NUCA-L1: A Non-Uniform Access Latency Level-1 Cache Architecture for Multicores Operating at Near-Threshold Voltages

FARRUKH HIJAZ and OMER KHAN, University of Connecticut

Research has shown that operating in the near-threshold region is expected to provide up to $10\times$ energy efficiency for future processors. However, reliable operation below a minimum voltage (V_{ccmin}) cannot be guaranteed due to process variations. Because SRAM margins can easily be violated at near-threshold voltages, their bit-cell failure rates are expected to rise steeply. Multicore processors rely on fast private L1 caches to exploit data locality and achieve high performance. In the presence of high bit-cell fault rates, traditionally an L1 cache either sacrifices capacity or incurs additional latency to correct the faults. We observe that L1 cache sensitivity to hit latency offers a design trade-off between capacity and latency. When fault rate is high at extreme V_{ccmin} , it is beneficial to recover L1 cache capacity, even if it comes at the cost of additional latency. However, at low fault rates, the additional constant latency to recover cache capacity degrades performance. With this trade-off in mind, we propose a Non-Uniform Cache Access L1 architecture (NUCA-L1) that avoids additional latency on accesses to fault-free cache lines. To mitigate the capacity bottleneck, it deploys a correction mechanism to recover capacity at the cost of additional latency. Using extensive simulations of a 64-core multicore, we demonstrate that at various bit-cell fault rates, our proposed private NUCA-L1 cache architecture performs better than state-of-the-art schemes, along with a significant reduction in energy consumption.

Categories and Subject Descriptors: C.2.2 [Computer-Computer Architecture]: Cache

General Terms: Design, Performance

Additional Key Words and Phrases: NTV

ACM Reference Format:

Farrukh Hijaz and Omer Khan. 2014. NUCA-L1: A non-uniform access latency level-1 cache architecture for multicores operating at near-threshold voltages. *ACM Trans. Architect. Code Optim.* 11, 3, Article 29 (October 2014), 28 pages.

DOI: <http://dx.doi.org/10.1145/2631918>

1. INTRODUCTION

Complex uniprocessors have hit the “power wall,” and multicores with simpler cores have emerged as the alternative. Multicores exploit concurrency to achieve performance and rely on the simplicity of design and operation of each core to achieve energy efficiency. However, with the integration of many cores on a single die, future multicores will still be constrained by their energy efficiency [Yelick 2009]. As energy quadratically scales with voltage, extreme voltage scaling can deliver energy-efficient processors [Chandrakasan et al. 2010]. However, reliable circuit operation cannot be guaranteed below a minimum voltage, V_{ccmin} , as the effects of process, voltage, and temperature (PVT) variations become predominant and the hardware components may start to fail. Despite the reliability issue, operating at near-threshold voltage (NTV) is an attractive

Authors' addresses: F. Hijaz and O. Khan, Department of Electrical and Computer Engineering, University of Connecticut, Storrs, CT; emails: farrukh.hijaz@uconn.edu; khan@uconn.edu.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.

© 2014 ACM 1544-3566/2014/10-ART29 \$15.00

DOI: <http://dx.doi.org/10.1145/2631918>

solution because it has the potential to deliver up to $10\times$ energy reduction and has been proven to be the most energy-efficient region in which to operate [Kaul et al. 2012; Dreslinski et al. 2013]. Logic elements are somewhat resilient to PVT variations because they can compensate across long logical paths. On the other hand, SRAM memory elements pose a critical limitation in low-voltage operating conditions, as their functionality margins are lower because of their aggressively sized transistors and architectural requirements to maximize array size for area efficiency [Chandrakasan et al. 2010]. Therefore, SRAM bit-cells are especially vulnerable to the PVT variations at NTV conditions, causing their margins to be easily violated.

The higher demand for on-chip multicore cache has been steadily increasing to alleviate expensive off-chip accesses. A private last-level cache (LLC) organization (e.g., Conway et al. [2010]) has low hit latency due to high data locality. However, its off-chip miss rate is high in workloads with large private working set and/or high degree of data sharing. A popular cache organization is to implement per-core fast private caches backed by a logically shared (physically distributed) LLC to minimize the off-chip miss rate [Kim et al. 2002]. The varying latency to access the shared LLC naturally gives rise to non-uniform cache access (NUCA) [Kim et al. 2002]. Although the shared-LLC organization enables large on-chip cache capacity, the average LLC access latency is considerably higher than a private-LLC system. Therefore, tiled multicores rely heavily on their *low-latency* private caches for common case optimizations and are relatively insensitive to the LLC latency [Hijaz et al. 2013].

In the context of NTV operation, error-correcting codes (ECCs) have been proposed to protect the LLC. However, private caches have been left unprotected, limiting their operation at lower voltages [Kaul et al. 2012]. Since future multicores will need to operate at NTV for energy efficiency, protecting private caches against the SRAM bit-cell faults is increasingly becoming critical. We conducted an experiment to quantify the performance sensitivity to variations in L1 and L2 cache access latencies in a 64-core private-L1, shared-L2 cache organization. The results show greater than 10% performance loss when L1 hit latency is increased from one to two or more cycles. On the other hand, when L1 hit latency is kept constant at one cycle and the latency of LLC slice (L2 cache) is increased, the performance only degrades by less than 2%. Motivated by this, we propose to focus on the latency-sensitive private-L1 cache operation at NTV. We assume that each LLC slice is protected using a scheme that sacrifices its access latency (e.g., by correcting bit-cell faults or combining multiple data blocks to recover the cache capacity [Alameldeen et al. 2011; Ansari et al. 2011]).

NTV proposals that set frequency and voltage such that the processor has zero faults have been explored. This approach delivers reliable operation without increasing hardware complexity. However, it either operates considerably above the near-threshold voltage [Karpuzcu et al. 2013] and hence does not fully exploit the energy efficiency, or runs at a low enough frequency to ensure zero faults [Dreslinski et al. 2013], degrading performance substantially. Another widely explored approach is to allow bit-cell faults to exist at NTV and fix them during design time or at runtime [Chandrakasan et al. 2010; Alameldeen et al. 2011; Ansari et al. 2011]. This approach ensures higher energy efficiency by operating near the threshold voltage and acceptable performance by setting the frequency higher than a safe frequency. Because the system now operates at a higher than safe frequency in the NTV region, the timing margins of SRAM bit-cells may be violated.

To mitigate the effect of rising SRAM bit-cell faults at near-threshold voltages, a private-L1 cache can implement three possible mechanisms. The first category relies on circuit-level techniques that up-size the transistors or implement a more robust SRAM cell (e.g., 8-T or 10-T SRAM cell instead of the traditional 6-T version) [Liu and Kursun 2007; Moradi et al. 2008; Morita et al. 2007; Chen et al. 2007; Maric

et al. 2013; Kulkarni et al. 2007]. This results in an increase in area (33% to 100%) of the SRAM cell, reducing the effective cache capacity within a given area budget. The second category deals with the high number of random bit-cell faults using error-correcting (ECC) techniques [Chen and Hsiao 1984; Kim et al. 2007; Yoon and Erez 2009; Chishti et al. 2009; Miller et al. 2010b; Alameldeen et al. 2011]. These techniques increase the available cache capacity by correcting one or more bit-cell faults but suffer from a constant latency overhead of one or more cycles for error detection and correction. As discussed previously, the private-L1 cache performance is sensitive to its access latency; therefore, traditional ECC-based techniques can quickly become ineffective for private caches at near-threshold voltages. The third category consists of architectural techniques [Wilkerson et al. 2008; Roberts et al. 2007; Abella et al. 2009, 2011; Ansari et al. 2011]. These techniques patch up and recover cache capacity at the fine granularity of subcache lines. These techniques add a constant latency overhead of up to three cycles in addition to disabling a portion of cache. Cache line disabling [Hijaz et al. 2012, 2013; Alameldeen et al. 2011] is a technique that does not incur additional latency but is limited by the amount of capacity that it can recover. Word-level disabling [Abella et al. 2009] works at a finer granularity, resulting in high available capacity. However, it evicts the cache line on an access to a faulty word. This generates excessive network traffic, resulting in significant loss in performance and energy efficiency.

The persistent bit-cell faults at nominal down to near-threshold voltages can be classified using the memory built-in self-test (MBIST) mechanism commonly deployed in commercial processors [Franklin and Saluja 1990]. Utilizing this a priori knowledge of bit-cell faults at NTV, it is possible to encode the number of faults per cache line (or even per word), and capture this information in the cache's tag array. We propose to deploy this mechanism and design a non-uniform access latency private cache architecture (NUCA-L1). The key idea is to access fault-free cache lines with no additional hit latency. For cache lines with one bit-cell fault, we propose to correct the fault site by implementing an error-correcting mechanism. The trade-off is the additional hit latency for accesses to such cache lines. Finally, for cache lines with faults that cannot be corrected (e.g., two or more bit-cell faults), we propose to disable them by disallowing allocation of such cache lines. We also propose NUCA-L1 cache variants that allow fine-grain correction and/or disabling capabilities.

We quantitatively compare our NUCA-L1 cache architecture and its variants to bit-fix [Wilkerson et al. 2008], word-disable [Wilkerson et al. 2008; Roberts et al. 2007], Archipelago [Ansari et al. 2011], cache-line-disable, single error correction double error detection (SECDED) with cache-line-disable, double error correction triple error detection (DECTED) with cache-line-disable, word-level disabling [Abella et al. 2009], and SECDED with word-level disabling. We show that our proposed NUCA-L1 architecture not only exploits the latency and capacity trade-off but also adapts seamlessly to varying bit-cell fault rates. To the best of our knowledge, this is the first proposal that combines the latency and capacity trade-offs for a private L1 cache operating at nominal voltage all the way down to the NTV region.

The novel contributions of this article are as follows:

- (1) A variable access latency private L1 cache architecture that exploits the latency and capacity trade-offs by avoiding additional hit latency for fault-free cache lines while only applying error correction/patch-up-and-recovery mechanism to faulty but correctable cache lines. The cache lines that are not correctable are disabled for allocation, resulting in an effective architecture for NTV operation.
- (2) The NUCA-L1 architecture seamlessly adapts to NTV conditions with different bit-cell fault rates. It effectively exploits the available capacity to deliver performance

and energy close to the ideal fault-free baseline by keeping the hit rate high and hit latency low.

- (3) Our proposed architecture performs 4% and 11% better than the best-performing ECC mechanism at high and low fault rates, respectively. It also performs on-par with state-of-the-art Archipelago scheme at high fault rate; however, it performs 11% better at low fault rate.
- (4) Our proposed architecture's energy is within 5% and 16% of the fault-free baseline system at low and high fault rates, respectively. It also consumes 14.5% and 3% less energy than Archipelago at low and high fault rates, respectively.

2. BACKGROUND

The earlier approaches to improve cache reliability have used true row/column-level redundancy by adding spare rows/columns to the cache array [Schuster 1978]. These approaches remap a row/column with bit defects to a functional spare row/column. Although this coarse-grain approach is effective at voltages with low fault rates, albeit with significant overhead, it cannot deal with thousands of random bit faults. Operating in the NTV region requires more robust solutions that are capable of dealing with thousands of random faults. These existing solutions are classified in the following categories.

2.1. Circuit-Level Solutions

Several circuit-level solutions have been proposed to operate reliably at low voltages. These solutions span two different dimensions. The first dimension covers designing SRAM bit-cells by up-sizing transistors to achieve higher functionality margins [Liu and Kursun 2007; Moradi et al. 2008; Morita et al. 2007; Chen et al. 2007]. The second dimension covers designing more robust SRAM cells (e.g., Schmitt trigger (ST) SRAM cell [Kulkarni et al. 2007]) to ensure improved performance at low fault rates. These approaches are more tolerant to different sources of parameter variations compared to the conventional 6T cell and allow reliable operation of the cache at lower voltages. However, larger SRAM cells result in higher leakage current in both low-voltage and high-voltage operation of the cache, as it is directly proportional to the number of transistors. Furthermore, these approaches incur significant area overhead (e.g., 100% area overhead for the ST SRAM cell). Therefore, the effective cache space available is reduced at a given area budget.

2.2. ECC-Based Solutions

Correcting persistent faults using error detection and correction schemes [Chen and Hsiao 1984] is a popular mechanism to enable NTV operation. Parity is the most popular design choice for L1 caches to detect bit errors. This scheme is easy to implement and is useful to detect an occasional soft error at high voltages [Gallager 1962]. However, it does not have any correction capability and is not suitable for caches operating at near-threshold voltages with many permanent bit-cell failures. Similarly, conventional SECDED provides single-bit correction capability but cannot operate at high fault rates with lots of multibit faults. Caches with conventional SECDED need to be augmented with cache line disabling to make it work at high bit fault rates [Hijaz et al. 2013]. Increasing the strength of ECC (DECTED and higher) can provide better correction capability, enabling more aggressive voltage scaling. However, this comes at a price of higher area and latency overhead.

Two-dimensional ECC schemes have been proposed previously to correct faults [Calingaert 1961]. Kim et al. [2007] proposed one such two-dimensional ECC scheme to effectively correct clustered multibit errors. The scheme works well for protection against soft errors but is not effective for situations with persistent faults, as it has

a high overhead for every cache write. Yoon and Erez [2009] proposed a two-tier ECC scheme for the LLC. Tier-2 ECC check bits are stored in cacheable DRAM memory space and is only fetched from DRAM if there is a bit error in a dirty line, which makes it very efficient for protection against soft errors. However, it cannot tolerate process variation induced high bit error rate at near-threshold voltages due to excessive cache pollution and off-chip communication. Chishti et al. [2009] proposed a multibit segmented ECC (MS-ECC) scheme, capable of correcting bit errors at a subcache line granularity. This scheme sacrifices 50% of area to store ECC check bits, which makes it ineffective for NTV operation. Alameldeen et al. [2011] presented a cache architecture based on variable-strength ECC (VS-ECC). Their scheme uses SECDED in the normal case and stronger ECC for cache lines with multibit errors. The resources for multibit correction are shared among a set, thereby reducing the area overhead. The higher latency of stronger ECC makes this scheme prohibitive for use in L1 caches, especially at near-threshold voltage with high bit error rate. All ECC-based schemes add constant latency to the L1 cache that limit their use at near-threshold voltages. SECDED is the simplest ECC scheme and incurs one additional cycle on top of L1 hit latency. The closest more complex scheme is DECTED, which requires two additional cycles to correct errors. We show in Section 5.3 that although DECTED recovers more capacity, the overall performance still degrades because of its higher access latency. Therefore, the previously mentioned schemes that require even higher latency to correct bit-cell faults would not be effective at NTV for L1 caches.

2.3. Architectural Patch-Up and Recovery Solutions

Techniques that leverage disabling portions of a cache at a coarse granularity to save power have been studied before [Albonesi 1999; Yang et al. 2001, 2002]. These coarse-grain techniques disable ways or sets (or a combination of both), which makes it impractical for NTV operation, as the high number of random faults could result in all of the cache being disabled.

Wilkerson et al. [2008] proposed two fine-grain subcache line-level disabling techniques for reliable low voltage operation of L1 caches, namely bit-fix and word-disable. Bit-fix sacrifices 25% of the cache capacity to correct bit faults in rest of the lines. This scheme incurs three-cycle additional latency to correct persistent faults, which is too excessive to even expect reasonable performance in the context of private L1 cache. Word-disable, on the other hand, sacrifices 50% of its capacity to correct persistent faults in rest of the cache. This scheme incurs one-cycle additional latency. Both of these schemes can only operate at low fault rates and cannot tolerate high fault rates. The reason being that it might not have enough fault-free sub-blocks to recover the rest of the ways. Roberts et al. [2007] also proposed a technique similar to word-disable.

Abella et al. [2009] proposed a sub-block-level disabling scheme that disables faulty words within a cache line. Accesses to these faulty words are treated as misses and the cache line is evicted, writing back only the valid words in that cache line. Accesses to fault-free words go through normally. When implemented on top of a parity protected cache, this scheme results in a high number of evictions, degrading the performance substantially. Implementing the scheme on an ECC-protected cache can recover most of the capacity; however, it suffers from the constant latency overhead of the ECC scheme. In addition, ECC needs to be implemented on a word level, resulting in a high storage overhead.

Ansari et al. [2011] proposed a fine-grain patch-up and recovery mechanism to enable NTV operation of private caches. In this multibank cache technique, collision-free groups (islands) of varying sizes can be formed using within cache flexible group formation. Each group contains one sacrificial set that is used to correct the other sets in that group. This scheme loses capacity due to sacrificial sets. It can also increase

the contention on some of the cache lines because it remaps the sacrificial sets to other working sets, resulting in performance loss. The cache access in this technique can be divided into three steps. The memory map is read to figure out the index for accessing the cache. After that, the cache and the fault map are accessed in parallel. Finally, the multiplexing layer assemble the requested data based on the information from the fault map. The L1 cache needs at least one additional cycle to complete these three steps.

2.4. Motivation for NUCA-L1 Architecture

Multicore processors rely on the low-latency private L1 caches for performance. The L1 cache has a high hit rate, and any increase in latency degrades performance significantly. An ideal solution would keep the latency of the private L1 caches as low as possible. Multicores also rely heavily on the capacity of the private L1 caches to exploit spatiotemporal locality. When the active working set of a workload is large, the limited L1 cache size increases its miss rate. This becomes even more pronounced at high fault rates, when the available capacity is low. Conventional correction/patch-up and recovery techniques can help increase the available capacity. However, performance suffers because of the additional latency for error correction/recovery. Cache line/word-level disabling optimizes for latency but negatively impacts the L1 cache hit rate.

Keeping the importance of both lower latency and higher available capacity in mind, we propose the NUCA-L1 cache architecture. Our proposal relies on the a priori knowledge of faulty cache lines at a given operating voltage. The intuition of applying correction or patch-up and recovery mechanism to only faulty cache lines is that fault-free lines need not incur additional latency. This way, most of the cache accesses occur with lowest hit latency. Furthermore, a correction or patch-up and recovery scheme is applied to the faulty cache lines to recover cache capacity. Although this recovered capacity comes at a higher latency, it is beneficial at higher fault rates.

3. NUCA-L1 ARCHITECTURE

3.1. Proposed Architecture

The main idea behind the proposed private L1 cache architecture (NUCA-L1) is to optimize for lower latency and incur no additional latency for fault-free cache lines. To remove the capacity bottleneck, a correction scheme is applied to the faulty cache lines to enable higher available capacity. Two disable bits are added to the tag entry for each cache line to enable the identification of fault-free, single-bit, and multibit faults. All cache lines identified as containing faults that cannot be corrected are permanently disabled and not allocated in the L1 cache. To avoid any additional latency for fault-free cache lines, the correction logic is bypassed. Cache lines that can be corrected are dealt with using a correction or patch-up and recovery mechanism. We assume that the tag array is protected using circuit techniques discussed in Section 2.1.

In this article, we implement SECDED as the correction scheme. The cache accesses to faulty cache lines take two hit cycles, and all faulty cache lines that cannot be corrected incur private cache miss latency. In the first cycle, NUCA-L1 reads the tag array to evaluate a hit and read the disable bits. If it is a hit and the disable bits identify the cache line as fault-free, the cache line is accessed in a normal way. If it is a hit but the access is to a cache line with single-bit fault, the SECDED scheme is used in the second cycle to correct the faulty bit. The hit path is modified to enable selective multiplexing of the SECDED logic. In Section 3.4.1, we discuss possible extensions of our proposed architecture to support other bit-cell correction schemes.

Permanently disabled cache lines are not allocated by the replacement/allocation logic. Sets with at least one working way are allocated as usual. The probability of all

ways in a set being permanently disabled is very low, but that scenario could arise at high fault rates. If this scenario does arise, we store the cache line in a special structure called the *resilient buffer*. The resilient buffer is a fully associative structure, in which each entry can store a cache line. Resilient buffer is accessed in parallel to the NUCA-L1 cache and stores data for sets that have all of their ways permanently disabled. More details about this buffer are discussed in Section 3.3.2.

As an alternative, a fine-grain disabling mechanism at the word level can be deployed [Abella et al. 2009]. A single bit per word (eight bits per cache line) is now required to classify each word within the cache line as disabled or not. This increases the storage overhead but can result in higher available NUCA-L1 capacity. Word-level disable mechanism does not require any changes to the replacement/allocation logic. However, an access to a word with bit-cell fault(s) is treated as a miss, and eviction is forced on the associated cache line to bring the requested word from the lower-level L2 cache. Moreover, this mechanism complicates the L2 cache since word-level write enables are required for write-backs from the L1 cache. Our default NUCA-L1 architecture assumes cache line disabling; however, variations that use word-level disabling are discussed in Section 3.4.

3.2. Architecture Operation: Functional Cache Lines

Functional cache lines are defined as those that are fault-free or contain a single-bit fault. When a core makes a request, the tag array is looked up to evaluate a hit and the disable bits are read out. In the following subsections, we describe how a read and a write request is completed in different scenarios in our proposed NUCA-L1 architecture (Figure 1).

3.2.1. Write Request to a Fault-Free Cache Line. In the case where the write request results in a hit and the disable bits identify the cache line as fault-free, the word is written to the cache line. The bypass multiplexers introduced in the hit path make sure that the access goes through the normal path. The disable bit is used as a select line for the multiplexers to route data through the normal hit path. Therefore, write requests to fault-free cache lines do not incur any additional latency on a cache hit.

In case of a miss, the cache line is brought in from a lower-level L2 cache and inserted in the NUCA-L1 cache. The word is then written to it through the normal data path.

3.2.2. Read Request to a Fault-Free Cache Line. In the case where the read request results in a hit and the disable bits identify the cache line as fault-free, the requested word is read from the cache and returned to the core. The bypass multiplexers introduced in the hit path make sure that the access goes through the normal path. Therefore, a read request to fault-free cache lines does not incur any additional latency.

In case of a miss, the cache line is brought in from a lower-level L2 cache and inserted in the NUCA-L1 cache. The requested word is also returned to the core through the normal data path.

3.2.3. Write Request to a Cache Line with Single-Bit Fault. If the write access is a hit and the disable bits identify the cache line as containing single-bit fault, the whole cache line is read out of the cache. The select line (the disable bit) to the bypass multiplexing logic is now set to one. This ensures that the cache line is forwarded to the SECDED encoder. The disable bit is also used as the enable signal for the SECDED encoder; therefore, a value of one enables the module. The SECDED encoder computes the check bits and writes back the new word and the check bits. The SECDED encoder takes an additional cycle to compute the check bits and write it back to the cache; hence, the access takes two cycles to complete. The output from the hit logic is latched and thus it can be used in the second cycle.

Fig. 1. The microarchitecture diagram of the proposed NUCA-L1 architecture with support for cache line disabling. The additional components required for the architecture are shaded grey. The tag array holds the additional disable bits, whereas the ECC check bits (per cache line) are stored in the data array. The SECDED encoder and decoder are gated with the disable bit (DB). The cache hit logic can operate in one- or two-cycle mode, depending on the disable bits for each cache line. Additional storage elements are introduced to provide the information, such as the hit result for the second cycle. The resilient buffer is a fully associative structure that holds the cache lines that are mapped to permanently disabled sets in the cache.

If the access is a miss and the disable bits identify the cache line as containing single-bit fault, the request is sent to the lower-level cache. When the lower-level cache returns the cache line, the cache allocation logic finds a suitable way for the cache line. In this step, the allocation/replacement logic takes into account the LRU bits and also the fact that all cache lines may not be available. The allocation/replacement logic gives priority to fault-free cache lines over cache lines with single-bit fault when both the cache lines are invalid, because fault-free cache lines incur no additional latency, whereas cache lines with single-bit faults incur additional latency. However, if the choice is between two valid cache lines, LRU replacement scheme is used. The cache line is written to the cache as described previously.

The multiplexing logic used to route the data to the SECDED encoder or the cache is gated with the disable bits. If the disable bits identify the cache line as fault-free, the multiplexing logic routes the word directly to the cache. If the disable bits identify the cache line as single-bit fault, the multiplexing logic routes the word and the read cache line to the SECDED encoder.

3.2.4. Read Request to a Cache Line with Single-Bit Fault. If the read access is a hit and the disable bits identify the cache line as containing single-bit fault, the whole cache line is read out of the cache. The disable bit value of one ensures that the cache line is routed to the SECDED decoder and that the SECDED decoder is enabled. The SECDED decoder locates and corrects the faulty bit and then forwards the requested word to the core.

The SECDED decoder takes an additional cycle to correct the faulty bit; hence, the access takes two cycles to complete.

If the access is a miss and the disable bits identify the cache line as containing single-bit fault, the request is sent to the lower-level cache. When the lower-level cache returns the cache line, the cache allocation logic finds a suitable way for the cache line. The allocation logic works the same way as described in the previous section. The requested word is also returned to the core through the normal path, as the word is brought in from the lower-level cache and hence is fault-free.

The multiplexing logic used to route the data to the SECDED decoder or the core is gated with the disable bits. If the disable bits identify the cache line as fault-free, the multiplexing logic routes the word directly to the core. If the disable bits identify the cache line as single-bit fault, the multiplexing logic routes the code word to the SECDED decoder.

3.3. Architecture Operation: Permanently Disabled Cache Lines

The cache lines that cannot be corrected by the employed correction scheme are classified as permanently disabled cache lines. The following discussion is based on SECDED correction scheme; hence, cache lines with multibit faults are considered to be permanently disabled. The same description can be generalized to stronger correction schemes as well. In the following subsections, we describe how a read and a write request is completed, when one or more ways are permanently disabled, in our proposed NUCA-L1 architecture (cf. Figure 1).

3.3.1. Set Has at Least One Way Available. If a hit is evaluated on an access to a cache line, the request is completed through the process explained in Section 3.2. On a miss, the data is retrieved from the lower-level cache and inserted in the cache. During the insertion process, the replacement/allocation logic is invoked. The replacement/allocation logic is modified to incorporate the fact that permanently disabled cache lines can be present in a set. In such sets, the cache line is allocated to one of the available ways.

3.3.2. Set Has All Ways Permanently Disabled. When all ways in a set are permanently disabled—that is, all the cache lines have multibit faults—that set has a very low probability of occurrence. Our analysis of the bit-cell fault distribution shows that on average there is one such set per cache at the highest fault rate evaluated. At the lowest fault rate evaluated, there are no such sets. We found in our experiments that even a single highly used set can impact performance significantly. Therefore, to deal with this case, we propose to implement a fully associative structure—the resilient buffer. Each entry in this buffer stores the tag, index, and the associated bits, as well as the cache line data.

If all ways in a set are identified as permanently disabled, the resilient buffer is used to store the cache line. This resilient buffer is accessed in parallel to the NUCA-L1 cache. Accessing this buffer is exactly the same as accessing a normal fault-free cache line. If the buffer is completely filled, any new cache line allocation evicts one of the currently valid lines. LRU replacement policy is used for the resilient buffer. The area overhead of this structure is low and is accounted for while calculating overheads in Section 3.5.

3.4. NUCA-L1 Architectural Extensibility

The proposed NUCA-L1 architecture can be extended in two dimensions: the correction strength and the disabling granularity. In this section, we present three variations of the proposed NUCA-L1 architecture along these two dimensions.

3.4.1. NUCA-L1_{DEC}. We explore a variation of our architecture with stronger ECC correction capability. This variation, called NUCA-L1_{DEC}, has the capability to correct up

to two-bit faults. All cache lines that are fault-free incur no additional latency. Cache lines with one-bit faults are corrected using the SECDED decoder logic and thus need an extra cycle to complete. Cache lines with two-bit faults are corrected using the DECTED decoder logic and thus need two extra cycles to complete. All cache lines with three-bit or more faults are permanently disabled and are not allocated. The stronger fault correction capability comes at a cost of higher hardware complexity. The area overhead for this variation is almost prohibitive for an L1 cache. However, this results in performance improvement over the SECDED version at higher fault rates.

It is possible to use even stronger ECC to make the system work at extreme fault rates. However, the potential benefits may not amortize the additional complexity of ECC. For example, the opportunity to correct more cache lines decline as there are a small number of cache lines with more than two-bit faults. In addition to that, the latency overhead of stronger ECC quickly approaches that of the L2 cache hit latency.

3.4.2. NUCA-L1_{CL-SEC_WLD} (Cache Line SECDED with Word-Level Disabling). We modify the NUCA-L1 architecture by introducing word-level disabling scheme for cache lines with multi-bit faults [Abella et al. 2009]. Fault-free cache lines are accessed without any latency overhead, whereas cache lines with single-bit faults are accessed with additional one-cycle latency for fault correction. Cache lines with multibit faults are accessed using word-level disabling. In this case, all accesses to fault-free words go through normally, whereas accesses to faulty words force a miss and the cache line is evicted. This enables higher available capacity, since words from otherwise permanently disabled cache line can now be utilized. The higher available capacity is beneficial to the overall performance, especially at high bit-cell fault rates (cf. Section 5.1). However, this comes at a cost of higher storage overhead, as word-level disable bits are now required in each tag entry. Moreover, it also complicates the hit path and write-back of evicted data to L2.

The modifications on top of the NUCA-L1 architecture are as follows. To treat accesses to words with multibit faults as misses, the hit logic is modified. The cache controller evicts the cache line on such an access and writes back only the valid (fault-free) words to the lower-level L2 cache. The L2 cache controller is modified to write back only valid words instead of the evicted cache line. The allocation/replacement logic is the same as a fault-free baseline, with the capability of updating the LRU bits on forced evictions. This is needed to ensure that the evicted cache line is brought in and inserted in a different cache way [Abella et al. 2009].

3.4.3. NUCA-L1_{WL-SEC_WLD} (Word-Level SECDED with Word-Level Disabling). We consider SECDED at word level, accompanied by word-level disabling for words that cannot be corrected. Fault-free words are accessed without any latency overhead, and words with single-bit faults are accessed with additional one-cycle latency for fault correction. Accesses to words with multibit faults are treated as misses, and the associated cache lines are evicted. As the correction scheme is applied at the word level, this variation of the architecture can correct most of the bit-cell faults, resulting in high available cache capacity (even at high bit-cell fault rates). However, the word-level SECDED comes with a prohibitively high storage overhead (64 bits per cache line for word-level SECDED).

This variation also requires the changes discussed in Section 3.4.2. Moreover, further changes are made to implement the word-level SECDED. Word-level SECDED implementation either incurs high area overhead or high latency overhead, because when a cache line is brought in from a low-level cache and inserted in L1 cache, the ECC check bits are to be computed for each word. In a low area overhead approach, this computation is serialized with seven additional cycles. In a low-latency approach, eight parallel SECDED encoders are needed to complete the process in one cycle. The first approach may cause excessive latency overhead if a set with multiple disabled words

is highly utilized, as the accesses to these words will result in eviction of the cache line. We model the low-latency approach, incurring high logic overhead, to achieve the best performance.

3.5. Overhead Analysis

In this section, we calculate the area and latency overheads of our proposed NUCA-L1 architecture and its variations.

3.5.1. NUCA-L1.

Storage. Our proposed architecture requires 2 bits per cache line to classify the cache line as fault-free, 1-bit fault, and 2-bit fault. SECDED correction requires additional 11 bits for each cache line to store the ECC check bits. The following calculations are for one core but are applicable to the entire system since all cores are identical. The storage overhead shown is for L1-D cache; L1-I cache has similar storage overheads. Assuming 32KB L1 size, the L1-D cache needs $2 \times \frac{32\text{KB}}{64\text{B}} = 128\text{B}$ for storing the disable bits. The storage overhead for storing ECC bits is $11 \times \frac{32\text{KB}}{64\text{B}} = 704\text{B}$. The single entry resilient buffer results in an overhead of 72B. Therefore, the NUCA-L1 architecture uses 904B more storage per L1 cache.

Logic. We describe the logic overhead associated with SECDED encoder and decoder. The logic structure of the encoder and the first step of the decoder is the same (cf. Section 3.6.1). Since we are either encoding or decoding at the L1 cache at any given time, the XOR trees of the encoder are reused for the first step of the decoder. This requires 2k XOR gates and a latency of 8 XORs [Alameldeen et al. 2011]. Following Alameldeen et al. [2011], steps 2 and 3 require 512 XOR and 9.2k AND gates, respectively. The latency of step 2 is 1 XOR gate and that of step 3 is 5 AND gates. Based on these latencies, our SECDED encoder and decoder implementation takes an additional clock cycle each. Considering that each AND gate equals two SRAM bit-cells and each XOR gate equals four SRAM bit-cells, this translates into 3.55KB overhead per SECDED controller.

Summary. The storage and logic overheads for SECDED per L1 cache requires 4.43KB. Considering an inclusive L1-L2 cache hierarchy, a shared SECDED controller per core is sufficient. However, each L1 cache needs the 904B overhead for storage. Using 32KB L1-I, 32KB L1-D, and 256KB L2 cache, the SECDED area overhead comes out to $\sim 1.64\%$ per core. If an independent SECDED controller is assumed for each L1 cache, the area overhead would be $\sim 2.7\%$ per core.

3.5.2. NUCA-L1_{CL-SEC_WLD}. The overhead analysis from Section 3.5.1 holds for this architecture variation as well. This implementation adds an additional one bit per word to classify it as either disabled or not (a total of eight bits per cache line). This comes out to 512B per L1 cache, an increase of $\sim 57\%$ in storage overhead on top of the NUCA-L1 architecture. The logic overhead also increases over NUCA-L1 because word-level disabling increases the hardware complexity of the L1 cache and requires modifications to the lower-level cache controller as well.

3.5.3. NUCA-L1_{WL-SEC_WLD}. NUCA-L1_{WL-SEC_WLD} requires 8 bits per word to store the ECC check bits and 1 bit per word to store the word-disable status. This results in a storage overhead of 72 bits per cache line, which is almost $5\times$ storage overhead on top of NUCA-L1. Moreover, this architecture variation also suffers from higher hardware complexity due to required modifications to the lower-level cache controller.

3.6. Discussion

In this section we discuss how an ECC based scheme is implemented and what goes into the encoder and decoder module. After that, we discuss the implications of L1 cache and core pipelining, and soft-error mitigation mechanism for the proposed NUCA-L1 architecture.

3.6.1. Error-Correcting Codes. Binary BCH codes, a class of linear cyclic block codes, are the most popular choice for correcting random bit errors in memories [Rao and Fujiwara 1989]. A binary BCH code is defined over a finite Galois field $GF(2^m)$. BCH code words are produced using a generating matrix based on a set of generator polynomials. Checking a code word for errors involves using a parity matrix to obtain the syndromes [Rao and Fujiwara 1989; Imai and Kamiyanagi 1977]. The generated syndrome is then checked to determine whether there are any errors in the code word.

Hamming code is a special type of BCH code that can correct one random bit error [Chen and Hsiao 1984]. Prior work has proposed parallel designs for simple codes, such as SECDED and DECTED [Matsushima et al. 1996; Strukov 2006], which are faster than iterative designs [Strukov 2006]. These parallel implementations are significantly faster but incur a larger area overhead. We now explain the working of the two components of the ECC logic.

Encoder. The encoder takes a cache line as input and computes the ECC check bits for it. In a completely bit-parallel implementation, the encoder is made up of XOR trees, with each tree computing a single check bit [Li et al. 2011]. The data bits are concatenated with the check bits to obtain the final code word. The storage and logic overheads of the SECDED encoder are discussed in Section 3.5.

Decoder. The decoder detects and corrects all errors in a stored code word. The error-correcting logic can pinpoint all bit errors and then correct them. The decoder operation can be divided into three steps [Rao and Fujiwara 1989]. In the first step, the syndrome for the stored code word is calculated. The hardware for this stage is very similar to the encoder—that is, each syndrome bit is obtained by an XOR tree. If the calculated syndrome is equal to zero, it means that the cache line is error-free. A nonzero syndrome indicates the occurrence of one or more bit errors. In the second step, if the syndrome is nonzero, the error locator polynomial is determined from the syndrome calculated in the previous step [Berlekamp 1968; Massey 1965]. In the final step, the error locator polynomial is solved and the roots are determined. Correction is done by flipping the faulty bits using XOR gates. To keep the SECDED decoder latency as low as possible, a parallel SECDED decoder [Strukov 2006; Reed and Shih 1991] is implemented. As SECDED is the simplest BCH code, the hardware complexity for a parallel version is low and does not result in a steep increase in area.

3.6.2. Pipelining L1 Cache. To keep the hardware complexity of the L1 cache low, a single read/write shared port is implemented. A multiport cache increases its complexity dramatically. In the single-port cache implementation, all accesses to the L1 cache are serialized. An access to a cache line with a single-bit error takes two cycles to complete. If a pipelined cache is assumed, the two-cycle latency can be hidden. We argue that this is not possible on each cache access. The write accesses to a faulty cache line requires a read-modify-write operation over two cycles. As the cache has only a single read/write shared port, a second access cannot enter the cache. On a read access, however, the cache line is read out and sent to the SECDED decoder for correction. Hence, a second access can commence during the second cycle.

3.6.3. Compute Core Pipeline. The proposed NUCA-L1 architecture is evaluated using single issue, in-order cores. In such a core, the pipeline stalls if it is waiting on a

Table I. Architectural Parameters for Evaluation

Architectural Parameter	Value	Memory Subsystem	
Number of cores	64 @ 1GHz	L1-I cache per core	32KB, 4-way assoc., 1 cycle
Compute core pipeline	In-order, single-issue	L1-D cache per core	32KB, 4-way assoc., 1 cycle
Physical address length	48 bits	L2 cache per core	256KB, 8-way assoc., 6 cycles, inclusive, R-NUCA
Electrical Two-Dimensional Mesh with XY Routing		Cache line size	64 bytes
Hop latency	2 cycles (1-router, 1-link)	Directory protocol	Invalidation-based MESI
Contention model	Only link contention (Infinite input buffers)	# memory controllers	8
Flit width	64 bits	DRAM bandwidth	5GBps per Controller
		DRAM latency	100ns

memory access to return the requested word. In normal operation, the memory access can result in either an “L1 hit,” “Local L2 hit,” “Remote L2 hit,” or an “L2 miss.” These scenarios result in different access latencies, ranging from one cycle to tens or even hundreds of cycles. To support the proposed NUCA-L1 cache, the modifications needed to baseline compute the pipeline are limited to the instruction stall logic. This adds additional latency to the compute pipeline; however, NUCA-L1 minimizes this overhead by not incurring additional latency for fault-free cache lines.

Out-of-order processors are more tolerant to unexpected delays as compared to in-order processors because an out-of-order processor can continue execution on independent instructions while the outstanding loads are being serviced. In contrast, an in-order processor stalls and waits for the load to complete before it can proceed. In this respect, our implementation provides an insight into the scenario inclining toward the worst case.

3.6.4. Soft-Error Mitigation. Our proposed architecture efficiently deals with persistent errors caused by PVT variations at near-threshold voltages. However, random soft errors caused by alpha-particle strikes can still be a problem. If these errors are not dealt with, the reliable operation of the system cannot be ensured. There are several mechanisms that one can use for soft-error protection. However, since the focus of this article is on more common process variation-induced hard errors at NTV, we discuss soft-error support briefly. A detailed treatment of soft-error mitigation mechanisms will be explored as future work.

The proposed NUCA-L1 architecture’s ECC scheme can be utilized to operate the cache in “soft-error protection mode.” However, this results in a constant latency overhead to access the L1 cache, degrading performance. As an alternative, a word-level parity-based scheme can be deployed in parallel to the proposed NUCA-L1 architecture to detect soft errors.

If a soft error is detected, the cache line is handled in two different ways, based on the state of the cache line. If the cache line is clean, the copy in L1 cache is discarded and a miss is forced to bring in the cache line from the lower-level cache. If the cache line is dirty, the copy in L1 cache cannot be discarded. In this scenario, a rollback mechanism is invoked through the operating system to revert to a previously known good state [Smolens et al. 2006].

4. EVALUATION METHODOLOGY

We evaluate a 64-core shared memory multicore. The default architectural parameters used for evaluation are shown in Table I. The baseline system is a tiled multicore

Table II. Projected Transistor Parameters for 11nm Tri-Gate

Parameter	Value
Near threshold voltage (V_{NTV})	0.45V
Gate length	14nm
Contacted gate pitch	44nm
Gate cap/width	2.420fF/ μm
Drain cap/width	1.150fF/ μm
Effective on current /width (N/P)	739/668 $\mu\text{A}/\mu\text{m}$
Off current/width	1nA/ μm

with an electrical two-dimensional mesh interconnection network. Each core consists of a compute pipeline, private L1 instruction and data caches, a physically distributed shared LLC cache with integrated directory, an SECCED encoder and decoder, and a network router. The coherence directory is integrated with the LLC slices by extending the tag arrays (in-cache directory organization [Bell et al. 2008]) and tracks the sharing status of the cache lines in the per-core private L1 caches. The private L1 caches are kept coherent using the ACKwise limited directory-based coherence protocol [Kurian et al. 2010]. Some cores have a connection to a memory controller as well. We use the Reactive-NUCA's data placement, replication, and migration mechanisms to manage the LLC [Hardavellas et al. 2009]. Private data is placed at the LLC slice of the requesting core, shared data is address interleaved across all LLC slices, and instructions are replicated at a single LLC slice for every cluster of 4 cores using a rotational interleaving mechanism.

4.1. Performance Models

All experiments are performed using the core, cache hierarchy, coherence protocol, memory system, and on-chip interconnection network models implemented within the Graphite multicore simulator [Miller et al. 2010a]. The Graphite simulator requires the memory system (including the cache hierarchy) to be functionally correct to complete simulation.

4.2. Energy Models

For energy evaluations of on-chip electrical network routers and links, we use the DSENT tool [Sun et al. 2012]. Energy estimates for the L1-I, L1-D, L2 (with integrated directory) caches are obtained using the McPAT tool [Li et al. 2009]. The ECC encoder/decoder energy numbers are estimated from Li et al. [2011].

The energy evaluation is performed at the 11nm technology node to account for future technology trends. We derive models for a tri-gate 11nm electrical technology node using the virtual-source transport models of Khakifirooz et al. [2009] and the parasitic capacitance model of Wei et al. [2011]. These models are used to obtain electrical technology parameters (Table II) used by both McPAT and DSENT. The static energy (subthreshold and gate leakage) is projected to be the dominant component of the overall energy at NTV [Kaul et al. 2012]. Therefore, in addition to dynamic energy, we also model static energy for the NTV evaluation.

The overall tool flow is as follows. Graphite runs a benchmark for the chosen cache configuration, producing event counters and performance results. The specified cache and network configurations are also fed into McPAT and DSENT to obtain dynamic per-event energies as well as static energy for each component. Event counters and completion time output from Graphite are then combined with per-event energies to obtain the overall energy usage of the benchmark.

4.3. Simulated Private L1 Cache Configurations

We evaluate the following fault-tolerant mechanisms for the private L1 caches.

4.3.1. Baselines.

- (1) *Ideal fault-free baseline* implements the L1 cache with one-cycle hit latency and 100% available capacity at all voltages.
- (2) *Archipelago* [Ansari et al. 2011] is an architectural proposal to operate caches at NTV. The available cache capacity depends on the bit fault rate and ranges from 92% to 99%. The L1 cache incurs one extra hit cycle (total of two cycles) to access the cache lines (cf. Section 2.3). Because Archipelago accesses two cache banks for each access, with each bank containing multiple ways, the dynamic energy spent for an L1 access is twice that of the baseline system. We ignore the energy consumption of the fault map and the multiplexing layer.
- (3) *Word-dis* [Wilkerson et al. 2008] patches up half of the cache capacity (50%) at the expense of one extra cycle latency (total of two cycles). It should be noted that this scheme cannot operate at high fault rates. We present the result just as a comparison point.
- (4) *Bit-fix* [Wilkerson et al. 2008] is able to recover 75% of cache capacity while incurring three extra cycles (total of four cycles). This scheme also cannot operate at high fault rates.
- (5) *Cache-line-disable* (CLD) delivers a one-cycle L1 cache hit latency. However, due to cache line-level disabling, the available cache capacity depends on the single- and multibit fault rate (or the operating voltage).
- (6) *SECDED with cache-line-disable* (SEC_{CL}) corrects cache lines with single-bit faults and incurs a two-cycle L1 cache hit latency. The available cache capacity depends on the bit fault rate.
- (7) *DECTED with cache-line-disable* (DEC_{CL}) can correct up to two faults at the expense of an additional one cycle (total of two cycles) for encoder and two cycles for decoder (total of three cycles).
- (8) *Word-level disabling* [Abella et al. 2009] (WLD) works at a fine granularity of word level. It accesses fault-free words with one-cycle hit latency. It treats accesses to faulty words as misses and evicts those cache lines to reload the fault-free word from the lower-level cache.
- (9) *SECDED with word-level disabling* [Abella et al. 2009] (SEC_{WL}) implements word-level SECDED on top of word-level disabling system. It incurs constant one-cycle latency overhead on hits to correctable words. Access to a word with multibit fault is treated as a miss and is evicted.

4.3.2. Proposed Architecture and Variations.

- (1) **NUCA-L1** is our proposed architecture. It can operate without any latency overhead on fault-free cache lines. It incurs a single-cycle additional latency for cache lines with single-bit faults. Cache lines with multibit faults are permanently disabled.
- (2) **NUCA-L1_{CL-SEC_WLD}** is a variation of the proposed architecture with word-level disabling. It operates without any latency overhead on fault-free cache lines and incurs a single-cycle overhead on accesses to cache lines with single-bit faults. Word-level disabling is used for cache lines with multibit faults. In these cache lines, access to a word with multibit fault is treated as a miss and is evicted.
- (3) **NUCA-L1_{WL-SEC_WLD}** is a variation of the proposed architecture with word-level SECDED for correction built on top of word-level disabling. It operates without any latency overhead on fault-free words and incurs a single-cycle overhead on

accesses to words with single-bit faults. Access to a word with multibit fault is treated as a miss and is evicted.

- (4) **NUCA-L1_{DEC}** is a variation of our proposed architecture. It operates without any latency overhead on fault-free cache lines. It incurs a single-cycle additional latency for cache lines with single-bit faults. Cache lines with two-bit faults incur additional latency of one cycle for encoder and two cycles for decoder, and cache lines with more than two bit faults are permanently disabled.

4.4. NTV Model

MBIST is a popular mechanism used to detect memory faults at runtime. Most state-of-the-art processors incorporate MBIST technology to detect SRAM bit-cell faults. We deploy MBIST to test the integrity of on-chip caches and identify faulty cache lines. To track whether a cache line has zero-, single- or multibit faults, we add two bits per cache line. MBIST is run during the boot-up process of the system at various operating voltages. It identifies all faulty bits at the initialized voltage and constructs a bit mask of the disable bits for each cache line accordingly. These disable bit-mask vectors are stored in main memory and are only accessible to the operating system software.

Once the system is up and running, the operating system loads the appropriate disable bit-mask vector in the L1 caches and the processor executes user applications. Any significant change in voltage can result in a different disable bit-mask vector, necessitating the operating system to context switch. In this work, we do not allow rapid dynamic adjustments to the operating voltage. One way to make a significant change in voltage is to wait for the processor utilization to change and remain steady for a certain time duration. When that happens, the operating system is invoked to populate the new disable bit-mask vector and resume normal operation at the new voltage. Another approach could be to populate the disable bit-mask vector of the lowest voltage in a range of voltages and then let the system dynamically adjust voltage within that range.

4.5. Bit Fault Masks for L1 Caches at NTV

Near-threshold voltage depends on the process technology and can vary within and across generations. At a given NTV operating point, each bit-cell can be modeled as operational or not. Bit-cell fault probabilities have been shown to correlate with NTV [Qureshi and Chishti 2013; Alameldeen et al. 2011]. Furthermore, these probabilities exhibit normal distribution and random occurrence [Kulkarni et al. 2007].

We consider a range of possible NTV operating points that are captured as a separate bit fault rate for each L1 cache in a multicore. These are *Low* (0.05% probability of a bit-cell fault), *Medium* (0.1% probability of a bit-cell fault), *High* (0.2% probability of a bit-cell fault), and *Very High* (0.3% probability of a bit-cell fault).

The disable bit mask represents zero-, single-, and multibit faults per cache line/word and is loaded in the tag array of the L1 cache when the processor is initialized to run at NTV. Figure 2(a) shows the average L1 cache capacity that is available with zero-bit, single-bit, two-bit, and more than two bit faults per cache line. At *low fault rate*, each L1 cache has on average 76% of the cache lines with no faults, ~21% with single-bit faults, ~2% with two-bit faults, and only ~0.2% with more than two bit faults. On the other hand, at *very high fault rate*, only 29.2% of the cache lines have no faults, ~39% have single-bit faults, 18.1% have two-bit faults, and ~13.6% of the cache lines have more than two bit faults. Similarly, Figure 2(b) shows the average L1 cache capacity that is available with zero-bit, single-bit, two-bit, and more than two bit faults per word. At the fine granularity of word level, each L1 cache has on average ~96% of the words with no faults, ~3.9% with single-bit faults, and ~0.1% with two-bit faults at *low*

Fig. 2. Based on the probability of a bit-cell fault at the NTV condition, each cache line/word in the private L1 cache is marked with zero, one, two, or greater than two bit-cell faults. (a) For low to high fault rates, the number of cache lines with single- and multibit faults increases from 24% to 71% of the cache capacity. (b) Similarly, the number of words with single- and multibit faults increases from 4% to 15% of the cache capacity.

fault rate. This changes to $\sim 85.2\%$ of the words with no faults, $\sim 13.7\%$ with single-bit faults, and $\sim 1.1\%$ with two-bit faults at *very high fault rate*.

4.6. Benchmarks and Evaluation Metrics

We simulate 12 SPLASH-2 [Woo et al. 1995] benchmarks (RADIX, FFT, LU_CONTIGUOUS, CHOLESKY, BARNES, FMM, OCEAN_CONTIGUOUS, WATER-SPATIAL, WATER-NSQUARED, RAYTRACE, VOL-REND, and RADIOSITY), 7 PARSEC [Bienia et al. 2008] benchmarks (BLACKSCHOLES, SWAP-TIONS, FLUIDANIMATE, CANNEAL, STREAMCLUSTER, DEDUP, and FERRET), 2 Parallel-MI-Bench [Iqbal et al. 2010] benchmarks (DIJKSTRA, and PATRICIA), and 2 graph benchmarks (CONNECTED-COMPONENTS, STATIC-COMMUNITY) [UHPC 2010]. The graph benchmarks model social networking-based applications.

Each application is run to completion using the medium or large input sets. For each simulation run, we measure the *Completion Time*—that is, the time in parallel region of the benchmark. The access latency is broken down into six components:

- (1) *Compute latency* is the processing delay in compute pipeline including the private L1 hit latency.
- (2) *L1 to L2 cache latency* is the time spent accessing the shared L2 cache including the round-trip time on the network. At the L2 cache, a cache line access can incur additional latency due to coherence overhead.
- (3) *L2 cache waiting time* is the queuing delay incurred because requests to the same cache line must be serialized to ensure memory consistency.
- (4) *L2 cache to sharers latency* is the round-trip time needed to invalidate private sharers and receive their acknowledgments. This also includes time spent requesting and receiving synchronous write-backs.
- (5) *L2 cache to off-chip memory latency* is the time spent accessing memory including the time spent communicating with the memory controller and the queuing delay incurred due to finite off-chip bandwidth.
- (6) *Synchronization latency* is the time spent waiting due to application synchronization operations such as acquiring locks or barriers.

We also measure the energy consumption of the memory system, which includes the L1-I cache, L1-D cache, L2 cache, directory, network routers, network links, and ECC logic. Our goal with the NUCA-L1 architecture is to minimize *Completion Time* to deliver high performance while operating at an energy-efficient voltage.

5. RESULTS

We perform a detailed per-benchmark analysis of our proposed architecture and compare the performance and energy consumption results to the ideal fault-free baseline.

We evaluate the systems at four V_{ccmin} points, resulting in four different bit fault rates. We abstract the voltage point being used, as it is technology dependent, and present results for the four bit fault rates ranging from low (0.05%) to very high (0.3%). The results are organized as follows.

We present the results and analysis of our proposed architecture compared to Archipelago and SECDED with word-level disabling in Section 5.1. In Section 5.2, our proposed NUCA-L1 architecture performance is compared with the different patch-up and recovery schemes available. Similarly, Section 5.3 presents the comparison with the different ECC-based techniques. Section 5.4 shows the comparison between NUCA-L1, NUCA-L1_{DEC}, NUCA-L1_{CL-SEC_WLD}, and NUCA-L1_{WL-SEC_WLD}, along with the different trade-offs involved. Finally, Section 5.5 presents a study on the sensitivity of performance to the L1 cache size.

5.1. Comparison with the Best-Performing ECC and Patch-Up and Recovery Scheme

We evaluated different schemes and found SECDED with word-level disabling and Archipelago to be the best of ECC (cf. Section 5.3) and patch-up and recovery (cf. Section 5.2) mechanisms, respectively. In this section, we present the per-benchmark completion time and energy results for our proposed architecture compared to SECDED with word-level disabling and state-of-the-art Archipelago systems. Results for NUCA-L1_{CL-SEC_WLD} and NUCA-L1_{WL-SEC_WLD} are also presented for comparison.

5.1.1. Fault Rate of 0.3%. The 0.3% fault rate is the highest that we evaluate in our experiments. This corresponds to an extreme near-threshold voltage.

Per-Benchmark Completion Time. Figure 3 shows the per-benchmark completion time results for 0.3% fault rate. At this extreme fault rate, the usable capacity (cache lines with zero- and single-bit faults) for our proposed architecture and SECDED is $\sim 68.3\%$, whereas that for Archipelago is $\sim 92\%$. The lower capacity available for our proposed architecture, in comparison to Archipelago, results in a higher L1 miss rate. However, the lower average hit latency compensates for the higher miss rate. There is a trade-off in play that dictates the performance of a benchmark based on its access pattern and its active working set. Benchmarks with a large active working set tend to put high stress on the L1 caches, resulting in higher-capacity misses at lower available capacity. On the other hand, benchmarks that have a smaller active working set take advantage of the lower average L1 hit latency.

SWAPTIONS is a benchmark that shows the lower average hit latency and capacity trade-off. Our proposed architecture reduces the average L1 hit latency significantly (shown by “Compute” in Figure 3). However, the reduced available capacity impacts the L1 miss rate by 1.2%, which is enough to offset any gains. A similar trend of lower average hit latency and higher miss rate is observed in several benchmarks such as FFT, BARNES, BLACKSHOLES, FLUIDANIMATE, and DIJKSTRA.

The active working set and access pattern of WATER-SPATIAL is such that the L1 capacity miss rate is very low. Although the L1 capacity miss rate increases for our proposed NUCA-L1 architecture over Archipelago, it is only $\sim 0.5\%$ increase. This small change in L1 miss rate does not impact the performance in a big way. However, the performance improvement gained by lowering the average L1 hit latency is significant. This results in an overall improvement in completion time for WATER-SPATIAL by 5% in comparison to the baseline. Similar behavior can be noticed in WATER-NSQUARED, DIJKSTRA, and STATIC-COMMUNITY.

CONNECTED-COMPONENTS exhibits a high L1 miss rate. Due to this high L1 miss rate, the completion time is dominated by the time the system spends in servicing L1 misses. The NUCA-L1 architecture does improve the average L1 hit latency, but the L1 miss rate

Fig. 3. Completion time results at 0.3% fault rate for Archipelago, SECDDED with word-level disabling, NUCA-L1, NUCA-L1_{CL-SEC_WLD}, and NUCA-L1_{WL-SEC_WLD}. Results are normalized to the ideal fault-free baseline.

is rather high and dominates the overall completion time. This high miss rate is due to the access pattern of the workload, as the miss rate does not increase on the NUCA-L1 architecture.

The proposed NUCA-L1 architecture improves the average L1 hit latency in RADIOSITY. However, the capacity becomes a major bottleneck. The lower available capacity results in an increase in the L1 miss rate, which is enough to overcome any gain due to lowering of average L1 hit latency. The resulting overall performance for Archipelago is lower by 7% than NUCA-L1, as it has more available capacity to work with. Other benchmarks showing the same trend but with lower degradation in performance include OCEAN_CONTIGUOUS, RAYTRACE, BLACKSHOLES, and PATRICIA.

The available capacity for SECDDED with word-level disabling is very high (~99%), which results in a very low L1 miss rate. However, the constant latency overhead and forced evictions due to accesses to disabled words degrade the performance significantly. The lower average L1 hit latency becomes the major contributing factor in the difference in completion time for the NUCA-L1 architecture. This can be seen in RADIOSITY, SWAPTIONS, FLUIDANIMATE, DIJKSTRA, FFT, WATER-SPATIAL, and WATER-NSQUARED (see Figure 3).

NUCA-L1_{CL-SEC_WLD} and NUCA-L1_{WL-SEC_WLD} perform better than NUCA-L1 in all benchmarks because of the higher available capacity in both the configurations. However, it should be noted that these systems come with additional storage overhead, along with an increase in hardware complexity, as discussed in Section 3.4.

NUCA-L1_{WL-SEC_WLD} performs better than NUCA-L1_{CL-SEC_WLD} because it selectively applies SECDDED at a finer granularity of word level. This effectively reduces the number of

Fig. 4. Energy results at 0.3% fault rate for Archipelago, SECDDED with word-level disabling, NUCA-L1, NUCA-L1_{CL-SEC_WLD}, and NUCA-L1_{WL-SEC_WLD}. Results are normalized to the ideal fault-free baseline.

requests to the ECC encoder and decoder, as only faulty words in the cache line need to be corrected, whereas each access to such a cache line incurs additional latency in the NUCA-L1_{CL-SEC_WLD} configuration.

Our proposed NUCA-L1 architecture performs within 16% of the ideal fault-free baseline, which is on par with Archipelago. SECDDED with word-level disabling performs 20% worse than the ideal fault-free baseline. NUCA-L1_{WL-SEC_WLD} performs only 6.7% worse than the ideal fault-free baseline; however, it comes with additional overhead and complexity on top of NUCA-L1.

Per-Benchmark Energy. The per-benchmark energy results are shown in Figure 4. As static energy is the biggest component of the overall energy, it dictates the energy trends observed. The static energy tracks the completion time closely, and the systems that reduce completion time end up reducing static energy as well. As NUCA-L1 and Archipelago have similar completion time, their static energy consumption is also very close. NUCA-L1_{CL-SEC_WLD}, and NUCA-L1_{WL-SEC_WLD} reduce the completion time, hence reducing the static energy as well.

We observe that the dynamic energy for the Archipelago system is considerably higher as compared to the other systems. Although it improves on the L2 cache, network router, and network link dynamic energy and overall static energy, the impact of increase in dynamic energy in L1 caches is high. The reason for such an increase in L1 cache dynamic energy is the way that Archipelago accesses the cache. For each

access to the L1 cache, it accesses two cache banks, each with multiple ways in it.¹ This doubles the dynamic energy spent on each L1 cache access.

Archipelago decreases the L1 miss rate in most of the benchmarks, as discussed previously. This in turn helps decrease the L2 cache, network router, and network link dynamic energy by $\sim 1\%$. Moreover, the static energy consumption of Archipelago is on par with NUCA-L1. However, the dynamic energy of L1 caches increases by $\sim 4\%$, resulting in an overall increase in energy of 3% over NUCA-L1.

As the L1 miss rate for SECDED with word-level disabling is very low compared to NUCA-L1, the resulting dynamic energy consumption is also very low. SECDED with word-level disabling improves on the L1 data cache, the L2 cache, and the network dynamic energy but spends more dynamic energy on ECC. Furthermore, it spends significantly higher static energy than NUCA-L1.

NUCA-L1_{CL-SEC_WLD} and NUCA-L1_{WL-SEC_WLD} are helped by their high available capacity and result in lower overall energy consumption overhead of 12% and 11%, respectively. NUCA-L1 architecture spends 16% more energy over the ideal fault-free baseline. In comparison, Archipelago spends $>19\%$ more energy than the ideal fault-free baseline.

5.1.2. Fault Rate of 0.05%.

Per-Benchmark Completion Time. The 0.05% fault rate is the lowest fault rate that we consider in our evaluation. At this fault rate, the usable capacity is $\sim 97.5\%$ for NUCA-L1 and $\sim 99.9\%$ for SECDED with word-level disabling. Most accesses are made to fault-free cache lines, resulting in optimal average hit latency in NUCA-L1.

The miss rate of BARNES increases by 30% over the ideal fault-free baseline's miss rate for NUCA-L1, resulting in a performance degradation of 9.7%. NUCA-L1_{WL-SEC_WLD} has a similar miss rate. However, it improves greatly on L1 cache access latency, as 96% of the words are accessed without any additional latency and 3.9% word accesses incur a single-cycle latency overhead. In comparison, NUCA-L1 accesses 76% of the cache lines without any latency overhead, and 21% cache line accesses incur additional one-cycle latency. Similar behavior is observed in CHOLESKY, WATER-SPATIAL, WATER-NSQUARED, RADIOSITY, BLACKSCHOLES, SWAPTIONS, and FLUIDANIMATE.

At 0.05% bit-cell fault rate, the average hit latency of NUCA-L1 architecture is very close to the ideal fault-free baseline. NUCA-L1 performs within 5% of the ideal fault-free baseline (Figure 5). The average completion time is $>10\%$ better, relative to baseline, than Archipelago and SECDED.

Per-Benchmark Energy. The energy result also shows a similar improvement (Figure 6) as completion time. We see a similar trend in static energy consumption as 0.3% fault rate. The systems that improve completion time also end up reducing static energy.

Most of the L1 misses due to capacity are converted into L1 hits. This in turn reduces the L2 cache, network router, and network link dynamic energy and static energy. The NUCA-L1 consumes 4.5% more energy than the ideal fault-free baseline.

The energy overhead over fault-free baseline decreases from 16% to 4.5% when the bit fault rate is decreased from 0.3% to 0.05%. From these results, we can see that the dynamic energy consumption of NUCA-L1 is heavily impacted by the number of L1 misses. Hence, improving the L1 hit rate results in better performance and lower dynamic energy for NUCA-L1. This can be seen in NUCA-L1_{WL-SEC_WLD}, where the available capacity is $\sim 99.9\%$. The high available capacity results in energy consumption on par (only 0.7% more) with the ideal fault-free baseline.

¹The banks can be accessed in a more intelligent way to avoid looking up multiple banks on each access. However, we model the system as proposed in Ansari et al. [2011].

Fig. 5. Completion time results at 0.05% fault rate for Archipelago, SECDDED with word-level disabling, NUCA-L1, NUCA-L1_{CL-SEC_WLD}, and NUCA-L1_{WL-SEC_WLD}. Results are normalized to the ideal fault-free baseline.

5.1.3. Sensitivity to Different Fault Rates. From Figures 7(a) and 7(b), we observe that as we decrease the bit fault rate from 0.3% to 0.05%, NUCA-L1 adapts and exploits the increased usable capacity. Its performance degradation of 16% at the highest fault rate is reduced to <5% at the lowest fault rate. It also reduces the energy consumption overhead from 16% < 5%. As usable capacity for Archipelago is high at all bit fault rates, we do not observe significant performance improvement. The energy consumption improves by a small amount and is 19% higher with respect to fault-free baseline. SECDDED with word-level disabling slightly improves the completion time (from 20% to 16%) as the bit-cell fault rate goes down, along with decrease in energy consumption from 20% to 15%. Similar improvements in completion time and energy consumption are observed in NUCA-L1_{CL-SEC_WLD} and NUCA-L1_{WL-SEC_WLD}.

5.2. Comparison with Patch-Up and Recovery Schemes

In this section, we compare the completion time results of NUCA-L1 to systems based on architectural innovations to deal with NTV operation. The systems compared include bit-fix, word-disable, cache-line-disable, word-level disabling, and Archipelago. NUCA-L1 is able to outperform all systems at all fault rates (Figure 8(a)). We observe that cache-line-disable performs poorly at high fault rates but improves and performs on par with NUCA-L1 at a 0.05% fault rate. The reason for this behavior is the higher available capacity at the 0.05% fault rate. One can argue that NUCA-L1 should perform better than cache-line-disable, as it has more usable capacity. Although NUCA-L1 has 21% higher usable capacity than cache-line-disable, this capacity incurs an extra cycle

Fig. 6. Energy results at 0.05% fault rate for Archipelago, SECDED with word-level disabling, NUCA-L1, NUCA-L1_{CL-SEC_WLD}, and NUCA-L1_{WL-SEC_WLD}. Results are normalized to the ideal fault-free baseline.

Fig. 7. Completion time (a) and energy consumption (b) sensitivity to different fault rates for Archipelago, SECDED with word-level disabling, NUCA-L1, NUCA-L1_{CL-SEC_WLD}, and NUCA-L1_{WL-SEC_WLD}. Results are normalized to the ideal fault-free baseline.

Fig. 8. Geomean of normalized completion time of patch-up and recovery (a) and ECC-based (b) schemes.

Fig. 9. Geomean of normalized completion time (a) and associated storage overhead of NUCA-L1, NUCA-L1CL-SEC_WLD, and NUCA-L1WL-SEC_WLD (b).

for each hit. The result here clearly shows the dependence of overall performance on capacity at higher fault rates and latency at lower fault rates.

5.3. Comparison with ECC-Based Schemes

In this section, we present the completion time results for NUCA-L1 compared to SECDED with cache-line-disable, DECTED with cache-line-disable, and SECDED with word-level disabling. We observe that correction strength above SECDED is overkill and does not improve the system performance. We can see from Figure 8(b) that SECDED performs better than DECTED at all fault rates. This shows that any increase in L1 hit latency degrades the overall performance significantly. We qualitatively argue that any scheme stronger than DECTED would be suboptimal because of the excessive increase in L1 hit latency. We can also see that SECDED with word-level disabling performs better than SECDED with cache-line-disable at higher fault rates because of the higher available capacity. However, this slight increase in performance comes at a high cost ($\sim 6\times$ increase in storage overhead).

5.4. Architecture Variations for NUCA-L1

Figure 9(a) shows the normalized geometric mean of the completion time for NUCA-L1, NUCA-L1_{DEC}, NUCA-L1_{CL-SEC_WLD}, and NUCA-L1_{WL-SEC_WLD} at different fault rates. The performance of NUCA-L1_{DEC} is 3% better than NUCA-L1 with SCEDED correction capability at a 0.3% fault rate. This improvement diminishes as the fault rates goes down, and performance is on par with NUCA-L1 at a 0.05% fault rate. The reason for this behavior is that at higher fault rates, the DECTED correction capability can correct more cache lines, and hence the available capacity is higher. This higher available capacity translates to a lower L1 miss rate. However, at low fault rates, the opportunity to recover more capacity is low.

NUCA-L1_{CL-SEC_WLD} performs better than NUCA-L1 at all fault rates; however, the difference in performance decreases as the fault rate goes down and is $<1\%$ at the lowest fault rate. It performs better than NUCA-L1_{DEC} and has lower storage overhead. However, it has higher hardware complexity, as the L2 cache controller needs to be modified as well for it to work (cf. Section 3.4.2). Taking full advantage of the high available capacity, NUCA-L1_{WL-SEC_WLD} performs $>4\%$ better than NUCA-L1 at all fault rates. These variations improve the overall performance; however, they come with a high storage overhead and/or increase in hardware complexity (Figure 9(b)). If high performance is the ultimate goal and the overheads are not a problem, one can opt for NUCA-L1_{WL-SEC_WLD}. NUCA-L1_{CL-SEC_WLD} provides a balance between the overheads and the performance improvement. However, the performance improvement is not significant enough at lower fault rates. Similarly, NUCA-L1_{DEC} comes with a small overhead, but the performance is

Fig. 10. Geomean of normalized completion time at private L1 cache sizes of 32KB, 16KB, and 8KB for NUCA-L1 variants.

not worth the extra logic and storage overhead. NUCA-L1 is a practical architecture that delivers high performance while keeping the overheads low.

5.5. L1 Cache Size Sensitivity

Figure 10 shows the sensitivity of the proposed NUCA-L1 architecture to three different L1 cache sizes. We observe that at the highest fault rate, NUCA-L1 performance worsens from ~16% at the 32KB L1 cache size to ~21% at the 8KB L1 cache size. However, the other architecture variants do not degrade, because they are able to recover more cache capacity at the highest fault rate. We also observe that as fault rates drop, all cache sizes evaluated show a decreasing trend in degradation of the completion time. We note that at low fault rates, the performance dependence on cache size minimizes since the available capacity approaches the fault-free baseline.

6. CONCLUSION

In the imminent era of large-scale shared memory multicores, energy-efficient operation will be critical. To enable operation at near-threshold voltages, such processors will need to handle the persistent faults due to PVT variations. Multicore processors rely on fast private L1 caches to achieve high performance. In the presence of bit-cell faults at NTV conditions, an L1 cache can either sacrifice capacity or incur additional latency to correct the faults. With this trade-off in mind, we have proposed a novel private NUCA-L1 architecture that balances performance and energy consumption at NTV. Our proposed NUCA-L1 architecture's performance and energy degradation is as low as 4.5% and 2%, respectively, in comparison to the fault-free baseline. It performs better than state-of-the-art Archipelago and SECDED with cache-line-disable by up to 11% relative to baseline. It also consumes lower energy than Archipelago by up to 14.5%.

REFERENCES

- Jaume Abella, Javier Carretero, Pedro Chaparro, Xavier Vera, and Antonio González. 2009. Low Vccmin fault-tolerant cache with highly predictable performance. In *Proceedings of the International Symposium on Microarchitecture*.
- Jaume Abella, Eduardo Quiñones, Francisco J. Cazorla, Yanos Sazeides, and Mateo Valero. 2011. RVC: A mechanism for time-analyzable real-time processors with faulty caches. In *Proceedings of the 6th International Conference on High Performance and Embedded Architectures and Compilers (HiPEAC'11)*. ACM, New York, NY, 97–106.
- Alaa R. Alameldeen, Ilya Wagner, Zeshan Chishti, Wei Wu, Chris Wilkerson, and Shih-Lien Lu. 2011. Energy-efficient cache design using variable-strength error-correcting codes. In *Proceedings of the International Symposium on Computer Architecture*.
- David H. Albonesi. 1999. Selective cache ways: On-demand cache resource allocation. In *Proceedings of the International Symposium on Microarchitecture*.
- Amin Ansari, Shuguang Feng, Shantanu Gupta, and Scott A. Mahlke. 2011. Archipelago: A polymorphic cache design for enabling robust near-threshold operation. In *Proceedings of the International Symposium on High Performance Computer Architecture*. 539–550.

- Shane L. Bell, Bruce Edwards, John Amann, Rich Conlin, Kevin Joyce, Vince Leung, John MacKay, Mike Reif, Liewei Bao, John Brown, Matthew Mattina, Chyi-Chang Miao, Carl Ramey, David Wentzlaff, Walker Anderson, Ethan Berger, Nat Fairbanks, Durlov Khan, Froilan Montenegro, Jay Stickney, and John Zook. 2008. TILE64 - Processor: A 64-core SoC with mesh interconnect. In *Proceedings of the International Solid-State Circuits Conference*. 182–183.
- Elwyn R. Berlekamp. 1968. *Algebraic Coding Theory*. McGraw-Hill.
- Christian Bienia, Sanjeev Kumar, Jaswinder Pal Singh, and Kai Li. 2008. The PARSEC benchmark suite: Characterization and architectural implications. In *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques*.
- Peter Calingaert. 1961. Two-dimensional parity checking. *Journal of the ACM* 8, 2, 186–200.
- Anantha P. Chandrakasan, Denis C. Daly, Daniel F. Finchelstein, Joyce Kwong, Yogesh K. Ramadass, Mahmut E. Sinangil, Vivienne Sze, and Naveen Verma. 2010. Technologies for ultradynamic voltage scaling. *Proceedings of the IEEE* 98, 2, 191–214.
- Chin-long Chen and Mu-yue Hsiao. 1984. Error-correcting codes for semiconductor memory applications: A state-of-the-art review. *IBM Journal of Research and Development* 28, 2, 124–134. DOI: <http://dx.doi.org/10.1147/rd.282.0124>
- Gregory K. Chen, David Blaauw, Trevor Mudge, Dennis Sylvester, and Nam Sung Kim. 2007. Yield-driven near-threshold SRAM design. In *Proceedings of the International Conference on Computer-Aided Design*.
- Zeshan Chishti, Alaa R. Alameldeen, Chris Wilkerson, Wei Wu, and Shih-Lien Lu. 2009. Improving cache lifetime reliability at ultra-low voltages. In *Proceedings of the International Symposium on Microarchitecture*.
- Pat Conway, Nathan Kalyanasundharam, Gregg Donley, Kevin Lepak, and Bill Hughes. 2010. Cache hierarchy and memory subsystem of the AMD opteron processor. *IEEE Micro* 30, 2, 16–29. DOI: <http://dx.doi.org/10.1109/MM.2010.31>
- DARPA UHPC Program (DARPA-BAA-10-37), March 2010.
- Ronald G. Dreslinski, David Fick, Bharan Giridhar, Gyouho Kim, Sangwon Seo, Matthew Fojtik, Sudhir Satpathy, Yoonmyung Lee, Daeyeon Kim, Nurrachman Liu, Michael Wiecekowsky, Gregory Chen, Dennis Sylvester, David Blaauw, and Trevor Mudge. 2013. Centip3De: A 64-Core, 3D stacked near-threshold system. *IEEE Micro* 33, 2, 8–16.
- Manoj Franklin and Kewal K. Saluja. 1990. Built-in self-testing of random-access memories. *Computer* 23, 10, 45–56. DOI: <http://dx.doi.org/10.1109/2.58236>
- Robert G. Gallager. 1962. Low-density parity-check codes. *IRE Transactions on Information Theory* 8, 1, 21–28. DOI: <http://dx.doi.org/10.1109/TIT.1962.1057683>
- Nikos Hardavellas, Michael Ferdman, Babak Falsafi, and Anastasia Ailamaki. 2009. Reactive NUCA: Near-optimal block placement and replication in distributed caches. In *Proceedings of the International Symposium on Computer Architecture*.
- Farrukh Hijaz, Qingchuan Shi, and Omer Khan. 2012. Low-latency mechanisms for near-threshold operation of private caches in shared memory multicores. In *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture Workshops (MICROW'12)*. IEEE, Los Alamitos, CA, 68–73. DOI: <http://dx.doi.org/10.1109/MICROW.2012.10>
- Farrukh Hijaz, Qingchuan Shi, and Omer Khan. 2013. A private l1 cache architecture to exploit the latency and capacity tradeoffs in multicores operating at near-threshold voltages. In *Proceedings of the 2013 31st IEEE International Conference on Computer Design (ICCD'13)*. DOI: <http://dx.doi.org/10.1109/ICCD.2013.6657029>
- Hideki Imai and Y. Kamiyanagi. 1977. A construction method for double error correcting codes for application to main memories. *Transactions of the IECE Japan* J60-D, 861–868.
- Syed M. Z. Iqbal, Yuchen Liang, and Hakan Grahn. 2010. ParMiBench—an open-source benchmark for embedded multiprocessor systems. *IEEE Computer Architecture Letters* 9, 2, 45–48.
- Ulya R. Karpuzcu, Abhishek Sinkar, Nam Sung Kim, and Josep Torrellas. 2013. EnergySmart: Toward energy-efficient manycores for near-threshold computing. In *Proceedings of the International Symposium on High Performance Computer Architecture*.
- Himanshu Kaul, Mark Anders, Steven Hsu, Amit Agarwal, Ram Krishnamurthy, and Shekhar Borkar. 2012. Near-threshold voltage (NTV) design: Opportunities and challenges. In *Proceedings of the Design Automation Conference*.
- Ali Khakifirooz, Osama M. Nayfeh, and Dimitri Antoniadis. 2009. A simple semiempirical short-channel MOSFET current-voltage model continuous across all regions of operation and employing only physical parameters. *IEEE Transactions on Electron Devices* 56, 8, 1674–1680.

- Changkyu Kim, Doug Burger, and Stephen W. Keckler. 2002. An adaptive, non-uniform cache structure for wire-delay dominated on-chip caches. In *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS X)*. 211–222.
- Jangwoo Kim, Nikos Hardavellas, Ken Mai, Babak Falsafi, and James C. Hoe. 2007. Multi-bit error tolerant caches using two-dimensional error coding. In *Proceedings of the International Symposium on Microarchitecture*. 197–209. DOI: <http://dx.doi.org/10.1109/MICRO.2007.19>
- Jaydeep P. Kulkarni, Keejong Kim, and Kaushik Roy. 2007. A 160 mV robust Schmitt trigger based sub-threshold SRAM. *IEEE Journal of Solid-State Circuits* 42, 10, 2303–2313.
- George Kurian, Jason Miller, James Psota, Jonathan Eastep, Jifeng Liu, Jurgen Michel, Lionel Kimerling, and Anant Agarwal. 2010. ATAC: A 1000-core cache-coherent processor with on-chip optical network. In *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques*.
- Sheng Li, Jung Ho Ahn, Richard D. Strong, Jay B. Brockman, Dean M. Tullsen, and Norman P. Jouppi. 2009. McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-42)*. 469–480.
- Sheng Li, Ke Chen, Ming-Yu Hsieh, Naveen Muralimanohar, Chad D. Kersey, Jay B. Brockman, Arun F. Rodrigues, and Norman P. Jouppi. 2011. System implications of memory reliability in exascale computing. In *Proceedings of the 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*. 1–12.
- Zhiyu Liu and Volkan Kursun. 2007. High read stability and low leakage cache memory cell. In *Proceedings of the International Symposium on Circuits and Systems*.
- Bojan Maric, Jaume Abella, and Mateo Valero. 2013. APPLE: Adaptive performance-predictable low-energy caches for reliable hybrid voltage operation. In *Proceedings of the 2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC'13)*. 1–8.
- James Massey. 1965. Step-by-step decoding of the Bose-Chaudhuri-Hocquenghem codes. *IEEE Transactions on Information Theory* 11, 4, 580–585.
- Tomoko K. Matsushima, Toshiyasu Matsushima, and Shigeichi Hirasawa. 1996. Parallel encoder and decoder architecture for cyclic codes. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* E79-A, 9, 1313–1323.
- Jason E. Miller, Harshad Kasture, George Kurian, Charles Gruenwald, Nathan Beckmann, Christopher Celio, Jonathan Eastep, and Anant Agarwal. 2010a. Graphite: A distributed parallel simulator for multicores. In *Proceedings of the 2010 IEEE 16th International Symposium on High Performance Computer Architecture (HPCA'10)*. 1–12.
- Timothy N. Miller, Renji Thomas, James Dinan, Bruce Adcock, and Radu Teodorescu. 2010b. Parichute: Generalized turbocode-based error correction for near-threshold caches. In *Proceedings of the 2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture*. 351–362.
- Farshad Moradi, Dag T. Wisland, Snorre Aunet, Hamid Mahmoodi, and Tuan-Vu Cao. 2008. 65NM sub-threshold 11T-SRAM for ultra low voltage applications. In *Proceedings of the 2008 IEEE International SOC Conference*. 113–118.
- Yasuhiro Morita, Hidehiro Fujiwara, Hiroki Noguchi, Yusuke Iguchi, Koji Nii, Hiroshi Kawaguchi, and Masahiko Yoshimoto. 2007. An area-conscious low-voltage-oriented 8T-SRAM design under DVS environment. In *Proceedings of the Symposium on VLSI Circuits*.
- Moinuddin K. Qureshi and Zeshan Chishti. 2013. Operating SECDED-based caches at ultra-low voltage with FLAIR. In *Proceedings of the 2013 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'13)*. 1–11. DOI: <http://dx.doi.org/10.1109/DSN.2013.6575314>
- Thammavarapu R. N. Rao and Eiji Fujiwara. 1989. *Error-Control Coding for Computer Systems*. Prentice Hall.
- Irving S. Reed and Ming-Tang Shih. 1991. VLSI design of inverse-free Berlekamp-Massey algorithm. *IEEE Proceedings E (Computers and Digital Techniques)* 138, 5, 295–298.
- David Roberts, Nam Sung Kim, and Trevor Mudge. 2007. On-chip cache device scaling limits and effective fault repair techniques in future nanoscale technology. In *Proceedings of the 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD'07)*. 570–578.
- Stanley E. Schuster. 1978. Multiple word/bit line redundancy for semiconductor memories. *IEEE Journal of Solid-State Circuits* 13, 5, 698–703.
- Jared C. Smolens, Brian T. Gold, Babak Falsafi, and James C. Hoe. 2006. Reunion: Complexity-effective multicore redundancy. In *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-39)*. 223–234. DOI: <http://dx.doi.org/10.1109/MICRO.2006.42>

- Dmitri Strukov. 2006. The area and latency tradeoffs of binary bit-parallel BCH decoders for prospective nanoelectronic memories. In *Proceedings of the 40th Asilomar Conference on Signals, Systems and Computers (ACSSC'06)*. 1183–1187.
- Chen Sun, Chia-Hsin Owen Chen, George Kurian, Lan Wei, Jason Miller, Anant Agarwal, Li-Shiuan Peh, and Vladimir Stojanovic. 2012. DSENT—a tool connecting emerging photonics with electronics for optoelectronic networks-on-chip modeling. In *Proceedings of the International Symposium on Networks-on-Chip*.
- Lan Wei, Frederic Boeuf, Thomas Skotnicki, and Hon-Sum-S P. Wong. 2011. Parasitic capacitances: Analytical models and impact on circuit-level performance. *IEEE Transactions on Electron Devices* 58, 5, 1361–1370.
- Chris Wilkerson, Hongliang Gao, Alaa R. Alameldeen, Zeshan Chishti, Muhammad Khellah, and Shih-Lien Lu. 2008. Trading off cache capacity for reliability to enable low voltage operation. In *Proceedings of the International Symposium on Computer Architecture*.
- Steven Cameron Woo, Moriyoshi Ohara, Evan Torrie, Jaswinder Pal Singh, and Anoop Gupta. 1995. The SPLASH-2 programs: Characterization and methodological considerations. In *Proceedings of the International Conference on Computer Architecture*.
- Se-Hyun Yang, Babak Falsafi, Michael D. Powell, Kaushik Roy, and T. N. Vijaykumar. 2001. An integrated circuit/architecture approach to reducing leakage in deep-submicron high-performance i-caches. In *Proceedings of the 7th International Symposium on High-Performance Computer Architecture*. 147–157.
- Se-Hyun Yang, Babak Falsafi, Michael D. Powell, and T. N. Vijaykumar. 2002. Exploiting choice in resizable cache design to optimize deep-submicron processor energy-delay. In *Proceedings of the 8th International Symposium on High-Performance Computer Architecture*. 151–161.
- Kathy Yelick. 2009. Ten ways to waste a parallel computer. In *Keynote at International Symposium on Computer Architecture*.
- Doe Hyun Yoon and Mattan Erez. 2009. Memory mapped ECC: Low-cost error protection for last level caches. In *Proceedings of the International Symposium on Computer Architecture*.

Received January 2014; revised May 2014; accepted May 2014