Variable Read Disturbance (VRD)

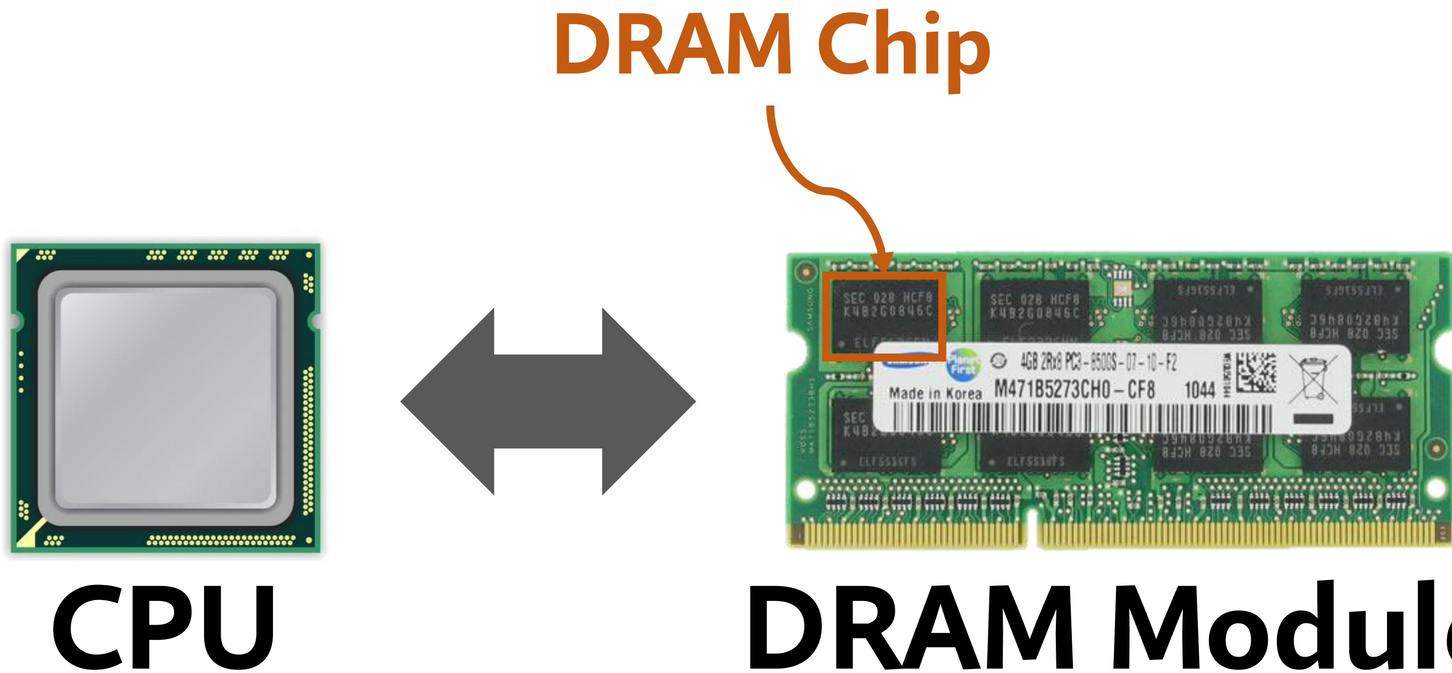
An Experimental Analysis of Temporal Variation in DRAM Read Disturbance

Ataberk Olgun, F. Nisa Bostancı, İsmail Emir Yüksel
Oğuzhan Canpolat, Haocong Luo, Geraldo F. Oliveira
A. Giray Yağlıkçı, Minesh Patel, Onur Mutlu

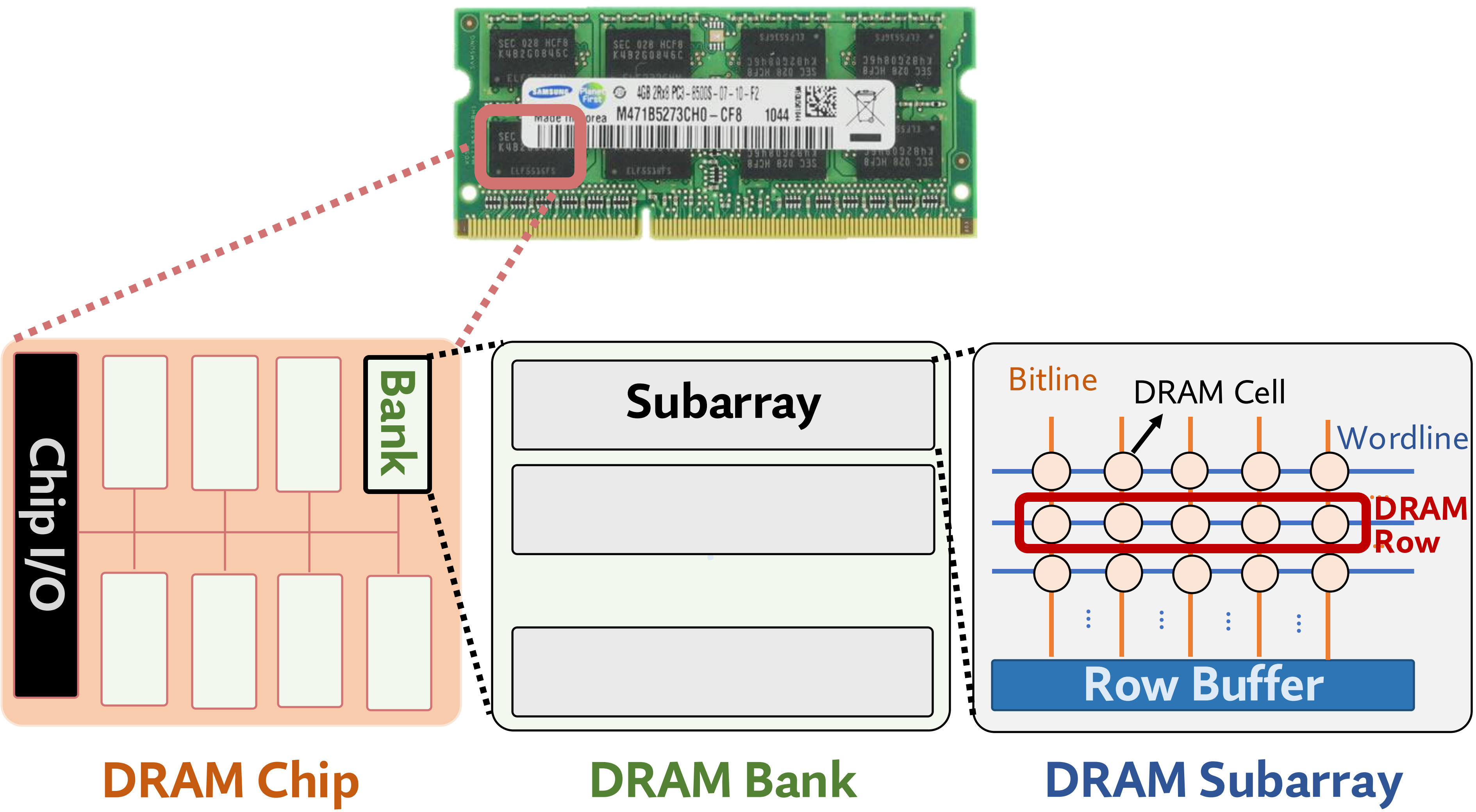
https://arxiv.org/pdf/2502.13075

ETH zürich SAFARI RUTGERS THE STATE UNIVERSITY OF NEW JERSEY

A Typical DRAM-Based Computing System

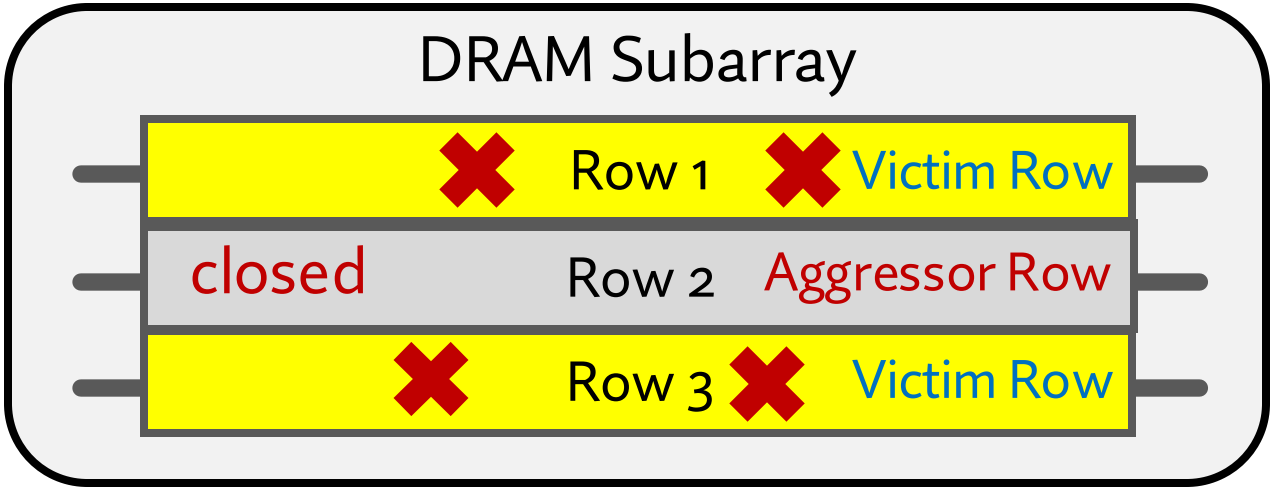


DRAM Organization



Read Disturbance in DRAM (I)

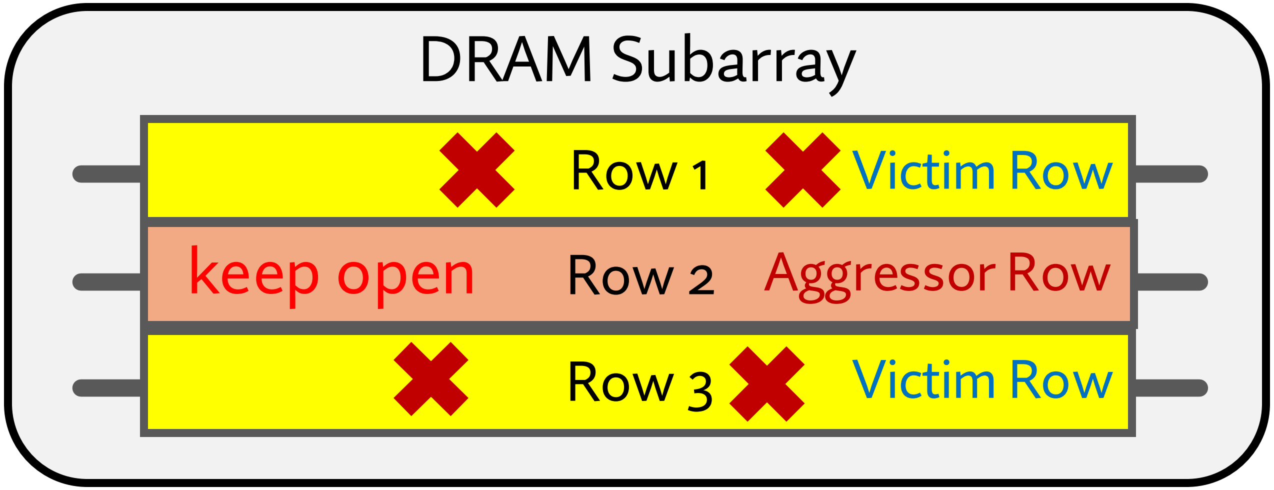
- Read disturbance in DRAM breaks memory isolation
- Prominent example: RowHammer



Repeatedly opening (activating) and closing a DRAM row many times causes RowHammer bitflips in adjacent rows

Read Disturbance in DRAM (II)

- Read disturbance in DRAM breaks memory isolation
- A new read disturbance phenomenon: RowPress



Keeping a DRAM row open for a long time
causes bitflips in adjacent rows

[Luo+, ISCA'23]

Read Disturbance Solutions

There are many solutions to mitigate read disturbance bitflips

- More robust DRAM chips and/or error-correcting codes
- Increased refresh rate
- Physical isolation
- Row remapping
- Preventive refresh
- Proactive throttling

Each solution offers a different system design point
in reliability, performance, energy, and area tradeoff space

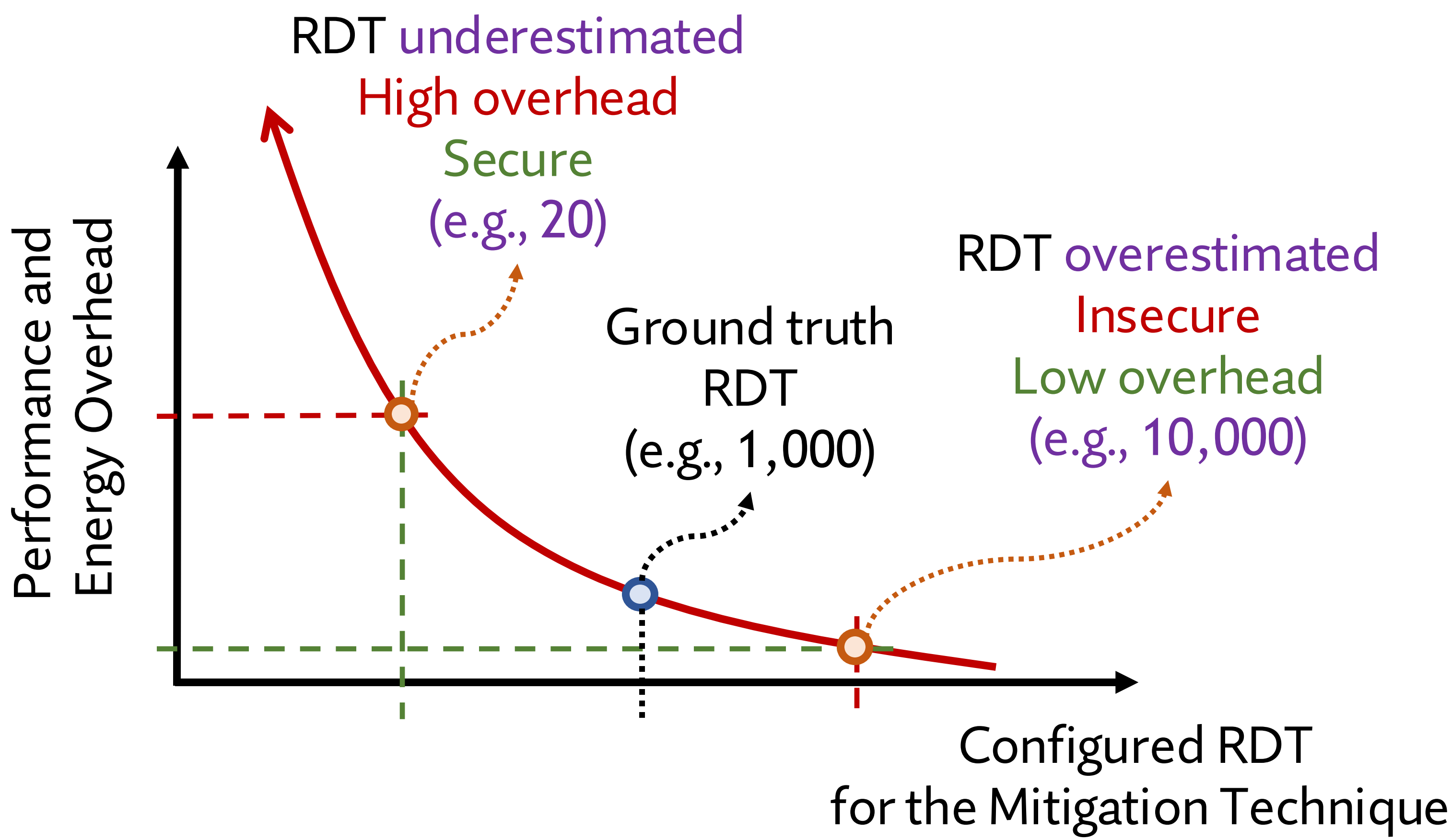
The Read Disturbance Threshold (RDT)

- Many secure read disturbance solutions take a preventive action before a bitflip manifests
  - E.g., preventively refresh a victim row

- Must **accurately quantify** the amount of disturbance that a row can withstand before experiencing a bitflip
  - Typically identified by testing for read disturbance failures

- Read Disturbance Threshold (RDT): The number of aggressor row activations needed to induce the first bitflip

Accurate Identification of Read Disturbance Threshold is Critical for System Security and Performance



To securely prevent bitflips at low overhead
RDT must be **accurately identified** and carefully configured

Variable Read Disturbance (VRD) Summary

Research Question

- How accurately and efficiently can we measure the read disturbance threshold (RDT) of each DRAM row?

Experimental Characterization

- Record >100M RDT measurements across 3750 rows and many test parameters (e.g., temperature, data pattern) in 160 DDR4 and 4 HBM2 chips

Key Observations

- RDT changes significantly and unpredictably over time: VRD
- Maximum observed RDT is 3.5X higher than minimum (for a row)
- Smallest RDT (for a row) may appear after 94,467 measurements

Implications for System Security and Robustness

- RDT cannot be accurately identified quickly
- *Given our limited dataset*, guardbands (>10%) and ECC (SECDED or Chipkill) *may* prevent VRD-induced bitflips at significant performance cost
  - More data and analyses needed to make definitive conclusion
- Call for future work on understanding and efficiently mitigating VRD

Talk Outline

I. Motivation

II. Experimental Characterization Methodology

III. Foundational Results

IV. In-Depth Analysis of VRD

V. Implications for System Security and Robustness

VI. Conclusion

Talk Outline

I. **Motivation**

II. Experimental Characterization Methodology

III. Foundational Results

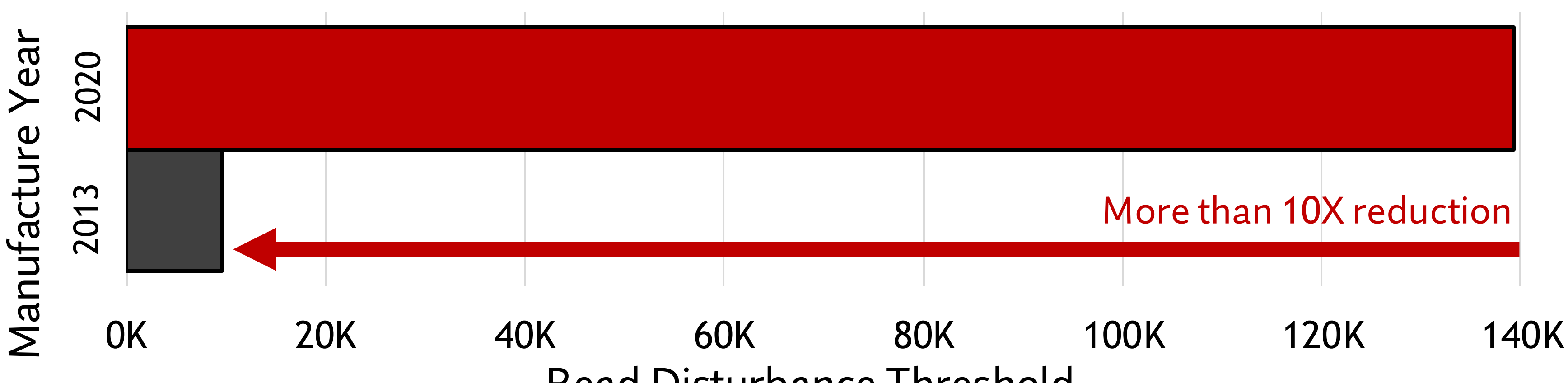
IV. In-Depth Analysis of VRD

V. Implications for System Security and Robustness

VI. Conclusion

Motivation

DRAM chips are increasingly more vulnerable to read disturbance with technology scaling



[Yaglikci+, HPCA'24]

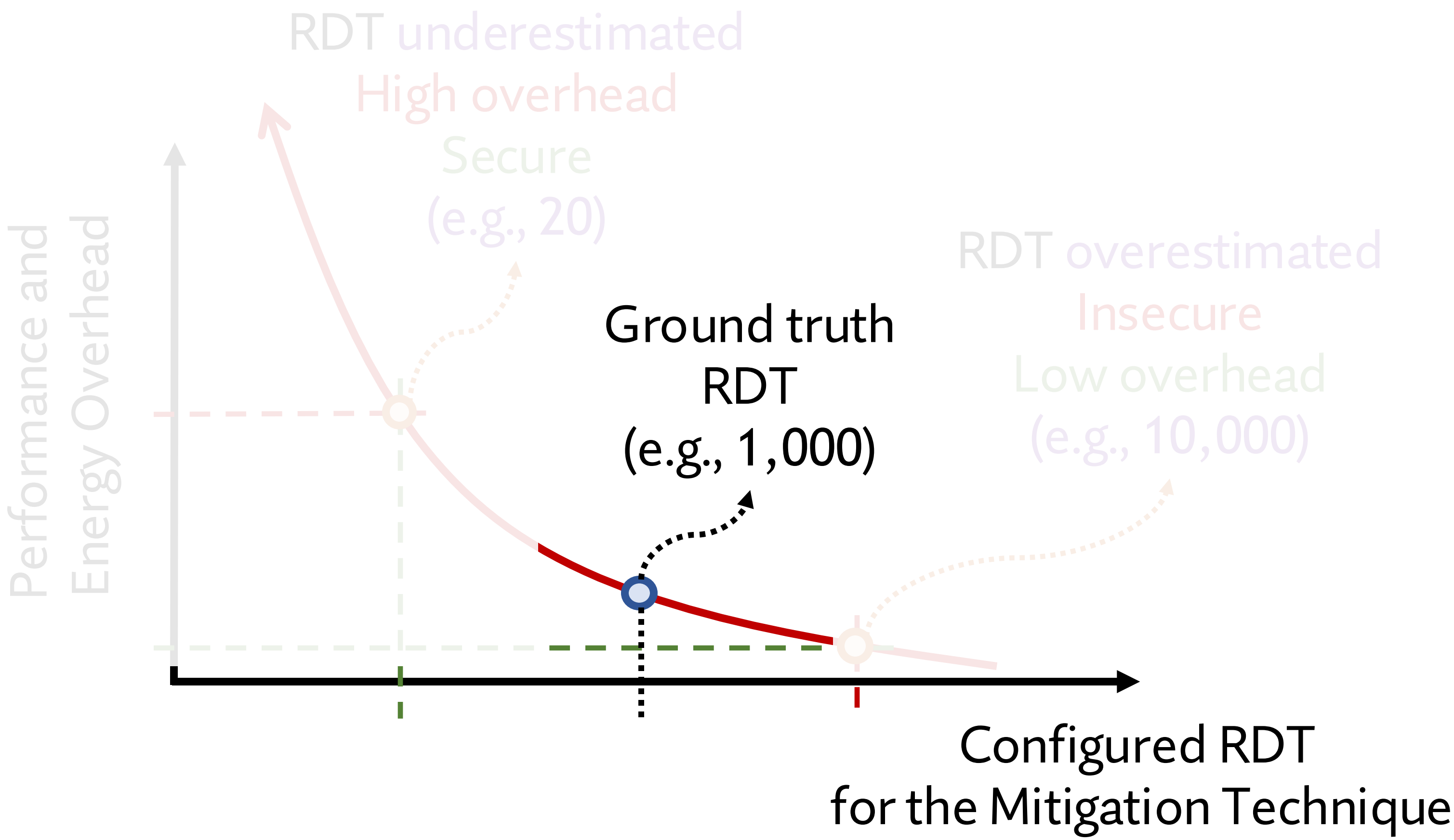
Motivation

DRAM read disturbance worsens
as DRAM chip density increases

Existing solutions become more aggressive

Aggressive preventive actions make
existing solutions prohibitively expensive

Motivation



Prior works assume that the **ground truth**
Read Disturbance Threshold (RDT) **can be identified**

Problem

No prior work rigorously studies

temporal variation of

DRAM read disturbance threshold

&

implications for future solutions

Our Goal

Answer two research questions:

1. Does RDT change over time?

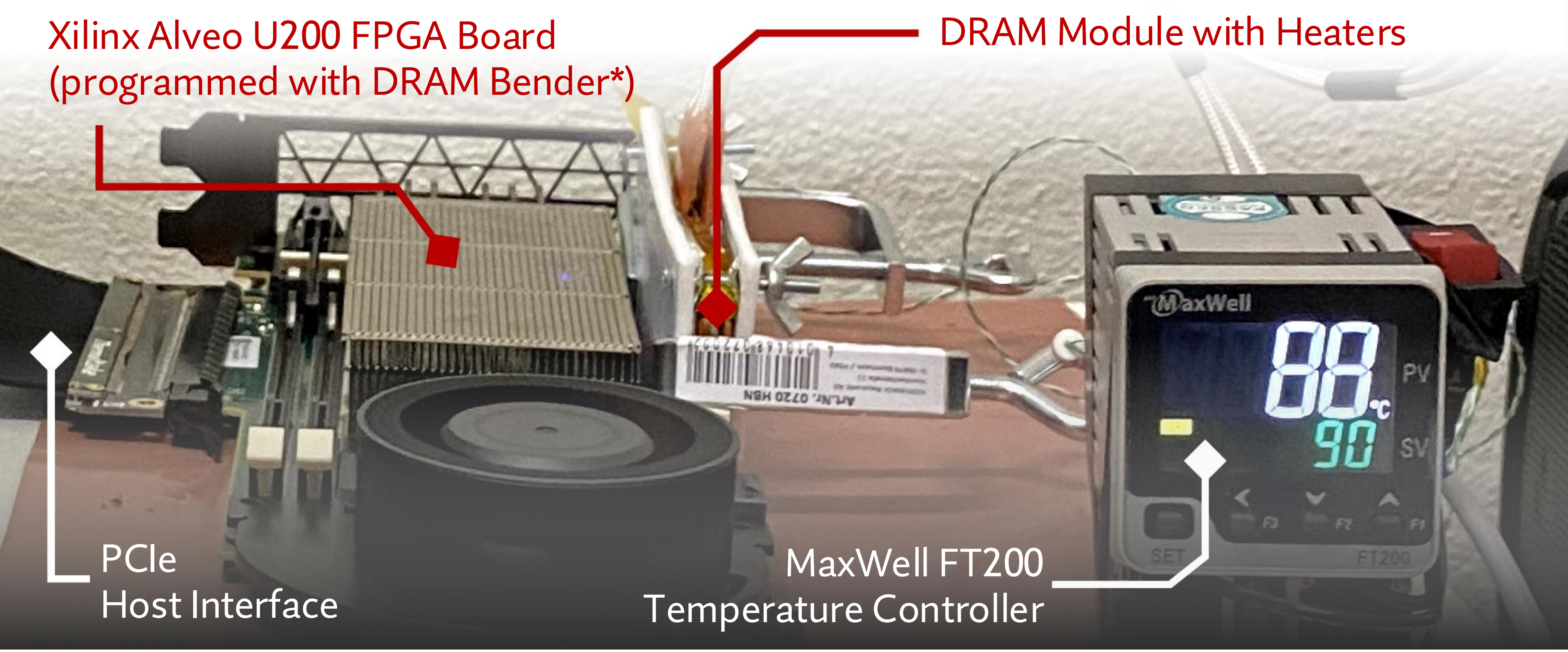
2. How reliably and efficiently can RDT be measured?

Analyze implications for read disturbance solutions

Talk Outline

DDR4 DRAM Testing Infrastructure

DRAM Bender on a Xilinx Virtex UltraScale+ XCU200

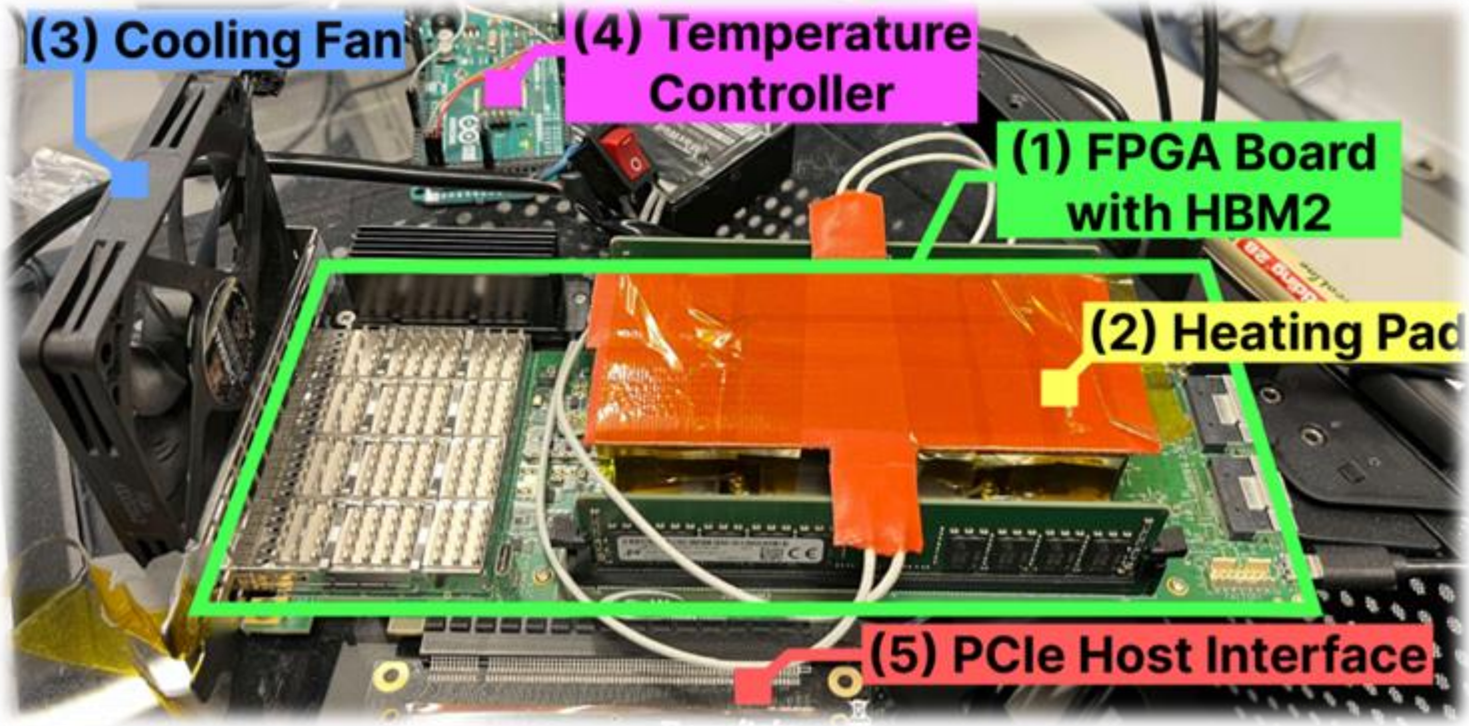


Fine-grained control over DRAM commands,
timing parameters (±1.5ns), and temperature (±0.5°C)

*Olgun et al., **"DRAM Bender: An Extensible and Versatile FPGA-Based Infrastructure to Easily Test State-of-the-Art DRAM Chips,"** TCAD, 2023. [GitHub: https://github.com/CMU-SAFARI/DRAM-Bender]

HBM2 DRAM Testing Infrastructure

DRAM Bender on a Bittware XUPVVH



Fine-grained control over DRAM commands, timing parameters (±1.67ns), and temperature (±0.5°C)

*Olgun et al., "DRAM Bender: An Extensible and Versatile FPGA-Based Infrastructure to Easily Test State-of-the-Art DRAM Chips," TCAD, 2023. [GitHub: https://github.com/CMU-SAFARI/DRAM-Bender]

Tested DRAM Chips

160 DDR4 and 4 HBM2 Chips from SK Hynix, Micron, Samsung

Mfr.	DDR4 Module	# of Chips	Density Die Rev.	Chip Org.	Date (ww-yy)
Mfr. H (SK Hynix)	H0	8	8Gb – J	x8	N/A
	H1	8	16Gb – C	x8	36-21
	H2	8	8Gb – A	x8	43-18
	H3, H4	8	8Gb – D	x8	38-19
	H5, H6	8	8Gb – D	x8	24-20
Mfr. M (Micron)	M0	4	16Gb – E	x16	46-20
	M1	8	16Gb – F	x8	37-22
	M2	8	16Gb – F	x8	37-22
	M3, M4	8	8Gb – R	x8	12-24
	M5	8	8Gb – R	x8	10-24
	M6	8	16Gb – F	x8	12-24
Mfr. S (Samsung)	S0	8	8Gb – C	x8	N/A
	S1	8	8Gb – B	x8	53-20
	S2	8	8Gb – D	x8	10-21
	S3	8	16Gb – A	x8	20-23
	S4	4	4Gb – C	x16	19-19
	S5, S6	8	16Gb – B	x16	15-23
Mfr. S (Samsung)	HBM2 Chip Chip0 – Chip3	4	N/A	N/A	N/A

Testing Methodology

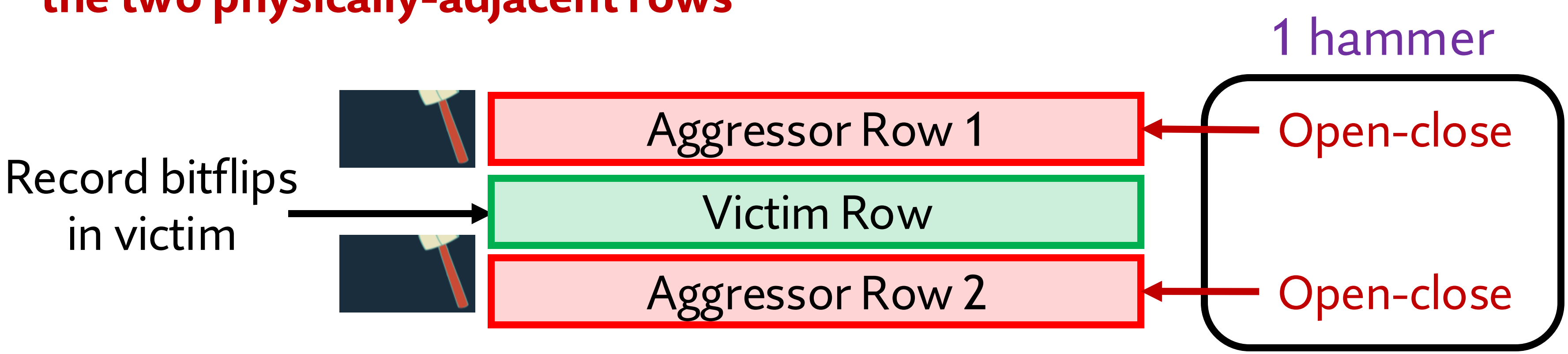
To characterize our DRAM chips at **worst-case** conditions:

1. Prevent sources of interference during core test loop

- **No DRAM refresh**: to avoid refreshing victim row
- **No read disturbance mitigation mechanisms**: to observe circuit-level effects
- **No error correcting codes (ECC)**: to observe all bitflips
- Test for **less than a refresh window (32ms)** to avoid retention failures

2. Worst-case read disturbance access sequence

- We use **worst-case** read disturbance access sequence based on prior works' observations
- Double-sided read disturbance: **repeatedly access the two physically-adjacent rows**



Talk Outline

I. Motivation

II. Experimental Characterization Methodology

III. Foundational Results

IV. In-Depth Analysis of VRD

V. Implications for System Security and Robustness

VI. Conclusion

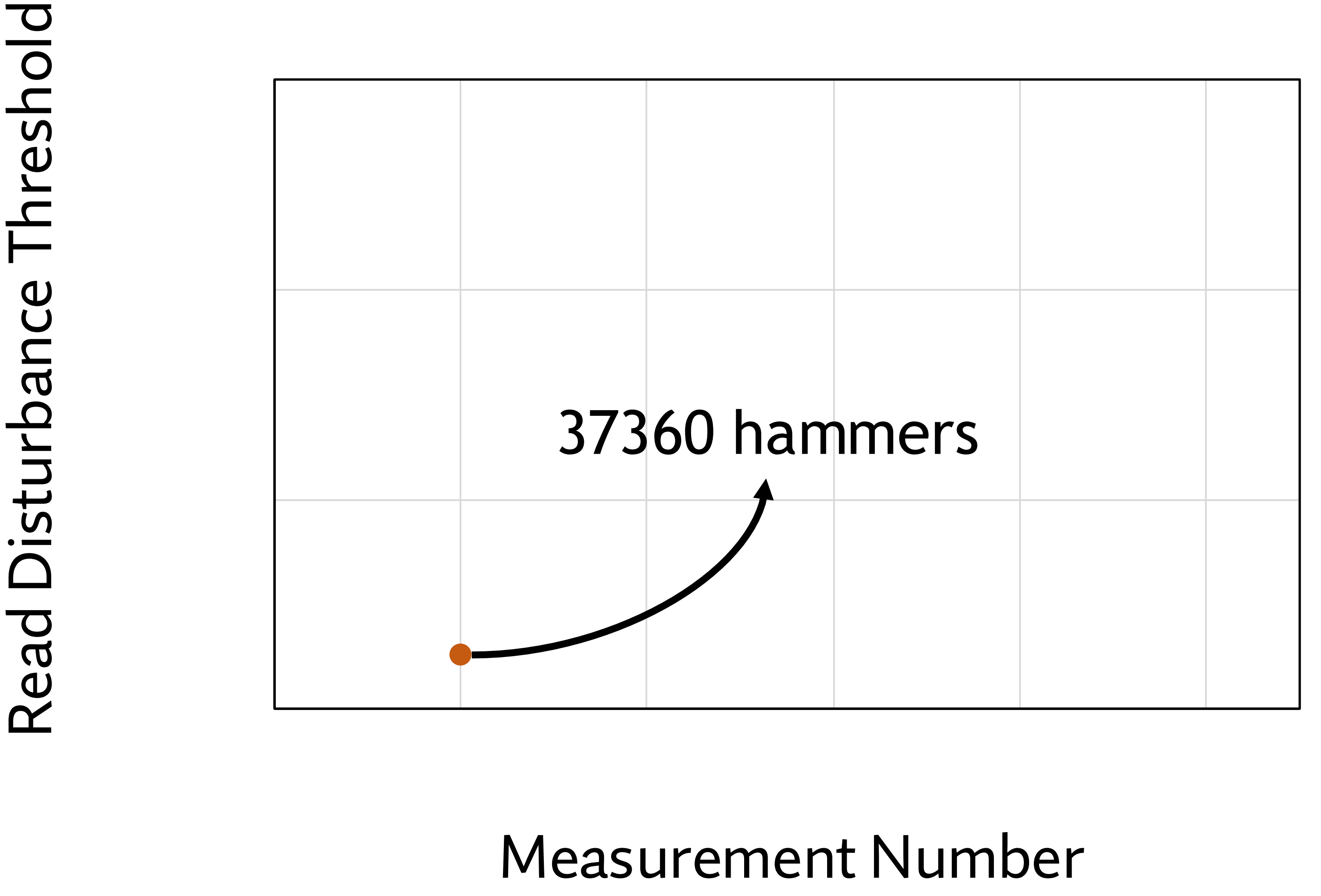
Foundational Results: Key Takeaway

Key Takeaway

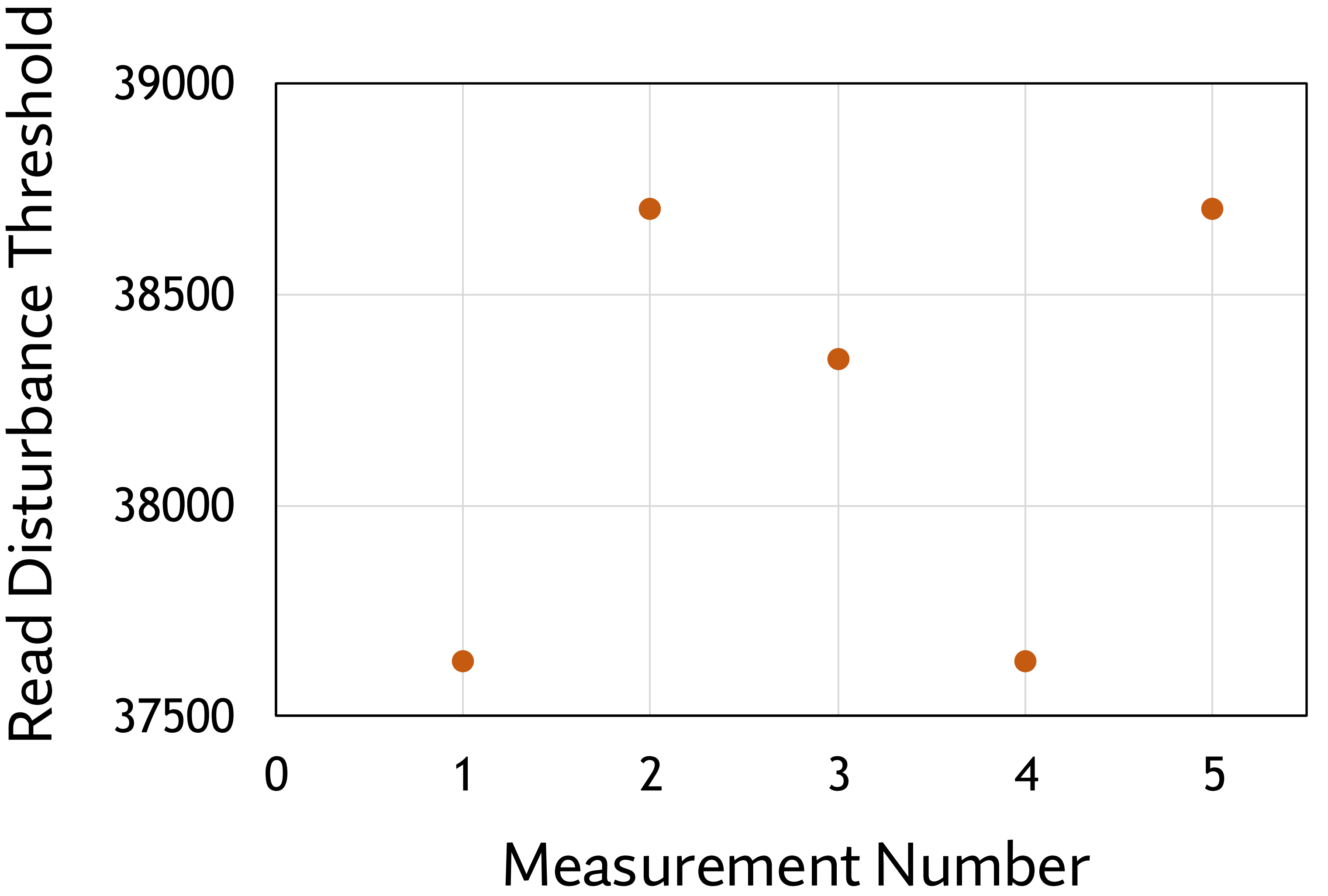
The Read Disturbance Threshold (RDT) of a row changes randomly and unpredictably over time

Accurately identifying RDT is challenging

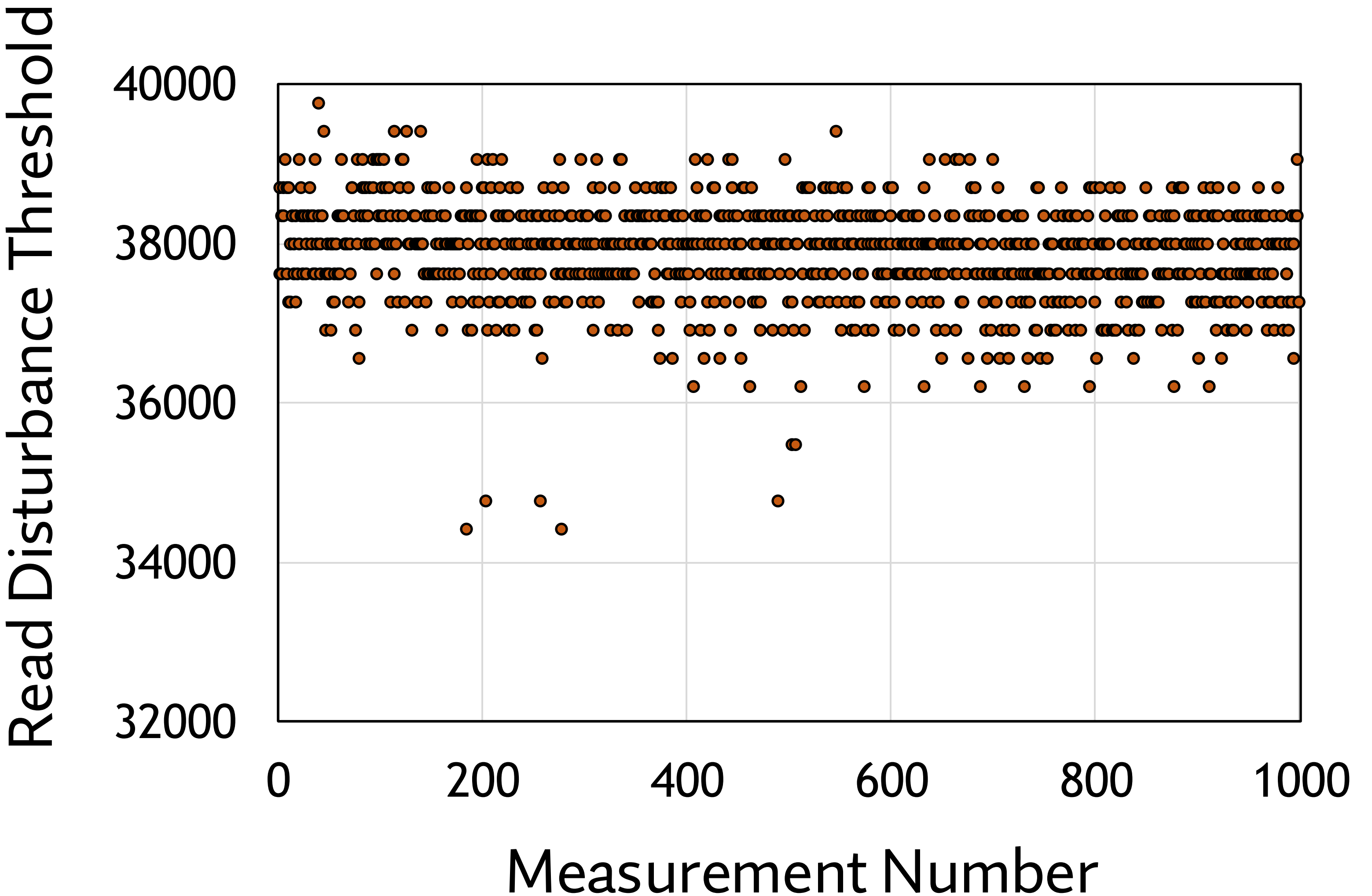
Read Disturbance Threshold Changes Over Time



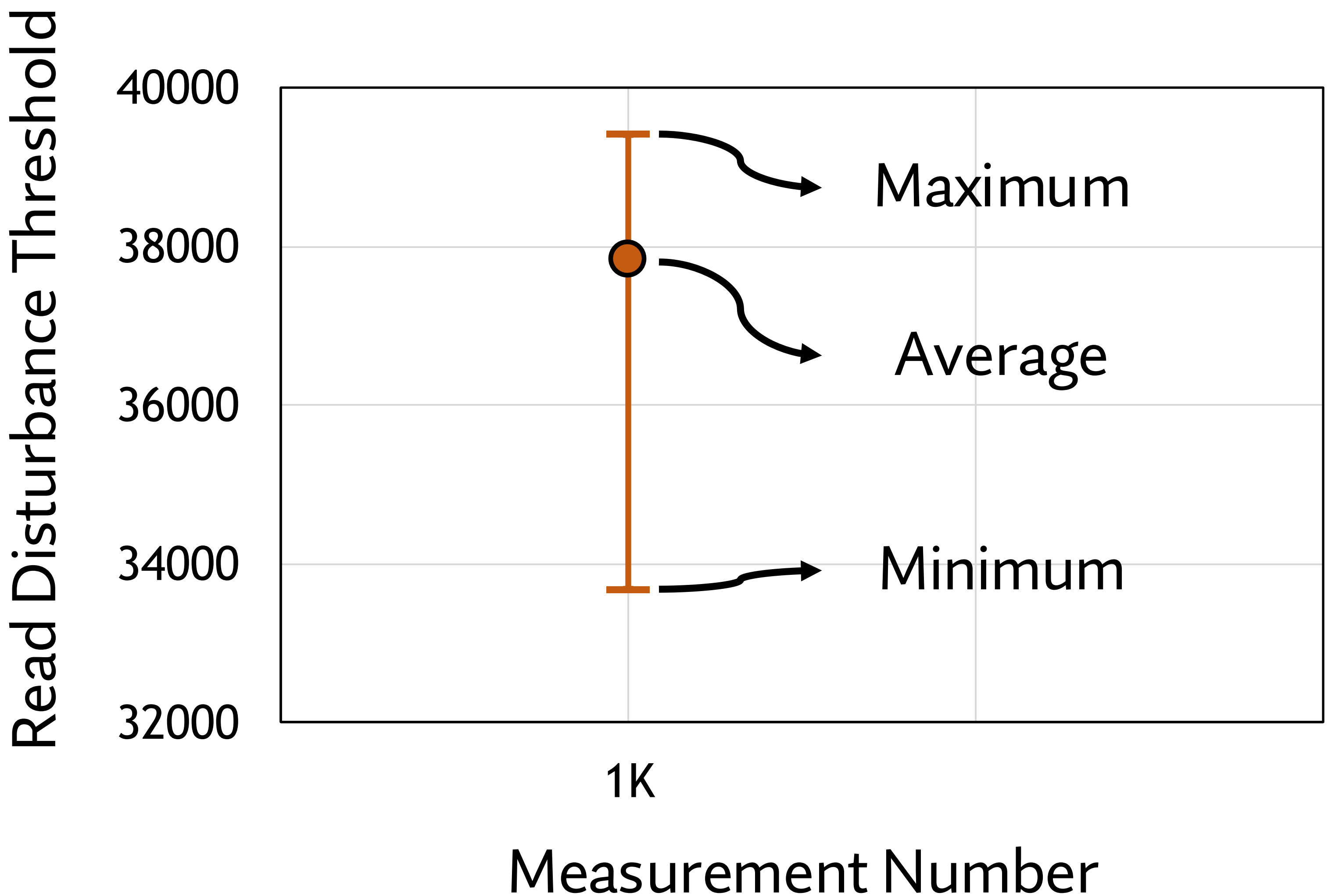
Read Disturbance Threshold Changes Over Time



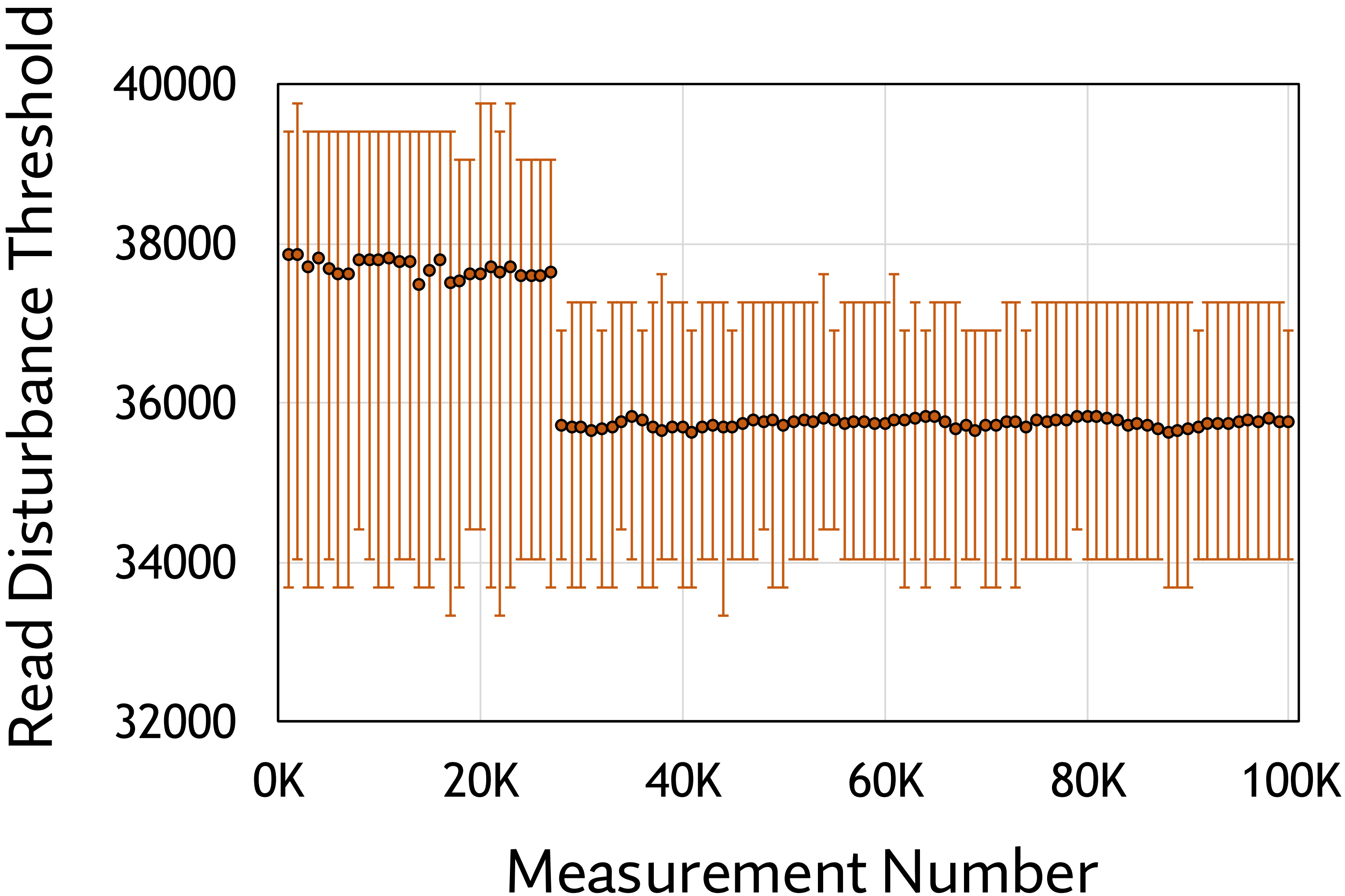
Read Disturbance Threshold Changes Over Time



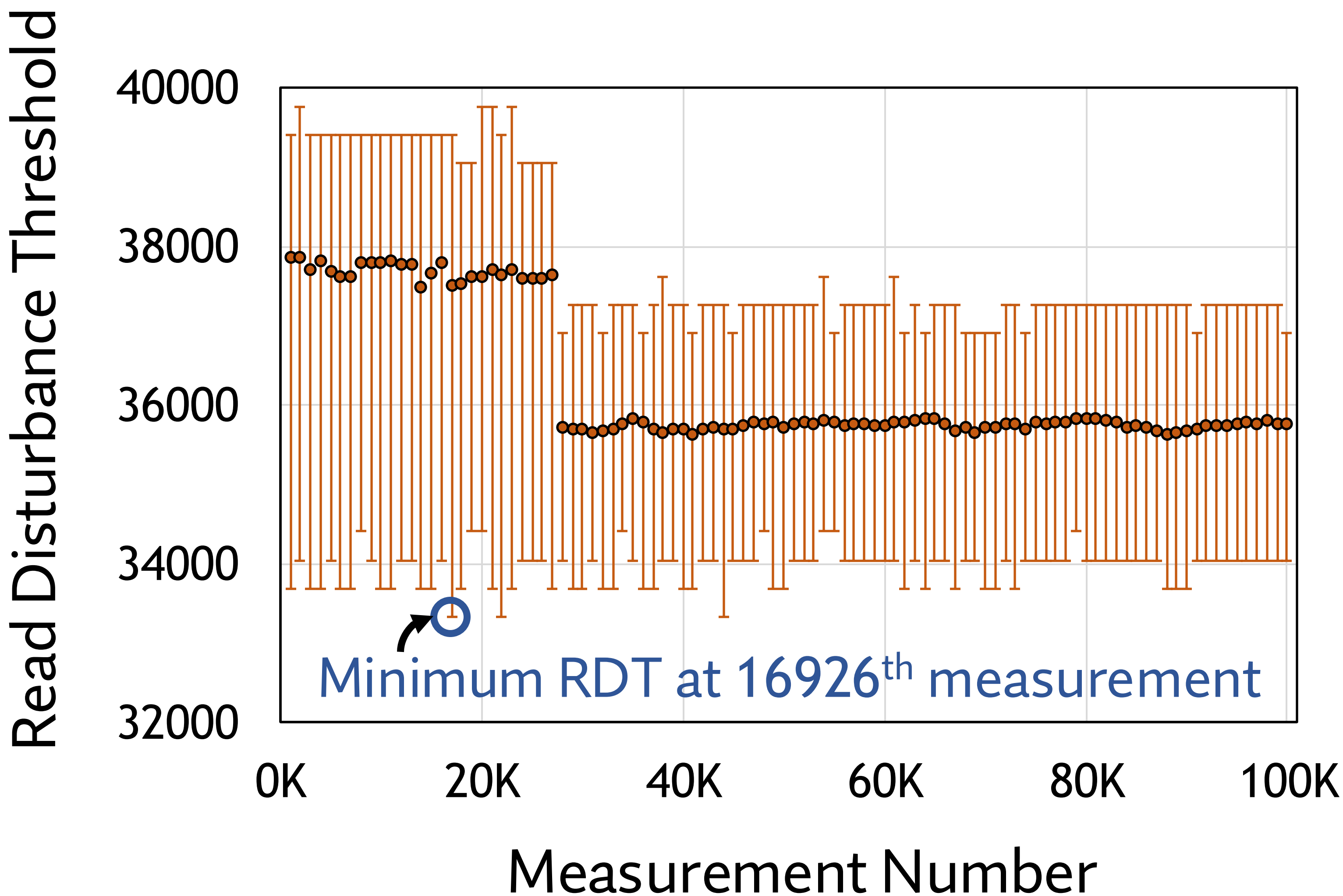
Read Disturbance Threshold Changes Over Time



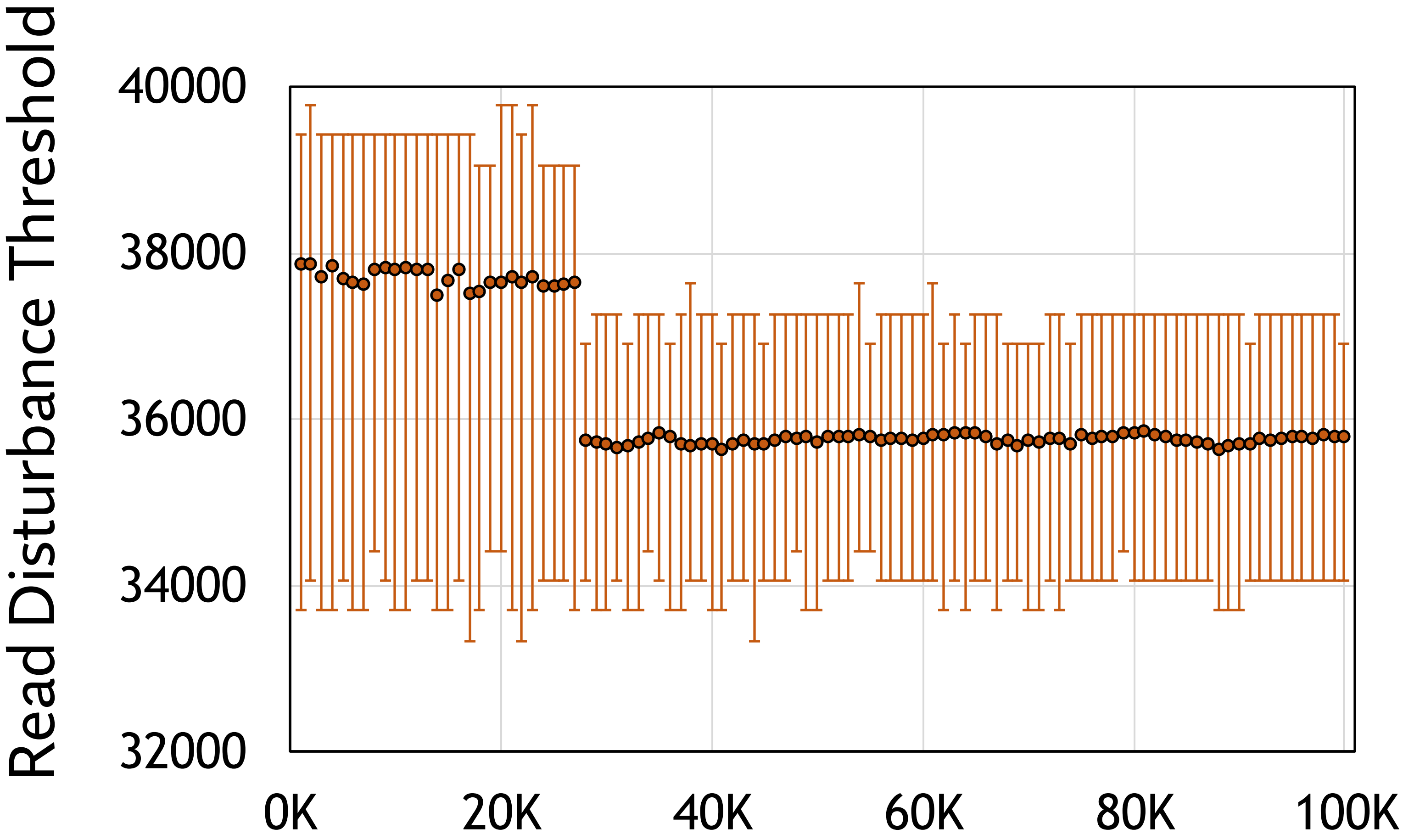
Read Disturbance Threshold Changes Over Time



Read Disturbance Threshold Changes Over Time

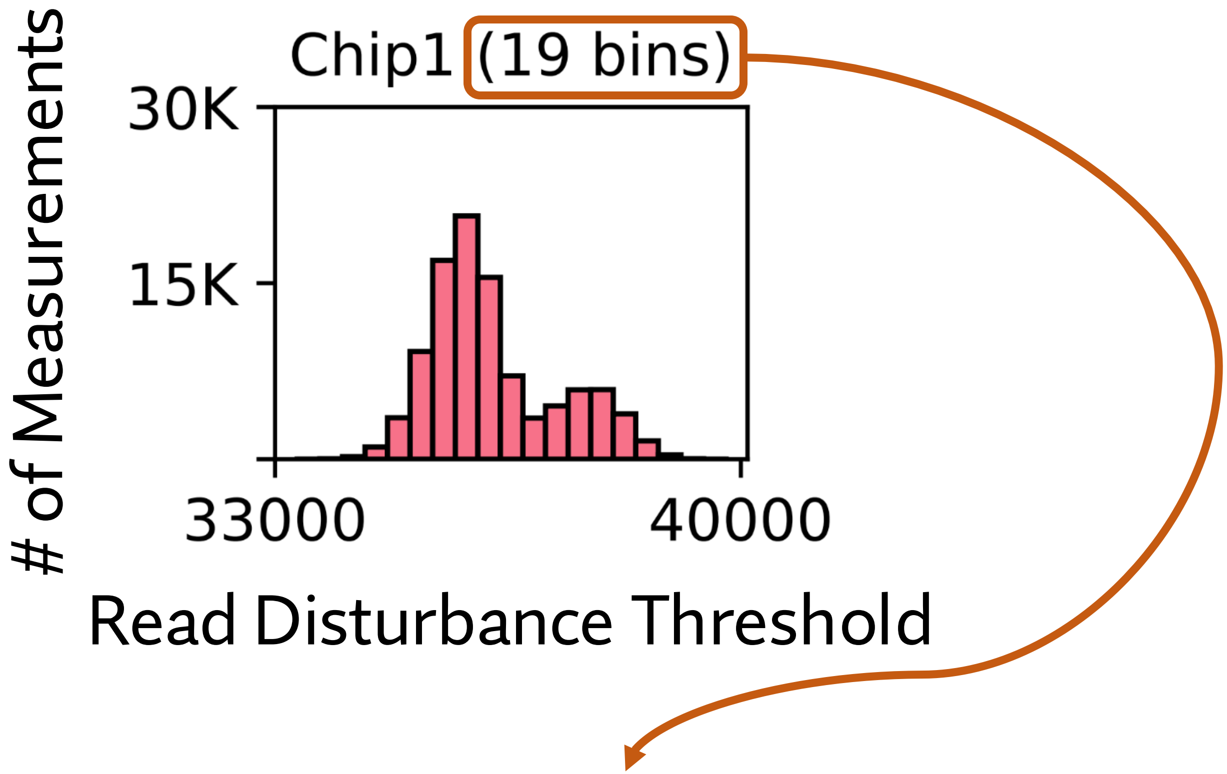


Read Disturbance Threshold Changes Over Time



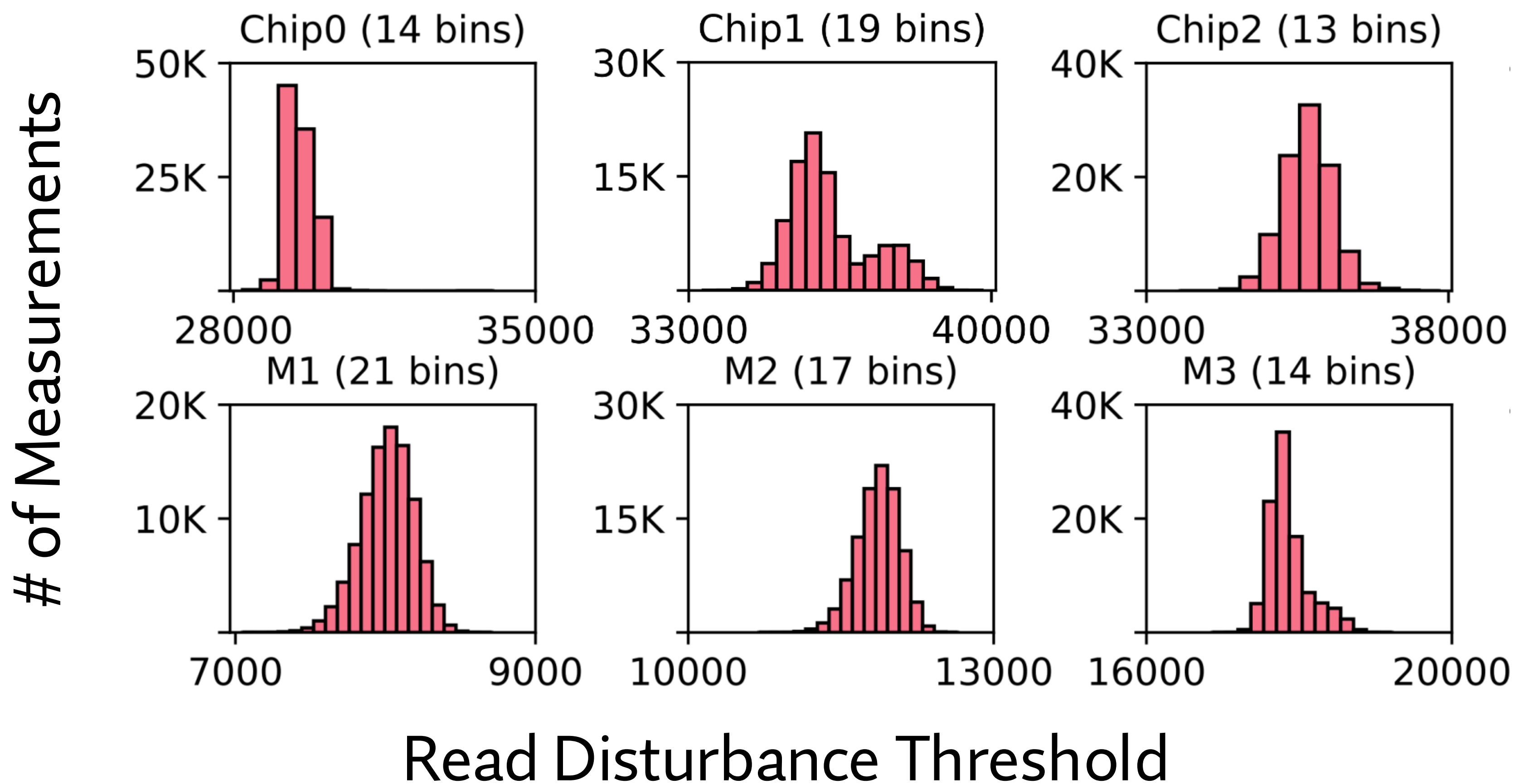
Read Disturbance Threshold of a DRAM row
varies over time: Variable Read Disturbance (VRD)

The RDT of a Row Has Multiple States



The RDT of a row takes various different values across 100,000 measurements

Variable Read Disturbance Across DRAM Chips



RDT **consistently varies** over time
across all tested DRAM **chips**

Variable Read Disturbance Across DRAM Chips

https://arxiv.org/pdf/2502.13075

Variable Read Disturbance:
An Experimental Analysis of Temporal Variation in DRAM Read Disturbance

Ataberk Olgun† F. Nisa Bostancı† İsmail Emir Yüksel† Oğuzhan Canpolat† Haocong Luo†
Geraldo F. Oliveira† A. Giray Yağlıkçı† Minesh Patel‡ Onur Mutlu†
ETH Zurich† *Rutgers University‡*

Modern DRAM chips are subject to read disturbance errors. These errors manifest as security-critical bitflips in a victim DRAM row *that is physically nearby a repeatedly activated (opened) aggressor row (RowHammer) or an aggressor row that is kept open for a long time (RowPress). State-of-the-art read disturbance mitigations rely on accurate and exhaustive characterization of the* read disturbance threshold *(RDT) (e.g., the number of aggressor row activations needed to induce the first RowHammer or RowPress bitflip) of every DRAM row (of which there are millions or billions in a modern system) to prevent read disturbance bitflips securely and with low overhead.*

We experimentally demonstrate for the first time that the RDT of a DRAM row significantly and unpredictably changes over time. We call this new phenomenon variable read disturbance (VRD). Our extensive experiments using 160 DDR4 chips and 4 HBM2 chips from three major manufacturers yield three key observations. First, it is very unlikely that relatively few RDT measurements can accurately identify the RDT of a DRAM row. The minimum RDT of a DRAM row appears after tens of thousands of measurements (e.g., up to 94,467), and the minimum RDT of a DRAM row is 3.5× smaller than the maximum RDT observed for that row. Second, the probability of accu-

row) *many times* (e.g., tens of thousands of times) induces *RowHammer bitflips* in physically nearby rows (i.e., victim rows) [1]. Keeping the aggressor row open for a long period of time amplifies the effects of read disturbance and induces *RowPress bitflips*, *without* requiring *many* repeated aggressor row activations [4].

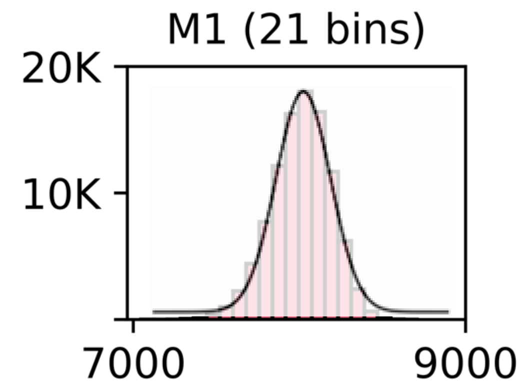
A large body of work [1, 3, 26, 32, 39, 45, 69–141] proposes various techniques to mitigate DRAM read disturbance bitflips. Many high-performance and low-overhead mitigation techniques [1, 73, 74, 76, 79, 82–84, 86, 87, 91, 97, 133–135, 137–139, 142–146], including those that are used and standardized by industry [121, 126, 138, 139, 144], prevent read disturbance bitflips by *preventively* refreshing (i.e., opening and closing) a victim row *before* a bitflip manifests in that row.

To securely prevent read disturbance bitflips at low performance and energy overhead, it is important to *accurately* identify the amount of read disturbance that a victim row can withstand before experiencing a read disturbance bitflip. This amount is typically quantified using the *hammer count (the number of aggressor row activations) needed to induce the first read disturbance bitflip* in a victim row. We call this metric the *read disturbance threshold (RDT)* of the victim row.

VRD is (Likely) Unpredictable

- The outcome of the next read disturbance threshold (RDT) measurement cannot be predicted given past measurements

1 RDT histograms well resemble* random probability distributions e.g., normal distribution



2 Analyze and find no repeating patterns in the series of consecutively measured RDT values using the autocorrelation function

** Resemblance quantified using statistical tests in the paper*

VRD is (Likely) Unpredictable

https://arxiv.org/pdf/2502.13075

Variable Read Disturbance:
An Experimental Analysis of Temporal Variation in DRAM Read Disturbance

Ataberk Olgun† F. Nisa Bostancı† İsmail Emir Yüksel† Oğuzhan Canpolat† Haocong Luo†
Geraldo F. Oliveira† A. Giray Yağlıkçı† Minesh Patel‡ Onur Mutlu†
ETH Zurich† *Rutgers University‡*

Modern DRAM chips are subject to read disturbance errors. These errors manifest as security-critical bitflips in a victim DRAM row *that is physically nearby a repeatedly activated (opened) aggressor row (RowHammer) or an aggressor row that is kept open for a long time (RowPress). State-of-the-art read disturbance mitigations rely on accurate and exhaustive characterization of the* read disturbance threshold *(RDT) (e.g., the number of aggressor row activations needed to induce the first RowHammer or RowPress bitflip) of every DRAM row (of which there are millions or billions in a modern system) to prevent read disturbance bitflips securely and with low overhead.*

We experimentally demonstrate for the first time that the RDT of a DRAM row significantly and unpredictably changes over time. We call this new phenomenon variable read disturbance (VRD). Our extensive experiments using 160 DDR4 chips and 4 HBM2 chips from three major manufacturers yield three key observations. First, it is very unlikely that relatively few RDT measurements can accurately identify the RDT of a DRAM row. The minimum RDT of a DRAM row appears after tens of thousands of measurements (e.g., up to 94,467), and the minimum RDT of a DRAM row is 3.5× smaller than the maximum RDT observed for that row. Second, the probability of accu-

row) *many times* (e.g., tens of thousands of times) induces *RowHammer bitflips* in physically nearby rows (i.e., victim rows) [1]. Keeping the aggressor row open for a long period of time amplifies the effects of read disturbance and induces *RowPress bitflips*, *without* requiring *many* repeated aggressor row activations [4].

A large body of work [1, 3, 26, 32, 39, 45, 69–141] proposes various techniques to mitigate DRAM read disturbance bitflips. Many high-performance and low-overhead mitigation techniques [1, 73, 74, 76, 79, 82–84, 86, 87, 91, 97, 133–135, 137–139, 142–146], including those that are used and standardized by industry [121, 126, 138, 139, 144], prevent read disturbance bitflips by *preventively* refreshing (i.e., opening and closing) a victim row *before* a bitflip manifests in that row.

To securely prevent read disturbance bitflips at low performance and energy overhead, it is important to *accurately* identify the amount of read disturbance that a victim row can withstand before experiencing a read disturbance bitflip. This amount is typically quantified using the *hammer count (the number of aggressor row activations) needed to induce the first read disturbance bitflip* in a victim row. We call this metric the *read disturbance threshold (RDT)* of the victim row.

Talk Outline

In-Depth Analysis: Parameter Space

- Four data patterns

Row Addresses	***Rowstripe0***	***Rowstripe1***	***Checkered0***	***Checkered1***
Victim (V)	0x00	0xFF	0x55	0xAA
Aggressors ($V \pm 1$)	0xFF	0x00	0xAA	0x55
$V \pm [2{:}8]$	0x00	0xFF	0x55	0xAA

- Three temperature levels: 50°C, 65°C, 80°C

- Three aggressor row on time values (RowPress):
  - Minimum t_{RAS} = ~35ns
  - Interval between two periodic refresh commands t_{REFI} = 7.8μs (DDR4)
  - Maximum interval between two refresh $9 \times t_{REFI}$ = 70.2μs (DDR4)

- Test 3750 rows and measure RDT 1000 times per row
  - Aside: what would happen if we measure >1M times?

In-Depth Analysis: Key Takeaways

Takeaway 1

All tested DRAM rows exhibit VRD

Takeaway 2

Relatively few (<500) RDT measurements are unlikely to yield the minimum RDT of a row

Takeaway 3

Data patterns, temperature, and aggressor row on time affect VRD

In-Depth Analysis: Key Takeaways

Takeaway 1

All tested DRAM rows exhibit VRD

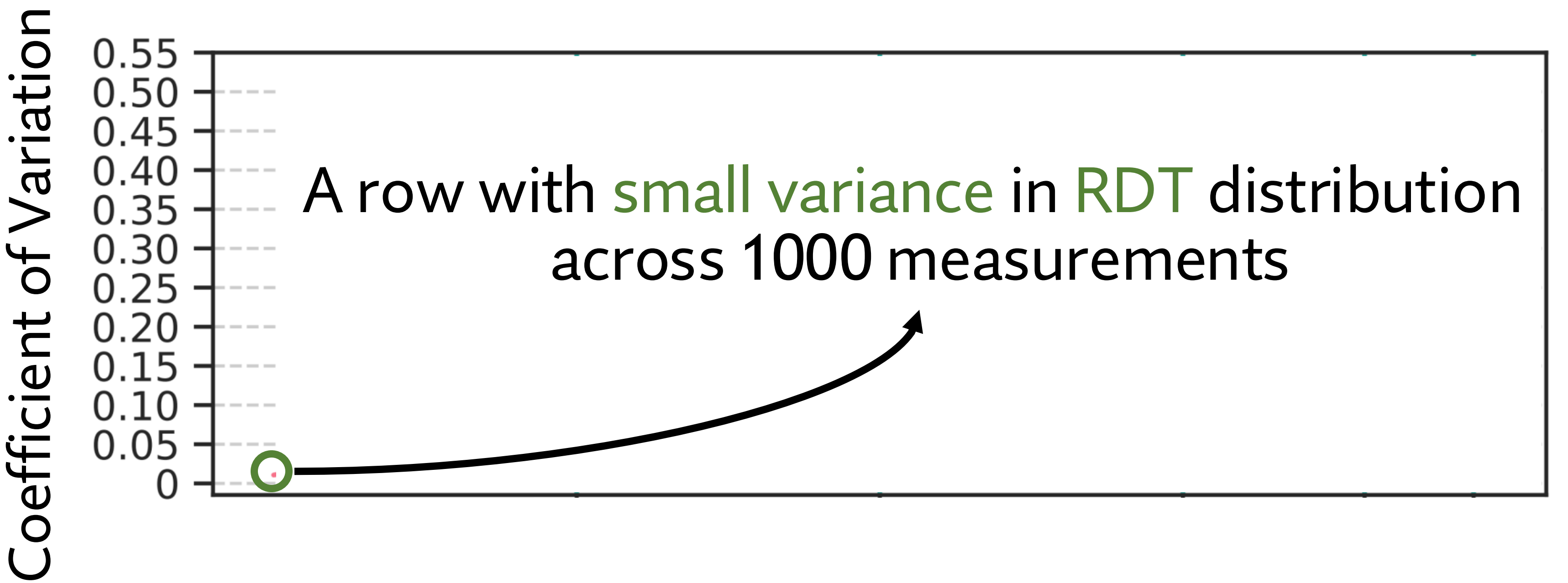
Takeaway 2

Relatively few (<500) RDT measurements are unlikely to yield the minimum RDT of a row

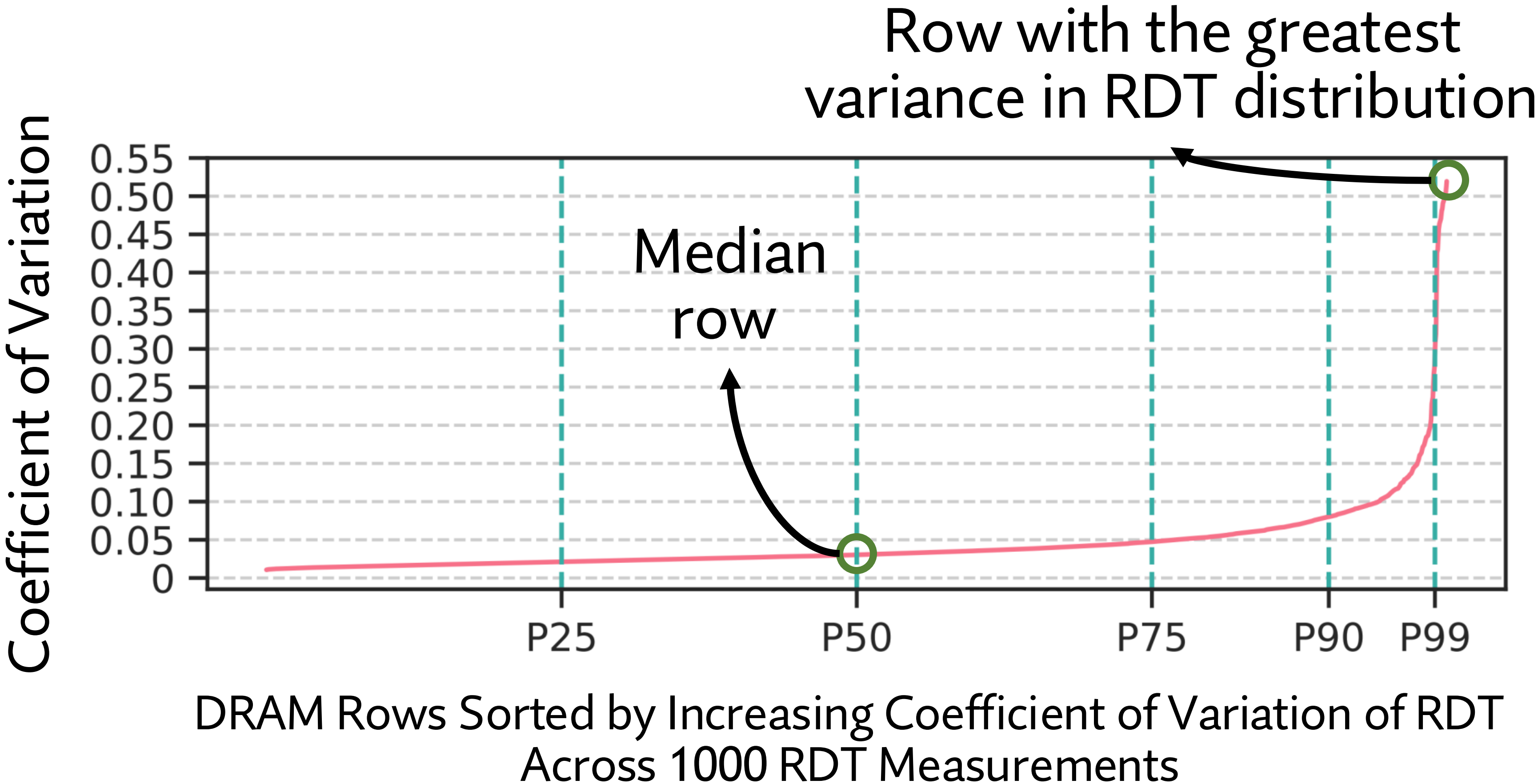
Takeaway 3

Data patterns, temperature, and aggressor row on time affect VRD

VRD Across DRAM Rows

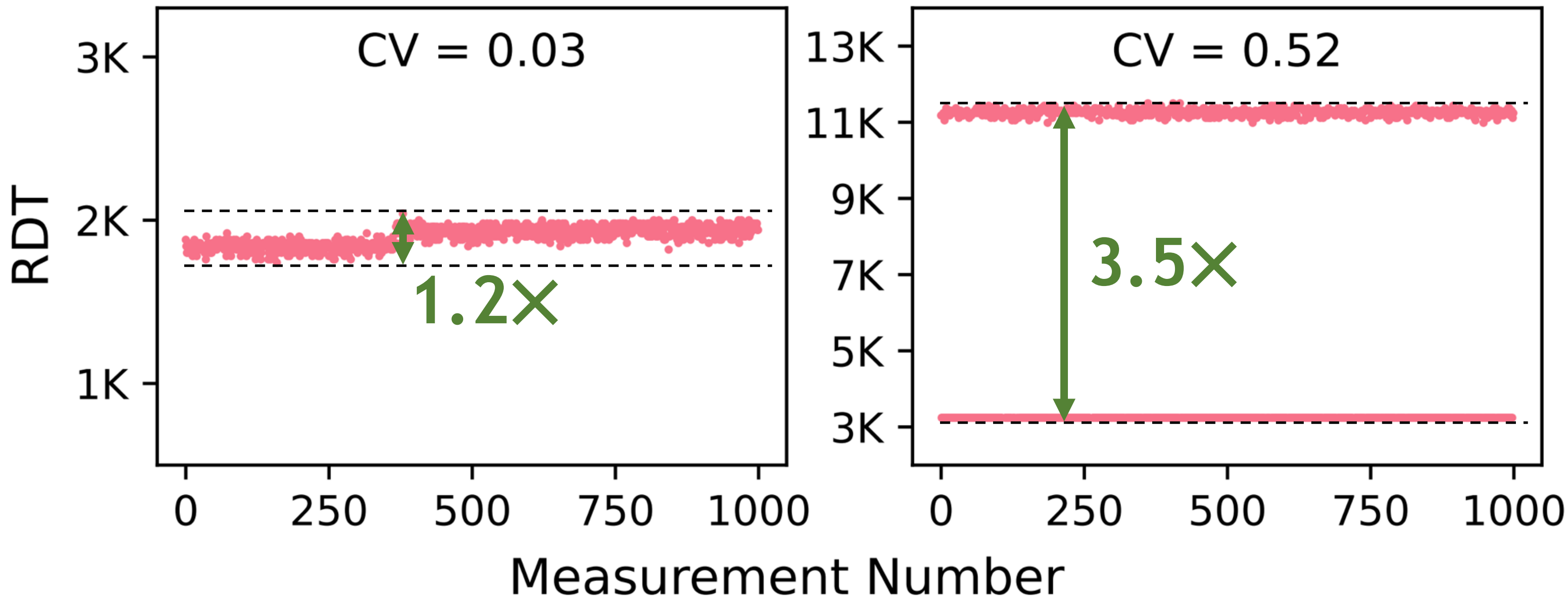


VRD Across DRAM Rows



All tested rows exhibit VRD

VRD in Two Example Rows



Variation in read disturbance threshold
can reach 3.5×

In-Depth Analysis: Key Takeaways

Takeaway 1

All tested DRAM rows exhibit VRD

Takeaway 2

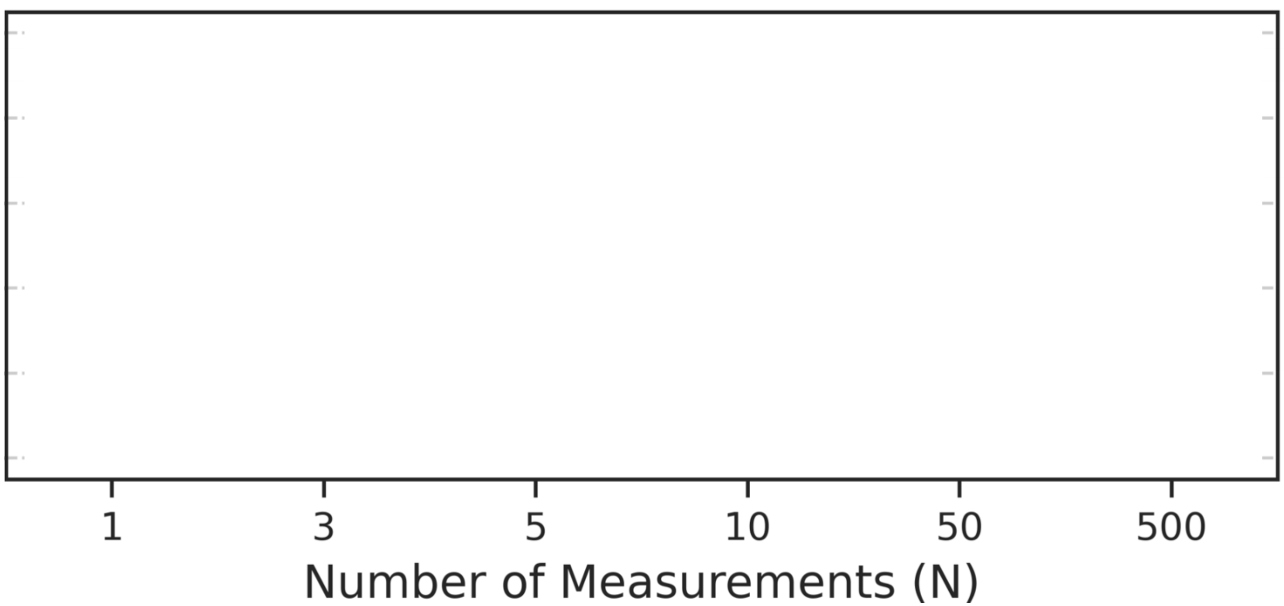
Relatively few (<500) RDT measurements are unlikely to yield the minimum RDT of a row

Takeaway 3

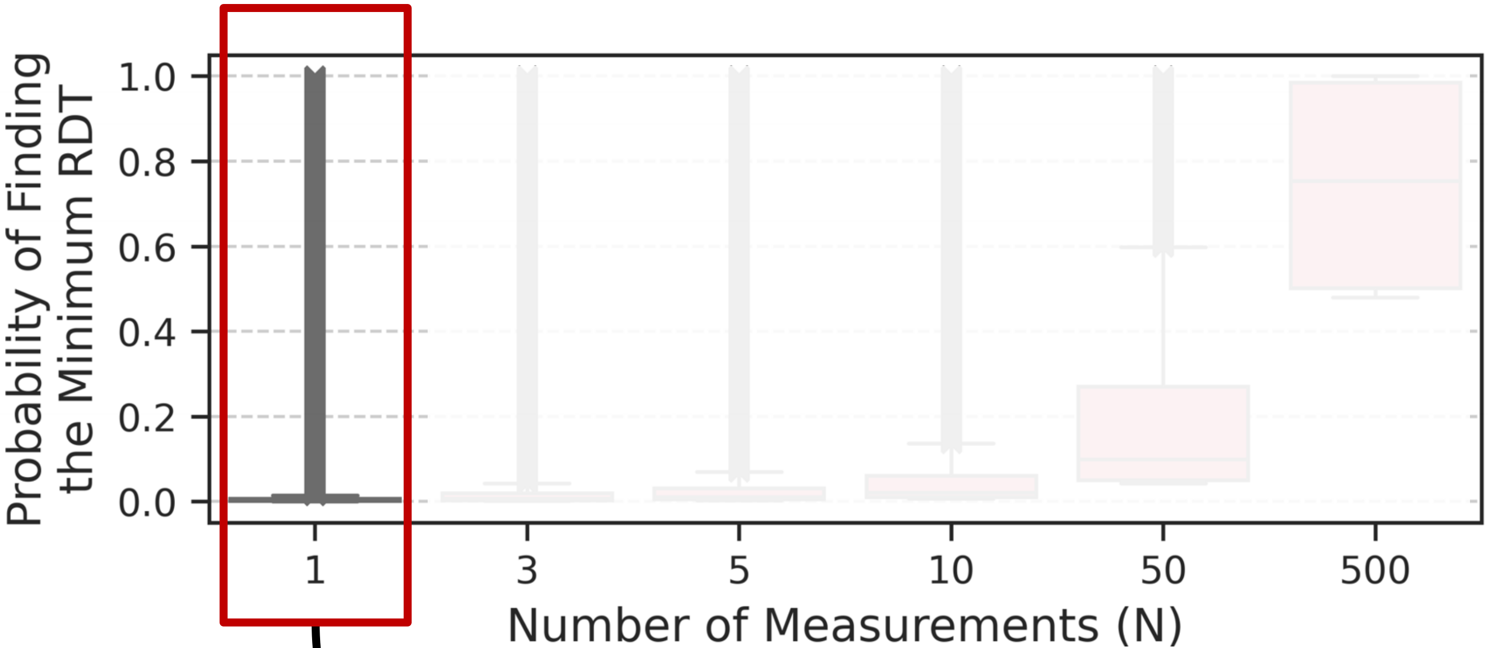
Data patterns, temperature, and aggressor row on time affect VRD

Probability of Identifying the Minimum RDT

- How likely is it that N < 1000 measurements yield the minimum RDT value across 1000 measurements?
- N = 1, 3, 5, 10, 50, and 500
- Monte Carlo simulations for 10K iterations



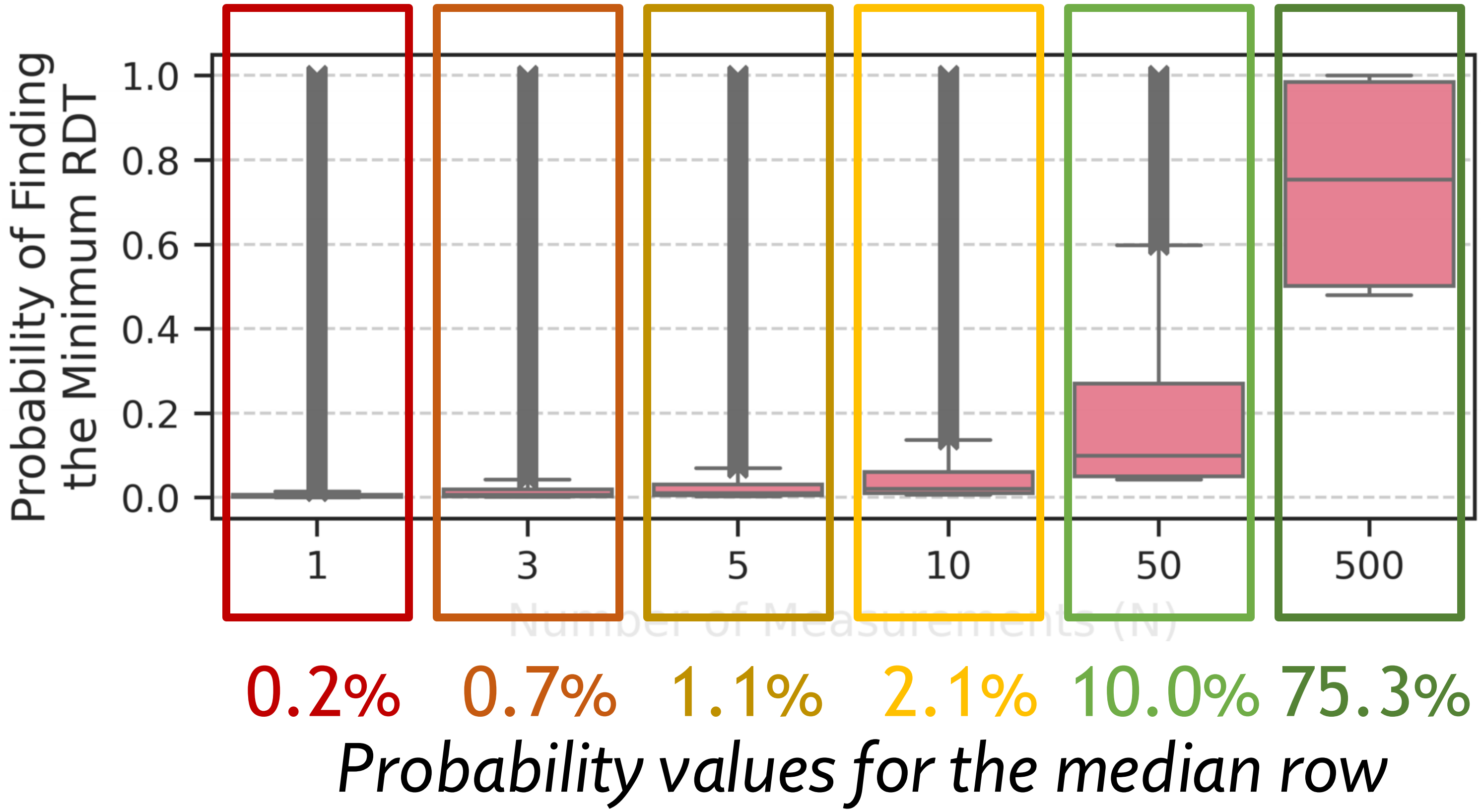
Probability of Identifying the Minimum RDT



only 0.2% for the median row

Very unlikely to find the minimum RDT of a DRAM row with N = 1 measurement

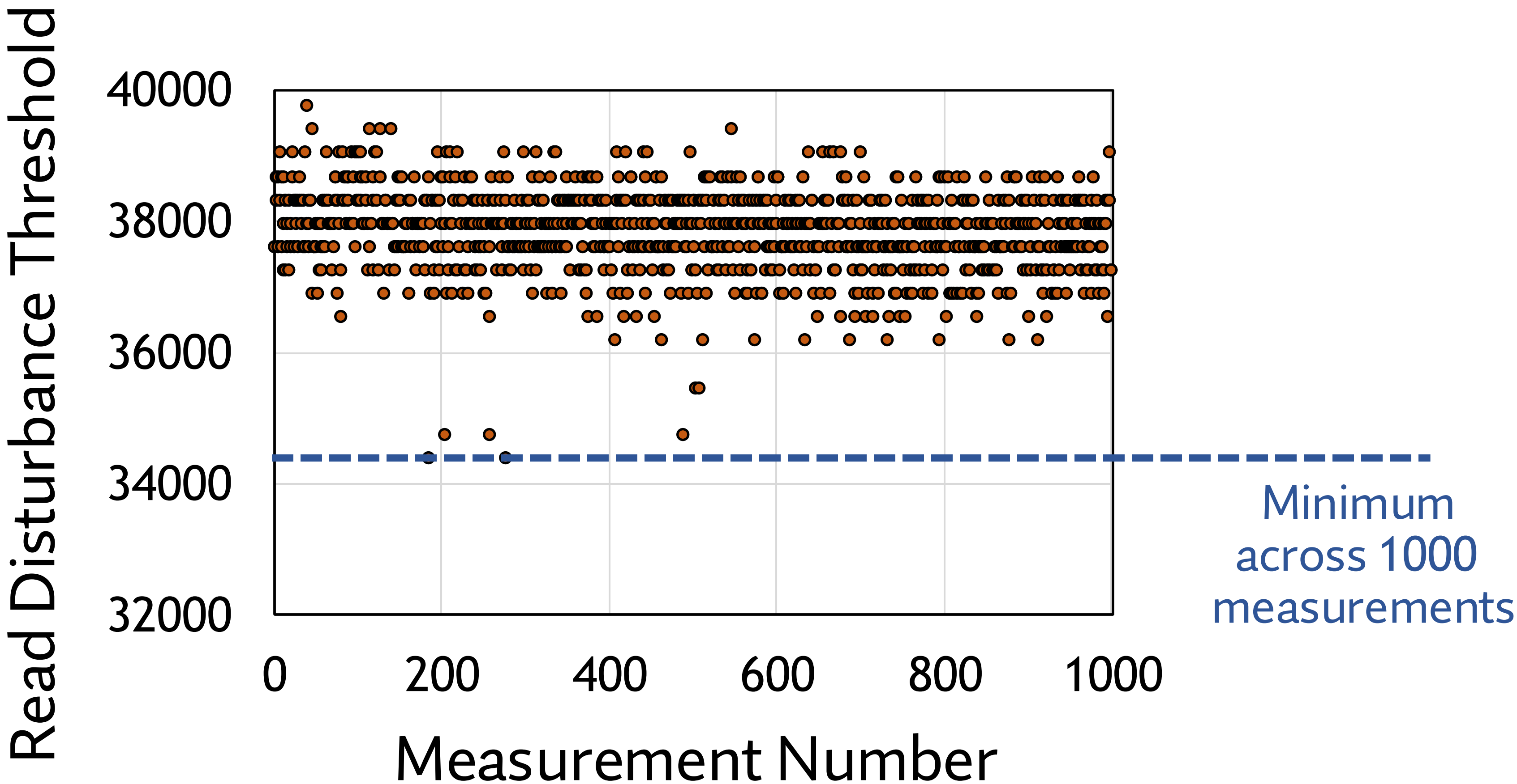
Probability of Identifying the Minimum RDT



Probability of finding the minimum read disturbance threshold increases with N (i.e., with more and more testing)

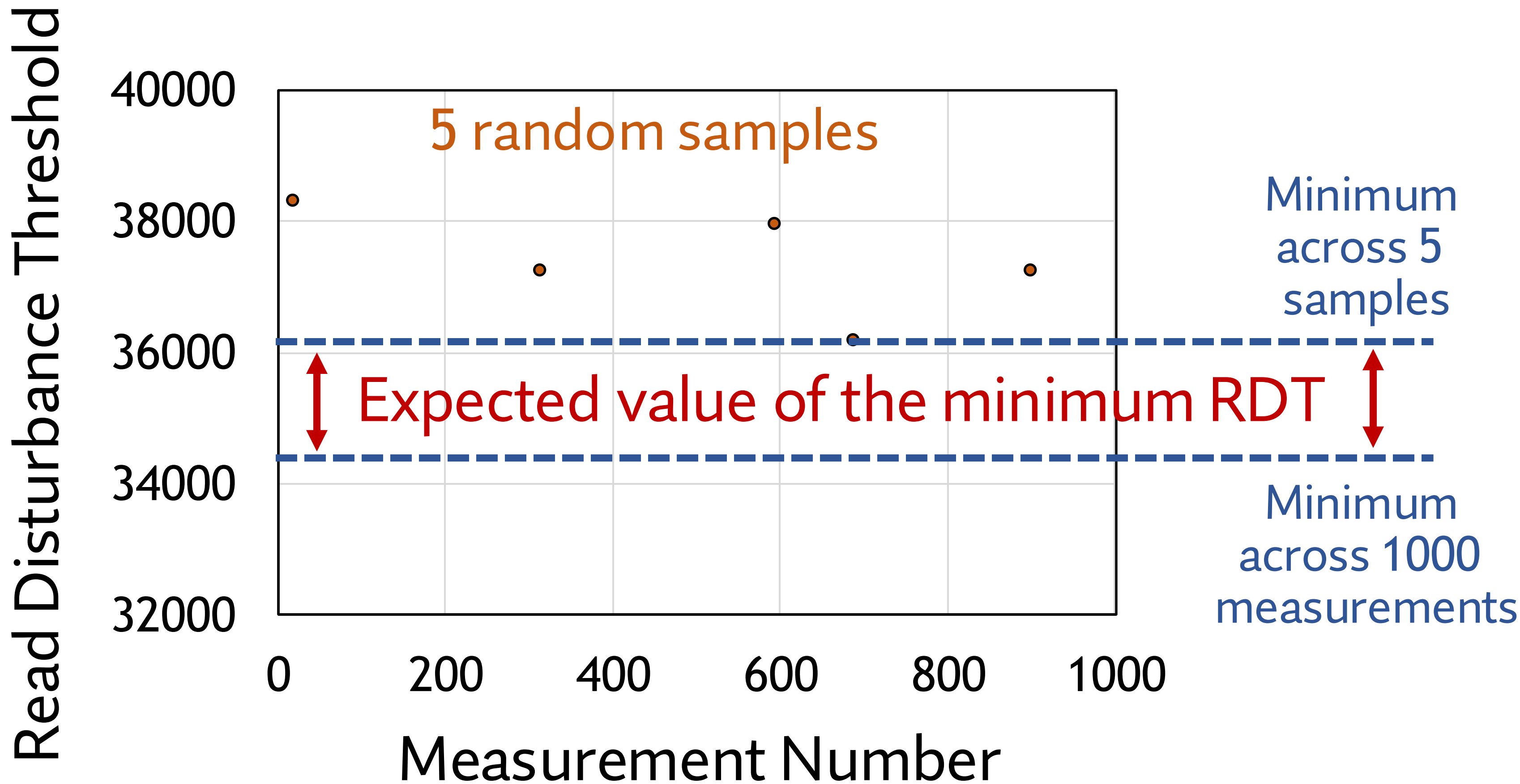
Expected Value of the Minimum RDT

- With only N < 1000 RDT measurements
  how far are we from the minimum RDT across 1000 measurements

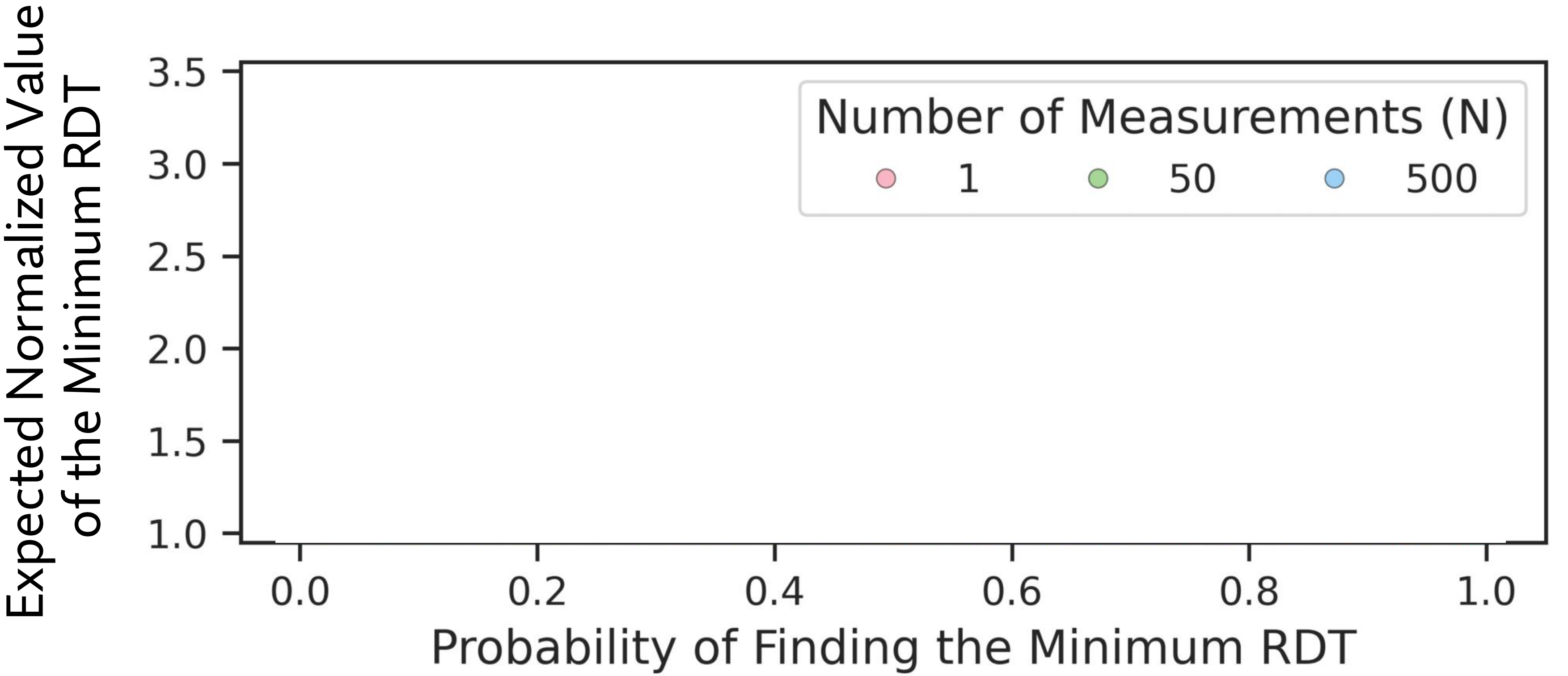


Expected Value of the Minimum RDT

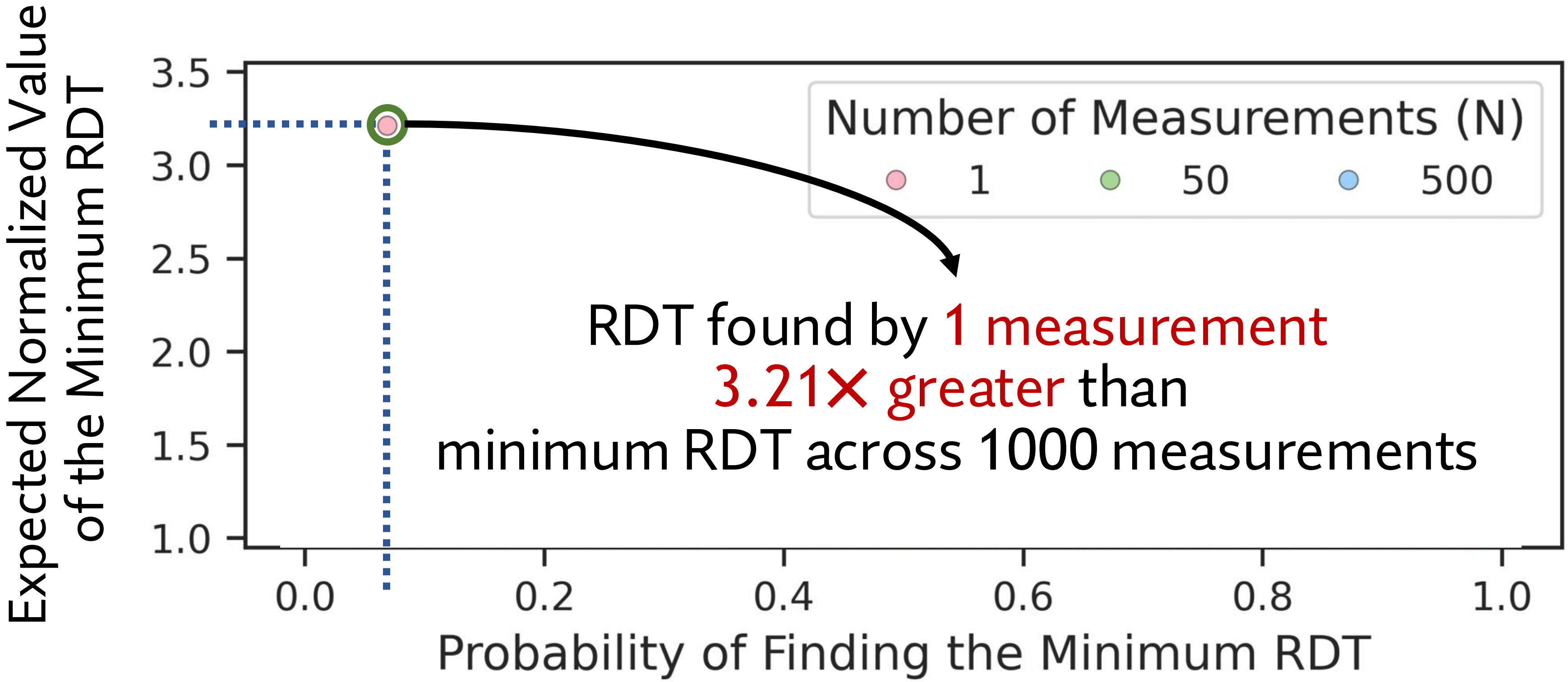
- With only N < 1000 RDT measurements
  how far are we from the minimum RDT across 1000 measurements



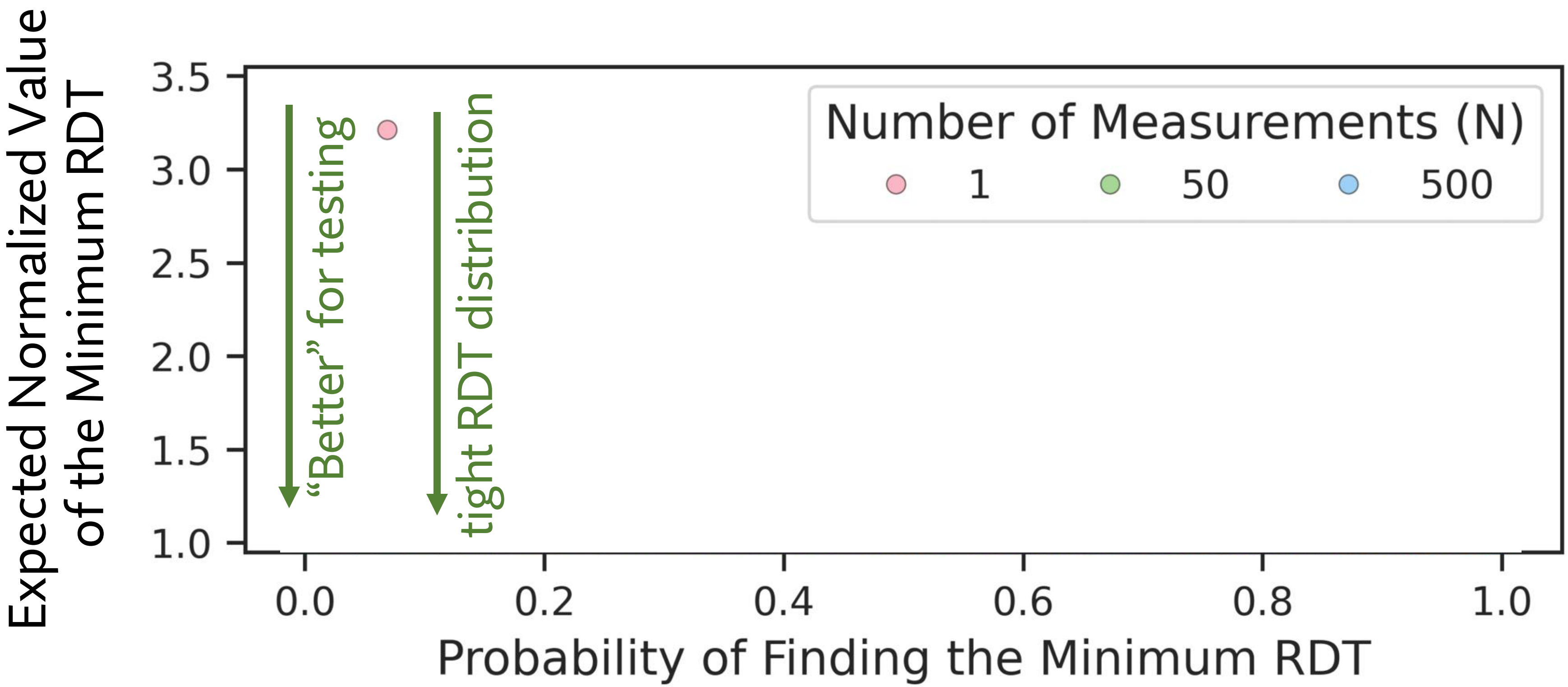
Expected Value of the Minimum RDT



Expected Value of the Minimum RDT



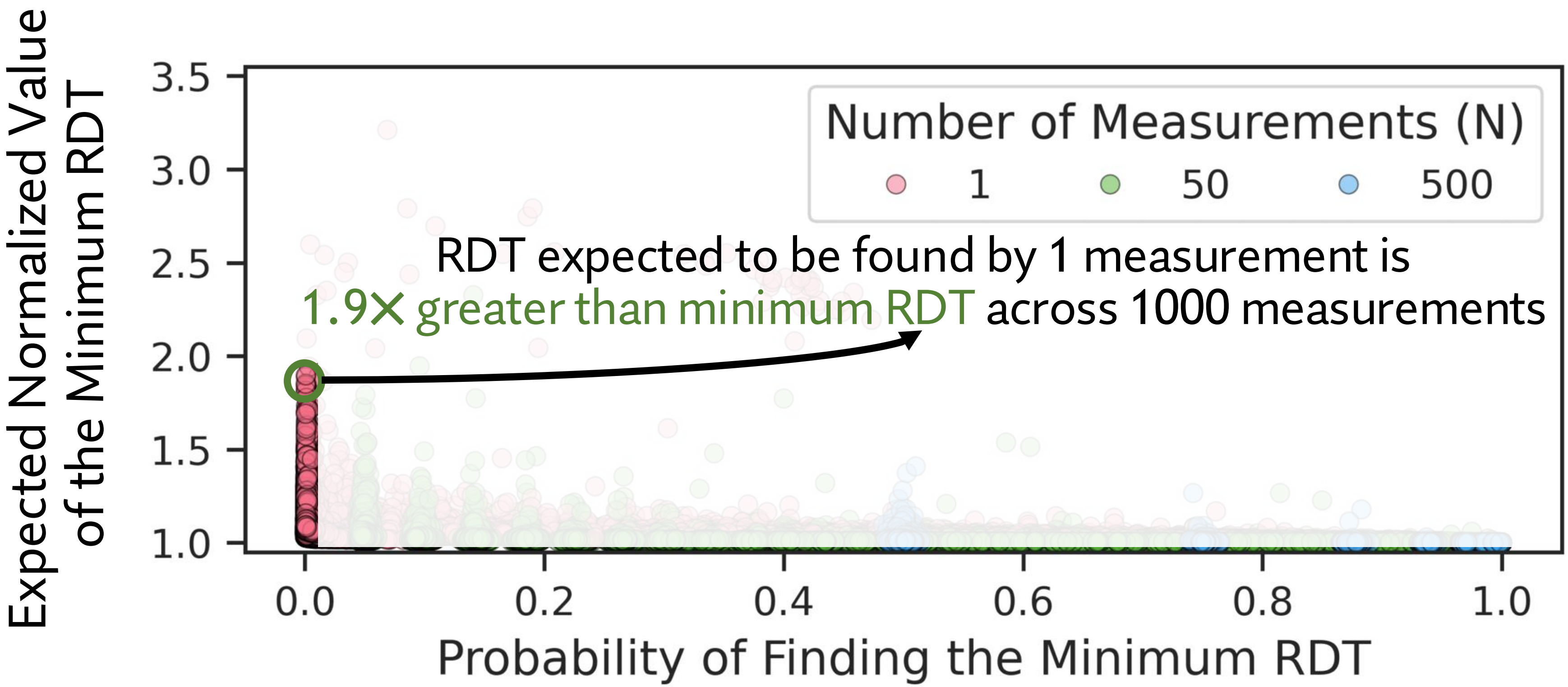
Expected Value of the Minimum RDT



Plot interpretation

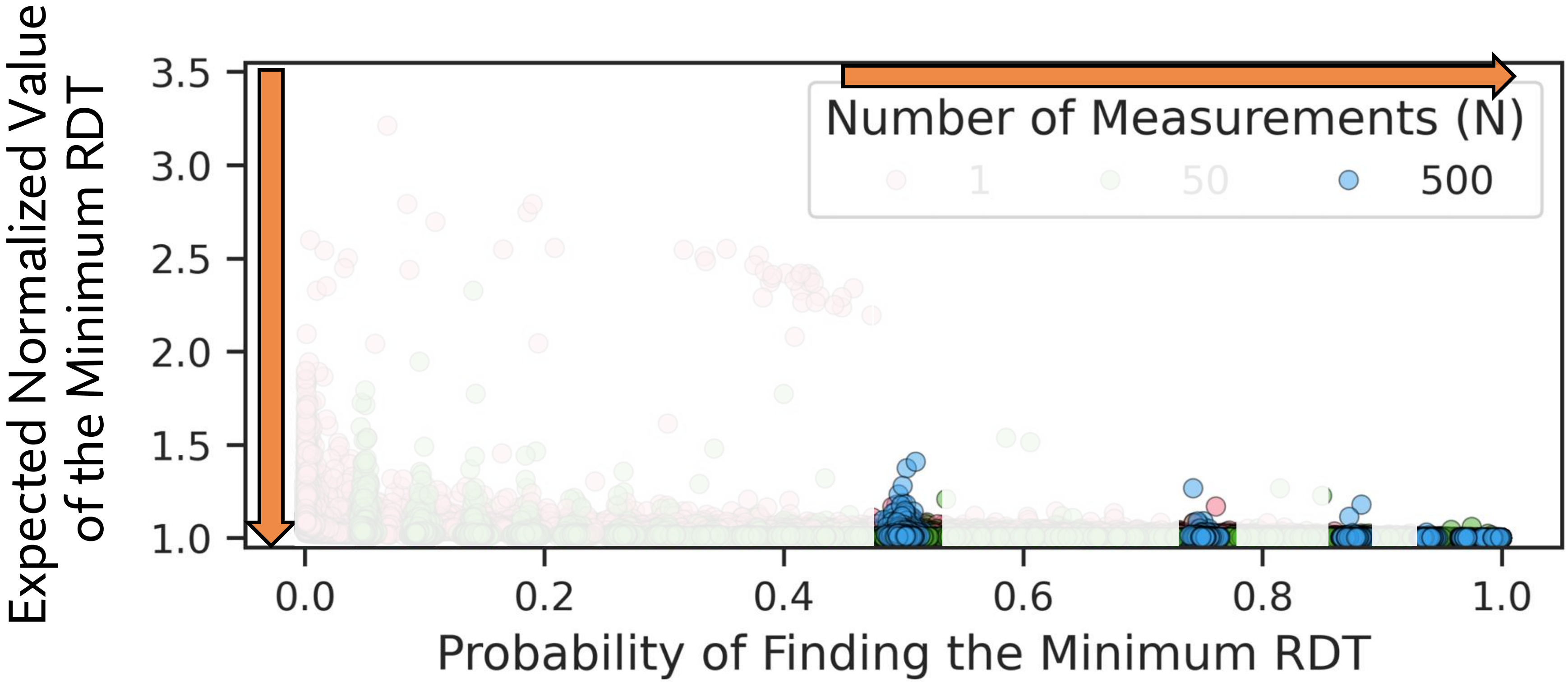
"Better" for testing == as tight an RDT distribution as possible

Expected Value of the Minimum RDT



The minimum RDT is significantly smaller than the one expected to be found with N = 1 measurement

Expected Value of the Minimum RDT



With increasing N (number of measurements) we expect to identify an RDT value closer to the minimum across 1000 measurements

In-Depth Analysis: Key Takeaways

Takeaway 1

All tested DRAM rows exhibit VRD

Takeaway 2

Relatively few (<500) RDT measurements are unlikely to yield the minimum RDT of a row

Takeaway 3

Data patterns, temperature, and aggressor row on time affect VRD

Talk Outline

I. Motivation

II. Experimental Characterization Methodology

III. Foundational Results

IV. In-Depth Analysis of VRD

V. Implications for System Security and Robustness

VI. Conclusion

Implications Summary

- Security guarantees provided by mitigation techniques rely on accurately identified minimum read disturbance threshold (RDT)
- Accurate identification of minimum RDT (for each row) is extremely challenging (even with 1000s measurements) because RDT unpredictably changes over time
- We analyze the use of a guardband and ECC
  - May prevent VRD-induced bitflips
  - Large guardbands induce performance overhead
- Call for future work on online RDT profiling and runtime configurable read disturbance mitigations

Important Caveat

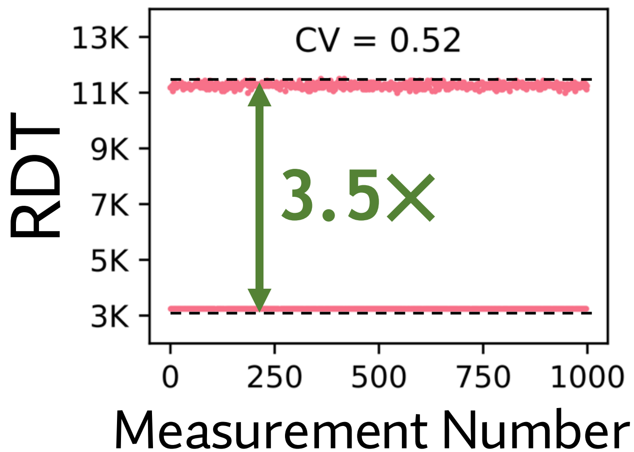
- VRD solution analysis based on 1K or 10K read disturbance threshold measurements per row

- More measurements could yield **worse** results
  - Read disturbance threshold distribution tail could expand

- What results would millions or billions of RDT measurements yield?

Challenges of Accurately Identifying RDT

Variation in read disturbance threshold across 1000 measurements can reach 3.5× and may not be bounded

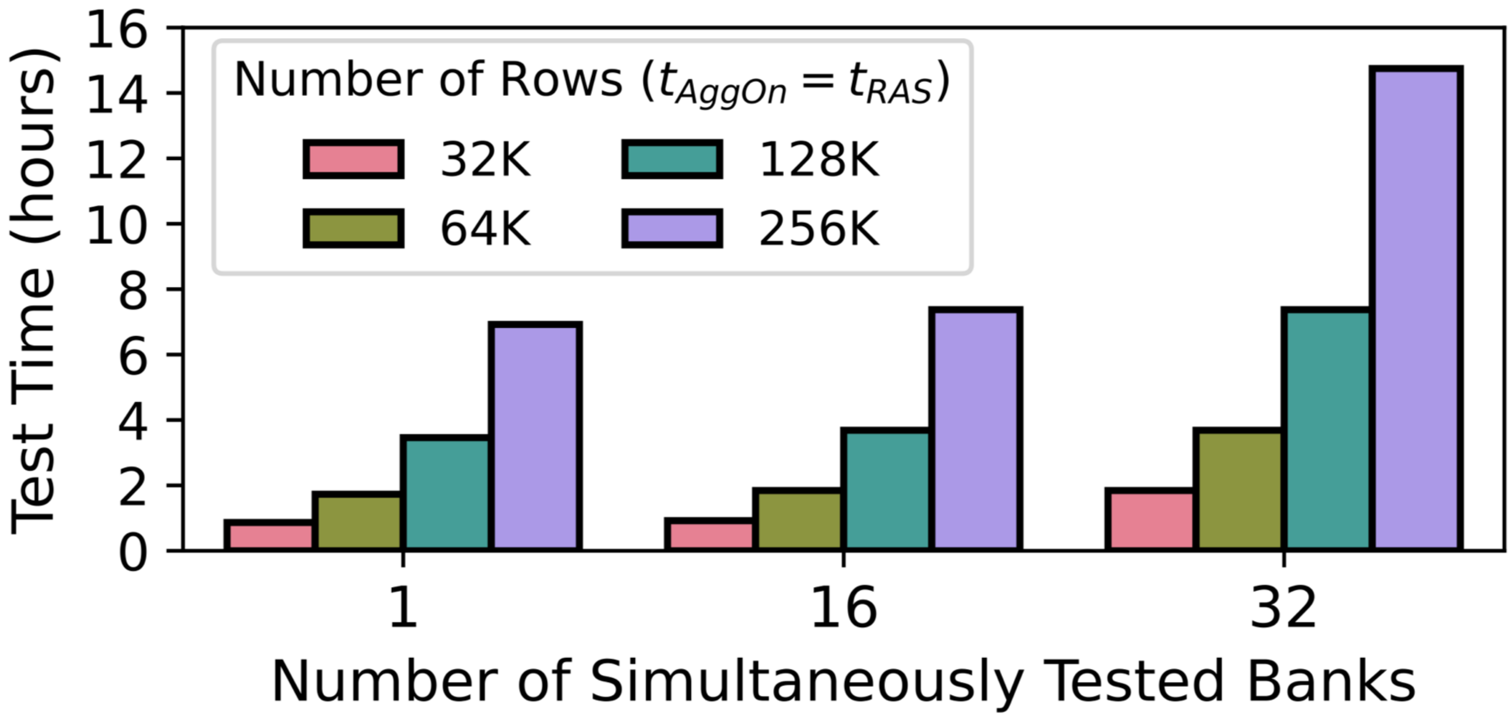


VRD is affected by data pattern, temperature, aggressor row on time

Comprehensive RDT profiling is **time-intensive**

Measuring RDT of *each row only once* with 8000 hammers using four data patterns, at three temperature levels takes 39 minutes in a bank of 256K rows

RDT Profiling is Time-Intensive



Comprehensive RDT testing can take tens of hours
(***only*** 1000 measurements, one data pattern, one temperature level, one aggressor row on time)

RDT Profiling is Time-Intensive

https://arxiv.org/pdf/2502.13075

Variable Read Disturbance:
An Experimental Analysis of Temporal Variation in DRAM Read Disturbance

Ataberk Olgun† F. Nisa Bostancı† İsmail Emir Yüksel† Oğuzhan Canpolat† Haocong Luo†
Geraldo F. Oliveira† A. Giray Yağlıkçı† Minesh Patel‡ Onur Mutlu†

ETH Zurich† *Rutgers University‡*

Modern DRAM chips are subject to read disturbance errors. These errors manifest as security-critical bitflips in a victim DRAM row *that is physically nearby a repeatedly activated (opened) aggressor row (RowHammer) or an aggressor row that is kept open for a long time (RowPress). State-of-the-art read disturbance mitigations rely on accurate and exhaustive characterization of the* read disturbance threshold *(RDT) (e.g., the number of aggressor row activations needed to induce the first RowHammer or RowPress bitflip) of every DRAM row (of which there are millions or billions in a modern system) to prevent read disturbance bitflips securely and with low overhead.*

We experimentally demonstrate for the first time that the RDT of a DRAM row significantly and unpredictably changes over time. We call this new phenomenon variable read disturbance (VRD). Our extensive experiments using 160 DDR4 chips and 4 HBM2 chips from three major manufacturers yield three key observations. First, it is very unlikely that relatively few RDT measurements can accurately identify the RDT of a DRAM row. The minimum RDT of a DRAM row appears after tens of thousands of measurements (e.g., up to 94,467), and the minimum RDT of a DRAM row is 3.5× smaller than the maximum RDT observed for that row. Second, the probability of accu-

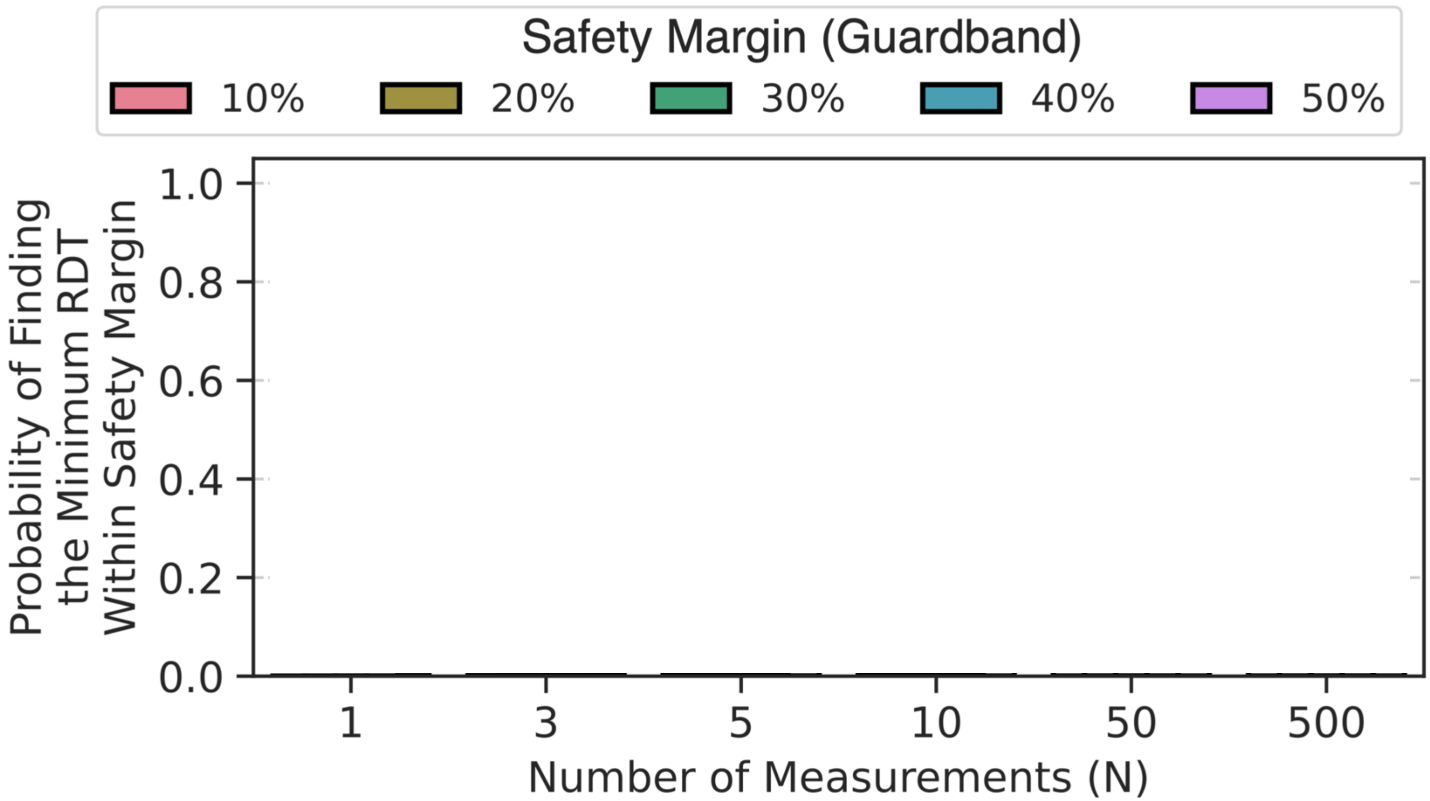
row) *many times* (e.g., tens of thousands of times) induces *RowHammer bitflips* in physically nearby rows (i.e., victim rows) [1]. Keeping the aggressor row open for a long period of time amplifies the effects of read disturbance and induces *RowPress bitflips*, *without* requiring *many* repeated aggressor row activations [4].

A large body of work [1, 3, 26, 32, 39, 45, 69–141] proposes various techniques to mitigate DRAM read disturbance bitflips. Many high-performance and low-overhead mitigation techniques [1, 73, 74, 76, 79, 82–84, 86, 87, 91, 97, 133–135, 137–139, 142–146], including those that are used and standardized by industry [121, 126, 138, 139, 144], prevent read disturbance bitflips by *preventively* refreshing (i.e., opening and closing) a victim row *before* a bitflip manifests in that row.

To securely prevent read disturbance bitflips at low performance and energy overhead, it is important to *accurately* identify the amount of read disturbance that a victim row can withstand before experiencing a read disturbance bitflip. This amount is typically quantified using the *hammer count (the number of aggressor row activations) needed to induce the first read disturbance bitflip* in a victim row. We call this metric the *read disturbance threshold (RDT)* of the victim row.

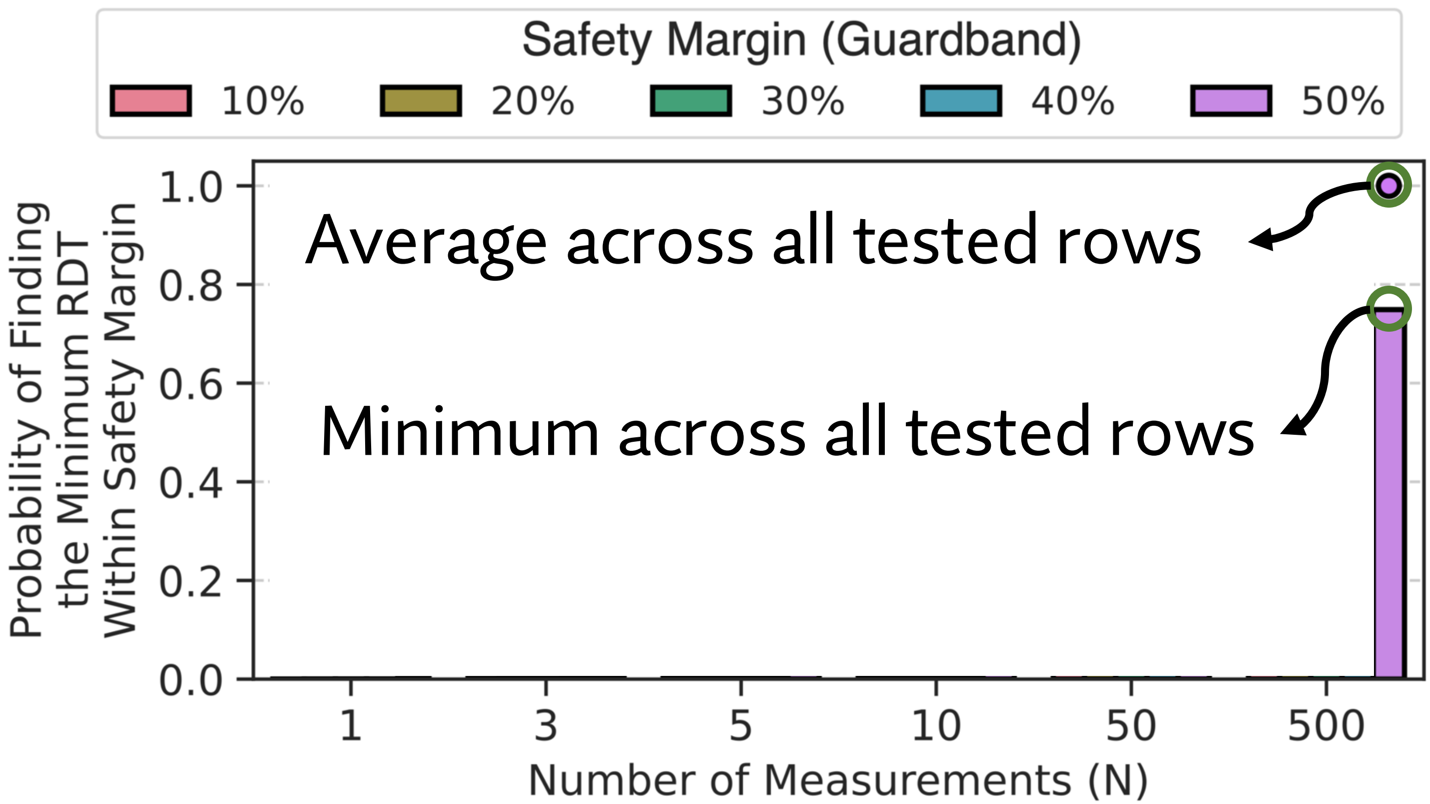
Making Do With Few RDT Measurements

- A system designer might measure RDT a few times and apply a safety margin (guardband) to the minimum observed value

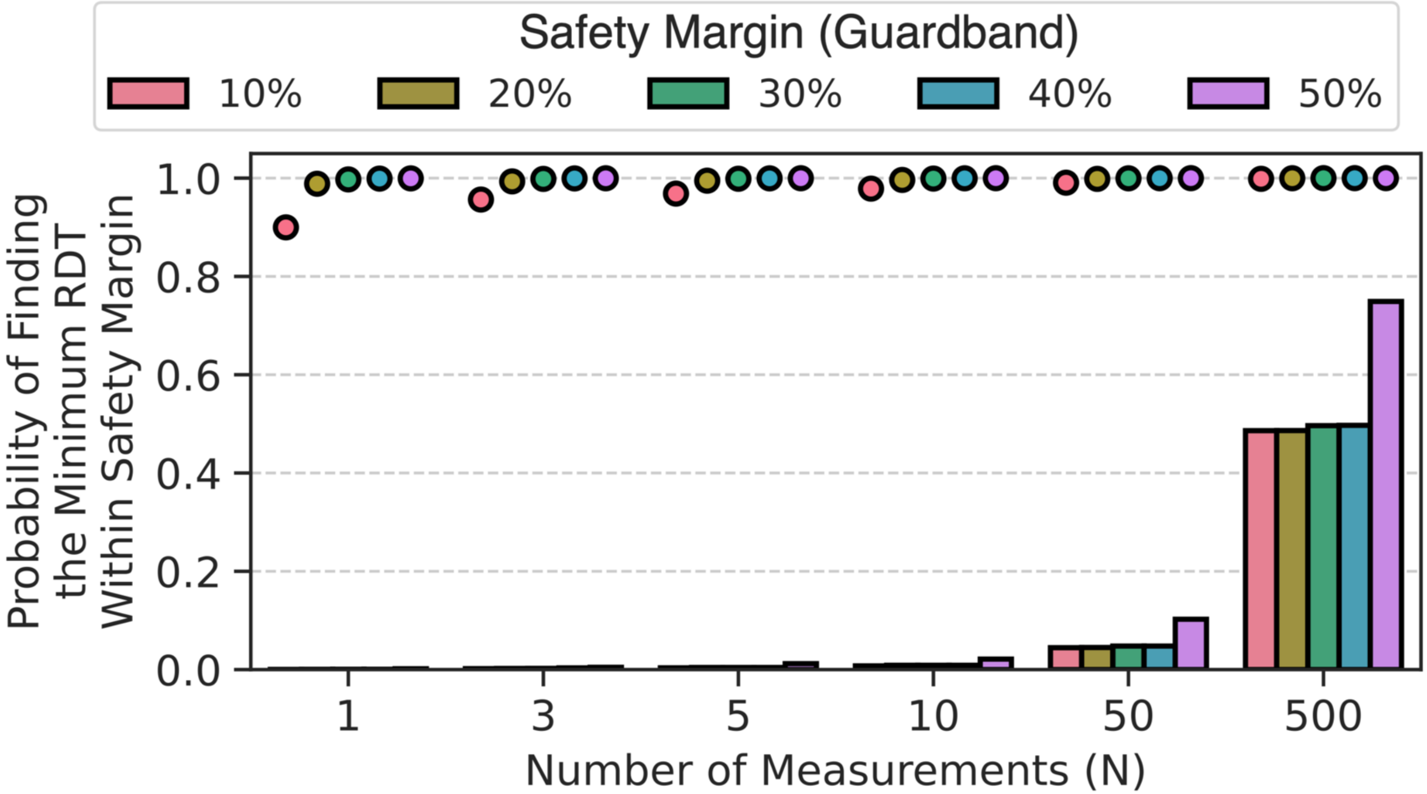


Making Do With Few RDT Measurements

- A system designer might measure RDT a few times and apply a safety margin (guardband) to the minimum observed value



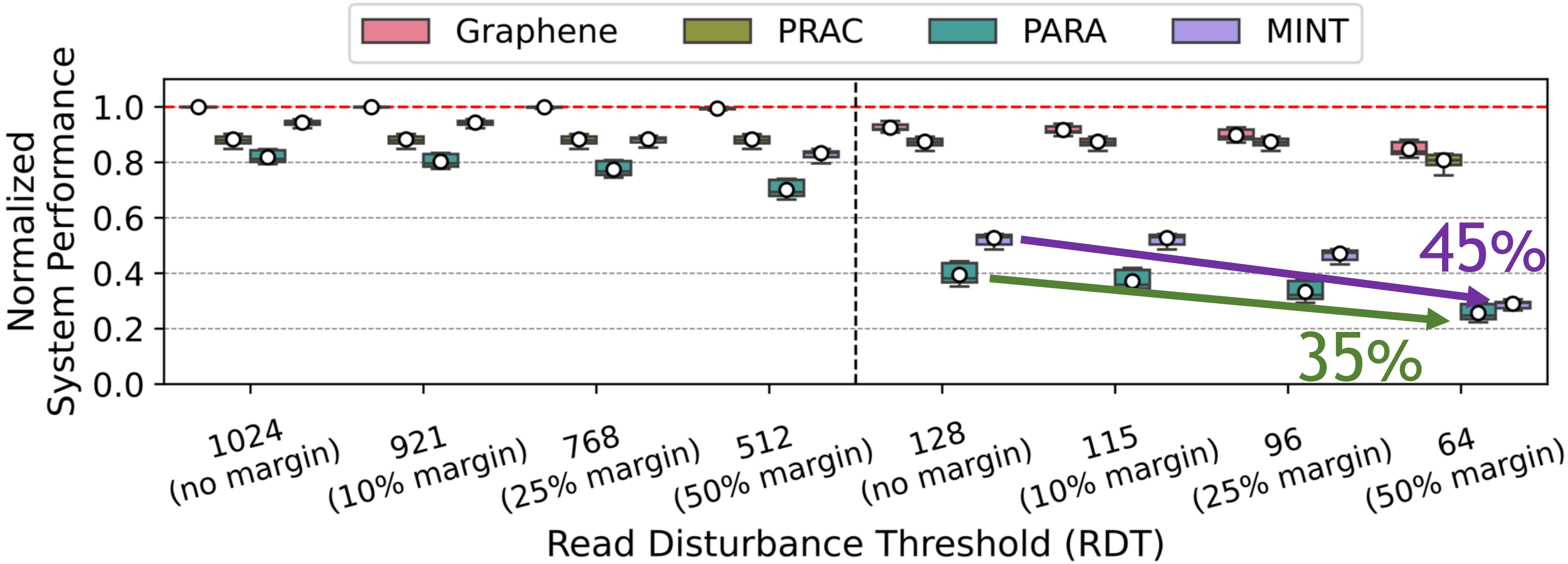
Making Do With Few RDT Measurements



A large guardband does not guarantee that the minimum RDT is always identified

Using guardbands alone is likely not effective

RDT Guardband Increases Performance Overheads



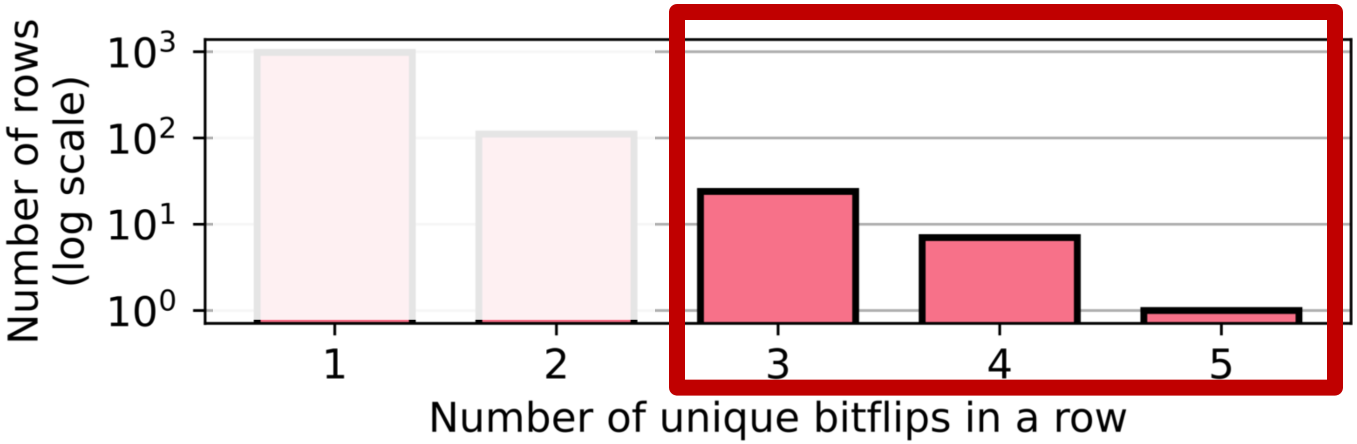
50% RDT safety margin can induce
45% additional overhead (over no margin)

Relying **solely** on guardbands **not** recommended

Combining ECC and Guardbands (I)

- Single-error correcting double-error detecting (SECDED) or Chipkill ECC combined with guardbands could mitigate VRD-induced bitflips

Unique bitflips when 10% RDT guardband applied



10% guardband combined w/ ECC is likely unsafe

Combining ECC and Guardbands (II)

RDT guardbands ≥20% yield 1 unique bitflip in a row

Given our limited measurement dataset (10K measurements)
RDT guardbands ≥20% combined with ECC
may prevent VRD-induced read disturbance bitflips

More detailed analysis (following a large-scale study)
needed to make a definitive conclusion

More in the Paper

- Hypothetical explanation for VRD
- Effect of True- and Anti-Cell Layout
  - Presence of true- and anti-cells in the victim row does not significantly affect the RDT distribution
- Read disturbance mitigation evaluation methodology
- Probability of errors at the worst observed bitflip rate for 10% RDT guardband
  - SEC, SECDED, and Chipkill-like (SSC)
- Read disturbance testing time and energy consumption
- Detailed information on tested modules and chips

More in the Paper

https://arxiv.org/pdf/2502.13075

Variable Read Disturbance:
An Experimental Analysis of Temporal Variation in DRAM Read Disturbance

Ataberk Olgun† F. Nisa Bostancı† İsmail Emir Yüksel† Oğuzhan Canpolat† Haocong Luo†
Geraldo F. Oliveira† A. Giray Yağlıkçı† Minesh Patel‡ Onur Mutlu†

ETH Zurich† *Rutgers University‡*

Modern DRAM chips are subject to read disturbance errors. These errors manifest as security-critical bitflips in a victim DRAM row *that is physically nearby a repeatedly activated (opened) aggressor row (RowHammer) or an aggressor row that is kept open for a long time (RowPress). State-of-the-art read disturbance mitigations rely on accurate and exhaustive characterization of the* read disturbance threshold *(RDT) (e.g., the number of aggressor row activations needed to induce the first RowHammer or RowPress bitflip) of every DRAM row (of which there are millions or billions in a modern system) to prevent read disturbance bitflips securely and with low overhead.*

We experimentally demonstrate for the first time that the RDT of a DRAM row significantly and unpredictably changes over time. We call this new phenomenon variable read disturbance (VRD). Our extensive experiments using 160 DDR4 chips and 4 HBM2 chips from three major manufacturers yield three key observations. First, it is very unlikely that relatively few RDT measurements can accurately identify the RDT of a DRAM row. The minimum RDT of a DRAM row appears after tens of thousands of measurements (e.g., up to 94,467), and the minimum RDT of a DRAM row is 3.5× smaller than the maximum RDT observed for that row. Second, the probability of accu-

row) *many times* (e.g., tens of thousands of times) induces *RowHammer bitflips* in physically nearby rows (i.e., victim rows) [1]. Keeping the aggressor row open for a long period of time amplifies the effects of read disturbance and induces *RowPress bitflips*, *without* requiring *many* repeated aggressor row activations [4].

A large body of work [1, 3, 26, 32, 39, 45, 69–141] proposes various techniques to mitigate DRAM read disturbance bitflips. Many high-performance and low-overhead mitigation techniques [1, 73, 74, 76, 79, 82–84, 86, 87, 91, 97, 133–135, 137–139, 142–146], including those that are used and standardized by industry [121, 126, 138, 139, 144], prevent read disturbance bitflips by *preventively* refreshing (i.e., opening and closing) a victim row *before* a bitflip manifests in that row.

To securely prevent read disturbance bitflips at low performance and energy overhead, it is important to *accurately* identify the amount of read disturbance that a victim row can withstand before experiencing a read disturbance bitflip. This amount is typically quantified using the *hammer count (the number of aggressor row activations) needed to induce the first read disturbance bitflip* in a victim row. We call this metric the *read disturbance threshold (RDT)* of the victim row.

Talk Outline

I. Motivation

II. Experimental Characterization Methodology

III. Foundational Results

IV. In-Depth Analysis of VRD

V. Implications for System Security and Robustness

VI. Conclusion

VRD Conclusion

Variable Read Disturbance (VRD)

The read disturbance threshold changes unpredictably over time

Minimum RDT (of a row) may appear after many measurements

RDT for a DRAM row can vary by 3.5X

Identifying the minimum RDT is challenging and time-intensive

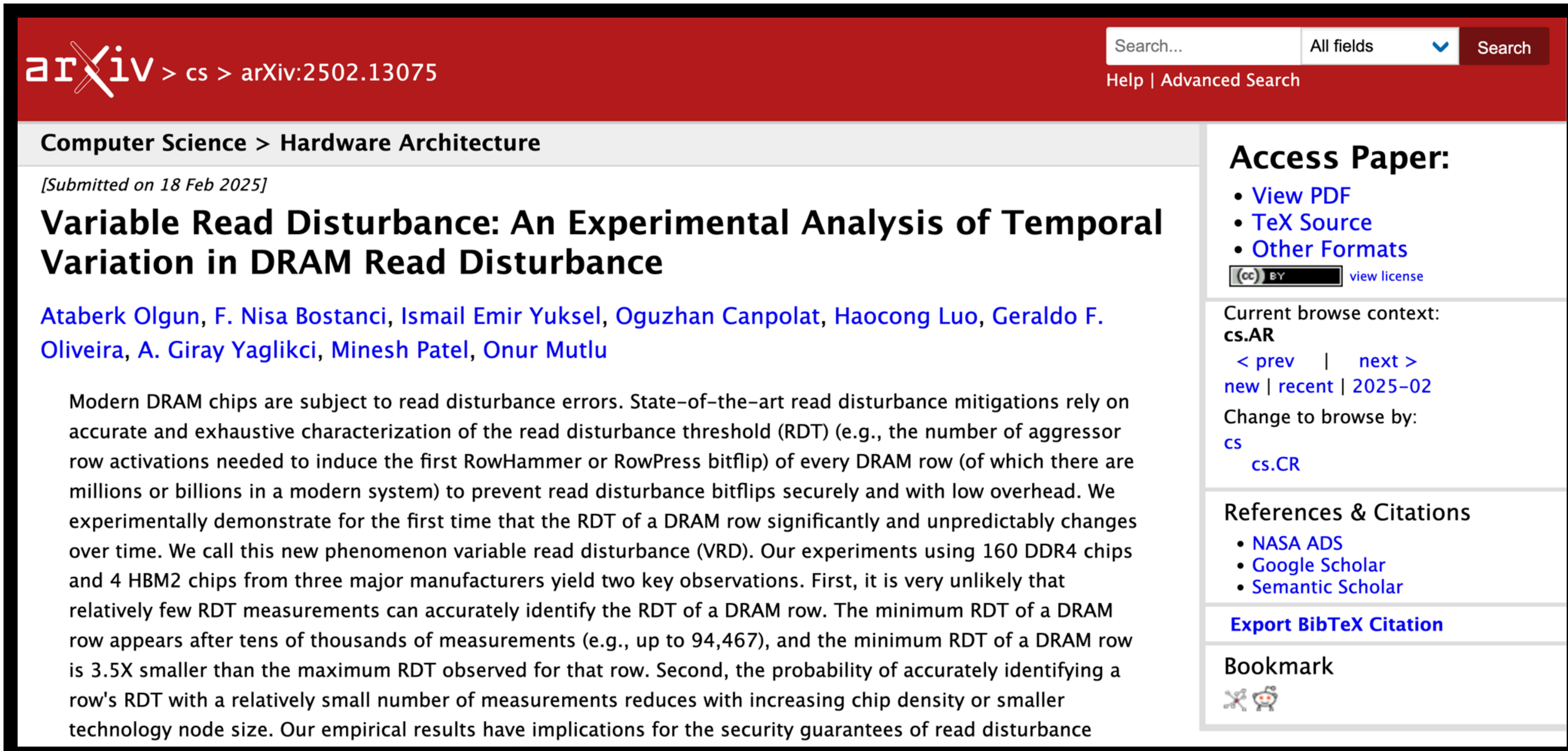
Given our limited read disturbance bitflip dataset,
guardbands combined with error-correcting codes
may be a solution for VRD-induced bitflips.

More data and analyses needed to make definitive conclusion

Future work could alleviate the shortcomings of existing mitigations & develop better understanding of inner workings of VRD

Extended Version on arXiv

https://arxiv.org/pdf/2502.13075



arXiv > cs > arXiv:2502.13075

Search... All fields Search

Help | Advanced Search

Computer Science > Hardware Architecture

[Submitted on 18 Feb 2025]

Variable Read Disturbance: An Experimental Analysis of Temporal Variation in DRAM Read Disturbance

Ataberk Olgun, F. Nisa Bostanci, Ismail Emir Yuksel, Oguzhan Canpolat, Haocong Luo, Geraldo F. Oliveira, A. Giray Yaglikci, Minesh Patel, Onur Mutlu

Modern DRAM chips are subject to read disturbance errors. State-of-the-art read disturbance mitigations rely on accurate and exhaustive characterization of the read disturbance threshold (RDT) (e.g., the number of aggressor row activations needed to induce the first RowHammer or RowPress bitflip) of every DRAM row (of which there are millions or billions in a modern system) to prevent read disturbance bitflips securely and with low overhead. We experimentally demonstrate for the first time that the RDT of a DRAM row significantly and unpredictably changes over time. We call this new phenomenon variable read disturbance (VRD). Our experiments using 160 DDR4 chips and 4 HBM2 chips from three major manufacturers yield two key observations. First, it is very unlikely that relatively few RDT measurements can accurately identify the RDT of a DRAM row. The minimum RDT of a DRAM row appears after tens of thousands of measurements (e.g., up to 94,467), and the minimum RDT of a DRAM row is 3.5X smaller than the maximum RDT observed for that row. Second, the probability of accurately identifying a row's RDT with a relatively small number of measurements reduces with increasing chip density or smaller technology node size. Our empirical results have implications for the security guarantees of read disturbance

Access Paper:

- View PDF
- TeX Source
- Other Formats

Current browse context:
cs.AR
< prev | next >
new | recent | 2025-02
Change to browse by:
cs
cs.CR

References & Citations

- NASA ADS
- Google Scholar
- Semantic Scholar

Export BibTeX Citation

Bookmark

Variable Read Disturbance (VRD)
An Experimental Analysis of Temporal Variation in DRAM Read Disturbance

Ataberk Olgun, F. Nisa Bostancı, İsmail Emir Yüksel
Oğuzhan Canpolat, Haocong Luo, Geraldo F. Oliveira
A. Giray Yağlıkçı, Minesh Patel, Onur Mutlu

https://arxiv.org/pdf/2502.13075

ETH zürich SAFARI RUTGERS THE STATE UNIVERSITY OF NEW JERSEY

Variable Read Disturbance

Backup Slides

Is this Experiment Noise?

- Short answer: **no**

- Hypothetical explanation for VRD:
  Randomly changing charge trap state in the active region

- Long answer:
  We cannot identify any independent variables
  within our control that allow reliably predicting
  the minimum RDT despite extensive testing

- Device-level studies should confirm our hypotheses

Hypothetical Explanation for VRD

- No device-level study shows temporal variations in read disturbance vulnerability
- Electron migration and injection into victim cell is a major read disturbance failure mechanism
- This mechanism is assisted by charge traps in the shared active region of the victim and aggr. cell
- Temporal variation attributed to randomly changing occupied/unoccupied states of charge traps

RDT Distribution Across Chips

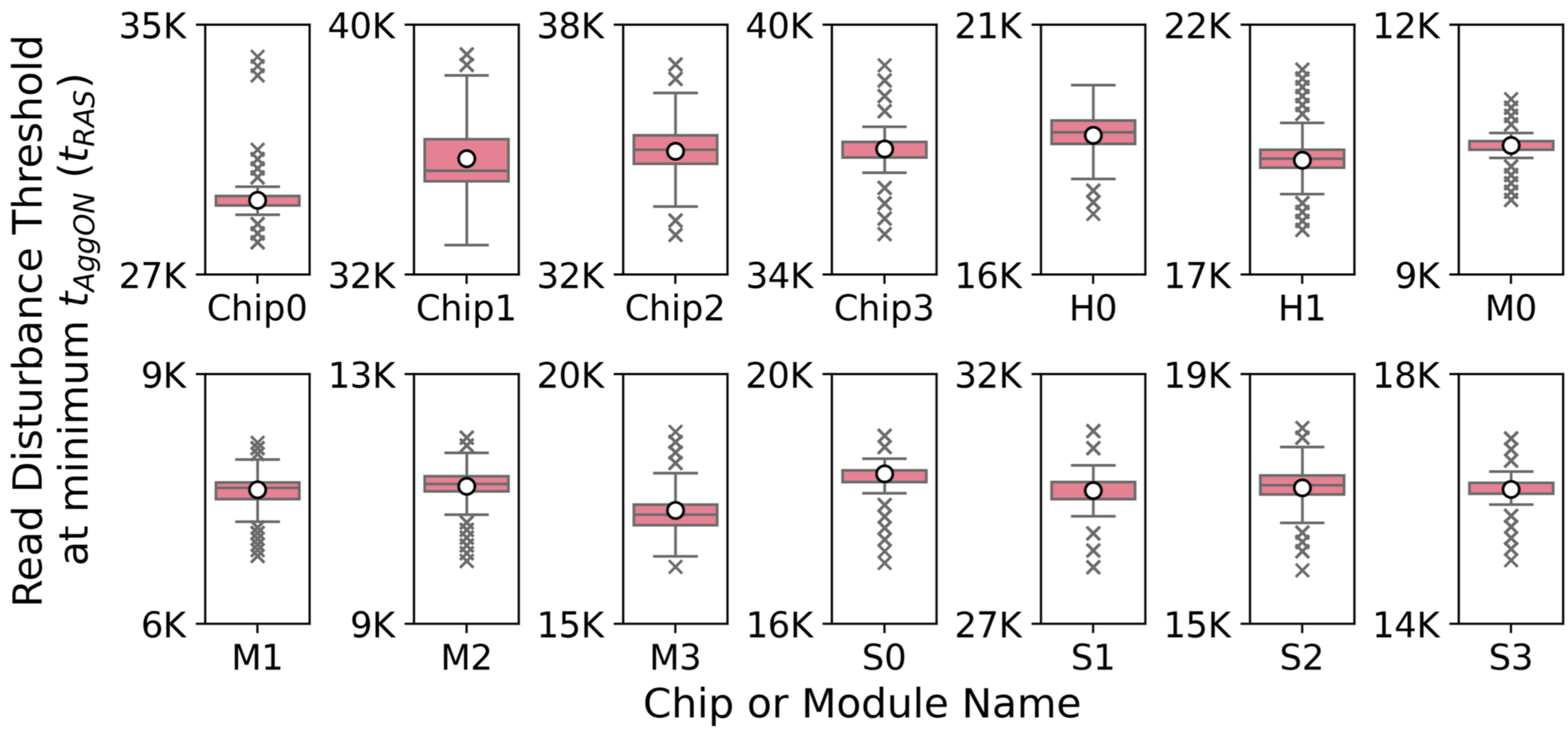


Figure 3: RDT distribution of a single victim row in each tested module and chip

of Consecutive Measurements That Yield the Same RDT Value

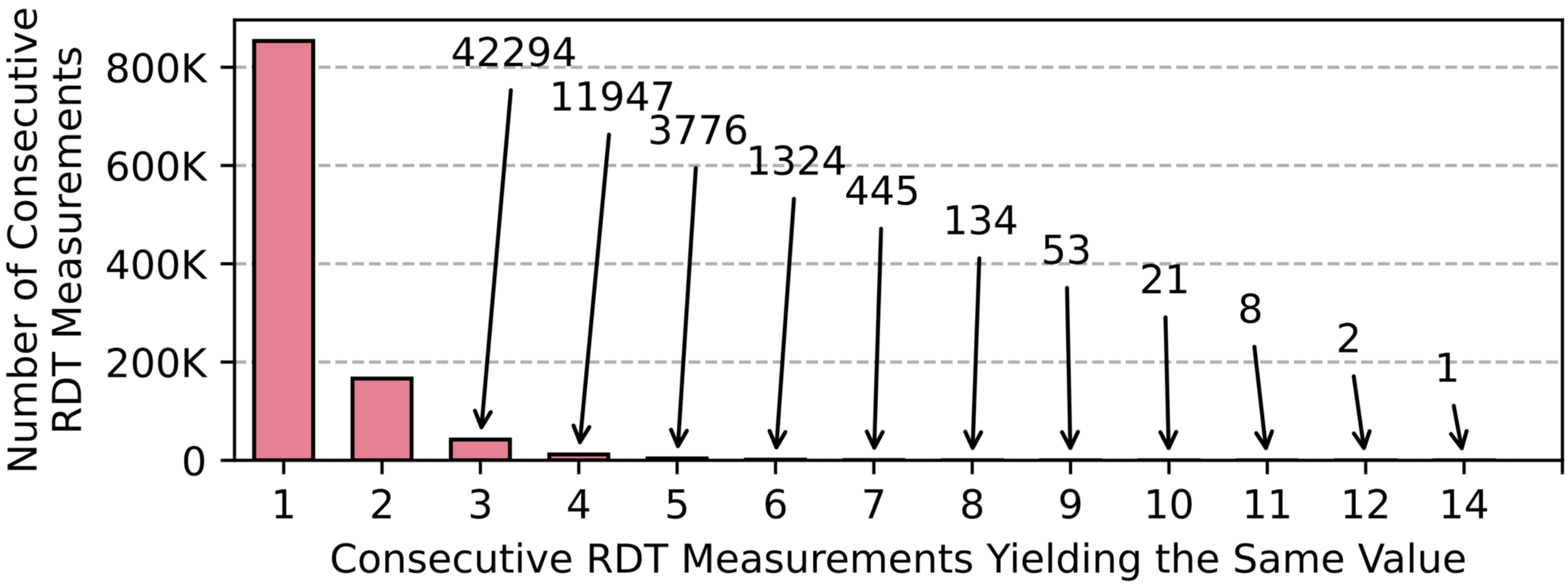
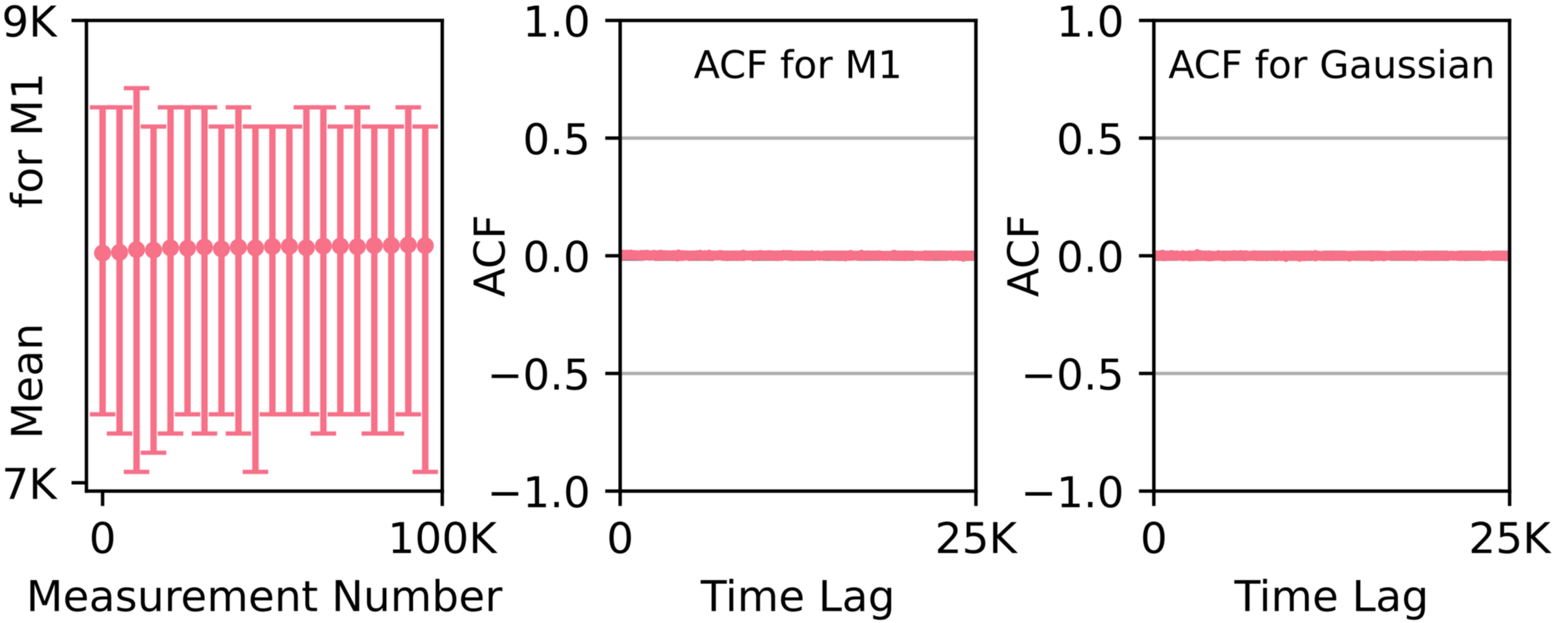
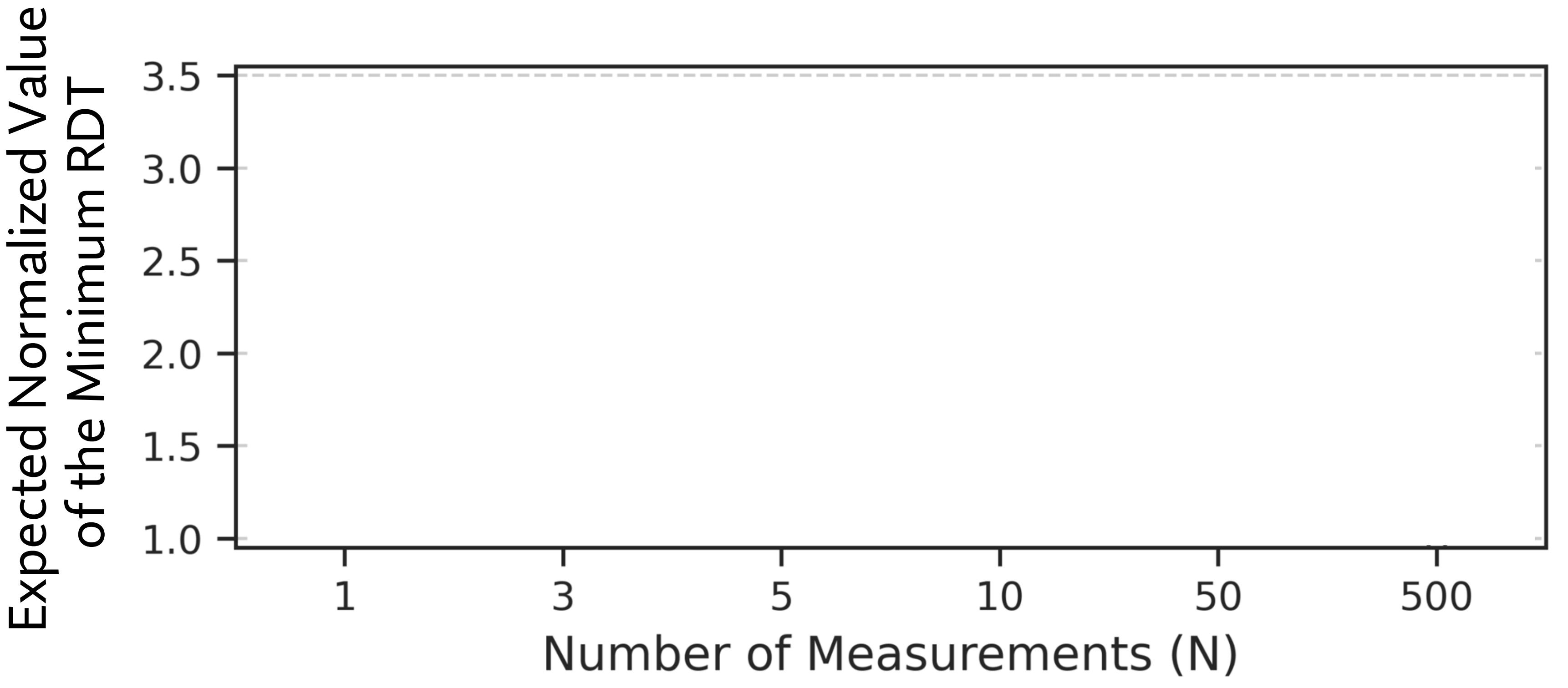


Figure 5: Histogram of the number of measurements across which a row's RDT exhibits the same value

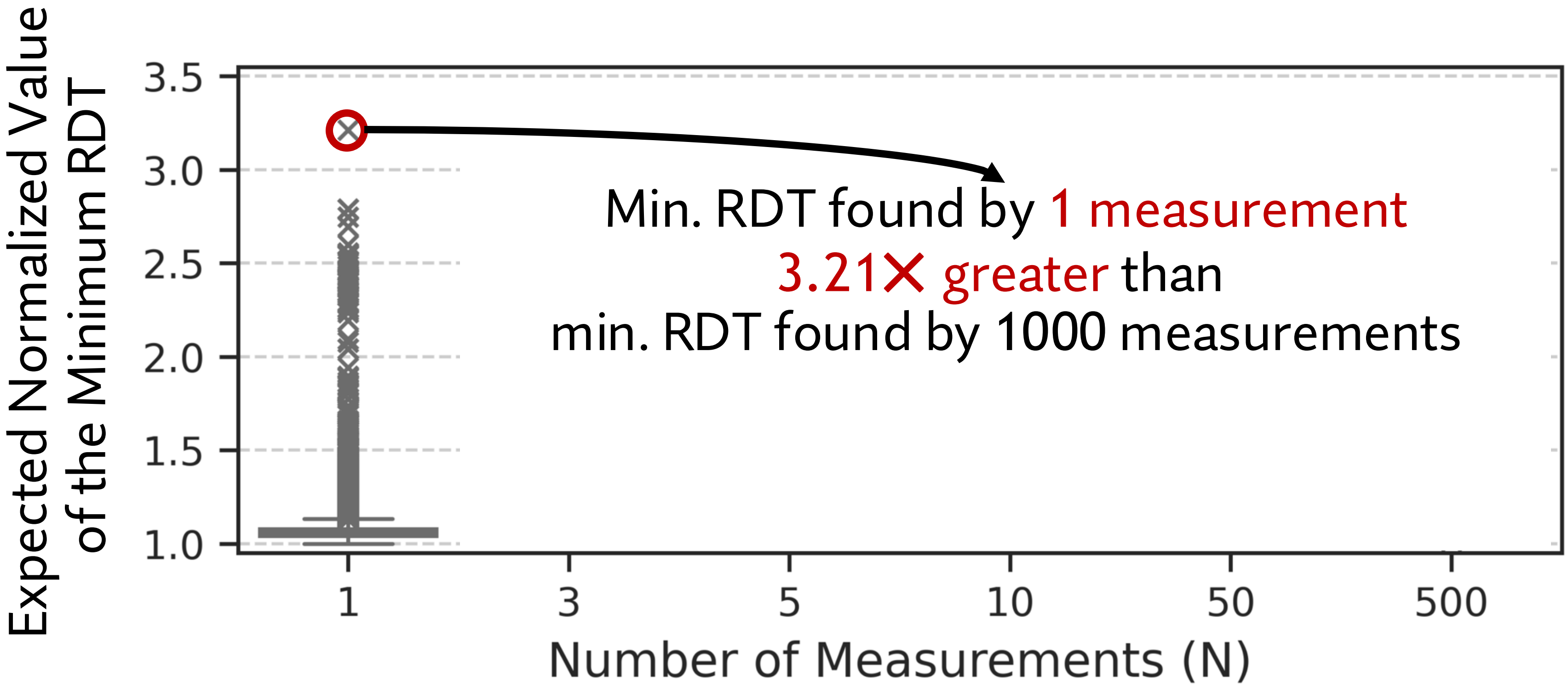
Autocorrelation Function Tests



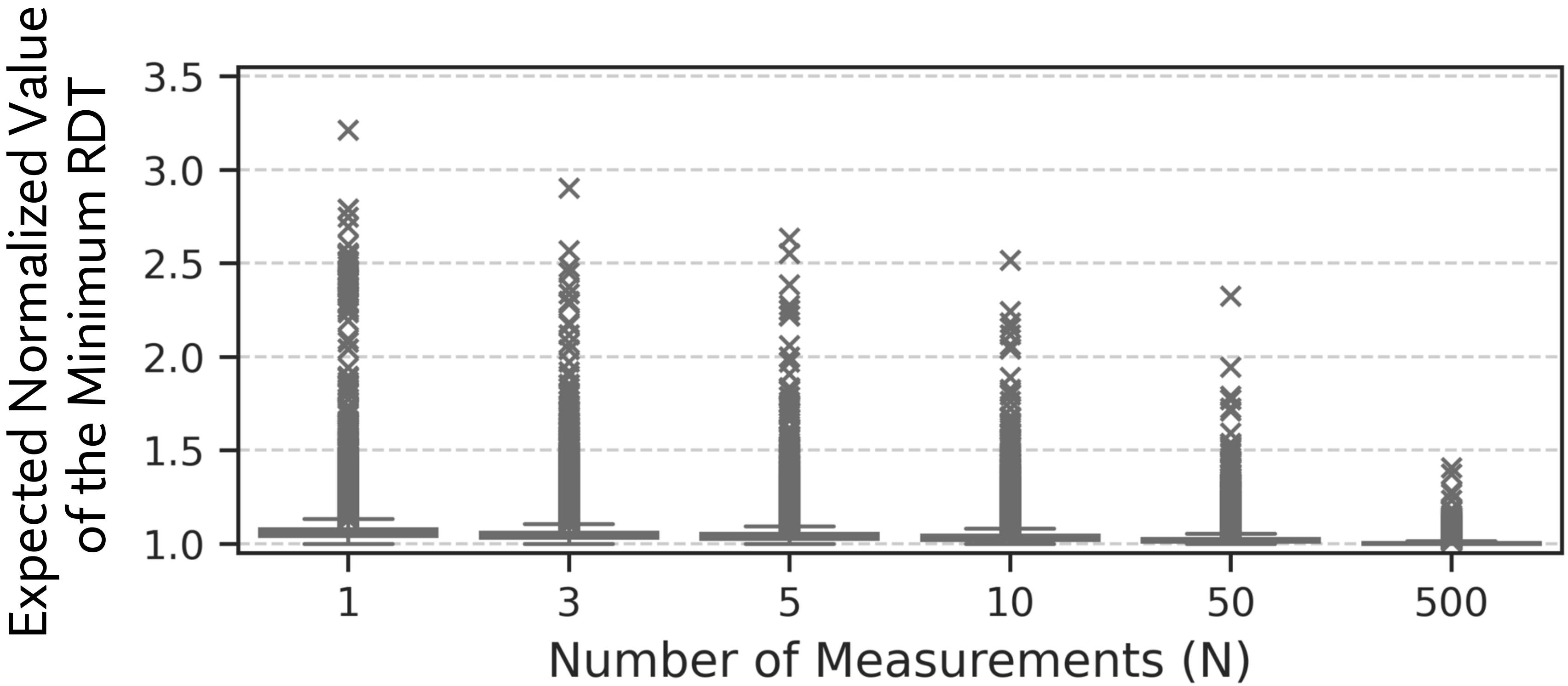
Expected Value of the Minimum RDT



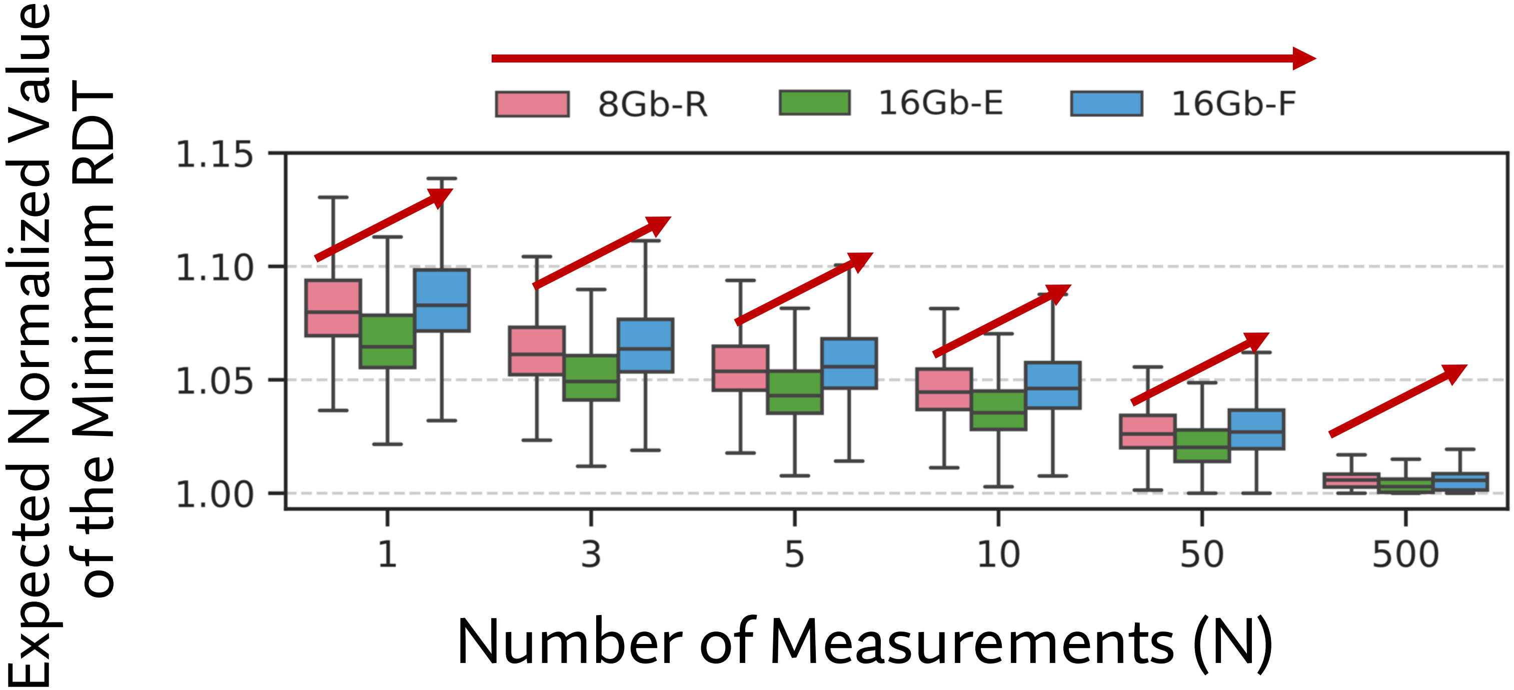
Expected Value of the Minimum RDT



Expected Value of the Minimum RDT

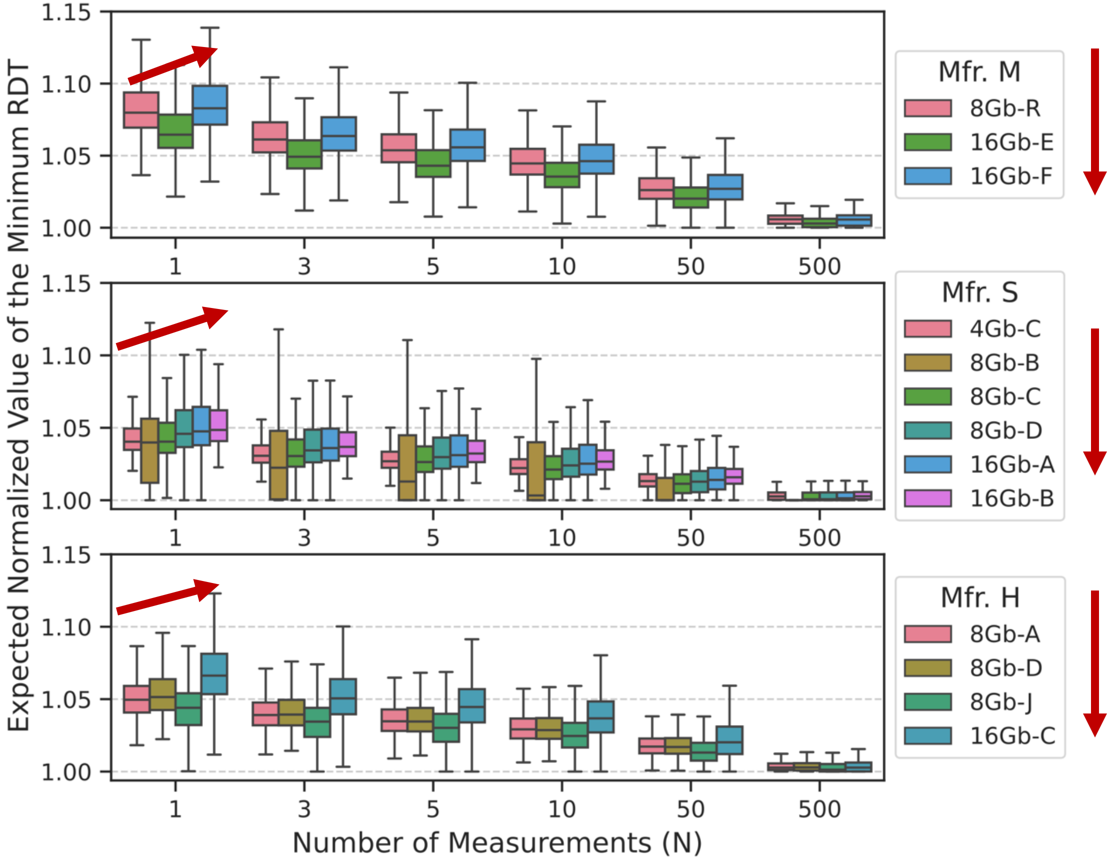


Effect of Die Density and Die Revision (I)



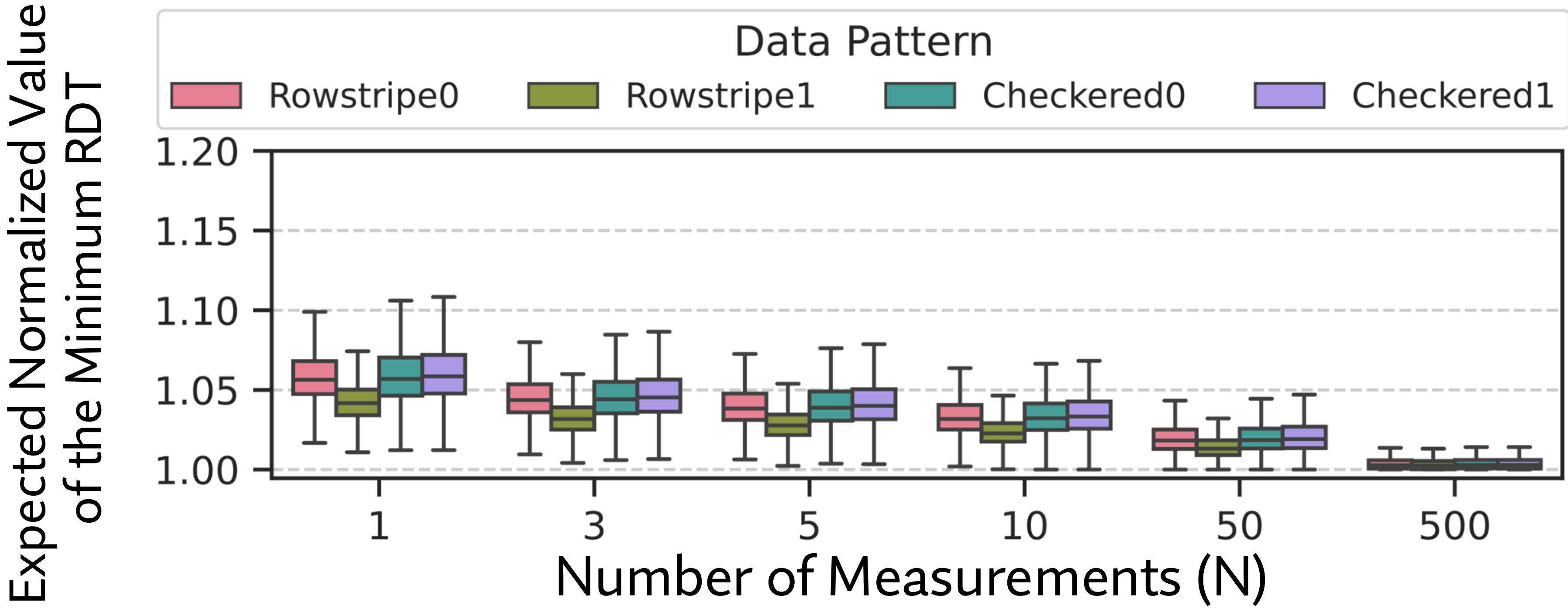
RDT distribution worsens with increasing die density and with advanced DRAM technology

Effect of Die Density and Die Revision (II)

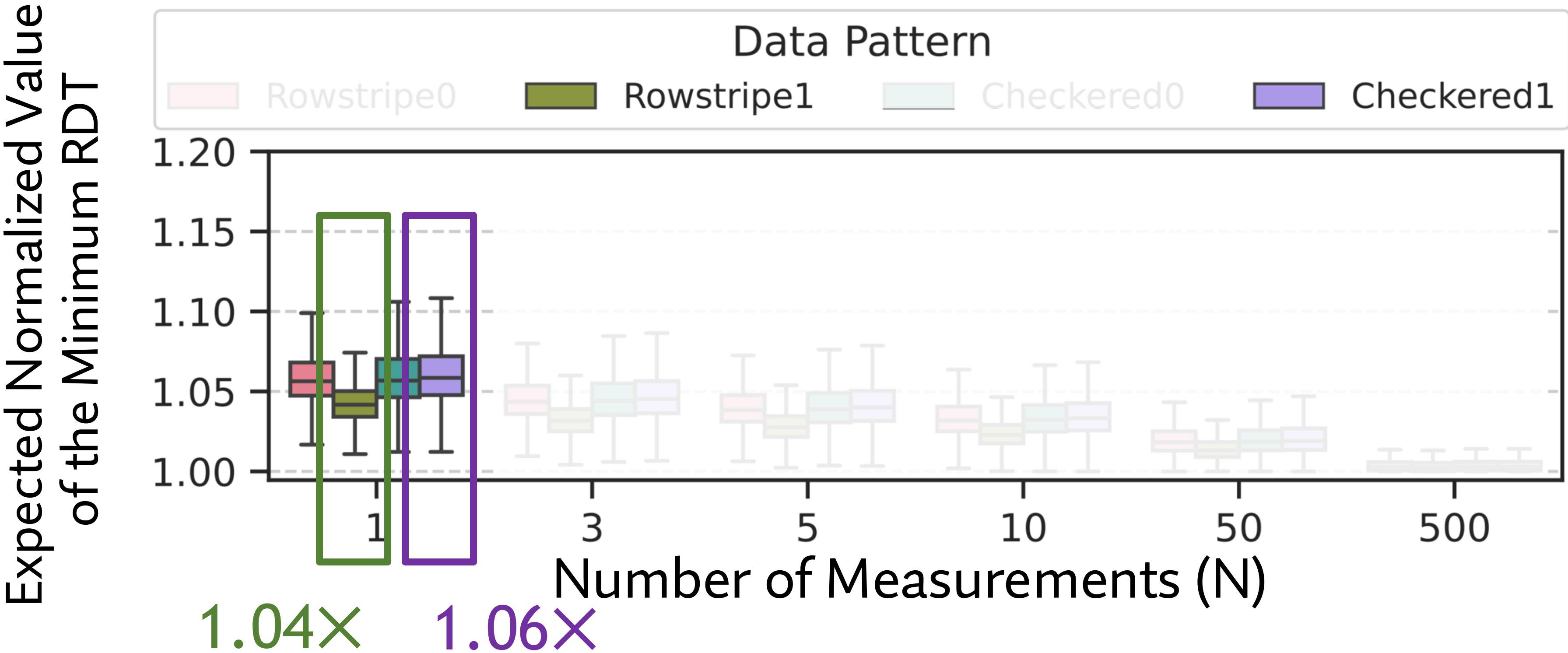


The effect of die density and die revision is consistent across all tested modules

Effect of Data Pattern (I)



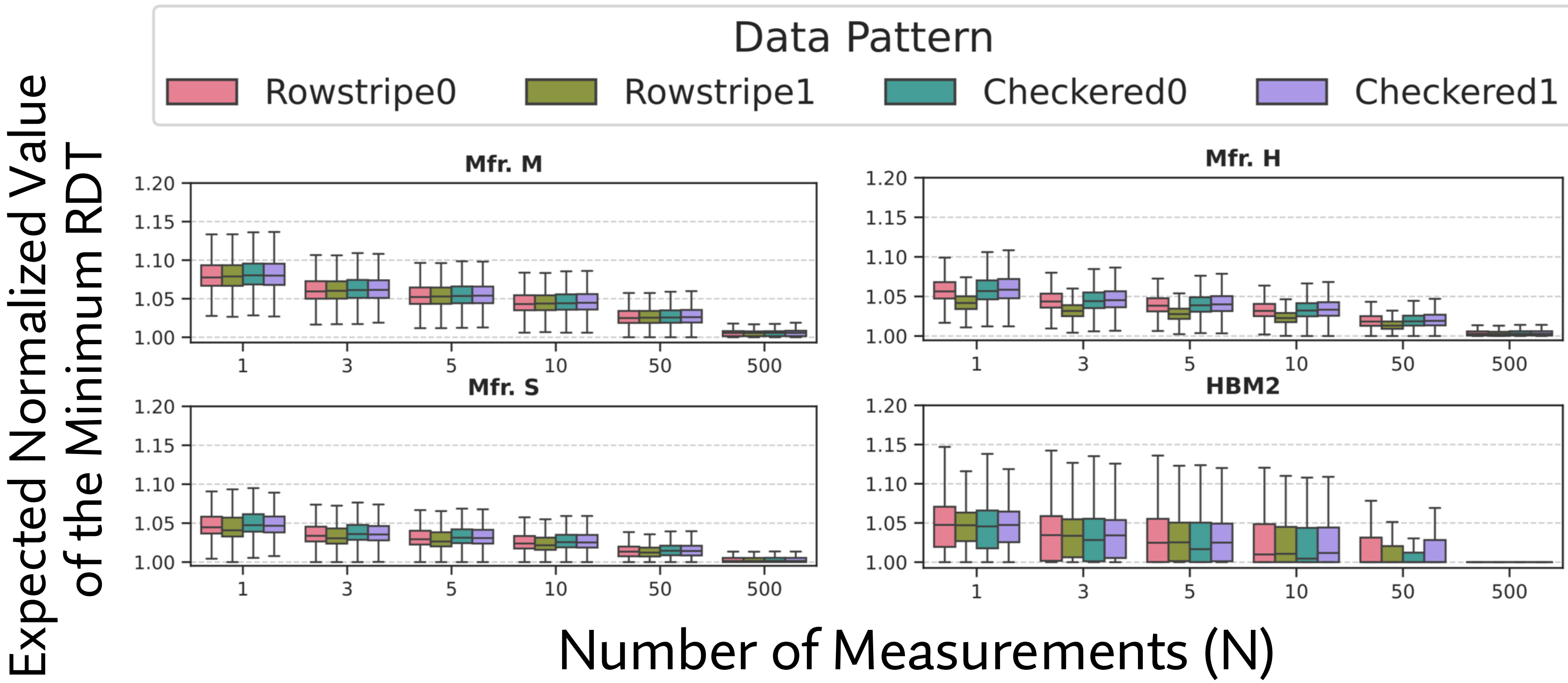
Effect of Data Pattern (I)



expected normalized value for the median row

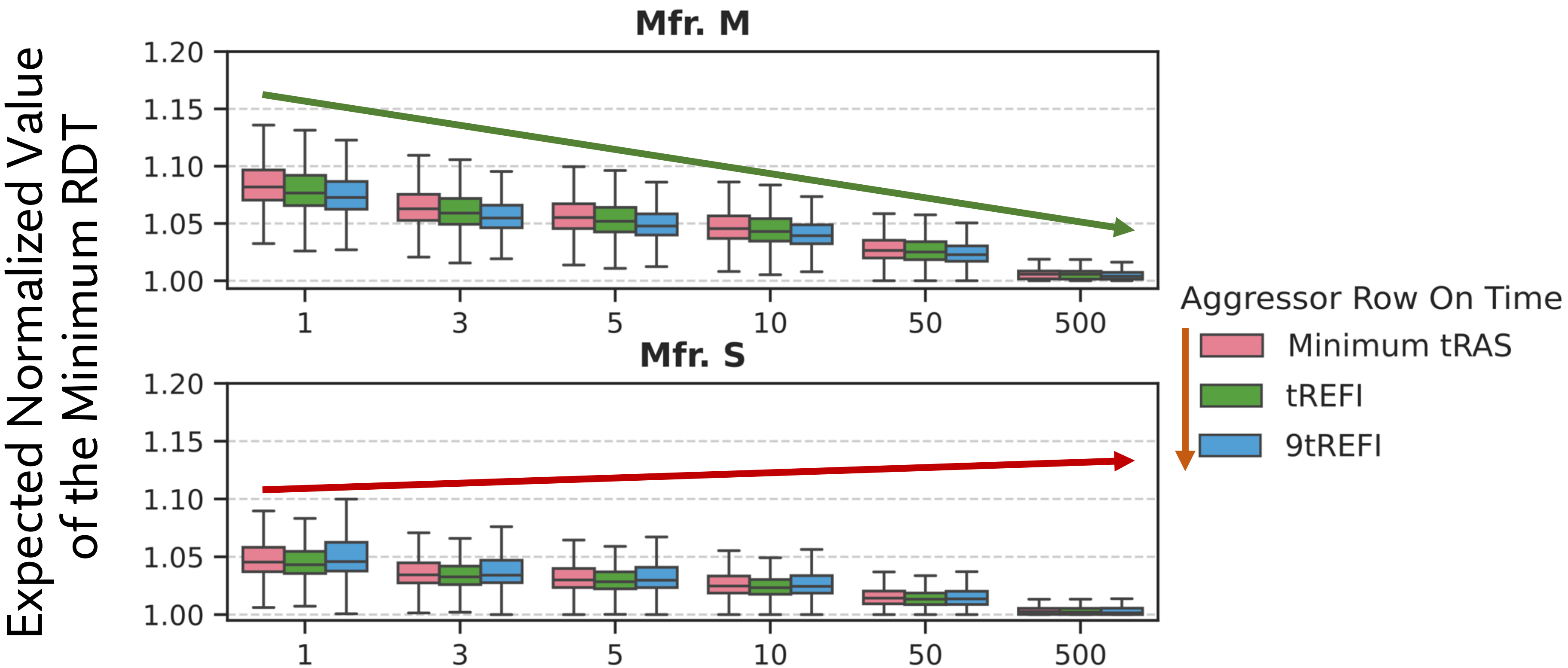
RDT distribution changes with data pattern

Effect of Data Pattern (II)



No single data pattern causes the worst RDT distribution across all tested DRAM chips

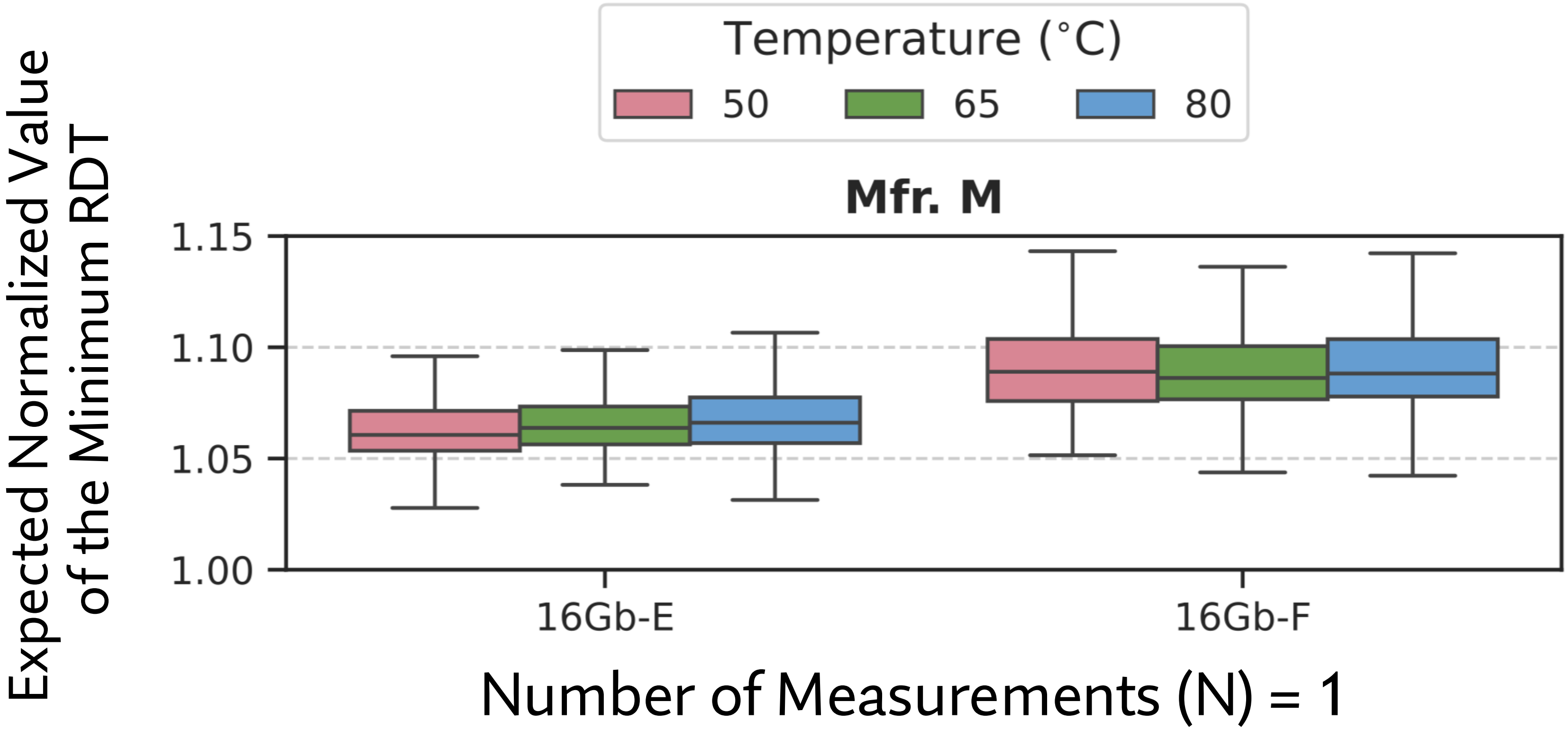
Effect of Aggressor Row On Time



RDT distribution changes with aggressor row on time

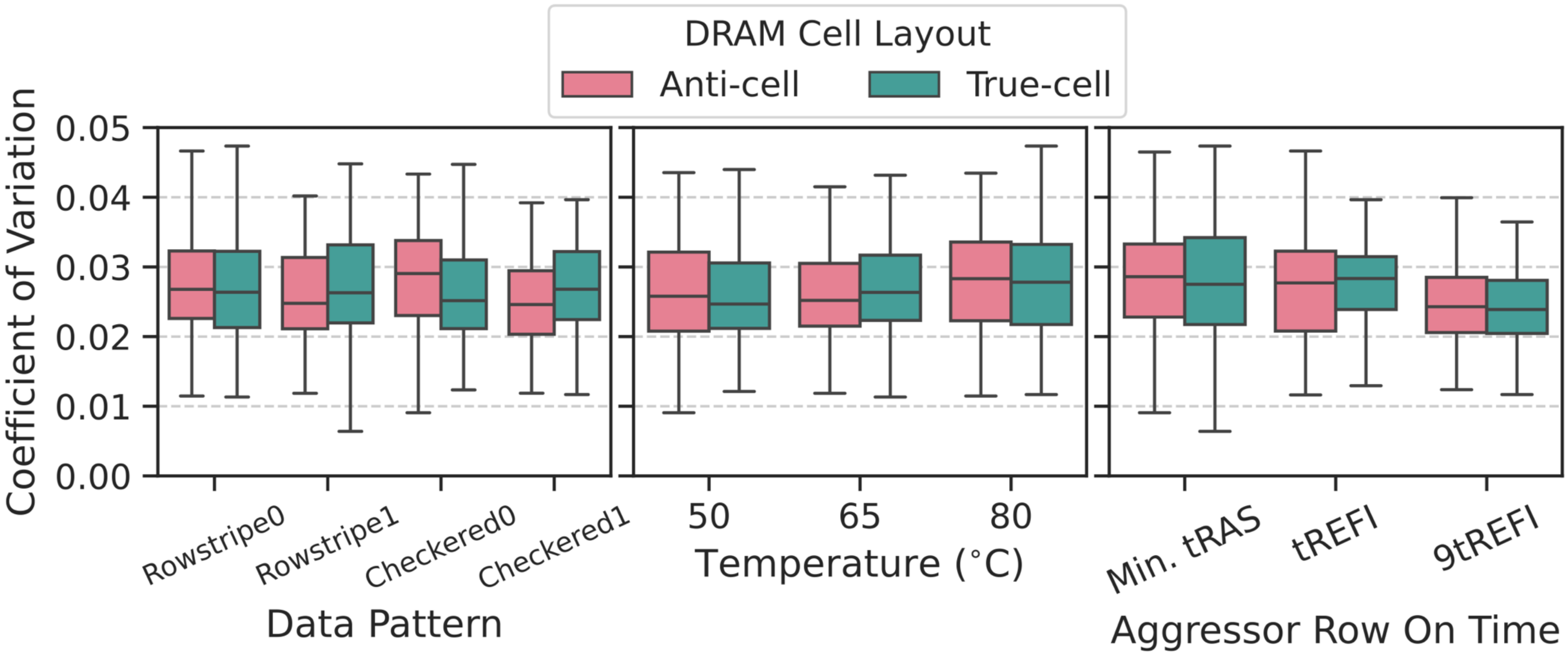
RDT distribution can become better or worse with increasing row on time

Effect of Temperature



RDT distribution tends to change with temperature

Effect of True- and Anti-Cell Layout



The presence of true- and anti-cells in the victim row does not significantly affect the RDT distribution

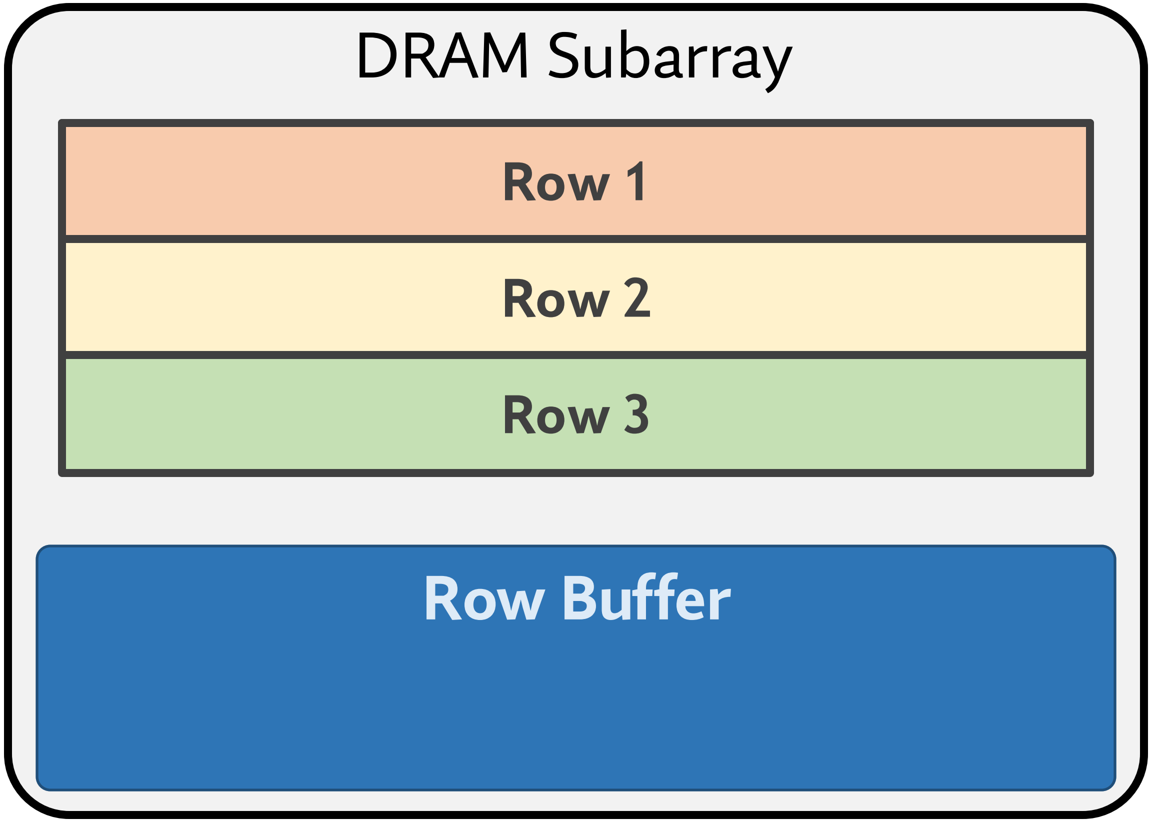
Error Probability Analysis

Table 3: Probability of uncorrectable, undetectable, and detectable uncorrectable errors at the worst error rate we observed empirically so far ($7.6e-5$) using an RDT safety margin of 10% for SEC, SECDED, and Chipkill-like SSC codes. N/A indicates the result category does not exist for the shown ECC type.

Type of error	SEC	SECDED	Chipkill-like (SSC)
Uncorrectable	1.48e-05	1.48e-05	5.66e-05
Undetectable	1.48e-05	2.64e-08	5.66e-05
Detectable uncorrectable	N/A	1.48e-05	N/A

DRAM Operation: Activate and Precharge

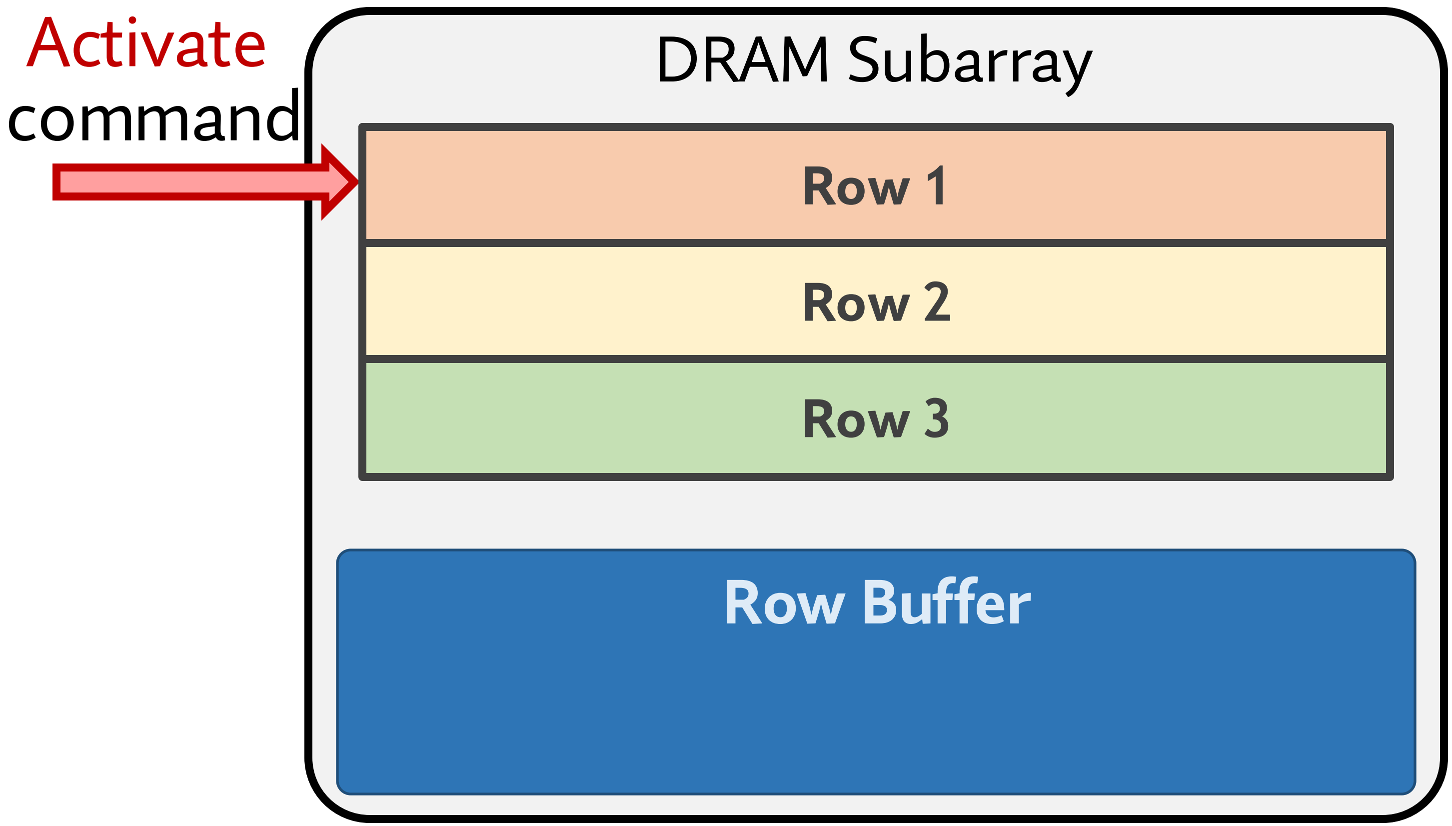
Access data in Row 1



Row 1 is **closed**

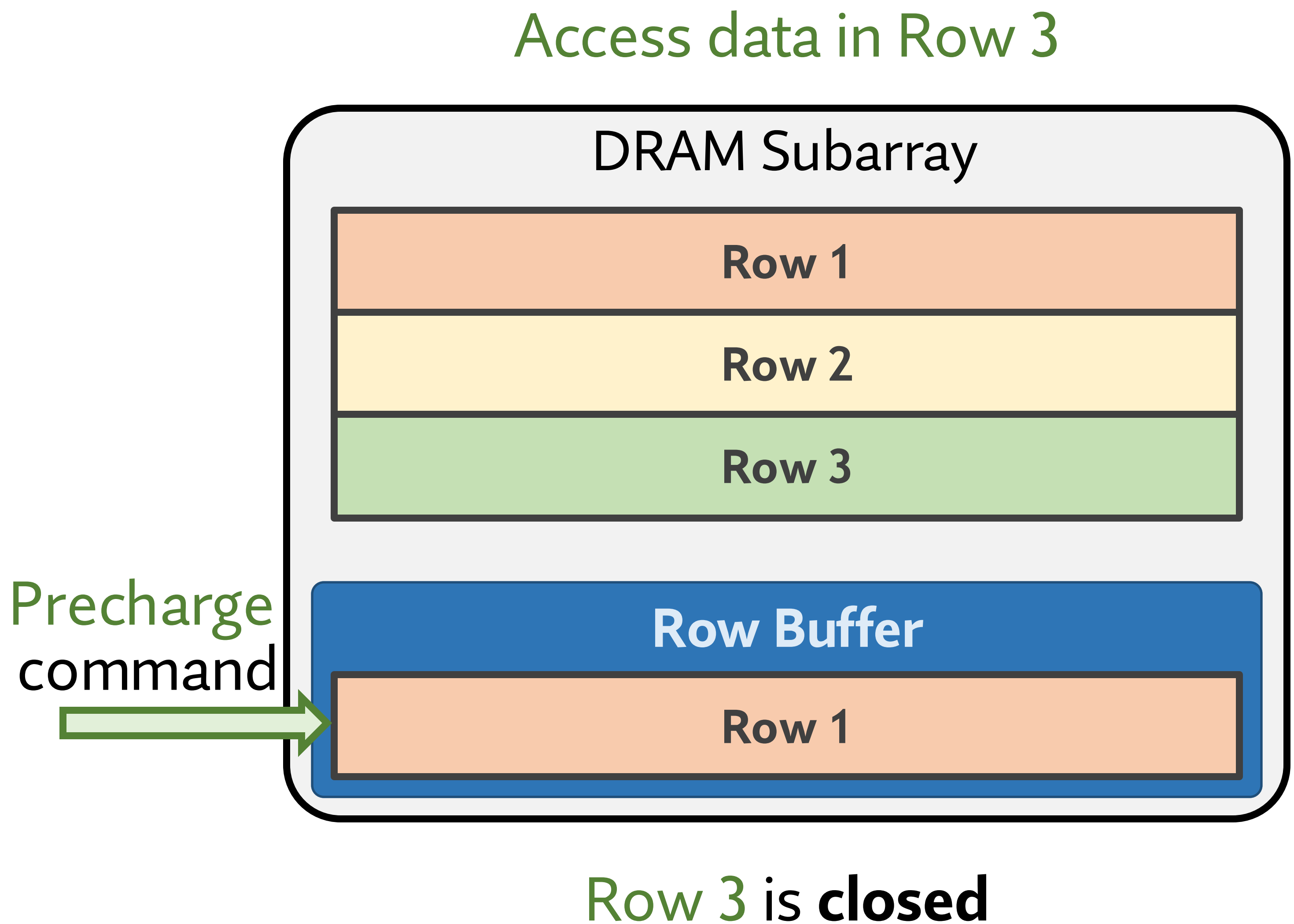
DRAM Operation: Activate and Precharge

Access data in Row 1



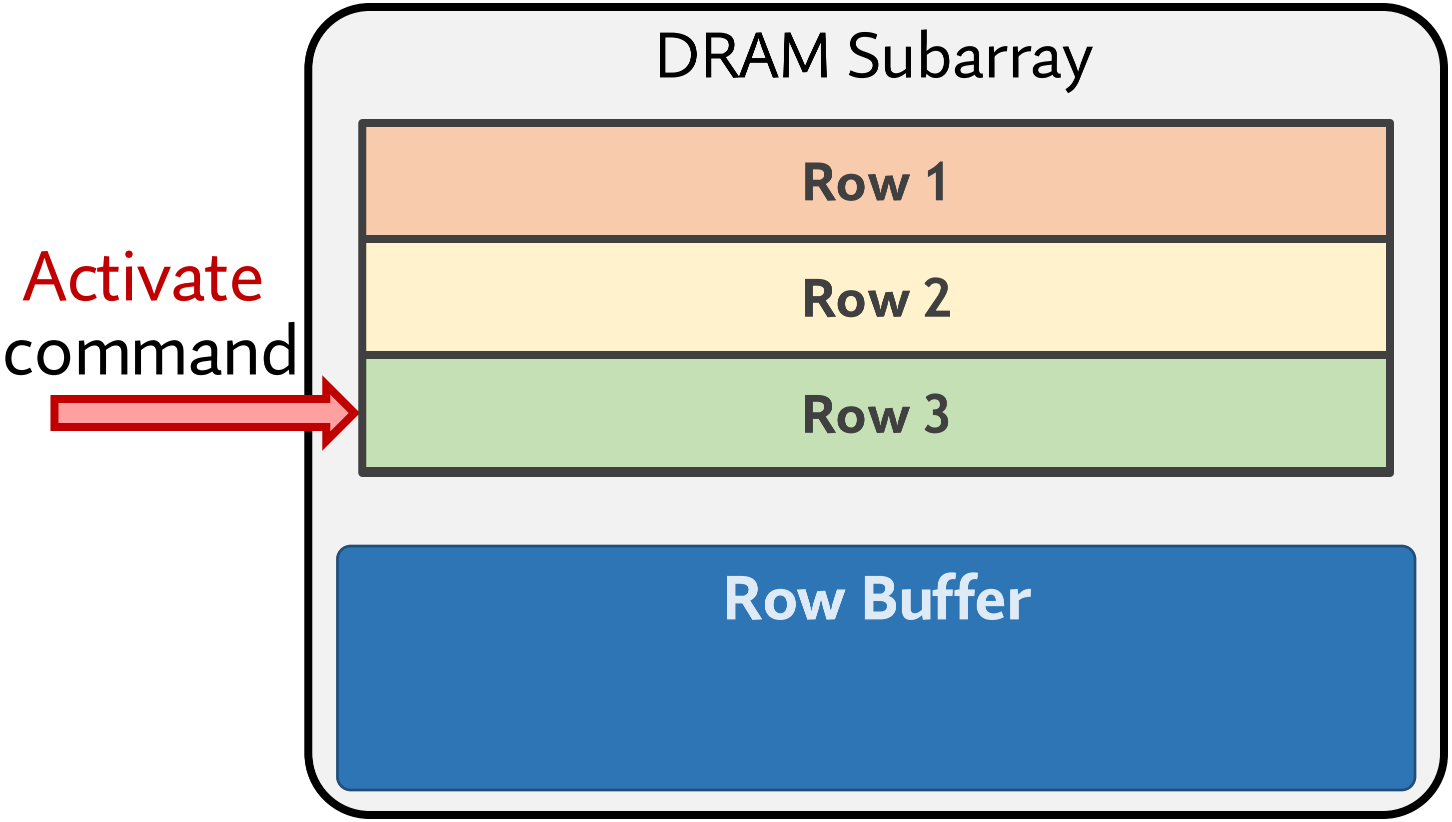
Row 1 is opened
Row 1 is closed

DRAM Operation: Activate and Precharge



DRAM Operation: Activate and Precharge

Access data in Row 3

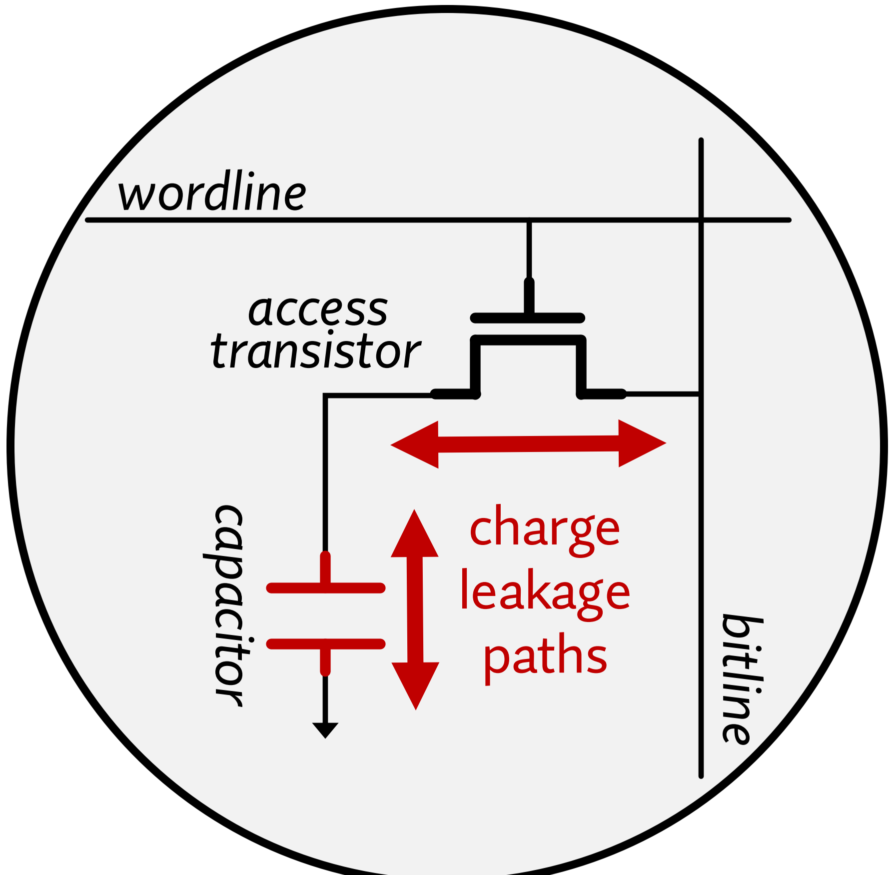


Row 3 is opened

Row 3 is closed

DRAM Cell Leakage

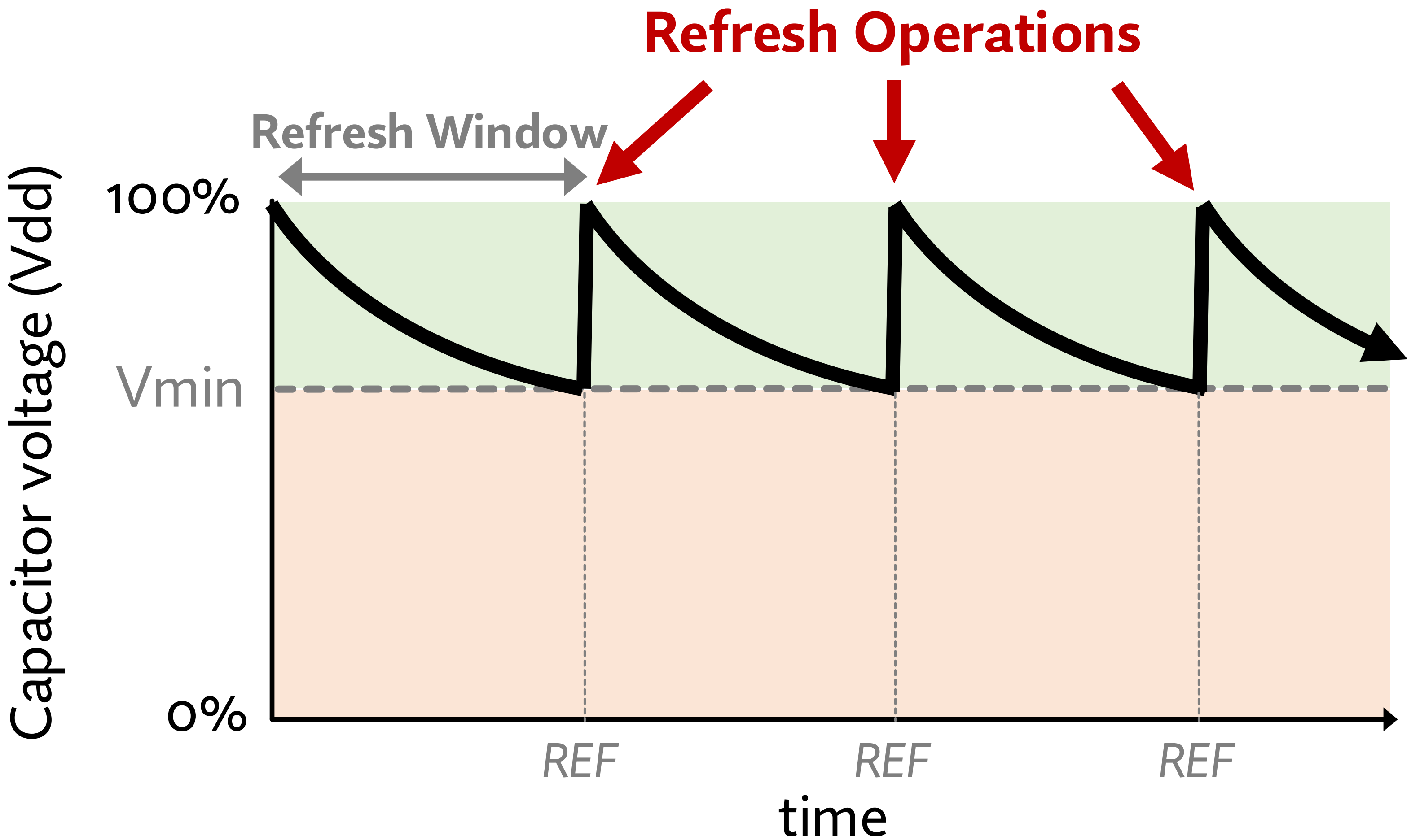
Each cell encodes information in **leaky** capacitors



Stored data is **corrupted** if too much charge leaks (i.e., the capacitor voltage degrades too much)

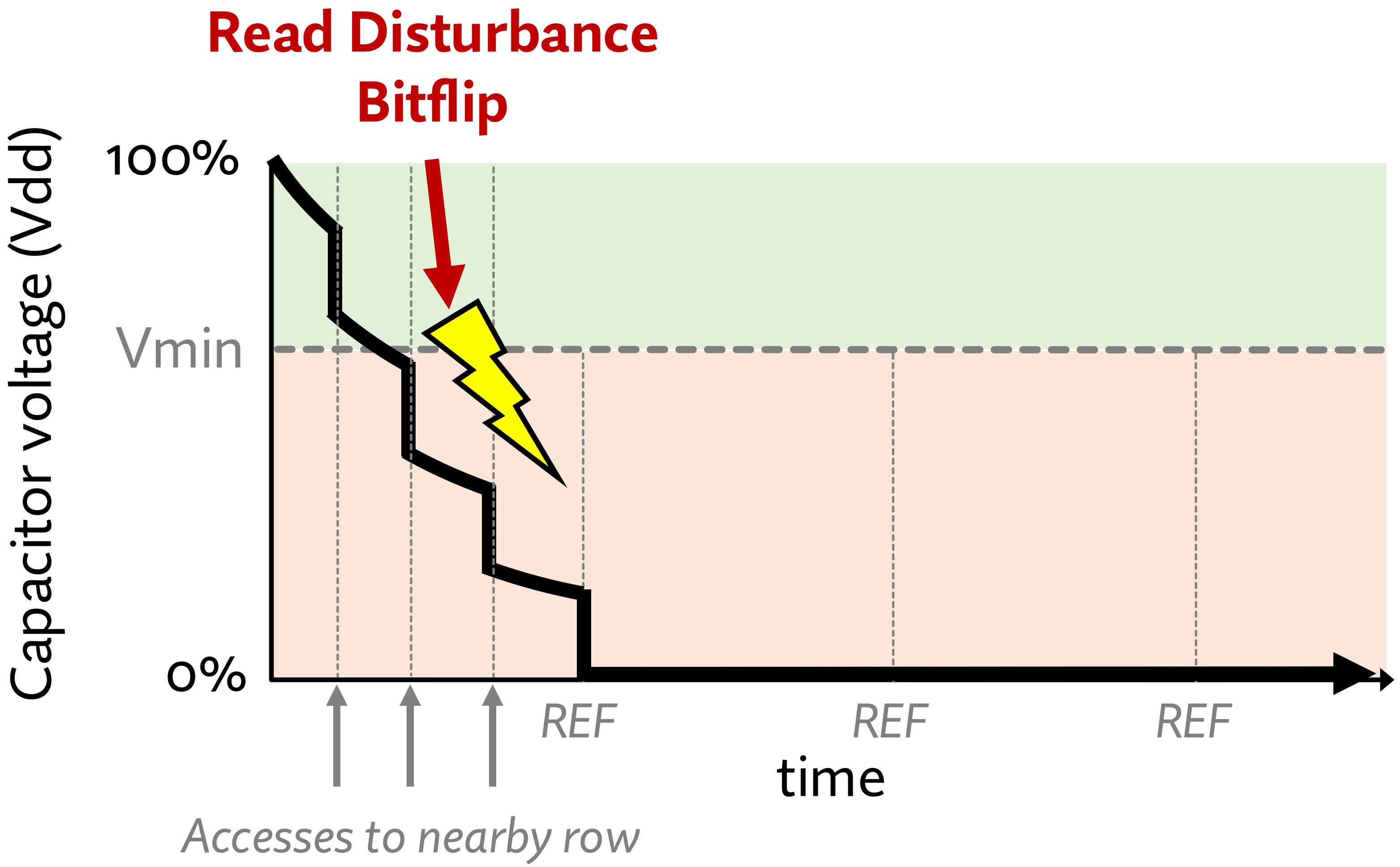
[Patel+, ISCA'17]

DRAM Refresh



Periodic **refresh operations** preserve stored data

Read Disturbance Bitflips



Self-Managing DRAM (SMD)

A Low-Cost Framework for Enabling Autonomous and Efficient DRAM Maintenance Operations

Hasan Hassan, Ataberk Olgun,
A. Giray Yaglikci, Haocong Luo, Onur Mutlu

https://arxiv.org/pdf/2207.13358

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

ETH zürich

SAFARI

Self-Managing DRAM (SMD) Summary

Problem: Implementing new in-DRAM maintenance operations requires modifications in the DRAM interface and other system components

- Modifying the DRAM interface requires a multi-year effort by JEDEC

Goal: Ease and accelerate the process of implementing new in-DRAM maintenance operations and enable more efficient maintenance operations

Key Idea: With a single, simple DRAM interface modification:

- The DRAM chip can reject memory accesses that target an under-maintenance *DRAM region* (e.g., a subarray)
- Implement and modify maintenance operations without future changes

Use Cases: Demonstrate the usefulness and versatility of SMD

- In-DRAM refresh, RowHammer protection, and memory scrubbing

Evaluation: Demonstrate that SMD performs maintenance operations with high performance and high energy efficiency at relatively small DRAM chip and memory controller area costs

SMD Outline

1. Motivation

2. Self-Managing DRAM (SMD)

3. Use Cases

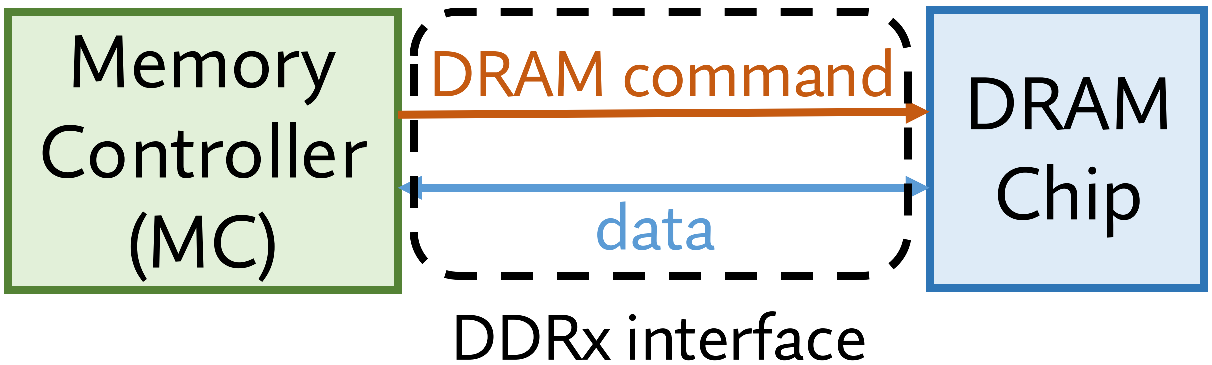
4. Evaluations

5. Conclusion and Takeaways

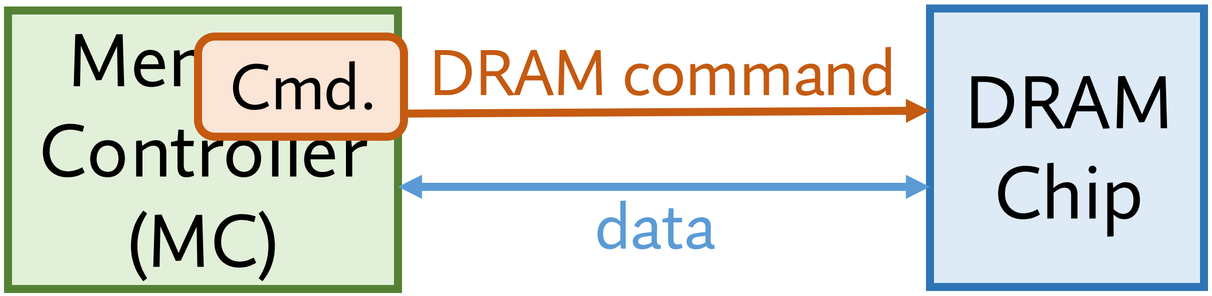
SMD Outline

1. **Motivation**
2. Self-Managing DRAM (SMD)
3. Use Cases
4. Evaluations
5. Conclusion and Takeaways

DRAM Interface Status Quo



DRAM Interface is Rigid



orchestrates all DRAM operations

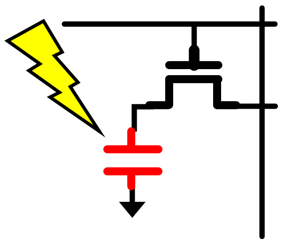
executes all DRAM commands

- by issuing *DRAM commands*

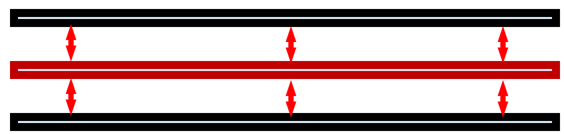
DRAM interface is completely controlled by one side

DRAM Maintenance Mechanisms

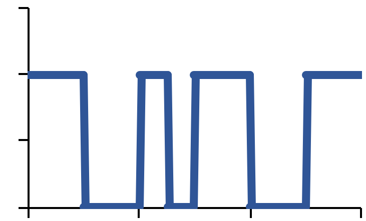
Data Retention



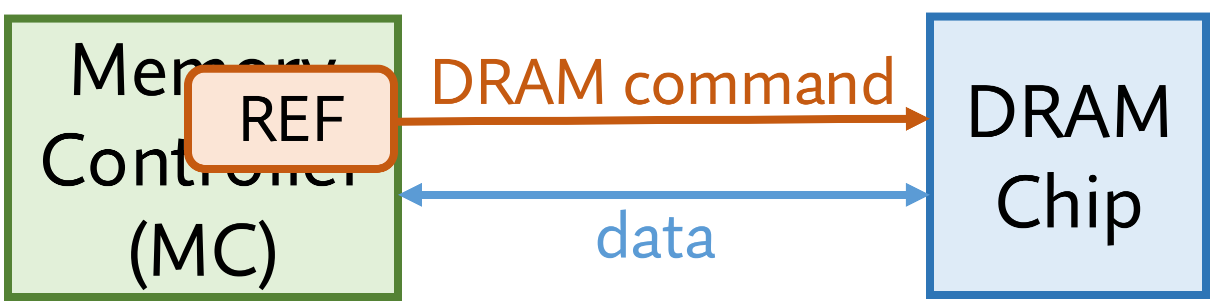
Read Disturbance (e.g., RowHammer)



Variable Retention Time



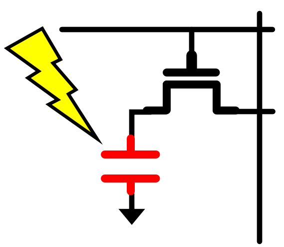
- DRAM failure modes necessitate maintenance mechanisms
- Perform operations to maintain DRAM data integrity
  - A prominent example is periodic refresh



New Maintenance Mechanisms are Needed

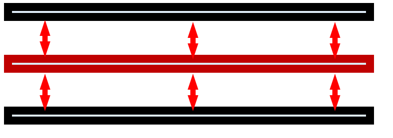
- Density scaling increases memory error rates

Data Retention



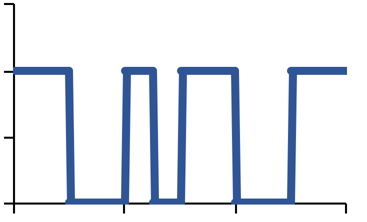
shrinking capacitance
worsening leakage

Read Disturbance (e.g., RowHammer)



increasing interference

Variable Retention Time

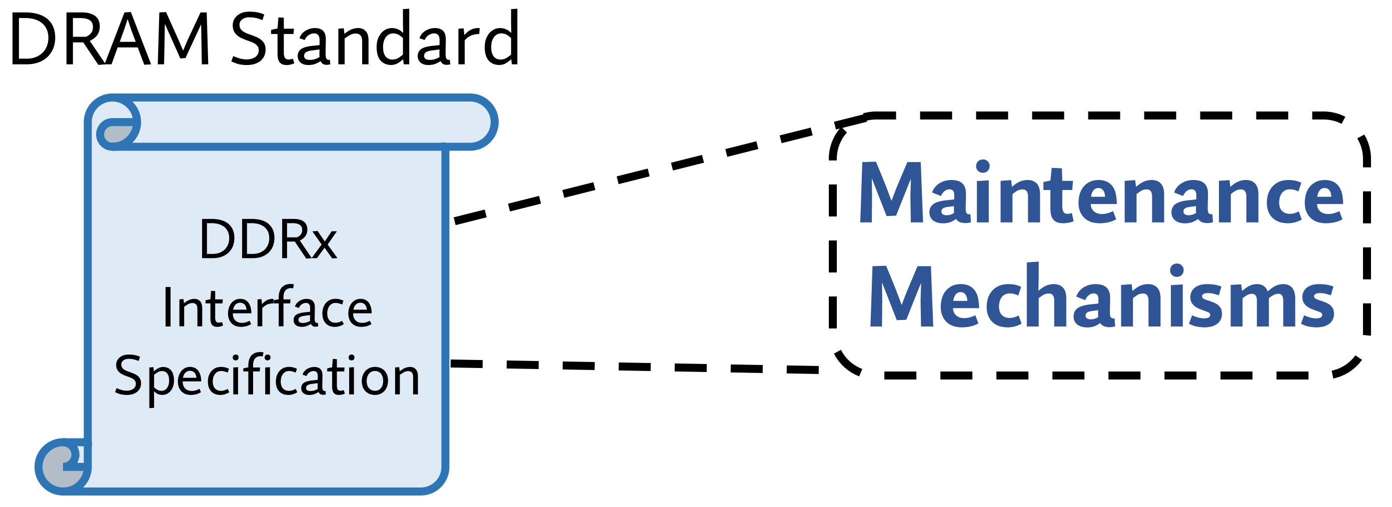


shrinking capacitance
worsening leakage

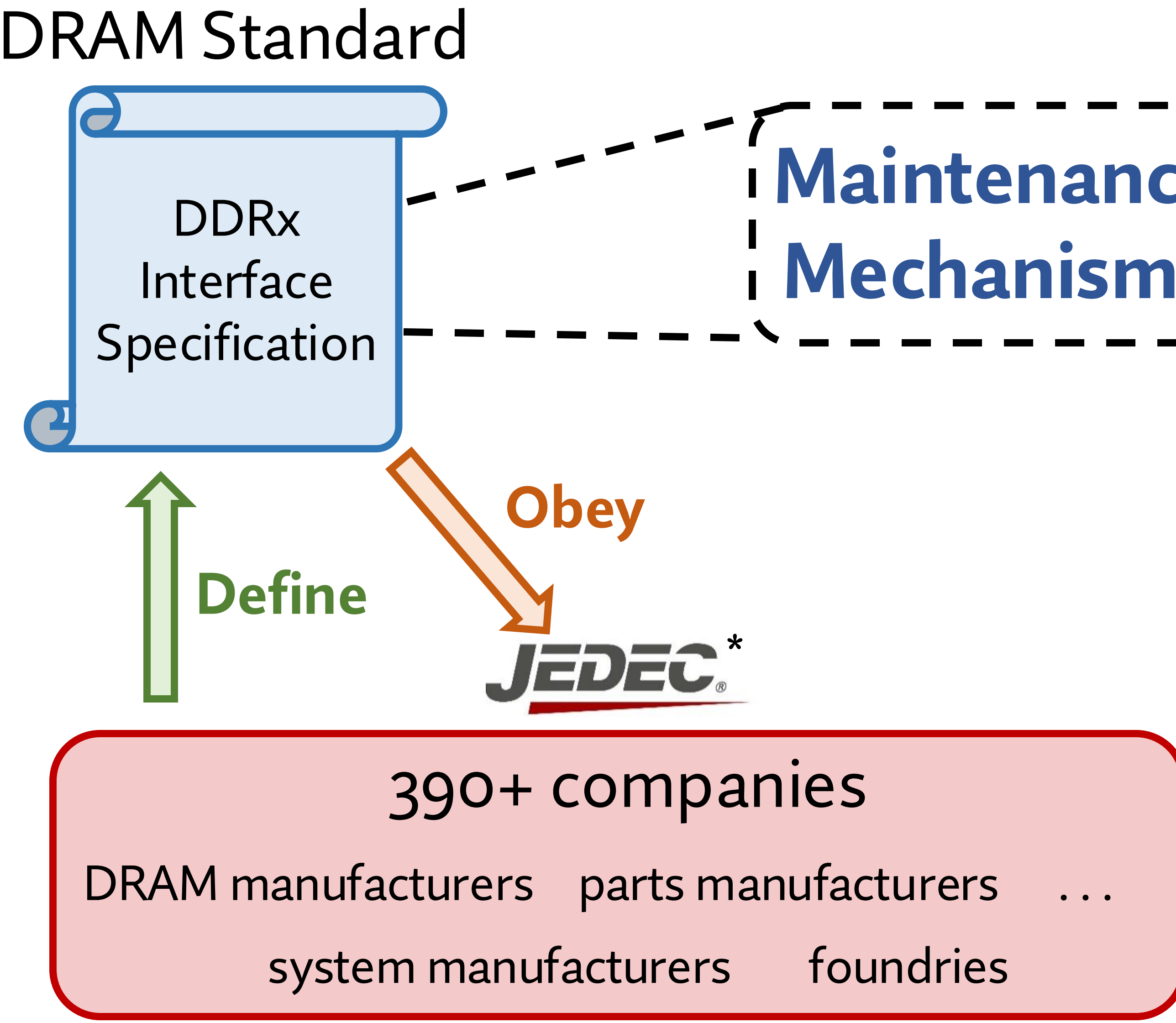
Continued DRAM process scaling necessitates
new efficient maintenance mechanisms

[Patel+, MICRO'21]

DRAM Standard Interface Specification

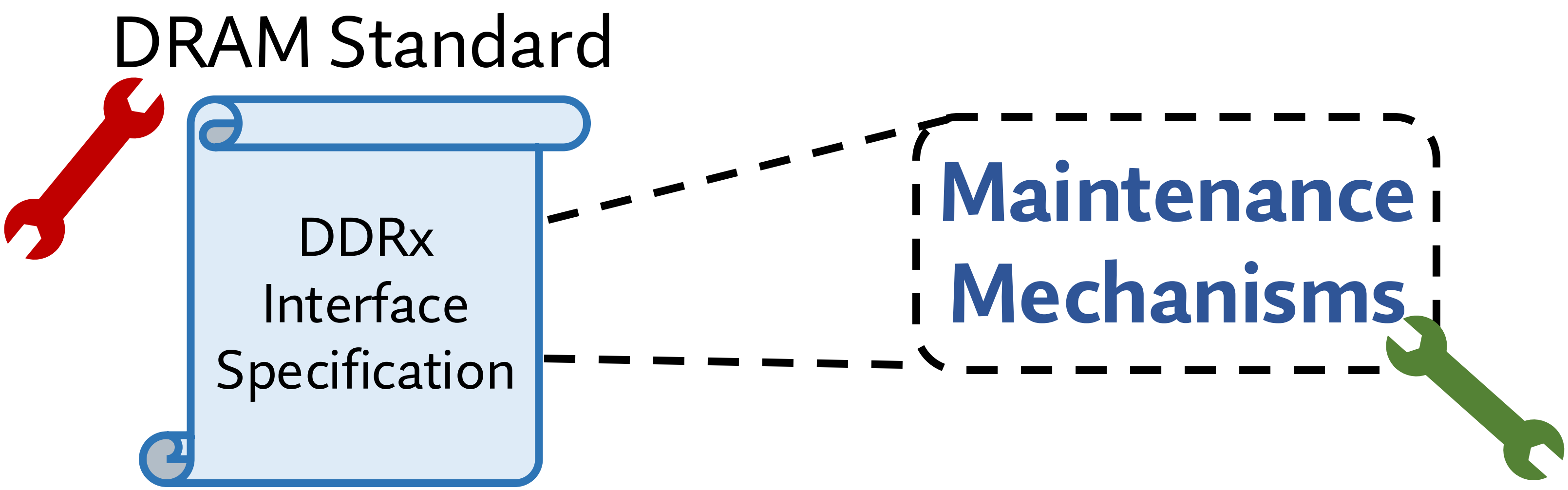


DRAM Standard Body – JEDEC*



*Joint Electron Device Engineering Council

Barrier to New Maintenance Mechanisms

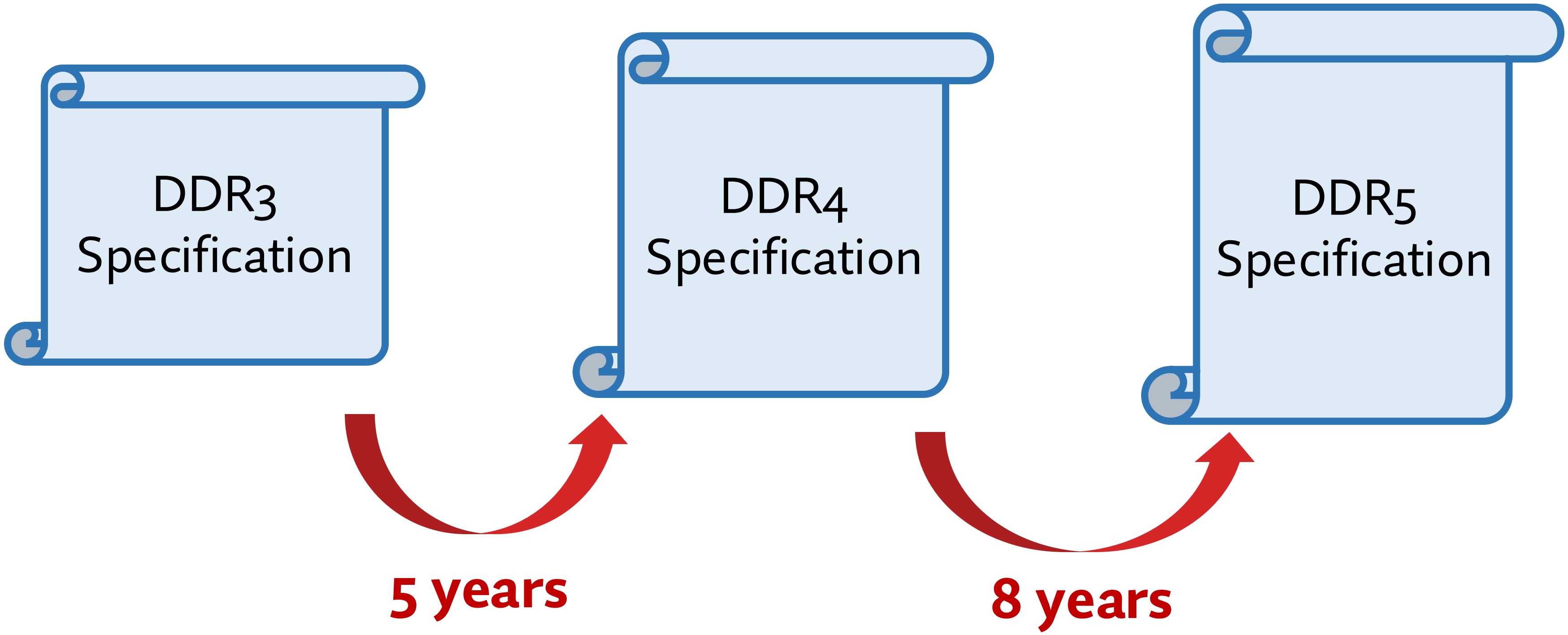


- Adding new or modifying existing maintenance mechanisms requires lengthy modifications to

1. **DRAM specifications** and
2. other system components that obey the specifications

DRAM interface is rigid

DRAM Specifications Evolve Slowly



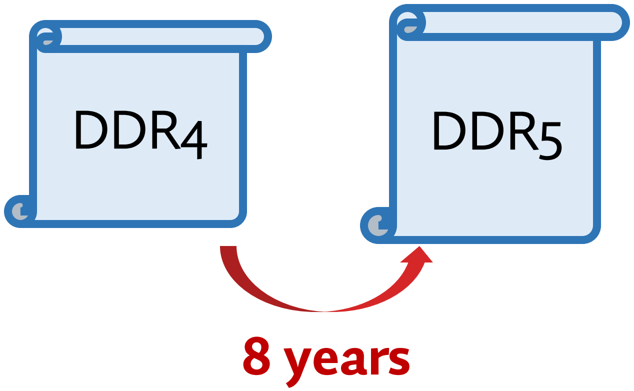
Multi-year effort by the JEDEC committee

Introducing new maintenance operations
takes a long time

Recently-Introduced Maintenance Mechanisms

- DDR5 introduces three maintenance techniques

1. **Same Bank Refresh**
   - improve bank-level parallelism
2. **Refresh Management (RFM)**
   - improve robustness
3. **In-DRAM ECC scrubbing**
   - improve error tolerance



These improvements could have been **released earlier**

Problem and Our Goal

Problem

Introducing new maintenance operations
takes a long time

Our Goal

Ease and **accelerate** the process of implementing
new efficient in-DRAM maintenance operations

DRAM Access and Maintenance

- Categorize DRAM operations into **two** classes:

1 Access

- Performed to serve memory requests
- Uses information available *only* to the memory controller
  - e.g., load *address*, store *data*

2 Maintenance

- Performed to maintain DRAM data integrity
- Uses information available *only* to the DRAM chip
  - e.g., in-DRAM row activation counter

DRAM Access and Maintenance

- Categorize DRAM operations into **two** classes:

Key observation:
A DRAM chip could **"maintain" itself**

2 Maintenance

- Performed to maintain DRAM data integrity
- Uses information available *only* to the DRAM chip
  - e.g., in-DRAM row activation counter

A DRAM Chip Should Maintain Itself

- Two benefits of DRAM chip "autonomously" performing maintenance operations

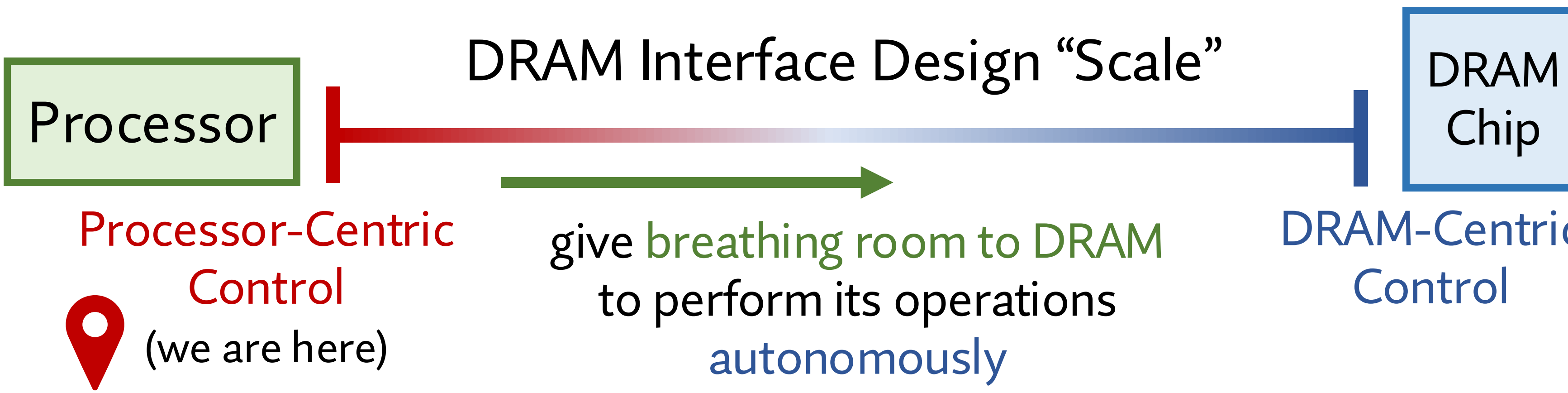
1. Maintenance mechanisms can be implemented more easily and rapidly
   - DRAM interface modifications are not required

2. Enable DRAM manufacturers with breathing room to perform architectural optimizations *without* exposing DRAM-internal proprietary information

Solution Approach

Enable autonomous maintenance operations

- **Key Challenge:** DRAM interface is too rigid to accommodate autonomous in-DRAM maintenance operations



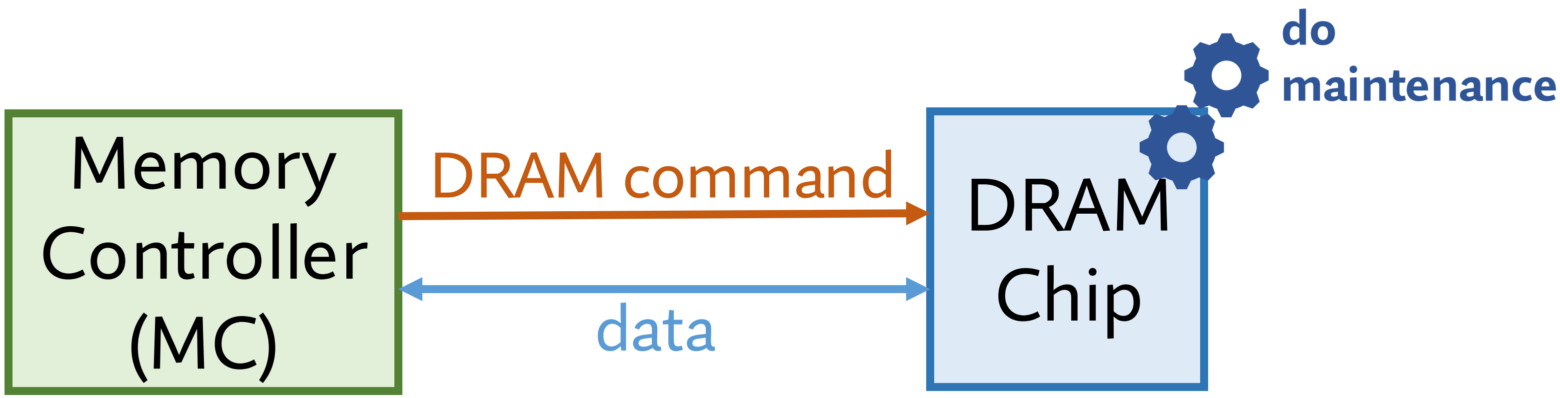
- **Goal:** Make a simple, one-time change to the DRAM interface that enables autonomous maintenance operations

SMD Outline

1. Motivation
2. **Self-Managing DRAM (SMD)**
3. Use Cases
4. Evaluations
5. Conclusion and Takeaways

SMD Key Idea: Autonomous Maintenance

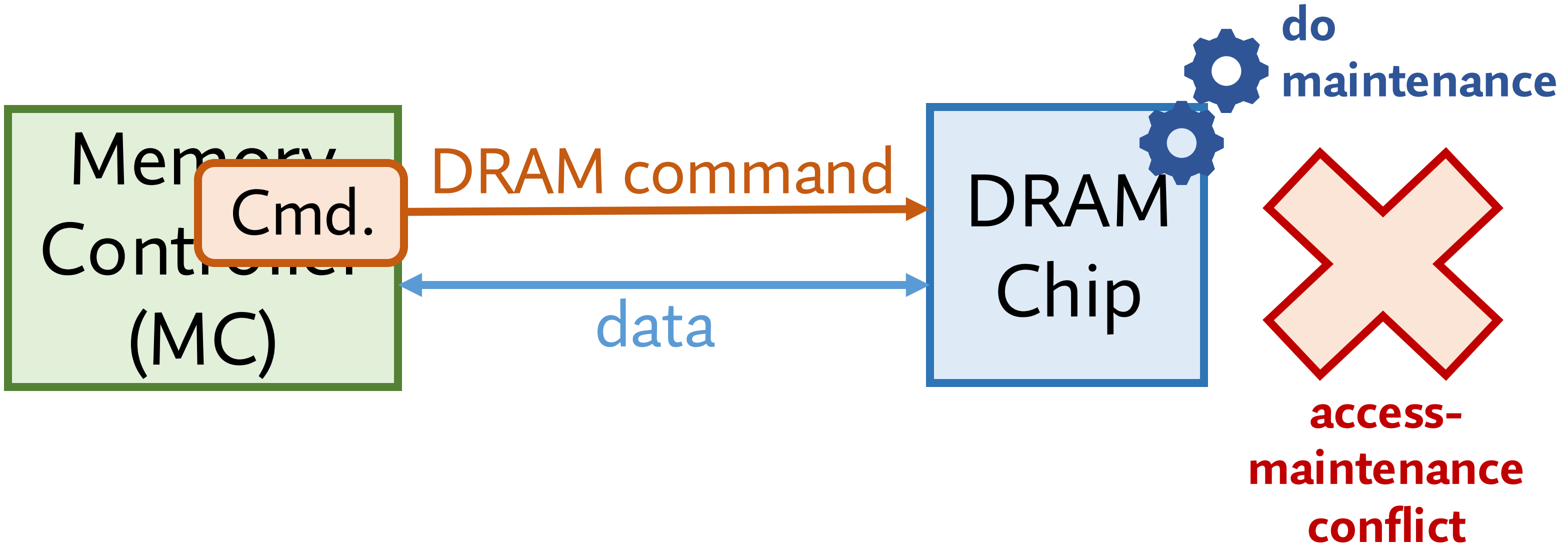
DRAM chip controls in-DRAM maintenance operations



Enable implementing **new maintenance mechanisms** **without** modifying the standard and exposing **DRAM-internal proprietary** information

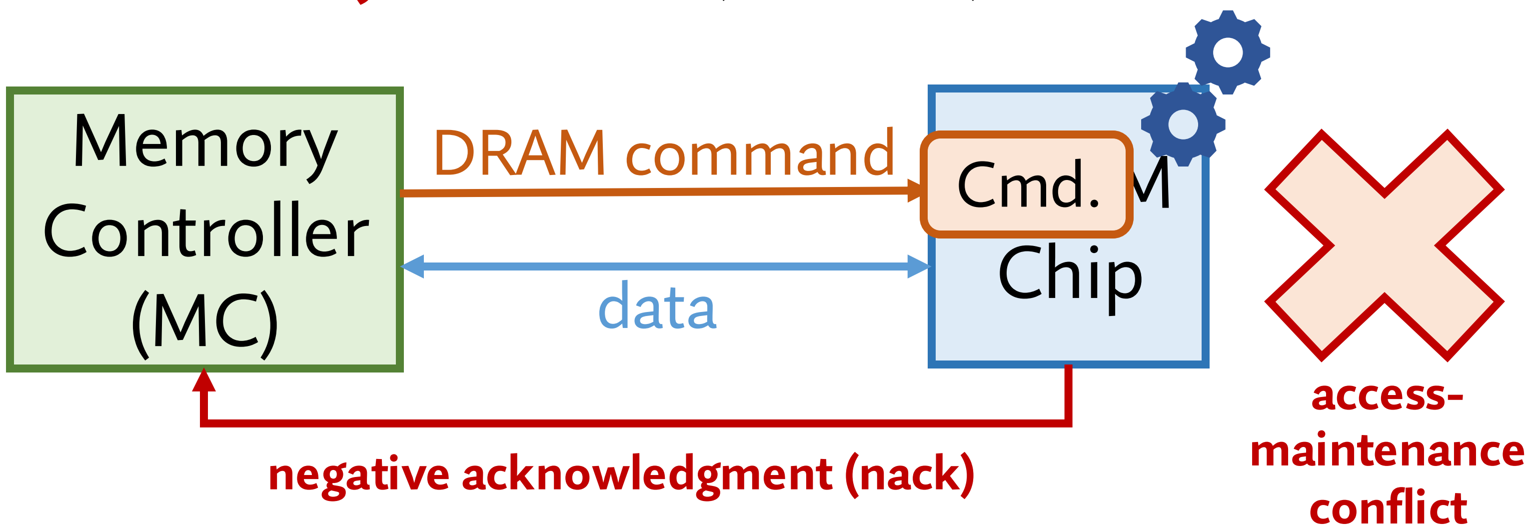
Access-Maintenance Conflicts

- **Problem:** Access-maintenance conflict



SMD Key Mechanism

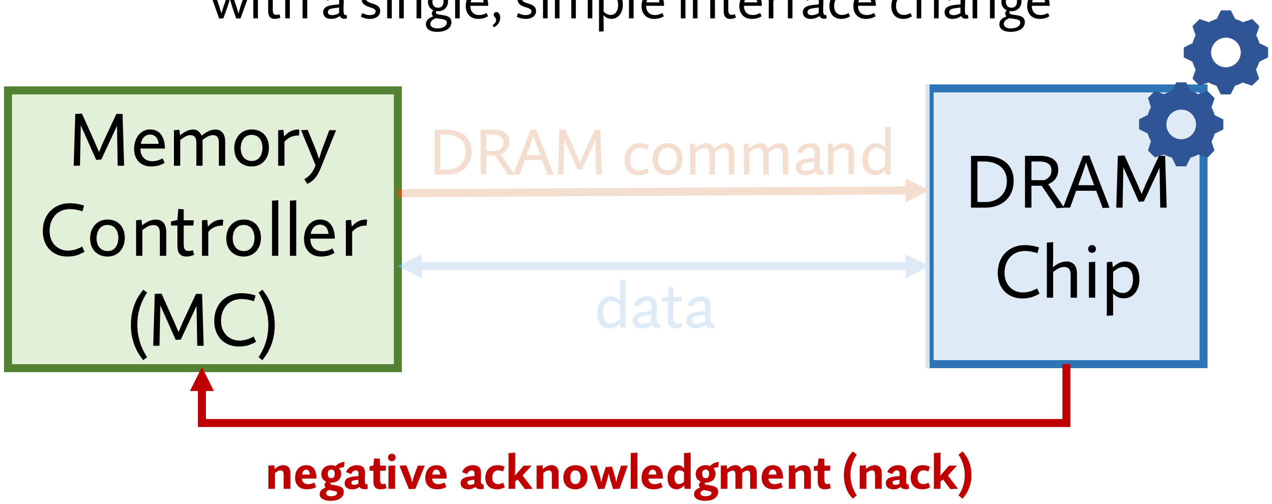
- **Problem:** Access-maintenance conflict
- **Key mechanism: Reject** access (activate) commands



SMD Key Contribution

DRAM chip controls in-DRAM maintenance operations

with a single, simple interface change



orchestrates all access operations

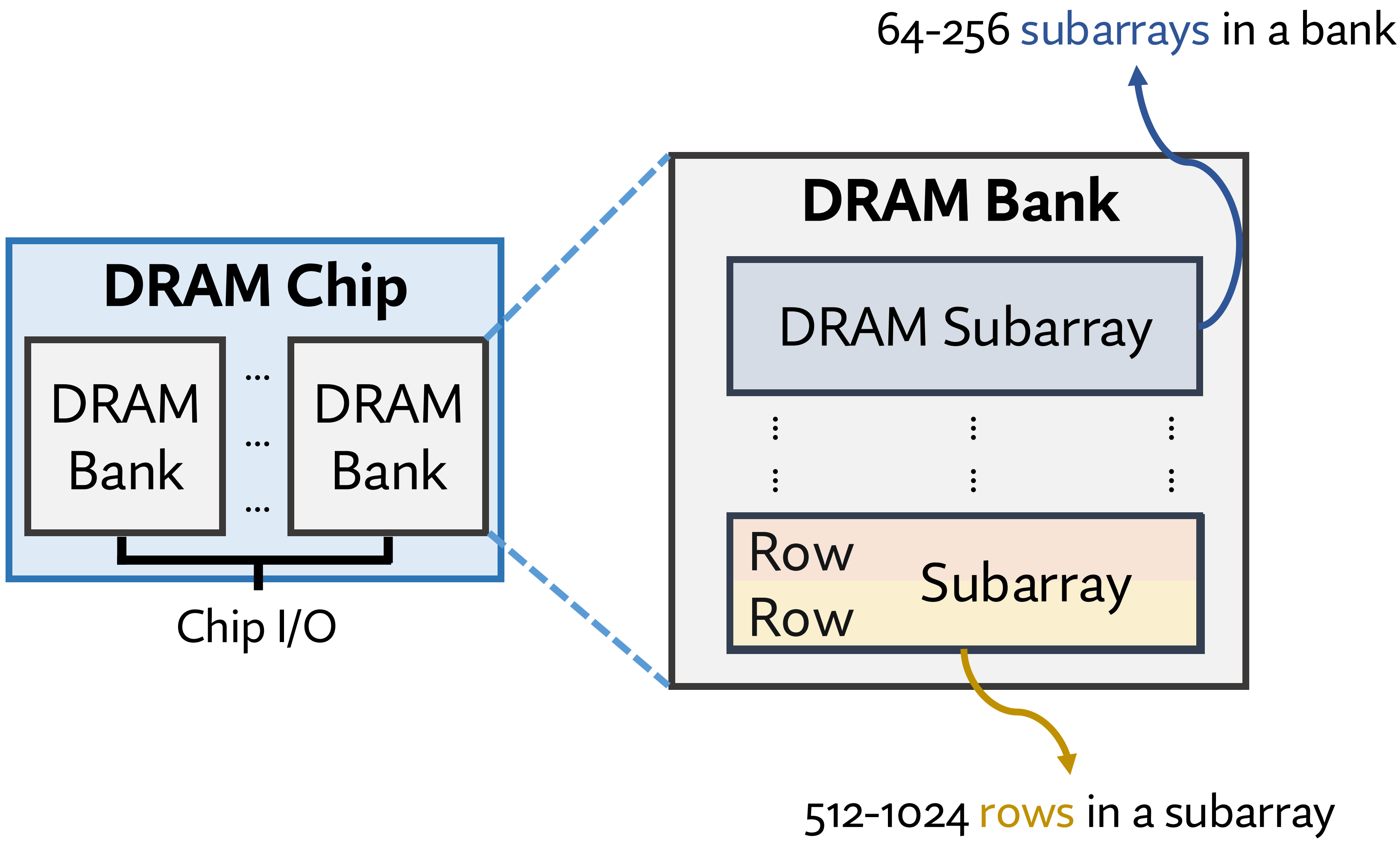
can now perform its own maintenance autonomously

Partition the work nicely between the memory controller and the DRAM chip

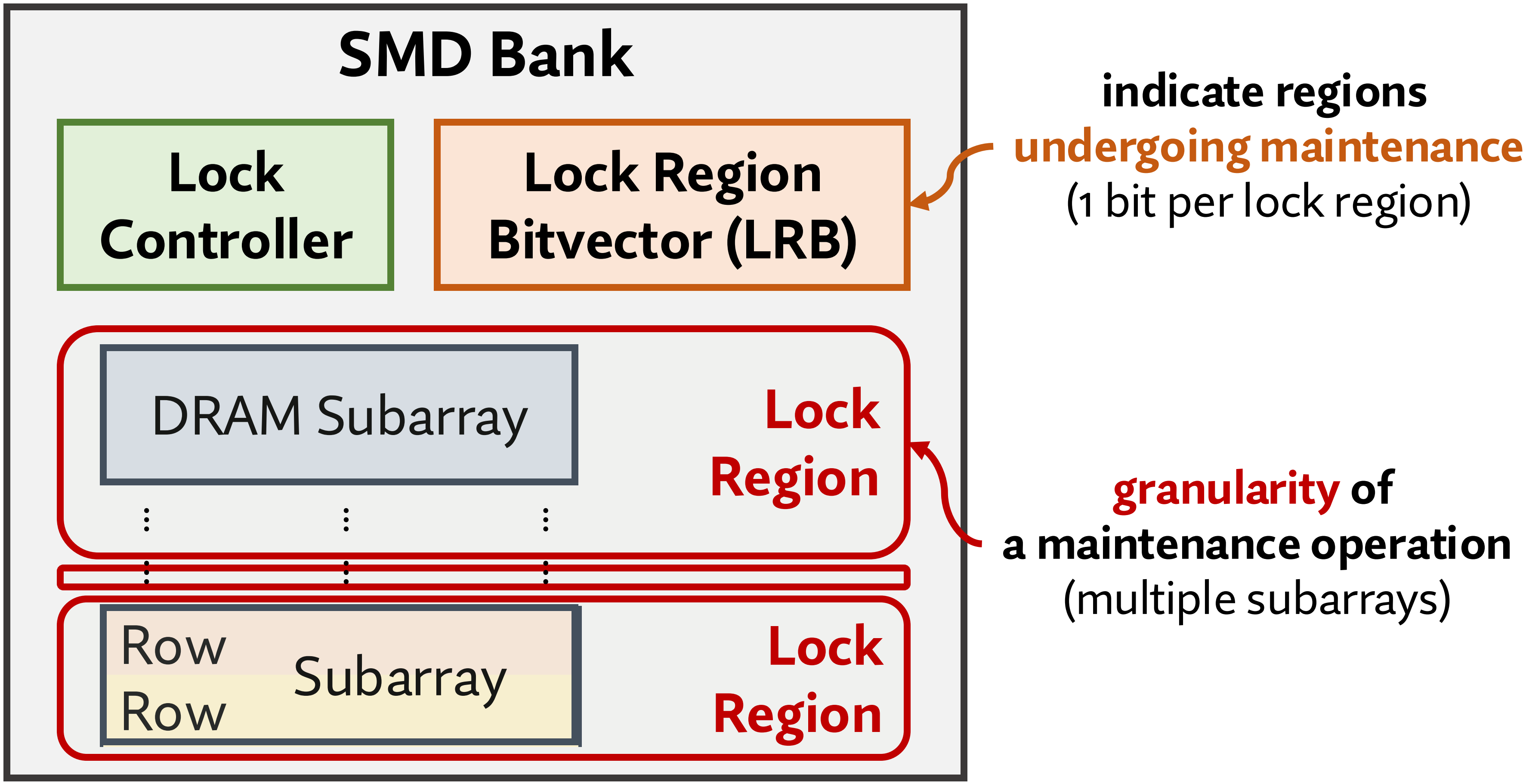
Deeper Look at SMD

1 **SMD Bank Organization**

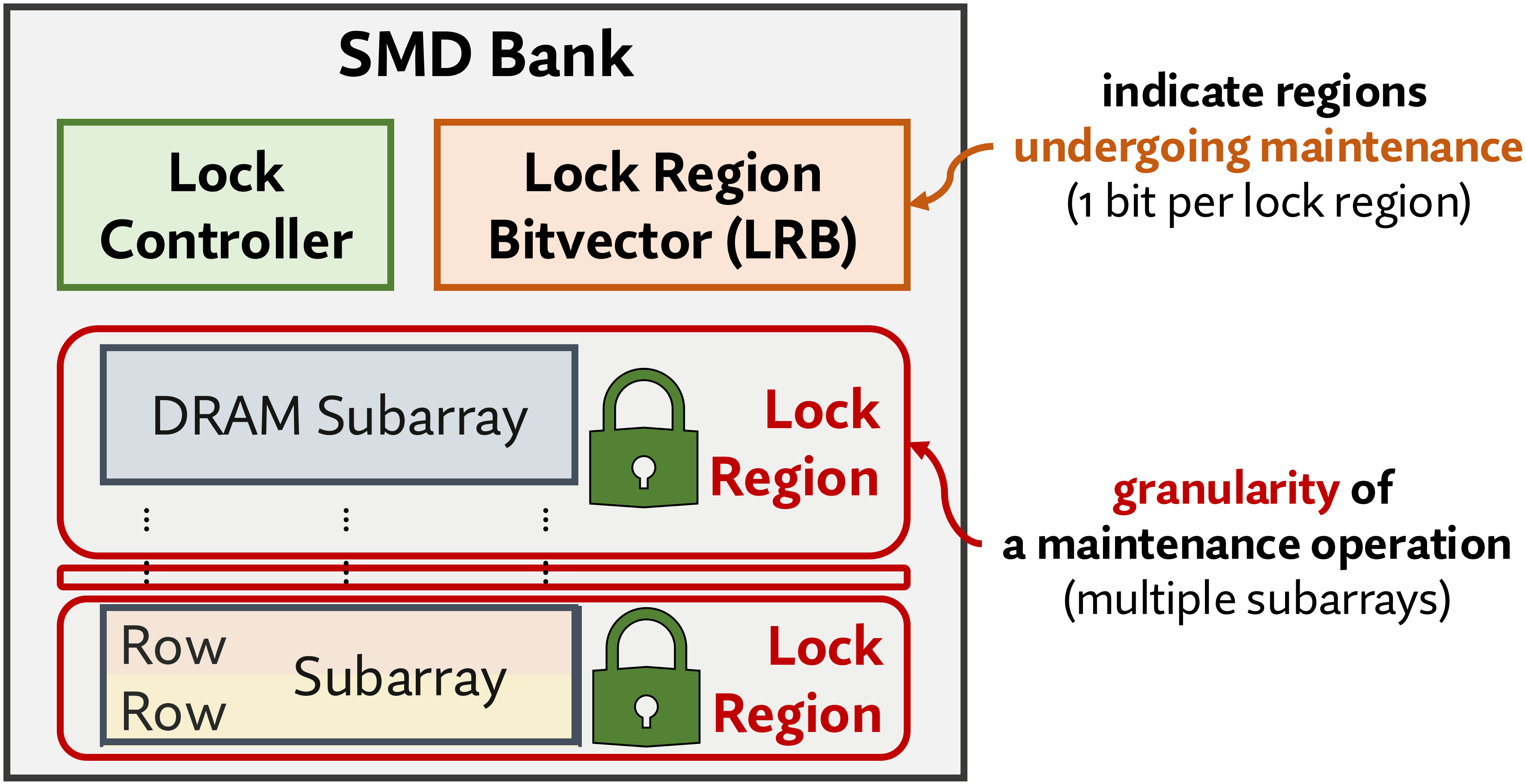
DRAM Chip Organization



DRAM Bank with SMD



Locking Regions for Maintenance



Lock a region before starting maintenance

Deeper Look at SMD

1 SMD Bank Organization

2 Region Locking Mechanism

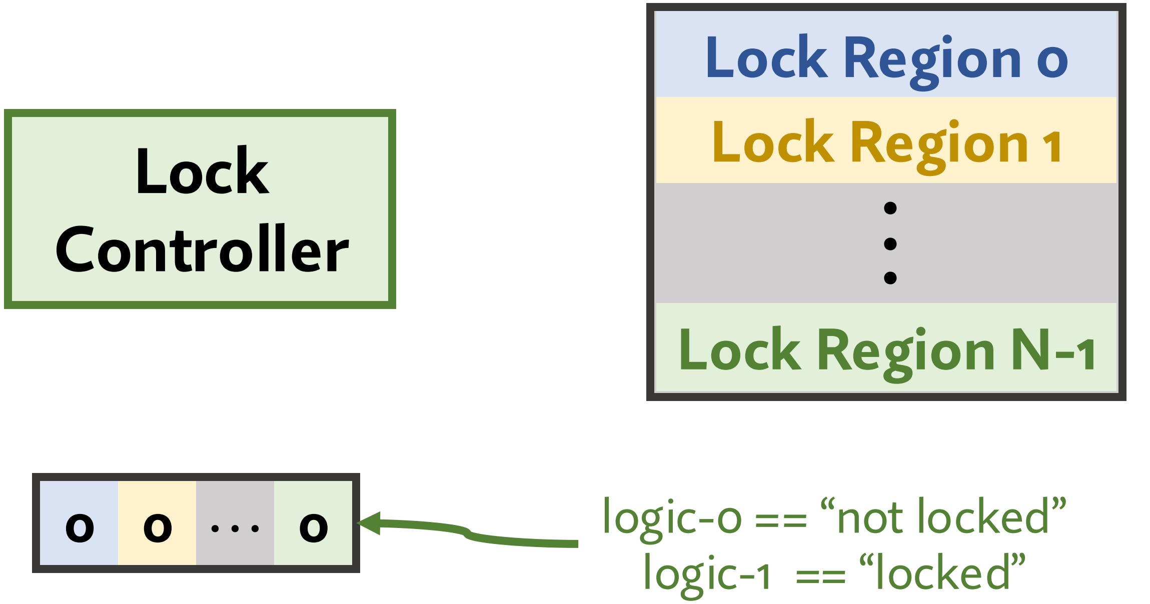
Summary of Region Locking Mechanism

1. Maintenance operation "locks" a region
2. Memory controller can access "not locked" regions
3. Access to locked region receives negative ack
4. Locked region released at the end of maintenance

Summary of Region Locking Mechanism

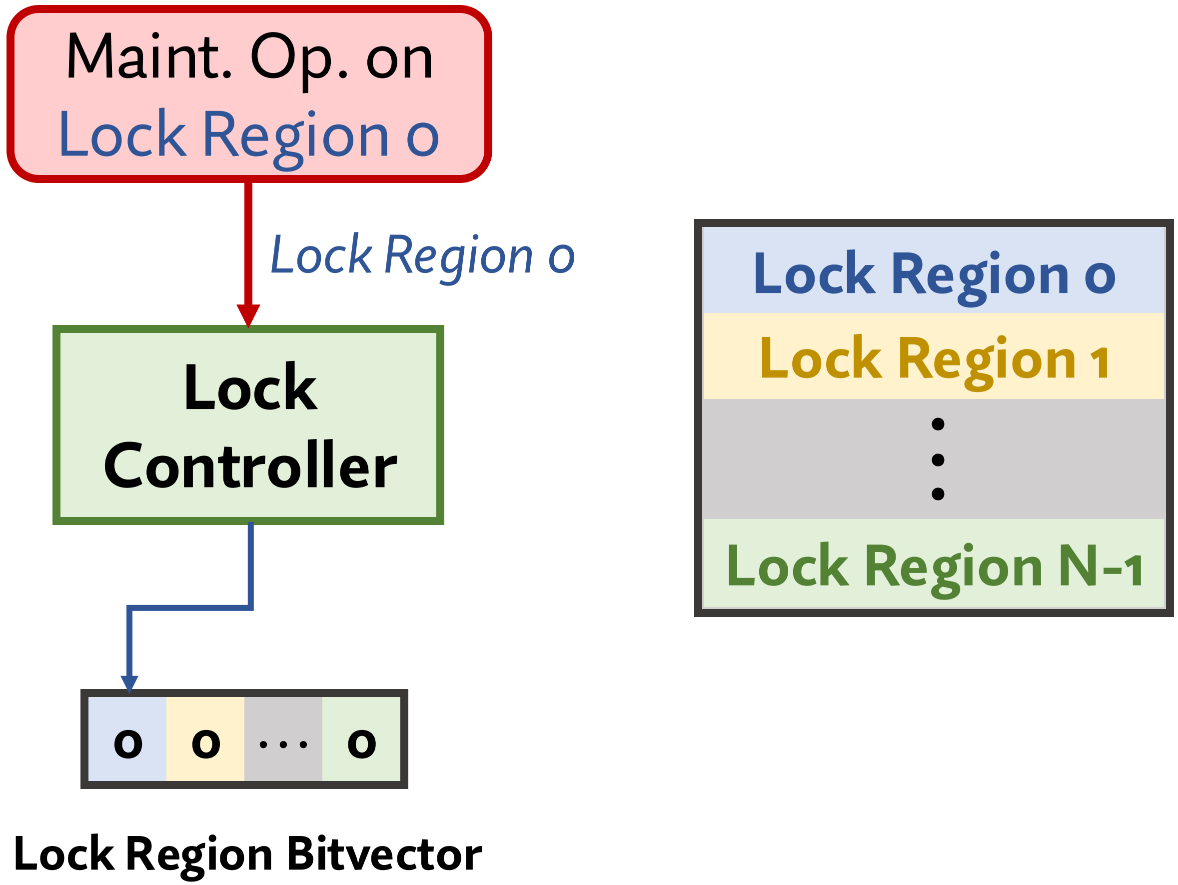
1. Maintenance operation "locks" a region
2. Memory controller can access "not locked" regions
3. Access to locked region receives negative ack
4. Locked region released at the end of maintenance

Locking a Region

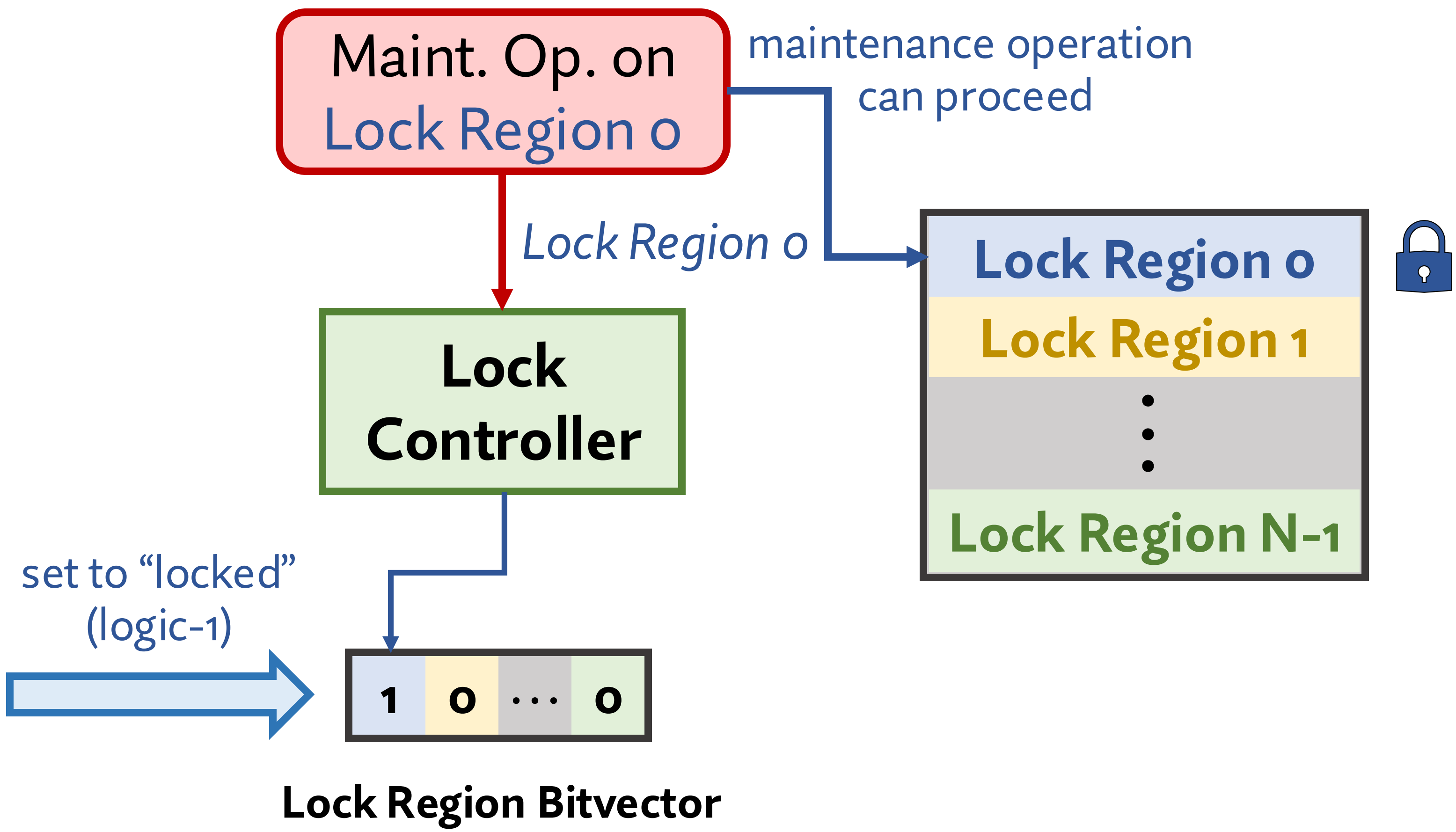


Lock Region Bitvector

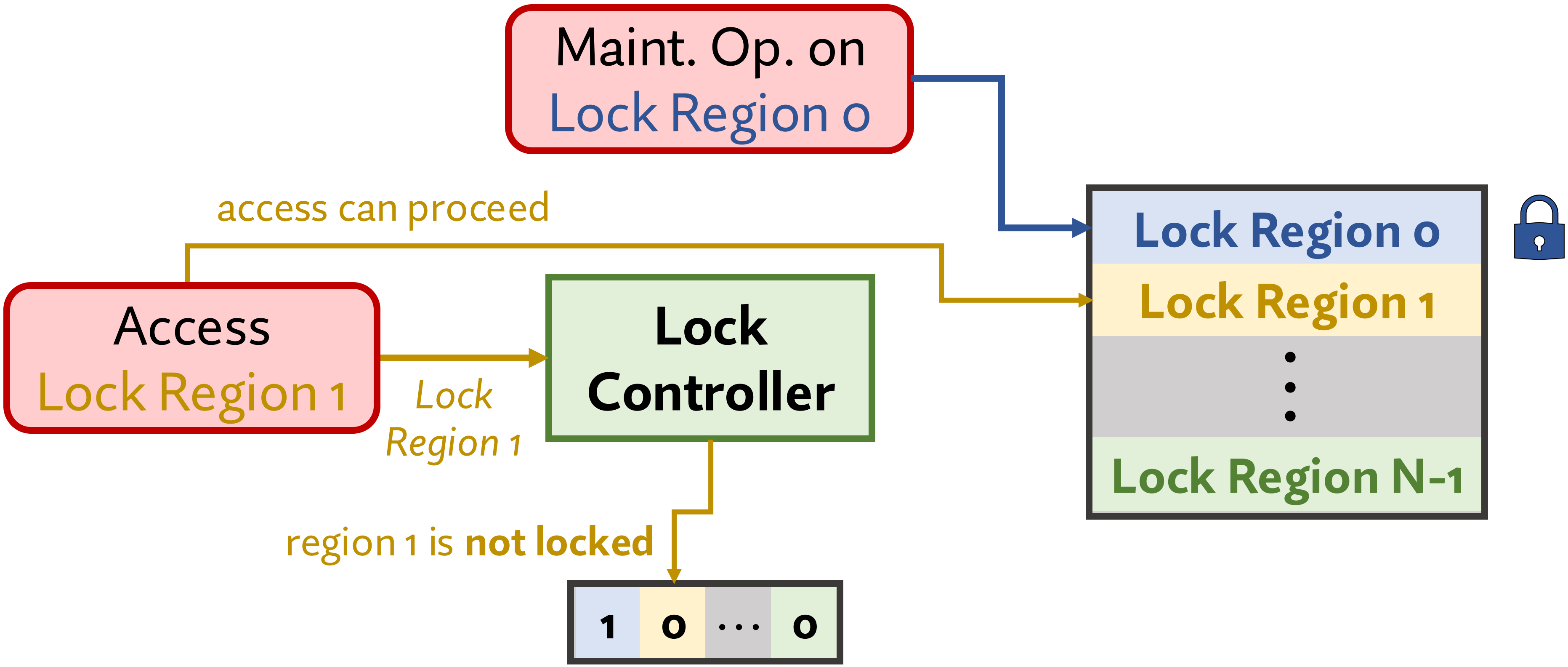
Locking a Region



Locking a Region

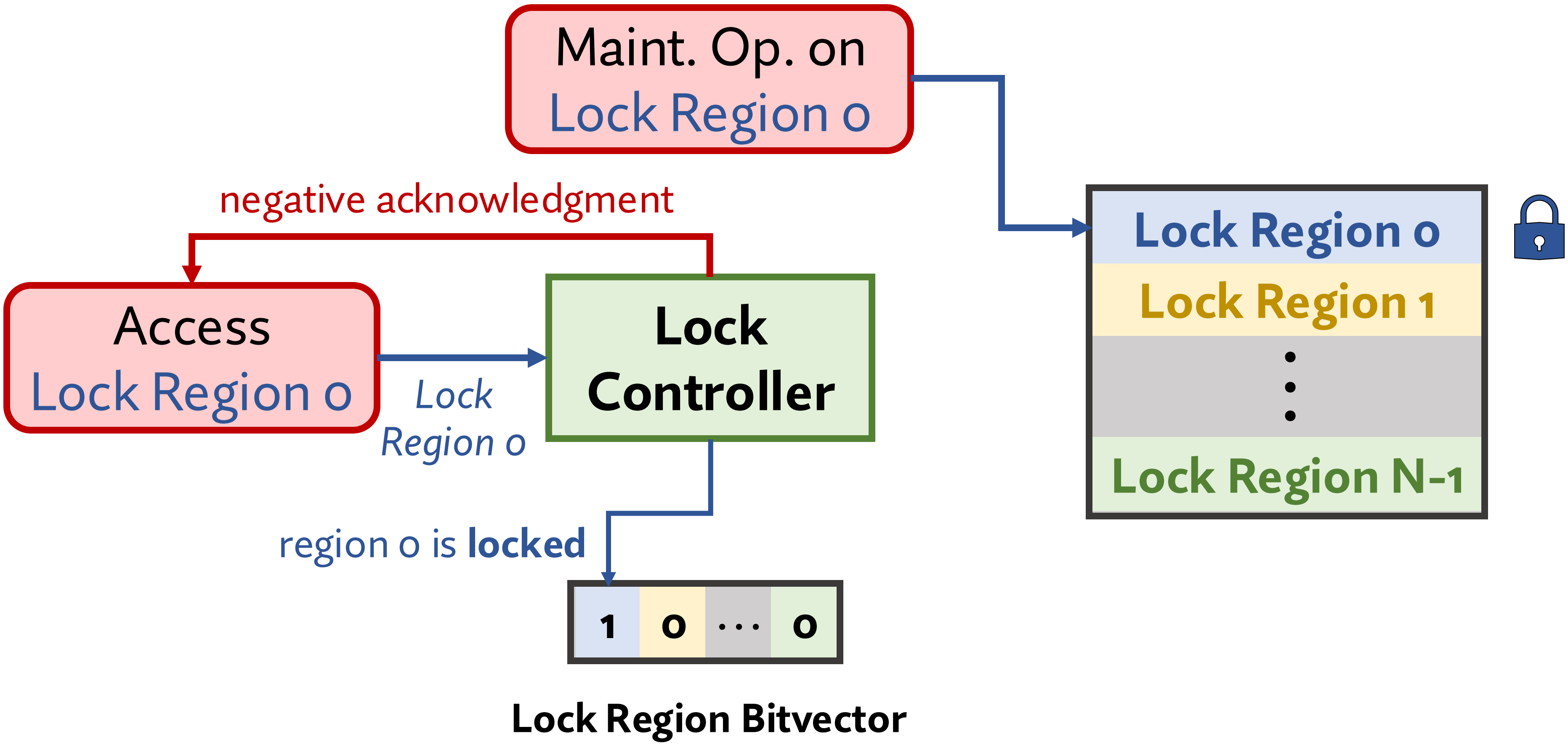


Accessing a Not Locked Region

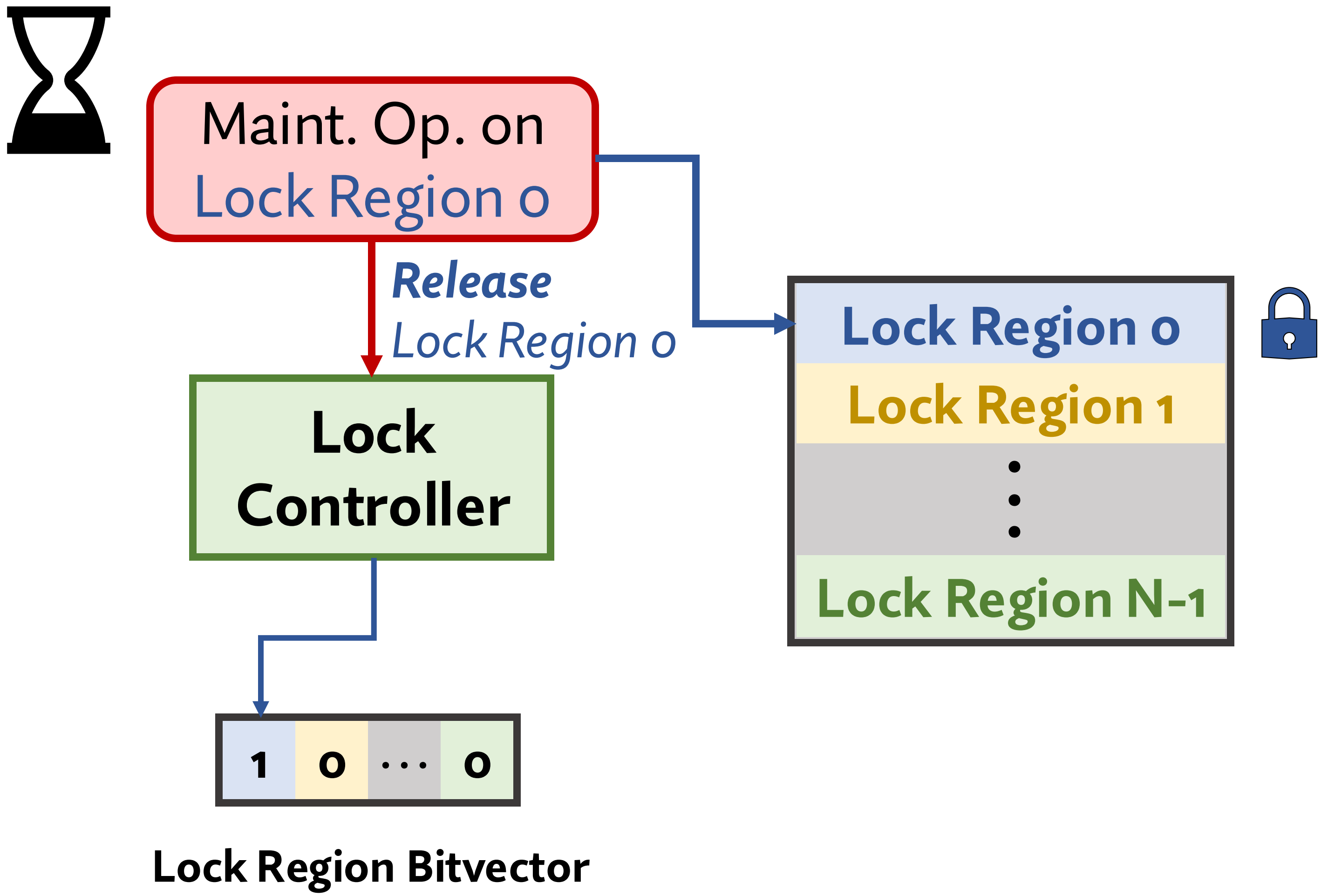


Lock Region Bitvector

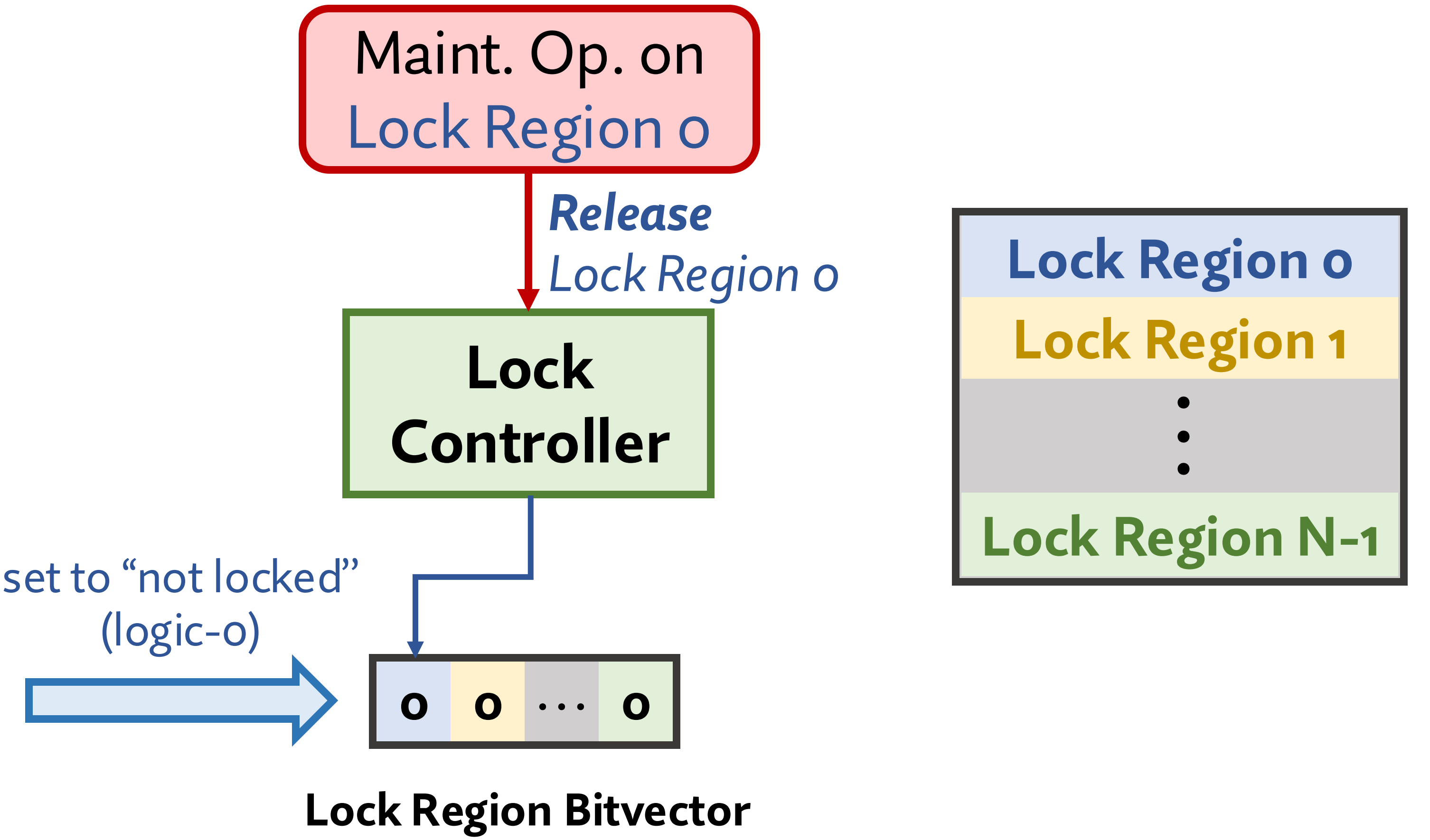
Accessing a Locked Region



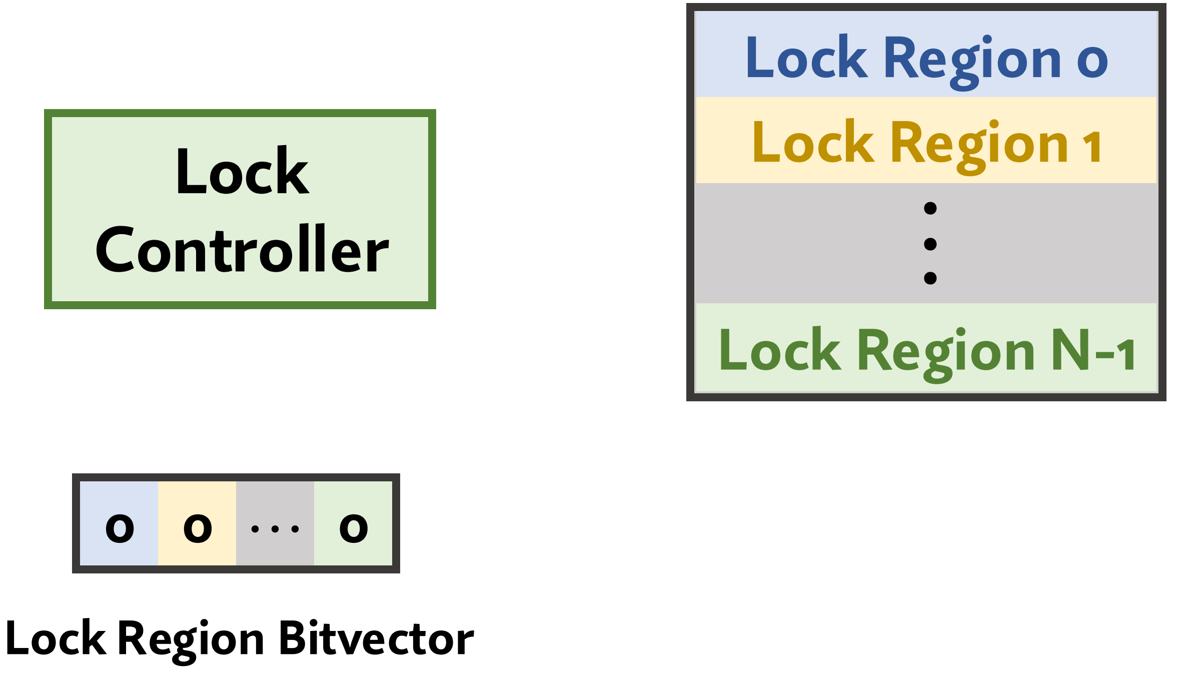
Releasing a Region



Releasing a Region



Releasing a Region



Deeper Look at SMD

1. SMD Bank Organization

2. Region Locking Mechanism

3. **Controlling an SMD Chip**

Summary of SMD Chip Control

1. Activate commands can get rejected (negative ack)
2. Memory controller retries rejected commands
3. Memory controller can attempt to access other lock regions
4. SMD chip and memory controller ensure forward progress for memory requests

Summary of SMD Chip Control

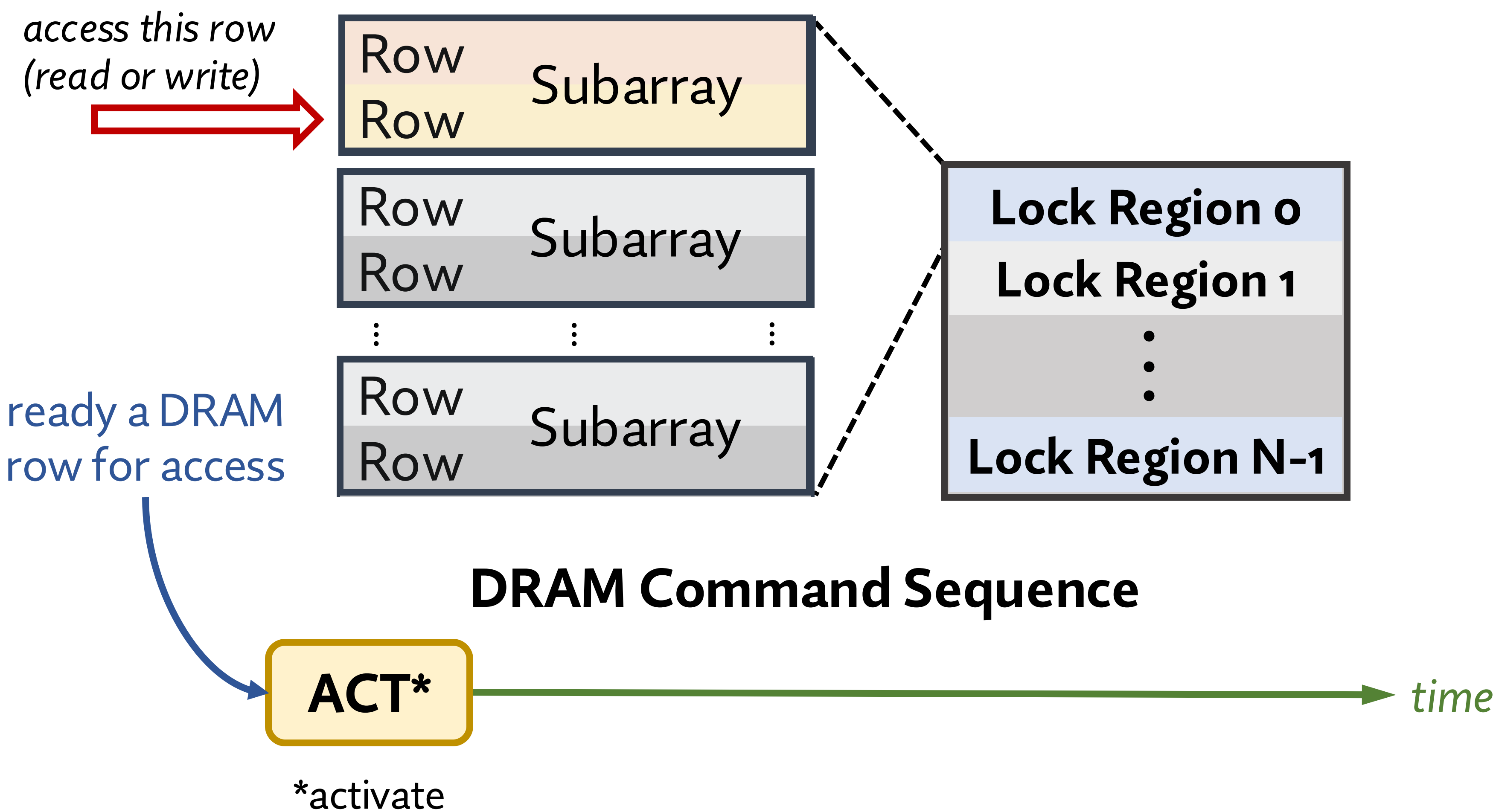
1. Activate commands can get rejected (negative ack)

2. Memory controller retries rejected commands

3. Memory controller can attempt to access other lock regions

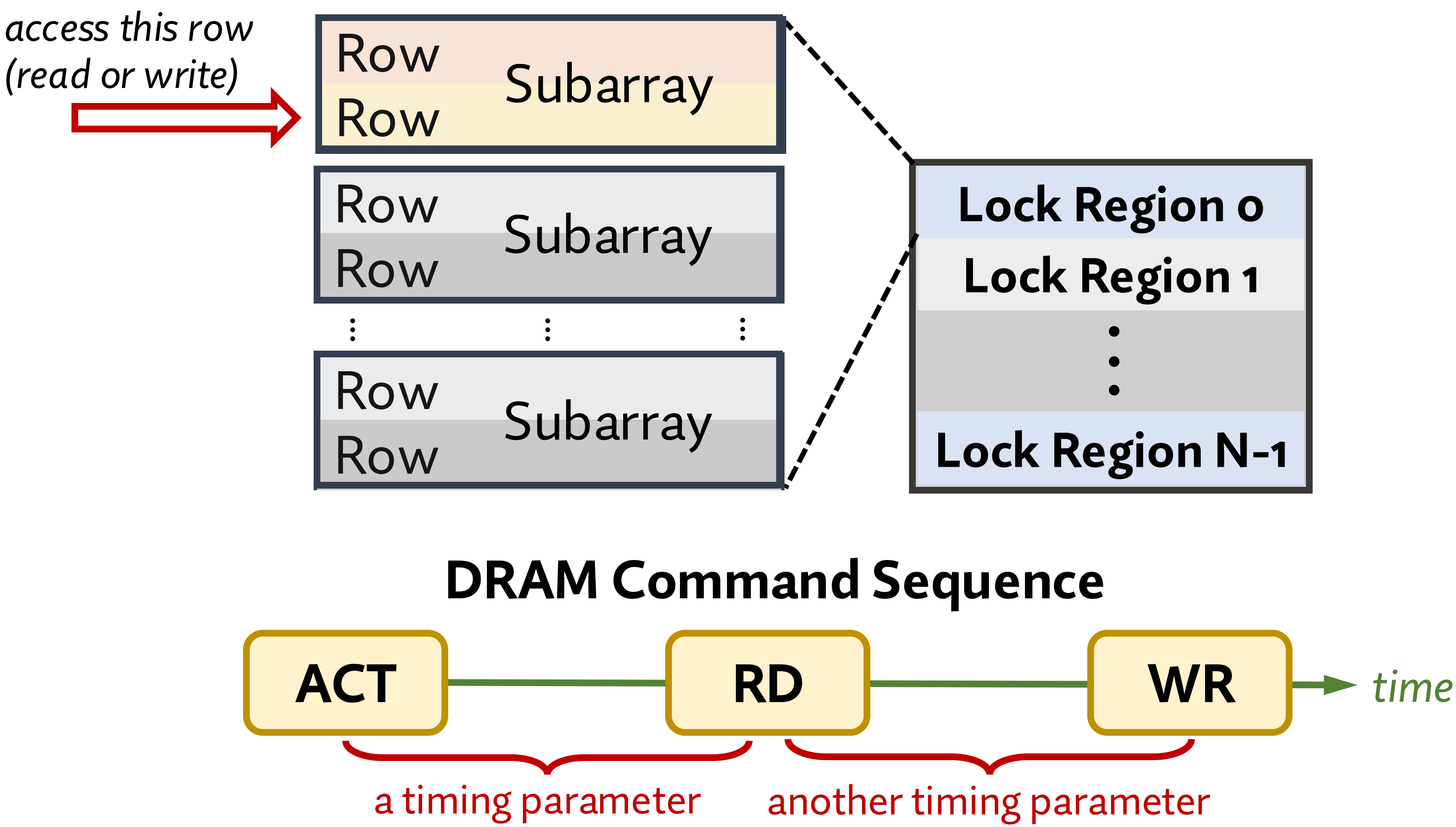
4. SMD chip and memory controller ensure forward progress for memory requests

DRAM Control – The “Activate” Command

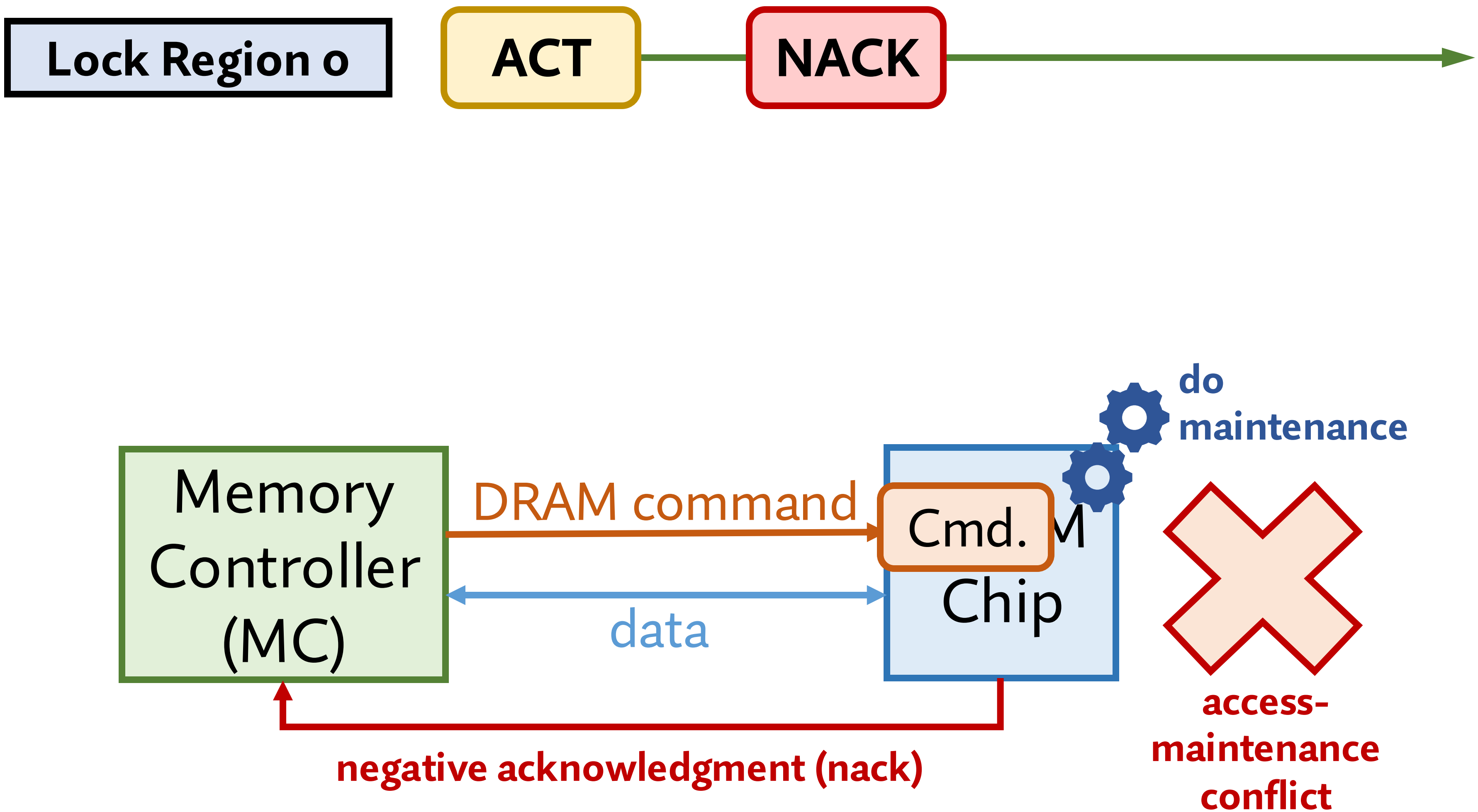


DRAM Control – Timing Parameters

- Timing parameter: Minimum delay between two commands

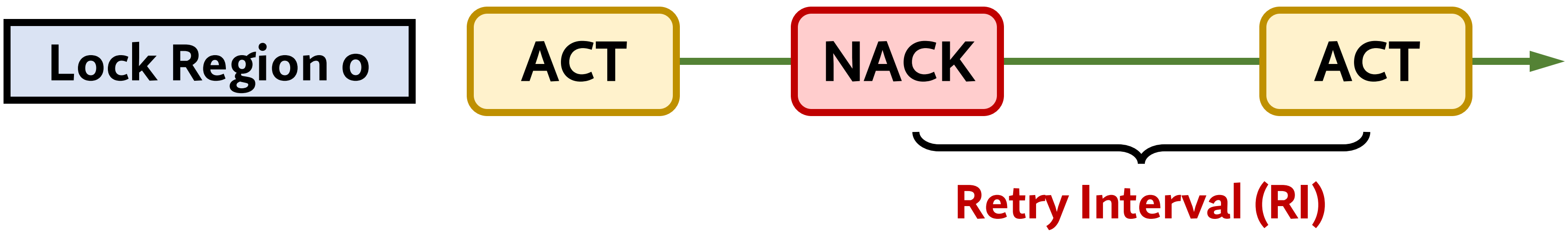


SMD Control – Handling a Rejection



SMD Control – Handling a Rejection

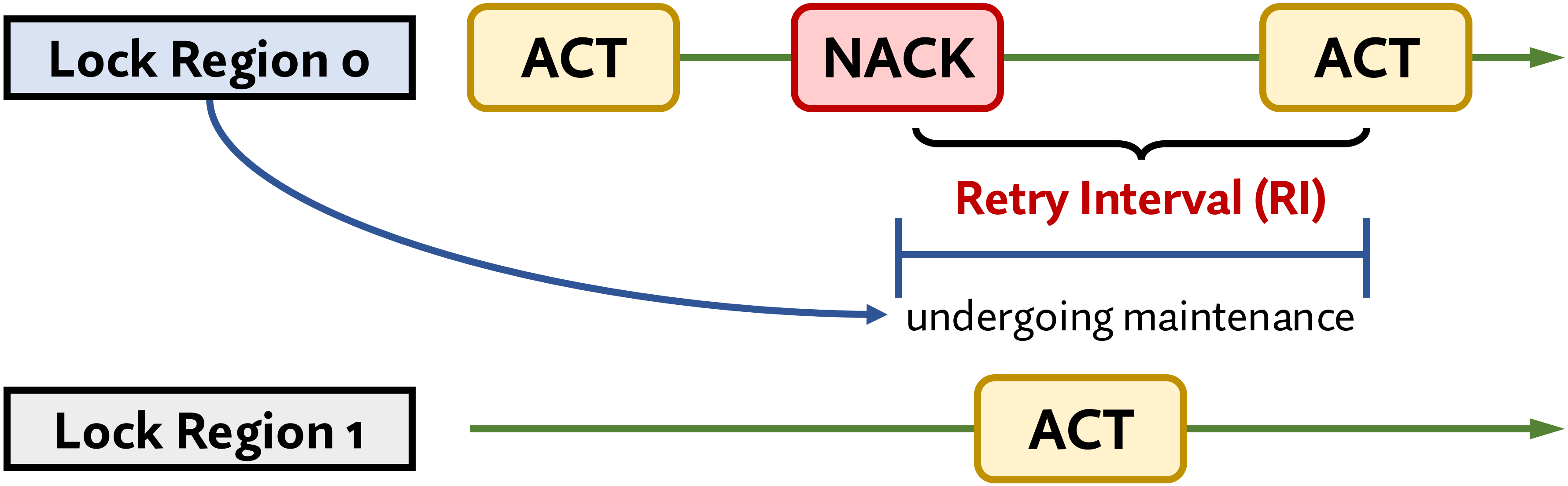
Key idea: Introduce a new timing parameter



- **Retry** ACT **every retry interval** until ACT is **not rejected**

Maintenance-Access Parallelization

Key idea: Introduce a new timing parameter



Overlap RI latency with a **useful operation** to another lock region

building on basic design in SALP
[Kim+, ISCA'12] [Zhang+, HPCA'14] [Chang+, HPCA'14]

More details in our paper
https://arxiv.org/pdf/2207.13358

Proof of Forward Progress

https://arxiv.org/pdf/2207.13358

Self-Managing DRAM: A Low-Cost Framework for Enabling Autonomous and Efficient DRAM Maintenance Operations

Hasan Hassan† Ataberk Olgun† A. Giray Yağlıkçı Haocong Luo Onur Mutlu

ETH Zürich

The memory controller is in charge of managing DRAM maintenance operations (e.g., refresh, RowHammer protection, memory scrubbing) to reliably operate modern DRAM chips. Implementing new maintenance operations often necessitates modifications in the DRAM interface, memory controller, and potentially other system components. Such modifications are only possible with a new DRAM standard, which takes a long time to develop, likely leading to slow progress in the adoption of new architectural techniques in DRAM chips.

We propose a new low-cost DRAM architecture, Self-Managing DRAM (SMD), that enables autonomous in-DRAM maintenance operations by transferring the responsibility for controlling maintenance operations from the memory controller to the SMD chip. To enable autonomous maintenance operations, we make a single, simple modification to the DRAM interface, such that an SMD chip rejects memory controller accesses to DRAM regions (e.g., a subarray or a bank) under maintenance, while allowing memory accesses to other DRAM regions. Thus, SMD enables

tion [12, 18, 47–122], and 3) memory scrubbing [17, 123–135].[1] New DRAM chip generations necessitate making existing maintenance operations more aggressive (e.g., lowering the refresh period [119, 136, 137]) and introducing new types of maintenance operations (e.g., targeted refresh [64, 66, 138], DDR5 RFM [119], and PRAC [119] as RowHammer defenses).[2]

Two problems likely hinder the adoption of effective and efficient maintenance mechanisms in modern and future DRAM-based computing systems. First, it is difficult to modify existing maintenance mechanisms and introduce new maintenance operations because doing so often necessitates changes to the DRAM interface, which takes a long time (due to various issues related to standardization and agreement across many vendors with conflicting interests [4, 6]). Second, it is challenging to keep the overhead of DRAM maintenance mechanisms low as DRAM reliability characteristics worsen and DRAM chips require more aggressive maintenance operations. We expand on the two problems in the next two paragraphs.

SMD Outline

1. Motivation

2. Self-Managing DRAM (SMD)

3. **Use Cases**

4. Evaluations

5. Conclusion and Takeaways

SMD-Based Maintenance Mechanisms

Demonstrate the usefulness and versatility of SMD

1. **Fixed-Rate Refresh (SMD-FR)**
2. **Deterministic RowHammer Protection (SMD-DRP)**
3. **Memory Scrubbing (SMD-MS)**

https://arxiv.org/pdf/2207.13358

Evaluate

Variable-Rate Refresh
Probabilistic RowHammer Protection

Discuss

Online Error Profiling
Power Management
Processing in/near Memory
...

SMD-Based Maintenance Mechanisms

Demonstrate the usefulness and versatility of SMD

1 **Fixed-Rate Refresh (SMD-FR)**

2 **Deterministic RowHammer Protection (SMD-DRP)**

3 **Memory Scrubbing (SMD-MS)**

https://arxiv.org/pdf/2207.13358

Evaluate

Variable-Rate Refresh
Probabilistic RowHammer Protection

Discuss

Online Error Profiling
Power Management
Processing in/near Memory
...

SMD-Based Maintenance Mechanisms

Demonstrate the usefulness and versatility of SMD

1 Fixed-Rate Refresh (SMD-FR)

2 Deterministic RowHammer Protection (SMD-DRP)

3 Memory Scrubbing (SMD-MS)

https://arxiv.org/pdf/2207.13358

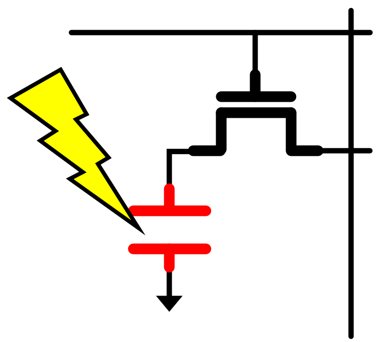
Evaluate

Variable-Rate Refresh
Probabilistic RowHammer Protection

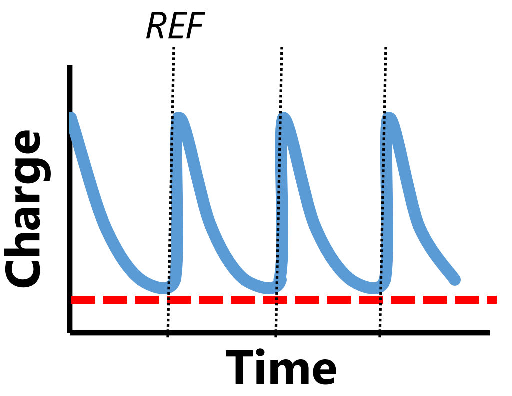
Discuss

Online Error Profiling
Power Management
Processing in/near Memory
...

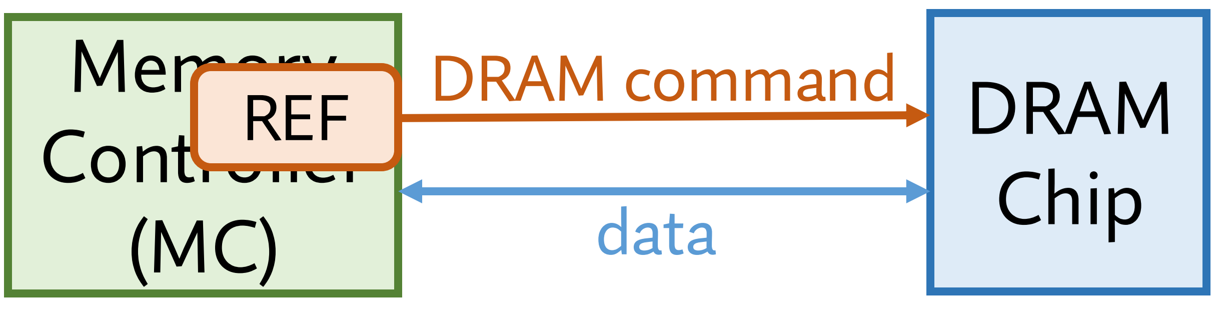
DRAM Periodic Refresh



DRAM encodes data in **leaky capacitors**



Necessitates periodic **refresh operations**



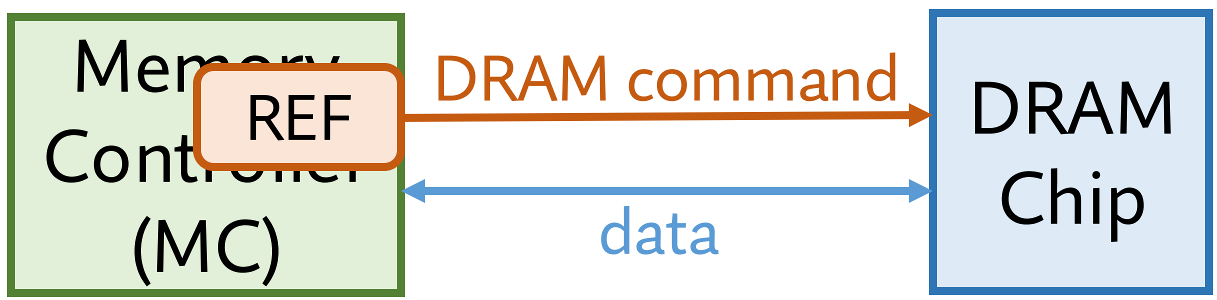
Alleviating the Drawbacks of Periodic Refresh

i Refresh commands spend command bus energy

- e.g., 8192 REF commands in 64 milliseconds in DDR4

ii Entire chip or bank inaccessible during refresh

- e.g., for 350 nanoseconds in DDR4



Alleviating the Drawbacks of Periodic Refresh

i Refresh commands spend command bus energy

- e.g., 8192 REF commands in 64 milliseconds in DDR4

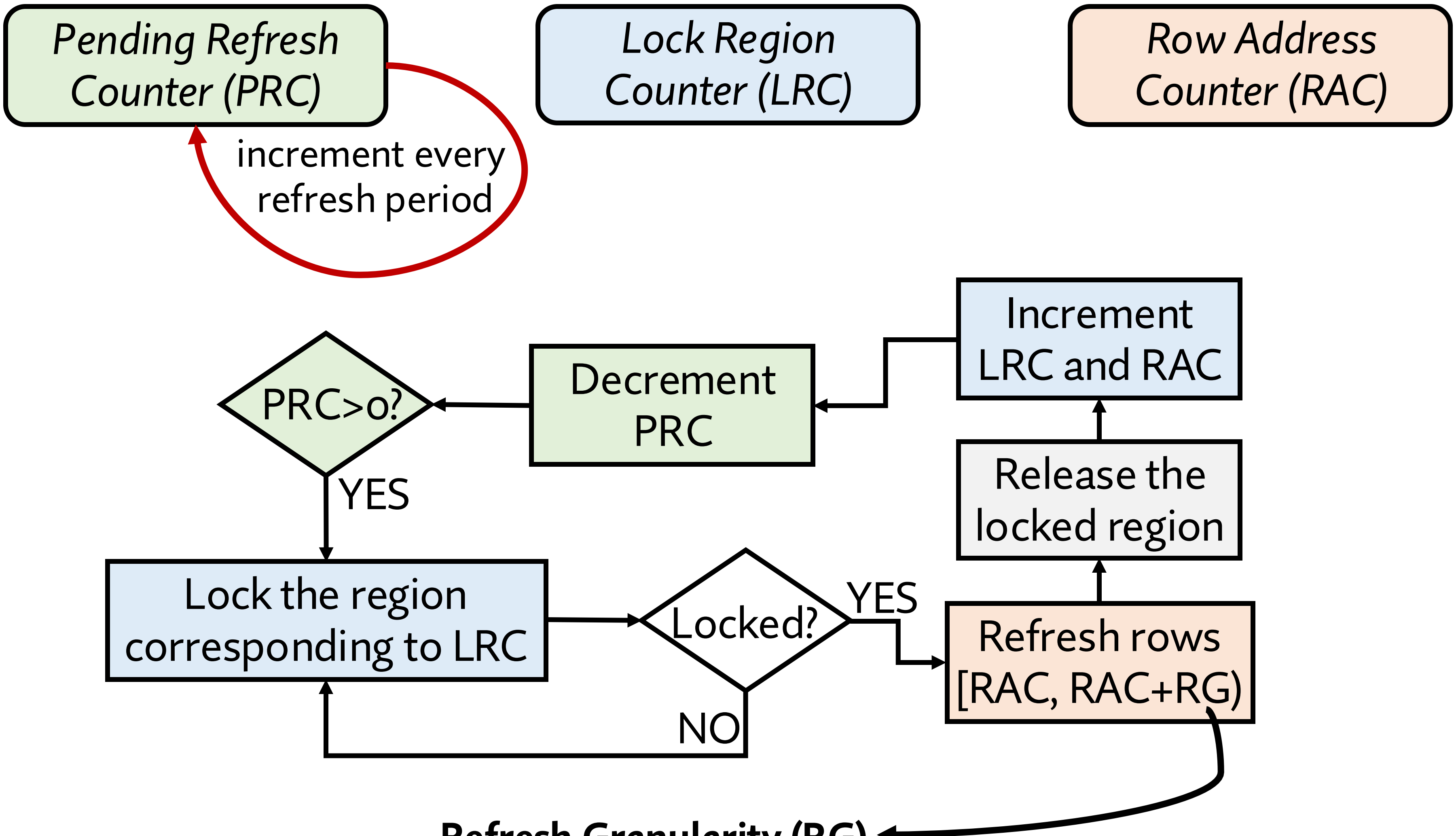
ii Entire chip or bank inaccessible during refresh

- e.g., for 350 nanoseconds in DDR4

i No refresh commands sent over the command bus

ii Allow access to most of the chip
that is not under maintenance

SMD-FR – Implementation



Refresh Granularity (RG)
number of rows refreshed every time a region is locked

SMD-Based Maintenance Mechanisms

Demonstrate the usefulness and versatility of SMD

i Fixed-Rate Refresh (SMD-FR)

ii **Deterministic RowHammer Protection (SMD-DRP)**

iii **Memory Scrubbing (SMD-MS)**

https://arxiv.org/pdf/2207.13358

Evaluate

Variable-Rate Refresh
Probabilistic RowHammer Protection

Discuss

Online Error Profiling
Power Management
Processing in/near Memory
...

SMD Outline

1. Motivation

2. Self-Managing DRAM (SMD)

3. Use Cases

4. **Evaluations**

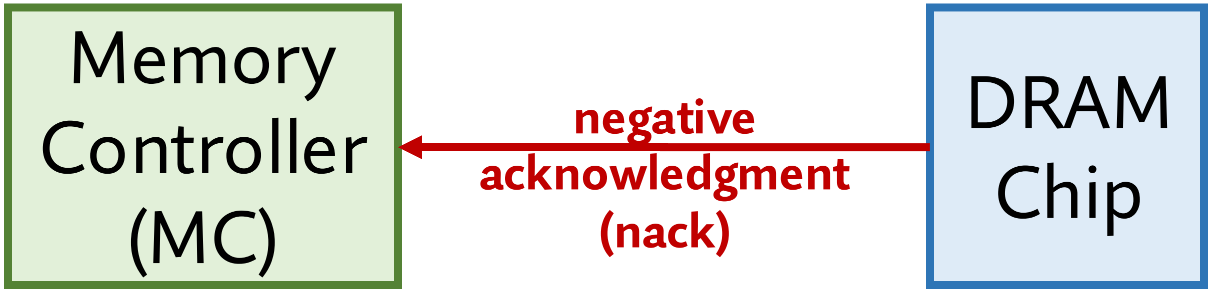
5. Conclusion and Takeaways

Hardware Implementation and Overhead (I)

1 DRAM interface modifications

Two options:

1. Use existing *alert_n* signal at no additional pin cost OR
2. Add a new pin for each rank of DRAM chips (~1.6% processor pin count)



One interface change to end all interface changes for new in-DRAM maintenance mechanisms

Hardware Implementation and Overhead (II)

2 DRAM chip modifications

i **Lock Region Bitvector (LRB)** — 0.001%* of a 45.5 mm^2 DRAM chip

ii **Maintenance-access parallelization**
1.1%* of a 45.5 mm^2 DRAM chip

iii **Maintenance mechanisms (orthogonal to SMD)**
https://arxiv.org/pdf/2207.13358

*modeled using CACTI 6.0

Hardware Implementation and Overhead (III)

3 **Memory controller modifications**

- 288 bytes of storage to keep track of locked regions
- Leverage existing memory request scheduling logic for handling rejected ACT commands

Detailed explanation:
https://arxiv.org/pdf/2207.13358

Evaluation Methodology

- Cycle-level simulations using **Ramulator** [Kim+, CAL'15]

- **Baseline** system configuration
  - **Processor:** 4GHz, 4-wide issue, 8 MSHRs/core
  - **Last-Level Cache:** 8-way associative, 4 MiB/core
  - **Memory Controller:** 64-entry read/write request queue
FR-FCFS-Cap with Cap = 7
  - **DRAM:** DDR4-3200, 32 ms refresh period
4 channels, 2 ranks, 16 banks, 128K rows

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

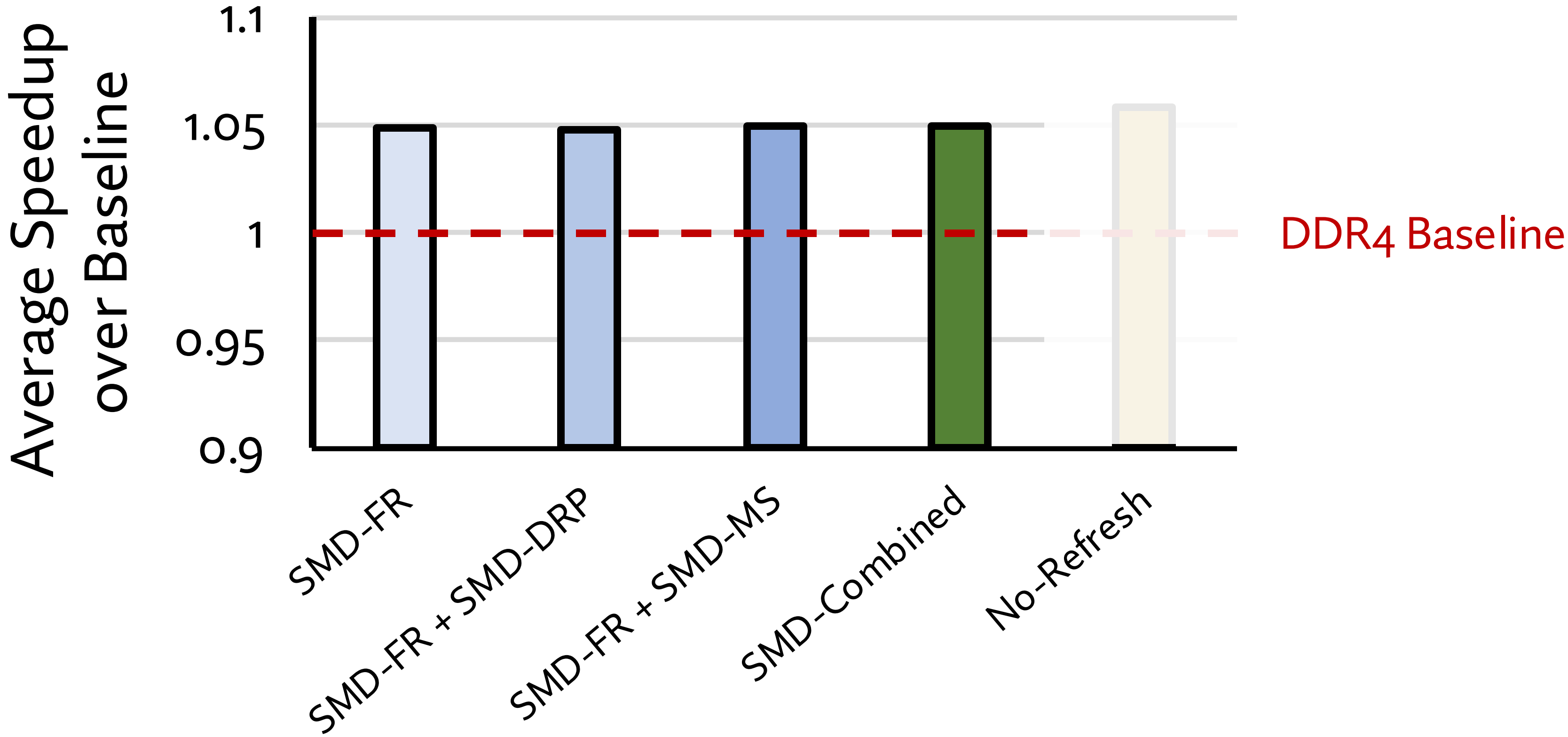
- **SMD** parameters
  - 16 lock regions in a DRAM bank
  - 16 subarrays in one lock region
  - Retry Interval (RI) = 62.5 nanoseconds

- 62 single-core and 60 four-core **workloads**
  - SPEC CPU2006/2017, TPC, STREAM, MediaBench

Evaluated System Configurations

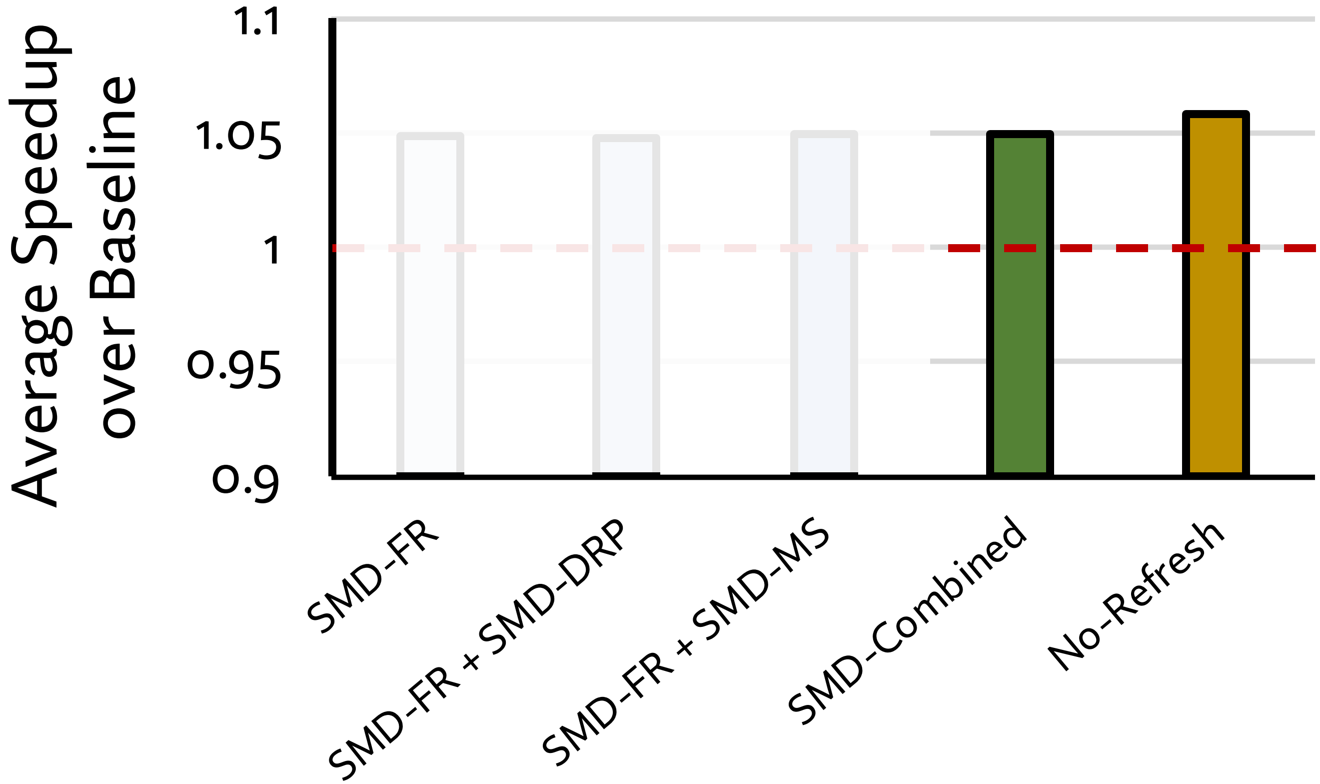
- Baseline DDR4 system
  - refresh window = 32 millisecond
- Fixed-Rate Refresh (**SMD-FR**)
  - refresh window = 32 millisecond, refresh granularity = 8
- Deterministic RowHammer Protection (**SMD-FR + SMD-DRP**)
  - refresh neighbor rows of a row that gets activated 512 times
- Memory Scrubbing (**SMD-FR + SMD-MS**)
  - 5-minute scrubbing period
- **SMD-Combined** combines SMD-FR + SMD-DRP + SMD-MS
- **No-Refresh** DDR4 system that does **not** do maintenance

Single-Core Performance



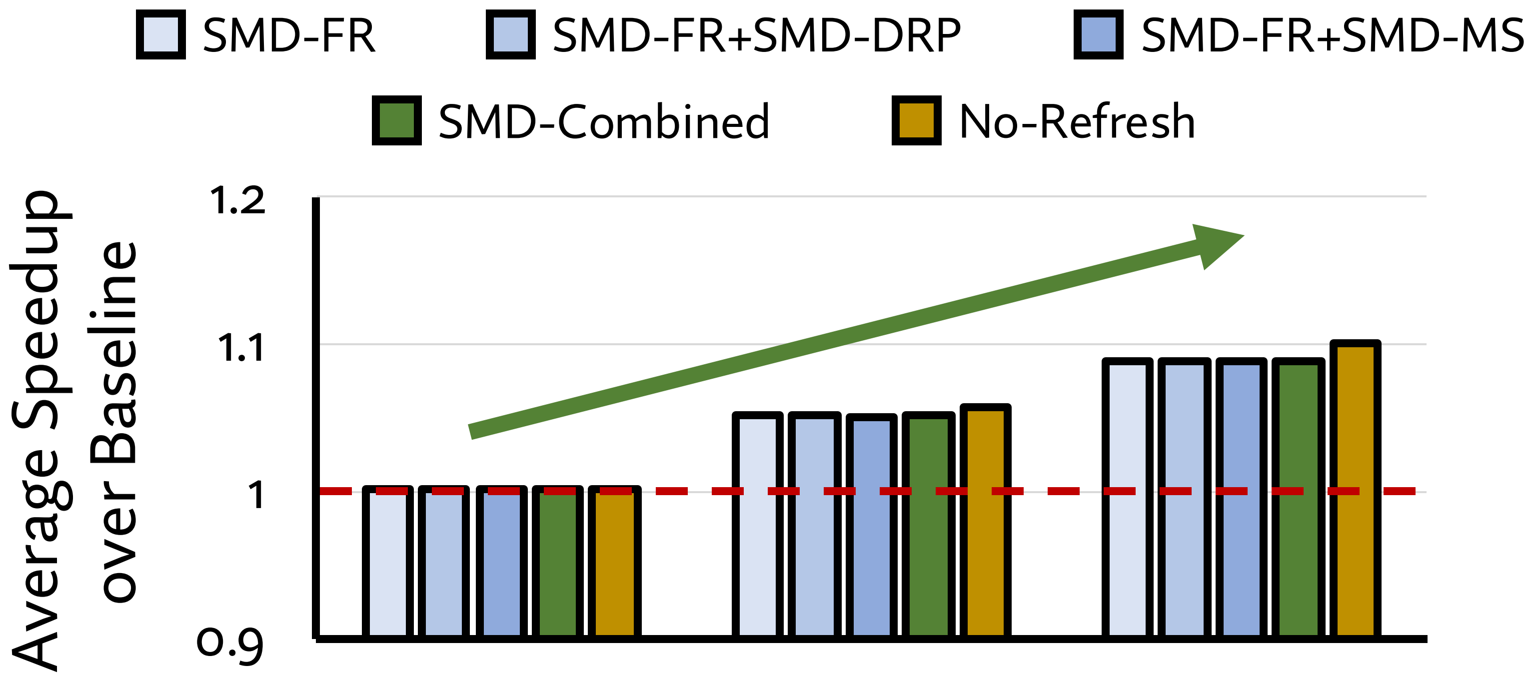
SMD provides 4.8% to 5.0% average speedup

Single-Core Performance



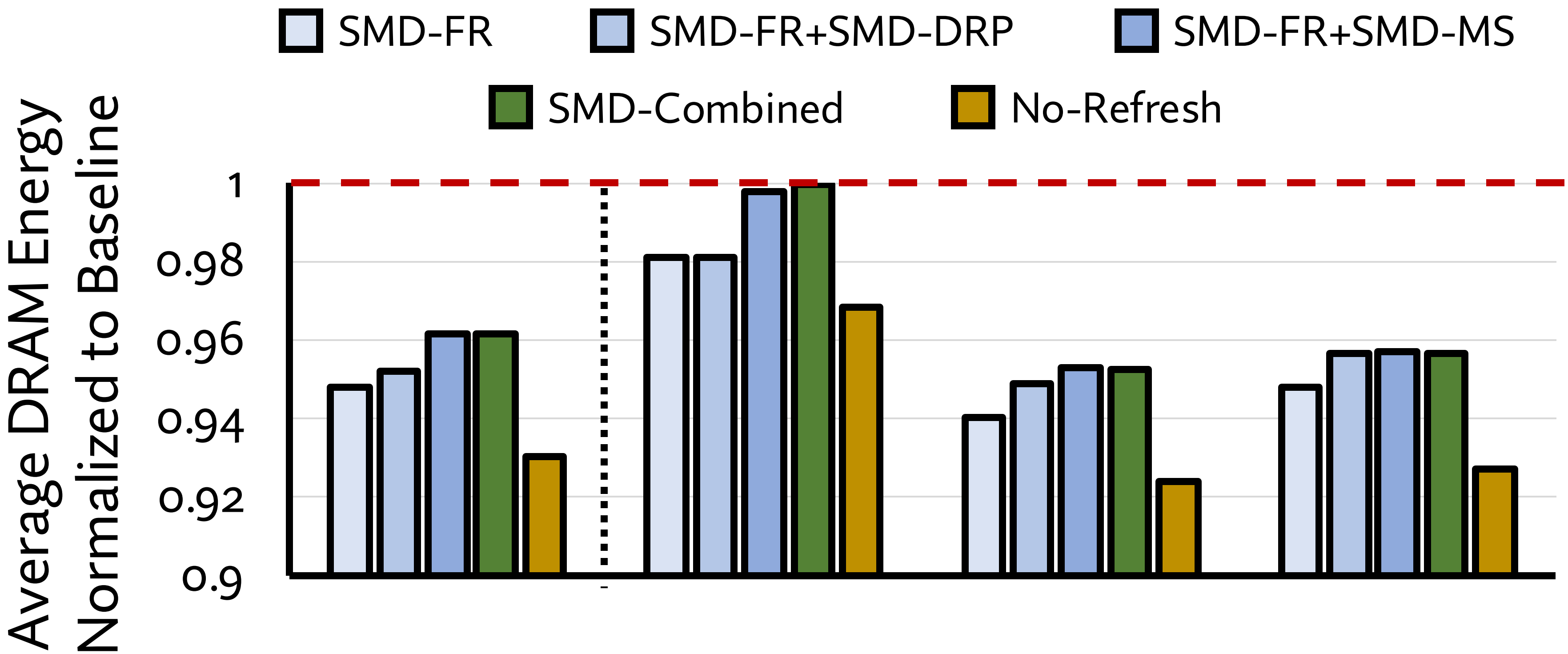
SMD-Combined provides
84.7% the speedup of No-Refresh

Four-Core Performance



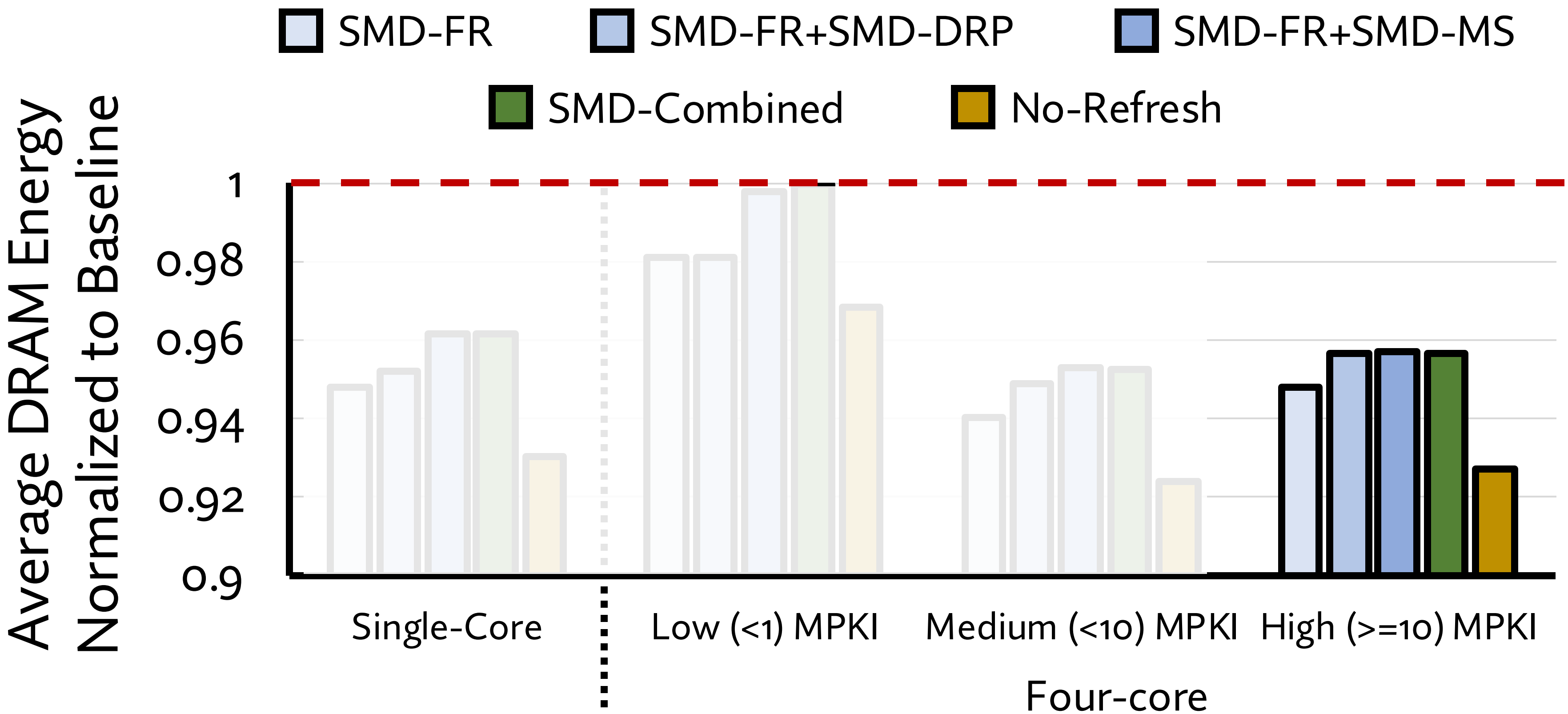
SMD provides higher speedups
with increasing workload memory intensity

DRAM Energy



All SMD configurations provide energy savings

DRAM Energy



SMD-Combined provides
59.6% of the energy savings of No-Refresh

Performance and Energy Summary

SMD provides performance and energy benefits comparable to a hypothetical system without maintenance while improving system robustness

- Benefits over the baseline system attributed to:

1. Overlapping the latency of maintenance operations with useful access operations
2. Reduced command interference and energy use: MC does not issue maintenance commands

More in the Paper

- Proof of forward progress for memory requests
- Discussion of more use cases
  - Variable rate refresh, RowHammer defenses, online error profiling…
  - Power management, processing-near-memory
- Design choices
  - Evaluation of a policy that pauses maintenance operations
  - Discussion of a predictable SMD interface
- Sensitivity analyses
  - Performance improves with number of lock regions
  - Benefits increase with reducing refresh period
  - Provide similar benefits across 1-, 2-, 4-, 8-core workloads
- SMD-based scrubbing vs. MC-based scrubbing
  - SMD induces ~8X less overhead at a very high scrubbing rate

More in the Paper

https://arxiv.org/pdf/2207.13358

Self-Managing DRAM: A Low-Cost Framework for Enabling Autonomous and Efficient DRAM Maintenance Operations

Hasan Hassan† Ataberk Olgun† A. Giray Yağlıkçı Haocong Luo Onur Mutlu

ETH Zürich

The memory controller is in charge of managing DRAM maintenance operations (e.g., refresh, RowHammer protection, memory scrubbing) to reliably operate modern DRAM chips. Implementing new maintenance operations often necessitates modifications in the DRAM interface, memory controller, and potentially other system components. Such modifications are only possible with a new DRAM standard, which takes a long time to develop, likely leading to slow progress in the adoption of new architectural techniques in DRAM chips.

We propose a new low-cost DRAM architecture, Self-Managing DRAM (SMD), that enables autonomous in-DRAM maintenance operations by transferring the responsibility for controlling maintenance operations from the memory controller to the SMD chip. To enable autonomous maintenance operations, we make a single, simple modification to the DRAM interface, such that an SMD chip rejects memory controller accesses to DRAM regions (e.g., a subarray or a bank) under maintenance, while allowing memory accesses to other DRAM regions. Thus, SMD enables

tion [12, 18, 47–122], and 3) memory scrubbing [17, 123–135].[1] New DRAM chip generations necessitate making existing maintenance operations more aggressive (e.g., lowering the refresh period [119, 136, 137]) and introducing new types of maintenance operations (e.g., targeted refresh [64, 66, 138], DDR5 RFM [119], and PRAC [119] as RowHammer defenses).[2]

Two problems likely hinder the adoption of effective and efficient maintenance mechanisms in modern and future DRAM-based computing systems. First, it is difficult to modify existing maintenance mechanisms and introduce new maintenance operations because doing so often necessitates changes to the DRAM interface, which takes a long time (due to various issues related to standardization and agreement across many vendors with conflicting interests [4, 6]). Second, it is challenging to keep the overhead of DRAM maintenance mechanisms low as DRAM reliability characteristics worsen and DRAM chips require more aggressive maintenance operations. We expand on the two problems in the next two paragraphs.

SMD Outline

1. Motivation

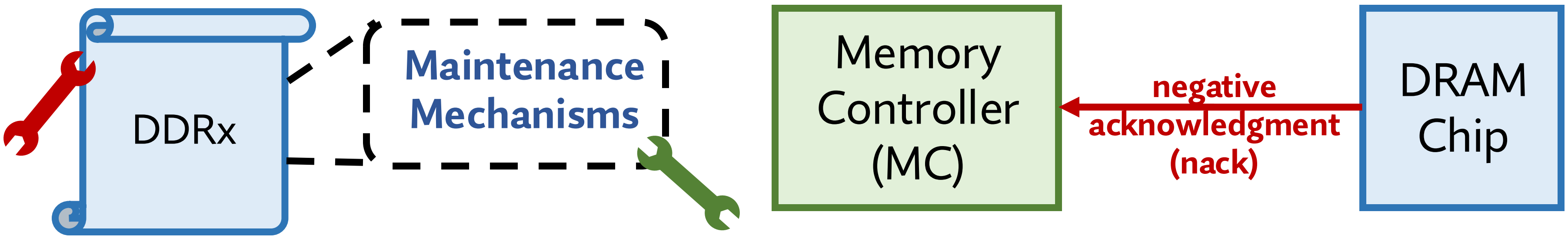
2. Self-Managing DRAM (SMD)

3. Use Cases

4. Evaluations

5. **Conclusion and Takeaways**

Self-Managing DRAM Conclusion



New maintenance mechanisms require changes to DRAM standards

With a simple, single modification to the DRAM interface,
SMD enables implementing new in-DRAM maintenance mechanisms
with no further changes to the DRAM interface and other components

We showcase three high-performance and energy-efficient
SMD-based in-DRAM maintenance mechanisms

Our Hope

SMD enables practical adoption of innovative ideas in DRAM design and inspires better ways of partitioning work between processor and DRAM

Extended Version on ArXiv

https://arxiv.org/pdf/2207.13358

Self-Managing DRAM: A Low-Cost Framework for Enabling Autonomous and Efficient DRAM Maintenance Operations

Hasan Hassan† Ataberk Olgun† A. Giray Yağlıkçı Haocong Luo Onur Mutlu

ETH Zürich

The memory controller is in charge of managing DRAM maintenance operations (e.g., refresh, RowHammer protection, memory scrubbing) to reliably operate modern DRAM chips. Implementing new maintenance operations often necessitates modifications in the DRAM interface, memory controller, and potentially other system components. Such modifications are only possible with a new DRAM standard, which takes a long time to develop, likely leading to slow progress in the adoption of new architectural techniques in DRAM chips.

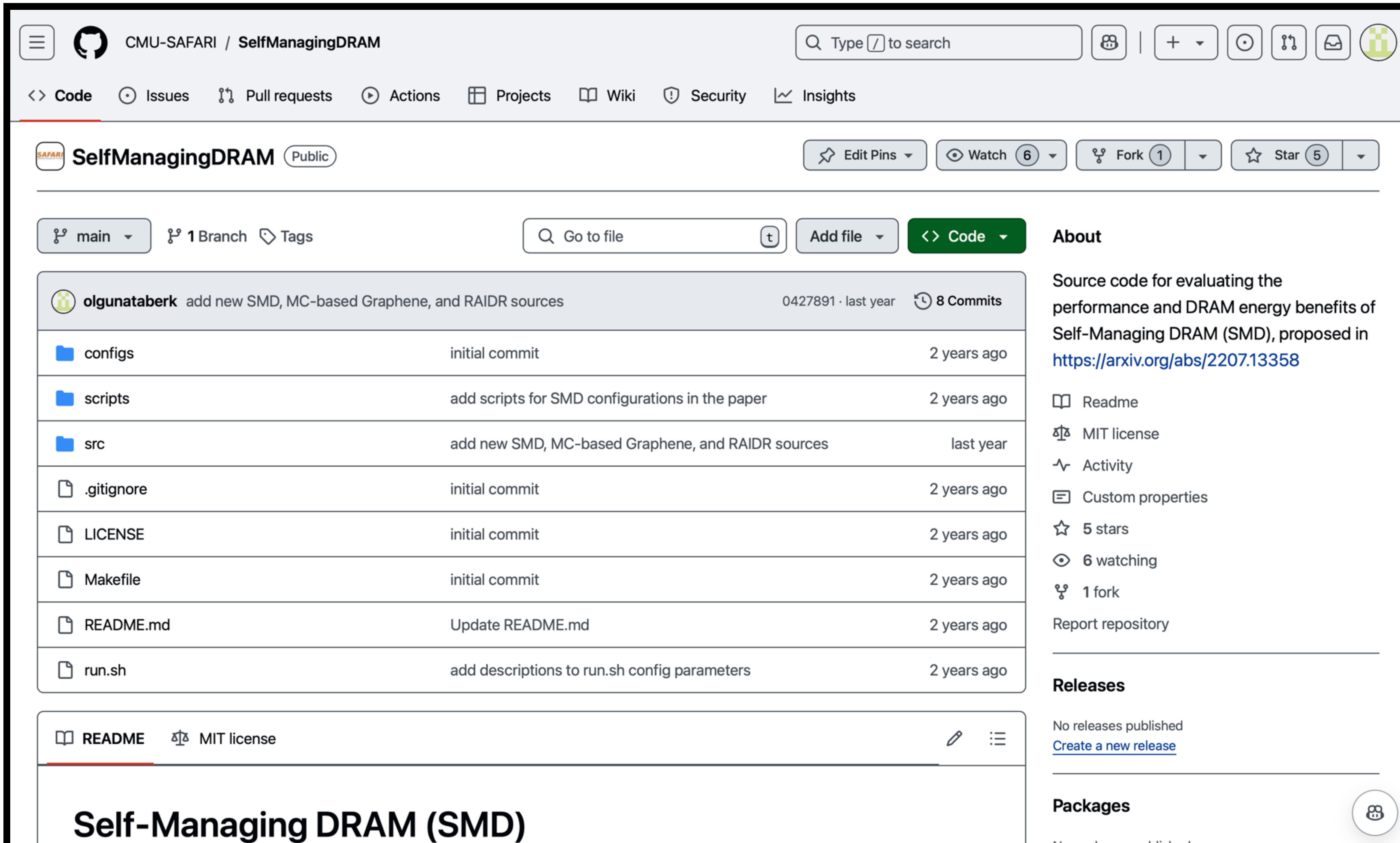
We propose a new low-cost DRAM architecture, Self-Managing DRAM (SMD), that enables autonomous in-DRAM maintenance operations by transferring the responsibility for controlling maintenance operations from the memory controller to the SMD chip. To enable autonomous maintenance operations, we make a single, simple modification to the DRAM interface, such that an SMD chip rejects memory controller accesses to DRAM regions (e.g., a subarray or a bank) under maintenance, while allowing memory accesses to other DRAM regions. Thus, SMD enables

tion [12, 18, 47–122], and 3) memory scrubbing [17, 123–135].[1] New DRAM chip generations necessitate making existing maintenance operations more aggressive (e.g., lowering the refresh period [119, 136, 137]) and introducing new types of maintenance operations (e.g., targeted refresh [64, 66, 138], DDR5 RFM [119], and PRAC [119] as RowHammer defenses).[2]

Two problems likely hinder the adoption of effective and efficient maintenance mechanisms in modern and future DRAM-based computing systems. First, it is difficult to modify existing maintenance mechanisms and introduce new maintenance operations because doing so often necessitates changes to the DRAM interface, which takes a long time (due to various issues related to standardization and agreement across many vendors with conflicting interests [4, 6]). Second, it is challenging to keep the overhead of DRAM maintenance mechanisms low as DRAM reliability characteristics worsen and DRAM chips require more aggressive maintenance operations. We expand on the two problems in the next two paragraphs.

SMD is Open-Sourced

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



Self-Managing DRAM (SMD)

A Low-Cost Framework for Enabling Autonomous and Efficient DRAM Maintenance Operations

Hasan Hassan, Ataberk Olgun,
A. Giray Yaglikci, Haocong Luo, Onur Mutlu

https://arxiv.org/pdf/2207.13358

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

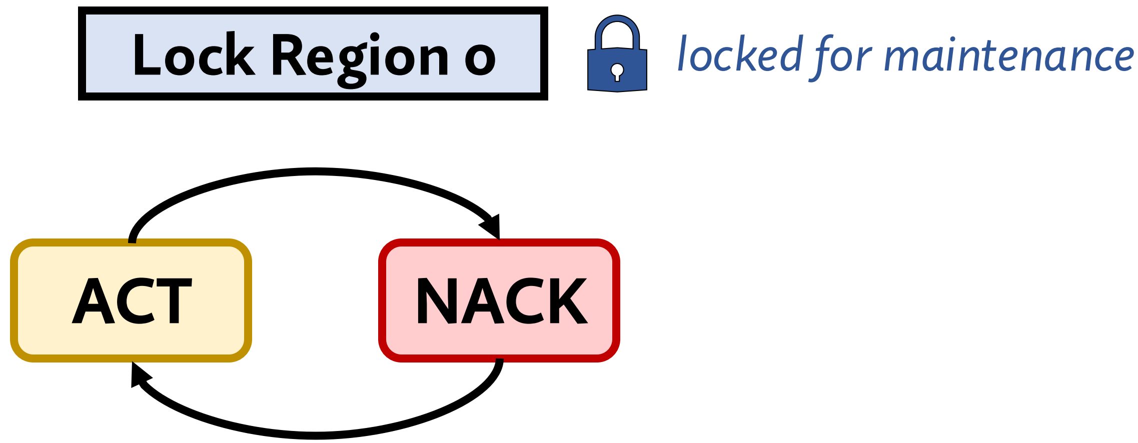
ETH zürich

SAFARI

Backup Slides

Ensuring Forward Progress

- SMD breaks the chain of ACT commands and rejections



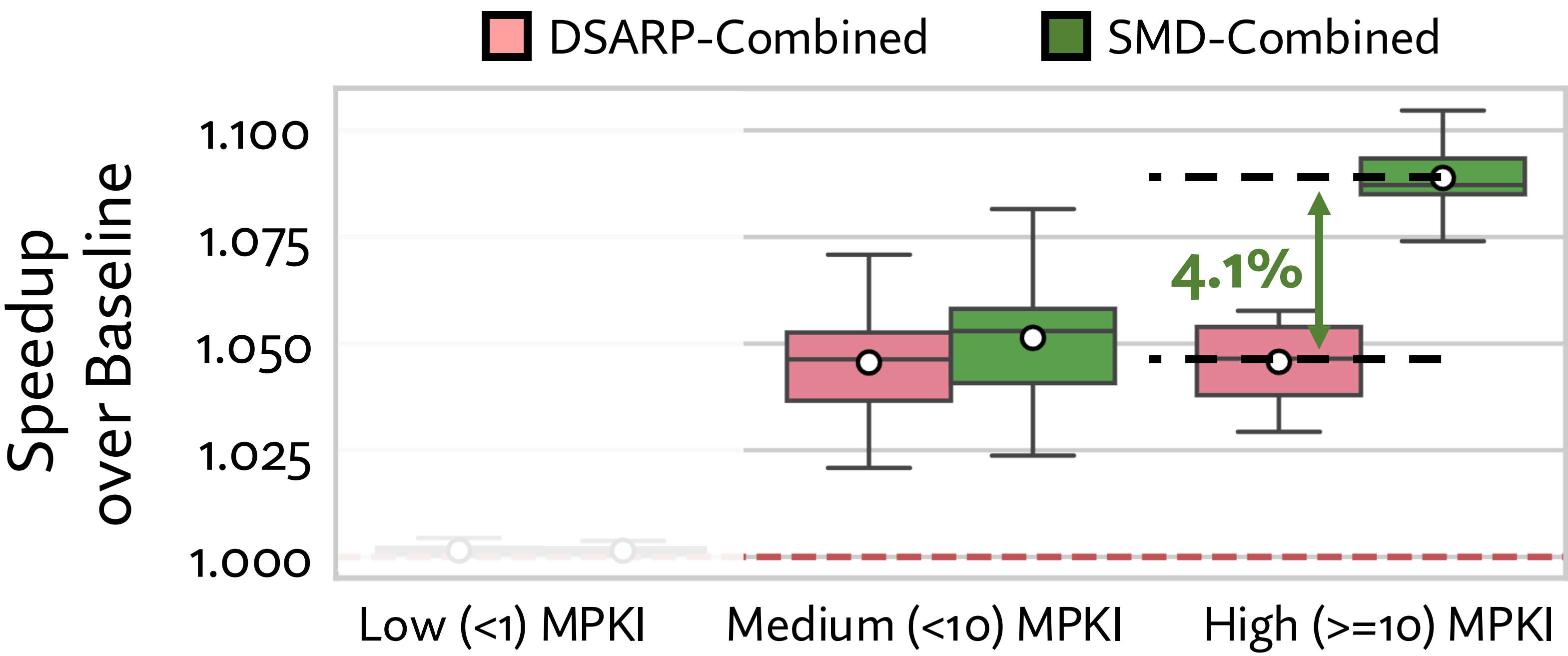
- because:

(i) MC issues the rejected ACT at the end of every RI

(ii) region is not locked for at least one RI after maintenance ends

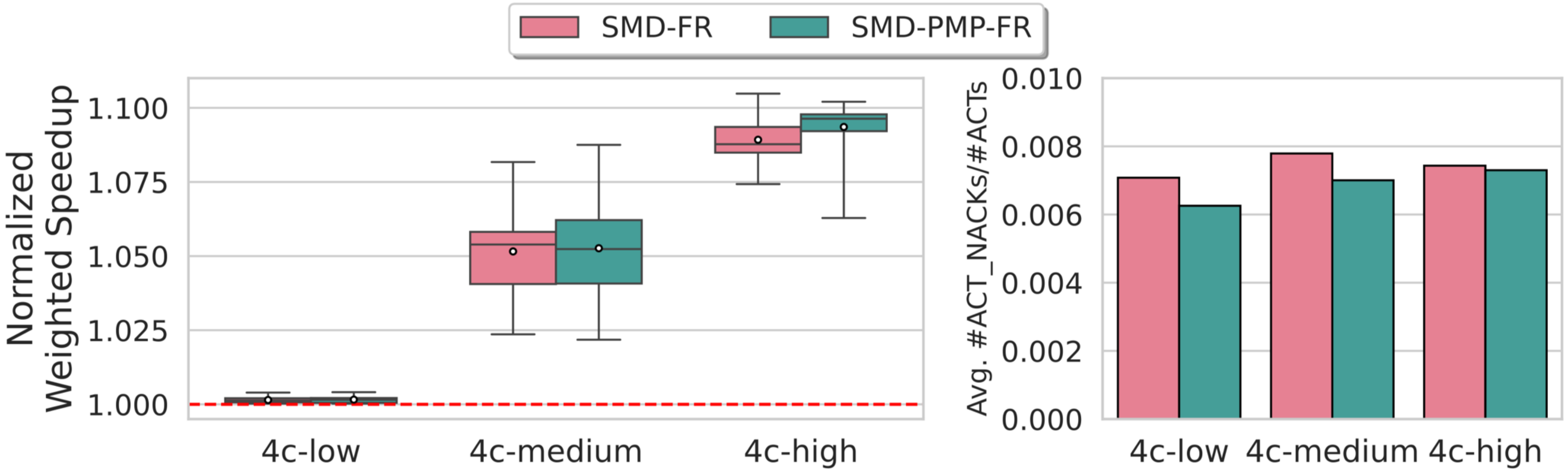
Performance Comparison

- DSARP [Chang+, HPCA'14]
  - MC-based maintenance-access parallelization

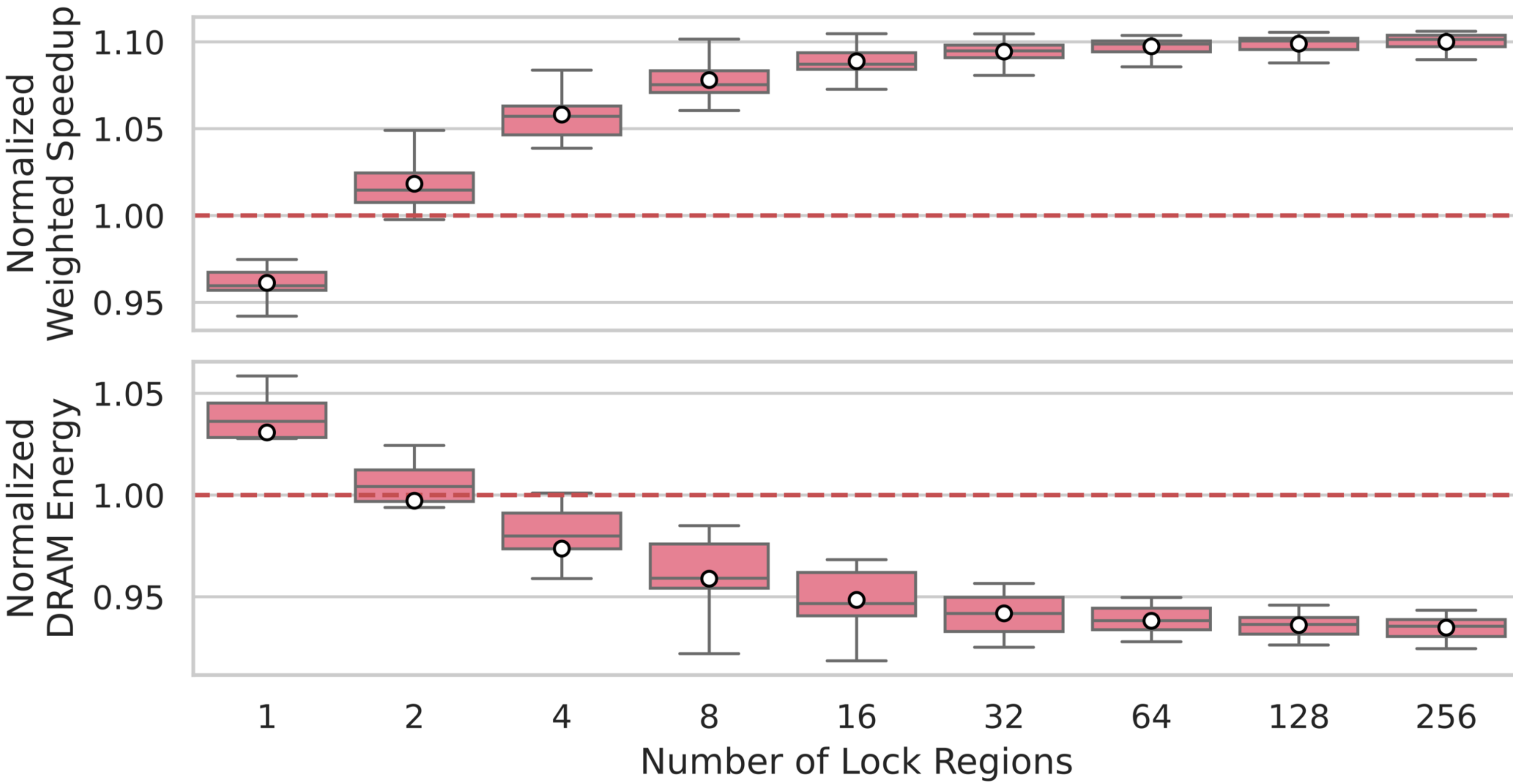


SMD outperforms DSARP

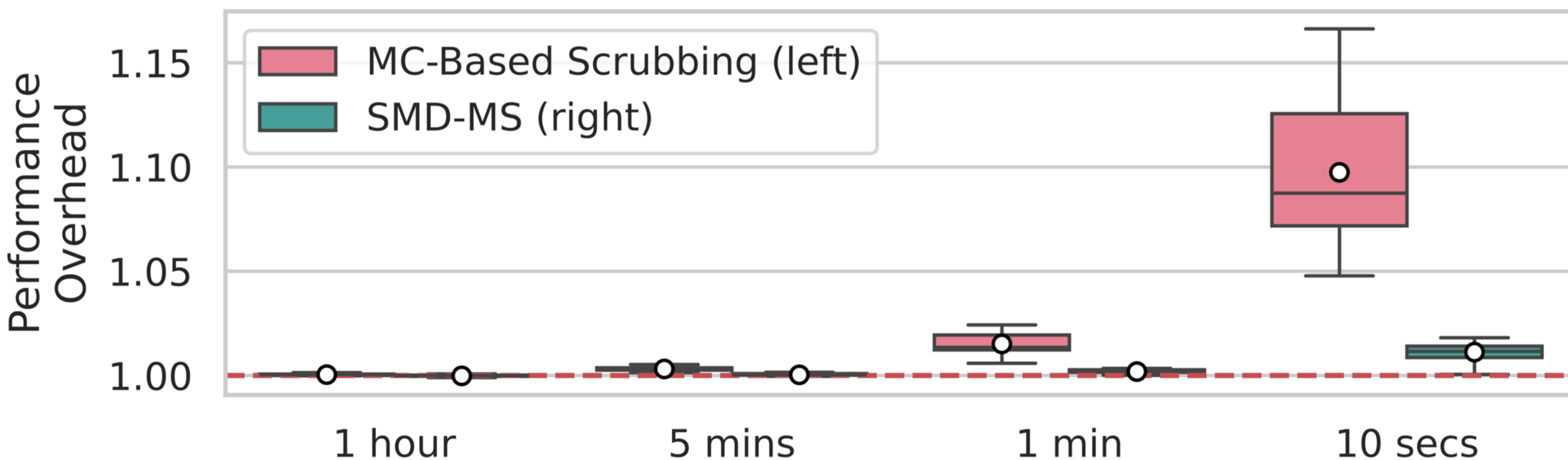
Pause Maintenance Policy



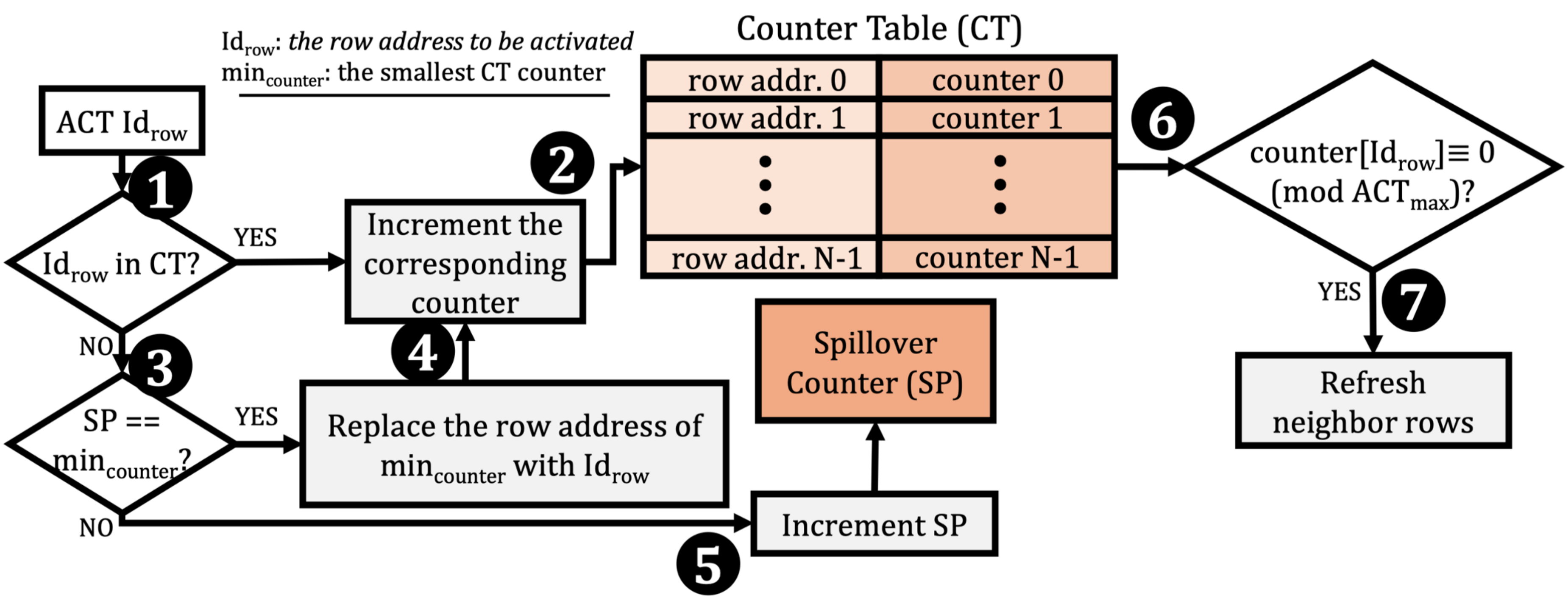
Sensitivity to Number of Lock Regions



SMD-based vs. MC-based Scrubbing

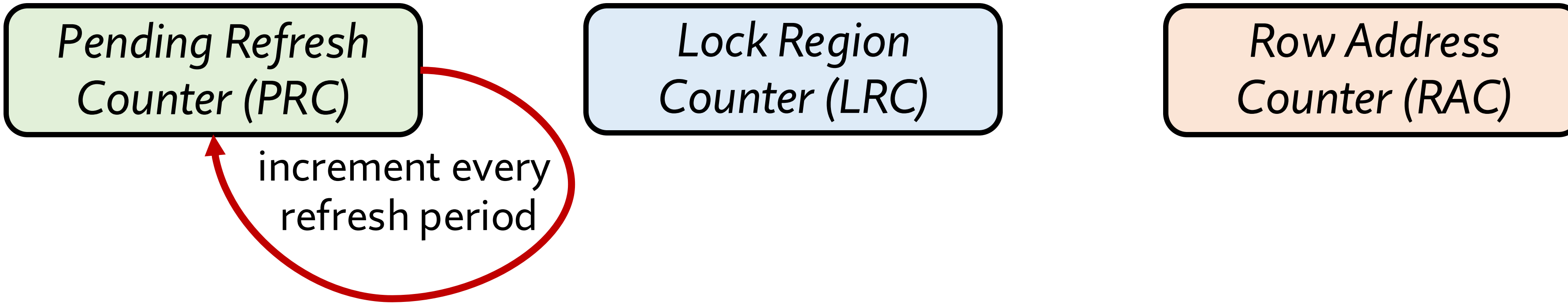


SMD-DRP

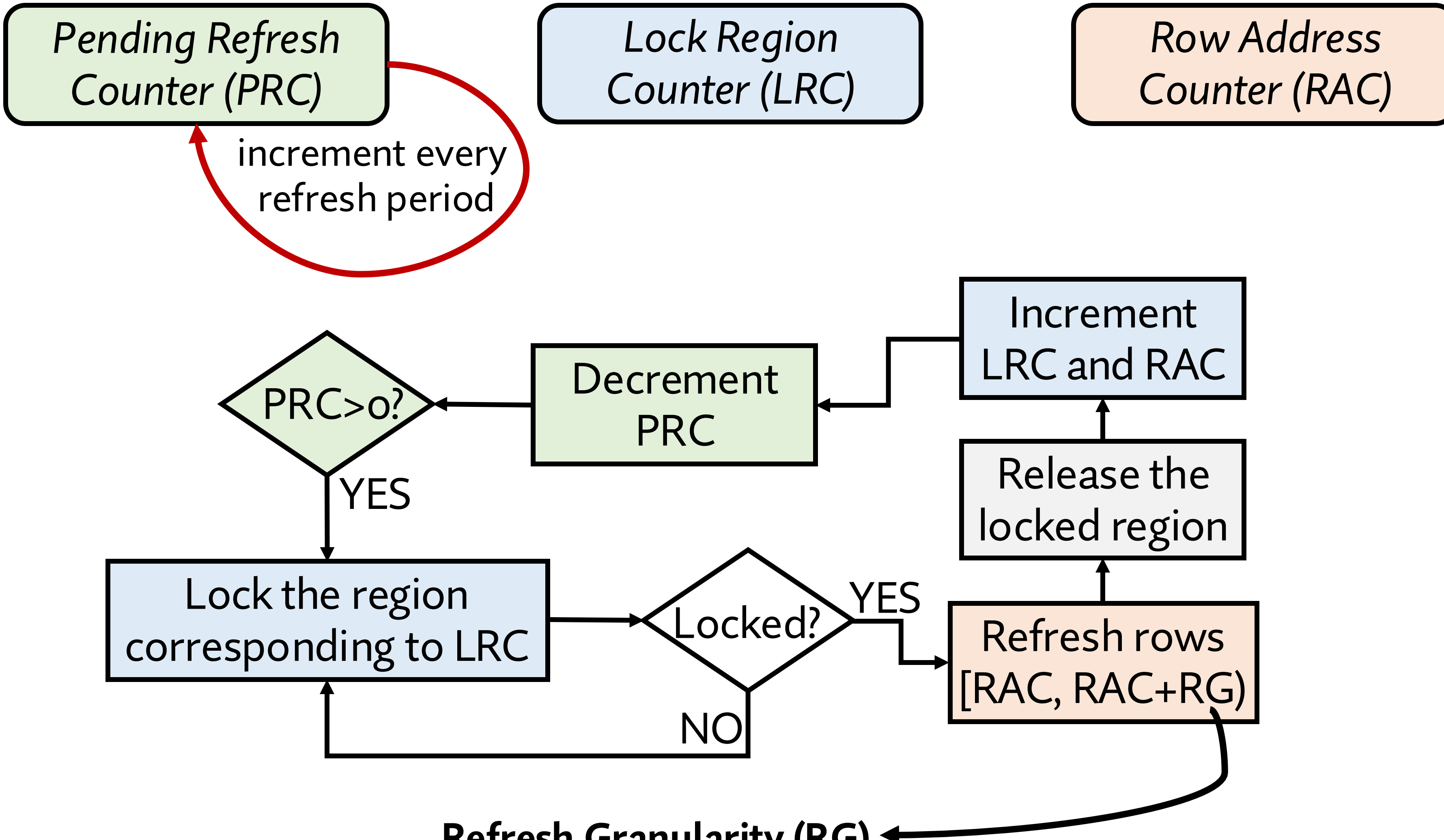


Counter Table (CT)
Id_row: the row address to be activated
min_counter: the smallest CT counter
ACT Id_row
Id_row in CT?
YES
NO
SP == min_counter?
YES
NO
Increment the corresponding counter
Replace the row address of min_counter with Id_row
row addr. 0
counter 0
row addr. 1
counter 1
row addr. N-1
counter N-1
counter[Id_row] ≡ 0 (mod ACT_max)?
YES
Refresh neighbor rows
Spillover Counter (SP)
Increment SP
1
2
3
4
5
6
7

SMD-FR – Implementation

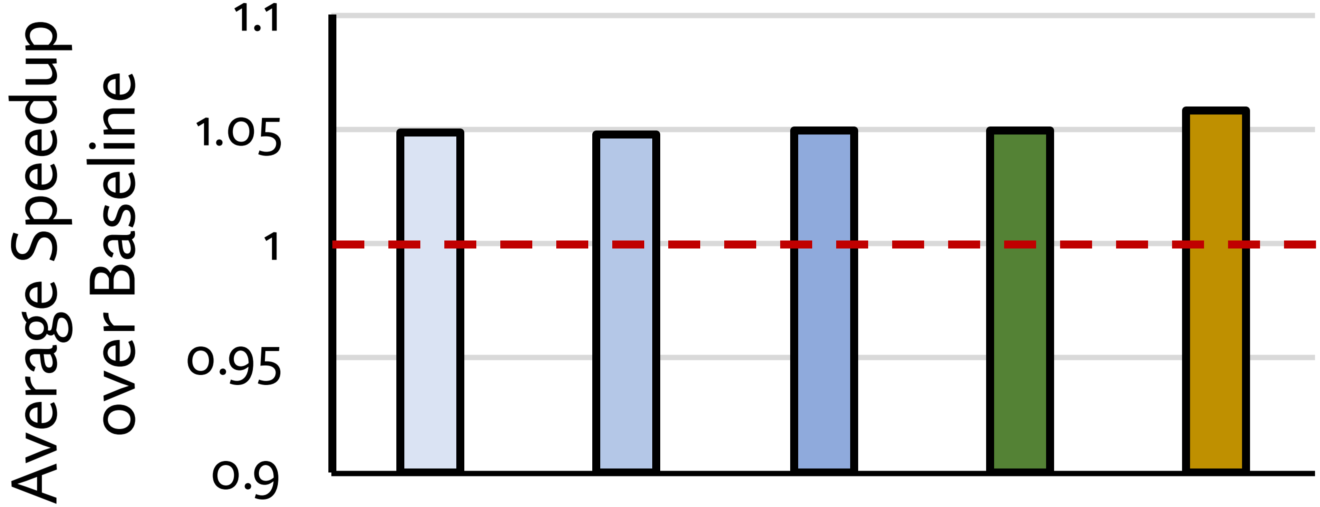


SMD-FR – Implementation



Refresh Granularity (RG)
number of rows refreshed every time a region is locked

Single-Core Performance



"PRAC already does this?"

Acknowledgments

We thank the anonymous reviewers of MICRO 2022, HPCA 2023, ISCA 2023, MICRO 2023, HPCA 2024, ISCA 2024, and MICRO 2024 for the feedback. We thank the SAFARI Research Group members for their valuable and constructive feedback along with the stimulating scientific and intellectual environ-

[2] A very recent update to the DDR5 standard [119] introduces PRAC, which is an on-DRAM-die read disturbance mitigation mechanism. PRAC requires more changes to the DRAM interface and continues to use RFM. Note that PRAC is concurrent with this work, as the initial version of this paper [139] was placed on arXiv on 27 July 2022 and initial submission to the MICRO 2022 conference was made on 22 April 2022.

Sectored DRAM
A Practical Energy-Efficient and High-Performance Fine-Grained DRAM Architecture

Ataberk Olgun
olgunataberk@gmail.com

F. Nisa Bostanci Geraldo F. Oliveira Yahya Can Tugrul Rahul Bera
A. Giray Yaglikci Hasan Hassan Oguz Ergin Onur Mutlu

ETH zürich SAFARI kasırga TOBB ETÜ Ekonomi ve Teknoloji Üniversitesi

Sectored DRAM Summary

Problem: DRAM-based systems suffer from two sources of energy inefficiency

1. Coarse-grained cache-block-sized (typically 64-byte) data transfer
2. Coarse-grained DRAM-row-sized (typically 8-kilobyte) activation

A workload does not use all data fetched from DRAM

Goal: Design a fine-grained, low-cost, and high-throughput DRAM substrate

- Mitigate excessive energy consumption from coarse-grained DRAM

Key Ideas: Small modifications to memory controller and DRAM chip enable

1. Transferring sub-cache-block-sized data in a variable number of clock cycles
2. Activating relatively small physically isolated regions of a DRAM row

based on the workload memory access pattern

Key Results: For the evaluated memory-intensive workloads, Sectored DRAM

- Improves system energy consumption by 14%, system performance by 17%
- Incurs 0.39 mm^2 (1.7%) DRAM chip area overhead
- Performs within 11% of a state-of-the-art prior work (Half-DRAM), with 12% smaller DRAM energy and 34% smaller area overhead

Outline

1. Background & Motivation

2. Sectored DRAM: Design

3. Sectored DRAM: System Integration

4. Evaluation

5. Conclusion

Outline

1. Background & Motivation

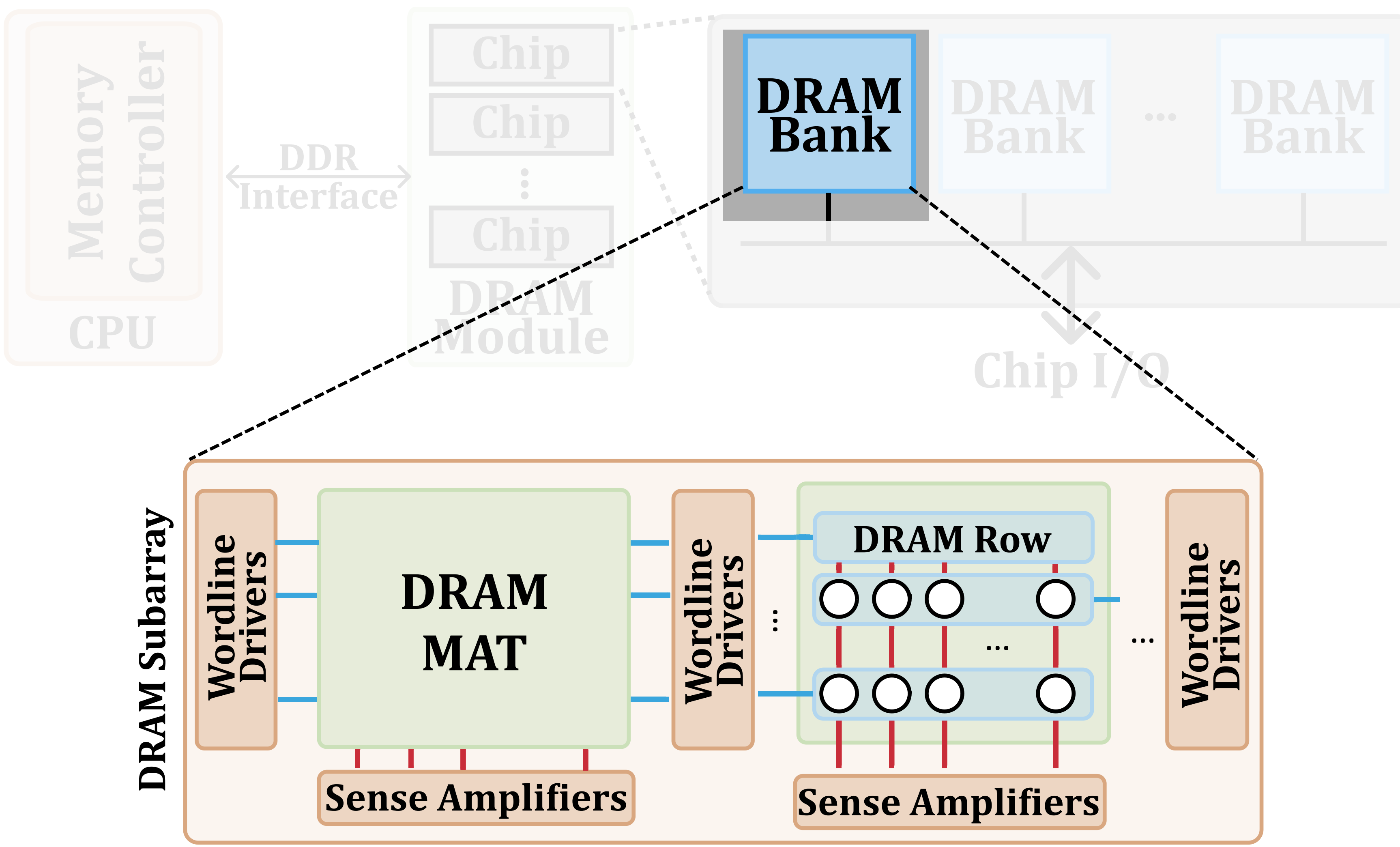
2. Sectored DRAM: Design

3. Sectored DRAM: System Integration

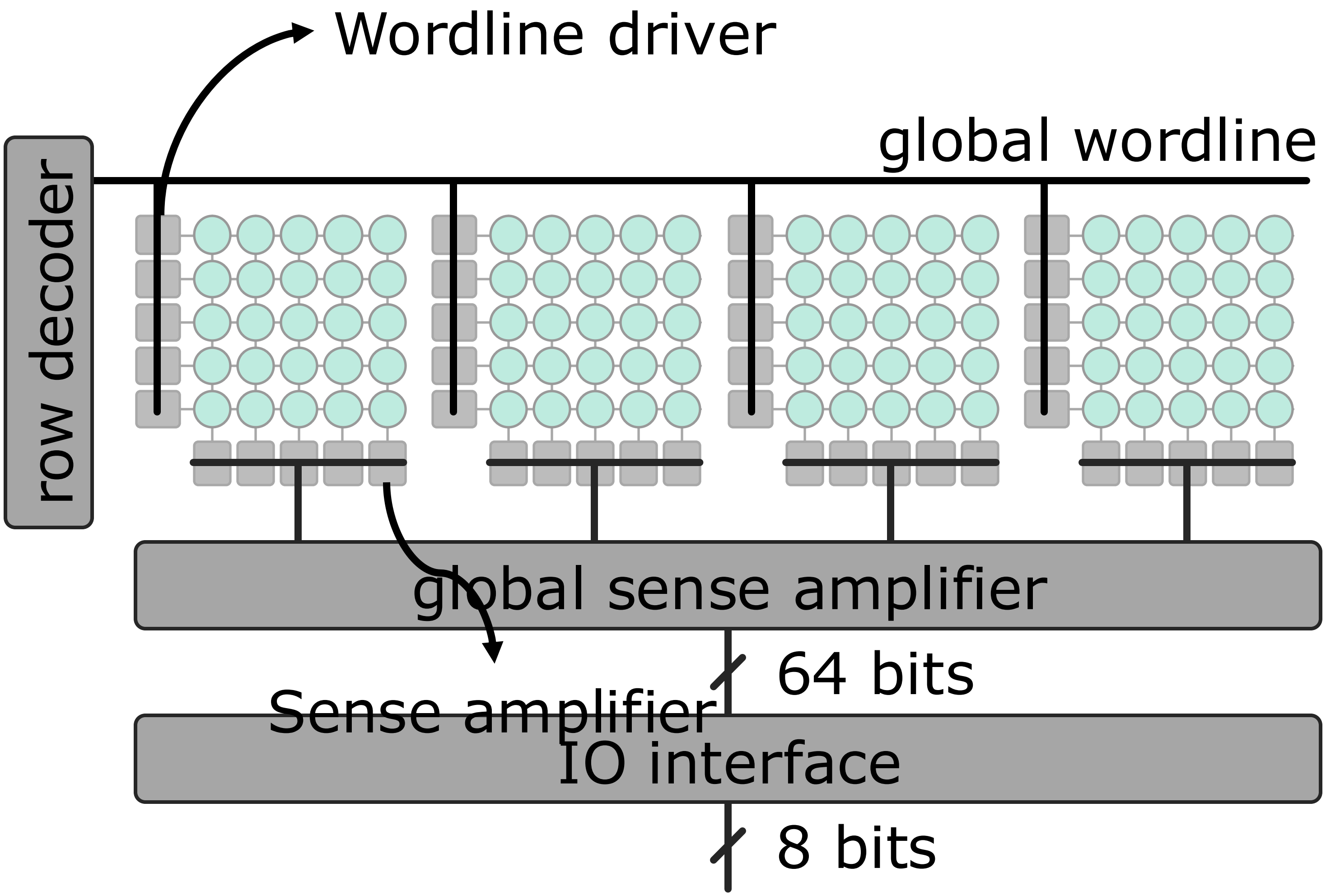
4. Evaluation

5. Conclusion

DRAM is Organized Hierarchically

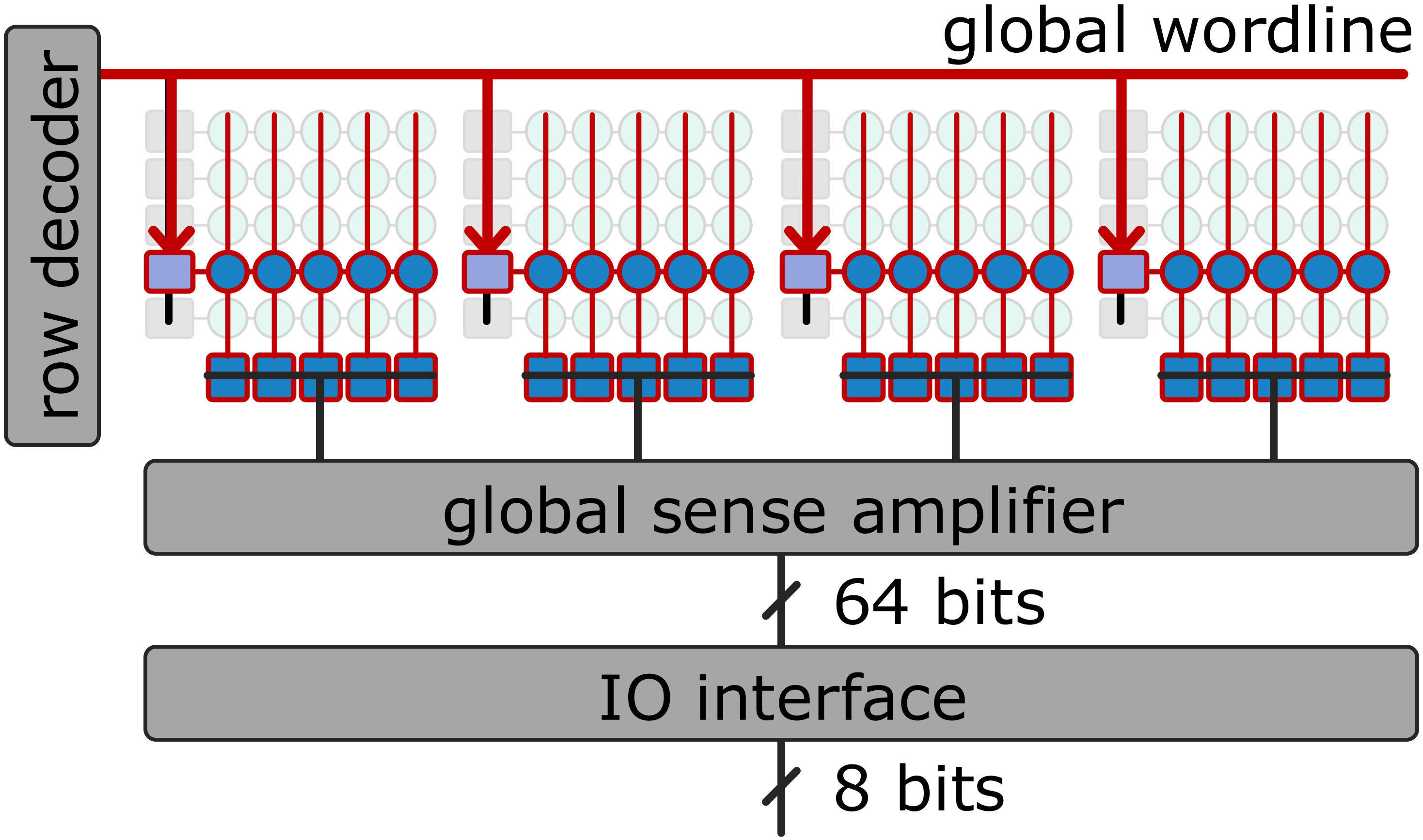


DRAM Row Activate Operation

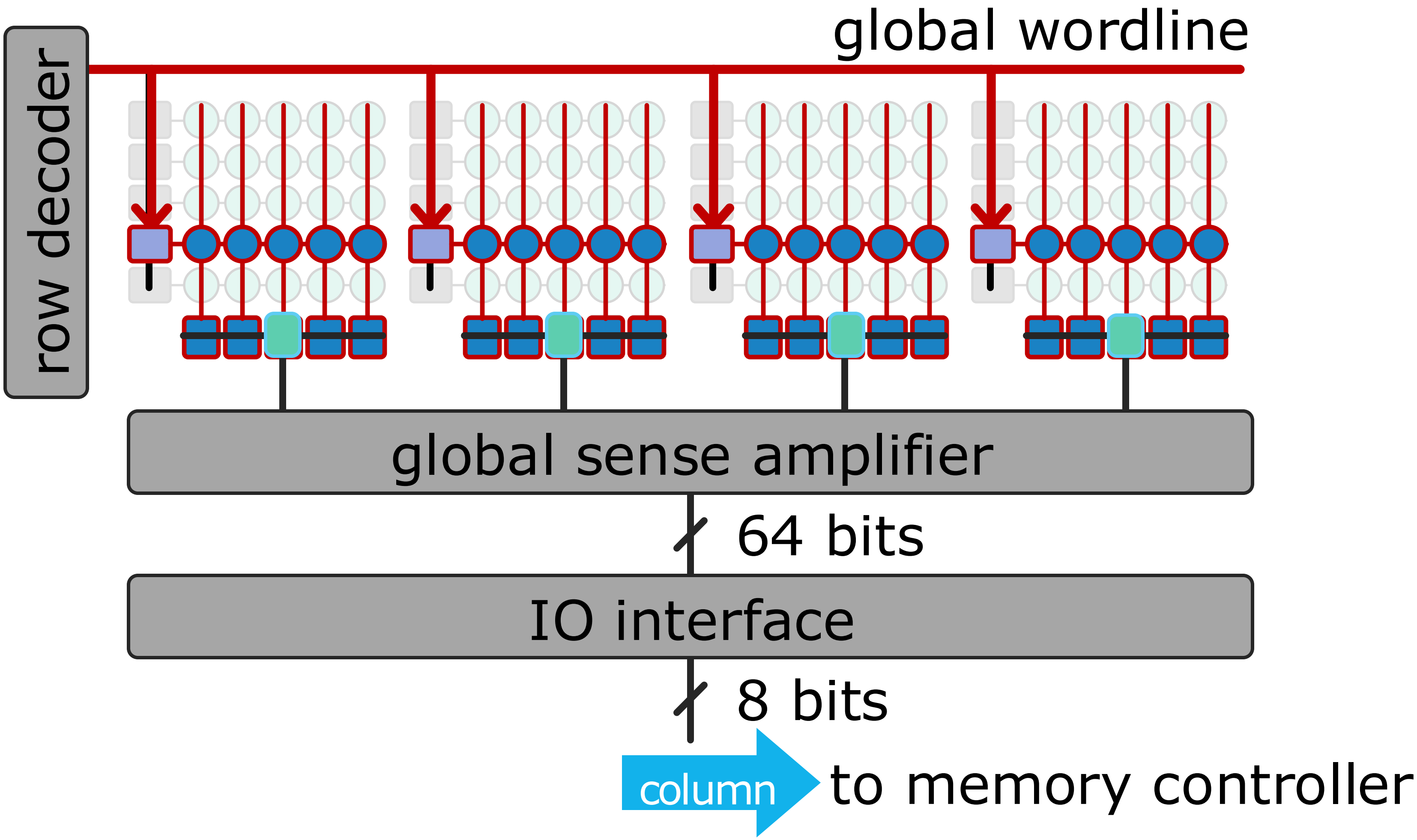


[Oliveira+, HPCA'24]

DRAM Row Activate Operation



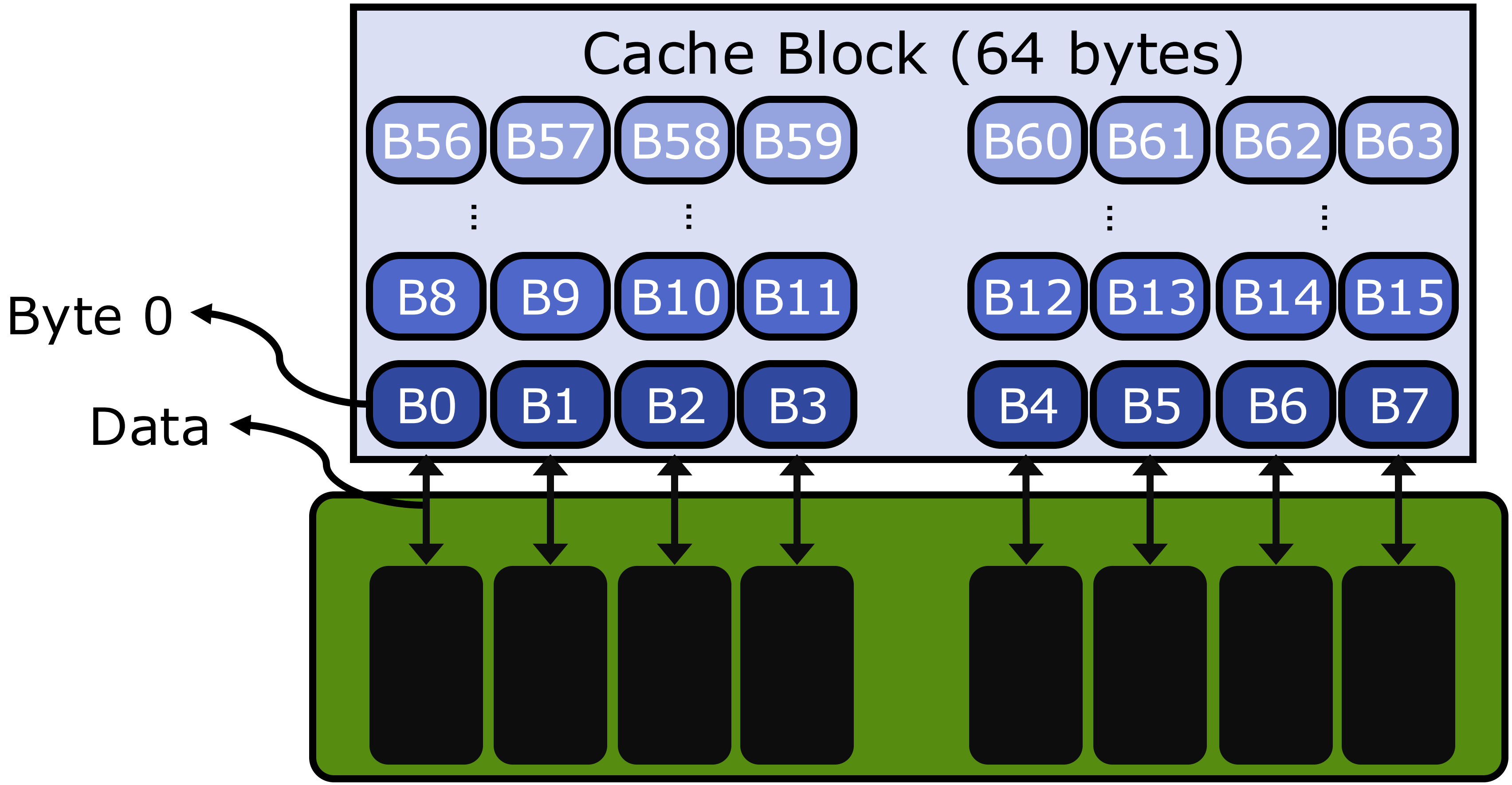
DRAM Column Read Operation



[Oliveira+, HPCA'24]

DRAM Data Transfer (I)

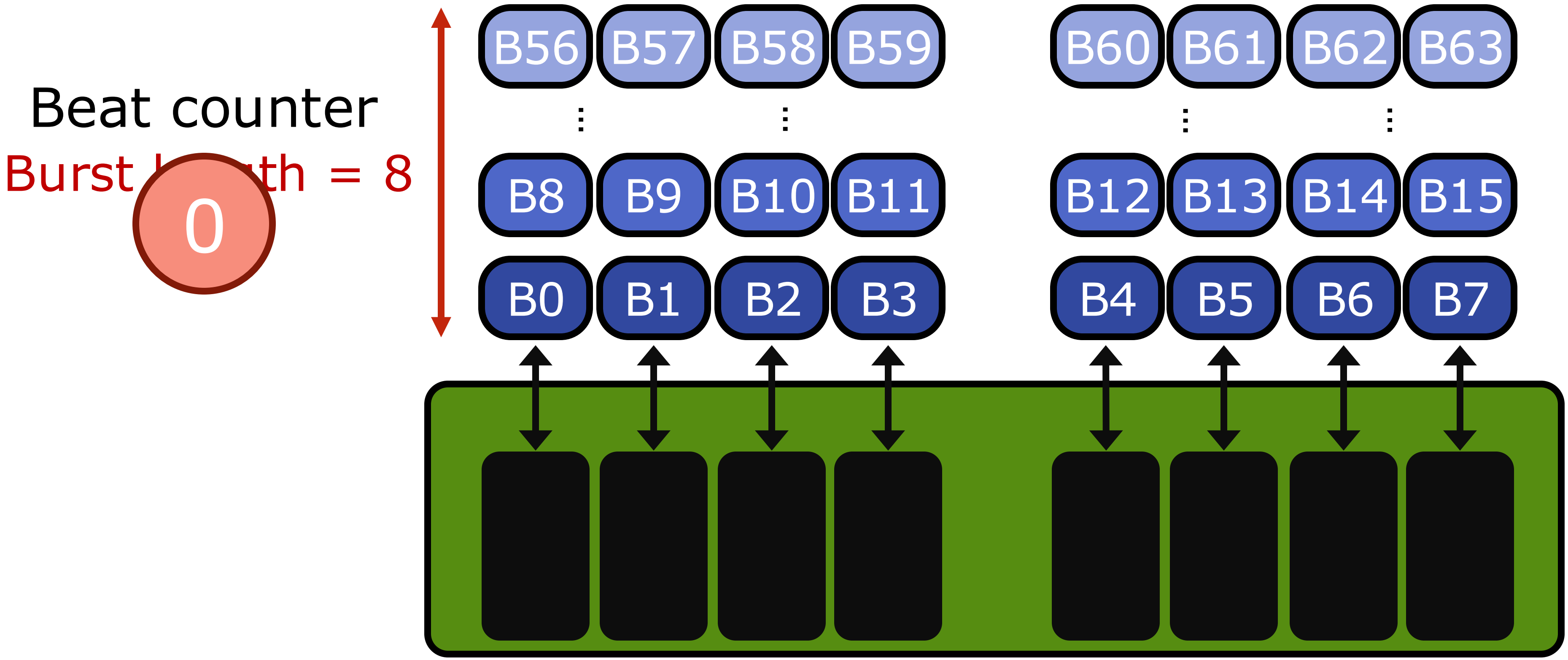
- DRAM data transfer happens in cache block granularity



[Seshadri+, MICRO'13]

DRAM Data Transfer (I)

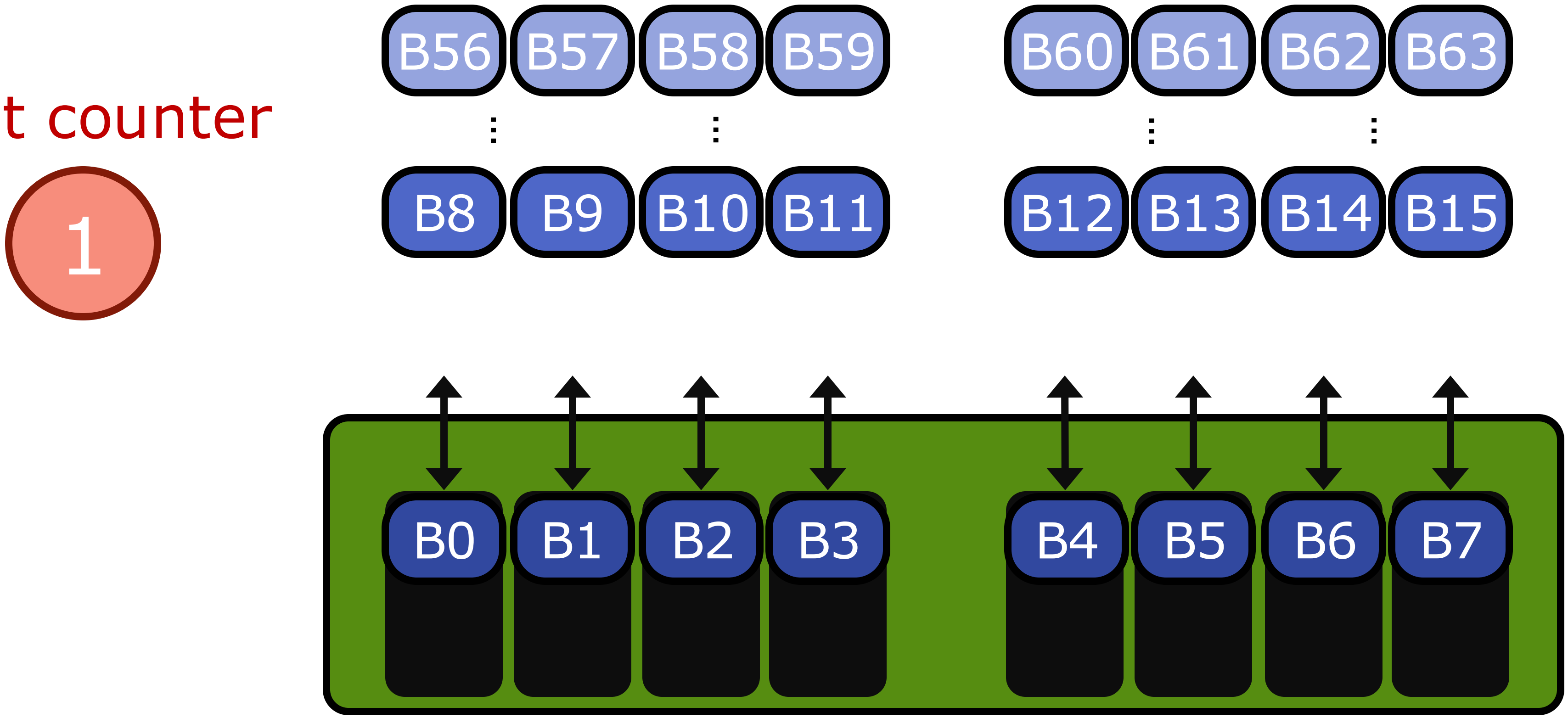
- DRAM data transfer happens in cache block granularity
- Using data transfer bursts (or bursts)



DRAM Data Transfer (I)

- DRAM data transfer happens in cache block granularity
- Using data transfer bursts (or bursts)

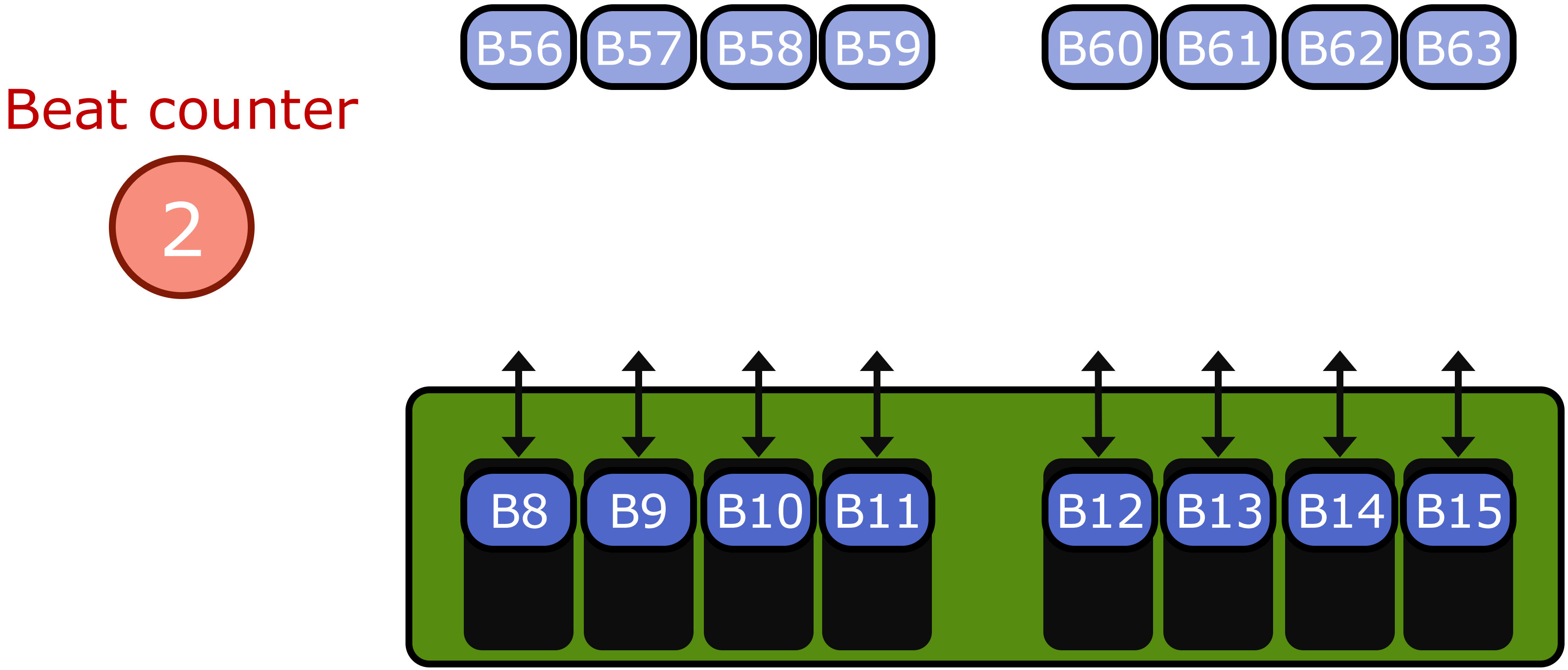
Beat counter



[Seshadri+, MICRO'13]

DRAM Data Transfer (I)

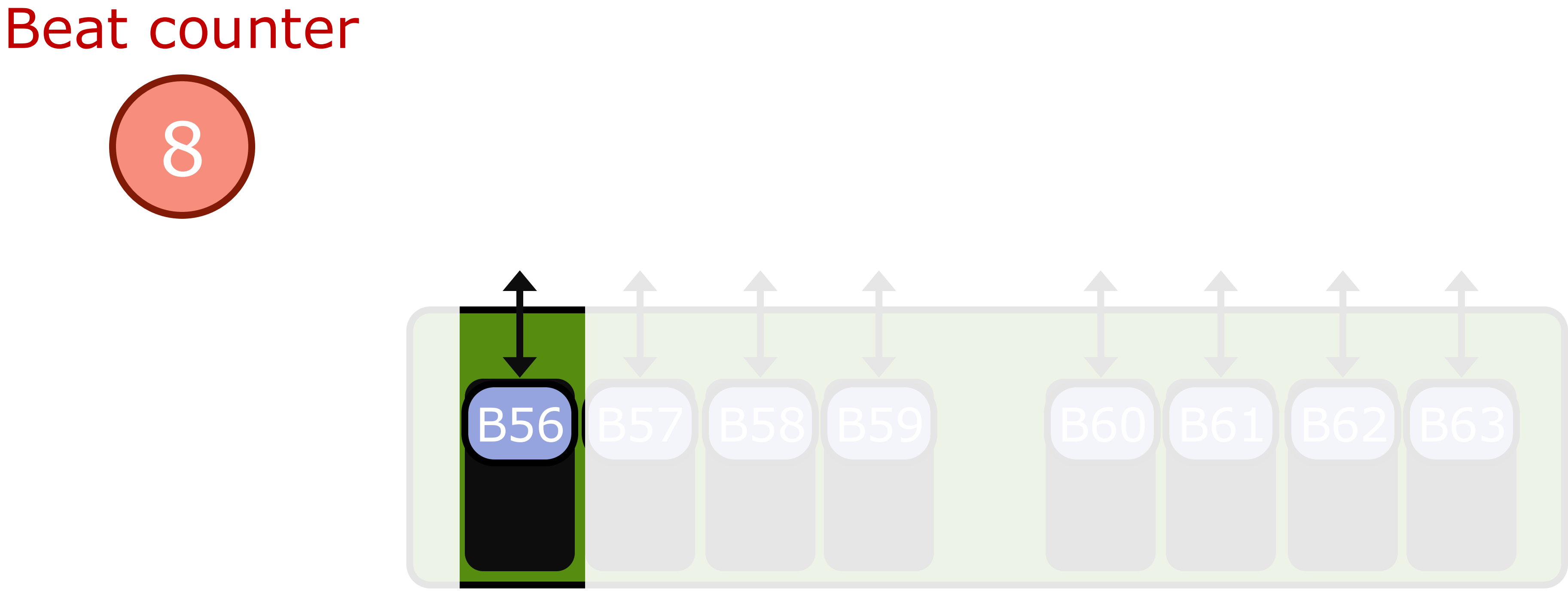
- DRAM data transfer happens in cache block granularity
- Using data transfer bursts (or bursts)



[Seshadri+, MICRO'13]

DRAM Data Transfer (I)

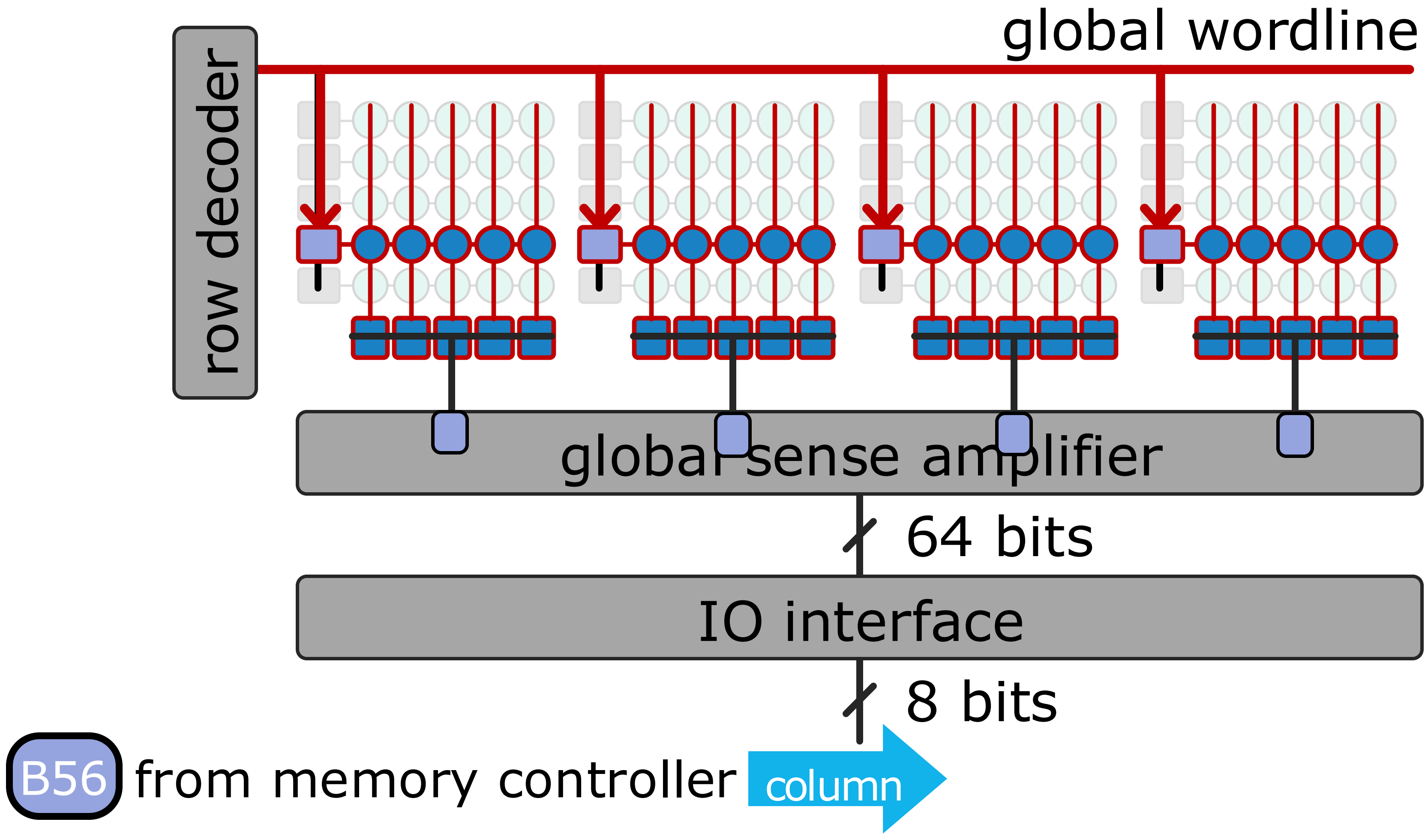
- DRAM data transfer happens in cache block granularity
- Using data transfer bursts (or bursts)



[Seshadri+, MICRO'13]

DRAM Data Transfer (II)

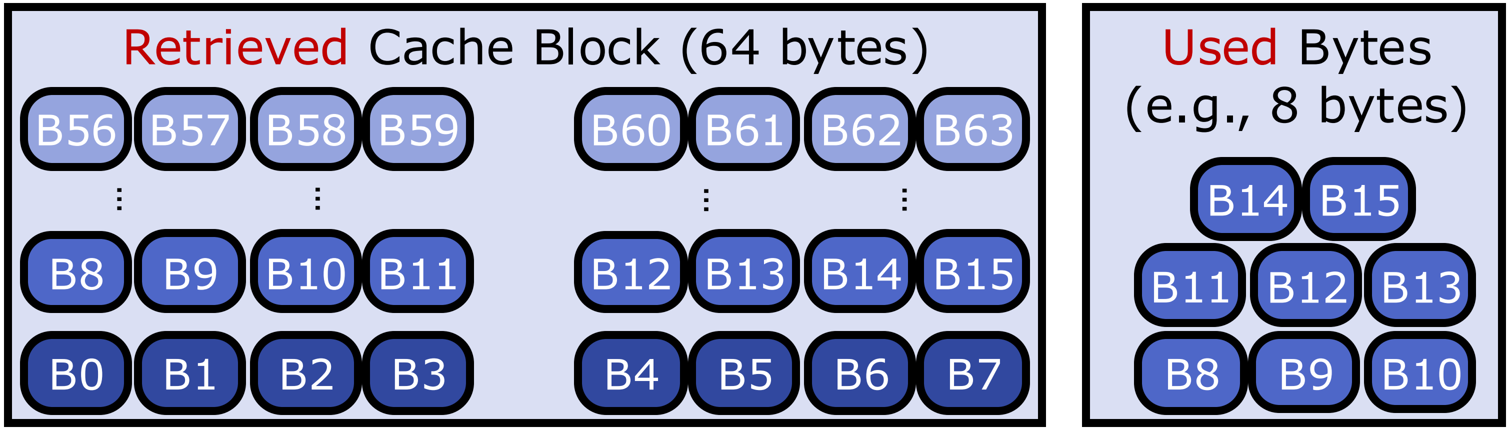
- Bits of a burst split across DRAM mats



[Oliveira+, HPCA'24]

Coarse-Grained DRAM Data Transfer Wastes Energy

- Retrieve more bytes than necessary with each word (e.g., 8 bytes) access

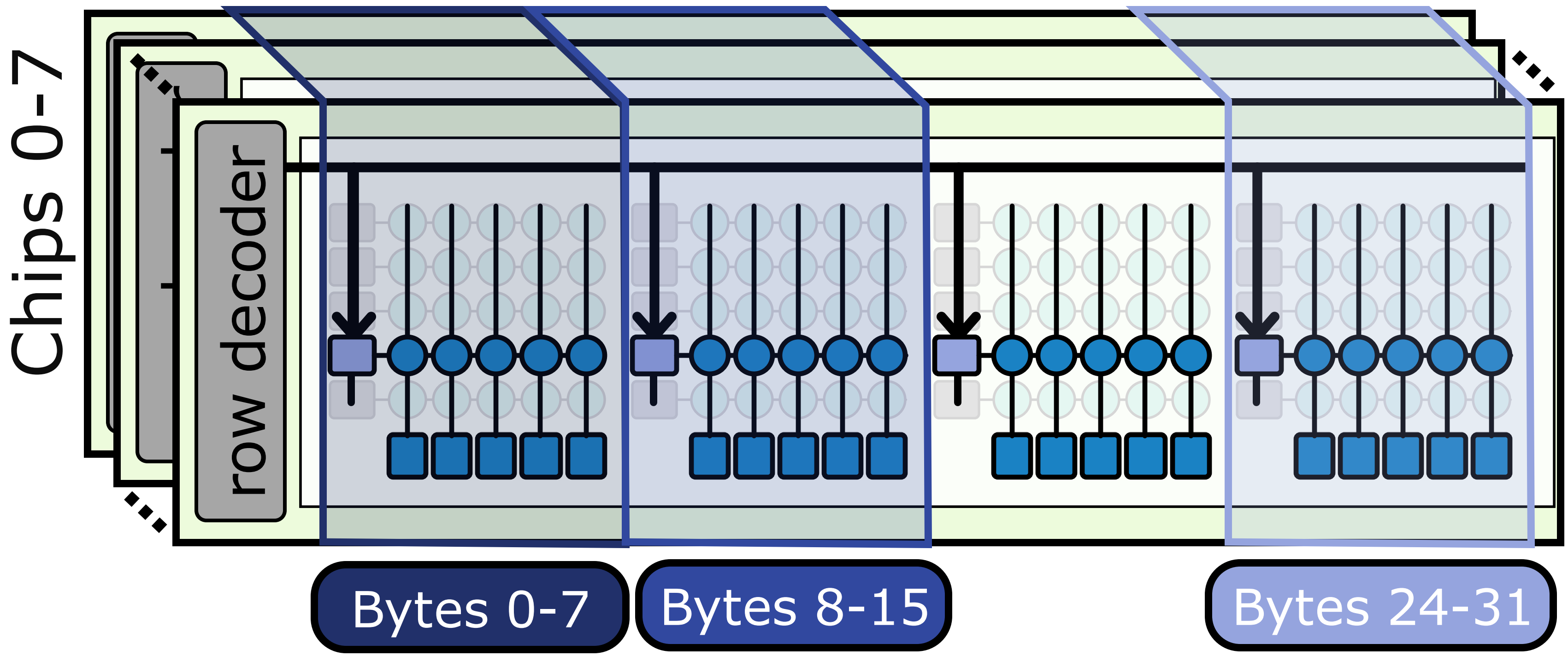


- Goal: Exploit spatial locality
- Problem: Not all words in a cache block are referenced by CPU load/store instructions

Less than 60% of words used on average (e.g., [Qureshi+, HPCA'07])

Coarse-Grained DRAM Row Activation Wastes Energy

- Activate more mats than necessary with each DRAM row activation

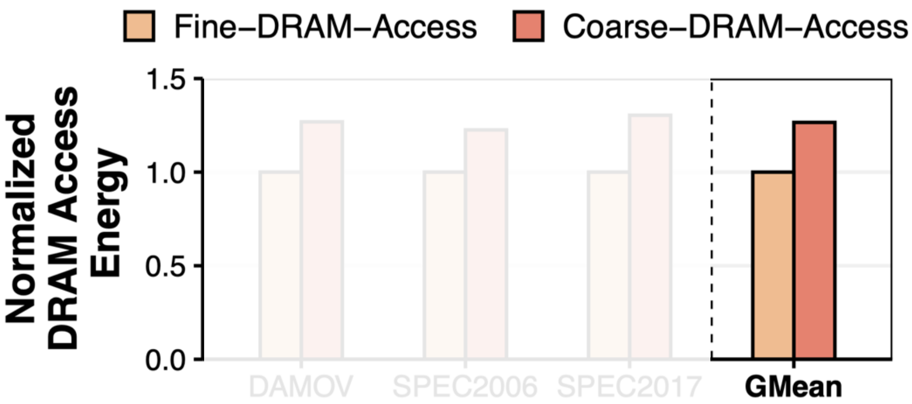


- Goal: Transfer in a burst, all words of a cache block
- Problem: Not all mats need to be read or updated

Fine-Grained DRAM Can Greatly Improve System Energy Efficiency

Fine-DRAM-Access: Enable word-sized (8-byte) data transfers

Fine-DRAM-Act: Enable per-mat DRAM row activation



Fine-Grained DRAM can improve READ/WRITE (ACTIVATE) energy by 27% (4%)

Challenges of Enabling Fine-Grained DRAM

Prior works
FGA
SBA
HalfDRAM
HalfPage
PRA

1. Maintaining high DRAM data transfer throughput
2. Incurring low DRAM area overhead
3. Fully exploiting fine-grained DRAM

Problem and Goal

1. Maintaining high DRAM data transfer throughput
2. Incurring low DRAM area overhead
3. Fully exploiting fine-grained DRAM

Problem

No prior work overcomes all three challenges

Goal

Develop a new, low-cost, and high-throughput DRAM substrate that can mitigate the excessive energy consumption of coarse-grained DRAM

Outline

1. Background & Motivation

2. Sectored DRAM: Design

3. Sectored DRAM: System Integration

4. Evaluation

5. Conclusion

Two Key Design Components

Two key observations regarding DRAM chip design enable Sectored DRAM at low cost

- **Observation:** DRAM mats naturally split DRAM rows into small fixed-size portions

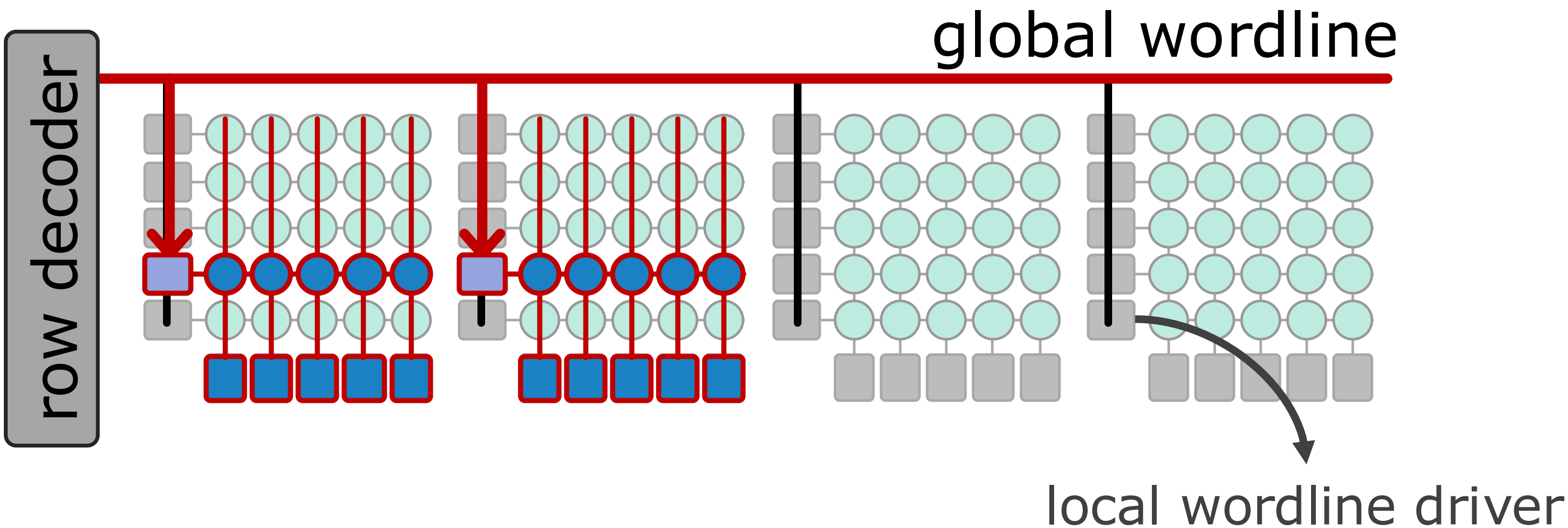
1 Sectored Activation (SA)

- **Observation:** DRAM I/O circuitry can already transfer a small portion of a cache block in one beat

2 Variable Burst Length (VBL)

Component 1: Sectored Activation

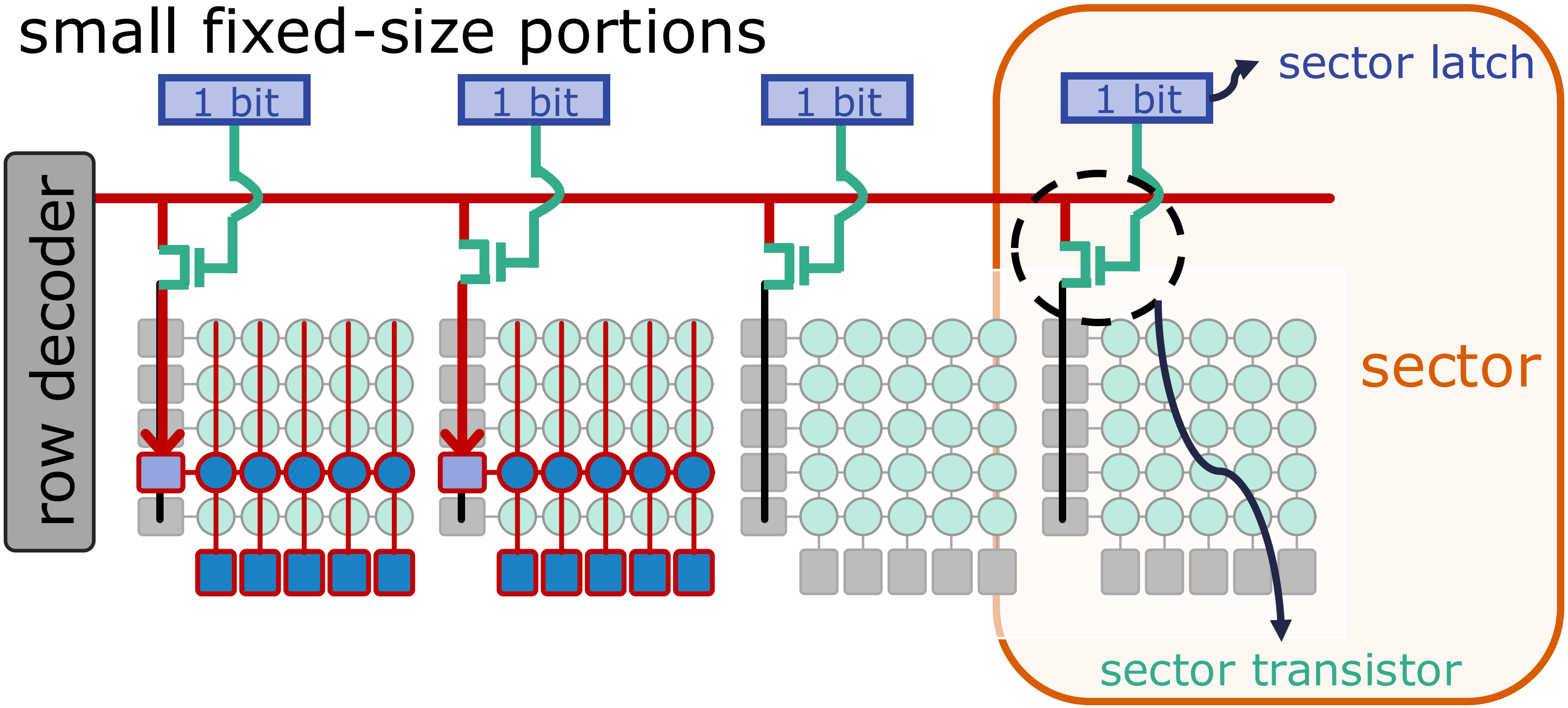
- **Observation:** DRAM mats naturally split DRAM rows into small fixed-size portions



- To select and activate one or multiple mats:
    1. Isolate the global wordline from local wordline drivers

Component 1: Sectored Activation

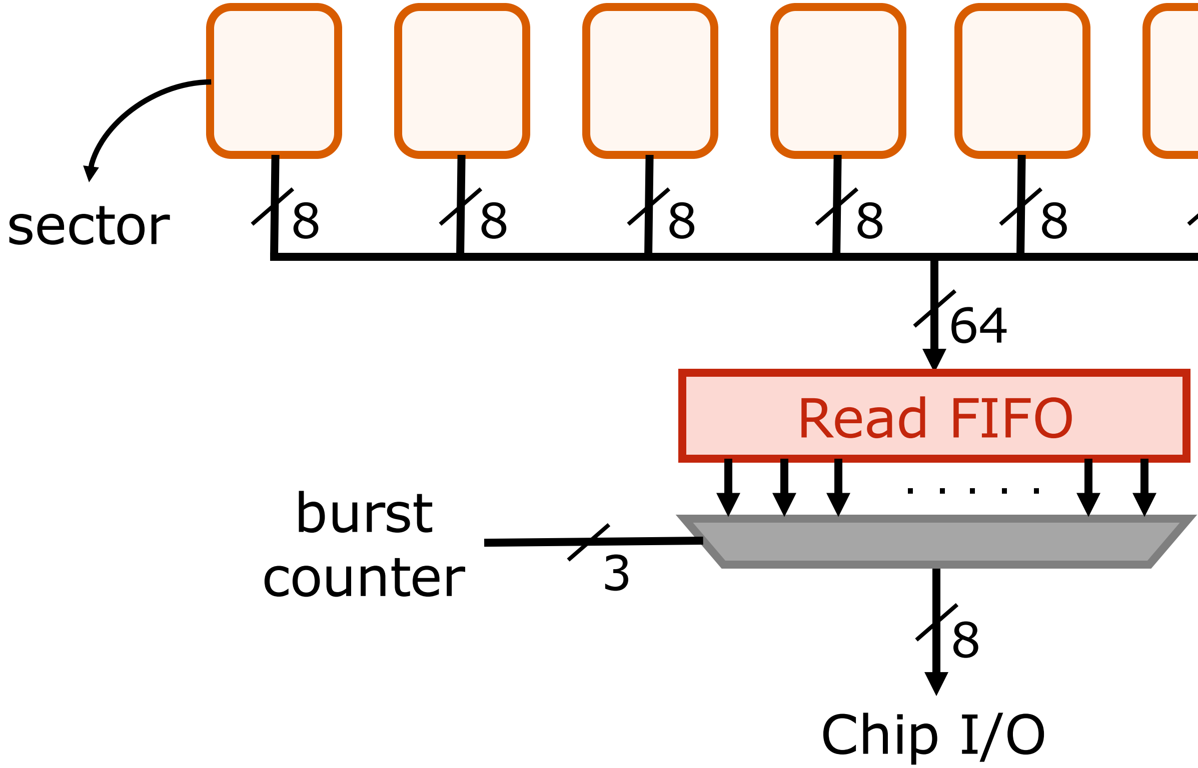
- **Observation:** DRAM mats naturally split DRAM rows into small fixed-size portions



- To select and activate one or multiple mats:
  1. Isolate the global wordline from local wordline drivers
  2. Add a control signal (1 bit) for each mat

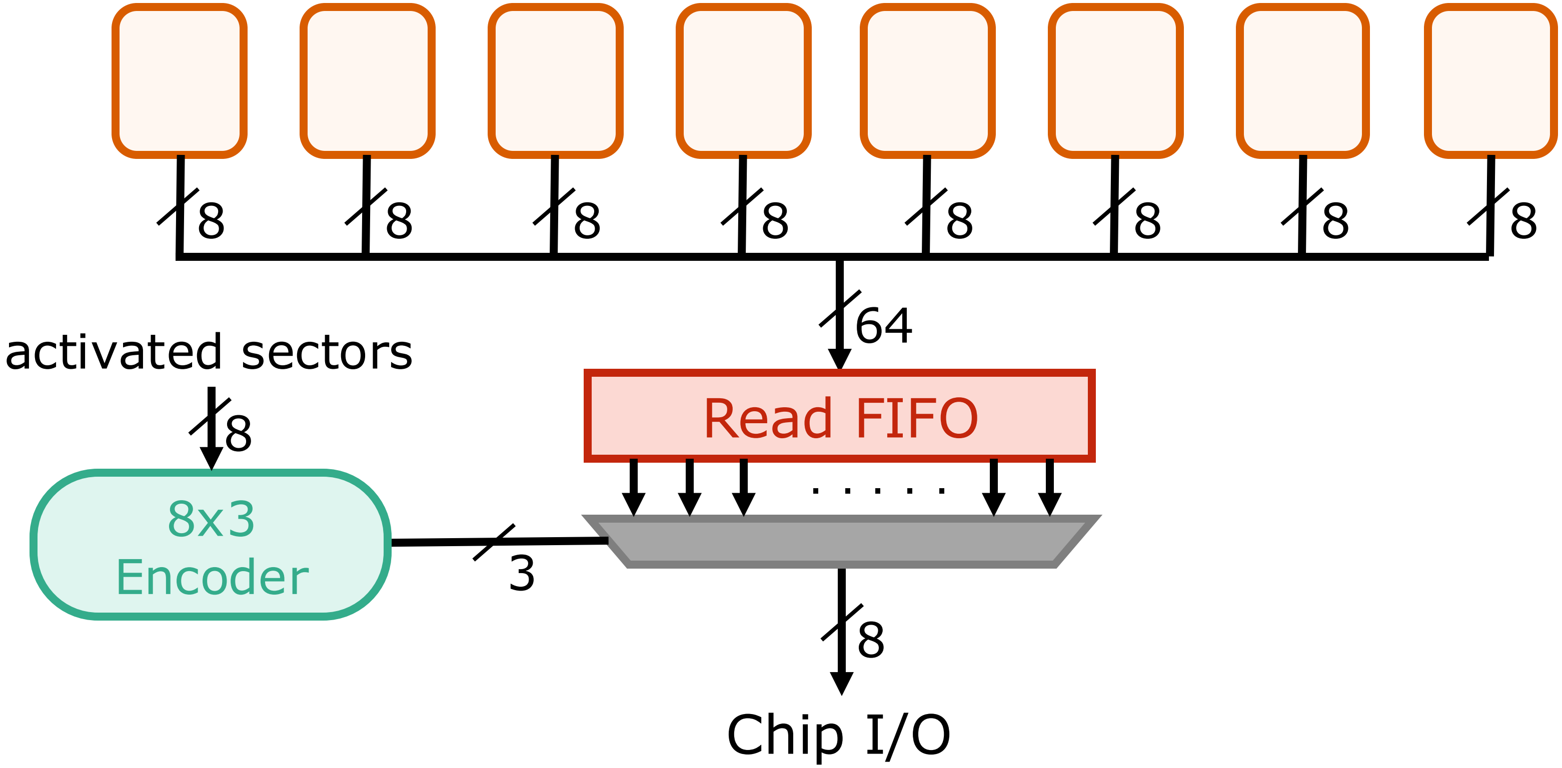
Component 2: Variable Burst Length

- **Observation:** DRAM I/O circuitry can already transfer a small portion of a cache block in one beat



Component 2: Variable Burst Length

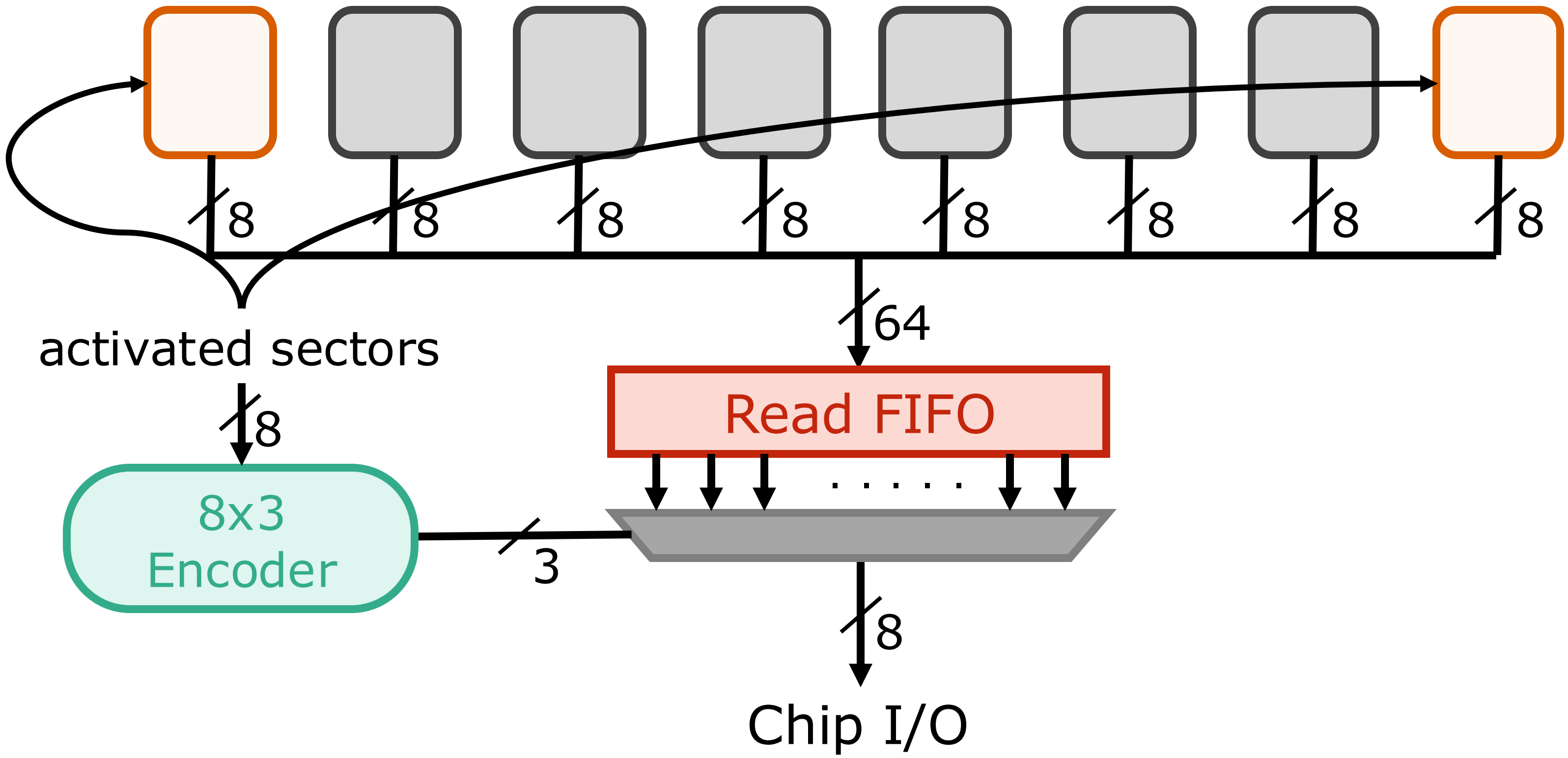
- **Observation:** DRAM I/O circuitry can already transfer a small portion of a cache block in one beat



- Replace the burst counter with an encoder that selects only the open/activated sectors

Component 2: Variable Burst Length

- **Observation:** DRAM I/O circuitry can already transfer a small portion of a cache block in one beat



- Replace the burst counter with an encoder that selects only the open/activated sectors

A memory controller can leverage Sectored DRAM without any physical DRAM interface modifications

Sectored DRAM: A Practical Energy-Efficient and High-Performance Fine-Grained DRAM Architecture

Ataberk Olgun[§] F. Nisa Bostancı[§†] Geraldo F. Oliveira[§] Yahya Can Tuğrul[§†] Rahul Bera[§]
A. Giray Yağlıkcı[§] Hasan Hassan[§] Oğuz Ergin[†] Onur Mutlu[§]

[§]*ETH Zürich* [†]*TOBB University of Economics and Technology*

Modern computing systems access data in main memory at coarse granularity *(e.g., at 512-bit cache block granularity). Coarse-grained access leads to wasted energy because the system does not use all individually accessed small portions (e.g.,* words, *each of which typically is 64 bits) of a cache block. In modern DRAM-based computing systems, two key coarse-grained access mechanisms lead to wasted energy: large and fixed-size (i) data transfers between DRAM and the memory controller and (ii) DRAM row activations.*

We propose Sectored DRAM, a new, low-overhead DRAM substrate that reduces wasted energy by enabling fine-grained *DRAM data transfer and DRAM row activation. To retrieve only useful data from DRAM, Sectored DRAM exploits the observation that many cache blocks are not fully utilized in many workloads due to poor spatial locality. Sectored DRAM predicts the words in a cache block that will likely be accessed during the cache block's*

1. Introduction

DRAM [22] is hierarchically organized to improve scaling in density and performance. At the highest level of the hierarchy, a DRAM chip is partitioned into banks that can be accessed simultaneously [87, 57, 58, 59, 63]. At the lowest level, a collection of DRAM rows (DRAM cells that are activated together) are typically divided into multiple *DRAM mats* that can operate individually [52, 42, 125, 58]. Even though DRAM chips are hierarchically organized, standard DRAM interfaces (e.g., DDRx [43, 44, 45]) do *not* expose DRAM mats to the memory controller. To access even a single DRAM cell, the memory controller needs to activate a large number of DRAM cells (e.g., 65,536 DRAM cells in a DRAM row in DDR4 [80]) and transfer many bits (e.g., a cache block, typically 512 bits [32]) over the memory channel. Thus, in current systems, both DRAM data transfer and activation are *coarse-grained.* Coarse-grained data

- Low overhead popcount circuitry to count set (logic-1) sector bits

https://arxiv.org/pdf/2207.13795.pdf

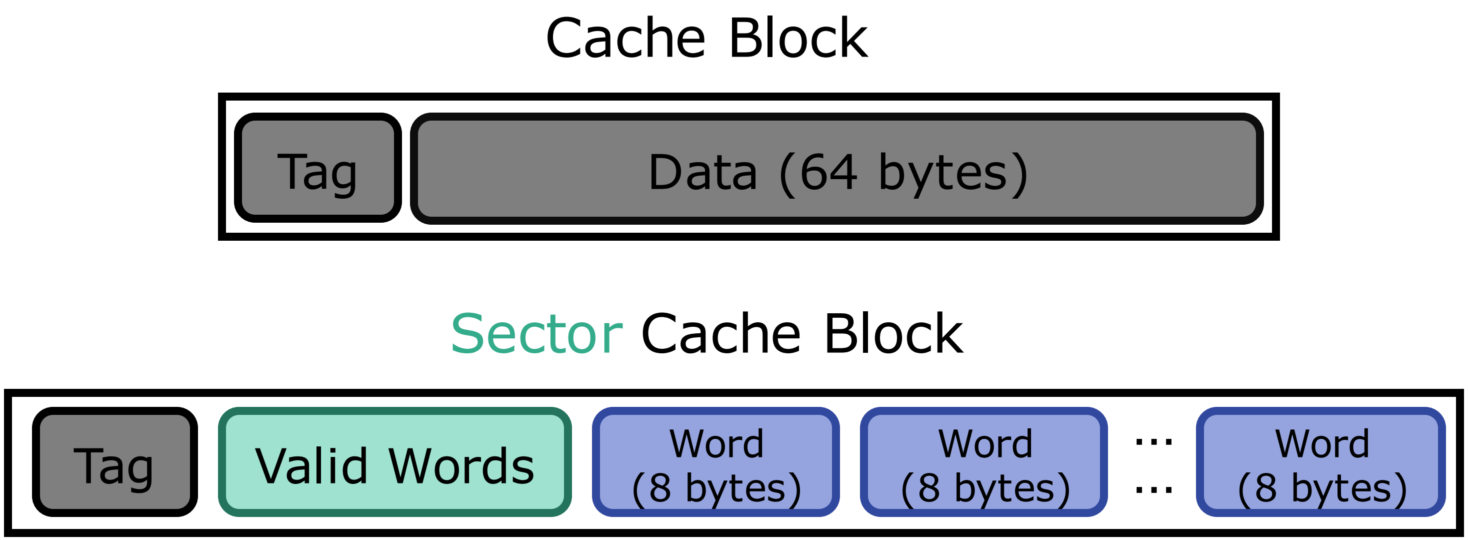
Outline

Efficient System Integration of Sectored DRAM is Challenging (I)

Challenge 1: Requires system-wide modifications to enable sub-cache-block (e.g., word) granularity data transfers

Solution: Use sector caches (e.g., [Liptay+,1968])

- Extend a cache block with 1 bit for each word
- A bit indicates if its corresponding word is valid

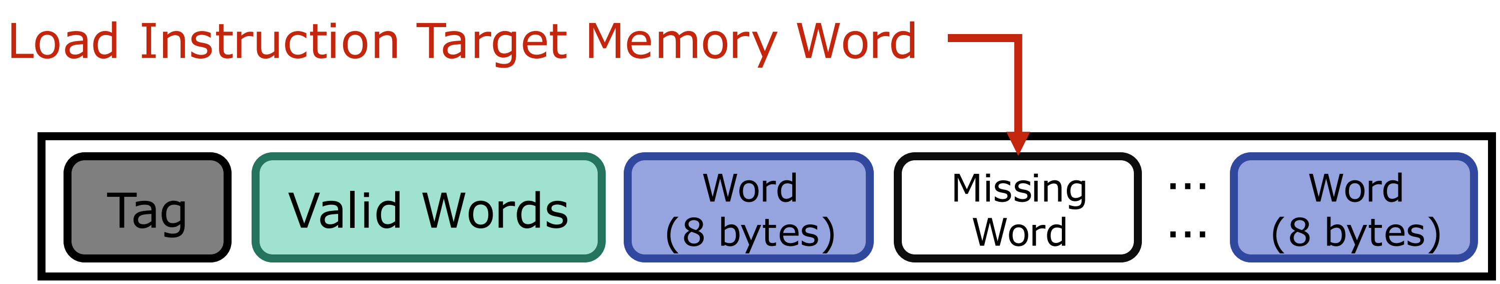


Efficient System Integration of Sectored DRAM is Challenging (II)

Challenge 2: Missing words (sectors) in a cache block cause additional performance overhead

Solution: Develop two prediction techniques

1) A technique to exploit the spatial locality in subsequent load/store (LD/ST) instructions
2) A spatial pattern predictor (e.g., [Kumar+,1998]) tailored for predicting useful words (similar to [Yoon+, 2012])



Efficient System Integration of Sectored DRAM is Challenging (II)

Challenge 2: Missing words (sectors) in a cache block cause additional performance overhead

Solution: Develop two prediction techniques

1) A technique to exploit the spatial locality in subsequent load/store (LD/ST) instructions

2) A spatial pattern predictor (e.g., [Kumar+,1998]) tailored for predicting useful words (similar to [Yoon+, 2012])

Load/Store Queue (LSQ) Lookahead

- One load/store instruction *references* one word in main memory
- **Key Mechanism:** 1) Collect references from *younger* load/store instructions
  2) store the collected references in the *oldest* load/store instr.

A load/store instruction retrieves all words in a cache block that will be referenced in the near future to the L1 cache with only one cache access

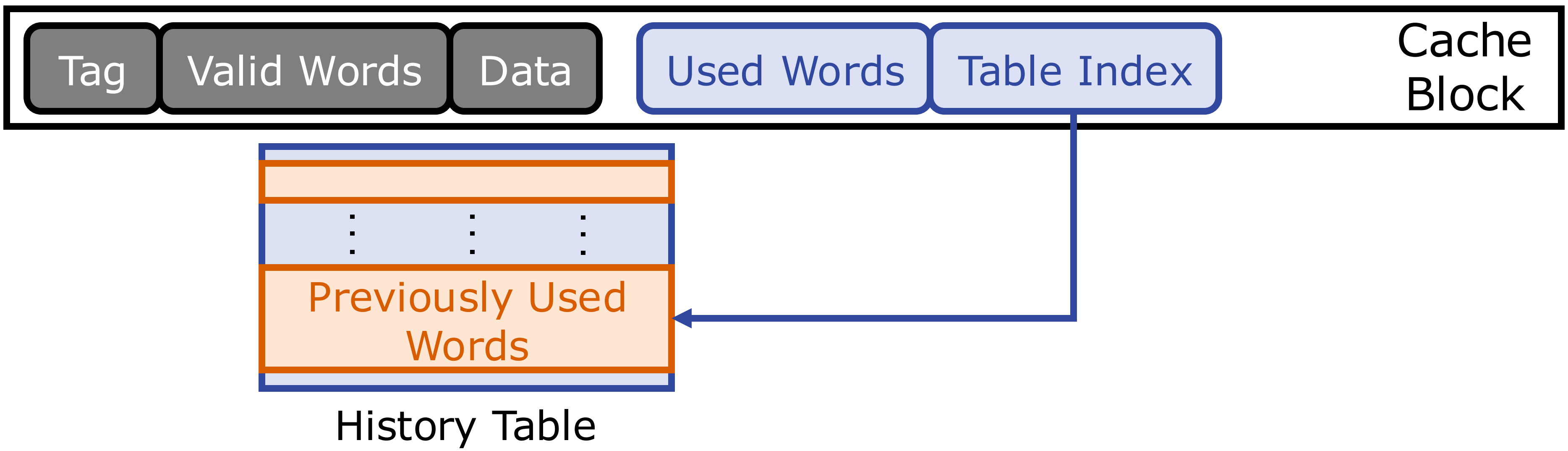
LSQ Lookahead has two key drawbacks

- LSQ is not large enough to store many LD/ST instructions
- Dependencies prevent computation of future LD/ST instruction addresses

Sector Predictor (SP)

Key Idea: Complement LSQ Lookahead and minimize sector misses

- Used (referenced) words in a cache block form a signature
- Reuse this signature when the same cache block misses in the cache



Outline

1. Background & Motivation

2. Sectored DRAM: Design

3. Sectored DRAM: System Integration

4. Evaluation

5. Conclusion

Evaluation Methodology

- **Performance and energy consumption evaluation:**
  Cycle-level simulations using Ramulator
  Rambus Power Model and DRAMPower for DRAM energy
  CACTI & McPAT for processor energy estimation

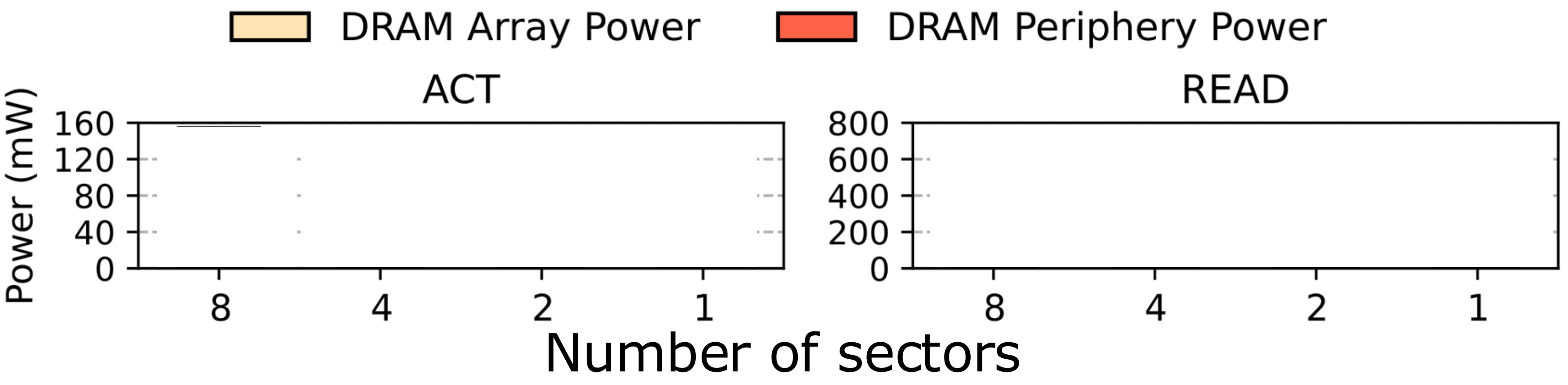
- **System Configuration:**

Processor	1-16 cores, 3.6GHz clock frequency, 4-wide issue, 128-entry instruction window 32 KiB L1, 256 KiB L2, and 8 MiB L3 caches
DRAM	DDR4, 1-4 channel, 4 rank/channel, 4 bank groups, 4 banks/bank group, 32K rows/bank, 3200 MT/s
Memory Ctrl.	64-entry read and write requests queues, Scheduling policy: FR-FCFS with a column cap of 16

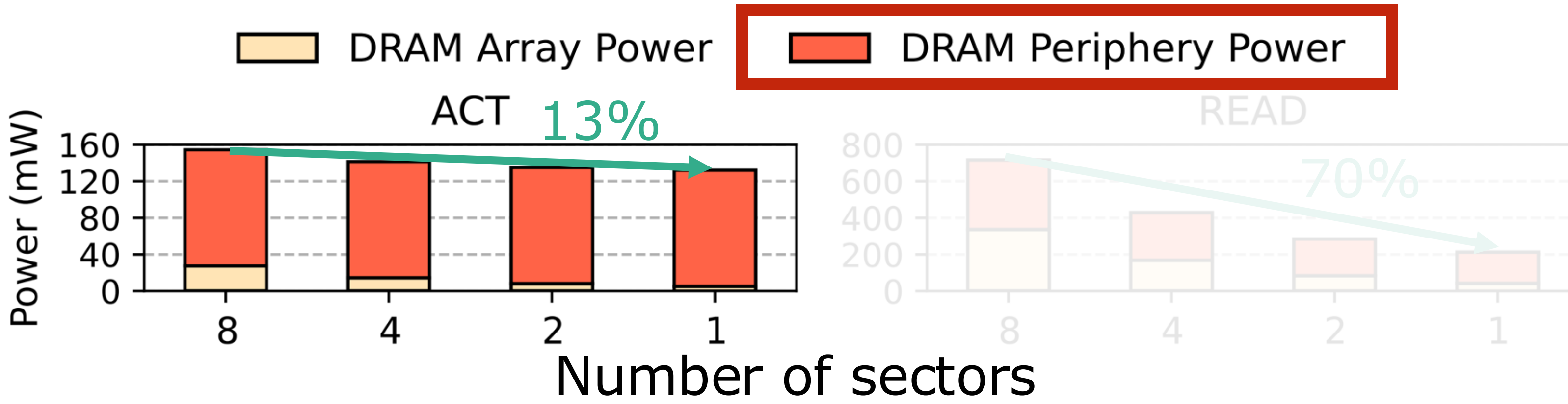
- **Comparison Points:** 3 state-of-the-art fine-grained DRAM mechanisms
  - HalfDRAM (best performing), Fine-Grained Activation (lowest area overhead), and Partial Row Activation

- **Workloads:** 41 1-,2-,4-,8-,16-core (multiprogrammed) workloads
  - SPEC CPU2006, SPEC CPU2017, DAMOV benchmark suites

Sectored DRAM Can Greatly Reduce DRAM ACT and READ Power



Sectored DRAM Can Greatly Reduce DRAM ACT and READ Power



Reading from (activating) one sector takes 70% (13%) less power than reading from (activating) all 8 sectors

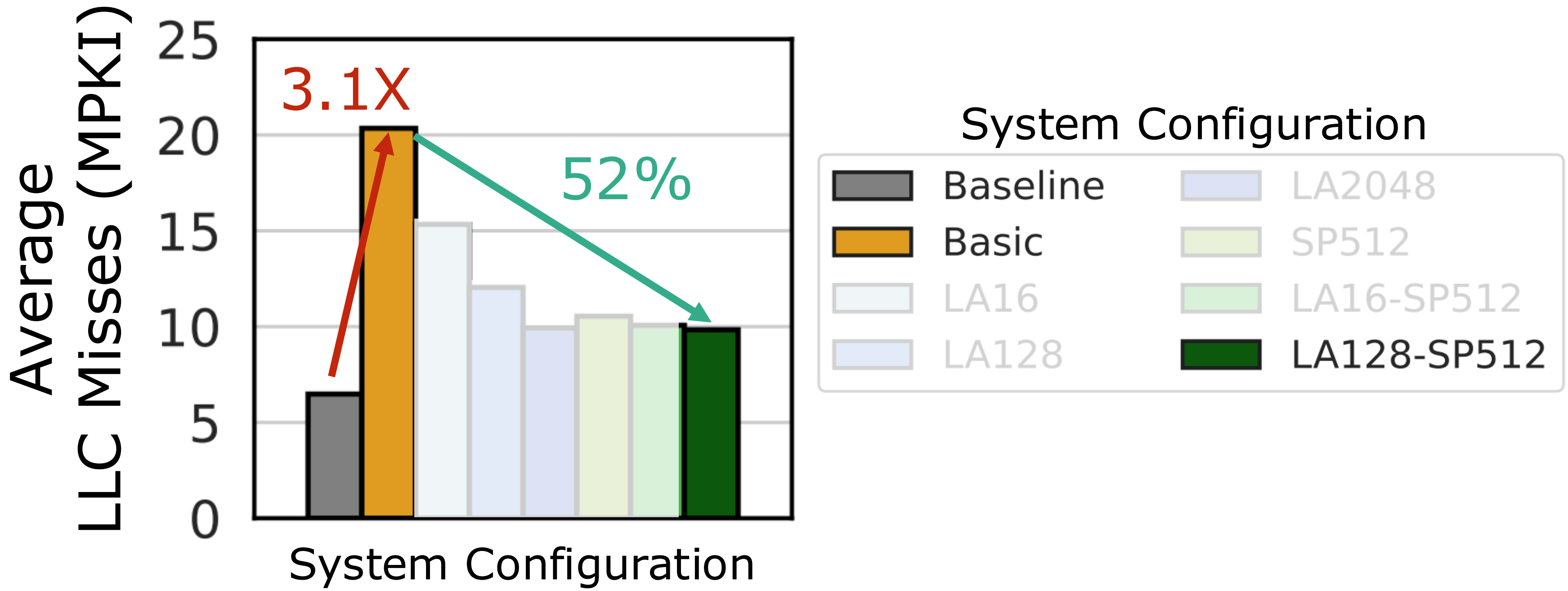
ACT power is dominated by periphery power not affected by the number of sectors activated

Number of Sector Misses

Basic = Sectored DRAM without any sector prediction

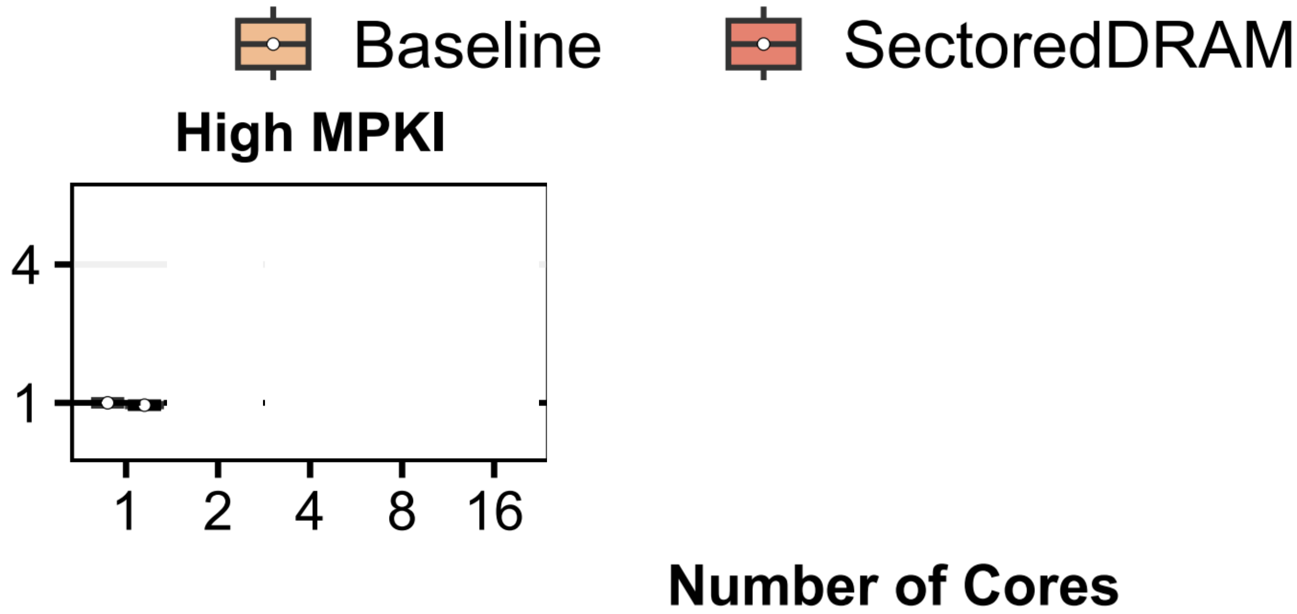
LA<N> = LSQ Lookahead with N LSQ entries

SP512 = Sector Predictor with a history table size of 512

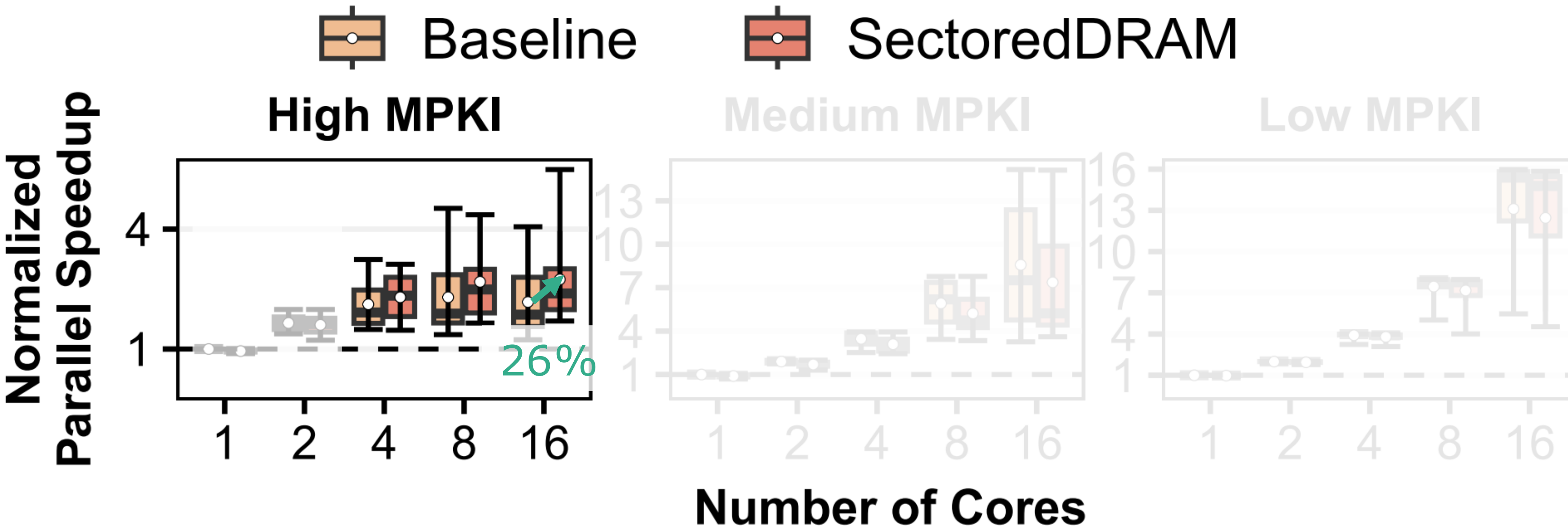


LSQ Lookahead 128 with SP 512
minimizes the LLC misses caused by sector misses

Speedup

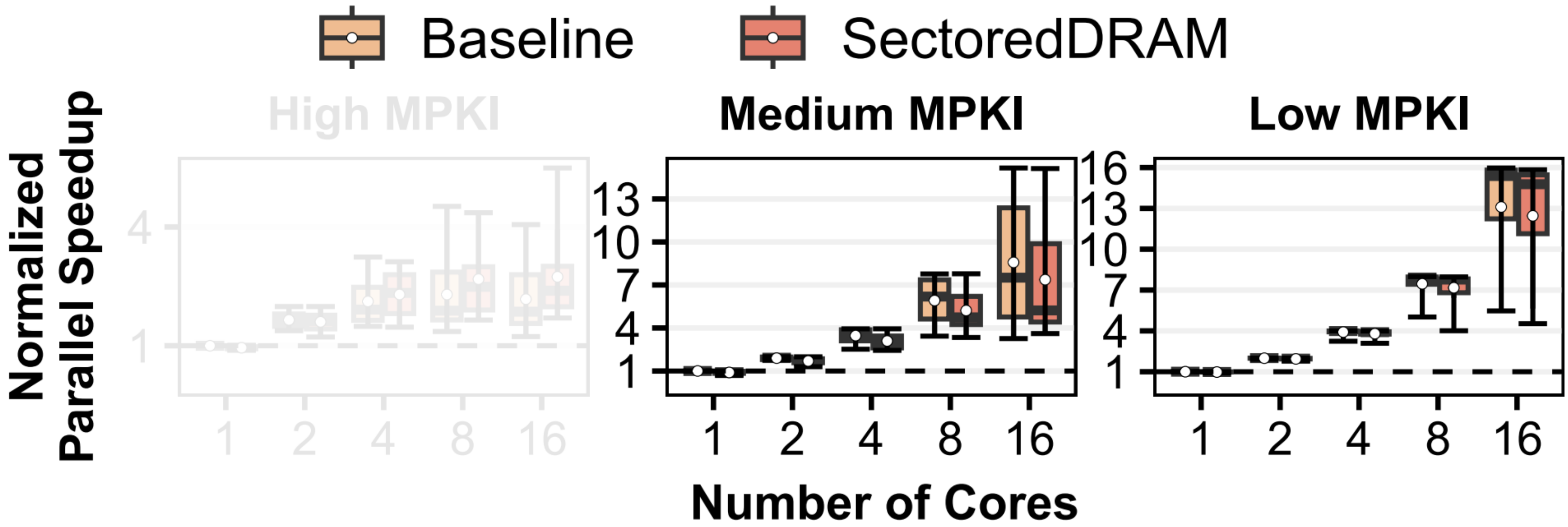


Speedup



Sectored DRAM provides significant speedups for highly memory intensive workloads at core count > 2

Speedup

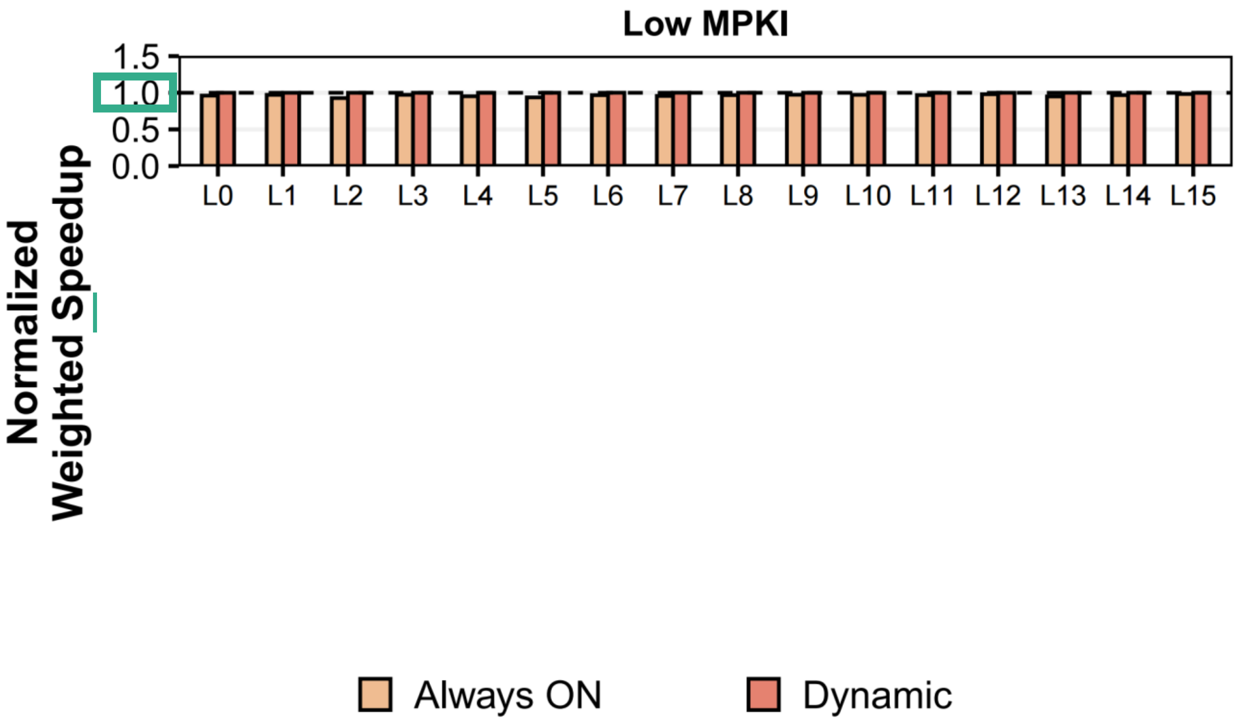


Sectored DRAM provides significant speedups for highly memory intensive workloads at core count > 2

Sectored DRAM provides smaller parallel speedup than Baseline for non-memory-intensive workloads

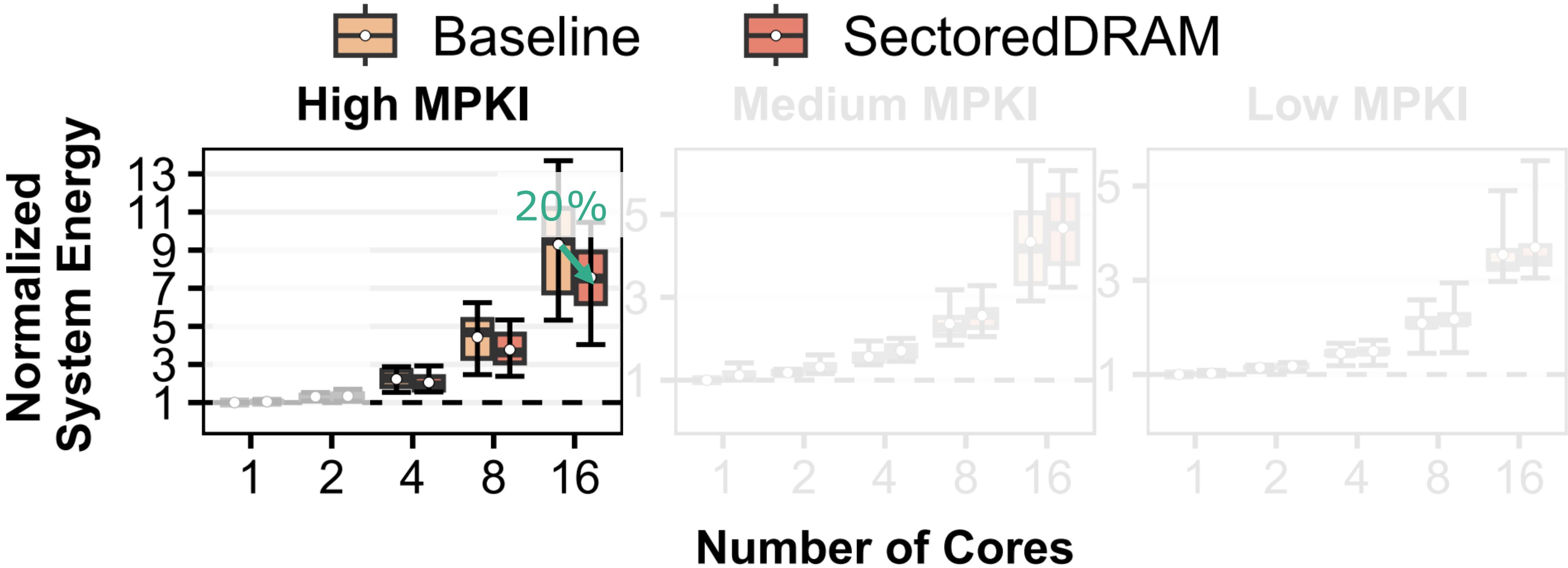
Performance Degradation for Non-Memory-Intensive Workloads

- Fetch all sectors of a cache block if the workload access pattern does not favor sub-cache-block data transfers
  - Based on average MPKI and thresholding



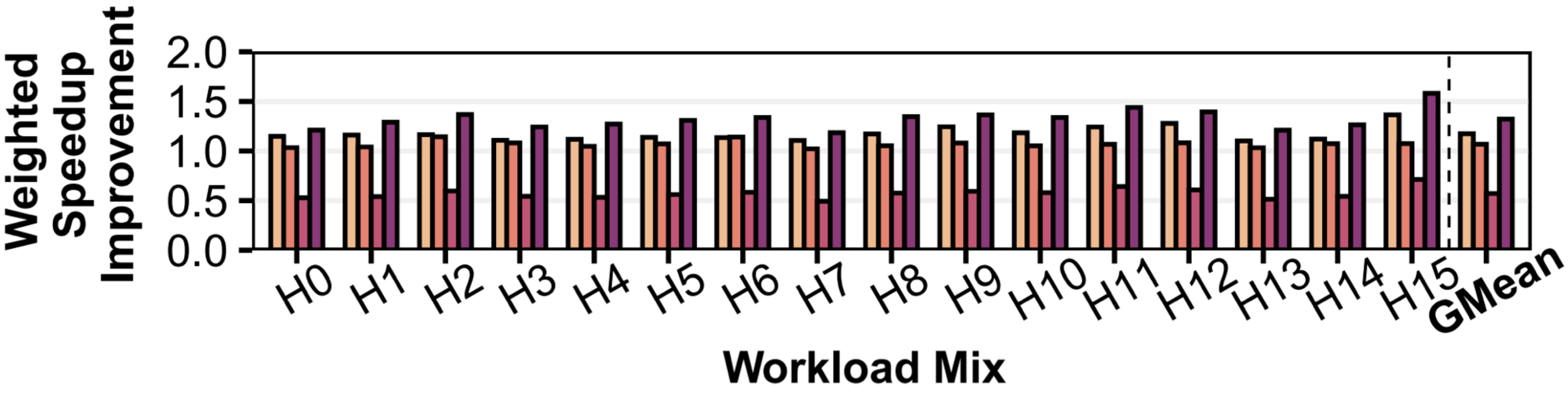
Dynamic policy overcomes the performance degradation in non-memory-intensive workloads

System Energy

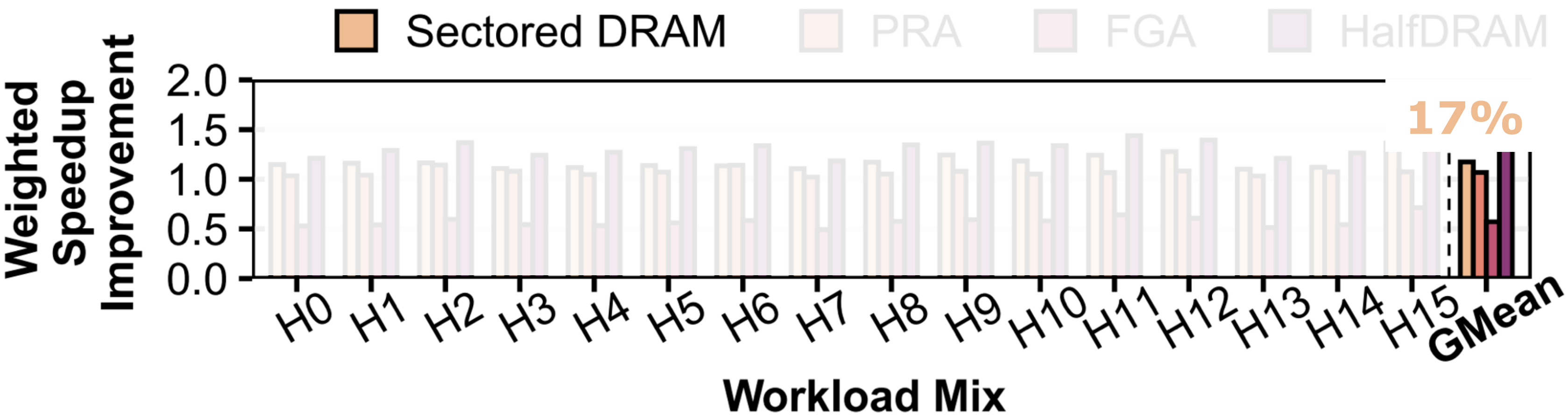


Sectored DRAM provides significant system energy savings for highly memory intensive workloads at core count > 2

Workload Mix Performance Comparison

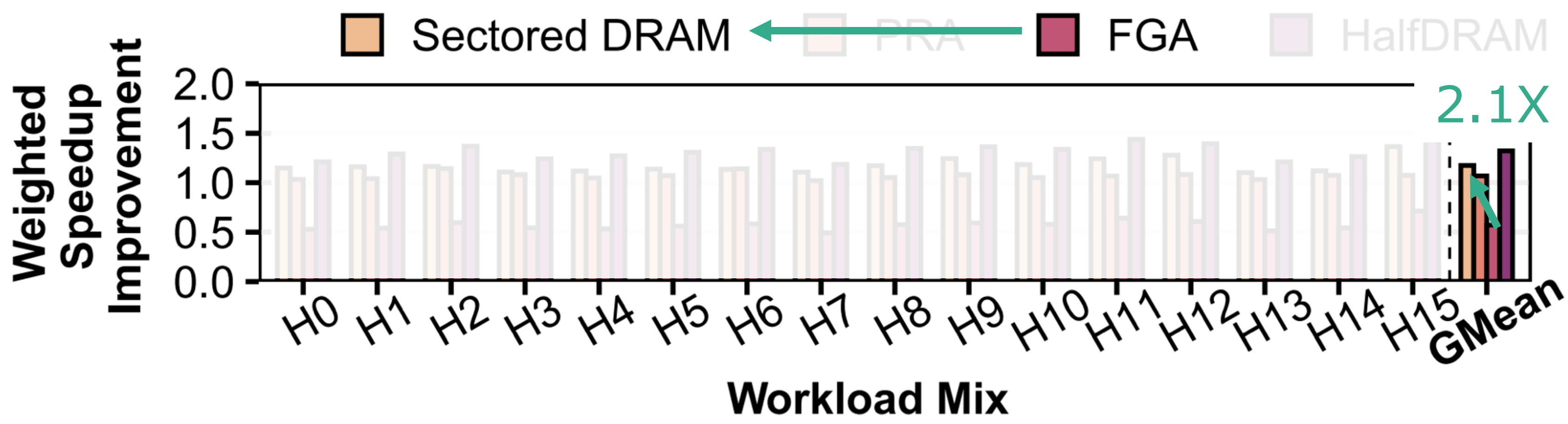


Workload Mix Performance Comparison



Sectored DRAM provides 17% average speedup across all mixes

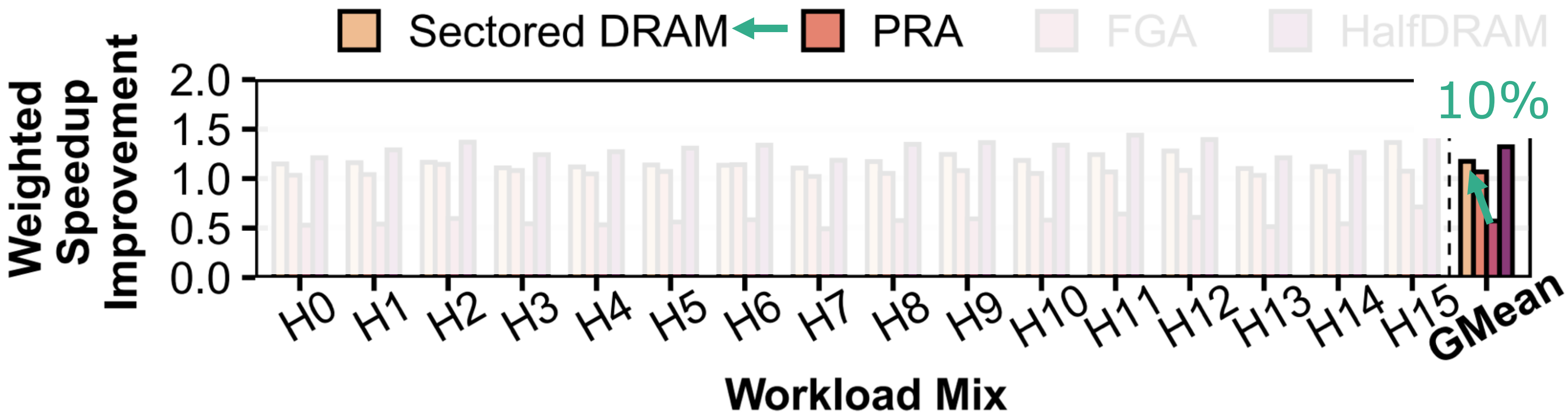
Workload Mix Performance Comparison



Sectored DRAM provides 17% average speedup across all mixes

Outperforms fine-grained activation by 2.1X

Workload Mix Performance Comparison

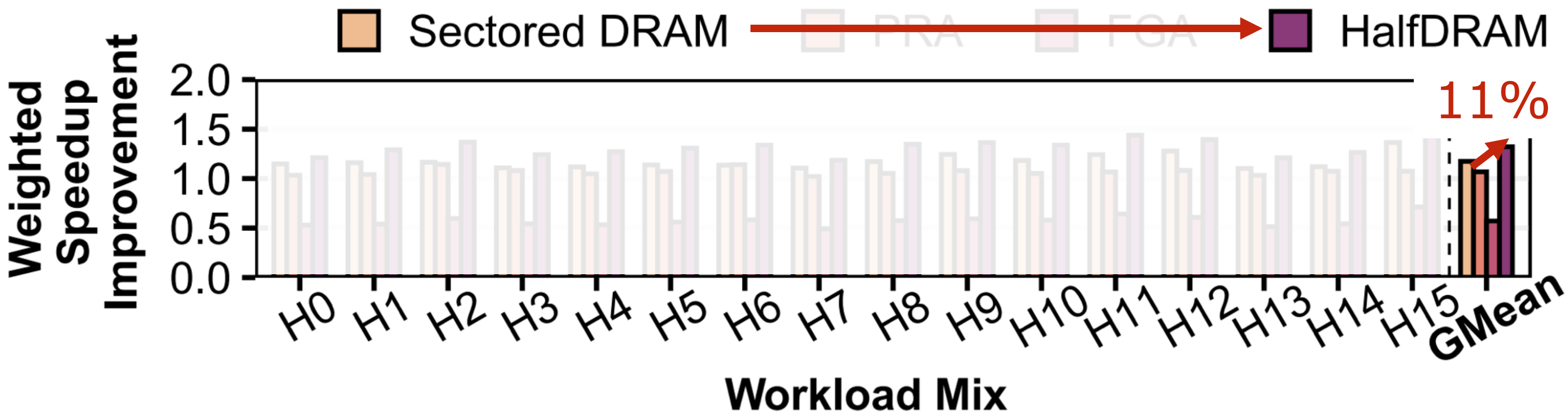


Sectored DRAM provides 17% average speedup across all mixes

Outperforms fine-grained activation by 2.1X

Outperforms Partial Row Activation by 10%

Workload Mix Performance Comparison



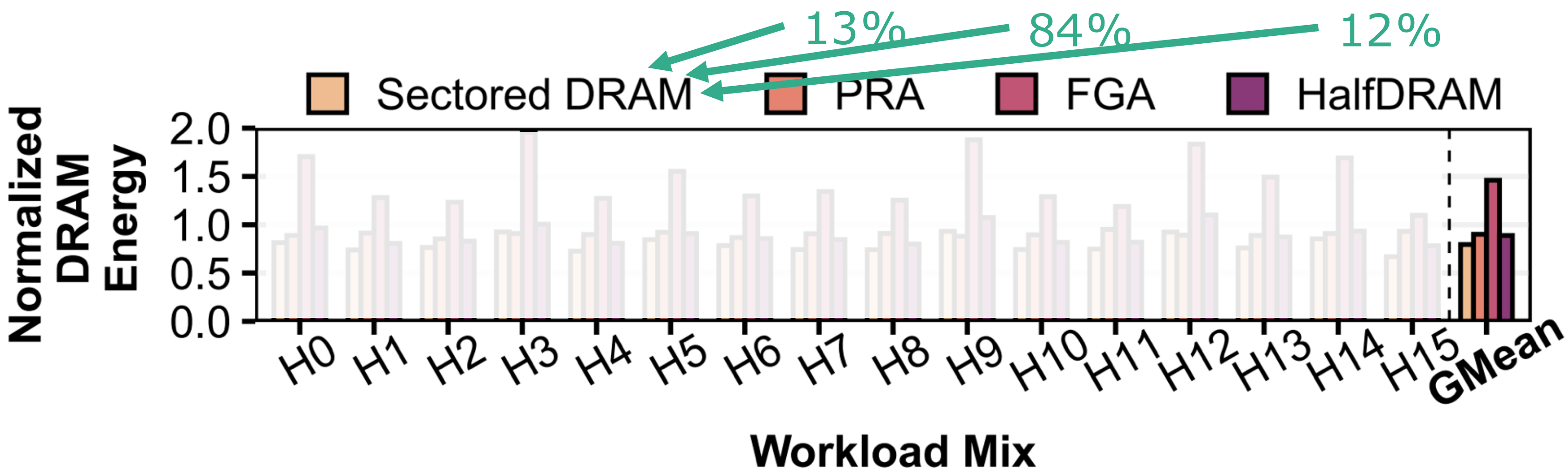
Sectored DRAM provides 17% average speedup across all mixes

Outperforms fine-grained activation by 2.1X

Outperforms Partial Row Activation by 10%

Performs within 11% of HalfDRAM

Workload Mix DRAM Energy Comparison



Sectored DRAM enables larger DRAM energy savings compared to prior works

Savings are attributed to
i) finer-grained data transfer and activation than HalfDRAM
ii) background power reduction compared to PRA and FGA

Area Overhead Estimation

DRAM

- Sector transistors, sector latches, wiring
- 8 additional local wordline driver stripes
- Model DRAM chip using CACTI
    - Sectored DRAM: 1.7% of DRAM chip area
    - Partial Row Activation and Fine Grained Activation: 1.7%
    - HalfDRAM: 2.6%

Processor

- Sector bits (indicate valid words): 1 byte/cache block
- Sector predictor: 1088 bytes/core
- Model processor storage area overhead using CACTI
    - 8-core processor area increases by 1.2%

More in the Paper

- Microbenchmark performance evaluation
  - Sectored DRAM greatly benefits random access workloads

- Performance & energy sensitivity analysis
  - Number of DRAM channels
  - Performance with prefetching enabled

- Discussion on
  - Finer-granularity sector support (i.e., >8 sectors)
  - Compatibility with DRAM Error Correcting Codes

More in the Paper

Sectored DRAM: A Practical Energy-Efficient and High-Performance Fine-Grained DRAM Architecture

Ataberk Olgun[§] F. Nisa Bostancı[§†] Geraldo F. Oliveira[§] Yahya Can Tuğrul[§†] Rahul Bera[§]
A. Giray Yağlıkçı[§] Hasan Hassan[§] Oğuz Ergin[†] Onur Mutlu[§]

[§]*ETH Zürich* [†]*TOBB University of Economics and Technology*

Modern computing systems access data in main memory at coarse granularity *(e.g., at 512-bit cache block granularity). Coarse-grained access leads to wasted energy because the system does* not *use all individually accessed small portions (e.g.,* words, *each of which typically is 64 bits) of a cache block. In modern DRAM-based computing systems, two key coarse-grained access mechanisms lead to wasted energy: large and fixed-size (i) data transfers between DRAM and the memory controller and (ii) DRAM row activations.*

We propose Sectored DRAM, a new, low-overhead DRAM substrate that reduces wasted energy by enabling fine-grained *DRAM data transfer and DRAM row activation. To retrieve only useful data from DRAM, Sectored DRAM exploits the observation that many cache blocks are not fully utilized in many workloads due to poor spatial locality. Sectored DRAM predicts the words in a cache block that will likely be accessed during the cache block's*

1. Introduction

DRAM [22] is hierarchically organized to improve scaling in density and performance. At the highest level of the hierarchy, a DRAM chip is partitioned into banks that can be accessed simultaneously [87, 57, 58, 59, 63]. At the lowest level, a collection of DRAM rows (DRAM cells that are activated together) are typically divided into multiple *DRAM mats* that can operate individually [52, 42, 125, 58]. Even though DRAM chips are hierarchically organized, standard DRAM interfaces (e.g., DDRx [43, 44, 45]) do *not* expose DRAM mats to the memory controller. To access even a single DRAM cell, the memory controller needs to activate a large number of DRAM cells (e.g., 65,536 DRAM cells in a DRAM row in DDR4 [80]) and transfer many bits (e.g., a cache block, typically 512 bits [32]) over the memory channel. Thus, in current systems, both DRAM data transfer and activation are *coarse-grained.* Coarse-grained data

- Discussion on
  - Finer-granularity sector support (i.e., >8 sectors)

https://arxiv.org/pdf/2207.13795.pdf

Outline

1. Background & Motivation

2. Sectored DRAM: Design

3. Sectored DRAM: System Integration

4. Evaluation

5. Conclusion

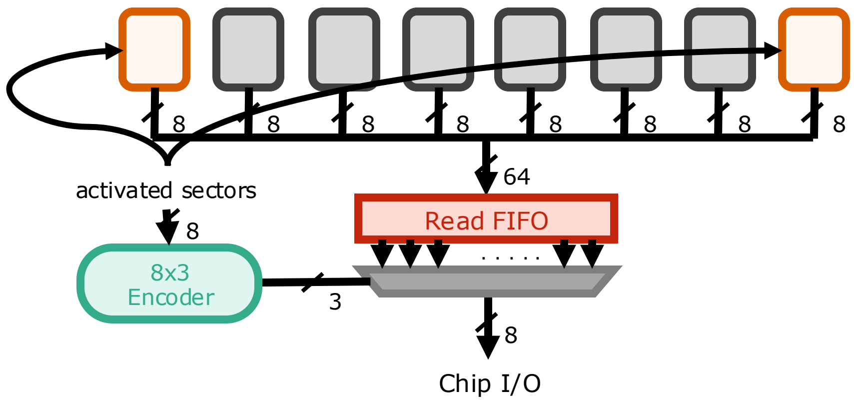
Sectored DRAM Conclusion

Designed a fine-grained, low-cost, and high-throughput DRAM substrate

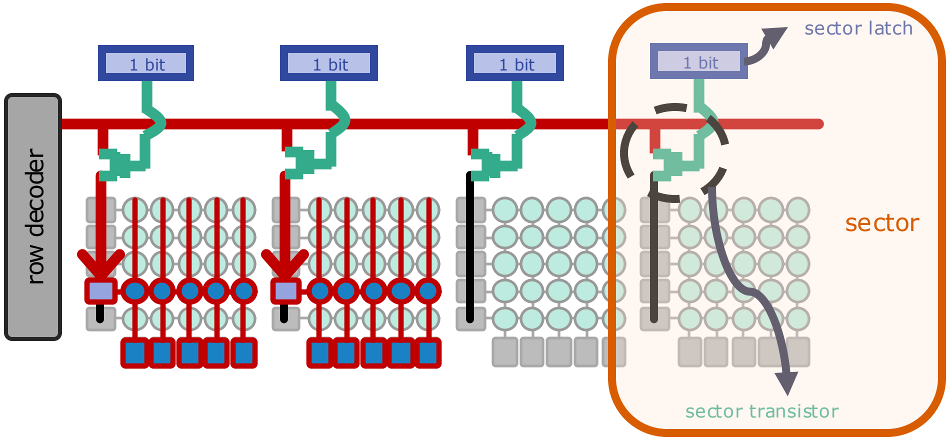
- Mitigates excessive energy consumption of coarse-grained DRAM

Key Ideas: Small modifications to memory controller and DRAM chip enable

Variable Burst Length



Sectored Activation



Key Results: For the evaluated memory-intensive workloads, Sectored DRAM

- Improves system energy consumption by 14%, system performance by 17%
- Incurs 0.39 mm^2 (1.7%) DRAM chip area overhead
- Performs within 11% of a state-of-the-art prior work (Half-DRAM), with 12% less DRAM energy and 34% less area overhead

Sectored DRAM is Published in ACM TACO

https://dl.acm.org/doi/abs/10.1145/3673653

RESEARCH-ARTICLE | OPEN ACCESS

Just Accepted

Sectored DRAM: A Practical Energy-Efficient and High-Performance Fine-Grained DRAM Architecture

Authors: Ataberk Olgun, Fatma Bostanci, Geraldo Francisco de Oliveira Junior, Yahya Can Tugrul, Rahul Bera, Abdullah Giray Yaglikci, Hasan Hassan, Oguz Ergin, Onur Mutlu | Authors Info & Claims

ACM Transactions on Architecture and Code Optimization • Accepted on 06 February 2024 • https://doi.org/10.1145/3673653

Online AM: 14 June 2024 Publication History

Abstract

Modern computing systems access data in main memory at *coarse granularity* (e.g., at 512-bit cache block granularity). Coarse-grained access leads to wasted energy because the system does *not* use all individually accessed small portions (e.g., *words*, each of which typically is 64 bits) of a cache block. In modern DRAM-based computing systems, two key

Extended Version on Arxiv

https://arxiv.org/pdf/2207.13795.pdf

arXiv > cs > arXiv:2207.13795

Search... | All fields | Search

Help | Advanced Search

Computer Science > Hardware Architecture

[Submitted on 27 Jul 2022 (v1), last revised 9 Jun 2024 (this version, v4)]

Sectored DRAM: A Practical Energy-Efficient and High-Performance Fine-Grained DRAM Architecture

Ataberk Olgun, F. Nisa Bostanci, Geraldo F. Oliveira, Yahya Can Tugrul, Rahul Bera, A. Giray Yaglikci, Hasan Hassan, Oguz Ergin, Onur Mutlu

We propose Sectored DRAM, a new, low-overhead DRAM substrate that reduces wasted energy by enabling fine-grained DRAM data transfers and DRAM row activation. Sectored DRAM leverages two key ideas to enable fine-grained data transfers and row activation at low chip area cost. First, a cache block transfer between main memory and the memory controller happens in a fixed number of clock cycles where only a small portion of the cache block (a word) is transferred in each cycle. Sectored DRAM augments the memory controller and the DRAM chip to execute cache block transfers in a variable number of clock cycles based on the workload access pattern with minor modifications to the memory controller's and the DRAM chip's circuitry. Second, a large DRAM row, by design, is already partitioned into smaller independent physically isolated regions. Sectored DRAM provides the memory controller with the ability to activate each such region based on the workload access pattern via small modifications to the DRAM chip's array access circuitry. Activating smaller regions of a large row relaxes DRAM power delivery constraints and allows the memory controller to schedule DRAM accesses faster.
Compared to a system with coarse-grained DRAM, Sectored DRAM reduces the DRAM energy consumption of highly-memory-intensive workloads by up to (on average) 33% (20%) while improving their performance by up to (on average) 36% (17%). Sectored DRAM's DRAM energy savings, combined with its system performance improvement, allows system-wide energy savings of up to 23%. Sectored DRAM's DRAM chip area overhead is 1.7% the area of a modern DDR4 chip. We hope and believe that Sectored DRAM's ideas and results will help to enable more efficient and high-performance memory systems. To this end, we open source Sectored DRAM at this https URL.

Access Paper:

- View PDF
- HTML (experimental)
- TeX Source
- Other Formats

Current browse context:
cs.AR

< prev | next >

new | recent | 2022-07

Change to browse by:
cs

References & Citations

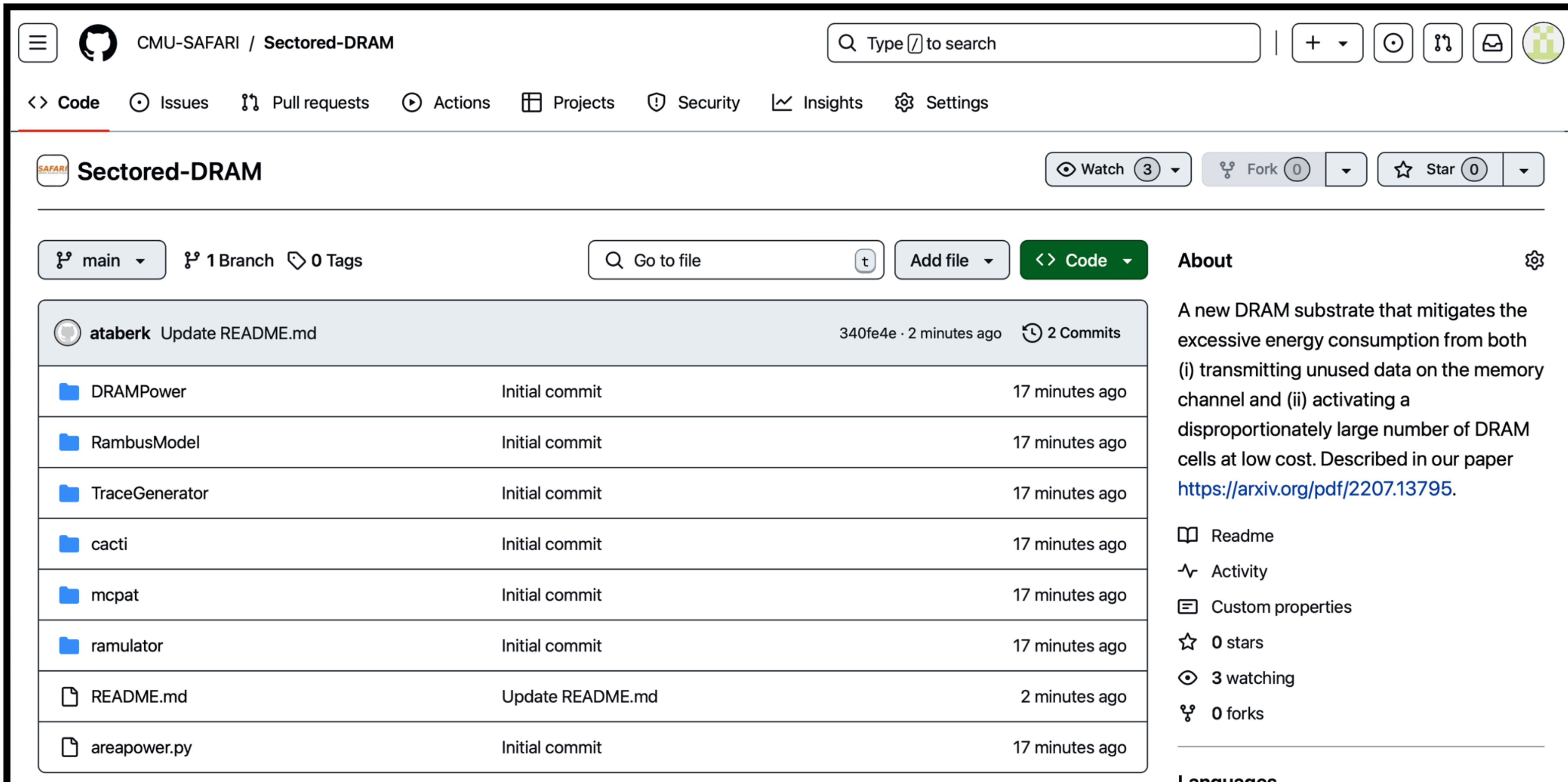
- NASA ADS
- Google Scholar
- Semantic Scholar

Export BibTeX Citation

Bookmark

Sectored DRAM is Open Source

https://github.com/CMU-SAFARI/Sectored-DRAM



Sectored DRAM

A Practical Energy-Efficient and High-Performance Fine-Grained DRAM Architecture

Paper

GitHub

Ataberk Olgun
olgunataberk@gmail.com

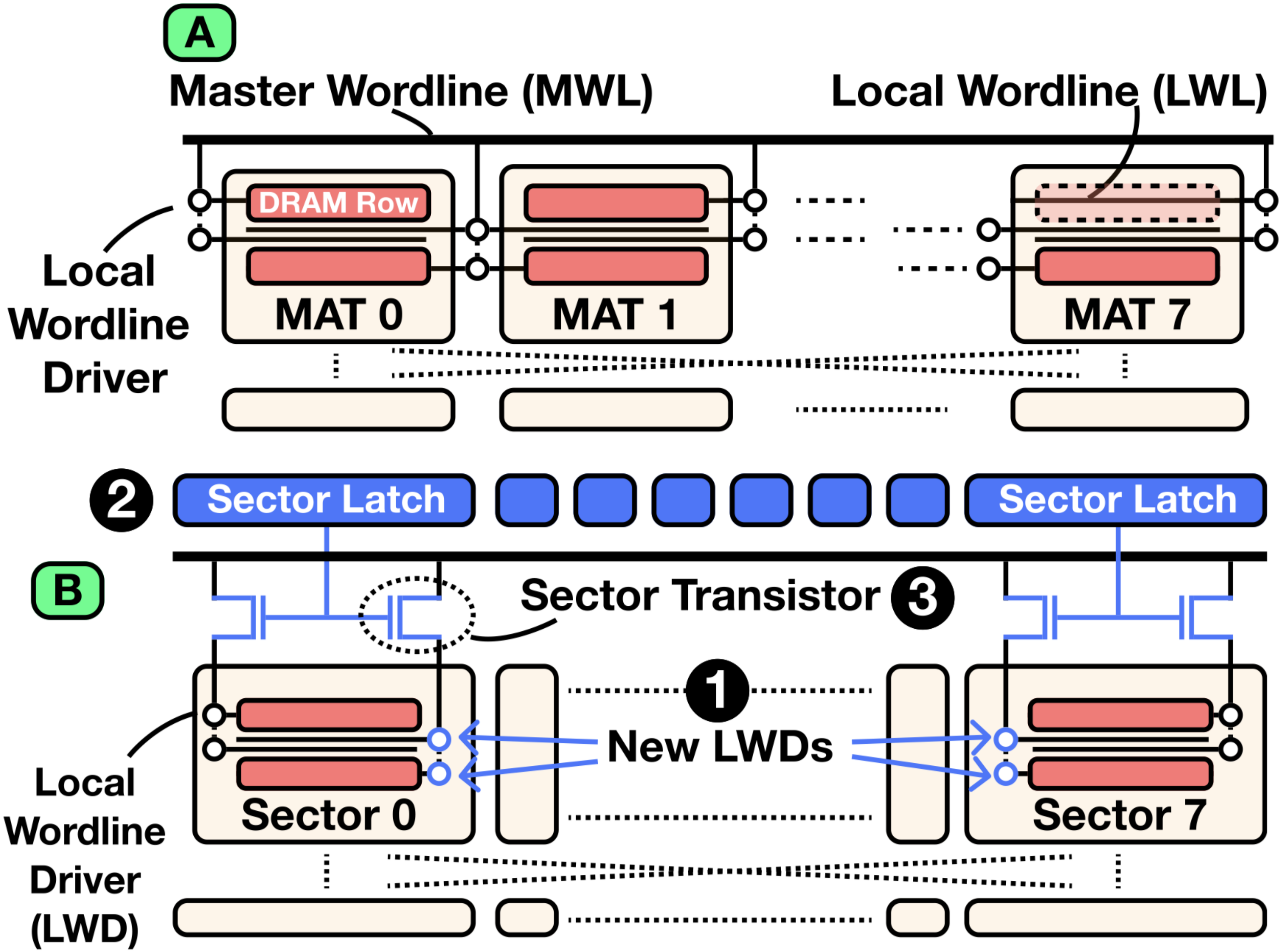
F. Nisa Bostanci Geraldo F. Oliveira Yahya Can Tugrul Rahul Bera

A. Giray Yaglikci Hasan Hassan Oguz Ergin Onur Mutlu

ETH zürich SAFARI kasırga TOBB ETÜ Ekonomi ve Teknoloji Üniversitesi

Backup Slides

Sectored DRAM Subarray Organization



Exposing Sectored DRAM to the Memory Controller with No Interface Modifications

1 Sectored Activation (SA)

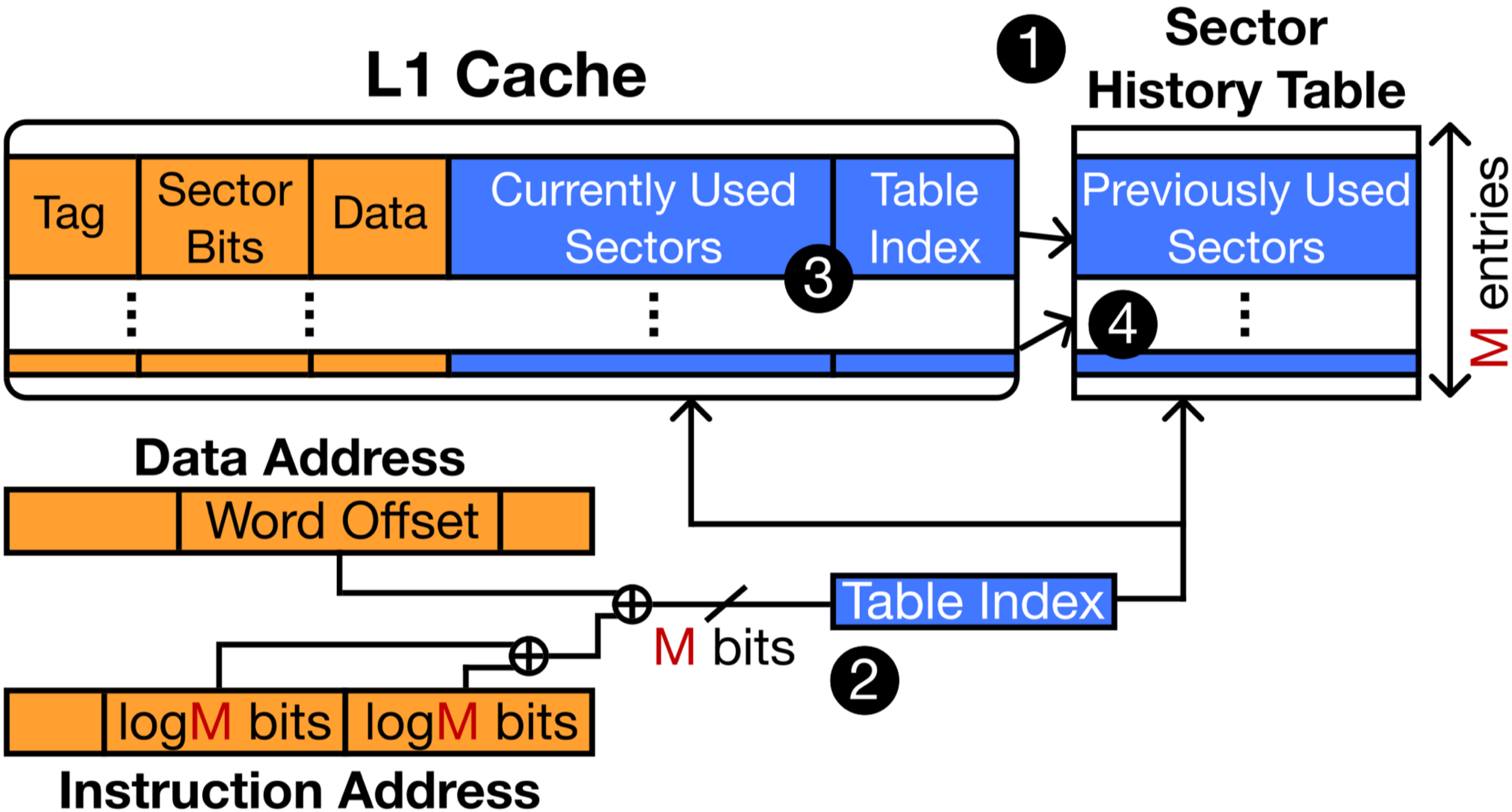
- More than 10 unused bits in precharge (PRE) command encoding
- Determine the sectors opened for the next activate (ACT) command

Activating fewer than all 8 sectors relaxes power constraints allows for higher ACT command throughput

2 Variable Burst Length (VBL)

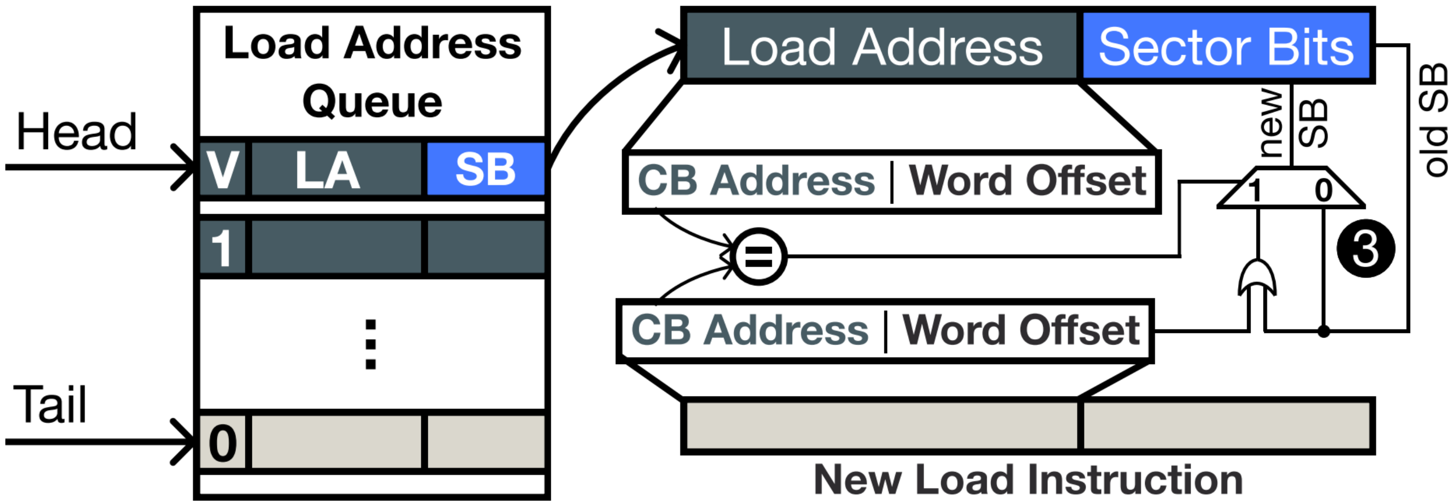
- DRAM and memory controller must agree on burst length
- DRAM and memory controller store sector bits for each bank
- Low overhead popcount circuitry to count set (logic-1) sector bits

Sector Predictor



Load/Store Queue (LSQ) Lookahead

- One load/store instruction *references one word* in main memory
- **Key Mechanism:** 1) Collect references from *younger* load/store instructions
  2) store the collected references in the *oldest* load/store instr.

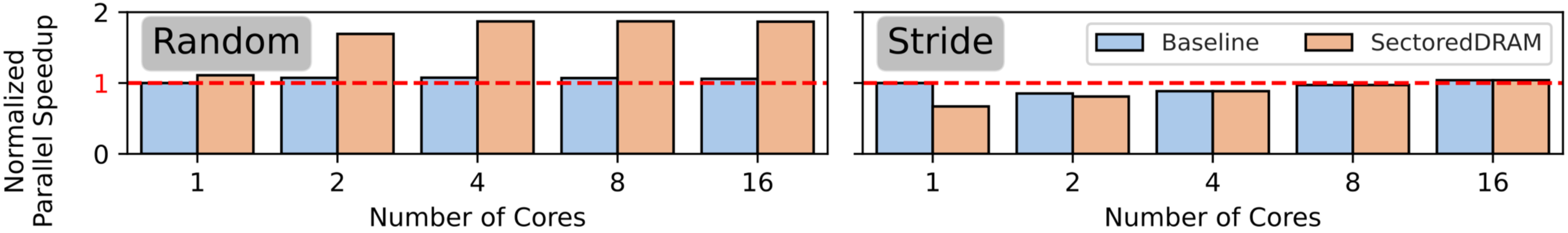


A load/store instruction retrieves all words in a cache block that will be referenced in the near future to the L1 cache with only one cache access

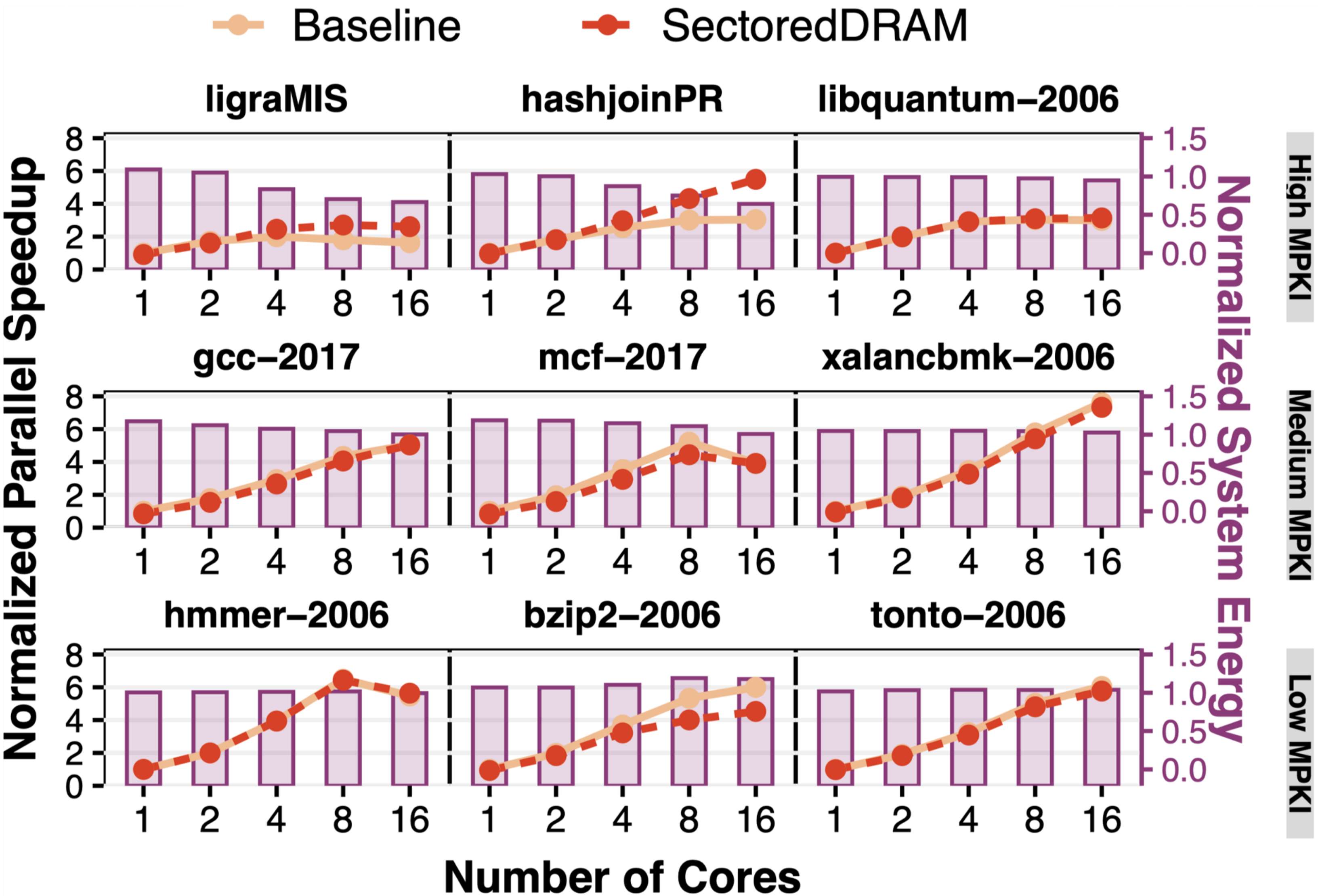
Evaluated Workloads

LLC MPKI	Workloads
≥ 10 (High)	ligraPageRank, mcf-2006, libquantum-2006, gobmk-2006, ligraMIS, GemsFDTD-2006, bwaves-2006, lbm-2006, lbm -2017, hashjoinPR
1..10 (Medium)	omnetpp-2006, gcc-2017, mcf-2017, cactusADM-2006, zeusmp-2006, xalancbmk-2006, ligraKCore, astar-2006, cactus-2017, parest-2017, ligraComponents
≤ 1 (Low)	splash2Ocean, tonto-2006, xz-2017, wrf-2006, bzip2-2006, xalancbmk-2017, h264ref-2006, hmmer-2006, namd-2017, blender-2017, sjeng-2006, perlbench-2006, x264-2017, deepsjeng-2017, gromacs-2006, gcc-2006, imagick-2017, leela-2017, povray-2006, calculix-2006

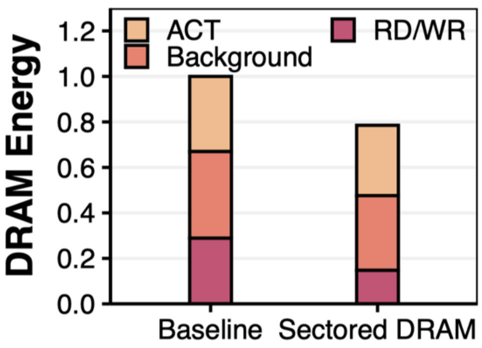
Microbenchmark Performance

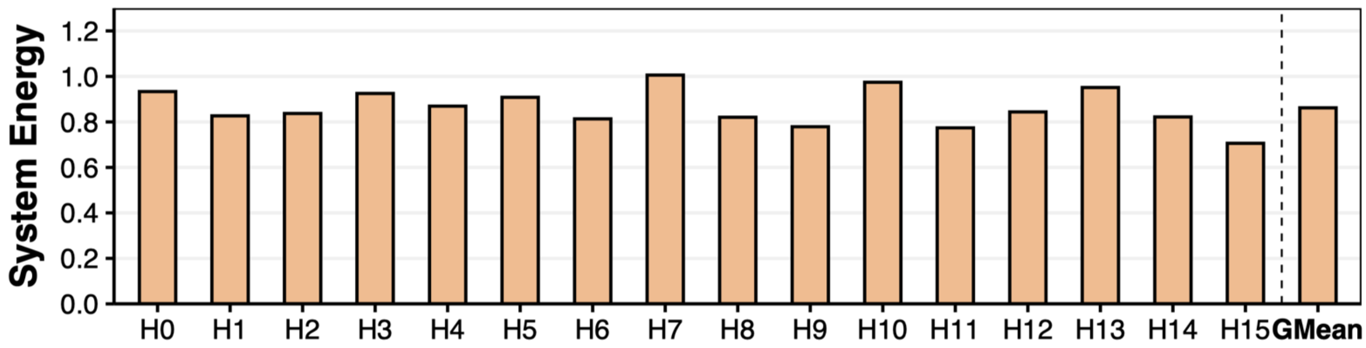


Parallel Speedup and System Energy per Workload

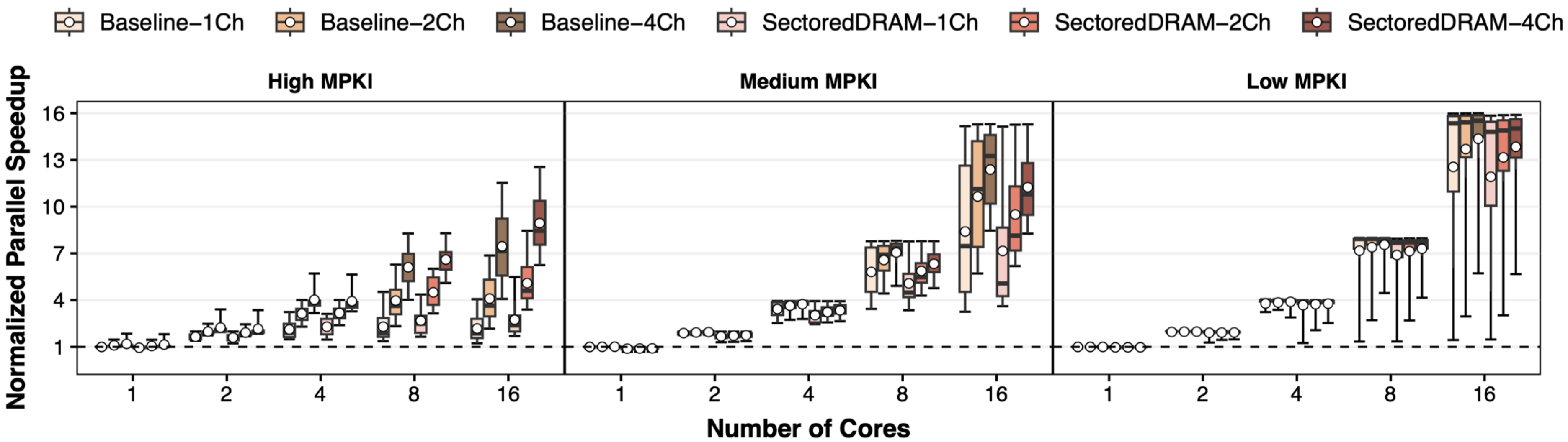


DRAM Energy Breakdown and System Energy

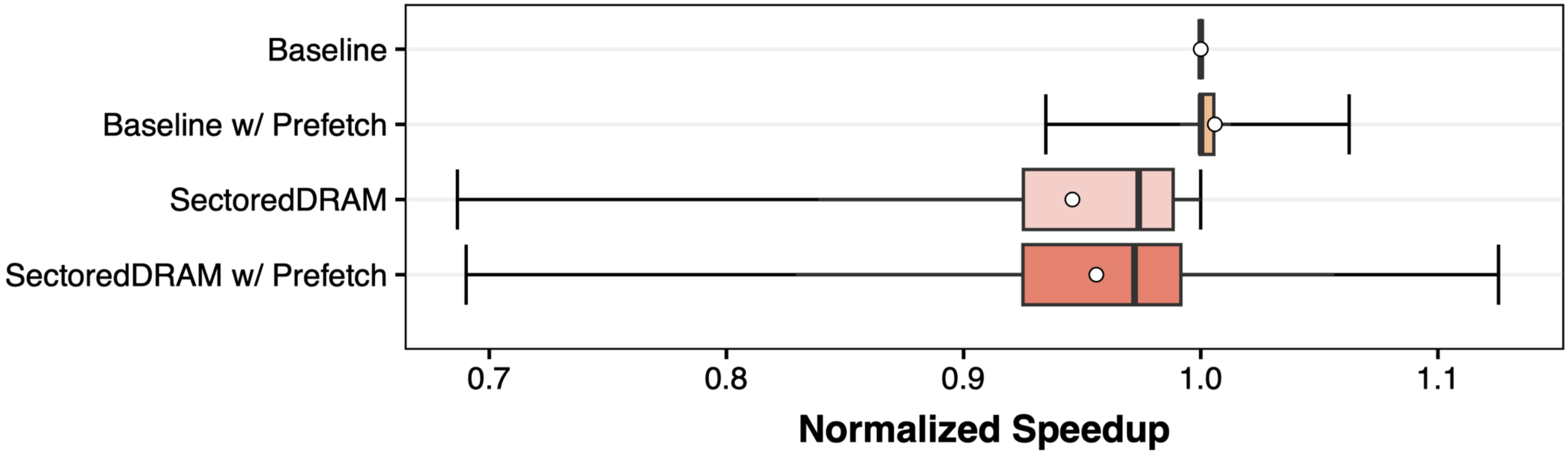




Performance Sensitivity to Number of Channels



Sectored DRAM with Prefetching



Enabling Higher Row Activation Rate

- tFAW = 25 nanoseconds (ns)
- 32 sectors can be activated in a tFAW
- Only 10 activate commands can be issued in 25 ns due to tRRD_L and tRRD_S

- 10 ACT, each of which activate one sector
takes 20% less power than
4 ACT, each of which activates 8 sectors

Sectored DRAM vs Module-Level Mechanisms

- DRAM interface modifications vs. DRAM chip modifications

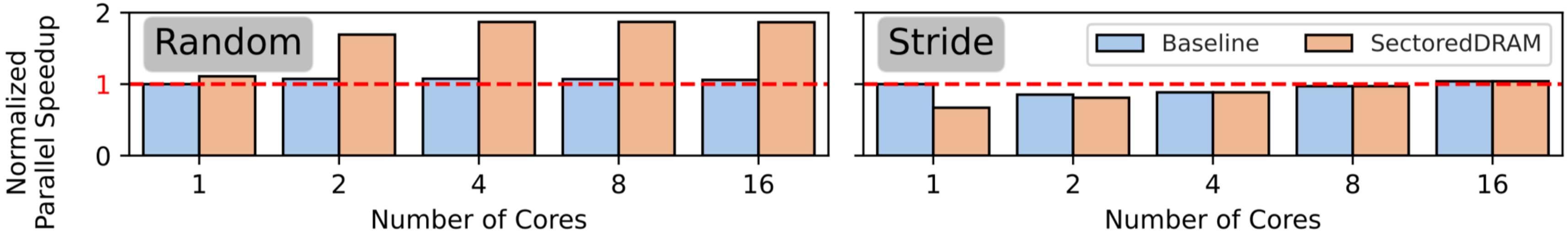
- Low overhead module-level mechanism
  induces 23% overhead
  where Sectored DRAM provides 17% speedup
  - Command bus becomes the bottleneck
  - Alleviating command bus bottleneck is area expensive

- System integration heavily inspired by DGMS

Discussion (I)

- Mitigate Sectored DRAM's performance overheads
  - Better sector prediction/prefetching
  - Sector annotation (?): Software-guided sector ``prefetching''
  - Enable subarray-level parallelism
    - Scatter-gather DRAM (inside a chip)

- Better explore Sectored DRAM's use cases



- Memory compression and Sectored DRAM
  - Compress cache blocks in main memory
  - Transfer compressed cache block using Sectored DRAM
  - Benefit: NO performance overhead because no sector misses

Discussion (II)

- Even finer-grained DRAM
  - Activate only as many cells as you read
    - Terribly area-expensive
      - Either 1) Need 64 data lines (word size) coming out of every mat
      - 2) Need a way to “mask” activation of many cells in a row
      - Need very small (64 or fewer cells wide) DRAM mats
    - How to leverage row buffer locality
      if we activate only 64 cells (a word)?
      - Activate more than 64 cells