CiDRA: A Cache-inspired DRAM Resilience Architecture

Young Hoon Son† Sukhan Lee† Seongil O† Sanghyuk Kwon† Nam Sung Kim‡ Jung Ho Ahn†
†Seoul National University ‡University of Wisconsin-Madison
{yhson96, infy1026, swdfish, kkwon114, gajh}@snu.ac.kr nskim3@wisc.edu

***Abstract*—Although aggressive technology scaling has allowed manufacturers to integrate Giga bits of cells into a cost-sensitive main memory DRAM device, these cells have become more defect-prone. With increased cell failure rates, conventional solutions such as populating spare DRAM rows and relying on error-correcting codes (ECCs) have shown limited success due to high area overhead, the latency penalties of data coding, and interference between ECC within a device (in-DRAM ECC) and other ECC across devices (rank-level ECC).**

In this paper, we propose CiDRA, a cache-inspired DRAM resilience architecture, which substantially reduces the area and latency overheads of correcting bit errors on random locations due to these faulty cells. We put a small SRAM cache within a DRAM device to replace accesses to the addresses including the faulty cells with ones that correspond to the cache data array. This CiDRA cache is paired with a Bloom filter to minimize the energy overhead of accessing the cache tags for every DRAM access and is also partitioned into small pieces, each being associated with the I/O pads for better area efficiency. Both the cache and DRAM banks are accessed in parallel while the banks are much slower. Consequently, the cache and filter are not in the critical path for normal DRAM accesses and incur no latency overhead. We also enhance the traditional in-DRAM ECC with error position bits and the appropriate error detecting capability while preventing interference with the traditional rank-level ECC scheme. By combining this enhanced in-DRAM ECC with the cache and Bloom filter, CiDRA becomes more area efficient because the in-DRAM ECC corrects most bit errors that are sporadic while the cache deals with the remaining relatively few pathological cases.

I. Introduction

DRAM, the de-facto standard for main memory for decades, has enjoyed steady improvement in capacity and bandwidth due to continuous advances in manufacturing technology. Smaller DRAM cells and peripheral circuitry, however, entail higher vulnerability to manufacturing variations and imperfections [13]. This not only causes severe fluctuations in DRAM retention time [25], [29], but also makes various components malfunction, leading to DRAM device failures. The defect statistics are top secrets of DRAM manufactures [31], but increasing DRAM failures are serious concerns to both the high-performance computing [43], [44] and datacenter [39] communities, where bit errors in random locations are most dominant [22]. Recently, Samsung and Intel have attempted to alleviate these bit errors by protecting each (small) dataset (e.g. 32/64/128b word) in a DRAM device with an error-correcting code (ECC) [17], and studies have reported raw single-bit error rates (SBERs) as high as 10^{-4} [31].

Single-bit errors are traditional and universal problems for all storage devices, which typically adopt ECCs and duplication to limit device failure rates. However, the main memory DRAM device has unique challenges. Its access time is orders of magnitude shorter than that of hard-disk drives (HDD) and solid-state drives (SSD). Although strong ECCs such as Reed-Solomon and low-density parity-check (LDPC) codes are widely used for HDD/SSD [14], [49], it takes a long time to encode and decode them. Therefore, such strong ECCs cannot be applied directly to latency sensitive DRAM devices [42]. Because DRAM has a higher operating voltage and lower frequency than SRAM, correcting multiple faulty bits on a small dataset is not a high priority, and DRAM has relatively a longer time to deal with errors. Compared to other emerging memory technologies [36], [38], main memory DRAM is volatile and has different fault characteristics. Therefore, resilience solutions for main memory DRAM should be different from those of other storage technologies.

DRAM manufacturers have provided spare rows and columns [18] within each bank, which are two-dimensional arrays of cells each storing a bit of data, or have protected each (small) dataset with parity bits (in-DRAM ECC [5], [10]) to cope with various types of permanent faults [18] during the fabrication and testing procedures, but these schemes suffer from the following limitations. First, single-bit errors, known to be uniform-randomly distributed [34], occupy the majority of DRAM failures [22], [44] especially when a fabrication process becomes mature. Because the size of a DRAM row or column surpasses several thousand bits, dedicating a row or column per bit error is a huge waste of resources. Second, conventional in-DRAM ECC schemes have significant area or energy overheads depending on their dataset sizes that ECC is applied to [17]. Third, it lies in the critical path of normal DRAM operations either to identify a row or column with faulty cells for substitution or to encode and decode data for ECC. This induces a trade-off between the strength of the replacement and error-correcting logic and the energy and latency of the DRAM operations.

In this paper, we propose a DRAM architecture capable of tolerating single-bit errors with significantly improved area efficiency with no or minimal modification to conventional processor-memory interfaces. We add a small partitioned cache per DRAM device to replace the accesses to the addresses including the faulty cells with ones that correspond to the cache data array. Even if the cache consists of bulky SRAM cells, dedicating a few dozen SRAM cells for tagging and supplying substitute cells is still much more area-efficient than wasting thousands of DRAM cells to fix each single-bit fault. To minimize the energy overhead of accessing the caches per DRAM access, we filter the cache accesses using a Bloom filter [9]. These DRAM-side cache and filter are completely out of the critical path on normal accesses because both the cache and banks can be accessed concurrently, and the cache and filter operate much faster than that of normal DRAM structures due to their small capacity. In addition, we enhance the area and energy efficiency of the traditional in-DRAM ECC scheme by applying parity bits to only a portion of the datasets with the help of error position bits. We identify the conflict of the

ECC schemes within and between DRAM devices and devise novel architectural solutions that resolve this conflict with no or minimal impact on the existing interfaces [2].

We show that the area overhead of this novel cache-inspired DRAM resilience architecture, CiDRA, is several times smaller than that of the conventional resilience structures over a wide range of SBERs [31] when targeting the same device failure rates due to single-bit faults. The CiDRA caches and filters provide a decent area efficiency for SBERs up to 10^{-5}. When combined with the enhanced in-DRAM ECC scheme that is intended to deal with sporadic bit failures, the CiDRA caches further improve the area efficiency with negligible impact on performance. The relative energy of CiDRA over the conventional schemes depends on the target failure rates, the bit-error rates, and the device access patterns; however, CiDRA is at least as efficient as the conventional architecture for most typical usage scenarios. For a SBER of 10^{-4} and a target device failure rate of 10^{-3}, the area and read energy overheads of the CiDRA with the enhanced in-DRAM ECC are 5.5% and 1.7%, respectively, which are 2.6 and 10.5 times lower than those of the conventional spare-row-based scheme with in-DRAM ECC assuming a DRAM row-buffer conflict rate of 27%, which is the average for SPEC CPU2006 applications on a typical chip-multiprocessor system [20].

The key contributions of this paper are the following:

- We assess two traditional main memory DRAM resilience schemes: (i) populating spare rows and (ii) providing single-bit error correction per small dataset with ECC per device (traditional in-DRAM ECC). Then we identify their limitations.
- We propose a novel DRAM resilience architecture coping with bit failures at random locations. It has small SRAM caches to replace the faulty cells, places a Bloom filter ahead of the caches to reduce the cache access energy, and accesses the caches and the filter in parallel with normal DRAM accesses and hence incurs no overhead in latency.
- We enhance the in-DRAM ECC with error position bits to specify a subset to fix within each dataset paired with parity bits that control the number of correctable bits to minimize huge overhead in area, energy, and latency. We then combine the SRAM cache and filter with this enhanced in-DRAM ECC scheme to further improve the efficiency of providing resilience when the SBERs are high.
- We quantify the impact of this cache-inspired DRAM resilience (CiDRA) architecture on the performance, area, and energy efficiency compared to the traditional resilience schemes over various SBERs and target device failure rates.

II. Properties and Limitations of Conventional DRAM Resilience Schemes

We first describe how modern main memory DRAM devices are organized and operate, and which properties are exploited by conventional DRAM resilience schemes, such as spare rows/columns and in-DRAM ECC. Then, we explain conventional resilience architectures and highlight their advantages and limitations.

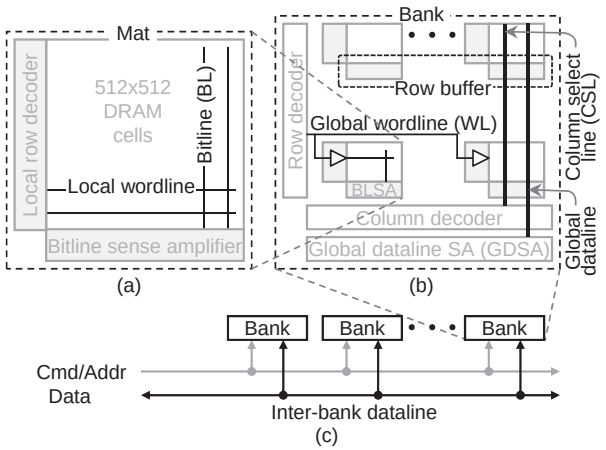


Fig. 1: A modern main memory DRAM organization. (a) A two-dimensional (2D) mat has hundreds of cells per dimension, (b) a 2D bank has dozens of mats per dimension, and (c) a device has several banks that operate mostly independently.

A. Modern DRAM Organizations and Operations

Contemporary DRAM devices use hierarchical structures to accommodate billions of cells with high area efficiency [18]. A DRAM cell consists of an access transistor and a capacitor that stores a bit of data. The cells are organized in a two-dimensional (2D) array called a mat, where a row of cells shares a local wordline (WL) and a column of cells shares a bitline (BL) for high area efficiency (Figure 1). A dedicated sense amplifier denoted by BLSA is needed per BL because the capacitance of a cell is about an order of magnitude smaller than that of the corresponding BL and other inactive transistors connected to the BL. The mats are again organized in a 2D array to constitute a bank. Within a bank, a global WL, driven by a row decoder, traverses an entire row and drives the corresponding local WLs. A global dataline traverses an entire column to access the connected BLSA. A stringent area constraint enforces the global dataline to be shared by many BLSAs within a row of mats requiring local datalines per mat, which work as (de)multiplexors controlled by column select lines. Also, a sense amplifier called GDSA is used per global dataline to reduce the access latency. Each DRAM device has several of these banks connected by inter-bank datalines and a control path, which communicate with a memory controller (MC) through I/O pads. All DRAM devices within a rank operate in tandem, where a few of the devices can store ECC information to fix bit or device failures. Ranks are connected to a MC through a memory channel. Few ranks that are mounted on a shared printed circuit board are called a module.

To access data within a DRAM device, a sequence of commands is applied while following various timing constraints that reflect the internal device structure. An activate command (`ACT`) is first applied to the device and a corresponding row in a bank is latched to a row buffer, which is a group of BLSAs in the mats containing the row. Only one row per bank can be active at any given time in conventional devices such as DDR3/4 [2], so if the row buffer holds a row other than the one to be accessed, it must first be precharged through a `PRE` command, taking *tRP*. For *tRCD* after `ACT` is sent, column commands, such as read (`RD`) and write (`WR`), are issued to access the active row. It takes *tAA* for the first data returns after

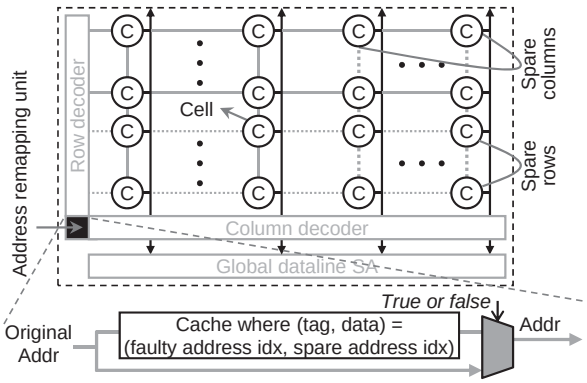


Fig. 2: A conventional resilience scheme with spare rows and columns. An address remapping unit in front of each decoder compares/supersedes faulty indices with good spares.

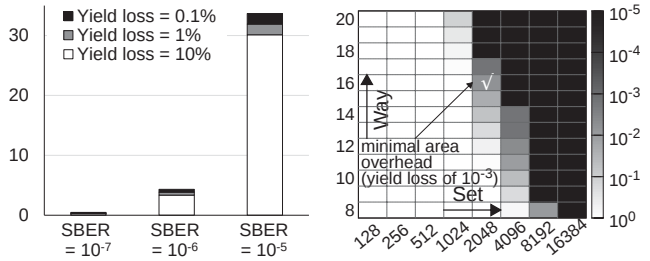
a RD is applied. The data in the passive cells are lost during the charge sharing and sensing processes, which are then restored by the BLSAs. This requires that the minimal time from ACT to PRE be *tRAS*. *tREFI* defines the refresh period of the leaky cells, which depends on the temperature [11].

The peak device bandwidth is determined by the number of data I/O pads ($\times$N) and the data transfer rate of each pad, while the effective bandwidth is limited by the timing constraints between the DRAM commands. The data transfer rate of a pad reaches multi Gbps whereas banks still operate in the low hundred MHz range. This mismatch is resolved by making the data path within a device wider than $\times$N and (de)serializing the data. The degree of this (de)serialization is called a burst length (e.g. 8 for DDR3/4) and determines the size of an access per device transferred in the minimal interval of *tCCD*, which in turn is the peak bandwidth. The timing constraints such as *tFAW* and *tRRD* reflect the limited capability of the internal power delivery networks and determine the interval between the ACTs. Consecutive reads and writes are well pipelined; however, a switch from write to read induces a latency penalty reflected by *tWTR* because a single data path is used for both reads and writes [14].

B. Improving Resilience with Spare Rows and Columns

A conventional DRAM resilience scheme remaps faulty rows and columns to populated redundant rows and columns during decoding processes [18] (Figure 2). An address remapping unit located in front of a decoder replaces a row or column with faults into a spare one. The remapping unit compares an input address with the known faulty row/column indices and if both match, it supersedes the address with a fault-free spare one. The faulty rows and columns are identified during the testing phase of DRAM manufacturing and their addresses are recorded using laser or electric fuses [45]. The remapping unit is functionally similar to a cache where its tag array stores the addresses of faulty rows or columns (i.e., faulty addresses); the data array holds spare ones, and the way is the number of addresses to be compared.

This resilience scheme with spare rows and columns is ideal when entire rows or columns malfunction; however, it is extremely area-inefficient for single-bit faults prevalent in modern DRAM devices [22] because there are thousands of cells per row or column in a bank. Between the two, it is more



(a) Area overhead (%) (b) Target yield loss (SBER 10^{-6})

Fig. 3: (a) The area overhead of the spare-row scheme as we vary the target yield loss and single-bit error rate (SBER) and (b) the yield loss as we vary the number of ways and sets of the remapping unit when SBER is 10^{-6}.

efficient to utilize a spare row instead of a spare column to fix single-bit fault because far fewer DRAM cells are associated with a row than columns; the row size, fixed by a DRAM specification, is around several thousand bits (e.g. 8Kb on a $\times$8 DDR4 device [2]) to limit the energy consumed per row activate while the column size is several times larger than the row size and it grows with the increasing bank size over generations. Even if we use spare rows to deal with single-bit faults, dedicating thousands of cells per faulty bit is still a huge waste of resources.

Another limitation of this scheme is that the address remapping unit is in the critical path of the activate, read, and write operations. It is desirable for a row with faulty bits or a faulty row to be mapped to any spare row in a bank by the remapping unit like a fully associative cache which can incur fewer misses (cover more faulty rows/columns) with fewer sets (redundant rows/columns). This in turn reduces device failure rates for a given SBER. However, having fewer sets or more ways for given cache increases both the energy and latency of an access. Therefore, it is critical for DRAM manufacturers to find the right balance between the coverage of the remapping unit and its impact on the activate time (*tRCD*) and energy.

It is reasonable to use spare rows to fix single-bit errors when SBER is low; however, as the rate increases, the overhead grows rapidly. We sweep the SBER because the component fault rates are top secrets of DRAM manufacturers [31]. Figure 3 shows the device failure rate (yield loss) and the area overhead of using spare rows as we vary the number of ways and sets of the remapping unit on a 8Gb DDR4 die using the modeling methodology described in Section IV-A. Due to limited space, we only present the area overhead, which fluctuates more than the latency and energy overheads, and choose a SBER of 10^{-6} to show the trends of the yield loss over the number of sets and ways. When SBER is lower than 10^{-7}, single-bit errors cause negligible overhead. For a given SBER, more sets and/or ways reduce the yield loss, while the rate is more sensitive to the ways. However, populating more ways to the remapping unit increases the row activate energy and latency as well as the device area. As the SBER increases, these overheads grow rapidly and for a SBER of 10^{-5} and a target yield loss of 10^{-3}, the area overhead becomes 33.7%.

C. Improving DRAM Resilience with In-DRAM ECC

Another conventional scheme for fixing bit failures is to populate in-DRAM ECCs. DRAM typically uses a block code [35] where data is divided into messages, each having a

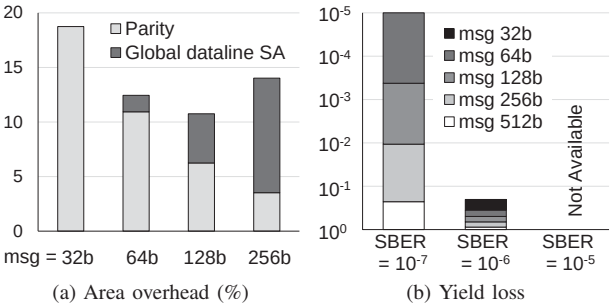


Fig. 4: The (a) area overhead and (b) yield loss of conventional in-DRAM ECC with various SBERs and message sizes.

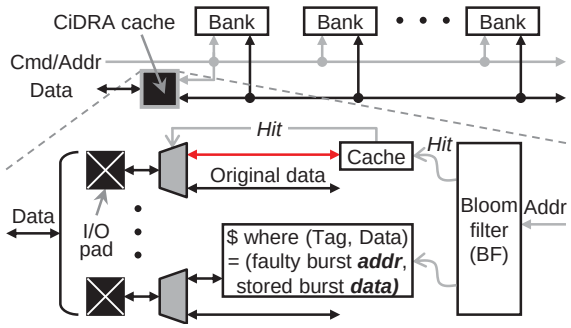


Fig. 5: The CiDRA cache is located close to I/O pads, takes the command and address information concurrently with a DRAM bank access, and replaces it if the address turns out to include faulty DRAM cells.

fixed size of k symbols and being encoded into a codeword of n symbols denoted by (n, k). The alphabet of a symbol is a bit by default, but it can be multiple bits such as the width of a device ($\times$N). The (Hamming) distance of a block code is the minimum number of symbols that should be changed for a codeword to be another. A code with a distance of 3 can correct a symbol error called single error correction (SEC); one with a distance of 4 can detect one more error called single error correction, double error detection (SECDED). The most typical rank-level ECC uses an extended Hamming code with (72, 64) for SECDEC. Stronger codes can be employed for more symbol errors, such as double error correction (DEC). Conventional in-DRAM ECCs are applied at the granularity of up to few DRAM bursts in a device. For example, the burst size of $\times$4 and $\times$8 DDR4 devices is 32 and 64 bits, respectively, requiring 6 and 7 parity bits for SEC. Therefore, in-DRAM ECCs need much fewer bits to fix a bit error whereas the spare-row scheme requires thousands of bits. Its additional strength is that it is not needed to store error locations because the block code identifies and corrects the error.

However, the conventional in-DRAM ECC has the following significant overheads. First, it has a high capacity (in turn, area) overhead and yield losses even with low SBERs. It requires a small number of bits to fix, but the parity bits are paired with *every* message even if most codewords do not have errors to correct. Still, the device fails if *any* codeword has more than one error, which is exacerbated as the device size increases. Even if we apply a stronger code, the chances that it fails to correct a codeword are still high especially for large devices. Figure 4(a) shows that the capacity overhead is as high as 19% when the message length is 32, which is the burst length of a $\times$4 device. Longer messages lead to a lower capacity overhead, but the device failure rates rapidly increase for an 8Gb device at a given SBER (Figure 4(b)). The in-DRAM ECC scheme itself is even worse than the spare row scheme, and both should be combined as suggested by [16].

The in-DRAM ECC also incurs noticeable performance and energy overheads. ECC logic is in the critical path of both the read and write operations. These took tens of nanoseconds decades ago, which was the main obstacle of this scheme from being adopted traditionally [5], [10]. Even if these are reduced to few nanoseconds or less as manufacturing technologies have improved [7], [32], their impact on *tAA* should be carefully considered. Grouping multiple DRAM bursts into a message alleviates capacity overhead, but it causes the following additional issues. First, even if a single burst is read through device I/Os per read, *all the bursts within the corresponding message and parity bits should be read through GDSAs* and decoded (Figure 4). Second, a `WR` is translated to a read-modify-write internally because the old value of the burst to be replaced should first be read and compared with the new value to update the parity bits. Therefore, the write bandwidth (*tCCD_WR* [17]) is significantly impaired. A previous study exploited sub-array level parallelism (SALP [21]), which exposes more row buffers per bank to a MC than conventional DRAM architectures to mitigate performance degradation due to this lower write bandwidth. However, even with SALP, the inferior write bandwidth still hurts system performance (Section IV-D).

III. Cache-inspired DRAM Resilience

In order to address the area, energy, and latency inefficiency problems of the conventional resilience schemes against prevalent and sporadic single-bit errors, we propose a novel cache-inspired DRAM resilience architecture called CiDRA, which consists of a DRAM-side SRAM[1] cache, a Bloom filter [9], and an enhanced in-DRAM ECC scheme.

A. A Small On-Chip Cache and Bloom Filter to Correct Bit Errors with High Area and Energy Efficiency

CiDRA cache: The first component of CiDRA is a cache to deal with faulty bits (Figure 5). The DRAM-side cache, or the CiDRA cache, is placed close to the I/Os that are assumed to be located at the center of a device without a loss of generality in this paper. It receives DRAM commands and addresses from an external MC along with the bank indices and supersedes a column access in case the access targets a burst or its portion that includes faulty bits. Because an address in the cache tag specifies a burst or its portion, the CiDRA cache stores a row address coincident with `ACT` and combines it with a column address along `RD` or `WR`. The tag array of the CiDRA cache stores the DRAM addresses associated with faulty bits, which is similar to the address remapping unit for spare rows and columns in Section II-B. The data array of the cache, in contrast, stores data for the addresses at which certain cells malfunction instead of the addresses to spare rows and columns.

The area overhead of CiDRA is substantially smaller than that of the conventional resilience scheme with spare rows

[1]Modern DRAM devices already have considerable amount of SRAM cells, such as sense amplifiers and buffers [15].

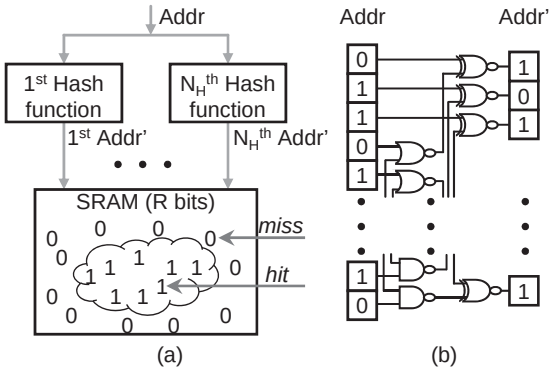


Fig. 6: (a) The Bloom filter consists of hash functions and SRAM arrays. (b) The circuit diagram of a hash function.

because the former dedicates address and data bits, at most a few dozens each, to replace a faulty DRAM cell while the latter consumes thousands of DRAM cells per faulty cell. A DRAM cell is about an order of magnitude smaller than an SRAM cell used to store the cache tag and data arrays ($6F^2$ vs. $150F^2$ where F is the minimal feature size of the technology), but the ratio of SRAM over DRAM in cell size is much smaller than the ratio of the number of DRAM cells (8Kb per $\times 8$ DDR4 device) over cache SRAM cells needed per faulty bit.

A DRAM device may have one CiDRA cache; each data I/O pad or bank of a device may have one cache, or a few pins or banks can share a cache depending on the device layout constraints. It is better to place a cache close to the I/O pads than to the banks because it takes a considerable amount of time and energy for an internal access to traverse inter-bank datalines and control path (on the order of up to several nanoseconds and few pJ/b on a device with several Giga bits). Placing one cache per device has a relatively low tag access energy and makes the CiDRA cache design simpler. However, this makes the data entry size equal to the burst size, which is relatively inefficient in area because each entry that is made of bulky SRAM cells typically corrects only one faulty DRAM cell. Making each I/O pad have a portion of the cache alleviates this area efficiency issue, but in turn brings access energy overhead because an entire cache needs to be involved per read or write.

Because the CiDRA cache and the other DRAM banks can be accessed in parallel, populating more cache ways and sets does not increase the latency of DRAM operations unlike the case of the address remapping unit. For a given cache capacity, having more ways leads to a lower chance of the cache being full, which is equal to the device failure rate. Even if populating more ways makes the cache slower, it is much smaller than a DRAM bank and still has a lower access latency than *tAA*. However, the energy to read the tags and to compare them with the incoming address to detect if it is in the cache is not negligible, especially for caches with many ways and when the cache is partitioned into I/Os and all the portions are accessed in parallel, as quantified in Section IV-B.

Bloom filter: CiDRA utilizes a Bloom filter [9] to substantially reduce this cache access energy overhead by placing it between the I/O pads and the cache (Figure 6). A Bloom filter is used to identify if a certain number (an address to access for CiDRA) belongs to a set of known numbers (addresses including faulty DRAM cells for CiDRA). It utilizes N_H independent hash functions that translate the input numbers to other numbers within a finite range ($\mathtt{R}$). The filter also has SRAM storage that consists of $\mathtt{R}$ bits, each being cleared (set to 0) in the beginning. During the training phase (the testing procedure for CiDRA), it runs the hash functions for the known numbers and sets the locations corresponding to the hash outputs of the storage to 1. During the normal operating phase, it checks the locations corresponding to the hash function outputs for a given input number. If there is any location with value 0 in the storage, the number does not belong to the known ones. Otherwise, the input number might belong to the set. Bloom filters have been applied to various computer systems [6], [40].

The Bloom filter is ideal for CiDRA because of the following reasons. First, the Bloom filter has false positive cases, i.e., some of the addresses not having faulty DRAM cells can be recognized as faulty addresses, but these can be naturally handled by the CiDRA cache located next to the filter. Second, the low bit-error rates of DRAM devices enable the false positive rates of the filter to be low with few hash functions and a reasonably sized filter storage, resulting in much lower energy consumption than accessing the CiDRA cache. A faulty address sets up to N_H bits in a Bloom filter storage, whereas several dozens of bits are needed in a CiDRA cache for the faulty address. Even if the filter storage needs a certain portion of bits staying cleared to limit the rate that an address passes the filter (which is less than $(\frac{N_H \times N_{fc}}{R})^{N_H}$, where N_{fc} is the number of faulty cells per device), a filter storage is much smaller than a CiDRA cache. The design space of the Bloom filter is explored in Section IV-C. Third, the faulty addresses are identified *once* at the testing phase. They can stay unchanged during normal DRAM accesses. If the device supports masking new permanent faults that occur after it is deployed, the CiDRA cache entry and the Bloom filter storage are updated but not cleared because the permanent faults are not reversible.

We exploit the huge advantage in the latency of the CiDRA cache with the Bloom filter over normal DRAM banks to make CiDRA more energy efficient. First, when the filter has multiple hash functions and each function can be evaluated within half of *tCCD*, the functions can be divided into multiple groups. Then, a function needs to be evaluated only if the results of all the functions in the earlier groups indicate that the address to compare might still belong to the faulty addresses, i.e., all the locations specified by the hash functions in the group have the value 1. The range ($\mathtt{R}$) of the hash functions of the filter is narrower than the address space of the DRAM device, and we implemented a hash function by randomly merging multiple input address bits by XOR gates and shuffling them (Figure 6(b)). Second, we can configure the CiDRA cache to be pseudo associative such that a cache set is divided into multiple subsets and each subset is evaluated sequentially until the cache hits, or all the subsets are tested. This effectively increases the cache associativity, influencing the device failure rate significantly as discussed in Section II-B. This increases the cache cycle time; however, as far as the cycle time does not surpass *tCCD*, it does not slow down DRAM operations. These optimizations are possible because the bank read access time (*tAA*) is typically higher than the twice of *tCCD*, and the DRAM banks and CiDRA cache augmented with the Bloom filter are accessed concurrently, not

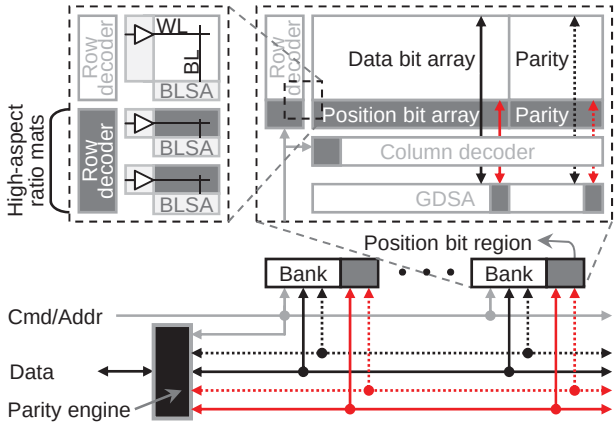


Fig. 7: A new in-DRAM ECC structure with a dedicated error position bit array. Data and parity bits are stored at normal DRAM banks while the error position bits and their parities are located at a separate array close to normal DRAM banks.

sequentially. The remapping unit for spare rows and columns has the same issue and opportunity (i.e. the unit's associativity limits the number of rows that can be replaced), but because the remapping logic is in the critical path of DRAM reads and writes, it is not possible to access multiple subsets.

B. An Enhanced In-DRAM ECC Architecture to be Combined with the CiDRA Cache and Bloom Filter

Even though the CiDRA cache and the Bloom filter achieve higher efficiency in both area and energy than that of conventional resilience schemes, dozens of SRAM cells are still needed per permanent bit failure. This becomes considerable overheads for higher SBERs. For example, there would be around a million faulty cells on an 8Gb DRAM die when the SBER is 10^{-4}. To reduce these overheads, we propose a new cost-effective in-DRAM ECC and combine it with the CiDRA cache and the Bloom filter.[2] We partition a DRAM bank into fixed-sized chunks, and let in-DRAM ECC repair one bit error in a chunk. If the chunk has two or more bit errors, we use the CiDRA cache to correct the remaining errors of these pathological chunks. Because we use the in-DRAM ECC to fix *most* of the sporadic bit errors, *not all* of them, and most chunks would have up to a single bit error each as long as the chunk size is determined properly (which is still much larger than the burst size), the area overhead of the in-DRAM ECC would be much lower than that quantified in Section II-C. However, the energy and performance overheads of the traditional in-DRAM ECC still exist, which we resolve with following novel schemes.

The traditional in-DRAM ECC schemes protect every word of a device. Therefore, all message and parity bits that constitute a codeword must be accessed per read to check if there is an error bit to fix even if only a DRAM burst out of a message (consisting of one or more bursts) is being delivered externally. Instead, we propose grouping multiple DRAM bursts into a chunk and allowing only one burst of the chunk to be fixed with the parity bits (Figure 7). Error position bits are used per chunk to associate the location to be fixed with the parity bits. Then, only one DRAM burst, the parity bits, and the position bits are needed per read, in which the position bits are compared with the address coincident with `RD` to determine whether the parity bits should be applied to the burst. For example, in a device with a row and a chunk size of 8Kb, a conventional in-DRAM ECC having a message size of 128 bits needs to read 136 bits regardless of the burst size [17], while our enhanced in-DRAM ECC needs to read 88 or 55 bits including the burst, parity, and position bits when the burst length is 64 or 32 bits, respectively. Because one burst and the corresponding parity bits constitute a codeword, it is not needed to read the old value of the burst to update the parity bits, making the write bandwidth much better than that of the conventional in-DRAM ECC and equal to the read bandwidth. The size of a chunk can be as large as the size of a bank, but it is more practical to limit it to the row size so that a burst and parity bits are ready at the BLSAs after a row is activated. The proposed in-DRAM ECC scheme is applicable to a wide range of SBERs by changing the ECC chunk size, which effectively controls the fixable number of single bit errors. Larger chunk sizes are more suitable to lower SBERs, where the CiDRA cache alone provides a sufficient area efficiency.

Error position bits: The position bits could be regarded as similar to Error-Correcting Pointers (ECP [38]) proposed for Phase-Change Memory, but have the following differences. DRAM cells are volatile and have limited retention time. This means that, if we store the position bits within the banks like normal DRAM bursts and parity bits, the information also gets lost when a bank enters a deep power-down mode, which is supported by the latest DRAM devices for power saving [2], [28]. Instead, we dedicate a separate DRAM array for storing the position bits (Figure 7). This array is accessed once during normal bank activation, and the corresponding position bits are latched to be used for subsequent `RD`s and `WR`s. As far as the access time (including an activate and a read) of the array is smaller than the activate time (*tRCD*) of normal banks, the position bits can be used safely. We locate the position bit array close to a DRAM bank edge, leverage the column decoder of the bank to latch the position bits while the bank is activated, and exploit high-aspect ratio mats [42] to reduce the activate time of the dedicated array. Because the array is orders of magnitude smaller than a DRAM bank (e.g. several bits per chunk), the read time of the array can be much lower as well. The position bits are protected by SECDED per chunk.

Interaction with rank-level ECC: Conventional processor-memory interfaces, especially the ones for datacenters and high-performance computing, have rank-level ECCs [14], in which one or two devices within a DRAM rank are dedicated for providing parity bits. The in-DRAM ECC scheme, which is applied to correct errors before the rank-level ECC is involved, must work in harmony with the rank-level ECCs. Unfortunately, this is not the case for the conventional in-DRAM ECCs (Figure 8). Typically, SECDED is used for rank-level ECCs, especially for transient errors, and its codeword consists of data being transferred by a rank at a moment, i.e., one bit per data I/O, which is 72 bits. The conventional in-DRAM ECC assumes SEC [17] for the codeword per device. Consider a case in which there is a faulty bit within a burst of a device, which is fixed by the SEC code of the in-DRAM

[2]The idea of combining the in-DRAM ECC and spare row schemes has be proposed before [16], but the coding logic took dozens of nanoseconds when the idea was proposed, restricting the idea from widely being adopted.

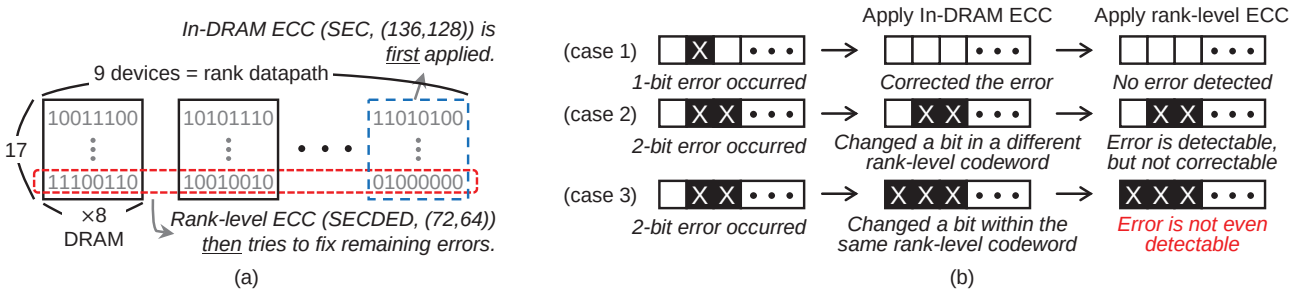


Fig. 8: Interference between in-DRAM ECC and rank-level ECC. (a) An in-DRAM ECC codeword overlaps with a rank-level ECC, but neither includes the other. (b) The in-DRAM ECC fixes a one-bit error (case 1). When it encounters a two-bit error, the in-DRAM ECC can change an innocent bit, totalling three error bits interspersed in the corresponding rank-level ECC codewords in ways that the rank-level ECC can detect but cannot correct the error (case 2) or cannot even detect it (case 3).

ECC (case 1 of Figure 8(b)). If an additional (transient or permanent) error occurs within the burst, it is expected that the rank-level ECC corrects this error, which is true for other resilience mechanisms such as spare rows and the CiDRA cache. However, the SEC code of the in-DRAM ECC has a Hamming distance of 3, recognizes the codeword with this additional error (now two-bit errors from the original valid codeword) as another valid codeword with one-bit error, and fixes it to that codeword, which is different from the original valid codeword by three bits. A rank-level ECC codeword overlaps with a in-DRAM ECC, but neither includes the other. Therefore, two of these three-bit errors may belong to one rank-level ECC codeword, where the rank-level ECC can detect the error but cannot correct it (case 2). Even worse is that the rank-level ECC cannot detect the error because all three error bits are located at one rank-level ECC codeword (case 3). The SECDED code of the rank-level ECC is not strong enough for these pathological cases.

This is definitely not acceptable, and we propose two solutions for this problem. One is to use DEC, and the other is to use a code with a distance of 4 (equivalent to SECDED) and exploit erasure or masking [19], [35], by which we detect the locations of the faulty bits by iterative reads and writes and fix them accordingly. If we use DEC but only use a single bit permanent fault within a chunk, an additional bit error is fixed by the DEC in-DRAM ECC code if it happens at the location specified by the position bits, and by the SECDED rank-level ECC code if it happens at the other bursts. Therefore, this technique is transparent to MCs.

If we use SECDED instead, we perform the following mask error correction procedure for one permanent and one soft (transient) errors in a burst during a read transaction: (i) we make the DRAM device still transfer the uncorrected burst data to the MC, store them in a buffer in the in-DRAM ECC logic, and assert the ALERT_n I/O pin, which has been added to DDR4 [2]; (ii) the MC complements the received burst data and writes them back to the DRAM device; (iii) the MC sends `PRE` after the write transaction to transfer the modified data from the row buffer to the corresponding row in the bank; (iv) the MC sends `RD` to the same burst address; the retrieved burst data are XORed with the uncorrected burst data buffered in the in-DRAM ECC logic, revealing the location of the defective bit and allows the in-DRAM ECC to repair the permanent error by flipping the bit value stored in the buffer; and (v) the rank-level ECC will repair the soft error during the subsequent read transaction. Overall, this incurs two extra memory transactions. Even though MCs are involved to deal with an additional error, it rarely occurs and hence has minimal impact on system performance. We assume that the position bits are also protected by separate SECDED.

C. Techniques to Further Improve the Performance and Area Efficiency of CiDRA

The performance and area efficiency of CiDRA can further be improved with the following techniques. For the enhanced in-DRAM ECC scheme, ECC logic is in the critical path of all DRAM reads and writes. Even if finer fabrication processes have improved the decoding and encoding latency, carrying this extra latency for all accesses causes too much a burden because only few bursts in a DRAM row would be fixed by the parity bits. If a MC knows which bursts of active DRAM rows need the extra latency for the in-DRAM ECC, it can provide two types of access latency values and apply them accordingly [42]. Unfortunately, faulty cells exist at random locations, which are different both within a device and across devices in a rank. This enforces the MC to add coding latency to all accesses. Instead, we can exploit the position bits of the dedicated arrays to *remap* the locations of faulty bursts such that they become fixed positions within a DRAM row from the perspective of a MC. This remapping incurs negligible timing overhead because remapping is needed for DRAM read and write accesses anyway to repair faulty columns for yield issues [18]. Then, the ECC logic is applied to the same locations (only a small portion of column locations) over all the DRAM devices within a rank, through which we can minimize the impact of latency increases because of the in-DRAM ECC.

The information to identify and replace faulty rows, columns, and cells should be loaded into proper places, such as the address remapping units (Section II-B), the CiDRA cache tags (Section III-A), and the error position bit arrays (Section III-B), ahead of normal DRAM operations. One way is to test DRAM chips every time a DRAM device is powered up and generate these remapping information on the fly, similar to the method suggested in [31]. The other is to use non-volatile memories (NVRAMs) [26], [45]. Traditionally, laser or electrical fuses have been used to store the faulty row and column information. However, using these bulky fuses (around $200F^2$ per cell [26], which is even larger than a SRAM cell) can nullify the area benefit of CiDRA. Instead, we advocate embedding a small non-volatile memory device, such as multi-level cell NAND flash ($2F^2$ per bit), to a memory buffer [1]

Parameter	Value	Parameter	Value	Parameter	Value
$tCCD$	3.3ns	$tRCD$	13.3ns	tAA	13.3ns
$tRAS$	32.0ns	tRP	13.3ns	$tWTR$	13.3ns
$tFAW$	21.0ns	$tREFI$	7.8μs	E_{ACT}	12.0nJ
E_{RD}	8.0nJ	E_{WR}	8.0nJ	E_{PRE}	5.0nJ

TABLE I: Default DRAM timing and energy values. E_{RD} and E_{WR} include the energy dissipation both within DRAM devices and between DRAM and MC packages.

chip per DRAM rank. The memory buffer chips are gaining popularity to alleviate the signal integrity issue of not only command/address but also data path between DRAM devices and the connected MCs. By storing the remapping information in the memory buffer, we can exploit high-speed and wide data path between the memory buffer and the DRAM devices within a rank, which greatly reduces data transfer time during a system boot up. We included the area overhead of these NVRAM arrays during the evaluation.

IV. Evaluation

We quantified the benefits of CiDRA over the conventional resilience schemes on area, energy, and latency perspectives. Then, we explored the various design spaces of CiDRA, such as the number of ways and sets of the CiDRA cache, the number of hash functions of the Bloom filter, and the chunk size of the enhanced in-DRAM ECC. We also evaluated the system-level impact of CiDRA for cases of high SBERs using single-threaded and multi-programmed applications.

A. Experimental Setup

We heavily modified CACTI-D [46] to apply modern main memory DRAM organizations and operations, which are summarized in Section II-A for power, area, and timing models. We used SPICE to find the sizes of the transistors to satisfy timing constraints, which were reflected during area and power computation. We considered the overhead due to limited metal layers of DRAM processes (three layers in this paper), and assumed a fanout-of-4 (FO4) latency of 30ps, a SRAM cell size of 150F^2, and a 22nm 8Gb $\times 8$ DDR4-2400 die, which has 4 bank groups and 4 banks per bank group (512Mb per bank) and a tCCD of 3.3ns. For each SBER, the baseline is the device with 640 spare rows and 80 spare columns per bank without SALP [21], which is derived from the overhead of row/column redundancy reported in [22]. We ran the Monte Carlo simulation for 10M times per configuration to obtain a device failure rate. We assumed that only DRAM and SRAM cells may malfunction. If a spare DRAM row or the cache entry contained faulty cells, it was not used. We used SECDED for the enhanced in-DRAM ECC, applied the burst position remapping technique, and included the latency for remapping row and column addresses in DRAM bank accesses throughout the experiments. Table I summarizes the default timing and energy values used for the evaluation.

We simulated a multi-core processor system with multiple MCs to evaluate the performance and energy efficiency impact of CiDRA. The default parameters are summarized in Table II. The following tuple was used for memory address interleaving from most to least significant bits: (row, bank, rank, channel, and column). We used McPAT [23] for the core and MC modeling, and McSimA+ [3] for the performance simulation. We used the SPEC CPU2006 [12] benchmark suite for the evaluation and identified the simulation phases using Simpoint [41], in which 100 million instructions were set as a slice and two slices with the highest weights were chosen per application. We sorted the applications by the main memory accesses per kilo-instructions and categorized 8 memory demanding applications (429.mcf, 433.milc, 437.leslie3d, 450.soplex, 459.GemsFDTD, 462.libquantum, 470.lbm, and 471.omnetpp) as spec-high and applied two slices each to constitute a multi-programmed mix (spec-mix). We populated one core and one MC for the SPEC CPU2006 single-threaded workloads to stress the main memory bandwidth.

Resource	Value
Number of (cores, MCs)	(16, 4)
Coherence (policy, type)	(MESI, Reverse dir)
Per core:	
(Frequency, issue/commit width)	(3GHz, 4 / 4)
Issue policy	Out-of-Order
L1 I/D cache size/associativity	16KB / 4
L2 cache size/associativity	1MB / 16
L1, L2 cache line size	64B
Per memory controller (MC):	
(# of channels, request queue size)	(1, 64)
(Capacity per rank, bandwidth)	(8GB, 19.2GB/s)
(# of ranks, scheduling policy)	(2, PAR-BS [30])
# of subarrays per bank for SALP [21]	64

TABLE II: Default core and memory controller (MC) parameters of the simulated multicore system.

B. Comparing the Area, Energy, and Latency of CiDRA with Conventional Resilience Schemes

We demonstrated the area and energy benefits of CiDRA by setting the target device failure rate to 0.1% and showed the relative area and energy of the conventional schemes (in-DRAM ECC, spare rows) and CiDRA schemes (cache only (\$), cache with a Bloom filter (\$/BF), and also with the enhanced in-DRAM ECC (\$/BF/ECC)) for a SBER of 10^{-7}, 10^{-6}, 10^{-5}, and 10^{-4} as shown in Figure 9. As for the conventional in-DRAM ECC, we used the proposal that enables a codeword message consisting of multiple DRAM bursts [17] because it allows for more flexibility in area overhead depending on the SBERs. We also combined the conventional spare row and in-DRAM ECC schemes for a fair comparison with the CiDRA (spare/ECC). We showed \$, \$/BF, and \$/BF/ECC side by side to highlight the impact of individual CiDRA techniques on area, energy, and latency. The read and write energy included the activate energy amortized by row-buffer conflict ratios. The column access latency (*tAA*) and the write bandwidth (*tCCD_WR*), which were most sensitive to the resilience schemes and SBERs, were chosen to compare the latency overhead. We used 4, 5, 3, and 2 hash functions for the Bloom filter for SBERs of 10^{-7}, 10^{-6}, 10^{-5}, and 10^{-4}, respectively, providing the right balance between the energy consumed by the filter and the one saved in the CiDRA cache, which is further elaborated in Section IV-C.

We made the following key observations. First, the area overheads of the conventional schemes were about an order of magnitude higher than that of the CiDRA schemes. The area overhead of CiDRA with and without the Bloom filter was

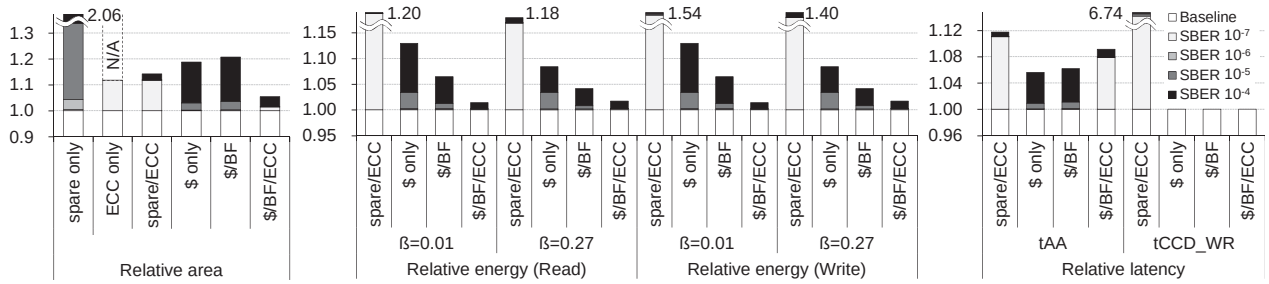


Fig. 9: The relative area, energy, and latency of the conventional resilience schemes (in-DRAM ECC, spare rows, and both being combined) and CiDRA schemes (cache only ($), cache with a Bloom filter ($/BF), and also with the enhanced in-DRAM ECC ($/BF/ECC)) for two β values, 0.01 and 0.27. β is the ratio of activate commands over the sum of reads and writes. The average β, the row-buffer conflict rate, for SPEC CPU2006 benchmarks reported in [20] is 0.27.

3.1% and 3.6%, respectively, while that of the conventional scheme with spare rows was 33.7% for a SBER of 10^{-5}. It is not even possible to achieve a device failure rate of 0.1% with the conventional in-DRAM ECC for a SBER of 10^{-5}. The impact of combining the enhanced in-DRAM ECC with the CiDRA cache and the Bloom filters (CiDRA $/BF/ECC) was most prominent for a SBER of 10^{-4}, for which CiDRA $/BF had a 20.68% area overhead while CiDRA $/BF/ECC had only a 5.46% overhead because the in-DRAM ECC takes care of most sporadic bit errors with a high area efficiency. Combining the conventional spare-row and in-DRAM ECC schemes (spare/ECC[3]) also provided a better area efficiency than separately, but was inferior to CiDRA $/BF/ECC in area. Henceforth, spare/ECC represents the conventional schemes.

Second, a small area overhead from adding the Bloom filter to CiDRA was compensated by a huge gain in energy efficiency. Because the conventional spare-row-based scheme incurs energy overhead per activate command, its impact on access energy is amortized by the ratio of activates over the sum of the reads and writes, or DRAM row-buffer conflict rate (β). In contrast, CiDRA incurs energy overhead per read or write access. As CPU-side caches are assumed to exploit the temporal locality of memory accesses, β is minimal when an entire row is read or written after being activated, which is about 0.01 (= (the burst size) / (the row size of a $\times 8$ device) = 64b / 8,192b) and the most favorable case in the conventional scheme. In this case, the read energy overhead of spare/ECC was 19.63% while that of CiDRA $ was 3.44% for a SBER of 10^{-5}. With the Bloom filter, the energy overhead of CiDRA became 1.21%, 3 times lower than that of CiDRA without it. For β of 0.27, the average row-buffer conflict rate for SPEC CPU2006 benchmarks reported in [20], the read energy overhead of CiDRA $, $/BF, and $/BF/ECC was 3.44%, 0.79%, and 0.27%, respectively. Similar trends were observed for the write energy, but spare/ECC had an order of magnitude or more higher overhead than that of CiDRA $/BF/ECC because a read-modify-write is required per write and a longer message necessitates more GDSAs. Because each application has a different β, we analyze the system-level impact of CiDRA in real applications in Section IV-D.

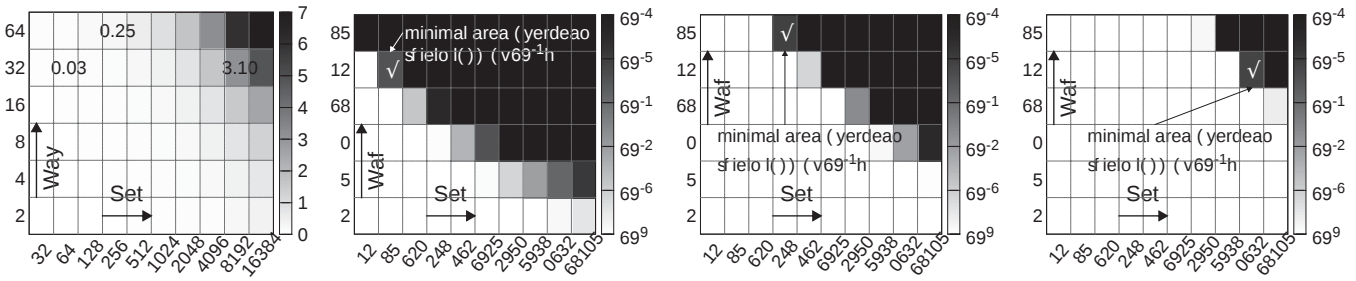
Third, CiDRA improved both the read access latency (*tAA*) and write bandwidth (*tCCD_WR*) compared to the conventional schemes because ECC encoding/decoding processes are mostly off the critical paths, read-modify-writes are not needed, and the CiDRA schemes are more area efficient. ECC decoding latency was added to all read accesses for the conventional in-DRAM ECC scheme. The spare/ECC had a much higher *tCCD_WR* than CiDRA because reading old values, comparing with new ones, and encoding ECC values are needed to perform a write. Because its area overhead increases rapidly with higher SBERs, the inter-bank datalines and control path get longer and incurs further increases in *tAA*. This area overhead brings the same problem to other resilience schemes. The enhanced in-DRAM ECC scheme also requires ECC decoding and encoding for the bursts specified by the position bits, which worsens the *tAA* as shown in Figure 9. However, the burst position remapping technique explained in Section III-C mostly amortized the impact of these coding latencies on *tAA*.

C. Exploring the Design Space of CiDRA Architecture

CiDRA is similar to the conventional resilience scheme with spare rows in that its area overhead grows and its device failure rate (yield loss) improves as more sets or ways are populated, but CiDRA is more area efficient. Figure 10 shows the device failure rate and the area overhead of CiDRA $/BF where the number of ways and sets of the CiDRA cache is varied. Between the ways and sets, the yield loss is more sensitive to the former, similar to the address remapping unit case of the conventional scheme using spare rows.

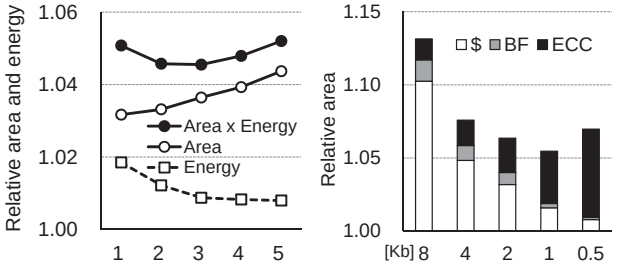
The number of hash functions for the Bloom filter influences both the energy efficiency and the area overhead, and their correlation should be carefully considered. Figure 11(a) shows the area and energy overhead as we varied the number of the Bloom filter hash functions for a SBER of 10^{-5}. Populating more hash functions decreases the probability of false positive cases, but increases the Bloom filter storage and the average number of storage accesses, making 3 hash functions an optimal case in the product of energy and area for a SBER of 10^{-5}. The circuit-level analysis shows that the hash functions can be accessed sequentially so that only one Bloom filter storage is needed. Partitioning the CiDRA cache and associating each portion with I/O pads is more area and energy efficient than having a single CiDRA cache per device. The partitioned CiDRA cache was used throughout the evaluation. For example, at a SBER of 10^{-5}, Figure 10(d) shows that a 32-way cache with 8,192 sets minimizes the area overhead for a device failure rate of 0.1%. This cache requires 17 bits per tag way on a $\times 8$ 8Gb DRAM device, corresponding

[3]We set the message size of the conventional in-DRAM ECC as 128 bits, which gives the best area efficiency as shown in Figure 4(a).



(a) Area overhead (%) (b) Yield loss (SBER 10^{-7}) (c) Yield loss (SBER 10^{-6}) (d) Yield loss (SBER 10^{-5})

Fig. 10: (a) The area overhead and the device yield loss of CiDRA with a cache and Bloom filter (\$/BF) when SBER is (b) 10^{-7}, (c) 10^{-6}, and (d) 10^{-5}, where we mark the configurations that achieve minimal area overhead for a yield loss of 10^{-3}.



(a) Number of hash functions (b) ECC chunk size (Kb)

Fig. 11: (a) The relative area and energy when the number of hash functions is varied for SBER of 10^{-5} on CiDRA \$/BF and (b) the relative area when the chunk size is varied for SBER of 10^{-4} on CiDRA \$/BF/ECC.

to a 4.5Mb tag array size and $17 \times 32 = 544$ bits compared per cache access. With three hash functions, the Bloom filter storage is 0.34Mb and is accessed 1.8 times when the access addresses are evenly distributed. The row size of an SRAM mat in the access-energy optimized Bloom filter storage found by CACTI-D is 64b, resulting in three times energy difference between the Bloom filter and the CiDRA cache.

Combining the enhanced in-DRAM ECC with the CiDRA cache and Bloom filter is critical for high SBERs, for which the primary design choice is the chunk size of the in-DRAM ECC. Figure 11(b) shows the area overhead breakdown of CiDRA \$/BF/ECC as we swept the chunk size of the in-DRAM ECC for a SBER of 10^{-4} and a target device failure rate of 0.1%. As the chunk size decreases, the in-DRAM ECC can correct more bit errors with a higher area overhead, which reduces the number of remaining bit errors that should be fixed by the CiDRA cache. Due to the diminishing return of the number of fixable errors by populating more chunks within a device, the chunk size that gives the minimum area overhead is 1Kb.

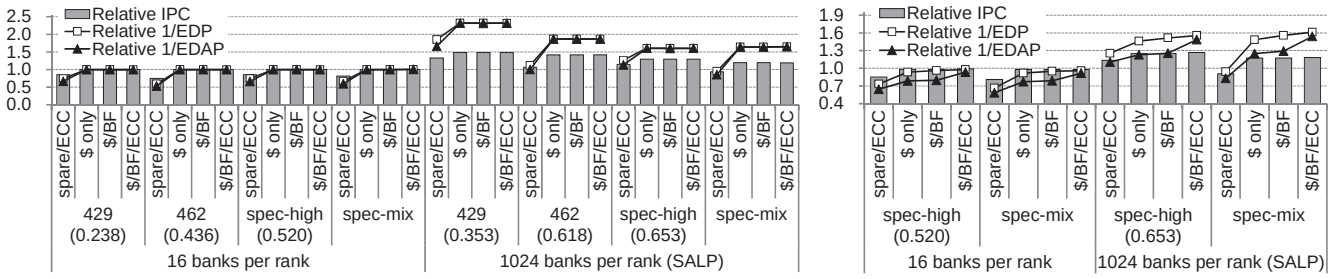
D. Evaluating the System-Level Impact of CiDRA

The energy and latency gains of the CiDRA architectures translate into system-level benefits over various SBERs as shown in Figure 12, in which the numbers within the parentheses are the IPC values of the baseline configurations. Again, we assumed a target yield loss of 0.1%, while the same trends were observed for other target failure rates. For a SBER of 10^{-6}, the CiDRA schemes provided better instructions per cycle (IPCs), energy-delay products (EDP), and energy-delay-area products (EDAP) than spare/ECC, for which we report the reciprocals for EDP and EDAP. SALP [21] effectively increased the number of row buffers exploitable in a system and mitigated the performance penalty of the spare/ECC due to a high *tAA* and *tCCD_WR*, but CiDRA also took advantages of SALP and still performed noticeably better. There is little difference between CiDRA \$, \$/BF, and \$/BF/ECC because the area and energy overheads are not much distinguished. For a SBER of 10^{-4}, both the Bloom filter and the enhanced in-DRAM ECC became more effective. The Bloom filter provided 2.1% and 3.3% gains in 1/EDAP, respectively, over CiDRA \$ on average for main memory bandwidth demanding applications (spec-high) and the their mixes on a multi-programmed workload (spec-mix). The enhanced in-DRAM ECC provided 18.4% and 19.3% additional gains in 1/EDAP, respectively, over CiDRA \$/BF on spec-high and spec-mix. The conventional scheme combining in-DRAM ECC with spare rows can support a SBER of 10^{-4}, but only with significant area, energy, and performance penalties.

V. Related Work

DRAM often employs ECC to protect against soft errors [14]. Because soft errors occur rarely, a SECDED code per 64-bit message (a unit of data transfer of a DIMM) is sufficient for this, requiring one or two more DRAM devices per rank to store 8 bits of parity for every 64-bit message. Although such a rank-level ECC can be used to repair defective cells, SECDED codes are not sufficient to cope with higher cell failure rates arising from further DRAM technology scaling, which often leads to more than one failure per 64 bits. To battle against higher cell failure rates, we can provide stronger ECC, such as DEC and Chipkill [8], [14]. However, such solutions have significant area and energy overhead because more DRAM devices are needed per rank to store stronger codes (DEC). For Chipkill, two DRAM ranks should be ganged together and operate in tandem or $\times 4$ devices are required to deal with a device failure. In contrast, our CiDRA architecture combines the cache and the enhanced in-DRAM ECC in an area efficient manner and allows the rank-level ECC to focus on soft errors.

Prior studies have also suggested alternative implementations which dedicate a small portion of the overall memory space to metadata for ECC or other resilience schemes [31], [47], [50]. Such implementations allow stronger resilience schemes or codes and may not require additional ECC DRAM devices. Yoon et al. propose Virtualized ECC [50], which introduces a structure similar to a page table that allows ECC metadata to be retrieved. Similarly, LOT-ECC [47] incorporates simple ECC into each DRAM row. Nair et al. propose ArchShield [31] to tolerate high failure rates by allocating



(a) SBER = 10^{-6} (b) SBER = 10^{-4}

Fig. 12: The IPC, 1/EDP (energy×delay), and 1/EDAP (EDP×area) of conventional (in-DRAM ECC with spare rows) and CiDRA ($, $/BF, $/BF/ECC) schemes on single- and multi-programed workloads for (a) SBER of 10^{-6} and (b) SBER of 10^{-4}.

a small region of memory to duplicate all faulty words. ArchShield uses standard SECDED to correct failures when possible and checks a fault map, which is located at main memory and cached in LLC, on every main memory access. Therefore, a main memory access on ArchShield accompanies one or more additional memory accesses, which could induce high main memory bandwidth and latency penalties depending on access patterns, especially when the accesses have low spatial locality and frequent writes. Also, there have been studies improving resilience for caches that operate at low supply voltages. VS-ECC [4] targets high SBERs (10^{-3} or even even higher) and requires changes in operating systems. Bit-fix [48] demands multiple steps to repair cache lines, which would incur significant latency penalty to DRAMs that have slow transistors. CiDRA is different from these proposals in that it is transparent to operating systems, requires no or minimal modification to existing MCs, and incurs no penalty to the main memory bandwidth and minimal penalty to the access latency values. Preliminary studies augmenting a DRAM device with an SRAM cache [27] and Bloom filter [33] neither analyzed the interaction between the in-DRAM and rank-level ECCs nor combined the CiDRA cache and filter with an enhanced in-DRAM ECC.

Aggressively reducing the refresh rate to lower the power dissipated by DRAM refreshes can lead to a high SBER because leakier cells will lose their values before they can be refreshed [24], [25]. While all the aforementioned resilience techniques can be used to tolerate lower refresh rates, some approaches dedicated for this particular problem are proposed. RAIDR [25] groups DRAM rows into different retention time bins so that each bin can be refreshed at its own minimum rate; this approach cannot repair dead cells that CiDRA targets because they still malfunctions regardless of the refresh rate. SECRET, on the other hand, selectively allocates ECC metadata in a MC to correct leaky cells [24].

Resistive memory, such as Phase-Change Memory, as a potential DRAM replacement can suffer from even higher BERs because of the defect mechanisms unique to the resistive memory (e.g., wearout) [38]. Such higher failure rates of the resistive memory must be coped with resilience schemes using more complicated codes (e.g., [37]) because the resistive memory uses DRAM for large LLC. Therefore, the solutions that improve the resilience of the resistive memory often sacrifice latency, especially the worst case latencies [36], [38]. In contrast, using even SECDED codes for conventional in-DRAM ECC can notably degrade memory system performance because encoding/decoding the codes is on the DRAM access critical path. On the other hand, our enhanced in-DRAM ECC incurs such coding penalty only for bursts with defects.

VI. Conclusion

We have proposed a novel DRAM resilience architecture that targets single-bit errors prevalent in modern DRAM devices called CiDRA. It has a cache accessed concurrently with normal DRAM banks to detect and replace the accesses to the bursts having faulty cells. In order to significantly reduce the energy consumed for checking each access, CiDRA attaches a low-complexity Bloom filter in front of the cache, which is ideal because the absolute single-bit error rate (SBER) is low, and the faulty cells are identified once during the testing phase or system boot-up. The lower access time of the CiDRA cache with the Bloom filter over the normal DRAM banks is further exploited by partitioning the cache and sequentializing the filter and cache accesses. For high SBERs, even the size of the area-efficient CiDRA cache and Bloom filer becomes problematic, and we proposed an enhanced in-DRAM ECC and combined it with the cache and filter. We introduced chunks, each holding multiple bursts while only one burst specified by position bits can be fixed by parity bits, to improve the area efficiency over the conventional in-DRAM ECC by making the enhanced in-DRAM ECC deal with most sporadic bit failures whereas the CiDRA cache corrects the remaining pathological errors. CiDRA is an order of magnitude better in area efficiency than traditional resilience schemes with spare rows and in-DRAM ECC over a wide range of SBERs and target device failure rates without sacrificing energy efficiency. CiDRA shows that the cost of achieving a certain level of DRAM resilience can be significantly reduced by adequately applying architectural techniques to conventional DRAM structures.

Acknowledgment

This work was supported in part by the National Research Foundation of Korea grant funded by the Korea government (2014R1A2A1A11052936) and by generous grants from NSF (CCF-1016262) and DARPA (HR0011-12-2-0019). Nam Sung Kim has a financial interest in AMD and Samsung Electronics.

References

[1] "JEDEC DDR3 Memory Buffer Standard." [Online]. Available: http://www.jedec.org/standards-documents/docs/jesd82-30

[2] "JEDEC DDR4 SDRAM Standard." [Online]. Available: http://www.jedec.org/standards-documents/docs/jesd79-4

[3] J. Ahn, S. Li, S. O, and N. P. Jouppi, "McSimA+: A Manycore Simulator with Application-level+ Simulation and Detailed Microarchitecture Modeling," in *ISPASS*, Apr 2013.

[4] A. R. Alameldeen, I. Wagner, Z. Chishti, W. Wu, C. Wilkerson, and S.-L. Lu, "Energy-efficient Cache Design Using Variable-strength Error-correcting Codes," in *ISCA*, 2011.

[5] K. Arimoto, Y. Matsuda, K. Furutani, M. Tsukude, T. Ooishi, K. Mashiko, and K. Fujishima, "A Speed-enhanced DRAM Array Architecture with Embedded ECC," *JSSC*, vol. 25, no. 1, 1990.

[6] A. S. Balkir, I. Foster, and A. Rzhetsky, "A Distributed Look-up Architecture for Text Mining Applications using MapReduce," in *SC*, 2011.

[7] S. Baloch, T. Arslan, and A. Stoica, "Efficient Error Correcting Codes for On-Chip DRAM Applications for Space Missions," in *Aerospace Conference*, 2005.

[8] R. Blankenship, D. Brzezinski, and E. Valverde, "Memory Error Detection and/or Correction," 2012, US Patent 8,250,435.

[9] B. H. Bloom, "Space/Time Trade-offs in Hash Coding with Allowable Errors," *Commun. ACM*, vol. 13, no. 7, Jul. 1970.

[10] K. Furutani, K. Arimoto, H. Miyamoto, T. Kobayashi, K. Yasuda, and K. Mashiko, "A Built-in Hamming Code ECC Circuit for DRAMs," *JSSC*, vol. 24, no. 1, 1989.

[11] M. Ghosh and H. H. S. Lee, "Smart Refresh: An Enhanced Memory Controller Design for Reducing Energy in Conventional and 3D Die-stacked DRAMs," in *MICRO*, 2007.

[12] J. L. Henning, "SPEC CPU2006 Benchmark Description," *Computer Architecture News*, vol. 34, no. 4, 2006.

[13] S. Hong, "Memory technology trend and future challenges," in *IEDM*, 2010.

[14] B. Jacob, S. Ng, and D. Wang, *Memory Systems: Cache, DRAM, Disk*. Morgan Kaufmann Publishers Inc., 2007.

[15] R. Kaiser and F. Schamberger, "Method for Comparing the Address of a Memory Access with an Already Known Address of a Faulty Memory Cell," 2007, US Patent 7,181,643.

[16] H. L. Kalter, C. H. Stapper, J. E. Barth Jr, J. DiLorenzo, C. E. Drake, J. A. Fifield, G. A. Kelley Jr, S. C. Lewis, W. B. van der Hoeven, and J. A. Yankosky, "A 50-ns 16-Mb DRAM with a 10-ns data rate and on-chip ECC," *JSSC*, vol. 25, no. 5, 1990.

[17] U. Kang, H.-S. Yu, C. Park, H. Zheng, J. Halbert, K. Bains, S. Jang, and J. S. Choi, "Co-Architecting Controllers and DRAM to Enhance DRAM Process Scaling," in *The Memory Forum Workshop collocated at ISCA*, 2014.

[18] B. Keeth, R. J. Baker, B. Johnson, and F. Lin, *DRAM Circuit Design: Fundamental and High-Speed Topics*. IEEE Press, 2007.

[19] N. S. Kim, S. Draper, S.-T. Zhou, S. Katariya, H. Ghasemi, and T. Park, "Analyzing the Impact of Joint Optimization of Cell Size, Redundancy, and ECC on Low-Voltage SRAM Array Total Area," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 20, no. 12, 2012.

[20] Y. Kim, M. Papamichael, O. Mutlu, and M. Harchol-Balter, "Thread cluster memory scheduling: Exploiting differences in memory access behavior," in *MICRO*, 2010.

[21] Y. Kim, V. Seshadri, D. Lee, J. Liu, and O. Mutlu, "A Case for Exploiting Subarray-Level Parallelism (SALP) in DRAM," in *ISCA*, Jun 2012.

[22] J. G. Lee, Y. H. Jun, K. H. Kyung, C. Yoo, Y. H. Cho, and S. I. Cho, "A New Column Redundancy Scheme for Yield Improvement of High Speed DRAMs with Multiple Bit Pre-fetch Structure," in *VLSI*, 2001.

[23] S. Li, J. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "The McPAT Framework for Multicore and Manycore Architectures: Simultaneously Modeling Power, Area, and Timing," *ACM TACO*, vol. 10, no. 1, Apr 2013.

[24] C.-H. Lin, D.-Y. Shen, Y.-J. Chen, C.-L. Yang, and M. Wang, "SECRET: Selective Error Correction for Refresh Energy reducTion in DRAMs," in *ICCD*, 2012.

[25] J. Liu, B. Jaiyen, R. Veras, and O. Mutlu, "RAIDR: Retention-Aware Intelligent DRAM Refresh," in *ISCA*, 2012.

[26] Y. Liu, M. Chi, A. Mittal, G. Aluri, S. Uppal, P. Paliwoda, E. Banghart, K. Korablev, B. Liu, M. Nam, M. Eller, and S. Samavedam, "Anti-fuse Memory Array Embedded in 14nm FinFET CMOS with Novel Selectorless Bit-cell Featuring Self-rectifying Characteristics," in *VLSI*, 2014.

[27] M. Lucente, C. Harris, and R. Muir, "Memory System Reliability Improvement Through Associative Cache Redundancy," *JSSC*, vol. 26, no. 3, Mar 1991.

[28] K. T. Malladi, I. Shaeffer, L. Gopalakrishnan, D. Lo, B. C. Lee, and M. Horowitz, "Rethinking DRAM Power Modes for Energy Proportionality," in *MICRO*, 2012.

[29] Y. Mori, K. Ohyu, K. Okonogi, and R.-i. Yamada, "The Origin of Variable Retention Time in DRAM," in *IEDM*, 2005.

[30] O. Mutlu and T. Moscibroda, "Parallelism-Aware Batch Scheduling: Enhancing both Performance and Fairness of Shared DRAM Systems," in *ISCA*, 2008.

[31] P. Nair, D.-H. Kim, and M. K. Qureshi, "ArchShield: Architectural Framework for Assisting DRAM Scaling by Tolerating High Error Rates," in *ISCA*, 2013.

[32] R. Naseer and J. Draper, "Parallel Double Error Correcting Code Design to Mitigate Multi-bit Upsets in SRAMs," in *European Solid-State Circuits Conference*, 2008.

[33] S. O, S. Kwon, Y. H. Son, Y. Park, and J. Ahn, "CIDR: A Cache Inspired Area-Efficient DRAM Resilience Architecture against Permanent Faults," *IEEE Computer Architecture Letters*, preprint.

[34] S.-C. Oh, J.-H. Kim, H.-J. Choi, S.-D. Choi, K.-T. Park, J.-W. Park, and W.-J. Lee, "Automatic Failure Analysis System for High Density DRAM," in *Proceedings of International Test Conference*, 1994.

[35] W. W. Peterson and J. E. J. Weldon, *Error Correcting Code*, 2nd ed. MIT Press, 1972.

[36] M. K. Qureshi, "Pay-As-You-Go: Low-overhead Hard-error Correction for Phase Change Memories," in *MICRO*, 2011.

[37] M. K. Qureshi, J. Karidis, M. Franceschini, V. Srinivasan, L. Lastras, and B. Abali, "Enhancing Lifetime and Security of PCM-based Main Memory with Start-gap Wear Leveling," in *MICRO*, 2009.

[38] S. Schechter, G. H. Loh, K. Straus, and D. Burger, "Use ECP, Not ECC, for Hard Failures in Resistive Memories," in *ISCA*, 2010.

[39] B. Schroeder, E. Pinheiro, and W.-D. Weber, "DRAM Errors in the Wild: A Large-Scale Field Study," in *SIGMETRICS*, 2009.

[40] S. Sethumadhavan, R. Desikan, D. Burger, C. R. Moore, and S. W. Keckler, "Scalable Hardware Memory Disambiguation for High ILP Processors," in *MICRO*, 2003.

[41] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder, "Automatically Characterizing Large Scale Program Behavior," in *ASPLOS*, Oct 2002.

[42] Y. H. Son, S. O, Y. Ro, J. W. Lee, and J. Ahn, "Reducing Memory Access Latency with Asymmetric DRAM Bank Organizations," in *ISCA*, Jun 2013.

[43] V. Sridharan and D. Liberty, "A Study of DRAM Failures in the Field," in *SC12*, 2012.

[44] V. Sridharan, J. Stearley, N. DeBardeleben, S. Blanchard, and S. Gurumurthi, "Feng Shui of Supercomputer Memory: Positional Effects in DRAM and SRAM Faults," in *SC13*, 2013.

[45] T. Takahashi, T. Sekiguchi, R. Takemura, S. Narui, H. Fujisawa, S. Miyatake, M. Morino, K. Arai, S. Yamada, S. Shukuri, M. Nakamura, Y. Tadaki, K. Kajigaya, K. Kimura, and K. Itoh, "A multigigabit DRAM Technology with 6F2 Open-Bitline Cell, Distributed Overdriven Sensing, and Stacked-Flash Fuse," *JSSC*, vol. 36, no. 11, 2001.

[46] S. Thoziyoor, J. Ahn, M. Monchiero, J. B. Brockman, and N. P. Jouppi, "A Comprehensive Memory Modeling Tool and Its Application to the Design and Analysis of Future Memory Hierarchies," in *ISCA*, 2008.

[47] A. N. Udipi, N. Muralimanohar, R. Balsubramonian, A. Davis, and N. P. Jouppi, "LOT-ECC: Localized and Tiered Reliability Mechanisms for Commodity Memory Systems," in *ISCA*, 2012.

[48] C. Wilkerson, H. Gao, A. R. Alameldeen, Z. Chishti, M. Khellah, and S.-L. Lu, "Trading off Cache Capacity for Reliability to Enable Low Voltage Operation," in *ISCA*, 2008.

[49] C. Yang, Y. Emre, C. Chakrabarti, and T. Mudge, "Flexible Product Code-based ECC Schemes for MLC NAND Flash Memories," in *Signal Processing Systems, IEEE Workshop on*, 2011.

[50] D. H. Yoon and M. Erez, "Virtualized and Flexible ECC for Main Memory," in *ASPLOS*, 2010.