Modeling and Analysis of Fault-tolerant Distributed Memories for Networks-on-Chip

Abbas BanaiyanMofrad and Nikil Dutt
Center for Embedded Computer Systems
University of California, Irvine, CA, USA
{abanaiya, dutt}@uci.edu

Gustavo Girão
Federal University of Rio Grande do Sul
Institute of Informatics, Porto Alegre, Brazil
ggbsilva@inf.ufrgs.br

Abstract— **Advances in technology scaling increasingly make Network-on-Chips (NoCs) more susceptible to failures that cause various reliability challenges. With increasing area occupied by different on-chip memories, strategies for maintaining fault-tolerance of distributed on-chip memories become a major design challenge. We propose a system-level design methodology for scalable fault-tolerance of distributed on-chip memories in NoCs. We introduce a novel reliability clustering model for fault-tolerance analysis and shared redundancy management of on-chip memory blocks. We perform extensive design space exploration applying the proposed reliability clustering on a block-redundancy fault-tolerant scheme to evaluate the tradeoffs between reliability, performance, and overheads. Evaluations on a 64-core chip multiprocessor (CMP) with an 8x8 mesh NoC show that distinct strategies of our case study may yield up to 20% improvements in performance gains and 25% improvement in energy savings across different benchmarks, and uncover interesting design configurations.**

I. INTRODUCTION

Technology scaling has an increasing impact on the resilience of CMOS circuits [1], resulting in a host of reliability challenges such as manufacturing defects, wear-out, and parametric variations [2]. For example, effects such as process variation (PV) and negative bias temperature instability (NBTI) are increasingly threatening the reliability and lifetime of emerging NoC-based multi/many-core processors [6]. Beside process variation, variation in voltage and temperature and manufacturing defects coupled with voltage/frequency scaling make systems more vulnerable to both permanent and transient faults [3]. By increasing the number, amount, and hierarchy of on-chip memory blocks in multicore processors, reliability of the memory architecture becomes more challenging in the design of such systems [4]. To overcome such reliability challenges, we need to develop new approaches that integrate reliability strategies at multiple levels of the design hierarchy, and which take into account power, performance, cost, as well as user requirements.

Many research efforts have already investigated fault-tolerant schemes for NoC architectures [5][6], with a primary focus on fault-aware routing, reliable communication, and fault-tolerant routers. However, research on coupling memory and NoC reliability is still in its infancy [3]. Indeed, there are few works studying the reliability of on-chip memories at the network level [7]. On the other hand, while fault modeling [21], fault-tolerance analysis [19], and reliability modeling and analysis [18][20] of NoCs has been studied extensively, reliability modeling and analysis of memory sub-system in NoCs has not received much attention.

There is a large body of previous work on fault-tolerant design of on-chip cache memories, mostly proposed for single core processors [8][9]. However, they face severe limitations when they are applied to emerging NoC-based multi-core/many-core architectures with distributed on-chip memories where the number of access points (cores) and memory banks are numerous, access latencies are not unified, interconnect backbone affects the reliability, and both memory and redundancy are shared among all cores. Unfortunately, existing efforts have not addressed the need for a system-level framework that explicitly models distributed memories, analyzes redundancy organization, and which supports exploration of reliable distributed on-chip memories for emerging multi/many-core architectures. Our paper addresses these drawbacks through the following **main contributions**:

- We propose a system-level design methodology for scalable fault-tolerance of distributed on-chip memory blocks in NoC architectures.
- We introduce a novel reliability clustering model for efficient fault-tolerance analysis and shared redundancy management of on-chip memory blocks. Each cluster represents a group of cores that have access to shared redundancy resources for protection of their memory blocks.
- We perform extensive design space exploration applying the proposed reliability clustering on a block-redundancy fault-tolerant scheme to evaluate the tradeoffs between reliability, performance, and overheads.

We believe that our proposed design methodology will engender NoC architectures capable of efficiently responding to the multiple challenges of increasing fault rates, variation in fault behaviors, local interconnects, non-uniform memory access latency, limited shared redundancy, and susceptibility to transient variations.

II. EXEMPLAR NOC ARCHITECTURE

To illustrate our approach, we experiment on an exemplar tiled NoC-based CMP, where each tile comprises a processor core, private L1 data and instruction caches, a shared L2 cache bank, and network router/switch. Tiles are interconnected as a 2-D mesh via a network-on-chip infrastructure. Figure 1 shows our baseline 64-core CMP with an 8x8 mesh NoC. Based on our cache organization, the L2 bank is a portion of the larger distributed shared last-level cache (LLC). The baseline design

978-3-9815370-0-0/DATE13/©2013 EDAA

assumes a Non-Uniform Cache Architecture (NUCA) [10] for LLC. A directory-based protocol is implemented in order to maintain cache coherence.

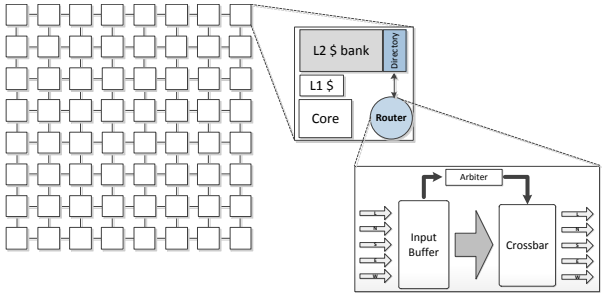

Fig. 1. Baseline architecture.

III. System-level Fault-tolerance Modeling

A methodology for fault-tolerance of distributed memory systems for scalable NoC platforms faces several inter-related challenges, requirements, and features: communication must be localized to reduce interconnect overheads; fault-tolerance schemes need to be topology-aware, and must organize and distribute redundancy based on criticality of communication paths; redundancy needs to be managed as a shared resource across the platform; and fault-tolerance should be addressed at higher design abstraction levels to gain more leverage over its effects on performance, cost, power, etc.

To meet these needs, we develop analytical models and a system-level methodology for fault-tolerant design of distributed on-chip cache memories for a tiled NoC-based CMP. This paper focuses on shared L2 banks that need to be protected. However, our approach can be applied to any form of on-chip memory architecture in NoCs that need reliability. To estimate design parameters we develop analytical models include a Reliability model, Access latency model, and Area overhead model. Using these models, we can analyze the trade-offs between yield, performance, and energy/area overheads of a fault-tolerant design of on-chip memory subsystem. Due to lack of space, we describe details of these models in our technical report [23]. Next, we present a novel reliability clustering concept for scalable, modular, and efficient design and implementation of fault-tolerant schemes applied to protect the memory blocks.

A. Reliability Clustering

To model fault-tolerance and organize redundancy we divide the whole NoC into *clusters* comprised of multiple groups of tiles. Each cluster is a subsection of the base NoC with the same topology but in a smaller group of tiles. The clustering is used to partition redundancy sharing among the tiles inside the cluster. In each cluster, some specific tiles (e.g., center tiles) contain the redundancy used for fault-tolerance of all tiles inside the cluster; these are labeled as redundancy nodes. These redundancy nodes can be used flexibly to accommodate different fault-tolerance schemes and varying forms of redundancy, such as redundant rows/columns/blocks; exploiting sections of available clear/faulty blocks as redundancy; and ECC codes. Furthermore, we leverage the available interconnect backbone to support implementation of a variety of modular, scalable, and efficient fault-tolerance schemes. For instance, in our exemplar tiled NoC architecture, we modify the direct router connected to the redundancy nodes to support our selected fault-tolerant scheme.

Because such clustering organization is independent of a particular topological structure, various physical topologies can serve as fixed-silicon but dynamically reprogrammable reliable multicore clusters on top of NoC platforms. Meanwhile, the inherent reconfigurability allows customizing clusters according to not only the clustered and varying fault rates but also to application communication patterns and resilience needs. Therefore, clusters can be defined statically or dynamically during runtime. For the sake of simplicity, here we consider static clustering.

Our mesh-based NoC is composed of N nodes, C clusters, with each cluster containing a dxd mesh of tiles (d = cluster dimension size). The size and number of clusters can affect the yield, overhead, and network latency. The cluster dimension size (d) would determine the upper bound on latency of fault-tolerant LLC accesses. Depending on the shape and size of each cluster, number of clusters, amount of redundancy, and distribution of redundancy among clusters, we can explore different design strategies which meet various design constraints.

IV. Evaluation

To illustrate the flexibility and utility of our exploration methodology, we outline a sample design space exploration study using the exemplar NoC platform.

A. Experimental Setup

We use SoCIN, a cycle-accurate SystemC model of a Mesh-based NoC [11] that uses Wormhole switching with a 4-phit buffer size and an XY routing to avoid deadlocks. SoCIN router contains input buffers, the arbiter, the crossbar, and the links as depicted in Fig. 1. Each router is connected to a 1MB L2 cache bank. All L2 cache banks share the same address space of 64 MB LLC. Additionally, for each router, there is a module generating data and instruction requests based on memory traces obtained from a Simics [12] simulation using 64 sparcV8 as cores. A workload of parallel programs with very distinct behaviors is created using benchmarks from SPLASH2 [13], PARSEC [14], and a parallel version of MiBench [15] suites. With these memory traces as input to our framework, we are able to extract performance and energy results. The performance results are the total number of cycles to execute all 64 parallel tasks (for each application) and the energy results are obtained from Cacti 6.5 [16] for the memory subsystem and from Orion 2.0 [17] for the network-on-chip. Table I summarizes the experimental setup of our architecture.

For the sake of this study, faulty LLC banks are modeled randomly since SRAM cell faults occur as random events due to the major contribution of the random dopant fluctuation to the process variation [8]. Based on the results of a recent work [1], the predicted probability of failure for SRAM cells can be up to 2.6e-4. Here, since our case study is a block-redundancy fault-tolerant scheme and the analysis is at system level, we model failures at the block level. Based on that, the probability of block failure would be up to 7e-2. To enable a fair analysis during our evaluation, we consider 8 fault rates ranging from 1e-3 to 7e-2. To determine the amount of redundancy based on our reliability model, to keep the reliability more than 99%, the redundancy (R) should be more than 2%. On the other hand, to

keep effective yield more than 95%, we should keep the redundancy less than 5%. Thus we consider redundancy for any value ranging from 2% to 5%.

TABLE I. EXPERIMENTAL SETUP

Processor Cores	64 SPARC V8
L1 Inst./Data Cache	2 Banks, 32KB each, 4-Way, 32B block, 2 cycle
L2 Cache (shared LLC)	64 Banks, 1MB each, 8-Way, 64B block, 10 cycle
Memory Latency	250 cycles
Interconnect	NoC of 2D mesh (8x8 for 64 banks) 32-byte links (2 flits per memory access), 1-cycle link latency, 2-cycle router, XY routing, 4 phit buffer size
Integrated Technology	45 nm

B. *Design Space Exploration Studies*

We evaluate the impact of system-level reliability clustering on performance, energy, and area overhead of the system using simulation runs of the experimental platform described earlier. For different system configurations we change one design parameter such as fault rate or amount of redundancy and study its effect on different design metrics of the memory subsystem. We consider the design space at two levels: cluster-level and system-level.

1) Cluster-level (Intra-cluster): Here, we study the effect of redundancy distribution (number and location of redundancy nodes) on system design metrics. In this set of results, we consider the whole NoC as one cluster, fix either the amount of redundancy or fault rate, and change the redundancy distribution. In these experiments we explore the redundancy organization by changing either the number of nodes that contain redundancy (redundancy nodes) or their location. Redundancy node distribution can be studied in two directions, ranging from central nodes to all outward nodes or ranging from corner nodes to all inward nodes. Redundant elements are spread equally among all nodes which contain redundancy. Proposed distributions are selected based on a regular and scalable pattern which is independent of the size of the cluster. Figure 2.a presents some possible distributions for our base architecture with four redundancy nodes. Here, for each redundancy distribution we have other variations of the distribution by spreading the nodes from the center (Option 1) towards the corners (Option 4). Note that the label N.X means that the configuration has N nodes with redundancy and X representing the distribution option. Higher values of X represent redundancy nodes that are closer to the corners.

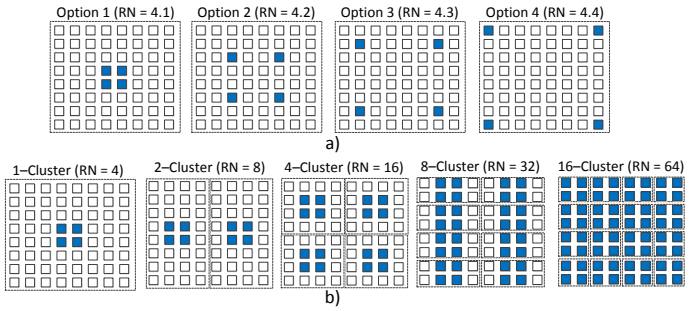

Fig. 2. Possible configuration patterns in a) cluster-level, b) system-level, with four redundancy nodes per cluster.

2) System level (Inter-cluster): Here, we investigate the effect of node clustering and shared redundancy management among clusters on system design metrics. In this set of results, we change the number and size of clusters while fixing the total redundancy in the system. We select the size and number of clusters based on a regular and scalable pattern. Redundancy is spread equally among all clusters and all redundancy nodes inside each cluster. Here, we put the redundancy nodes in the center of each cluster. The intuition behind this approach is to guarantee that the redundancy nodes are always surrounded by a certain number of cores. This has the goal of not only minimizing the average distance between the cores and the redundancy nodes but also minimizing the variance in the average distance for each case. Figure 2.b presents some possible clustering and distribution of redundancy for our base architecture with four central redundancy nodes per cluster. Here we illustrate sample distributions for 1, 2, 4, 8, and 16 clusters with 4, 8, 16, 32, and redundancy nodes, respectively.

V. SAMPLE EXPLORATION RESULTS

We summarize the normalized performance/energy results of the block-redundancy scheme using the proposed clustering methodology, with respect to a baseline system without fault-tolerance support (i.e., where access to a faulty memory address results in an on off-chip memory access). Detailed experimental results are in our technical report [23].

A. *Cluster-level Results*

Fig. 3.a shows the gains of performance – the normalized execution time to the baseline – for each configuration shown in Fig. 2.a with 3% of memory redundancy and also 3% of block fault rate in the system. Note that 3% block fault rate means that 3% of all memory blocks in the system are faulty. This figure shows the susceptibility of performance gains varying from one application to another. While the performance improvements vary across most applications, as expected, configurations with more redundancy nodes (e.g., configuration 16.2) present better results.

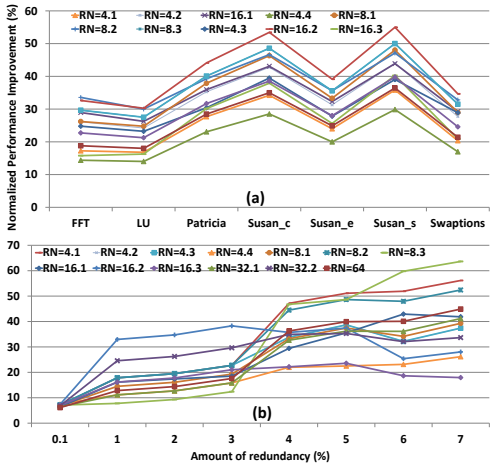

Fig. 3. Normalized performance improvement of different cluster-level configurations across different a) benchmarks, b)amount of redundancy.

In another experiment we evaluate the effect of different amounts of redundancy for cluster level while setting the block fault rate to 3% (Fig. 3.b). Note that it is not sufficient to have redundancy greater than the fault rate, since the faults may not be evenly distributed, and other factors (e.g., routing and contention) may affect performance. Indeed, Fig. 3.b shows an

inversion from 4% of redundancy memory: configurations that were inferior in the previous experiments start to present better performance and those that were previously superior now appear to become worse. This suggests that at some point when the total of redundancy memory is higher than the amount of faulty blocks in the system, it is better to have the redundancy nodes in the corners. This could be due to the fact that as the XY routing tends to concentrate the load in the center of the NoC [22], these packets for redundancy memory requests (which now occur less frequently) may generate less contention if avoiding the middle of the NoC.

B. System-level Results

For the inter-cluster case we put the redundancy nodes in the center of cluster and change the size and number of clusters. Similar to Fig. 3.a, Fig. 4 presents the results at the system-level in terms of performance gains when fixing the redundancy memory at 3% and the block fault rate also at 3%. In this case, for all configurations except 16-cluster, there is not much variation from one application to another. This is a configuration that spreads the redundancy data equally throughout all the nodes and therefore is more susceptible to different memory access rates. For lower memory accesses it works well because there are few redundancy memories per node across all nodes. For high memory accesses it may not work well since it has a higher chance where the redundancy nodes may be too far away.

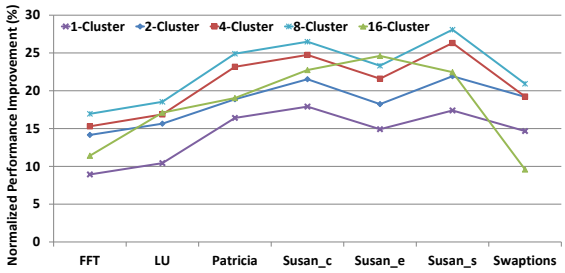

Fig. 4. Normalized performance improvement of different system-level configurations across different benchmarks.

In another set of experiments we run both cluster-level and system-level configurations across all benchmarks when using 3% of memory redundancy and changing the block fault rate in the system. We note that configurations using too many redundancy nodes do not present good results. This may suggest that the best distribution must not have a small amount of redundant memory per node because this will spread the redundant area too much, creating a higher average NoC distance between the nodes.

Overall, by looking at the results on both sets of experiments, we observe that good decisions on redundancy organization or selecting proper cluster-level and system-level configurations can lead up to 20% improvement in normalized performance gains over the baseline. This is a relevant differential since it incurs no increase in overhead and only requires changing the redundancy organization.

C. Energy Results

We also studied the energy saving (normalized to baseline) for both sets of experiments, as detailed in our technical report [23]. Overall, depending on different configurations the normalized energy savings can be improved up to 24% in the cluster-level approach and 18% in the system-level approach for different amounts of redundancy.

VI. Conclusion

In this paper, we proposed a system-level design methodology for scalable fault-tolerance of distributed on-chip memories in NoCs. We introduced a new concept of reliability clustering for efficient redundancy management and fault-tolerant design of memory blocks. Experimental results on an exemplar 64-core CMP with an 8x8 mesh NoC show that distinct design strategies or reliability clustering configurations of a block-redundancy scheme may yield up to 20% improvement in normalized performance gain and up to 24% in normalized energy gain, uncovering many interesting design configurations. Future work will explore implementation and architecture definition to support the proposed reliability clustering approach.

Acknowledgment

This work was partially supported by NSF Variability Expedition Grant Number CCF-1029783. Also, this work was partially supported by CNPq Brazillian agency.

References

[1] S. R. Nassif, et al., "A resilience roadmap," *in Proc. DATE*, 2010.
[2] S. Borkar, "Designing reliable systems from unreliable components: The challenges of transistor variability and degradation," *IEEE Micro*, 2005.
[3] R. Marculescu, et al., "Outstanding Research Problems in NoC Design: System, Microarchitecture, and Circuit Perspectives" *IEEE TCAD*,2009.
[4] N. Aymerich, et al., "New reliability mechanisms in memory design for sub-22nm technologies," *in Proc. IOLTS*, 2011.
[5] D. Park, et al., "Exploring Fault-Tolerant Network-on-Chip Architectures," *in Proc. DSN*, 2006.
[6] X. Fu, et al., "Architecting reliable multi-core network-on-chip for small scale processing technology," *in Proc. DSN*, 2010.
[7] F. Angiolini, et al., "Reliability Support for On-Chip Memories Using Networks-on-Chip," *in Proc. ICCD*, 2006.
[8] A. Agarwal, et al., "A process-tolerant cache architecture for improved yield in nanoscale technologies," *IEEE TVLSI*, 2005.
[9] S. Ozdemir, D. Sinha, G. Memik, J. Adams, and H. Zhou, "Yield-aware cache architectures," *in Proc. MICRO*, 2006.
[10] C. Kim, D. Burger, and S. W. Keckler, "An adaptive, non-uniform cache structure for wire-delay dominated on-chip caches," *in ASPLOS*, 2002.
[11] C.A. Zeferino and A.A. Susin, "SoCIN: A Parametric and Scalable Network-on-Chip," *in Proc. SBCCI*, 2003.
[12] P.S. Magnusson, et al., "Simics: A Full System Simulation Platform," *IEEE Computer*, vol. 35, no. 2, pp. 50-58, 2002.
[13] S.C. Woo, et al., "The SPLASH-2 Programs: Characterization and Methodological Considerations," *in Proc. ISCA*, 1995.
[14] C. Bienia, et al,"The PARSEC benchmark suite: characterization and architectural implications," *in Proc. PACT*, 2008.
[15] S.M.Z. Iqbal, Y. Liang, H. Grahn, "ParMiBench: An Open Source Benchmark for Embedded Multiprocessor Systems,"*in Proc CAL*, 2010.
[16] N. Muralimanohar, R. Balasubramonian, and N. Jouppi, Cacti 6.5, In HP Laboratories, *Technical Report*, 2009.
[17] A.B. Kahng, et al., ORION 2.0: a fast and accurate NoC power and area model for early-stage design space exploration," *in Proc. DATE*, 2009.
[18] J. Kim, et al., "Design and analysis of an NoC architecture from performance, reliability and energy perspective," *in Proc. ANCS*, 2005.
[19] T. Lehtonen, P. Liljeberg and J. Plosila, "Fault Tolerance Analysis of NoC Architectures", *in Proc. ISCAS*, May 2007.
[20] A. Dalirsani, et al., "An Analytical Model for Reliability Evaluation of NoC Architectures," *in Proc. IOLTS*, 2007.
[21] K. Aisopos, O. Chen, and L.-S. Peh, "Enabling System-Level Modeling of Variation-Induced Faults in Networks-on-Chips," *in DAC*, 2011.
[22] M. Dehyadgari, et al., "Evaluation of Pseudo Adaptive XY Routing Using an Object Oriented Model for NOC," *in Proc. ICM*, 2005.
[23] A. BanaiyanMofrad et al., "Analyzing and Exploring Fault-tolerant Distributed Memories for NoCs," *UCI CECS TR 12-15*, 2012.