Characterizing and Mitigating Soft Errors in GPU DRAM

Michael B. Sullivan
Nirmal Saxena
Mike O'Connor
misullivan@nvidia.com
nsaxena@nvidia.com
moconnor@nvidia.com
NVIDIA
Santa Clara, California, USA

Donghyuk Lee
Paul Racunas
Saurabh Hukerikar
donghyukl@nvidia.com
pracunas@nvidia.com
shukerikar@nvidia.com
NVIDIA
Santa Clara, California, USA

Timothy Tsai
Siva Kumar Sastry Hari
Stephen W. Keckler
timothyt@nvidia.com
shari@nvidia.com
skeckler@nvidia.com
NVIDIA
Santa Clara, California, USA

ABSTRACT

GPUs are used in high-reliability systems, including high-performance computers and autonomous vehicles. Because GPUs employ a high-bandwidth, wide-interface to DRAM and fetch each memory access from a single DRAM device, implementing full-device correction through ECC is expensive and impractical. This challenge is compounded by worsening relative rates of multi-bit DRAM errors and increasing GPU memory capacities. This paper first presents high-energy neutron beam testing results for the HBM2 memory on a compute-class GPU. These results uncovered unexpected intermittent errors that we determine to be caused by cell damage from the high-intensity beam. As these errors are an artifact of the testing apparatus, we provide best-practice guidance on how to identify and filter them from the results of beam testing campaigns. Second, we use the soft error beam testing results to inform the design and evaluation of system-level error protection mechanisms by reporting the relative error rates and error patterns from soft errors in GPU DRAM. We observe locality in the multi-bit errors, which we attribute to the underlying structure of the HBM2 memory. Based on these error patterns, we propose several novel ECC schemes to decrease the silent data corruption risk by up to five orders of magnitude relative to SEC-DED ECC, while also reducing the number of uncorrectable errors by up to 7.87×. We compare novel binary and symbol-based ECC organizations that differ in their design complexity, hardware overheads, and permanent error correction abilities, ultimately recommending two promising organizations. These schemes replace SEC-DED ECC with no additional redundancy, likely no performance impacts, and modest area and complexity costs.

ACM Reference Format:
Michael B. Sullivan, Nirmal Saxena, Mike O'Connor, Donghyuk Lee, Paul Racunas, Saurabh Hukerikar, Timothy Tsai, Siva Kumar Sastry Hari, and Stephen W. Keckler. 2021. Characterizing and Mitigating Soft Errors in GPU DRAM. In *MICRO'21: 54th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO '21), October 18–22, 2021, Virtual Event, Greece.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3466752.3480111

MICRO '21, October 18–22, 2021, Virtual Event, Greece

ACM ISBN 978-1-4503-8557-2/21/10…$15.00
https://doi.org/10.1145/3466752.3480111

1 INTRODUCTION

Business-critical servers, datacenters, autonomous vehicles, and high performance computing systems typically rely on commodity hardware with error checking and correcting (ECC) codes applied to large memory structures to increase hardware reliability. GPU acceleration in these systems is critical for high performance and energy efficiency, making GPU resilience a priority. Modern compute-class GPUs and accelerators, such as GPUs [3, 5, 11], Google TPU [51], Fujitsu A64FX [21], and SiPearl Rhea [31] employ wide on-package HBM2 memory to sustain high main memory bandwidths.

Main memory is the largest and most vulnerable storage structure in most systems, making effective DRAM ECC central to any comprehensive error protection scheme. Memory DIMMs in most CPU nodes are composed of narrow 4b wide DRAM devices, meaning that soft errors in these memories are typically confined to the 4B of data coming from a single DRAM chip. Tailored "chipkill" CPU ECC codes are able to detect and correct the data coming from an entirely-faulty 4b DRAM device using the 12.5% of available ECC redundancy, offering high soft error protection while remaining oblivious to underlying error patterns [15, 39]. In contrast, GPU memory is much wider (64b wide per HBM2 pseudo-channel), and each memory entry is fetched from a single DRAM device, making whole-device protection impossible without a prohibitively large memory access granularity or additional redundancy. Understanding the underlying data corruption patterns is therefore crucial for HBM2 memory, as an effective ECC code must be tailored to detect or correct the most prevalent multi-bit errors. GPUs reportedly employ single-bit-error-correcting and double-bit-error-detecting (SEC-DED) ECC for DRAM [2, 52], which we find to be suboptimal for HBM2. This paper demonstrates that tailoring the code to the most prevalent multi-bit soft error patterns can provide vast resilience improvements over SEC-DED ECC for GPUs.

This paper presents high-energy neutron beam testing results for the HBM2 memory on a compute-class GPU with the primary goal of tailoring error protection to the most prevalent soft error patterns. Our neutron beam tests uncovered unexpected intermittent errors in DRAM, however, that we determine to be caused by cell damage from the high-intensity beam. We first provide guidance for beam-testing campaigns on how to identify and filter out these intermittent errors. This guidance is important for HBM2, as it resides on-package and is forced in the beam, even if main memory is not the intended target of study. Having filtered out the intermittent errors from our results, we report the relative error rates and error patterns from soft errors in GPU DRAM. We observe locality in the multi-bit errors, which we attribute to the underlying structure of

the HBM2 memory. We then propose specific improvements over SEC-DED ECC for GPU DRAM, vastly improving the uncorrectable error rate and increasing resilience against silent data corruption (SDC). The main contributions of our paper are:

- **Beam Testing Guidance:** We have identified a physical effect called *displacement damage* as the cause of intermittent errors we observe in beam-tested DRAM. Energetic radiation can physically damage the DRAM access transistor, increasing cell leakage current and reducing the retention time of the DRAM cell by orders of magnitude. The weakened cells cause randomly-distributed single-bit errors (up to several thousand per GPU), and they can be separated from soft errors through post-processing or by enabling GPU DRAM ECC. Displacement damage does *not* happen in the field, and thus the intermittent errors do not need to be addressed by an error protection scheme.
- **Soft Error Patterns:** Our high-energy neutron beam tests show soft errors in HBM2 to be rare but severe—~31.5% of single event upsets (SEUs) in device memory affect multiple bits in at least one word. These severe errors likely originate in DRAM logic structures, and the majority of these severe errors (~75%) are confined to a logically-contiguous byte in each memory word. This tendency towards byte-aligned errors follows intuitively from the observation that HBM2 is deeply hierarchical, with data likely residing in mats with an 8b access granularity.
- **Improved HBM2 ECC:** We observe multi-bit DRAM errors to be heavily structured and amenable to a tailored DRAM ECC code. We investigate binary and symbol-based ECC codes, ultimately proposing two organizations. DuetECC/TrioECC offers a drop-in replacement for SEC-DED ECC, operating within the same hardware footprint and maintaining single-bit and single-pin correction. DuetECC and TrioECC both decrease the SDC risk relative to SEC-DED, and they expose a correction/SDC trade-off using a reconfigurable decoder. DuetECC decreases the SDC risk by over three orders of magnitude. TrioECC performs more aggressive correction, reducing the number of uncorrectable errors by 7.87× and decreasing the SEC-DED silent data corruption risk by two orders of magnitude. We also propose an alternate symbol-based ECC code called SSC-DSD+, which further reduces the SDC risk over DuetECC by two more orders of magnitude, while simultaneously approaching the correction rate of TrioECC. SSC-DSD+ is a promising organization if a larger departure from SEC-DED ECC is permissible—it has a larger and slower decoder, and it sacrifices the ability to correct permanent pin failures.

2 BACKGROUND

2.1 Neutron Beam Testing

Soft errors are non-destructive events that corrupt memory until a following write. Such errors are concerning, as they pose a significant risk of silent data corruption [8, 61]. Energetic neutrons produced by cosmic rays are the dominant soft error cause in the terrestrial environment. Neutron beam testing offers a way to perform accelerated testing to judge the terrestrial soft error rate and error patterns [32]. Neutron beam tests have been performed on GPUs and other accelerators in the past, with a focus on the logic die [17, 46, 54]. In contrast, our focus in this paper is on the on-package HBM2 memory of a GPU.

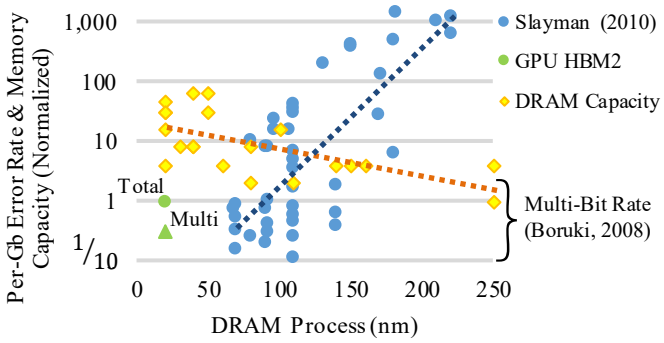


Fig. 1: Historical neutron beam testing data and memory capacities over process generations. Exponential regressions of the historical data are shown with dotted lines.

2.2 GPU DRAM Organization

Modern compute-class GPUs [3, 5, 11] rely on high-bandwidth on-package HBM2 memory to support many concurrent DRAM accesses. The minimum access granularity of HBM2 memory is 32B [36]. In this paper, we assume that GPUs also support a fine-grained 32B accesses granularity; this is corroborated by prior microbenchmarking work on NVIDIA GPUs [34] as well as work that shows fine-grained GPU accesses to be more performant for important workloads [57]. We refer to each 32B block in HBM2 as a *memory entry*, such that each read fetches a single entry.

2.3 Soft Error Trends in DRAM

Trends over the past two decades have led to the situation where DRAM errors are rare, but relatively broad and severe when they occur. DRAM error rates have fallen during this time due to shrinking memory technologies with roughly constant capacitance per DRAM bitcell, meaning that the charge required to flip a bitcell is constant but the chance of a cell strike has fallen with memory scaling. The falling rate of per-bit errors has historically outweighed the DRAM capacity increases, leading to a gently-decreasing overall error rate.[1] Figure 1 shows the historically-falling DRAM error rate alongside the DRAM capacity increases and our measured HBM2 error rate. Prior neutron beam testing data (shown in blue) for older-generation DRAM are taken from [60], and the DRAM capacities are taken from a list of various DRAM vendors and generations [69]. Exponential regressions of the historical data are shown in dotted lines; they demonstrate a decrease in the per-chip DRAM failure rate that outpaces the increase in DRAM capacities.

Logic errors in DRAM have not scaled like those in bitcells—as DRAM supply voltage has decreased, the rate of potentially-severe *non-bitcell* DRAM errors has stayed roughly constant or increased slowly [6, 60]. This makes the *relative* contribution of severe and broad non-bitcell errors more pronounced [55]. We use the term *breadth* to indicate the number of 32B memory entries that can be affected by a single fault, and *severity* to indicate the number of bits per entry that can be affected. Boruki *et al.* observe that the non-bitcell upset rate stays within a two-order-of-magnitude range,

[1] With recent DRAM generations, the capacitance per bitcell is falling, meaning that this historical decrease no longer holds [26].

and it does not show any strong trend with technology scaling [6]; we mark this range using a bracket to the right of Figure 1. We overlay the GPU HBM2 error rate we measure through neutron beam testing as a green circle, and the multi-bit error rate as a green triangle. The low error rate of HBM2 and the high relative multi-bit error rate are within expectations given the historical trends.

Field studies of large-scale systems have also reported high error breadth and severity, both in server-class main memory [42, 62–64] and in GPU memory [16, 50, 66, 67]. Field studies differ from neutron beam testing in several regards. They contain error contributions both from soft errors and from other sources, such as permanent errors due to marginal memory devices that escape fabrication-time testing. Field studies also cannot examine the rate or error pattern of silent data corruption, since it is never present in the error logs.

2.4 The Structure of HBM2 Memory

Modern DRAM is complex and deeply hierarchical, with components that can fail at multiple granularities and affect a varying number of bits per memory entry. Each HBM2 stack is composed of eight 512MB channels, each with separate data, control, and power pins.[2] Each HBM2 channel is split into 16 banks that share the channel pins. DRAM is accessed by a split row and column address—a *row activation* command brings 2KB of data from the DRAM bit-cells into a bank-local row buffer, and following read and write commands access one 32B column of the row buffer at a time. There is not a single row buffer per bank, but rather each bank is composed of 32 *subarrays*, each with its own row buffer. A 32B DRAM read draws all of its data from a single subarray, and only one subarray is selected at a time. Each subarray is further split into 32 *data mats*, each of which fills an 8b wide slice of the 2KB row buffer. The DRAM bit-cells reside within these data mats, and each mat is composed of a 512×512 array of bits. Rows of the mat are activated by *local word lines* and each bit of an activated row is transmitted to the 512b mat-local row buffer slice through a *bitline*. Each DRAM read selects an 8b column from each mat through a local column mux.

2.5 Permanent DRAM Errors

Each HBM2 stack connects to the GPU through 1024 data wires on a silicon interposer. Signals pass to and from the silicon interposer through microbumps. The dimensions of the constituent microbumps and wires are tiny [56], and single-pin permanent failures are an important failure mode for HBM2 in the field. Pin failures can develop due to cracking microbumps or marginal joint defects, each of which can develop weeks or longer after the GPU leaves the factory [37, 71]. Single-pin correction is therefore desirable, as it allows a GPU to gracefully degrade in the field, making error diagnosis and the scheduling of GPU replacements less difficult and potentially reducing downtime. Accordingly, we take care to preserve single-pin correction in some ECC schemes, despite the fact that pin errors are not prevalent in our soft error patterns.

Field studies have also reported permanent non-pin errors with patterns that are similar to those of soft errors [62–64]. While our primary focus is on precisely quantifying the soft error risk, our ECC improvements also give corresponding benefits for permanent errors with similar patterns—for instance, byte detection and correction are important for permanent local wordline failures.

[2]To be more precise, each channel has separate through-silicon-vias (TSVs) that eventually lead to microbumps that are wired through an interposer to the GPU; we refer to the TSV, microbump, and wire as a "pin."

2.6 Error Correcting Codes

An error checking and correcting (ECC) code detects and possibly corrects errors using redundant values that are algorithmically generated from the protected data. An (N, K) code has N total symbols with K information symbols and $R = (N - K)$ check-symbols. We consider ECC schemes that use binary symbols, 2b symbols, and 1B symbols in this paper. The worst-case error detecting and correcting capabilities of an ECC code are often characterized by the maximum number of bytes or bits that it can detect and correct. An error that is within the theoretical correction capabilities of an ECC code will result in a detected-and-corrected error (DCE), and those that are within the detection capability will be a detected-yet-uncorrected error (DUE). Errors that exceed the theoretical coverage of a code can either be detected (resulting in a DUE) or lead to silent data corruption (SDC).

HBM2 memory devotes 12.5% redundancy for ECC check-bits, and compute-class GPUs reportedly apply a single-bit-error-correcting and double-bit-error-detecting (SEC-DED) code to DRAM [2, 52]. These codes are fast and efficient, but they work best for isolated direct-cell strikes and logic errors can exceed their maximum detection and correction capabilities. Sections 6 and 7 investigate the design space of both binary and symbol-based ECC codes, ultimately proposing two low-overhead organizations that decrease the SDC risk of SEC-DED ECC by many orders of magnitude, and increase the correction rate by up to 7.87×.

Server-class CPUs commonly employ DRAM ECC codes that can correct the data coming from a whole DRAM device [1, 15, 27, 29]. Whole-device correction is possible for server memory because DIMMs in CPU nodes are often composed of narrow 4b wide DRAM devices, meaning that a symbol-based ECC code (such as a Reed-Solomon code with 8b symbols) can correct the data coming from an entirely-faulty chip [1, 22, 39]. Numerous recent works [10, 33, 35, 48, 49] have investigated *whole-die* correction for stacked DRAM (such as HBM2) with the expectation that it is analogous to whole-device correction in server memories. These approaches rely on multi-tiered ECC, where error correction uses a second ECC check-bit pool in a separate DRAM channel. Our results indicate that soft errors do *not* span an entire HBM2 DRAM die, and these approaches are expensive and microarchitecturally complex due to their need for channel-to-channel communication, write amplification, and additional redundancy. Accordingly, we only consider single-tiered ECC codes in this paper (similar to the existing SEC-DED protection in current GPUs), and do not focus on HBM2 die correction, as it would not give any additional protection for the soft errors we observe through neutron beam testing.

3 METHODOLOGY

Neutron Beam Testing Details: Results come from two neutron beam testing campaigns at the ChipIR beamline of the ISIS Neutron and Muon Source [20]. ChipIR is purpose-built for microelectronics reliability testing, with neutron energies that are tuned to resemble

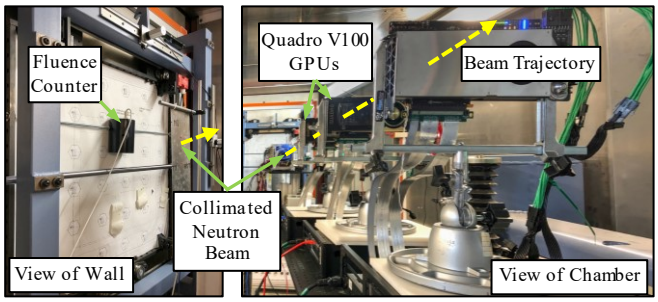


Fig. 2: A picture of our neutron beam testing setup.

the terrestrial energy spectrum [9]. We test NVIDIA Quadro V100 compute-class GPUs, each with 32GB of HBM2 DRAM. We use results from one campaign for intermittent error experiments and the other for soft error patterns. Within each campaign, we see no statistically-significant differences between different Quadro V100 GPUs, and present their results combined. Figure 2 shows the experimental setup. Our beam testing experiments take place with the GPU running at full voltage and speed, using the full NVIDIA software stack. The host machine is a Xeon-based server with ECC-protected on and off-chip memories. Furthermore, the host machine is separated from the GPU (and thus the radiation) by a 500mm PCIe extender cable. The average flux during our DRAM experiments was 9.8e5 neutrons/cm^2/second, representing an acceleration factor of 2.52e8 over the terrestrial neutron flux of 14 neutrons/cm^2/hour at sea level in New York City [32].

Accelerator DRAM Beam Testing Methodology: We run a targeted CUDA microbenchmark to observe and quantify the effects of DRAM errors in on-package HBM2 DRAM. The DRAM microbenchmark writes a known pattern to every memory entry and reads back the memory repeatedly, recording and time-stamping any mismatches in duplicated pinned host memory (and not on the device). The outer write loop executes 10 times per run and the inner loop reads 20 times per write. The microbenchmark streams through DRAM with no cache reuse, and uses duplicated execution, duplicated logging, on-chip SRAM ECC, and programmer assertions to guard against errors outside of DRAM; the $\frac{11}{1830}$ (0.60%) of runs that fail these checks are discarded. We run the microbenchmark with the GPU DRAM ECC disabled to prevent frequent DUEs from crashing the experiment and to give us visibility into severe error patterns that may be misclassified as single or double-bit errors by the ECC hardware. We leave the GPU SRAM ECC enabled using a modified GPU BIOS.

We run experiments using three data patterns. The first pattern writes all 0s (or all 1s) to memory, the second writes a pseudo-checkerboard pattern of 0x555..., 0xAAA..., the third pattern writes an AN-encoded data value [7] to each 8B word, representing the index of that word in the virtual memory space $\times 2^{32} - 1$. We use the AN-encoded pattern because it offers a less-synthetic mix of 1s and 0s within each codeword. For each of these data patterns, we alternate write cycles between writing the data and its inverse in order to properly diagnose unidirectional intermittent errors.

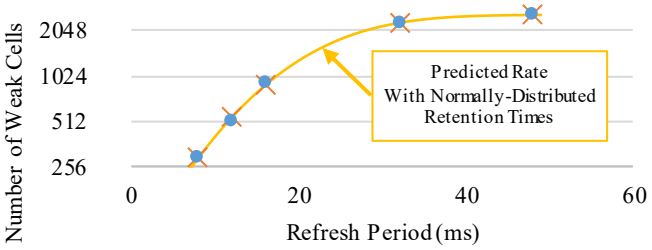
Intermittent Error Exploration Methodology: We investigate the intermittent HBM2 errors that develop and persist outside of the beam using two experiments. The first continually runs the DRAM microbenchmark during the beginning of GPU exposure to track the accumulation of intermittent errors with cumulative fluence. The second removes one GPU from the beam and compares the effect of varying DRAM refresh rates on the number of observable intermittent errors. We modulate the DRAM refresh rate using a modified GPU BIOS. Before beam-testing the GPUs, we first verify that they run error-free overnight with a 48ms refresh period (the lowest used in our experiments), meaning that all intermittent errors are due to beam exposure.

4 INTERMITTENT ERRORS IN HBM2

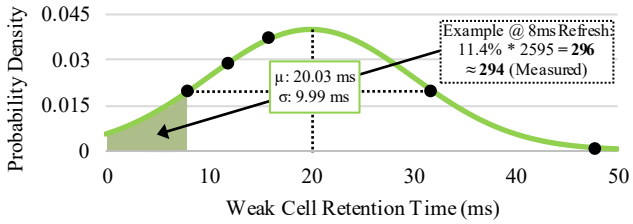
We have strong evidence that a physical effect called *displacement damage* is the cause of the observed intermittent errors in DRAM. Energetic radiation can physically damage the DRAM access transistor, increasing cell leakage current [12, 13, 19, 59]. This increased leakage reduces the retention time of the DRAM cell by orders of magnitude, and the degraded retention capabilities can persist outside of exposure to radiation [58, 59]. Displacement damage has previously been observed for DDR3 DRAMs [43] and DDR4 DRAMs (in a similar process technology as HBM2) [55]. We settled on the hypothesis of displacement damage for the HBM2 DRAM because of the speed with which the errors occur—the cumulative fluence exposure of the DRAM is not nearly enough to cause other radiation-induced faults such as those related to total-ionizing-dose [68]. Our beam testing experiments support this hypothesis, and we we characterize the refresh impact of displacement damage, describe the proper methodology to filter out these effects in post-processing, and draw some practical conclusions below.

Retention Errors: We expect displacement-damaged cells to exhibit reduced retention capabilities. To test weak cell retention, we run the microbenchmark outside the beam on a heavily damaged GPU while modulating the DRAM refresh rate. Figure 3a uses blue markers to show the measured weak cell counts when modulating the DRAM refresh rate. The number of weak cells monotonically increases with increasing refresh period, as we expect from a refresh-related effect. The rate of this increase is stronger at low refresh periods; this trend can be explained if the weak cell retention times are normally distributed. Figure 3b shows a normal curve that closely fits the data, formed using non-linear least squares regression. We then cast this normal curve back to weak cell counts using its CDF; the results are overlaid in Figure 3a as an orange line. The predicted weak cell counts at the measured refresh periods are shown with orange "X" marks; the predictions agree closely with our measurements.

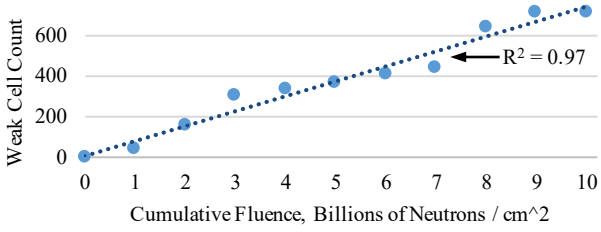
Retention Error Behaviors: We expect retention-related errors to be single-bit and unidirectional, as they correspond to a bitcell capacitor leaking before the memory entry is refreshed. The logical direction of this unidirectional error depends on whether a given bitcell is stored active high or low. This differs between vendors—for example, Lim *et al.* studied the effects of displacement damage on memory from four anonymous vendors, and found that two of the vendors exhibited only $1 \rightarrow 0$ logical errors, whereas the other two vendors exhibited both $1 \rightarrow 0$ and $0 \rightarrow 1$ errors, depending on bit position[43]. In our error experiments, 99.8% (±0.16%) of the intermittent errors were always in the $1 \rightarrow 0$ direction, possibly indicating that this memory stores the logical values directly to the bitcells. While we need to further investigate the cause of the few



(a) Intermittent Error Rate vs. the Refresh Period



(b) Normally Distributed Refresh Times Fit to the Data



(c) The Accumulation of Weak Cells

Fig. 3: Intermittent error experiments: (a) weak cell counts when modulating DRAM refresh rates, (b) normally-distributed weak cell retention times fit to the data, (c) the accumulation of weak cells with cumulative exposure.

intermittent errors that exhibit $0 \rightarrow 1$ transitions, we suspect that they are due to displacement-damage related leakage current in non-bitcell structures. We reproduce prior findings on weak cells that report single bit errors, with no obvious pattern, and no overlap between soft error and weak cell locations [55].

Weak Cell Accumulation: Rodriguez *et al.* observe that the weak cell count is linearly proportional to the total neutron exposure of the DRAM [59]. We plot the cumulative count of intermittently-classified errors at the beginning of our beam testing campaign in Figure 3c. The trend agrees with the previous characterization—a linear regression of the data ($R^2 = 0.97$) is shown with a blue dotted line. We do *not* see the linear weak cell count increase indefinitely with beam exposure, but rather the increase eventually asymptotes to roughly a thousand cells with a 16ms refresh period, as can be seen in Figure 3a. Prior work has noted that DRAM cells can be separated into *leaky* and *strong* cells [40]. One hypothesis is that we observe linear scaling in the number of weak cells until all leaky cells are weak, after which point weak cell accumulation slows.

Error Annealing: We observe partial annealing over time, which is consistent with prior reports [43, 44, 59]. Roughly three and a half hours before we ran the refresh rate experiment above, we ran a trial experiment with refresh periods of 8ms and 48ms. (The machine sat outside of the beam in the intervening period.) The number of weak cells with an 8ms refresh period fell from 397 during the trial to 294 in the full experiment (a 26% decrease). However, the 48ms refresh period shows both a smaller relative and absolute decline from 2,656 weak cells to 2,589 (only a 2.5% decrease). The normally-distributed cell retention time model from above can explain this refresh-period dependence—if the retention time curve shifts upon annealing, we expect a relatively large decrease with short retention periods and a much smaller decrease at large ones.

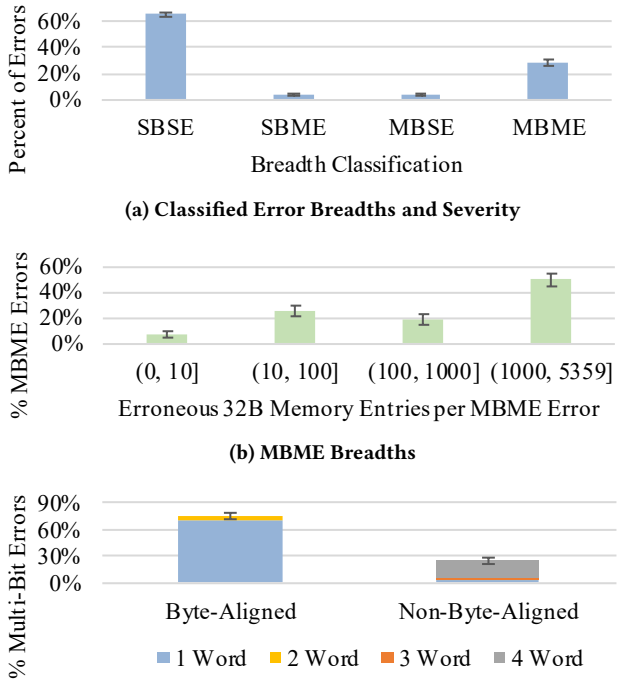
Post-Processing: Our experiments show that even a heavily-damaged GPU contains only roughly a thousand weak cells out of 32GB of DRAM (with the default 16ms HBM2 refresh period). This, combined with the isolated single-bit nature of displacement damage, means that the intermittent errors have a vanishingly small probability of overlapping with soft errors in the beam. This allows us to filter out the intermittent errors by classifying any memory entry with repeated errors as damaged, and ignoring any error contribution from these damaged entries. We never see any broad soft error overlap with a damaged memory entry.

Overall Impact of our Findings: The isolated, single-bit nature of displacement damage means that single-bit error correction can correct all of the intermittent errors, and beam testing campaigns that are not focused on main memory need not model an SDC contribution from damaged DRAM cells if ECC is enabled. The relatively low weak-cell counts allow us to post-process out the intermittent errors by classifying any memory entry with repeated errors as damaged. The gradual weak cell accumulation rate and annealing speed suggest that displacement damage will *not* be a field effect for HBM2 devices at terrestrial flux rates, and thus the intermittent errors do not need to be addressed by an error protection scheme.

5 MEASURED SOFT ERROR PATTERNS

This section presents the soft error patterns we observed in HBM2 memory after filtering out any intermittent errors. Figure 4 gives high-level information on the error severity and breadth we observe. The All0/All1, pseudo-checkerboard, and AN-encoded error patterns have overlapping error bars in these metrics so we present them combined. The mean-time-to event in the beam is in seconds while HBM2 memory only takes milliseconds to perform a single read or write loop. Events that hit during different loop iterations will not be classified as a single multi-bit upset, avoiding any significant risk of multiple events being classified as a single broad-and-severe error.

Error Breadth: Figure 4a gives the relative breakdown of the breadth and severity of each error based on four classes. (1) SBSE: Single-Bit, Single-Entry; (2) SBME: Single-Bit, Multiple-Entry; (3) MBSE: Multiple-Bit, Single-Entry; and (4) MBME: Multiple-Bit, Multiple-Entry. Isolated single-bit events represent 65% ± 2.3% of the errors. Multiple-bit, multiple-entry (MBME) events are the second most common (28% ± 2.1%), and they can be quite broad. Figure 4b shows the number of 32B memory entries affected by each MBME error, grouped into bins of exponentially-increasing sizes. The breadth of MBME errors is a long-tailed distribution, with the most broad error affecting 5,359 memory entries.

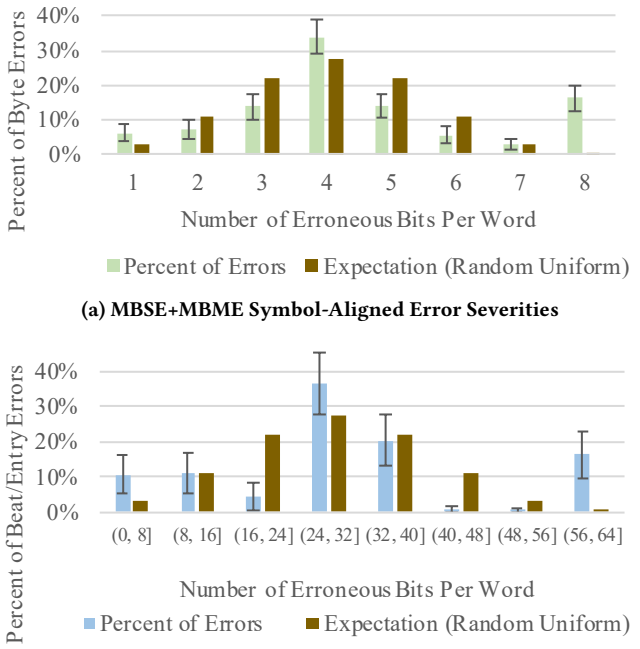


(a) Classified Error Breadths and Severity

(b) MBME Breadths

(c) MBSE+MBME Symbol Alignment and Words per Entry

Fig. 4: High-level information on the severity and breadth of the observed HBM2 soft errors. SBSE: Single-Bit, Single-Entry; SBME: Single-Bit, Multiple-Entry; MBSE: Multiple-Bit, Single-Entry; MBME: Multiple-Bit, Multiple-Entry.



(a) MBSE+MBME Symbol-Aligned Error Severities

(b) MBSE+MBME Non-Symbol-Aligned Error Severities

Fig. 5: Multi-bit error severity in bits per word.

Error Severity: The severity of multi-bit (MBSE+MBME) errors is also a concern. To reason about their severity, we split the multi-bit errors into two classes that are illustrated in Figure 4c. Byte-aligned multi-bit errors are the most common, representing 74.6% ± 3.8% of multi-bit errors. These byte-aligned errors can affect many 32B entries, but within each 64b word the error is confined to an aligned byte of memory. Non-byte-aligned errors are less frequent (25.4% ± 3.8% of multi-bit events), and they can affect up to all 64b data in a word of memory.

The tendency towards single-bit and byte-aligned errors follows intuitively from the structure of DRAM with the observation that most soft errors are mat-local. Because HBM2 supports per-byte write enables,[3] it is likely that logically-contiguous bytes of transmitted data are mapped directly to the 8b data mats with per-mat enable signals. Thus each byte in a 32B HBM2 read (with 4B ECC) likely comes from its own mat. Any mat-local failure is therefore confined to a logically-contiguous byte of a memory access.

Figure 5 shows the severity of multi-bit (MBSE+MBME) errors in terms of the number of affected bits per erroneous word. Byte-aligned errors (Figure 5a) and non-byte-aligned errors (Figure 5b) show similar error patterns. Flipping half of the bits in a byte or word is the most common error modality, with probabilities falling towards the extremes. The distribution of error severities matches the random-corruption expectation reasonably well, as shown by the brown bars. There *is* an anomalous tendency for ~15% of errors to flip *all* of the bits in a byte or word. These inversion errors are data-dependent, affecting the All0/All1 data pattern more frequently than the other patterns. We choose uniform random corruption as the error model for evaluation in the remainder of the paper. Erroneous byte or word inversions are generally simpler for ECC to detect and correct than random errors, and we do not want to make an assumption about the data in memory so we conservatively choose the harder error pattern. The random error model also closely matches prior ECC evaluation methodologies [23, 24, 39].

Words Per Entry: Figure 4c also shows the number of words affected per erroneous entry, as stacked bars. Byte-aligned errors tend to affect only a single word per entry (and occasionally two), whereas non-byte-aligned errors tend to affect all four words in the entry but occasionally are confined to a single word.

Effect of DRAM Utilization: If multi-bit errors are symptoms of erroneous DRAM logic, then the rate of MBME errors could depend on the memory utilization of a program. By sweeping the microbenchmark DRAM utilization, we confirm that the fraction of broad-and-severe logic errors (MBSE+MBME) is proportional to the number of memory accesses, while narrow array errors (SBSE+SBME) are proportional to the exposure time. This provides support to the hypothesis that our observed multi-bit errors originate in DRAM logic structures, rather than from direct cell strikes.

Proposed Error Model for Evaluation: Our beam testing results show some error patterns are much more prevalent than others. These patterns should be taken into account when designing and evaluating the system-level error protection mechanisms of GPUs and other accelerators. We propose a simple analytical error model for ECC evaluation that weights the random bit, byte, beat, pin, and whole-entry errors with probabilities drawn from the beam

[3] Per-byte write enables are only supported in HBM2 ECC disabled, but it is likely that the data layout is the same with ECC on or off.

Tab. 1: Soft Error Pattern Probabilities.

Severity	Bits	Probability
1 Bit	1	73.98%
1 Pin	2–4	0.19%
1 Byte	2–8	22.56%
2 Bits	2	0.11%
3 Bits	3	0.03%
1 Beat	4–64	0.90%
1 Entry	4–256	2.23%

testing data. We classify each error according to these 7 patterns, with their probabilities shown in Table 1. Patterns are sorted in increasing ECC difficulty for correction, and priority is given to less-difficult errors whenever multiple patterns fit—for example, a 2b error can be thought of as any error with 2 erroneous bits that are *not* in the same byte or pin.

6 IMPROVEMENTS TO HBM2 ECC

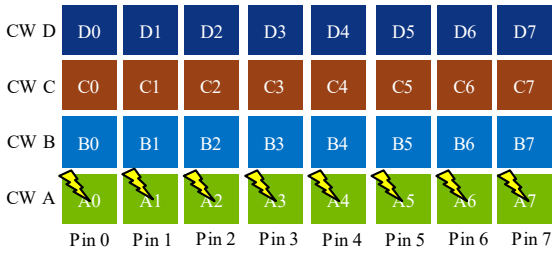
The detailed HBM2 error patterns from beam testing allow us to judge the efficacy of different GPU ECC codes. We investigate the design space of both binary and symbol-based ECC codes, considering ECC schemes that use binary, 2b, and 1B symbols. As part of this investigation, we describe three complementary optimizations below that can combine to create a low-cost drop-in replacement for SEC-DED ECC called DuetECC/TrioECC. We also expand the state-of-the-art in byte-correcting Reed-Solomon codes by describing a fast and efficient scheme we call SSC-DSD+ that offers higher resilience against soft errors, albeit with a larger hardware overhead and without the pin error correction offered by SEC-DED ECC.

6.1 Binary ECC Code Designs

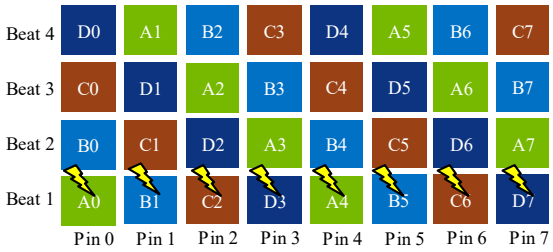
SEC-DED Baseline: As a binary ECC baseline code, we consider a SEC-DED implementation utilizing a minimum-odd-weight Hsiao code ("(72, 64) SEC-DED version 1" from [28]). We utilize the 12.5% redundancy by splitting the 36B memory access (32B data, 4B ECC) into four 72b codewords, one per DRAM beat. This closely resembles the conventional way to employ SEC-DED ECC on DIMM-based main memories.

Decoding Behavior with Multiple Codewords: As our baseline uses four codewords per memory entry, it enjoys some additional opportunistic error detection and correction capabilities. If any multi-bit error only affects a single bit per codeword, it can be handled via single-bit correction. Likewise, if a multi-bit error causes a DUE in *any* of the four constituent codewords, the entire memory entry is discarded such that an SDC in any other codeword may be avoided.

Logical Codeword Interleaving: As an optimization, we logically interleave the four codewords from each memory entry such that a byte error is distributed between all four. When combined with SEC-DED ECC, our interleaving pattern provides half-byte error correction and single-byte detection, while preserving single-pin correction. Figure 6 contrasts non-interleaved SEC-DED with this logical interleaving. An example byte error is denoted in the first transmitted burst. Our chosen interleave scheme uses Equations 1 and 2 to swizzle and deswizzle each 32B memory access.



(a) SEC-DED with a Byte Error



(b) Logically Interleaved SEC-DED with a Byte Error

Fig. 6: A comparison of transmission on the first 8 data pins for SEC-DED and logically interleaved SEC-DED. Each data bit is labeled with its index, and a byte error is shown.

Variables I_bits and NI_bits represent the interleaved and non-interleaved bit vectors; the constants 73 and 288 are the codeword size plus one and the memory entry size (including ECC check-bits), respectively. We use $|x|_A$ as shorthand to denote the residue of x modulo A.

$$I\_bits[i] = NI\_bits[|i * 73|_{288}],\ \forall i \in \mathbb{N} : 0 \leq i < 288 \quad (1)$$
$$NI\_bits[|i * 73|_{288}] = I\_bits[i],\ \forall i \in \mathbb{N} : 0 \leq i < 288 \quad (2)$$

This interleaving scheme respects a per-beat rotation between the interleaved codewords to maintain pin correction. This per-beat rotation forms a "checkerboard" pattern that spreads a pin error among single-bit errors in each codeword.

Correction Sanity Check: As a further optimization, we apply a *correction sanity check* to the four interleaved codewords per memory entry. The correction sanity check works by considering the location of corrected errors if there are multiple codewords performing correction. If the corrected bits reside in the same pin or same byte, correction is allowed to proceed as normal. Otherwise, the correction sanity check raises a DUE. This sacrifices opportunistic correction for unexpected multi-cell, non byte-aligned, or non-pin aligned errors, which has next-to-no correction rate impact since such errors are extremely rare in the field. The benefit of the correction sanity check is significantly higher protection against SDC in the presence of a broad multi-byte error, such as the whole-beat or whole-entry errors we observe. This correction sanity check is inspired by similar mechanisms for multi-symbol error correction [1, 39], but we are unaware of work that applies the correction sanity check to interleaved codes to garner its error detection benefit without the cost of multi-symbol error correction.

DuetECC: We refer to the combination of interleaving and the correction sanity check as DuetECC. DuetECC is a safe and conservative code that is able to detect all byte errors and provide high detection capabilities against more severe errors. Ultimately,

it is able to maintain a three order-of-magnitude SDC reduction over the SEC-DED baseline.

2b-Symbol Correction: A third optimization is to migrate from a SEC-DED code to one that can correct 2b symbols. Inspired by prior work in double-adjacent-bit error correcting codes [18], we develop a 2b symbol correcting code using a genetic algorithm. Equation 3 gives the parity check matrix for the SEC-2bEC code, presented in the Crockford Base32 binary-to-text encoding scheme [14]. The code maps aligned 2b errors to unique syndromes, allowing for 2b error correction with only slight modifications to the SEC-DED decoder. Our code differs from [18] in that it is optimized to only correct aligned 2b errors (and not all 2b-adjacent errors), reducing the non-neighboring 2b error miscorrection risk by ~20%. Our chosen code is constrained to operate as a SEC-DED code if 2b symbol correction is not attempted—this allows us to dynamically switch between a SEC-DED or SEC-2bEC code with little additional hardware. Note that we print the code for non-interleaved use (where 2b symbols are bit-adjacent), but when combining with interleaving we swizzle the H matrix and modify the 2b-adjacent correction pattern to compose each 2b symbol of bits that are stride-4 apart.

$$H_{SEC-2bEC}\ (72,\ 64) = \begin{pmatrix} \texttt{2JZXMJP4K6FNWM0} \\ \texttt{0CRW9M5962TJMA0} \\ \texttt{1N9NJ8ZACKPQGH0} \\ \texttt{1B5B40P8S9A8H0G} \\ \texttt{2V3K9DWNJE0Z6G8} \\ \texttt{1ZDTJP8Z0CHGQR4} \\ \texttt{3MMQ5N4E4H1CA02} \\ \texttt{1FEYAZNM9J64DR1} \end{pmatrix} \quad (3)$$

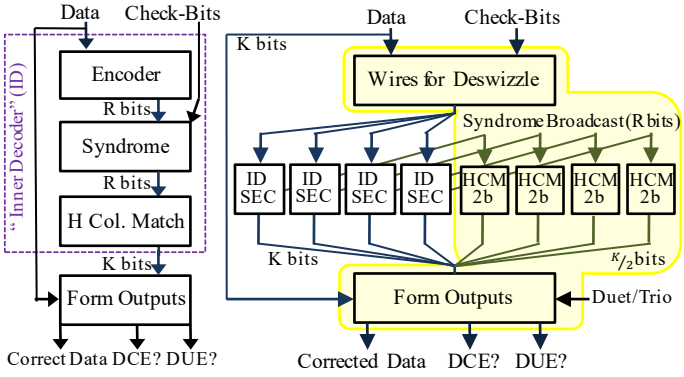
TrioECC: The SEC-2bEC code complements the two other optimizations we describe above. The main advantage SEC-2bEC is its ability to correct all byte errors when combined with codeword interleaving. The main weakness of the SEC-2bEC code is its relatively high risk of severe error miscorrection—a weakness that is greatly diminished by the correction sanity check. We refer to the scheme combining all three optimizations as TrioECC. TrioECC offers a more correction-oriented and aggressive design than DuetECC, providing strong byte error correction while maintaining a two order-of-magnitude SDC reduction over SEC-DED ECC.

DuetECC vs. TrioECC: A fundamental downside of aggressive correction is an increased risk of SDC-causing miscorrection in the presence of severe errors. Thus, the expanded correction capabilities of TrioECC introduce some risk relative to DuetECC, and the two schemes expose a correction/SDC trade-off. We show later how a single decoder can implement *both* DuetECC and TrioECC, allowing system architects to use whichever code best suits their purposes.

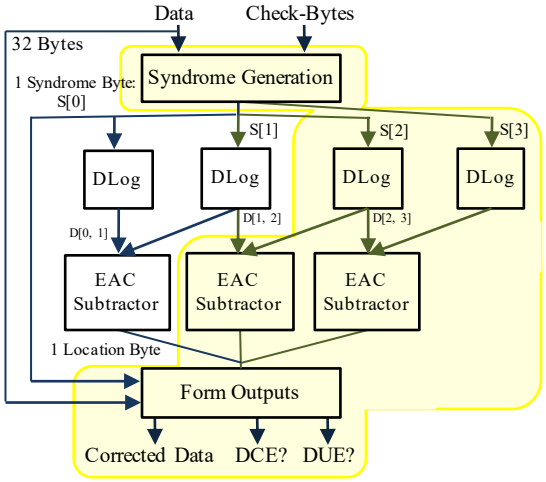
6.2 Symbol-Based ECC Code Designs

The prevalence of observed byte errors also motivates an exploration into symbol-based ECC codes. We explore the design space of Reed-Solomon codes with 8b symbols; these codes can potentially offer byte correction and (in some organizations) pin correction.

Reed-Solomon Baseline: As a symbol-based ECC baseline, we consider an interleaved Reed-Solomon code with two (18, 16) Single-Symbol-Correct (SSC) codewords. This code can provide byte error correction while preserving pin correction by using a 4-pin by 2-beat 8b symbol layout, offering correction capabilities akin



(a) SEC-DED (b) Decoder Changes for DuetECC/TrioECC



(c) Decoder Changes for SSC-DSD+

Fig. 7: The hardware for SEC-DED, DuetECC/TrioECC, and SSC-DSD+ decoders. New or modified structures are highlighted in yellow. HCM: H-Column Match, DLog: Discrete Log, EAC: End-Around-Carry (mod 255).

to TrioECC. Interleaving occurs similar to the description above for binary codes, but at a byte granularity. This code is amenable to the correction sanity check. It also allows the use of a low-cost and low-latency *one-shot* Reed-Solomon decoder design [38], which performs error location in fully-parallel combinational logic.

DSC or SSC-TSD: The 12.5% of available HBM2 redundancy can be used to create a single (36, 32) codeword, offering either DSC (double-symbol correct) or SSC-TSD (single-symbol correct, triple-symbol detect) protection. Both of these codes come with high decoder cost and latency, however, precluding the use of a one-shot Reed-Solomon decoder. The use of either a DSC or SSC-TSD code would require the decoder to solve for the roots of the error locator polynomial, requiring at least 8 cycles based on iterative algebraic decoding procedures [45]. Accordingly, we do not consider these codes appropriate for GPU DRAM, due to the need for fast and efficient ECC decoders.

SSC-DSD+: As an alternative to expensive SSC-TSD decoding, we propose a one-shot-decodable scheme we call SSC-DSD+ that provides single-symbol correct, double-symbol detect, and *almost*

Tab. 2: ECC Scheme SDC Risk. (C=100% Corrected, D=100% Detected)

	SEC-DED (NI:SEC-DED)	I:SEC-DED	DuetECC (I:SEC-DED+CSC)	NI:SEC-2bEC	I:SEC-2bEC	TrioECC (I:SEC-2bEC+CSC)	I:SSC	I:SSC+CSC	SSC-DSD+
# CWs	4	4	4	4	4	4	1	1	1
Symbol	1b	1b	1b	2b	2b	2b	8b	8b	8b
1 Bit	**C**	**C**	**C**	**C**	**C**	**C**	**C**	**C**	**C**
1 Pin	**C**	**C**	**C**	**C**	**C**	**C**	**C**	**C**	D
1 Byte	22.6721%	D	D	39.4062%	**C**	**C**	**C**	**C**	**C**
2 Bits	**D**	**D**	**D**	5.0813%	5.0813%	5.0813%	9.6545%	9.6545%	**D**
3 Bits	3.4080%	3.4080%	3.4080%	14.9347%	14.9347%	4.7010%	16.8407%	3.8781%	**D**
1 Beat	28.5201%	0.6615%	0.0013%	42.2054%	3.1670%	0.0089%	0.4898%	0.0543%	**0.0002%**
1 Entry	0.6640%	0.6603%	0.0013%	3.1646%	3.1643%	0.0085%	0.4898%	0.0543%	**0.0002%**

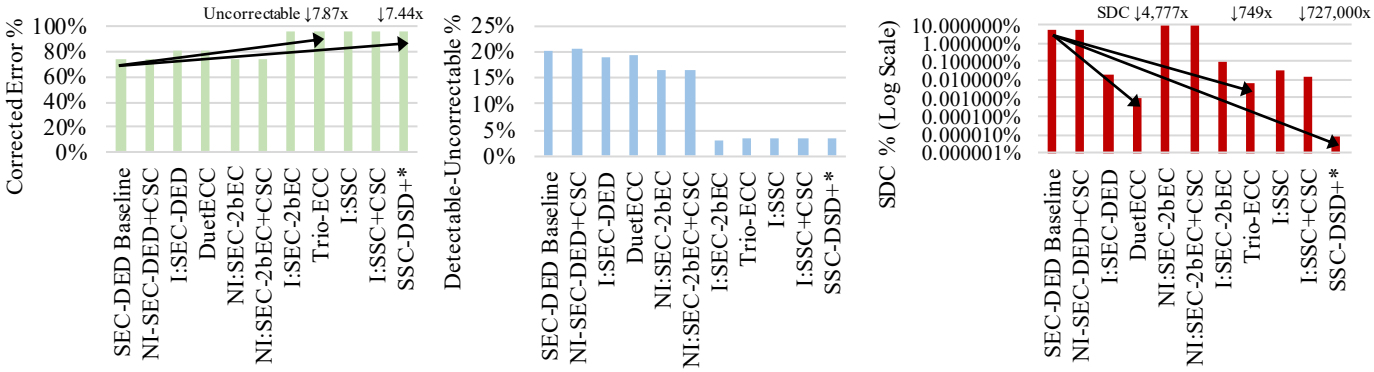


Fig. 8: The correction, detection, and silent data corruption probabilities of each scheme, given a random single event. * - SSC-DSD+ is the only scheme lacking pin error correction.

triple-symbol detection (> 99.999964%) with a single-cycle decoder. Unlike all other considered schemes, SSC-DSD+ does not provide pin error correction. This limitation could be problematic for HBM2 if correction is used to gracefully degrade in the presence of permanent pin errors.

6.3 Hardware Implementations

SEC-DED Hardware: Binary linear block codes are uniquely determined by an $R \times N$ parity-check matrix, "H." The H matrix dictates the structure of the encoder/decoder and the error correction and detection capabilities of the code. To provide single-bit error correction, the constraints on the H matrix are very simple: every column must be unique. To provide double-bit error detection, the XOR of any two columns—used for addition in Galois Field (GF(2)) arithmetic—must not equal any column of the H matrix.

Figure 7a shows the SEC-DED decoder, which first generates a *syndrome* by encoding the received data value and XORing it with the received check-bits. If there is no error, then the generated check-bits equal the received ones and the syndrome is all-0. Upon a single-bit error, the bit from the column that equals the syndrome needs to be corrected. This is communicated through a H-column-match (HCM) signal to the output logic, which either corrects the error or reports a DUE. We call the first three steps the "Inner Decoder (ID)."

DuetECC/TrioECC Hardware: The three optimizations that make up DuetECC and TrioECC require modest changes to the whole-entry ECC decoder, as shown in Figure 7b. Interleaving takes place at synthesis time, using Equation 2, and it is implemented by wires in the final design. The baseline SEC-DED decoder has four SEC-DED Inner Decoders that feed shared output logic; changes to this output logic are needed for the Correction Sanity Check.

The SEC-2bEC decoder adds four half-width HCM circuits that identify 2b symbol errors. Whereas the SEC IDs localize errors to one of the K data bits, the new HCMs localize errors to one of the $\frac{K}{2}$ 2b symbols. The SEC-2bEC optimization also augments the output logic to respect the 2b error correction signals. Our chosen SEC-2bEC code can also operate as a SEC-DED code if 2b error correction is never attempted. Thus, it is straightforward to add a DuetECC/TrioECC enable signal to the output formation logic. We envision that system architects can toggle between the two codes, either with a global setting per GPU or potentially on a per-CUDA-context basis, allowing different programs to prioritize error detection or correction.

Reed-Solomon Hardware: Reed-Solomon codes are determined by an irreducible polynomial, α; we use the primitive polynomial $\alpha = x^8 + x^6 + x^5 + x + 1$. Figure 7c shows a single one-shot Reed-Solomon SSC decoder; changes for SSC-DSD+ are highlighted in yellow. Syndrome generation is implemented using parallel GF(2^8) multipliers, and one-shot error location uses discrete logarithm logic (DLog$_\alpha$) for efficient polynomial division [4, 47] and end-around-carry (EAC) subtractors for low-cost modular subtraction [70]. The SSC-DSD+ code performs single-symbol error location using each pair of check-bytes, and correction is only allowed if all three error locations agree. This is conceptually similar

to the correction sanity check, and it provides complete double-symbol error detection and nearly-triple-symbol error detection in a heuristic manner, without solving the error locator polynomial.

7 RESILIENCE AND OVERHEADS

7.1 Resilience Per Event

Table 2 shows the SDC risk of the different ECC schemes and optimizations, given the 7 error patterns from Table 1. Entries marked "C" are always corrected, and those marked "D" are always detected, meaning that they have 0% SDC risk. Bit, pin, and byte errors are averaged over all possible error patterns; beat and entry errors are averaged over 1e7/1e9 uniformly random error patterns for binary and symbol-based codes, respectively, such that the worst-case 99% confidence interval is less than ±0.0003%/±0.00003%.

The results show the complementary impacts of our three binary code optimizations. The SEC-DED baseline is weak against severe-yet-common byte and beat errors, failing to correct or detect 23%–29% of such errors. Interleaving (denoted by "I:") provides error detection of byte errors, and it improves the coverage against beat errors by spreading the error among all four codewords. The correction sanity check (denoted by "+CSC") greatly increases beat and whole-entry error detection, allowing DuetECC (I:SEC-DED+CSC) to detect or correct all but 0.0013% of byte and beat errors. The SEC-2bEC code represents a resilience regression if it is employed alone in a non-interleaved manner (NI:SEC-2bEC) as it has significant miscorrection risk, even for double-bit errors. Interleaving the SEC-2bEC code provides perfect byte error correction, and it gives similar beat error improvements as with the SEC-DED code. TrioECC (I:SEC-2bEC+CSC) combines the advantages of all three optimizations, resulting in a compelling scheme that maintains both strong correction as well as relatively low SDC risk. There is a natural trade-off between aggressive correction and the detection capabilities of a code, but the combination of interleaving and the correction sanity check allow TrioECC to maintain 0.0085% SDC risk in the presence of severe beat and whole-entry errors.

The results show mixed performance for symbol-based ECC codes. The interleaved SSC baseline provides similar correction capabilities to TrioECC, but with higher SDC risk in the presence of 2-bit, 3-bit, beat, and whole-entry errors. The correction sanity check improves the resilience of the interleaved SSC code somewhat, but it still falls short of TrioECC. One notable result is that the relative improvement from the correction sanity check is far better for interleaved bit-correcting codewords than for symbol-based correction, due to the larger combinatorial space of rejected miscorrections from the four codewords and the fine correction granularity. Adding the correction sanity check to I:SSC results in a 2.34× decrease in the whole-entry SDC probability, whereas it results in a much larger 19× decrease for I:SEC-DED when moving to DuetECC. The SSC-DSD+ code provides by far the best resilience against soft errors, resulting in SDC for only 0.0002% of whole-entry errors. Also, while the SSC-DSD+ code does not provide perfect triple or quadruple-symbol detection, problematic error patterns are sparse enough that SSC-DSD+ manages to maintain 100% detection of 3b and pin errors at this codeword size.

Our beam testing results show some error patterns to be more prevalent than others. Accordingly, Figure 8 shows the resulting correction, detection, and SDC probability estimates when weighting by the error pattern probabilities from Table 1. The SEC-DED baseline corrects 74% of events, detecting another 20%, leaving a 5.4% SDC probability. Interleaving is able to correct 6.6% more events (due to opportunistic half-byte correction), while decreasing the SDC risk by 247×. DuetECC (I:SEC-DED+CSC) lowers the SDC risk by another 19× over interleaving alone to achieve 0.0013% SDC risk. DuetECC sacrifices a scant 0.53% correction probability relative to interleaving, due to some discarded opportunistic correction. The SEC-2bEC code, alone, has a prohibitively-high 9.3% SDC risk. Interleaving and the correction sanity check are able to vastly improve the safety of the SEC-2bEC code such that TrioECC (I:SEC-2bEC+CSC) offers a 97% correction probability with only 0.0085% SDC risk.

The interleaved SSC codes (I:SSC and I:SSC+CSC) offer correction capabilities that rival those of TrioECC, but with higher SDC risk—I:SSC has 4.3× higher SDC risk than TrioECC, and the correction sanity check only narrows this to 1.8× higher SDC. We show later that the interleaved SSC codes also have higher overheads than TrioECC. The SSC-DSD+ code has correction capabilities that approach those of TrioECC, but the latter has slightly higher correction capabilities due to pin correction. SSC-DSD+ has by far the lowest SDC risk, meaning that it should be preferred to DuetECC/TrioECC if a larger departure from SEC-DED ECC is permissible. The SSC-DSD+ decoder is larger and slower than those for the binary code-based schemes, and it sacrifices the ability to correct permanent pin failures.

Tab. 3: Hardware Overheads of the Considered Schemes.

Scheme	Area (AND2 / +%)	Delay (ns / +%)
Encoders		
SEC-DED Encoder	1176 / -	0.09 / -
SEC-2bEC (Perf.)	1693 / +44.02%	0.09 / -
SEC-2bEC (Eff.)	1346 / +17.53%	0.10 / +11.11%
SSC (Perf.)	1565 / +33.11%	0.10 / +11.11%
SSC (Eff.)	1245 / +5.870%	0.11 / +22.22%
SSC-DSD+ (Perf.)	5205 / +442.8%	0.12 / +33.33%
SSC-DSD+ (Eff.)	2908 / +247.4%	0.15 / +66.66%
Decoders		
SEC-DED Decoder	2467 / -	0.20 / -
DuetECC (Perf.)	4969 / +101.4%	0.20 / -
DuetECC (Eff.)	2733 / +10.79%	0.24 / +20.00%
SEC-2bEC (Perf.)	3810 / +54.45%	0.20 / -
SEC-2bEC (Eff.)	2806 / +13.73%	0.24 / +20.00%
TrioECC (Perf.)	4890 / +98.23%	0.21 / +05.00%
TrioECC (Eff.)	3020 / +22.43%	0.25 / +25.00%
SSC (Perf.)	6013 / +143.7%	0.32 / +60.00%
SSC (Eff.)	4880 / +97.82%	0.35 / +75.00%
SSC-DSD+ (Perf.)	10769 / +336.5%	0.39 / +95.00%
SSC-DSD+ (Eff.)	7177 / +190.9%	0.42 / +210.0%

7.2 Hardware Overheads

Table 3 shows the encoder and decoder hardware overheads. We estimate the overheads using Verilog designs synthesized by the Synopsys toolchain [65] with a 16nm industrial technology library. Area

and delay are estimated through standard-cell synthesis and static timing analysis, with area presented in a technology-independent manner using the equivalent AND2-gate count per circuit. Minimizing the area-time product of the circuits, the baseline SEC-DED encoder operates with 0.09ns delay and the decoder with 0.20ns delay. Each alternate design is presented at two speeds: coming as close to the delay of the baseline as possible (denoted "Perf." for *performant*) and operating at the area-time efficient design point for the modified circuit (denoted "Eff." for *efficient*).

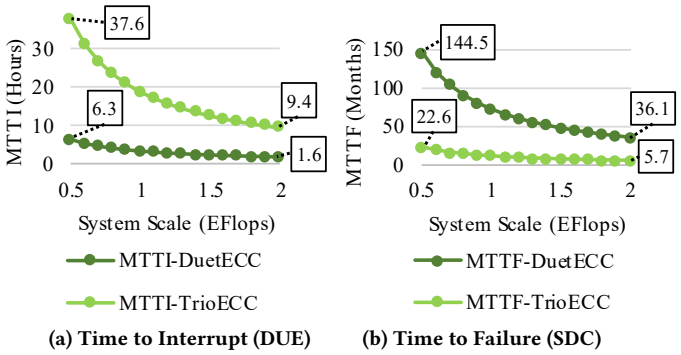
DuetECC and TrioECC have modest area and timing overheads, and they should offer a drop-in replacement for SEC-DED ECC with minimal microarchitectural impact on the GPU. At worst, the performant variant of TrioECC requires roughly 2500 extra AND2-gates of area per memory channel. We do not expect the 0.01–0.05ns of added decoder delay for TrioECC to have a negative impact on the memory system GPU clock rate, as compute-class GPU clock frequencies are limited to <1.5GHz/0.66ns for energy efficiency [41].

The overheads of the symbol-based codes are higher than those of DuetECC/TrioECC. The interleaved SSC code has an efficient encoder, but its decoder suffers from up to a 143.7% area and 75% delay overhead relative to SEC-DED ECC. The SSC-DSD+ code has a significantly larger and slower encoder and decoder that is roughly 2–4x as large and 66–210% slower than SEC-DED. Thus, while the symbol-based codes still likely offer single-cycle encoding and decoding, they represent a larger departure from the efficient and fast SEC-DED ECC than the sub-cycle DuetECC/TrioECC designs.

7.3 System-Level Resilience and Availability

Our resilience and hardware-overhead analyses show two promising organizations. Symbol-based SSC-DSD+ offers very high resilience against soft errors, but with hardware overhead and permanent pin correction drawbacks. Binary DuetECC/TrioECC offer a flexible and lightweight drop-in replacement for SEC-DED ECC. The resilience analysis clearly shows SEC-DED to be insufficient against soft errors, and SSC-DSD+ to be more-than-sufficient at providing low SDC rates at scale. The system-level resilience of DuetECC/TrioECC at scale are less clear, but we demonstrate below that they provide useful levels of soft error resilience and availability in high-performance computers and autonomous vehicles.

High-Performance Computing: Figure 9 shows the estimated mean-time-to-interrupt (MTTI, or DUE rate) and mean-time-to-failure (MTTF, or SDC rate) of DuetECC/TrioECC in an exascale supercomputer using NVIDIA A100 GPUs [53]. The figure is drawn assuming a failure rate of 12.51 FIT/Gb, inspired by the GDDR5 memory failure rates in the Titan supercomputer [67]. The correction/SDC tradeoff of the two binary schemes is apparent by the superiority of TrioECC for system MTTI (Figure 9a) and DuetECC for MTTF (Figure 9b). DuetECC suffers from frequent DUEs (every 1.6–6.3 hours) at scale, but it is completely safe with an HBM2 SDC period in years. TrioECC crashes every 9.4–37.6 hours, which could make the system significantly more usable for large jobs. However, it only reduces the MTTF to 5.7–22.6 months at scale. The SEC-DED baseline is not shown, as it has SDC rates in hours—at the smallest scale shown (0.5 exaflops), we expect an SDC every 22.5 hours. SSC-DSD+ is also not shown, as it almost completely avoids soft error SDC risk, with an MTTF in hundreds of years.



(a) Time to Interrupt (DUE) **(b) Time to Failure (SDC)**

Fig. 9: The system-level failure rates of DuetECC/TrioECC.

Autonomous Vehicles: Autonomous vehicles require stringent safety standards, and the number of vehicles on the road surpasses the GPU count of the largest supercomputers. The ISO 26262 functional safety standard for autonomous vehicles [30] dictates that vehicles satisfying the highest safety level must maintain $\leq$10 FIT of SDC. Assuming a 12.51 FIT/Gb raw HBM2 error rate, as above, we conclude that SEC-DED ECC is likely insufficient to satisfy this safety standard for a GPU-accelerated AV system. A SEC-DED protected A100 GPU suffers from 216 FIT of HBM2 SDC, while TrioECC reduces this to 0.29 FIT, and DuetECC to 0.045 FIT—well within the ISO 26262 requirements.

Strong and safe HBM2 ECC is also societally desirable for autonomous vehicles. In 2016–2017, 225.8 million drivers in the United States spent an average of 51 minutes per day driving a car [25], for a total of 1.92e8 hours per day. If all cars were autonomous, with a single A100 GPU per car, then SEC-DED protected HBM2 would result in an expected 41 SDC events on the road each day. DuetECC/TrioECC reduce the expected SDC counts to one every 18 days and one every 115 days, respectively. It is likely that an HBM2 DUE would interrupt the regular course of the car, meaning that on average 148 DuetECC-protected vehicles would require soft error-related recovery per day, while TrioECC and SSC-DSD+ reduce these events to 25 daily cars.

8 CONCLUSION

This paper presents neutron beam testing results for GPU HBM2. It describes, models, and draws conclusions about displacement-damaged DRAM cells in the neutron beam, guiding beam testing campaigns with on-package memory. Detailed soft error corruption patterns are then shown, which demonstrate a prevalence of severe byte and whole-entry errors due to particle strikes in DRAM logic structures. Based on these soft error corruption patterns, we propose two tailored ECC schemes, DuetECC/TrioECC and SSC-DSD+, which are able to decrease the silent data corruption risk by up to five orders of magnitude relative to SEC-DED ECC, while reducing the number of uncorrectable errors by as much as 7.87×. Despite these vast resilience improvements, DuetECC/TrioECC and SSC-DSD+ require no additional redundancy, no performance impacts, and they have modest area and complexity costs.

REFERENCES

[1] Advanced Micro Devices (AMD), Inc. 2013. BIOS and Kernel Developer's Guide (BKDG) for AMD Family 15h Models 00h-0Fh Processors. https://www.amd.com/system/files/TechDocs/42301_15h_Mod_00h-0Fh_BKDG.pdf.

[2] AMD. 2012. AMD Graphics Cores Next (GCN) Architecture. https://www.techpowerup.com/gpu-specs/docs/amd-gcn1-architecture.pdf.

[3] AMD. 2017. AMD's Radeon Next Generation GPU Architecture: Vega10. In *Proceedings of the Symposium on High Performance Chips (HOTCHIPS)*.

[4] Elwyn R Berlekamp. 1984. *Algebraic Coding Theory*. Aegean Park Press.

[5] David Blythe. 2020. The Xe GPU Architecture. In *Proceedings of the Symposium on High Performance Chips (HOTCHIPS)*.

[6] L. Borucki, G. Schindlbeck, and C. Slayman. 2008. Comparison of Accelerated DRAM Soft Error Rates Measured at Component and System Level. In *Proceedings of the International Reliability Physics Symposium (IRPS)*. 482–487.

[7] D. T. Brown. 1960. Error Detecting and Correcting Binary Codes for Arithmetic Operations. *IRE Transactions on Electronic Computers* EC-9, 3 (1960), 333–337.

[8] J.M. Caffrey. 2008. The Resiliency Challenge Presented by Soft Failure Incidents. *IBM Systems Journal* 47 (2008), 641–652.

[9] Carlo Cazzaniga and Christopher D. Frost. 2018. Progress of the Scientific Commissioning of a Fast Neutron Beamline for Chip Irradiation. *Journal of Physics: Conference Series* 1021, 1 (May 2018).

[10] Hsing-Min Chen, Carole-Jean Wu, Trevor Mudge, and Chaitali Chakrabarti. 2016. RATT-ECC: Rate Adaptive Two-Tiered Error Correction Codes for Reliable 3D Die-Stacked Memory. *ACM Transactions on Architecture and Code Optimization (TACO)* 13, 3 (2016), 24:1–24:24.

[11] Jack Choquette and Wish Gandhi. 2020. NVIDIA A100 GPU: Performance & Innovation for GPU Computing. In *Proceedings of the Symposium on High Performance Chips (HOTCHIPS)*.

[12] A. M. Chugg, A. J. Burnell, P. H. Duncan, S. Parker, and J. J. Ward. 2009. The Random Telegraph Signal Behavior of Intermittently Stuck Bits in SDRAMs. *IEEE Transactions on Nuclear Science* 56, 6 (2009), 3057–3064.

[13] A. M. Chugg, J. McIntosh, A. J. Burnell, P. H. Duncan, and J. Ward. 2010. Probing the Nature of Intermittently Stuck Bits in Dynamic RAM Cells. *IEEE Transactions on Nuclear Science* 57, 6 (2010), 3190–3198.

[14] Douglas Crockford. 2019. Base 32. https://www.crockford.com/base32.html.

[15] Timothy J Dell. 1997. A White Paper on the Benefits of Chipkill-Correct ECC for PC Server Main Memory. *IBM Microelectronics Division* (1997), 1–23.

[16] Catello Di Martino, Zbigniew Kalbarczyk, Ravishankar K Iyer, Fabio Baccanico, Joseph Fullop, and William Kramer. 2014. Lessons Learned from the Analysis of System Failures at Petascale: The Case of Blue Waters. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN)*.

[17] Fernando Fernandes dos Santos and Paolo Rech. 2017. Analyzing the Criticality of Transient Faults-induced SDCS on GPU Applications. In *Proceedings of the 8th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems*. 1:1–1:7.

[18] A. Dutta and N.A. Touba. 2007. Multiple Bit Upset Tolerant Memory Using a Selective Cycle Avoidance Based SEC-DED-DAEC Code. In *Proceedings of the VLSI Test Symposium (VTS)*. 349–354.

[19] L. D. Edmonds and L. Z. Scheick. 2008. Physical Mechanisms of Ion-Induced Stuck Bits in the Hyundai 16M ×4 SDRAM. *IEEE Transactions on Nuclear Science* 55, 6 (2008), 3265–3271.

[20] DJS Findlay. 2007. ISIS-Pulsed Neutron and Muon Source. In *2007 IEEE Particle Accelerator Conference (PAC)*. 695–699.

[21] Fujitsu Ltd. 2020. FUJITSU Processor A64FX. https://www.fujitsu.com/downloads/JP/jsuper/a64fx/a64fx_datasheet.pdf.

[22] Bharan Giridhar, Michael Cieslak, Deepankar Duggal, Ronald Dreslinski, Hsing Min Chen, Robert Patti, Betina Hold, Chaitali Chakrabarti, Trevor Mudge, and David Blaauw. 2013. Exploring DRAM Organizations for Energy-Efficient and Resilient Exascale Memories. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC)*. 1–12.

[23] Seong-Lyong Gong, Jungrae Kim, Sangkug Lym, Michael Sullivan, Howard David, and Mattan Erez. 2018. DUO: Exposing On-Chip Redundancy to Rank-Level ECC for High Reliability. In *Proceedings of the International Symposium on High Performance Computer Architecture (HPCA)*. 683–695.

[24] Seong-Lyong Gong, Minsoo Rhu, Jungrae Kim, Jinsuk Chung, and Mattan Erez. 2015. Clean-ECC: High Reliability ECC for Adaptive Granularity Memory System. In *Proceedings of the International Symposium on Microarchitecture (MICRO)*. 611–622.

[25] Andrew Gross. 2019. Think You're In Your Car More? You're Right. Americans Spend 70 Billion Hours Behind the Wheel. https://newsroom.aaa.com/2019/02/think-youre-in-your-car-more-youre-right-americans-spend-70-billion-hours-behind-the-wheel/.

[26] Sudhanva Gurumurthi. 2020. Advanced Memory Device Correction (AMDC) for Servers. https://www.amd.com/system/files/documents/advanced-memory-device-correction.pdf.

[27] Hewlett-Packard. 2021. Memory RAS Technologies for HPE ProLiant/Synergy/Blade Gen10 Servers with Intel Xeon Scalable Processors. https://www.hpe.com/psnow/doc/4AA4-3490ENW.

[28] M.Y. Hsiao. 1970. A Class of Optimal Minimum Odd-weight-column SEC-DED Codes. *IBM Journal of Research and Development* 14, 4 (1970), 395–401.

[29] Intel Corp. 2011. Intel Xeon Processor E7 Family: Reliability, Availability, and Serviceability. https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/xeon-e7-family-ras-server-paper.pdf.

[30] ISO. 2011. ISO 26262-9:2011 Preview Road Vehicles Functional Safety. https://www.iso.org/standard/51365.html.

[31] Aakash Jani. 2020. SiPearl Develops ARM HPC Chip. https://www.linleygroup.com/mpr/login.php?return_url=/mpr/article.php?id=12379&num=6227. The Linley Group Microprocessor Report from October 19, 2020.

[32] JEDEC Solid State Technology Association 2006. *Measurement and Reporting of Alpha Particle and Terrestrial Cosmic Ray-Induced Soft Errors in Semiconductor Devices, JeSD89A*. JEDEC Solid State Technology Association.

[33] H. Jeon, G.H. Loh, and M. Annavaram. 2014. Efficient RAS Support for Die-stacked DRAM. In *Proceedings of the International Test Conference (ITC)*. 1–10.

[34] Zhe Jia, Marco Maggioni, Benjamin Staiger, and Daniele P Scarpazza. 2018. Dissecting the NVIDIA Volta GPU Architecture via Microbenchmarking. *arXiv preprint arXiv:1804.06826* (2018).

[35] X. Jian, V. Sridharan, and R. Kumar. 2016. Parity Helix: Efficient Protection for Single-Dimensional Faults in Multi-Dimensional Memory Systems. In *Proceedings of the International Symposium on High Performance Computer Architecture (HPCA)*. 555–567.

[36] Joint Electron Device Engineering Council 2013. *High Bandwidth Memory (HBM) DRAM, JESD235*. Joint Electron Device Engineering Council.

[37] H. Jun, J. Cho, K. Lee, H. Son, K. Kim, H. Jin, and K. Kim. 2017. HBM (High Bandwidth Memory) DRAM Technology and Architecture. In *Proceedings of the International Memory Workshop (IMW)*. 1–4.

[38] Y. Katayama and S. Morioka. 2000. One-Shot Reed-Solomon Decoding for High-Performance Dependable Systems. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN)*. 390–399.

[39] Jungrae Kim, Michael B. Sullivan, and Mattan Erez. 2015. Bamboo ECC: Strong, Safe, and Flexible Codes for Reliable Computer Memory. In *Proceedings of the International Symposium on High Performance Computer Architecture (HPCA)*.

[40] K. Kim and J. Lee. 2009. A New Investigation of Data Retention Time in Truly Nanoscaled DRAMs. *IEEE Electron Device Letters* 30, 8 (2009), 846–848.

[41] Ronny Krashinsky, Olivier Giroux, Stephen Jones, Nick Stam, and Sridhar Ramaswamy. 2020. NVIDIA Ampere Architecture In-Depth. https://developer.nvidia.com/blog/nvidia-ampere-architecture-in-depth/.

[42] Scott Levy, Kurt B. Ferreira, Nathan DeBardeleben, Taniya Siddiqua, Vilas Sridharan, and Elisabeth Baseman. 2018. Lessons Learned from Memory Errors Observed over the Lifetime of Cielo. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC)*. 43:1–43:12.

[43] C. Lim, H. S. Jeong, G. Bak, S. Baeg, S. J. Wen, and R. Wong. 2015. Stuck Bits Study in DDR3 SDRAMs Using 45-MeV Proton Beam. *IEEE Transactions on Nuclear Science* 62, 2 (2015), 520–526.

[44] C. Lim, K. Park, and S. Baeg. 2017. Active Precharge Hammering to Monitor Displacement Damage Using High-Energy Protons in 3x-nm SDRAM. *IEEE Transactions on Nuclear Science* 64, 2 (2017), 859–866.

[45] Shu Lin and Daniel J Costello. 2004. *Error Control Coding* (second ed.). Prentice Hall.

[46] Caio Lunardi, Fritz Previlon, David Kaeli, and Paolo Rech. 2018. On the Efficacy of ECC and the Benefits of FinFET Transistor Layout for GPU Reliability. *IEEE Transactions on Nuclear Science* 65, 8 (2018), 1843–1850.

[47] Florence Jessie MacWilliams and Neil James Alexander Sloane. 1977. *The Theory of Error Correcting Codes*. Vol. 16. Elsevier.

[48] Georgios Mappouras, Alireza Vahid, Robert Calderbank, Derek R Hower, and Daniel J Sorin. 2017. Jenga: Comprehensive Fault Tolerance for Stacked DRAM. In *Proceedings of the International Conference on Computer Design (ICCD)*. 361–368.

[49] Prashant J. Nair, David A. Roberts, and Moinuddin K. Qureshi. 2016. Citadel: Efficiently Protecting Stacked Memory from TSV and Large Granularity Failures. *ACM Transactions on Architecture and Code Optimization (TACO)* 12, 4 (2016), 49:1–49:24.

[50] B. Nie, D. Tiwari, S. Gupta, E. Smirni, and J. H. Rogers. 2016. A Large-Scale Study of Soft-Errors on GPUs in the Field. In *Proceedings of the International Symposium on High Performance Computer Architecture (HPCA)*. 519–530.

[51] Thomas Norrie, Nishant Patil, Doe Hyun Yoon, George Kurian, Sheng Li, James Laudon, Cliff Young, Norman P. Jouppi, and David Patterson. 2020. Google's Training Chips Revealed: TPUv2 and TPUv3. In *Proceedings of the Symposium on High Performance Chips (HOTCHIPS)*.

[52] NVIDIA. 2016. NVIDIA Tesla P100—The Most Advanced Data Center Accelerator Ever Built Featuring Pascal GP100, the World's Fastest GPU. http://www.nvidia.com/object/pascal-architecture-whitepaper.html.

[53] NVIDIA. 2020. NVIDIA A100 Tensor Core GPU Architecture: Unprecedented Acceleration at Every Scale. https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/nvidia-ampere-architecture-whitepaper.pdf.

[54] Daniel Oliveira, Laércio Pilla, Nathan DeBardeleben, Sean Blanchard, Heather Quinn, Israel Koren, Philippe Navaux, and Paolo Rech. 2017. Experimental and Analytical Study of Xeon Phi Reliability. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC)*.

[55] M. Park, S. Jeon, G. Bak, C. Lim, S. Baeg, S. Wen, R. Wong, and N. Yu. 2017. Soft Error Study on DDR4 SDRAMs Using a 480 MeV Proton Beam. In *Proceedings of the International Reliability Physics Symposium (IRPS)*. SE–3.1–SE–3.6.

[56] Suresh Ramalingam. 2026. HBM Package Integration: Technology Trends, Challenges and Applications. In *Proceedings of the Symposium on High Performance Chips (HOTCHIPS)*.

[57] Minsoo Rhu, Michael Sullivan, Jingwen Leng, and Mattan Erez. 2013. A Locality-Aware Memory Hierarchy for Energy-Efficient GPU Architectures. In *Proceedings of the International Symposium on Microarchitecture (MICRO)*. 86–98.

[58] A. Rodriguez, F. Wrobel, A. Michez, A. Touboul, F. Bezerra, R. Ecoffet, E. Lorfèvre, and F. Saigné. 2016. TCAD Simulations of Leakage Currents Induced by SDRAM Single-Event Cell Degradation. In *Proceedings of the European Conference on Radiation and Its Effects on Components and Systems (RADECS)*. 1–5.

[59] A. Rodriguez, F. Wrobel, A. Samaras, F. Bezerra, B. Vandevelde, R. Ecoffet, A. Touboul, N. Chatry, L. Dilillo, and F. Saigne. 2015. Proton-Induced SDRAM Cell Degradation. In *Proceedings of the European Conference on Radiation and Its Effects on Components and Systems (RADECS)*. 1–4.

[60] C. Slayman. 2011. Soft Error Trends and Mitigation Techniques in Memory Devices. In *Proceedings of the Reliability and Maintainability Symposium (RAMS)*. 1–5.

[61] Marc Snir, Robert W Wisniewski, Jacob A Abraham, Sarita V Adve, Saurabh Bagchi, Pavan Balaji, Jim Belak, Pradip Bose, Franck Cappello, Bill Carlson, Andrew A Chien, Paul Coteus, Nathan A DeBardeleben, Pedro C Diniz, Christian Engelmann, Mattan Erez, Saverio Fazzari, Al Geist, Rinku Gupta, Fred Johnson, Sriram Krishnamoorthy, Sven Leyffer, Dean Liberty, Subhasish Mitra, Todd Munson, Rob Schreiber, Jon Stearley, and Eric Van Hensbergen. 2014. Addressing Failures in Exascale Computing. *The International Journal of High Performance Computing Applications* 28, 2 (2014), 129–173.

[62] Vilas Sridharan, Nathan DeBardeleben, Sean Blanchard, Kurt B. Ferreira, Jon Stearley, John Shalf, and Sudhanva Gurumurthi. 2015. Memory Errors in Modern Systems: The Good, The Bad, and The Ugly. In *Proceedings of the International Symposium on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. 297–310.

[63] Vilas Sridharan and Dean Liberty. 2012. A Study of DRAM Failures in the Field. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC)*.

[64] Vilas Sridharan, Jon Stearley, Nathan DeBardeleben, Sean Blanchard, and Sudhanva Gurumurthi. 2013. Feng Shui of Supercomputer Memory: Positional Effects in DRAM and SRAM Faults. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC)*.

[65] Synopsys Inc. 2019. Design Compiler P-2019.03.

[66] Devesh Tiwari, Saurabh Gupta, George Gallarno, Jim Rogers, and Don Maxwell. 2015. Reliability Lessons Learned from GPU Experience with the Titan Supercomputer at Oak Ridge Leadership Computing Facility. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*.

[67] D. Tiwari, S. Gupta, J. Rogers, D. Maxwell, P. Rech, S. Vazhkudai, D. Oliveira, D. Londo, N. DeBardeleben, P. Navaux, L. Carro, and A. Bland. 2015. Understanding GPU Errors on Large-Scale HPC Systems and the Implications for System Design and Operation. In *Proceedings of the International Symposium on High Performance Computer Architecture (HPCA)*. 331–342.

[68] Alyson D Topper, Michael J Campola, Dakai Chen, Megan C Casey, Ka-Yen Yau, Donna J Cochran, Kenneth A LaBel, Raymond L Ladbury, Timothy K Mondy, Martha V O'Bryan, et al. 2017. Compendium of Current Total Ionizing Dose and Displacement Damage Results from NASA Goddard Space Flight Center and NASA Electronic Parts and Packaging Program. In *IEEE Radiation Effects Data Workshop (REDW)*. 1–11.

[69] Wikipedia. 2020. Random-Access Memory. https://en.wikipedia.org/wiki/Random-access_memory#DRAM. [Online; accessed 24-November-2020].

[70] R. Zimmermann. 1999. Efficient VLSI Implementation of Modulo ($2^n \pm 1$) Addition and Multiplication. In *Proceedings of the IEEE Symposium on Computer Arithmetic*. 158–167.

[71] S. M. Zulkifli, B. Zee, W. Qiu, and A. Gu. 2017. High-Res 3D X-ray Microscopy for Non-Destructive Failure Analysis of Chip-to-Chip Micro-Bump Interconnects in Stacked Die Packages. In *Proceedings of the International Symposium on Physical and Failure Analysis of Integrated Circuits (IPFA)*. 1–5.