades detection further increases the probability of silent data corruption (SDC) and full system failure. Therefore the goal should be to boost coverage and switch to stronger ECC modes when the BER increases and causes the RFER in weaker ECCs to climb. Due to multiple variables that exist in equation 2.1 and the adverse performance and power penalties incurred by boosting coverage too early, we will evaluate my proposed dynamic design in the next chapter across a series of thresholds in effort to adequately adjust fault coverage on demand.
Figure 2.7: TOP: The residual flit error rate (RFER) of the compared networks as the bit error rate (BER) is increased. BOTTOM: Counter threshold needed to measure the BER for each injection rate.
3 Performance Evaluation

In this chapter, we evaluate my proposed multi-layered encoding scheme, the effectiveness of scrubbing packets and my s2s configurable encoder design in terms of power, area overhead, network speedup, and reliability. In order to compare and assess the effectiveness, we compare RAS with serveral static approaches and evaluate RAS with different thresholds to dynamically localize faults and boost fault coverage on demand.

3.1 Static Analysis

In the static mode analysis of RAS, we compare my proposed RAS techniques to baseline networks with equivalent single layer s2s ECC encoding schemes. Each scheme is evaluated on a typical 8 x 8 64-core mesh topology with a pair of unidirectional links connecting adjacent routers with a supply voltage of 1.0 V and 2 GHz clock. Each router has four VCs for each input port and four 79 bit (64 data, 15 ECC) wide buffer slots per VC. For e2e communication, all packets are assumed to be 256 bits. Before injection into the NoC, each packet is split into 4 equal flits consisting of 64 bits. Flits where 64 bits Traffic in each network is routed deterministically using fault tolerant up / down [31] routing to determine the location of fault and adapt routes when links fail. A failed link is a link that is unreliable for communication because BER in that link exceeds the fault tolerance of the ECC protecting it. In the event of a high BER or permanent faults causing a link failure, NoCs must upgrade to a stronger ECC or reroute packets around the link.

To highlight the effectiveness of hybrid encoding and the benefit of scrubbing flits as they traverse the network, we first compare RAS to static s2s encoding schemes. Second, we will compare my proposed s2s configurable encoder to DUAL-Layers’ [30] two mode ECC encoder and Ariadne [31] as the baseline with one ECC mode. For fair comparison with DUAL-Layer we assume an equivalent split transmission controller proposed in [30] to manage flit serialization in lower bandwidth modes and expanded links to match flit
Table 3.1: Simulation parameters [1] for obtaining Splash-2/PARSEC traces using the SIMICS application suite.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1/L2 coherence</td>
<td>MOESI</td>
</tr>
<tr>
<td>L2 cache size/assoc</td>
<td>4MB/16-way</td>
</tr>
<tr>
<td>L2 cache line size</td>
<td>64</td>
</tr>
<tr>
<td>L2 access latency (cycles)</td>
<td>4</td>
</tr>
<tr>
<td>L1 cache/assoc</td>
<td>64KB/4-way</td>
</tr>
<tr>
<td>L1 cache line size</td>
<td>64</td>
</tr>
<tr>
<td>L1 access latency (cycles)</td>
<td>2</td>
</tr>
<tr>
<td>Core Frequency (GHz)</td>
<td>5</td>
</tr>
<tr>
<td>Threads (core)</td>
<td>2</td>
</tr>
<tr>
<td>Issue policy</td>
<td>In-order</td>
</tr>
<tr>
<td>Memory Size (GB)</td>
<td>4</td>
</tr>
<tr>
<td>Memory Controllers</td>
<td>16</td>
</tr>
<tr>
<td>Memory Latency (cycle)</td>
<td>160</td>
</tr>
<tr>
<td>Directory latency (cycle)</td>
<td>80</td>
</tr>
</tbody>
</table>

widths. The configurable encoder in DUAL-Layer has two modes but requires separate hardware for each mode. In mode one (ECC1) flits are encoded with a single SEC ECC. In mode two (ECC2) flits are encoded in four interleaved SEC codes. After expanding flits widths for fair comparison, ECC1 became a H(71,63) SEC hamming encoder, and ECC2 expanded to a four way interleaved H(19,14) SEC hamming encoder.

For power and area analysis of each compared design, we used the Synopsys Design Compile tool using the TSMC 45 nm technology libraries with a 1.0 V supply voltage and
2 GHz clock. To simulate real network traffic, benchmarks were collected from PARSEC traces using the SIMICS simulator using the simulation parameters shown in table 3.1 from [1] for the following applications: blackscholes, facesim, ferret, fft, fluidanimate, freqmine, streamcluster, and swaptions. To model behavior-level faults in the network, we used the fault model proposed in [3] to inject faults in every link and router. As the fault rates are low in the proposed model, we further extrapolated fault rates for high fault injection in my evaluation. For every fault successfully detected on a link, s2s ECC attempts to correct the fault in the decode stage. If correction fails the flit is corrupt and TMR control lines must request immediate retransmission. If a fault is not detected on the s2s layer when it is injected, correcting the fault is no longer possible. If e2e encoding is in place such as CRC-32 in RAS, the fault may still be detected, but will require retransmission. When a fault completely evades s2s and e2e encoding, a 200 cycle penalty was assumed before hardware higher up in the network stack could signal for retransmission. In the following subsections evaluation of network speedup, power, area, and reliability will be discussed.

3.1.1 Network Speedup

As MPSoCs continue to scale, the average hop count in NoCs will continue to rise. With each technology generation, interconnects are becoming more susceptible to crosstalk, process variation and soft errors. The consequence of providing inadequate fault tolerance will significantly impact overall network performance more as core counts increase, resulting in additional retransmissions and increased chances of silent data corruption. In s2s schemes, since encoders and decoders are placed in every router, the typical ECC implemented in this type of scheme is relatively weak, in the effort to keep area overhead cost down. The risk as the BER climbs is that purely static s2s schemes will not provide sufficient coverage. To highlight the advantage of a hybrid scheme and the network speedup gained by scrubbing flits on the s2s layer, we test each s2s encoding mode of RAS
with and without e2e encoding as the flit error rate was increased in Figure 3.1. In Figure 3.1 each RAS mode with e2e encoding is shown in solid colors while each corresponding mode without e2e encoding is shown as the same color with stripes. RAS-S, RAS-W and RAS-P are equivalent to SECDED, JTEC+FW, and JTEC respectively, without CRC-32 e2e encoding. In every case, each networks with e2e and s2s encoding outperformed its counter part without e2e encoding. As the flit error rate was increased in each mode, with e2e scrubbing the improved speedup over techniques that were implemented without the encoding. The baseline network was chosen as the SECDED network because it is a very common ECC implemented in NoCs. In low flit error rates the cost of splitting retransmission in RAS-S, RAS-P, JTEC and JTEC+FW can be observed as they all perform
Figure 3.2: Network speedup as link failures increase. As links fail, DUAL-Layer and RAS boost fault coverage to continue using the unreliable links while Ariadne must reroute packets around them.

below the baseline. The extra cycle penalty on all transmissions however is overcome as the flit error rate increases beyond 5% for most cases because each network with split transmission encodes with a stronger ECC. RAS-S which encodes flits in JTEC+FW outperforms RAS-P in every instance because RAS-S has no mechanisms in place to detect intra-router faults, therefore causing costly e2e retransmissions penalties for every intra-router fault. On average the significance of combining e2e and s2s encoding to prevent faults from accumulating on the e2e layer improves network speedup by 10%.

Now that we have shown scrubbing flits on the s2s layer can improve network speedup in a hybrid encoding scheme, we also wanted to compare the advantages of employing configurable s2s encoders to highlight the trade-off between boosting fault coverage in faulty links versus the cost of rerouting around them. To evaluate the effect these trade-
offs had on network speedup we compared RAS, DUAL-Layer [30] and Ariande [31] as the baseline, as the amount of failed unidirectional links was increased in Figure 3.2. For the evaluation setup, we assumed each network initially started with s2s encoding enabled. In RAS, all routers were in RAS-W, ECC1 for DUAL-Layer, and a standard SECDED ECC for Ariadne. When a unidirectional link fails and s2s encoding can not be increased, the adjacent link in the opposite direction is also considered unavailable and both must be rerouted around. To reroute traffic around failed links, fault tolerant up / down routing is used to adapt routing tables and prevent deadlocks scenarios that would normally be caused in an evolving network topology. A deadlock is a severe network condition capable of halting all network traffic when contention builds up enough back
pressure to stop all flit progression. When a link failure occurs in either RAS or DUAL-layer, both networks boost coverage by upgrading modes to RAS-S and ECC2 respectively. Since both RAS and DUAL-Layer split transmission to encode packets with a stronger ECC, the speedup is the same, however both networks will differ in power consumption and reliability which we will show later in the section. In Figure 3.2, the results show that as the network experiences a low percentage of link failures between 0%-5%, overall network speedup for each network is about the same. In a 8 x 8 mesh, 224 unidirectional links exist. With 5% link failures an 8 x 8 mesh network with 224 unidirectional links, we observe that we will have at most 11-12 links failures. As long as the link failures are randomly distributed throughout the network and not creating subnetworks, both methods, rerouting and boosting coverage, seem to balance each other and provide the same speedup. Subnetworks are formed when routers can no longer communicate with every router caused by link failures. As the number of link failures increases, the amount of subnetworks created is shown in the bottom of Figure 3.3. It is also important to note how many routers are in each mode. In the top of Figure 3.3 we show how many routers are in RAS-S and RAS-W. Although we only show modes for the RAS network, the same number of modes in RAS-S and RAS-W would be equivalent to DUAL-Layer’s ECC1 and ECC2 respectively. When link failures reach 10% RAS and DUAL-Layer begin to show network speedup improvements over Ariande which must reroute packets. The performance fluctuations between each application can be explained by the differences in traffic patterns. The traffic pattern for each application is different, but the link failures affect the same links in the network for each case.

That concludes the network speedup analysis for my first order analysis of RAS using static mode switching. Further evaluation of speedup will be examined in dynamic mode section of this chapter.
3.1.2 Area

When comparing fault tolerant networks, evaluating the area overhead is an important metric to examine because additional fault coverage typically implies more hardware. By designing area efficient fault tolerant techniques we can minimize the total area footprint of the NoC. One of the primary contributions in my research is my 3-mode configurable s2s encoder. In Figure 3.4 we compare total router area cost with the following encoders: RAS, DUAL-Layer, s2s-JTEC, cross-disjoint (cdd) JTEC and e2e. Even though we assume the CRC-32 (256,224,32) encoder for e2e encoding is implemented within the network interface of each core, for fair comparison we show the cost as if the e2e encoder was in each router. Figure 3.4 shows that even though DUAL-Layer has two ECC modes it requires the most area of the five and 15% more than RAS, followed by cdd-JTEC, s2s-JTEC and e2e. If the area cost of the e2e encoder was included with RAS, RAS would
still be 1% more area efficient and offer more reliability than DUAL-Layer which will be discussed later in this section.

3.1.3 Power

Thus far we have shown RAS improves overall network speedup and reduces area overhead over traditional s2s schemes. In this subsection, we evaluate how my design stacks up when it comes to overall power consumption. Since additional hardware is generally required to increase fault coverage and improve network resiliency, additional power consumption is not uncommon, but as NoCs consume more overall chip power, researchers should continue to strive for power efficient designs. In Figure 3.6 the total power (mW) consumed to send a single flit through each stage of the router pipeline and the next adjacent router is shown for each s2s encoder. For the RAS and DUAL-
Layer configurable encoders, the power consumption was broken down further to show the differences of each mode. In Figure 3.6 cdd-JTEC consumed the most power even though cdd-JTEC is a single mode encoder, however in cdd schemes, twice the number of encoders and decoders are used in each router. The reasons for additional encoders and decoders in cdd is to provide intra-router fault coverage. After cdd-JTEC, RAS-W and RAS-S consume the most power. Of the three RAS modes, RAS-W and RAS-S consume the most because nearly all components in each encoder and decoder are active. Compared to single s2s schemes such as s2s-JTEC, each RAS mode consumes more power due to the additional hardware in my design. Despite RAS-P consuming less power than DUAL-Layer’s ECC1 and ECC2, on average RAS consumes 6.6% more power than DUAL-Layer, but we save 15% in area overhead and employ stronger ECCs.
Now that we are aware of the power penalty induced by each type of s2s encoding, we can evaluate how those penalties translate into overall network power consumption. To compare the energy tradeoffs of each scheme, we measured the energy delay-product (EDP) shown in Figure 3.5 with the same setup as described in the previous speedup section. EDP is a valuable metric used to compare the power efficiency of a network because it correlates power consumption with latency into a single product. As link failures are increased, networks with configurable encoders boost coverage to continue using the faulty links, while networks such as Ariadne must reroute packets around them. Results displayed in Figure 3.5 allow us to measure the tradeoffs of each approach. Every packet that does not take a minimal route in Ariadne, increases the average packet latency and overall power consumption with each additional hop. In RAS and DUAL-Layer, split transmission is traded for additional fault coverage to sustain the usage of those links. When routers are upgraded, every packet incurs an additional cycle delay for each flit, and the power penalty to do so. For low link failures the configurable schemes such as RAS and DUAL-Layer produce similar EDP to Ariadne. This same pattern was also seen in Figure 3.2 when comparing network speedup. In some cases producing a higher EDP because when routers upgrade to modes with split transmission, all packets traversing the router are affected. In Ariadne, only the packets forced to reroute cause additional latency and power consumption. However, similar to Figure 3.2, as the percentage of link failures increases beyond 10%, the advantage of boosting coverage exceeds the cost of rerouting. This can be explained by the formations of subnetworks and increased dissection of the network where there is as little as 10% link failures. As NoCs continue to scale to support increasing core counts, the more beneficial fault tolerant NoCs will be with adaptive schemes over static schemes. Although RAS on average produces a 6.6% higher EDP than DUAL-Layer, my s2s configurable encoder requires less area overhead and offers higher network resiliency.
3.1.4 Reliability

To compare the reliability of RAS to DUAL-Layer and commonly used ECCs, we calculate the residual flit error rate (RFER) of each ECC implemented in the compared networks. The RFER as shown in Figure 3.7 is the probability a flit will contain an undetected fault and nullify the additional fault tolerance supplied by the chosen ECC. Measuring the RFER of each ECC allows us to compare the strength of the code used and delineate the overall reliability of each fault tolerant architecture. As the bit error rate (BER) increases, the RFER of every ECC eventually saturates, except for CRC-32. CRC-32 does not saturate like the rest because beyond the small bit and burst errors detectable in CRC-32, it can also detect all odd bit errors, which places an upper bound on the RFER from exceeding 50%. In Figure 3.7, ECC1 and ECC2 of the DUAL-Layer architecture are the first two modes to completely saturate after a flit with no ECC. After which, RAS-W and RAS-S follows respectively. The RFER of ECC1 and ECC2 increases before each mode of RAS because DUAL-Layer employs the weakest ECCs with only single error correction (SEC) and 4-way interleaved SEC. RAS-W on the other hand encodes flits with 2-way interleaved single-error correction double-error detection (SECDED). The RFER of RAS-S saturates last among each s2s flit encoding ECC because it employs the strongest ECC with joint crosstalk avoidance and triple-error correction, quadruple error detection (JTEC-QED). As mentioned previously in the micro-architecture chapter, the RFER offers insight on when to dynamically switch modes from one ECC to another. It would be desirable for configurable architectures to switch modes as the BER increases to reduce the increase of the RFER in the network.
Figure 3.7: We show how the BER affects the RFER of each ECC. By examining their RFER we can measure the fault tolerance of each scheme.

### 3.2 Dynamic Analysis

In this section, we evaluate my proposed model for dynamically adjusting ECC coverage in RAS on demand using my s2s configurable encoder. A major challenge with adaptive ECC schemes is predicting when to upgrade coverage. Boosting fault tolerance too early can consume excess power and cause unneeded performance penalties when fault rates are low. Failing to adequately upgrade coverage when fault rates increase, raises the risk of full system failure and the chance of silent data corruption. Additionally, while one prediction model may work for one adaptive scheme, there is no guarantee the same model will work for another because each scheme has the possibility to exhibit different trade offs for each mode. For example, in RAS-W, while additional power will be consumed after switching from CRC-mode, upgrading will have no negative effects on
Table 3.2: Test cases for evaluating my proposed dynamic switching model.

<table>
<thead>
<tr>
<th>Case</th>
<th>Threshold 1 (t1)</th>
<th>Threshold 2 (t2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>9</td>
</tr>
</tbody>
</table>

For the experimental setup of my dynamic model, five routers in an 8 x 8 mesh were randomly selected to inject faults on every outgoing link as the BER was increased. In Figure 2.7, we associate both the threshold value and RFER with the BER and hypothesized that a RFER of 5% is unacceptable. By correlating the two graphs in Figure 2.7 we chose a series of test cases shown in table 3.2. To simulate real network traffic, benchmarks were collected from PARSEC traces using the SIMICS simulator with the simulation parameters previously shown in Table 3.1 for the following applications: blackscholes, facesim, ferret, fft, fluidanimate, freqmine, streamcluster, and swaptions. When faults are injected, if they are successfully detected on the s2s layer, decoders handle fault mitigation and log each occurrence. If a packet reaches its destination containing faults, e2e retransmission is requested if the fault is detected during the CRC decoding stage. If CRC is unable to detect, the fault we assume a 200 cycle delay before retransmission is requested higher up in the network stack. In the following subsections we will evaluate how thresholds affect network speedup, EDP, mode transitions, decoding and retransmissions.
Figure 3.8: In this figure we show how we chose $t_1$ and $t_2$ for each test case in Table 3.2. The BER used to test each case were chosen before, in between and after each vertical line. More details on the evaluation setup for will be discussed in my thesis defense.
3.2.1 Network Speedup and EDP

In Figure 3.9 we show the network speedup and EDP for each test case as the BER is increased. In order to evaluate an upper bound on the first order analysis of my dynamic model, all test cases for network speedup and EDP are normalized to a fault free network. As the BER is increased, a gradual decline in network speedup is observed for all test cases but does not exceed a decline larger than 2.12%. As the BER does increase test cases with higher t2 thresholds perform slightly better than those with lower t2 thresholds. For low BER, test case 5 performs significantly worse for some applications because it is the only case where t1 is 6 bits instead of 4. With a high t1 threshold, routers will transition back
to CRC mode more often and fail to detect injected faults. Although each fault missed will more than likely be detected on the e2e layer, the amount of retransmissions will degrade the speedup. This also causes a cascading effect because as routers turn off s2s encoding, fewer faults will be logged in the counters, resulting in inaccurate measurements of the BER. Inaccurately low measurements of the BER will prevent routers from switching to the most efficient mode.

For EDP in the bottom of Figure 3.9, a much larger penalty is observed as the BER increases. In scenarios with the highest BER, EDP increases to a maximum of nearly 50% more compared to the baseline of a fault free network but on average 20%. As mentioned above, for test case 5, with the higher t1 threshold, EDP increases much faster than cases where t1 is lower. As the BER is increased, test cases with lower t2 thresholds increase the EDP faster than those with larger t2 thresholds. This can be explained by the power consumption trade off between RAS-W and RAS-S. When routers are in RAS-S, all incoming traffic is split to augment the stronger ECC used in this mode. Split transmission not only creates an extra cycle delay for each flit, but doubles the link traversal cost. For large traffic traces, the additional cycle delay caused by split transmission can be hidden in overall speedup, but the power penalty as shown in Figure 3.9 will be much more pronounced when EDP is measured.

### 3.2.2 Mode Transitions

To try to gain further insight on how each test case affects the architecture in the background, we tracked the number of times each router transitioned from one mode to the next. In Figure 3.10, we show the total number of transitions that occurred in the network for the Blacksholes application and break it down by mode. In all test cases, except for test case 5, a low BER caused much more mode transitions than when the BER is increased. In test case 5, t1 is larger, therefore routers are more likely to transition back to CRC-mode.
from RAS-W because it is less than likely t1 is reached. By observing the results in Figure 3.10, and the difference shown between test case 5 from the rest, it may be more beneficial to use a t1 lower than 4 bits, which was the lowest t1 evaluated. Doing so, could minimize the amount of transitions for lower BERs. Lastly, when the BER reaches $10^{-02}$, we see the number of transitions rapidly increase. The increase in transitions in this scenario is because the BER of this magnitude is so high, the ECC in RAS-S is failing to detect faults on the s2s layer. When they are detected on the e2e layer, every router in the path the packet took is then upgraded to RAS-W if it is in CRC-mode, causing a lot of routers to toggle back and forth. If we recall Figure 3.7, a BER of $10^{-02}$, causes the JTEC-QED ECC employed by RAS-S, to fail at detecting faults, which supports this claim.

### 3.2.3 Decoding and Retransmissions

Lastly, to observe any remaining differences between test cases evaluated in my dynamic switching model, Figure 3.11 shows the percentage of flits encoded and decoded
in each mode. As expected in Figure 3.11, the majority of flits were encoded by routers in RAS-W. As the BER was increased, more routers upgraded to RAS-S and the percentage of flits encoded with RAS-S rose. For test cases with larger t2 thresholds, the percentage of flits encoded in RAS-S increased at a slower rate.

Figure 3.12 shows the total number of packets with errors at the destination (corrupt packets). If routers are upgraded to the appropriate mode, providing adequate fault coverage, we should see the total number corrupt packets decrease. As mentioned above, a t1 of four bits is potentially too high for low BERs. A smaller t1 threshold should capture the BERs and reduce the amount of e2e retransmissions. In the previous section we claimed that the BER of $10^{-02}$ was also the cause for the spike observed in mode transitions shown in Figure 3.10. The same spike in the number of corrupted packets is seen and further supports that claim.
Figure 3.11: As the BER was increased we tracked how each flit was decoded for each case.

Figure 3.12: As the BER was increased the number of retransmissions in each case was logged.
4 CONCLUSIONS

In this thesis we proposed Runtime Adaptive Scrubbing (RAS) an adaptive hybrid (e2e + s2s) ECC encoding scheme that tunes fault coverage on demand and permits graceful network degradation as fault rates fluctuate in links. Simulation results showed that by scrubbing flits on the s2s layer with my configurable encoder and encoding packets on the e2e layer, network speedup improved on average by 10% over single layer schemes. We then evaluated whether boosting coverage in faulty links with additional hardware would improve network speedup and EDP instead of rerouting packets around them. We found that as 10% of links failed in a 8 x 8 mesh NoC, adaptive ECC schemes significantly reduced EDP and improved overall network speedup over static schemes. When link failures were below 10%, the cost of boosting coverage and rerouting packets roughly matched in terms of speedup and EDP for most benchmark applications. Since both RAS and DUAL-Layer augmented the cost of boosted fault tolerance by splitting transmission, network speedup for both architectures were equivalent. Although network speedup gains were measured the same, RAS offers 3-modes RAS-W, RAS-S and RAS-P compared to ECC1 and ECC2 of DUAL-Layer. When we compared the area overhead of my hybrid scheme to DUAL-Layer and popular s2s encoding methods we showed, RAS was found to cost more than single mode s2s architectures but reduced area overhead by 15% to that of DUAL-Layer. In terms of power consumption, RAS did incur on average a 6.6% power penalty over DUAL-Layer because most components in my configurable encoder are active in each mode. RAS does however implement stronger ECCs than DUAL-Layer. To summarize, while my hybrid encoding scheme does consume 6.6% more overall power than DUAL-Layer, RAS reduces area overhead by 15%, and RAS employs stronger ECCs in each of its 3-modes, improving overall resiliency.
REFERENCES


