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

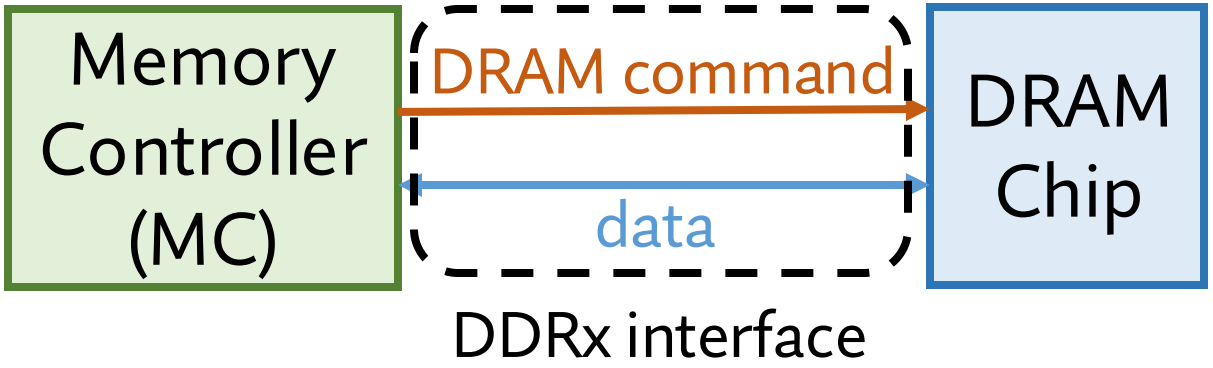
SMD Outline

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

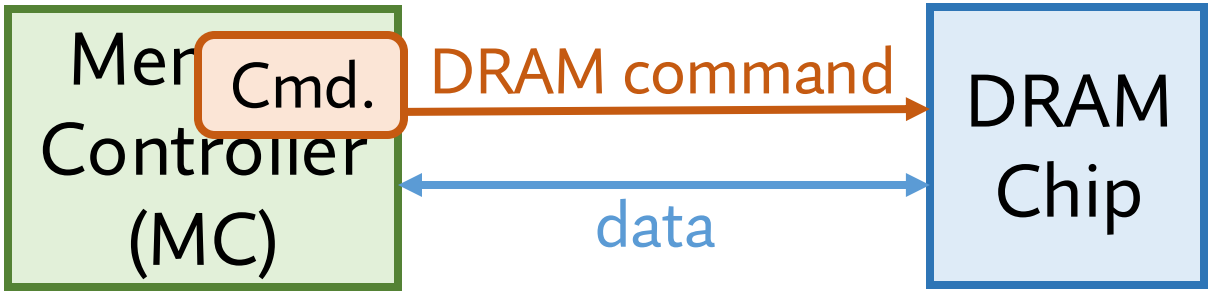
SMD Outline

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

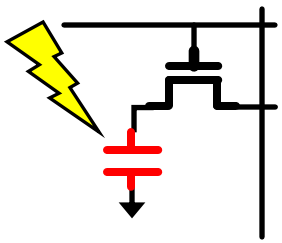
executes
all DRAM commands

- by issuing *DRAM commands*

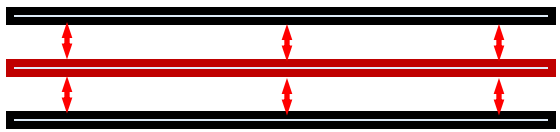
DRAM interface is completely controlled by one side

DRAM Maintenance Mechanisms

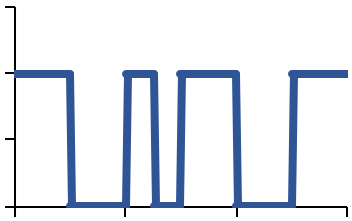
Data Retention



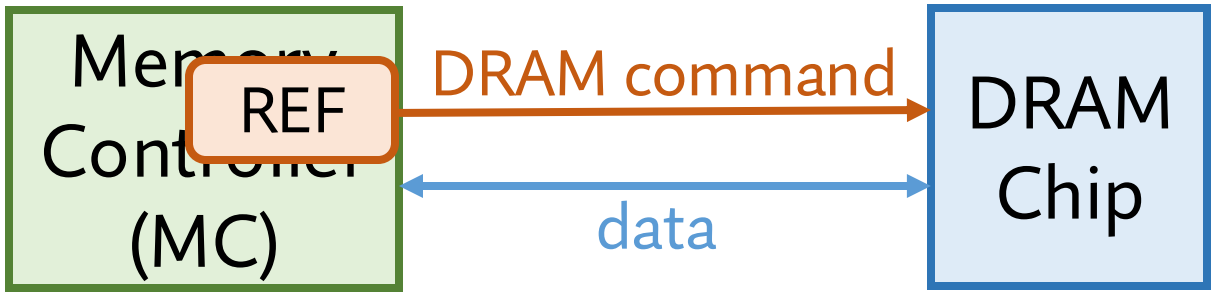
Read Disturbance (e.g., RowHammer)



Variable Retention Time



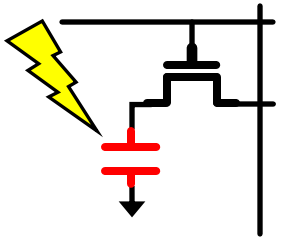
- DRAM failure modes necessitate maintenance mechanisms
- Perform operations to maintain DRAM data integrity
  - A prominent example is periodic refresh



New Maintenance Mechanisms are Needed

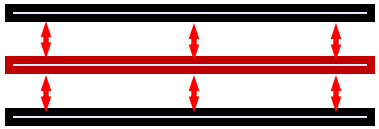
- Density scaling increases memory error rates

Data Retention



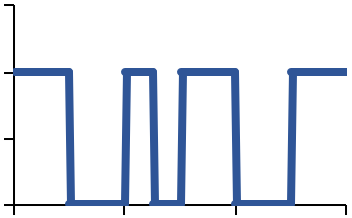
shrinking capacitance worsening leakage

Read Disturbance (e.g., RowHammer)



increasing interference

Variable Retention Time

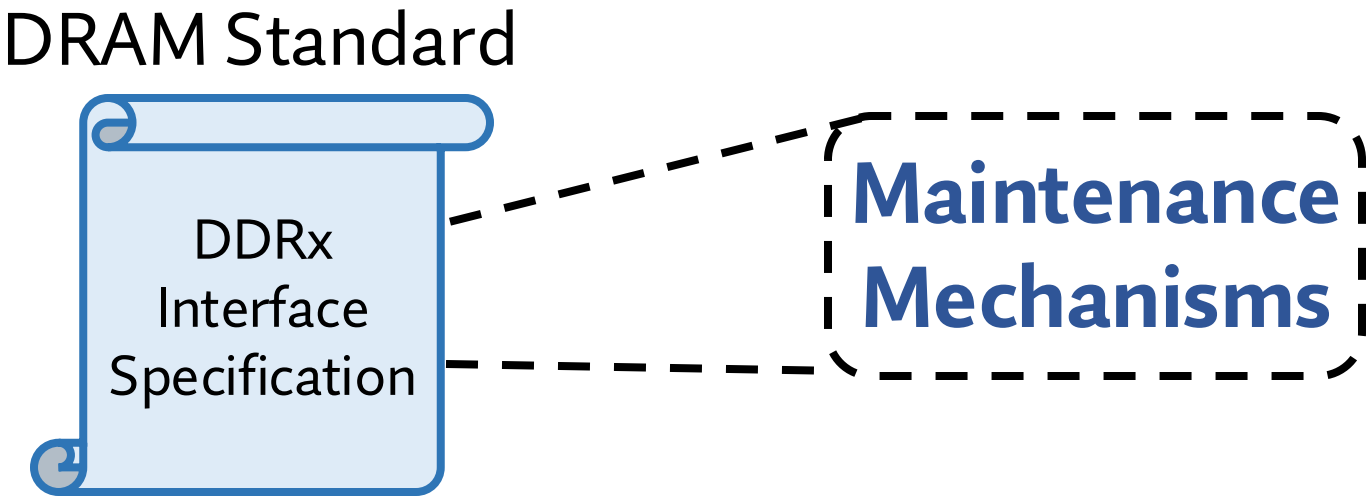


shrinking capacitance worsening leakage

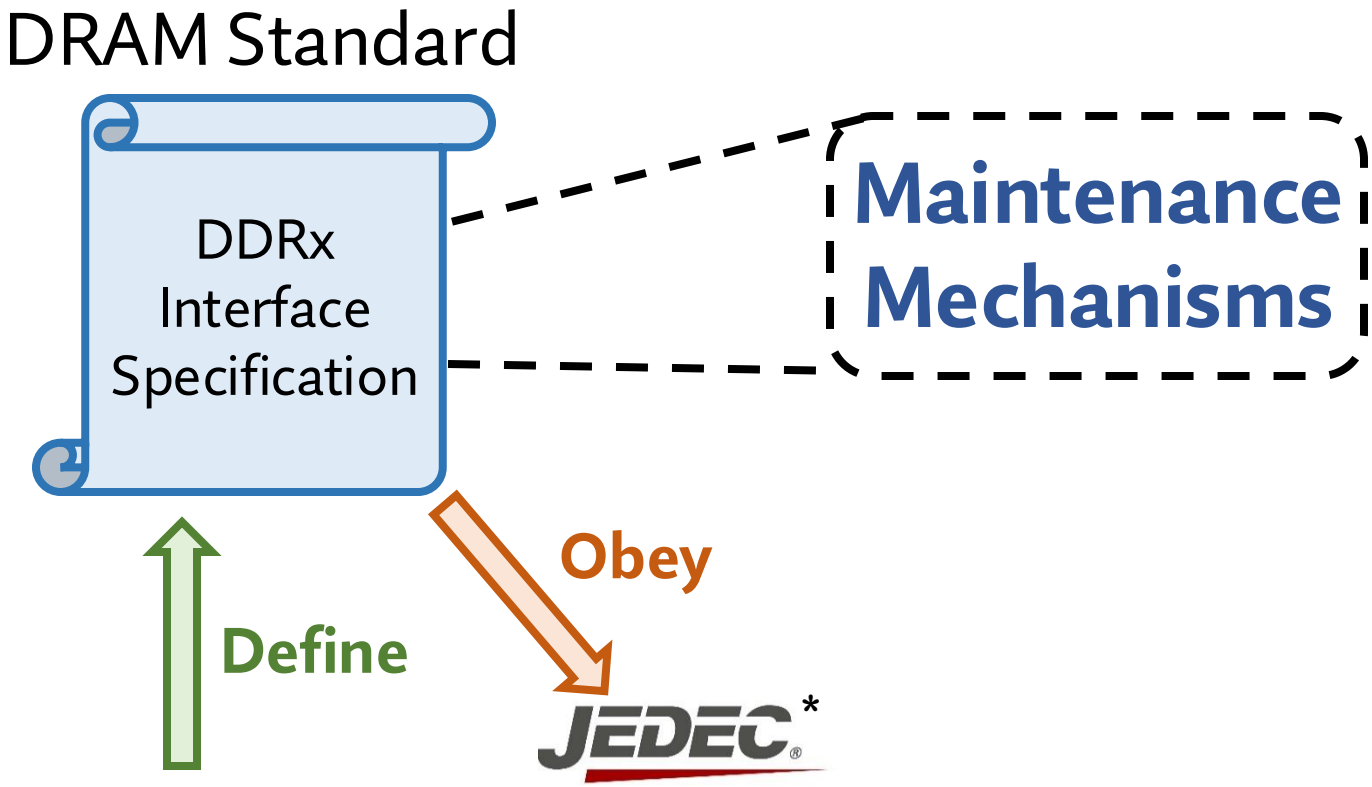
Continued DRAM process scaling necessitates new efficient maintenance mechanisms

[Patel+, MICRO'21]

DRAM Standard Interface Specification



DRAM Standard Body – JEDEC*



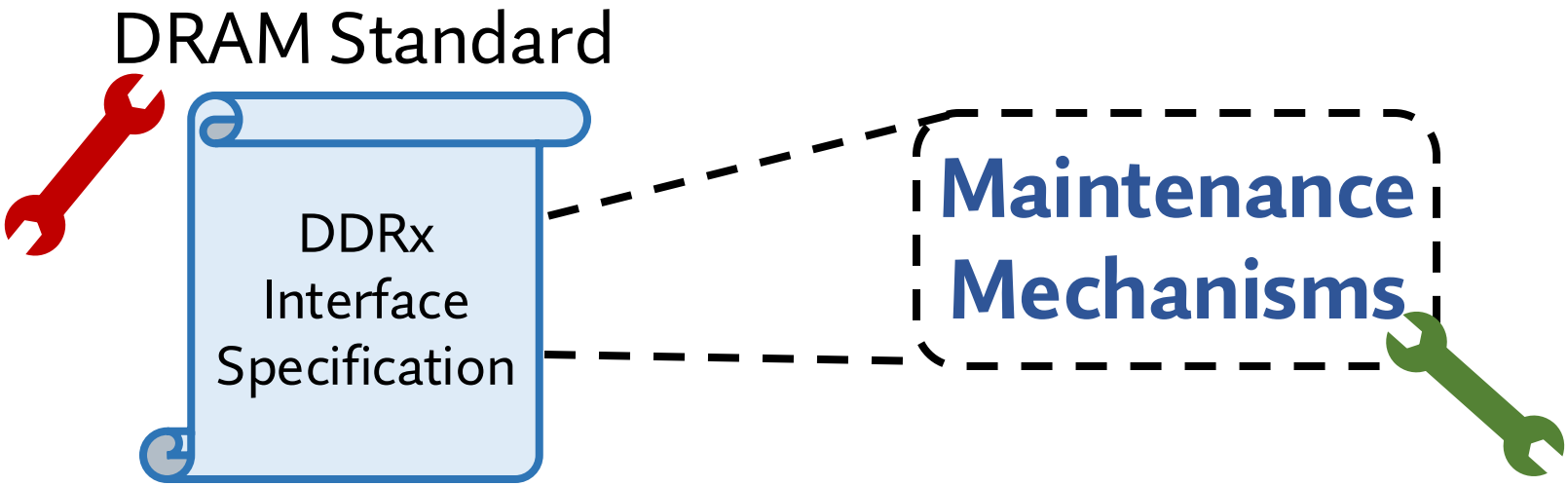
390+ companies

DRAM manufacturers parts manufacturers . . .

system manufacturers foundries

*Joint Electron Device Engineering Council

Barrier to New Maintenance Mechanisms

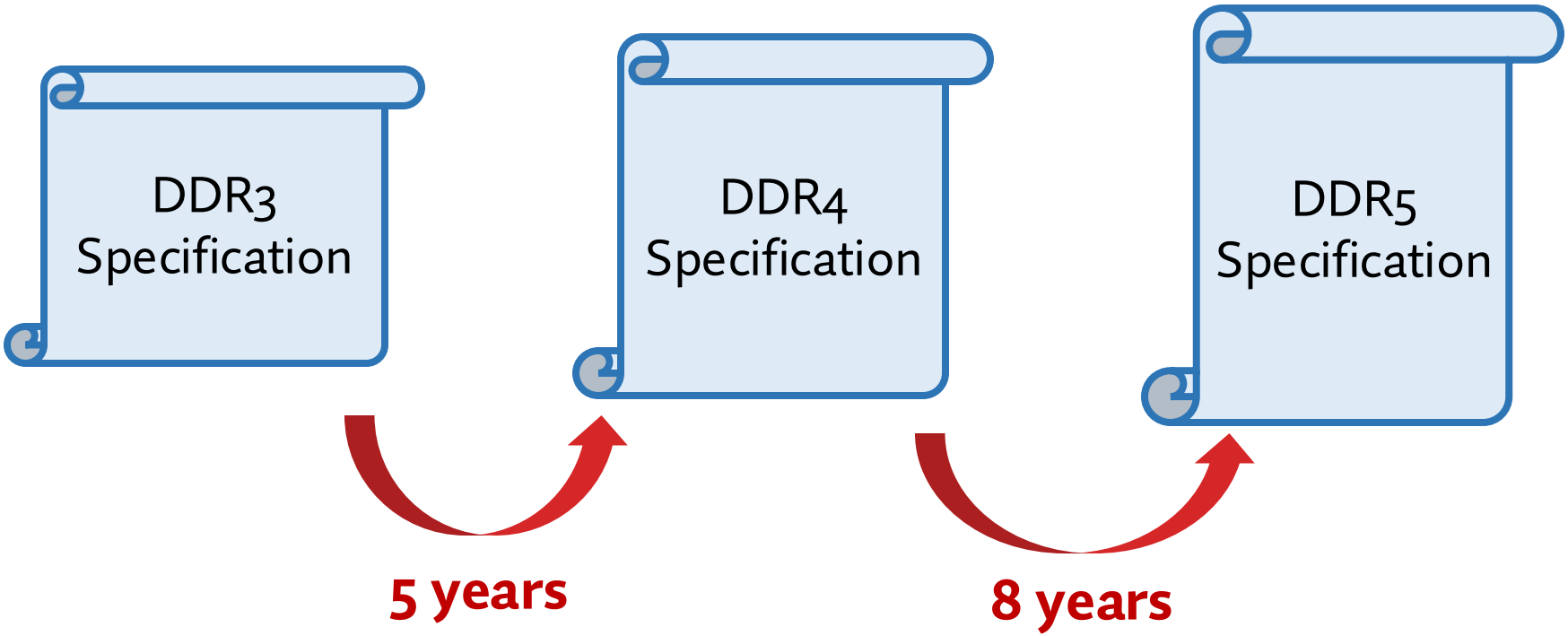


- Adding new or modifying existing maintenance mechanisms requires lengthy modifications to

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

DRAM interface is rigid

DRAM Specifications Evolve Slowly



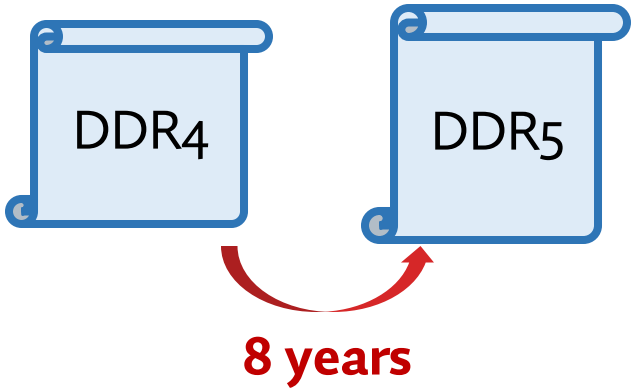
Multi-year effort by the JEDEC committee

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

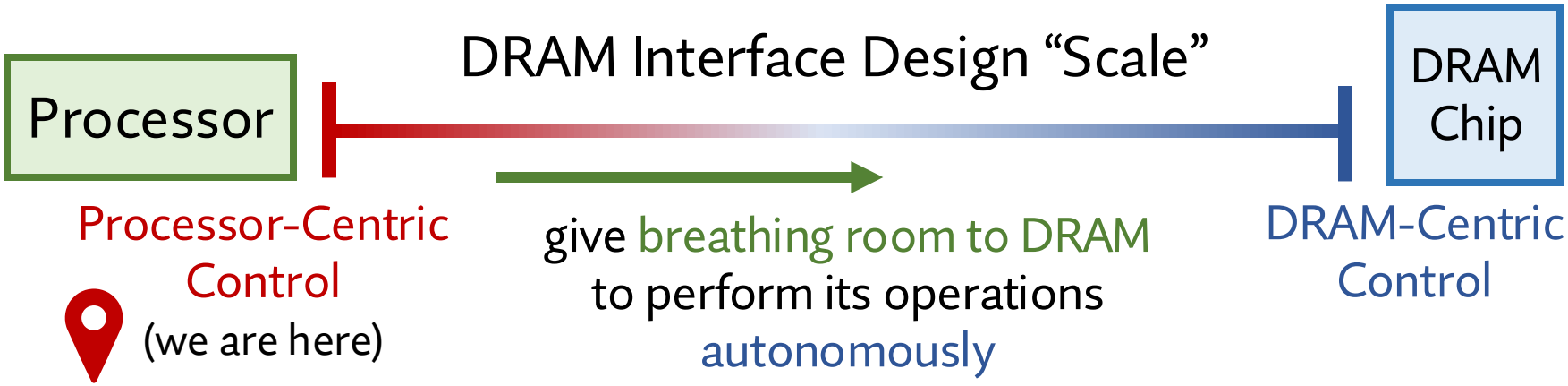
1. Maintenance mechanisms can be implemented more easily and rapidly
   - DRAM interface modifications are not required

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



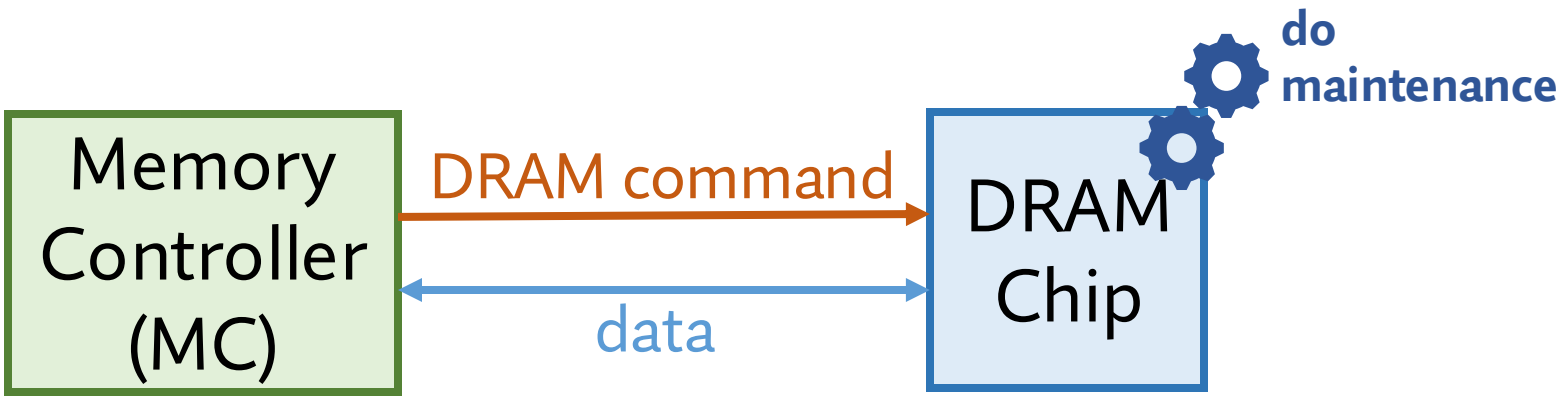
- **Goal:** Make a simple, one-time change to the DRAM interface that enables autonomous maintenance operations

SMD Outline

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

SMD Key Idea: Autonomous Maintenance

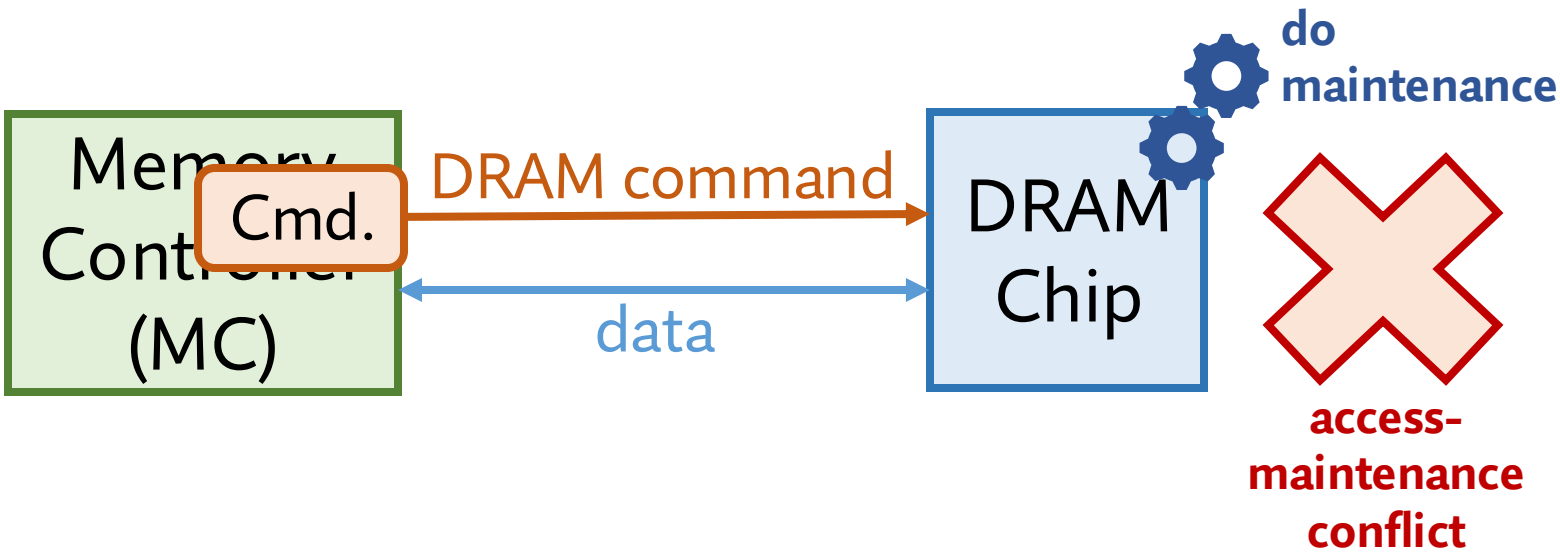
DRAM chip controls in-DRAM maintenance operations



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

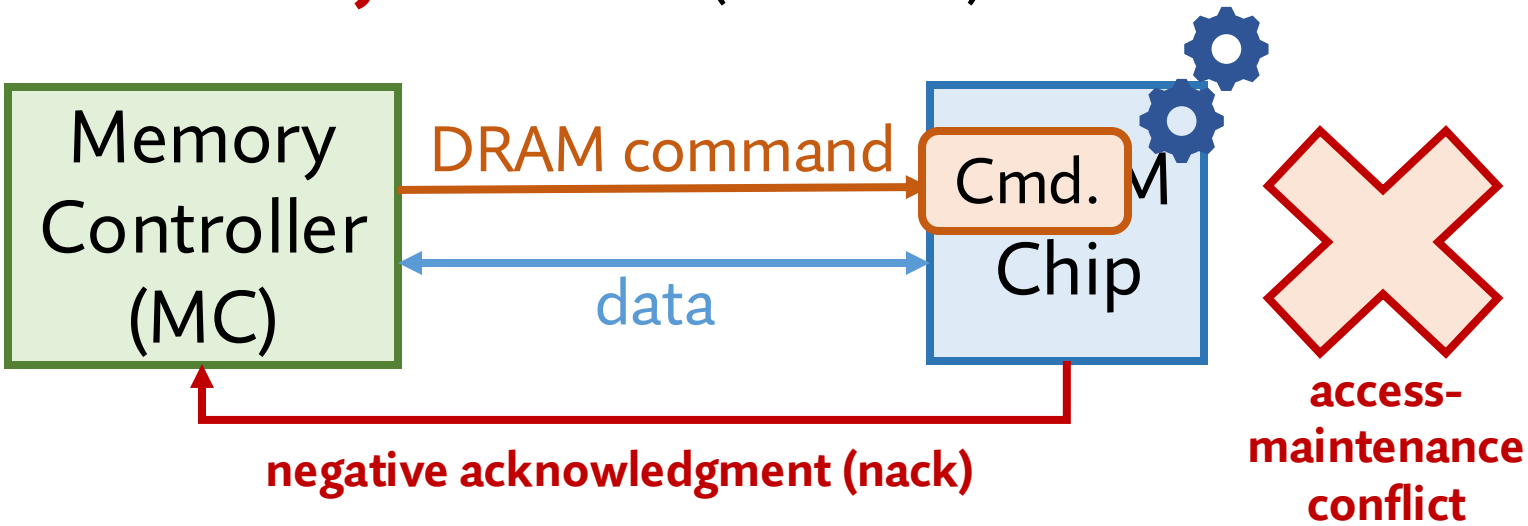
Access-Maintenance Conflicts

- **Problem:** Access-maintenance conflict



SMD Key Mechanism

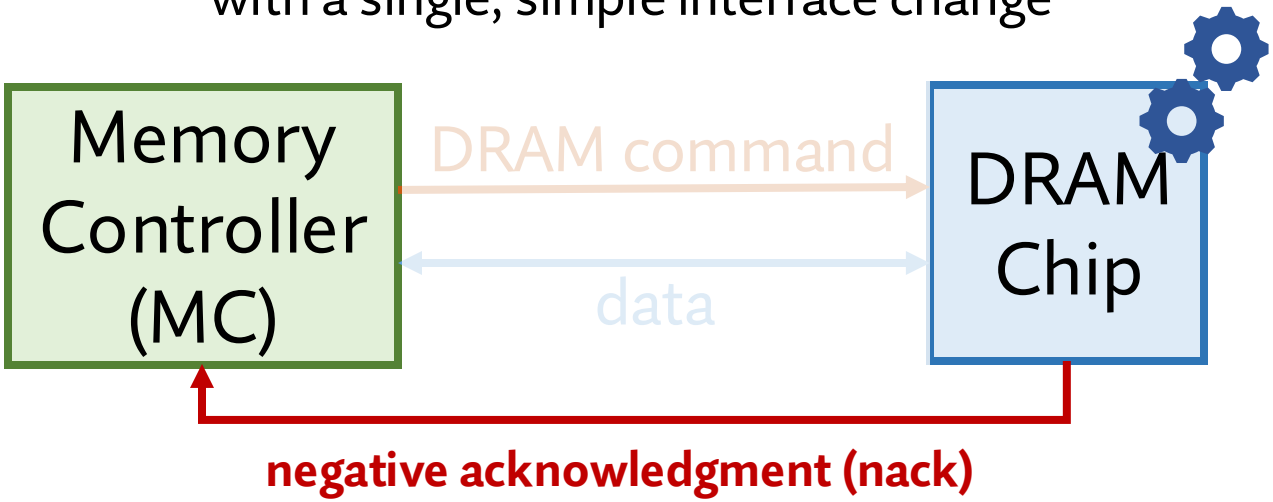
- **Problem:** Access-maintenance conflict
- **Key mechanism: Reject** access (activate) commands



SMD Key Contribution

DRAM chip controls in-DRAM maintenance operations

with a single, simple interface change



orchestrates all access operations

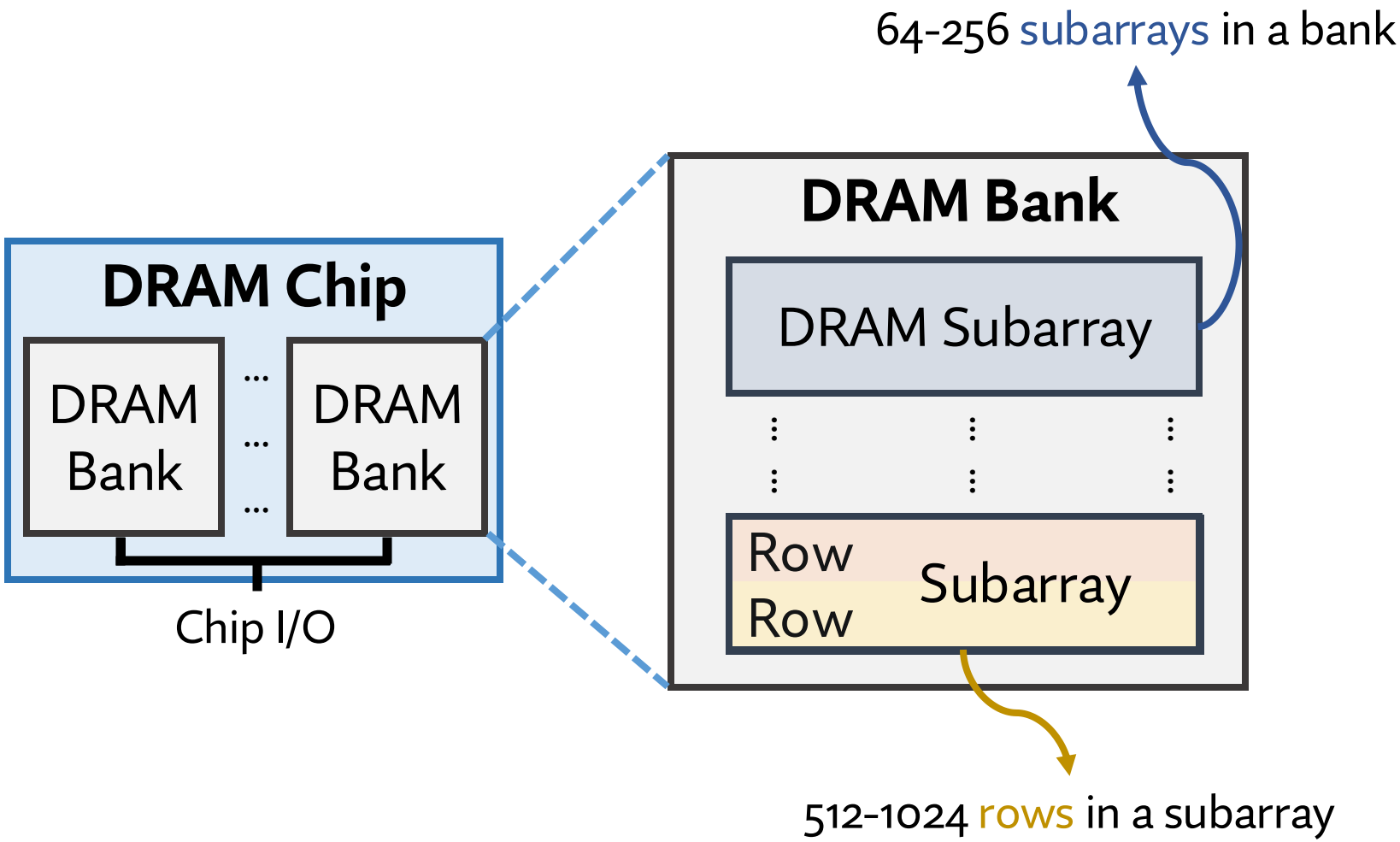
can now perform its own maintenance autonomously

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

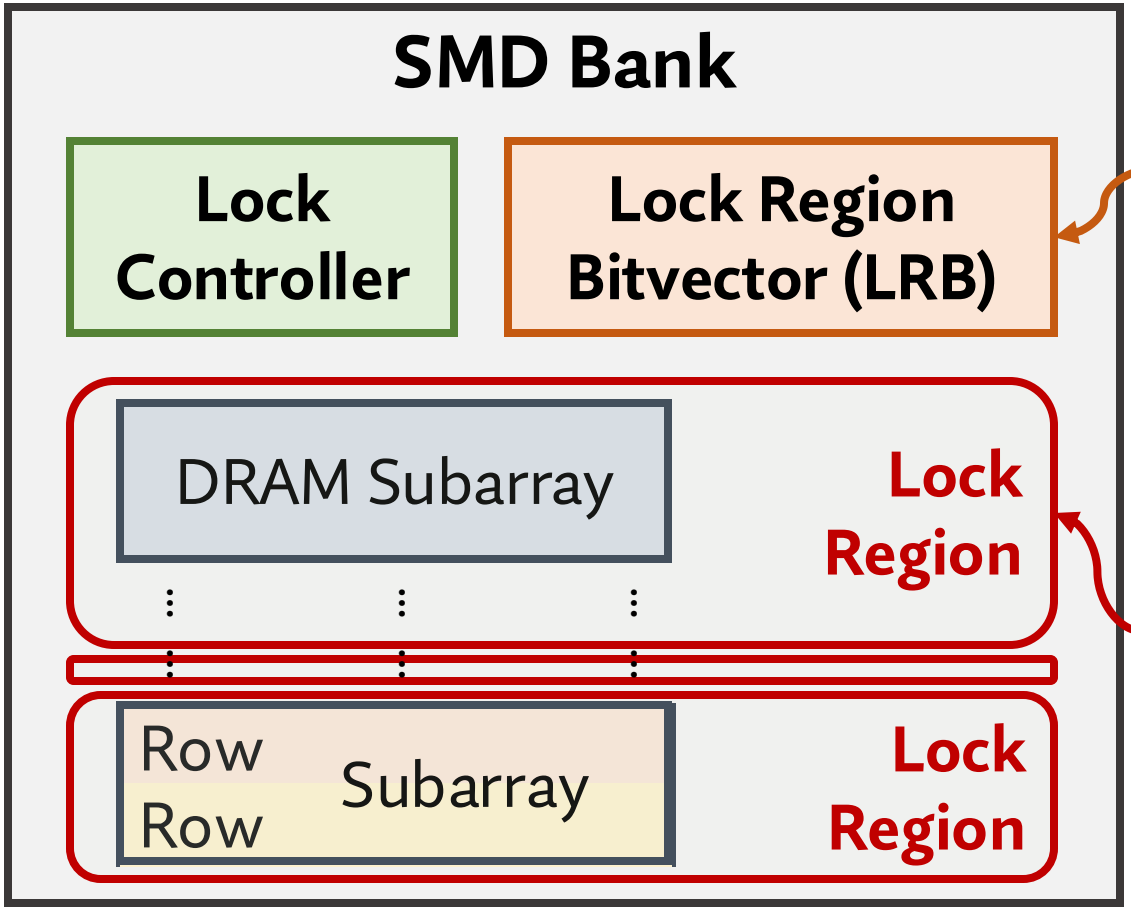
Deeper Look at SMD

1 SMD Bank Organization

DRAM Chip Organization



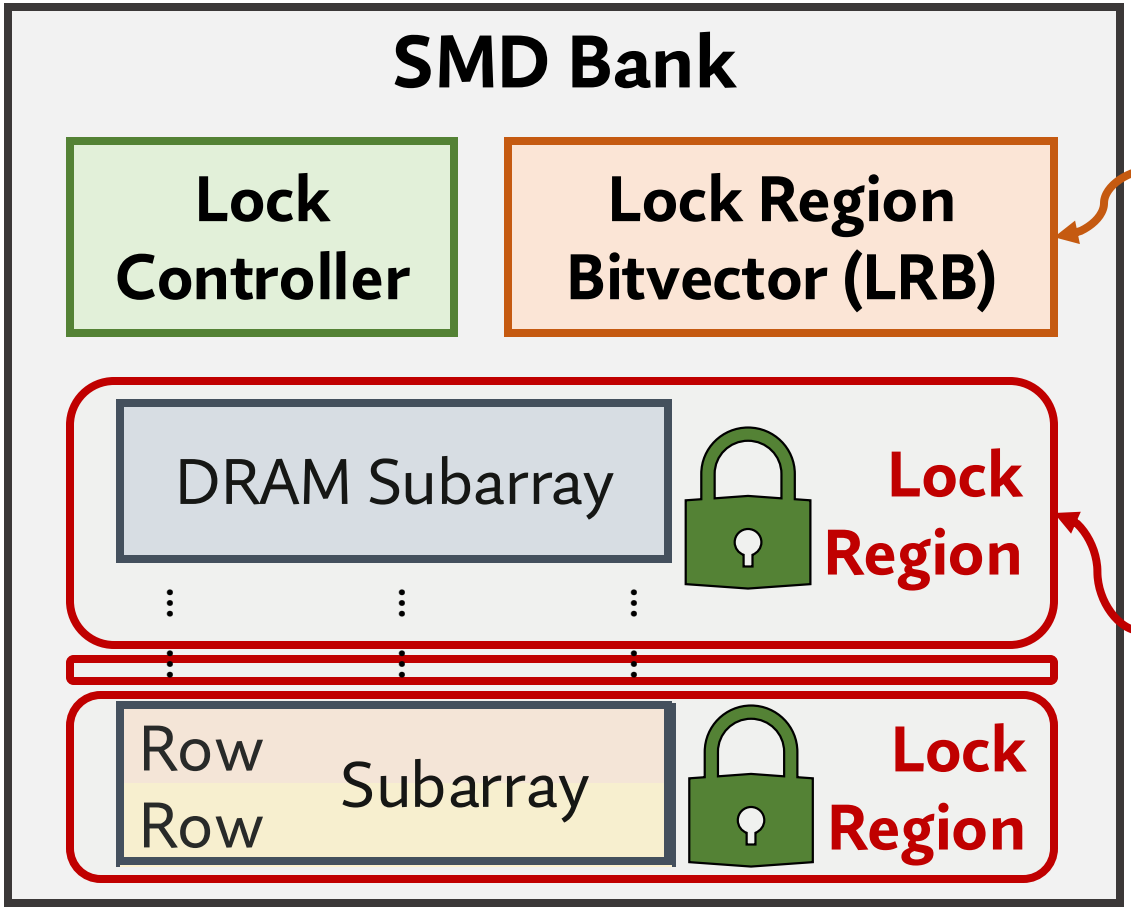
DRAM Bank with SMD



indicate regions
undergoing maintenance
(1 bit per lock region)

granularity of
a maintenance operation
(multiple subarrays)

Locking Regions for Maintenance



indicate regions undergoing maintenance
(1 bit per lock region)

granularity of a maintenance operation
(multiple subarrays)

Lock a region before starting maintenance

Deeper Look at SMD

1 SMD Bank Organization

2 Region Locking Mechanism

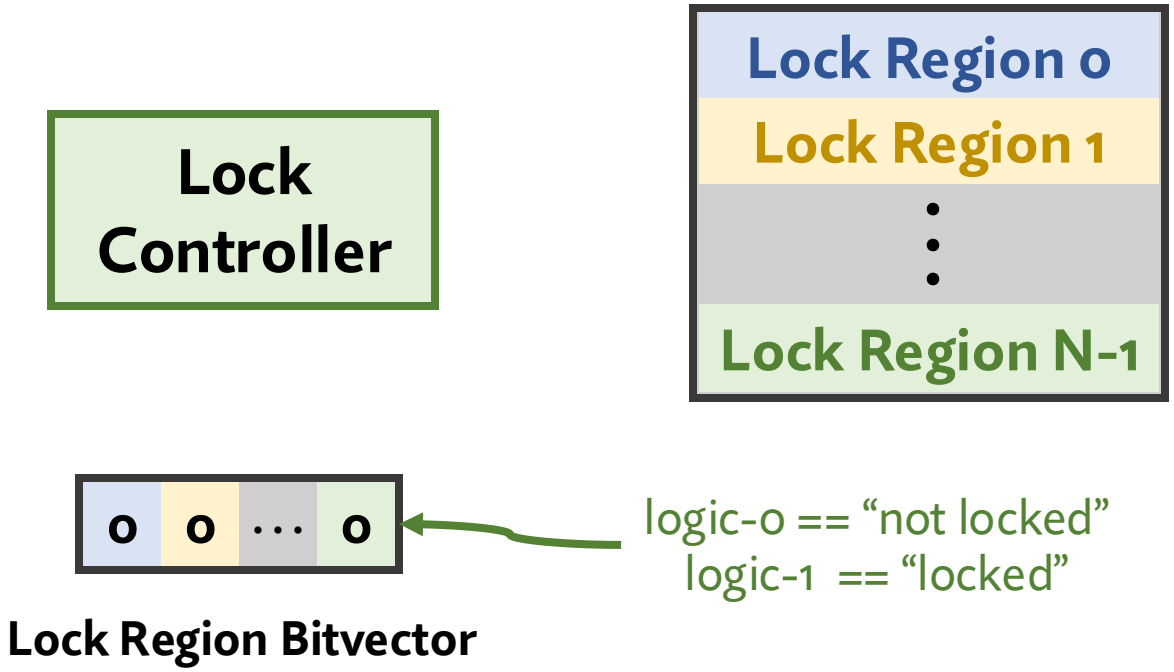
Summary of Region Locking Mechanism

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

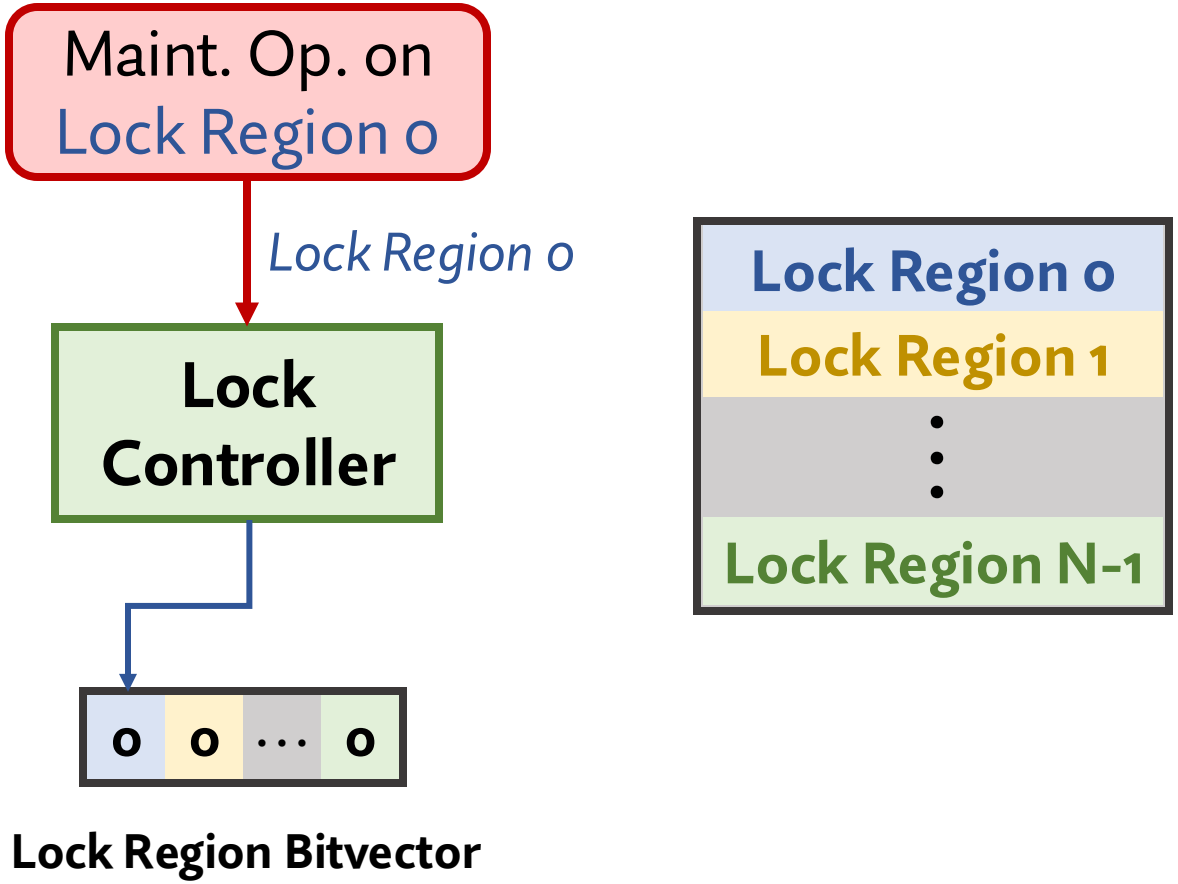
Summary of Region Locking Mechanism

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

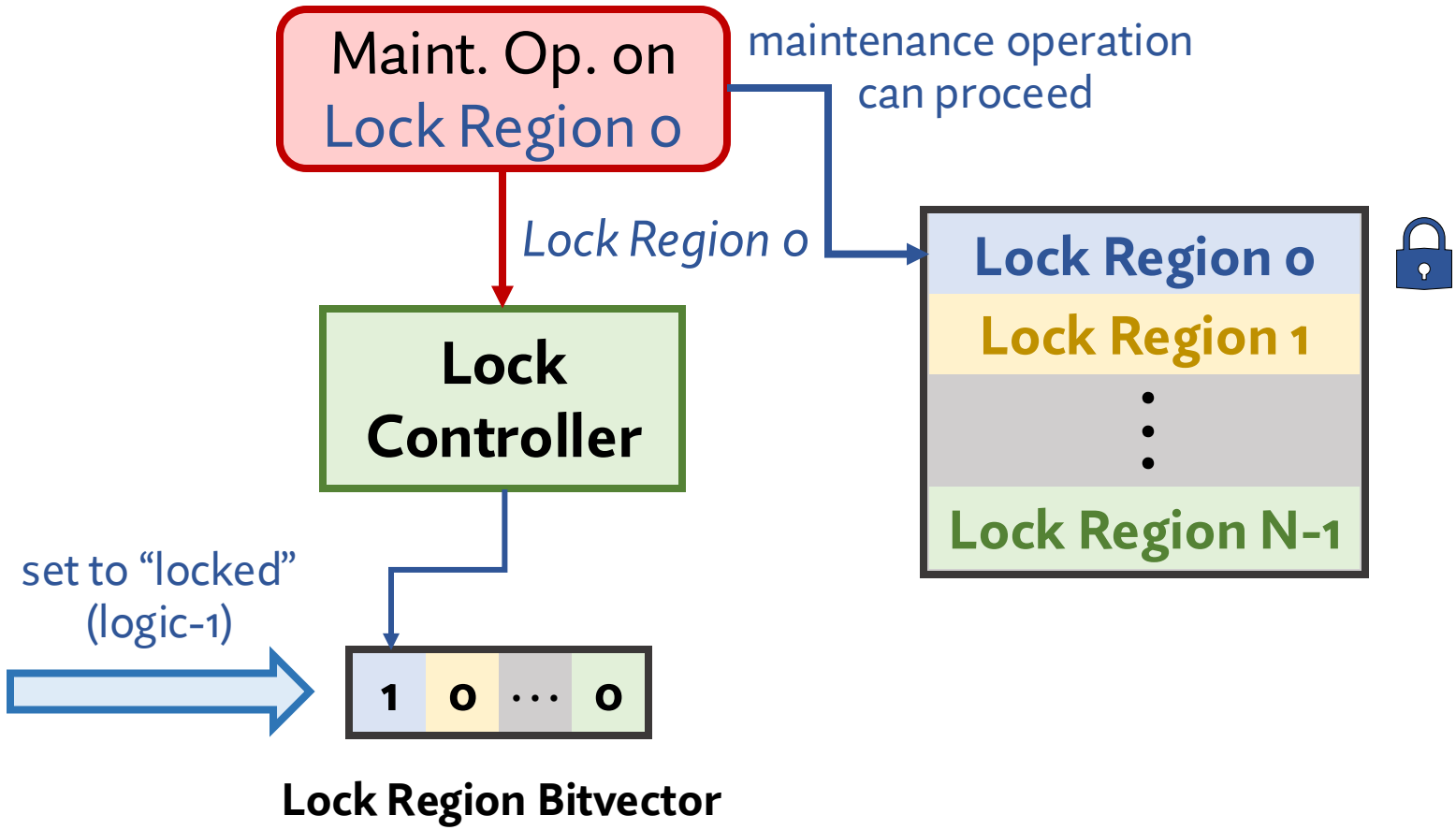
Locking a Region



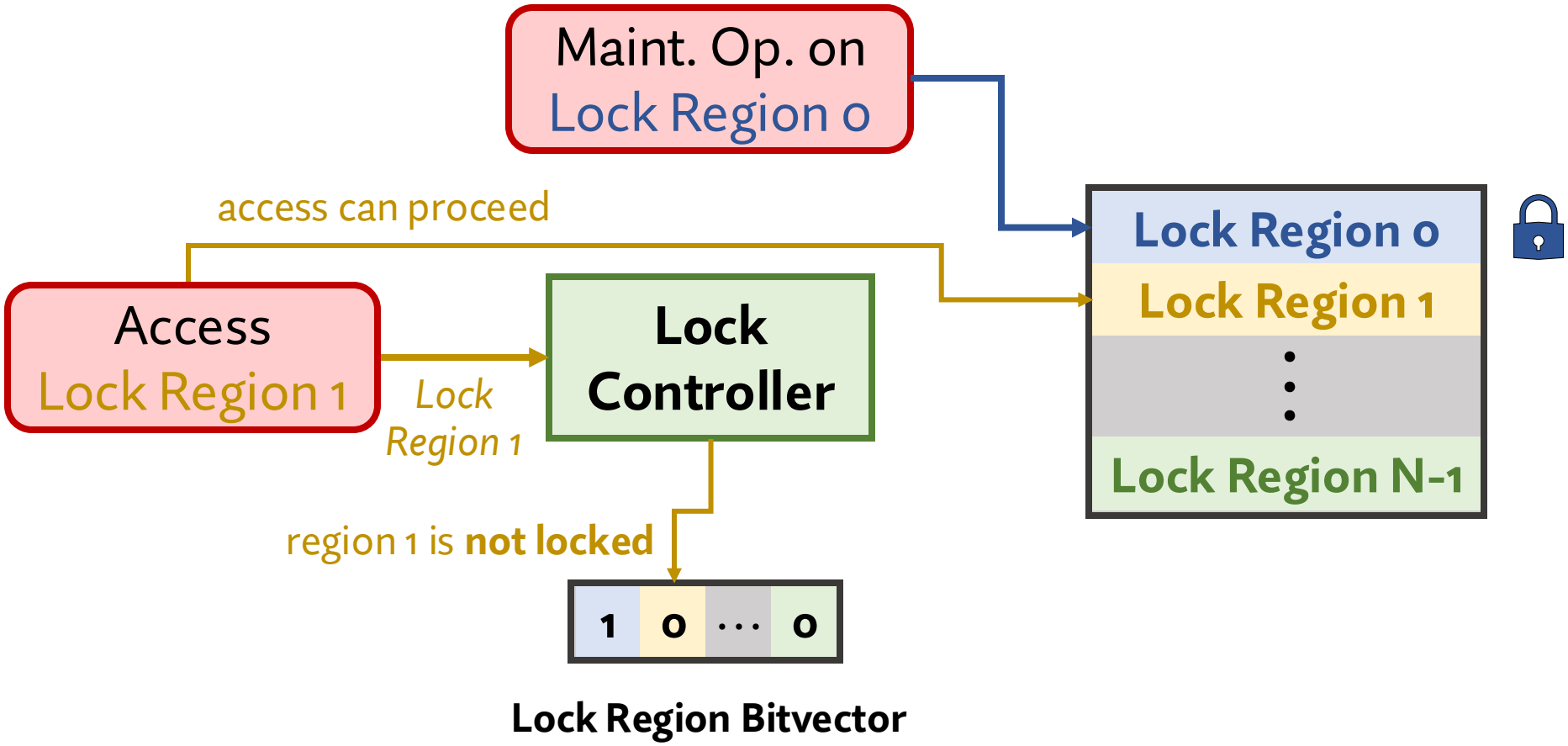
Locking a Region



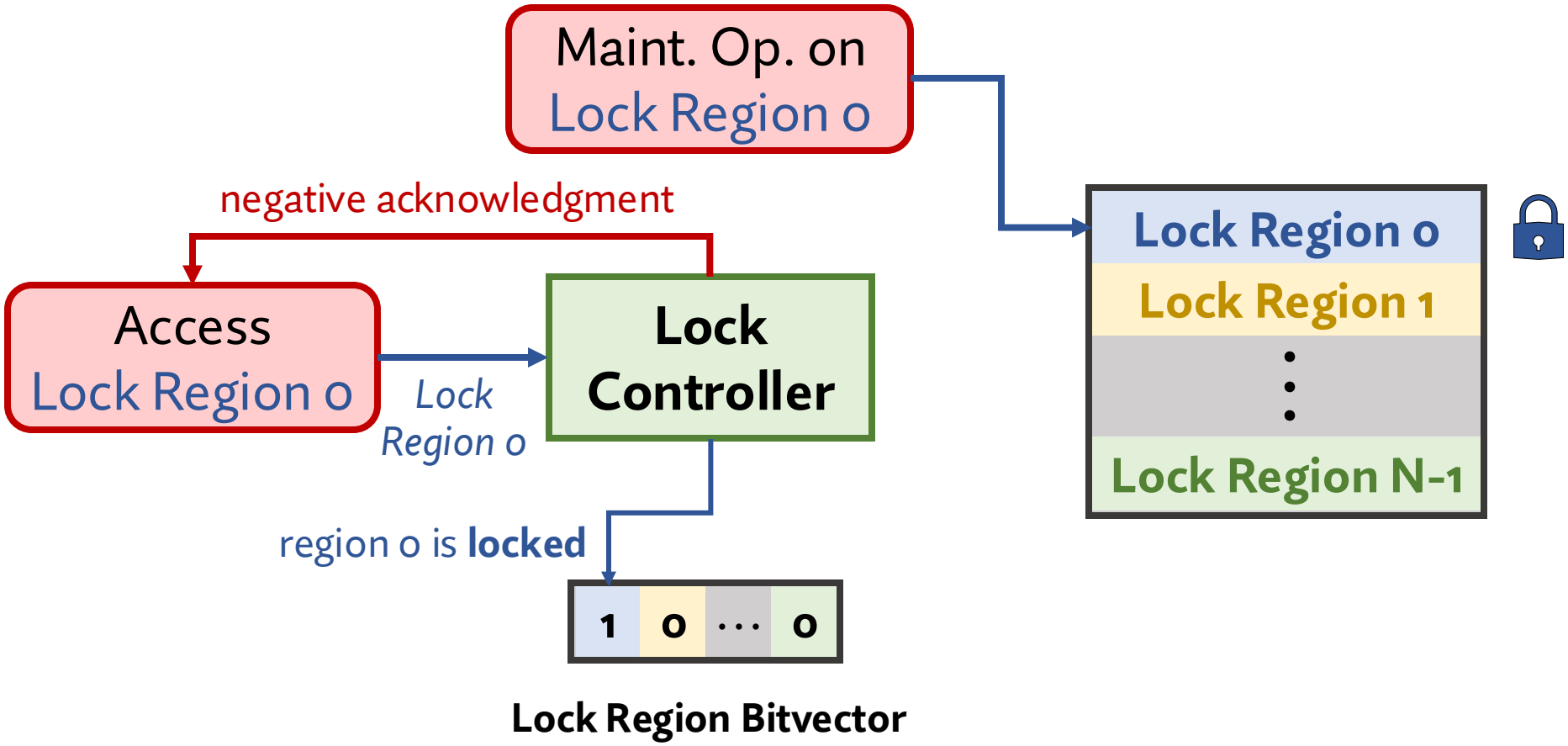
Locking a Region



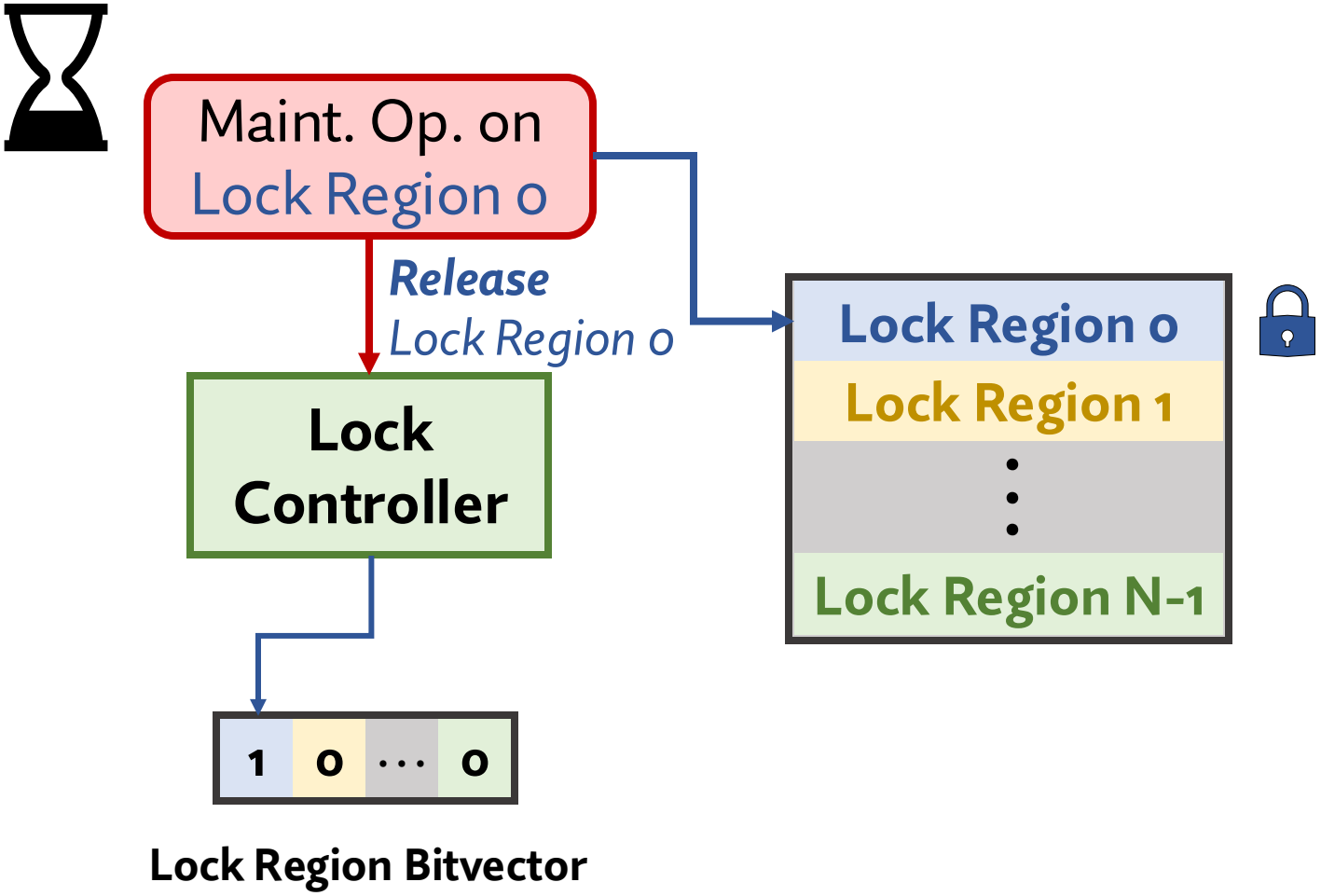
Accessing a Not Locked Region



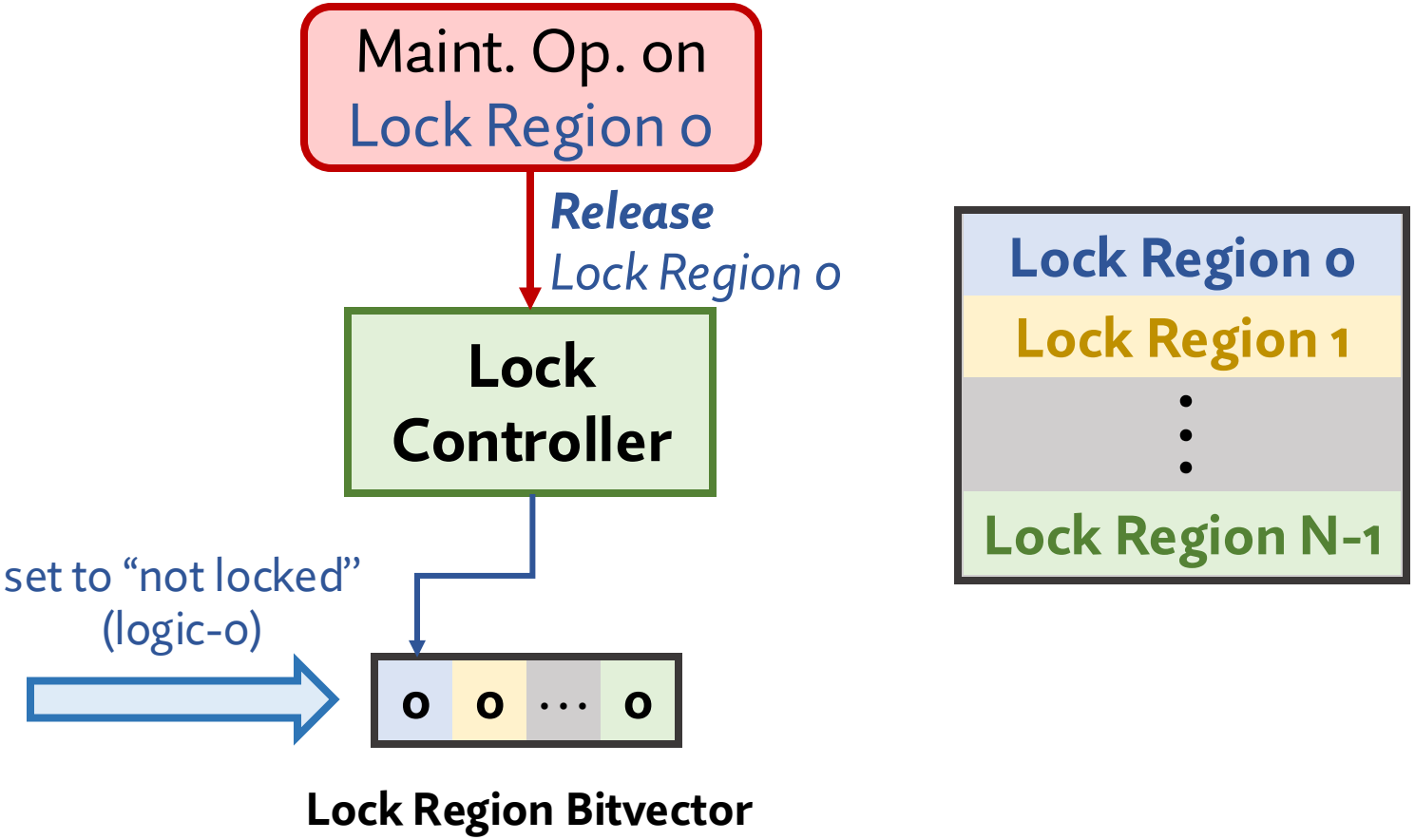
Accessing a Locked Region



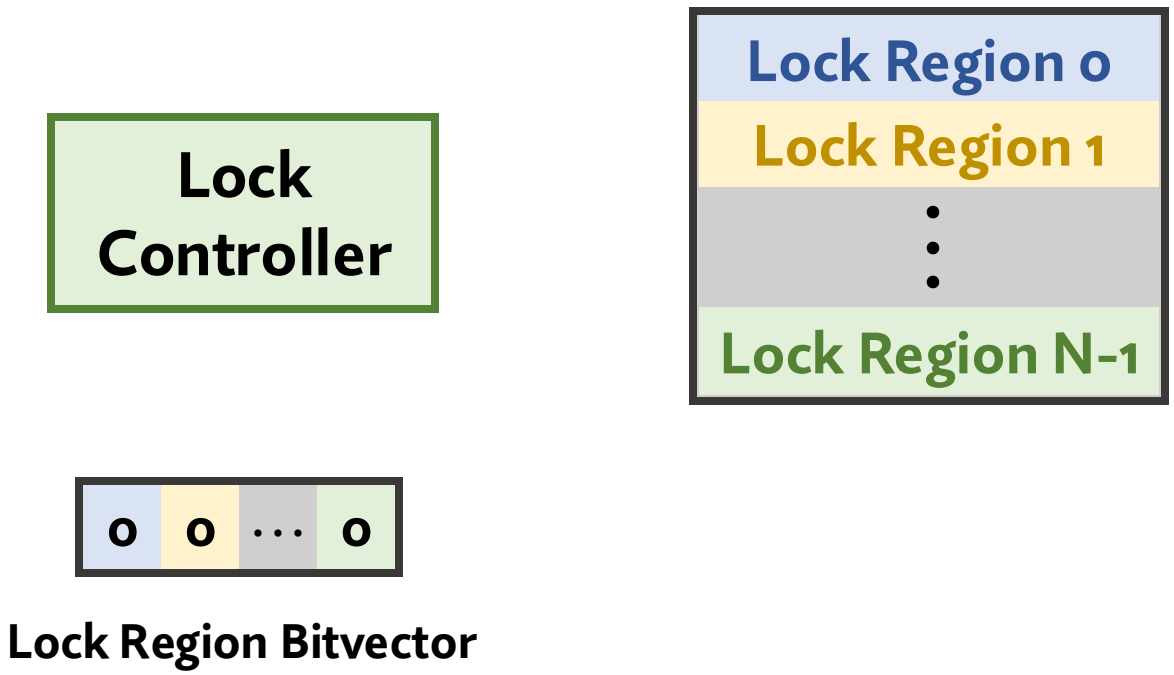
Releasing a Region



Releasing a Region



Releasing a Region



Deeper Look at SMD

1 SMD Bank Organization

2 Region Locking Mechanism

3 Controlling an SMD Chip

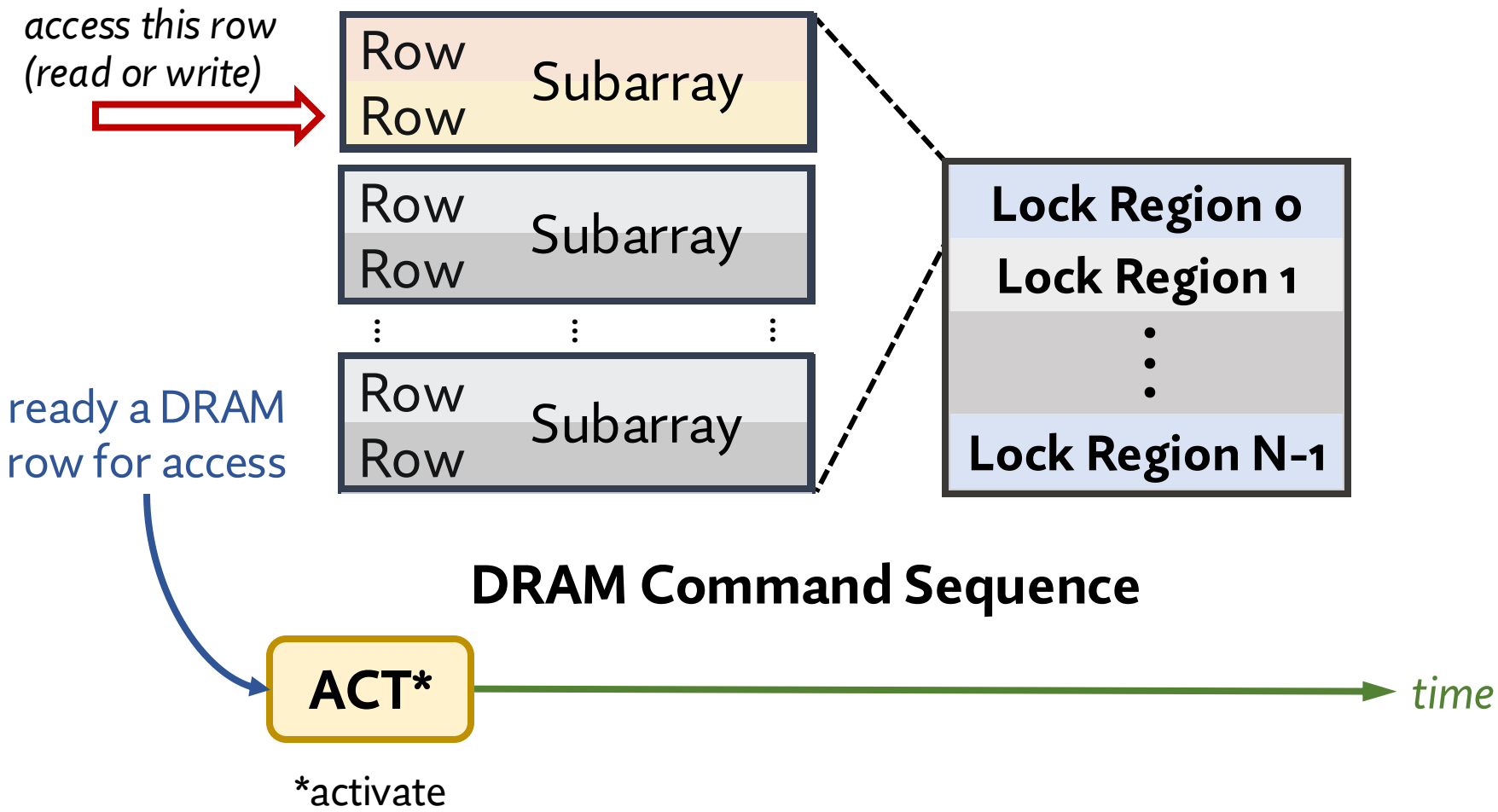
Summary of SMD Chip Control

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

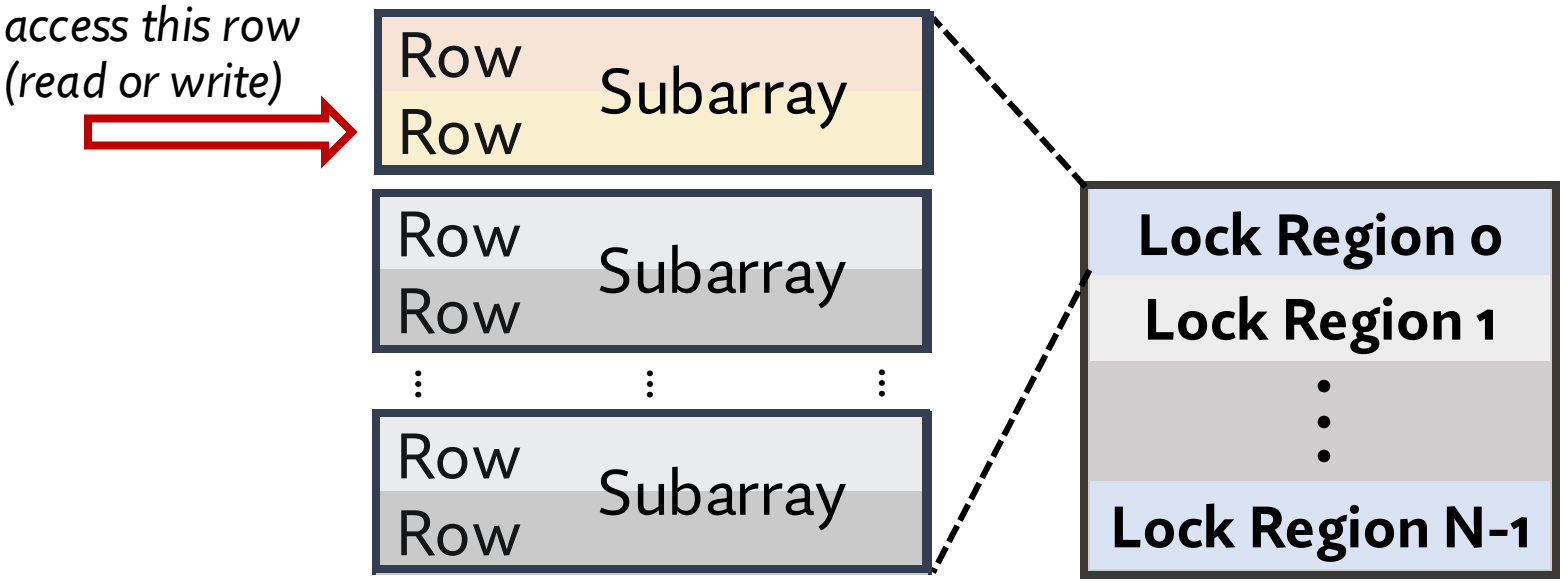
1. Activate commands can get rejected (negative ack)
2. Memory controller retries rejected commands
3. Memory controller can attempt to access other lock regions
4. SMD chip and memory controller ensure forward progress for memory requests

DRAM Control – The "Activate" Command

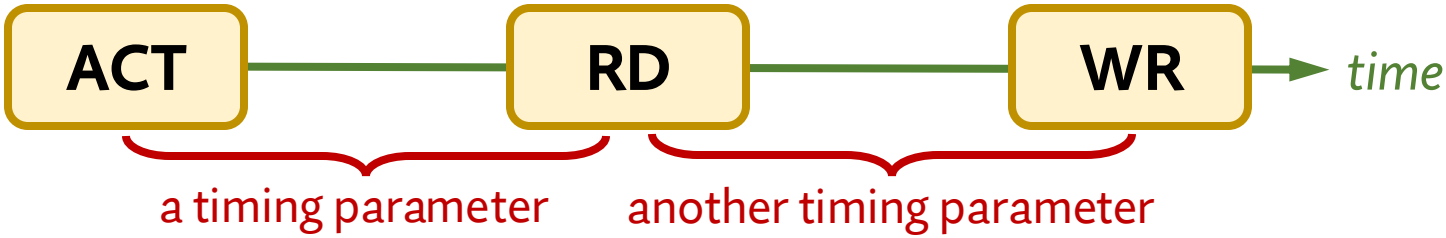


DRAM Control – Timing Parameters

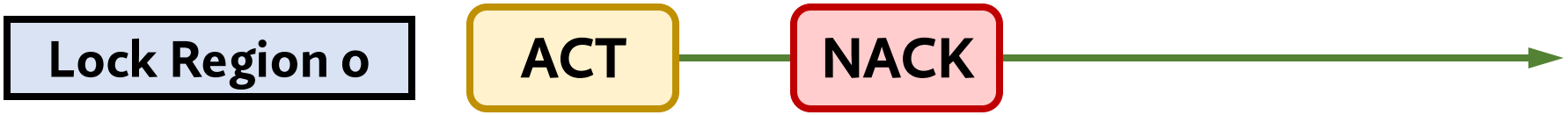
- Timing parameter: Minimum delay between two commands

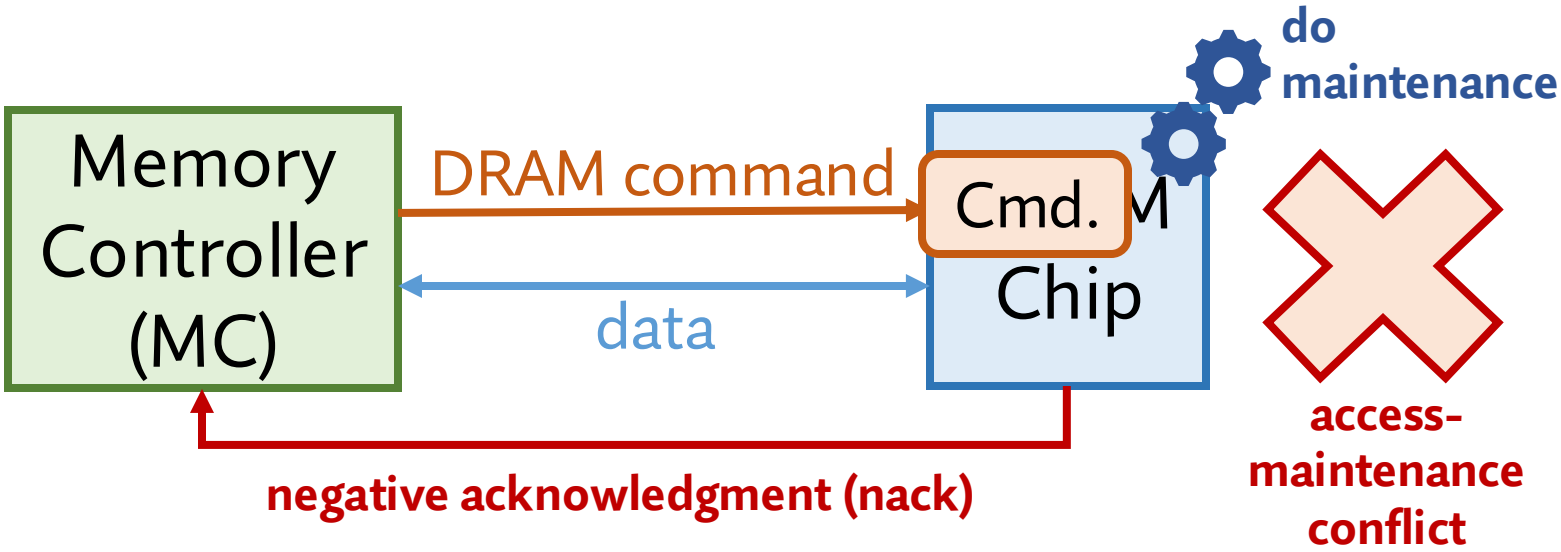


DRAM Command Sequence



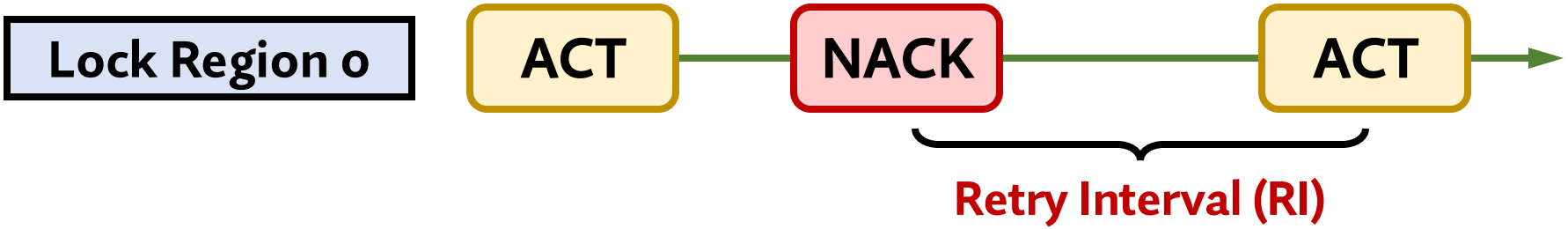
SMD Control – Handling a Rejection





SMD Control – Handling a Rejection

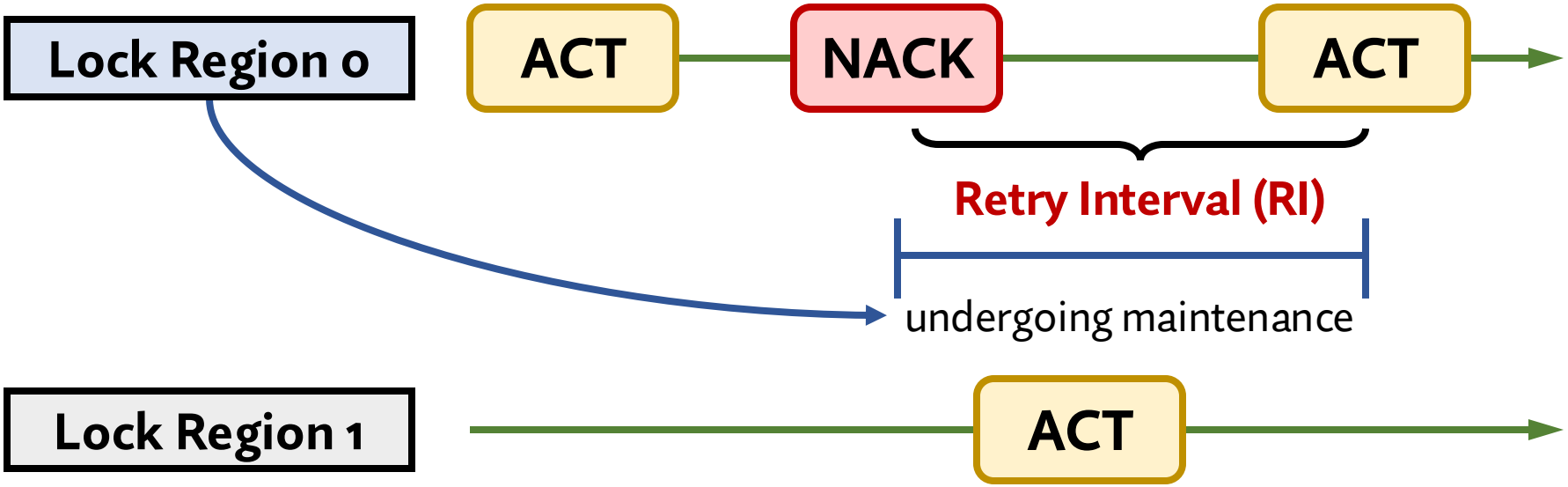
Key idea: Introduce a new timing parameter



- **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

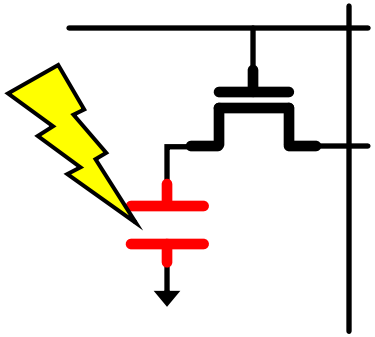
Evaluate

Variable-Rate Refresh
Probabilistic RowHammer Protection

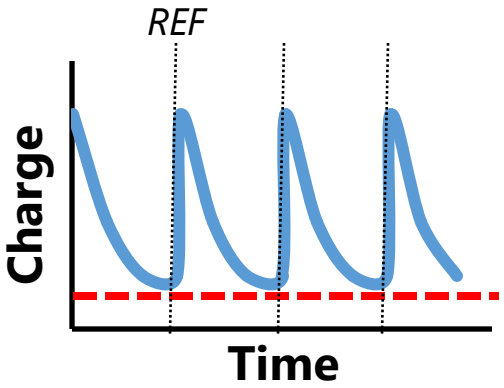
Discuss

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

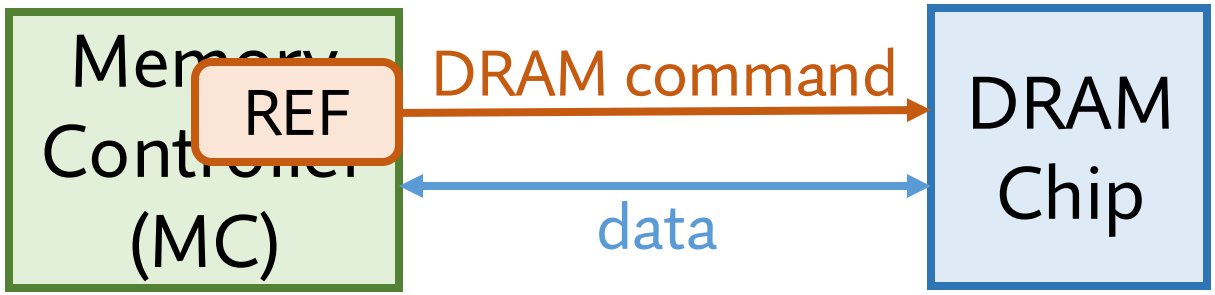
DRAM Periodic Refresh



DRAM encodes data in **leaky capacitors**



Necessitates periodic **refresh operations**



[Patel+, DSN'19]

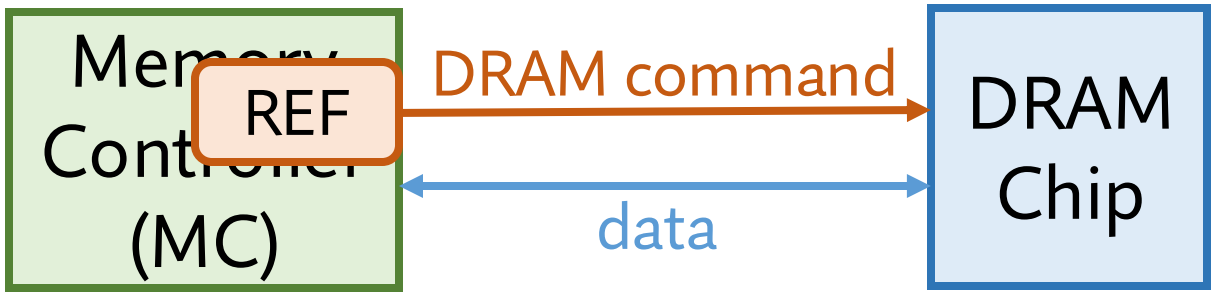
Alleviating the Drawbacks of Periodic Refresh

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

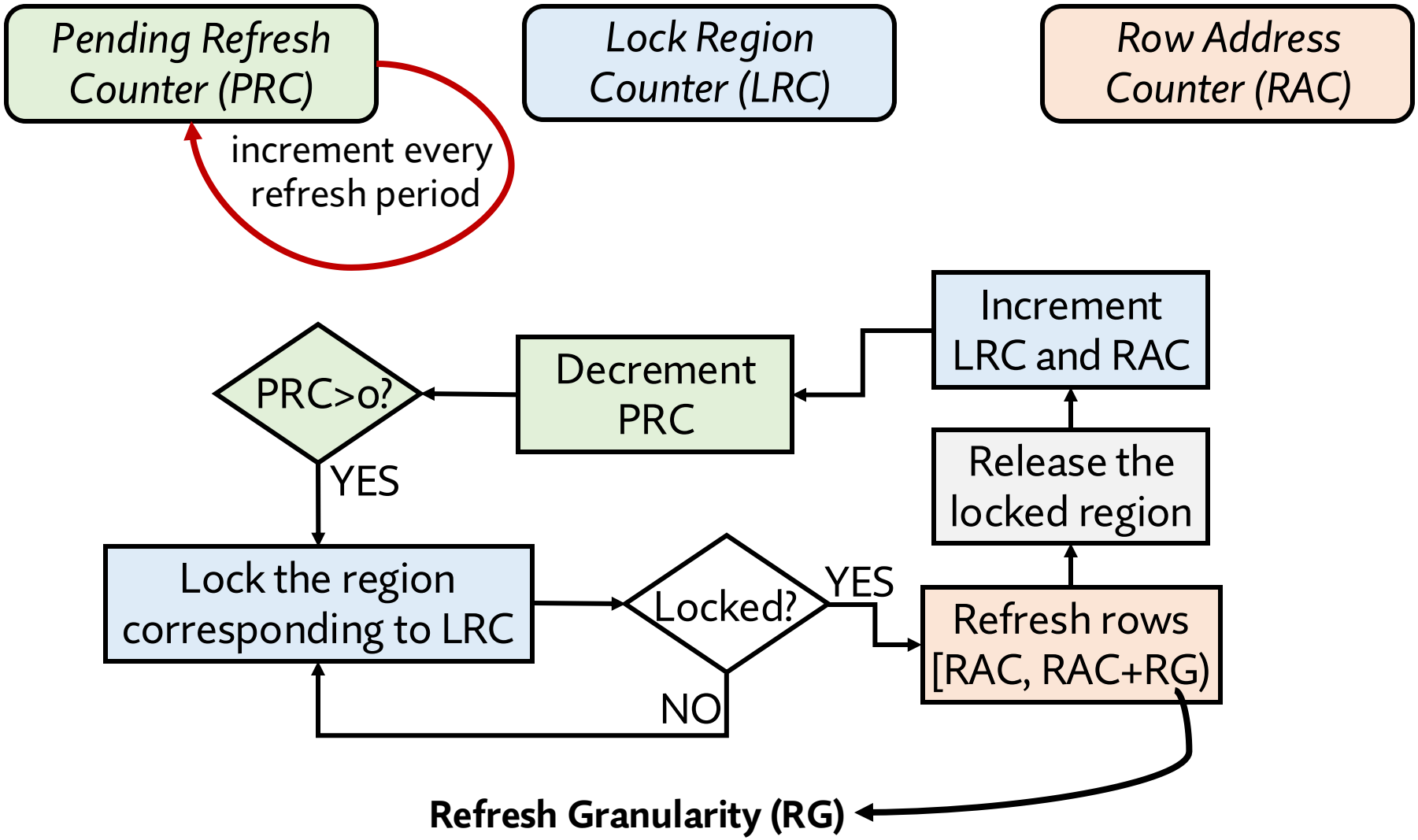
ii Entire chip or bank inaccessible during refresh

- 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

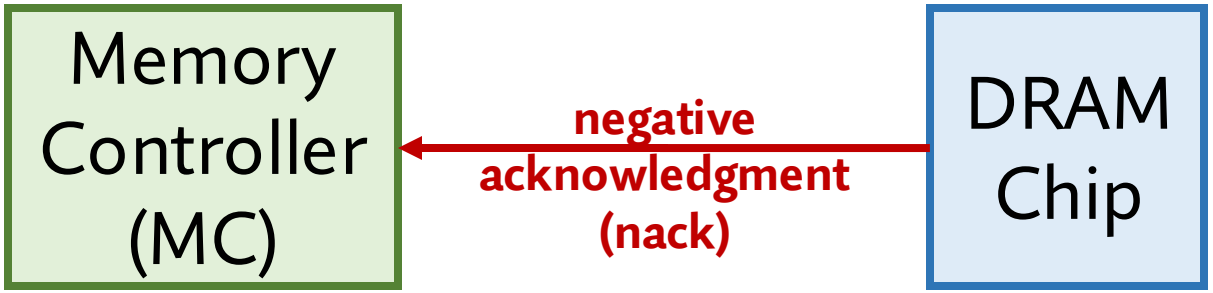
1. Motivation
2. Self-Managing DRAM (SMD)
3. Use Cases
4. **Evaluations**
5. Conclusion and Takeaways

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 mm² DRAM chip

ii

Maintenance-access parallelization

1.1%*
of a 45.5 mm² 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

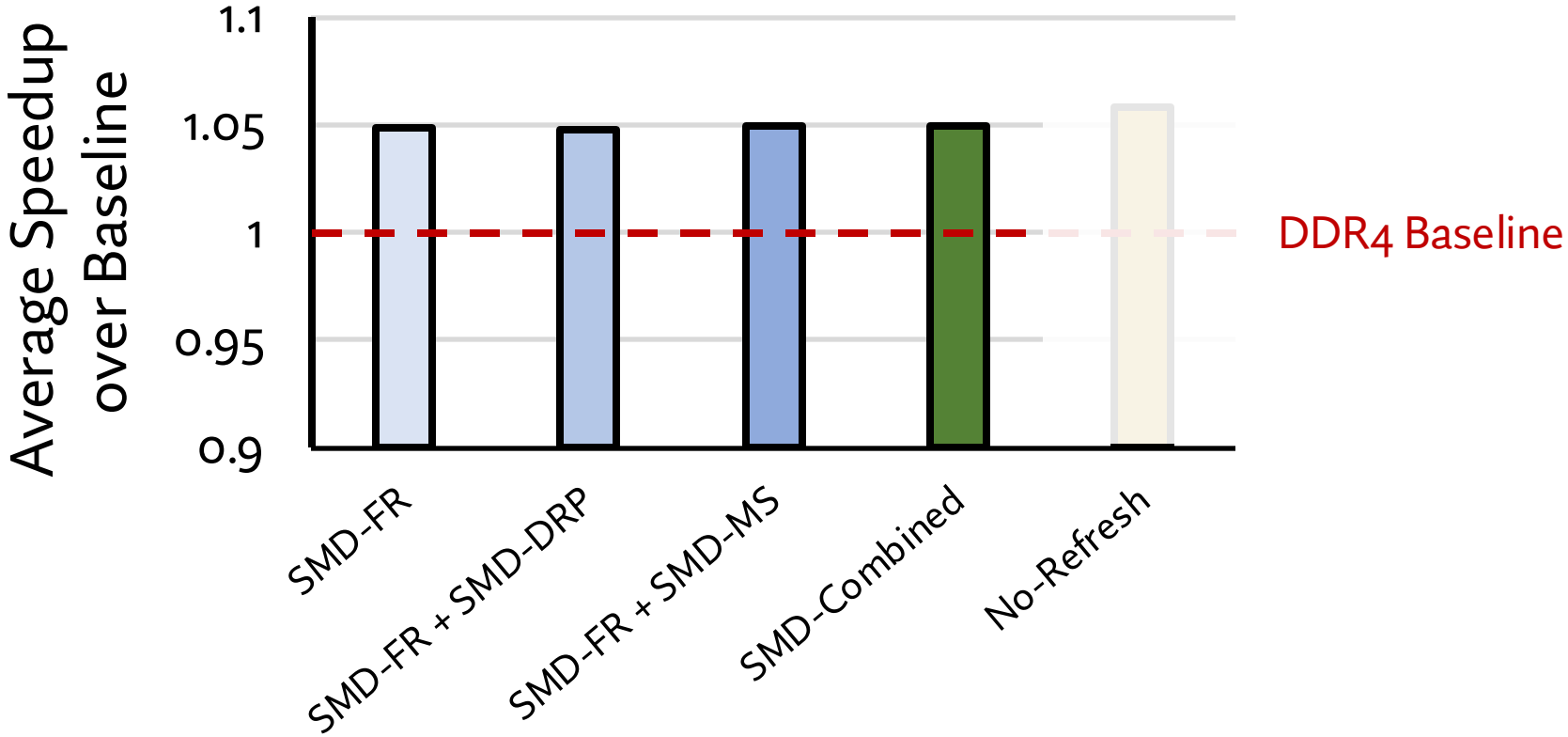
- **SMD** parameters
  - 16 lock regions in a DRAM bank
  - 16 subarrays in one lock region
  - Retry Interval (RI) = 62.5 nanoseconds

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

Evaluated System Configurations

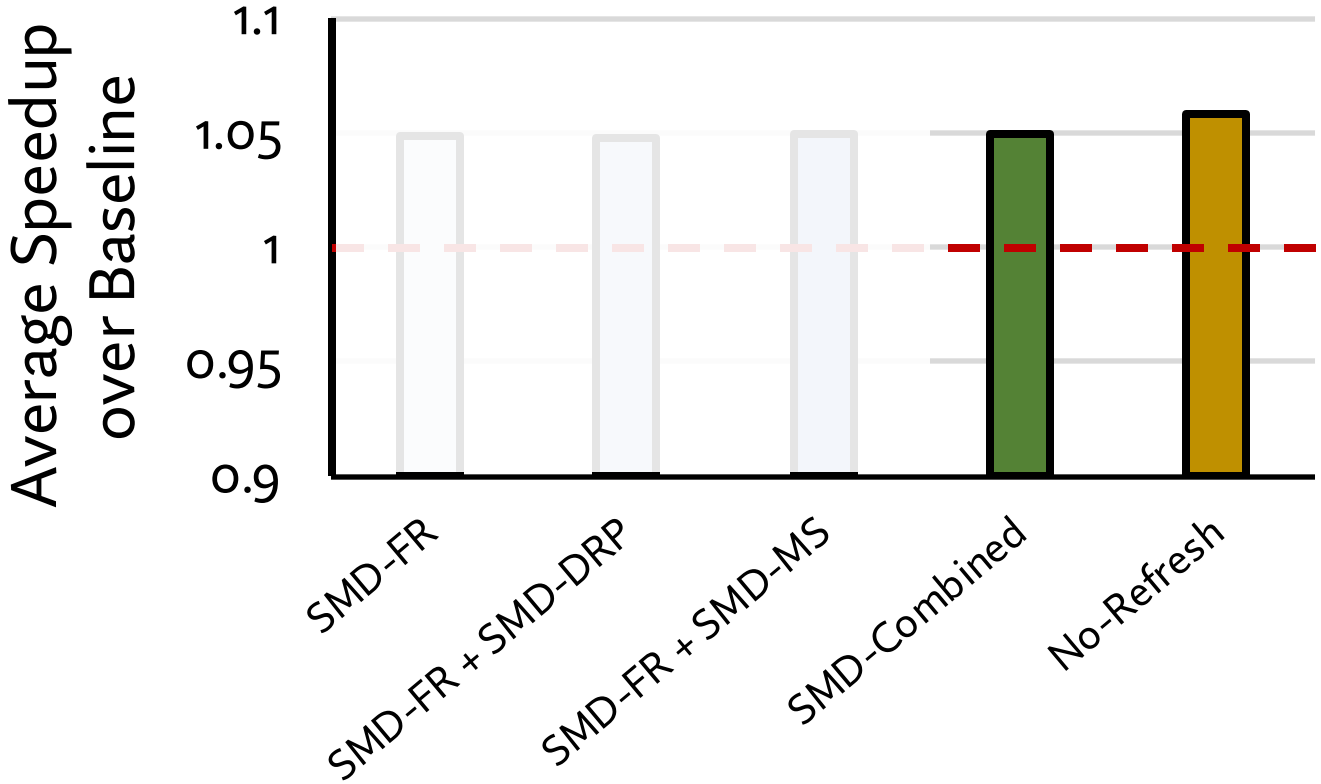
- Baseline DDR4 system
  - refresh window = 32 millisecond
- Fixed-Rate Refresh **(SMD-FR)**
  - refresh window = 32 millisecond, refresh granularity = 8
- Deterministic RowHammer Protection **(SMD-FR + SMD-DRP)**
  - refresh neighbor rows of a row that gets activated 512 times
- Memory Scrubbing **(SMD-FR + 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



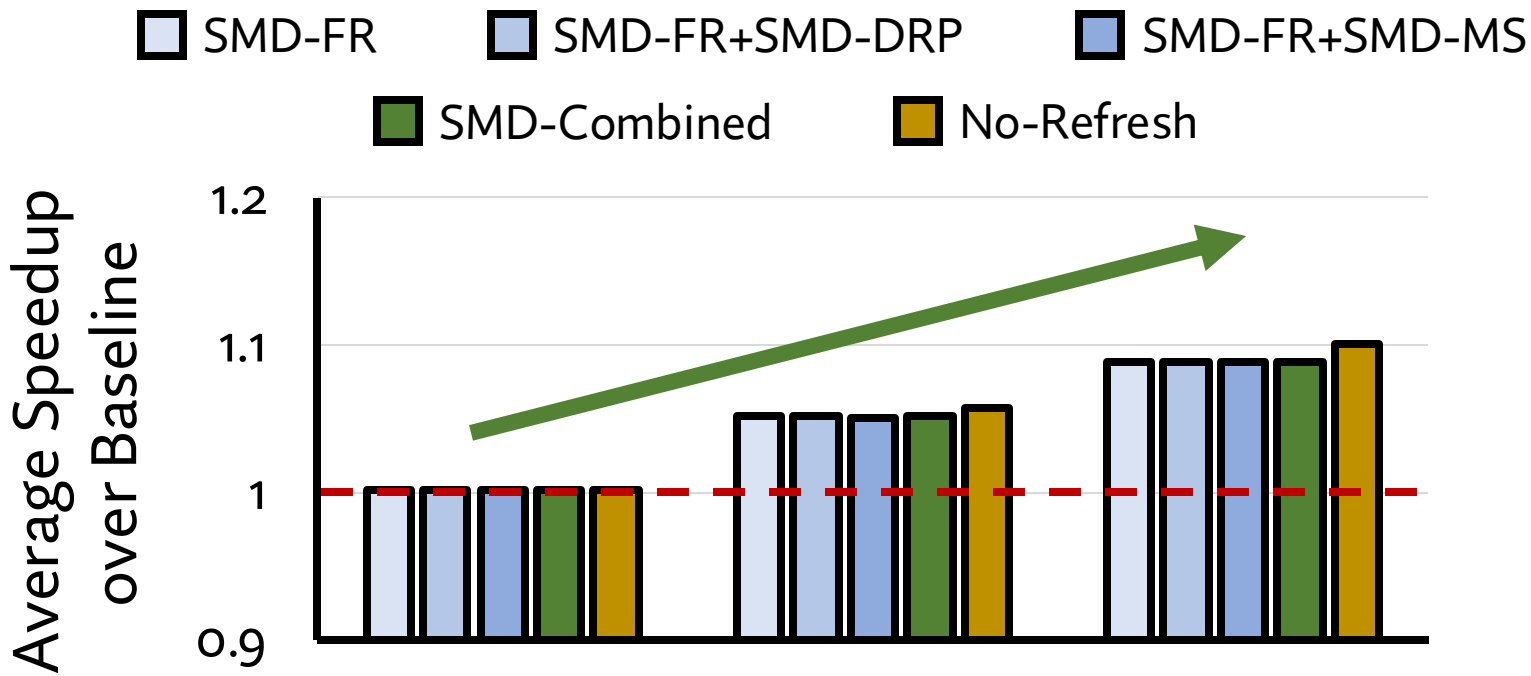
SMD provides 4.8% to 5.0% average speedup

Single-Core Performance



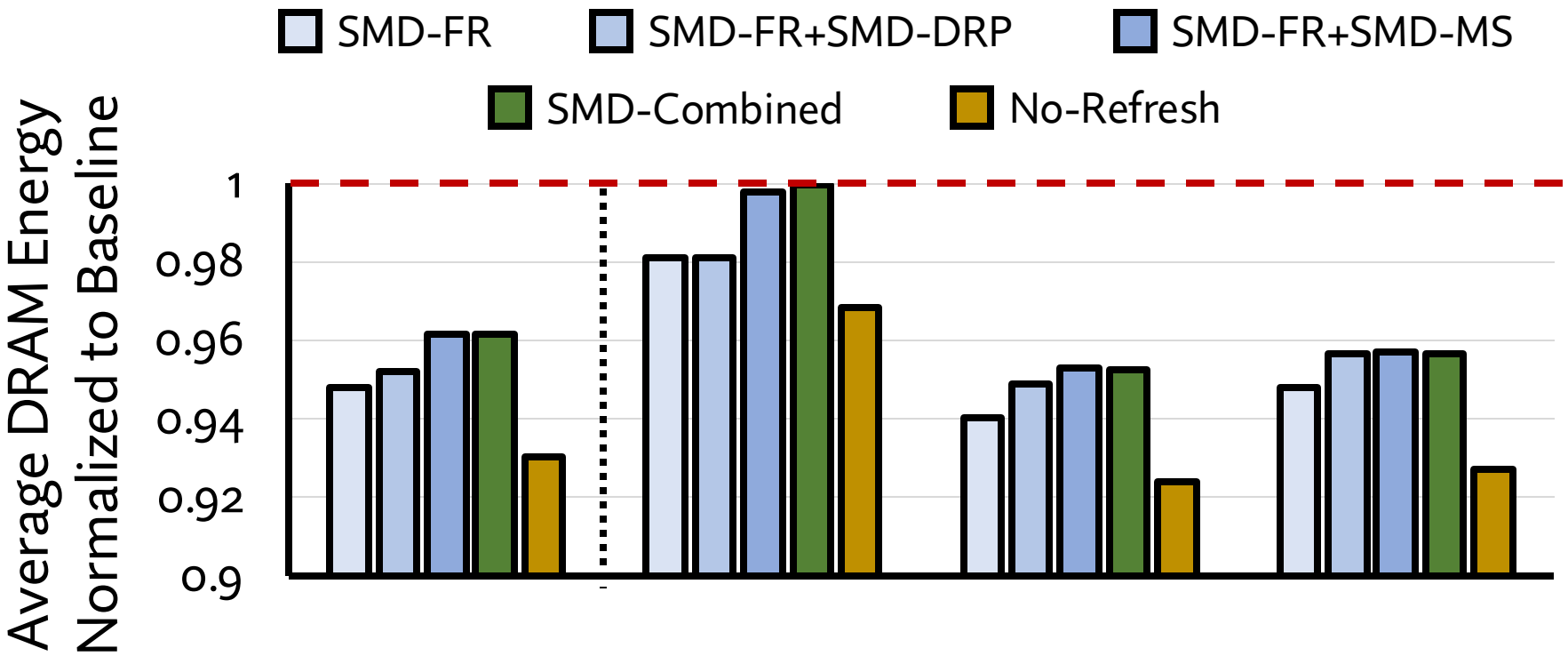
SMD-Combined provides
84.7% the speedup of No-Refresh

Four-Core Performance



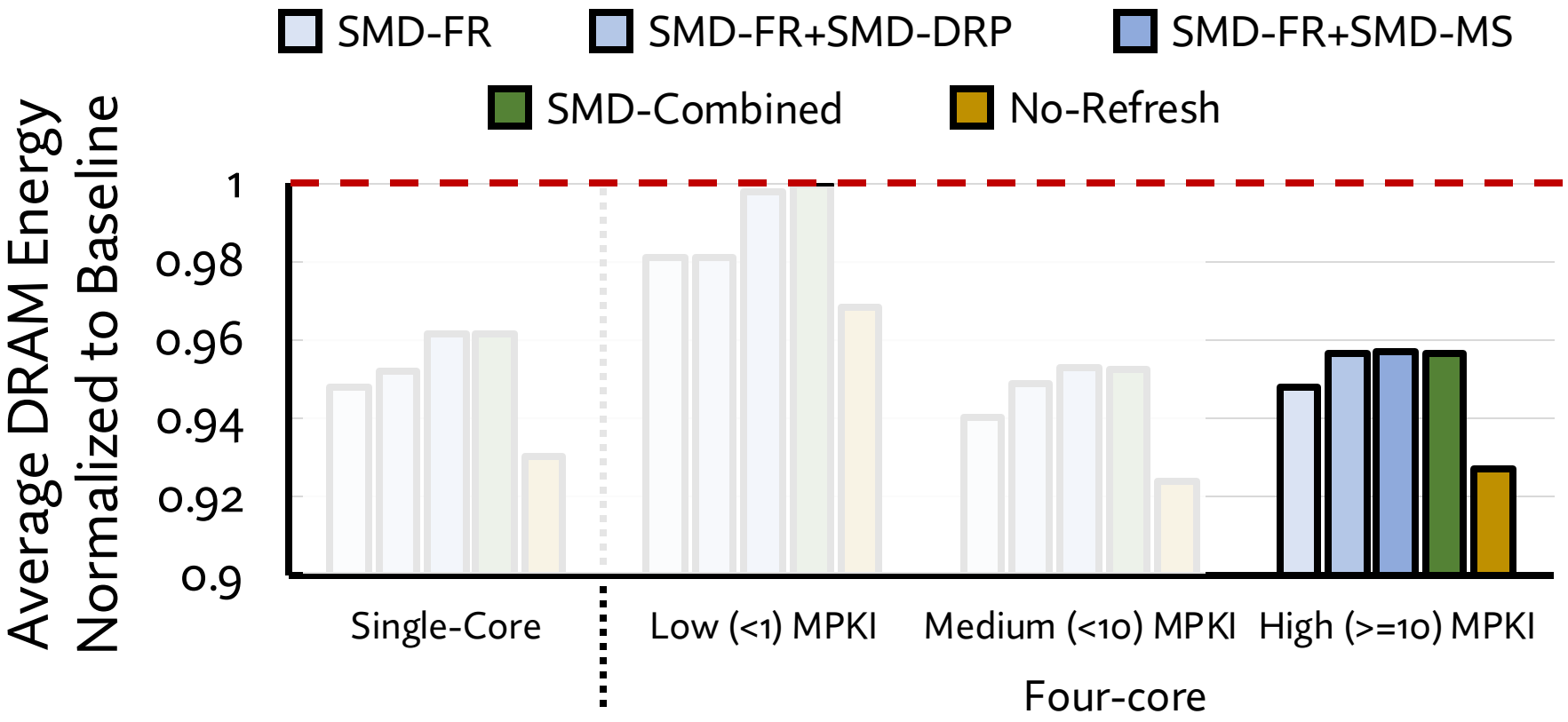
SMD provides higher speedups with increasing workload memory intensity

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

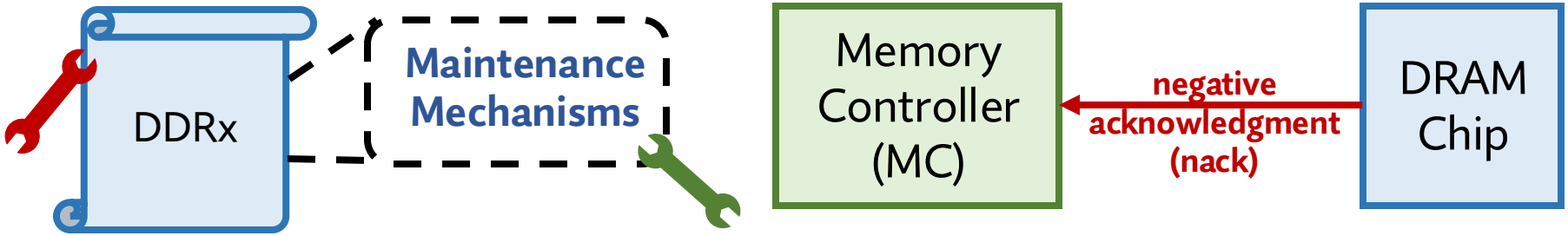
tion [12, 18, 47–122], and 3) memory scrubbing [17, 123–135].[1] New DRAM chip generations necessitate making existing maintenance operations more aggressive (e.g., lowering the refresh period [119, 136, 137]) and introducing new types of maintenance operations (e.g., targeted refresh [64, 66, 138],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

CMU-SAFARI / SelfManagingDRAM

Type / to search

Code | Issues | Pull requests | Actions | Projects | Wiki | Security | Insights

SelfManagingDRAM (Public)

Edit Pins | Watch 6 | Fork 1 | Star 5

main | 1 Branch | Tags | Go to file | Add file | Code

olgunataberk add new SMD, MC-based Graphene, and RAIDR sources 0427891 · last year 8 Commits

configs	initial commit	2 years ago
scripts	add scripts for SMD configurations in the paper	2 years ago
src	add new SMD, MC-based Graphene, and RAIDR sources	last year
.gitignore	initial commit	2 years ago
LICENSE	initial commit	2 years ago
Makefile	initial commit	2 years ago
README.md	Update README.md	2 years ago
run.sh	add descriptions to run.sh config parameters	2 years ago

README | MIT license

Self-Managing DRAM (SMD)

About

Source code for evaluating the performance and DRAM energy benefits of Self-Managing DRAM (SMD), proposed in https://arxiv.org/abs/2207.13358

Readme
MIT license
Activity
Custom properties
5 stars
6 watching
1 fork
Report repository

Releases

No releases published
Create a new release

Packages

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

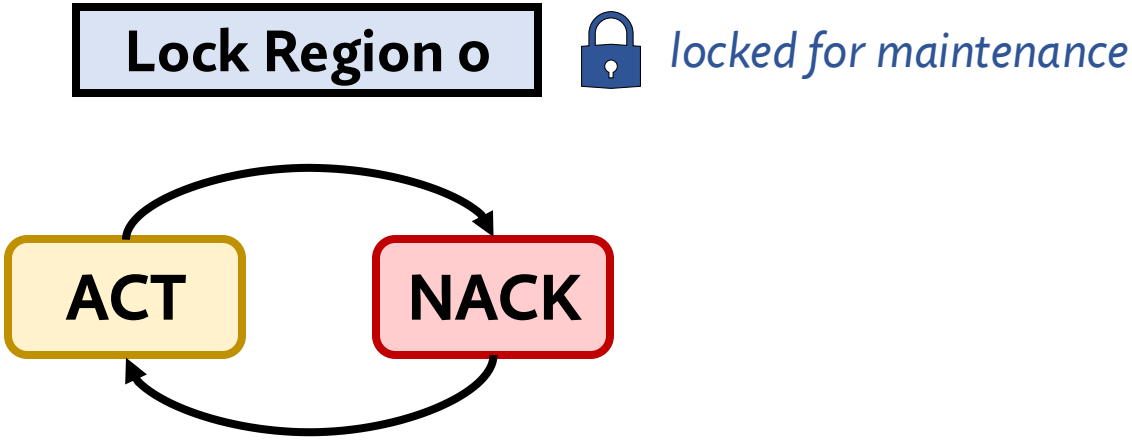
ETH zürich

SAFARI

Backup Slides

Ensuring Forward Progress

- SMD breaks the chain of ACT commands and rejections



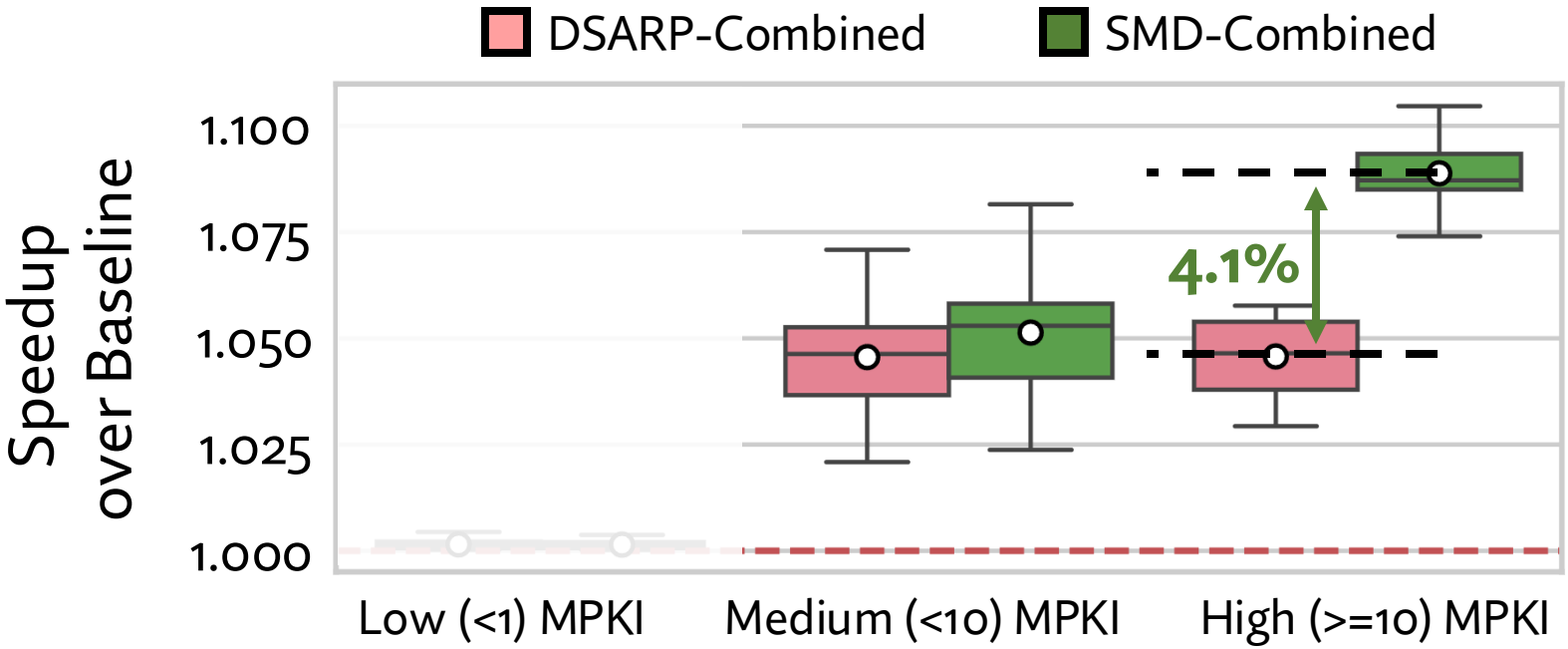
- because:

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

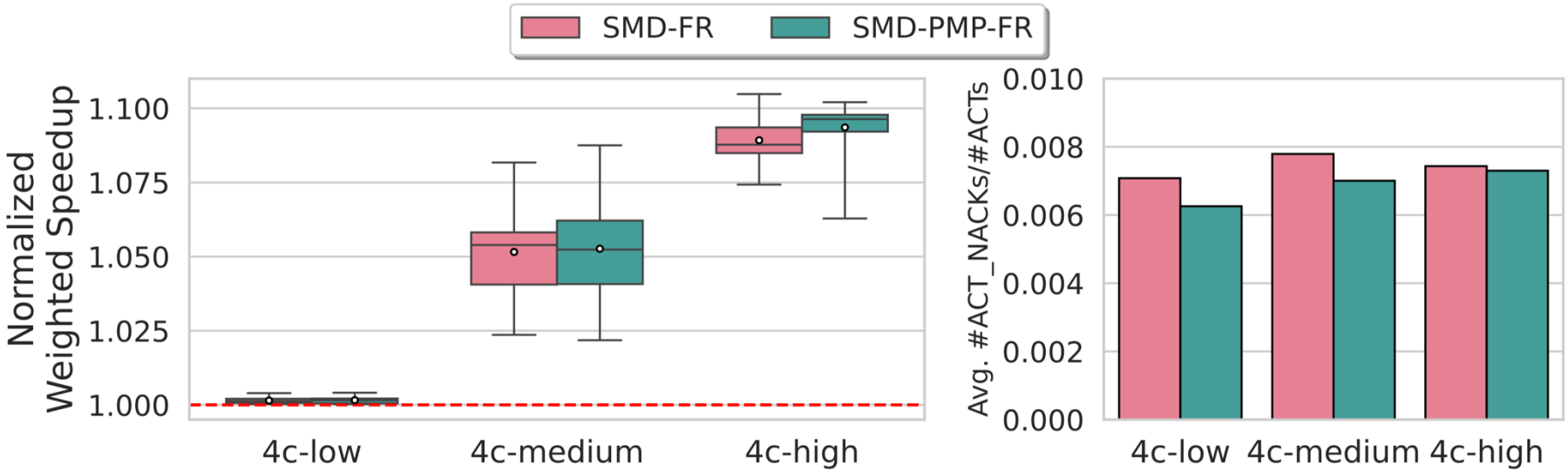
Performance Comparison

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

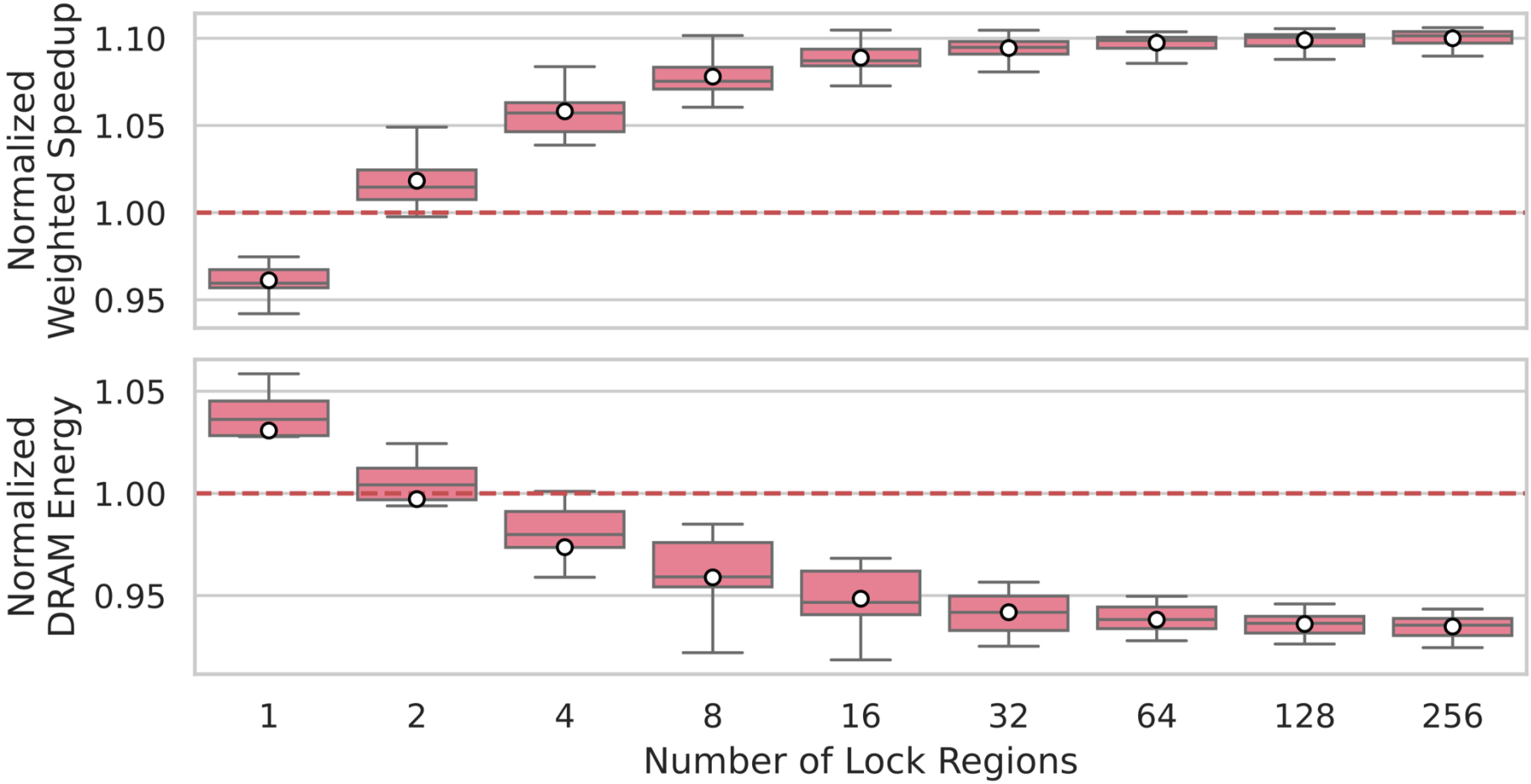


SMD outperforms DSARP

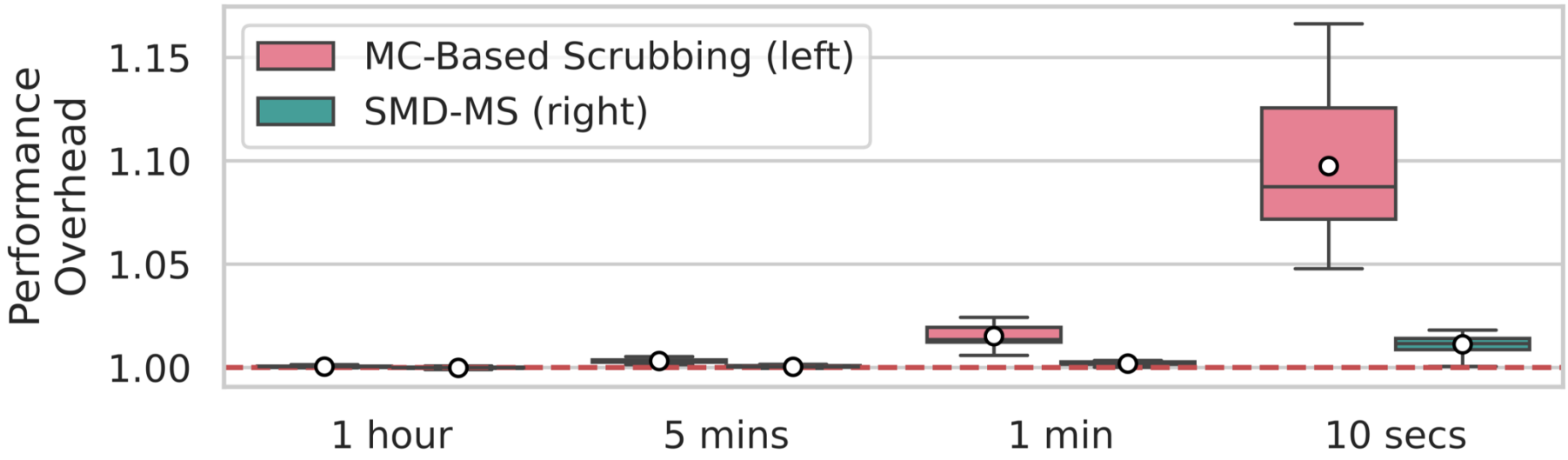
Pause Maintenance Policy



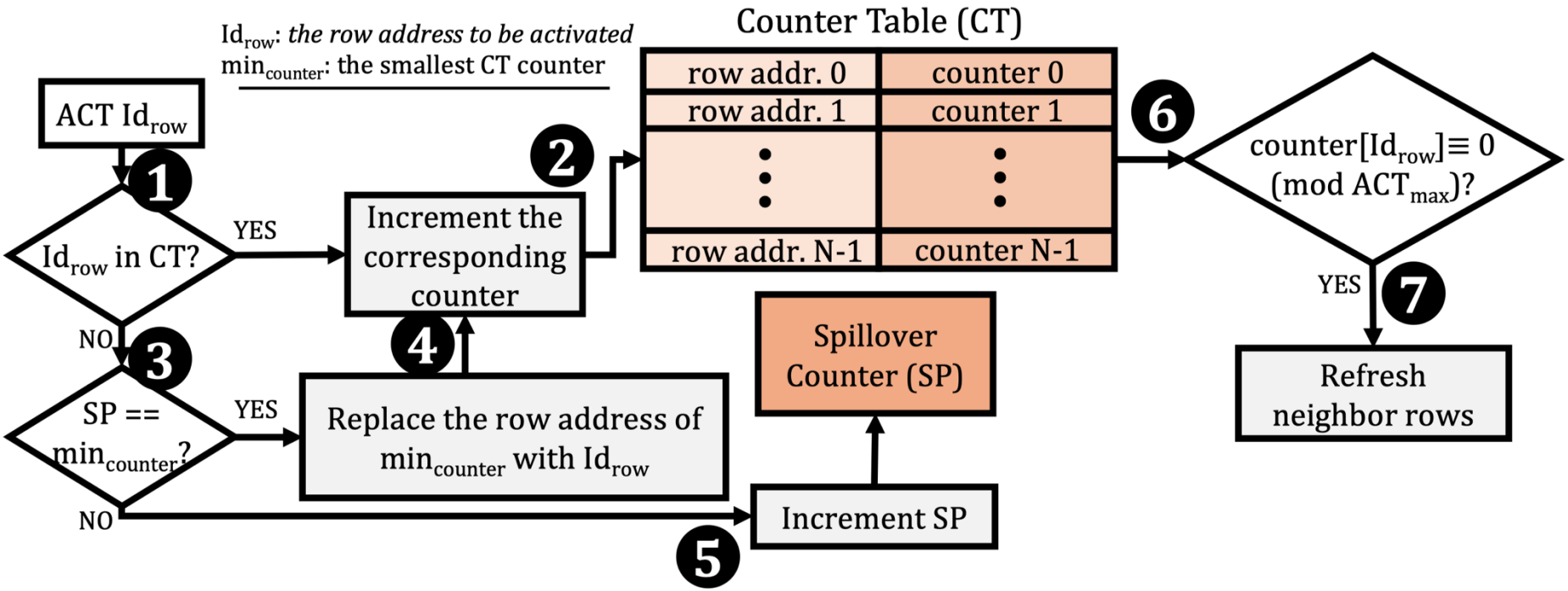
Sensitivity to Number of Lock Regions



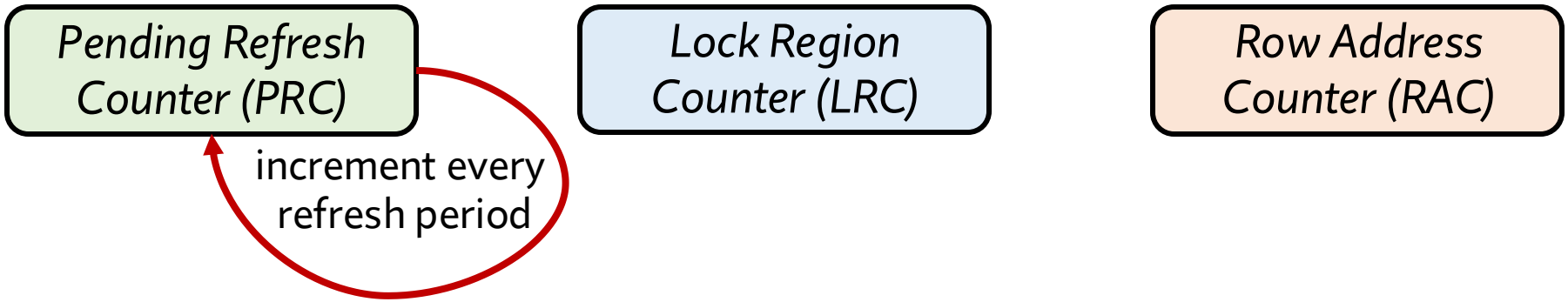
SMD-based vs. MC-based Scrubbing



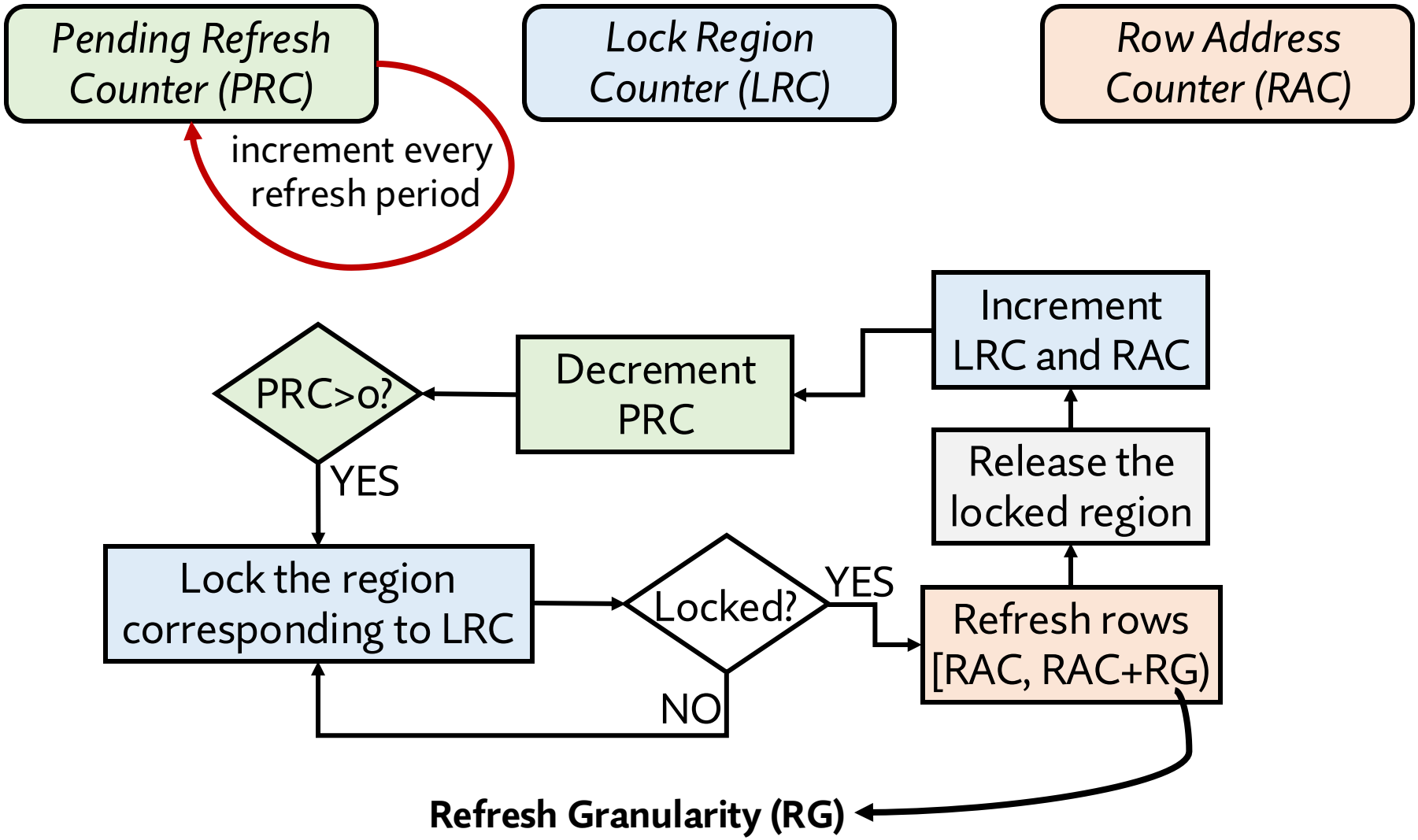
SMD-DRP


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

SMD-FR – Implementation

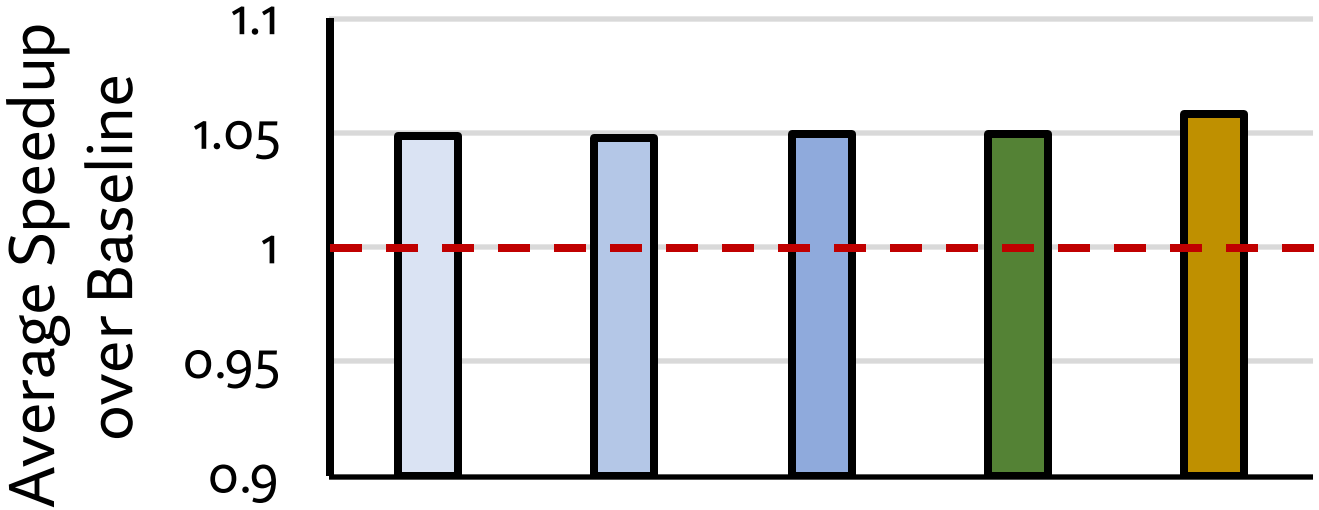


SMD-FR – Implementation



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.