XED: Exposing On-Die Error Detection Information for Strong Memory Reliability

Prashant J. Nair† Vilas Sridharan‡ Moinuddin K. Qureshi†
†*School of Electrical and Computer Engineering*
Georgia Institute of Technology
{pnair6,moin}@ece.gatech.edu
‡*RAS Architecture*
Advanced Micro Devices Inc.
vilas.sridharan@amd.com

You are in a pitiable condition if you have to conceal what you wish to tell." -Publilius Syrus

***Abstract*—Large-granularity memory failures continue to be a critical impediment to system reliability. To make matters worse, as DRAM scales to smaller nodes, the frequency of unreliable bits in DRAM chips continues to increase. To mitigate such scaling-related failures, memory vendors are planning to equip existing DRAM chips with On-Die ECC. For maintaining compatibility with memory standards, On-Die ECC is kept invisible from the memory controller.**

This paper explores how to design high reliability memory systems in presence of On-Die ECC. We show that if On-Die ECC is not exposed to the memory system, having a 9-chip ECC-DIMM (implementing SECDED) provides almost no reliability benefits compared to an 8-chip non-ECC DIMM. We also show that if the error detection of On-Die ECC can be exposed to the memory controller, then Chipkill-level reliability can be achieved even with a 9-chip ECC-DIMM. To this end, we propose *eXposed On-Die Error Detection (XED)*, which exposes the On-Die error detection information without requiring changes to the memory standards or consuming bandwidth overheads. When the On-Die ECC detects an error, XED transmits a pre-defined *"catch-word"* instead of the corrected data value. On receiving the catch-word, the memory controller uses the parity stored in the 9-chip of the ECC-DIMM to correct the faulty chip (similar to RAID-3). Our studies show that XED provides Chipkill-level reliability (172x higher than SECDED), while incurring negligible overheads, with a 21% lower execution time than Chipkill. We also show that XED can enable Chipkill systems to provide Double-Chipkill level reliability while avoiding the associated storage, performance, and power overheads.

***Keywords*-On-Die ECC, Chipkill, Double-Chipkill, RAID-3**

I. Introduction

Technology scaling has been the prime driver of increasing the capacity of the Dynamic Random Access Memory (DRAM) modules. Unfortunately, as technology scales to smaller nodes, DRAM cells tend to become unreliable and exhibit errors [1,2]. The industry plans to continue DRAM scaling by placing Error Correcting Codes (ECC) inside DRAM dies, calling it *On-Die ECC* (also known as *In-DRAM ECC*) [3]. On-Die ECC enables DRAM manufacturers to correct errors from broken cells [4]. Consequently, DRAM chips with On-Die ECC are already proposed for systems with DDR3, DDR4 and LPDDR4 standards [3,5,6]. For maintaining compatibility with DDR standards and to reduce the bandwidth overheads for transmitting On-Die ECC information, manufacturers plan to conceal the On-Die error information to remain within the DRAM chips [3,6]. Thus, On-Die ECC is invisible to the system and cannot be leveraged to improve resilience against runtime faults. This paper looks at how to design systems with stronger memory resilience in the presence of On-Die ECC.

Recent field studies from super-computing clusters show that DRAM reliability continues to be a critical bottleneck for the overall system reliability [7]–[9]. Furthermore, these studies also highlight that large-granularity failures that happen at runtime, such as row-failures, column-failures and bank-failures, are almost as common as bit failures. DRAM modules can be protected from single bit failures using an ECC-DIMM that provisions an extra chip for error correction. However, tolerating large-granularity failures in the memory system is expensive and high-reliability systems often need to implement Chipkill to tolerate a chip failure at runtime. Unfortunately, implementing Chipkill requires activating 18 chips, which necessitates either using a non-commodity DIMM (x4 devices), and or accessing two memory ranks (x8 devices) simultaneously, which increases power and reduces parallelism. Ideally, we want to implement Chipkill using commodity memory modules and without the storage, performance, and power overheads.

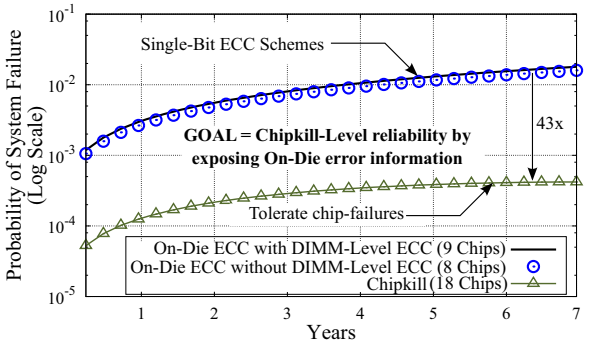


Figure 1. Effectiveness of reliability solutions in presence of On-Die ECC.

We analyze how On-Die ECC affects the reliability of DIMM-based ECC and Chipkill. Figure 1 shows the probability of system failure, by considering real world failure-rates, for the memory system over a period of 7 years. We compare three systems: (a) Non-ECC DIMM with 8 chips, (b) ECC-DIMM with 9 chips, and (c) Chipkill-based system with 18 chips. We observe that if the system is provisioned with On-Die ECC there is almost no benefit of having the DIMM-level ECC. Furthermore, Chipkill-based systems provide 43x more reliability than ECC-DIMM. From this analysis, we may conclude that the 9-chip ECC-DIMM solution is superfluous in the presence of On-Die ECC. We argue that this is an effect of concealing the On-Die ECC information from the external system. We show that revealing the On-Die ECC error detection to the memory controller can enable Chipkill-level reliability while avoiding the associated overheads.

1063-6897/16 $31.00 © 2016 IEEE
DOI 10.1109/ISCA.2016.38

Unfortunately, exposing On-Die ECC to the memory system requires that more bits be transferred from the DRAM chips to the memory controller [3,5,6]. This can accomplished by either providing more lanes or using additional bursts, both of which are incompatible with existing DDR standards [10,11]. Ideally, we would like to expose the On-Die error information without any bandwidth or latency overheads and without changing the existing standards. We leverage the observation that the memory controller does not need to have visibility of the On-Die ECC bits; it simply needs to know if the On-Die ECC has detected an error. The memory system can then use the On-Die error detection information in conjunction with the DIMM-Level parity and correct errors in a manner similar to RAID-3. To this end, this paper proposes *XED* (pronounced as "zed", the British pronunciation of the letter "z"), a technique that eXposes On-Die Error Detection information while avoiding the bandwidth overheads and changes to the memory standards.

To efficiently communicate that On-Die ECC has detected an error to the memory controller, XED relies on *Catch-Word*. A Catch-Word is predefined randomly selected data-value that is transmitted from the chip to memory controller to convey that a fault has occurred in a given DRAM chip. Both the memory controller and the DRAM chip a priori agrees on the given Catch-Word. XED uses the 9th chip in the ECC-DIMM to store parity information of all the other chips. When the On-Die ECC identifies an error, the DRAM chip transmits the Catch-Word instead of the requested data-value to the memory controller. When the memory controller recognizes the Catch-Word, it ignores the value from the associated chip and uses the parity from the 9th chip to reconstruct the data of the faulty chip. XED exploits the observation that typically a chip (x8 devices) provides a 64-bit data-value on each memory access. However, the chip cannot store all possible 2^{64} data-values. In fact, even a chip as large as 8Gb stores only 2^{27} 64-bit words. Even if all these words had unique values, the likelihood that the chip stores a data-value that matches with the Catch-Word is negligibly small (2^{-37}, or 1 in 140 billion).[1]

We discuss how XED can mitigate a chip failure by using an ECC-DIMM. We also discuss how XED can mitigate scaling faults in multiple chips. We then show how XED can perform correction when runtime chip failure occurs concurrently with scaling faults. Our evaluations show that XED provides 172x higher reliability than ECC-DIMM alone. Furthermore, XED incurs negligible performance overheads ($< 0.01\%$) and provides a 21% lower execution time compared to traditional Chipkill. We also analyze XED for a system that implements Chipkill and show that XED enables this system to achieve Double-Chipkill level reliability without the overheads of Double-Chipkill.

Overall, our paper makes the following contributions:

1) We show that DIMM-Level ECC provides no added reliability benefit to a memory system with On-Die ECC. This is because large-granularity runtime-faults are the main cause of memory failures [7,8,12]–[16]. Therefore, implementing the conventional DIMM-level SECDED with the 9th chip incurs area and power overheads without providing any reliability benefits.
2) We propose XED, a technique that uses Catch-Words to reveal On-Die ECC error detection information to the memory controller without relying on extra bandwidth and changes to the memory interface.
3) We propose a simple correction scheme for XED that uses the ECC-DIMM to store parity information and relies on RAID-3 based correction to tolerate a chip failure. We also show that XED is effective at tolerating chip failure in the presence of scaling failures. Our evaluation shows that XED enables Chipkill-level reliability without the power and performance overheads of traditional Chipkill implementations.
4) We show that XED can enable conventional Chipkill systems to provide Double-Chipkill level reliability while obviating the storage, performance, and power overheads of Double-Chipkill.

II. Background

We provide a brief background on the DRAM organization and On-Die ECC. We also discuss the sources of errors and discuss the typical techniques for error mitigation.

A. DRAM Organization

DRAM memory is typically implemented as Dual Inline Memory Modules (DIMM), consisting of eight chips (x8 devices) providing a 64-bit wide databus. Each chip is further divided into banks, and each bank is further divided into rows and columns [17,18]. An access to a DRAM DIMM activates a given bank in all of the chips. The access may activate a row of cells in the DRAM and then only a small portion from this row (corresponding to a cache line size, typically 64 bytes) is streamed out over the data bus [19]–[21]. Thus, each chip is responsible for providing 64-bit per access, which is sent using 8 bursts of 8 bits each.

If the DIMM is equipped with ECC, it will have a 9th chip and will support 72 data lines (64 for data and 8 for ECC). Each chip is still responsible for providing 64 bits for each memory access.

B. On-Die ECC: The Why and the How

As technology scales to smaller nodes, the number of faulty cells in a DRAM chip is expected to increase significantly. Mitigating these design-time faults with row-sparing or column-sparing will be prohibitively expensive.

[1]While the likelihood of data-value matching the Catch-Word is negligibly small (once every million years for an x8 DIMM), XED can continue to operate reliably even when this occurs. In fact, XED can reliably detect the episode of a data-value matching the Catch-Word, and use this information to change the Catch-Word. We discuss detecting collision and updating the Catch-Word in Section V-D3

To increase yield, DRAM companies would like to use DRAM chips with scaling faults while still ensuring reliable operation and without significant overheads. To achieve this, DRAM companies are planning to equip each chip with *On-Die ECC* (also called as *in-DRAM ECC*), whereby each 64-bit data within the chip is protected by an 8-bit SECDED code. DRAM errors are handled internally within the DRAM chip and this information is not made visible to the memory controller. As such, the On-Die ECC works transparently without requiring any changes to the existing memory interfaces and without making the memory controller aware that the chip is equipped with On-Die ECC.

C. Fault Modes: Birthtime versus Runtime

We classify the faults into two categories: birthtime faults and runtime faults. Birthtime faults are those that occur at manufacturing time and can be detected by the memory vendors. To ensure reliable operation of the chips, it is important that the memory vendors mitigate the birthtime faults or simply discard the faulty chips. Scaling faults [1]–[3,22,23] are birthtime faults and the On-Die ECC is designed such that these faults do not become visible to the external system. To ensure that a chip with On-Die ECC is not faulty, the manufactures will need to ensure that no 64-bit word has more than 1 faulty bit (if a word had multi-bit scaling-faults then use row sparing or column sparing to fix those uncommon cases). In our paper, we assume that scaling faults are limited to at most 1 bit per 64-bit word.

Runtime faults are those that occur during the operation of the DRAM chip. Runtime failures can be either transient or permanent, and can occur at different granularities, such as bit-failure, word-failure, column-failure, row-failure, bank-failure or rank-failure [7]–[9]. Recent field studies show that large-granularity runtime failures are almost as common as bit-failures. Therefore, we need solutions to efficiently handle not only bit-failures but also large-granularity failures. The typical error mitigation techniques are described next.

D. Typical Error Mitigation Techniques

1) SECDED: Memory systems may develop single-bit faults due to alpha-particle strikes and weak cells [24,25]. To protect against single-bit faults, memory systems can use a variant of ECC codes that corrects single-bit errors and detect two-bit errors (SECDED) [26]–[28]. DIMMs equipped with SECDED typically provide 8 bits of ECC for every 64 bits of data, and while activating a single rank.

2) Chipkill: Large-granularity failures, such as chip failures, can be tolerated by Chipkill, which employs symbol-based error correction code. Each data chip provides one symbol and there are extra chips provisioned for storing "check" symbols that are used locate and correct the faulty symbol (chip). With two check symbols, Chipkill can correct one faulty symbol (chip) and detect up to two faulty symbols (chips) [29]. As Chipkill needs two extra chips for storing these symbols, commercial implementations of Chipkill require that 18 chips be activated for each memory access (16 for data and two for check symbols). Unfortunately, this would mean that we either use non-commodity chips (x4 devices) or obtain two cachelines for each access (x8 devices), causing a 100% overfetch which increases power consumption and reduces parallelism.

3) Erasures: The Chipkill design tries to do both, locate the faulty chip as well as correct the faulty chip. If we have an alternative means of knowing which chip is faulty, then we can tolerate a chip failure by simply relying on one chip (in general, for tolerating N chip failures we need only N extra chips). This is called as *Erasure Coding* [29]–[31].

E. Our Goal

Our goal is to obtain Chipkill-level reliability without the associated overheads of area, power, and performance. We observe that if the DRAM chips already have On-Die ECC, then having the information about which chip encountered a fault can help us design an Erasure-based scheme to tolerate chip failures. However, we want to expose the On-Die error detection information from inside the DRAM chip to the memory controller without incurring extra bandwidth and changing the memory interfaces. To that end, we propose eXposed On-Die Error Detection (XED). Before we describe XED, we describe our reliability evaluation infrastructure.

III. Reliability Evaluation

To evaluate reliability of our proposed schemes we use FaultSim, an industry-grade fault and repair simulator [32]. We extend FaultSim to accommodate scaling-faults faults. Based on prior studies, we assume a scaling-fault rate of 10^{-4} [1,3]. To model runtime-faults, we use real-world field data from Sridharan et al. [7] as shown in Table I.

Table I
DRAM failures per billion hours (FIT) [7]

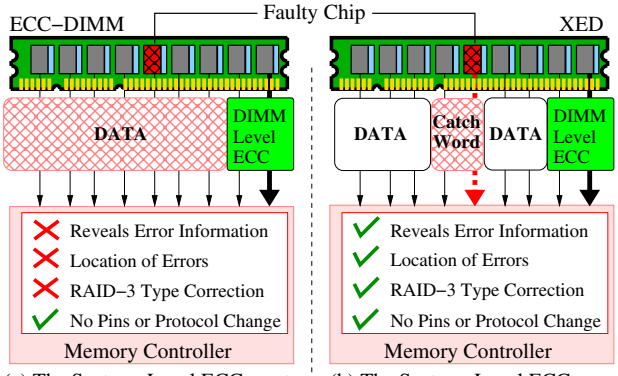
DRAM Chip Failure Mode	Fault Rate (FIT)	
	Transient	Permanent
Single bit	14.2	18.6
Single word	1.4	0.3
Single column	1.4	5.6
Single row	0.2	8.2
Single bank	0.8	10
Multi-bank	0.3	1.4
Multi-rank	0.9	2.8

Our memory system has 4 channels, each containing dual-ranked DIMM of 4GB capacity (x8 devices of 2Gb each). We perform Monte-Carlo simulations over a period of 7 years and check if the system encounters an uncorrectable, mis-corrected, or silent error at anytime during the 7-year period. If so, we deemed the system as a "failed" system. We compute *Probability of System Failure* as the fraction of systems that failed at anytime during the 7-year period. We simulate a total of 1 billion systems and report the average Probability of System-Failure as the figure of merit.

IV. XED: An Overview

We investigate a memory system in which all DRAM chips are equipped with On-Die ECC. Our key observation is that exposing the information about On-Die error detection to the memory controller can enable high-reliability memory systems at low cost. XED exposes the information that the On-Die ECC detected (or corrected) an error to the memory controller without requiring any changes to the bus interface or requiring extra bandwidth. We implement XED using a conventional ECC-DIMM consisting of 9 chips.

Figure 2 shows an overview of XED. Unlike conventional ECC-DIMM, which uses the 9th chip to store the ECC code, XED uses the 9th chip to store the parity information computed across the remaining eight chips. XED transfers error information by replacing data with Catch-Words. Thereafter, XED can correct data errors with the help of the error location and the parity information stored in the 9th chip (similar to RAID-3 [33]). This enables XED to not only identify the faulty chip, but also to reconstruct the data for that faulty chip.



(a) The System–Level ECC must correct invisible errors (WEAK)

(b) The System–Level ECC can correct visible errors (STRONG)

Figure 2. (a) Conventional ECC-DIMM is not useful in presence of On-Die ECC (b) XED exposes detection information of On-Die ECC to provide stronger reliability (using RAID-3) without any interface changes.

This paper provides an interface that enables exposing the On-Die error detection information using *Catch-Words* that act as error indicators. XED relies on the observation that a typical memory chip (with x8 devices) provides a 64-bit data-value on each transfer. However, the chip does not store all possible 2^{64} values. In-fact, even a relatively large 8Gb chip stores only 2^{27} words of 64-bit each. Even if all the stored 64-bit words were unique, the likelihood of the chip storing the data-value that matches the randomly selected Catch-Word is negligibly small (2^{-37}, or 1 in 140 billion). So the appearance of Catch-Word at the memory controller signals that an episode of error detection or correction by On-Die ECC occurred within the DRAM chip. This paper analyzes the effectiveness of XED in the presence of chip failures and scaling faults.

V. Efficient Chipkill With XED

This section describes the implementation of XED. We also discuss how correction is performed using the position of faulty chip and the DIMM-level parity stored in the 9th chip of XED. In this section, we assume that at most one chip is faulty. We discuss the case of multiple scaling faults (Section VII-B) and a chip failure in the presence of scaling faults (Section VII-C) in later sections.

A. Implementing XED using an ECC-DIMM

We equip each chip with two registers: *XED-Enable* and *Catch-Word-Register (CWR)*. To enable XED on the DIMM, the XED-Enable register is set to 1. Furthermore, the CWR is also set to a randomly selected 64-bit value by the memory controller. Fortunately, DRAM DIMMs use a separate interface to update internal parameters using *Mode Set Registers (MRS)*. XED-Enable and CWR registers can also be configured using the MRS. As the Catch-Word is 64-bits long and XED-Enable is 1-bit long, the total storage overhead for enabling XED is only 65 bits per chip.

XED-Enable register is set at boot time and the memory controller generates a unique random Catch-Word and stores it in each chip. The memory controller also retains a copy of CWR. This helps the memory controller in deciding if the data provided by the chip matches with the Catch-Word. To implement XED, DRAM chips are also equipped with a *Data-Catch-Word Multiplexer (DC-Mux)* that dynamically selects between the requested data value and Catch-Words based on the correction or detection of errors. Figure 3 shows the internals of a DRAM chip equipped with a DC-Mux. If no error detected or corrected by the On-Die ECC then DC-Mux selects the data. However, if the On-Die ECC detects or corrects an error, the DC-Mux to selects the Catch-Word. Note that this selection happens on if the XED-Enable bit is set. If XED-Enable is not set, then the DRAM Chip supplies the data value and acts as the baseline ECC-DIMM.

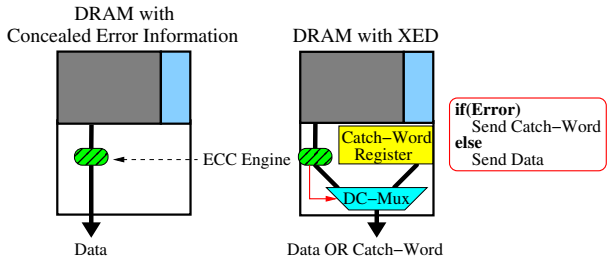


Figure 3. XED uses a multiplexer to provide the Catch-Word or the data value, depending on if the error is detected or corrected by On-Die ECC.

B. Detection: A By-Product Of On-Die ECC

The SECDED code corrects one faulty bit and detects of two faulty bits. As SECDED code always detect the error before correcting it, we can also reuse the SECDED code to find out if an error was detected. The distance between valid code-words is called as the hamming distance and any valid

data would always land on valid code-words [4]. However, if the data is erroneous then it tends to land on an invalid code-word. An ECC scheme mitigates errors by selecting a unique nearest valid code-word for the detected invalid code-word. Therefore, On-Die ECC can implicitly serve as a strong detection code if it informs the memory system whenever an invalid code-word is encountered.

For example, Figure 4 depicts a scenario where an invalid code-word is encountered by the ECC engine, so XED would use the DC-Mux to transmit a Catch-Words instead of the requested data. Thus, the DC-Mux transmits the requested data only when the On-Die ECC engine detects a valid code-word, or when XED-Enable is set to 0.

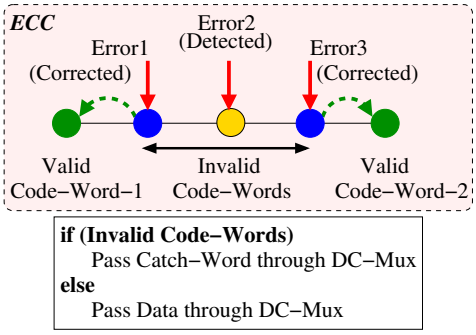


Figure 4. Leveraging ECC-based correction for stronger detection. For example, a three-bit error may get mis-corrected with conventional SECDED DIMM, but XED will be able to correct it.

C. Mitigate a Chip Failure Using XED

XED uses of Catch-Words to indentify the faulty chip and the DIMM-level ECC to correct erroneous data of the faulty chip. We describe the correction performed by XED.

1) Using Catch-Words and Parity To Locate Errors: The ninth chip in a XED is provisioned to store "Parity" of the data words in a burst. A parity code enables the memory controller to identify any single erroneous data word. For example, if data words D0 to D7 form a data burst, then Parity is computed as an XOR ($\oplus$) of all words between D0 to D7, as shown in Equation (1)

$$Parity = D0 \oplus D1 \oplus D2 \oplus D3 \oplus D4 \cdots \oplus D7$$
$$\Rightarrow Parity \oplus D0 \oplus D1 \oplus D2 \oplus D3 \oplus D4 \cdots \oplus D7 = 0 \quad (1)$$

During a write, the parity is stored in the 9th DRAM-chip. On a subsequent read, if any data word or the Parity gets corrupted, then Equation (1) will not be satisfied. Consequently, memory system detects a data error. The key drawback of this technique is that, using Parity alone, a memory system cannot identify which data was erroneous. To identify the faulty chips, we use the On-Die error code that is provisioned to act as a strong error detection code within each chip. On detecting an error, the chip relays the Catch-Word rather than transmitting the erroneous data. As the memory controller can identify the Catch-Word, it can detect the faulty chip. For example, Figure 5 shows a faulty chip that sends a Catch-Word (CW3) instead of Data (D3).

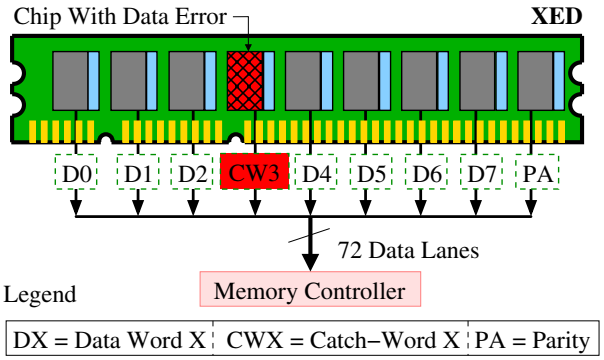


Figure 5. Catch-Words are transmitted instead of Data if On-Die error code detects errors. When used with Parity, Catch-Words enable the memory system to identify the faulty chip.

In this case, using Equation (1), we get Equation (2) which represents the case of a single erroneous chip.

$$Parity \oplus D0 \oplus D1 \oplus D2 \oplus \mathbf{CW3} \oplus D4 \cdots \oplus D7 \neq 0 \quad (2)$$

Using the Catch-Words as an identifier and Equation (2) to detect errors, the memory system can identify that Chip-3 is the faulty chip. As catch-words are transmitted instead of valid data, there is no change in the memory protocol. Therefore, XED is compatible with existing memory interfaces and can relay the error information without any changes in the memory protocols.

2) Using Parity to Correct Errors: On detecting only a single Catch-Word, the memory controller can correct the erroneous data by using Parity. For example, a corrupted data-word D3 can be recovered using Parity as shown in Equation (3). Using Parity and other valid data-words, the memory controller reconstructs the corrupted data-word that is pointed by the Catch-Word.

$$Parity \neq D0 \oplus D1 \oplus D2 \oplus CW3 \oplus D4 \cdots \oplus D7 \cdots [\text{from (2)}]$$
$$\textbf{Solving for D3 instead of CW3}$$
$$\Rightarrow D3 = D0 \oplus D1 \oplus D2 \oplus Parity \oplus D4 \cdots \oplus D7 \quad (3)$$

Therefore, XED-based systems can achieve Chipkill-level reliability by activating only one rank of 9 chips. Thus, XED enables computer systems to obtain Chipkill-level reliability by using commodity x8 DRAM-chips.

D. Collisions of Catch-Words with Data

It is possible that a legitimate data-word matches a Catch-Word. We refer to such an incident as the collision of Catch-Words with data-words. Note that occurrence of a collision does not indicate loss of reliability with XED. If collision happens, XED will ignore the data value from the given chip

assuming it as a Catch-Word, and recreate the same value using the parity information stored in the ninth chip. So, even in the rare case of a collision, XED still provides the correct value, albeit with unnecessary correction.

1) Identifying a Collision: A collision can easily be identified if a Catch-Word is encountered, and the value corrected from XED (using the parity stored in the 9th chip) matches with the Catch-Word.

2) Chances of Collision: We quantitatively identify the chance of a collision for a XED-based DRAM chip. If we conservatively assume that a different data-word is written in every transaction, we can measure the probability of collision of Catch-Words for each DRAM chip. Figure 6 depicts the probability of collision over time. As we use x8 DRAM-chips, we have a randomly selected 64-bit Catch-Word. Therefore, the probability that a given data value being written to the DRAM chip matches with the Catch-Word is 1 out of 2^{64}, an extremely unlikely event. On average, an x8 DRAM-chip will have a collision once every 3.2 million years, assuming a memory write every 4ns.

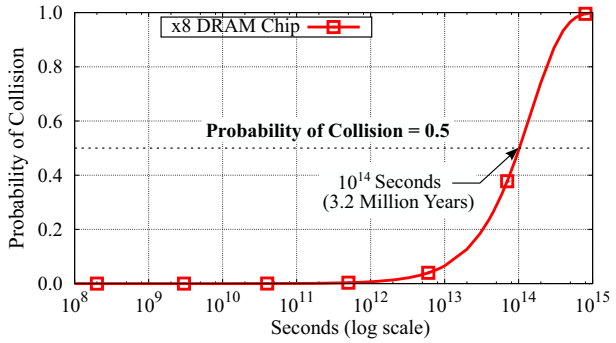


Figure 6. XED with x8 chips is likely to encounter collisions once every 3.2 million years, on average.

3) Updating Catch-Words on Detecting a Collision: When a collision with Catch-Word is detected, we recommend that the memory controller regenerate a new Catch-Word and update all the DRAM chips with new Catch-Words. Doing so, would increase the average time between collisions. For updating the Catch-Word, the memory controller does not have to read the entire data from the chip or update all ECC values within each chip. This is because, randomly generating a Catch-Word will reduce the average chances of collision to be every 3.2 million years irrespective of the data value within each chip.

E. The Need for Strong On-Die Error Detection

We assume 8-bits of On-Die ECC for every 64-bits of data, with an aim of implementing SECDED on 64-bit granularity [3]. If there is freedom in choosing the code for On-Die ECC, we want to explore codes that not only guarantee single-bit correction but also are highly effective at multi-bit detection. While Hamming-Code [4] is popular for implementing SECDED in memory systems, we recommend that the On-Die ECC use CRC8-ATM code [34,35] for implementing SECDED. CRC8-ATM has previously been used in computer networks [34,35]. Both Hamming-Code and CRC8-ATM provide the functionality of SECDED, however CRC8-ATM code has stronger error detection capabilities. Table II shows the invalid-code detection capability of the Hamming Code and the CRC8-ATM code under both random errors as well as burst errors.

Table II
DETECTION-RATE OF RANDOM AND BURST ERRORS

Errors	(72,64) Hamming Code		(72,64) CRC8-ATM Code	
	Random	Burst	Random	Burst
1	100%	100%	100%	100%
2	100%	100%	100%	100%
3	100%	100%	100%	100%
4	98.3%	50.73%	99.2%	100%
5	100%	100%	100%	100%
6	99.1%	100%	99.22%	100%
7	100%	100%	100%	100%
8	99.16%	50.75%	99.22%	100%

Hamming Code has as low as 50.7% detection-rate for invalid code-words in the presence of burst errors. On the other hand, a CRC8-ATM code has 100% detection-rate of invalid code-words in the presence of burst errors. Therefore, the CRC8-ATM code is more effective than Hamming Code for detecting burst errors. Therefore, we recommend using CRC8-ATM code as a design choice for On-Die ECC.

The SECDED code should incur low latency for encoding and decoding. Fortunately, CRC8-ATMs implementations can be performed within one cycle by using only 256 entry lookup tables [34,35]. The CRC8-ATM computation consumes only a single cycle latency as it only uses a tree of XOR-gates for encoding-decoding. The current the On-Die ECC specifications do not provision any additional latency for encoding or decoding and leverage the timing slacks within DRAM chips for ECC computation (1 to 2 cycles).

VI. MITIGATING CHIP FAILURES WHEN ON-DIE ECC FAILS TO DETECT AN ERROR

XED relies on On-Die ECC to detect faults within the DRAM chips. Unfortunately, this detection is imperfect, and there is a small (0.8%) likelihood that a multi-bit error within the chip remains undetected. If the system encounters a multi-bit failure in a chip, and the On-Die ECC fails to detect this fault, XED will still be able to detect this fault at the system level because of the parity mismatch at the DIMM-level ECC. We could deem such a scenario to be an uncorrectable error, and inform the system that an uncorrectable error has occurred. Unfortunately, the resilience of such a design would be much worse than Chipkill, as we are unable to correct the faulty chip. However, if we could identify the faulty chip, then we could use system-level parity to reconstruct the data of the faulty chip. We describe two schemes to identify the faulty chip when the On-Die ECC fails to detect an error.

A. Inter-Line Fault Diagnosis

A multi-bit failure can occur at runtime due to large-granularity faults, such a row-failure, column-failure or bank-failure. Such error modes cause not only the requested line to fail, but also the spatially close lines to fail. We use the insight that even if the error in a single cacheline goes undetected by On-Die ECC, it is highly unlikely that errors in the neighboring faulty lines will also go undetected by On-Die ECC. Therefore, if we read multiple neighboring lines, then we are likely to notice errors in the neighboring lines for the faulty chip. The chip with the highest number of faults in the neighboring lines is deemed as the faulty chip. We propose to stream out the entire row buffer (128 lines), and use a threshold of 10% faulty lines to identify the faulty chip. Our analysis in Section VIII shows that using this threshold is sufficient to avoid identifying chips as faulty simply due to scaling faults. We term this scheme as *Inter-Line Fault Diagnosis*.

Performing Inter-Line Fault Diagnosis incurs high latency (128 reads), so we want to avoid performing this diagnosis frequently. We propose to store the result of this diagnosis in a hardware structure called the *Faulty-Row Chip Tracker (FCT)* that tracks the location of the faulty row and the corresponding faulty chip identified using Inter-Line Fault Diagnosis. An FCT-entry is a tuple of the row-address (32-bits) and the faulty chip (4 bits). We use a small FCT with few entries (4-8) as the system is either likely to encounter 1 or 2 faulty rows (due to a row failure) or thousands of faulty rows (due to column failure or bank failure). If only a single row-failure occurs, only one FCT entry is updated and the chip is not marked as faulty. However, for column or bank failure, all FCT entries would get used and point to the same chip. This chip is permanently marked as faulty, and for all subsequent accesses to this chip, XED would reconstruct the data for this chip using parity information.

B. Intra-Line Fault Diagnosis

While Inter-Line Fault Diagnosis is effective at detecting errors that span across multiple lines, it is ineffective when the multi-bit error is constrained to be within the given line. In such scenarios, the neighboring lines will be error free and the Inter-Line Fault Diagnosis will be unable to identify the faulty chips. When this occurs, we perform an *Intra-Line Fault Diagnosis* that tries to detect permanent errors in the requested line. To accomplish this, we first copy the data of the requested line in a buffer. A diagnosis is then performed by writing sequences of '`all-zeros`' and '`all-ones`' into the requested memory line and reading the value. The chip with the permanent word faults or bit faults will get detected by this diagnosis. If the fault occurred in only one chip, then the data for the chip can be recovered using parity information.

We note that Intra-Line Fault Diagnosis will be unable to detect word failures that are transient. Fortunately, the rate of a transient word fault is relatively small (7.7×10^{-4} over a period of 7 years) and the likelihood that the On-Die ECC will be unable to detect it is also quite small (0.8%), so these cases happen with a negligibly low rate (6.1×10^{-6}, two orders of magnitude smaller than a multi-chip failure).[2]

C. Results: Effectiveness of XED

We perform reliability evaluations with a system that employs DRAM chips with On-Die ECC. Figure 7 shows, that XEDs provide 172x more reliability than Ordinary DIMMs. XEDs are also more 4x more resilient than any ECC-DIMM based Chipkill. This is because, Chipkill operates over 18-DRAM chips, whereas XEDs operate over only 9-DRAM chips. A larger number of chips reduces the mean time to failure (MTTF) for a system.

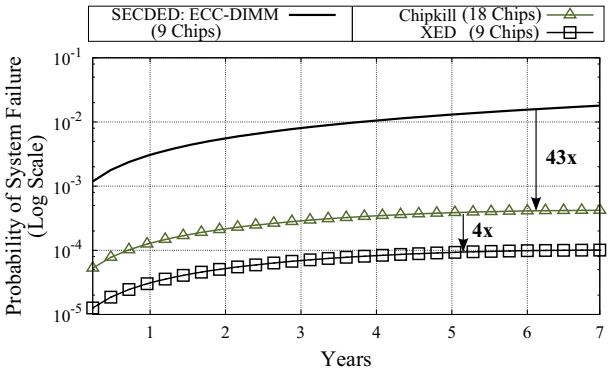


Figure 7. Reliability of ECC-DIMM, XED, and Chipkill. XED is 172x more reliable than ECC-DIMM and 4x more reliable than Chipkill.

We noted (in Figure 1) that if error detection information of On-Die ECC is not exposed to the external system, the having the 9th chip in the ECC-DIMM does not provide any added reliability benefits. This is because, once the chips can tolerate single bit failures, the dominant source of failure is due to large-granularity failures such a row or column or bank failures. Simply using a 9th chip to store SECDED is ineffective at mitigating such large-granularity faults.

VII. XED for Mitigating Scaling Errors

The On-Die ECC is meant to protect the DRAM chip against scaling faults. While the DRAM manufactures will ensure that there are no two faulty bits are placed within the same 64-bit word of the given chip, it is possible that two separate chips can each encounter 1 faulty bit while providing data for a single 64 byte access. Ideally, XED should correct all of these scaling faults when there are no runtime errors. We analyze the effectiveness of XED at mitigating scaling faults for both when they occur without runtime faults and in the presence of runtime faults.

[2]There is a small probability that two words within a line will each have 1-bit scaling fault. If a single-bit runtime fault occurs in either of these two words, it would result in an detectable uncorrectable error (DUE). Fortunately, the rate of this event is negligibly small (10^{-15} over 7 years).

A. Chance of Receiving Multiple Catch-Words

It is possible that two or more DRAM chips can detect scaling errors simultaneously and relay Catch-Words. As scaling errors are single-bit failures, they will always be detected by the On-Die Error Code. Fortunately, the chances of two Catch-Words for any memory transaction are extremely low. Table III shows that even at an error rate of 10^{-4}, there is only 2×10^{-5} chance of getting multiple Catch-Words in a given access. On receiving Catch-Words from multiple chips, XED is able to correct the data for all these chips, as longs as the errors are only due to scaling faults.

Table III
LIKELIHOOD OF DIRECTED ON-DIE CORRECTION WITH XED

Scaling-Fault Rate	Chance of Receiving Multiple Catch-Words
10^{-4}	2×10^{-5}
10^{-5}	2×10^{-7}
10^{-6}	2×10^{-9}

B. Correcting Scaling Errors in Multiple Chips

To correct scaling-faults, XED relies on the error correction capability of On-Die ECC, which is guaranteed to correct the single bit error. On receiving a line with multiple Catch-Words, the memory controller enters a serial mode, where it allows only one request to go through the DIMM. The memory controller resets the XED-Enable bit, reads the data from the given location (as XED-Enable is not set, the DIMM will send the corrected values), and then set the XED-Enable bit. It will then use the parity information in the 9th chip to ensure that the data read from this operation matches with the parity. Note that correcting scaling errors requires multiple read and write operations. Fortunately, this overhead is incurred infrequently – once every 200K accesses even for a high error rate of 10^{-4}.

C. Correcting Runtime Failures along-with Scaling Errors

A runtime failure in one chip can occur concurrently with scaling-related faults in other chips to generate multiple Catch-Words. This can be detected as the system-level, as the parity of the 9th chip will cause a mismatch. In this case, the memory controller needs to identify the chip with the large granularity fault and use the parity to recover correct data for the chip failure. To achieve this, the memory controller instructs the On-Die ECC to correct these errors and performs Inter-Line and Intra-Line diagnosis on the faulty chip. A scaling fault is corrected by On-Die ECC and the chip failure is identified by using Inter-Line Fault Diagnosis and Intra-Line Fault Diagnosis. If the diagnosis is successful at identifying a faulty chip, the memory controller can recover the data of the faulty chip using parity information. However, if the diagnosis cannot determine a faulty, then XED signals an episode of Detected Uncorrectable Error (DUE) so that the system can restart or to restore an earlier checkpoint.

D. Results: XED for Runtime Errors + Scaling Errors

Figure 8 shows the effectiveness of XED, ECC-DIMM, and Chipkill in the presence of scaling errors. We assume the rate of scaling errors to be 10^{-4}. We observe that, even in the presence of scaling errors, XED continues to provide stronger reliability than even Chipkill. Chipkill provides 43x stronger reliability than ECC-DIMM, whereas XED provides 172x stronger reliability than ECC-DIMM. This is because, On-Die ECC enables the memory system to correct scaling-faults in addition to runtime-faults.

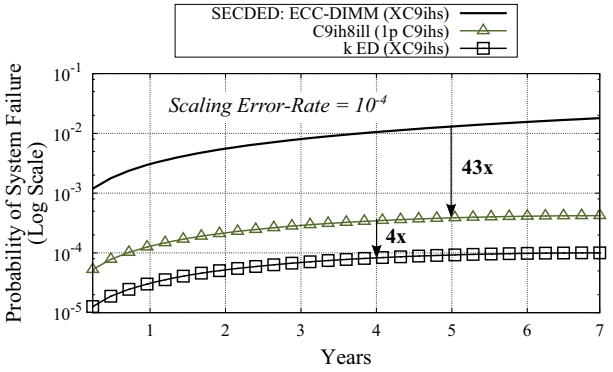


Figure 8. Reliability of ECC-DIMM, XED and Chipkill for runtime faults occurring in the presence of scaling-faults (10^{-4}).

VIII. SDC AND DUE RATE OF XED

XED is guaranteed to correct scaling errors in any number of chips. However, for a chip failure, there is a small likelihood that the error may go unnoticed, resulting in a mis-correction, or cause a detectable error which cannot be corrected. We quantify the vulnerability of XED using two metrics: *Detected Uncorrectable Error (DUE)* and *Silent Data Corruption (SDC)*. DUE indicates the scenario when the system encounters an uncorrectable error, whereas SDC captures the scenarios where the error remains undetected or gets mis-corrected.

DUE: The dominant cause of DUE are transient word-faults. When this occurs, XED first performs Inter-Line Fault Diagnosis followed by Intra-Line Fault Diagnosis, both of which fail to identify the faulty chip. In this case, even though XED detects the error due to parity mismatch of the DIMM-level parity, XED is unable to perform correction and reports an uncorrectable error. Fortunately, the rate of encountering a transient word-fault during a 7 year period is only 7.7×10^{-4}. Furthermore, the likelihood that this fault is undetected by On-Die ECC is only 0.8%. Therefore, the rate that XED reports an uncorrectable error due to transient word-fault, over a period of 7 years, is 6.1×10^{-6}.

SDC: The dominant cause of SDC is an incorrect identification of a faulty chip by Inter-Line Fault Diagnosis. This diagnosis relies on a faulty chip encountering a large number of errors and the other chips not encountering as many errors. We use a threshold of 10% faulty-lines within

a row to identify the faulty chip. Under high rate of scaling-related faults, there is a small probability that 10% of the lines in the row will have scaling errors. This may cause the diagnosis to deem the incorrect chip as faulty. Fortunately, even at a high error rate of scaling related fault, the chance that 10% of the lines in a row will have errors is negligibly small (10^{-12} under scaling-related fault rate of 10^{-4}).

Table IV shows the DUE and SDC rate for XED, assuming runtime failures are constrained to be within one chip. The SDC rate is 1.4×10^{-13} and the DUE rate is 6.1×10^{-6}. Note that the DUE rate is two orders of magnitude smaller than the likelihood of data loss due to multi-chip failure. Given that our solution is not designed to tolerate multi-chip failures, such failures will determine the overall reliability of the system, rather than the SDC and DUE rates of XED.

Table IV
SDC AND DUE RATE OF XED

Source of Vulnerability	Rate over 7 years
XED: Scaling-Related Faults	**No SDC or DUE**
XED: Row/ Column/ Bank Failure	1.4×10^{-13} (**SDC**)
XED: Word Failure	6.1×10^{-6} (**DUE**)
Data Loss from Multi-Chip Failures	5.8×10^{-4}

IX. DOUBLE-CHIPKILL WITH XED

Memory systems that seek stronger reliability than Chipkill implement *Double-Chipkill* to correct up-to two faulty chips. Double-Chipkill requires four extra symbols, two each for identifying the faulty chips and for correcting the data of these faulty chips. Therefore, it is typically implemented with 36 chips, whereby 32 chips store the data and 4 chips store the check symbols. Unfortunately, accessing 36 chips requires activation of upto two ranks over non-commodity DIMMs consisting of x4 DRAM-chips. Thus, even with x4 devices, Double-Chipkill requires overfetch of 100%. It would be desirable to obtain Double-Chipkill level reliability on a single cache line, without activating multiple ranks or channels. We show how XED can be applied to conventional Chipkill designs (with x4 devices) to obtain the reliability similar to Double-Chipkill. For this section only, we assume all systems are designed with x4 devices.

A. Use Erasure Coding For Error Correction

When XED is implemented on the top of conventional Chipkill design, we would have two extra chips (16 data chips plus two extra symbol chips). Given that XED can provide the location of the faulty chips, we can perform *erasure* based error correction using the two symbol chips to correct upto two chip failures. As this implementation uses 18 chips of x4 devices, each access obtains only a single cacheline, and avoids the power and performance overheads of Double-Chipkill. We note that, with x4 devices, the Catch-Word is only 32-bits, so the expected time to collision is approximately 6.6 hours (fortunately, the latency to update the Catch-Word is only a few hundred nanoseconds).

B. Results: Double-Chipkill with XED

Figure 9 compares the reliability of Double-Chipkill, Single-Chipkill, and XED implemented with Single-Chipkill systems, all evaluated in the absence of scaling errors. Overall, Double-Chipkill provides almost an order of magnitude improvement over Single-Chipkill. Unfortunately, it incurs significant power and performance overheads compared with Single-Chipkill. XED allows the memory system to get Double-Chipkill level reliability while retaining the hardware of Single-Chipkill. In fact, given that XED on the top of Chipkill has only 18 chips instead of the 36 chips for Double-Chipkill, we observe that XED provides almost 8.5x higher reliability than Double-Chipkill while obviating the performance and power overheads of Double-Chipkill.

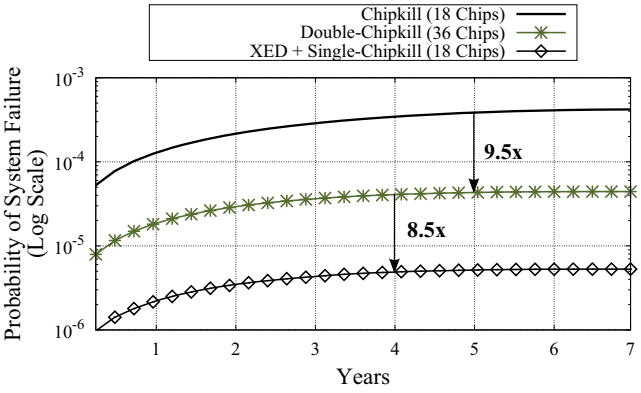


Figure 9. Reliability of Single-Chipkill, Double-Chipkill, and XED-based Single-Chipkill. Even with hardware similar to Single-Chipkill, XED provides 8.5x more reliability than Double-Chipkill.

Figure 10 compares the reliability of Double-Chipkill, Single-Chipkill, and XED on top of Single-Chipkill in the presence of scaling errors. We assume the rate of scaling errors to be 10^{-4}. We note that, in the presence of scaling errors, Double-Chipkill is 5.5x more effective than Single-Chipkill. XED implemented with Single-Chipkill continues to provide 8.5x better reliability than Double-Chipkill, primarily due to fewer chips.

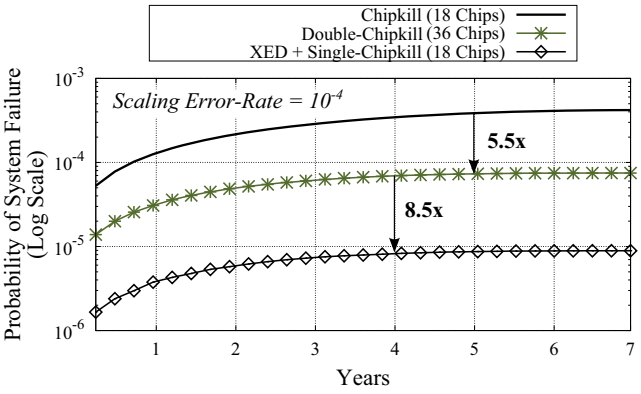


Figure 10. Reliability of Single-Chipkill, Double-Chipkill, and XED-based Single-Chipkill in the presence of scaling faults. XED on Single-Chipkill provides 8.5x more reliability than Double-Chipkill.

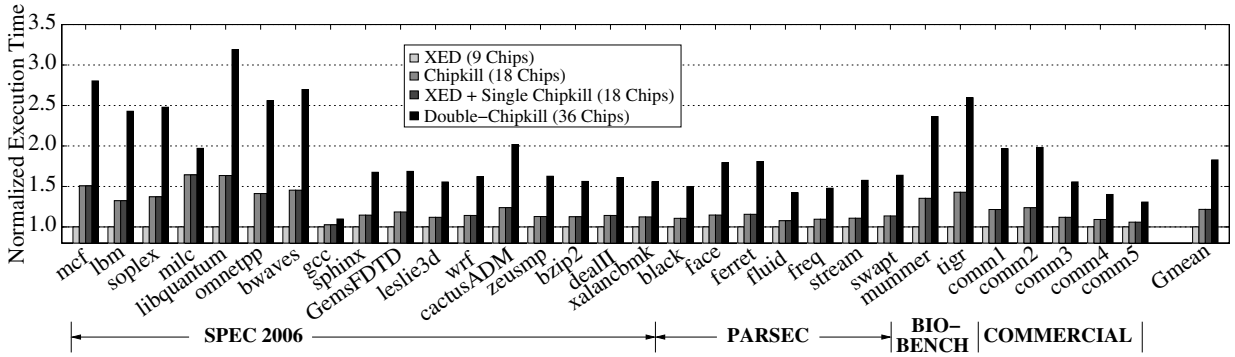


Figure 11. Normalized Execution Time (with respect to ECC-DIMM) for XED, Chipkill, XED on the top of Chipkill and Double-Chipkill. XED activates 2x fewer ranks and has 21% (61%) lower execution time than Chipkill (Double-Chipkill).

X. Experimental Methodology

To evaluate memory power and performance impact, we use USIMM, a cycle accurate memory system simulator [36,37]. USIMM enforces strict timing and also models all JEDEC DDR3 protocol specifications. USIMM is configured with the power parameters from industrial 2Gb x8-DRAM chips and x4-DRAM chips [38]. As On-Die ECC needs 12.5% more DRAM cells per die, we increase the background current and the current for refreshes, activation and precharge by 12.5%. Since error detections require only a syndrome check, it is assumed to consume 1 core cycle. The error correction at the memory controller is assumed to consume 4 core cycles. For erasure codes, the error correction is conservatively assumed to incur 60 core cycles. Table V shows the parameters for the baseline system.

Table V
Baseline System Configuration

Number of cores	8
Processor clock speed	3.2GHz
Processor ROB size	160
Processor retire width	4
Processor fetch width	4
Last Level Cache (Shared)	8MB, 16-Way, 64B lines
Memory bus speed	800MHz
DDR3 Memory channels	4
Ranks per channel	2
Banks per rank	8
Rows per bank	32K
Columns (cache lines) per row	128

For our evaluations, we chose benchmarks which have greater than "1 Miss Per 1000 Instructions" from Last Level Cache, from the SPECCPU 2006 [39], PARSEC [40] and BioBench [41] suites. We also include five commercial applications [37]. For simulations, we generate a representative slice of 1 billion instructions using Pinpoints. Our evaluations execute the benchmark in rate mode and all cores execute the same benchmark. We perform timing simulation until all the benchmarks in the workload finish execution, and measure the average execution time of all cores.

XI. Results

A. Impact on Performance

Figure 11 shows the impact on execution time for Chipkill and Double-Chipkill-level protection using ECC-DIMMs and compares them to their XED implementations. On a baseline that is normalized to a ECC-DIMM based SECDED, a conventional Chipkill reduces the rank-level parallelism by 2x (by activating two ranks) and increases execution time by 21% on an average. Furthermore, applications that are bandwidth bound (eg. `libquantum`) shows upto 63.5% increase in execution time. Furthermore, even latency sensitive applications like `mcf` shows upto 50.7% increase in execution time. XED activates only a single rank and consumes no performance overheads. The overheads of XED happen only on receiving multiple Catch-Words, something that happens rarely (once every 200K accesses).

For Double-Chipkill, XED on the top of Chipkill activates 18 DRAM-chips (by activating two ranks) instead of to 36 DRAM-chips (by activating four ranks) in traditional Double-Chipkill. Consequently, by activating 18 DRAM-chips, XED based Double-Chipkill has the same overheads as traditional ECC-DIMM based Chipkill. Due to this, XED based Double-Chipkill increases the execution time by 21% which is similar to conventional Chipkill. Unfortunately, traditional Double-Chipkill systems increase the execution time by 82%. Furthermore, bandwidth sensitive applications such as `libquantum` increase the execution time by 220%. Even in latency sensitive benchmarks like `mcf`, a Double-Chipkill increases the execution time by 180%.

B. Impact on Power

Figure 12 shows the impact of memory power while providing Chipkill and Double-Chipkill using ECC-DIMMs when compared to XED based systems. On a baseline that is normalized to an ECC-DIMM based SECDED, a conventional Chipkill not only activates two ranks but also increases execution time. Since power is "energy spent over the total execution" of the application, ECC-DIMM based Chipkill reduces the memory power consumption by 8%. On

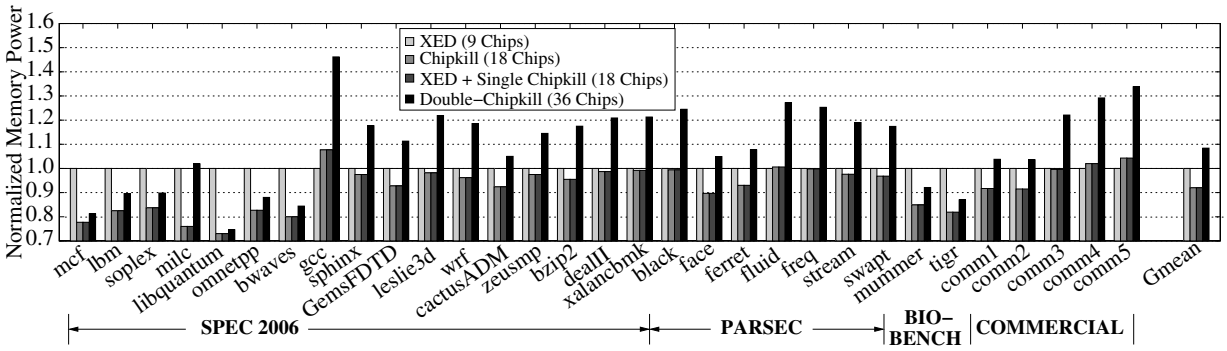


Figure 12. Normalized Memory Power (with respect to ECC-DIMM) for XED, Chipkill, XED on the top of Chipkill and Double-Chipkill.. The reduction in memory power in Chipkill is due to the increased execution time. Double-Chipkill activates two channels and consumes significantly more power.

the contrary, XED consumes the same amount of power as ECC-DIMM based SECDED implementation as it activates only a single rank. Furthermore, because it activates only a single rank, XED also takes almost the same amount of execution time as SECDED systems.

Conventional Double-Chipkill systems consume 8.4% more memory power than ECC-DIMM based SECDED implementation. This is because, even though ECC-DIMM based Double-Chipkill systems increase execution time by 63.5%, they also activate 36-DRAM chips (by activating four ranks). This higher execution time does not compensate for the activation overheads and increases the memory power consumption by 8.4%. XED based Double-Chipkill reduces the memory power consumption by 8% by activating only 18 DRAM-chips instead of 36 DRAM-chips for traditional Double-Chipkill. Furthermore, the likelihood of receiving multiple Catch-Words are rare (1 in every 200K accesses) and therefore they consume negligible power overheads.

C. Impact of adding a Burst or Transaction

XED relies on Catch-Word to convey error detection information. There are alternative ways to convey this information such as using additional bursts or transactions. The memory vendors can change the DDR protocol to expose On-Die ECC information by adding a burst. Adding another burst incurs a 25% overhead in current memory systems as it increases the burst size from 8 to 10. Furthermore, DRAM vendors are reducing the burst-size to one or two [42,43] which would increase this overhead to about 50%-100%. Alternatively, the memory controller can issue another transaction to fetch the On-Die ECC. Figure 13 shows the normalized execution time and power for these two alternatives (additional burst or additional transaction) compared to XED for both Chipkill and Double-Chipkill. Both these alternative implementations increase power consumption and execution time significantly compared to XED implementations for both Chipkill and Double-Chipkill.

The recently introduced DDR4 standards provide an ALERT_n pin [6,11] to indicate errors in address, command,

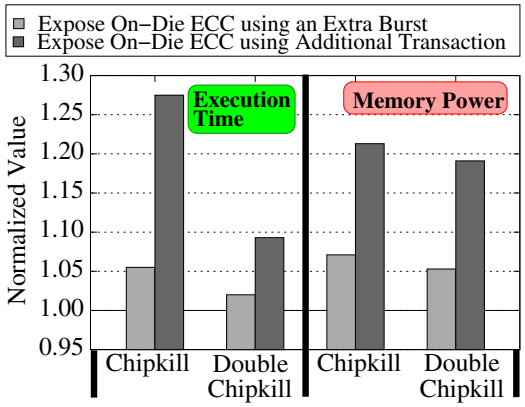


Figure 13. The performance and power overheads of exposing On-Die ECC using adding an additional two bursts or a transaction, instead of XED.

or write operations. As there is only one ALERT_n pin provisioned for the entire DIMM, the ALERT_n signal can only convey that one of the chip is faulty, however it cannot identify the chip that encountered the fault. If future standards [44] could extend the ALERT_n pin to also convey the location of the faulty chip, then XED can be implemented using ALERT_n instead of using Catch-Words.

XII. Related Work

A. Strong Memory Reliability: Orthogonal Proposals

Our paper implements high-reliability memory systems in the presence of On-Die ECC. Several prior studies have looked at enhancing memory reliability, albeit they do not leverage On-Die ECC, and are orthogonal to XED. For instance, Memguard [45] tries to use ordinary Non-ECC DIMMs to provide strong reliability by storing hashes of data and check-pointing data. Memguard stores hashes of data values to detect errors. Memguard does not expose or reuse On-Die ECC and incurs checkpointing overheads for tolerating chip-failures. In a similar vein, COP [46] and Frugal-ECC [47] can use ordinary DIMMs to provide ECC

protection by storing ECC alongside compressed lines. Unlike XED, COP and Frugal-ECC are vulnerable to cachelines are incompressible. XED enables all cachelines, whether they are compressible or not, to be protected and guarantees very high reliability. Virtualized ECC (VECC) [12] enables memory systems to have tiers of ECC and can provide Chipkill-level ECC using x8 DRAM-chips. However, VECC requires support from the OS for managing the locations of these ECC tiers. Bamboo-ECC [48] and ARCC [15] tries to tradeoff reliability with the storage and performance overheads of maintaining ECC. These schemes will benefit from XED as XED can be plugged into these schemes to provide additional reliability.

Prior work have also looked at RAID schemes and applied them to DRAM-DIMMs. Unfortunately, these RAID inspired schemes tend to have read modify write and parity update overheads. For instance, Multi-ECC [49] provides Chipkill using x8 DRAM-chips by using Checksum based detection and parity-based correction. Unfortunately, Multi-ECC has additional write overheads to update the checksum. Another related work is the LOT-ECC [13] design that uses x8 chips to provide Chipkill by having tiers of error detection and correction code. We compare LOT-ECC and with XED. Figure 14 shows the execution time of LOT-ECC and XED when compared to a baseline ECC-DIMM. LOT-ECC has 6.6% higher execution time compared to XED, as it increases the number of writes to the memory system.

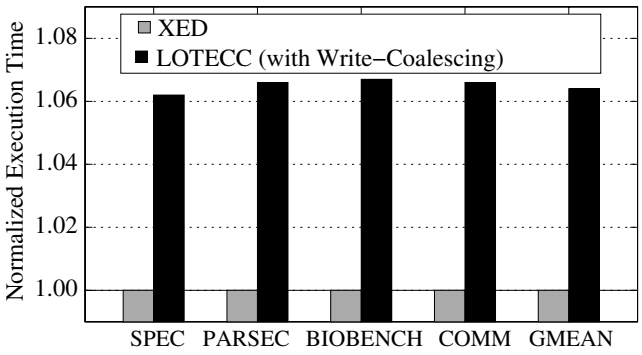


Figure 14. Execution time of LOT-ECC [13] with respect to XED. LOT-ECC causes a slowdown of 6.6%.

B. Enabling DRAM Scaling By Tolerating Faults

Prior works such as Archshield [1] and CiDRA [2] have been proposed to mitigate scaling-faults. ArchShield is designed specifically to handle scaling faults and can tolerate runtime failures at only a single bit granularity. CiDRA also discusses mitigating multiple runtime single-bit failures using On-Die ECC and uses a small SRAM cache to mitigate multi-bit failures. Unfortunately, it is impractical to extend this design to handle a chip failure. For example, to tolerate chip failures, CiDRA will need to provision an SRAM structure that is sized for at-least one DRAM chip (upto a few GBs), incurring prohibitive overheads. On the contrary, XED avoids such SRAM overheads and enables On-Die ECC to be seamlessly used to tolerate both scaling-faults and runtime-faults.

Going forward, Citadel [16], Freefault [50] and Parity Helix [51] tries to address large-granularity faults in stacked memories. XED can be used with these techniques to provide higher reliability even for stacked memories.

XIII. Summary

As DRAM technology scales to smaller nodes, the rate of unreliable bits within the DRAM chips is increasing [3,22]. Memory vendors are planning to provision *On-Die ECC* to handle the scaling-induced faulty bits [3,5,6]. To maintain compatibility with DDR standards, and to avoid the bandwidth overheads of transmitting the ECC code, the On-Die ECC information is not currently exposed to the memory controller and therefore, this information cannot be used to improve memory reliability. To enable low-cost higher-reliability memory systems in presence of On-Die ECC, this paper proposes proposes *XED* (pronounced as "zed", the British pronunciation of the letter "z"), a technique that eXposes On-Die Error Detection information to the memory controller while avoiding the bandwidth overheads and changes to the memory standards. Our proposed implementation of *XED* has the following features:

1) XED exposes On-Die error detection information using *Catch-Words*, thereby avoiding any changes to the DDR protocol or incurring bandwidth overheads.
2) XED uses the 9-th chip in the ECC-DIMM to store parity information of all the chips, and uses the error detection information from the On-Die ECC to correct the data from the faulty chip using a RAID-3 scheme.
3) XED not only tolerates chip-failure, but also mitigate scaling faults even at very high error rates (10^{-4}).

XED provides Chipkill-level reliability using only a single 9-chip ECC-DIMM, and Double-Chipkill on a conventional implementation of Single-Chipkill. Our reliability evaluations show that XED provides 172x higher reliability than an ECC-DIMM and reduces execution time by 21% compared to traditional Chipkill implementations. As DRAM technology ventures into sub 20nm regime, we believe solutions such as XED that spans across multiple sub-systems will become necessary to provide high reliability at low-cost.

Acknowledgment

We thank the anonymous reviewers for their valuable comments and feedback. We also thank our shepherd, Mattan Erez, for his comments and feedback which was helpful in shaping our paper. We are grateful to all members of our research lab and Kevin Lepak of AMD research for providing insightful feedback. This work was supported in part by NSF grant 1319587 and C-FAR, one of the six SRC STARnet Centers, sponsored by MARCO and DARPA.

REFERENCES

[1] P. J. Nair, D.-H. Kim, and M. K. Qureshi, "Archshield: architectural framework for assisting dram scaling by tolerating high error rates," in *ISCA 2013*.

[2] Y. H. Son *et al.*, "Cidra: A cache-inspired dram resilience architecture," in *HPCA 2015*.

[3] K. Uksong *et al.*, "Co-architecting controllers and DRAM to enhance DRAM process scaling," in *The Memory Forum, ISCA*, 2014.

[4] R. W. HAMMING, "Error detecting and error correcting codes," *BELL SYSTEM TECHNICAL JOURNAL*, vol. 29, no. 2, pp. 147–160, 1950.

[5] M. Greenberg, "Reliability, availability, and serviceability (ras) for ddr dram interfaces," in *memcon*, 2014.

[6] T.-Y. Oh *et al.*, "25.1 a 3.2gb/s/pin 8gb 1.0v lpddr4 sdram with integrated ecc engine for sub-1v dram core operation," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014 IEEE International*, Feb 2014, pp. 430–431.

[7] V. Sridharan and D. Liberty, "A study of dram failures in the field," in *SC 2012*.

[8] V. Sridharan *et al.*, "Feng shui of supercomputer memory: Positional effects in dram and sram faults," in *SC 2013*.

[9] ——, "Memory errors in modern systems: The good, the bad, and the ugly," in *ASPLOS 2015*.

[10] JEDEC Standard, "DDR3 Standard," in *JESD79-3E*, 2015.

[11] ——, "DDR4 Standard," in *JESD79-4*, 2015.

[12] D. H. Yoon and M. Erez, "Virtualized and flexible ecc for main memory," in *ASPLOS 2010*.

[13] A. Udipi *et al.*, "Lot-ecc: Localized and tiered reliability mechanisms for commodity memory systems," in *ISCA 2012*.

[14] S. Li *et al.*, "System implications of memory reliability in exascale computing," in *SC 2011*.

[15] X. Jian and R. Kumar, "Adaptive reliability chipkill correct (arcc)," in *HPCA 2013*.

[16] P. J. Nair, D. A. Roberts, and M. K. Qureshi, "Citadel: Efficiently protecting stacked memory from large granularity failures," in *MICRO 2014*.

[17] B. L. Jacob, S. W. Ng, and D. T. Wang, *Memory Systems: Cache, DRAM, Disk*. Morgan Kaufmann, 2008.

[18] Y. Kim *et al.*, "A case for exploiting subarray-level parallelism (salp) in dram," in *ISCA 2012*.

[19] A. N. Udipi *et al.*, "Rethinking dram design and organization for energy-constrained multi-cores," in *ISCA 2010*.

[20] T. Zhang *et al.*, "Half-dram: A high-bandwidth and low-power dram architecture from the rethinking of fine-grained activation," in *ISCA 2014*.

[21] H. Zheng *et al.*, "Mini-rank: Adaptive dram architecture for improving memory power efficiency," in *MICRO 2008*.

[22] S. Hong, "Memory technology trend and future challenges," in *Electron Devices Meeting (IEDM), 2010 IEEE International*, Dec 2010, pp. 12.4.1–12.4.4.

[23] B. Gu *et al.*, "Challenges and future directions of laser fuse processing in memory repair," *Proc. Semicon China*, 2003.

[24] K. Takeuchi *et al.*, "Alpha-particle-induced charge collection measurements for megabit dram cells," *Electron Devices, IEEE Transactions on*, Sep 1989.

[25] M. K. Qureshi *et al.*, "Avatar: A variable-retention-time (vrt) aware refresh for dram systems," in *DSN 2015*.

[26] C. Chen and M. Hsiao, "Error-correcting codes for semiconductor memory applications: a state-of-the-art review," *IBM Journal*, vol. 28, no. 2, pp. 124–134, March 1984.

[27] R. T. Chien, "Cyclic decoding procedures for bose-chaudhuri-hocquenghem codes," in *IEEE Transactions on Information Theory*, vol. 10, no. 4, Oct 1964, pp. 357–363.

[28] R. Bose and D. Ray-Chaudhuri, "On a class of error correcting binary group codes," *Information and Control*, vol. 3, no. 1, pp. 68 – 79, 1960.

[29] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the society for industrial and applied mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[30] J. Nerl *et al.*, "System and method for controlling application of an error correction code (ecc) algorithm in a memory subsystem," Patent US 7 437 651 B2.

[31] ——, "System and method for applying error correction code (ecc) erasure mode and clearing recorded information from a page deallocation table," Patent US 7 313 749 B2.

[32] P. J. Nair, D. A. Roberts, and M. K. Qureshi, "Faultsim: A fast, configurable memory-reliability simulator for conventional and 3d-stacked systems," in *ACM-TACO 2015*.

[33] E. Marcus and H. Stern, *Blueprints for High Availability*. Wiley, 2003.

[34] B. Lin, "Correcting single-bit errors with crc8 in atm cell headers," Freescale Semiconductor, Inc., Tech. Rep., 2005.

[35] INTERNATIONAL TELECOMMUNICATION UNION (ITU), "Series i: Integrated services digital network isdn user -network interfaces - layer 1 recommendations," ITU-T, Tech. Rep. I.432.1, 1999.

[36] N. Chatterjee *et al.*, "Usimm: the utah simulated memory module," University of Utah and Intel Corp, Tech. Rep. UUCS-12-002, Feb. 2012.

[37] (2012) Memory scheduling championship (msc).

[38] *TN-41-01: Calculating Memory System Power for DDR3: Rev. B 8/07 EN*, Micron Technology Inc, 2007.

[39] "Spec cpu2006 benchmark suite," in *Standard Performance Evaluation Corporation*.

[40] C. Bienia, "Benchmarking modern multiprocessors," in *Ph.D. Thesis, Princeton University*, 2011.

[41] K. Albayraktaroglu *et al.*, "Biobench: A benchmark suite of bioinformatics applications."

[42] JEDEC Standard, "High Bandwidth Memory (HBM) DRAM," in *JESD235*, 2013.

[43] ——, "WIDE-IO DRAM," in *JESD229*, 2013.

[44] S. Kwon, Y. H. Son, and J. H. Ahn, "Understanding ddr4 in pursuit of in-dram ecc," in *SoC Design Conference (ISOCC), 2014 International*, 2014, pp. 276–277.

[45] L. Chen and Z. Zhang, "Memguard: A low cost and energy efficient design to support and enhance memory system reliability," in *ISCA 2014*.

[46] D. J. Palframan, N. S. Kim, and M. H. Lipasti, "Cop: To compress and protect main memory," in *ISCA 2015*.

[47] J. Kim *et al.*, "Frugal ecc: Efficient and versatile memory error protection through fine-grained compression," in *SC 2015*.

[48] ——, "Bamboo ecc: Strong, safe, and flexible codes for reliable computer memory," in *HPCA 2015*.

[49] X. Jian *et al.*, "Low-power, low-storage-overhead chipkill correct via multi-line error correction," in *SC 2013*.

[50] D. W. Kim and M. Erez, "Balancing reliability, cost, and performance tradeoffs with freefault," in *HPCA 2015*.

[51] X. Jian, V. Sridharan, and R. Kumar, "Parity helix: Efficient protection for single-dimensional faults in multi-dimensional memory systems," in *HPCA 2016*.