Computer Architecture

Lecture 8: Data Retention and Memory Refresh

A. Giray Yaglikci
Prof. Onur Mutlu
ETH Zürich

Fall 2022

20 October 2023

Recall: Crossing the Abstraction Layers

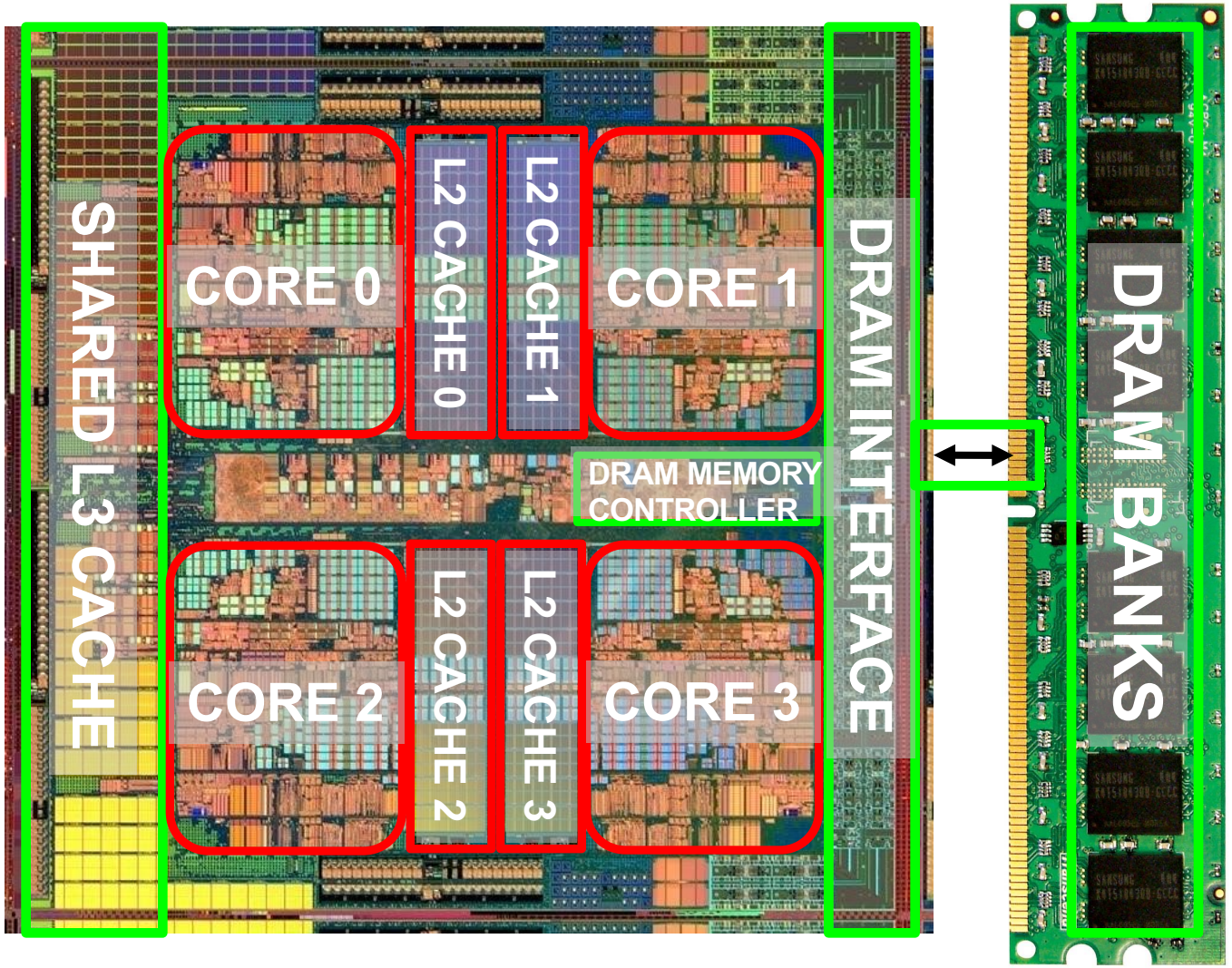
- Two key goals of this course are
  - to understand how a computing system works underneath the software layer and how decisions made in hardware affect the software/programmer
  - to enable you to be comfortable in making design and optimization decisions that cross the boundaries of different layers and system components

Another Example

- DRAM Refresh

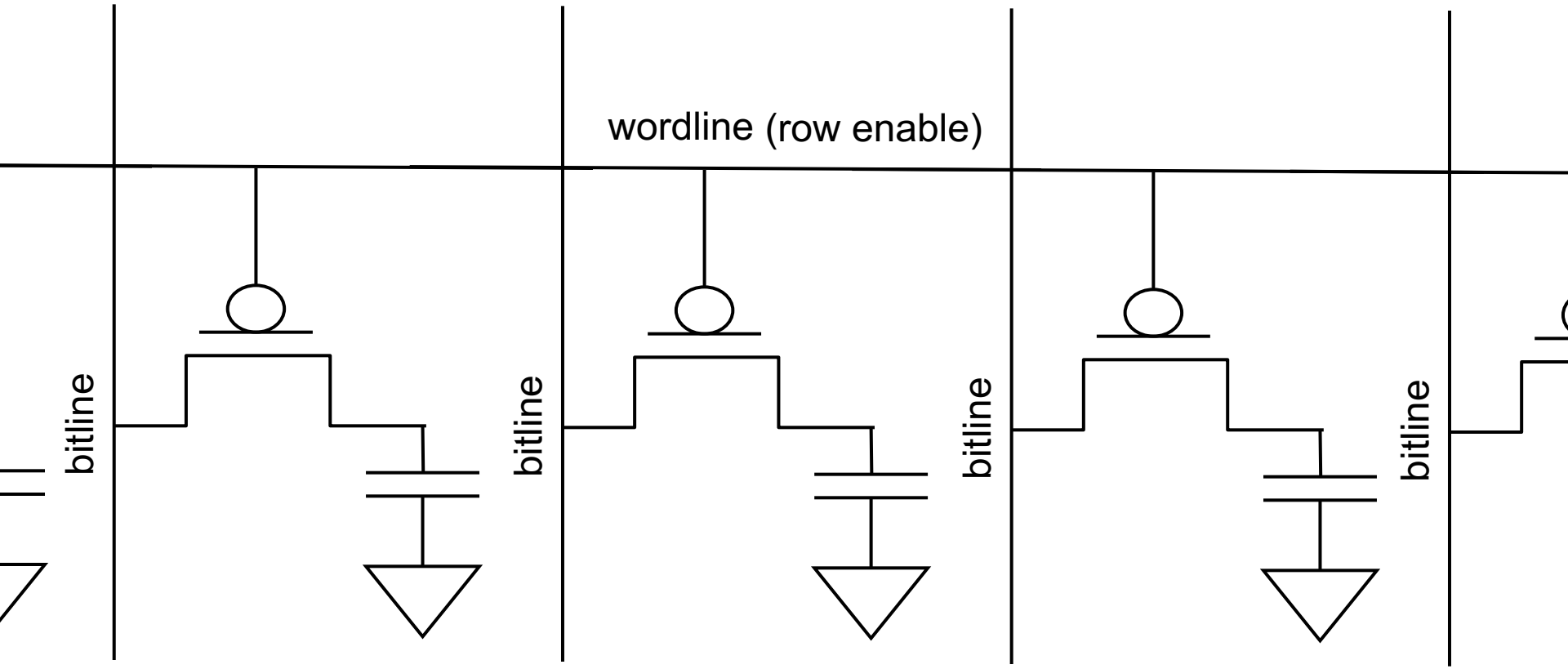
DRAM in the System

Multi-Core Chip



*Die photo credit: AMD Barcelona

A DRAM Cell



- 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?

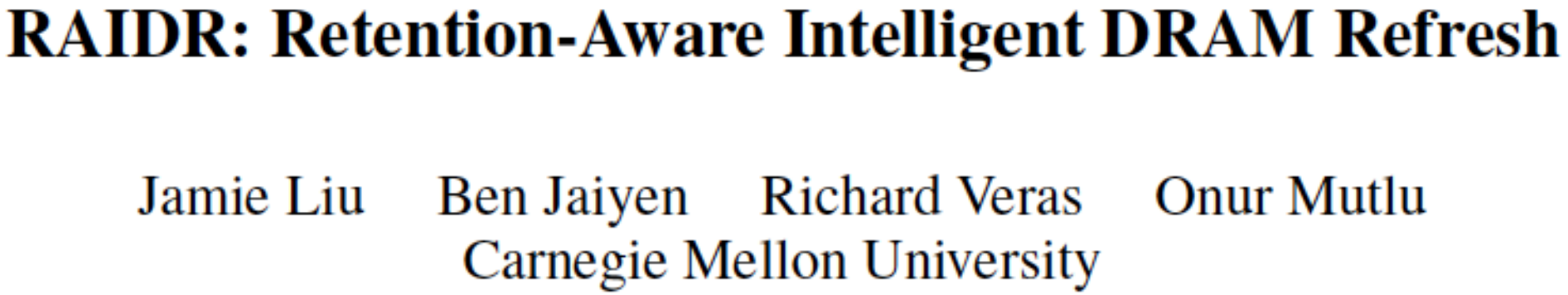
- A good exercise…

A Leaky DRAM Cell

DRAM Leakage in ISCA-50 25-Year Retrospective Issue

RAIDR: Heterogeneous 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)]
  Selected to the ISCA-50 25-Year Retrospective Issue covering 1996-2020 in 2023 (Retrospective (pdf) Full Issue).



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)]
  Selected to the ISCA-50 25-Year Retrospective Issue covering 1996-2020 in 2023 (Retrospective (pdf) Full Issue).

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"
  *Proceedings of the 41st International Symposium on Computer Architecture (**ISCA**)*, Minneapolis, MN, June 2014.
  [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).
  Selected to the ISCA-50 25-Year Retrospective Issue covering 1996-2020 in 2023 (Retrospective (pdf) Full Issue).

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

Yoongu Kim[1] Ross Daly* Jeremie Kim[1] Chris Fallin* Ji Hye Lee[1]
Donghyuk Lee[1] Chris Wilkerson[2] Konrad Lai Onur Mutlu[1]

[1]Carnegie Mellon University [2]Intel Labs

RAIDR Retrospective [ISCA 2012]

Retrospective: RAIDR: Retention-Aware Intelligent DRAM Refresh

Onur Mutlu
ETH Zürich

***Abstract*—Dynamic Random Access Memory (DRAM) is the prevalent memory technology used to build main memory systems of almost all computers. A fundamental shortcoming of DRAM is the need to refresh memory cells to keep stored data intact. DRAM refresh consumes energy and degrades performance. It is also a technology scaling challenge as its negative effects become worse as DRAM cell size reduces and DRAM chip capacity increases.**

Our ISCA 2012 paper, RAIDR [1], examines the DRAM refresh problem from a modern computing systems perspective, demonstrating its projected impact on systems with higher-capacity DRAM chips expected to be manufactured in the future. It proposes and evaluates a simple and low-cost solution that greatly reduces the performance & energy overheads of refresh by exploiting variation in data retention times across DRAM rows. The key idea is to group the DRAM rows into bins in terms of their minimum data retention times, store the bins in low-cost Bloom filters, and refresh rows in different bins at different rates. Evaluations in our paper (and later works) show that the idea greatly improves performance & energy efficiency and its benefits increase with DRAM chip capacity. The paper embodies an approach we have termed *system-DRAM co-design*.

This short retrospective provides a brief analysis of our RAIDR paper and its impact. We briefly describe the mindset and circumstances that led to our focus on the DRAM refresh problem and RAIDR's development, discuss later works that provided improved analyses and solutions, and make some educated guesses on what the future may bring on the DRAM refresh problem (and more generally in DRAM technology scaling).

I. Background, Approach & Mindset

At the time we began our focus on solving the DRAM refresh (i.e., data retention) challenge in late 2010, my research group, SAFARI, had already been working on memory controllers and memory technology scaling issues, motivated by many challenges memory systems, in particular the DRAM technology [2], have been facing (as described in, e.g., [3-5]). Our intense work on memory systems started during my tenure at Microsoft Research from 2006 and continued at CMU from 2009. For example, we had developed better memory schedulers for multi-core processors (e.g., [6-10]), developed platforms to perform voltage and frequency scaling to save DRAM energy (e.g., [11]) and architected emerging memory technologies to replace or augment DRAM (e.g., [12-14]). We were quite excited about the prospect of much more capable memory controllers in enabling better memory systems. As such, we were pursuing new memory-controller and system-level techniques to 1) overcome the challenging device- and circuit-level scaling issues of memory technologies and 2) better exploit underlying characteristics of memory technology; an approach we termed *system-DRAM co-design* [4, 5].

RAIDR is a product of this approach. Our focus on data retention issues and other low-level issues in DRAM especially increased via discussions with the Samsung DRAM Design Team, who visited us in April 2011 and encouraged the development of our system-level solutions to DRAM issues, enabling strong support both technically and funding-wise. In fact, much of our ensuing research in DRAM was supported by generous gift funding by and technical discussions with Samsung based on a proposal entitled *"New ideas to enhance DRAM scaling: Scaling-aware controller design and co-design of DRAM and controllers"* (Intel provided similar gift funding and technical discussions).

II. Contributions and Impact of RAIDR

RAIDR is the first work to propose a low-cost memory controller technique that reduces refresh operations by exploiting variation in data retention times across DRAM rows. Its appeal comes from its simplicity and low cost, enabled by the careful use of Bloom filters [15]. Exploiting the DRAM data retention time distribution [16], RAIDR can eliminate a very large fraction (e.g., ~75% or more) of refresh operations with very small hardware cost at the memory controller.

Apart from the new technique it introduced, we believe the RAIDR paper made two other major contributions that have enabled a large number of future works and new ideas. First, it provided an empirical scaling analysis that clearly demonstrated the importance of the DRAM refresh problem in modern systems: if nothing is done about it, DRAM refresh would waste almost half of the throughput and half of the energy of a high-capacity 64-Gb DRAM chip! This analytical prediction encouraged more works in the topic area. Second, it demonstrated a methodical way of exploiting cell-level heterogeneous data retention times at the system (e.g., memory controller) level: if data retention times of DRAM rows are accurately known, the system can use them to optimize DRAM refresh and get rid of most refresh operations. This demonstration enabled other works to develop 1) methods for accurately determining DRAM data retention times and 2) other system-level approaches to optimize DRAM behavior using data retention time information.

III. Building on RAIDR and Making It Work

We believe RAIDR enabled a refreshing approach to DRAM refresh. Its largest contribution could be the works it has inspired that rigorously examined the questions of 1) how to perform accurate DRAM data retention time profiling, 2) how to overcome potential hurdles that stand in the way of obtaining accurate minimum data retention times, 3) how to reliably get rid of unnecessary refresh operations.

We wanted to make RAIDR work in a real system setting. To this end, collaboratively with Intel, we developed an FPGA-based flexible DRAM testing infrastructure [17] that enabled us to rigorously test data retention times of cells in real DDR3 DRAM chips. Using this infrastructure, later open sourced as SoftMC [18, 19] and DRAM Bender [20, 21], we experimentally examined practical issues that affect the accuracy (and performance) of DRAM data retention time profiling. We analyzed two major issues that make such profiling very challenging: 1) data pattern dependence (DPD) of retention times [17, 22], and 2) the variable retention time (VRT) phenomenon [17, 23, 24]. Our follow-up work, which appeared at ISCA 2013 [17], provides a detailed experimental analysis of these challenges in cutting-edge DRAM chips, demonstrating that ideas like RAIDR that depend on accurate identification of retention times are not easy to exploit in practice. Later works (e.g., [25-32]) developed new methods for making RAIDR-like techniques more practical by tackling especially the DPD and VRT problems and enhancing retention time profiling methods to work in the presence of DPD and VRT, usually by exploiting ECC techniques that have since become mainstream in DRAM chips (see [31, 33]) to tolerate VRT [34].

The development of our flexible FPGA-based DRAM testing infrastructure also enabled experimental DRAM research in directions that are completely different from retention time profiling and refresh. These include studies that provided valuable experimental data on various DRAM characteristics, including RowHammer [20, 35, 44], latency [45-48], voltage-latency-reliability relationship [49], power consumption and modeling [50]. Using this infrastructure, later research also demonstrated the ability of real off-the-shelf DRAM chips to perform data copy/initialization and bulk bitwise operations [51-55], implement physical unclonable functions [56], and generate true random numbers [57, 58]. We believe the investment we made to try to make RAIDR work using a real FPGA-based infrastructure helped us and the broader research community uncover many interesting characteristics of DRAM chips and propose new ideas to make DRAM-based systems more secure, reliable, efficient, and high performance.

Other later works provided refined models of DRAM refresh's impact on system performance (e.g., [59, 60]) and developed new methods to reduce DRAM refresh's negative impact on performance & energy (e.g., [41, 59-67]). Our HPCA 2014 paper [25] developed a more refined projection of the effect of DRAM refresh as technology scales. AVATAR in DSN 2015 [26] and REAPER in ISCA 2017 [30] enabled more practical ways of exploiting heterogeneous retention times in the presence of VRT. Our recent work [66] shows that with a more flexible DRAM interface that gives some autonomy to DRAM chips, RAIDR can be more efficiently implemented inside the DRAM chip.

IV. Summary and Future Outlook

RAIDR is a nice example of how enthusiastic support from industry can foster new ideas that can open up many new analyses and other ideas. We were inspired by our deep technical discussions with especially Samsung and Intel, along with prior works that described DRAM technology scaling challenges (e.g., [3]) and that developed promising solutions (e.g., [68, 69]). Engineers from Samsung and Intel later wrote an insightful paper [34] on DRAM scaling challenges, which described refresh as a key problem and advocated a controller-DRAM co-design approach as we had been advocating [1, 4]. RAIDR was also a nice example of how teaching & research smoothly feed each other: much of the research was done as part of a group project in the Parallel Computer Architecture class I taught at CMU in Fall 2011.

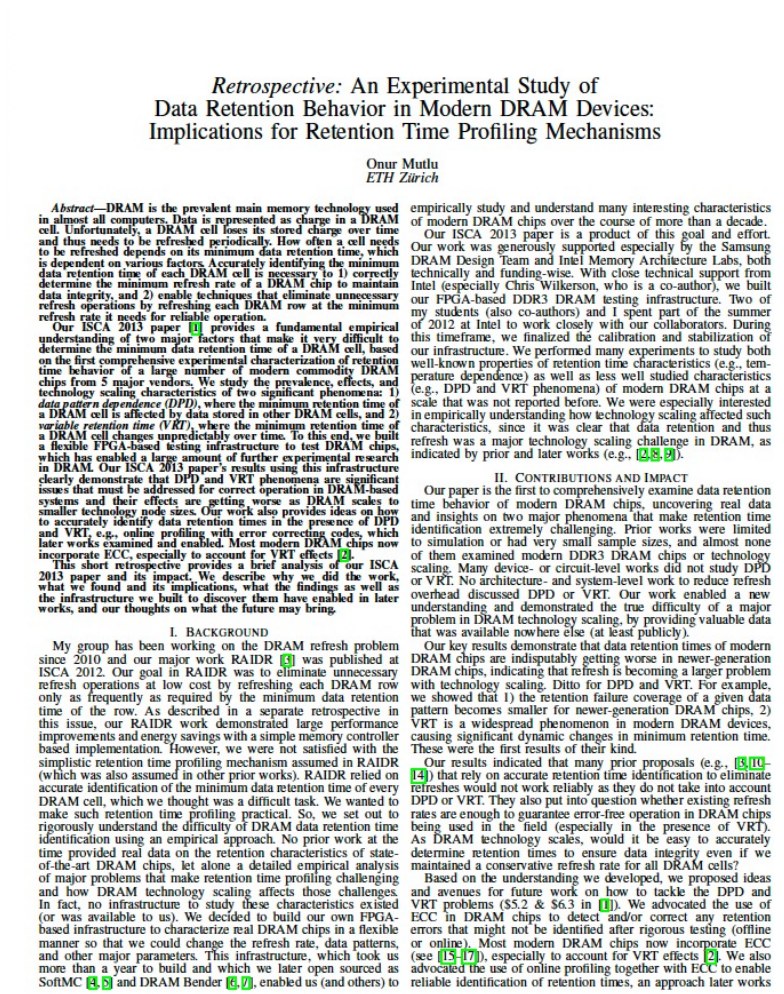
Looking forward, DRAM technology scaling is getting worse and data retention will continue to be an important issue [34, 70]. The negative effects of DRAM refresh will be (and are being) exacerbated by other technology scaling issues like RowHammer [35] that require even more refreshes as a solution [41, 44, 71]. We believe there are a lot more new ideas and techniques to develop to minimize the impact of refresh on computing systems.

References

[1] J. Liu *et al.*, "RAIDR: Retention-Aware Intelligent DRAM Refresh," in *ISCA*, 2012.
[2] R. H. Dennard, "Field-effect Transistor Memory," 1968, US Patent 3,387,286.
[3] J. A. Mandelman *et al.*, "Challenges and Future Directions for the Scaling of Dynamic Random-Access Memory (DRAM)," *IBM JRD*, 2002.
[4] O. Mutlu, "Memory Scaling: A Systems Architecture Perspective," *IMW*, 2013.
[5] O. Mutlu and L. Subramanian, "Research Problems and Opportunities in Memory Systems," *SUPERFRI*, 2014.
[6] O. Mutlu and T. Moscibroda, "Stall-Time Fair Memory Access Scheduling for Chip Multiprocessors," in *MICRO*, 2007.
[7] O. Mutlu *et al.*, "Parallelism-Aware Batch Scheduling: Enhancing Both Performance and Fairness of Shared DRAM Systems," in *ISCA*, 2008.
[8] Y. Kim *et al.*, "Thread Cluster Memory Scheduling: Exploiting Differences in Memory Access Behavior," in *MICRO*, 2010.
[9] Y. Kim *et al.*, "ATLAS: A Scalable and High-performance Scheduling Algorithm for Multiple Memory Controllers," in *HPCA*, 2010.
[10] S. Muralidhara, "Reducing Memory Interference in Multicore Systems via Application-aware Memory Channel Partitioning," in *MICRO*, 2011.
[11] H. David *et al.*, "Memory power management via dynamic voltage/frequency scaling," in *ICAC*, 2011.
[12] B. C. Lee *et al.*, "Architecting Phase Change Memory as a Scalable DRAM Alternative," in *ISCA*, 2009.
[13] H. Yoon *et al.*, "Row Buffer Locality Aware Caching Policies for Hybrid Memories," in *ICCD*, 2012.
[14] J. Meza *et al.*, "Enabling Efficient and Scalable Hybrid Memories using Fine-granularity DRAM Cache Management," *CAL*, 2012.
[15] B. Bloom, "Space/Time Trade-Offs in Hash Coding with Allowable Errors," *CACM*, 1970.
[16] K. Kim and J. Lee, "A new investigation of data retention time in truly nanoscaled DRAMs," *IEEE EDL*, vol. 30, no. 8, 2009.
[17] J. Liu *et al.*, "An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms," in *ISCA*, 2013.
[18] H. Hassan *et al.*, "SoftMC: A Flexible and Practical Open-Source Infrastructure for Enabling Experimental DRAM Studies," in *HPCA*, 2017.
[19] "SoftMC Repository," https://github.com/CMU-SAFARI/SoftMC/.
[20] A. Olgun *et al.*, "DRAM Bender: An Extensible and Versatile FPGA-based Infrastructure to Easily Test State-of-the-art DRAM Chips," *TCAD*, 2023.
[21] "DRAM Bender," https://github.com/CMU-SAFARI/DRAM-Bender/.
[22] Y. Nakagome *et al.*, "The impact of data-line interference noise on DRAM scaling," *JSSC*, 1988.
[23] D. Yaney *et al.*, "A Meta-stable Leakage Phenomenon in DRAM Charge Storage - Variable Hold Time," in *IEDM*, 1987.
[24] P. J. Restle *et al.*, "DRAM Variable Retention Time," in *IEDM*, 1992.
[25] S. Khan, "The Efficacy of Error Mitigation Techniques for DRAM Retention Failures: A Comparative Experimental Study," in *SIGMETRICS*, 2014.
[26] M. K. Qureshi *et al.*, "AVATAR: A Variable-Retention-Time (VRT) Aware Refresh for DRAM Systems," in *DSN*, 2015.
[27] S. Khan *et al.*, "PARBOR: An Efficient System-Level Technique to Detect Data-Dependent Failures in DRAM," in *DSN*, 2016.
[28] S. Khan *et al.*, "A Case for Memory Content-Based Detection and Mitigation of Data-Dependent Failures in DRAM," *CAL*, 2016.
[29] S. Khan *et al.*, "Detecting and Mitigating Data-Dependent DRAM Failures by Exploiting Current Memory Content," in *MICRO*, 2017.
[30] M. Patel, "The Reach Profiler (REAPER): Enabling the Mitigation of DRAM Retention Failures via Profiling at Aggressive Conditions," in *ISCA*, 2017.
[31] M. Patel *et al.*, "Bit-Exact ECC Recovery (BEER): Determining DRAM On-Die ECC Functions by Exploiting DRAM Data Retention Characteristics," in *MICRO*, 2020.
[32] M. Patel *et al.*, "HARP: Practically and Effectively Identifying Uncorrectable Errors in Main Memory Chips That Use On-Die ECC," in *MICRO*, 2021.
[33] M. Patel *et al.*, "Understanding and Modeling On-die Error Correction in Modern DRAM: An Experimental Study using Real Devices," in *DSN*, 2019.
[34] U. Kang *et al.*, "Co-Architecting Controllers and DRAM to Enhance DRAM Process Scaling," in *The Memory Forum*, 2014.
[35] Y. Kim *et al.*, "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors," in *ISCA*, 2014.
[36] J. S. Kim *et al.*, "Revisiting RowHammer: An Experimental Analysis of Modern DRAM Devices and Mitigation Techniques," in *ISCA*, 2020.
[37] L. Orosa, "A Deeper Look into RowHammer's Sensitivities: Experimental Analysis of Real DRAM Chips and Implications on Future Attacks and Defenses," in *MICRO*, 2021.
[38] M. Farmani *et al.*, "RHAT: Efficient RowHammer-Aware Test for Modern DRAM Modules," in *ETS*, 2021.
[39] P. Frigo *et al.*, "TRRespass: Exploiting the Many Sides of Target Row Refresh," in *S&P*, 2020.
[40] H. Hassan *et al.*, "Uncovering In-DRAM RowHammer Protection Mechanisms: A New Methodology, Custom RowHammer Patterns, and Implications," in *MICRO*, 2021.
[41] A. G. Yağlıkçı *et al.*, "HiRA: Hidden Row Activation for Reducing Refresh Latency of Off-the-Shelf DRAM Chips," in *MICRO*, 2022.
[42] A. G. Yağlıkçı *et al.*, "Understanding RowHammer Under Reduced Wordline Voltage: An Experimental Study Using Real DRAM Devices," in *DSN*, 2022.
[43] A. Olgun *et al.*, "An Experimental Analysis of RowHammer in HBM2 DRAM Chips," in *DSN Disrupt*, 2023.
[44] H. Luo *et al.*, "RowPress: Amplifying Read Disturbance in Modern DRAM Chips," in *ISCA*, 2023.
[45] D. Lee *et al.*, "Adaptive-Latency DRAM: Optimizing DRAM Timing for the Common-Case," in *HPCA*, 2015.
[46] K. K. Chang *et al.*, "Understanding Latency Variation in Modern DRAM Chips: Experimental Characterization, Analysis, and Optimization," in *SIGMETRICS*, 2016.
[47] D. Lee *et al.*, "Design-induced Latency Variation in Modern DRAM Chips: Characterization, Analysis, and Latency Reduction Mechanisms," *POMACS*, 2017.
[48] J. Kim *et al.*, "Solar-DRAM: Reducing DRAM Access Latency by Exploiting the Variation in Local Bitlines," in *ICCD*, 2018.
[49] K. Chang *et al.*, "Understanding Reduced-Voltage Operation in Modern DRAM Devices: Experimental Characterization, Analysis, and Mechanisms," in *SIGMETRICS*, 2017.
[50] S. Ghose *et al.*, "What Your DRAM Power Models Are Not Telling You: Lessons from a Detailed Experimental Study," in *SIGMETRICS*, 2018.
[51] V. Seshadri *et al.*, "Fast Bulk Bitwise AND and OR in DRAM," *CAL*, 2015.
[52] V. Seshadri *et al.*, "Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology," in *MICRO*, 2017.
[53] A. Olgun *et al.*, "PiDRAM: A Holistic End-to-end FPGA-based Framework for Processing-in-DRAM"," *TACO*, 2023.
[54] F. Gao *et al.*, "ComputeDRAM: In-Memory Compute Using Off-the-Shelf DRAMs," in *MICRO*, 2019.
[55] F. Gao *et al.*, "FracDRAM: Fractional Values in Off-the-Shelf DRAM," in *MICRO*, 2022.
[56] J. S. Kim *et al.*, "The DRAM Latency PUF: Quickly Evaluating Physical Unclonable Functions by Exploiting the Latency-Reliability Tradeoff in Modern Commodity DRAM Devices," in *HPCA*, 2018.
[57] J. Kim, "D-RaNGe: Using Commodity DRAM Devices to Generate True Random Numbers with Low Latency and High Throughput," in *HPCA*, 2019.
[58] A. Olgun *et al.*, "QUAC-TRNG: High-Throughput True Random Number Generation Using Quadruple Row Activation in Commodity DRAMs," in *ISCA*, 2021.
[59] K. K. Chang *et al.*, "Improving DRAM Performance by Parallelizing Refreshes with Accesses," in *HPCA*, 2014.
[60] J. Mukundan *et al.*, "Understanding and Mitigating Refresh Overheads in High-Density DDR4 DRAM Systems," in *ISCA*, 2013.
[61] C.-H. Lin *et al.*, "SECRET: Selective Error Correction for Refresh Energy Reduction in DRAMs," in *ICCD*, 2012.
[62] P. J. Nair *et al.*, "ArchShield: Architectural Framework for Assisting DRAM Scaling by Tolerating High Error Rates," in *ISCA*, 2013.
[63] P. Nair *et al.*, "A Case for Refresh Pausing in DRAM Memory Systems," in *HPCA*, 2013.
[64] T. Zhang *et al.*, "CREAM: A Concurrent-Refresh-Aware DRAM Memory Architecture," in *HPCA*, 2014.
[65] H. Hassan *et al.*, "CROW: A Low-Cost Substrate for Improving DRAM Performance, Energy Efficiency, and Reliability," in *ISCA*, 2019.
[66] H. Hassan *et al.*, "A Case for Self-Managing DRAM Chips: Improving Performance, Efficiency, Reliability, and Security via Autonomous in-DRAM Maintenance Operations," arXiv:2207.13358, 2022.
[67] A. Das *et al.*, "VRL-DRAM: Improving DRAM Performance via Variable Refresh Latency," in *DAC*, 2018.
[68] R. Venkatesan *et al.*, "Retention-Aware Placement in DRAM (RAPID): Software Methods for Quasi-Non-Volatile DRAM," in *HPCA*, 2006.
[69] S. Liu *et al.*, "Flikker: Saving DRAM Refresh-Power through Critical Data Partitioning," in *ASPLOS*, 2011.
[70] W. Kim, "A 1.1V 16Gb DDR5 DRAM with Probabilistic-Aggressor Tracking, Refresh-Management Functionality, Per-Row Hammer Tracking, a Multi-Step Precharge, and Core-Bias Modulation for Security and Reliability Enhancement," in *ISSCC*, 2023.
[71] O. Mutlu *et al.*, "Fundamentally Understanding and Solving RowHammer," in *ASP-DAC*, 2023.

https://people.inf.ethz.ch/omutlu/pub/RAIDR_50YearsOfISCA-Retrospective_isca23.pdf

Retention Analysis Retrospective [ISCA 2013]



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

Onur Mutlu
ETH Zürich

***Abstract*—DRAM is the prevalent main memory technology used in almost all computers. Data is represented as charge in a DRAM cell. Unfortunately, a DRAM cell loses its stored charge over time and thus needs to be refreshed periodically. How often a cell needs to be refreshed depends on its minimum data retention time, which is dependent on various factors. Accurately identifying the minimum data retention time of each DRAM cell is necessary to 1) correctly determine the minimum refresh rate of a DRAM chip to maintain data integrity, and 2) enable techniques that eliminate unnecessary refresh operations by refreshing each DRAM row at the minimum refresh rate it needs for reliable operation.**

Our ISCA 2013 paper [1] provides a fundamental empirical understanding of two major factors that make it very difficult to determine the minimum data retention time of a DRAM cell, based on the first comprehensive experimental characterization of retention time behavior of a large number of modern commodity DRAM chips from 5 major vendors. We study the prevalence, effects, and technology scaling characteristics of two significant phenomena: 1) *data pattern dependence (DPD)*, where the minimum retention time of a DRAM cell is affected by data stored in other DRAM cells, and 2) *variable retention time (VRT)*, where the minimum retention time of a DRAM cell changes unpredictably over time. To this end, we built a flexible FPGA-based testing infrastructure to test DRAM chips, which has enabled a large amount of further experimental research in DRAM. Our ISCA 2013 paper's results using this infrastructure clearly demonstrate that DPD and VRT phenomena are significant issues that must be addressed for correct operation in DRAM-based systems and their effects are getting worse as DRAM scales to smaller technology node sizes. Our work also provides ideas on how to accurately identify data retention times in the presence of DPD and VRT, e.g., online profiling with error correcting codes, which later works examined and enabled. Most modern DRAM chips now incorporate ECC, especially to account for VRT effects [2].

This short retrospective provides a brief analysis of our ISCA 2013 paper and its impact. We describe why we did the work, what we found and its implications, what the findings as well as the infrastructure we built to facilitate them have enabled in later works, and our thoughts on what the future may bring.

I. Background

My group has been working on the DRAM refresh problem since 2010 and our major work RAIDR [3] was published at ISCA 2012. Our goal in RAIDR was to eliminate unnecessary refresh operations at low cost by refreshing each DRAM row only as frequently as required by the minimum data retention time of the row. As described in a separate retrospective in this issue, our RAIDR work demonstrated large performance improvements and energy savings with a simple memory controller based implementation. However, we were not satisfied with the simplistic retention time profiling mechanism assumed in RAIDR (which was also assumed in other prior works). RAIDR relied on accurate identification of the minimum data retention time of every DRAM cell, which we thought was a difficult task. We wanted to make such retention time profiling practical. So, we set out to rigorously understand the difficulty of DRAM data retention time identification using an empirical approach. No prior work at the time provided real data on the retention characteristics of state-of-the-art DRAM chips, let alone a detailed empirical analysis of major problems that make retention time profiling challenging and how DRAM technology scaling affects those challenges. In fact, no infrastructure to study these characteristics existed (or was available to us). We decided to build our own FPGA-based infrastructure to characterize real DRAM chips in a flexible manner so that we could change the refresh rate, data patterns, and other major parameters. This infrastructure, which took us more than a year to build and which we later open sourced as SoftMC [4, 5] and DRAM Bender [6, 7], enabled us (and others) to empirically study and understand many interesting characteristics of modern DRAM chips over the course of more than a decade.

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 and funding-wise. With close technical support from Intel (especially Chris Wilkerson, who is a co-author), we built our FPGA-based DDR3 DRAM testing infrastructure. Two of my students (also co-authors) and I spent part of the summer of 2012 at Intel to work closely with our collaborators. During this timeframe, we finalized the calibration and stabilization of our infrastructure. We performed many experiments to study both well-known properties of retention time characteristics (e.g., temperature dependence) as well as less well studied characteristics (e.g., DPD and VRT phenomena) of modern DRAM chips at a scale that was not reported before. We were especially interested in empirically understanding how technology scaling affected such characteristics, since it was clear that data retention and thus refresh was a major technology scaling challenge in DRAM, as indicated by prior and later works (e.g., [2, 8, 9]).

II. 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 very small sample sizes, and almost none of them examined modern DDR3 DRAM chips or technology scaling. Many device- or circuit-level works did not study DPD or VRT. No architecture- and system-level work to reduce refresh overhead discussed DPD or VRT. Our work enabled a new understanding and demonstrated the true difficulty of a major problem in DRAM technology scaling, by providing valuable data that was available nowhere else (at least publicly).

Our key results demonstrate that data retention times of modern DRAM chips are indisputably getting worse in newer-generation DRAM chips, indicating that refresh is becoming a larger problem with technology scaling. Ditto for DPD and VRT. For example, we showed that 1) the retention failure coverage of a given data pattern becomes smaller for newer-generation DRAM chips, 2) VRT is a widespread phenomenon in modern DRAM devices, causing significant dynamic changes in minimum retention time. These were the first results of their kind.

Our results indicated that many prior proposals (e.g., [3, 10–14]) that rely on accurate retention time identification to eliminate refreshes would not work reliably as they do not take into account DPD or VRT. They also put into question whether existing refresh rates are enough to guarantee error-free operation in DRAM chips being used in the field (especially in the presence of VRT). As DRAM technology scales, would it be easy to accurately determine retention times to ensure data integrity even if we maintained a conservative refresh rate for all DRAM cells?

Based on the understanding we developed, we proposed ideas and avenues for future work on how to tackle the DPD and VRT problems ($5.2 & $6.3 in [1]). We advocated the use of ECC in DRAM chips to detect and/or correct any retention errors that might not be identified after rigorous testing (offline or online). Most modern DRAM chips now incorporate ECC (see [15–17]), especially to account for VRT effects [2]. We also advocated the use of online profiling together with ECC to enable reliable identification of retention times, an approach later works rigorously investigated and enabled (e.g., [16–23]). As such, our ISCA 2013 paper enabled system-level techniques to overcome a major DRAM scaling challenge, an approach we call *system-DRAM co-design* [24, 25]. We believe developing such system-level techniques that can detect and exploit DRAM characteristics online, during system operation, will be increasingly valuable as such characteristics will become much more difficult to accurately determine and exploit due to technology scaling.

A key contribution of our work was the development of our flexible FPGA-based DRAM testing infrastructure, which was the first of its kind. It enabled a large amount of research into DRAM chips by enabling rigorous experimental study of real DRAM chip characteristics, including the rigorous study of the RowHammer vulnerability [6, 26–36], another major DRAM technology scaling problem. We discuss some new insights and studies enabled by this infrastructure in our RAIDR retrospective and our SoftMC [4] and DRAM Bender [6] works.

III. Influence on Later Works

Many later works (e.g., [16–23]) ensued to solve the DPD & VRT problems. Some provided a more detailed characterization of the DPD and VRT phenomena: [18] analyzed both DPD & VRT and examined the effectiveness of online profiling versus ECC of varying strength. AVATAR [19] provided heterogeneous refresh rates using a combination of online profiling, ECC, and memory scrubbing, working from the empirical observation that new VRT errors are discovered infrequently at a steady rate. PARBOR [20] introduced detailed DPD analyses and a new technique to efficiently detect data-dependent failures. REAPER [23] analyzed the DPD & VRT phenomena in newer LPDDR4 DRAM chips, demonstrating that the problems are getting worse, and developed the reach profiling technique to tolerate the two problems. We believe AVATAR & REAPER enabled practical ways of exploiting heterogeneous retention times.

ECC is mainstream in DRAM chips today [15–17]. We believe this is a direct result of the analysis that showed the prevalence and importance of VRT and the difficulty of handling VRT-caused retention errors due to their fundamentally unpredictable nature. A later work by Samsung & Intel engineers [2] described that ECC is needed to deal with VRT, just as our ISCA'13 paper advocated.

IV. Summary and Future Outlook

Our ISCA 2013 paper was a nice example of harmonious collaboration between academia and industry: Intel helped us build the infrastructure and both Intel & Samsung gave us significant technical feedback along with generous funding. Our paper also highlights the importance of investing into building infrastructure to analyze real chips: doing so enabled not only the new understanding developed in our work, but also many future works that analyzed various other DRAM characteristics (e.g., [6, 16–23, 26–42]) and uncovered fascinating undocumented capabilities in real DRAM chips, e.g., the ability to perform data copy/initialization and bitwise operations [43, 47], implement physical unclonable functions [48], and generate true random numbers [49, 50].

Since ISCA 2013, we have come a long way in understanding fundamental characteristics of DRAM devices and enabling practical solutions to overcome DRAM shortcomings. Yet, there is a lot more to be empirically discovered and understood in DRAM to solve the fundamental scaling, performance, and energy challenges of the technology (as shown by very recent works in 2022-2023, e.g. [6, 30–33, 45–47]), which can enable solutions also applicable to other technologies. We conclude that the future is bright in experimental memory systems research using real memory chips.

References

[1] J. Liu *et al.*, "An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms," *ISCA*, 2013.
[2] U. Kang *et al.*, "Co-Architecting Controllers and DRAM to Enhance DRAM Process Scaling," in *The Memory Forum*, 2014.
[3] J. Liu *et al.*, "RAIDR: Retention-Aware Intelligent DRAM Refresh," in *ISCA*, 2012.
[4] H. Hassan *et al.*, "SoftMC: A Flexible and Practical Open-Source Infrastructure for Enabling Experimental DRAM Studies," in *HPCA*, 2017.
[5] SoftMC Source Code, https://github.com/CMU-SAFARI/SoftMC
[6] A. Olgun *et al.*, "DRAM Bender: An Extensible and Versatile FPGA-based Infrastructure to Easily Test State-of-the-art DRAM Chips," *TCAD*, 2023.
[7] "DRAM Bender," https://github.com/CMU-SAFARI/DRAM-Bender
[8] J. A. Mandelman *et al.*, "Challenges and Future Directions for the Scaling of Dynamic Random-Access Memory (DRAM)," *IBM JRD*, 2002.
[9] W. Kim, "A 1.1V 16Gb DDR5 DRAM with Probabilistic-Aggressor Tracking, Refresh-Management Functionality, Per-Row Hammer Tracking, a Multi-Step Precharge, and Core-Bias Modulation for Security and Reliability Enhancement," in *ISSCC*, 2023.
[10] J.-H. Ahn *et al.*, "Adaptive self refresh scheme for battery operated high-density mobile DRAM applications," in *ASSCC*, 2006.
[11] T. Ohsawa *et al.*, "Optimizing the DRAM Refresh Count for Merged DRAM/Logic LSIs," in *ISLPED*, 1998.
[12] J. Kim and M. C. Papaefthymiou, "Dynamic memory design for low data-retention power," in *PATMOS*, 2000.
[13] R. Venkatesan *et al.*, "Retention-Aware Placement in DRAM (RAPID): Software Methods for Quasi-Non-Volatile DRAM," in *HPCA*, 2006.
[14] K. Yanagisawa, "Semiconductor Memory," 1988, U.S. Patent 4,736,344.
[15] M. Patel *et al.*, "Understanding and Modeling On-die Error Correction in Modern DRAM: An Experimental Study using Real Devices," in *DSN*, 2019.
[16] M. Patel *et al.*, "Bit-Exact ECC Recovery (BEER): Determining DRAM On-Die ECC Functions by Exploiting DRAM Data Retention Characteristics," in *MICRO*, 2020.
[17] M. Patel *et al.*, "HARP: Practically and Effectively Identifying Uncorrectable Errors in Main Memory Chips That Use On-Die ECC," in *MICRO*, 2021.
[18] S. Khan, "The Efficacy of Error Mitigation Techniques for DRAM Retention Failures: A Comparative Experimental Study," in *SIGMETRICS*, 2014.
[19] M. K. Qureshi *et al.*, "AVATAR: A Variable-Retention-Time (VRT) Aware Refresh for DRAM Systems," in *DSN*, 2015.
[20] S. Khan *et al.*, "PARBOR: An Efficient System-Level Technique to Detect Data Dependent Failures in DRAM," in *DSN*, 2016.
[21] S. Khan *et al.*, "A Case for Memory Content-Based Detection and Mitigation of Data-Dependent Failures in DRAM," *CAL*, 2016.
[22] S. Khan *et al.*, "Detecting and Mitigating Data-Dependent DRAM Failures by Exploiting Current Memory Content," in *MICRO*, 2017.
[23] M. Patel, "The Reach Profiler (REAPER): Enabling the Mitigation of DRAM Retention Failures via Profiling at Aggressive Conditions," in *ISCA*, 2017.
[24] O. Mutlu, "Memory Scaling: A Systems Architecture Perspective," *IMW*, 2013.
[25] O. Mutlu and L. Subramanian, "Research Problems and Opportunities in Memory Systems," *SUPERFRI*, 2014.
[26] Y. Kim *et al.*, "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors," in *ISCA*, 2014.
[27] J. S. Kim *et al.*, "Revisiting RowHammer: An Experimental Analysis of Modern DRAM Devices and Mitigation Techniques," in *ISCA*, 2020.
[28] P. Frigo *et al.*, "TRRespass: Exploiting the Many Sides of Target Row Refresh," in *S&P*, 2020.
[29] H. Hassan *et al.*, "Uncovering In-DRAM RowHammer Protection Mechanisms: A New Methodology, Custom RowHammer Patterns, and Implications," in *MICRO*, 2021.
[30] A. G. Yağlıkçı *et al.*, "BlockHammer: Preventing RowHammer at Low Cost by Blacklisting Rapidly-Accessed DRAM Rows," in *HPCA*, 2021.
[31] A. G. Yağlıkçı *et al.*, "HiRA: Hidden Row Activation for Reducing Refresh Latency of Off-the-Shelf DRAM Chips," in *MICRO*, 2022.
[32] A. G. Yağlıkçı *et al.*, "Understanding RowHammer Under Reduced Wordline Voltage: An Experimental Study Using Real DRAM Devices," in *DSN*, 2022.
[33] A. Olgun *et al.*, "An Experimental Analysis of RowHammer in HBM2 DRAM Chips," in *DSN Disrupt*, 2023.
[34] H. Luo *et al.*, "RowPress: Amplifying Read Disturbance in Modern DRAM Chips," in *ISCA*, 2023.
[35] M. Farmani *et al.*, "RHAT: Efficient RowHammer-Aware Test for Modern DRAM Modules," in *ETS*, 2021.
[36] O. Mutlu and J. S. Kim, "RowHammer: A Retrospective," *IEEE TCAD*, 2019.
[37] O. Mutlu *et al.*, "Fundamentally Understanding and Solving RowHammer," in *ASP-DAC*, 2023.
[38] D. Lee *et al.*, "Adaptive-Latency DRAM: Optimizing DRAM Timing for the Common-Case," in *HPCA*, 2015.
[39] K. K. Chang *et al.*, "Understanding Latency Variation in Modern DRAM Chips: Experimental Characterization, Analysis, and Optimization," in *SIGMETRICS*, 2016.
[40] D. Lee *et al.*, "Design-induced Latency Variation in Modern DRAM Chips: Characterization, Analysis, and Latency Reduction Mechanisms," *POMACS*, 2017.
[41] J. Kim *et al.*, "Solar-DRAM: Reducing DRAM Access Latency by Exploiting the Variation in Local Bitlines," in *ICCD*, 2018.
[42] K. Chang *et al.*, "Understanding Reduced-Voltage Operation in Modern DRAM Devices: Experimental Characterization, Analysis, and Mechanisms," in *SIGMETRICS*, 2017.
[43] S. Ghose *et al.*, "What Your DRAM Power Models Are Not Telling You: Lessons from a Detailed Experimental Study," in *SIGMETRICS*, 2018.
[44] V. Seshadri *et al.*, "Fast Bulk Bitwise AND and OR in DRAM," *CAL*, 2015.
[45] V. Seshadri *et al.*, "Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology," in *MICRO*, 2017.
[46] F. Gao *et al.*, "DRAMA: An Approach for Accelerating Irregular Graph Computation in Memory Systems," in *MICRO*, 2019.
[47] F. Gao *et al.*, "PIDRAM: A Holistic End-to-end FPGA-based Framework for Processing-in-DRAM," *TACO*, 2023.
[48] F. Gao *et al.*, "CompactDRAM: In-Memory Compute Using Off-the-Shelf DRAMs," in *MICRO*, 2019.
[49] F. Gao *et al.*, "FracDRAM: Fractional Values in Off-the-Shelf DRAM," in *MICRO*, 2022.
[50] J. S. Kim *et al.*, "The DRAM Latency PUF: Quickly Evaluating Physical Unclonable Functions by Exploiting the Latency-Reliability Tradeoff in Modern Commodity DRAM Devices," in *HPCA*, 2018.
[51] J. Kim, "D-RaNGe: Using Commodity DRAM Devices to Generate True Random Numbers with Low Latency and High Throughput," in *HPCA*, 2019.
[52] A. Olgun *et al.*, "QUAC-TRNG: High-Throughput True Random Number Generation Using Quadruple Row Activation in Commodity DRAMs," in *ISCA*, 2021.

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

Onur Mutlu
ETH Zürich

***Abstract*—Our ISCA 2014 paper [1] provided the first scientific and detailed characterization, analysis, and real-system demonstration of what is now popularly known as the RowHammer phenomenon (or vulnerability) in modern commodity DRAM chips, which are used as main memory in almost all modern computing systems. It experimentally demonstrated that more than 80% of all DRAM modules we tested from the three major DRAM vendors were vulnerable to the RowHammer read disturbance phenomenon: one can predictably induce bitflips (i.e., data corruption) in real DRAM modules by repeatedly accessing a DRAM row and thus causing electrical disturbance to physically nearby rows. We showed that a simple unprivileged user-level program induced RowHammer bitflips in multiple real systems and suggested that a security attack can be built using this proof-of-concept to hijack control of the system or cause other harm. To solve the RowHammer problem, our paper examined seven different approaches (including a novel probabilistic approach that has very low cost), some of which influenced or were adopted in different industrial products.**

Many later works from various research communities examined RowHammer, building real security attacks, proposing new defenses, further analyzing the problem at various (e.g., device/circuit, architecture, and system) levels, and exploiting RowHammer for various purposes (e.g., to reverse-engineer DRAM chips). Industry has worked to mitigate the problem, changing both memory controllers and DRAM standards/chips. Two major DRAM vendors finally wrote papers on the topic in 2023, describing their current approaches to mitigate RowHammer. Research & development on RowHammer in both academia & industry continues to be very active and fascinating.

This short retrospective provides a brief analysis of our ISCA 2014 paper and its impact. We describe the circumstances that led to our paper, mention its influence on later works and products, describe the mindset change we believe it has helped enable in hardware security, and discuss our predictions for future.

I. Background and Circumstances

Our stumbling on the RowHammer problem and creation of its first scientific analysis happened as a result of a confluence of multiple factors. First, my group was working on DRAM technology scaling issues since late 2010. We were very interested in failure mechanisms that appear or worsen due to aggressive technology scaling. To study such issues (e.g., data retention errors [2]), we built an FPGA-based DRAM testing infrastructure [2] between 2011-2012, which we later open sourced as SoftMC [3, 4] and DRAM Bender [5, 6]. Second, around the same timeframe, we were investigating similar technology scaling issues in flash memory using real NAND flash chips [7, 8]. We knew read disturbance errors were significant in NAND flash memory [7–11] and were very interested in how prevalent they were in DRAM. Third, we were collaborating with Intel (e.g., [9]) to understand and solve DRAM technology scaling problems and build our DRAM infrastructure. Three of my students and I spent the summer of 2012 at Intel to work closely with our collaborators (two are co-authors): during this time, we finalized the calibration and stabilization of our infrastructure and had significant technical discussions and experimentation on DRAM scaling problems.

Although there was awareness of the RowHammer problem in industry in 2012 (see Footnote 1 in [1]), there was no comprehensive experimental analysis and detailed real-system demonstration of it. We believed it was critical to provide a rigorous scientific analysis using a wide variety of DRAM chips and scientifically establish major characteristics and prevalence of RowHammer. Hence, in the summer of 2012, we set out to use our DRAM testing infrastructure to analyze RowHammer. Our initial results showed how widespread the read disturbance problem was across the (at the time) recent DRAM chips we tested, so we studied the problem comprehensively and developed many solutions to it. The resulting paper was submitted to MICRO in May 2013 but was rejected. We strengthened the results, especially of the mitigation mechanisms and the number of tested chips, and made the analysis more comprehensive before it was accepted to ISCA 2014 (2 of the 6 reviewers still rejected it for interesting reasons).

II. Major Contribution and Influence

The major contribution of our paper is the exposure and detailed analysis of a fundamental hardware failure mechanism that breaks memory isolation in real systems and thus has huge implications on system reliability, security, and safety. Our paper is a comprehensive study of a major DRAM technology scaling problem, RowHammer, including its first scientific analysis, experimental characterization, real system demonstration, and solutions with their evaluation. To our knowledge, RowHammer is the first example of a hardware failure mechanism that creates a significant and widespread system security vulnerability [12–15], as our ISCA 2014 paper suggested.

Our work has had large influence on both industry & academia. Individual follow-on works are many to list here; we refer the reader to longer invited retrospectives we wrote [12–14]. We give major examples of influence, focusing on RowHammer's effect on the collective mindset of security research and major industry milestones related to RowHammer.

RowHammer Attacks & Mindset Shift in Hardware Security. Our demonstration that one can easily and predictably induce bitflips in commodity DRAM chips using a real user-level program enabled a major mindset shift in hardware security. It showed that general-purpose hardware is fallible in a very widespread manner and its problems are exploitable. Tens of works (see [13, 14]) built directly on our work to exploit RowHammer bitflips to develop many attacks that compromise system integrity and confidentiality, starting from the first RowHammer exploit by Google Project Zero in 2015 [16, 17] to recent works in 2022-2023 (e.g., [18, 19]). These attacks showed increasingly sophisticated ways by which an unprivileged attacker can exploit RowHammer bitflips to circumvent memory protection and gain complete control of a system (e.g., [16, 20–28]), gain access to confidential data (e.g., [18, 19, 29]), or maliciously destroy the safety and accuracy of a system, e.g., an otherwise accurate machine learning inference engine (e.g., [30, 31]). The mindset enabled by RowHammer bitflips caused a renewed interest in hardware security research, enticing many researchers to deeply understand hardware's inner workings and find new vulnerabilities. Thus, hardware security issues have become mainstream discussion in top security & architecture venues, some having sessions entitled RowHammer.

RowHammer Defenses. Tens of works proposed mitigations against RowHammer, some of which were inspired by the solutions we discussed in our ISCA 2014 paper. To date, the search for more efficient and low-cost RowHammer solutions continues. We refer the reader to our prior overview papers [13, 14, 32] and more recent works in 2023 (e.g., [33–35]).

RowHammer Analyses. Our paper initiated works at both architectural & circuit/device-levels to better understand RowHammer and reverse-engineer DRAM chips, to develop better models, defenses, and attacks (see [13, 14]). Our ISCA'20 work [36] revisited RowHammer, comprehensively analyzed of 1580 DRAM chips of three different types from at least two generations, showing that RowHammer has gotten much worse with technology scaling & existing solutions are not effective at future vulnerability levels.

Industry Reaction: Attacks, Analyses, and Mitigations. Folks developing industrial memory testing programs immediately included RowHammer tests, e.g., in memtest86 [37], citing our work. Industry needed to immediately protect RowHammer-vulnerable chips already in the field, so almost all system vendors increased refresh rates; a solution we examined in our paper and deemed costly for performance and energy, yet it was the only practical lever that could be used in the field. Apple publicly acknowledged our work in their security release [38] that announced higher refresh rates to mitigate RowHammer. Intel designed memory controllers that performed probabilistic activations (i.e., pTRR [39, 40]), similar to our PARA solution [1]. DRAM vendors modified the DRAM standard to introduce TRR (target row refresh) mechanisms [39] and claimed their new DDR4 chips to be RowHammer-free [39, 41]. This bold claim was later refuted by our TRRespass work [39] in 2020, which introduced the many-sided RowHammer attack to circumvent internal protection mechanisms added to the DRAM chips. Our later work, Uncovering TRR [41] showed that one can almost completely reverse-engineer and thus easily bypass RowHammer mitigations employed in all tested DRAM chips, i.e., RowHammer solutions in DRAM chips are broken. The analysis done by our two major works in 2020 [36, 39] caused the industry to reorganize the RowHammer task group at JEDEC, which produced two white papers on mitigating RowHammer [42, 43]. Nine years after our paper, in 2023, two major DRAM vendors, SK Hynix and Samsung, finally wrote papers [44, 45] on the RowHammer problem, describing their solutions. Several of these industry solutions build on the probabilistic & access-counter-based solution approaches our ISCA 2014 paper introduced.

Major Internet and cloud systems companies also took a deep interest in RowHammer as it can greatly impact their system security, dependability, and availability. Multiple works from Google, e.g., by Google Project Zero in 2015 [16, 17] and Half Double in 2021-2022 [46] directly built on our paper to demonstrate attacks in real systems. Researchers from Microsoft have developed deeper analyses of RowHammer [47], along with new RowHammer attacks [48] and defenses (e.g., [48–51]).

III. Summary and Future Outlook

Since 2012-2014, RowHammer vulnerability has become much worse due to technology scaling: without mitigation, one can now induce RowHammer bitflips with orders of magnitude smaller number of activations (e.g., ∼10K) and cause much higher rates of errors in cutting-edge DRAM chips [36, 41]. Sophisticated attacks are continuously developed to circumvent the mitigations employed in real DRAM chips. Fortunately, we have also come a long way in further understanding and better mitigating the RowHammer vulnerability. The industry is now (hopefully) fully aware of the importance of the problem and of avoiding bitflips. Unfortunately, an efficient and complete/100%-secure solution is not found yet. The solution space poses a rich area of tradeoffs in terms of security, performance, power/energy, cost/complexity. All solutions forego some desirable properties in favor of others. As such, a critical direction for the future is to find solutions superior to what we have today. We believe system-DRAM cooperation [14, 52] will be important to enabling complete solutions. We also believe it is critical to deeply understand the properties of RowHammer under many different conditions so that we can develop effective solutions that work under all circumstances. Unfortunately, we do not yet fully understand many facets of RowHammer (see [14, 53–55]).

DRAM technology scaling will continue to create problems that will exacerbate the bitflips and the resulting robustness (i.e., safety/security/reliability) problems. Our ISCA' 2023 paper on RowPress [55] provides the first scientific and detailed characterization, analysis, and real-system demonstration of yet another read disturbance mechanism in DRAM. What other fascinating problems will we see and can we completely solve them efficiently? Will we ever be free of bitflips at the system and application levels?

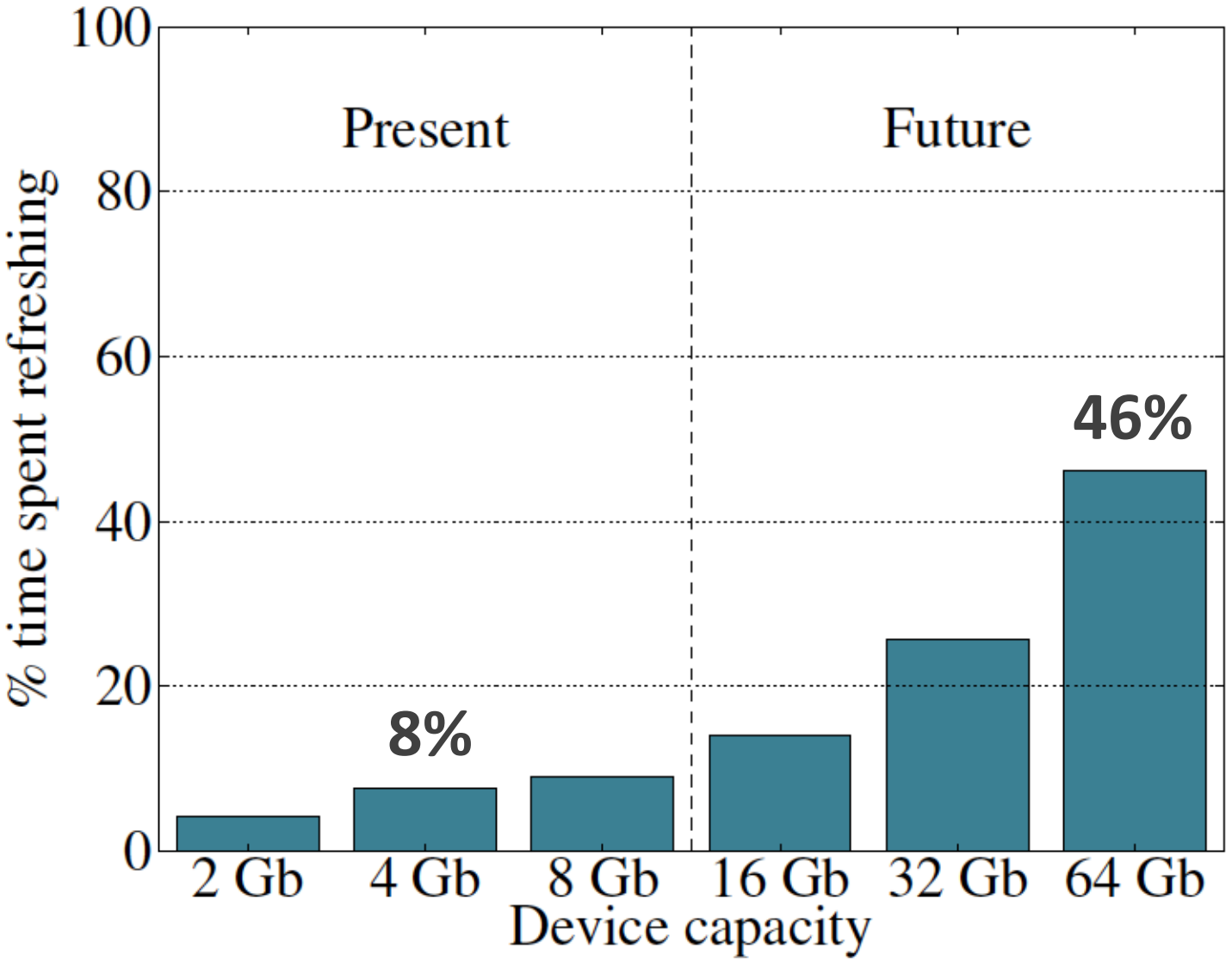
References

[1] Y. Kim *et al.*, "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors," in *ISCA*, 2014.
[2] J. Liu *et al.*, "An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms," *ISCA*, 2013.
[3] H. Hassan *et al.*, "SoftMC: A Flexible and Practical Open-Source Infrastructure for Enabling Experimental DRAM Studies," in *HPCA*, 2017.
[4] SoftMC Source Code, https://github.com/CMU-SAFARI/SoftMC/
[5] A. Olgun *et al.*, "DRAM Bender: An extensible and versatile FPGA-based Infrastructure to Easily Test State-of-the-art DRAM Chips," *TCAD*, 2023.
[6] "DRAM Bender," https://github.com/CMU-SAFARI/DRAM-Bender
[7] Y. Cai *et al.*, "Error Patterns in MLC NAND Flash Memory: Measurement, Characterization, and Analysis," in *DATE*, 2012.
[8] Y. Cai *et al.*, "Error Analysis and Retention-Aware Error Management for NAND Flash Memory," *ITJ*, 2013.
[9] Y. Cai *et al.*, "Program Interference in MLC NAND Flash Memory: Characterization, Modeling, and Mitigation," in *ICCD*, 2013.
[10] Y. Cai *et al.*, "Read Disturb Errors in MLC NAND Flash Memory: Characterization, Mitigation, and Recovery," in *DSN*, 2015.
[11] Y. Cai *et al.*, "Error Characterization, Mitigation, and Recovery in Flash Memory Based Solid-State Drives," *Proc. IEEE*, 2017.
[12] O. Mutlu, "The RowHammer Problem and Other Issues we may Face as Memory Becomes Denser," in *DATE*, 2017.
[13] O. Mutlu and J. Kim, "RowHammer: A Retrospective," *IEEE TCAD Special Issue on Top Picks in Hardware and Embedded Security*, 2019.
[14] O. Mutlu *et al.*, "Fundamentally Understanding and Solving RowHammer," in *ASP-DAC*, 2023.
[15] T. Dullien, "Security, Moore's Law, and the Anomaly of Cheap Complexity," in *CCDCOE*, 2018, https://www.youtube.com/watch?v=q086toLaAIA
[16] M. Seaborn and T. Dullien, "Exploiting the DRAM rowhammer Bug to Gain Kernel Privileges," http://googleprojectzero.blogspot.com.tr/2015/03/exploiting-dram-rowhammer-bug-to-gain.html, 2015.
[17] M. Seaborn and T. Dullien, "Exploiting the DRAM Rowhammer Bug to Gain Kernel Privileges," *Black Hat*, 2015.
[18] A. S. Rakin *et al.*, "DeepSteal: Advanced Model Extractions Leveraging Efficient Weight Stealing in Memories," in *S&P*, 2022.
[19] K. Mus *et al.*, "Jolt: Recovering TLS Signing Keys via Rowhammer Faults," in *S&P*, 2023.
[20] D. Gruss *et al.*, "Rowhammer.js: A Remote Software-Induced Fault Attack in Javascript," in *DIMVA*, 2016.
[21] V. van der Veen *et al.*, "Drammer: Deterministic Rowhammer Attacks on Mobile Platforms," in *CCS*, 2016.
[22] Y. Xiao *et al.*, "One Bit Flips, One Cloud Flops: Cross-VM Row Hammer Attacks and Privilege Escalation," in *USENIX Security*, 2016.
[23] K. Razavi *et al.*, "Flip Feng Shui: Hammering a Needle in the Software Stack," *USENIX Security*, 2016.
[24] A. Tatar *et al.*, "Throwhammer: Rowhammer Attacks Over the Network and Defenses," in *USENIX ATC*, 2018.
[25] M. Lipp *et al.*, "Nethammer: Inducing Rowhammer Faults Through Network Requests," arXiv:1805.04956, 2018.
[26] L. Cojocar *et al.*, "Exploiting Correcting Codes: On the Effectiveness of ECC Memory Against Rowhammer Attacks," in *S&P*, 2019.
[27] F. de Ridder *et al.*, "SMASH: Synchronized Many-Sided Rowhammer Attacks from JavaScript," in *USENIX Security*, 2021.
[28] P. Jattke *et al.*, "Blacksmith: Scalable Rowhammering in the Frequency Domain," in *S&P*, 2022.
[29] A. Kwong *et al.*, "RAMBleed: Reading Bits in Memory Without Accessing Them," in *S&P*, 2020.
[30] S. Hong *et al.*, "Terminal Brain Damage: Exposing the Graceless Degradation in Deep Neural Networks Under Hardware Fault Attacks," in *SS*, 2019.
[31] F. Yao *et al.*, "Deephammer: Depleting the Intelligence of Deep Neural Networks Through Targeted Chain of Bit Flips," in *USENIX Security*, 2020.
[32] A. G. Yağlıkcı *et al.*, "BlockHammer: Preventing RowHammer at Low Cost by Blacklisting Rapidly-Accessed DRAM Rows," in *HPCA*, 2021.
[33] M. Marazzi *et al.*, "ProTRR: Principled yet Optimal In-DRAM Target Row Refresh," in *S&P*, 2023.
[34] M. Wi *et al.*, "SHADOW: Preventing Row Hammer in DRAM with Intra-Subarray Row Shuffling," in *HPCA*, IEEE, 2023.
[35] J. Juffinger *et al.*, "CSI: Rowhammer-Cryptographic Security and Integrity against Rowhammer (to appear)," in *S&P*, 2023.
[36] J. S. Kim *et al.*, "Revisiting RowHammer: An Experimental Analysis of Modern DRAM Devices and Mitigation Techniques," in *ISCA*, 2020.
[37] PassMark Software, "MemTest86: The Original Industry Standard Memory Diagnostic Utility," http://www.memtest86.com/#ounieshooting.htm, 2015.
[38] Apple Inc., "About the Security Content of Mac EFI Security Update 2015-001," https://support.apple.com/en-us/H1 204934, 2015.
[39] P. Frigo *et al.*, "TRRespass: Exploiting the Many Sides of Target Row Refresh," in *S&P*, 2020.
[40] M. Kaczmarski, "Thoughts on Intel Xeon E5-2600 v2 Product Family Performance Optimisation – Component Selection Guidelines," http://infobazy.edu.pl/Uploads/prezentacje/dUSz64-Kaczmarski-Optymalna.pdf, page 13, 2014.
[41] P. Hassan *et al.*, "Uncovering In-DRAM RowHammer Protection Mechanisms: A New Methodology, Custom RowHammer Patterns, and Implications," in *MICRO*, 2021.
[42] JEDEC, *JEP300-1: Near-Term DRAM Level RowHammer Mitigation*, 2021.
[43] JEDEC, *JEP301-1: System Level RowHammer Mitigation*, 2021.
[44] W. Kim, "A 1.1V 16Gb DDR5 DRAM with Probabilistic-Aggressor Tracking, Refresh-Management Functionality, Per-Row Hammer Tracking, a Multi-Step Precharge, and Core-Bias Modulation for Security and Reliability Enhancement," in *ISSCC*, 2023.
[45] S. Hong *et al.*, "HSAC: Low-Cost Rowhammer Mitigation Using In-DRAM Stochastic and Approximate Counting Algorithm," arXiv:2302.03591, 2023.
[46] A. Kogler *et al.*, "Half-Double: Hammering From the Next Row Over," in *USENIX Security*, 2022.
[47] L. Cojocar *et al.*, "Are We Susceptible to Rowhammer? An End-to-End Methodology for Cloud Providers," in *S&P*, 2020.
[48] K. Loughlin *et al.*, "MOESI-Prime: Preventing Coherence-Induced Hammering in Commodity Workloads," in *ISCA*, 2022.
[49] T. Bennett *et al.*, "Panopticon: A Complete In-DRAM Rowhammer Mitigation," in *DMSec*, 2021.
[50] K. Loughlin *et al.*, "Stop! Hammer Time: Rethinking Our Approach to Rowhammer Mitigations," in *HotOS*, 2021.
[51] S. Saroiu and A. Wolman, "How to Configure Row-Sampling-Based Rowhammer Defenses," *DRAMSec*, 2022.
[52] O. Mutlu, "Memory Scaling: A Systems Architecture Perspective," in *IMW*, 2013.
[53] L. Orosa, "A Deeper Look into RowHammer's Sensitivities: Experimental Analysis of Real DRAM Chips and Implications on Future Attacks and Defenses," in *MICRO*, 2021.
[54] A. G. Yağlıkcı *et al.*, "Understanding RowHammer Under Reduced Wordline Voltage: An Experimental Study Using Real DRAM Devices," in *DSN*, 2022.
[55] H. Luo *et al.*, "RowPress: Amplifying Read Disturbance in Modern DRAM Chips," in *ISCA*, 2023.

https://people.inf.ethz.ch/omutlu/pub/RowHammer_50YearsOfISCA-Retrospective_isca23.pdf

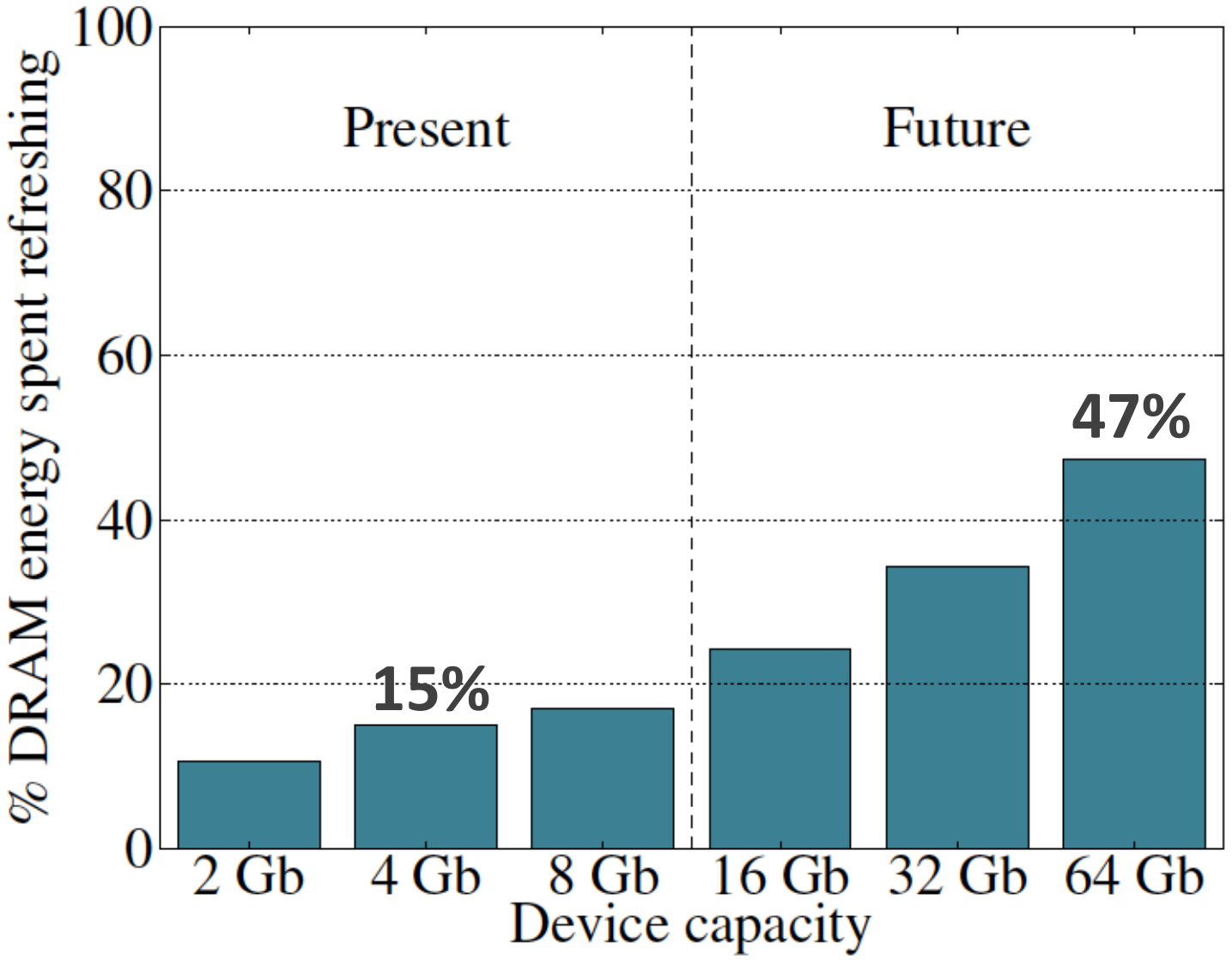
DRAM Refresh Overhead

Refresh Overhead: Performance



Liu et al., "RAIDR: Retention-Aware Intelligent DRAM Refresh," ISCA 2012.

Refresh Overhead: Energy

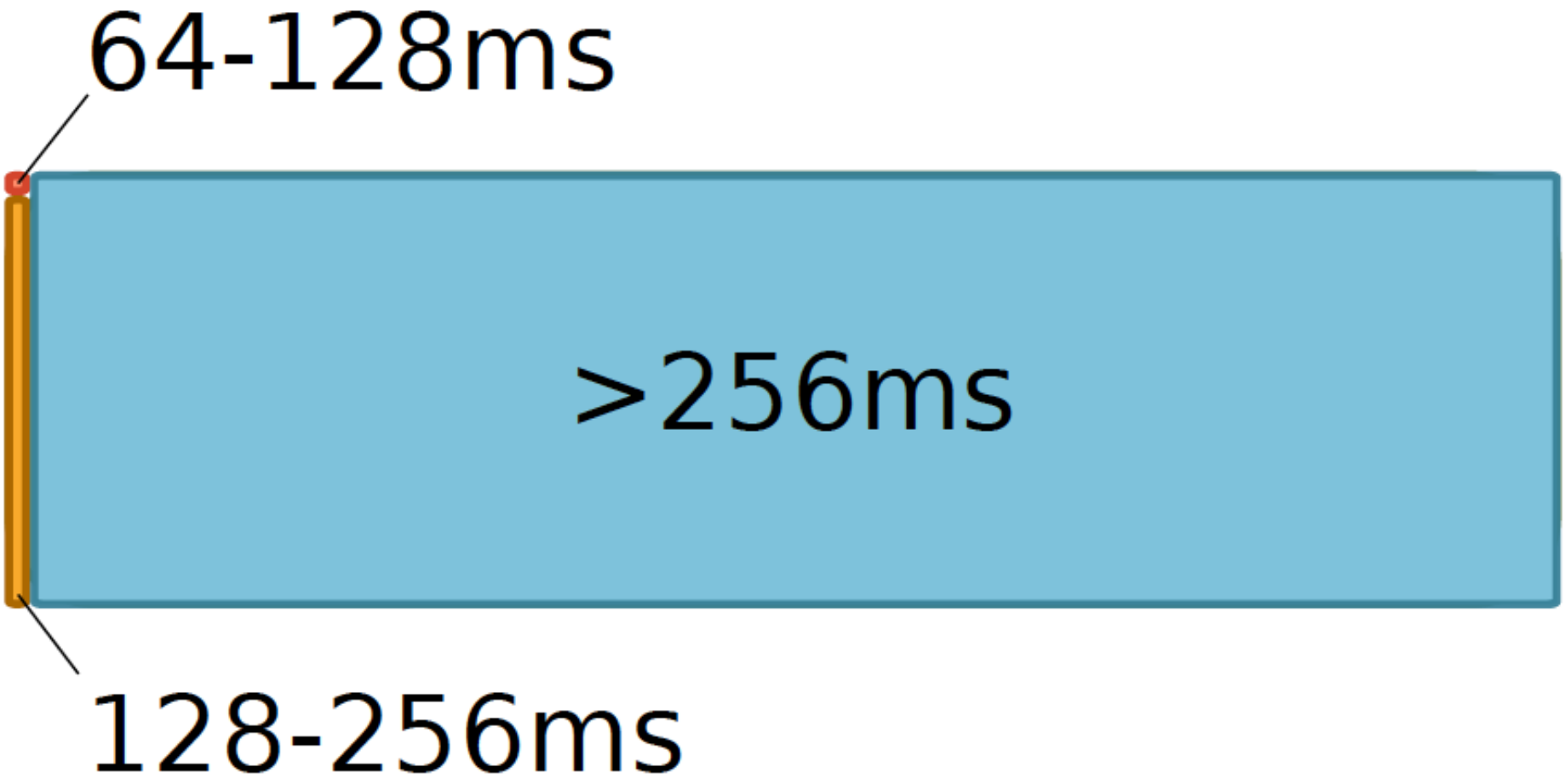


Liu et al., "RAIDR: Retention-Aware Intelligent DRAM Refresh," ISCA 2012.

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



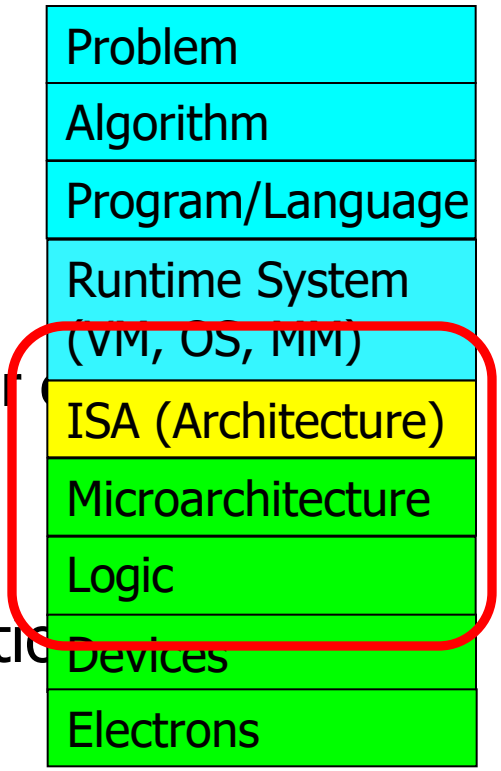
Liu et al., "RAIDR: Retention-Aware Intelligent DRAM Refresh," ISCA 2012.

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

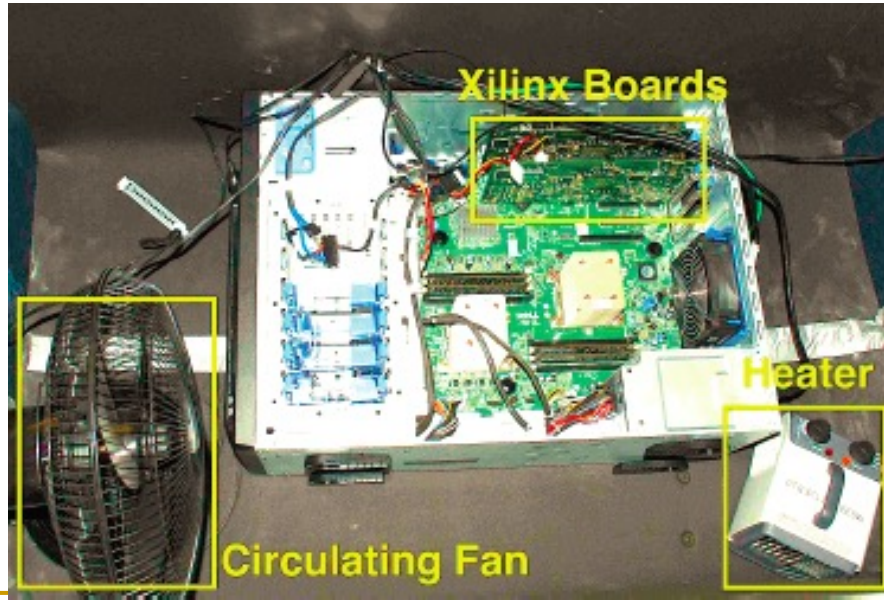
- Assume we know the retention time of each row exactly
- What can we do with this information?
- Who do we expose this information to?
- How much information do we expose?
  - Affects hardware/software overhead, power
    verification complexity, cost
- How do we determine this profile informatio
  - Also, who determines it?



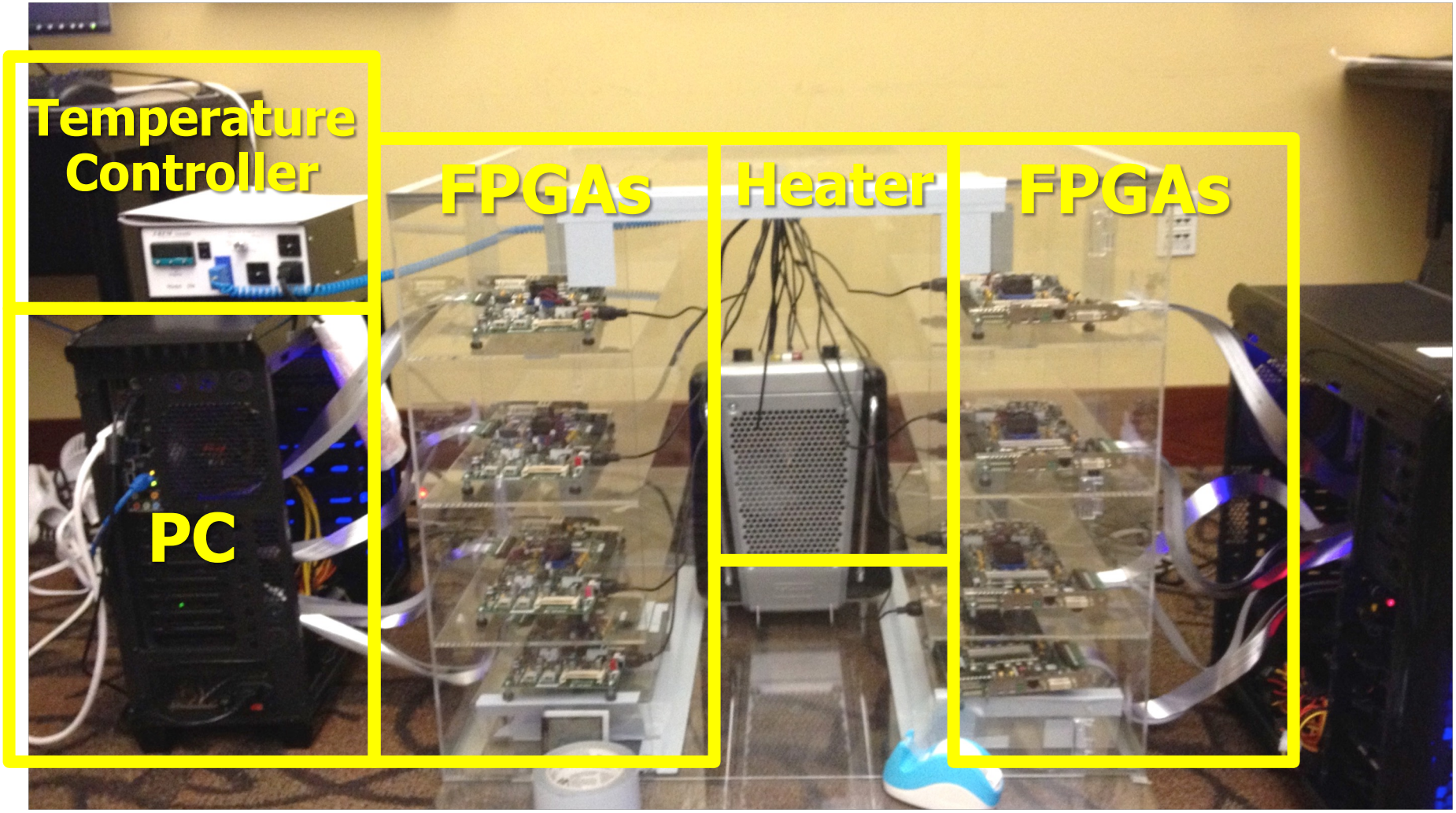
Experimental Infrastructure (DRAM)

Liu+, "An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms", ISCA 2013.

Khan+, "The Efficacy of Error Mitigation Techniques for DRAM Retention Failures: A Comparative Experimental Study," SIGMETRICS 2014.



Experimental Infrastructure (DRAM)



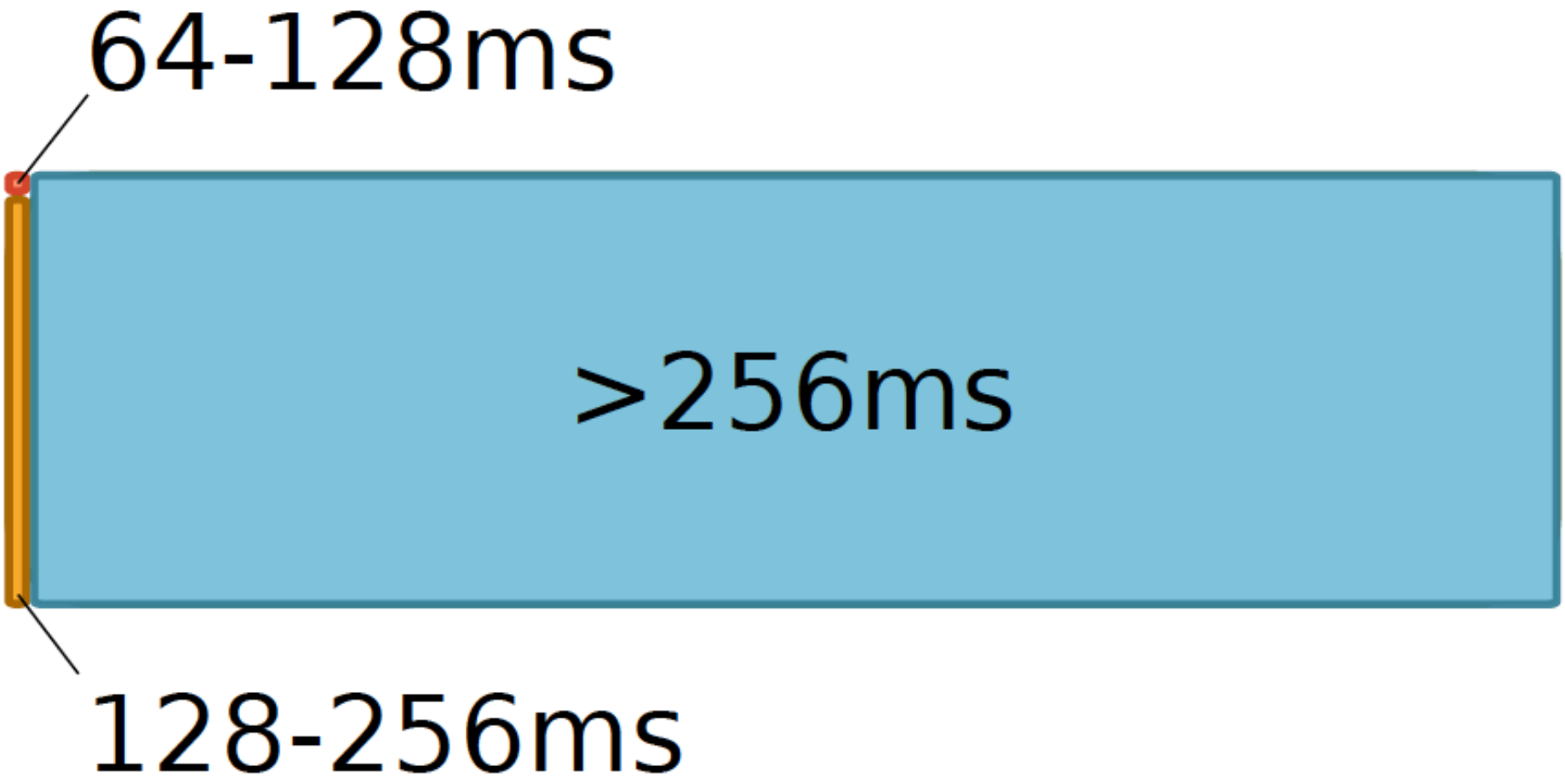
Kim+, "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors," ISCA 2014.

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

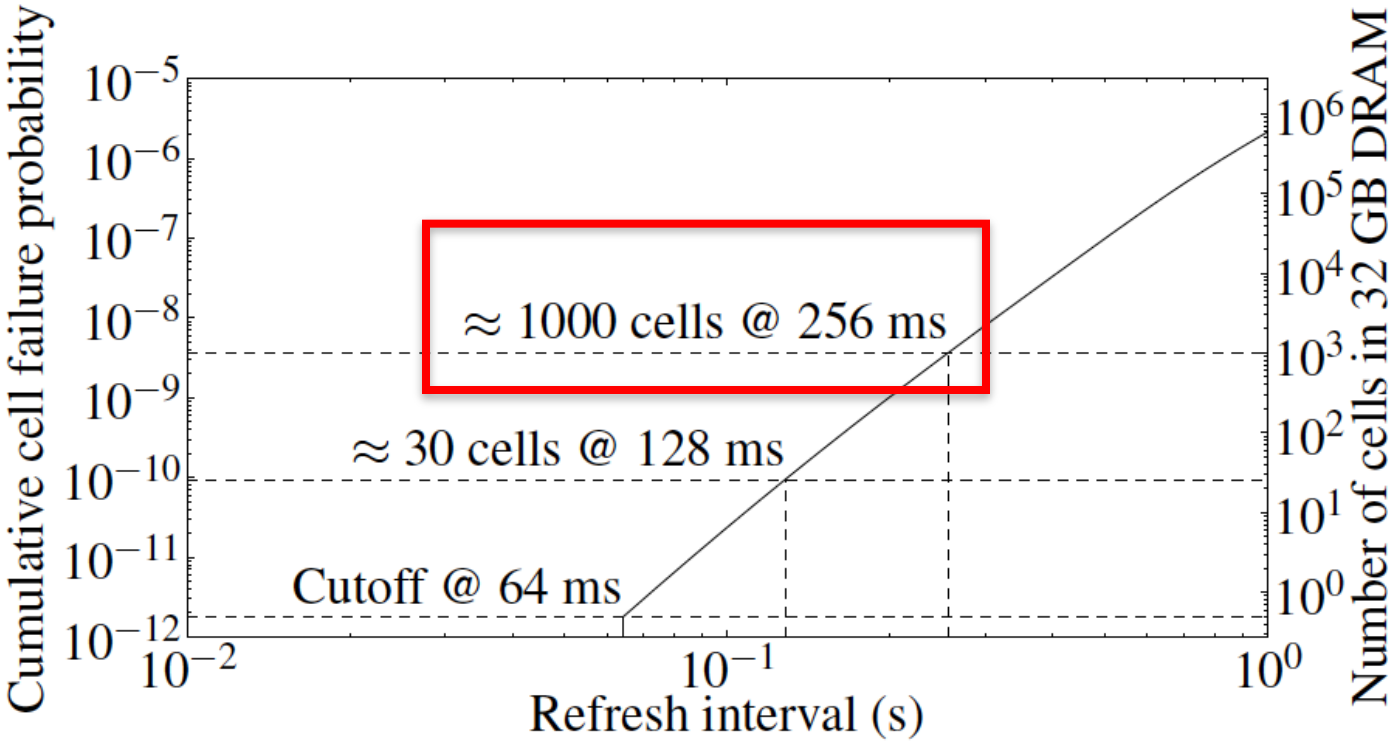
Underneath: Retention Time Profile of DRAM



Liu et al., "RAIDR: Retention-Aware Intelligent DRAM Refresh," ISCA 2012.

Retention Time of DRAM Rows

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



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

Liu et al., "RAIDR: Retention-Aware Intelligent DRAM Refresh," ISCA 2012.

RAIDR: Heterogeneous Refresh [ISCA'12]

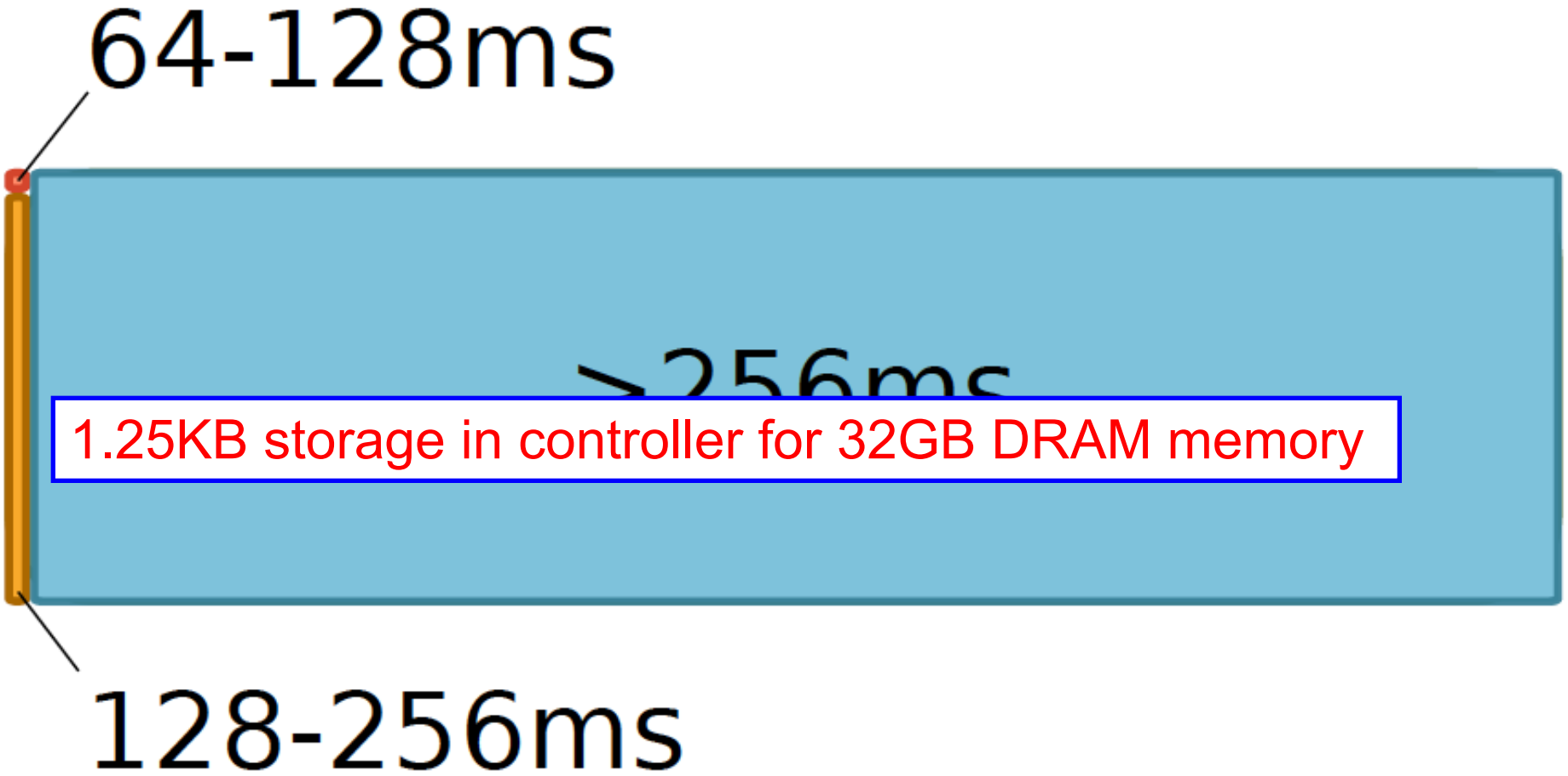
- Jamie Liu, Ben Jaiyen, Richard Veras, and Onur Mutlu, **"RAIDR: Retention-Aware Intelligent DRAM Refresh"** *Proceedings of the 39th International Symposium on Computer Architecture (**ISCA**)*, Portland, OR, June 2012. Slides (pdf) [Invited Retrospective at 50 Years of ISCA, 2023 (pdf)]
  Selected to the ISCA-50 25-Year Retrospective Issue covering 1996-2020 in 2023 (Retrospective (pdf) Full Issue).

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

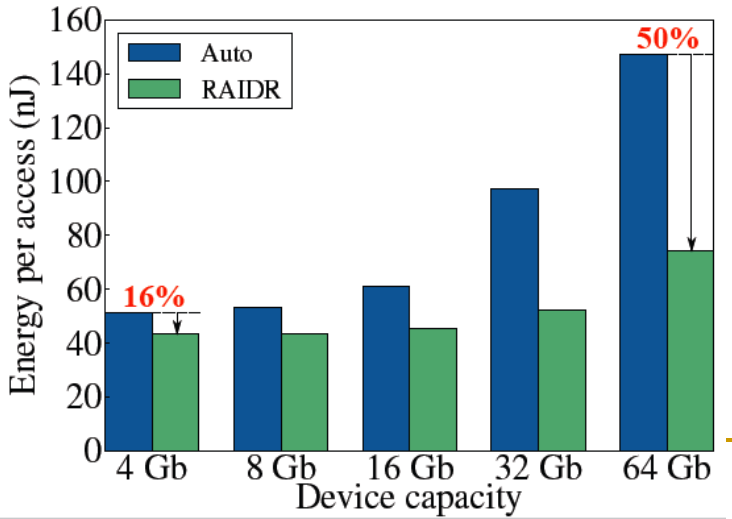


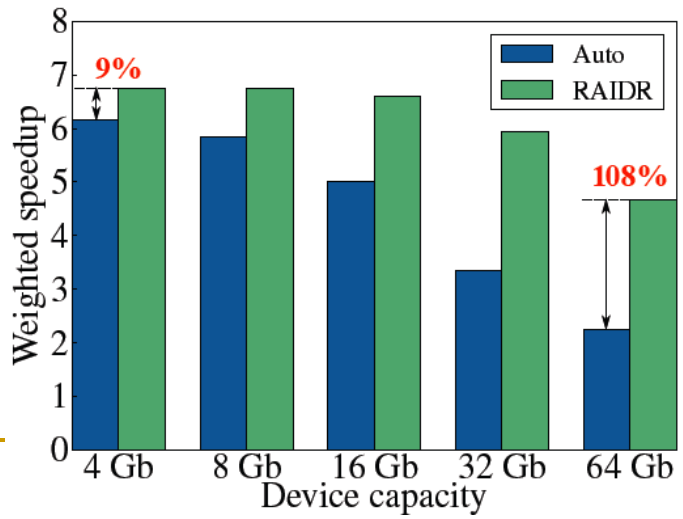
→ check the bins to determine refresh rate of a row

Liu et al., "RAIDR: Retention-Aware Intelligent DRAM Refresh," ISCA 2012.

RAIDR: Results and Takeaways

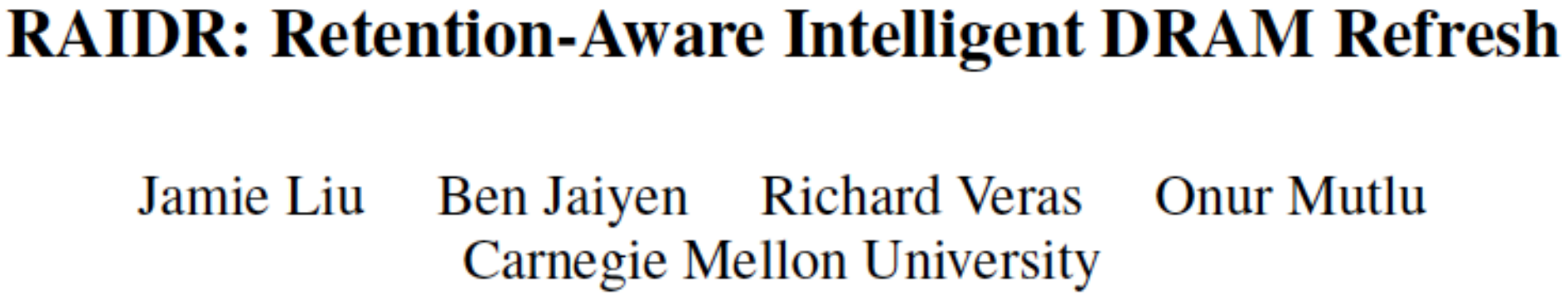
- System: 32GB DRAM, 8-core; Various workloads
- RAIDR hardware cost: 1.25 kB (2 Bloom filters)
- Refresh reduction: 74.6%
- Dynamic DRAM energy reduction: 16%
- Idle DRAM power reduction: 20%
- Performance improvement: 9%
- Benefits increase as DRAM scales in density





RAIDR: Heterogeneous 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)]
  Selected to the ISCA-50 25-Year Retrospective Issue covering 1996-2020 in 2023 (Retrospective (pdf) Full Issue).



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"
  *Proceedings of the 20th International Symposium on High-Performance Computer Architecture (**HPCA**)*, Orlando, FL, February 2014. Slides (pptx) (pdf)

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

Liu et al., "RAIDR: Retention-Aware Intelligent DRAM Refresh," ISCA 2012.

1. Profiling

To profile a row:

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

	Row 1	Row 2	Row 3
Initially	11111111...	11111111...	11111111...
After 64 ms	11111111...	11111111...	11111111...
After 128 ms	11011111...	11111111...	11111111...
	(64–128ms)		
After 256 ms		11111011...	11111111...
		(128–256ms)	(>256ms)

DRAM Retention Time Profiling

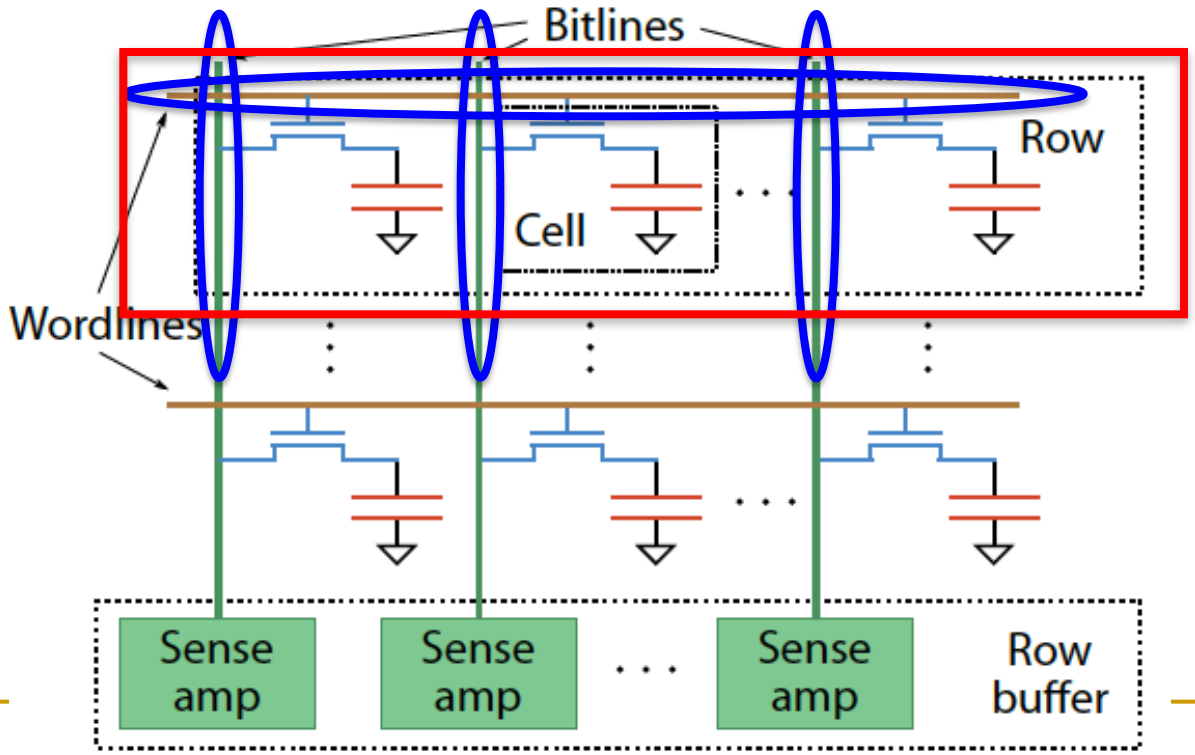
- Q: Is it really this easy?
- A: No…

Two Challenges to Retention Time Profiling

- Data Pattern Dependence (DPD) of retention time
- Variable Retention Time (VRT) phenomenon

Two Challenges to Retention Time Profiling

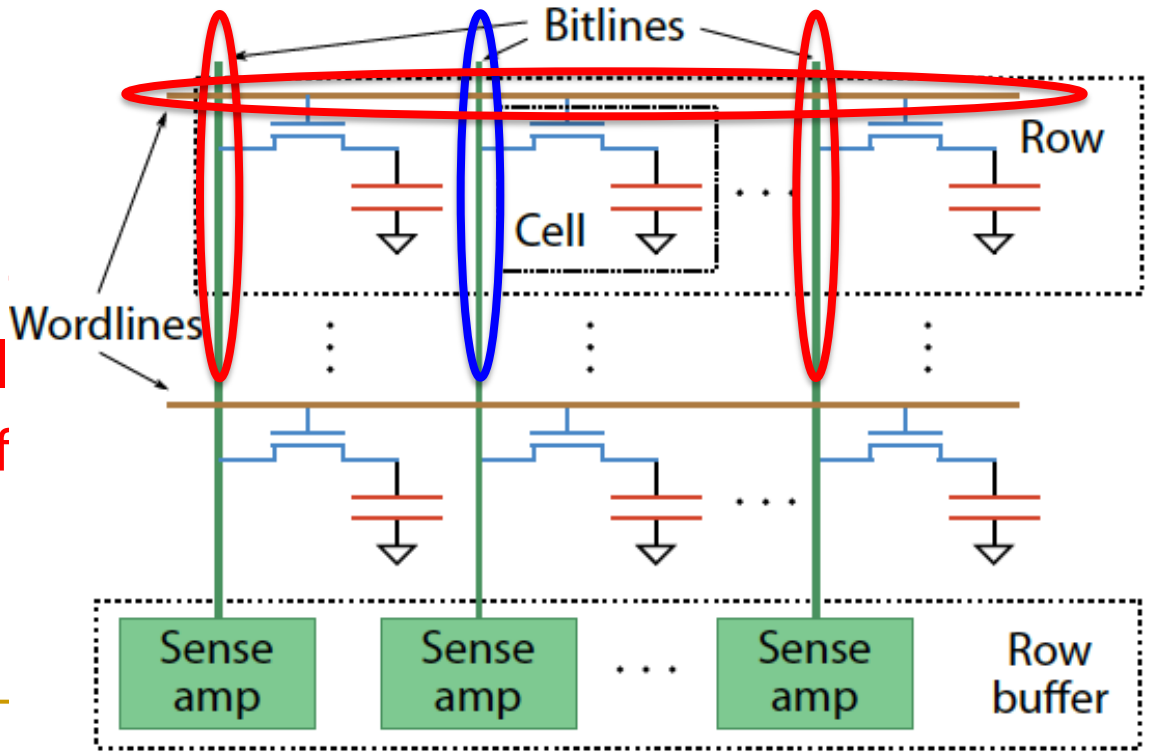
- Challenge 1: Data Pattern Dependence (DPD)
  - Retention time of a DRAM cell depends on its value and the values of cells nearby it
  - When a row is activated, all bitlines are perturbed simultaneously



Data Pattern Dependence

- 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
  nearby cel
  → need to f



tored in

etention time

DPD: Implications on Profiling Mechanisms

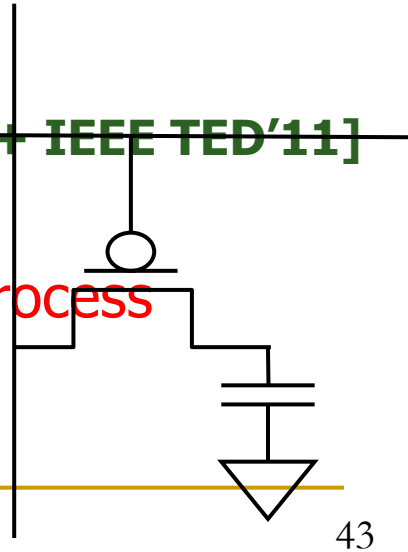
- Any retention time profiling mechanism must handle data pattern dependence of retention time
- Intuitive approach: Identify the data pattern that induces the worst-case retention time for a particular cell or device

- 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

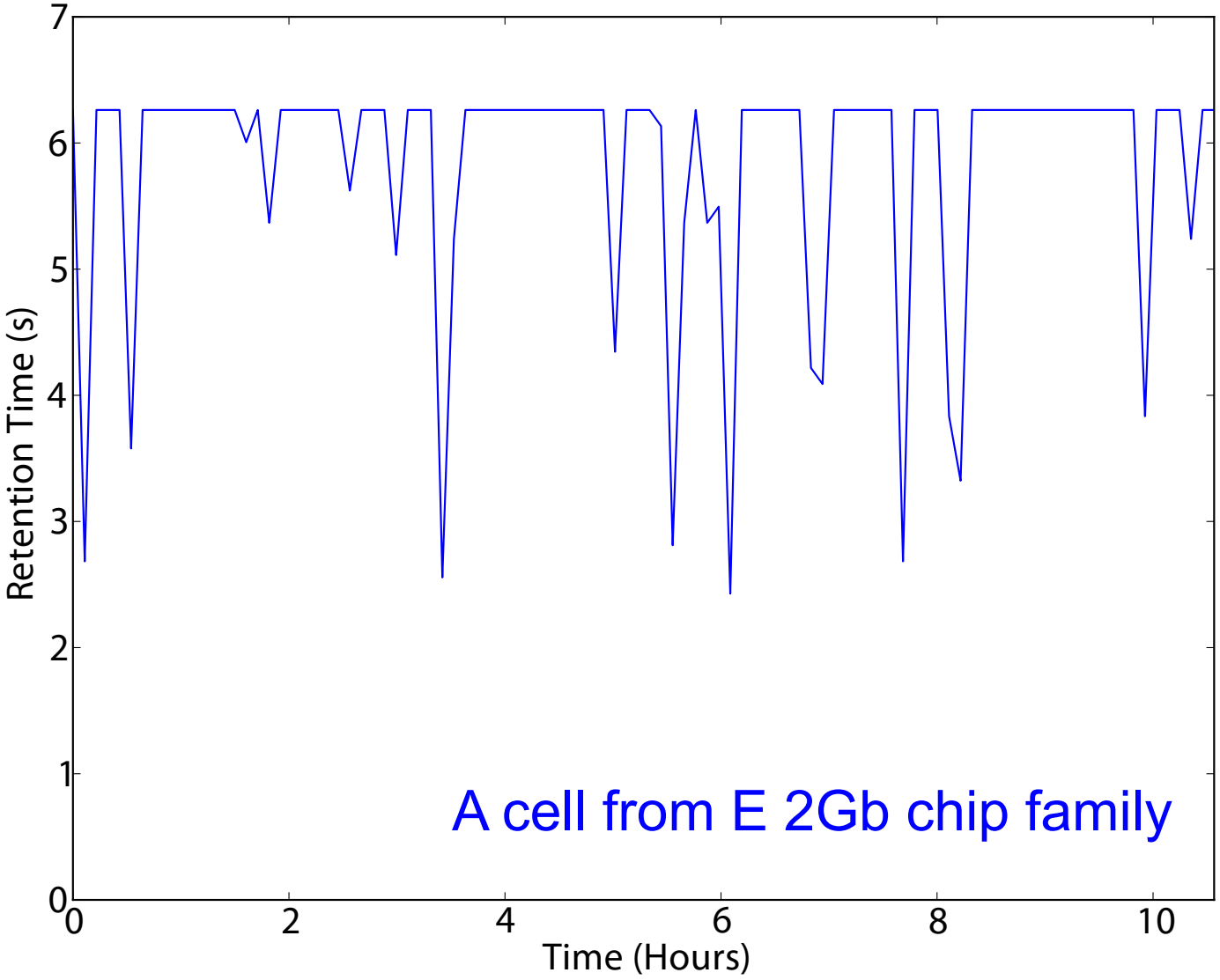
- Problem 2: Worst-case coupling noise is affected by non-obvious second order bitline coupling effects

Two Challenges to Retention Time Profiling

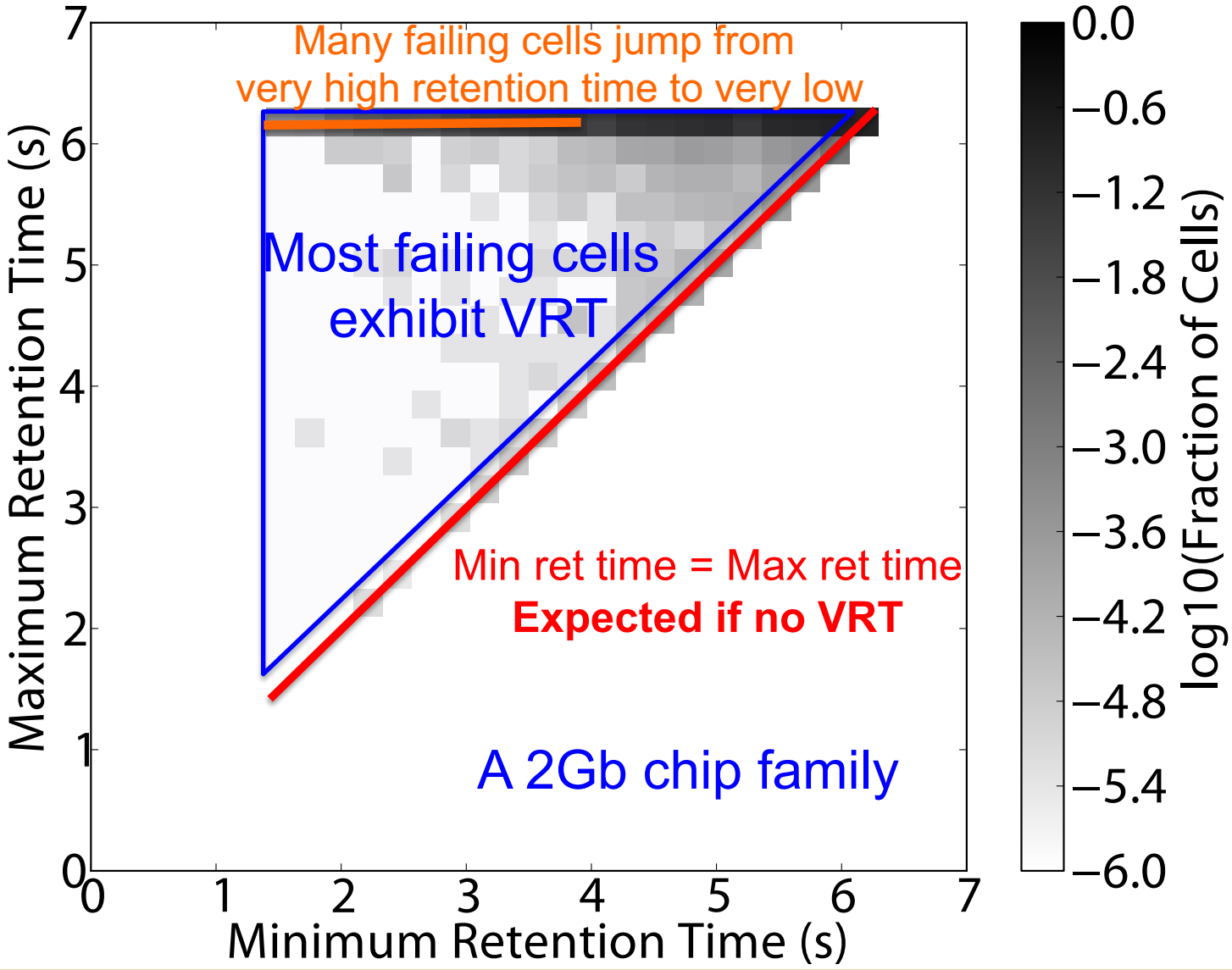
- Challenge 2: Variable Retention Time (VRT)
  - Retention time of a DRAM cell changes randomly over time
    - a cell alternates between multiple retention time states
  - Leakage current of a cell changes sporadically due to a charge trap in the gate oxide of the DRAM cell access transistor
  - When 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



Variable Retention Time



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)]
  Selected to the ISCA-50 25-Year Retrospective Issue covering 1996-2020 in 2023 (Retrospective (pdf) Full Issue).

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"
  *Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (**SIGMETRICS**)*, Austin, TX, June 2014. [Slides (pptx) (pdf)] [Poster (pptx) (pdf)] [Full data sets]

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

Samira Khan†⋆
samirakhan@cmu.edu

Donghyuk Lee†
donghyuk1@cmu.edu

Yoongu Kim†
yoongukim@cmu.edu

Alaa R. Alameldeen⋆
alaa.r.alameldeen@intel.com

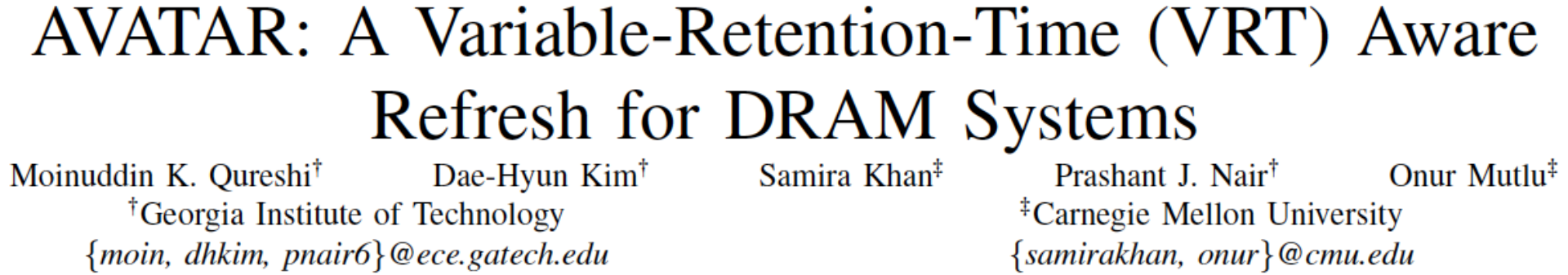
Chris Wilkerson⋆
chris.wilkerson@intel.com

Onur Mutlu†
onur@cmu.edu

†Carnegie Mellon University ⋆Intel Labs

Handling 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)]



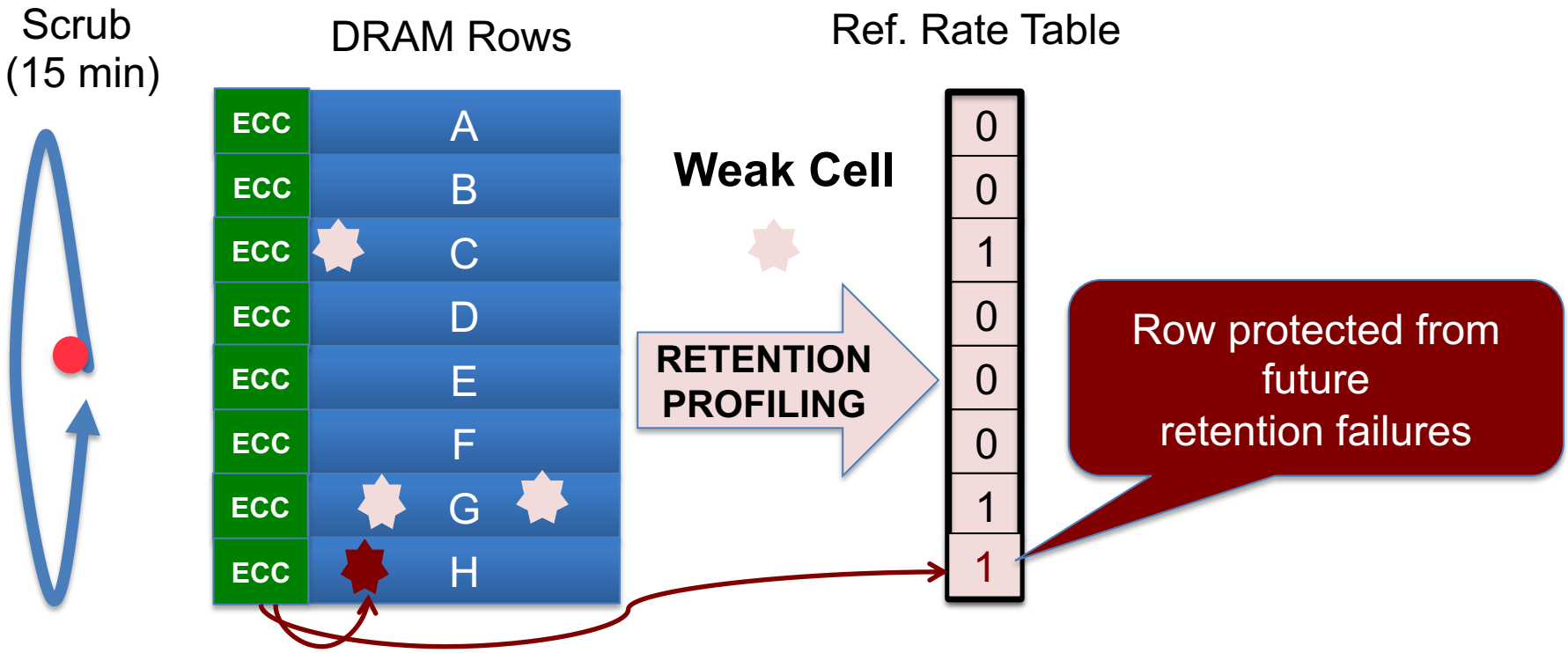
AVATAR: A Variable-Retention-Time (VRT) Aware Refresh for DRAM Systems

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

AVATAR

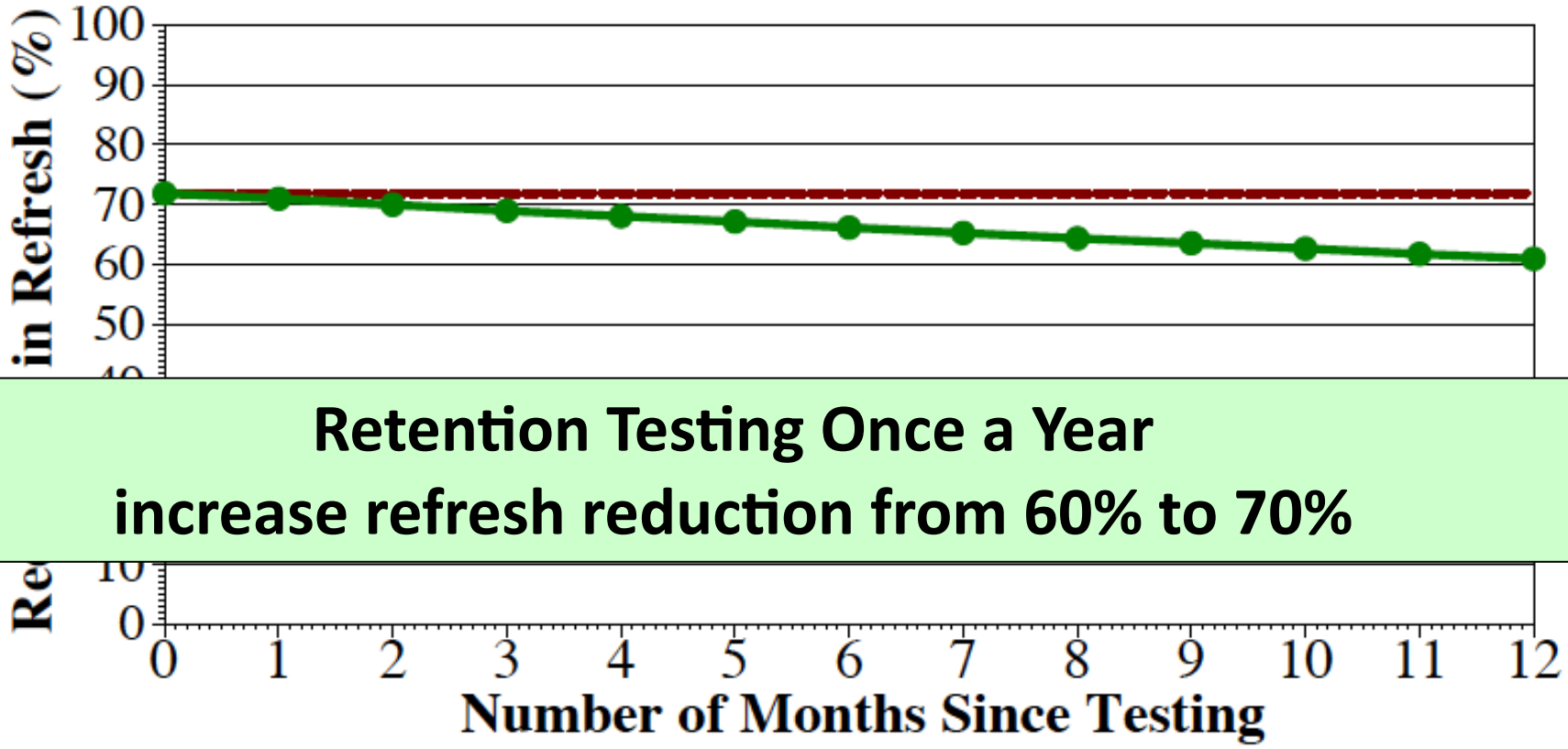
Insight: Avoid retention failures ➔ Upgrade row on ECC error

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



AVATAR mitigates VRT by increasing refresh rate on error

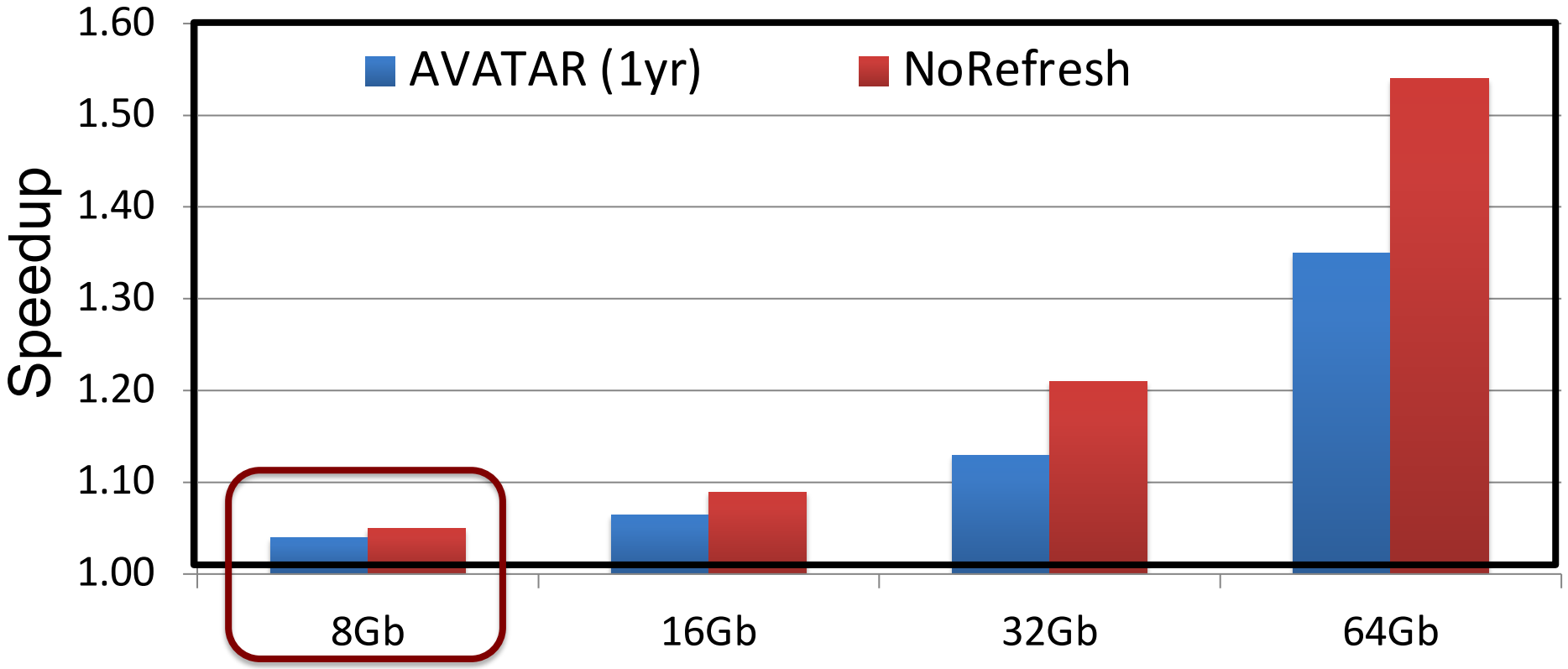
RESULTS: REFRESH REDUCTION



Retention Testing Once a Year
increase refresh reduction from 60% to 70%

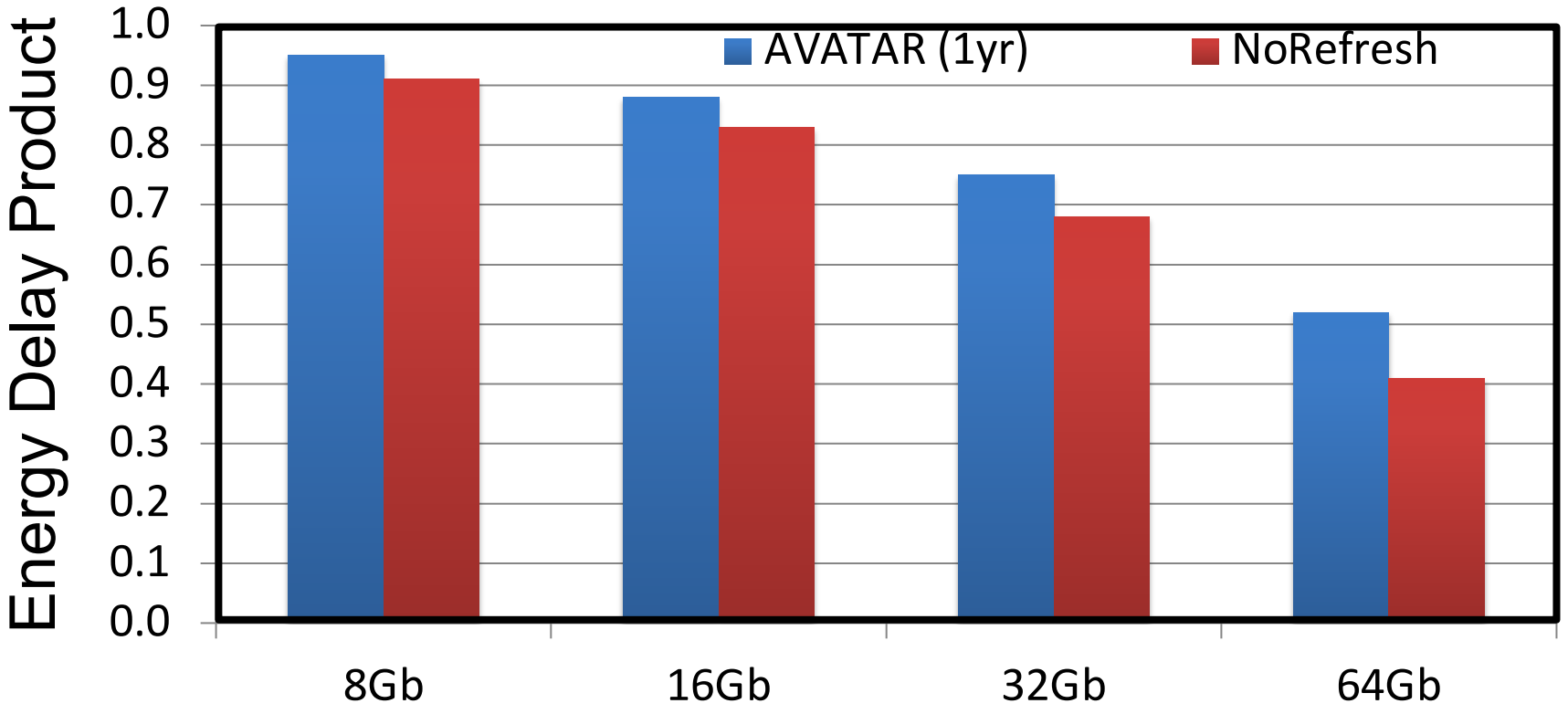
AVATAR reduces refresh by 60%-70%,
similar to multi-rate refresh but with VRT tolerance

SPEEDUP



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.

More on AVATAR [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

Handling Data-Dependent Failures [DSN'16]

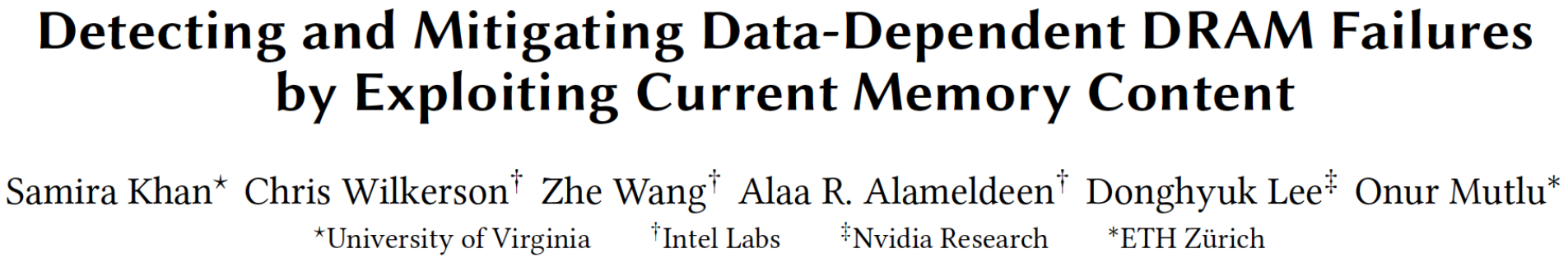
- Samira Khan, Donghyuk Lee, and Onur Mutlu,
  "PARBOR: An Efficient System-Level Technique to Detect Data-Dependent Failures in DRAM"
  *Proceedings of the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (**DSN**)*, Toulouse, France, June 2016.
  [Slides (pptx) (pdf)]

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]

- Minesh Patel, Jeremie S. Kim, and Onur Mutlu,
  "The Reach Profiler (REAPER): Enabling the Mitigation of DRAM Retention Failures via Profiling at Aggressive Conditions"
  *Proceedings of the 44th International Symposium on Computer Architecture (**ISCA**)*, Toronto, Canada, June 2017.
  [Slides (pptx) (pdf)]
  [Lightning Session Slides (pptx) (pdf)]

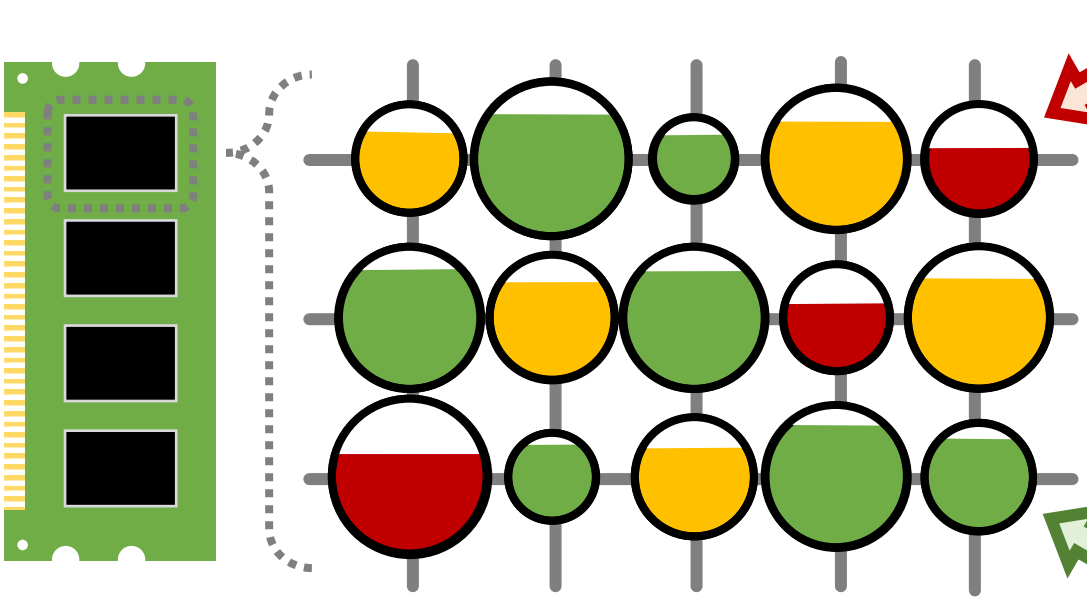
- First experimental analysis of (mobile) LPDDR4 chips
- Analyzes the complex tradeoff space of retention time profiling
- Idea: enable fast and robust profiling at higher refresh intervals & temperatures

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

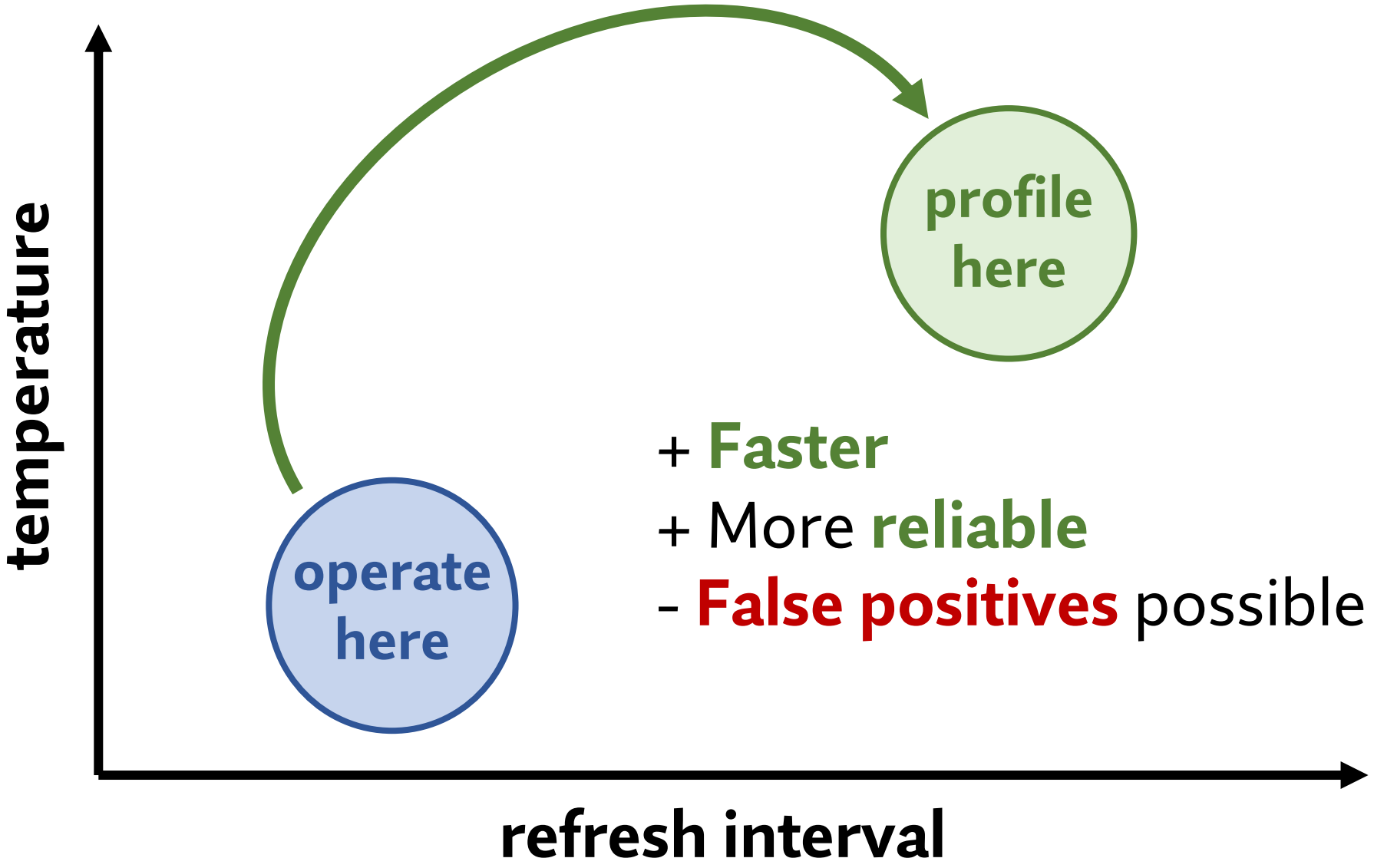
1

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



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"
  *Proceedings of the 44th International Symposium on Computer Architecture (**ISCA**)*, Toronto, Canada, June 2017.
  [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"
  *Proceedings of the 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (**DSN**)*, Portland, OR, USA, June 2019.
  [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*

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"
  *Proceedings of the 53rd International Symposium on Microarchitecture (**MICRO**)*, Virtual, October 2020.
  [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)]

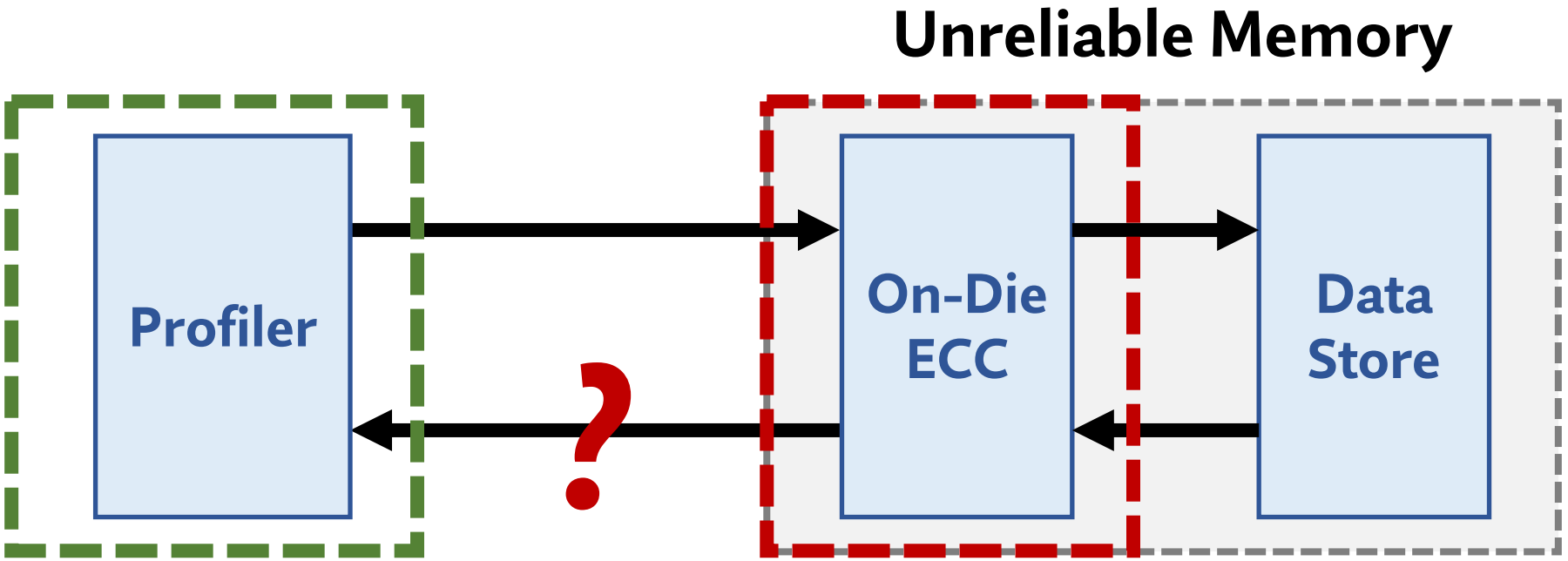
HARP: Practically and Effectively Identifying Uncorrectable Errors in Memory Chips That Use On-Die Error-Correcting Codes

Minesh Patel
ETH Zürich

Geraldo F. Oliveira
ETH Zürich

Onur Mutlu
ETH Zürich

Profiling a Memory Chip with On-Die ECC



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

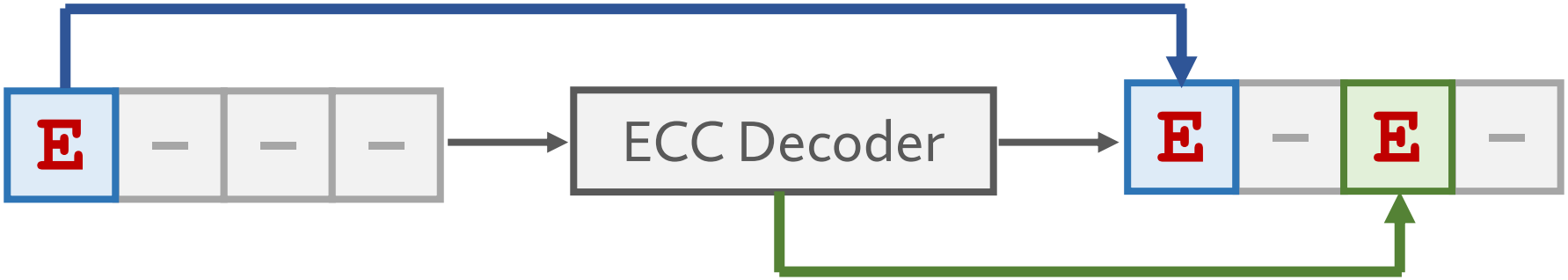
2. Makes it **harder to identify** individual at-risk bits

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**

Key Idea: **decouple** profiling for **direct** and **indirect** errors

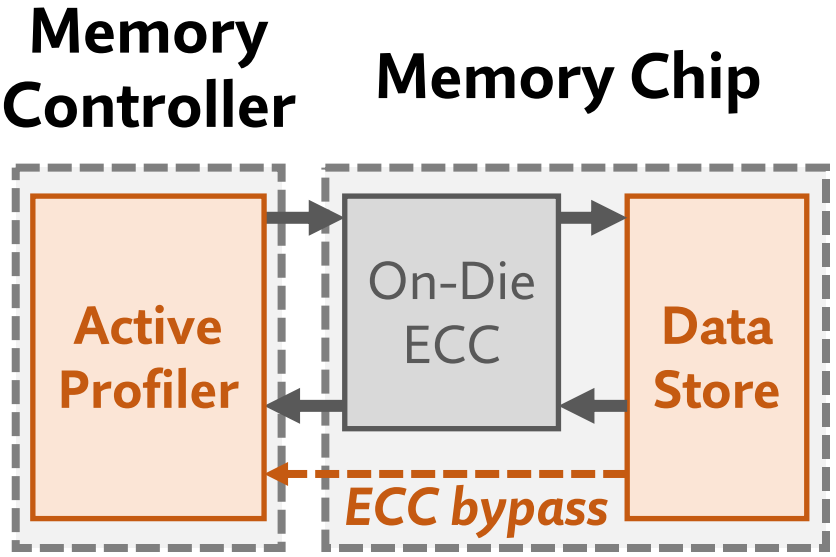
2 **Indirect error** — **Artifact** of the on-die ECC algorithm

Upper-bounded by the ECC algorithm

Hybrid Active-Reactive Profiling (HARP)

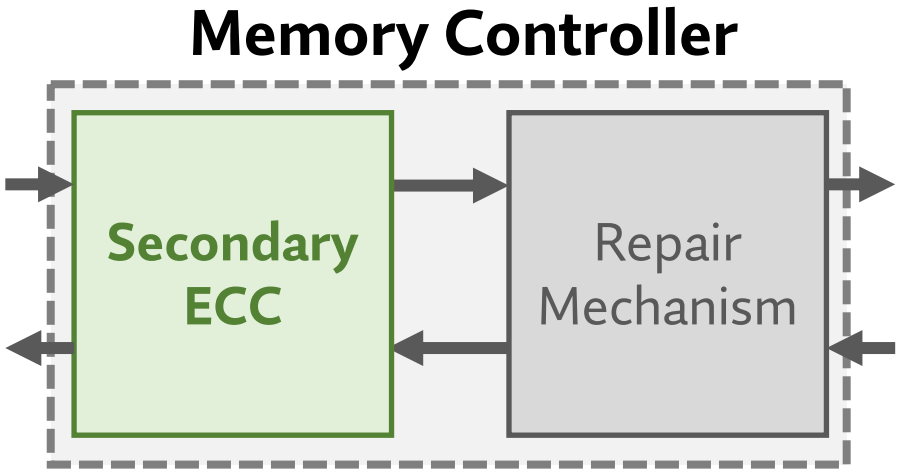
1 Active Profiling

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

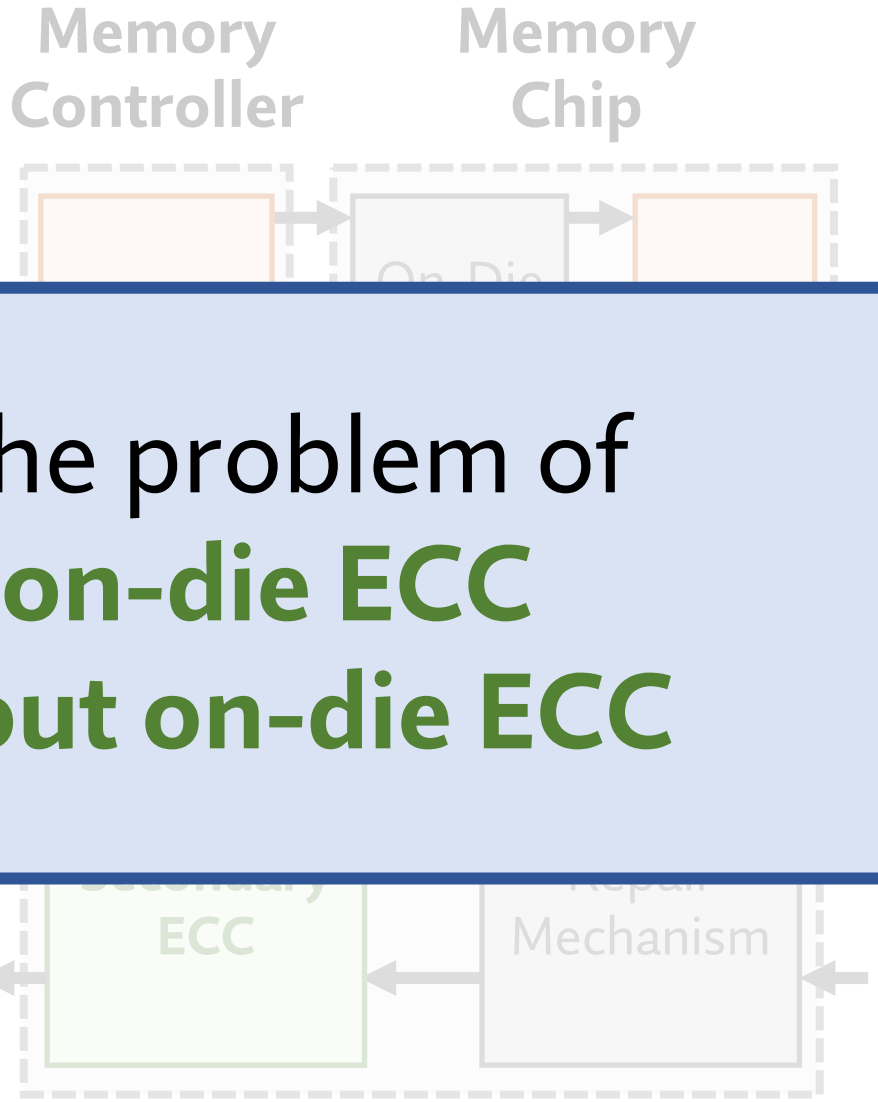


2 Reactive Profiling

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



Hybrid Active-Reactive Profiling (HARP)



1 Active Profiling

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**

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 99th-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

   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

Liu et al., "RAIDR: Retention-Aware Intelligent DRAM Refresh," ISCA 2012.

2. Binning

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

Bloom, "Space/Time Trade-offs in Hash Coding with Allowable Errors", CACM 1970.

Bloom Filter

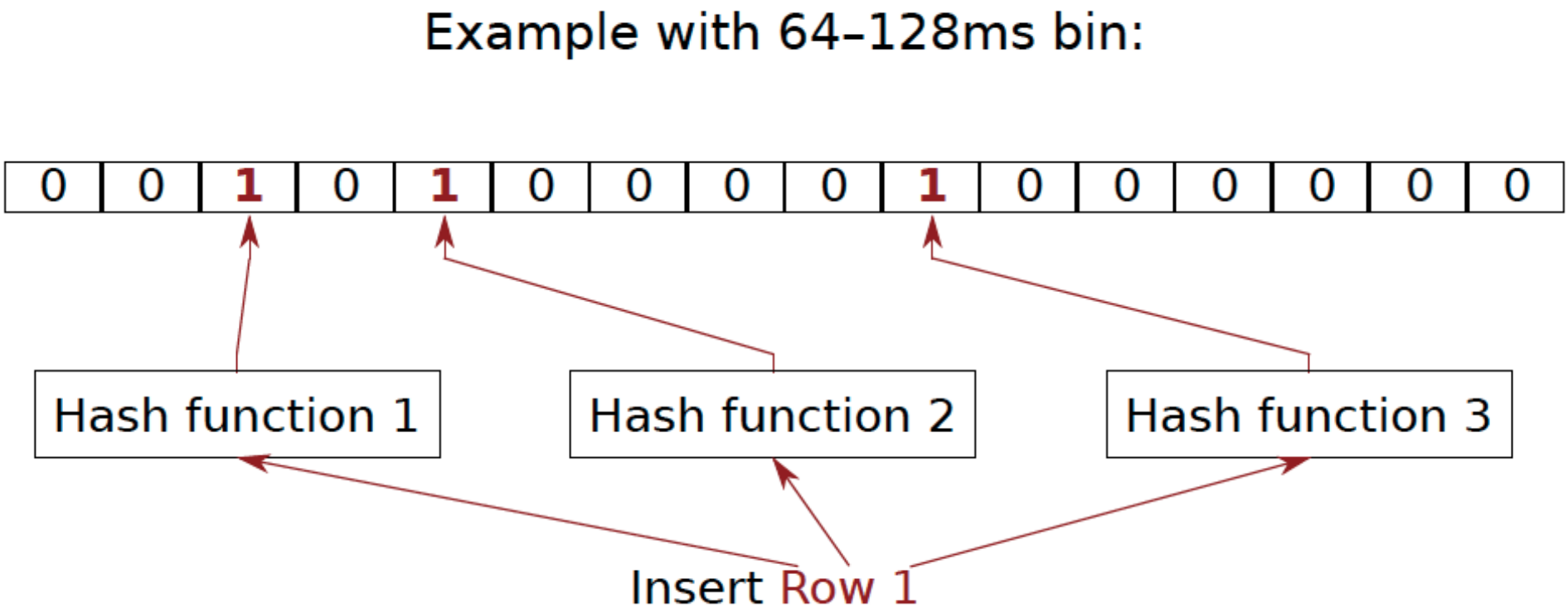
- [Bloom, CACM 1970]
- Probabilistic data structure that compactly represents set membership (presence or absence of element in a set)

- 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

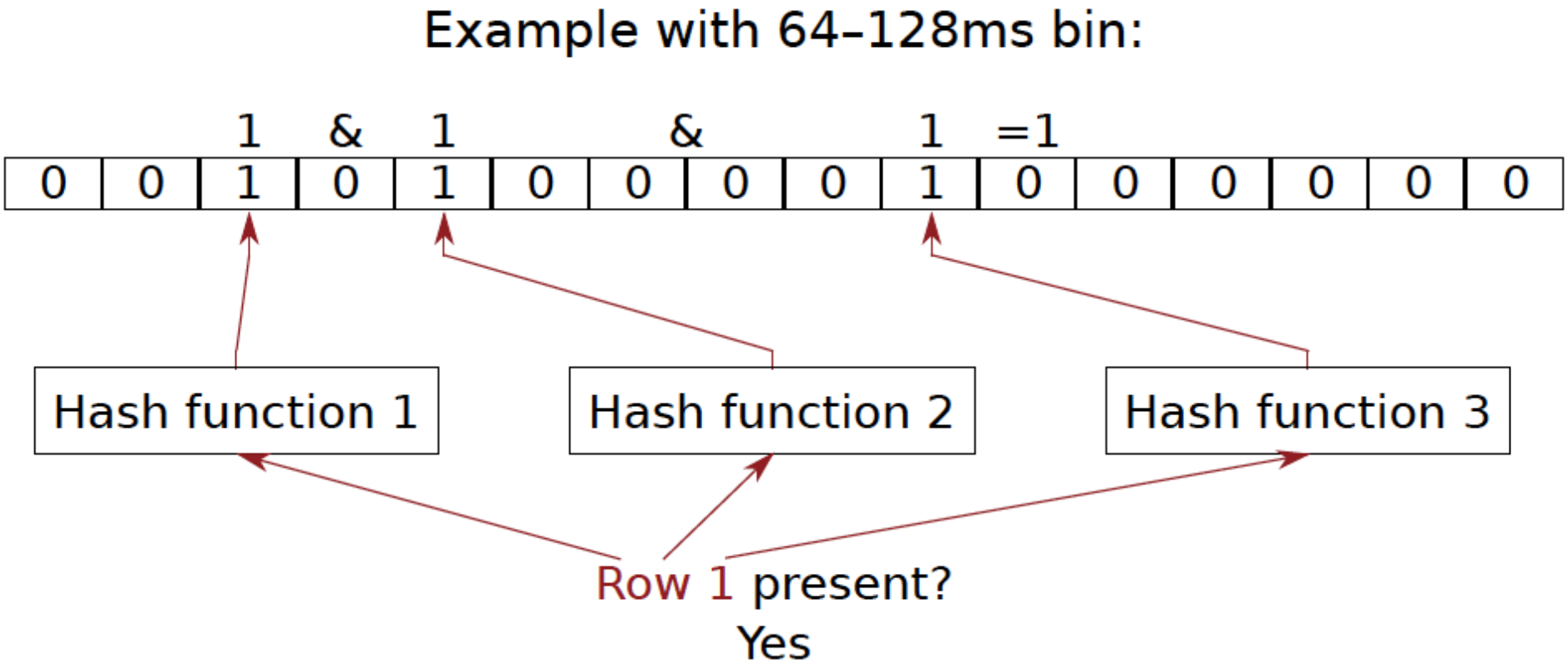
Bloom, "Space/Time Trade-offs in Hash Coding with Allowable Errors", CACM 1970.

Bloom Filter Operation Example

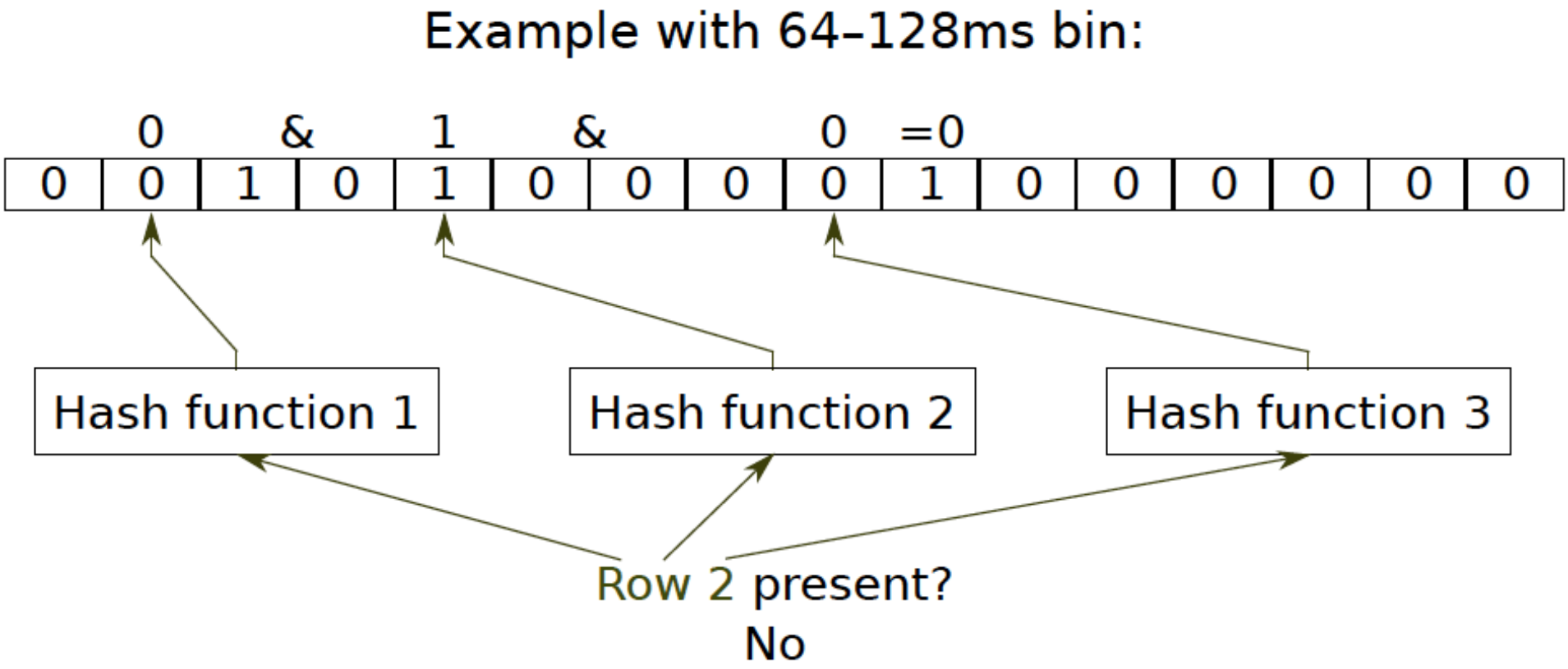


Bloom, "Space/Time Trade-offs in Hash Coding with Allowable Errors", CACM 1970.

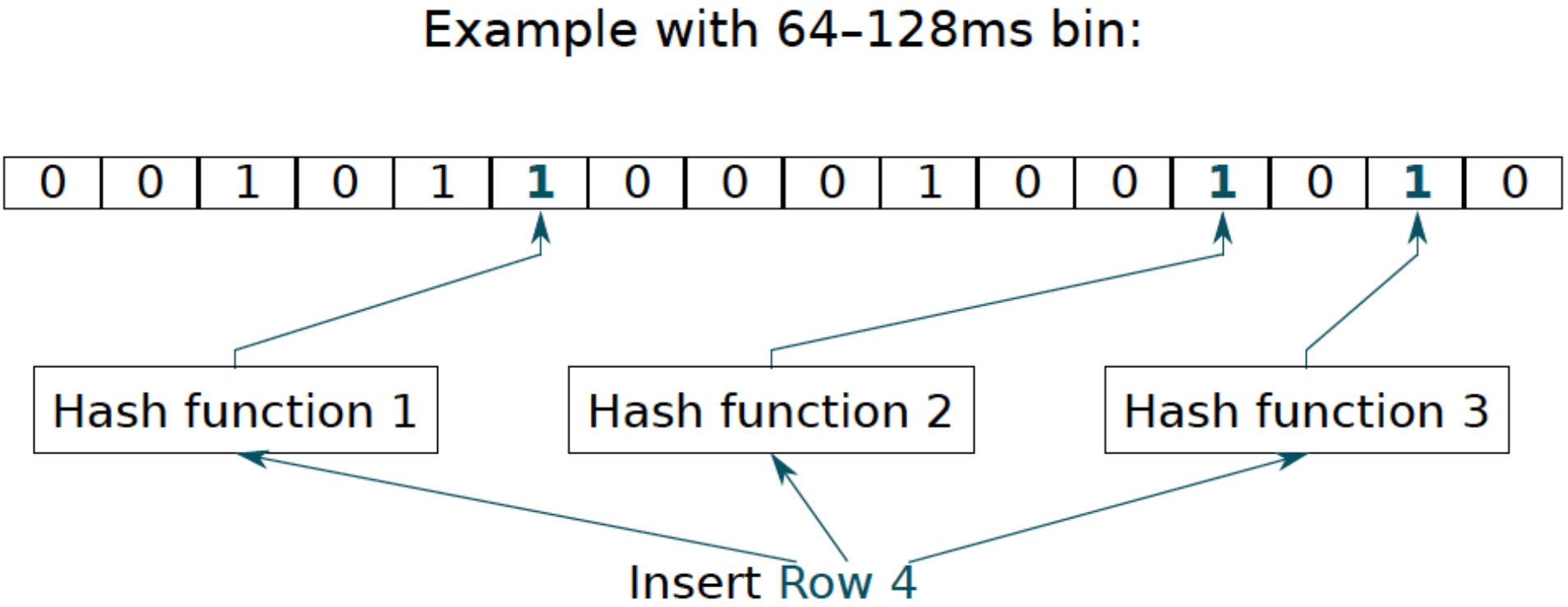
Bloom Filter Operation Example



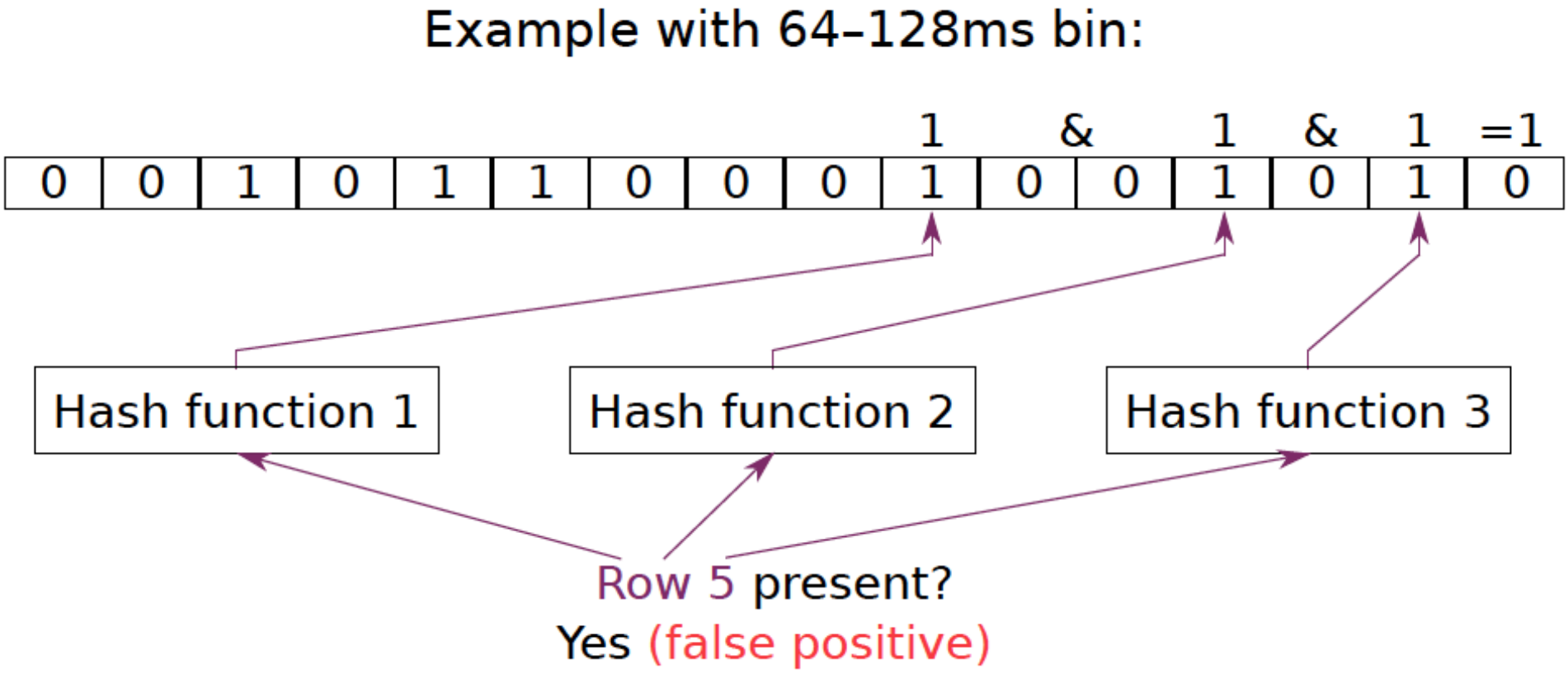
Bloom Filter Operation Example



Bloom Filter Operation Example



Bloom Filter Operation Example



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
Computer Usage Company, Newton Upper Falls, Mass.

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, "Space/Time Trade-offs in Hash Coding with Allowable Errors", CACM 1970.

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

Bloom, "Space/Time Trade-offs in Hash Coding with Allowable Errors", CACM 1970.

Benefits of Bloom Filters as Refresh Rate Bins

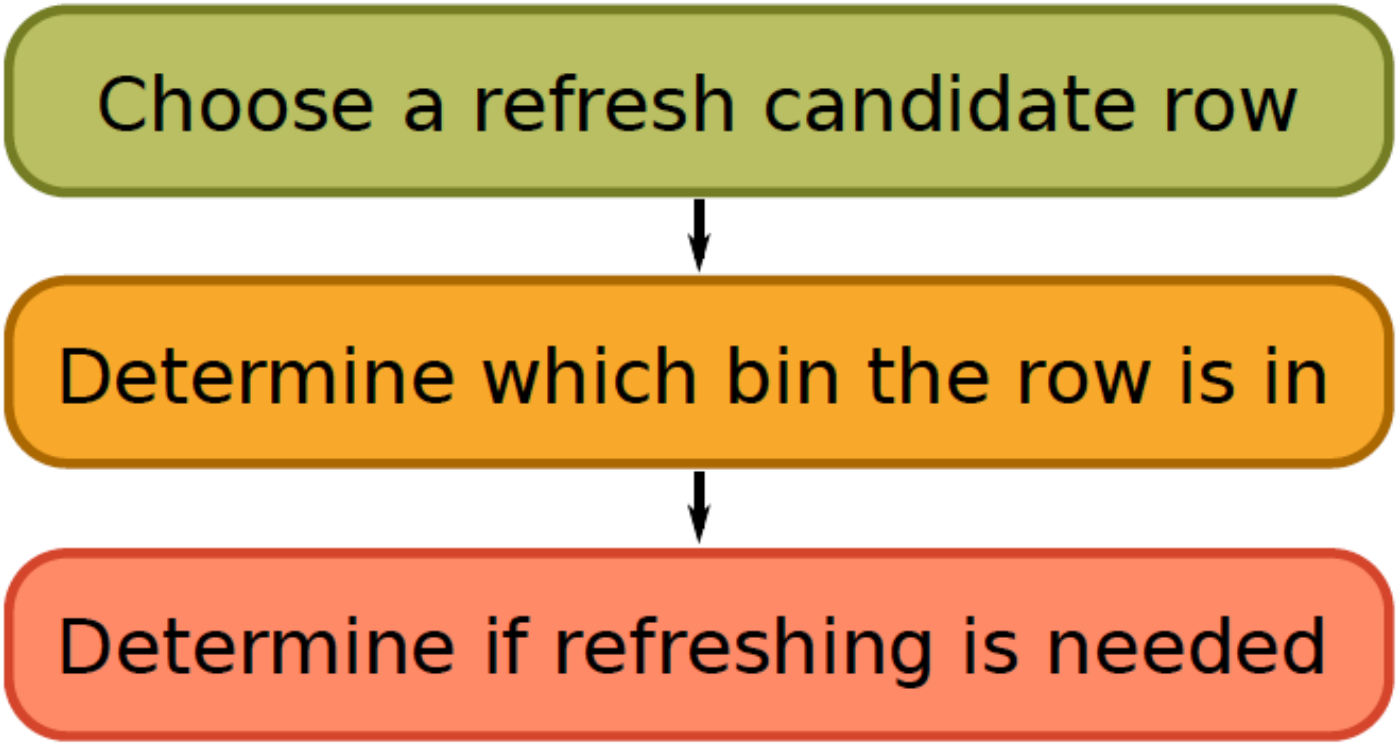
- False positives: a row may be declared present in the Bloom filter even if it was never inserted
  - Not a problem: Refresh some rows more frequently than needed
- No false negatives: rows are never refreshed less frequently than needed (no correctness problems)
- Scalable: a Bloom filter never overflows (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

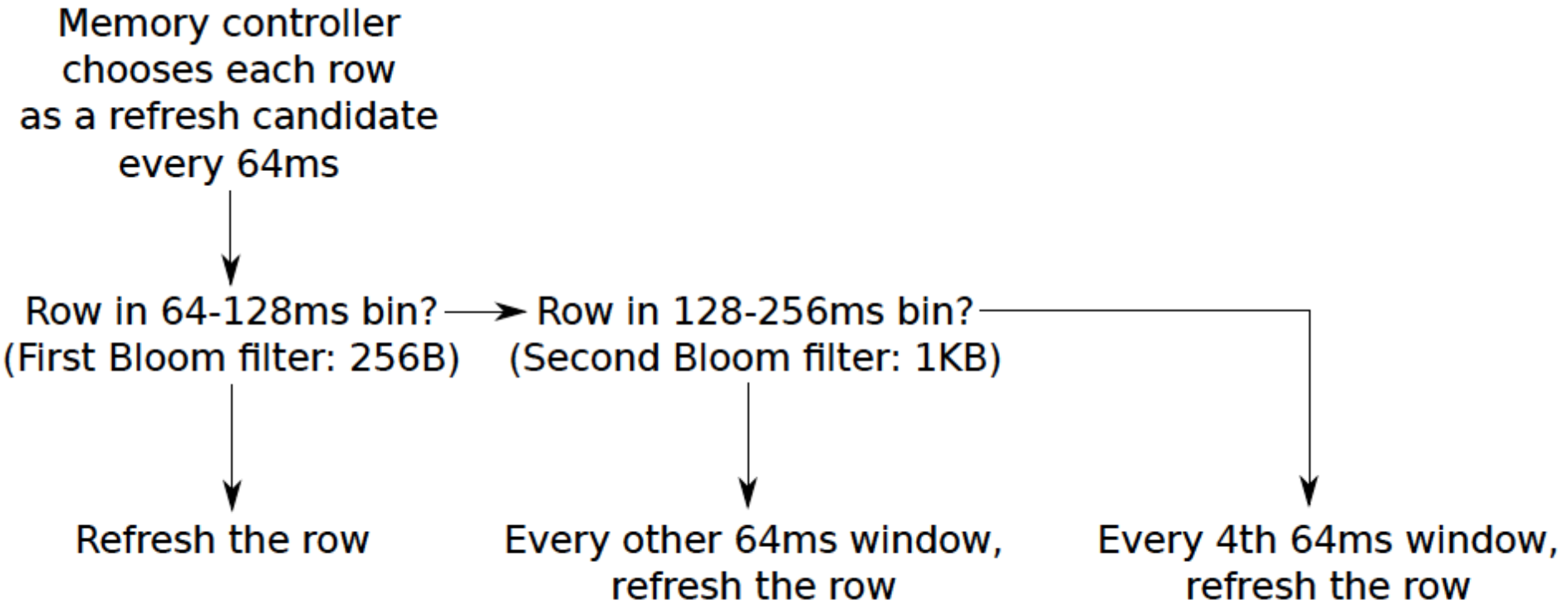
- Useful when you can tolerate false positives in set membership tests

- See the following recent examples for clear descriptions of how Bloom Filters are used
  - Liu et al., "RAIDR: Retention-Aware Intelligent DRAM Refresh," ISCA 2012.
  - Seshadri et al., "The Evicted-Address Filter: A Unified Mechanism to Address Both Cache Pollution and Thrashing," PACT 2012.
  - Yaglikci et al., "BlockHammer: Preventing RowHammer at Low Cost by Blacklisting Rapidly-Accessed DRAM Rows," HPCA 2021.

3. Refreshing (RAIDR Refresh Controller)

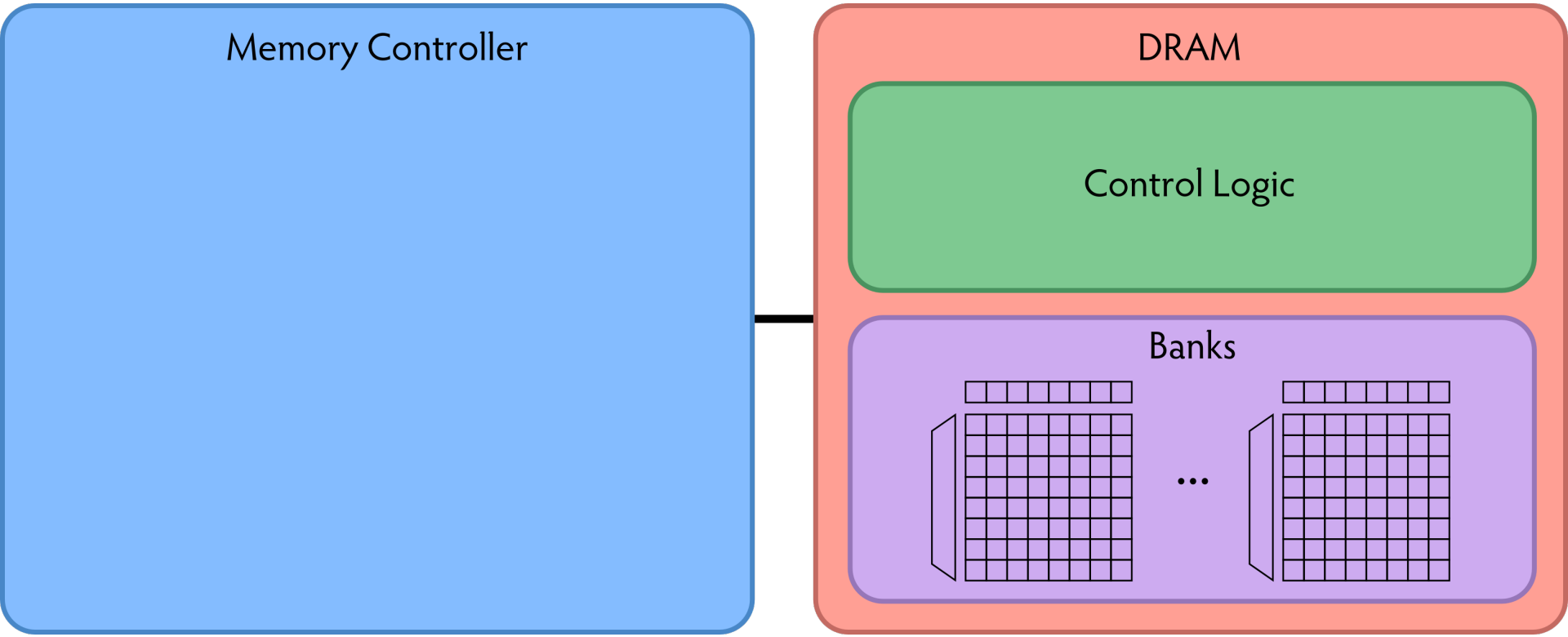


3. Refreshing (RAIDR Refresh Controller)



Liu et al., "RAIDR: Retention-Aware Intelligent DRAM Refresh," ISCA 2012.

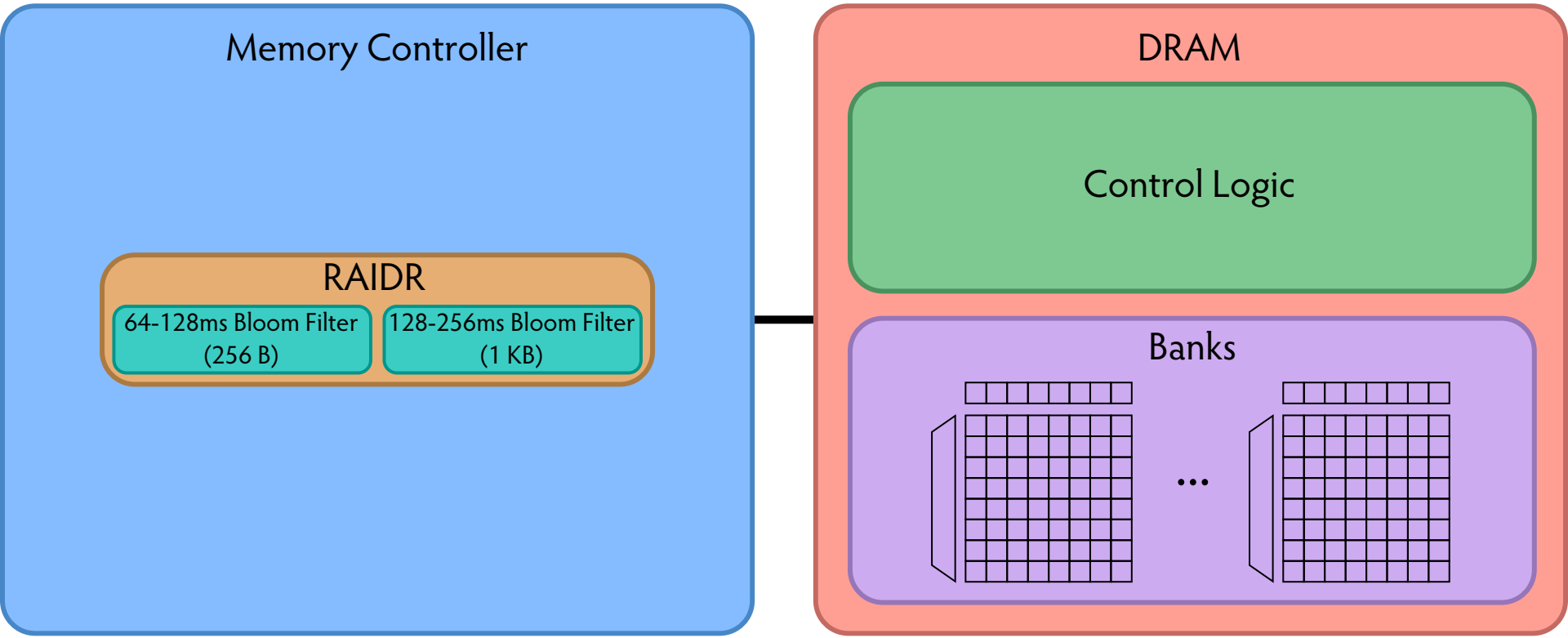
RAIDR: Baseline Design



Refresh control is in DRAM in today's auto-refresh systems

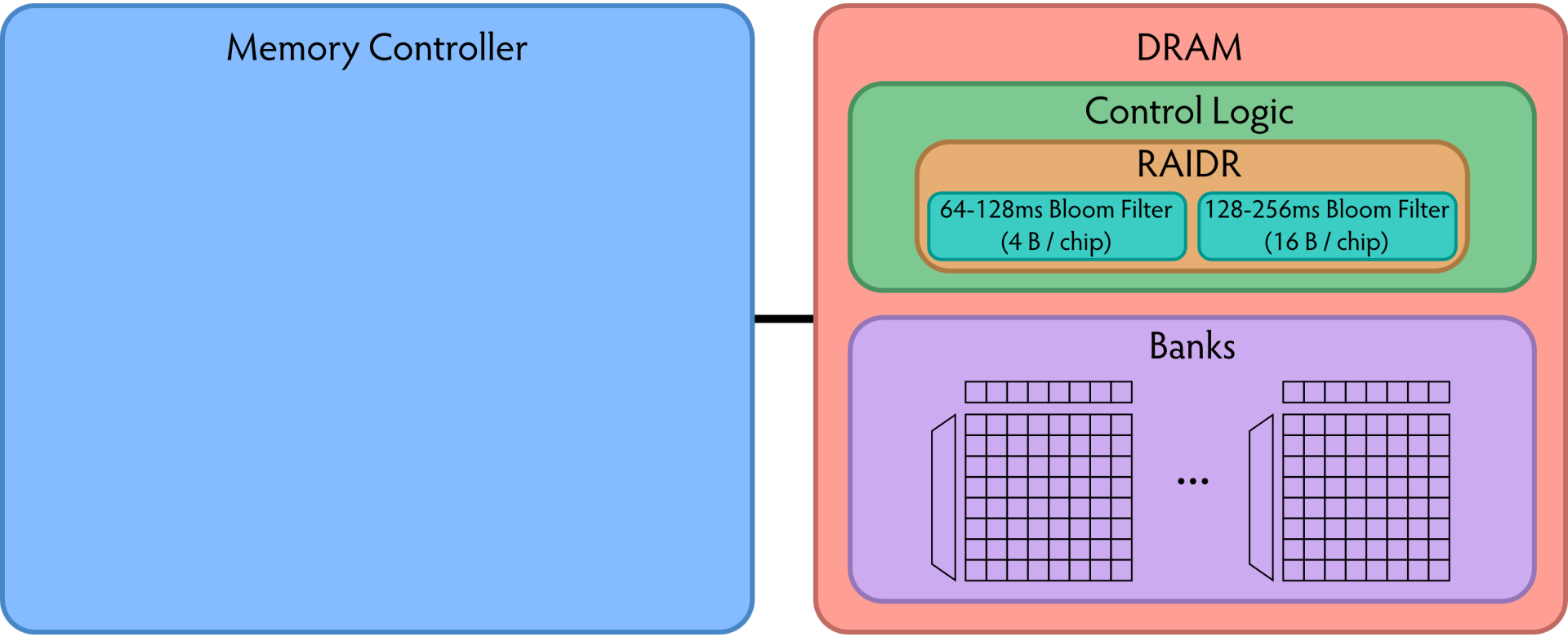
RAIDR can be implemented in either the controller or DRAM

RAIDR in Memory Controller: Option 1



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)

RAIDR in DRAM Chip: Option 2



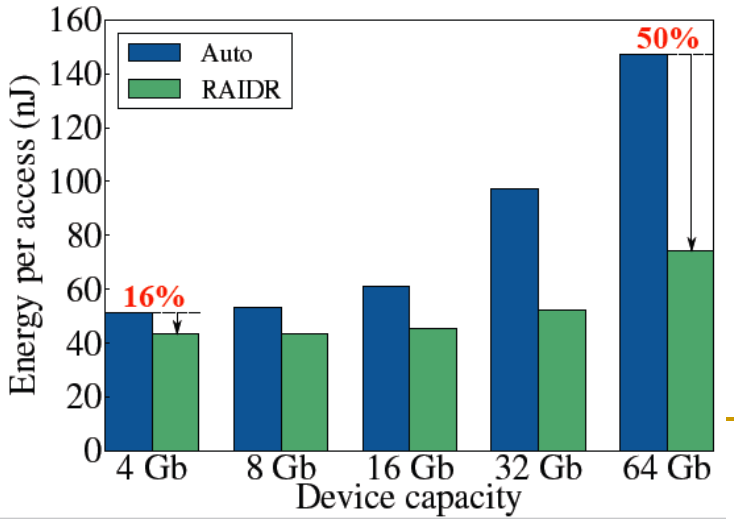
Overhead of RAIDR in DRAM chip:

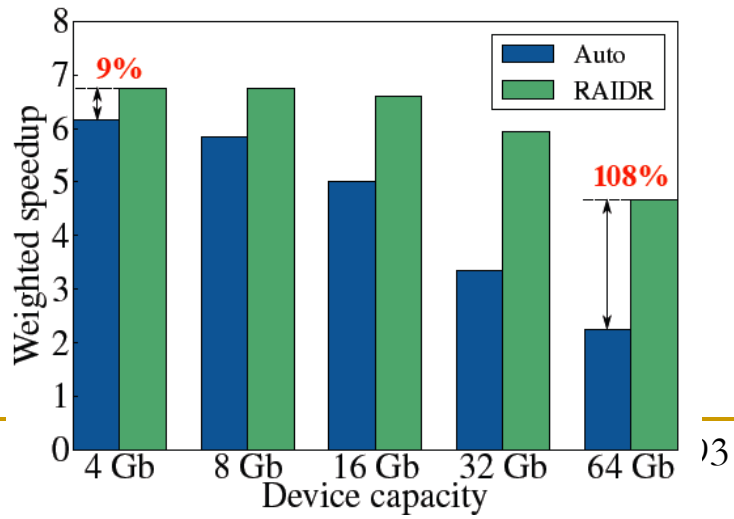
Per-chip overhead: 20B Bloom Filters, 1 counter (4 Gbit chip)

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





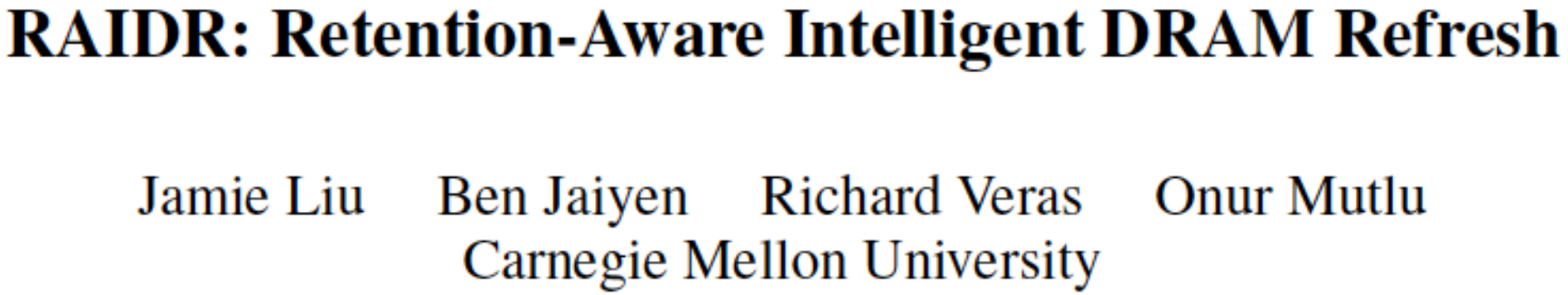
DRAM Refresh: More Questions

- 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:
  - Liu et al., "An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms," ISCA 2013.

DRAM Leakage in ISCA-50 25-Year Retrospective Issue

RAIDR: Heterogeneous 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)]
  Selected to the ISCA-50 25-Year Retrospective Issue covering 1996-2020 in 2023 (Retrospective (pdf) Full Issue).



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)]
  Selected to the ISCA-50 25-Year Retrospective Issue covering 1996-2020 in 2023 (Retrospective (pdf) Full Issue).

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"
  *Proceedings of the 41st International Symposium on Computer Architecture (**ISCA**)*, Minneapolis, MN, June 2014.
  [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).
  Selected to the ISCA-50 25-Year Retrospective Issue covering 1996-2020 in 2023 (Retrospective (pdf) Full Issue).

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

Yoongu Kim[1] Ross Daly* Jeremie Kim[1] Chris Fallin* Ji Hye Lee[1]
Donghyuk Lee[1] Chris Wilkerson[2] Konrad Lai Onur Mutlu[1]

[1]Carnegie Mellon University [2]Intel Labs

RAIDR Retrospective [ISCA 2012]

Retrospective: RAIDR: Retention-Aware Intelligent DRAM Refresh

Onur Mutlu
ETH Zürich

***Abstract*—Dynamic Random Access Memory (DRAM) is the prevalent memory technology used to build main memory systems of almost all computers. A fundamental shortcoming of DRAM is the need to refresh memory cells to keep stored data intact. DRAM refresh consumes energy and degrades performance. It is also a technology scaling challenge as its negative effects become worse as DRAM cell size reduces and DRAM chip capacity increases.**

Our ISCA 2012 paper, RAIDR [1], examines the DRAM refresh problem from a modern computing systems perspective, demonstrating its projected impact on systems with higher-capacity DRAM chips expected to be manufactured in the future. It proposes and evaluates a simple and low-cost solution that greatly reduces the performance & energy overheads of refresh by exploiting variation in data retention times across DRAM rows. The key idea is to group the DRAM rows into bins in terms of their minimum data retention times, store the bins in low-cost Bloom filters, and refresh rows in different bins at different rates. Evaluations in our paper (and later works) show that the idea greatly improves performance & energy efficiency and its benefits increase with DRAM chip capacity. The paper embodies an approach we have termed *system-DRAM co-design*.

This short retrospective provides a brief analysis of our RAIDR paper and its impact. We briefly describe the mindset and circumstances that led to our focus on the DRAM refresh problem and RAIDR's development, discuss later works that provided improved analyses and solutions, and make some educated guesses on what the future may bring on the DRAM refresh problem (and more generally in DRAM technology scaling).

I. Background, Approach & Mindset

At the time we began our focus on solving the DRAM refresh (i.e., data retention) challenge in late 2010, my research group, SAFARI, had already been working on memory controllers and memory technology scaling issues, motivated by many challenges memory systems, in particular the DRAM technology [2], have been facing (as described in, e.g., [3-5]). Our intense work on memory systems started during my tenure at Microsoft Research from 2006 and continued at CMU from 2009. For example, we had developed better memory schedulers for multi-core processors (e.g., [6-10]), developed platforms to perform voltage and frequency scaling to save DRAM energy (e.g., [11]) and architected emerging memory technologies to replace or augment DRAM (e.g., [12-14]). We were quite excited about the prospect of much more capable memory controllers in enabling better memory systems. As such, we were pursuing new memory-controller and system-level techniques to 1) overcome the challenging device- and circuit-level scaling issues of memory technologies and 2) better exploit underlying characteristics of memory technology; an approach we termed *system-DRAM co-design* [4, 5].

RAIDR is a product of this approach. Our focus on data retention issues and other low-level issues in DRAM especially increased via discussions with the Samsung DRAM Design Team, who visited us in April 2011 and encouraged the development of our system-level solutions to DRAM issues, enabling strong support both technically and funding-wise. In fact, much of our ensuing research in DRAM was supported by generous gift funding by and technical discussions with Samsung based on a proposal entitled *"New ideas to enhance DRAM scaling: Scaling-aware controller design and co-design of DRAM and controllers"* (Intel provided similar gift funding and technical discussions).

II. Contributions and Impact of RAIDR

RAIDR is the first work to propose a low-cost memory controller technique that reduces refresh operations by exploiting variation in data retention times across DRAM rows. Its appeal comes from its simplicity and low cost, enabled by the careful use of Bloom filters [15]. Exploiting the DRAM data retention time distribution [16], RAIDR can eliminate a very large fraction (e.g., ~75% or more) of refresh operations with very small hardware cost at the memory controller.

Apart from the new technique it introduced, we believe the RAIDR paper made two other major contributions that have enabled a large number of future works and new ideas. First, it provided an empirical scaling analysis that clearly demonstrated the importance of the DRAM refresh problem in modern systems: if nothing is done about it, DRAM refresh would waste almost half of the throughput and half of the energy of a high-capacity 64-Gb DRAM chip! This analytical prediction encouraged more works in the topic area. Second, it demonstrated a methodical way of exploiting cell-level heterogeneous data retention times at the system (e.g., memory controller) level: if data retention times of DRAM rows are accurately known, the system can use them to optimize DRAM refresh and get rid of most refresh operations. This demonstration enabled other works to develop 1) methods for accurately determining DRAM data retention times and 2) other system-level approaches to optimize DRAM behavior using data retention time information.

III. Building on RAIDR and Making It Work

We believe RAIDR enabled a refreshing approach to DRAM refresh. Its largest contribution could be the works it has inspired that rigorously examined the questions of 1) how to perform accurate DRAM data retention time profiling, 2) how to overcome potential hurdles that stand in the way of obtaining accurate minimum data retention times, 3) how to reliably get rid of unnecessary refresh operations.

We wanted to make RAIDR work in a real system setting. To this end, collaboratively with Intel, we developed an FPGA-based flexible DRAM testing infrastructure [17] that enabled us to rigorously test data retention times of cells in real DDR3 DRAM chips. Using this infrastructure, later open sourced as SoftMC [18, 19] and DRAM Bender [20, 21], we experimentally examined practical issues that affect the accuracy (and performance) of DRAM data retention time profiling. We analyzed two major issues that make such profiling very challenging: 1) data pattern dependence (DPD) of retention times [17, 22], and 2) the variable retention time (VRT) phenomenon [17, 23, 24]. Our follow-up work, which appeared at ISCA 2013 [17], provides a detailed experimental analysis of these challenges in cutting-edge DRAM chips, demonstrating that ideas like RAIDR that depend on accurate identification of retention times are not easy to exploit in practice. Later works (e.g., [25-32]) developed new methods for making RAIDR-like techniques more practical by tackling especially the DPD and VRT problems and enhancing retention time profiling methods to work in the presence of DPD and VRT, usually by exploiting ECC techniques that have since become mainstream in DRAM chips (see [31, 33]) to tolerate VRT [34].

The development of our flexible FPGA-based DRAM testing infrastructure also enabled experimental DRAM research in directions that are completely different from retention time profiling and refresh. These include studies that provided valuable experimental data on various DRAM characteristics, including RowHammer [20, 35, 44], latency [45-48], voltage-latency-reliability relationship [49], power consumption and modeling [50]. Using this infrastructure, later research also demonstrated the ability of real off-the-shelf DRAM chips to perform data copy/initialization and bulk bitwise operations [51-55], implement physical unclonable functions [56], and generate true random numbers [57, 58]. We believe the investment we made to try to make RAIDR work using a real FPGA-based infrastructure helped us and the broader research community uncover many interesting characteristics of DRAM chips and propose new ideas to make DRAM-based systems more secure, reliable, efficient, and high performance.

Other later works provided refined models of DRAM refresh's impact on system performance (e.g., [59, 60]) and developed new methods to reduce DRAM refresh's negative impact on performance & energy (e.g., [41, 59-67]). Our HPCA 2014 paper [25] developed a more refined projection of the effect of DRAM refresh as technology scales. AVATAR in DSN 2015 [26] and REAPER in ISCA 2017 [30] enabled more practical ways of exploiting heterogeneous retention times in the presence of VRT. Our recent work [66] shows that with a more flexible DRAM interface that gives some autonomy to DRAM chips, RAIDR can be more efficiently implemented inside the DRAM chip.

IV. Summary and Future Outlook

RAIDR is a nice example of how enthusiastic support from industry can foster new ideas that can open up many new analyses and other ideas. We were inspired by our deep technical discussions with especially Samsung and Intel, along with prior works that described DRAM technology scaling challenges (e.g., [3]) and that developed promising solutions (e.g., [68, 69]). Engineers from Samsung and Intel later wrote an insightful paper [34] on DRAM scaling challenges, which described refresh as a key problem and advocated a controller-DRAM co-design approach as we had been advocating [1, 4]. RAIDR was also a nice example of how teaching & research smoothly feed each other: much of the research was done as part of a group project in the Parallel Computer Architecture class I taught at CMU in Fall 2011.

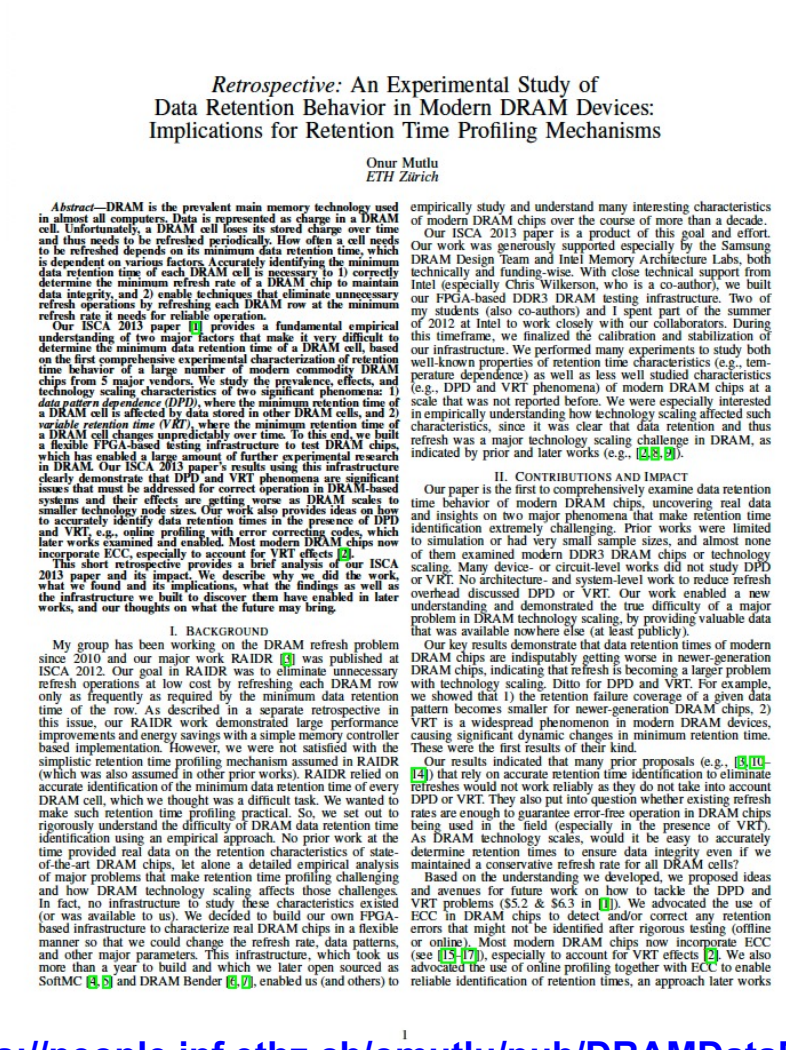
Looking forward, DRAM technology scaling is getting worse and data retention will continue to be an important issue [34, 70]. The negative effects of DRAM refresh will be (and are being) exacerbated by other technology scaling issues like RowHammer [35] that require even more refreshes as a solution [41, 44, 71]. We believe there are a lot more new ideas and techniques to develop to minimize the impact of refresh on computing systems.

References

[1] J. Liu *et al.*, "RAIDR: Retention-Aware Intelligent DRAM Refresh," in *ISCA*, 2012.
[2] R. H. Dennard, "Field-effect Transistor Memory," 1968, US Patent 3,387,286.
[3] J. A. Mandelman *et al.*, "Challenges and Future Directions for the Scaling of Dynamic Random-Access Memory (DRAM)," *IBM JRD*, 2002.
[4] O. Mutlu, "Memory Scaling: A Systems Architecture Perspective," *IMW*, 2013.
[5] O. Mutlu and L. Subramanian, "Research Problems and Opportunities in Memory Systems," *SUPERFRI*, 2014.
[6] O. Mutlu and T. Moscibroda, "Stall-Time Fair Memory Access Scheduling for Chip Multiprocessors," in *MICRO*, 2007.
[7] O. Mutlu *et al.*, "Parallelism-Aware Batch Scheduling: Enhancing Both Performance and Fairness of Shared DRAM Systems," in *ISCA*, 2008.
[8] Y. Kim *et al.*, "Thread Cluster Memory Scheduling: Exploiting Differences in Memory Access Behavior," in *MICRO*, 2010.
[9] Y. Kim *et al.*, "ATLAS: A Scalable and High-performance Scheduling Algorithm for Multiple Memory Controllers," in *HPCA*, 2010.
[10] S. Muralidhara, "Reducing Memory Interference in Multicore Systems via Application-aware Memory Channel Partitioning," in *MICRO*, 2011.
[11] H. David *et al.*, "Memory power management via dynamic voltage/frequency scaling," in *ICAC*, 2011.
[12] B. C. Lee *et al.*, "Architecting Phase Change Memory as a Scalable DRAM Alternative," in *ISCA*, 2009.
[13] H. Yoon *et al.*, "Row Buffer Locality Aware Caching Policies for Hybrid Memories," in *ICCD*, 2012.
[14] J. Meza *et al.*, "Enabling Efficient and Scalable Hybrid Memories using Fine-granularity DRAM Cache Management," *CAL*, 2012.
[15] B. Bloom, "Space/Time Trade-Offs in Hash Coding with Allowable Errors," *CACM*, 1970.
[16] K. Kim and J. Lee, "A new investigation of data retention time in truly nanoscaled DRAMs," *IEEE EDL*, vol. 30, no. 8, 2009.
[17] J. Liu *et al.*, "An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms," in *ISCA*, 2013.
[18] H. Hassan *et al.*, "SoftMC: A Flexible and Practical Open-Source Infrastructure for Enabling Experimental DRAM Studies," in *HPCA*, 2017.
[19] "SoftMC Repository," https://github.com/CMU-SAFARI/SoftMC/.
[20] A. Olgun *et al.*, "DRAM Bender: An Extensible and Versatile FPGA-based Infrastructure to Easily Test State-of-the-art DRAM Chips," *TCAD*, 2023.
[21] "DRAM Bender," https://github.com/CMU-SAFARI/DRAM-Bender/.
[22] Y. Nakagome *et al.*, "The impact of data-line interference noise on DRAM scaling," *JSSC*, 1988.
[23] D. Yaney *et al.*, "A Meta-stable Leakage Phenomenon in DRAM Charge Storage - Variable Hold Time," in *IEDM*, 1987.
[24] P. J. Restle *et al.*, "DRAM Variable Retention Time," in *IEDM*, 1992.
[25] S. Khan, "The Efficacy of Error Mitigation Techniques for DRAM Retention Failures: A Comparative Experimental Study," in *SIGMETRICS*, 2014.
[26] M. K. Qureshi *et al.*, "AVATAR: A Variable-Retention-Time (VRT) Aware Refresh for DRAM Systems," in *DSN*, 2015.
[27] S. Khan *et al.*, "PARBOR: An Efficient System-Level Technique to Detect Data Dependent Failures in DRAM," in *DSN*, 2016.
[28] S. Khan *et al.*, "A Case for Memory Content-Based Detection and Mitigation of Data-Dependent Failures in DRAM," *CAL*, 2016.
[29] S. Khan *et al.*, "Detecting and Mitigating Data-Dependent DRAM Failures by Exploiting Current Memory Content," in *MICRO*, 2017.
[30] M. Patel, "The Reach Profiler (REAPER): Enabling the Mitigation of DRAM Retention Failures via Profiling at Aggressive Conditions," in *ISCA*, 2017.
[31] M. Patel *et al.*, "Bit-Exact ECC Recovery (BEER): Determining DRAM On-Die ECC Functions by Exploiting DRAM Data Retention Characteristics," in *MICRO*, 2020.
[32] M. Patel *et al.*, "HARP: Practically and Effectively Identifying Uncorrectable Errors in Main Memory Chips That Use On-Die ECC," in *MICRO*, 2021.
[33] M. Patel *et al.*, "Understanding and Modeling On-die Error Correction in Modern DRAM: An Experimental Study using Real Devices," in *DSN*, 2019.
[34] U. Kang *et al.*, "Co-Architecting Controllers and DRAM to Enhance DRAM Process Scaling," in *The Memory Forum*, 2014.
[35] Y. Kim *et al.*, "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors," in *ISCA*, 2014.
[36] J. S. Kim *et al.*, "Revisiting RowHammer: An Experimental Analysis of Modern DRAM Devices and Mitigation Techniques," in *ISCA*, 2020.
[37] L. Orosa, "A Deeper Look into RowHammer's Sensitivities: Experimental Analysis of Real DRAM Chips and Implications on Future Attacks and Defenses," in *MICRO*, 2021.
[38] M. Farmani *et al.*, "RHAT: Efficient RowHammer-Aware Test for Modern DRAM Modules," in *ETS*, 2021.
[39] P. Frigo *et al.*, "TRRespass: Exploiting the Many Sides of Target Row Refresh," in *S&P*, 2020.
[40] H. Hassan *et al.*, "Uncovering In-DRAM RowHammer Protection Mechanisms: A New Methodology, Custom RowHammer Patterns, and Implications," in *MICRO*, 2021.
[41] A. G. Yağlıkçı *et al.*, "BlockHammer: Preventing RowHammer at Low Cost by Blacklisting Rapidly-Accessed DRAM Rows," in *HPCA*, 2021.
[42] A. G. Yağlıkçı *et al.*, "HiRA: Hidden Row Activation for Reducing Refresh Latency of Off-the-Shelf DRAM Chips," in *MICRO*, 2022.
[43] A. G. Yağlıkçı *et al.*, "Understanding RowHammer Under Reduced Wordline Voltage: An Experimental Study Using Real DRAM Devices," in *DSN*, 2022.
[44] A. Olgun *et al.*, "An Experimental Analysis of RowHammer in HBM2 DRAM Chips," in *DSN Disrupt*, 2023.
[45] H. Luo *et al.*, "RowPress: Amplifying Read Disturbance in Modern DRAM Chips," in *ISCA*, 2023.
[46] D. Lee *et al.*, "Adaptive-Latency DRAM: Optimizing DRAM Timing for the Common-Case," in *HPCA*, 2015.
[47] K. K. Chang *et al.*, "Understanding Latency Variation in Modern DRAM Chips: Experimental Characterization, Analysis, and Optimization," in *SIGMETRICS*, 2016.
[48] D. Lee *et al.*, "Design-induced Latency Variation in Modern DRAM Chips: Characterization, Analysis, and Latency Reduction Mechanisms," *POMACS*, 2017.
[49] J. Kim *et al.*, "Solar-DRAM: Reducing DRAM Access Latency by Exploiting the Variation in Local Bitlines," in *ICCD*, 2018.
[50] K. Chang *et al.*, "Understanding Reduced-Voltage Operation in Modern DRAM Devices: Experimental Characterization, Analysis, and Mechanisms," in *SIGMETRICS*, 2017.
[51] S. Ghose *et al.*, "What Your DRAM Power Models Are Not Telling You: Lessons from a Detailed Experimental Study," in *SIGMETRICS*, 2018.
[52] V. Seshadri *et al.*, "Fast Bulk Bitwise AND and OR in DRAM," *CAL*, 2015.
[53] V. Seshadri *et al.*, "Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology," in *MICRO*, 2017.
[54] A. Olgun *et al.*, "PiDRAM: A Holistic End-to-end FPGA-based Framework for Processing-in-DRAM," *TACO*, 2023.
[55] F. Gao *et al.*, "ComputeDRAM: In-Memory Compute Using Off-the-Shelf DRAMs," in *MICRO*, 2019.
[56] F. Gao *et al.*, "FracDRAM: Fractional Values in Off-the-Shelf DRAM," in *MICRO*, 2022.
[57] J. S. Kim *et al.*, "The DRAM Latency PUF: Quickly Evaluating Physical Unclonable Functions by Exploiting the Latency-Reliability Tradeoff in Modern Commodity DRAM Devices," in *HPCA*, 2018.
[58] J. Kim, "D-RaNGe: Using Commodity DRAM Devices to Generate True Random Numbers with Low Latency and High Throughput," in *HPCA*, 2019.
[59] A. Olgun *et al.*, "QUAC-TRNG: High-Throughput True Random Number Generation Using Quadruple Row Activation in Commodity DRAMs," in *ISCA*, 2021.
[60] K. K. Chang *et al.*, "Improving DRAM Performance by Parallelizing Refreshes with Accesses," in *HPCA*, 2014.
[61] J. Mukundan *et al.*, "Understanding and Mitigating Refresh Overheads in High-Density DDR4 DRAM Systems," in *ISCA*, 2013.
[62] C.-H. Lin *et al.*, "SECRET: Selective Error Correction for Refresh Energy Reduction in DRAMs," in *ICCD*, 2012.
[63] P. J. Nair *et al.*, "ArchShield: Architectural Framework for Assisting DRAM Scaling by Tolerating High Error Rates," in *ISCA*, 2013.
[64] P. Nair *et al.*, "A Case for Refresh Pausing in DRAM Memory Systems," in *HPCA*, 2013.
[65] T. Zhang *et al.*, "CREAM: A Concurrent-Refresh-Aware DRAM Memory Architecture," in *HPCA*, 2014.
[66] H. Hassan *et al.*, "CROW: A Low-Cost Substrate for Improving DRAM Performance, Energy Efficiency, and Reliability," in *ISCA*, 2019.
[67] H. Hassan *et al.*, "A Case for Self-Managing DRAM Chips: Improving Performance, Efficiency, Reliability, and Security via Autonomous in-DRAM Maintenance Operations," arXiv:2207.13358, 2022.
[68] A. Das *et al.*, "VRL-DRAM: Improving DRAM Performance via Variable Refresh Latency," in *DAC*, 2018.
[69] R. Venkatesan *et al.*, "Retention-Aware Placement in DRAM (RAPID): Software Methods for Quasi-Non-Volatile DRAM," in *HPCA*, 2006.
[70] S. Liu *et al.*, "Flikker: Saving DRAM Refresh-Power through Critical Data Partitioning," in *ASPLOS*, 2011.
[71] W. Kim, "A 1.1V 16Gb DDR5 DRAM with Probabilistic-Aggressor Tracking, Refresh-Management Functionality, Per-Row Hammer Tracking, a Multi-Step Precharge, and Core-Bias Modulation for Security and Reliability Enhancement," in *ISSCC*, 2023.
[72] O. Mutlu *et al.*, "Fundamentally Understanding and Solving RowHammer," in *ASP-DAC*, 2023.

https://people.inf.ethz.ch/omutlu/pub/RAIDR_50YearsOfISCA-Retrospective_isca23.pdf

Retention Analysis Retrospective [ISCA 2013]



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

Onur Mutlu
ETH Zürich

***Abstract*—DRAM is the prevalent main memory technology used in almost all computers. Data is represented as charge in a DRAM cell. Unfortunately, a DRAM cell loses its stored charge over time and thus needs to be refreshed periodically. How often a cell needs to be refreshed depends on its minimum data retention time, which is dependent on various factors. Accurately identifying the minimum data retention time of each DRAM cell is necessary to 1) correctly determine the minimum refresh rate of a DRAM chip to maintain data integrity, and 2) enable techniques that eliminate unnecessary refresh operations by refreshing each DRAM row at the minimum refresh rate it needs for reliable operation.**

Our ISCA 2013 paper [1] provides a fundamental empirical understanding of two major factors that make it very difficult to determine the minimum data retention time of a DRAM cell, based on the first comprehensive experimental characterization of retention time behavior of a large number of modern commodity DRAM chips from 5 major vendors. We study the prevalence, effects, and technology scaling characteristics of two significant phenomena: 1) *data pattern dependence (DPD)*, where the minimum retention time of a DRAM cell is affected by data stored in other DRAM cells, and 2) *variable retention time (VRT)*, where the minimum retention time of a DRAM cell changes unpredictably over time. To this end, we built a flexible FPGA-based testing infrastructure to test DRAM chips, which has enabled a large amount of further experimental research in DRAM. Our ISCA 2013 paper's results using this infrastructure clearly demonstrate that DPD and VRT phenomena are significant issues that must be addressed for correct operation in DRAM-based systems and their effects are getting worse as DRAM scales to smaller technology node sizes. Our work also provides ideas on how to accurately identify data retention times in the presence of DPD and VRT, e.g., online profiling with error correcting codes, which later works examined and enabled. Most modern DRAM chips now incorporate ECC, especially to account for VRT effects [2].

This short retrospective provides a brief analysis of our ISCA 2013 paper and its impact. We describe why we did the work, what we found and its implications, what the findings as well as the infrastructure we built to discover them have enabled in later works, and our thoughts on what the future may bring.

I. Background

My group has been working on the DRAM refresh problem since 2010 and our major work RAIDR [3] was published at ISCA 2012. Our goal in RAIDR was to eliminate unnecessary refresh operations at low cost by refreshing each DRAM row only as frequently as required by the minimum data retention time of the row. As described in a separate retrospective in this issue, our RAIDR work demonstrated large performance improvements and energy savings with a simple memory controller based implementation. However, we were not satisfied with the simplistic retention time profiling mechanism assumed in RAIDR (which was also assumed in other prior works). RAIDR relied on accurate identification of the minimum data retention time of every DRAM cell, which we thought was a difficult task. We wanted to make such retention time profiling practical. So, we set out to rigorously understand the difficulty of DRAM data retention time identification using an empirical approach. No prior work at the time provided real data on the retention characteristics of state-of-the-art DRAM chips, let alone a detailed empirical analysis of major problems that make retention time profiling challenging and how DRAM technology scaling affects those challenges. In fact, no infrastructure to study these characteristics existed (or was available to us). We decided to build our own FPGA-based infrastructure to characterize real DRAM chips in a flexible manner so that we could change the refresh rate, data patterns, and other major parameters. This infrastructure, which took us more than a year to build and which we later open sourced as SoftMC [4, 5] and DRAM Bender [6, 7], enabled us (and others) to empirically study and understand many interesting characteristics of modern DRAM chips over the course of more than a decade.

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 and funding-wise. With close technical support from Intel (especially Chris Wilkerson, who is a co-author), we built our FPGA-based DDR3 DRAM testing infrastructure. Two of my students (also co-authors) and I spent part of the summer of 2012 at Intel to work closely with our collaborators. During this timeframe, we finalized the calibration and stabilization of our infrastructure. We performed many experiments to study both well-known properties of retention time characteristics (e.g., temperature dependence) as well as less well studied characteristics (e.g., DPD and VRT phenomena) of modern DRAM chips at a scale that was not reported before. We were especially interested in empirically understanding how technology scaling affected such characteristics, since it was clear that data retention and thus refresh was a major technology scaling challenge in DRAM, as indicated by prior and later works (e.g., [2, 8, 9]).

II. 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 very small sample sizes, and almost none of them examined modern DDR3 DRAM chips or technology scaling. Many device- or circuit-level works did not study DPD or VRT. No architecture- and system-level work to reduce refresh overhead discussed DPD or VRT. Our work enabled a new understanding and demonstrated the true difficulty of a major problem in DRAM technology scaling, by providing valuable data that was available nowhere else (at least publicly).

Our key results demonstrate that data retention times of modern DRAM chips are indisputably getting worse in newer-generation DRAM chips, indicating that refresh is becoming a larger problem with technology scaling. Ditto for DPD and VRT. For example, we showed that 1) the retention failure coverage of a given data pattern becomes smaller for newer-generation DRAM chips, 2) VRT is a widespread phenomenon in modern DRAM devices, causing significant dynamic changes in minimum retention time. These were the first results of their kind.

Our results indicated that many prior proposals (e.g., [3, 10–14]) that rely on accurate retention time identification to eliminate refreshes would not work reliably as they do not take into account DPD or VRT. They also put into question whether existing refresh rates are enough to guarantee error-free operation in DRAM chips being used in the field (especially in the presence of VRT). As DRAM technology scales, would it be easy to accurately determine retention times to ensure data integrity even if we maintained a conservative refresh rate for all DRAM cells?

Based on the understanding we developed, we proposed ideas and avenues for future work on how to tackle the DPD and VRT problems (§5.2 & §6.3 in [1]). We advocated the use of ECC in DRAM chips to detect and/or correct any retention errors that might not be identified after rigorous testing (offline or online). Most modern DRAM chips now incorporate ECC (see [15–17]), especially to account for VRT effects [2]. We also advocated the use of online profiling together with ECC to enable reliable identification of retention times, an approach later works rigorously investigated and enabled (e.g., [16–23]). As such, our ISCA 2013 paper enabled system-level techniques to overcome a major DRAM scaling challenge, an approach we call *system-DRAM co-design* [24, 25]. We believe developing such system-level techniques that can detect and exploit DRAM characteristics online, during system operation, will be increasingly valuable as such characteristics will become much more difficult to accurately determine and exploit due to technology scaling.

A key contribution of our work was the development of our flexible FPGA-based DRAM testing infrastructure, which was the first of its kind. It enabled a large amount of research into DRAM chips by enabling rigorous experimental study of real DRAM chip characteristics, including the rigorous study of the RowHammer vulnerability [6, 26–36], another major DRAM technology scaling problem. We discuss some new insights and studies enabled by this infrastructure in our RAIDR retrospective and our SoftMC [4] and DRAM Bender [6] works.

III. Influence on Later Works

Many later works (e.g., [16–23]) ensued to solve the DPD & VRT problems. Some provided a more detailed characterization of the DPD and VRT phenomena: [18] analyzed both DPD & VRT and examined the effectiveness of online profiling versus ECC of varying strength. AVATAR [19] provided heterogeneous refresh rates using a combination of online profiling, ECC, and memory scrubbing, working from the empirical observation that new VRT errors are discovered infrequently at a steady rate. PARBOR [20] introduced detailed DPD analyses and a new technique to efficiently detect data-dependent failures. REAPER [23] analyzed the DPD & VRT phenomena in newer LPDDR4 DRAM chips, demonstrating that the problems are getting worse, and developed the reach profiling technique to tolerate the two problems. We believe AVATAR & REAPER enabled practical ways of exploiting heterogeneous retention times.

ECC is mainstream in DRAM chips today [15–17]. We believe this is a direct result of the analysis that showed the prevalence and importance of VRT and the difficulty of handling VRT-caused retention errors due to their fundamentally unpredictable nature. A later work by Samsung & Intel engineers [2] described that ECC is needed to deal with VRT, just as our ISCA'13 paper advocated.

IV. Summary and Future Outlook

Our ISCA 2013 paper was a nice example of harmonious collaboration between academia and industry: Intel helped us build the infrastructure and both Intel & Samsung gave us significant technical feedback along with generous funding. Our paper also highlights the importance of investing into building infrastructure to analyze real chips: doing so enabled not only the new understanding developed in our work, but also many future works that analyzed various other DRAM characteristics (e.g., [6, 16–23, 26–42]) and uncovered fascinating undocumented capabilities in real DRAM chips, e.g., the ability to perform data copy/initialization and bitwise operations [43, 47], implement physical unclonable functions [48], and generate true random numbers [49, 50].

Since ISCA 2013, we have come a long way in understanding fundamental characteristics of DRAM devices and enabling practical solutions to overcome DRAM shortcomings. Yet, there is a lot more to be empirically discovered and understood in DRAM to solve the fundamental scaling, performance, and energy challenges of the technology (as shown by very recent works in 2022-2023, e.g. [6, 30–33, 45–47]), which can enable solutions also applicable to other technologies. We conclude that the future is bright in experimental memory systems research using real memory chips.

References

[1] J. Liu *et al.*, "An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms," *ISCA*, 2013.
[2] U. Kang *et al.*, "Co-Architecting Controllers and DRAM to Enhance DRAM Process Scaling," in *The Memory Forum*, 2014.
[3] J. Liu *et al.*, "RAIDR: Retention-Aware Intelligent DRAM Refresh," in *ISCA*, 2012.
[4] H. Hassan *et al.*, "SoftMC: A Flexible and Practical Open-Source Infrastructure for Enabling Experimental DRAM Studies," in *HPCA*, 2017.
[5] SoftMC Source Code, https://github.com/CMU-SAFARI/SoftMC
[6] A. Olgun *et al.*, "DRAM Bender: An Extensible and Versatile FPGA-based Infrastructure to Easily Test State-of-the-art DRAM Chips," *TCAD*, 2023.
[7] "DRAM Bender," https://github.com/CMU-SAFARI/DRAM-Bender
[8] J. A. Mandelman *et al.*, "Challenges and Future Directions for the Scaling of Dynamic Random-Access Memory (DRAM)," *IBM JRD*, 2002.
[9] W. Kim, "A 1.1V 16Gb DDR5 DRAM with Probabilistic-Aggressor Tracking, Refresh-Management Functionality, Per-Row Hammer Tracking, a Multi-Step Precharge, and Core-Bias Modulation for Security and Reliability Enhancement," in *ISSCC*, 2023.
[10] J.-H. Ahn *et al.*, "Adaptive self refresh scheme for battery operated high-density mobile DRAM applications," in *ASSCC*, 2006.
[11] T. Ohsawa *et al.*, "Optimizing the DRAM Refresh Count for Merged DRAM/Logic LSIs," in *ISLPED*, 1998.
[12] J. Kim and M. C. Papaefthymiou, "Dynamic memory design for low data-retention power," in *PATMOS*, 2000.
[13] R. Venkatesan *et al.*, "Retention-Aware Placement in DRAM (RAPID): Software Methods for Quasi-Non-Volatile DRAM," in *HPCA*, 2006.
[14] K. Yanagisawa, "Semiconductor Memory," 1988, U.S. Patent 4,736,344.
[15] M. Patel *et al.*, "Understanding and Modeling On-die Error Correction in Modern DRAM: An Experimental Study using Real Devices," in *DSN*, 2019.
[16] M. Patel *et al.*, "Bit-Exact ECC Recovery (BEER): Determining DRAM On-Die ECC Functions by Exploiting DRAM Data Retention Characteristics," in *MICRO*, 2020.
[17] M. Patel *et al.*, "HARP: Practically and Effectively Identifying Uncorrectable Errors in Main Memory Chips That Use On-Die ECC," in *MICRO*, 2021.
[18] S. Khan, "The Efficacy of Error Mitigation Techniques for DRAM Retention Failures: A Comparative Experimental Study," in *SIGMETRICS*, 2014.
[19] M. K. Qureshi *et al.*, "AVATAR: A Variable-Retention-Time (VRT) Aware Refresh for DRAM Systems," in *DSN*, 2015.
[20] S. Khan *et al.*, "PARBOR: An Efficient System-Level Technique to Detect Data Dependent Failures in DRAM," in *DSN*, 2016.
[21] S. Khan *et al.*, "A Case for Memory Content-Based Detection and Mitigation of Data-Dependent Failures in DRAM," *CAL*, 2016.
[22] S. Khan *et al.*, "Detecting and Mitigating Data-Dependent DRAM Failures by Exploiting Current Memory Content," in *MICRO*, 2017.
[23] M. Hassan *et al.*, "Reach Profiler (REAPER): Enabling the Mitigation of DRAM Retention Failures via Profiling at Aggressive Conditions," in *ISCA*, 2017.
[24] O. Mutlu, "Memory Scaling: A Systems Architecture Perspective," *IMW*, 2013.
[25] O. Mutlu and L. Subramanian, "Research Problems and Opportunities in Memory Systems," *SUPERFRI*, 2014.
[26] Y. Kim *et al.*, "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors," in *ISCA*, 2014.
[27] J. S. Kim *et al.*, "Revisiting RowHammer: An Experimental Analysis of Modern DRAM Devices and Mitigation Techniques," in *ISCA*, 2020.
[28] P. Frigo *et al.*, "TRRespass: Exploiting the Many Sides of Target Row Refresh," in *S&P*, 2020.
[29] H. Hassan *et al.*, "Uncovering In-DRAM RowHammer Protection Mechanisms: A New Methodology, Custom RowHammer Patterns, and Implications," in *MICRO*, 2021.
[30] A. G. Yağlıkçı *et al.*, "BlockHammer: Preventing RowHammer at Low Cost by Blacklisting Rapidly-Accessed DRAM Rows," in *HPCA*, 2021.
[31] A. G. Yağlıkçı *et al.*, "HiRA: Hidden Row Activation for Reducing Refresh Latency of Off-the-Shelf DRAM Chips," in *MICRO*, 2022.
[32] A. G. Yağlıkçı *et al.*, "Understanding RowHammer Under Reduced Wordline Voltage: An Experimental Study Using Real DRAM Devices," in *DSN*, 2022.
[33] A. Olgun *et al.*, "An Experimental Analysis of RowHammer in HBM2 DRAM Chips," in *DSN Disrupt*, 2023.
[34] H. Luo *et al.*, "RowPress: Amplifying Read Disturbance in Modern DRAM Chips," in *ISCA*, 2023.
[35] M. Farmani *et al.*, "RHAT: Efficient RowHammer-Aware Test for Modern DRAM Modules," in *ETS*, 2021.
[36] O. Mutlu and J. S. Kim, "RowHammer: A Retrospective," *IEEE TCAD*, 2019.
[37] O. Mutlu *et al.*, "Fundamentally Understanding and Solving RowHammer," in *ASP-DAC*, 2023.
[38] D. Lee *et al.*, "Adaptive-Latency DRAM: Optimizing DRAM Timing for the Common-Case," in *HPCA*, 2015.
[39] K. K. Chang *et al.*, "Understanding Latency Variation in Modern DRAM Chips: Experimental Characterization, Analysis, and Optimization," in *SIGMETRICS*, 2016.
[40] D. Lee *et al.*, "Design-induced Latency Variation in Modern DRAM Chips: Characterization, Analysis, and Latency Reduction Mechanisms," *POMACS*, 2017.
[41] J. Kim *et al.*, "Solar-DRAM: Reducing DRAM Access Latency by Exploiting the Variation in Local Bitlines," in *ICCD*, 2018.
[42] K. Chang *et al.*, "Understanding Reduced-Voltage Operation in Modern DRAM Devices: Experimental Characterization, Analysis, and Mechanisms," in *SIGMETRICS*, 2017.
[43] S. Ghose *et al.*, "What Your DRAM Power Models Are Not Telling You: Lessons from a Detailed Experimental Study," in *SIGMETRICS*, 2018.
[44] V. Seshadri *et al.*, "Fast Bulk Bitwise AND and OR in DRAM," *CAL*, 2015.
[45] V. Seshadri *et al.*, "Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology," in *MICRO*, 2017.
[46] A. Olgun *et al.*, "PiDRAM: A Holistic End-to-end FPGA-based Framework for Processing-in-DRAM," *TACO*, 2023.
[47] F. Gao *et al.*, "ComputeDRAM: In-Memory Compute Using Off-the-Shelf DRAMs," in *MICRO*, 2019.
[48] F. Gao *et al.*, "FracDRAM: Fractional Values in Off-the-Shelf DRAM," in *MICRO*, 2022.
[49] J. S. Kim *et al.*, "The DRAM Latency PUF: Quickly Evaluating Physical Unclonable Functions by Exploiting the Latency-Reliability Tradeoff in Modern Commodity DRAM Devices," in *HPCA*, 2018.
[50] J. Kim, "D-RaNGe: Using Commodity DRAM Devices to Generate True Random Numbers with Low Latency and High Throughput," in *HPCA*, 2019.
[51] A. Olgun *et al.*, "QUAC-TRNG: High-Throughput True Random Number Generation Using Quadruple Row Activation in Commodity DRAMs," in *ISCA*, 2021.

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

Onur Mutlu
ETH Zürich

***Abstract*—Our ISCA 2014 paper [1] provided the first scientific and detailed characterization, analysis, and real-system demonstration of what is now popularly known as the RowHammer phenomenon (or vulnerability) in modern commodity DRAM chips, which are used as main memory in almost all modern computing systems. It experimentally demonstrated that more than 80% of all DRAM modules we tested from the three major DRAM vendors were vulnerable to the RowHammer read disturbance phenomenon: one can predictably induce bitflips (i.e., data corruption) in real DRAM modules by repeatedly accessing a DRAM row and thus causing electrical disturbance to physically nearby rows. We showed that a simple unprivileged user-level program induced RowHammer bitflips in multiple real systems and suggested that a security attack can be built using this proof-of-concept to hijack control of the system or cause other harm. To solve the RowHammer problem, our paper examined seven different approaches (including a novel probabilistic approach that has very low cost), some of which influenced or were adopted in different industrial products.**

Many later works from various research communities examined RowHammer, building real security attacks, proposing new defenses, further analyzing the problem at various (e.g., device/circuit, architecture, and system) levels, and exploiting RowHammer for various purposes (e.g., to reverse-engineer DRAM chips). Industry has worked to mitigate the problem, changing both memory controllers and DRAM standards/chips. Two major DRAM vendors finally wrote papers on the topic in 2023, describing their current approaches to mitigate RowHammer. Research & development on RowHammer in both academia & industry continues to be very active and fascinating.

This short retrospective provides a brief analysis of our ISCA 2014 paper and its impact. We describe the circumstances that led to our paper, mention its influence on later works and products, describe the mindset change we believe it has helped enable in hardware security, and discuss our predictions for future.

I. Background and Circumstances

Our stumbling on the RowHammer problem and creation of its first scientific analysis happened as a result of a confluence of multiple factors. First, my group was working on DRAM technology scaling issues since late 2010. We were very interested in failure mechanisms that appear or worsen due to aggressive technology scaling. To study such issues (e.g., data retention errors [2]), we built an FPGA-based DRAM testing infrastructure [3] between 2011-2012, which we later open sourced as SoftMC [3, 4] and DRAM Bender [5, 6]. Second, around the same timeframe, we were investigating similar technology scaling issues in flash memory using real NAND flash chips [7, 8]. We knew read disturbance errors were significant in NAND flash memory [7–11] and were very interested in how prevalent they were in DRAM. Third, we were collaborating with Intel (e.g., [9]) to understand and solve DRAM technology scaling problems and build our DRAM infrastructure. Three of my students and I spent the summer of 2012 at Intel to work closely with our collaborators (two are co-authors): during this time, we finalized the calibration and stabilization of our infrastructure and had significant technical discussions and experimentation on DRAM scaling problems.

Although there was awareness of the RowHammer problem in industry in 2012 (see Footnote 1 in [1]), there was no comprehensive experimental analysis and detailed real-system demonstration of it. We believed it was critical to provide a rigorous scientific analysis using a wide variety of DRAM chips and scientifically establish major characteristics and prevalence of RowHammer. Hence, in the summer of 2012, we set out to use our DRAM testing infrastructure to analyze RowHammer. Our initial results showed how widespread the read disturbance problem was across the (at the time) recent DRAM chips we tested, so we studied the problem comprehensively and developed many solutions to it. The resulting paper was submitted to MICRO in May 2013 but was rejected. We strengthened the results, especially of the mitigation mechanisms and the number of tested chips, and made the analysis more comprehensive before it was accepted to ISCA 2014 (2 of the 6 reviewers still rejected it for interesting reasons).

II. Major Contribution and Influence

The major contribution of our paper is the exposure and detailed analysis of a fundamental hardware failure mechanism that breaks memory isolation in real systems and thus has huge implications on system reliability, security, and safety. Our paper is a comprehensive study of a major DRAM technology scaling problem, RowHammer, including its first scientific analysis, experimental characterization, real system demonstration, and solutions with their evaluation. To our knowledge, RowHammer is the first example of a hardware failure mechanism that creates a significant and widespread system security vulnerability [12–15], as our ISCA 2014 paper suggested.

Our work has had large influence on both industry & academia. Individual follow-on works are many to list here; we refer the reader to longer invited retrospectives we wrote [12–14]. We give major examples of influence, focusing on RowHammer's effect on the collective mindset of security research and major industry milestones related to RowHammer.

RowHammer Attacks & Mindset Shift in Hardware Security. Our demonstration that one can easily and predictably induce bitflips in commodity DRAM chips using a real user-level program enabled a major mindset shift in hardware security. It showed that general-purpose hardware is fallible in a very widespread manner and its problems are exploitable. Tens of works (see [13, 14]) built directly on our work to exploit RowHammer bitflips to develop many attacks that compromise system integrity and confidentiality, starting from the first RowHammer exploit by Google Project Zero in 2015 [16, 17] to recent works in 2022-2023 (e.g., [18, 19]). These attacks showed increasingly sophisticated ways by which an unprivileged attacker can exploit RowHammer bitflips to circumvent memory protection and gain complete control of a system (e.g., [16, 20–28]), gain access to confidential data (e.g., [18, 19, 29]), or maliciously destroy the safety and accuracy of a system, e.g., an otherwise accurate machine learning inference engine (e.g., [30, 31]). The mindset enabled by RowHammer bitflips caused a renewed interest in hardware security research, enticing many researchers to deeply understand hardware's inner workings and find new vulnerabilities. Thus, hardware security issues have become mainstream discussion in top security & architecture venues, some having sessions entitled RowHammer.

RowHammer Defenses. Tens of works proposed mitigations against RowHammer, some of which were inspired by the solutions we discussed in our ISCA 2014 paper. To date, the search for more efficient and low-cost RowHammer solutions continues. We refer the reader to our prior overview papers [13, 14, 32] and more recent works in 2023 (e.g., [33–35]).

RowHammer Analyses. Our paper initiated works at both architectural & circuit/device-levels to better understand RowHammer and reverse-engineer DRAM chips, to develop better models, defenses, and attacks (see [13, 14]). Our ISCA'20 work [36] revisited RowHammer, comprehensively analyzed of 1580 DRAM chips of three different types from at least two generations, showing that RowHammer has gotten much worse with technology scaling & existing solutions are not effective at future vulnerability levels.

Industry Reaction: Attacks, Analyses, and Mitigations. Folks developing industrial memory testing programs immediately included RowHammer tests, e.g., in memtest86 [37], citing our work. Industry needed to immediately protect RowHammer-vulnerable chips already in the field, so almost all system vendors increased refresh rates; a solution we examined in our paper and deemed costly for performance and energy, yet it was the only practical lever that could be used in the field. Apple publicly acknowledged our work in their security release [38] that announced higher refresh rates to mitigate RowHammer. Intel designed memory controllers that performed probabilistic activations (i.e., pTRR [39, 40]), similar to our PARA solution [1]. DRAM vendors modified the DRAM standard to introduce TRR (target row refresh) mechanisms [39] and claimed their new DDR4 chips to be RowHammer-free [39, 41]. This bold claim was later refuted by our TRRespass work [39] in 2020, which introduced the many-sided RowHammer attack to circumvent internal protection mechanisms added to the DRAM chips. Our later work, Uncovering TRR [41] showed that one can almost completely reverse-engineer and thus easily bypass RowHammer mitigations employed in all tested DRAM chips, i.e., RowHammer solutions in DRAM chips are broken. The analysis done by our two major works in 2020 [36, 39] caused the industry to reorganize the RowHammer task group at JEDEC, which produced two white papers on mitigating RowHammer [42, 43]. Nine years after our paper, in 2023, two major DRAM vendors, SK Hynix and Samsung, finally wrote papers [44, 45] on the RowHammer problem, describing their solutions. Several of these industry solutions build on the probabilistic & access-counter-based solution approaches our ISCA 2014 paper introduced.

Major Internet and cloud systems companies also took a deep interest in RowHammer as it can greatly impact their system security, dependability, and availability. Multiple works from Google, e.g., by Google Project Zero in 2015 [16, 17] and Half Double in 2021-2022 [46] directly built on our paper to demonstrate attacks in real systems. Researchers from Microsoft have developed deeper analyses of RowHammer [47], along with new RowHammer attacks [48] and defenses (e.g., [48–51]).

III. Summary and Future Outlook

Since 2012-2014, RowHammer vulnerability has become much worse due to technology scaling: without mitigation, one can now induce RowHammer bitflips with orders of magnitude smaller number of activations (e.g., ∼10K) and cause much higher rates of errors in cutting-edge DRAM chips [36, 41]. Sophisticated attacks are continuously developed to circumvent the mitigations employed in real DRAM chips. Fortunately, we have also come a long way in further understanding and better mitigating the RowHammer vulnerability. The industry is now (hopefully) fully aware of the importance of the problem and of avoiding bitflips. Unfortunately, an efficient and complete(!) secure solution is not found yet. The solution space poses a rich area of tradeoffs in terms of security, performance, power/energy, cost/complexity. All solutions forego some desirable properties in favor of others. As such, a critical direction for the future is to find solutions superior to what we have today. We believe system-DRAM cooperation [14, 52] will be important to enabling complete solutions. We also believe it is critical to deeply understand the properties of RowHammer under many different conditions so that we can develop effective solutions that work under all circumstances. Unfortunately, we do not yet fully understand many facets of RowHammer (see [14, 53–55]).

DRAM technology scaling will continue to create problems that will exacerbate the bitflips and the resulting robustness (i.e., safety/security/reliability) problems. Our ISCA' 2023 paper on RowPress [55] provides the first scientific and detailed characterization, analysis, and real-system demonstration of yet another read disturbance mechanism in DRAM. What other fascinating problems will we see and can we completely solve them efficiently? Will we ever be free of bitflips at the system and application levels?

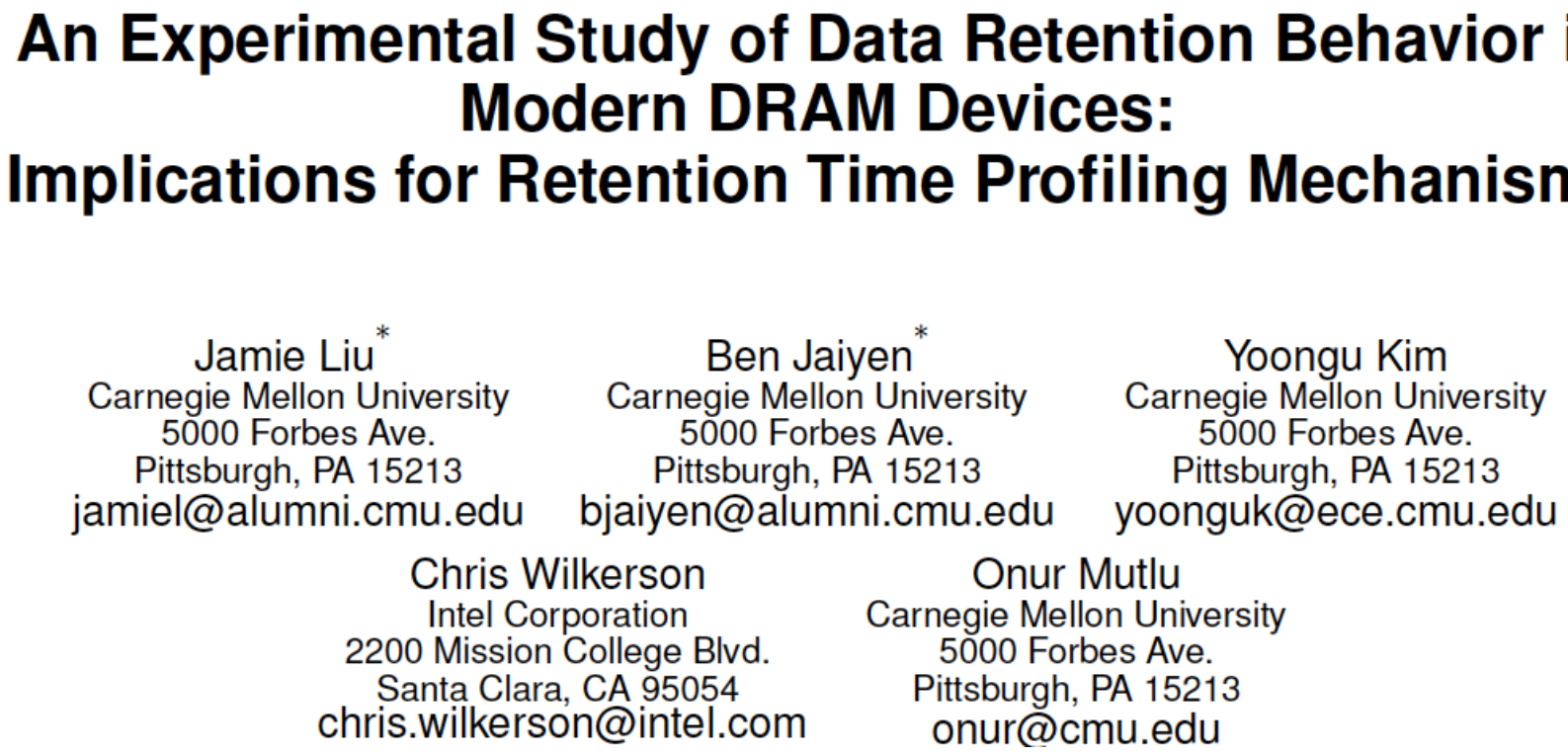
References

[1] Y. Kim *et al.*, "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors," in *ISCA*, 2014.
[2] J. Liu *et al.*, "An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms," *ISCA*, 2013.
[3] H. Hassan *et al.*, "SoftMC: A Flexible and Practical Open-Source Infrastructure for Enabling Experimental DRAM Studies," in *HPCA*, 2017.
[4] SoftMC Source Code, https://github.com/CMU-SAFARI/SoftMC/
[5] A. Olgun *et al.*, "DRAM Bender: An extensible and versatile FPGA-based Infrastructure to Easily Test State-of-the-art DRAM Chips," *TCAD*, 2023.
[6] "DRAM Bender," https://github.com/CMU-SAFARI/DRAM-Bender
[7] Y. Cai *et al.*, "Error Patterns in MLC NAND Flash Memory: Measurement, Characterization, and Analysis," in *DATE*, 2012.
[8] Y. Cai *et al.*, "Error Analysis and Retention-Aware Error Management for NAND Flash Memory," *ITJ*, 2013.
[9] Y. Cai *et al.*, "Program Interference in MLC NAND Flash Memory: Characterization, Modeling, and Mitigation," in *ICCD*, 2013.
[10] Y. Cai *et al.*, "Read Disturb Errors in MLC NAND Flash Memory: Characterization, Mitigation, and Recovery," in *DSN*, 2015.
[11] Y. Cai *et al.*, "Error Characterization, Mitigation, and Recovery in Flash Memory Based Solid-State Drives," *Proc. IEEE*, 2017.
[12] O. Mutlu, "The RowHammer Problem and Other Issues we may Face as Memory Becomes Denser," in *DATE*, 2017.
[13] O. Mutlu and J. Kim, "RowHammer: A Retrospective," *IEEE TCAD Special Issue on Top Picks in Hardware and Embedded Security*, 2019.
[14] O. Mutlu *et al.*, "Fundamentally Understanding and Solving RowHammer," in *ASP-DAC*, 2023.
[15] T. Dullien, "Security, Moore's Law, and the Anomaly of Cheap Complexity," in *CCDCOE*, 2018, https://www.youtube.com/watch?v=q98toLaAIX8
[16] M. Seaborn and T. Dullien, "Exploiting the DRAM rowhammer Bug to Gain Kernel Privileges," http://googleprojectzero.blogspot.com.tr/2015/03/exploiting-dram-rowhammer-bug-to-gain.html, 2015.
[17] M. Seaborn and T. Dullien, "Exploiting the DRAM Rowhammer Bug to Gain Kernel Privileges," *Black Hat*, 2015.
[18] A. S. Rakin *et al.*, "DeepSteal: Advanced Model Extractions Leveraging Efficient Weight Stealing in Memories," in *S&P*, 2022.
[19] K. Mus *et al.*, "Jolt: Recovering TLS Signing Keys via Rowhammer Faults," in *S&P*, 2023.
[20] D. Gruss *et al.*, "Rowhammer.js: A Remote Software-Induced Fault Attack in Javascript," in *DIMVA*, 2016.
[21] V. van der Veen *et al.*, "Drammer: Deterministic Rowhammer Attacks on Mobile Platforms," in *CCS*, 2016.
[22] Y. Xiao *et al.*, "One Bit Flips, One Cloud Flops: Cross-VM Row Hammer Attacks and Privilege Escalation," in *USENIX Security*, 2016.
[23] K. Razavi *et al.*, "Flip Feng Shui: Hammering a Needle in the Software Stack," *USENIX Security*, 2016.
[24] A. Tatar *et al.*, "Throwhammer: Rowhammer Attacks Over the Network and Defenses," in *USENIX ATC*, 2018.
[25] M. Lipp *et al.*, "Nethammer: Inducing Rowhammer Faults Through Network Requests," arXiv:1805.04956, 2018.
[26] L. Cojocar *et al.*, "Exploiting Correcting Codes: On the Effectiveness of ECC Memory Against Rowhammer Attacks," in *S&P*, 2019.
[27] F. de Ridder *et al.*, "SMASH: Synchronized Many-Sided Rowhammer Attacks from JavaScript," in *USENIX Security*, 2021.
[28] P. Jattke *et al.*, "Blacksmith: Scalable Rowhammering in the Frequency Domain," in *S&P*, 2022.
[29] A. Kwong *et al.*, "RAMBleed: Reading Bits in Memory Without Accessing Them," in *S&P*, 2020.
[30] S. Hong *et al.*, "Terminal Brain Damage: Exposing the Graceless Degradation in Deep Neural Networks Under Hardware Fault Attacks," in *SS*, 2019.
[31] F. Yao *et al.*, "Deephammer: Depleting the Intelligence of Deep Neural Networks Through Targeted Chain of Bit Flips," in *USENIX Security*, 2020.
[32] A. G. Yağlıkçı *et al.*, "BlockHammer: Preventing RowHammer at Low Cost by Blacklisting Rapidly-Accessed DRAM Rows," in *HPCA*, 2021.
[33] M. Marazzi *et al.*, "ProTRR: Principled yet Optimal In-DRAM Target Row Refresh," in *S&P*, 2023.
[34] M. Wi *et al.*, "SHADOW: Preventing Row Hammer in DRAM with Intra-Subarray Row Shuffling," in *HPCA*, IEEE, 2023.
[35] J. Juffinger *et al.*, "CSI: Rowhammer-Cryptographic Security and Integrity against Rowhammer (to appear)," in *S&P*, 2023.
[36] J. S. Kim *et al.*, "Revisiting RowHammer: An Experimental Analysis of Modern DRAM Devices and Mitigation Techniques," in *ISCA*, 2020.
[37] PassMark Software, "MemTest86: The Original Industry Standard Memory Diagnostic Utility," http://www.memtest86.com/#ourniesnooting.htm, 2015.
[38] Apple Inc., "About the Security Content of Mac EFI Security Update 2015-001," https://support.apple.com/en-us/H1 204934, 2015.
[39] P. Frigo *et al.*, "TRRespass: Exploiting the Many Sides of Target Row Refresh," in *S&P*, 2020.
[40] M. Kaczmarski, "Thoughts on Intel Xeon E5-2600 v2 Product Family Performance Optimisation – Component Selection Guidelines," http://infobazy.edu.pl/Uploads/Prezentacje/dd3cb4-Kaczmarski-Optymalna.pdf, page 13, 2014.
[41] H. Hassan *et al.*, "Uncovering In-DRAM RowHammer Protection Mechanisms: A New Methodology, Custom RowHammer Patterns, and Implications," in *MICRO*, 2021.
[42] JEDEC, *JEP300-1: Near-Term DRAM Level RowHammer Mitigation*, 2021.
[43] JEDEC, *JEP301-1: System Level RowHammer Mitigation*, 2021.
[44] W. Kim, "A 1.1V 16Gb DDR5 DRAM with Probabilistic-Aggressor Tracking, Refresh-Management Functionality, Per-Row Hammer Tracking, a Multi-Step Precharge, and Core-Bias Modulation for Security and Reliability Enhancement," in *ISSCC*, 2023.
[45] S. Hong *et al.*, "HSAC: Low-Cost Rowhammer Mitigation Using In-DRAM Stochastic and Approximate Counting Algorithm," arXiv:2302.03591, 2023.
[46] A. Kogler *et al.*, "Half-Double: Hammering From the Next Row Over," in *USENIX Security*, 2022.
[47] L. Cojocar *et al.*, "Are We Susceptible to Rowhammer? An End-to-End Methodology for Cloud Providers," in *S&P*, 2020.
[48] K. Loughlin *et al.*, "MOESI-Prime: Preventing Coherence-Induced Hammering in Commodity Workloads," in *ISCA*, 2022.
[49] T. Bennett *et al.*, "Panopticon: A Complete In-DRAM Rowhammer Mitigation," in *DMSec*, 2021.
[50] K. Loughlin *et al.*, "Stop! Hammer Time: Rethinking Our Approach to Rowhammer Mitigations," in *HotOS*, 2021.
[51] S. Saroiu and A. Wolman, "How to Configure Row-Sampling-Based Rowhammer Defenses," *DRAMSec*, 2022.
[52] O. Mutlu, "Memory Scaling: A Systems Architecture Perspective," in *IMW*, 2013.
[53] L. Orosa, "A Deeper Look into RowHammer's Sensitivities: Experimental Analysis of Real DRAM Chips and Implications on Future Attacks and Defenses," in *MICRO*, 2021.
[54] A. G. Yağlıkçı *et al.*, "Understanding RowHammer Under Reduced Wordline Voltage: An Experimental Study Using Real DRAM Devices," in *DSN*, 2022.
[55] H. Luo *et al.*, "RowPress: Amplifying Read Disturbance in Modern DRAM Chips," in *ISCA*, 2023.

https://people.inf.ethz.ch/omutlu/pub/RowHammer_50YearsOfISCA-Retrospective_isca23.pdf

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)]
  Selected to the ISCA-50 25-Year Retrospective Issue covering 1996-2020 in 2023 (Retrospective (pdf) Full Issue).



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

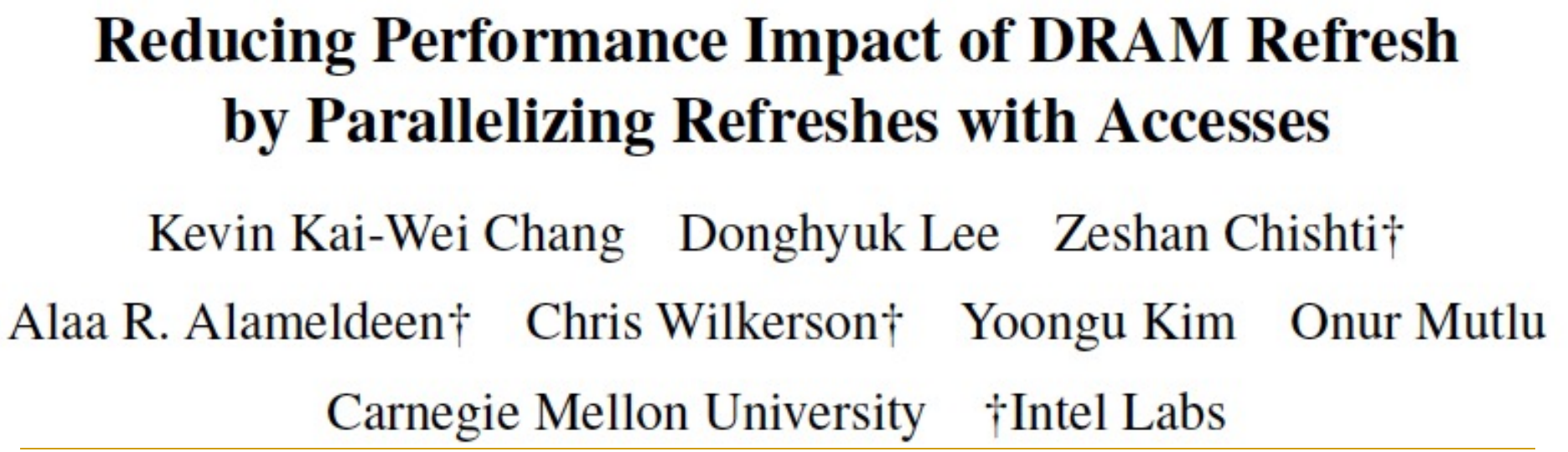
Onur Mutlu
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
onur@cmu.edu

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"
  *Proceedings of the 20th International Symposium on High-Performance Computer Architecture (**HPCA**)*, Orlando, FL, February 2014.
  [Summary] [Slides (pptx) (pdf)]



Reducing Performance Impact of DRAM Refresh by Parallelizing Refreshes with Accesses

Kevin Kai-Wei Chang Donghyuk Lee Zeshan Chishti†

Alaa R. Alameldeen† Chris Wilkerson† Yoongu Kim Onur Mutlu

Carnegie Mellon University †Intel Labs

Refresh-Access Parallelization

- A. Giray Yaglıkcı, Ataberk Olgun, Minesh Patel, Haocong Luo, Hasan Hassan, Lois Orosa, Oguz Ergin, and Onur Mutlu,
  "HiRA: Hidden Row Activation for Reducing Refresh Latency of Off-the-Shelf DRAM Chips,"
  *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 Yağlıkçı[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)*

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

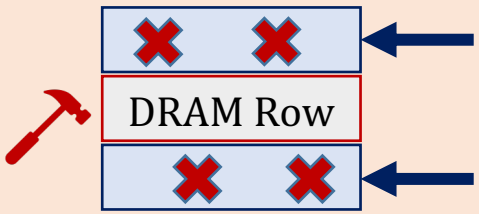
SAFARI

ETH zürich CESGA TOBB ETÜ University of Economics & Technology

Two Main Types of DRAM Refresh

1 **Periodic Refresh:** Periodically **restores** the charge DRAM cells leak **over time**

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

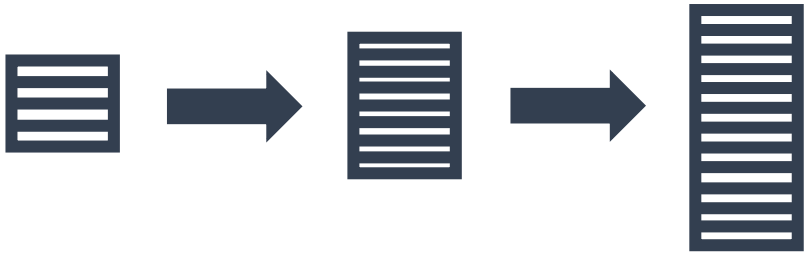


Preventive Refresh: Mitigates RowHammer by **refreshing physically nearby rows** of a repeatedly accessed row

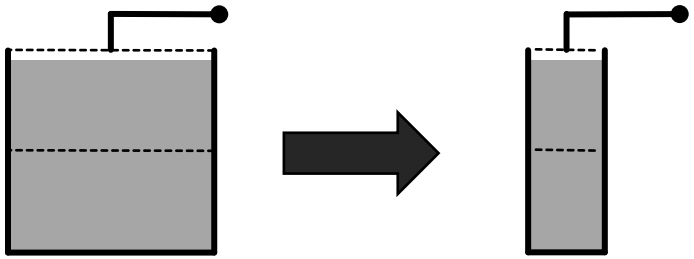
Periodic Refresh

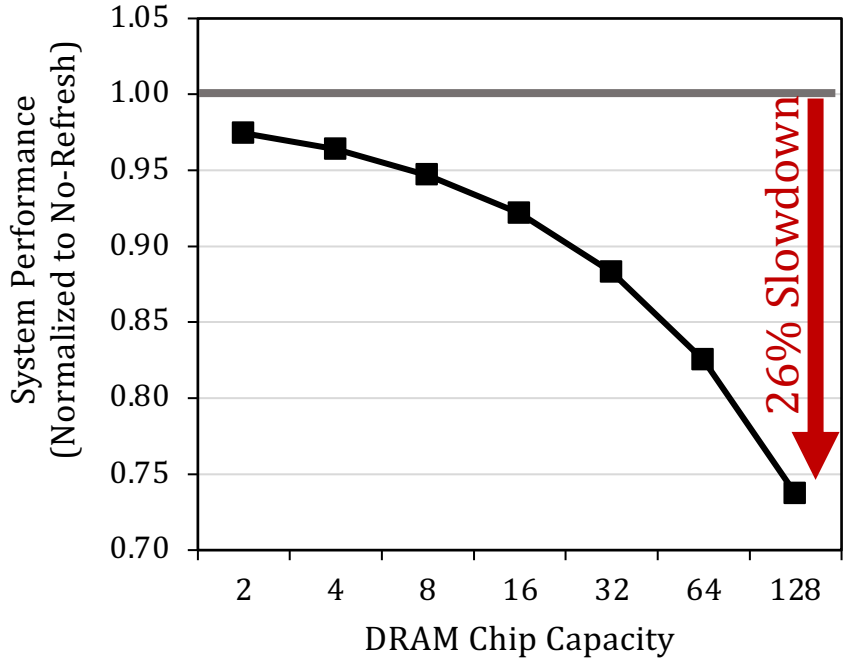
with Increasing DRAM Chip Density

A **larger capacity** chip has **more rows to be refreshed**



A **smaller** cell stores **less charge**

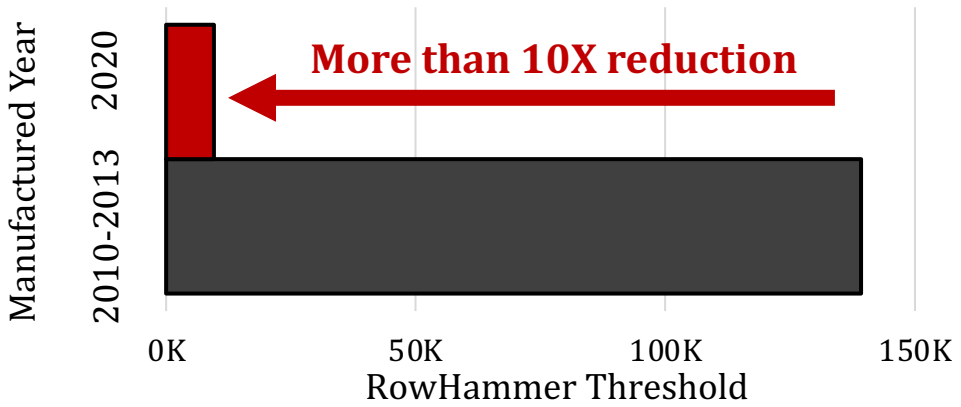




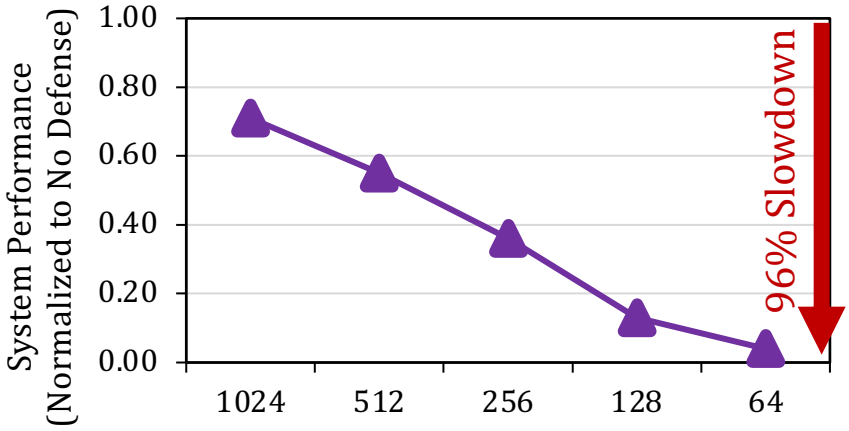
More periodic refresh operations incur
larger performance overhead as DRAM **chip density increases**

RowHammer and Preventive Refresh
with Increasing DRAM Chip Density

RowHammer vulnerability worsens
as DRAM **chip density increases**



The Minimum Activation Count needed to induce the first RowHammer bit flip



The Minimum Activation Count needed to induce the first RowHammer bit flip

Preventive refresh operations need to be performed
more aggressively as DRAM **chip density increases**

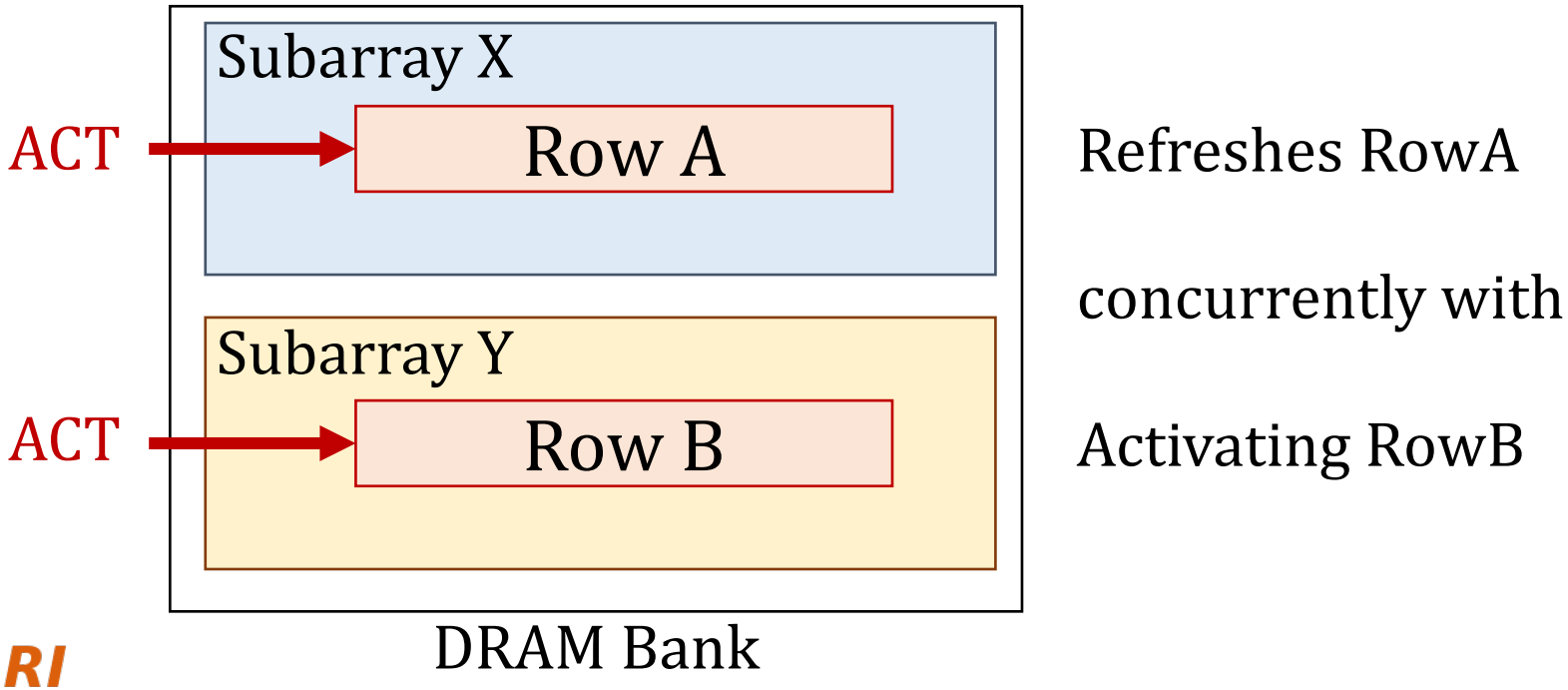
Goal and Key Idea

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**

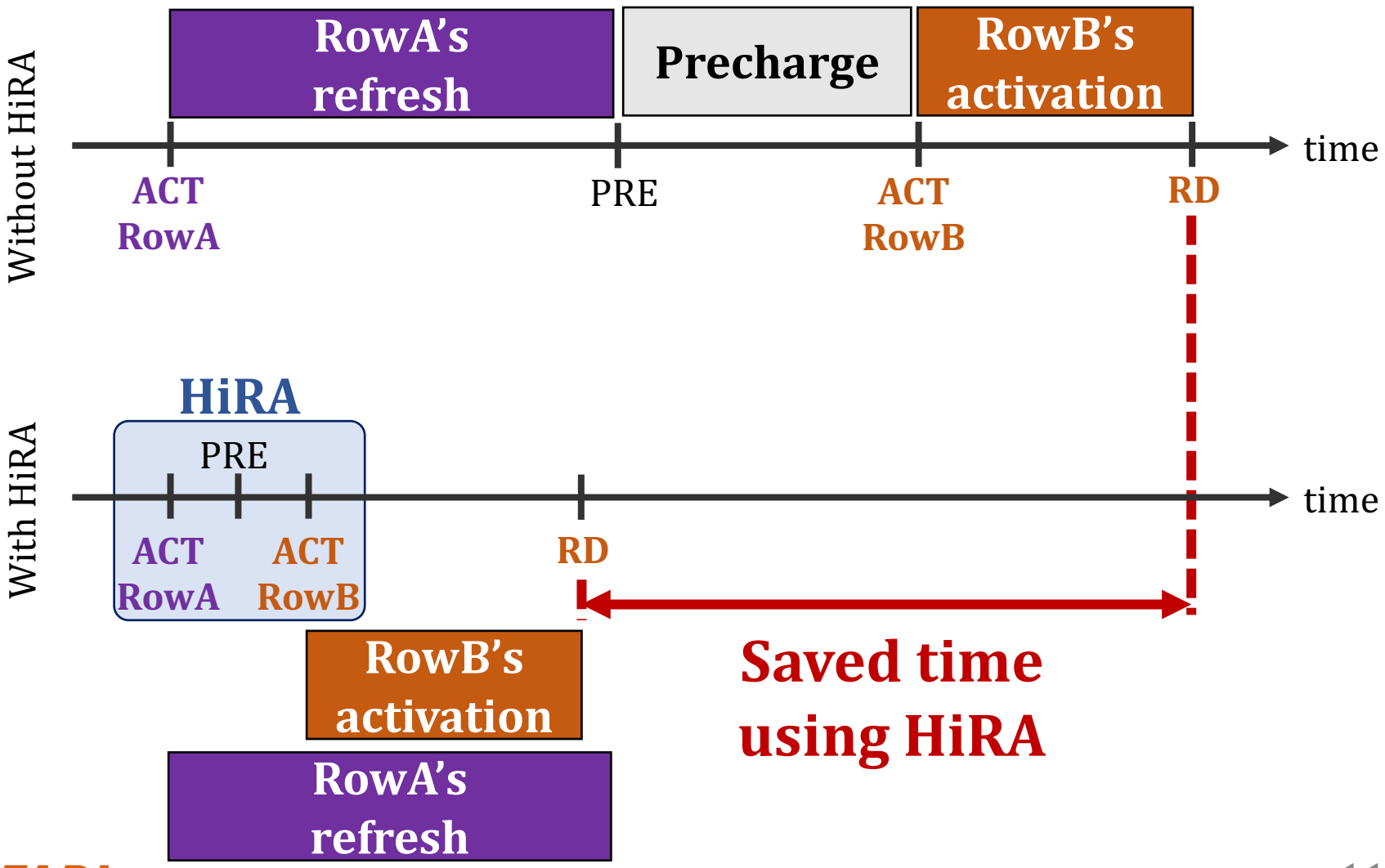
HiRA: Hidden Row Activation – Key Insight

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

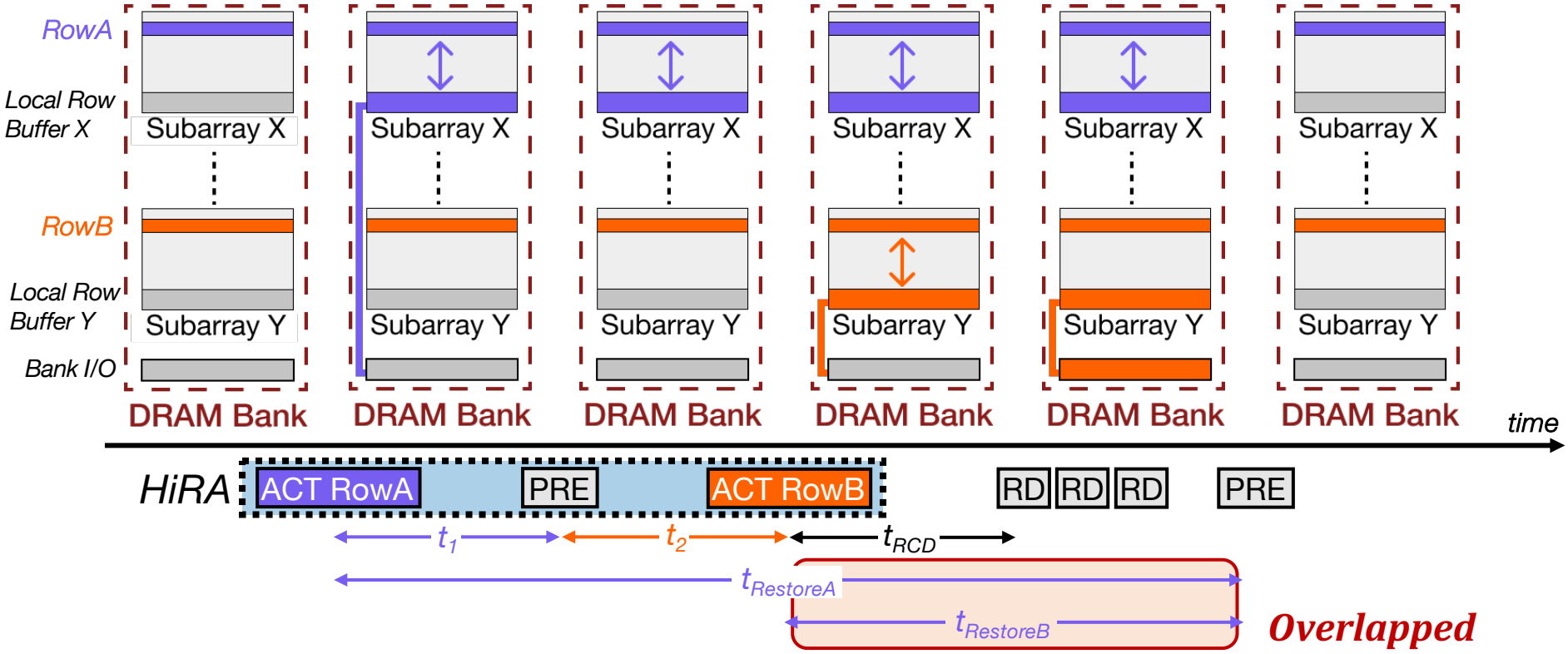


HiRA: Hidden Row Activation

Refresh RowA concurrently with **Activating RowB**



HiRA Operation



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

DRAM Testing Infrastructure

FPGA-based SoftMC (Xilinx Virtex UltraScale+ XCU200)

Fine-grained control over **DRAM commands**,
timing parameters (±1.5ns), and **temperature (±0.1°C)**

*Hassan et al., "SoftMC: A Flexible and Practical Open-Source Infrastructure for Enabling Experimental DRAM Studies," in HPCA, 2017. [Available on GitHub: https://github.com/CMU-SAFARI/SoftMC]

HiRA in Off-the-Shelf DRAM Chips: Key Result 1

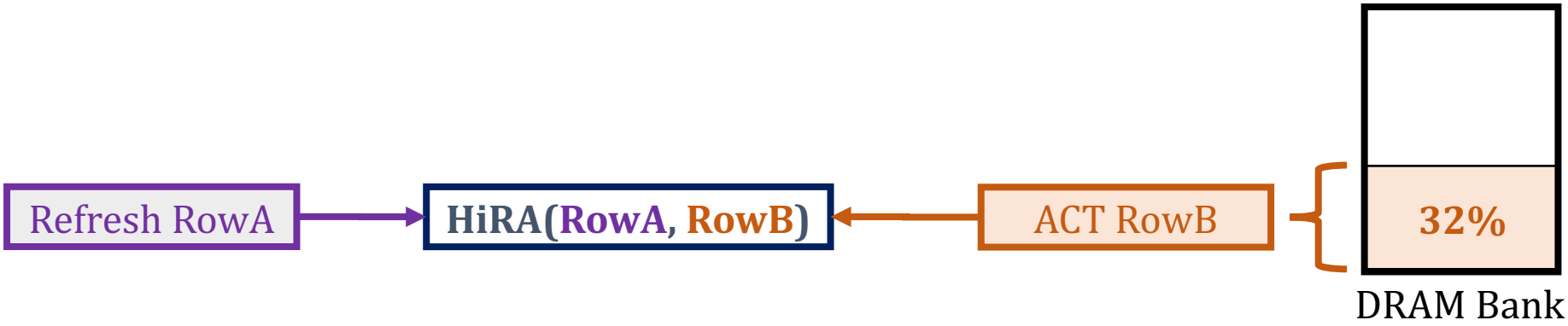
- HiRA works in **56 off-the-shelf DRAM chips** from **SK Hynix**

Table 4: Characteristics of the tested DDR4 DRAM modules.

Module Label	Module Vendor	Module Identifier / Chip Identifier	Freq (MT/s)	Date Code	Chip Cap.	Die Rev.	Chip Org.	HiRA Coverage Min.	HiRA Coverage Avg.	HiRA Coverage Max.	Norm. N_{RH} Min.	Norm. N_{RH} Avg.	Norm. N_{RH} Max.
A0	G.SKILL	DWCW (Partial Marking)*	2400	42-20	4Gb	B	x8	24.8%	25.0%	25.5%	1.75	1.90	2.52
A1		F4-2400C17S-8GNT [39]						24.9%	26.6%	28.3%	1.72	1.94	2.55
B0	Kingston	H5AN8G8NDJR-XNC	2400	48-20	4Gb	D	x8	25.1%	32.6%	36.8%	1.71	1.89	2.34
B1		KSM32RD8/16HDR [87]						25.0%	31.6%	34.9%	1.74	1.91	2.51
C0	SK Hynix	H5ANAG8NAJR-XN	2400	51-20	4Gb	F	x8	25.3%	35.3%	39.5%	1.47	1.89	2.23
C1		HMAA4GU6AJR8N-XN [109]						29.2%	38.4%	49.9%	1.09	1.88	2.27
C2								26.5%	36.1%	42.3%	1.49	1.96	2.58

* 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 4: Characteristics of the tested DDR4 DRAM modules.

Module Label	Module Vendor	Module Identifier / Chip Identifier	Freq (MT/s)	Date Code	Chip Cap.	Die Rev.	Chip Org.	HiRA Coverage Min.	HiRA Coverage Avg.	HiRA Coverage Max.	Norm. N_{RH} Min.	Norm. N_{RH} Avg.	Norm. N_{RH} Max.
A0	G.SKILL	DWCW (Partial Marking)*	2400	42-20	4Gb	B	x8	24.8%	25.0%	25.5%	1.75	1.90	2.52
A1		F4-2400C17S-8GNT [39]						24.9%	26.6%	28.3%	1.72	1.94	2.55
B0	Kingston	H5AN8G8NDJR-XNC	2400	48-20	4Gb	D	x8	25.1%	32.6%	36.8%	1.71	1.89	2.34
B1		KSM32RD8/16HDR [87]						25.0%	31.6%	34.9%	1.74	1.91	2.51
C0	SK Hynix	H5ANAG8NAJR-XN	2400	51-20	4Gb	F	x8	25.3%	35.3%	39.5%	1.47	1.89	2.23
C1		HMAA4GU6AJR8N-XN [109]						29.2%	38.4%	49.9%	1.09	1.88	2.27
C2								26.5%	36.1%	42.3%	1.49	1.96	2.58

* 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 Yağlıkçı[1] Ataberk Olgun[1] 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)*

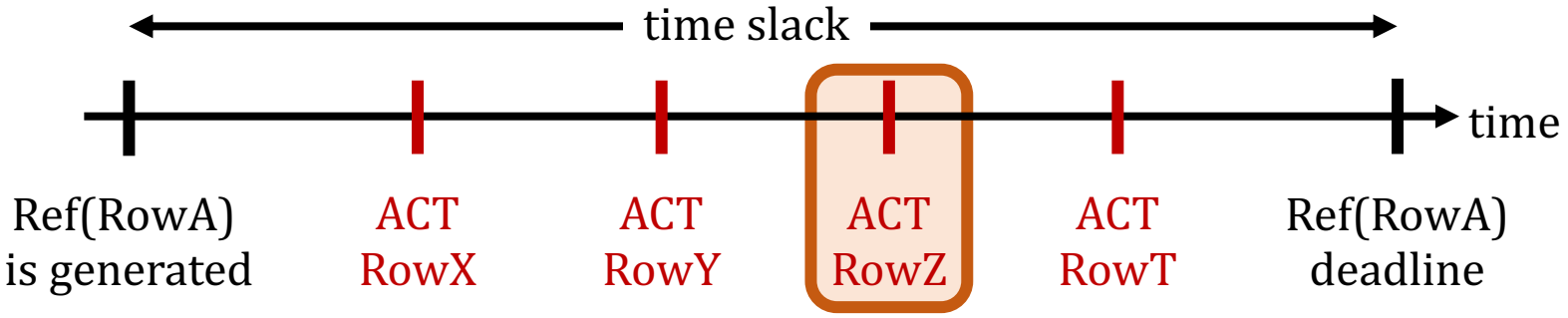
DRAM is the building block of modern main memory systems. DRAM cells must be periodically refreshed to prevent data loss. Refresh operations degrade system performance by interfering with memory accesses. As DRAM chip density increases with technology node scaling, refresh operations also increase because: 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-

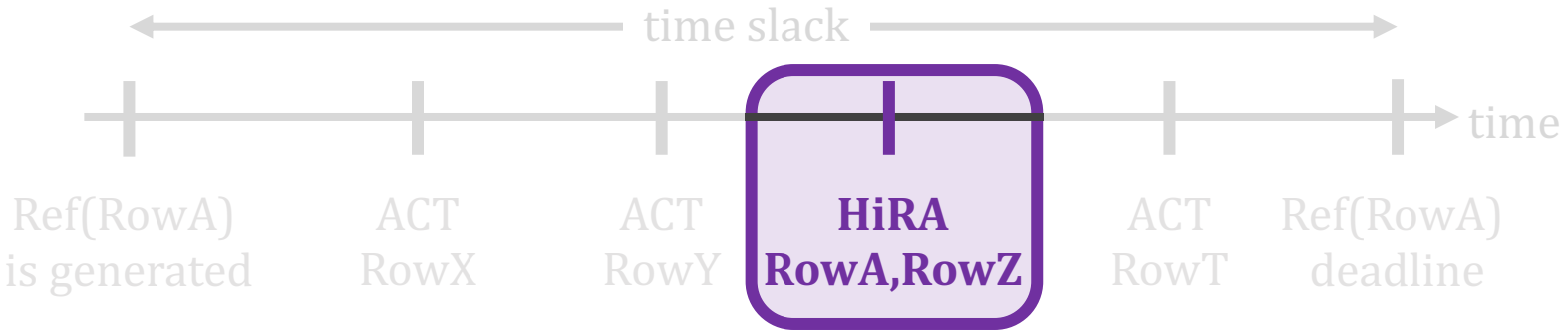
https://arxiv.org/pdf/2209.10198.pdf

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**



HiRA-MC: HiRA Memory Controller

1. Generates each **periodic refresh** and **RowHammer-preventive refresh** **with a deadline**

2. **Buffers** each **refresh request** and **performs** the refresh request **until** the **deadline**

3. 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 Yağlıkçı[1] Ataberk Olgun[1] 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)*

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-

https://arxiv.org/pdf/2209.10198.pdf

Performance Evaluation

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

- **System Configuration**:

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

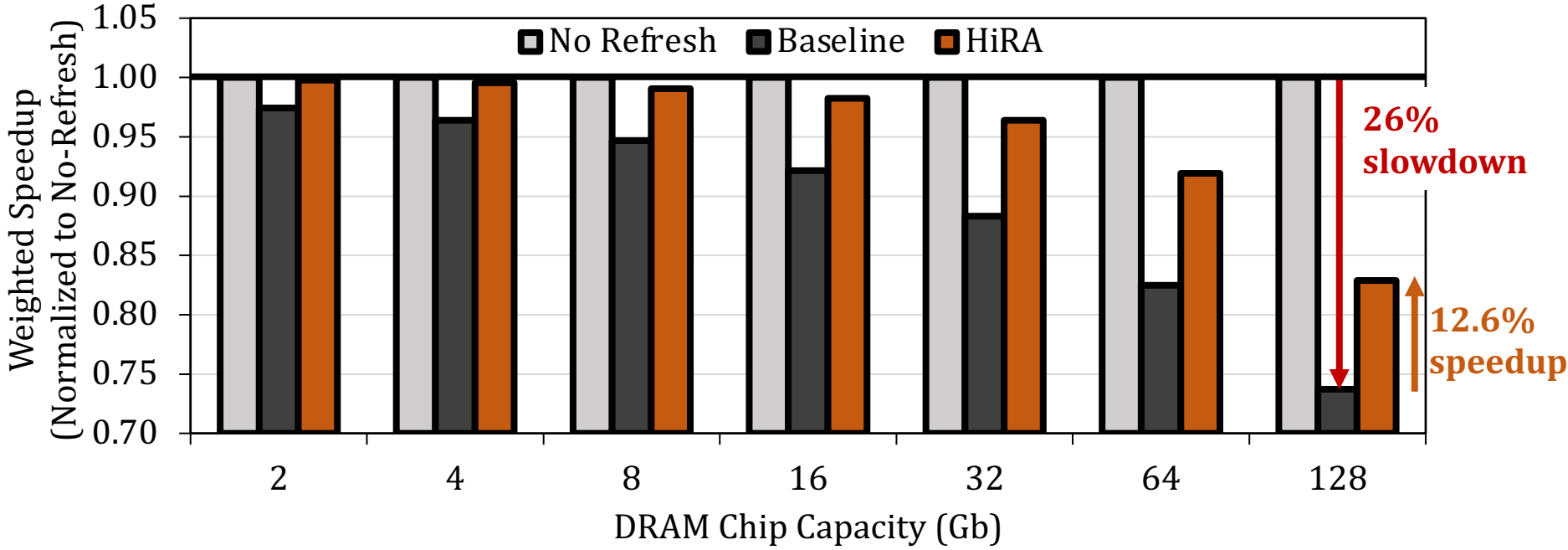
- **Workloads**: **125** different **8-core** multiprogrammed workloads from the SPEC2006 benchmark suite

- **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)

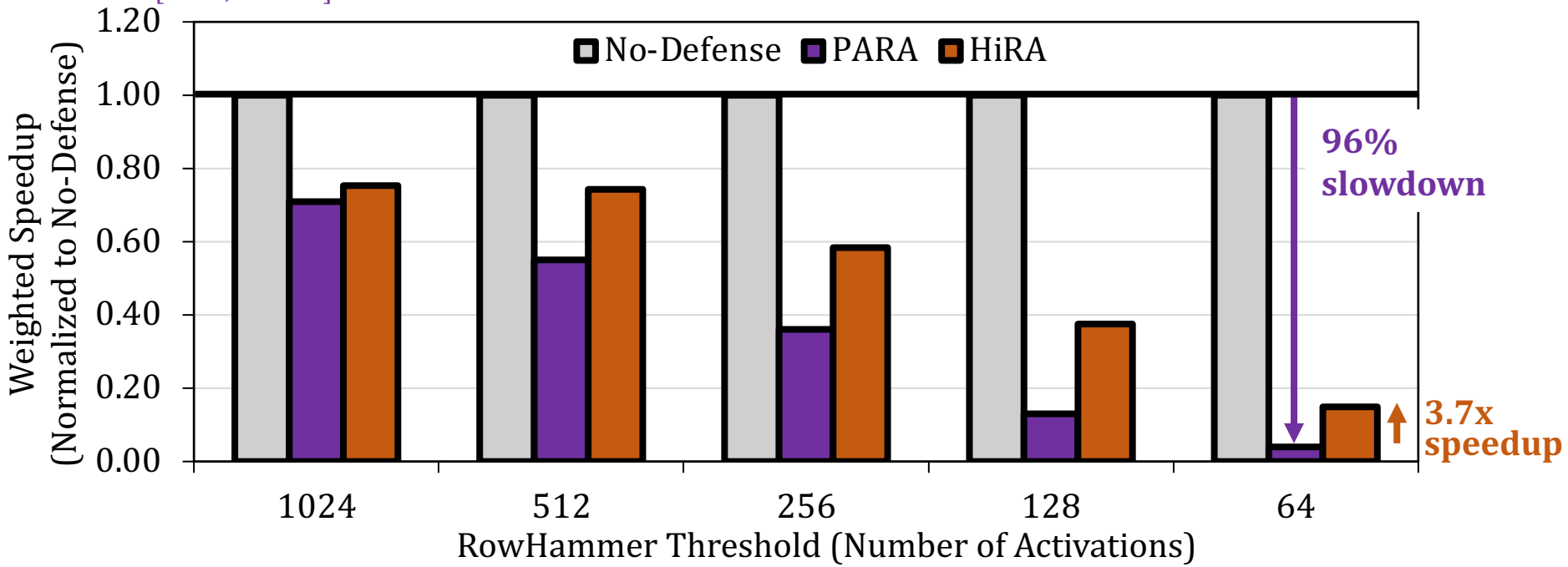


Periodic refreshes cause *significant* **(26%) performance overhead**

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**



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**

More in the Full Paper

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

A. Giray Yağlıkçı[1] Ataberk Olgun[1] 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)*

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-

https://arxiv.org/pdf/2209.10198.pdf

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

ETH zürich CESGA TOBB ETÜ University of Economics & Technology

Refresh-Access Parallelization

- A. Giray Yaglıkcı, Ataberk Olgun, Minesh Patel, Haocong Luo, Hasan Hassan, Lois Orosa, Oguz Ergin, and Onur Mutlu,
  "HiRA: Hidden Row Activation for Reducing Refresh Latency of Off-the-Shelf DRAM Chips,"
  *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]

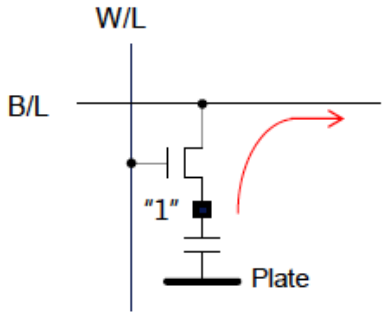
HiRA: Hidden Row Activation for Reducing Refresh Latency of Off-the-Shelf DRAM Chips

A. Giray Yağlıkçı[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)*

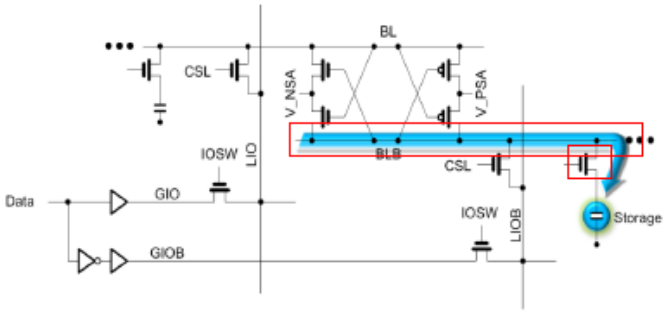
Industry Is Writing Papers About It, Too

DRAM Process Scaling Challenges

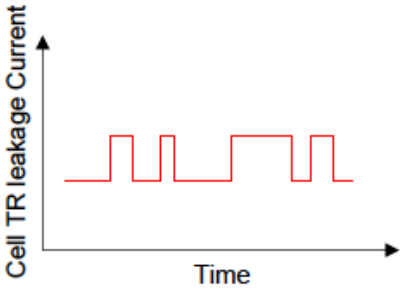
- **Refresh**
  - Difficult to build high-aspect ratio cell capacitors decreasing cell capacitance
  - Leakage current of cell access transistors increasing
- **tWR**
  - Contact resistance between the cell capacitor and access transistor increasing
  - On-current of the cell access transistor decreasing
  - Bit-line resistance increasing
- **VRT**
  - Occurring more frequently with cell capacitance decreasing



Refresh



tWR



VRT

Call for Intelligent Memory Controllers

DRAM Process Scaling Challenges

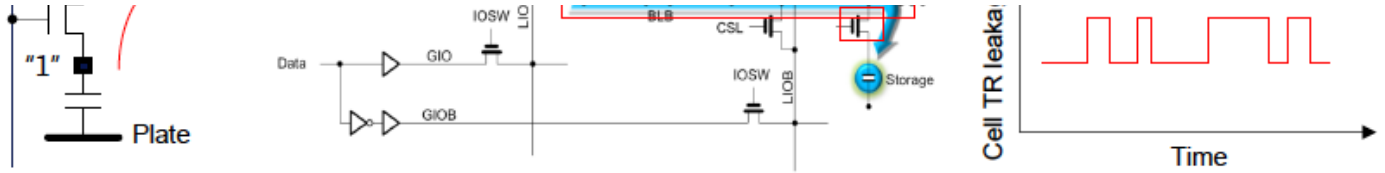
- **Refresh**
  - Difficult to build high-aspect ratio cell capacitors decreasing cell capacitance

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

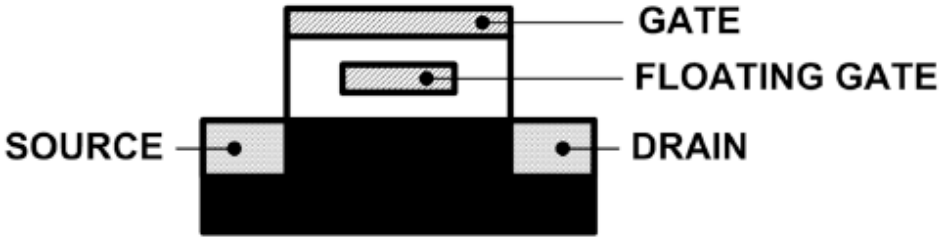
*Samsung Electronics, Hwasung, Korea / *Samsung Electronics, San Jose / **Intel*

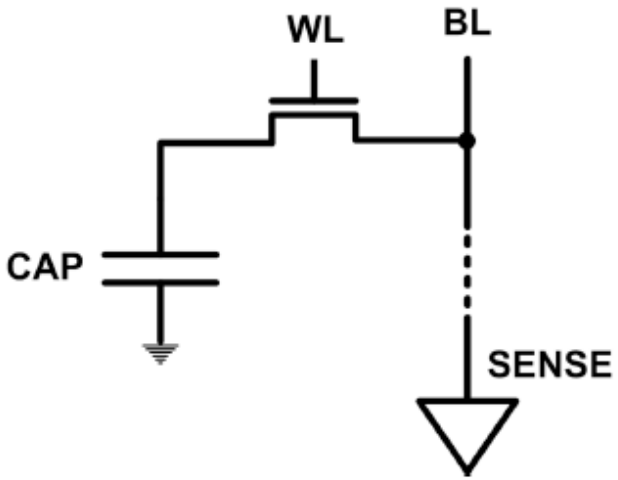


Data Retention in Flash Memory

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





TECHSPOT

TRENDING ▾ REVIEWS ▾ FEATURES ▾ DOWNLOADS ▾ PRODUCT FINDER ▾ FORUMS ▾ TEC

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** nsistently slower than norma **as slow as 30MB/s** whereas **newly-written files** ones used in benchmarks, perform as fast as new – aro **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.

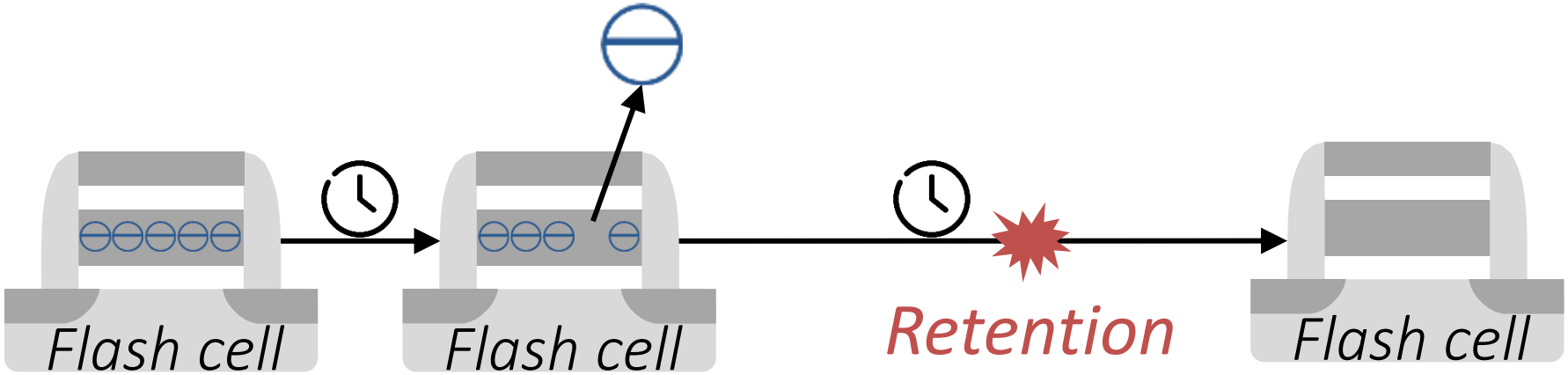
Reference: (May 5, 2015) Per Hansson, "*When SSD Performance Goes Awry*"
http://www.techspot.com/article/997-samsung-ssd-read-performance-degradation/

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

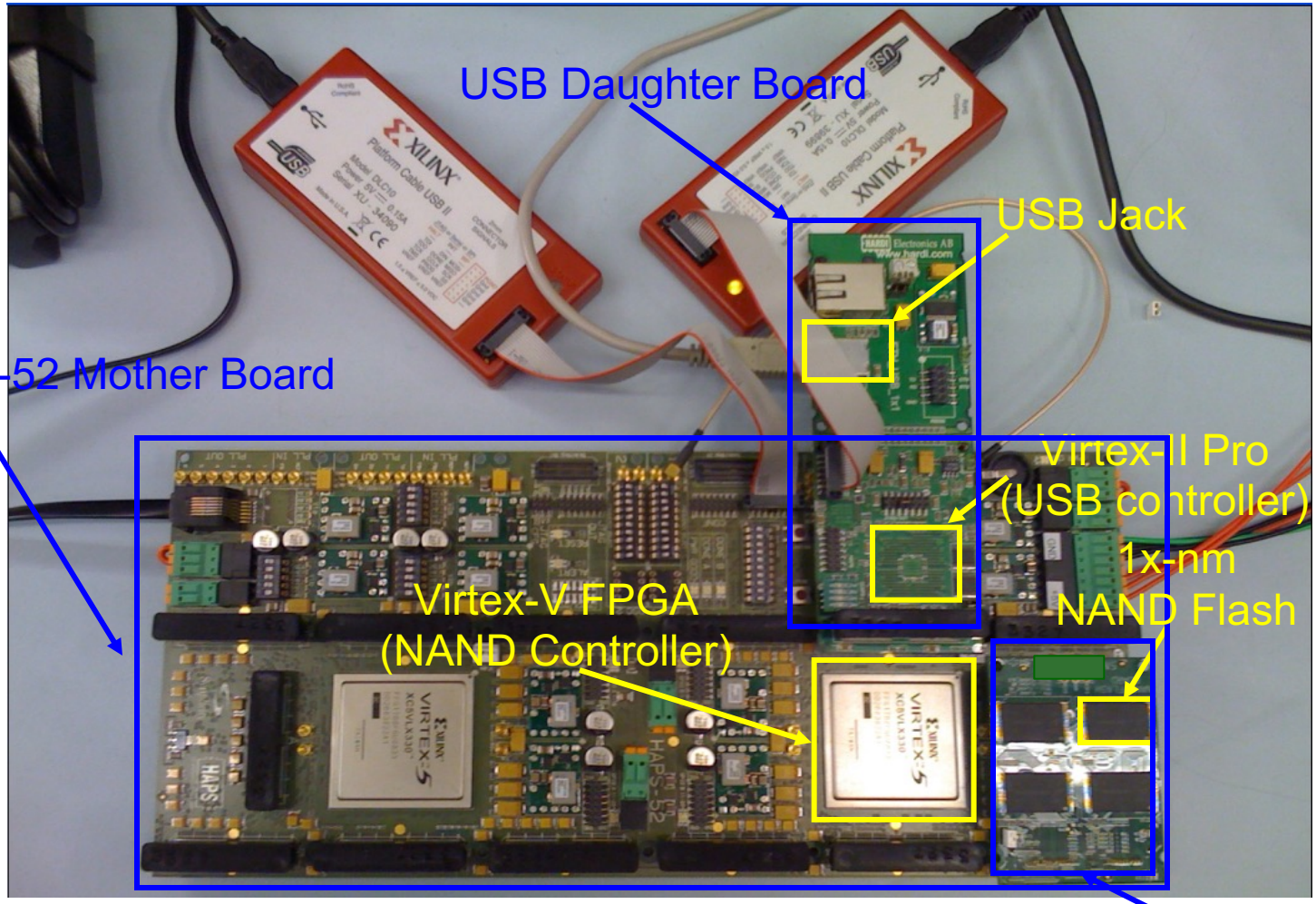
NAND Flash Error Types

- 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

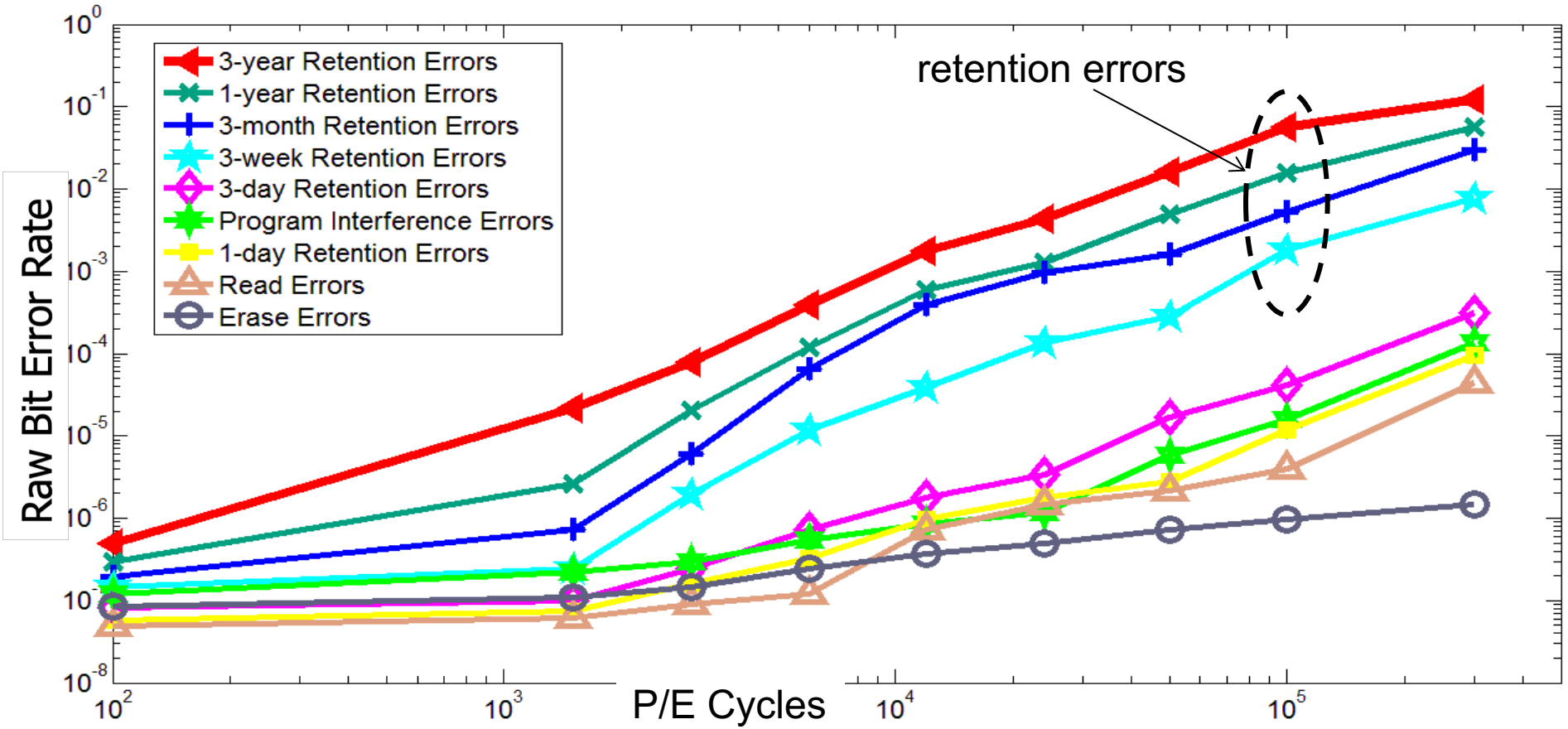
Flash Experimental Testing Platform



[DATE 2012, ICCD 2012, DATE 2013, ITJ 2013, ICCD 2013, SIGMETRICS 2014, HPCA 2015, DSN 2015, MSST 2015, JSAC 2016, HPCA 2017, DFRWS 2017, PIEEE 2017, HPCA 2018, SIGMETRICS 2018]

Cai+, "Error Characterization, Mitigation, and Recovery in Flash Memory Based Solid State Drives," Proc. IEEE 2017.

Observations: Flash Error Analysis

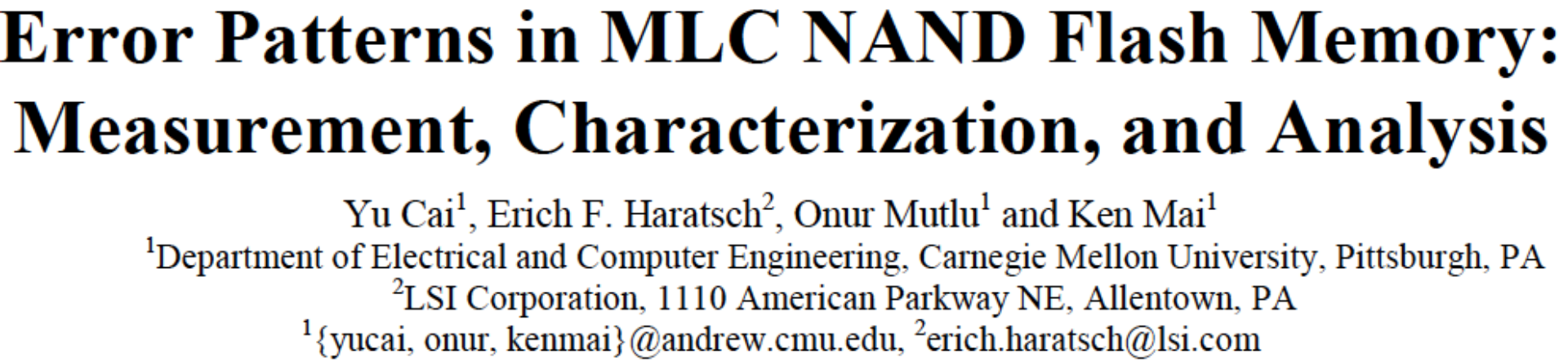


- Raw bit error rate increases exponentially with P/E cycles
- **Retention errors are dominant** (>99% for 1-year ret. time)
- Retention errors increase with retention time requirement

Cai et al., Error Patterns in MLC NAND Flash Memory, DATE 2012.

More on Flash Error Analysis

- Yu Cai, Erich F. Haratsch, Onur Mutlu, and Ken Mai, **"Error Patterns in MLC NAND Flash Memory: Measurement, Characterization, and Analysis"** *Proceedings of the Design, Automation, and Test in Europe Conference (**DATE**)*, Dresden, Germany, March 2012. Slides (ppt)



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

Yu Cai[1], Erich F. Haratsch[2], Onur Mutlu[1] and Ken Mai[1]
[1]Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA
[2]LSI Corporation, 1110 American Parkway NE, Allentown, PA
[1]{yucai, onur, kenmai}@andrew.cmu.edu, [2]erich.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
- Cai et al. "Flash Correct-and-Refresh: Retention-Aware Error Management for Increased Flash Memory Lifetime", ICCD 2012.

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 Cai[1], Gulay Yalcin[2], Onur Mutlu[1], Erich F. Haratsch[3], Adrian Cristal[2], Osman S. Unsal[2] and Ken Mai[1]
[1]DSSC, Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA
[2]Barcelona Supercomputing Center, C/Jordi Girona 29, Barcelona, Spain
[3]LSI Corporation, 1110 American Parkway NE, Allentown, PA

More on Flash Error Analysis [Intel Tech J'13]

- 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"
  Intel Technology Journal (***ITJ***) *Special Issue on Memory Resiliency*, Vol. 17, No. 1, May 2013.

Flash Memory Data Retention Analysis

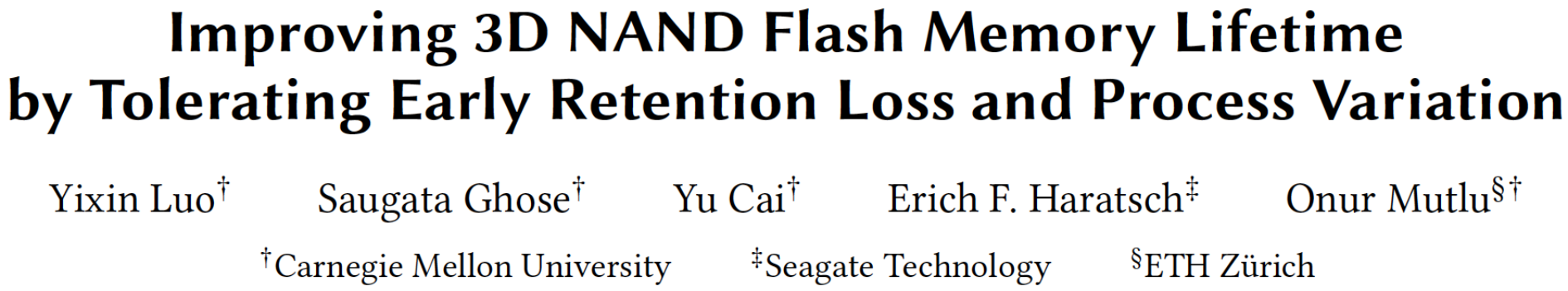
- Yu Cai, Yixin Luo, Erich F. Haratsch, Ken Mai, and Onur Mutlu,
  "Data Retention in MLC NAND Flash Memory: Characterization, Optimization and Recovery"
  *Proceedings of the 21st International Symposium on High-Performance Computer Architecture (**HPCA**)*, Bay Area, CA, February 2015.
  [Slides (pptx) (pdf)] [Poster (pdf)]
  Best paper session.

Data Retention in MLC NAND Flash Memory: Characterization, Optimization, and Recovery

Yu Cai, Yixin Luo, Erich F. Haratsch*, Ken Mai, Onur Mutlu
Carnegie Mellon University, *LSI Corporation
yucaicai@gmail.com, yixinluo@cs.cmu.edu, erich.haratsch@lsi.com, {kenmai, omutlu}@ece.cmu.edu

3D Flash Data Retention [SIGMETRICS'18]

- Yixin Luo, Saugata Ghose, Yu Cai, Erich F. Haratsch, and Onur Mutlu, **"Improving 3D NAND Flash Memory Lifetime by Tolerating Early Retention Loss and Process Variation"**
  *Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (**SIGMETRICS**)*, Irvine, CA, USA, June 2018.
  [Abstract]
  [POMACS Journal Version (same content, different format)]
  [Slides (pptx) (pdf)]



Many Errors and Their Mitigation [PIEEE'17]

Proceedings of the IEEE, Sept. 2017

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

More Up-to-date Version

- Yu Cai, Saugata Ghose, Erich F. Haratsch, Yixin Luo, and Onur Mutlu, **"Errors in Flash-Memory-Based Solid-State Drives: Analysis, Mitigation, and Recovery"**
  Invited Book Chapter in Inside Solid State Drives, 2018.
  [Preliminary arxiv.org version]

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

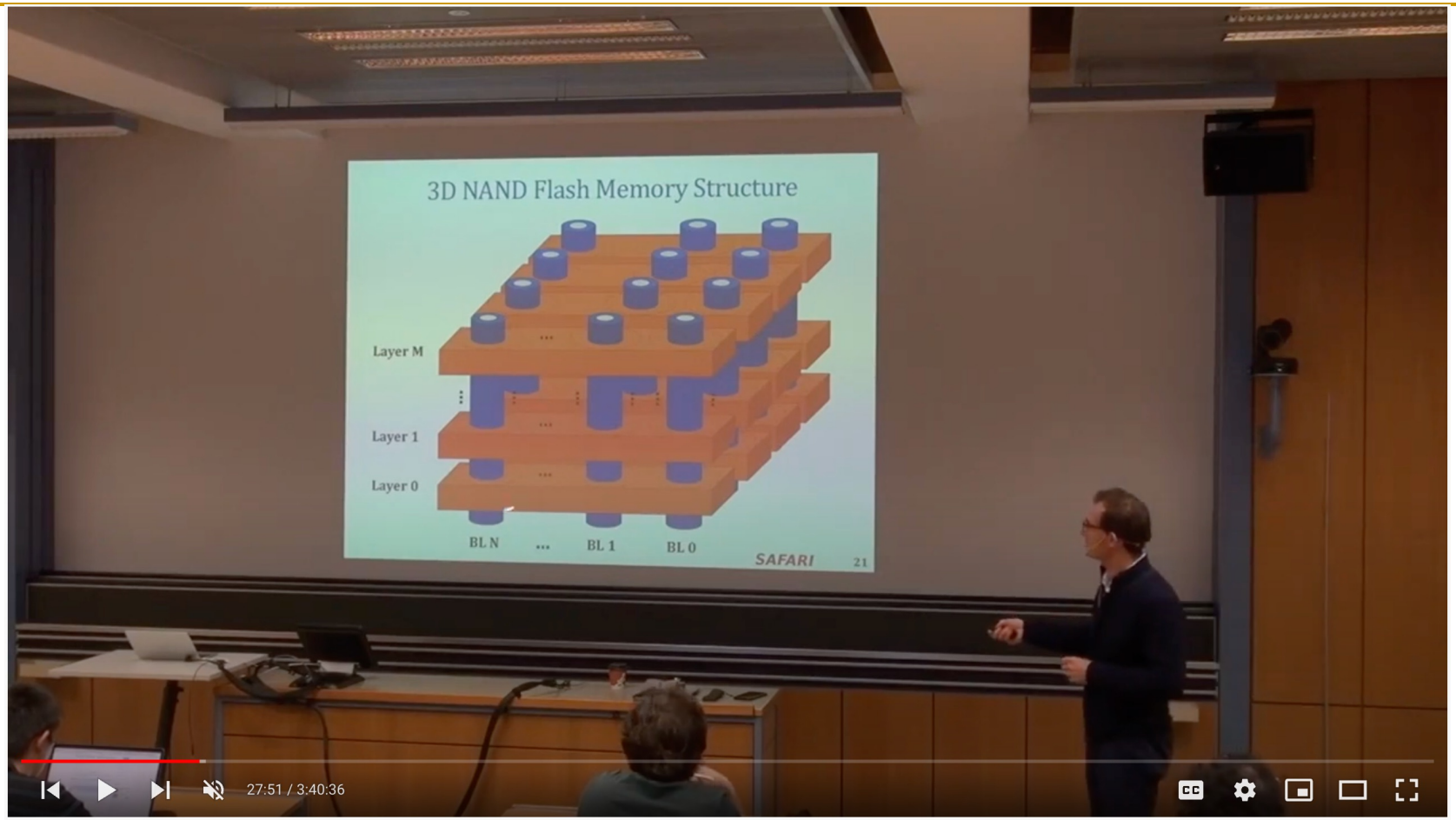
YU CAI, SAUGATA GHOSE
Carnegie Mellon University

ERICH F. HARATSCH
Seagate Technology

YIXIN LUO
Carnegie Mellon University

ONUR MUTLU
ETH Zürich and Carnegie Mellon University

Complete Lecture on Flash Memory & SSDs



ETH ZÜRICH HAUPTGEBÄUDE

Computer Architecture - Lecture 26: Flash Memory and Solid-State Drives (ETH Zürich, Fall 2020)

1,610 views • Dec 31, 2020

39 0 SHARE SAVE ...

Onur Mutlu Lectures
19.1K subscribers

ANALYTICS EDIT VIDEO

https://www.youtube.com/watch?v=rninK6KWBeM&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=47

We Will Dig Deeper More In This Course

"Good ideas are a dime a dozen"

"Making them work is oftentimes the real contribution"

Computer Architecture

Lecture 8: Data Retention and Memory Refresh

A. Giray Yaglikci
Prof. Onur Mutlu
ETH Zürich

Fall 2022

20 October 2023

Backup Slides

HiRA: Hidden Row Activation

for Reducing Refresh Latency of Off-the-Shelf DRAM Chips

Backup Slides

Abdullah Giray Yağlıkçı

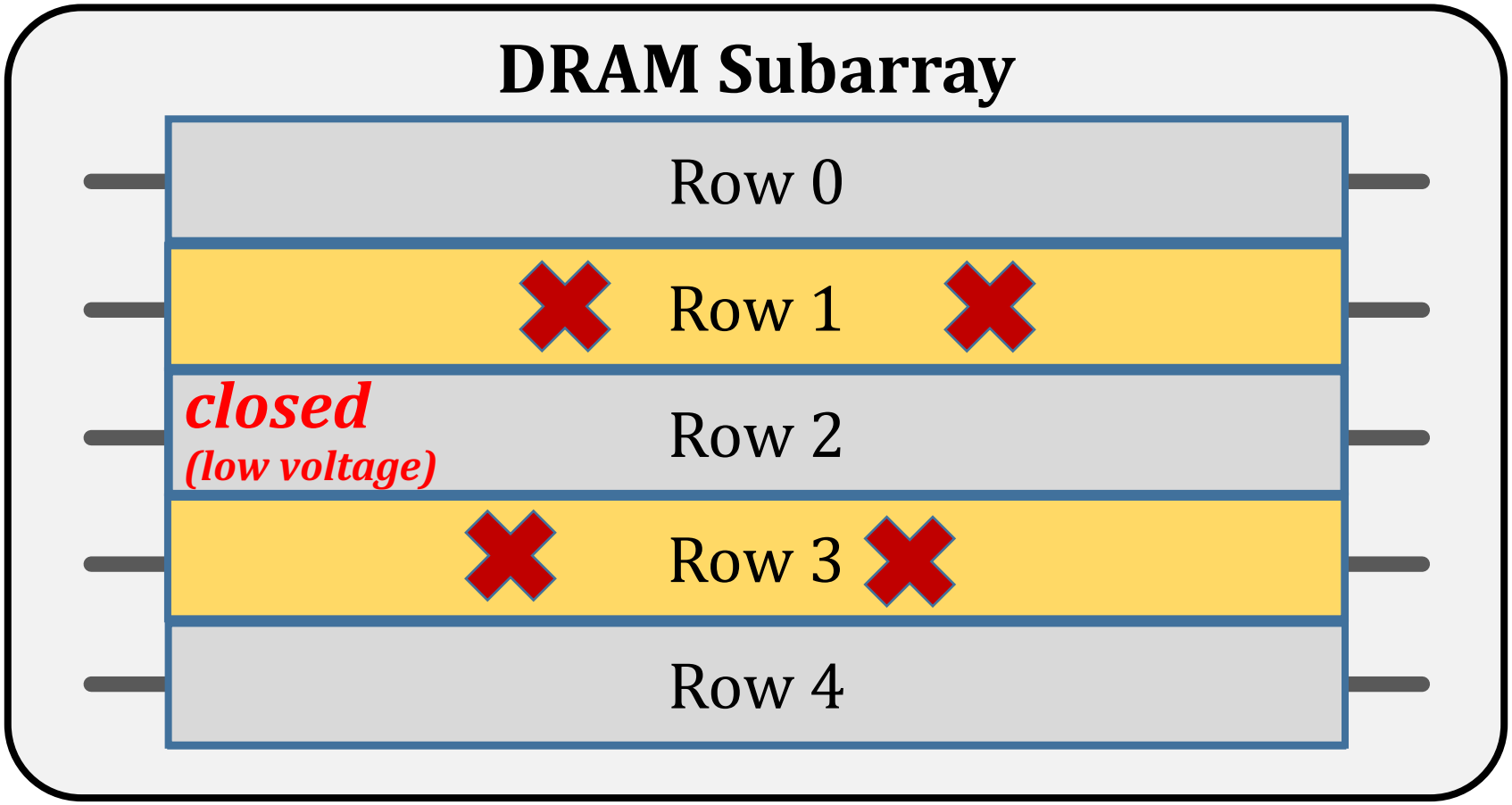
Ataberk Olgun Minesh Patel Haocong Luo Hasan Hassan

Lois Orosa Oğuz Ergin Onur Mutlu

SAFARI

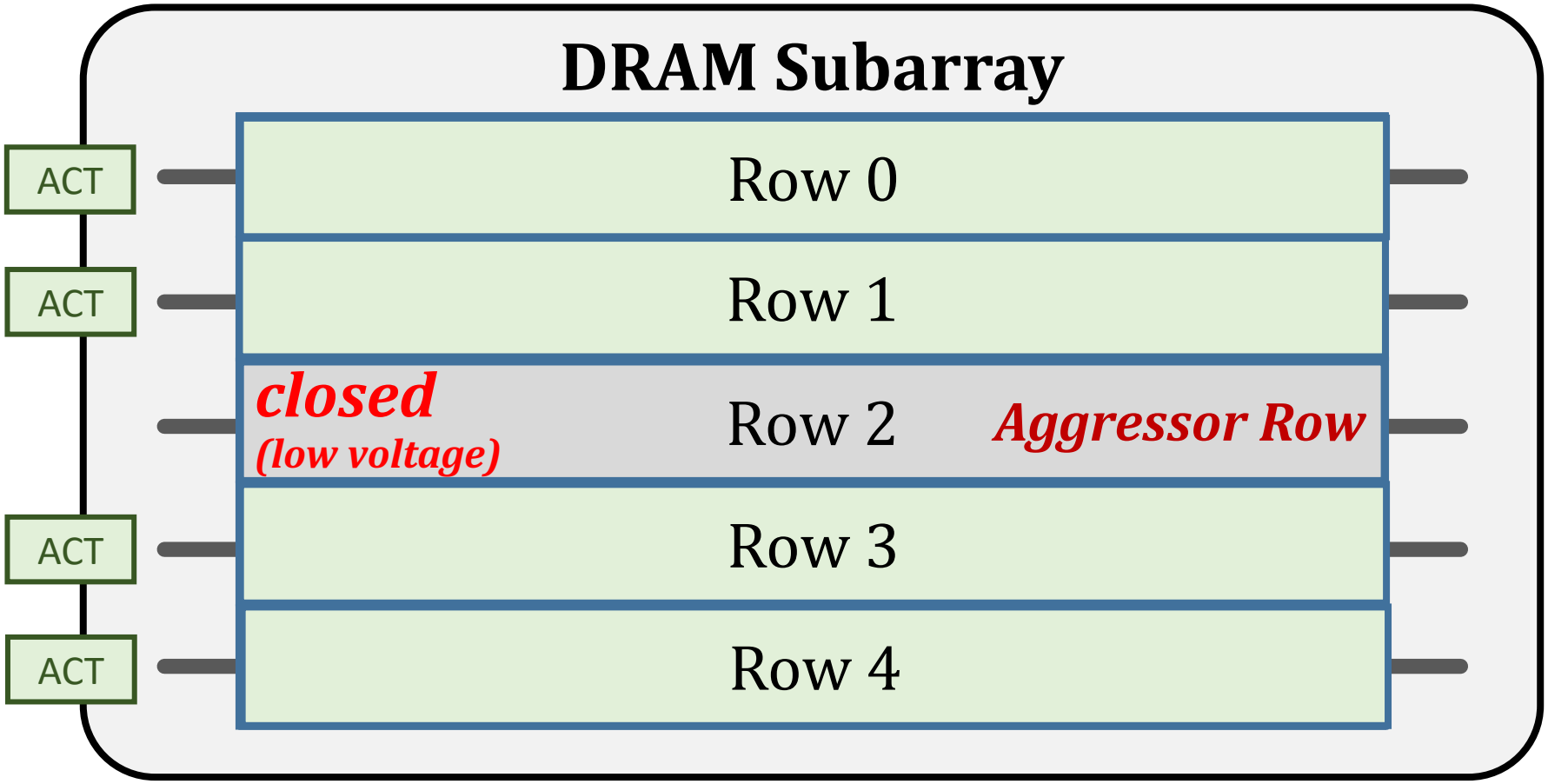
ETH zürich CESGA TOBB ETÜ University of Economics & Technology

The RowHammer Vulnerability



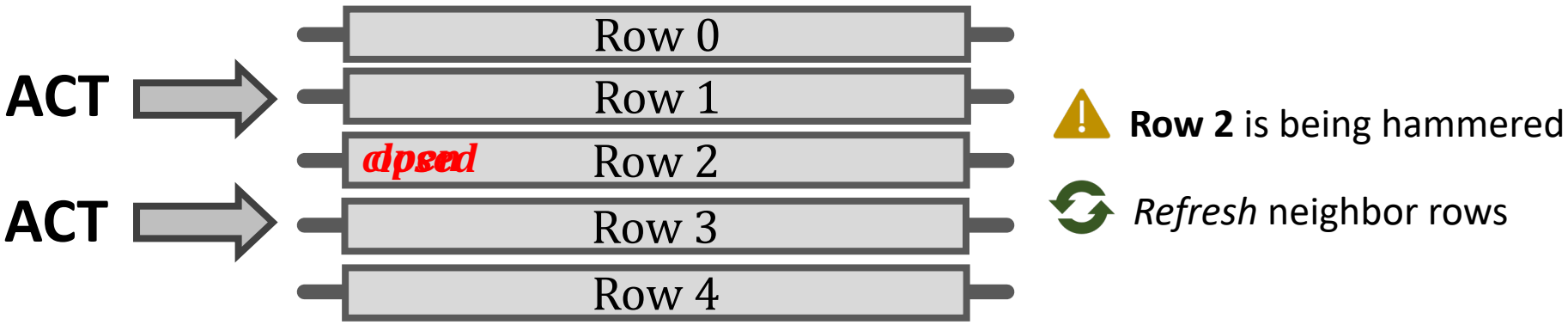
Repeatedly **opening** (activating) and **closing** (precharging) a DRAM row in **real DRAM chips** causes **RowHammer bit flips** in nearby cells

Preventive Refresh



Activating a DRAM row **refreshes the row**
and **prevents RowHammer bit flips**

Mitigating RowHammer

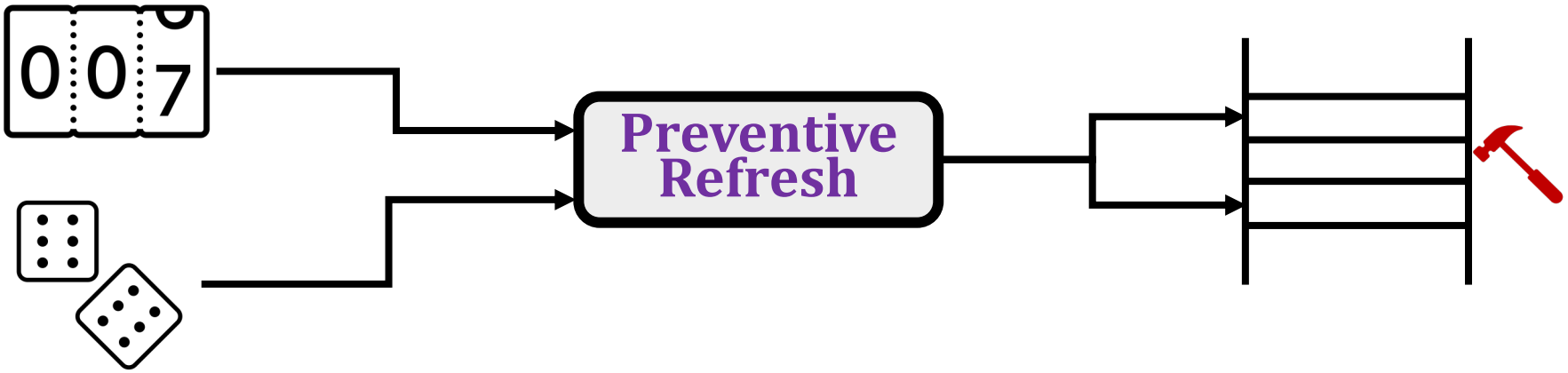


Preventive Refresh

Activating potential victim rows **mitigate RowHammer** by refreshing them

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

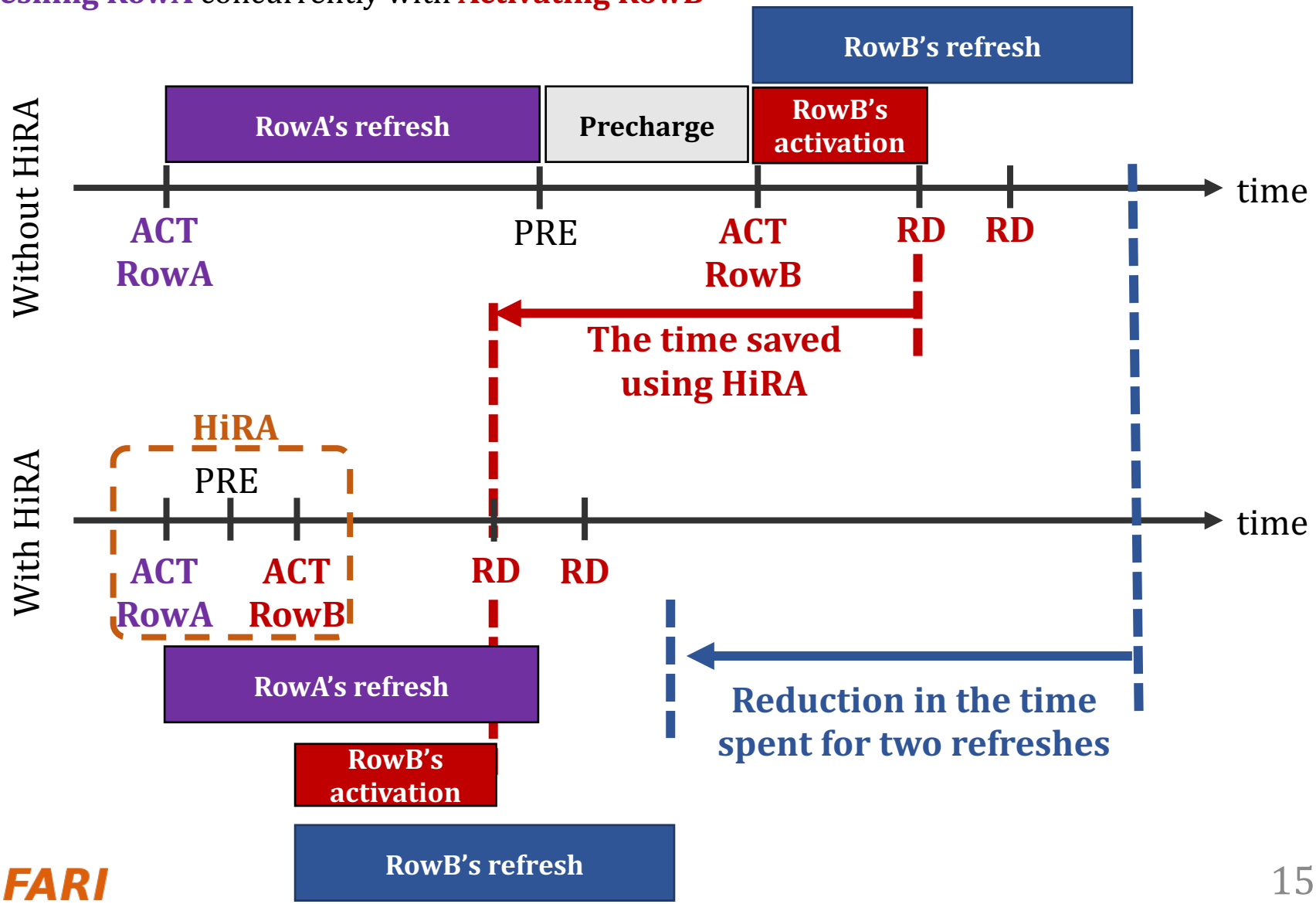
- HiRA issues **two row activation (ACT)** commands **in quick succession**
  - **Challenge 2:** DRAM chips **ignore the second** activation before precharge
  - **Solution 2 :** HiRA issues a **precharge (PRE)** command **between two ACTs**

- 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**

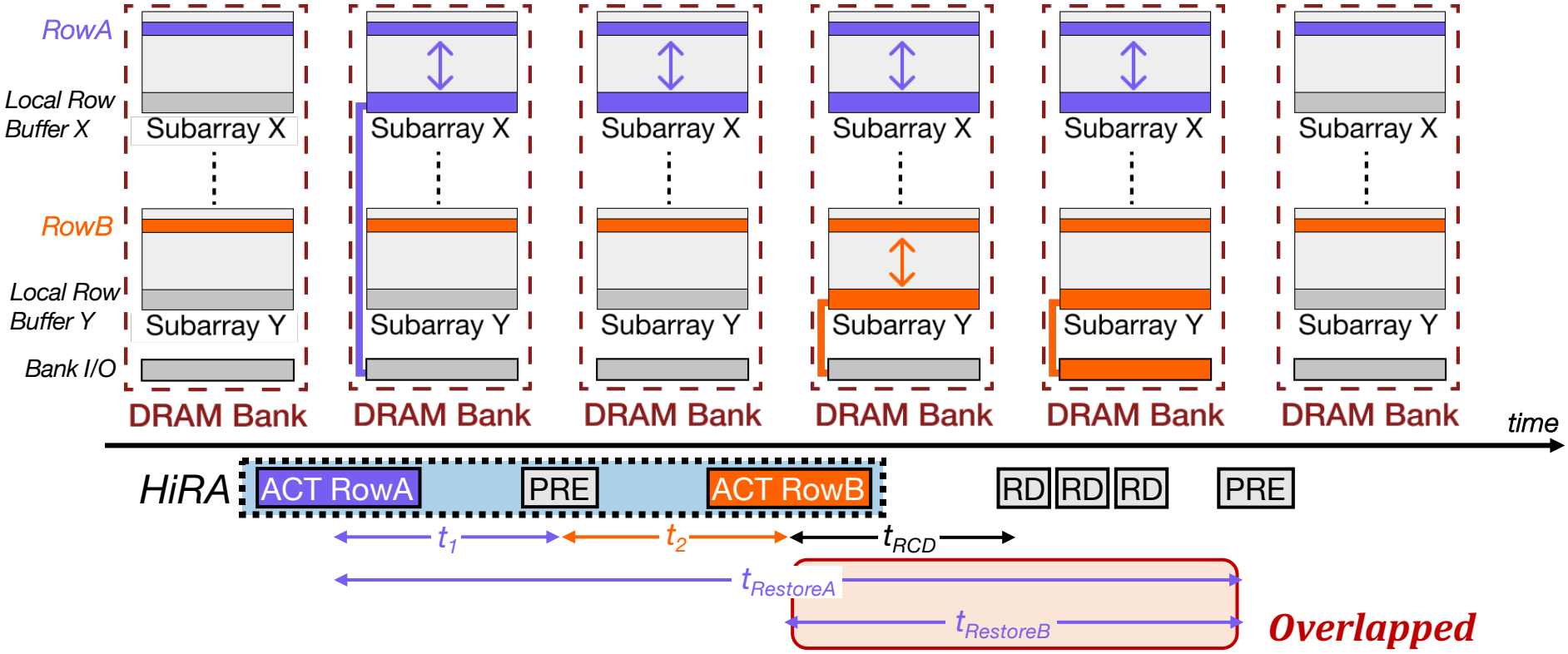
HiRA **violates DRAM timing constraints**
by issuing a sequence of **ACT-PRE-ACT** commands
that target two rows in two **electrically disconnected subarrays**

HiRA: Hidden Row Activation

Refreshing RowA concurrently with **Activating RowB**



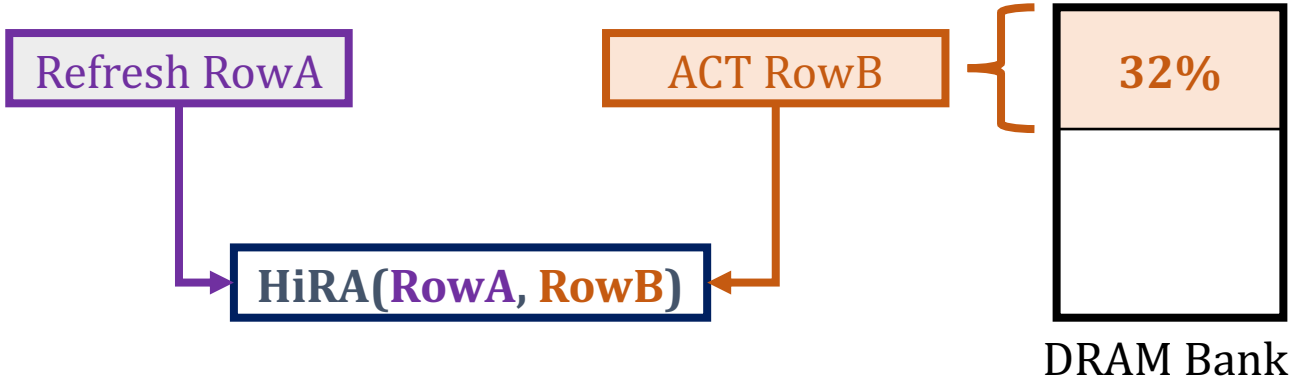
HiRA Operation



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

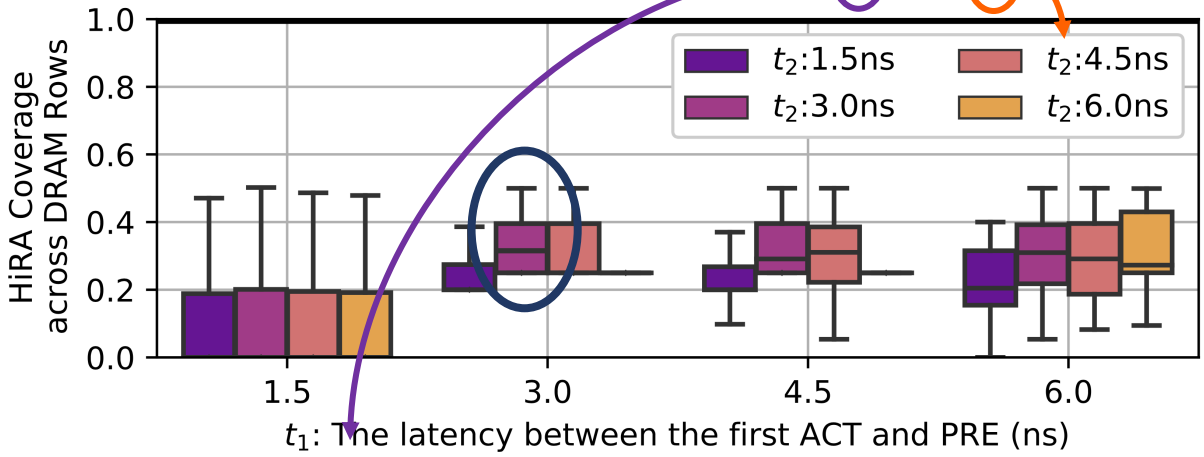
- HiRA works in **56 off-the-shelf DRAM chips** from **SK Hynix**
- **51.4% reduction** in the time spent for refresh operations
- HiRA performs a given row's **refresh** *concurrently with* **activating** any of the **32% of the rows** in the same bank

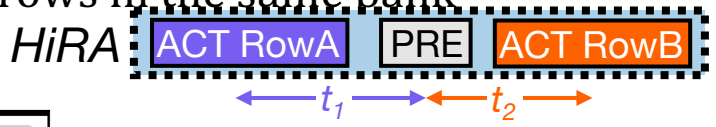


HiRA **effectively reduces the time spent** for **refresh** operations in **off-the-shelf** DRAM chips

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

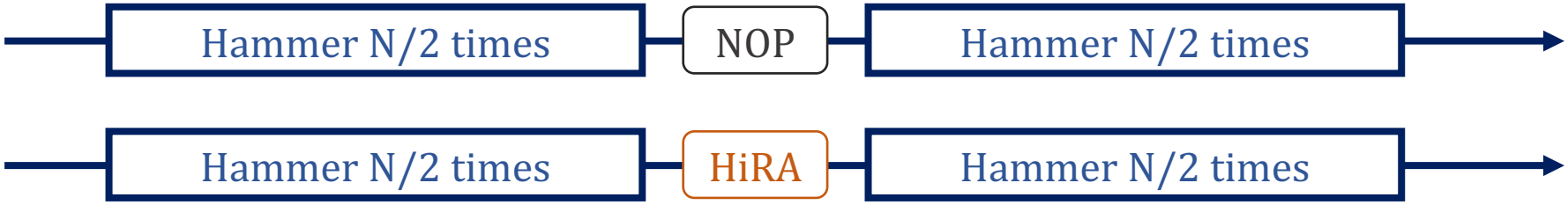




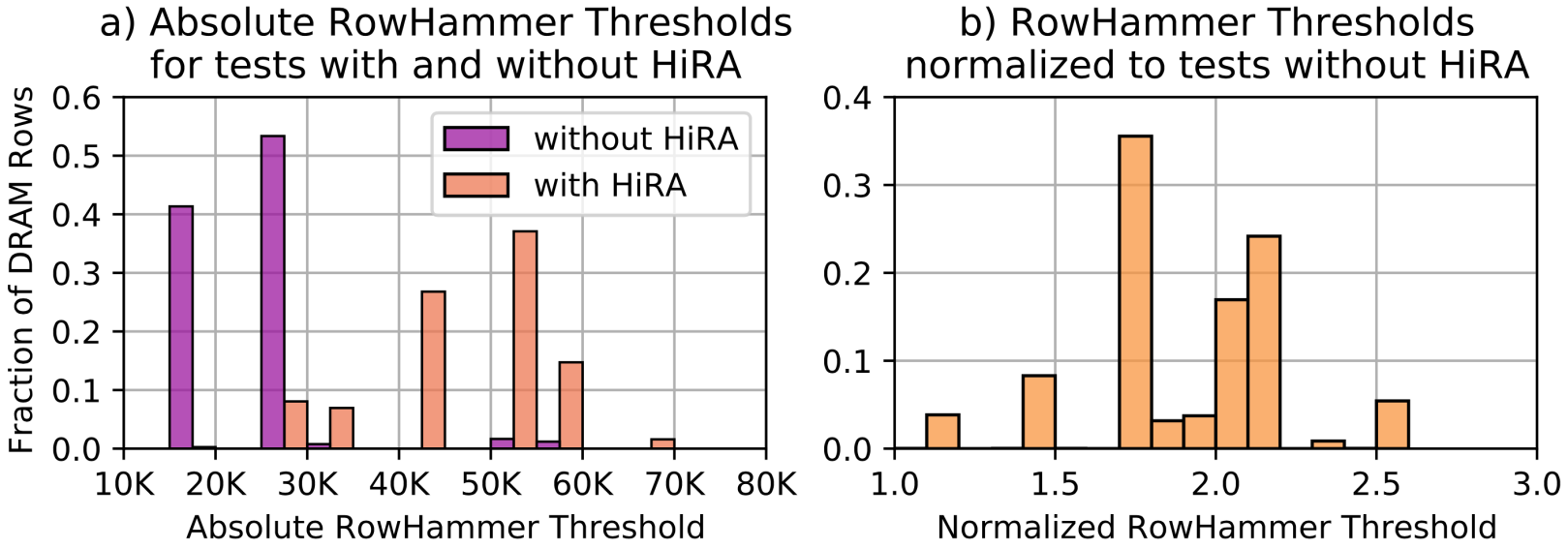
t_1 and t_2 can be *as small as **3ns***

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

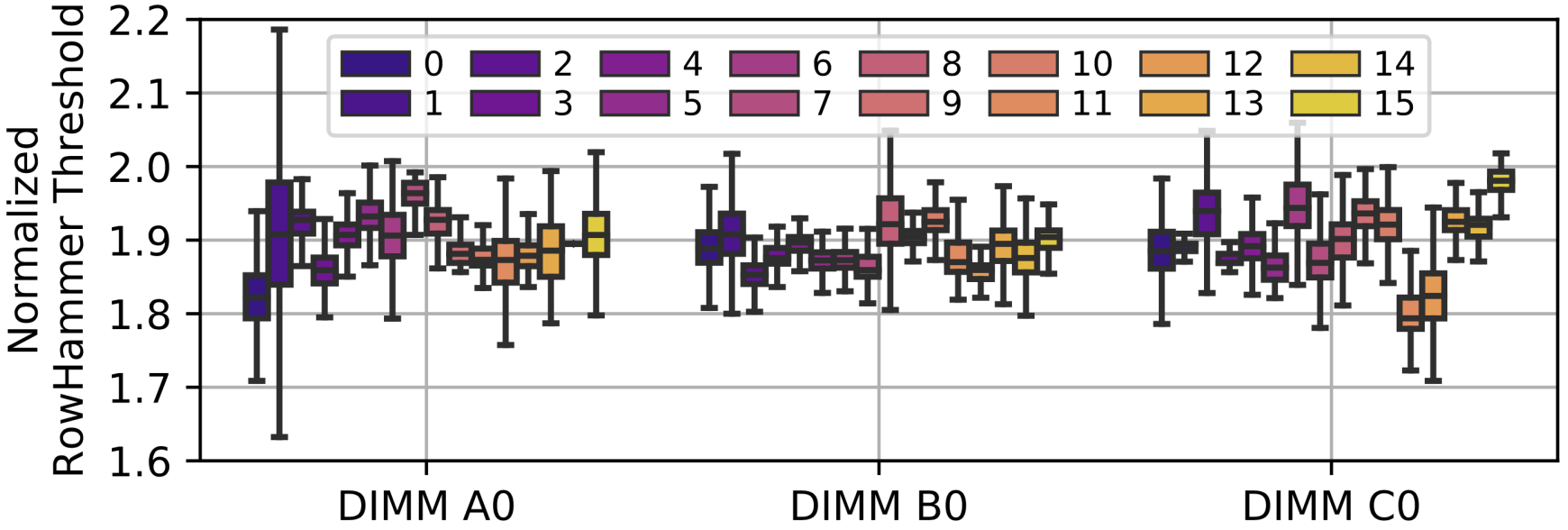


- Does performing HiRA in between refresh the victim row?
  - If HiRA's second row activation is performed, more activations are needed to induce RowHammer bit flips
  - If HiRA's second row activation is ignored, RowHammer threshold should not change



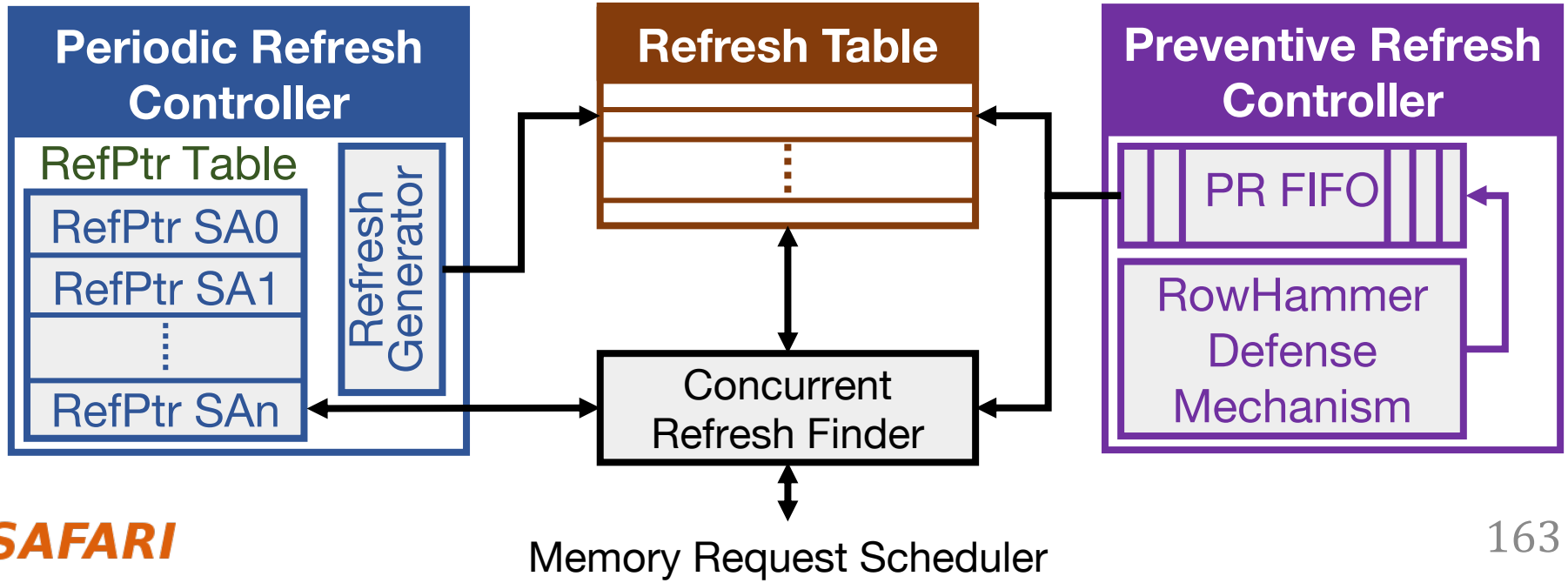
Variation across DRAM Banks

- Coverage: Identical across banks
- The effect of second row activation

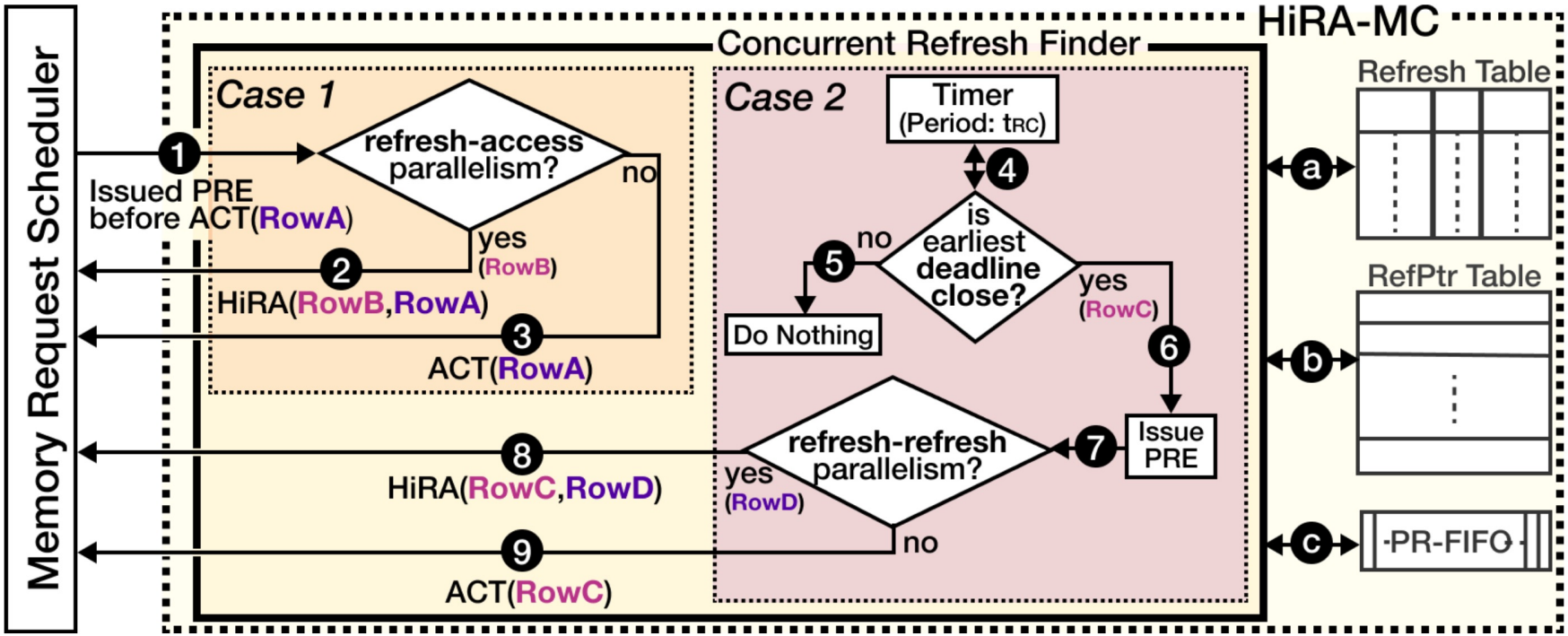


HiRA-MC: HiRA Memory Controller

- **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

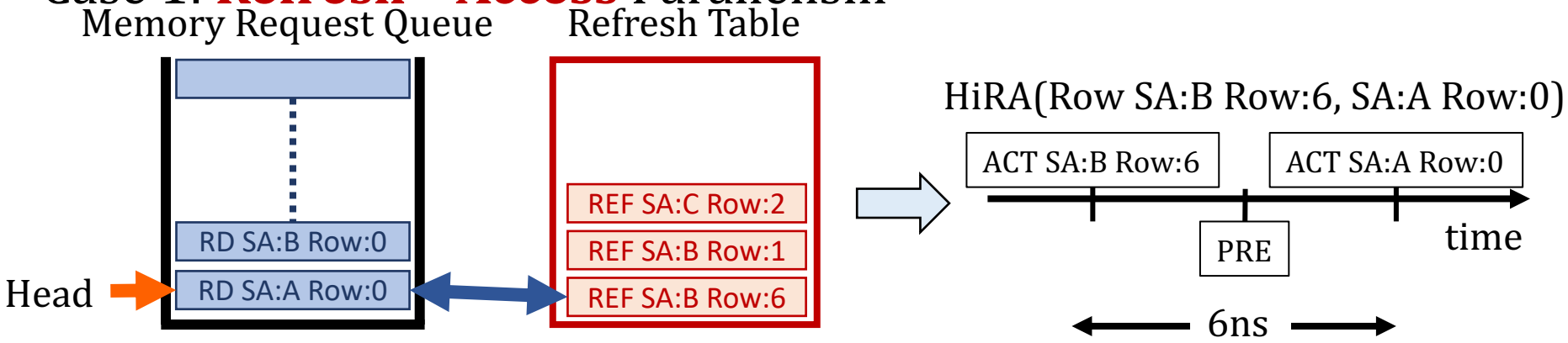


Case 1: Executes when a precharge is issued (completes before the precharge completes)

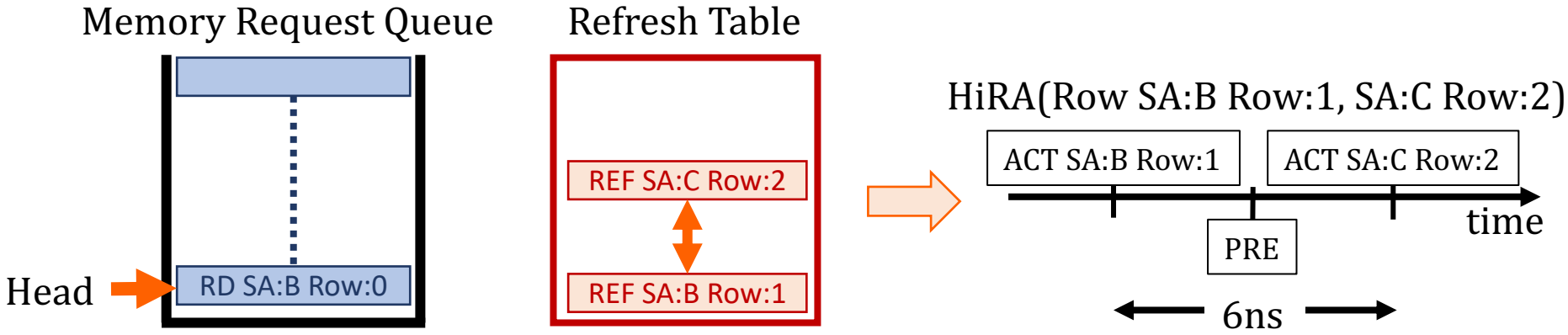
Case 2: Periodically executes after every t_{RC} (completes before t_{RC})

HiRA-MC Example

- Case 1: **Refresh – Access** Parallelism



- Case 2: **Refresh – Refresh** Parallelism



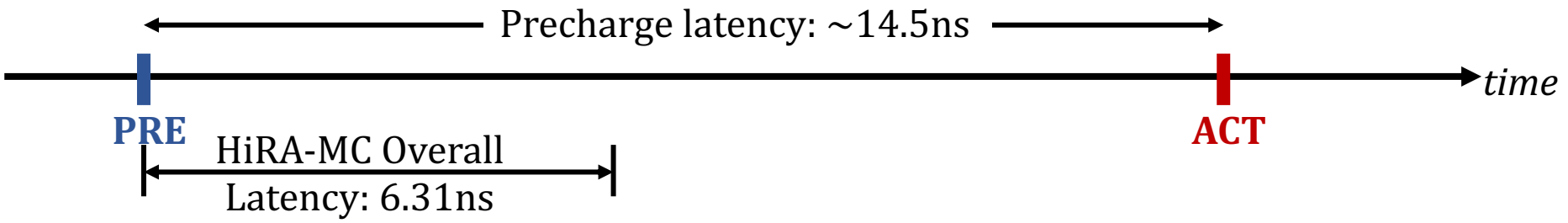
HiRA-MC provides **refresh-access** and **refresh-refresh** parallelism

HiRA-MC Hardware Complexity

- We use CACTI with 22nm technology node

HiRA-MC Component	Area (mm^2)	Area (% of Chip Area)	Access Latency
Refresh Table	0.00031	<0.0001%	0.07ns
RefPtr Table	0.00683	0.0017%	0.12ns
PR-FIFO	0.00029	<0.0001%	0.07ns
Subarray Pairs Table	0.00180	0.0005%	0.09ns
Overall	**0.00923**	**0.0023%**	**6.31ns**

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



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

DRAM Chip Capacity

2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture

Nonblocking Memory Refresh

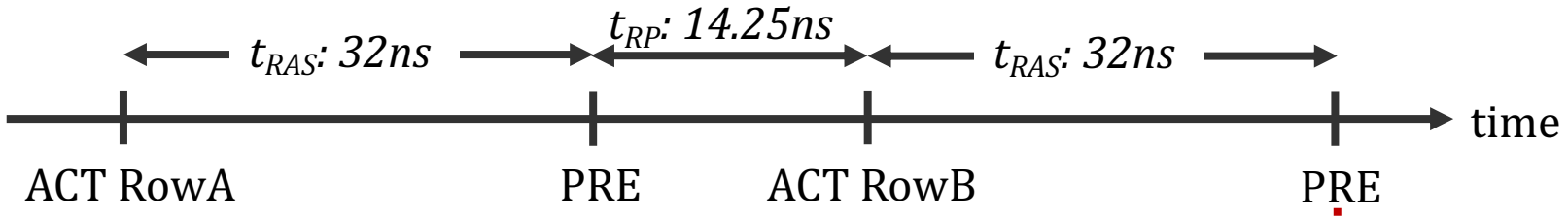
Kate Nguyen, Kehan Lyu, Xianze Meng
Department of Computer Science
Virginia Tech
Blacksburg, Virginia
katevy@vt.edu, kehan@vt.edu, xianze@vt.edu

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

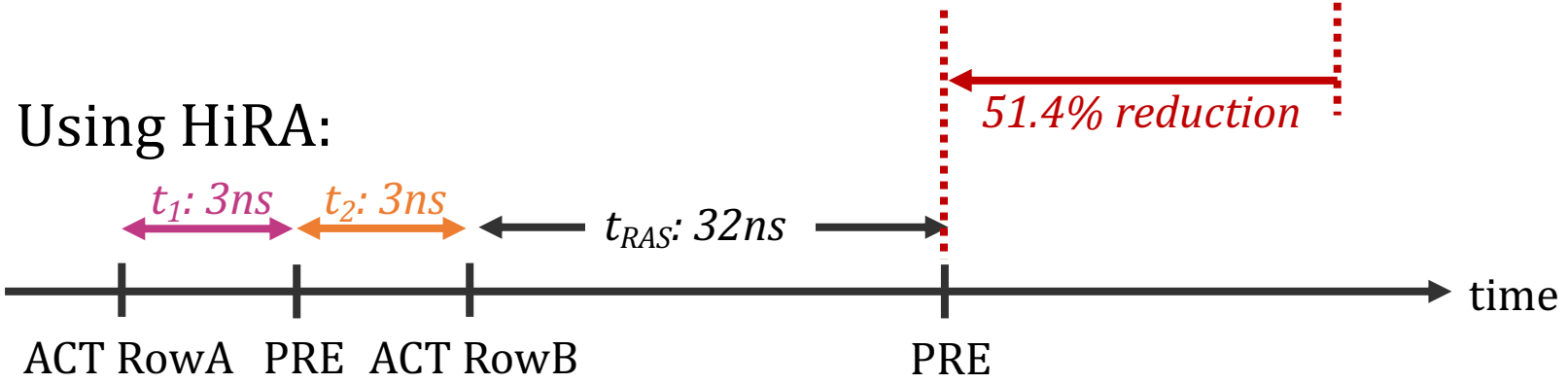
Xun Jian
Department of Computer Science
Virginia Tech
Blacksburg, Virginia
xunj@vt.edu

Reducing Overall Latency of Two Refreshes

- Refreshing two rows using nominal timing parameters:



- Using HiRA:



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.

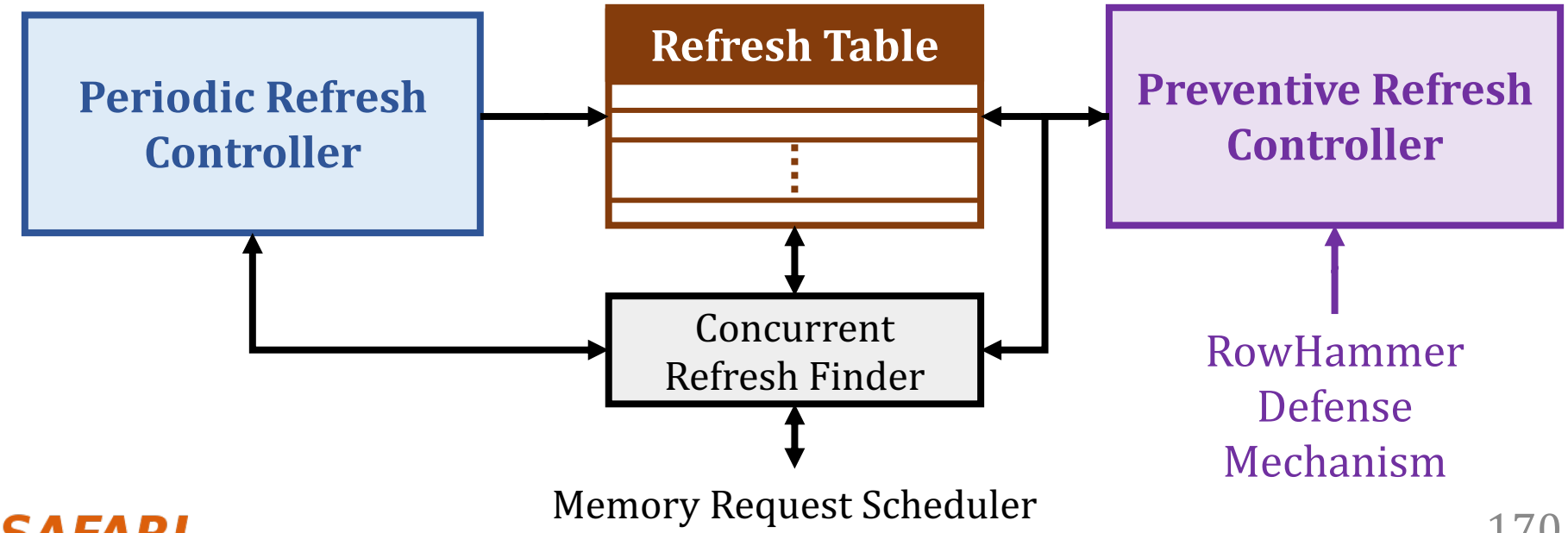
Module Label	Module Vendor	Module Identifier / Chip Identifier	Freq (MT/s)	Date Code	Chip Cap.	Die Rev.	Chip Org.	HiRA Coverage Min.	HiRA Coverage Avg.	HiRA Coverage Max.	Norm. N_{RH} Min.	Norm. N_{RH} Avg.	Norm. N_{RH} Max.
A0	G.SKILL	DWCW (Partial Marking)*	2400	42-20	4Gb	B	x8	24.8%	25.0%	25.5%	1.75	1.90	2.52
A1		F4-2400C17S-8GNT [39]						24.9%	26.6%	28.3%	1.72	1.94	2.55
B0	Kingston	H5AN8G8NDJR-XNC	2400	48-20	4Gb	D	x8	25.1%	32.6%	36.8%	1.71	1.89	2.34
B1		KSM32RD8/16HDR [87]						25.0%	31.6%	34.9%	1.74	1.91	2.51
C0	SK Hynix	H5ANAG8NAJR-XN	2400	51-20	4Gb	F	x8	25.3%	35.3%	39.5%	1.47	1.89	2.23
C1		HMAA4GU6AJR8N-XN [109]						29.2%	38.4%	49.9%	1.09	1.88	2.27
C2								26.5%	36.1%	42.3%	1.49	1.96	2.58

* 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.

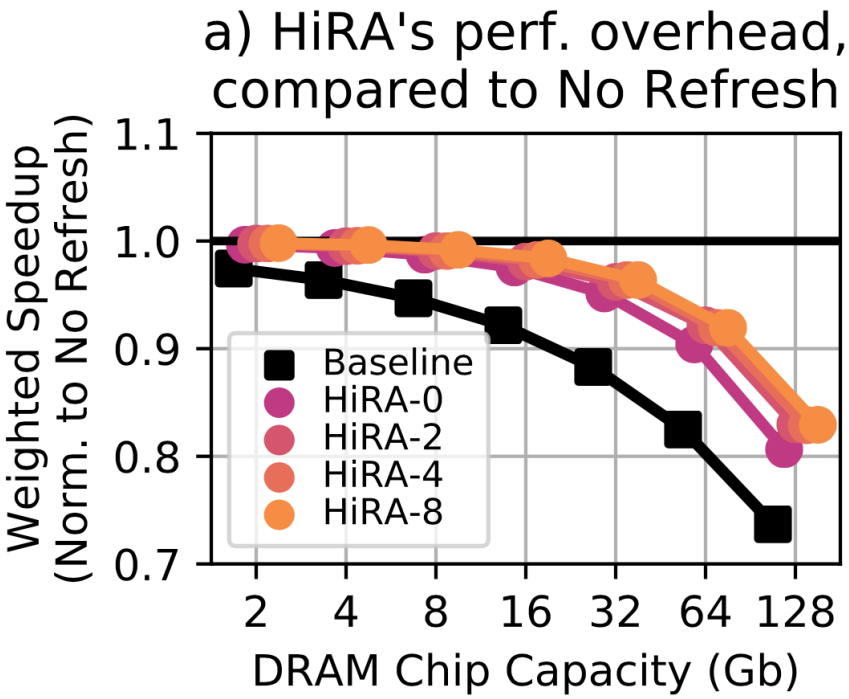
https://arxiv.org/pdf/2209.10198.pdf

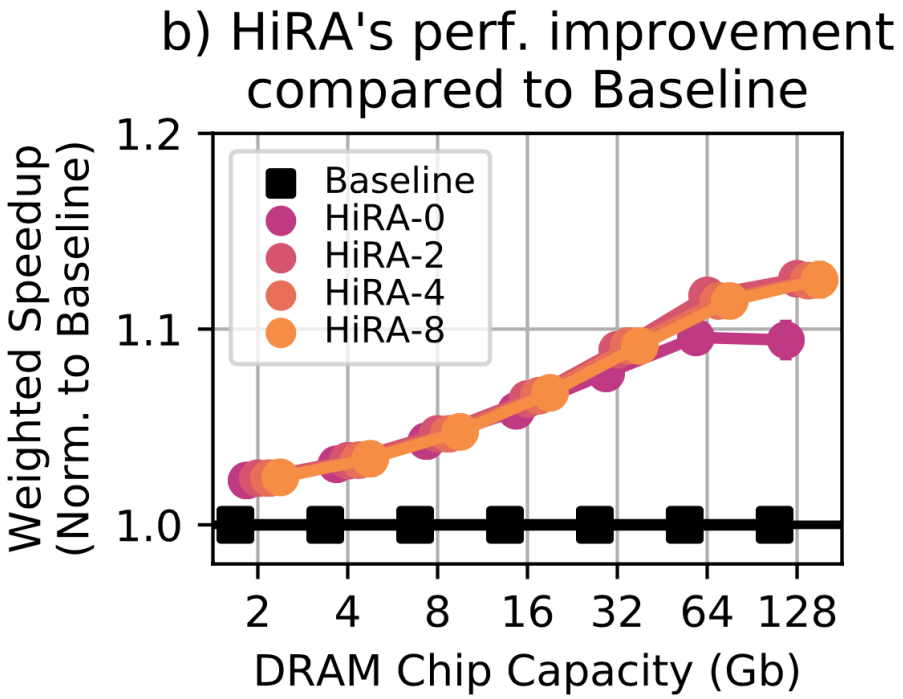
HiRA-MC: HiRA Memory Controller

- **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**

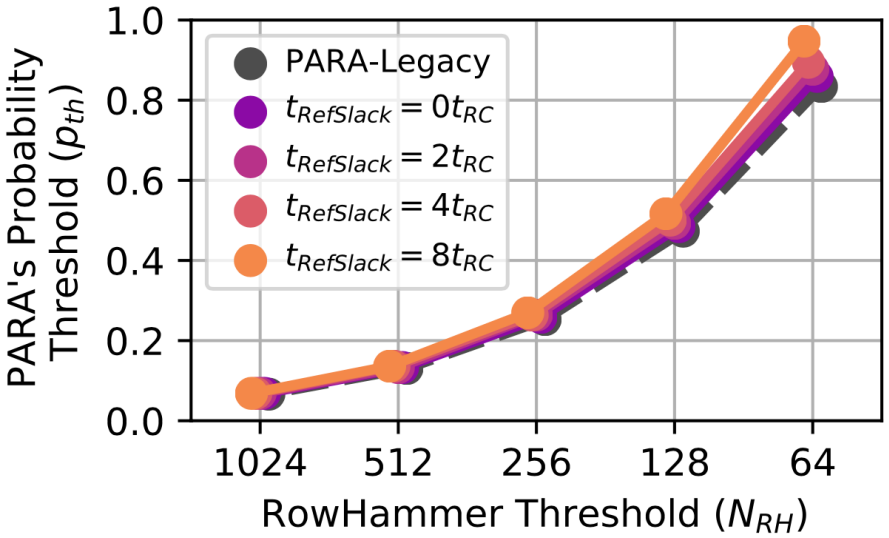


HiRA for Periodic Refreshes

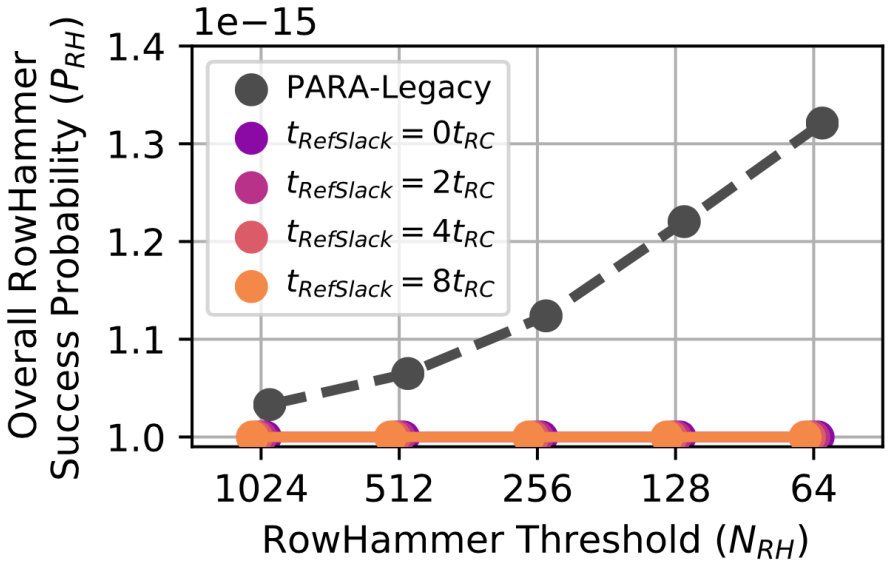




RowHammer Thresholds

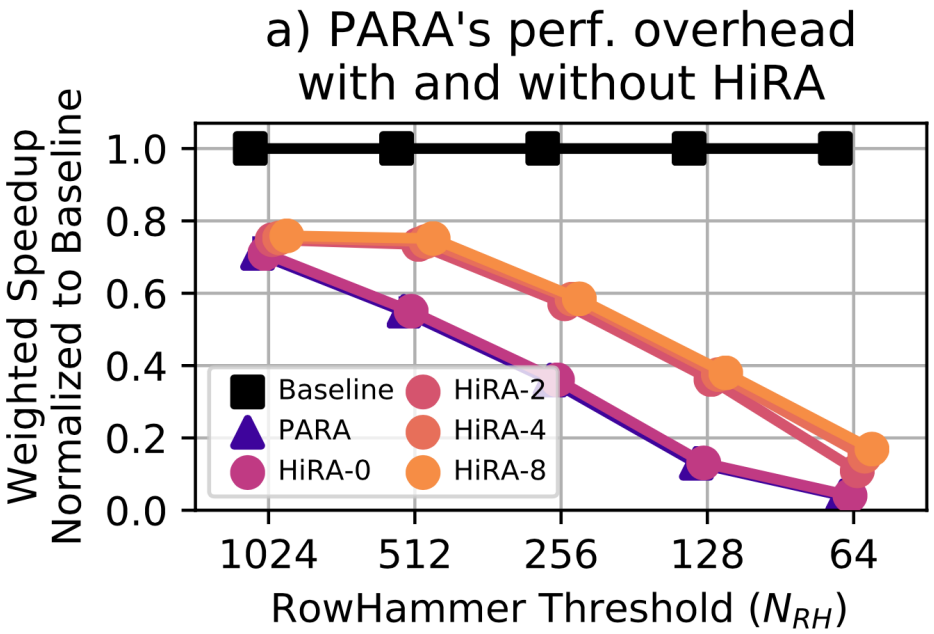


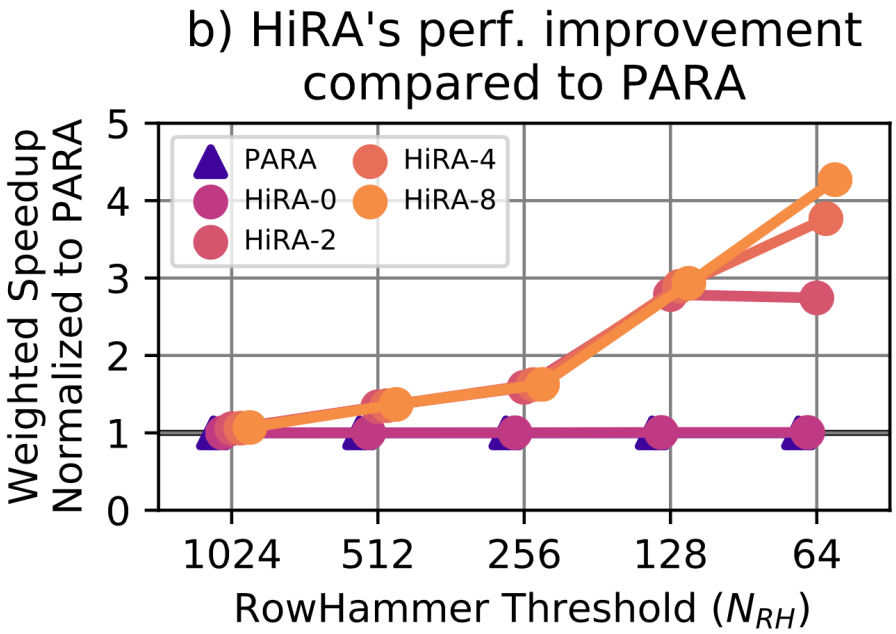
a) PARA's probability threshold (p_{th}) for different values of N_{RH} and $t_{RefSlack}$



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

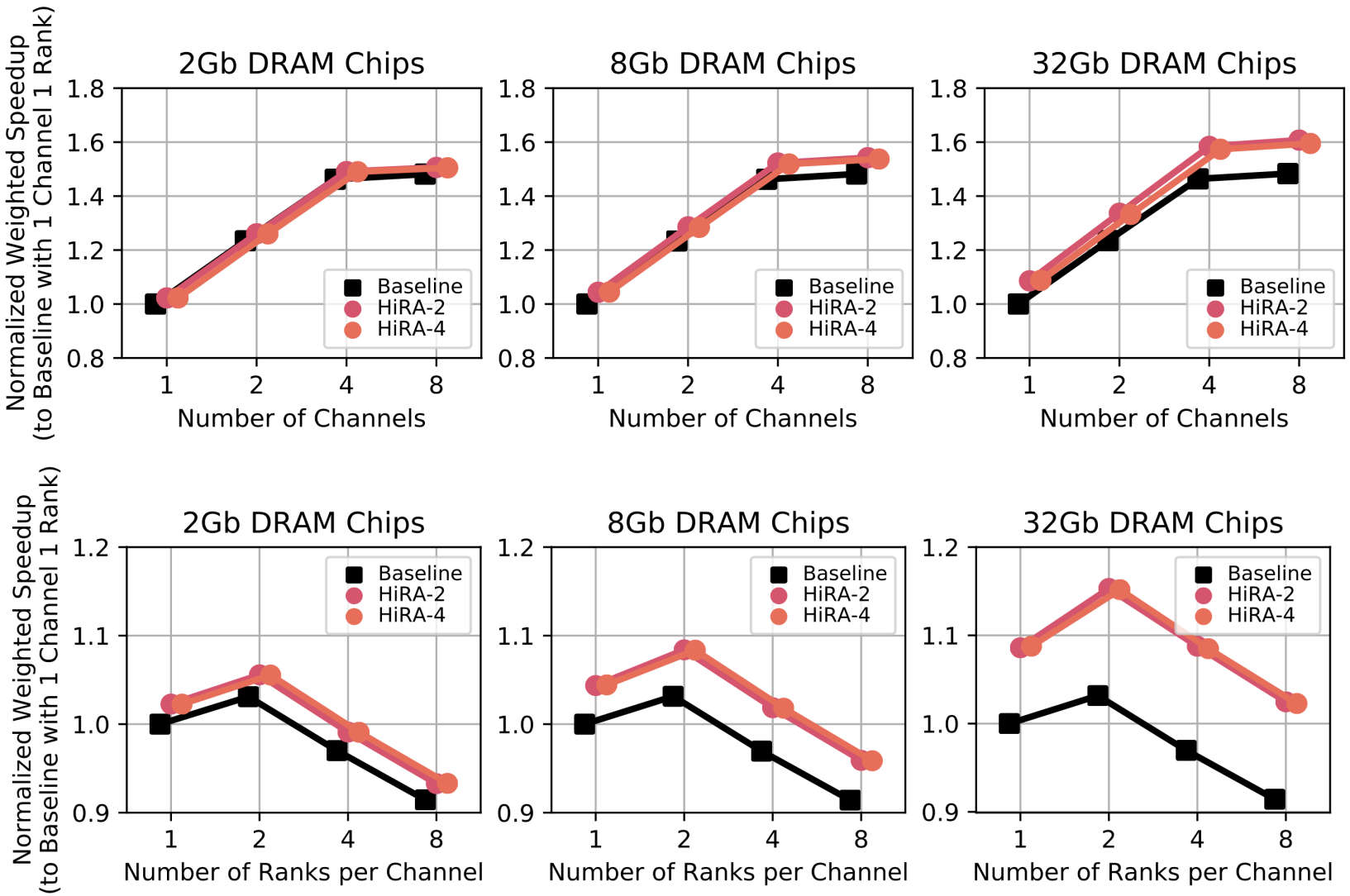
HiRA for Preventive Refreshes





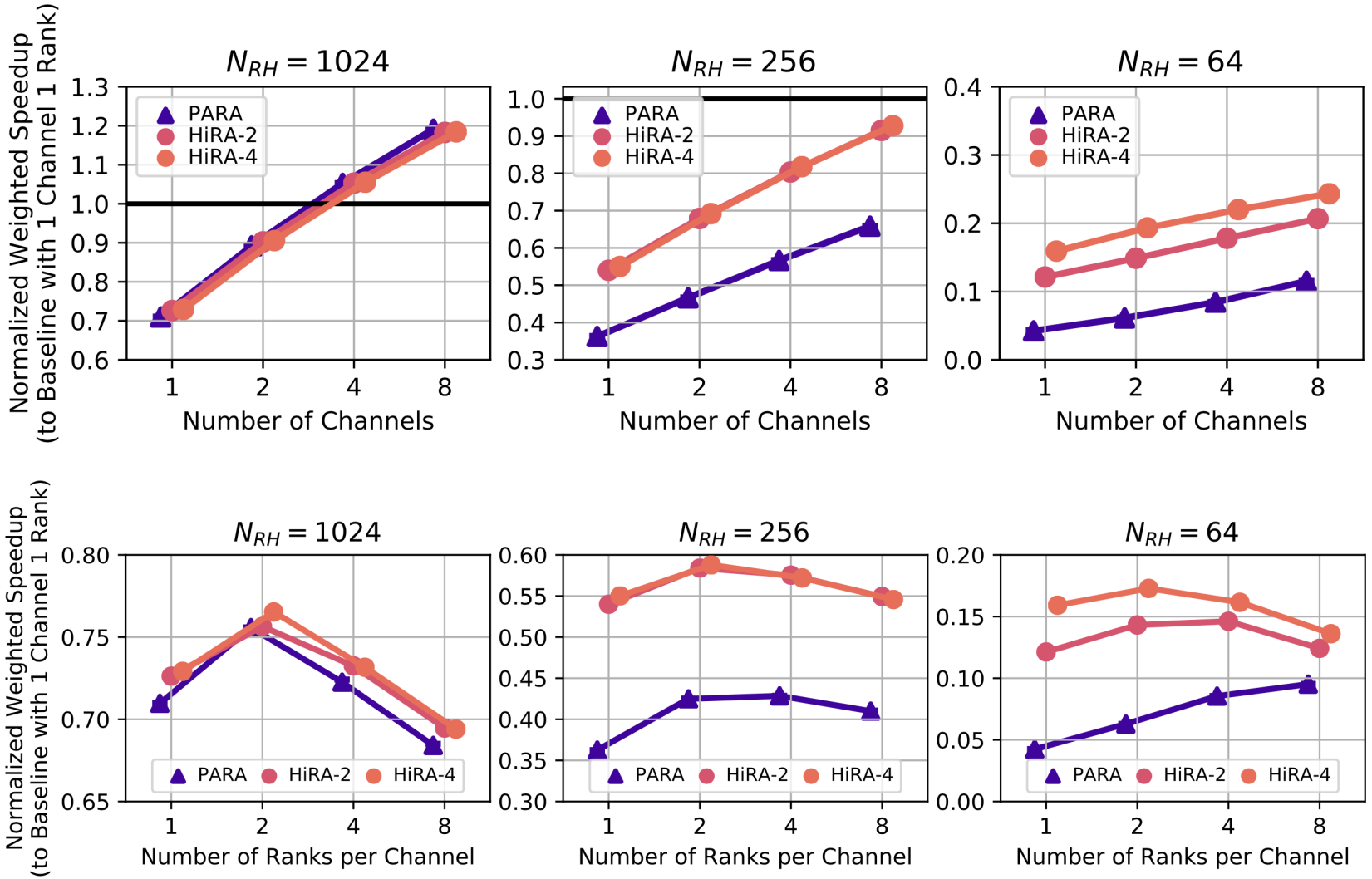
HiRA for Periodic Refresh

Sensitivity to Number of Channels and Ranks



HiRA for Preventive Refresh

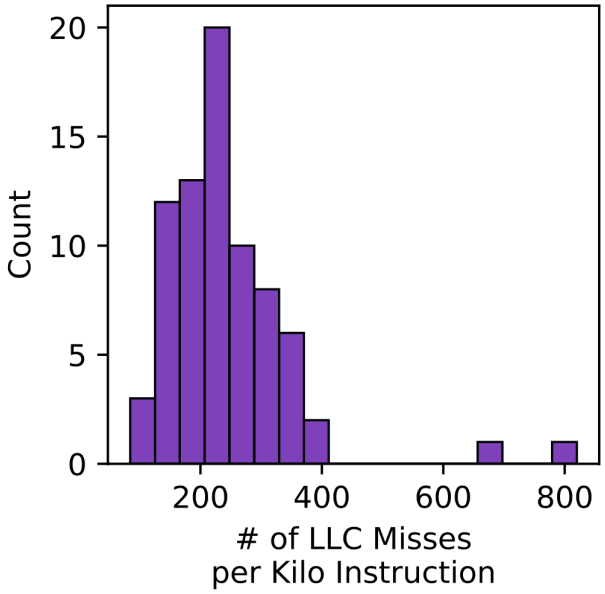
Sensitivity to Number of Channels and Ranks

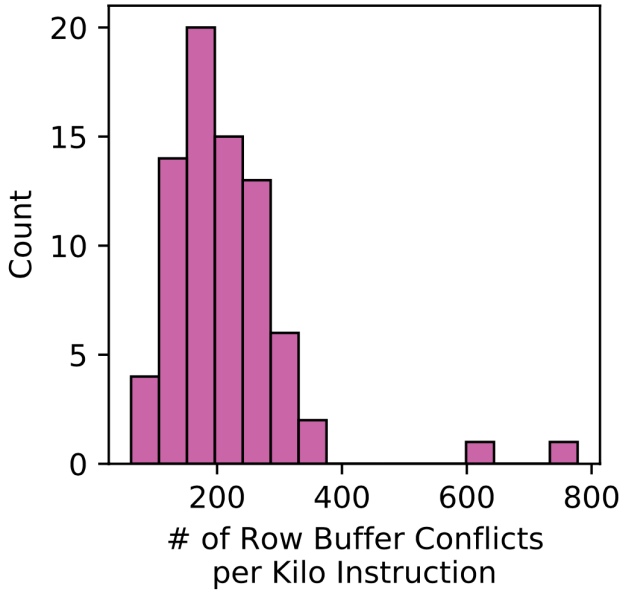


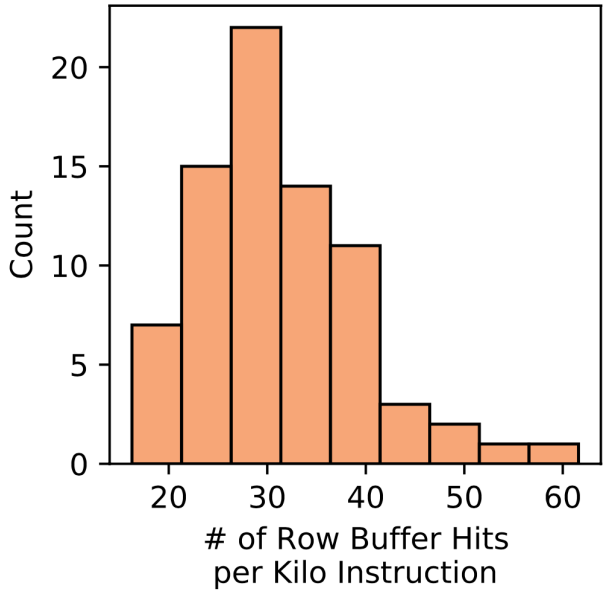
Workload Memory Access Characteristics

- 125 different 8-core multiprogrammed workloads

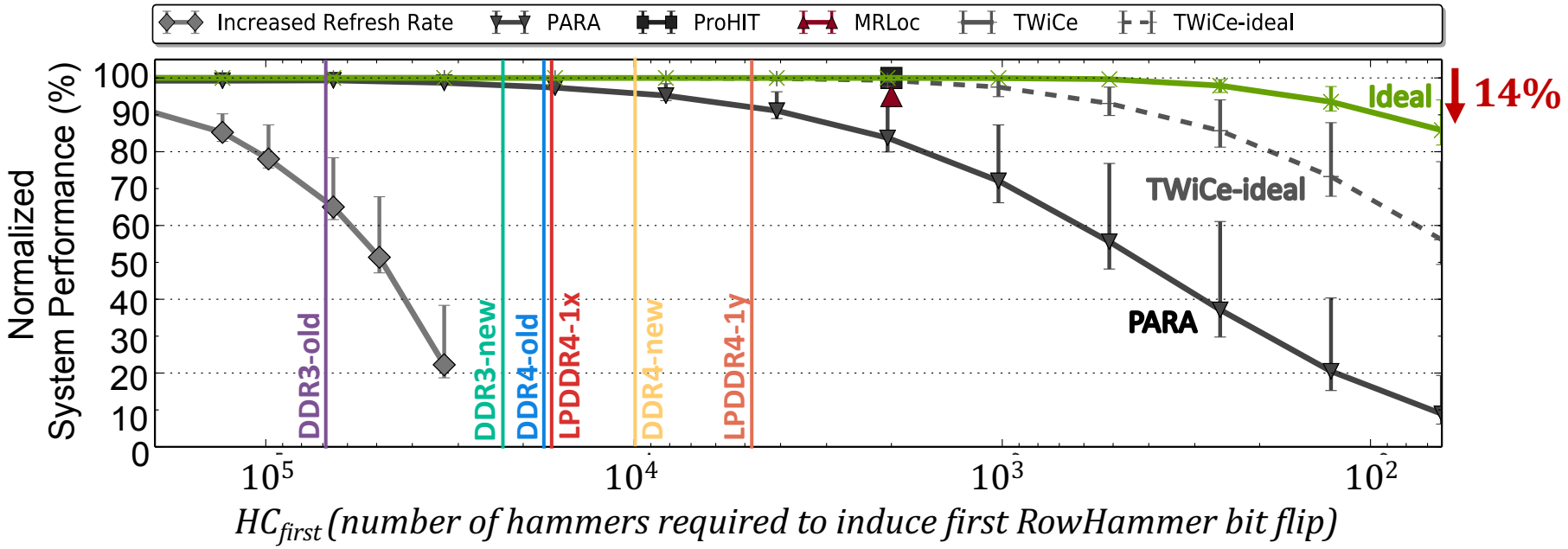
- Three histograms showing MPKI, RBCPKI, and RBHPKI respectively







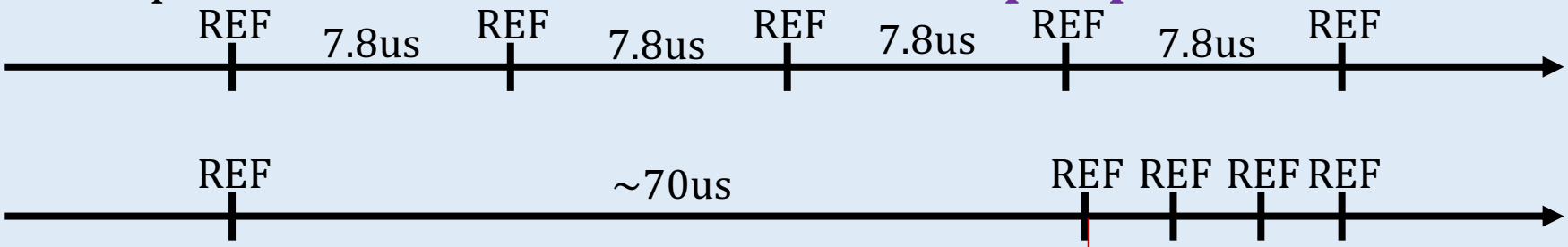
RowHammer Mitigation across Generations



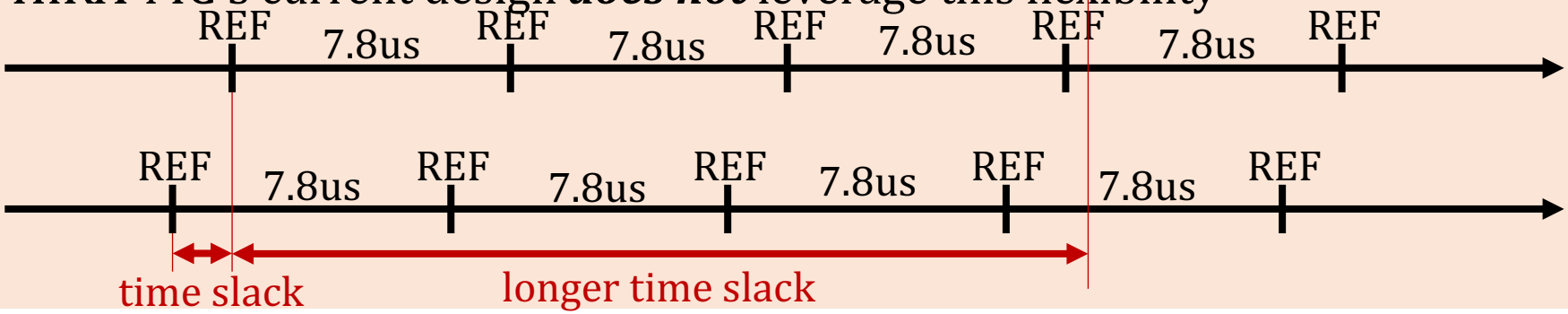
J. S. Kim, M. Patel, A. G. Yaglikci, H. Hassan, R. Azizi, L. Orosa, and O. Mutlu, "Revisiting RowHammer: An Experimental Analysis of Modern Devices and Mitigation Techniques," in *ISCA*, 2020.

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

Energy

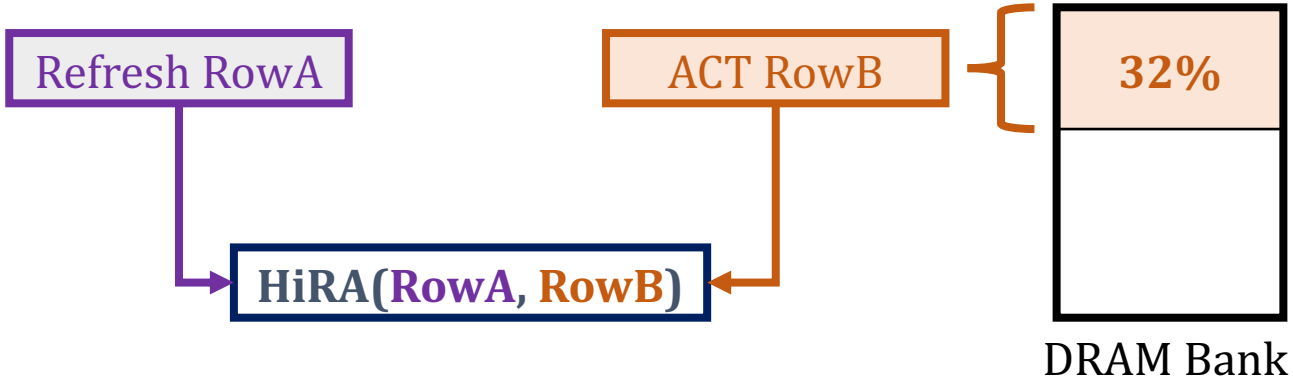
- HiRA *does not change* the **number of refresh operations** at **a given time window**
  - Overall energy consumed for refresh operations is the same

- 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 University of Economics & Technology

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

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"
  *Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (**SIGMETRICS**)*, Austin, TX, June 2014. [Slides (pptx) (pdf)] [Poster (pptx) (pdf)] [Full data sets]

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

Samira Khan†⋆
samirakhan@cmu.edu

Donghyuk Lee†
donghyuk1@cmu.edu

Yoongu Kim†
yoongukim@cmu.edu

Alaa R. Alameldeen⋆
alaa.r.alameldeen@intel.com

Chris Wilkerson⋆
chris.wilkerson@intel.com

Onur Mutlu†
onur@cmu.edu

†Carnegie Mellon University ⋆Intel Labs

Towards an Online Profiling System

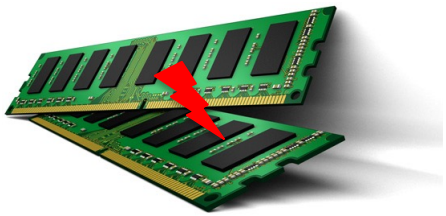
Key Observations:

- **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**

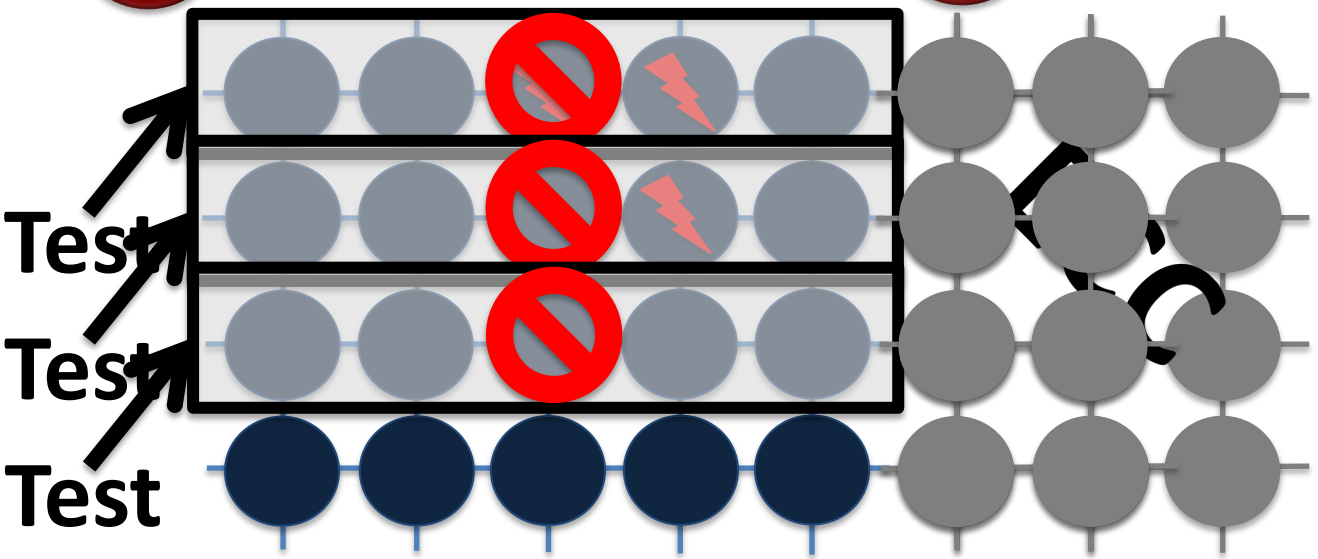
Khan+, "The Efficacy of Error Mitigation Techniques for DRAM Retention Failures: A Comparative Experimental Study," SIGMETRICS 2014.

Towards an Online Profiling System

Initially Protect DRAM with Strong ECC 1



Periodically Test Parts of DRAM 2

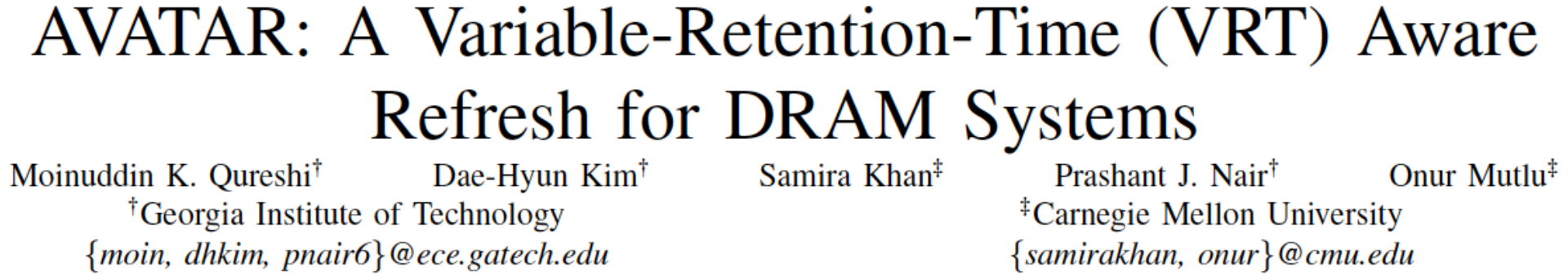


Mitigate errors and reduce ECC 3

Run tests periodically after a short interval at smaller regions of memory

Handling 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)]



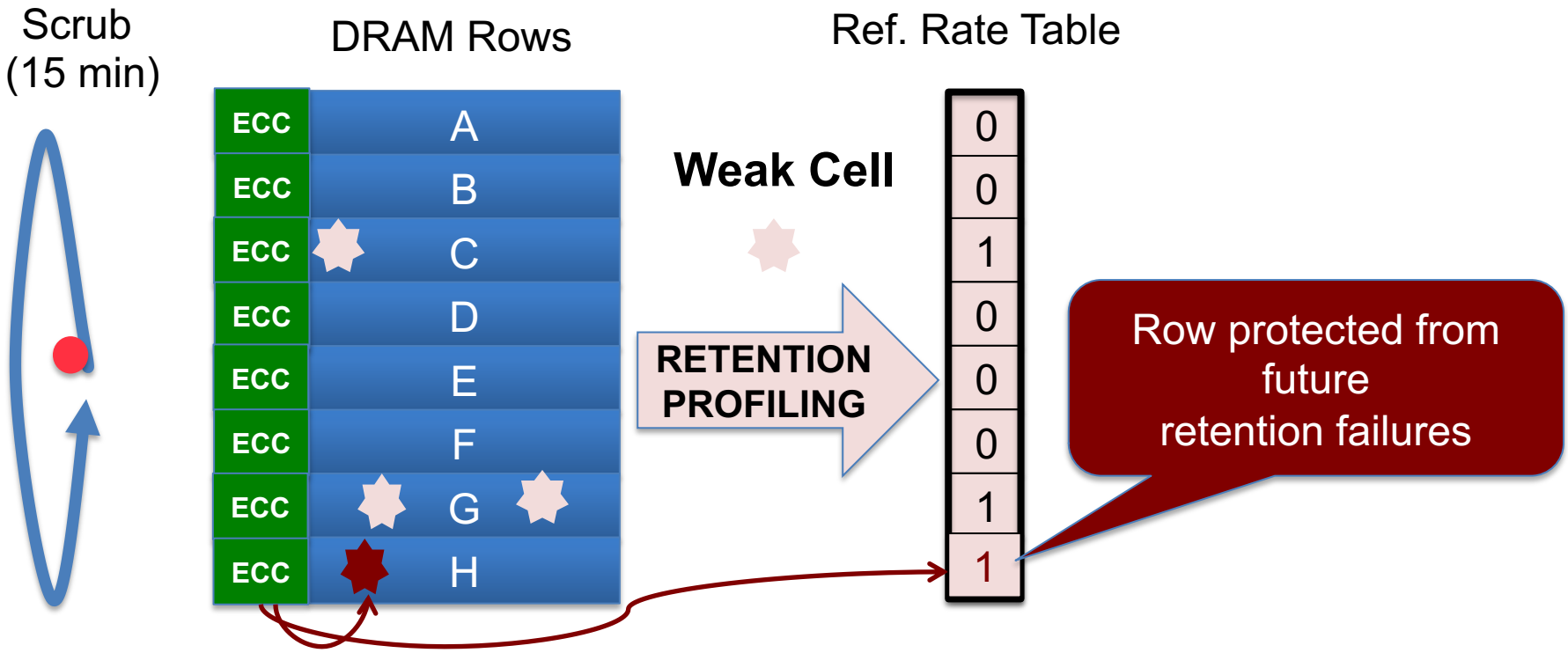
AVATAR: A Variable-Retention-Time (VRT) Aware Refresh for DRAM Systems

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

AVATAR

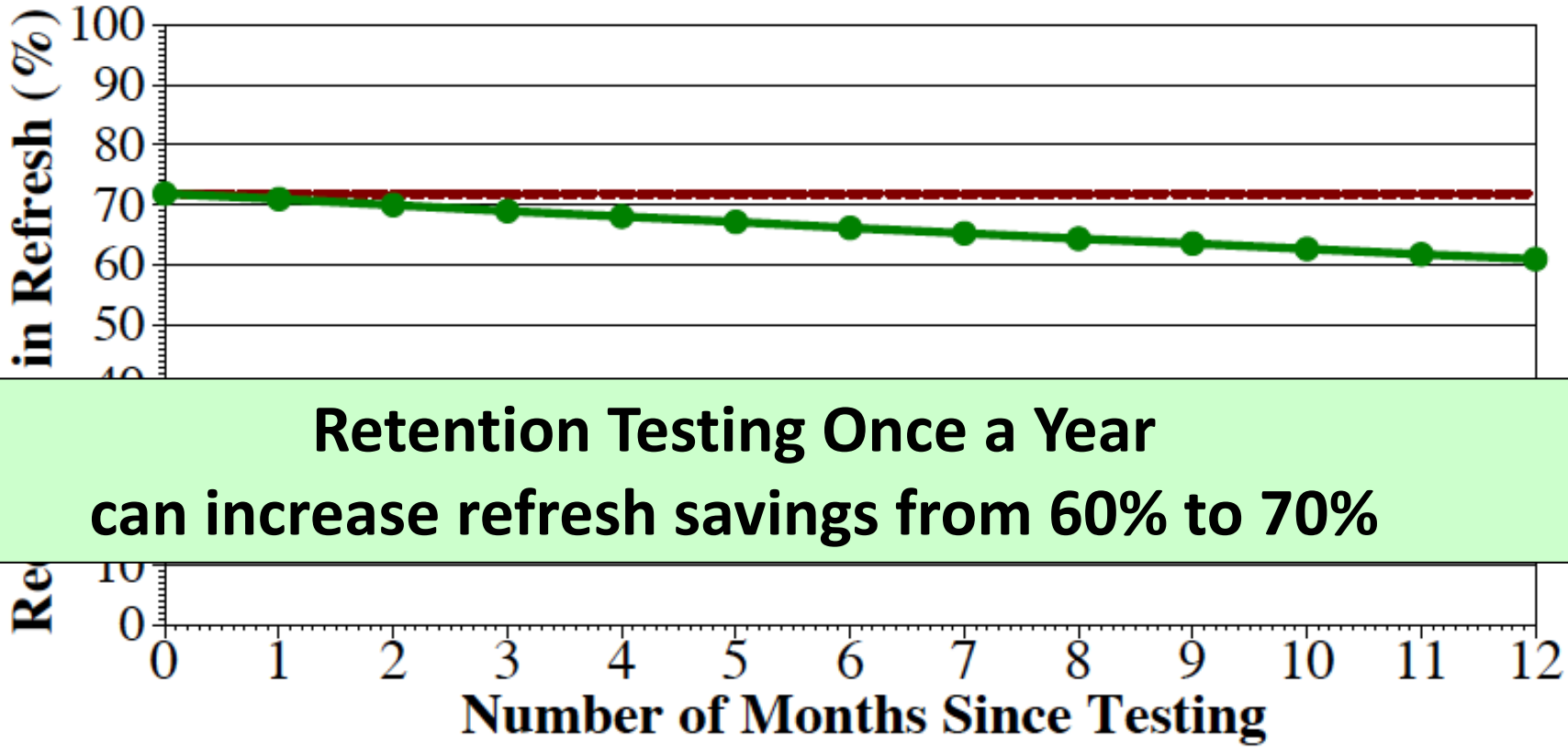
Insight: Avoid retention failures ➔ Upgrade row on ECC error

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



AVATAR mitigates VRT by increasing refresh rate on error

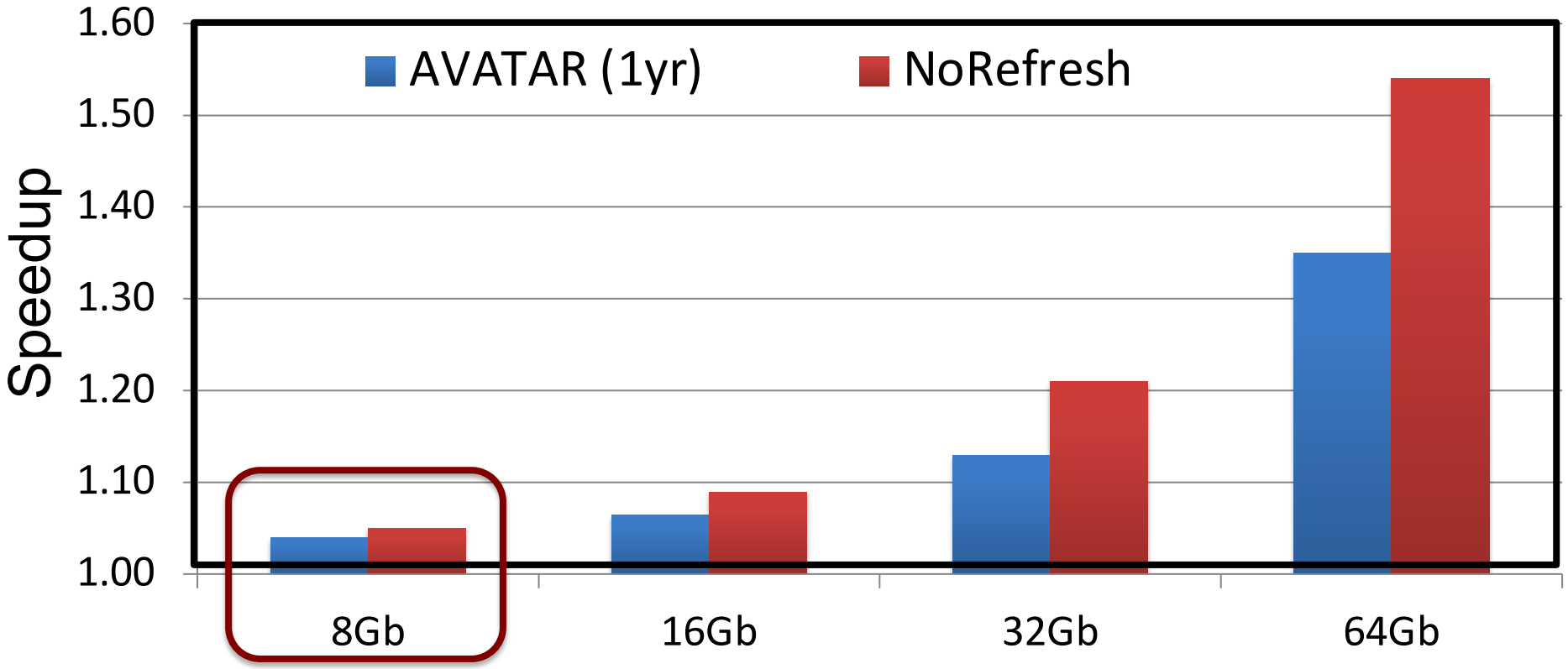
RESULTS: REFRESH SAVINGS



Retention Testing Once a Year
can increase refresh savings from 60% to 70%

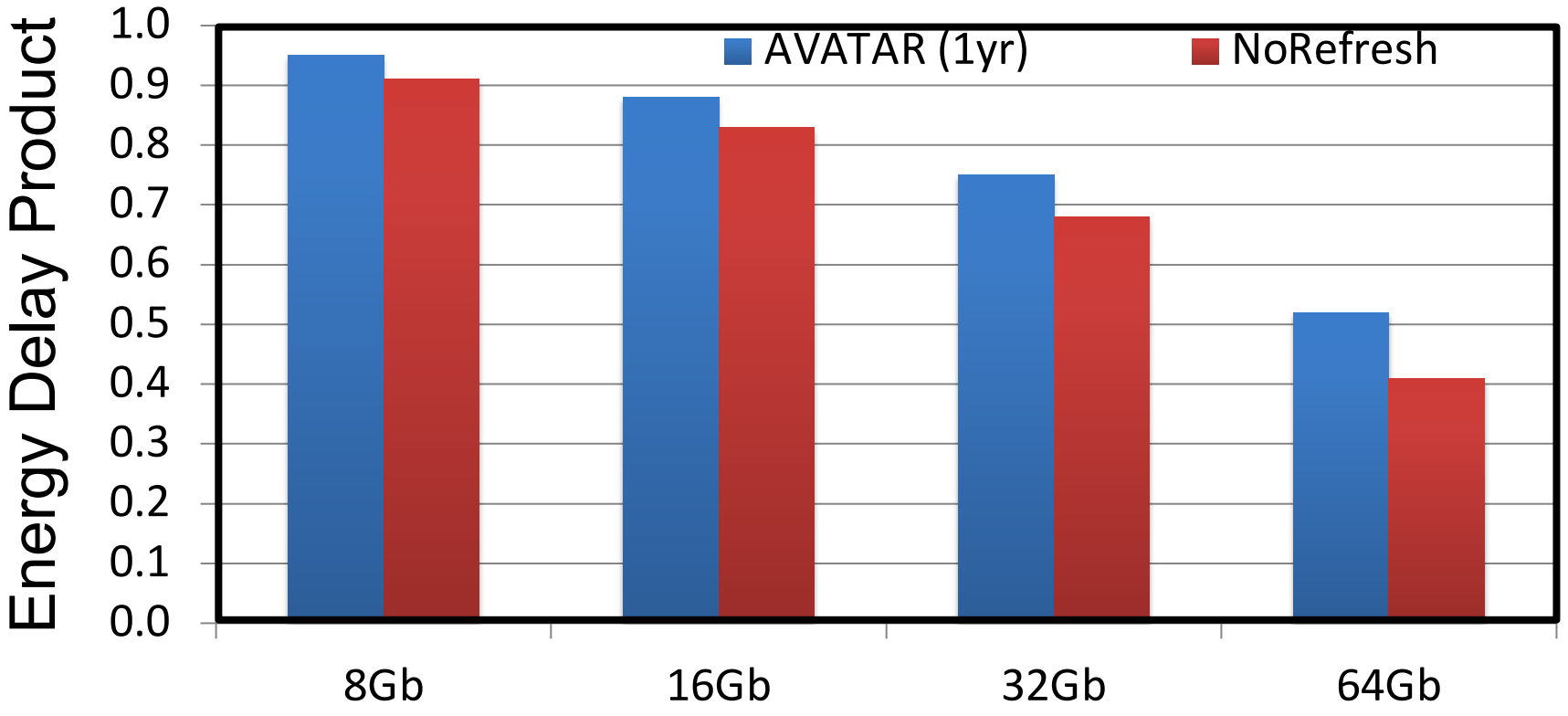
AVATAR reduces refresh by 60%-70%,
similar to multi-rate refresh but with VRT tolerance

SPEEDUP



AVATAR obtains 2/3rd the performance of NoRefresh. Higher benefits at higher capacity nodes.

ENERGY DELAY PRODUCT



AVATAR reduces EDP.
Significant reduction at higher capacity nodes.

Handling Data-Dependent Failures [DSN'16]

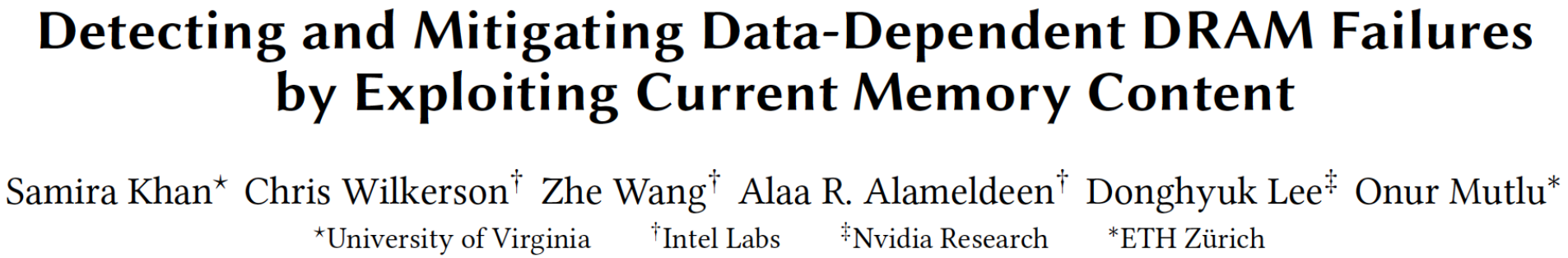
- Samira Khan, Donghyuk Lee, and Onur Mutlu,
  "PARBOR: An Efficient System-Level Technique to Detect Data-Dependent Failures in DRAM"
  *Proceedings of the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (**DSN**)*, Toulouse, France, June 2016.
  [Slides (pptx) (pdf)]

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]

- Minesh Patel, Jeremie S. Kim, and Onur Mutlu,
  "The Reach Profiler (REAPER): Enabling the Mitigation of DRAM Retention Failures via Profiling at Aggressive Conditions"
  *Proceedings of the 44th International Symposium on Computer Architecture (**ISCA**)*, Toronto, Canada, June 2017.
  [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

The Reach Profiler (REAPER):

Enabling the Mitigation of DRAM Retention Failures via Profiling at Aggressive Conditions

Minesh Patel **Jeremie S. Kim**

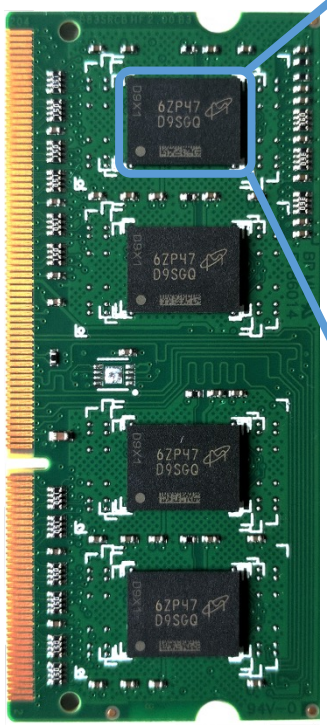
Onur Mutlu

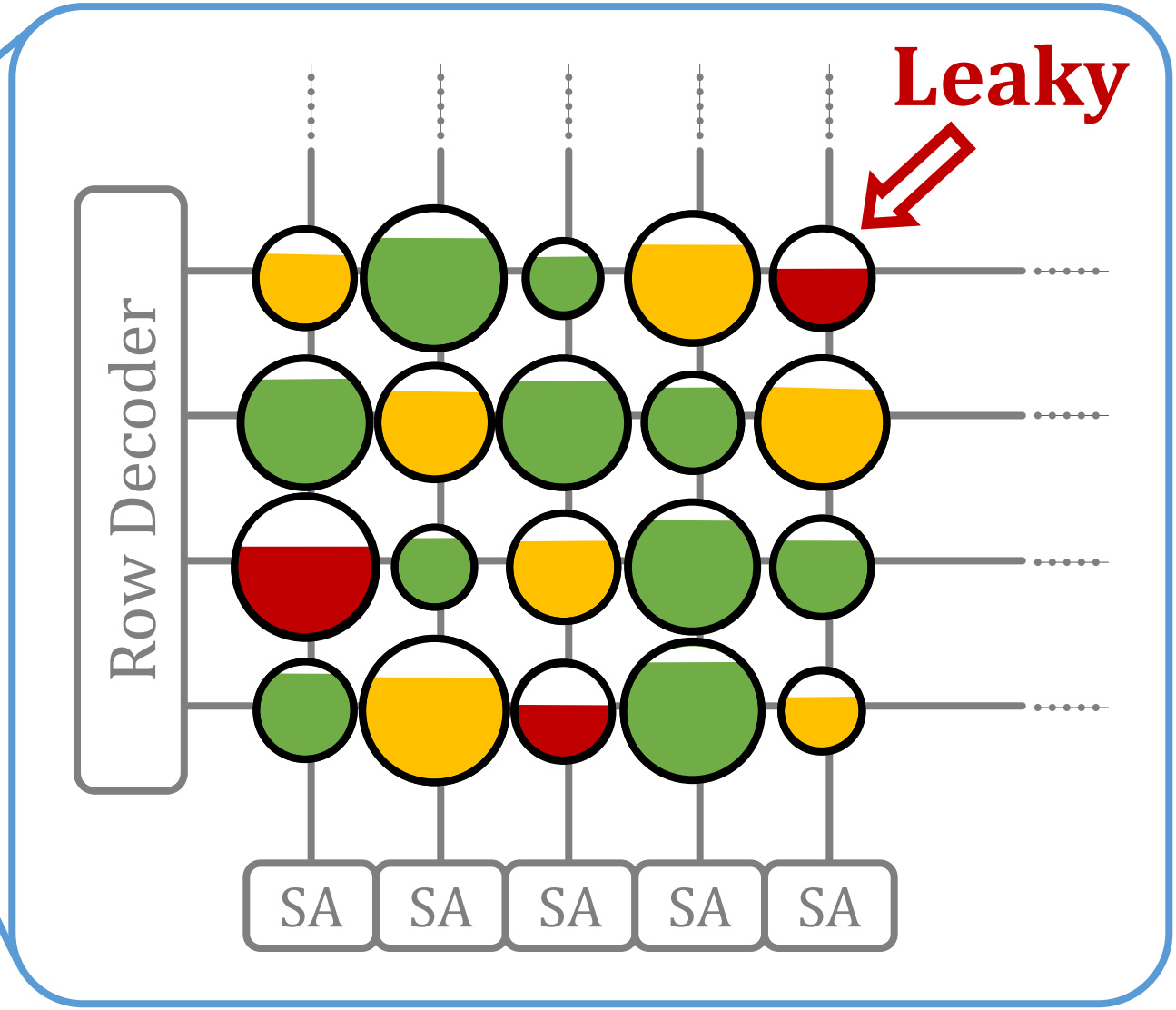
SAFARI

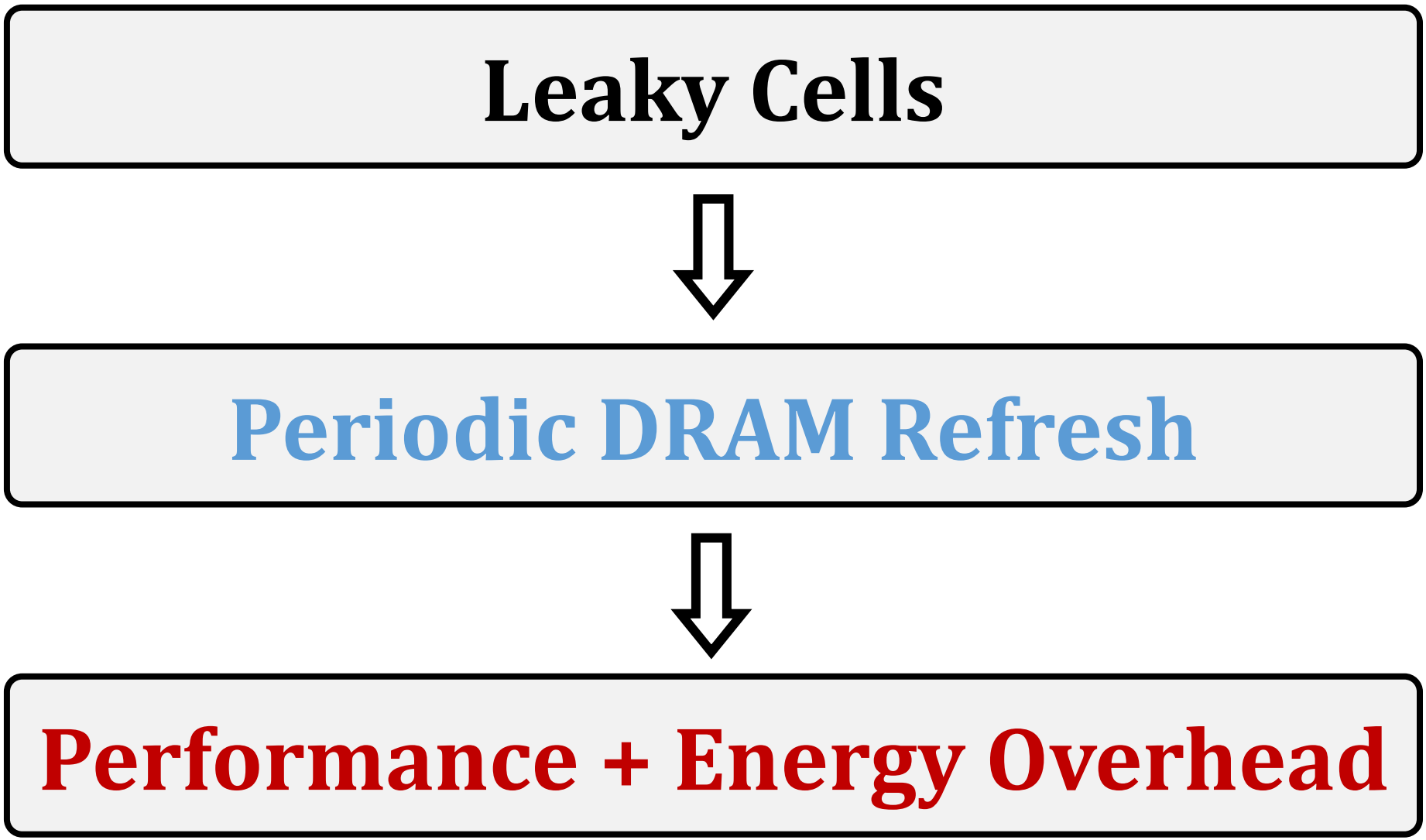
ETH Zürich

Carnegie Mellon

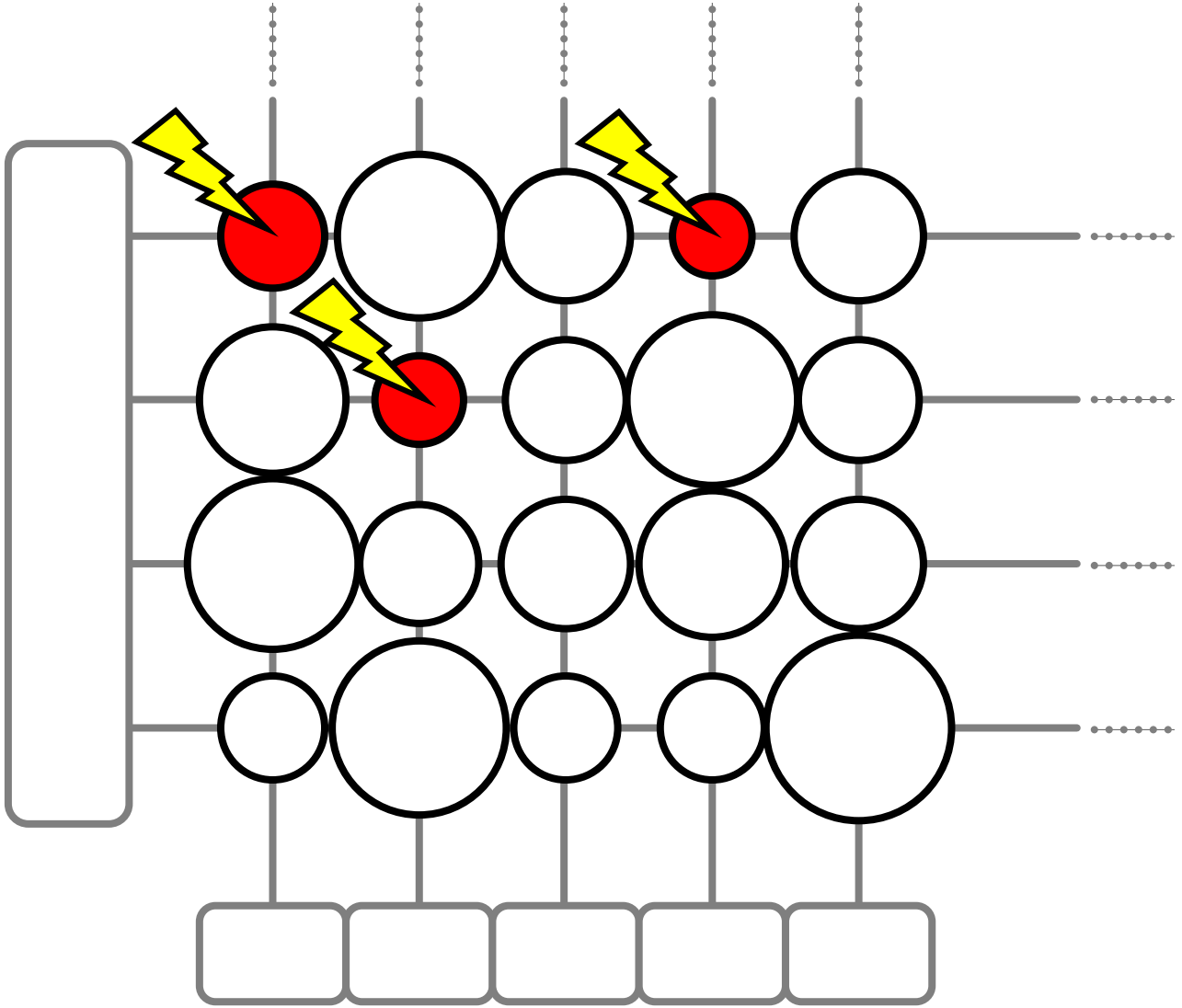
DRAM






Leaky Cells
Periodic DRAM Refresh
Performance + Energy Overhead

Goal: find *all* retention failures for a refresh interval T > default (64ms)



Process, voltage, temperature

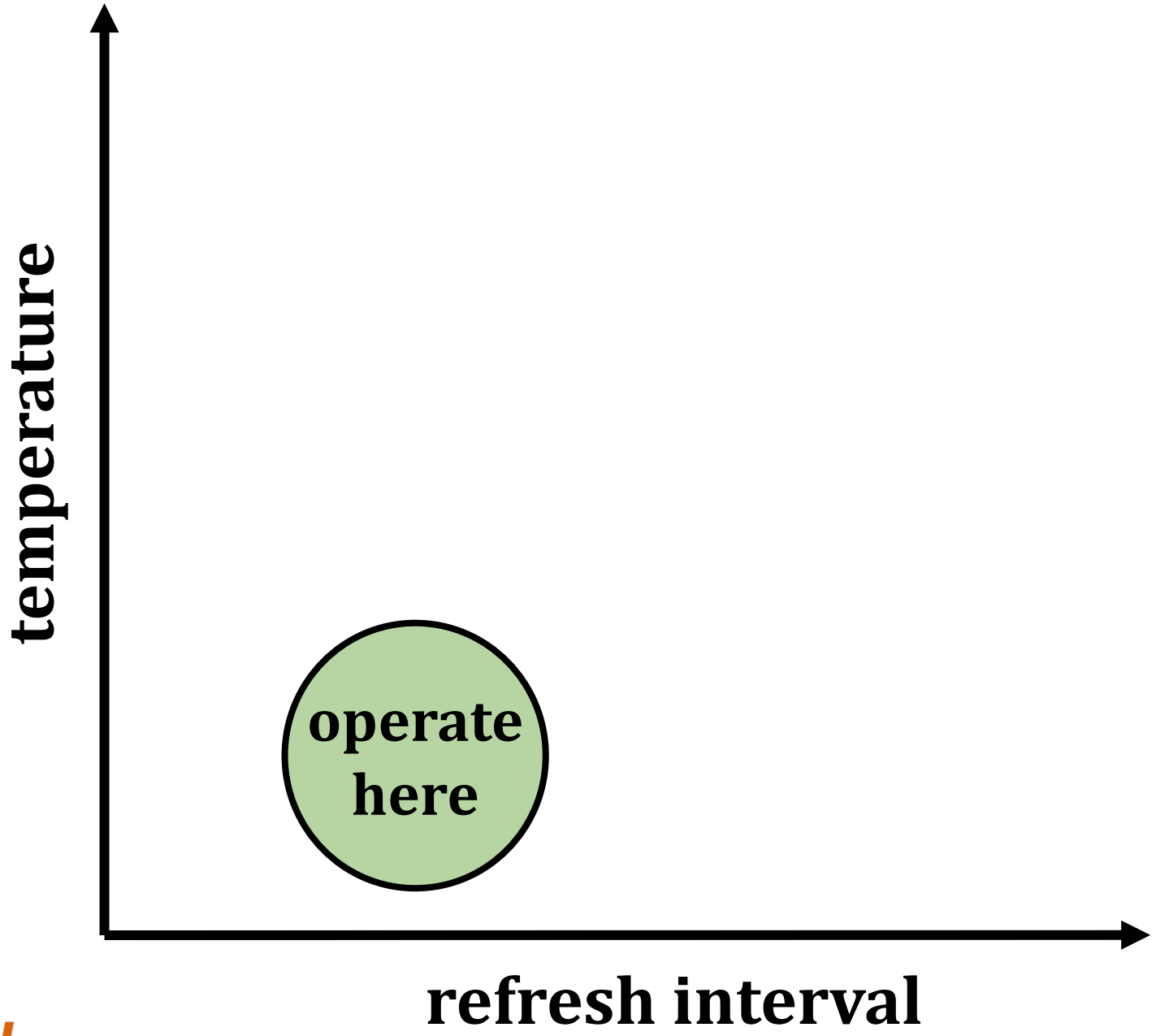
Variable retention time

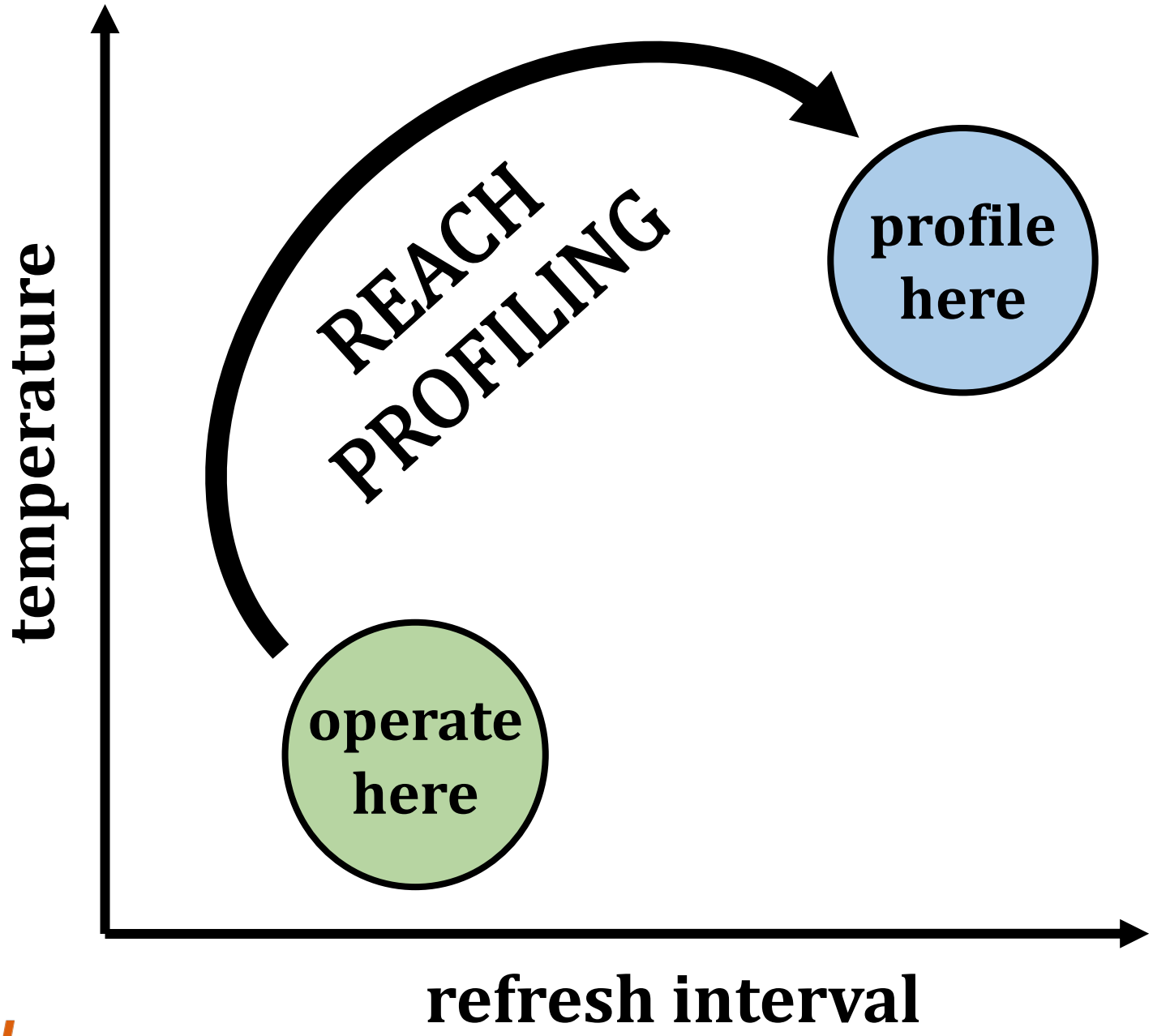
Data pattern dependence

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)**


temperature
operate here
refresh interval


temperature
REACH
PROFILING
profile
here
operate
here
refresh interval

Reach Profiling

A new DRAM retention failure profiling methodology

+ **Faster** and **more reliable** than current approaches

+ Enables **longer refresh intervals**

REAPER Outline

1. DRAM Refresh Background
2. Failure Profiling Challenges
3. Current Approaches
4. **LPDDR4 Characterization**
5. Reach Profiling
6. End-to-end Evaluation

Experimental Infrastructure

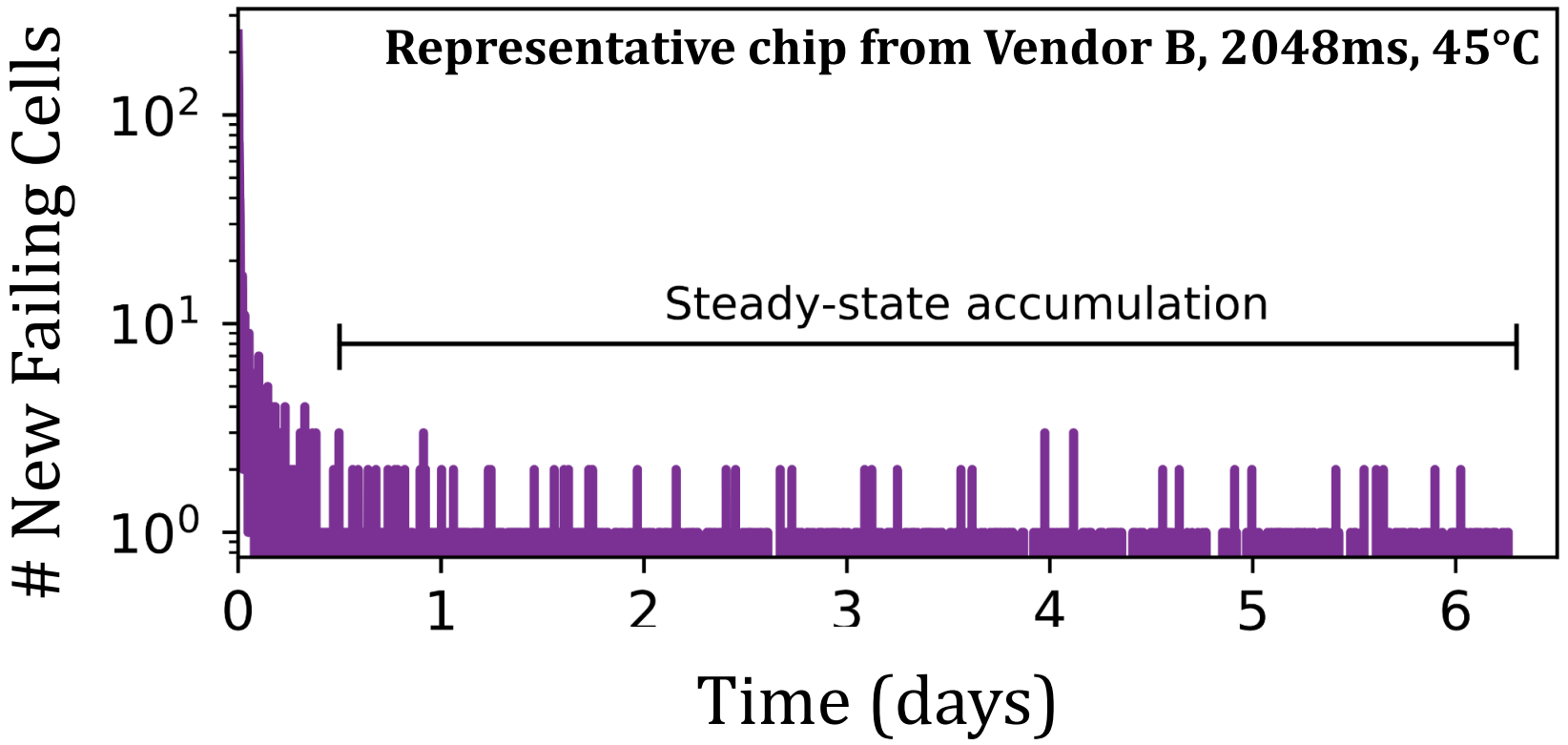
- **368 2y-nm LPDDR4 DRAM chips**
  - 4Gb chip size
  - From 3 major DRAM vendors

- **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

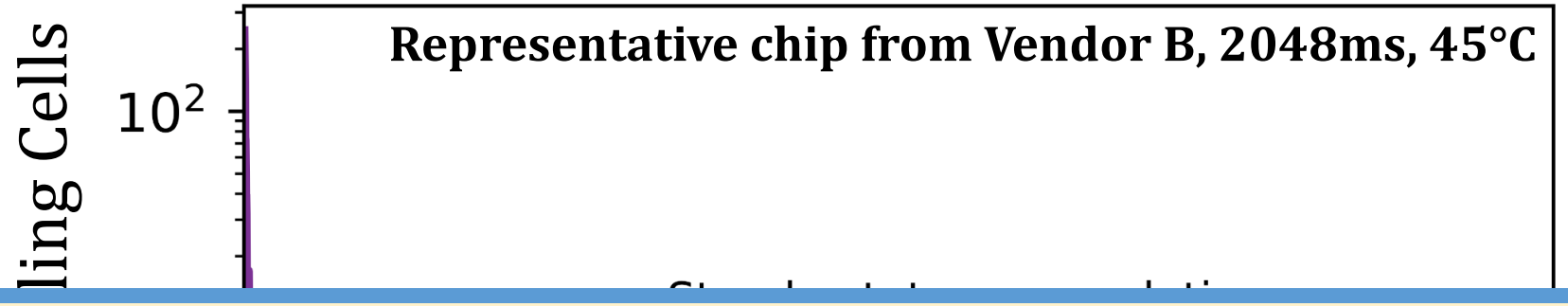
1. Temperature
2. Data Pattern Dependence
3. Retention Time Distributions
4. **Variable Retention Time**
5. **Individual Cell Characterization**

Long-term Continuous Profiling



- New failing cells continue to appear over time
  - Attributed to **variable retention time (VRT)**
- The set of failing cells changes over time

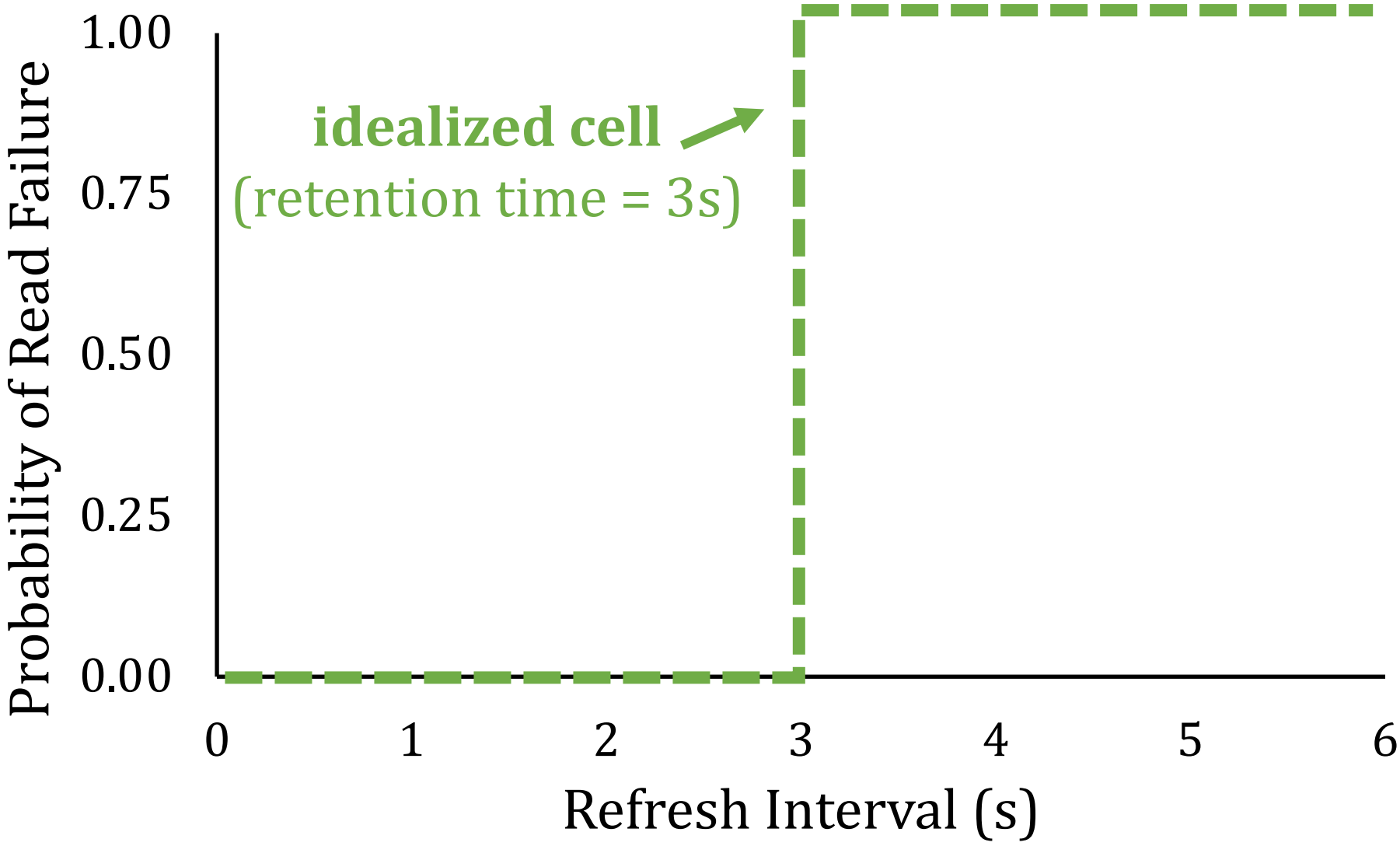
Long-term Continuous Profiling



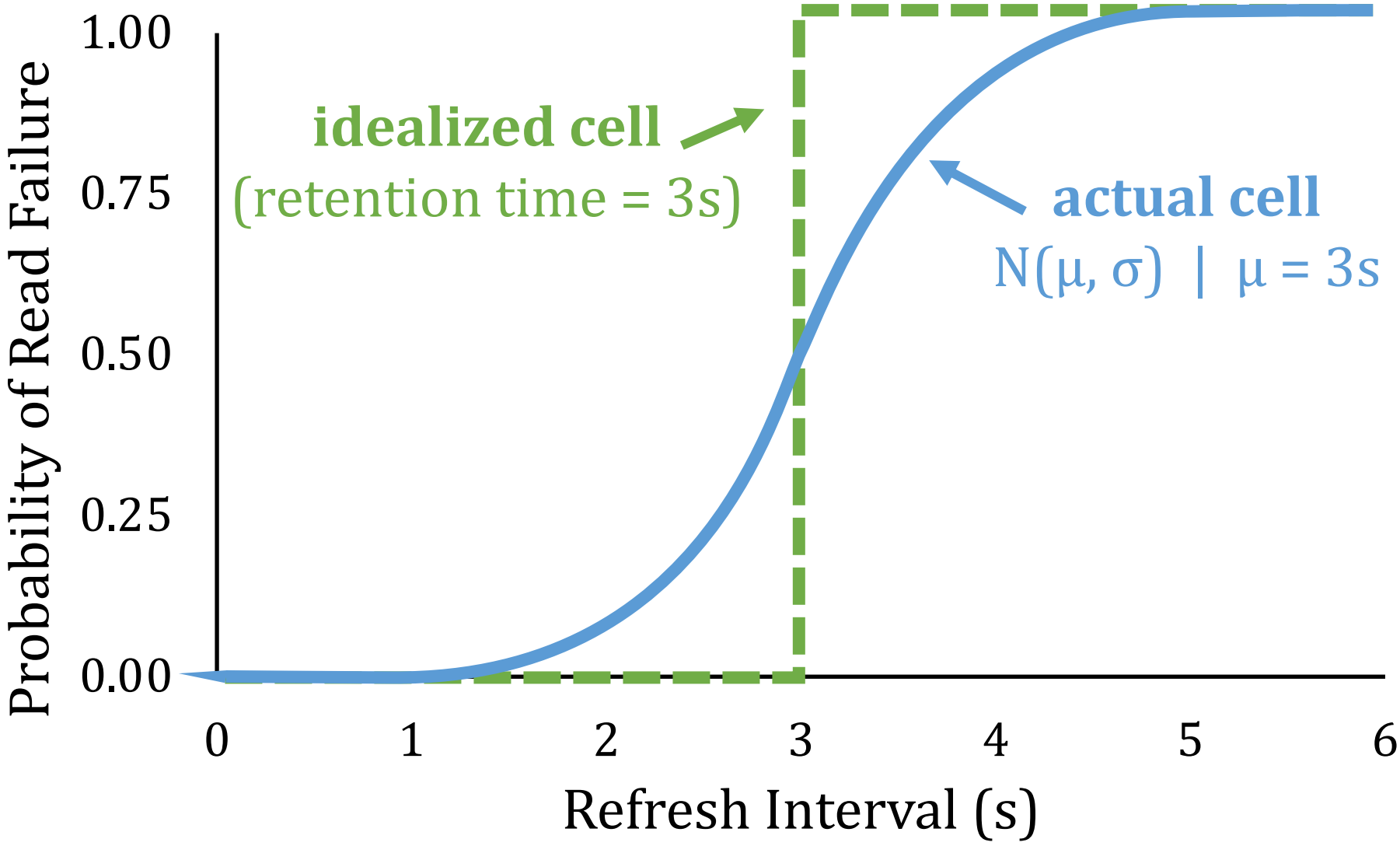
Error correction codes (ECC) and online profiling are *necessary* to manage new failing cells

- New failing cells continue to appear over time
  - Attributed to **variable retention time (VRT)**
- The set of failing cells changes over time

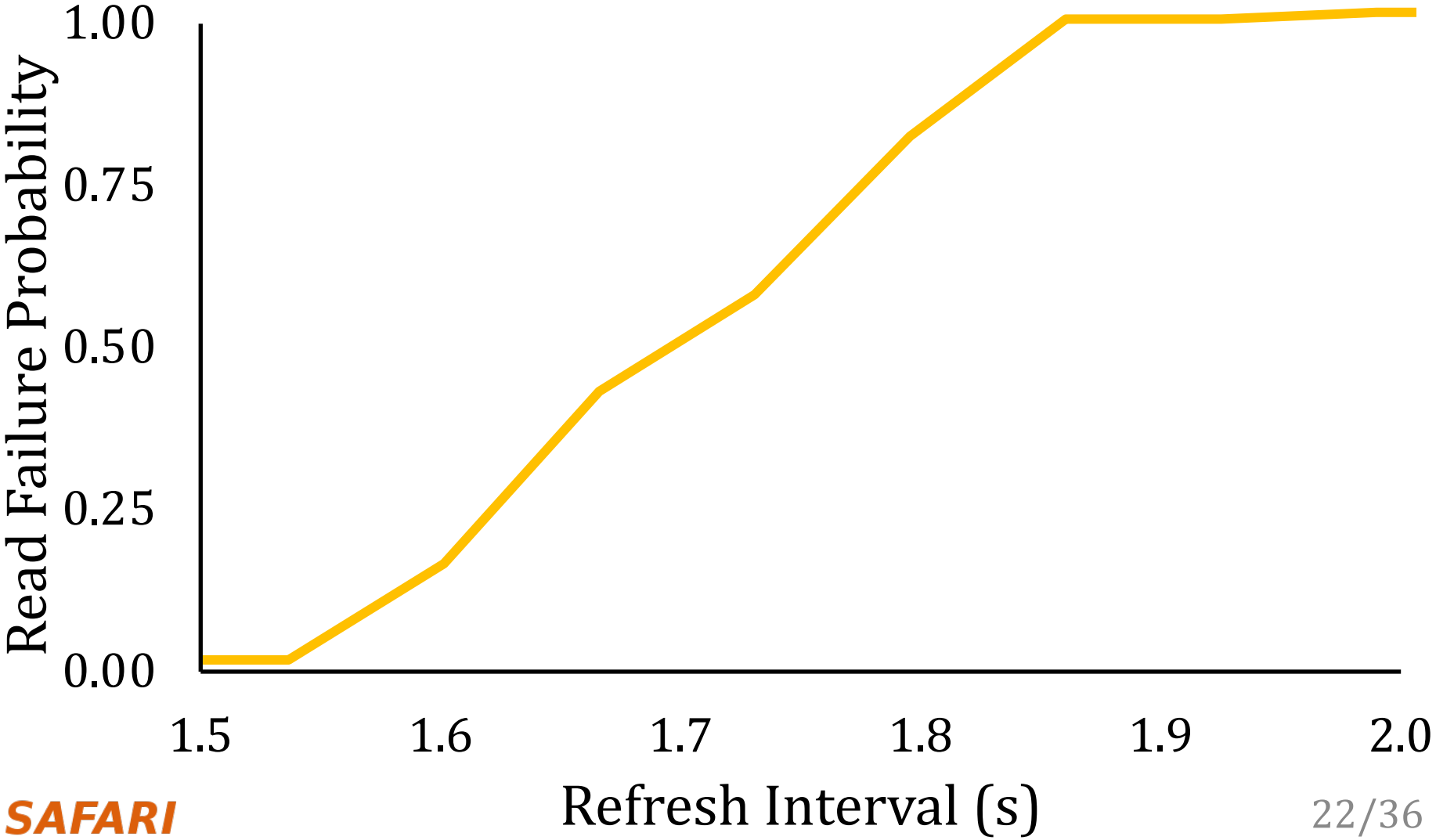
Single-cell Failure Probability (Cartoon)



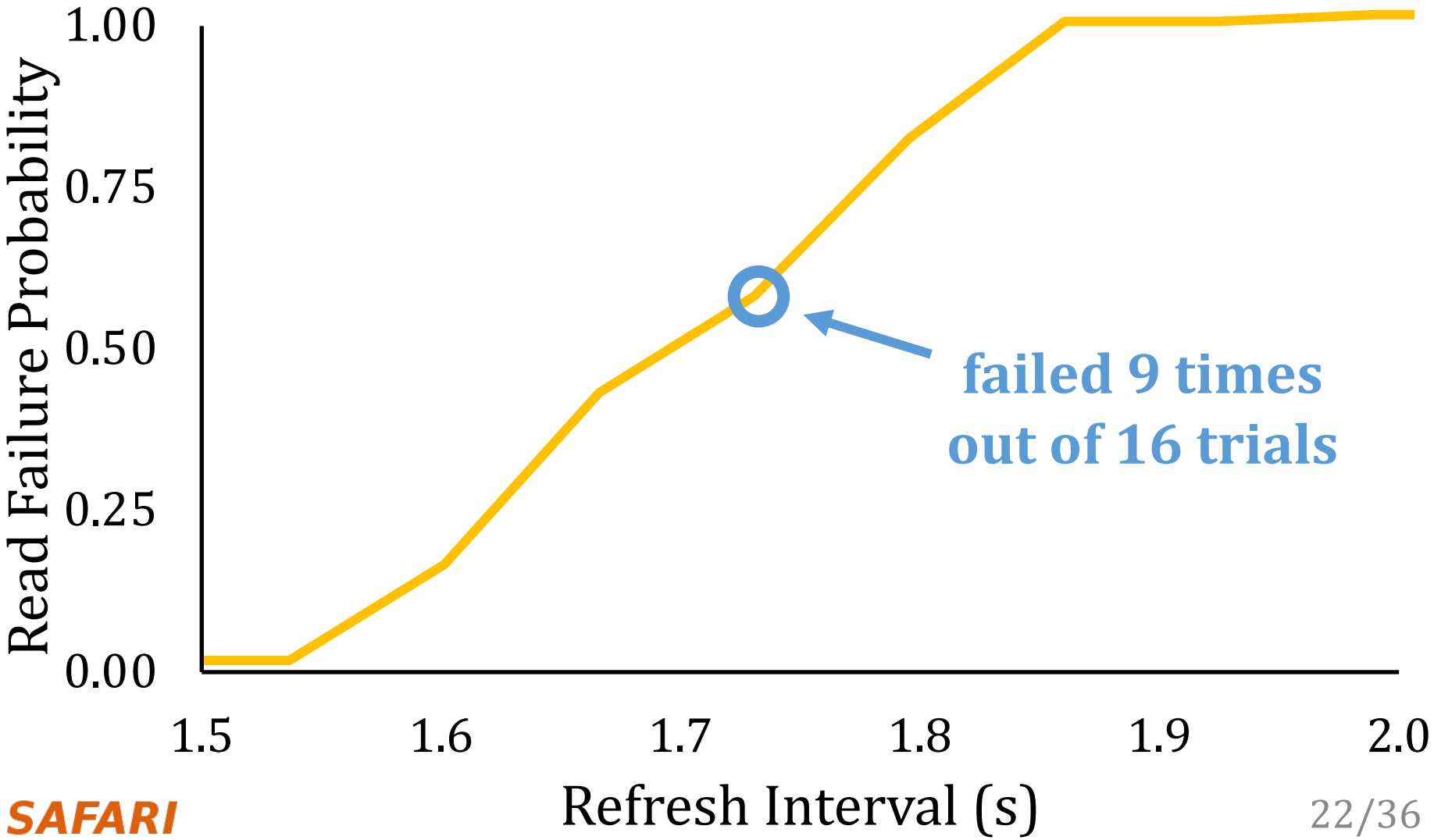
Single-cell Failure Probability (Cartoon)



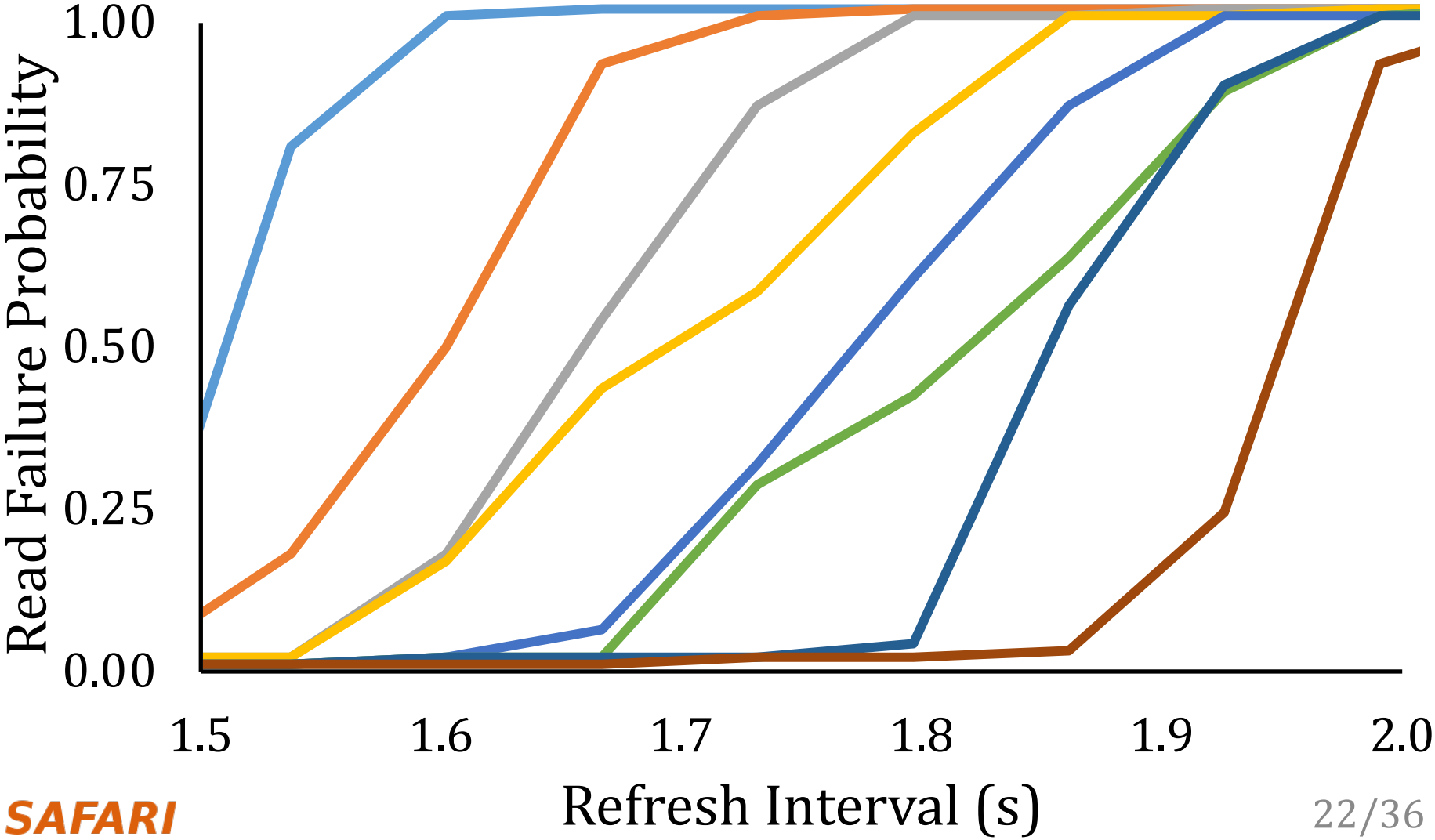
Single-cell Failure Probability (Real)



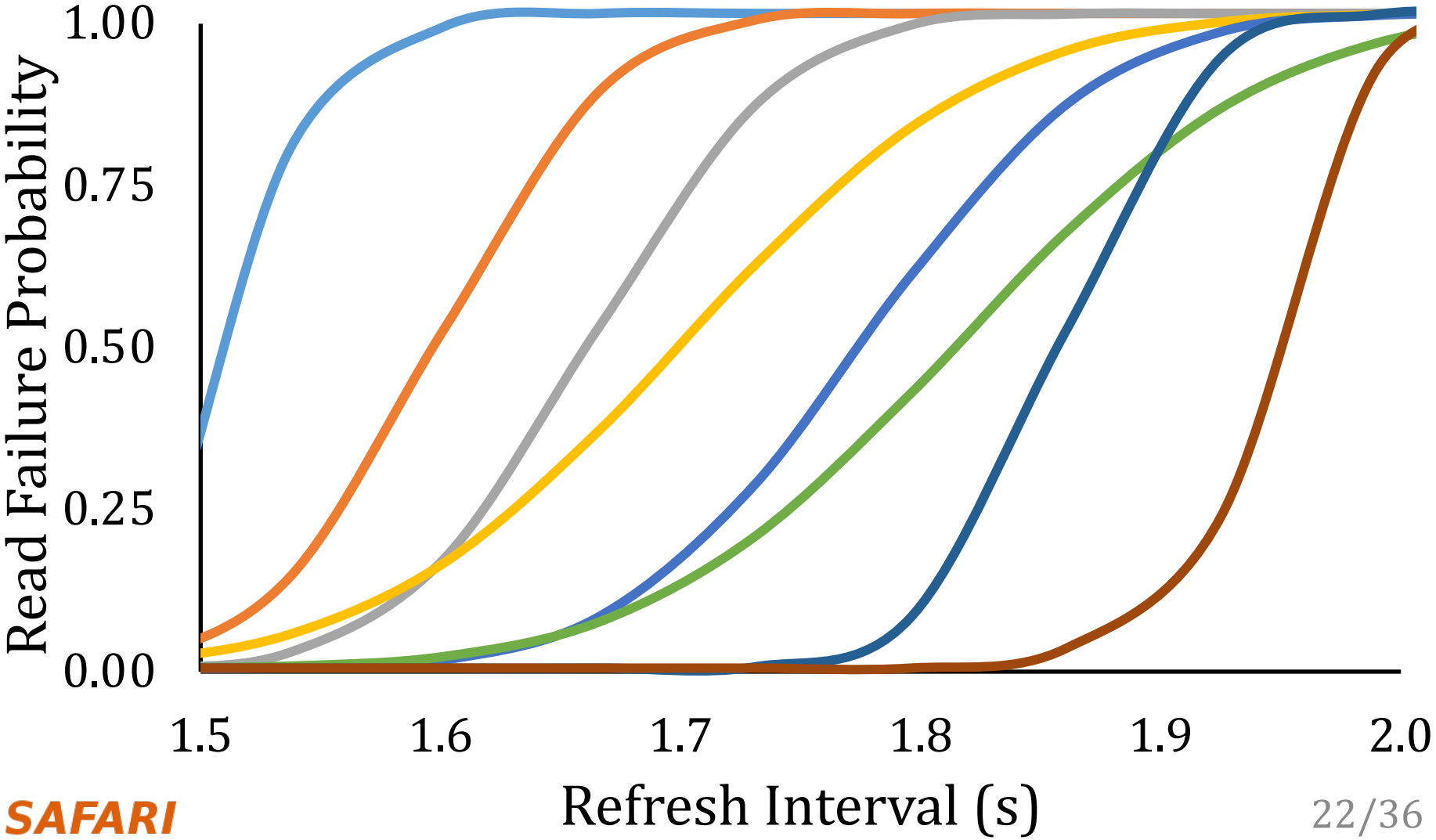
Single-cell Failure Probability (Real)



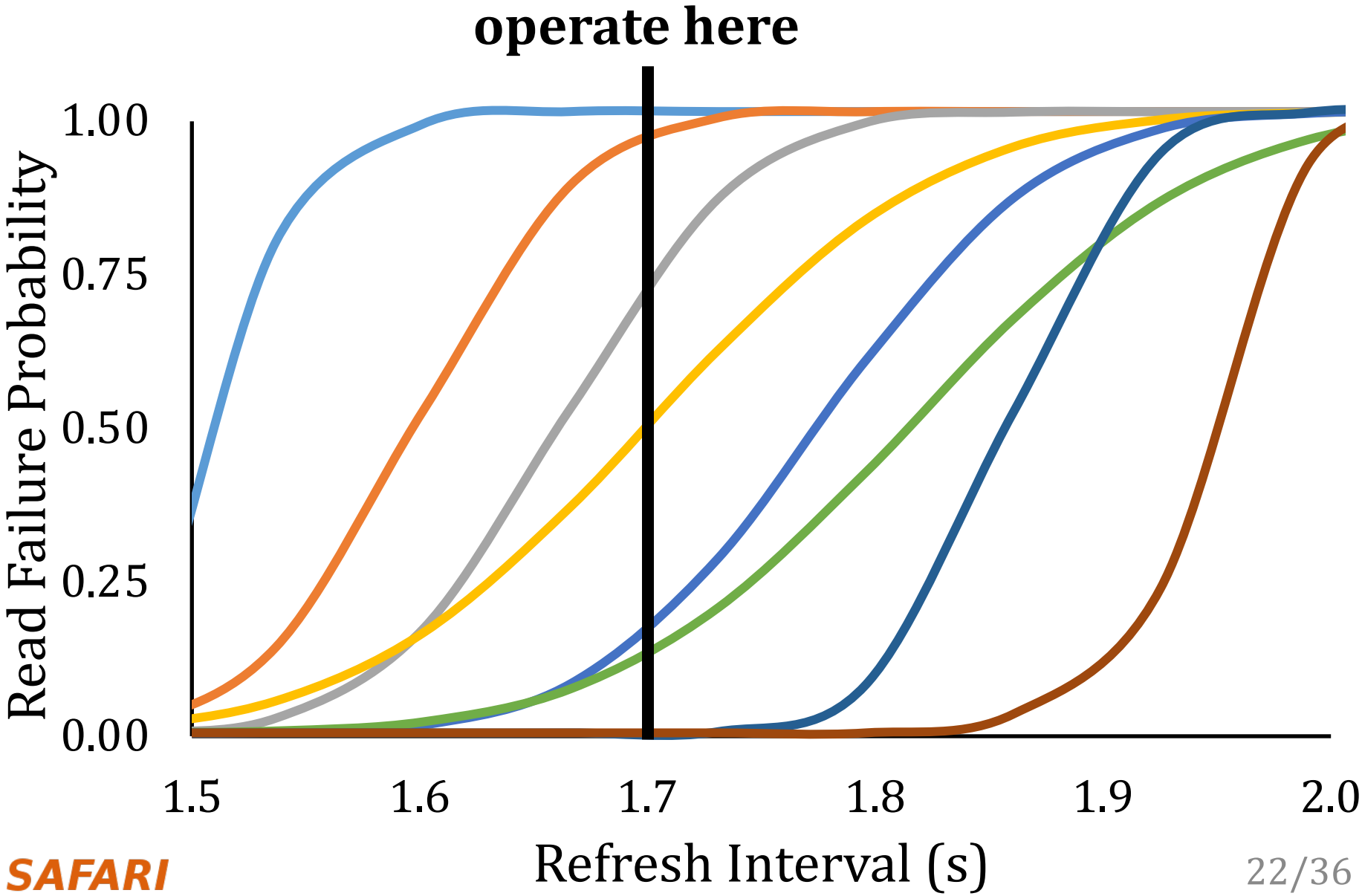
Single-cell Failure Probability (Real)



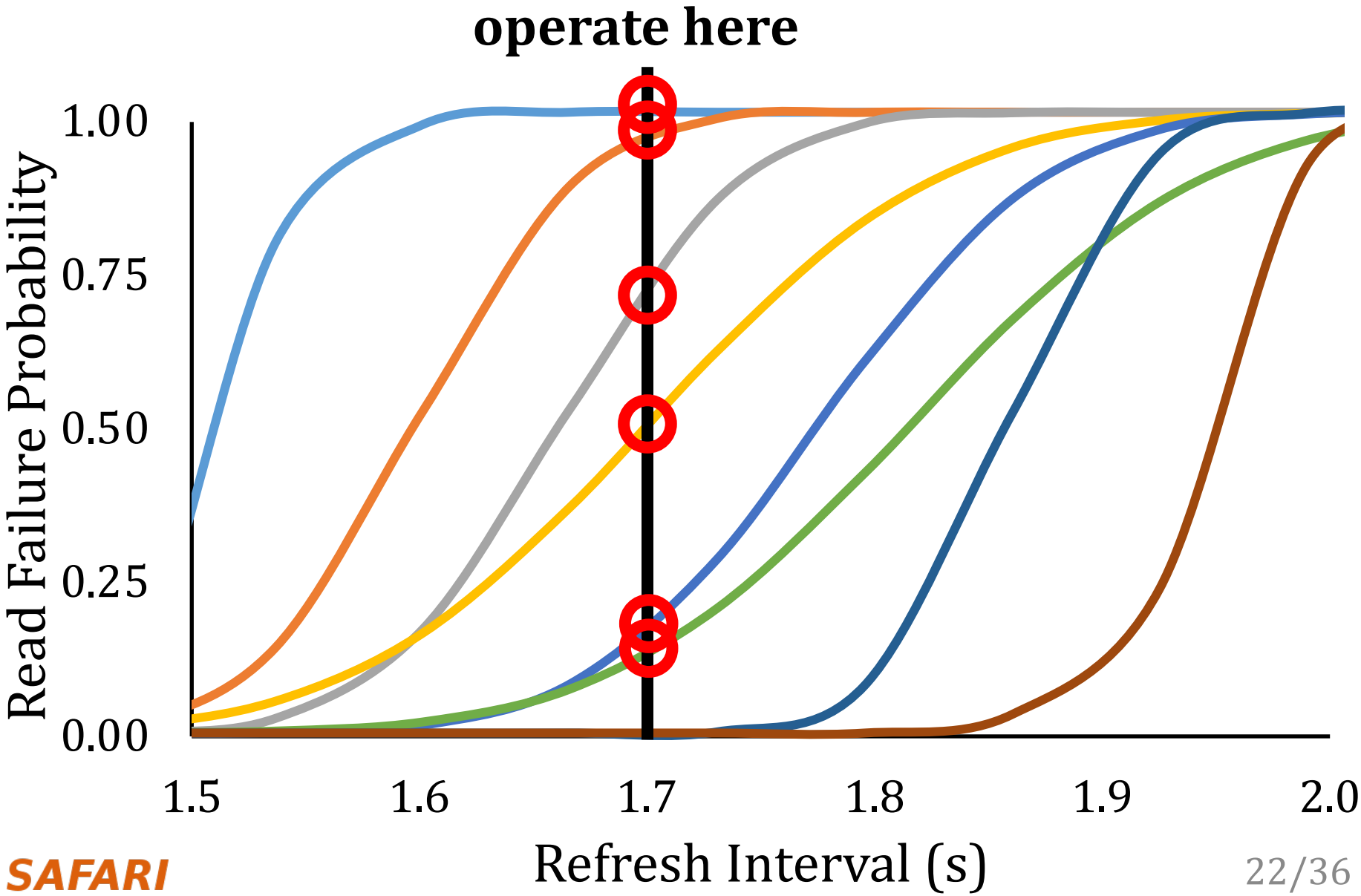
Single-cell Failure Probability (Real)



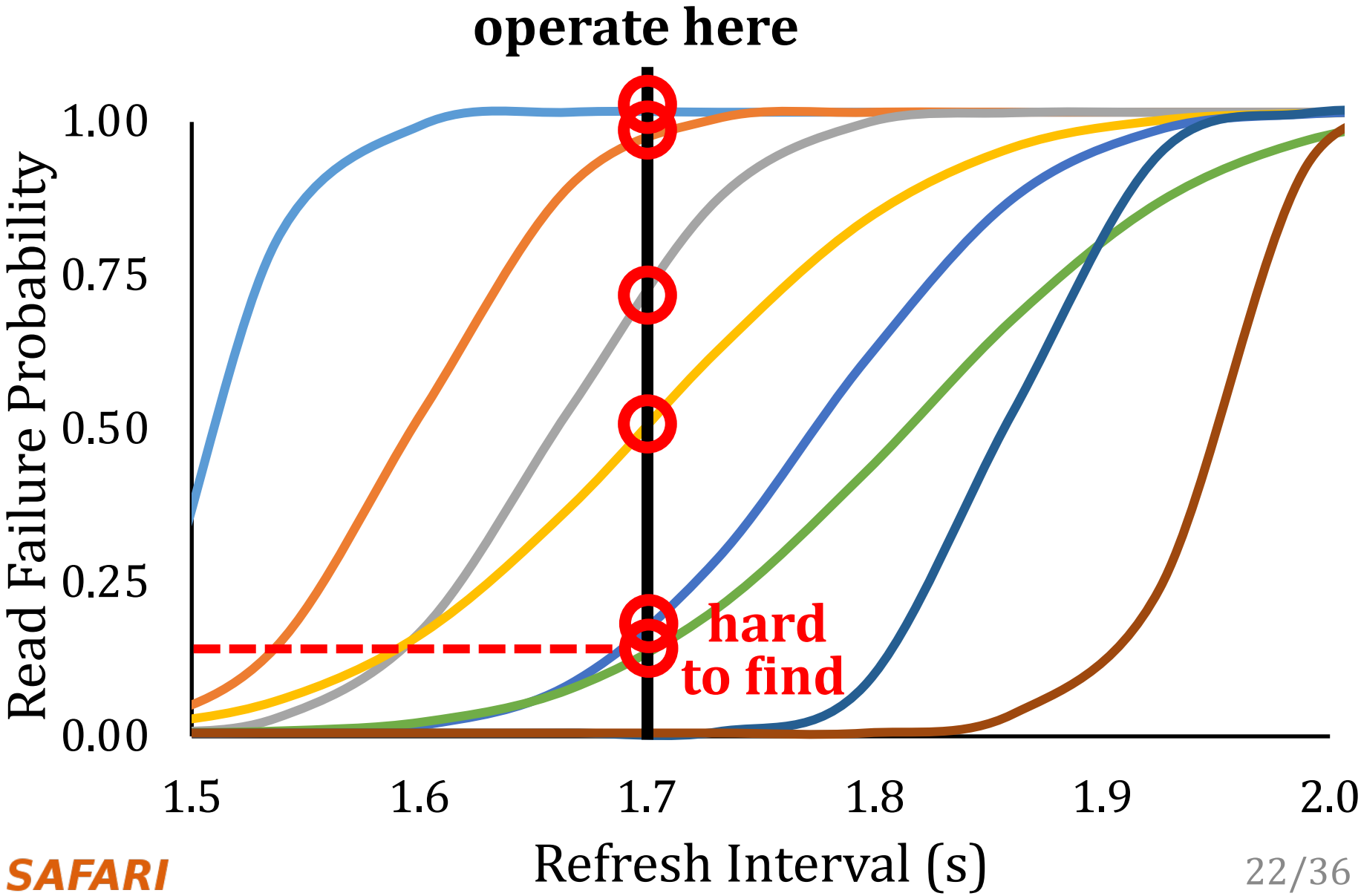
Single-cell Failure Probability (Real)



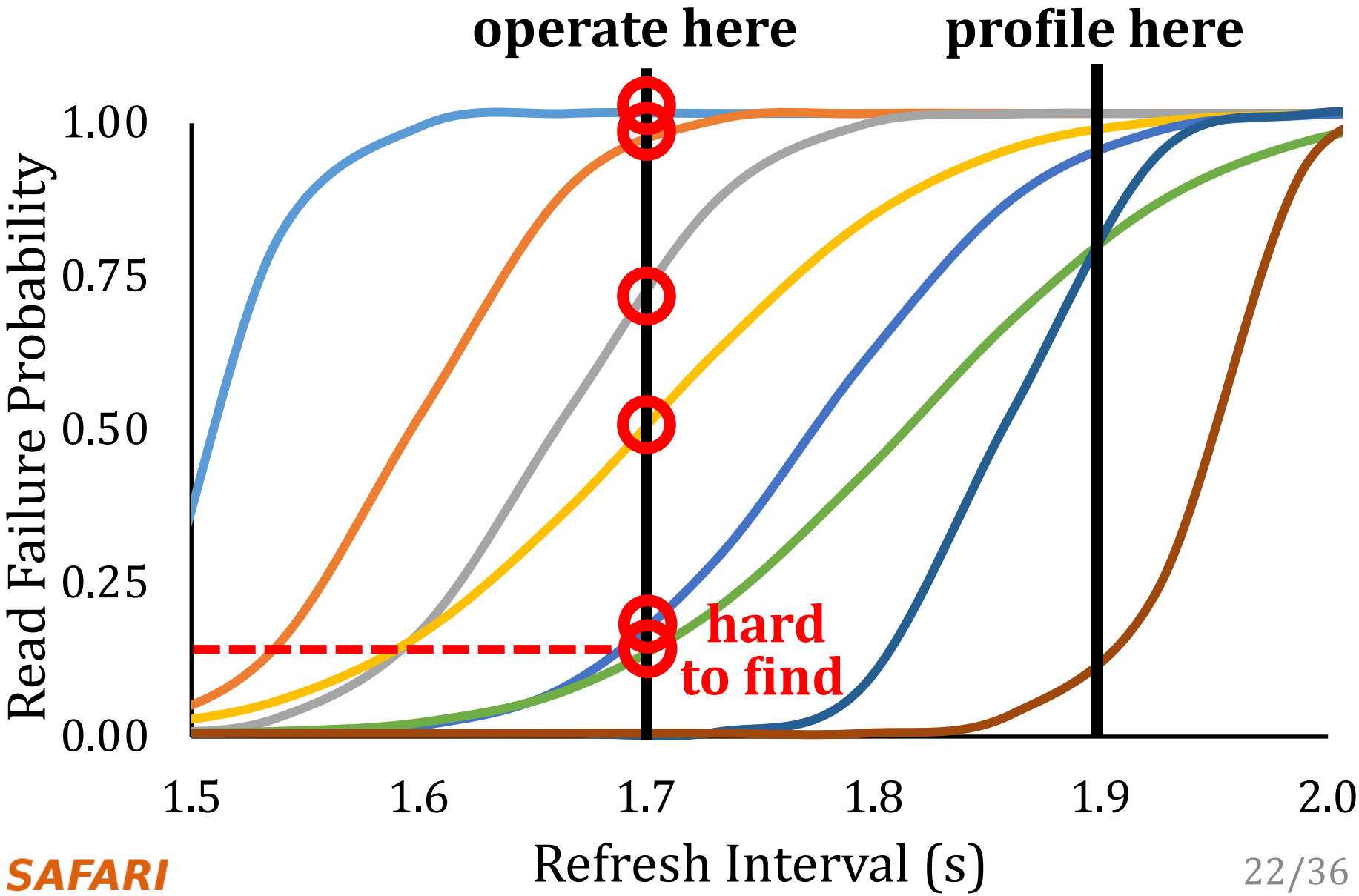
Single-cell Failure Probability (Real)



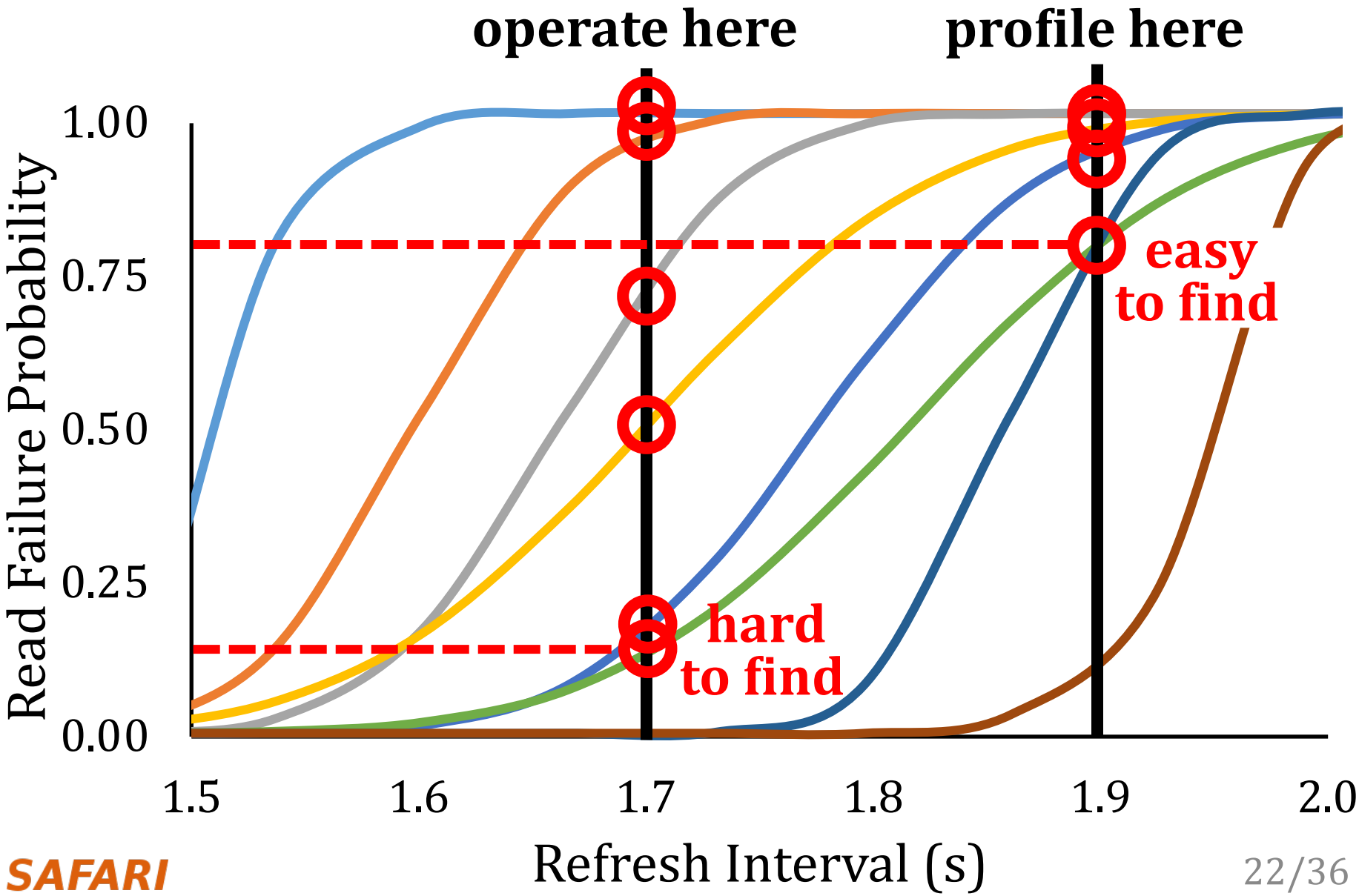
Single-cell Failure Probability (Real)



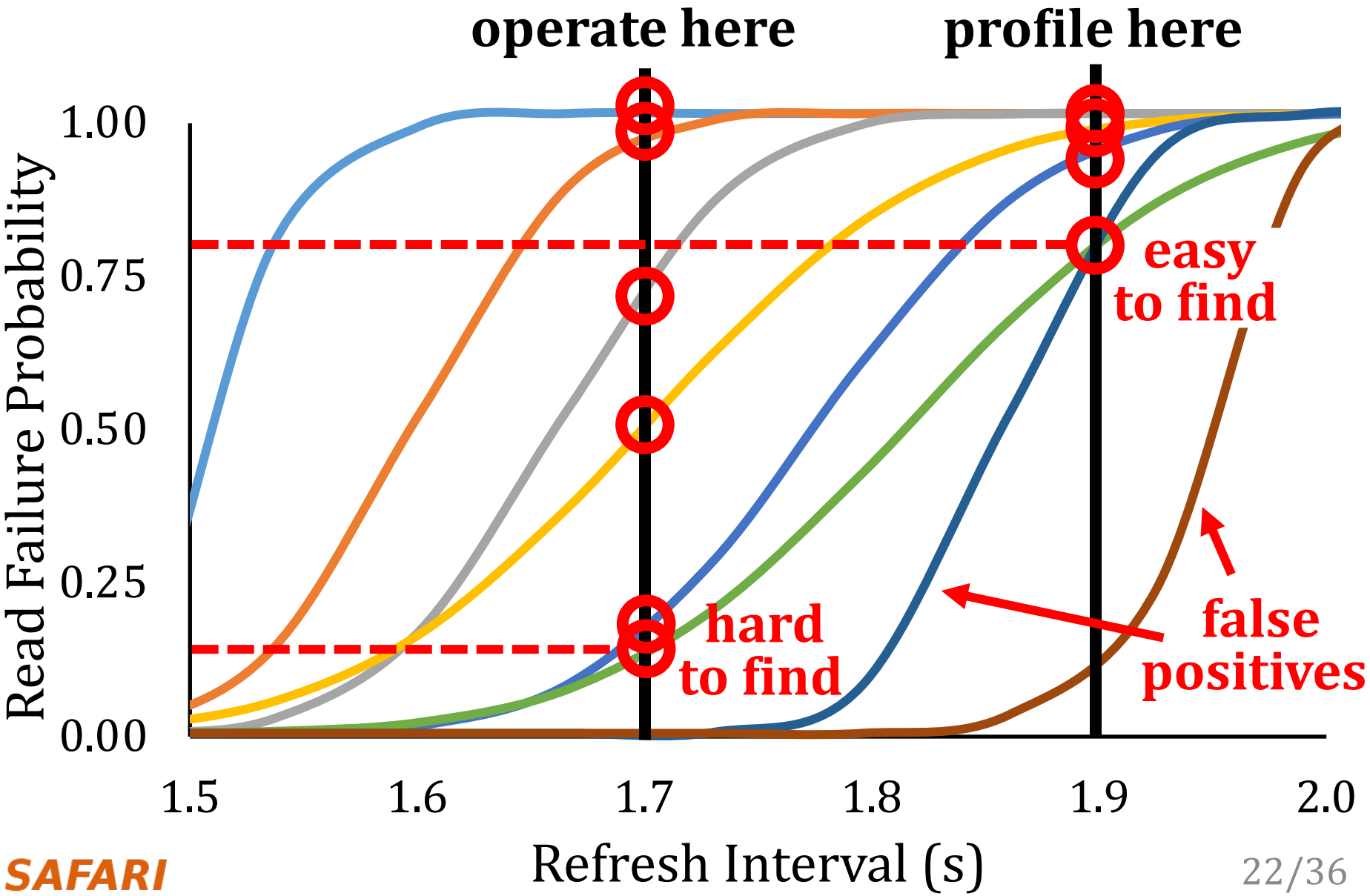
Single-cell Failure Probability (Real)



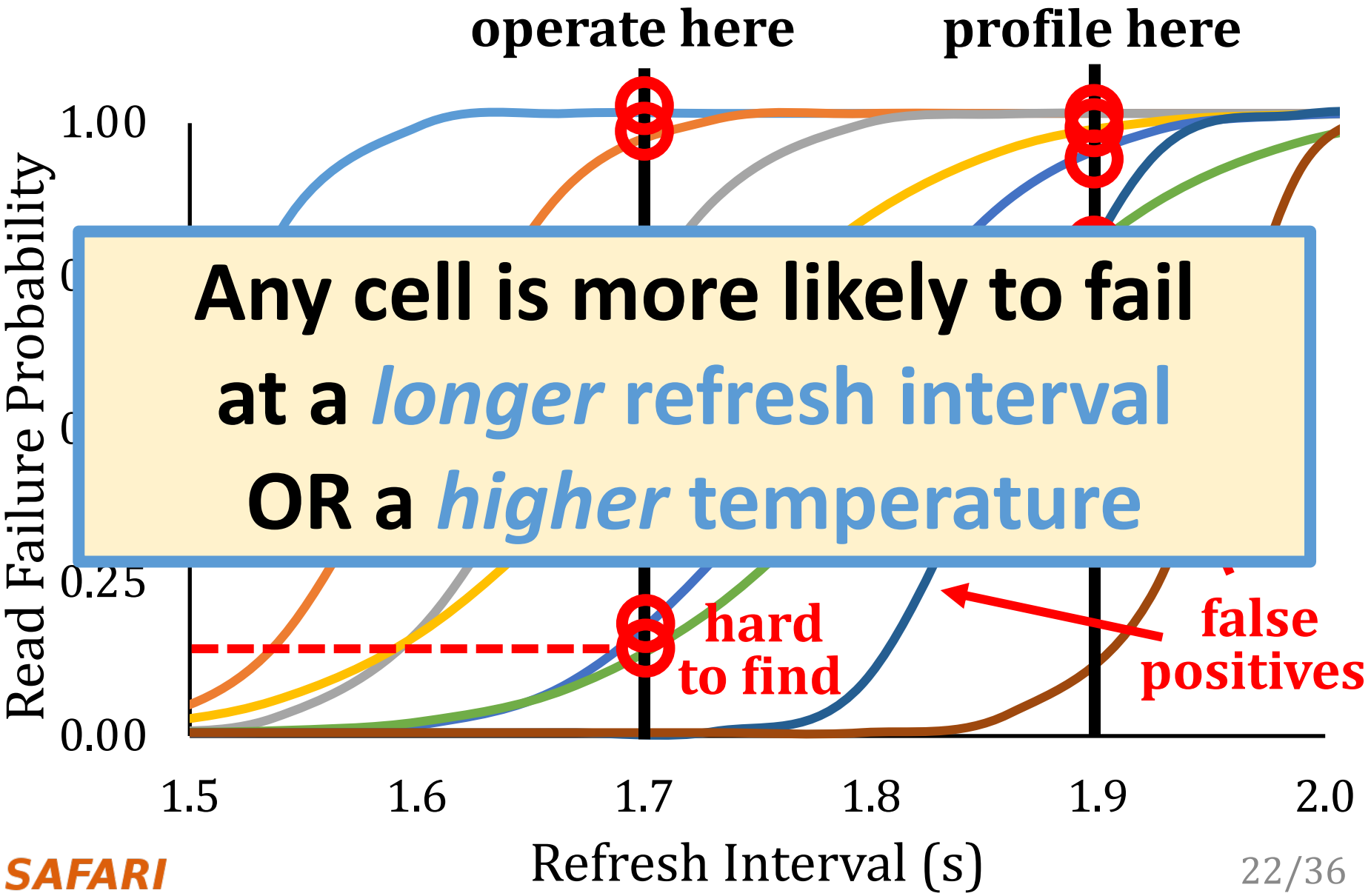
Single-cell Failure Probability (Real)



Single-cell Failure Probability (Real)



Single-cell Failure Probability (Real)

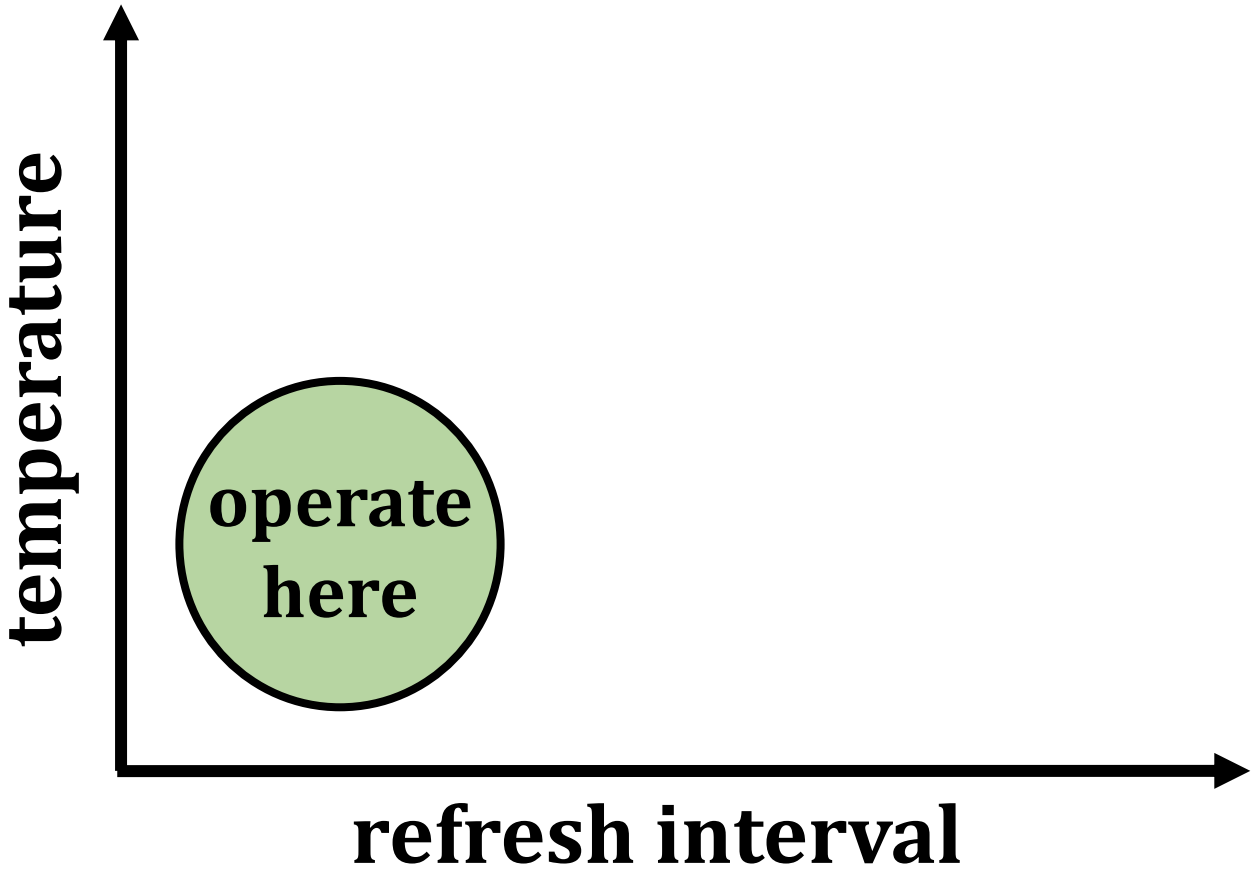


REAPER Outline

1. DRAM Refresh Background
2. Failure Profiling Challenges
3. Current Approaches
4. LPDDR4 Characterization
5. **Reach Profiling**
6. End-to-end Evaluation

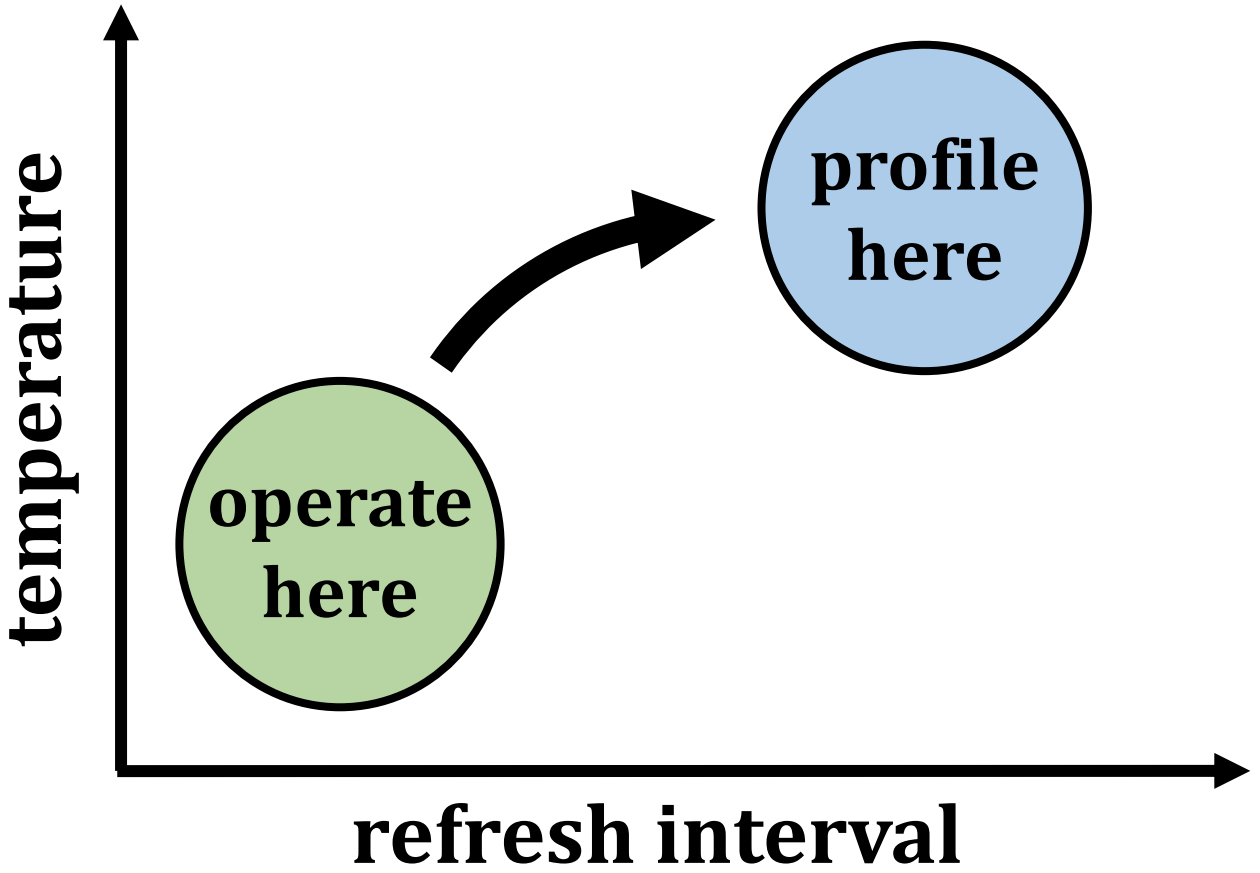
Reach Profiling

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*



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

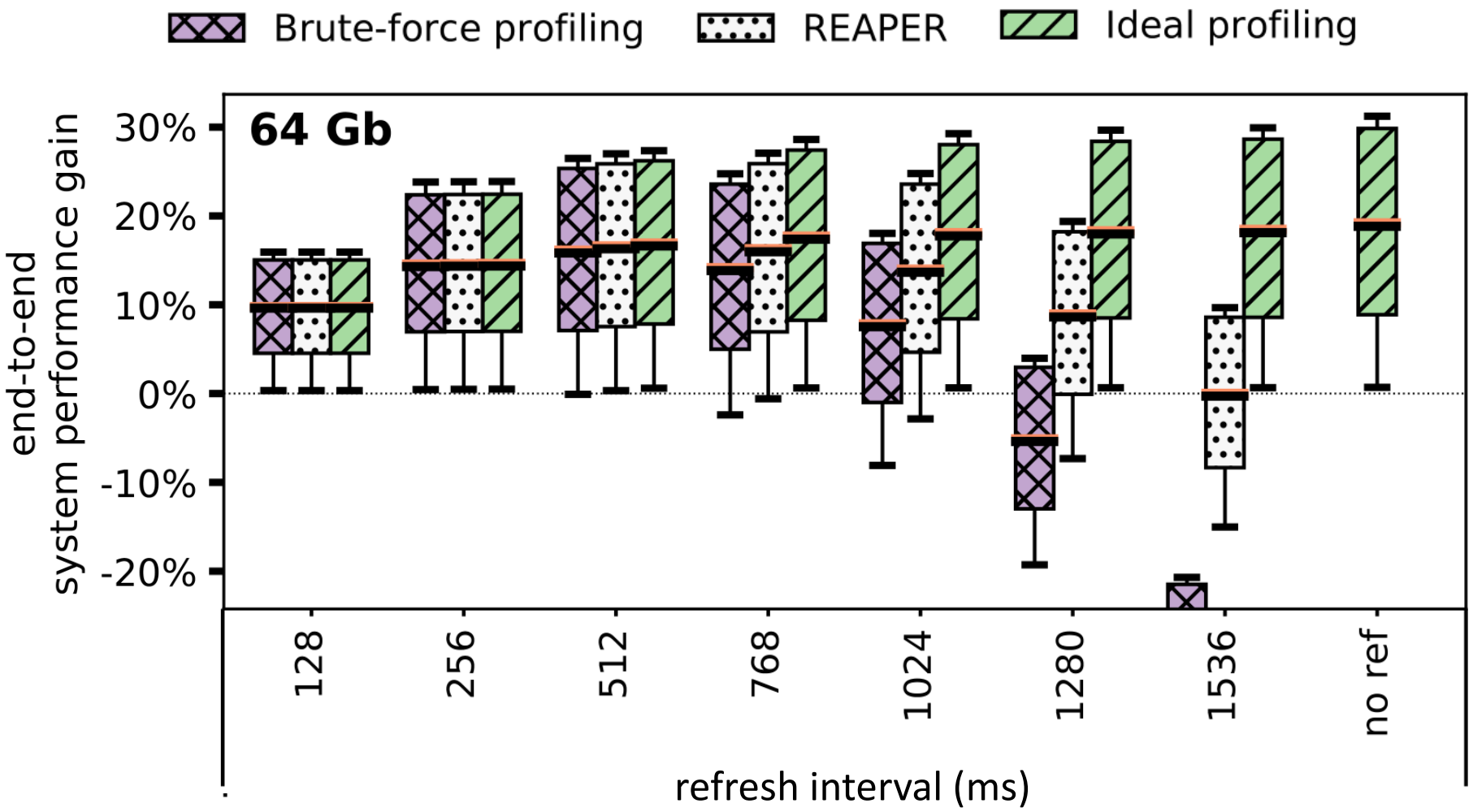
Evaluation Methodology

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

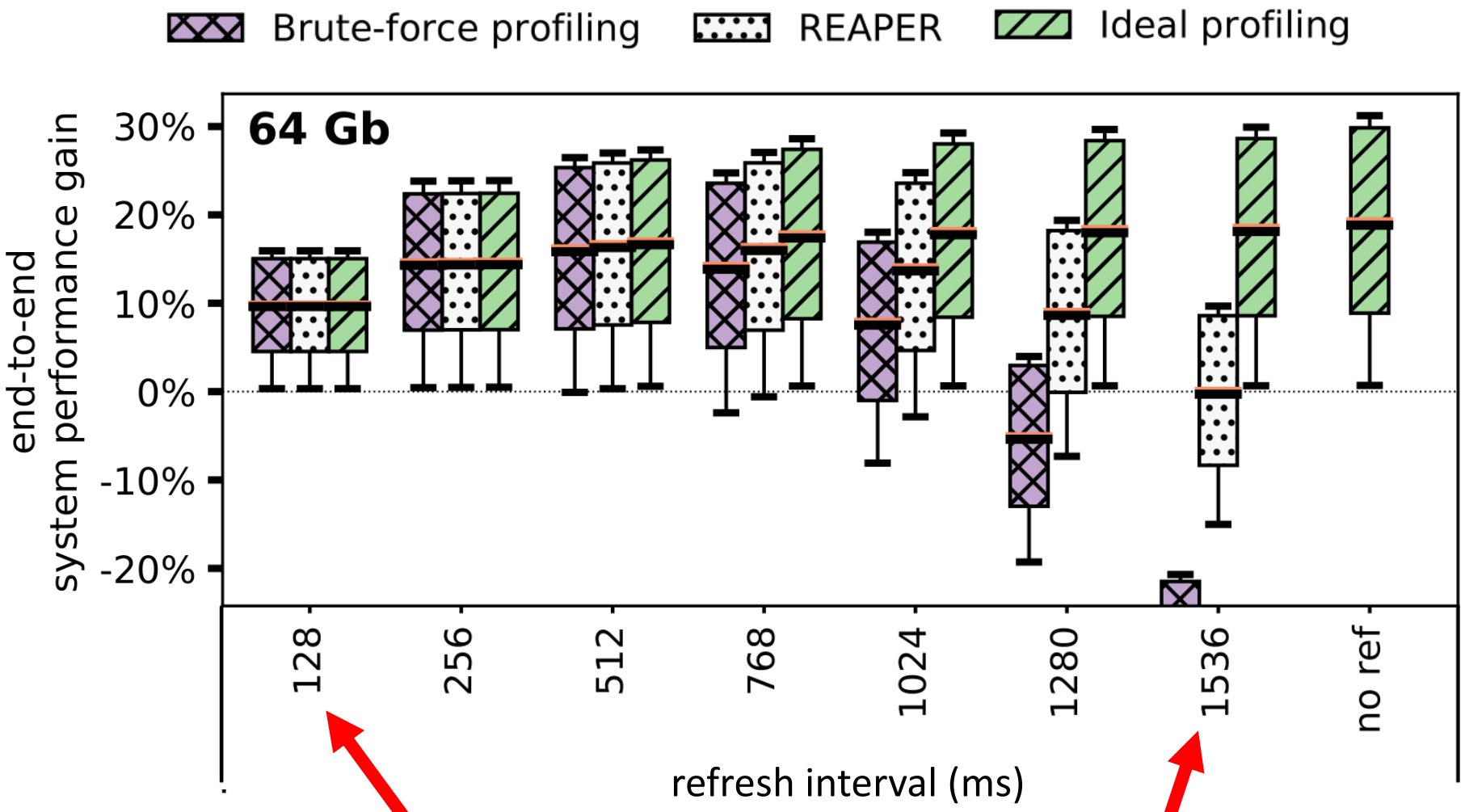
- Configuration
  - 4-core (4GHz), 8MB LLC
  - LPDDR4-3200, 4 channels, 1 rank/channel

- Workloads
  - 20 random 4-core benchmark mixes
  - SPEC CPU2006 benchmark suite

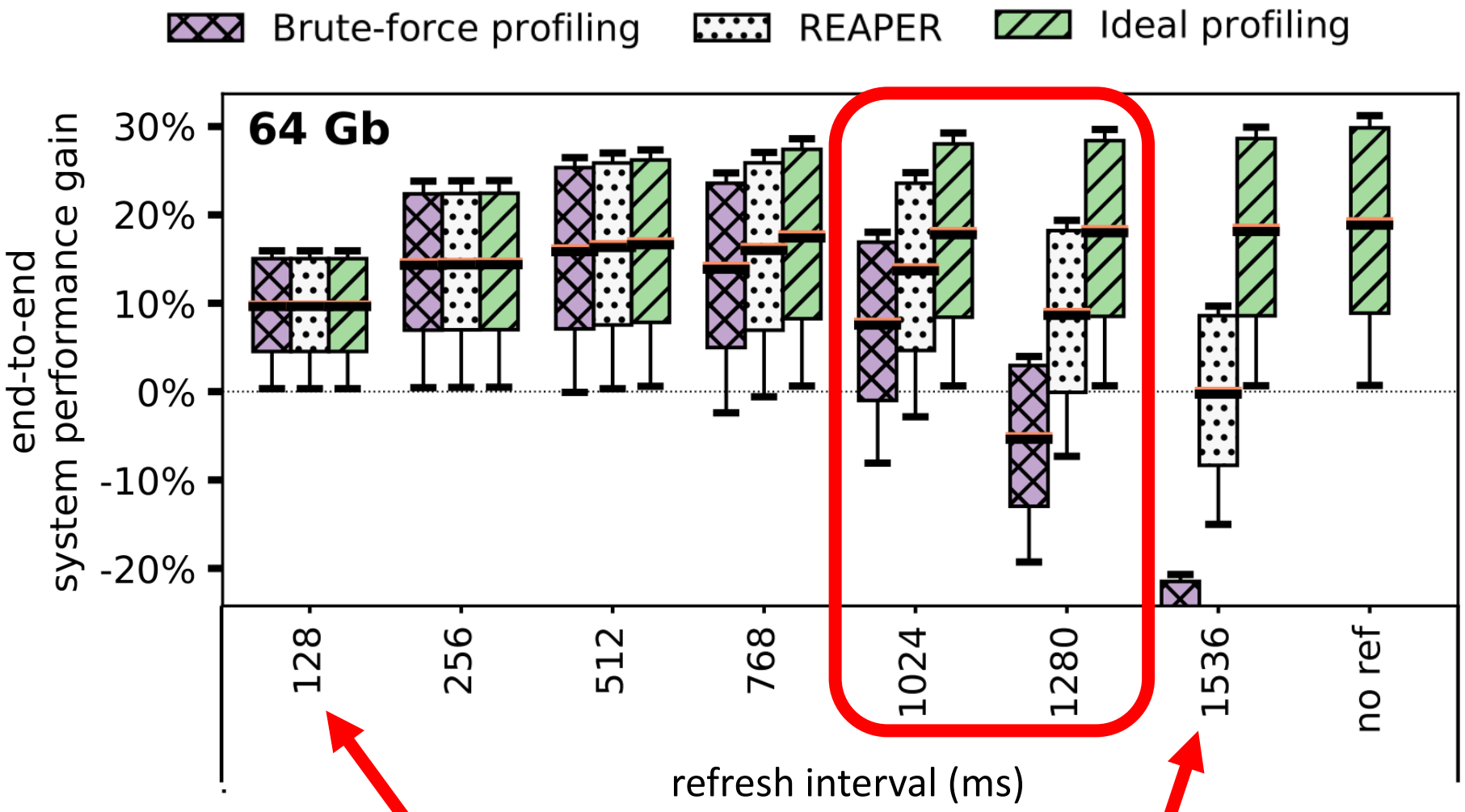
Simulated End-to-end Performance



Simulated End-to-end Performance



Simulated End-to-end Performance

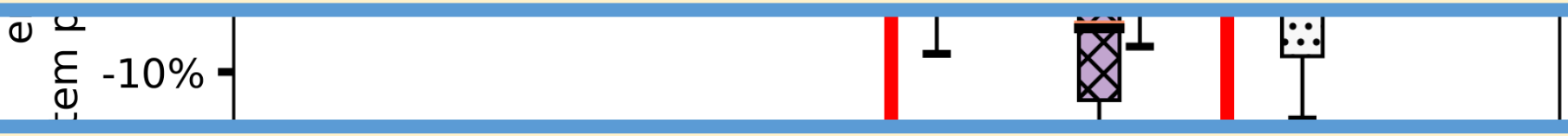


Simulated End-to-end Performance

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

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"
  *Proceedings of the 44th International Symposium on Computer Architecture (**ISCA**)*, Toronto, Canada, June 2017.
  [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"
  *Proceedings of the 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (**DSN**)*, Portland, OR, USA, June 2019.
  [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*

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"
  Proceedings of the 53rd International Symposium on Microarchitecture (***MICRO***), Virtual, October 2020.
  [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