CLEAN-ECC: High Reliability ECC for Adaptive Granularity Memory System

Seong-Lyong Gong
ECE, UT Austin
sl.gong@utexas.edu

Minsoo Rhu
NVIDIA
mrhu@nvidia.com

Jungrae Kim
ECE, UT Austin
dale40@gmail.com

Jinsuk Chung
ECE, UT Austin
chungdna@gmail.com

Mattan erez
ECE, UT Austin
mattan.erez@utexas.edu

ABSTRACT

Adaptive-granularity memory architectures have been considered mainly because of main memory bottleneck and power efficiency. Meanwhile, highly reliable protection schemes are getting popular especially in large computing systems. Unfortunately, conventional ECC mechanisms including *Chipkill* require a large number of symbols to guarantee strong protection with acceptable overhead. We propose a novel memory protection scheme called CLEAN (Chipkill-LEvel reliable and Access granularity Negotiable), which enables us to balance the contradicting demands of fine-grained (FG) access and strong & efficient ECC. To close a potentially significant detection coverage gap due to CLEAN's detection mechanism coupled with permanent faults, we design a simple mechanism access granularity enforcement. By enforcing coarse-grained (CG) access, we can get only the advantage of higher protection comparable to *Chipkill* instead of achieving the adaptive access granularity together. CLEAN showed *Chipkill* level reliability as well as improvement in performance, system and memory power efficiency by up to 11.8%, 10.8% and 64.9% with mixes of SPEC2006 benchmarks.

Categories and Subject Descriptors

B.3.4 [**Memory Structures**]: Reliability, Testing, and Fault-Tolerance; C.4 [**Performance of Systems**]: Reliability, Availability, and Serviceability

Keywords

DRAM Memory, Reliability, Chipkill, Adaptive Granularity Memory System

This is the author version of the work. It is posted here by permission of ACM for your personal use. Not for redistribution. The definitive version was published in the Proceedings of MICRO-48 December 5âĂŞ9, Waikiki, Hawaii, USA

1. INTRODUCTION

Given the high arithmetic performance of chip multiprocessor, the performance bottleneck is often the main memory system. This is particularly true for applications with poor spatial locality because memory systems are typically optimized for cache-line granularity access and squander performance and power efficiency when frequent finer-grained accesses are required. Memory systems that can adapt to available spatial locality have been shown to alleviate the memory bottleneck and improve both performance and efficiency [1, 2, 3]. These mechanisms enable accessing only a subset of the memory devices in each memory rank to reduce power consumption and allow the overlap of memory requests to different groups of devices in a rank (*subranks*) to improve performance.

While these adaptive-granularity architectures offer significant improvement for some applications (e.g., Yoon et al. [2] showed up to 31% speedup on average for a range of memory-intensive workloads), prior work did not sufficiently address the crucial aspect of memory reliability. Recent studies have shown two important trends about memory errors [4, 5, 6, 7, 8]. The first is that memory errors that affect a large number of bits in a single memory transfer are common, necessitating memory protection schemes that are stronger than those proposed previously for adaptive granularity. The second is that memory errors can be quite frequent. Error rates are significant because the rate at which permanent faults occur is on par with transient fault rates, but once a permanent fault occurs many accesses to the faulty device result in memory errors. Thus, there is a practical need for strong memory protection that can tolerate multi-bit errors as well as permanent faults.

Highly-reliable coarse-grained memory systems rely on *chipkill*-level protection and can tolerate errors where all data transferred from a single memory device is erroneous [9, 10, 11]. Prior work on adaptive granularity however does not provide for such strong protection. As we show later, the Dynamic Granularity Memory System(DGMS) [2] cannot correct all errors resulting from 50.3% of faults (which corresponds to fault rate about 31.3 FIT/device) when estimated with the fault rates reported in [6] since it cannot correct single-chip errors.

We propose a memory protection scheme specifically for adaptive-granularity systems. Our *Chip-LEvel reliable and Access-granularity Negotiable* (CLEAN) ECC uses a concatenated code with an inner code used only for error detection and an outer code for correction. This code design enables us to balance the contradicting demands of fine-grained access and strong ECC. We use the inner code for strong detection when performing fine-grained access on subranks and use the outer code for rare correction events, to ensure a low rate of undetected errors (that may lead to silent data corruption), and opportunistically when performing coarse-grained accesses. While this design is reminiscent of RAID [12, 13], we make several new contributions to meet the constraints of memory system design and to adapt granularity and protection, briefly:

- We design a concatenated (two-tier) ECC code that simultaneously meets the requirements of using standard DRAM devices with the standard 12.5% storage overhead and achieving chipkill-level reliability and still enable fine-grained access to fault-free memory locations. The key insights to our design are that: (1) detection is performed for all accesses (fine or coarse) while correction may be limited to when the entire rank is read, and (2) our correction mechanism is far less complex than typical symbol-based *Chipkill* ECC.

- We carefully evaluate both the correction and detection coverage and identify a potentially significant coverage gap that can result in a higher-than-desired undetected error rate. We analyze the main reason for this coverage gap, which is a result of our use of a short code for detection coupled with permanent and intermittent faults or significant read-only data. Based on the analysis, we design a novel hardware technique to close the coverage gap and closely approach both the undetected and corrected error rates of *Chipkill*. Three additional main insights underly the proposed mechanism: (1) our code can be used in a way that increases detection coverage when coarse-grain accesses are performed, (2) the probability of having two independent faults simultaneously occur is extremely low, and (3) coarse-grain access can be forced by hardware to meet coverage goals.

- We combine the above techniques in order to form the CLEAN memory system, which improves performance, system power efficiency, and memory power efficiency by up to 11.8%, 11%, and 65% respectively when compared to a baseline *Chipkill* system; we show that reliability level is similar for important system configurations and that redundancy level is identical.

- We evaluate the hardware overhead of CLEAN using a Verilog implementation and show that it is on par with that of our baseline coarse-grained chipkill-level ECC.

2. BACKGROUND

2.1 Fine-Grained Memory Accesses

Current memory systems generally seek to achieve high capacity and high throughput, which is effectively achieved by optimizing the memory interface for coarse-grained, sequential accesses. Hence a coarse-grained memory system performs well on programs with high spatial locality, increasing peak memory bandwidth while amortizing control overheads. Not all applications, however, can be re-factored to maximize memory bandwidth utilization because non-unit strides, indexed gather / scatter accesses, and other complex access patterns inherently exhibit very low spatial locality. Previous studies have demonstrated that, for applications with low spatial locality, having *fine-grained* memory access capability can help in avoiding unnecessary data transfers, providing substantial improvements in terms of power-efficiency, memory bandwidth utilization, and overall system performance [1, 2, 3]. The cache hierarchy and the memory subsystem of these *adaptive granularity memory systems* (AGMS) can effectively handle a mix of both coarse and fine granularity memory accesses. Below we discuss some key architectural features that are necessary to enable fine-grained management and storage of data in the cache-memory system. We do not propose any innovation in these basic mechanisms and therefore keep the discussion short and refer the reader to prior publications for details [1, 2, 3].

Granularity Decision.

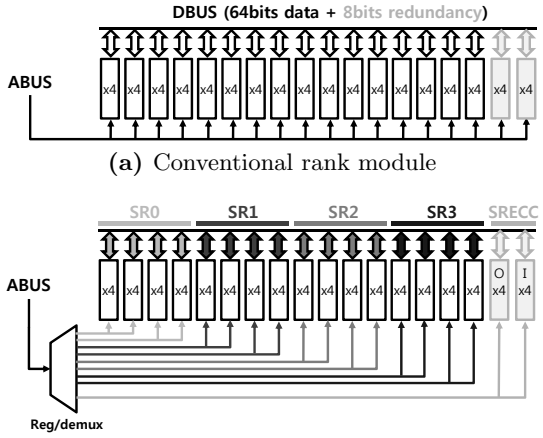
In order to mix access granularities it is necessary to first decide which accesses are coarse and which are fine. This can be done either statically per address or memory instruction [1] or with a dynamic granularity predictor [2, 3]. In addition, the interaction of different granularity accesses may degrade memory scheduling effectiveness and granularity decision may also be based on such factors [2, 3].

Fine-Grained Cache Management.

Because AGMS grants both coarse and fine-grained accesses, the cache hierarchy must be capable of effectively handling both types of accesses. To amortize cache tag-array overhead, a sectored cache design [14, 15] or a more sophisticated derivative, can be used. In a sectored cache, each cache-line is partitioned into multiple sectors where; each sector has its own valid and dirty bits but all the sectors within the cache-line share a common address tag. Thus, the cache maintains the information needed for fine-grained reads and writes with very small overhead.

Fine-Grained Memory Interface.

A conventional CPU uses multiple DRAM chips organized into ranks to provide large coarse-grained accesses to memory (Figure 1 (a)). To provide finer-grained access capability with low overhead, an AGMS leverages a *sub-ranked* memory module design (Figure 1 (b)). Sub-ranked memory systems were originally pro-



(a) Conventional rank module

(b) Sub-ranked module

Figure 1: In our baseline sub-ranked memory system, 18 ×4 chips (16 for data/2 for ECC) are divided into 9 physical subranks (2 chips per subrank); baseline A/DGMS can accesses data in a single 8b-wide subrank while CLEAN requires data accesses of paired subranks to enable strong protection.

posed, for the most part, to reduce the memory access power of many-threaded CPUs by improving row-buffer locality. Such designs include HP's MC-DIMM (multi-core dual in-line memory module) [16, 17], Rambus's threaded memory module [18], the mini-rank memory system [19], and Convey's S/G DIMM (scatter/gather DIMM) [20].In a sub-ranked DIMM, a register chip is used to direct memory command signals to a subset of the DRAM chips in a rank without changing the DRAM device structure itself. Figure 1 shows the sub-ranked memory configuration we assume in our work. The baseline A/DGMS can perform 8B accesses, but to support strong ECC CLEAN requires a minimum access granularity of 16B.

Reliability.

Note that despite AGMS's superior memory bandwidth utilization, the use of fine-grained accesses significantly compromises ECC coverage. Prior work on AGMS [1] only provided SECDED (Single Error Correction, Double Error Detection) protection with very high storage overhead or a weak form of SECDED at conventional 12.5% storage overhead. Unlike current SECDED ECC on full ranks, which can tolerate any single-bit or single-*pin* fault, the SECDED configuration of DGMS [2] can only tolerate single-*bit* faults [2]. This difference results in markedly compromised reliability as shown in Table 1. In particular, DGMS can detect only 74.6% of single chip errors while SECDED can detect almost 99% of single chip errors – this implies that DGMS can result in many more *silent data corruption events* (SDCs) even with single faults (see also Figure 11b). The difference is even greater when compared to currently-standard chipkill-level techniques that can detect and correct all faults confined to a single DRAM.

type	result	SEC-DED	DGMS [2]	*Chipkill* [9]
1bit	CE	100.0000%	100.0000%	100.0000%
	DUE	0.0000%	0.0000%	0.0000%
	SDC	0.0000%	0.0000%	0.0000%
1word	CE	26.6706%	34.8022%	100.0000%
	DUE	55.5483%	53.7213%	0.0000%
	SDC	17.7811%	11.4765%	0.0000%
1pin	CE	100.0000%	3.1393%	100.0000%
	DUE	0.0000%	74.9011%	0.0000%
	SDC	0.0000%	21.9596%	0.0000%
1chip	CE	0.0142%	0.0000%	100.0000%
	DUE	98.8388%	74.6222%	0.0000%
	SDC	1.1470%	25.3778%	0.0000%

Table 1: Estimated fault coverage with respect to single chip errors (methodology detailed in Section 4); CE, DUE, and SDC are corrected errors, detected but uncorrectable errors, and potential silent data corruption, respectively. The overall reliability of DGMS is much worse than even fairly weak SECDED ECC.

2.2 Memory Errors and Protection Mechanisms

Errors in DRAM.

Recent studies [6, 7, 4, 5] indicate that memory system error rates are rising and that the rate of permanent faults is on par with that of transient ones. With a high permanent-fault rate, the likelihood of multi-bit faults, overall error rate, and possibly even accumulated faults increases. As a result, the memory protection level provided with SECDED is no longer sufficient to guarantee reliable operation of the main memory system and stronger *Chipkill*-level ECC is required [11, 9, 10].

Chipkill for Enhanced DRAM Reliability.

Chipkill is a popular ECC mechanism widely adopted in high reliability systems such as commercial server markets. A typical symbol-based *Chipkill* protection scheme provides SSCDSD (Single Symbol Correction, Double Symbol Detection) protection. This means that a *Chipkill*-enabled system is able to correct all errors resulting from a fault to a single chip and detect all errors resulting from faults to two chips. In this way, severe faults to a single chip do not impact reliability and even faults to multiple chips do not result in silent data corruption.

To achieve this, symbols are aligned with chips such that the data transfered from a single chip in a single codeword matches a symbol in the codeword. Because of this alignment, *Chipkill* requires either unacceptably high redundancy or large minimum access granularity (e.g., 64 – 128 bytes per memory access, even with narrow ×4 chips). Thus, it can only be applied to a coarse-grained, full-rank memory system with narrow-interface chips. Recent work presents several alternative DRAM system organizations that minimize ECC storage and which can be used with ×8 and wider DRAM chips while providing strong *Chipkill* reliability [25, 26, 24, 23]. The focus of these studies, however, is on wider interfaces or stronger protection and they are not applicable to our goal of mixing fine-grained and coarse-grained memory accesses.

We summarize the access-granularity and redundancy characteristics of these *Chipkill*-level schemes in Table 2. The challenge with fine-grain access is that re-

	DGMS [2]	*CLEAN*	*Chipkill* [9]	ARCC [21]	MAGE [22]	BambooECC [23]	MultiECC [24]	VECC [25]	LOT-ECC[26]
Fine-grained Access	**Yes**	**Yes**	No	No	No	No	No	No	No
Strong Reliability	No	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**
Access Granularity	8-64B	16-64B	64B	64-128B	64-256B	64B	64B	64B	64B
Adaptive Reliability	No	**Yes**	No	**Yes**	**Yes**	**Yes**	No	No	No
Storage Overhead	**12.5%**	**12.5%**	**12.5%**	**12.5%**	**12.5%**	**3.1-12.5%**	12.9%	18.75%	26.5%
Chip Width	×8	×4	×4	×8	×4-×16	×4	×8	×8	×8

Table 2: Summary of related ECC work; only CLEAN provides both fine-grained memory access and high reliability.

silience is first and foremost bounded by the error detection coverage and that this coverage improves rapidly as the codeword length grows; as explained and demonstrated by Kim et al. [23] for memory systems. As a result, nearly all prior work maintains long codewords and redundancy for strong detection and cannot be used for fine-grain access. This is true even when the schemes are appropriate and effective for a small number of wider chips [25, 22, 21, 24, 27]. For example, Multi-ECC relies on coarse-grain access for detection and only uses a narrow CRC code for localizing an error before correcting it as an erasure [24].

Two exceptions are schemes designed for stacked memories [28, 26] and the design presented for DGMS [2]. These are similar to CLEAN in the use of a RAID parity code for correction and an inner code for detection. However, these schemes were not evaluated for their impact on silent data corruption risk in detail. Using our evaluation framework, we found that in our configuration a scheme like LOT-ECC would have an unacceptable risk of SDC, which is orders of magnitude greater than that with CLEAN; we do not present these results in detail. The same is true for the chipkill-level ECC sketched by Yoon et al. [2], which was not evaluated in detail.

Finally, an inner/outer coding scheme similar to CLEAN has been patented [29], but the patent is vague and we are not aware of any analysis and evaluation. To the best of our knowledge, no prior work has addressed the issue of guaranteeing strong reliability while simultaneously providing fine-grained memory accesses at reasonable overhead. Our two-tier, inner/outer ECC coding mechanism effectively achieves the conflicting goals of *chipkill-level* reliability and the ability to provide fine-grained accesses, while simultaneously incurring low ECC storage overheads (12.5%). We detail our CLEAN ECC in the following section.

3. CLEAN ERROR PROTECTION

This section describes the mechanisms for implementing CLEAN (*Chipkill-LEvel reliable, Access-granularity Negotiable*) ECC design using the baseline dynamic-granularity memory system described in Section 3.1 as a concrete example. Our goal is to provide reliability that is on par with that of *Chipkill*. To achieve this goal of strong protection in concert with fine-grained memory access capability, we utilize a two-tier coding mechanism that relies on a concatenated error code – an *inner* code is used for error detection and an *outer* code is used mostly for correction.

In this section we describe the insights behind CLEAN and the mechanisms for achieving access granularity-negotiable strong protection. We first briefly review the baseline adaptive granularity memory system for which CLEAN is designed (Section 3.1). We then introduce the basic outer code and correction procedure including a description of how the ECC information needed for CLEAN is accessed and updated with minimized memory traffic overhead (Section 3.2). We then describe the inner code and detection mechanisms (Section 3.3), followed by a discussion of how the two codes are combined to localize errors (Section 3.4) and minimize the rate of undetected errors that can lead to silent data corruption (SDC) events (Section 3.5).

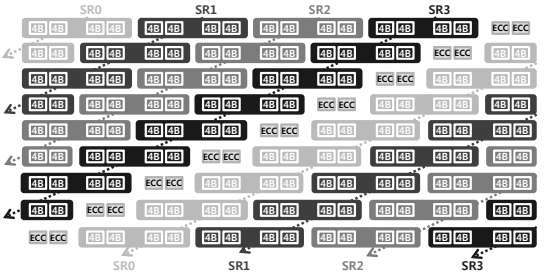


Figure 2: Data/ECC layout used by CLEAN for spreading ECC locations to reduce subrank conflicts; For calculating the rotated positions, a modulo-9 residue generator using efficient parallel designs is available [30].

3.1 Baseline Memory Hierarchy

Memory Subsystem.

As discussed in Section 2, we assume a sub-ranked memory system that enables adaptive (coarse/fine) granularity access. Concretely, we assume that memory ranks consist of 18 ×4 DDR3 DRAM chips (16 chips for data and 2 chips for ECC, having 12.5% ECC overhead), which is standard for current commodity systems that support *Chipkill*-level reliability [9, 10]. CLEAN ECC logically divides the 18 DRAM chips into 5 effective subranks with 4 of the subranks providing data at a minimum granularity of 16B (each subrank consisting of four x4 chips) and the fifth subrank used for storing 2B of ECC information (consisting of two x4 chips). As discussed in the following subsections, one of these two chips is used to store the inner code and the second for the outer-code information. Logically, 16 chips are mapped to the 4 data subranks and 2 chips to the ECC subrank. Physically, the mapping of chips to logical subranks alternates with DRAM addresses to reduce subrank conflicts (Figure 2 as is done in RAID systems and prior work on DGMS [2]). We denote these two chips with 'I' for the inner (detection) code and 'O' for the outer code.

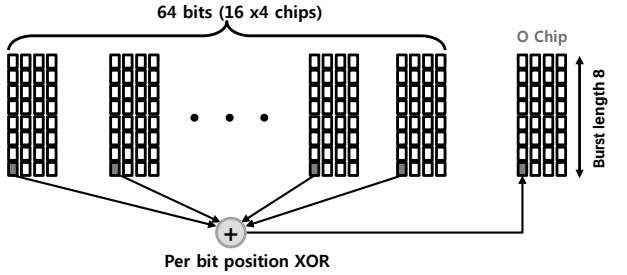


Figure 3: Layout of the outer code. Each bit in the outer-code chip (o) is the bit-wise XOR of the same bit position in all 16 data chips.

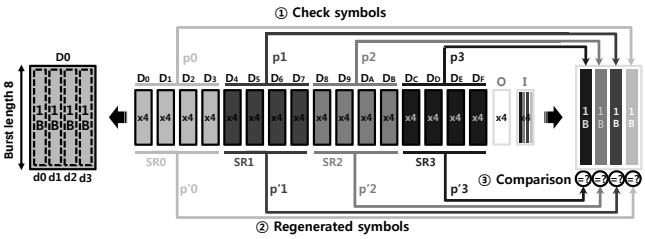


Figure 4: Detection: Check symbols are encoded when writing data into memory as in conventional ECC (①). Check symbols are then recomputed for each accessed subrank when reading data (②) and are compared with the stored symbols (③).

Cache Hierarchy.

Because CLEAN accepts both coarse and fine-grained accesses, the cache hierarchy must be capable of handling both data granularities. CLEAN utilizes a simple sectored cache to manage on-chip management and storage of fine-grained data. Each 64B cache-line is divided into four 16B sectors. Cache coherence is maintained at the granularity of a full cache line; sectors cannot be modified separately by multiple core.

Data Access Granularity Decisions.

Yoon et al. [1, 2] provide a detailed analysis of the different considerations and design options for dynamically deciding the granularity of each DRAM access. We adopt the same hardware-based spatial-pattern predictor design [31, 32] used in prior work [2]. The predictor is configured with 32 sets of 8 ways each. More detailed microarchitectural descriptions of the dynamic predictor can be found in [31, 32, 2].

3.2 Error Correction

The outer code is used primarily for correction. The correction requirement is for errors originating from a single chip. Thus, for each word of the outer (correction) code, at most one chip has an error that requires correction. Using a simple XOR-based RAID code [12, 13], exactly one chip is dedicated for correction. In each full rank and for each coarse-grained access, the O chip holds the per-bit-location bit-wise XOR of the 16 data chips; the outer code consists of 32 parity bits, one parity for each of the 32 bits in a burst, computed across the 16 data chips. This is represented by Equation 1, where $d_{j,k}$ denotes the k-th bit of the j-th data chip D_j and o_k is one of the 32 bits of the outer code, which are stored in 8 consecutive $\times 4$ beats; this organization is shown in part ① of Figure 3. Correction is straightforward as it simply requires computing the XOR of all the data from the 16 non-erring chips (15 data chips and the O chip).

$$o_k = d_{0k} \oplus d_{1k} \oplus \cdots \oplus d_{15k}, k = 0..31 \quad (1)$$

The outer code requires a single coarse-grained read regardless of whether the error was detected on a fine-grained access or not. This access is needed both for correction events, which are rare, and also for maintaining the correction information because it is encoded based on the data from the entire rank. Updating the outer code when writing back only a subset of sectors from the LLC is challenging.

Performing a coarse-grained read before every partial writeback is likely to have a very high memory traffic overhead. Instead, we rely on two important characteristics of the CLEAN architecture. First, the codes used in CLEAN are linear and systematic. With such codes, the updated ECC information (o'') can be computed from only the modified data (d'_{*i}) and the original ECC information (o) by subtracting the original FG data read (d_{*i}) from o. This procedure is defined in Equation 2 and Equation 3.

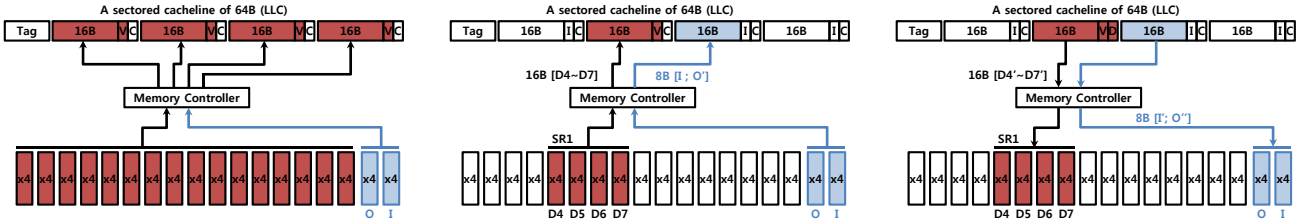
The second characteristic is that a partial writeback implies that there are invalid sectors in the cache block. Thus, to avoid reading the entire rank or even just the ECC information from memory before a partial writeback, we store the original ECC information read when the cache was filled in an empty sector, as shown in Figure 5b. When writing back a partial line, the first invalid sector following the last sector to be written holds the original ECC information, which is then updated as shown in Equation 3 and written back to memory along with the data (Figure 5c). Note that we read and cache, but do not use all the ECC information on a FG access. Hence, data transfer errors that may occur on this unchecked ECC information may go undetected and in such a case a partial writeback may lead to data corruption. We have not seen fault models for transfers and do not evaluate the risk; however, we expect that it is very small.

$$o' = d_{4i} \oplus d_{5i} \oplus d_{6i} \oplus d_{7i} \oplus o_i, \quad (2)$$

$$o'' = d'_{4i} \oplus d'_{5i} \oplus d'_{6i} \oplus d'_{7i} \oplus o'_i, \quad (3)$$

3.3 Error Detection

The goal of CLEAN ECC is to enable independent fine-grained access to multiple subranks. CLEAN must therefore provide an inner codeword for each data subrank within the single remaining ECC chip (chip I). In our configuration, we partition the 32 bits provided in one transfer (4-bits $\times 8$ beats) of the ECC chip used for the inner code into four 8-bit symbols; each symbol is used as the check symbol for one of the four sub-



(a) Coarse-grained DRAM read **(b)** Fine-grained DRAM read **(c)** Fine-grained DRAM write

Figure 5: CLEAN DRAM read/write examples. Compared to CG reads, FG cache fills store the ECC information within an empty sector of the LLC. This eliminates an extra memory access when writing back a partial cache block.

ranks (Figure 4). We propose a simple-to-implement Reed-Solomon-like generalized symbol code for detection, which we describe below; although any strong detection code, (e.g., CRC) will work well.

Each fine-grained access is composed of 16 8-bit data symbols from the four data chips in a subrank, and the single 8-bit symbol from the I chip in the ECC subrank (4 check symbols are read, but only one is associated with the specific FG read). The check symbol is a linear combination (weighted sum) of the data symbols over a Galois Field, as shown in Equation 4. In the equation, p_i denotes the check symbol of the i^{th} subrank, d_{jk} denotes the k^{th} 8-bit symbol of the j^{th} data chip, and a_{jk} are the weights for calculating the check symbol; we chose the weights at random and verified their effectiveness with Monte Carlo simulation.

$$p_i = \sum_{j,k} a_{jk} d_{jk},\ j \in \{4i, \cdots, 4i+3\}, k = 0, \cdots, 3. \quad (4)$$

When a data is read out of the i-th subrank, the formula of Equation 4 is again used to regenerate a check symbol p'_i, which is compared to the stored symbol read from the I chip. Any single corrupted symbol will cause the check symbol to always be different and thus detected. Errors in multiple symbols may lead to an undetected error because the limited domain of the Galois field may result in a random match between the recomputed and stored symbols. With a good choice of weights the probability of such an undetected error is just $\frac{1}{2^n}$, where n is the symbol length (0.39% probability in our code).

To minimize the likelihood that a fault affects multiple symbols, inner-code symbols are aligned with data pins (DQs); any single-bit or single-pin error is guaranteed to be detected, while other multi-bit errors are detected with high probability, which in practice is much better than the pessimistic bound of 0.39%. We evaluate the practical protection in Section 5.2.

Coarse-Grained Detection.

While the error coverage of the inner code is already high, the outer code can be used to further improve detection and curb the risk of SDC. When performing a coarse-grained access, if no error is reported by any of the inner-code checks, we can use the outer-code information to verify the parity of each bit location in the access. If any parities do not match, at least one chip has an error that was not detected by the inner code or the O chip has an error. In other words, if any single chip, other than the O chip has an error, coarse-grained detection is guaranteed to detect the error, no matter how many bits in the data chip are erroneous.

Proof by contradiction. Assume, w.l.o.g., that:
$d'_{0i} \neq d_{0i}$ but is not detected
Undetected error implies:
$\Leftrightarrow o'_i = o_i$
$\Leftrightarrow d'_{0i} \oplus d_{1i} \oplus \cdots \oplus d_{15i} = d_{0i} \oplus d_{1i} \oplus \cdots \oplus d_{15i}$
$\Leftrightarrow d'_{0i} = d_{0i}$
which is a contradiction. □

If the O chip has an error and the inner-code does not detect an error, the implications are that either multiple chips faulted, or that all four independent inner-code checks had an undetected error simultaneously despite having no errors – an extremely low probability $\left(\left(2^{-8}\right)^4\right)$ that does not impact practical coverage.

3.4 Error Localization

While the outer code can effectively recover a corrupted chip by using information from the full rank, the specific chip within the erring subrank that caused the symbol error still needs to be identified; on its own, the inner code only determines which subrank has an error but not the exact faulting chip. We localize the error to a single chip, or detect multiple-chip faults, using a verification step that utilizes the additional ECC information of the outer code and its XOR correction mechanism. Localization is best-effort inference given the restriction of FG access, whereas current *Chipkill* (e.g. [9]) determines the error location by generating and solving the error polynomial based on the syndromes.

The process is to attempt multiple corrections (possibly in parallel), each time under the assumption that a different data chip in the accessed subrank is corrupted (Figure 6). After a correction attempt, the check symbol is regenerated and compared to the stored check symbol of the inner code. If the assumption that only a single data chip is corrupted is true, then only one of the correction attempts will succeed with high probability. If, on the other hand, more than one of the corrections appears to succeed or if no correction attempt succeeds, then more than one chip is corrupted concurrently, one

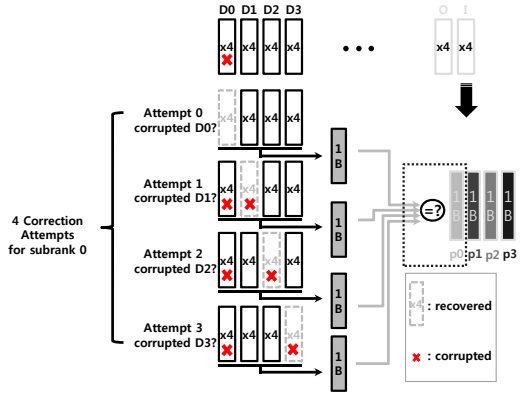


Figure 6: Localization. Suppose that an error in subrank 0 is detected. Four corrections are attempted in parallel with each attempt assuming that a different single chip is corrupted. Because D_0 is corrupted, only the first correction attempt (D_0?) can regenerate the check symbol correctly; because the code is not perfect, other attempts may also regenerate the symbol with very low probability.

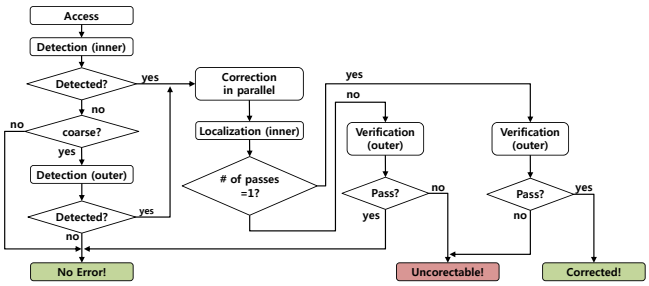


Figure 7: Summary of CLEAN operations.

of the correction attempts resulted in an undetected error when rechecked, or the check symbol has an error. In the first two cases a detected uncorrectable error should be reported. The third case, where the I chip is corrupted, can return corrected data and notify of the detected error in the I chip for diagnostics.

To differentiate between the first two cases from the third, we again utilize the outer code for detection. Specifically, if all correction attempts fail but the outer code detects no error in the original data, we assume the error is in the I chip only because the outer code does not include the I chip in its parity calculation. Otherwise, a detected uncorrectable error is reported. Similarly, we conclude that only O chip is faulty if only the outer code detects errors but none of the inner codes do and correction attempts using the outer code lead to inner-code detected errors (verification failure). With high-probability, we are also able to differentiate the case where only O is faulty from cases in which multiple chips, including the O chip, return erroneous data.

The full detection and correction flow of CLEAN is illustrated in Figure 7. CLEAN first checks for errors in each accessed subrank with its corresponding inner codes. If no errors are detected and if the access was CG, CLEAN double-checks for errors with the outer code. If an error is detected, CLEAN attempts to correct all possibly erroneous chips (chips in subranks indicated by the inner code) using the outer code; each correction attempt is checked with the corresponding inner code and a successful correction increments a counter. After all chips are attempted, the outer code is used to once again verify the result. If the outer code confirms a correct value and only a single correction attempt succeeded, CLEAN concludes that an error was detected and corrected. If no correction attempt succeeded, but the outer code did not detect an error, the most likely conclusion is that the I chip had an error, so no data error occurred. If more than one correction attempt succeeded or if the outer code could not verify the correction, an uncorrectable error is reported.

3.5 SDC-Rate Minimization

Our evaluation (Section 5.2) shows that the rate of undetected errors when using CLEAN is dominated by errors experienced during FG reads from subranks with single-chip faults. In fact, because of the layout of the inner code, only faults that impact multiple DQ pins across multiple bus beats in a single chip during a fine-grained transfer or faults that affect multiple chips in the same transfer can lead to possible SDCs. Furthermore, those faults that affect multiple chips are detected with higher probability with CLEAN than when using the commercial *Chipkill* code of recent AMD processors [9]. Thus, to approach the SDC rate of *Chipkill*, we focus on minimizing the number of potential SDCs caused by single-chip multi-DQ faults.

The raw rate of such single-chip/multi-DQ/multi-beat faults, which includes single-chip/multi-DQ, single-bank/multi-DQ, and single-row/multi-DQ faults, is quite low according to Sridharan et al. [6, 7, 8] and the inner code is still able to detect/correct all but 0.26% of errors resulting from such faults[1]. Hence, the real danger of a high SDC rate is from permanent faults that may lead to numerous errors, and thus a high SDC error rate.

An SDC occurs when the inner code fails to detect an error, which happens at most once out of 2^8 different data/error pairs on average. Thus, multiple SDCs will only occur if unmodified data is read from a faulty location multiple times without either being overwritten or without other reads to different memory blocks that are also impacted by the same fault. In other words, the danger of multiple SDCs is from scenarios that are quite unlikely where software repeatedly performs FG reads from a single memory block with no intervening writes to that location and no accesses to other locations in the same DRAM row, bank, or chip (the faults that may lead to SDCs in the first place). Any read of different data or a different memory location under the same fault has an independent chance of detecting the, thus the probability of an SDC decreases exponentially (a Geometric distribution) with the number of such detection attempts.

[1]The undetected rate of errors from one of the four data chips is $2^{-8} = 0.39\%$, but errors from the I or O chips return correct data with CLEAN and do not cause an SDC.

Even though we did not observe problematic cases in the applications we evaluated, it is possible that repeated SDCs can occur in some software. CLEAN avoids even this unlikely scenario of a possibly unbounded number of errors from a single fault by relying on the highly-effective CG detection, which utilizes the outer code. If multiple errors are detected CG from a rank, CLEAN forces all further accesses to the rank to be CG, avoiding the problematic case of insufficient coverage of the inner code. To ensure multiple errors are detected, even when in the problematic scenario, CLEAN randomly overrides one out of every n FG accesses to be CG. Thus, at most n SDCs are expected from even the most insidious faults and access patterns. In Section 5.1 we show an occasional CG override has negligible overhead.

SDC-minimization approach requires only a single very narrow saturating error counter per rank to identify the threshold of forcing all-CG access. If desired, the counter can be reset on every scrub interval to avoid transient faults forcing CG for long periods of time.

4. EVALUATION METHODOLOGY

We evaluate CLEAN in terms of system performance and efficiency, reliability, and hardware overhead. To estimate the impact on system performance, we use gem5 [33] integrated with DrSim [34], a detailed cycle-accurate DRAM simulator that supports subranked memory systems and fine-grained accesses (Section 3). To quantify reliability, we run Monte Carlo simulations (of 1 billion iterations) using a DRAM fault/error evaluation framework to analyze both error coverage of different fault scenarios and the overall expected rates of faults that can lead to uncorrectable errors and faults that can results in SDCs. To gauge the impact on hardware, we implement CLEAN and *Chipkill* in Verilog and synthesize the designs to estimate their area and delay.

4.1 Performance Evaluation

Workloads.

We evaluate CLEAN using multi-programmed SPEC CPU2006 benchmarks [35] that are known to be memory intensive [36] (Table 3). In the mixes, there is no biased choices for spatial locality or access granularity. To warm up the cache hierarchy, each benchmark mixes are fast-forwarded 10 billion instructions[2] and are then simulated in detail for 300 million instructions; we were not able to set correct and consistent multi-programmed SimPoints (we tried) and opted for fixed-count fast-forwarding instead.

System Configurations.

Our baseline system is configured as an out-of-order 8-core processor with a three-level cache hierarchy. Table 4 summarizes our baseline system configuration, which is simulated using gem5 [33] and DrSim [34].

[2]1 billion fast-forwarding shows practically-identical results

Metrics.

To report the system throughput of multiprogrammed workloads, we use *Weighted Speedup*(WS) as defined by Eyerman et al. [37]: $WS = \sum_{i=0}^{N-1} \frac{IPC_i^{shared}}{IPC_i^{alone}}$. Here N refers to the number of cores, IPC_i^{shared} is the IPC of the i-th program when CMP resources are shared, and IPC_i^{alone} is the IPC of i-th program when running alone on the CMP. We estimated DRAM power consumption based on the power modeling framework published by Micron [38]. Although our study focuses on the reliability of the memory subsystem, the power consumption of the cores is also estimated by using McPAT [39] based on the parameters of a 40-nm ARM-A9.

MIX0	libquantum×2,lbm×2,bwaves×2,GemsFDTD×2
MIX1	bzip2×2,mcf×2,leslie3d×2,soplex×2
MIX2	bwaves×2,GemsFDTD×2,leslie3d×2,soplex×2
MIX3	lbm×2,libquantum×2,bzip2×2,mcf×2
MIX4	leslie3d×2,bzip2×2,bwaves×2,libquantum×2
MIX5	soplex×2,mcf×2,GemsFDTD×2,lbm×2
MIX6	GemsFDTD×2,bzip2×2,lbm×2,leslie3d×2
MIX7	mcf×2,soplex×2,libquantum×2,bwaves×2

Table 3: Simulated SPEC CPU2006 workloads

Processor	4GHz ARM out-of-order, 8 cores
L1 I-caches	32kB private, 64B cache line, 2 cycle latency
L1 D-caches	32kB private, 64B cache line, 2 cycle latency
L2 caches	256kB private, 64B cache line, 6 cycle latency
LLC	4MB shared, 64B cache line, 18 cycle latency
Main memory	16GB, 72-bit wide DDR3-1600 channel, ×4, 8 banks/rank, 4 ranks/channel scheduling - FR-FCFS [40] parameters - Micron DRAM [41]

Table 4: Simulated system parameters.

4.2 Reliability Evaluation

To measure system-level failure probability over time (Figures 11 and 12),we use the two-stage Monte Carlo simulation approach described by Kim et al. [23]. The first stage randomly injects faults into each memory channel[3], using the fault modes and rates described in the literature [6][4] while keeping a history of the injected faults to model fault accumulation, as also described by Roberts et al. [42]. The second stage then invokes a random error pattern generator for each cache FG or CG access, where the error pattern corresponds to the worse-possible average error pattern based on the injected and accumulated faults. To summarize, we inject faults with rates corresponding to those measured on a large-scale system [6] and generate random error patterns from the resulting faults.

Specifically, the per-chip fault modes discussed by Sridharan et al. [6] are mapped into one of four per-

[3]Fault injection is based on error reports that include transmission errors. Note that from ECC perspective, error patterns from either transmission or storage are equivalent and can be detected or corrected within ECC capacity.

[4]We do not use the model for DDR3 presented more recently [7] because the earlier model provides additional detail that we use to determine potentially uncorrectable errors; there is consensus that DRAM fault rates are more strongly related to the number of DRAM devices than to the device generation [6, 7].

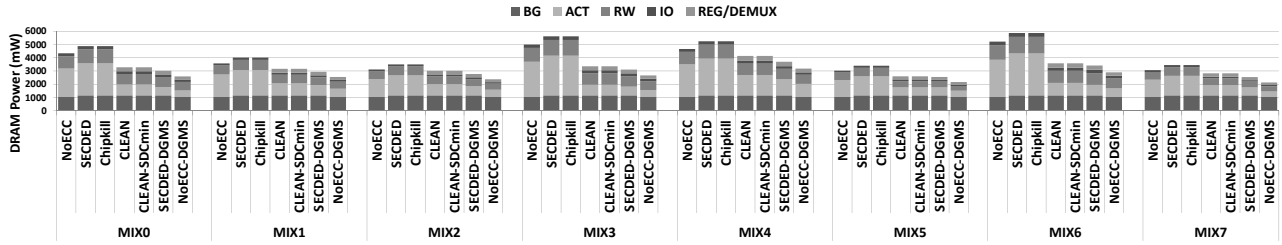


Figure 8: DRAM power

cache-line fault types: bit, word, pin, or chip, based on the number of failing DQs and beats they can cause. A bit fault causes a single bit error at a random position and corresponds to bit faults of [6]. A word fault in a cache line transfer is a result of either word or column faults using Sridharan et al.'s terminology; a column fault may affect multiple DQ pins, but those cannot map onto a single cache-line transfer. A pin fault affects two or more bits, but in the same DQ; pin faults correspond to all single-DQ faults reported in [6]. Finally, chip faults correspond to all faults that affect more than one DQ pin and more than one bus beat in a transfer; chip faults correspond to Sridharan et al.'s single-row, single-bank, multi-bank, and multi-rank faults that also affect multiple DQs.

Once the possible erroneous bits are determined, an error pattern is generated by randomly flipping each potentially erroneous bit with 50% probability. We then evaluate coverage for each error pattern, we test CLEAN and other ECC schemes to check whether the error is a correctable error (CE), detected but uncorrectable (DUE), or possibly a silent data corruption event because it was either not detected or detected but miscorrected to the wrong codeword. After determining error coverage, we combine the fault rates to determine the expected probability that a fault and potential access that lead to a DUE or an SDC may occur in a rank. We report these probabilities over time for systems with different memory capacities. This methodology yields the practical coverage of an ECC technique rather than its, typically exaggerated, worse-case behavior [23].

5. RESULTS AND ANALYSIS

In this section we present the efficiency and performance benefits of CLEAN and compare them to prior work. We do not dwell on these performance and efficiency results because the main contribution of the paper is in improving reliability. We then evaluate and discuss the reliability implications in detail, followed by an analysis of the hardware implementation requirements.

5.1 Efficiency and Performance

The largest benefits from adapting granularity are the increased energy efficiency of the DRAM subsystem because fewer chips are accessed and the performance improvements from better bandwidth utilization. We compare DRAM power consumption, system throughput, and power efficiency of three coarse granularity

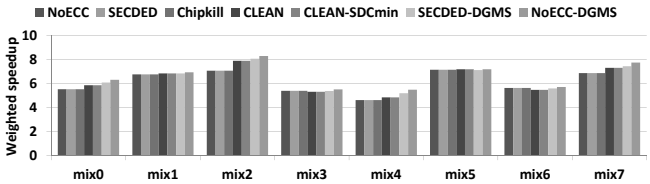


Figure 9: System throughput

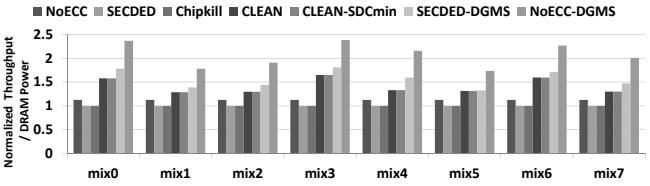
only systems (NoECC, SECDED, and *Chipkill*) and three adaptive granularity systems (CLEAN, DGMS [2], and DGMS-NoECC [2]).

DRAM Power Consumption.

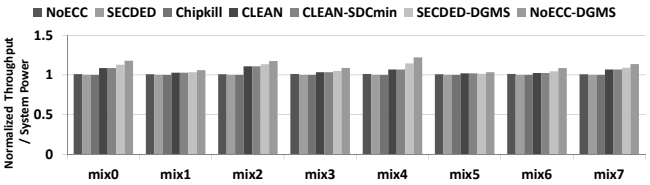
CLEAN achieves up to $21 - 40\%$ reduction (32% on average) in terms of DRAM power consumption, mainly because the fine-grained accesses in benchmarks with low spatial locality reduces both the number of chips that are activated and those that communicate data (Figure 8). Looking into the breakdown of power consumption, the reduction comes from both fewer activations (ACT) and fewer reads and writes (RW). Background power is roughly proportional to the number of chips in a rank and is unchanged. CLEAN provides a more modest reduction than DGMS because CLEAN requires 16b-wide data subranks as opposed to the 8b subranks used by DGMS. The benefit of CLEAN is in its much higher reliability.

System Throughput.

Figure 9 shows that CLEAN generally outperforms the baseline CG-only system, but that the gains in efficiency are more prominent. Four of the seven benchmark mixes we evaluated showed noticeable gains in weighted speedup (up to 11.8% improvement), and three showed little gain or a minor performance loss (-1.5% at worse). On average CLEAN provides a 3.3% improvement in system throughput across the seven benchmark mixes we evaluate. We attribute this minor performance degradation to the different memory scheduling policy required for supporting a mix of FG and CG accesses and again want to emphasize the significant energy efficiency benefits. CLEAN does not support the critical-word first optimization technique, but fine-grained access already provides some of those benefits; while our baseline ECC schemes utilize critical-word first in performance evaluation.



(a) Normalized Throughput / DRAM Power Consumption



(b) Normalized Throughput / System Power Consumption

Figure 10: System / DRAM power efficiency.

CLEAN is not as effective as DGMS and only achieves $39-88\%$ of the performance improvement of DGMS. This is because of two reasons: the larger minimum access granularity and the data layout used for CLEAN compared to DGMS. Because the finest access granularity of CLEAN is 16B compared to 8B with DGMS, the throughput improvement due to the decreased DRAM off-chip traffic is smaller than with DGMS. Moreover, the rotating position of the ECC chips is done with fewer effective "banks", which leads to more subrank conflicts and further diminishes performance improvements over CG-only systems.

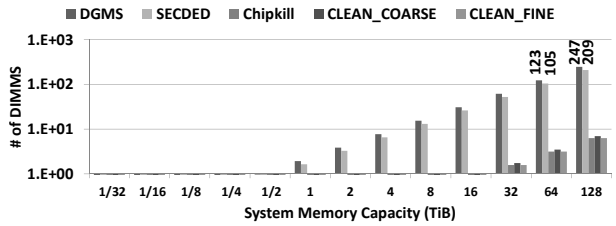
Importantly, the **SDC-minimization scheme** that forces one out of every $10-100$ FG accesses to be CG did not have a significant impact on performance or power; the impact is (less than 1% degradation the numbers reported above *include this overhead*.

System / DRAM Power Efficiency.

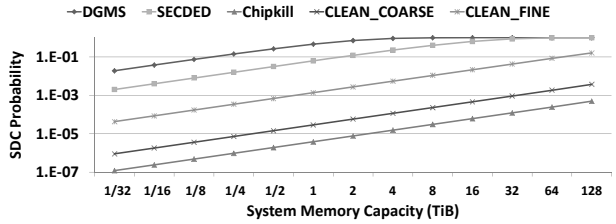
Figure 10a shows DRAM power efficiency, defined as the normalized throughput divided by DRAM power only. Figure 10b shows system power efficiency, defined as the normalized throughput divided by the total system power, including cores, caches, and DRAM as estimated by McPAT. CLEAN improves both system and DRAM power efficiency when compared to *Chipkill* by up to 11% and 65% and 5.5% and 42% on average, respectively. These correspond to 67% and 74% (on average) of the benefits of DGMS. Because the cores consume most of system power as we estimate by McPAT ($42-46$W out of 50W total), system power efficiency shows less improvement than DRAM power efficiency. Note that some studies have reported a much larger proportion of power in the memory subsystem, in which case the benefit of CLEAN will be closer to the improvement shown in Figure 10a.

5.2 Reliability

This section quantitatively shows the high reliability level of CLEAN, which approaches that of a commercial *Chipkill* implementation [9]. We chose this particular commercial implementation because it matches our



(a) Expected number of DIMMs that experience DUEs



(b) SDC probability

Figure 11: System level reliability over 5 years and varying memory capacity (using 16GiB DIMMs): (a) the expected reliability impact of DUEs; and (b) the risk that the system has at least one SDC.

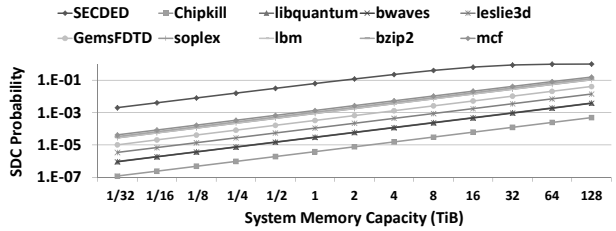


Figure 12: System level reliability over 5 years and varying memory capacity (using 16GiB DIMMs) for individual SPEC 2006 applications (profiled with a Pintool[43]): the risk that the system has at least one SDC.

configuration of 64B cache-line access granularity and its details are clearly described in a published manual. Note that stronger protection is possible on wider interfaces, such as when coupling two DRAM channels [10] but we do not evaluate such protection in this paper. Figure 11a shows the expected number of faults that lead to a DUE over a five-year lifetime of systems with varying memory capacity, and Figure 11b shows the probability that those systems experience at least one SDC event over their five-year lifetime.

We discuss three key takeaway points from our evaluation: (1) CLEAN correction capability is on par with *Chipkill*; (2) when performing CG accesses, CLEAN detection coverage is on par with *Chipkill*; and (3) the detection coverage of CLEAN with FG accesses is acceptable for a large range of systems sizes for which DGMS and SECDED are insufficient. We explain these results by referring to Table 5 in which we break down the fault scenarios; we do not consider the rates at which the different scenarios occur for this analysis.

The correction capability of CLEAN is very close to that of *Chipkill* and is orders of magnitude better

chip	type	result	CLEAN FG	CLEAN CG	*Chipkill*
single	1bit	CE	100.0000%	100.0000%	100.0000%
	1word	CE	100.0000%	100.0000%	100.0000%
	1pin	CE	100.0000%	100.0000%	100.0000%
	1chip	CE	99.7401%	99.6491%	100.0000%
		DUE	0.0000%	0.3509%	0.0000%
		SDC	0.2599%	0.0000%	0.0000%
double	1bit+1bit	CE	1.1344%	0.0000%	0.0000%
		DUE	98.8303%	99.9749%	98.9244%
		SDC	0.0353%	0.0251%	1.0756%
	1bit+1pin	CE	1.1291%	0.0000%	0.0000%
		DUE	98.7982%	99.9440%	99.9407%
		SDC	0.0727%	0.0560%	0.0593%
	1bit+1word	CE	0.8956%	0.0000%	0.0000%
		DUE	99.0614%	99.9833%	98.6447%
		SDC	0.0430%	0.0167%	1.3553%
	1bit+1chip	CE	0.5306%	0.0000%	0.0000%
		DUE	99.2258%	99.9276%	100.0000%
		SDC	0.2436%	0.0724%	0.0000%
	1pin+1word	CE	0.8955%	0.0000%	0.0000%
		DUE	99.0453%	99.9751%	99.9343%
		SDC	0.0592%	0.0249%	0.0657%
	1pin+1pin	CE	1.1353%	0.0000%	0.0000%
		DUE	98.7933%	99.9415%	99.9889%
		SDC	0.0714%	0.0585%	0.0111%
	1pin+1chip	CE	0.5317%	0.0000%	0.0000%
		DUE	99.2230%	99.9208%	100.0000%
		SDC	0.2453%	0.0792%	0.0000%
	1word+1word	CE	0.6749%	0.0000%	0.0000%
		DUE	99.2809%	99.9926%	100.0000%
		SDC	0.0442%	0.0074%	0.0000%
	1word+1chip	CE	0.3265%	0.0000%	0.0000%
		DUE	99.4450%	99.9318%	98.6040%
		SDC	0.2285%	0.0682%	1.3960%
	1chip+1chip	CE	0.0000%	0.0000%	0.0000%
		DUE	99.6118%	99.8789%	100.0000%
		SDC	0.3882%	0.1211%	0.0000%

Table 5: Breakdown of ECC results for realistic fault types (10^9/type random errors). CE, DUE and SDC stand for corrected fault, detected but uncorrectable fault and silent data corruption.

	Chipkill [9]	$CLEAN$	$CLEAN_{par}$
Area(μm^2)	2527.65	1237.10	4948.29
Delay(ns)	4.84	2.52	0.63

Table 6: Synthesis results for overhead estimation.

than that of DGMS. With CLEAN, some single-chip faults that occur in the O chip that holds the outer-code ECC information can lead to errors that are reported as uncorrectable, even when the errors are confined to the O chip and are therefore correctable as the data is not erroneous. This is evident in the single-chip fault mode in Table 5 and is discussed in Section 3.3 and Section 3.4. The practical absolute difference in correction capability is just about 0.6, which is 0.01% of the entire system in the largest system we evaluate.

The detection coverage of CLEAN for CG accesses is on par with that of *Chipkill*; About 10% more DUEs resulted from single chip faults (Figure 11a), only 0.35% of which are DUEs with CG CLEAN(Table 5). Similarly, we see that the expected DUE rate for FG CLEAN and *Chipkill* is almost the same (Figure 11a), however, FG CLEAN has a higher SDC rate. Even in the largest system we evaluate, the SDC risk of CG CLEAN is only 0.3% larger than that of the baseline *Chipkill*. This small difference results from the small fraction of two-chip faults that can be detected by the longer codeword used by AMD *Chipkill*, especially when including its history mechanism [9, 23], but which are missed by the shorter concatenated code of CLEAN. This can be seen by observing the small SDC probabilities of the double-chip fault cases in Table 5.

FG CLEAN suffers from a higher SDC risk than *Chipkill* as discussed in Section 3.5. This higher SDC rate is almost entirely the result of single-chip faults because these suffer from both having the highest probability going undetected by the inner code (0.26%) and are more frequent than double-chip faults. This manifests in a significant degradation of reliability with respect to SDCs compared to CG CLEAN and *Chipkill* . However, FG CLEAN is still sufficiently strong for large systems with memory capacities of many TiB (Figure 11b) and is orders of magnitude better than SECDED and DGMS. In addition, some smaller systems that utilize ECC today can benefit from the performance and efficiency improvements of CLEAN-ECC without effective impact on reliability.

Because there is a gap between the SDC risk of FG and CG accesses, each application has a different SDC rate; a larger fraction of FG accesses results in greater SDC risk. Figure 12 shows that the estimated SDC risk of the memory bandwidth intensive SPEC CPU2006 application based on their ratio of FG and CG accesses (profiled with a Pintool [43]). Applications with high spatial locality have low SDC whereas the SDC probability of applications that have lower spatial locality is dominated by the lower reliability of the inner code. Applications that are not bandwidth intensive use CLEAN-CG only.

5.3 Hardware Overheads

While the storage overhead of CLEAN equals that of *Chipkill*, the hardware implementation of the encoders and decoders is different. We estimate the expected hardware requirements for CLEAN and compare them to those of *Chipkill* by implementing its encoder in Verilog and synthesizing and evaluating our designs with the Cadence Design Vision (DVE) targeting the *FreePDK45-gscl45nm* [44].

We implemented an aggressively parallelized *Chipkill* encoder that is hard-coded for a specific polynomial to reduce latency. This leads to a larger area than required for the well-known shift-register based encoder in which input symbols are sequentially computed, but results in far better latency which we believe is a better design point for the main memory system. The design of both the inner and outer encoders and decoders of CLEAN is simple because they inherently rely on straightforward parity operations. We also note that a single layer of muxes and wires is needed for aligning the 32 inner and 32 outer-code parity bits with the encoders and decoders because of the modulo-9 RAID-like data layout described in Section 3.1; the data symbols are all symmetric and don't require long shifts. We implemented two versions of CLEAN encoder: one that operates on all four subranks in parallel ($CLEAN_{par}$) and a lower-area version that operates serially ($CLEAN$). Table 6 summarizes our synthesis results and shows that CLEAN provides lower latency decoding at a comparable area to *Chipkill*'s encoder, which is the simplest part in *Chipkill*. If correction is required, the CLEAN localization is equivalent to four encoding operations.

6. CONCLUSIONS

In this paper, we presented, for the first time, a highly-reliable memory protection scheme that is able to perform fine-grained subrank accesses in a dynamic fashion that is completely transparent to software. We conclude that prior work on adaptive granularity does not provide sufficiently strong memory protection because the data layout and codes it uses [1, 2] cannot account for chip faults or even single-pin faults. In contrast, CLEAN can approach the reliability of *Chipkill* when performing coarse-grained accesses, provides similar correction capability even with fine-grained accesses, and manages the rate of possible silent data corruption events. This is all achieved while maintaining the customary 12.5% storageoverhead, using standard DDR3 DRAM chips, and with no additional on-chip components compared to prior adaptive-granularity architectures.

7. ACKNOWLEDGEMENTS

The authors acknowledgement the support of the National Science Foundation under Grant #0954107, which partially funded this reasearch.

8. REFERENCES

[1] D. H. Yoon, M. K. Jeong, and M. Erez, "Adaptive granularity memory systems: A tradeoff between storage efficiency and throughput," in *Proc. of ISCA*, 2011.

[2] D. H. Yoon, M. K. Jeong, M. B. Sullivan, and M. Erez, "The dynamic granularity memory system," in *Proc. of ISCA*, 2012.

[3] M. Rhu, M. Sullivan, J. Leng, and M. Erez, "A locality-aware memory hierarchy for energy-efficient gpu architectures," in *Proc. of MICRO*, 2013.

[4] B. Schroeder, E. Pinheiro, and W.-D. Weber, "DRAM Errors in the Wild: a Large-Scale Field Study," in *Proc. of the International Joint Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, 2009.

[5] A. A. Hwang, I. A. Stefanovici, and B. Schroeder, "Cosmic Rays Don't Strike Twice: Understanding the Nature of DRAM Errors and the Implications for System Design," in *Proc. of ASPLOS*, 2012.

[6] V. Sridharan and D. Liberty, "A study of dram failures in the field," in *Proc. of SC*, pp. 76:1–76:11, 2012.

[7] 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. of SC*, pp. 22:1–22:11, 2013.

[8] V. Sridharan, N. DeBardeleben, S. Blanchard, K. B. Ferreira, J. Stearley, J. Shalf, and S. Gurumurthi, "Memory errors in modern systems: The good, the bad, and the ugly," in *Proc. of ASPLOS*, 2015.

[9] "BIOS and Kernel Developer's Guide (BKDG) for AMD Family 15h Models 00h-0Fh Processors," Jan 2013.

[10] "Intel Xeon Processor E7 Family: Reliability, Availability, and Serviceability," 2011.

[11] T. J. Dell, "A white paper on the benefits of chipkill-correct ecc for pc server main memory," 1997.

[12] IBM, *IBM 3330 data storage*.

[13] D. A. Patterson, G. Gibson, and R. H. Katz, "A case for redundant arrays of inexpensive disks (raid)," in *Proc. of the ACM International Conference on Management of Data(SIGMOD)*, pp. 109–116, 1988.

[14] J. S. Liptay, "Structural aspects of the system/360 model 85: Ii the cache," *IBM Syst. J.*, vol. 7, no. 1, pp. 15–21, 1968.

[15] J. B. Rothman and A. J. Smith, "The pool of subsectors cache design," in *Proc. of the 13th International Conference on Supercomputing(ICS)*, pp. 31–42, 1999.

[16] J. H. Ahn, N. P. Jouppi, C. Kozyrakis, J. Leverich, and R. S. Schreiber, "Future scaling of processor-memmory interfaces," in *Proc. of SC*, Nov. 2009.

[17] J. H. Ahn, J. Leverich, R. Schreiber, and N. P. Jouppi, "Multicore DIMM: An energy efficient memory module with independently controlled DRAMs," *IEEE Computer Architecture Letters*, vol. 8, pp. 5–8, Jan. - Jun. 2009.

[18] F. A. Ware and C. Hampel, "Improving power and data efficiency with threaded memory modules," in *Proceedings of the International Conference on Computer Design (ICCD)*, 2006.

[19] H. Zheng, J. Lin, Z. Zhang, E. Gorbatov, H. David, and Z. Zhu, "Mini-rank: Adaptive DRAM architecture for improving memory power efficiency," in *Proc. of MICRO*, Nov. 2008.

[20] T. M. Brewer, "Instruction set innovations for the Convey HC-1 computer," *IEEE Micro*, vol. 30, no. 2, pp. 70–79, 2010.

[21] X. Jian and R. Kumar, "Adaptive reliability chipkill correct (arcc)," vol. 0, pp. 270–281, 2013.

[22] S. Li, D. H. Yoon, K. Chen, J. Zhao, J. H. Ahn, J. B. Brockman, Y. Xie, and N. P. Jouppi, "Mage: Adaptive granularity and ecc for resilient and power efficient memory systems," in *Proc. of SC*, 2012.

[23] J. Kim, M. Sullivan, and M. Erez, "Bamboo ecc: Strong, safe, and flexible codes for reliable computer memory," in *Proc. of HPCA*, Feb. 2015.

[24] X. Jian, H. Duwe, J. Sartori, V. Sridharan, and R. Kumar, "Low-power, low-storage-overhead chipkill correct via multi-line error correction," in *Proc. of SC*, pp. 24:1–24:12, 2013.

[25] D. H. Yoon and M. Erez, "Virtualized and flexible ecc for main memory," in *Proc. of ASPLOS*, pp. 397–408, ACM, 2010.

[26] A. N. Udipi, N. Muralimanohar, R. Balsubramonian, A. Davis, and N. P. Jouppi, "LOT-ECC: localized and tiered reliability mechanisms for commodity memory systems," *SIGARCH Comput. Archit. News*, vol. 40, pp. 285–296, June 2012.

[27] X. Jian and R. Kumar, "Ecc parity: A technique for efficient memory error resilience for multi-channel memory systems," in *Proc. of SC*, 2014.

[28] A. N. Udipi, N. Muralimanohar, N. Chatterjee, R. Balasubramonian, A. Davis, and N. P. Jouppi, "Rethinking dram design and organization for energy-constrained multi-cores," *SIGARCH Comput. Archit. News*, vol. 38, pp. 175–186, June 2010.

[29] R. Blankenship, D. Brzezinski, and E. Valverde, "Memory error detection and/or correction," Aug. 21 2012. US Patent 8,250,435.

[30] Z. Wang, G. A. Jullien, and W. C. Miller, "An efficient tree architecture for modulo 2 n + 1 multiplication," *Journal of VLSI Signal Processing*, pp. 241–248, Dec. 1996.

[31] S. Kumar and C. Wilkerson, "Exploiting spatial locality in data caches using spatial footprints," in *Proc. of ISCA*, pp. 357–368, 1998.

[32] C. F. Chen, S.-H. Yang, B. Falsafi, and A. Moshovos, "Accurate and complexity-effective spatial pattern prediction," in *Proc. of HPCA*, 2004.

[33] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, pp. 1–7, Aug. 2011.

[34] M. K. Jeong, D. H. Yoon, and M. Erez, "Drsim: A platform for flexible DRAM system research." `http://lph.ece.utexas.edu/public/DrSim`.

[35] Standard Performance Evaluation Corporation, "SPEC CPU 2006." `http://www.spec.org/cpu2006/`, 2006.

[36] M. K. Jeong, D. H. Yoon, D. Sunwoo, M. Sullivan, I. Lee, and M. Erez, "Balancing dram locality and parallelism in shared memory cmp systems," in *Proc. of HPCA*, pp. 1 –12, feb. 2012.

[37] S.Eyerman and L.Eeckout, "System-level performance metrics for multiprogram workoads," vol. 28, no. 3, pp. 42–53, 2008.

[38] M. Technology, "Calculating memory system power for ddr3," *Technical Report TN-41-01*, 2007.

[39] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "Mcpat: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Proc. of MICRO*, pp. 469–480, 2009.

[40] S. Rixner, W. J. Dally, U. J. Kapasi, P. Mattson, and J. D. Owens, "Memory access scheduling," in *Proc. of ISCA*, pp. 128–138, 2000.

[41] Micron Corp., *Micron 1 Gb ×4, ×8, ×16, DDR3 SDRAM: MT41J256M4, MT41J128M8, and MT41J64M16*, 2006.

[42] D. Roberts and P. Nair, "FAULTSIM: A fast, configurable memory-resilience simulator," in *The Memory Forum: In conjunction with ISCA*, vol. 41.

[43] C.-K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. J. Reddi, and K. Hazelwood, "PIN: Building customized program analysis tools with dynamic instrumentation," in *Proc. the ACM Conf. Programming Language Design and Implementation (PLDI)*, Jun. 2005.

[44] "FreePDK45." `http://www.eda.ncsu.edu/wiki/FreePDK45:Contents`, 2006.