Two key goals of this course are

- to understand how a computing system works underneath the software layer and how decisions made in hardware affect the software/programmer

- to enable you to be comfortable in making design and optimization decisions that cross the boundaries of different layers and system components
Another Example

- DRAM Refresh
DRAM in the System

*Die photo credit: AMD Barcelona*
A DRAM cell consists of a capacitor and an access transistor. It stores data in terms of charge in the capacitor. A DRAM chip consists of (10s of 1000s of) rows of such cells.
DRAM Refresh

- DRAM capacitor charge leaks over time

- The memory controller needs to refresh each row periodically to restore charge
  - Activate each row every N ms
  - Typical N = 64 ms

- Downsides of refresh
  -- **Energy consumption**: Each refresh consumes energy
  -- **Performance degradation**: DRAM rank/bank unavailable while refreshed
  -- **QoS/predictability impact**: (Long) pause times during refresh
  -- **Refresh rate limits DRAM capacity scaling**
First, Some Analysis

- Imagine a system with 8 ExaByte DRAM ($2^{63}$ bytes)
- Assume a row size of 8 KiloBytes ($2^{13}$ bytes)

- How many rows are there?
- How many refreshes happen in 64ms?
- What is the total power consumption of DRAM refresh?
- What is the total energy consumption of DRAM refresh during a day?

- A good exercise...
A Leaky DRAM Cell
DRAM Leakage in ISCA-50
25-Year Retrospective Issue
RAIDR: Retention-Aware Intelligent Intelligent DRAM Refresh

Jamie Liu, Ben Jaiyen, Richard Veras, and Onur Mutlu, "RAIDR: Retention-Aware Intelligent DRAM Refresh"
[Invited Retrospective at 50 Years of ISCA, 2023 (pdf)]
Analysis of Data Retention Failures [ISCA’13]

Josie Liu, Ben Jaiyen, Yoongu Kim, Chris Wilkerson, and Onur Mutlu,
"An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms"
Proceedings of the 40th International Symposium on Computer Architecture (ISCA), Tel-Aviv, Israel, June 2013. Slides (ppt) Slides (pdf)
[Invited Retrospective at 50 Years of ISCA, 2023 (pdf)]

An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms

Jamie Liu*
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
jamiel@alumni.cmu.edu

Ben Jaiyen*
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
bjaiyen@alumni.cmu.edu

Yoongu Kim
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
yoonguk@ece.cmu.edu

Chris Wilkerson
Intel Corporation
2200 Mission College Blvd.
Santa Clara, CA 95054
chris.wilkerson@intel.com

Onur Mutlu
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
onur@cmu.edu
First RowHammer Analysis [ISCA’14]

- Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu,

"Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors"
[Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)] [Source Code and Data] [Lecture Video (1 hr 49 mins), 25 September 2020]

One of the 7 papers of 2012-2017 selected as Top Picks in Hardware and Embedded Security for IEEE TCAD (link).

Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors

Yoongu Kim¹  Ross Daly*  Jeremie Kim¹  Chris Fallin*  Ji Hye Lee¹
Donghyuk Lee¹  Chris Wilkerson²  Konrad Lai  Onur Mutlu¹

¹Carnegie Mellon University  ²Intel Labs
RAIDR Retrospective [ISCA 2012]

Retrospective: RAIDR: Retention-Aware Intelligent DRAM Refresh

Our Mafia
ETH Zurich

Abstract—Dynamic Random Access Memory (DRAM) is the dominant memory technology used to build main memory systems in all computers. A fundamental shortcoming of current DRAM refresh control is that it fails to meet the requirements of low-power, high-performance systems. A novel technique, RAIDR, is proposed as a way to address DRAM refresh inefficiencies. RAIDR is a new architecture that addresses the refresh problem from a modern computing systems perspective, demonstrating a high refresh performance with substantially lower DRAM chips expected in the future. It provides a new alternative for providing a performance and power benefit to next generation computing systems. RAIDR is designed to maintain a high refresh efficiency in DRAM chips despite the increase in DRAM refresh requirements with time. The key idea is to group the DRAM rows into small bit-sliced columns, and then refresh this group of rows in parallel. This approach allows us to exploit the DRAM refresh data rate and reduce the latency and power consumption by increasing the DRAM refresh efficiency in a DRAM chip.

The RAIDR development, despite later works that provided improved techniques and new architectures, has set the future stage on the DRAM refresh problem (and more generally in memory technology).

I. BACKGROUND, APPROACH & MISSION

At the time we began our focus on solving the DRAM refresh problem (i.e., data retention), RAIDR, in 2008, the research group at ETH Zurich, led by O. Mutlu, had already been working on memory controllers and memory technology scaling issues, motivated by many challenges in memory systems, in particular the DRAM technology. We have started working on this topic as early as 2009. Our initial work on memory systems started during my time at Microsoft Research from 2006 to 2009. At that time, we had the main idea of using multiple-core processors (e.g., Xeon 5000) to develop parallelism and frequency scaling to address future DRAM energy (e.g., [1]) and architectural approaches to memory technology scaling that were much less energy-efficient compared to the DRAM technology. As a result, we wanted to explore the possibility of designing a new architecture that could improve the energy efficiency of the system by providing a better balance between the DRAM refresh and refresh requirements.

RAIDR is a product of this approach. Our work on data retention in DRAM especially increased in the years following RAIDR. We proposed new ideas to employ low-level issues in DRAM, and we still have some ongoing work in the area. RAIDR has been proposed as a way to address DRAM refresh inefficiencies. RAIDR is a new architecture that addresses the refresh problem from a modern computing systems perspective, demonstrating a high refresh performance with substantially lower DRAM chips expected in the future. RAIDR provides a new alternative for providing a performance and power benefit to next generation computing systems. RAIDR is designed to maintain a high refresh efficiency in DRAM chips despite the increase in DRAM refresh requirements with time. The key idea is to group the DRAM rows into small bit-sliced columns, and then refresh this group of rows in parallel. This approach allows us to exploit the DRAM refresh data rate and reduce the latency and power consumption by increasing the DRAM refresh efficiency in a DRAM chip.

II. BUILDING ON RAIDR AND MAKING IT WORK

We believe RAIDR enabled a refreshing approach to DRAM refresh. Its impact could be the works it has inspired that have rigorously examined the questions of how to perform accurate DRAM data retention timing (e.g., [2]) and how to overcome the interference of process and temperature variations (e.g., [3]). RAIDR has enabled others to develop methods for reducing the impact of refresh on performance and power, and a new DRAM refresh architecture has been developed to minimize the impact of refresh on performance.

We wanted to make RAIDR work in a real system setting. To this end, we collaborated with Intel, we developed an FPGA-based flexible DRAM testing infrastructure that enabled us to accurately measure data retention times of DRAM chips. Using this infrastructure, we were able to conduct a detailed study of the impact of refresh on performance and power, and we were able to show that RAIDR has a significant potential to improve the energy efficiency of DRAM chips. RAIDR was also able to provide a new tool for measuring the impact of refresh on performance and power, and a new DRAM refresh architecture has been developed to minimize the impact of refresh on performance.

III. REFERENCES


https://people.inf.ethz.ch/omutlu/pub/RAIDR_50YearsOfISCA-Retrospective_isca23.pdf
Retention Analysis Retrospective [ISCA 2013]

Retention Analysis: An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms

Omar Mutla
ETH Zürich

Abstract - DRAM is the prevalent main memory technology used in modern data centers. Unfortunately, DRAM cells are volatile, so they need to be refreshed regularly to maintain current value. Refresh, the refresh rate, is a critical parameter that determines the minimum refresh rate of a DRAM chip to maintain data integrity. DRAM refresh rate is also an important factor in ensuring that memory can be used efficiently and effectively in real-world applications. In this work, we present a comprehensive study of the refresh behavior of a large number of modern DRAM chips and analyze the impact of refresh rates on memory performance and energy efficiency. Our study reveals that the refresh rate is one of the most important factors affecting DRAM performance and energy efficiency. We provide a detailed analysis of the refresh behavior of several different DRAM chips, including both consumer and server-grade chips. Our results show that the refresh rate can vary significantly, even within the same technology node. This variation can have a significant impact on memory performance and energy efficiency. Finally, we propose several techniques to improve DRAM refresh efficiency, including hardware and software-based approaches.

1. INTRODUCTION
Our group has been working on the DRAM refresh problem since 2010 and our major work RADIUS [13] was published at ISCA 2012. Our goal in RADIUS was to improve refresh operations at low cost by refreshing each DRAM row only as frequently as required to maintain data integrity. Refresh rate is the time interval between two consecutive refresh operations and is the most important factor affecting DRAM performance and energy efficiency. In order to reduce the refresh rate, we developed large performance improvements and energy savings with a simple memory controller called RADIUS. However, the refresh rate is also limited by the refresh time of the row. As described in a separate paper [14], the refresh time is the time required to refresh all of the rows of a DRAM chip. Refresh time is one of the most important factors affecting DRAM performance and energy efficiency. Our study reveals that the refresh rate can vary significantly, even within the same technology node. This variation can have a significant impact on memory performance and energy efficiency. Finally, we propose several techniques to improve DRAM refresh efficiency, including hardware and software-based approaches.

2. CONTRIBUTIONS AND IMPACT
Our paper is the first to comprehensively examine data retention time behavior of modern DRAM chips, uncovering real data and insights on two major phenomena that make retention time identification extremely challenging. Prior works were limited to simulation or had only a small sample size, and most did not examine modern DDR3 DRAM chips or technology. Many device designers and system architects believed that ECC could reliably detect errors, whereas our work demonstrated that ECC alone cannot reliably detect errors. In contrast, this work clearly demonstrates that DPD and VRT phenomena are significant issues that must be addressed for proper operation of DRAM-based systems and their effects are getting worse as DRAM scales to smaller technology node. This work also provides insights into the mechanism by which refresh operations are used to force refresh and volatile data to be re-read into memory, which is critical for DRAM design and data recovery.

3. SUMMARY OF FUTURE WORK
In summary, our work provides a comprehensive understanding of the DRAM refresh problem and highlights several key findings that could lead to improved DRAM refresh efficiency. We believe that our work will be of great interest to device designers and system architects, as well as to those working on memory research and development. We hope that our work will help to improve the performance and energy efficiency of modern DRAM chips, and we believe that our work will provide valuable insights into the future of DRAM technology.

4. ACKNOWLEDGMENTS
This work was supported in part by the Swiss National Science Foundation grant 200021-156893/1.

5. REFERENCES

https://people.inf.ethz.ch/omutlu/pub/DRAMDataRetention_50YearsOfISCA-Retrospective_isca23.pdf
RowHammer Retrospective [ISCA 2014]

Retrospective: Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors

Omer Maza
ETH Zurich

Abstract—Our ISCA 2014 paper [1] provided the first scientific study of the RowHammer phenomenon, a memory disturbance mechanism discovered in 2013 that exposes a critical vulnerability in many DRAM chips. It demonstrated that new DRAM designs are still vulnerable to RowHammer attacks, even though they employ complex mitigation techniques. Our work helped to guide future research into these vulnerabilities and helped to mitigate RowHammer attacks.

I. BACKGROUND AND CIRCUMSTANCES

Our study of the RowHammer problem and creation of our RowHammer simulator was a result of a convergence of multiple factors. First, my group was working on DRAM technology scaling issues since 2006. We were very focused on the problem of memory failure mechanisms that appear due to aggressive technology scaling. To study such failures, our team conducted comprehensive experiments in various environments and settings, including in our lab and in real DRAM chips.

The second factor that influenced our work was the research community’s growing interest in the RowHammer phenomenon, which had been discovered in 2013. As we delved deeper into the problem, we began to see a pattern of behavior that could be exploited to attack DRAM chips. We continued to explore this pattern through our RowHammer simulator.

II. MAJOR CONTRIBUTIONS AND INFLUENCE

The major contribution of our paper was to provide a detailed analysis of the RowHammer phenomenon and to demonstrate the potential for RowHammer attacks on real DRAM chips. Our work quickly gained attention from the research community, who saw it as a valuable contribution to the field.

Our work also had a large impact on both industry and academia. Many companies and researchers have cited our paper as a reference in their work, and our RowHammer simulator has become a valuable tool for studying the phenomenon.

The influence of our work is still felt today, as researchers continue to explore the potential for RowHammer attacks and to develop new mitigation techniques.

III. SUMMARY AND FUTURE OUTLOOK

Since 2012, RowHammer has become much worse due to technology scaling and the lack of mitigation. New attacks continue to be discovered, and new methods of defense are being developed. The future of RowHammer research remains exciting, and we look forward to continued progress in this area.

References


DRAM Refresh Overhead
Refresh Overhead: Performance

Refresh Overhead: Energy

How Do We Solve the Problem?

- Observation: All DRAM rows are refreshed every 64ms.

- Critical thinking: Do we need to refresh all rows every 64ms?

- What if we knew what happened underneath (in DRAM cells) and exposed that information to upper layers?
Underneath: Retention Time Profile of DRAM

64-128ms

>256ms

128-256ms

Aside: Why Do We Have Such a Profile?

- Answer: Manufacturing is not perfect
- Not all DRAM cells are exactly the same
- Some cells are more leaky than others
- This is called Manufacturing Process Variation
Opportunity: Taking Advantage of This Profile

- Assume we know the retention time of each row exactly

- What can we do with this information?

- Who do we expose this information to?

- How much information do we expose?
  - Affects hardware/software overhead, power, verification complexity, cost

- How do we determine this profile information?
  - Also, who determines it?

Experimental Infrastructure (DRAM)

DRAM Testing Platform and Method

- **Test platform:** Developed a DDR3 DRAM testing platform using the Xilinx ML605 FPGA development board
  - Temperature controlled

- **Tested DRAM chips:** 248 commodity DRAM chips from five manufacturers (A,B,C,D,E)

- Seven families based on equal capacity per device:
  - A 1Gb, A 2Gb
  - B 2Gb
  - C 2Gb
  - D 1Gb, D 2Gb
  - E 2Gb
Underneath: Retention Time Profile of DRAM

64-128ms

>256ms

128-256ms

Retention Time of DRAM Rows

- Observation: Overwhelming majority of DRAM rows can be refreshed much less often without losing data.

- Only ~1000 rows in 32GB DRAM need refresh every 256 ms, but we refresh all rows every 64 ms.

Key Idea of RAIDR: Refresh weak rows more frequently, all other rows less frequently.

RAIDR: Heterogeneous Refresh [ISCA’12]

- Jamie Liu, Ben Jaiyen, Richard Veras, and Onur Mutlu, "RAIDR: Retention-Aware Intelligent DRAM Refresh"
  [Invited Retrospective at 50 Years of ISCA, 2023 (pdf)]

RAIDR: Retention-Aware Intelligent DRAM Refresh

  Jamie Liu    Ben Jaiyen    Richard Veras    Onur Mutlu
  Carnegie Mellon University
RAIDR: Mechanism

1. Profiling: Identify the retention time of all DRAM rows

- 64-128ms
- >256ms

1.25KB storage in controller for 32GB DRAM memory

- 128-256ms
- check the bins to determine refresh rate of a row

RAIDR: Results and Takeaways

- System: 32GB DRAM, 8-core; Various workloads
- RAIDR hardware cost: 1.25 kB (2 Bloom filters)
- Refresh reduction: 74.6%
- Dynamic DRAM energy reduction: 16%
- Idle DRAM power reduction: 20%
- Performance improvement: 9%
- Benefits increase as DRAM scales in density

![Graph showing energy per access and weighted speedup across different device capacities with RAIDR and Auto compared.]
RAIDR: Retention-Aware Intelligent DRAM Refresh

Jamie Liu, Ben Jaiyen, Richard Veras, and Onur Mutlu,
"RAIDR: Retention-Aware Intelligent DRAM Refresh"
[Invited Retrospective at 50 Years of ISCA, 2023 (pdf)]
If You Are Interested … Further Readings

- Onur Mutlu,
  "Memory Scaling: A Systems Architecture Perspective"
  Technical talk at MemCon 2013 (MEMCON), Santa Clara, CA, August 2013.
  Slides (pptx) (pdf) Video

- Kevin Chang, Donghyuk Lee, Zeshan Chishti, Alaa Alameldeen, Chris Wilkerson, Yoongu Kim, and Onur Mutlu,
  "Improving DRAM Performance by Parallelizing Refreshes with Accesses"
Takeaway 1

Breaking the abstraction layers (between components and transformation hierarchy levels) and knowing what is underneath enables you to understand and solve problems.
Takeaway 2

Cooperation between multiple components and layers can enable more effective solutions and systems.
Digging Deeper: Making RAIDR Work

“Good ideas are a dime a dozen”

“Making them work is oftentimes the real contribution”
Recall: RAIDR: Mechanism

1. Profiling: Identify the retention time of all DRAM rows
   → can be done at design time or during operation

2. Binning: Store rows into bins by retention time
   → use Bloom Filters for efficient and scalable storage
   
   **1.25KB storage in controller for 32GB DRAM memory**

3. Refreshing: Memory controller refreshes rows in different bins at different rates
   → check the bins to determine refresh rate of a row

1. Profiling

To profile a row:
1. Write data to the row
2. Prevent it from being refreshed
3. Measure time before data corruption

<table>
<thead>
<tr>
<th>Initially</th>
<th>Row 1</th>
<th>Row 2</th>
<th>Row 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>After 64 ms</td>
<td>111111111...</td>
<td>111111111...</td>
<td>111111111...</td>
</tr>
<tr>
<td>After 128 ms</td>
<td>110111111... (64–128ms)</td>
<td>111111111...</td>
<td>111111111...</td>
</tr>
<tr>
<td>After 256 ms</td>
<td>111110111... (128–256ms)</td>
<td>111111111... (&gt;256ms)</td>
<td>111111111...</td>
</tr>
</tbody>
</table>
DRAM Retention Time Profiling

- Q: Is it really this easy?
- A: No...
Two Challenges to Retention Time Profiling

- Data Pattern Dependence (DPD) of retention time
- Variable Retention Time (VRT) phenomenon
Two Challenges to Retention Time Profiling

- **Challenge 1: Data Pattern Dependence (DPD)**
  - Retention time of a DRAM cell depends on its value and the values of cells nearby it
  - When a row is activated, all bitlines are perturbed simultaneously
Electrical noise on the bitline affects reliable sensing of a DRAM cell. The magnitude of this noise is affected by values of nearby cells via:

- Bitline-bitline coupling → electrical coupling between adjacent bitlines
- Bitline-wordline coupling → electrical coupling between each bitline and the activated wordline

Retention time of a cell depends on data patterns stored in nearby cells → need to find the worst data pattern to find worst-case retention time.
DPD: Implications on Profiling Mechanisms

- Any retention time profiling mechanism must handle data pattern dependence of retention time
- Intuitive approach: Identify the data pattern that induces the worst-case retention time for a particular cell or device

Problem 1: Very hard to know at the memory controller which bits actually interfere with each other due to
  - Opaque mapping of addresses to physical DRAM geometry → logically consecutive bits may not be physically consecutive
  - Remapping of faulty bitlines/wordlines to redundant ones internally within DRAM

Problem 2: Worst-case coupling noise is affected by non-obvious second order bitline coupling effects
Two Challenges to Retention Time Profiling

Challenge 2: Variable Retention Time (VRT)
- Retention time of a DRAM cell changes randomly over time
  - a cell alternates between multiple retention time states

- Leakage current of a cell changes sporadically due to a charge trap in the gate oxide of the DRAM cell access transistor
- When the trap becomes occupied, charge leaks more readily from the transistor’s drain, leading to a short retention time
  - Called Trap-Assisted Gate-Induced Drain Leakage
- This process appears to be a random process [Kim+ IEEE TED’11]
  - Worst-case retention time depends on a random process
    → need to find the worst case despite this
An Example VRT Cell

A cell from E 2Gb chip family
Variable Retention Time

Many failing cells jump from very high retention time to very low.

Most failing cells exhibit VRT.

Min ret time = Max ret time
Expected if no VRT

A 2Gb chip family

log10(Fraction of Cells)

0.0
-0.6
-1.2
-1.8
-2.4
-3.0
-3.6
-4.2
-4.8
-5.4
-6.0
VRT: Implications on Profiling Mechanisms

Problem 1: There does not seem to be a way of determining if a cell exhibits VRT without actually observing a cell exhibiting VRT

- VRT is a memoryless random process [Kim+ JJAP 2010]

Problem 2: VRT complicates retention time profiling by DRAM manufacturers

- Exposure to very high temperatures can induce VRT in cells that were not previously susceptible
  - can happen during soldering of DRAM chips
  - manufacturer’s retention time profile may not be accurate

One option for future work: Use ECC to continuously profile DRAM online while aggressively reducing refresh rate

- Need to keep ECC overhead in check
More on DRAM Retention Analysis

- Jamie Liu, Ben Jaiyen, Yoongu Kim, Chris Wilkerson, and Onur Mutlu, "An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms" Proceedings of the 40th International Symposium on Computer Architecture (ISCA), Tel-Aviv, Israel, June 2013. Slides (ppt) Slides (pdf) [Invited Retrospective at 50 Years of ISCA, 2023 (pdf)]


An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms

Jamie Liu*
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
jamiel@alumni.cmu.edu

Ben Jaiyen*
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
bjaiyen@alumni.cmu.edu

Yoongu Kim
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
yoonguk@ece.cmu.edu

Chris Wilkerson
Intel Corporation
2200 Mission College Blvd.
Santa Clara, CA 95054
chris.wilkerson@intel.com

Onur Mutlu
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
onur@cmu.edu
Finding DRAM Retention Failures
Finding DRAM Retention Failures

- How can we reliably find the retention time of all DRAM cells?

- Goals: so that we can
  - Make DRAM reliable and secure
  - Make techniques like RAIDR work
    → improve performance and energy
Mitigation of Retention Issues [SIGMETRICS’14]

- Samira Khan, Donghyuk Lee, Yoongu Kim, Alaa Alameldeen, Chris Wilkerson, and Onur Mutlu,

"The Efficacy of Error Mitigation Techniques for DRAM Retention Failures: A Comparative Experimental Study"

AVATAR: A Variable-Retention-Time (VRT) Aware Refresh for DRAM Systems

Moinuddin K. Qureshi†  Dae-Hyun Kim†  Samira Khan‡  Prashant J. Nair†  Onur Mutlu‡
†Georgia Institute of Technology
‡Carnegie Mellon University

{moin, dhkim, pnair6}@ece.gatech.edu
{samirakhan, onur}@cmu.edu
AVATAR

Insight: Avoid retention failures ➔ Upgrade row on ECC error

Observation: Rate of VRT >> Rate of soft error (50x-2500x)

AVATAR mitigates VRT by increasing refresh rate on error
AVATAR reduces refresh by 60%-70%, similar to multi-rate refresh but with VRT tolerance.

Retention Testing Once a Year
increase refresh reduction from 60% to 70%
AVATAR obtains 2/3rd the performance of NoRefresh. Higher benefits in higher density DRAM chips.
ENERGY DELAY PRODUCT REDUCTION

AVATAR reduces EDP.
Higher benefits in higher density DRAM chips.
Moinuddin Qureshi, Dae Hyun Kim, Samira Khan, Prashant Nair, and Onur Mutlu, "AVATAR: A Variable-Retention-Time (VRT) Aware Refresh for DRAM Systems"
Proceedings of the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Rio de Janeiro, Brazil, June 2015. [Slides (pptx) (pdf)]
Handling Data-Dependent Failures [DSN’16]

- Samira Khan, Donghyuk Lee, and Onur Mutlu,
"PARBOR: An Efficient System-Level Technique to Detect Data-Dependent Failures in DRAM"

[Slides (pptx) (pdf)]

PARBOR: An Efficient System-Level Technique to Detect Data-Dependent Failures in DRAM

Samira Khan*  Donghyuk Lee†‡  Onur Mutlu*†
*University of Virginia  †Carnegie Mellon University  ‡Nvidia  *ETH Zürich
Handling Data-Dependent Failures [MICRO’17]

- Samira Khan, Chris Wilkerson, Zhe Wang, Alaa R. Alameldeen, Donghyuk Lee, and Onur Mutlu,

"Detecting and Mitigating Data-Dependent DRAM Failures by Exploiting Current Memory Content"

Proceedings of the 50th International Symposium on Microarchitecture (MICRO), Boston, MA, USA, October 2017.

[Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)] [Poster (pptx) (pdf)]
Handling Both DPD and VRT [ISCA’17]

  [Slides (pptx) (pdf)]
  [Lightning Session Slides (pptx) (pdf)]

- First experimental analysis of (mobile) LPDDR4 chips
- Analyzes the complex tradeoff space of retention time profiling
- Idea: enable fast and robust profiling at higher refresh intervals & temperatures

The Reach Profiler (REAPER): Enabling the Mitigation of DRAM Retention Failures via Profiling at Aggressive Conditions

Minesh Patel§§ Jeremie S. Kim‡§ Onur Mutlu§§
§ETH Zürich ‡Carnegie Mellon University
Making Refresh More Efficient

Only a few cells require frequent refreshing

Fast-leaking

Hard to identify
1. Process, voltage, temperature
2. Variable retention time
3. Data pattern dependence

Slow-leaking

Goal: quickly and efficiently identify the error-prone cells
Experimental Error Characterization

• We study the data-retention error characteristics in 368 real LPDDR4 DRAM chips

1. Cells are more likely to fail at an increased (1) refresh interval; or (2) temperature

2. Profiling involves a complex tradeoff space: (1) speed; (2) coverage; and (3) false positives
Reach Profiling

- Faster
- More reliable
- False positives possible

operate here

profile here

refresh interval

temperature
Evaluating Reach Profiling

1. **2.5x faster** than the state-of-the-art baseline for 99% coverage and a 50% false positive rate
   - **Even faster** (>3.5x) with more false positives (>100%)

2. Enables operating at **longer refresh intervals** by reducing the overall profiling overhead
   - 16.3% **end-to-end performance** improvement
   - 36.4% **DRAM power** reduction
More on Reach Profiling [ISCA’17]

- Minesh Patel, Jeremie S. Kim, and Onur Mutlu, "The Reach Profiler (REAPER): Enabling the Mitigation of DRAM Retention Failures via Profiling at Aggressive Conditions"


[Slides (pptx) (pdf)]
[Lightning Session Slides (pptx) (pdf)]

- First experimental analysis of (mobile) LPDDR4 chips
- Analyzes the complex tradeoff space of retention time profiling
- Idea: enable fast and robust profiling at higher refresh intervals & temperatures

The Reach Profiler (REAPER): Enabling the Mitigation of DRAM Retention Failures via Profiling at Aggressive Conditions

Minesh Patel§§ Jeremie S. Kim‡§ Onur Mutlu§§
§ETH Zürich ‡Carnegie Mellon University
In-DRAM ECC Complicates Things [DSN'19]

- Minesh Patel, Jeremie S. Kim, Hasan Hassan, and Onur Mutlu, "Understanding and Modeling On-Die Error Correction in Modern DRAM: An Experimental Study Using Real Devices"


[Slides (pptx) (pdf)]
[Talk Video (26 minutes)]
[Full Talk Lecture (29 minutes)]
[Source Code for EINSim, the Error Inference Simulator]

Best paper award.

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 ‡Carnegie Mellon University

SAFARI
More on In-DRAM ECC [MICRO’20]

- Minesh Patel, Jeremie S. Kim, Taha Shahroodi, Hasan Hassan, and Onur Mutlu,
  "Bit-Exact ECC Recovery (BEER): Determining DRAM On-Die ECC Functions by Exploiting DRAM Data Retention Characteristics"
  [Slides (pptx) (pdf)]
  [Short Talk Slides (pptx) (pdf)]
  [Lightning Talk Slides (pptx) (pdf)]
  [Lecture Slides (pptx) (pdf)]
  [Talk Video (15 minutes)]
  [Short Talk Video (5.5 minutes)]
  [Lightning Talk Video (1.5 minutes)]
  [Lecture Video (52.5 minutes)]
  [BEER Source Code]
  Best paper award.

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‡‡
†ETH Zürich  ‡‡Carnegie Mellon University

SAFARI
Profiling In The Presence of ECC [MICRO’21]

- Minesh Patel, Geraldo F. de Oliveira Jr., and Onur Mutlu,
"HARP: Practically and Effectively Identifying Uncorrectable Errors in Memory Chips That Use On-Die Error-Correcting Codes"
Proceedings of the 54th International Symposium on Microarchitecture (MICRO), Virtual, October 2021.
- [Slides (pptx) (pdf)]
- [Short Talk Slides (pptx) (pdf)]
- [Lightning Talk Slides (pptx) (pdf)]
- [Talk Video (20 minutes)]
- [Lightning Talk Video (1.5 minutes)]
- [HARP Source Code (Officially Artifact Evaluated with All Badges)]

HARP: Practically and Effectively Identifying Uncorrectable Errors in Memory Chips That Use On-Die Error-Correcting Codes

Minesh Patel  
ETH Zürich

Geraldo F. Oliveira  
ETH Zürich

Onur Mutlu  
ETH Zürich
Profiling a Memory Chip with On-Die ECC

Unreliable Memory

Profiler

On-Die ECC

Data Store

Which bits are at risk of error?

On-die ECC changes how errors appear to the profiler

Goal: understand and address any challenges that on-die ECC introduces for error profiling
Challenges Introduced by On-Die ECC

1. Exponentially increases the total number of at-risk bits

2. Makes it **harder to identify** individual at-risk bits

3. **Interferes** with commonly-used data patterns for memory testing
Key Observation: Two Sources of Errors

1. Direct error
   - Due to errors in the memory chip

2. Indirect error
   - Artifact of the on-die ECC algorithm
   - Upper-bounded by the ECC algorithm
Key Observation: Two Sources of Errors

1. Direct error
   - Due to errors in the memory chip

2. Indirect error
   - Artifact of the on-die ECC algorithm

Key Idea: decouple profiling for direct and indirect errors

Upper-bounded by the ECC algorithm
Hybrid Active-Reactive Profiling (HARP)

1. **Active Profiling**
   - Quickly identifies all direct errors with existing profiling techniques using an on-die ECC bypass path

2. **Reactive Profiling**
   - Safely identifies indirect errors using secondary ECC at least as strong as on-die ECC
Hybrid Active-Reactive Profiling (HARP)

1. Active Profiling
   - Quickly identifies direct errors with existing profiling techniques using an on-die ECC bypass path.

2. Reactive Profiling
   - Safely identifies indirect errors using secondary ECC at least as strong as on-die ECC.

HARP reduces the problem of profiling with on-die ECC to profiling without on-die ECC.

Safely identifies indirect errors using secondary ECC at least as strong as on-die ECC.
Evaluations

1. HARP improves **coverage** and **performance** relative to two state-of-the-art baseline profiling algorithms
   - E.g., **20.6-62.1% faster** to achieve 99\textsuperscript{th}-percentile coverage for 2-5 raw-bit errors per on-die ECC word

2. HARP **outperforms** the best-performing baseline in a case study of mitigating data-retention errors
   - E.g., **3.7x faster** given a per-bit error probability of 0.75

We conclude that HARP **overcomes** all three profiling challenges
More on HARP [MICRO’21]

- Minesh Patel, Geraldo F. de Oliveira Jr., and Onur Mutlu,
  "HARP: Practically and Effectively Identifying Uncorrectable Errors in Memory Chips That Use On-Die Error-Correcting Codes"
  Proceedings of the 54th International Symposium on Microarchitecture (MICRO), Virtual, October 2021.
  [Slides (pptx) (pdf)]
  [Short Talk Slides (pptx) (pdf)]
  [Lightning Talk Slides (pptx) (pdf)]
  [Talk Video (20 minutes)]
  [Lightning Talk Video (1.5 minutes)]
  [HARP Source Code (Officially Artifact Evaluated with All Badges)]

---

HARP: Practically and Effectively Identifying Uncorrectable Errors in Memory Chips That Use On-Die Error-Correcting Codes

- Minesh Patel
  ETH Zürich

- Geraldo F. Oliveira
  ETH Zürich

- Onur Mutlu
  ETH Zürich
Recall: RAIDR: Mechanism

1. Profiling: Identify the retention time of all DRAM rows
   → can be done at design time or during operation

2. Binning: Store rows into bins by retention time
   → use Bloom Filters for efficient and scalable storage

3. Refreshing: Memory controller refreshes rows in different bins at different rates
   → check the bins to determine refresh rate of a row

2. Binning

- How to efficiently and scalably store rows into retention time bins?
- Use Hardware Bloom Filters [Bloom, CACM 1970]
Bloom Filter

- [Bloom, CACM 1970]
- Probabilistic data structure that compactly represents set membership (presence or absence of element in a set)

- Non-approximate set membership: Use 1 bit per element to indicate absence/presence of each element from an element space of N elements

- Approximate set membership: use a much smaller number of bits and indicate each element’s presence/absence with a subset of those bits
  - Some elements map to the bits other elements also map to

- Operations: 1) insert, 2) test, 3) remove all elements
Bloom Filter Operation Example

Example with 64-128ms bin:

0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0

Hash function 1  Hash function 2  Hash function 3

Insert Row 1

Bloom Filter Operation Example

Example with 64-128ms bin:

```
0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0
```

Hash function 1

Hash function 2

Hash function 3

Row 1 present? Yes
Bloom Filter Operation Example

Example with 64-128ms bin:

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
</table>

Hash function 1

Hash function 2

Hash function 3

Row 2 present? No
Bloom Filter Operation Example

Example with 64-128ms bin:

Hash function 1

Hash function 2

Hash function 3

Insert Row 4
Bloom Filter Operation Example

Example with 64-128ms bin:

```
0 0 1 0 1 1 0 0 0 1 0 0 0 1 1 0 0 0 1 0 0 1 0 1 0
```

- **Hash function 1**: 0
- **Hash function 2**: 1
- **Hash function 3**: 1

Row 5 present?
Yes (false positive)
Bloom Filters

Space/Time Trade-offs in Hash Coding with Allowable Errors

In such applications, it is envisaged that overall performance could be improved by using a smaller core resident hash area in conjunction with the new methods and, when necessary, by using some secondary and perhaps time-consuming test to “catch” the small fraction of errors associated with the new methods. An example is discussed which illustrates possible areas of application for the new methods.

Burton H. Bloom

In this paper trade-offs among certain computational factors in hash coding are analyzed. The paradigm problem considered is that of testing a series of messages one-by-one for membership in a given set of messages. Two new hash-coding methods are examined and compared with a particular conventional hash-coding method. The computational factors considered are the size of the hash area (space), the time required to identify a message as a nonmember of the given set (reject time), and an allowable error frequency.

Bloom Filters: Pros and Cons

■ **Advantages**
  + Enables *storage-efficient* representation of set membership
  + Insertion and testing for set membership (presence) are *fast*
  + **No false negatives**: If Bloom Filter says an element is not present in the set, the element must not have been inserted
  + Enables *tradeoffs* between *time* & *storage efficiency* & *false positive rate* (via sizing and hashing)

■ **Disadvantages**
  -- **False positives**: An element may be deemed to be present in the set by the Bloom Filter but it may never have been inserted

  Not the right data structure when you cannot tolerate false positives

Benefits of Bloom Filters as Refresh Rate Bins

- **False positives:** a row may be declared present in the Bloom filter even if it was never inserted
  - **Not a problem:** Refresh some rows more frequently than needed

- **No false negatives:** rows are never refreshed less frequently than needed (no correctness problems)

- **Scalable:** a Bloom filter never overflows (unlike a fixed-size table)

- **Efficient:** No need to store info on a per-row basis; simple hardware → 1.25 KB for 2 filters for 32 GB DRAM system
Use of Bloom Filters in Hardware

- Useful when you can tolerate false positives in set membership tests

- See the following recent examples for clear descriptions of how Bloom Filters are used
3. Refreshing (RAIDR Refresh Controller)

Choose a refresh candidate row

Determine which bin the row is in

Determine if refreshing is needed
3. Refreshing (RAIDR Refresh Controller)

Memory controller chooses each row as a refresh candidate every 64ms

Row in 64-128ms bin?  \(\rightarrow\)  Row in 128-256ms bin?
(First Bloom filter: 256B)  (Second Bloom filter: 1KB)

- Refresh the row
- Every other 64ms window, refresh the row
- Every 4th 64ms window, refresh the row

RAIDR: Baseline Design

Refresh control is in DRAM in today’s auto-refresh systems.

RAIDR can be implemented in either the controller or DRAM.
Overhead of RAIDR in DRAM controller:
1.25 KB Bloom Filters, 3 counters, additional commands issued for per-row refresh (all accounted for in evaluations)
Overhead of RAIDR in DRAM chip:
Per-chip overhead: 20B Bloom Filters, 1 counter (4 Gbit chip)
Total overhead: 1.25KB Bloom Filters, 64 counters (32 GB DRAM)
RAIDR: Results and Takeaways

- System: 32GB DRAM, 8-core; SPEC, TPC-C, TPC-H workloads
- RAIDR hardware cost: 1.25 kB (2 Bloom filters)
- Refresh reduction: 74.6%
- Dynamic DRAM energy reduction: 16%
- Idle DRAM power reduction: 20%
- Performance improvement: 9%
- Benefits increase as DRAM scales in density
DRAM Refresh: More Questions

- What else can you do to reduce the impact of refresh?
- What else can you do if you know the retention times of rows?
- How can you accurately measure the retention time of DRAM rows?

Recommended reading:
DRAM Leakage in ISCA-50
25-Year Retrospective Issue
RAIDR: Retention-Aware Intelligent DRAM Refresh [ISCA’12]

- Jamie Liu, Ben Jaiyen, Richard Veras, and Onur Mutlu, "RAIDR: Retention-Aware Intelligent DRAM Refresh" Proceedings of the 39th International Symposium on Computer Architecture (ISCA), Portland, OR, June 2012. Slides (pdf) [Invited Retrospective at 50 Years of ISCA, 2023 (pdf)]

Analysis of Data Retention Failures [ISCA’13]

- Jamie Liu, Ben Jaiyen, Yoongu Kim, Chris Wilkerson, and Onur Mutlu, "An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms" Proceedings of the 40th International Symposium on Computer Architecture (ISCA), Tel-Aviv, Israel, June 2013. Slides (ppt) Slides (pdf) [Invited Retrospective at 50 Years of ISCA, 2023 (pdf)]


An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms

Jamie Liu*
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
jamiel@alumni.cmu.edu

Ben Jaiyen*
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
bjaiyen@alumni.cmu.edu

Yoongu Kim
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
yoonguk@ece.cmu.edu

Chris Wilkerson
Intel Corporation
2200 Mission College Blvd.
Santa Clara, CA 95054
chris.wilkerson@intel.com

Onur Mutlu
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
onur@cmu.edu
First RowHammer Analysis [ISCA’14]

- Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu,

"Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors"

[Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)] [Source Code and Data] [Lecture Video (1 hr 49 mins), 25 September 2020]

One of the 7 papers of 2012-2017 selected as Top Picks in Hardware and Embedded Security for IEEE TCAD (link).

Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors

Yoongu Kim¹ Ross Daly* Jeremie Kim¹ Chris Fallin* Ji Hye Lee¹ Donghyuk Lee¹ Chris Wilkerson² Konrad Lai Onur Mutlu¹

¹Carnegie Mellon University ²Intel Labs
RAIDR Retrospective [ISCA 2012]

Retrospective: RAIDR: Retention-Aware Intelligent DRAM Refresh

Our Story ETH Zurich

Abstract—Dynamic Random Access Memory (DRAM) is the dominant memory technology used to build high-performance systems. In the need to refresh memory cells to keep stored data intact, DRAM refresh consumes about 40% of the total power. It is also an important limiting factor as a performance bottleneck. Efforts have been made to improve performance at the DRAM cell level, but the refreshing process is a critical problem from a modern computing perspective: demonstrating a method to improve the refresh efficiency through macrocell refresh techniques is expected to be manufactured in the future. In pursuit of this problem, we proposed a DRAM refresh architecture called RAIDR (Retention-Aware Intelligent DRAM Refresh). RAIDR improves the refresh efficiency by utilizing the natural lifetime of DRAM cells to avoid unnecessary refresh operations. We identified two major issues that make refreshing very challenging and proposed a solution to improve refresh efficiency. We implemented RAIDR in a real-life system and demonstrated its effectiveness.

I. BACKGROUND, APPROACH & MINDSET

At the time we began our focus on solving the DRAM refresh problem (i.e., data retention) a significant challenge in late 2007, our group, RAIDR, had already been working on various controller and memory technology scaling issues, motivated by many challenges in memory systems. In particular, the DRAM technology [1] had been scaling as described in [2]. Our initial work on memory, initiated during our time at IBM, continued at ETH Zurich from 2006 and onward. From there, we had gained knowledge on memory technology scaling issues, and on the complexity of DRAM refresh. RAIDR is an attempt to develop new methods to improve refresh efficiency by reducing the number of refresh operations and other unnecessary refresh operations. RAIDR, developed later works provided improved metrics and rarefied information on what the future may bring on the DRAM refresh problem (more generally in memory technology scaling).

III. BUILDING ON RAIDR AND MAKING IT WORK

We believe RAIDR enabled a refreshing approach to DRAM refresh. Its high refresh efficiency could be the first step towards a memory technology that doesn’t require refresh. RAIDR is an attempt to develop new methods to improve refresh efficiency by reducing the number of refresh operations and other unnecessary refresh operations. We first presented RAIDR in a real-life system setting. To this end, we developed an FPGA-based flexible DRAM testing infrastructure that enabled us to accurately test data retention times of DRAM in real DRAM chips. This system was then used to test DRAM chips in a real-life system setting, including dynamic refresh and near-endurance refresh. Our approach demonstrated that RAIDR is an effective solution for improving refresh efficiency.

References


https://people.inf.ethz.ch/omutlu/pub/RAIDR_50YearsOfISCA-Retrospective_isca23.pdf
Retention Analysis Retrospective [ISCA 2013]

Retention Analysis: An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms

Omar Mutlu
ETH Zürich

Abstract—DRAM is the prevalent main memory technology used in current computers. However, the endurance of DRAM cells is limited, and a refresh is needed to repeatedly update data in them. Unfortunately, DRAM cells will lose stored data over time due to physical effects such as a) electronic charges in the storage nodes, b) variations in the storage nodes, and c) changes in the dielectric medium. To prevent data loss, refresh is required, which is done periodically. In this work, we present an empirical study of the data retention behavior of a large number of modern non-DRAM devices. We show that, contrary to what is commonly believed, DRAM devices do not exhibit an exponential decay of the retention time. Instead, we find that the retention time distribution is skewed to the right, and there are significant variations in the retention times across different devices and within the same device. We also show that the retention time distribution is highly correlated with the device characteristics, including the retention time distribution, which is typically a lognormal distribution. Our results suggest that the retention time distribution is more predictable than previously believed, and that it can be used to improve the accuracy of refresh mechanisms.

Our ISCA 2013 paper is a product of this goal and effort. Our work was generously supported especially by the Samsung DRAM Design Team and Intel Memory Architecture Labs, both technically advanced organizations. We also received support from Intel (especially Christine Winterstein, who is a co-author), we built our FPGA-based DDR3 circuitry, and we are grateful to all of them for their contributions.

1. INTRODUCTION AND LATER WORKS

Many prior works (e.g., [1, 2]) have focused on the problem of DRAM retention, and have observed that retention time is a critical factor in determining the quality of DRAM devices. In this work, we present an empirical study that provides a more detailed understanding of the factors that affect DRAM retention time. We show that retention time varies significantly across different devices and within the same device. We also show that the retention time distribution is highly correlated with the device characteristics, including the retention time distribution, which is typically a lognormal distribution. Our results suggest that the retention time distribution is more predictable than previously believed, and that it can be used to improve the accuracy of refresh mechanisms.

Further work is needed to investigate the impact of these findings on the design of DRAM refresh mechanisms. This is an area of ongoing research, and we look forward to future work in this area.

2. CONTRIBUTIONS AND IMPACT

Our ISCA 2013 paper is a product of this goal and effort. Our work was generously supported especially by the Samsung DRAM Design Team and Intel Memory Architecture Labs, both technically advanced organizations. We also received support from Intel (especially Christine Winterstein, who is a co-author), we built our FPGA-based DDR3 circuitry, and we are grateful to all of them for their contributions.

Our ISCA 2013 paper was a nice example of how a simple experimental approach can lead to significant insights about data retention in DRAM devices. We hope that it will encourage others to explore this area further, and that it will ultimately lead to better understanding and improved performance of DRAM devices.
RowHammer Retrospective [ISCA 2014]

Retrospective: Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors

Omer Muthu
ETH Zurich

Abstract—Our ISCA 2014 paper [1] provided the first scientific explanation of the RowHammer phenomenon. We named it RowHammer and explained how it worked, the implications it had on system security, and the threats it posed to existing DRAM memory models. Our work was groundbreaking because we showed that a simple, non-intrusive research tool, RowHammer, could be used to investigate the security and reliability of computer systems. We also demonstrated that RowHammer could be used to manipulate memory in ways that were previously unknown.

1. BACKGROUND AND CONTEXTS

Our stumbling on the RowHammer problem and creation of a technical write-up were the result of a confluence of factors. First, my group was working on DRAM technology scaling issues since 2005. We were very technical in failure mechanisms that appear or women due to aggressive technology scaling. To study such effects, we used numerical models to simulate the behavior of the DRAM cell and its components under different conditions. Second, our work identified a novel failure mechanism that we called RowHammer. Third, we noticed a similarity with previous research that had reported similar effects. We decided to investigate this further, and our findings led to the publication of our ISCA 2014 paper.

2. MAJOR CONTRIBUTIONS AND IMPACT

The major contributions of our paper were as follows:

- **RowHammer Phenomenon:** We provided a comprehensive description of the RowHammer phenomenon, including its causes, implications, and potential solutions.
- **Security Implications:** We demonstrated that RowHammer could be used to manipulate memory in ways that were previously unknown, and that this poses significant security risks.
- **Reliability Implications:** We showed how RowHammer could be used to exacerbate existing memory reliability issues, such as bit flips and data corruption.
- **Proposed Solutions:** We proposed several solutions to mitigate the impact of RowHammer, including hardware and software countermeasures.

Our work has had a significant impact on the field, leading to further research and developments in the area of DRAM technology and memory security. The RowHammer phenomenon has also been studied extensively in subsequent years, with numerous papers and publications exploring its implications and potential solutions.
Recommended Reading

- Jamie Liu, Ben Jaiyen, Yoongu Kim, Chris Wilkerson, and Onur Mutlu, "An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms" Proceedings of the 40th International Symposium on Computer Architecture (ISCA), Tel-Aviv, Israel, June 2013. Slides (ppt) Slides (pdf) [Invited Retrospective at 50 Years of ISCA, 2023 (pdf)]


An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms

Jamie Liu*
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
jamiel@alumni.cmu.edu

Ben Jaiyen*
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
bjaiyen@alumni.cmu.edu

Yoongu Kim
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
yoonguk@ece.cmu.edu

Chris Wilkerson
Intel Corporation
2200 Mission College Blvd.
Santa Clara, CA 95054
chris.wilkerson@intel.com

Onur Mutlu
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
onur@cmu.edu
DRAM Refresh: Summary and Conclusions

- DRAM refresh is a critical challenge
  - in scaling DRAM technology efficiently to higher capacities

- Several promising solution directions
  - Eliminate unnecessary refreshes [Liu+ ISCA’12]
  - Reduce refresh rate w/ online profiling and detect/correct any errors [Khan+ SIGMETRICS’14, Qureshi+ DSN’15, Patel+ ISCA’17]
  - Parallelize refreshes with accesses [Chang+ HPCA’14; Yaglikci+ MICRO’22]

- Examined properties of retention time behavior [Liu+ ISCA’13]
  - Enable realistic VRT-Aware refresh techniques [Qureshi+ DSN’15]

- Many avenues for overcoming DRAM refresh challenges
  - Handling DPD/VRT phenomena
  - Enabling online retention time profiling and error mitigation
  - Exploiting application behavior
Refresh-Access Parallelization

- Kevin Chang, Donghyuk Lee, Zeshan Chishti, Alaa Alameldeen, Chris Wilkerson, Yoongu Kim, and Onur Mutlu,

"Improving DRAM Performance by Parallelizing Refreshes with Accesses"


[Summary] [Slides (pptx) (pdf)]
HiRA: Hidden Row Activation for Reducing Refresh Latency of Off-the-Shelf DRAM Chips

A. Giray Yağlıkçı, Ataberk Olgun, Minesh Patel, Haocong Luo, Hasan Hassan, Lois Orosa, Oğuz Ergin, and Onur Mutlu,

Proceedings of the 55th International Symposium on Microarchitecture (MICRO), Chicago, IL, USA, October 2022.

[Slides (pptx) (pdf)]
[Longer Lecture Slides (pptx) (pdf)]
[Lecture Video (36 minutes)]
[arXiv version]
HiRA: Hidden Row Activation for Reducing Refresh Latency of Off-the-Shelf DRAM Chips

Abdullah Giray Yağılkıcı
Ataberk Olgun  Minesh Patel  Haocong Luo  Hasan Hassan
Lois Orosa  Oğuz Ergin  Onur Mutlu

SAFARI

ETH zürich  CESGA  TOBB ETÜ
Two Main Types of DRAM Refresh

1. **Periodic Refresh**: Periodically restores the charge in DRAM cells as they leak over time.

2. **RowHammer**: Repeatedly accessing a DRAM row can cause bit flips in other physically nearby rows.

   - **Preventive Refresh**: Mitigates RowHammer by refreshing physically nearby rows of a repeatedly accessed row.
Periodic Refresh with Increasing DRAM Chip Density

A larger capacity chip has more rows to be refreshed

A smaller cell stores less charge

More periodic refresh operations incur larger performance overhead as DRAM chip density increases
RowHammer and Preventive Refresh with Increasing DRAM Chip Density

RowHammer vulnerability worsens as DRAM chip density increases

Preventive refresh operations need to be performed more aggressively as DRAM chip density increases
Goal and Key Idea

Reduce the **performance overhead** of DRAM Refresh (both **periodic** and **preventive**)

Hide refresh latency by **refreshing** a DRAM row **concurrently with activating** another row in a **different subarray** of the **same bank**
HiRA: Hidden Row Activation – Key Insight

Activating two rows in **quick succession** that are in **different subarrays** in the **same bank** can **refresh one row** concurrently with **activating the other row**.

![Diagram](image_url)

- **Subarray X**
  - **Row A**: Refreshes Row A concurrently with
  - **ACT**
  - **Subarray Y**
  - **Row B**: Activating Row B

**DRAM Bank**
HiRA: Hidden Row Activation

*Refresh RowA concurrently with Activating RowB*

Without HiRA:
- **RowA**'s refresh
- Precharge
- **RowB**'s activation

With HiRA:
- **HiRA**
- **PRE**
- **ACT** RowA, **ACT** RowB
- **RD**

*Saved time using HiRA*
HiRA Operation

HiRA refreshes RowA concurrently with activating RowB by issuing **ACT-PRE-ACT** commands in quick succession.

**SAFARI**
DRAM Testing Infrastructure

FPGA-based SoftMC (Xilinx Virtex UltraScale+ XCU200)

Xilinx Alveo U200 FPGA Board (programmed with SoftMC*)

DRAM Module with Heaters

PCIe Host Interface

MaxWell FT200 Temperature Controller

Fine-grained control over **DRAM commands**, timing parameters (**±1.5ns**), and **temperature** (**±0.1°C**)
HiRA in Off-the-Shelf DRAM Chips: Key Result 1

- HiRA works in **56 off-the-shelf DRAM chips** from **SK Hynix**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A0</td>
<td>G.SKILL</td>
<td>DWCW (Partial Marking)* F4-2400C17S-8GNT [39]</td>
<td>2400</td>
<td>42-20</td>
<td>4Gb</td>
<td>B</td>
<td>x8</td>
<td>24.8%</td>
<td>1.75</td>
</tr>
<tr>
<td>A1</td>
<td></td>
<td></td>
<td></td>
<td>24.9%</td>
<td></td>
<td></td>
<td></td>
<td>25.0%</td>
<td>1.72</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>24.9%</td>
<td></td>
<td></td>
<td></td>
<td>25.5%</td>
<td>1.74</td>
</tr>
<tr>
<td>B0</td>
<td>Kingston</td>
<td>H5AN8G8NDJR-XNC KSM32RD8/16HDR [87]</td>
<td>2400</td>
<td>48-20</td>
<td>4Gb</td>
<td>D</td>
<td>x8</td>
<td>25.1%</td>
<td>1.71</td>
</tr>
<tr>
<td>B1</td>
<td></td>
<td></td>
<td></td>
<td>25.0%</td>
<td></td>
<td></td>
<td></td>
<td>32.6%</td>
<td>1.74</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>25.0%</td>
<td></td>
<td></td>
<td></td>
<td>36.8%</td>
<td>2.34</td>
</tr>
<tr>
<td>C0</td>
<td>SK Hynix</td>
<td>H5ANAG8NAJR-XN HMAA4GU6AJR8N-XN [109]</td>
<td>2400</td>
<td>51-20</td>
<td>4Gb</td>
<td>F</td>
<td>x8</td>
<td>25.3%</td>
<td>1.47</td>
</tr>
<tr>
<td>C1</td>
<td></td>
<td></td>
<td></td>
<td>25.3%</td>
<td></td>
<td></td>
<td></td>
<td>35.3%</td>
<td>2.23</td>
</tr>
<tr>
<td>C2</td>
<td></td>
<td></td>
<td></td>
<td>25.3%</td>
<td></td>
<td></td>
<td></td>
<td>39.5%</td>
<td>1.09</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>29.2%</td>
<td></td>
<td></td>
<td></td>
<td>38.4%</td>
<td>2.27</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>26.5%</td>
<td></td>
<td></td>
<td></td>
<td>36.1%</td>
<td>1.49</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>42.3%</td>
<td>2.58</td>
</tr>
</tbody>
</table>

* The chip identifier is partially removed on these modules. We infer the chip manufacturer and die revision based on the remaining part of the chip identifier.

- HiRA performs a given row’s **refresh concurrently with activating** any of the **32% of the rows** in the same bank
HiRA in Off-the-Shelf DRAM Chips: Key Result 2

- HiRA works in **56 off-the-shelf DRAM chips** from **SK Hynix**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A0, A1</td>
<td>G.SKILL</td>
<td>DWCW (Partial Marking)*</td>
<td>2400</td>
<td>42-20</td>
<td>4Gb</td>
<td>B</td>
<td>x8</td>
<td>24.8%</td>
<td>25.0%</td>
<td>25.5%</td>
<td>1.75</td>
<td>1.90</td>
<td>2.52</td>
<td></td>
</tr>
<tr>
<td>B0, B1</td>
<td>Kingston</td>
<td>H5AN8G8NDJR-XNC KSM32RD8/16HDR [87]</td>
<td>2400</td>
<td>48-20</td>
<td>4Gb</td>
<td>D</td>
<td>x8</td>
<td>25.1%</td>
<td>32.6%</td>
<td>36.8%</td>
<td>1.71</td>
<td>1.89</td>
<td>2.34</td>
<td></td>
</tr>
<tr>
<td>C0, C1, C2</td>
<td>SK Hynix</td>
<td>H5ANAG8NAJR-XN HMAA4GU6AJR8N-XN [109]</td>
<td>2400</td>
<td>51-20</td>
<td>4Gb</td>
<td>F</td>
<td>x8</td>
<td>25.3%</td>
<td>35.3%</td>
<td>39.5%</td>
<td>1.47</td>
<td>1.89</td>
<td>2.23</td>
<td></td>
</tr>
</tbody>
</table>

* The chip identifier is partially removed on these modules. We infer the chip manufacturer and die revision based on the remaining part of the chip identifier.

- **51.4% reduction** in the time spent for refresh operations

**HiRA effectively reduces the time spent for refresh operations in off-the-shelf DRAM chips**
HiRA in Off-the-Shelf DRAM

HiRA: Hidden Row Activation for Reducing Refresh Latency of Off-the-Shelf DRAM Chips

A. Giray Yaşlıkçı¹  Ataberk Olgun¹  Minesh Patel¹  Haocong Luo¹  Hasan Hassan¹
Lois Orosa¹,³  Öğuz Ergin²  Onur Mutlu¹

¹ETH Zürich  ²TOBB University of Economics and Technology  ³Galicia Supercomputing Center (CESGA)

As DRAM density increases with technology node scaling, the performance overhead of refresh also increases due to three major reasons. First, as the DRAM chip density increases, more DRAM rows need to be periodically refreshed in a DRAM chip [55, 57–61]. Second, as DRAM technology node scales down, DRAM cells become smaller and thus can store less amount of charge, requiring them to be refreshed more frequently [10, 20, 67, 102, 103, 118, 122–124]. Third, with increasing DRAM density, DRAM cells are placed closer to each other, exacerbating charge leakage via a disturbance error mechanism called RowHammer [79, 84, 119, 120, 133, 134, 167, 180, 183], and thus requiring additional refresh operations (called preventive refreshes) to avoid data corruption due to RowHam-

HiRA-MC: HiRA Memory Controller

• **Goal**: Leverage HiRA’s parallelism as much as possible

• **Key Insight**: A time slack is needed to find a row activation and a refresh to perform HiRA

RowA and RowZ are in two electrically disconnected subarrays

SAFARI
HiRA-MC: HiRA Memory Controller

Generates each periodic refresh and RowHammer-preventive refresh with a deadline

1. Buffers each refresh request and performs the refresh request until the deadline

2. Finds if it can refresh a DRAM row concurrently with a DRAM access or another refresh
HiRA-MC: HiRA Memory

HiRA: Hidden Row Activation
for Reducing Refresh Latency of Off-the-Shelf DRAM Chips

A. Giray Yağlıkçı¹ Ataberk Olgun¹ Minesh Patel¹ Haocng Luo¹ Hasan Hassan¹
Lois Orosa¹,² Oğuz Ergin² Onur Mutlu¹
¹ETH Zürich ²TOBB University of Economics and Technology ³Galicia Supercomputing Center (CESGA)

DRAM is the building block of modern main memory systems. DRAM cells must be periodically refreshed to prevent data loss. Refresh operations degrade system performance by interfering with memory accesses. As DRAM chip density increases with technology node scaling, refresh operations also increase because: 1) the number of DRAM rows in a chip increases; and 2) DRAM cells need additional refresh operations to mitigate bit failures caused by RowHammer, a failure mechanism that becomes worse with technology node scaling. Thus, it is critical to enable refresh operations at low performance overhead. To this end, we propose a new operation, Hidden Row Activation (HiRA), and the HiRA Memory Controller (HiRA-MC) to perform HiRA operations.

As DRAM density increases with technology node scaling, the performance overhead of refresh also increases due to three major reasons. First, as the DRAM chip density increases, more DRAM rows need to be periodically refreshed in a DRAM chip [55, 57–61]. Second, as DRAM technology node scales down, DRAM cells become smaller and thus can store less amount of charge, requiring them to be refreshed more frequently [10, 20, 67, 102, 103, 118, 122–124]. Third, with increasing DRAM density, DRAM cells are placed closer to each other, exacerbating charge leakage via a disturbance error mechanism called RowHammer [79, 84, 119, 120, 133, 134, 167, 180, 183], and thus requiring additional refresh operations (called preventive refreshes) to avoid data corruption due to RowHam-

Performance Evaluation

• Cycle-level simulations using **Ramulator** [Kim+, CAL 2015]

**System Configuration:**

<table>
<thead>
<tr>
<th>Component</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>3.2 GHz, 8 core, 4-wide issue, 128-entry instr. window</td>
</tr>
<tr>
<td>Last-Level Cache</td>
<td>64-byte cache line, 8-way set-associative, 8 MB</td>
</tr>
<tr>
<td>Memory Scheduler</td>
<td>FR-FCFS</td>
</tr>
<tr>
<td>Address Mapping</td>
<td>Minimalistic Open Pages</td>
</tr>
<tr>
<td>Main Memory</td>
<td>DDR4, 4 bank group, 4 banks per bank group (16 banks per rank)</td>
</tr>
<tr>
<td>Timing Parameters</td>
<td>(t_1 = t_2 = 3) ns, (t_{RC} = 46.25) ns, (t_{FAW} = 16) ns</td>
</tr>
</tbody>
</table>

• **Workloads**: 125 different 8-core multiprogrammed workloads from the SPEC2006 benchmark suite

• **DRAM Chip Capacity**: \(\{2, 4, 8, 16, 32, 64, 128\}\) Gb

• **RowHammer Threshold**: \(\{1024, 512, 256, 128, 64\}\) activations
  
  *The minimum number of row activations needed to induce the first RowHammer bit flip*
HiRA for Periodic Refreshes

- **No-Refresh**: No periodic refresh is performed (Ideal case)
- **Baseline**: Auto-Refresh (using conventional REF commands)

Periodic refreshes cause **significant (26%) performance overhead**

HiRA improves system performance by **12.6%** over the baseline
HiRA for Preventive Refreshes

- **No Defense**: No RowHammer mitigation employed (i.e., no preventive refresh)
- **PARA** [Kim+, ISCA’14]: the RowHammer defense with the **lowest hardware overhead**

![Bar chart showing weighted speedup for different RowHammer thresholds (No-Defense, PARA, HiRA)]

- **No-Defense**:
  - No RowHammer mitigation employed (i.e., no preventive refresh)
- **PARA**:
  - The RowHammer defense with the lowest hardware overhead

**Key Observations**:
- **PARA** significantly reduces (by 96%) system performance
- **HiRA** improves system performance by 3.7x over PARA
More in the Full Paper

• **Real DRAM Chip** Experiments
  - Verification of **HiRA’s functionality**
  - **Variation** in HiRA’s characteristics **across banks**

• Sensitivity to
  - length of **time slack** for refreshes
  - number of **channels**
  - number of **ranks**

• **Hardware Complexity Analysis**
  - Chip **area cost of 0.0023%** of a processor die per DRAM rank
  - **No additional latency** overhead

• **Experimental Methodology**
  - **Detailed algorithms** for each set of real chip experiments
  - Extensive **security analysis** for RowHammer-preventive refreshes

• Detailed Algorithm of **Finding Concurrent Refreshes**
HiRA: Hidden Row Activation for Reducing Refresh Latency of Off-the-Shelf DRAM Chips

A. Giray Yağlıkçı¹ Ataberk Olgun¹ Minesh Patel¹ Haocong Luo¹ Hasan Hassan¹
Lois Orosa¹,³ Oğuz Ergin² Onur Mutlu¹
¹ETH Zürich ²TOBB University of Economics and Technology ³Galicia Supercomputing Center (CESGA)

DRAM is the building block of modern main memory systems. DRAM cells must be periodically refreshed to prevent data loss. Refresh operations degrade system performance by interfering with memory accesses. As DRAM chip density increases with technology node scaling, refresh operations also increase because: 1) the number of DRAM rows in a chip increases; and 2) DRAM cells need additional refresh operations to mitigate bit failures caused by RowHammer, a failure mechanism that becomes worse with technology node scaling. Thus, it is critical to enable refresh operations at low performance overhead. To this end, we propose a new operation, Hidden Row Activation (HiRA), and the HiRA Memory Controller (HiRA-MC) to perform HiRA operations.

As DRAM density increases with technology node scaling, the performance overhead of refresh also increases due to three major reasons. First, as the DRAM chip density increases, more DRAM rows need to be periodically refreshed in a DRAM chip [55, 57–61]. Second, as DRAM technology node scales down, DRAM cells become smaller and thus can store less amount of charge, requiring them to be refreshed more frequently [10, 20, 67, 102, 103, 118, 122–124]. Third, with increasing DRAM density, DRAM cells are placed closer to each other, exacerbating charge leakage via a disturbance error mechanism called RowHammer [79, 84, 119, 120, 133, 134, 167, 180, 183], and thus requiring additional refresh operations (called preventive refreshes) to avoid data corruption due to RowHam-

Conclusion

• **HiRA**: Hidden Row Activation – a new DRAM operation
  - First technique that refreshes a DRAM row concurrently with activating another row in the same bank in off-the-shelf DRAM chips
  - **Real DRAM chip** experiments:
    • HiRA works on **56 real off-the-shelf DRAM chips**
    • **51.4% reduction** in the time spent for refresh operations

• **HiRA-MC**: HiRA Memory Controller – a new mechanism
  - Leverages HiRA to perform refresh requests concurrently with DRAM accesses and other refresh requests
  - **HiRA-MC provides**:
    • **12.6% speedup** by hiding periodic refresh latency
    • **3.7x speedup** by hiding RowHammer-preventive refresh latency
HiRA: Hidden Row Activation
for Reducing Refresh Latency of Off-the-Shelf DRAM Chips

Abdullah Giray Yağlıkçı
Ataberk Olgun  Minesh Patel  Haocong Luo  Hasan Hassan
Lois Orosa  Oğuz Ergin  Onur Mutlu

SAFARI
A. Giray Yaglıkcı, Ataberk Olgun, Minesh Patel, Haocong Luo, Hasan Hassan, Lois Orosa, Oguz Ergin, and Onur Mutlu,
Proceedings of the 55th International Symposium on Microarchitecture (MICRO), Chicago, IL, USA, October 2022.
[Slides (pptx) (pdf)]
[Longer Lecture Slides (pptx) (pdf)]
[Lecture Video (36 minutes)]
[arXiv version]

**HiRA: Hidden Row Activation for Reducing Refresh Latency of Off-the-Shelf DRAM Chips**

A. Giray Yağlıkcı\(^1\)  Ataberk Olgun\(^1,2\)  Minesh Patel\(^1\)  Haocong Luo\(^1\)  Hasan Hassan\(^1\)
Lois Orosa\(^1,3\)  Oğuz Ergin\(^2\)  Onur Mutlu\(^1\)

\(^1\)ETH Zürich  \(^2\)TOBB University of Economics and Technology  \(^3\)Galicia Supercomputing Center (CESGA)
Industry Is Writing Papers About It, Too

DRAM Process Scaling Challenges

- **Refresh**
  - Difficult to build high-aspect ratio cell capacitors decreasing cell capacitance
  - Leakage current of cell access transistors increasing

- **tWR**
  - Contact resistance between the cell capacitor and access transistor increasing
  - On-current of the cell access transistor decreasing
  - Bit-line resistance increasing

- **VRT**
  - Occurring more frequently with cell capacitance decreasing
Call for Intelligent Memory Controllers

DRAM Process Scaling Challenges

- Refresh
  - Difficult to build high-aspect ratio cell capacitors decreasing cell capacitance

THE MEMORY FORUM 2014

Co-Architecting Controllers and DRAM to Enhance DRAM Process Scaling

Uksong Kang, Hak-soo Yu, Churoo Park, *Hongzhong Zheng, **John Halbert, **Kuljit Bains, SeongJin Jang, and Joo Sun Choi

Samsung Electronics, Hwasung, Korea / *Samsung Electronics, San Jose / **Intel
Data Retention in Flash Memory
Foreshadowing: Limits of Charge Memory

- **Difficult charge placement and control**
  - Flash: floating gate charge
  - DRAM: capacitor charge, transistor leakage

- **Data retention and reliable sensing become difficult as charge storage unit size reduces**
An unfortunate tale about Samsung's SSD 840 read performance degradation

An avalanche of reports emerged last September, when owners of the usually speedy Samsung SSD 840 and SSD 840 EVO detected the drives were no longer performing as they used to.

The issue has to do with older blocks of data: reading **old files** consistently slower than normal as slow as 30MB/s whereas **newly-written files** ones used in benchmarks, perform as fast as new – around 500 MB/s for the well-regarded SSD 840 EVO. The reason no one had noticed (we reviewed the drive back in September 2013) is that data has to be several weeks old to show the problem. Samsung promptly admitted the issue and proposed a fix.

Why is old data slower?

Retention loss!
Retention loss

Charge leakage over time

One dominant source of flash memory errors [DATE ‘12, ICCD ‘12]

Side effect: Longer read latency
NAND Flash Error Types

- Four types of errors [Cai+, DATE 2012]

- Caused by common flash operations
  - Read errors
  - Erase errors
  - Program (interference) errors

- Caused by flash cell losing charge over time
  - Retention errors
    - Whether an error happens depends on required retention time
    - Especially problematic in MLC flash because threshold voltage window to determine stored value is smaller
Flash Experimental Testing Platform

HAPS-52 Mother Board

Virtex-V FPGA (NAND Controller)

USB Daughter Board

USB Jack

Virtex-II Pro (USB controller)

1x-nm NAND Flash

NAND Daughter Board


Observations: Flash Error Analysis

- Raw bit error rate increases exponentially with P/E cycles
- **Retention errors are dominant** (>99% for 1-year retention time)
- Retention errors increase with retention time requirement

---

Cai et al., *Error Patterns in MLC NAND Flash Memory*, DATE 2012.
More on Flash Error Analysis


Error Patterns in MLC NAND Flash Memory: Measurement, Characterization, and Analysis

Yu Cai¹, Erich F. Haratsch², Onur Mutlu¹ and Ken Mai¹
¹Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA
²LSI Corporation, 1110 American Parkway NE, Allentown, PA
¹{yucai, onur, kenmai}@andrew.cmu.edu, ²erich.haratsch@lsi.com
Solution to Retention Errors

- Refresh periodically

- Change the period based on P/E cycle wearout
  - Refresh more often at higher P/E cycles

- Use a combination of in-place and remapping-based refresh

Flash Correct-and-Refresh [ICCD’12]

Yu Cai, Gulay Yalcin, Onur Mutlu, Erich F. Haratsch, Adrian Cristal, Osman Unsal, and Ken Mai,
"Flash Correct-and-Refresh: Retention-Aware Error Management for Increased Flash Memory Lifetime"
Proceedings of the 30th IEEE International Conference on Computer Design (ICCD), Montreal, Quebec, Canada, September 2012. Slides (ppt)(pdf)

Flash Correct-and-Refresh: Retention-Aware Error Management for Increased Flash Memory Lifetime

Yu Cai¹, Gulay Yalcin², Onur Mutlu¹, Erich F. Haratsch³, Adrian Cristal², Osman S. Unsal² and Ken Mai¹
¹DSSC, Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA
²Barcelona Supercomputing Center, C/Jordi Girona 29, Barcelona, Spain
³LSI Corporation, 1110 American Parkway NE, Allentown, PA
Yu Cai, Gulay Yalcin, Onur Mutlu, Erich F. Haratsch, Adrian Cristal, Osman Unsal, and Ken Mai,
"Error Analysis and Retention-Aware Error Management for NAND Flash Memory"
Flash Memory Data Retention Analysis


[Slides (pptx) (pdf)] [Poster (pdf)] Best paper session.

[Abstract]
[POMACS Journal Version (same content, different format)]
[Slides (pptx) (pdf)]
Error Characterization, Mitigation, and Recovery in Flash-Memory-Based Solid-State Drives

This paper reviews the most recent advances in solid-state drive (SSD) error characterization, mitigation, and data recovery techniques to improve both SSD’s reliability and lifetime.

By Yu Cai, Saugata Ghose, Erich F. Haratsch, Yixin Luo, and Onur Mutlu

https://arxiv.org/pdf/1706.08642
Yu Cai, Saugata Ghose, Erich F. Haratsch, Yixin Luo, and Onur Mutlu,
"Errors in Flash-Memory-Based Solid-State Drives: Analysis, Mitigation, and Recovery"
[Preliminary arxiv.org version]

Errors in Flash-Memory-Based Solid-State Drives: Analysis, Mitigation, and Recovery

YU CAI, SAUGATA GHOSE
Carnegie Mellon University

ERICH F. HARATSCH
Seagate Technology

YIXIN LUO
Carnegie Mellon University

ONUR MUTLU
ETH Zürich and Carnegie Mellon University
We Will Dig Deeper More
In This Course

“Good ideas are a dime a dozen”

“Making them work is oftentimes the real contribution”
Backup Slides
HiRA: Hidden Row Activation for Reducing Refresh Latency of Off-the-Shelf DRAM Chips

Backup Slides

Abdullah Giray Yağlıkçı
Ataberk Olgun  Minesh Patel  Haocong Luo  Hasan Hassan
Lois Orosa  Oğuz Ergin  Onur Mutlu

SAFARI

ETHzürich  CESGA  TOBB ETÜ
The RowHammer Vulnerability

Repeatedly **opening** (activating) and **closing** (precharging) a DRAM row in real DRAM chips causes **RowHammer bit flips** in nearby cells.
Activating a DRAM row refreshes the row and prevents RowHammer bit flips.
Mitigating RowHammer

Preventive Refresh

Activating potential victim rows mitigate RowHammer by refreshing them

ACT

Row 0
Row 1
Row 2
Row 3
Row 4

⚠️ Row 2 is being hammered

🔄 Refresh neighbor rows
RowHammer and Preventive Refresh

- **RowHammer**: Repeatedly accessing a DRAM row can cause bit flips in other physically nearby rows.
- **Preventive Refresh**: Refresh a DRAM row when a physically nearby row is activated based on activation counts or probabilistic processes.

Preventive refresh mitigates RowHammer bit flips.
HiRA: Hidden Row Activation

- **HiRA concurrently activates** two rows in a DRAM bank
  - **Challenge 1:** Only one row can be activated in a DRAM bank at a given time
  - **Solution 1:** HiRA violates timing constraints for concurrent row activations

- **HiRA issues two row activation (ACT) commands in quick succession**
  - **Challenge 2:** DRAM chips ignore the second activation before precharge
  - **Solution 2:** HiRA issues a precharge (PRE) command between two ACTs

- **HiRA activates two DRAM rows in the same bank**
  - **Challenge 3:** The two rows can override each other’s data via shared bitlines
  - **Solution 3:** HiRA uses rows from two electrically disconnected subarrays

HiRA violates DRAM timing constraints by issuing a sequence of ACT-PRE-ACT commands that target two rows in two electrically disconnected subarrays.
HiRA: Hidden Row Activation

**Refreshing RowA concurrently with Activating RowB**

The time saved using HiRA

**Without HiRA**
- RowA’s refresh
- Precharge
- RowB’s activation

**With HiRA**
- HiRA PRE
- RowA’s refresh
- RowB’s activation

**Reduction in the time spent for two refreshes**
HiRA Operation

HiRA refreshes RowA concurrently with activating RowB by issuing **ACT-PRE-ACT** commands in quick succession.
HiRA in Off-the-Shelf DRAM Chips: Key Results

- HiRA works in **56 off-the-shelf DRAM chips** from **SK Hynix**

- **51.4% reduction** in the time spent for refresh operations

- HiRA performs a given row’s **refresh concurrently with activating** any of the **32% of the rows** in the same bank

---

**HiRA effectively reduces the time spent for refresh operations in off-the-shelf DRAM chips**
HiRA Support in Off-the-Shelf DRAM Chips

- 56 off-the-shelf DDR4 DRAM chips support HiRA (from SK Hynix)
- HiRA Coverage of a given DRAM row:
  - Refresh a given DRAM row while activating other rows in the same bank
  - We sweep two timing parameters: $t_1$ and $t_2$

HiRA can refresh a DRAM row concurrently with 32% of any of the other DRAM rows in the same bank

HiRA Coverage across DRAM Rows

$t_1$ and $t_2$ can be as small as 3ns

$HiRA$ can refresh a DRAM row concurrently with 32% of any of the other DRAM rows in the same bank
HiRA’s Second Row Activation

- Does performing HiRA in between refresh the victim row?
  - If HiRA’s second row activation is performed, more activations are needed to induce RowHammer bit flips
  - If HiRA’s second row activation is ignored, RowHammer threshold should not change

![](image)

**a)** Absolute RowHammer Thresholds for tests with and without HiRA

**b)** RowHammer Thresholds normalized to tests without HiRA
Variation across DRAM Banks

- Coverage: Identical across banks
- The effect of second row activation
HiRA-MC: HiRA Memory Controller

- **Goal:** Leverage HiRA’s parallelism as much as possible
- **Periodic** and **preventive** refresh controllers generate each refresh request **with a deadline**
- **Refresh Table** buffers a refresh request until its **deadline**
- **Concurrent Refresh Finder** finds if HiRA can refresh a row
  - *Concurrently with a memory request*
  - *Concurrently with another refresh request*
The Concurrent Refresh Finder

**Case 1:** Executes when a precharge is issued (completes before the precharge completes)

**Case 2:** Periodically executes after every $t_{RC}$ (completes before $t_{RC}$)
HiRA-MC Example

- **Case 1:** Refresh – Access Parallelism
  - Memory Request Queue
  - Refresh Table
  - HiRA(Row SA:B Row:6, SA:A Row:0)
  - ACT SA:B Row:6, ACT SA:A Row:0
  - PRE
  - 6ns

- **Case 2:** Refresh – Refresh Parallelism
  - Memory Request Queue
  - Refresh Table
  - HiRA(Row SA:B Row:1, SA:C Row:2)
  - ACT SA:B Row:1, ACT SA:C Row:2
  - PRE
  - 6ns

HiRA-MC provides **refresh-access** and **refresh-refresh** parallelism
HiRA-MC Hardware Complexity

- We use CACTI with 22nm technology node

<table>
<thead>
<tr>
<th>HiRA-MC Component</th>
<th>Area (mm²)</th>
<th>Area (% of Chip Area)</th>
<th>Access Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Refresh Table</td>
<td>0.00031</td>
<td>&lt;0.0001%</td>
<td>0.07ns</td>
</tr>
<tr>
<td>RefPtr Table</td>
<td>0.00683</td>
<td>0.0017%</td>
<td>0.12ns</td>
</tr>
<tr>
<td>PR-FIFO</td>
<td>0.00029</td>
<td>&lt;0.0001%</td>
<td>0.07ns</td>
</tr>
<tr>
<td>Subarray Pairs Table</td>
<td>0.00180</td>
<td>0.0005%</td>
<td>0.09ns</td>
</tr>
<tr>
<td><strong>Overall</strong></td>
<td><strong>0.00923</strong></td>
<td><strong>0.0023%</strong></td>
<td><strong>6.31ns</strong></td>
</tr>
</tbody>
</table>

HiRA-MC consumes only **0.0023%** of CPU chip area per DRAM rank

HiRA-MC Overall Latency: **6.31ns**

Precharge latency: ~**14.5ns**

**HiRA-MC does not increase** memory access latency
Estimating Periodic Refresh Overhead

\[ t_{RFC} = 110 \times C_{chip}^{0.6} \]

Latency of a REF command \hspace{2cm} DRAM Chip Capacity


Nonblocking Memory Refresh

Kate Nguyen, Kehan Lyu, Xianze Meng
Department of Computer Science
Virginia Tech
Blacksburg, Virginia
katevy@vt.edu, kehan@vt.edu, xianze@vt.edu

Vilas Sridharan
RAS Architecture
Advanced Micro Devices, Inc
Boxborough, Massachusetts
vilas.sridharan@amd.com

Xun Jian
Department of Computer Science
Virginia Tech
Blacksburg, Virginia
xunj@vt.edu
Reducing Overall Latency of Two Refreshes

• Refreshing two rows using nominal timing parameters:

\[ t_{RAS}: 32\text{ns} \quad t_{RP}: 14.25\text{ns} \quad t_{RAS}: 32\text{ns} \]

\[ \text{ACT RowA} \quad \text{PRE} \quad \text{ACT RowB} \quad \text{PRE} \]

\[ t_1: 3\text{ns} \quad t_2: 3\text{ns} \quad t_{RAS}: 32\text{ns} \]

\[ \text{ACT RowA} \quad \text{PRE} \quad \text{ACT RowB} \quad \text{PRE} \]

• Using HiRA:

\[ 51.4\% \text{ reduction} \]

Overall latency of refreshing two rows reduces by 51.4% from 78.25ns down to 38ns
# Tested DRAM Chips

## Table 4: Characteristics of the tested DDR4 DRAM modules.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A0</td>
<td>G.SKILL</td>
<td>DWCW (Partial Marking)*&lt;br&gt;F4-2400C17S-8GNT [39]</td>
<td>2400</td>
<td>42-20</td>
<td>4Gb</td>
<td>B</td>
<td>x8</td>
<td>24.8%</td>
<td>25.0%</td>
<td>25.5%</td>
<td>1.75</td>
<td>1.90</td>
<td>2.52</td>
</tr>
<tr>
<td>A1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>24.9%</td>
<td>26.6%</td>
<td>28.3%</td>
<td>1.72</td>
<td>1.94</td>
<td>2.55</td>
</tr>
<tr>
<td>B0</td>
<td>Kingston</td>
<td>H5AN8G8NJ16K00-XNC&lt;br&gt;KSM32RD8/16HDR [87]</td>
<td>2400</td>
<td>48-20</td>
<td>4Gb</td>
<td>D</td>
<td>x8</td>
<td>25.1%</td>
<td>32.6%</td>
<td>36.8%</td>
<td>1.71</td>
<td>1.89</td>
<td>2.34</td>
</tr>
<tr>
<td>B1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>25.0%</td>
<td>31.6%</td>
<td>34.9%</td>
<td>1.74</td>
<td>1.91</td>
<td>2.51</td>
</tr>
<tr>
<td>C0</td>
<td>SK Hynix</td>
<td>H5ANAG8NAJ16K00-&lt;br&gt;HMAA4GU6AJR8N-XN [109]</td>
<td>2400</td>
<td>51-20</td>
<td>4Gb</td>
<td>F</td>
<td>x8</td>
<td>25.3%</td>
<td>35.3%</td>
<td>39.5%</td>
<td>1.47</td>
<td>1.89</td>
<td>2.23</td>
</tr>
<tr>
<td>C1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>29.2%</td>
<td>38.4%</td>
<td>49.9%</td>
<td>1.09</td>
<td>1.88</td>
<td>2.27</td>
</tr>
<tr>
<td>C2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>26.5%</td>
<td>36.1%</td>
<td>42.3%</td>
<td>1.49</td>
<td>1.96</td>
<td>2.58</td>
</tr>
</tbody>
</table>

* The chip identifier is partially removed on these modules. We infer the chip manufacturer and die revision based on the remaining part of the chip identifier.

HiRA-MC: HiRA Memory Controller

• **Periodic** and **preventive** refresh controllers generate each refresh request **with a deadline**
• **Refresh Table** buffers a refresh request **until its deadline**
• **Concurrent Refresh Finder** finds if HiRA can refresh a row
  - *Concurrently with a DRAM access*
  - *Concurrently with another refresh request*
HiRA for Periodic Refreshes

a) HiRA's perf. overhead, compared to No Refresh

b) HiRA's perf. improvement compared to Baseline
RowHammer Thresholds

a) PARA's probability threshold ($p_{th}$) for different values of $N_{RH}$ and $t_{RefSlack}$

b) Overall RowHammer success probability for different values of $N_{RH}$ and $t_{RefSlack}$
HiRA for Preventive Refreshes

a) PARA's perf. overhead with and without HiRA

b) HiRA's perf. improvement compared to PARA
HiRA for Periodic Refresh

Sensitivity to Number of Channels and Ranks
HiRA for Preventive Refresh
Sensitivity to Number of Channels and Ranks

[Graphs showing normalized weighted speedup for different numbers of RH and ranks per channel]
Workload Memory Access Characteristics

- 125 different 8-core multiprogrammed workloads
- Three histograms showing MPKI, RBCPKI, and RBHPKI respectively
RowHammer Mitigation across Generations

Refresh Delay

- DDRx protocols allow a REF command to be **postponed** for ~70us

- HiRA-MC’s current design *does not* leverage this flexibility

- A **longer time slack** allows
  - the baseline to **better utilize** DRAM idle time to perform refresh operations
  - HiRA to find **more opportunities** to perform a refresh operation **concurrently with** a DRAM access

- **Future sensitivity study:** the effect of long refresh delays

*SAFARI*
Energy

• HiRA does not change the number of refresh operations at a given time window
  - Overall energy consumed for refresh operations is the same

• HiRA improves system performance
  - Reduces the background energy consumption

• Evaluation requires an accurate power model based on real system measurements, similar to VAMPIRE [Ghose+ SIGMETRICS’17], but for HiRA operations
HiRA in Off-the-Shelf DRAM Chips: Key Results

• HiRA works in **56 off-the-shelf DRAM chips** from **SK Hynix**

• **51.4% reduction** in the time spent for refresh operations

• A given row’s refresh can be performed **concurrently with** the activation of any of the **32% of the rows** in the same bank

HiRA effectively reduces the time spent for refresh operations in **off-the-shelf** DRAM chips
HiRA: Hidden Row Activation
for Reducing Refresh Latency of Off-the-Shelf DRAM Chips

Backup Slides

Abdullah Giray Yağlıkçı
Ataberk Olgun  Minesh Patel  Haocong Luo  Hasan Hassan
Lois Orosa  Oğuz Ergin  Onur Mutlu

SAFARI

ETH zürich  CESGA  TOBB ETÜ
Profiling for DRAM Data Retention Failures
Finding DRAM Retention Failures

- How can we reliably find the retention time of all DRAM cells?

- Goals: so that we can
  - Make DRAM reliable and secure
  - Make techniques like RAIDR work
    - improve performance and energy
The Efficacy of Error Mitigation Techniques for DRAM Retention Failures: A Comparative Experimental Study

Samira Khan, Donghyuk Lee, Yoongu Kim, Alaa Alameldeen, Chris Wilkerson, and Onur Mutlu,
"The Efficacy of Error Mitigation Techniques for DRAM Retention Failures: A Comparative Experimental Study"
Towards an Online Profiling System

**Key Observations:**

- **Testing alone cannot detect** all possible failures
- **Combination of ECC and other mitigation techniques** is much more **effective**
  - But degrades performance
- **Testing can help to reduce the ECC strength**
  - Even when starting with a **higher strength ECC**

Towards an Online Profiling System

1. Initially Protect DRAM with Strong ECC
2. Periodically Test Parts of DRAM
3. Mitigate errors and reduce ECC

Run tests periodically after a short interval at smaller regions of memory
Handing Variable Retention Time [DSN’15]

- Moinuddin Qureshi, Dae Hyun Kim, Samira Khan, Prashant Nair, and Onur Mutlu, "AVATAR: A Variable-Retention-Time (VRT) Aware Refresh for DRAM Systems"
  Proceedings of the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Rio de Janeiro, Brazil, June 2015.
  [Slides (pptx) (pdf)]

AVATAR: A Variable-Retention-Time (VRT) Aware Refresh for DRAM Systems

Moinuddin K. Qureshi†  Dae-Hyun Kim†  Samira Khan†  Prashant J. Nair†  Onur Mutlu‡
†Georgia Institute of Technology
†{moin, dhkim, pnair6}@ece.gatech.edu
‡Carnegie Mellon University
‡{samirakhan, onur}@cmu.edu
Insight: Avoid retention failures ➔ Upgrade row on ECC error

Observation: Rate of VRT >> Rate of soft error (50x-2500x)

AVATAR mitigates VRT by increasing refresh rate on error
RESULTS: REFRESH SAVINGS

AVATAR reduces refresh by 60%-70%, similar to multi-rate refresh but with VRT tolerance.

Retention Testing Once a Year can increase refresh savings from 60% to 70%.
AVATAR obtains 2/3\textsuperscript{rd} the performance of NoRefresh. Higher benefits at higher capacity nodes.
AVATAR reduces EDP.
Significant reduction at higher capacity nodes.
Handling Data-Dependent Failures [DSN’16]

- Samira Khan, Donghyuk Lee, and Onur Mutlu, "PARBOR: An Efficient System-Level Technique to Detect Data-Dependent Failures in DRAM"

[Slides (pptx) (pdf)]

PARBOR: An Efficient System-Level Technique to Detect Data-Dependent Failures in DRAM

- Samira Khan*  
  *University of Virginia

- Donghyuk Lee†‡  
  †Carnegie Mellon University

- Onur Mutlu*†  
  ‡Nvidia  
  *ETH Zürich
Samira Khan, Chris Wilkerson, Zhe Wang, Alaa R. Alameldeen, Donghyuk Lee, and Onur Mutlu,
"Detecting and Mitigating Data-Dependent DRAM Failures by Exploiting Current Memory Content"
Proceedings of the 50th International Symposium on Microarchitecture (MICRO), Boston, MA, USA, October 2017.
[Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)] [Poster (pptx) (pdf)]
Handling Both DPD and VRT [ISCA’17]

- Minesh Patel, Jeremie S. Kim, and Onur Mutlu, "The Reach Profiler (REAPER): Enabling the Mitigation of DRAM Retention Failures via Profiling at Aggressive Conditions"
  
  
  [Slides (pptx) (pdf)]
  [Lightning Session Slides (pptx) (pdf)]

- First experimental analysis of (mobile) LPDDR4 chips
- Analyzes the complex tradeoff space of retention time profiling
- Idea: enable fast and robust profiling at higher refresh intervals & temperatures

The Reach Profiler (REAPER): Enabling the Mitigation of DRAM Retention Failures via Profiling at Aggressive Conditions

Minesh Patel$\dagger\dagger$  Jeremie S. Kim$\dagger\dagger\ddagger$  Onur Mutlu$\dagger\dagger$

$\dagger$ETH Zürich  $\ddagger$Carnegie Mellon University
The Reach Profiler (REAPER): Enabling the Mitigation of DRAM Retention Failures via Profiling at Aggressive Conditions

Minesh Patel     Jeremie S. Kim
Onur Mutlu
Leaky Cells

Periodic DRAM Refresh

Performance + Energy Overhead
Goal: find all retention failures for a refresh interval $T >$ default (64ms)
Process, voltage, temperature

Variable retention time

Data pattern dependence
Characterization of 368 LPDDR4 DRAM Chips

1. Cells are more likely to fail at an increased (refresh interval | temperature)

2. Complex tradeoff space between profiling (speed & coverage & false positives)
refresh interval

temperature

operate here

REACH PROFILING

profile here
Reach Profiling

A new DRAM retention failure profiling methodology

+ Faster and more reliable than current approaches

+ Enables longer refresh intervals
Experimental Infrastructure

- 368 2y-nm LPDDR4 DRAM chips
  - 4Gb chip size
  - From 3 major DRAM vendors

- Thermally controlled testing chamber
  - Ambient temperature range: \{40°C – 55°C\} ± 0.25°C
  - DRAM temperature is held at 15°C above ambient
LPDDR4 Studies

1. Temperature

2. Data Pattern Dependence

3. Retention Time Distributions

4. Variable Retention Time

5. Individual Cell Characterization
New failing cells continue to appear over time
  - Attributed to variable retention time (VRT)
• The set of failing cells changes over time
New failing cells continue to appear over time
- Attributed to variable retention time (VRT)

The set of failing cells changes over time
Single-cell Failure Probability (Cartoon)

- **Probability of Read Failure**
- **Refresh Interval (s)**

*idealized cell* (retention time = 3 s)
Single-cell Failure Probability (Cartoon)

- **idealized cell**
  - (retention time = 3s)

- **actual cell**
  - $N(\mu, \sigma) \mid \mu = 3s$

Diagram showing the probability of read failure vs. refresh interval (s) with idealized and actual cell models.
Single-cell Failure Probability (Real)

Read Failure Probability vs. Refresh Interval (s)

- Single-cell Failure Probability (Real)
  - 22/36
Single-cell Failure Probability (Real)

- Read Failure Probability
- Refresh Interval (s)

Failed 9 times out of 16 trials
Single-cell Failure Probability (Real)

Read Failure Probability

Refresh Interval (s)
Single-cell Failure Probability (Real)

Read Failure Probability

Refresh Interval (s)
Single-cell Failure Probability (Real)

operate here

Read Failure Probability

Refresh Interval (s)

SAFARI
Single-cell Failure Probability (Real)

Read Failure Probability vs. Refresh Interval (s)

operate here

hard to find
Single-cell Failure Probability (Real)

Read Failure Probability

Refresh Interval (s)

operate here

profile here

hard to find
Single-cell Failure Probability (Real)

- Operate here
- Profile here

Read Failure Probability

Refresh Interval (s)

- Hard to find
- Easy to find

22/36
Single-cell Failure Probability (Real)

Read Failure Probability

Refresh Interval (s)

operate here

profile here

easy to find

false positives

hard to find

SAFARI

22/36
Any cell is more likely to fail at a *longer* refresh interval OR a *higher* temperature.
1. DRAM Refresh Background
2. Failure Profiling Challenges
3. Current Approaches
4. LPDDR4 Characterization
5. Reach Profiling
6. End-to-end Evaluation
Reach Profiling

**Key idea:** profile at a *longer refresh interval* and/or a *higher temperature*
Reach Profiling

**Key idea:** profile at a *longer refresh interval* and/or a *higher temperature*

![Graph showing refresh interval vs. temperature with 'operate here' and 'profile here' points]
Reach Profiling

**Key idea:** profile at a *longer refresh interval* and/or a *higher temperature*

**Pros**

- **Fast + Reliable:** reach profiling searches for cells *where they are most likely to fail*

**Cons**

- **False Positives:** profiler may identify cells that fail under profiling conditions, but not under operating conditions
Towards an Implementation

Reach profiling is a general methodology

3 key questions for an implementation:

- What are desirable profiling conditions?
- How often should the system profile?
- What information does the profiler need?
Three Key Profiling Metrics

1. **Runtime**: how long profiling takes

2. **Coverage**: portion of all possible failures discovered by profiling

3. **False positives**: number of cells observed to fail during profiling but never during actual operation
Three Key Profiling Metrics

1. **Runtime**: how long profiling takes

2. **Coverage**: portion of all possible failures discovered by profiling

We explore how these three metrics change under many different profiling conditions
Evaluation Methodology

• Simulators
  - **Performance**: Ramulator [Kim+, CAL’15]
  - **Energy**: DRAMPower [Chandrasekar+, DSD’11]

• Configuration
  - 4-core (4GHz), 8MB LLC
  - LPDDR4-3200, 4 channels, 1 rank/channel

• Workloads
  - 20 random 4-core benchmark mixes
  - SPEC CPU2006 benchmark suite
Simulated End-to-end Performance

- Brute-force profiling
- REAPER
- Ideal profiling

64 Gb

end-to-end system performance gain

refresh interval (ms)

- 128
- 256
- 512
- 768
- 1024
- 1280
- 1536
- no ref
Simulated End-to-end Performance

- Brute-force profiling
- REAPER
- Ideal profiling

**64 Gb**

end-to-end performance gain

refresh interval (ms)

<table>
<thead>
<tr>
<th>Interval (ms)</th>
<th>128</th>
<th>256</th>
<th>512</th>
<th>768</th>
<th>1024</th>
<th>1280</th>
<th>1536</th>
<th>no ref</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reprofile</td>
<td>rarely</td>
<td>Reprofile</td>
<td>often</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Simulated End-to-end Performance

- Brute-force profiling
- REAPER
- Ideal profiling

**64 Gb**

- Refresh interval (ms):
  - 128
  - 256
  - 512
  - 768
  - 1024
  - 1280
  - 1536
  - No ref

**End-to-end system performance gain**

- Reprofile rarely
- Reprofile often
Simulated End-to-end Performance

On average, REAPER enables:

- **16.3%** system performance improvement
- **36.4%** DRAM power reduction

REAPER enables longer refresh intervals, which are unreasonable using brute-force profiling.

SAFARI

Reprofile rarely

Reprofile often
Other Analyses in the Paper

• Detailed LPDDR4 characterization data
  - Temperature dependence effects
  - Retention time distributions
  - Data pattern dependence
  - Variable retention time
  - Individual cell failure distributions

• Profiling tradeoff space characterization
  - Runtime, coverage, and false positive rate
  - Temperature and refresh interval

• Probabilistic model for tolerable failure rates

• Detailed results for end-to-end evaluations
**REAPER Summary**

**Problem:**
- DRAM refresh performance and energy overhead is high
- Current approaches to retention failure profiling are slow or unreliable

**Goals:**
1. Thoroughly analyze profiling tradeoffs
2. Develop a **fast** and **reliable** profiling mechanism

**Key Contributions:**
1. **First** detailed characterization of 368 LPDDR4 DRAM chips
2. **Reach profiling:** Profile at a **longer refresh interval** or **higher temperature** than target conditions, where cells are more likely to fail

**Evaluation:**
- **2.5x** faster profiling with **99%** coverage and **50%** false positives
- REAPER enables **16.3% system performance improvement** and **36.4% DRAM power reduction**
- Enables longer refresh intervals that were previously unreasonable
Handling Both DPD and VRT [ISCA’17]

- Minesh Patel, Jeremie S. Kim, and Onur Mutlu,
  "The Reach Profiler (REAPER): Enabling the Mitigation of DRAM Retention Failures via Profiling at Aggressive Conditions"

[Slides (pptx) (pdf)]
[Lightning Session Slides (pptx) (pdf)]

- First experimental analysis of (mobile) LPDDR4 chips
- Analyzes the complex tradeoff space of retention time profiling
- Idea: enable fast and robust profiling at higher refresh intervals & temperatures

The Reach Profiler (REAPER):
Enabling the Mitigation of DRAM Retention Failures via Profiling at Aggressive Conditions

Minesh Patel‡‡ Jeremie S. Kim‡‡ Onur Mutlu‡‡
‡ETH Zürich ‡‡Carnegie Mellon University
In-DRAM ECC Complicates Things [DSN’19]

- Minesh Patel, Jeremie S. Kim, Hasan Hassan, and Onur Mutlu, "Understanding and Modeling On-Die Error Correction in Modern DRAM: An Experimental Study Using Real Devices"


[Slides (pptx) (pdf)]
[Talk Video (26 minutes)]
[Full Talk Lecture (29 minutes)]
[Source Code for EINSim, the Error Inference Simulator]

Best paper award.
More on In-DRAM ECC [MICRO’20]

- Minesh Patel, Jeremie S. Kim, Taha Shahroodi, Hasan Hassan, and Onur Mutlu,
  "Bit-Exact ECC Recovery (BEER): Determining DRAM On-Die ECC Functions by Exploiting DRAM Data Retention Characteristics"


[Slides (pptx) (pdf)]
[Short Talk Slides (pptx) (pdf)]
[Lightning Talk Slides (pptx) (pdf)]
[Lecture Slides (pptx) (pdf)]
[Talk Video (15 minutes)]
[Short Talk Video (5.5 minutes)]
[Lightning Talk Video (1.5 minutes)]
[Lecture Video (52.5 minutes)]
[BEER Source Code]

Best paper award.

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‡‡
†ETH Zürich    ‡Carnegie Mellon University
Profiling In The Presence of ECC [MICRO’21]

- Minesh Patel, Geraldo F. de Oliveira Jr., and Onur Mutlu,
  "HARP: Practically and Effectively Identifying Uncorrectable Errors in
  Memory Chips That Use On-Die Error-Correcting Codes"
  Proceedings of the 54th International Symposium on Microarchitecture (MICRO),
  Virtual, October 2021.
  [Slides (pptx) (pdf)]
  [Short Talk Slides (pptx) (pdf)]
  [Lightning Talk Slides (pptx) (pdf)]
  [Talk Video (20 minutes)]
  [Lightning Talk Video (1.5 minutes)]
  [HARP Source Code (Officially Artifact Evaluated with All Badges)]

HARP: Practically and Effectively
Identifying Uncorrectable Errors in Memory Chips
That Use On-Die Error-Correcting Codes

Minesh Patel
ETH Zürich

Geraldo F. Oliveira
ETH Zürich

Onur Mutlu
ETH Zürich