The Efficacy of Error Mitigation Techniques for DRAM Retention Failures: A Comparative Experimental Study

Samira Khan[†⋆]
samirakhan@cmu.edu

Donghyuk Lee[†]
donghyuk1@cmu.edu

Yoongu Kim[†]
yoongukim@cmu.edu

Alaa R. Alameldeen[⋆]
alaa.r.alameldeen@intel.com

Chris Wilkerson[⋆]
chris.wilkerson@intel.com

Onur Mutlu[†]
onur@cmu.edu

[†]Carnegie Mellon University [⋆]Intel Labs

ABSTRACT

As DRAM cells continue to shrink, they become more susceptible to retention failures. DRAM cells that permanently exhibit short retention times are fairly easy to identify and repair through the use of memory tests and row and column redundancy. However, the retention time of many cells may vary over time due to a property called *Variable Retention Time (VRT)*. Since these cells intermittently transition between failing and non-failing states, they are particularly difficult to identify through memory tests alone. In addition, the high temperature packaging process may aggravate this problem as the susceptibility of cells to VRT increases after the assembly of DRAM chips. A promising alternative to manufacture-time testing is to detect and mitigate retention failures after the system has become operational. Such a system would require mechanisms to detect and mitigate retention failures in the field, but would be responsive to retention failures introduced after system assembly and could dramatically reduce the cost of testing, enabling much longer tests than are practical with manufacturer testing equipment.

In this paper, we analyze the efficacy of three common error mitigation techniques (memory tests, guardbands, and error correcting codes (ECC)) in real DRAM chips exhibiting both intermittent and permanent retention failures. Our analysis allows us to quantify the efficacy of recent system-level error mitigation mechanisms that build upon these techniques. We revisit prior works in the context of the experimental data we present, showing that our measured results significantly impact these works' conclusions. We find that mitigation techniques that rely on run-time testing alone [38, 27, 50, 26] are unable to ensure reliable operation even after many months of testing. Techniques that incorporate ECC [4, 52], however, can ensure reliable DRAM operation after only a few hours of testing. For example, VS-ECC [4], which couples testing with variable strength codes to allocate the strongest codes to the most error-prone memory regions, can ensure reliable operation for 10 years after only 19 minutes of testing. We conclude that the viability of these mitigation techniques depend on efficient online profiling of DRAM performed without disrupting system operation.

SIGMETRICS'14, June 16–20, 2014, Austin, Texas, USA.

http://dx.doi.org/10.1145/2591971.2592000.

Categories and Subject Descriptors

B.3.1 [**Memory Structure**]: Semiconductor Memories—*Dynamic memory (DRAM)*; B.3.4 [**Memory Structure**]: Reliability, Testing, and Fault-Tolerance

Keywords

DRAM, retention failures, system-level detection and mitigation, error correction, ECC, fault tolerance, memory scaling

1. INTRODUCTION

The increasing number of cores and extensive use of data-intensive applications demand high capacity main memories. Scaling of DRAM to smaller technology nodes enabled higher capacity in the same die area for the past few decades. However, as DRAM cells scale down in size, they become significantly less reliable. Ensuring reliability with technology scaling is a key challenge for DRAM [36, 31, 18, 34].

A DRAM cell cannot retain its data permanently as the capacitor used to store the data leaks its charge gradually over time. The time a cell can retain its data is called the *retention time* of the cell. In order to maintain correct data in DRAM, cells are periodically refreshed every 64 ms. A cell that cannot retain its data for 64 ms results in a failure, referred to as *retention failure*.[1] In order to avoid retention failures, the worst-case retention time of cells must be no less than 64 ms. However, several technology trends make this requirement increasingly difficult to efficiently satisfy. On the one hand, as cell dimensions shrink, cells become more susceptible to retention failures (or, reducing retention times) [10, 13, 11, 54], motivating higher refresh rates. On the other hand, increasing DRAM chip capacity makes even the current nominal refresh rate untenable in future technology DRAMs. For example, one study showed that a 32 Gb DRAM chip will spend 35% of the memory power and 25% of the available DRAM chip bandwidth on refreshes [27]. Another study showed that the performance penalty of refresh in a system with 32 Gb DRAM chips can be 20% [6]. A higher refresh rate will result in a significant throughput loss, power overhead, and performance loss [27, 6, 37].

Currently, DRAM vendors guarantee worst-case retention time of 64 ms by detecting and mitigating cells having retention time less than 64 ms during manufacturing time. However, detection of retention failures is extremely challenging as the same cell can fail or operate correctly at different times. These retention failures are defined as *intermittent retention failures* in DRAM cells.

[1] We assume each failure can potentially cause an error and use failure and error interchangeably in the rest of the paper.

There are two sources of intermittent retention failures in DRAM: 1) *Data pattern sensitivity*: A DRAM cell can fail depending on the data stored in the neighboring cells [23, 28], and 2) *Variable retention time (VRT)*: A cell can transition among different retention times at different points in time and can fail when it moves to a retention time less than the current refresh interval [41, 55, 33, 28]. Due to these intermittent retention failures, manufacturers perform exhaustive testing to detect and mitigate these failures. A recent work showed that it can take days to test a chip to identify all intermittent retention failures [28]. In fact, one of the major DRAM vendors confirmed that detection of such failures leads to some of the most expensive tests during manufacturing time. As retention failures become more prominent with technology scaling [10, 13, 11, 54, 28], the manufacturing-time retention failure tests will become longer, making it more difficult and more costly for the DRAM manufacturers to scale the density of DRAM chips as aggressively as was done in the past. More importantly, some of the DRAM cells start exhibiting VRT characteristics as they are exposed to high temperature during the packaging of the DRAM chips [44, 21, 28], so pre-packaging tests cannot always screen all VRT failures.

Instead of DRAM manufacturers providing retention-error-free DRAM chips after exhaustive and expensive manufacturing-time tests, an alternative approach is detecting and mitigating retention failures of DRAM cells in the field, during the operation of DRAM in the system. We refer to system-level error detection as *online profiling*. In this case, the memory controller would be responsible for detecting and mitigating the retention failures while the system is running. An online profiling mechanism has three significant advantages over traditional manufacturing time testing. First, such a system can reduce manufacturing cost and increase yield by enabling the manufacturers to ship DRAM chips without fully ensuring that all cells operate correctly at a minimum specified retention time. Second, online testing is capable of tolerating the new intermittent failures induced by the high temperature packaging process. Third, an online profiling system is more suitable compared to manufacturing tests to detect the intermittent retention failures that takes hours/days of testing, as it can amortize the cost of profiling by performing online tests in the background spread across a very large time interval (days/months).

System-level detection and mitigation of intermittent retention failures seems to be a promising direction into the future. However, there is no prior work that assesses the efficacy of different mitigation techniques in the presence of intermittent failures. Some recent works proposed error mitigation techniques that rely on system-level testing, but assume there are no intermittent failures. These mechanisms perform a simple test at system boot-up to detect the failing cells and cannot tolerate any new failures after the initial testing [38, 27, 50, 26]. If error mitigation techniques do not consider intermittent failures, data loss can occur due to potentially inaccurate identification of failing cells.

Our goal is to analyze the efficacy of existing error mitigation techniques in the system in the presence of intermittent retention failures to enable the development of more efficient and effective detection and mitigation techniques. We use an FPGA-based DRAM testing infrastructure to test 96 DRAM chips from three different manufacturers to analyze the effectiveness of system-level testing, guardbanding, and ECC for intermittent retention failures. We use our analysis to quantify the effectiveness of recently proposed architectural error mitigation techniques [38, 27, 50, 4, 52, 26] in the presence of intermittent failures.

To our knowledge, this paper presents the first work that quantifies system-level detection and mitigation techniques in the presence of intermittent failures. Our major contributions are:

- We analyze the effectiveness of testing in discovering retention failures. We show that only a small amount of testing can discover the majority of failing cells. We quantify testing using the metric *rounds*, where a round consists of writing the entire module, waiting for some specific time to make sure all the cells have been idle for the refresh interval and reading back the data again to locate the failing cells. From our experiments, we show that only 5 rounds of tests can discover most of the intermittent failures and reduce the probability of discovering a new failure by 100 times. However, even after thousands of rounds of testing, we find that a very small number of cells exhibit new failures not discovered before, which indicates that testing is not enough by itself to mitigate all retention failures.
- We present the efficacy of using a guardband to avoid intermittent retention failures. Adding a guardband X on the refresh interval of 64 ms means that all DRAM chips that are detected to have cell failures at X times 64 ms (i.e., $64X$ ms) refresh interval are discarded by the manufacturer [51, 2]. If VRT cells exhibit retention times less than the applied guardband, these cells will fail and get detected at $64X$ ms, making the guardband 100% effective. We show that a small amount of guardband (e.g., $2X$) can avoid 85-95% of the intermittently failing cells and reduce the probability of a new retention failure by 10 times. However, even a large guardband (e.g., $5X$) is not effective for the remaining VRT cells, indicating that using a guardband alone is not effective to mitigate all failures.
- We show the efficacy of error correction codes (ECC) in the presence of other error mitigation techniques. We find that a combination of error mitigation techniques is much more effective in tolerating a high number of retention errors than using only ECC. An uncorrectable retention error in the presence of ECC results in a failure. Using only single error correction codes can reduce the probability of retention failure by only 100 times, but using single error correction codes together with testing and guardbanding can reduce the probability of failure by as much as 10^{12} times.
- We revisit prior works [38, 27, 50, 4, 52, 26] in the context of the experimental data and analysis, showing that our measured results significantly impact these works' conclusions. We show that bit repair mechanisms that rely on testing cannot provide strong reliability guarantees even after performing system-level testing for months. However, ECC-based mitigation techniques can be effective at ensuring reliable DRAM operation with online testing in the presence of intermittent failures after just a few hours of testing. We conclude that the viability of these techniques depend on efficient continuous online profiling performed periodically at a longer interval without disrupting system operation.

Based on our experimental evaluation, we discuss a possible online profiling system to detect and mitigate intermittent retention failures. We hope that our findings will be useful in developing new low-cost architectural techniques to mitigate DRAM retention failures, thereby enabling better scaling of DRAM into future technology generations.

2. BACKGROUND AND RELATED WORK

In this section, we briefly provide background information necessary to understand our technical contributions. Interested readers can find more details on DRAM in [22, 20].

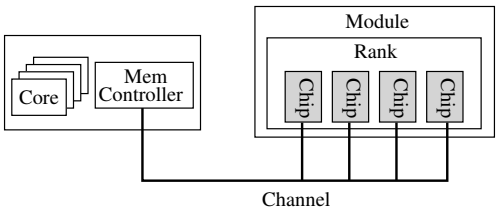


Figure 1: DRAM Organization

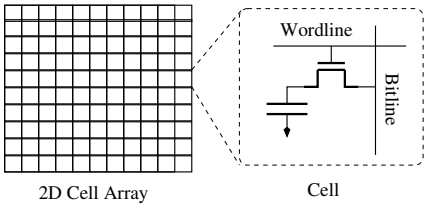


Figure 2: DRAM Cells in a Bank

2.1 DRAM Basics

Figure 1 shows the high level DRAM organization. DRAM is hierarchically organized as channels, modules, ranks, and chips. Each channel consists of multiple modules. On every module there can be more than one rank that consists of multiple DRAM chips. For example, a typical 2 GB module with 64-wide data bus can have one rank with 8 chips. Each chip can have 2 Gb capacity with an 8-bit bus. All chips in a rank respond to the same DRAM command. With this configuration, 8 chips can transfer 64 bits of data in parallel. DRAM cells are organized as 2D arrays called banks. Figure 2 shows the structure of a bank and a cell within the bank. A cell consists of a capacitor and a transistor. The cells in a row are connected through a wire (wordline) and the cells in a column are connected to another wire (bitline). In order to access the data stored in a cell, a high voltage is applied to the wordline of the row containing the cell, which turns on the access transistor and connects the capacitor to the bitline. When the access transistor is on, charge can flow to/from the capacitor through the bitline and data can be accessed.

Charge stored in the capacitor leaks over time. The time a cell can retain its data is called the retention time of the cell. To retain the data in the cell, DRAM cells have to be periodically refreshed. The DDR3 specification guarantees the lowest retention time of 64 ms [15], which means that each row in DRAM is refreshed every 64 ms to maintain the data in the cells.

2.2 Retention Failures

DRAM cells that cannot hold data for 64 ms cause retention failures. Retention time is a key constraint to enabling high density DRAM. As the cell size shrinks with technology scaling, the manufacturers are facing increasing challenges to fabricate a low leakage cell with adequate capacitance to retain the data for 64 ms [31, 18, 34]. DRAM yield is limited by a small fraction of leaky cells whose retention time is significantly lower than other cells [19, 11]. Detecting retention failures (i.e., cells that cannot maintain data as long as the minimum specified retention time) requires exhaustive testing as these failures can occur intermittently. Next, we provide two prevalent characteristics of the retention failures.

2.2.1 *Data Pattern Sensitivity*

The retention time of a cell is strongly dependent on the charge stored in that cell, as well as in the neighboring cells, a phenomenon referred to as *data pattern sensitivity* [23, 28]. A data pattern sensitive retention failure is hard to detect as each cell in the DRAM chip has to be exhaustively tested with different kinds of patterns stored in the cell itself as well as the neighboring cells.

2.2.2 *Variable Retention Time*

The retention time of a cell also depends on the current state of the cell. Some DRAM cells can randomly transition between different retention states. These cells are said to demonstrate *variable retention time* (VRT) [41, 55, 33, 28] and are called VRT cells. Depending on the current retention time, a cell may or may not fail at that particular time. In order to guarantee that all cells have retention time greater than 64 ms, every VRT cell with a potential retention time less than 64 ms must get discovered during testing. Manufacturing tests can detect VRT cells if they are actually in the failing retention state during the tests. Unfortunately, the existence of VRT cells cause two major challenges in deploying retention-failure-free DRAM chips in the field. First, discovering whether or not VRT cells have a failing retention state may require prohibitively long tests, as some of the cells can spend a long time in the high retention state before moving to a low retention state [28, 17, 55]. Second, thermal stress during packaging of DRAM chips induces new VRT cells [44, 21]. As such, tests done by DRAM manufacturers before packaging cannot discover these failures.

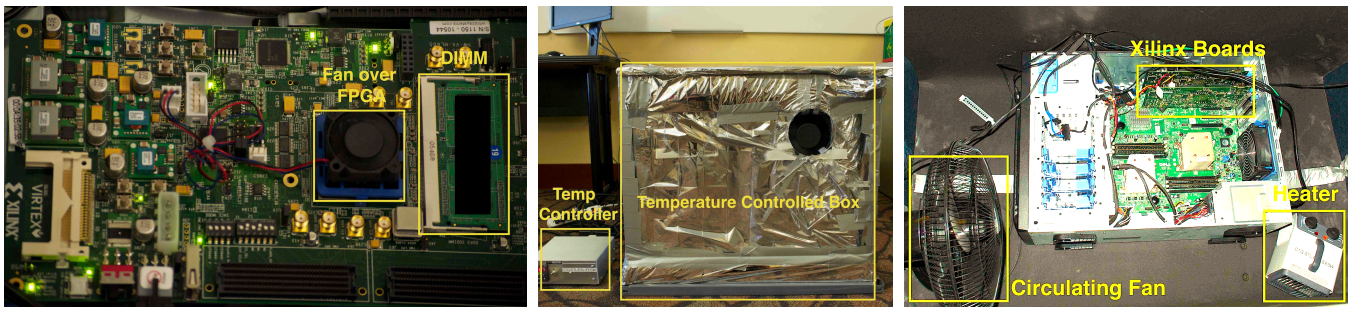
2.3 Mitigating Retention Failures

2.3.1 *Testing*

Conventional DRAM retention tests verify that all cells retain their charge for the specified refresh interval (64 ms). The phenomena of data pattern sensitivity and variable retention states make retention testing a challenging problem. Several test algorithms with different patterns need to be applied to achieve a reasonable fault coverage for pattern sensitivity. On the other hand, VRT cells are hard to detect even with very long tests. Previous works have described the effectiveness of different test patterns and algorithms for manufacturing-time testing [49, 28, 3]. However, no previous work shows the efficacy of testing for retention failures at the system level in the presence of both pattern sensitivity and VRT cells.

2.3.2 *Guardbanding*

A guardband on the refresh interval is defined as adding an extra margin (X) on the refresh interval of 64 ms such that all cells have a retention time greater than X times 64 ms (i.e., $64Xms$) [51, 2]. The process of *guardbanding* (adding a guardband X) consists of three steps. First, a screening test is run at a higher refresh interval ($64X$ ms) to detect all the cells with retention time less than $64X$ ms. This includes VRT cells with retention times less than the added guardband. Second, either all the failing bits that are found during the screening test get repaired through some repair mechanism, such as row and column sparing, or the chip exhibiting the bit failures gets discarded, making sure that the chips passing the tests will operate correctly at 64 ms. Finally, the original refresh interval of 64 ms is reintroduced. Ideally, with a large enough guardband, one would hope that all causes of potential intermittent retention failures can be avoided at manufacturing time. However, there are two disadvantages of using a large guardband. First, a large number of chips would get discarded since the number of retention failures increases exponentially as refresh interval increases [28, 11, 14], resulting in a significant yield loss. Second, a guardband applied during manufacturing time may not be enough to avoid the new VRT failures induced during the packaging process. To our knowledge, no prior work shows the effectiveness of guardbanding for retention failures in the presence of VRT cells.



(a) (b) (c)

Figure 3: Testing Infrastructure: (a) ML605 Board (b) Temperature Controller and Heat Chamber, and (c) Inside of the Heat Chamber

2.3.3 *Error Correction Codes*

Error correction codes and parity are used to repair faulty bits using redundant information. Prior works proposed to use ECC to tolerate retention errors [8, 52]. There are some studies on the bit error rate (number of bit errors divided by the total number of bits during a specific time interval) of industrial DIMMs in the field [43, 24, 46, 45]. However, no prior work provided an experimental study and analysis of the retention error rate reduction with ECC in presence of other error mitigation techniques for intermittent retention failures.

2.3.4 *Recent Error Mitigation Techniques*

Recent system-level error detection and mitigation techniques can be divided into two categories:

1. Bit Repair Mechanisms: These mechanisms rely on tests at system boot time to locate failing bits and then use different repair mechanisms to repair those faulty bits. For example, ArchShield [38] remaps the faulty bits in a specified region of memory. RAIDR [27] uses higher refresh rates for rows containing weak cells to avoid retention failures. RAPID [50] uses software remapping to disable pages with potential retention failures. SECRET [26] applies error correcting pointers [42] only to the faulty bits found during the initial testing. However, all these works assume that a simple initial test can detect all the failures and do not consider the intermittent failures caused by data pattern sensitivity and VRT.

2. ECC-Based Mitigation Techniques: These mechanisms rely on strong ECC and apply such codes in ways to minimize the overall cost of ECC. We discuss two examples. VS-ECC [4] is an ECC-based mitigation mechanism that uses online testing to determine the needed ECC strength for different cache lines. Any cache line with one or more potential errors is protected by a strong ECC, a 4-bit correcting, 5-bit detecting code (4EC5ED). All other lines are protected using simple single error correcting, double error detecting (SECDED) codes, which reduces the overall cost of ECC by applying strong codes to only those lines that need strong codes. However, VS-ECC does not consider intermittent failures and as a result can fail in the presence of a 2-bit failure in lines protected only by SECDED code. Hi-ECC [52] proposes to amortize the cost of strong ECC by protecting data at a coarse granularity (1KB instead of 64B). This mechanism can potentially tolerate intermittent failures with the strong ECC code applied uniformly. However, it has a significant bandwidth and performance overhead, as the memory controller reads the entire 1KB data chunk (as opposed to the much smaller 64B cache line) upon each access to verify ECC.

2.4 Our Goal and Scope

Our goal in this work is to analyze the system-level efficacy of existing mitigation techniques for retention failures using experimental data from commodity DRAM chips. We evaluate recently proposed mitigation techniques in the context of intermittent retention failures. We do not take into consideration the effect of other functional failure mechanisms (e.g., stuck-at faults, decoder faults, etc. [48, 1, 39]) as these mechanisms lead to consistent and repeatable failures which can be detected relatively easily during manufacturing tests. We also do not consider alpha particle or cosmic ray induced soft failures [40, 32] as previous works have already provided strong analyses for these [5, 12, 47, 35], which can be used in conjunction with the findings of our paper.

3. TESTING INFRASTRUCTURE

To study the efficacy of system-level detection and mitigation techniques for intermittent retention failures, we would like to experimentally analyze how retention time of cells change at different times with different data patterns. To do these studies, we have devised an FPGA-based experimental infrastructure that consists of Xilinx ML605 boards (Figure 3a) [53]. An ML605 board has a slot for a *small outline dual in-line memory module* (SO-DIMM) and integrates a DDR3 interface. The host PC communicates with the boards through a PCI Express bus. We have customized the memory interface such that the refresh interval can be changed from the specific software in the host PC. DRAM manufacturers ensure that there are no failures at the nominal refresh interval of 64 ms by discarding or repairing chips with failures during manufacturing tests. As a result, DRAM chips in the field do not exhibit any retention failures at the refresh interval of 64 ms. In order to expose the retention failures in commodity DRAM chips, we increase the refresh interval in our experiments. A higher refresh interval results in a higher number of retention failures in the system. This allows us to emulate future generation DIMMs with chips not fully tested for retention failures where many cells can actually fail at the nominal refresh interval. We analyze the efficacy of detection and mitigation techniques for retention failures at different refresh intervals, and show that our experimental observations on the intermittent failures hold, irrespective of the refresh interval.

Prior works have demonstrated that the retention time of DRAM cells decreases exponentially as temperature increases [11, 28]. In order to isolate the effect of temperature on retention failures, we perform temperature controlled experiments in a heat chamber (Figure 3b and 3c). Manufacturing-time retention tests are usually performed at a temperature of 85 °C. However, any system-level detection and mitigation technique will be deployed at run-time,

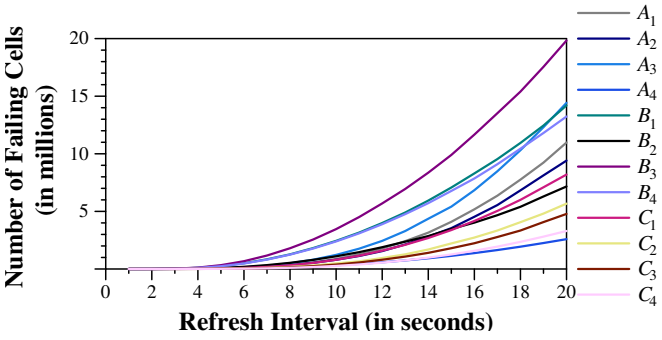


Figure 4: Number of Failures at Different Refresh Intervals

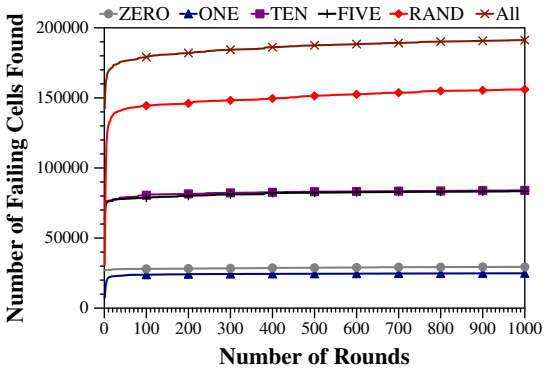


Figure 5: Number of Failures with Testing in *Module* A_1

where the operating conditions would not reach such high temperatures most of the time (e.g., the system might stop being operational at around 60 °C as hard-disks, for example, can have a thermal rating less than 60 °C). Our experiments are done at 45 °C to analyze the retention behavior in a typical system operating in the field.

Our results can be compared to prior work on retention time distribution at 85 °C, by scaling the refresh intervals used in our experiments to account for the change in temperature. Prior works showed that a 10 °C increase in temperature approximately halves the refresh interval [11, 28]. Based on experiments in our infrastructure, increasing the temperature by 10 °C reduces the refresh interval by 46.5% (Refer to the Appendix for details) [28]. We analyze retention failures with refresh intervals 1 second to 20 seconds at 45 °C, which correspond to 82 ms to 1640 ms at 85 °C.

We have tested twelve modules containing 96 chips from three major DRAM manufacturers. The capacity of each module is 2 GB. All the modules have a single rank and eight 2 Gb chips in the rank. The assembly dates of the modules are listed in Table 1.

Manufacturer	Module Name	Assembly Date (Year-Week)	Number of Chips
$\mathcal{A}$	A_1	2013-18	8
$\mathcal{A}$	A_2	2012-26	8
$\mathcal{A}$	A_3	2013-18	8
$\mathcal{A}$	A_4	2014-08	8
$\mathcal{B}$	B_1	2012-37	8
$\mathcal{B}$	B_2	2012-37	8
$\mathcal{B}$	B_3	2012-41	8
$\mathcal{B}$	B_4	2012-20	8
$\mathcal{C}$	C_1	2012-29	8
$\mathcal{C}$	C_2	2012-29	8
$\mathcal{C}$	C_3	2013-22	8
$\mathcal{C}$	C_4	2012-29	8

Table 1: Tested DRAM Modules

We perform a simple test to validate our infrastructure. Figure 4 shows the number of failing cells in each module at various refresh intervals of 1 to 20 seconds with an increment of 1 second at 45 °C (which corresponds to various refresh intervals between 82 ms to 1640 ms with 82 ms increment at 85 °C). At each interval, we write all ones to the entire module and then change the refresh interval. We wait for some specific amount of time to make sure that all the rows in the DIMM have been refreshed according to the new interval. Then we read out the entire DIMM and determine the number of cells that do not contain the original value written to them (i.e., failing cells). Then we repeat the experiment with all zeros to get the total number of failing cells at that refresh interval. Figure 4 presents the number of failing cells for all the modules. The number of retention failures increases exponentially with the refresh interval, as reported in the prior works [28, 11, 14]. We also performed other tests validating the failure rate with different data patterns in modules manufactured by different vendors and found that the results are consistent with previous works [28, 14, 45]. As our results are consistent with prior works, we conclude that our apparatus and methodology are sound.

4. EFFICACY OF TESTING

In this section, we analyze the adequacy of testing in detecting retention failures. We perform experiments to answer these questions: 1) How many rounds of testing are required for detecting the VRT failures? 2) How does the probability of finding a new failure reduce with rounds of testing? 3) How long does a VRT cell stay in different retention states?

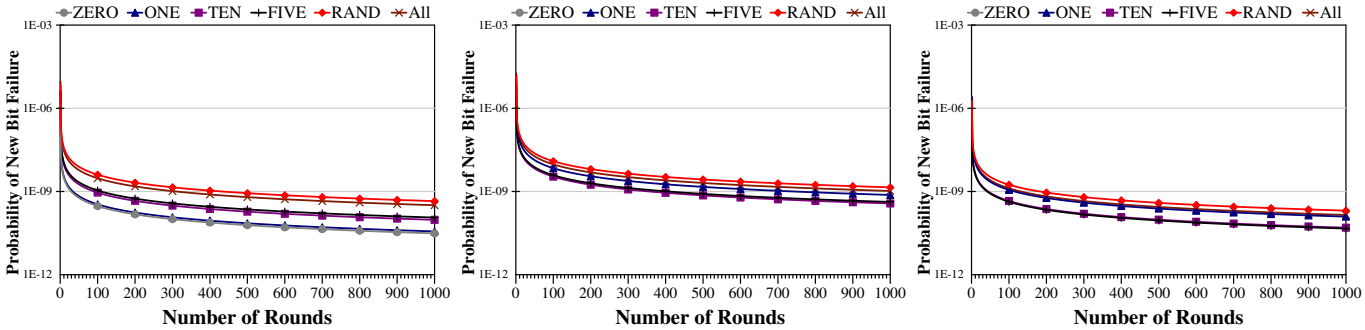
4.1 Detecting Retention Failures with Testing

4.1.1 Description of the Experiment

In this experiment, we test the modules at a refresh interval of 5 s for 1000 rounds at 45 °C (410 ms at 85 °C). The intent of the experiment is to study the effectiveness of testing in the context of a high number of retention failures to emulate future modules with chips not fully tested for retention failures, where many cells can actually fail at the nominal refresh rate. We found that a refresh interval of 5 seconds yields a retention failure rate of 10^{-6}, which provides insights into the behavior of intermittent failures with a relatively higher failure rate. We also provide results at other refresh intervals to show that the observations hold irrespective of the refresh interval deployed in the system.

We run experiments with different data patterns (zeroes (0*b*0000), ones (0*b*1111), tens (0*b*1010), fives (0*b*0101), and random) written to the entire DRAM for 1000 rounds. In the experiment with the random patterns, we write a randomly generated data pattern at each round. Thus, in tests with random data patterns, the pattern changes in each round. For all other tests, the data pattern (ones, zeroes, tens, and fives) remains the same across rounds. We also perform an experiment where all patterns are tested in each round (denoted as "All" in graphs). We count the number of failing cells discovered so far in each round. Our methodology of testing with rounds is similar to that described in [28].

Figure 5 shows the number of failing cells found versus number of rounds used for testing for the module A_1. We plot the number of failing cells for each of the patterns tested (ZERO, ONE,



(a) *Module* A_1 (b) *Module* B_1 (c) *Module* C_1

Figure 6: Probability of Discovering New Errors with Testing

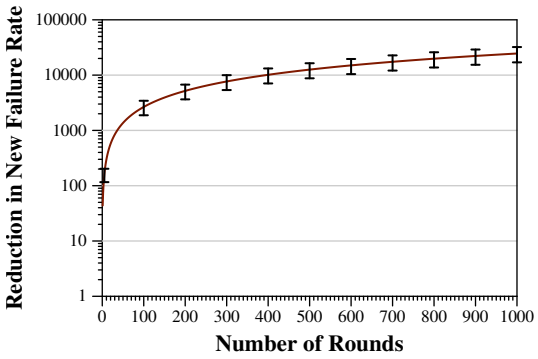


Figure 7: Reduction in Failure Rate in All Tested Modules

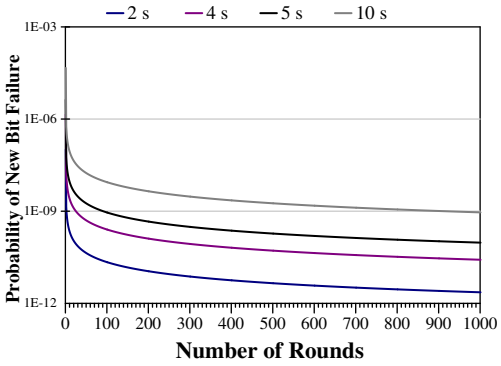


Figure 8: Probability of Retention Failure at Different Refresh Intervals in *Module* A_1

TEN, FIVE, RAND, All). We observe that there is a common trend among all the patterns: There is a sharp increase in the number of new failing cells within the first few rounds, but then the curves become relatively flat. This implies that the first few rounds of tests discover most of the intermittent failures and not too many new cells are found to fail after the first few rounds. However, a small number of new cells fail even after a considerable number of rounds. We observe that other modules demonstrate very similar behavior. Due to space constraints, we do not present all the results for each of the modules, but present results for one module from each vendor and a summary of the results over all the tested modules. However, detailed figures for all modules and data sets can be found online at the SAFARI Research Group website [16].

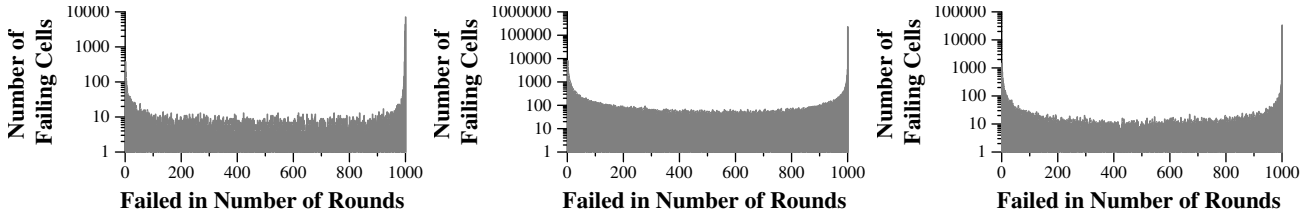
Observation: *Only a few rounds of tests can discover most of the retention failures. However, even after thousands of rounds of testing, a very small number of new cells are discovered to be failing.*

Implication: *Significant retention failure rate reduction is possible with only a few rounds of tests, but testing alone cannot detect all possible retention failures.*

4.1.2 Reducing Retention Failure Rate with Testing

We have empirically observed that only a few rounds of tests can detect most of the retention failures. This observation implies that the probability of detecting a new failure reduces significantly after only a few rounds of testing. We use the number of new failures detected at each round from Figure 5 and calculate the probability of discovering a new failure per round (using Equation (1) in Appendix). Figure 6 presents the reduction in failure rate with rounds of testing for one module from each vendor. This reduction in failure rate with testing can enable the estimation of number of required rounds of online testing to achieve a target reliability guarantee. A system-level profiling mechanism can observe the number of failures incurred at run-time and can determine the number of testing rounds required to reduce the probability of discovering a new failure to an acceptable limit. For example, Figure 5 shows there are more than 170000 retention failures in *module* A_1 with all tested patterns (retention failure rate of 10^{-6}). The system-level profiling mechanism can be configured to perform 300 rounds of testing (at which point there are around 18 new bit failures occurring every round, as can be calculated from Figure 5), reducing the retention failure rate to 10^{-9}. The number of required rounds is obtained by observing the point at which the curve for the probability of a new failure with all the tested patterns crosses 10^{-9} in Figure 6a.

Figure 7 shows the average reduction in the probability of a new failure versus the number of testing rounds for all the tested modules. Standard deviation of the reduction in failure rate is also plotted (as error bars on data points). This figure shows that only 5 rounds of tests can reduce the new failure rate by more than 100 times, but the reduction in failure rate becomes greatly diminished as the number of testing rounds increases beyond 100: each additional round finds a small number of retention failures. We also present the probability of a new failure with testing at different refresh intervals in Figure 8. This figure shows that the probability of discovering a new failure with testing reduces at a similar rate irrespective of the refresh interval.



(a) *Module* A_1 (b) *Module* B_1 (c) *Module* C_1

Figure 9: Number of Cells Failing in Different Number of Rounds

Based on our experiments, we conclude that 1) although the absolute number of retention failures varies across modules and refresh intervals, only a few rounds of tests can reduce the probability of a new failure significantly, 2) however, even after 1000 rounds of testing there is a small number of new failures discovered every round, and 3) the probability of discovering a new failure decreases slowly after the initial rounds of testing. These results imply that testing by itself, is not enough to detect and mitigate all the retention failures.

4.1.3 Retention Failure Coverage with Testing Rounds

The efficacy of testing depends on the consistency of a particular cell failing in different rounds of tests. We discuss three cases to demonstrate how consistency of failure can affect the efficacy of testing for a cell: 1) if a cell consistently fails in *all* rounds of tests, it is likely a very weak cell, perhaps with a permanent retention failure; these can be effectively discovered with testing, 2) if a cell fails in a majority of rounds of tests, it is again likely that testing will effectively discover it, 3) if a cell fails in only a few rounds of tests, then it is less likely to be discovered with testing (at least within a reasonable amount of time). We present the total number of rounds a particular cell fails in our experiments to determine how effective testing could be across all cells. Instead of showing each unique cell, we group all cells that fail in the same number of rounds. Figure 9 plots the number of cells that fail in a total of N rounds, where N varies from 1 to 1000. The peak at 1000 rounds suggests that the majority of the failing cells are very weak cells that fail in every round. All the other cells are VRT cells that fail only intermittently. The second highest peak around round 1 shows that a significant number of cells fail only in one round, indicating that it will likely be difficult to discover a significant portion of intermittent failures through system-level testing. Figure 10 shows the average, minimum and maximum number of cells failing in one to 1000 rounds across all tested modules. Based on the similarity of the curves, and the consistent existence of a large fraction of cells that fail in only a small number of testing rounds, we conclude that testing alone is likely ineffective at efficiently discovering a large fraction of the intermittently failing cells.

4.2 Undetected Retention Failures

The previous experiment showed that a large fraction of intermittently failing cells can remain undetected by testing as there are some VRT cells that operate correctly (pass) for a long time and then fail for a short period of time. Next, we analyze the time a VRT cell spends in its different retention (time) states. The intent of the experiment is to determine the percentage of VRT cells that fail after spending a long time in the non-failing retention state, and that are therefore are hard to detect through testing.

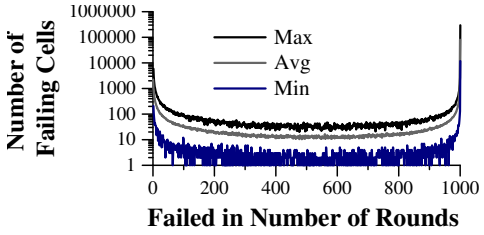


Figure 10: Average, Minimum and Maximum Number of Cells Failing in Different Number of Rounds

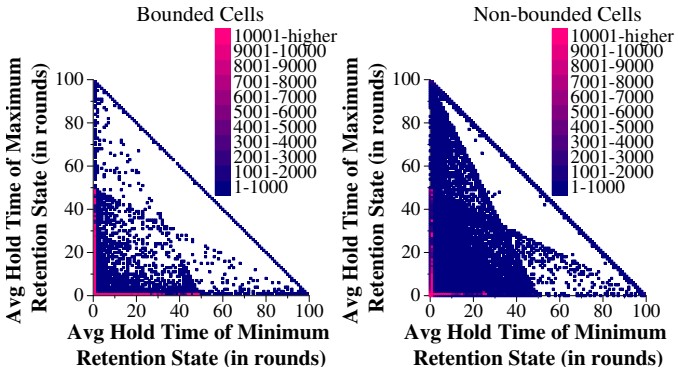
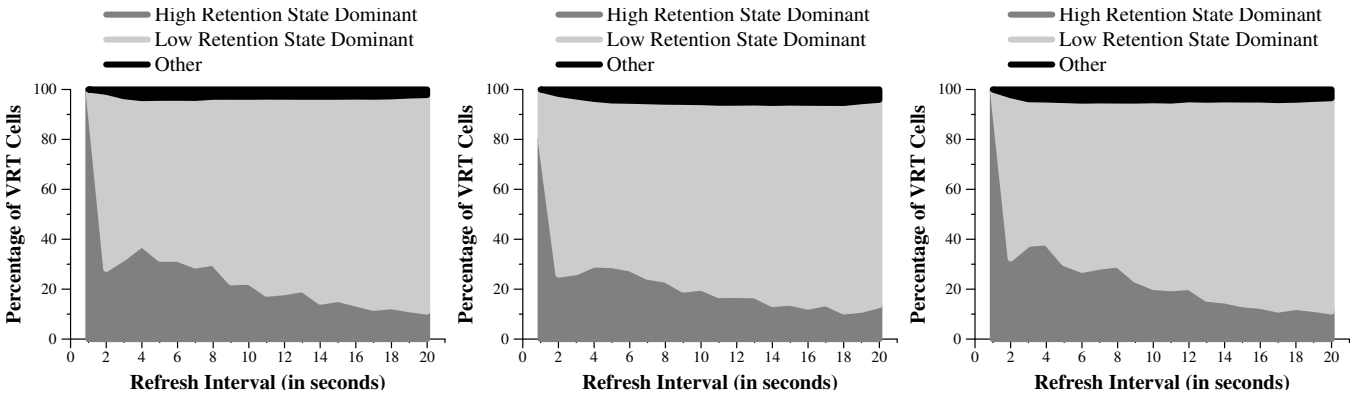


Figure 11: Average Hold Time of Cells in *Module* A_1

4.2.1 Description of the Experiment

In this experiment, we measure the average amount of time a cell spends in its different retention states. We have tested the modules at a refresh interval of 1 to 20 seconds with a 1-second increment for 100 rounds at 45 °C (corresponding to 82 ms to 1640 ms with an 82 ms increment at 85 °C). We monitor the number of rounds a cell spends at a specific retention state before it moves to some other retention state, referred to as *hold time*. After 100 rounds, we calculate the average hold time a cell spends in its minimum and maximum retention state.

We divide the cells into two different categories: cells with bound-ed and unbounded maximum retention times. The first category, called the bounded cells, consists of the cells that always have retention states lower than 20 s. These cells always fail at potentially different refresh intervals within our tested refresh interval of 20 s. The other category, called the non-bounded cells, consists of cells that fail within the 20 s of refresh interval in some rounds, but do not fail at all in at least one round. Since they do not fail in a round, we cannot accurately determine (hence, bound) the maximum retention time of these cells. Figure 11 plots the average hold times of the maximum and minimum retention states of bounded



(a) *Module* A_1 (b) *Module* B_1 (c) *Module* C_1

Figure 12: Percentage of VRT Cells with High or Low Retention States Dominant

and non-bounded cells in module A_1. Instead of representing each cell, we group the failing cells that have the same hold times. Each point (x, y) in the figure represents a group of cells that spends on average x rounds in the minimum retention state and y rounds in the maximum retention state. We use color intensity to represent the number of cells at each point. The more cells belonging to a specific bin, the brighter the color of that bin.

We observe from the figure that most of the cells appear in the bins that are very close to either the x axis or the y axis. The cells near the y axis spend a very short time (1-2 rounds) in the low retention state. Similarly, the cells near the x axis spend a very short time in the high retention state. This characteristic is similar for bounded and non-bounded cells in all the tested modules. This trend implies that most cells have dominant states. We call those cells that spend more than 98 (out of 100) rounds in either the low or high retention states as dominant-state cells. Cells with a dominant low retention state frequently fail in tests and can be discovered easily. However, cells with a dominant high retention state mostly spend their time in the high retention state. Therefore, these cells tend to not fail in most of the testing rounds. As a result, these cells are hard to discover through testing.

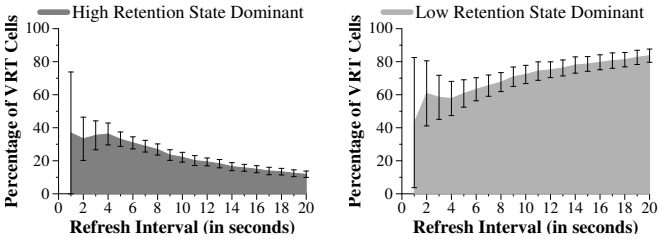
Observation: *Most VRT cells have dominant retention time states: they spend most of the time either at a low or high retention state.*

Implication: *VRT cells with a dominant high retention state are hard to discover through testing.*

4.2.2 Number of Undiscovered Retention Failures

Based on our data, we present the percentage of cells that spend a long time in the high retention state among all the VRT cells. Figure 12 shows the percentage of dominant-state cells at each refresh interval for one module from three manufacturers. Figure 13 shows the percentage of dominant-state cells averaged over all the tested modules, along with the standard deviation. Based on the figures, we make three conclusions on the behavior of VRT cells. First, on average around 90% of the cells are either low-state-dominant or high-state-dominant cells. Second, a significant portion of the VRT cells (around 40-20%) are high-state-dominant cells irrespective of the refresh interval. These cells would be hard to discover. Third, the percentage of high-state-dominant cells decreases with refresh interval. The reason is high refresh interval increases the chance that a cell is more likely to have a lower retention state.

Based on these experimental observations, we conclude that there will likely be a significant portion of VRT cells that remain



(a) High State Dominant (b) Low State Dominant

Figure 13: Percentage High and Low Retention State Dominant Cells in All Tested Modules

undetected by testing at any refresh interval deployed in the system. This reinforces our previous observation (in Section 4.1) that testing alone is likely not enough to discover all retention failures.

5. EFFICACY OF GUARDBANDING

The effectiveness of the guardband depends on the differences in retention states of a VRT cell. A small guardband would be effective if the retention time difference between the retention states of a cell is small. We want to answer these questions: 1) How much difference is present in the retention time states of VRT cells? 2) How effective is adding a guardband at the system-level, in the presence of a large number of intermittent VRT failures?

5.1 Description of the Experiment

In this experiment, similar to Section 4.2, we have tested the modules at a refresh interval of 1 to 20 seconds with a 1-second increment for 100 rounds at 45 °C (corresponding to 82 ms to 1640 ms with an 82 ms increment at 85 °C). However, this time, we monitor the retention state of the cells in each round. VRT cells exhibit more than one retention time state in different rounds. We categorize the cells into two-state and multi-state cells. The two-state cells are the cells that move back and forth between only two retention states. The multi-state cells exhibit more than two states. Similar to the prior experiment, we further categorize these cells as bounded and non-bounded cells (Section 4.2 provides the definition of these). Recall that a bounded cell always has retention states lower than 20 s, the maximum tested retention time, whereas a non-bounded cell does not fail the retention test in at least one

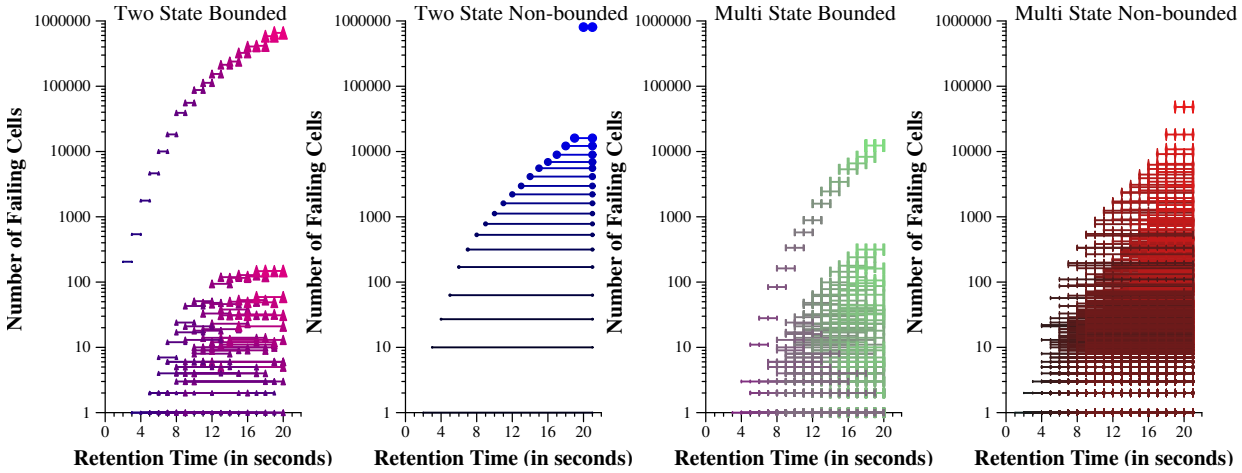
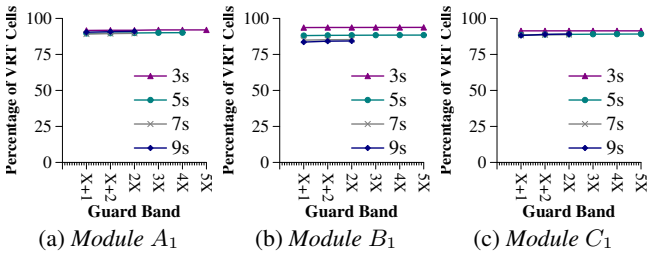


Figure 14: Retention States in *Module* A_1



(a) *Module* A_1 (b) *Module* B_1 (c) *Module* C_1

Figure 15: Coverage of Guardbanding in Module A_1, B_1, and C_1

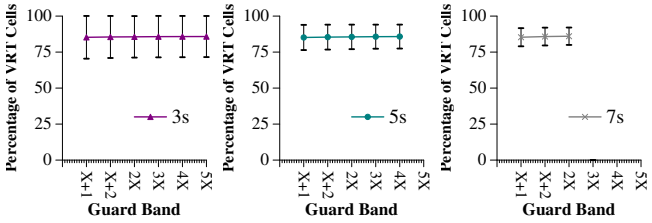


Figure 16: Coverage of Guardbanding in All Modules

round, which means that its maximum retention time cannot be determined with our experiments. Figure 14 presents the retention states of all the VRT cells found in this experiment for *module* A_1. Due to space constraints, we place the results from other modules online [16]. The behavior remains similar across the tested modules and we summarize the results for one module from each vendor in Figure 15.

Figure 14 shows the retention states of two-state (bounded and non-bounded) cells and multi-state (bounded and non-bounded) cells in different plots. Instead of plotting every cell, we group the cells having the same retention states. Each line in the plots represents a group of cells with the same retention states. The markers in the line represent the retention states of that group of cells. For example, a line at (x_1, y) to (x_2, y) represents that there are y cells that have retention states x_1 and x_2. The overlapping lines indicate that there are some groups of cells with different retention states but the same number of cells in the group. To distinguish among these groups, we use different marker sizes for different groups. a larger marker size represents a higher retention state. We plot the non-bounded retention time state as 21 s in the plots.

The first plot shows the retention states of the two-state bounded cells. We observe that most of the cells have very close retention states (notice the log scale in the y axis). There are only a few cells that have non-consecutive retention states. The next plot shows the retention states of the two-state non-bounded cells. We observe that the number of cells that exhibit non-bounded behavior is almost 10X lower than the bounded cells. For example, for module A_1, 110 cells have retention states of 3 s and 4 s, but only 10 cells move between the retention state of 3 s and the non-bounded state. This implies that a small guardband can be effective for a large fraction of the cells for tolerating intermittent retention time changes. We see a similar trend with the multi-state cells: most of the cells have close-by retention states and only a small number of cells shows a large difference among the multiple states. For cells where there is a large difference in the different retention time states, even a large amount of guardband may not be effective at tolerating intermittent retention time changes.

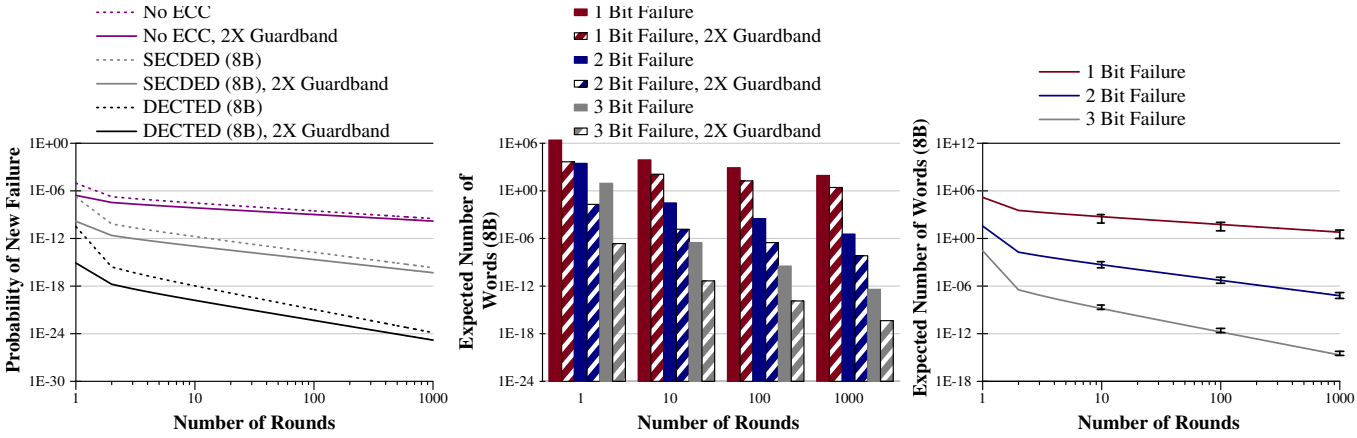
Observation: *Most of the intermittently failing cells have very close retention states in terms of retention time. However, there also exists VRT cells with large differences in retention states.*

Implication: *Even a small guardband is likely effective for most of the VRT cells. However, even a large guardband is likely ineffective for the remaining VRT cells with large differences in retention time states.*

5.2 Coverage of Guardbanding

We determine the fraction of failing cells that different amounts of guardbanding can effectively mitigate (i.e., avoid failures for). To determine the coverage of each guardband, we perform three steps as described in Section 2.3.2. For example, at a refresh interval of 3 s the steps to determine the coverage of a $2X$ guardband are: 1) We identify the failing bits at a refresh interval of 6 seconds ($2X$). 2) These faulty bits are assumed to be repaired using different mitigation techniques and do not cause any further failures in the original refresh interval of 3 s. 3) Then, we determine the coverage of the guardband at 3 s by determining the fraction of VRT cells observed in our experiment that now operate correctly with the guardband but otherwise would have failed.

The guardbands considered are $X + 1s$, $X + 2s$, $2X$, $3X$, $4X$, and $5X$ at refresh intervals of 3, 5, 7, and 9 seconds. We show the



(a) *Module* A_1 (b) *Module* A_1 (c) All Modules

Figure 17: Effectiveness of ECC with Testing and Guardbanding

coverage of failing VRT cells with each guardband in Figure 15 for one module from each vendor. We also present the coverage averaged over all tested modules at each refresh interval, along with the standard deviation of coverage across all tested modules in Figure 16. Based on the figures, we make three observations. First, even a small guardband can achieve $85-95\%$ coverage. The reason is that most of the cells exhibit very close retention states and get repaired by the guardband. Second, the coverage does not change significantly even if we increase the amount of guardband to $5X$. This characteristic can be explained by our experimental observation that only a very small fraction of cells have large differences in retention time states. Third, coverage does not depend on the refresh interval and remains mostly the same.

We conclude that a small guardband (e.g., $2X$) can avoid most of the failing cells. However, even a large guardband (e.g., $5X$) is not effective for the remaining VRT cells, indicating that using a guardband alone is not effective to mitigate all intermittent retention failures.

6. EFFICACY OF ECC

In this section, we present the effectiveness of ECC when used with other mitigation techniques. We focus on two aspects of ECC, 1) reduction in the probability of a new failure and 2) expected number of multi-bit failures when the system can perform online testing and guardbanding.

When a system is capable of online testing, it can detect retention failures and repair those bits. Repairing the bits reduces the probability of a new bit failure and thus the system can adjust the amount of ECC required to correct the random VRT failures that occur after the system-level testing. The dotted lines in Figure 17a show the number of rounds required to reduce the probability of a new retention failure in the presence of single error correcting, double error detecting (SECDED) code and double error correcting, triple error detecting (DECTED) code versus the number of rounds rounds of tests employed. The solid lines also represent the probability of retention failure with SECDED and DECTED, but with an added guardband ($2X$). The retention failure rate with all the patterns (All) from Figure 6 is used to calculate the probability of failure in the presence of ECC at 8B granularity (derived from the Equations (2), (3), and (4) in Appendix). We make two observations from this figure. First, we can achieve a much higher reduction in the probability of a new failure, when the system employs testing and guardbanding *along with* ECC. Only a few rounds of testing can reduce the retention failure rate by $10^7/10^{12}$ times, when used in conjunction with SECDED/DECTED and a $2X$ guardband. Contrast this with the much smaller $100/10^5$ times reduction in retention failure rate when only SECDED/DECTED is used. Second, a higher number of rounds of tests *along with* ECC can further reduce the retention failure rate. The probability of a new failure can be reduced by as much as $10^{12}/10^{18}$ times after 1000 rounds of testing when testing is used in conjunction with SECDED/DECTED.

These observations imply that ECC, when used with other mitigation techniques, can effectively tolerate a high error rate.

Observation: *Testing and guardbanding along with ECC protection can significantly reduce the retention failure rate.*

Implication: *A combination of error mitigation techniques is more effective at tolerating a high error rate than each technique alone.*

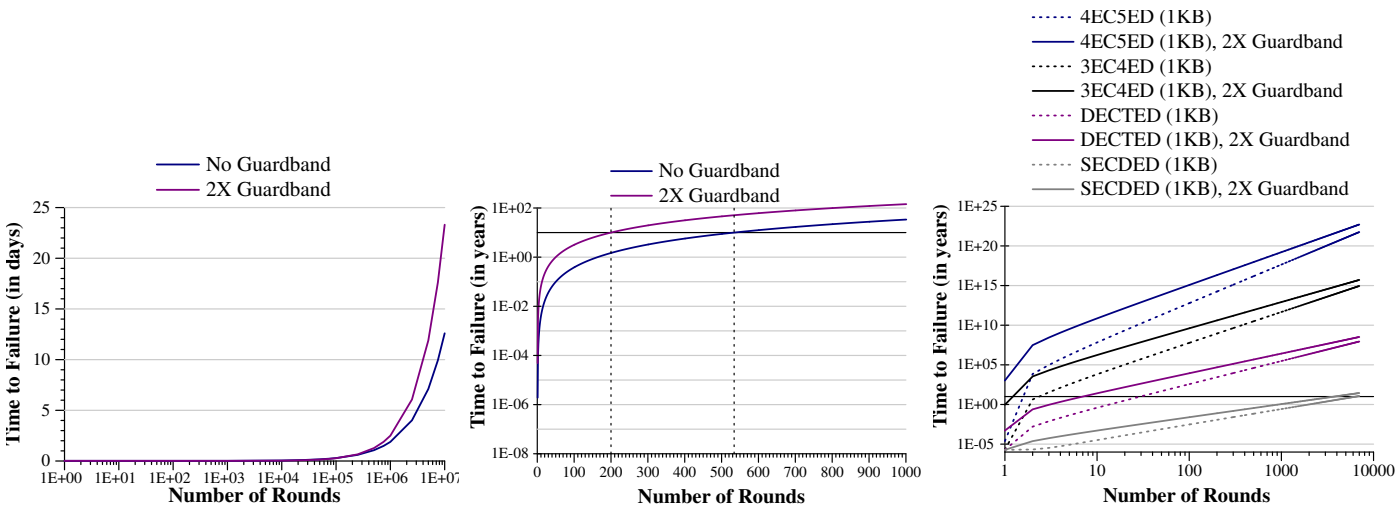
The strength of the required ECC depends on the probability of the multi-bit failures in a module. We present the expected number of words with 1, 2 and 3 bit failures when the system employs other mitigation techniques in Figures 17b and 17c (refer to Equation (5) in Appendix). These figures clearly show that the number of multi-bit failures reduces with rounds of testing. We make two observations. First, the expected number of single-bit failures reduces with testing, but does not reach to zero. Second, with 1000 rounds of testing, the expected number of 2-bit and 3-bit failures become negligible (10^{-6}). We conclude that a system-level online profiling mechanism can tolerate a higher failure rate when used in conjunction with other mitigation techniques.

7. EFFICACY OF SOPHISTICATED MITIGATION TECHNIQUES

In light of the efficacy of testing, guardbanding, and ECC observed and quantified in our experiments, we evaluate the adequacy of some recently proposed error mitigation techniques.

7.1 Bit Repair Techniques

System-level bit repair techniques perform online testing and repair the detected faulty bits using different mechanisms (remapping, higher refresh rate for the faulty rows, disabling faulty pages, error correcting pointers, etc) [38, 27, 50, 26]. These mechanisms assume that all bit failures can be detected by testing after the initial



(a) Bit Repair [38, 27, 50] (b) VS-ECC [4] (c) Hi-ECC [52]

Figure 18: Effectiveness of Recent Error Mitigation Techniques with Testing and Guardbanding

system boot-up and no new errors occur in DRAM after repairing the failing bits detected by the initial test. In our experiments, we observe that even after thousands rounds of testing, new bits keep failing at a very low rate. This implies that all these mechanisms will fail even after very long (initial) tests. Figure 18a presents the expected time to failure of these mechanisms when the system can run a million rounds of tests. The time to failure of the system is calculated using the probability of a new failure from Figure 6 (refer to Equation (6) in Appendix). With each round of testing, the detected faulty bits get repaired and the probability of a new failure keeps decreasing. However, in these mechanisms, the system will fail even if a single bit failure occurs during runtime after the initial testing. Figure 18a shows that the system will fail within hours after 10 thousand rounds of tests. Even after 10 million rounds of initial testing, the system will fail in 13 days. We also present the expected time to failure with an added guardband. Even though adding a guardband improves time to failure, the system still fails within 23 days after more than 10 million rounds of initial testing. In a real system, each round of test consists of writing one specific data pattern in the entire module, waiting for 64 ms to keep the cells idle so that the retention failures can manifest and then reading the entire module to detect the failing cells. It will take 413.96 ms to test a 2 GB module for a round with just one data pattern (refer to Equation (8) in Appendix for details). 10 million rounds of initial testing with all our tested patterns would take more than 7.9 months. Even so, the system would fail within a month after the initial test. Our analysis, thus shows that bit-repair mechanisms that perform only initial tests (as they were proposed) are not feasible to deploy in a practical system in the presence of intermittent retention failures.

7.2 Variable Strength ECC (VS-ECC)

VS-ECC uses variable strength ECC to protect different lines in memory [4]. The system performs an initial test to detect lines with one or more errors and protect them using 4EC5ED code, but uses SECDED for rest of the memory. VS-ECC will fail if there are two or more errors in the lines protected by SECDED. We calculate the probability of failure for VS-ECC and determine the expected time to failure using the probability of failure from Figure 6 (refer to Equation (7) in Appendix). After each round of testing, the new failing words found are protected by stronger ECC. Figure 18b shows the expected time to failure with number of rounds of testing. Within 550 rounds of testing (which takes around 19 minutes) the system achieves a time-to-failure of 10 years. This figure also shows that adding a guardband can reduce the number of rounds of testing required to achieve the same guarantee. With a $2X$ guardband, the system needs only 200 rounds of testing (which takes around 7 minutes). Our analysis illustrates that a system has to run the initial test for several minutes to achieve a reasonable reliability guarantee even with a 2 Gb chip. The length of the initial test could be longer in future high density chips. Blocking entire memory for a significant amount of time may result in throughput loss and thus such a mechanism would be difficult to deploy in a practical system. This implies that a practical and efficient online profiling system should be designed to make VS-ECC effective. In such a mechanism, testing would be spread across a large interval and be performed in isolated parts of the memory such that other programs can still use the remaining parts of the memory.

7.3 Higher Strength ECC (Hi-ECC)

Hi-ECC is an ECC based mechanism that uses very strong ECC (5EC6ED) to tolerate a high error rate [52]. It amortizes the cost of ECC by protecting a larger chunk of data (1KB). We show that testing and guardbanding can be very effective at providing the same reliability guarantee but using weaker ECC. In Figure 18c, we present the expected time to failure of a system that can use online testing and guardbanding. This figure shows that with only one round of test (which takes 2.06 s), the system can effectively reduce ECC strength to 3-bit correcting, 4-bit detecting (3EC4ED) code and can still provide a 10-year time-to-failure guarantee. After 100 rounds of tests (which takes around 3.5 minutes), ECC strength can be reduced to DECTED. To decrease the strength to SECDED, the system will need to perform around 7000 rounds of tests (taking almost 4 hours), reducing the overhead of ECC by 80%. This figure also shows that with an added guardband, this system can start with 4EC5ED and can reduce ECC strength to DECTED within 10 rounds of testing (20.6 s). However, the system will still need to go through 7000 rounds of tests (taking almost 4 hours) to apply SECDED. Though blocking memory for 4 hours is not acceptable, if these 7000 rounds of tests are spread over time for only some

rows in memory (such that testing can be efficient), we can still provide 10-years of time to failure guarantee using only SECDED.

This observation implies an interesting characteristic of online mitigation techniques. We show that we can enable many different optimizations if an effective online profiling mechanism can be designed to run continuous tests in the background without disrupting other programs. A system can start with a strong ECC code at the beginning (same as Hi-ECC), but instead of paying the latency, area, and power penalty for strong ECC at every access, an effective online profiling mechanism can use SECDED after profiling for errors for some time. Based on our observations and analysis, we sketch a high-level architectural design for an efficient system-level online profiling mechanism in the next section.

8. ENABLING A SYSTEM-LEVEL ONLINE PROFILING MECHANISM

An online profiling technique for DRAM would test the module while the system is running and the memory is in use. The memory controller would be responsible for locating and repairing the faulty cells to mitigate the errors and ensure reliable DRAM operation. Such a system would not only improve DRAM reliability, but would also increase DRAM yield even in the presence of high error rates, by identifying and repairing bit errors rather than discarding chips with errors. An effective online profiling mechanism can address DRAM scaling challenges and play a critical role in enabling high-density, low-cost DRAM in the future.

8.1 Designing an Online Profiling Mechanism

An online profiler needs to be non-intrusive, operating in the background to allow continual testing without disrupting the use of the system. Long tests are undesirable as they would prevent user programs from accessing memory, resulting in a significant performance overhead. In light of our observations, we suggest that an online profiling mechanism can be designed by using a combination of testing, guardbanding, and ECC. The goal of an effective online profiling mechanism is to detect most of the errors with short tests performed at regular intervals. Here, we sketch the steps involved in a preliminary online profiling mechanism. The evaluation of such a mechanism is outside the scope of this work.

Initial Reliability Guarantee using ECC: Initially, before any profiling, the chips are protected by ECC to guarantee reliable DRAM operation. The memory controller is responsible for correcting erroneous data through a mechanism like virtualized ECC [56] whose error correction strength can be varied dynamically. Without any testing and guardbanding, the cost of ECC would likely be high.

Discovering Errors with Short Tests: Next, the memory controller runs tests to discover and repair the faulty bits. Adding a guardband can reduce the error rate by ten times(Section 5). The test for adding a guardband involves testing the chips at a high refresh interval and can be done just after the system starts. The guardband test can be done within a short period as it does not require rounds of tests. Later, during the regular use of the memory, the memory controller can run short rounds of tests to discover the intermittently failing cells. In order to prevent major performance overheads, a round of test is run after some regular interval. Thus, in order to reduce the overhead of testing, we propose that future online profilers test small regions of memory at regular intervals.

Adjusting the ECC strength: After some number of testing rounds, most of the failing bits get discovered and effectively get repaired. When the rate of retention failures reduces by an acceptable amount, the strength of the employed ECC can be correspondingly adjusted to reduce the ECC overhead.

8.2 Challenges and Opportunities of an Online Profiling Mechanism

We briefly describe some of the challenges and potential opportunities of designing an online profiler.

Reducing Performance Overhead: In a real system, the overhead of running even one round of test in a specific region of memory can have noticeable performance overhead. A round of test consists of writing some data pattern in the region under test, waiting for a certain amount of time to make sure cells are idle for the entire refresh interval and then reading out that region to locate the failures. This test would make the region under test unavailable to programs for hundreds of milliseconds. We argue that future online profiling works must address the challenges of reducing this performance overhead by designing intelligent mechanisms. We sketch some potential directions that can be used to mitigate the performance overhead: 1) One mechanism can be pinning the specific memory region into the cache while that region is being tested. Memory requests to the region under test would be satisfied by the cache and would not block the programs. However, there is a trade-off between the size of the region that can be temporally stored in the cache without evicting a large portion of the working set vs. the number of individual tests that would be required to profile the whole module. 2) Keeping a pool of unallocated pages and testing them extensively before they are allocated to a program can improve performance at the cost of some effective memory capacity loss. This hardware-software collaborative mechanism would need an interface to the system software to prevent allocating pages under test. 3) Many applications have periodic compute and memory phases. A memory controller can be designed to predict phases where a portion of memory remains idle and run online tests during those periods to reduce the performance overhead.

Reducing Mitigation Overhead: The ultimate goal of detecting failures through online profiling is to mitigate those failures and provide reliable operation. Recent mechanisms proposed efficient techniques to reduce the overhead of mitigation mechanisms. For example, ArchShield, which mitigates failures by remapping faulty words in a region of memory, uses a fault map to efficiently determine the location of remapping [38]. Using a higher refresh rate for rows with failures is another mitigation technique. RAIDR uses Bloom filters to efficiently store the location of the rows that are required to be refreshed more frequently [27]. However, as we have shown in Section 7, these mechanisms as proposed, do not consider intermittent failures and lead to potential data loss. We envision these techniques would be extended with online profiling mechanisms to dynamically determine the current set of failures and optimize the overheads depending on the current failure rate.

Enabling Failure-aware Optimizations: An online profiling mechanism enables optimization techniques to take advantage of the inherent resiliency of some applications at run-time. Many of the previously proposed resiliency techniques that allocate error-prone regions to data that can tolerate errors [29, 7, 30, 25, 9] would directly benefit from an efficient online profiling mechanism, by being able to determine the error-prone locations at run-time.

9. CONCLUSION

We have studied and analyzed the effectiveness of different system-level error mitigation techniques for retention failures in commodity DRAM, with the goal of enabling efficient and effective reliability techniques for future DRAM. We make several observations on the error mitigation techniques based on the experimental data collected from 96 DRAM chips manufactured by three different vendors, using an FPGA-based DRAM testing infrastructure. First, we show that only a small amount of testing can discover

the majority of cells with intermittent retention failures. From our experiments, only 5 rounds of tests can discover most of the intermittent failures and reduce the probability of retention failure by 100 times. However, even after thousands of rounds of testing, a very small number of cells exhibit new failures not discovered before. Second, we show that even a small guardband (e.g., $2X$) can avoid 85-95% of the intermittently failing cells and reduce the probability of retention failure by ten times. At the same time, even a large guardband (e.g., $5X$) is not effective for the remaining intermittently failing cells. Third, we show that using only single error correction codes can reduce the retention error rate by only 100 times, but using single error correction codes together with testing and guardbanding can reduce the error rate by as much as 10^{12} times. Fourth, based on our data, we quantify recently proposed system-level error mitigation techniques that do not consider intermittent failures, showing that our measured results significantly impact these works' conclusions. We show that bit repair mechanisms that rely on testing [38, 27, 50, 26] cannot provide strong reliability guarantees even after months of testing. On the other hand, ECC-based mitigation techniques [4, 52] can ensure reliable DRAM operation when employed in conjunction with online testing in a relatively short amount of time. We conclude that the viability of these techniques depend on the development of an efficient online profiling mechanism that does not significantly disrupt the operation of the programs running on the system.

We hope that the empirical study and analysis of retention error mitigation techniques, driven by experimental measurements from real DRAM chips, presented in this paper can enable new, effective, and efficient mechanisms in the future that will lead to more reliable design and operation of future DRAM systems. In particular, we believe the development of efficient online retention time profiling techniques for DRAM is a promising area of immediate future work that can benefit from our characterizations and analyses.

ACKNOWLEDGEMENTS

We are grateful to Uksong Kang from Samsung for his helpful comments. We thank the anonymous reviewers for their helpful feedback, and gratefully acknowledge the SAFARI Research Group members for providing useful feedback. We acknowledge the support of the Intel Science and Technology Center on Cloud Computing. We thank our industrial partners for their support: IBM, Intel, Qualcomm, and Samsung. This research was also partially supported by grants from NSF (CAREER Award CCF 0953246, CCF 1212962, and CNS 1065112).

REFERENCES

[1] R. D. Adams. *High performance memory testing: Design principles, fault modeling and self-test*. Springer, 2003.

[2] J.-H. Ahn et al. Adaptive self refresh scheme for battery operated high-density mobile DRAM applications. ASSCC, 2006.

[3] Z. Al-Ars et al. DRAM-specific space of memory tests. ITC, 2006.

[4] A. R. Alameldeen et al. Energy-efficient cache design using variable-strength error-correcting codes. ISCA, 2011.

[5] R. Baumann. The impact of technology scaling on soft error rate performance and limits to the efficacy of error correction. IEDM, 2002.

[6] K. Chang et al. Improving DRAM performance by parallelizing refreshes with accesses. HPCA, 2014.

[7] M. de Kruijf et al. Relax: An architectural framework for software recovery of hardware faults. ISCA, 2010.

[8] P. G. Emma et al. Rethinking refresh: Increasing availability and reducing power in DRAM for cache applications. *IEEE Micro*, 28(6), Nov. 2008.

[9] H. Esmaeilzadeh et al. Neural acceleration for general-purpose approximate programs. MICRO, 2012.

[10] D. Frank et al. Device scaling limits of Si MOSFETs and their application dependencies. *Proceedings of the IEEE*, 89(3), 2001.

[11] T. Hamamoto et al. On the retention time distribution of Dynamic Random Access Memory (DRAM). 1998.

[12] P. Hazucha and C. Svensson. Impact of CMOS technology scaling on the atmospheric neutron soft error rate. *TNS*, 47(6), 2000.

[13] A. Hiraiwa et al. Local-field-enhancement model of DRAM retention failure. IEDM, 1998.

[14] C.-S. Hou et al. An FPGA-based test platform for analyzing data retention time distribution of DRAMs. VLSI-DAT, 2013.

[15] JEDEC. *Standard No. 79-3F. DDR3 SDRAM Specification*, July 2012.

[16] S. Khan et al. The efficacy of error mitigation techniques for DRAM retention failures: A comparative experimental study – Full data sets. http://www.ece.cmu.edu/~safari/tools/dram-sigmetrics2014-fulldata.html.

[17] H. Kim et al. Characterization of the variable retention time in dynamic random access memory. *IEEE Trans. Electron Dev.*, 58(9), 2011.

[18] K. Kim. Technology for sub-50nm DRAM and NAND flash manufacturing. IEDM, 2005.

[19] K. Kim and J. Lee. A new investigation of data retention time in truly nanoscaled DRAMs. *IEEE Electron Device Letters*, 30(8), 2009.

[20] Y. Kim et al. A case for exploiting subarray-level parallelism (SALP) in DRAM. ISCA, 2012.

[21] Y. I. Kim et al. Thermal degradation of DRAM retention time: Characterization and improving techniques. IRPS, 2004.

[22] D. Lee et al. Tiered-latency DRAM: A low latency and low cost DRAM architecture. HPCA, 2013.

[23] M. J. Lee and K. W. Park. A mechanism for dependence of refresh time on data pattern in DRAM. *Electron Device Letters*, 31(2), 2010.

[24] X. Li et al. A realistic evaluation of memory hardware errors and software system susceptibility. ATC, 2010.

[25] X. Li and D. Yeung. Application-level correctness and its impact on fault tolerance. HPCA, 2007.

[26] C.-H. Lin et al. SECRET: Selective error correction for refresh energy reduction in DRAMs. ICCD, 2012.

[27] J. Liu et al. RAIDR: Retention-aware intelligent DRAM refresh. ISCA, 2012.

[28] J. Liu et al. An experimental study of data retention behavior in modern DRAM devices: Implications for retention time profiling mechanisms. ISCA, 2013.

[29] S. Liu et al. Flikker: Saving DRAM refresh-power through critical data partitioning. ASPLOS, 2011.

[30] Y. Luo. Characterizing application memory error vulnerability to optimize data center cost. DSN, 2014.

[31] J. A. Mandelman et al. Challenges and future directions for the scaling of dynamic random-access memory (DRAM). *IBM J. of Res. and Dev.*, 2002.

[32] T. C. May et al. Alpha-particle-induced soft errors in dynamic memories. *IEEE Trans. Electron Dev.*, 1979.
[33] Y. Mori et al. The origin of variable retention time in DRAM. IEDM, 2005.
[34] W. Mueller et al. Challenges for the DRAM cell scaling to 40nm. IEDM, 2005.
[35] S. S. Mukherjee et al. The soft error problem: An architectural perspective. HPCA, 2005.
[36] O. Mutlu. Memory scaling: A systems architecture perspective. *IMW*, 2013.
[37] P. Nair et al. A case for refresh pausing in DRAM memory systems. HPCA, 2012.
[38] P. J. Nair et al. ArchShield: Architectural framework for assisting DRAM scaling by tolerating high error rates. ISCA, 2013.
[39] H.-D. Oberle et al. Enhanced fault modeling for DRAM test and analysis. VTS, 1991.
[40] T. J. O'Gorman. The effect of cosmic rays on the soft error rate of a DRAM at ground level. *IEEE Trans. Electron Dev.*, 41(4), 1994.
[41] P. J. Restle, J. W. Park, and B. F. Lloyd. DRAM variable retention time. IEDM, 1992.
[42] S. E. Schechter, G. H. Loh, et al. Use ECP, not ECC, for hard failures in resistive memories. ISCA, 2010.
[43] B. Schroeder et al. DRAM errors in the wild: A large-scale field study. SIGMETRICS, 2009.
[44] H. W. Seo et al. Charge trapping induced DRAM data retention time degradation under wafer-level burn-in stress. IRPS, 2002.
[45] V. Sridharan et al. Feng Shui of supercomputer memory: Positional effects in DRAM and SRAM faults. SC, 2013.
[46] V. Sridharan and D. Liberty. A study of DRAM failures in the field. SC, 2012.
[47] G. R. Srinivasan et al. Accurate, predictive modeling of soft error rate due to cosmic rays and chip alpha radiation. IRPS, 1994.
[48] A. J. van de Goor et al. An overview of deterministic functional RAM chip testing. *ACM Computing Surveys*, 1990.
[49] A. J. van de Goor and A. Paalvast. Industrial evaluation of DRAM SIMM tests. ITC, 2000.
[50] R. K. Venkatesan et al. Retention-aware placement in DRAM (RAPID): Software methods for quasi-non-volatile DRAM. HPCA, 2006.
[51] M.-J. Wang et al. Guardband determination for the detection of off-state and junction leakages in DRAM testing. ATS, 2001.
[52] C. Wilkerson et al. Reducing cache power with low-cost, multi-bit error-correcting codes. ISCA, 2010.
[53] Xilinx. *ML605 Hardware User Guide*, Oct. 2012.
[54] K. Yamaguchi. Theoretical study of deep-trap-assisted anomalous currents in worst-bit cells of dynamic random-access memories (DRAM's). *IEEE Trans. Electron Dev.*, 47(4), 2000.
[55] D. Yaney et al. A meta-stable leakage phenomenon in DRAM charge storage - Variable hold time. IEDM, 1987.
[56] D. H. Yoon and M. Erez. Virtualized and flexible ECC for main memory. ASPLOS, 2010.

APPENDIX

- **Effect of Temperature:** In our experiments, the curve for normalized retention time at different temperature corresponds to $e^{-0.0625T}$, where T is the temperature. A $10\,^{\circ}\mathrm{C}$ increase in temperature results in a reduction of $1 - e^{-0.0625*10} = 46.5\%$.
- **Probability of a New Failure with Rounds:** The probability of finding a new error with r rounds of tests is calculated as,
probability of a new bit failure at round r
$= \frac{\text{number of new cells failing at round } r}{\text{total number of cells}} \ldots (1)$
- **Probability of Failure with ECC:** Error correction codes are associated with blocks of data. All our results assume that ECC protects 8B of data. We calculate the probability of failure in n bits from the bit error rate, where $n = 64$ bits. If p is the probability of bit failure, probability of failure in n bits, p_n
$=$ probability of any of the n bits failing
$= 1 -$ probability of all of the n bits not failing
$= 1 - (1-p)^n \ldots (2)$
The probability of failure with *single error correction, double error detection* (SECDED) in n bits is calculated as, $p_{nSECDED}$
$=$ probability of more than one bit in n bits failing
$= 1 -$ (probability of all of the n bits not failing $+$
probability of $n-1$ of the n bits not failing)
$= 1 - ((1-p)^n + \binom{n}{1}p(1-p)^{(n-1)}) \ldots (3)$
Similarly, the probability of failure with *double error correction, triple error detection* (DECTED) in n bits is calculated as, $p_{nDECTED}$
$=$ probability of more than two bits in n bits failing
$= 1 -$ (probability of all of the n bits not failing $+$
probability of $n-1$ of the n bits not failing $+$
probability of $n-2$ of the n bits not failing)
$= 1 - ((1-p)^n + \binom{n}{1}p(1-p)^{(n-1)} + \binom{n}{2}p^2(1-p)^{(n-2)}) \ldots (4)$
- **Expected Number of Multi-Bit Failures:** The expected number of faulty bits per word is $p * n$. If $p * n \ll 1$, then the probability (P_k) that the word has k errors $(k \geq 1)$ can be approximated by $P_k = (pn)^k/k! \ldots (5)$
- **Time to Failure with Bit-Repair:** Let the probability of a bit failing be p and the probability of the system failing be p_{system}. If there are n bits in the module, then, p_{system}
$= 1 -$ probability of no error in the entire module
$= 1 - (1-p)^n$
The expected number of trials to fail the system, $t = 1/p_{system}$. In our infrastructure, there are np bit failures at every minute. Therefore, Time to Failure in hours$= t/60 \ldots (6)$
- **Time to Failure with ECC:** Let the system can correct $k-1$ bits per word, the probability of k or more bit failure is p_k and the probability of the system failing is p_{system}. if there are n words in the module, then, p_{system}
$= 1 -$ probability of no k or more bit error in n words
$= 1 - (1-p_k)^n$
The expected number of trials to fail the system, $t = 1/p_{system}$. In our infrastructure, we test for error every minute. Therefore, Mean Time to Failure in hours$= t/60 \ldots (7)$
- **Time to Test a Module:** To test a module, we need to write a data pattern in the entire module, wait for 64 ms and read the module to detect the failures. In order to read/write a row, we need to read the data from the cells in the row buffer (t_{RCD}), transfer the row to the memory controller, and close the row (t_{RP}). According to DDR3-1600 timing, time to read/write an $8KB$ row, $t_r = t_{RCD} + t_{CCD} * (8KB/64B) + t_{RP} = 13.75 + 5 * 128 + 13.75 = 667.5$ ns. In a 2GB module, there are 262144 rows, so reading/writing the entire module would take $t_r * 262144$ ns = 174.98 ms. So, to test a module for a pattern it would take $174.98 + 64 + 174.98$ ms = 413.96 ms. We test the modules with five data patterns, so each round with all data patterns takes 2.06 s. $\ldots (8)$