XED: EXPOSING ON-DIE ERROR DETECTION INFORMATION FOR STRONG MEMORY RELIABILITY

Prashant Nair, Georgia Tech
Vilas Sridharan, AMD Inc.
Moinuddin Qureshi, Georgia Tech

ISCA-43, June 20th 2016
Seoul, Republic of Korea
ArchShield is a novel architecture-level framework tailored to the vulnerability level of each word. We propose using DIMMs to tolerate both faulty cells due to manufacturing and additional ECC to tolerate soft errors. Ideally, we want to use soft error resilience for tolerating faulty cells, and would only use ECC when the error rate is higher. For tolerating an error-rate of more than one error per million, we would need higher levels of ECC. For errors due to alpha particles, we would need 3-bit error correction per word. Such high level of ECC is expensive in terms of storage and latency. Furthermore, this approach sacrifices performance per 64-bit word. This reduces the effective main memory visible to the operating system. Fortunately, the visible address space can be emulated by the ArchShield system.

Another alternative to tolerate errors in DRAM is to use Error Correcting Code (ECC) DIMMs, which can correct one bit out of the 8-byte word. While this works well for words with one-bit error, it is still ineffective for high bit error rates. Two types of DRAM faults are possible: random bit errors and words with multi-bit error. On a memory access, the fault may have a word with 1-bit error, the replicated copy is accessed to obtain the replicated data. In the case of words with multi-bit error, while providing soft error protection, we can also use selective Word Level Replication (SWLR) to the memory controller to do runtime testing, orchestrate word-level repair, and provide protection against soft errors. ArchShield requires 4% memory overhead and 2% overhead due to the extra memory traffic of Fault Map and SWLR. ArchShield provides this while maintaining a soft error protection of 100x higher error-rates than can be handled by ECC DIMMs. We also show how ArchShield can be used to reduce refresh operations in DRAM systems. With ArchShield, the system can tolerate a word with 1-bit error even when an uncorrectable fault is encountered at the original location, without any software changes (except that the memory is deemed to smaller nodes).

The Fault Map and word-level repair of ArchShield is inspired by similar approaches to dealing with high error-rate problems in magnets and other devices. ArchShield contains a Fault Map, which stores information about the faulty DRAM of the DIMM to identify the faulty cells in the memory. In part, ArchShield is designed to have smaller capacity. Similarly, ArchShield does not require any intervention from the operating system. Fortunately, the visible address space can be emulated by the ArchShield system.

We perform evaluations with 8GB DIMM. We show that for tolerating an error-rate as high as 10^6 per million, we need 3-bit error correction per word. The third problem is the increase in gate induced drain leakage due to the challenges from shrinking dimensions and variability. The first problem is the increase in dielectric material. As shown in Figure 2, the aspect ratio has been increased to meet the DRAM retention time, the capacitance stored on the DRAM device needs to be approximately doubled to obtain the same capacitance value, given the unreliability of the undoped polysilicon layer. To meet the DRAM retention time, the capacitance must be increased. In addition to the typical problems of shrinking dimensions and variability, the vertical structure must be doubled to obtain the same capacitance value.
INTRODUCTION

DRAM Scaling → High Capacity Memories

Two types of DRAM faults

[ArchShield ISCA’13, CiDRA HPCA’15]
DRAM vendors plan to use “On-Die ECC”
• Mitigates scaling faults transparently
• Enables good DIMM with bad chips (yield)
• Part of: LPDDR4, DDR4, DDR5 (proposed)
ON-DIE ECC: MITIGATE SCALING FAULTS

x8 DIMM

DATA

REQUEST-A
ON-DIE ECC: MITIGATE SCALING FAULTS

x8 DIMM

REQUEST-A
ON-DIE ECC: MITIGATE SCALING FAULTS

On-Die ECC: Single Error Correction, Double Error Detection Code (SECDED)
ON-DIE ECC: MITIGATE SCALING FAULTS

On-Die ECC fixes scaling faults invisibly
MITIGATING RUNTIME FAULTS

Runtime faults

<table>
<thead>
<tr>
<th>Fault Mode</th>
<th>Transient Fault Rate (FIT)</th>
<th>Permanent Fault Rate (FIT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit</td>
<td>14.2</td>
<td>18.6</td>
</tr>
</tbody>
</table>

ECC-DIMM (9-Chips)
**Runtime faults**

- Chip faults common
- Need strong ECC

<table>
<thead>
<tr>
<th>Fault Mode</th>
<th>Transient Fault Rate (FIT)</th>
<th>Permanent Fault Rate (FIT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit</td>
<td>14.2</td>
<td>18.6</td>
</tr>
<tr>
<td>Word</td>
<td>1.4</td>
<td>0.3</td>
</tr>
<tr>
<td>Column</td>
<td>1.4</td>
<td>5.6</td>
</tr>
<tr>
<td>Row</td>
<td>0.2</td>
<td>8.2</td>
</tr>
<tr>
<td>Bank</td>
<td>0.8</td>
<td>10</td>
</tr>
<tr>
<td>*Total</td>
<td>18</td>
<td>42.7</td>
</tr>
</tbody>
</table>

*MITIGATING RUNTIME FAULTS*
Runtime chip faults $\rightarrow$ Chipkill (strong ECC)
Runtime chip faults → Chipkill (strong ECC)

Cost: 18 Chips, Performance and Power Inefficient
GOAL: Use On-Die ECC to mitigate runtime faults “Chipkill-level reliability using x8 ECC-DIMM”

CHALLENGE: On-Die ECC is invisible, expose it without changing the memory interface
OUTLINE

• BACKGROUND

• XED

• CASE STUDIES

• EVALUATION

• SUMMARY
What if the chip can inform that it failed?
What if the chip can inform that it failed?

Parity + Location → Reconstruct Data for Faulty Chip

Fix chip-faults using only 9 Chips
XED consists of three components
• Strong detection in addition to SEC
• Parity-based correction
• Transparently identifying faulty chip
**On-Die Error Correction Code**

<table>
<thead>
<tr>
<th></th>
<th>Corrects?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-Bit Failures</td>
<td>✓</td>
</tr>
<tr>
<td>Chip Failures</td>
<td>✗</td>
</tr>
</tbody>
</table>

Diagram showing a 64-Bits Data input, with arrows indicating Detect and Correct processes.
XED: ON-DIE ECC AS DETECTION CODE

On-Die Error Strong Detection + Correction Code

<table>
<thead>
<tr>
<th></th>
<th>Corrects?</th>
<th>Detects?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-Bit Failures</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Chip Failures</td>
<td>✗</td>
<td>✔️ (99.9%)</td>
</tr>
</tbody>
</table>

CRC-8 ATM-code instead of Hamming-code

On-Die ECC can detect chip-failures
XED: RAID-3 BASED CORRECTION

If we could expose On-Die Error Detection → Chipkill

On-Die ECC detected it

Reconstruct Data in Failed Chip
OPTION 1: Use additional wires

EXPOSE ON-DIE ERROR INFO

Memory Controller
OPTION 1: Use additional wires

Incompatible with DDR memory standards

Needs a new protocol

Worse for pin-constrained future systems!
OPTION 2: Use additional burst/transaction
OPTION 2: Use additional burst/transaction
OPTION 2: Use additional burst/transaction

Additional 12.5% to 100% bandwidth overheads

Performance and Power Inefficient

Expose On-Die error detection with minor changes
On detecting an error, the DRAM chip sends a 64-bit “Catch-Word” (CW) instead of data.
XED: MUX TO SEND CATCH-WORDS

Simple MUX to choose between Data and Catch-Word
On detecting an error, the DRAM chip sends a 64-bit “Catch-Word” (CW) instead of data.

- Chips provisioned with a unique Catch-Word
- No additional wires/bandwidth overheads
- Compatible with existing memory protocols

64-bit Catch-Words identify the faulty chip
WHY DO CATCH-WORDS WORK?

Catch Word (CW) ≠ Valid Data (D2)
WHY DO CATCH-WORDS WORK?

Catch Word (CW) ≠ Valid Data (D2)

Then $\rightarrow$ \( PA \neq D0 \oplus D1 \oplus CW \oplus \ldots \oplus D7 \)
WHY DO CATCH-WORDS WORK?

Catch Word (CW) ≠ Valid Data (D2)

Then → PA ≠ D0 ⊕ D1 ⊕ CW ⊕ ... ⊕ D7

Location Identified

\[ D2 = D0 \oplus D1 \oplus D3 \oplus ... \oplus PA \]
WHY DO CATCH-WORDS WORK?

Catch Word (CW) = Valid Data (D2)
WHY DO CATCH-WORDS WORK?

Catch Word (CW) = Valid Data (D2) [Collision]

Then $\rightarrow$ $PA = D0 \oplus D1 \oplus CW \oplus ... \oplus D7$

No Error as Parity Matches

Catch-Word collision: Doesn’t affect correctness
COLLISIONS: NOT A PROBLEM

• A chip stores 64 bits/cache-line $\Rightarrow 2^{64}$ combinations
• However even a 16Gb chip has only $2^{28}$ cachelines
• Even if this entire chip contained different data there are nearly $2^{63.99}$ data combinations free!

The catch-word will most likely not collide
OUTLINE

• BACKGROUND

• XED

• CASE STUDIES

• EVALUATION

• SUMMARY
XED FOR SCALING ERRORS

On-Die ECC

- Single Error Correction
- Always detects scaling errors (single-bit)
CASE STUDY 1: SINGLE SCALING FAULT

Scaling fault within a single chip

Parity reconstructs data from chip with scaling error
Scaling faults within multiple chips

Memory Controller: No SDC, No DUE

Disable XED + Retry
CASE STUDY 3: CHIP FAULT

Catch-Word identifies the faulty chip

Parity reconstructs data from failed chip

No SDC, No DUE
CASE STUDY 4: CHIP + SCALING FAULT

Parity detects error even after retry → Chip Failure

Memory Controller

Very Small
SDC (10^{-6}), DUE (10^{-13})

Disable XED + Diagnosis to locate chip failure
OUTLINE

• BACKGROUND
• XED
• CASE STUDIES
• EVALUATION
• SUMMARY
USIMM: 8 Cores, 4 Channels, 2 Ranks, 8 Banks

FaultSim*: Memory Reliability Simulator
- Real World Fault Data
- 7 year system lifetime,
- Billion Monte-Carlo Trails
- Metric: Probability of System Failure
- Scaling Fault-Rate: $10^{-4}$

* Nair et. al. HiPEAC 2016
XED vs Commercial ECC schemes

<table>
<thead>
<tr>
<th>Probability of System Failure (Log Scale)</th>
<th>Years</th>
</tr>
</thead>
<tbody>
<tr>
<td>10^{-5}</td>
<td>1</td>
</tr>
<tr>
<td>10^{-4}</td>
<td>2</td>
</tr>
<tr>
<td>10^{-3}</td>
<td>3</td>
</tr>
<tr>
<td>10^{-2}</td>
<td>4</td>
</tr>
<tr>
<td>10^{-1}</td>
<td>5</td>
</tr>
<tr>
<td>10^{0}</td>
<td>6</td>
</tr>
<tr>
<td>10^{1}</td>
<td>7</td>
</tr>
</tbody>
</table>

**Table:***

<table>
<thead>
<tr>
<th>SECDED: ECC-DIMM (9 Chips)</th>
<th>ChipKill (18 Chips)</th>
<th>XED (9 Chips)</th>
</tr>
</thead>
<tbody>
<tr>
<td>43x</td>
<td>4x</td>
<td>43x</td>
</tr>
</tbody>
</table>

**Figure:**

- **SECDED:** ECC-DIMM (9 Chips)
- **ChipKill (18 Chips):**
- **XED (9 Chips):**

**XED provides strong reliability while using fewer chips**
RESULTS: PERFORMANCE AND EDP

Normalized Execution Time

Lower the better

21%
RESULTS: PERFORMANCE AND EDP

Normalized Execution Time

Lower the better

Normalized Memory-Delay Product (EDP)

Lower the better

Execution time: 21% ↓, EDP: 34% ↓
OUTLINE

• BACKGROUND
• XED
• CASE STUDIES
• EVALUATION
• SUMMARY
SUMMARY

• DRAM Scaling introduces errors → On-Die ECC
• On-Die ECC is invisible to the memory system
• Exposing On-Die ECC: Efficient Runtime ECC
• XED
  – Exposes On-Die Error Detection using Catch-Words
  – 2X fewer chips as compared to Chipkill
  – 4X higher reliability as compared to Chipkill
  – 21% lower execution time as compared to Chipkill
• XED → No change in memory protocols
“You are in a pitiable condition, if you have to conceal what you wish to tell”
- Publilius Syrus
BACKUP
RANDOM DATA?

• What if only half the data is random
  1. Then average time for collision increases by 2x
    (3.2 Million Years → 6.4 Million Years)
  2. Less random data increases collision time

• DIMMs today store scrambled (randomized) data
  1. To equalize the number of 1’s and 0’s
  2. Reduce Bit Error Rate on the bus
  3. Scrambling using address based hash

1. Lower randomization → Longer time till collision
2. Current systems anyway scramble data for fidelity
MTTF: XED VS CHIPKILL

2-Chip Failures

XED (9-chips)  FAILED

Chipkill (18-chips)  FAILED
MTTF: XED VS CHIPKILL

2-Chip Failures

XED (9-chips)

FAILED

Chipkill

FAILED
MTTF: XED VS CHIPKILL

2-Chip Failures → Extend to Multi-Chip Failures

XED (9-chips)

PASSED

Chipkill

FAILED
## SDC AND DUE Rate of XED

<table>
<thead>
<tr>
<th>Source of Vulnerability</th>
<th>Rate over 7 years</th>
</tr>
</thead>
<tbody>
<tr>
<td>XED: Scaling-Related Faults</td>
<td>No SDC or DUE</td>
</tr>
<tr>
<td>XED: Row/Column/Bank Failure</td>
<td>$1.4 \times 10^{-13}$ (SDC)</td>
</tr>
<tr>
<td>XED: Word Failure</td>
<td>$6.1 \times 10^{-6}$ (DUE)</td>
</tr>
<tr>
<td>Data Loss from Multi-Chip Failures</td>
<td>$5.8 \times 10^{-4}$</td>
</tr>
</tbody>
</table>
IN MEMORY POWER IN CHIPKILL IS DUE TO THE INCREASED EXECUTION

FIGURE 12. NORMALIZED MEMORY POWER (WITH RESPECT TO ECC-DIMM) FOR BOTH CHIPKILL AND DOUBLE-CHIPKILL.

ALERT

Figure 12 shows the normalized execution time and power for these two alternatives. Alternatively, the memory controller can issue another transaction to fetch the On-Die ECC. This increases the burst size from 8 to 10. Furthermore, DRAM vendors are reducing the burst-size to one or two [42,43].

On-Die ECC information by adding a burst. Adding another burst incurs a 25% overhead in current memory systems as it is not provisioned for the entire DIMM. 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 execution time significantly compared to XED implementations for both Chipkill and Double-Chipkill. Both these alternatives increase power consumption and execution time significantly compared to XED implementations for both Chipkill and Double-Chipkill. Furthermore, the likelihood of receiving correct catch-words is rare (1 in every 200K accesses). In a similar vein, COP [46] and Memguard store hashes of data values to detect errors. Memguard does not expose the location of the faulty chip, then XED can be implemented using ALERT instead of XED. Frugal-ECC [47] can use ordinary DIMMs to provide ECC for tolerating chip-failures. In a similar vein, COP [46] and Memguard 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 checkpointing data. Memguard stores hashes of

A. STRONG MEMORY RELIABILITY: ORTHOGONAL PROPOSALS

...CONVENTIONAL DOUBLE-CHIPKILL SYSTEMS CONSUME 8.4% EXECUTION TIME AS SECDED SYSTEMS. RANK, XED ALSO TAKES THE SAME AMOUNT OF POWER AS SECDED IMPLEMENTATION AS IT ACTIVATES ONLY A SINGLE RANK. FURTHERMORE, BECAUSE IT ACTIVATES ONLY 18 DRAM-CHIPS INSTEAD OF 36 DRAM-CHIPS FOR TRADITIONAL SECDED IMPLEMENTATION, XED REDUCES THE MEMORY POWER CONSUMPTION BY 8% BY ACTIVATING ONLY 9 DRAM-ChIPS. THE DOUBLE-CHIPKILL SYSTEMS CONSUME 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 EXECUTION TIME AS SECDED SYSTEMS.
SRAM structure that is sized for at least one DRAM chip to tolerate chip failures, CiDRA will need to provision an extra chip to extend this design to handle a chip failure. For example, LOT-ECC has been proposed to mitigate multi-bit failures. Unfortunately, it is impractical to use On-Die ECC and uses a small SRAM cache to tolerate runtime failures at only a single bit granularity. ArcShield also discusses mitigating multiple runtime single-bit failures. Chipkill-level ECC using x8 DRAM-chips by using Checksum based update overheads. For instance, Multi-ECC provides additional write overheads to update the checksum. Chipkill using x8 DRAM-chips by using Checksum based parity information of all the chips, and uses the error detection and parity-based correction. Unfortunately, Multi-ECC tries to tradeoff reliability with the storage and performance. Virtualized ECC (VECC) enables memory systems to have tiers of ECC and can provide very high reliability. Virtualized ECC can be plugged into these schemes to provide additional reliability. Memory vendors are planning to provisions from XED as XED can be plugged into these schemes to improve memory reliability. To enable low-cost higher-reliability memory systems in presence of On-Die ECC, controller and therefore, this information cannot be used. ECC information is not currently exposed to the memory controller while avoiding the bandwidth overheads of transmitting the ECC code, the On-Die ECC has 6.6% higher execution time compared to XED, as ECC causes a slowdown of 6.6%. Prior work have also looked at RAID schemes and applied them to DRAM-DIMMs. Unfortunately, these RAID schemes tend to have read modify write and parity based organisation which is not compatible with DDR standards, and to avoid the bandwidth overheads of maintaining ECC. These schemes will benefit from XED that spans across multiple sub-systems will try to tradeoff reliability with the storage and performance. As DRAM technology scales to smaller nodes, the rate of unreliability bits within the DRAM chips is increasing. Going forward, Citadel, Freefault, and Parity protection by storing ECC alongside compressed lines. Unfortunately, it is impractical to use On-Die ECC to be seamlessly used to tolerate both scaling-faults even at very high error rates (10^−4). XED not only tolerates chip-failure, but also mitigates scaling-faults even at very high error rates (10^−4). XED provides Chipkill-level reliability using only a single chip in the ECC-DIMM and reduces execution time by 21% compared to traditional Chipkill implementations of Single-Chipkill. Our reliability evaluation shows that XED provides 172x higher reliability than an implementation of Single-Chipkill. Our reliability evaluation shows that XED provides 172x higher reliability than an implementation of Single-Chipkill.