ETH Zürich

SAFARI

Bit-Exact ECC Recovery (BEER):

Determining DRAM On-Die ECC Functions by Exploiting DRAM Data Retention Characteristics

Minesh Patel, Jeremie S. Kim

Taha Shahroodi, Hasan Hassan, Onur Mutlu

MICRO 2020 (Session 2C – Memory)

Executive Summary

Problem: DRAM on-die ECC **complicates** third-party reliability studies

- **Proprietary** design **obfuscates** raw bit errors in an **unpredictable** way
- **Interferes** with (1) design, (2) test & validation, and (3) characterization

Goal: understand **exactly how** on-die ECC obfuscates errors

Contributions:

1. **BEER**: new testing methodology that determines a DRAM chip's **unique on-die ECC function** (i.e., its parity-check matrix)
   - Exploits **ECC-function-specific** uncorrectable error patterns
   - Requires no hardware support, inside knowledge, or metadata access
2. **BEEP**: new error profiling methodology that infers the **raw bit error locations** of error-prone cells from the **observable uncorrectable errors**

BEER Evaluations:

- Apply BEER to 80 real LPDDR4 chips from 3 major DRAM manufacturers
- Show correctness in simulation for 115,300 codes (4-247b ECC words)

We hope BEER and BEEP enable valuable studies in the future

Talk Outline

Challenges Caused by Unknown On-Die ECCs

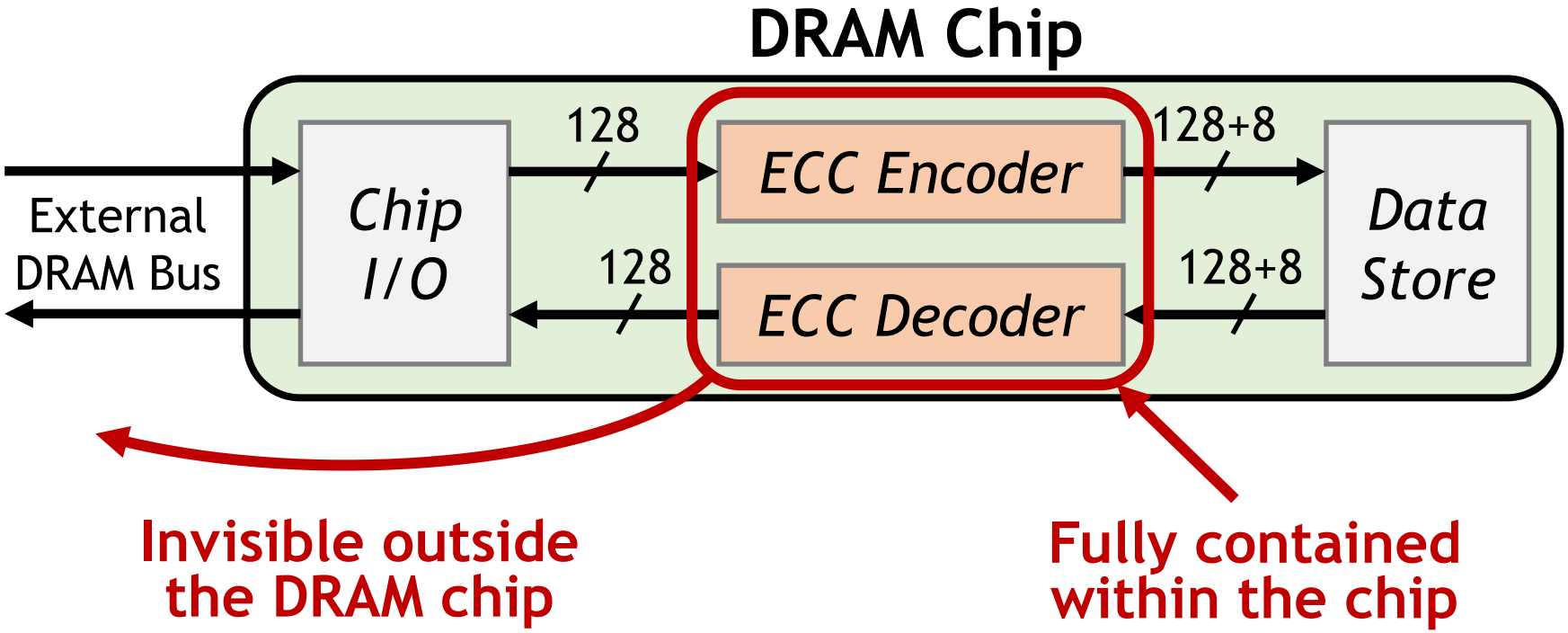
BEER: Determining the On-Die ECC Function

Evaluating BEER in Experiment and Simulation

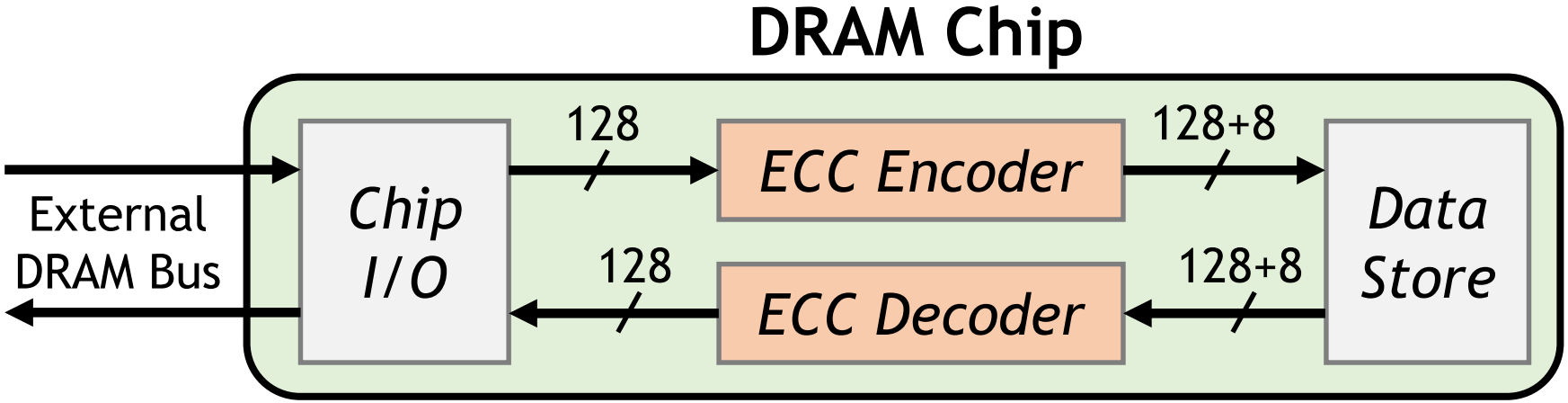
BEEP and Other Practical Use Cases for BEER

A Typical DRAM On-Die ECC Design

- 128-bit single-error correcting (SEC) Hamming code



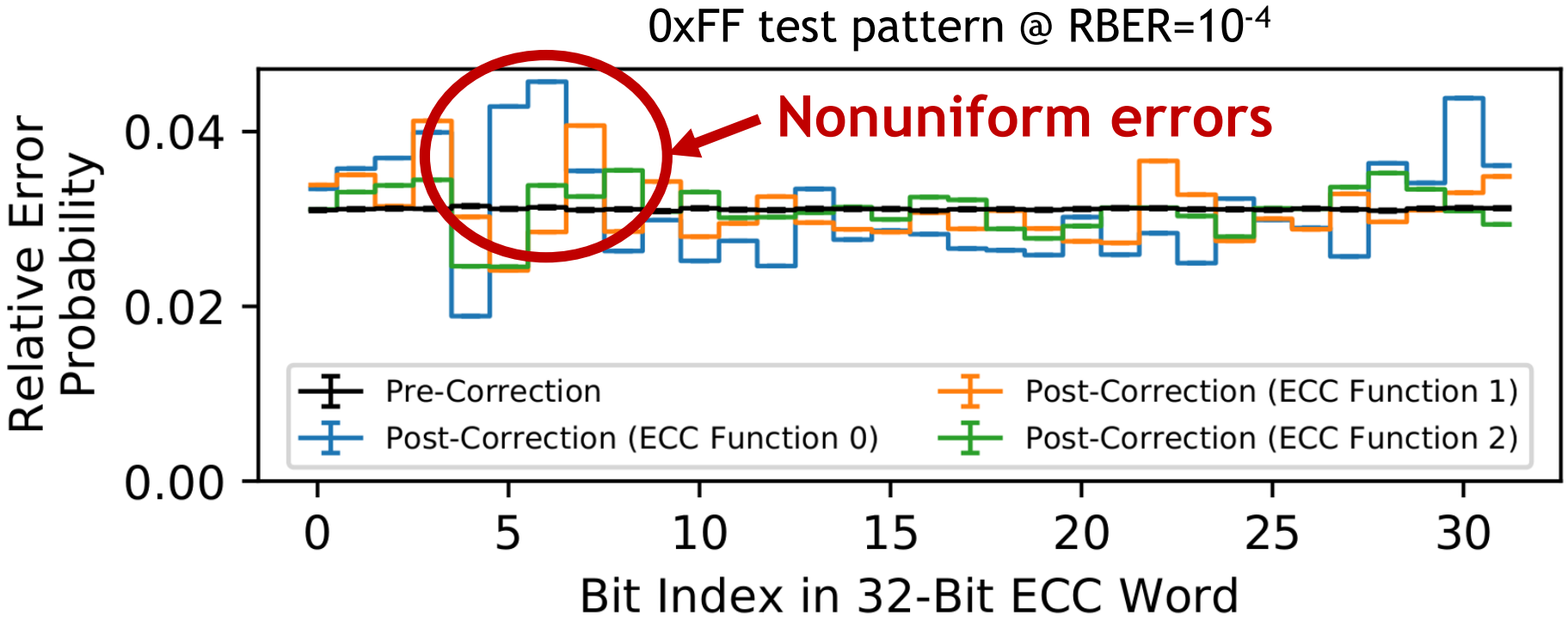
A Typical DRAM On-Die ECC Design



- Many ways to implement a 128-bit Hamming code
  - Different **ECC functions**
    - Known as **parity-check matrices** (i.e., ***H*-matrices**)
    - All correct 1 error, but act **differently** on 2+ errors
- Manufacturers are free to choose any design
  - Circuit optimization goals (e.g., area, power)
  - Details are **highly proprietary** (even under NDA)

Effect of Different On-Die ECC Designs

- Simulating **uniform-random errors** in a 32b ECC word



- 32-bit single-error correction Hamming codes
- Three different parity-check matrices

Effect of Different On-Die ECC Designs

The **same** error characteristics
can appear very **different**
with different ECC functions

Challenges for Third Parties

System Architects: Designing Error Mitigations

- On-die ECC forces system architects to support **unpredictable**, **chip-dependent** memory reliability characteristics

Test/Validation Engineers: Post-Manufacturing Testing

- On-die ECC **hides** the root-causes of uncorrectable errors and **defeats** test patterns designed to target physical cells

Research Scientists: Error-Characterization Studies

- On-die ECC **conflates** raw bit errors with ECC artifacts, effectively **obfuscating** the true physical cell characteristics

Challenges for Third Parties

These challenges all arise
from the **inability** to predict
how ECC transforms error patterns

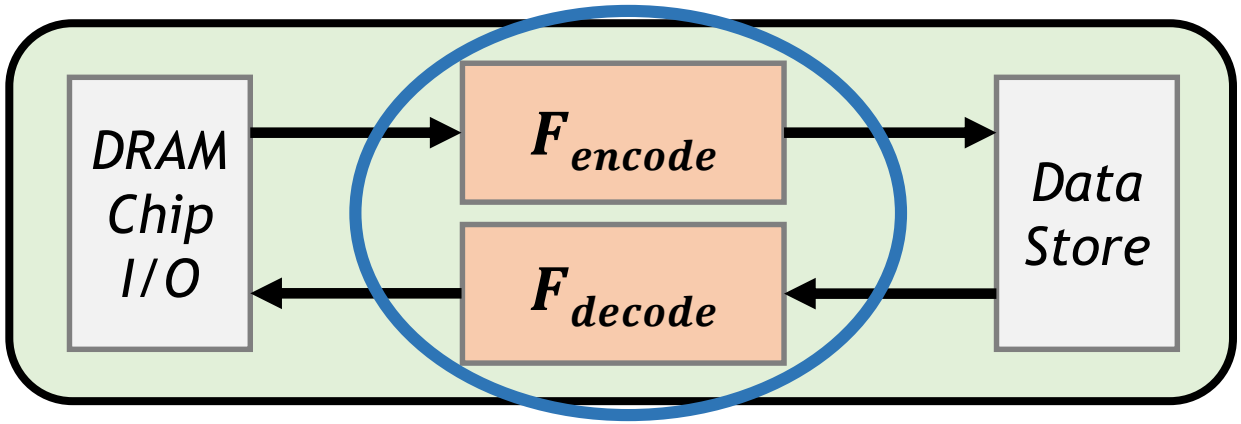
Overcoming Challenges of On-Die ECC

Our goal: Determine the on-die ECC function **without**:

(1) hardware support or tools

(2) prior knowledge about on-die ECC

(3) access to ECC metadata (e.g., syndromes)



- Reveals how on-die ECC scrambles errors (BEER)
- Allows inferring raw bit error locations (BEEP)

Talk Outline

Challenges Caused by Unknown On-Die ECCs

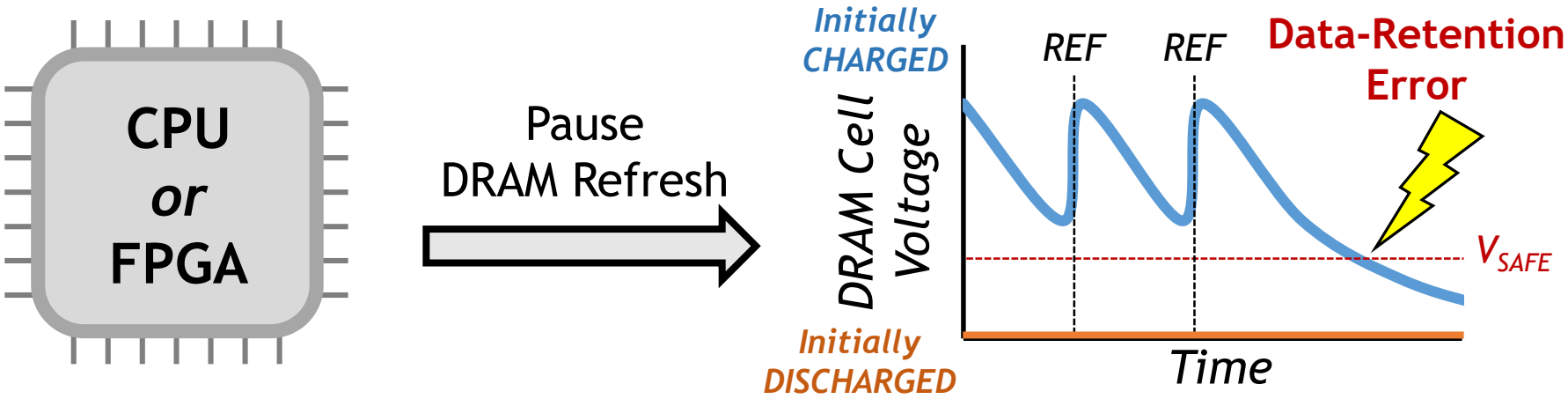
BEER: Determining the On-Die ECC Function

Evaluating BEER in Experiment and Simulation

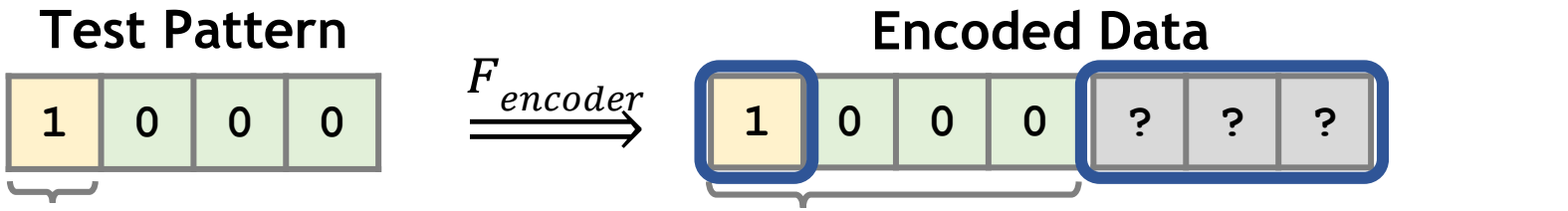
BEEP and Other Practical Use Cases for BEER

Determining the ECC Function (1/2)

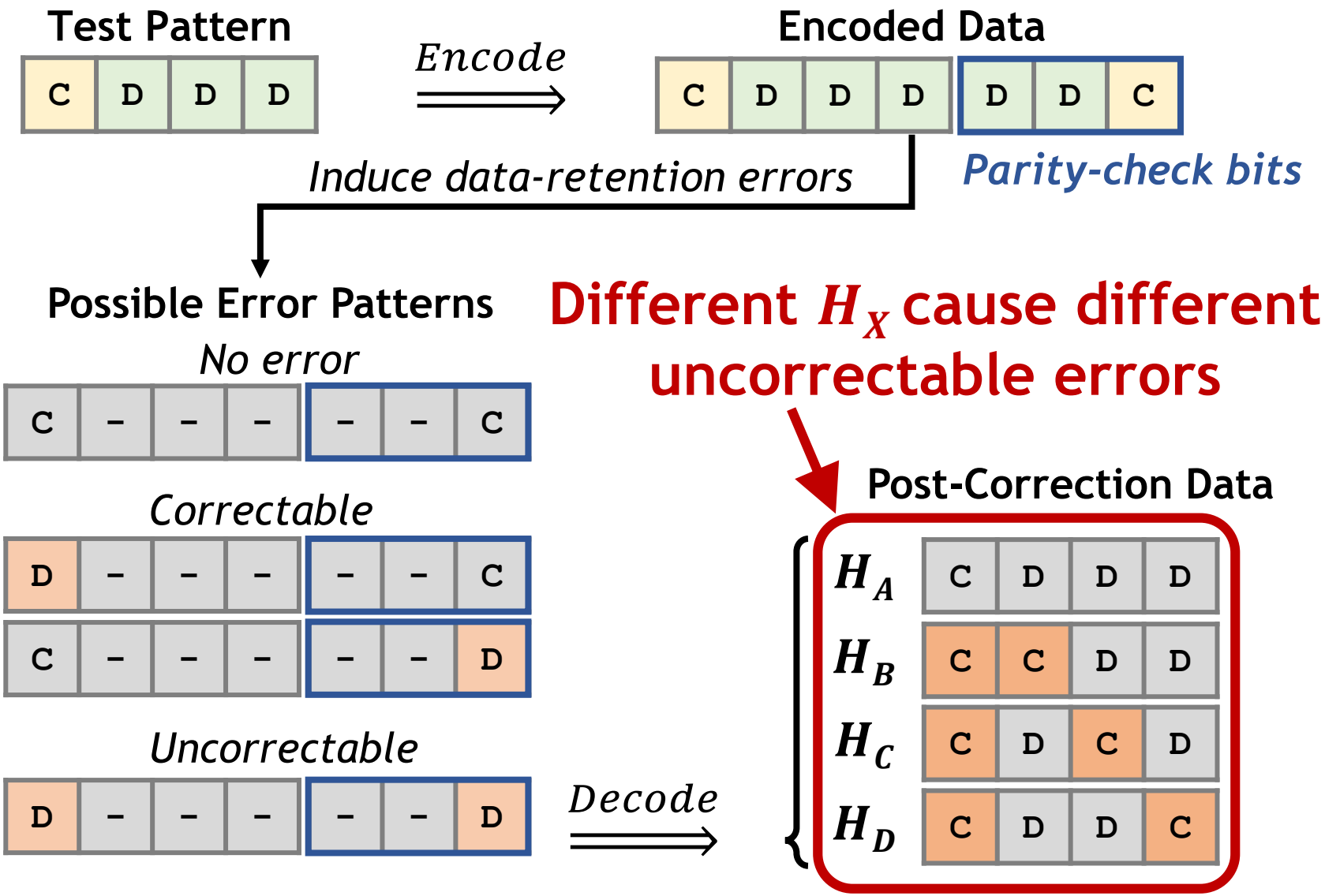
- **Key idea:** identify the ECC function by how it responds to uncorrectable data-retention errors



- Difference between CHARGED and DISCHARGED cells allows us to restrict errors to specific bit positions



Determining the ECC Function (2/2)



Determining the ECC Function (2/2)

We can differentiate ECC functions from their uncorrectable error patterns

Choosing a Set of Test Patterns

- We consider the "n-CHARGED" test patterns:

1-CHARGED = { C D D D , D C D D , ... , D D D C }

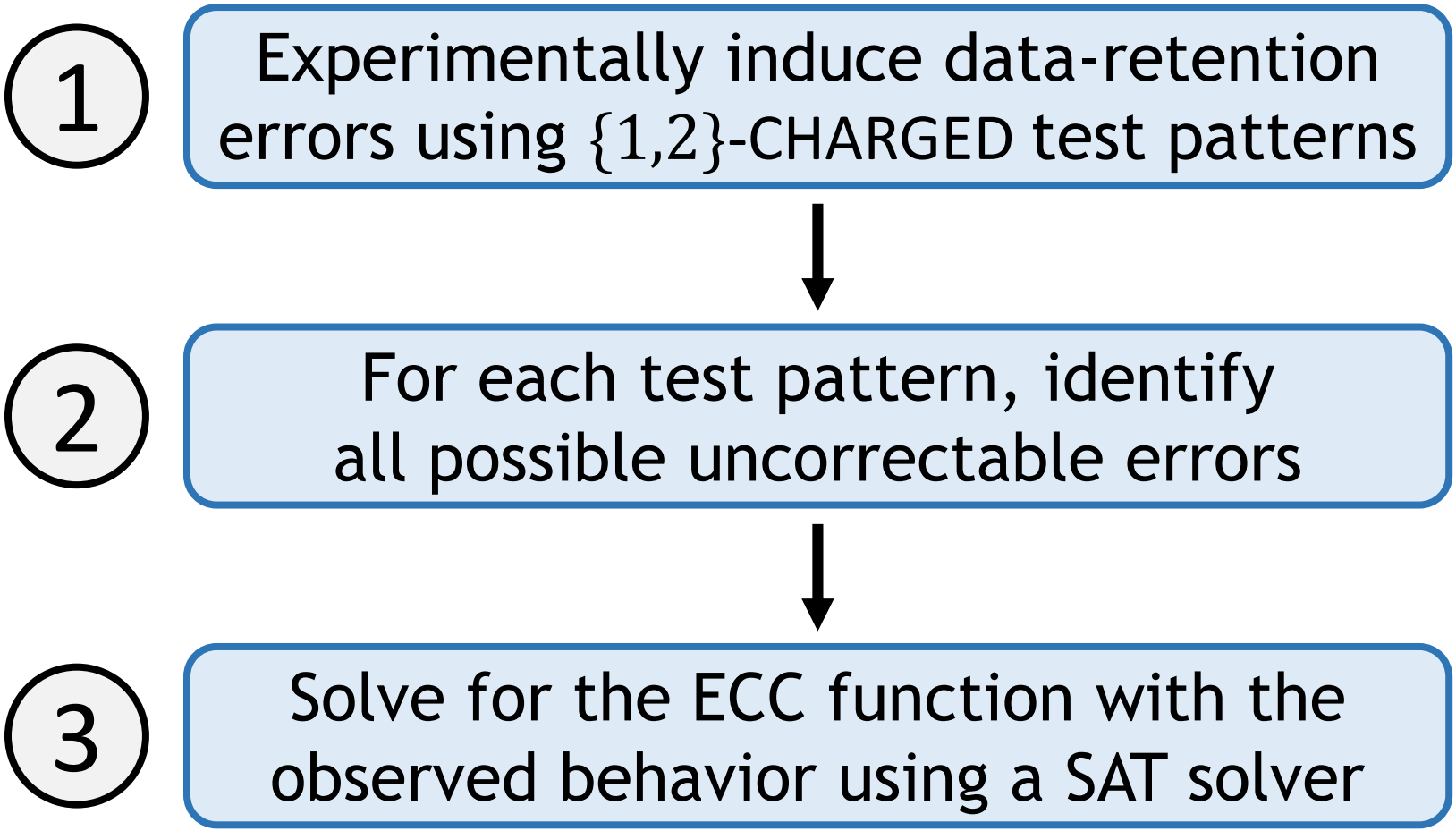
2-CHARGED = { C C D D , C D C D , ... , D D C C }

3-CHARGED = { C C C D , C D C C , ... , D D C C }

- Our paper explains that the combined $\{1,2\}$-CHARGED patterns are sufficient to identify the ECC function

- For each test pattern, we find **all possible** uncorrectable errors that can occur
  - Exploit **uniform-randomness** of data-retention errors
  - Even one DRAM chip provides millions of samples
    - E.g., 2 GiB DRAM module yields 2^{24} 128-bit words

BEER: Bit-Exact ECC Recovery



Talk Outline

Challenges Caused by Unknown On-Die ECCs

BEER: Determining the On-Die ECC Function

Evaluating BEER in Experiment and Simulation

BEEP and Other Practical Use Cases for BEER

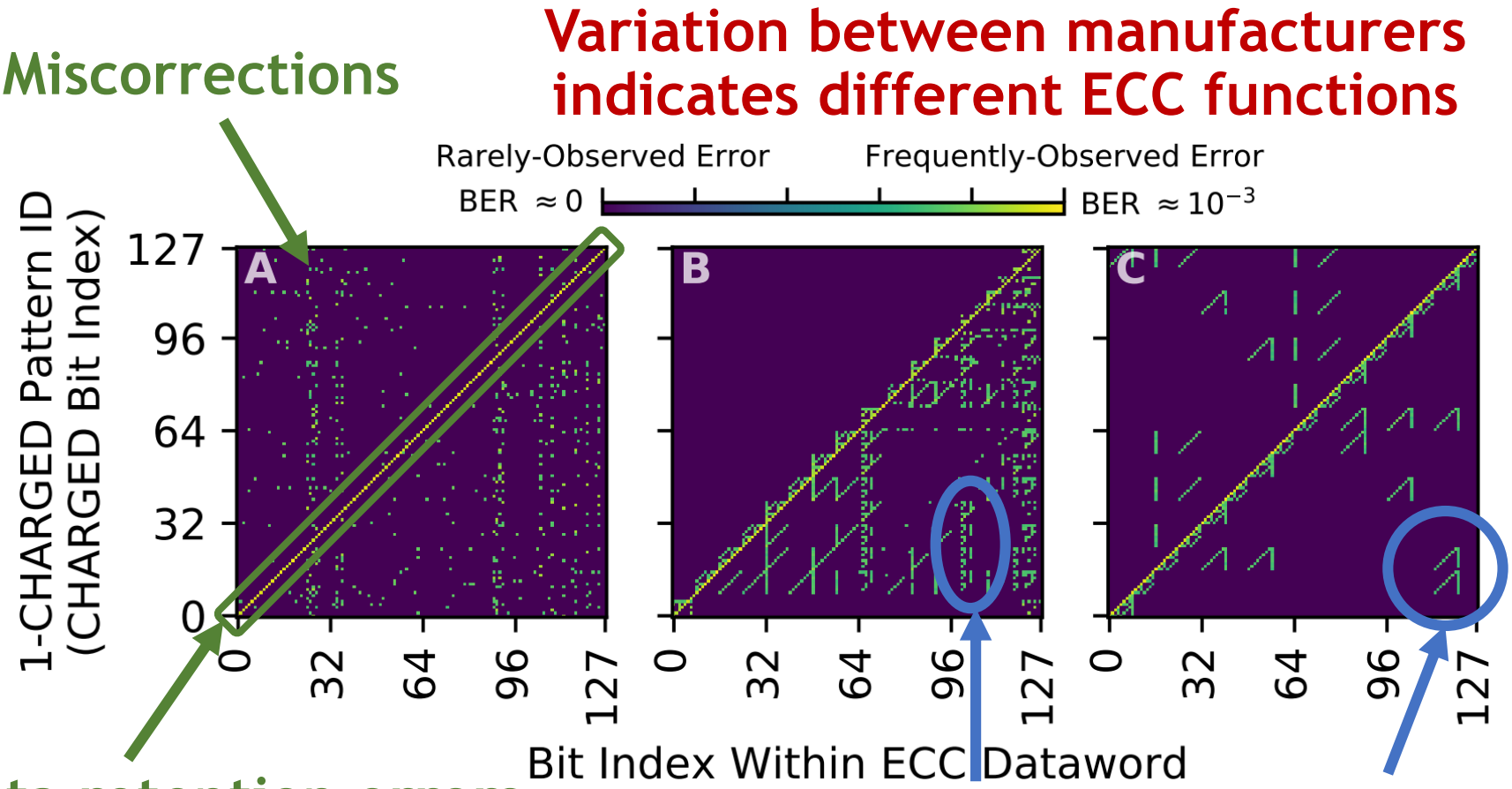
Experimental Methodology

- 80 LPDDR4 chips from 3 DRAM manufacturers
  - Manufacturers anonymized as ‘A’, ‘B’, and ‘C’
  - Temperature-controlled testing infrastructure
  - Control over DRAM timings (including refresh)

- Refresh windows between 1-30 minutes at 30-80°C
  - Leads to bit error rates (BERs) between 10^{-7} and 10^{-3}
  - BERs far larger than other soft error rates

Applying BEER to LPDDR4 Chips

- Study the uncorrectable errors in the 1-CHARGED patterns



Applying BEER to LPDDR4 Chips

1. **Different manufacturers** appear to use **different** on-die ECC functions
2. Chips of the **same model number** appear to use **identical** ECC functions

(shown in our paper)

Solving for the ECC Function

- We use the **Z3[†] SAT solver** to identify the H-matrix
  - We demonstrate BEER for SEC Hamming codes, but it should readily extend to **all** linear block codes (e.g., BCH)
- We **open-source** our BEER implementation on **GitHub**
  - **https://github.com/CMU-SAFARI/BEER**
- Unfortunately, we face two **limitations** to validation:
  1. No way to check the **final results** since we cannot see into the on-die ECC implementation
  2. We cannot share our final matrices due to **confidentiality** reasons

[†]L. De Moura and N. Bjørner, "Z3: An Effient SMT Solver," TACAS, 2008.

Solving for the ECC Function

We validate BEER in **simulation** to:

1. Evaluate **correctness**
2. **Overcome** confidentiality issues
3. Test a **larger** set of ECC codes

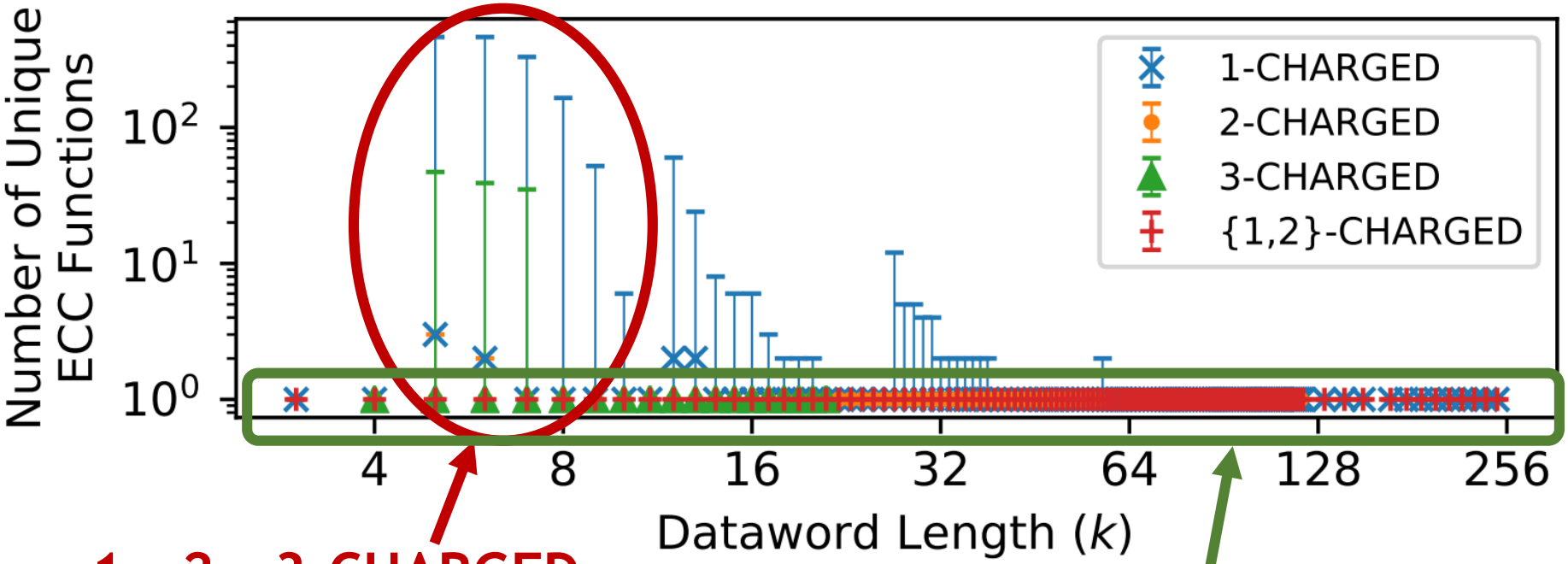
Simulation Methodology

- We use the EINSim† DRAM error-correction simulator
- We simulate 115,300 different SEC Hamming codes
  - ECC dataword lengths from 4 to 247 bits
  - 1-, 2-, 3-, and {1,2}-CHARGED test patterns
- For each test pattern:
  - Simulate 10^9 ECC words (≈14.9 GiB for 128-bit words)
  - Simulate data-retention errors with BER between 10^{-5} and 10^{-2}

†Patel et al., "*Understanding and Modeling On-Die Error Correction in Modern DRAM: An Experimental Study Using Real Devices*," DSN, 2019.

BEER Correctness Evaluation

- Evaluate the number of SAT solutions found by BEER
  - Shows whether the ‘unique’ solution is identified



1-, 2-, 3-CHARGED patterns individually do not always succeed

{1,2} -CHARGED patterns succeed for all test cases

BEER Correctness Evaluation

BEER **successfully** identifies
the ECC function using
the **{1,2}-CHARGED** test patterns

Talk Outline

Challenges Caused by Unknown On-Die ECCs

BEER: Determining the On-Die ECC Function

Evaluating BEER in Experiment and Simulation

BEEP and Other Practical Use Cases for BEER

Practical Use Cases for BEER

- We provide 5 **use cases** in our paper to show how knowing the ECC function is **useful in practice**

Error Profiling	**BEEP:** identifying **raw bit error** locations corresponding to observed **post-correction errors**
System Design	Architecting DRAM controller **error mitigations** that are **informed** about on-die ECC
Testing	Crafting **worst-case test patterns** to enable **efficient** testing and validation
	Root-cause analysis for uncorrectable errors
Error Characterization	Studying the **statistical properties** of **raw bit errors** (e.g., spatial distributions)

Other Information in the Paper

- **Formalism** for BEER and the n-CHARGED test patterns
- **BEER** evaluations using **experiment** and **simulation**
  - Sensitivity to experimental noise
  - Analysis of experimental runtime
  - Practicality of the SAT problem (i.e., runtime, memory)
- **BEEP** evaluations in **simulation**
  - Accuracy at different error rates
  - Sensitivity to different ECC codes and word sizes
- Detailed discussion of **use-cases** for BEER
- Discussion on BEER's **requirements** and **limitations**

Executive Summary

Problem: DRAM on-die ECC **complicates** third-party reliability studies

- **Proprietary** design **obfuscates** raw bit errors in an **unpredictable** way
- **Interferes** with (1) design, (2) test & validation, and (3) characterization

Goal: understand **exactly how** on-die ECC obfuscates errors

Contributions:

1. **BEER**: new testing methodology that determines a DRAM chip's **unique on-die ECC function** (i.e., its parity-check matrix)
2. **BEEP**: new error profiling methodology that infers the **raw bit error locations** of error-prone cells from the **observable uncorrectable errors**

BEER Evaluations:

- Apply BEER to 80 real LPDDR4 chips from 3 major DRAM manufacturers
- Show correctness in simulation for 115,300 codes (4-247b ECC words)

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

We hope that both BEER and BEEP enable many valuable studies going forward

ETH Zürich

SAFARI

Bit-Exact ECC Recovery (BEER):

Determining DRAM On-Die ECC Functions by Exploiting DRAM Data Retention Characteristics

Minesh Patel, Jeremie S. Kim

Taha Shahroodi, Hasan Hassan, Onur Mutlu

MICRO 2020 (Session 2C – Memory)