Memory Systems
and Memory-Centric Computing Systems
Lecture 5, Topic 4: Low-Latency Memory

Prof. Onur Mutlu

omutlu@gmail.com

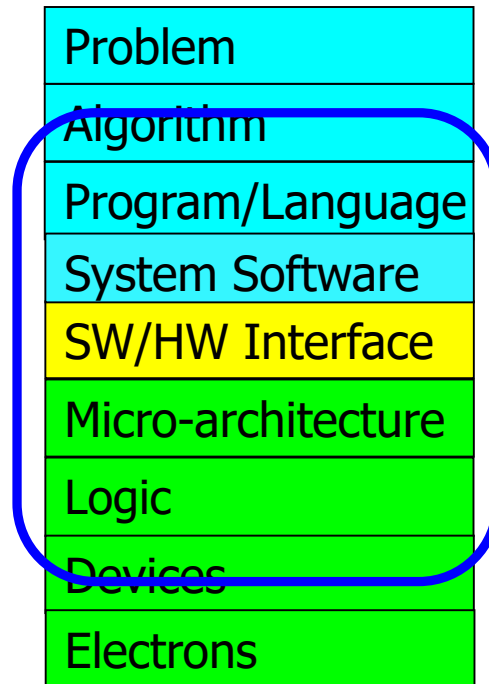https://people.inf.ethz.ch/omutlu

13 July 2018

HiPEAC ACACES Summer School 2018

Systems@ETH Zürich

SAFARI

ETHzürich

Carnegie Mellon
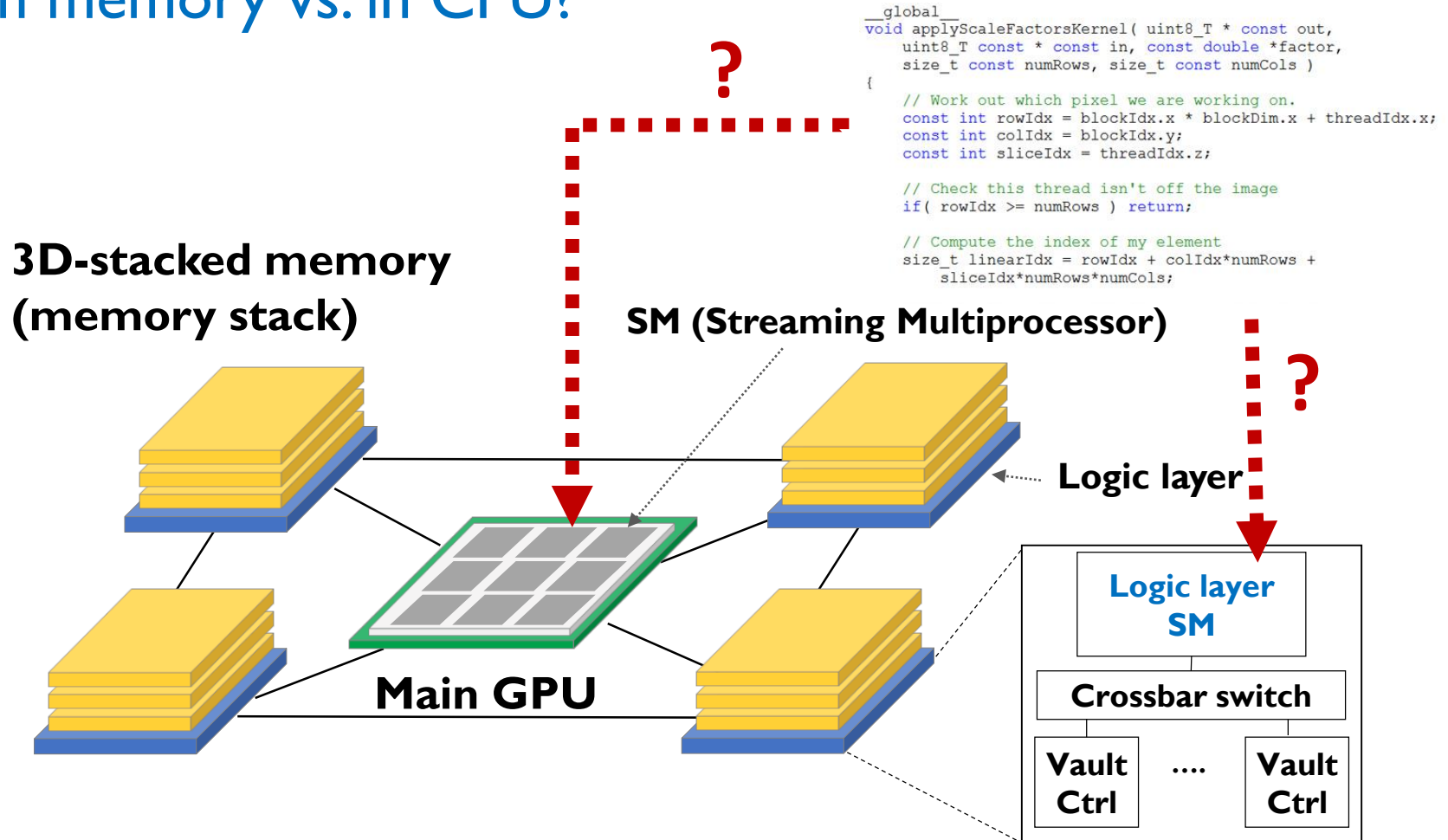
How to Enable Adoption of Processing in Memory

Barriers to Adoption of PIM

1. Functionality of and applications for PIM

2. Ease of programming (interfaces and compiler/HW support)

3. System support: coherence & virtual memory

4. Runtime systems for adaptive scheduling, data mapping, access/sharing control

5. Infrastructures to assess benefits and feasibility

We Need to Revisit the Entire Stack

Problem
Algorithm
Program/Language
System Software
SW/HW Interface
Micro-architecture
Logic
Devices
Electrons

Key Challenge 1: Code Mapping

- **Challenge 1:** Which operations should be executed in memory vs. in CPU?

```
__global__
void applyScaleFactorsKernel( uint8_T * const out,
    uint8_T const * const in, const double *factor,
    size_t const numRows, size_t const numCols )
{

    // Work out which pixel we are working on.
    const int rowIdx = blockIdx.x * blockDim.x + threadIdx.x;
    const int colIdx = blockIdx.y;
    const int sliceIdx = threadIdx.z;

    // Check this thread isn't off the image
    if( rowIdx >= numRows ) return;

    // Compute the index of my element
    size_t linearIdx = rowIdx + colIdx*numRows +
        sliceIdx*numRows*numCols;
```

3D-stacked memory (memory stack)

SM (Streaming Multiprocessor)

Logic layer

Main GPU

Logic layer SM

Crossbar switch

Vault Ctrl .... **Vault Ctrl**

Key Challenge 2: Data Mapping

- **Challenge 2:** How should data be mapped to different 3D memory stacks?
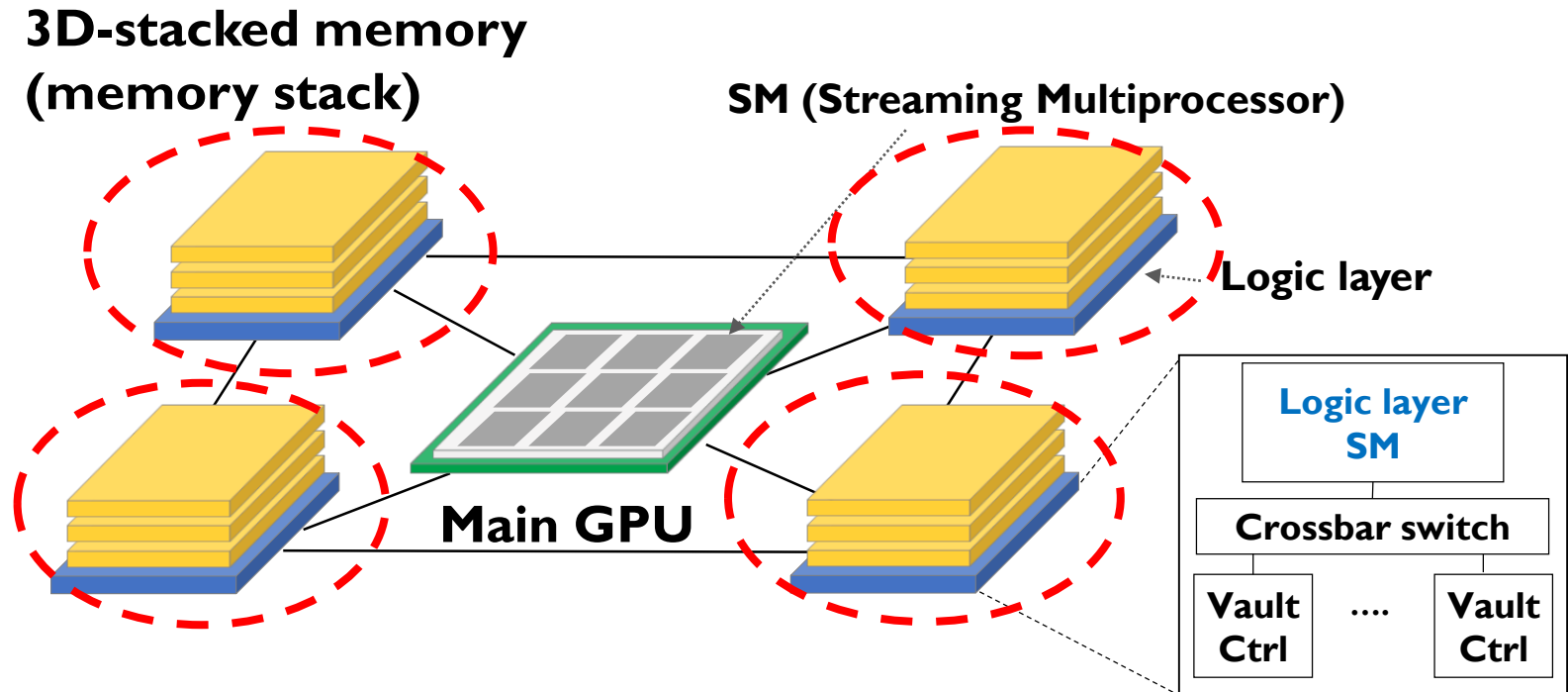
How to Do the Code and Data Mapping?

- Kevin Hsieh, Eiman Ebrahimi, Gwangsun Kim, Niladrish Chatterjee, Mike O'Connor, Nandita Vijaykumar, Onur Mutlu, and Stephen W. Keckler,
"Transparent Offloading and Mapping (TOM): Enabling Programmer-Transparent Near-Data Processing in GPU Systems"
Proceedings of the 43rd International Symposium on Computer Architecture (**ISCA**), Seoul, South Korea, June 2016.
[Slides (pptx) (pdf)]
[Lightning Session Slides (pptx) (pdf)]

Transparent Offloading and Mapping (TOM):
Enabling Programmer-Transparent Near-Data Processing in GPU Systems

Kevin Hsieh[‡]   Eiman Ebrahimi[†]   Gwangsun Kim[*]   Niladrish Chatterjee[†]   Mike O'Connor[†]
Nandita Vijaykumar[‡]   Onur Mutlu[§‡]   Stephen W. Keckler[†]

[‡]**Carnegie Mellon University**   [†]**NVIDIA**   [*]**KAIST**   [§]**ETH Zürich**

How to Schedule Code?

- Ashutosh Pattnaik, Xulong Tang, Adwait Jog, Onur Kayiran, Asit K. Mishra, Mahmut T. Kandemir, <u>Onur Mutlu</u>, and Chita R. Das,
"Scheduling Techniques for GPU Architectures with Processing-In-Memory Capabilities"
Proceedings of the <u>25th International Conference on Parallel Architectures and Compilation Techniques</u> (**PACT**), Haifa, Israel, September 2016.

Scheduling Techniques for GPU Architectures with Processing-In-Memory Capabilities

Ashutosh Pattnaik[1]   Xulong Tang[1]   Adwait Jog[2]   Onur Kayıran[3]
Asit K. Mishra[4]   Mahmut T. Kandemir[1]   Onur Mutlu[5,6]   Chita R. Das[1]

[1]Pennsylvania State University   [2]College of William and Mary
[3]Advanced Micro Devices, Inc.   [4]Intel Labs   [5]ETH Zürich   [6]Carnegie Mellon University

Challenge: Coherence for Hybrid CPU-PIM Apps
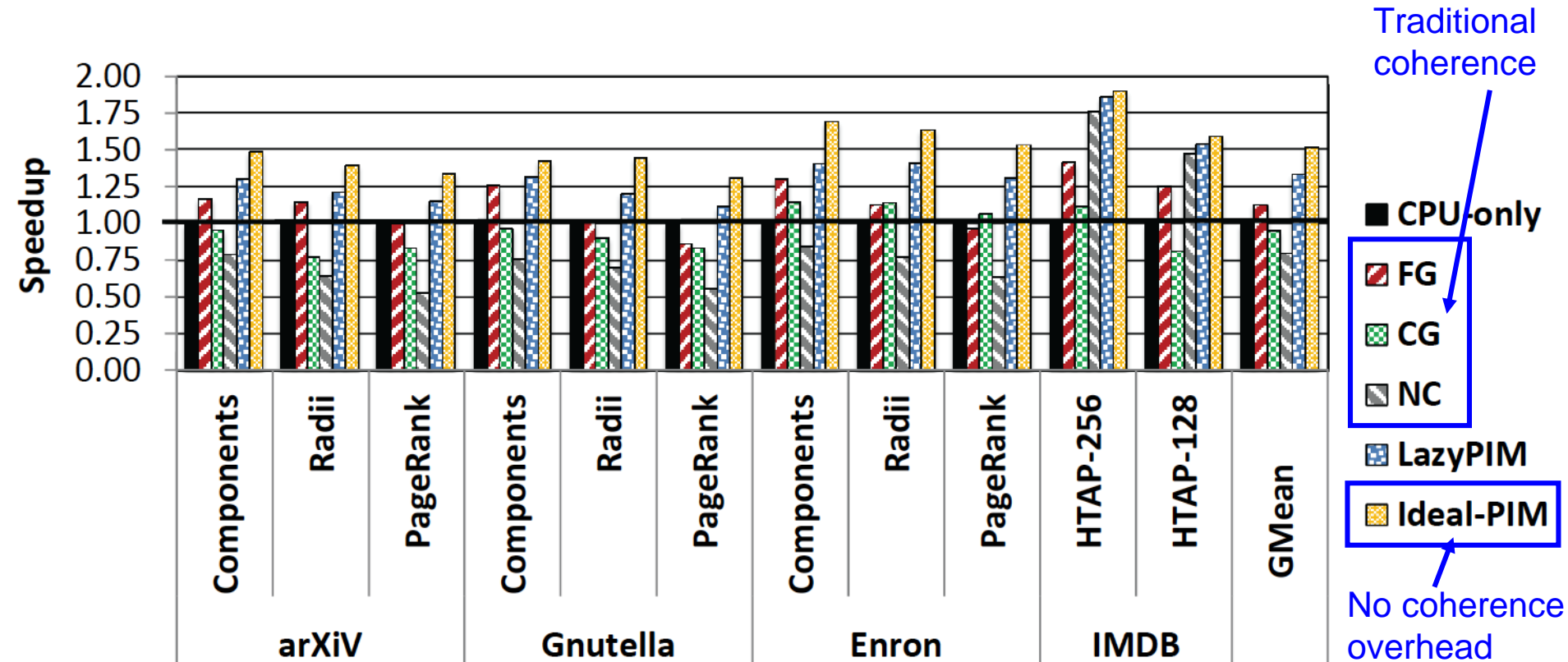
How to Maintain Coherence?

- Amirali Boroumand, Saugata Ghose, Minesh Patel, Hasan Hassan, Brandon Lucia, Kevin Hsieh, Krishna T. Malladi, Hongzhong Zheng, and Onur Mutlu,
  "LazyPIM: An Efficient Cache Coherence Mechanism for Processing-in-Memory"
  IEEE Computer Architecture Letters (**CAL**), June 2016.

LazyPIM: An Efficient Cache Coherence Mechanism for Processing-in-Memory

Amirali Boroumand[†], Saugata Ghose[†], Minesh Patel[†], Hasan Hassan[†§], Brandon Lucia[†],
Kevin Hsieh[†], Krishna T. Malladi[*], Hongzhong Zheng[*], and Onur Mutlu[‡†]

[†]*Carnegie Mellon University*   [*]*Samsung Semiconductor, Inc.*   [§]*TOBB ETÜ*   [‡]*ETH Zürich*

How to Support Virtual Memory?

- Kevin Hsieh, Samira Khan, Nandita Vijaykumar, Kevin K. Chang, Amirali Boroumand, Saugata Ghose, and Onur Mutlu,
"Accelerating Pointer Chasing in 3D-Stacked Memory: Challenges, Mechanisms, Evaluation"
Proceedings of the *34th IEEE International Conference on Computer Design* (**ICCD**), Phoenix, AZ, USA, October 2016.

Accelerating Pointer Chasing in 3D-Stacked Memory: Challenges, Mechanisms, Evaluation

Kevin Hsieh[†]   Samira Khan[‡]   Nandita Vijaykumar[†]

Kevin K. Chang[†]   Amirali Boroumand[†]   Saugata Ghose[†]   Onur Mutlu[§†]

[†]*Carnegie Mellon University*   [‡]*University of Virginia*   [§]*ETH Zürich*

How to Design Data Structures for PIM?

- Zhiyu Liu, Irina Calciu, Maurice Herlihy, and Onur Mutlu,
  "Concurrent Data Structures for Near-Memory Computing"
  Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures (**SPAA**), Washington, DC, USA, July 2017.
  [Slides (pptx) (pdf)]

Concurrent Data Structures for Near-Memory Computing

Zhiyu Liu
Computer Science Department
Brown University
zhiyu_liu@brown.edu

Irina Calciu
VMware Research Group
icalciu@vmware.com

Maurice Herlihy
Computer Science Department
Brown University
mph@cs.brown.edu

Onur Mutlu
Computer Science Department
ETH Zürich
onur.mutlu@inf.ethz.ch

Simulation Infrastructures for PIM

- **Ramulator** extended for PIM
  - ❑ Flexible and extensible DRAM simulator
  - ❑ Can model many different memory standards and proposals
  - ❑ Kim+, "**Ramulator: A Flexible and Extensible DRAM Simulator**", IEEE CAL 2015.
  - ❑ https://github.com/CMU-SAFARI/ramulator

Ramulator: A Fast and Extensible DRAM Simulator

Yoongu Kim[1]   Weikun Yang[1,2]   Onur Mutlu[1]
[1]Carnegie Mellon University   [2]Peking University

An FPGA-based Test-bed for PIM?

- Hasan Hassan et al., **SoftMC: A Flexible and Practical Open-Source Infrastructure for Enabling Experimental DRAM Studies** HPCA 2017.

- **Flexible**
- **Easy to Use (C++ API)**
- **Open-source**

  github.com/CMU-SAFARI/SoftMC
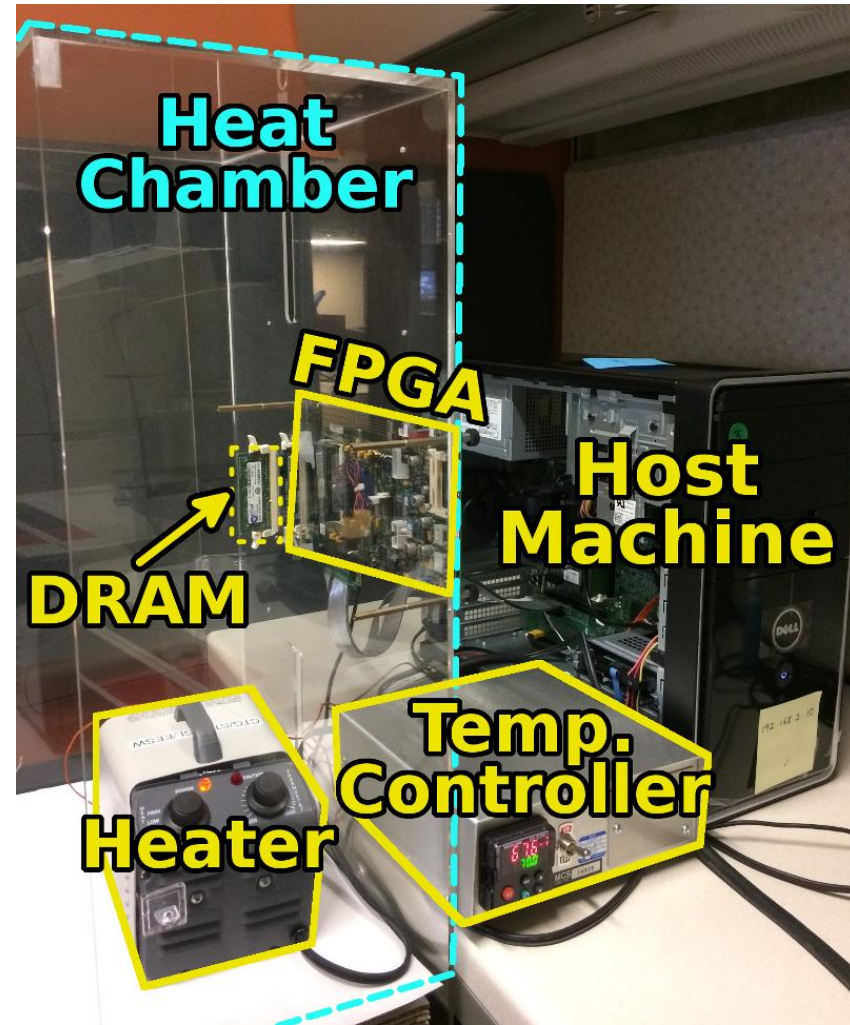
New Applications and Use Cases for PIM

- Jeremie S. Kim, Damla Senol Cali, Hongyi Xin, Donghyuk Lee, Saugata Ghose, Mohammed Alser, Hasan Hassan, Oguz Ergin, Can Alkan, and Onur Mutlu,
"GRIM-Filter: Fast Seed Location Filtering in DNA Read Mapping Using Processing-in-Memory Technologies"
BMC Genomics, 2018.
Proceedings of the 16th Asia Pacific Bioinformatics Conference (**APBC**), Yokohama, Japan, January 2018.
arxiv.org Version (pdf)

GRIM-Filter: Fast seed location filtering in DNA read mapping using processing-in-memory technologies

Jeremie S. Kim[1,6*], Damla Senol Cali[1], Hongyi Xin[2], Donghyuk Lee[3], Saugata Ghose[1], Mohammed Alser[4], Hasan Hassan[6], Oguz Ergin[5], Can Alkan[4*] and Onur Mutlu[6,1*]

Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks

Amirali Boroumand

Saugata Ghose, Youngsok Kim, Rachata Ausavarungnirun, Eric Shiu, Rahul Thakur, Daehyun Kim, Aki Kuusela, Allan Knies, Parthasarathy Ranganathan, Onur Mutlu

Executive Summary

- **Genome Read Mapping** is a very important problem and is the first step in many types of genomic analysis
  - Could lead to improved health care, medicine, quality of life

- Read mapping is an **approximate string matching** problem
  - Find the best fit of 100 character strings into a 3 billion character dictionary
  - **Alignment** is currently the best method for determining the similarity between two strings, but is **very expensive**

- We propose an in-memory processing algorithm **GRIM-Filter** for accelerating read mapping, by reducing the number of required alignments

- We implement GRIM-Filter using **in-memory processing** within **3D-stacked memory** and show up to **3.7x speedup**.

GRIM-Filter in 3D-stacked DRAM
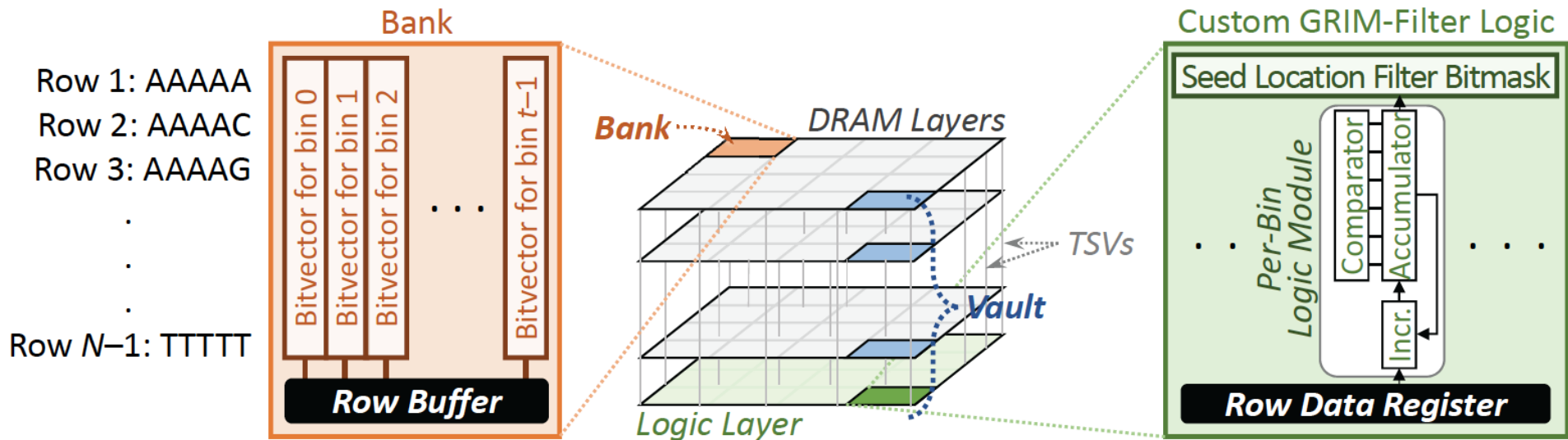
Figure 7: *Left block:* GRIM-Filter bitvector layout within a DRAM bank. *Center block:* 3D-stacked DRAM with tightly integrated logic layer stacked underneath with TSVs for a high intra-DRAM data transfer bandwidth. *Right block:* Custom GRIM-Filter logic placed in the logic layer.
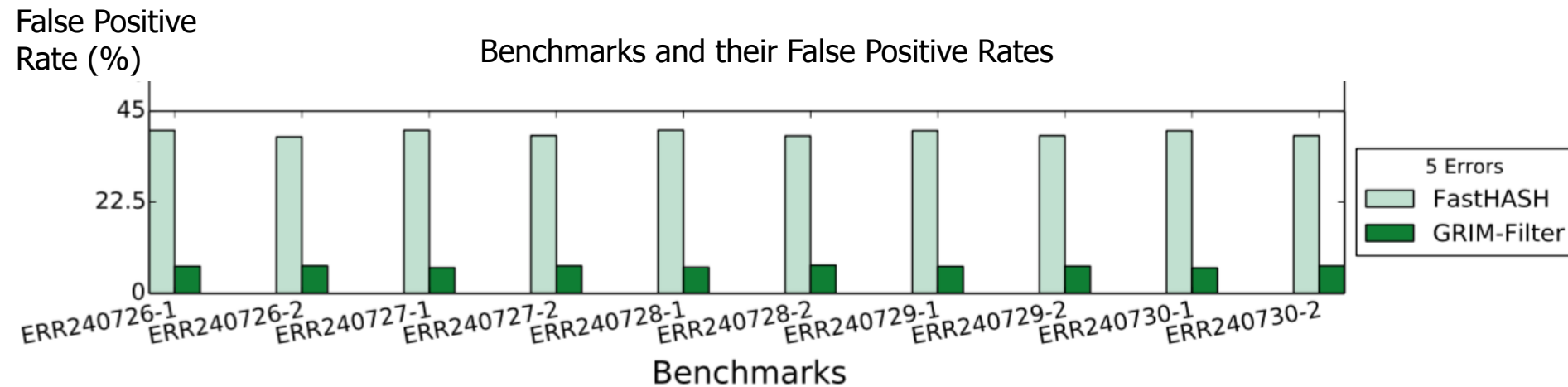
- The layout of bit vectors in a bank enables filtering many bins in parallel
- Customized logic for accumulation and comparison per genome segment
  - Low area overhead, simple implementation

GRIM-Filter Performance

Time (x1000 seconds)

Benchmarks and their Execution Times

1.8x-3.7x performance benefit across real data sets

SAFARI

GRIM-Filter False Positive Rate

False Positive Rate (%)

Benchmarks and their False Positive Rates

5.6x-6.4x False Positive reduction across real data sets

Conclusions

- We propose an in memory filter algorithm to accelerate end-to-end genome read mapping by reducing the number of required alignments

- Compared to the previous best filter
  - We observed 1.8x-3.7x speedup
  - We observed 5.6x-6.4x fewer false positives

- GRIM-Filter is a universal filter that can be applied to any genome read mapper

SAFARI

In-Memory DNA Sequence Analysis

- Jeremie S. Kim, Damla Senol Cali, Hongyi Xin, Donghyuk Lee, Saugata Ghose, Mohammed Alser, Hasan Hassan, Oguz Ergin, Can Alkan, and Onur Mutlu,
"GRIM-Filter: Fast Seed Location Filtering in DNA Read Mapping Using Processing-in-Memory Technologies"
BMC Genomics, 2018.
Proceedings of the 16th Asia Pacific Bioinformatics Conference (**APBC**), Yokohama, Japan, January 2018.
arxiv.org Version (pdf)

GRIM-Filter: Fast seed location filtering in DNA read mapping using processing-in-memory technologies

Jeremie S. Kim[1,6*], Damla Senol Cali[1], Hongyi Xin[2], Donghyuk Lee[3], Saugata Ghose[1], Mohammed Alser[4], Hasan Hassan[6], Oguz Ergin[5], Can Alkan[4*] and Onur Mutlu[6,1*]

Open Problems: PIM Adoption

Enabling the Adoption of Processing-in-Memory: Challenges, Mechanisms, Future Research Directions

SAUGATA GHOSE, KEVIN HSIEH, AMIRALI BOROUMAND,
RACHATA AUSAVARUNGNIRUN

Carnegie Mellon University

ONUR MUTLU

ETH Zürich and Carnegie Mellon University

https://arxiv.org/pdf/1802.00320.pdf

Enabling the Paradigm Shift

Computer Architecture Today

- You can revolutionize the way computers are built, if you understand both the hardware and the software (and change each accordingly)

- You can invent new paradigms for computation, communication, and storage

- Recommended book: Thomas Kuhn, "The Structure of Scientific Revolutions" (1962)
  - Pre-paradigm science: no clear consensus in the field
  - Normal science: dominant theory used to explain/improve things (business as usual); exceptions considered anomalies
  - Revolutionary science: underlying assumptions re-examined

Computer Architecture Today (IV)

- You can revolutionize the way computers are built, if you understand both the hardware and the software (and change each ac...

- You can ir... communic...

- Recomme... ...ure of Scientific I... REV...
  - Pre-para... ...ield
  - Normal s... things (b... ...improve ...anomalies
  - Revoluti... ...examined

What Will You Learn in This Course?

- **Memory Systems and Memory-Centric Computing Systems**
  - July 9-13, 2018

- Topic 1: Main Memory Trends and Basics
- Topic 2: Memory Reliability & Security: RowHammer and Beyond
- Topic 3: In-memory Computation
- Topic 4: Low-Latency (and Low-Energy) Memory
- Topic 5 (unlikely): Enabling and Exploiting Non-Volatile Memory
- Topic 6 (unlikely): Flash Memory and SSD Scaling

- Major Overview Reading:
  - Mutlu and Subramaniam, "Research Problems and Opportunities in Memory Systems," SUPERFRI 2014.

Agenda

- Brief Introduction
- A Motivating Example
- Memory System Trends
- What Will You Learn In This Course
  - And, how to make the best of it…
- Memory Fundamentals
- Key Memory Challenges and Solution Directions
  - Security, Reliability, Safety
  - Energy and Performance: Data-Centric Systems
  - Latency and Latency-Reliability Tradeoffs
- Summary and Future Lookout

SAFARI

Four Key Directions

- Fundamentally Secure/Reliable/Safe Architectures

- Fundamentally Energy-Efficient Architectures
  - Memory-centric (Data-centric) Architectures

- Fundamentally Low-Latency Architectures

- Architectures for Genomics, Medicine, Health

Maslow's Hierarchy of Needs, A Third Time

Maslow, "A Theory of Human Motivation,"
Psychological Review, 1943.

Maslow, "Motivation and Personality,"
Book, 1954-1970.

Source: https://www.simplypsychology.org/maslow.html
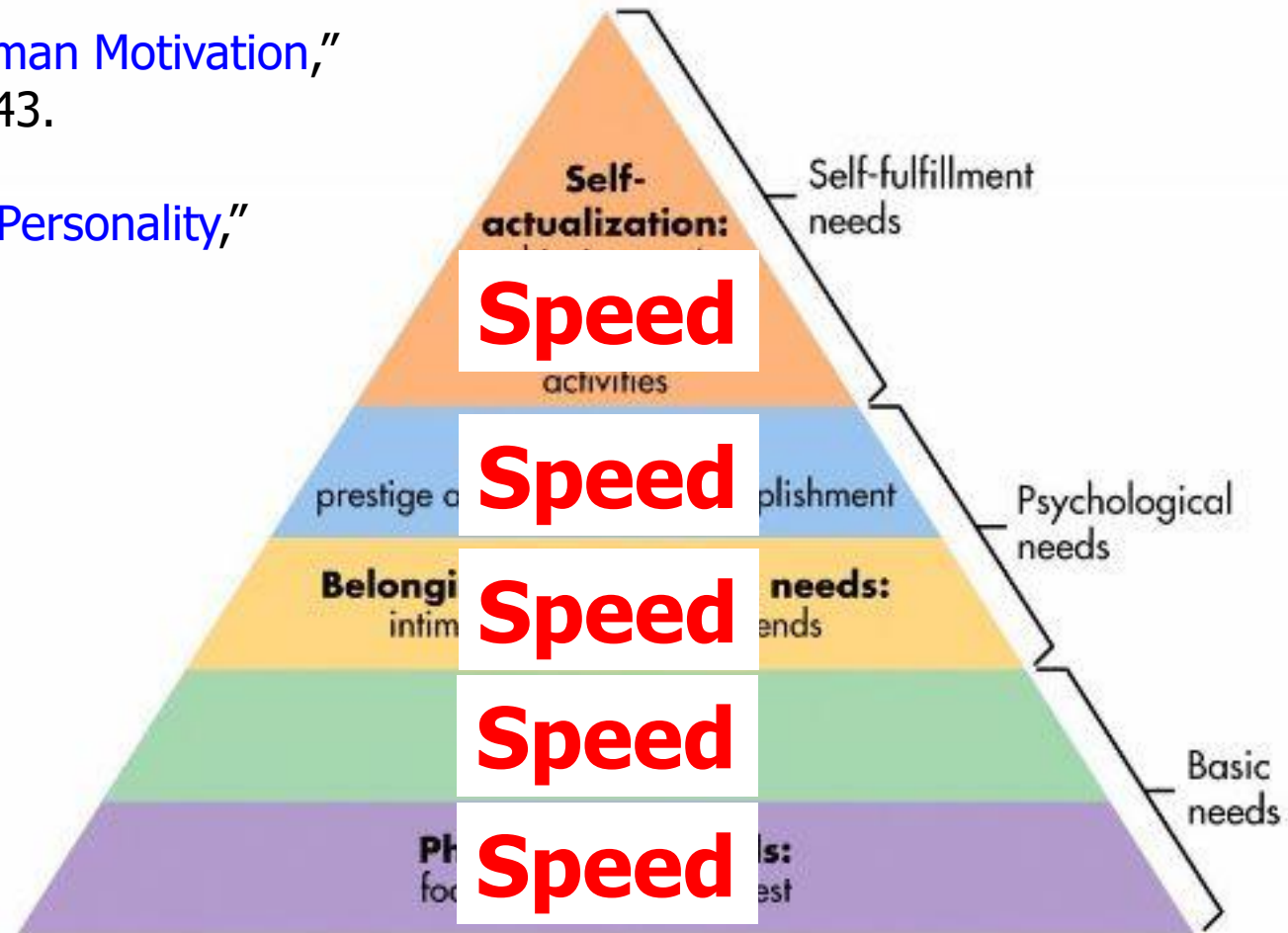
Fundamentally Low-Latency Computing Architectures

Memory Latency: Fundamental Tradeoffs

Review: Memory Latency Lags Behind
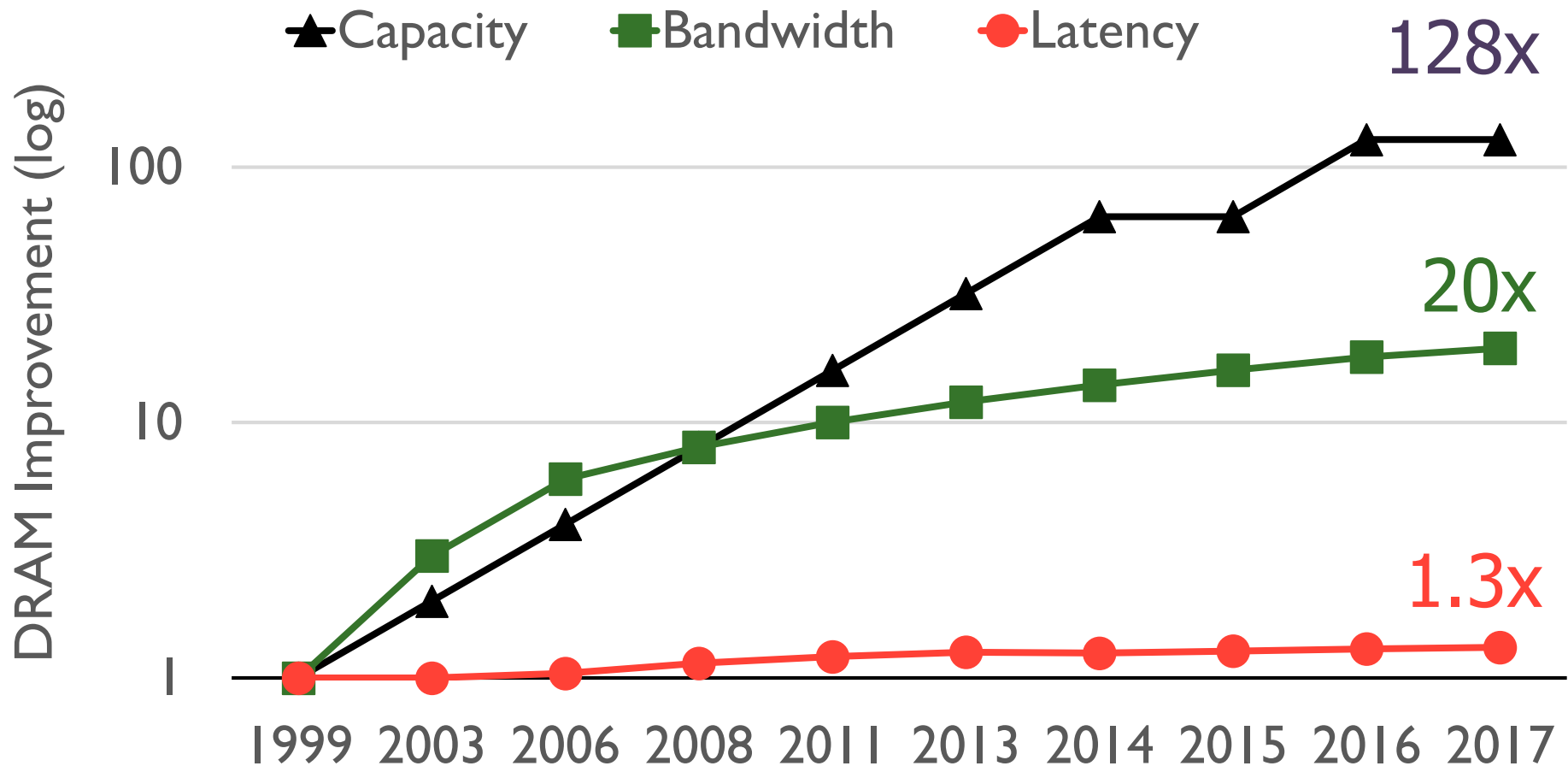
Memory latency remains almost constant
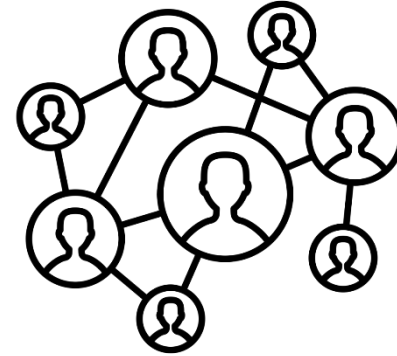
DRAM Latency Is Critical for Performance
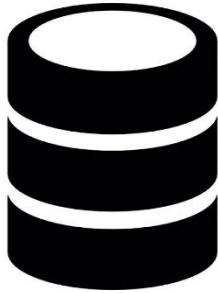
In-memory Databases

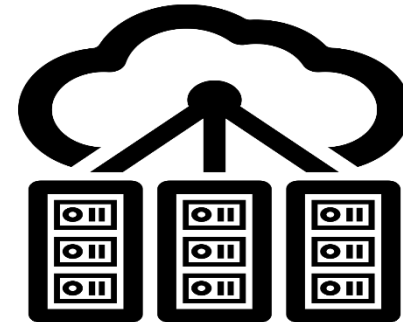[Mao+, EuroSys'12;
 Clapp+ (**Intel**), IISWC'15]

Graph/Tree Processing

[Xu+, IISWC'12; Umuroglu+, FPL'15]

In-Memory Data Analytics

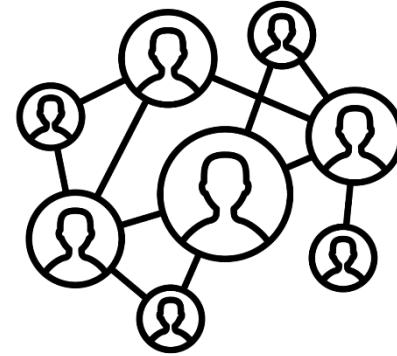[Clapp+ (**Intel**), IISWC'15;
 Awan+, BDCloud'15]

Datacenter Workloads

[Kanev+ (**Google**), ISCA'15]

SAFARI

DRAM Latency Is Critical for Performance
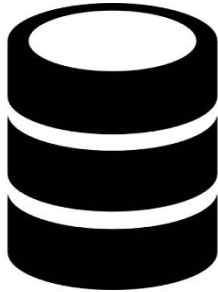
In-memory Databases

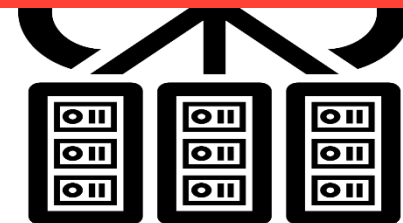**Graph/Tree Processing**

Long memory latency → performance bottleneck

In-Memory Data Analytics
[Clapp+ (**Intel**), IISWC'15;
 Awan+, BDCloud'15]

Datacenter Workloads
[Kanev+ (**Google**), ISCA'15]

APACHE Spark™

SAFARI

The Memory Latency Problem

- High memory latency is a significant limiter of system performance and energy-efficiency

- It is becoming increasingly so with higher memory contention in multi-core and heterogeneous architectures
  - Exacerbating the bandwidth need
  - Exacerbating the QoS problem

- It increases processor design complexity due to the mechanisms incorporated to tolerate memory latency

SAFARI

Retrospective: Conventional Latency Tolerance Techniques

- Caching [initially by Wilkes, 1965]
  - Widely used, simple, effective, but inefficient, passive
  - Not all applications/phases exhibit temporal or spatial locality

- Prefetching [initially in IBM 360/91, 1967]

None of These Fundamentally Reduce Memory Latency

ongoing research effort

- Out-of-order execution [initially by Tomasulo, 1967]
  - Tolerates cache misses that cannot be prefetched
  - Requires extensive hardware resources for tolerating long latencies

Two Major Sources of Latency Inefficiency

- **Modern DRAM is not designed for low latency**
  - Main focus is cost-per-bit (capacity)

- **Modern DRAM latency is determined by worst case conditions and worst case devices**
  - Much of memory latency is unnecessary

Our Goal: Reduce Memory Latency at the Source of the Problem

What Causes
the Long Memory Latency?

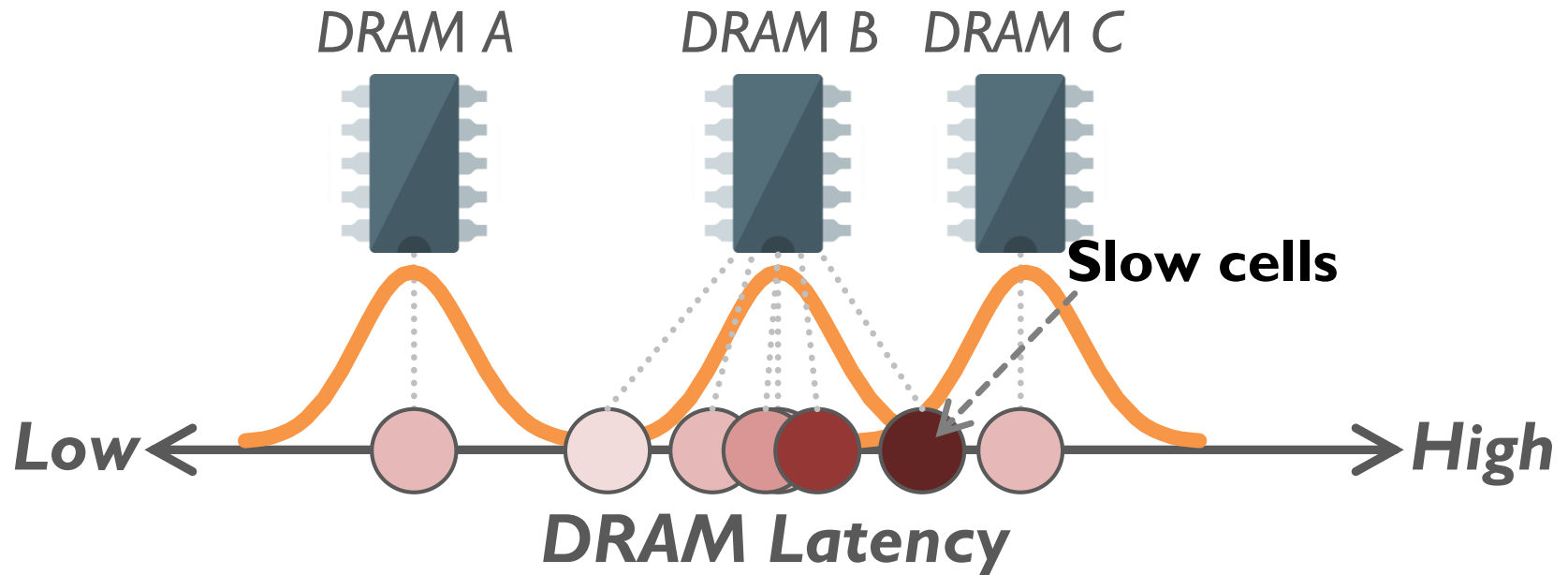# Why the Long Memory Latency?

- **Reason 1: Design of DRAM Micro-architecture**
  - Goal: Maximize capacity/area, not minimize latency

- **Reason 2: "One size fits all" approach to latency specification**
  - Same latency parameters for all temperatures
  - Same latency parameters for all DRAM chips (e.g., rows)
  - Same latency parameters for all parts of a DRAM chip
  - Same latency parameters for all supply voltage levels
  - Same latency parameters for all application data
  - ...

Tackling the Fixed Latency Mindset

- Reliable operation latency is actually very heterogeneous
  - Across temperatures, chips, parts of a chip, voltage levels, …

- Idea: Dynamically find out and use the lowest latency one can reliably access a memory location with
  - Adaptive-Latency DRAM [HPCA 2015]
  - Flexible-Latency DRAM [SIGMETRICS 2016]
  - Design-Induced Variation-Aware DRAM [SIGMETRICS 2017]
  - Voltron [SIGMETRICS 2017]
  - DRAM Latency PUF [HPCA 2018]
  - …

- We would like to find sources of latency heterogeneity and exploit them to minimize latency
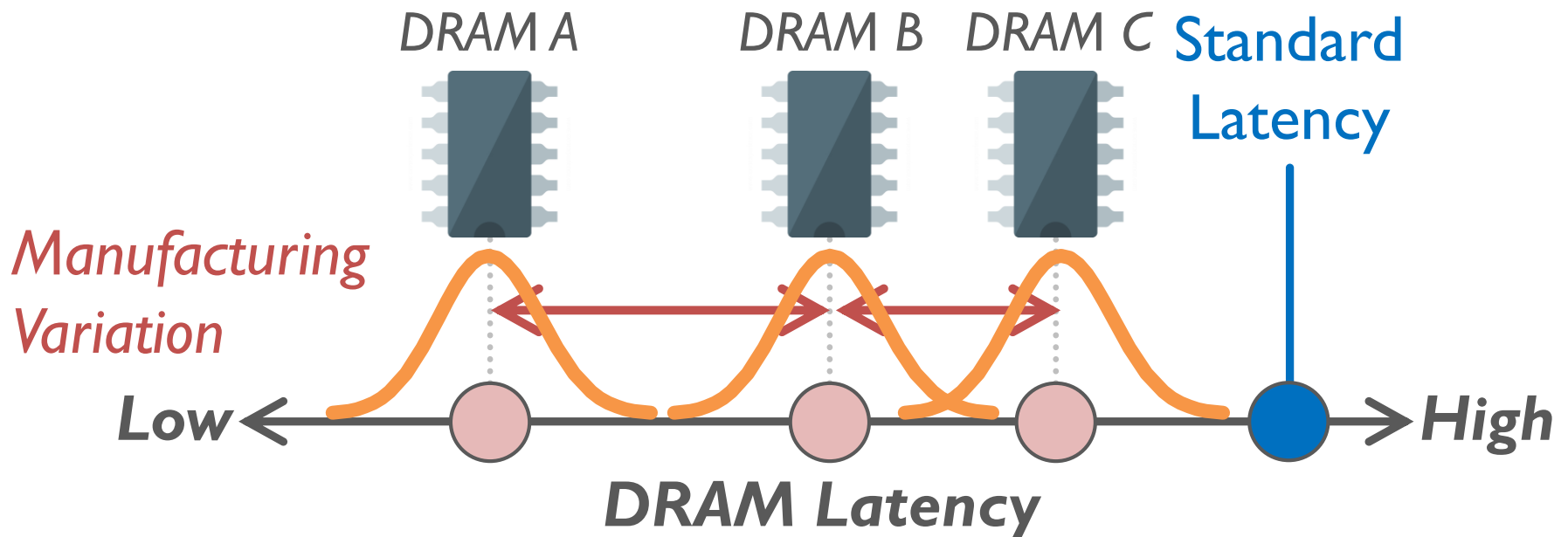
Latency Variation in Memory Chips

Heterogeneous manufacturing & operating conditions →
latency variation in timing parameters

Why is Latency High?

- DRAM latency: Delay as specified in DRAM standards
  - Doesn't reflect true DRAM device latency
- Imperfect manufacturing process → latency variation
- **High standard latency** chosen to increase yield

What Causes the Long Memory Latency?

- **Conservative timing margins!**

- DRAM timing parameters are set to cover the worst case

- Worst-case temperatures
    - 85 degrees vs. common-case
    - to enable a wide range of operating conditions
- Worst-case devices
    - DRAM cell with smallest charge across any acceptable device
    - to tolerate process variation at acceptable yield

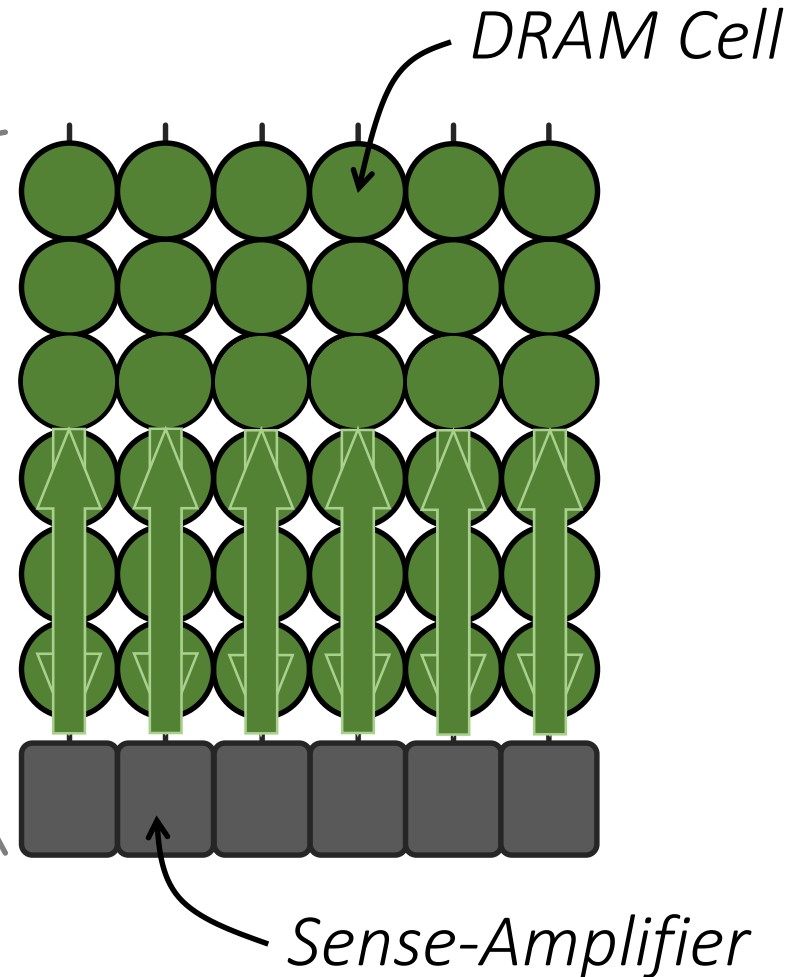- This leads to large timing margins for the common case

Understanding and Exploiting Variation in DRAM Latency
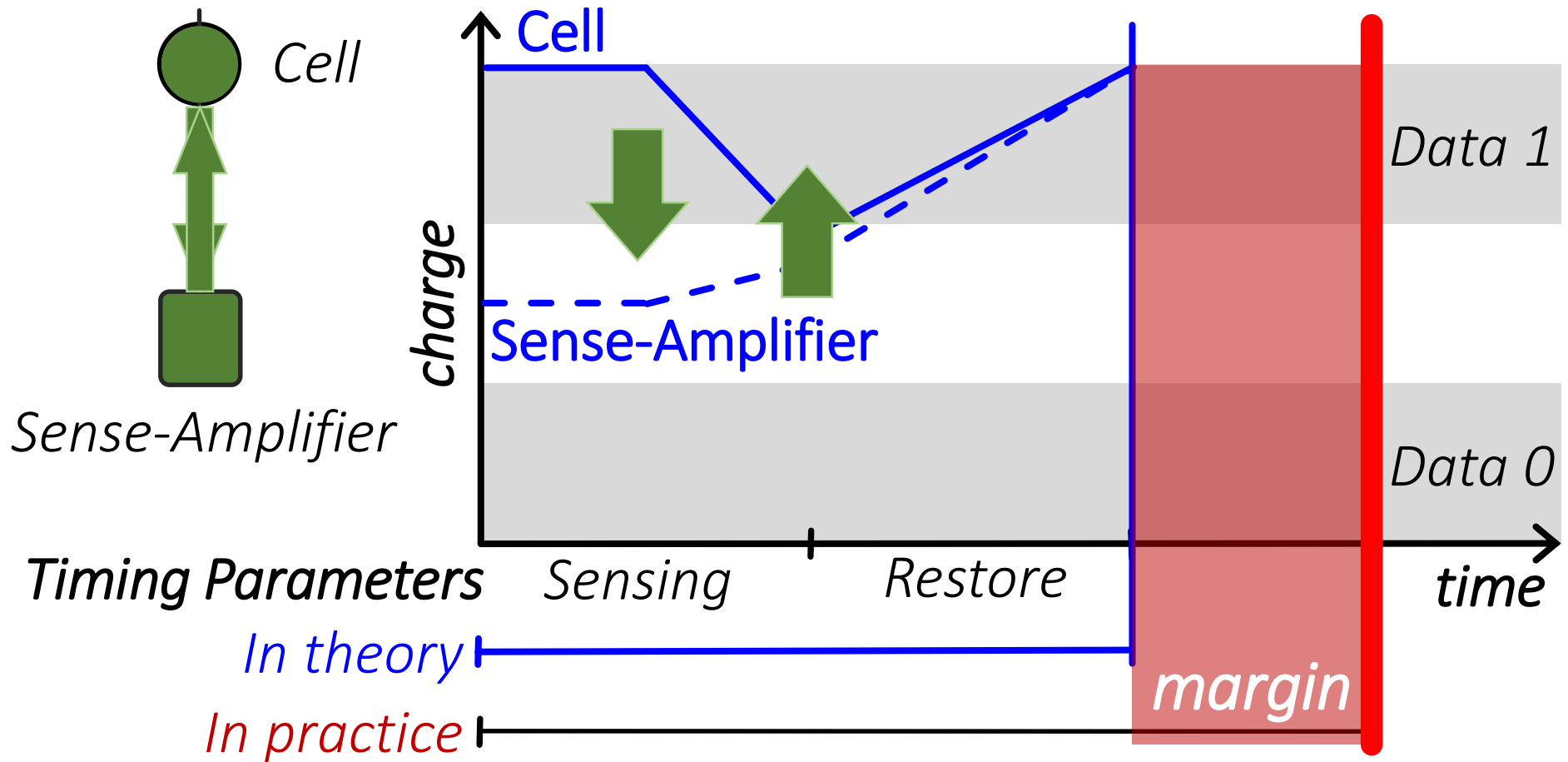
DRAM Stores Data as Charge

DRAM Cell

Three steps of charge movement

1. Sensing
2. Restore
3. Precharge

Sense-Amplifier

DRAM Charge over Time

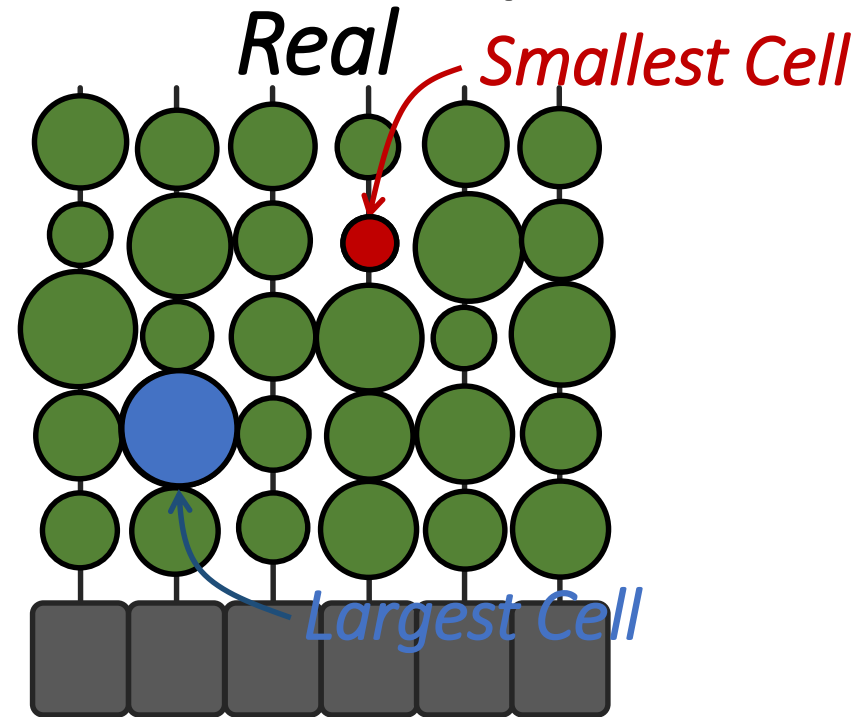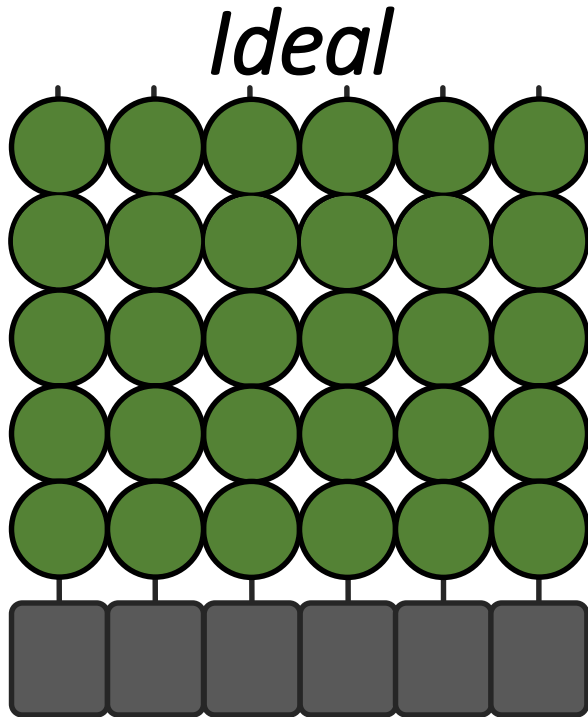*Why does DRAM need the extra timing margin?*

Two Reasons for Timing Margin

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

2. Temperature Dependence

DRAM Cells are Not Equal

Ideal

Real

Smallest Cell

Largest Cell

Same Size ➔   Different Size ➔

Large variation in cell size

Same Charge ➔   Different Charge ➔

Large variation in charge

Same Latency   Different Latency

Large variation in access latency

Process Variation

DRAM Cell

Capacitor

Contact

Bitline

Access Transistor

ACCESS

❶ Cell Capacitance

❷ Contact Resistance

❸ Transistor Performance

Small cell can store small charge

- *Small cell capacitance*
- *High contact resistance*
- *Slow access transistor*

➜ *High access latency*

Two Reasons for Timing Margin

1. Process Variation

- DRAM cells are not equal
- Leads to extra timing margin for a cell that can store a large amount of charge
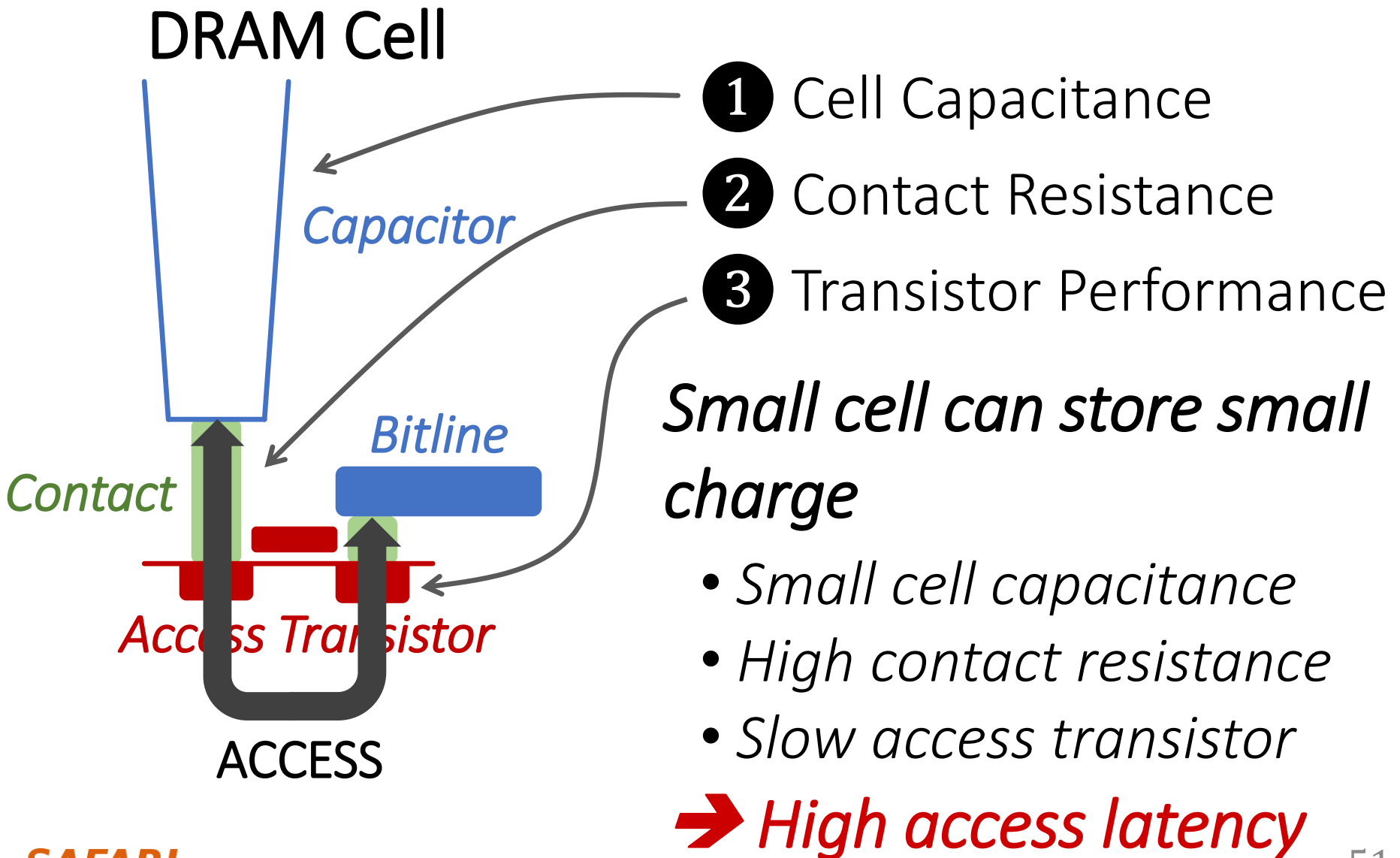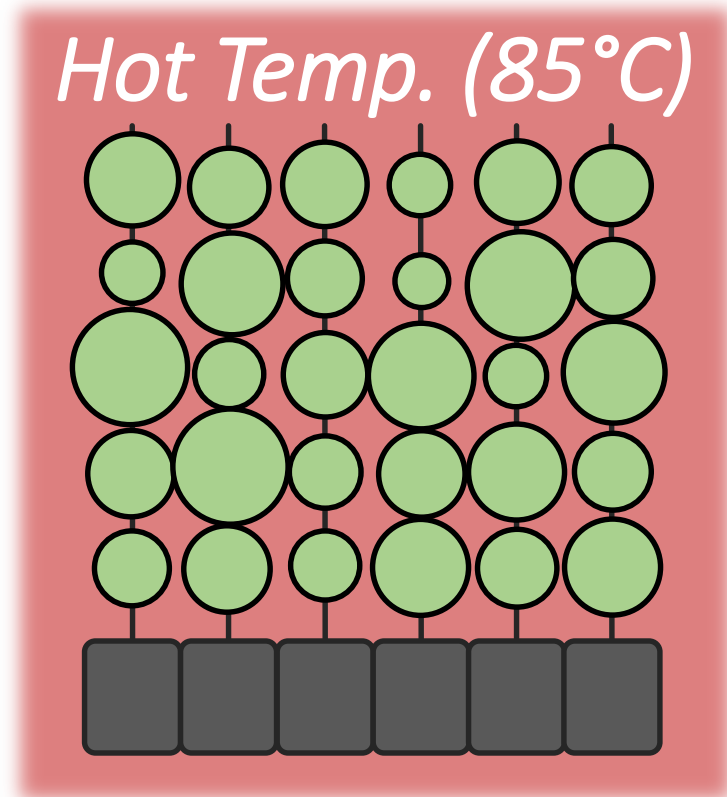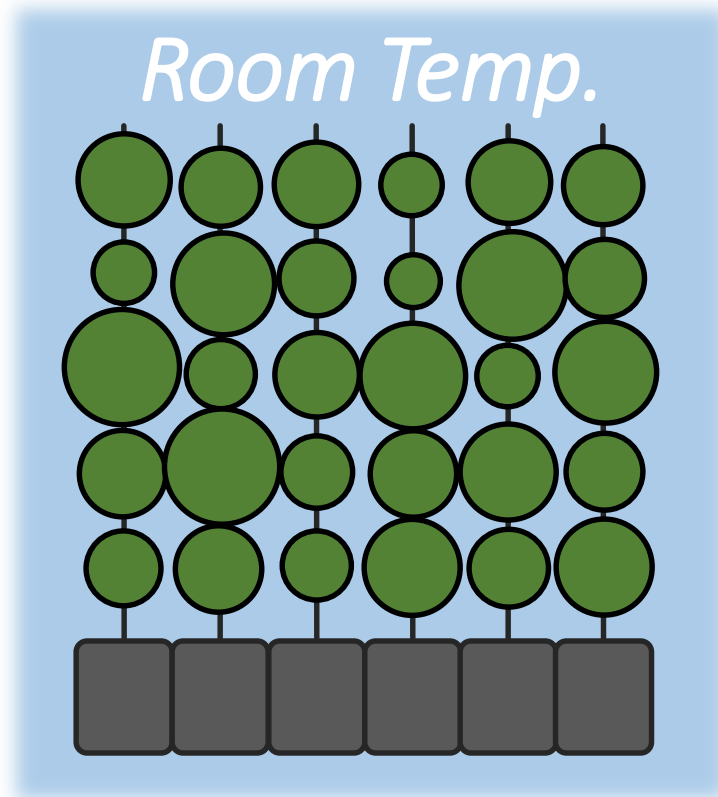
2. Temperature Dependence

- DRAM leaks more charge at higher temperature
- Leads to extra timing margin for cells that operate at low temperature

SAFARI

Charge Leakage Temperature

Room Temp.

Hot Temp. (85°C)

Small Leakage

Large Leakage

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*

Adaptive-Latency DRAM [HPCA 2015]

- Idea: Optimize DRAM timing for the common case
  - Current temperature
  - Current DRAM module

- Why would this reduce latency?

  - A DRAM cell can store much more charge in the common case (low temperature, strong cell) than in the worst case

  - More charge in a DRAM cell
    - → Faster sensing, charge restoration, precharging
    - → Faster access (read, write, refresh, …)

Lee+, "Adaptive-Latency DRAM: Optimizing DRAM Timing for the Common-Case," HPCA 2015.

Extra Charge → Reduced Latency

1. Sensing

Sense **cells with extra charge** faster

→ Lower sensing latency

2. Restore

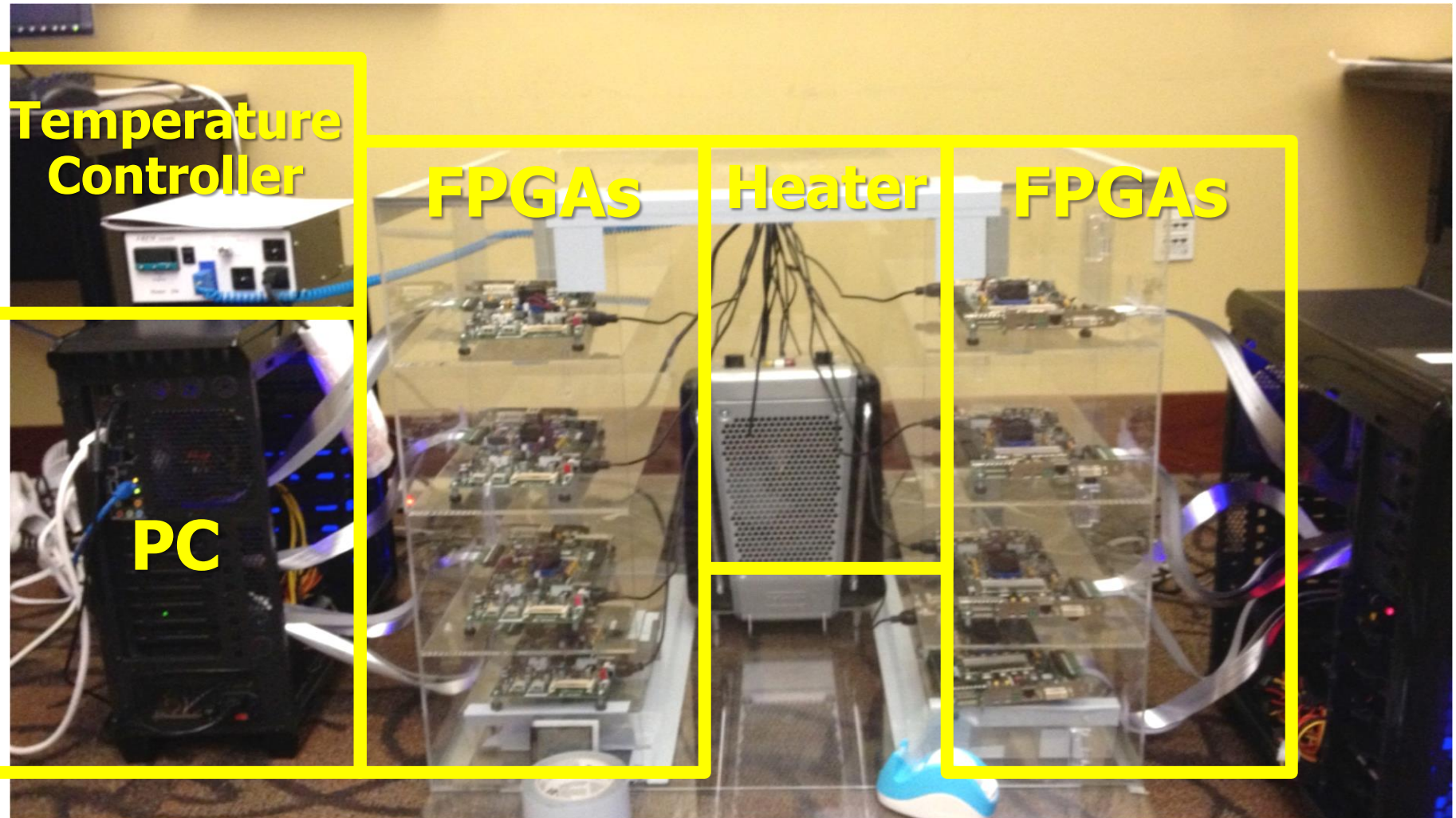No need to fully restore **cells with extra charge**

→ Lower restoration latency

3. Precharge

No need to fully precharge bitlines for **cells with extra charge**

→ Lower precharge latency

DRAM Characterization Infrastructure

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

DRAM Characterization Infrastructure

- Hasan Hassan et al., **SoftMC: A Flexible and Practical Open-Source Infrastructure for Enabling Experimental DRAM Studies**, HPCA 2017.

- **Flexible**
- **Easy to Use (C++ API)**
- **Open-source**

  github.com/CMU-SAFARI/SoftMC
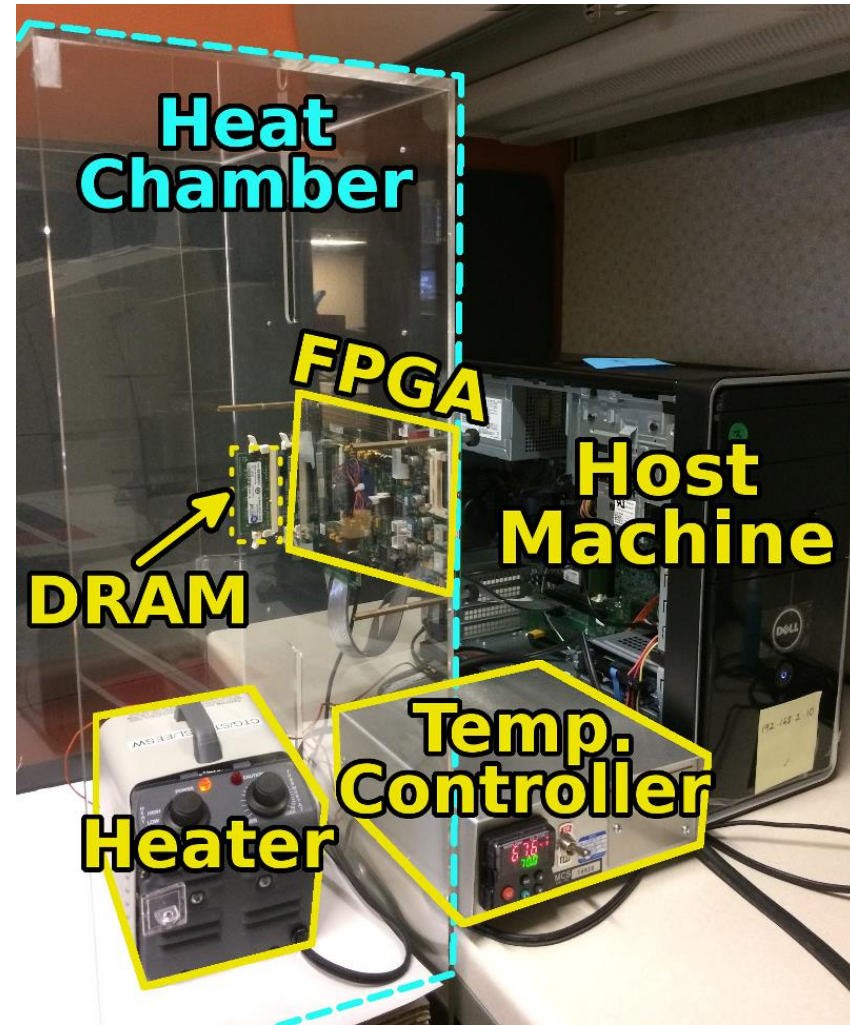
SoftMC: Open Source DRAM Infrastructure

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

SoftMC: A Flexible and Practical Open-Source Infrastructure for Enabling Experimental DRAM Studies

Hasan Hassan[1,2,3]   Nandita Vijaykumar[3]   Samira Khan[4,3]   Saugata Ghose[3]   Kevin Chang[3]
Gennady Pekhimenko[5,3]   Donghyuk Lee[6,3]   Oguz Ergin[2]   Onur Mutlu[1,3]

[1]ETH Zürich   [2]TOBB University of Economics & Technology   [3]Carnegie Mellon University
[4]University of Virginia   [5]Microsoft Research   [6]NVIDIA Research

Observation 1. Faster Sensing

Typical DIMM at Low Temperature

115 DIMM Characterization
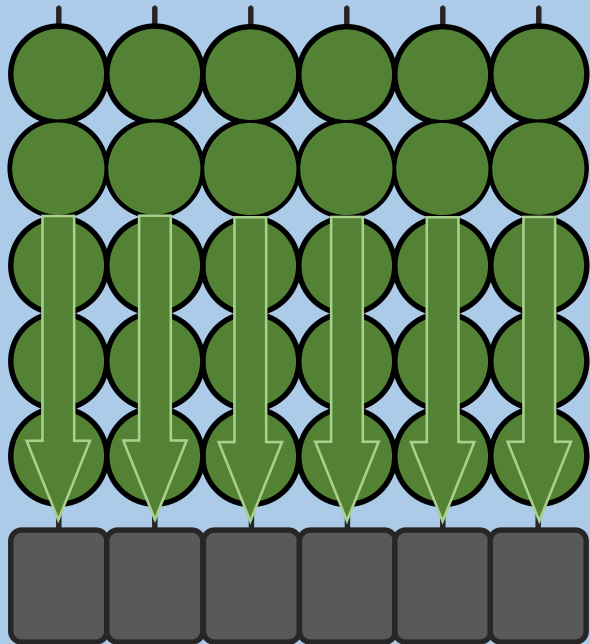
More Charge

Strong Charge Flow
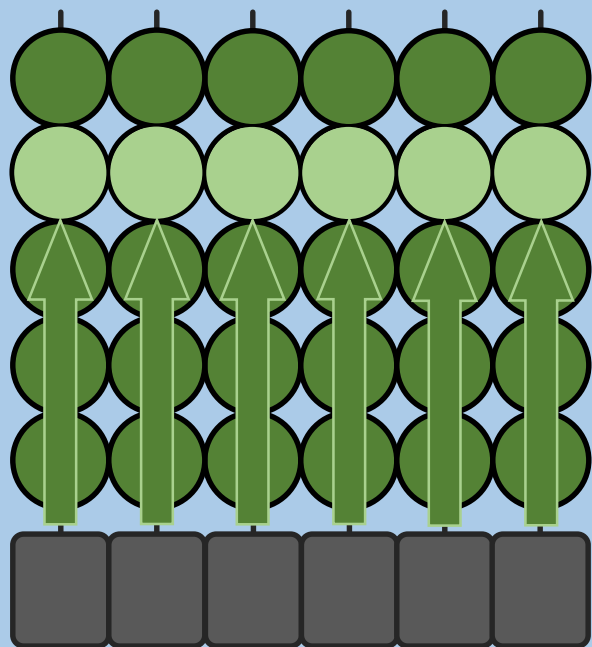
Faster Sensing

Timing (tRCD)

17% ↓

No Errors

Typical DIMM at Low Temperature
➔ *More charge* ➔ *Faster sensing*

Observation 2. Reducing Restore Time

Typical DIMM at Low Temperature

Less Leakage ➜ Extra Charge

No Need to Fully Restore Charge

115 DIMM Characterization

Read ($tRAS$)
37% ↓

Write (tWR)
54% ↓
No Errors

Typical DIMM at lower temperature
➜ *More charge* ➜ *Restore time reduction*

AL-DRAM

- *Key idea*
  - Optimize DRAM timing parameters online

- *Two components*
  - DRAM manufacturer provides multiple sets of reliable DRAM timing parameters at different temperatures for each DIMM
  - System monitors DRAM temperature & uses appropriate DRAM timing parameters

DRAM Temperature

- *DRAM temperature measurement*
  - Server cluster: Operates at under **34℃**
  - Desktop: Operates at under **50℃**
  - *DRAM standard optimized for 85 ℃*

DRAM operates at low temperatures in the common-case

- *Previous works – Maintain low DRAM temperature*
  - David+ ICAC 2011
  - Liu+ ISCA 2007
  - Zhu+ ITHERM 2008

SAFARI

Latency Reduction Summary of 115 DIMMs

- *Latency reduction for read & write (55°C)*
  - *Read Latency: **32.7%***
  - *Write Latency: **55.1%***

- *Latency reduction for each timing parameter (55°C)*
  - *Sensing: **17.3%***
  - *Restore: **37.3%** (read), **54.8%** (write)*
  - *Precharge: **35.2%***

Lee+, "Adaptive-Latency DRAM: Optimizing DRAM Timing for the Common-Case," HPCA 2015.

SAFARI

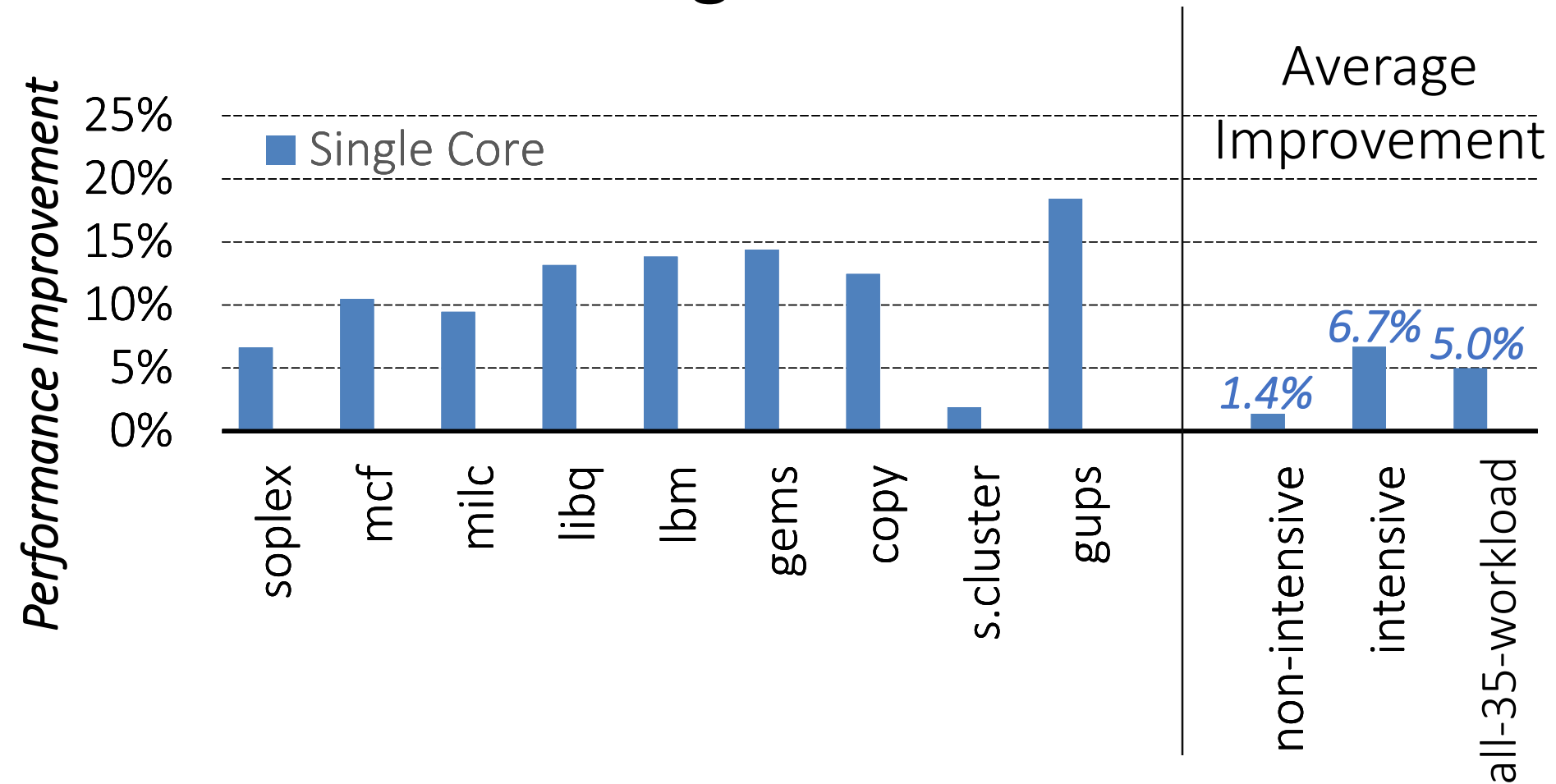# AL-DRAM: Real System Evaluation

- *System*
  - *CPU: AMD 4386 (8 Cores, 3.1GHz, 8MB LLC)*

D18F2x200_dct[0]_mp[1:0] DDR3 DRAM Timing 0

Reset: 0F05_0505h. See 2.9.3 [DCT Configuration Registers].

Bits	Description
31:30	Reserved.
29:24	**Tras: row active strobe**. Read-write. BIOS: See 2.9.7.5 [SPD ROM-Based Configuration]. Specifies the minimum time in memory clock cycles from an activate command to a precharge command, both to the same chip select bank. Bits Description 07h-00h Reserved 2Ah-08h \<Tras\> clocks 3Fh-2Bh Reserved
23:21	Reserved.
20:16	**Trp: row precharge time**. Read-write. BIOS: See 2.9.7.5 [SPD ROM-Based Configuration]. Specifies the minimum time in memory clock cycles from a precharge command to an activate command or auto refresh command, both to the same bank.

AL-DRAM: Single-Core Evaluation

AL-DRAM improves performance on a real system

AL-DRAM: Multi-Core Evaluation
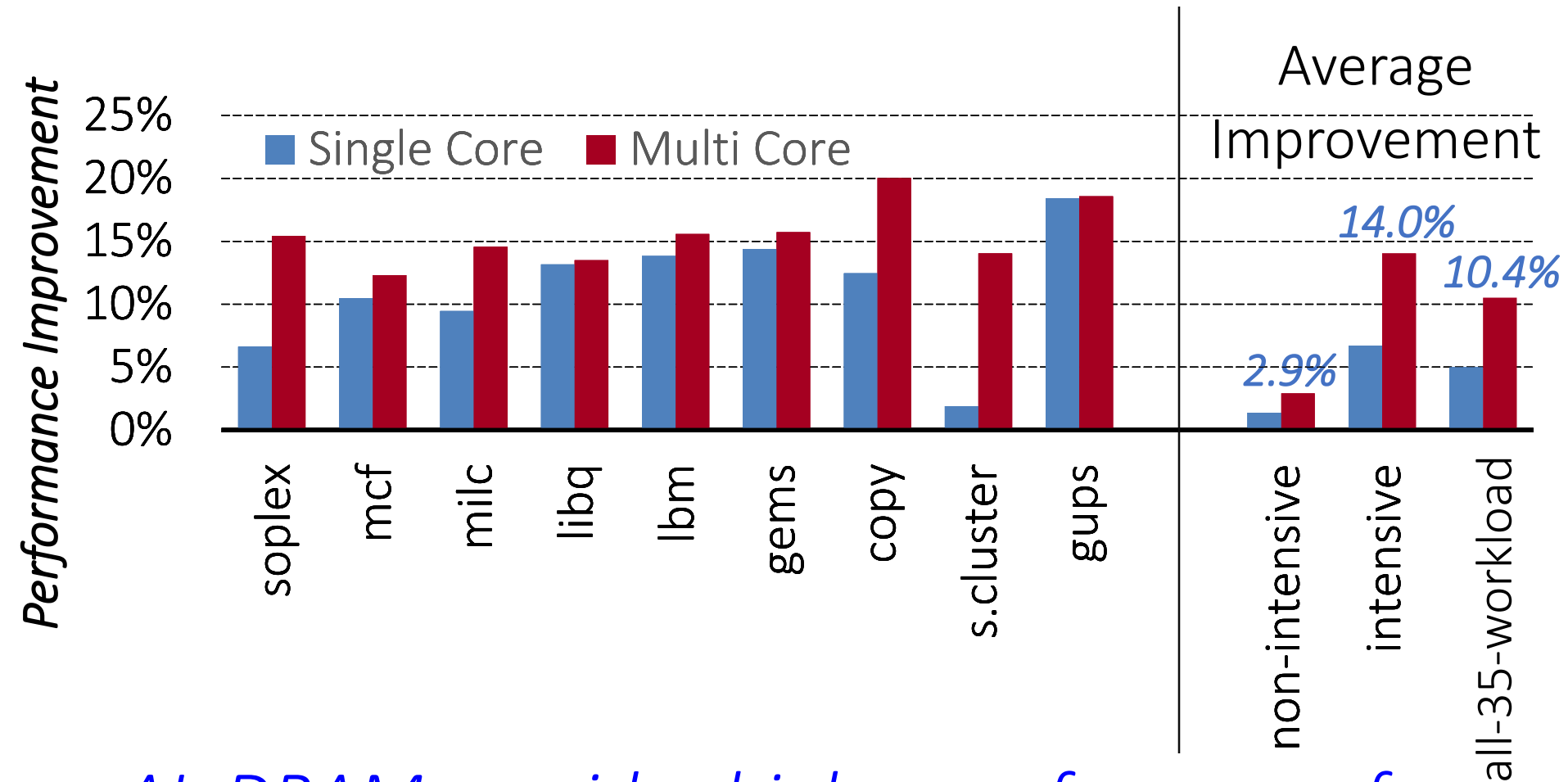
AL-DRAM provides higher performance for
multi-programmed & multi-threaded workloads

Reducing Latency Also Reduces Energy

- AL-DRAM reduces DRAM power consumption by 5.8%

- Major reason: reduction in row activation time

SAFARI

AL-DRAM: Advantages & Disadvantages

- **Advantages**

  + Simple mechanism to reduce latency

  + Significant system performance and energy benefits

    + Benefits higher at low temperature

  + Low cost, low complexity

- **Disadvantages**

  - Need to determine reliable operating latencies for different temperatures and different DIMMs → higher testing cost

    (might not be that difficult for low temperatures)

More on AL-DRAM

- Donghyuk Lee, Yoongu Kim, Gennady Pekhimenko, Samira Khan, Vivek Seshadri, Kevin Chang, and Onur Mutlu,
  "Adaptive-Latency DRAM: Optimizing DRAM Timing for the Common-Case"
  Proceedings of the 21st International Symposium on High-Performance Computer Architecture (**HPCA**), Bay Area, CA, February 2015.
  [Slides (pptx) (pdf)] [Full data sets]

Adaptive-Latency DRAM: Optimizing DRAM Timing for the Common-Case

Donghyuk Lee   Yoongu Kim   Gennady Pekhimenko

Samira Khan   Vivek Seshadri   Kevin Chang   Onur Mutlu
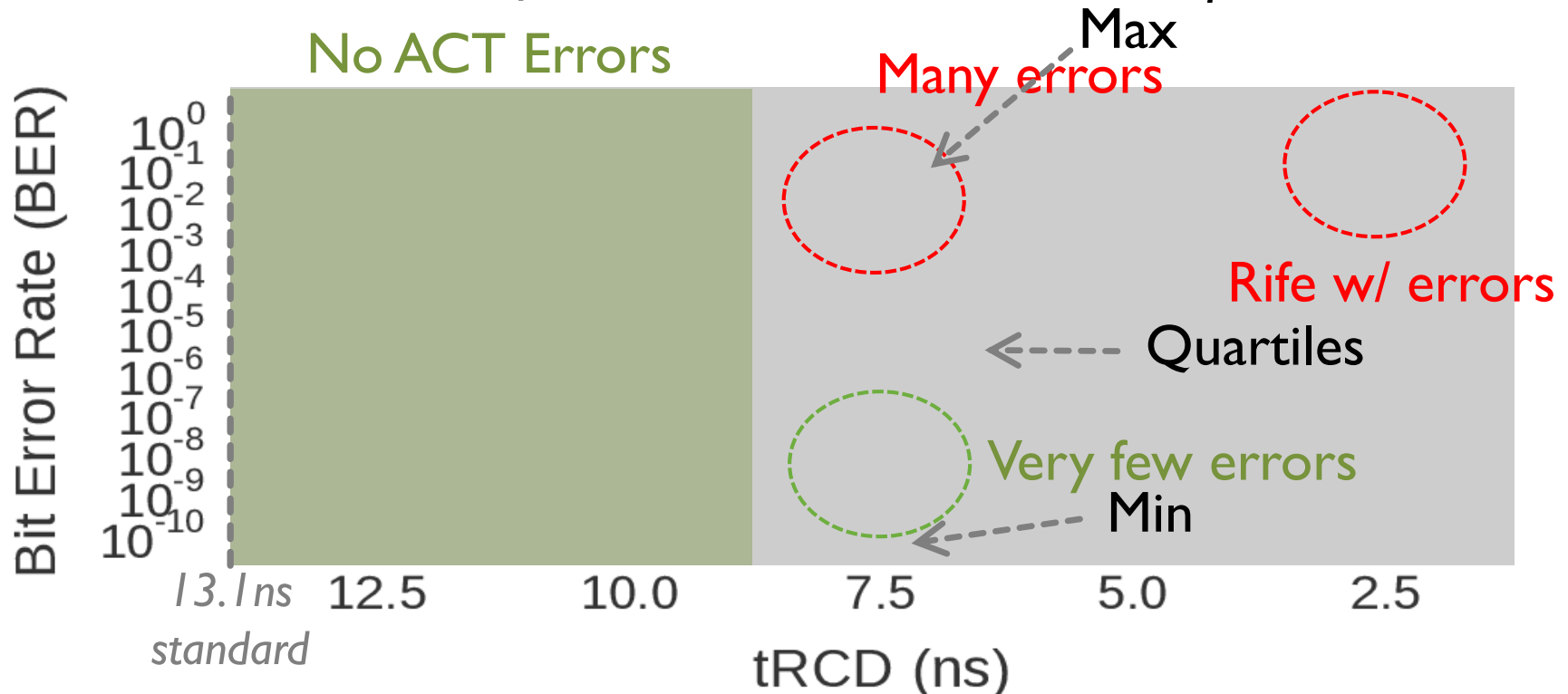
Carnegie Mellon University

SAFARI

Different Types of Latency Variation

- AL-DRAM exploits latency variation
  - Across time (different temperatures)
  - Across chips

- Is there also latency variation within a chip?
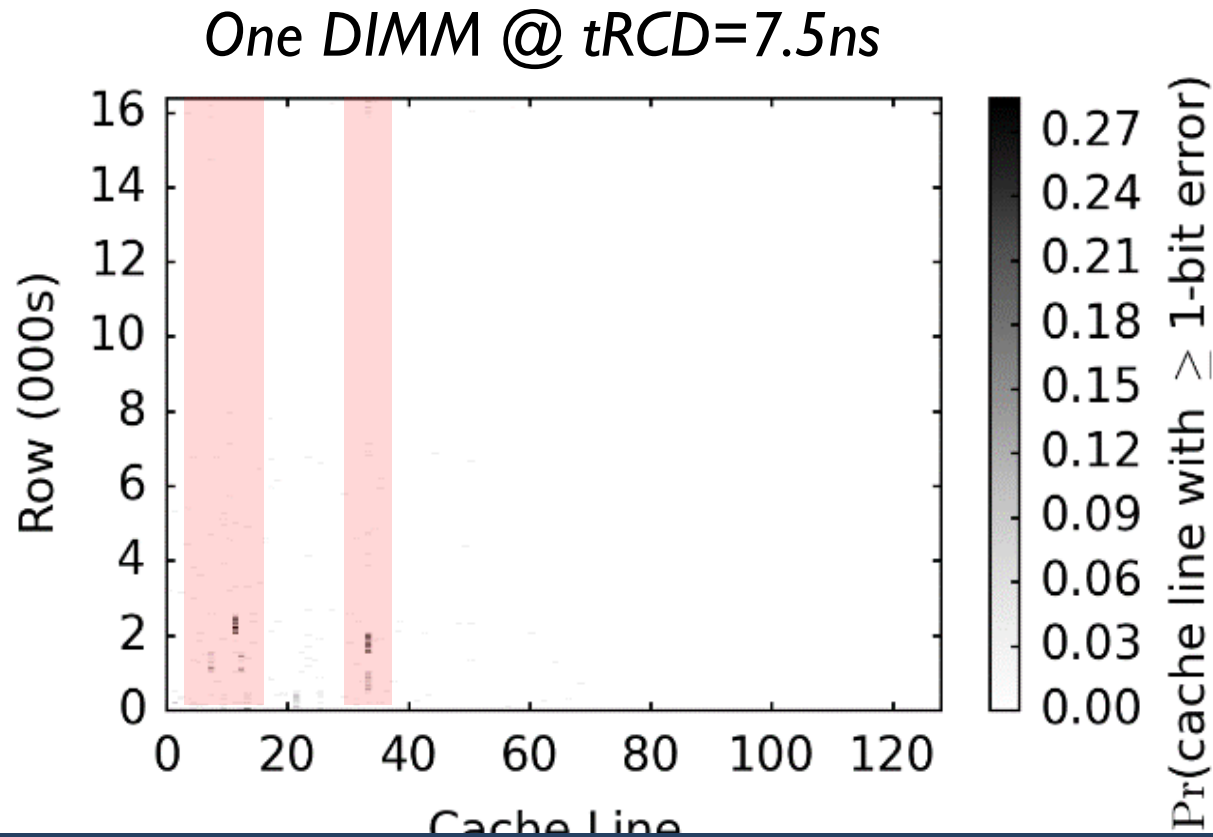 - Across different parts of a chip

Variation in Activation Errors

Results from 7500 rounds over 240 chips

Modern DRAM chips exhibit significant variation in activation latency

Spatial Locality of Activation Errors
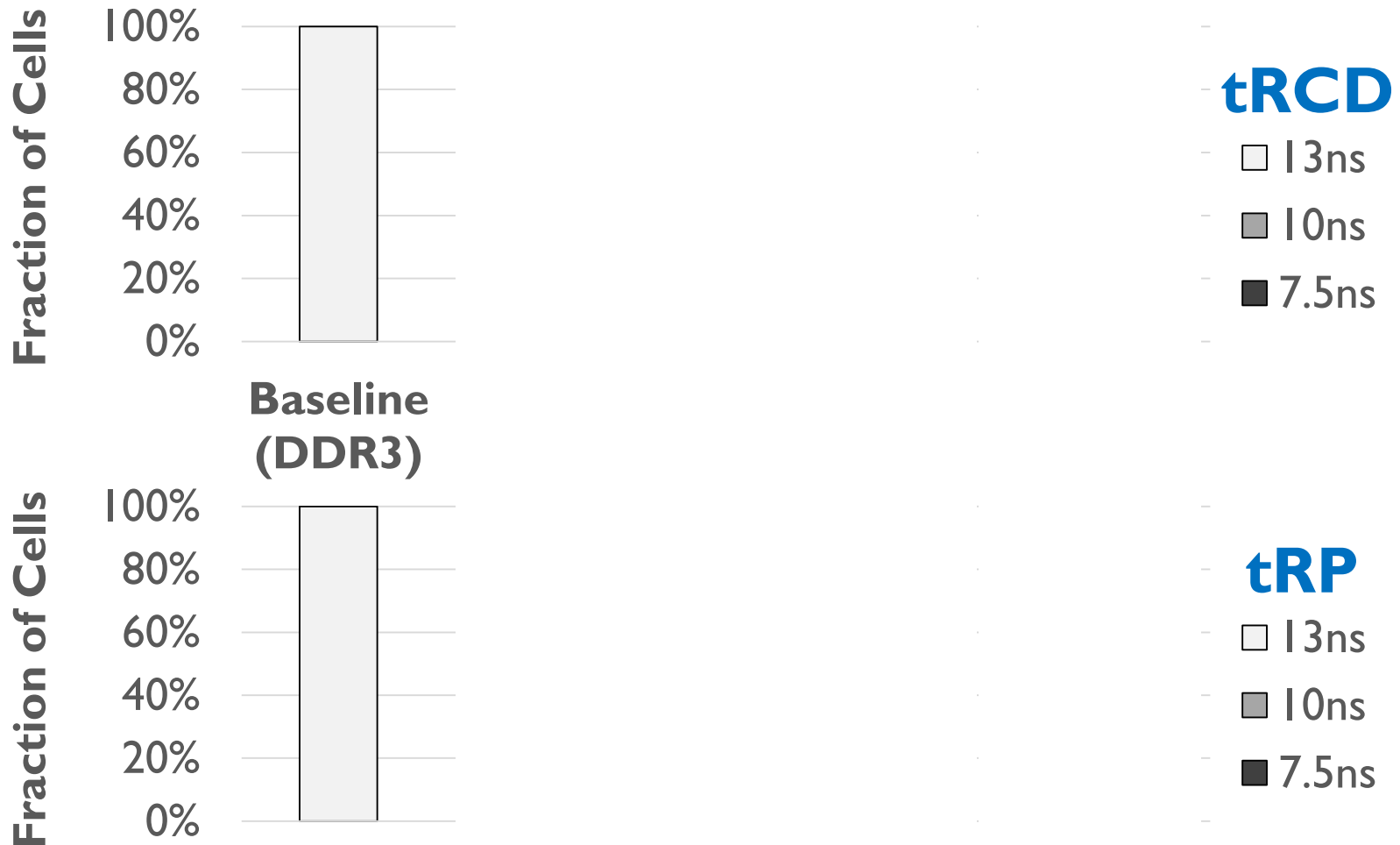

One DIMM @ tRCD=7.5ns

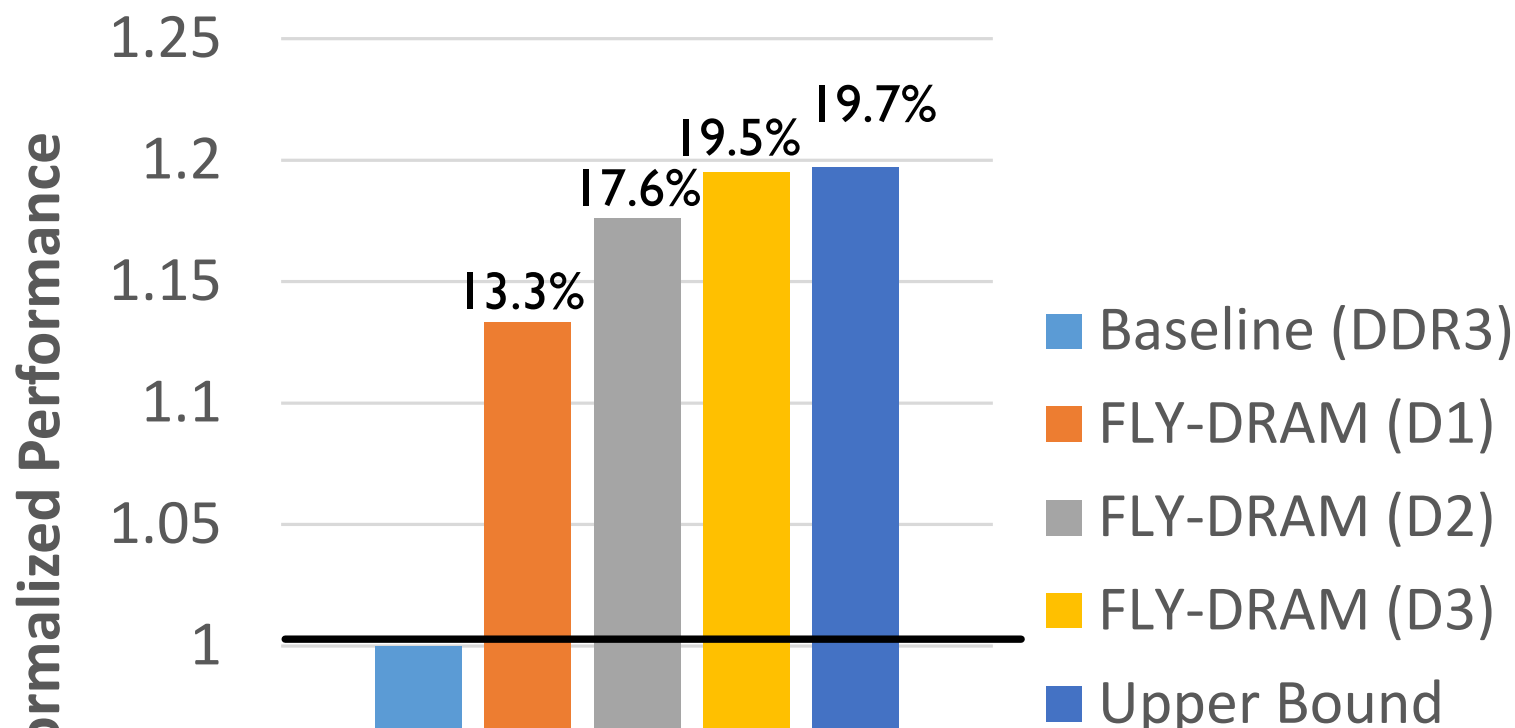**Activation errors are concentrated at certain columns of cells**

Mechanism to Reduce DRAM Latency

- **Observation:** DRAM timing errors (slow DRAM cells) are concentrated on certain regions

- **Flexible-LatencY (FLY) DRAM**
  - A software-transparent design that reduces latency

- **Key idea**:
  1) Divide memory into regions of different latencies
  2) *Memory controller:* Use lower latency for regions without slow cells; higher latency for other regions

Chang+, "**Understanding Latency Variation in Modern DRAM Chips: Experimental Characterization, Analysis, and Optimization"**," SIGMETRICS 2016.

FLY-DRAM Configurations

tRCD
- □ 13ns
- ▨ 10ns
- ■ 7.5ns

tRP
- □ 13ns
- ▨ 10ns
- ■ 7.5ns

Chang+, "**Understanding Latency Variation in Modern DRAM Chips: Experimental Characterization, Analysis, and Optimization"**," SIGMETRICS 2016.

Results

**FLY-DRAM improves performance
by exploiting spatial latency variation in DRAM**

Chang+, "**Understanding Latency Variation in Modern DRAM Chips: Experimental Characterization, Analysis, and Optimization"**," SIGMETRICS 2016.

FLY-DRAM: Advantages & Disadvantages

- **Advantages**

  + Reduces latency significantly

    + Exploits significant within-chip latency variation

- **Disadvantages**

  - Need to determine reliable operating latencies for different parts of a chip → higher testing cost

  - Slightly more complicated controller

SAFARI

Analysis of Latency Variation in DRAM Chips

- Kevin Chang, Abhijith Kashyap, Hasan Hassan, Samira Khan, Kevin Hsieh, Donghyuk Lee, Saugata Ghose, Gennady Pekhimenko, Tianshi Li, and Onur Mutlu,
  "Understanding Latency Variation in Modern DRAM Chips: Experimental Characterization, Analysis, and Optimization"
  Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (**SIGMETRICS**), Antibes Juan-Les-Pins, France, June 2016.
  [Slides (pptx) (pdf)]
  [Source Code]

Understanding Latency Variation in Modern DRAM Chips: Experimental Characterization, Analysis, and Optimization

Kevin K. Chang[1]   Abhijith Kashyap[1]   Hasan Hassan[1,2]

Saugata Ghose[1]   Kevin Hsieh[1]   Donghyuk Lee[1]   Tianshi Li[1,3]
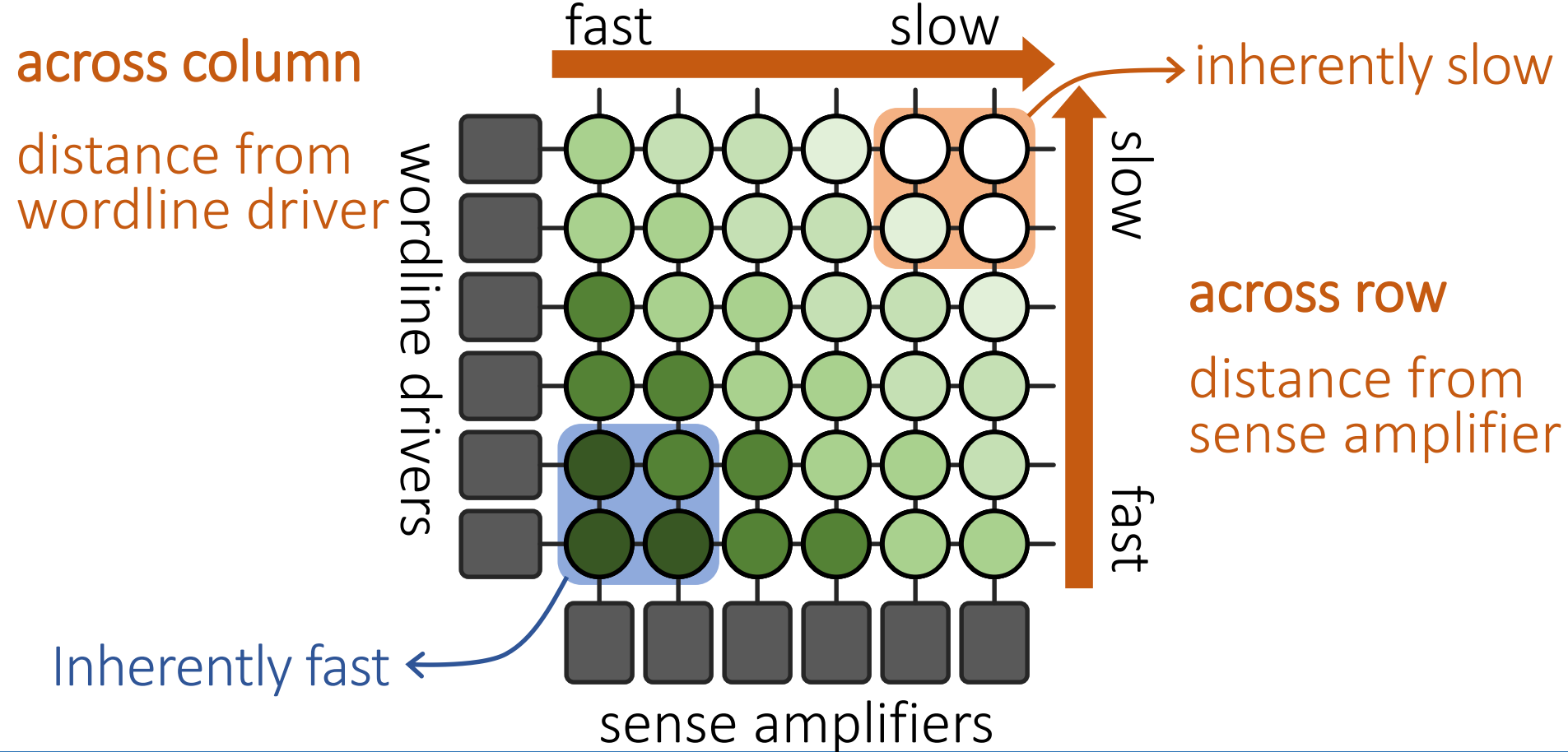
Gennady Pekhimenko[1]   Samira Khan[4]   Onur Mutlu[5,1]

[1]Carnegie Mellon University   [2]TOBB ETÜ   [3]Peking University   [4]University of Virginia   [5]ETH Zürich

SAFARI

/8

Why Is There Spatial Latency Variation Within a Chip?

What Is Design-Induced Variation?

fast   slow

across column

distance from wordline driver

inherently slow

wordline drivers

slow

across row

distance from sense amplifier

fast

Inherently fast

sense amplifiers

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

DIVA Online **Profiling**

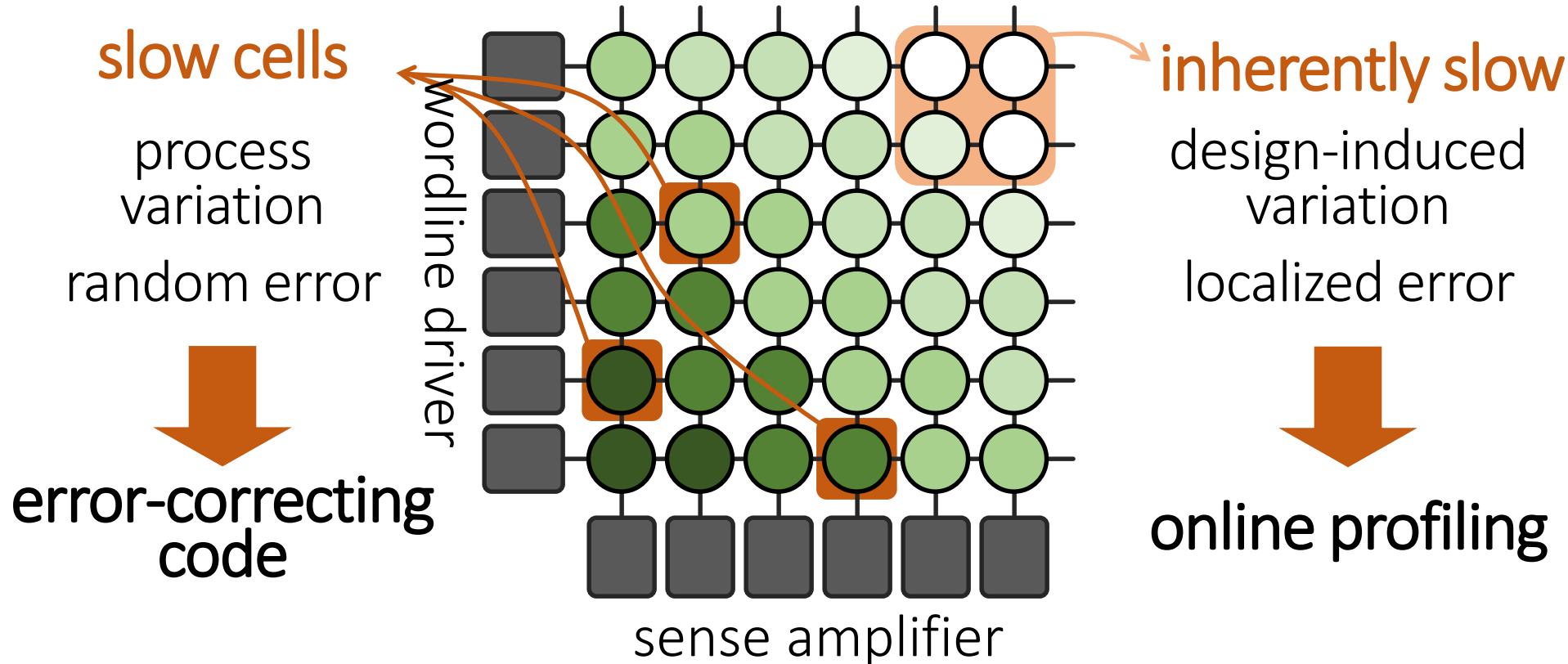**D**esign-**I**nduced-**V**ariation-**A**ware

wordline driver

inherently slow

sense amplifier
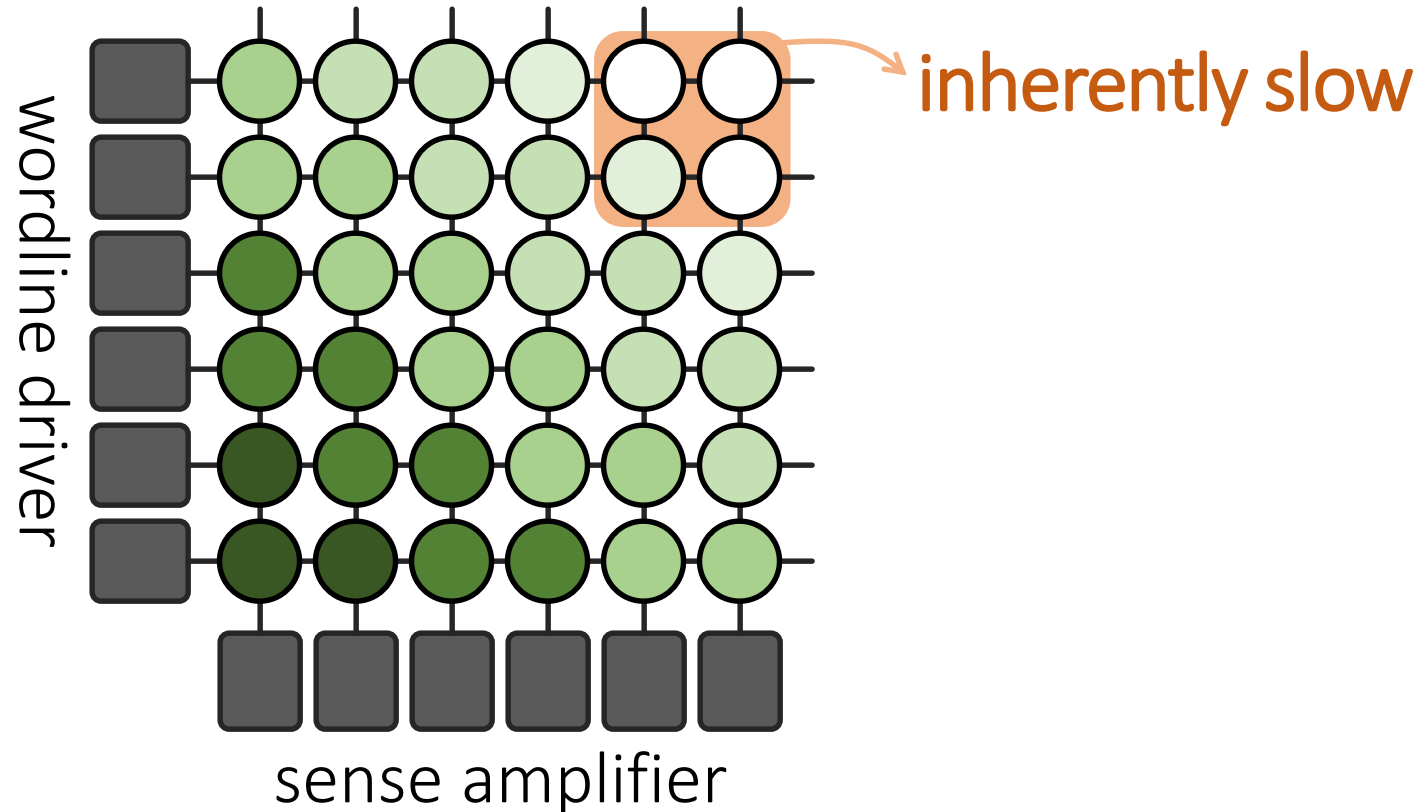
Profile *only slow regions* to determine min. latency
→ *Dynamic* & *low cost* latency optimization

DIVA Online **Profiling**

Design-**I**nduced-**V**ariation-**A**ware

slow cells

inherently slow

process variation

design-induced variation

random error

localized error

Wordline driver

sense amplifier

error-correcting code

online profiling

Combine error-correcting codes & online profiling → Reliably reduce DRAM latency

DIVA-DRAM Reduces Latency

Read

Write

Latency Reduction

AL-DRAM: 55°C 31.2%, 85°C 25.5%
DIVA Profiling: 55°C 35.1%, 85°C 34.6%
DIVA Profiling + Shuffling: 55°C 36.6%, 85°C 35.8%

AL-DRAM: 55°C 36.6%, 85°C 27.5%
DIVA Profiling: 55°C 39.4%, 85°C 38.7%
DIVA Profiling + Shuffling: 55°C 41.3%, 85°C 40.3%
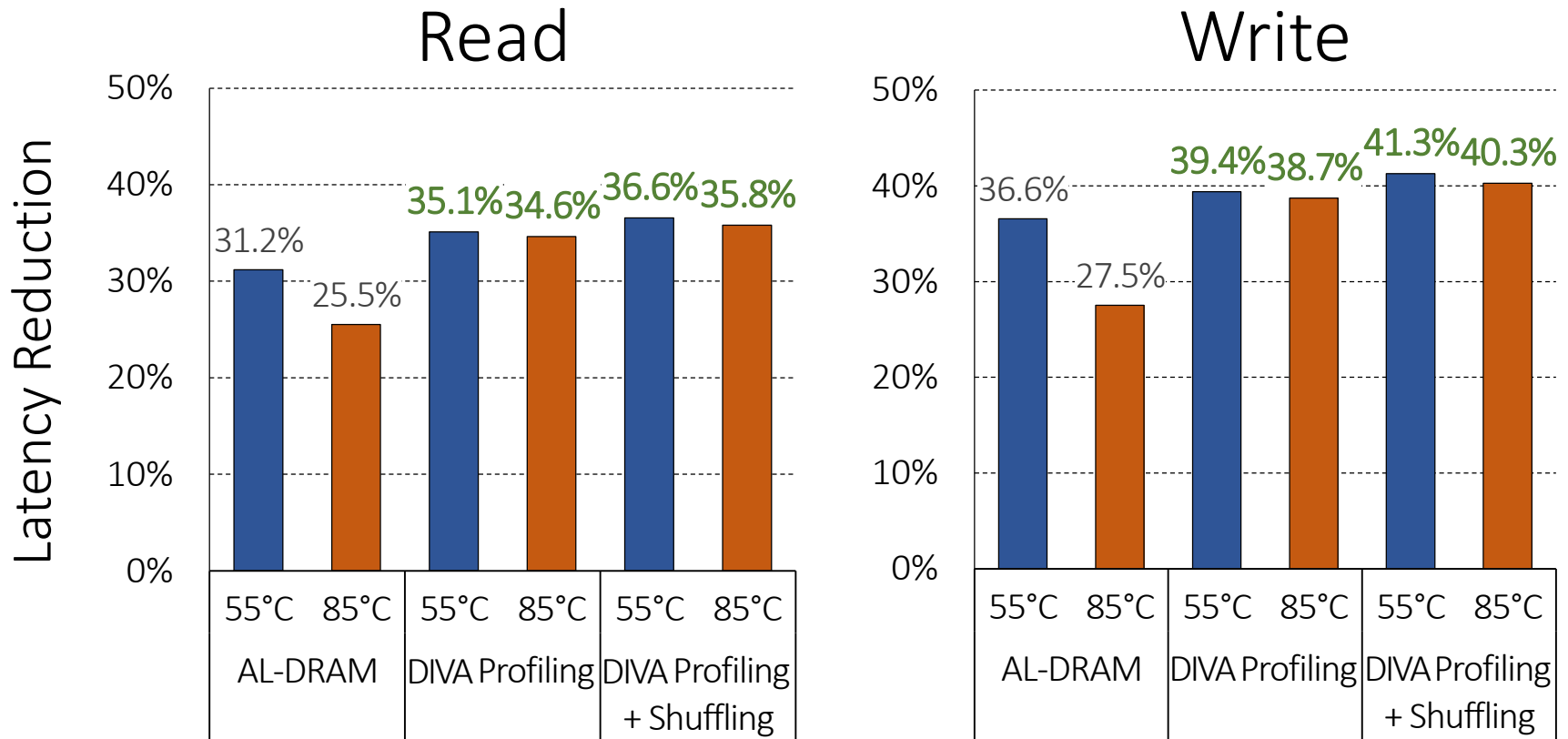
DIVA-DRAM *reduces latency more aggressively* and uses ECC to correct random slow cells

DIVA-DRAM: Advantages & Disadvantages

- **Advantages**

  ++ Automatically finds the lowest reliable operating latency at system runtime (lower production-time testing cost)

  + Reduces latency more than prior methods (w/ ECC)

  + Reduces latency at high temperatures as well

- **Disadvantages**

  - Requires knowledge of inherently-slow regions

  - Requires ECC (Error Correcting Codes)

  - Imposes overhead during runtime profiling

SAFARI

Design-Induced Latency Variation in DRAM

- Donghyuk Lee, Samira Khan, Lavanya Subramanian, Saugata Ghose, Rachata Ausavarungnirun, Gennady Pekhimenko, Vivek Seshadri, and Onur Mutlu,
  "Design-Induced Latency Variation in Modern DRAM Chips: Characterization, Analysis, and Latency Reduction Mechanisms"
  Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (**SIGMETRICS**), Urbana-Champaign, IL, USA, June 2017.

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

Donghyuk Lee, NVIDIA and Carnegie Mellon University
Samira Khan, University of Virginia
Lavanya Subramanian, Saugata Ghose, Rachata Ausavarungnirun, Carnegie Mellon University
Gennady Pekhimenko, Vivek Seshadri, Microsoft Research
Onur Mutlu, ETH Zürich and Carnegie Mellon University

Understanding & Exploiting the Voltage-Latency-Reliability Relationship

High DRAM Power Consumption

- <u>Problem</u>: High DRAM (memory) power in today's systems

>40% in POWER7 (Ware+, HPCA'10)   >40% in GPU (Paul+, ISCA'15)

Low-Voltage Memory

- Existing DRAM designs to help reduce DRAM power by lowering supply voltage conservatively
  - $Power \propto Voltage^2$

- DDR3L (low-voltage) reduces voltage from 1.5V to 1.35V (**-10%**)

- LPDDR4 (low-power) employs low-power I/O interface with 1.2V (lower bandwidth)

Can we reduce DRAM power and energy by *further* reducing supply voltage?

SAFARI

Goals

1 Understand and characterize the various characteristics of DRAM under **reduced voltage**

2 Develop a mechanism that reduces DRAM energy by **lowering voltage** while keeping performance loss within a target

Key Questions

- How does reducing voltage affect *reliability* (errors)?

- How does reducing voltage affect *DRAM latency*?

- How do we design a new DRAM energy reduction mechanism?

SAFARI

Supply Voltage Control on DRAM

DRAM Module
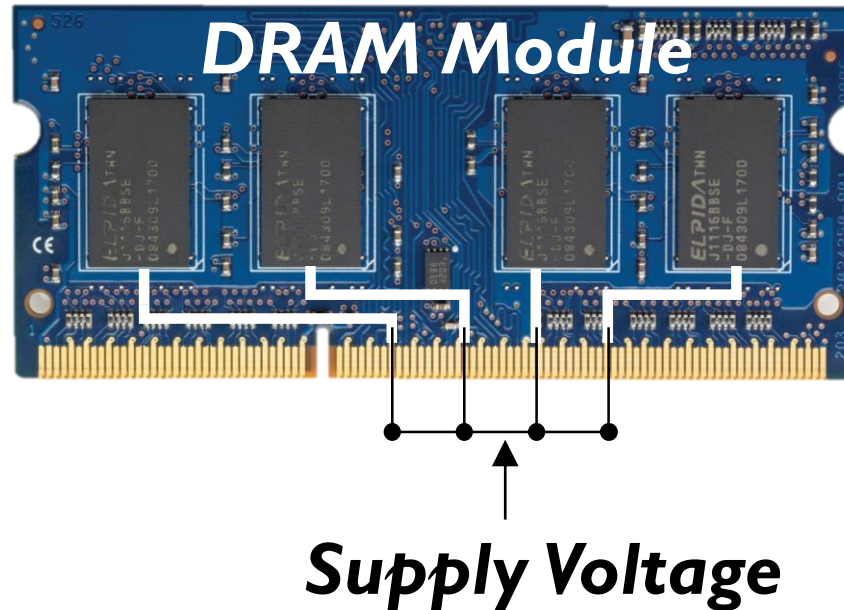
Supply Voltage

Adjust the *supply voltage* to every chip on the same module

SAFARI

Custom Testing Platform

SoftMC [Hassan+, HPCA'17]: FPGA testing platform to

1) Adjust supply voltage to DRAM modules

2) Schedule DRAM commands to DRAM modules

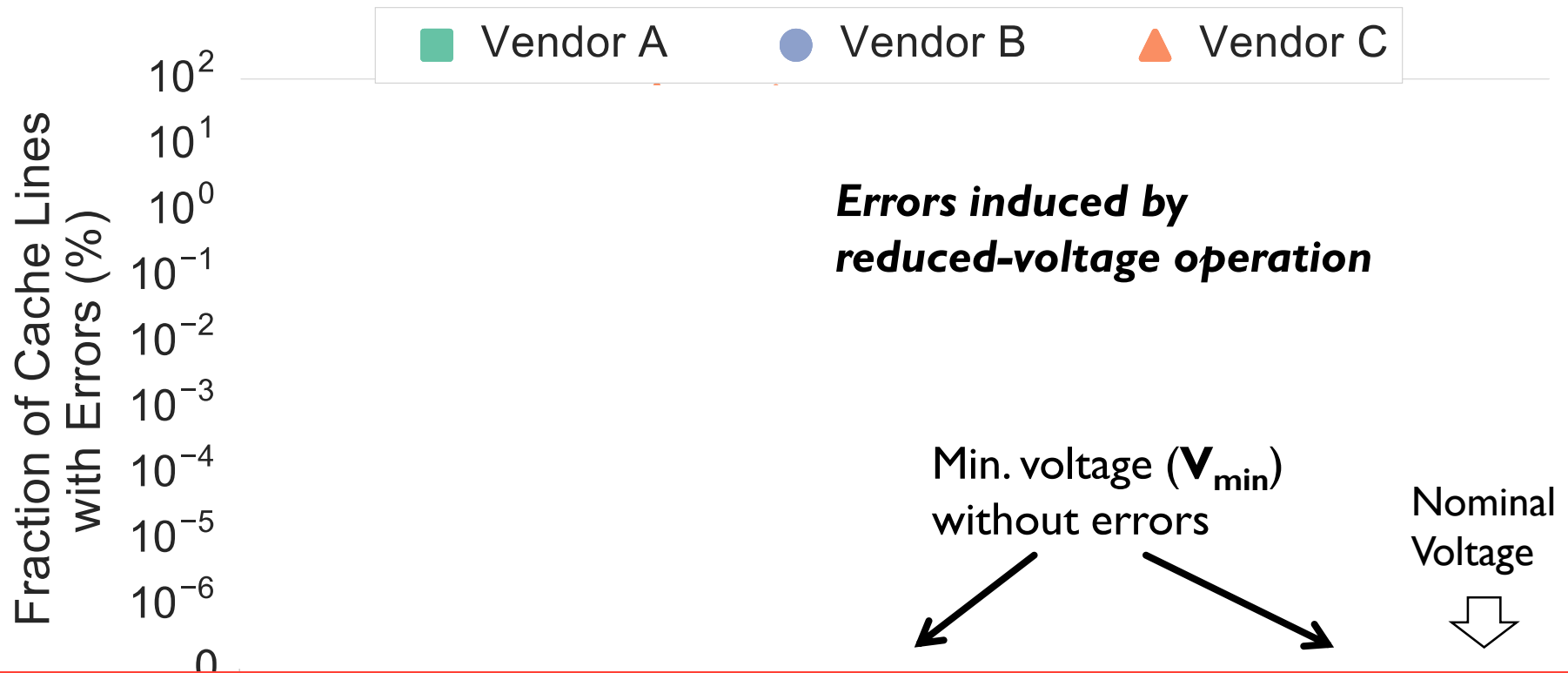Existing systems: DRAM commands not exposed to users

DRAM module → **FPGA** ← **Voltage controller**

https://github.com/CMU-SAFARI/DRAM-Voltage-Study

SAFARI
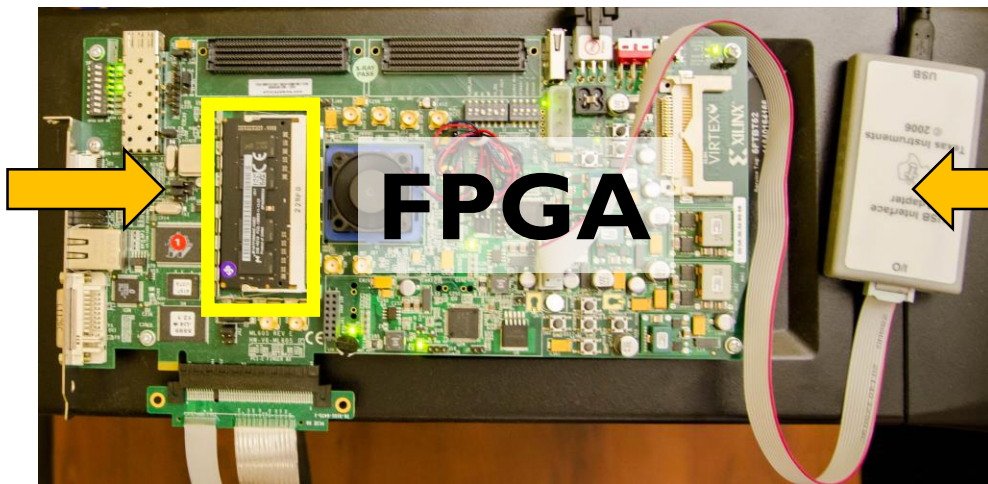
Tested DRAM Modules

- 124 **DDR3L** (low-voltage) DRAM chips
  - **31 SO-DIMMs**
  - **1.35V** (DDR3 uses 1.5V)
  - Density: 4Gb per chip
  - Three major vendors/manufacturers
  - Manufacturing dates: 2014-2016

- Iteratively read every bit in each 4Gb chip under a wide range of supply voltage levels: 1.35V to 1.0V (**-26%**)

SAFARI

Reliability Worsens with Lower Voltage

Vendor A ● Vendor B ▲ Vendor C

10^2
10^1
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
0

Fraction of Cache Lines with Errors (%)

Errors induced by reduced-voltage operation

Min. voltage (V_{min}) without errors

Nominal Voltage

Reducing voltage below V_{min} causes an increasing number of errors

SAFARI

Source of Errors

Detailed circuit simulations (SPICE) of a DRAM cell array to model the behavior of DRAM operations
https://github.com/CMU-SAFARI/DRAM-Voltage-Study

Circuit model

⬤ Activate   ▲ Precharge

Nominal Voltage

Latency (ns): 20, 15, 10, 5
Supply Voltage (V): 0.9, 1.0, 1.1, 1.2, 1.3

Reliable low-voltage operation requires higher latency

SAFARI

DIMMs Operating at Higher Latency

Measured minimum latency that *does **not** cause errors* in DRAM modules
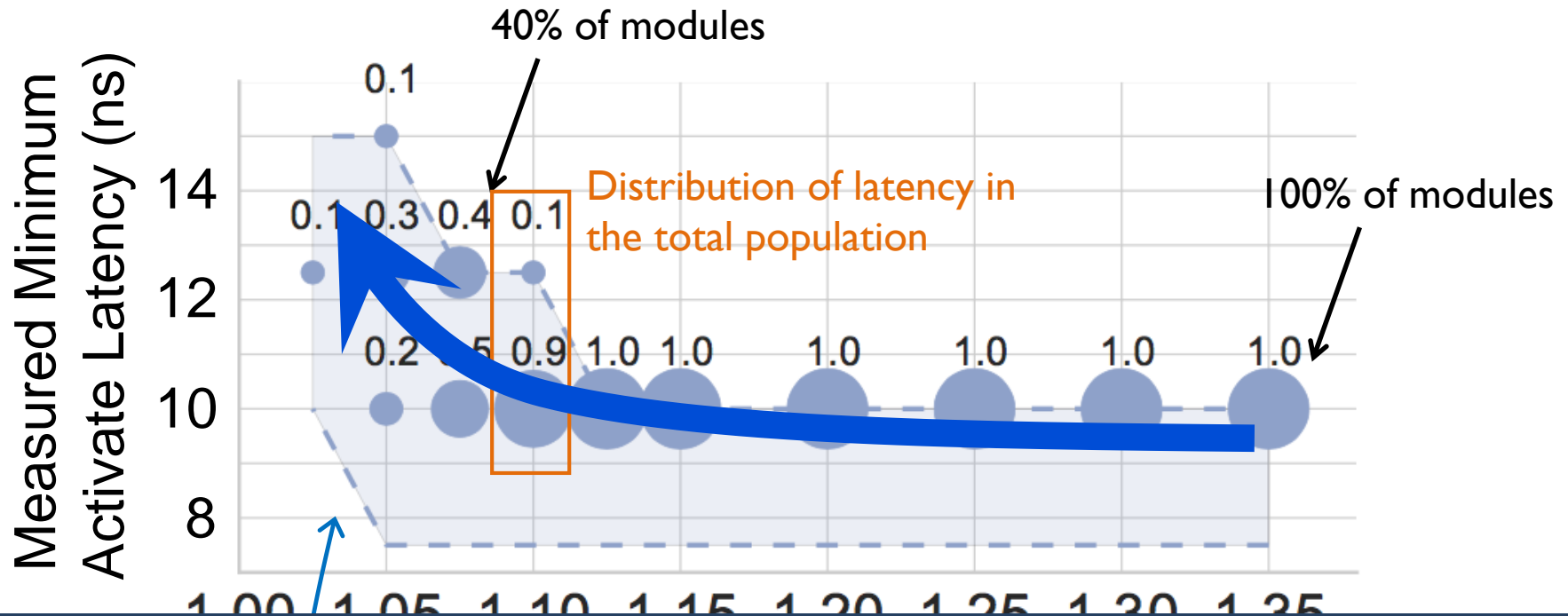
DRAM requires longer latency to access data **without errors** at lower voltage

SAFARI

Spatial Locality of Errors

A module under 1.175V (**12% voltage reduction**)
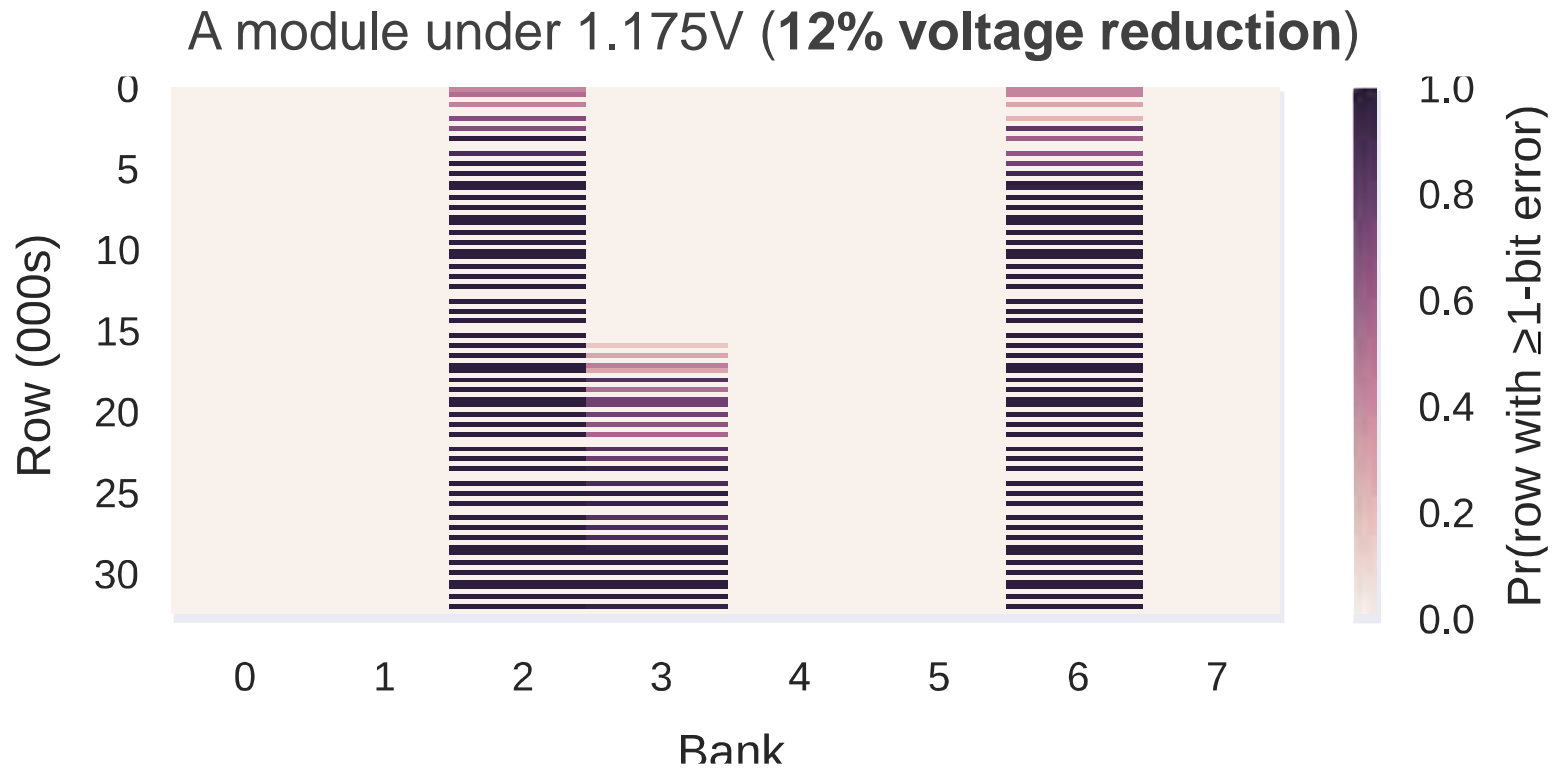
Errors concentrate in certain regions

SAFARI

Summary of Key Experimental Observations

- Voltage-induced errors increase as voltage reduces further below V_{min}

- Errors exhibit spatial locality

- Increasing the latency of DRAM operations mitigates voltage-induced errors

SAFARI

DRAM Voltage Adjustment to Reduce Energy

- <u>Goal</u>: Exploit the trade-off between voltage and latency to reduce energy consumption

- <u>Approach</u>: Reduce DRAM voltage **reliably**
  - Performance loss due to increased latency at lower voltage
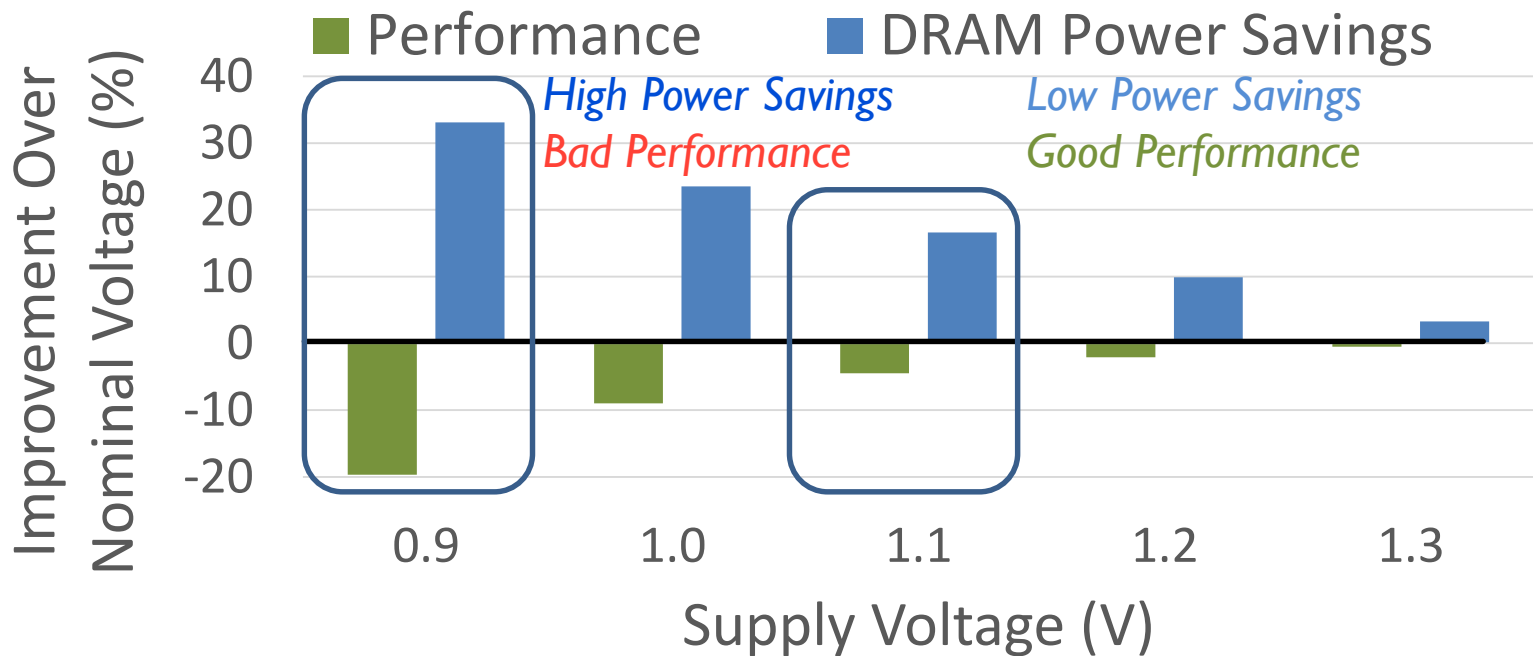
Voltron Overview

Voltron

User specifies the
performance loss target
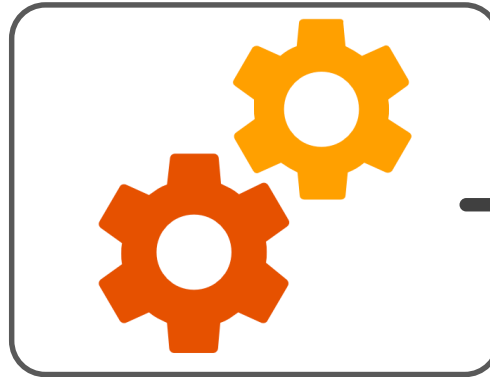
Select the **minimum** DRAM voltage
without violating the target

**How do we predict performance loss due to
increased latency under low DRAM voltage?**

Linear Model to Predict Performance

Voltron

User specifies the
performance loss target

Select the **minimum** DRAM voltage
without violating the target

*Application's
characteristics*

[-1%, -3%, ...]
*Predicted
performance loss*

*Min.
Voltage*

Final
Voltage

[1.3V, 1.25V, ...]
DRAM Voltage

Target

Linear regression model

SAFARI
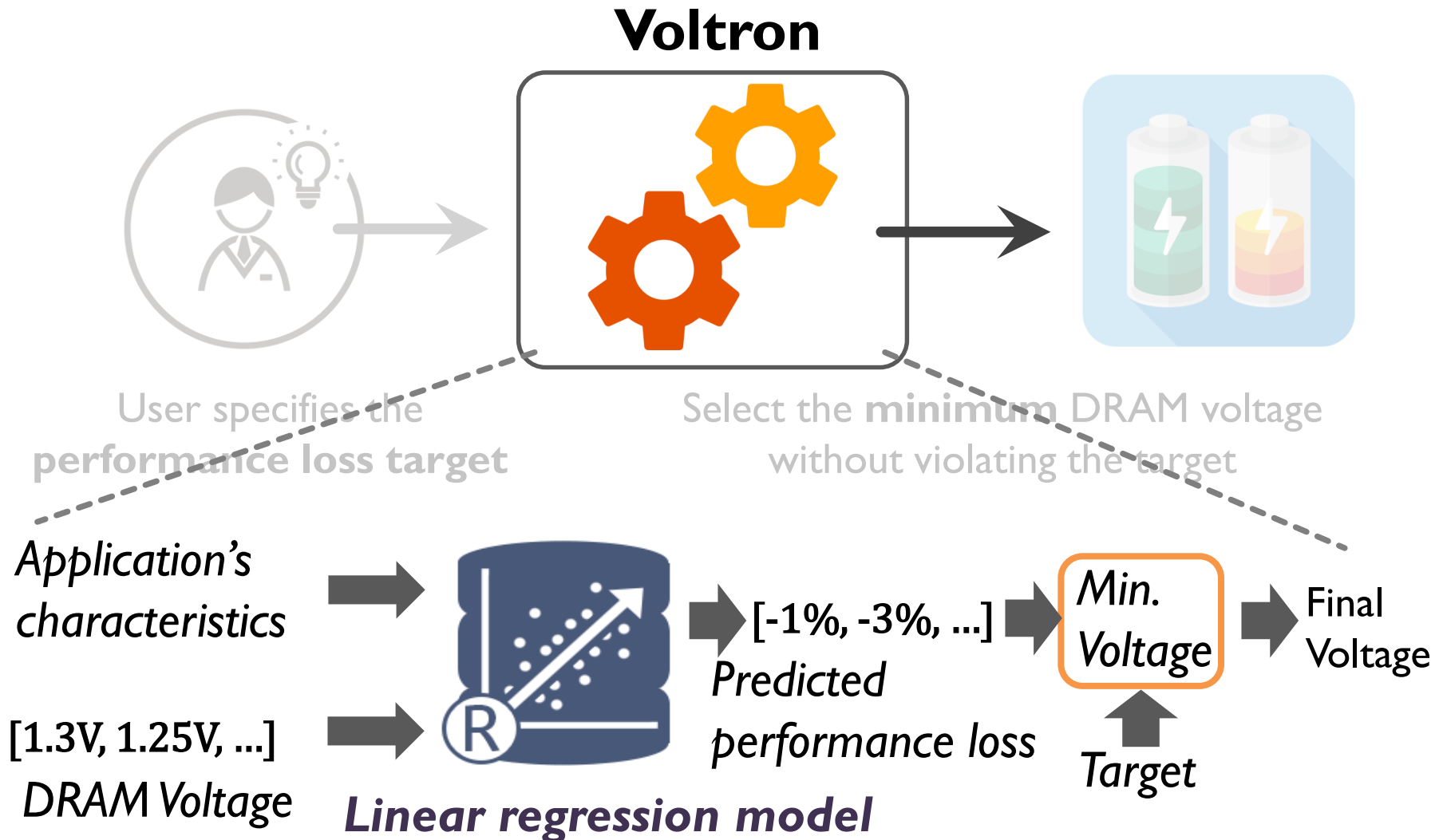
Regression Model to Predict Performance

- Application's characteristics for the model:
  - ***Memory intensity***: Frequency of last-level cache misses
  - ***Memory stall time***: Amount of time memory requests stall commit inside CPU

- Handling multiple applications:
  - Predict a performance loss for each application
  - Select the minimum voltage that satisfies the performance target for all applications

SAFARI

Comparison to Prior Work

- <u>Prior work</u>: Dynamically scale *frequency and voltage* of the entire DRAM based on bandwidth demand [David+, ICAC'11]

    - – <u>Problem</u>: Lowering voltage on the peripheral circuitry decreases channel frequency (memory data throughput)

- <u>Voltron</u>: Reduce voltage to only DRAM array without changing the voltage to peripheral circuitry

SAFARI
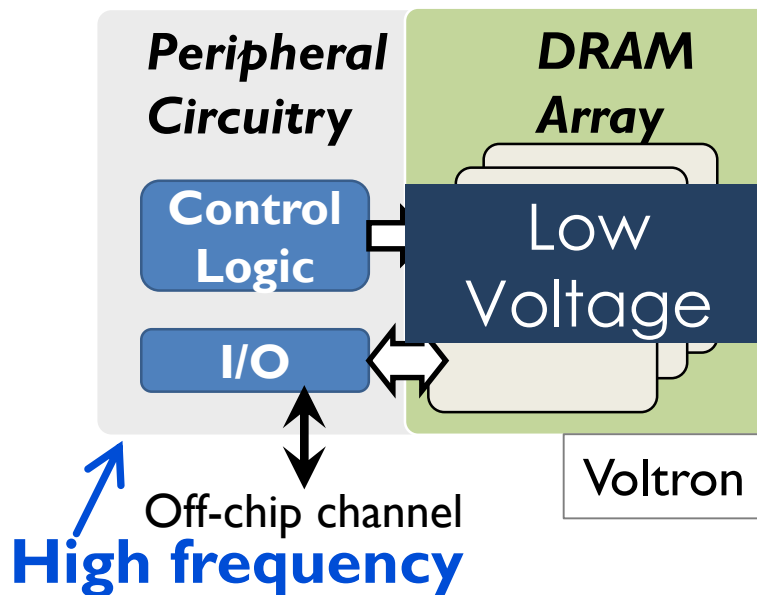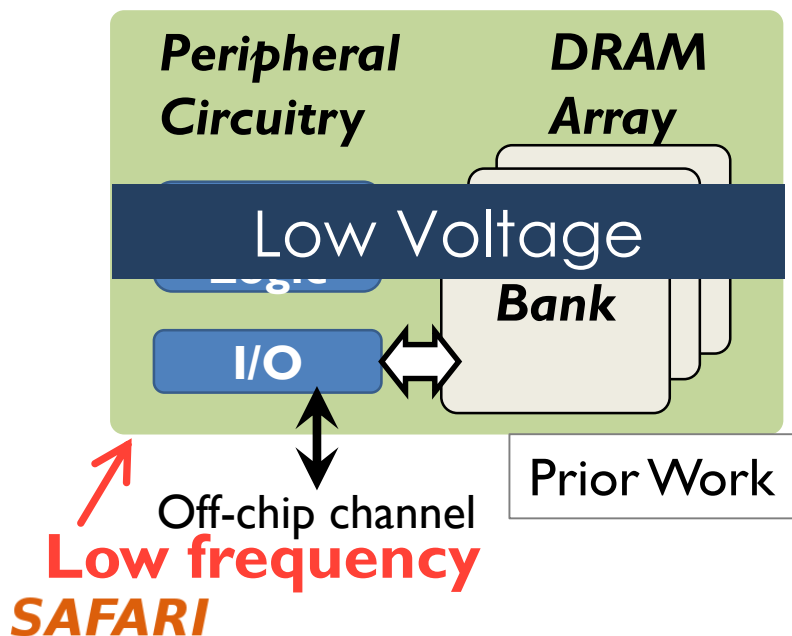
Exploiting Spatial Locality of Errors

<u>Key idea</u>: Increase the latency only for DRAM banks that observe errors under low voltage

- <u>Benefit</u>: Higher performance

High latency   **Low latency**
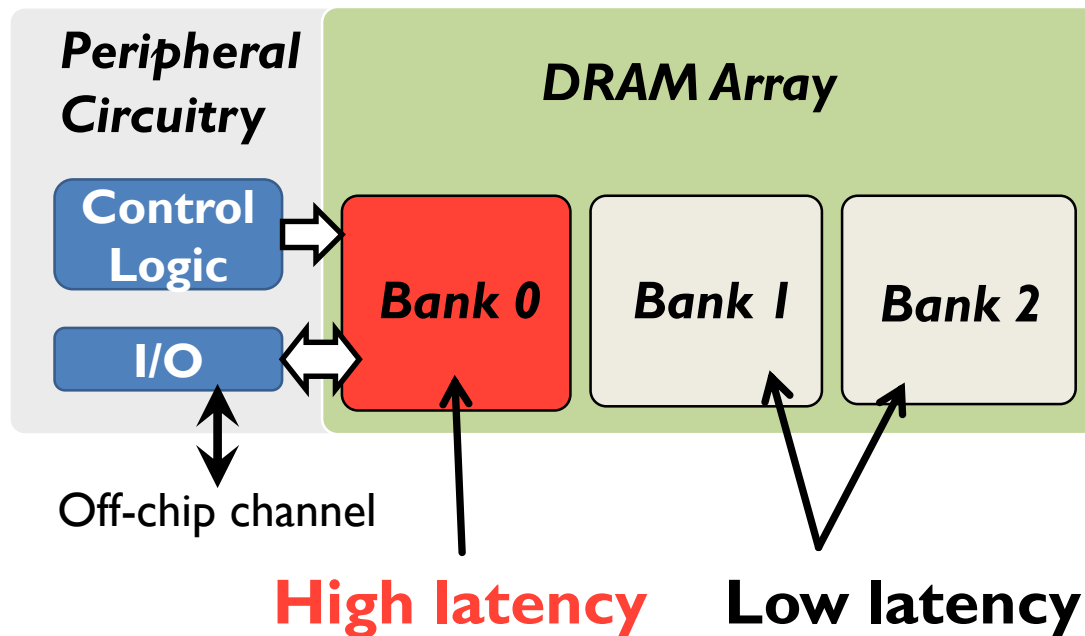
Voltron Evaluation Methodology

- **Cycle-level simulator**: Ramulator [CAL'15]
  - **McPAT** and **DRAMPower** for energy measurement

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

- **4-core** system with DDR3L memory

- **Benchmarks**: SPEC2006, YCSB

- Comparison to prior work: **MemDVFS** [David+, ICAC'11]
  - Dynamic DRAM frequency and voltage scaling
  - Scaling based on the *memory bandwidth consumption*

Energy Savings with Bounded Performance

Left chart legend: ■ MemDVFS [David+, ICAC'11] ■ Voltron

Left chart — y-axis: CPU+DRAM Energy Savings (%), scale 0 to 8; x-axis: Memory Intensity (Low, High)

More savings for high bandwidth applications

3.2%

7.3%

Right chart — Meets performance target

y-axis: Performance Loss (%), scale 0 to -6; x-axis: Memory Intensity (Low, High)

-1.6%

-1.8%

Performance Target

SAFARI
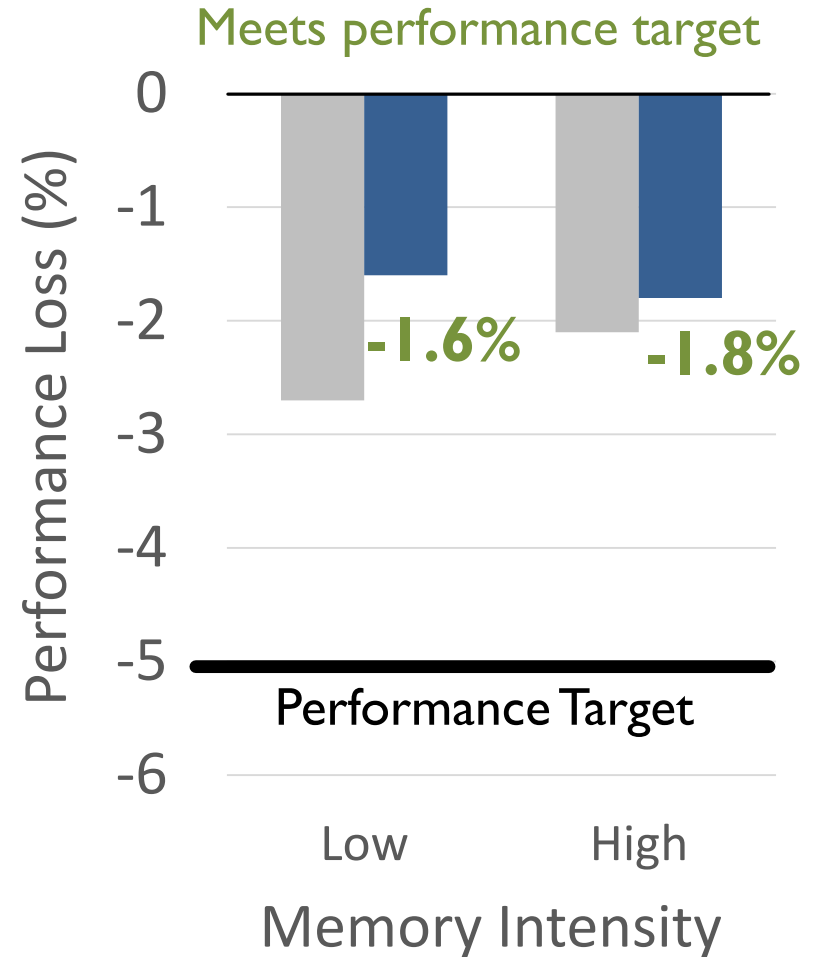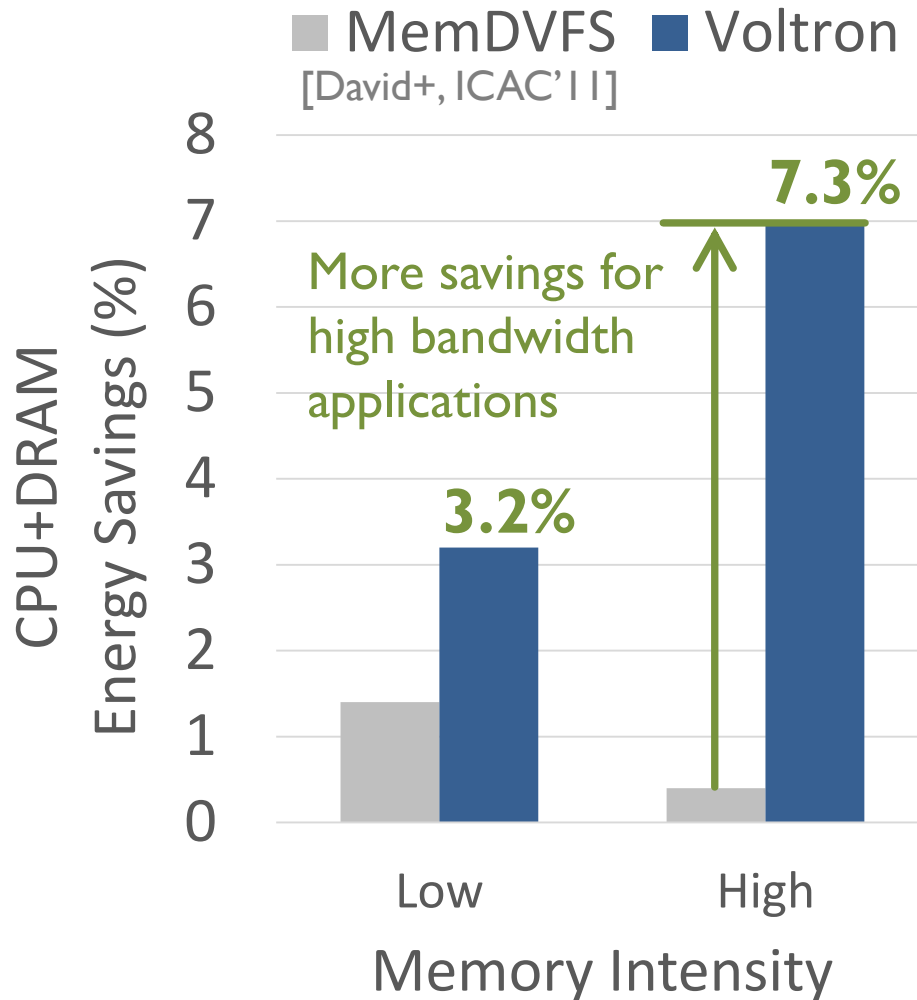
Voltron: Advantages & Disadvantages

- **Advantages**

  + Can trade-off between voltage and latency to improve energy or performance

  + Can exploit the high voltage margin present in DRAM

- **Disadvantages**

  - Requires finding the reliable operating voltage for each chip → higher testing cost

Analysis of Latency-Voltage in DRAM Chips

- Kevin Chang, A. Giray Yaglikci, Saugata Ghose, Aditya Agrawal, Niladrish Chatterjee, Abhijith Kashyap, Donghyuk Lee, Mike O'Connor, Hasan Hassan, and Onur Mutlu,
  "Understanding Reduced-Voltage Operation in Modern DRAM Devices: Experimental Characterization, Analysis, and Mechanisms"
  Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (**SIGMETRICS**), Urbana-Champaign, IL, USA, June 2017.

Understanding Reduced-Voltage Operation in Modern DRAM Chips: Characterization, Analysis, and Mechanisms

Kevin K. Chang[†]   Abdullah Giray Yağlıkçı[†]   Saugata Ghose[†]   Aditya Agrawal[¶]   Niladrish Chatterjee[¶]

Abhijith Kashyap[†]   Donghyuk Lee[¶]   Mike O'Connor[¶,‡]   Hasan Hassan[§]   Onur Mutlu[§,†]

[†]Carnegie Mellon University   [¶]NVIDIA   [‡]The University of Texas at Austin   [§]ETH Zürich

And, What If …

- … we can sacrifice reliability of some data to access it with even lower latency?

The DRAM Latency PUF:

Quickly Evaluating Physical Unclonable Functions by Exploiting the Latency-Reliability Tradeoff in Modern Commodity DRAM Devices

Jeremie S. Kim   **Minesh Patel**

Hasan Hassan   **Onur Mutlu**

Systems@ETH Zürich

SAFARI

ETH Zürich

Carnegie Mellon

Motivation

- A **PUF** is function that generates a signature **unique** to a given device

- Used in a **Challenge-Response Protocol**
  - Each device generates a unique **PUF response** depending the inputs

  - A trusted server **authenticates** a device if it generates the expected PUF response

DRAM Latency Characterization of 223 LPDDR4 DRAM Devices

- Latency failures come from accessing DRAM with **reduced** timing parameters.

- **Key Observations:**
  1. A cell's **latency failure** probability is determined by **random process variation**

  2. Latency failure patterns are **repeatable and unique to a device**

DRAM Latency PUF Key Idea

High % chance to fail with reduced t_{RCD}

Low % chance to fail with reduced t_{RCD}

Row Decoder

SA SA SA SA SA SA SA

DRAM Accesses and Failures

Bitline Voltage

V_{dd}
V_{min}
0.5 V_{dd}

Ready to Access
Voltage Level

Bitline Charge Sharing

wordline

access transistor

capacitor

bitline

Process variation *during manufacturing leads to cells having unique characteristics*

Time

ACTIVATE   SA Enable   READ

t_{RCD}

DRAM Accesses and Failures

Bitline Voltage

V_{dd}

V_{min}

Ready to Access Voltage Level

0.5 V_{dd}

weaker *cells have a* **higher probability** *to fail*

Time

ACTIVATE

SA Enable

READ

t_{RCD}
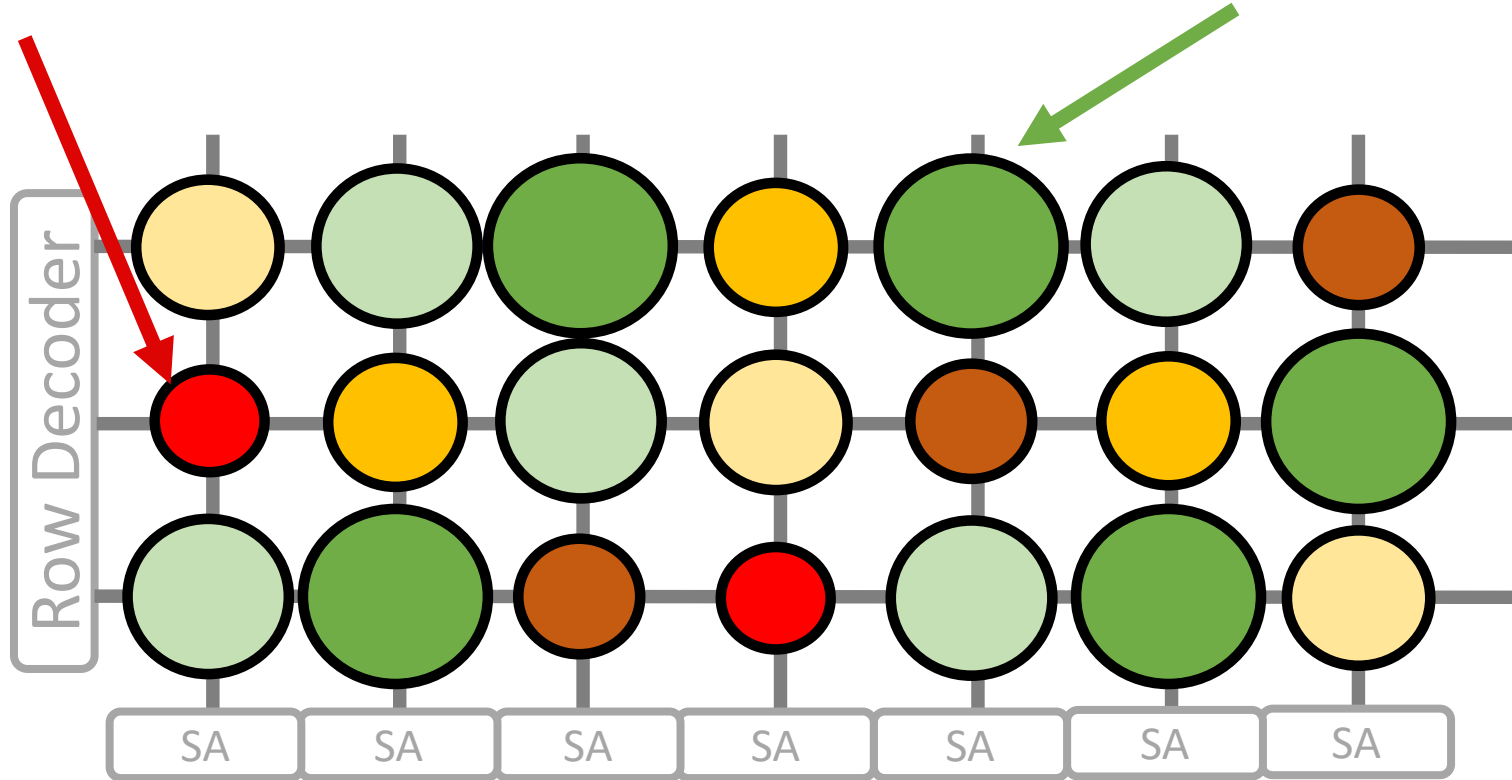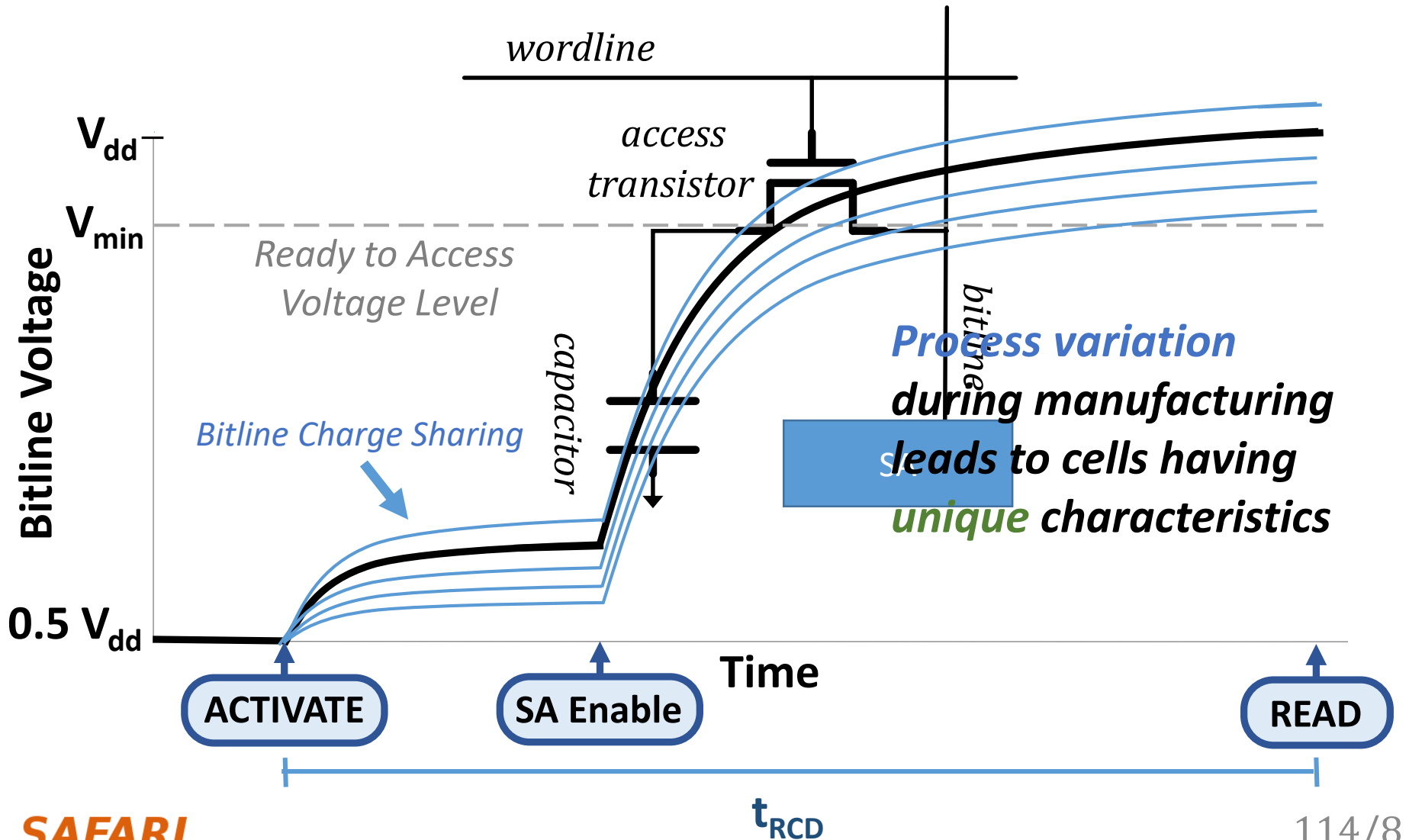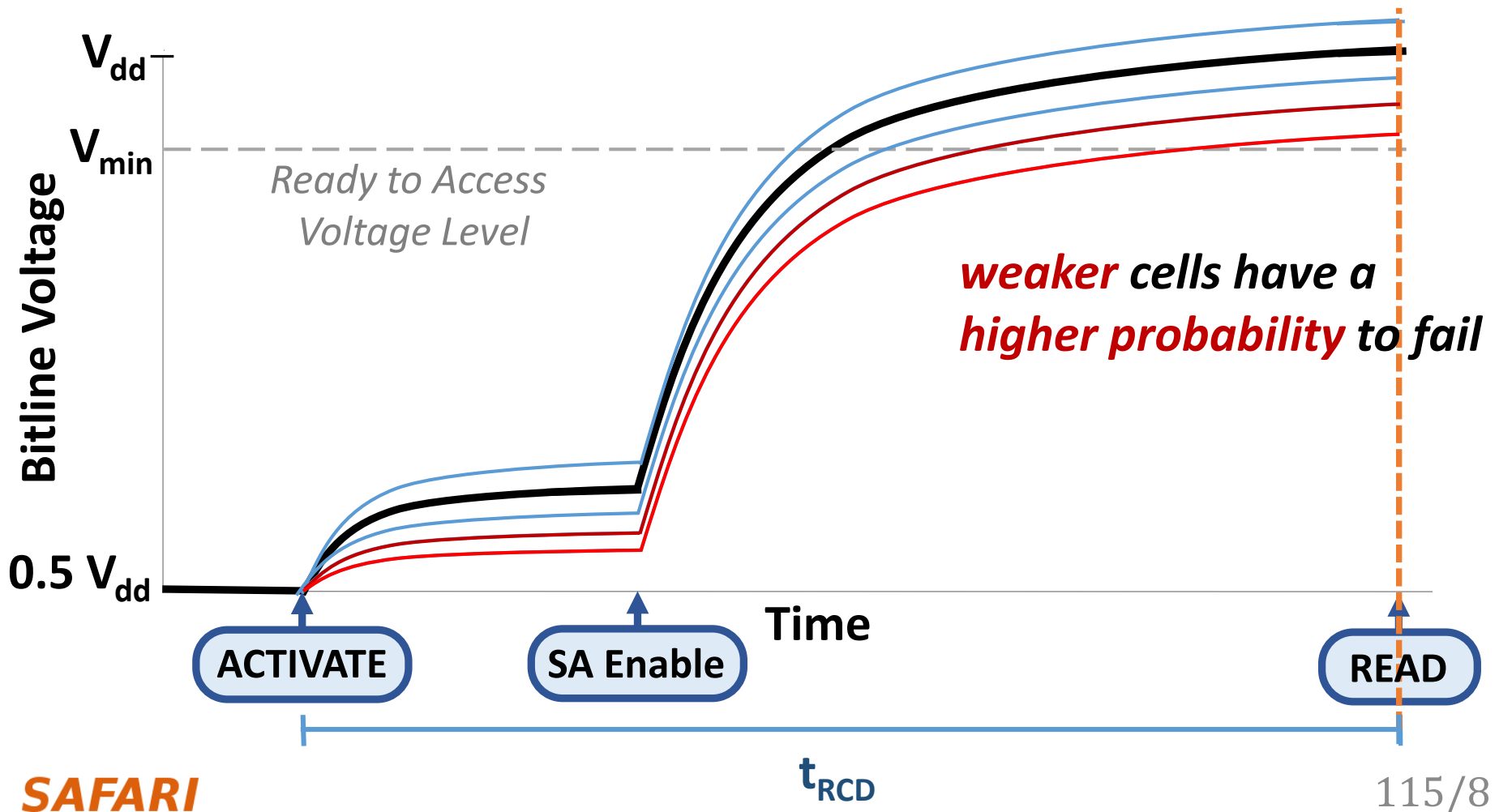
The DRAM Latency PUF Evaluation

- We generate PUF responses using **latency errors** in a region of DRAM

- The latency error patterns **satisfy PUF requirements**

- The DRAM Latency PUF **generates PUF responses in 88.2ms**

Results
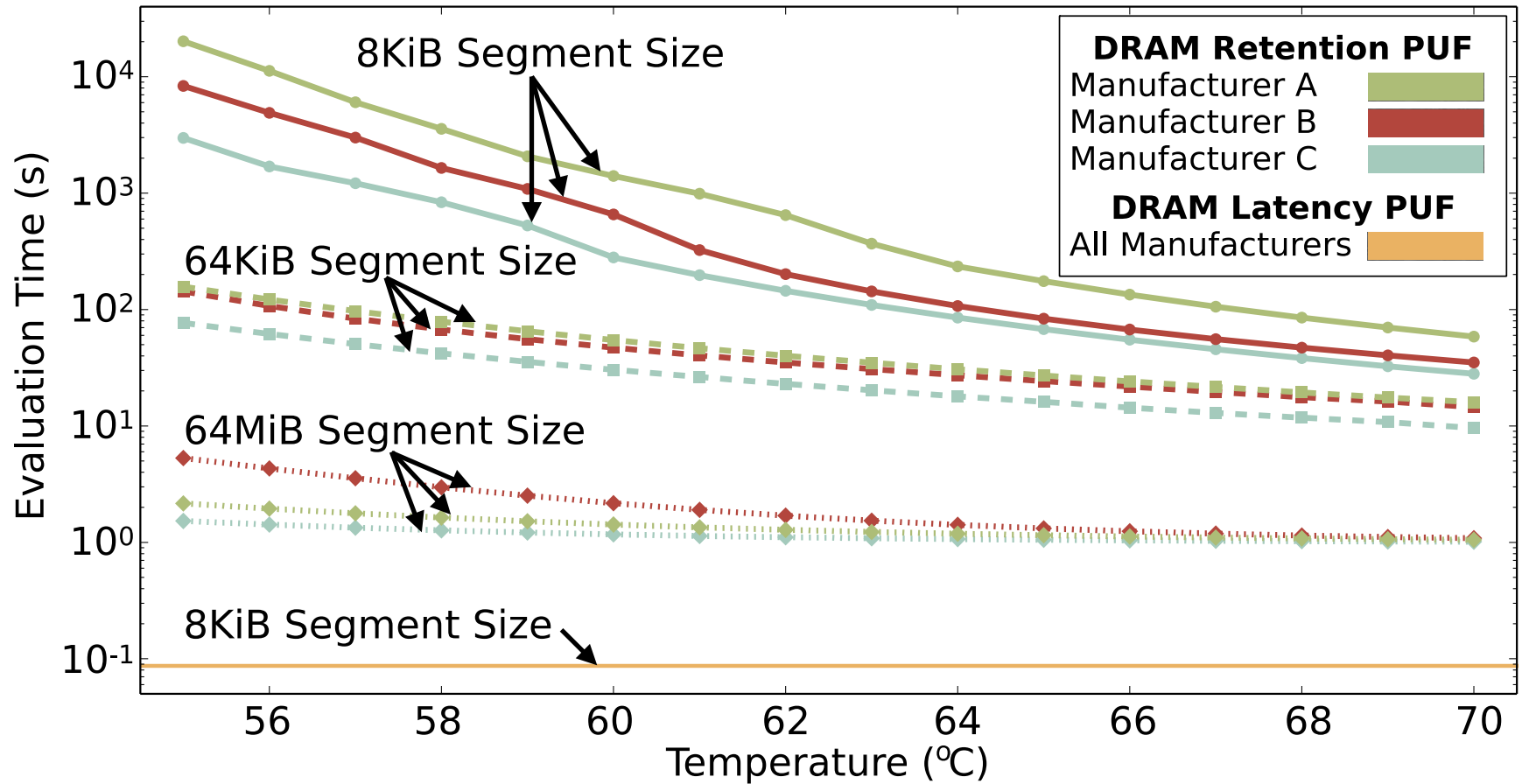
- DL-PUF is **orders of magnitude faster** than prior DRAM PUFs!

The DRAM Latency PUF:

Quickly Evaluating Physical Unclonable Functions by Exploiting the Latency-Reliability Tradeoff in Modern Commodity DRAM Devices

Jeremie S. Kim   **Minesh Patel**

Hasan Hassan   **Onur Mutlu**

HPCA 2018

QR Code for the paper
https://people.inf.ethz.ch/omutlu/pub/dram-latency-puf_hpca18.pdf

Systems@**ETH** *zürich*

ETH *zürich*

SAFARI

Carnegie Mellon

DRAM Latency PUFs

- Jeremie S. Kim, Minesh Patel, Hasan Hassan, and Onur Mutlu,
  "The DRAM Latency PUF: Quickly Evaluating Physical Unclonable Functions by Exploiting the Latency-Reliability Tradeoff in Modern DRAM Devices"
  Proceedings of the 24th International Symposium on High-Performance Computer Architecture (**HPCA**), Vienna, Austria, February 2018.
  [Lightning Talk Video]
  [Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)]

The DRAM Latency PUF:
Quickly Evaluating Physical Unclonable Functions
by Exploiting the Latency-Reliability Tradeoff in Modern Commodity DRAM Devices

Jeremie S. Kim[†§]   Minesh Patel[§]   Hasan Hassan[§]   Onur Mutlu[§†]
[†]Carnegie Mellon University   [§]ETH Zürich

Reducing Refresh Latency

On Reducing Refresh Latency

- Anup Das, Hasan Hassan, and Onur Mutlu,
  "VRL-DRAM: Improving DRAM Performance via Variable Refresh Latency"
  Proceedings of the 55th Design Automation Conference (**DAC**), San Francisco, CA, USA, June 2018.

VRL-DRAM: Improving DRAM Performance via Variable Refresh Latency

Anup Das
Drexel University
Philadelphia, PA, USA
anup.das@drexel.edu

Hasan Hassan
ETH Zürich
Zürich, Switzerland
hhasan@ethz.ch

Onur Mutlu
ETH Zürich
Zürich, Switzerland
omutlu@gmail.com

Why the Long Memory Latency?

- **Reason 1: Design of DRAM Micro-architecture**
  - Goal: Maximize capacity/area, not minimize latency

- **Reason 2: "One size fits all" approach to latency specification**
  - Same latency parameters for all temperatures
  - Same latency parameters for all DRAM chips (e.g., rows)
  - Same latency parameters for all parts of a DRAM chip
  - Same latency parameters for all supply voltage levels
  - Same latency parameters for all application data
  - …

Tiered Latency DRAM

What Causes the Long Latency?

DRAM Chip
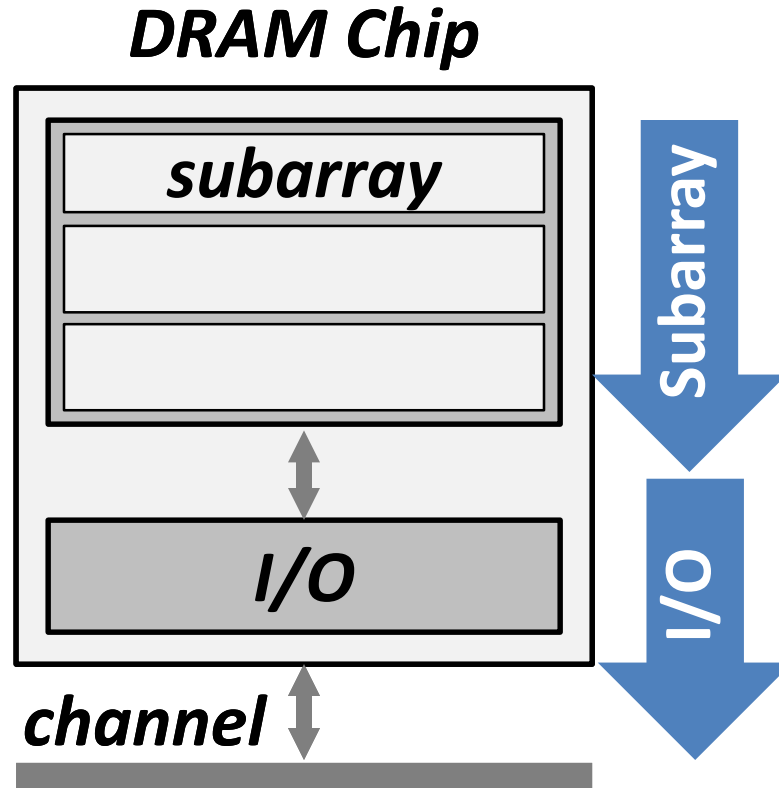
DRAM Latency = ~~Subarray Latency~~ + ~~I/O Latency~~

Dominant

Why is the Subarray So Slow?

Subarray

cell

row decoder

bitline: 512 cells

sense amplifier

Cell

row decoder

wordline

capacitor

access transistor

bitline

sense amplifier

large sense amplifier
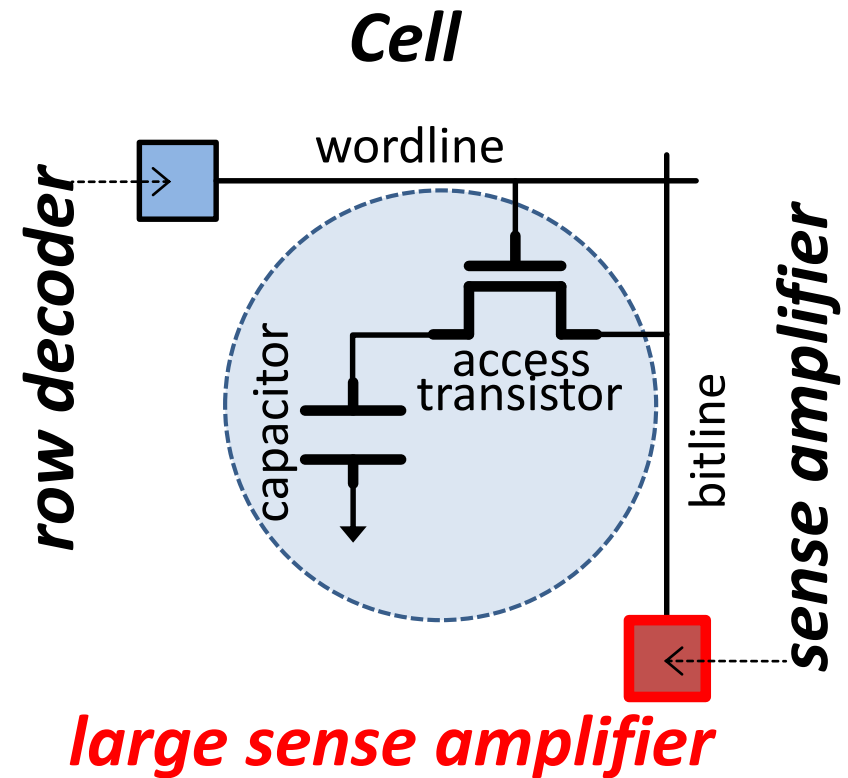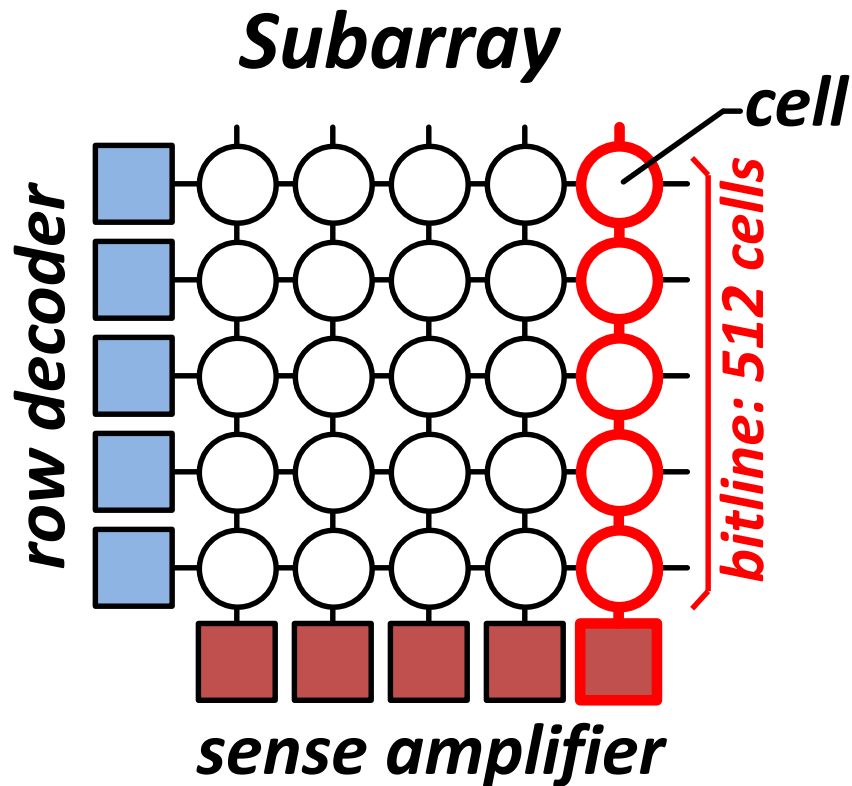
- **Long bitline**
  - **Amortizes sense amplifier cost → Small area**
  - **Large bitline capacitance → High latency & power**

Trade-Off: Area (Die Size) vs. Latency

Long Bitline   **Short Bitline**

Faster

Smaller

Trade-Off: Area vs. Latency
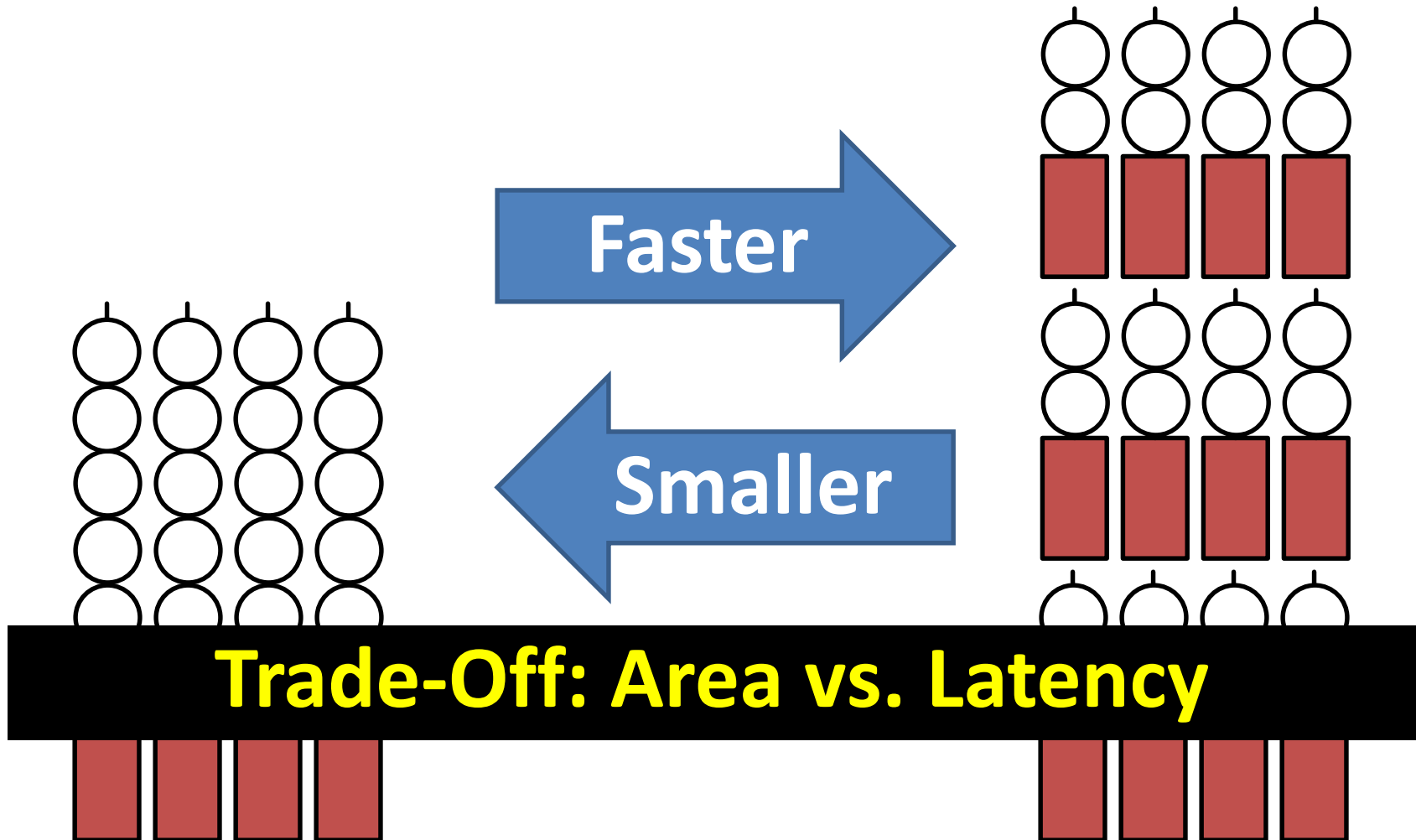
Trade-Off: Area (Die Size) vs. Latency

Approximating the Best of Both Worlds
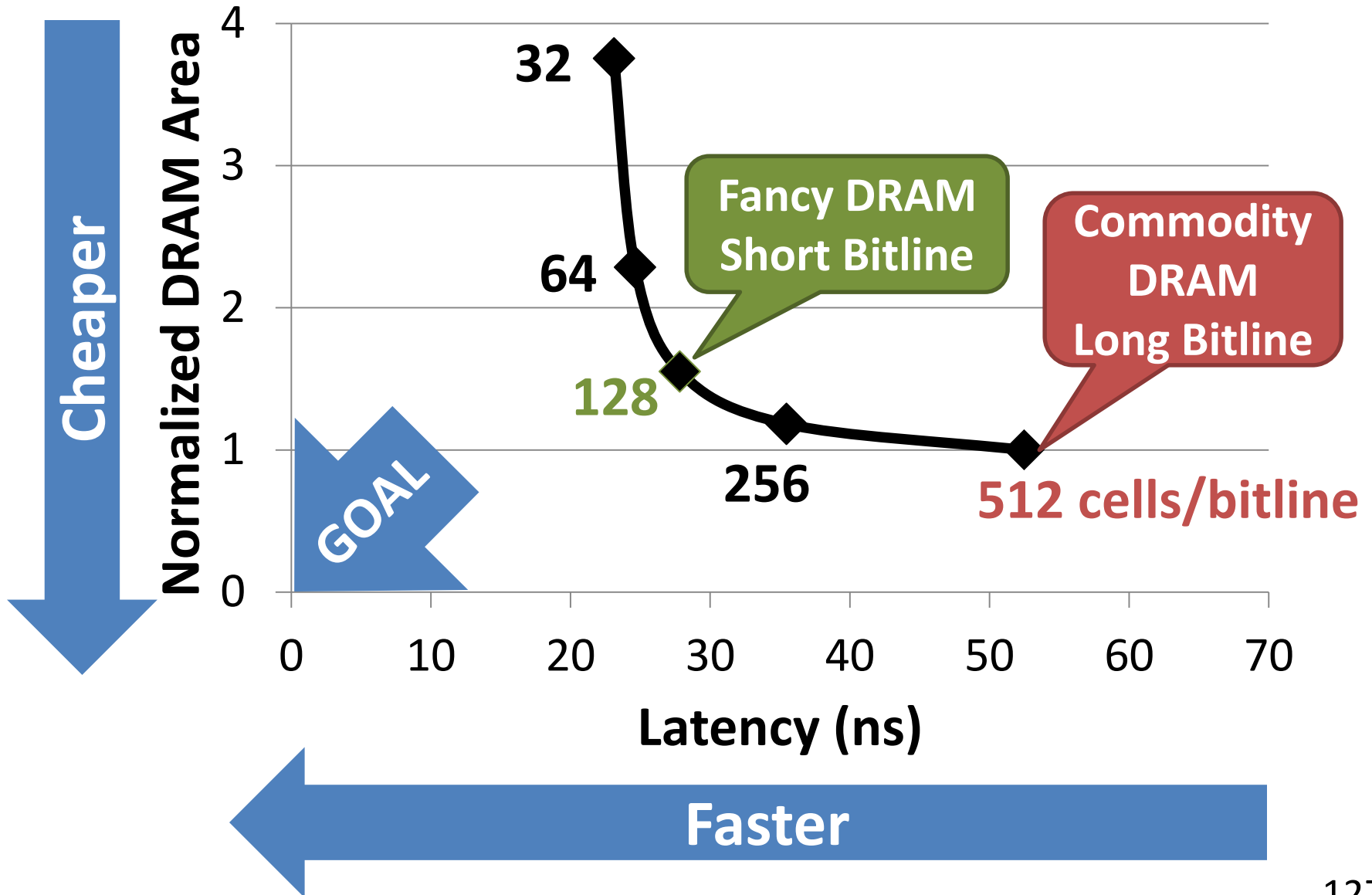
Long Bitline

Our Proposal

Short Bitline

*Small Area*

~~*Large Area*~~
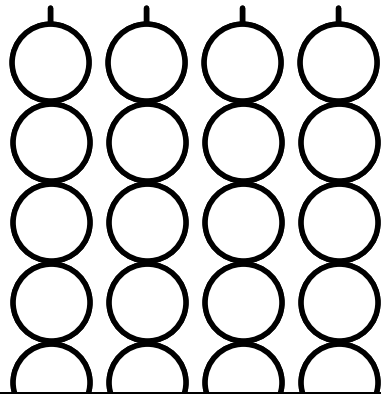
~~*High Latency*~~

Low Latency
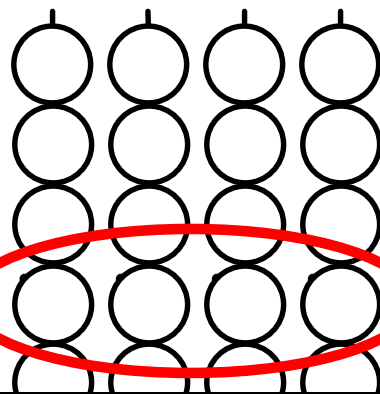
Need Isolation

Add Isolation Transistors

tline → Fast

128
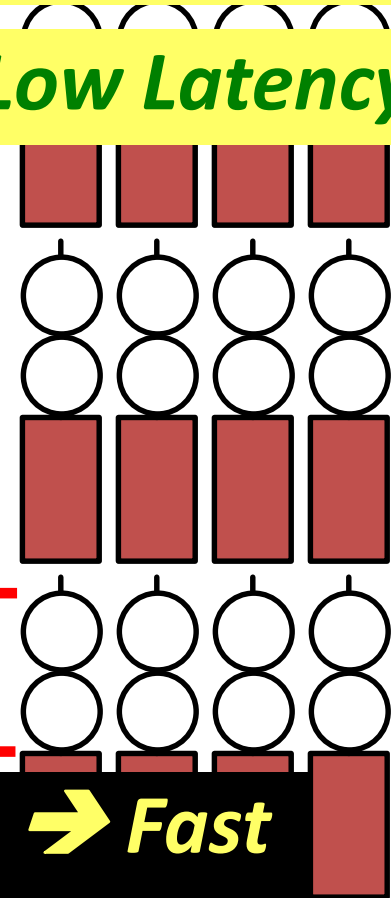
Approximating the Best of Both Worlds

Long Bitline **Tiered-Latency DRAM** **Short Bitline**

Small Area   *Small Area*   ~~*Large Area*~~

~~*High Latency*~~   *Low Latency*   *Low Latency*

Small area using long bitline

Low Latency
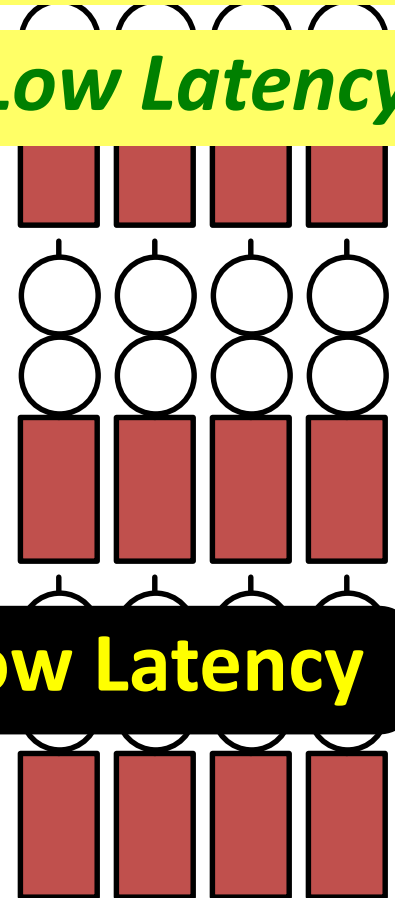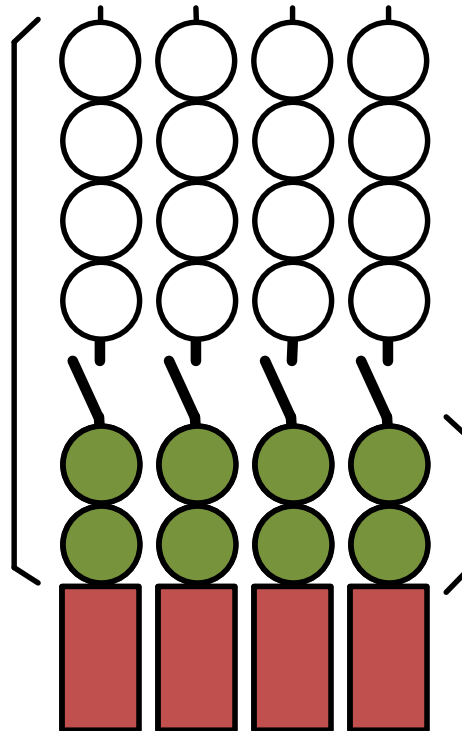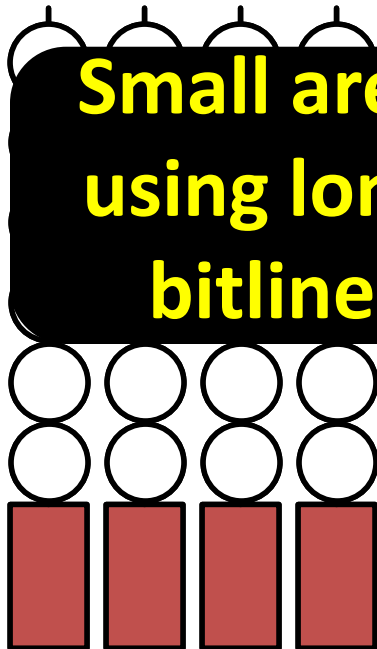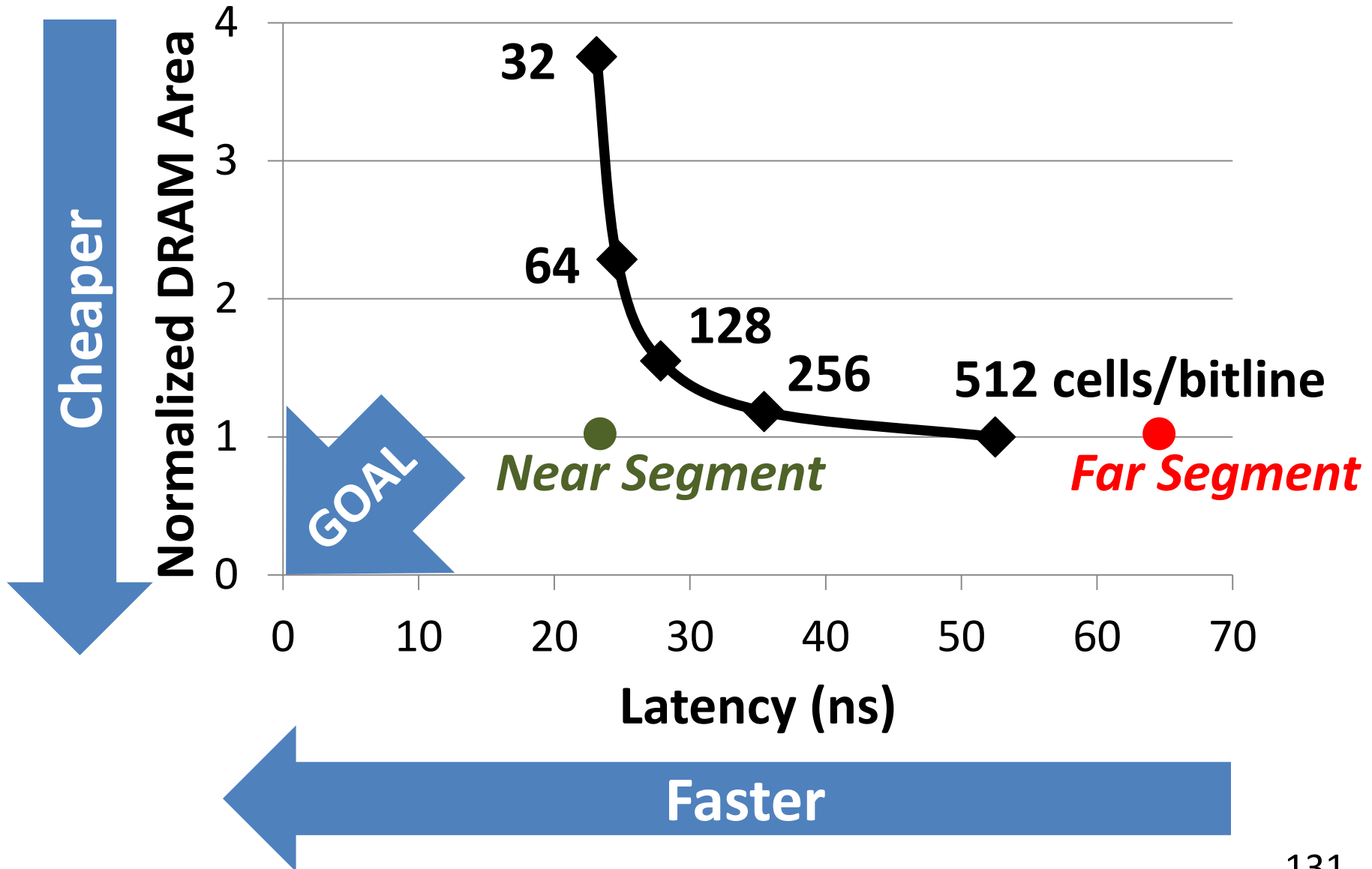
Commodity DRAM vs. TL-DRAM [HPCA 2013]

- **DRAM Latency** (tRC) • **DRAM Power**
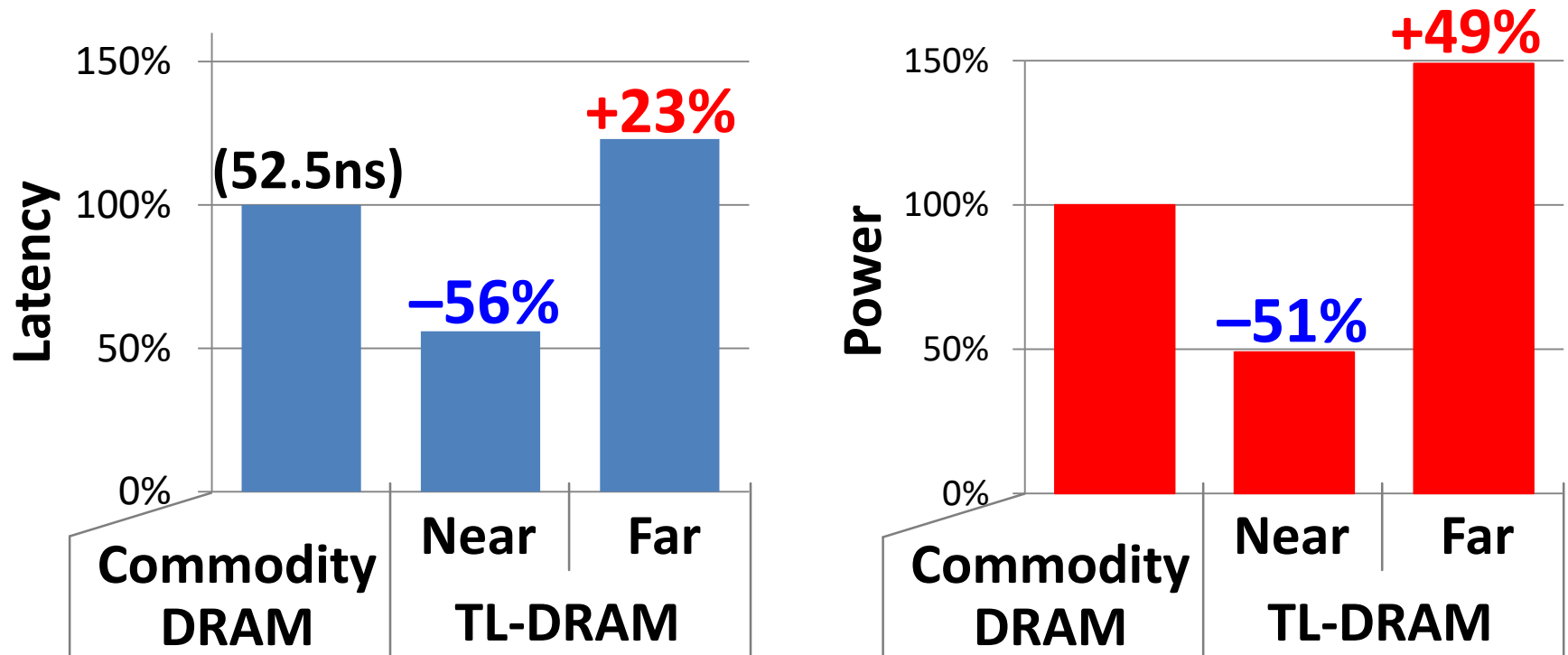
- **DRAM Area Overhead**

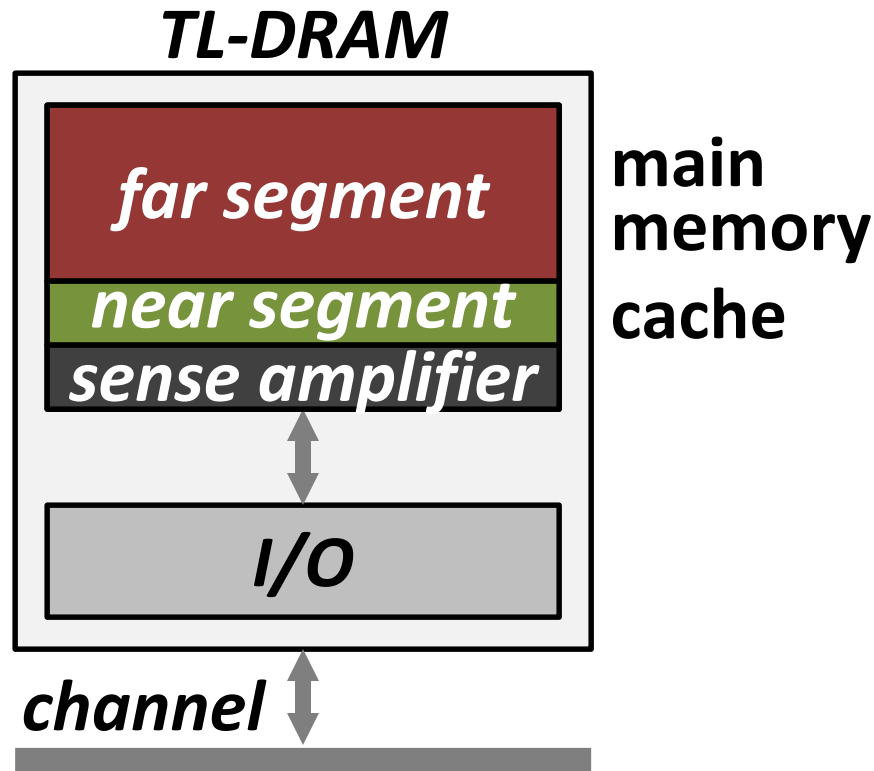 **~3%**: mainly due to the isolation transistors

Trade-Off: Area (Die-Area) vs. Latency

Leveraging Tiered-Latency DRAM

- TL-DRAM is a **substrate** that can be leveraged by the hardware and/or software

- Many potential uses

  1. Use near segment as hardware-managed *inclusive* cache to far segment
  2. Use near segment as hardware-managed *exclusive* cache to far segment
  3. Profile-based page mapping by operating system
 4. Simply replace DRAM with TL-DRAM

Lee+, "Tiered-Latency DRAM: A Low Latency and Low Cost DRAM Architecture," HPCA 2013.

Near Segment as Hardware-Managed Cache

TL-DRAM

far segment → main memory

near segment → cache

sense amplifier

I/O

channel

- **Challenge 1:** How to efficiently migrate a row between segments?
- **Challenge 2:** How to efficiently manage the cache?

Inter-Segment Migration

- **Goal:** Migrate source row into destination row
- **Naïve way:** Memory controller reads the source row *byte by byte* and writes to destination row *byte by byte*
  → *High latency*
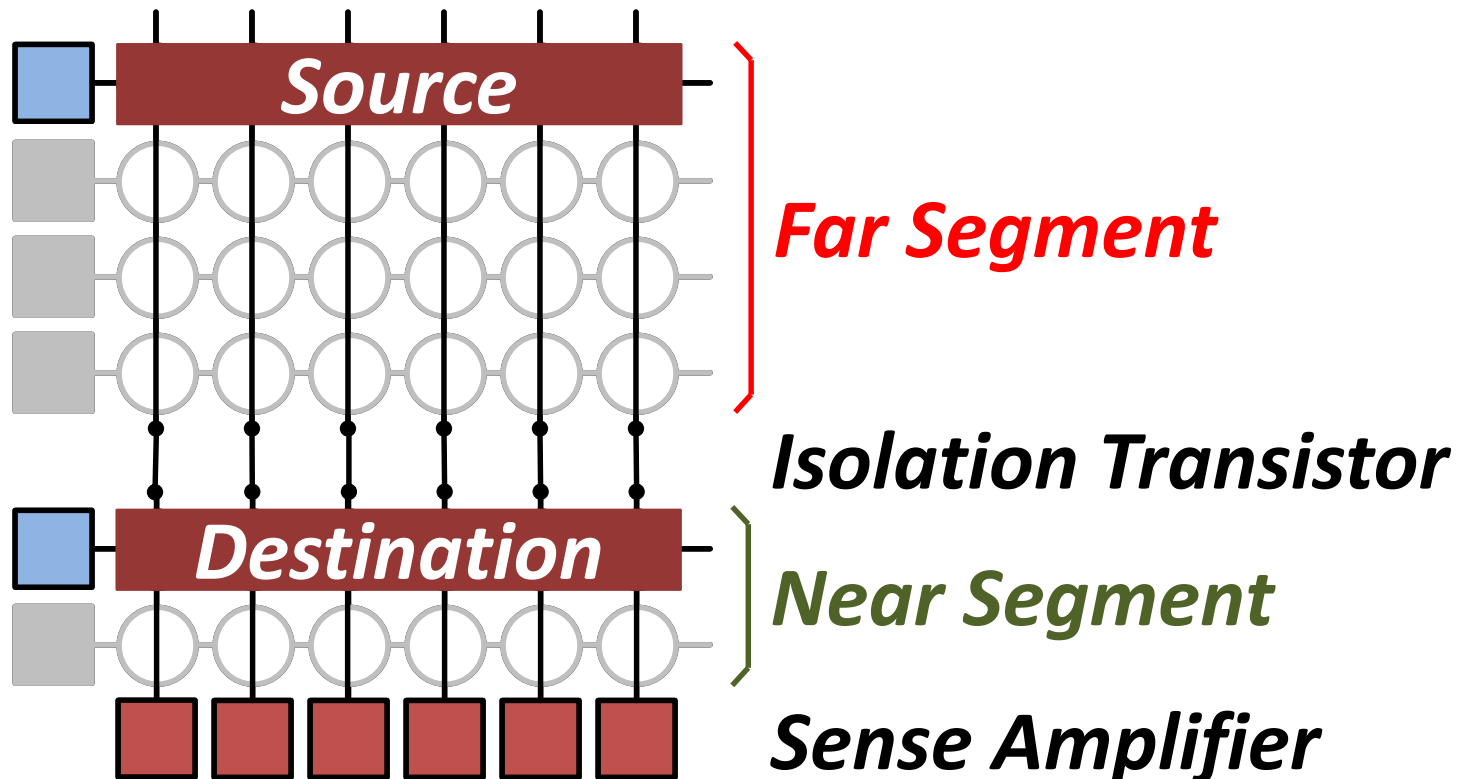
Source

Far Segment

Isolation Transistor

Destination

Near Segment

Sense Amplifier

Inter-Segment Migration

- **Our way:**
  - Source and destination cells *share bitlines*
 - Transfer data from source to destination across *shared bitlines* concurrently
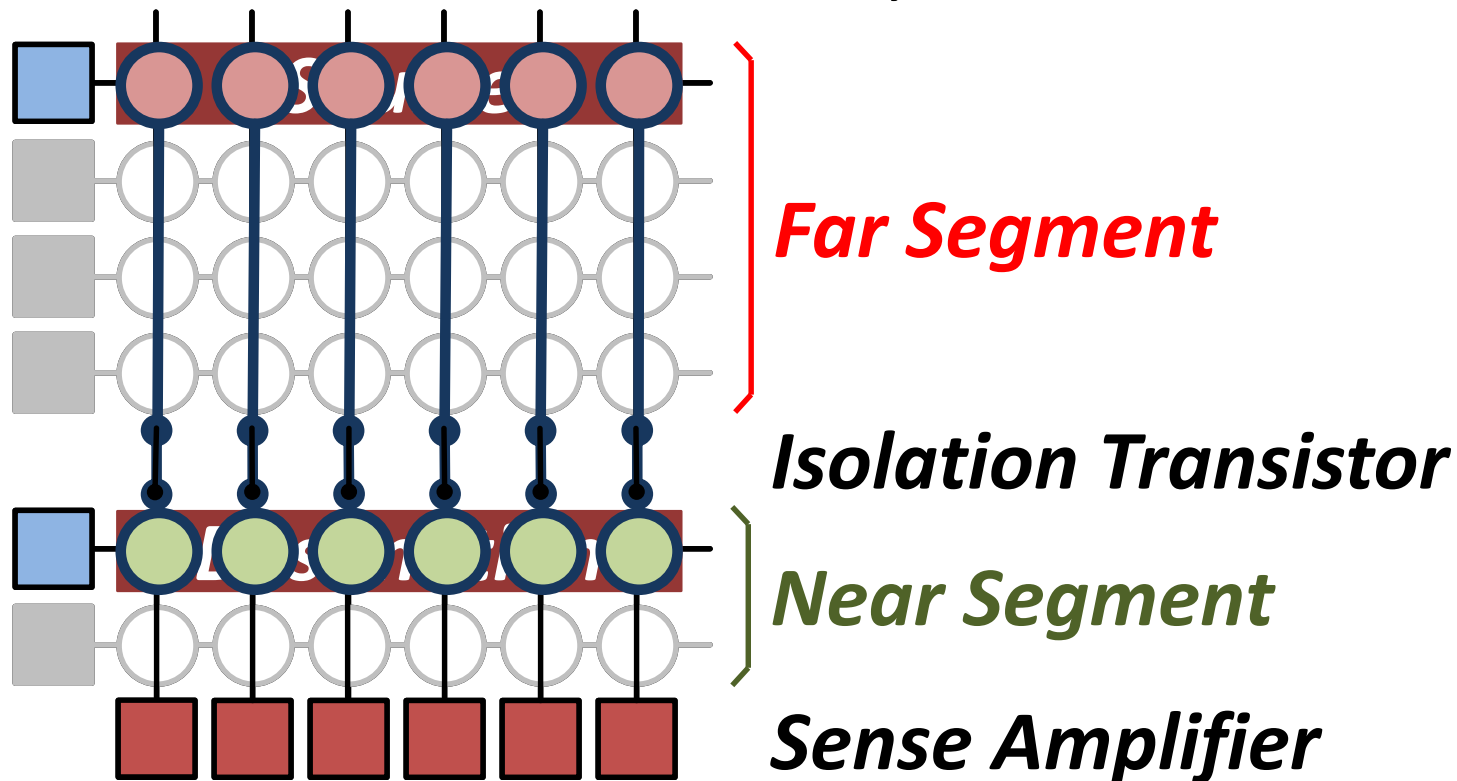
Far Segment

Isolation Transistor

Near Segment

Sense Amplifier

Inter-Segment Migration

- **Our way:**
  - Source and destination cells *share bitlines*
  - Transfer data from source to destination *shared bitlines* concurrently

Step 1: Activate source row

Migration is overlapped with source row access
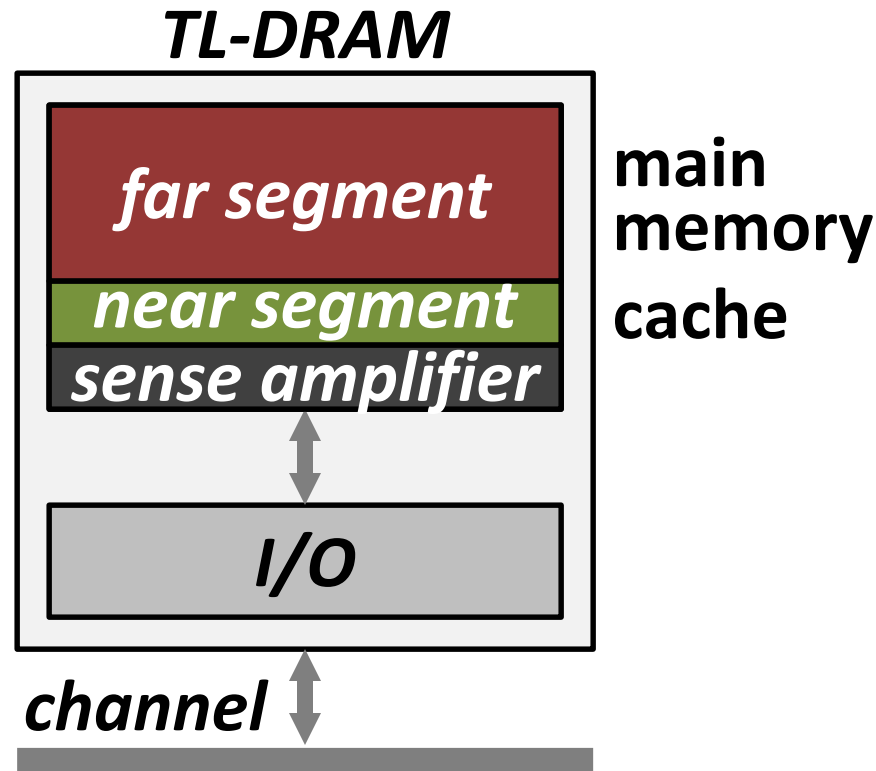
Additional ~4ns over row access latency
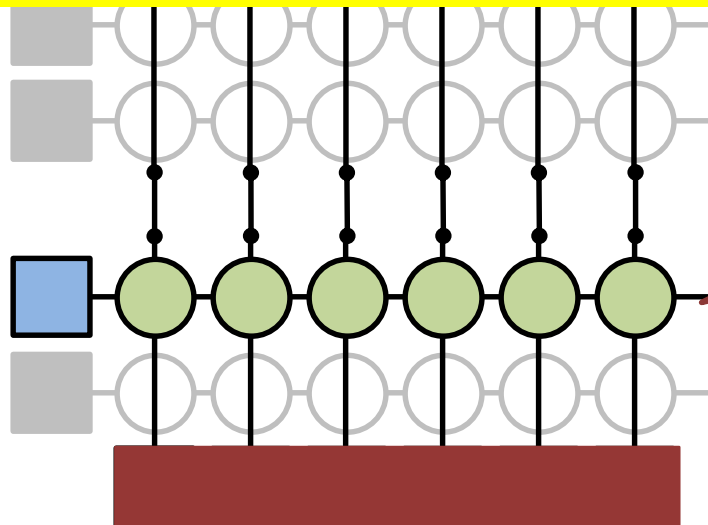
Step 2: Activate destination row to connect cell and bitline
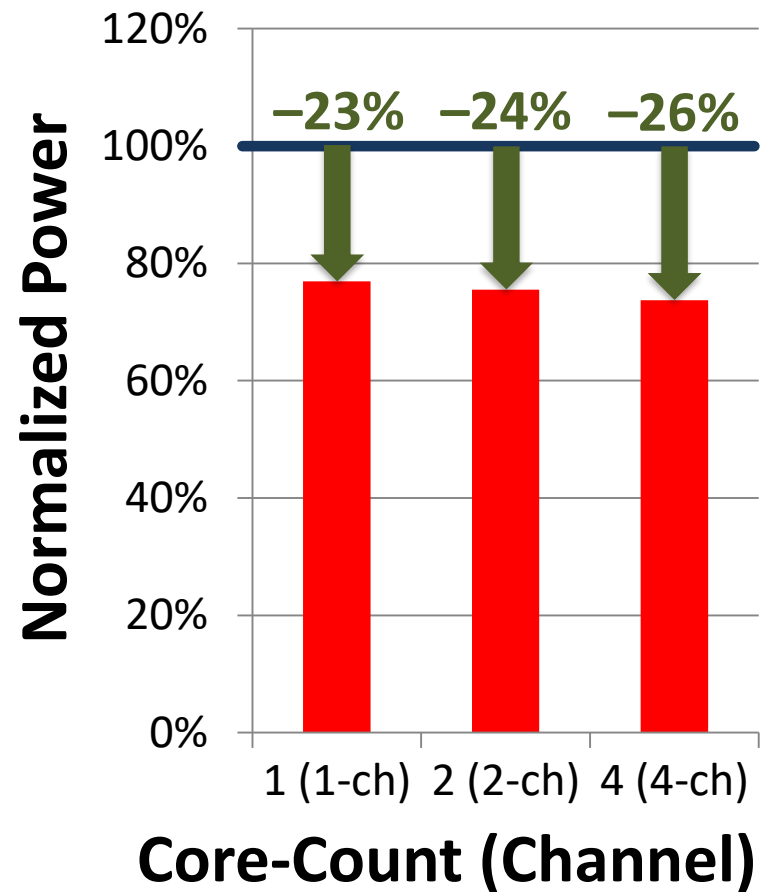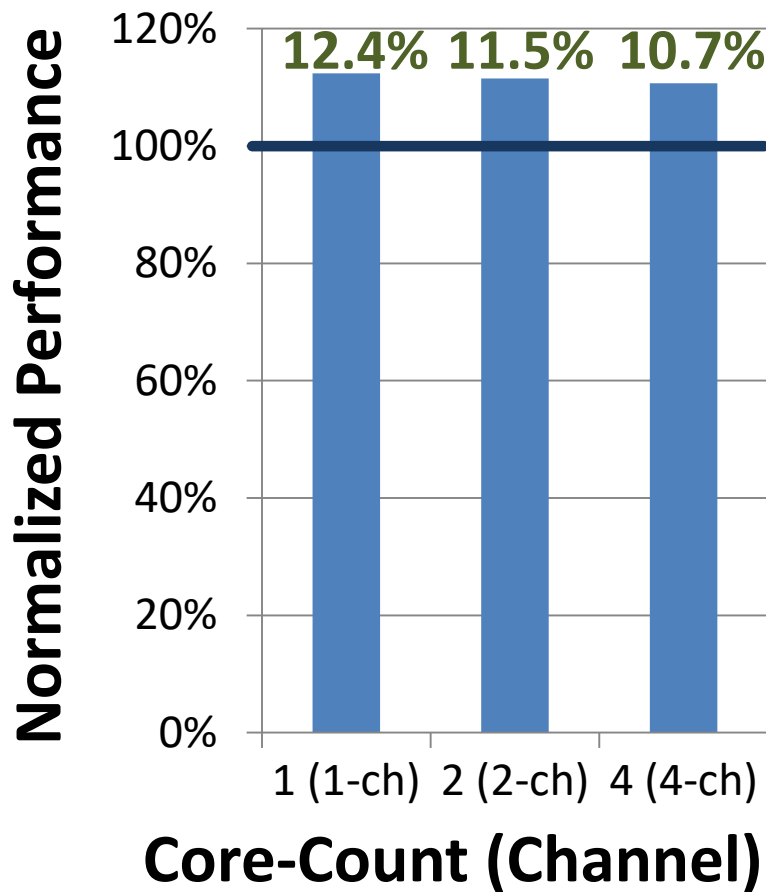
Isolation Transistor

Near Segment

Sense Amplifier

Near Segment as Hardware-Managed Cache

TL-DRAM

far segment

near segment

sense amplifier

main memory
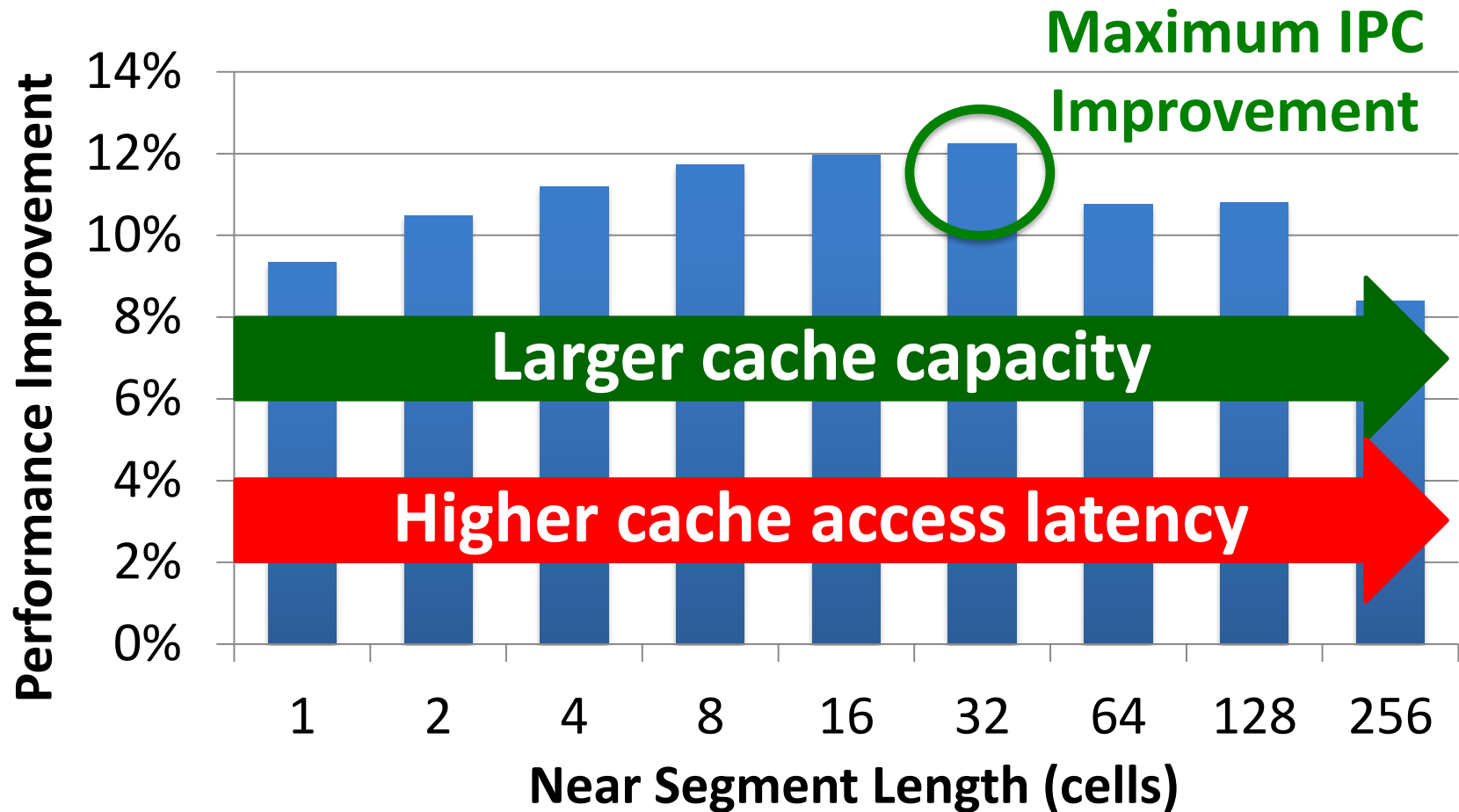
cache

I/O

channel

- **Challenge 1:** How to efficiently migrate a row between segments?
- **Challenge 2:** How to efficiently manage the cache?

Performance & Power Consumption

Using near segment as a cache improves performance and reduces power consumption

Lee+, "Tiered-Latency DRAM: A Low Latency and Low Cost DRAM Architecture," HPCA 2013.

Single-Core: Varying Near Segment Length

By adjusting the near segment length, we can trade off cache capacity for cache latency

More on TL-DRAM

- Donghyuk Lee, Yoongu Kim, Vivek Seshadri, Jamie Liu, Lavanya Subramanian, and Onur Mutlu,
  "Tiered-Latency DRAM: A Low Latency and Low Cost DRAM Architecture"
  Proceedings of the 19th International Symposium on High-Performance Computer Architecture (**HPCA**), Shenzhen, China, February 2013. Slides (pptx)

Tiered-Latency DRAM: A Low Latency and Low Cost DRAM Architecture
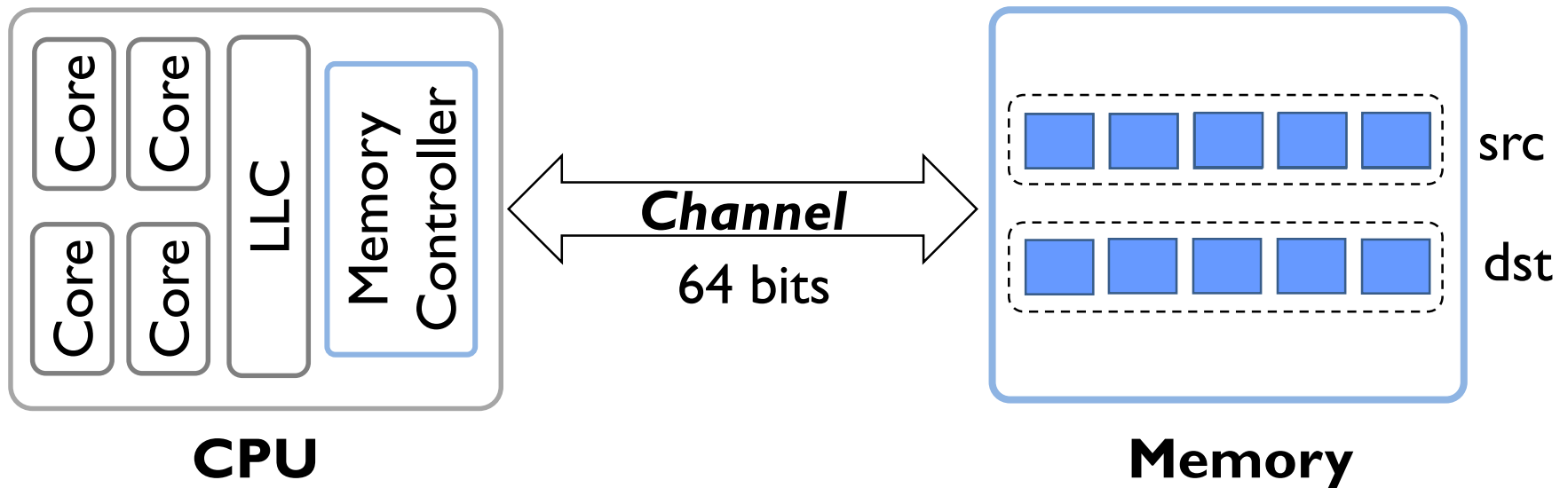
Donghyuk Lee   Yoongu Kim   Vivek Seshadri   Jamie Liu   Lavanya Subramanian   Onur Mutlu

Carnegie Mellon University

LISA: Low-Cost Inter-Linked Subarrays
[HPCA 2016]
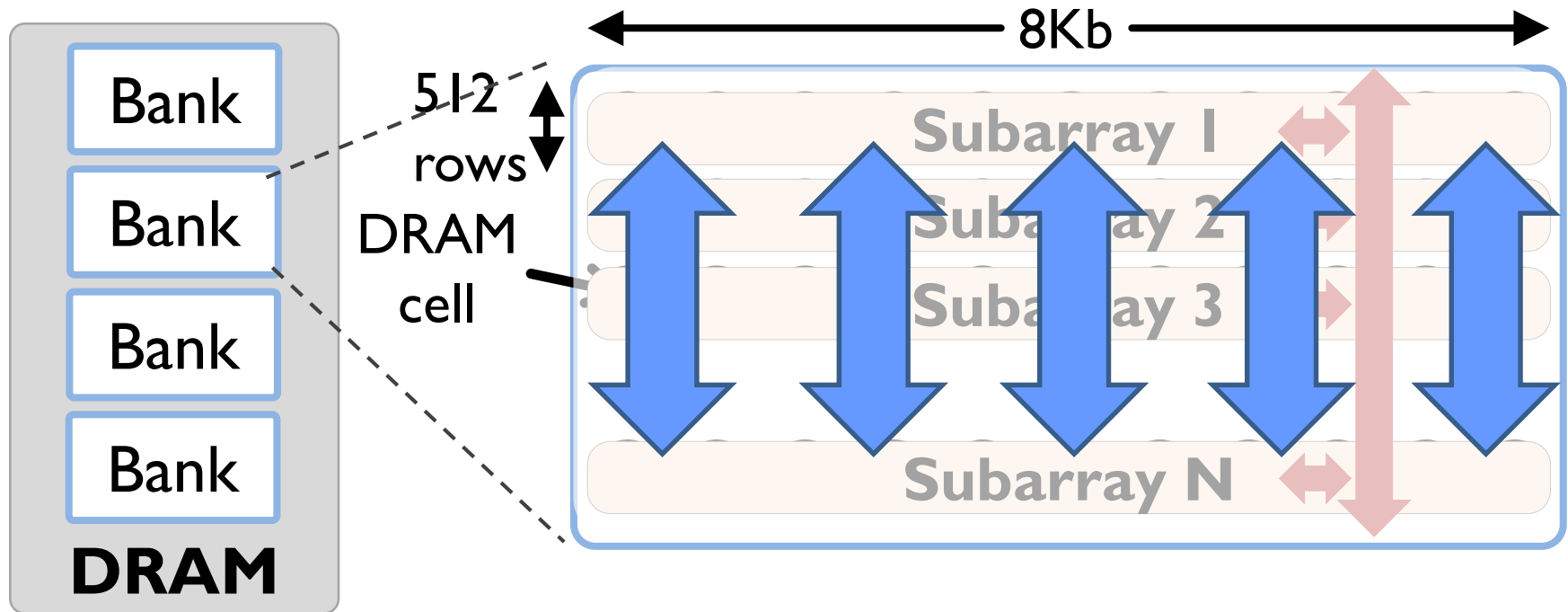
Problem: Inefficient Bulk Data Movement

Bulk data movement is a key operation in many applications

– *memmove & memcpy:* 5% cycles in Google's datacenter [Kanev+ ISCA'15]

CPU

Memory

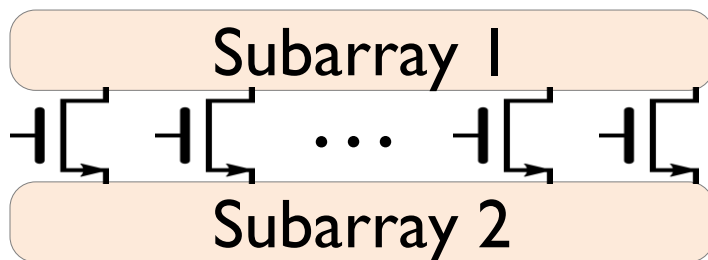**Long latency and high energy**

Moving Data Inside DRAM?

Goal: Provide a new substrate to enable wide connectivity between subarrays

Key Idea and Applications

- **Low-cost Inter-linked subarrays (LISA)**
  - Fast bulk data movement between subarrays
  - Wide datapath via isolation transistors: 0.8% DRAM chip area

- LISA is a **versatile substrate** → new applications
  Fast bulk data copy: Copy latency 1.363ms→0.148ms (9.2x)
      → 66% speedup, -55% DRAM energy

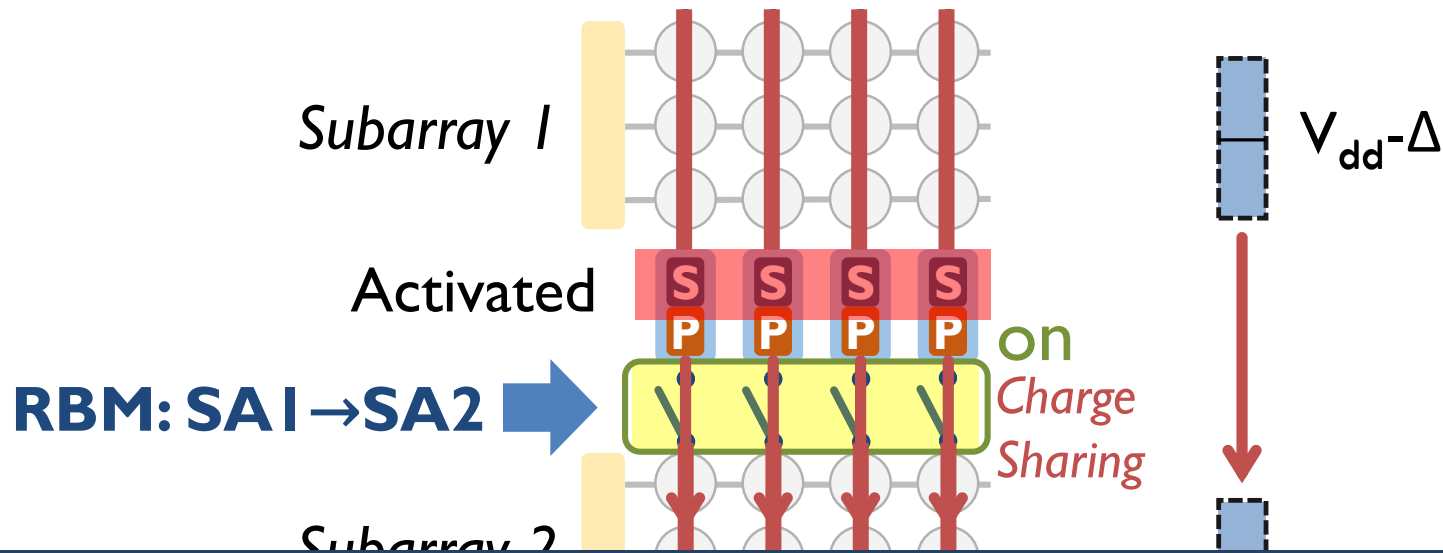 In-DRAM caching: Hot data access latency 48.7ns→21.5ns (2.2x)
      → 5% speedup

  Fast precharge: Precharge latency 13.1ns→5.0ns (2.6x)
      → 8% speedup

New DRAM Command to Use LISA
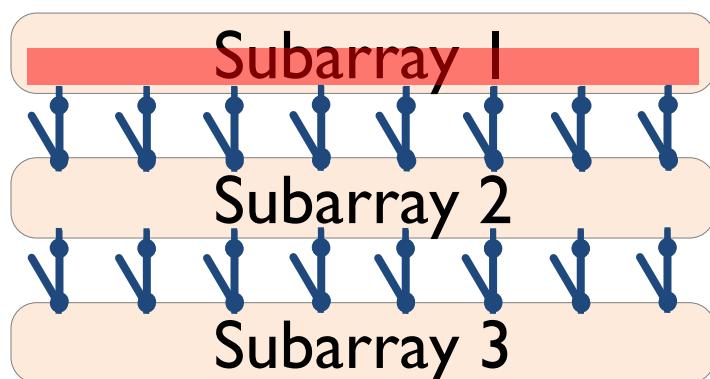
Row Buffer Movement (RBM): Move a row of data in an activated row buffer to a precharged one

Subarray 1

Activated

RBM: SA1→SA2

on *Charge Sharing*

$V_{dd}-\Delta$

Subarray 2

## RBM transfers an entire row b/w subarrays

RBM Analysis

- The range of RBM depends on the DRAM design
  - Multiple RBMs to move data across > 3 subarrays

- Validated with SPICE using worst-case cells
  - NCSU FreePDK 45nm library

- **4KB data in 8ns** (w/ 60% guardband)
  → **500 GB/s, 26x** bandwidth of a DDR4-2400 channel

- **0.8%** DRAM chip area overhead [O+ ISCA'14]

1. Rapid Inter-Subarray Copying (RISC)

- **Goal:** Efficiently copy a row across subarrays

- **Key idea**: Use *RBM* to form a new command sequence

1 **Activate** src row
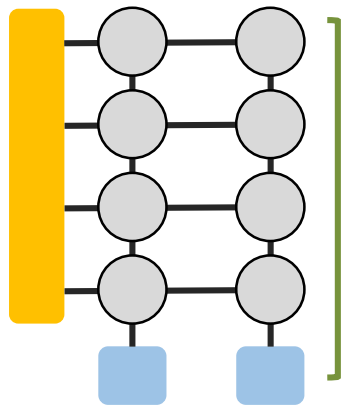
Subarray 1

src row

2 **RBM** SA1→SA2
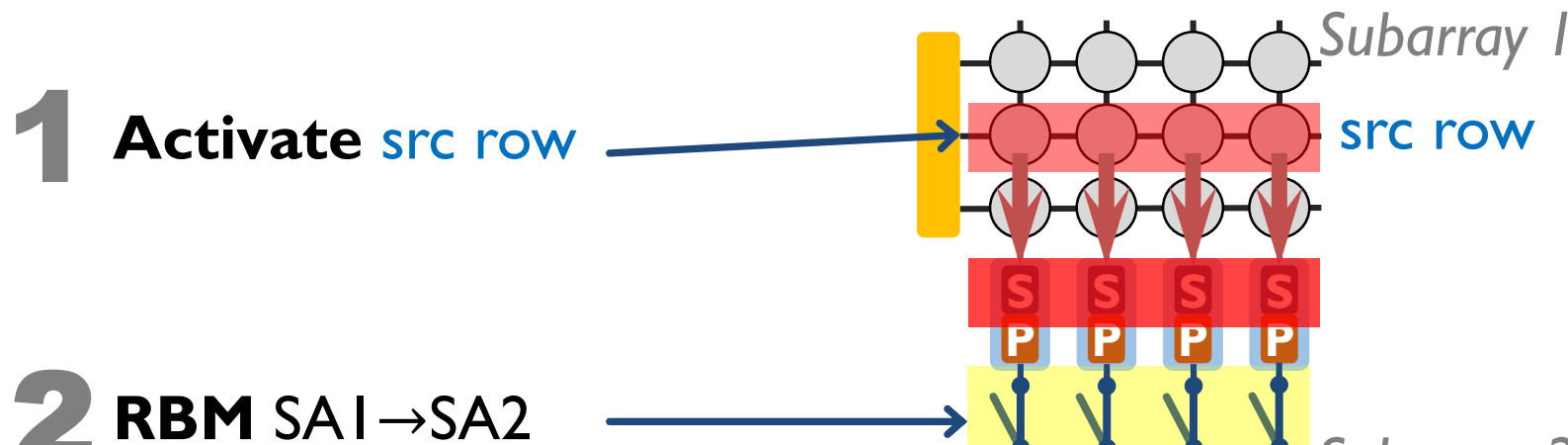
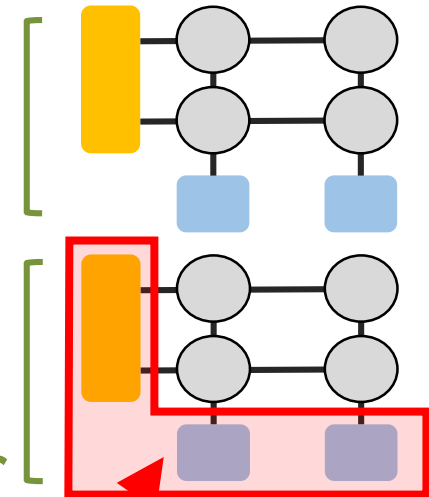**Reduces row-copy latency by 9.2x, DRAM energy by 48.1x**

2. Variable Latency DRAM (VILLA)

- **Goal**: Reduce DRAM latency with low area overhead

- **Motivation**: Trade-off between area and latency

Long Bitline (DDRx)

Short Bitline (RLDRAM)

Shorter bitlines → faster **activate** and **precharge** time

High area overhead: >40%

2. Variable Latency DRAM (VILLA)

- **Key idea**: Reduce access latency of hot data via a **heterogeneous DRAM** design [Lee+ HPCA'13, Son+ ISCA'13]

- **VILLA**: Add fast subarrays as a **cache** in each bank

Slow Subarray

512 rows
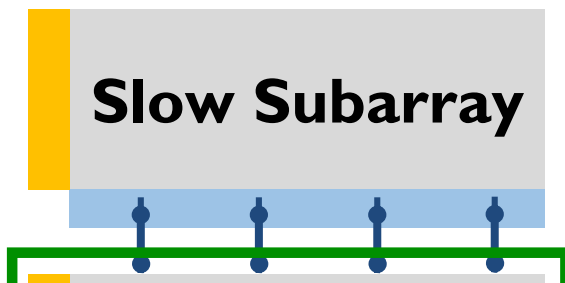
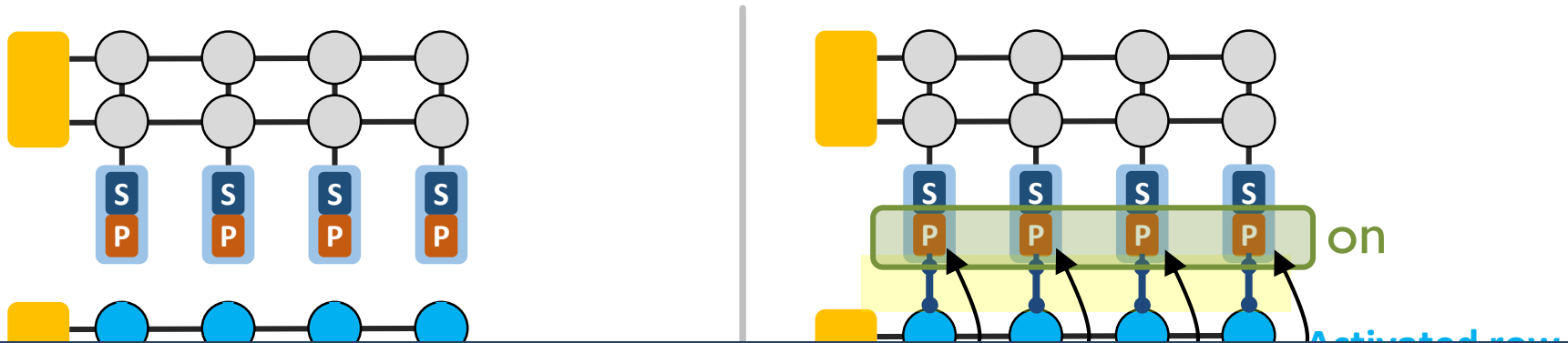**Challenge**: VILLA cache requires frequent movement of data rows

Reduces hot data access latency by 2.2x at only 1.6% area overhead

3. Linked Precharge (LIP)

- **Problem**: The precharge time is limited by the strength of one precharge unit

- **Linked Precharge (LIP)**: LISA precharges a subarray using multiple precharge units

on

Activated row

Reduces precharge latency by 2.6x (43% guardband)

More on LISA

- Kevin K. Chang, Prashant J. Nair, Saugata Ghose, Donghyuk Lee, Moinuddin K. Qureshi, and Onur Mutlu,
  "Low-Cost Inter-Linked Subarrays (LISA): Enabling Fast Inter-Subarray Data Movement in DRAM"
  Proceedings of the 22nd International Symposium on High-Performance Computer Architecture (**HPCA**), Barcelona, Spain, March 2016.
  [Slides (pptx) (pdf)]
  [Source Code]

Low-Cost Inter-Linked Subarrays (LISA): Enabling Fast Inter-Subarray Data Movement in DRAM

Kevin K. Chang[†], Prashant J. Nair[⋆], Donghyuk Lee[†], Saugata Ghose[†], Moinuddin K. Qureshi[⋆], and Onur Mutlu[†]

[†]*Carnegie Mellon University*   [⋆]*Georgia Institute of Technology*
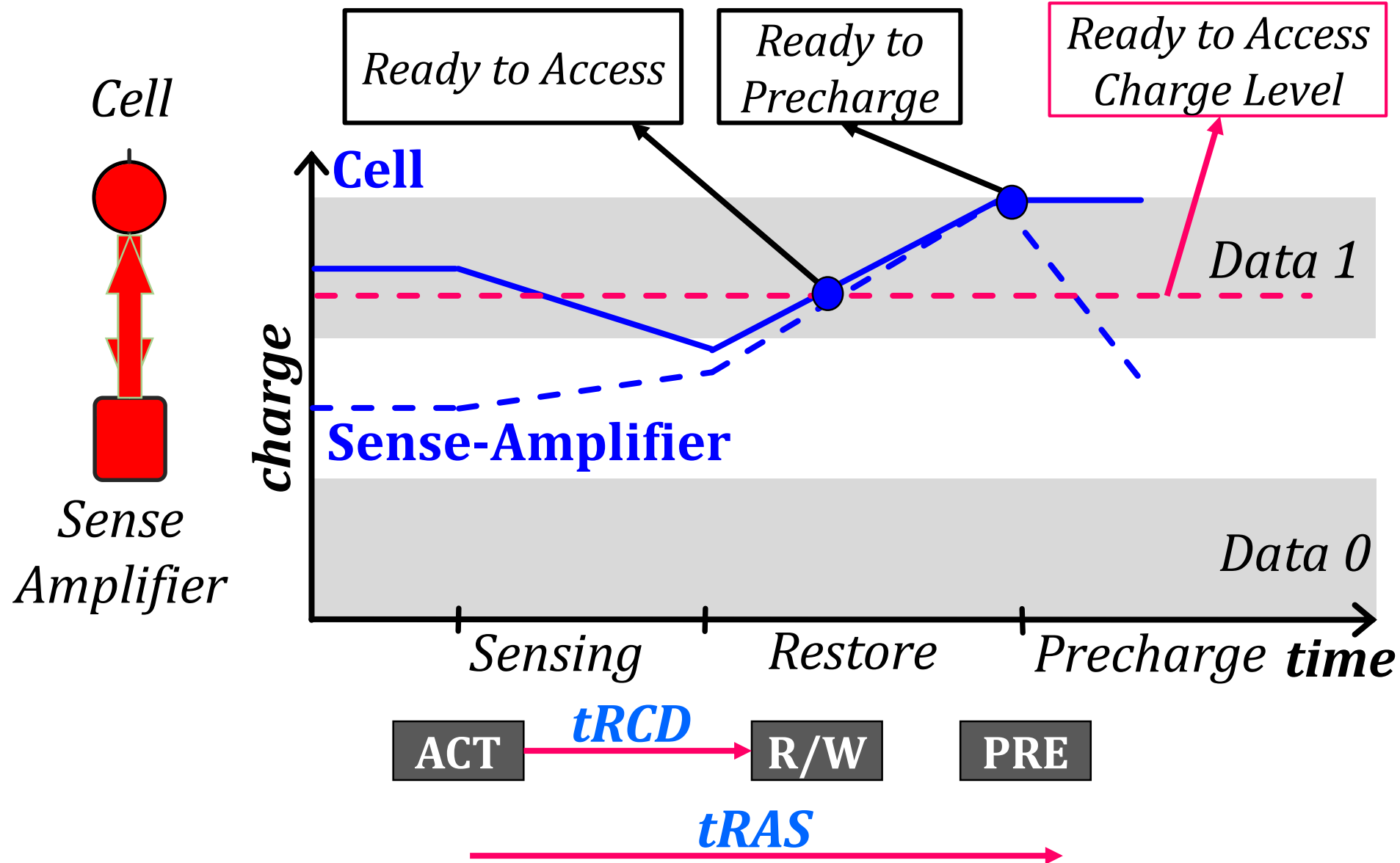
Reducing Memory Latency by Exploiting Memory Access Patterns

ChargeCache: Executive Summary

- **<u>Goal</u>**: Reduce average DRAM access latency with no modification to the existing DRAM chips

- **<u>Observations</u>**:
    1) A highly-charged DRAM row can be accessed with low latency
    2) A row's charge is restored when the row is accessed
    3) A recently-accessed row is likely to be accessed again**:**
       Row Level Temporal Locality (RLTL)

- **<u>Key Idea</u>**: Track recently-accessed DRAM rows and use lower timing parameters if such rows are accessed again

- **<u>ChargeCache</u>:**
    – Low cost & no modifications to the DRAM
    – Higher performance (**8.6-10.6%** on average for 8-core)
    – Lower DRAM energy (**7.9%** on average)

DRAM Charge over Time

Cell

Sense Amplifier

Ready to Access

Ready to Precharge

Ready to Access Charge Level

Cell

Sense-Amplifier

Data 1

Data 0

charge

Sensing *Restore* *Precharge* **time**

ACT *tRCD* R/W PRE

tRAS

SAFARI

154

Accessing Highly-charged Rows

Ready to Access

Ready to Precharge

Cell

Sense-Amplifier

charge

Data 1

Data 0

Sensing   *Restore*   *Precharge*   **time**

| ACT | *tRCD* → | R/W | | PRE |
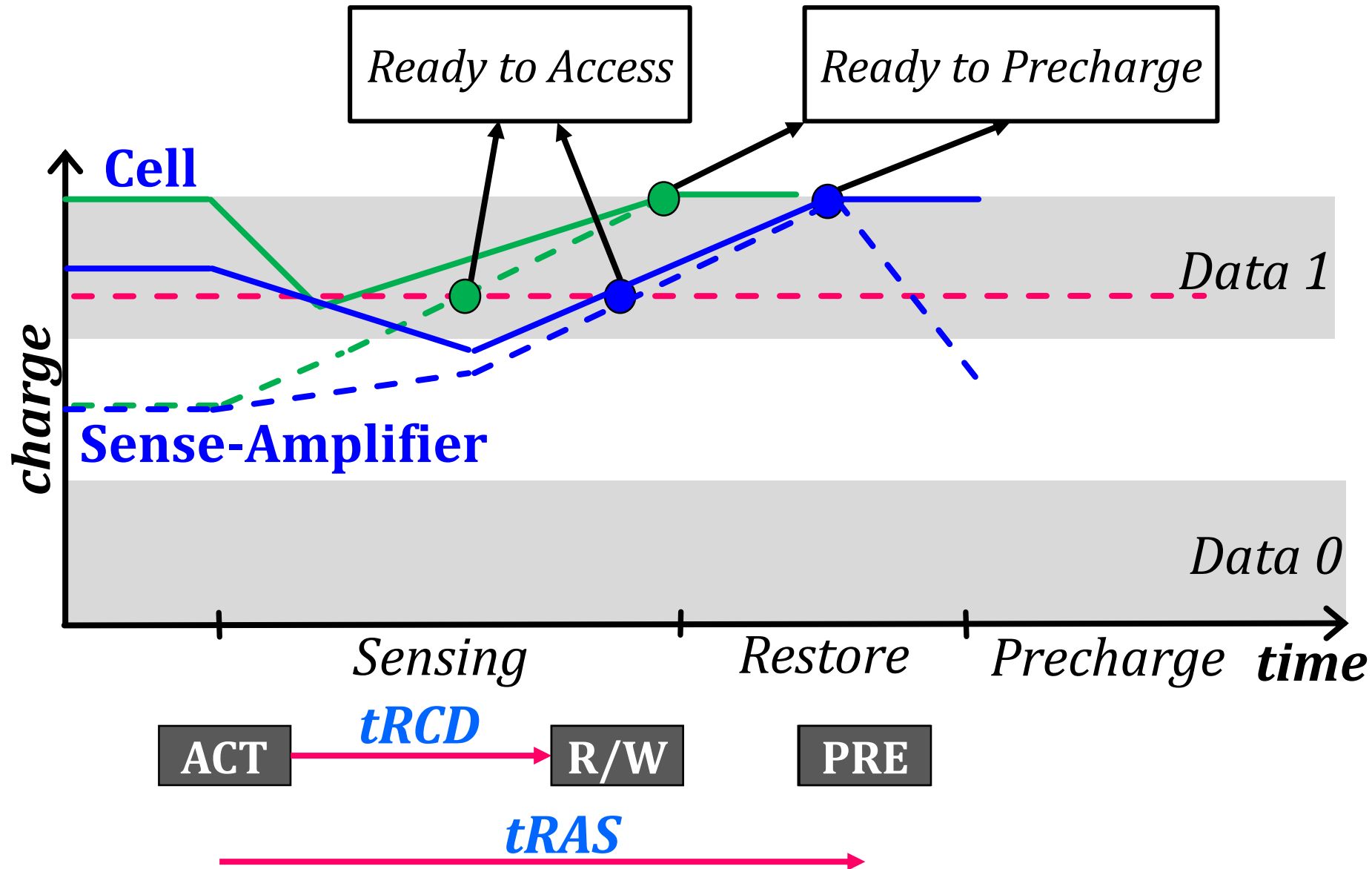
tRAS

Observation 1

A highly-charged DRAM row can be accessed with low latency

- tRCD: **44%**
- tRAS: **37%**
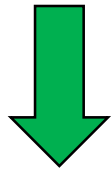
How does a row become highly-charged?

How Does a Row Become Highly-Charged?

DRAM cells **lose charge** over time

Two ways of restoring a row's charge:

- Refresh Operation

- Access
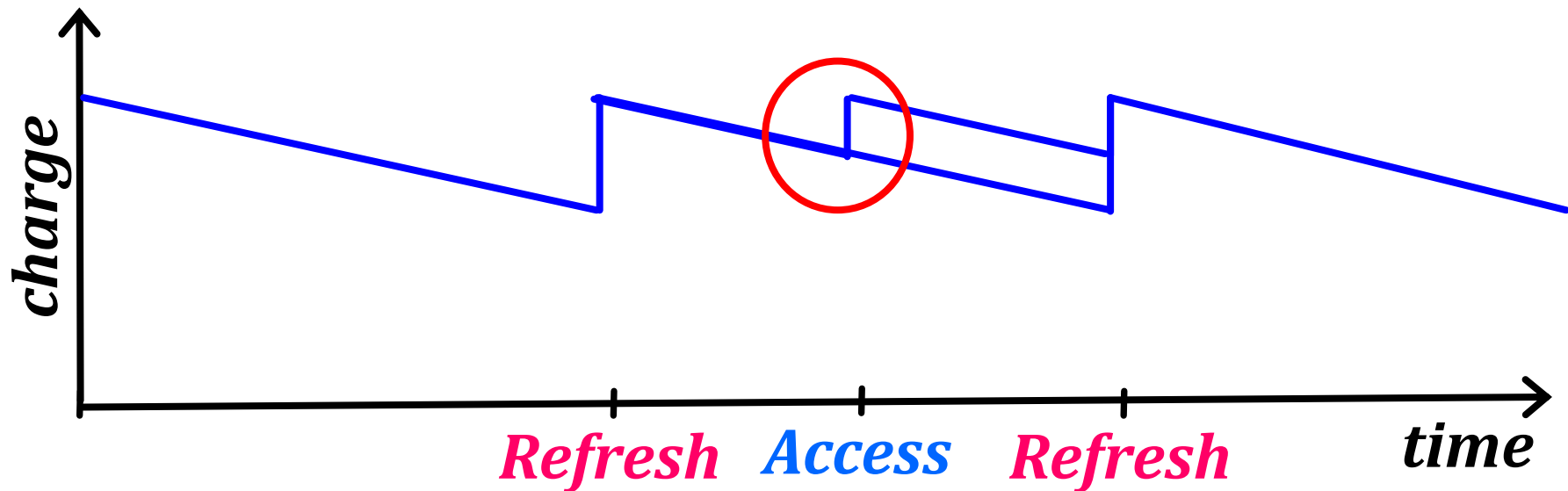
Observation 2

A row's charge is restored when the row is accessed

How likely is a recently-accessed row to be accessed again?

Row Level Temporal Locality (RLTL)

A **recently-accessed** DRAM row is likely to be accessed again.

- *t*-RLTL: Fraction of rows that are accessed within time *t* after their previous access

97%
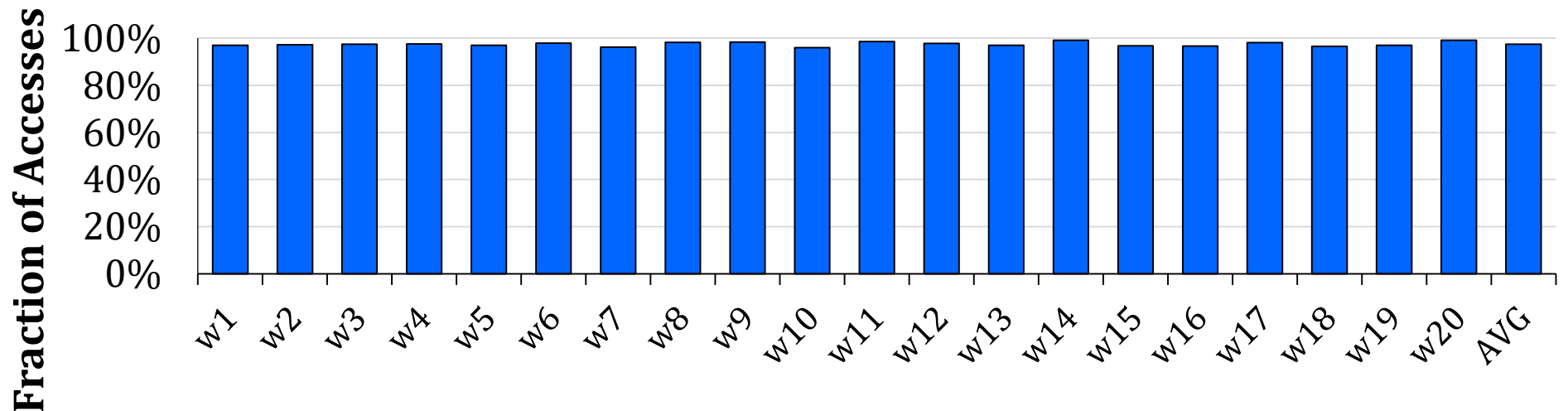
88ms RLTL for eight-core workloads

SAFARI

Key Idea

Track **recently-accessed** DRAM rows and use **lower timing parameters** if such rows are accessed again

SAFARI

ChargeCache Overview
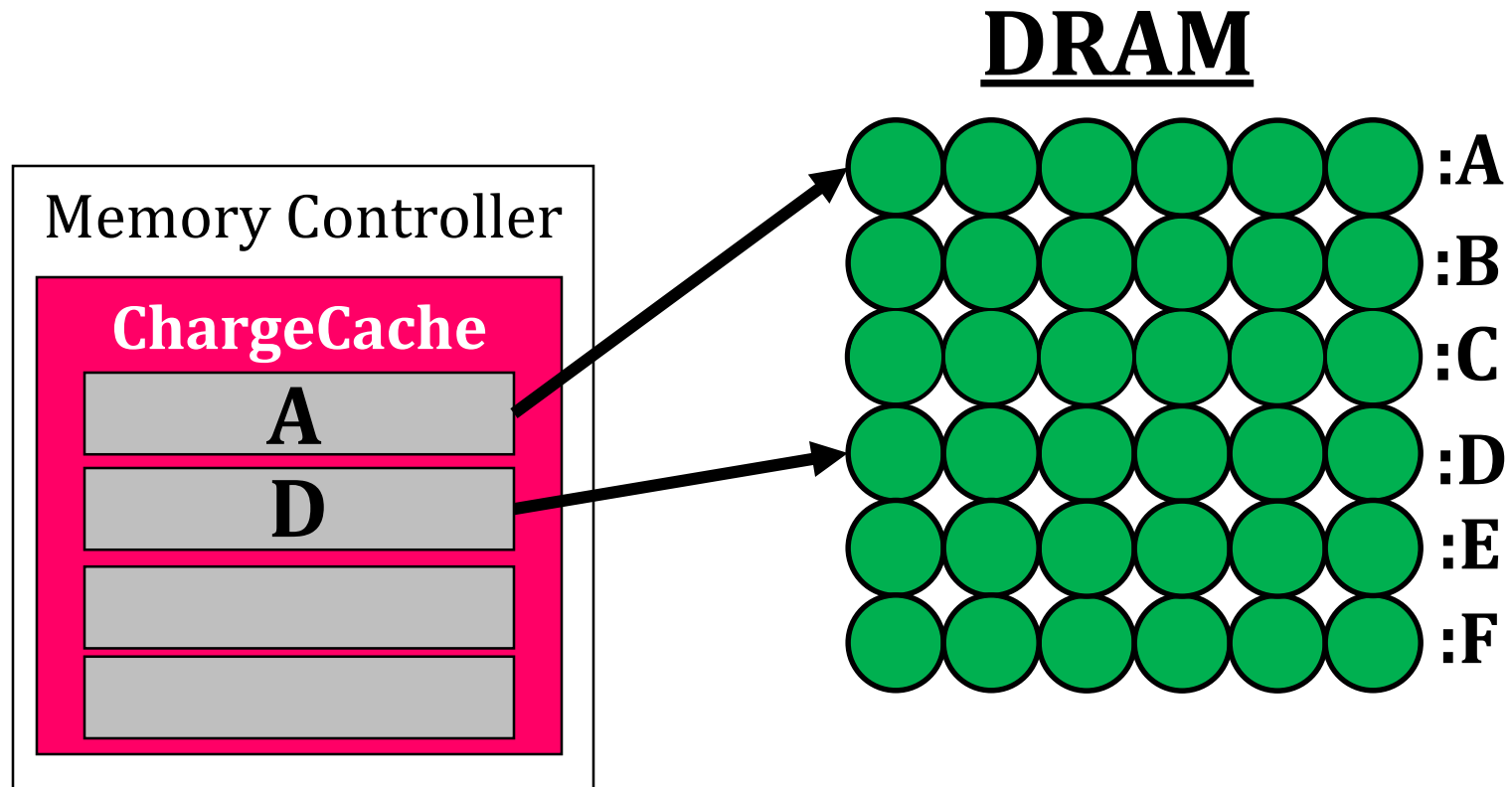
Area and Power Overhead

- **Modeled with CACTI**

- **Area**
  - ~5KB for 128-entry ChargeCache
  - 0.24% of a 4MB Last Level Cache (LLC) area

- **Power Consumption**
  - 0.15 mW on average (static + dynamic)
  - 0.23% of the 4MB LLC power consumption

SAFARI

Methodology

- **Simulator**
  - DRAM Simulator (Ramulator *[Kim+, CAL'15]*)
    https://github.com/CMU-SAFARI/ramulator

- **Workloads**
  - 22 single-core workloads
    - SPEC CPU2006, TPC, STREAM
  - 20 multi-programmed 8-core workloads
    - By randomly choosing from single-core workloads
  - Execute at least 1 billion representative instructions per core (Pinpoints)

- **System Parameters**
  - 1/8 core system with 4MB LLC
  - Default tRCD/tRAS of 11/28 cycles

Single-core Performance

Legend:
- NUAT (gray)
- ChargeCache (pink)
- ChargeCache + NUAT (blue)
- LL-DRAM (Upper bound) (black)

Y-axis: **Speedup** — 0%, 4%, 8%, 12%, 16%

X-axis labels: milc, ves, e20, plex, ie3d, ADM, lbm, tpp, DTD, mcf, t.22
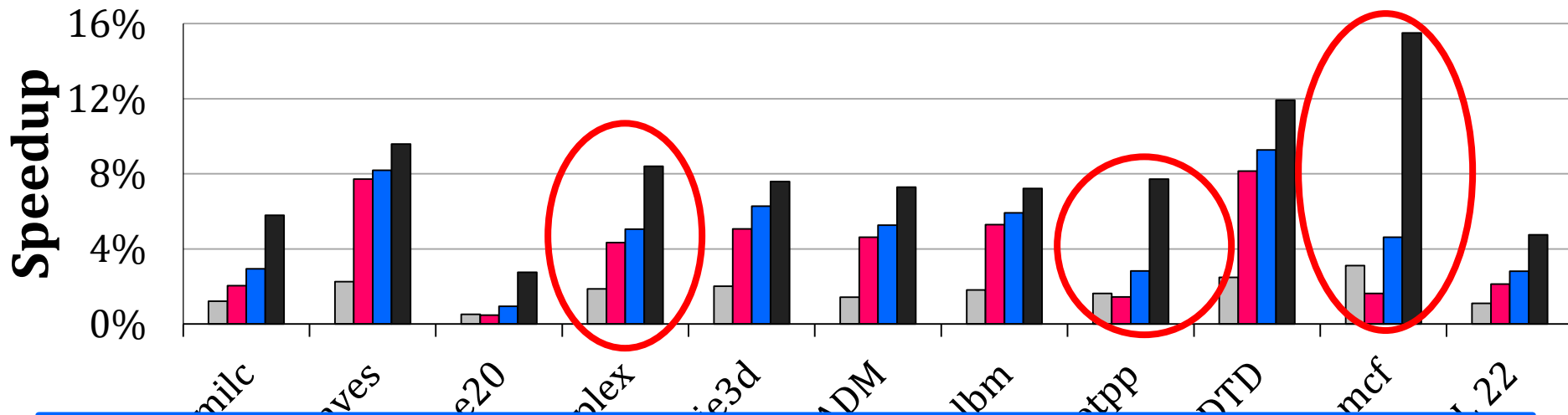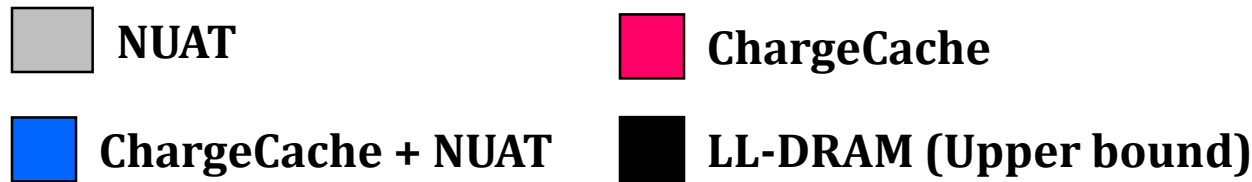
ChargeCache improves single-core performance

Eight-core Performance
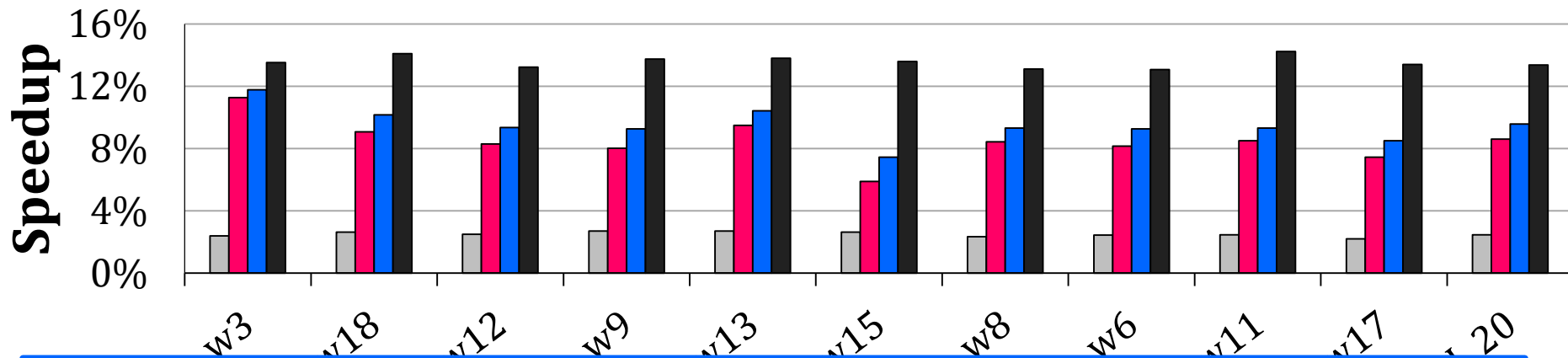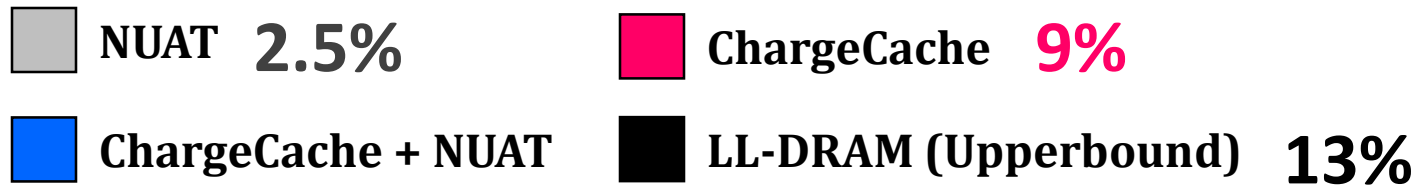
ChargeCache significantly improves multi-core performance

DRAM Energy Savings
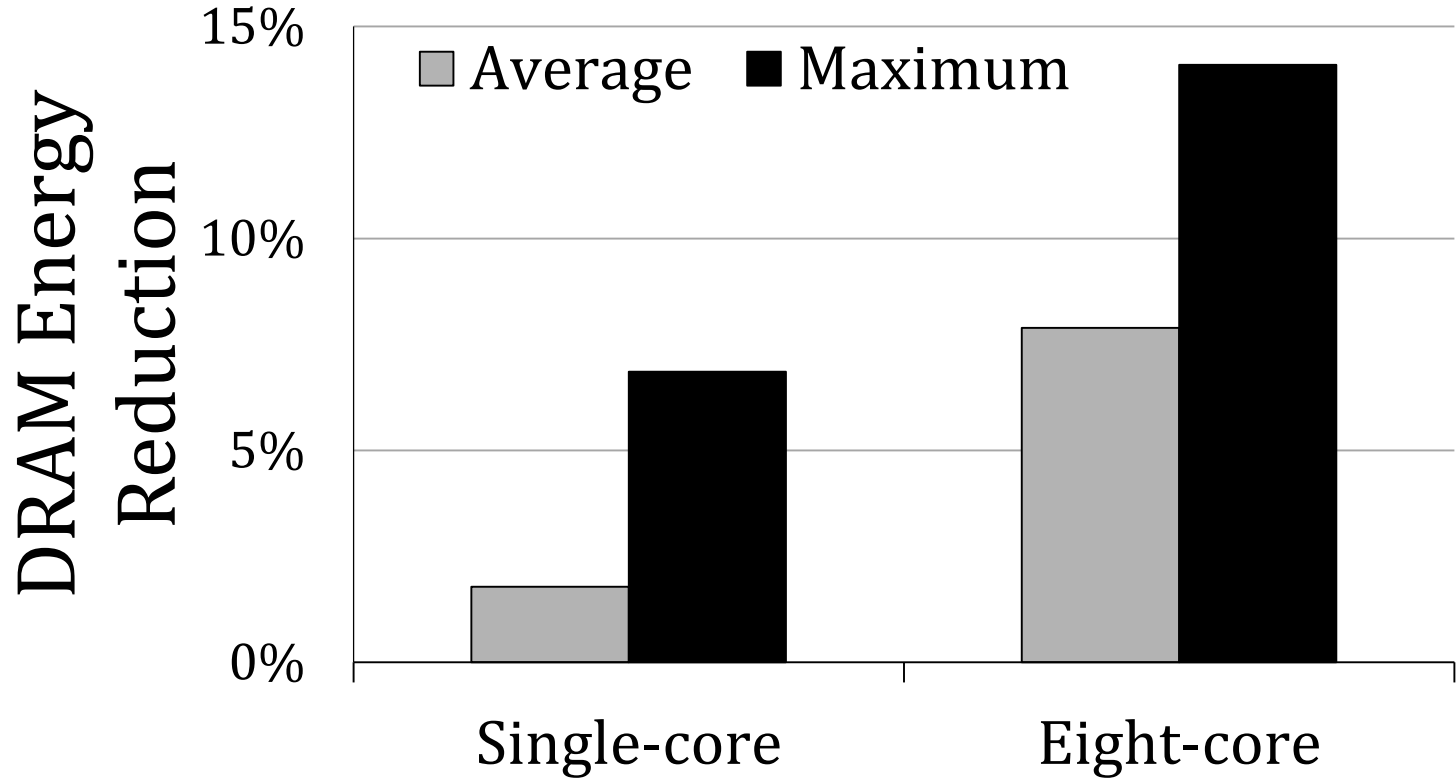
ChargeCache reduces DRAM energy

More on ChargeCache

- Hasan Hassan, Gennady Pekhimenko, Nandita Vijaykumar, Vivek Seshadri, Donghyuk Lee, Oguz Ergin, and Onur Mutlu,
"ChargeCache: Reducing DRAM Latency by Exploiting Row Access Locality"
Proceedings of the 22nd International Symposium on High-Performance Computer Architecture (**HPCA**), Barcelona, Spain, March 2016.
[Slides (pptx) (pdf)]
[Source Code]

ChargeCache: Reducing DRAM Latency by Exploiting Row Access Locality

Hasan Hassan[†*], Gennady Pekhimenko[†], Nandita Vijaykumar[†]
Vivek Seshadri[†], Donghyuk Lee[†], Oguz Ergin[*], Onur Mutlu[†]

[†] *Carnegie Mellon University*   [*] *TOBB University of Economics & Technology*

Summary: Low-Latency Memory

Summary: Tackling Long Memory Latency

- **Reason 1: Design of DRAM Micro-architecture**
  - Goal: Maximize capacity/area, not minimize latency

- **Reason 2: "One size fits all" approach to latency specification**
  - Same latency parameters for all temperatures
  - Same latency parameters for all DRAM chips (e.g., rows)
  - Same latency parameters for all parts of a DRAM chip
  - Same latency parameters for all supply voltage levels
  - Same latency parameters for all application data
  - …

SAFARI

Fundamentally Low Latency Computing Architectures

On DRAM Power Consumption

VAMPIRE DRAM Power Model

- Saugata Ghose, A. Giray Yaglikci, Raghav Gupta, Donghyuk Lee, Kais Kudrolli, William X. Liu, Hasan Hassan, Kevin K. Chang, Niladrish Chatterjee, Aditya Agrawal, Mike O'Connor, and Onur Mutlu,
"What Your DRAM Power Models Are Not Telling You: Lessons from a Detailed Experimental Study"
*Proceedings of the [ACM International Conference on Measurement and Modeling of Computer Systems](#) (**SIGMETRICS**)*, Irvine, CA, USA, June 2018.
[[Abstract](#)]

What Your DRAM Power Models Are Not Telling You: Lessons from a Detailed Experimental Study

Saugata Ghose[†]   Abdullah Giray Yağlıkçı[‡†]   Raghav Gupta[†]   Donghyuk Lee[§]
Kais Kudrolli[†]   William X. Liu[†]   Hasan Hassan[‡]   Kevin K. Chang[†]
Niladrish Chatterjee[§]   Aditya Agrawal[§]   Mike O'Connor[§¶]   Onur Mutlu[‡†]

[†]Carnegie Mellon University   [‡]ETH Zürich   [§]NVIDIA   [¶]University of Texas at Austin

SAFARI

Conclusion

Agenda

- Brief Introduction

- A Motivating Example

- Memory System Trends

- What Will You Learn In This Course
  - And, how to make the best of it…

- Memory Fundamentals

- Key Memory Challenges and Solution Directions
  - Security, Reliability, Safety
  - Energy and Performance: Data-Centric Systems
  - Latency and Latency-Reliability Tradeoffs

- Summary and Future Lookout

SAFARI

Four Key Directions

- Fundamentally Secure/Reliable/Safe Architectures

- Fundamentally Energy-Efficient Architectures
  - Memory-centric (Data-centric) Architectures

- Fundamentally Low-Latency Architectures

- Architectures for Genomics, Medicine, Health

SAFARI

What Have We Learned In This Course?

- **Memory Systems and Memory-Centric Computing Systems**
  - July 9-13, 2018

- Topic 1: Main Memory Trends and Basics
- Topic 2: Memory Reliability & Security: RowHammer and Beyond
- Topic 3: In-memory Computation
- Topic 4: Low-Latency (and Low-Energy) Memory
- Topic 5 (unlikely): Enabling and Exploiting Non-Volatile Memory
- Topic 6 (unlikely): Flash Memory and SSD Scaling

- Major Overview Reading:
  - Mutlu and Subramaniam, "Research Problems and Opportunities in Memory Systems," SUPERFRI 2014.

Some Solution Principles (So Far)

- **More data-centric system design**
  - Do not center everything around computation units

- **Better cooperation across layers of the system**
  - Careful co-design of components and layers: system/arch/device
  - Better, richer, more expressive and flexible interfaces

- **Better-than-worst-case design**
  - Do not optimize for the worst case
  - Worst case should not determine the common case

- **Heterogeneity in design (specialization, asymmetry)**
  - Enables a more efficient design (No one size fits all)

It Is Time to …

- … design **principled system architectures** to solve the **memory problem**

- … design complete systems to be balanced, high-performance, and energy-efficient, i.e., data-centric (or memory-centric)

- … **make memory a key priority** in system design and optimize it & integrate it better into the system

- This can
  - Lead to **orders-of-magnitude** improvements
  - **Enable new applications & computing platforms**
  - **Enable better understanding of nature**
  - **…**

We Need to Revisit the Entire Stack

Problem
Algorithm
Program/Language
System Software
SW/HW Interface
Micro-architecture
Logic
Devices
Electrons
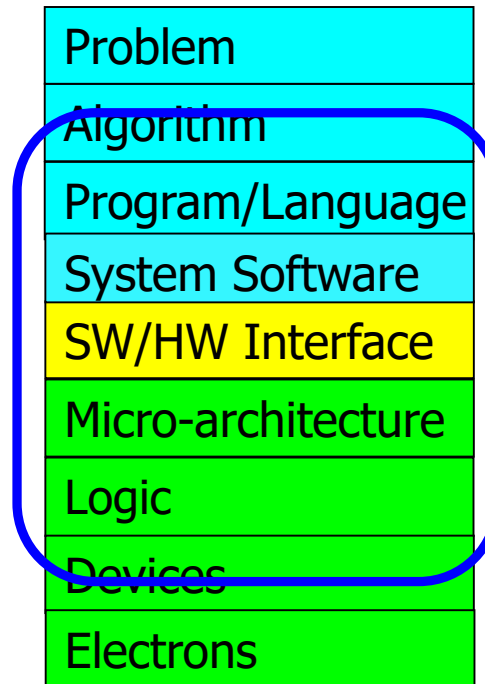
Course Materials and Beyond

- Website for Course Slides and Papers
  - https://people.inf.ethz.ch/omutlu/acaces2018.html
  - https://people.inf.ethz.ch/omutlu/projects.htm
  - Final lecture notes and readings (for all topics)

SAFARI

You Can Contact Me Any Time

- **My Contact Information**
  - Onur Mutlu
  - omutlu@gmail.com
  - https://people.inf.ethz.ch/omutlu/index.html
  - +41-79-572-1444 (my cell phone)
  - You can contact me any time with questions and ideas.

SAFARI

Thank You!

Keep in Touch!

Memory Systems
and Memory-Centric Computing Systems
Lecture 5, Topic 4: Low-Latency Memory

Prof. Onur Mutlu

omutlu@gmail.com

https://people.inf.ethz.ch/omutlu

13 July 2018

HiPEAC ACACES Summer School 2018

Systems@ETH *zürich*

SAFARI

ETH *zürich*

Carnegie Mellon

Readings, Videos, Reference Materials

Reference Overview Paper I

Enabling the Adoption of Processing-in-Memory: Challenges, Mechanisms, Future Research Directions

SAUGATA GHOSE, KEVIN HSIEH, AMIRALI BOROUMAND,
RACHATA AUSAVARUNGNIRUN

Carnegie Mellon University

ONUR MUTLU

ETH Zürich and Carnegie Mellon University

Saugata Ghose, Kevin Hsieh, Amirali Boroumand, Rachata Ausavarungnirun, Onur Mutlu,
"Enabling the Adoption of Processing-in-Memory: Challenges, Mechanisms, Future Research Directions"
Invited Book Chapter, to appear in 2018.
[Preliminary arxiv.org version]

Reference Overview Paper II

- Onur Mutlu and Lavanya Subramanian,
  "Research Problems and Opportunities in Memory Systems"
  Invited Article in Supercomputing Frontiers and Innovations (**SUPERFRI**)*, 2014/2015.*

Research Problems and Opportunities in Memory Systems

Onur Mutlu[1]*, Lavanya Subramanian*[1]

Reference Overview Paper III

- Onur Mutlu,
  "The RowHammer Problem and Other Issues We May Face as Memory Becomes Denser"
  Invited Paper in Proceedings of the Design, Automation, and Test in Europe Conference (**DATE**), Lausanne, Switzerland, March 2017.
  [Slides (pptx) (pdf)]

The RowHammer Problem
and Other Issues We May Face as Memory Becomes Denser

Onur Mutlu
ETH Zürich
onur.mutlu@inf.ethz.ch
https://people.inf.ethz.ch/omutlu

https://people.inf.ethz.ch/omutlu/pub/rowhammer-and-other-memory-issues_date17.pdf

Reference Overview Paper IV

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

Memory Scaling: A Systems Architecture Perspective

Onur Mutlu
Carnegie Mellon University
onur@cmu.edu
http://users.ece.cmu.edu/~omutlu/

https://people.inf.ethz.ch/omutlu/pub/memory-scaling_memcon13.pdf

Reference Overview Paper V

INVITED
PAPER

Error Characterization, Mitigation, and Recovery in Flash-Memory-Based Solid-State Drives

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

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

SAFARI

https://arxiv.org/pdf/1706.08642

Related Videos and Course Materials (I)

- **Undergraduate Computer Architecture Course Lecture Videos** (**2015**, **2014**, **2013**)

- **Undergraduate Computer Architecture Course Materials** (**2015**, **2014**, **2013**)

- **Graduate Computer Architecture Course Lecture Videos** (**2017**, **2015**, **2013**)

- **Graduate Computer Architecture Course Materials** (**2017**, **2015**, **2013**)

- **Parallel Computer Architecture Course Materials (Lecture Videos)**

Related Videos and Course Materials (II)

- **Freshman Digital Circuits and Computer Architecture Course Lecture Videos** (**2018**, **2017**)

- **Freshman Digital Circuits and Computer Architecture Course Materials** (**2018**)

- **Memory Systems Short Course Materials** (**Lecture Video on Main Memory and DRAM Basics**)

Some Open Source Tools (I)

- Rowhammer – Program to Induce RowHammer Errors
  - https://github.com/CMU-SAFARI/rowhammer
- Ramulator – Fast and Extensible DRAM Simulator
  - https://github.com/CMU-SAFARI/ramulator
- MemSim – Simple Memory Simulator
  - https://github.com/CMU-SAFARI/memsim
- NOCulator – Flexible Network-on-Chip Simulator
  - https://github.com/CMU-SAFARI/NOCulator
- SoftMC – FPGA-Based DRAM Testing Infrastructure
  - https://github.com/CMU-SAFARI/SoftMC

- Other open-source software from my group
  - **https://github.com/CMU-SAFARI/**
  - **http://www.ece.cmu.edu/~safari/tools.html**

SAFARI

Some Open Source Tools (II)

- MQSim – A Fast Modern SSD Simulator
  - https://github.com/CMU-SAFARI/MQSim
- Mosaic – GPU Simulator Supporting Concurrent Applications
  - https://github.com/CMU-SAFARI/Mosaic
- IMPICA – Processing in 3D-Stacked Memory Simulator
  - https://github.com/CMU-SAFARI/IMPICA
- SMLA – Detailed 3D-Stacked Memory Simulator
  - https://github.com/CMU-SAFARI/SMLA
- HWASim – Simulator for Heterogeneous CPU-HWA Systems
  - https://github.com/CMU-SAFARI/HWASim

- Other open-source software from my group
  - **https://github.com/CMU-SAFARI/**
  - **http://www.ece.cmu.edu/~safari/tools.html**

SAFARI

More Open Source Tools (III)

- **A lot more open-source software from my group**
  - **https://github.com/CMU-SAFARI/**
  - **http://www.ece.cmu.edu/~safari/tools.html**

Referenced Papers

- All are available at

  https://people.inf.ethz.ch/omutlu/projects.htm

  http://scholar.google.com/citations?user=7XyGUGkAAAAJ&hl=en

  https://people.inf.ethz.ch/omutlu/acaces2018.html

Ramulator: A Fast and Extensible DRAM Simulator

[IEEE Comp Arch Letters'15]

Ramulator Motivation

- DRAM and Memory Controller landscape is changing
- Many new and upcoming standards
- Many new controller designs
- A fast and easy-to-extend simulator is very much needed

Segment	DRAM Standards & Architectures
Commodity	DDR3 (2007) [14]; DDR4 (2012) [18]
Low-Power	LPDDR3 (2012) [17]; LPDDR4 (2014) [20]
Graphics	GDDR5 (2009) [15]
Performance	eDRAM [28], [32]; RLDRAM3 (2011) [29]
3D-Stacked	WIO (2011) [16]; WIO2 (2014) [21]; MCDRAM (2015) [13]; HBM (2013) [19]; HMC1.0 (2013) [10]; HMC1.1 (2014) [11]
Academic	SBA/SSA (2010) [38]; Staged Reads (2012) [8]; RAIDR (2012) [27]; SALP (2012) [24]; TL-DRAM (2013) [26]; RowClone (2013) [37]; Half-DRAM (2014) [39]; Row-Buffer Decoupling (2014) [33]; SARP (2014) [6]; AL-DRAM (2015) [25]

Table 1. Landscape of DRAM-based memory

Ramulator

- Provides out-of-the box support for many DRAM standards:
  - DDR3/4, LPDDR3/4, GDDR5, WIO1/2, HBM, plus new proposals (SALP, AL-DRAM, TLDRAM, RowClone, and SARP)
- ~2.5X faster than fastest open-source simulator
- Modular and extensible to different standards

Simulator (clang -O3)	Cycles (10^6)		Runtime (sec.)		Req/sec (10^3)		Memory (MB)
	Random	Stream	Random	Stream	Random	Stream	
Ramulator	652	411	752	249	133	402	2.1
DRAMSim2	645	413	2,030	876	49	114	1.2
USIMM	661	409	1,880	750	53	133	4.5
DrSim	647	406	18,109	12,984	6	8	1.6
NVMain	666	413	6,881	5,023	15	20	4,230.0

Table 3. Comparison of five simulators using two traces

SAFARI

Case Study: Comparison of DRAM Standards

Standard	Rate (MT/s)	Timing (CL-RCD-RP)	Data-Bus (Width×Chan.)	Rank-per-Chan	BW (GB/s)
DDR3	1,600	11-11-11	64-bit × 1	1	11.9
DDR4	2,400	16-16-16	64-bit × 1	1	17.9
SALP†	1,600	11-11-11	64-bit × 1	1	11.9
LPDDR3	1,600	12-15-15	64-bit × 1	1	11.9
LPDDR4	2,400	22-22-22	32-bit × 2*	1	17.9
GDDR5 [12]	6,000	18-18-18	64-bit × 1	1	44.7
HBM	1,000	7-7-7	128-bit × 8*	1	119.2
WIO	266	7-7-7	128-bit × 4*	1	15.9
WIO2	1,066	9-10-10	128-bit × 8*	1	127.2
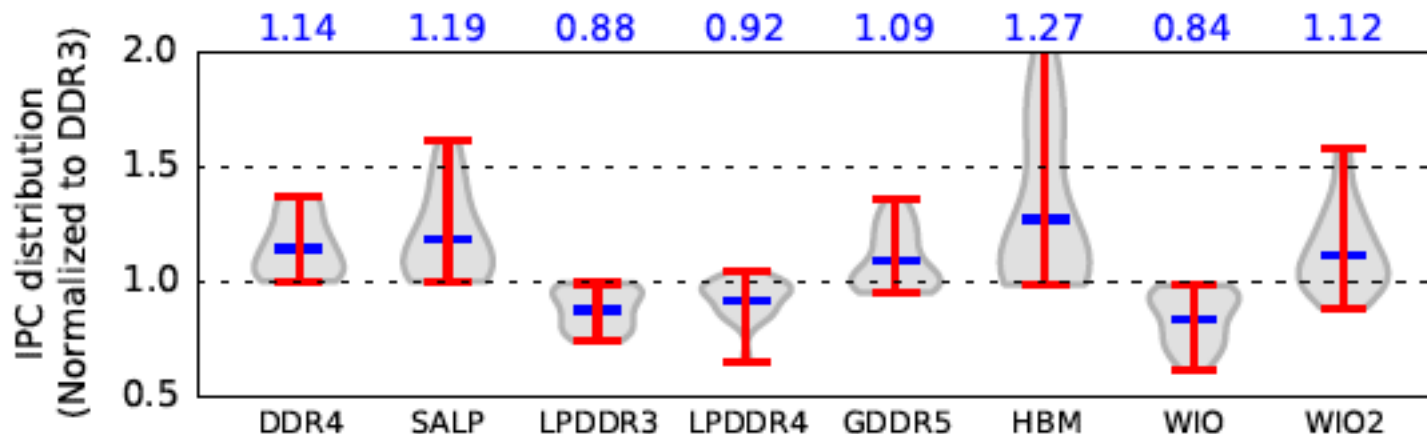
Across 22 workloads, simple CPU model

Figure 2. Performance comparison of DRAM standards

Ramulator Paper and Source Code

- Yoongu Kim, Weikun Yang, and Onur Mutlu,
  "Ramulator: A Fast and Extensible DRAM Simulator"
  IEEE Computer Architecture Letters (**CAL**), March 2015.
  [Source Code]

- Source code is released under the liberal MIT License
  - https://github.com/CMU-SAFARI/ramulator

Ramulator: A Fast and Extensible DRAM Simulator

Yoongu Kim[1]   Weikun Yang[1,2]   Onur Mutlu[1]
[1]Carnegie Mellon University   [2]Peking University

SAFARI