Boosting Chipkill Capability under Retention-Error Induced Reliability Emergency

Xianwei Zhang*
University Pittsburgh
Pittsburgh, PA, USA
xiz81@pitt.edu

Rujia Wang†
University of Pittsburgh
Pittsburgh, PA, USA
rujia.w@pitt.edu

Youtao Zhang, Jun Yang
University of Pittsburgh
Pittsburgh, PA, USA
zhangyt@cs.pitt.edu,juy9@pitt.edu

ABSTRACT

The DRAM based main memory of high embedded systems faces two design challenges: (i) degrading reliability; and (ii) increasing power and energy consumption. While chipkill ECC (error correction code) and multi-rate refresh may be adopted to address them, respectively, a simple integration of the two results in 3× or more SDC (silent data corruption) errors and failing to meet the system reliability guarantee. This is referred to as *reliability emergency*.

In this paper, we propose PlusN, a hardware-assisted memory error protection design that adaptively boosts the baseline chipkill capability to address the reliability emergency. Based on the error probability assessment at runtime, the system switches its memory protection between the baseline chipkill and PlusN — the latter generates a stronger ECC with low storage and access overheads. Our experimental results show that PlusN can effectively enforce the system reliability guarantee under different reliability emergency scenarios.

KEYWORDS

DRAM, Reliability, Chipkill, Refresh, Power

ACM Reference Format:
Xianwei Zhang, Rujia Wang, and Youtao Zhang, Jun Yang. 2019. Boosting Chipkill Capability under Retention-Error Induced Reliability Emergency. In *ASPDAC '19: 24th Asia and South Pacific Design Automation Conference (ASPDAC '19), January 21–24, 2019, Tokyo, Japan.* ACM, New York, NY, USA, Article 4, 6 pages. https://doi.org/10.1145/3287624.3287639

1 INTRODUCTION

Today's high end embedded systems often integrate large capacity main memory to meet the increasing memory demands of modern applications, e.g., AMD Opteron 4300 series embedded processors [1] adopt socket C32 to support four DDR3 memory channels. However, scaling DRAM cell in deep sub-micron regime faces severe reliability and leakage problems. On the one hand, field studies have shown that large capacity memory frequently encounters memory errors [4, 20, 21], which demand strong ECC (error correction code) such as chipkill protection to ensure high system reliability and availability. On the other hand, the DRAM power demand grows rapidly with increasing DRAM module numbers and shrinking feature sizes [11, 12].

Enforcing high memory reliability demands the chipkill ECC. DRAM memory suffers from three types of memory faults: hard, intermittent, and soft faults [8, 22]. Memory faults may be *active* when they manifest as memory errors, or *dormant* when they do not. For memory systems that consist of large numbers of DRAM modules, fields studies have shown that, under typical system conditions, hard and intermittent errors appear as frequent as or even more frequent than soft errors [4, 19]. This has promoted the adoption of chipkill schemes over the traditional SECDED protection (single error correction double error detection). Chipkill schemes reduce SDC (silent data corruptions) errors by 2× to 42× over the traditional SECDED [8, 21].

Multi-rate refresh faces issues. DRAM cells require periodical refresh operations to avoid data loss. By default, a DRAM row is refreshed at 64ms refresh interval. DRAM refresh consumes up to 50% of total DRAM power [11]. For power reduction, recent studies proposed *multi-rate refresh*, which exploits the large process variations of DRAM cells in one module to relax the refresh interval of most rows, e.g., RAIDR[11]. Those schemes profile the rows that need short refresh intervals, referred to as *SRI* set, and exclude them from runtime refresh interval relaxation. However, the SRI set is semi-dynamic — while the set size remains stable, there are cells entering to and existing from the set [16, 17]. When adopting online profiling schemes with reasonable overhead, the newly appearing cells within the profiling interval may manifest as retention errors. Therefore, a viable multi-refresh scheme has to effectively address the increase of retention errors at runtime.

Reliability emergency. [1] It seems to be natural to integrate multi-rate refresh in chipkill protected memory, which addresses both reliability and power consumption issues. Unfortunately, relaxing refresh interval could introduce magnitudes of more dynamically appearing retention errors, which dramatically degrade system reliability, e.g., there could be 3× or more SDC errors. More seriously, the systems under relaxed refresh intervals cannot guarantee the high reliability that they are designed for.

*Xianwei Zhang is now working at AMD Research, and this paper was finished during his PhD study at University of Pittsburgh.
†Rujia Wang is now working at Illinois Institute of Technology, and this paper was finished during her PhD study at University of Pittsburgh.

ASPDAC '19, January 21–24, 2019, Tokyo, Japan

ACM ISBN 978-1-4503-6007-4/19/01…$15.00
https://doi.org/10.1145/3287624.3287639

[1] Whereas this paper targets at retention induced reliability emergency, the proposed design applies to emergencies caused by temporal [4] or geological [20] factors.

Adopting stronger ECC in hardware is a simple but expensive solution. For example, to double the baseline chipkill capability, we need to add two ×4 ECC chips, which demand 25% ECC space overhead and less popular bus width configuration. Adopting 32 ×4 data chips and 4 ×4 ECC chips keeps the 12.5% ECC space overhead, but demand a 144-bit wide rank. Studies have shown that wide ranks fetch more data than necessary, which reduces energy efficiency and contradicts with the design goals of multi-rate refresh.

In this paper, we propose PlusN, a hardware assisted memory protection scheme to adaptively boost chipkill capability under reliability emergency. We summarize our contributions as follows:

- We study the *reliability emergency* and show that the dramatic increase of retention errors jeopardizes the system reliability. Since the severity of reliability emergency depends on various factors, such as refresh interval, profiling interval, it is beneficial to devise dynamically adjustable error protection schemes to effectively address it.
- We propose PlusN to adaptively address reliability emergency. Based on the error probability assessment at runtime, the system switches its memory protection from the baseline chipkill to stronger PlusN.
- We evaluate PlusN and compare it to the state-of-the-art. Our experimental results show that PlusN effectively enforces the system reliability guarantee with moderate overheads.

2 BACKGROUND

2.1 Chipkill Schemes

Chipkill schemes typically adopt RS (Reed Solomon) codes [10]. When there are $2t$ redundant symbols, a RS code can detect and correct up to t symbol errors. Stronger chipkill can be achieved by having more symbols, and hence larger storage overheads.

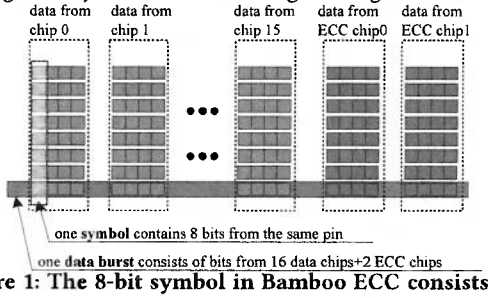

Figure 1: The 8-bit symbol in Bamboo ECC consists of bits transmitted in one pin [8].

In this paper, we adopt Bamboo ECC [8] as the baseline chipkill. As shown in Figure 1, one rank has 16 ×4 data chips and two ×4 ECC chips. A 64B memory line is communicated in eight data bursts with each chip providing four bits in each burst. The eight bits from one pin form an 8-bit symbol so that there are 72 symbols in total. This setting can detect five symbol errors and correct three, or detect four symbol errors and correct four, referred to as 3SC5SD and 4SC4SD, respectively. Based on the observation that four-pin chip failures are rare [21], 3SC5SD Bamboo chipkill tends to have better error detection when there are more retention errors. We therefore adopt 3SC5SD as the baseline in this paper. We will evaluate both in the experiment section. While our results show that 3SC5SD has better SDC than that of 4SC4SD, our design is independent of the baseline chipkill selection and can be applied to different chipkill designs.

2.2 Retention Errors

The default refresh interval of modern DRAMs is 64ms, which is set to meet the worst case acceptable memory cells. However, given a commodity DRAM DIMM, most cells can be refreshed at a larger refresh interval, e.g., T_{large}. The cells that demand less than T_{large} refresh interval, referred to as SRI (short refresh interval) set, leak their charge at faster speeds than others.

Recent studies [16, 17] have revealed that, due to variable retention time (VRT) and data pattern dependence (DPD) issues, the SRI set is semi-dynamic — while the set size remains stable, there are cells entering to and exiting from the set at runtime. Therefore, a multi-rate refresh scheme has to adopt online profiling strategy to periodically profile the SRI set $\text{SRI}_{profiled}$ at runtime.

The real runtime SRI set, i.e., $\text{SRI}_{current}$, contains two types of faults — the faults that overlap with $\text{SRI}_{profiled}$, and the faults that do not. While the former can be made dormant by refreshing their corresponding rows at 64ms, the latter shall likely manifest as retention errors. Patel *et al.* [16] developed a well fitting polynomial for the new retention errors, i.e., $\text{SRI}_{current}$ - $\text{SRI}_{profiled}$.

For computers that expect the uncorrectable bit error rate to be below 10^{-15}, using SECDED can relax the refresh interval to 1024ms or larger and the profiling interval to more than two days [16]. Such a setting eliminates most refresh operations with low profiling overhead.

3 MOTIVATION

Given that (1) chipkill can ensure high level memory reliability and (2) multi-rate refresh schemes have demonstrated significant power and energy consumption benefits, it is intuitive to apply multi-rate refresh to chipkill enhanced systems for improving both reliability and power/energy consumption. However, we need to answer the following question before the integration. **Q: is it reliable to adopt multi-rate refresh schemes on chipkill enhanced memory?** We next study the simple integration and motivate our design using Faultsim [14] analysis.

3.1 FaultSim Simulation Results

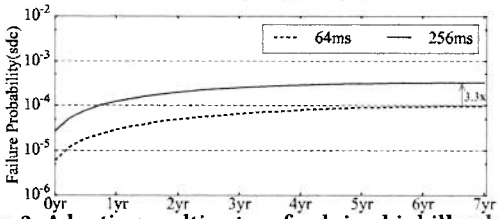

Figure 2: Adopting multi-rate refresh in chipkill enhanced memory leads to reliability emergency.

Since the retention faults that overlap with $\text{SRI}_{profiled}$ can be made dormant by refreshing them at 64ms, we focus on newly appearing retention errors in the following study.

When relaxing the refresh intervals, we expect increasing retention errors while other types of errors remain stable. We use Faultsim [14] to study the DUE (detectable but uncorrectable errors) and SDC (silent data corruption) when there is an increase of retention errors. The experimental settings are listed in Section 5. In the experiments, we refresh all rows except those identified by $\text{SRI}_{profiled}$ at 256ms. The results are summarized in Figure 2. We use 3SC5SD as the baseline. While DUE is stable (not shown in the

figure), SDC exhibits more than 3× increase due to new retention errors. Having more SDC errors is serious as neither the hardware nor the software is aware of such errors [8]. As a comparison, a DUE error may trigger OS intervention to recover from a system checkpoint. Given high end embedded systems are typically constructed with strict reliability demand, the large SDC increase is intolerable, which is referred to as **reliability emergency** in this paper. From the figure, the baseline chipkill ensures the system SDC being 9.7×10^{-5} in 7 years. It is set as the system reliability guarantee in this paper.

3.2 Enhanced Chipkill and Refresh/Profiling

We further adopt the formulas of failure probability estimation in Faultsim [14] to study the system reliability in seven years. Based on the FIT of single bit transient error, under the normal working condition, the single bit errors, for the 1Gb device in [21], appear at a low probability of 8.6×10^{-4} in seven years.

Assume we adopt 256ms refresh interval and 48-hour online profiling interval, i.e., (RI=256ms, PI=48H). We may accumulate 0.102 new errors in each profiling interval [16]. That is, the retention errors at (RI=256ms, PI=48H) exhibits 120× increase of the single bit errors at 64ms refresh interval and no profiling. For comparison purpose, if we skip the thorough profiling and let the retention error accumulate in 7 years, the retention error increase corresponds to 150,000 × FIT increase of a single bit error source.

With the above results and analysis, we conclude that it is not safe to adopt a multi-rate refresh scheme with reasonable online profiling overhead, e.g., (RI=256ms, PI=48H). Given the correlation among retention bit error rate, refresh interval, and profile interval, Patel *et al.* [16] proposed to pause the whole system to profile the retention weak cells. It tends to be intrusive and introduces large performance degradation when profiling frequently. In this paper, we mitigate this constraint by proposing an alternative profiling using one extra DIMM $DIMM_{extra}$ in the system. Other DIMMs are profiled alternatively at runtime. To profile $DIMM_d$, we first move all its data to $DIMM_{extra}$, and redirect all accesses to $DIMM_{extra}$ as well. We then profile $DIMM_d$ without pausing the system. After profiling, the data are copied back such that $DIMM_{extra}$ may be used to assist profiling other DIMMs. While a thorough evaluation of the new profiling strategy is beyond this paper, we argue that a computing system may profile at a shorter interval while being less intrusive to the system performance. This broadens the choices of multi-rate refresh settings.

We test a large number of (RI, PI) combinations and identify two multi-rate refresh configurations that are safe to use: 1) 4SC6SD protection with (RI=256ms, PI=48H); 2) 5SC7SD protection with (RI=512ms, PI=8H). Due to space limit, we skip the combinations that are not safe.

4 THE DESIGN DETAILS

4.1 An Overview

The PlusN scheme, as a memory protection design to address retention error induced reliability emergency, activates stronger protection when the system is about to adopt multi-rate refresh optimization, and restores to the baseline chipkill protection when rows are refreshed at the default 64ms.

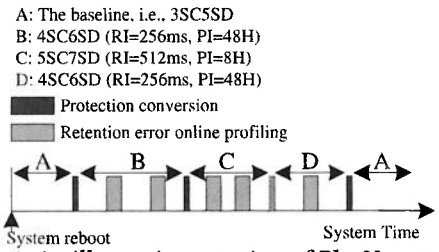

Figure 3: An illustrative overview of PlusN protection.

Figure 3 presents an illustrative overview of the scheme. when the system reboots, it adopts the baseline chipkill, i.e., the 3SC5SD mode. If the system decides to adopt a multi-rate refresh scheme, based on the refresh interval and profiling interval setting, it chooses a stronger protection mode, e.g., 4SC6SD in the figure, and converts the memory protection before entering the new mode. The protection conversion is an operation in which the memory controller reads all rows, generates the new ECC symbols, and writes them back to the memory system. We discuss the locations of data and ECC symbols for different schemes in next section.

4.2 The PlusN Protection

Figure 4 illustrates how PlusN works. Since one physical memory rank consists of 16 ×4 data chips and two ×4 ECC chips, one memory access is serviced with 72 8-bit symbols. For the baseline, the data bits of one 64B logic memory line are partitioned to 64 data symbols based on the pins used in transmission. The data symbols and ECC symbols are transmitted in parallel.

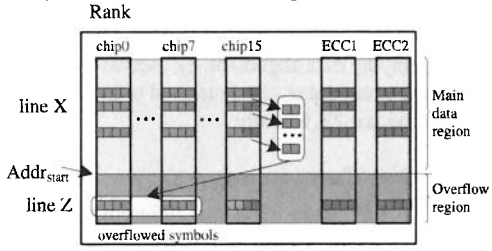

Figure 4: PlusN adopts 4SC6SD and moves 2 data symbols from 16 consecutive rows in main data region to one line in the overflow region.

To detect T+1 and correct T-1 ($T > 4$) error symbols for one memory access, we need 2T ECC symbols. PlusN chooses to save all ECC symbols together such that, without increasing the physical chip count and/or bus width, we can save and transmit up to (72-2T) out of 64 data symbols of one logic line. Therefore, PlusN partitions the physical memory space to two regions — the *main data region* and the *overflow region*. The majority, i.e., (72-2T) out of 64 data symbols, are saved in the data region while the rest, i.e., (2T-8) data symbols, are saved in the overflow region. In Figure 4, we are to protect memory line X with 4SC6SD, we move two data symbols (that originally transmitted using the last two pins of chip#15) to the overflow region and save the two extra ECC symbols using the freed space in chip#15.

4.2.1 Address Mapping. Since the overflow region is also refreshed using relaxed refresh intervals, its memory lines need to save 2T ECC symbols as well. For 4SC6SD, one overflow memory line can save 62 data symbols, i.e., it can hold the overflowed data symbols from up to 31 data lines. In PlusN, mapping address X in data region to its overflow line address Z in overflow region is accomplished by

the hardware. To achieve high performance without introducing large hardware overhead, PlusN saves the overflowed data symbols from up to 16 consecutive data lines. The address mapping function is

$$Addr_Z = Addr_{start} + Addr_X >> 4 \quad (1)$$

where $Addr_{start}$ is the starting address of the overflow region. Note that the protection conversion is for physical address space, which is independent of the OS. The addresses $Addr_X$ and $Addr_Z$ are all physical addresses.

When adopting 5SC7SD, PlusN needs to relocate four data symbols from each memory line in the data region to reserve the space for extra ECC symbols. One overflow line thus can service up to 15 data lines. To simplify the address mapping, PlusN saves the overflowed data symbols from up to 8 consecutive data lines.

The overflow region accounts for 6.25% and 12.5% of the data region for 4SC6SD and 5SC7SD, respectively.

4.2.2 Supporting PlusN with Sector Cache. After moving data symbols to the *overflow* region, one memory access to the main data region can only fetch a partial cache line. An additional fetch is necessary when the missing data is requested. In this paper, we adopt the sector cache design to support transparent access to the data stored in overflow lines.

PlusN attaches a two-bit flag $f_1 f_2$ to each cache line in the cache – f_2 indicates if the last 8B of the cache line are available in the cache while f_1 indicates if other bits are ready. Assume 4SC6SD is adopted, a partial cache line may miss the last two data symbols, i.e., 2B data. To simplify data alignment, e.g., accessing an 8B double precision floating point value, the f_2 is used to indicate if the last 8B, rather than the last 2B, is ready.

4.3 Protection Conversion

Since 3SC5SD, 4SC6SD, and 5SC7SD use different number of ECC symbols, PlusN needs to recompute the ECC for all memory lines before switching to a new mode. It consists of the following steps:

- PlusN first allocates and adjusts the overflow region based on the current and new protection modes. For example, for 4SC6SD and 5SC7SD, the overflow region needs to be be 6.25% and 12.5%, respectively, of the main data region. When upgrading the protection from the baseline to 4SC6SD, the hardware marks the region in the high memory address space as the overflow region, and swaps the original data in this region to the SSD.
- PlusN then walks through the main data region and converts the lines sequentially. It keeps an address register $PTR_{current}$ pointing to the current memory line being converted. It may need to fetch required data symbols from the overflow region, e.g., when converting from 4SC6SD to baseline. When having data symbols ready, PlusN generates new ECC symbols and writes the data together with the ECC symbols back to the memory.
- The overflow region can only be legally accessed by *hit-not-ready* cache accesses. PlusN records its start address $Addr_{start}$ in a register. If an application accesses the swapped data, i.e., it generates an access to the physical address in the overflow region, e.g., $Addr_1 > Addr_{start}$, the memory controller intercepts the request and notifies the OS to trigger I/O access.

4.4 Hardware Overheads

Cache modification. To enable fetching and saving partial cachelines, PlusN attaches a two-bit flag to each cacheline, introducing 0.4% space overhead. The flag bit is checked in parallel with that of the valid bit in the cache, which is faster than tag comparison and data array access. Therefore, both space and timing overhead are negligible.

Protection conversion. PlusN needs protection conversion before adopting a different protection, which involves two types of overhead – allocating overflow region and generating new ECC code. When performing upwards conversion, e.g., from the baseline to 5SC7SD, PlusN swaps the memory data to SSD. 5SC7SD needs 128MB overflow region for each 1GB memory space. For a typical SSD setting, e.g., Samsung 850 Pro [3], that has more than 400MB sequential writing bandwidth [5], it takes about 320ms to write 128MB, or 5.1s for a 16GB memory system.

ECC overhead. PlusN needs to generate new ECC code for all rows in the memory. However, encoding and encoding can still be finished in one cycle [7, 8, 10]. Therefore, the conversion overhead comes mainly from memory accesses. The memory controller reaches close to peak bandwidth to read and write sequential data, which finishes the conversion of 16GB memory in 2.5s.

The area overhead of RS codec is $m^2 \cdot k \cdot r$ [13], where m is the symbol width in bits and k and r are the number of data and redundant symbols per codeword. The area overhead is comparable to commercial AMD chipkill [2, 8]. Pin-based ECC may decode a message only after receiving all data bits while traditional per data burst based ECC decodes after each burst. To compensate the latency penalty, we charged four extra cycles on read and one extra cycle on write, likewise to [8].

5 EXPERIMENT METHODOLOGY

5.1 System Failure Rate

To evaluate the effectiveness of the proposed scheme, we used FaultSim [14], an event-based DRAM fault simulator, to study the failure rates in different ECC designs. Similar as that in [8], we extended FaultSim to support pin symbol error detection and correction, i.e., working at cache block instead of data burst granularity.

We performed Monte-Carlo simulation of 1000M runs, and injected faults on basis of the in-field study results in [16, 21]. For the VRT errors in this paper, we modeled them as an extra single-bit fault source in the simulation. The faults are inserted to simulate a time period of 364 weeks (approximately 7 years). Scrubbing is performed every four hours. We reported the probability of DUE and SDC to measure the failure probability of the system.

5.2 Performance and Power

We used a cycle accurate simulator, Ramulator [9], to compare performance and energy consumption of the designs. A 4-core chip multiprocessor is simulated with 64KB private L1, 4MB unified L2, and 8GB DRAM. Table 1 describes the system configuration, where each rank is composed of 16 ×4 data devices and two extra ×4 ECC devices. A wide rank can be formed by combining every two ranks in two channels to double the data bus width and data access granularity. We estimated DRAM energy consumption based on Micron power calculator with Samsung DDR3-1600 parameters

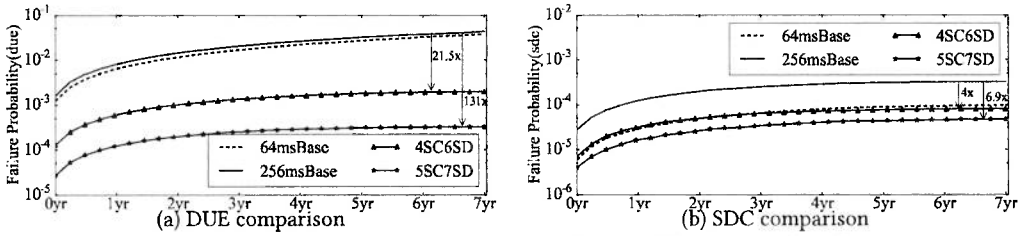

Figure 5: The system reliability when adopting (RI=256ms, PI=48H).

[18]. The runs take a suite of memory intensive benchmarks from SPEC CPU2006, PARSEC, Olden and micro benchmarks.

Table 1: System Configuration

Processor	4 cores, 3.2Ghz, 3-wide issue, 128-entry inst window
L1 cache	private, 64KB, 4-way, 64B line, write-back, 3 cycle latency
LLC	shared, 4MB, 8-way, 64B line, write-back, 12 cycle latency
Memory controller	128-entry buffer, open page, FR-FCFS scheduling address mapping: rw:bk:rk:cl:ch:offset
DRAM	2 channels, 2-rank, 64-bit data bus (6.4GB/s) 2Gb x8 DDR3-1600K (11-11-11) [18]
Bench-marks	SPEC(401.bzip, 429.mcf, 450.sop, 458.sje, 471.omn, 484.xal) PARSEC(canneal, ferret), Olden(em3d, mst), Micro(gups)

5.3 Evaluated Schemes

We evaluated the following schemes for comparison.

- `Baseline`: this is the scheme that adopts Bamboo chipkill. It can detect five symbol errors and correct up to three, i.e., 3SC5SD.
- `4SC6SD` and `5SC7SD`: these are two variants of PlusN, with two and four more redundant symbols, respectively.
- `WideChannel`: this is the scheme that combine two ranks to work in lockstep. It consists of 32 data chips and 4 ECC symbols. It can detect up to 9 symbol errors and correct up to 7 symbol errors.
- `4SC4SD`: the baseline Bamboo can also be configured to detect and correct four symbol errors.

6 RESULTS

6.1 System Reliability

We first studied the system reliability under different chipkill protection schemes. Figure 5 summarizes the DUE and SDC errors in seven years. `64msBase` is the baseline chipkill with 64ms refresh interval; others are different chipkill schemes with (RI=256ms, PI=48H). From the figure, relaxing refresh interval leads to large reliability degradation, due to significantly more retention errors. 4SC6SD achieves 21.5× and 4× DUE and SDC improvements over `256msBase`, respectively. That is, adopting 4SC6SD can ensure both DUE and SDC are below those of `64msBase`. With more protection symbols, 5SC7SD achieves much lower failure probabilities.

6.2 Performance Impact

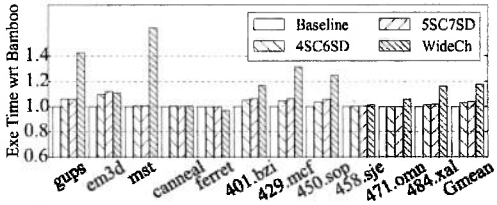

Figure 6: Comparing the performance of different chipkill schemes (Lower is better).

We then compared the performance of different chipkill schemes. Figure 6 reports the program execution time only,. The results are normalized to 3SC5SD. From the figure, the workloads with more memory accesses, e.g., *gups* and *em3d*, exhibit larger performance degradation. This is because there are more accesses falling in the overflowed region, which introduce additional memory access overhead. For the workloads with low memory access intensity, e.g., 458.*sjeng*, the degradation is negligible. On average, `4SC6SD` and `5SC7SD` are 2.7% and 3.1% worse than the baseline, respectively. The degradation is small because overflow lines are brought to L2 such that the extra accesses to the main memory are few. The difference between `4SC6SD` and `5SC7SD` is small because both trigger extra accesses if accessing the last 8B of an incomplete cacheline.

`WideChannel` only benefits few workloads, e.g., *ferret*, that have super high locality. For others, `WideChannel` brings in more data than necessary – a big portion of them are evicted before usage. With higher memory bandwidth demand and low parallelism, the performance degradation in `WideChannel` is 8% on average.

6.3 Energy Consumption

Figure 7 reports the system energy consumption, including DRAM, cache and processor. The refresh energy saved from relaxed refresh interval relaxation is not included either. On average, `PlusN` introduces 4%-6% extra energy consumption, which comes from more memory accesses (to the overflowed lines) and longer program execution. As a comparison, `WideChannel` consumes more than 30% energy. `WideChannel` doubles the energy consumption per memory access but fails to halve the number of total memory accesses. To summarize, `PlusN` reaches a good tradeoff among reliability, performance, and energy consumption.

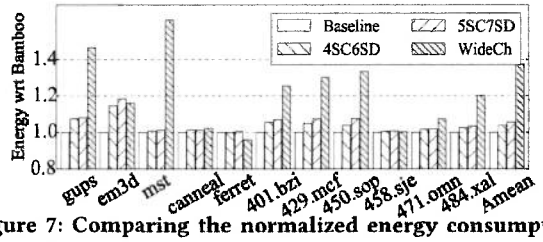

Figure 7: Comparing the normalized energy consumption under different schemes.

We further consider the performance and energy overheads involved in protection conversion (with settings in Section 4.3). The performance overhead is trivial when each protection mode lasts for a sufficient long duration, e.g., 48H. And, the extra energy consumption is negligible comparing to the saved refresh energy.

6.4 Additional Studies

6.4.1 Cache misses. We compared the cache hit rates under different schemes. `PlusN` variants have worse cache hit rates due to

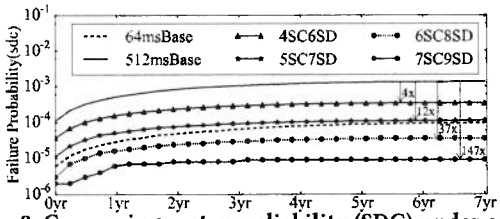

Figure 8: Comparing system reliability (SDC) under severer reliability emergency (RI=512ms, PI=8H).

hit-not-ready accesses. However, most *hit-not-ready* accesses are satisfied by accessing the overflow lines buffered in L2, which have low access latency. The overall performance impacts are small.

6.4.2 Severer Reliability Emergency. Larger refresh intervals, e.g., 512ms, and/or larger profiling intervals tend to introduce more newly appearing retention errors. As shown in Figure 8, When adopting (RI=512ms, PI=8H), without enhancing chipkill capability, 3SC5SD exhibits 12× SDC error degradation. While adopting stronger ECC helps to reduce the SDC errors, 4SC6SD is not strong enough. 5SC7SD barely matches the SDC of the setting (RI=64ms, PI=∞). A more stronger ECC code would be necessary if higher system reliability is needed.

6.4.3 Different Baseline. We then re-evaluated the system reliability by choosing a different baseline chipkill scheme of 4SC4SD. Results show that adopting multi-rate refresh (RI=256ms, PI=48H) leads to 6× SDC degradation. 5SC5SD helps to reduce the SDC errors close to the baseline. When adopting (RI=512ms, PI=8H), we need to enhance the chipkill to detect and correct two more symbols, i.e., choosing 6SC6SD.

6.4.4 Applying PlusN without Multi-rate Refresh. By comparing against the 3SC5SD baseline in Figure 5, we observed that 4SC4SD has about 3.8 × SDC degradation and 100× DUE improvement over 3SC5SD. The large DUE improvement is due to better correction capability in 4SC4SD. As such, we could adopt PlusN for improving memory reliability even when we do not adopt multi-rate refresh. That is, to achieve the low SDC of 3SC5SD and the low DUE of 4SC4SD, it is possible to enhance the chipkill with one or two more symbol error detection and correction, to solve further scaling errors [28], high error rate NVM [25, 26], or other emergencies [4].

7 RELATED WORK

Memory errors and VRT. Multi-rate refresh designs, e.g., [11] run profiling to identify less-leaky rows and then skip refreshing them. Due to VRT, online profiling techniques are necessary for ensuring high level reliability. AVATAR [17] adopts SECDED enhanced scrubbing to detect retention errors at runtime. REAPER [16] adopts SECDED to reduce profiling frequency. Khan *et al.* [27] analyzed the efficacy of common ECCs. These designs target at desktop computers with lower reliability demands than that of computing servers.

ECC code and design. Recent chipkill designs focus on reducing storage requirements and applying to ×8 or wider chips [7, 8, 23, 24]. V-ECC [24] and LOT-ECC [23] are representative designs that split error detection and correction. Multi-ECC [7] and Bamboo [8] re-organize codes to provide chipkill correction with reduced storage overhead. Differing from those designs with the implicit assumption of constantly more errors, our design is targeting at reliability emergencies.

Memory fault rates are expected to rise with technology scaling [4, 22]. The large memory capacity in exascale systems demands stronger chipkill based ECC protection [4, 8, 20]. While SDDC+1 and DDDC+1 [6] share similarity with PlusN, their design details are unavailable, and they only provide one additional bit correction, which is less flexible than PlusN.

8 CONCLUSION

In this paper, we addressed an important memory design problem for high end embedded systems, i.e., whether it is reliable to adopt multi-rate refresh together with chipkill. With the naive integration, the memory system could be suffering from reliability emergency. Our proposed PlusN can effectively enforce the system reliability guarantee under different reliability emergency scenarios.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their constructive feedbacks. This work is supported in part by NSF under grants CCF-1422331, CCF-1535755 and CCF-1617071.

REFERENCES

[1] AMD OPTERON 4000 SERIES EMBEDDED PLATFORM: Delivering Performance and Scalability at the Right Power with Overall Customer Value. Product brief.
[2] AMD, "BIOS and Kernel Developer's Guide for AMD NPT Family 0Fh Processors." *Technical report*, AMD Inc., 2009.
[3] Samsung 950 Pro SSD, www.samsung.com.
[4] L. Bautista-Gomez, F. Zyulkyarov, *et al*, "Unprotected Computing: A Large-scale Study Of DRAM Raw Error Rate On A Supercomputer," *SC*, 2016.
[5] C. M. Compagnoni, A. Goda, *et al*, "Reviewing the Evolution of the nand Flash Technology," *Proc. of The IEEE*, 2017.
[6] HP, "How Memory RAS Technologies can Enhance the Uptime of HP ProLiant Servers," , 2013.
[7] X. Jian, H. Duwe, *et al*, "Low-power, Low-storage-overhead Chipkill Correct Via Multi-line Error Correction," *SC*, 2013.
[8] J. Kim, M. Sullivan, *et al*, "Bamboo ECC: Strong, Safe, And Flexible Codes For Reliable Computer Memory," *HPCA*, 2015.
[9] Y. Kim *et al*, "Ramulator: A Fast and Extensible DRAM Simulator," *CAL*, 2005.
[10] R. S. Lim, "A Decoding Procedure for the Reed-Solomon Codes", 1978.
[11] J. Liu, *et al*, "RAIDR: Retention-aware Intelligent DRAM Refresh," *ISCA*, 2012.
[12] X. Zhang and Y. Zhang, *et al*, "Exploiting DRAM Restore Time Variations in Deep Sub-micron Scaling," *DATE*, 2015.
[13] S. Morioka and Y. Katayama, "Design Methodology for a One-Shot Reed-Solomon Encoder and Decoder," *ICCD*, 1999.
[14] P. J. Nair, D. A. Roberts, *et al*, "FaultSim: A Fast, Configurable Memory-Reliability Simulator for Conventional and 3D-Stacked Systems," *TACO*, 2016.
[15] P. J. Nair, V. Sridharan, *et al*, "XED: Exposing On-Die Error Detection Information For Strong Memory Reliability," *ISCA*, 2016.
[16] M. Patel, J. S. Kim, *et al*, "The Reach Profiler (REAPER): Enabling The Mitigation Of DRAM Retention Failures Via Profiling At Aggressive Conditions," *ISCA*, 2017.
[17] M. K. Qureshi, D.-H. Kim, *et al*, "AVATAR: A Variable-Retention-Time (VRT) Aware Refresh For DRAM Systems," *DSN*, 2015.
[18] Samsung, "DDR3 SDRAM memory," *Technical report*, 2012.
[19] C. Slayman, "Soft Error Trends and Mitigation Techniques in Memory Devices," *RAMS*, 2011.
[20] V. Sridharan, N. DeBardeleben, *et al*, "Memory Errors In Modern Systems: The Good, The Bad, And The Ugly," *ASPLOS*, 2015.
[21] V. Sridharan and D. Liberty, "A Study Of DRAM Failures In The Field," *SC*, 2012.
[22] V. Sridharan, J. Stearley, *et al*, "Feng Shui Of Supercomputer Memory: Positional Effects In DRAM And SRAM Faults," *SC*, 2013.
[23] A. N. Udipi, N. Muralimanohar, *et al*, "LOT-ECC: Localized And Tiered Reliability Mechanisms For Commodity Memory Systems," *ISCA*, 2012.
[24] D. H. Yoon and M. Erez, "Virtualized And Flexible ECC For Main Memory," *ASPLOS*, 2010.
[25] R. Wang and Y. Zhang, *et al*, "ReadDuo: Constructing reliable MLC phase change memory through fast and robust readout," *DSN*, 2016.
[26] R. Wang and L. Jiang, *et al*, "SD-PCM: Constructing reliable super dense phase change memory under write disturbance," *ASPLOS*, 2015.
[27] S. Khan and D. Lee, *et al*, "The Efficacy of Error Mitigation Techniques for DRAM Retention Failures: A Comparative Experimental Study," *SIGMETRICS*, 2014.
[28] X. Zhang and Y. Zhang, *et al*, "Restore Truncation for Performance Improvement in Future DRAM Systems," *HPCA*, 2016.