Design-Induced Latency Variation in Modern DRAM Chips: Characterization, Analysis, and Latency Reduction Mechanisms

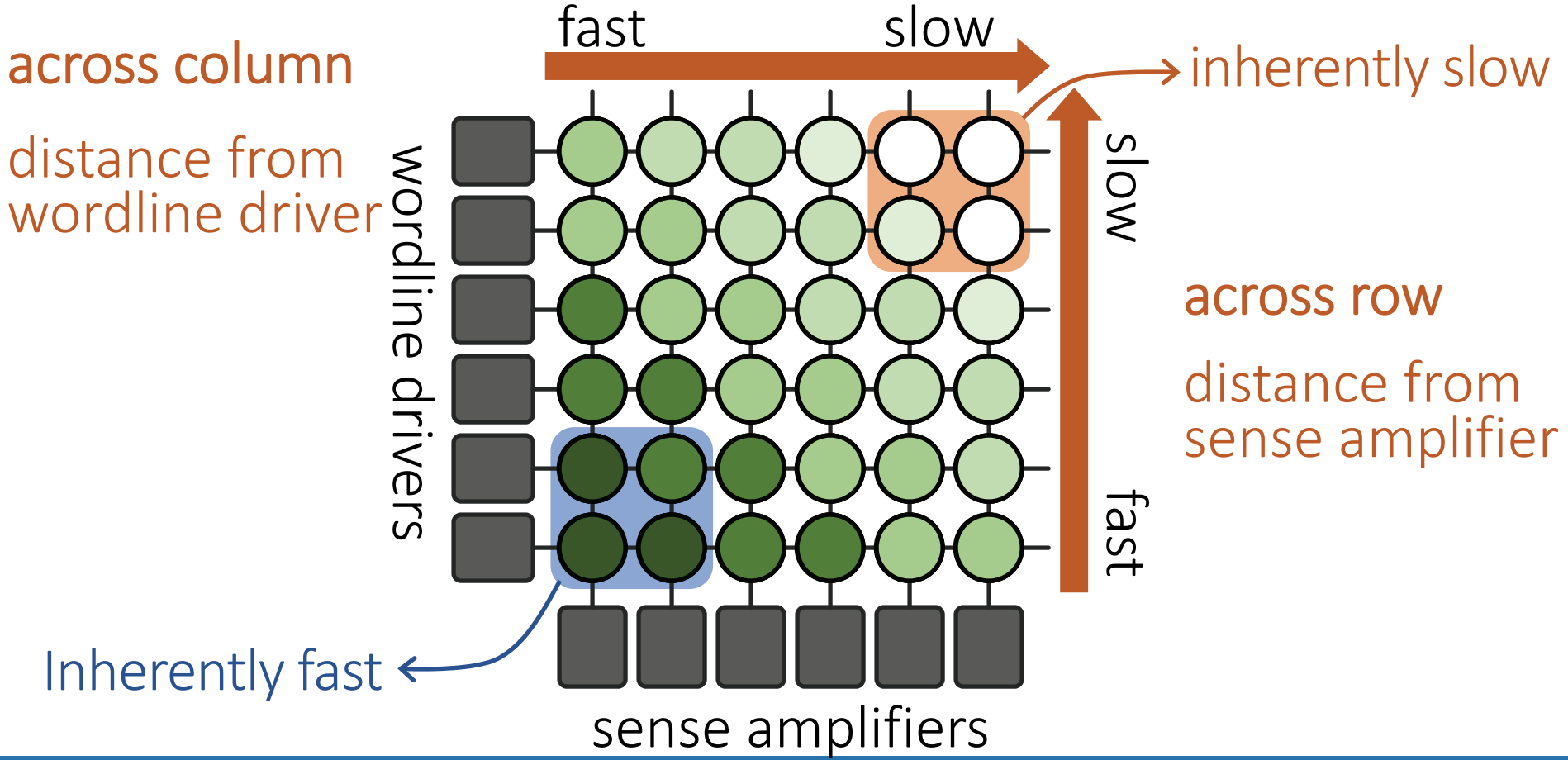
Donghyuk Lee[1,2] Samira Khan[3] Lavanya Subramanian[2]
Saugata Ghose[2] Rachata Ausavarungnirun[2]
Gennady Pekhimenko[4] Vivek Seshadri[4] Onur Mutlu[5,2]

[1]NVIDIA [2]Carnegie Mellon University
[3]University of Virginia [4]Microsoft Research [5]ETH Zürich

June 7, 2017

SAFARI

What Is Design-Induced Variation?



Systematic variation in cell access times caused by the ***physical organization*** of DRAM

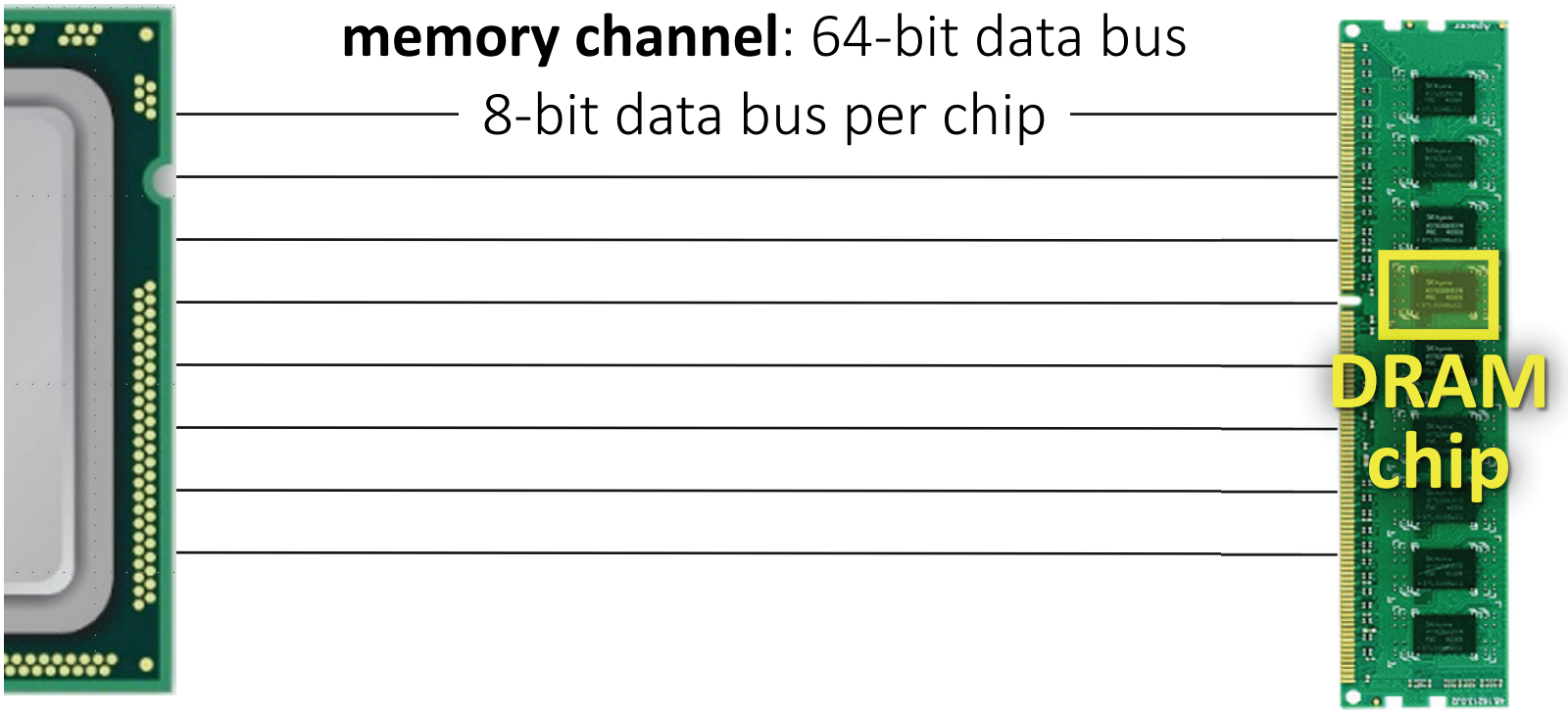
Executive Summary

- **Design-Induced Variation**
  - **Inherently slow regions** due to DRAM cell array organization
  - Goal: Use design-induced variation to reduce DRAM latency
- Analysis: Characterization of **96 real DRAM modules**
  - Three types of *systematic* variation due to design
  - Great ***potential*** to ***reduce DRAM latency*** at low cost
- Our Approach: **DIVA-DRAM**
  - **DIVA Profiling**: Reliably reduce DRAM latency
    - Profile *only* cells in *inherently slow regions*
    - Use error correction (ECC) for slow cells that are not profiled
  - **DIVA Shuffling**: Exploit variation to improve ECC reliability
- **15.1%/14.2% *higher performance*** for 2-/8-core workloads

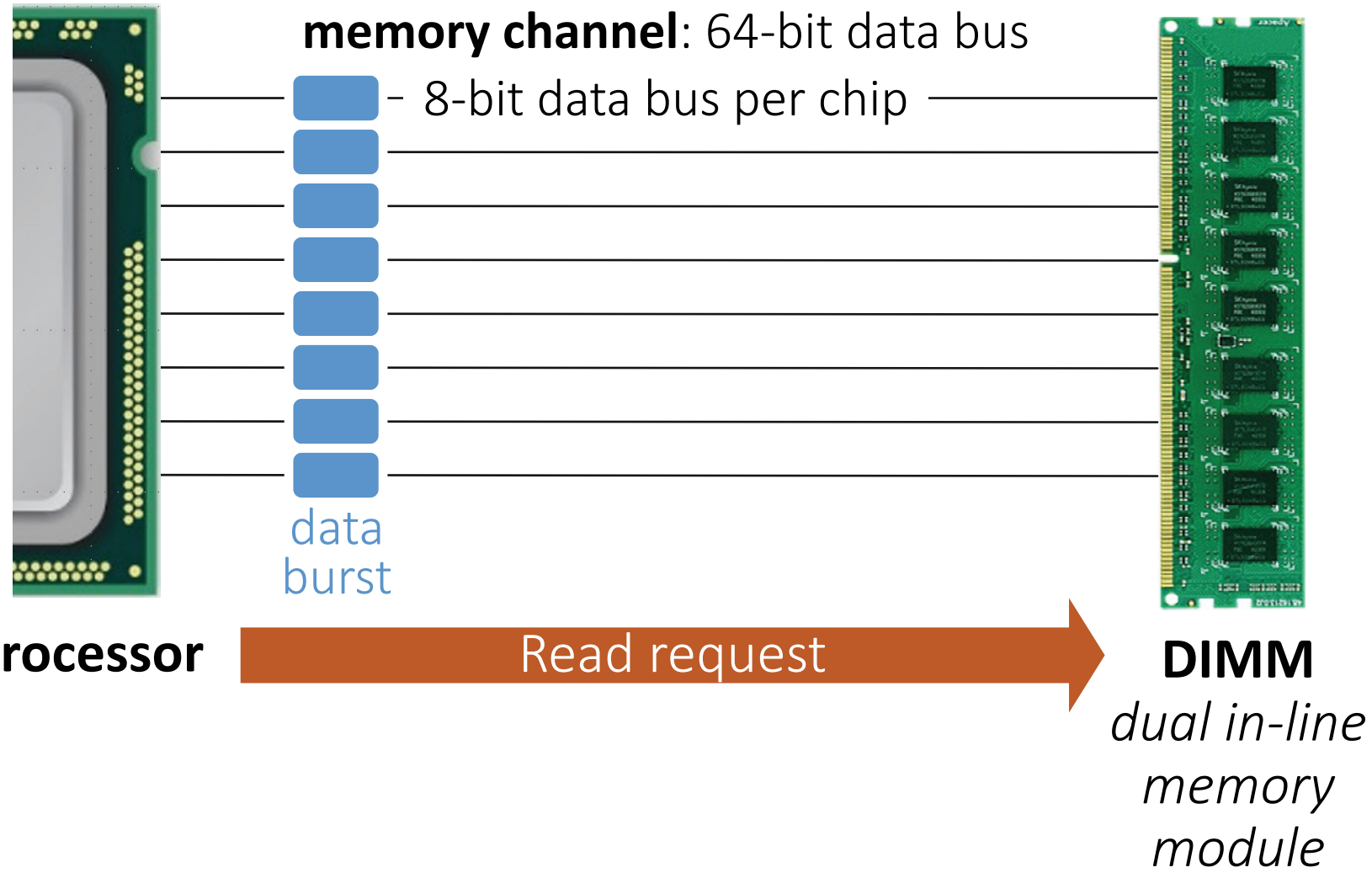
Presentation Outline

- DRAM Background
- Experimental Study of Design-Induced Variation
  - Goal: Understand, identify inherently slower regions
  - Methodology: Profile 96 real DRAM modules by using FPGA-based DRAM test infrastructure
- Exploiting Design-Induced Variation
  - DIVA Profiling: Using low-cost slow region profiling to reliably and dynamically reduce DRAM latency
  - DIVA Shuffling: Using variation across data bursts to reduce uncorrectable errors

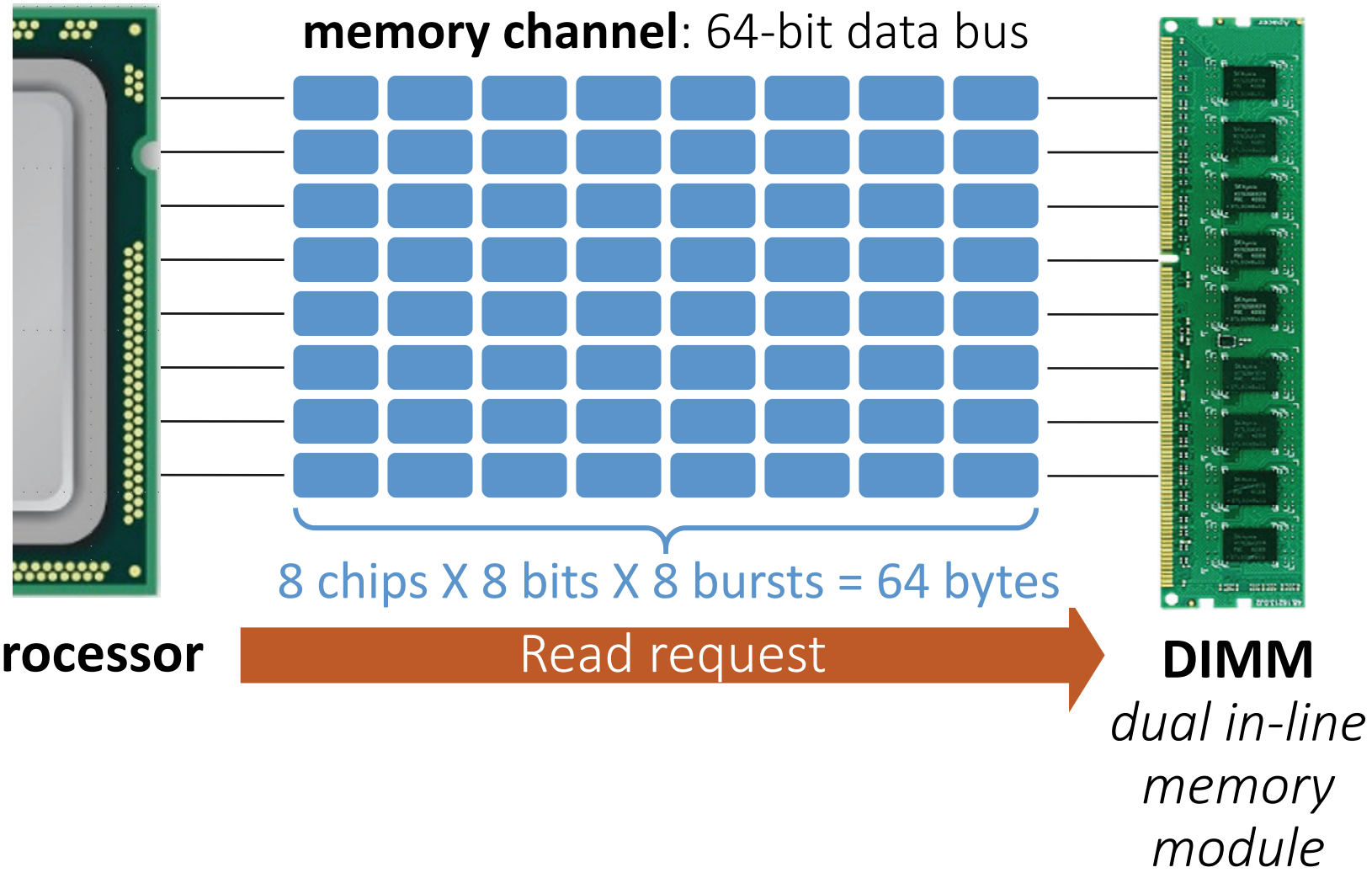
High-Level DRAM Organization



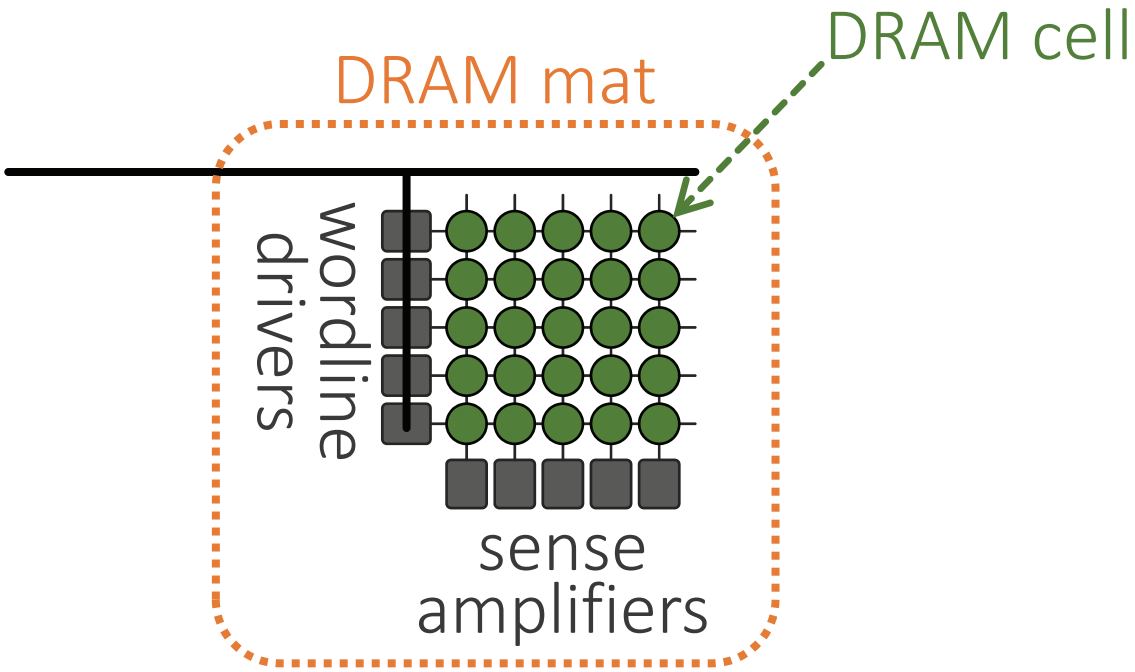
High-Level DRAM Organization



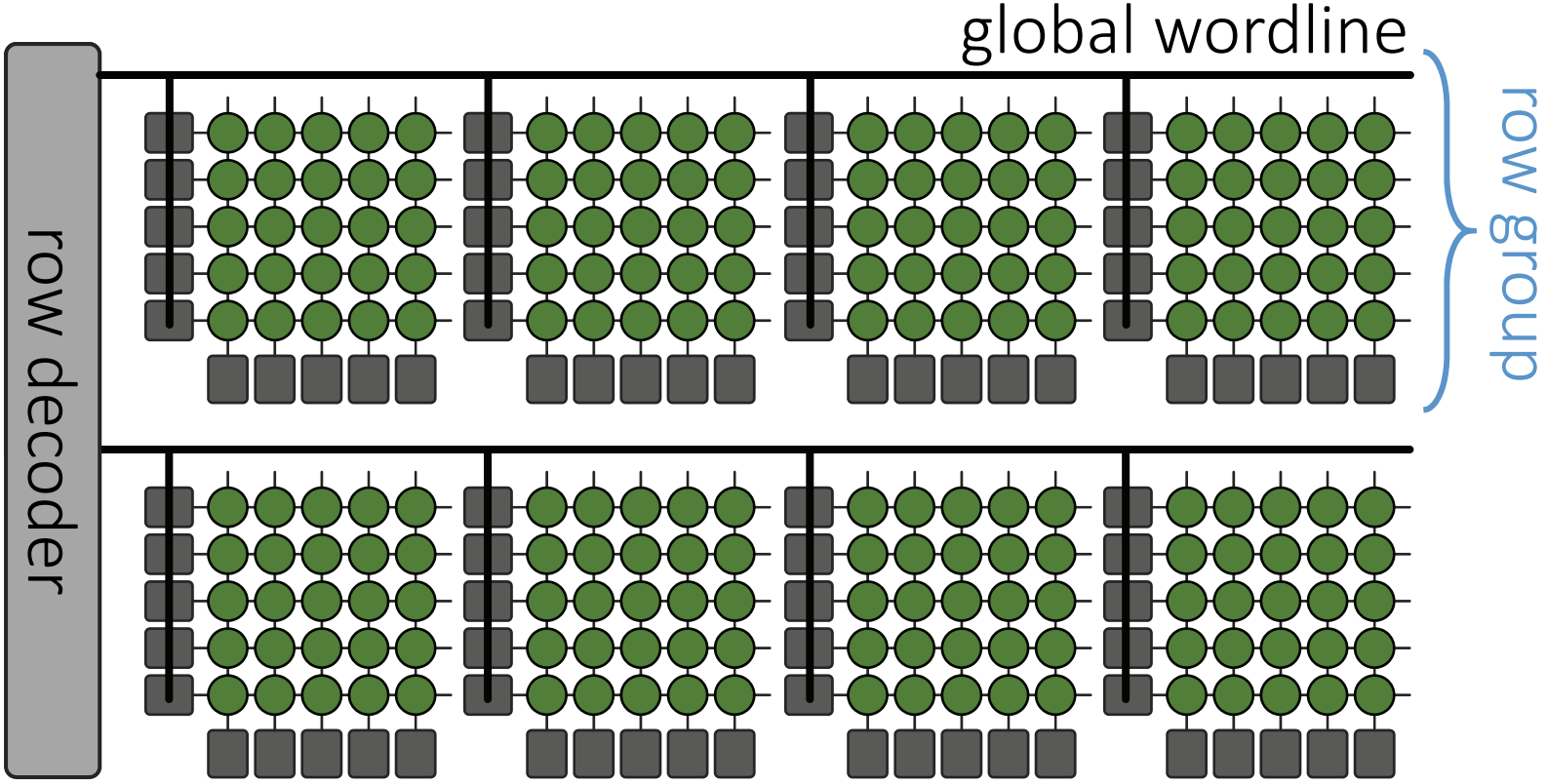
High-Level DRAM Organization



Organization Inside a DRAM Chip

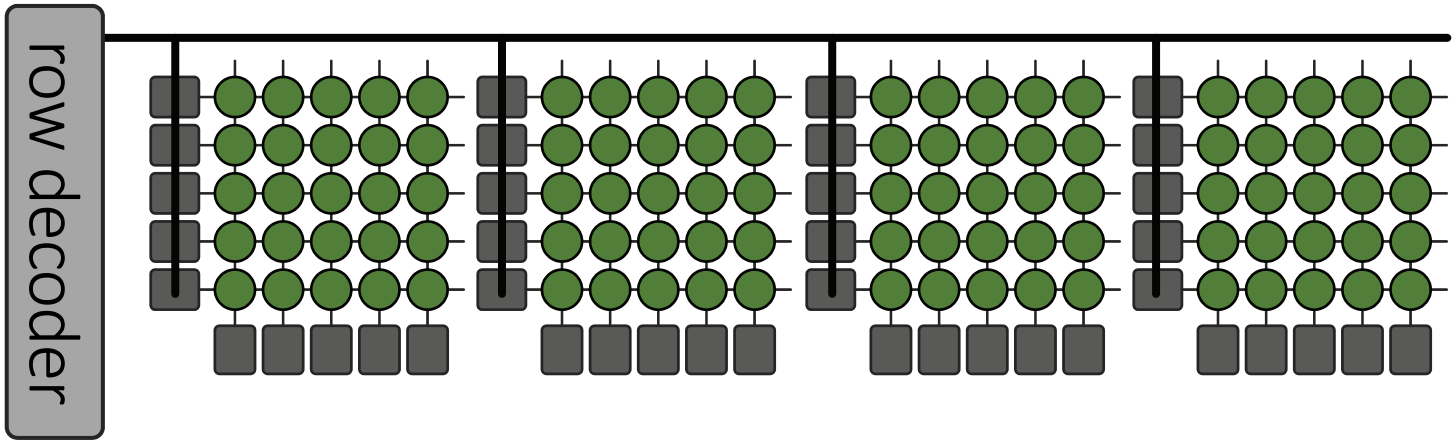


Organization Inside a DRAM Chip



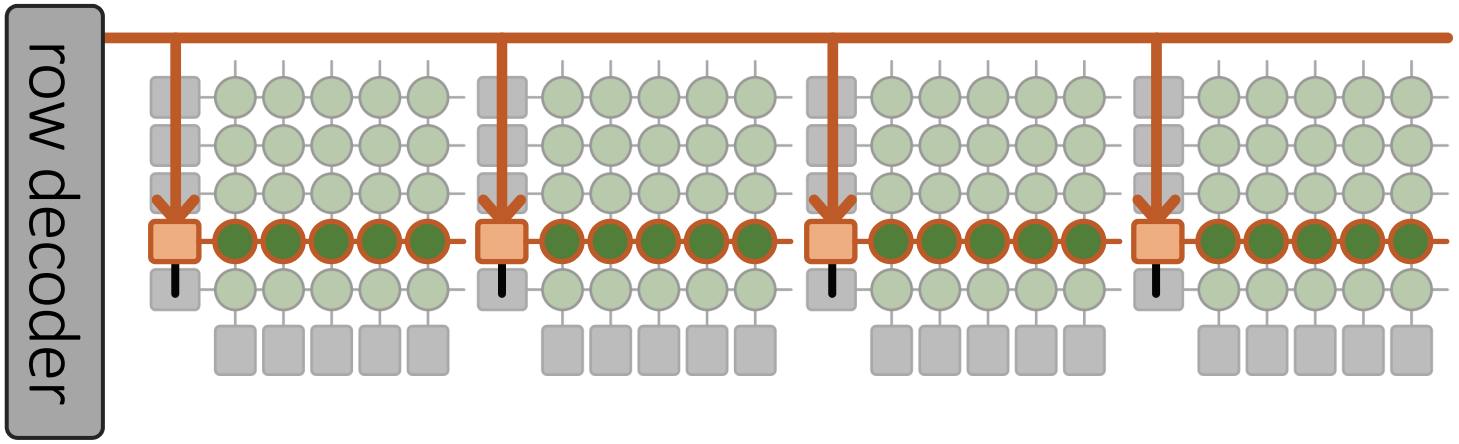
DRAM Operations

- *Memory controller* sends commands to DRAM



DRAM Operations

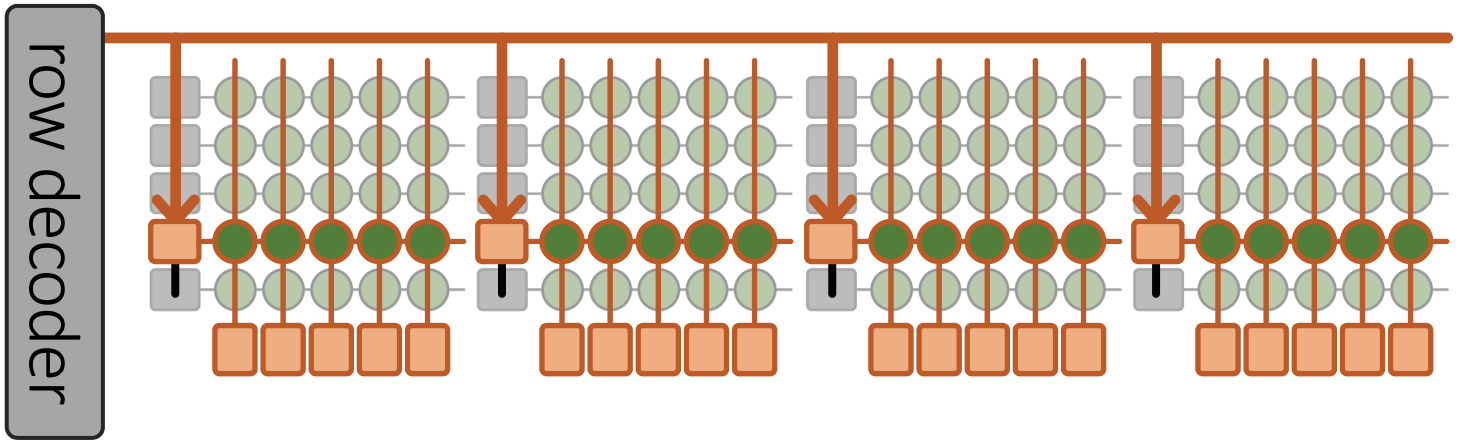
- *Memory controller* sends commands to DRAM



- **Activation**: Open one row in each mat
  - Row decoder, wordline drivers select a row of cells

DRAM Operations

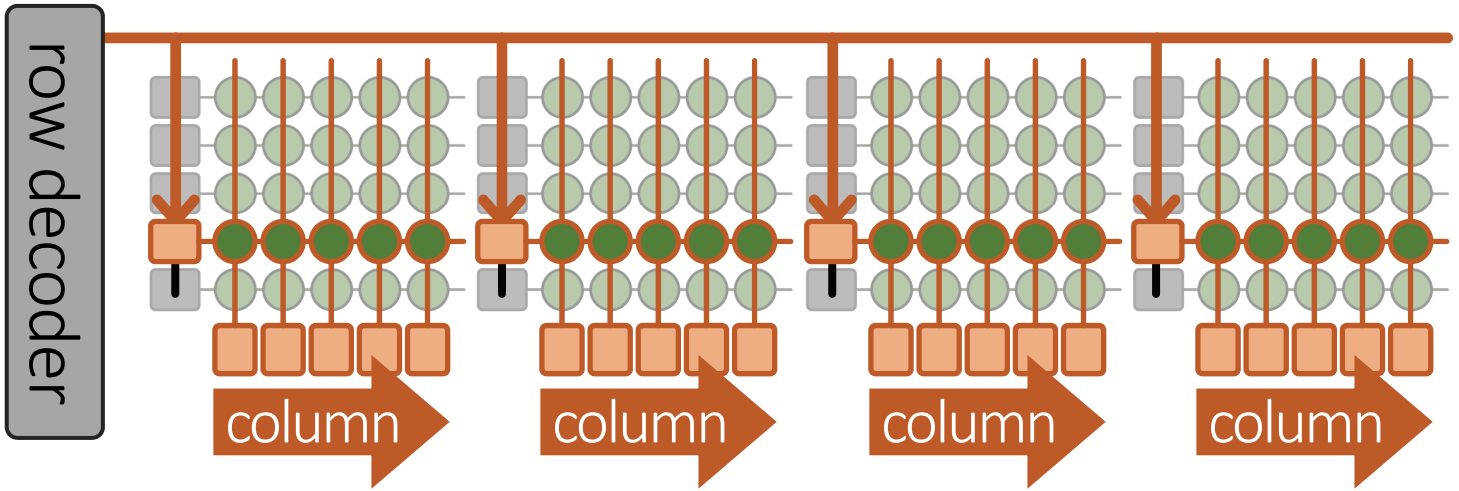
- *Memory controller* sends commands to DRAM



- **Activation**: Open one row in each mat
  - Row decoder, wordline drivers select a row of cells
  - Each cell in the row shares charge with a sense amplifier

DRAM Operations

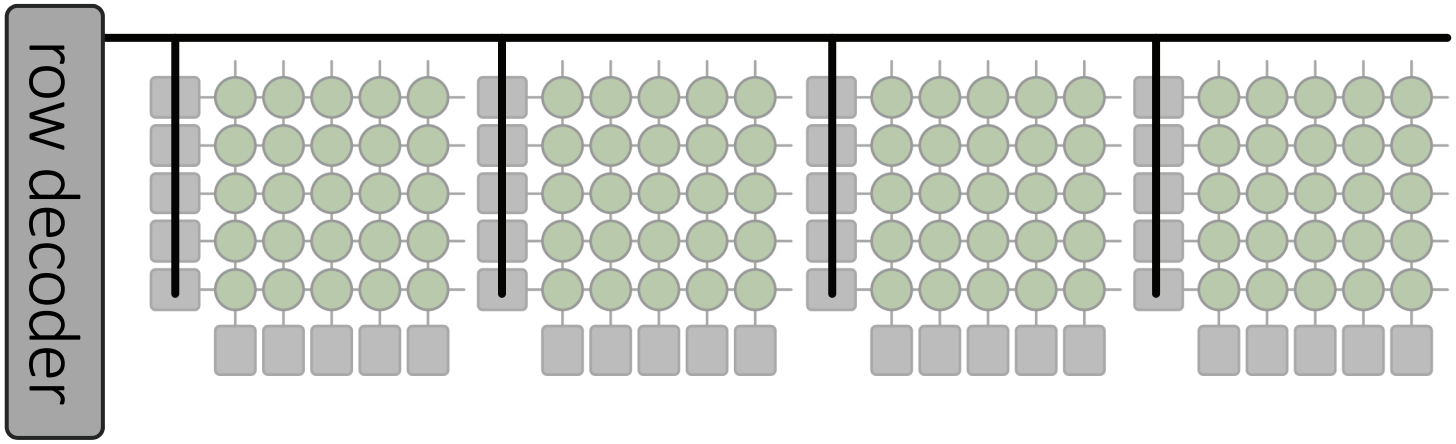
- *Memory controller* sends commands to DRAM



- **Activation**: Open one row in each mat
  - Row decoder, wordline drivers select a row of cells
  - Each cell in the row shares charge with a sense amplifier
- **Read**: Send one column of data from sense amplifiers to CPU

DRAM Operations

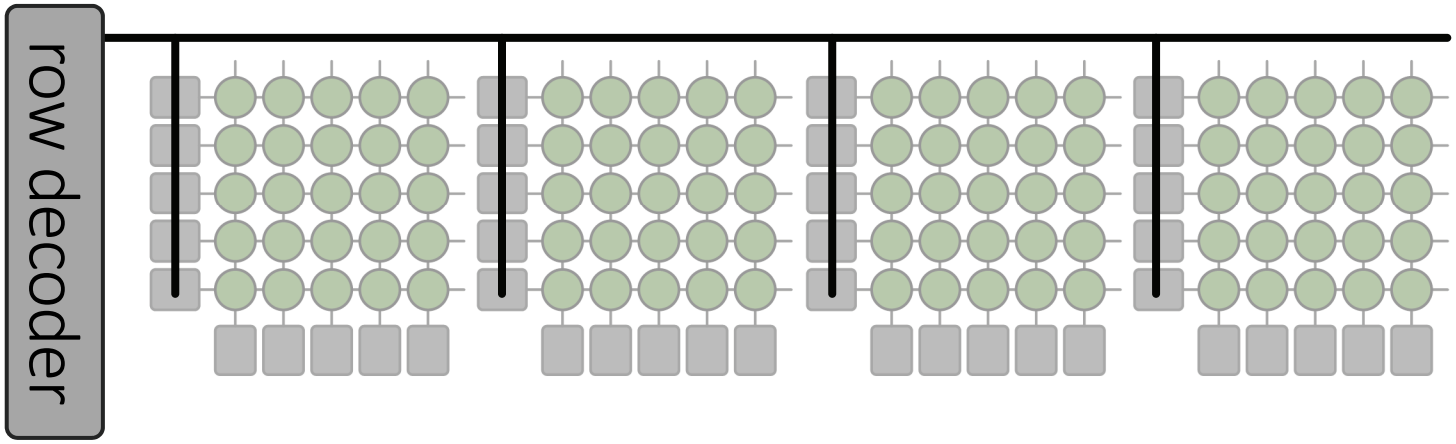
- *Memory controller* sends commands to DRAM



- **Activation**: Open one row in each mat
  - Row decoder, wordline drivers select a row of cells
  - Each cell in the row shares charge with a sense amplifier
- **Read**: Send one column of data from sense amplifiers to CPU
- **Precharge**: Prepare mats for next row

DRAM Operations

- *Memory controller* sends commands to DRAM



- **Activation**: Open one row in each mat
  - Row decoder, wordline drivers select a row of cells
  - Each cell in the row shares charge with a sense amplifier
- **Read**: Send one column of data from sense amplifiers to CPU
- **Precharge**: Prepare mats for next row
- *Timing parameters*: how long to wait for each step to finish

DRAM Timing Parameters

- Standard timing parameters are dictated by ***the worst case***
- Must ensure reliable operation for:
  - The smallest cell with the smallest charge in ***all DRAM modules*** (*process variation*)
  - The ***highest*** operating temperature allowed (*charge leakage*)
- ***Large timing margin*** for the common case

→ **Goal: Lower common-case latency**

DRAM Timing Parameters

- Standard timing parameters are dictated by ***the worst case***
- Must ensure reliable operation for:
  - The smallest cell with the smallest charge in ***all DRAM modules*** (*process variation*)
  - The ***highest*** operating temperature allowed (*charge leakage*)

Can we use *design-induced variation* to **find and use common-case latency at low cost**?

Presentation Outline

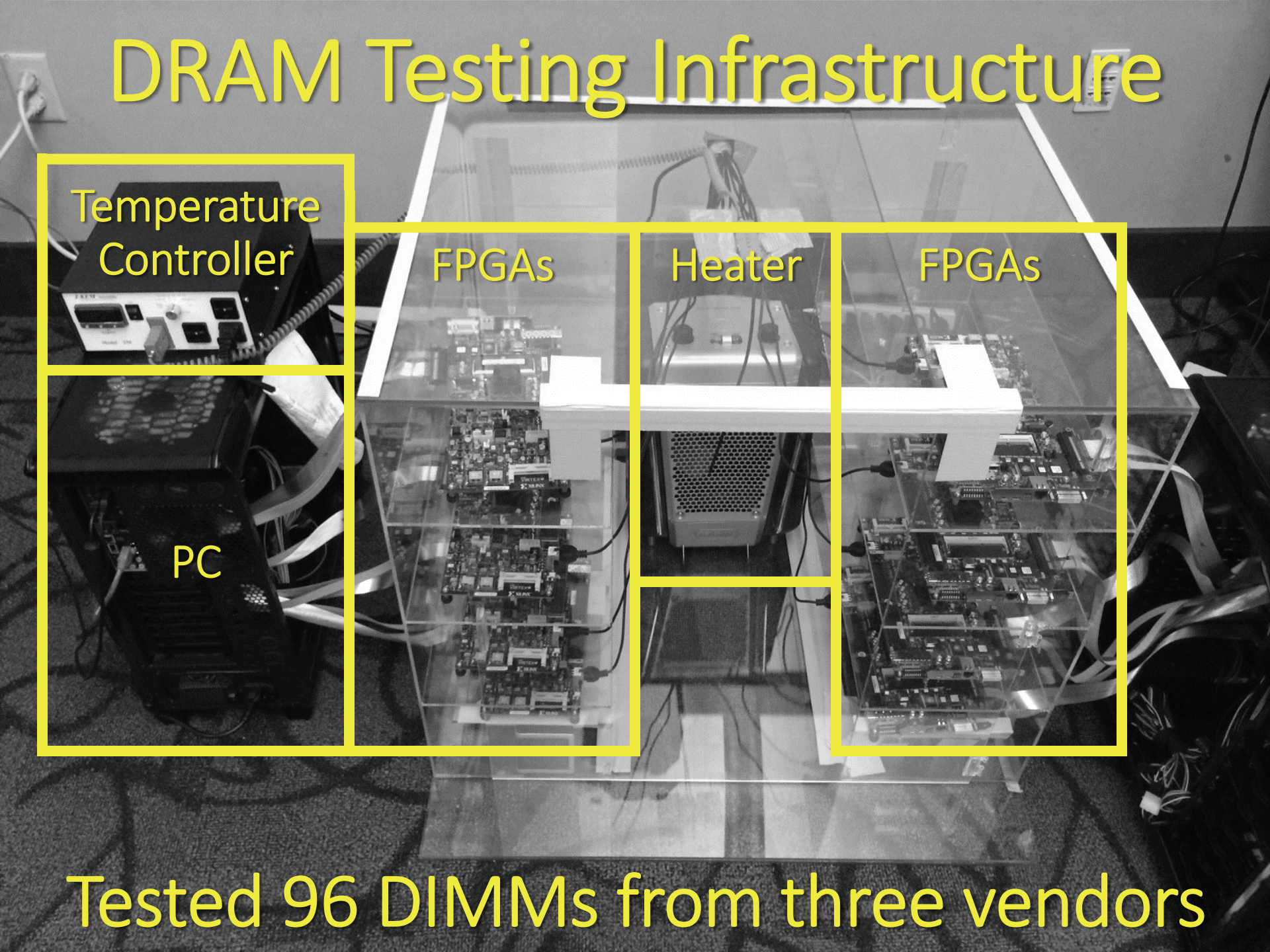
- DRAM Background
- Experimental Study of Design-Induced Variation
  - **Goal:** Understand, identify inherently slower regions
  - **Methodology:** Profile 96 real DRAM modules by using FPGA-based DRAM test infrastructure
- Exploiting Design-Induced Variation
  - **DIVA Profiling**: Using low-cost slow region profiling to reliably and dynamically reduce DRAM latency
  - **DIVA Shuffling**: Using variation across data bursts to reduce uncorrectable errors

Expected DRAM Characteristics

- Variation
  - Some regions are **slower** than others
  - If we reduce DRAM latency, slower regions are **more vulnerable** to errors
- Repeatability
  - Latency (error) characteristics **repeat periodically**, if the same component (e.g., mat) is duplicated
- Similarity
  - Characteristics repeat across different organizations (e.g., chip/DIMM) that share same design

Characterization Methodology

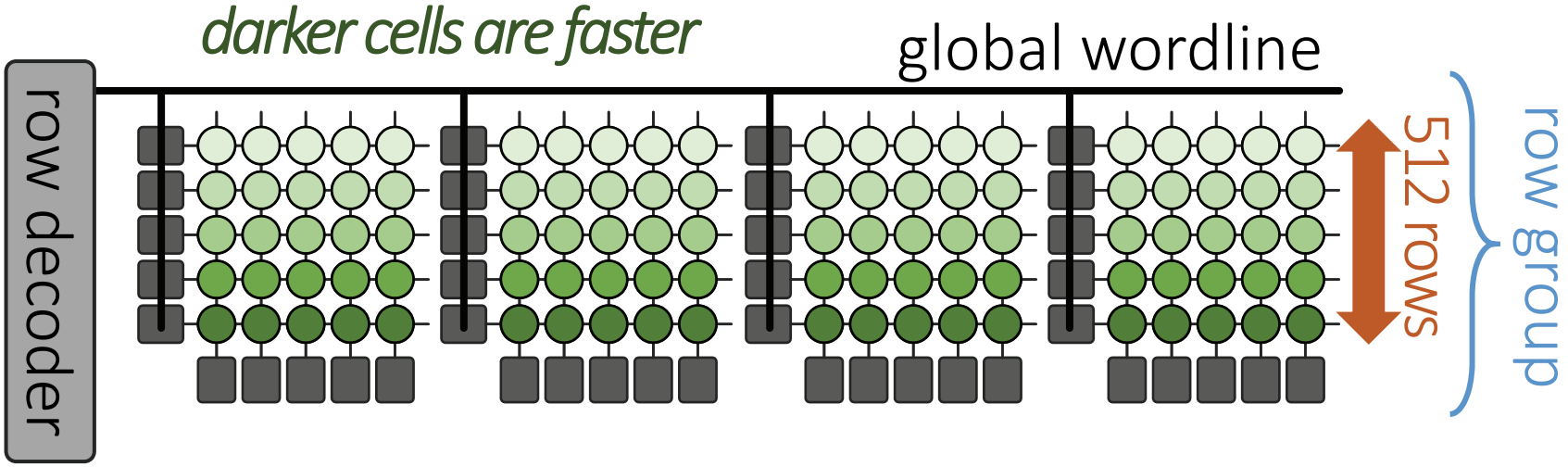
- We use **error behavior when we reduce latency** to infer DRAM organization and characteristics
  - FPGA-based memory controller infrastructure


DRAM Testing Infrastructure
Temperature Controller
PC
FPGAs
Heater
FPGAs
Tested 96 DIMMs from three vendors

Characterization Methodology

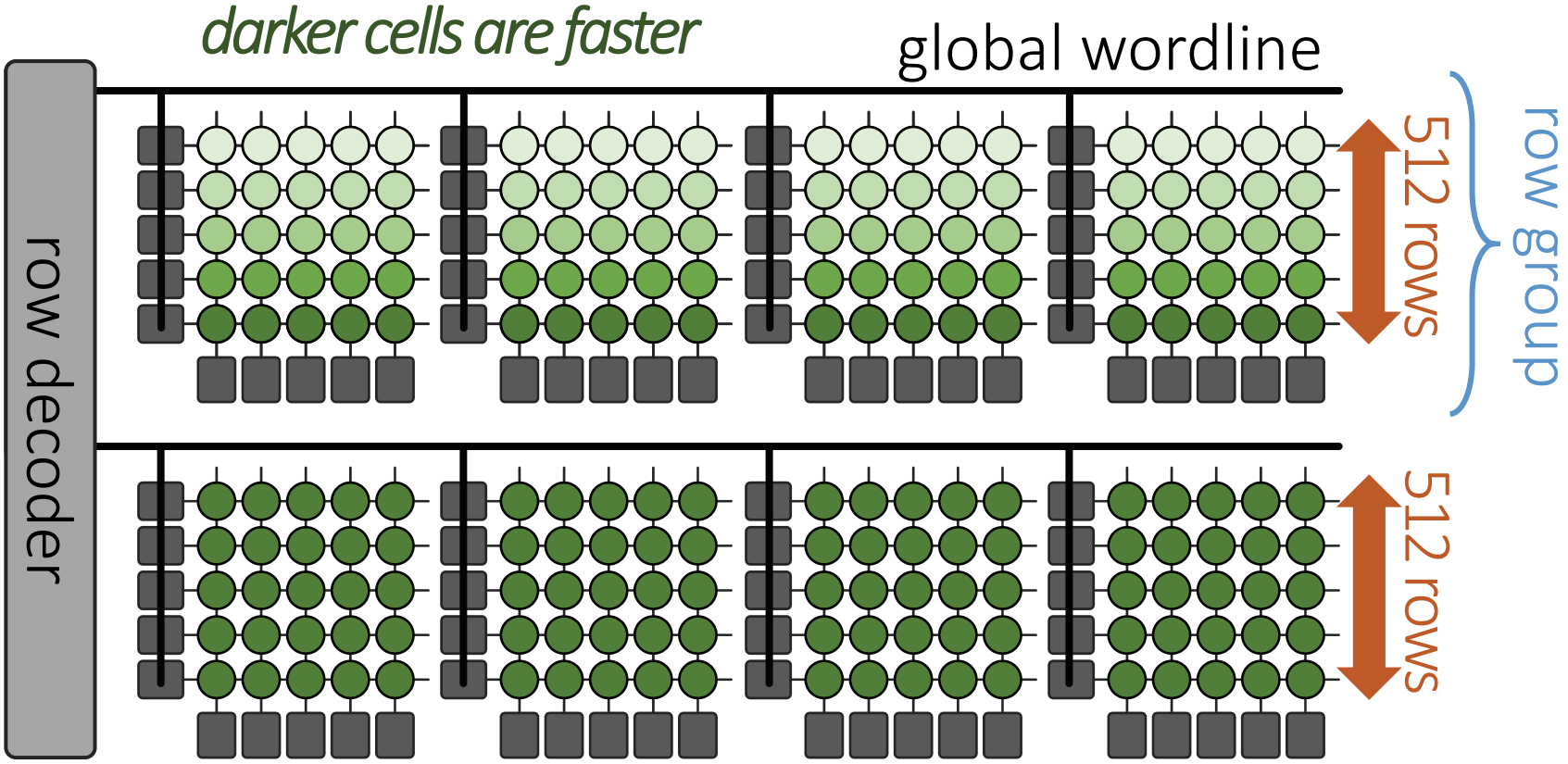
- We use **error behavior when we reduce latency** to infer DRAM organization and characteristics
  - FPGA-based memory controller infrastructure
  - We reverse engineer row address mapping
  - Lower tRP (precharge timing parameter) to 7.5 ns
- We characterize three types of variation
  - Variation across columns
  - Variation across rows
  - Variation across data bursts
- Data and circuit model will be available on GitHub: https://github.com/CMU-SAFARI/DIVA-DRA

1. Variation Across Rows



Latency characteristics ***vary*** across 512 rows

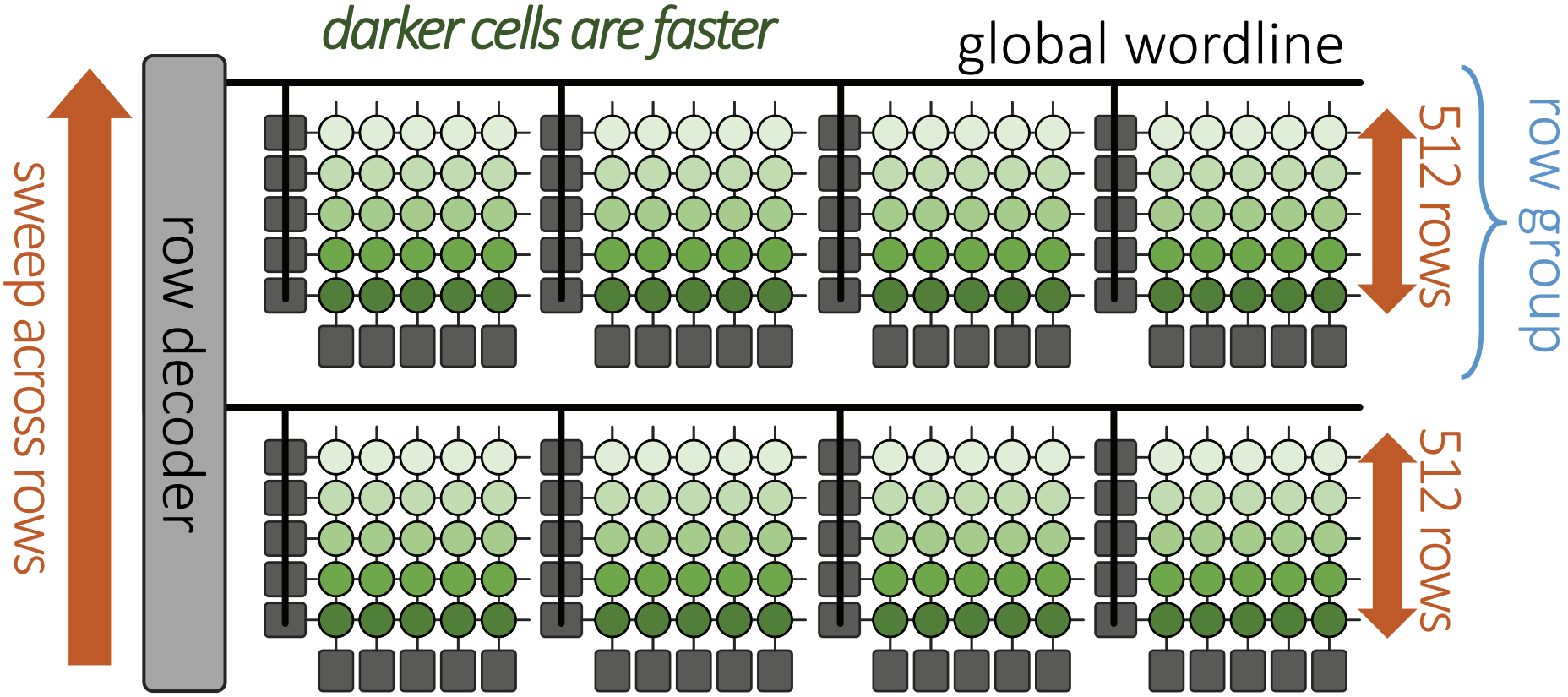
1. Variation Across Rows



Latency characteristics *vary* across 512 rows

Same organization repeats every 512 rows

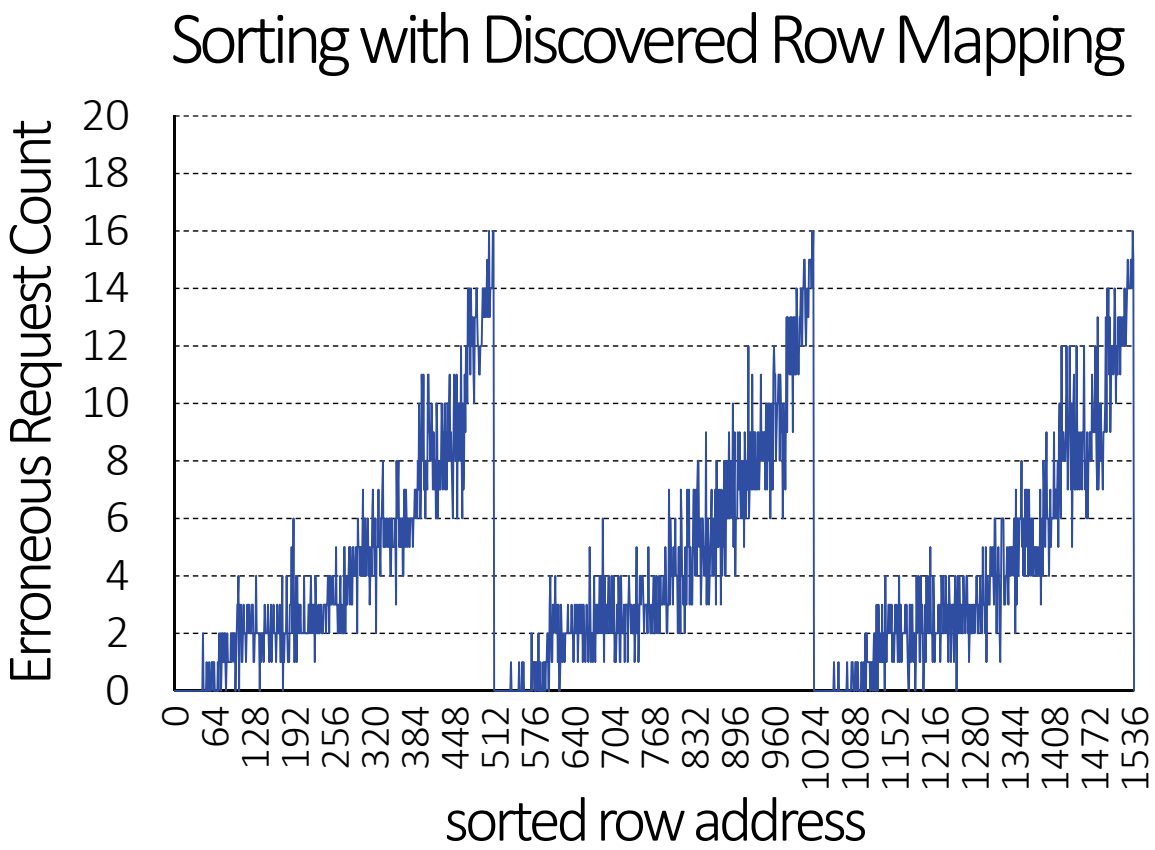
1. Variation Across Rows



Latency characteristics ***vary*** across 512 rows

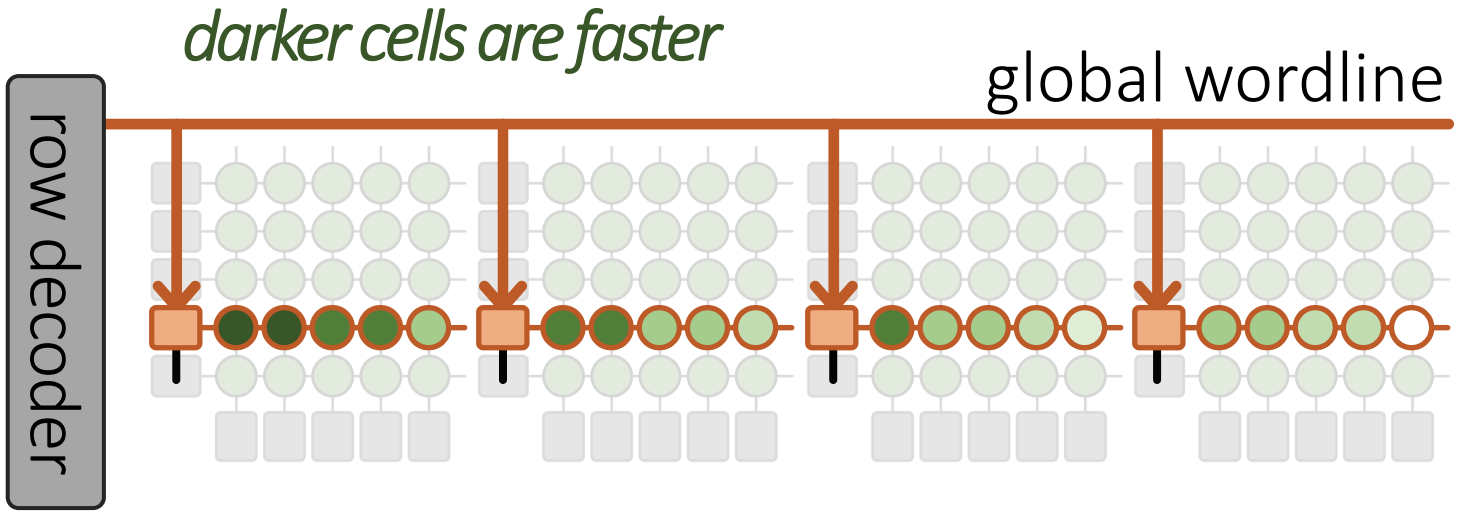
Latency characteristics ***repeat*** every 512 rows

1.2. Periodic Row Variation Behavior



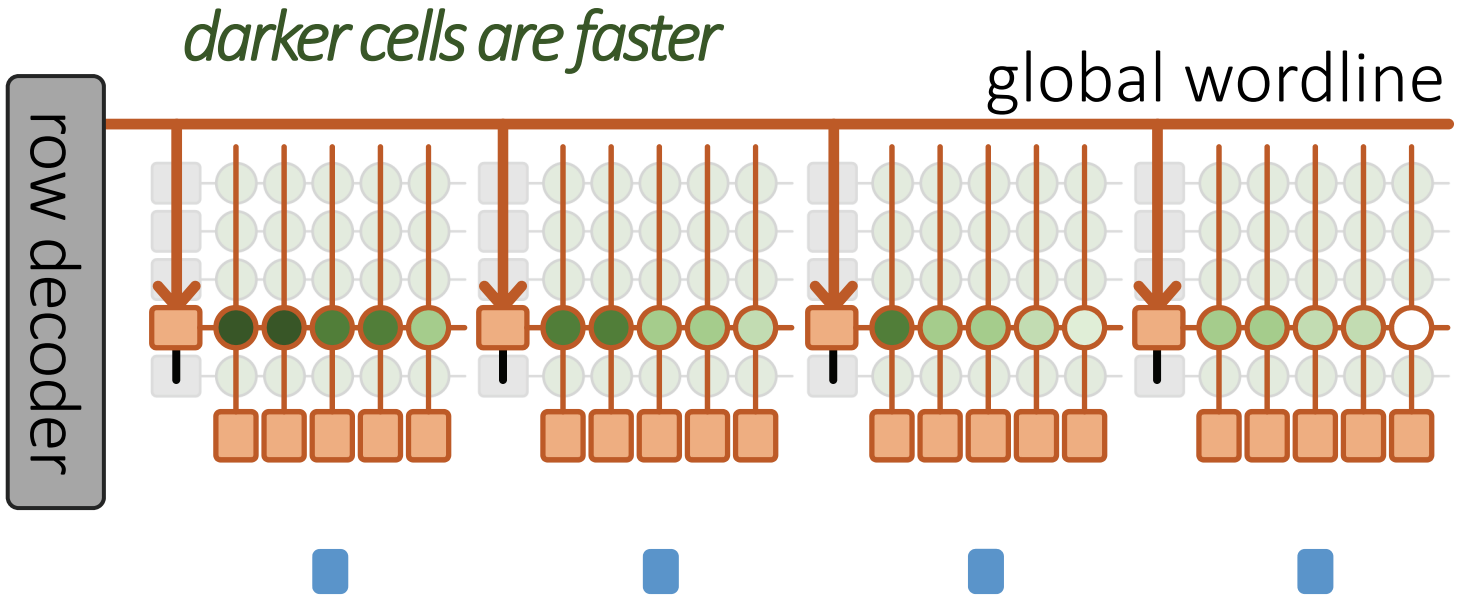
Row error (latency) characteristics
periodically repeat every 512 rows

2. Variation Across Columns



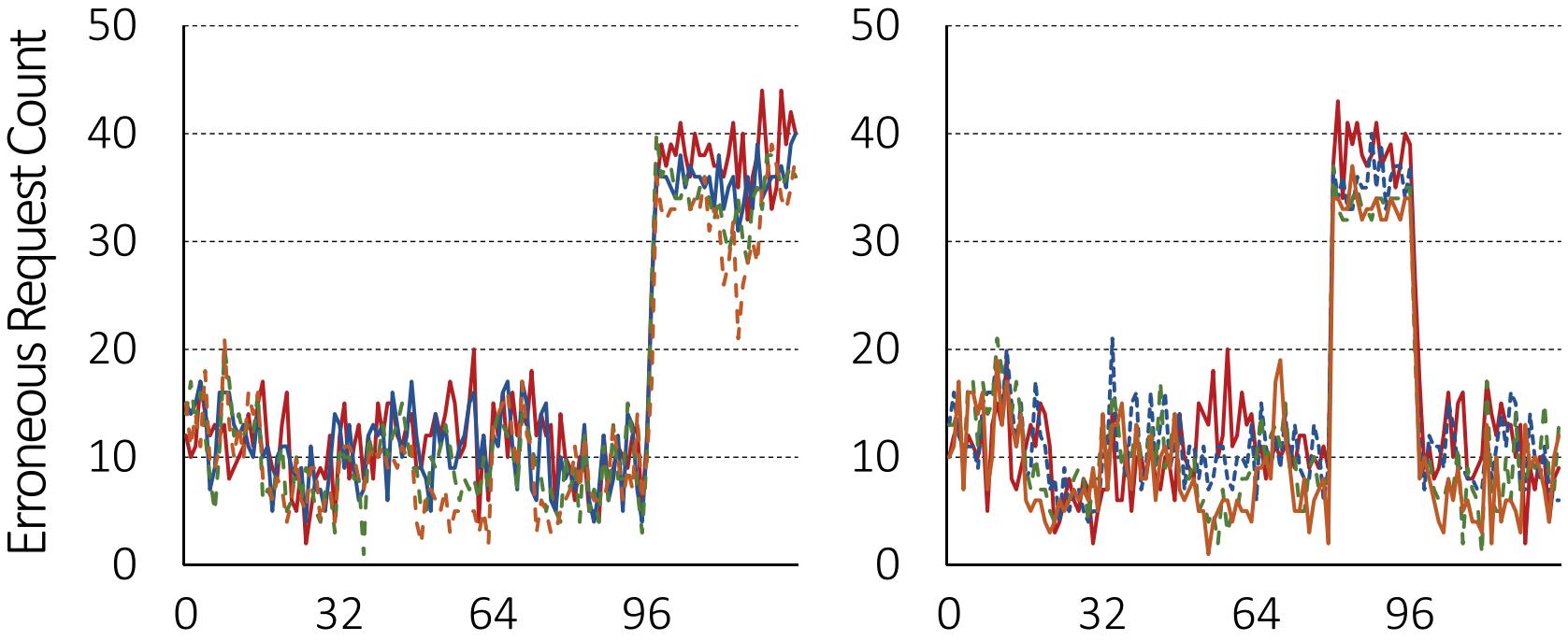
Column latency depends on
distance from row decoder, wordline driver

2. Variation Across Columns



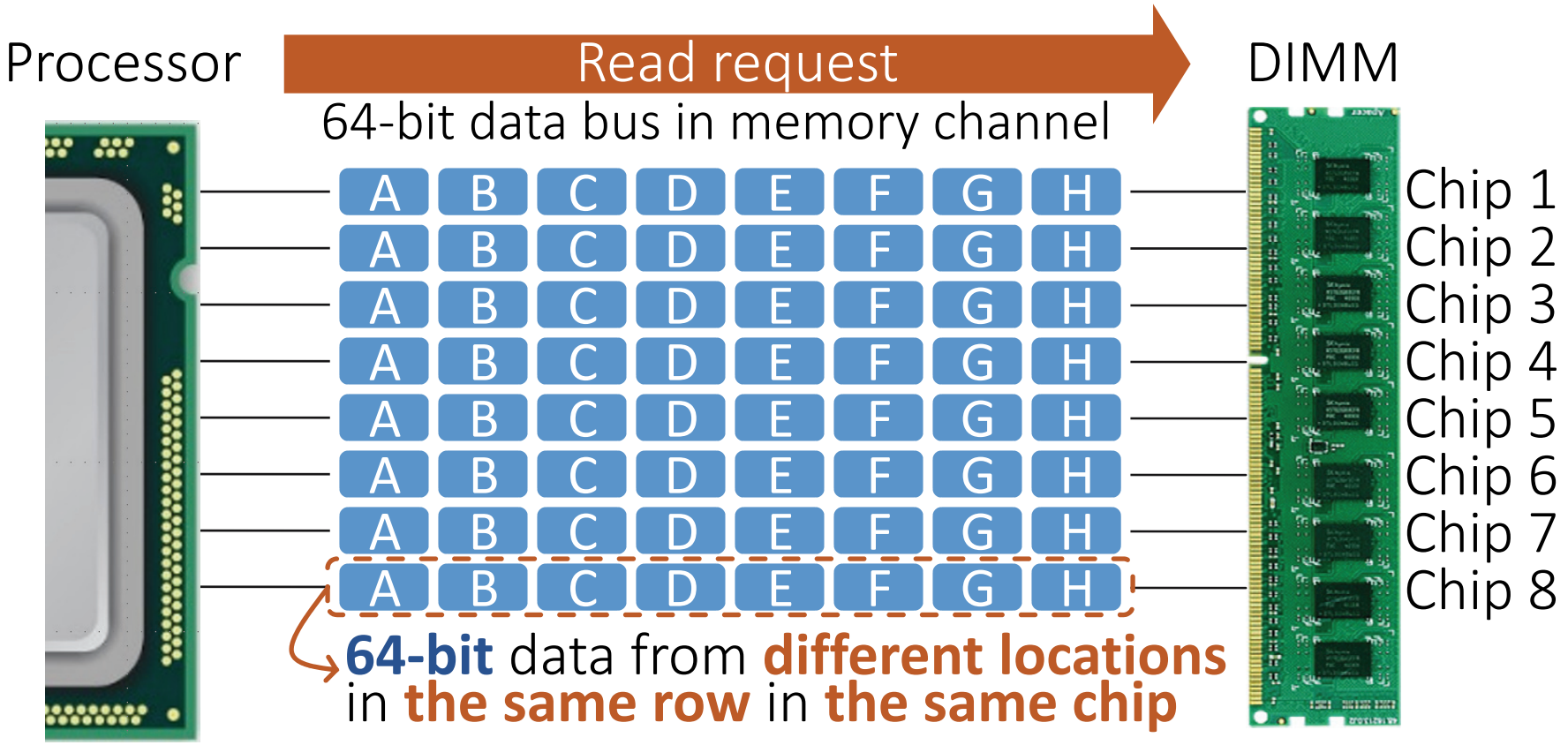
Column latency depends on
distance from row decoder, wordline driver

2. Variation Across Columns

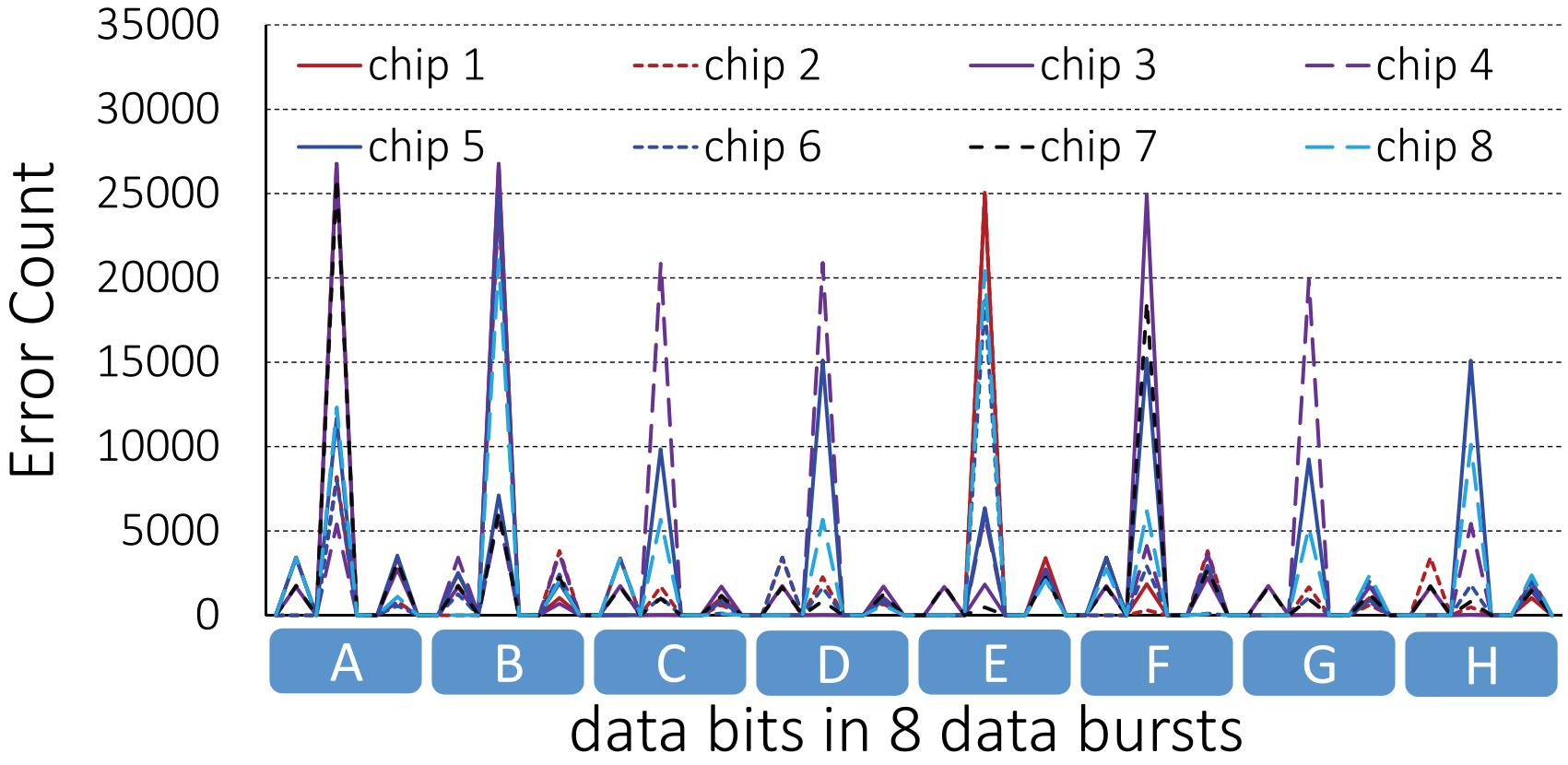


Column error (latency) characteristics have ***specific patterns*** that repeat across row groups

3. Variation Across Data Bursts



3. Variation Across Data Bursts



Specific bits in a request ***induce more errors***

Summary: Design-Induced Variation

- **Systematic variation across rows**
  - Slow cells further from sense amplifier
- **Systematic variation across columns**
  - Slow cells further from row decoder
  - Slow cells further from wordline driver
- **Systematic variation across data bursts**
  - Slow cells at certain bits in a burst
  - Clustering of errors

Can we use *design-induced variation*
to **find and use common-case latency at low cost**?

Presentation Outline

- DRAM Background
- Experimental Study of Design-Induced Variation
  - Goal: Understand, identify inherently slower regions
  - Methodology: Profile 96 real DRAM modules by using FPGA-based DRAM test infrastructure
- Exploiting Design-Induced Variation
  - DIVA Profiling: Using low-cost slow region profiling to reliably and dynamically reduce DRAM latency
  - DIVA Shuffling: Using variation across data bursts to reduce uncorrectable errors

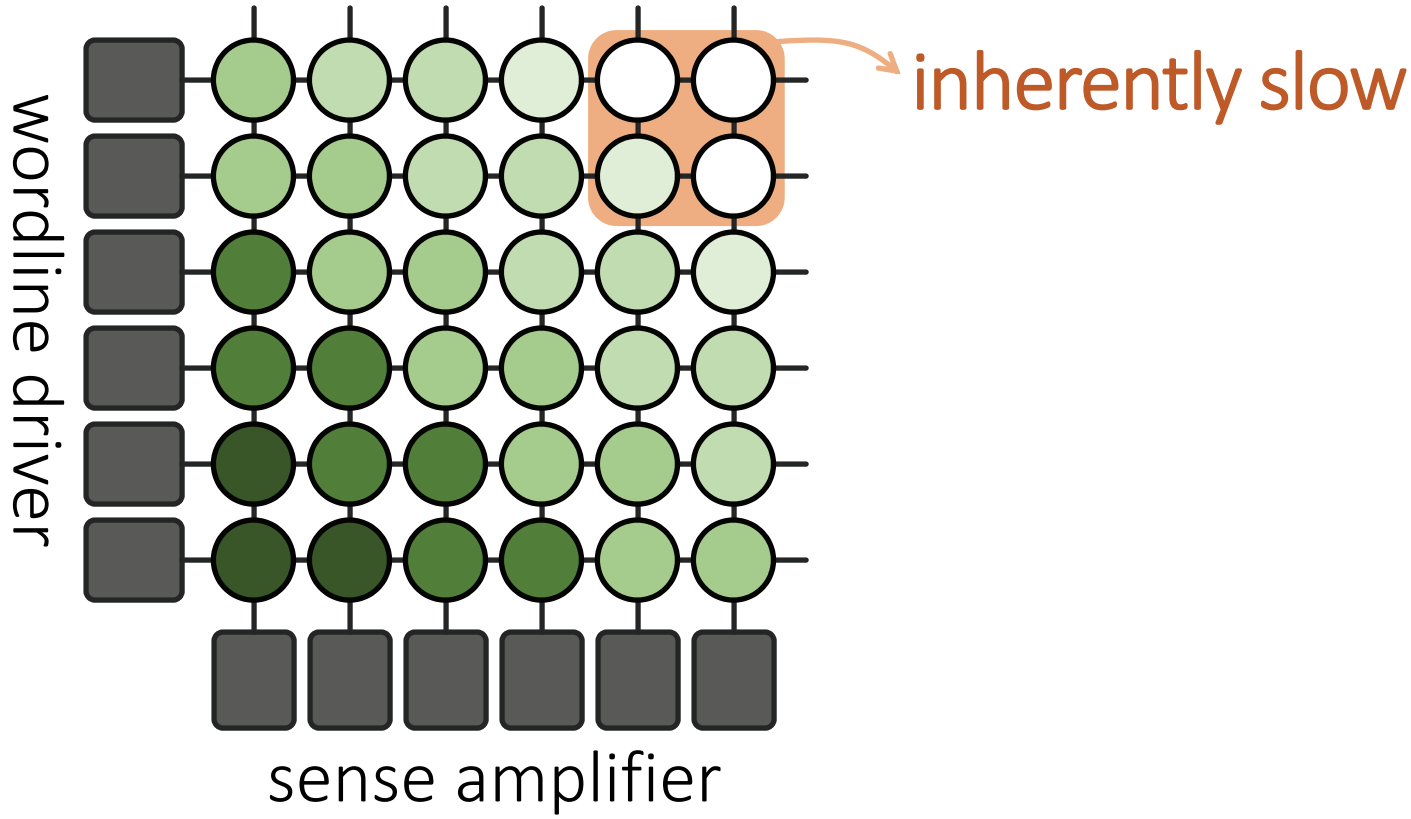
Challenges of Lowering Latency

- **Static DRAM latency** (e.g., AL-DRAM [HPCA 2015])
  - DRAM vendors need to provide ***fixed*** timings, *increasing testing costs*
  - Doesn't account for *latency changes* over time (e.g., aging and wear out)
- **Conventional online profiling**
  - Takes long time (**high cost**) to profile ***all*** DRAM cells

Our Goal:
Use design-induced variation to ***minimize profiling***

1. DIVA Profiling

Design-Induced-Variation-Aware



Profile *only slow regions* to determine latency

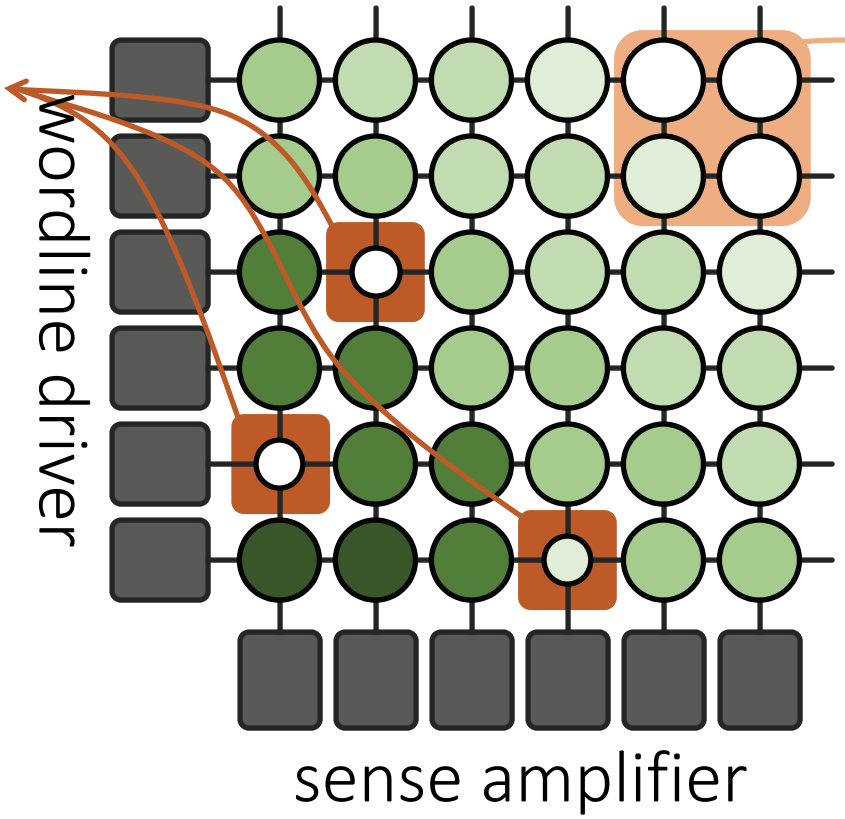
What About Process Variation?

Design-Induced-Variation-Aware

slow cells

process variation

random error



inherently slow

architectural variation

localized error

What About Process Variation?

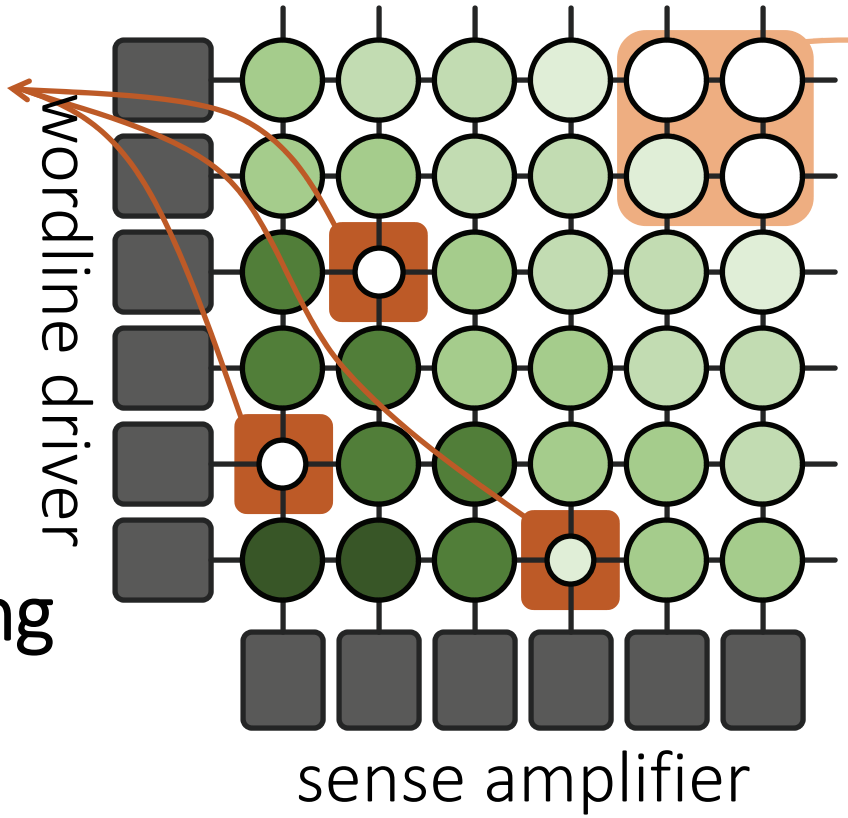
Design-Induced-Variation-Aware

slow cells

process variation

random error

error-correcting codes (ECC)



inherently slow

architectural variation

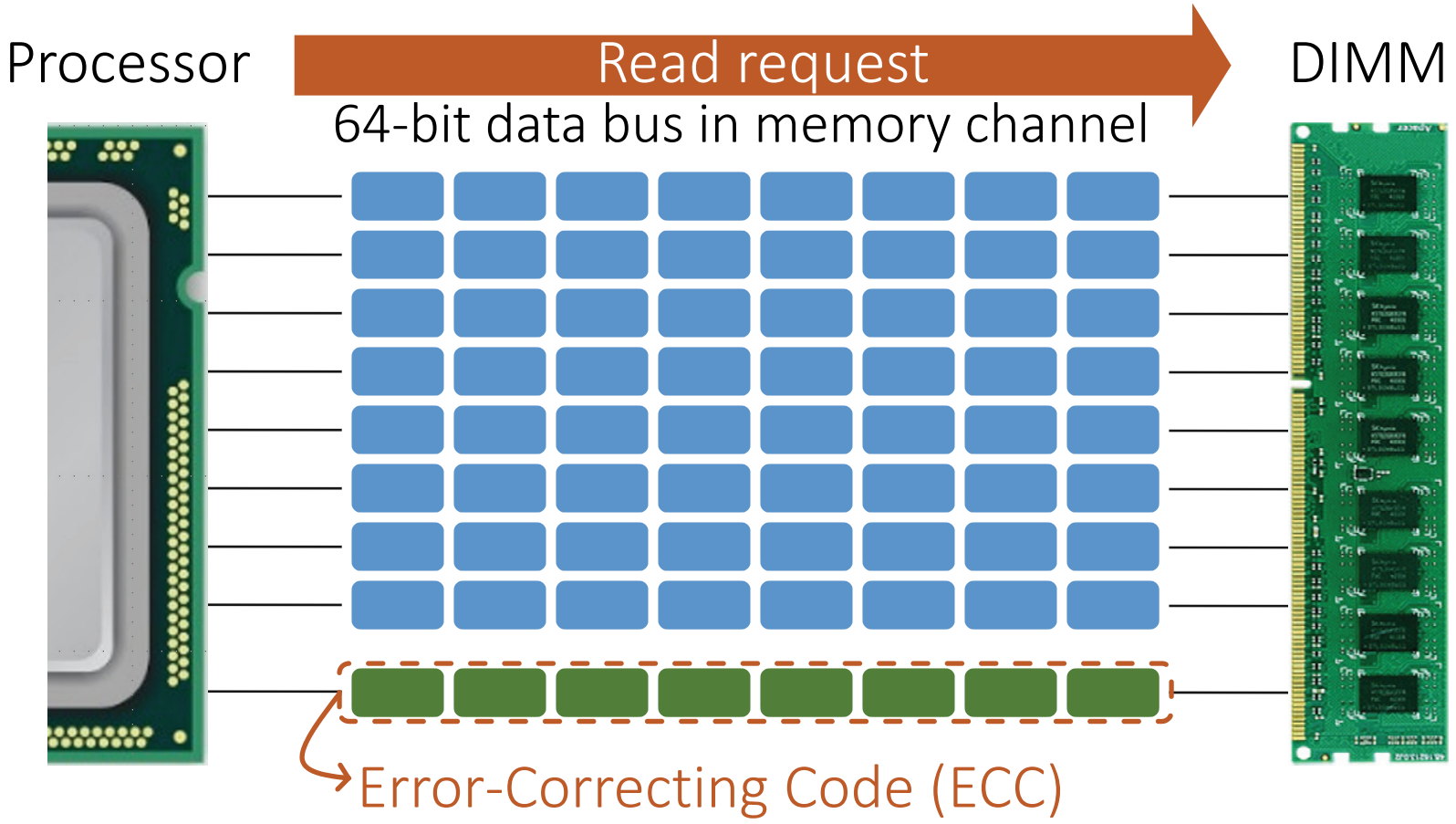
localized error

online profiling

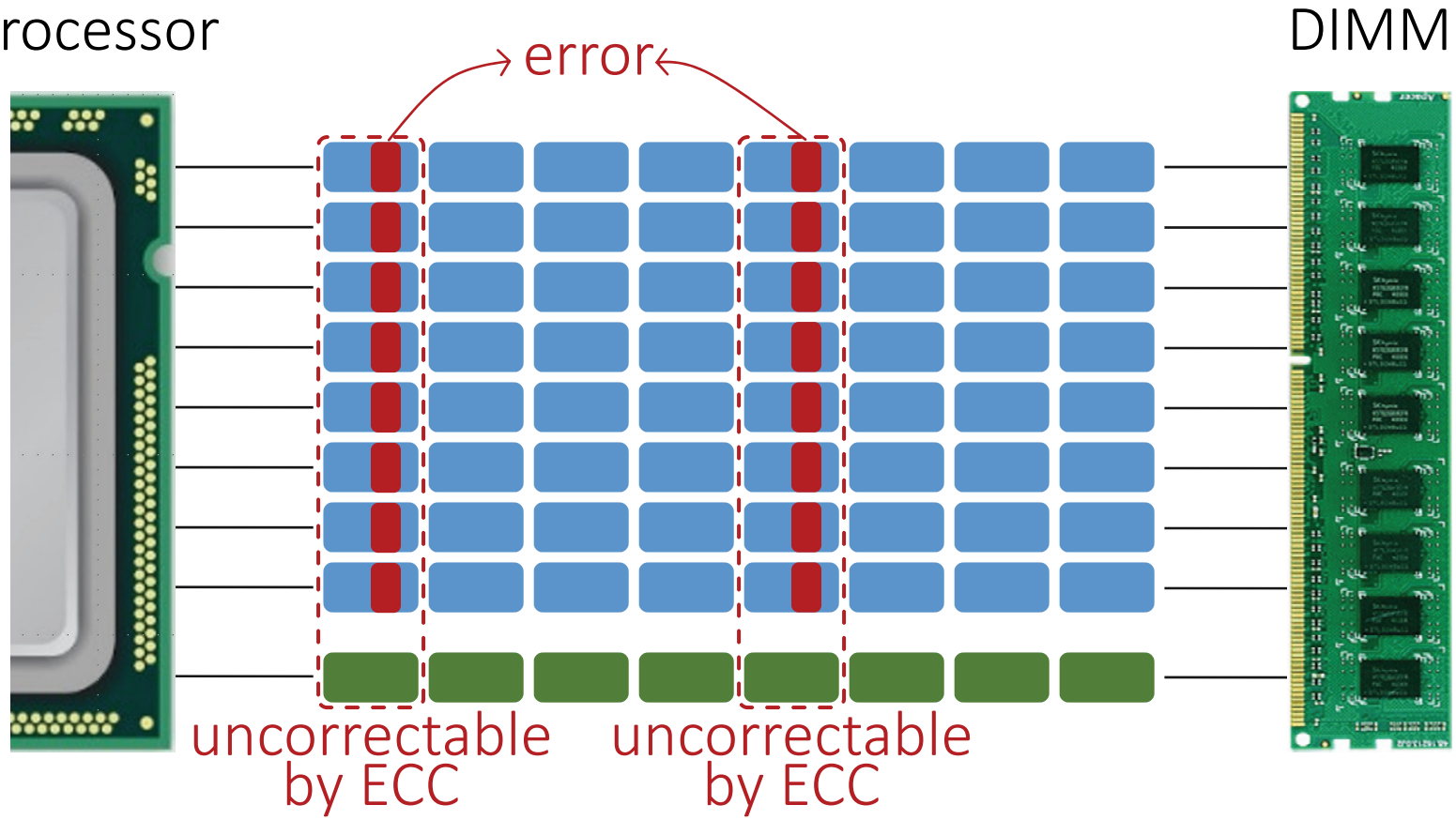
Combine ECC & online profiling

→ ***Reliably*** reduce DRAM latency ***at low cost***

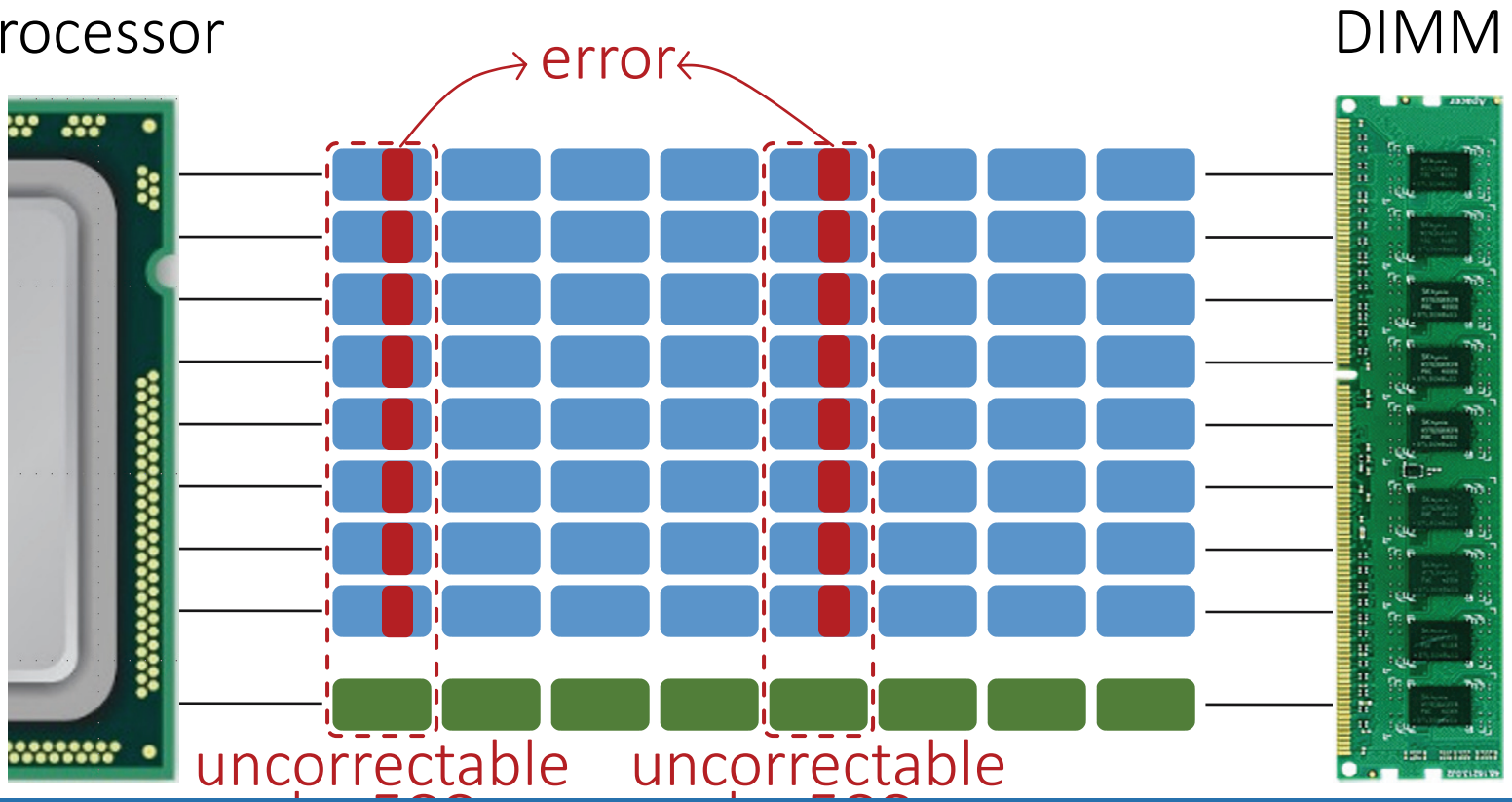
Correction with Conventional ECC



Challenge of Conventional ECC



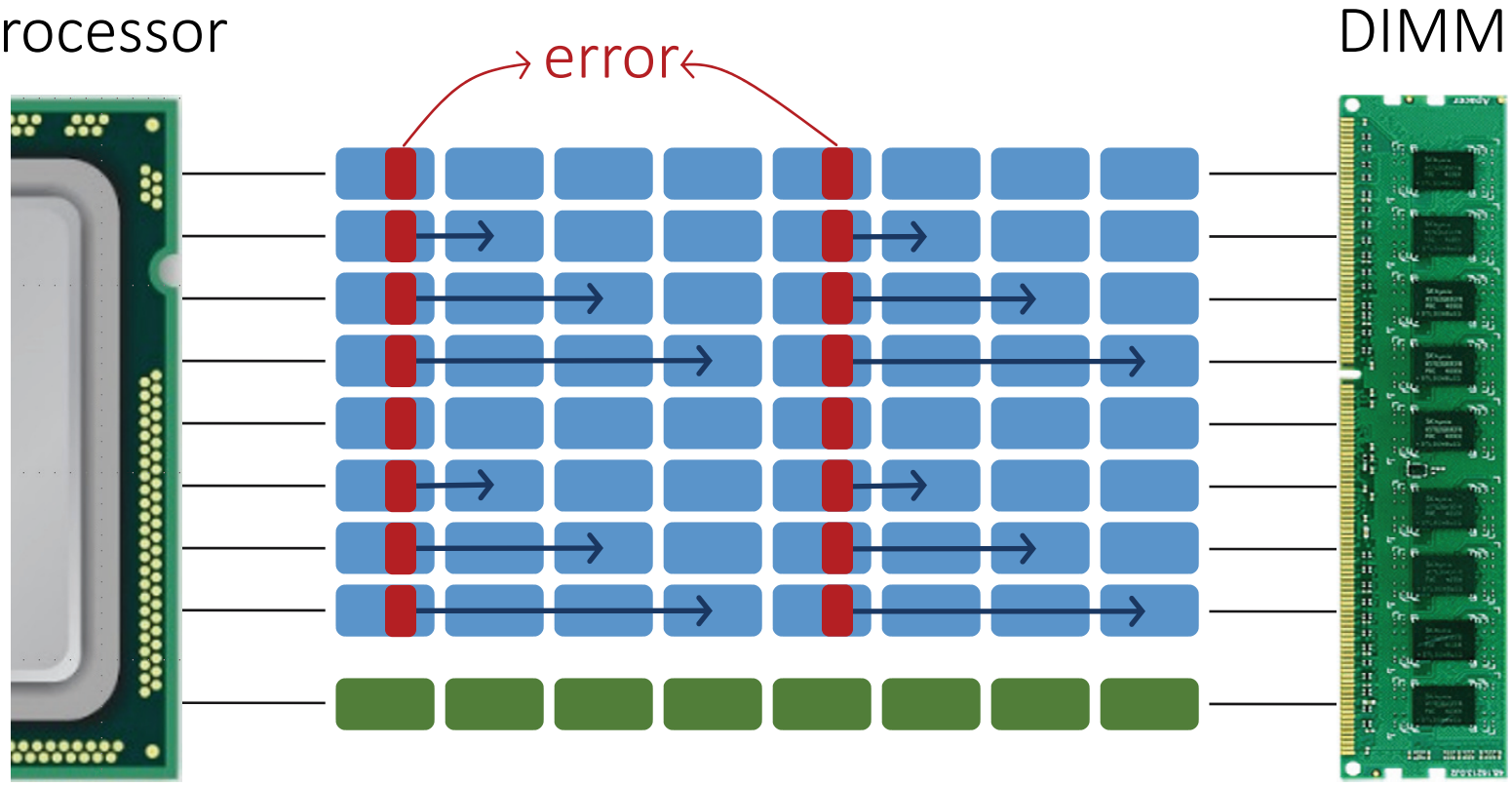
Challenge of Conventional ECC



Clusters of slow cells due to design-induced variation lead to ***more uncorrectable errors***

2. DIVA Shuffling

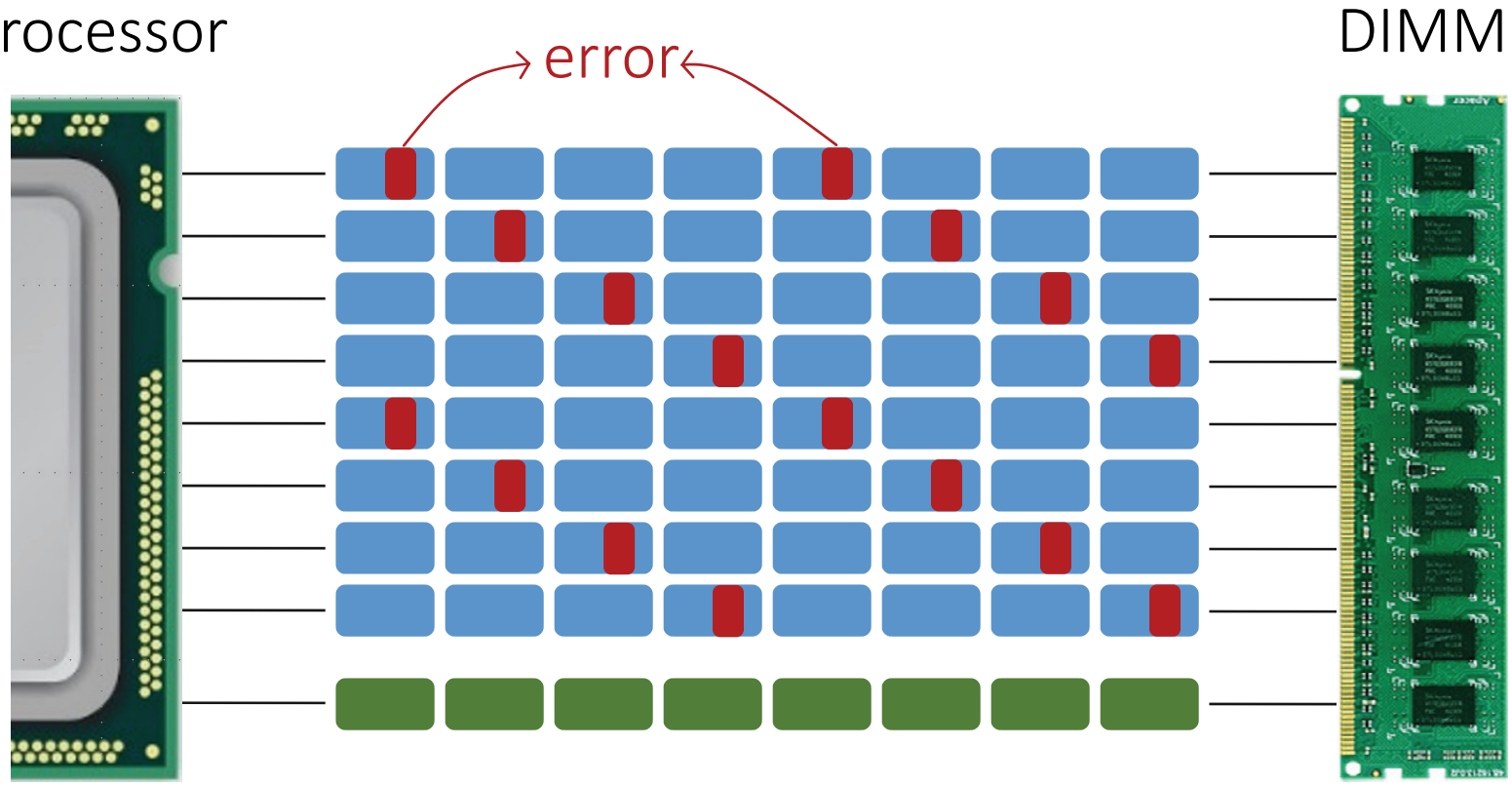
Design-Induced-**V**ariation-**A**ware



Shuffle data bursts

2. **DIVA Shuffling**

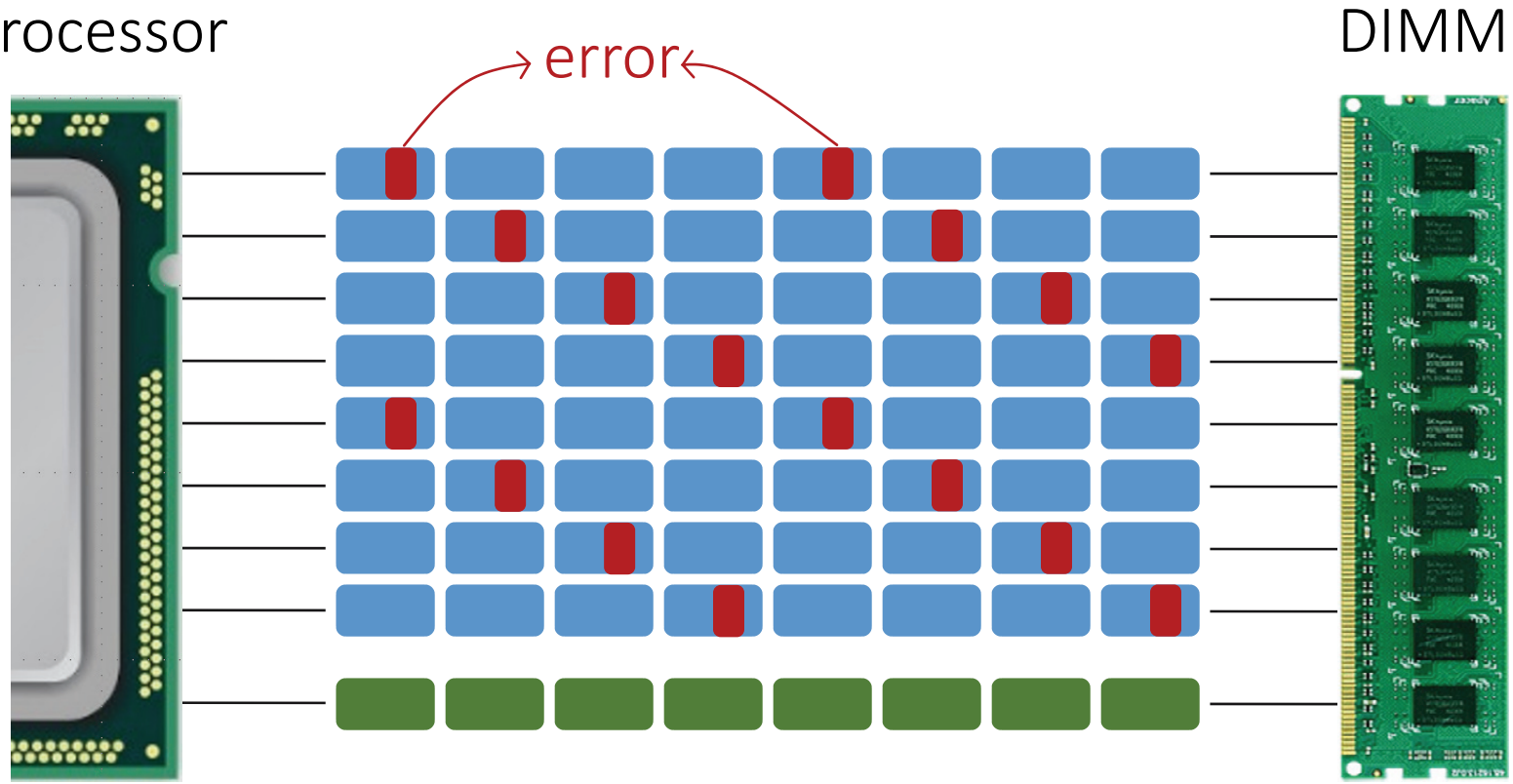
Design-Induced-**V**ariation-**A**ware



Shuffle data bursts → *Reduce uncorrectable errors*

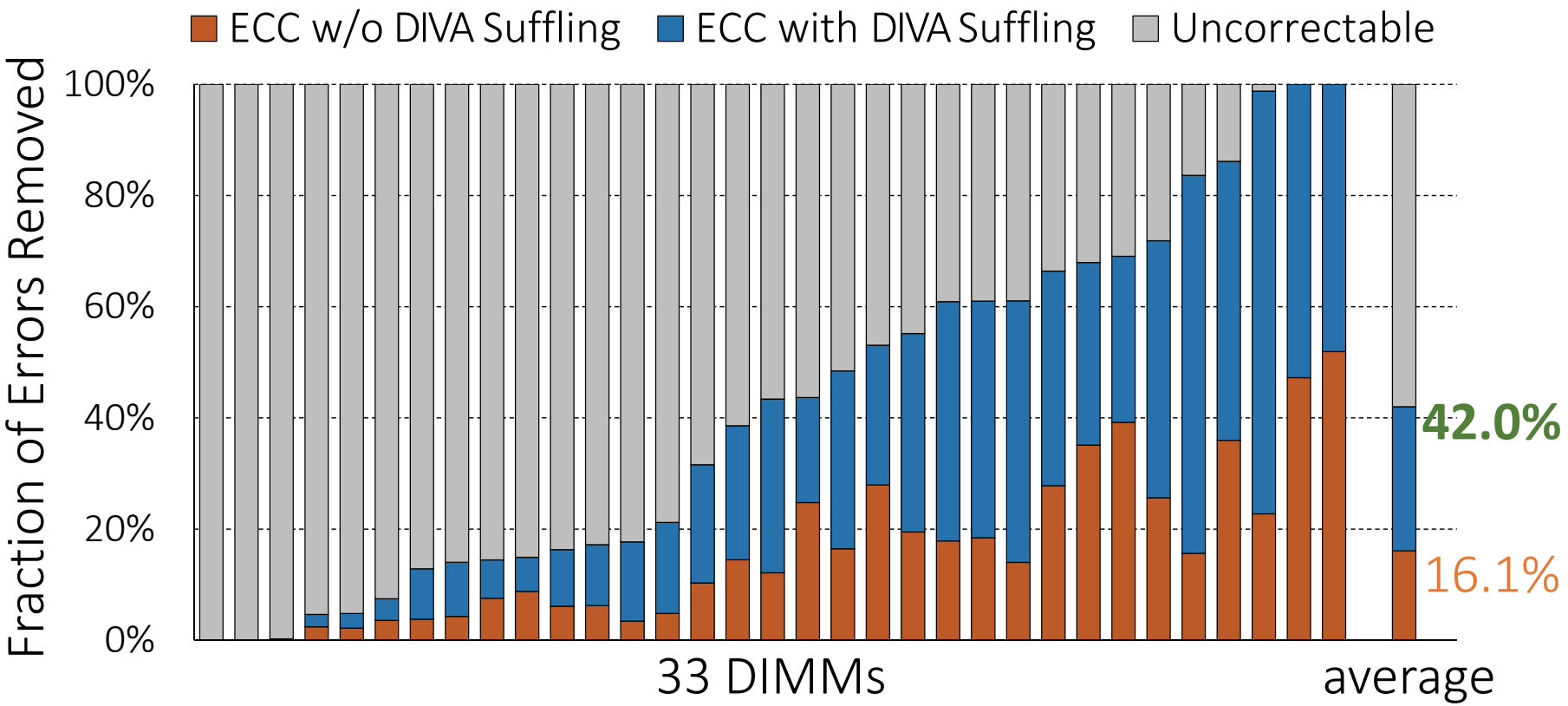
2. DIVA Shuffling

Design-Induced-**V**ariation-**A**ware



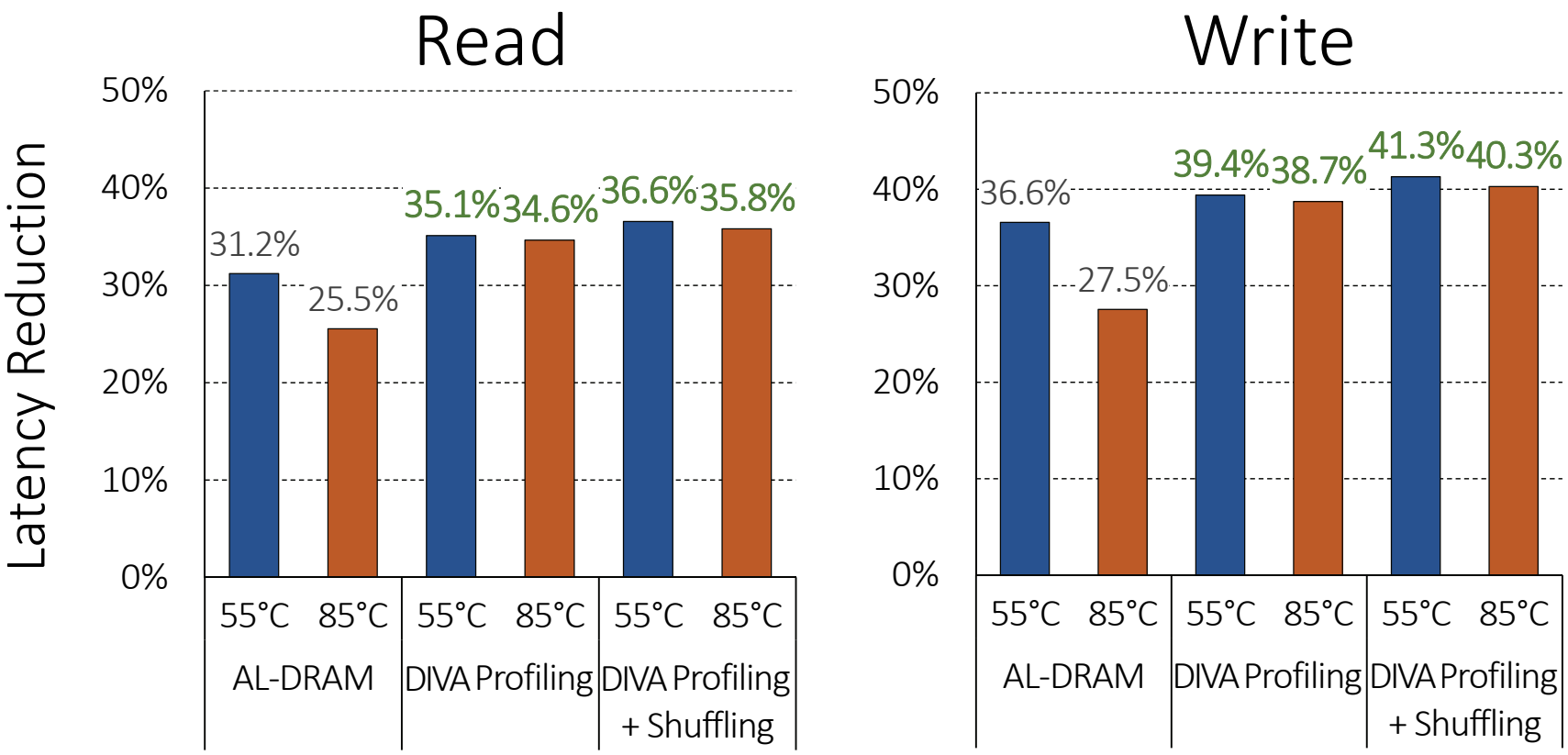
How do DIVA Profiling and DIVA Shuffling perform?

DIVA Shuffling Improves ECC

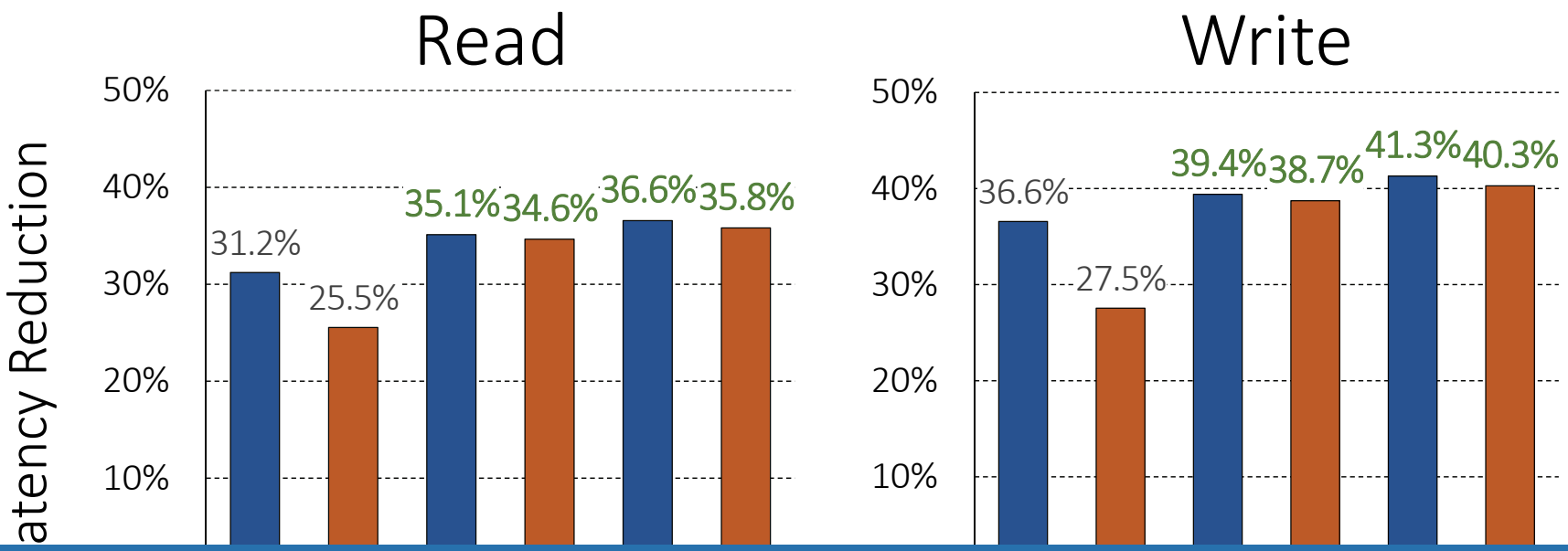


DIVA Shuffling uses architectural variation to ***improve error correction*** using the same codeword

DIVA-DRAM Reduces Latency



DIVA-DRAM Reduces Latency

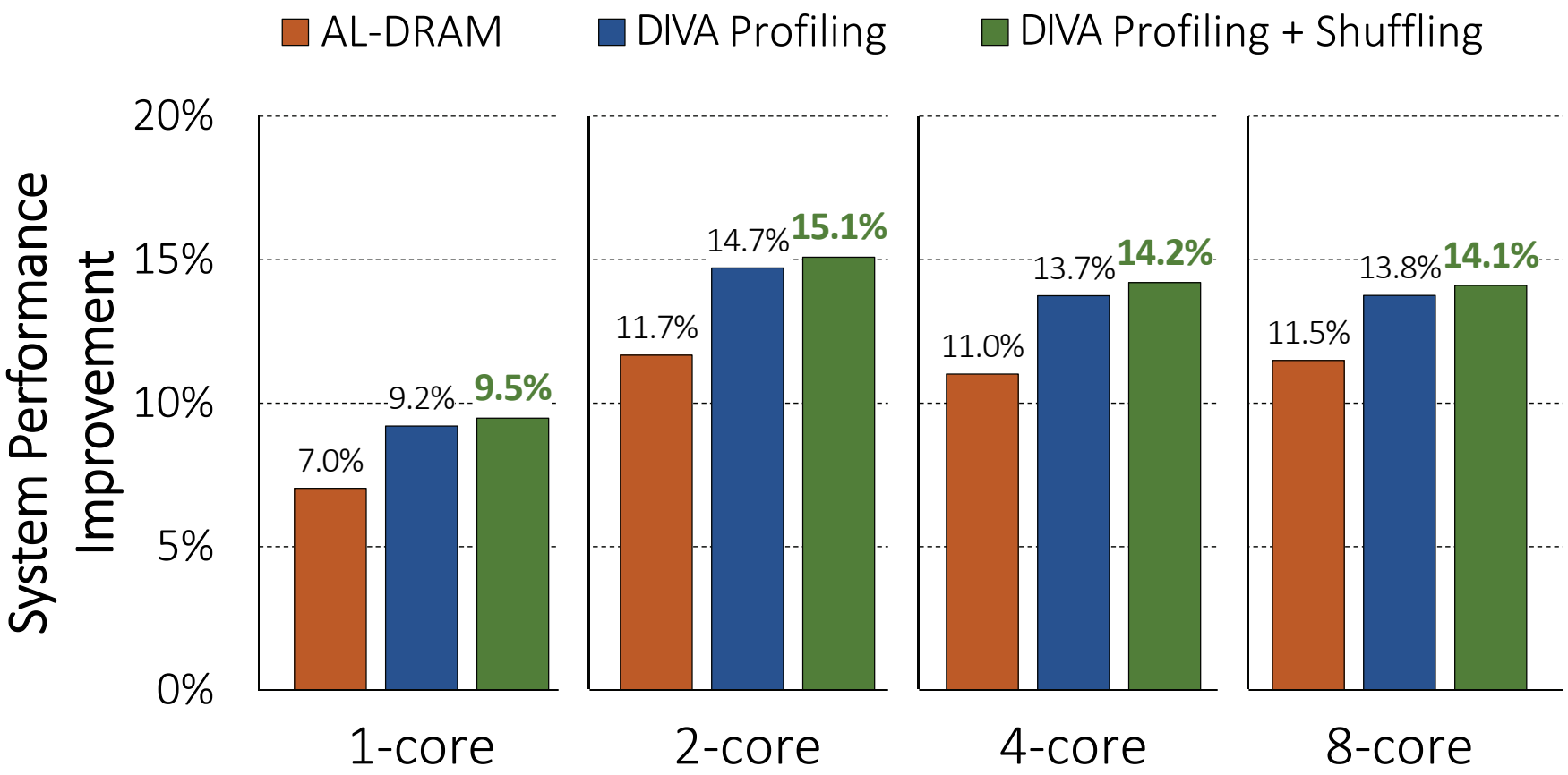


DIVA-DRAM ***reduces latency more aggressively*** and uses ECC to correct random slow cells

How do DRAM latency reductions translate to system performance?

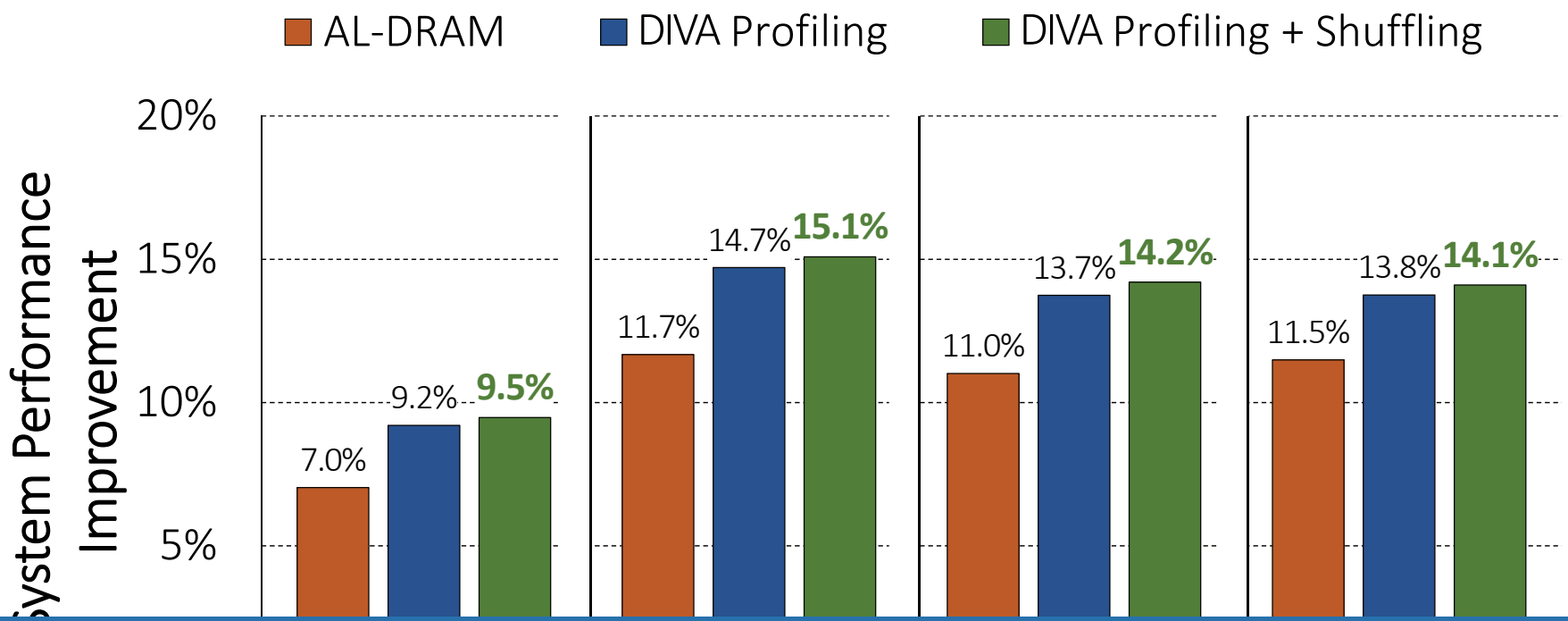
DIVA-DRAM Improves Performance

- 32 single-core benchmarks: SPEC, Stream, TPC, GUPS
- 96 multicore workloads constructed with benchmarks



DIVA-DRAM Improves Performance

- 32 single-core benchmarks: SPEC, Stream, TPC, GUPS
- 96 multicore workloads constructed with benchmarks



DIVA-DRAM outperforms the best prior work
and can adapt to dynamic latency changes

Conclusion

- **Design-Induced Variation**: **Inherently slow regions** due to DRAM cell array organization

- Analysis: Characterization of **96 real DRAM modules**
  - Three types of *systematic* variation due to design
  - Great ***potential*** to ***reduce DRAM latency*** at low cost

- Our Approach: **DIVA-DRAM**
  - **DIVA Profiling**: Reliably reduce DRAM latency
    - Profile *only* cells in *inherently slow regions*
    - Use error correction (ECC) for slow cells that are not profiled
  - **DIVA Shuffling**: Exploit variation to improve ECC reliability

- **15.1%/14.2% *higher performance*** for 2-/8-core workloads

Design-Induced Latency Variation in Modern DRAM Chips: Characterization, Analysis, and Latency Reduction Mechanisms

Donghyuk Lee[1,2] Samira Khan[3] Lavanya Subramanian[2]
Saugata Ghose[2] Rachata Ausavarungnirun[2]
Gennady Pekhimenko[4] Vivek Seshadri[4] Onur Mutlu[5,2]

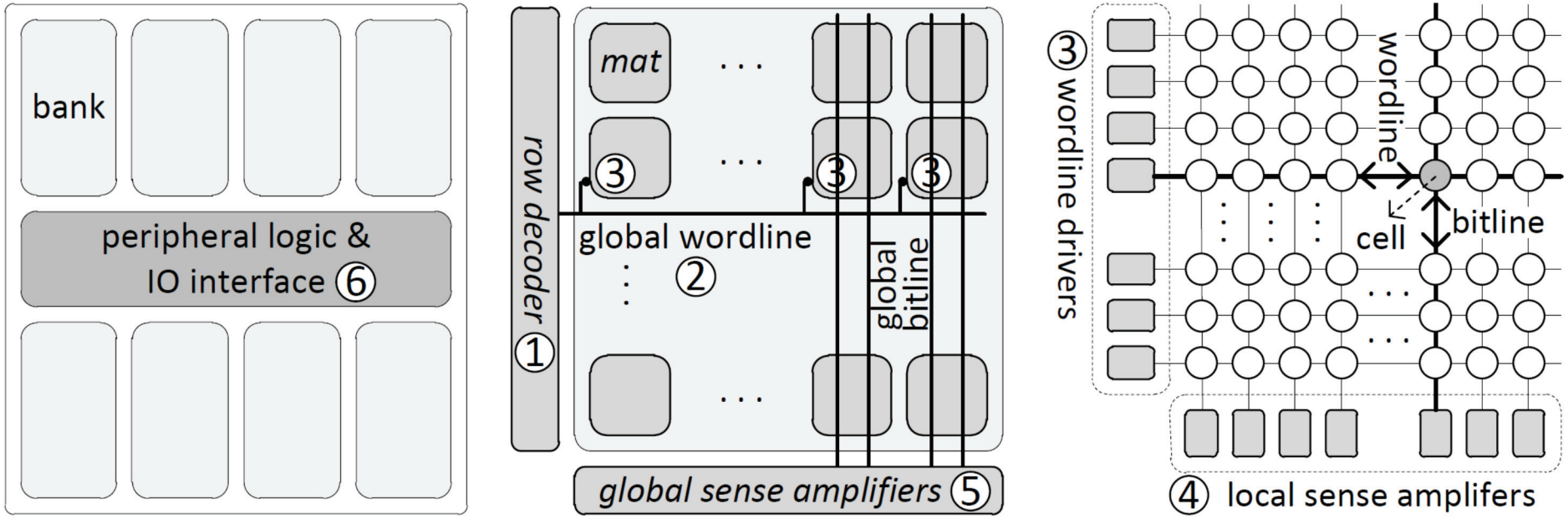
[1]NVIDIA [2]Carnegie Mellon University
[3]University of Virginia [4]Microsoft Research [5]ETH Zürich

Data, Circuit Model Will Be Available at https://github.com/CMU-SAFARI/DIVA-DRAM

SAFARI

Backup Slides

Hierarchical Organization of DRAM

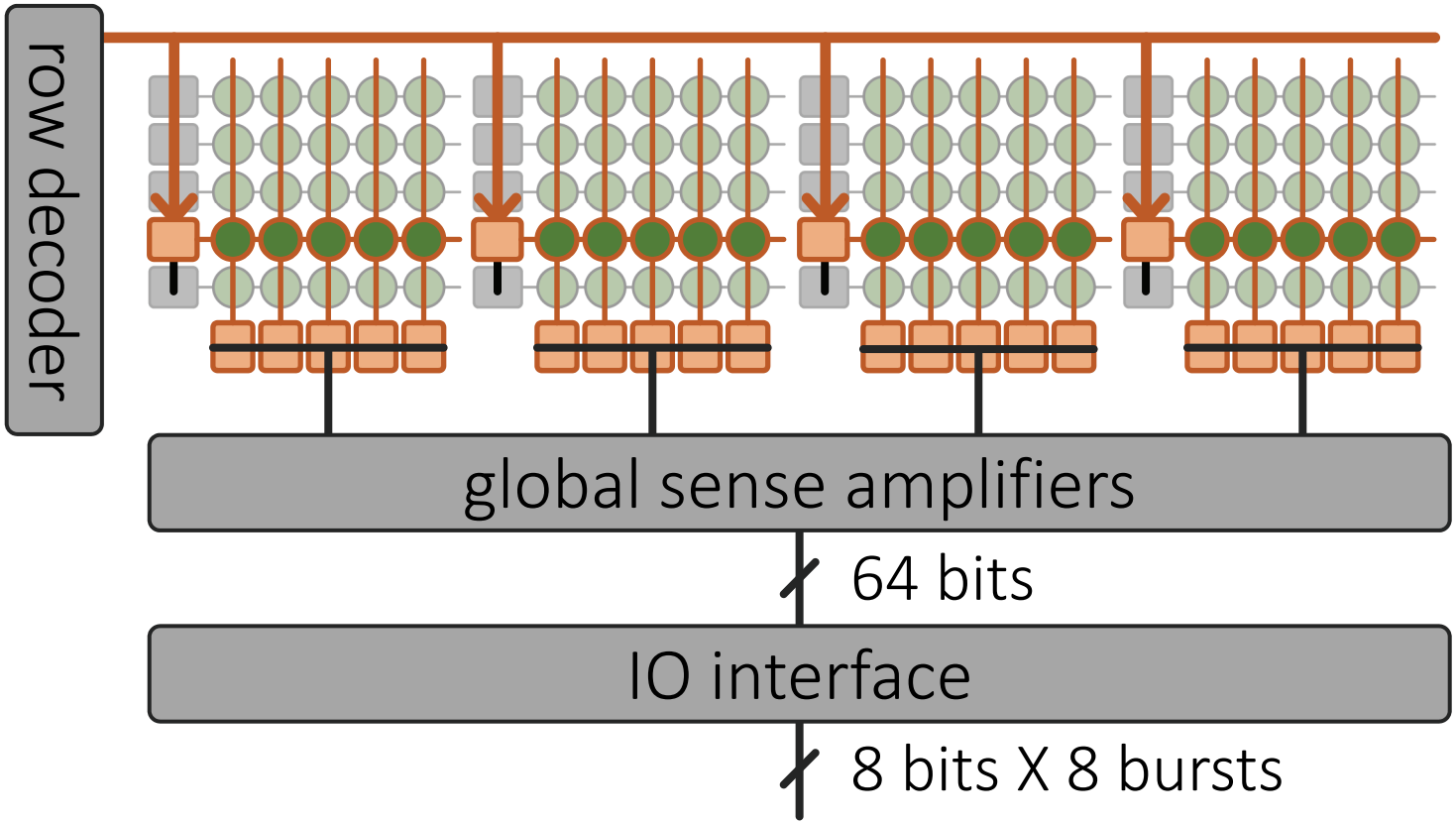


(a) Chip (8 banks)

(b) Bank

(c) Mat (cell array)

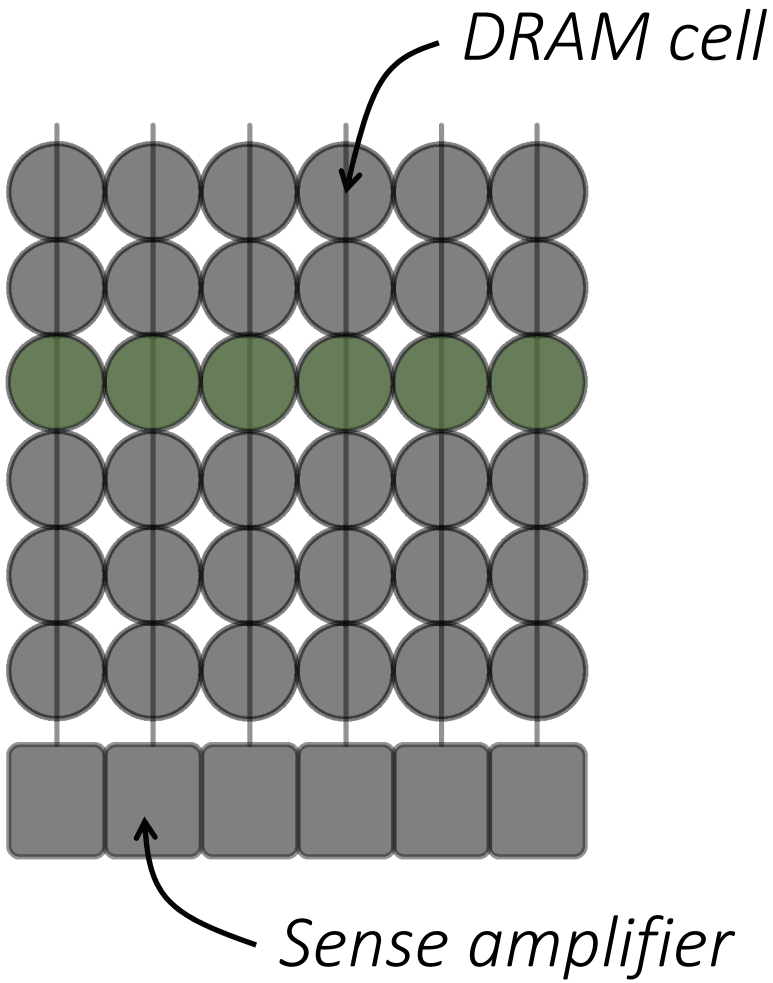
Sending Data From a DRAM Chip



Data in a request → transferred as multiple data bursts

DRAM Stores Data as Charge

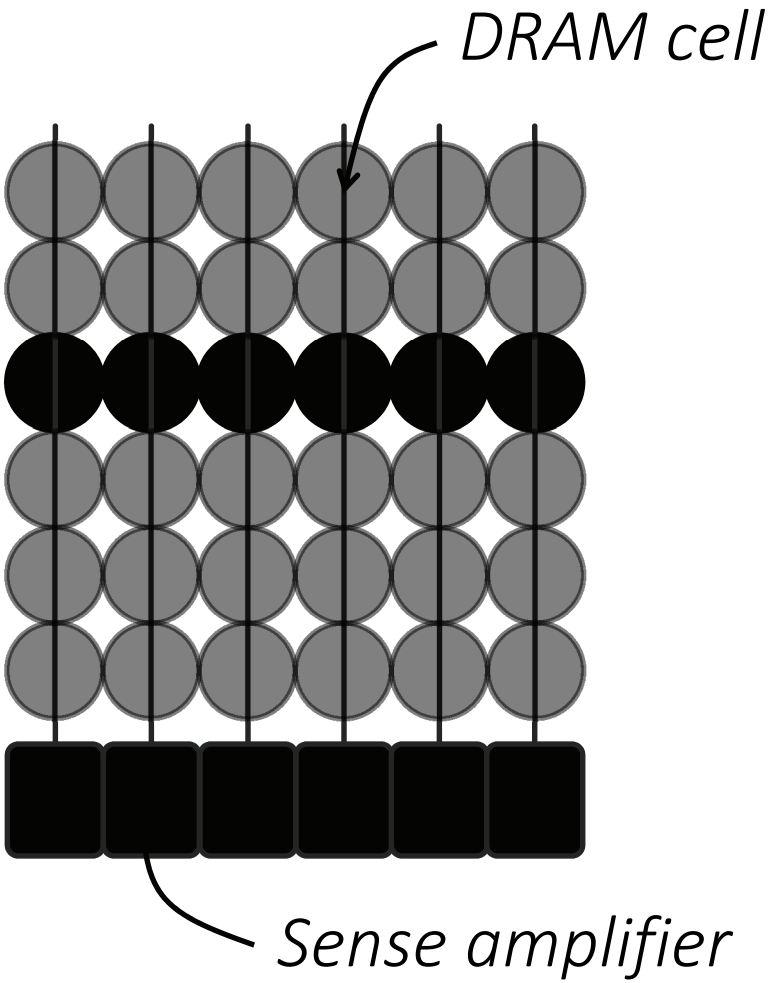
Three steps of charge movement



DRAM Stores Data as Charge

Three steps of charge movement

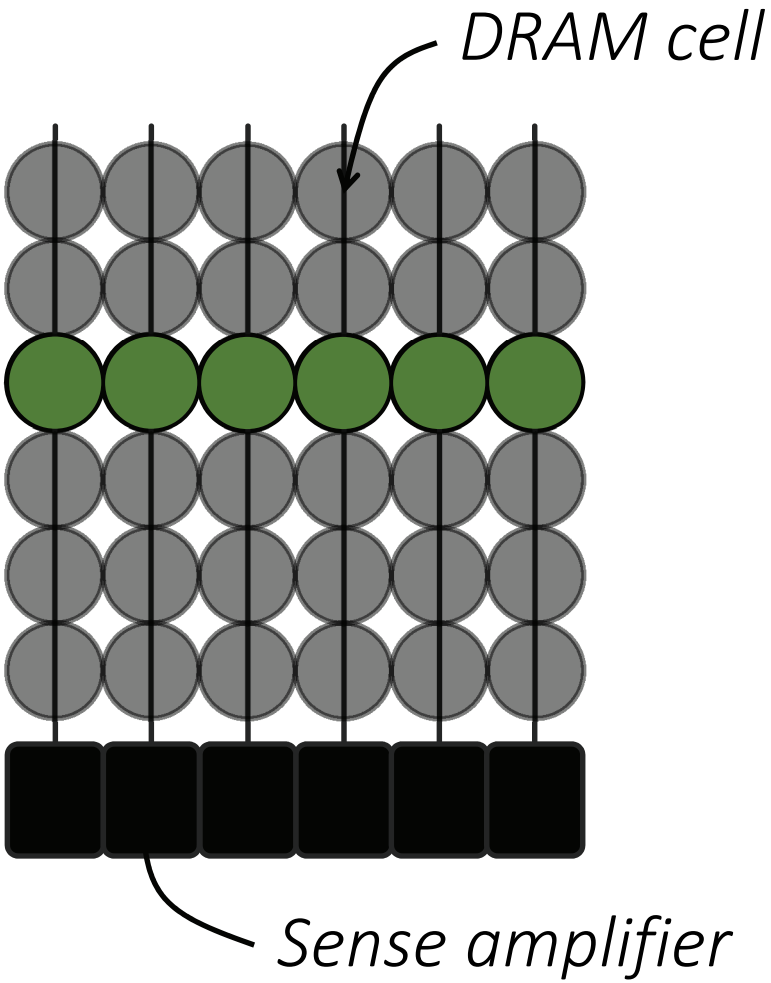
1. Sensing



DRAM Stores Data as Charge

Three steps of charge movement

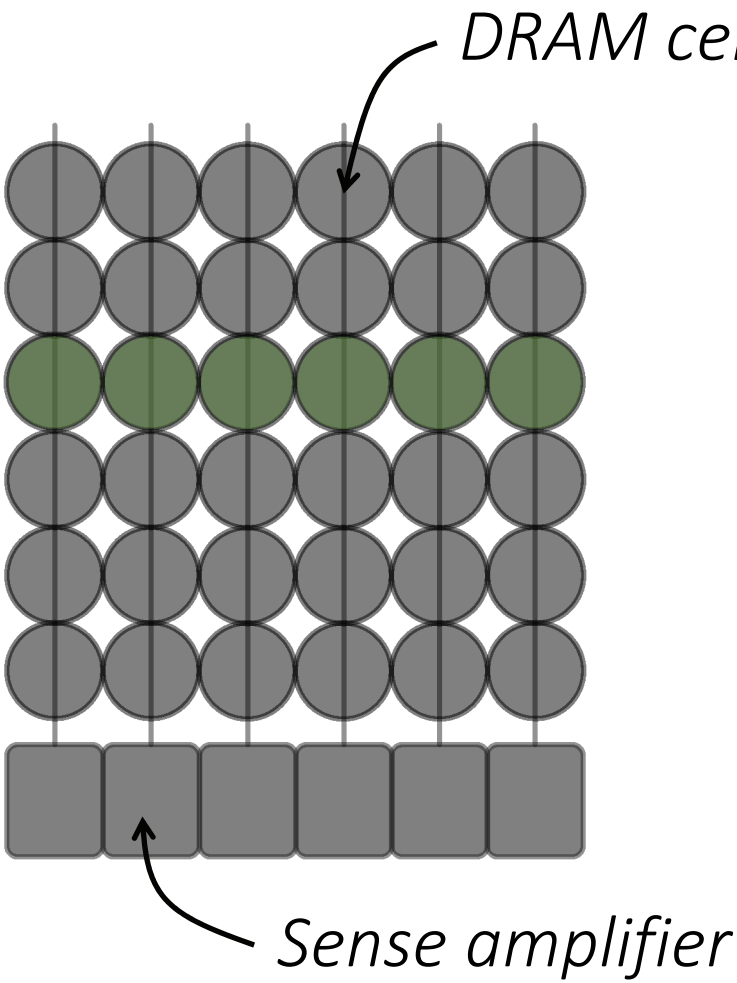
1. Sensing
2. Restore



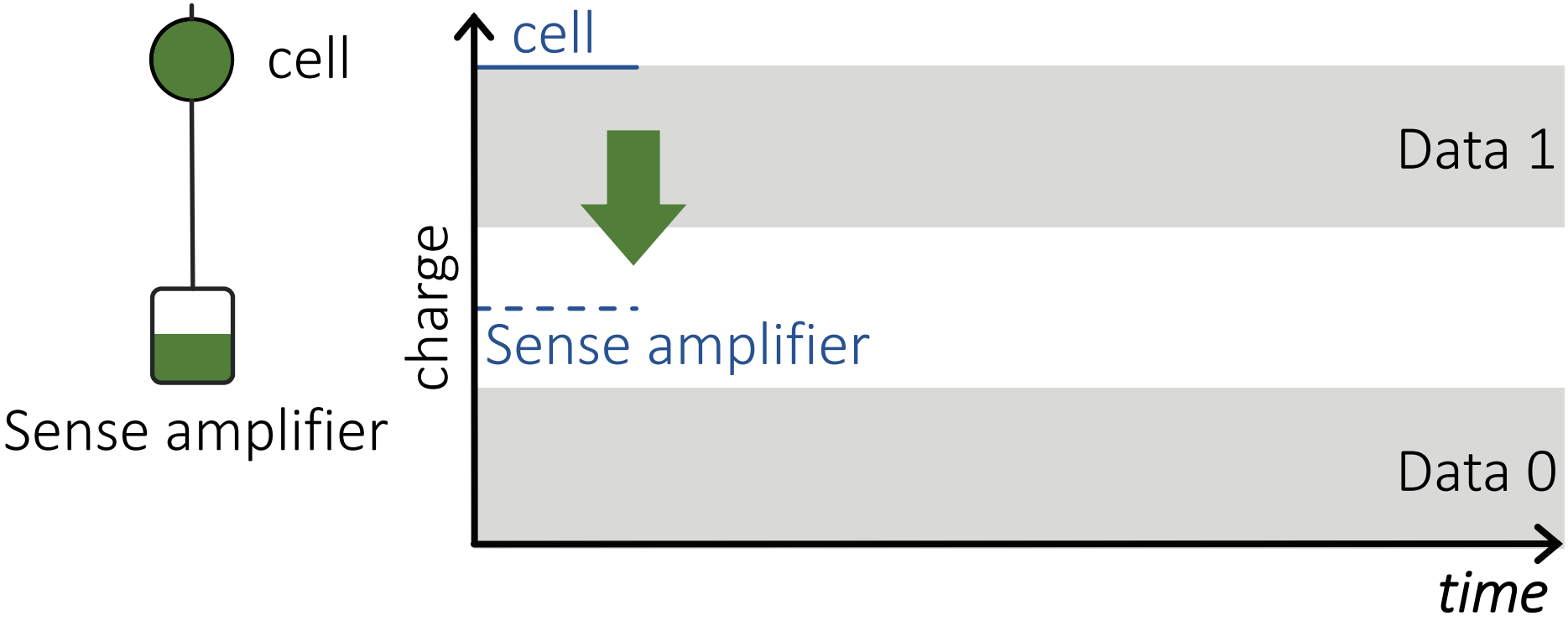
DRAM Stores Data as Charge

Three steps of charge movement

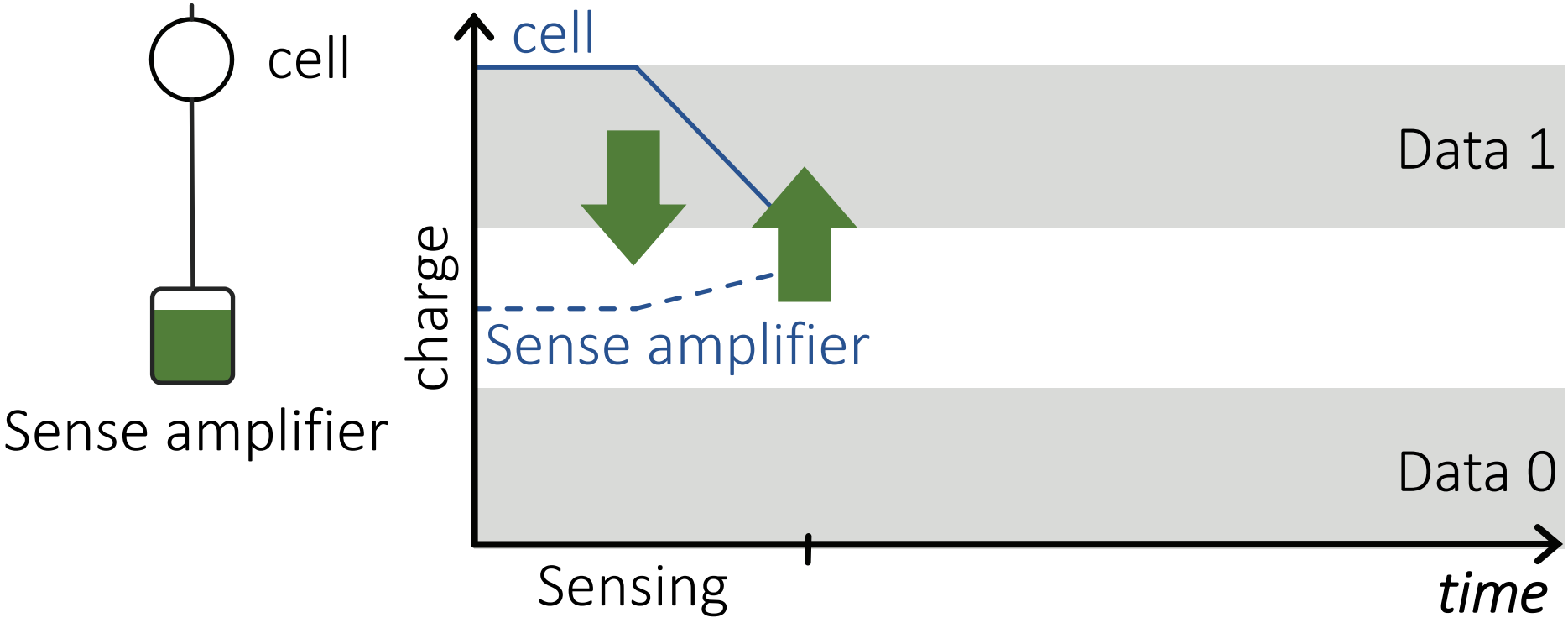
1. Sensing
2. Restore
3. Precharge



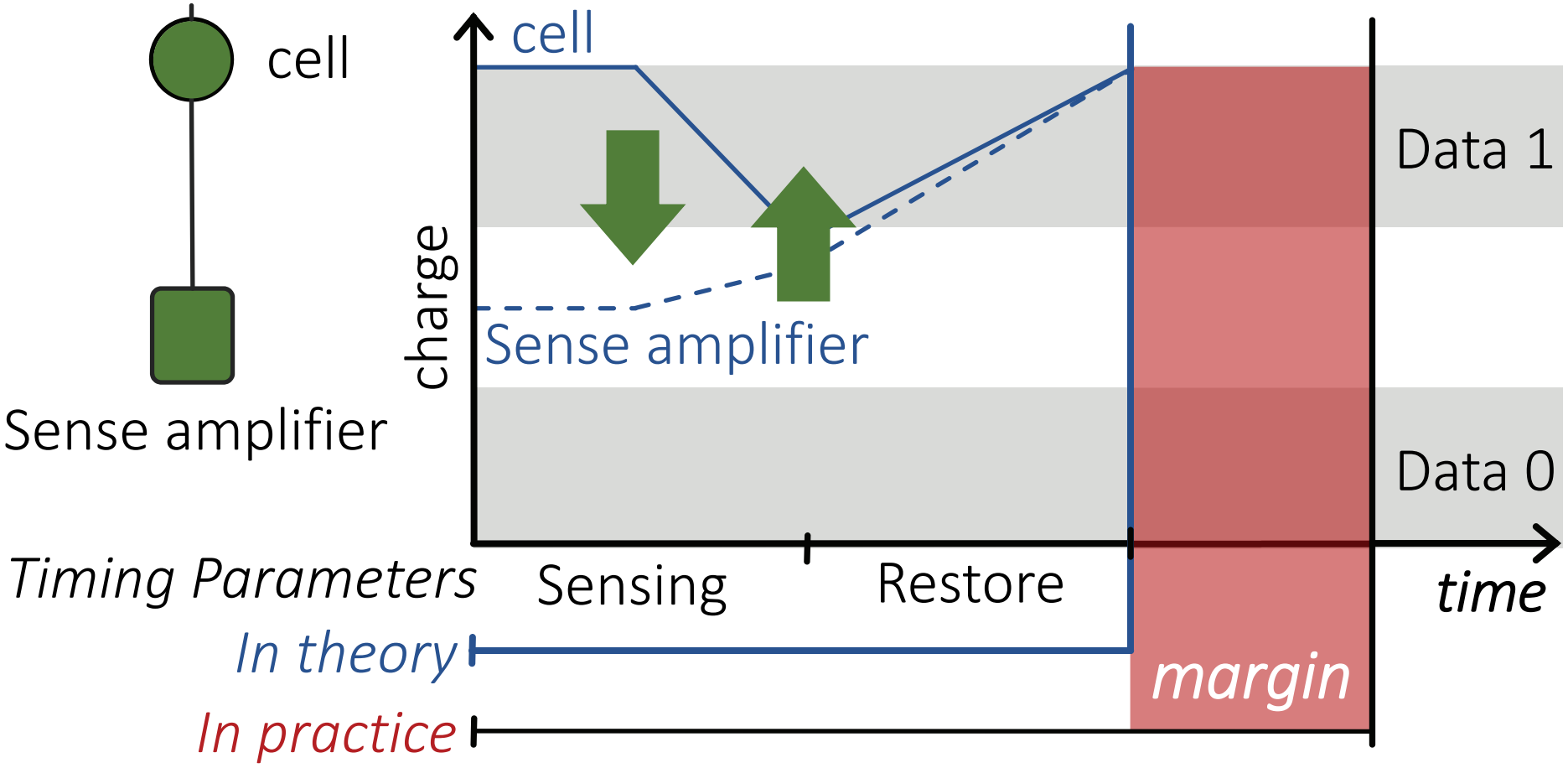
DRAM Charge over Time



DRAM Charge over Time



DRAM Charge over Time



Why does DRAM need the extra timing margin?

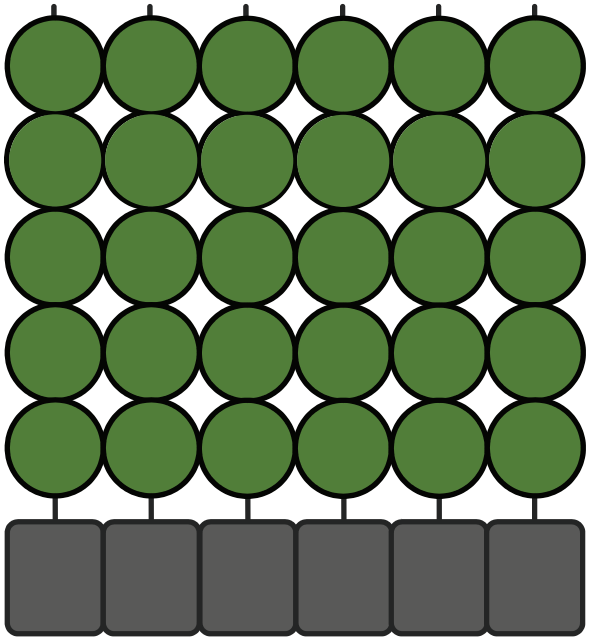
Two Reasons for Timing Margin

1. Process Variation
   - DRAM cells are not equal
   - Leads to extra timing margin for cells that can store large amount of charge

2. Temperature Dependence

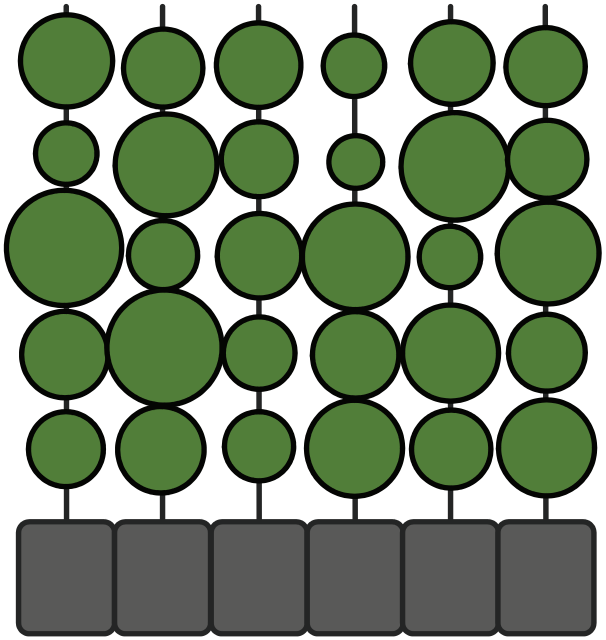
DRAM Cells are Not Equal

Ideal



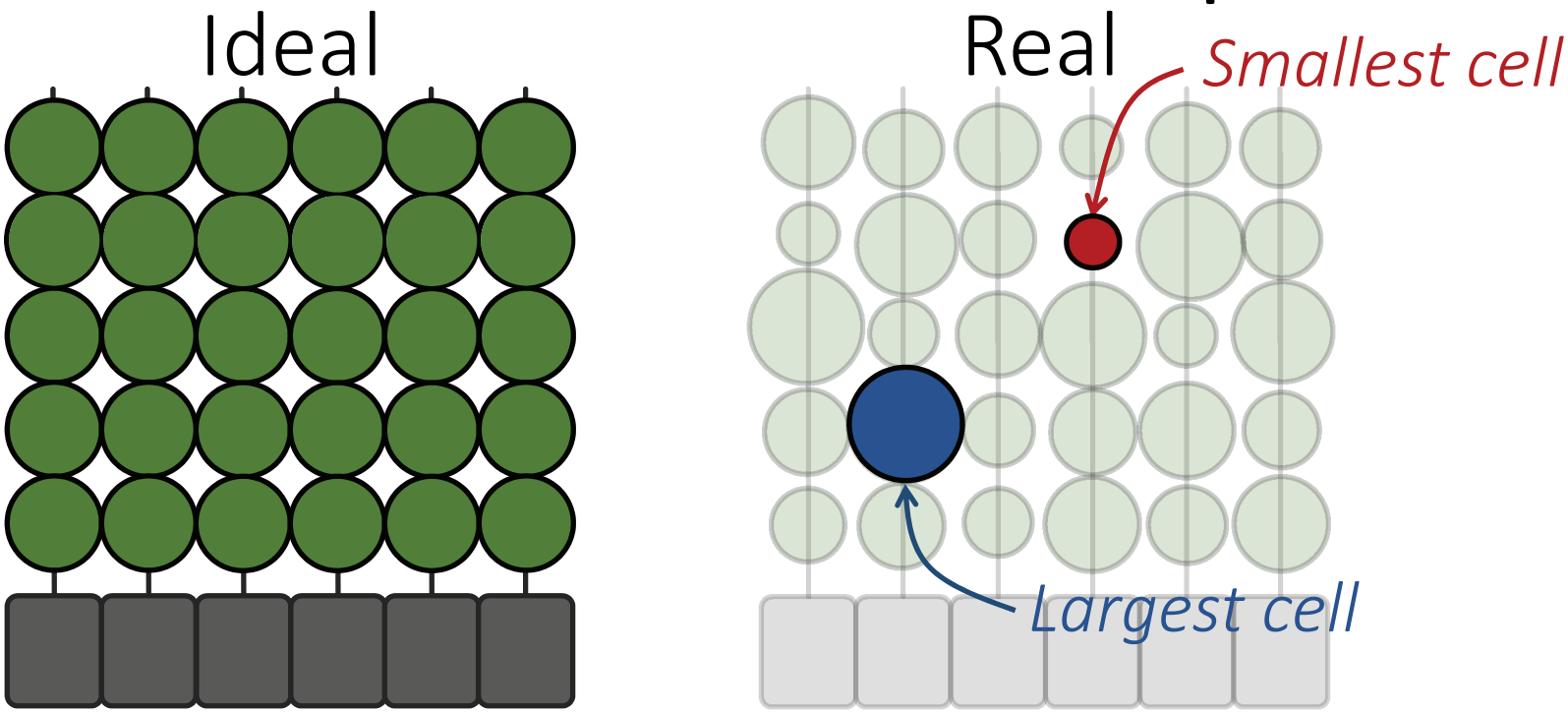
Same size →
Same charge →
Same latency

Real



Different size →
Different charge →
Different latency

DRAM Cells are Not Equal



Large variation in cell size →

Large variation in charge →

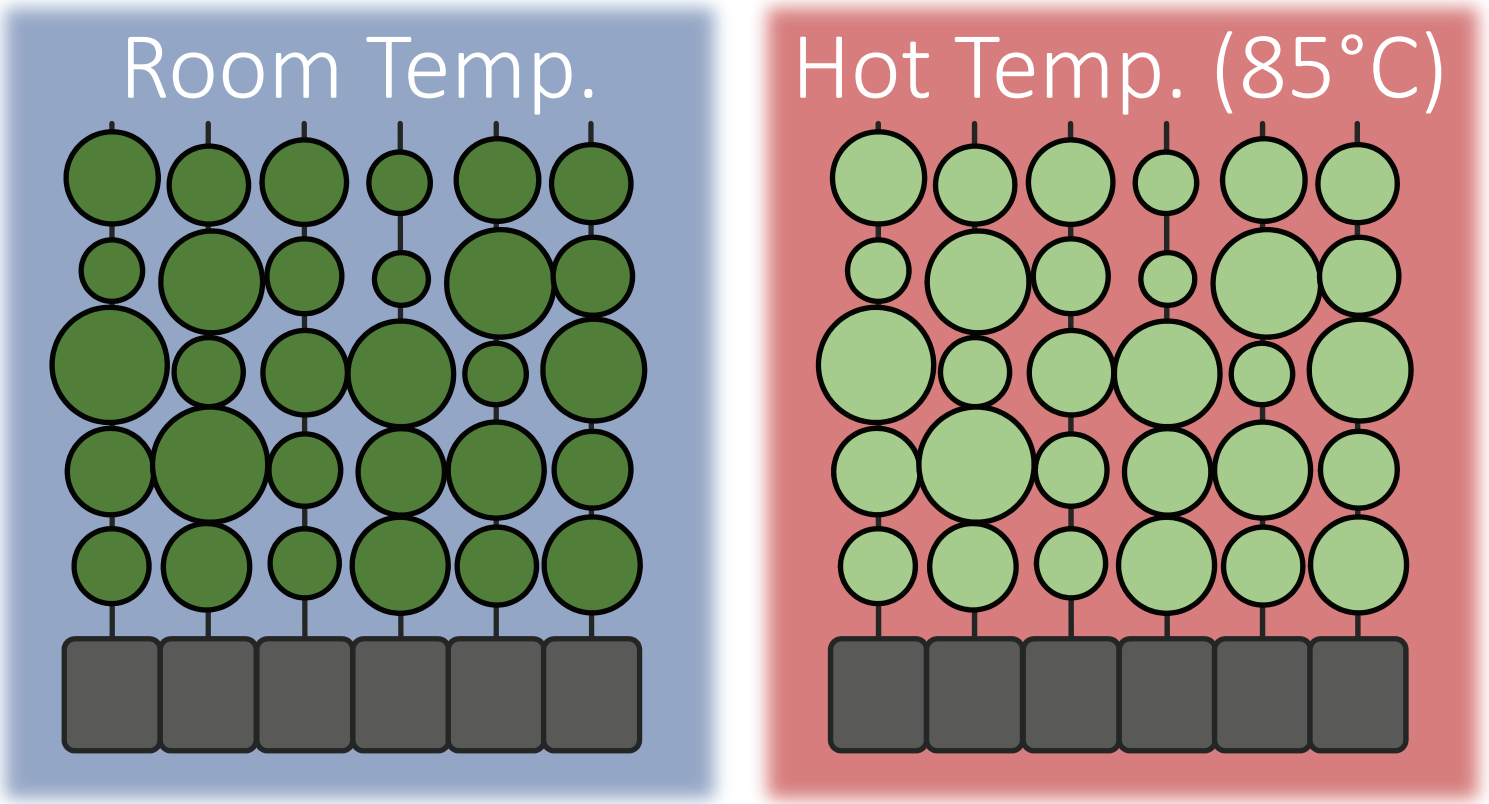
Large variation in access latency

Two Reasons for Timing Margin

1. Process Variation
   - DRAM cells are not equal
   - Leads to *extra timing margin* for cells that can store large amount of charge

2. Temperature Dependence
   - DRAM leaks more charge at higher temperature
   - Leads to extra timing margin when operating at low temperature

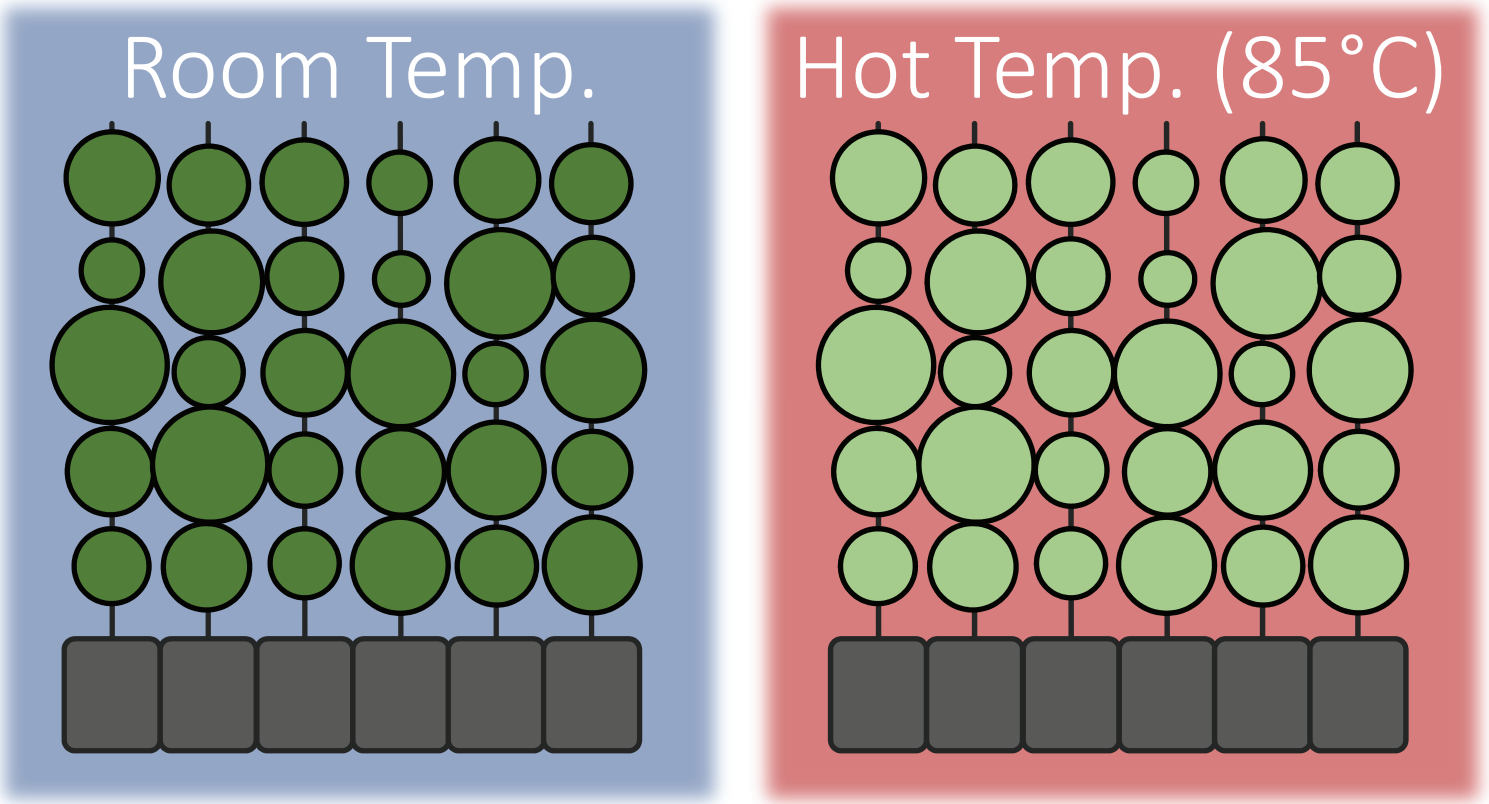
Charge Leakage ∝ Temperature



Small leakage

Large leakage

Charge Leakage ∝ Temperature



Cells store small charge at high temperature and large charge at low temperature
→ Large variation in access latency

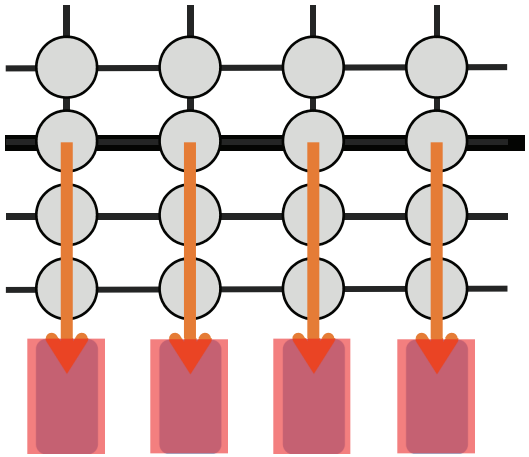
DRAM Timing Parameters

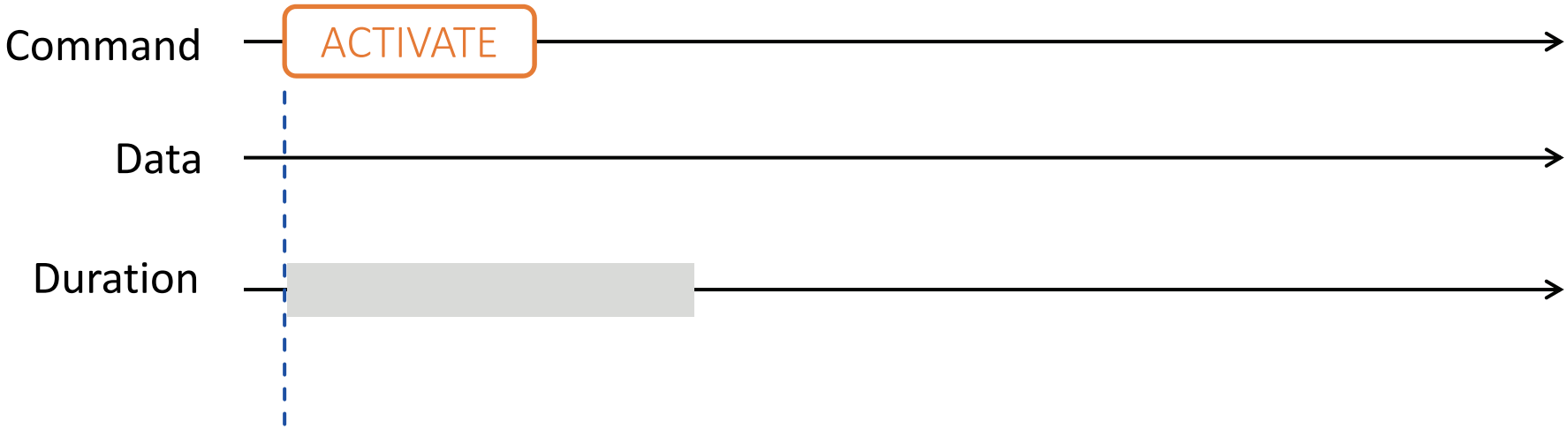
- DRAM timing parameters are dictated by *the worst case*
  - The smallest cell with the smallest charge **in all DRAM products**
  - Operating at **the highest temperature**

- Large timing margin for the common case

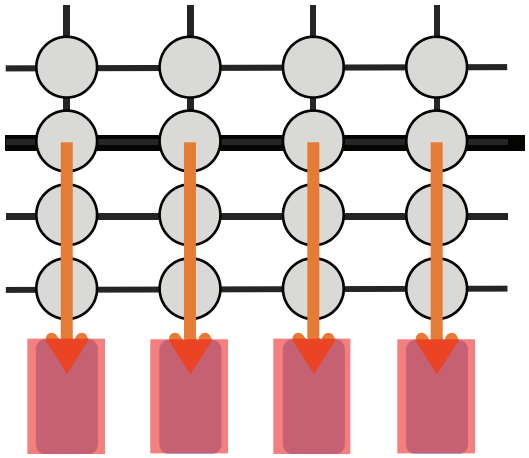
→ Can lower latency for the common case

DRAM Timing Parameters



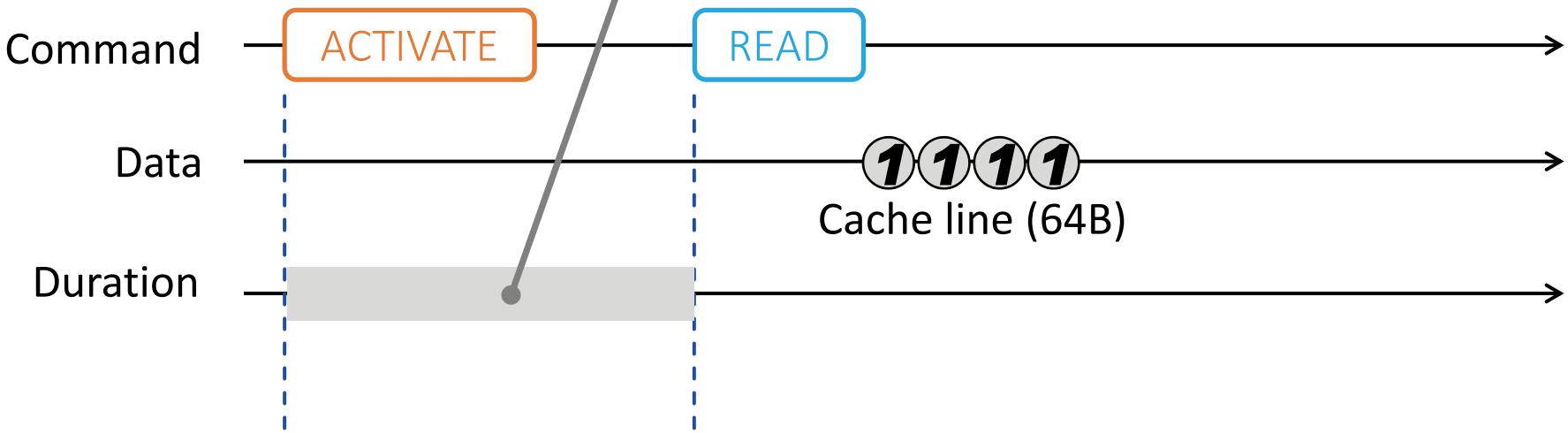


DRAM Timing Parameters

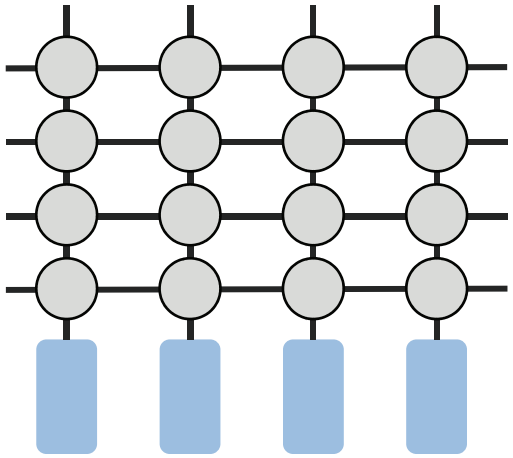


1 **Activation latency: tRCD**

(13ns / 50 cycles)

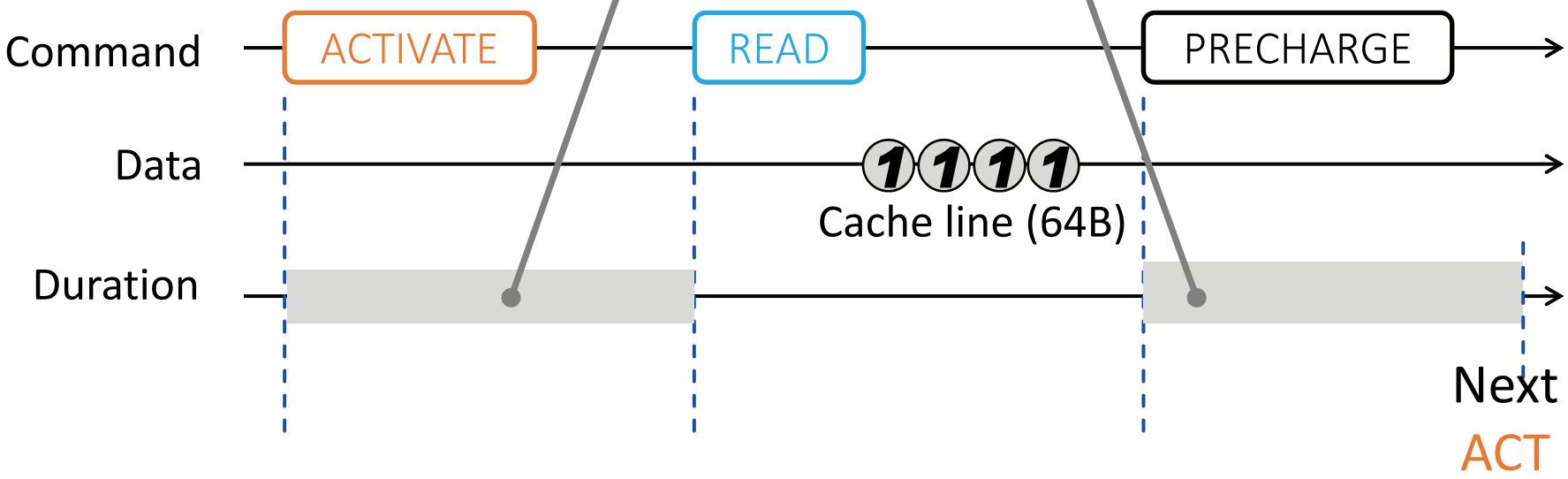


DRAM Timing Parameters

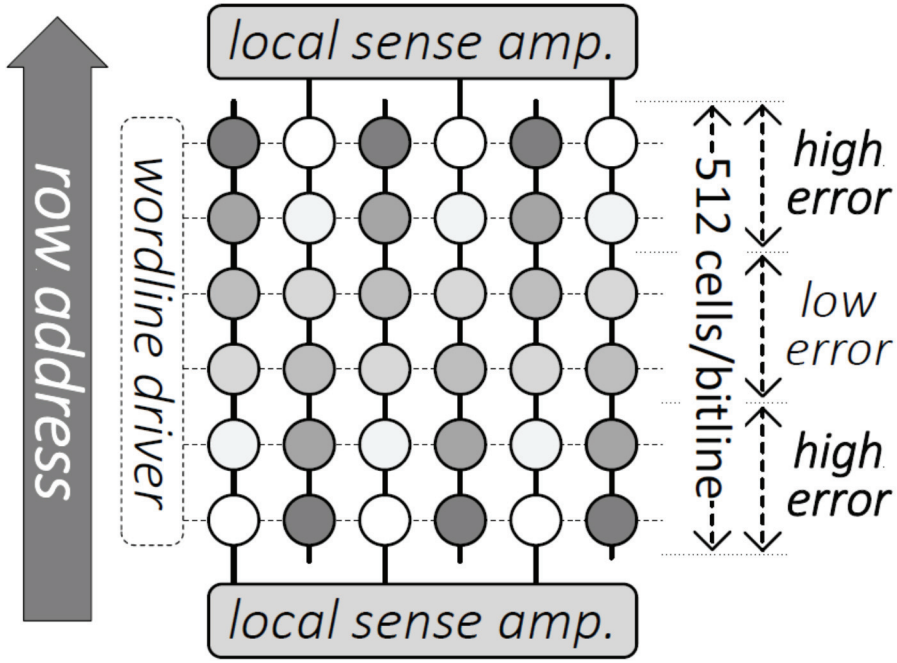


1 **Activation latency: tRCD**
(13ns / 50 cycles)

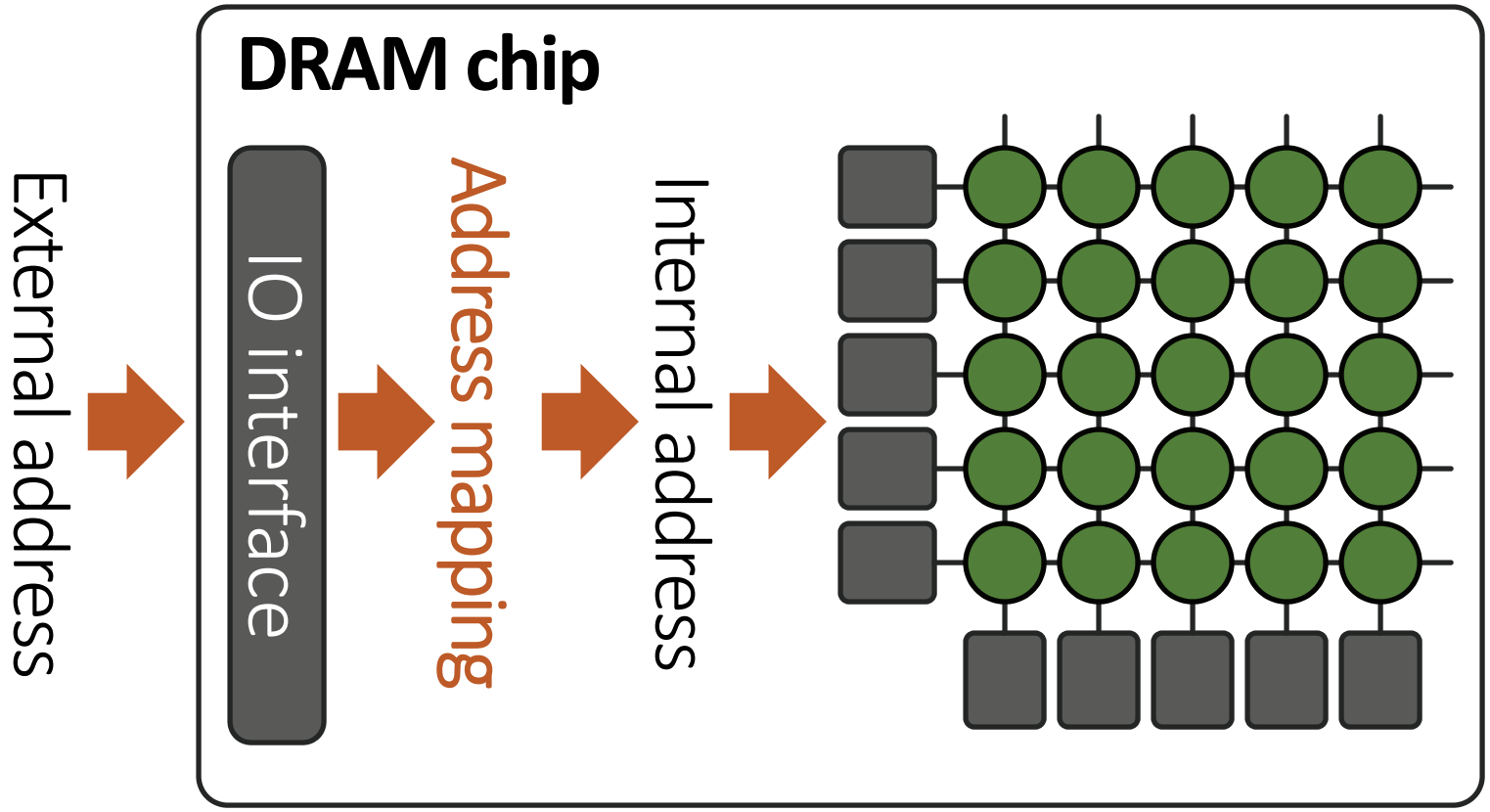
2 **Precharge latency: tRP**
(13ns / 50 cycles)



Design-Induced Variation in Open Bitline DRAM



Challenge: External ≠ Internal



External address ≠ Internal address

DRAM-Internal vs. DRAM-External

IntMSB	IntMID	IntLSB	
1	1	1	7
1	1	0	6
1	0	1	5
1	0	0	4
0	1	1	3
0	1	0	2
0	0	1	1
0	0	0	0

near distance from s/a far

ExtMID	ExtMSB	ExtLSB	
1	1	1	7
1	1	0	6
0	1	1	5
0	1	0	4
1	0	0	3
1	0	1	2
0	0	1	1
0	0	0	0

low error count high

Estimated Mapping External→Internal

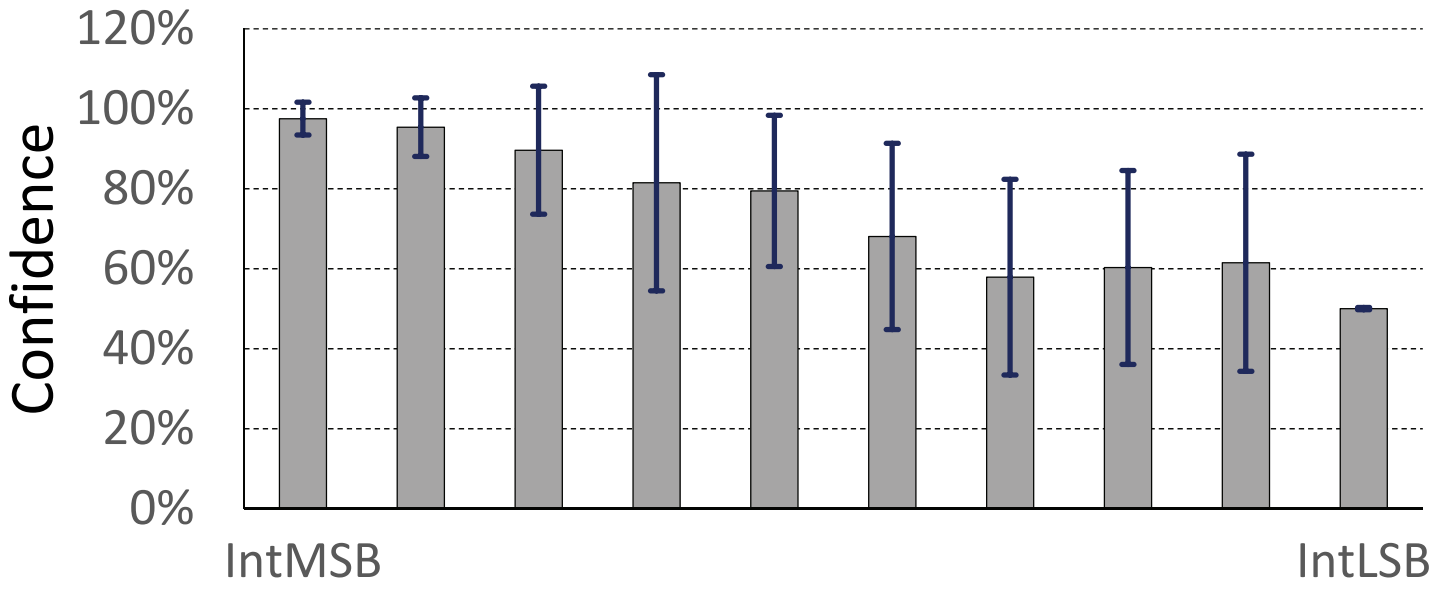
ExtMSB → IntMID
4/4 = 100%

ExtMID → IntMSB
4/4 = 100%

ExtLSB → IntLSB
3/4 = 75%

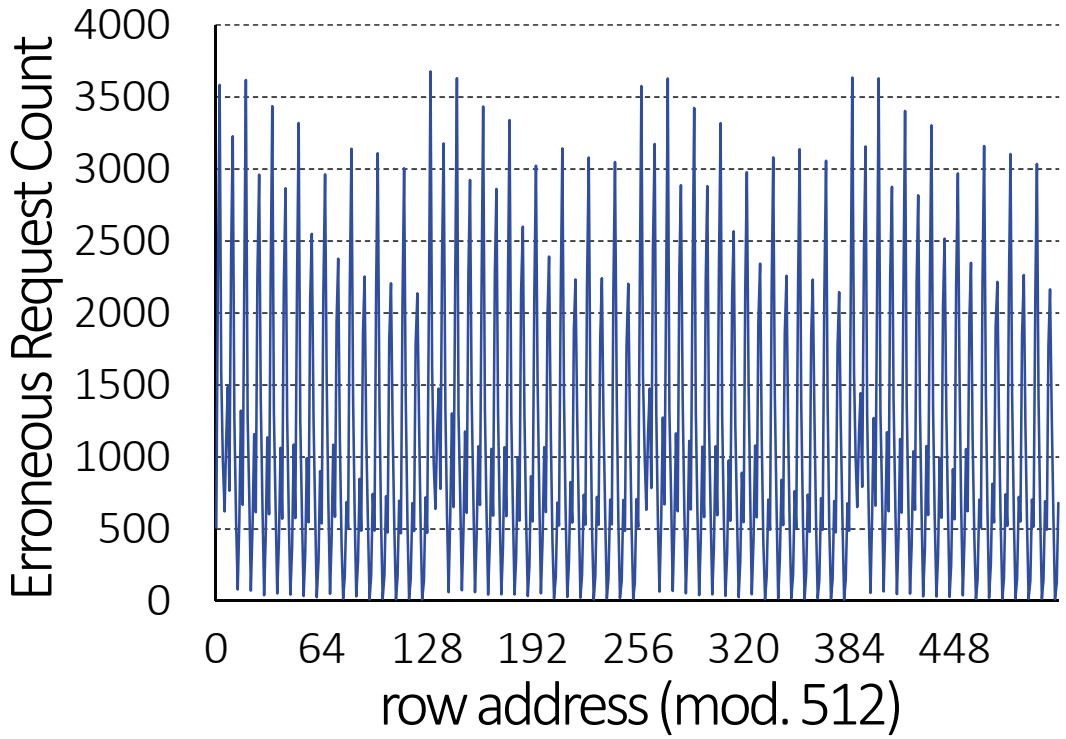
Estimated Mapping (External → Internal)
Based on Error Counts for the External Address

Row Address Mapping Confidence



1.1. Measuring Row Variation

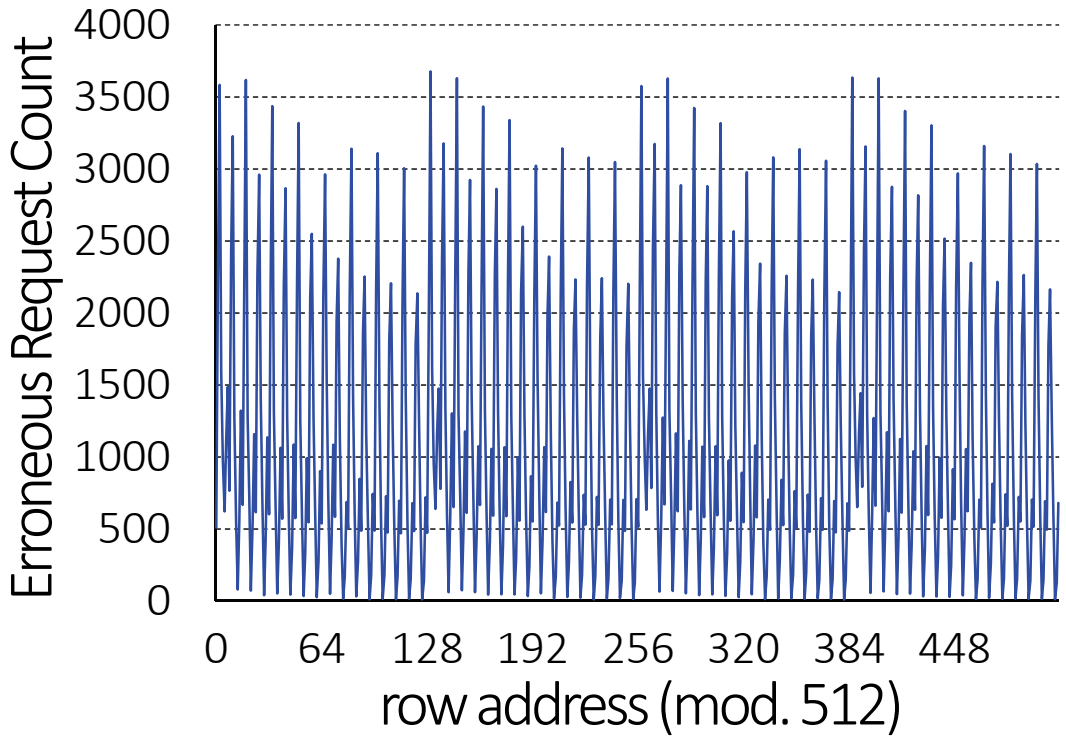
Lower tRP (precharge timing parameter) to 7.5 ns



Periodic Errors

1.1. Measuring Row Variation

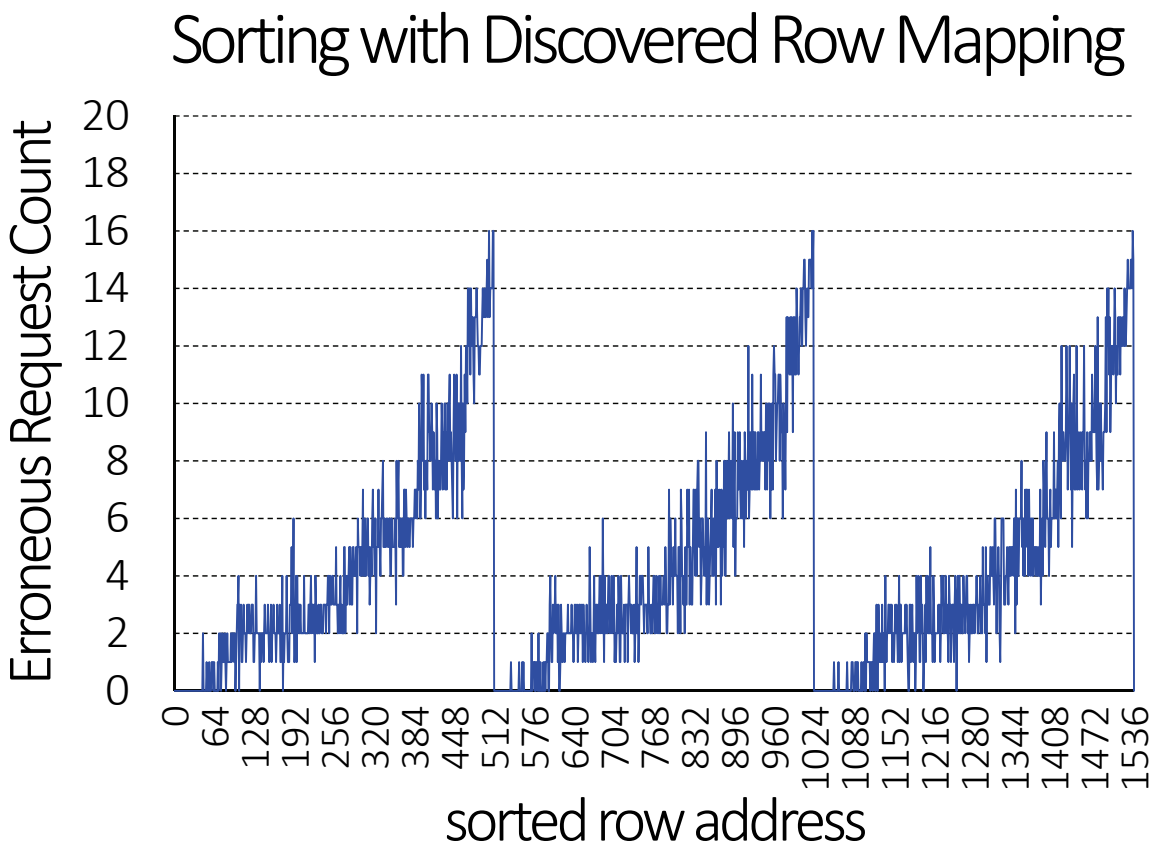
Lower tRP (precharge timing parameter) to 7.5 ns



Need to reverse engineer **row address mapping**

details in our paper

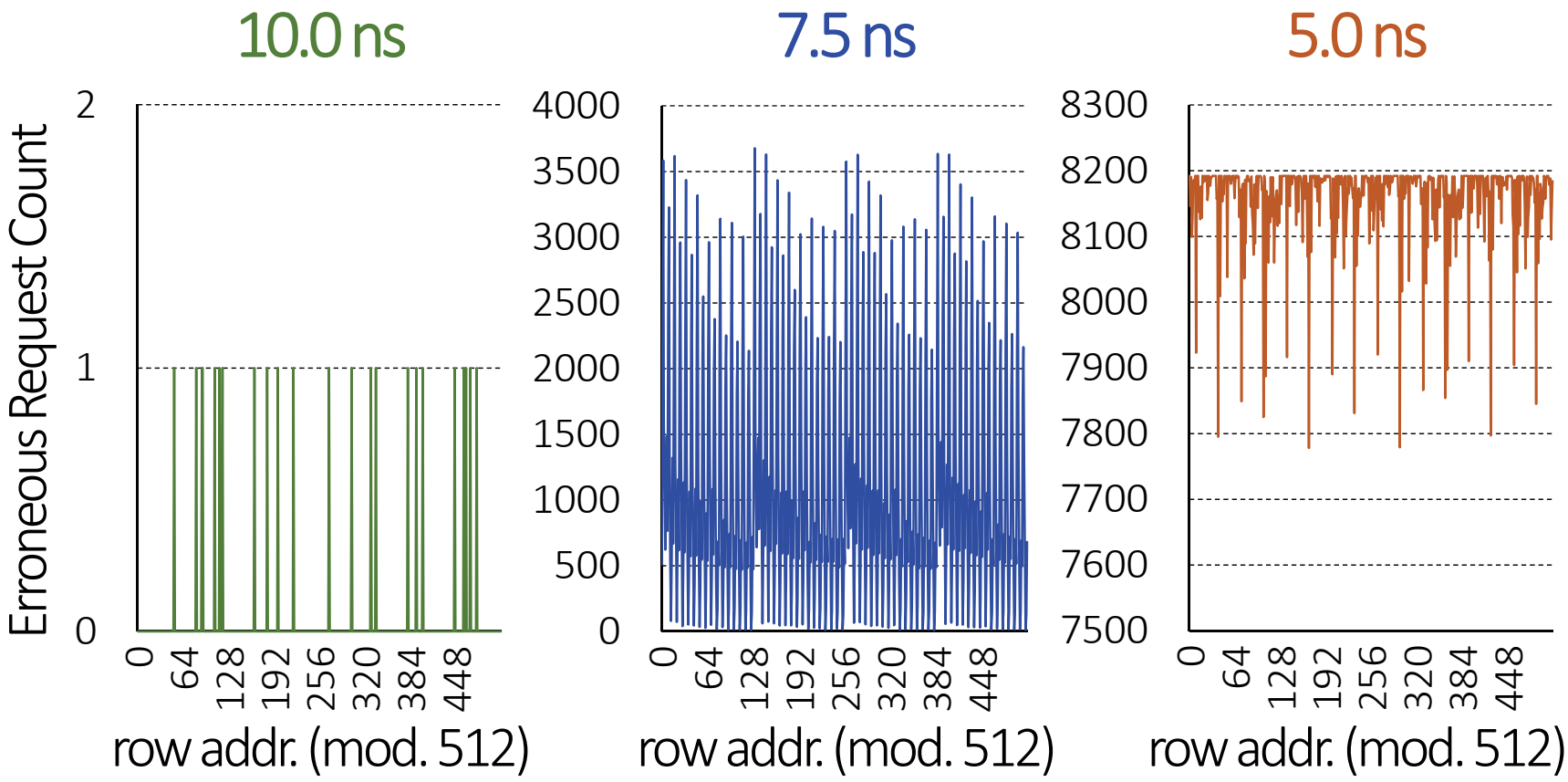
1.2. Periodic Row Variation Behavior



Row error (latency) characteristics
periodically repeat every 512 rows

1.1. Variation in Rows

tRP (precharge timing parameter)

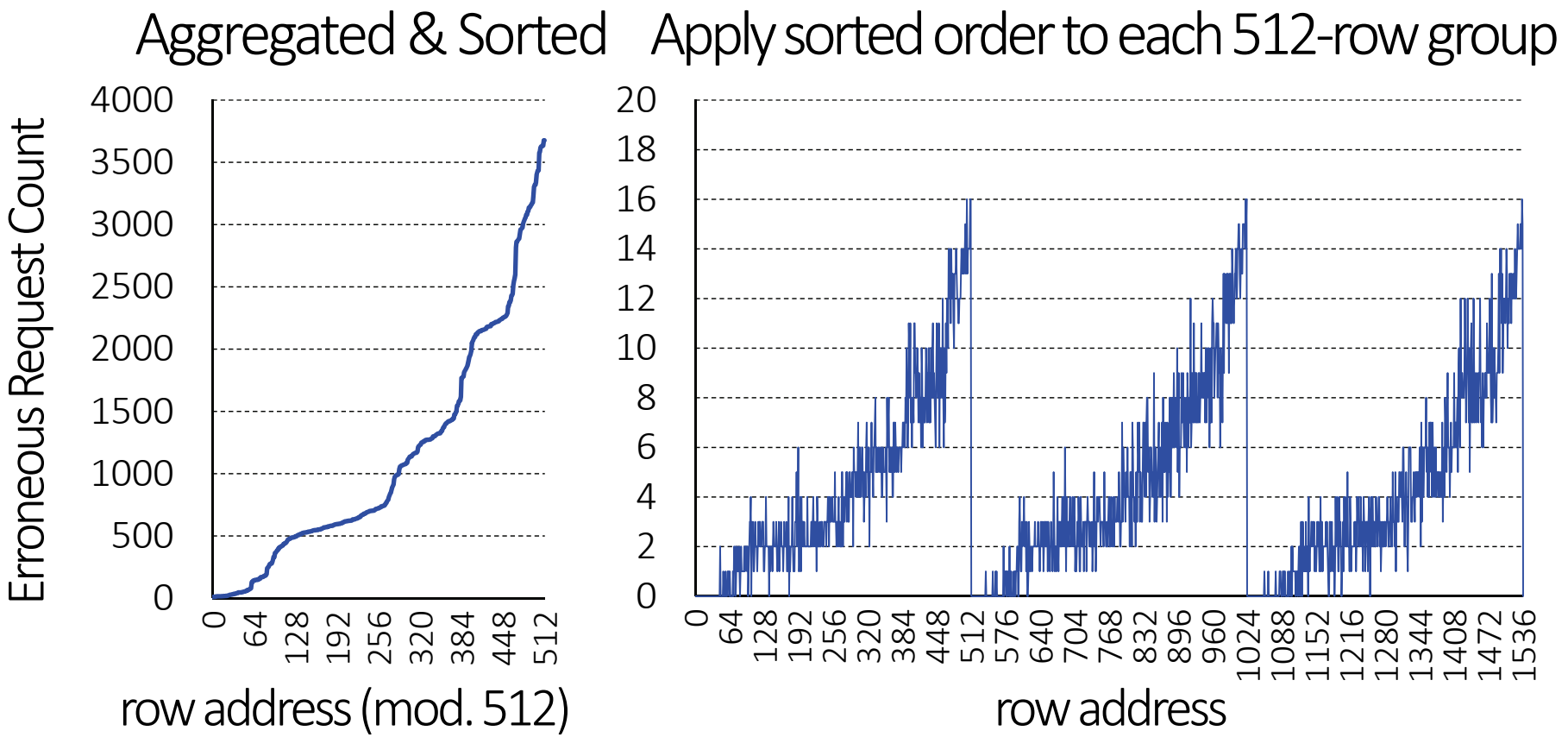


Random Errors

Periodic Errors

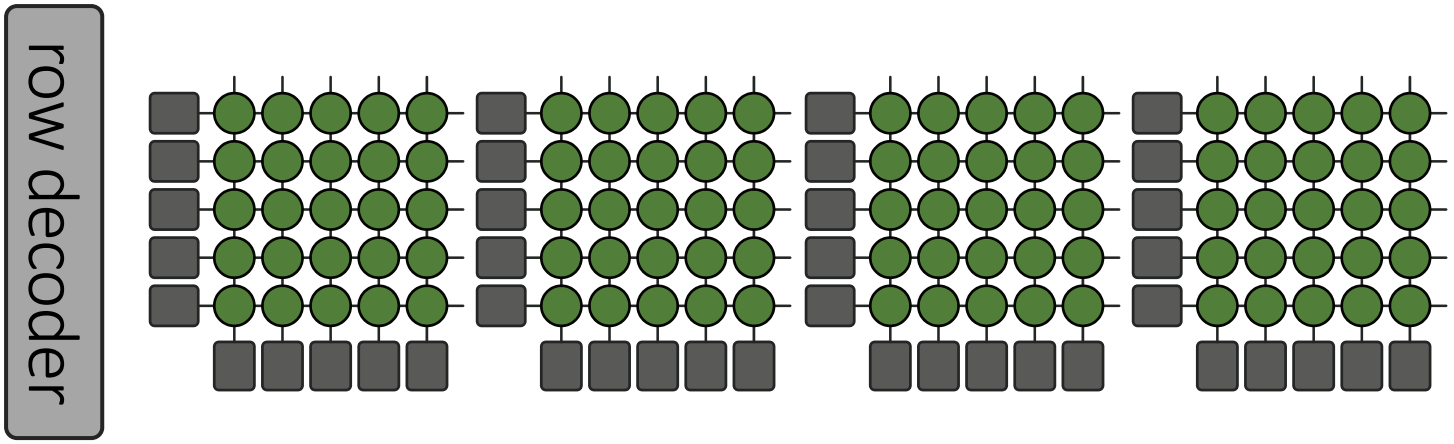
Mostly Errors

1.2. Periodic Row Variation Behavior

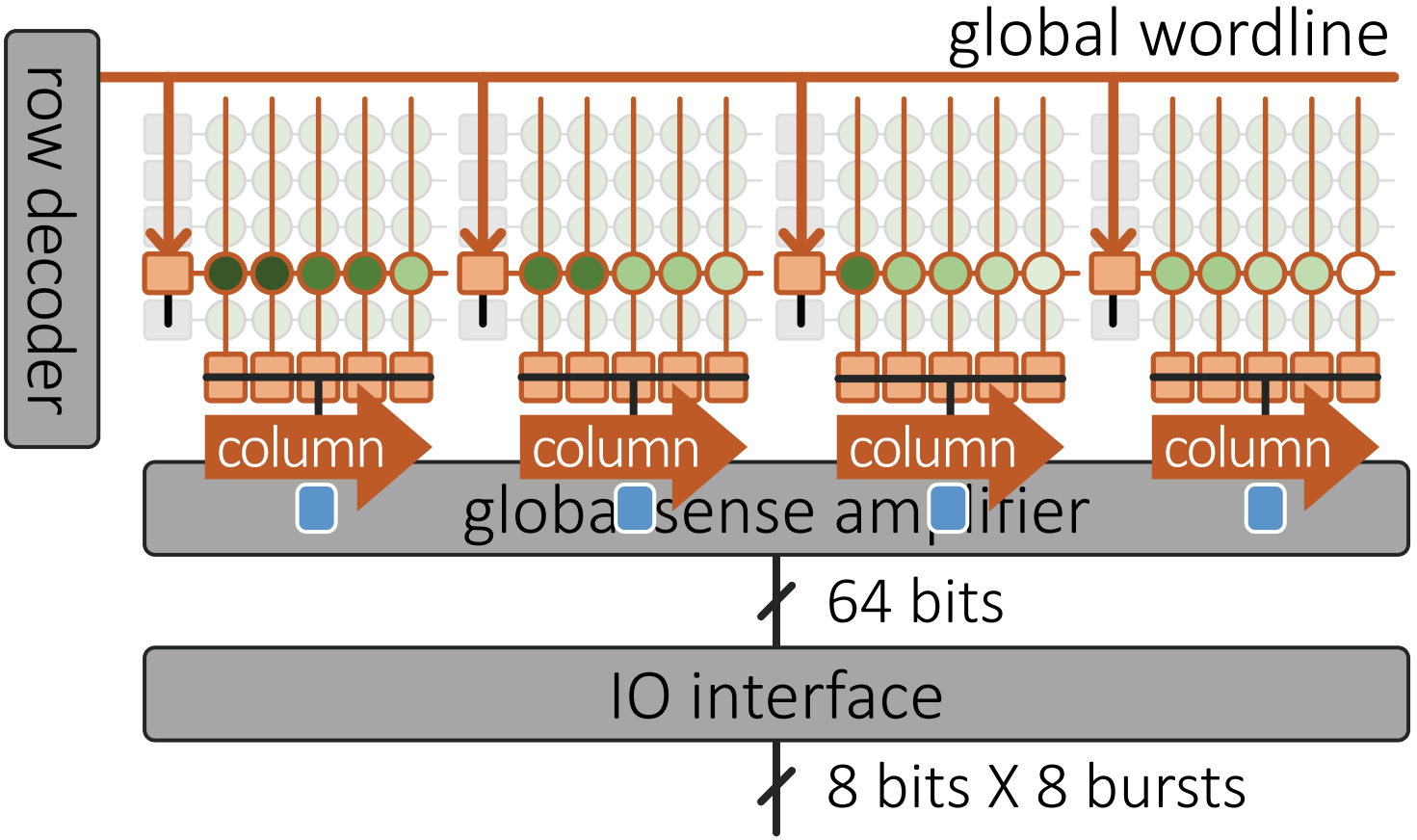


Error (latency) characteristics ***periodically repeat*** every 512 rows

2. Variation Across Columns

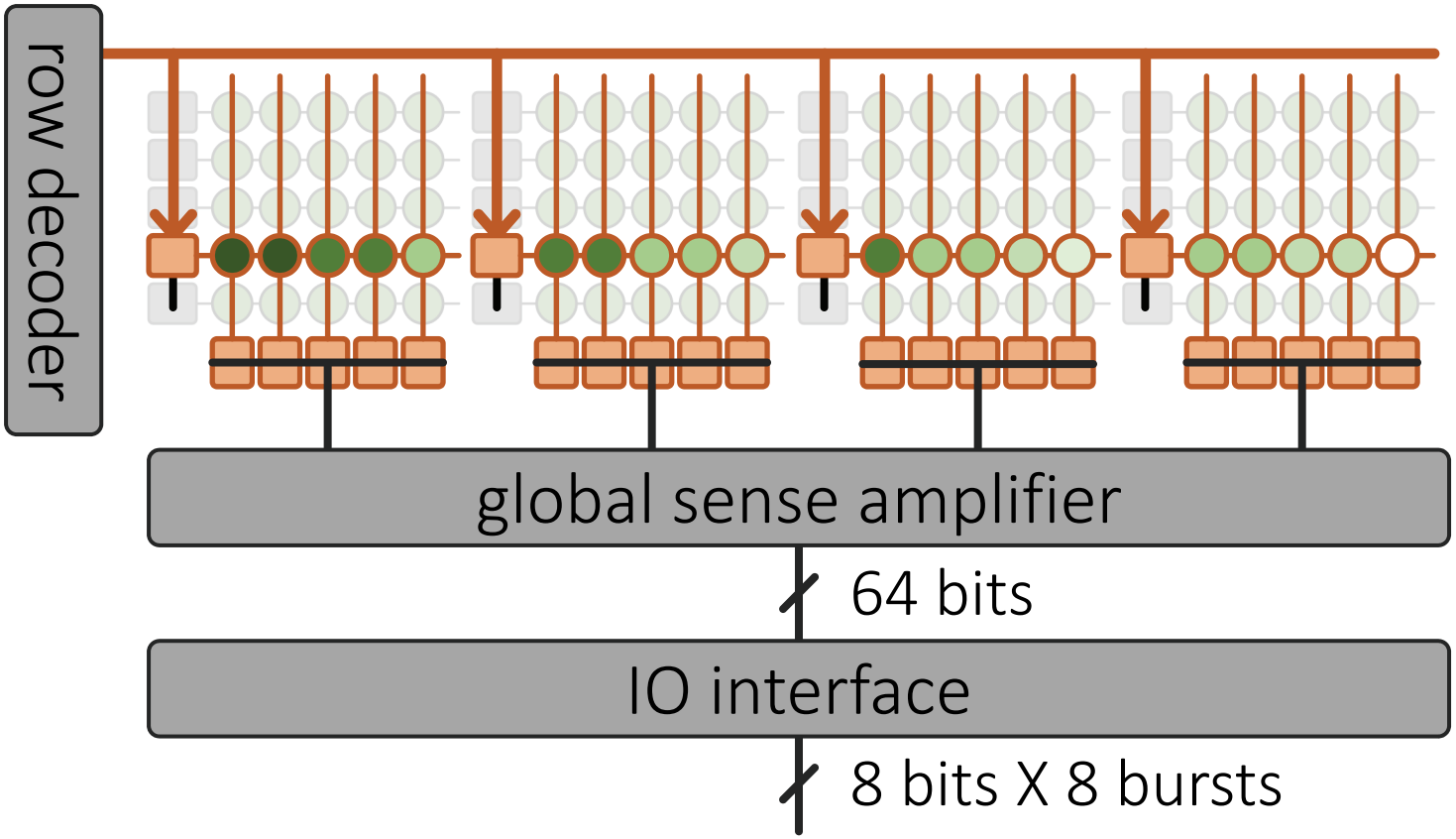


2. Variation Across Columns



Different columns → data from different locations
→ different characteristics

3. Variation in Data Bits

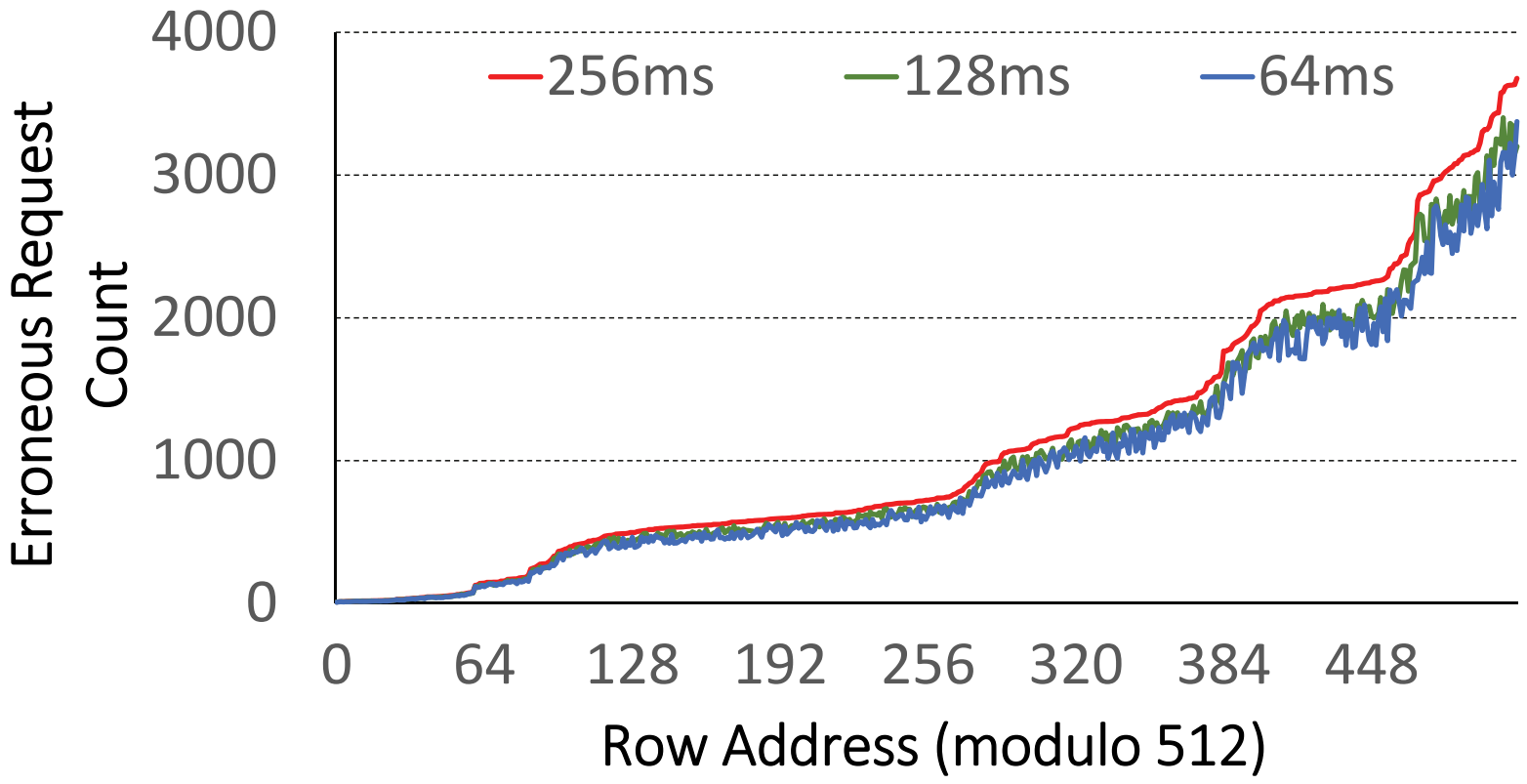


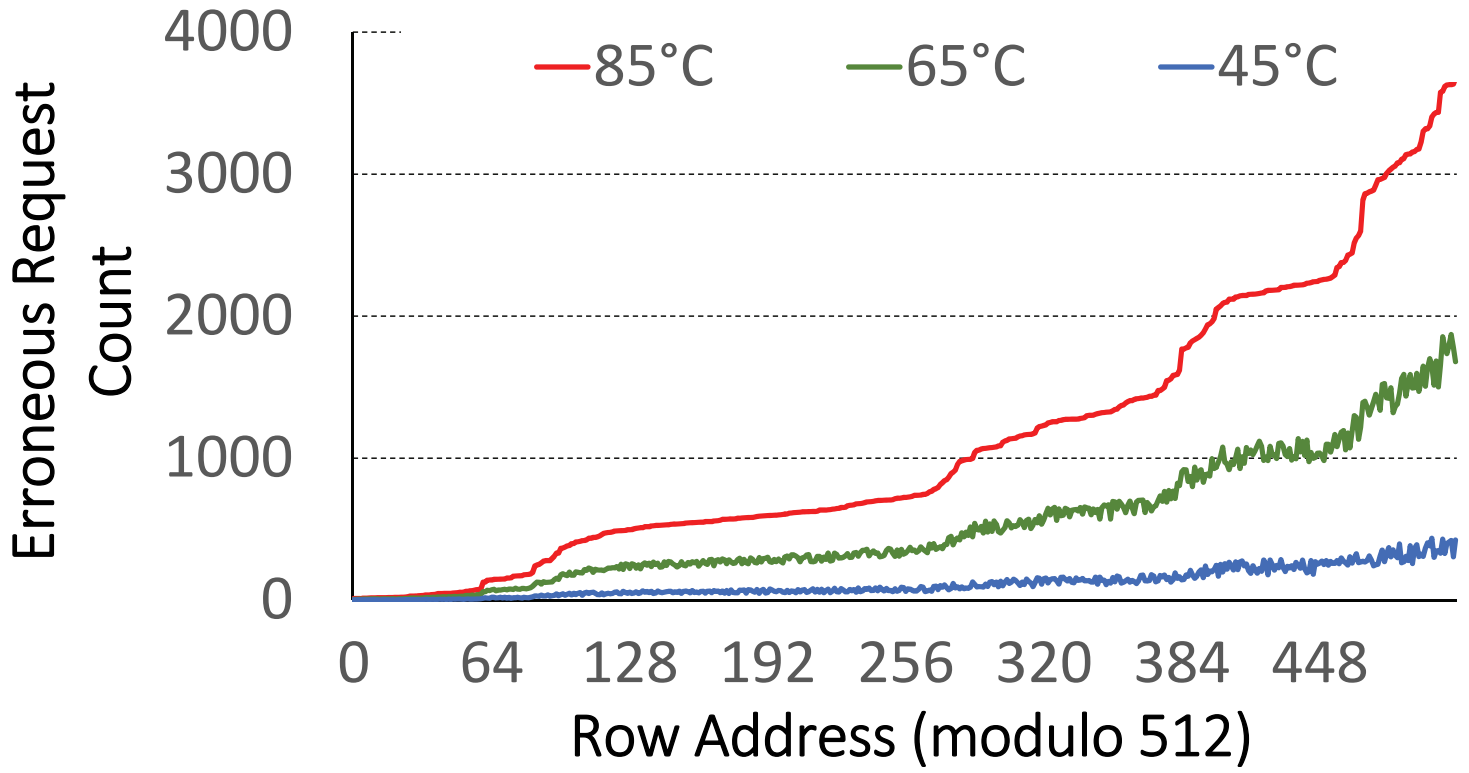
Data in a request → transferred as multiple data bursts

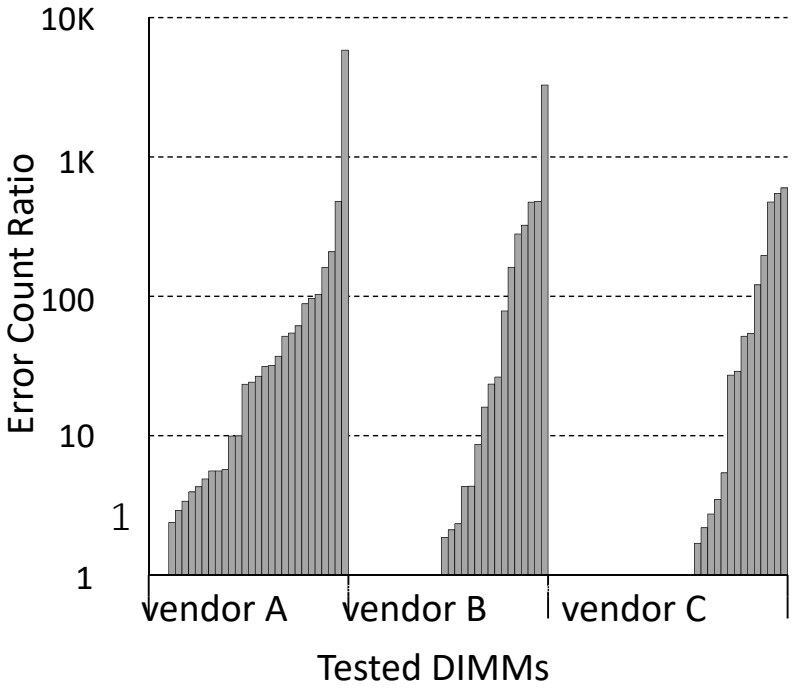
DIVA-DRAM Evaluation Methodology

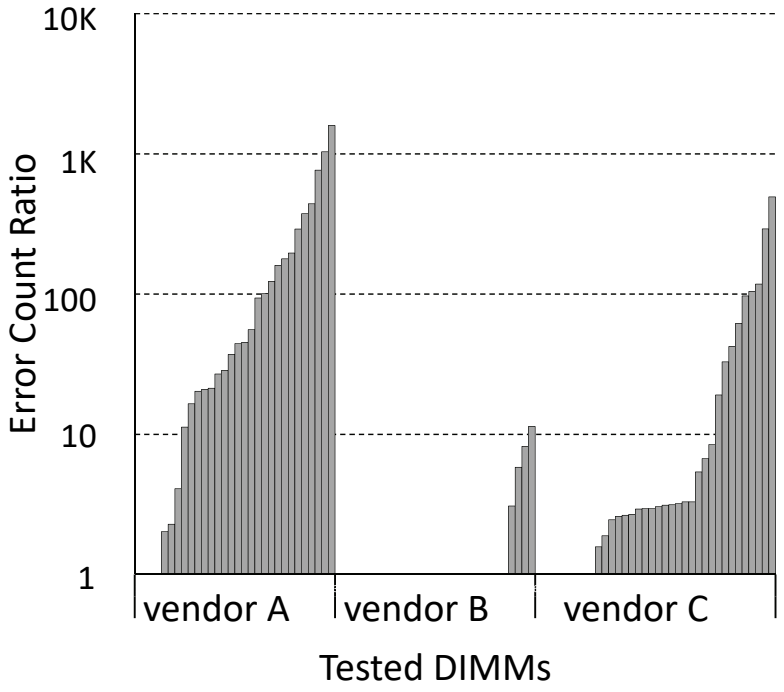
Modified version of Ramulator (cycle accurate DRAM simulator)

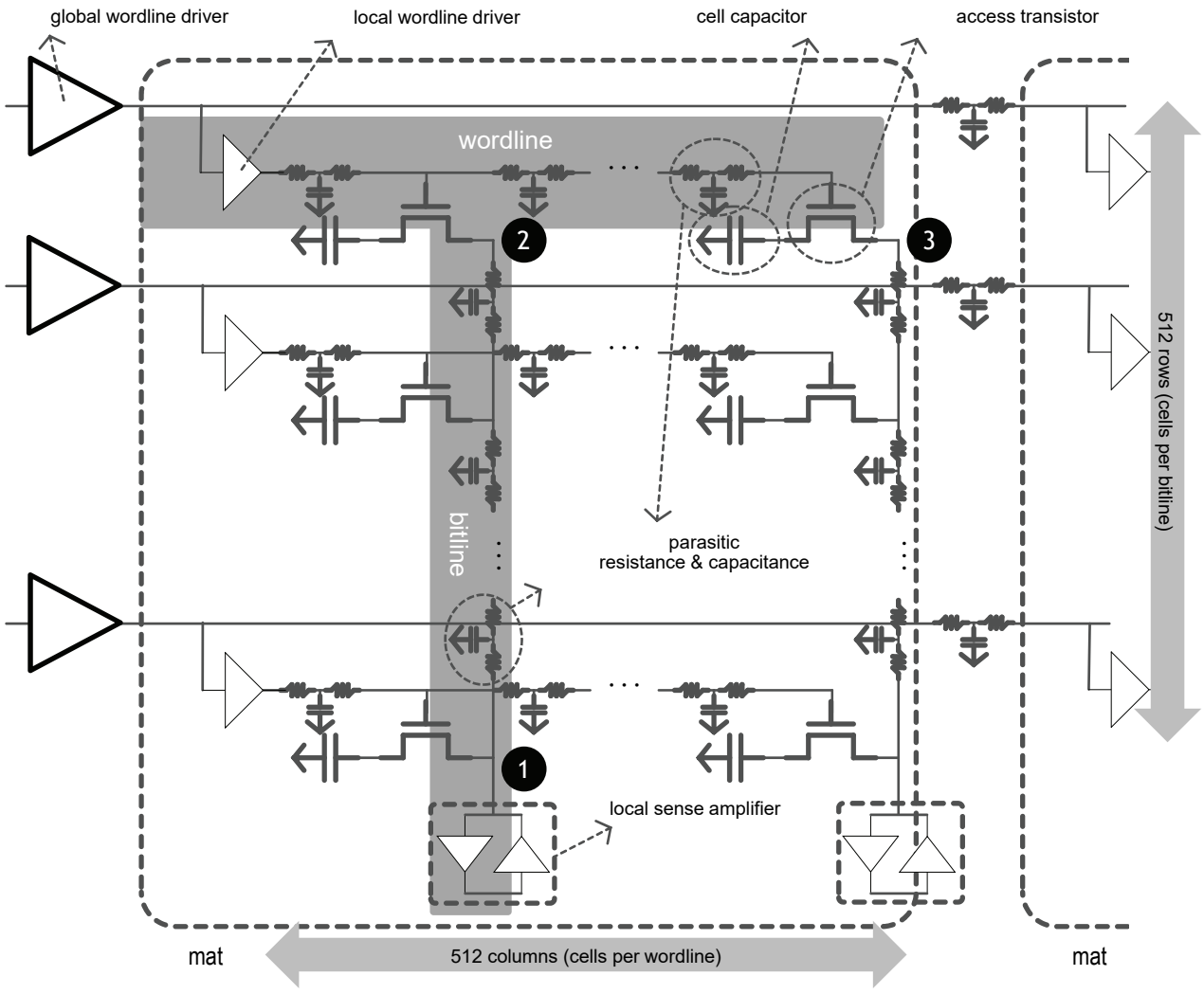
Component	Parameters
Processor	8 cores, 3.2GHz, 3-wide issue, 8 MSHRs/core, 128-entry inst. window
Last-level cache	64B cache-line, 16-way associative, 512KB private cache-slice per core
Mem. Controller	64/64-entry read/write queues, FR-FCFS [77, 104]
Memory system	DDR3-1600 [31], 2 channels, 2 ranks-per-channel

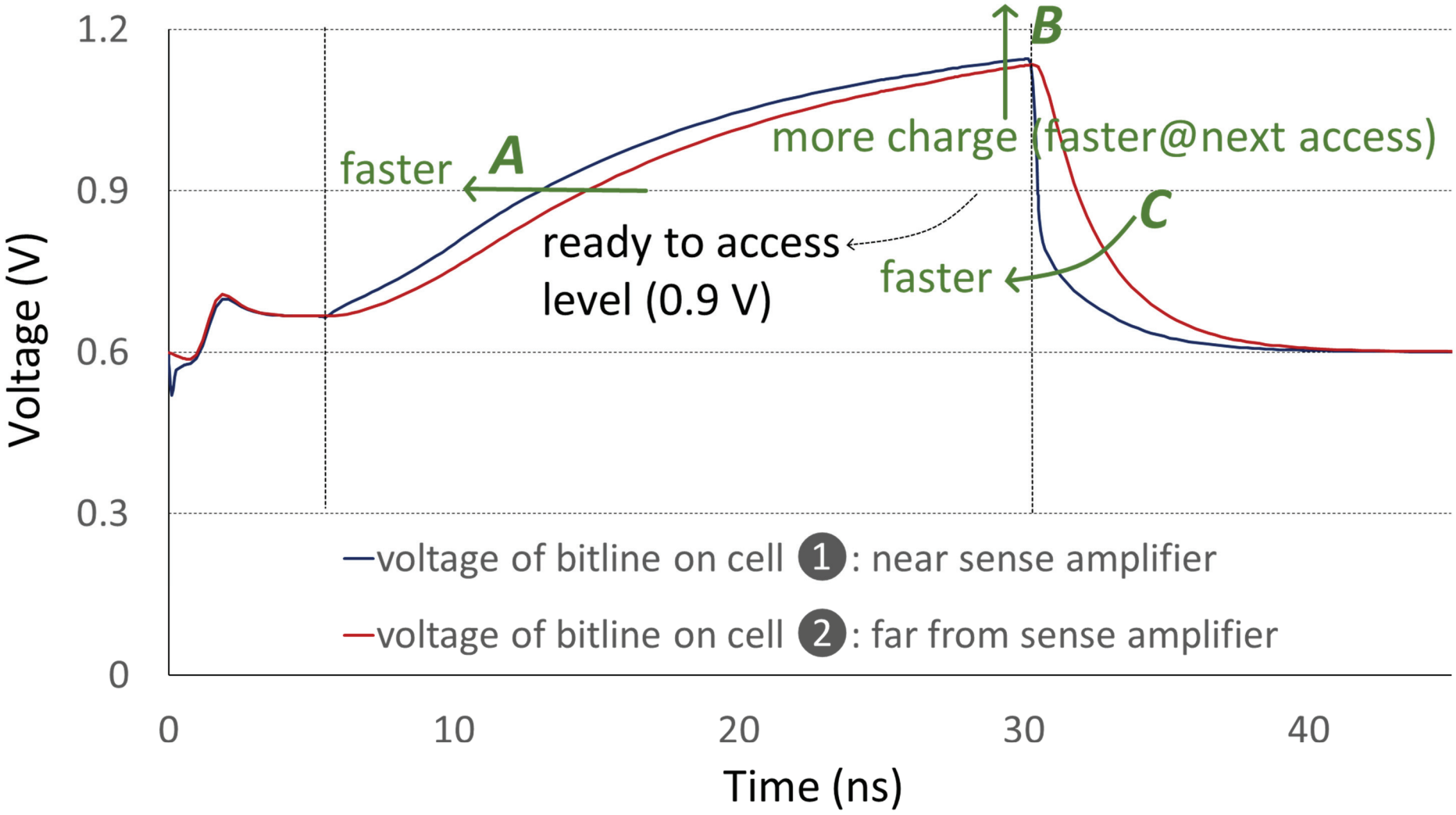

Erroneous Request Count
4000
3000
2000
1000
0
256ms
128ms
64ms
0
64
128
192
256
320
384
448
Row Address (modulo 512)

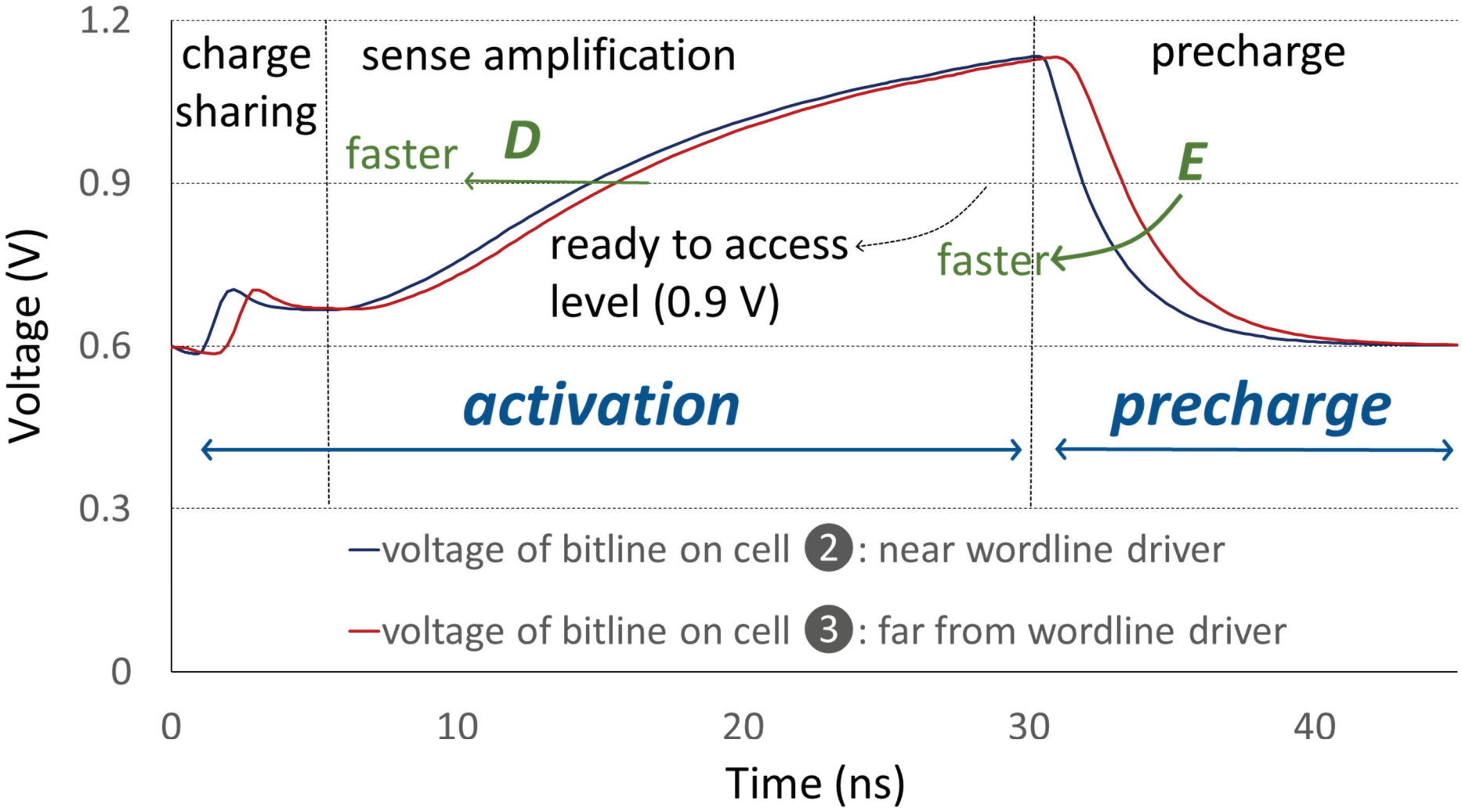

Erroneous Request Count
4000
3000
2000
1000
0
85°C
65°C
45°C
0
64
128
192
256
320
384
448
Row Address (modulo 512)


10K
1K
100
10
1
1
Error Count Ratio
vendor A
vendor B
vendor C
Tested DIMMs


10K
1K
100
10
1
Error Count Ratio
vendor A
vendor B
vendor C
Tested DIMMs


global wordline driver
local wordline driver
cell capacitor
access transistor
wordline
bitline
parasitic
resistance & capacitance
local sense amplifier
1
2
3
512 rows (cells per bitline)
mat
512 columns (cells per wordline)
mat


1.2
0.9
0.6
0.3
0
Voltage (V)
0
10
20
30
40
Time (ns)
A
B
C
faster
more charge (faster@next access)
ready to access level (0.9 V)
faster
voltage of bitline on cell 1 : near sense amplifier
voltage of bitline on cell 2 : far from sense amplifier


1.2
0.9
0.6
0.3
0
Voltage (V)
charge sharing
sense amplification
precharge
faster
D
E
faster
ready to access level (0.9 V)
activation
precharge
voltage of bitline on cell 2: near wordline driver
voltage of bitline on cell 3: far from wordline driver
0
10
20
30
40
Time (ns)