CIDR: A Cache Inspired Area-Efficient DRAM Resilience Architecture against Permanent Faults

Seongil O, Sanghyuk Kwon, Young Hoon Son, Yujin Park, and Jung Ho Ahn

Abstract—Faulty cells have become major problems in cost-sensitive main-memory DRAM devices. Conventional solutions to reduce device failure rates due to cells with permanent faults, such as populating spare rows and relying on error-correcting codes, have had limited success due to high area overheads. In this paper, we propose CIDR, a novel cache-inspired DRAM resilience architecture, which substantially reduces the area overhead of handling bit errors from these faulty cells. A DRAM device adopting CIDR has a small cache next to its I/O pads to replace accesses to the addresses that include the faulty cells with ones that correspond to the cache data array. We minimize the energy overhead of accessing the cache tags for every read or write by adding a Bloom filter in front of the cache. The augmented cache is programmed once during the testing phase and is out of the critical path on normal accesses because both cache and DRAM arrays are accessed in parallel, making CIDR transparent to existing processor-memory interfaces. Compared to the conventional architecture relying on spare rows, CIDR lowers the area overhead of achieving equal failure rates over a wide range of single-bit error rates, such as 23.6× lower area overhead for a bit-error rate of 10^{-5} and a device failure rate of 10^{-3}.

Index Terms—DRAM, error resilience, permanent faults, row and column sparing, Bloom filter, DRAM-side caching

1 Introduction

DRAM has been the de-facto standard for main-memory for decades and has enjoyed steady improvement in bandwidth and capacity thanks to advances in process technology. At the same time, smaller DRAM cells and peripheral circuitry have become more vulnerable to defects, such as manufacturing imperfections and process variations. This causes not only severe fluctuations in the DRAM retention time [8], [9], but also the malfunctioning of various components, leading to DRAM device failures [13]. However, there have been few architectural proposals to improve the resilience of main-memory DRAM devices under various faults [10] except for the ones dealing with the DRAM retention time issue.

DRAM makers have provided spare rows and columns within the 2D arrays of cells (each storing a bit) to cope with various types of permanent faults [5] during the fabricating/testing procedures, but this scheme suffers from the following limitations. First, single-bit failures, known to be uniform-randomly distributed [11], occupy the majority of DRAM failures [7], [13] 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 single-bit error is a huge waste in resources. Second, it is in the critical path of normal DRAM operations to identify a row or column with faulty cells and to replace it with a spare. This induces a trade-off between the strength of the replacement logic and the energy/latency of DRAM operations.

We propose a DRAM architecture capable of tolerating permanent single-bit failures with significantly improved area efficiency with no modification to conventional processor-memory interfaces. We add a small cache per DRAM device to replace the access to one of the addresses that include the faulty cells with one that corresponds to the cache data array. Even if the cache consists of bulky SRAM cells, dedicating a few dozen SRAM cells for tagging and supplying a substitute is still much more efficient in the area than wasting thousands of DRAM cells for a single-bit fault. To minimize the energy overhead of accessing the cache for each DRAM read or write, we filter the cache accesses using a Bloom filter [3]. This DRAM-side cache is completely out of the critical path on normal accesses because both the cache and DRAM arrays can be accessed concurrently. In addition, the cache and Bloom filter operate much faster than normal DRAM structures due to their small capacity. We exploit this advantage in latency by sequentializing the filter and cache accesses [4] to further improve the energy and area efficiency of the proposed architecture.

We show that this novel cache-inspired DRAM resilience architecture, CIDR, has an order of magnitude smaller area overhead than that of the conventional resilience structure over a wide range of single-bit error rates (SBER) [10] when targeting the same device failure rates due to single-bit faults. The relative energy of the CIDR over the conventional architecture depends on the target failure rates, the bit-error rates, and the device access patterns, but CIDR is at least as efficient as the conventional architecture for most typical usage scenarios. For a bit-error rate of 10^{-5} and a target device failure rate of 10^{-3}, the area and energy overhead of the CIDR is 1.38 and 0.36 percent, respectively, which is 23.6 and 3.2 times lower than those of the conventional architecture assuming a DRAM row-buffer conflict rate of 27 percent, the average of the SPEC CPU2006 applications on a typical chip-multiprocessor system [6].

2 Conventional DRAM Redundancy Structures

Contemporary DRAM devices are structured to accommodate billions of cells in an area-efficient manner and exploit the spatial locality abundant in main-memory accesses. A device is organized as several two-dimensional arrays of cells called banks (Fig. 1). A command called activate is first applied to the device and a row in a bank specified by an address accompanying the command is activated to a row buffer, and then, the data in the buffer are accessed by one of more read or write commands to the active row. A bank has a row decoder to specify an active row and a column decoder to select the locations in the row buffer for access. The number of bits transferred per read or write command is the product of the number of data pins (data I/O width) per device (×N) and its burst length (e.g., 8 for DDR3/4 [1]). tCCD, the minimal timing interval between any read or write command, determines the peak data transfer rate of a device.

Each DRAM bank has spare rows and columns to improve the yield or reduce failure rates that originate from various faulty components [5]. In order to replace a row or column with faults with a spare, an address remapping unit is placed in front of a decoder. The unit compares an input address with the known faulty row or column indices and if both match, it supersedes the address with a fault-free spare. The faulty rows and columns are identified during the testing phase of DRAM manufacturing and their addresses are recorded in laser or electric fuses. These bulky fuses can be gathered at one place or replaced with non-volatile memories within or outside of the device. The remapping unit is functionally similar to a cache where its tag array stores faulty addresses, the data array holds spares, and the way is the number of addresses to be compared.

This conventional resilience scheme with spare rows and columns is ideal when entire rows or columns malfunction, but it is extremely area-inefficient for single-bit faults prevalent in modern DRAM devices [7] because there are thousands of cells per row or column. As the capacity of a bank approaches a billion bits, the

• *The authors are with the Department of Transdisciplinary Studies, Seoul National University, Seoul, Korea.*
E-mail: {swdfish, kkwon114, yhson96, comesay, gajh}@snu.ac.kr.

Manuscript received 19 Feb. 2014; revised 6 May 2014; accepted 12 May 2014. Date of publication 15 May 2014; date of current version 19 June 2015.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/LCA.2014.2324894

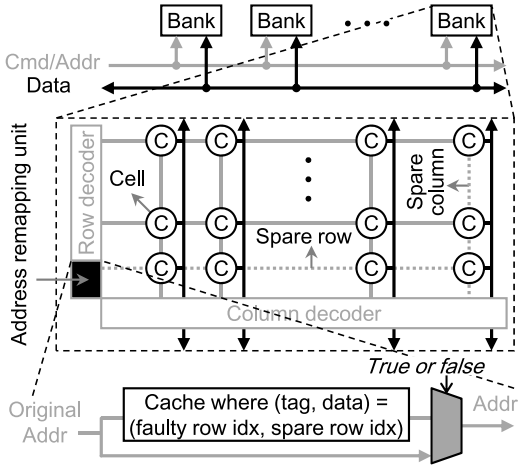

Fig. 1. A conventional DRAM redundancy structure with spare (redundant) rows and columns.

size of a row which is fixed by a DRAM specification is around several thousand bits (e.g., 8K bits on ×8 8 Gb DDR4 devices [1]) to limit row activate energy, and multiple columns are accessed per read/write command because DRAM accesses are bursty. These all cause far fewer DRAM cells to be affected by a row decoder than a column decoder, which make it more efficient to utilize a spare row instead of a spare column to fix single-bit faults. 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 the conventional scheme is that the address remapping unit is in the critical path of the activate, read, and write DRAM operations. It is desirable that a row with faulty bits or a faulty row can be mapped to any spare row in a bank by the remapping unit like a fully associative cache because fewer sets typically lead to lower miss rates, which in turn reduce device failure rates for a given single-bit error rate. However, having fewer sets or more ways per cache increases not only the energy but also the latency of an access. Therefore, it is critical for DRAM manufacturers to find the right balance between the strength 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, but as the rate increases the overheads grow rapidly. We consider various SBERs because the component fault rates are top secrets to DRAM manufacturers [10]. Fig. 2 shows the device failure rates and the area overheads of using spare rows for various ways and sets of the remapping unit. The latency and energy change as well, but they fluctuate far smaller than the area and hence omitted due to limited space. We assume a 28 nm 8 Gb ×8 DDR4-2400 die, which has four bank groups and four banks per group. We modified CACTI-D [14] to model the remapping units and spare rows/columns. For each SBER, the baseline is the device with 640 spare rows and 80 spare columns per bank, which is derived from the overhead of the row/column redundancy reported in [7]. We only present the cases of SBERs higher than or equal to 10^{-7} when single-bit errors cause noticeable overheads. For a given SBER, having more sets and ways reduces the failure rate, while it is more sensitive to the ways. However, populating more ways to the remapping unit increases the row activate energy, latency, and device area. As SBER increases, these overheads grow rapidly and for a SBER of 10^{-5} and a target failure rate of 10^{-3}, the area overhead becomes 32.5 percent, which is too high and necessitates a solution dedicated to bit errors.

3 Cache-Inspired DRAM Resilience Structure

In order to address the area inefficiency problem of the conventional resilience scheme against prevalent and sporadic single-bit

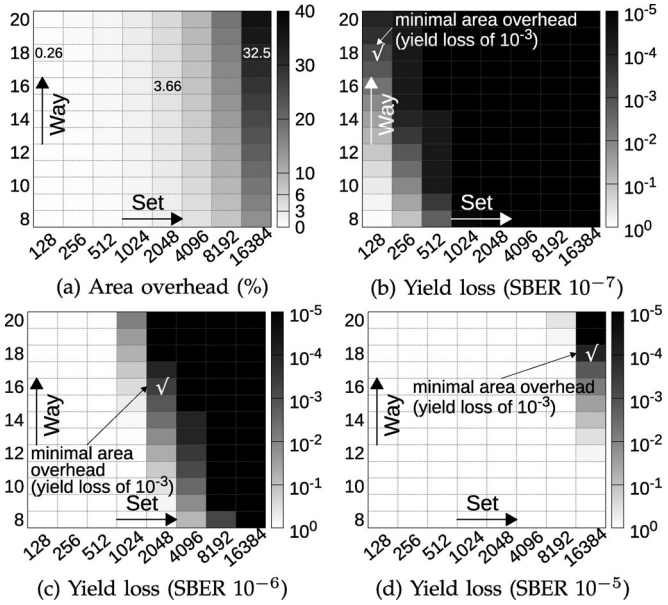

Fig. 2. (a) The area overhead and the device failure rate of the conventional resilience architecture when single-bit error rate is (b) 10^{-7}, (c) 10^{-6}, and (d) 10^{-5}.

errors, we propose a novel cache-inspired DRAM resilience architecture called CIDR. Fig. 3 shows the block diagram of CIDR, which consists of a cache to deal with faulty bits and a Bloom filter to reduce the frequency of vainly approaching the cache for accesses to non-faulty addresses. The DRAM-side cache, or the CIDR cache, is placed close to the I/Os within a device. It receives commands and addresses from the external memory controllers along the banks, stores and combines the row and column addresses on DRAM reads or writes, and supersedes accesses to DRAM structures in case the combined address targets a burst that includes faulty bits. The tag array of the cache stores the DRAM addresses including faulty bits, which is similar to the remapping unit in Section 2. The data array of the cache, in contrast, stores data for addresses at which certain cells malfunction. A device may have one cache; a data pin of a device may have one cache, or a few pins in a device can share a cache depending on the device layout constraints. In this paper, we assume that each data pin has a cache and leave the design space exploration of different cache configurations, such as multi-bank structures, for future work.

CIDR has substantially lower area overhead than the conventional scheme because the former dedicates an address and data bits, at most a few dozen each, to replace a faulty DRAM cell

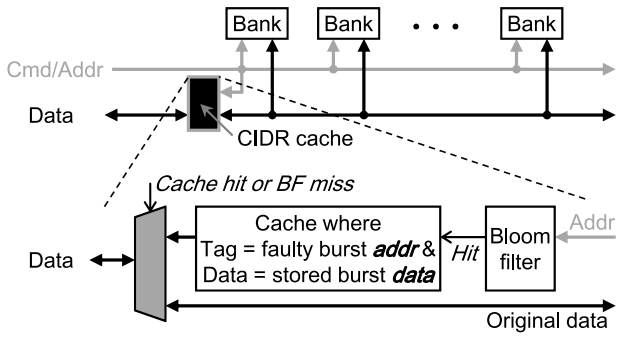

Fig. 3. CIDR consists of a cache and a Bloom filter. It takes the command and address information concurrently with a DRAM access and replaces it if the address turns out to include faulty cells. The Bloom filter reduces access frequency to the cache.

while the latter consumes thousands of DRAM cells per faulty cell. Even though a DRAM cell is smaller than an SRAM cell used to store the cache tag and data arrays, the ratio of SRAM over DRAM in cell size is much smaller than the ratio of the number of DRAM cells over cache SRAM cells needed for a faulty bit. Because the cache and the other DRAM banks can be accessed in parallel, populating more cache ways does not increase the latency of DRAM operations. 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. However, the energy to read the tags and to compare them with the incoming address to check the cache hit is not negligible, especially for the caches with many ways.

CIDR utilizes a Bloom filter [3] to alleviate this cache access energy overhead. A Bloom filter identifies if a certain number belongs to a set of known numbers (faulty addresses for CIDR). It utilizes N_H hash functions that translate the input numbers to other numbers within a finite range (R). The filter also has a storage that consists of R bits, each being cleared (set to 0) in the beginning. During the training phase (the testing procedure for CIDR), 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 phase, it checks the locations corresponding to the outputs of the hash functions for a given input (address for CIDR). If there is any location with the value 0, 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 [2], [12]. The Bloom filter is ideal for CIDR because 1) the inherent false positive cases are handled by the CIDR cache, 2) the target bit-error rates are low so that the false positive rates by the filter can be kept low with few hash functions, and 3) the faulty addresses are identified during the testing phase and do not change during normal DRAM accesses.

We exploit the huge advantage in latency of the CIDR cache with the Bloom filter over normal DRAM banks to make CIDR more energy efficient. First, when the Bloom 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 hash 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 faulty addresses. Second, we can configure the CIDR cache to be pseudo associative [4] such that a cache set is divided into multiple subsets, where each 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 2. Even if the cache cycle time is increased, but as far as the cycle time does not surpass tCCD, it does not slow down DRAM operations. These optimizations are possible because the DRAM bank access time for a read, tAA, is typically higher than twice the tCCD, and the DRAM banks and the CIDR with the Bloom filter are accessed concurrently.

CIDR is complementary to the schemes with Error Correcting Codes (ECC), which mostly target transient faults, and ArchShield [10], which exposes faulty DRAM cells to the architectural level. For example, CIDR can make fewer faulty cells detected to the ArchShield framework, which then tolerates more faults. CIDR can replace not only malfunctioning but also leaky DRAM cells to reduce the average memory access latency. This is similar to SECRET [8], but CIDR is transparent to a processor-memory interface or a memory controller while SECRET augments a memory controller with a bit-level replacement structure. We leave the study on mismatch between DRAM and SRAM in transient fault rates for future work.

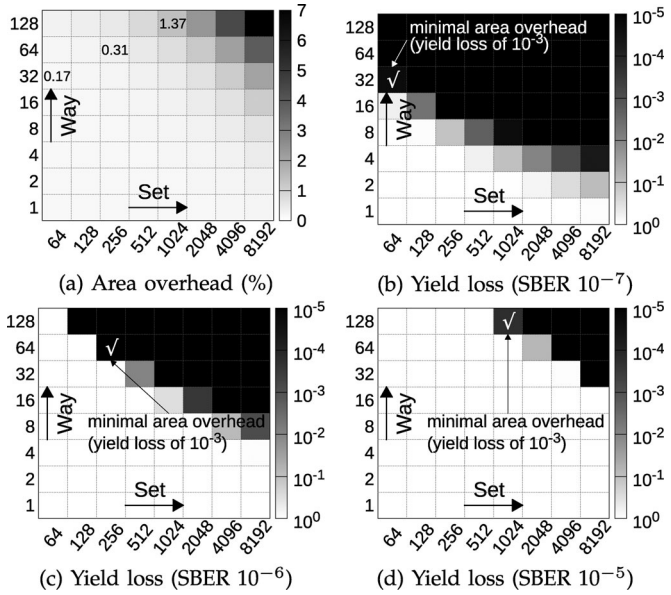

Fig. 4. (a) The area overhead and the device failure rate of the CIDR architecture with a CIDR cache and a Bloom filter when single-bit error rate is (b) 10^{-7}, (c) 10^{-6}, and (d) 10^{-5}.

4 Evaluation

We quantified the area efficiency benefits of the CIDR over the conventional resilience scheme and compared the energy overheads of both schemes for various usage scenarios. We assumed that the SRAM cell size of the CIDR cache and Bloom filter is $150F^2$, which accounts for the layout overhead due to the limited number of metal layers for the DRAM processes (three layers in this study). The range (R) of the hash functions for the Bloom filter is narrower than the address space of the DRAM device, and we implemented a hash function by randomly merging multiple input address bits with XOR gates and shuffling them. CACTI-D was used to model the CIDR cache. Even if the number of data bits per CIDR cache entry can be as small as 1, we assumed that it is equal to the burst length of the device. We assumed that only the DRAM and SRAM cells may malfunction. A spare DRAM row or the CIDR cache entry with faulty cells is not used.

CIDR is similar to the conventional resilience architecture in that its area overhead grows and its device failure rate (yield loss) improves as more sets or ways are populated, but CIDR is more area efficient. Fig. 4 shows the device failure rate and the area overhead of the CIDR with the Bloom filter in which the number of ways and sets of the CIDR cache are varied. Between the ways and sets, the device failure rate is more sensitive to the former, which is similar to the address remapping unit case of the conventional scheme. We show the benefits of CIDR by setting the target device failure rate to 0.1 percent and comparing the relative area and energy of the conventional scheme and the CIDR without/with the Bloom filter for SBERs of 10^{-7}, 10^{-6}, and 10^{-5} in Fig. 5. We report the read energy amortized by the activate energy, but because the read and write energy is similar, observations made here are also applied to the writes. The baseline is again the device with 640 spare rows and 80 spare columns per bank. The conventional scheme has more than an order or magnitude higher area overhead than the CIDR. For a SBER of 10^{-5}, the area overhead of the CIDR with and without the Bloom filter is 1.2 and 1.4 percent, respectively, while that of the conventional scheme is 32.5 percent.

This small area overhead by adding the Bloom filter to the CIDR is compensated by the huge gain in energy efficiency. Because the conventional spare-row-based scheme incurs an energy overhead per activate command, its impact on the DRAM access energy is amortized by the ratio of activates over the sum

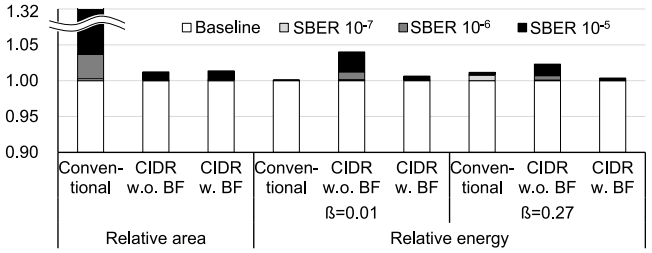

Fig. 5. The relative area and energy of the conventional resilience scheme and CIDR without/with the Bloom filter (CIDR w.o./w. BF). β is the ratio of activate commands over the sum of reads and writes.

of the reads and writes, or the DRAM row-buffer conflict rate (β). In contrast, CIDR incurs an energy overhead per read or write access. Assuming that the CPU-side caches exploit the temporal locality of the memory accesses, β is minimal when an entire row is read or written after being activated, which is 0.01, and is the most favorable case to the conventional scheme. In this case, the energy overhead of the conventional scheme is 0.07 percent while that of the CIDR without the Bloom filter is 4.0 percent for a SBER of 10^{-5}. With the Bloom filter, the energy overhead of the CIDR becomes 0.63 percent, 6.2 times lower than that of the CIDR without it and comparable to the conventional scheme. For a β of 0.27, the average row-buffer conflict rate for the SPEC CPU2006 benchmarks reported in [6], the energy overhead of the CIDR with and without the Bloom filter is 0.36 and 2.3 percent, respectively. Note that we used four hash functions for the Bloom filter, which provides the right balance between the energy consumed by the filter and the energy saved in the CIDR cache. Critical path analysis showed that these four hash functions can be accessed sequentially, where the Bloom filter storage was accessed only 1.8 times on average due to the early pruning explained in Section 3. The CIDR cache tag arrays are several times larger in capacity and compares orders of magnitude more bits than the Bloom filter storage, consuming more energy than the filter.

5 Conclusion

We have proposed CIDR, a novel DRAM resilience architecture that targets permanent single-bit errors prevalent in modern DRAM devices. CIDR has a cache that is accessed concurrently with normal DRAM banks to detect and replace the accesses to the bursts including faulty cells. In order to minimize the energy consumption for checking each access, CIDR attaches a simple Bloom filter in front of the cache, which is ideal because bit-error rates are low and the faulty cells are identified only once during the testing phase. CIDR is transparent to processor-memory interfaces and an order of magnitude better in area efficiency than the spare-row-based resilience scheme over a wide range of bit-error rates and target device failure rates without sacrificing energy efficiency. CIDR 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.

Acknowledgments

This work was supported by IDEC (EDA Tool) and by the Future Semiconductor Device Technology Development Program (10044735) funded By MOTIE and KSRC. Jung Ho Ahn is also with Inter-university Semiconductor Research Center, Seoul National University.

References

[1] JEDEC DDR4 SDRAM Standard [Online]. Available: http://www.jedec.org/standards-documents/docs/jesd79-4, 2012.
[2] A. S. Balkir, I. Foster, and A. Rzhetsky, "A distributed look-up architecture for text mining applications using mapreduce," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, 2011, pp. 1–11.
[3] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, Jul. 1970.
[4] B. Calder, D. Grunwald, and J. Emer, "Predictive sequential associative cache," in *Proc. IEEE 2nd Symp. High-Perform. Comput. Archit.*, 1996, pp. 244–253.
[5] B. Keeth, *DRAM Circuit Design: Fundamental and High-Speed Topics*. Hoboken, NJ, USA: Wiley, 2007.
[6] Y. Kim, M. Papamichael, O. Mutlu, and M. Harchol-Balter, "Thread cluster memory scheduling: Exploiting differences in memory access behavior," in *Proc. IEEE/ACM 43rd Annu. Int. Symp. Microarchit.*, 2010, pp. 65–76.
[7] 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 *Proc. Symp. VLSI*, 2001, pp. 69–70.
[8] 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 *Proc. IEEE 30th Int. Conf. Comput. Des.*, 2012, pp. 67–74.
[9] J. Liu, B. Jaiyen, R. Veras, and O. Mutlu, "RAIDR: Retention-aware intelligent DRAM refresh," in *Proc. 39th Annu. Int. Symp. Comput. Archit.*, 2012, pp. 1–12.
[10] P. J. Nair, D.-H. Kim, and M. K. Qureshi, "ArchShield: Architectural framework for assisting DRAM scaling by tolerating high error rates," in *Proc. 40th Annu. Int. Symp. Comput. Archit.*, 2013, pp. 72–83.
[11] 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 *Proc. Int. Test Conf.*, 1994, pp. 526–530.
[12] S. Sethumadhavan, R. Desikan, D. Burger, C. R. Moore, and S. W. Keckler, "Scalable hardware memory disambiguation for high ILP processors," in *Proc. IEEE/ACM 36th Annu. Int. Symp. Microarchit.*, 2003, pp. 399–410.
[13] V. Sridharan, J. Stearley, N. DeBardeleben, S. Blanchard, and S. Gurumurthi, "Feng Shui of supercomputer memory: Positional effects in DRAM and SRAM faults," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, 2013, pp. 1–11.
[14] S. Thoziyoor, J. H. 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 *Proc. 35th Int. Symp. Comput. Archit.*, 2008, pp. 51–62.