UNIVERSITY OF CALIFORNIA,
IRVINE

Reliability Enhancement of Many-core Processors

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

MASTER OF SCIENCE

in Electrical and Computer Engineering

by

Maryam SeyyedHosseini

2017

DEDICATION

To my beloved husband and best friend,
Pooria,
For his boundless love, timeless care and support.
To my Mom and Dad,
For their endless love, support and encouragement.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

First of all, I would like to thank my advisor, Professor Nader Bagherzadeh, for his continuous support , patience, motivation, and immense knowledge throughout my study and research. He was always present to direct me and encourage me in all the time of research and writing of this thesis. I could not have imagined having a better advisor for my master study.

I also would like to express my gratitude to Professor Jean-Luc Gaudiot, and Professor Philip Sheu for serving as my committee members.

CURRICULUM VITAE

Maryam SeyyedHosseini

EDUCATION

M.S. in Computer System and Software **March. 2017**
University of California, Irvine *Irvine, California*

B.S. in Computer Engineering **Sep. 2008**
Ferdowsi University of Mashhad *Mashhad, Iran*

TEACHING EXPERIENCE

Teaching Assistant **Sep.2016 – Present**
University of California, Irvine *Irvine, California*

RESEARCH INTERESTS

Reliable system design, Computer architecture, many-core architectures, High performance computing.

AWARDS

Ph.D. Bridge Fellowship **Sep. 2016 – Jun. 2017**
University of California, Irvine *Irvine, California*

ABSTRACT OF THE DISSERTATION

Reliability Enhancement of Many-core Processors

By

Maryam SeyyedHosseini

Master of Science in Electrical and Computer Engineering

University of California, Irvine, 2017

Professor Nader Bagherzadeh, Chair

Many-core systems are of great importance for building the exascale computing machine targeted for 2020. Last-Level Cache (LLC), as the largest on-chip shared memory in many-core systems, plays a crucial role in power, area, and more important in reliability. Reliability in LLC depends on both distributed banks and the communication fabric (Network-on-Chip (NoC) interconnect). In order to achieve high reliability factor, they both need to be protected against errors. Existent error coding methods protect the cache and communication fabric, but in isolation of each other. Based on the observations in this thesis, when cache and NoC interconnect are considered together, the delay overhead of LLC protection has been decreased. In this thesis, the main contribution is NARC , an integrated method that minimize the delay overhead of error protection in many-core architectures by integrating the error coding of cache and interconnection network. This new approach sets up a linked error coding scheme that guarantees the end-to-end protection of shared cache data blocks throughout the on-chip network against both hard and soft errors. NARC partitions each shared cache block into multiple equally-sized segments. It extends each segment with a low-cost ECC, and transmits each extended segment as a flit in NoC. NARC eliminates the large ECC encoder/decoder blocks from the critical path of shared cache remote access through the network. Using this technique, NARC minimizes latency in the common case of accessing a shared LLC bank over the network, and potentially accessing a local LLC bank, while providing almost the same error protection as strong multi-bit ECC in cache blocks using a segmented per flit ECC. It has been evaluated that on a 6 by 6 platform with mesh NoC, NARC improves the performance of the many-core systems on average of 9.6% about and it can go up to 22% compared to a baseline approach.

Chapter 1

Introduction

1.1 Background

Every major shift in computer architecture has led to dramatic changes, and the move to exascale computing will be no exception. At the hardware level, feature size in silicon will almost continue to decrease at the Moore's law pace by the end of this decade. To remain effective in high-end computing systems and in consumer electronics, computer chips must change in radical ways. Exascale computer systems are needed for the growing number of problems where experiments are impossible, dangerous, or expensive. These machines, along with parallel computing, will enable the analysis, modeling and processing of massive amount of data which leads to advances in various areas of science and technology.

Exascale computing is challenging due to the strict constraints in power requirements, requiring new communication infrastructure and software approaches to exploit parallelism and scalability. Based on the current technology, scaling today's systems to an exaflop level would consume more than a gigawatt of power [43]. Reducing the power requirement by a factor of at least 100 is a challenge for both future hardware and software technologies.

Today's top supercomputer systems include approximately 0.5 to 3 million cores [1]. By 2020, due to design and power constraints, the clock frequency is unlikely to change, which means that a HPC system will have approximately one billion cores [39][15]. An immediate consequence is that the power consumption and hence temperature will increase while timely power and thermal management become much more difficult. Mathematical models, numerical methods, and software implementations will all need new conceptual and programming paradigms to make effective use of unprecedented levels of concurrency.

1.2 Why Many-core systems

Power wall, memory wall, and ILP wall which were the most big problem in single core systems bold the need for multi-core systems. Although these limitations were no longer in multi-core systems, there are still some limitations such as imperfect scaling, difficulty in software optimization, and maintaining concurrency over a number of cores for multi-core processors. These limitations in multi-core systems led to the need for many-core systems.

Many-core platforms have become mainstream for the many-core systems where Last-Level Cache (LLC) shared memory architecture supports low-level communication among cores. Last-level caches in many-core systems are formed of multiple distributed memory banks which are linked by a sophisticated interconnection infrastructure, often a Network-on-Chip (NoC). Since NoC has become the foundation of many cores' interconnection network, reliability becomes a major design challenge by increasing the complexity of these platforms. An important increase in both permanent faults and transient errors is expected due to advanced technology nodes, higher integration, voltage scaling, parametric variations, and higher power density that endanger the reliability of many-core platforms [42] [66]. Hence, both cache banks and interconnection fabric (NoC) in shared cache extremely suffer from possible fault occurrences. Though researchers have extensively explored techniques to protect data in caches, most of the proposed solutions ignore interconnects or assume a simplified case in which data transmission among cores are reliably done through the interconnections. Error correcting code (ECC) approaches are very common in both cache memories and communication/links in NoCs independently to protect cache banks and interconnection fabric against errors, respectively [32][8]. In cache memories, Error correcting Code (ECC) schemes can be as simple as a single-bit error-correcting (SECDED) or as complex as strong multi-bit ECCs [32]. Such approaches suffer from inefficiencies such as high cost and lack of scalability when applied to large shared caches in many-core systems. In case of NoCs, ECC schemes can be implemented in the switch-to-switch (s2s) and end-to-end (e2e) levels [41][47][53]. The choice of either an e2e or s2s ECC scheme in NoCs has power, area, and performance tradeoffs that vary with the fault rate, size, and topology of the system. However, due to limited level of error protection, an e2e scheme is inefficient in emerging many-core systems with high error rates. We can observe from all related work that there is no previous effort that consider reliability of both cache and interconnection fabric together. However, in many-core architecture with distributed shared caches that a large amount of memory accesses are passed through the network, it is necessary to protect them against errors in a unified manner. One simple solution is directly integrating conventional error protection schemes available for both cache and NoC interconnect to protect them together. However, since such schemes are designed in isolation of each other, this approach would be costly and inefficient. Consequently, a unified system-wide approach that guarantees the e2e protection of shared cache blocks without imposing a conspicuous overhead is essential for future many-core architectures.

In this thesis, the main contribution is NARC , an integrated method that minimize the delay overhead of error protection in many-core architectures by integrating the error coding of cache and interconnection network. This new approach sets up a linked error coding scheme that guarantees the end-to-end protection of shared cache data blocks throughout the on-chip network against both hard and soft errors. NARC partitions each shared cache block into multiple equally-sized segments. It extends each segment with a low-cost ECC, and transmits each extended segment as a flit in NoC. NARC eliminates the large ECC encoder/decoder blocks from the critical path of shared cache remote access through the network. Using this technique, NARC minimizes latency in the common case of accessing a shared LLC bank over the network, and potentially accessing a local LLC bank, while providing almost the same error protection as strong multi-bit ECC in cache blocks using a segmented per flit ECC. It has been evaluated that on a 6 by 6 platform with mesh NoC, NARC improves the performance of the many-core systems on average of 9.6% about and it can go up to 22% compared to a baseline approach.

Chapter 2

Background and Related Work

2.1 NoC Fundamentals

2.1.1 Network-on-Chip Systems

With the continuous rising developments of semiconductor fabrication technology, incorporating additional transistors within a single chip turned to be visibly achievable. The idea of Network-on-chip (NoC) has been brought by Benini, which provides better communication capabilities and performance for the System-on-Chip (SoC). Network-on-Chip is a technology that supports a new scheme for communication in the system-on-chip and it provides better performance measures [45]. Thanks to this fast growth of the technology of semiconductor integration, the complexity of a uni-processor chip grows as the Moore's law defines, while the growth of the capability of memory system increases so little. This leads to a huge gap between the processors and memories called memory gap. So, improving the capability of memory systems tries to close this gap [7].

Network-on-chip (NoC) is an interconnect network for SoCs, MPSoC, and FPGAs and it has become an important resource in the Chip Multiprocessors (CMPs). NoC is a scalable communication infrastructure and its main objective is to provide a scalable efficiency/performance in hardware area and power. Figure 2.1 shows basic components of a 2D mesh NoC, where each light gray tile represent a core, and to have those cores communicate with each other, we use the routers, signified by the dark gray tiles, for each core. Communication happens through the physical links and directed by the routers.

The baseline NoC router, utilized in this thesis, is designed with five bidirectional ports, to support mesh and torus topologies. It is composed of three main components including an input buffer, a management and routing unit, and a crossbar switch [21, 66, 6, 61, 3], and error handling (Hamming).

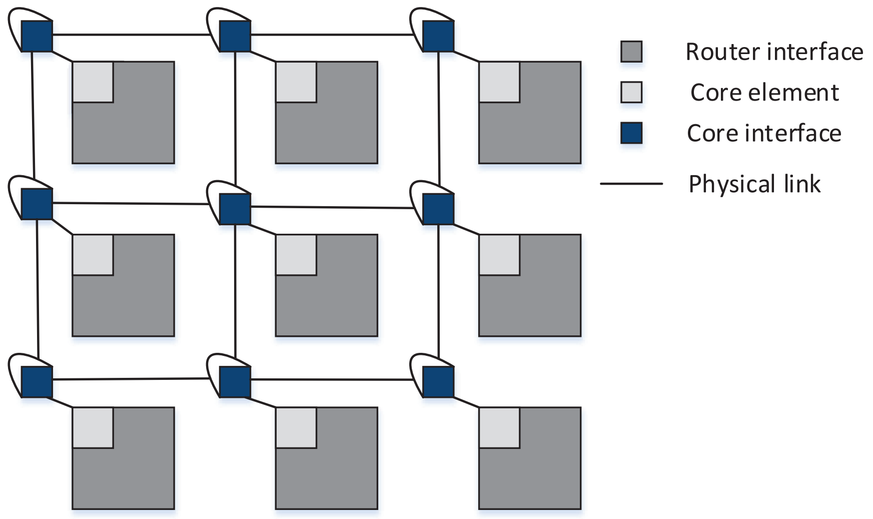

Figure 2.1: Mesh-based NoC

2.2 Related Work

This work represents a convergence of three main bodies of related research: fault-tolerant cache design, fault-tolerant NoC interconnect, and memory reliability in multi/many-core architectures.

Fault-tolerant Cache Design: There is a rich body of literature on design for error resiliency in cache memories. Earlier experiments on fault-tolerant cache design suggested using redundant rows/columns to the cache [56] or applying cache down-sizing techniques by disabling a faulty line or block of cache [44]. Some of them use circuit-level techniques to improve the reliability of each SRAM cell. Apart from the familiar 6T SRAM cell, 8T SRAM cell and 10T SRAM cell [13] have been proposed. Most of these designs have a large area overhead which poses a significant limitation for performance and power consumption of caches. At the system level, a wide range of EDC and ECC have been used [14]. ECC is proven as an effective mechanism for handling soft errors [58]. However, in a high-failure rate situation, most coding schemes are not practical because of the strict bound on the number of tolerable faults in each protected data chunk. In addition, using ECC codes incurs a high overhead in terms of storage for the correction code, large latency, slow and complex decoding [30]. There is a recent approach developed by [25] as a fault-tolerant cache coherence protocols.

Several architectural techniques have also been proposed to improve reliability of on-chip caches by employing relatively sophisticated fault tolerance mechanisms, such as block/set pairing [2, 60, 51], address remapping [5] and so on. In sum, since most of these techniques are designed for a single-core processor with one or two cache banks and uniform cache access designs, none of them considers the effects of a faulty interconnect on their approach, unlike our proposed approach.

Fault-tolerant NoC Interconnect: Network-on-Chip communication might be subject to errors. Reliability and fault-tolerance are outstanding research challenges in NoC design [36]. There are many efforts to investigate the robustness of NoCs, mostly in the areas of routing algorithms [50, 49], communication infrastructure [12, 40, 46, 54], or micro-architecture [33, 38]. A fault-aware IP-core mapping to NoC routers is proposed in [35]. They address the problem of transient link failures by means of temporally and spatially redundant transmission of messages. A stochastic communication paradigm is proposed in [12] to provide a fault-tolerant communication. This approach is based on probabilistic broadcast where packets are forwarded randomly to the neighboring nodes. However, none of them protect faulty datapath inside routers. In [9], the authors deploy some error correcting schemes to achieve combined energy minimization and reliability optimization design goals. A fault-tolerant router architecture is proposed in [22] to guarantee the functionality of the NoC in the presence of faults. They detect the most tenuous components of router against different sources of faults first and then tolerate them to save power and area overhead of employing fault-tolerant techniques. An adaptive fault-tolerant NoC router with the power consumption consideration is proposed in [38]. They improve error correction capability of their design based on the frequency of fault occurrence. Furthermore, all of the related experiments tolerate failures inside the NoC, ignoring any failure of their memory/cache components; in this thesis, using the consolidation of those approaches it is tried to address this issue.

There is no ending for demanding the fault-tolerant design feature which is still one the major concern of future technologies [62, 23, 17, 64, 63, 16, 19, 31, 67, 68, 18].

Memory Reliability in Multi/Many-cores: There are some recent efforts addressing memory resiliency in NoC-based CMP architectures. [4] presented a mechanism to maintain a reliable integrated memory subsystem for a NoC-based system. The idea is to have a reliable backup memory (in addition to the main memory) to store all the critical data. The NIs are responsible for redirecting the critical data access in case of a failure in the main memory. Although, this scenario provides some percentage of reliability to the memory, it has drawbacks like redundancy of data and not being transparent to the programmer. In another work [57], Wang addresses fault-tolerance of NUCA cache in NoC-based CMPs. They proposed a utility-driven address remapping technique to tackle the capacity loss in NUCA cache of NoC-based CMP architectures. However, their address remapping technique is at the bank-level, and they have not leveraged the NoC fabric for fault-tolerance of the cache. In a work done in 2010 by [59], they proposed Hi-ECC technique that incorporates multi-bit error-correcting codes on eDRAM caches to reduce refresh time and cache power consumption. However, their approach achieves a significant reduction in refresh power in cache, they have not considered SRAM caches in this approach.

Some recent work by [69, 65, 20, 24] use qualitative techniques to addresses resiliency of NoC routers.

Proposed approach is different from all the related work as follows. Unlike all previous efforts in fault-tolerant cache design that consider an error-free and reliable interconnect, proposed approach considers errors of both cache and interconnects simultaneously. Also, the proposed is the first work that proposes an integrated approach to error coding of cache and interconnect in large interconnected many-core platforms to minimize the error protection costs.

Chapter 3

Proposed Architecture

This chapter explains the NARC approach idea and architecture in detail and discusses the possible questions and concerns to better elaborate and justifies the decisions made in this thesis.

3.1 Background and Motivation

Many-core systems usually use large last-level shared caches which are shared and NUCA among all cores and access latency of LLCs has a considerable effect on the cost of system [29]. Figure 3.1 shows the percentage of the total number of local cache accesses in a 6×6 platform with shared LLC running PARSEC benchmarks. As demonstrated in this figure, remote accesses to LLC banks through the interconnect are much higher than the local ones. The remote cache access number for each workload would be the subtraction of local access rate from 100%. Having calculated the local versus remote cache accesses, the latency contributions of the ECC logic to the overall shared cache access latency can be estimated.

Also, Figure 3.11 clarifies the fact that in such platforms most of the cache block accesses

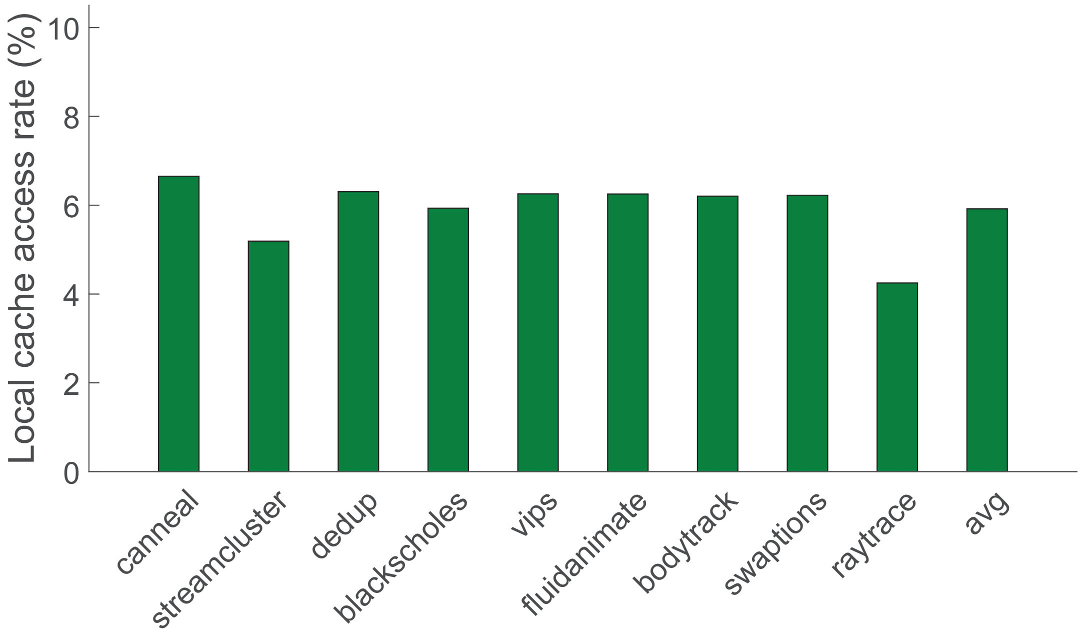

Figure 3.1: Percentage of the total number of local cache accesses in a 6×6 platform with shared LLC running PARSEC benchmarks. The remote cache access number for each workload would be the subtraction of local access rate from 100%.

are remote and through the interconnect. It can be observed that a large amount of remote shared cache access latency is imposed by LLC cache ECC logic; especially if the requesting core is closer to the target LLC cache. The observation from both Figure 3.1 and Figure 3.11 inspires that there is a high chance of improving the performance of the shared-memory systems by reducing the latency of error handling architecture of shared caches which is in the critical path of LLC accesses.

3.1.1 Error Correction Coding Overview

This section discusses the various aspects of coding methods as the dominant fault tolerant approach being used in many-core systems.

Fault tolerant approaches leveraging error correcting codes can protect systems from errors which have a significant effect on system reliability. A common solution for addressing errors in memory components is applying Error Detecting/Correcting Codes (EDC/ECC). Typically, EDC techniques are composed of parity bits, while the most common ECC methods employ Hamming [27] or Hsiao [28] codes with the capability of Single bit Error Correction and Double bit Error Detection (SECDED). More complex codes such as DECTED, QECPED (Quad (4-bit) Error Corrections and Panta (5-bit) Error Detection) [30] and also BCH [48] are considered when higher error detection is needed. However, they are merely used in large cache memories, since the area, delay, and power overheads of ECC grow drastically as the correction capability is increased [30]. Depending on the level of error detection/correction and complexity of implemented coding algorithm, there are various reliability, performance, power and area trade-offs [52].

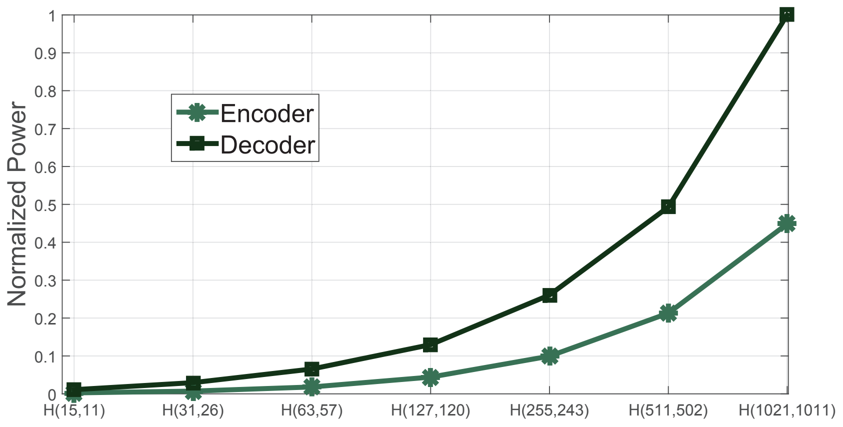

Figure 3.2: Power consumption of Hamming(n,k) encoder and decoder for different data (k) sizes

To better understand the overhead of error correction using coding techniques, Figures 3.2, 3.3, and 3.4 illustrates the effect of Hamming SECDED scheme on power, latency, and area for various data sizes. This result is provided by implementing different sizes of Hamming decoder and encoder, and synthesized it using Synopsys Design Compiler tool in 28nm process technology. As it is observed, as the data size increases, the power consumption and area grows exponentially,but the latency increases linearly. The reason is that Hamming is implemented using balanced XOR tree. Based on this observation, which is one of the key motivations of this thesis, it is tried to utilize the low ECC sizes when architecting the NoC and cache to avoid exponential growth in power consumption.

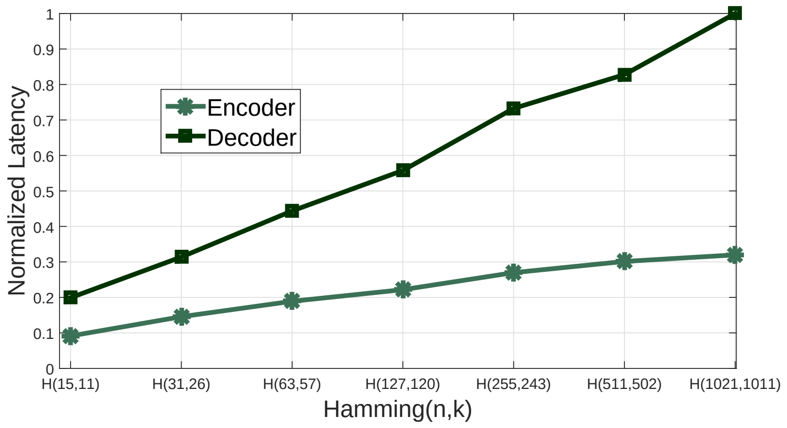

Figure 3.3: latency of Hamming(n,k) encoder and decoder for different data (k) sizes

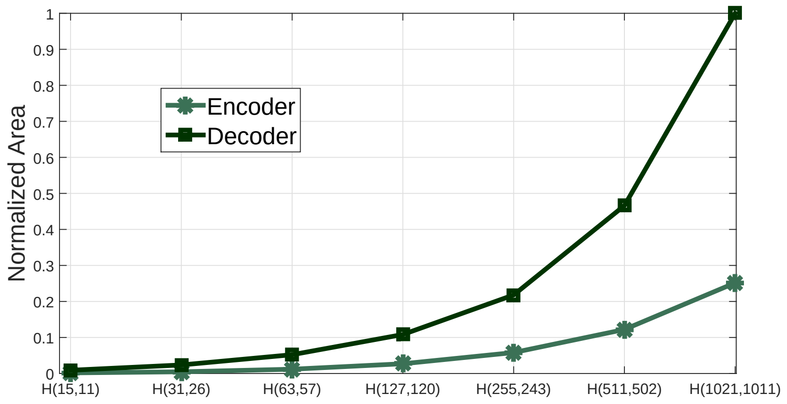

Figure 3.4: Area of Hamming(n,k) encoder and decoder for different data (k) sizes

For interconnected architectures using NoCs, if the data paths are infected by any faults, instantaneous fault detection can be provided by a combination of forward error correction (FEC) and error detection codes. ECC schemes can be implemented in the data link layer as s2s or in transport layer as e2e scheme [40]. In the s2s approach, the flits are encoded and decoded in each hop of the transmission from the sender to the receiver and the encoders (decoders) of the ECC are implemented in the output and/or input ports of the routers of the NoC. In the e2e approach, encoding flits happens in the sender side and decoding them happens only at the receiver side and the encoders (decoders) of the ECC are implemented in the network interface (NI) of the cores. The area overhead of e2e approach includes one encoder, one decoder, and registers of extended buffer, while for s2s policy, the area overhead equals to overhead of four decoder, four encoder, and registers of extended buffer [26]. When faults rates are low, robust s2s encoding schemes consume excess power which is unnecessary and can be avoided by e2e schemes. At the same time, s2s schemes can provide better fault coverage than e2e schemes when fault rates are high, especially in large hop networks. The choice of either e2e or s2s fault detection scheme in NoCs has power, area and performance trade-offs that vary with the number of CMPs and fault rate in the system. The e2e error handling method is better to be selected for low-noise applications whereas the s2s can be utilized in noisy environments. The e2e error handling architecture is considered as the future work for this thesis.

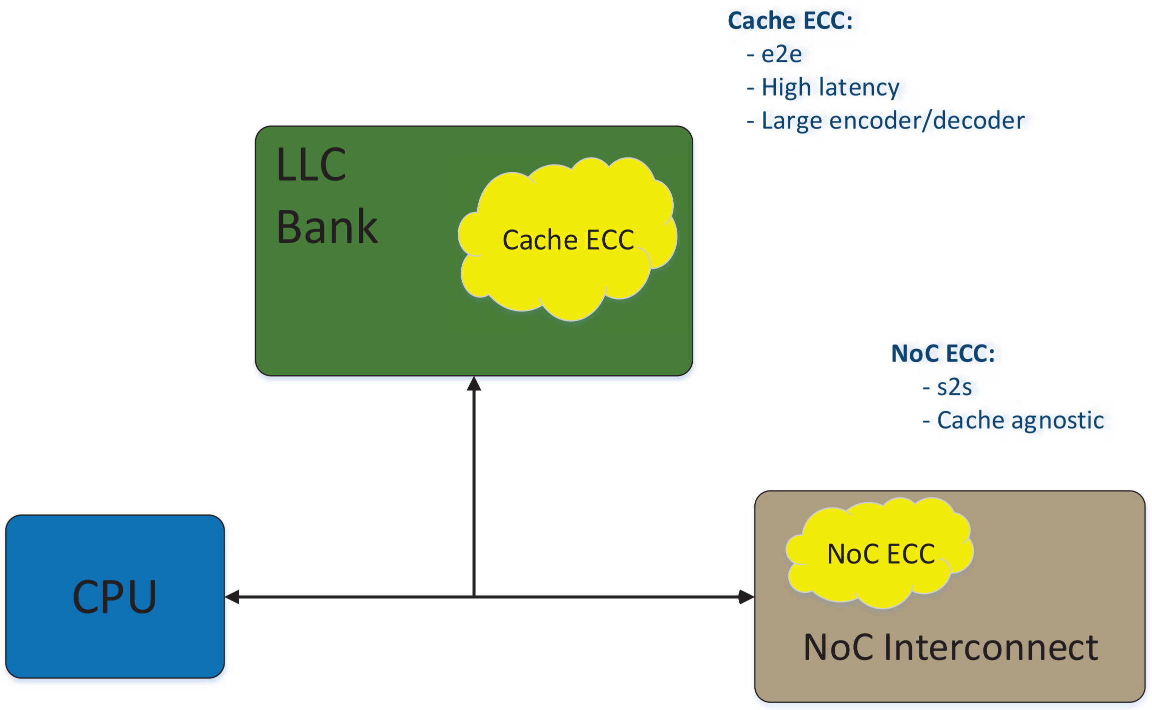

Figure 3.5: An abstract view of error handling in baseline approach; There are separate error detection/correction approaches in NoC and shared cache.

3.1.2 Overview of Baseline Approach

The baseline approach uses ECC separately in NoC interconnect and shared cache banks to protect remote cache accesses through the network from unexpected errors. Figure 3.5 shows an abstract schematic of error coding for both LLC and NoC interconnect in baseline approach. As the figure depicts, the most common error coding methods consider NoC and cache banks separately.

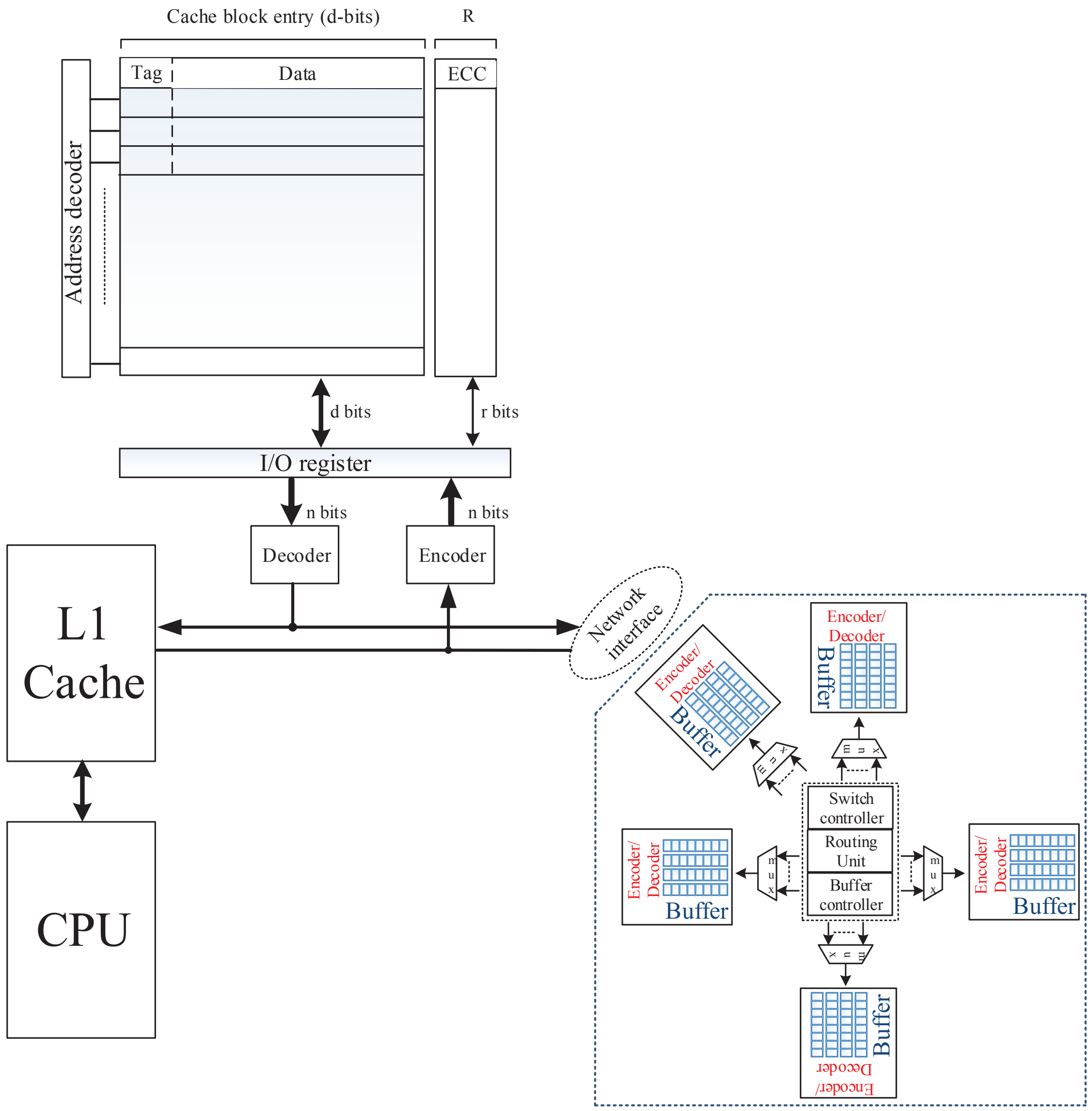

Figure 3.6: Baseline architecture with a relatively expensive Hamming encoder/decoder in the local/remote cache access path.

Figure 3.6 shows a more detailed architecture of baseline method in which each cache line owns an ECC extension, and every read/write requires error detection/correction encoding, respectively. Also on the NoC side, each flit is protected with s2s (hop by hop) ECC which protects data packet against error over the interconnect. Figure 3.6 depicts that each cache bank has its own encoder/decoder which are in critical path of both local core and network accesses. Based on this architecture, for each shared cache access (regardless of access type; local or remote), the entire requested cache line is required to be encoded in write path or decoded in read path. And as discussed before, encoding/decoding the entire message at one shot imposes a large amount of power consumption. Since in shared Non-Uniform Cache Access (NUCA) caches significant number of accesses to cache banks are from remote cores, the cache blocks need to be protected in the interconnection network as well. On the other hand, based on Figure 3.6, cache ECC uses large encoder/decoder which are in critical path of each LLC bank access and due to significant number of remote accesses in large CMPs, it will impose high latency and energy overhead to the system.

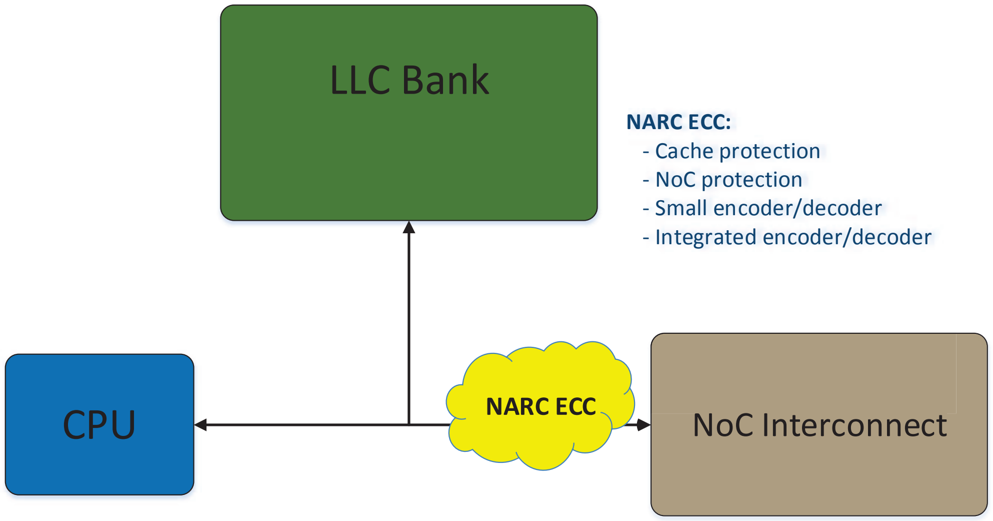

Figure 3.7: NARC in essence; The idea is to integrate the NoC and shared cache error handling mechanism by leveraging from existing NoC error recovery method.

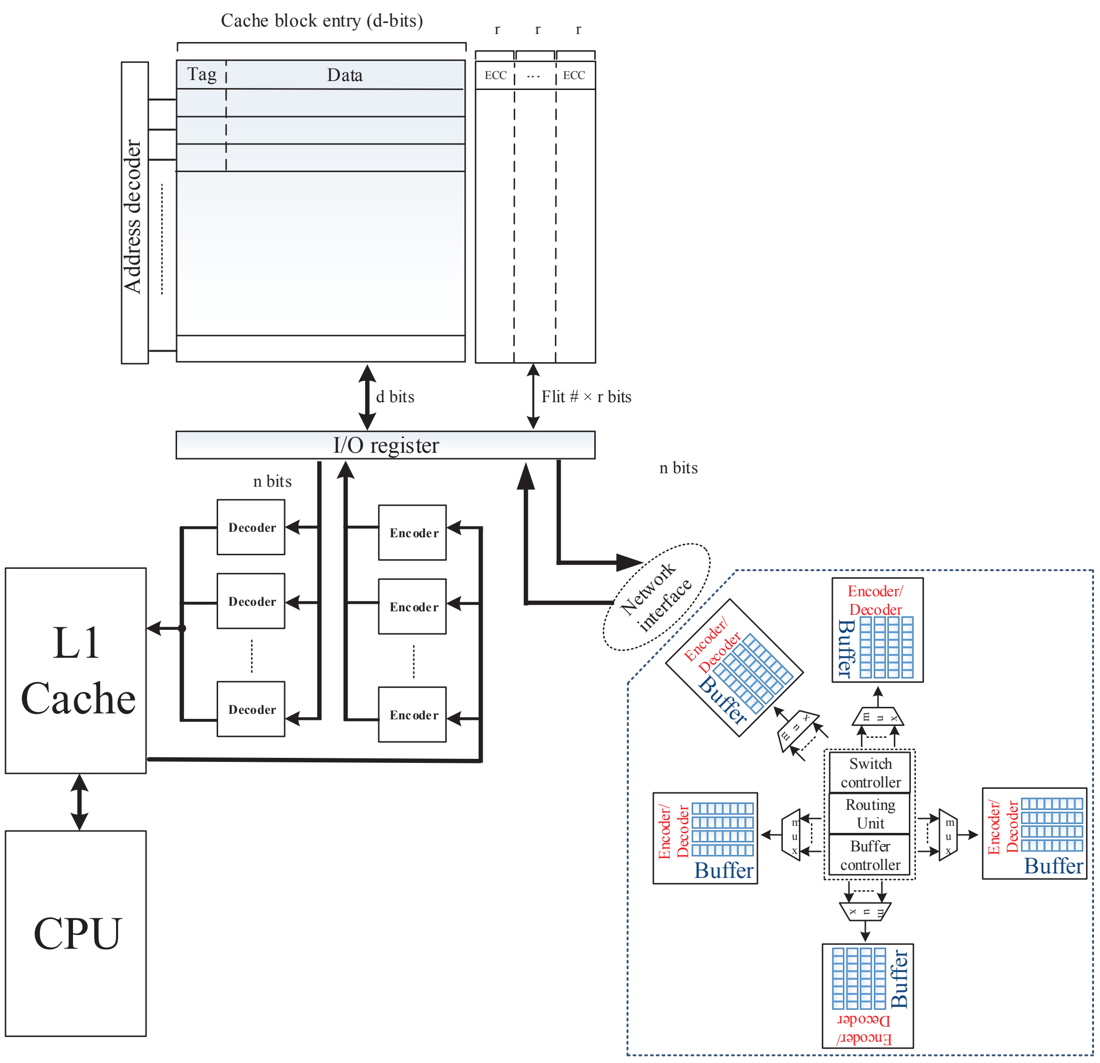

Figure 3.8: Detailed NARC architecture; The encoder/decoder path for both local and remote access is modified. Local LLC cache access includes four small encoder/decoder to handle error detection/correction for each data segment in the cache and the corresponding ECC. Also the remote cache access is directly connected to network interface which improves the remote shared cache access latency. The ECC section in the LLC is now broken into multiple smaller ECCs.

3.2 NARC, a NoC-Assisted Reliable Cache method

The proposed architecture in this thesis, called as NARC, a NoC-Assisted Reliable Cache scheme, is an integrated error coding method which leverages from switch-to-switch ECCs used in NoC flits, in order to protect shared cache memory banks. These ECC blocks are located in segmented manner in each cache line. Figure 3.7 shows an abstract figure of proposed error coding for both LLC and NoC interconnect in NARC approach. This approach reduces the latency and power of shared LLC protection while providing the same level of reliability compared to conventional approaches. In NARC, block-level strong ECC in LLC banks are replaced with a segmented low-cost ECC and ECC bit chunks are sent along the block through the network as it is presented in Figure 3.8. Using this design, NARC removes the large encoder/decoder of cache banks from the critical path of LLC access. Compared to the baseline design, NARC incurs a small ECC storage overhead compared to the baseline approach and that's because of having multiple smaller ECCs for the entire cache line As depicted in Figure 3.8, LLC controller and network interface components inside each tile require small architectural modification to be able to support NARC.

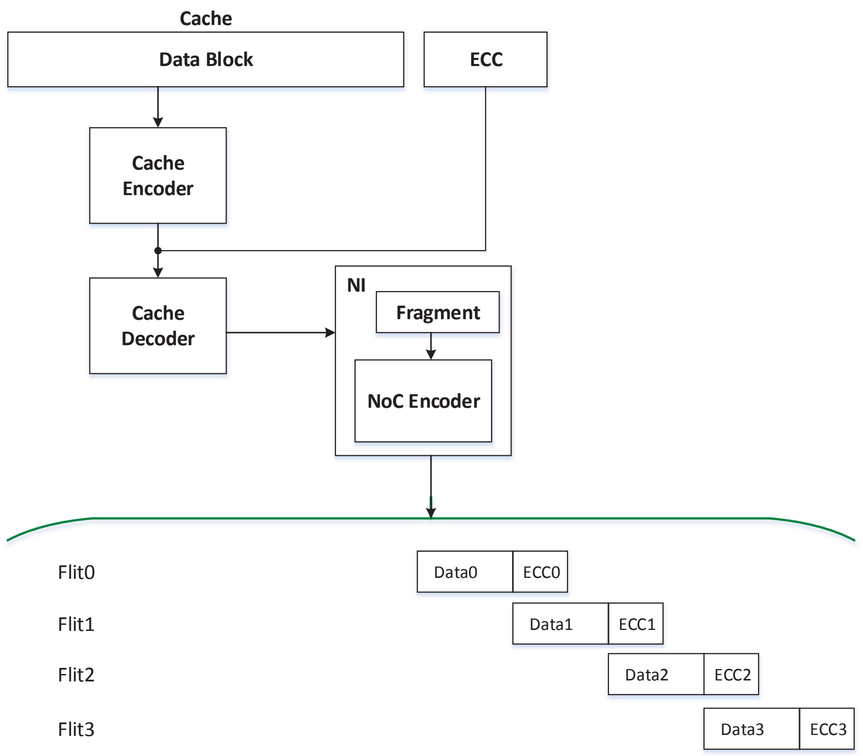

Figure 3.9: Network interface and LLC controller interaction in baseline method. The ECC for each flit is generated inside network interface in the read path, and in the write path, the ECC of each flit is stripped off and only the data is delivered to LLC controller for further error handling.

In LLC banks, the data array needs to store the ECC segments from each NoC flit at the end of each cache line. The large cache-level encoder/decoder blocks are now substituted by multiple less stronger and hence faster ones, but parallel and only on the path between shared cache and private local L1 cache/core. With this scheme, there is no large cache-level encoder/decoder blocks in the critical path of remote data accesses, as shown in Figure 3.8.

The encoder/decoder blocks inside network interface are also removed from the LLC access channel and its internal logic is modified to transfer the ECC bits of each cache segment along the data during flit de-fragmentation(assembly) process. As it is depicted in Figure 3.9, baseline approach uses large encoder/decoder to fragment each cache data block. Figure 3.10 illustrates the fragmentation process of each cache block and construction of data flits in the proposed architecture, based on 3.9 and 3.10, in proposed approach, for each flit in each cache line, there is an ECC which is sent to network interface too.

3.3 Discussion

In this section, some of the possible concerns about the devised approach is being addressed in style of question and answer.

3.3.1 Theoretical Reliability Calculation

Question: *One big flaw of the method is the theoretical calculations of error probability. Recent studies have shown, that errors created by for example radiation generates correlated multiple errors. Moreover the errors generated by the transport net, like crosstalk, also generate correlated errors. So the assumption of uncorrelated errors is not the state of the art.*

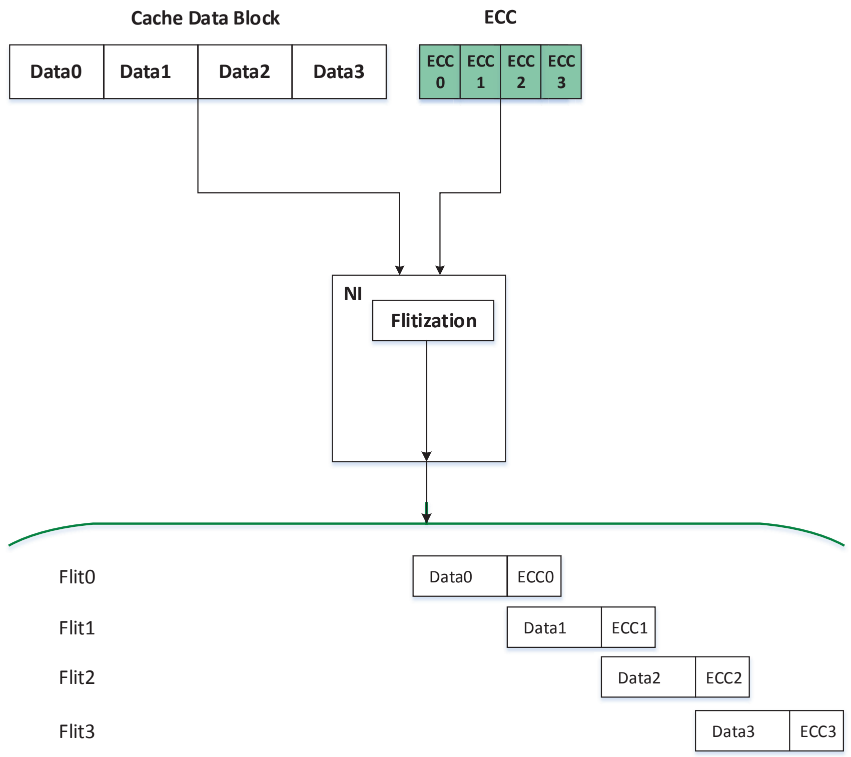

Figure 3.10: Network interface and LLC controller interaction in NARC approach. There is no need for error handling inside network interface, since flits will be error checked in the first NoC router, and each data segment of a cache line fits into one flit, hence the corresponding ECC for each data segment can be applied to each flit.

Answer: Recent development in cache physical fabrication has resolved the correlated multiple-bit error issue. Article [55] discusses the interleaving of SRAM or DRAM cells to create a logical checkword from physically disperse areas of the memory array. This separation of bit lines and word lines is often called column and row stride. Hence with this solution, the multi-cell upset will lead to multiple single-bit errors, NOT a multi-bit error which helps us to consider the error rate uniform throughout the cache.

3.3.2 Error Rate Impact

Question: *This approach will be better if the impact of error rate is evaluated from various aspects, since the error rate has correlation with area, wire length, impedance, capacitance,* *radiation, etc.*

Answer: Perhaps with growth of error rate, there will be a need for stronger ECC for each segment. In this case, the cache block size will grow and so the cache access time. Also there is a need for more complex encoder and decoder which will add extra timing overhead to the cache access. Considering the exponential behavior of decoder/encoder in power consumption and area occupation, in NARC approach due to breaking the data blocks into multiple smaller segments for local accesses, less power consumption and area will be observed compared to baseline method. For remote cache accesses, with elimination of decoding/encoding inside cache, also less latency and power and area will be observed compared to baseline method.

3.3.3 Network Protocol

Question: *A major concern might be that the network protocol flits will need to be encoded and decoded at each router to allow routing and provide reliability, but how about the flits generated by the router?*

Answer: In NARC approach the Network-on-Chip protocol is intact. Overall in Fault Tolerant (FT) NoC architectures, in order to keep the critical path untouched the header flits never get encoded; hence no need for decoding.

3.3.4 ECC Latency to Cache Access Latency Ratio

Question: *The motivation of this thesis might be still questionable without having the latency contributions of the ECC logic (both cache line and s2s or e2e) in the overall access latency. If the delay percentages of the ECC logic are not significant, the proposed scheme will not have any benefits.*

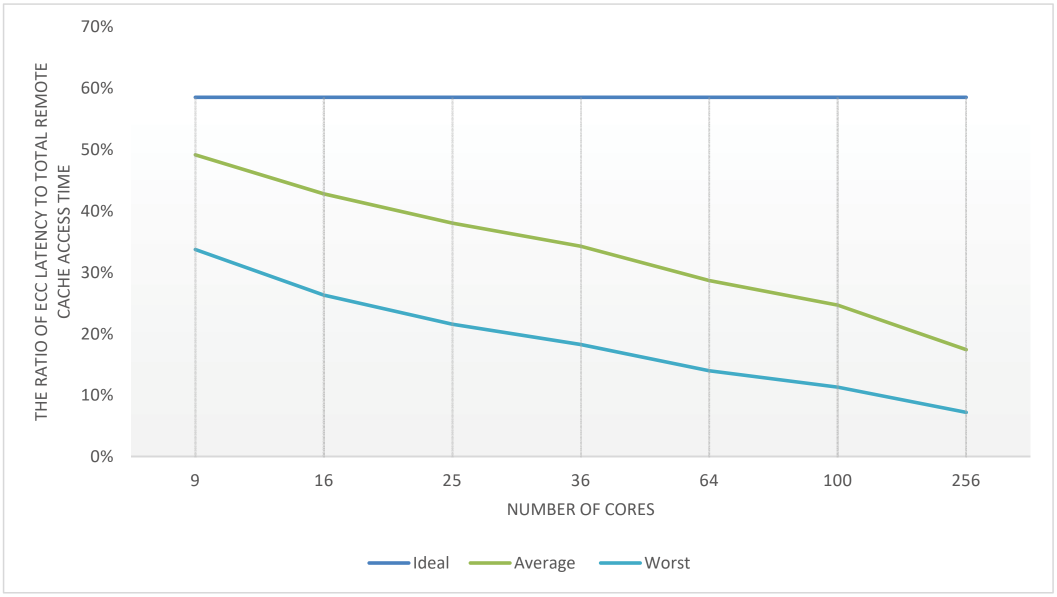

Figure 3.11: Total ratio of error recovery latency contribution to remote shared cache access latency for different NoC sizes with XY dimension-order routing for three different scenarios.

Answer: The ratio of ECC logic latency to total remote LLC cache access time is depicted in Figure 3.11. This question is a valid concern. For example, if the remote cache access latency is 500ns, and the average ECC generation/check latency is 5ns, the overall gain of NARC will be about 1%, which would be negligible.

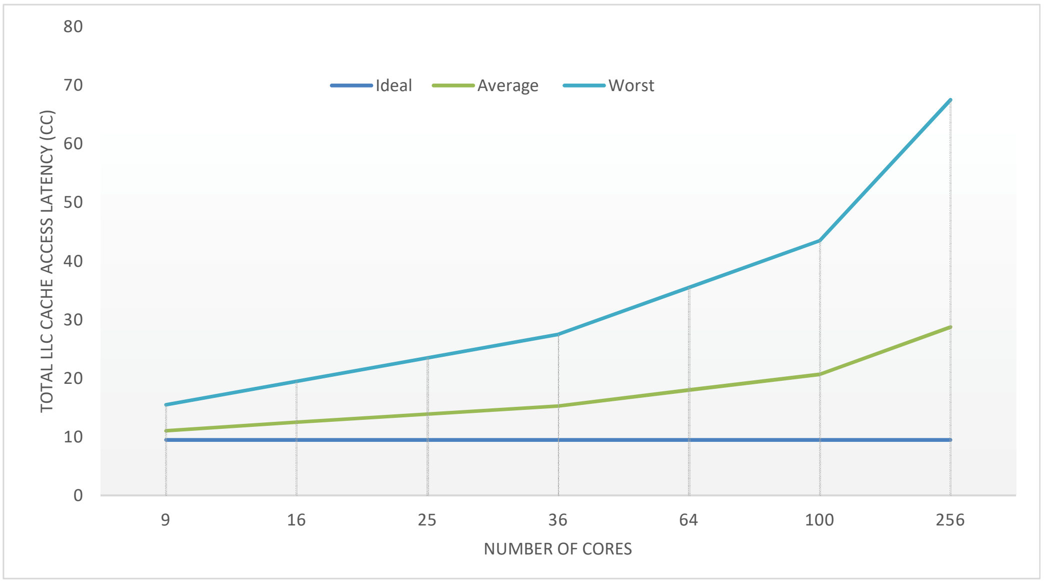

Figure 3.12: Total LLC cache access latency

However, as it is shown in Figure 3.11, the is not the case for even a 256-core SoC. The ideal, average, and worst case scenarios for ECC logic latency to total remote LLC cache access time ratio depicted in Figure 3.11. In ideal case the least remote cache access is achieved when the requested cache line resides in the neighbor tile which with one hop of network communication it can be fetched.

In average and worst cases, the NoC routing protocol plays a critical role in number of hop counts. In this thesis, all evaluations are performed with the XY dimension-order routing protocol. Therefore, the worst-case cache access would be when the cache line requester and responder cores are located on the two sides of network diameter. As it is observed, NARC can achieve up to 60% improvement in lowering the remote cache access latency. On average and having 256 core in a network of 8x8 tiles, which is an extreme case, still NARC gains about 20% lower cache access latency.

Similarly, Figure 3.12 depicts total LLC cache access latency versus number of cores. As it is shown, on average the cache access latency of networks with higher number of cores is non-linearly higher.

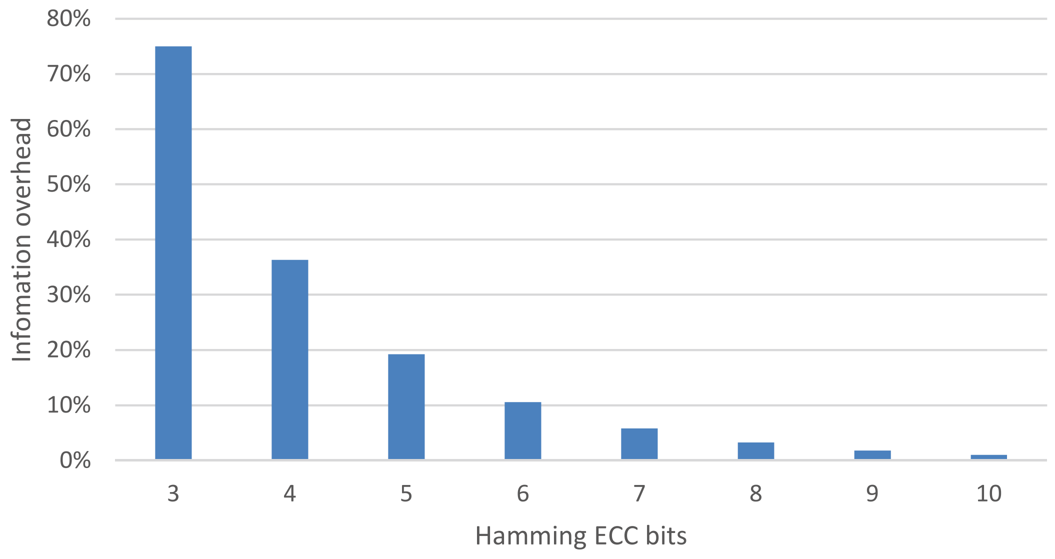

Figure 3.13: Hamming redundancy ratio

3.3.5 ECC Information Redundancy Overhead

Question: *How does ECC overhead affects in choosing the segment size?*

Answer: In order to better understand the ECC overhead, Figure 3.13 demonstrates the redundancy overhead rate for Hamming method (selected as a SECDED method in this thesis).

As it is observed, as the information size grows and so does the ECC bits, the overhead rate of redundancy bits lowers exponentially. Therefore, for choosing the segment size, it is desired to have the largest size possible for each segment. However, on the other hand, as the data size in each segment grows, the probability of failure in each segment and in total is higher, as depicted in Figure 3.14. In order to elaborate the effect of cache line/segment data size

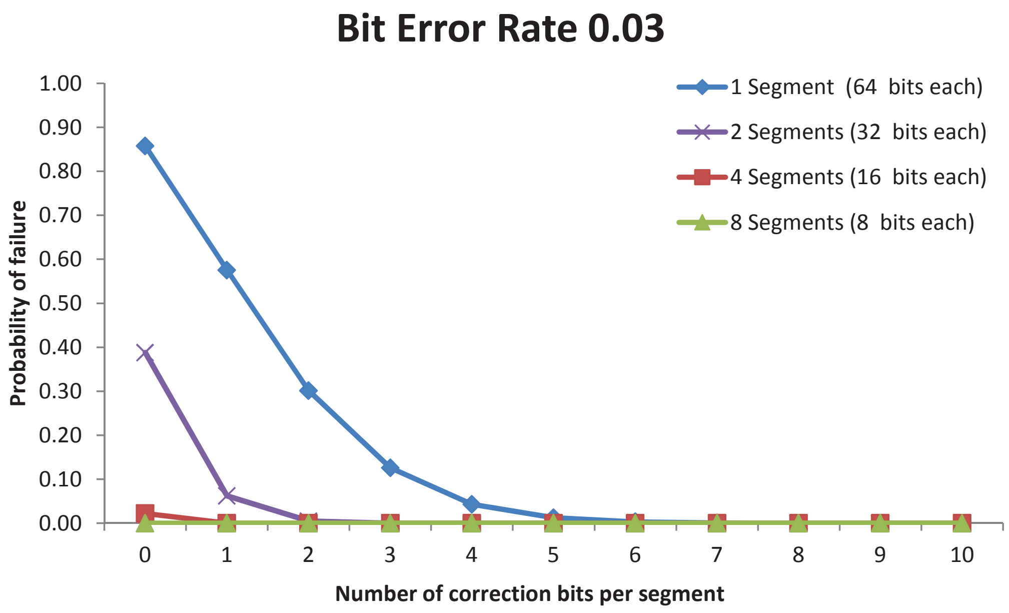

Figure 3.14: Probability of failure versus correction capability per segment for cache line size of 64-bit and BER of 0.03.

and bit error rate on the probability of failure, couple of experiments are performed and the result is depicted in Figures 3.14-3.18. As it is shown, as the data size in each segment grows, the probability of failure increases accordingly. The reason is that the correction capability is fixed to 1-bit due to utilization of SECDED hamming method. It can be concluded that in the application with higher bit error rate, the architecture can be invigorated by using stronger ECC methods like BCH.

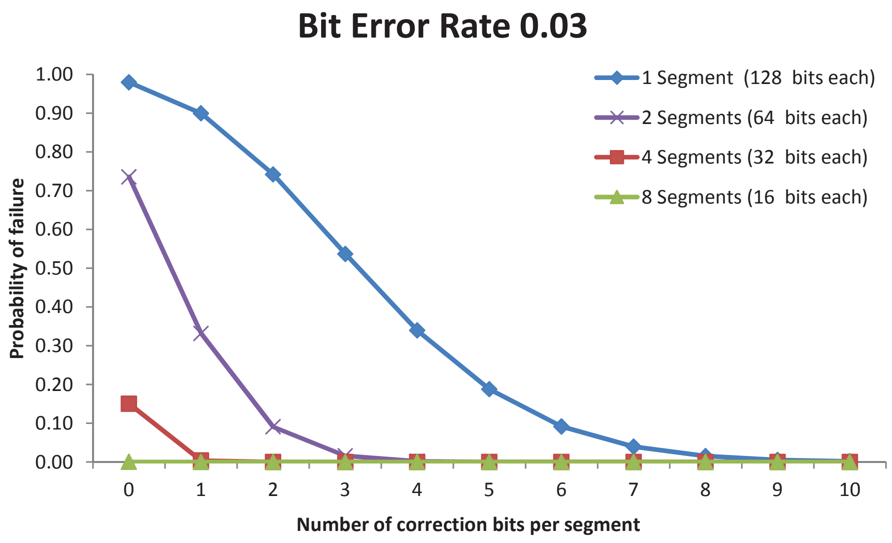

Figure 3.15: Probability of failure versus correction capability per segment for cache line size of 128-bit and BER of 0.03.

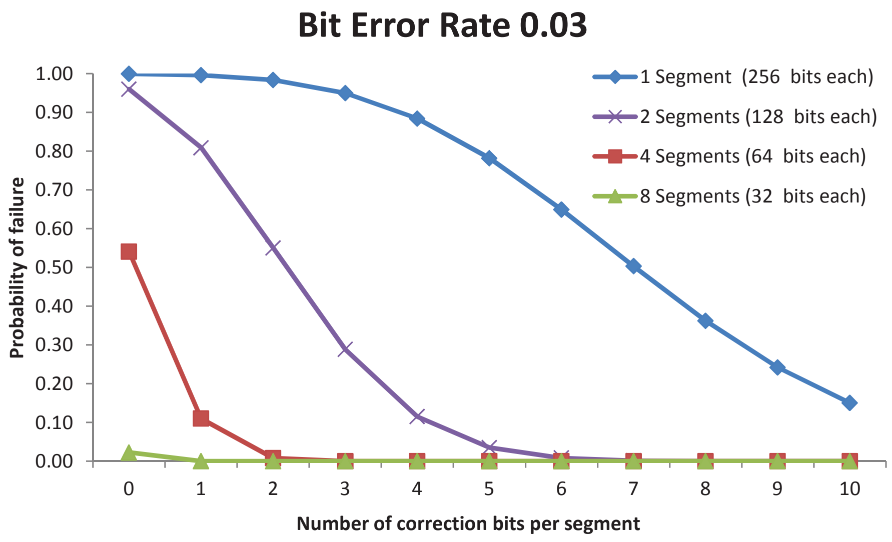

Figure 3.16: Probability of failure versus correction capability per segment for cache line size of 256-bit and BER of 0.03.

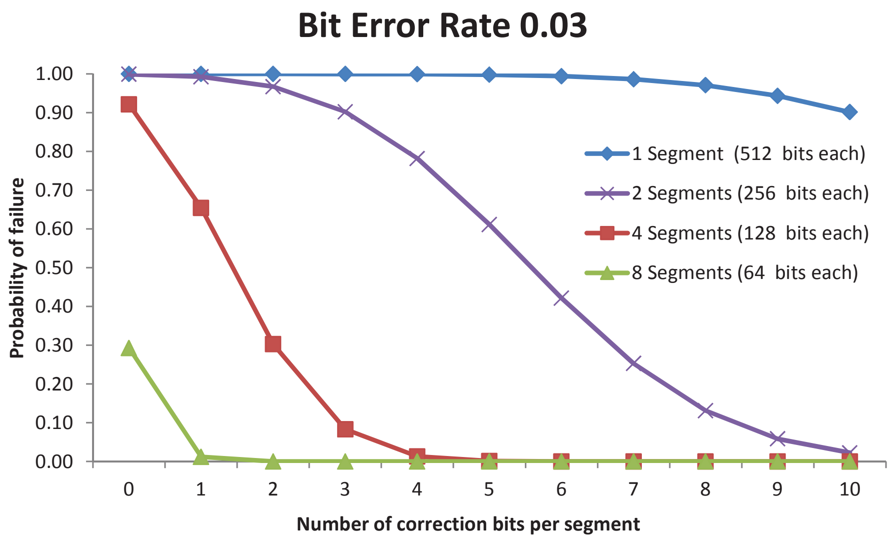

Figure 3.17: Probability of failure versus correction capability per segment for cache line size of 512-bit and BER of 0.03.

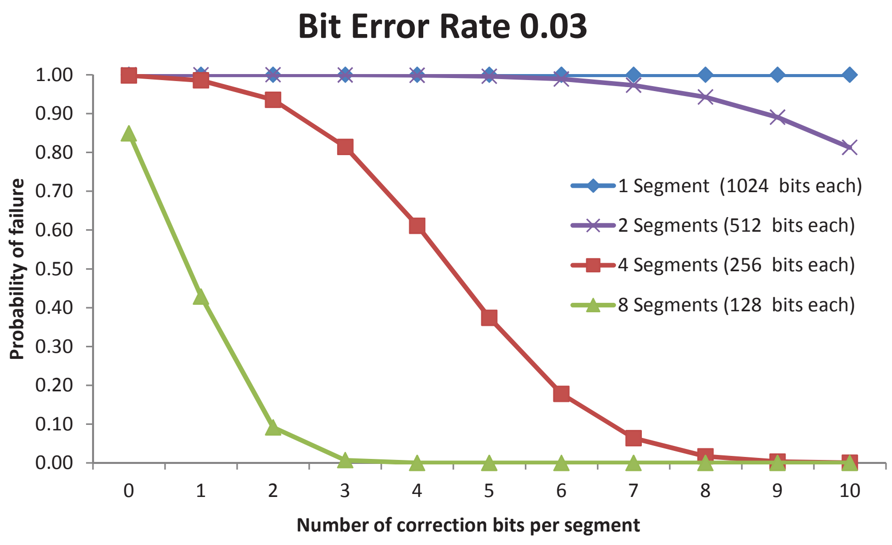

Figure 3.18: Probability of failure versus correction capability per segment for cache line size of 1024-bit and BER of 0.03.

In a similar experiment, the data size in each cache line is kept the same at 512-bits, and the bit error rate is increased from 0.005 to 0.01. As it is expected, as the bit error rate grows the probability of failure increases as well, however having more number of segments leads to less probability of failure.

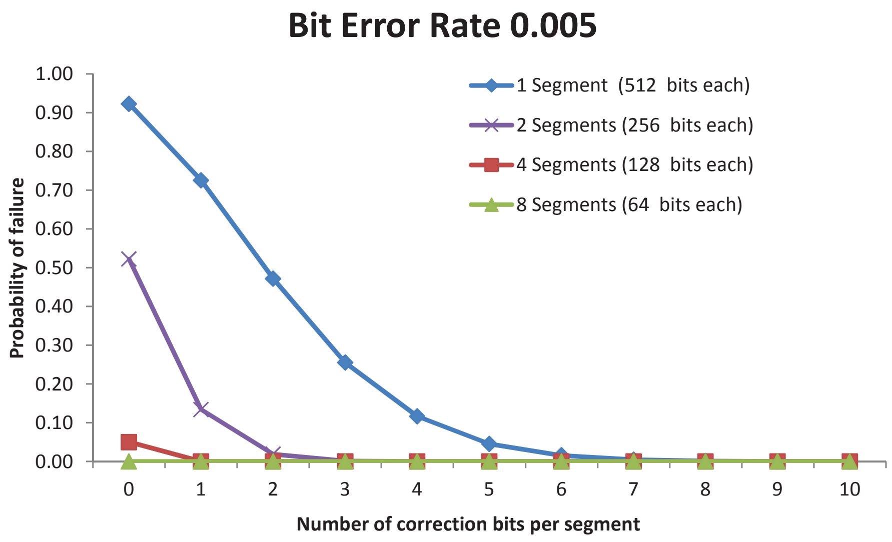

Figure 3.19: Probability of failure versus correction capability per segment for cache line size of 512-bit and BER of 0.005.

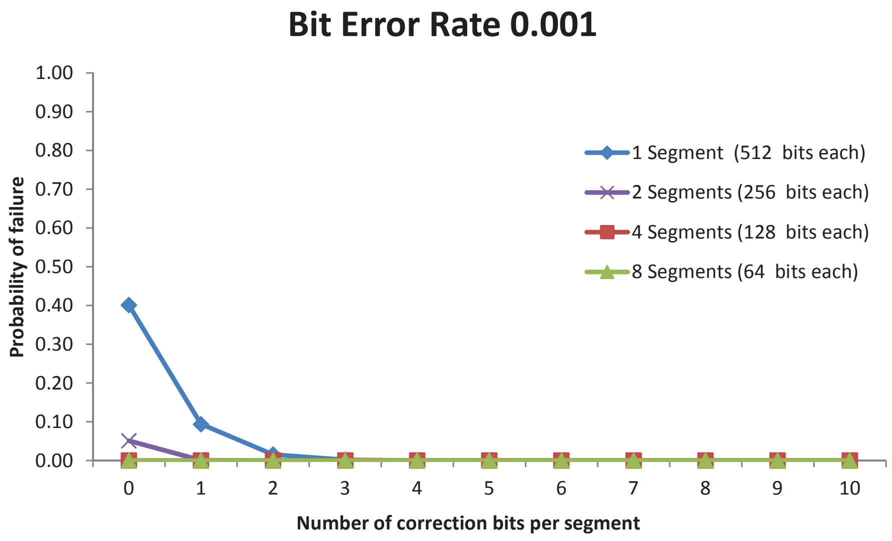

Figure 3.20: Probability of failure versus correction capability per segment for cache line size of 512-bit and BER of 0.001.

In order to elaborate the effect of having more segments per cache line on the total information redundancy, Figure 3.22 depicts the total required ECC bits versus various correction capabilities for different number of segments. It is observed that having more number of segments per cache line leads to the additional redundancy memory overhead. In this thesis,

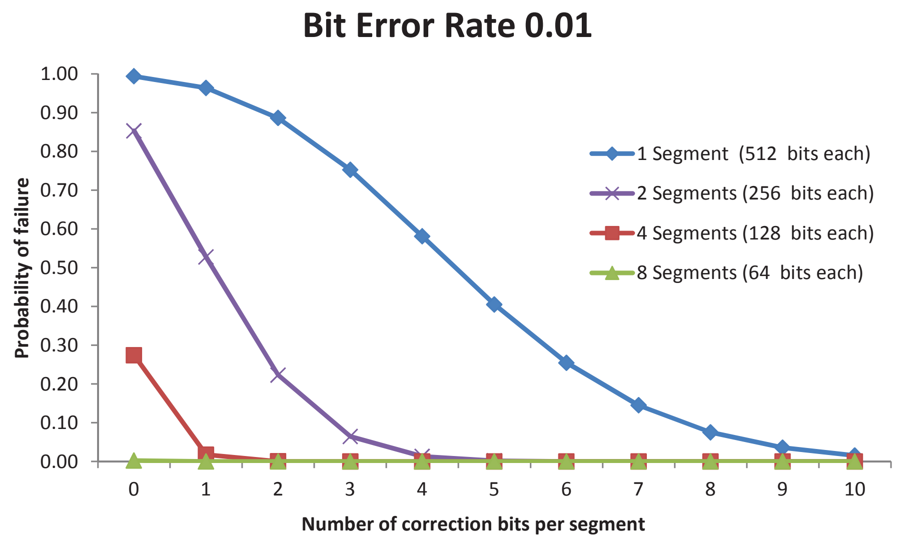

Figure 3.21: Probability of failure versus correction capability per segment for cache line size of 512-bit and BER of 0.01.

the cache/NoC is tuned to support four segment cache blocks. The reason is that the ECC information redundancy rate for 4-segment, considering 1-bit correction, is relatively small, also the probability of failure for 4-segment is tolerable.

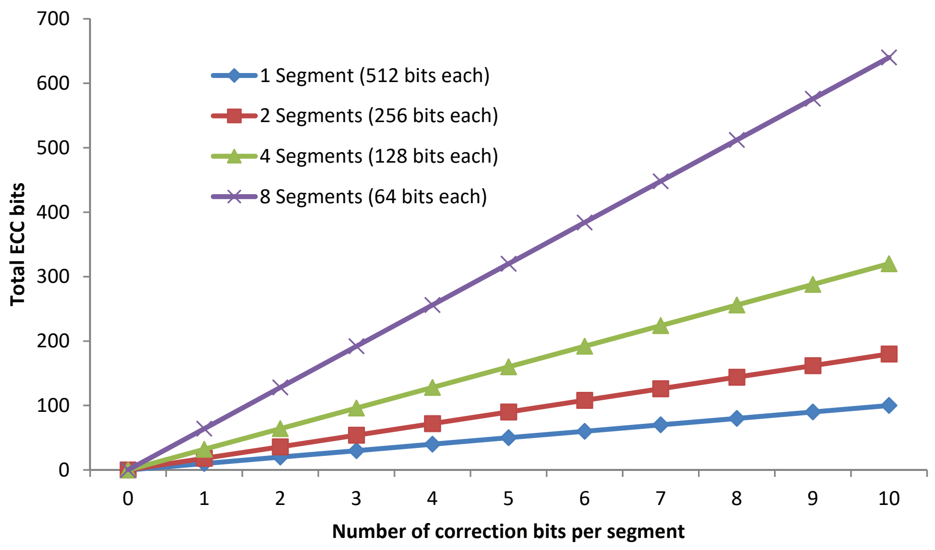

Figure 3.22: Total ECC bits versus number of correction bits per segment

Chapter 4

Empirical Evaluations

4.1 Methodology

To evaluate the efficacy of NARC, full system simulations are performed using the GEM5 simulator with the Ruby memory system and the Garnet network on chip model [11]. The primary results are for a chip multiprocessor (CMP) with 36 in-order Alpha cores [37]. The focus is on in-order cores as their small size and power efficiency allow aggressive exploitation of thread-level parallelism. The rest of the configuration is shown in Table 4.1. Unless otherwise specified, there is no prefetching. Unless otherwise specified, we use applications from the PARSEC benchmark suite [10] with their pre-defined medium-size inputs. Results for all benchmarks that execute in our simulation environment are shown. Cacti 6.5 is utilized with a 32nm process [34] to estimate the power and area of caches.

Figure 4.1 demonstrates the shared cache power consumption, including the data memory, ECC memory, and error detection/recovery logic. The numbers are normalized to the DECTED, which shows the maximum power consumption. Hamming and BCH are utilized as the SECDED and DECTED methods, respectively. As it is shown, a major portion

Table 4.1: Simulated system configuration

Cores	36 cores, 2GHz
ISA	ALPHA
Private (L1) Caches	32KB, 2-way set, 3 cycle-access
Shared LLC Cache	Shared, 2MB per core, 8-way set, distributed (one per core), 15 cycle-access
Cache lines	64 bytes
Coherence protocol	MESI
Memory controllers	4, one at each corner, each one DDR3-1600 x64 channel, Micron MT41J512M8
Topology	6x6 mesh, deterministic routing, single-cycle 16-byte channels, 5-stage routers

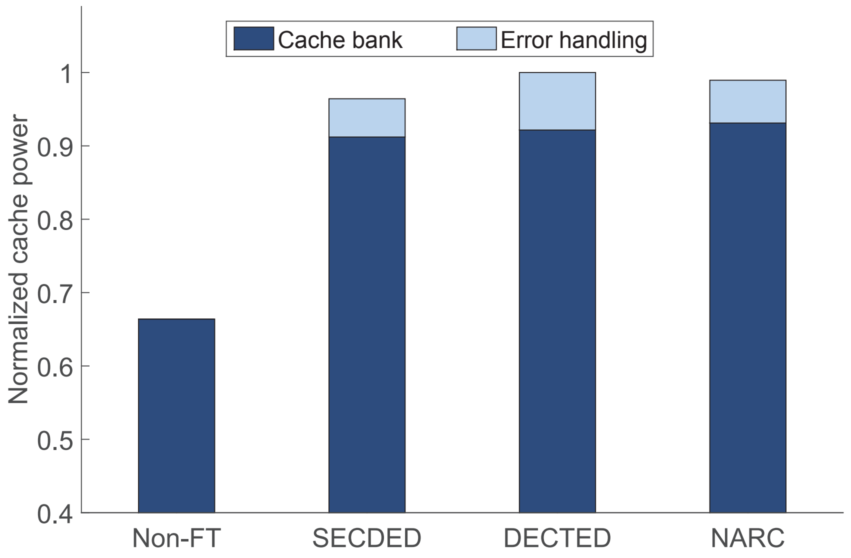

Figure 4.1: Shared cache power consumption, including the data memory, ECC memory, and error detection/recovery logic. The result is normalized to the DECTED, which shows the maximum power consumption. Hamming and BCH are utilized as the SECDED and DECTED methods, respectively.

of power is consumed by the extra memory allocated to ECC, which is slightly more in NARC approach due to more number of ECC bits in this method. However, having segmented ECC in NARC approach leads to lower error detection/recovery logic compared to DECTED scheme. All in all, it can be seen that NARC consumes less power in total compared to DECTED.

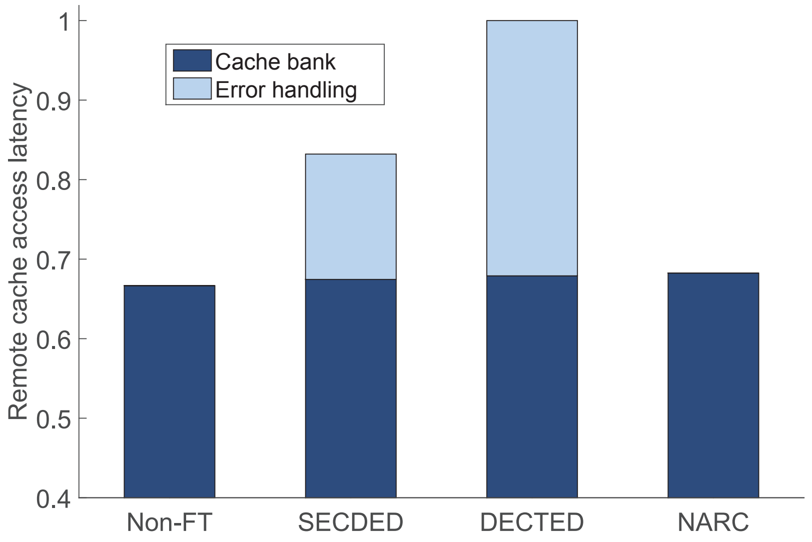

Figure 4.2: Remote cache access latency for each approach.

Figure 4.2 depicts the remote LLC cache access latency for different approaches. As it is seen and expected, in NARC the remote cache access latency is the least of all fault-tolerant approaches, and the reason is that the expensive error handling is eliminated from the data path.

In order to elaborate more on shared cache power, Figures 4.3 and 4.4 illustrate the shared cache dynamic power consumption for read and write operations, respectively, versus different number of segments in a cache line. The results include the data memory, ECC memory, and error detection/recovery logic, for various number of segments. As it is shown, as the number of segment increases, the ECC encoder (in write path) and the decoder (in the read path) power consumption decreases, and they are getting less stronger, and hence less number of XOR gates. However, the cache memory power increases as the number of segments grows due to slight grow in the number of segmented ECC bits stored in the shared cache.

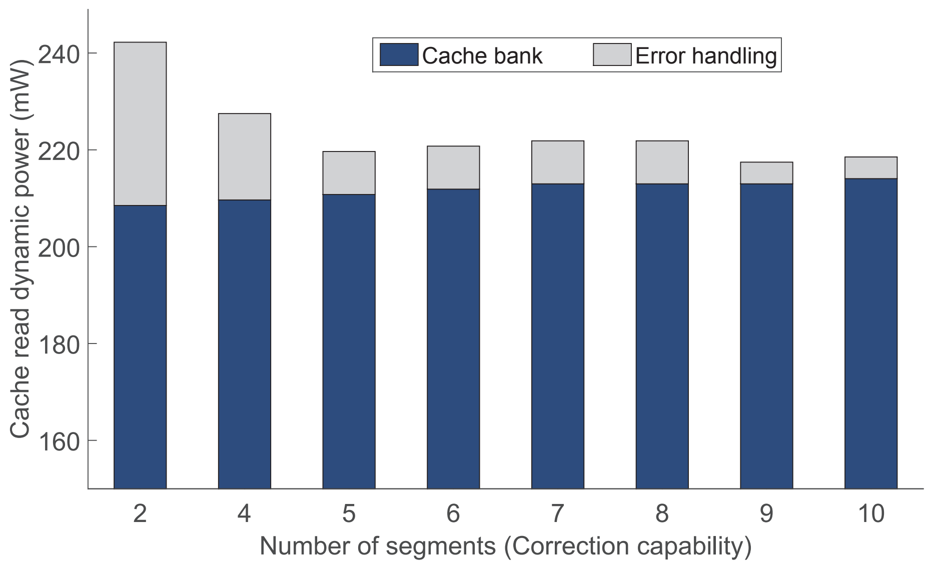

Figure 4.3: Shared cache dynamic power consumption for read operation, including the data memory, ECC memory, and error detection/recovery logic, for various number of segments.

The inherent higher power consumption in read path is due to the fact that decoding logic is always more complex than encoding which leads to more power consumption in the read operation.

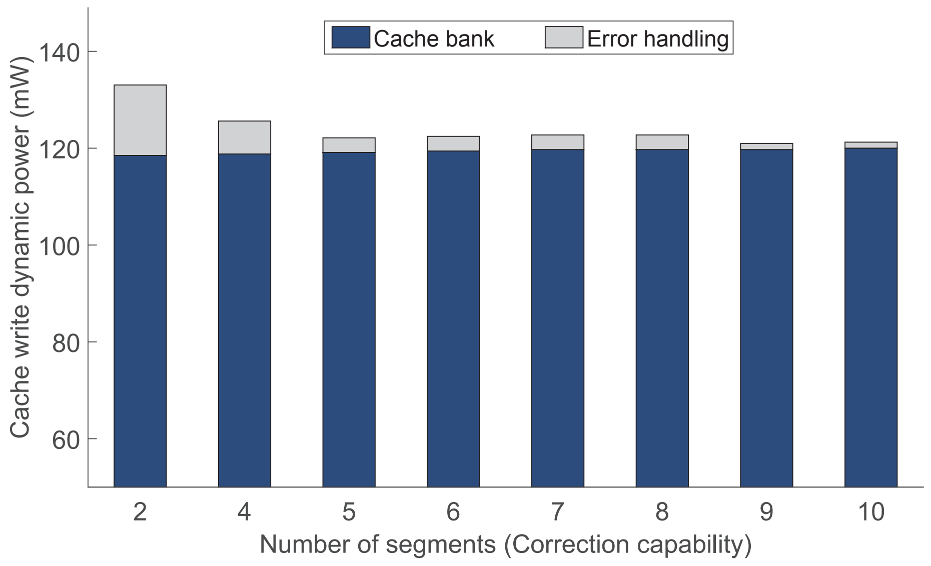

Figure 4.4: Shared cache dynamic power consumption for write operation, including the data memory, ECC memory, and error detection/recovery logic, for various number of segments.

NARC does not modify NoC router architecture, rather baselines a fault-tolerant NoC router architecture, and modifies the flit size in accordance with the segment size. In order to clarify more on the NoC architecture used for NARC method, the NoC router power and area is reported in Figures 4.5 and 4.6, respectively. The number of ECC bits varies according to the flit sizes; Data flit sizes in range for these results are varied from 32 to 256 bits.

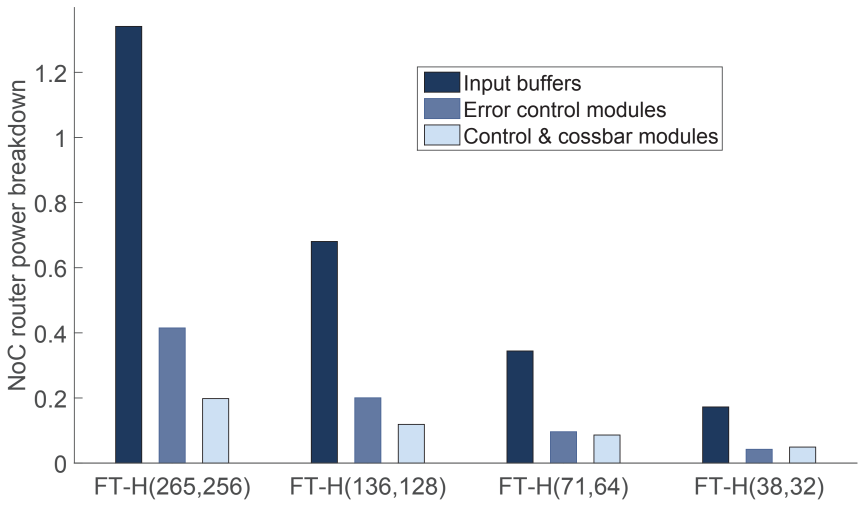

Figure 4.5: NoC router power breakdown for main components: input buffer, control unit, and error handing logic. This result is reported for four different routers based on their resiliency. FT-H stands for Fault tolerant with Hamming method, and combination of (n,k) represents code and data, respectively.

Figure 4.5 depicts NoC router power breakdown for main components: input buffer, control unit, and error handing logic. This result is reported for four different routers based on their resiliency. FT-H stands for Fault tolerant with Hamming method, and combination of (n,k) represents code and data, respectively. Input buffers, because of their inherent sequential logic (implemented either with SRAM or Flip-Flop) and being in the critical path of NoC routers, consume the most amount of power among NoC router components. Error handling logic, for high bandwidth routers, consumes more power than the control logic, as ECC logic resides in the router data path, imposing dynamic power.

Similarly, Figure 4.6 shows NoC router area occupation breakdown for main components:

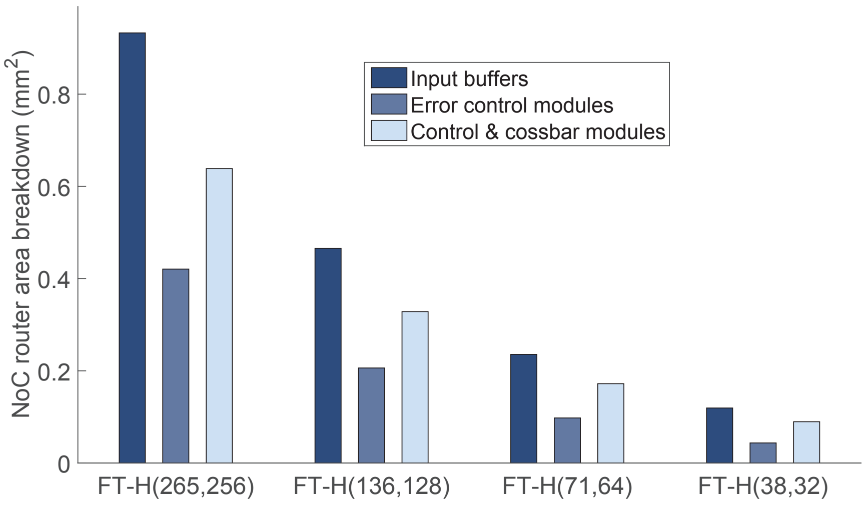

Figure 4.6: NoC router area occupation breakdown for main components: input buffer, control unit, and error handing logic. This result is reported for four different routers based on their resiliency. FT-H stands for Fault tolerant with Hamming method, and combination of (n,k) represents code and data, respectively.

input buffer, control unit, and error handing logic. This result is reported for four different routers based on their resiliency. ECC logic consumes less area than the control unit but lower than input buffers.

Chapter 5

Conclusions and Future Roadmap

5.1 Summary of Contributions

I summarize the contributions of this thesis in the following sections. This thesis devises NARC, a Network-on-Chip Assisted Reliable Cache architecture to lower the last-level shared cache access latency in shared-memory many-core systems. It is explained the severity of error coding codes in latency and power. Subsequently, it is demonstrated by leveraging from existing error detection/correction logic in NoC routers, remote cache accesses throughout a many-core system are accessed faster, with almost the same reliability rate. Throughout the thesis, various trade-offs are discussed and explored to provide more elaboration on the efficacy of the proposed method. At the end, experimental results are provided to support the effectiveness of NARC. Also, a comprehensive study and analysis of reliability issues in cache memory banks are detailed.

5.2 Future Work

The potential future research work worth pursuing are categorized in the following areas:

5.2.1 Extending NARC to support DMA and memory (DRAM) controllers.

5.2.2 Explore NARC architecture for end-to-end fault tolerant scheme.

5.3 Concluding Remarks

In conclusion, the work presented in this dissertation provides an insight to make the shared memory many-core architectures faster at the same time reliable.

Bibliography

[1] TOP500 Supercomputer Site.

[2] A. Agarwal, B. C. Paul, H. Mahmoodi-Meimand, A. Datta, and K. Roy. A process-tolerant cache architecture for improved yield in nanoscale technologies. *IEEE Trans. VLSI Syst.*, 13(1):27–38, 2005.

[3] H. Aliee, H. R. Zarandi, and P. M. Yaghini. K2router: A low-power and high-performance router design for networks-on-chip. *Journal on Computer Science and Engineering*, 7(2):8–23, 2011.

[4] F. Angiolini, D. Atienza, S. Murali, L. Benini, and G. D. Micheli. Reliability support for on-chip memories using networks-on-chip. In *ICCD*, 2006.

[5] A. Ansari, S. Feng, S. Gupta, and S. A. Mahlke. Archipelago: A polymorphic cache design for enabling robust near-threshold operation. In *HPCA*, 2011.

[6] S. A. Asghari, H. Pedram, M. Khademi, and P. Yaghini. Designing and implementation of a network on chip router based on handshaking communication mechanism. *World Applied Sciences Journal*, 6(1):88–93, 2009.

[7] L. Benini and G. De Micheli. Networks on chips: a new soc paradigm. *Computer*, 35(1):70–78, Jan 2002.

[8] D. Bertozzi, L. Benini, and G. De Micheli. Error control schemes for on-chip communication links: the energy-reliability tradeoff. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(6):818–831, 2005.

[9] D. Bertozzi, L. Benini, and G. D. Micheli. Error control schemes for on-chip communication links: The energy-reliability tradeoff. volume 24, pages 818–831, 2000.

[10] C. Bienia, S. Kumar, J. P. Singh, and K. Li. The parsec benchmark suite: characterization and architectural implications. In *PACT '08: Proceedings of the 17th international conference on Parallel architectures and compilation techniques*, pages 72–81, 2008.

[11] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood. The gem5 simulator. *SIGARCH Comput. Archit. News*, 39(2):1–7, Aug. 2011.

[12] P. Bogdan, T. Dumitras, and R. Marculescu. Stochastic communication: A new paradigm for fault-tolerant networks-on-chip. In *VLSI Design*, 2007.

[13] B. H. Calhoun and A. Chandrakasan. A 256kb sub-threshold sram in 65nm cmos. In *ISSCC*, 2006.

[14] C. Chen and M. Hsiao. Error-correcting codes for semiconductor memory applications: A state of the art review. volume 28, pages 124–134, 1984.

[15] J. Chen, D.-W. Chang, C.-P. Young, G.-Y. Huang, S.-L. Chu, C.-Y. Ke, S.-T. Yen, and T.-S. Kuo. Building a multi-kernel embedded system with high performance ipc mechanism. In *High Performance Computing and Communications (HPCC), 2011 IEEE 13th International Conference on*, pages 506–511. IEEE, 2011.

[16] A. Eghbal, P. M.Yaghini, N. Bagherzadeh, and M. Khayambashi. Tsv analytical fault tolerance assessment for 3d network-on-chip. *Computers, IEEE Transactions on*, 2015.

[17] A. Eghbal, P. M. Yaghini, and N. Bagherzadeh. Capacitive coupling mitigation for tsv-based 3d ics. IEEE, 2015.

[18] A. Eghbal, P. M. Yaghini, and N. Bagherzadeh. Capacitive coupling mitigation for tsv-based 3d ics. In *2015 IEEE 33rd VLSI Test Symposium (VTS)*, pages 1–6, April 2015.

[19] A. Eghbal, P. M. Yaghini, N. Bagherzadeh, and M. Khayyambashi. Analytical fault tolerance assessment and metrics for tsv-based 3d network-on-chip. 2015.

[20] A. Eghbal, P. M. Yaghini, H. Pedram, and H. R. Zarandi. Fault injection-based evaluation of a synchronous noc router. In *On-Line Testing Symposium, 2009. IOLTS 2009. 15th IEEE International*, pages 212–214. IEEE, 2009.

[21] A. Eghbal, P. M. Yaghini, H. Pedram, and H. R. Zarandi. Designing fault-tolerant network-on-chip router architecture. *International Journal of Electronics*, 97(10):1181–1192, 2010.

[22] A. Eghbal, P. M. Yaghini, H. Pedram, and H. R. Zarandi. Designing fault-tolerant network-on-chip router architecture. *International Journal of Electronics*, 97(10):1181–1192, 2010.

[23] A. Eghbal, P. M. Yaghini, S. S. Yazdi, and N. Bagherzadeh. Tsv-to-tsv inductive coupling-aware coding scheme for 3d network-on-chip. In *Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2014 IEEE International Symposium on*, pages 92–97. IEEE, 2014.

[24] A. Eghbal, H. R. Zarandi, and P. M. Yaghini. Fault tolerance assessment of pic microcontroller based on fault injection. In *Test Workshop, 2009. LATW'09. 10th Latin American*, pages 1–6. IEEE, 2009.

[25] R. Fernandez-Pascual, J. M. Garcia, M. E. Acacio, and J. Duato. Dealing with transient faults in the interconnection network of cmps at the cache coherence level. *IEEE Trans. Parallel Distrib. Syst.*, 21(8):1117–1131, Aug. 2010.

[26] A. Garamoun and M. Radetzki. Error correction techniques on noc protocol layers. In *Haupt-Seminar on Reliable Network-on-Chip in the Many-Core Era*, volume 23, 2009.

[27] R. W. Hamming. Error correcting and error detecting codes. *Bell System Technical J.*, (29):147–160, 1950.

[28] M. Y. Hsiao. A class of optimal minimum odd-weight-column sec-ded codes. *IBM J. Reserach and Development*, 25(14):295–301, 1970.

[29] J. Huh, C. Kim, H. Shafi, L. Zhang, D. Burger, and S. Keckler. A nuca substrate for flexible cmp cache sharing. *Parallel and Distributed Systems, IEEE Transactions on*, 18(8):1028–1040, Aug 2007.

[30] K. Jangwoo, N. Hardavellas, K. Mai, B. Falsafi, and J. Hoe. Multi-bit error tolerant caches using two-dimensional error coding. In *Microarchitecture, 2007. MICRO 2007. 40th Annual IEEE/ACM International Symposium on*, pages 197–209, Dec 2007.

[31] M. Khayambashi, P. M. Yaghini, A. Eghbal, and N. Bagherzadeh. Analytical reliability analysis of 3d noc under tsv failure. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 11(4):43, 2015.

[32] J. Kim, N. Hardavellas, K. Mai, B. Falsafi, and J. Hoe. Multi-bit error tolerant caches using two-dimensional error coding. In *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 40, pages 197–209, Washington, DC, USA, 2007. IEEE Computer Society.

[33] J. Kim, C. Nicopoulos, and D. Park. A gracefully degrading and energy-efficient modular router architecture for on-chip networks. In *ISCA*, 2006.

[34] S. Li, K. Chen, J. H. Ahn, J. B. Brockman, and N. P. Jouppi. CACTI-P: Architecture-level modeling for sram-based structures with advanced leakage reduction techniques. In *Proceedings of the International Conference on Computer-Aided Design*, ICCAD '11, pages 694–701, 2011.

[35] S. Manolache, P. Eles, and Z. Peng. Fault and energy-aware communication mapping with guaranteed latency for applications implemented on noc. In *DAC*, 2005.

[36] R. Marculescu, m. Y. Ogras, L.-S. Peh, N. D. E. Jerger, and Y. V. Hoskote. Outstanding research problems in noc design: System, microarchitecture, and circuit perspectives. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 28(1):3–21, 2009.

[37] E. McLellan. The alpha axp architecture and 21064 processor. *IEEE Micro*, 13(3):36–47, May 1993.

[38] F. Mohammadian. Falp: A fault adaptive and low power method for network on chip router. In *Architecture of Computing Systems (ARCS), 2014 27th International Conference on*, pages 1–7, Feb 2014.

[39] A. Munir, S. Ranka, and A. Gordon-Ross. High-performance energy-efficient multi-core embedded computing. *IEEE Transactions on Parallel and Distributed Systems*, 23(4):684–700, 2012.

[40] S. Murali, T. Theocharides, N. Vijaykrishnan, M. Irwin, L. Benini, and G. De Micheli. Analysis of error recovery schemes for networks on chips. *Design Test of Computers, IEEE*, 22(5):434–442, 2005.

[41] S. Murali, T. Theocharides, N. Vijaykrishnan, M. J. Irwin, L. Benini, and G. D. Micheli. Analysis of error recovery schemes for networks on chips. *IEEE Des. Test*, 22(5):434–442, Sept. 2005.

[42] S. R. Nassif, N. Mehta, and Y. Cao. A resilience roadmap: (invited paper). In *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '10, pages 1011–1016, 3001 Leuven, Belgium, Belgium, 2010. European Design and Automation Association.

[43] T. A. S. on Exascale Computing. *The Opportunities and Challenges of Exascale Computing*, 2010.

[44] S. Ozdemir, D. Sinha, G. Memik, J. Adams, and H. Zhou. Yield-aware cache architectures. In *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 39, pages 15–25, Washington, DC, USA, 2006. IEEE Computer Society.

[45] D. Pamunuwa. Memory technology for extended large-scale integration in future electronics applications. In *Design, Automation and Test in Europe, 2008. DATE '08*, pages 1126–1127, March 2008.

[46] D. Park, C. Nicopoulos, J. Kim, N. Vijaykrishnan, and C. Das. Exploring fault-tolerant network-on-chip architectures. In *DSN*, pages 93–104, 2006.

[47] D. Park, C. Nicopoulos, J. Kim, N. Vijaykrishnan, and C. R. Das. Exploring fault-tolerant network-on-chip architectures. In *Proceedings of the International Conference on Dependable Systems and Networks*, DSN '06, pages 93–104, Washington, DC, USA, 2006. IEEE Computer Society.

[48] S. Paul, F. Cai, X. Zhang, and S. Bhunia. Reliability-driven ecc allocation for multiple bit error resilience in processor cache. *Computers, IEEE Transactions on*, 60(1):20–34, Jan 2011.

[49] M. Pirretti, G. M. Link, R. R. Brooks, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin. Fault tolerant algorithms for network-on-chip interconnect. In *IEEE Symp. VLSI*, 2004.

[50] V. Puente, J. A. Gregorio, F. Vallejo, and R. Beivide. Immunet: A cheap and robust fault-tolerant packet routing mechanism. In *ISCA*, 2004.

[51] D. Roberts, N. S. Kim, and T. N. Mudge. On-chip cache device scaling limits and effective fault repair techniques in future nanoscale technology. 2008.

[52] D. Rossi, N. Timoncini, M. Spica, and C. Metra. Error correcting code analysis for cache memory high reliability and performance. In *DATE*, pages 1620–1625. IEEE, 2011.

[53] S. Shamshiri, A.-A. Ghofrani, and K.-T. Cheng. End-to-end error correction and online diagnosis for on-chip networks. In *Test Conference (ITC), 2011 IEEE International*, pages 1–10, Sept 2011.

[54] S. Shamshiri, A.-A. Ghofrani, and K.-T. Cheng. End-to-end error correction and online diagnosis for on-chip networks. In *Test Conference (ITC), 2011 IEEE International*, pages 1–10, Sept 2011.

[55] C. W. Slayman. Cache and memory error detection, correction, and reduction techniques for terrestrial servers and workstations. *IEEE Transactions on Device and Materials Reliability*, 5(3):397–404, 2005.

[56] T. Thomas and B. Anthony. Area, performance, and yield implications of redundancy in on-chip caches. In *Computer Design, 1999. (ICCD '99) International Conference on*, pages 291–292, 1999.

[57] Y. Wang, L. Zhang, Y. Han, H. Li, and X. Li. Address remapping for static nuca in noc-based degradable chip-multiprocessors. In *Dependable Computing (PRDC), 2010 IEEE 16th Pacific Rim International Symposium on*, pages 70–76, Washington DC, USA, Dec 2010. IEEE.

[58] C. Wilkerson, A. R. Alameldeen, Z. Chishti, W. Wu, D. Somasekhar, and S.-l. Lu. Reducing cache power with low-cost, multi-bit error-correcting codes. In *Proceedings of the 37th Annual International Symposium on Computer Architecture*, ISCA '10, pages 83–93, New York, NY, USA, 2010. ACM.

[59] C. Wilkerson, A. R. Alameldeen, Z. Chishti, W. Wu, D. Somasekhar, and S.-l. Lu. Reducing cache power with low-cost, multi-bit error-correcting codes. In *Proceedings of the 37th Annual International Symposium on Computer Architecture*, ISCA '10, pages 83–93, New York, NY, USA, 2010. ACM.

[60] C. Wilkerson, H. Gao, A. R. Alameldeen, Z. Chishti, M. Khellah, and S.-L. Lu. Trading off cache capacity for reliability to enable low voltage operation. In *Proceedings of the 35th Annual International Symposium on Computer Architecture*, ISCA '08, pages 203–214, Washington, DC, USA, 2008. IEEE Computer Society.

[61] P. M. Yaghini, A. Eghbal, S. Asghari, and H. Pedram. Power comparison of an asynchronous and synchronous network on chip router. *Computer Conference*, pages 242–246, 2009.

[62] P. M. Yaghini, A. Eghbal, and N. Bagherzadeh. A gals router for asynchronous network-on-chip. In *Proceedings of International Workshop on Manycore Embedded Systems*, page 52. ACM, 2014.

[63] P. M. Yaghini, A. Eghbal, and N. Bagherzadeh. On the design of hybrid routing mechanism for mesh-based network-on-chip. *Integration, the VLSI Journal*, 50:183–192, 2015.

[64] P. M. Yaghini, A. Eghbal, M. Khayambashi, and N. Bagherzadeh. Coupling mitigation in 3d multiple-stacked devices. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 23(12):2931–2944, 2015.

[65] P. M. Yaghini, A. Eghbal, H. Pedram, and H. R. Zarandi. Investigation of transient fault effects in an asynchronous noc router. In *Parallel, Distributed and Network-Based Processing (PDP), 2010 18th Euromicro International Conference on*, pages 540–545. IEEE, 2010.

[66] P. M. Yaghini, A. Eghbal, H. Pedram, and H. R. Zarandi. Investigation of transient fault effects in synchronous and asynchronous network on chip router. *J. Syst. Archit.*, 57(1):61–68, Jan. 2011.

[67] P. M. Yaghini, A. Eghbal, S. S. Yazdi, and N. Bagherzadeh. Accurate system-level tsv-to-tsv capacitive coupling fault model for 3d-noc. In *Proceedings of the 9th International Symposium on Networks-on-Chip*, page 3. ACM, 2015.

[68] P. M. Yaghini, A. Eghbal, S. S. Yazdi, N. Bagherzadeh, and M. M. Green. Capacitive and inductive tsv-to-tsv resilient approaches for 3d ics. *IEEE Transactions on Computers*, 65(3):693–705, March 2016.

[69] P. M. Yaghini, H. R. Zarandi, A. Eghbal, A. Jafarzadeh, and S. Eskandari. An investigation of fault tolerance behavior of 32-bit dlx processor. In *Dependability, 2009. DEPEND'09. Second International Conference on*, pages 93–98. IEEE, 2009.