Understanding and Modeling On-Die Error Correction in Modern DRAM: An Experimental Study Using Real Devices

Minesh Patel
Hasan Hassan
Jeremie S. Kim
Onur Mutlu
Executive Summary

• **Motivation:** Experimentally studying DRAM error mechanisms provides insights for improving performance, energy, and reliability

• **Problem:** on-die error correction (ECC) makes studying errors difficult
  - Distorts true error distributions with *unstandardized, invisible* ECC functions
  - *Post-correction* errors lack the insights we seek from *pre-correction* errors

• **Goal:** Recover the *pre-correction* information masked by on-die ECC

• **Key Contributions:**
  1. **Error INference (EIN):** statistical inference methodology that:
     • Infers the ECC scheme (i.e., type, word length, strength)
     • Infers the *pre-correction* error characteristics beneath the on-die ECC mechanism
     • Works without any hardware intrusion or insight into the ECC mechanism
  2. **EINSim:** open-source tool for using EIN with real DRAM devices
     • Available at: [https://github.com/CMU-SAFARI/EINSim](https://github.com/CMU-SAFARI/EINSim)
  3. **Experimental demonstration:** using 314 LPDDR4 devices
     • EIN infers (i) the on-die ECC scheme and (ii) *pre-correction* error characteristics

We hope EIN and EINSim enable many valuable studies going forward
1. Error Characterization and On-Die ECC

2. EIN: Error INference
   I. The Inference Problem
   II. Formalization
   III. EIN in Practice: EINSim

3. Demonstration Using LPDDR4 Devices
What is DRAM Error Characterization?

Studying how DRAM behaves when we deliberately induce *bit-flips*

Operating Timing Constraints

Error Mechanisms & Technology Scaling

System-Level Interactions

Environmental Effects

Understanding + Exploitable Insights

**SAFARI**
How Do We Characterize DRAM?

**Test Routine**
1. Write data
2. Induce errors
3. Read data
4. Record errors

**Tester**
e.g., CPU, FPGA

**Error Distributions**

**Spatial Distributions**

**Temporal Distributions**

**Cell-to-cell Variation**

**Device Comparisons**

SAFARI
Why Study DRAM Errors?

- Errors provide insight into how a DRAM device works
  - Error mechanisms are based on physical phenomena
  - Patterns in errors can indicate opportunity for improvement

- Reliability
  - e.g., Defending against vulnerabilities (e.g., RowHammer)

- Energy
  - e.g., Reducing the cost of refresh and other operations

- Performance
  - e.g., Reliably reducing conservative operating timings

- Reliability
  - e.g., Efficiently profiling for and mitigating errors

Characterization-Driven Insights
Three Key Types of DRAM

No ECC (Standard)

- Data Tester
- Raw Data Is Unmodified

Rank-Level ECC (Server-style)

- Data Tester
- ECC Logic
- Raw Data Is Unmodified

On-Die ECC (or Integrated ECC)

- Data Tester
- ECC Logic
- ECC Modifies Raw Data
- Corrected Data

SAFARI
Three Key Types of DRAM

No ECC (Standard)

Rank-Level ECC (Server-style)

On-Die ECC (or Integrated ECC)

Unfortunately, the on-die ECC scheme:
1. **Cannot** be bypassed
2. Is **unknown** and proprietary
3. Is completely **invisible**
ECC Complicates Error Characterization

Original Data

1111

Scheme A
Encoder A
1011001
Decoder A (SEC)
1011
1 Error

Scheme B
Encoder B
0001100
Decoder B (SEC)
1111
0 Errors

Scheme C
Encoder C
0011001
Decoder C (SEC)
1010
2 Errors

SAFARI
ECC Complicates Error Characterization

Observed errors can change depending on the ECC scheme

Original Data

1111

Scheme A
Encoder A

Scheme B
Encoder B

Scheme C
Encoder C

Decoder A (SEC)

Decoder B (SEC)

Decoder C (SEC)

1011
1 Error

1111
0 Errors

1010
2 Errors
ECC Makes Error Characterization Difficult

- ECC causes two key problems:
  - Prevents comparing error characteristics between devices
  - Obfuscates the well-studied error distributions we expect

**ECC Makes Error Characterization Difficult**

- Pre-ECC Error Distribution
  - Based on a physical DRAM error mechanism
- Post-ECC Error Distribution
  - ECC-scheme specific; Error mechanism influence lost

**• ECC causes two key problems:**

- Prevents comparing error characteristics between devices
- Obfuscates the well-studied error distributions we expect
Example: Technology Scaling Study

- Goal: study how errors evolve over technology generations

![Graph showing distribution of all bit-errors and pre- and post-ECC error distributions across technology generations.](image)

- Consistent with a physical phenomenon (e.g., process scaling)
- Post-ECC error distributions
- ECC artifact
- Pre-ECC error distributions

SAFARI
Example: Technology Scaling Study

- Goal: study how errors evolve over technology generations

Our goal:
Recover pre-correction error characteristics obfuscated by on-die ECC

Consistent with a physical phenomenon (e.g., process scaling)

Pre-ECC error distributions

Post-ECC error distributions

ECC artifact

Distribution of all bit-errors
Presentation Outline

1. Error Characterization and On-Die ECC

2. EIN: Error INference
   I. The Inference Problem
   II. Formalization
   III. EIN in Practice: EINSim

3. Demonstration Using LPDDR4 Devices
Key Observation

DRAM error mechanisms have predictable characteristics that are intrinsic to DRAM technology.
Example: Data-Retention Errors

- DRAM encodes data in **leaky capacitors**
- Leakage rates differ due to **process variation**
- Necessitates periodic **refresh operations**

- By **disabling** refresh, we induce **data-retention errors**
- Well-studied and fundamental to DRAM technology
- Errors exhibit **predictable** statistical characteristics
  - **Exponential** bit-error rate (BER) with respect to temperature
  - **Uniform-random** spatial distribution

SAFARI
Inferring the ECC Scheme

- Exploit error characteristics to **infer** the ECC scheme
  - Works for any DRAM susceptible to the error mechanism
  - Independent of any particular device or manufacturer
Inferring the ECC Scheme

- Exploit error characteristics to infer the ECC scheme
  - Works for any DRAM susceptible to the error mechanism
  - Independent of any particular device or manufacturer

**EIN’s key idea:** use predictable error characteristics to infer:
(i) the ECC scheme
(ii) the pre-correction error rate
Presentation Outline

1. Error Characterization and On-Die ECC

2. EIN: Error INference
   I. The Inference Problem
   II. Formalization
   III. EIN in Practice: EINSim

3. Demonstration Using LPDDR4 Devices
Formalizing the Inference Problem

• Model the entire DRAM transformation as a function:

\[ w' = f(w \mid S, \theta) \]

We want to infer \( \{S, \theta\} \) given observed \( \{w, w'\} \)
Formalizing the Inference Problem

\[ w' = f(w \mid S, \theta) \]

- **S**: ECC encoding/decoding algorithms
- **\( \theta \)**: Spatial distribution of errors (e.g., uniform-random)
- **\( w, w' \)**: Probability of each value (i.e., 0x0, 0x1, ...)
  - \( w \) is typically defined by the data pattern we write
Formalizing the Inference Problem

\[ w' = f(w \mid S, \theta) \]

- **Unfortunately**: \( w' \) is hard to measure
  - 64-bit dataword \( \rightarrow 2^{64} \) possible values
  - Typical 8GiB DRAM only has \( \sim 2^{30} \) datawords \( (\ll 2^{64}) \)
  - Hard to get a representative sample of \( w' \) even with all 8GiB

- \( w_N' \): Probability that \( w' \) has \( N \in [0, 1, \ldots, n] \) errors
  - Easy to experimentally measure: simply count errors
  - Meaningful in the context of ECC (e.g., \( n \)-error correction)
Inferring the ECC Scheme

Want the **most likely** ECC scheme **given** an experiment

\[
\text{argmax}_S P[S \mid X] = \text{argmax}_S P[X \mid S] \ast P[S]
\]

- **ECC Scheme**
- **Experiment** \(\theta\): error distribution
  - \(w, w'_N\): inputs/outputs
- **Bayes’ Theorem**
- **Likelihood**
  - Are these results reasonable?
- **Prior**
  - How likely is \(S\)?

- This is a **maximum-a-posteriori** (MAP) estimation
- We provide a rigorous derivation in the paper
  - Full optimization objective function
  - **Extension** for inferring error distribution characteristics \(\theta\)

SAFARI
Error INference (EIN) Methodology

1. Define experimental inputs (i.e., data pattern, error mechanism)
2. Identify candidate ECC Schemes
3. Run Experiments
4. Compute MAP estimation

Most likely ECC scheme
Presentation Outline

1. Error Characterization and On-Die ECC

2. EIN: Error INference
   I. The Inference Problem
   II. Formalization
   III. EIN in Practice: EINSim

3. Demonstration Using LPDDR4 Devices
MAP Estimation in Practice

1. **Input**
   (i.e., data pattern, error distribution)

2. **Suspected ECC Schemes**
   \{A, B, C, D, \ldots\}

3. **Device to Test**
   (unknown ECC scheme)

4. **Monte-Carlo Simulation**
   \((EINSim)\)

   **Calculation**
   Hard to calculate analytically

   **Outputs**
   (# errors per word)

   **Likelihood per Scheme**

   **Experiment**
   Most Likely ECC Scheme

   **Observations**
   (# errors per word)
EINSim: A Tool for Using EIN

• Evaluates MAP estimation via Monte-Carlo simulation
  - Simulates the life of a dataword through a real experiment
  - Configuration knobs to replicate the experimental setup

• Flexible and extensible to apply to a wide variety of:
  - DRAM devices
  - Error mechanisms
  - ECC schemes

Open-source C++/Python project
https://github.com/CMU-SAFARI/EINSim

• Example datasets provided (same as used in paper)
EINSim: A Tool for Using EIN

• Evaluates MAP estimation via Monte-Carlo simulation
• Simulates the life of a dataword through a real experiment
• Configuration knobs to replicate the experimental setup

• Flexible and configurable to apply to a wide variety of:
  - DRAM devices
  - Error mechanisms
  - ECC schemes

Open-source C++/Python project
https://github.com/CMU-SAFAReIINSim

Example datasets provided (same as used in paper)

Give EINSim a try at:
https://github.com/CMU-SAFAReIINSim
1. Error Characterization and On-Die ECC

2. EIN: Error INference
   I. The Inference Problem
   II. Formalization
   III. EIN in Practice: EINSim

3. Demonstration Using LPDDR4 Devices
Methodology

- We experimentally test LPDDR4 DRAM devices
  - 232 with on-die ECC (one major manufacturer)
  - 82 without on-die ECC (three major manufacturers)

- Thermally controlled testing chamber
  - 55°C - 70°C
  - Tolerance of ±1°C

- Precise control over the commands sent to DRAM
  - Ability to enable/disable self-/auto-refresh
  - Control over CAS (i.e., read/write) commands
# Experimental Design

**Goal:** infer which ECC scheme is used in real LPDDR4 devices with on-die ECC

<table>
<thead>
<tr>
<th><strong>Parameter</strong></th>
<th><strong>Experiment</strong></th>
<th><strong>Simulation (EINSim)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Word Size</td>
<td>256 bits</td>
<td>256 bits</td>
</tr>
<tr>
<td>ECC Schemes</td>
<td><strong>Unknown</strong></td>
<td>Hamming (32, 64, 128, 256) BCH-2EC (32, 64, 128, 256) BCH-3EC (32, 64, 128, 256) Repetition (3, 5, 7)</td>
</tr>
<tr>
<td>Data Pattern</td>
<td>RANDOM</td>
<td>RANDOM, 0xFF</td>
</tr>
<tr>
<td>Error Mechanism</td>
<td>Data-Retention</td>
<td>Data-Retention</td>
</tr>
</tbody>
</table>
MAP Estimation Methodology

• Assume a uniform prior distribution
  - Avoids biasing results towards our preconceptions
  - Demonstrates EIN in the worst case

• Simulate $10^6$ 256-bit words per ECC scheme

• Error estimation using bootstrapping ($10^4$ samples)
MAP Estimation Results

- Log(Likelihood)

0.4×10^7
0.2×10^7
0.0×10^7

Most Likely ECC Scheme

ECC Schemes
(#code bits, #data bits, #errors correctable)

Lower is MORE Likely

Confidence interval is extremely tight
MAP Estimation Results

ECC Schemes

\[ \text{(\#code bits, \#data bits, \#errors correctable)} \]

- Log(Likelihood)

0.0 \times 10^7
0.2 \times 10^7
0.4 \times 10^7

BCH(44, 32, 2)
BCH(78, 64, 2)
Ham(38, 32, 1)
Ham(71, 64, 1)
Ham(265, 256, 1)
BCH(274, 256, 2)
BCH(144, 128, 2)
Ham(136, 128, 1)

Most Likely ECC Scheme

Confidence interval is extremely tight

Lower is MORE Likely

Less Likely Models
EIN effectively infers the ECC scheme in LPDDR4 devices with on-die ECC to be a (128 + 8) Hamming Code

EIN infers the ECC scheme without:
- Visibility into the ECC mechanism
- Disabling ECC
- Tampering with the hardware
EIN Applies Beyond On-Die ECC

- EIN technically applies for any device for which:
  - Communication channel protected by ECC
  - Can induce uncorrectable errors
  - Errors follow predictable statistical characteristics

**DRAM Rank-Level ECC**

**Flash Memory ECC**

- NAND Flash

---

**Diagram:**

- CPU
- Normal Data
- ECC Data
- NAND Flash
Other Contributions in our Paper

• Two error-characterization studies showing EIN’s value
  1. EIN enables comparing BERs of the DRAM technology itself
  2. EIN recovers expected distributions that ECC obfuscates

• Using EIN to infer additional information:
  - The data pattern written to DRAM
  - The pre-correction error characteristics (e.g., pre-ECC BER)

• Formal derivation of EIN + discussion of its limitations

• Verify uniform-randomly spaced data-retention errors
  - Reverse-engineering DRAM design characteristics that affect uniformness (e.g., true-/anti-cell layout)
Talk & Paper Recap

**Motivation:** Experimentally studying DRAM error mechanisms provides insights for improving performance, energy, and reliability

**Problem:** on-die error correction (ECC) makes studying errors difficult
- Distorts true error distributions with *unstandardized, invisible* ECC functions
- *Post-correction* errors lack the insights we seek from *pre-correction* errors

**Goal:** Recover the *pre-correction* information masked by on-die ECC

**Key Contributions:**
1. **Error INference (EIN):** statistical inference methodology that:
   - Infers the ECC scheme (i.e., type, word length, strength)
   - Infers the *pre-correction* error characteristics beneath the on-die ECC mechanism
   - Works without any hardware intrusion or insight into the ECC mechanism

2. **EINSim:** open-source tool for using EIN with real DRAM devices
   - Available at: [https://github.com/CMU-SAFARI/EINSim](https://github.com/CMU-SAFARI/EINSim)

3. **Experimental demonstration:** using 314 LPDDR4 devices
   - EIN infers (i) the on-die ECC scheme and (ii) *pre-correction* error characteristics

We hope EIN and EINSim enable many valuable studies going forward
Understanding and Modeling On-Die Error Correction in Modern DRAM:
An Experimental Study Using Real Devices

Minesh Patel       Jeremie S. Kim
Hasan Hassan       Onur Mutlu

ETH Zürich       SAFARI
Backup
Slides
EIN: 3 Concrete Use Cases

1. Rapid error profiling using statistical distributions
   - Use properties of the *error mechanisms* to model errors
   - Use EIN to determine model parameters at runtime
   - Replacement for laborious, per-device characterization

2. Comparison studies (e.g., technology scaling)
   - Use EIN to compare *pre-correction* error rates
   - Study + predict industry and future technology trends

3. Reverse-engineering proprietary ECC schemes
   - Applies beyond just DRAM with on-die ECC
   - Can be useful for security research
   - E.g., vulnerability evaluation, patent infringement, competitive analysis, forensic analysis
Observed BER Depends on ECC

Assume errors occur independently, uniform-randomly
- Fixed per-bit \( P[\text{error}] \) = “bit error rate” (BER)

\[
\begin{align*}
\text{BCH}(64, 2) & \quad \text{Hamming}(32, 1) & \quad \text{Hamming}(256, 1) \\
\text{BCH}(128, 2) & \quad \text{Hamming}(64, 1) \\
\text{BCH}(256, 2) & \quad \text{Hamming}(128, 1) & \quad 3\text{-Repetition} & \quad \text{None}
\end{align*}
\]

\( \text{ECC}(k, t): \)

\( k = \# \text{ data bits} \)

\( t = \# \text{ correctable errors} \)
A Closer Look at On-Die ECC

Primarily mitigates technology scaling issues [1]
- Transparently mitigates random single-bit errors (e.g., VRT)
- Fully backwards compatible (no changes to DDRx interface)

Unfortunately, has side-effects for error characterization
- Unspecified, black-box implementation
- Obfuscates errors in an ECC-specific manner

On-Die ECC in Literature

• Two types of ECC mentioned
  - (128 + 8) Hamming code
  - (64 + 7) Hamming code

• Paper contains references to both of these
On-Die ECC Research Challenge

**Good** for DRAM manufacturers:
- ✓ Transparently improves reliability
- ✓ Decreases power required for data retention
- ✓ Low latency/power overhead
- ✓ No changes to DRAM interface (i.e., backwards compatible)

**Bad** for researchers studying DRAM errors:
- ✗ Hides errors in a black-box, device-specific way
- ✗ Distorts well-understood statistical distributions
- ✗ Prevents fairly comparing BER of the DRAM itself
EINSim Functional Description

- Simulates the dataflow through a real experiment
  - Configuration parameters replicate experimental setup
  - Simulate enough words to resolve the output distribution

Input Configuration (i.e., $w, S, \theta$)

Output Distribution (i.e., $\overline{w_N}$)

Tester

DRAM

Word Generator

ECC Encoder

Error Injector

ECC Decoder

Error Checker
EINSim Configuration + Features

**Input Configuration** (i.e., $w, S, \theta$)

- Word Generator
- ECC Encoder
- Error Injector

**Output Distribution** (i.e., $\overline{w_N}$)

- Error Checker
- ECC Decoder

<table>
<thead>
<tr>
<th>Module</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word Generator</td>
<td>Word length, Data Pattern</td>
</tr>
<tr>
<td>ECC Encoder, ECC Decoder</td>
<td>ECC code {type, length, strength} code details (e.g., generator polynomial)</td>
</tr>
<tr>
<td>Error Injector</td>
<td>Spatial error distribution</td>
</tr>
<tr>
<td>Error Checker</td>
<td>Measurement (e.g., #errors per word)</td>
</tr>
</tbody>
</table>
• Creates an \(N\)-bit word
  - Commonly used data patterns (e.g., \(0xFF\), RANDOM)
  - Effectively sampling the \(w\) distribution

• \(N\) may be multiple datawords long
  - Useful if we don’t know how datawords are laid out
  - Split into datawords according to a configurable mapping
  - More details about this in the paper
ECC Encoder/Decoder

- EINSim implements ECC algorithms
  - Currently supports common codes (e.g., Hamming, BCH)
  - Modularly designed and easily extensible to others
  - Validated by hand + using unit tests (available on GitHub)

- Configurable parameters for:
  - Number of data bits, correction capability
  - Details of implementation (e.g., generator polynomials)
Error Injector

- Injects errors according to a spatial error distribution
  - Configurable parameters depend on particular distribution
  - Extensible to many different error distributions

- Uniform-random for data-retention errors
  - We experimentally validate this using real LPDDR4 devices
  - Experiment and analysis discussed in detail in the paper
Error Checker

• Computes a configurable output distribution
  - Corresponds to the experimental measurement we make
  - E.g., number of errors per \( \text{dataword} \) (i.e., \( w_N \))
Validating Uniform-Randomness

- We model data-retention errors as **uniform random**
  - Well-studied throughout prior work
  - Error count per N-bit word follows a **binomial distribution**
- We experimentally validate uniform-randomness
  - 82 LPDDR4 devices **without** on-die ECC
  - Disable refresh operations for 20s @ 60°C
Anatomy of a DRAM Bank

• DRAM cells can encode data in two ways:
  - Data ‘1’ as ‘charged’ -> “True-cell”
  - Data ‘1’ as ‘discharged’ -> “Anti-cell”

• Retention errors typically “charged” -> “discharged”
Incidence of “Outlier Rows”

- Some rows do not follow the true-/anti-cell layout
- Appear to follow typical “remapped row” distributions
  - Extra memory rows used for post-manufacturing repair
MAP Estimation Shown Graphically

Non-uniformities
(details in paper)

Low sample count
Example Error-Characterization Studies

• We provide two studies to demonstrate EIN’s value
  - Measure data-retention error rates
  - Have 314 LPDDR4 devices (with + without on-die ECC)

1. BER vs. refresh rates
   - We compare devices with on-die ECC to those without it
   - EIN infers the *pre-correction* BER beneath on-die ECC
   - Enables comparing BER of the DRAM technology itself

2. BER vs. temperature
   - On-die ECC distorts the expected exponential relationship
   - EIN recovers the obfuscated statistical distribution
Finding the “Right” Answer

• MAP estimation selects between suspected models
  - EIN cannot tell if the MAP estimate is “right”
  - “Likelihood” is a relative measure

1. Techniques for gaining confidence in the answer:
   - Using confidence intervals (e.g., statistical bootstrap)
   - Testing across many different error conditions

2. Unlikely that the ECC scheme used is unknown
   - ECC is a well-studied area
   - Manufacturers are unlikely to a completely unknown code

3. Typically we may suspect some schemes already
   - Academic/industry papers, datasheets, etc.
Control of Errors

• EIN requires *knowledge* and *control* of errors
  1. Understand the spatial distribution of errors
  2. Be able to induce uncorrectable errors

• Not a limitation in practice for DRAM
  - Many well-studied easily-controlled error mechanisms exist
    • E.g., data retention
    • E.g., access-latency reduction (i.e., tRCD, tRP, etc.)
    • E.g., RowHammer
Error Localization

• EIN cannot identify *bit-exact* error locations
  - ECC decoding function is *lossy* (i.e., many-to-one)
  - We are unaware of a way to reverse the decoding function

• Not a limitation in practice since we can still infer:
  - The ECC scheme
  - Pre-correction error rates