2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)

Killi: Runtime Fault Classification to Deploy Low Voltage Caches without MBIST

Shrikanth Ganapathy† John Kalamatianos† Bradford M. Beckmann† Steven Raasch† Lukasz Szafaryn‡1

†AMD Research, Advanced Micro Devices, Inc.
‡Intel

{shrikanth.ganapathy, john.kalamatianos, brad.beckmann, steven.raasch}@amd.com, lukasz.g.szafaryn@intel.com

***Abstract*—Supply voltage (V_{DD}) scaling is one of the most effective mechanisms to reduce energy consumption in high-performance microprocessors. However, V_{DD} scaling is challenging for SRAM-based on-chip memories such as caches due to persistent failures at low voltage (LV). Previously designed LV-enabling mechanisms require additional Memory Built-in Self-Test (MBIST) steps, employed either offline or online to identify persistent failures for every LV operating mode. However, these additional MBIST steps are time consuming, resulting in extended boot time or delayed power state transitions. Furthermore, most prior techniques combine MBIST-based solutions with customized Error Correction Codes (ECC), which suffer from non-trivial area or performance overheads.**

In this paper, we highlight the practical challenges for deploying LV techniques and propose a new low-cost error protection scheme, called Killi, which leverages conventional ECC and parity to enable LV operation. Foremost, the failing lines are discovered dynamically at runtime using both parity and ECC, negating the need for extra MBIST testing. Killi then provides on demand error protection by decoupling cheap error detection from expensive error correction. Killi provides error detection capability to all lines using parity but employs Single Error Correction, Double Error Detection (SECDED) ECC for a subset of the lines with a single LV fault. All lines with more than one fault are disabled. We evaluate this completely hardware enclosed solution on a GPU write-through L2 cache and show that the V_{min} (minimum reliable V_{DD}) can be reduced to 62.5% of nominal V_{DD} when operating at 1GHz with only a maximum of 0.8% performance degradation. As a result, an 8-CU GPU with Killi can reduce the power consumption of the L2 cache by 59.3% compared to the baseline L2 cache running at nominal V_{DD}. In addition, Killi reduces the error protection area overhead by 50% compared to SECDED ECC.

***Keywords*—cache, energy-efficiency, GPU, low voltage, reliability**

1. INTRODUCTION

As more applications continue to exploit throughput-oriented architectures such as Graphics Processing Units (GPUs), designers strive to maximize performance by increasing effective memory bandwidth through the integration of large on-chip SRAM caches. Last level GPU caches are now multi-megabyte and service a significant volume of memory traffic, leading to high dynamic power consumption. Under-volting of such GPU L2 caches in addition to improving GPU energy efficiency, also allows for graceful over-volting of compute units for improved performance within the allowed power budget.

GPU L2 caches are ideal candidates for low voltage (LV) operation because: a) they are intrinsically latency insensitive so they can tolerate LV error protection latencies [1] and, b) they can operate as write-through caches [2] [2], which eliminates the necessity for error correction when coping with detected but uncorrectable errors. Instead error correction can be employed as a performance optimization to prevent main memory accesses when errors are detected.

Recent research has proposed variations of ECC to tolerate LV errors in caches [3]–[7]. As a result, these techniques can lower V_{min} and improve energy efficiency, while ensuring reliable operation. However, ECC can only correct up to a fixed number of errors, and mechanisms leveraging stronger ECC often sacrifice cache capacity to accommodate an increased number of ECC checkbits. As a result, these approaches reduce the percentage of useable cache capacity and can cause non-trivial performance degradation. Furthermore, the identification of lines that have more errors than what ECC can protect, requires additional MBIST to run at each LV transition. This requirement extends boot time and delays power state transitions. In contrast, if the additional MBIST steps are delayed until runtime, cache bandwidth and capacity are lost [7]. As an alternative to running MBIST,

[1] Work performed while employed at AMD Research.

[2] Specifically, writes are “set up to bypass (the L2 cache) to ensure system coherence”.

per-voltage fault population could be maintained in memory, but that solution is costly and complex.

In this work, we avoid these shortcomings and describe a new LV fault classification and correction technique, called Killi[3]. Killi dynamically detects and adapts to failures on-the-fly. We demonstrate Killi by applying it to a GPU L2 cache design to enable LV operation. Specifically, our proposal makes the following contributions:

1. Killi detects the number of faults per cache line at runtime without using additional MBIST, or sacrificing cache capacity and with minimal loss of cache bandwidth. By doing so, Killi minimizes the impact on system boot time and normal program execution.
2. Killi reduces error protection area overhead by 50% versus prior LV techniques by separating error detection from error correction, which in turn decouples error correction overhead from the cache organization.
3. Killi optimizes the performance-versus-storage tradeoff for a write-through GPU L2 cache that is running at both a *peak* operating frequency and under a LV.

Killi is able to provide all of the above features by leveraging observations from 14nm FinFET technology that the majority (>95%) of the cache lines have zero or one LV failure [8]. Since LV failures are persistent (both in space and time for a given voltage), they only need to be detected once to mark the lines as faulty [9]. In cases of masked faults (or incorrect characterization), which may take multiple cache line accesses before discovery, we show that Killi learns on-the-fly and corrects itself. The absence of a separate characterization (or classification) phase allows Killi to continuously operate the cache under LV and at full bandwidth. For lines that are found to be fault-free, Killi employs only parity for detecting transient errors. For lines with one LV fault, Killi uses both parity and SECDED ECC for error detection and correction. Killi disables all lines with two or more failures. Finally, Killi distinguishes between transient and persistent faults, and adapts the required level of protection accordingly.

2. RELATED WORK

Several prior proposals have investigated LV operation due its potentially significant energy-efficiency gains. In this section, we summarize those prior efforts and distinguish Killi's unique benefits.

2.1 Pulsing and Upsizing SRAM Cell

Circuit-level techniques like bitline and wordline pulsing modulate SRAM timing to facilitate reliable operation [10]. Similarly, wordline boosting increases the wordline voltage for a short period of time to facilitate reliable operation [11], [12]. One of the main disadvantages of such approaches is the requirement of post-silicon tuning as LV SRAM failures are influenced by process variations.

Orthogonally, design-time circuit-level techniques utilize upsized SRAM cells or variation of the SRAM cell based on varying transistor counts such as 8T, 10T, etc. [13]–[15]. Previous work has shown that standard 6T SRAM cells outperform other cell designs and can simultaneously provide both peak frequency operation and sufficient robustness [16]. Therefore, changing the fundamental design of the memory cell would most often result in trading-off performance for improved robustness and/or area overhead.

2.2 Architecture-Circuit Co-Design

Gottscho *et al.* propose two variants of power/capacity scaling (PCS) where faulty blocks are power gated when operating under a LV [17]. The technique requires apriori knowledge of faulty blocks and uses that information to build a fault map. Every time the voltage is lowered, the faulty blocks need to be updated and fault map needs to be at least partially rebuilt. Khan *et al.* evaluate the benefit of using a combination of SRAM cells with different robustness levels [18]. Under LV, the robust (and upsized) cells are used for storing critical data while the non-robust cells are power-gated or store only non-critical data. However, SRAM arrays are tightly pitch-matched structures and incorporating multiple bitcell layouts is difficult.

2.3 Remapping and Error Correction Codes

Most architectural schemes targeting LV cache operation can be broadly classified into schemes that exploit either some form of remapping logic to move data into non-faulty locations or employ customized ECC-based solutions for recalculating non-faulty data based on redundant coded information (checkbits). Remapping schemes require apriori knowledge of faulty cell locations and assume that the fault map can be obtained at system startup. PADded Cache, for example, uses a programmable address decoder to remap faulty cache lines from one set into a set with non-faulty lines [19]. As the number of on-chip SRAM arrays and their size increases, the time needed to generate those fault maps becomes prohibitively long and techniques such as PADded Cache are no longer scalable.

Other techniques choose to reduce the total amount of usable cache space in LV mode by disabling defective blocks and/or storing the faulty locations in the cache itself. Archipelago is one such technique that remaps in a more fine-grain manner wherein chunks of data within a cache line are grouped together and remapped to a different physical line [20]. Such a complex reorganization of the data requires additional metadata for storing the source and destination locations of the data needing two address translation tables that operate in the critical path of the cache access. Similarly, Bit-fix and Word-disable either disable a large portion of the cache or combine non-faulty physically disjoint blocks to create a logical cache block and operate in the LV region [3]. For last-level CPU caches, Intel® Cache Safe Technology detects and disables faulty cache lines such that only the fault-free lines are operable [21].

[3] The word Killi is inspired from the Atlantic killifish which has known to have quickly evolved from increased levels of toxicity. https://www.ucdavis.edu/news/against-tide-fish-adapts-quickly-lethal-levels-pollution/

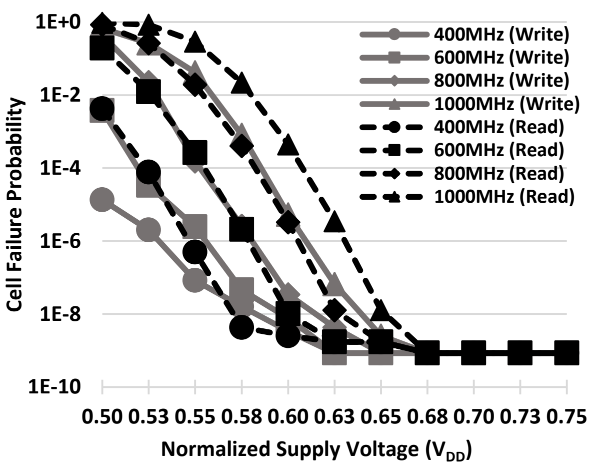

Figure 1: Measured (from 14nm FinFET test chips) SRAM cell failure probabilities for read disturbance and writeability tests. The supply voltage is normalized to nominal supply voltage [8].

Orthogonal to remapping schemes, several customized ECC approaches have been proposed. For example, , Multi-bit Segmented ECC (MS-ECC) configures the amount of cache capacity to enable strong ECC protection (Orthogonal Latin Square Codes) and allows multi-bit fault detection and correction [22]. In comparison, Miller *et al.* proposed Parichute which augments a certain class of ECC with additional checkbits to improve coverage under LV [23]. It requires rearchitecting the cache to store the additional checkbits and dynamically trades off cache capacity for improved reliability. As we show later, Killi enables a family of error correction codes and is not constrained by a specific ECC scheme.

Variable-Strength ECC (VS-ECC) is an ECC-based technique and similar to Killi in that it uses a two-layered recovery scheme where all lines in the cache have standard SECDED protection and a fixed subset of lines in each set have higher-order ECC protection [4]. The amount of storage required for storing additional checkbits for the higher-order ECC is designed to be the same across every set in the cache. If there is a non-uniform failure distribution, this approach results in non-optimal storage usage.

Correction-prediction is another two-layered approach that uses a combination of fast (weak) correction and slow (strong) correction for recovery [6]. It relies on a fast correction and starts speculative execution assuming correct data. Upon detecting an error (wrong output of fast correction), the pipeline is squashed, and the correct value is fed into the pipeline before the next instruction is restarted. The technique does not exploit the persistent nature of LV failures and performs both weak and strong correction upon each request to the data leading to lower performance.

FLAIR uses a combination of standard SECDED-based ECC and Dual Modular Redundancy (DMR) [7]. It relies on an online LV fault detection phase, just like Killi, but uses MBIST testing to detect LV failures for a portion of the cache while simultaneously relying on ECC and DMR to operate the other portion of the cache that is not under test. Thus, FLAIR sacrifices cache capacity and bandwidth during the fault characterization phase. FLAIR disables lines with multi-bit errors and protects the remaining lines with SECDED ECC per line leading to higher area overhead than Killi. Finally, FLAIR may not be able to detect a multi-bit soft-error on a line with a LV fault because of its exclusive reliance on SECDED ECC and its associated error protection capabilities.

Manoochehri *et al* propose the idea of decoupling error detection from error correction by leveraging a combination of parity and data encoding for error detection and recovery [24]. In the event that parity detects an error in a dirty line, the recovery mechanism requires reading data from all of the dirty lines in the cache to retrieve the clean uncorrupted data. The cost of such a mechanism is prohibitive for LV operations because the probability of failure is much higher than soft-errors.

2.4 Distinguishing Killi

In relation to prior work, Killi is a completely enclosed hardware solution with no reliance on MBIST, making it easier to implement. When a chip starts executing under a LV, Killi identifies which cache lines are failing and applies the appropriate protection mechanism. When the voltage is changed, Killi resets its prior fault location knowledge and relearns the failure distribution for the new voltage without MBIST. As a result, from an architect's perspective, Killi has only one mode of execution and from an implementation perspective, Killi does not require complicated multi-block interactions like previously proposed MBIST techniques.

Furthermore, Killi targets peak frequencies while simultaneously operating under LV. This is possible by leveraging a dual-rail V_{DD} circuit design where the bitcell, wordlines, and write drivers operate under LV while the periphery and controller logic operate under the main V_{DD} (as the rest of the chip). Therefore, the failures are limited to bitflips within the SRAM array and not failing speed-paths on control circuits that primarily limit frequency at LV.

Finally, while Killi is designed to be independent of cache policies, we evaluate it for a GPU L2 cache in write-through mode for the following two reasons: 1) the SRAM LV data obtained from silicon measurements is limited to a maximum of 1GHz which matches the peak frequency of commercially available GPUs and, 2) in order to be able to support fine-grained coherence with the Heterogenous System Architecture (HSA) memory model for compute applications, a coherent GPU write-through L2 is needed as there is no support for cache flushes in write-back mode [2].

3. LOW VOLTAGE SRAM FAILURES

In this section, we review the measured LV SRAM fault rates that guided our Killi design. Specifically, we leverage the 14nm FinFET-based SRAM fault characterization data from Ganapathy *et al.* [8]. The data comprises silicon-measurements from a large number of test chips (103 dies) spread across two wafers. All SRAM arrays were tested for a range of frequencies from 400MHz – 1000MHz and

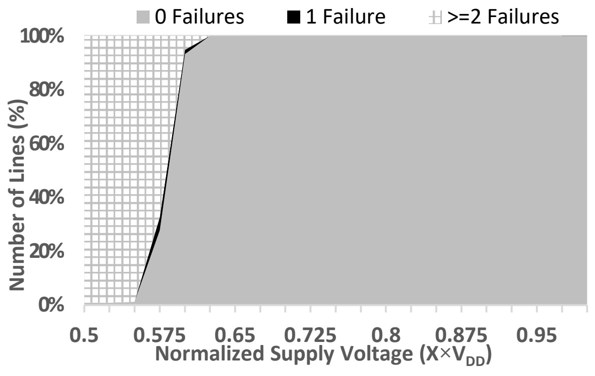

Figure 2: Percentage of fault-free lines (zero errors), lines with one error, and lines with two or more errors [8].

operating between $0.5 \times V_{DD}$ and V_{DD}[4]. The tests measure for stability under two conditions – writeability and read stability. The writeability check determines the ability to change state from 0 to 1 and vice-versa in the time that the wordlines are held high. The read disturbance test checks for a cell-flipping state when the wordline is turned on with the write data not being driven. The measured SRAM cell failure probabilities are shown in Figure 1. The voltage values are normalized to the nominal supply voltage (V_{DD}).

As can be seen in Figure 1, for voltages lower than $0.675 \times V_{DD}$, the cell failure probabilities start to increase exponentially. The authors in that study observed that at 1GHz and $0.625 \times V_{DD}$, >95% of the SRAM array rows have fewer than two (read or write) failures. Further, it was observed that the failures were monotonically increasing in that if they occurred at a given voltage or frequency, they occurred always for all voltages lower and all frequencies higher, respectively. Further, the tests were repeated for the same input conditions to ensure that the observed failures were persistent and not random transient bit-flips.

A GPU L2 cache is required to operate at peak frequency to sustain peak bandwidth. Therefore, Killi is designed to cope with the failures that occur at LV and 1GHz operating frequency. As observed from silicon measurements, Figure 2 estimates the percentage of lines with 0, 1 and 2 (or more) failures in a 64-byte line array. The majority of lines are fault-free in the voltage range of interest. For lower voltages, the percentage of lines with two or more errors increases drastically such that the area and energy cost of error protection may negate the benefits of LV operation. While Killi is agnostic to the particular type of failure (i.e. it responds to transient, ageing, and high-voltage errors the same way), in this paper we consider only LV failures.

4. KILLI: LV CACHES WITHOUT MBIST

Killi is a high-performance mechanism for ensuring reliable operation under LV for lines with fewer than two failures. This involves classifying lines into groups (enabled/disabled) at runtime and providing sufficient error correction capabilities against both LV failures and other types of failures. Because only a small percentage of enabled lines encounter failures, Killi provides on-demand error correction by decoupling the ECC storage from the main cache data array. This decoupling allows us to scale the ECC cost independently from the overall cache size.

4.1 Cache Organization

In this section, we describe the modifications to the cache needed to implement Killi. As shown in Figure 3, each line in the cache is protected with segmented and interleaved parity. The cache line is logically divided into multiple segments and 1-bit parity is generated for each segment. The segments are interleaved to improve the coverage for multi-bit soft-errors, which are known to occur in adjacent bit cells [25]. Interleaving parity does not offer additional benefits for LV failures because they occur randomly.

Killi's decoupling of error-detection and error-correction allows the cache to operate reliably under LV while significantly lowering area overhead. As we target a write-through cache, parity sufficiently protects lines with no LV failures because any detected soft error can fetch a correct data copy from the main memory. As soft-errors are rare, the performance impact of fetching data from main memory is trivial. For lines with one LV failure, Killi uses SECDED ECC protection and disables lines with two or more failures at the expense of reduced cache capacity. It should be noted

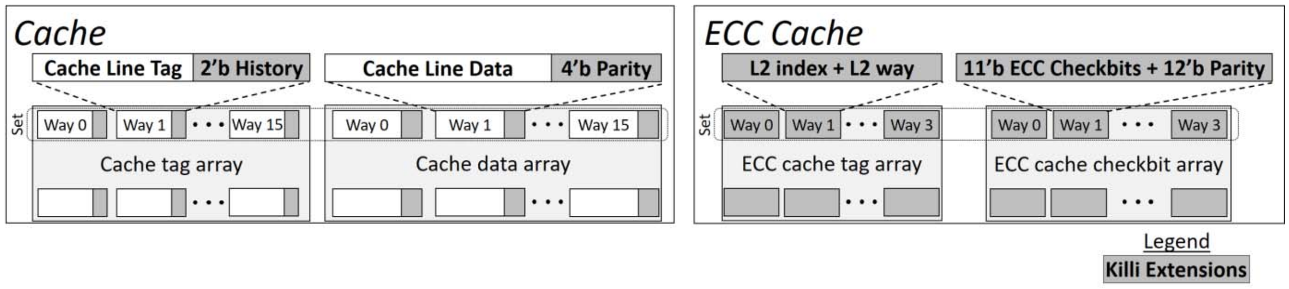

Figure 3: Killi organization is composed of a cache working in tandem with an ECC cache. The cache is augmented with history and parity bits for all lines in the cache. The ECC cache stores the checkbits and parity of a subset of cache lines. The 2 bits of History will be referred to as DFH (*Detected Fault History*) as explained in Section 4.1.

[4] The paper provides only normalized supply voltage values (due to confidentiality agreements with the foundry).

that the lines disabled at a particular LV may be reclaimed at higher voltages. To distinguish between fault-free lines and lines containing one or two-or-more LV failures[5], Killi uses *detected fault history* (DFH) bits. The cache tag array stores the DFH bits and on every cache tag lookup the DFH bits indicate the number of faults/line.

For lines with one LV failure protected with SECDED ECC, Killi introduces a separate cache for storing the associated error protection metadata. The small set-associative cache called the *ECC cache* is much smaller in size compared to the L2 and caches error protection metadata for a subset of L2 lines. The ECC cache is indexed by the same physical address as the main GPU L2 cache. The ECC cache tags hold the index and way of the L2 line that they protect, instead of the physical address, to reduce area. In this study we evaluate Killi for various ECC cache sizes, each defined by the proportion of L2 cache lines covered by the ECC cache. Going forward, we will refer to the main L2 data cache simply as *cache* to differentiate it from the *ECC cache*.

As mentioned previously, each cache line is logically divided into 16 interleaved segments (each segment is 32 bits wide). For each segment, Killi generates 1-bit parity. The 16 bits of parity split between the cache and the ECC cache. The cache stores 4 parity bits while the remaining 12 bits are stored in the ECC cache. Each line in the ECC cache also holds 11 bits of checkbits generated using SECDED ECC. The 11 ECC checkbits can help correct one failure or detect up to two failures in a 64B cache line. Killi initially generates 16 bits of parity when the failures within the cache line are unknown. Once the failures per line are identified, Killi uses only 4 bits of parity per fault-free cache line (each parity bit protects a 128-bit wide segment) and the associated ECC cache is freed. This optimization helps maintain the ECC cache's low area overhead, since most lines are classified as fault-free at any point in time.

In addition to the extra storage (DFH bits, parity, ECC checkbits), Killi also requires the circuitry for ECC encoding/decoding, parity generation and checking. The tag array operates under nominal voltage and protected with parity. Therefore, it is immune to LV faults and requires only protection against soft-errors.

4.2 Fault Group Classification

At reset (system startup or LV mode entry), all cache lines set their DFH bits to a state indicating unknown fault status (marked with the DFH state of b'01) as shown in Table 1. In this state Killi assumes that all lines require 16 parity bits and SECDED ECC (11 bits) protection. As lines get accessed or evicted, the lines get classified to one of the stable states using ECC and parity feedback. For lines that are enabled and have a stable DFH state (b'00 or b'10), Killi reduces the length of parity and stores 4 bits of interleaved and segmented parity in the cache. In addition, for b'l0 lines that are discovered to have one LV error, Killi generates ECC checkbits as well. For

DFH	State	Errors/Line	Protection
b'00	Stable	0	4-bits Parity
b'01	Initial	Unknown	16-bits Parity + SECDED ECC
b'10	Stable	1	4-bits Parity + SECDED ECC
b'11	Stable	2 or more	N/A (Disabled)

Table 1: DFH bits for tracking status of each cache line

lines that are disabled (two or more failures), Killi sets their DFH bits to b'11.

4.3 Recovery Using Killi

Killi's robust fault classification methodology is driven by the requirement that Killi should identify the correct number of faults per cache line without external guidance (MBIST). Therefore, at reset, Killi marks all caches lines as having an unknown number of faults (state b'01). When storing data into lines with state b'01, both the ECC and 16 parity bits are generated. Upon the first eviction or cache hit, both ECC and parity determine the number of faults and the DFH bits are updated accordingly. Note that updating the DFH bits does not require additional (or forced) read or write accesses to the cache. For lines that have zero LV failures, Killi generates only 4 bits of parity to detect non-LV faults (soft-errors or ageing related failures). For lines with one failure, both Killi generates ECC and parity to detect combinational failures (LV, aging or soft-errors) with higher probability.

Because the ECC cache stores the error recovery metadata for all cache lines not in a stable state (b'01), there is heavy contention for ECC cache space during cache warmup. As ECC cache contention increases, cache evictions rise as well, causing additional cache misses. This happens because the ECC cache is much smaller than the cache, and thus addresses from disjoint cache sets store their checkbits in the same ECC cache set. As a result, an eviction in the ECC cache can cause an address from a disjoint cache set to be evicted. As cache lines are accessed (cache hits) or evicted, Killi discovers lines with no errors (that no longer require checkbits to be stored in the ECC cache). This reduces the number of cache misses due to ECC cache evictions[6].

Upon the first load hit to a cache line (in initial DFH state of b'01), the cache controller triggers one of the five actions listed in Table 2. When reading the data from the cache, Killi checks for errors using both segmented parity and ECC. Because the DFH bits are stored in the tag array, the controller conditionally accesses the ECC cache to retrieve the checkbits and generate the syndrome and global parity. The ECC cache access latency is hidden under the cache data access and does not lengthen the L2 hit latency. If both parity and ECC agree that the line only has one LV fault, then the DFH bits are update to b'10. If segmented parity determines that the parity mismatches in more than one segment, it disables the line by changing the DFH bits to b'11.

[5] The terms *failure* and *error* are used interchangeably.

[6] Note that the process of training the DFH bits happens once per reset cycle and not on context switches.

Current DFH	S.Parity	Syndrome	G.Parity	Next DFH	Action	Comment
00	✓	N/A	N/A	00	Send clean line	No error
00	✗	N/A	N/A	01	Signal error-induced cache miss; Invalidate line and trigger new load request	1-bit error discovered after training; Initial classification incorrect
00	✗✗	N/A	N/A	11	Signal error-induced cache miss; Disable line and trigger new load request	Multi-bit error discovered after training
01	✓	✓	✓	00	Invalidate entry in ECC cache; Send clean line	No Error. Most frequent scenario
01	✗	✗	✗	10	Correct error using checkbits stored in ECC cache; Send clean line	1-bit LV error
01	✓ or ✗✗	✗	✓	11	Signal error-induced cache miss; Disable line and trigger new load request	Multi-bit error
01	✗✗	✓ or ✗	✓	11	Signal error-induced cache miss; Disable line and trigger new load request	Even number of errors
01	✗✗	✓ or ✗	✗	11	Signal error-induced cache miss; Disable line and trigger new load request	Odd number of multi-bit errors
10	✓	✓	✓	00	Invalidate entry in ECC cache; Send clean line	Non-LV transient error that was subsequently overwritten
10	✗ or ✗✗	✓	✓	11	Signal error-induced cache miss; Disable line and trigger new load request	Likely non-LV error + LV error
10	Don't Care	✗	✗	10	Correct error using checkbits stored in ECC cache; Send clean line	Single-bit LV error
10	✗✗	✗	✓	11	Signal error-induced cache miss; Disable line and trigger new load request	Error on line with existing 1-bit LV error.
10	✗✗	✓	✗	11	Signal error-induced cache miss; Disable line and trigger new load request	Error on line with existing 1-bit LV error
11	N/A	N/A	N/A	N/A	None	Line disabled; replacement/allocation policy will not select line

Table 2: State transitions for reliable LV cache operation under single-bit and multi-bit errors leveraging Killi. S.Parity and G.Parity indicate Segmented Parity and Global Parity (ECC) respectively. A checkmark in syndrome indicates a zero syndrome and a cross indicates a non-zero syndrome. A checkmark in parity indicates a match and cross indicates a mismatch in one segment. A double cross indicates a mismatch in at least two segments.

Orthogonally, ECC will detect cases where both failures occur in the same segment (which parity will fail to detect).

Disabled lines are never accessed/allocated again until the next reset. A line can be temporarily disabled because of the combination of one LV fault and one soft error (or a two-bit soft-error). This line will be reclaimed only after resetting the DFH bits[7]. Upon resetting of the DFH bits (b'01), the line will be checked with ECC and parity. If ECC and parity agree that there is only one error, then the line transitions to b'10 state. In case of a previous two-bit soft-error, parity and ECC may detect zero errors after reset and Killi will mark the line as fault-free (DFH bits set to b'00).

As mentioned previously, Killi learns on the fly and adapts across the lifetime of the chip. For lines that are found to have zero LV faults, the DFH bits transition from b'01 to b'00. This also entails invalidating the line in the ECC cache. Because most lines do not have a LV failure, this scenario is the most likely to happen. It can also incorrectly transition to b'00 if there is a masked fault. A masked fault is a type of fault where a fault is hidden and is not responsible for corrupting the data. Similarly, a single-bit soft-error can

[7] DFH reset occurs when voltage changes or on a reboot. Disabled lines due to soft errors can also be reclaimed by a scrubber.

transition a line from b'01 to b'10. Therefore, Killi will not always characterize faults correctly on the first reference to the line. Upon subsequent writes to that line, the error may disappear making the line error-free with respect to LV errors (or unmask the fault in case of a previously masked fault). In such situations, the output of ECC is compared in tandem with segmented parity to transition the line from b'10 to b'00. In summary, Killi inherently allows the DFH bits to oscillate and adapt to the different failures that may occur between resets of the history bits. This helps it recover from incorrect initial classifications.

4.4 Optimized Allocation and Replacement Strategies

In order to reduce the possibility of evicting data with high reuse, Killi coordinates the replacement policies of the cache and the ECC cache. Whenever a cache line with one fault is touched and promoted to the MRU position, its associated data in the ECC cache is also promoted. As previously mentioned, DFH state warmup can result in lower performance due to ECC cache contention and reducing the ECC cache size exacerbates the contention. In order to reduce this performance impact, Killi optimizes the cache eviction policy to speed up DFH state warmup. Specifically, for evicted lines in state b'01, Killi reads out the evicted data and generates the segmented parity and checkbits. Then Killi compares the stored values and updates the DFH bits appropriately. Note that the DFH bits for a given line are persistent across data block evictions.

To further expedite the training of DFH bits, Killi uses a modified replacement policy that prioritizes installing data into invalid lines with the following DFH: b'01 > b'00 > b'10. As a result, Killi prioritizes allocating into lines in the initial transient state (DFH state b'01), which as discussed, accelerates fault characterization. Furthermore, Killi among a group of available lines prioritizes allocating to lines with b'00 over b'10 to reduce the probability of Silent Data Corruptions (SDC) due to a combination of soft-errors and LV faults.

5. EXPERIMENTAL EVALUATION

5.1 Methodology

We evaluated the performance impact of Killi by modifying the gem5 execution-driven simulator [26]. The gem5 simulator has been augmented with a GPU model that executes GCN3 ISA binaries [27]. We added Killi on top of the GPU L2 cache model and Table 3 shows the overall GPU configuration. Because Killi leverages only parity and SECDED ECC for fault characterization and recovery, Table 3 provides the modeled latencies for those circuits. In our experiments, we examine ten HPC GPGPU workloads and we sum the total cycles across all GPU kernels as our performance metric. In addition to Killi, we evaluate systems that use Double Error Correction, Triple Error Detection (DECTED), FLAIR and MS-ECC [7], [22]. For the DECTED, FLAIR and MS-ECC systems, we assume a pre-characterization phase (MBIST) where each line in the cache is bitmapped and flagged either as enabled or disabled. For those systems, the reported execution times only includes the GPU runtime and does not include the pre-characterization phase. Because Killi does not rely on a separate pre-characterization phase, Killi's GPU execution time includes the DFH bit updates. Finally, we assume only the L2 cache operating voltage is modified and its frequency (1GHz) matches with the rest of the GPU. We use the fault rates obtained when running the SRAM arrays at 1GHz in our performance evaluation.

Parameter	Value
Number of CUs	8
GPU Frequency	1GHz
L1 size/CU	16KB
L2 size	2MB
L2 associativity	16
L2 banks	16
L1/L2 line size	64B
ECC cache associativity	4
L2 V_{DD}	$0.625\times V_{DD}$
L2 tag latency	2 cycles
L2 data latency	2 cycles
ECC cache tag latency	1 cycle
ECC cache data latency	1 cycle
SECDED/Parity latency	1 cycle
ECC cache line size	41 bits

Table 3: GPU hardware configuration.

5.2 Performance Comparison

Figure 4 shows the performance impact normalized to a baseline fault-free system. The baseline fault-free system is operating under nominal V_{DD}. FLAIR uses SECDED codes (for protecting lines with 1-bit failure) and disables all lines with 2 or more failures. For DECTED, we disable all lines with 3 or more failures. MS-ECC can correct up to 11 errors in a 64B line. We evaluate Killi for a range of ECC cache sizes. The ECC cache size is provided as the ratio of ECC cache lines over the L2 cache lines.

As mentioned previously, at $0.625\times V_{DD}$, the majority of the lines have either 0 or 1 failure. Because all of the techniques can correct at least one failure, the majority of the cache lines are enabled and thus the performance degradation is minimal. The additional number of lines with two failures that are enabled with DECTED and FLAIR, but not Killi, is very small and does not offer significant performance benefits. While MS-ECC can help lower the V_{min} further below $0.625\times V_{DD}$, it incurs the highest area penalty that might not justify the additional benefits of improved energy efficiency with lowering of V_{min}. For 8 out of the 10 applications, Killi incurs a performance penalty of less than 1% across all ECC cache sizes. For the remaining two (XSBENCH and FFT), the performance penalty is a maximum of 2.4% and 5% respectively when the ECC cache size is lowest (656B for the 1:256 ratio). It should be noted that this configuration has the lowest area overhead among all the techniques. For the largest ECC cache size (10.25KB

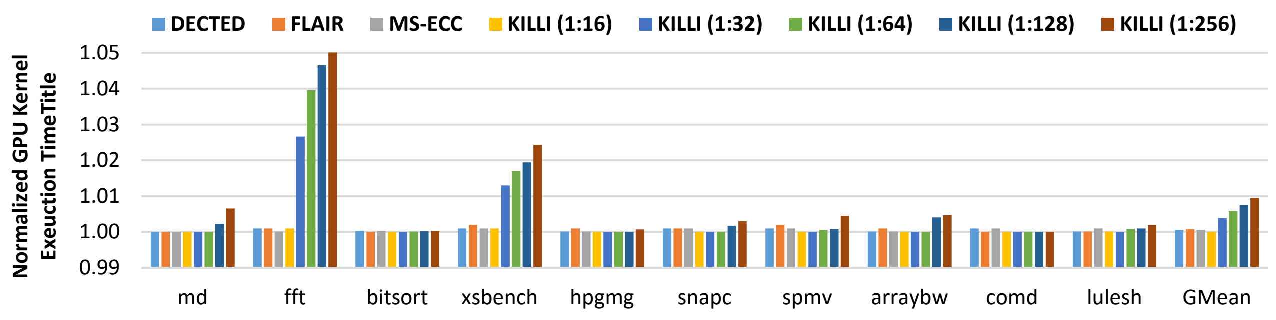

Figure 4: GPU Kernel execution time normalized to a baseline fault-free system operating at 1.0×V$_{DD}$. Comparisons for varying Killi's ECC cache size is also included. DECTED, FLAIR, MS-ECC, and Killi operating at 0.625×V$_{DD}$.

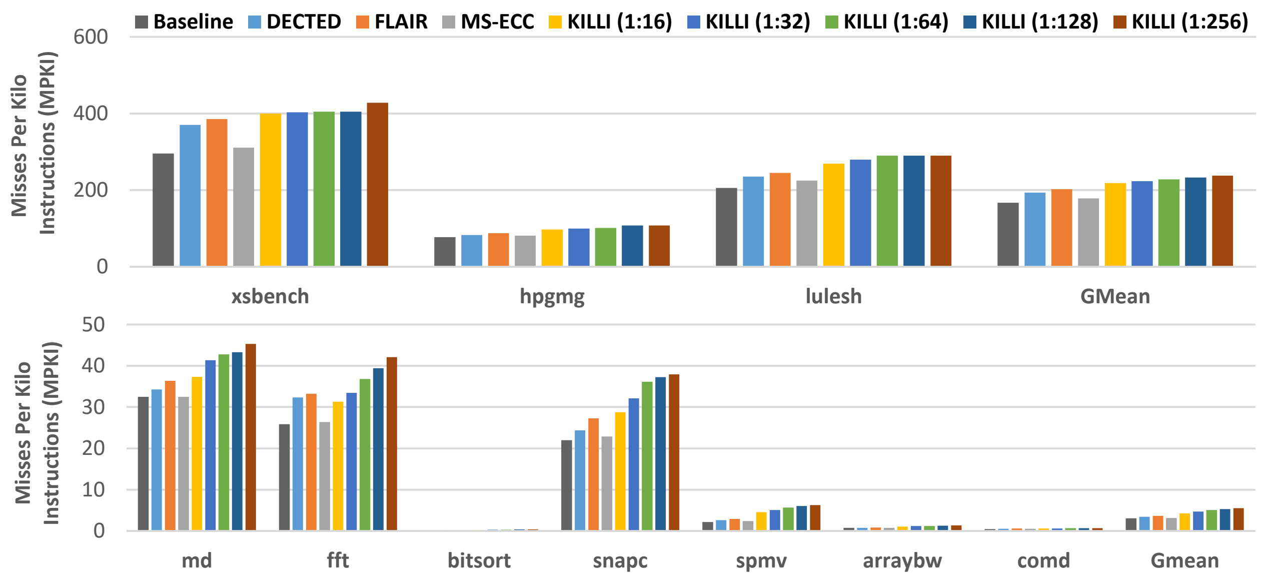

Figure 5: Misses-per-kilo-instructions (MPKI) for the GPU L2 cache implemented with Killi. Comparisons for varying ECC cache sizes is also provided. Killi, DECTED, FLAIR and MS-ECC operating at 0.625×V$_{DD}$.

for the 1:16 ratio), Killi's performance is similar to the baseline system that operates at nominal V$_{DD}$.

Figure 5 measures misses-per-kilo-instruction (MPKI) across the ten applications to determine if the performance degradation corresponds to an increase number of cache misses. We observed that ECC cache contention and subsequent ECC-cache-induced L2 cache replacements cause more L2 cache misses. The top plot of Figure 5 includes compute-bound applications that have a MPKI lower than 50, while the bottom plot includes memory-bound applications with a MPKI greater than 100. Because MS-ECC can correct the greatest number of errors offering the highest effective L2 capacity, it achieves the L2 miss rate closest to the baseline fault-free system. The difference in MPKI between the 1:16 and 1:256 configurations (35% in FFT and 10% in XSBENCH respectively) is due to: (a) disabling lines with 2 or more faults, (b) not using the subset of lines with one fault that cannot be protected with SECDED checkbits due to limited ECC cache size, and (c) the training phase during which there is increased contention in the ECC cache. As can be seen in Figure 4 and Figure 5, performance under Killi can be regulated by varying the ECC cache size. This is because larger ECC caches allow more L2 cache lines with one failure to reliably hold data for a given voltage and frequency increasing dynamically the total usable cache capacity at any given time.

As stated above, Killi loses performance due to online training of the DFH bits. In comparison, FLAIR uses a dynamic MBIST technique, which periodically sacrifices more than half of the cache capacity in order to detect failures at a certain LV and suffers from higher L2 miss traffic during these training phases. This overhead is not included in our results because we skip training for the simulations with FLAIR and pre-train their DFH bits.

One strategy for lowering L2 cache MPKI (to the levels closer to that of MS-ECC) is to adopt stronger error protection and enable lines with 2-bit failures as well. This can enable operation at voltages lower than 0.625×V$_{DD}$. A potential solution would be to use DECTED in each ECC cache line instead of SECDED. With Killi, this can be accomplished at no additional storage area cost. Because the 12 bits (of parity in ECC cache) remain unused after the training phase, we can combine them with 11 bits of ECC to store the checkbits of a stronger ECC that requires no more than 23'b. Therefore, it is possible to augment Killi such that it uses SECDED+16 bits of parity during training and transition to DECTED+4 bits of parity during regular operation. DECTED ECC for 64B data requires only 21 bits for checkbits which is within the available 23 bits per ECC cache line. The additional cost of the extra DECTED encoder/decoder is small.

5.3 Ensuring Correct Classification

In this section we demonstrate Killi's ability to correctly characterize a cache line with a certain probability. Since Killi disables all lines with two or more failures, Killi only needs to know if a line has a multi-bit failure or not. In other words, Killi does not need to determine the exact number of multi-bit failures. However, the challenge is that SECDED codes can only detect up to 2 errors in a cache line (and specific combinations of more than 2 errors). For the sake of simplicity, we will assume that SECDED ECC will fail for all cases where the line has more than two failures. Therefore, the onus is on our segmented parity scheme to correctly detect multi-bit failures of 3 or more.

In Killi, each line is partitioned into 16 segments with 1-bit parity protection per segment. When the DFH bits are in the initial state b'01, every line is protected with SECDED ECC as well. Segmented parity will fail to protect a cache line when there is a combination of at most 1 segment with >=2 failures and all remaining segments each experience an even number of failures. It will also fail when there is a combination of segments with no error and the remaining segments each experience an even number of errors. Since segmented parity and SECDED ECC function independently, Killi as a whole will fail only when both segmented parity and SECDED ECC fail. The probability of Killi failing, $P_{fail}Killi$ for a 512-bit line can be derived as follows,

$$P_{fail}Killi = P_{fail}SECDED * P_{fail}Seg.Parity$$

$$P_{fail}SECDED = \sum_{k=3}^{523} \binom{523}{k} {P_{cell}}^{k} (1 - P_{cell})^{523-k}$$

where P_{cell} is the SRAM cell failure probability at a given V_{DD}.

SECDED ECC requires 11 checkbits to protect 523-bits of data (512 bits of data and 11 ECC checkbits). We assume that the checkbits can also fail under LV. To estimate $P_{fail}Killi$, we partition a line into 16 segments, each 4-bytes wide.

We generate a 1-bit parity for each segment making the segment length 33 bits. We assume that each of the 1-bit parity can also fail under LV. We denote the probability of having zero, even number (>=2) or odd number of failures (>=3), in exactly one segment, as $P_{seg}0$, $P_{seg}Even$ and $P_{seg}Odd$ respectively.

$$P_{seg}0 = (1 - P_{cell})^{33}$$

$$P_{seg}Even = \sum_{i=2}^{33} \binom{33}{i} {P_{cell}}^{i} (1 - P_{cell})^{33-i} \quad i \in (2, 4 \ldots 32)$$

$$P_{seg}Odd = \sum_{i=3}^{33} \binom{33}{i} {P_{cell}}^{i} (1 - P_{cell})^{33-i} \quad i \in (3, 5, \ldots 33)$$

The probability of having 0 or even number of errors in n segments (in a total of 16 segments) is denoted as $P_{seg}^{n}0$ and $P_{seg}^{n}Even$ respectively and can be derived as,

$$P_{fail}Seg.Parity = \left(P_{seg}^{15}0 * P_{seg}Odd\right) + \sum_{i=0}^{15} (P_{seg}^{16-i}Even * P_{seg}^{i}0)$$

Based on the above equations, the probability of segmented parity failing can be estimated as,

$$P_{seg}^{n}0 = \binom{16}{n} (P_{seg}0)^{n} (1 - P_{seg}0)^{16-n}$$

$$P_{seg}^{n}\text{Even} = \binom{16}{n} (P_{seg}Even)^{n} (1 - P_{seg}Even)^{16-n}$$

Using the above estimates for the failure probability of SECDED and segmented parity, we estimate the Killi failure probability and calculate the fault coverage of Killi as,

$$Killi_{coverage} = \left(1 - P_{fail}Killi\right) * 100$$

Figure 6 shows the percentage of lines with correct classification across a range of voltages. We assume here that DECTED codes can detect up to three errors[8]. For MS-ECC, we assume a configuration where we can detect up to 11 errors in a 64B line. To better understand the theoretical benefits provided by each of the techniques, we assume that there is no MBIST (and pre-characterization). Pre-

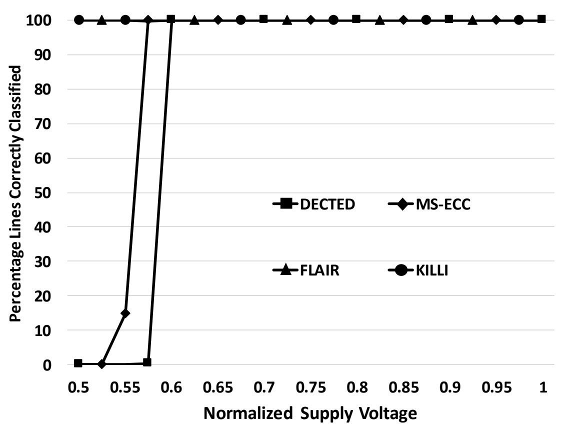

Figure 6 Percentage of cache lines where a single-bit or a multi-bit LV error can be detected correctly.

[8] We do not consider the case where DECTED may detect some groups of four or more errors.

characterization is time consuming and cannot be discounted as explained by Qureshi *et al.* [7]. As we scale down the supply voltage, the number of errors increase, and the fault coverage reduces. Up to 0.6×V_{DD}, all techniques correctly classify all lines in the cache. For lower voltages, only Killi and FLAIR have the capability of providing near 100% coverage. For Killi, this is primarily driven by the combined parity and SECDED ECC protection. By using 16-bit parity per cache line, Killi successfully detects any number of multi-bit errors in a line as long as there are at least two segments with odd number of errors. When segmented parity fails, SECDED ECC successfully detects the two errors ensuring that Killi succeeds in correctly detecting the number of faults per line. Further, in Killi, the fault coverage is independent of the size of the ECC cache.

FLAIR is the only other technique that can provide near 100% fault coverage similar to Killi during training. FLAIR uses a combination of DMR and SECDED for fault characterization and disables lines with more than one failure. After characterization, it relies on SECDED for error detection and correction. Thus, it is unable to cope with multi-bit soft-errors on a line with one-bit failure and leads to a higher failure probability after training. Furthermore, FLAIR relies on using DMR for a portion of the cache to allow LV operation while training the remaining portion using MBIST. By disabling two ways (in a 16-way cache) for testing and using DMR in the remaining 14-ways, the cache capacity is effectively 7/16 of the original capacity. This can cause significant performance degradation in large caches not just because of the reduced cache capacity. As the number of lines to be tested increases with cache size, there is increased contention for cache bandwidth between demand requests to the cache and requests emanating from MBIST testing.

5.4 Area and Energy Overhead

Table 5 shows the estimated area overhead for Killi and compare it to other LV error-protection techniques. Killi's area overhead includes the ECC cache (Tag + Data) and the parity and DFH bits in the L2 cache. All other error protection techniques include in their area overhead one bit per L2 cache line to enable disabling lines that cannot be protected. The Killi area overhead is shown for different ratios of ECC cache lines over L2 cache lines. For a 2MB L2, the Killi area overhead ranges from 24.6KB (1:256) to 34.25KB (1:16). The third row of Table 5 shows the storage area ratios normalized to SECDED ECC (used by FLAIR). The fourth row shows the area overhead of each error protection

	Killi (variants based on ratio of ECC cache lines to L2 cache lines)				
	1:256	**1:128**	**1:64**	**1:32**	**1:16**
DECTED	0.51	0.53	0.55	0.61	0.71
TECQED	0.52	0.54	0.58	0.66	0.82
6EC7ED	0.53	0.56	0.62	0.74	0.97

Table 4: Storage area of Killi using DECTED, TECQED and 6EC7ED ECC. Area is normalized to storage area of SECDED ECC.

mechanism over the original L2 size. It is clear from the table that Killi offers the lowest area overhead compared to prior techniques. Killi offers a unique proposition in that it allows a trade-off between area, performance and energy centered around the sizing of the ECC cache. By reducing the number of ECC cache entries, it is possible to reduce the area overhead to a minimum of 0.51×.

Table 4 shows the storage overhead of Killi with different strength ECC codes versus using SECDED ECC per L2 cache line (adopted for LV operation by adding a bit per line to mark disabled lines). As mentioned previously, Killi's area overhead scales with only with ECC cache size and ECC code strength. Kill can integrate any type of ECC and. allows more L2 cache lines to be enabled under LV at a much lower area cost. Because we allocate only 4 bits of parity per cache line, we can dedicate additional storage to either have stronger ECC (enabling lines with multi-bit faults) or upsize the ECC cache enabling more lines with a single LV fault.

As the error correction overhead is independent of the L2 cache size, for applications that can tolerate moderate performance loss, the smallest ECC cache configuration (1:256 ratio) can be used when there are a small number of lines each with multiple-bit LV failures. For comparison, when Killi is coupled with an ECC cache storing 6EC7ED ECC for one out of 16 L2 cache lines, Killi has lower area overhead than using SECDED ECC protection per L2 cache line for the same size (2MB) L2 cache as shown in Table 4. Because we use ECC + 16 bits of parity during training for all lines and the coverage capability is essentially the ability to detect single and multi-bit failures during this time, by storing part of the parity data in the ECC cache we help limit the area overhead tremendously. Most other techniques provision every line in the cache with some form of protection and there is potential wastage of area from two sources: a) Lines that have zero LV failures which would otherwise require only error detection capability and b) Lines with multi-bit LV failures that are disabled. The additional storage allocated for storing checkbits in these lines are not used. Killi mitigates such issues by generating and storing the checkbits in the ECC cache on-demand.

Table 6 shows the estimated normalized power (in %) across different techniques. The estimates are normalized to a baseline fault-free system operating at nominal V_{DD}. Also, the provided estimates account only for the power consumption of the L2 data and tag arrays and do not take into consideration the L2 cache controller and its associated queues. Additionally, for Killi, we include the power consumption of the ECC cache, memory accesses on account of cache misses due to contention in the ECC cache and reading of cache lines during evictions (for lines with DFH b'01). All techniques except for MS-ECC have similar power consumption profiles. The difference in power consumption stems from the difference in power consumed due to the checkbits, ECC encoding/decoding circuitry and memory accesses on account of reduced cache capacity (due to disabled lines). Due to the low number of failures at 0.625*V_{DD}, there is no significant loss in cache capacity

Protection Scheme	DECTED	MS-ECC	SEC DED	Killi (ratio of ECC cache lines to L2 cache lines)				
				1:256	1:128	1:64	1:32	1:16
Ratio	1.9	18	1	0.51	0.52	0.55	0.60	0.71
% area over L2	4.3%	38.6%	2.3%	1.2%	1.23%	1.29%	1.42%	1.67%

Table 5: Area comparison across error protection techniques. Area is normalized to SECDED ECC's area in 2nd row. Additional area required to protect a 2MB L2 cache with each mechanism is shown in 3rd row.

Existing Schemes			Killi (ratio of ECC cache lines to L2 cache lines)				
DECTED	MS-ECC	FLAIR	1:256	1:128	1:64	1:32	1:16
43.7	55.3	42.6	40.3	40.7	41.1	41.7	42.4

Table 6: Power consumption (in %) normalized to fault-free cache operating at nominal V_{DD}. All techniques operate at 0.625×V_{DD} and 1GHz.

resulting in a trivial increase in cache misses. As a result, the power consumption of the smallest (1:256) Killi configuration is lower than the largest (1:16) configuration. The power consumed by the ECC cache is trivial because: a) it is accessed only for lines in b'10 (during training) and b'01 (lines with 1 LV failure), and b) the size of the ECC cache line is much smaller (23 bits) compared to a 64B cache line.

5.5 Optimizing For Lower V_{min}

To demonstrate Killi's scalability to lower voltages and its ability to seamlessly integrate disparate ECC schemes, we evaluate Killi at 0.6×V_{DD} and 0.575×V_{DD}, and compare it with MS-ECC, which provides the absolute lowest V_{min} amongst all techniques considered. To run at such low voltages, both Killi's ECC cache and MS-ECC must use ECC based on Orthogonal Latin Square Codes (OLSC). In comparison, Killi's standard ECC cache uses SECDED codes, though Killi's parity support remains unchanged.

Table 7 summarizes the achieved L2 cache capacity using OLSC codes (second column) and Killi's storage savings (third column). For Killi to provide the same cache capacity, performance, and reliability as MS-ECC at 0.6×V_{DD} and 0.575×V_{DD}, its ECC cache is sized to protect one of out eight and two lines, respectively. In contrast, MS-ECC provisions all cache lines with ECC. As a result, Killi requires far less area (21% and 72% of the MS-ECC area for each voltage level) and can incorporate the stronger ECC without significant architecture changes.

5.6 Current Limitations of Killi

5.6.1 Supporting Write-Back Operation

While this paper discusses an implementation of Killi for write-through caches, it can support write-back caches with minor modifications. One implementation would select error protection of lines storing dirty (modified) data based on the DFH of the line. For example, if dirty data are stored in a line with DFH of b'00 (no LV error), Killi can protect the data with SECDEC ECC in the ECC cache in order to match the failure probability of a safe voltage cache that uses SECDED ECC. If dirty data are stored in a line with DFH of b'10 (one LV error) then Killi employs DECTED ECC in the ECC cache to match the failure probability of a safe voltage cache protecting dirty data with SECDED ECC. Note that Killi can use DECTED ECC for lines with DFH of b'10 without increasing its area, by adding the ECC cache's unused 12 parity bits to its 11-bit SECDED code to store DECTED's 21-bit correcting code. This implementation increases ECC cache contention for lines with DFH of b'00 and holding dirty data. In order to reduce the resulting cache misses from higher ECC cache contention, dirty L2 cache lines can be controlled with techniques proposed by Khan *et al.* [28].

5.6.2 Handling Multi-Bit Masked Faults

Multi-bit masked faults are particularly problematic for error correction schemes because ECC and parity cannot immediately detect them. Consequently, prior schemes have relied on MBIST to detect these faults, but since Killi lacks MBIST, Killi must handle multi-bit masked faults by other means. Specifically consider the scenario where a two-bit masked LV fault occurs in the same 128-bit segment. Upon classification, this line would be classified as having no-errors and therefore will not require ECC support for subsequent accesses. However, upon subsequent writes to the line, the two-bit fault may unmask and result in a potential SDC (as both errors are in same segment, parity will also fail to detect). We determined the probability of such a scenario to be 0.003%. In other words, for 99.997% of lines, when operating at 0.625V_{DD}, Killi will protect against such type of fault scenarios.

A simple mitigation technique is to augment error checking for the inverted state of data with the following flow: write original data → read original data (error check) → write inverted data → read inverted data (error check). The first write is done when data is first installed in a L2 cache line. The two reads are accomplished either by demand accesses or by a single demand access and data eviction. The only extra flow is the second write. We train the DFH bits only after the error check of the second read. This would require one additional bit to track the polarity (inverted versus non-inverted) and minor-modifications to Killi's FSM. Any performance loss would result from increased ECC cache contention to protect a line until all masked faults are detected.

Voltage	% L2 Capacity Target	Killi (w/OLSC)
0.6*VDD	99.8%	17%
0.575*VDD	69.6%	65%

Table 7: Storage area overhead of Killi normalized to MS-ECC for achieving target cache capacity at 0.6×V$_{DD}$ and 0.575×V$_{DD}$, respectively.

6. CONCLUSIONS

In this paper, we present Killi, a scalable error protection scheme that enables LV caches at a significantly lower area than previous techniques. Killi's decoupled approach provisions parity-based error detection for all cache lines and provides on-demand ECC-based error-correction for faulty lines. The checkbits for on-demand ECC are stored in a separate set-associative structure and are only used by cache lines that encounter LV faults. Our results show that when the GPU L2 operates at 0.625×V$_{DD}$, Killi can reduce the GPU L2 power consumption by as much as 59.3%. Furthermore, Killi only incurs a modest performance penalty of 0.8% compared to a fault-free system operating at nominal V$_{DD}$ because it targets peak frequencies. At the same time, the error protection area overhead can be reduced by 50% compared to SECDED. Finally, Killi allows online characterization of LV fault populations without using additional MBIST testing. It utilizes a combination of parity and ECC to identify LV faults during program execution and takes advantage of LV fault masking to enable a higher number of cache lines than full knowledge of faults would allow. Thus, Killi is easier to productize and more likely to enable LV caches in future chips.

7. ACKNOWLEDGMENTS

We thank the anonymous reviewers for their insightful comments and feedback. We also thank Vilas Sridharan and Sooraj Puthoor for their participation in many constructive discussions. AMD, the AMD Arrow logo, Radeon, and combinations thereof are trademarks of Advanced Micro Devices, Inc. This research was supported by the US Department of Energy (DOE) under the PathForward program. Any opinions, findings and conclusions or recommendations expressed herein are those of the authors and do not necessarily reflect the views of the DOE.

REFERENCES

[1] J. Hestness, S. W. Keckler, and D. A. Wood, "A comparative analysis of microarchitecture effects on CPU and GPU memory system behavior," in *2014 IEEE International Symposium on Workload Characterization (IISWC)*, 2014, pp. 150–160.

[2] "AMD GPU LLVM Memory Model." https://llvm.org/docs/AMDGPUUsage.html#memory-model.

[3] C. Wilkerson, H. Gao, A. R. Alameldeen, Z. Chishti, M. Khellah, and S. L. Lu, "Trading off Cache Capacity for Reliability to Enable Low Voltage Operation," in *2008 International Symposium on Computer Architecture*, 2008, pp. 203–214.

[4] A. R. Alameldeen, I. Wagner, Z. Chishti, W. Wu, C. Wilkerson, and S. L. Lu, "Energy-efficient cache design using variable-strength error-correcting codes," in *2011 38th Annual International Symposium on Computer Architecture (ISCA)*, 2011, pp. 461–471.

[5] D. H. Yoon and M. Erez, "Flexible cache error protection using an ECC FIFO," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, 2009, pp. 1–12.

[6] H. Duwe, X. Jian, and R. Kumar, "Correction prediction: Reducing error correction latency for on-chip memories," in *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, 2015, pp. 463–475.

[7] M. K. Qureshi and Z. Chishti, "Operating SECDED-based caches at ultra-low voltage with FLAIR," in *2013 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2013, pp. 1–11.

[8] S. Ganapathy, J. Kalamatianos, K. Kasprak, and S. Raasch, "On characterizing near-threshold SRAM failures in FinFET technology," in *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2017, pp. 1–6.

[9] G. K. Chen, D. Blaauw, T. Mudge, D. Sylvester, and N. S. Kim, "Yield-driven near-threshold SRAM design," in *ICCAD*, 2007.

[10] M. Khellah *et al.*, "Wordline Bitline Pulsing Schemes for Improving SRAM Cell Stability in Low-Vcc 65nm CMOS Designs," in *2006 Symposium on VLSI Circuits, 2006. Digest of Technical Papers.*, 2006, pp. 9–10.

[11] M. M. Khellah, A. Keshavarzi, D. Somasekhar, T. Karnik, and V. De, "Read and write circuit assist techniques for improving Vccmin of dense 6T SRAM cell," in *2008 IEEE International Conference on Integrated Circuit Design and Technology and Tutorial*, 2008, pp. 185–188.

[12] Y. Pan, J. Kong, S. Ozdemir, G. Memik, and S. W. Chung, "Selective wordline voltage boosting for caches to manage yield under process variations," in *2009 46th ACM/IEEE Design Automation Conference*, 2009, pp. 57–62.

[13] J. P. Kulkarni, A. Goel, P. Ndai, and K. Roy, "A Read-Disturb-Free, Differential Sensing 1R/1W Port, 8T Bitcell Array," *IEEE Trans. Very Large Scale Integr. VLSI Syst.*, vol. 19, no. 9, pp. 1727–1730, Sep. 2011.

[14] I. J. Chang, J. J. Kim, S. P. Park, and K. Roy, "A 32kb 10T Subthreshold SRAM Array with Bit-Interleaving and Differential Read Scheme in 90nm CMOS," in *2008 IEEE International Solid-State Circuits Conference - Digest of Technical Papers*, 2008, pp. 388–622.

[15] S. P. Park, S. Y. Kim, D. Lee, J. J. Kim, W. P. Griffin, and K. Roy, "Column-selection-enabled 8T SRAM array with ~ 1R/1W multi-port operation for DVFS-enabled processors," in *IEEE/ACM International*

Symposium on Low Power Electronics and Design, 2011, pp. 303–308.

[16] M. Clinton, R. Singh, M. Tsai, S. Zhang, B. Sheffield, and J. Chang, "A 5GHz 7nm L1 cache memory compiler for high-speed computing and mobile applications," in *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, 2018, pp. 200–201.

[17] M. Gottscho and others, "Power/Capacity Scaling: Energy Savings With Simple Fault-Tolerant Caches," in *Proc. ACM DAC '14*, 2014.

[18] S. M. Khan, A. R. Alameldeen, C. Wilkerson, J. Kulkarni, and D. A. Jiménez, "Improving multi-core performance using mixed-cell cache architecture," in *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, 2013, pp. 119–130.

[19] P. P. Shirvani and E. J. McCluskey, "PADded cache: a new fault-tolerance technique for cache memories," in *Proceedings 17th IEEE VLSI Test Symposium (Cat. No.PR00146)*, 1999, pp. 440–445.

[20] A. Ansari, S. Feng, S. Gupta, and S. Mahlke, "Archipelago: A Polymorphic Cache Design for Enabling Robust Near-threshold Operation," in *Proceedings of the 2011 IEEE 17th International Symposium on High Performance Computer Architecture*, Washington, DC, USA, 2011, pp. 539–550.

[21] J. Chang *et al.*, "The 65-nm 16-MB Shared On-Die L3 Cache for the Dual-Core Intel Xeon Processor 7100 Series," *IEEE J. Solid-State Circuits*, vol. 42, no. 4, pp. 846–852, Apr. 2007.

[22] Z. Chishti, A. R. Alameldeen, C. Wilkerson, W. Wu, and S.-L. Lu, "Improving cache lifetime reliability at ultra-low voltages," in *MICRO*, 2009.

[23] T. N. Miller, R. Thomas, J. Dinan, B. Adcock, and R. Teodorescu, "Parichute: Generalized Turbocode-Based Error Correction for Near-Threshold Caches," in *2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture*, 2010, pp. 351–362.

[24] M. Manoochehri, M. Annavaram, and M. Dubois, "CPPC: Correctable parity protected cache," in *2011 38th Annual International Symposium on Computer Architecture (ISCA)*, 2011, pp. 223–234.

[25] J. Maiz, S. Hareland, K. Zhang, and P. Armstrong, "Characterization of multi-bit soft error events in advanced SRAMs," in *IEEE International Electron Devices Meeting 2003*, 2003, pp. 21.4.1-21.4.4.

[26] N. Binkert *et al.*, "The Gem5 Simulator," *SIGARCH Comput Arch. News*, vol. 39, no. 2, pp. 1–7, Aug. 2011.

[27] A. Gutierrez *et al.*, "Lost in Abstraction: Pitfalls of Analyzing GPUs at the Intermediate Language Level," in *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2018, pp. 608–619.

[28] S. Khan, A. R. Alameldeen, C. Wilkerson, O. Mutluy, and D. A. Jimenezz, "Improving cache performance using read-write partitioning," in *2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*, 2014, pp. 452–463.