REMEDIATE: A Scalable Fault-tolerant Architecture for Low-Power NUCA Cache in Tiled CMPs

Abbas BanaiyanMofrad[1], Houam Homayoun[2], Vasileios Kontorinis[3], Dean Tullsen[3], Nikil Dutt[1]
[1] Department of Computer Science, University of California, Irvine
[2]Department of Electrical and Computer Engineering, George Mason University
[3]Department of Computer Science and Engineering, University of California, San Diego
{abanaiya, dutt}@ics.uci.edu, hhomayou@gmu.edu, {vkontori, tullsen}@cs.ucsd.edu

***Abstract* - Technology scaling and process variation severely degrade the reliability of Chip Multiprocessors (CMPs), especially their large cache blocks. To improve cache reliability, we propose REMEDIATE, a scalable fault-tolerant architecture for low-power design of shared Non-Uniform Cache Access (NUCA) cache in Tiled CMPs. REMEDIATE achieves fault-tolerance through redundancy from multiple banks to maximize the amount of fault remapping, and minimize the amount of capacity lost in the cache when the failure rate is high. REMEDIATE leverages a scalable fault protection technique using two different remapping heuristics in a distributed shared cache architecture with non-uniform latencies. We deploy a graph coloring algorithm to optimize REMEDIATE's remapping configuration. We perform an extensive design space exploration of operating voltage, performance, and power that enables designers to select different operating points and evaluate their design efficacy. Experimental results on a 4x4 tiled CMP system voltage scaled to below 400mV show that REMEDIATE saves up to 50% power while recovering more than 80% of the faulty cache area with only modest performance degradation.**

Keywords— Fault-tolerant cache, Remapping, Aggressive voltage scaling

I. INTRODUCTION

Tiled CMP architectures, which are designed as arrays of identical or very similar basic blocks, are a design complexity scalable alternative to current small-scale CMP designs. A Tiled CMP with shared Last-level Cache (LLC) and Non-Uniform Cache Access (NUCA) organization logically shares the physically distributed cache banks among all its cores. Technology scaling and process variation has an increasing impact on the resilience of CMOS circuits including Tiled CMPs [1]. Due to large size of LLC in CMPs, we are increasingly challenged by the often-conflicting constraints of reliability, manageable power, and temperature in design of such architectures. Operating in reduced-Vdd (e.g., near-threshold) voltage is a common solution to reduce LLC power consumption. However, aggressive voltage scaling of SRAMs increases the effect of process variation. So it results in a high incidence of failure, to the point that it can overwhelm traditional error correction techniques (e.g., SECDED) [5].

There is a large body of previous work on fault-tolerant design of cache memories. Section II summarizes some of those fault-tolerant techniques. Most of them have been primarily proposed for uniform cache access (UCA) in a single core architecture. Additionally, most of them are not efficient for high failure rates. More importantly, they do not address the challenges of a shared NUCA cache design in CMP architecture like multiple access points, concurrent banks accesses, non-uniform latency, variation in fault-rate, and criticality of different banks. For instance in a fault-tolerant NUCA design, access latency, network traffic, network power, and effective capacity are all sensitive to the location of the spare block or redundancy used for protection of a faulty data, relative to the requesting core and the accessed line.

In this work, we propose REMEDIATE, a scalable fault-tolerant architecture for low-power design of shared LLC with non-uniform latencies, distributed banks, and multiple access points in a multicore architecture. REMEDIATE leverages a flexible fault protection technique while considering the implications of two different remapping heuristics in the presence of cache banking, non-uniform latency, and interconnects. Furthermore, it uses the opportunity to remap faulty lines across different distributed banks in a shared NUCA cache. Thus, a faulty block can be remapped to anywhere in the existing NUCA LLC. REMEDIATE's remapping policies aim to achieve a balance between the conflicting goals of minimizing latency, minimizing power, maximizing capacity, and minimizing network traffic for NUCA caches.

The **main contributions** of this paper are that we: 1) Introduce REMEDIATE, a scalable and highly reconfigurable architecture that can be leveraged to protect large shared NUCA LLC in tiled CMPs against permanent faults[1]; 2) Propose two scalable remapping policies for efficient fault tolerance of distributed shared NUCA cache banks in tiled CMPs; and 3) Perform design space exploration of various cache design configurations at 65nm to demonstrate the efficacy of REMEDIATE. *To the best of our knowledge, REMEDIATE is the first design to address the fault tolerance of distributed NUCA caches for CMP architectures.*

II. RELATED WORK

Several researches on improving SRAM reliability in face of process variation-induced faults at low voltage operation have been proposed in different levels of system hierarchy from circuit level [13][14][26] to system level [27][28][29][30]. At the system level, a variety of Error Detection Code (EDC) and Error Correcting code (ECC) techniques have been used. ECC is proven as an effective mechanism for handling soft errors [22]. However, in a high-failure rate situation, most coding schemes are not practical because of the strict bound on the number of tolerable faults in each protected data chunk. In addition, using ECC incurs a high overhead in terms of storage for the correction code, large encoding latency, and slow and complex decoding [17][18].

[1] While REMEDIATE can also be modified to address soft failures, in this work our focus is on permanent failures which for instance could be the result of process variation, aggressive voltage scaling and aging effects.

Several architectural techniques have also been proposed to improve reliability of on-chip cache and/or lower the minimum achievable voltage scaling bound by using redundancy. Wilkerson et al. [24] proposed two schemes called Word-disable and Bit-fix. The first scheme combines two consecutive cache blocks into a single cache block, thereby reducing the capacity by 50%. The second one sacrifices a functional cache block to repair defects in three other cache blocks, thereby reducing the capacity by 25%. Bit-Fix method also adds three cycles of latency to the cache access time. Roberts et al. [19] proposed a block grouping (pairing) scheme to form a new, fully working logical block. RDC-Cache [5] replicates a faulty word by another clean word in one way of the next cache bank in a chain of cache banks. The salvage cache [4] uses a single non-functional block to repair several other blocks in the same set. A similar idea has been proposed independently in [20]. Ansari et al. [10] deployed a fault-tolerant cache that groups faulty cache sets together and sacrifices a different cache set to recover failures. The sets are grouped together in such a way that the sacrificial set and the data sets reside in different banks. They improved their technique in another work, named Archipelago [11], that uses a modified version of minimum clique covering algorithm to cluster the cache into different groups. A similar but more flexible method was proposed in FFT-Cache[12], where a flexible defect map is used to configure the cache architecture using a portion of faulty cache blocks/sets as redundancy to tolerate other faulty cache blocks/sets. All these techniques rely on two banks and a single fault map that tracks the fault-tolerance information of both banks. Wang et. al [6] proposed a utility-driven address remapping technique at a coarser-grain (bank level) to tackle the capacity loss in NUCA cache of NoC-based CMP architectures. BanaiyanMofrad et al. 0 proposed a scalable design to protect LLC banks in a NoC-based CMP which leverages the interconnect network to implement a remapping-based fault-tolerant technique.

Unfortunately all these previous efforts are neither scalable nor flexible to be leveraged for protection of NUCA caches with distributed banks in CMP architectures. Since CMP architectures have more than one core that can access one or different distributed banks simultaneously, fault mapping and protection needs to be scalable and distributed. Furthermore, existing fault-tolerant techniques typically use a single fault map and a centralized fault protection scheme which cannot be scaled for large NUCA CMP architectures: as a result each access needs to go through the fault map first, and no parallel cache accesses are allowed. Another difference for NUCA architectures is that the access latency, failure rate, and criticality of distributed banks are different depending on the location of each cache bank and type of NUCA (SNUCA or DNUCA). Therefore, the fault-tolerant technique needs to be scalable and configurable to address these issues. However, all previous fault-tolerant methods have been designed for at most two banks within a UCA architecture which are not configurable. Moreover, they use a unified remapping technique to protect all faulty blocks without considering differences among different banks. In contrast to previous efforts, REMEDIATE leverages a scalable architecture allows efficient remapping across multiple banks in a NUCA cache. It uses a flexible fault mapping and fault protection scheme which results in higher cache reliability. REMEDIATE uses a configurable fine-grained remapping technique using graph coloring to match the target (redundant) blocks to a group of original accessed blocks/sets in a multi-bank cache architecture. REMEDIATE's configuration algorithm attempts to minimize the number of target blocks by considering the variation of probability of failure and available space of different cache banks.

III. Baseline Architecture

We experiment on a tiled CMP architecture, where each tile comprises a processor core, caches, and network router/switch. Tiles are interconnected as a 2D mesh via a network-on-chip (NoC). Fig. 1 shows our Baseline 16-tile configuration, where each tile includes private L1 data and instruction caches and a shared L2 bank. The L2 bank is a portion of the larger distributed shared LLC. Each L2 bank includes multiple cache sets, with a set of lines (blocks with the same index) all mapped to the same bank. The baseline design assumes a NUCA [2] LLC. With multiple banks within the LLC, we have the choice of either always putting a block into a designated bank (static mapping) or allowing a block to reside in one of multiple banks (dynamic mapping). We consider static mapping in our baseline design and model static NUCA policy for CMP architectures (CMP-SNUCA) [3]. CMP-SNUCA statically partitions the address space across cache banks connected via a 2D mesh interconnection network.

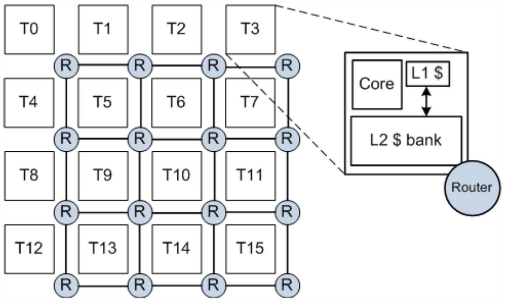

Fig. 1. Baseline Tiled CMP architecture.

IV. REMEDIATE Mechanism

REMEDIATE attempts to sacrifice the minimal number of cache lines to minimize cache capacity loss and tolerate the maximum amount of defects. This is done by using line-level replication in the same set or among multiple sets between different banks. In REMEDIATE, the information of faulty locations in each LLC bank is kept in a small Permanent Fault Map (PFM) inside that bank, which is then used to configure the address remapping. Instead of maintaining explicit additional cache space to replace faulty lines, it exploits the lines already marked as faulty to provide redundant storage to *mask* faults in other lines. This has a critical advantage: while a static redundancy technique is based on worst-case estimates of faults, *our approach sacrifices no more space than is absolutely needed for redundancy*. This is particularly useful for a cache that might be used at multiple voltage levels, as we automatically adjust the level of redundancy according to the errors manifested at a particular voltage.

REMEDIATE divides each line (block) of LLC into multiple sub-blocks, that defines the granularity at which failures are identified and remapped. A sub-block is labeled faulty if it has at least one faulty bit, as determined by Built-In Self Test (BIST) analysis during boot-up of the system. Thus, a smaller sub-block minimizes the number of bits lost due to a single fault, but increases the hardware overhead of reconstructing lines. If two lines have faults in the same sub-block, we say they conflict and one cannot be used to mask the faults in the other one. When a line is detected as faulty, the system attempts to remap faulty portions of the line (Original line) to another line (called the Target line) that has already been marked as faulty and does not conflict with the original line. It then sacrifices and disables the target line to replicate all faulty sub-blocks of the original line. Thus, the correct line can be reconstructed from a combination of the two lines; i.e., the original

line together with the target line.

In REMEDIATE cache architecture, each cache access in low power mode which expected to be error-prone, first accesses the PFM. Based on the fault information of the accessed line, we either read one or two lines from one bank (the same set), or one line from the original bank and one line from the target bank (different sets), as specified by the target line information in the PFM of the original bank. In many cases, a good choice of the target is another line in the same set, referred to as a local target line, since both original and target lines can be read in a single access. Barring this case, we should always select target lines from a different bank, so that both lines can be read without two serialized accesses to the same bank. We refer to this target line as a remote target line. In this case, the latency of the access is the maximum of the two bank access latencies that determined by the farther bank from the core. Because the two original and target line are known not to have sub-block conflicts, and we know which sub-blocks contain errors, the cache controller can always reconstruct the original line using single bank multiplexing techniques.

A. Reconstructing Cache Lines

Fig. 22 shows both the conventional and modified architecture of a tile with a core and a 2-way set associative LLC bank where the ways (blocks) are further divided into 2 sub-blocks. Figure 2(b) illustrates the architecture of the REMEDIATE multiplexing layer responsible for reconstructing correct lines. The new modules (MUXs and PFM) added to the conventional cache are highlighted in the figure.

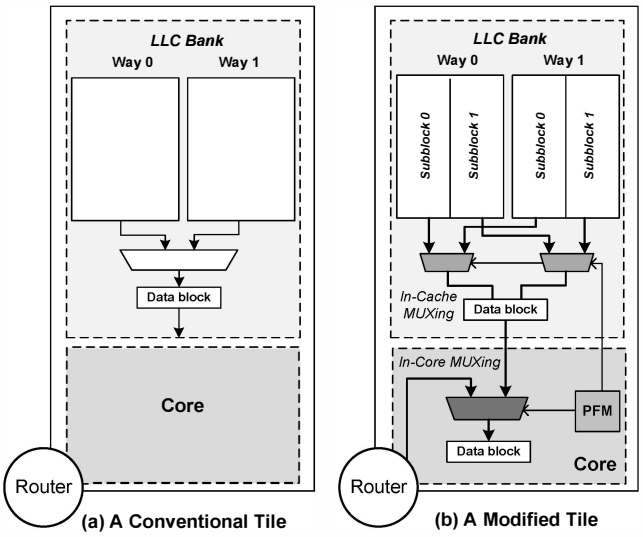

Fig. 2. Architecture details of (a) a conventional tile and (b) the modified tile in REMEDIATE Cache with PFM and 2 sub-blocks per line.

REMEDIATE requires two additional levels of multiplexing to compose the final fault-free line, based on either multiple lines within a set or between two or more sets in different banks. The first multiplexing layer, *In-Cache MUXing* is added on the cache bank side to perform the remapping within a set. Note that this layer replaces the original block-level multiplexer, available in set-associative caches (Fig. 2.a). The second multiplexing layer, *In-Core MUXing* is added on the core side to perform remapping between sets in original (local) and target (remote) banks.

The total number of *n-to-1 ss-bit* multiplexers required in cache side to compose the final fault-free line is $k \times b$, where n is the number of ways (associativity), ss is sub-block size, k is the number of sub-blocks in a line, and b is the number of banks. Also, a *2-to-1* block-level multiplexer is required for each core. For instance, for a 16-core tiled processor with 16 banks of 4-way set associative LLC, with 64-byte blocks which composed of 32 16-bit sub-blocks, its 16 4-to-1 64-byte original multiplexers would be replaced by a total of 512 4-to-1 16-bit multiplexers in cache banks. Also, it needs extra 16 2-to-1 64-byte multiplexers in core side. For the described cache, the PFM size will be less than 9% of total cache area. For a quantitative comparison, we synthesized the multiplexing layer and output logic of the REMEDIATE, as well as the multiplexer and output driver of a conventional cache (CC) using the Synopsys Design Compiler for TSMC 65nm standard cell library. The area and delay of various multiplexers are used to estimate the overall area/delay overhead of the multiplexing network in REMEDIATE and the CC under nominal Vdd. We found out that the delay of REMEDIATE output logic increased by only 5% compared to the conventional cache output MUX network while area and power consumption are increased by less than 1% compared to a CC MUX network. These overheads are modeled in our experiments.

B. PFM Configuration

We use a heuristic graph-coloring algorithm which is a modification of the Saturation Degree Ordering (SDO) algorithm [16] to optimize selection of a remote target line for a group of faulty lines that have no conflict with each other. We call a graph coloring problem solvable, if for a graph G we can find an integer $K \geq 0$ such that the nodes of G can be colored with K colors while no edge exists between the same colored nodes. We construct a graph based on the conflicts between lines of different entries in the PFM. Each node in this graph represents either a target line or a faulty entry (representing an entire one set) in the PFM. The edges represent a conflict between a pair of lines or between a line and a set. For example, the two nodes connected by an edge represent two lines, two sets, or one line and a set that have a conflict.

We modify the above graph coloring algorithm based on the following constraints: 1) We force the algorithm to color nodes from at least two different banks; 2) We force the algorithm to first pick up line nodes and try to color them with set nodes of other banks; 3) We force the algorithm to consider the PFM configuration settings during its coloring process.

We apply the modified graph coloring algorithm to our graph to find a solution such that neighboring nodes are not assigned the same color. Therefore, after completion of coloring algorithm, nodes with the same color are guaranteed to have no edges between them; implying that the corresponding cache sets/lines have no conflicts between them. We set all nodes with the same color in a group and set one of them as Remote Target for other nodes in the group.

Now, we describe the configuration process for REMEDIATE cache. Initially, a raw fault map is generated at boot time; using the memory BIST unit to test the LLC cache under low voltage conditions. The output of the BIST is used to initialize the PFM of each bank. If there are multiple operating points for different combinations of voltage, temperature and frequency, the BIST operation is repeated for each of these settings. The obtained fault map is then modified and configured based on architecture settings like type of NUCA, location of banks, and criticality of some specific banks. After initialization of PFM in low power mode, the processor switches back to high power mode and constructs the conflict graph and solves the graph coloring problem. This solution contains the remapping information that is required to be stored in the PFM. This configuration information can be stored on the system memory storage and is written to the PFM during the next system boot-up. In addition, in order to protect the defect map and the tag arrays, we use the stable 8T SRAM cells [13] that can operate at

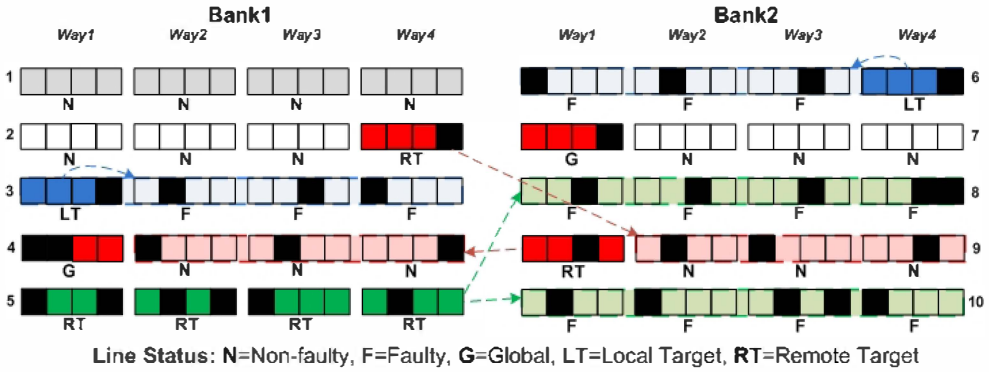

Fig. 3. An example of PFM configuration and remapping for a given distribution of faults in a 4-way set associative cache.

ultra-low voltages (below 400mv) without failing. Since the defect map and tag arrays are relatively small, we are able to tolerate the ~30% area overhead for these larger cells.

In contrast with some of the previous approaches[10][11], PFM size is fixed and it does not depend on the failure rate. It has one entry for each cache set. Each PFM entry includes multiple configuration bits, the defect bitmap of each line in the set, status of each line and the address of the target line, if any. Each bit in the defect map section represents the fault state of a sub-block. Line status bits represent the status of each line in a cache set. The status of each cache line can be one of the followings: 1) Non-Faulty, 2) Faulty, 3) Global, 4) Local Target, and 5) Remote Target. Initially, the status of all lines in the cache is Non-Faulty, representing the absence of any faulty sub-blocks. If a line contains at least one faulty sub-block, its status will be set as Faulty. A line used as a target line for other lines in the same set gets the status of Local Target. A line that has a conflict with other lines in a set and cannot be used as a local target line, gets the status of Global. A Global line can be used as a target line of another set, and becomes a remote target line and gets the status of Remote Target. We define the *Max Global Line (MGL)* parameter as the maximum number of lines in a set that can be set as Global lines; the remaining lines can then be composed as a group of lines without conflict, which allows them to find a Global line and set it as their Remote Target line.

We categorize the cache sets based on the number of conflicts among the lines inside each set to four groups:

- **min-faulty:** if the number of faulty lines in a set is lower than the predefined MGL
- **no-conflict:** if there is no conflict between faulty lines in the set
- **low-conflict:** if the number of conflicts between the lines within a set is lower than the predefined MGL
- **high-conflict:** if the number of conflicts between lines within a set is higher that the predefine MGL

Figure 3 shows an example of the PFM configuration for a given distribution of faults in 10 sets of a 2-banked 4-way set associative cache with 4 sub-blocks in each line and MGL=1. As shown in the figure, set 1 is clean, without any faulty lines. Set 2 is a member of the *min-faulty* group while its faulty line is configured as a Remote Target line for set 9. Set 7 is another member of this group and its single faulty line is set as a Global line. Set 3 is an example of a *no-conflict* group in which the first line (way 1) is set as a Local Target line to be sacrificed for fault-tolerance of the other faulty lines in the set. Set 6 is another member of this group with one of its lines (way 4) set as a Local Target Line. Set 4 is a member of the *low-conflict* group which one of its lines (way 1) is set as a Global line, since it has a conflict with other lines. The first line of set 9 is set as Remote Target line for this set (set 4). Set 5 is a member of the *high-conflict* group with two conflicts between its lines where way 1 has conflict with way 3 and way 2 with way 4. All lines of this set are configured as Remote Target line for both *low-conflict* sets 8 and 10.

V. Remapping for NUCA Caches

There are several challenges to applying REMEDIATE in a distributed shared NUCA LLC with multiple access points. The mapping of lines to cache banks in a NUCA cache has a significant impact on performance and power. Adding REMEDIATE remapping on top of that gives another dimension to the problem. Thus the REMEDIATE remapping policy must balance several conflicting goals, including:

1. *Minimizing the maximum distance from core to either original or target bank.* Cache latency will be determined by the maximum distance to the original and target banks. We cannot always predict what core will access the data, but if the target bank is near the host bank, then it is more likely the maximum distance will not be inflated significantly.
2. *Minimizing the distance from original bank to target bank.* Power and traffic congestion and, as a result, latency are minimized by minimizing the distance of between original and target banks.
3. *Maximizing total cache capacity.* The more freedom we have to select target lines from different banks, and the host lines that map to those target lines, the fewer total lines will be sacrificed.

Since the cache bank access pattern depends on the application that each core is running and dynamically changes with program behavior at run-time, a one-solution-fits-all approach may not deliver optimal results. Instead, we propose two heuristics for REMEDIATE remapping policies which are simple but scalable and aimed at achieving a different balance of these goals. The first policy attempts to minimize access latency by exploiting adjacency while the second one places highest priority on preserving cache capacity.

A. Adjacent Mapping Policy

In this remapping policy, the highest priority for placing the target line is given to LLC banks in adjacent tiles. This attempts to minimize cache access latency and network traffic, particularly in the case where we cannot predict the requester core. Based on the number of allowed hops (tiles in the NoC) to reach the target bank from original bank, we can define different modes where a faulty line is being remapped into any configuration of a single up to multiple adjacent banks. For instance, we can have a mode that allows remapping to only one target bank that can be reached at one hop (mode M1). Alternatively we can have a different mode that allows remapping to at most two banks that each can be reached with at most one hop (mode M2). Fig. 4(a) shows an example of M2 mode for all banks in CMP. Other modes are also possible based on the number of target banks and maximum hops to reach them.

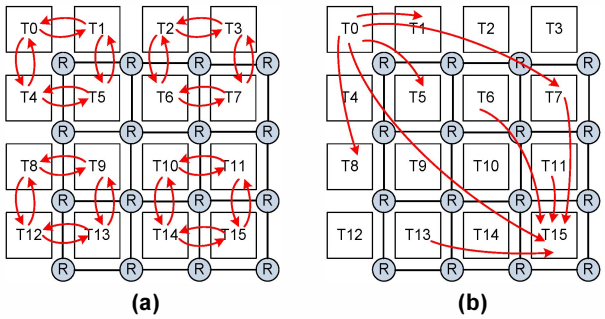

Fig. 4. Example showing the (a) Adjacent mapping scheme (b) Global mapping scheme.

B. *Global Mapping Policy*

The adjacent remapping policy potentially sacrifices cache capacity (i.e., prevents the technique from finding the optimal mapping by limiting it to just adjacent banks) for latency or traffic considerations. However, when the cache lost capacity induces a miss, it can have far greater impact on both performance and power than a suboptimal line target mapping. This remapping policy thus imposes no location restrictions for the mapper, and the target lines can be selected from LLC banks in any tiles in CMP. However, this policy gives priority to the banks of adjacent tiles, and if the target line is not found in adjacent banks, it examines other banks. Fig. 4 (b) represents an example in which LLC bank of Tile0 can has five target banks in different distances and LLC bank of Tile15 can be target of five different banks. Note that in both policies the remapping information is stored in the PFM inside of each bank.

C. *Remapping Comparison with Recent Techniques*

In this section we compare efficiency of remapping for fault recovery in REMEDIATE against state-of-the-art remapping-based techniques including RDC-Cache [5], Salvage Cache [4], Ansari [10], Archipelago [11], and FFT-Cache [12]. In Fig. 5 we report the relative effective cache size for each of these techniques and across various voltage levels. These results obtained using failure rates reported in [13] for a 65nm technology. The results reported in Fig. 5(a) are based on a 1000-run Monte Carlo simulation for a 8MB 8-way set associative LLC cache with two banks and 64 bytes block size. To have a fair comparison, for all these techniques including REMEDIATE, we assume the fault map area overhead is at most 9% of the cache size which equals to the overhead of REMEDIATE with sub-block size of 16-bit. Based on this assumption and to meet the overhead constraint, the Archipelago and FFT-Cache use 16-bit sub-block size. RDC-cache and Salvage-cache use sub-block size for 128-bit and 64-bit, respectively. The results show in Fig. 5(a) show that in ultra-low voltage region (below 400mv) as the failure rate increases, the more flexible remapping methods like FFT-Cache and REMEDIATE result in smaller cache capacity loss.

In Fig. 5(b) we report the results for the same cache configuration but double the number of banks to evaluate the effect of increasing the number of banks on the effective cache size. Some techniques, such as REMEDIATE, benefit from increasing the number of banks while others such as RDC-Cache do not. In fact for the RDC-Cache and Salvage cache the effective cache size for the two bank and four bank cases are fairly similar. For the Salvage cache this is mainly due to its intra-bank remapping style, where additional banks do not provide any more opportunity for remapping. For the Ansari, Archipelago and FFT-Cache techniques that are more scalable than other methods, the improvement is small: comparing two and four banks cases, only up to 5% effective cache size improvement is achieved across different voltage points. For REMEDIATE, we achieve up to 13% improvement in effective cache size compared to the two bank case. In addition, in the Ansari and Archipelago techniques, the fault map area increases noticeably as we reduce the voltage below 375mv. To achieve same effective cache size as REMEDIATE's at such voltage points, Ansari, Archipelago, and FFT-Cache techniques need to rely on a sub-block size of 4-bit or lower, which incurs an overhead of more than 20% for their fault map. Note that Archipelago and Ansari's fault map size is proportional to the number of faulty sets while for REMEDIATE, the amount of fault map area overhead is fixed at 9% across all voltage points. Overall, these previous techniques have noticeably lower flexibility compared to REMEDIATE, where all banks can participate in remapping.

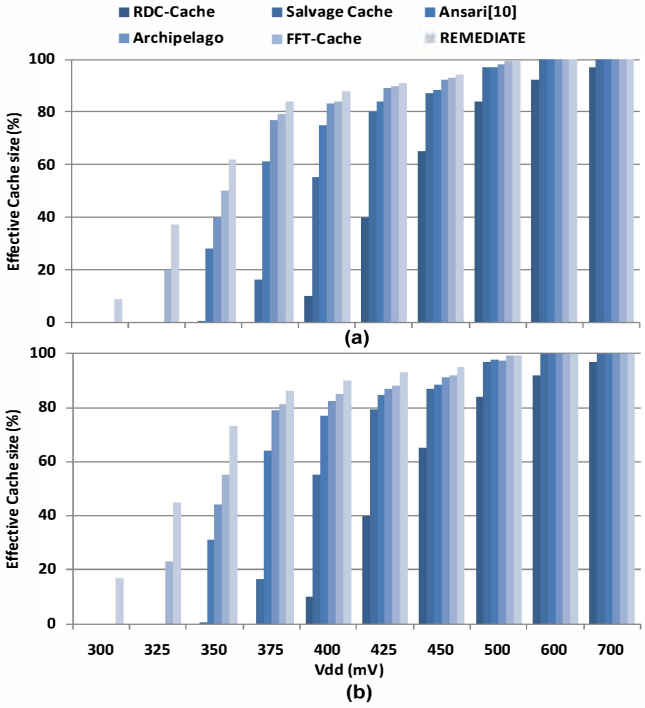

Fig. 5. Effective cache size for different remapping techniques with (a) two banks and (b) four banks of LLC.

VI. Experimental Setup

We use SMTSIM simulator [7] for our performance simulations. To simulate accurate behavior of a NUCA cache architecture and memory network we integrate it with HORNET [8] NoC simulator that simulates NoC architectures with high accuracy and detailed latency, network traffic, and power analysis. To enable power analysis, HORNET combines a dynamic power model based on ORION [23] with a leakage power model. Table I summarizes the configuration parameters used in our study. We use CACTI 6.5 [9] to obtain cache bank access latencies and energy. The dynamic energy consumed by the NUCA cache network within the chip was modeled with the HORNET tool-set. The extra network traffic introduced by our proposal is also taken into account and accurately modeled in the simulator. For estimating cache failure probabilities for different voltage levels, we use the data for failure probability of each SRAM cell for 65nm from [13]. We use same fault model as previous works in remapping-based fault-tolerant cache design [10][11][12].

TABLE I. Baseline Configuration

Processor Cores	16 dual issue cores (out-of-order), 2 GHz
L1 Instruction/Data Cache	8KB, 4-way (LRU), 64 B Blocks, 2 cycle
L2 Cache	64KB, 4-way (LRU), 64 B Block,10 cycle
L3 cache--LLC (shared)	8 MB NUCA, 16 banks, 8-way (LRU), 64 bytes Blocks, 16-bit subblocks, 20 cycle latency (per bank)
Memory Latency	250 cycles
Interconnect	NoC of 2D mesh (4x4 for 16 banks and 16 cores) 32-byte links (2 flits per memory access), 1-cycle link latency, 2-cycle router, XY routing

A. Workload characterization

We compose multi-program workloads consisting of 16 threads. The applications are selected from the SPEC2000 and SPEC2006 benchmark suites, by using a mix of memory-intensive and compute-intensive benchmarks. The sixteen-thread groupings are selected randomly to avoid bias. Table II summarizes our workload mixes. For each application in the mix we fast-forward to skip the initialization phase and then simulate until all threads execute 200 million instructions.

TABLE II. Workload Mix

Workloads	Benchmarks
WL1	sphinx3_06, facerec, applu, sixtrack,astar_06_rivers, applu, fma3d, gromacs_06, vortex_3, milc_06, equake, gobmk_06, bzip2_source,vpr_route, cactusADM_06, soplex_06_ref
WL2	art_470, sjeng_06, perlbench_06_checkspam, mesa, povray_06, omnetpp_06, eon_rushmeier, swim, soplex_06, fma3d, art_470, eon_rushmeier, h264ref_06_sss_encoder, facerec,vpr_route, lucas
WL3	bwaves_06, leslie3d_06,omnetpp_06, mesa, libquantum_06, vortex_3,crafty, perlbench_06, namd_06, povray_06 hmmer_06_nph3,sixtrack, sjeng_06,swim, apsi,applu,
WL4	galgel, apsi, lucas,omnetpp_06, eon_rushmeier, parser, swim,bzip2_source,mgrid, perlbench_06, namd_06,milc_06, equake, bzip2_source,vpr_route, cactus_adm_06
WL5	h264ref_06_sss_encoder_main, hmmer_06_nph3, gap,lbm_06,art_470, sjeng_06, perlbench_06, swim,bzip2_source,art_470, sjeng_06, sjeng_06, mesa, perlbench_06_checkspam, hmmer_06_nph3,sixtrack
WL6	applu, fma3d, gromacs_06, swim,soplex_06_ref, leslie3d_06,omnetpp_06, facerec, galgel, gcc_06, gzip_log,parser,gzip_log,bwaves_06, vortex_3,crafty
WL7	gobmk_06_nngs, galgel,equake, gap, lbm_06, apsi, astar_06_rivers,milc_06, mesa, povray_06, gcc_06, gzip_log,equake, bzip2_source, leslie3d_06, omnetpp_06

B. Design Space Exploration

We study various design parameters to show how REMEDIATE can be effective in improving the fault tolerance for various cache designs. The main design parameters considered in our study include: remapping policy and remapping mode (as discussed in Sections V.A and V.B), bit failure rate of memory SRAM cell for various operating voltage levels, and memory network configurations. The major network configuration parameters we consider include: the number of virtual channel (VC), number of crossbar ports (CP), and bandwidth (BW) of NoC routers. We assume XY routing and study three network configurations based on the router parameters: 1) low-performance (VC=2, BW=1, CP=1), 2) mid-performance (VC=4, BW=2, CP=2), and 3) high-performance (VC=8, BW=4, CP=4). We run our simulation process for eight different failure rates, two proposed remapping policies, and three different network configurations. Note that for all of these simulations we use the baseline configuration as in Table I with 16-bit sub-block size.

VII. RESULTS

In this section we evaluate the impact of REMEDIATE remapping on performance and power in the presence of aggressive voltage scaling. For each workload we report performance in terms of normalized IPC, weighted speedup [25], and LLC network traffic statistics. For normalized IPC and weighted speedup measurements, we use the CMP architecture in which the LLC is operating at nominal Vdd with no fault-tolerant mechanism as the baseline. We also report total power consumption of LLC banks and NoC components (links and routers) for various REMEDIATE remapping policies. In all figures presented in this section we use the following abbreviation for different REMEDIATE mapping techniques, P0: Adjacent Remapping, P1: Global Remapping, Mode M1: one target bank at one hop adjacency, M2: two target banks at one hop adjacency, M3: three target banks at up to two hops adjacency, and M4: four target banks at up to two hops adjacency.

Table III shows the relative disabled cache area after applying our fault-tolerant method. As shown for high probability of failure (low operating voltage), Global policy is far more effective than all adjacent policy modes in preserving cache capacity. For instance, at 450mv, Global Policy reclaims more than 85% of lost cache capacity (i.e., from 99% loss down to 10.5% loss). This is expected since global mapping imposes no location restrictions that would limit the REMEDIATE mapper. Therefore more cache capacity can be saved as more candidates for remapping are available to the REMEDIATE mapper. The same reasoning explains why M4 achieves the highest capacity savings among all adjacent policy modes.

TABLE III. Percentage of disabled cache area using REMEDIATE

Policy	Mode	Bit failure rate (Voltage)							
		7e-8 (0.6)	*2e-6 (0.55)*	*3e-5 (0.5)*	*3e-4 (0.45)*	*7e-4 (0.425)*	*1.5e-3 (0.4)*	*3e-3 (0.375)*	*6e-3 (0.35)*
Adjacent (P0)	*M1*	0.8	6.4	10.9	13.8	22.5	49.2	55.1	78.4
	M2	0.8	6.3	9.8	11.4	17.5	47.5	56.4	70.4
	M3	0.8	6.3	9.6	11.7	17.0	38.4	45.3	65.3
	M4	0.8	6.3	9.6	11.6	16.1	35.0	41.7	62.0
Global (P1)	-	0.7	5.4	7.3	10.5	14.5	20.5	24.2	45.2
No Remap	-	11.8	71	94	99	100	100	100	100

A. Overheads

Figure 6 summarizes the overheads of our scheme over baseline, no matter which remapping policy has been selected. The power overhead in this figure is for the nominal Vdd (0.7V). We account for the overheads of using 8T SRAM cells [13] for protecting the tag and defect map arrays in low-power mode. To reduce the effect of leakage and dynamic power consumption of PFM in high-power mode, we assume clock gating and power gating is applied in the PFM array. Therefore, the main source of dynamic power in nominal Vdd relates to bypass MUXs. As shown in this figure it is trivial and less than 1%. As evident in Fig. 6, the fault map area is the major component of area overhead. The total area overhead is less than 12%. In REMEDIATE, the PFM and MULTIPLEXING layer are on the critical path of LLC cache access. Based on our timing analysis, we consider 2 additional cycles as LLC access latency overhead.

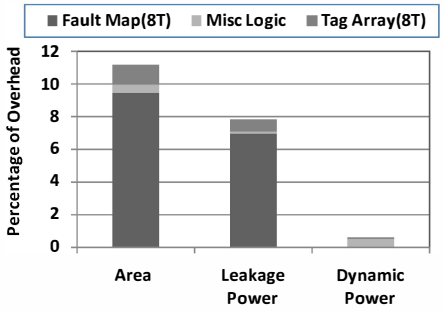

Fig. 6. Area, leakage, and dynamic power overheads of REMEDIATE.

B. *Performance Analysis*

In Fig. 7 we report performance metrics in terms of the weighted speed up and normal arithmetic mean IPC for various workloads and policies in Vdd=0.4V. We observe that P1 demonstrates the worst performance among all polices. We know that this policy saves more cache capacity than other policies (Table III) and also increases the network traffic and access latency as depicted in Fig.8(b). In fact, at this voltage network latency and traffic have more impact on performance than cache capacity. Also, we observe that in adjacent remapping policy (P0), by increasing the number of target blocks from one to three (M1 to M3) which comes with saving more cache capacity (Table III), both IPC and speedup are increased. However, there is a falloff in performance for P0M4 that similar to P1 policy, at this point the impact of network traffic and latency is more affecting the performance than cache capacity (Fig. 7).

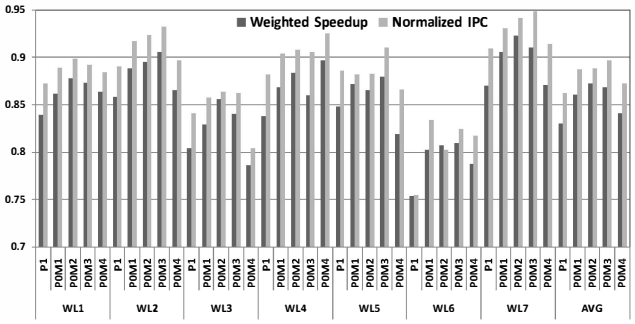

Fig. 7. Normalized performance results of four selected policies for different workloads in Vdd=400mV.

Figure 8(a) shows the number of LLC accesses for different policies. Obviously, the policies that have more target options for remapping have higher accesses. Also, decreasing the voltage causes more remapping which leads to higher cache accesses. However, going below 0.4 V causes a large section of the cache to be disabled which leads to decrease in the number of accesses.

In Fig. 8(b) we report the average packet latency (excluding LLC bank hit latency) across different voltage levels. A clear trend seen in

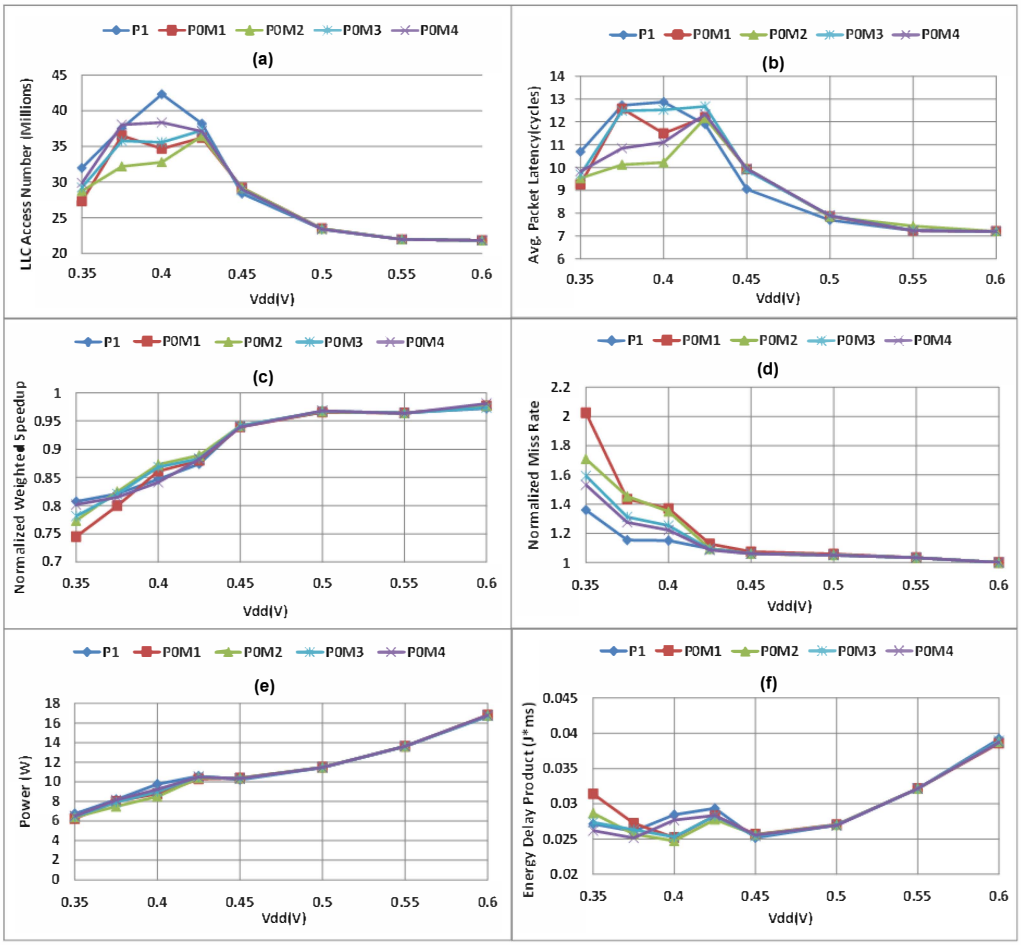

Fig. 8. (a) LLC average number of accesses, (b) Average flit latency (excluding LLC bank latency), (c) Normalized miss rate, (d) Weighted speedup, (e) Total power (network + LLC banks), (f) EDP for various REMEDIATE policies and across different voltage levels.

this figure is the increase in flit latency as the voltage scales down below 0.50 V. Above that point there is not a noticeable difference in flit latency as the probability of failure and the lost cache capacity due to voltage scaling is pretty small (results in Table III). Fig. 8(c) shows the normalized speedup results as performance evaluation of different policies. As we can see, performance across various workloads and policies drops significantly for voltage lower than 0.45V because of large cache capacity loss and higher access latency. Another interesting observation is how the various policies impact performance at different voltage levels. In *mid-range voltages* (0.4~0.45V), P0M4 impacts performance more than other adjacent policies. In fact, although P0M4 saves more cache capacity compared to P0M1-P0M3, it degrades performance as it increases network traffic and the total number of LLC accesses, as reported in Fig. 8(a). So, different modes of P0 policy represent a tradeoff between effective cache capacity and network traffic and latency.

At the lowest voltage level we observe a different behavior; P1 policy can save larger cache capacity compared to other policies and has the lowest impact on the miss rate. However, the network overhead in terms of number of LLC bank access and average flit latency, in this policy is higher than other policies. For the *mid-range voltages* we observe a large variation in network traffic (Fig.8(a) and (b)) for various policies. For *high voltages* (above 0.450V) and *low voltages* (below 0.4V) we observe a small variation in network traffic. This can be explained as follows: for high voltages there is not much opportunity for REMEDIATE remapping as the probability of failure is small. For low voltages, due to very high probability of failure, a large portion of LLC cache is being disabled, which gives a small remapping opportunity for REMEDIATE. Overall, global mapping policy (P1) results in highest capacity savings and lowest LLC cache miss rate (reported in Fig. 8(d)). However, it comes with highest number of LLC accesses and network latency. For adjacent policies (P0M1-P0M4), as we allow more banks available for remapping, a higher cache capacity is saved (Table III) which results in lowering the LLC miss rate and increasing the network overhead.

C. Power and Energy-Delay Analysis

In Fig.8(e), we report total power include both memory and network power for various voltage levels and policies. Reducing the voltage comes with exponential increase in failure rate. Therefore, reducing voltage causes more remapping of faulty areas by REMEDIATE and leads to higher network traffic and a larger number of LLC cache accesses (Fig.8(a) and (b)). In spite of that, the overall power reduces, as dynamic power is reduced quadratically and leakage power reduces linearly with voltage scaling. Unlike memory banks, the network power does not change noticeably. Since the cache banks are major source of power consumption in LLC, the voltage scaling is only applied to the banks and not the interconnect subsystem. Moreover, applying the voltage scaling to the interconnect network is more complex and comes with non-trivial performance degradation. As we lower the voltage below 0.450 V, the policies with higher network traffic and higher number of LLC cache accesses dissipate slightly higher power than others.

The energy-delay product (EDP) results are shown in Fig.8(f). For *high voltages*, EDP reduces significantly as voltage scales down. For *low voltages*, EDP increases as the performance degrades significantly. For mid-range voltages, we observe a good trade-off between power saving and performance loss (Fig.8(d) and (e)), achieving the lowest energy delay product.

D. Network Analysis

In this section we analyze the impact of network configuration on power and performance of our proposed policies. We report the results for three separate router configurations, including High-Performance (HP), Moderate-Performance (MP) network (our baseline), and Low-Performance (LP) as explained in Section VI.B. Figure 9 shows the effect of router configuration on performance and power results for P1 policy. As we lower the voltage, we observe smaller performance impact in MP and HP network. In fact, these routers have higher capacity to tolerate higher network traffic. For voltage points in 0.375~0.425V, the performance gap in terms of weighted speedup and average packet latency between low performance and high performance network is widening. This is expected since in this voltage range applying REMEDIATE remapping policies results in highest network traffic as shown in Fig. 9(a). Hence, a low performance network can severely degrade performance. The power results reported in Fig. 9(c), indicate the network with high performance routers has higher power dissipation as compared to other network configurations.

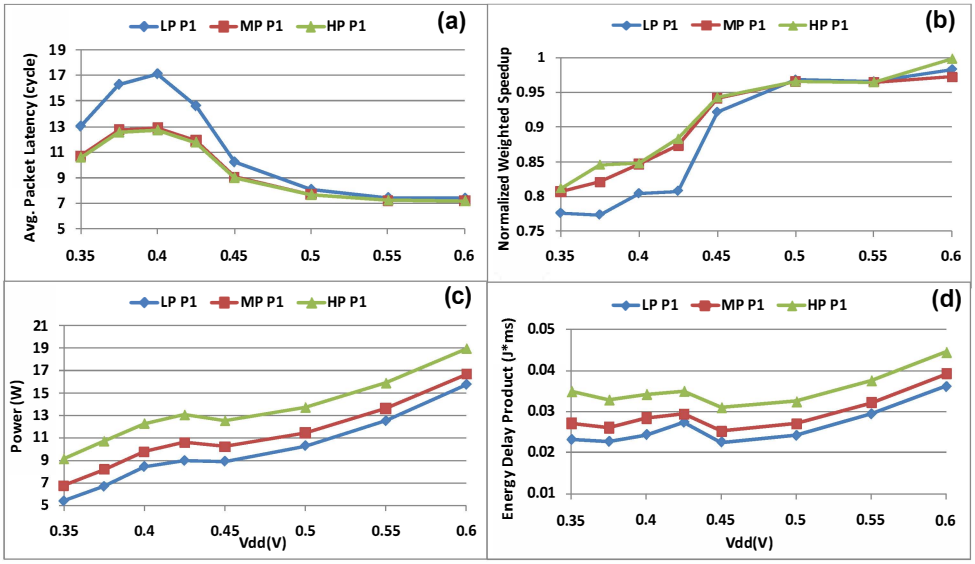

Fig. 9. (a) Average flit latency (excluding LLC bank latency); (b) Weighted speedup; (c) Total power (network + LLC banks); (d) EDP for various network configurations and across different voltage levels.

E. Quantitative Comparison to Alternative Techniques

In order to illustrate the benefits of our design, we quantitatively compare REMEDIATE with the baseline 6T SRAM cell, two recent multi-bit ECC-based techniques (2D ECC [18] and MS-ECC [17]), three recent remapping-based techniques (Ansari[10], FFT-Cache[12], and Archipelago[11]), and two other state of the art works (Bit-fix[24] and 10T SRAM cell[14]). Table IV summarizes this comparison based on the minimum achievable Vdd, area overhead for the caches, power overhead, normalized IPC, and normalized power. In order to have a fair comparison, the remapping configuration and coding granularities are set so that the coding overheads of the ECC-based techniques are equal/comparable to disabled capacity of other methods (i.e 25%). In this table, different techniques are sorted based on their minimum achievable Vdd, when targeting 99.9% yield for on-chip caches.

TABLE IV. Comparison of different cache protection Schemes

Scheme	Vdd-min (mV)	Area over. (%)	Power over. (%)	Norm. IPC	Power norm. to REMEDIATE
6T cell	660	0	0	1.0	4.51
2D ECC [18]	470	6.5	15	0.96	1.82
MS-ECC [17]	440	6	10	0.90	1.61
Bit-fix [24]	420	8	20	0.89	1.41
Ansari [10]	410	15	8	0.96	1.33
10T cell [14]	380	66	24	1.0	1.24
FFT-Cache [12]	375	10	8	0.95	1.21
Archipelago [11]	370	12	7	0.95	1.18
REMEDIATE	360	11	8	0.95	1.0

Overall, REMEDIATE achieves the lowest operating voltage (360mv) and the highest power reduction compared to all other techniques. Since ECC-based techniques cannot tolerate high failure rates in very low voltages, their minimum Vdd is higher than remapping based methods. The closest techniques to ours are Archipelago, FFT-Cache, and 10T cell. However, 10T cell incurs a 66% area overhead and 24% power which are much more than our method overheads. Comparing to FFT-Cache and Archipelago, overheads are almost equal, but our scheme can achieve a lower Vdd and higher power saving. Overall, the scalability of REMEDIATE for large shared NUCA caches along with efficient remapping, high configurability, and inherent flexibility allows it to tolerate higher failure rates compared to other similar techniques.

VIII. Conclusion

The design of NUCA LLC in CMP architectures is challenging due to the often conflicting tradeoffs of reliability, manageable power, and performance. In this work, we proposed REMEDIATE, a fault-tolerant scalable cache architecture for shared NUCA LLC in tiled CMPs. REMEDIATE leverages address remapping to replicate faulty blocks of shared LLC banks with blocks from other banks. It utilizes two remapping policies for efficient selection of redundancy from different cache banks considering design challenges in a NUCA memory organization. Our experimental analysis shows that as cache operating voltage scales down, REMEDIATE increases available cache capacity and hence maintains performance even in the presence of high failure rates. We show that REMEDIATE is most effective in lowering power and EDP in the mid-range voltages (0.375 ~ 0.450 V). Our results indicate that REMEDIATE saves up to 50% power consumption of LLC cache in a 4x4 CMP architecture operating below 0.4V while recovering up to 80% of the faulty cache capacity with only modest performance degradation.

Acknowledgment

This work was partially supported by NSF Variability Expedition Grant Number CCF-1029783.

References

[1] S. R. Nassif, et al. A Resilience Roadmap. In *Proc. DATE*, 2010.

[2] C. Kim, D. Burger, and S. W. Keckler. An adaptive, non-uniform cache structure for wire-delay dominated on-chip caches. In *ASPLOS*, 2002.

[3] Beckmann, B.M. and Wood, D.A. Managing Wire Delay in Large Chip-Multiprocessor Caches. In *Proc. MICRO*, 2004.

[4] C.-K. Koh, et al. The Salvage Cache: A fault-tolerant cache architecture for next-generation memory technologies. In *Proc. ICCD*, 2009.

[5] A. Sasan, et al. A fault tolerant cache architecture for sub 500mv operation: resizable data composer cache (RDC-Cache). In *Proc. CASES*, 2009.

[6] Y. Wang, et al. Address remapping for static NUCA in NoC-based degradable chip-multiprocessors. In *Proc. PRDC*, 2010.

[7] D.M. Tullsen. Simulation and Modeling of a Simultaneous Multithreading Processor. In *Proc. ACMGC*, 1996.

[8] Mieszko Lis, et al. Scalable, Accurate Multicore Simulation in the 1000-core era. In *Proc. ISPASS*, 2011.

[9] N. Muralimanohar, R. Balasubramonian, and N.P. Jouppi. CACTI 6.5. *HP Laboratories*, Technical Report, 2009.

[10] A. Ansari, et al. Enabling ultra low voltage system operation by tolerating on-chip cache failures. In *Proc. ISLPED*, 2009.

[11] A. Ansari, et al. Archipelago: A polymorphic cache design for enabling robust near-threshold operation. In *Proc. HPCA*, pages 539-550, 2011.

[12] A. BanaiyanMofrad, et al. FFT-Cache: A Flexible Fault-Tolerant Cache Architecture for Ultra Low Voltage Operation. In *Proc. CASES*, 2011.

[13] G. Chen, D. Sylvester, D. Blaauw, and T. Mudge. Yield-driven near-threshold sram design. *IEEE TVLSI*, 18(11):1590–1598, 2010.

[14] B. Calhoun and A. Chandrakasan. A 256 kb sub-threshold SRAM in 65nm cmos. In *Proc. ISSCC*, pages 240.251, 2006.

[15] W. Klotz. Graph coloring algorithms, 2002. Mathematik-Bericht 5, Clausthal University of Technology, Clausthal, Germany.

[16] H. Al-Omari and K. Sabri. New graph coloring algorithms. *Am. J. Math. & Stat.*, 2(4):739–741, 2006.

[17] Z. Chishti, et al. Improving cache lifetime reliability at ultra-low voltages. In *Proc. MICRO*, 2009.

[18] J. Kim, et al. Multi-bit Error Tolerant Caches Using Two-Dimensional Error Coding. In *Proc. MICRO*, 2007.

[19] D. Roberts, et al. On-chip cache device scaling limits and effective fault repair techniques in future nanoscale technology. In *Proc. DSD*, 2007.

[20] C. K. Koh, et al. Tolerating process variations in large, set associative caches: The buddy cache. *ACM TACO*, 6(2):1–34, June 2009.

[21] D. Gizopoulos, et al. Architectures for online error detection and recovery in multicore processors. In *Proc. DATE*, 2011.

[22] C. Chen and M. Hsiao. Error-correcting codes for semiconductor memory applications: A state of the art review. *IBM J. of R&D*, 1984.

[23] H. Wang, X. Zhu, L. Peh, and S. Malik. Orion: a power-performance simulator for interconnection networks. In *Proc. MICRO*, 2002.

[24] C. Wilkerson, et al. Trading Off Cache Capacity for Reliability to Enable Low Voltage Operation. In *Proc. ISCA*, June 2008.

[25] A. Snavely and D. M. Tullsen. Symbiotic job scheduling for a simultaneous multithreading architecture. In *Proc. ASPLOS*, 2000.

[26] A. Sasan, et al. Process variation aware sram/cache for aggressive voltage-frequency scaling. In *Proc. DATE*, 2009.

[27] A. Chakraborty, et al. E< MC^2: less energy through multi-copy cache. In *Proc. CASES*, 2010.

[28] A. Sasan, et al. Inquisitive defect cache: a means of combating manufacturing induced process variation. *IEEE TVLSI*, 19(9):1597-1609, 2011.

[29] A. Sasan, et al. Variation Trained Drowsy Cache (VTD-Cache): A History Trained Variation Aware Drowsy Cache for Fine Grain Voltage Scaling. *IEEE TVLSI*, 20(4): 630-642, 2012.

[30] A. Sasan, et al. History & Variation Trained Cache (HVT-Cache): A process variation aware and fine grain voltage scalable cache with active access history monitoring. In *Proc. ISQED*, 2012.

A. BanaiyanMofrad, Gustavo Girao, and Nikil Dutt. A Novel NoC–based Design for fault-tolerance of Last-level Caches in CMPs. In *Proc. CODES+ISSS*, 2012.