Address Remapping for Static NUCA in NoC-based Degradable Chip-Multiprocessors

Ying Wang[1, 2], Lei Zhang[1], Yinhe Han[1], Huawei Li[1], Xiaowei Li[1]
[1]Key Laboratory of Computer System and Architecture
Institute of Computing Technology, Chinese Academy of Sciences, Beijing, P.R. China
[2]Graduate University of Chinese Academy of Sciences, Beijing, P.R. China
{wangying2009, zlei, yinhes, lihuawei, lxw}@ict.ac.cn

***Abstract*—Large scale Chip-Multiprocessors (CMPs) generally employ Network-on-Chip (NoC) to connect the last level cache (LLC), which is generally organized as distributed NUCA (non-uniform cache access) arrays for scalability and efficiency. On the other hand, aggressive technology scaling induces severe reliability problems, causing on-chip components (e.g., cores, cache banks, routers) failure due to manufacture defects or on-line hardware faults. Typical degradable CMPs should possess the ability to work around defects by disabling faulty components. For static NUCA architecture, when cache banks attached to a computing node are disabled, however, certain physical address sections will no longer be accessible. Prior approaches such as sets reduction introduced in Intel Xeon processor 7100 series enable turning off cache banks by masking certain sets bits in physical address1, which greatly wastes cache capacity. In this paper, we propose to tackle the above problem in a finer granularity to restrict the capacity loss in NUCA cache. Cache accesses to isolated nodes are redirected based on the utility-driven address remapping scheme that reduces data blocks conflicts in fault-tolerant shared-LLC. We evaluate our technique using GEMS simulator. Experimental results show that address remapping achieves significant improvement over the conventional cache sizing scheme.**

Keywords-CMP; NoC; NUCA; Address remapping; Fault-tolerant cache;

I. Introduction

As feature size continues to shrink, device parameters are expected to fluctuate significantly in future designs. Variations in critical process parameters [1], manufacture defects, and device aging [2] make the hardware failure emerge more frequently. Especially, the future processors are likely to integrate large numbers of cores and large capacity cache, which increases the manufacture difficulty and makes the whole system more vulnerable to hardware faults [3].

In current typical CMPs, non-uniform cache architecture (NUCA) designs effectively mitigate the intra-core communication delays, and allow free cache capacity sharing among different processor cores, thus NUCA is widely accepted as a way to organize the last level cache [4]. Huh et al. [5] concluded that the static NUCA cache with sharing degrees of two or four works best across a suite of commercial and scientific parallel workloads in their experiments.

In static NUCA based CMPs, tiled architecture is an attractive design choice saving large amount of time to market in development. Tiled CMPs are built by replicating processor tiles and connecting them together via NoC, thus much of design efforts and complexity will be reduced due to the simple structure and promising scalability.

Besides the processor cores and cache, the networks of CMPs also consume a significant portion of silicon estate. Different components constituting the networks such as metal links, routers will be affected by variations of physical design parameters in both manufacture and aging process. All these unreliable effects cause potential threats to the reliability of the whole system. Particularly, on-line permanent faults induced by effects such as negative bias temperature instability (NBTI), oxide breakdown, and electromigration, cannot be discovered in post-fabrication testing. It's a challenge to design a network which is resilient enough to prevent faults from influencing the normal operation of the CMP.

Generally, in a fault-tolerant NoC-based CMP, tile-level fault isolation and fault-tolerant routing are employed to salvage the fault-stricken CMP. To completely remove the impacts of faulty parts on normal tiles, fault-tolerant routing algorithm necessitates the isolation of the fault regions, which include both faulty nodes and non-faulty nodes imposed by shape restriction. Unfortunately, its influence on static NUCA cache has been neglected in fault-tolerant NoC.

It is our observation that fault isolation technology in NoC-based CMPs will cause severe impacts on NUCA cache with fixed address mapping. Because NUCA cache banks are distributed in different tiles to form an integral physical address space for the entire CMP, removing any bank from shared cache means there will be sections of physical address space no longer accessible. When processors fetch or write back data of the lost address, miss exceptions will emerge. To the best of our knowledge, the address hollow effect caused by node isolation in NoC has not been considered in prior works.

In this paper we consider the system influence on NUCA imposed by fault isolation in degradable CMPs, and propose the utility-driven address remapping technique to hide the address space loss in NUCA cache. Additionally, through Mattson's stack distance algorithm [6], our address remapping technique can effectively mitigate the overall throughput degradation caused by faulty nodes.

II. Background

A. Base architecture and static NUCA

In this work, tiled CMP constructed by identical, general-purpose compute grids is assumed to be the target platform. Instead of using buses or rings to connect the abundant cores on-chip, tiled CMPs uses direct network of certain topology to couple the processors, which scales better than bus or ring [7]. As shown in Figure 1, the CMP of 16 tiles leverages a mesh network to provide communication channels for the distributed processors and shared cache. L2 cache of large capacity are divided into many banks for parallel access, and the cache banks belonging to different tiles are connected together via the mesh network, thus any processor can flexibly access the banks of L2 cache in local or remote tiles. Consequently, L2 cache access time will be determined by the Manhattan Distance between the requesting processor and the target cache bank rather than a single fixed latency [4].

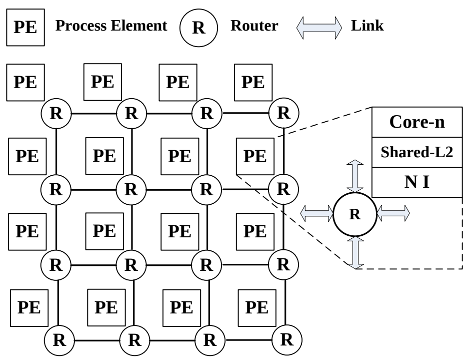

Figure 1. Typical NoC-based CMP (4X4)

Depending on memory data mapping and data search mechanism, there are static NUCA and dynamic NUCA. In static NUCA, the mapping of data into cache banks is fixed. Based on the block index in physical address, data can only reside on one bank. Therefore, every bank in compute grids corresponds to predetermined sections of physical address space. In dynamic NUCA data can migrate from remote banks to local banks, reducing access latencies and exploiting the data locality. However, dynamic NUCA relies on "smart search" mechanism and complex protocol, and the resulted design complexity, hardware and power overhead are not always affordable compared with static NUCA [8] [5] [4]. Recent researches have proved for certain application environment static NUCA can achieve performance than dynamic NUCA in CMP [4] [9].

B. Degradable NoC-based CMP and NUCA cache reliability

In NoCs-based CMP, each tile, including a processor, L2 cache banks, a router, is a unique node in network. Tiles use a shared interconnects system to transmit packets of information between them. Once the network element in one node suffers from permanent fault such as link open faults or router failure, the entire system faces the threat to fail because the NoC is responsible for the data moving in CMP. In case of components failure, faults resilient NoC should be able to work around faulty nodes and links by isolating the faulty tiles. Then the faulty tiles will become unavailable. To ensure the normal function of the degraded CMP, a fault tolerant network should adapt to the situation, for an instance, routing-table based network will auto-reconfigure the table entries according to the position of failure occurrence, maintaining the full connection of the healthy nodes through fault-tolerant routing algorithm [10].

If L2 cache banks belonging to an isolated tile get removed from the whole shared L2 cache, a severe problem will be caused in static NUCA based CMP. It has been mentioned that in static NUCA, the cache banks distributed in network shares the same physical address space. As illustrated in Figure 2, with the data mapping predetermined, each cache bank in compute node is associated with certain sections of physical address space. If a node is isolated, the related physical address space also becomes invalid in CMP system, assuming that the address decoder and routing logic are implemented in hardware without flexibility to reconfigure. Therefore, address hollows like Area-A and Area-B are formed in cache. The involved performance cost will be quite high, because the process allocated with this section of page frames will frequently encounter L2 misses and has to fetch data from the off-chip memory.

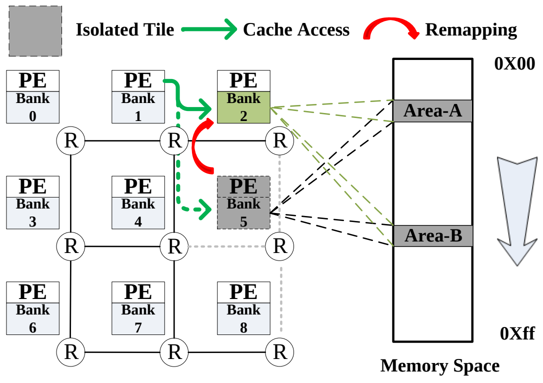

Figure 2. The formation of address hollows in shared cache

In general, there are two sources leading to the generation of address hollows in cache:

- Fault isolation in fault-tolerant NoCs.
- Defects in cache itself, especially in peripheral circuits of SRAM that make the bank inaccessible. Compared with the first case, the probability of this scenario is much lower.

To combat the problem induced by address hollows in cache, existing techniques can be resorted to. In conventional processors of large capacity cache, sets reduction technology is usually provided to support cache capacity reconfiguration for the user, in Intel Xeon processor 7100 series [11], by setting two designated sets reduction bits in physical address, cache capacity can be configured to three major configurations: 4Mb, 8Mb, and 16Mb. Sets reduction is intended to harvest static leakage power in chips, however it can also be used to tackle with the problem of address space loss in static NUCA cache. If there are several cache capacity configuration offered, we can just choose the configuration with all the cache banks in healthy nodes by masking necessary sets bits, and map the physical address space to the new continuum of cache banks. However, sets reduction technique can only supply limited number of configurations because masking one bit in physical address means turning off half of the remaining cache, which is an over-coarse grained solution.

III. Address Remapping Technique Implementation

The basic idea in address remapping technique is to make the data lines, which are mapped to the address hollows, share the healthy cache bank in neighbor nodes with the original data blocks residing in those banks. As illustrated in Figure 2, by employing address remapping, data blocks from two hollows can be loaded into bank 2 instead of the isolated bank 5. Considering that in group-associative cache structure, data blocks of different addresses can effectively share the limited n-way space in cache sets by promotion and eviction, remapping more data blocks from the other physical address areas to these sets will be feasible.

A. *Basic architecture support for address Remapping*

Because tiles in CMP are all identical, diverting a cache request from the original destination node to another involves few changes to the hardware. Firstly, several tag bits should be added to tag arrays, which is also necessary for the cache architecture supporting capacity reconfiguration, since only the banks those are chosen as the host bank of remapped address space will turn on the extra bits, and the bits number is negligible compared with the original tag length, the added search latency and power are inadequate to influence the overall performance.

The proposed node remapping technique diverts the cache requests from a faulty destination node to a healthy new one, it is achieved by adding slight modifications to the communication protocol in network interface (NI) without any intrusive hardware changes to core logic like cache controller [12][13].

The NI controller is responsible for the packtetization and de-packetization of data, which means that all the requests accessing the cache banks in remote tiles should go through NI controller before reaching the network. Therefore, it is suitable to do address remapping in NI controller.

As illustrated in Figure 3, based on the principle applied in [1], configurable logics like table-controlled multiplexers are implemented in the packetization module, so that NI controller can attach a proper header to the requesting message according to its destination node.

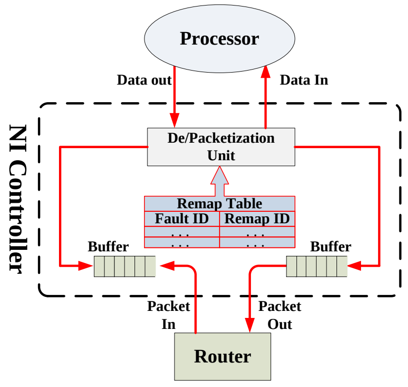

Figure 3. Reconfigurable NI controller

First of all, to enable address remapping, the NI controller should be properly configured according the location of node faults. When the NoC-based CMP encounters hardware faults in its components, the defect-tolerant network should detect and diagnose them, for example, the neighbor node send a test signal to possible faulty node and start timing. If no response returns within the time threshold, the checking node will broadcast the faulty node ID to all the routers, which then forward them to Ni on receiving the ID. In our special reconfigurable NI controller, faulty node IDs are recorded in correspondent entries to get remap table updated.

After fault isolation and remapping configuration, if a processor requests the data block in the faulty node, its request will enter into the NI controller before goes into network. However, in packetization stage, NI controller will find that the destination node of the request should be remapped, and then attach the packet with a proper head that redirect the packet to a healthy node, making the lost address space totally transparent to processor.

B. *Address remapping strategy*

When a faulty node is detected and isolated, the scattered remapping tables will be updated simultaneously. Firstly only the faulty ID is stored as a new hash key in the allocated table entry and the hash value is not determined yet. By freely programming the value associated with the hash key, we can remap the cache address hollows to any healthy banks in the CMP. If there are enough available nodes left in the CMP, an optimal remapping configuration may exist for a certain optimization goal, which motivates us to exploit the choice space for performance improvement.

An intuitive idea for deciding the target node for cache remapping is that the pressure should be balanced among all the tiles to avoid the formation of hotspot. When the cache banks of a certain node are over-loaded with data lines from different address spaces, conflict misses and local communication hotspots will become a problem. To achieve a balanced data layout in cache, our remapping process goes as follows:

1) *Profiling cache address space utilization:* Before determining a balanced remapping configuration, an effective method to profile the cache banks pressure should be found. Mattson's stack distance algorithm [6] is able to reflect the data reusing feature of the program, and it is used to provide miss rate predictions in works as [13] [15] [16]. Similarly, in this paper we use Mattson's stack distance algorithm to characterize the space non-uniformity of cache access, and predict cache space utilization level in each node.

Mattson's stack distance (MSD) profiling for cache space utilization is based on the inclusion property of widely accepted LRU or LFU replacement policy. It precisely projects the reusalility of the data lines in N-way set associative cache. As shown in Figure 4, in the 4-way set associative cache, way numbers rank in MRU order from 1 to 4, and each position corresponds to a stack distance. The bank access number of Figure 4 demonstrates the MSD profile for ocean of SPLASH-2 suite. From the profiled access patterns, we can see that bank 1 shows more access pressure than bank 0.

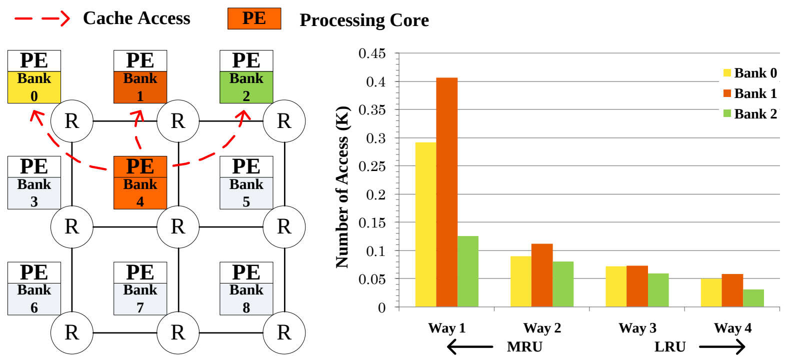

Figure 4. Example of cache bank utilization analysis of benchmark *ocean*

There're different ways to profile the MSD distribution in cache banks. One method is to attain them from complier which can automatically analyze the data trace of the application. The other way is partially pre-executing the application before running.

In a degraded system with unusable nodes, counter-based hardware profile like [16] is not practical, so software method is adopted in our scheme. During the pre-execution stage, by scheduling a kernel thread monitoring the memory access instructions to different physical address areas, the same effects as the hardware counters can be achieved.

After pre-execution, the approximate MSD distribution collected by profiler will be used to estimate the "temperature" of each cache bank. We measure the bank temperature by predicting miss increase caused by space sharing with a baseline data trace. If certain numbers of ways in cache sets get occupied by other sharers, there will be extra cache misses caused by conflicts in the cache bank. We can predict the miss increase in a bank via Mattson's stack distance algorithm and use that value to indicate the utilization level of the bank.

2) *Determining address remapping configuration:* With the bank utilization information, in our remapping algorithm the faulty bank IDs and the healthy bank IDs are ordered separately according to their utilization levels. Those banks that rank higher are considered hot banks. By delicately matching the hot bank and the cool bank together, an address remapping configuration with balanced cache bank utilization can be found and thus miss rate can be greatly reduced.

However, data flow characteristics differ from application to application. In many cases, the cache banks may exhibit similar utilization levels, and it means we have more than one choice to remap for the address hollows. A further step should be taken to exhaust the optimization space if we want to fill the remapping table with the most suitable node ID. In this step, the Manhattan Distances between the original isolated cache node and its potential couples are estimated. The node closest to the faulty node will be chosen as the remapping target. It's a classic Maximum Matching problem when there is still more than one node for some faulty banks to remap. This step is intended to reduce the communication latency considering that the data layout in L2 cache is intentionally optimized to exploit application's spatial locality. Remapping the cache address space from the original node to a neighbor one avoids disorganizing the original data layout in distributed shared cache, so that the latency to fetch a remapped data block may be much lower for the requestor.

For address remapping method, MSD profiling through program pre-execution, involves in no hardware overhead. Therefore the only hardware costs are the remapping tables, which consume little chip area because both the entry number and the entry length are negligible.

IV. EVALUATION

A. Experiment environment

We validate our technique by simulating SPLASH-2[17] suite using GEMS-2.1 simulator [18]. Both multithreaded and multiprogrammed workloads are picked to comprehensively evaluate the proposed method. For multiprogrammed workload, we use various combinations of SPLASH-2 benchmarks. By inserting breakpoints in programs, initialization phases are skipped to ensure that only the main functionality is included. The simulator configuration matches the baseline shown in Table I.

TABLE I. BASELINE CMP CONFIGURATION

Cores	In-order, 2GHz
Networks Topology	3X3,2D-Mesh
Network Link Latency	4 cycles
Base memory Hierarchy Parameters	
L2 Cache Size	Shared, NUCA, Exclusive, 4MB, 8 way, 8 banks,64B lines, LRU replacement, 6 cycles hit
Memory	4G DRAM, 260 cycles access
Coherence	MOESI directory

B. Experiment Mechanism

In fault-tolerant simulation, we randomly injected permanent faults to the CMP in node-level granularity by varying injection rate. Since fault detection and diagnosis are not considered in this paper, we assume the locations of faults are known to the system globally so that fault detection phase is avoided.

In simulation, we imposed restriction on fault injector to ensure that at least one tile is not corrupted by faults. For each scenario of workload, the fault injection simulation will be repeated many times so that average results could be attained.

We implemented both proposed address remapping technique and Xeon-like cache sizing scheme in GEMS simulator and compared the effectiveness of the two schemes using miss rate as the evaluation metric.

C. Experimental Results

Figure 5 shows the average miss rates of 4 single-task applications in different fault distribution density. The plotted values are normalized to the miss rate in fault-free CMP running the same applications. There are three schemes we evaluated in experiment: sets reduction, random remapping and utility-driven remapping (UDR). From the results, we can see UDR achieve the least performance degradation. By employing UDR, three programs (lu, radix, ocean) shows more than 40% miss rate reduction while one task (radiosity) gains no dramatic improvement compared with sets reduction as well as random remapping. By interpreting the program behavior, we concluded that the phenomenon is due to the problem size difference between programs. For radiosity in our experiment, the miss rate is not sensitive to the capacity of LLC in a certain range because of its working set size and unstructured data distribution. In this case, UDR exhibits limited advantage over sets reduction.

We also combine different programs and run them simultaneously on CMP as multi-programmed workload. Assuming that there would be obvious non-uniformity in cache bank access for multi-programmed application, we expected much better performance will be witnessed in UDR over random remapping. Figure 6 shows that UDR still outperforms sets reduction in lowering the miss rate. However, compared with random remapping, UDR only achieves about 10% reduction in miss rate. One possible explanation to the results is that when more programs are executed on processors, the cache capacity requirement rises and most of nodes become hot. If there are only hot banks within the L2 cache, UDR performs no better than random remapping.

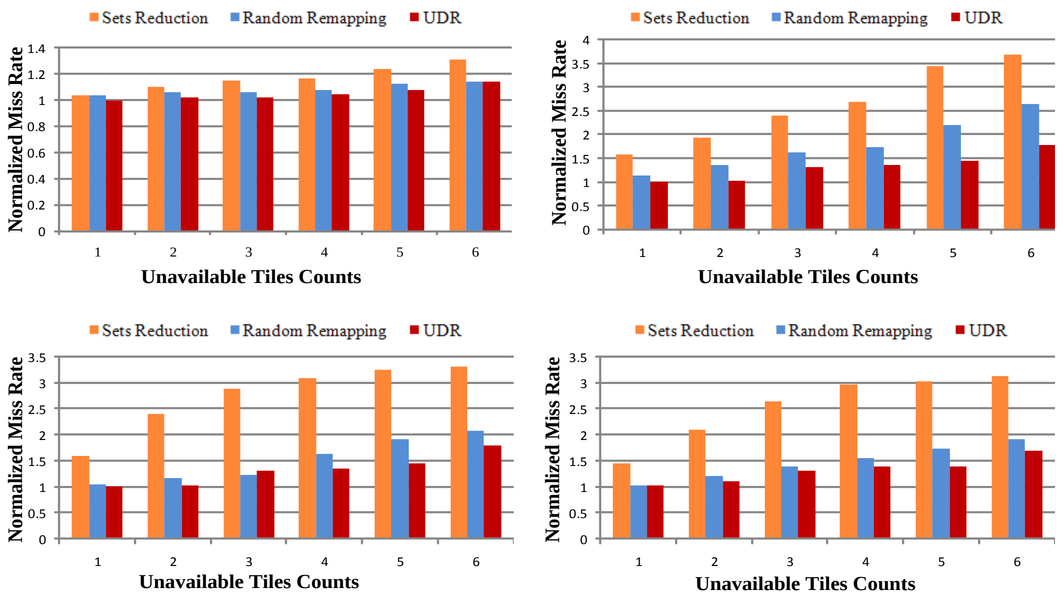

Figure 5. Performance comparison for different application: (top-left, radio), (top-right, lu), (down-left, radix), (down-right, ocean)

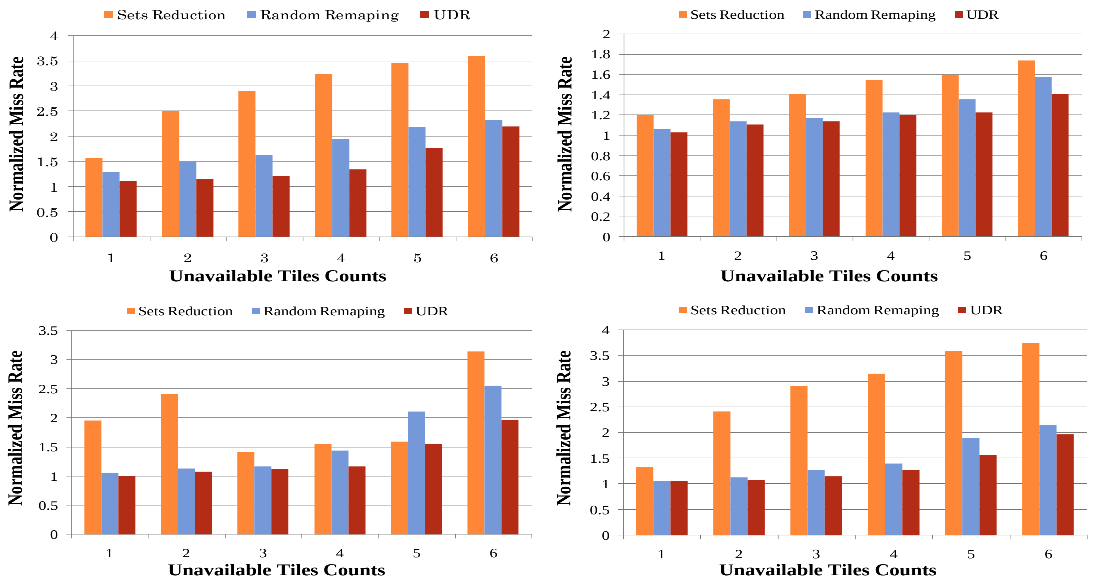

Figure 6. Performance comparison for multi-programmed workload: (top-left, cholesky and lu), (top-right, radio and radix), (down-left, lu and fft), (down-right, lu and radix)

V. Related Work

In this paper, we conduct research on static NUCA cache reliability in NoC-based CMP. As an effective way to organize the shared cache, NUCA has been studied in context of CMP for years since Kim et al. proposed it to mitigate the access latency of large on-chip caches [8] [4]. Bechmann evaluated NUCA architecture and concluded that dynamic NUCA can benefit performance but costs a lot for intensive data migration and complex search mechanism [9]. There are also a lot of researches on new cache architectures such as Nahalal [19] and R-NUCA [20] based on the original static or dynamic NUCA. We directed our work to solve the possible reliability problem in widely-applied NoC-based CMP, based on the observation that there will be address hollows in static NUCA cache caused by fault isolation.

The proposed address remapping technique is inspired by recent work intended to salvage cache from PV-induced faults or aging effects [21] [12] [22] [23]. Shirvani et al. [12] use a special programmable address decoder to disable faulty blocks and map their references to normal blocks, so that cache block defects can be tolerated for improvement of yield. This work solves cache fault in a very fine granularity and require intrusive circuit modification. There is also another scheme using mapping table to map a faulty block onto a healthy one [22], and it is applicable to direct-mapped caches.

Some reliable cache architecture has self-adaptive features like dynamic resizing to survive occasional hardware defects. Agarwal et al. [1] isolate cache faults by capacity resizing in SRAM arrays, allowing healthy cells in different rows to form a whole SRAM row.

In storage circuit designs, redundant array is often used to provide replacement for faulty block. In [23] redundant SRAM array elements and permutation network are combined to promote the availability of cache in high fault density technology.

All these schemes are related to our goal of tolerating cache faults and optimizing capacity utilization. However, they mainly focus on circuit-level or microarchitecture-level redundancy or sharing. Our proposed address remapping technique, concerning about the system level reliability problem in NoC-based NUCA cache, is orthogonal to such methods.

VI. Conclusion

In this paper, we presented an analysis of a possible failure mechanism for static NUCA, which is caused by fault isolation in degradable NoC-based CMP, and proposed the address remapping technique to get around address space loss in NUCA cache. This scheme is transparent to processors and raises negligible overhead via the reconfigurable NI controller. The experimental results show that the proposed method considerably mitigates the fault-induced performance degradation compared with sets reduction.

Besides, this work leaves space for further exploration. In future work, we plan to improve the technique by allowing adaptive address remapping in program execution stage.

Acknowledgment

This research is supported in part by National Natural Science Foundation of China (NSFC) under grant No.(60806014, 60831160526, 60633060, 60921002, 60906018, 61076037), in part by National Basic Research Program of China (973) under grant No. 2011CB302503, and in part by Hi-Tech Research and Development Program of China (863) under grant No. 2009AA01Z126. The authors would also like to thank anonymous reviewers for their comments on the paper.

References

[1] A. Agarwal et al., "A Process-Tolerant Cache Architecture for Improved Yield in Nanoscale Technologies", IEEE Trans on VLSI Systems, Jan. 2005.

[2] J . Srinivasan et al. "The Impact of Technology Scaling on Lifetime Reliability," International Conference on Dependable Systems and Networks, June 2004.

[3] Miller, M, "Manufacturing-aware Design Helps Boost IC Yield", http://www.eetimes.com/news/design/features/

[4] Kim. C, D. Burger, et al., "An adaptive, non-uniform cache structure for wire-delay dominated on-chip caches", Proceedings of the 10th international conference on Architectural support for programming languages and operating systems, 2002

[5] Jaehyuk Huh, J, C. Changkyu Kim, et al., "A NUCA Substrate for Flexible CMP Cache Sharing", IEEE Transactions on Parallel and Distributed Systems, 18(8): 1028-1040, 2007.

[6] R.L.M. et al.,"Evaluation Techniques for Storage Hierarchies", IBM Systems Journal, 1970.

[7] Wentzlaff. D, P. Griffin, et al., "On-Chip Interconnection Architecture of the Tile Processor.", Micro, IEEE 27(5): 15-31,2007

[8] Zhang. M, K. Asanovic, "Victim Replication: Maximizing Capacity while Hiding Wire Delay in Tiled Chip Multiprocessors", Proceedings of the 32nd annual international symposium on Computer Architecture, 2005

[9] B. M. Beckmann and D. A. Wood, "Managing wire delay in large chip-multiprocessor caches", In Proceedings of the 37th Annual IEEE/ACM International Symposium on Microarchitecture, Dec. 2004.

[10] Fick. D, DeOrio, A. Chen. G, Bertacco. V, Sylvester. D, Blaauw. D, "A highly resilient routing algorithm for fault-tolerant NoCs", Design, Automation & Test in Europe Conference & Exhibition, Page(s): 21 – 26, 2009

[11] Chang, J., H. Ming, et al., "The 65-nm 16-MB Shared On-Die L3 Cache for the Dual-Core Intel Xeon Processor 7100 Series", IEEE Journal of Solid-State Circuits, 2007

[12] Shirvani, P. P. and E. J. McCluskey, "PADded cache: a new fault-tolerance technique for cache memories", VLSI Test Symposium, 1999

[13] Mutyam, M. and V. Narayanan, "Working with Process Variation Aware Caches", Design, Automation & Test in Europe Conference & Exhibition, 2007.

[14] M. K. Qureshi and Yale N. Patt, "Utility-Based Cache Partitioning: A Low-Overhead, High-Performance, Runtime Mechanism to Partition Shared Caches", Proc. of Annual IEEE/ACM International Symposium on Microarchitecture, 2006

[15] P. Zhou, et al. "Dynamic Tracking of Page Miss Ratio Curve for Memory Management", Architectural Support for Programming Languages and Operating Systems, 2004.

[16] Kaseridis. D, J. Stuecheli. et al.," A Bandwidth-Aware Memory Subsytem Resource Management Using Non-Invasive Resource Profilers for Large CMP Systems". Proc. of International Symposium on High-Performance Computer Architecture, 2010.

[17] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The SPLASH-2 programs: Characterization and methodological considerations", In International. Symposium on Computer Architecture, June 1995.

[18] M. Martin et al., "Multifacet's General Execution-Driven Multiprocessor Simulator (GEMS) Toolset", Computer Architecture Newsletters, 2005.

[19] Z. Guz, I. Keidar, A. Kolodny, and U. C. Weiser, "Utilizing shared data in chip multiprocessors with the Nahalal architecture", In Proceedings of the 20th Annual ACM Symposium on Parallelism in Algorithms and Architectures, Jun. 2008.

[20] Hardavellas. N, M. Ferdman, et al., "Reactive NUCA: near-optimal block placement and replication in distributed caches." SIGARCH Computer Archititecture News, 2009

[21] Hyunjin, L., C. Sangyeun, et al., "Performance of Graceful Degradation for Cache Faults", IEEE Computer Society Annual Symposium on VLSI, 2007.

[22] Luong, D. H., G. Masahiro, et al.," SEVA: A Soft-Error- and Variation-Aware Cache Architecture. Dependable Computing", Pacific Rim International Symposium on Dependable Computing, 2006

[23] Amin Ansari, S. G., Shuguang Feng, Scott Mahlke, "ZerehCache: Armoring Cache Architectures in High Defect Density Technologies." Proc. of Annual IEEE/ACM International Symposium on Microarchitecture, 2009.