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

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*
  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

Presentation Outline

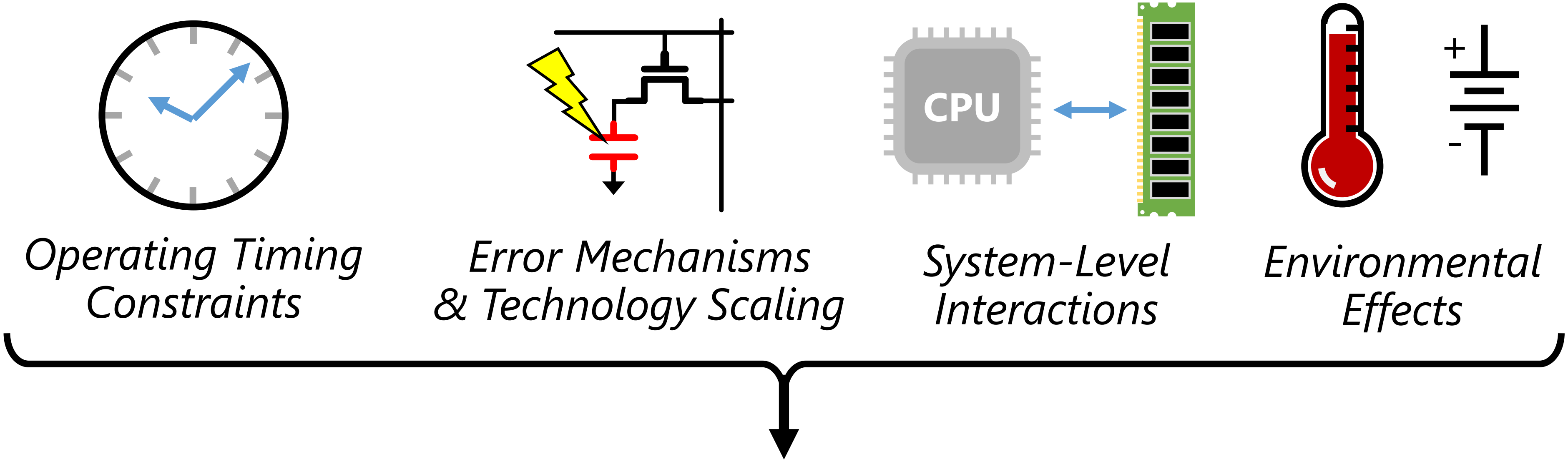
1. Error Characterization and On-Die ECC

2. EIN: Error INference
   1. The Inference Problem
   2. Formalization
   3. 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

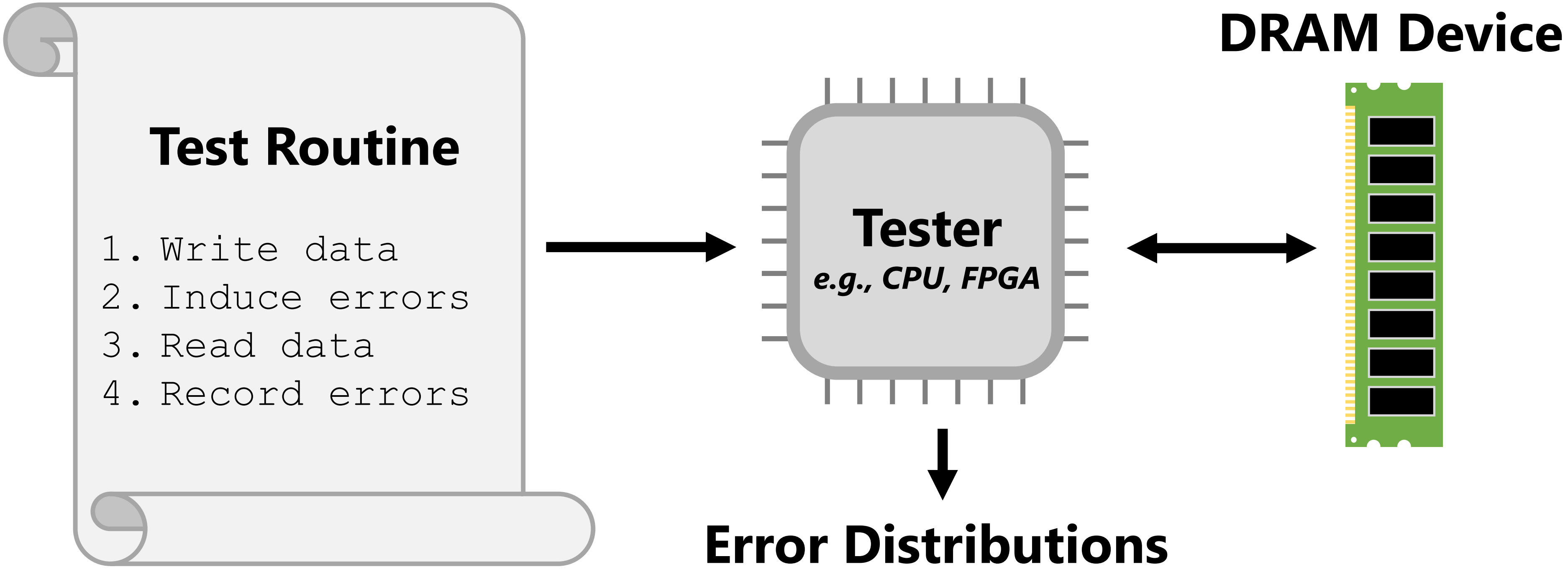
Environmental Effects

Understanding

+

Exploitable Insights

How Do We Characterize DRAM?



Test Routine

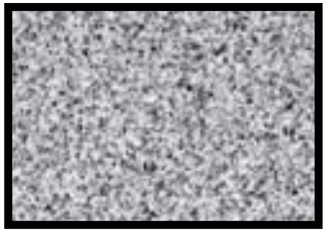
1. Write data
2. Induce errors
3. Read data
4. Record errors

Tester

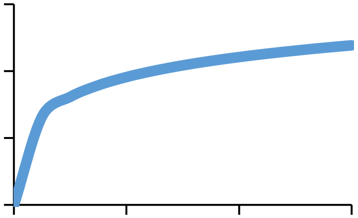
e.g., CPU, FPGA

DRAM Device

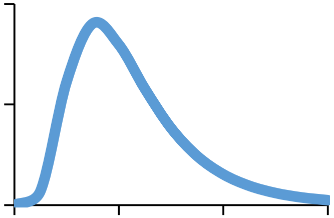
Error Distributions



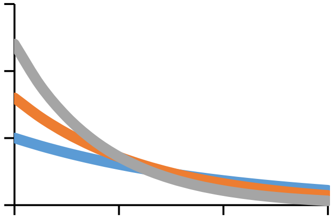
Spatial Distributions



Temporal Distributions



Cell-to-cell Variation

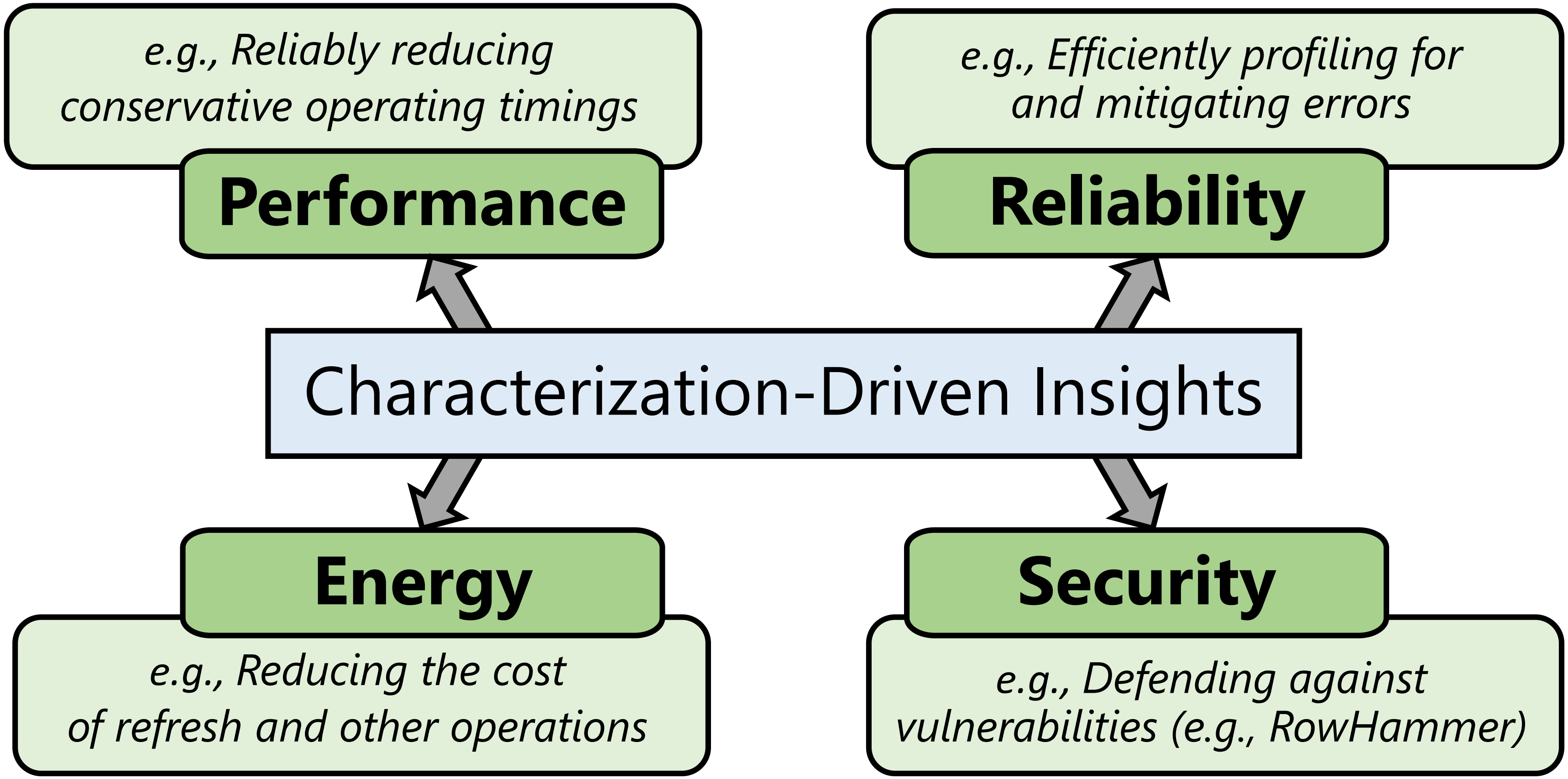


Device Comparisons

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



e.g., Reliably reducing conservative operating timings

Performance

e.g., Efficiently profiling for and mitigating errors

Reliability

Characterization-Driven Insights

Energy

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

Security

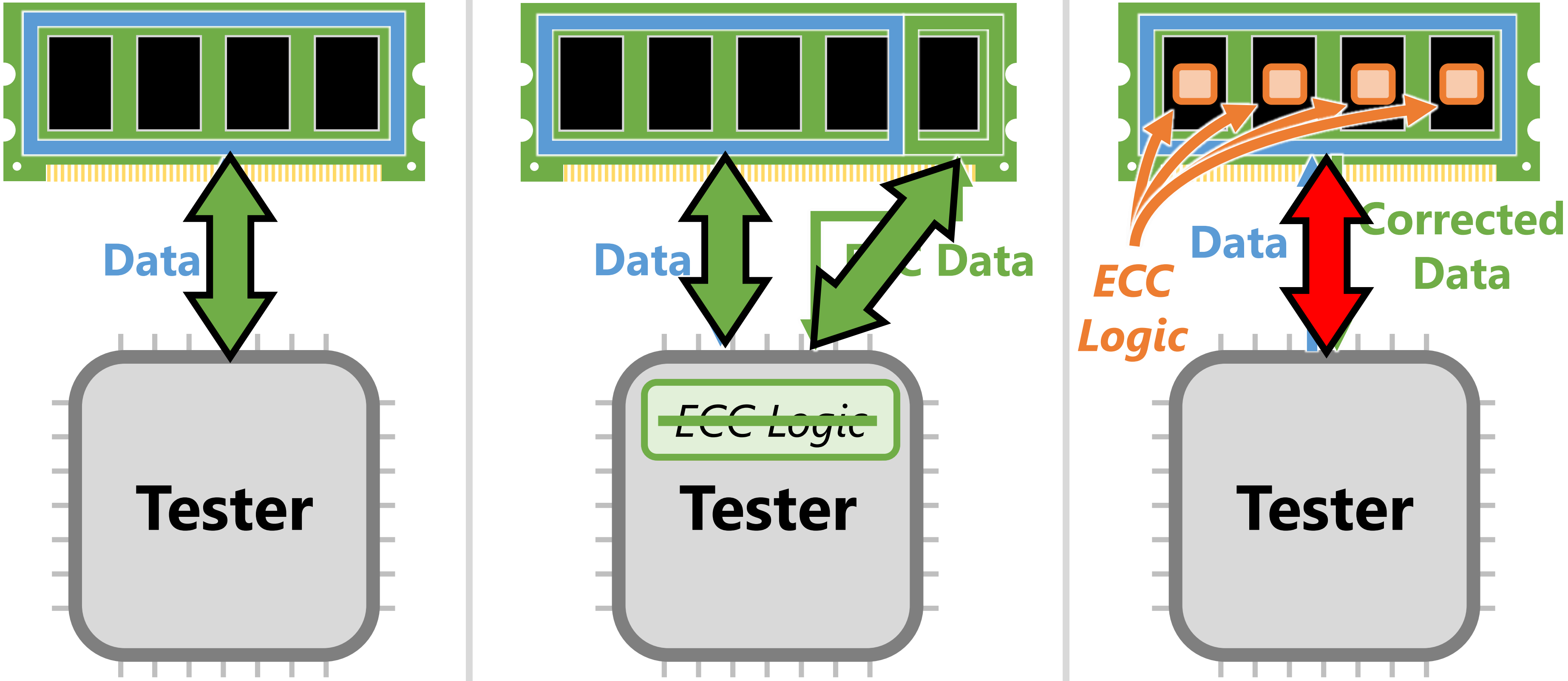
e.g., Defending against vulnerabilities (e.g., RowHammer)

Three Key Types of DRAM

No ECC (Standard)

Rank-Level ECC (Server-style)

On-Die ECC (or Integrated ECC)



✓ Raw Data Is Unmodified

✓ Raw Data Is Unmodified

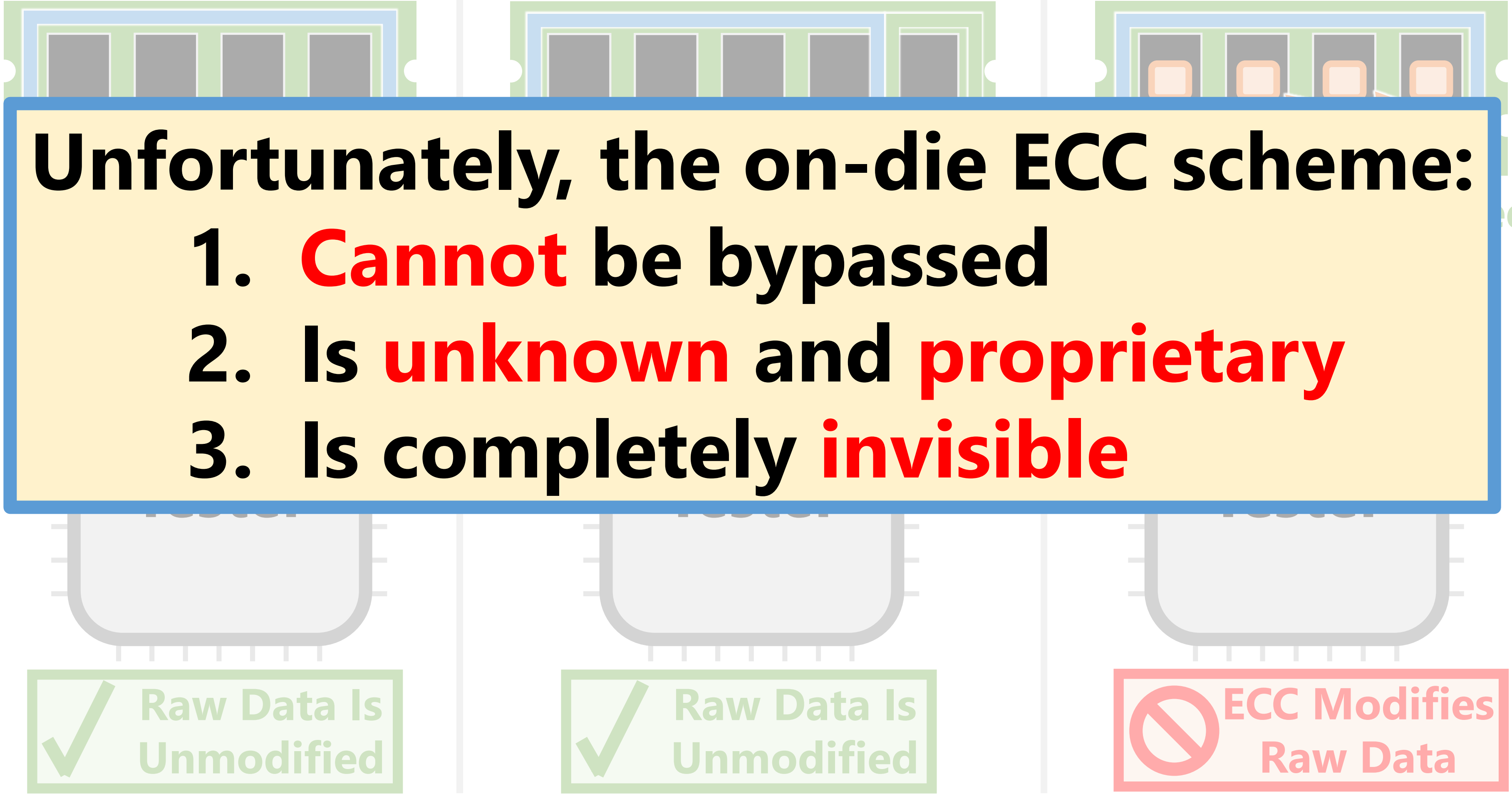
🛇 ECC Modifies Raw Data

Three Key Types of DRAM

No ECC
(Standard)

Rank-Level ECC
(Server-style)

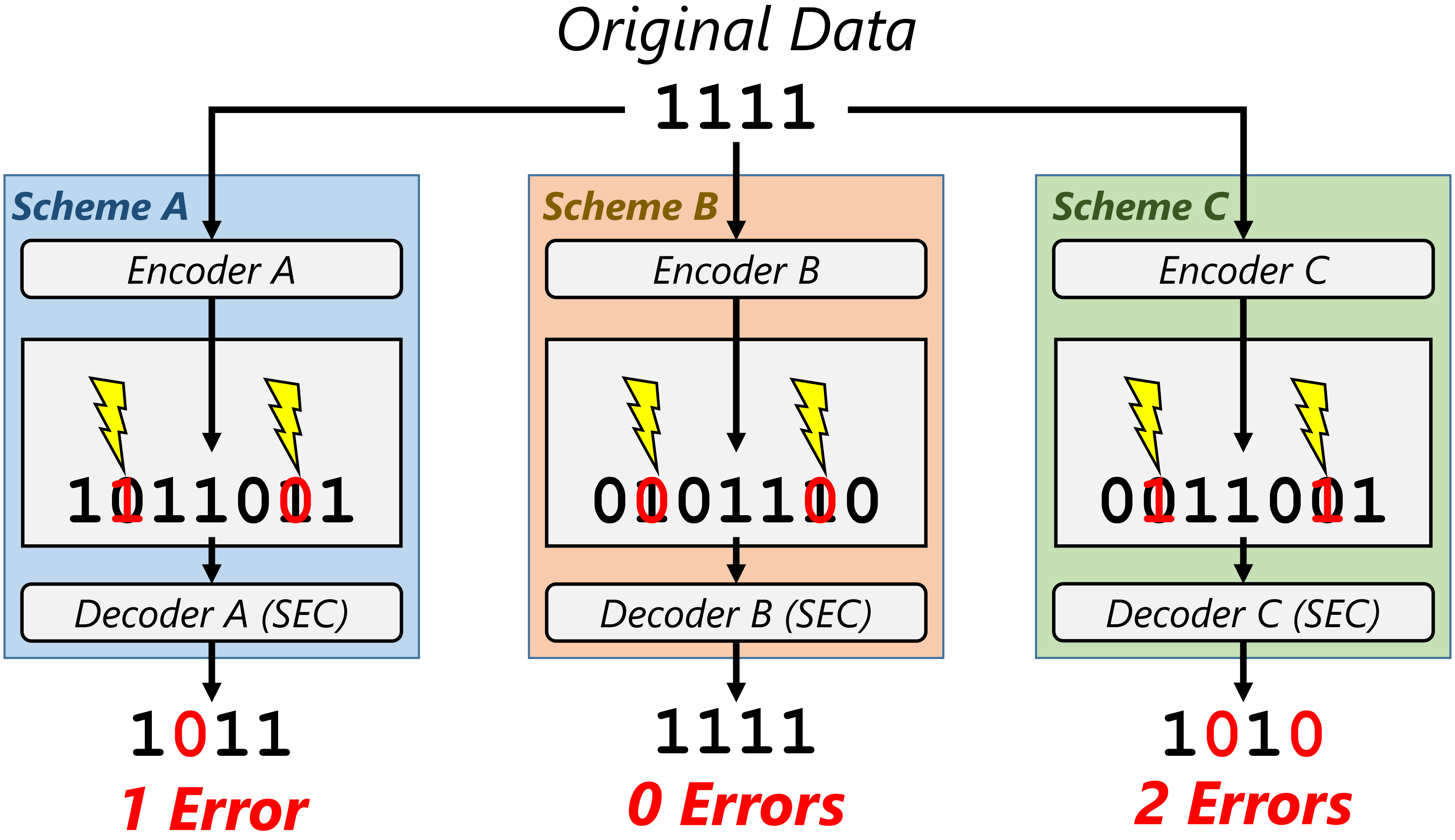
On-Die ECC
(or Integrated ECC)



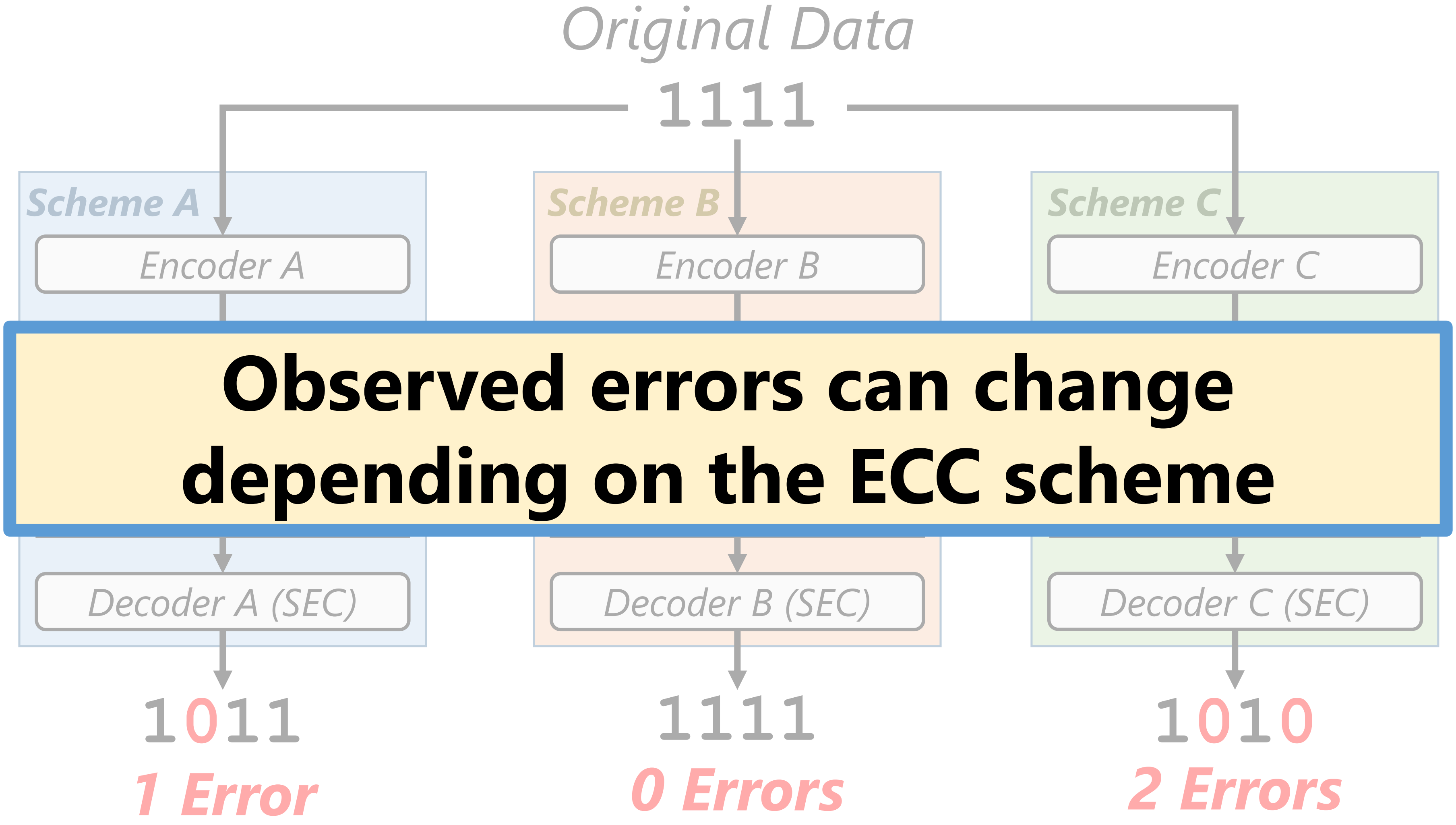
Unfortunately, the on-die ECC scheme:

1. **Cannot be bypassed**
2. **Is unknown and proprietary**
3. **Is completely invisible**

ECC Complicates Error Characterization

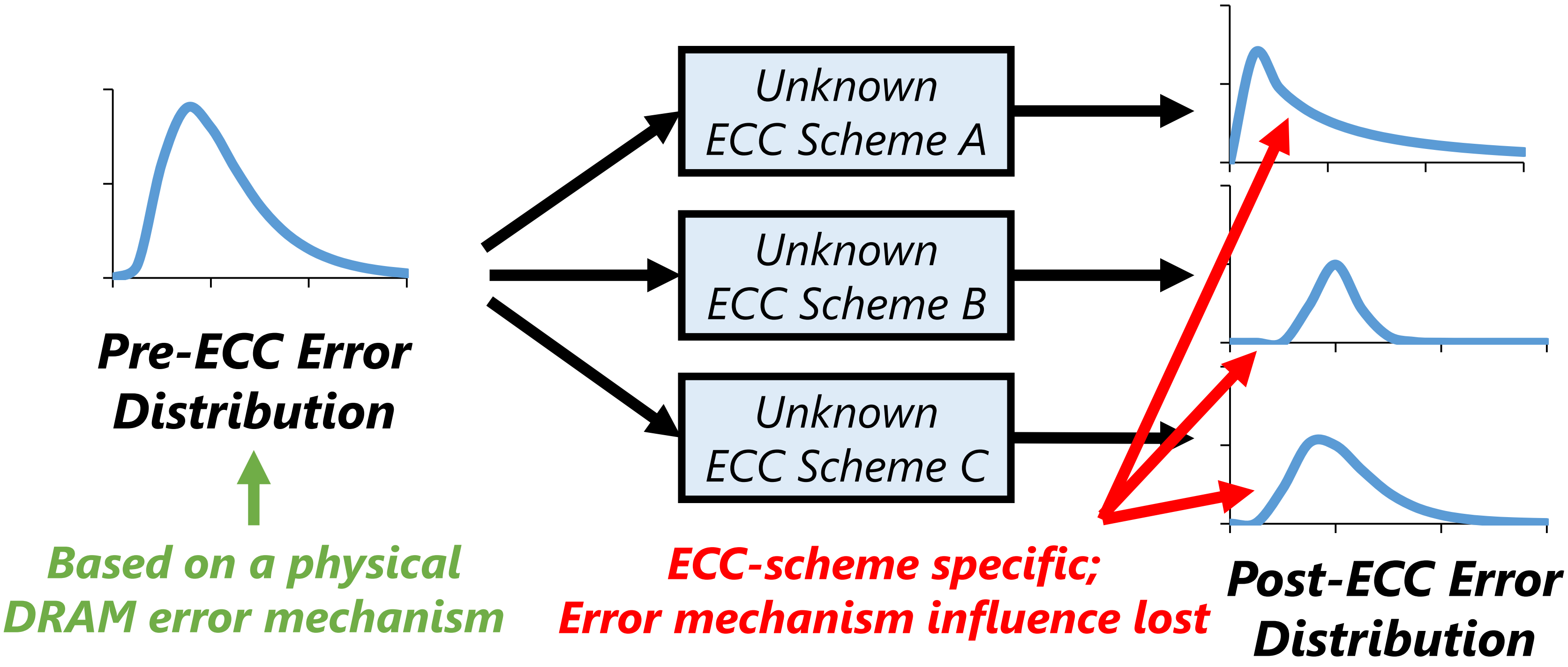


ECC Complicates Error Characterization



Observed errors can change depending on the ECC scheme

ECC Makes Error Characterization Difficult


Pre-ECC Error Distribution
Based on a physical DRAM error mechanism
Unknown ECC Scheme A
Unknown ECC Scheme B
Unknown ECC Scheme C
ECC-scheme specific; Error mechanism influence lost
Post-ECC Error Distribution

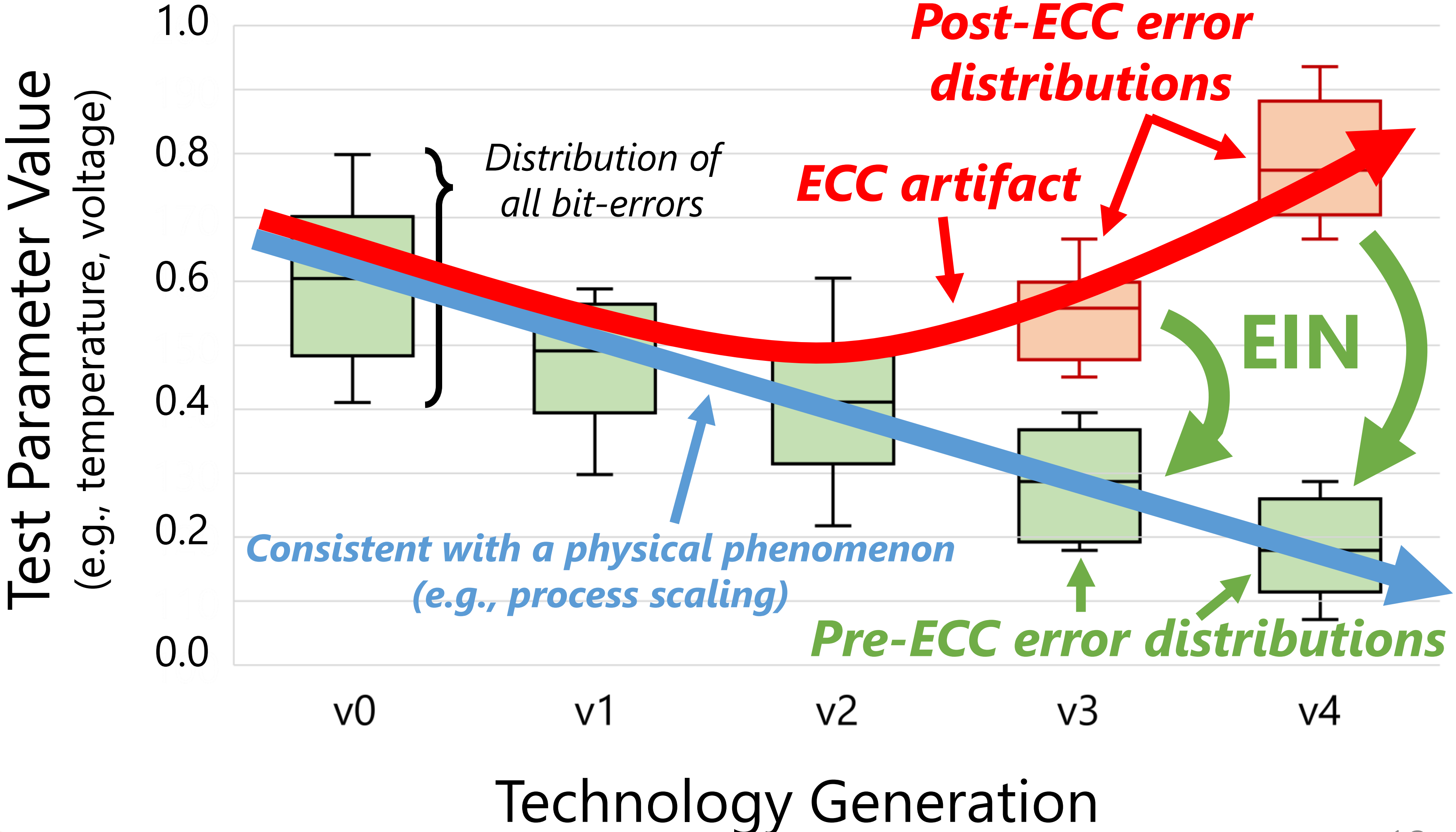
- ECC causes two key **problems**:

Prevents comparing error characteristics between devices

Obfuscates the well-studied error distributions we expect

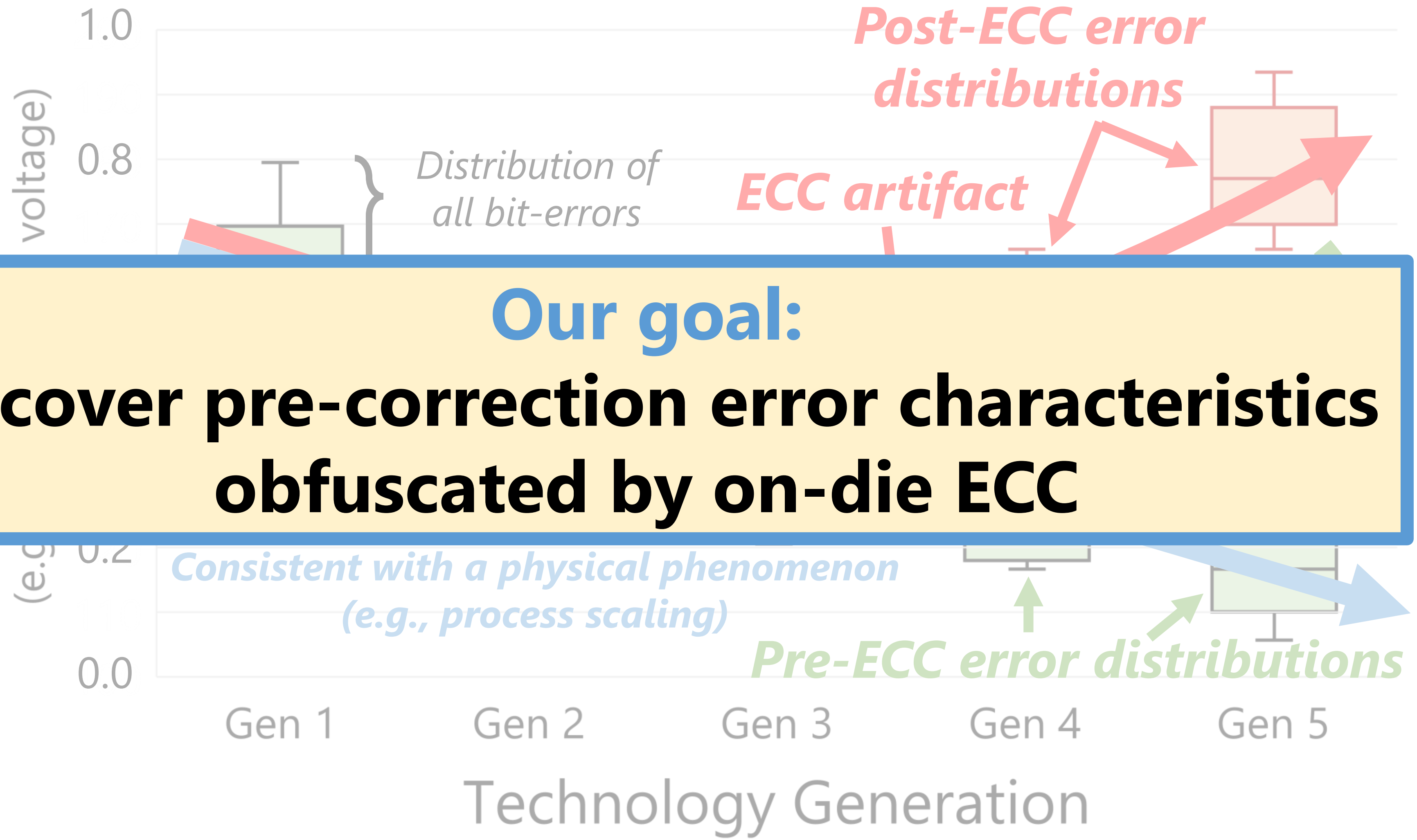
Example: Technology Scaling Study

- Goal: study how errors evolve over technology generations



Example: Technology Scaling Study

- Goal: study how errors evolve over technology generations



Our goal:

Recover pre-correction error characteristics obfuscated by on-die ECC

Presentation Outline

1. Error Characterization and On-Die ECC

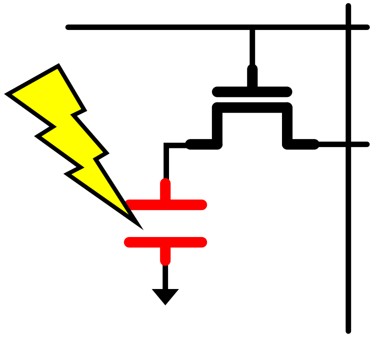
2. **EIN: Error INference**
   I. **The Inference Problem**
   II. Formalization
   III. EIN in Practice: EINSim

3. Demonstration Using LPDDR4 Devices

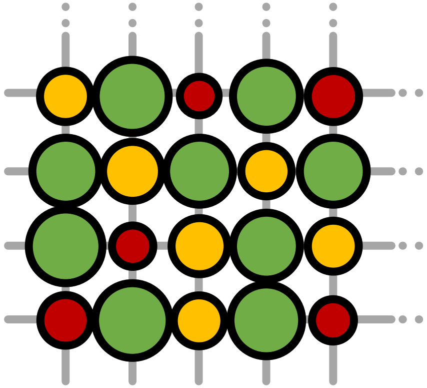
Key Observation

DRAM error mechanisms have
predictable characteristics
*that are **intrinsic** to DRAM technology*

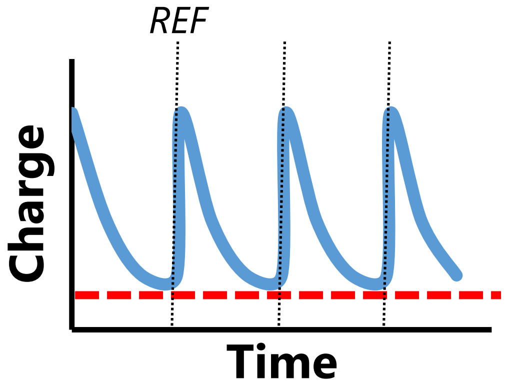
Example: Data-Retention Errors



DRAM encodes data in **leaky capacitors**



Leakage rates differ due to **process variation**

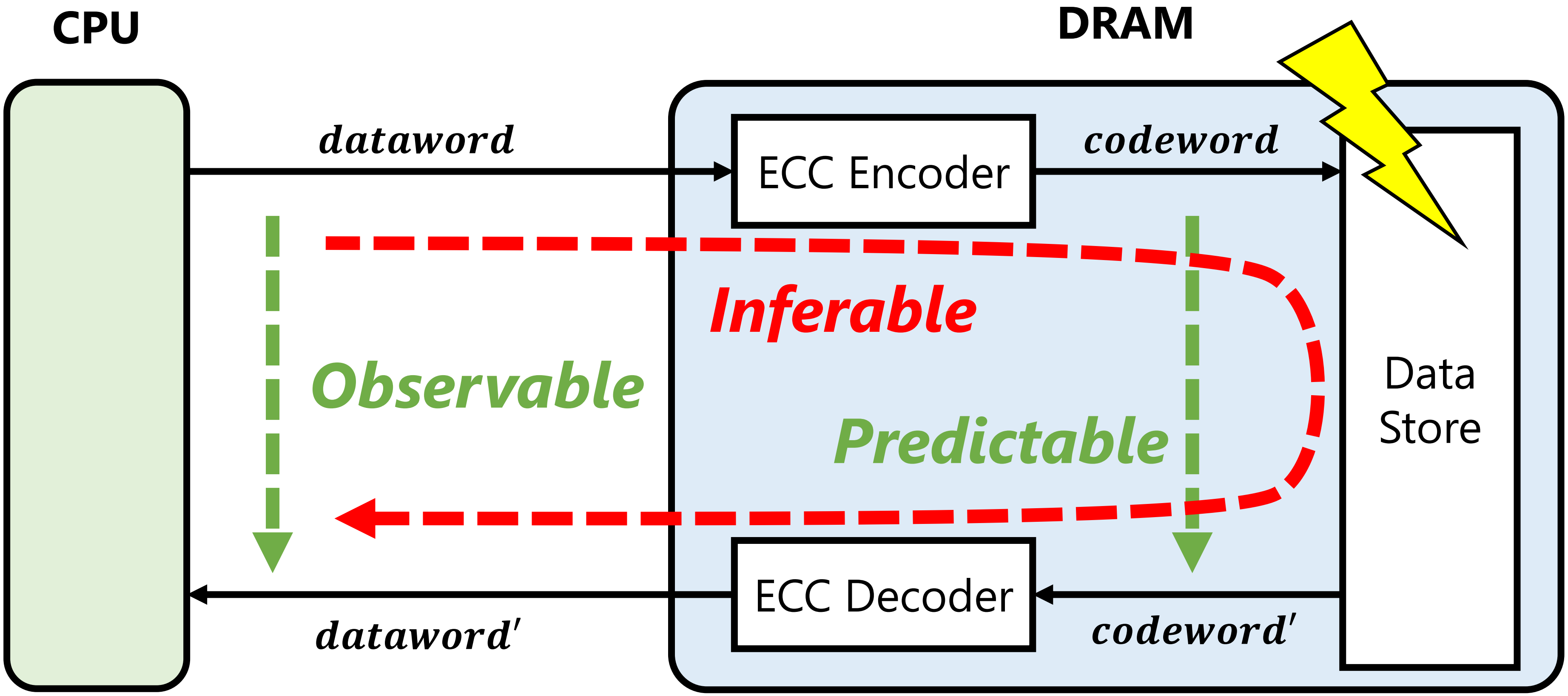


Necessitates periodic **refresh operations**

- 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

Inferring the ECC Scheme

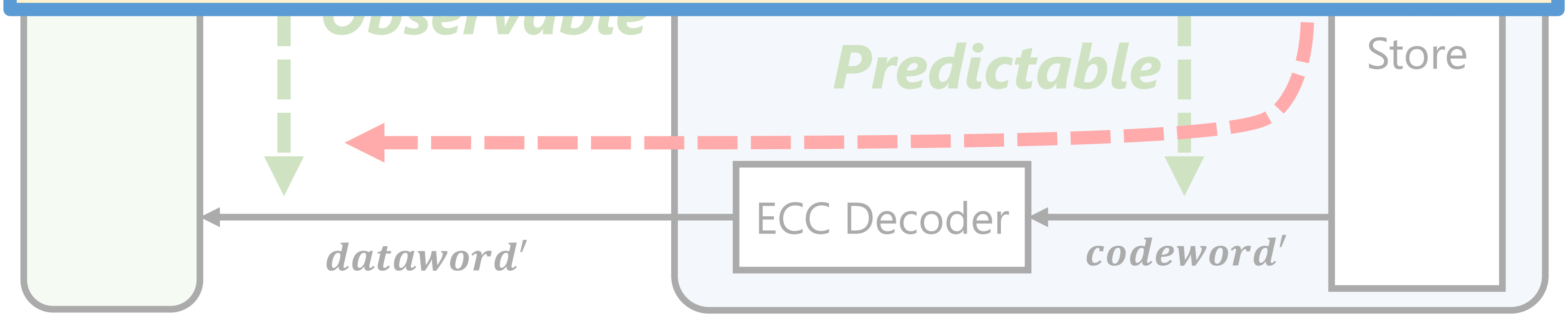
- Exploit error characteristics to **infer** the ECC scheme
  - Works for any DRAM susceptible to the error mechanism
  - Independent of any particular device or manufacturer



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



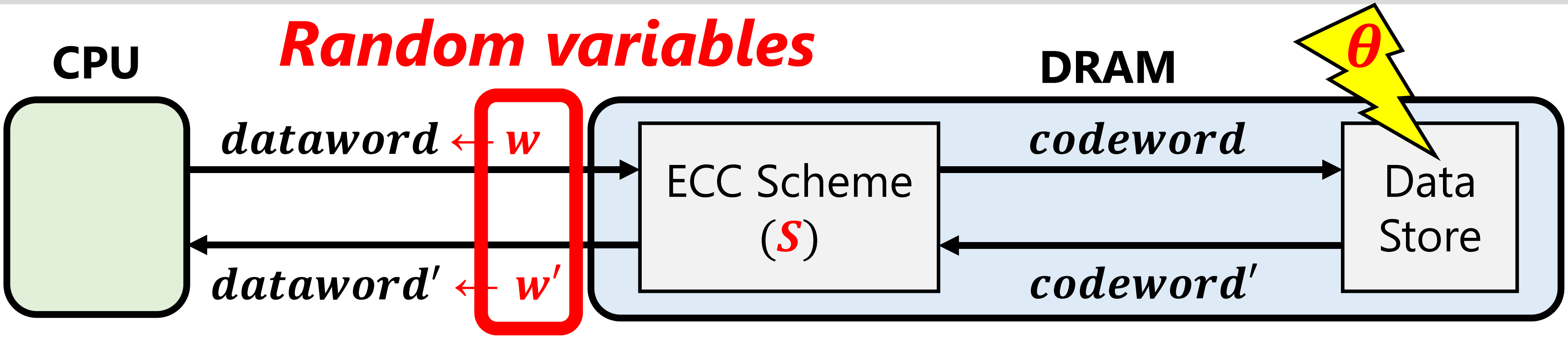
Presentation Outline

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**:

Distribution of outputs

Distribution of inputs

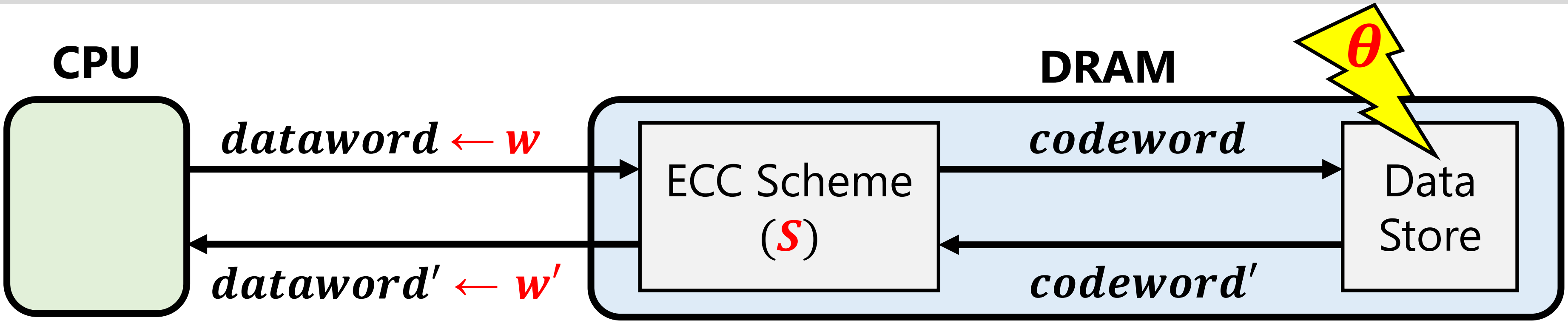
ECC Scheme

Error Distribution

$$w' = f(w \mid S, \theta)$$

We want to **infer** $\{S, \theta\}$ given **observed** $\{w, w'\}$

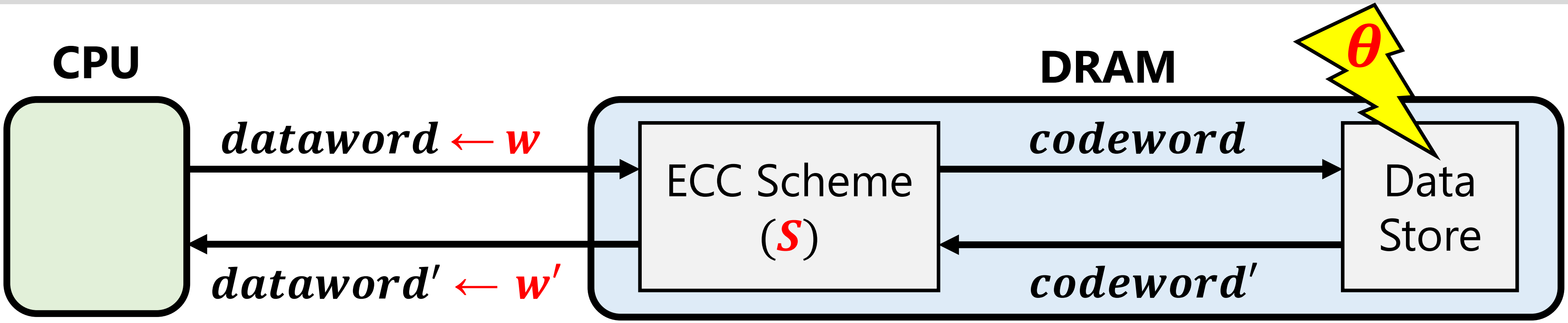
Formalizing the Inference Problem



$$w' = f(w \mid S, \theta)$$

- S: ECC encoding/decoding algorithms
- θ: 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

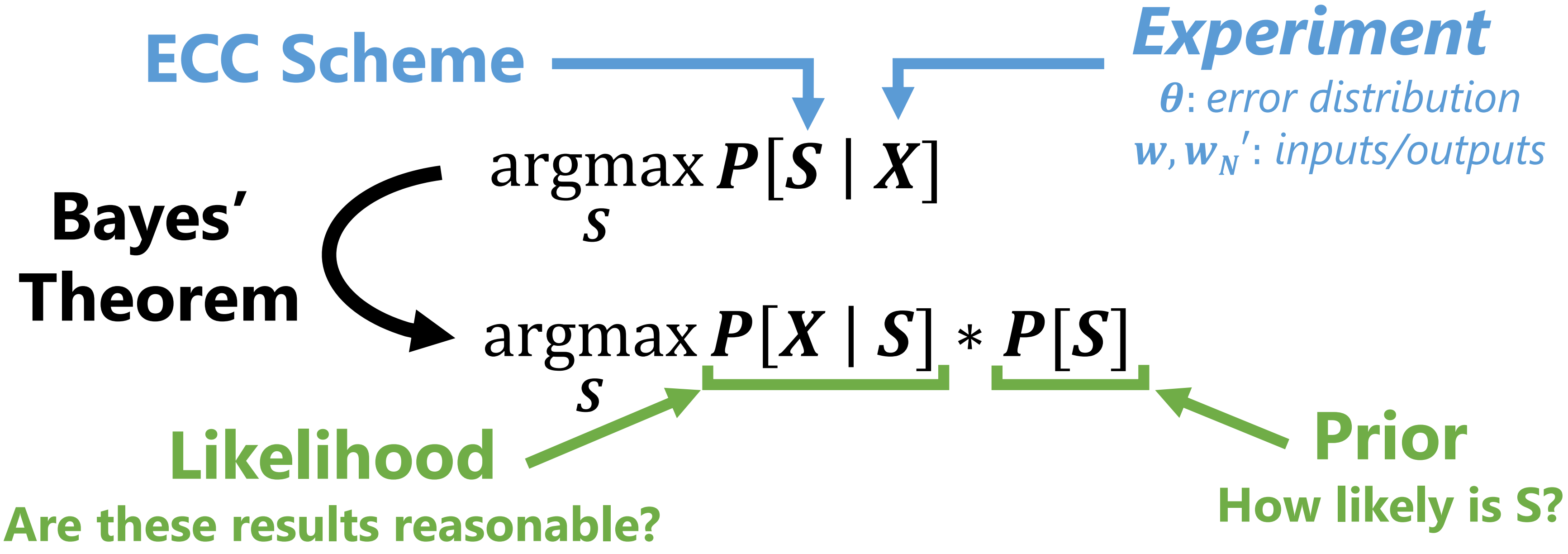


$$w' = f(w \mid S, \theta)$$

- **Unfortunately:** w' is hard to measure
  - 64-bit $dataword$ → 2^{64} possible values
  - Typical 8GiB DRAM only has ~2^{30} $datawords$ (<< 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



ECC Scheme

Experiment

$\boldsymbol{\theta}$: error distribution

$\boldsymbol{w}, \boldsymbol{w_N}'$: inputs/outputs

$$\operatorname*{argmax}_{\boldsymbol{S}} \boldsymbol{P}[\boldsymbol{S} \mid \boldsymbol{X}]$$

Bayes' Theorem

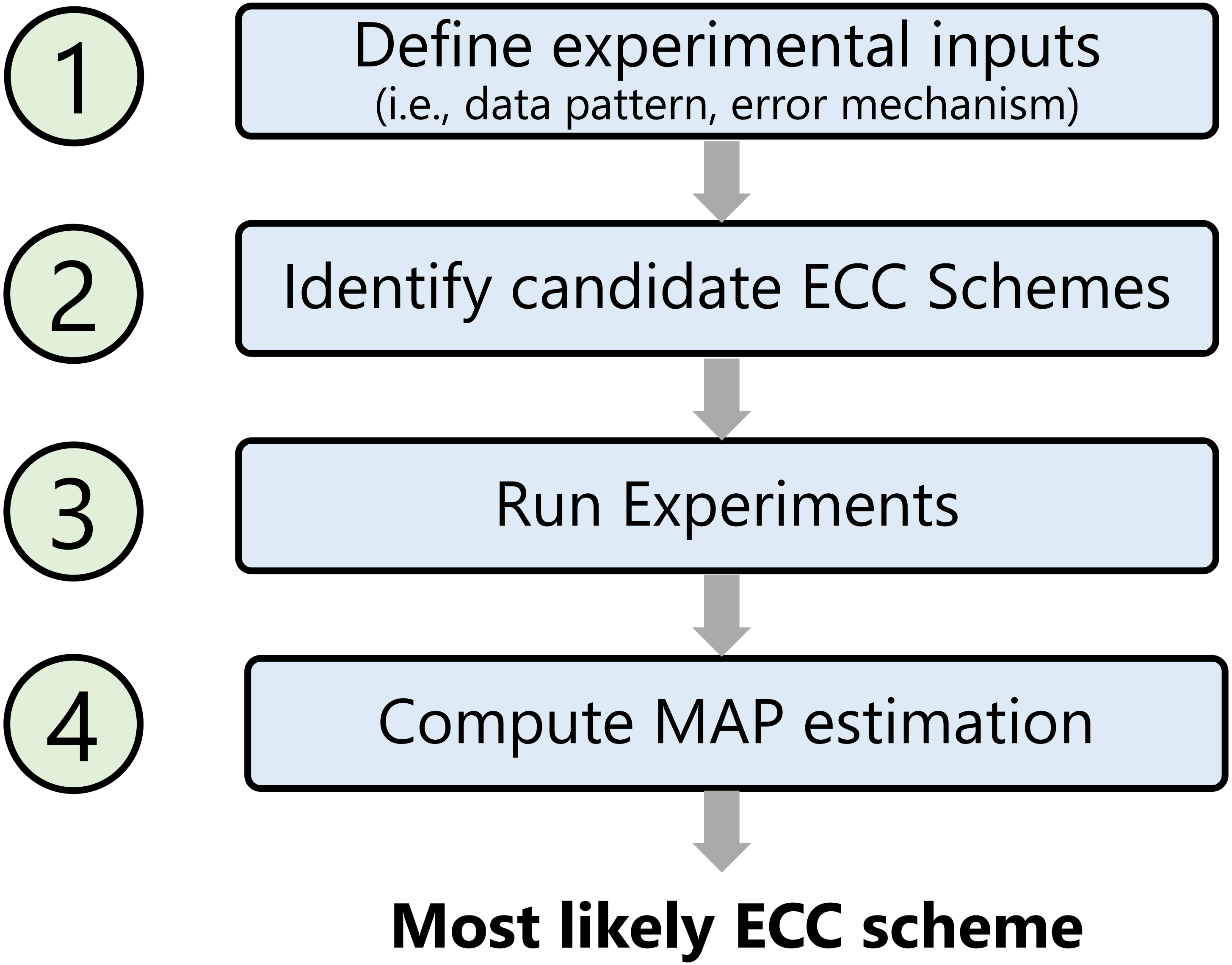
$$\operatorname*{argmax}_{\boldsymbol{S}} \underbrace{\boldsymbol{P}[\boldsymbol{X} \mid \boldsymbol{S}]}_{\text{Likelihood}} * \underbrace{\boldsymbol{P}[\boldsymbol{S}]}_{\text{Prior}}$$

Are these results reasonable?

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 $\boldsymbol{\theta}$

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

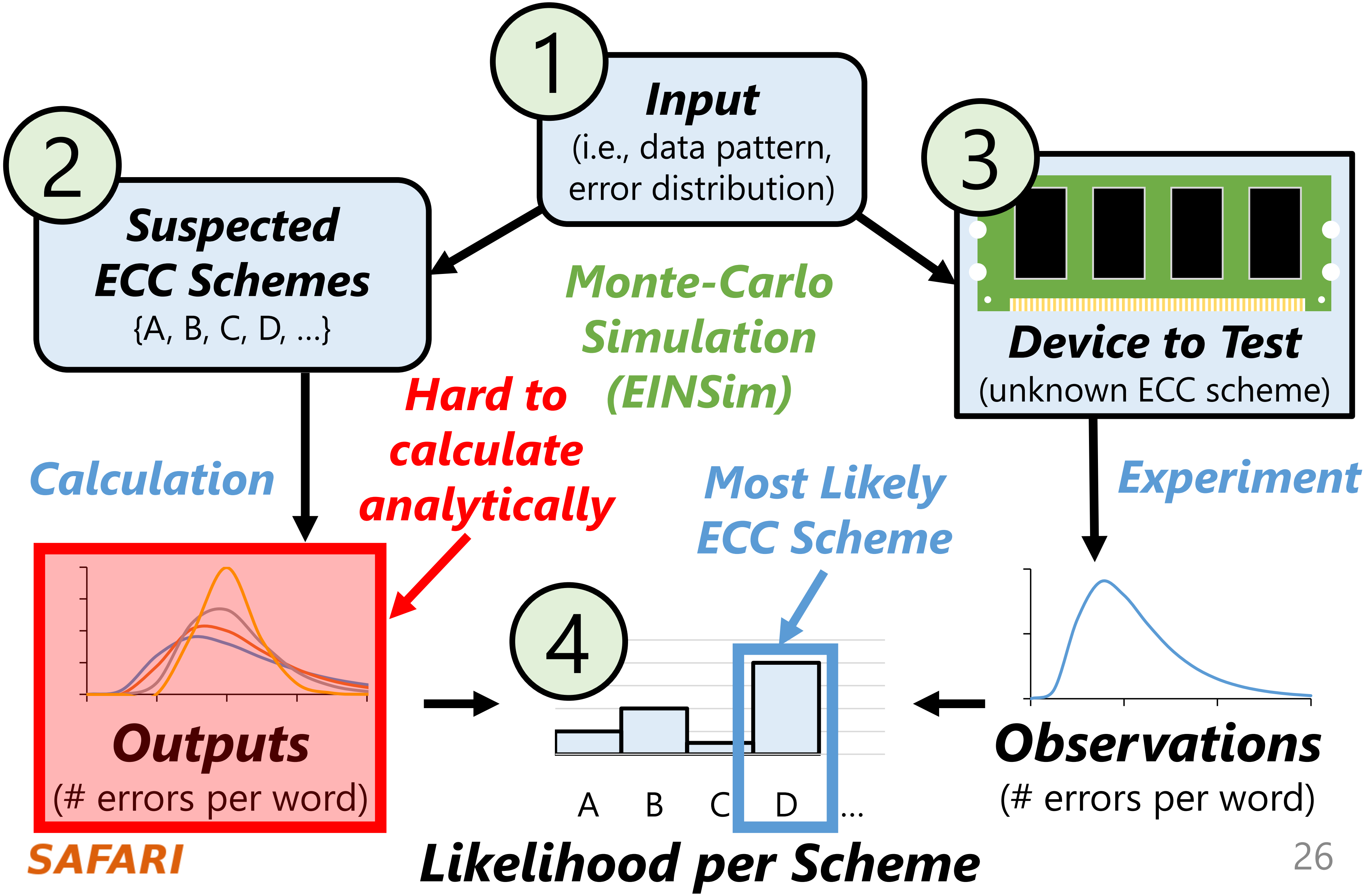
Presentation Outline

1. Error Characterization and On-Die ECC

2. **EIN: Error INference**
    1. The Inference Problem
    2. Formalization
    3. **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, …}

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

Monte-Carlo Simulation (EINSim)

Calculation

Hard to calculate analytically

Experiment

Outputs
(# errors per word)

4

Most Likely ECC Scheme

Observations
(# errors per word)

Likelihood per Scheme

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

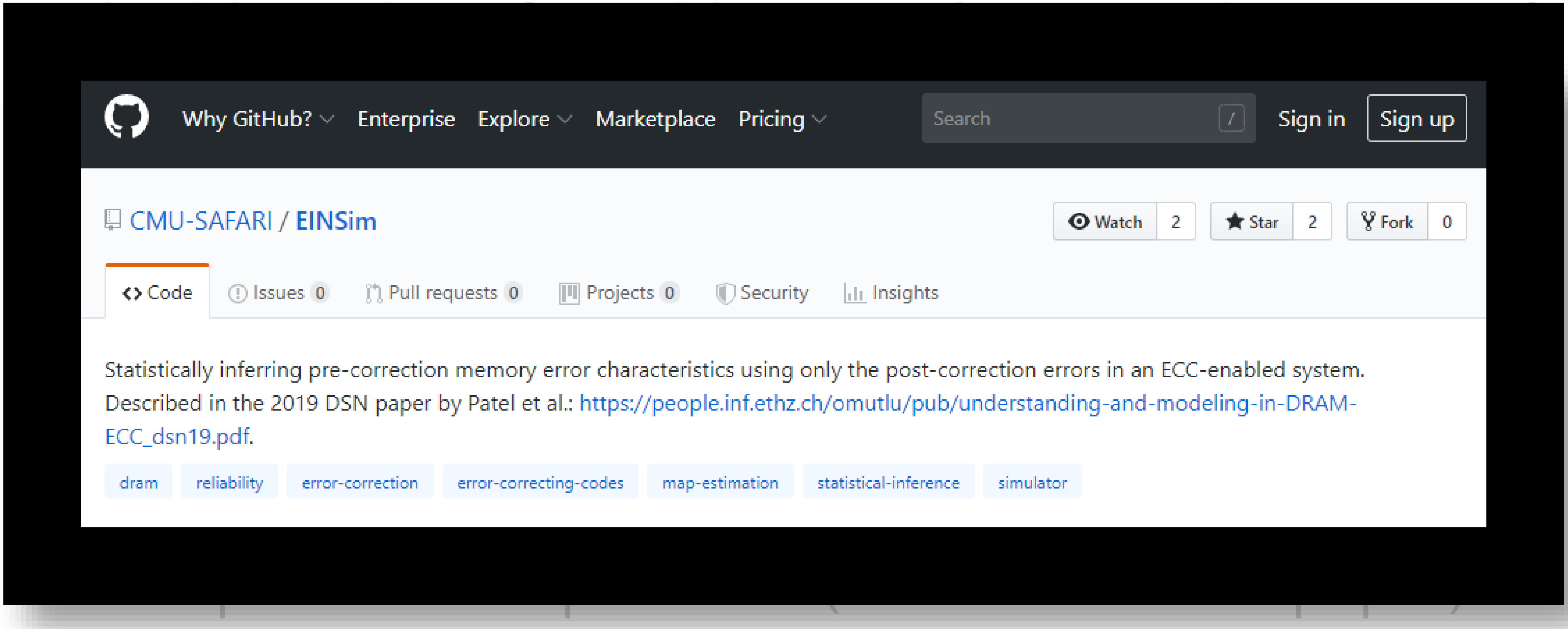
https://github.com/CMU-SAFARI/EINSim

- Example datasets provided (same as used in paper)

EINSim: A Tool for Using EIN

- Evaluates MAP estimation via Monte-Carlo simulation

Give EINSim a try at:
https://github.com/CMU-SAFARI/EINSim



Statistically inferring pre-correction memory error characteristics using only the post-correction errors in an ECC-enabled system. Described in the 2019 DSN paper by Patel et al.: https://people.inf.ethz.ch/omutlu/pub/understanding-and-modeling-in-DRAM-ECC_dsn19.pdf.

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

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

Parameter	***Experiment***	***Simulation (EINSim)***
Word Size	256 bits	256 bits
ECC Schemes	**Unknown**	Hamming (32, 64, 128, 256) BCH-2EC (32, 64, 128, 256) BCH-3EC (32, 64, 128, 256) Repetition (3, 5, 7)
Data Pattern	RANDOM	RANDOM, 0xFF
Error Mechanism	Data-Retention	Data-Retention

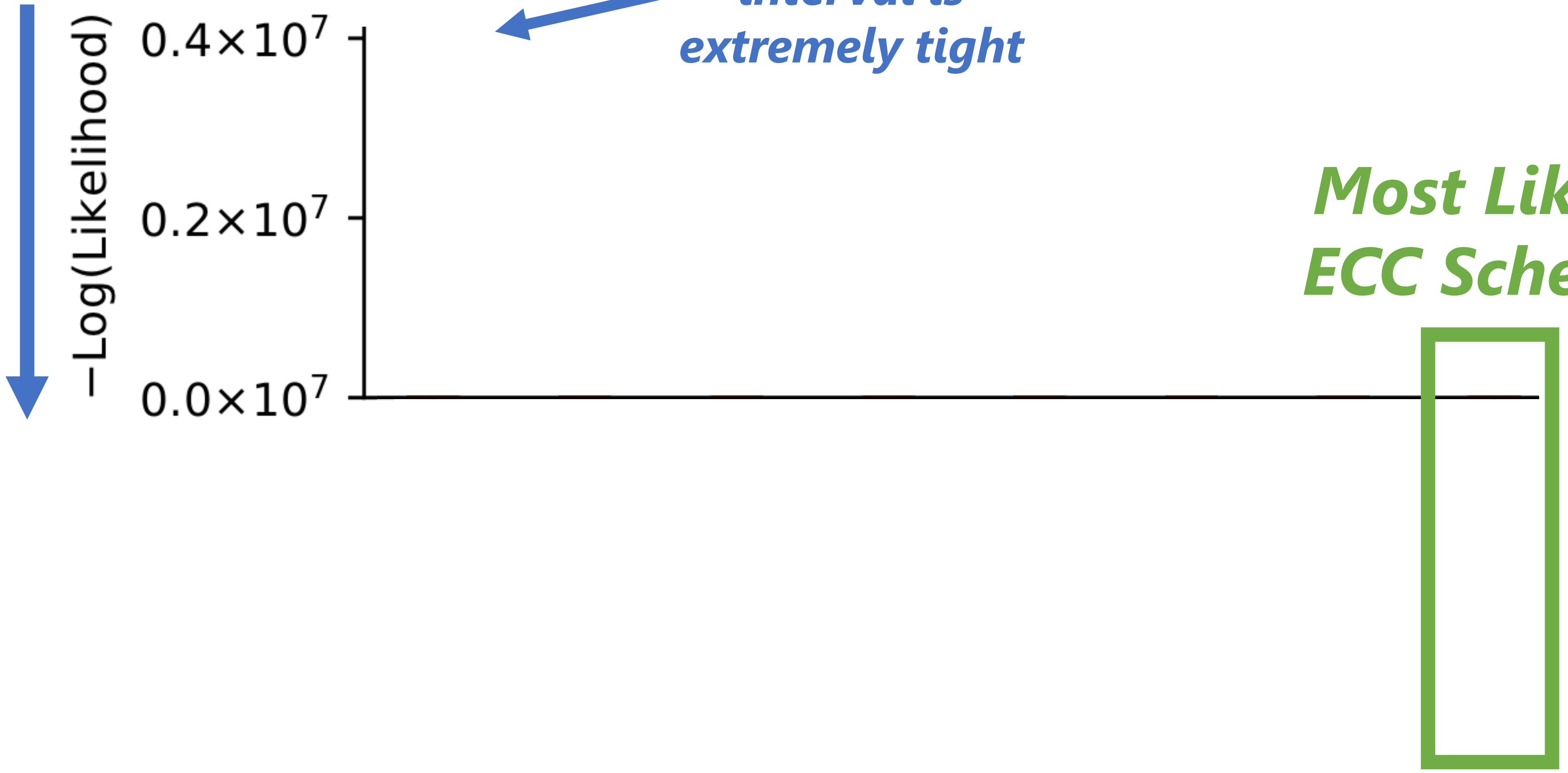
MAP Estimation Methodology

- 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



Lower is MORE Likely

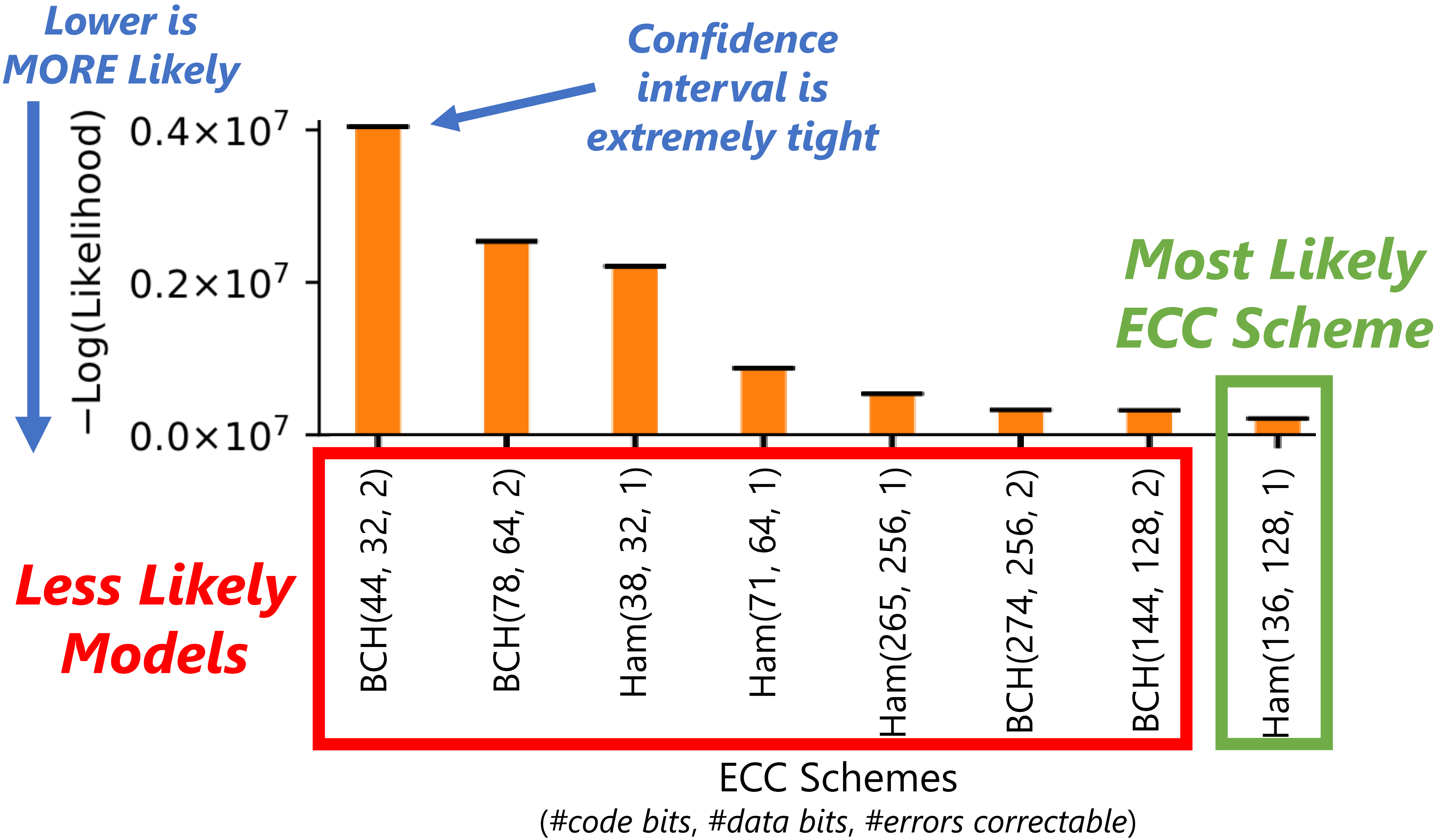
Confidence interval is extremely tight

Most Likely ECC Scheme

ECC Schemes

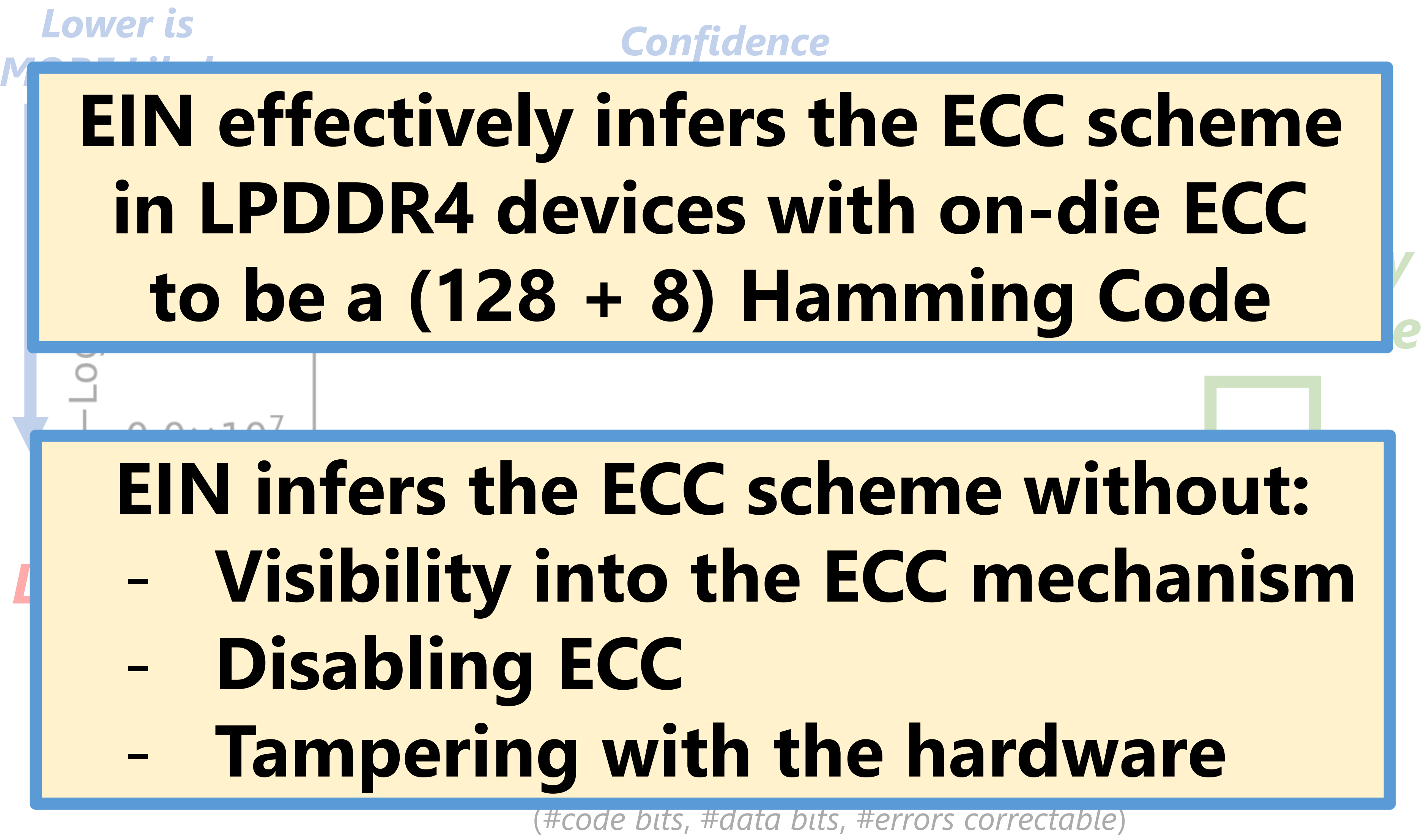
(#code bits, #data bits, #errors correctable)

MAP Estimation Results



SAFARI

MAP Estimation Results



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

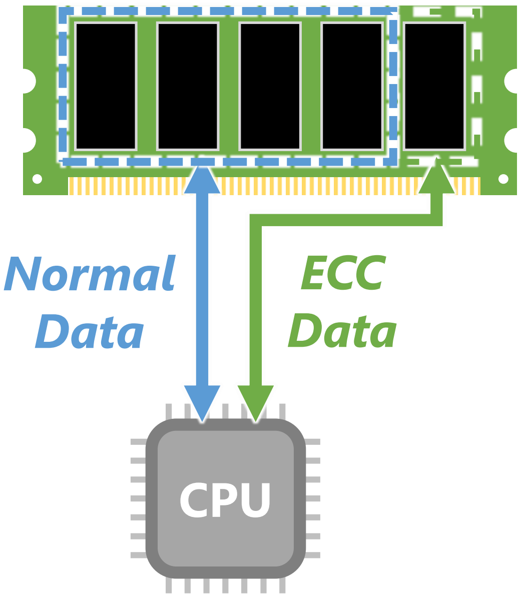
EIN infers the ECC scheme without:

- **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

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*
  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

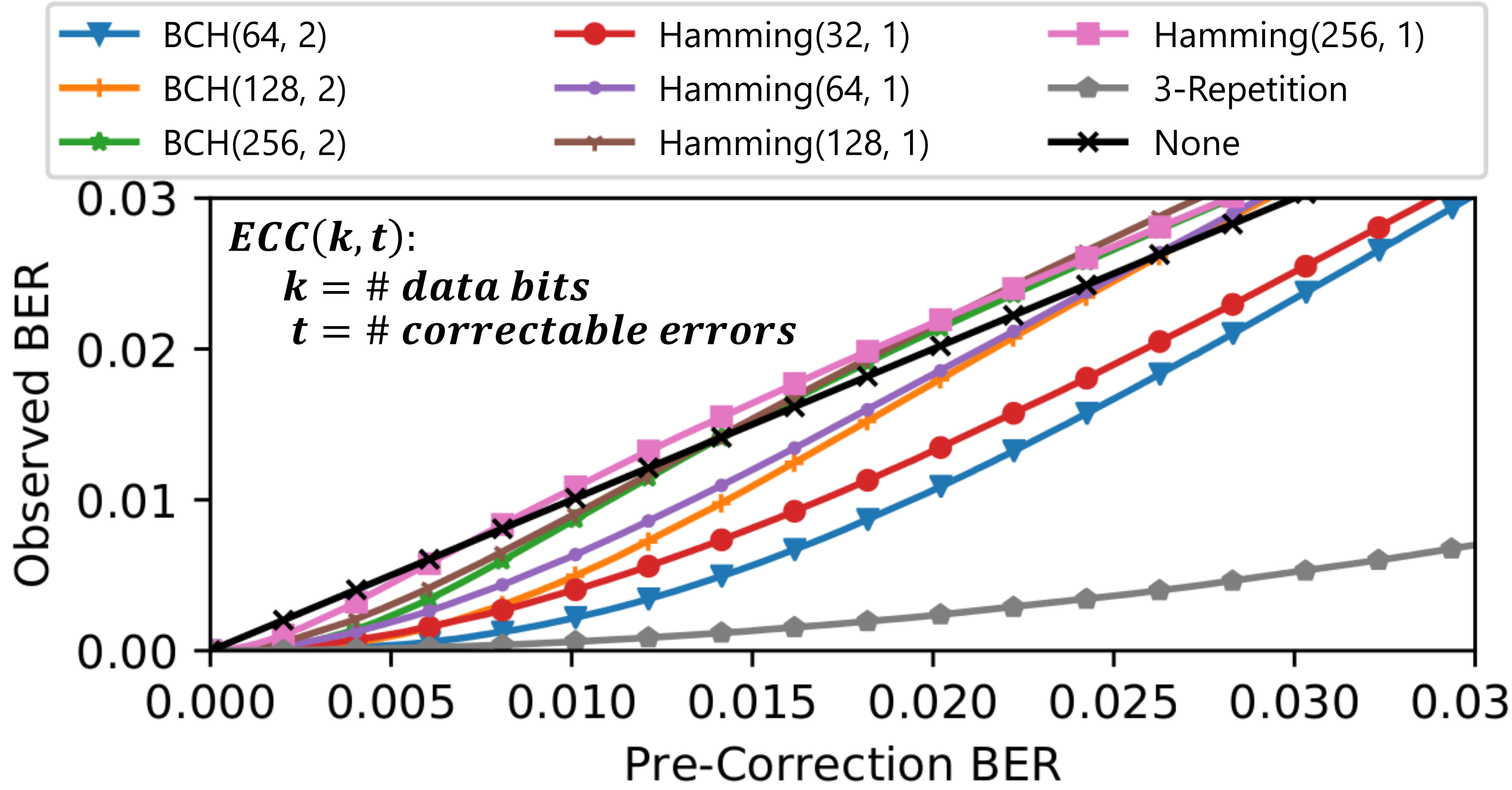
2. Comparison studies (e.g., technology scaling)
   - Use EIN to compare *pre-correction* error rates
   - Study + predict industry and future technology trends

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 $\boldsymbol{P[error]}$ = "bit error rate" (BER)



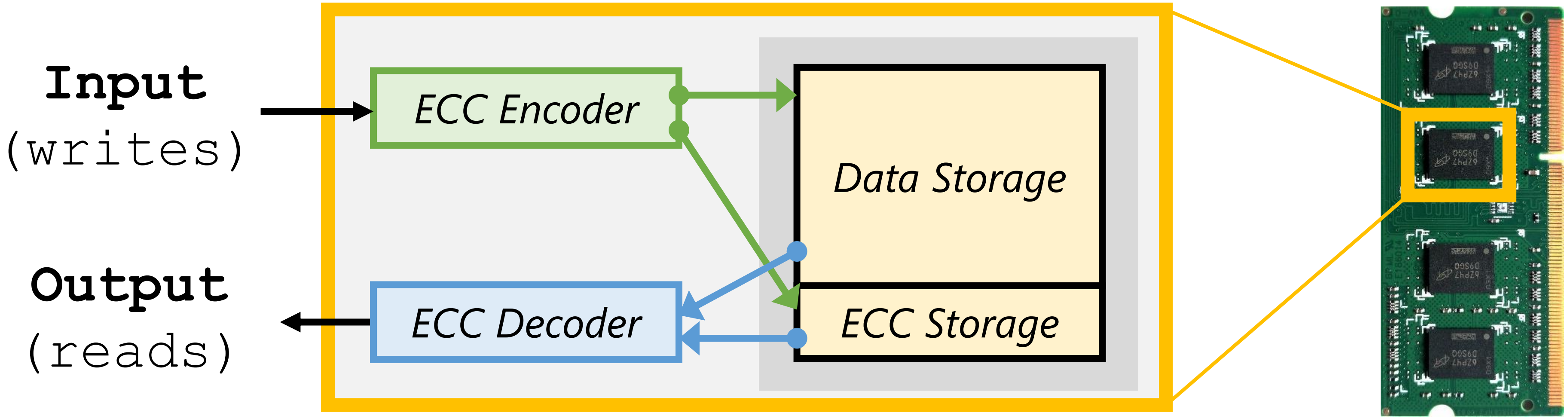
$\boldsymbol{ECC(k,t)}$:

$\boldsymbol{k = \#\ data\ bits}$

$\boldsymbol{t = \#\ correctable\ errors}$

A Closer Look at On-Die ECC

DRAM with 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

[1] *"ECC Brings Reliability and Power Efficiency to Mobile Devices,"* Micron Technology, Inc, Whitepaper, 2017

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:

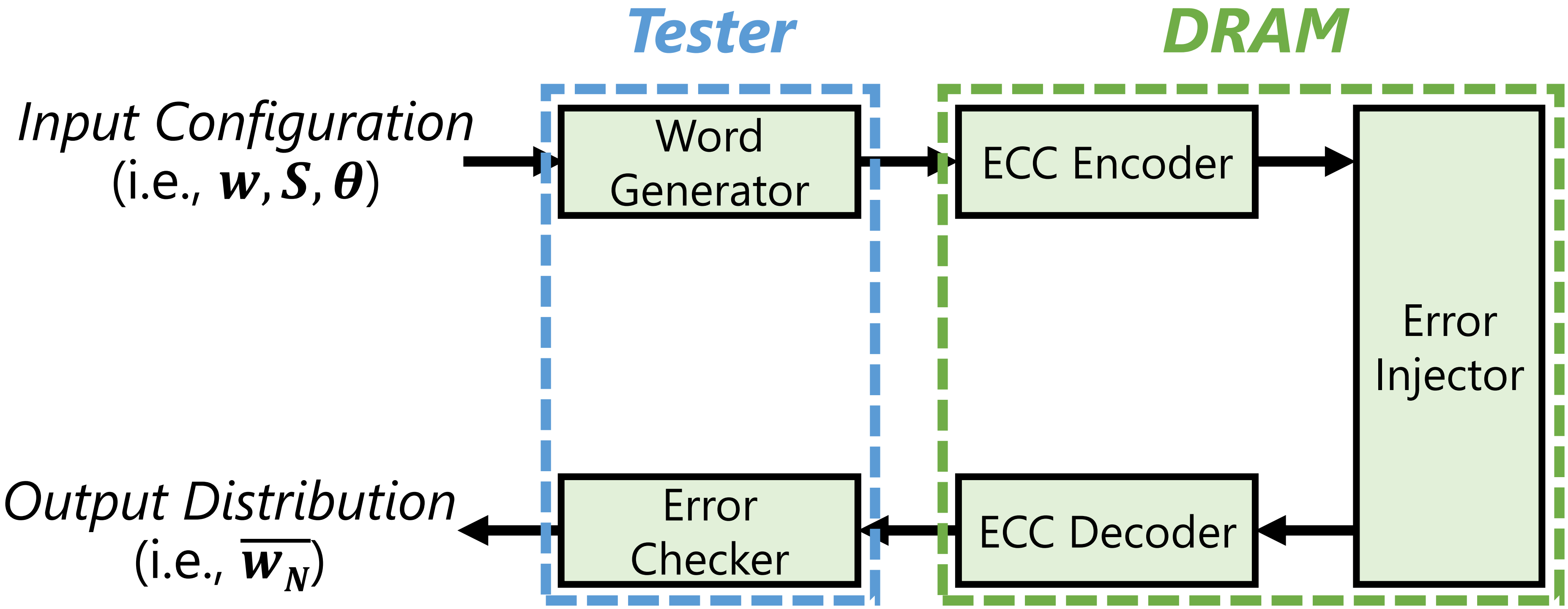
- ✓ Transparently improves reliability
- ✓ Decreases power required for data retention
- ✓ Low latency/power overhead
- ✓ No changes to DRAM interface (i.e., backwards compatible)

Bad for researchers studying DRAM errors:

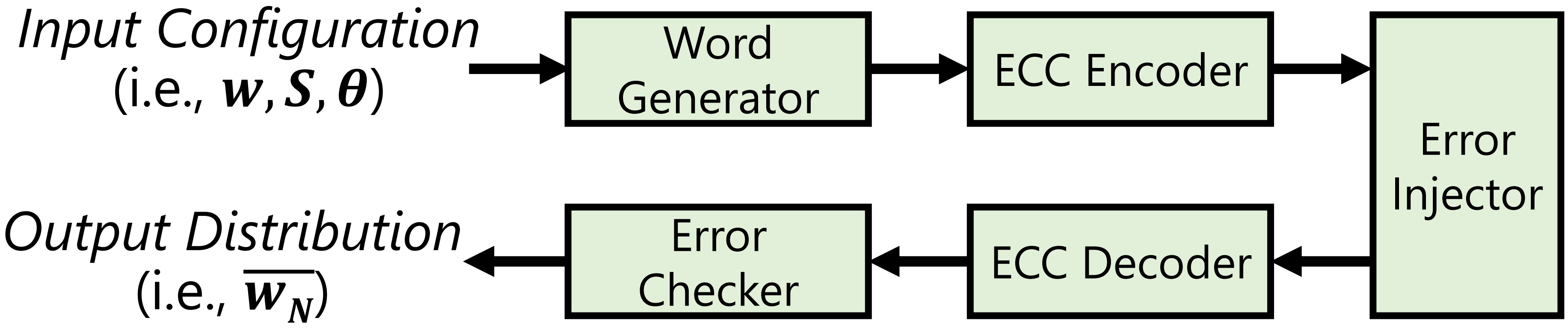
- ✗ Hides errors in a black-box, device-specific way
- ✗ Distorts well-understood statistical distributions
- ✗ Prevents fairly comparing BER of the DRAM itself

EINSim Functional Description

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

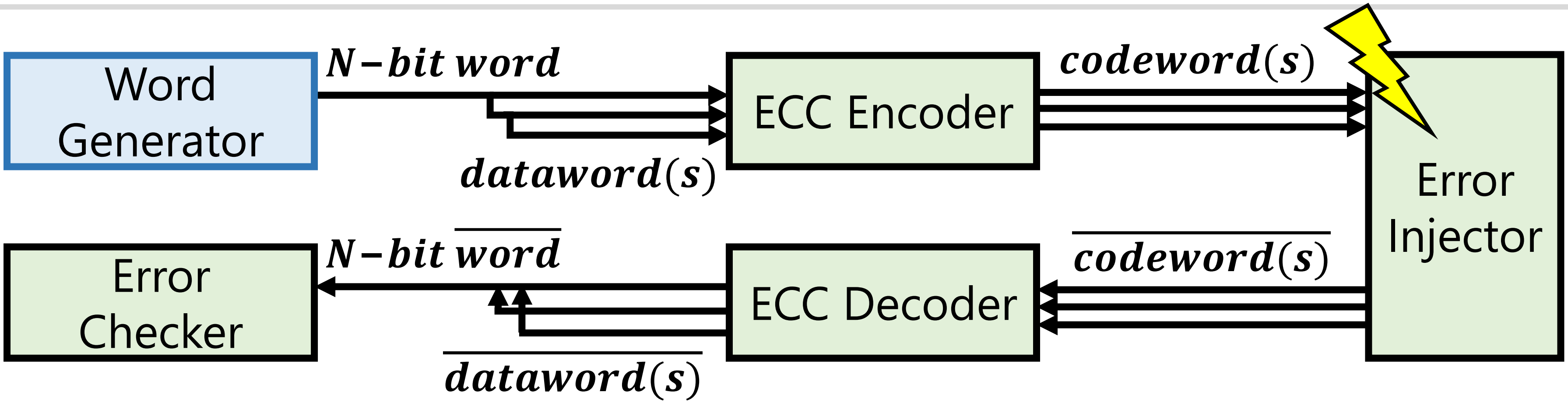


EINSim Configuration + Features



Module	***Parameters***
Word Generator	Word length Data Pattern
ECC Encoder, ECC Decoder	ECC code {type, length, strength} code details (e.g., generator polynomial)
Error Injector	Spatial error distribution
Error Checker	Measurement (e.g., #errors per word)

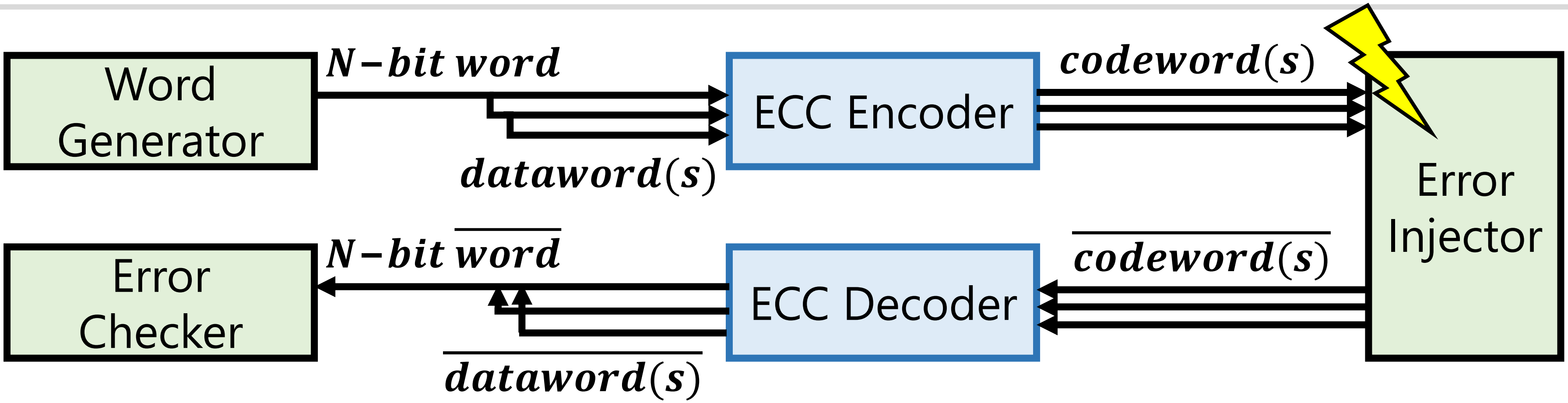
Word Generator



- Creates an $\boldsymbol{N-bit\ word}$
  - Commonly used data patterns (e.g., `0xFF`, `RANDOM`)
  - Effectively sampling the $\boldsymbol{w}$ distribution

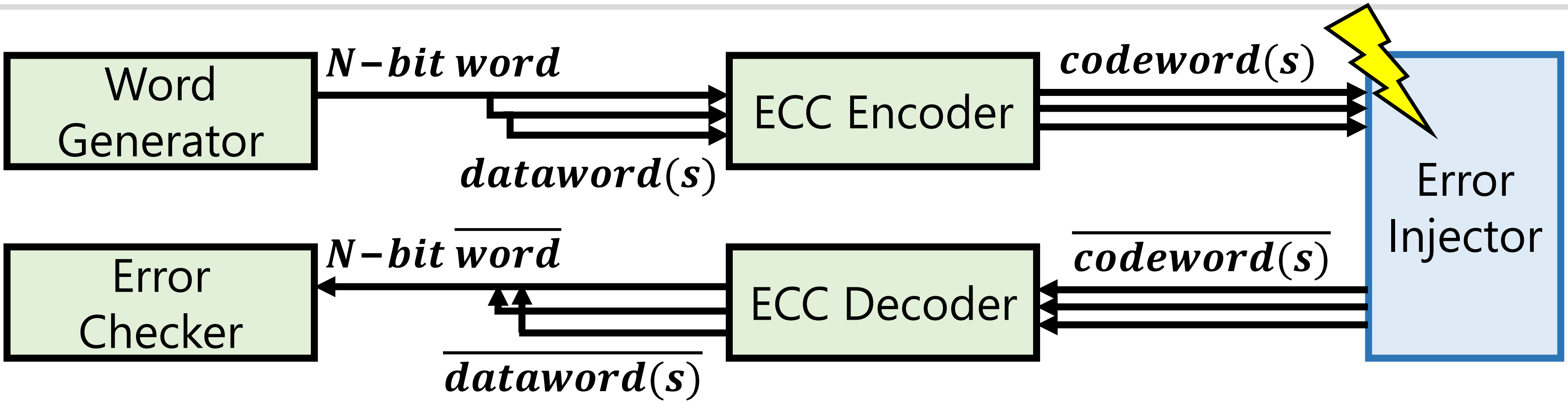
- N may be multiple $\boldsymbol{datawords}$ long
  - Useful if we don’t know how $\boldsymbol{datawords}$ are laid out
  - Split into $\boldsymbol{datawords}$ according to a configurable mapping
  - More details about this in the paper

ECC Encoder/Decoder



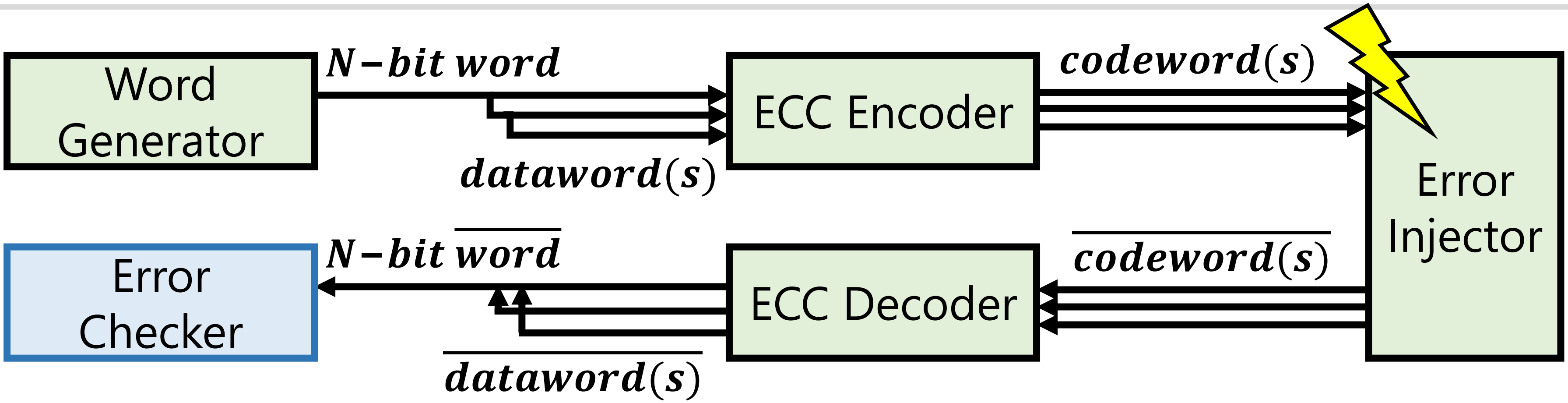
- EINSim implements ECC algorithms
  - Currently supports common codes (e.g., Hamming, BCH)
  - Modularly designed and easily extensible to others
  - Validated by hand + using unit tests (available on GitHub)
- Configurable parameters for:
  - Number of data bits, correction capability
  - Details of 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

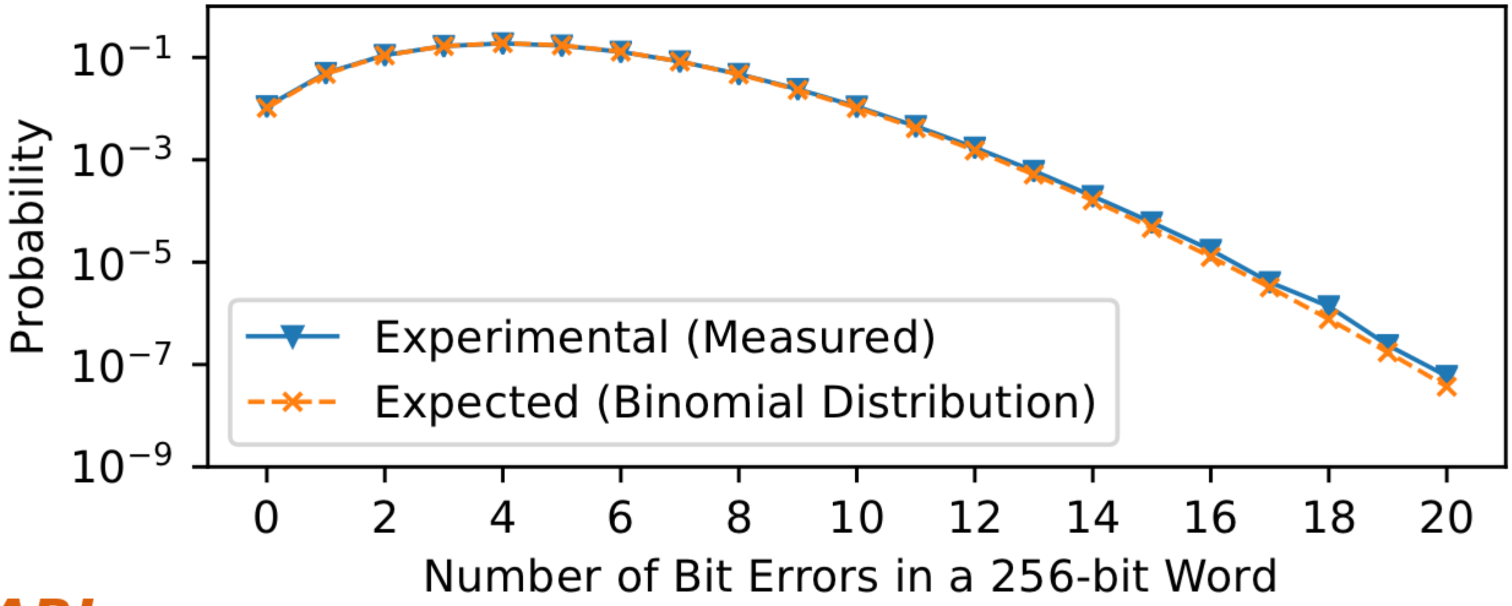
Error Checker



- Computes a configurable output distribution
  - Corresponds to the experimental measurement we make
  - E.g., number of errors per $\overline{\boldsymbol{dataword}}$ (i.e., $\overline{\boldsymbol{w}_{\boldsymbol{N}}}$)

Validating Uniform-Randomness

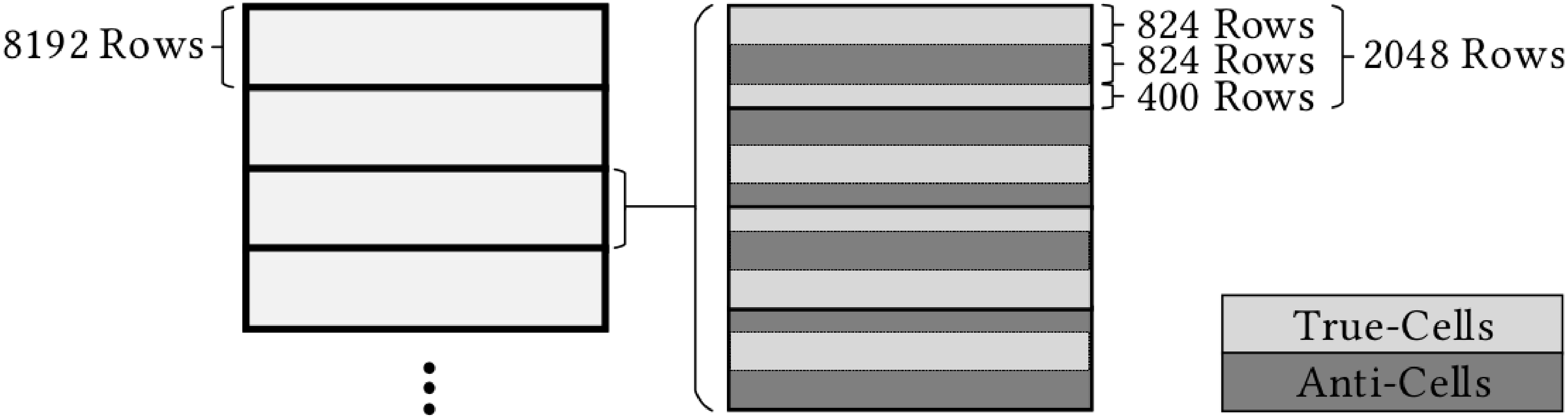
- We model data-retention errors as **uniform random**
  - Well-studied throughout prior work
  - Error count per N-bit word follows a **binomial distribution**
- We experimentally validate uniform-randomness
  - 82 LPDDR4 devices **without** on-die ECC
  - Disable refresh operations for 20s @ 60°C



SAFARI

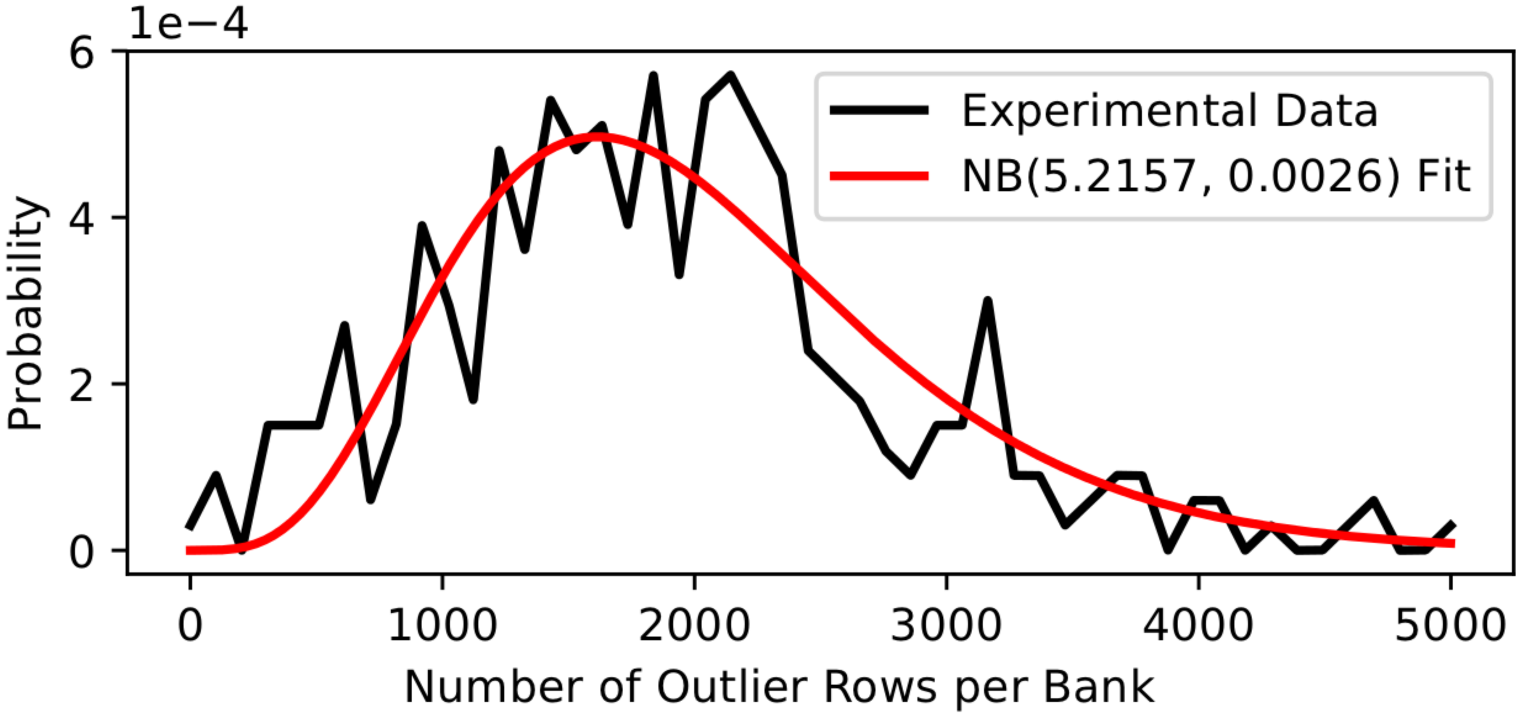
Anatomy of a DRAM Bank

- 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”

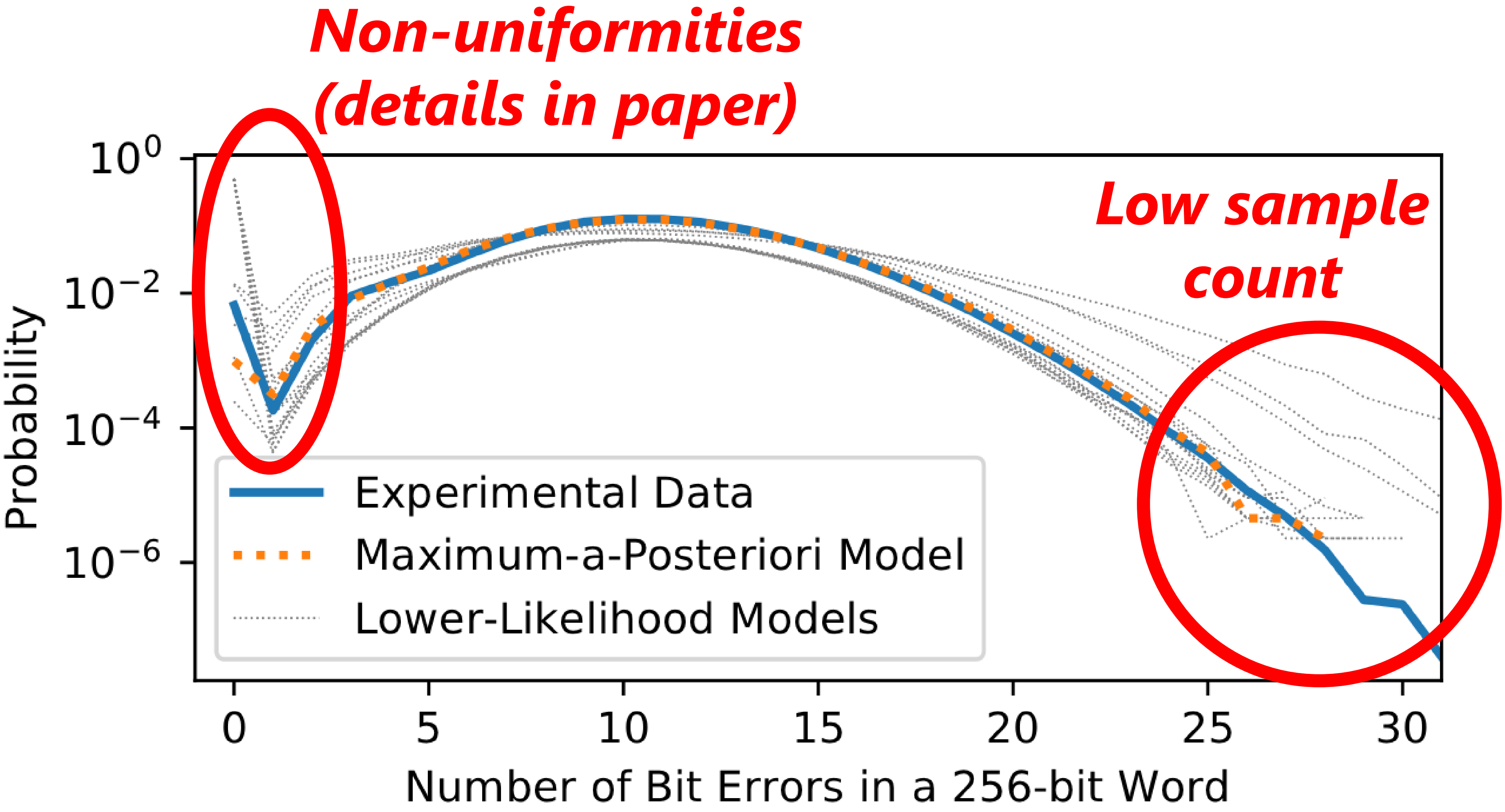


Incidence of "Outlier Rows"

- 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



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