Copyright
by
Jungrae Kim
2016

The Dissertation Committee for Jungrae Kim
certifies that this is the approved version of the following dissertation:

Strong, Thorough, and Efficient Memory Protection against Existing and Emerging DRAM Errors

Committee:

Mattan Erez, Supervisor

Yale Patt

Nur Touba

Calvin Lin

Alaa Alameldeen

Strong, Thorough, and Efficient Memory Protection against Existing and Emerging DRAM Errors

by

Jungrae Kim, B.S.E., M.S.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

December 2016

To loving parents,

Namho Kim and Eunsung Lee,

for their endless love, support, and encouragement.

Acknowledgments

First of all, I thank my great advisor, Professor Mattan Erez. Mattan is my role model as an open-minded, hard-working, and brilliant researcher. He kept encouraging me to develop new research ideas with in-depth discussion and rigorous feedback. I was fortunate to have him as my advisor and enjoyed working with him over the past 5 years.

I also would like to thank the other members of my dissertation committee: Professor Yale N. Patt, Professor Nur A. Touba, Professor Calvin Lin, and Dr. Alaa Alameldeen. Their valuable comments helped me develop and elaborate the concepts in my dissertation.

I thank Seh-Woong Jeong, Jaehong Park, Young Jun Kwon, Kyung-Mook Lim, Jinpyo Park, Joonseok Kim, Nak Hee Seong, Yoonhwan Kim, Suknam Kwon, and Sangyoon Lee at Samsung Electronics. I learned various aspects of computing systems from them and am proud to have been a member of a team which changed daily lives of the mankind. I am also thankful to the company for supporting my Ph.D study.

I am also thankful to Min Kyu Jeong, Ikhwan Lee, Joonsoo Kim, and Dam Sunwoo for their friendship. From the day one in Austin, their help and support made me and my family stand here. I also should acknowledge other friends in the LPH research group: Michael Sullivan, Dong Wan Kim, Jinsuk Chung, Minsoo Rhu, Seong-Lyong Gong, Tianhao Zheng, Song Zhang, Haishan Zhu, Esha Choukse, Sangkug Lym, Yongkee Kwon, Kyushick Lee, Benjamin Cho, and Nicholas Kelly. They gave me invaluable comments and feedback to my research and papers, and I would not have completed my Ph.D. research without their help; especially Mike, who collaborated on all of my papers.

Finally, I want to thank my family for their consistent support and endless love. Very special thanks to my wife, Sun Young Hwang; she has been supportive throughout the years and made my life full of joy with three loving children; Dana Dongha Kim, Julia Dongjoo Kim, and Justin Kim.

Strong, Thorough, and Efficient Memory Protection against Existing and Emerging DRAM Errors

Publication No. ____________________

Jungrae Kim, Ph.D.
The University of Texas at Austin, 2016

Supervisor: Mattan Erez

Memory protection is necessary to ensure the correctness of data in the presence of unavoidable faults. As such, large-scale systems typically employ *Error Correcting Codes (ECC)* to trade off redundant storage and bandwidth for increased reliability. *Single Device Data Correction (SDDC)* ECC mechanisms are required to meet the reliability demands of servers and large-scale systems by tolerating even severe faults that disable an entire memory chip.

In the future, however, stronger memory protection will be required due to increasing levels of system integration, shrinking process technology, and growing transfer rates. The energy-efficiency of memory protection is also important as DRAM already consumes a significant fraction of system energy budget. This dissertation develops a novel set of ECC schemes to provide strong, safe, flexible, and thorough protection against existing and emerging types of DRAM errors. This research also reduces energy consumption of such protection while only marginally impacting performance.

First, this dissertation develops Bamboo ECC, a technique with stronger-than-SDDC correction and very safe detection capabilities ($\geq$ 99.999994% of data errors with any severity are detected). *Bamboo ECC* changes ECC layout based on frequent DRAM error patterns, and can correct concurrent errors from multiple devices and all but eliminates the risk of silent data corruption. Also, Bamboo ECC provides flexible configurations to enable more adaptive graceful downgrade schemes in which the system continues to operate correctly after even severe chip faults, albeit at a reduced capacity to protect against future faults. These strength, safety, and flexibility advantages translate to a significantly more reliable memory sub-system for future exascale computing.

Then, this dissertation focuses on emerging error types from scaling process technology and increasing data bandwidth. As DRAM process technology scales down to below 10nm, DRAM cells are becoming more vulnerable to errors from an imperfect manufacturing process. At the same time, DRAM signal transfers are getting more susceptible to timing and electrical noises as DRAM interfaces keep increasing signal transfer rates and decreasing I/O voltage levels. With individual DRAM chips getting more vulnerable to errors, industry and academia have proposed mechanisms to tolerate these emerging types of errors; yet they are inefficient because they rely on multiple levels of redundancy in the case of cell errors and ad-hoc schemes with suboptimal protection coverage for transmission errors. *Active Guardband ECC* and *All-Inclusive ECC* make systematic use of ECC and existing mechanisms to provide thorough end-to-end protection without requiring redundancy beyond

what is common today.

Finally, this dissertation targets the energy efficiency of memory protection. *Frugal ECC* combines ECC with fine-grained compression to provide versatile and energy-efficient protection. Frugal ECC compresses main memory at cache-block granularity, using any left over space to store ECC information. Frugal ECC allows more energy-efficient memory configurations while maintaining SDDC protection. Its tailored compression scheme minimizes insufficiently compressed blocks and results in acceptable performance overhead.

The strong, thorough, and efficient protection described by this dissertation may allow for more aggressive design of future computing systems with larger integration, finer process technology, higher transfer rates, and better energy efficiency.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

DRAM errors are not rare. Field measurements show that a DRAM *Dual Inline Memory Module (DIMM)* suffers from an average of $0.65 - 330$ errors per month [122, 48, 128, 129, 93, 127]. In large-scale systems with tens of thousands of DIMMs, these numbers translate into significant system-wide error rates. On the Jaguar supercomputer with 18K nodes and 73K DDR2 DIMMs, there are 250,000 DRAM errors per month on average, or there is a DRAM error once every 10 seconds [128]. With such frequent errors, the output of a large-scale computation is unreliable and the reliability of main memory has become a significant and growing concern of system designers and operators.

Large-scale systems, datacenters, and business-critical servers currently employ *Error Correcting Codes (ECC)*, which trade off redundant storage and bandwidth for improved reliability. The most widely known ECC is *Single Error Correcting — Double Error Detecting (SEC-DED)*, which can correct all 1-bit errors and detect all 2-bit errors. On 64-bit data, 8-bit redundancy is added for SEC-DED. The standard ECC DIMM has 8 redundant pins with 64 data pins, resulting in 12.5% redundant storage and transfer.

Highly reliable systems require stronger levels of protection. *Single Device Data Correction (SDDC)* ECC[1] can restore data of a dead chip and can tolerate more severe errors than SEC-DED. Field analyses report that SDDC significantly improves memory reliability by correcting 99.94% of all errors, compared to 91% with SEC-DED [50], and achieves a 42× better uncorrected DRAM error rate than SEC-DED [128]. Recent SDDC implementations use the same ECC DIMMs originally designed for SEC-DED [9].

While SDDC protection brings needed reliability improvements to current large-scale systems, future systems will require even stronger error correction and detection capabilities due to three main reasons: growing system sizes, less-reliable process technology, and increasing transmission error rates. Also, energy-efficiency of protection is important because ECC costs extra chips that consume a significant portion of a system's energy budget. Furthermore, ECC often necessitates energy-inefficient configurations. The following sections briefly describe these DRAM reliability challenges, current approaches that address those challenges, limitations of these current approaches, and the contributions of this dissertation.

1.1 Errors with Growing System Sizes

The number of DRAM devices per system is rapidly growing to support exploding data volumes and computation throughput. The U.S. Department of

[1] SDDC is referred to as Chipkill, extended ECC, and ChipSpare protection by IBM, Sun (now Oracle), and HP, respectively [9, 53, 108, 42].

Energy expects exascale supercomputers, which are capable of a billion billion calculations per second, to be available by 2024 [94]. Such supercomputers may employ millions of DRAM DIMMs for more than 128 peta bytes of main memory [124]; this number of DRAM devices is 32× greater than the top supercomputer in 2016 [139].

1.1.1 Limitations of Current SDDC Protection

The correction and detection strength of current SDDC protection may not be sufficient to deal with the growing system integration levels and their higher error rates. The mean time between uncorrectable errors in current systems with SDDC protection is as low as 9 days [93]. Without a stronger correction capability, future systems will suffer from degraded system-level reliability with an uncorrectable error every few hours.

The detection capability of a protection scheme is as important as its correction capability in large-scale computing. ECC has limited correction capability due to physical constraints (e.g., the amount of redundancy) and large-scale systems typically employ runtime support (e.g., checkpoint and restart [38, 27]) to correct rare ECC-uncorrectable errors. Such schemes mostly rely on ECC detection capability to trigger the state-restoration procedure so that uncorrectable errors, if any, should at least be detected by ECC to prevent applications from producing incorrect results. Despite the importance of detection, some SDDC implementations trade off detection capability for improved efficiency, missing up to 1.4% of double-chip-errors (Section 3.4).

1.1.2 Contribution w.r.t. Errors in Future Large-Scale Systems

This dissertation presents a novel family of efficient ECC mechanisms for future large-scale systems. Bamboo ECC reduces service interruptions from uncorrectable errors by correcting more pin and chip errors than state-of-the-art SDDC mechanisms. Bamboo ECC also provides vastly superior detection capability against errors with any severity and eliminates the risk of silent data corruption to ensure safe system operation. As permanent faults generate more serious and uncorrectable errors with accumulation, Bamboo ECC provides fine-grained pin-level flexibility to aggressively retire such faults with minimal cost, thereby extending system lifetime.

1.2 Errors from Less-Reliable Process Technology

Shrinking process technology will make individual DRAM devices more vulnerable to errors. As DRAM technology continues to scale down to below 10nm, smaller transistors, capacitors, and wire pitches make DRAM cells more vulnerable to smaller defects, retention errors, and disturbance errors, respectively [70, 85, 82]. Current row and column sparing techniques fall short of screening out all the faulty cells, so these repeated and permanent errors can increase error rates even further, and can make error patterns more severe when faults that affect the same memory region accumulate over time.

1.2.1 In-DRAM ECC Protection

In response, industry and academia have been exploring alternatives that are based on integrating ECC within DRAM chips [68, 14, 81, 85, 70, 67, 107]. *In-DRAM ECC* works by reading and processing redundant information within the DRAM chip, commonly allocating 6.25% or 12.5% additional bits for each memory read or write. By keeping this ECC processing and redundancy within each chip, the DRAM external interface remains mostly unchanged. In-DRAM ECC allows a memory chip to present itself to the processor as a reliable part even if *inherent scaling faults* exist, enabling future DRAM devices to be manufactured with high yield.

1.2.2 Limitations of In-DRAM ECC

While in-DRAM ECC mitigates the issue of inherent faults at the boundary of a single device, its use within the context of highly-reliable systems raises an issue of increased redundancy compared to current practice. Highly-reliable systems already utilize ECC across a memory rank[2] (*rank-level ECC*) to protect against operational memory faults. As opposed to inherent faults, which exist when the system is first run, operational faults occur while the system operates (as a result of particle strikes, signaling errors, aging, etc.). Rank-level ECC commonly requires 12.5% redundancy in the form of redundant chips added to each rank. Thus, with in-DRAM ECC and rank-level ECC

[2] A memory rank is a set of DRAM chips accessed simultaneously to provide the desired channel data width.

both applied, the combined storage redundancy grows to 26.6% ($1.125 \times 1.125 = 1.266$).

1.2.3 Contribution w.r.t. Scaling-Induced Errors

This dissertation provides strong and efficient protection against both inherent and operational faults. *Active Guardband ECC (AG-ECC)* utilizes a novel and unique combination of ECC decoding and memory remapping that simultaneously achieves high efficiency, high reliability, and high yield. AG-ECC uses a strong rank-level ECC to tolerate both errors during operation and errors from imperfect process technology, eliminating the need for separate in-DRAM ECC protection against faulty cells. It not only restores the total amount of redundancy back to 12.5%, compared to up to 26.6% overhead in the proposed in-DRAM + rank-level ECC approaches, but also provides higher yield than in-DRAM ECC. AG-ECC can maintain reliability stronger than SDDC mechanisms available in current systems and significantly improves DRAM energy efficiency.

1.3 Errors from Less-Reliable Transfers

DRAM transfers are getting more vulnerable to *transmission errors*. DRAM has consistently evolved to provide more data with less energy. Each recent generation of DRAM has more-than-doubled the data transfer rate of its predecessor (Figure 1.1), and has also decreased core and I/O voltages for better energy efficiency (Figure 1.2). Increasing signal transfer rates and

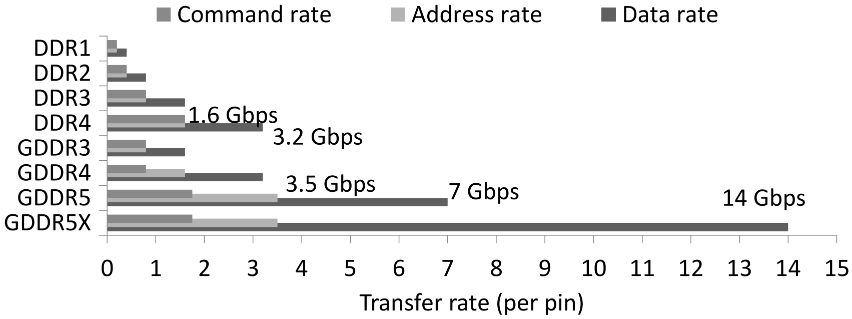


Figure 1.1: The increasing DRAM transfer rates of command, address, and data signals over the past 15 years.

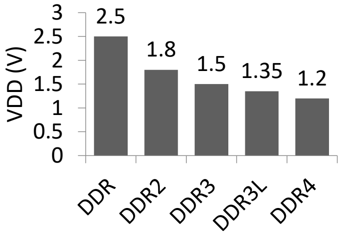


Figure 1.2: The decreasing DRAM supply voltage levels.

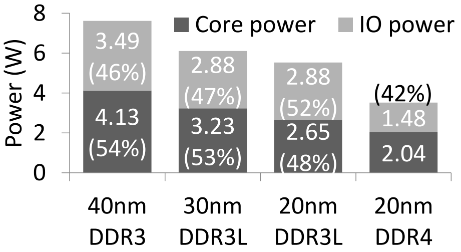


Figure 1.3: DRAM power breakdown over core and I/O.

lowering I/O voltages each exacerbate the problem of *transmission errors*. A transmission error occurs when a signal is incorrectly transferred to or from memory; higher transfer rates and lower voltages increase the vulnerability to timing and electrical noise, respectively.

While current protection techniques primarily focus on protecting data transmissions, *Clock, Control, Command, and Address (CCCA)* signals should not be neglected. These signals typically operate at half the transfer rate of data signals, yet they may suffer from more transmission errors due to DIMM topology. On a DDR4 DIMM, for example, each CCCA signal visits up to

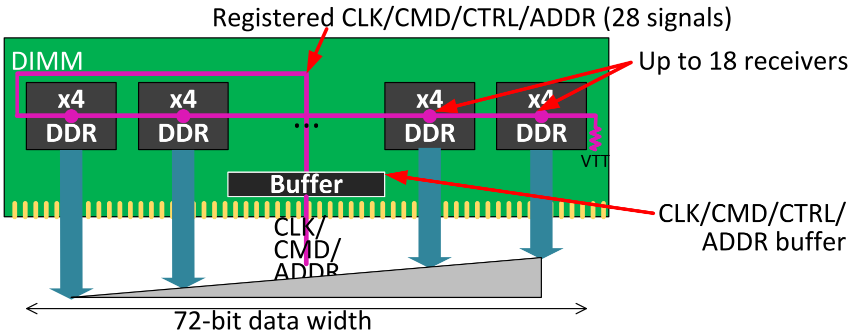


Figure 1.4: The standard fly-by topology of DDR4 registered DIMM.

18 DRAM chips sequentially and 27 CCCA signals are captured using a clock signal. On the other hand, each data signal has a dedicated connection and there is a strobe signal per up to 8 data signals. The larger number of receivers and the wider interface deteriorate CCCA signal quality by increasing reflections and signal skew, respectively, making CCCA transfers potentially more vulnerable to errors than data transfers despite the halved transfer rate. Signal and power integrity simulations on DDR4 registered-DIMM show that CCCA signals have a narrower time window to reliably receive signals than data signals. Also, evidence from a large-scale field study [127] strongly suggests that CCCA errors are already a severe problem for reliability-conscious systems, even at the modest 1.6 Gbps CCCA rate of DDR3.

1.3.1 Current Transmission Protection

Traditionally, DRAM designs have managed the transmission error rate using sophisticated circuit techniques to improve signal quality (e.g., delay-locked loops, phase-locked loops, on-die termination, differential signaling, and fly-by topology [56, 59, 60, 61, 66]) at the cost of consuming extra power. The price paid for reliable transmission is significant because roughly half of DRAM power is spent on I/O (Figure 1.3) [120]. Increasing transfer rates and tight power constraints have made circuit techniques alone insufficient to provide high levels of reliability and efficiency.

As a result, DRAM vendors have introduced ad-hoc architectural techniques in recent memory generations to protect transfers, such as *Cyclic Redundancy Check (CRC)* and *Command/Address Parity (CAP)* [60, 61, 66]. The 8-bit CRC on write data verifies the consistency of received data, while an even-parity on command/address signals verifies the consistency of received command/address. Once errors are detected by such schemes, they can be corrected by retransmitting the signals.

1.3.2 Limitations of Current Transmission Protection

CAP provides limited detection capability of CCCA errors; it can detect only odd bit errors on command/address signals. Errors on clock and control signals (e.g., chip select and clock enable) and even-bit errors on command/address signals are not detected by CAP. Such errors can have disastrous impact that is not correctable by conventional data-only ECC, leaving a serious hole in system reliability.

Due to the lack of strong protection and the severity of the problem, recent DRAM generations passively limit CCCA rates for high reliability. DDR4 introduced *gear-down* mode which transfers CCCA signals at a quarter the rate of data for reliability [60] and GDDR5X could not scale up CCCA rates along with data rates [66] (Figure 1.1). These designs trade off performance for CCCA reliability by lowering command bandwidth, adding command latency, and increasing access granularity—all of which degrade performance and efficiency.

1.3.3 Contribution w.r.t. Transmission Errors

This dissertation presents *All-Inclusive ECC (AI-ECC)*, a holistic memory error protection scheme that is able to safeguard DRAM data and CCCA signals against storage and transmission errors. Strong protection on any one component (i.e., data) provides limited benefits to overall reliability as any unprotected component will quickly become the reliability bottleneck. Therefore, current data-oriented ECC should be redesigned to thoroughly protect both data and CCCA. By leveraging the existing strong ECC schemes for the protection of CCCA errors, AI-ECC remedies the approach of protecting only or mostly DRAM data without discarding the extensive advancements made in the area of data protection.

1.4 Protection Energy Efficiency

In addition to the heightened need for stronger protection, energy efficiency of memory protection should be improved. DRAMs already consume more than 30% of total power in modern datacenters [97], and ECC overhead takes a significant portion of it; the 12.5% extra chips on ECC DIMMs translate into $\frac{1}{9}$ of the DRAM energy consumption.

1.4.1 Limitations of Current Protection

Current SDDC ECCs increase DRAM energy consumption even further by forcing energy-inefficient memory configurations. SDDC mechanisms rely on narrower DRAM devices (e.g., 16 ×4 chips per channel) to reduce the number of affected bits from a chip failure and/or a wider channel (e.g., two ECC DIMMs in parallel) to increase redundancy amount. Narrow devices double the number of chips per access than energy efficient configurations with wider devices (e.g., 8 ×8 chips per channel) and roughly double the DRAM energy consumption per access. While a wider channel can increase the amount of redundancy per access, its larger access granularity often also results in a serious performance degradation due to overfetching and reduced memory-level parallelism. Fujitsu reports that pairing two 64b channels together in an 8 DIMM system to form four 128b data channels degrades integer application performance (SPECint2006 [130]) by 6% and memory-intensive benchmark (STREAM [43]) performance by 43%. The performance hit is even more staggering on a lower capacity node—with 4 DIMMs, the performance degradation for the integer and memory benchmarks are 28.3% and 46.3%, respectively [32].

1.4.2 Contribution w.r.t. Energy Efficiency

This dissertation improves energy efficiency of main memory protection as well as protection strength. *Frugal ECC (FECC)* is an adaptive and strong

ECC technique that relies on opportunistic compression to offer an entirely new set of tradeoffs between reliability and ECC overheads. The insight behind FECC follows recent research which observes that compression at the cache-block granularity can free up enough space for other information [89, 123]; FECC uses this free space for storing the ECC redundancy. Thus, when compression succeeds, FECC can match the performance of a conventional ECC organization that uses dedicated ECC memory devices with less, or even zero, dedicated redundancy.

While the concept is simple, two crucial innovations are introduced to make FECC truly effective. First, a new compression scheme maximizes the fraction of blocks that compress just enough for ECC rather than needlessly aiming for greater levels of compression. The second innovation is how to protect poorly compressed blocks. Not all memory blocks are sufficiently compressible and some blocks fail to yield enough spare footprint for the redundant information. To address these *compression exceptions*, FECC applies different ECC layouts and protects its layout meta-data separately to guarantee up to true SDDC protection.

This careful combination of compression and ECC layout simultaneously achieves superior reliability and lower overhead when compared to state-of-the-art ECC mechanisms. As an example, this dissertation presents the first true SDDC ECC on a narrow channel (i.e., single ECC DIMM) with wider devices (i.e., 8-bit wide DRAM chips) to reduce the number of DRAM chips per access and to improve DRAM energy efficiency.

1.5 Dissertation Organization

The remainder of this dissertation proceeds as follows: Chapter 2 lays out the conceptual foundations of this dissertation by reviewing the terminology and function of system reliability, DRAM, and ECCs. Chapter 3 develops strong ECC mechanisms against data errors. Chapter 4 introduces errors from imperfect process scaling and presents strong and efficient protection. Chapter 5 discusses transmission errors and provides thorough protection on data and CCCA signals. Chapter 6 presents mechanisms to improve energy efficiency of protection using fine-grained compression. Chapter 7 summarizes current findings and presents future research directions.

Chapter 2

Background

This chapter reviews the concepts and terminology that are fundamental and common to the mechanisms of this dissertation: reliability concepts, DRAM, and ECC. More detailed motivations of particular mechanisms will be provided at the beginning of its corresponding chapter: data errors patterns (Section 3.1), inherent faults from process scaling (Section 4.1), and transmission errors (Section 5.1).

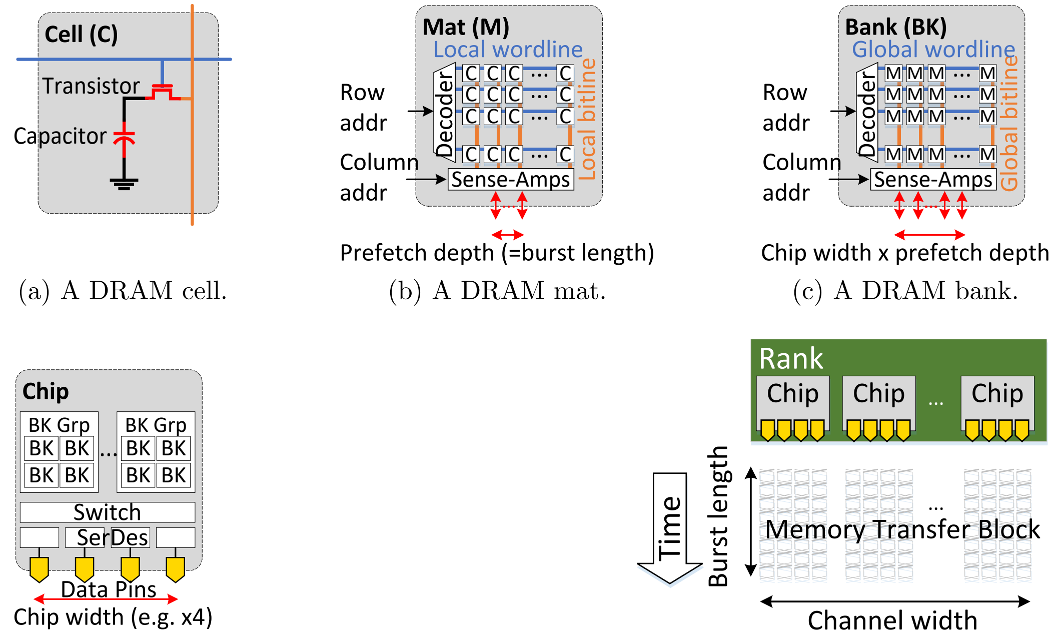
2.1 Reliability Concepts

As a matter of terminology: a *fault* is a physical phenomenon or defect that may cause an error, an *error* is a discrepancy between the intended and actual state of a system, and a *failure* is the event that a system deviates from its intended service [15]. A fault may or may not generate an error depending on circumstances, and an error can be masked or corrected by error control systems. The aim of this research is to efficiently provide such an error control system to prevent DRAM errors from developing into a system failure.

Faults can be classified based on their error manifestation and time of occurrence. A *transient* fault is introduced by some temporary environmental impetus (e.g., a high-energy particle strike), whereas a *permanent* fault is an irreversible physical defect that continually produces errors (e.g., stuck-at bit) [128]. An *inherent* fault is a permanent fault introduced during the manufacturing process, and an *operational* fault is a fault during operation (either transient or permanent).

Reliability indicates continuity of service without a failure, often measured in *Mean Time To Failure (MTTF)* or *Failures In Time (FIT, the number of failures in billion hours)*, and *manufacturability* indicates the ease with which a good product can be manufactured, measured in *yield*. A fault may affect reliability by incurring failures and an inherent fault may decrease manufacturability by making a chip defective. The aim of this dissertation is to efficiently provide fault tolerance mechanisms using *Error Correcting Codes (ECC)* to prevent DRAM faults from developing into a system failure or resulting in poor yield.

No error control system can diagnose and correct all possible errors. In general, an error control system handles an error in one of four ways: a *Detectable and Correctable Error (DCE)*, a *Detectable but Uncorrectable Error (DUE)*, a detectable but miscorrected error, or an undetectable and uncorrectable error. In the event of miscorrection or misdetection, incorrect data escapes the error control system, resulting in a possible *Silent Data Corruption (SDC)* failure. The implication of a DUE failure can vary depending on circumstances. A DUE during transmission can typically be corrected by retry, simplifying the correction of any detected errors. In the case of stor-



(a) A DRAM cell.

(b) A DRAM mat.

(c) A DRAM bank.

(d) A DRAM chip.

(e) A DRAM rank and memory transfer block.

Figure 2.1: The hierarchical organization of DRAM.

age, a DUE indicates that some data has been lost. This loss of data could be acknowledged and tolerated (by an error tolerant application), it may be corrected by some higher-level protection mechanisms (such as checkpoint and restart [38] or a hierarchical state preservation and restoration system [27]), or it may indicate a fail-stop condition where forward progress is halted (but no SDC occurs).

2.2 DRAM

Dynamic Random Access Memory (DRAM) is widely used as system main memory for its low cost and high density. The deeply hierarchical structure of DRAM is depicted in Figure 2.1. A DRAM *cell* stores each bit of data using a single transistor and a capacitor (Figure 2.1a). DRAM cells are organized into two-dimensional arrays, called *banks*, to amortize control overheads (Figure 2.1c). At a deeper look, each bank is physically sub-divided into 512×512 cell *mats* (Figure 2.1b) to reduce load capacitance and all data within neighboring mats are connected to one data pin for efficiency [142, 83, 60, 33]. A DRAM *chip* (or a device) consists of multiple banks to interleave per-bank accesses and hide long access latency (Figure 2.1d).

Most DRAMs use multiple *data pins (DQs)* to provide a parallel chip interface. A DRAM with an N-bit DQ interface is called a $\times N$ chip (e.g., a $\times 4$ or $\times 8$ DRAM); the set of DRAM chips that are accessed together in parallel to provide the desired data bus width is called a *rank* (Figure 2.1e). A *channel* is a set of ranks that time-share physical command/address and data transfer lanes. A rank/channel comprises of *Dual In-line Memory Modules (DIMMs)* that are built from multiple DRAM chips on a PCB board. A rank can consist of a single DIMM (e.g., a ×64 rank using a ×64 DIMM) or multiple DIMMs (e.g., a ×128 rank using two ×64 DIMMs in parallel), and a DIMM can contain multiple ranks (e.g., a ×64 DIMM contains two ×64 ranks).

DRAM requires three commands to access a fresh piece of data. An *activation* command (ACT) fetches a row of data into an internal *row buffer.*

Then, a *read* or *write* command (RD or WR) uses a column address to select and transfer or overwrite a particular block from the activated row. To exploit data locality and amortize command overheads, an RD or WR transfers a block of data over multiple cycles—one *beat* of data is transferred through the rank at a time. The number of beats transferred during each access is called the memory *burst length*, and most recent DRAMs use a burst length of 8. A *Memory Transfer Block (MTB)* is the unit of memory access and is defined by burst length and channel data width; typical MTB sizes from burst length of 8 and ×64/×128 channels are $64B/128B$, respectively (Figure 2.1e). Once a memory read or write completes, the DRAM bank must be restored to a ready state by issuing a *precharge* command (PRE). An access to an already-activated row (a row buffer hit) does not require an ACT or PRE command, potentially saving latency and command bandwidth for spatially local accesses. However, command bandwidth can be a limiting factor for programs lacking locality, especially in systems featuring fine-grained access granularities [10, 152, 153, 117]. Furthermore, DRAM requires periodic *refresh* commands (REF) to prevent data loss from leakage, further taxing the available command bandwidth.

2.3 Error Correcting Codes

Error Correcting Codes (ECC) have long been used to protect data against errors. ECC detects and corrects errors by adding redundant information whose value is generated algorithmically from the protected data. A data and check value pair is called an ECC *word.* A valid word whose check bits are consistent with its data is called a *codeword*, while an invalid pair due to errors is called a *non-codeword.* The process of generating a codeword from data is called *encoding* and the process of detecting errors from a word and (possibly) restoring the original data is called *decoding.*

The protection strength of an ECC can be defined by its *correction and detection coverage*: X-correcting and Y-detecting codes can correct all of up to X errors and detect all of up to Y errors. These protection coverages can be theoretically extracted from the code distance: in information theory, the *Hamming distance* between two words of equal length is the number of positions at which their symbols differ, and the *code distance* is the minimum Hamming distance between two distinct codewords. Codes with distance d, if used for detection only, can detect all errors with less-than-d erroneous symbols (i.e., $(d-1)$-detecting codes), because these errors always result in a non-codeword. Alternatively, the same codes can detect and correct all errors with less-than-$\lfloor (d{-}1)/2 \rfloor$ erroneous symbols (i.e., $\lfloor (d{-}1)/2 \rfloor$-correcting codes) by decoding a non-codeword into the nearest unique codeword.

Table 2.1 illustrates some examples of ECCs with different distances. In the figures, double-circles represent codewords; each diagram shows two neighboring codewords that are colored black and white. Black and white single-circles are non-codewords that are corrected to the codeword of the corresponding color. Codes with distance 3 can correct all single-symbol errors by finding the nearest unique codeword. However, two-symbol errors on codes

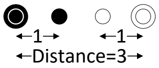
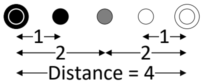
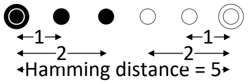
Code distance	Error handling between two neighboring codewords	Name	
		Binary symbol	Non-binary symbol
3		Single Error Correcting (SEC)	Single Symbol Correcting (SSC)
4		Single Error Correcting - Double Error Detecting (SEC-DED)	Single Symbol Correcting - Double Symbol Detecting (SSC-DSD)
5		Double Error Correcting (DEC)	Double Symbol Correcting (DSC)

Table 2.1: Examples of ECCs with different distances.

with distance 4 can make a word equally distant from two codewords (gray single-circle); in that case the error is detectable yet uncorrectable. A typical ECC scheme uses a t-symbol correcting and $(t+1)$ symbol detecting codes, which is effective in systems where errors accumulate symbol-by-symbol. Once the erroneous symbol count exceeds the correction capability, it will still be within the detection boundary.

ECCs can be categorized based on their symbol sizes. Binary ECCs use binary symbols and are most effective at protecting against random bit errors. Hamming [37] and BCH codes [44, 20] can correct a single erroneous bit in w bits of data using $\lceil log_2(w+1) \rceil$ redundant bits (and they can detect one more bit-error by adding a parity bit).

Non-binary ECCs partition data into fixed-size symbols; error detection and correction then occur at the symbol granularity. The most widely used non-binary ECCs are *Reed-Solomon (RS)* codes [116], which can achieve the minimum possible redundancy for a given distance [125]. RS codes with 8-bit symbols can correct up to t symbol errors using $2t$ redundant symbols (and they can detect one more symbol error using an additional symbol) on up to 255-symbol (2040-bit) words; many SDDC schemes [9, 41] use 8-bit symbol RS codes because of the efficiency. As most memory configurations do not fully utilize such large word sizes, however, these SDDC schemes use smaller words and regard the remaining symbols as zeros (i.e., a *shortened code*).[1] AI-ECC (Chapter 5) leverages the shortened nature of SDDC schemes to strongly protect address information without any additional redundancy.

The correction capability of RS codes can be increased using a priori knowledge about an error. Compared to a *random error* whose location and value are unknown, an *erasure* has its location known in advance (e.g., based on error history). Erasure decoding of RS codes can restore t symbol erasures using t redundant symbols, doubling the correction efficiency relative to random error decoding. AG-ECC (Chapter 4) utilizes erasure decoding to correct rare cases of overlapping inherent and severe operational faults, while either fault alone is corrected by random error decoding.

[1] Current 8-bit RS SDDC schemes use anywhere from 18 symbols [9] to 144 symbols [77].

Chapter 3

Data Errors Protection Mechanisms

This chapter presents and evaluates a novel family of efficient single-tiered ECC mechanisms, called *Bamboo ECC*, for future large-scale systems. The advantages of Bamboo ECC can be roughly characterized by three important improvements. **Strength:** Bamboo ECC has superior correction capabilities and can correct more pin and chip errors than the state-of-the-art single-tiered ECC mechanisms, while requiring the same or less redundant storage and off-chip bandwidth. **Safety:** The vastly superior detection capability of Bamboo ECC all but eliminates the risk of silent data corruption with currently observed fault modes, ensuring safe system operation. **Flexibility:** Bamboo ECC can increase redundancy at 8b granularity, compared to 8B for the state-of-the-art SDDC techniques; this fine-grained redundancy allows for more adaptive graceful downgrade schemes (i.e., memories with faulty devices can continue operation with reduced-strength codes), further improving both reliability and system lifetime. When combined into a system context, these

Parts of this chapter appear in [77]. The author is the main contributor of the idea, implementation, and evaluation.

improvements can lead to orders of magnitude fewer silent data corruptions or greatly extended system lifetime.

Three main insights lead to the innovative code design. First, many faults manifest errors on a single data pin (DQ) due to the DRAM internal structure (e.g., a mat fault) or the DRAM external interface (e.g., a faulty data pin). Secondly, aligning ECC symbols to prevalent error patterns allows for more frequent corrections. Finally, while an ECC code guarantees detection of errors up to a certain severity, careful analysis reveals that some codes provide superior detection capabilities beyond these guarantees. These insights are described in more details below.

3.1 Data Error Patterns

Recent field studies of DRAM faults [122, 87, 48, 128, 129] indicate that memory errors follow some idiosyncratic trends due to the deeply hierarchical structure of DRAM (Section 2.2). DRAM cell arrays are sub-divided into 512×512 cell *mats* and all data within neighboring mats are connected to a single DQ for efficiency [142, 83, 60, 33]. Therefore, a local fault commonly generates errors over a single DQ.

Table 3.1 presents DRAM fault rates observed in the Jaguar supercomputers with ×4 DDR2 chips [128]. It indicates that about half of all faults (49.6%) affect only a single bit. More importantly, 72.6% of all faults and 68.2% of permanent faults are confined to a single DQ. Furthermore, this analysis is likely to underestimate the true likelihood of a single DQ error within an access for methodological reasons, as intermittent single-DQ faults that are separated in time may be classified as multi-DQ faults.[1] Finally, the majority of all faults (71.0%) are permanent, indicating even more severe errors may accumulate. These findings show that the need for strong ECCs to protect against accumulating faults, especially the accumulation of single DQ faults over time. Bamboo ECC focuses on this important single-DQ fault mode to improve correction capability.

3.2 ECC Detection Coverage

This section presents theoretical background on how Bamboo ECC improves detection coverage to eliminate SDCs even in very large scale systems (e.g., exascale computing) without increasing the redundancy ratio. Section 2.3 presents theoretical correction and detection coverages of ECC. A code with distance d, if used for detection only, can detect all errors with fewer than d erroneous symbols, or detect and correct all errors with fewer than $\lfloor (d-1)/2 \rfloor$ erroneous symbols. Errors that exceed the theoretical coverage of a code can either be detected (resulting in a DUE) or they can lead to an SDC failure.

Another motivation of Bamboo ECC is that, in practice, the error de-

[1] The observation that single-DQ errors are dominant is supported by other measurements. IBM reports that 91% of erroneous beats have a single bit corrupted [50], and other results from Jaguar report that 2-bit correction provides only marginally better correction coverage than 1-bit correction; both of these results indicate that errors tend to be confined to a single DQ.

Fault Mode	Number of failing DQs				Fault Rate	
	1	2	3	4	Transient	Permanent
Single-bit	32.8 (49.6%)	N/A			14.2 (21.5%)	18.6 (28.1%)
Single-word	N/A	1.7(2.6%)			1.4 (2.1%)	0.3 (0.5%)
Single-column	6.0 (9.1%)	0.2 (0.3%)	0.1 (0.1%)	0.7 (1.1%)	1.4 (2.1%)	5.6 (8.5%)
Single-row	2.6 (4.0%)	5.6 (8.5%)	0.1 (0.2%)	0.1 (0.1%)	0.2 (0.3%)	8.2 (12.4%)
Single-bank	6.0 (9.1%)	2.5 (3.8%)	0.4 (0.6%)	1.9 (2.9%)	0.8 (1.2%)	10.0 (15.1%)
Multiple-bank	0.3 (0.5%)	0.6 (0.9%)	0.1 (0.1%)	0.8 (1.2%)	0.3 (0.5%)	1.4 (2.1%)
Multiple-rank	0.3 (0.4%)	0.3 (0.4%)	0.1 (0.1%)	3.1 (4.7%)	0.9 (1.4%)	2.8 (4.2%)
Total	48.0 (72.6%)	9.2 (13.9%)	0.7 (1.1%)	6.5 (9.9%)	19.2 (29.0%)	46.9 (71.0%)

Table 3.1: DRAM fault rates (in FIT) observed on the Jaguar supercomputer with 73K DDR2 DIMMs.

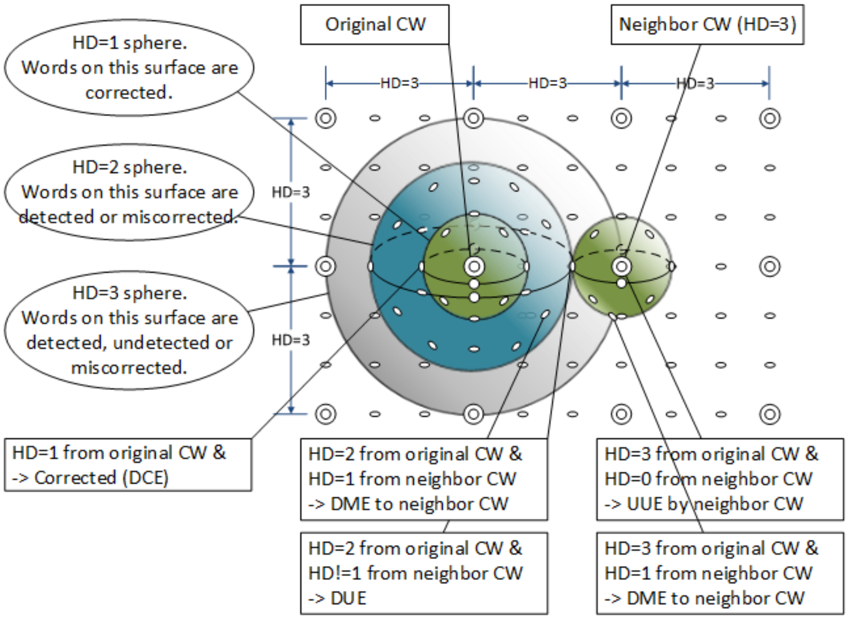

Original CW
Neighbor CW (HD=3)
HD=3
HD=3
HD=3
HD=1 sphere. Words on this surface are corrected.
HD=2 sphere. Words on this surface are detected or miscorrected.
HD=3 sphere. Words on this surface are detected, undetected or miscorrected.
HD=3
HD=3
HD=1 from original CW & -> Corrected (DCE)
HD=2 from original CW & HD=1 from neighbor CW -> DME to neighbor CW
HD=3 from original CW & HD=0 from neighbor CW -> UUE by neighbor CW
HD=2 from original CW & HD!=1 from neighbor CW -> DUE
HD=3 from original CW & HD=1 from neighbor CW -> DME to neighbor CW

Figure 3.1: Conceptual codespace of single symbol correcting codes.

tection coverage of all error codes is not equal: in general, scaling-up ECC codeword size (increasing codeword length, redundancy and correction capabilities proportionally) is associated with higher error detection coverage for severe errors. For example, two single-symbol correcting codes can be merged into a double symbol correcting one. While the latter can correct all of the errors that the former can, its larger codeword bestows stronger error detection capabilities. By maximizing its ECC word size, Bamboo ECC can safely reduce the SDC probability for severe errors down to ≤0.0000006%, without requiring more redundancy than what is currently available on a single ECC-DIMM.

An example illustrates why the practical error detection coverage of an ECC code is determined by its word size. Figure 3.1 shows a conceptual codespace for a distance 3 (or *Single Symbol Correcting (SSC)*) code. A double circle indicates a valid codeword (CW) and a single circle indicates an erroneous non-codeword. A ball of *Hamming Distance (HD) d* represents all words that are *d* symbols different from a CW. If there is a single symbol error in a CW, the erroneous word is on the HD=1 (innermost) ball and the SSC code can always restore the original data by finding the nearest CW. If there are two symbol errors, the erroneous word is on the HD=2 ball and the error will be miscorrected to a neighboring CW if and only if it also falls on the HD=1 ball of the neighbor; otherwise, the error will result in a DUE. Note that the theoretical coverage of this code is not double symbol detection as there exists this potential for miscorrection. From a practical coverage perspective, however, the ratio of the number of miscorrections to the total number of words on each ball decreases with larger ECC word sizes as the code space becomes increasingly sparse. Thus, it can be seen that the error detection of a code for errors beyond its guaranteed coverage is maximized with the ECC word length. Likewise, three symbol errors on the HD=3 (outermost) ball may be undetected (if they fall on a neighboring CW), miscorrected (if they fall on another HD=1 ball), or they are otherwise detected. Again, it can be seen that the error detection of a code for errors beyond its theoretical coverage increases with the ECC word length.

High error detection of severe errors is fundamental for Bamboo ECC to ensure safe system operation. Therefore, all of our analyses in Section 3.4.1

are performed using real ECC decoding behavior in order to capture the practical error detection coverage of the codes for errors beyond their worst-case protection guarantees. This is in contrast to other studies that characterize system failure rates using only the worst-case behavior of ECC codes [54, 118]. An important observation of Bamboo ECC is that, in practice, silent data corruption can be eliminated without resorting to expensive ECC with high worst-case error detection coverage. By manipulating the size of the Bamboo ECC codeword and by matching code layout with expected fault modes, SDC rates can be extensively reduced without introducing additional redundancy or constraints on the memory channel size.

3.3 Bamboo ECC

Bamboo ECC protects the burst of data from a DQ to provide stronger correction and detection with equal or less redundancy than previously proposed ECC mechanisms. The following subsections describe the motivation, operation, and overheads of Bamboo ECC in greater detail. Section 3.3.1 describes some useful Bamboo ECC organizations. Section 3.3.2 shows how the flexibility of Bamboo ECC can be utilized to provide superior levels of protection over the lifetime of a system. Finally, Section 3.3.3 describes the cost that Bamboo ECC schemes pay to achieve their high levels of protection and storage efficiency.

3.3.1 Organizations

Bamboo ECC changes ECC layout to have per-DQ symbols over a memory transfer block. It groups the data of a single DQ as an ECC symbol; on a conventional 8-beat memory transfer block of DDR3/4, a per-DQ symbol has 8 bits of data. By aligning ECC symbols to frequent single-DQ faults, Bamboo ECC reduces the number of symbols to be corrected in the common case and requires less redundancy to correct common errors. The 8-bit symbols are encoded by a Reed Solomon code to achieve the minimum possible redundancy for its level of correction (i.e., 2 redundant symbols per symbol correction, Section 2.3). The per-DQ symbols also increase the ECC word size to a memory transfer block. This large ECC word can detect almost all severe errors, while resulting in manageable increases in decoding complexity and latency (Section 3.3.3).

Bamboo ECC is a family of codes. With per-DQ symbols, Bamboo ECC introduces redundancy at pin granularity and provides finer control over correction and detection capabilities compared to previous ECC mechanisms with per-chip symbols. The following subsections describe some of the efficient Bamboo ECC organizations. Other organizations can be used to meet different memory system constraints and reliability requirements.

3.3.1.1 Single Pin Correcting ECC

The simplest Bamboo ECC is the *Single Pin Correcting (SPC)* organization (Figure 3.2), which can correct a bit or a pin error with 2 redundant

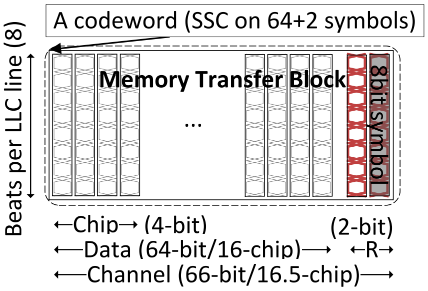


Figure 3.2: Single Pin Correcting Bamboo ECC on (64+2)-bit channel

pins. SPC requires just a quarter of the redundant storage of SEC-DED on a 64b data channel (3.1% vs. 12.5%) yet it provides a better correctable error rate.[2] One issue with SPC concerns the fixed granularity of commercial DRAM chips–while memories such as embedded DRAM (eDRAM) have the data width flexibility to support SPC, off-the-shelf memory chips are typically ×4 or ×8 DDR. Employing SPC with these chips will result in an inefficient use of pins and storage. Even on commodity DRAM chips, however, SPC can be efficiently employed as a component of a graceful degradation scheme, as described in Subsection 3.3.2.

3.3.1.2 Single Pin Correcting – Triple Pin Detecting ECC

An extra ×4 DDR chip can provide 4 redundant Bamboo ECC symbols (Figure 3.3). This redundancy can be used as either a *Double Pin Correcting (DPC)* or *Single Pin Correcting - Triple Pin Detecting (SPC-TPD)* scheme;

[2] SPC misses some SEC-DED-correctable error patterns. However, Subsection 3.4.2 shows that SPC has better uncorrectable error rates due to the rarity of these patterns and the fact that SPC has a lower raw error rate due to its lesser redundancy.

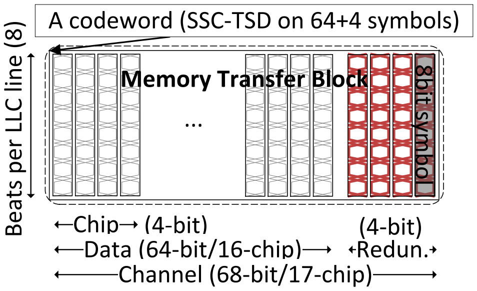


Figure 3.3: Single Pin Correcting – Triple Pin Detecting Bamboo ECC on (64+4)-bit channel

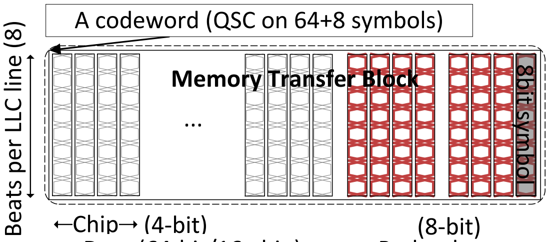


Figure 3.4: Quadruple Pin Correcting Bamboo ECC on (64+8)-bit channel

SPC-TPD usage has a very high detection coverage, detecting 100% of up-to-3-pin errors and ≥99.9996% errors beyond this point, and is thus preferred. The stronger correction capability of DPC, on the contrary, is less helpful as faults affecting exactly 2 pins are infrequent (a field measurement on 2-bit symbol correction [128] showed little improvement over SEC-DED) while it can increase the SDC probability by aggressively miscorrecting severe errors. SPC-TPD can be configured as a $(64+4)$-DQ configuration over a 64-bit data channel, halving the redundancy of SEC-DED (6.25% vs. 12.5%).

3.3.1.3 Quadruple Pin Correcting ECC

Bamboo ECC can provide stronger-than-SDDC protection with equal redundancy to state-of-the-art SDDC schemes by using a *Quadruple Pin Correcting (QPC)* organization. With two redundant ×4 chips, QPC has 8 redundant symbols (Figure 3.4) and can correct up to 4 symbol errors or any single chip-error. QPC can be configured in a (64+8)-DQ manner over a 64-bit data channel, in which case the redundant storage needs match those provided by conventional ECC DIMMs (12.5%). This storage efficiency for single-tiered, narrow-channel SDDC is only paralleled by the AMD chipkill scheme used in recent processors. QPC enjoys a stronger correction capability than AMD chipkill by correcting pin errors that are scattered over different chips. If there are two pin faults on two chips, QPC can correct both of them while AMD chipkill must report a DUE (or, in some cases, AMD chipkill results in SDC). As single-DQ faults are prevalent, this distributed error correction capability can reduce uncorrectable error rates significantly. In addition, QPC has a stronger detection capability due to its large codeword size. This leads to QPC detecting $\geq$99.9999994% of all errors. A more in-depth comparison of the two ECC schemes is evaluated in Section 3.4.

One optimization to QPC is to limit its correction capability to reduce SDC rate even further, in a manner similar to the history mechanism used by AMD chipkill [9]. AMD chipkill uses the history of corrected symbol locations within each access to reduce SDC rates. If corrected symbol locations differ among words within an access, AMD chipkill discards the correction results

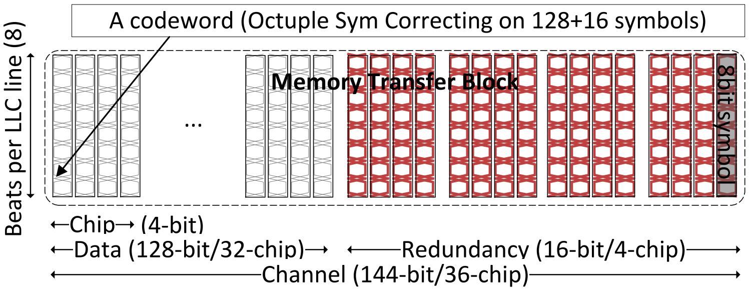

A codeword (Octuple Sym Correcting on 128+16 symbols)
Beats per LLC line (8)
Memory Transfer Block
...
8bit symbol
←Chip→ (4-bit)
←Data (128-bit/32-chip)→
←Redundancy (16-bit/4-chip)→
←Channel (144-bit/36-chip)→

Figure 3.5: Octuple Pin Correcting Bamboo ECC on a (128+16)-bit channel.

and reports a DUE. The rationale behind this decision is that errors on different chips over a single access are very rare so that the symptom is likely to have been generated from miscorrections arising from a severe error. Similarly, QPC pessimistically reports a DUE if the diagnosed pin errors are not confined to a single chip or to two pins on different chips.

3.3.1.4 Octuple Pin Correcting ECC

Bamboo ECC can correct errors on two ×4 chips or one ×8 chip by correcting 8 pin symbols. *Octuple Pin Correcting (OPC)* Bamboo ECC achieves 8 pin correction with 16 redundant pins, resulting in a 25% overhead on a 64-bit data channel or a 12.5% overhead on a 128-bit data channel (Figure 3.5). OPC is optimized in a manner similar to that of QPC and limits its correction capability to 2 chip errors or 4 independent pin errors to reduce the SDC rate.

3.3.2 Graceful Downgrade

The plethora of attractive organizations above allows a Bamboo ECC-based system to more fully utilize *redundant bit steering (RBS)* than existing

ECC schemes. RBS is a graceful downgrade scheme proposed by IBM [21] and uses parts of the ECC check bits as hot spares, remapping faulty pins through these spares. The state-of-the-art RBS-based protection is *Double Device Data Correction (DDDC)*+1 from Intel, which can tolerate two sequential chip failures and a bit failure over a 128-bit data channel (Section 3.5). A gracefully downgrading Bamboo ECC-based system can correct more initial chip faults, more sequential chip faults, more end-of-life faults and has a better end-of-life detection capability than DDDC+1. In addition, a gracefully degrading Bamboo ECC system can diagnose and remap errors at the pin granularity, offering slower degradation for accumulating pin errors than a system that operates on coarse-grained symbols. Such a gracefully degrading Bamboo ECC scheme is described below.

OPC over a 128-bit data channel uses the same amount of redundancy (12.5%, or 16 pin symbols) as DDDC+1. After a chip retirement, the available redundancy decreases to 12 pins, but Bamboo ECC can still operate in *hextuple pin correcting* mode to correct up to 6 pin errors. Successive pin errors can be diagnosed and retired at the pin granularity–a luxury that non-Bamboo ECC codes do not enjoy.

After the available spare pins are exhausted due to further pin or chip retirement, the redundancy of the Bamboo ECC-based system decreases down to 8 pins, which is sufficient for QPC. Finally, following a third round of pin or chip retirement, the system can downgrade to SPC-TPD using the remaining 4 redundant (non-spare) pins. Due to the storage efficiency of SPC-TPD, the

Bamboo ECC-based system will still be able to correct a pin error and detect $\geq$ 99.9996% of errors. As a result, this graceful downgrade scheme can correct two concurrent chip errors in its initial OPC phase and then can correct up to 1 sequential chip error and 1 pin error. It can also handle finer grained errors, retiring faulty bits or pins as they accumulate.

Bamboo ECC-based graceful downgrade enjoys superior flexibility, and can be modified to work on a narrower channel. For a system with a 64-bit data channel, QPC (using 8 redundant pins) can be gracefully downgraded to SPC-TPD (with a 4 pin redundancy), correcting a sequence of 1 chip error and 1 pin error or up to 5 sequential pin errors while detecting most end-of-life errors.

3.3.3 Overheads

The large ECC symbols and codewords of Bamboo ECC provide strong and efficient error protection at the expense of decoding complexity and latency. This section examines these costs, showing that the additional implementation overheads of Bamboo ECCs are modest and well-aligned to current technological trends. It is demonstrated later that Bamboo ECCs incur little performance overhead, and is expected that their memory bandwidth savings (for the same level of protection) will outweigh their costs.

Circuit overhead: RS codes with 8-bit symbols have modest encoding and decoding overheads to have a wide range of commercial applications, ranging from CDs to satellites. AMD chipkill uses 8-bit RS codes with 16 data symbols and 2 redundant symbols; a corresponding fully parallel encoder requires 6 XOR2 gates of delay and consumes an area equivalent to about $1,600$ NAND2 gates.[3]

With larger codewords, Bamboo ECCs have larger encoding/decoding overheads. Specifically, a fully parallel encoder for QPC (64-symbol data + 8-symbol redundancy) is 8 XOR2 gates deep, which is 2 gates more than AMD chipkill yet is still easily implementable within a single memory cycle (1.2GHz). Even the most complex Bamboo ECC presented, OPC with 128-symbol data and 16-symbol redundancy, requires only 10 logic levels. The fully parallel QPC encoder consumes about 25,000 NAND2 gates of area, a ×16 increase over that of AMD chipkill. DRAM data, however, does not arrive at the chip in an extremely wide parallel interface, rather transfers over multiple cycles, and as such narrower pipelined RS encoder [26] can be used to reduce the area of the encoder. Using a simplified gate-level model, a 16-way parallel QPC encoder derived from [26] is estimated to require 5,500 gates and a 32-way parallel encoder for OPC will require 11,000 gates. These area overheads do not represent a large amount of chip real estate considering the billions of gates available in recent processors and current trends indicate that logic cost and speed will continue to scale down more rapidly than DRAM [47].

[3] These delay and area estimates are found through standard-cell synthesis using the Synopsys toolchain and the 40nm TSMC standard cell library [135, 136], but are presented in a technology-independent manner.

Performance overhead: ECC decoding procedure consists of two phases: error detection, which happens on every memory read, and error correction, which happens with rare memory errors. Error detection is as simple as encoding and Bamboo ECC does not increase cycle latency compared to other ECC mechanisms. An 8-bit RS symbol correction can be done within two extra cycles (estimation based on [69, 24]) yet happens only in the rare cases of a memory error and does not affect performance in the common case.

Bamboo ECC introduces some additional latency in the memory controller due to the alignment of pin-based symbols. Until the completion of a transfer, per-pin symbols are not fully available, which may delay encoding and decoding. While encoding is on the less latency-critical write path (and data are usually buffered before writes), decoding is on the latency-critical read path. Bamboo ECC takes 4 DRAM cycles to transfer symbols (DDR with burst length of 8), compared to 1 cycle in existing ECC schemes with 1- or 2-beat codewords. Section 3.4.3 evaluates the performance cost from this increased latency, and shows that the impact is minor. This performance impact can decrease further in future memories, as DRAM is aggressively reducing cycle time to increase data bandwidth.

One way to overcome this additional latency is to use an asynchronous ECC check [103]. With asynchronous ECC checking, data are speculatively forwarded to the processing unit before decoding is complete. Later, if an error is detected, the forwarded data and any dependent calculations are discarded and corrected data are sent throughout the system. To be conservative, this dissertation does not apply an asynchronous ECC check—data are forwarded only after they are determined to be error-free.

Because Bamboo ECC requires an entire cache line of data for encoding and decoding, a *partial-update* of a cache line requires a *read-modify-write* to read existing values, re-generate check bits, and write them back to memory. Partial-update is a known issue for many existing ECC schemes [9]. Fortunately, when using a write-back last-level cache, all write-hits are buffered in the cache and flushed to main memory at the full cache-line granularity. As a result, only uncached or uncacheable partial writes (which are often not performance critical) will require read-modify-writes; this is not expected to significantly alter system performance or efficiency.

3.4 Evaluation

This section measures the error correction and detection coverage, system failure rate, and performance impact of Bamboo ECC schemes and compares them with other state-of-the-art single-tiered ECC. The range and flexibility of Bamboo ECC leads to a design space of error control schemes that vary in their error coverage, redundancy, and expected system lifetime. This design space is evaluated below, demonstrating the substantive strength, safety, and flexibility advantages of Bamboo ECC.

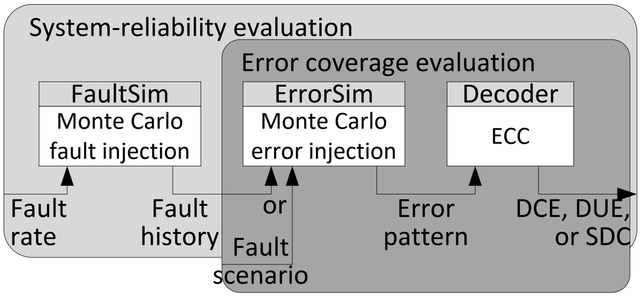

System-reliability evaluation
Error coverage evaluation
FaultSim
Monte Carlo fault injection
ErrorSim
Monte Carlo error injection
Decoder
ECC
Fault rate
Fault history
or
Fault scenario
Error pattern
DCE, DUE, or SDC

Figure 3.6: The protection coverage and system reliability evaluation environment.

3.4.1 Error Coverage Evaluation

The error correction/detection coverage of each ECC scheme is evaluated using Monte Carlo error injection experiments (Figure 3.6). Errors based on 5 fault models (bit/pin/word/chip/rank faults) are generated and injected into a cache line-sized memory block. These models represent faults at different levels of memory structures (e.g., cells, mats, chips and ranks) and they match the fault modes in recent DRAM field studies [128, 129, 127]. A bit-fault indicates that the block has a single bit-error at a random position, and a pin-fault represents a block that has a single corrupted DQ pin. Similarly, a word/chip/rank fault corrupts all bits within a single-chip/single-beat, a single-chip/all-beats, and all-chips/all-beats, respectively. A fault scenario can be one of the 5 fault models or a combination of them.

With given fault scenario, errors are randomly generated and injected into a block, assuming that each bit within a corrupted region has a 50% switching probability (but the error-free pattern is excluded). Error patterns are tested by each ECC scheme to determine whether the error is detectable, correctable, or miscorrected. An undetected error and a miscorrected error are categorized as SDC, while a detectable-but-uncorrectable error is reported as a DUE. If any word within a block reports a DUE, the block is reported as DUE. Similarly, if any word in a non-DUE block reports an SDC, the block is classified as SDC. Finally, if all the words are corrected to their original data, then the block is marked as a *detectable and correctable error (DCE)*. The number of experimental runs is 10 billion with 99.9% confidence intervals of ±0.0000001% for probabilities near 0.0000001% or 99.9999999%, and ±0.0003% for probabilities near 1% or 99%.

Table 3.2 shows error coverage of bit-level protection schemes (SEC-DED, SPC, and SPC-TPD) on a 64-bit data channel. SEC-DED, SPC, and SPC-TPD can all correct a bit or a pin fault. The single pin correction capability of SPC and SPC-TPD shows worse correction than SEC-DED in some fault scenarios as it cannot correct multiple 1-bit-error-per-beat errors with different DQ positions over multiple beats (a situation that is expected to be rare). However, SPC and SPC-TPD require only one quarter and one half the redundant storage of SEC-DED, respectively. The detection capability of SPC-TPD is very strong, detecting $\geq$99.9996% of all errors in all scenarios, compared to the weak error detection capabilities of SEC-DED (up to 23.5% SDC).

Table 3.3 shows the error coverage of SDDC schemes: AMD chipkill and QPC on a 64-bit data channel with ×4 devices and Single 8-bit Symbol Correcting codes (S8SC) and OPC on a 128-bit data channel with ×8 devices. S8SC builds 8-bit symbols using per-beat data from ×8 chip and applies the

		SEC-DED	Bamboo	Bamboo
			SPC	SPC-TPD
Codeword (bits x beats)		72 x 1	66 x 8	68 x 8
Codewords per MTB		8	1	1
Redundancy %		12.5%	3.13%	6.25%
1 bit/pin (%)	DCE	100.0000	100.0000	100.0000
1 word fault (%)	DCE	26.6667	26.6683	26.6667
	DUE	55.5553	47.9143	73.3333
	SDC	17.7780	25.4172	0.0000
1 chip fault (%)	DCE	0.0091	0.0000	0.0000
	DUE	98.8502	74.3182	99.9996
	SDC	1.1407	25.6817	0.0004
1 bit fault + 1 bit fault (%)	DCE	87.5000	0.0000	0.0000
	DUE	12.5000	75.4343	100.0000
	SDC	0.0000	24.5657	0.0000
1 bit fault + 1 pin fault (%)	DCE	49.8036	0.0000	0.0000
	DUE	50.1964	74.9024	100.0000
	SDC	0.0000	25.0976	0.0000
1 bit fault + 1 chip fault (%)	DCE	0.0018	0.0000	0.0000
	DUE	99.3297	74.3199	99.9996
	SDC	0.6685	25.6801	0.0004
1 pin fault + 1 pin fault (%)	DCE	9.3043	0.0000	0.0000
	DUE	90.6957	74.9016	100.0000
	SDC	0.0000	25.0984	0.0000
1 pin fault + 1 word fault (%)	DCE	13.2878	0.0000	0.0000
	DUE	63.1743	74.4093	100.0000
	SDC	23.5380	25.5907	0.0000
1 pin fault + 1 chip fault (%)	DCE	0.0001	0.0000	0.0000
	DUE	99.7537	74.3173	99.9996
	SDC	0.2462	25.6827	0.0004
1 chip fault + 1 chip fault (%)	DCE	0.0000	0.0000	0.0000
	DUE	99.9553	74.3190	99.9996
	SDC	0.0447	25.6810	0.0004
1 rank fault (%)	DCE	0.0000	0.0000	0.0000
	DUE	99.9956	74.3202	99.9996
	SDC	0.0044	25.6798	0.0004

Table 3.2: A comparison of the protection coverage of bit-level ECCs.

Chip data width		×4		×8	
Channel configuration		1 ECC DIMM		2 ECC DIMMs	
ECC		AMD chipkill	Bamboo QPC	S8SC	Bamboo OPC
Codeword (bits x beats)		72 x 2	72 x 8	144 x 2	144 x 8
Codewords per MTB		4	1	4	1
Redundancy %		12.5%	12.5%	12.5%	12.5%
1 bit/pin/ word/chip (%)	DCE	100.0000000	100.0000000	100.0000000	100.0000000
1 bit fault + 1 bit fault (%)	DCE	0.0000000	100.0000000	0.0000000	100.0000000
	DUE	98.9252058	0.0000000	99.4626331	0.0000000
	SDC	1.0747942	0.0000000	0.5373669	0.0000000
1 bit fault + 1 pin fault (%)	DCE	0.0000000	100.0000000	0.0000000	100.0000000
	DUE	99.9399162	0.0000000	99.9831498	0.0000000
	SDC	0.0600838	0.0000000	0.0168502	0.0000000
1 bit fault + 1 word fault (%)	DCE	0.0000000	26.6685560	0.0000000	3.1369578
	DUE	98.6450420	73.3314440	99.2156554	96.8630422
	SDC	1.3549580	0.0000000	0.7843446	0.0000000
1 bit fault + 1 chip fault (%)	DCE	0.0000000	0.0000262	0.0000000	0.0000000
	DUE	99.9999993	99.9999734	100.0000000	100.0000000
	SDC	0.0000007	0.0000004	0.0000000	0.0000000
1 pin fault + 1 pin fault (%)	DCE	0.0000000	100.0000000	0.0000000	100.0000000
	DUE	99.9887495	0.0000000	99.9831536	0.0000000
	SDC	0.0112505	0.0000000	0.0168464	0.0000000
1 pin fault + 1 chip fault (%)	DCE	0.0000000	0.0000259	0.0000000	0.0000000
	DUE	99.9999998	99.9999737	100.0000000	100.0000000
	SDC	0.0000002	0.0000004	0.0000000	0.0000000
1 chip fault + 1 chip fault (%)	DCE	0.0000000	0.0000000	0.0000000	0.0000000
	DUE	99.9999995	99.9999996	100.0000000	100.0000000
	SDC	0.0000005	0.0000004	0.0000000	0.0000000
1 rank fault (%)	DCE	0.0000000	0.0000000	0.0000000	0.0000000
	DUE	100.0000000	99.9999994	100.0000000	100.0000000
	SDC	0.0000000	0.0000006	0.0000000	0.0000000

Table 3.3: A comparison of the protection coverage of SDDC ECCs with ×4 and ×8 DRAM devices.

history mechanism used by AMD chipkill. Among the ×4 SDDC schemes, QPC has a better correction capability than AMD chipkill because of its ability to correct two independent pin errors (e.g., 1 bit fault + 1 bit fault). In addition, QPC has a very strong detection capability of $\geq 99.9999994\%$ in all scenarios, compared to the strong-yet-incomplete detection capability of AMD chipkill (up to 1.4% SDC in some scenarios). While the detection coverage of QPC is not a perfect 100% (a few pathological error patterns can result in SDC), it applies not only to two-chip faults but also to many-chip faults that represent very severe errors.

Despite the doubled redundancy for ×8 SDDC, S8SC still suffers from up to 0.5% SDC with single-beat faults, where history-based mechanism is not effective. Instead, no SDC is observed with OPC during the 10 billion runs, proving its vast superior detection capability.

3.4.2 System Reliability

The error coverage results from Section 3.4.1 show that Bamboo ECC provides superior error correction and vastly improves error detection relative to prior single-tiered SDDC solutions. The true mettle of an error control system, however, is tested by how much it improves the failure rate of a large system at scale.

A two-stage Monte Carlo simulation is used to evaluate the failure rates over time of a large system with different ECC schemes (Figure 3.6). The first stage injects faults into a simulated DRAM channel based on observed fault

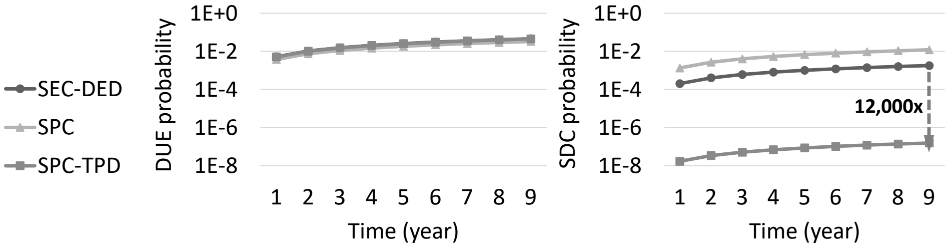


Figure 3.7: The failure probability of a DDR 64-bit data channel over time with different bit-level protection mechanisms.

modes and rates [128]. This first-stage fault injection methodology is similar to that of [118], yet uses more detailed fault modes broken down by the number of failing DQs to better evaluate the behavior of the pin-based Bamboo ECC. Once (possibly) overlapping faults are identified by the first stage, the second stage of Monte Carlo simulation maps the fault modes into one of the fault models described in Section 3.4.1 and generates error patterns based on the fault scenario. The error pattern is passed to ECC decoding in order to judge its outcome. This realistically models the practical error coverage of each ECC codes, as described in Section 3.2.

Figure 3.7 shows the failure rate of a single 64b data channel (2 ranks, 18 ×4 chips per rank) over time using different bit-level ECC mechanisms. SPC-TPD has a slightly (1.2%) lower DUE probability and a $12,000\times$ lower SDC probability than SEC-DED, despite requiring only half as much redundancy. The lower DUE rate of SPC-TPD is due to this lower redundancy, as the correspondingly lower raw fault rate is able to compensate for the slightly weaker correction capability of the code. Using only a quarter of the redundant storage, SPC reduces DUEs by 39% compared to SEC-DED as it has an

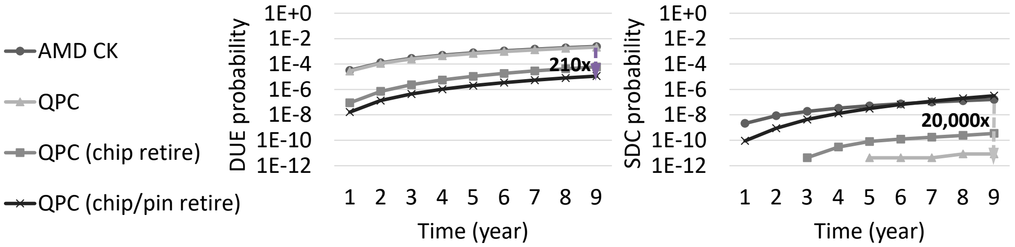


Figure 3.8: The failure probability of a DDR 64-bit data channel over time with different SDDC protection mechanisms.

even lower raw fault rate than SPC-TPD. However, the weaker error detection capability of SPC results in a 15× higher SDC probability than SEC-DED.

The failure rate of a 64b data channel (2 ranks, 18 ×4 chips per rank) using SDDC protection is shown in Figure 3.8. QPC has 12% lower overall failure probability than AMD chipkill because it can correct two independent single-DQ faults. In addition, the strong error detection coverage of QPC results in a 20,000× lower SDC probability. Graceful downgrade using QPC can correct a sequence of one chip and one pin (QPC to SPC-TPD with chip retirement) or a sequence of one chip and up to three pins (QPC to SPC-TPD to SPC with chip/pin retirement). The stronger correction of downgrade-based schemes lowers the overall failure probability of chip and pin-based retirement to 0.02 and 0.005 that of AMD, respectively. The SDC probability increases with downgrade schemes, however, as they continue operation with reduced-strength codes; despite this, downgraded QPC still demonstrates an SDC rate that is comparable to, or better than that of AMD.

Figure 3.9 demonstrates the failure rate of a 128b DDR data channel (2

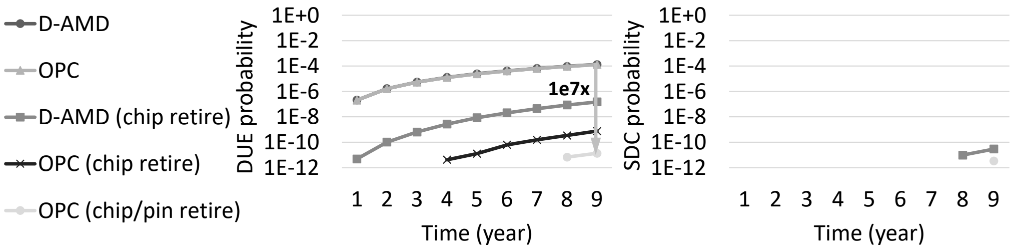


Figure 3.9: The failure probability of a DDR 128-bit data channel with different DDDC protection mechanisms.

ranks, 36 ×4 chips per rank). In this configuration, AMD chipkill was modified to provide *Double Device Data Correction (DDDC)* protection (i.e., {36,4} codes with history-based miscorrection detection), and is used as a baseline for OPC. OPC has a 2% lower failure probability than doubled AMD chipkill due to its ability to correct independent single-DQ faults. No occurrence of SDC is observed with OPC during the 200B runs, while doubled AMD chipkill shows a 10^{-10} probability of SDC. Doubled AMD chipkill can correct a sequence of up to 3 faults by gracefully degrading through chip retirement to AMD chipkill. The superior flexibility of OPC can correct a sequence of up to 3 chip faults and 1 pin fault by degrading down to SPC-TPD with chip retirement or a sequence of up to 3 chip and 3 pin faults by degrading down to SPC with pin retirement.

Full-system estimates and lifetime considerations: Figure 3.10 illustrates the failure probabilities of SDDC protection mechanisms on a large system with one million memory channels, which is an order of magnitude larger than the Jaguar system [128] and could match an exascale-computing

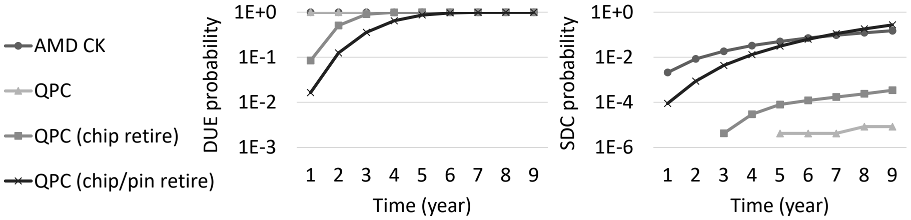


Figure 3.10: The failure probability of a large system with 1,000,000 channels over time with different SDDC protection mechanisms.

system. Several findings are readily apparent. First, the massive number of DRAM channels makes the memory system quickly reach a state where some locations report uncorrectable errors and must be repaired or replaced. RBS-based graceful degradation can be used to combat this rapid wearout of the memory system; by employing chip retirement and pin retirement, QPC can decrease the probability of a system repair in the first year down to 8.5% and 1.7%, respectively; beyond that point, a higher-level repair mechanism is required. Also, the safety of QPC relative to AMD chipkill is demonstrated through greatly reduced SDC rates; this safe detection can be combined with higher-level repair scheme to ensure reliable operation of the huge system over a long period of time.

There is a natural tradeoff between lifetime extension and guaranteed safety—an aggressive lifetime-extending scheme, such as Bamboo ECC with per-pin retirement, will spend the majority of its time in a degraded mode by design; as such, its safety may suffer compared to a code that conservatively reports a DUE. Through per-chip and per-pin retirement, Bamboo ECCs present a range of options that trade off safety and lifetime. More importantly,

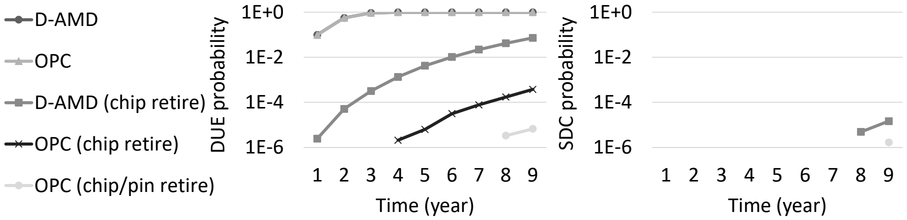


Figure 3.11: The failure probability of a large system with 1,000,000 channels over time with different DDDC protection mechanisms.

even the most aggressively degrading Bamboo ECC schemes maintain comparable SDC rates to AMD chipkill, such that even a long-lifetime Bamboo ECC organization will not compromise system safety relative to the current state-of-the-art.

To compare the system-level reliability among SDDC and DDDC mechanisms, Figure 3.11 presents the failure probabilities of DDDC schemes on a system with the same number of DIMMs but half the channels. Note that this wider channel configuration significantly sacrifices performance and energy efficiency for improved reliability. The double device correction capabilities have limited improvements on DUE probability while their bigger codewords provide very safe detection capability (i.e., no SDC reported). RBS mechanisms based on Bamboo ECC provide significantly better DUE and SDC probabilities than the one based on doubled AMD chipkill in this configuration as well.

3.4.3 Performance and Energy

The performance cost of Bamboo ECC is modeled by running the SPEC CPU 2006 benchmark suite [130, 39] on the Gem5 simulator [1, 19] (version 2.0). A 2GHz single-core processor with a $32KB$ I-cache, $64KB$ D-cache, and a $2MB$ L2 cache is used with 2GB of DRAM and an L2 cache stride prefetcher. The latency overhead of the ECC schemes is modeled in the read data queue and write request queue of the memory controller. Existing ECC schemes are given a +1 (memory) cycle penalty for both reads and writes, while Bamboo ECC schemes have a +4 cycle read and a +1 cycle write penalty. This 3 additional cycle increase models the waiting time for all beats of each symbol to be transferred. There are 2 memory configurations: DDR4-2400 (1.2GHz DDR) 64b data channel and DDR4-2400 128b data channel. For the 128b channel, the cache line size is increased from 64B to 128B to match the DRAM access granularity. The measurement period is 0.2 billion instructions using detailed simulation after 0.5 billion instructions of functional simulation (to warm up the caches). Among the 29 SPEC CPU2006 benchmark applications, 26 are evaluated; perlbench, tonto, and dealII fail to run due to simulator issues.

Figure 3.12 shows the execution time comparison between existing ECC schemes and Bamboo ECC. The benchmarks are sorted based on their memory traffic. Bamboo ECC shows 0.7% and 0.5% execution cycle increases compared to AMD chipkill on DDR4-2400 64b and DDR4-2400 128b, respectively. With memory-intensive applications, the slowdown increases to 2.3%

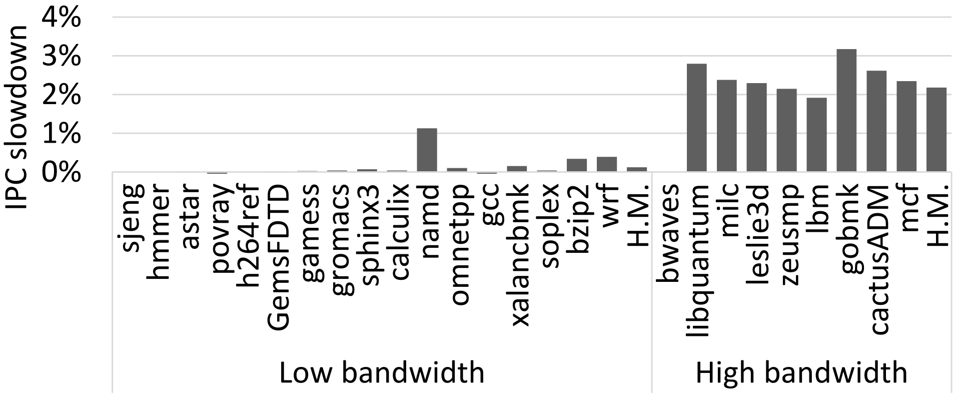


Figure 3.12: The execution time slowdown of Bamboo ECC schemes (normalized to AMD chipkill).

and 1.7%, respectively. However, with the trend of increasing DDR bandwidth and clock frequencies, the waiting time for the symbol transfer and its performance impact should decrease. The execution cycle increases in Bamboo ECC will be much smaller than the performance degradation from increasing the data width from 64b to 128b for stronger ECC protection, which is reported to be very performance limiting by Fujitsu [32].

DRAM energy consumption is estimated based on the Micron model [95] with DDR4-2400 parameters [96]. The average differences between the existing ECC and Bamboo ECC schemes are 0.5% and 0.4% on 64b and 128b channels, respectively. This increased energy consumption is mostly due to the increased execution times, as the number of activations and data transfers are nearly identical.

3.5 Related Work

ECC has long been used to detect and correct DRAM errors. A brief review of prior single-tiered error protection approaches is presented below. An emphasis is made on deciphering the state-of-the-art memory protection schemes used by industry.

SEC-DED: A simple but widely-used ECC scheme for DRAM applies a *Single Error Correcting-Double Error Detecting (SEC-DED)* code to each beat of a memory transfer (Figure 3.13). On a 64b data channel, 8b of redundancy are needed for SEC-DED, leading to the industry-standard 72b ECC DIMM with 12.5% redundancy.

Interleaved SEC-DED Codes: One straightforward way to provide SDDC-level memory protection is to interleave four SEC-DED codewords together. By distributing data from a ×4 DRAM chip over 4 different codewords, single-device-data-correction and double-device-data-detection can easily be achieved (Figure 3.14). While such an approach was employed by IBM, HP, and EMC in the past [29], it requires a 256b data channel. Forming such wide channels are often disastrous to system performance and efficiency [32] and thus are not competitive with Bamboo ECC.

Interleaved 2-bit Codes: Using the same methodology as interleaved SEC-DED codes, SDDC can be implemented by interleaving two *Single Er-*

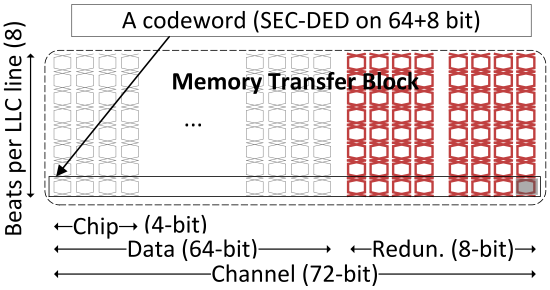


Figure 3.13: The SEC-DED codeword over (64+8)-bit channel / 1-beat.

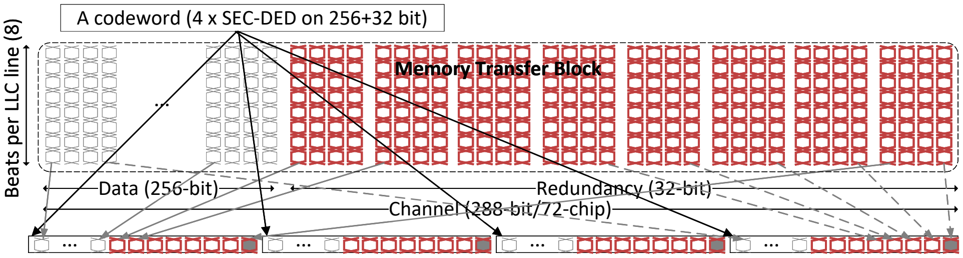


Figure 3.14: The interleaving of 4 SEC-DED codewords over 288 bit channel / 1-beat.

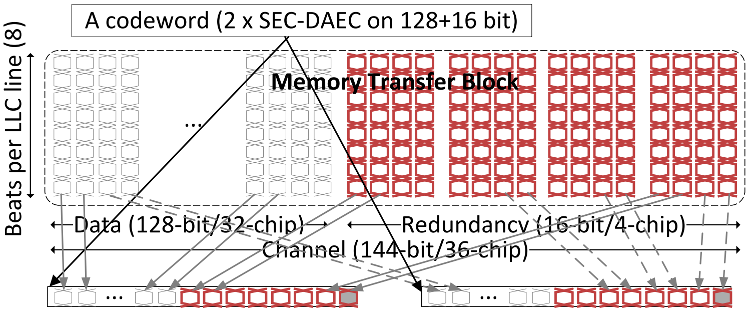

A codeword (2 x SEC-DAEC on 128+16 bit)
Beats per LLC line (8)
Memory Transfer Block
...
Data (128-bit/32-chip)
Redundancy (16-bit/4-chip)
Channel (144-bit/36-chip)

Figure 3.15: The interleaving of 2 SEC-DAEC codewords over 144 bit channel / 1-beat.

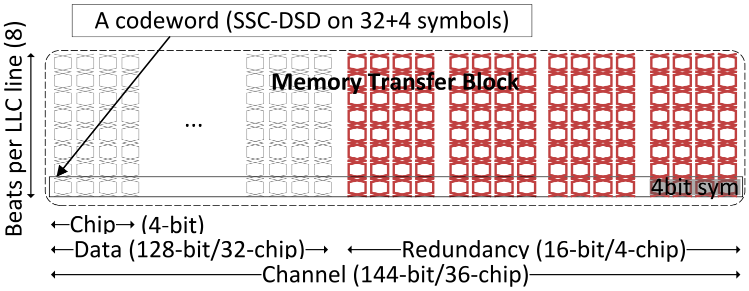


Figure 3.16: The 4-bit Symbol SDDC (SSC-DSD) over (128+16)-bit channel / 1-beat.

ror Correcting – Double Adjacent Error Correcting (SEC-DAEC) ECCs (Figure 3.15). Some IBM and Compaq products seem to have used this approach [51, 7]. This scheme requires a 144b channel, and only provides SDDC—poor detection capabilities are provided following this correction [90]. The wide channel and poor handling of accumulating errors preclude this scheme from being competitive with Bamboo ECC.

4-bit RS Codes: Sun and older AMD chips use a 4-bit symbol *Single Symbol Correcting-Double Symbol Detecting (SSC-DSD)* RS codes to provide SDDC on ×4 DRAM chips (Figure 3.16) [134, 8]. The symbols are aligned to chip boundaries so that a chip-fault is confined to a single symbol and can be corrected by SSC-DSD. Four symbols of redundancy are needed to provide SDDC protection because of the narrow (4-bit) symbol size. This scheme requires a 144b memory channel, and as such cannot compete with the efficiency of Bamboo ECC.

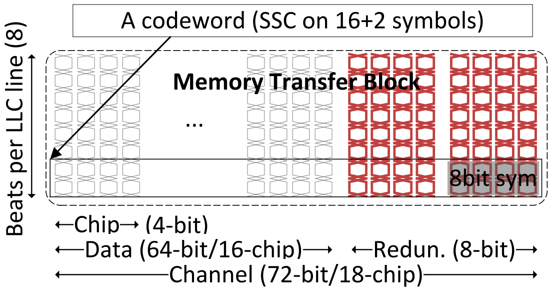


Figure 3.17: The 8-bit Symbol SDDC (SSC) over (64+8)-bit channel / 2-beat.

8-bit RS Codes: Newer AMD chips use a *Single Symbol Correcting (SSC)* RS codes with an 8-bit symbol to provide SDDC on ×4 DRAM chips [9, 41]. The 8-bit symbols are built from two beats from the same ×4 chip so that a chip fault is confined to a single symbol (Figure 3.17). The large, 8-bit symbol size achieves SSC with 2 redundant symbols, allowing a 72b memory channel to be serviced by a single DIMM. Operating on a narrow channel makes this scheme efficient; as such it is the closest related work to Bamboo ECC.

Because this scheme does not provide *Double Symbol Detecting (DSD)* protection with a 72b memory channel, 2-chip faults may lead to silent data corruption. To mitigate these concerns, AMD uses the history of corrected symbol locations within each cache line to heighten error detection (Section 3.3.1.3).

Redundant Bit Steering and Double Device Data Correction *Double Device Data Correction (DDDC)* is provided by some products for applications that demand higher levels of protection. *Redundant bit steering* (RBS)[4] is a technique that was developed by IBM for enterprise mainframe computers to provide DDDC protection [21]. Remapping is done in the memory controller and is transparent to both the OS and the user.

It seems as if DDDC capabilities are provided in current products through RBS. Most notably, Intel's *Double Device Data Correction (DDDC)* [42] (referred to by HP as Double ChipSpare [40]) appears to correct two sequential chip-errors by applying a chip-level protection through dynamic bit steering. If a chip fails, a spare chip is used to replace the failed chip. More recent products provide *DDDC+1*, which is able to correct an additional single bit-error on top of *DDDC* [42].

While the exact details of RBS are unknown publicly at this time, a sensible scheme that matches the reported redundancy requirements of commercial products follows. An RS code with 4-bit symbols and a 128b data channel requires a 3-symbol check code to provide SSC protection and an extra symbol to provide SSC-DSD. At the beginning of system operation (assuming no faults), memory uses all available pins to provide SSC-DSD protection. Upon a detected chip or pin error, the memory controller downgrades all affected memory to an SSC code and remaps the faulty chip through the fourth redundant symbol.

By downgrading affected memory locations from an SSC-DSD code to

[4] Now referred to as IBM *Memory ProteXion*.

an SSC code, a memory system can tolerate up to 2 successive chip failures (DDDC-level protection). It seems likely that DDDC+1 downgrades protection to an SEC code following a second successive chip failure to provide end-of-life bit-correction capabilities. Bamboo ECC is designed to be highly amenable to RBS. Compared to other SDDC ECCs, Bamboo ECCs can provide a fine-grained retirement (such as a pin retirement) to face single-DQ faults. In coordination with retirement, Bamboo ECC can provide superior correction capabilities for sequential faults, correcting two concurrent chip-faults and up to 3 sequential chip faults and 3 pin faults on the same 128b channel as DDDC+1.

3.6 Summary

This chapter presents and analyzes a family of strong error checking and correcting mechanisms for DRAM called Bamboo ECC. Bamboo ECC provides superior efficiency, operating as single-tiered mechanism that offer stronger-than-SDDC protection over a 64b data channel. Meanwhile, Bamboo ECC is strong and safe, delivering increased correction capabilities relative to the state-of-the-art single-tiered DRAM schemes while simultaneously decreasing the silent data corruption rate. It shows that Bamboo ECC is amenable to graceful downgrade using redundant bit steering and is able to offer an unprecedented level of accumulating error protection. Bamboo ECC with RBS also demonstrates superior flexibility, and is able to retire finer grained errors and operate on narrower channels than the current state-of-the-art ECC mechanisms, potentially extending the memory system lifetime.

Chapter 4

Mechanisms against Manufacturing Faults

This chapter presents and evaluates mechanisms that enable efficient and reliable memory protection against imperfect process scaling. As process technology scales past 10nm, it is getting more difficult to manufacture a fault-free DRAM chip even when considering row and column sparing techniques. Accordingly, DRAM vendors and academics have proposed *in-DRAM ECC* to maintain high yield. In-DRAM ECC can correct bit-level errors from inherent faults, improving manufacturability (Section 2.1). Meanwhile, high reliability systems continue to require stronger-than-bit-level protection against severe errors from operational faults. Current high-reliability systems employ SDDC ECC across a rank using extra chips and can restore data even if one chip fails. Separate management of in-DRAM ECC and rank-level ECC is wasteful and requires up to 26.6% storage overheads, compared to the 12.5% redundancy that is standard today.

Active Guardband ECC (AG-ECC) [80] utilizes a novel and unique combination of ECC decoding and memory remapping that simultaneously achieves high efficiency, high reliability, and high manufacturability. AG-ECC uses a strong rank-level ECC to tolerate most inherent and operational faults by treating their errors as completely random. Because of redundancy and granularity constraints, this common decoding for correcting random errors is insufficient in rare cases where severe operational faults overlap inherent faults. In such cases, AG-ECC uses the same rank-level ECC as a combination of erasure and random error decoding followed by remapping. This preserves the efficiency of future accesses and maintains reliability against additional operational faults. As a result, AG-ECC can achieve stronger-than-in-DRAM-ECC manufacturability and stronger-than-SDDC reliability with a total redundancy of just 12.5% without introducing new capabilities or storage within DRAM chips.

4.1 DRAM Faults from Process Scaling

Shrinking process technology makes DRAM more prone to inherent faults for several reasons. A DRAM cell is composed of a transistor and a capacitor. The smaller the transistor becomes, the more vulnerable it is to minor defects. At the same time, smaller capacitors have more difficulty in maintaining the required charge for correct operation due to aspect ratio constraints [99, 70]. Narrower pitches can result in disturbance errors, such as row hammering [82]. Increasing variability at smaller sizes can also cause severe fluctuation in DRAM cell retention time. While the details of inherent faults are unknown publicly at this time, prior research observes that they manifest as random bit errors [99, 72, 16, 115].

The increasing prevalence of inherent faults worsens DRAM yield and

	Reliability	Manufacturability
Obstacles	Operational faults (e.g., transistor wear-out)	Inherent faults (e.g., high-leakage cells)
Metric	Failure In Time (FIT)	Yield (%)
Cost center	System loses dependability	Vendor loses money
Correction	Required	Not required
Removal	Not required	Required

Table 4.1: Traditional approaches for high reliability and high manufacturability.

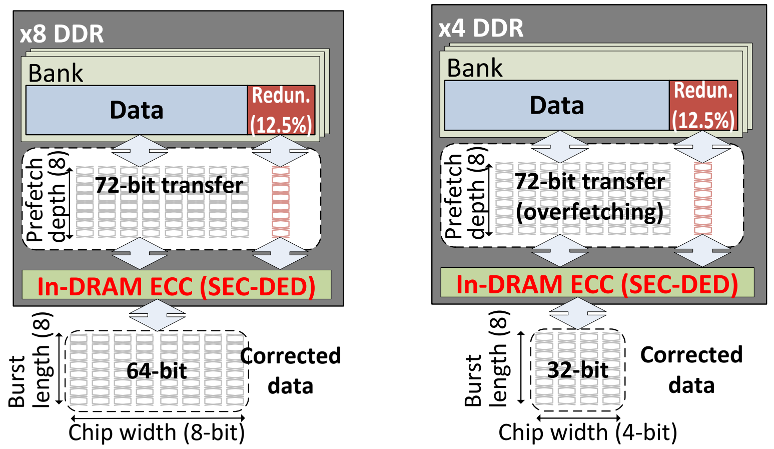
manufacturability, and screening the faults out at manufacture time is difficult because some of the faults generate errors occasionally (e.g., due to variable retention time). Such faults that escape testing may generate errors during operation, degrading system reliability. Despite this similarity, it is important to clearly distinguish these two separate goals, quantify their requirements, and optimize based on relevant metrics for each. Table 4.1 compares and contrasts the traditional focus of reliability and manufacturability techniques. While they both originate from faults in DRAM, their impact, cost center, and recovery requirements differ. AG-ECC simultaneously targets both reliability and manufacturability using a principled combination of ECC and fine-grained retirement to maintain both high yield and a low FIT rate.

4.2 In-DRAM ECC

The DRAM process has been highly optimized so far to have few inherent faults. Inherent faults, if any, have been removed by coarse-grained remapping schemes (e.g., spare rows and columns) to increase yield [137].

With increasing prevalence of inherent faults and limited testing time, DRAM vendors now provide *post package repair* [60, 67]. Post package repair allows a memory controller to change a few memory row mappings during operation to retire unscreened inherent faults and some operational faults. While current industry practices tend towards coarse-grained remapping, academia proposed fine-grained remapping at cache line granularity [99, 74], chip granularity [75], and bit granularity [126]. These schemes support larger numbers of remappings by reducing remapping overhead and can maintain high reliability in the face of permanent operational faults.

Memory remapping alone is not sufficient, especially given the current trend of increasing fine-grained inherent fault rates [45]; at a *Cell Fault Rate (CFR)* of 10^{-5} (one out of every hundred-thousand cells is faulty), a 16GB ECC DIMM has 1.5 million 64B blocks (0.5%) that need to be remapped. To improve manufacturability without the need for expensive remapping schemes and to tolerate latent inherent faults that escape testing, academics [155, 85] and DRAM suppliers [68, 81, 70, 107] are considering in-DRAM ECC. In-DRAM ECC stores ECC check bits in redundant array storage within each DRAM chip, generating this ECC internally before data is written and correcting any single-bit errors inline before sending data back to the host processor. In-DRAM ECC has been employed in low-power commodity DRAMs to improve yield and decrease operating voltages [68, 14, 81, 67, 107], as well as for refresh reduction [81, 155], and it is under standardization as a DDR4 extension.



(a) A ×8 chip. Each 72b word is fetched and decoded by SEC-DED ECC. All 64b data after correction are transferred.

(b) A ×4 chip. Each 72b word is fetched and decoded by SEC-DED ECC. Only 32 of the 64b data are transferred to the processor.

Figure 4.1: In-DRAM ECC for DRAM chips with different data interface widths but the same 12.5% redundancy.

In-DRAM ECC not only costs array storage and a modest amount of encoding and decoding logic within DRAM, but it also degrades performance [81, 70]. ECC decoding increases access latency and a mismatch between write data granularity and internal ECC codeword size requires read-modify-write operations that can also degrade performance. The best organization of in-DRAM ECC is an open research topic. In one proposal, in-DRAM ECC uses 8 bits of redundancy per 64 bits of data to provide internal SEC-DED protection [70]. On a ×8 chip (Figure 4.1a), the ECC data size matches the transfer granularity of the chip and implementation is straightforward. However, on a ×4 chip (Figure 4.1b), whose access granularity is 32*b*, the 64*b* ECC data size results in overfetching read data and read-modify-writes for write commands. Another in-DRAM ECC candidate increases the ECC word size to 128 bits to amortize ECC overheads further [85, 70]. This decreases the amount of redundancy needed for SEC to 6.25%, but it exacerbates the overfetching problem and decreases protection.

In-DRAM ECC is innately weaker than rank-level ECC because of its per-chip operation—at this word size, in-DRAM ECC can only provide bit-level correction (i.e., SEC-DED or SEC) without massive overfetching internal to the DRAM. Also, in-DRAM ECC cannot correct errors outside the chip, such as errors during transmission. In the context of a high-reliability system, end-to-end protection is needed or such external DRAM errors may dominate (Chapter 5). Meanwhile, current SDDC rank-level ECCs provide limited improvement in manufacturability by failing to tolerate distributed bit-level inherent faults across chips. Most importantly, providing high reliability in the presence of both operational faults and a high rate of inherent faults is challenging, so that future large-scale systems are expected to employ both in-DRAM ECC and rank-level ECC at the expense of up to 26.6% redundancy and overfetching.

4.3 ECC Guardband

High reliability in the presence of operational faults requires rank-level ECC. In other words, the correction capability of rank-level ECC operates as a *guardband* against operational faults. With increasing rates of inherent faults, however, some of the rank-level ECC correction capability can be used

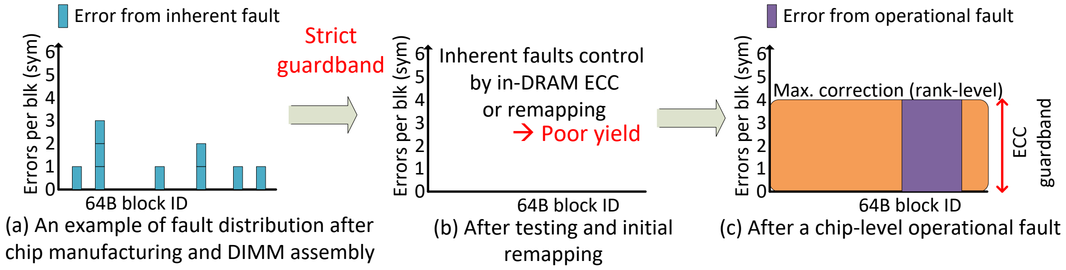
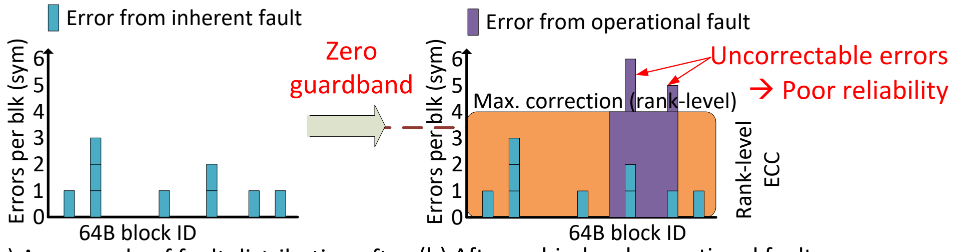


Figure 4.2: A strict ECC guardband example with QPC Bamboo ECC.

against inherent faults as well. This section presents three different uses of this correction capability.

4.3.1 Strict ECC Guardband

In the proposed in-DRAM + rank-level ECC schemes [60, 67, 126], all inherent faults are tolerated by in-DRAM ECC or remapping before the system is operational and the entire correction capability of the rank-level ECC serves as the *ECC guardband* against operational faults (Figure 4.2). Such mechanism is called *strict ECC guardband* because it can strictly tolerate the predetermined worst-case inherent (after remapping) and operational faults simultaneously. Figure 4.2 (a) shows an example distribution of inherent cell faults. Each bar represents the number of faulty cells within a particular memory block that is read in one memory transfer (e.g., a 64B block); blocks have up to 3 symbols with inherent faults in the example. After testing at manufacture time, all inherent cell faults are either removed by remapping or can be tolerated by the in-DRAM ECC and the rank-level ECC is used to correct operational faults only.



(a) An example of fault distribution after chip manufacturing and DIMM assembly

(b) After a chip-level operational fault

Figure 4.3: A zero ECC guardband example with QPC Bamboo ECC.

A strict ECC guardband provides high reliability at the expense of requiring in-DRAM ECC and/or significant remapping resources; this is because the in-DRAM ECC is weak given the constraints on redundancy and granularity within a DRAM chip. In fact, current in-DRAM ECC proposals are limited to correcting a single erroneous bit. Section 4.5.1 shows that to provide a strict ECC guardband under cell fault rates (CFRs) of 10^{-5} and 10^{-4}, more than 1,600 and 140,000 64B blocks, respectively, require remapping per 16GB ECC DIMM with in-DRAM ECC.

4.3.2 Zero ECC Guardband

The reason strict ECC guardband requires so much remapping capacity is the weak in-DRAM ECC. As an alternative, *zero ECC guardband* uses the rank-level ECC against both inherent and operational faults (Figure 4.3). A strong rank-level ECC can tolerate a large number of bit errors, and zero ECC guardband uses this strong correction capability to tolerate inherent faults without considering possible operational faults. It is a very economical scheme as just the rank-level ECC covers both inherent and operational faults, yet degrades reliability because it can no longer tolerate severe operational faults (e.g., a dead chip) after the rank-level ECC uses up some of its correction capability for inherent faults. Also, the manufacturability improvement of zero ECC guardband can be limited because strong rank-level ECCs are designed for faults that are expected to be confined to a single chip (i.e., SDDC), while inherent faults are distributed across all devices. In summary, zero ECC guardband can improve manufacturability at the cost of high reliability.

4.3.3 Active ECC Guardband

Active Guardband ECC (AG-ECC) efficiently utilizes rank-level ECC with 12.5% redundancy to tolerate overlapping inherent and operational faults. This satisfies both high reliability and high manufacturing yield requirements economically using reasonable remapping requirements and a standard level of redundancy. An AG-ECC follows the insight that in very rare cases, it is possible to allow high-overhead error correction as long as correction is cheap in the common case and that overall reliability is not compromised. Thus, memory blocks with existing faults only need to be remapped if even the expensive correction mechanism will not be able to tolerate the most severe error case targeted by the protection scheme. In other words, the ECC guardband must be maintained at a safe level, but that level is for exceptional cases rather than for common accesses. To avoid repeated high-overhead correction and maintain the safe ECC guardband, recent advancement on fine-grained memory remapping (Section 4.6) is actively and dynamically applied.

There are three mechanisms that enable AG-ECC to restore data from overlapping inherent and operational faults, whose errors can be beyond the conventional correction capability of a rank-level ECC.

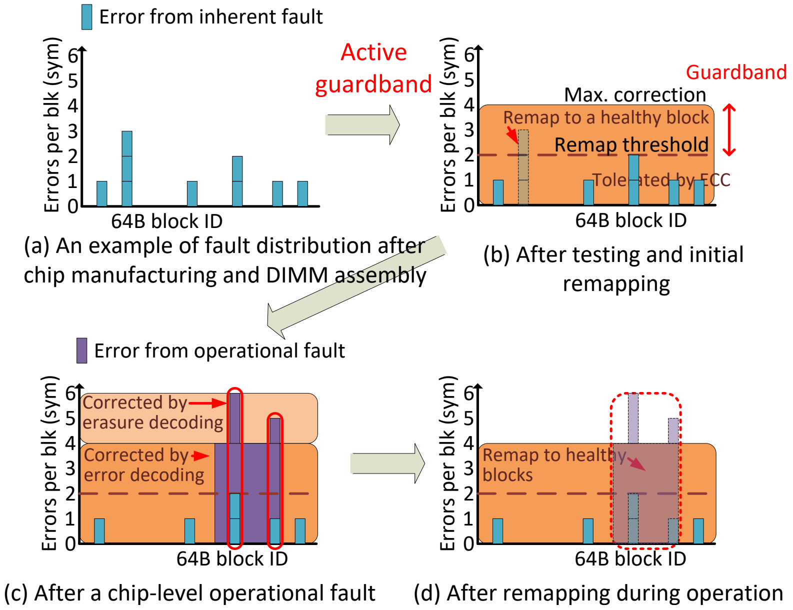
The first is to expand correction capability by using *erasure*-based correction. Erasure correction with known information of error locations doubles the number of corrections compared to random error correction (Section 2.3). For example, an ECC that can correct 4 random errors can replace 2 of its error correction capability with 4 erasure corrections, resulting in a total of 6 corrections (2 random errors + 4 erasures).

The second is to identify the location of an operational fault using *fault diagnosis*. With a DUE, AG-ECC starts a low overhead online diagnostic routine which samples blocks along the same and nearby rows and columns to the erring block to identify a block affected by the same operational fault but no inherent fault. Correcting such a block provides the device location of the operational fault, which can be used for erasure correction in the initial block. This diagnosis is efficient because inherent faults are rare and operational faults commonly affect multiple memory blocks. Even with a very high CFR of 10^{-5}, more than 99.5% of the blocks are expected to have no inherent fault. At the same time, severe operational faults typically affect several memory blocks (e.g., ≥90% of multi-pin faults affect a single DRAM row, single column, or a group of rows in [128]). Note that if the diagnostic procedures are unable to safely locate the operational fault, a DUE is reported. The same insight and a similar diagnosis procedure is also described by Nair et al. [100], though not evaluated to the same depth.

Lastly, AG-ECC uses to *remapping* to control the severity of faults and maintain the erasure-based SDDC guardband. The maximum number of corrections is still limited even with erasure decoding. In the previous example of 2-error and 4-erasure correcting codes, up to 6 symbol errors can be corrected in total. As a result, blocks with more than 2 existing errors will lose erasure-based correction capability if a future chip-level fault occurs. To prevent this reliability failure, AG-ECC preemptively retires blocks with more than 2 symbol errors after correction, so that all blocks are correctable by erasure-based decoding against a future chip-level fault.

Blocks with less severe faults (i.e., 1 or 2 symbol errors) are not remapped but continue to be used with cheap corrections; correction latency can be as low as 1 or 2 cycles per correction (Section 4.4.4). This selective remapping policy, which retires a faulty block only if it will be uncorrectable with a future chip-level fault and tolerates other faults, can efficiently and effectively reduce remapping overhead in the presence of frequent cell faults with minimal impact on reliability.

We call this scheme *active guardband* because it actively manage memory error protection in a way that is analogous to how voltage/timing guardbands are managed to maintain reliable circuit operation while improving performance and efficiency [86]. The following section presents a detailed example of AG-ECC using a strong rank-level ECC.



(a) An example of fault distribution after chip manufacturing and DIMM assembly

(b) After testing and initial remapping

(c) After a chip-level operational fault

(d) After remapping during operation

Figure 4.4: An active ECC guardband example with QPC Bamboo ECC.

4.4 Active Guardband QPC ECC

The goal of *Active Guardband Quadruple Pin Correcting (AG-QPC)* ECC is to provide SDDC capability in the presence of inherent scaling cell faults on standard 18-chip ECC DIMMs with ×4 DRAM devices. The multi-symbol correction and by far safer detection capabilities of QPC make it an excellent match for the AG-ECC approach. Following the AG-ECC flow discussed earlier, the threshold for remapping is set at two faults that affect different symbols within a single memory block.

The overall flow of AG-QPC is depicted in Figure 4.4. AG-QPC maintains fault severity below a remapping threshold (i.e., 2 symbol errors) by retiring blocks with more severe errors ((b) and (d)). Less severe faults are tolerated by the rank-level ECC. When a chip-level operational fault occurs on top of existing faults, 5-6 symbol errors are corrected via erasure decoding (c). More details about the decoding, fault diagnosis, and remapping schemes in AG-QPC are presented in the following subsections.

4.4.1 ECC decoding

AG-QPC uses the same encoding and organization of Bamboo QPC as that described in Chapter 3, but decoding is adapted for scaling faults. With operational faults, it is very unlikely for more than two chips to simultaneously manifest errors within the same memory block because operational faults are independent [48]. Therefore, when such rare corrections are reported, they can be pessimistically considered to result from a severe error miscorrection and ignored (i.e., the corrections are discarded and the block is reported as a DUE). This conservative correction reduces the likelihood that a severe error goes undetected and results in an SDC. Hence, the original QPC decoder does not attempt to correct some possibly-correctable errors and reports a DUE if the symbol corrections belong to neither a single chip nor two separate DQs.

With inherent faults, however, the high cell fault rate results in many memory blocks in which there are multi-chip faults, necessitating redesign of the post-processing heuristic to better balance correction and detection; if the same scheme were used, too many common correctable errors would be reported as uncorrectable. Table 4.2 compares the correction and detection capabilities of different post-processing schemes. QPC ECC decoding with n-

Rank-level ECC		QPC-2P	QPC-3P	QPC-4P
1 bit/pin/chip or 2×1 bit/pin faults (%)	DCE	100.0000000	100.0000000	100.0000000
3 × 1 bit/pin faults (%)	DCE	0.0000000	100.0000000	100.0000000
	DUE	100.0000000	0.0000000	0.0000000
4 × 1 bit/pin faults (%)	DCE	0.0000000	0.0000000	100.0000000
	DUE	100.0000000	100.0000000	0.0000000
1 bit fault + 1 chip fault (%)	DCE	0.0000262	0.0091167	1.5534977
	DUE	99.9999734	99.9908829	98.4286531
	SDC	0.0000004	0.0000004	0.0178492
2 × 1 bit faults + 1 chip fault (%)	DCE	0.0000000	0.0000162	0.0090661
	DUE	99.9999996	99.9999791	99.9680300
	SDC	0.0000004	0.0000047	0.0229039
Rank fault (%)	DCE	0.0000000	0.0000000	0.0000000
	DUE	99.9999994	99.9999954	99.9763927
	SDC	0.0000006	0.0000046	0.0236073

Table 4.2: A comparison of protection coverage using different QPC post-processing schemes.

pin correction (QPC-nP) accepts correction results if they belong to either a single chip or n separate DQs. QPC-2P is the same as the original QPC ECC, while QPC-4P has no post-processing. The result shows that QPC-3P has balanced correction and detection capabilities by correcting a chip error or up to 3 pin errors (e.g., 2 bit errors from inherent faults and 1 bit error from operational fault) and detecting more than 99.999995% of any errors, which makes it an excellent candidate to achieve both manufacturability (tolerating inherent cell scaling faults) and reliability.

4.4.2 Fault Diagnosis and Erasure Decoding

Because of the initial test and remapping step, there are at most two erroneous symbols from inherent cell faults in any block. Thus, errors from co-located inherent faults and a chip-level operational fault may result in errors in 5-6 symbols and exceed the maximum error correction capability of QPC-3P, which is limited to at most 4 symbols. In such cases, error decoding of QPC-3P will report a DUE, except in extremely rare cases ($< 5 \times 10^{-6}$%, Table 4.2). Before propagating an undetected error to the system, AG-ECC attempts fault diagnosis. If the location of an error is precisely known from the diagnosis, *erasure decoding* can correct it with only a single redundant symbol. Thus, if the faulting DQs are known, the redundancy of 8 symbols is split as up to 4 symbols for erasure correction (one per DQ) and the other (≥ 4) symbols for correcting the up to 2 random cell faults and for additional detection coverage.

4.4.3 Remapping

AG-ECC relies on a remapping mechanism for fault avoidance. The focus of this dissertation is not on the details of the remapping mechanism and any of the schemes described in Section 4.6 may be used. However, to provide a concrete example of a remapping mechanism, FreeFault [74] is considered. FreeFault dynamically remaps blocks from DRAM into blocks in a processor's large last-level cache (LLC) and even when a substantial number of blocks is remapped—up to several thousands of blocks—performance of the evaluated benchmarks is not significantly impacted. Section 4.5.1 evaluates the degree of remapping required and shows that remapping needs are reasonable and do not overly stress the remapping mechanism.

Another important aspect of utilizing remapping for active guardband management is that blocks exhibiting a severe combination of inherent and operational faults, and which therefore require higher-overhead erasure correction, are remapped after the first erasure procedure. Recall that blocks with up to two symbol errors do not need to be remapped because the ECC guardband is maintained. Once block-remapping capacity is saturated, it is also possible to map out an entire faulty device using one of Bamboo ECC's graceful-downgrade modes (Section 3.3), or replace the affected memory module.

4.4.4 Overheads

There are three sources of overhead for AG-QPC: the 12.5% redundant DRAM chips, the ECC circuits, and additional memory access latency. The redundancy level is standard and the circuits are not a significant concern as evaluated in Section 3.3.3. Latency overhead is discussed below.

The Bamboo ECC codes of AG-QPC requires the entire memory block to be transferred before decoding begins, which takes 4 DRAM bus cycles with burst length of 8. Once the memory block has been transferred, QPC ECC decoding proceeds: 2 cycles for syndrome generation and detection, an additional $\leq$ 2 cycles in rare cases of one symbol correction ($\sim$ 0.5% of accesses with a CFR of 10^{-5}), and another $\leq$ 2 cycles in the even rarer cases that 2 symbols require correction (based on [69, 24]). The erasure decoding procedure is estimated to take 12 cycles once the error locations are known. The diagnostics procedure requires reading a small number of blocks that manifest the same operational fault. While it may take hundreds of cycles, diagnosis occurs once per operational fault and erasure decoding is only performed once as well (followed by remapping); thus their overheads are negligible given that operational faults occur once per hundreds of years (or more) on average per DIMM.

In comparison, schemes that rely on in-DRAM ECC combined with rank-level ECC have high latency overheads as well, in addition to power overheads. First, in-DRAM ECC must internally decode the ECC requiring 2 cycles, followed by 1 cycle to transfer the information, and 2 cycles for the rank-level ECC. Thus, the read latency overhead of AG-QPC relative to these schemes is just a single cycle.

As explained in Section 4.2, every write operation on a ×4 DRAM with in-DRAM ECC requires a read-modify-write operation due to the smaller on-chip transfer granularity (32*b*) than the in-DRAM ECC data size (64b). While a write command is performing a read-modify-write operation, following read/write commands on the same bank cannot access the internal data bus and must be delayed. This increases DRAM write-to-write command delay (tCCD_L for write operations) and degrades performance. In-DRAM ECC also increases DRAM power consumption with extra cells, overfetching reads, and read-modify-write operations. This can significantly increase the DRAM energy consumption of applications with a fair amount of write traffic. Section 4.5.3 shows that AG-ECC with only rank-level ECC shows similar performance but significantly better energy efficiency than the combination of in-DRAM ECC and rank-level ECC.

4.5 Evaluation

This section measures the manufacturability, reliability, and performance impacts of AG-QPC and compares them with the state-of-the-art ECC schemes: DRAM with in-DRAM ECC + strict-ECC-guardband SDDC and XED-DDDC. XED [100] combines in-DRAM ECC and rank-level ECC to provide high reliability against inherent and operational faults, including chip failures (Section 4.6). It uses in-DRAM ECC to detect errors and rank-level ECC to correct the detected errors using erasure decoding in most cases. AG-QPC with 12.5% redundancy is directly compared to DDDC-level XED with 26.6% total storage overhead from in-DRAM ECC and rank-level ECC redundancy. The results demonstrate that AG-QPC provides stronger manufacturability than in-DRAM ECC and roughly matches the high reliability currently seen in high-performance systems despite the introduction of high inherent fault rates. Furthermore, it does so without any changes to current DRAM chips, modules, or interfaces while requiring reasonable fault remapping and retirement capabilities. AG-QPC also shows similar performance yet better energy efficiency than the in-DRAM + rank-level ECC based solutions.

4.5.1 Manufacturability

The manufacturability impact is evaluated by estimating DIMM yields and uncorrectable block counts with varying bit-level inherent fault rates (i.e., CFR). XED-DDDC can correct up to two device errors per memory block, yet using this full correction capability against inherent faults would sacrifice reliability against chip-level operational faults. Therefore, it is assumed that XED-DDDC tolerates 1-chip inherent faults (in addition to some bit faults correctable by in-DRAM ECC) but retires any block that exhibits faults in 2 chips. This selection provides a balanced design of high manufacturability and high reliability. Note that the manufacturability aspect is not discussed by Nair et al. [100].

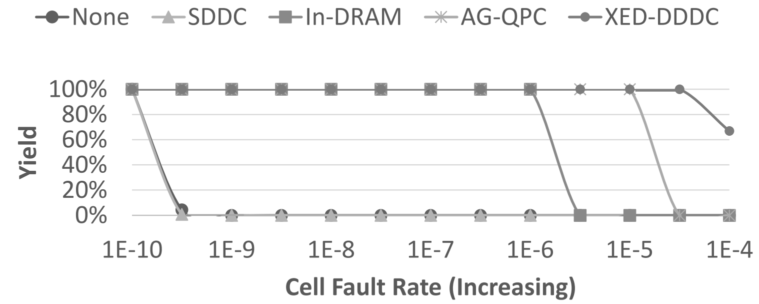


Figure 4.5: A comparison of $16GB$ DIMM yields using different ECC schemes and varying cell fault rates.

DIMM yield: Figure 4.5 shows $16GB$ DIMM functional yield of the different error control schemes with varying CFRs, assuming that up to 32 $64B$ blocks in a DIMM can be remapped at manufacture time (e.g., through spare rows and columns). In-DRAM ECC and XED remap memory at $128B$ granularity due to internal overfetching, while the others use $64B$ granularity. The manufacturability of an ECC scheme depends on its correction capability against inherent faults. With no ECC or standard SDDC rank-level ECC, DIMM yield quickly drops to 0% around 10^{-10} CFR, explaining why DRAM vendors have sought manufacturability solutions involving in-DRAM ECC. In-DRAM ECC corrects up to one inherent bit error per chip; it maintains almost 100% yield until a CFR of 10^{-7}, and it can achieve high yield with some block sparing until a CFR of 10^{-6}. Above a CFR of 10^{-6}, however, the rate of two-bit errors in the same chip becomes problematic and the yield of in-DRAM ECC quickly drops to 0%.

The fact that DRAM vendors employ In-DRAM ECC for the manufacturing of fault-free non-ECC DIMMs indicates that the expected inherent error

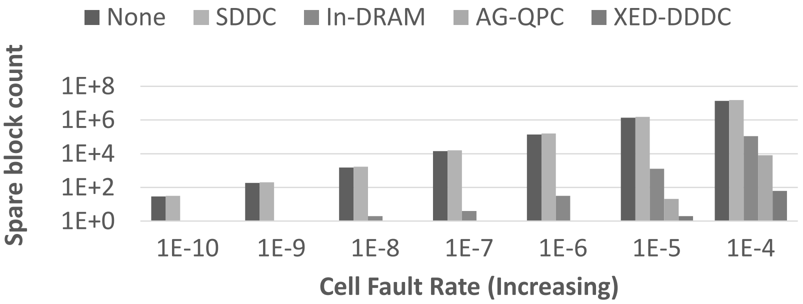


Figure 4.6: A comparison of the number of spare 64B blocks needed to achieve 99.99% yield on 16GB DIMMs.

rate is at most a CFR of 10^{-6}. AG-QPC easily handles an inherent fault rate 10× higher than this, despite requiring no DRAM changes and only 12.5% total redundancy. XED-DDDC extends this protection even further, at the cost of the increased redundancy and complexity of In-DRAM ECC. It is not clear that this increased manufacturability offers any substantive advantages unless an extreme CFR of $> 10^{-4}$ is expected.

Needed spare block count: As yield with a fixed retirement space often saturates to either 0 or 100%, the number of spare blocks needed to replace uncorrectable blocks is also measured as a metric for the cost needed to ensure a given yield (Figure 4.6). No ECC and SDDC rank-level ECC require 30 spare blocks to achieve 99.99% yield at a CFR of 10^{-10}, but the retirement needs rise sharply to 13 million blocks at 10^{-4}. These numbers will increase with process and capacity scaling, such that neither technique is viably manufacturable. Similarly, in-DRAM + rank-level ECC requires 38 spare blocks at a CFR of 10^{-6}, but the number explodes to 140,000 at 10^{-4}, requiring a total of 17MB

remapping capacity (assuming $128B$ block retirement due to in-DRAM ECC overfetching) to deal with inherent faults. Despite only requiring 12.5% redundancy, AG-QPC can achieve 99.99% yield at 10^{-5} and 10^{-4} CFRs with just 21 and 8,000 spare blocks, respectively, or 1KB and 500KB, respectively, of 64B retirement space; these remapping overheads are easily within reach of FreeFault, frame retirement, and other remapping schemes. XED-DDDC uses its double chip correction capability to minimize remapping and as such it requires just 62 block remappings at 10^{-4}; its overall 26.6% redundancy demands a large amount of extra storage for ECC, however.

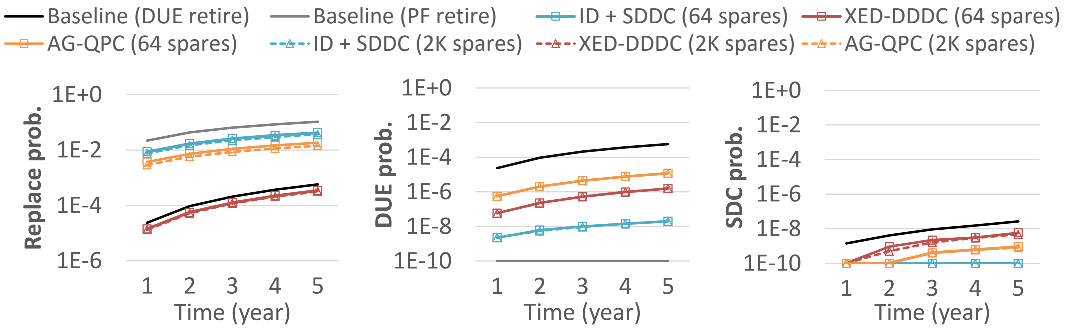
4.5.2 System Reliability

The reliability of the different ECC schemes is estimated by the DUE and SDC failure probabilities of a DRAM channel using the two-stage Monte Carlo simulation described in Section 3.4. The first Monte Carlo simulation stage randomly injects faults into a simulated DRAM channel based on the observed fault modes and rates of 2GB DDR2 DIMMs [128]. In the evaluation, however, the DRAM bank count is doubled and the row count is quadrupled to accommodate a 16GB DIMM capacity. Once (possibly overlapping) faults are identified by the first stage, the second stage randomly generates errors based on the underlying fault models, assuming each bit within a corrupted region has a 50% flipping probability (but the error-free pattern is excluded). Also, random bit errors are injected into each block based on the given inherent cell fault rate. A protection scheme decodes the error patterns to determine whether the errors are detectable, correctable, or miscorrected. Undetected errors and miscorrected errors are categorized as SDC. A corrected error may be retired, if its severity exceeds the retirement threshold set by an ECC scheme, before resuming the simulation run. If the number of remapped blocks exceeds the capacity of the ECC scheme, the DIMM is marked as bad and replaced. Periodic memory scrubbing with an 8-hour interval is used to prevent the accumulation of transient faults.

Remapping can be crucial for reliability because it reduces the accumulation of permanent faults and avoids the need for frequent DIMM replacement. It is assumed that 64 $64B$ blocks can be remapped by coarse-grained remapping schemes (e.g., 32 remappings per rank with spare rows and columns) and that several thousand blocks can be remapped by fine-grained remapping schemes (e.g., FreeFault). Furthermore, it is assumed that the same remapping scheme and storage space is shared between uncorrectable inherent faults and operational faults. At a CFR of 10^{-6}, for example, 99.99% of 16GB DIMMs with in-DRAM ECC and SDDC rank-level ECC require less than 38 remapped blocks, leaving the rest of the remapping capacity for operational faults in the field. For reliability analyses at higher CFRs, the remapping capacity can be increased with the help from fine-grained remapping schemes.

A large-granularity operational fault commonly corrupts only a small fraction of blocks within its affected region. Instead of pessimistically retiring all of the blocks within the fault range, distributions on the number of affected blocks per fault mode is extracted from the same paper as the fault rates [128] to determine the necessary amount of remapping. The number of experimental runs is 10 billion per scenario (CFR and scheme).

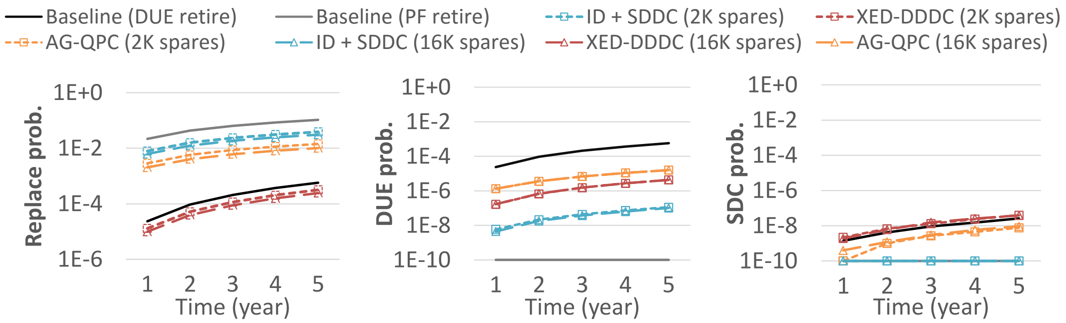
To compare reliability with current state-of-the-practice schemes, AMD chipkill against operational faults only is set as the baseline to represent the level of reliability in current high-reliability systems. There are two variants of AMD chipkill with different remapping policies: one with DUE retirement (which replaces a DIMM when a DUE is reported) and another with *permanent fault (PF) retirement* that replaces a DIMM after repeated corrections from a permanent fault are observed. DUE retirement is more efficient and requires 1/180× fewer DIMM replacements than PF retirement, but is less reliable because it generates DUEs and SDCs in the presence of accumulated faults (no DUE nor SDC is reported during 10B runs of PF retirement).



(a) DIMM replacement prob. (b) DUE probability. (c) SDC probability.

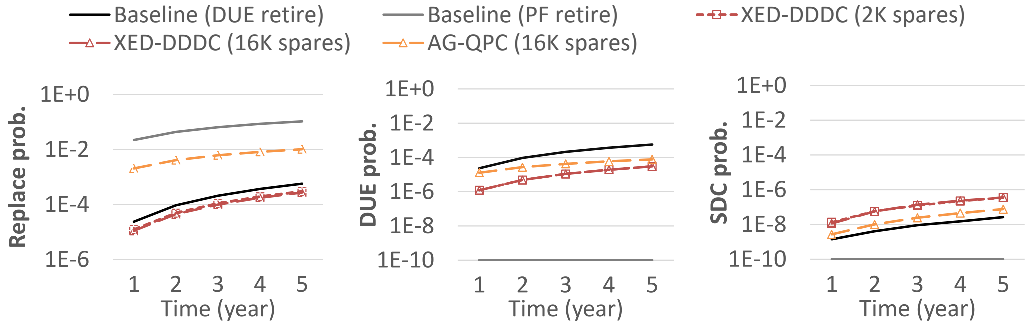
Figure 4.7: The reliability of a 2-rank DRAM channel against both inherent and operational faults (CFR = 10^{-6}).

To analyze the reliability of systems with both inherent and operational faults, 3 ECC schemes are evaluated: In-DRAM + AMD chipkill, XED-DDDC, and AG-QPC. Other protection schemes that fail to provide a rea-



(a) DIMM replacement prob. (b) DUE probability. (c) SDC probability.

Figure 4.8: The reliability of a 2-rank DRAM channel against both inherent and operational faults (CFR = 10^{-5}).



(a) DIMM replacement prob. (b) DUE probability. (c) SDC probability.

Figure 4.9: The reliability of a 2-rank DRAM channel against both inherent and operational faults (CFR = 10^{-4}).

sonable DIMM yield at a CFR of 10^{-6} are excluded. Figures 4.7, 4.8, and 4.9 show the failure probabilities of a 2-rank DRAM channel due to combined operational and inherent faults with a CFR of 10^{-6}, 10^{-5}, and 10^{-4}, respectively. The number of spare blocks ranges from 64 to $16K$, which represents coarse-grained retirement (e.g., 32 spare rows per rank) and fine-grained retirement (e.g., Freefault), respectively. At 10^{-5} CFR, In-DRAM + SDDC (2K/16K spares) and AG-QPC (2K spares) are excluded because they show 0% functional yield.

Several trends are notable. In-DRAM ECC with SDDC (26.6% redundancy) provides strong protection against both DUEs and SDCs, at the cost of frequent DIMM replacement. XED-DDDC (26.6% redundancy) reduces the DIMM replacement frequency by orders of magnitude, but suffers from an increased risk of silent data corruption. AG-QPC (12.5% redundancy) offers a compelling tradeoff between replacement probability and SDC rate, despite requiring no DRAM changes and using only half the redundancy of the other schemes. AG-QPC requires roughly 60% less frequent DIMM replacement than in-DRAM ECC with SDDC—only 1.0% – 1.8% of AG-QPC DIMMs need to be replaced within 5 years, depending on the inherent fault rate and remapping capacity. In addition, AG-QPC provides $4-7\times$ stronger protection against SDCs than XED-DDDC. All of the considered techniques make frugal use of the available retirement storage, and for the most part increasing the maximum number of spares does not make a meaningful reliability difference, except in the DIMM replacement probability. A larger remapping capacity can lower the DIMM replacement frequency (by up to 42% in our evaluation) by tolerating a larger fraction of operational faults.

4.5.3 Performance and Energy

In addition to maintaining high manufacturability and reliability, AG-QPC offers significant energy benefits over in-DRAM + rank-level ECC based solutions. This overall impact is quantified using the Gem5 simulator [1] with

Core	3.4GHz OoO core × 4 (issue width=8, ROB=192 entries) 32KB L1D, 1MB L2, 8MB L3 (64B lines) w/ stride prefetcher					
DRAM timing (cycle)	×4 16GB DIMM (DDR4-2400)					
		CAS latency	tCCD_L(WR)	tFAW		
	No in-DRAM	16	6	16		
	In-DRAM	18	24	26		
		RD data FIFO lat.	WR data FIFO lat.			
	AMD CK	+3	+2			
	QPC-3P	+6	+2			
DRAM current (mA)		IDD0	IDD3N	IDD4R	IDD4W	IDD5
	No in-DRAM	43	38	110	103	250
	In-DRAM	46.8	44.3	141.3	184.3	254.1

Table 4.3: The performance/energy simulation parameters of AG-ECC.

DRAMSim2 [119] running homogeneous multi-programmed SPEC CPU 2006 workload mixes with one program per core [130]. Table 4.3 shows the machine configuration, latency penalties, and DRAM power consumption used in this evaluation. These parameters are based on the Intel E7-4470 CPU (Haswell) and Micron DDR4-2400 memory [96]. Due to a lack of publicly available information, in-DRAM ECC power consumption is estimated based on differences between ×4 and ×8 configurations of the same DRAM chip. For example, ×4 and ×8 configurations activate $4Kb$ and $8Kb$ row buffers, and consume IDD0s of $43mA$ and $46mA$, respectively.[1] In-DRAM ECC requires a $9Kb$ row buffer, and accordingly it is estimated to consume $46.8mA$. The read overhead of the internal in-DRAM ECC read-modify-write operation is modeled as

[1] IDD0, IDD3N, IDD4R/IDD4W, and IDD5 are the DRAM activation current, active standby current, operating burst read/write current, and burst refresh current, respectively (see [60] for details.)

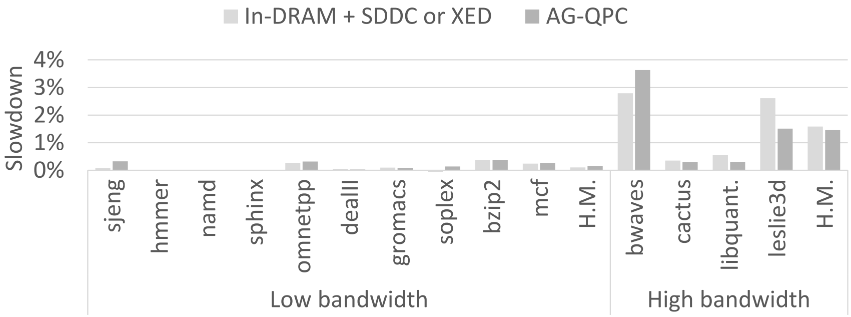


Figure 4.10: A comparison of the IPC slowdown comparison between AG-QPC and in-DRAM-ECC-based schemes.

a {IDD4R$_{\times 8}$ - IDD4R$_{\times 4}$}$\times\frac{9}{4}$ increase in IDD4W. The total power is calculated based on Micron DDR3 power model [95] and divided by the average IPC to calculate energy efficiency.

Figure 4.10 shows a performance comparison between AG-QPC and in-DRAM ECC based schemes. The figure presents the IPC slowdowns of the mechanisms against a system without ECC (neither in-DRAM nor rank-level). The increased tCCD_L$_{WR}$ (write-to-write delay) from read-modify-write operations and the increased CAS latency in in-DRAM ECC degrade the performance of memory-intensive applications by up to 3%. By eliminating in-DRAM ECC, AG-QPC not only improves storage efficiency but it also maintains similar performance to ECC-free operation in most applications. The memory-intensive benchmark bwaves shows anomalously poor performance with AG-QPC during the chosen program interval due to some memory scheduling conflict caused by the QPC decoding delay. This causes one processor core to starve relative to the others, slowing the overall performance; if this anomalous core statistics are discarded, the performance impact

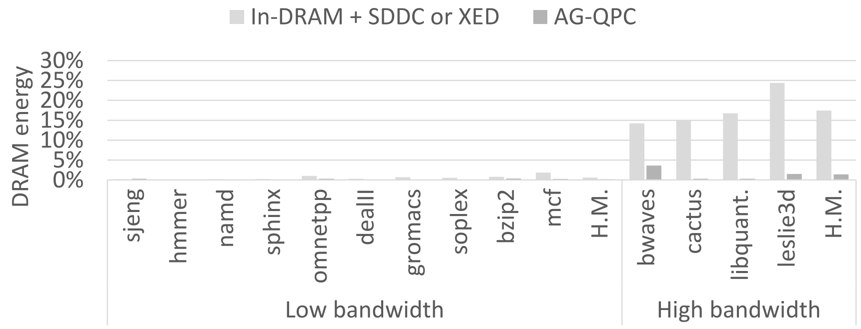


Figure 4.11: A comparison of the DRAM energy efficiency between AG-QPC and in-DRAM-ECC-based schemes.

is roughly 2% and it is less than that of in-DRAM + rank-level ECC.

Figure 4.11 shows that AG-QPC significantly improves energy efficiency relative to in-DRAM + rank-level ECC. For bandwidth intensive applications, in-DRAM + rank-level ECC consumes 18% more DRAM energy on average (and up to 25%) than an ECC-free system. AG-QPC shows a more modest 0.5% average degradation in energy efficiency, akin to the normal expectations for an ECC-protected system. The benchmark bwaves suffers from 3.6% energy degradation using AG-QPC due to its aforementioned 3.6% slowdown. Compared to in-DRAM + rank-level ECC solutions with similar reliability, AG-QPC offers more than 16% energy savings (up to 24%) in memory intensive applications.

4.6 Related Work

Traditionally, inherent DRAM faults have been controlled by fault-removal mechanisms (e.g., spare rows and columns), while operational faults have been tolerated via rank-level ECC. More recently, rank-level ECC has been employed to tolerate inherent faults caused by variability in retention time and to aggressively trim retention margins as well [72, 115, 71, 88]. These efforts use error coding against inherent faults, similar to this work, but they do not consider the operational reliability of their endeavors nor do they target SDDC level of reliability.

It has long been known that remapping can have a synergistic effect with ECC for tolerating persistent operational faults [131, 133]; a finding that has led to DRAM vendors providing *post package repair* in recent memories [60, 67]. Platform developers (such as NVIDIA [138]) retire memory frames and rely on non-volatile storage to maintain the mapping. Academia has proposed fine-grained remapping to reduce remapping overhead: ArchShield [99] and FreeFault [74] with the cache line granularity, RelaxFault [75] with subcacheline granularity, and CiDRA [126] with bit granularity. AG-ECC is remapping-mechanism agnostic and could work alongside any of these approaches.

CiDRA [126] has some relation to AG-ECC in that it considers the use of a fine-grained fault remapping cache alongside in-DRAM ECC and rank-level SEC-DED. While CiDRA does concern itself with maintaining SEC-DED levels of operational protection in the presence of inherent faults, it uses both in-DRAM ECC and rank-level ECC (26.6% redundancy) and does not consider SDDC levels of reliability. Also, the inherent fault model used by CiDRA is stuck-at permanent failures and it relies on mask error correction which does not apply to retention faults. ECC-Asprin [73] is similar to CiDRA but it uses rank-level ECC (12.5% redundancy) instead of in-DRAM ECC. Again, it differs from AG-ECC because it maintains only SEC-DED levels of protection and it relies on mask error correction for its fault diagnoses.

Several non-DRAM memory papers have observed that ECC and remapping can be combined to improve yield in a cost-effective manner [148, 91]. These approaches are similar in concept to AG-ECC, but they are not evaluated in the context of DRAM and only incorporate reliability to demonstrate that the effectiveness of bit-level ECC is not significantly compromised using a simple stuck-at random bit error model. HI-ECC [145] and FREE-p [154] adopt a similar approach for caches and non-volatile memories, respectively. AG-ECC differs in many ways, not the least of which is the target of inherent scaling faults combined with operational faults in DRAM.

VS-ECC [11] allows more aggressive voltage scaling by protecting weaker SRAM cells with stronger protection. E-ECC [24] combines error and erasure correction to provide SDDC-level protection on ×8 chips using one redundant chip. If a chip error corrupts multiple ECC words within an access, it utilizes the correction information of one word to apply erasure decoding on the remaining words. XED [100] is closely related work that combines error and erasure correction to provide high reliability against inherent and operational faults. XED uses in-DRAM ECC to detect errors and rank-level ECC to correct the detected errors using erasure decoding in most cases. In some cases, multiple chips may report errors from in-DRAM ECC, exceeding the erasure correction capability. When that happens, XED first attempts to use the single-error correction capability of its in-DRAM ECC to correct the errors. If that does not succeed, low-overhead online diagnostics are used to provide additional information on fault location and erasure decoding is attempted once more. On standard ECC DIMM with ×4 chips, XED can correct up to 2 chip errors using the 2 extra chips (and some 1-bit-per-chip errors using in-DRAM ECC). Despite this strength, its reliance on in-DRAM ECC increases the redundancy to 26.6% and degrades performance with overfetching.

4.7 Summary

While manufacturability and reliability are top priorities for DRAM vendors and large-scale system designers, respectively, separate in-DRAM ECC and rank-level ECC mechanisms incur storage overheads as high as 26.6% and they can degrade performance with overfetching. This chapter presents a novel and systematic management of rank-level ECC, called AG-ECC, to tolerate both inherent and operational faults. Rigorous evaluation shows that AG-QPC satisfies both the high reliability and high manufacturability requirements without modifying the conventional DRAM chips, modules, or interfaces. Accordingly, AG-ECC may enable the economical scaling of standard DRAM technology to meet future capacity demands without sacrificing yield, reliability, or efficiency.

Chapter 5

Mechanisms against Command/Address Errors

Any thorough system-level protection scheme must be holistic and provide end-to-end protection. Strong protection of any one component provides limited benefit to the overall reliability, as any unprotected component will quickly become the reliability bottleneck. While the strong SDDC mechanisms described in the previous chapters can reduce uncorrectable data error rates down to 1 FIT/chip [93], *Clock, Control, Command, and Address (CCCA)* signal transfers are poorly protected or left unprotected. Transmission errors on these signals are already as frequent as uncorrectable data errors with current DRAM [127]. Transmission errors are likely to become more frequent than uncorrectable data errors as transfer rates increase and I/O voltage levels decrease. Once manifested, these errors can result in serious data errors over a large memory region which are not detectable by data-only protection. Therefore, future systems will suffer from degraded overall memory reliability unless strong protection against this important emerging error type is provided.

Parts of this chapter appear in [79]. The author is the main contributor of the idea, implementation, and evaluation.

This chapter presents *All-Inclusive ECC (AI-ECC)*, a holistic memory protection scheme that is able to safeguard both DRAM data and CCCA signals against storage and transmission errors. AI-ECC augments existing data-protection schemes to provide very safe detection of CCCA errors without additional redundant storage or new signals and without degrading the effective level of data protection. It is meant to be an unobtrusive addition to future memory standards. AI-ECC as an extension to DDR4 memory is described, and is demonstrated to provide complete CCCA and data error protection; DDR4 is the most dominant main memory system design today for high-capacity servers and HPC systems, and is therefore used as a concrete example.

5.1 DRAM Transmission Errors

Increasing transfer rates and decreasing I/O voltage levels make DRAM transfers more vulnerable to transmission errors (Section 1.3). Attributes of transmission errors make them easier to correct than storage errors but more critical to detect and diagnose. An erroneous transmission can typically be corrected by retry, simplifying the correction of any detected transmission error. However, transmission errors are difficult to reproduce because most of them are intermittent, making identification of the error in the first place important for repairs.

For data signals, SDDC ECC can provide strong correction and safe detection of transmission errors and correction information can identify the

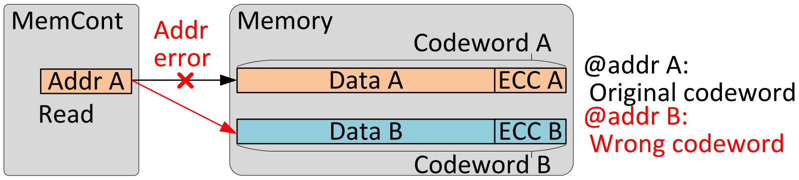


Figure 5.1: A read address error example.

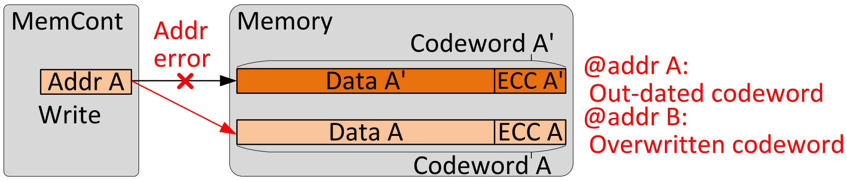


Figure 5.2: A write address error example.

faulty data lines. Recent memories also introduced *Cyclic Redundancy Check (CRC)* to detect write data transmission errors prior to updating storage data [60, 61, 66]. These schemes provide very robust protection against data transmission errors, enabling aggressive speed-ups of data transfer.

CCCA signals are more difficult to protect due to the fine-grained nature of the signals. Compared to data, which has a wide interface (e.g., 64-bit channel) and lengthy bursts (e.g., burst length of 8), CCCA has a narrower interface (e.g., 27 CCCA signals) and cycle-by-cycle transfers, which make amortization of redundancy overhead difficult and strong error coding expensive. As will be shown later, the current JEDEC response of adding one bit parity to the command and address signals is insufficient for many CCCA errors. Furthermore, even the most aggressive known mechanisms from industrial patents for dealing with CCCA errors only provide partial protection.

If undetected, CCCA transmission errors can result in severe failures;

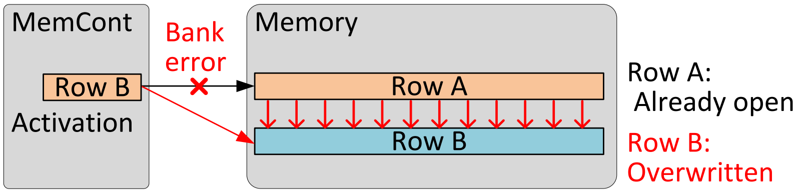


Figure 5.3: A command error example (duplicate activations).

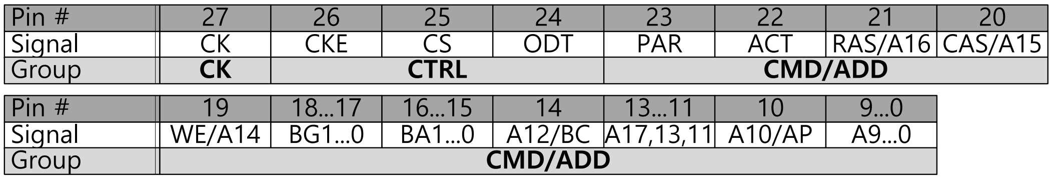
some escape existing protection and can compromise the final output (i.e., SDCs), while others cannot be corrected by retries and require expensive checkpoint-and-restart [38] to restore the data. Figures 5.1 and 5.2 give some examples of address transmission errors and their associated consequences. Figure 5.1 shows a transmission error that changes the address of a read operation. Despite reading the wrong location, the data-only ECC codeword ({data B, ECC B}) is valid and the error causes an SDC. An error in a write address (Figure 5.2) poses an even more serious risk—not only is the wrong location (address B) updated with incorrect data, but also the data in the originally intended destination (address A) becomes stale and incorrect. Both locations have their storage data corrupted (*memory data corruption (MDC)*) yet each location still holds a valid ECC codeword such that a following read will escape data-only ECC and result in SDC.

Transmission errors in the clock, control, and command signals can be catastrophic as well. Figure 5.3 shows a CCCA error that generates duplicate activations (for row A and B) on the same bank. In that case, the memory bit-lines are already activated with row A data and mistakenly opening a word-line copies this data into row B, destroying it and causing significant MDC.

Accordingly, a later read to row B will yield valid codewords yet incorrect data, resulting in SDC.

Rates of CCCA errors heavily depend on I/O and board designs and are proprietary information, yet are likely growing rapidly with increasing transfer rates. A large-scale field analysis on DDR3 DIMM shows that the number of reported command/address errors by an even-parity detection mechanism is as high as 72% of all uncorrectable data errors encountered using SDDC ECC [127]. With a conservative assumption that circuit techniques will ensure a fixed *Bit Error Ratio (BER)*, DDR4 with doubled command bandwidth will suffer from more raw CCCA errors than uncorrectable data errors. With a more realistic assumption that increasing transfer rates will increase BER, weak protection (e.g., DDR4 command/address parity bit) will not be enough and undetected CCCA errors will outpace uncorrectable data errors. Section 5.4 shows that CCCA BER should be improved to one millionth of the current JEDEC DDR4 standard (10^{-16} BER for data signals) [60], to make undetectable CCCA error rate by the parity bit comparable to uncorrectable data error rates by SDDC ECCs.

AI-ECC detects and diagnoses transmission errors as they occur, preventing a high BER from translating into silent data corruption or system failures. This allows the continuing use of commodity DRAMs (which are not necessarily optimized for high reliability) in business or safety-critical systems, even in the presence of increasing transmission error rates. Additionally, the strong protection of AI-ECC may expose tradeoffs between CCCA error rates



Pin #	27	26	25	24	23	22	21	20
Signal	CK	CKE	CS	ODT	PAR	ACT	RAS/A16	CAS/A15
Group	**CK**	**CTRL**			**CMD/ADD**			

Pin #	19	18...17	16...15	14	13...11	10	9...0
Signal	WE/A14	BG1...0	BA1...0	A12/BC	A17,13,11	A10/AP	A9...0
Group	**CMD/ADD**						

Figure 5.4: The CCCA signal interface for DDR4 memory.

and other design choices, such as increasing operating frequency.

5.2 Current DRAM Practices

This section introduces the latest DDR4, which will be used as the underlying DRAM for AI-ECC and future standards, with focuses on its CCCA signals and reliability features.

5.2.1 DDR4 CCCA Signals

DDR4 DRAM commands use 28 non-data pins to issue and control 4 types of signals: *clock ($\overline{CK}$), control ($\overline{CTRL}$), command ($\overline{CMD}$), and address ($\overline{ADD}$)*, as shown in Figure 5.4 [60]. Note that signal identifiers are $\overline{\text{overlined}}$ to differentiate them from DRAM commands. Also note that there is both an activate command, ACT, and an activate signal, $\overline{\text{ACT}}$.

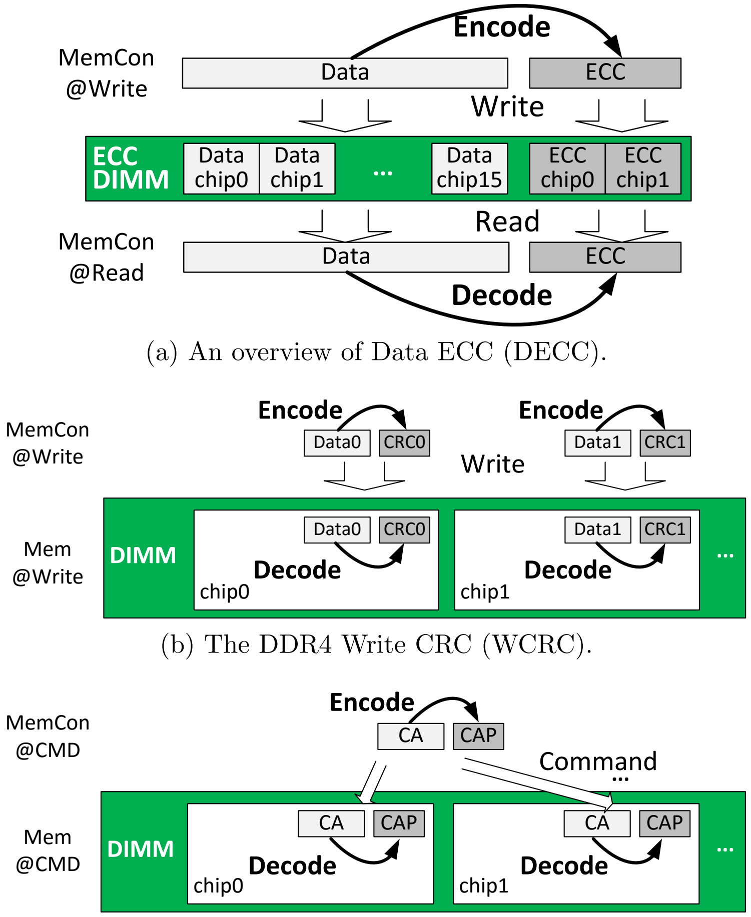
There are 3 signals in the $\overline{\text{CTRL}}$ group; $\overline{\text{CKE}}$, $\overline{\text{CS}}$, and $\overline{\text{ODT}}$. $\overline{\text{CKE}}$ (clock enable) saves energy by putting an idle rank into a power-save mode. $\overline{\text{CS}}$ (chip select) selects a rank in multi-rank configurations. $\overline{\text{ODT}}$ (on die termination) controls dynamic on-die termination of DRAM to improve transmission quality for write data.

The $\overline{\mathrm{CMD}}$ and $\overline{\mathrm{ADD}}$ signals time-multiplex physical pins, called CA (command and address). For non-ACT commands, pin 21/20/19 is used to indicate the $\overline{\mathrm{RAS}}$ (row address strobe)/$\overline{\mathrm{CAS}}$ (column address strobe)/$\overline{\mathrm{WE}}$ (write enable) signal, respectively. For ACT commands, which require the most address signals (up to 18 bits), DDR4 uses 21-19 pins to indicate address signals, while a separate $\overline{\mathrm{ACT}}$ pin indicates the command. A separate $\overline{\mathrm{PAR}}$ (command/address parity) signal transfers an even parity of the CA signals to protect against transmission errors, as will be explained in more details in Section 5.2.2. $\overline{\mathrm{BG}}$, $\overline{\mathrm{BA}}$, $\overline{\mathrm{BC}}$, and $\overline{\mathrm{AP}}$ stand for bank group, bank address, burst-chop, and auto-precharge, respectively. Transmission errors over any of these CCCA signals can have a disastrous impact that is not correctable by conventional data-only ECC.

5.2.2 DDR4 Reliability Features

Large-scale systems typically employ *Data ECC (DECC)* to protect memory data against storage and transmission errors (Figure 5.5a). On a read, DECC fetches both data and its ECC check bits, detecting and correcting inconsistencies due to errors. A data transmission error on a memory write, however, remains latent as MDC until a following read to the locations with erroneous data. Such MDC is problematic, as it is likely to cause severe data loss that is not correctable through ECC.

Accordingly, DDR4 introduces a *write CRC (WCRC)* for the early detection of write data transmission errors to reduce the chance of memory



(a) An overview of Data ECC (DECC).

(b) The DDR4 Write CRC (WCRC).

(c) The DDR4 CA Parity (CAP).

Figure 5.5: The reliability features available with DDR4.

data corruption (Figure 5.5b). WCRC generates an 8-bit CRC checksum of the write data to each chip and transmits this CRC over 2 additional beats that trail the standard 8-beat data transfer. Each DDR4 DRAM chip checks the consistency of this checksum with the received data before writing its memory array. On detecting an error, memory uses a separate signal ($\overline{\text{ALERT}}$) to let the memory controller to know and re-transmit.

DDR4 introduces two other weak ad-hoc mechanisms for dealing with CCCA errors: *CA parity (CAP)* and *Gear down* mode. CA parity (Figure 5.5c) uses a dedicated pin ($\overline{\text{PAR}}$) to transfer the even parity of the $\overline{\text{CMD}}/\overline{\text{ADD}}$ signals. Upon receiving a command, each DRAM chip computes the parity of its received $\overline{\text{CMD}}/\overline{\text{ADD}}$ and checks it against the received $\overline{\text{PAR}}$. CAP is a weak level of CCCA protection, as it does not cover the $\overline{\text{CK}}$ and $\overline{\text{CTRL}}$ groups and cannot detect an even number of bit-errors on the $\overline{\text{CMD}}/\overline{\text{ADD}}$ signals. Gear down mode halves the CCCA transfer rate to trade off latency and command bandwidth for signal quality, keeping the data transmission rate and the data bandwidth the same. While gear down mode reduces the CCCA error rate, it is not a viable solution for memory-intensive workloads that do not exhibit very high locality.

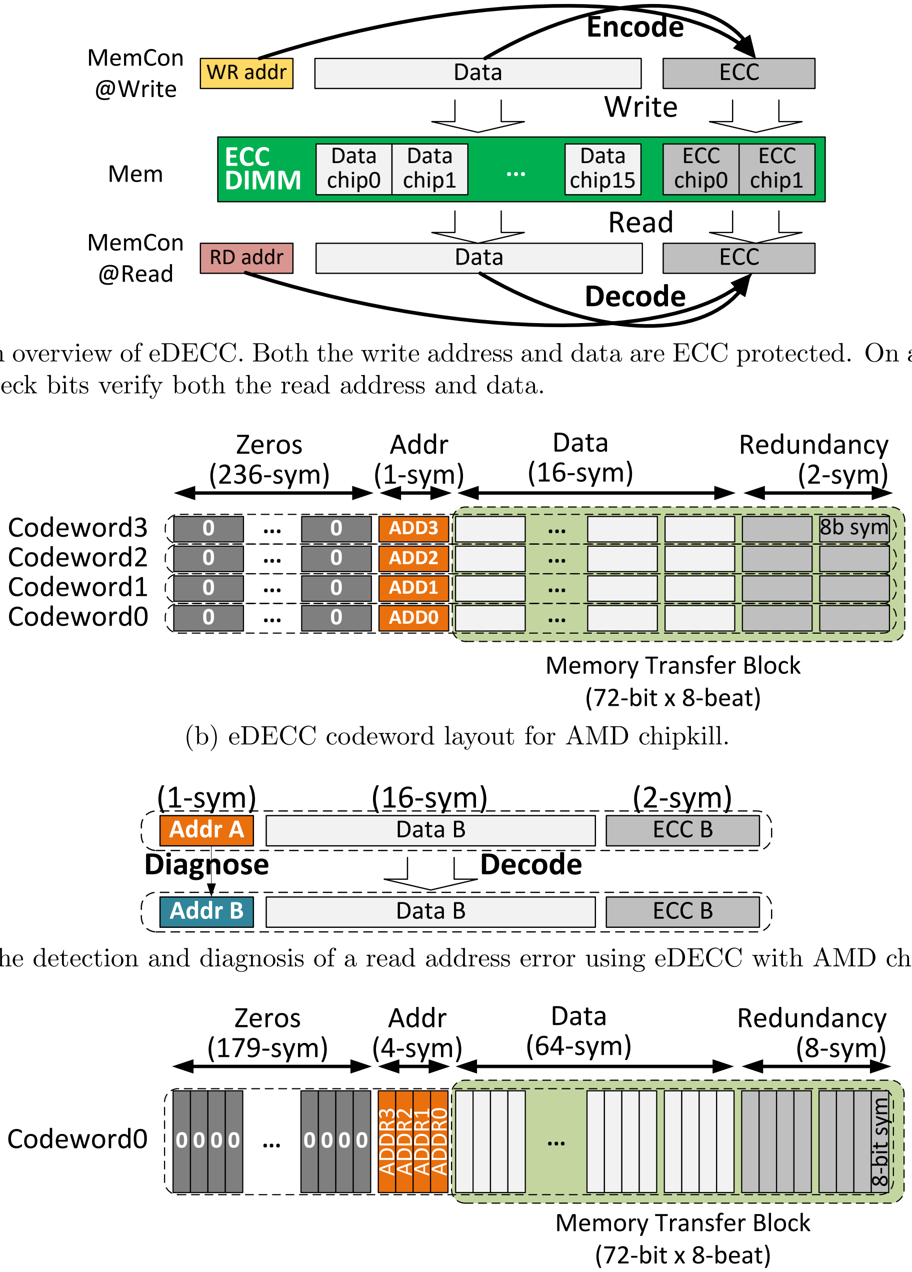
5.3 All-Inclusive ECC

AI-ECC provides strong and thorough protection by combining 4 complementary techniques. *Extended data ECC (eDECC)* and *extended write CRC (eWCRC)* protect memory against address errors by exploiting currently unused faculties of the data ECC and write CRC. They are able to strongly protect both the data and address simultaneously with no extra storage and transfer overheads. An architectural mechanism called the *Command State and Timing Checker (CSTC)* uses memory protocol information to detect illegal command sequences and protect against errors in the $\overline{\text{CK}}$, $\overline{\text{CTRL}}$, and $\overline{\text{CMD}}$ signals. Finally, *extended CA Parity (eCAP)* strengthens this command error protection, filling remaining coverage holes.

5.3.1 Extended Data ECC for Address Protection

Extended data ECC (eDECC) augments SDDC data ECC to protect address information without extra storage/transfer overheads (Figure 5.6). A ×4 DRAM transfers 32 bits of data per access; eDECC leverages the strength of SDDC ECC to detect and precisely diagnose up to 32 bits of address information. The 32-bit address used by AI-ECC is an MTB address that includes the rank, bank, row, and (partial) column address for the given 64B block of physical memory. By using the MTB address, AI-ECC can protect up to 256GB per channel (compared to 4GB if it uses the full byte address); a capacity of 256GB per channel is larger than any published DRAM standard and it should be sufficient for even the highest-capacity servers.

Commonly used error coding can protect longer codewords than those that are called for by conventional memory access granularities (i.e., *shortened codes*, Section 2.3). These underlying capabilities are used by eDECC to embed DRAM address information without additional redundancy, similar to their



(a) An overview of eDECC. Both the write address and data are ECC protected. On a read, the check bits verify both the read address and data.

(b) eDECC codeword layout for AMD chipkill.

(c) The detection and diagnosis of a read address error using eDECC with AMD chipkill.

(d) eDECC codeword layout for Bamboo QPC ECC.

Figure 5.6: A visualization of the extended Data ECC (eDECC) mechanisms.

use for embedding other types of metadata [34, 35]. Figure 5.6b shows eDECC with AMD chipkill [9]. The AMD chipkill data ECC codeword has 16 data symbols and 2 check symbols, though its 8-bit RS codes could potentially protect 237 more symbols without either extra redundancy or compromising correction capability. AI-ECC adds one extra write address symbol to the eDECC encoding using this auxiliary protection. On a read, the returned data and redundancy are decoded together with the read address (Figure 5.6a). If a read address error fetches data and redundancy from a wrong address (address B instead of address A in Figure 5.6c), the inconsistent tuple ({address A, data B, redundancy B}) will be detected by eDECC. Further decoding of the word will reconstruct the address symbol from data and redundancy (address B in Figure 5.6c), diagnosing the faulty address pins by revealing the erroneous address that DRAM received. Figure 5.6d shows another example of eDECC organization using Bamboo QPC ECC. The original Bamboo codeword with 64 data symbols and 8 check symbols is extended to hold 4 extra address symbols. This eDECC organization is also able to detect and diagnose any 32-bit address error without additional redundancy.

5.3.2 Extended Write CRC for Timely Write Address Protection

Extended write CRC (eWCRC) extends the write CRC of DDR4 to protect the write address as well as the data, in a similar manner to how eDECC extends data ECC. To implement eWCRC, the DRAM controller generates an 8-bit CRC checksum from both the write data and its MTB address as

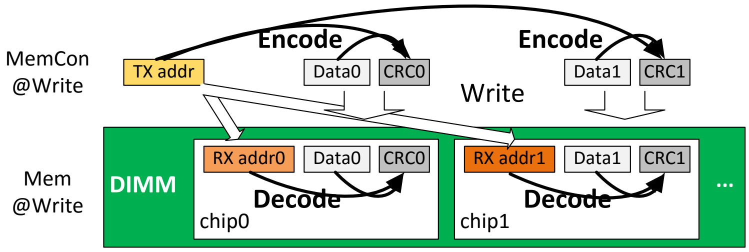


Figure 5.7: An overview of the Extended Write CRC (eWCRC).

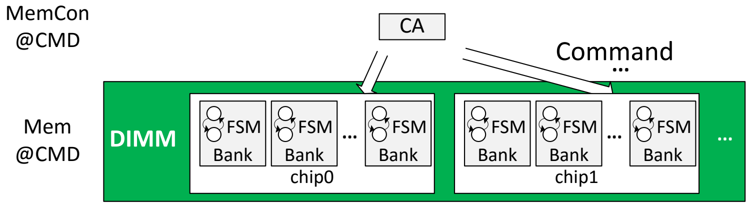


Figure 5.8: An overview of the Command State and Timing Checker.

shown in Figure 5.7. DRAM receives the CRC along with a write command and validates its data and address prior to changing the contents of memory. The detection coverage of eWCRC is 100% for any error that affects 8 or fewer contiguous address or data bits, and 99.6% for more severe errors. eWCRC detects write address errors prior to memory data corruption, allowing cheap common-case correction through write retry. In the rare case that an address error escapes the eWCRC, an MDC occurs; erroneously overwritten data can be diagnosed by eDECC, but the stale data that is left behind can result in SDC if read.

5.3.3 State and Timing Checker for Command Protection

The first tier of AI-ECC command protection is to detect illegal commands by tracking DRAM state transitions and command arrival times. AI-ECC adds a *Command State and Timing Checker (CSTC)* to the DRAM alongside each bank for this purpose; this CSTC checks the validity of received commands based on the memory protocol (Figure 5.8). At any given time, DRAM has a predetermined bank context (e.g., ACT/REF on an idle bank and RD/WR/PRE on an open bank) and a valid context-breaking command (e.g., ACT on an open bank) can be easily detected by monitoring bank states. In the previous example of duplicate activations (Figure 5.3), the second erroneous activation on the already open bank will be detected by CSTC before corrupting stored data. Missing activations are not immediately detected by the CSTC, but they change the bank state so that the next command to the bank will trigger a CSTC error and data integrity is not compromised. Additionally, command arrival times are monitored to detect erroneous commands that violate the DRAM timing guarantees. Each CSTC is implemented as a small *Finite State Machine (FSM)* using DRAM commands from the JEDEC standard and binned timing parameters that are known by the vendor. Table 5.1 shows the bank state and timing constraints for DDR4 DRAM, taken from the JEDEC specification [60]. Commands for DRAM initialization (mode register set and ZQ calibration) and for power saving modes (self-refresh and power down) are excluded for simplicity, but these commands are later included in our experimental evaluation.

Command Error Detection by Address Protection: While the CSTC can detect any state-altering command error (e.g., transition to or from ACT

Command	Bank state	Timing parameters
ACT	Idle	tRC, tRRD, tFAW, tRP, tRFC
REF	Idle	tRRD, tFAW, tRP, tRFC
RD	Open	tRCD, tCCD, tWTR
WR	Open	tRCD, tCCD
PRE	Open	tRAS, tRTP, tWR
NOP	Any	None

Table 5.1: DDR4 DRAM commands with their allowed bank state and timing constraints.

or PRE) and any timing violation, some errors are not necessarily caught by the CSTC; most of these errors either will be detected through the address checking mechanisms or do not compromise the functionality of the system.

Errors that corrupt a command to or from an RD result in an extra or missing read operation, respectively. An extra or missing read command corrupts the write pointer in the read data FIFO of the DDR PHY (physical interface) such that the memory controller receives a wrong entry from the PHY. This wrong entry is then detected by eDECC as it will not validate the address of the codeword taken from the read FIFO (this address is produced and stored within the memory controller itself and is not subject to transmission errors). An extra write command will attempt to interpret the I/O at the data pins as a value and will write this value back to the open row. The data interpreted from the undriven I/O pins can be random (if they are fully undriven) or all-ones (if they are partially driven by termination resistors); in either case, the erroneous write will be handled by the eWCRC and eDECC like any write data error.

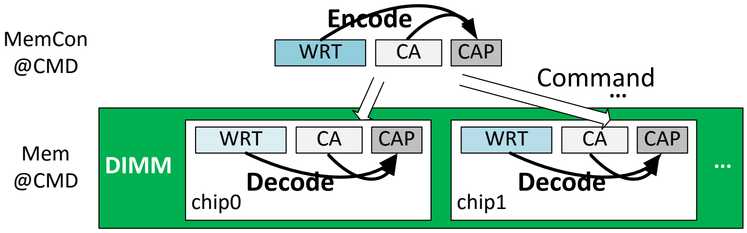


Figure 5.9: An overview of the Extended CA Parity (eCAP).

An extra refresh operation with valid timing may not be detected by the CSTC, but it does not affect correct operation. A missing refresh is not detected; DRAM has some retention time margins so that a lost refresh operation does not corrupt data.

5.3.4 Extended CA Parity (eCAP)

The only command error that is not covered by either the CSTC or the AI-ECC address checking mechanisms is a missing write. In this case, as DRAM never receives the command, it would not report an error and the memory controller would assume that an error-free write completes. To detect this erroneous situation, the CA parity of DDR4 is extended to cover missing WR commands (Figure 5.9). To enforce *extended CA parity (eCAP)*, the memory controller and memory maintain synchronized *write toggle (WRT)* bits that flip upon sending/receiving a WR command. The CA parity is then generated across both the 24 CA pin values and the WRT bit. If there is a missing WR, the WRT values in the memory controller and memory disagree on the next command and the error is detected.

5.3.5 Clock and Control Protection

Errors in the $\overline{\text{CK}}$ and $\overline{\text{CTRL}}$ signals can compromise commands as well. An additional toggle on $\overline{\text{CK}}$ results in the reception of an erroneous command, while a missing toggle causes a command to be lost. Errors in $\overline{\text{CKE}}$ (clock enable) and $\overline{\text{CS}}$ (chip select) can also incur an extra/missing command. These extra/missing commands are all detected by CSTC, eDECC, eWCRC, and eCAP as explained above. Errors in $\overline{\text{ODT}}$ (on die termination) degrade the data signal quality, which can be detected by eWCRC and eDECC.

5.3.6 Precise Diagnosis

eDECC not only detects address errors but can also pin-point faulty address pin(s) by restoring the original address and comparing it against the erroneous address. Such knowledge can be valuable to repair techniques, such as selectively tuning delay and drive parameters of the reported pin. Without this knowledge, extensive diagnostic routines are required to identify intermittent transmission errors or repeated CCCA errors may impact system reliability and availability.

5.3.7 Correction Details

AI-ECC correction is more straightforward than data correction via ECC. Because AI-ECC generally detects CCCA errors early (before possible data corruption), correction simply entails retrying the faulty command. In the rare case that AI-ECC detects an error late (after possible data corruption)

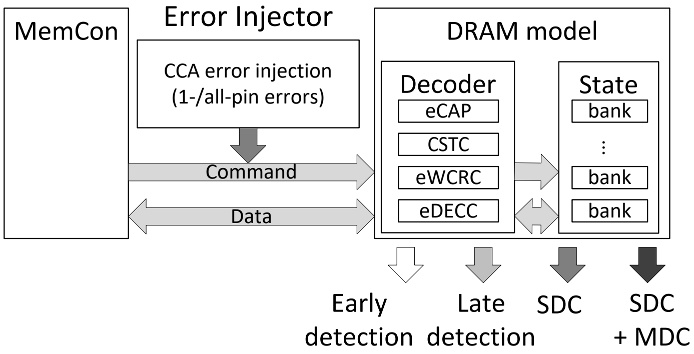


Figure 5.10: The CCCA reliability evaluation environment.

then a DUE must be flagged to higher system levels. This is no difference from current systems that encounter an uncorrectable error, and the uncorrectable AI-ECC error rate is shown in Section 5.4 to be low.

5.4 Evaluation

This section measures the reliability and efficiency of AI-ECC and compares them with current practices and state-of-the-art industrial patents. Section 5.4.1 analyzes the level of CCCA reliability provided by AI-ECC and prior protection techniques. Section 5.4.2 investigates the impact of AI-ECC on the strength of the SDDC data protection, finding it to be negligible. Section 5.4.3 estimates the system-level reliability of systems with transmission errors, showing the impact of AI-ECC to be significant in many scenarios. Finally, Section 5.4.4 describes the modest design changes (and correspondingly low overheads) required to extend a DDR4 memory to support AI-ECC.

5.4.1 CCCA Reliability

The CCCA error detection coverage is evaluated using Monte-Carlo error injection simulations, as depicted by Figure 5.10. Transmission errors are modeled as 1-pin, 2-pin, and all-pin errors; these models represent transmission noise such as inter-symbol interference, crosstalk with 2 victims, and power noise, respectively. A $\overline{\text{CK}}$ error is modeled as one source of all-pin errors, rather than individual 1-pin errors. Errors are injected into the 27 $\overline{\text{CTRL}}$, $\overline{\text{CMD}}$, and $\overline{\text{ADD}}$ signals of the target command. In the no-protection configuration, errors are not injected on the $\overline{\text{PAR}}$ pin because it is assumed to be non-existent or disconnected. A multi-pin error has 50% switching probability for each erroneous pin.

A CCCA error has different consequences with different DRAM commands; 5 dominant command patterns are tested: ACT (followed by WR), ACT (followed by RD), WR, RD, and PRE. ACT is sub-categorized because the consequence of an activate error depends on the following command—a missing ACT followed by a WR results in memory data corruption, while a missing ACT followed by an RD reads arbitrary data but does not corrupt storage. The impact of other command errors does not vary significantly based on the following command, and a single test sequence is used for each. A DRAM model is implemented to track DRAM state, decode erroneous CCCA signals, and interpret the impact of each error based on the decoded command and resultant DRAM state. Before each erroneous command, the simulated DRAM is set to have all banks open (except for erroneous ACTs where the

Erroneous Pin		DRAM Command in Error			
#	Name	ACT	WR	RD	PRE
27	CK	Clock error			
26,25	CKE, CS	ACT−	WR−	RD−	PRE−
24	ODT	On-die termination error			
23	PAR	No error			
22	ACT	ACT→?	ACT+		
21	RAS/A16	$\text{ACT}_{rowaddr}$	WR→MRS	RD→REF	PRE→ZQC
20	CAS/A15		WR→ZQC	RD→NOP	PRE→MRS
19	WE/A14		WR→RD	RD→WR	PRE→RFU
18∼15	BG1...0, BA1...0	$\text{ACT}_{bankaddr}$	$\text{WR}_{bankaddr}$	$\text{RD}_{bankaddr}$	$\text{PRE}_{bankaddr}$
14	A12/BC	$\text{ACT}_{rowaddr}$	$\text{WR}_{burstchop}$	$\text{RD}_{burstchop}$	No error
13∼11	A17,13,11		No error		
10	A10/AP		ACT−		
9∼0	A9...0		$\text{WR}_{coladdr}$	$\text{RD}_{coladdr}$	No error

SDC SDC and conditional MDC SDC and MDC

Table 5.2: The impact of 1-pin CCCA errors across pin locations and commands.

target bank is closed).

5.4.1.1 Impact of undetected CCCA errors

Table 5.2 presents results from undetected 1-pin CCCA errors. CMD-/CMD+/$\text{CMD}_A \rightarrow \text{CMD}_B$ indicate missing, extra, and altered commands (changed from CMD_A to CMD_B), respectively. A transition to MRS, ZQC, and RFU indicate that the DRAM was erroneously given a mode register set command, ZQ calibration command, or a reserved-for-future-use command, respectively. The key findings are summarized below (results of 2-pin and all-pin errors appear in later experiments, but they are not included in this analysis for simplicity). $\boxed{\textbf{ACT:}}$ Any undetected error during an ACT causes a failure. Errors in the row address select an incorrect row, resulting in SDC+MDC (if followed by WR) or SDC (if followed by RD). $\boxed{\textbf{WR:}}$ Three pins (A11, A13, and A17) do not participate in the WR operation and manifest no error. Errors on $\overline{\text{CKE}}$, $\overline{\text{CS}}$, $\overline{\text{ODT}}$, ACT, $\overline{\text{RAS}}$, $\overline{\text{CAS}}$, $\overline{\text{WE}}$, $\overline{\text{BC}}$, $\overline{\text{AP}}$, bank address, and column address manifest as SDC+MDC. $\boxed{\textbf{RD:}}$ Three pins (A11, A13, and A17) manifest no error. Errors on $\overline{\text{CKE}}$, $\overline{\text{CS}}$, $\overline{\text{CAS}}$, $\overline{\text{BC}}$, bank address, and column address manifest as SDC in the read data, while errors on other signals generate SDC+MDC. $\boxed{\textbf{PRE:}}$ Fourteen pins (A17, A13∼A11, A9∼A0) manifest no error, while errors on $\overline{\text{CKE}}$, $\overline{\text{CS}}$, ACT, $\overline{\text{RAS}}$, $\overline{\text{CAS}}$, $\overline{\text{WE}}$, A10 and bank address manifest SDC+MDC.

5.4.1.2 Detection Coverage

The generated CCCA errors are checked against four increasing levels of protection: no protection, DDR4 + data ECC, DDR4 + eDECC, and DDR4 + AI-ECC. DDR4 + eDECC is used to represent the strongest address protection available through some patents [104, 102].

Figure 5.11 shows the fraction of 1-/2-/all-pin CCCA errors that DECC, eDECC, and AI-ECC can detect. Some MDCs can be detected prior to data corruption (i.e., early detection), while others are detected after the corruption (i.e., late detection). $\boxed{\textbf{1-pin errors:}}$ CA parity detects 1-pin errors on the 24 CA signals but not on the 3 $\overline{\text{CTRL}}$ signals, two of which are problematic

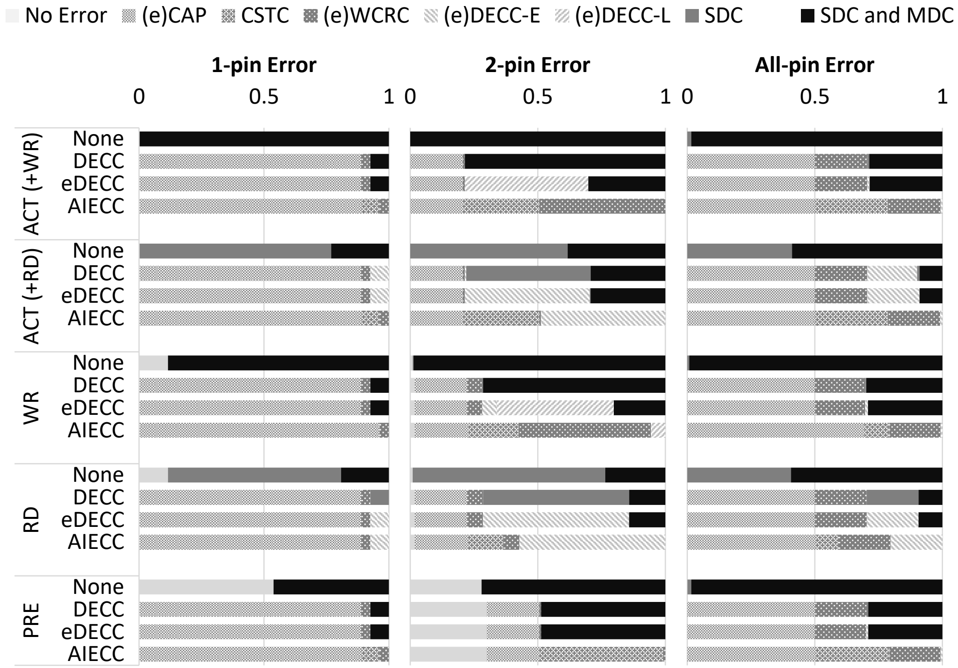


Figure 5.11: The CCCA error detection coverage of an unprotected DDR4 DIMM (None), DDR4+DECC (DECC), DDR4+eDECC (eDECC), and DDR4+AIECC (AIECC).

($\overline{\text{CKE}}$ and $\overline{\text{CS}}$). A missing RD command manifests as SDC with data-only DECC, yet it can be detected by eDECC. AI-ECC can detect all 1-pin errors. **2-pin errors:** The detection strength of CA parity is limited for 2-pin errors, resulting in large coverage holes in DDR4+DECC and DDR4+eDECC. AI-ECC fills the holes in CAP, avoiding nearly all SDC and MDC; only those errors that escape eWCRC (0.4% of such errors) cause an SDC when the target location that has stale data is read and cause an MDC at the erroneous location, which can then be detected by eDECC. **All-pin errors:** CA parity actually performs better in an all-pin scenario than with 2 erroneous pins, as it has a 50% chance

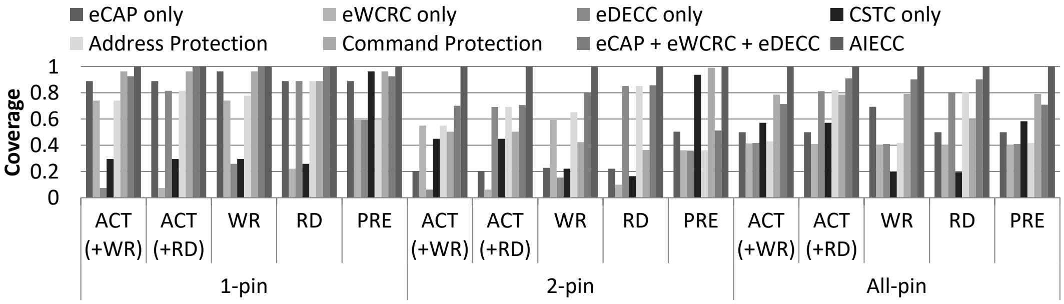


Figure 5.12: A quantitative evaluation of the different AI-ECC components: eDECC, eWCRC, address protection (eDECC+eWCRC), CSTC, eCAP, command protection (CSTC+eCAP), eDECC+eWCRC +eCAP (for completeness), and AI-ECC.

of detecting the error. However, as undetected command and control errors are likely to severely corrupt data, extra coverage is still needed. AI-ECC provides thorough command and control protection using CSTC and eCAP, and only errors that escape eWCRC may cause SDC+MDC.

Figure 5.12 shows which AI-ECC components detect different CCCA errors. It is apparent that the most effective CCCA protection mechanism heavily depends on the specific error scenario, such that all four AI-ECC mechanisms are required for robust CCCA error coverage. Address protection (eWCRC and eDECC) is crucial for protecting write and read commands. All-pin errors during activation are best detected by CSTC, as the errors frequently change the command into another (invalid) command. However, eCAP is the most effective mechanism for 1-pin activation errors, and 2-pin errors are protected by either address protection or CSTC, depending on whether the address or command is affected. Only through a combination of eDECC,

Address Error	Data Error	Protection	
		Data-only QPC	QPC+eDECC
None	1-bit	100% DCE	100% DCE
	1-chip	100% DCE	100% DCE
	rank	6×10^{-7}% SDC	6×10^{-7}% SDC
1-bit	None	100% SDC	100% DCE
	1-bit	100% SDC	100% DCE
	1-chip	100% SDC	4×10^{-7}% SDC
	rank	100% SDC	6×10^{-7}% SDC
32-bit	None	100% SDC	100% DCE
	1-bit	100% SDC	4×10^{-7}% SDC
	1-chip	100% SDC	6×10^{-7}% SDC
	rank	100% SDC	6×10^{-7}% SDC

Table 5.3: A comparison of the data and address protection coverage of Bamboo QPC with and without address protection.

eWCRC, CSTC, and eCAP is AI-ECC able to provide complete coverage.

5.4.2 Data Reliability

Despite its strong level of CCCA protection, AI-ECC has a negligible impact on the levels of existing data protection. To evaluate potential ECC coverage loss from protecting both data and address, Bamboo QPC is evaluated with and without eDECC against data and address errors using random error injection simulations (Section 3.4.1). Table 5.3 compares the level of data and address protection. With no address error, eDECC does not change the correction capability and does not show observable differences in the very strong detection capability against data errors. With concurrent address and data errors, eDECC can correct most of them with decoding (if the number of erroneous symbols is no greater than 4) or with a retry (otherwise) while data-only protection results in SDCs.

5.4.3 System Reliability

Section 5.4.1 shows that AI-ECC provides thorough and strong protection against CCCA errors on any and all pins. Its implication on system-level reliability, however, depends on the underlying CCCA error rate. Due to a lack of publicly available DDR4 CCCA error rates, a sensitivity sweep on the BER is performed and multiplied with command bandwidths and signal counts to estimate CCCA error rates. Other approaches for CCCA errors, such as simulation-based modeling and lab measurements, are excluded due to factors like confidential process and layout information and low confidence from limited samples.

Due to the dependence of CCCA error manifestation on the DRAM command stream, the DRAM behavior of 56 benchmarks from the NPB, SPEC CPU 2006, PARSEC, and SPLASH2X benchmark suites[1] [2, 130, 17, 147] is characterized using the Xeon E5 and E7 v3 memory controller performance counters [5]. Four representative clusters are identified by hierarchically clustering across the memory bandwidth utilization, read to write ratio, CAS to ACT ratio, and ACT→RD to ACT→WR ratio of each program. Three of the clusters differ mainly in their data bandwidth utilization, and the other

[1] The C, ref, and native input sets are used for these suites, with one thread per core. SPEC CPU 2006 is run with 10 replicated processes in order to fit in a 16GB memory footprint.

# Apps	Major Feature	Data BW	Command BW ($\times 10^6$ cmds/sec)				
			ACT (+WR)	ACT (+RD)	WR	RD	PRE
33	Low Data BW	0.50%	0.64	0.39	0.69	2.22	1.03
10	Med. Data BW	7.90%	9.18	16.7	8.57	33.3	25.9
11	High Data BW	22.0%	39.4	76.2	29.2	90.1	116
wat-ns	High RD/WR	4.31%	0.15	6.13	0.17	23.6	6.28

Table 5.4: Representative benchmark clusters and their bandwidths.

(SPLASH2X's wat-ns) is an outlier with an extreme read-to-write ratio.[2] Table 5.4 gives the data and command bandwidth for the median centroids of these clusters.

For a given BER, the CCCA FIT rate of each representative centroid is estimated using Equation 5.1. This equation accumulates the FIT contribution from each CCCA-sensitive command—ACT (+WR), ACT (+RD), WR, RD, and PRE—over all 1-pin errors. Apart from a $\overline{\text{CK}}$ error, which affects all pins, no multi-pin errors are modeled due to a lack of data on their rates and distributions. 3.6×10^{12} is multiplied to convert per-second bandwidth to FIT.

$$\begin{aligned} FIT_{CCCA} = BER \times \sum_{i \in \text{CMD}} \sum_{j \in \text{ERR}} \{ & \{\text{Command Bandwidth}\}_i \\ & \times \{\text{Signal Count}\}_j \\ & \times \{\text{Undetected Probability}\}_{i,j} \\ & \times 3.6 \times 10^{12}\} \end{aligned} \tag{5.1}$$

[2] NPB's cg is also an outlier with a high but less-extreme read-to-write ratio.

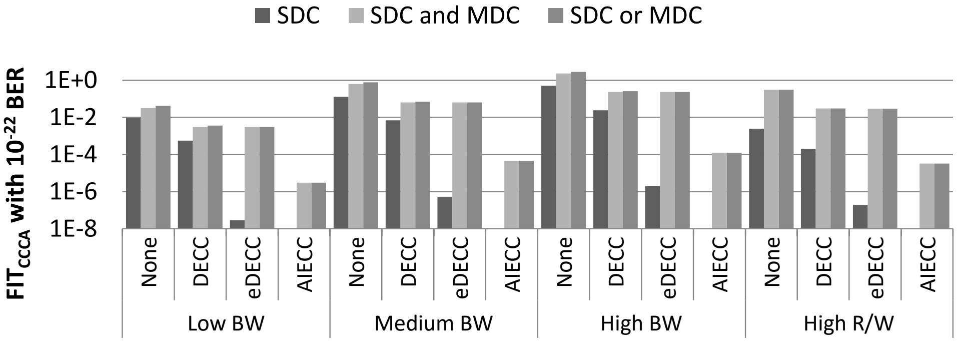


Figure 5.13: An estimation on ×4 DRAM CCCA FIT rates after protection with 10^{-22} BER.

$$\text{CMD} = \{\text{ACT(+WR), ACT(+RD), WR, RD, PRE}\}$$
$$\text{ERR} = \{\text{1-pin error, all-pin error}\}$$

The evaluation sweeps from 10^{-16} BER, which is the minimum design specification for data in the JEDEC DDR4 standard [60], up to 10^{-22}. The minimum JEDEC BER of 10^{-16} is very weak and corresponds to 2.8×10^{6} unprotected FIT_{CCCA} per device with the high-bandwidth centroid. This BER is likely to be higher than that in a real system—the FIT of a DDR2/DDR3 x4 DRAM device (including both storage and transmission errors) is around 25–66 [128, 129, 127]. The stronger BER of 10^{-22} corresponds to 2.8 FIT_{CCCA}, 3.4×10^{6} system-FIT and a 12-day MTTF on a system with 1.2M DRAM devices. This number could be within an order of magnitude of the CCCA error rate exhibited by the Cielo system [127], which has a similar number of DDR3 chips as the modeled system; however, the actual measurements on Cielo are not public at the time of writing this dissertation.

Figure 5.13 shows FIT_{CCCA} from 10^{-22} BER with and without protec-

System	BER	Protection			
		None	DECC	eDECC	AI-ECC
1.2M DRAM devices	10^{-22}	12 days	4 months	5 months	768 years
	10^{-21}	1 day	13 days	15 days	77 years
	10^{-20}	3 hours	32 hours	35 hours	8 years
36M DRAM devices	10^{-22}	10 hours	4 days	5 days	26 years
	10^{-21}	1 hour	10 hours	12 hours	3 years
	10^{-20}	6 minutes	1 hour	1 hour	3 months

Table 5.5: An evaluation of mean time to CCCA SDC failure on systems with 1.2M and 36M DRAM devices and high bandwidth utilization.

tion. Applications with more data bandwidth have a higher FIT_{CCCA} because they issue more commands. DDR4+DECC reduces the CCCA SDC and MDC rates by an order of magnitude using the DDR4 reliability mechanisms (e.g., CAP). eDECC reduces the SDC rate further by detecting read address errors. AI-ECC improves the unprotected CCCA failure rate by four orders of magnitude because it detects all read errors and nearly all write errors (as was shown in Section 5.4.1.2). Large-scale systems utilizing DDR4 can suffer from increased CCCA error rates due to its doubled transfer rate, and higher BER values of 10^{-21} and 10^{-20} are tested as well. These BERs change the Y-axis scale of Figure 5.13 yet the shape remains the same.

Table 5.5 shows the estimated CCCA *mean-time-to-failure (MTTF)* of a current-scale large system (1.2M DRAM chips in the Cielo supercomputer [127]) and a future-scale large system (36M DRAM chips or 1M 2-rank DRAM channels for the Exascale computing) with high bandwidth utilization. In the current-scale system, AI-ECC provides an MTTF of 8 years, even with the more severe BER. In contrast, other schemes have orders of magnitude worse reliability with an MTTF of just 1 or 2 days. Even in the future-scale system, AI-ECC can still provide high reliability against CCCA errors.

5.4.4 Hardware Overheads

AI-ECC has negligible hardware overhead and it represents a straightforward upgrade to existing DDR4 features and SDDC data ECC. It requires no additional pins or bandwidth to and from memory as it reuses the existing WCRC and CAP mechanisms and it does not require any additional ECC storage. A Verilog model of AI-ECC is implemented and synthesized with the Synopsys toolchain and the TSMC 40nm LP standard cell library [135, 136] to estimate its logic overheads. eCAP/eWCRC/eDECC+AMD/eDECC+QPC have area overheads equivalent to 30/180/140/2200 NAND2 gates and dynamic+static power increases of 0.01/0.1/0.05/0.8mW, respectively. The logic depth of the DECC decoder increases by 1 XOR gate and cycle latency is not impacted. On the DRAM side, eCAP/eWCRC/CSTC require the area equivalent of 30/180/9000 NAND2 gates per chip and consume 0.01/0.1/0.8mW, respectively. CSTC is off the critical path and it does not affect memory latency. The AI-ECC correction procedure requires support in the memory controller for command replay. Most systems, however, already support on-demand scrubbing[3] [9, 30] (writing back corrected data to DRAM to eliminate transient bit-flips), and the further additions for AI-ECC correction are ex-

[3] On-demand scrubbing is also called redirect scrubbing by AMD

pected to be modest.

While eDECC does not impact performance with QPC ECC, it could potentially increase read latency if AMD chipkill or other SDDC schemes are used. These schemes can detect data errors before a read transfer is complete, but would have to wait until the end of a block transfer for precise eDECC address error diagnosis. Prior simulation results of Bamboo ECC show this performance overhead to be modest (an average of 0.7% on a 64-bit DDR4-2400 data channel (Section 3.4.3)). Alternatively, a different eDECC organization could sacrifice precise diagnosis to enable early data error detection, or asynchronous ECC checking [103] could avoid this penalty in the common case; a detailed exploration of AI-ECC performance with alternative ECC schemes is left for future work.

5.5 Related Work

The topic of the end-to-end memory protection is of great importance to the server industry, and some form of CCCA protection has been the topic of patents from Freescale [98], Fujitsu [84], IBM [22, 23, 102], Intel [143], Micron [110, 140], Sun (now Oracle) [114, 146], Azul Systems [104, 105], and others [31, 144]. The different approaches taken by these protection mechanisms (as well as a single chain of academic literature [34, 35, 121]) are described below.

Separate Address Protection: Various authors describe techniques to protect the address signal using separate check bits [110, 140, 114]. While these approaches are straightforward, they are inefficient as they sacrifice transmission bandwidth and require additional signals to and from memory.

Combined Address and Data Protection: Address protection can be combined with data protection while sharing check bits. It appears that two prevailing techniques exist for such protection: codeword transformation and combined ECC.

Codeword transformation uses the address information to transform the data or check bits upon a write, reversing this transformation before ECC decoding. In the case of an address transmission error, this reverse transformation will corrupt the codeword; this corruption may be detected via data ECC. Nicholas [102] partitions a write data block into 32 sub-blocks and XORs each sub-block with a corresponding address bit. Sub-blocks should be organized to report address errors as detectable-yet-uncorrectable; no specific ECC or details are mentioned, however. Normoyle [104] and Wong et al. [146] generate checksums over the address and later XOR this information in with the DECC check bits. Normoyle generates a 4-bit CRC checksum from the address and merges it with 2 DECC nibbles, which results in a detectable-but-uncorrectable error in 4-bit symbol SSC-DSD codes. Wong merges a 1-bit address parity into at least two check bits for similar capabilities in SEC-DED codes.

An alternative mechanism, combined ECC, combines the address with the data prior to ECC encoding. Chen et al. [22, 23] describe a linear code that accepts the address along with the data and encodes them together for either a 140 or 146-bit channel, providing relatively weak address protection but taking special care to correctly diagnose address errors in most cases. Vogt [143] appends the address to the data before encoding, with no consideration of ECC specifics or correct address error diagnosis. The same approach was studied by Gumpertz for checking a variety of metadata, including the storage address [34] (also mentioned in [35]). The concept is described with respect to SEC-DED ECC and is not evaluated in the context of off-chip memory. Sazeides et al. [121] describe a scheme to combine metadata with ECC, but they do not consider address errors or off-chip memory. Normoyle and Hathaway [105] first take a CRC of the address and then append it and other metadata to the data before encoding.

Protection of Control Signals: The protection of control signals through separate ECC has been proposed, trading transmission bandwidth for reliability [110, 114]. Partial protection of the DRAM command stream through history tracking is proposed by Wang [144] and an abandoned patent application by Romdhane [31]. These approaches check for illegal command sequences or timing violations without requiring any additional signals to or from memory, but they cannot provide complete protection against command and control errors.

Relationship of Prior Work with AI-ECC: AI-ECC is designed to put the CCCA reliability mechanisms present in DDR4 [60] to good use in a cohesive and comprehensive protection scheme; its relationship with this work is clear and intentional. AI-ECC further protects the $\overline{\text{ADD}}$ signal by concatenating it with the data before encoding, similar to [143, 34]. AI-ECC is the first approach that considers strong levels of protection (100% address error detection with detailed error diagnosis for address transmission errors) and represents the first realistic evaluation of a combined ECC protection scheme for off-chip memory.

The CSTC proposed for AI-ECC is similar in principle to the mechanisms described by [31, 144], yet is more complete. AI-ECC is the first to evaluate the coverage of protocol tracking, and it augments the approach to provide the complete coverage of command errors by extending the DDR4 CA parity signal to fill gaps (such as a missing WR command).

When taken as a whole, AI-ECC both combines and extends the best aspects of prior CCCA protection efforts. No other approach simultaneously protects against data, address, clock, and command errors with error coverage as high as AI-ECC. AI-ECC also performs early write error detection, is able to correctly diagnose address errors, and it does not require additional redundancy or new signals to and from DRAM.

5.6 Summary

All-inclusive ECC is a readily-implementable suite of complementary error protection mechanisms for DRAM data and CCCA signals. Evaluation results demonstrate that data ECC, the DDR4 reliability mechanisms, and address protection similar to that which may be used by industry are insufficient for the complete end-to-end protection of DRAM; AI-ECC supplements these current practices to provide strong holistic protection and enables more aggressive growth in transfer rates. Despite its advantages, AI-ECC has minimal associated costs and requires no new signals to or from memory, no additional storage, negligible hardware real-estate, and it negligibly affects the level of data protection.

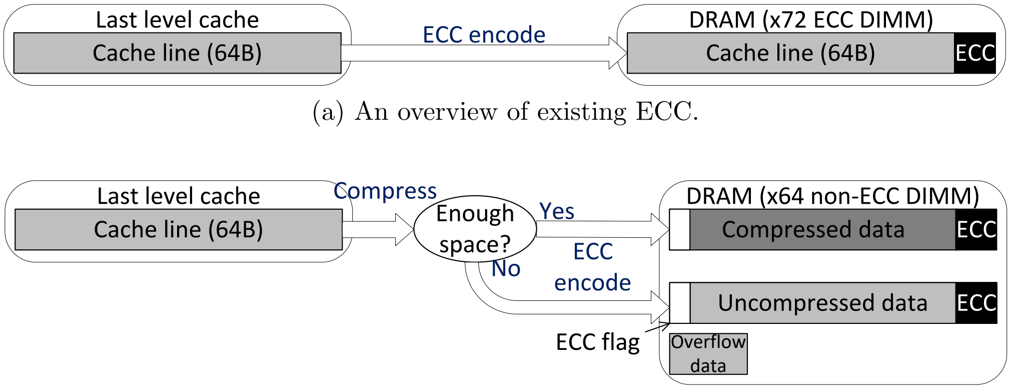
Chapter 6

Efficient Protection using Compression

Memory protection trades off redundant storage, bandwidth, and energy for increased reliability. DRAM ECC typically employs ECC DIMMs that have 12.5% more chips than non-ECC DIMMs; hence ECC typically adds 12.5% capacity, bandwidth, and energy overheads. Strong SDDC protection increases the energy consumption even further by necessitating compromises in memory system design; such as using narrow ×4 devices or using wide 128-bit channels.

This chapter presents a versatile and efficient ECC solution, called *Frugal ECC (FECC)* that offsets the ECC overheads through fine-grained compression. In conventional *single-tiered* ECC organizations, redundant information is stored in dedicated memory devices that are accessed together with the devices that store data (Figure 6.1a). In this way, ECC does not impact performance when compared to unprotected memory. FECC compresses main memory at cache-block granularity, using any leftover space to

Parts of this chapter appear in [78]. The author is the main contributor of the idea, implementation, and evaluation.



(a) An overview of existing ECC.

(b) An overview of Frugal ECC.

Figure 6.1: A comparison between Frugal ECC and existing ECC.

store ECC information. Compressed data and its ECC information are then frequently read with a single access even without redundant memory chips; insufficiently compressed blocks require additional storage and accesses (Figure 6.1b). FECC attempts to approach the performance and reliability of the conventional single-tiered organizations while at the same time reduce ECC overheads and enable more code design flexibility. As examples, this chapter presents SDDC ECCs on a non-ECC DIMM with ×4 chips and the first true SDDC ECC for ×8 devices using an ECC DIMM.

6.1 Frugal ECC

Figure 6.2 provides an overview of how FECC works. On a memory write, FECC losslessly compresses a data block to remove data-inherent redundancy and free up space. If FECC successfully creates enough space for the redundant ECC information, the compressed data and ECC check bits

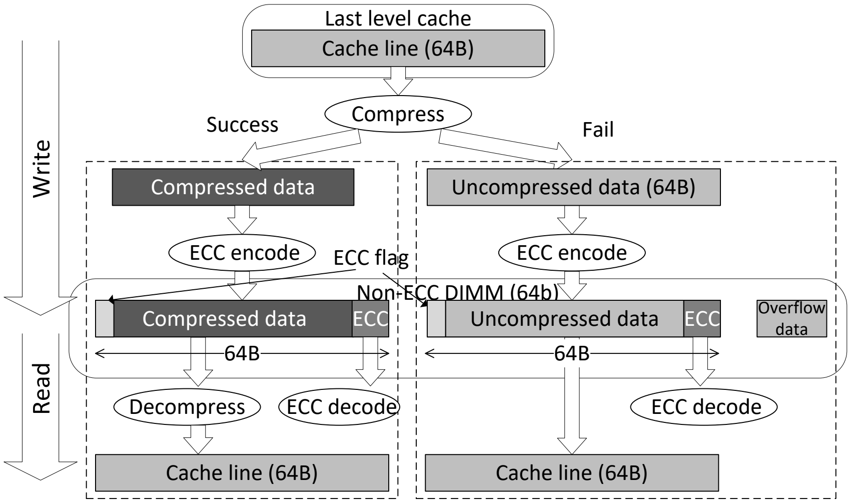


Figure 6.2: Memory reads and writes with Frugal ECC using a 64-bit non-ECC DIMM.

are stored in the memory footprint of the uncompressed data. As its starting address does not change, a compressible memory block can still be randomly accessed and no additional address translation procedure or level of indirection is needed. Furthermore, as with a conventional ECC, both data and redundant ECC information are accessed in unison. On a memory read, ECC checking and decompression occur in parallel as ECC is computed on the compressed data. If no errors are detected, the decompressed data is forwarded to the last-level cache. Otherwise, ECC attempts to recover any lost data and decompression repeats using the corrected data.

Not all blocks can be compressed to the reduced footprint needed for ECC. Memory blocks that fail to meet the compression threshold for a given ECC scheme result in a *compression exception*. Upon a compression exception, FECC uses one full block and one partial block to store the uncompressed data and ECC check bits. Each compression exception degrades performance, because both the write and any future read require two (potentially cached) accesses to memory instead of the single access used in a conventional ECC design.

As the ECC layouts for compressed and exception data differ, some ECC meta-data is used to identify whether compression was successful and to determine the resulting ECC layout. FECC stores the *ECC flag* that indicates which layout is used along with the compressed data and applies an additional level of error protection to the flag—this higher level of protection is necessary because a compromised ECC flag indicates that a wrong set of bits should be treated as ECC information; using incorrect bits for ECC decoding degrades the protection level. FECC therefore uses a 3 to 5-bit ECC flag, as described further in Section 6.3.

FECC statically reserves enough storage for its overflow data and accesses this overflow storage as-needed using a deterministic addressing scheme. After a read, if the ECC flag identifies an exception, FECC deterministically generates the address of its overflow data and checks the *Last Level Cache (LLC)*. Upon a miss, FECC fetches the overflow data from DRAM (similar to VECC [151]). The resulting worst-case latency is the LLC miss penalty plus the round-trip delay between the memory controller and LLC, but LLC caching reduces the latency in the common case.

The key to the success of FECC is a design that minimizes exceptions while maintaining enough redundancy to meet reliability goals. As such, FECC requires both state-of-the-art compression and ECC schemes. The following sections present a novel compression scheme, *Coverage-oriented Compression*, to increase compression coverage far beyond what is possible with prior compression techniques and also describe how FECC can utilize the most aggressive available ECC schemes.

6.2 Coverage-oriented Compression

While most existing compression schemes focus on improving the compression ratio, compressing data to a size as small as possible, left-over space from an over-compression is left unused in FECC. In the meantime, compression exceptions due to under-compression cost FECC additional memory accesses for overflow data, consuming additional latency, bandwidth, and energy. Therefore, FECC relies on a novel compression scheme, *Coverage-oriented Compression (CoC)*, which focuses on maximizing coverage for the modest compression ratio goals needed for ECC.

The fundamental observation behind CoC is that there are tradeoffs between compression ratio and compression coverage. A conventional compression-ratio-oriented compression assigns short codes to the most frequent values to reduce the average number of bits needed to store a value. A key drawback of this assignment is that the number of available "good" codes runs out quickly. Coverage-oriented compression, on the other hand, tries to cover as many values as possible with "acceptable" codes to maximize coverage. For example, an ideal coverage-oriented compression would assign m-bit codes to the most

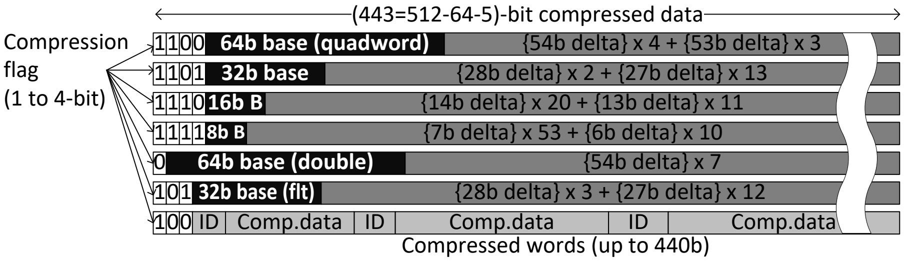

(443=512-64-5)-bit compressed data
Compression flag (1 to 4-bit)
1100 64b base (quadword) {54b delta} x 4 + {53b delta} x 3
1101 32b base {28b delta} x 2 + {27b delta} x 13
1110 16b B {14b delta} x 20 + {13b delta} x 11
1111 8b B {7b delta} x 53 + {6b delta} x 10
0 64b base (double) {54b delta} x 7
101 32b base (flt) {28b delta} x 3 + {27b delta} x 12
100 ID Comp.data ID Comp.data ID Comp.data
Compressed words (up to 440b)

Figure 6.3: The compressed data layout for the 448-bit target (64-bit redundancy).

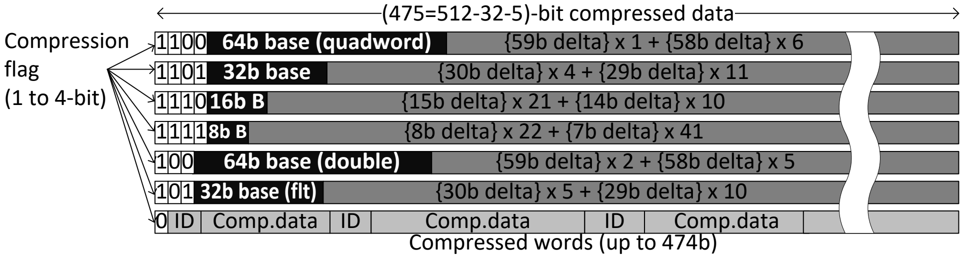

(475=512-32-5)-bit compressed data
Compression flag (1 to 4-bit)
1100 64b base (quadword) {59b delta} x 1 + {58b delta} x 6
1101 32b base {30b delta} x 4 + {29b delta} x 11
1110 16b B {15b delta} x 21 + {14b delta} x 10
1111 8b B {8b delta} x 22 + {7b delta} x 41
100 64b base (double) {59b delta} x 2 + {58b delta} x 5
101 32b base (flt) {30b delta} x 5 + {29b delta} x 10
0 ID Comp.data ID Comp.data ID Comp.data
Compressed words (up to 474b)

Figure 6.4: The compressed data layout for the 480-bit target (32-bit redundancy).

frequent 2^m values, so that all of these values can be compressed into an m-bit footprint.

CoC has three main components that together maximize coverage with acceptable implementation overheads: 1) fitting base + delta compression for homogeneously-typed data, 2) exponent compression for floating-point data, and 3) frequent word pattern compression for heterogeneously-typed data. These three components are described in detail below.

6.2.1 Fitting Base + Delta Compression

If a data block has homogeneously-typed data, values within the block are likely to have similar values. Base-Delta-Immediate (BDI) compression exploits this small dynamic range of integer and pointer types by storing the first value as a base and compressing all the other values as small differences from this base [113]. For blocks with mixed integer and pointer types, BDI adds another base value (implicitly assumed to be 0) that is used for integers while the original base is employed for pointers.

CoC extends the Base-Delta compression of BDI and increases delta sizes so that the overall compressed size fits within the modest thresholds needed for FECC. The top 4 organizations in Figure 6.3 and Figure 6.4 show the layouts used to represent integer data when targeting 448-bit and 480-bit layouts, respectively. The 448-bit and 480-bit compressed sizes yield room for 64-bit and 32-bit redundancy, respectively and reserve 5-bit for ECC layout information (Section 6.3).

Each organization includes the *compression flag* to indicate the type of compression used, a base value, and a set of deltas. The 448-bit footprint is used to hold a compression flag (up to 4-bit) and a 64-bit base value. The remaining 375 bits are used to store deltas for the other 7 64-bit values in the block, which allows 54-bit deltas for 4 of the 7 64-bit values and 53-bit deltas for 3 of the 7 64-bit values. By fully utilizing every bit of the available footprint, *Fitting Base+Delta (FBD)* supports large delta sizes and compresses more cache lines without exceptions. For 64-bit data, the delta sizes are large enough to compress a mixture of 64-bit integers and 64-bit pointers without the need for a separate base for pointers. FBD follows similar reasoning for maximizing delta sizes to fit within the compression footprint for 32-bit, 16-bit, and 8-bit integers, as shown in Figures 6.3 and 6.4.

6.2.2 Floating-point Compression

Low-latency floating-point (FP) compression is difficult, as the normalized floating-point significand can cause small value changes to manifest as a very different binary representation. As the target compression ratio in FECC is low (8:7 compression for 12.5% redundancy), FP compression in CoC only targets the exponent and sign of floating-point data. These fields often have higher value locality and comprise a larger number of bits than FECC compression goals (18.8% and 28.1% of double and single-precision numbers, respectively [52]). Similar to the use of delta coding for integer values, CoC can compress the sign and exponent with simple subtraction assuming that their values exhibit locality. The bottom organizations in Figure 6.3 and Figure 6.4 show how CoC compress homogeneously-typed floating-point data using FBD compression.

The 448-bit footprint is used to hold a 1-bit compression flag and a 64-bit base value with the remaining 378 bits evenly split to represent the 7 remaining FP values. Each 54-bit value represents the 1-bit original sign, 52-bit original mantissa, and a 1-bit exponent delta from the exponent of the base. While a 1-bit delta can cover only +0 and -1 changes in the exponent, Section 6.5.2 demonstrates good coverage with this scheme across a range of applications. For single-precision floating-point numbers, the delta size is increased to 3 or 4 bits, because our experiments show a need for a larger range of exponent differences. For the 480b compression target (32-bit redundancy for ECC), the compressed size for double-precision FP increases to 58 or 59 bits and CoC uses the additional bits to represent 6 or 7-bit exponent deltas.

6.2.3 Frequent Word Compression

While FBD provides high compression coverage for homogeneously-type data, it suffers from suboptimal coverage with heterogeneously-typed data (e.g., structs and classes). In particular, FBD works poorly with heterogeneously-typed data with mixed fixed and floating-point numbers, as no single base is appropriate.

To augment this weakness, CoC employs a secondary compression scheme, called *Frequent Word Compression (FWC)*, that compresses at the 64-bit granularity. Each 64-bit data is compressed as either a frequent pattern [12] or as a difference from a previous 64-bit data within its block. The per-word compressor suite in FWC is selected from a large pool of compressors with different bases, bit positions, and delta sizes. As per-word compression is designed to complement FBD, SPEC CPU 2006 benchmark suite (using "test" inputs) is run to collect the blocks that are incompressible with FBD for the 448-bit target. Then, the top 16 compressors based on their estimated coverage are selected with IDs based on Huffman coding [46] to reduce the necessary amount of meta-data. Section 6.5.2 shows that this choice of compressors and encoding is robust across other benchmark suites and inputs. Table 6.1 shows the complete list of the selected compressors.

Description	Size (bits)	ID
All zero 64-bit	0	001
Same as left 64-bit	0	1000
Same as left-left 64-bit	0	111110
Same as left-left-left-left 64-bit	0	11110
32-bit data, 32-bit zero	32	11010
2 x {16-bit sign-extension to 32-bit}	32	11011
32-bit sign-extension	32	1010
48-bit sign-extension	48	11100
44-bit delta from left 64-bit	44	11101
52-bit delta from left 64-bit	52	1001
12-bit delta from left-left 64-bit	12	1011
sign, exponent([63:52]) as 4-bit delta from bias (1023)	56	010
sign, exponent([63:52]) as 8-bit delta from bias (1023)	60	011
sign, exponent([63:52]) as 4-bit delta from left 64-bit	56	1100
Exponent([62:52]) as 3-bit delta from bias (1023)	56	111111
Incompressible 64-bit	64	000

Table 6.1: The frequent word compression mappings (per 64-bit data).

6.3 ECC Flag Protection

The ECC flag in FECC identifies which of several possible ECC layouts is used for a particular block; the specific layout depends on the code design and the level of compression achieved for a given block. In current designs, FECC aims to free up either 64 bits or 32 bits of data for ECC; the terms *full and half-compression* are used to denote whether the full 64 bits needed for some ECCs are freed or whether only half that amount are reclaimed, respectively. An error in the ECC flag can lead to undetected errors because data will be interpreted as ECC information or vice versa.

Different FECC schemes require a different number of possible layouts.

If there are only two layouts (uncompressed / compressed), triple-modular-redundancy can be used to encode the 1-bit ECC flag; the flag is replicated twice and a majority vote between the 3 copies determines its value (Figure 6.6). If there are three layouts (uncompressed / half-compressed / fully-compressed), three different ECC flags are needed. To guarantee correction from a single bit error on the flag, the 3 codewords are different in at least 3 bits (with a Hamming distance of 3) and codewords of 4 or 5 bits are used (Figure 6.5). In all cases, the bits used to store the ECC flag are distributed over different chips so that a chip error corrupts only a single bit of the encoded flag, and is therefore correctable. If two chip errors compromise a flag, a wrong data layout may be selected for decoding. However, even in this case, the codes used by FECC have very high detection coverage so that decoding with incorrect layout is very likely to result in a detected error.

6.4 Frugal ECC Organizations

FECC can flexibly trade off performance for error protection using different ECC schemes and by changing its target compression threshold. Several compelling configurations are presented in following subsections.

6.4.1 Frugal ECC for x4 SDDC

FECC can employ ×4 SDDC ECCs to provide the same level of protection and a similar level of performance without the use of redundant memory devices. Table 6.2 shows some of the efficient ×4 SDDC and SDDC-level pro-

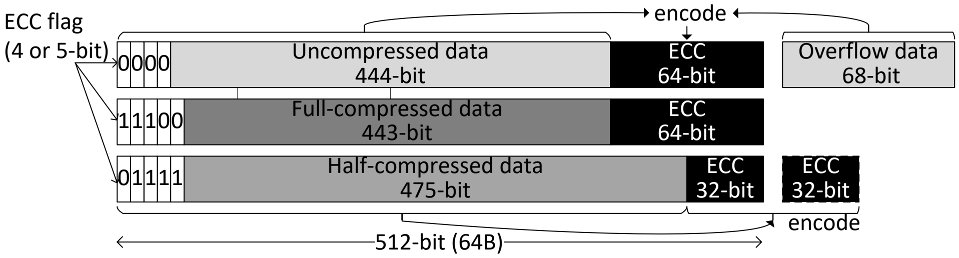


Figure 6.5: The Frugal ECC layout for AMD chipkill and QPC Bamboo ECC (64-bit redundancy).

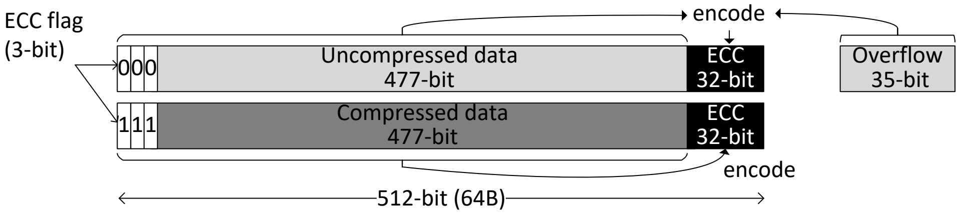


Figure 6.6: The Frugal ECC layout for ×4 Multi-ECC (32-bit redundancy).

ECC	Channel width	Protection	Overheads		
			Chip	Storage	Access
AMD chipkill	72-bit	1 chip correct	12.50%	12.50%	None
Bamboo QPC	72-bit	1 chip correct	12.50%	12.50%	None
FECC+Multi	64-bit	Almost 1 chip correct	0.00%	7.25%	R/W (exception)
FECC+QPC	64-bit	1 chip correct	0.00%	13.50%	R/W (exception)

Table 6.2: A comparison of different SDDC and SDDC-level ECCs for ×4 devices over a 64-bit data channel.

tection mechanisms with and without FECC. AMD chipkill [9] and Bamboo QPC (Section 3.3) are used as examples of ×4 SDDC. Both schemes require 12.5% redundancy (8 bits on a 64-bit channel) and FECC can provide this space by compressing a 512-bit block into 443 bits of data and 5 bits for the ECC flag. This 448b *compression target* determines the rate of compression exceptions, and thus the impact on performance. Therefore, to minimize performance degradation even further, the ECC schemes are treated as a two-tiered ECC [151] with a 6.25%-redundancy first-tier error code (T1EC) that can detect any single-chip error and a second-tier 6.25%-redundancy code (T2EC) that can correct any such error.[1]

If a block compresses to the 448-bit threshold, the T1EC and T2EC check bits are stored along with compressed data in the original uncompressed data footprint. If a block is half-compressed to a size in between 448 and 480 bits (including the 5b ECC flag), the T1EC is stored along with compressed data but the T2EC is stored in a separate block. If a block fails to compress to even a 480-bit footprint, part of the uncompressed data is stored along with the T1EC and T2EC in a separate memory block. These organizations are depicted in Figure 6.5. Note that the ECC flag is protected using a 4 or 5-bit encoding to ensure true SDDC reliability—even if a chip with ECC flag has errors, the ECC flag will be correctly decoded.

Multi-ECC [55] was published as SDDC-level protection for ×8 DRAM

[1] The T2EC code must be combined with the T1EC to correct a single-chip error. It is not independently capable of this level of correction.

ECC	Channel width	Protection	Overheads		
			Chip	Storage	Access
OPC	144-bit	1 chip correct	12.50%	12.50%	None
VECC	144-bit	1 chip correct - 2 chip detect	12.50%	18.75%	W
VECC	128-bit	1 chip correct - 2 chip detect	0.00%	18.75%	R/W
LOT-ECC	72-bit	Most 1 chip correct	12.50%	26.50%	None
Multi-ECC	72-bit	Almost 1 chip correct	12.50%	12.90%	W
FECC+OPC	72-bit	1 chip correct	12.50%	26.00%	R/W (exception)
FECC+Multi	64-bit	Almost 1 chip correct	0%	13.50%	R/W (exception)

Table 6.3: A comparison of different SDDC and SDDC-level ECCs for ×8 devices.

devices through erasure coding. It uses one redundant symbol to detect any single ×8 chip error and employs a separate checksum (which is shared among multiple lines to amortize storage overheads) to locate the faulty chip. Once the error location is identified, it uses erasure coding with the previous one-symbol redundancy to correct the error. For ×4 SDDC FECC, Multi-ECC is modified to build a 16-bit RS symbol from 4 data transfer beats, reducing its main ECC redundancy down to 6.25%; the same checksum mechanism is maintained as in the original Multi-ECC design. FECC with ×4 Multi-ECC can access any block that is compressed to a 480-bit footprint without accessing secondary storage. In the case of a compression exception, the data and its ECC information are split over two blocks. This is depicted in Figure 6.6. The ECC flag for Multi-ECC is protected using a 3-bit encoding because it has only two ECC layouts and triplicating the 1-bit flag can correct a single-bit error.

6.4.2 Frugal ECC for ×8 SDDC

The amount of redundancy required for the ×8 true SDDC (Bamboo OPC or S8SC in Section 3.3) over a narrow channel is 25%; 16 bits on 64-bit data channel. As CoC primarily focuses on 12.5% redundancy, FECC uses ECC-DIMMs with 12.5% redundant devices to split the 25% ECC overhead between the redundant chips and the space made available by compression. FECC with OPC uses a two-tiered ECC so that if a block is compressed to less than 448 bits (443 bits of data and the 5-bit ECC flag), the compressed data and the 128 bits of redundant information (T1EC + T2EC) is stored within a single block of an ECC DIMM. If a 512-bit block is compressible to 512 bits (some blocks fail due to compression and ECC flag overheads), the ECC flag, compressed data, and 64-bit T1EC are stored on the ECC DIMM. In this case, a write accesses an additional memory location for the T2EC but error-free reads require only a single memory access. If compression fails entirely, part of the uncompressed data, the T1EC, and T2EC are stored across the ECC chip and also in a separate memory block.

The ×8 SDDC-level Multi-ECC requires 12.5% redundancy; 8 bits on 64-bit data channel. FECC solely uses its 12.5% compression (448-bit target) to yield room for the redundancy without resorting to ECC DIMMs.

6.5 Evaluation

To understand the benefits of FECC and the tradeoffs involved, this section evaluates its reliability, performance impact, and energy efficiency in comparison to competing ECC schemes (including current commercial approaches as well as academic designs). First, it measures the protection coverage against chip failures to show FECC can maintain SDDC protection despite the reduced overheads. Then, the effectiveness of CoC is demonstrated to be both an excellent match for the goals of FECC and superior to previously published compression mechanisms in this context. Section 6.5.3 presents the significant energy and performance gains of FECC over existing approaches, and Section 6.5.4 shows the hardware cost of CoC is acceptable.

For the reliability, performance, and energy consumption evaluation, 14 ECC configurations are evaluated. The two commercial configurations are the S4SC-D4SD on a 144b channel (strong SDDC similar to [29]) and the AMD chipkill on a 72b channel, both with ×4 DRAM devices. Virtualized ECC [151] is evaluated with 136b and 128b channels for ×4 devices and 144b and 128b channels for ×8 devices and follows the configuration described in Section 6.6. The SDDC-level schemes (i.e., LOT-ECC [141] and Multi-ECC) are also evaluated. Finally, four FECC variants are evaluated: SDDC FECC+QPC and SDDC-level FECC+Multi-ECC on 64b channels with ×4 devices, SDDC FECC+OPC on a 72b channel with ×8 devices, and SDDC-level FECC+Multi-ECC on a 64b channel with ×8 devices. Note that the FECC configurations have the smallest number of redundant DRAM devices, with the exception of the 128b VECC configurations, though VECC is only defined for wide channels. Also note that the 72b FECC+OPC is, to the best of our knowledge, the only true SDDC ECC that can use both a 72b channel and ×8 DRAM devices.

For the compression coverage, performance impact, and energy consumption evaluation, the SPEC CPU 2006 [130, 39], PARSEC [3, 18], SPLASH2X[2] [147, 111], and NAS Parallel Benchmarks suites[3] are used.

6.5.1 Reliability

The reliability of SDDC and SDDC-level ECCs are evaluated using random error injection simulations (Section 3.4). Figure 6.7 and 6.8 show the rank-level and system-level reliability evaluation results, respectively. ECC flag protection is evaluated assuming blocks are fully-compressed/half-compressed/uncompressed with 70/20/10% probabilities, respectively. Due to the lack of decoding details, S4SC-D4SD is omitted from these results.

LOT-ECC and Multi-ECC are not true SDDC techniques and exhibit errors even with only single-chip errors (Section 6.6). LOT-ECC uses a 7-bit checksum at a chip granularity and thus exhibits a high SDC probability, which is particularly problematic for large systems (Figure 6.8). Multi-ECC with ×8 devices has a 2^{-16} probability of uncorrectable errors per codeword; with 2 codewords per 512b block, the probability of a DUE is 2^{-15}. Despite this problematic protection for single-device errors, the overall reliability result of Multi-ECC is very close to that of the other good SDDC-level ECCs, such as

[2] SPLASH2X is an update of the SPLASH2 benchmark suite [147] with larger input sets [111]. It is distributed as part of PARSEC v3.0 [3].

[3] While the majority of benchmarks are evaluated, some programs are omitted due to compilation issues or runtime errors.

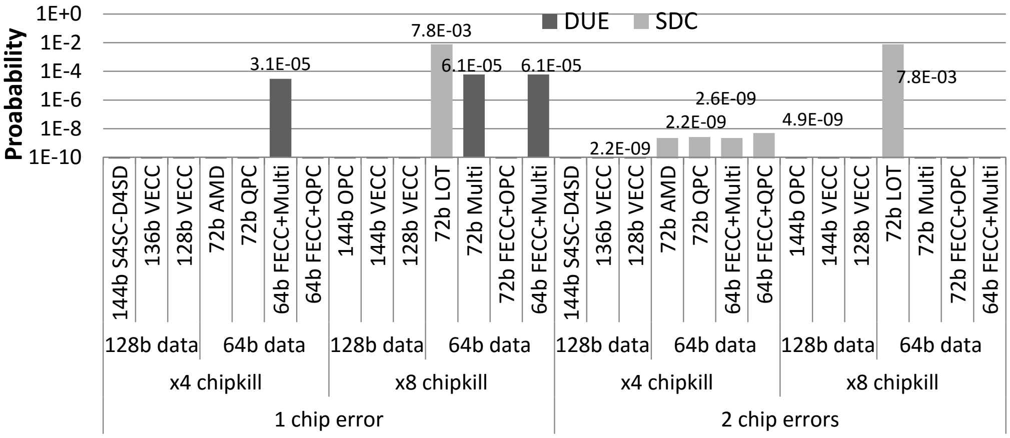


Figure 6.7: Probabilities of a DUE and SDC for a single rank with 1 or 2-chip errors.

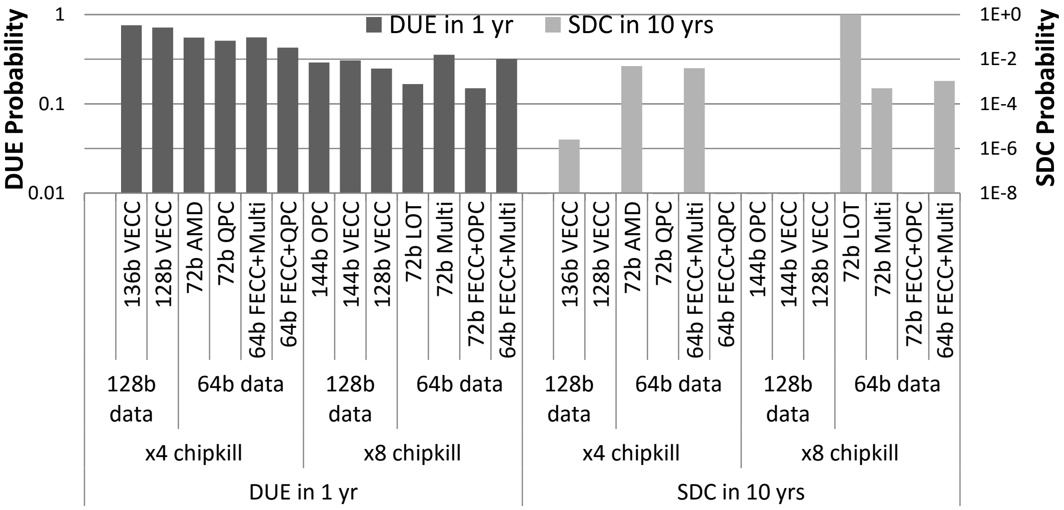


Figure 6.8: Probabilities of a DUE and SDC for a system with 100K DIMMs, assuming only 1 or 2-chip errors.

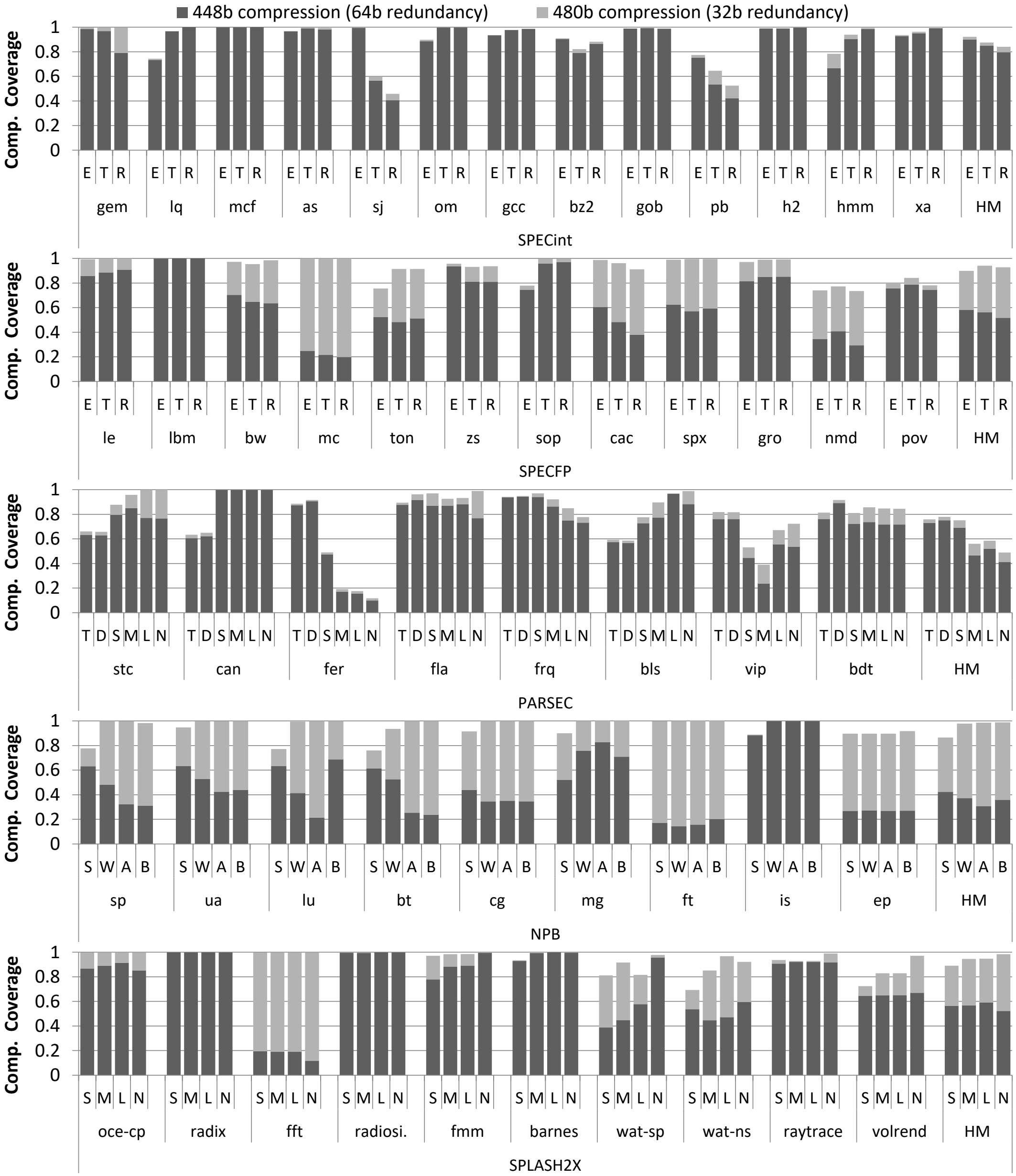
AMD chipkill and QPC, because the fault model does lead to some errors that span two or more chips. Importantly, FECC shows almost identical protection as the underlying ECC code it utilizes (i.e., QPC, OPC, or Multi×8), despite using fewer devices. In fact, the best overall protection in terms of both the DUE and SDC rates is provided by FECC+OPC, as it relies on fewer devices and therefore exhibits fewer faults, assuming only the faults reported by Sridharan and Liberty [128] can occur.

6.5.2 Compression Coverage

The compressibility of benchmark memory traffic and the effectiveness of coverage-oriented compression are evaluated using a Pin-based [92] cache model that implements CoC compression at the main memory interface. Serial program versions are used, with a weighted average over all program invocations taken in the case of fork-based multi-programmed benchmarks. Every main memory read and write-back is compressed[4] and its size tallied.

Figure 6.9 shows the overall compression coverage results. Because of the data dependence of CoC, the compression coverage experiments are conducted over a wide variety of benchmarks and input sets. Benchmarks are sorted within each suite by descending memory traffic and the harmonic mean behavior for each benchmark suite is given by a corresponding HM col-

[4] Stratified sampling [149] is used to reduce the experimental runtime of SPEC with the “ref” input set and PARSEC/SPLASH2X with the “native” input sets. Fast-forward, detailed warming, and execution periods of 16M, 2M, and 1M instructions are used, respectively.



E/T/R in SPECCPU stand for test/train/ref input set.
T/D/S/M/L/N in PARSEC stand for test/dev/small/medium/large/native input set.
S/W/A/B in NPB stand for small/workstation/A/B input set.
S/M/L/N in SPLASH2X stand for small/medium/large/native input set.

Figure 6.9: An evaluation on CoC compression coverage for 64-bit and 32-bit redundancy.

umn. It can be seen that CoC compresses most benchmarks well, with many benchmarks resulting in $\geq$ 99% compression coverage. In general, integer benchmarks compress sufficiently down to the 448b compression target, while many floating-point benchmarks must resort to the 480b level of compression. This is not unexpected, since the compressibility of floating-point numbers is limited (see Section 6.2.2 for more details).

The proper compression target varies according to the target ECC scheme. On a 64b rank with ×4 DRAM chips, FECC+Multi uses a 480b target coverage for both reads and writes and FECC+QPC uses a 448b target fully-compressed blocks and 480b for half-compressed blocks. On a 72b rank with ×8 DRAM chips, FECC+Multi uses a 448b target and FECC+OPC uses 448b and 512b targets full-compressed and half-compressed blocks, respectively. Note that Figure 6.9 gives results only on a 64b rank; compression coverage trends over a 72b rank are similar but they are not shown in the interest of brevity.

While most benchmarks and input sets compress satisfactorily with CoC, a few benchmarks show more lackluster compressibility. Ferret performs a content-based similarity search of images using JPEG images as query inputs. The inherent compressibility of these images is poor, perhaps leading to the low CoC coverage. Similarly, VIPS is an image processing system. Within PARSEC, the input images to VIPS are formatted in a native file format [4] and they happen to encode pixels in a non-IEEE754 floating-point format (using the “labq” encoding), which causes the lackluster coverage of CoC.

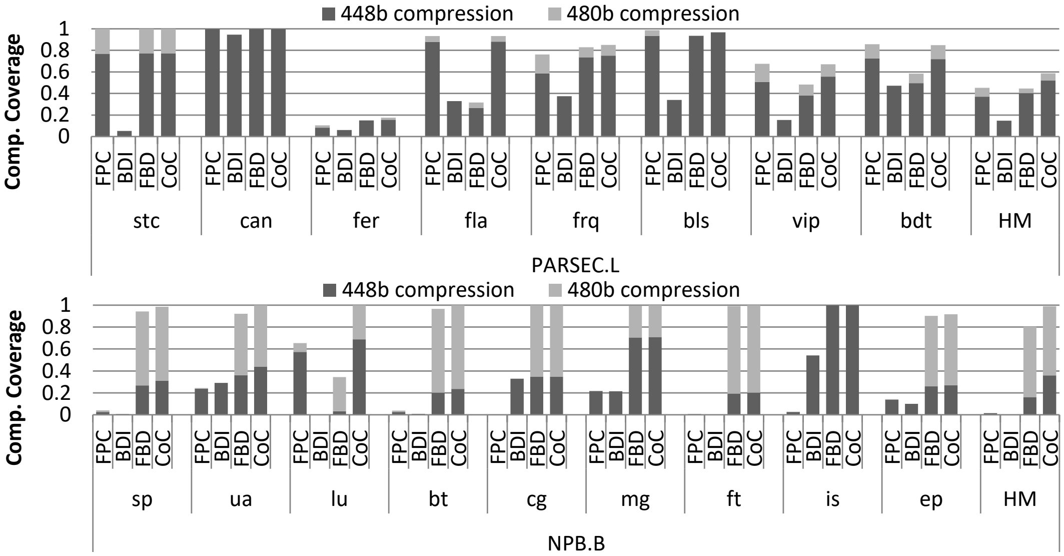


Figure 6.10: A compression coverage comparison between CoC and prior memory compression schemes.

Sjeng is an artificial intelligence program for chess. It makes heavy use of heterogeneous mixed-size data structures and it is expected that the data structures are not aligned or coalesced properly for CoC to fully capture the value locality within them. 64b-level compression does adapt somewhat to the heterogeneous nature of Sjeng—without it the compression coverage is dismal instead of lackluster, and enabling word-level compression increases the coverage by 5.54× and 6.25× at 448b and 480b, respectively. Perlbench is heavily multi-programmed through repeated forks; some of these program invocations are highly compressible and others compress poorly.

The average compressibility of programs in Figure 6.9 is high; this high compression coverage is a direct result of coverage-oriented compression scheme and it would not have been possible using prior fine-grained compres-

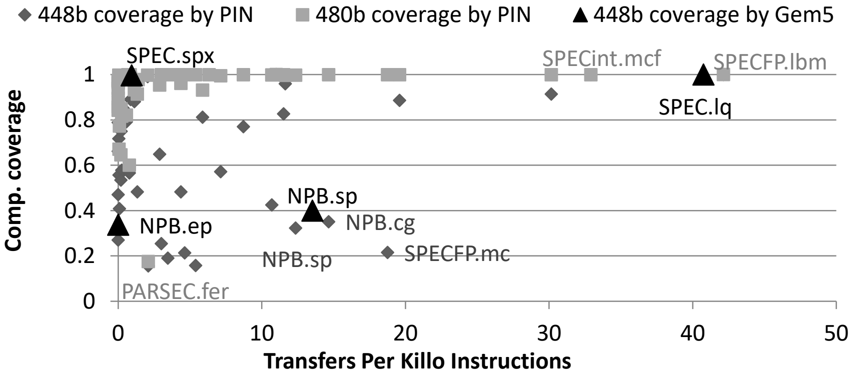


Figure 6.11: The memory traffic and compression coverage results of all benchmarks measured with Pin and the four representative detailed simulations selected.

sion approaches. Figure 6.10 demonstrates this through compression coverage experiments for PARSEC and NPB using BDI [113] and FPC [12] compression as well as fitting base + delta compression alone, which represents CoC without word-granularity compression. The results show that CoC always performs as well or better than the best of the other approaches, in some cases (like LU) outperforming all others. It is also readily apparent how insufficient BDI and FPC compression are for floating-point data—the floating-point compressors used for Frugal ECC are necessary for HPC benchmark suites such as NPB. Correspondingly, the compression approach used by Free ECC (combining BDI and FPC) [25] would not suffice for a large range of benchmark programs.

6.5.3 Performance and Energy

The performance impact and DRAM energy consumption of the different ECC schemes are estimated using the Gem5 simulator [1] and the Micron DDR3 power model [95]. Because of the long runtime of the simulator, not all the benchmarks used in compression analysis are evaluated for performance. Instead, four benchmarks are selected to represent different points in the space of parameter values that most-strongly impact the performance and energy behavior of FECC; off-chip data bandwidth (or last-level cache misses) and the frequency of compression exceptions. Figure 6.11 depicts how the benchmarks in coverage evaluation map across these two dimensions.

Sphinx from SPEC 2006 (denoted SPEC.spx) is chosen as representative of benchmarks with low traffic and high compression coverage, which are ideal for FECC. Libquantum from SPEC 2006 (SPECFP.lq) and EP from NPB (NPB.ep) are representative of benchmarks with high coverage / high traffic and low coverage / low traffic, respectively; both of which are expected to also perform well with FECC. As an example for an application that stresses FECC, SP from NPB (NPB.sp) is tested, which has poor compression coverage and also significant traffic. Note that these cases do not precisely match the extreme points observed with the Pin-based evaluation, but they are the closest ones found in detailed simulation.

The detailed single-core simulation parameters are summarized in Table 6.4. For FECC, CoC compression and decompression are assumed to have 2 memory cycle latency (Section 6.5.4), while ECC encoding and decoding

Core	2GHz OoO core, 16KB L1I, 64KB L1D		
LLC	2MB L2, 8-way (7-way for Multi-ECC) w/ stride prefetcher 20-cycle hit latency, 20 MSHR entries (40 for VECC, Multi-ECC, and FECC)		
DRAM	DDR3-1600, 2 rank, 2 channel for 64b data and 1 channel for 128b data		
		RD data FIFO lat.	WR data FIFO lat.
	AMD CK, S4SC-D4SD, VECC, and Multi-ECC	+1	+1
	QPC, OPC, and LOT-ECC	+4	+4
	FECC+Multi	+2	+3
	FECC+QPC and FECC+OPC	+4	+5
Simulation	400M cycles after 800M cycles of fast-forwarding		

Table 6.4: The simulation parameters for the performance and energy evaluation.

after the necessary data are fully available take 1 cycle. While multi-core simulations are more realistic and put more pressure on memory bandwidth, single-core simulations are used because FECC degrades performance mostly due to its extra latency and because single-core results allow for a clearer presentation of behavior and insights. The baseline IPC and DRAM energy are measured in the AMD chipkill configuration; the four benchmarks have 0.82 IPC / 0.23J for Libquantum, 1.33 IPC / 0.12J for Sphinx3, 1.24 IPC / 0.13J for NPB.ep, and 1.71 IPC / 0.10J for NPB.sp.

Figure 6.12 shows the estimated execution time and DRAM energy consumption over the chosen simulation interval of the selected benchmarks. Execution times are shown in bars (left y-axis) normalized to the AMD chipkill configuration and the relative energy is shown as lines (right y-axis). As expected, the FECC configurations exhibit essentially no impact on performance with the ideal SPEC.spx and also with the low-traffic NPB.ep.

The performance of SPEC.lq is impacted by type of code used and channel width, but as expected, the addition of FECC has a negligible impact ($< 1\%$) because of the high compression coverage. The only configuration for which FECC has a measurable impact is for NPB.sp, which has meaningful traffic and poor compression coverage; the 64b FECC+Multi configuration exhibits a 3.7% performance degradation because it requires a 480b level of compression for all reads and writes. FECC+OPC does not show a similar degradation because it relies on a 72b channel and does not add any latency for read operations, regardless of the compression achieved.

Energy results show that DRAM energy consumption varies more dramatically, from 1.98× down to 0.44×, among SDDC-level schemes. The main reason for the variation is the number of DRAM devices per access; the most energy inefficient configuration is 144b channel with ×8 devices (i.e., S4SC-D4SD with 36 devices) and the most efficient one is a 64b channel with ×8 devices (i.e., FECC+Multi-ECC with 8 devices). SPEC.lq with high data traffic and high locality benefits from the 128B cache lines and its reduced execution time lessens DRAM energy cost. FECC variants improves energy efficiency over their non-frugal counterparts: up to 11% in FECC+Multi-ECC. FECC+OPC is the only SDDC protection that that can use both a 72b channel and ×8 devices, and because of this it requires only about half the energy consumption of ECCs with competing levels of protection.

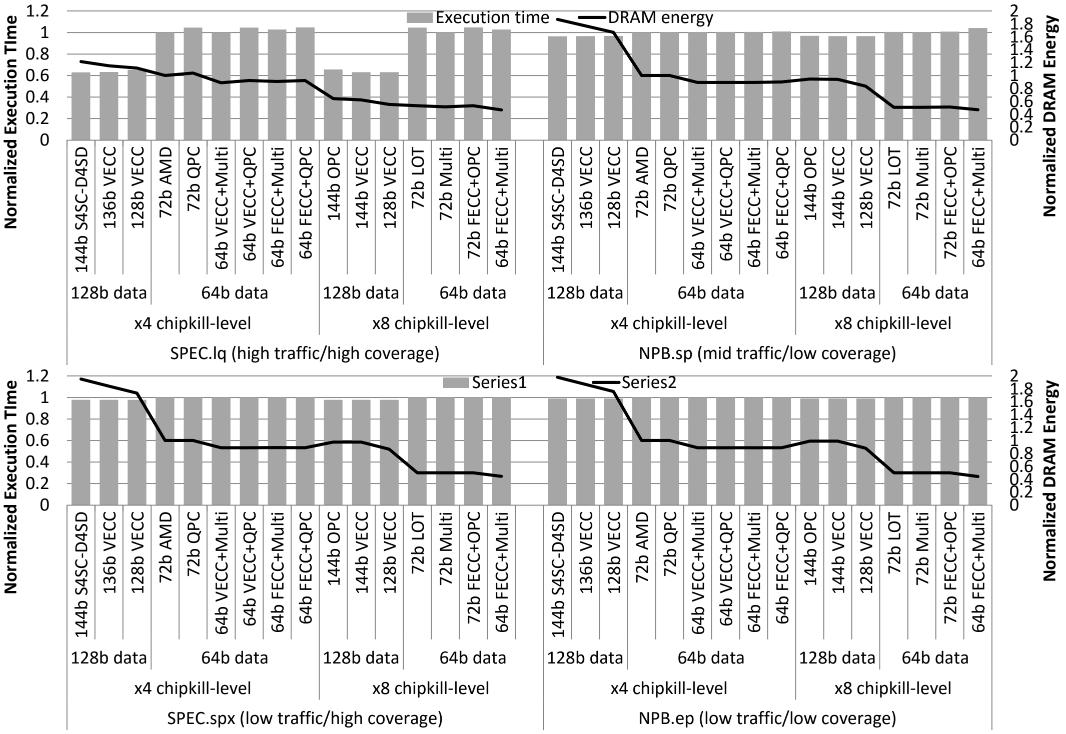


Figure 6.12: A comparison of execution time and DRAM energy consumption (normalized to AMD chipkill).

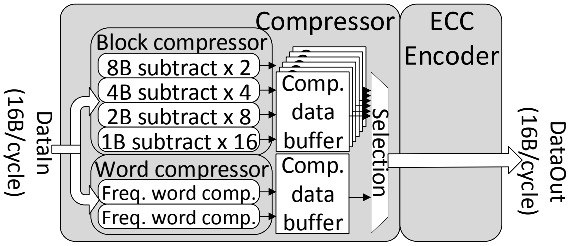


Figure 6.13: A block diagram of the CoC compressor.

6.5.4 Hardware Overheads

CoC compression requires some hardware to compress and decompress each memory transfer; yet they require a very small amount of chip area and are readily implementable in the 2-cycle latency assumed by the performance and energy evaluation. Figure 6.13 shows an overview of the compression process, broken into two stages. The first stage implements the block and word-granularity compressors through subtraction and equality testing. The output of this first-stage logic is the compressed data for each constituent compressor and the validity of each compression (whether it captures all of the data in its allotted space). This compressor and validity data is fed into a second stage that chooses the most appropriate compressor and outputs the final result.

The first-stage compressor logic is expected to consume the majority of the area for the CoC compressor. A straightforward, behavioral implementation of this first-stage logic easily achieves a 1ns latency. The block and word compressors consume areas equivalent to 4,383 and 5,866 NAND2 gates, respectively.[5] Combined, the block and word compressors consume about 25% more chip area than a single 32-bit fixed-point multiplier—an insignificant area compared to an entire processor.

The second compressor stage consists of a priority encoder with a small amount of ID generation logic, and is expected to easily and cheaply fit within the remaining 1ns of compression latency. The CoC decompressor is expected to consume less area than the compressor, because it does not need to speculatively perform all constituent compression schemes in parallel and can effectively use less internal arithmetic. Also, because decompression uses addition instead of subtraction for all delta operations, logic sharing between the different-width compression schemes should be a straightforward and potentially worthwhile optimization.

6.6 Related Work

This section reviews recent accomplishments in ECC and main memory compression that are closely related to FECC but not described above.

6.6.1 Efficient ECC

There is wealth of prior research that focuses on efficiency of memory protection. Recent ECC approaches attempt to improve reliability while stay-

[5] These delay and area estimates are found through standard-cell synthesis using the Synopsys toolchain and the 40nm TSMC standard cell library [135, 136] but are presented in a technology-independent manner.

ing within the bounds of acceptable system parameters (i.e., 64-bit memory data channels and near-12.5% redundant storage and bandwidth) by introducing new reliability and performance tradeoffs [28, 36, 151, 141, 55, 9]. One example is the use of ECCs that do not precisely match the definition of SDDC but still correct the vast majority of single chip errors [141, 55]; this class of protection is referred as *SDDC-level*. Another example is the use of multi-tiered codes in which a first ECC code is used for detection and a second code is used for correction [151, 141, 55]. This section lists some of the most important and efficient SDDC and SDDC-level ECC schemes with focuses on multi-tiered ECCs.

SDDC ECCs for ×4 DRAM Devices: *Virtualized ECC* (VECC) [151] provides SDDC protection on a wider variety of memory module organizations using its multi-tiered protection. VECC can provide ×4 SDDC protection on a 128-bit data interface (two non-ECC DIMMS), by virtualizing and storing 3 check symbols elsewhere in the memory space as data. Two separate accesses are required for each access to memory—one for data and the other for redundancy—though caching the ECC information is often effective and helps to reduce the performance impact of the additional memory traffic. Alternatively, VECC can separate the codes used for error detection and correction and can use a 136-bit data interface (built from one 72-bit ECC DIMM and one 64-bit non-ECC DIMM) to provide SDDC protection without needing two accesses in the common case. In this organization, VECC virtualizes a one-symbol error correcting code and stores it elsewhere in the memory space as data, but this secondary code needs only be accessed in the rare case when errors are detected (or upon a memory write, as it must be updated). Other approaches exist, similar to VECC, that embed ECC data elsewhere in the memory space to allow a more flexible use of non-ECC memory modules [28, 36].

SDDC ECCs for ×8 DRAM Devices: Wider ×8 DRAM chips save energy by activating fewer chips per memory access for a given data channel width. However, as a chip failure compromises a larger number of bits, ×8 SDDC is generally more challenging and requires more redundancy than its ×4 counterpart. To correct a ×8 chip error, the minimum redundancy is twice the correction size [125], giving a minimum of 25% redundancy for a 64-bit interface and 12.5% for a 128-bit interface.

There are, however, recent academic approaches that target SDDC-level protection on a narrow channel with ×8 devices and 12.5% redundancy. LOT-ECC [141] can correct most single chip errors ($\frac{127}{128}$ or 99.2%) on a 72-bit ECC DIMM by using a 4-tiered ECC scheme. Due to its use of a 7-bit error detecting checksum, however, $\frac{1}{128}$ of chip errors remain undetected by LOT-ECC and lead to silent data corruption. Multi-ECC [55] uses an error localization and erasure code to enable SDDC-level protection on a 72-bit ECC DIMM with 12.9% redundancy. The error localization procedure of Multi-ECC is based on a 16-bit checksum, and it fails to identify the faulty symbol with

2^{-16} probability, making 2^{-16} of single chip errors detectable-yet-uncorrectable. E-ECC [24] relies on error and erasure correction to correct most ×8 device failures on a single ECC DIMM.

6.6.2 Memory Compression

Memory compression has been actively researched and deployed in caches and main memory to increase the available memory capacity and reduce off-chip traffic. Frequent value compression [150] targets a small set of 2–8 frequent values that collectively occupy over 50% of memory entries in some benchmarks. It replaces each frequent value by a small index to reduce space.

Frequent pattern compression [12] targets frequent patterns rather than frequent values. Most of their patterns are based on the fact that large data types (e.g., 32 or 64-bit fixed-point numbers) often contain small values that do not fully utilize their allocated storage. Frequent pattern compression saves storage by opportunistically converting such values to use smaller data types.

IBM MXT technology [6] is a word-granularity compressor that relies on a derivative of adaptive dictionary based coding [156] to more than double the effective main memory capacity. However, the drawbacks of MXT include a large cache line size (1KB) to amortize its significant dictionary cost, long compression/decompression latencies from sequential processing and additional accesses to locate the compressed data.

Linearly compressed pages (LCP) [112] compresses all cache lines within a page to the same size, making address calculation straightforward and avoiding memory fragmentation. Upon a compression exception, LCP allocates separate regions within the physical page for exception flags and exception data. If the number of exceptions overflows this allocated storage, LCP traps to operating system and requests a bigger physical page for the virtual page.

MemZip [123] compresses cache lines for memory traffic reduction, not for capacity savings. The compressed data are stored in the same footprint as uncompressed data but memory traffic and energy are reduced on a memory channel that supports fine-grained rank subsetting. Space savings can also be used to store meta-data, such as data bus inversion for memory interface energy saving or ECC for opportunistically stronger protection (though the potential for ECC is largely unexplored). Other work also employs memory compression to store optional prefetch hints [89].

Free ECC [25] combines compression and ECC for the last level cache. Free ECC depends on customized tags to store compression meta-data, however, and it combines BDI and FPC compression. The lack of dedicated meta-data storage makes this approach inappropriate for DRAM; Section 6.5 also shows the ineffectiveness of BDI and FPC alone for a wide range of benchmark suites.

COP [109] uses compression to enable ECC protection using non-ECC DIMMs. The specifics of the COP approach make its level of error protection quite weak—significantly worse than SEC-DED—and preclude it from being a viable alternative to FECC for high-performance and high-availability systems. The COP ECC scheme cannot be extended in a straightforward manner to SDDC levels of protection, even in its strongest organization, because the implicit manner in which it tracks the compressibility of a memory block degrades the coverage of its ECC code. The compression scheme used by COP also targets 6.25% redundancy and it provides poor compression coverage for the industry-standard 12.5% ECC footprint. There is also a patent that describes the use of compression with non-ECC DIMMs [132] without any evaluation or consideration of SDDC-level protection.

Recent research on compression algorithms [76, 101, 13] has demonstrated higher compression ratios than earlier compression work. FECC's unique CoC scheme may utilize the insights from this recent research to improve compression coverage or reduce compressed data footprint further. In turn, this may enable even more energy-efficient memory configurations (e.g., true ×8 SDDC using non-ECC DIMM) while amortizing compression overheads with storage and bandwidth reductions.

6.7 Summary

Frugal ECC provides new tradeoffs between reliability, performance, and efficiency. By combining a novel compression scheme and meta-data encoding and management techniques, FECC designs are able to match the memory access characteristics of conventional ECC designs while requiring fewer, or even no, redundant DRAM devices. The coverage-oriented compression technique is carefully co-designed with its intended use and far exceeds the capabilities of previously-published fine-grained compression mechanisms, especially for floating-point intensive programs. The ECC layout can provide flexible tradeoffs between energy efficiency and performance without degrading levels of reliability.

Chapter 7

Conclusion

This dissertation presents and evaluates a set of novel memory protection mechanisms. The mechanisms, taken together, may provide thorough end-to-end protection of DRAM against existing and emerging types of errors, while the individual mechanisms are demonstrated to be stronger and safer than their current state-of-the-art counterparts. Despite their strength, the proposed mechanisms also improve energy efficiency by reducing redundancy, eliminating inefficient and separate protection and/or enabling more energy-efficient configurations. These mechanisms achieve such reliability benefits with minimal changes to DRAM design and no extra storage and bandwidth overheads compared to current schemes, making them readily implementable to influence industry and to change the world.

In Chapter 3, this dissertation develops a new data ECC mechanism, *Bamboo ECC*, that can detect most uncorrectable errors and which may allow future large-scale systems to operate safely without silent data corruptions. Its strong level of error correction and support for flexible graceful downgrade schemes can also greatly extend the lifetime of a system by tolerating permanent faults.

In Chapter 4, this dissertation provides a novel mechanism against emerging errors from imperfect process scaling. By combining ECC error and erasure decoding with fault remapping, AG-ECC preserves SDDC reliability in the presence of frequent inherent faults, while improving manufacturability without separate in-DRAM ECC protection. Its high reliability, high manufacturability and high efficiency may allow DRAM process technology to continue shrinking despite increasing cell fault rates without requiring modifications to DRAM chips themselves.

In Chapter 5, this dissertation presents ECC mechanisms against emerging transmission errors to provide holistic system-level protection. Transmission errors on the clock, control, command, and address signals are expected to grow as DRAM transfer rates increase and I/O voltage levels decrease; these CCCA errors can dominate system reliability after data is strongly protected by an aggressive ECC. AI-ECC is the first approach to provide strong and thorough protection against all aspects of this important error type. By augmenting existing data protection schemes, AI-ECC does not cost additional storage and bandwidth overheads and demands minimal changes to DRAM design. This strong protection may enable more aggressive speedups in DRAM signal transfers for future throughput-oriented computations.

In Chapter 6, this dissertation improves the energy efficiency of DRAM protection through fine-grained compression. Frugal ECC can decouple the amount of redundancy needed for protection and the number of chips needed for the redundancy, allowing for versatile and more energy-efficient memory configurations. With a carefully co-designed compression algorithm and robust ECC layouts, Frugal ECC provides flexible tradeoffs between energy-efficiency and performance without degrading the level of error protection.

This dissertation also contributes with its rigorous evaluation of reliability and manufacturability, including the practical impact of ECC detection coverage, consequences from CCCA errors, manufacturability from inherent faults, and protection coverage of SDDC-level ECCs.

7.1 Future Research Directions

The key findings of this dissertation can be extended to other memory products and other compute architectures. Some of the most exciting research opportunities are presented below.

AG-ECC based on In-DRAM Redundancy: AG-ECC eliminates the need for a separate in-DRAM ECC by utilizing rank-level ECC against both inherent and operational faults. For systems that do not employ similar schemes and systems with low reliability requirements (e.g., personal computers without rank-level ECC), however, DRAM vendors may produce devices with in-DRAM ECC only, or sell them at a cheaper price due to economies of scale.

In such cases, the proposed AG-ECC approach can be redesigned to rely on the extra cells of in-DRAM ECC, instead of extra chips on the DIMM for redundancy. With slight changes to the DRAM core and interface, the extra cells can be optionally exposed and utilized as rank-level ECC redundancy. Such an idea is promising, as rank-level ECC can provide SDDC-level protection with fewer or even no redundant chips. The biggest challenge of this research opportunity is how to deal with the increased level of error severity; now a dead ×4 chip generates not only up to 32 erroneous data bits but also up to 4-bit errors in the redundant information.

Protection of Memory with Wider Interface: Research should target the future, not the present. This dissertation uses DDR4 because it is a concrete and well-understood example for which good models exist; a requirement for a thorough and convincing evaluation. Other architectures, however, employ different lines of DRAM products (graphics DDR [61, 66], low-power DDR [57, 63, 64, 67], 3D-stacked memories [62, 58, 65, 49], etc.) to meet their computation throughput and energy efficiency requirements.

These DRAMs commonly utilize a wider data interface (16- or 32-bit) than DDR4 (4- or 8-bit) to reduce the number of chips per access and to improve energy efficiency at the cost of capacity and reliability. The current practice of using weak SEC-DED [106] provides limited reliability improvements, and it is very important to develop protection mechanisms as reliable as DDR4 SDDC protection to enable reliable computation using these devices. While ECC alone may be insufficient due to the large number of bit errors from a chip fault, the safe detection capability of Bamboo ECC may allow complementary protection schemes to achieve high reliability.

Bibliography

[1] The gem5 Simulator System: A Modular Platform for Computer System Architecture Research. `http://www.gem5.org`.

[2] The Nas Parallel Benchmarks. `http://www.nas.nasa.gov/publications/npb.html`.

[3] The Princeton Application Repository for Shared-Memory Computers (PARSEC). `http://parsec.cs.princeton.edu`.

[4] The VIPS file format. `http://www.vips.ecs.soton.ac.uk/supported/current/doc/html/libvips/file-format.html`.

[5] Intel Xeon Processor E5 and E7 v3 Family Uncore Performance Monitoring Reference Manual. Technical Report 331051-002, Intel Corporation, 2015.

[6] B. Abali, H. Franke, D. E. Poff, R. A. Saccone, C. O. Schulz, L. M. Herger, and T. B. Smith. Memory Expansion Technology (MXT): Software Support and Performance. *IBM Journal of Research and Development*, 45(2):287–301, March 2001.

[7] David G Abdoo and J David Cabello. Error correction system for N bits using error correcting code designed for fewer than N bits. US Patent, US 5,490,155, February 6 1996.

[8] Advanced Micro Devices (AMD), Inc. Kernel developers guide for AMD NPT family 0Fh processors. Technical report, 2007.

[9] Advanced Micro Devices (AMD), Inc. BIOS and Kernel Developers Guide (BKDG) for AMD Family 15h Models 00h-0Fh Processors, Jan 2013.

[10] Jung Ho Ahn, Mattan Erez, and William J. Dally. The Design Space of Data-Parallel Memory Systems. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC)*, pages 2–2, Nov 2006.

[11] Alaa R. Alameldeen, Ilya Wagner, Zeshan Chishti, Wei Wu, Chris Wilkerson, and Shih-Lien Lu. Energy-efficient Cache Design Using Variable-strength Error-correcting Codes. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, ISCA '11, pages 461–472, 2011.

[12] Alaa R. Alameldeen and David A. Wood. Frequent Pattern Compression: A Significance-Based Compression Scheme for L2 Caches. Technical report, Technical Report 1500, Computer Sciences Department, University of Wisconsin-Madison, 2004.

[13] Angelos Arelakis, Fredrik Dahlgren, and Per Stenstrom. HyComp: A Hybrid Cache Compression Method for Selection of Data-type-specific Compression Methods. In *Proceedings of the International Symposium on Microarchitecture (MICRO)*, MICRO-48, pages 38–49, 2015.

[14] K. Arimoto, Y. Matsuda, K. Furutani, M. Tsukude, T. Ooishi, K. Mashiko, and K. Fujishima. A speed-enhanced DRAM array architecture with embedded ECC. *IEEE Journal of Solid State Circuits (JSSC)*, 25(1):11–17, 1990.

[15] A. Avizienis, J. C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33, Jan 2004.

[16] A. Bacchini, M. Rovatti, G. Furano, and M. Ottavi. Characterization of data retention faults in DRAM devices. In *Proceedings of the International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT)*, pages 9–14, Oct 2014.

[17] C. Bienia and K. Li. Fidelity and scaling of the PARSEC benchmark inputs. In *Proceedings of the International Symposium on Workload Characterization (IISWC)*, 2010.

[18] Christian Bienia. *Benchmarking Modern Multiprocessors.* PhD thesis, Princeton University, January 2011.

[19] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R Hower, Tushar Krishna, Somayeh Sardashti, et al. The gem5 Simulator. *ACM SIGARCH Computer Architecture News*, 39(2):1–7, 2011.

[20] R. C. Bose and D. K. Ray-Chaudhuri. On a class of error correcting binary group codes. *Information and Control*, 3(1):68–79, March 1960.

[21] Mark T. Chapman. Introducing IBM Enterprise X-Architecture Technology. Technical report, 2001.

[22] Chin-Long Chen, Mu-Yue Hsiao, Patrick J. Meaney, and William Wu Shen. Detecting address faults in an ECC-protected memory. US Patent, US 6,457,154 B1, Sep 2002.

[23] Chin-Long Chen, R. Brett Tremaine, and Michael E. Wazlowski. (146,130) error correction code utilizing address information. US Patent, US 6,751,769 B2, Jun 2004.

[24] Hsing-Min Chen, Akhil Arunkumar, Carole-Jean Wu, Trevor Mudge, and Chaitali Chakrabarti. E-ECC: Low Power Erasure and Error Correction Schemes for Increasing Reliability of Commodity DRAM Systems. In *Proceedings of the International Symposium on Memory Systems (MEMSYS)*, pages 60–70, 2015.

[25] Long Chen, Yanan Cao, and Zhao Zhang. Free ECC: An efficient error protection for compressed last-level caches. In *IEEE International Conference on Computer Design (ICCD)*, 2013.

[26] Chang-Seok Choi and Hanho Lee. High throughput four-parallel RS decoder architecture for 60GHz mmWAVE WPAN systems. In *Pro-*

ceedings of the IEEE International Conference of Consumer Eletronics (ICCE), pages 225–226, Jan 2010.

[27] Jinsuk Chung, Ikhwan Lee, Michael Sullivan, Jee Ho Ryoo, Dong Wan Kim, Doe Hyun Yoon, Larry Kaplan, and Mattan Erez. Containment domains: A scalable, efficient, and flexible resilience scheme for exascale systems. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC)*, pages 58:1–58:11, Nov 2012.

[28] Radoslav Danilak. Transparent error correction code memory system and method. US Patent, US 7,117,421 B1, October 2006.

[29] T. J. Dell. A White Paper on the Benefits of Chipkill-Correct ECC for PC Server Main Memory. *IBM Microelectronics Division*, November 1997.

[30] Michael Demshki and Robert Shiveley. Advanced Reliability for Intel Xeon Processor-based Servers, 2010.

[31] Khaled Fekih-Romdhane. Illegal commands handling at the command decoder stage. US Patent, US 20070245036 A1, Oct 2007.

[32] Fujitsu Technology Solutions. White Paper: FUJITSU Server PRIMERGY & PRIMEQUEST Memory performance of Xeon E7-8800 / 4800 v2 (Ivy Bridge-EX) based systems. `http://globalsp.ts.fujitsu.com/dmsp/`

Publications/public/wp-ivy-bridge-ex-memory-performance-ww-en.pdf, 2014.

[33] Bharan Giridhar, Michael Cieslak, Deepankar Duggal, Ronald Dreslinski, Hsing Min Chen, Robert Patti, Betina Hold, Chaitali Chakrabarti, Trevor Mudge, and David Blaauw. Exploring DRAM Organizations for Energy-efficient and Resilient Exascale Memories. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC)*, pages 23:1–23:12, 2013.

[34] Richard Henry Gumpertz. *Error Detection with Memory Tags.* PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 1981.

[35] Richard Henry Gumpertz. Combining Tags with Error Codes. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, pages 160–165, 1983.

[36] Michael John Haertel, R. Stephen Polzin, Andrej Kocev, and Maurice Bennet Steinman. ECC implementation in non-ECC components. US Patent, US 8,135,935 B2, March 2012.

[37] R. W. Hamming. Error Detecting and Error Correcting Codes. *Bell System Technical Journal*, 29(2):147–160, 1950.

[38] Paul H Hargrove and Jason C Duell. Berkeley lab checkpoint/restart (BLCR) for Linux clusters. *Journal of Physics: Conference Series*, 46(1):494, 2006.

[39] John L. Henning. SPEC CPU2006 Benchmark Descriptions. *ACM SIGARCH Computer Architecture News*, 34(4):1–17, September 2006.

[40] Hewlett-Packard. Servers and Storage Technology for the Adaptive Infrastructure, 2006.

[41] Hewlett-Packard. HP Advanced Memory Error Detection Technology, 2011.

[42] Hewlett-Packard. How memory RAS technologies can enhance the uptime of HP ProLiant servers, 2013.

[43] High Productivity Computing Systems (HPCS). HPC challenge Benchmarks. `http://icl.cs.utk.edu/hpcc/hpcc_results.cgi`.

[44] A. Hocquenghem. Codes Correcteurs d'Erreurs. *Chiffres (Paris)*, 2:147–156, September 1959.

[45] S. Hong. Memory technology trend and future challenges. In *IEEE International Electron Devices Meeting*, pages 12.4.1–12.4.4, Dec 2010.

[46] David Huffman. A Method for the Construction of Minimum-Redundancy Codes. *Proceedings of the IRE*, 1952.

[47] Jaehyuk Huh, Doug Burger, and Stephen W Keckler. Exploring the design space of future CMPs. In *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques (PACT)*, pages 199–210, 2001.

[48] Andy A. Hwang, Ioan A. Stefanovici, and Bianca Schroeder. Cosmic Rays Don't Strike Twice: Understanding the Nature of DRAM Errors and the Implications for System Design. In *Proceedings of the International Symposium on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 111–122, 2012.

[49] Hybrid Memory Cube Consortium. *Hybrid Memory Cube Specification 2.0*, 2014.

[50] IBM. Chipkill Memory. Technical report, 2012.

[51] IBM, Inc. IBM Enterprise X-Architecture Technology: Reaching the Summit. Technical report, 2002.

[52] IEEE Task P754. *IEEE 754-1985, Standard for Binary Floating-Point Arithmetic*. IEEE, 1985.

[53] Intel corp. Intel Xeon Processor E7 Family: Reliability, Availability, and Serviceability, 2011.

[54] Xun Jian, Nathan DeBardeleben, Sean Blanchard, Vilas Sridharan, and Rakesh Kumar. Analyzing Reliability of Memory Sub-systems with Double-Chipkill Detect/Correct. In *Proceedings of IEEE Pacific Rim International Symposium on Dependable Computing*, pages 88–97. IEEE, 2013.

[55] Xun Jian, Henry Duwe, John Sartori, Vilas Sridharan, and Rakesh Kumar. Low-power, low-storage-overhead chipkill correct via multi-line er-

ror correction. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC)*, pages 1–12, Nov 2013.

[56] Joint Electron Device Engineering Council. *DDR2 SDRAM Specification, JESD79-2F*, Nov. 2009.

[57] Joint Electron Device Engineering Council. *Low Power Double Data Rate (LPDDR) SDRAM Standard, JESD209B*, Feb. 2010.

[58] Joint Electron Device Engineering Council. *Wide I/O Single Data Rate (Wide I/O SDR), JESD229*, Dec. 2011.

[59] Joint Electron Device Engineering Council. *DDR3 SDRAM STANDARD, JESD79-3F*, July 2012.

[60] Joint Electron Device Engineering Council. *DDR4 SDRAM STANDARD, JESD79-4*, Sep. 2012.

[61] Joint Electron Device Engineering Council. *Graphics Double Data Rate (GDDR5) SGRAM Standard, JESD212B.01*, Dec. 2013.

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

[63] Joint Electron Device Engineering Council. *Low Power Double Data Rate 2 (LPDDR2), JESD209-2F*, June 2013.

[64] Joint Electron Device Engineering Council. *Low Power Double Data Rate 3 (LPDDR3), JESD209-3B*, Aug. 2013.

[65] Joint Electron Device Engineering Council. *Wide I/O 2, JESD229-2*, Aug. 2014.

[66] Joint Electron Device Engineering Council. *Graphics Double Data Rate (GDDR5X) SGRAM Standard, JESD232*, Nov. 2015.

[67] Joint Electron Device Engineering Council. *Low Power Double Data Rate 4 (LPDDR4), JESD209-4A*, Nov. 2015.

[68] H. L. Kalter, C. H. Stapper, J. E. Barth, J. DiLorenzo, C. E. Drake, J. A. Fifield, G. A. Kelley, S. C. Lewis, W. B. van der Hoeven, and J. A. Yankosky. A 50-ns 16-Mb DRAM with a 10-ns data rate and on-chip ECC. *IEEE Journal of Solid State Circuits (JSSC)*, 25(5):1118–1128, 1990.

[69] H. J. Kang and I. C. Park. A high-speed and low-latency Reed-Solomon decoder based on a dual-line structure. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, volume 3, pages III–3180–III–3183, 2002.

[70] Uksong Kang, Hak soo Yu, Churoo Park, Hongzhong Zheng, John Halbert, Kuljit Bains, SeongJin Jang, and Joo Sun Choi. Co-Architecting Controllers and DRAM to Enhance DRAM Process Scaling. In *Proceedings of the Memory Forum*, 2014.

[71] Y. Katayama, Y. Negishi, and S. Morioka. Efficient error correction code configurations for quasi-nonvolatile data retention by DRAMs. In *Proceedings of the International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT)*, pages 201–209, 2000.

[72] Samira Khan, Donghyuk Lee, Yoongu Kim, Alaa R. Alameldeen, Chris Wilkerson, and Onur Mutlu. The Efficacy of Error Mitigation Techniques for DRAM Retention Failures: A Comparative Experimental Study. In *Proceedings of the International Joint Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, pages 519–532, 2014.

[73] Dae-Hyun Kim and Linda S. Milor. ECC-ASPIRIN: An ECC-assisted post-package repair scheme for aging errors in DRAMs. In *Proceedings of the VLSI Test Symposium (VTS)*, pages 1–6, 2016.

[74] Dong Wan Kim and Mattan Erez. Balancing reliability, cost, and performance tradeoffs with FreeFault. In *Proceedings of the International Symposium on High Performance Computer Architecture (HPCA)*, pages 439–450, Feb 2015.

[75] Dong Wan Kim and Mattan Erez. RelaxFault Memory Repair. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, 2016.

[76] Jungrae Kim, Michael Sullivan, Esha Choukse, and Mattan Erez. Bit-Plane Compression: Transforming Data for Better Compression in Many-

core Architectures. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, 2016.

[77] Jungrae Kim, Michael Sullivan, and Mattan Erez. Bamboo ECC: Strong, safe, and flexible codes for reliable computer memory. In *Proceedings of the International Symposium on High Performance Computer Architecture (HPCA)*, pages 101–112, 2015.

[78] Jungrae Kim, Michael Sullivan, Seong-Lyong Gong, and Mattan Erez. Frugal ECC: Efficient and Versatile Memory Error Protection Through Fine-grained Compression. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC)*, pages 12:1–12:12, 2015.

[79] Jungrae Kim, Michael Sullivan, Sangkug Lym, and Mattan Erez. All-Inclusive ECC: Thorough End-to-End Protection for Reliable Computer Memory. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, 2016.

[80] Jungrae Kim, Michael Sullivan, Sangkug Lym, Dong Wan Kim, Seong-Lyong Gong, and Mattan Erez. Active Guardband ECC for Strong and Efficient Reliability and Manufacturability. In *Submission to HPCA*, 2017.

[81] Saeng-Hwan Kim, Won-Oh Lee, Jung-Ho Kim, Seong-Seop Lee, Sun-Young Hwang, Chang-Il Kim, Tae-Woo Kwon, Bong-Seok Han, Sung-Kwon Cho, Dae-Hui Kim, Jae-Keun Hong, Min-Yung Lee, Sung-Wook

Yin, Hyeon-Gon Kim, Jin-Hong Ahn, Yong-Tark Kim, Yo-Hwan Koh, and Joong-Sik Kih. A low power and highly reliable 400Mbps mobile DDR SDRAM with on-chip distributed ECC. pages 34–37, 2007.

[82] Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu. Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, pages 361–372, 2014.

[83] Yoongu Kim, Vivek Seshadri, Donghyuk Lee, Jamie Liu, and Onur Mutlu. A Case for Exploiting Subarray-level Parallelism (SALP) in DRAM. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, pages 368–379, 2012.

[84] Masahiro Kuramoto, Masao Koyabu, Jun Tsuiki, and Junichi Inagaki. Storage control circuit, and method for address error check in the storage control circuit. US Patent, US 7,555,699 B2, Jun 2009.

[85] Sanghyuk Kwon, Young Hoon Son, and Jung Ho Ahn. Understanding DDR4 in pursuit of In-DRAM ECC. In *SoC Design Conference (ISOCC), 2014 International*, pages 276–277, Nov 2014.

[86] Charles R. Lefurgy, Alan J. Drake, Michael S. Floyd, Malcolm S. Allen-Ware, Bishop Brock, Jose A. Tierno, and John B. Carter. Active Management of Timing Guardband to Save Energy in POWER7. In *Pro-*

ceedings of the International Symposium on Microarchitecture (MICRO), pages 1–11, 2011.

[87] Xin Li, Michael C. Huang, Kai Shen, and Lingkun Chu. A Realistic Evaluation of Memory Hardware Errors and Software System Susceptibility. In *Proceedings of the USENIX Annual Technical Conference (USENIX)*, 2010.

[88] Chung-Hsiang Lin, De-Yu Shen, Yi-Jung Chen, Chia-Lin Yang, and M. Wang. SECRET: Selective error correction for refresh energy reduction in DRAMs. In *IEEE International Conference on Computer Design (ICCD)*, pages 67–74, 2012.

[89] Gang Liu, Jih-Kwon Peir, and V. Lee. Miss-Correlation Folding: Encoding Per-Block Miss Correlations in Compressed DRAM for Data Prefetching. In *Proceedings of the International Symposium on Parallel and Distributed Processing (IPDPS)*, pages 691–702, 2012.

[90] David Locklear. Chipkill Correct Memory Architecture. Technical report, 2000.

[91] S. K. Lu, C. J. Tsai, and M. Hashizume. Integration of Hard Repair Techniques with ECC for Enhancing Fabrication Yield and Reliability of Embedded Memories. pages 49–54, 2015.

[92] Chi-Keung Luk, Robert Cohn, Robert Muth, Harish Patil, Artur Klauser, Geoff Lowney, Steven Wallace, Vijay Janapa Reddi, and Kim Hazel-

wood. Pin: Building Customized Program Analysis Tools with Dynamic Instrumentation. In *Proceedings of the International Symposium on Programming Languages Design and Implementation (PLDI)*, 2005.

[93] C. D. Martino, Z. Kalbarczyk, R. K. Iyer, F. Baccanico, J. Fullop, and W. Kramer. Lessons Learned from the Analysis of System Failures at Petascale: The Case of Blue Waters. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN)*, pages 610–621, June 2014.

[94] Paul Messina and Stephen Lee. Exascale Computing Project (ECP) Update. `http://science.energy.gov/~/media/ascr/ascac/pdf/meetings/201604/ECP_ASCAC_Overview_final.pdf`, April 2016.

[95] Micron Technology Co. Calculating Memory System Power for DDR3. `http://www.micron.com/~/media/documents/products/technical-note/dram/tn41_01ddr3_power.pdf`, 2007.

[96] Micron Technology Co. 8Gb DDR4 SDRAM datasheet. `https:/www.micron.com/~/media/documents/products/data-sheet/dram/ddr4/8gb_ddr4_sdram.pdf`, 2015.

[97] Sparsh Mittal. A Survey of Architectural Techniques for DRAM Power Management. *International Journal of High Performance System Architecture*, 4(2):110–119, December 2012.

[98] William C. Moyer. Selective masking for error correction. US Patent, US 8,990,657 B2, Mar 2015.

[99] Prashant J Nair, Dae-Hyun Kim, and Moinuddin K Qureshi. ArchShield: Architectural Framework for Assisting Dram Scaling by Tolerating High Error Rates. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, 2013.

[100] Prashant J Nair, Vilas Sridharan, and Moinuddin K Qureshi. XED: Exposing On-Die Error Detection Information for Strong Memory Reliability. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, 2016.

[101] Tri M. Nguyen and David Wentzlaff. MORC: A Manycore-oriented Compressed Cache. In *Proceedings of the International Symposium on Microarchitecture (MICRO)*, MICRO-48, pages 76–88, 2015.

[102] Richard Nicholas. Address error detection. US Patent, US 8,949,694 B2, Feb 2015.

[103] Michael Nicolaidis, Thierry Bonnoit, and Nacer-Eddine Zergainoh. Eliminating speed penalty in ECC protected memories. In *Proceedings of Design, Automation, and Test in Europe (DATE)*, pages 1–6, 2011.

[104] Kevin B. Normoyle. Address error detection by merging a polynomial-based CRC code of address bits with two nibbles of data or data ECC bits. US Patent, US 7,203,890 B1, Apr 2007.

[105] Kevin B. Normoyle and Robert G. Hathaway. Encoding 64-bit data nibble error correct and cyclic-redundancy code (CRC) address error detect for use on a 76-bit memory module. US Patent, US 7,398,449 B1, Jul 2008.

[106] NVIDIA Corp. NVIDIA's Next Generation CUDA Compute Architecture: Kepler GK110/210. `http://images.nvidia.com/content/pdf/tesla/NVIDIA-Kepler-GK110-GK210-Architecture-Whitepaper.pdf`, 2014.

[107] Tae-Young Oh, Hoeju Chung, Jun-Young Park, Ki-Won Lee, Seunghoon Oh, Su-Yeon Doo, Hyoung-Joo Kim, ChangYong Lee, Hye-Ran Kim, Jong-Ho Lee, Jin-Il Lee, Kyung-Soo Ha, YoungRyeol Choi, Young-Chul Cho, Yong-Cheol Bae, Taeseong Jang, Chulsung Park, Kwangil Park, SeongJin Jang, and Joo Sun Choi. A 3.2 Gbps/pin 8 Gbit 1.0 V LPDDR4 SDRAM With Integrated ECC Engine for Sub-1 V DRAM Core Operation. *IEEE Journal of Solid State Circuits (JSSC)*, 50(1):178–190, 2015.

[108] Oracle, Inc. Oracle SPARC Server RAS Comparison.

[109] David J. Palframan, Nam Sung Kim, and Mikko H. Lipasti. COP: To Compress and Protect Main Memory. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, pages 682–693, 2015.

[110] Ward D. Parkinson and Edward J. Heitzeberg. Parity and error correction coding on integrated circuit addresses. US Patent, US 5,173,905 A, Dec 1992.

[111] PARSEC Group. A Memo on Exploration of SPLASH-2 Input Sets, 2011.

[112] Gennady Pekhimenko, Vivek Seshadri, Yoongu Kim, Hongyi Xin, Onur Mutlu, Michael A. Kozuch, Phillip B. Gibbons, and Todd C. Mowry. Linearly Compressed Pages: A Main Memory Compression Framework with Low Complexity and Low Latency. In *Proceedings of the International Symposium on Microarchitecture (MICRO)*, 2013.

[113] Gennady Pekhimenko, Vivek Seshadri, Onur Mutlu, Phillip B. Gibbons, Michael A. Kozuch, and Todd C. Mowry. Base-delta-immediate Compression: Practical Data Compression for On-chip Caches. In *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques (PACT)*, September 2012.

[114] Andrew Phelps. Memory subsystem including an error detection mechanism for address and control signals. US Patent, US 6,941,493 B2, Sep 2005.

[115] M. K. Qureshi, D. H. Kim, S. Khan, P. J. Nair, and O. Mutlu. AVATAR: A Variable-Retention-Time (VRT) Aware Refresh for DRAM Systems. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN)*, pages 427–437, June 2015.

[116] Irving Reed and Golomb Solomon. Polynomial codes over certain finite fields. *J. Soc. for Industrial and Applied Mathematics*, 8(2):300–304, 06/1960 1960.

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

[118] D Roberts and P Nair. FAULTSIM: A fast, configurable memory-resilience simulator. In *Proceedings of the Memory Forum*, volume 41, 2014.

[119] Paul Rosenfeld, Elliott Cooper-Balis, and Bruce Jacob. DRAMSim2: A Cycle Accurate Memory System Simulator. *IEEE Computer Architecture Letters*, 10(1):16–19, January 2011.

[120] Samsung Electronics Co. Samsung DDR4 SDRAM. `http://samsung.com/global/business/semiconductor/file/media/DDR4_Brochure-0.pdf`, 2013.

[121] Yiannakis Sazeides, Emre Özer, Danny Kershaw, Panagiota Nikolaou, Marios Kleanthous, and Jaume Abella. Implicit-Storing and Redundant-Encoding-of-Attribute Information in Error-Correction-Codes. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, 2013.

[122] Bianca Schroeder, Eduardo Pinheiro, and Wolf-Dietrich Weber. DRAM Errors in the Wild: A Large-scale Field Study. In *Proceedings of the International Joint Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, pages 193–204, 2009.

[123] A. Shafiee, M. Taassori, R. Balasubramonian, and A. Davis. MemZip: Exploring unconventional benefits from memory compression. In *Proceedings of the International Symposium on High Performance Computer Architecture (HPCA)*, 2014.

[124] John Shalf, Sudip Dosanjh, and John Morrison. Exascale Computing Technology Challenges. In *Proceedings of the International Conference on High Performance Computing for Computational Science*, pages 1–25, 2011.

[125] Richard Singleton. Maximum distance q-nary codes. *IEEE Transactions on Information Theory (ITIT)*, 10(2):116–118, Apr 1964.

[126] Y. H. Son, S. Lee, O. Seongil, S. Kwon, N. S. Kim, and J. H. Ahn. CiDRA: A cache-inspired DRAM resilience architecture. In *Proceedings of the International Symposium on High Performance Computer Architecture (HPCA)*, pages 502–513, Feb 2015.

[127] Vilas Sridharan, Nathan DeBardeleben, Sean Blanchard, Kurt B. Ferreira, Jon Stearley, John Shalf, and Sudhanva Gurumurthi. Memory Errors in Modern Systems: The Good, The Bad, and The Ugly. In *Proceedings of the International Symposium on Architectural Support for*

Programming Languages and Operating Systems (ASPLOS), pages 297–310, 2015.

[128] Vilas Sridharan and Dean Liberty. A Study of DRAM Failures in the Field. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC)*, pages 76:1–76:11, 2012.

[129] Vilas Sridharan, Jon Stearley, Nathan DeBardeleben, Sean Blanchard, and Sudhanva Gurumurthi. Feng Shui of Supercomputer Memory: Positional Effects in DRAM and SRAM Faults. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC)*, pages 22:1–22:11, 2013.

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

[131] Charles H. Stapper and Hsing-San Lee. Synergistic fault-tolerance for memory chips. *IEEE Transactions on Computers*, 41(9):1078–1087, 1992.

[132] Henry Stracovsky, Michael Espig, Victor W. Lee, and Daehyun Kim. Reliability support in memory systems without error correcting code support. US Patent, US 8,495,464 B2, 2013.

[133] Chin-Lung Su, Yi-Ting Yeh, and Cheng-Wen Wu. An integrated ECC and redundancy repair scheme for memory reliability enhancement. In

Proceedings of the International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT), pages 81–89, 2005.

[134] Sun Microsystems, Inc. T2 core microarchitecture specification.

[135] Synopsys Inc. Design Compiler I-2013.12-SP5-2, September 2014.

[136] Taiwan Semiconductor Manufacturing Company. 40nm CMOS Standard Cell Library v120b, 2009.

[137] T. Takahashi, T. Sekiguchi, R. Takemura, S. Narui, H. Fujisawa, S. Miyatake, M. Morino, K. Arai, S. Yamada, S. Shukuri, M. Nakamura, Y. Tadaki, K. Kajigaya, K. Kimura, and K. Itoh. A multigigabit DRAM technology with 6F2 open-bitline cell, distributed overdriven sensing, and stacked-flash fuse. *IEEE Journal of Solid State Circuits (JSSC)*, 36(11):1721–1727, Nov 2001.

[138] Devesh Tiwari, Saurabh Gupta, George Gallarno, Jim Rogers, and Don Maxwell. Reliability lessons learned from GPU experience with the Titan supercomputer at Oak Ridge leadership computing facility. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC)*, 2015.

[139] Top500 Supercomputer Sites, http://top500.org.

[140] Alberto Troia. Integrity of an address bus. US Patent, US 9,009,570 B2, Apr 2015.

[141] Aniruddha N. Udipi, Naveen Muralimanohar, Rajeev Balasubramonian, Al Davis, and Norman P. Jouppi. LOT-ECC: Localized and Tiered Reliability Mechanisms for Commodity Memory Systems. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, pages 285–296, 2012.

[142] Aniruddha N. Udipi, Naveen Muralimanohar, Niladrish Chatterjee, Rajeev Balasubramonian, Al Davis, and Norman P. Jouppi. Rethinking DRAM Design and Organization for Energy-constrained Multi-cores. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, pages 175–186, 2010.

[143] Pete D. Vogt. Combined command and data code. US Patent, US 7,827,462 B2, Nov 2010.

[144] David Wang. Protocol checking logic circuit for memory system reliability. US Patent, US 8,966,327 B1, Feb 2015.

[145] Chris Wilkerson, Alaa R. Alameldeen, Zeshan Chishti, Wei Wu, Dinesh Somasekhar, and Shih-lien Lu. Reducing Cache Power with Low-Cost, Multi-Bit Error-Correcting Codes. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, 2010.

[146] Samson S. Wong, Kandasamy Aravinthan, Gideon N. Levinsky, Shahar Dor, Richard T. Van, and Jiejun Lu. Methods and systems for detecting memory address transfer errors in an address bus. US Patent, US 7,293,221 B1, Nov 2007.

[147] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta. The SPLASH-2 Programs: Characterization and Methodological Considerations. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, 1995.

[148] T. H. Wu, P. Y. Chen, M. Lee, B. Y. Lin, C. W. Wu, C. H. Tien, H. C. Lin, H. Chen, C. N. Peng, and M. J. Wang. A memory yield improvement scheme combining built-in self-repair and error correction codes. In *IEEE International Test Conference (ITC)*, pages 1–9, 2012.

[149] R.E. Wunderlich, T.F. Wenisch, B. Falsafi, and J.C. Hoe. SMARTS: Accelerating Microarchitecture Simulation via Rigorous Statistical Sampling. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, pages 84–95, 2003.

[150] Jun Yang, Youtao Zhang, and Rajiv Gupta. Frequent Value Compression in Data Caches. In *Proceedings of the International Symposium on Microarchitecture (MICRO)*, pages 258–265, 2000.

[151] Doe Hyun Yoon and Mattan Erez. Virtualized and Flexible ECC for Main Memory. In *Proceedings of the International Symposium on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 397–408, 2010.

[152] Doe Hyun Yoon, Min Kyu Jeong, and Mattan Erez. Adaptive Granularity Memory Systems: A Tradeoff Between Storage Efficiency and

Throughput. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, pages 295–306, 2011.

[153] Doe Hyun Yoon, Min Kyu Jeong, Michael Sullivan, and Mattan Erez. The Dynamic Granularity Memory System. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, pages 548–559, 2012.

[154] Doe Hyun Yoon, Naveen Muralimanohar, Jichuan Chang, Parthasarathy Ranganathan, Norman P. Jouppi, and Mattan Erez. FREE-p: Protecting non-volatile memory against both hard and soft errors. In *Proceedings of the International Symposium on High Performance Computer Architecture (HPCA)*, pages 466–477, Feb 2011.

[155] Y. C. Yu, C. S. Hou, L. J. Chang, J. F. Li, C. Y. Lo, D. M. Kwai, Y. F. Chou, and C. W. Wu. A hybrid ECC and redundancy technique for reducing refresh power of DRAMs. In *Proceedings of the VLSI Test Symposium (VTS)*, pages 1–6, 2013.

[156] Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory (ITIT)*, 23(3):337–343, 1977.

Vita

Jungrae Kim was born in Seoul, Korea on October 30, 1978. He graduated from Seoul Science High School in 1997. Jungrae studied at Seoul National University, Seoul, Korea, where he received a B.S. degree in Electrical Engineering in 2001 and an M.S. degree in Electrical Engineering and Computer Science in 2003. He has worked in the lab of Professor Soo-Mook Moon, researching Java Just-In-Time compilers for the Intel IA-64 architecture.

Jungrae joined Samsung Electronics in 2003 and developed System-on-a-Chips for smart phones and digital media players. In 2011, he was awarded a full financial sponsorship for his Ph.D study from Samsung owing to his accomplishments.

Jungrae started graduate studies at the University of Texas at Austin in August, 2011. His research is focused on memory sub-systems, including DRAM reliability and memory compression. His research has been published in major computer architecture conferences, such as ISCA, HPCA, and SC. One paper was nominated as best paper in HPCA '15.

Permanent address: 54 Giheung-ro 38beon-gil Suite 101-105,
Yongin-si, Gyeonggi-do, Korea 16975

This dissertation was typeset with LaTeX[†] by the author.

[†] LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.