Center for Embedded Computer Systems
University of California, Irvine

Analyzing and Exploring Fault-tolerant Distributed Memories for NoCs

Abbas BanaiyanMofrad[1], Gustavo Girão[2], and Nikil Dutt[1]

[1]Center for Embedded Computer Systems
University of California, Irvine
Irvine, CA 92697-2620, USA

[2]Federal University of Rio Grande do Sul
Institute of Informatics, Porto Alegre, Brazil

{abanaiya, dutt}@uci.edu, ggbsilva@inf.ufrgs.br

CECS Technical Report 12-15
December 22, 2012

Analyzing and Exploring Fault-tolerant Distributed Memories for NoCs

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 [7]. Beside process variation, variation in voltage and temperature and manufacturing defects coupled with voltage/frequency scaling makes systems more vulnerable to both permanent and transient faults [3]. Meanwhile, major variances in fault behavior are expected, as faults are highly sensitive to temperature, background noise, voltage changes, and other factors [4]. By increasing the number, amount, and hierarchy of on-chip memory blocks in such systems, reliability of the memory architecture becomes more challenging in the design of such systems, for both late CMOS and emerging technologies [4][5]. 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 [6][7], 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 [8]. However, to the best of our knowledge, there have been no previous system-level efforts that model and analyze redundancy management and fault-tolerance of distributed on-chip memories in NoCs.

This paper makes 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 develop analytical models for analysis of reliability, latency, and area overhead leveraging reliability clustering, and use these for design space exploration of fault-tolerant distributed memories for NoCs.
- 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. *To the best of our knowledge, ours is the first work to comprehensively analyze and explore distributed memory reliability in NoC-based multicore architectures.*

II. Related Work

There is a large body of work addressing NoC reliability, from the perspectives of NoC interconnects [6][7], routing algorithms [10], communication infrastructure [9], and micro-architecture [8][11]. Park et al. [6] propose a set of techniques to protect against the most common sources of transient failures in on-chip interconnects including link errors, and single-event upsets within the router. Fu et al. [7] propose novel techniques at both intra-router and inter-router levels to mitigate the impact of PV and NBTI on NoC. A stochastic communication paradigm is proposed in [9] to provide fault-tolerant communication. Similarly, Puente et al. [10] explore

how NoC routing algorithms can route around faults and sustain network functionality in presence of faults.

While reliability modeling and analysis of NoCs has been studied extensively [23][24][25], reliability analysis of memory sub-system in NoCs has not received much attention. Kim et al. [23] present a queuing-theory-based model for evaluating the performance and energy behavior of on-chip networks. In [24] a fault-tolerant analysis of different mesh NoC architectures including the topology, the router structure, and the number of network interfaces is presented. Dalirsani et al. [25] propose an analytical model to assess the reliability factor of a mesh NoC against transient faults effects on switches and routing algorithms. Recently, Yamamoto et al. [26] proposed an architecture level unified reliability evaluation methodology to explore the impact of NoC router layout on the system's lifetime. Aisopos et al. [27] proposed a circuit-level fault modeling tool that can be integrated in system-level network simulators to capture runtime PV-induced faults in the NoC.

In the context of memories, there is a large body of previous work on fault-tolerant design of on-chip cache memories, mostly proposed for single core processors [12][13][14]. 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 probably shared among all cores. Since the redundancy is limited and shared in such architectures, effective redundancy organization can greatly affect various design metrics such as reliability, performance, power, and area overhead. The efficient use of an interconnect backbone in NoCs helps designers develop more powerful fault-tolerant schemes for protection of distributed memories using its high flexibility to manage redundancy among different banks. 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 -- the key contribution of our work.

III. System-level Fault-tolerance Modeling

A methodology for fault-tolerance of distributed memory systems for scalable NoC platforms faces several challenges, requirements, and features:

- Wiring overhead and interconnect delay are dominant challenges; communication becomes extremely expensive and must be strictly limited to localized interconnect.
- In contrast with traditional fault-tolerance schemes, new approaches must be highly topology-aware. This limits the physical placement and access policy of redundant elements as well as available strategies to verify the correctness of memory blocks that use them.
- Fault-tolerant schemes must consider not only the amount of redundancy, but also how it is organized and distributed to deal with increasing fault rates and criticality of communication paths.
- Redundancy must be managed flexibly as a shared (rather than a fixed) resource to deal with variable fault rates and high rates of failure. The shared redundancy must be configurable and must exploit the underlying interconnect.
- Low cost fault-tolerance schemes must exploit and leverage a system's "inherent" redundancy.
- Fault tolerance should be addressed at higher design abstraction levels to gain more control over design issues such as communication cost, limited redundancy, and variance in fault behaviors. Typically, the main cost for fault tolerance is the overhead of redundant elements. However, the interplay between redundancy and interconnect complexity is a major design issue in reliable NoCs.

In the following, we present analytical models and a system-level methodology for fault-tolerant design of distributed on-chip cache memories for a tiled NoC-based CMP. Note that for this work we focus on shared L2 banks as the memory blocks which needs to be protected. However, our approach can be applied to any form of on-chip memory architecture in NoCs that need reliablity. 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. Our analytical models cover 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 distributed memories in NoCs.

A. 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 assumes a Non-Uniform Cache Architecture (NUCA) [15] for LLC. A directory-based protocol is implemented in order to maintain cache coherence.

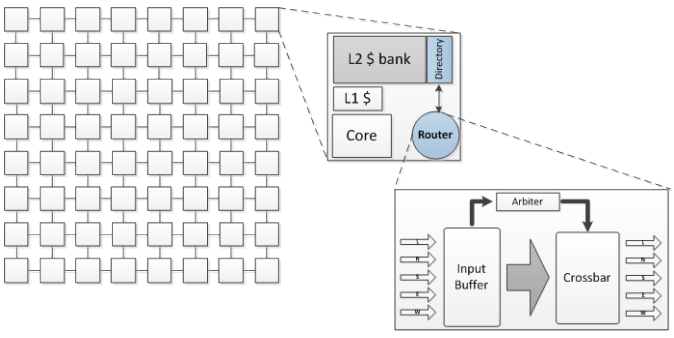

Fig. 1. Baseline Architecture.

B. 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.

C. Reliability Model

To estimate the yield and reliability of the cache, we model the probability of having an operational cache using a fault tolerant scheme. Here, we consider a block-redundancy fault-tolerant scheme as our case study. Let's assume an *n* way set-associative cache with *m* sets, *k* bits per block with a fault probability P_{fbit}. We consider a block as faulty if it contains at least one faulty bit. So the probability of failure for each block that has at least one faulty bit is:

$$P_{\text{fblock}} = 1-(1-P_{\text{fbit}})^{k} \quad (1)$$

We consider a set as faulty if at least half of its blocks are faulty. So the probability of failure for each set is:

$$P_{\text{fset}} = \sum_{n/2}^{n} \binom{n}{i} P_{\text{fblock}}^{i} \quad (2)$$

Let us assume that there are *R* redundant rows. Hence, for a good cache, among rows, at most *R* rows can be faulty. Hence, the probability that a cache is operational is:

$$P_{\text{Cache}} = \sum_{i=0}^{R} \binom{m+R}{i} P_{\text{fset}}^{i} (1 - P_{\text{fset}})^{m+R-i} \quad (3)$$

Here, we consider yield of a single cache (Y) equal to probability of being operational. So the efficient yield of the cache [13] would be:

$$Y_{eff} = \frac{m}{m+R}\, Y \quad (4)$$

D. Latency Model

We model LLC access latency as the time between issuing a LLC request and receiving a fault-free block. This timing depends on the access latency of requested block, redundancy access latency, and delay of applying the fault-tolerant method. Thus:

$$D_{avg} = max(D_M, D_R) + D_F \quad (5)$$

Here, D_{avg} is the average delay of accessing redundancy from all nodes in a cluster. D_M , D_R , and D_F are average delay of accessing the requested memory block, accessing the redundancy, and fault repair operation, respectively.

We consider D_M as the latency of accessing a memory block (here LLC block) from the requesting core which given by:

$$D_M = D_{MAccess} + D_{MNet} \quad (6)$$

where, $D_{MAccess}$ is the L2 access latency and D_{MNet} is the latency to access the requested block throughout the network. Note that in case the requested block is in the local L2 bank, this latency is zero.

We consider D_R as the latency of accessing the redundancy (here spare block) from the requesting core which given by:

$$D_R = D_{RAccess} + D_{RNet} \quad (7)$$

where, $D_{RAccess}$ □is the redundant element access latency and D_{RNet} is the latency to access the redundant element throughout the network.

E. Area Overhead Model

In this model we consider the area overhead of redundant elements in addition to the overhead of fault-tolerant scheme:

$$A = C \times (R_c + F_c) \quad (8)$$

Here, *A* is total area overhead of the system protection. R_C and F_C are the area overheads of redundant elements and fault-tolerant logic per cluster, respectively. The overhead of redundancy per cluster equals the total redundancy divided by the number of clusters. The overhead of fault-tolerant logic is proportional to the number of redundancy nodes (N_R) and is given by:

$$F_c = F_n \times N_R \quad (9)$$

Here, F_n is the overhead of the fault-tolerant scheme logic added to each node that contains redundancy. The F_n overhead depending on the fault-tolerant scheme, can be either fixed or depends on other parameters like cluster size or amount of redundancy.

IV. Sample Design Space Exploration

To illustrate the flexibility and utility of our exploration methodology, we outline a sample design space exploration study using the exemplar NoC platform, and present results of two exploration studies with redundancy sharing at the intra-cluster level and inter-cluster level configurations.

A. Methodology

We use SoCIN, a cycle-accurate SystemC model of a Mesh-based NoC [16] 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. Also, the flow control in this NoC is handshake based. 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 [17] simulation using 64 sparcV8 as cores. Table I summarizes the experimental setup of our architecture.

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
Frequency	1.0 GHz
Integrated Technology	45 nm

A workload of parallel programs with very distinct behaviors is created using benchmarks from SPLASH2 [18], PARSEC [19], and a parallel version of MiBench [20] suites. Using the Simics simulator, we extract all memory requests (data and instructions) by executing these applications in parallel. 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 applications and the energy results are obtained from Cacti 6.5 [21] for the memory subsystem and from Orion 2.0 [22] for the network-on-chip.

For the sake of this study, faulty LLC banks are modeled randomly since SRAM cell faults occur as random events. These random faults are due to the major contribution of the random dopant fluctuation to the process variation [13]. A recent work in resilience roadmap reports the predicted probability of failure for inverters, latches, and SRAM cells as a function of technology node [1]. Based on their results the 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 the probability of operation of cache in Eq. 2, if we want 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%.

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: intra-cluster and inter-cluster.

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 which 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.

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.

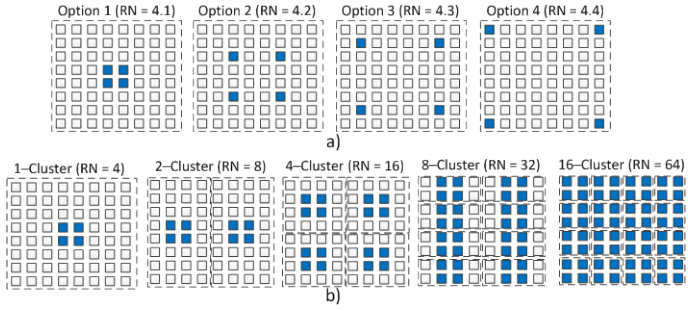

Fig. 2. Possible configuration patterns in a) cluster-level, b) system-level, with four redundancy node per cluster

V. SAMPLE EXPLORATION RESULTS

The results in this section represent 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. This means that any access to a faulty memory address implies on off-chip memory access to retrieve the data.

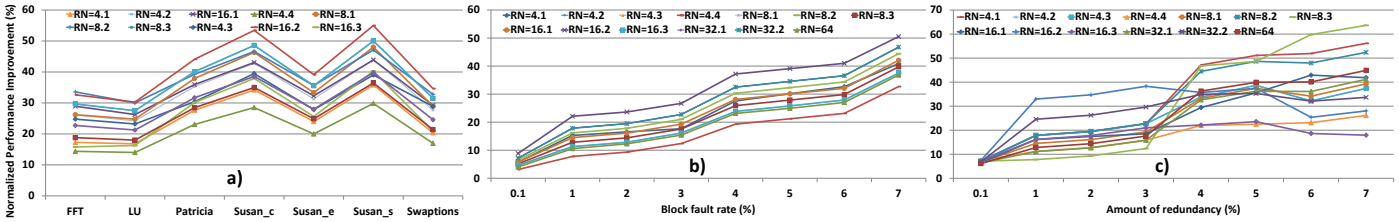

Fig. 3. Normalized performance improvement of different cluster-level configurations across different a) benchmarks, b)fault rates, c)amount of redundancy

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. Some configurations present better result than others depending on the application but for most applications, in this particular case of 3% of block fault rate and memory redundancy, the configuration 16.2 seems to be the best option for this particular amount of redundancy and block fault rate.

Figure 3.b presents the average performance results over 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.

For a second experiment we evaluate the effect of different amounts of redundancy for cluster level (Fig. 3.c). In this experiment, we fix the block fault rate to 3%. It is important to clarify that even with an amount of redundancy higher than the block fault rate, the location of the faults is not equally distributed and some portions of the system can, in the worst case, contain all the faulty blocks. Hence, there is still the occasional necessity to reach for redundancy memory on other nodes and eventually, nodes that are far way. In Fig. 3.c we see 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 [28], 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.a presents the results at the inter-cluster level in terms of performance gains when fixing the redundancy memory at 3% and the block fault rate also at 3%. In this particular case there is not much variation from one application to another, but it is possible to see a lot of variation for the 64 configuration. 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 access it works well because there are few redundancy memories per node across all nodes. For high memory access it may not work well since it has a higher chance to have a situation where the redundancy node may be too far away.

Figure 4.b presents performance results for these configurations with a total of 3% of redundancy memory while changes the block fault rate, ranging from 0.1% to 7%. The figure shows that the 8-cluster configuration has the better results. This is due to the fact that this configuration presents a low average distance between cores and redundancy area and also because this average distance does not vary too much considering all cores.

Figure 4.c presents results regarding the same exploration presented in Fig. 3.c but this time with this set of system level configurations. Similarly, we observe that a few changes occur after the point of 3% of redundancy memory but differently from the case of cluster level configurations, there is no clear change of scenario, i.e. the better result remain unchanged. This could be due to the fact that in this set of configurations there is no configuration that places the redundancy nodes in the edges of the NoC.

Overall, by looking at the results on both sets of experiments, it is possible to see that a good decision on where to place the redundancy nodes in the NoC can lead to a performance improvement of 20% in some of the cases. This is a relevant differential since it incurs no increase in overhead and only requires changing the redundancy organization.

C. Energy Results

Figure 5 presents the energy results (normalized to baseline) for both sets of experiments fixing a fault-rate of 3% and varying the redundancy from 0.1% to 7%. The trends here are quite similar to the one presented in Figures 3.c and 4.c. The difference is that since off-chip memory accesses consume more energy than regular on-chip memory accesses, the penalty for not having a redundancy memory hits the baseline the hardest. The results presented here show the energy savings of the redundancy cluster approaches compared to this baseline. The energy saved reaches 47% in the cluster-level approach and 51% in the system-level approach.

In Fig. 5.a we present the results for the intra-cluster level approach. The energy consumption in this experiment for the most part follows the same trend as the performance results. Some situations can present better results depending on the average distance between the cores and the redundancy nodes (like RN=16.2). If better distributed, the energy consumption to

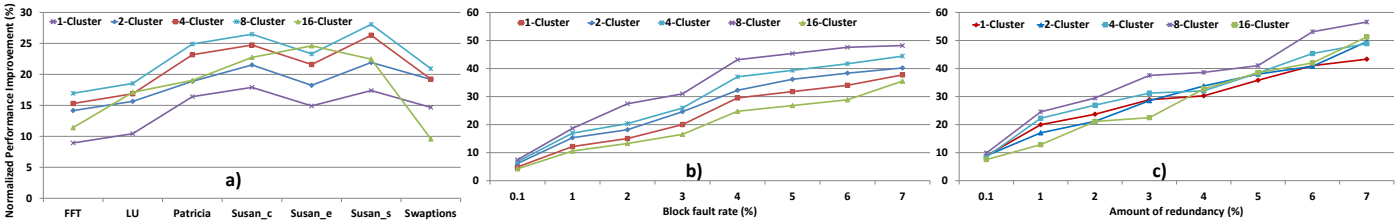

Fig. 4. Normalized performance improvement of different system-level configurations across different a) benchmarks, b)fault rates, c)amount of redundancy

reach the redundancy nodes will also be smaller. After the point where the amount of redundancy surpasses the block fault rate, some configurations that place the redundancy blocks far away from the center of the NoC present better results (like RN=8.3). This could be because of the tendency of XY routing to create more contention in the center of the NoC.

In Fig. 5.b the inter-cluster (system-level) approach energy savings are presented and although the trends are similar to the one presented in Fig. 4.c, the differences between the configurations with different number of clusters seem to be increased and therefore, the 8-cluster configuration presents the best results due to an even distribution of redundant blocks in the NoC without reducing the amount of redundancy per node too much. The 16-cluster results present a better result at 7% redundancy rate. This is due to the fact that in this system-level approach if the redundancy rate is much higher than the block fault rate, the amount of redundancy per node reaches a point where fewer and closer redundancy nodes are enough to cover the faults. With lower distance between the nodes, lower NoC energy is consumed.

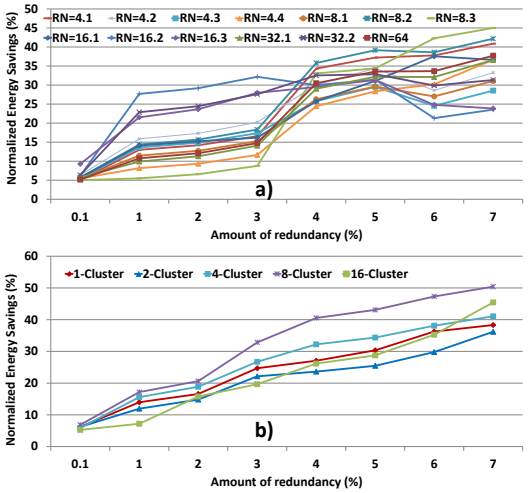

Fig. 5. Energy results across different amount of redundancy for (a) Intra cluster level, (b) Inter-cluster (system-level) configurations.

D. Area Overhead Analysis

For both set of intra-cluster level and inter-cluster (system-level) configurations, the area overhead based on Eq. 8 is proportional to the amount of redundancy in addition to fault-tolerance overhead per cluster (F_C). In case of fixed amount of redundancy, increasing the number of redundancy nodes would increase the F_C overhead based on Eq. 9. However, this increase in F_C contributes little to the total overhead in Eq. 8 and its effect is therefore negligible.

Our sample experimental results have shown that there is a large design space of alternatives and a need to explore the effects of different configurations.

VI. Conclusion

To construct reliable memory architectures in emerging multi/many-core NoC platforms, designers must consider the interconnect, distribute and utilize redundancy efficiently to alleviate the high cost of fault-tolerance schemes, employ a hierarchical set of fault-tolerance strategies, and create novel design paradigms that consider system-level issues. 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 of our block-redundancy scheme may present up to 20% of performance gains, and can uncover interesting design configurations. Future work will explore application-aware dynamic clustering of cores and shared memory redundancy for fault-tolerance.

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, N. Mehta, and Y. Cao, “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] W. Rao, et al., “Toward future systems with nanoscale devices: Overcoming the reliability challenge,” IEEE Computer Magazine, 2011.
[5] N. Aymerich, et al., “New reliability mechanisms in memory design for sub-22nm technologies,” In Proc. IOLTS, 2011.
[6] D. Park, et al., “Exploring Fault-Tolerant Network-on-Chip Architectures,” In Proc. DSN, 2006.
[7] X. Fu, et al., “Architecting reliable multi-core network-on-chip for small scale processing technology,” In Proc. DSN, 2010.
[8] F. Angiolini, et al., “Reliability Support for On-Chip Memories Using Networks-on-Chip,” In Proc. ICCD, 2006.
[9] P. Bogdan, et al., “Stochastic communication: A new paradigm for fault-tolerant networks-on-chip,” In Proc. VLSI Design, Feb. 2007.
[10] V. Puente, et al., “Immunet: A cheap and robust fault-tolerant packet routing mechanism,” In Proc. ISCA, Jun. 2004.
[11] J. Kim, et al., “A Gracefully Degrading and Energy-Efficient Modular Router Architecture for On-Chip Networks,” In Proc. ISCA, 2006.
[12] T. Thomas and B. Anthony, “Area, Performance, and Yield Implications of Redundancy in On-Chip Caches,” In Proc. ICCD, Feb. 1999.
[13] A. Agarwal, et al., “A process-tolerant cache architecture for improved yield in nanoscale technologies,” IEEE TVLSI, 2005.

[14] S. Ozdemir, D. Sinha, G. Memik, J. Adams, and H. Zhou, “Yield-aware cache architectures,” In Proc. MICRO, 2006.
[15] C. Kim, D. Burger, and S. W. Keckler, “An adaptive, non-uniform cache structure for wire-delay dominated on-chip caches,” In ASPLOS, 2002.
[16] C.A. Zeferino and A.A. Susin, “SoCIN: A Parametric and Scalable Network-on-Chip,” In Proc. SBCCI, 2003.
[17] P.S. Magnusson, et al., “Simics: A Full System Simulation Platform,” IEEE Computer, 35(2): 50-58, 2002.
[18] S.C. Woo, et al., “The SPLASH-2 Programs: Characterization and Methodological Considerations,” In Proc. ISCA, 1995.
[19] C. Bienia, et al,“The PARSEC benchmark suite: characterization and architectural implications,” In Proc. PACT, 2008.
[20] S.M.Z. Iqbal, Y. Liang, H. Grahn, “ParMiBench: An Open Source Benchmark for Embedded Multiprocessor Systems,”In Proc CAL, 2010.
[21] N. Muralimanohar, R. Balasubramonian, and N. Jouppi, Cacti 6.5, In HP Laboratories, Technical Report, 2009.
[22] 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.
[23] J. Kim, et al., “Design and analysis of an NoC architecture from performance, reliability and energy perspective,” In Proc. ANCS, 2005.
[24] T. Lehtonen, P. Liljeberg and J. Plosila, “Fault Tolerance Analysis of NoC Architectures", In Proc. ISCAS, May 2007.
[25] A. Dalirsani, et al., “An Analytical Model for Reliability Evaluation of NoC Architectures,” In Proc. IOLTS, 2007.
[26] A.Y. Yamamoto, et al., “Unified System Level Reliability Evaluation Methodology for Multiprocessor Systems-on-Chip,”in Proc IGCC, 2012.
[27] K. Aisopos, O. Chen, and L.-S. Peh, “Enabling System-Level Modeling of Variation-Induced Faults in Networks-on-Chips,” In DAC, 2011.
[28] M. Dehyadgari, et al., “Evaluation of Pseudo Adaptive XY Routing Using an Object Oriented Model for NOC,” In Proc. ICM, 2005.