

Computer Architecture: Main Memory (Part III)

Prof. Onur Mutlu
Carnegie Mellon University

Main Memory Lectures

- These slides are from the **Scalable Memory Systems** course taught at ACACES 2013 (July 15-19, 2013)
- Course Website:
 - <http://users.ece.cmu.edu/~omutlu/acaces2013-memory.html>
- This is the third lecture:
 - Lecture 3a (July 17, 2013): DRAM Basics and DRAM Scaling: New DRAM Architectures II ([pptx](#)) ([pdf](#))

Scalable Many-Core Memory Systems

Lecture 3, Topic 1: DRAM Basics and DRAM Scaling

Prof. Onur Mutlu

<http://www.ece.cmu.edu/~omutlu>

onur@cmu.edu

HiPEAC ACACES Summer School 2013

July 17, 2013

Carnegie Mellon

New DRAM Architectures

- RAIDR: Reducing Refresh Impact
- TL-DRAM: Reducing DRAM Latency
- SALP: Reducing Bank Conflict Impact
- RowClone: Fast Bulk Data Copy and Initialization

Subarray-Level Parallelism: Reducing Bank Conflict Impact

Yoongu Kim, Vivek Seshadri, Donghyuk Lee, Jamie Liu, and Onur Mutlu,
"A Case for Exploiting Subarray-Level Parallelism (SALP) in DRAM"
Proceedings of the 39th International Symposium on Computer Architecture (ISCA),
Portland, OR, June 2012. [Slides \(pptx\)](#)

The Memory Bank Conflict Problem

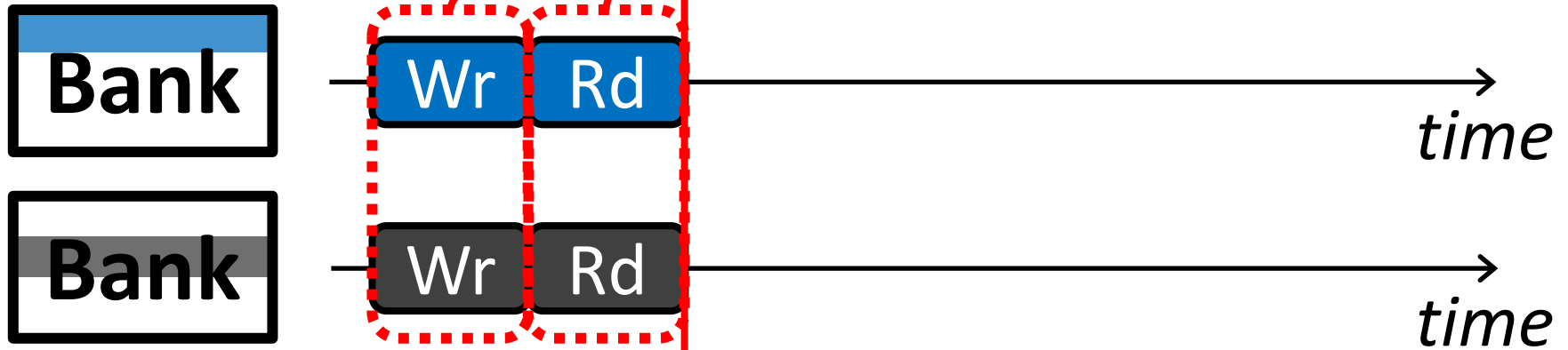
- Two requests to the same bank are serviced serially
- Problem: Costly in terms of performance and power
- Goal: We would like to reduce bank conflicts without increasing the number of banks (at low cost)

- Idea: Exploit the internal sub-array structure of a DRAM bank to parallelize bank conflicts
 - By reducing global sharing of hardware between sub-arrays

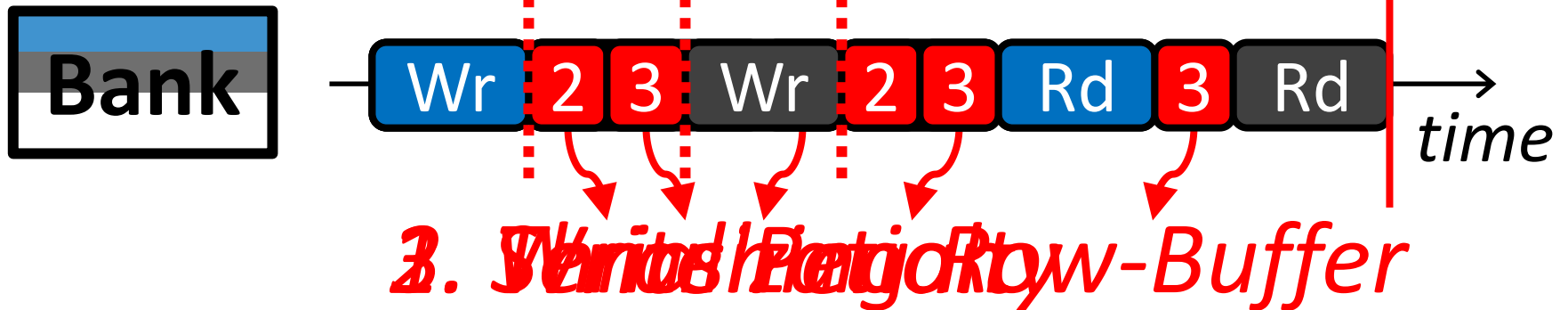
- Kim, Seshadri, Lee, Liu, Mutlu, "A Case for Exploiting Subarray-Level Parallelism in DRAM," ISCA 2012.

The Problem with Memory Bank Conflicts

- **Two Banks**



- **One Bank**



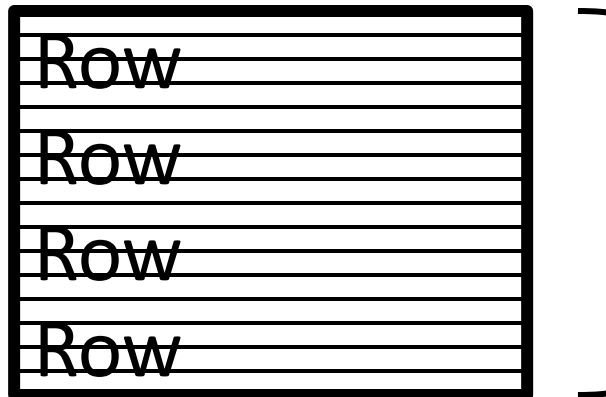
Goal

- **Goal:** *Mitigate the detrimental effects of bank conflicts in a cost-effective manner*
- **Naïve solution:** Add more banks
 - **Very expensive**
- **Cost-effective solution:** Approximate the benefits of more banks without adding more banks

Key Observation #1

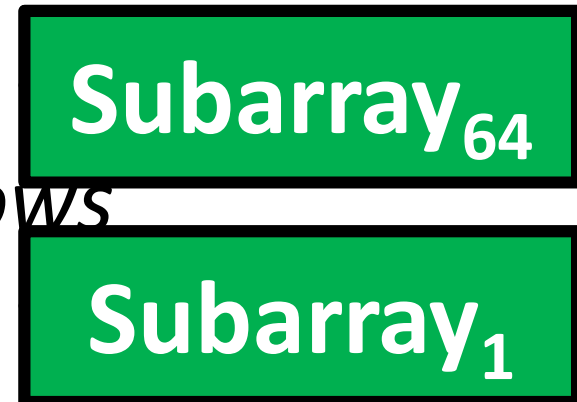
A DRAM bank is divided into *subarrays*

Logical Bank



Row-Buffer

Physical Bank



Global Row-Buf

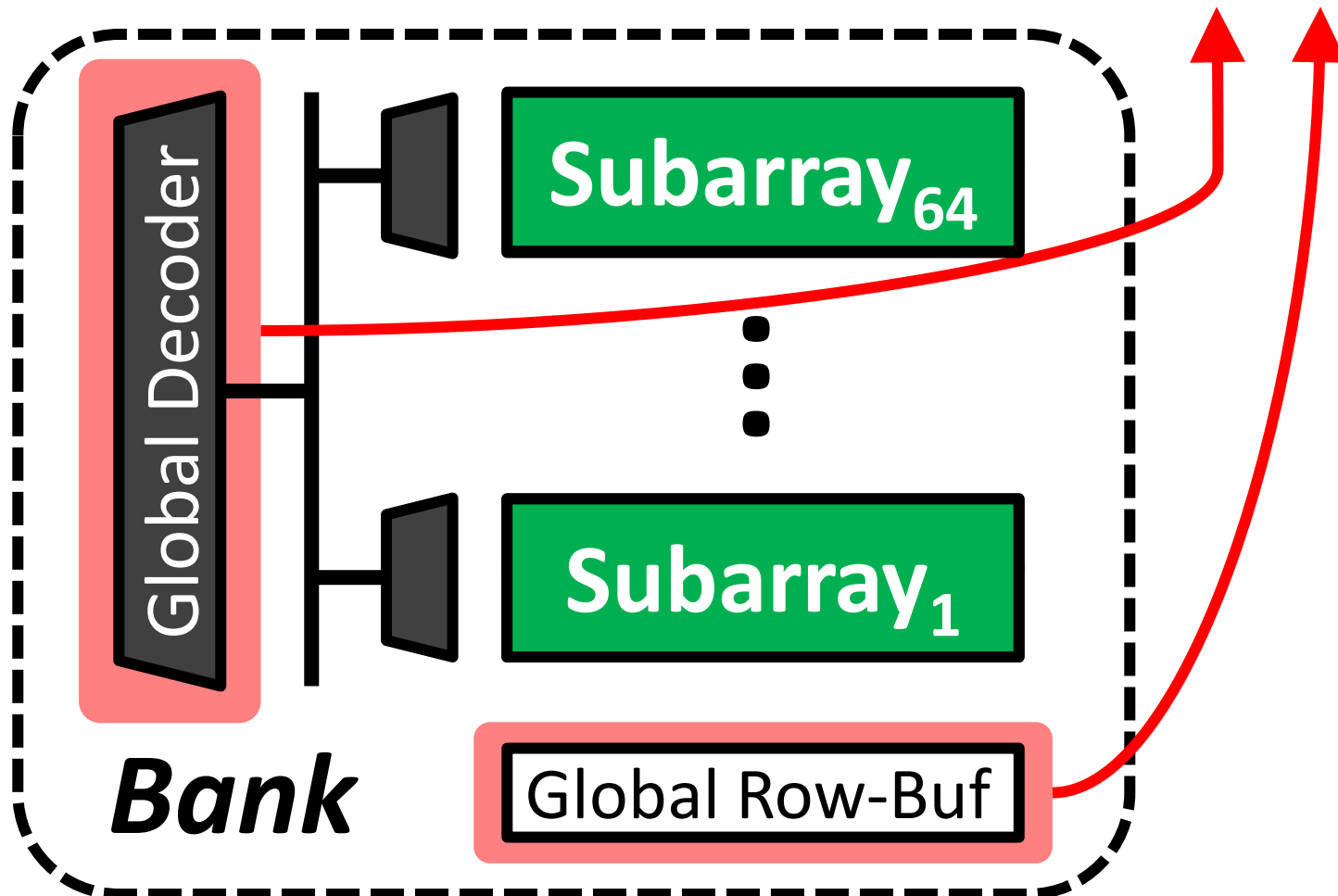
A single row-buffer **cannot** drive all rows

Many *local row-buffers*, one at each *subarray*

Key Observation #2

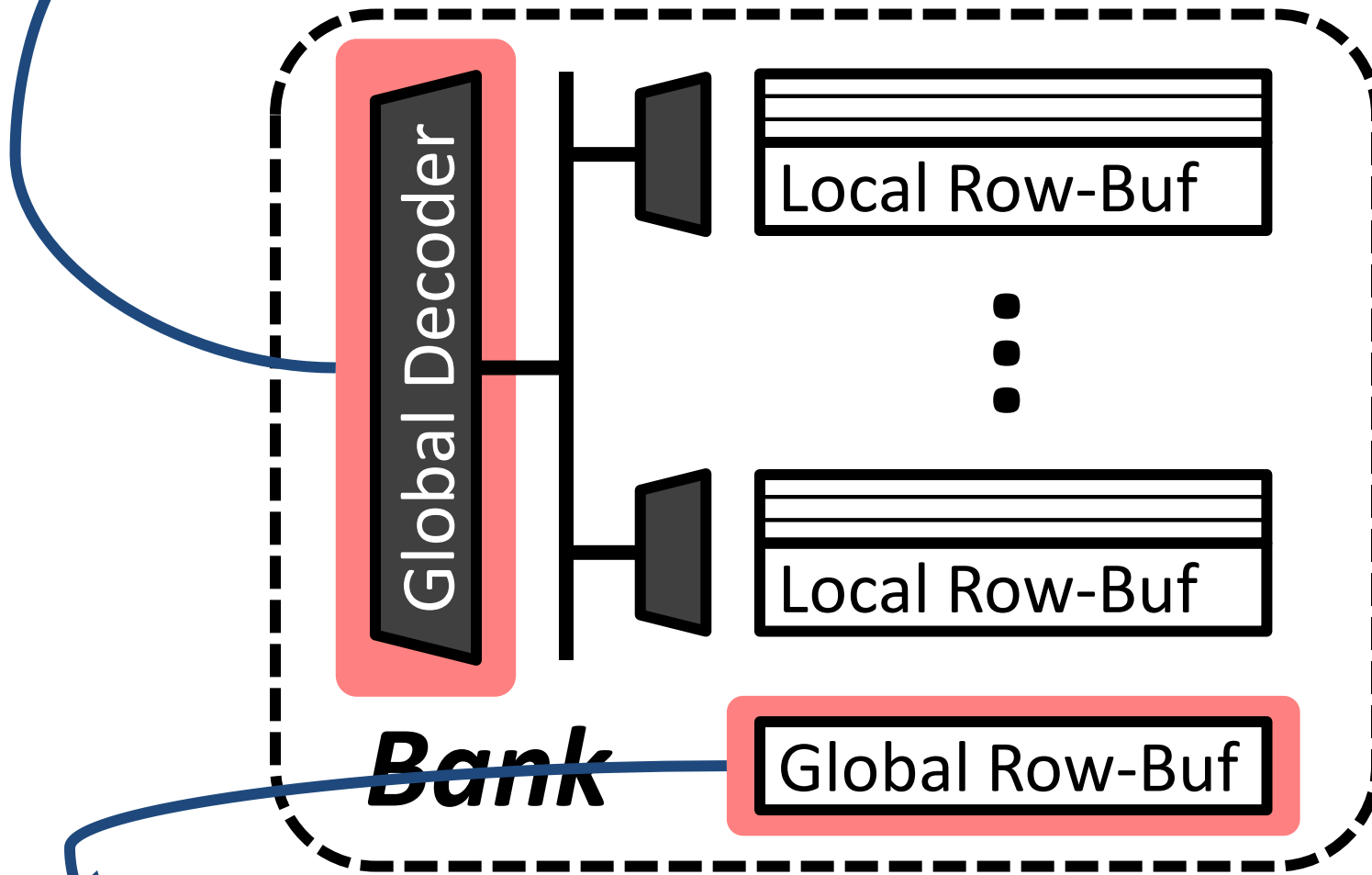
Each subarray is mostly independent...

- except occasionally sharing *global structures*



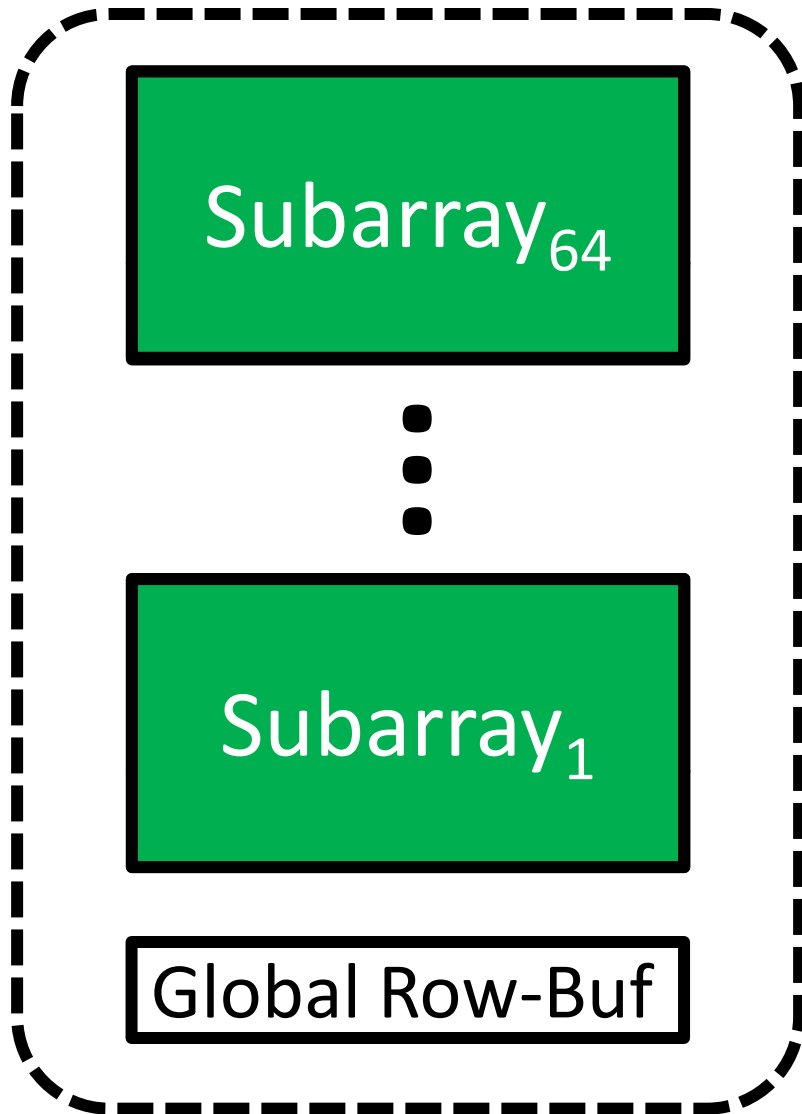
Key Idea: Reduce Sharing of Globals

1. Parallel access to subarrays



2. Utilize multiple local row-buffers

Overview of Our Mechanism



1. Parallelize

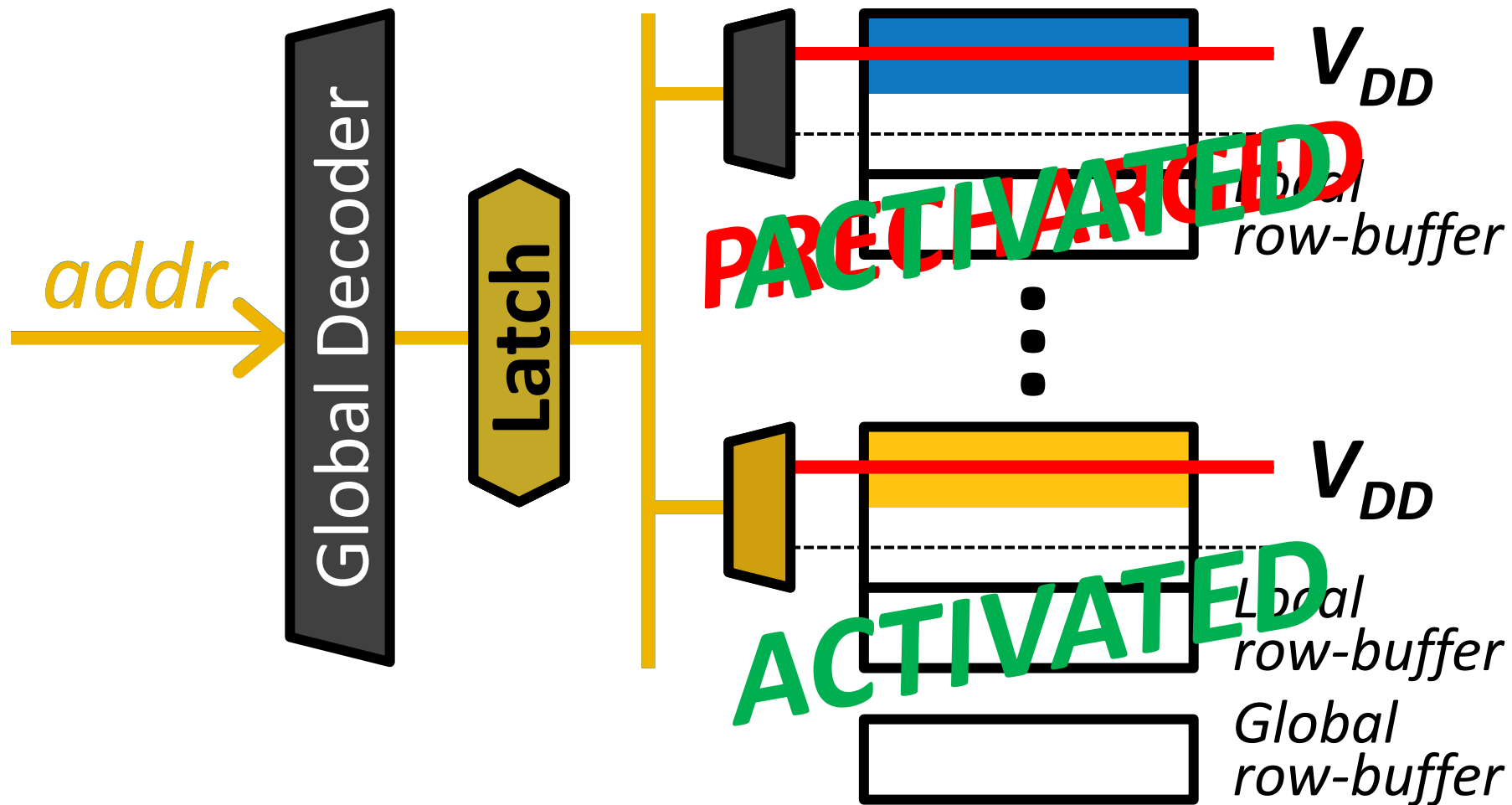
2. Utilize multiple **Req** **Req** **Req** **Req**
To same buffers
but diff. subarrays

Challenges: Global Structures

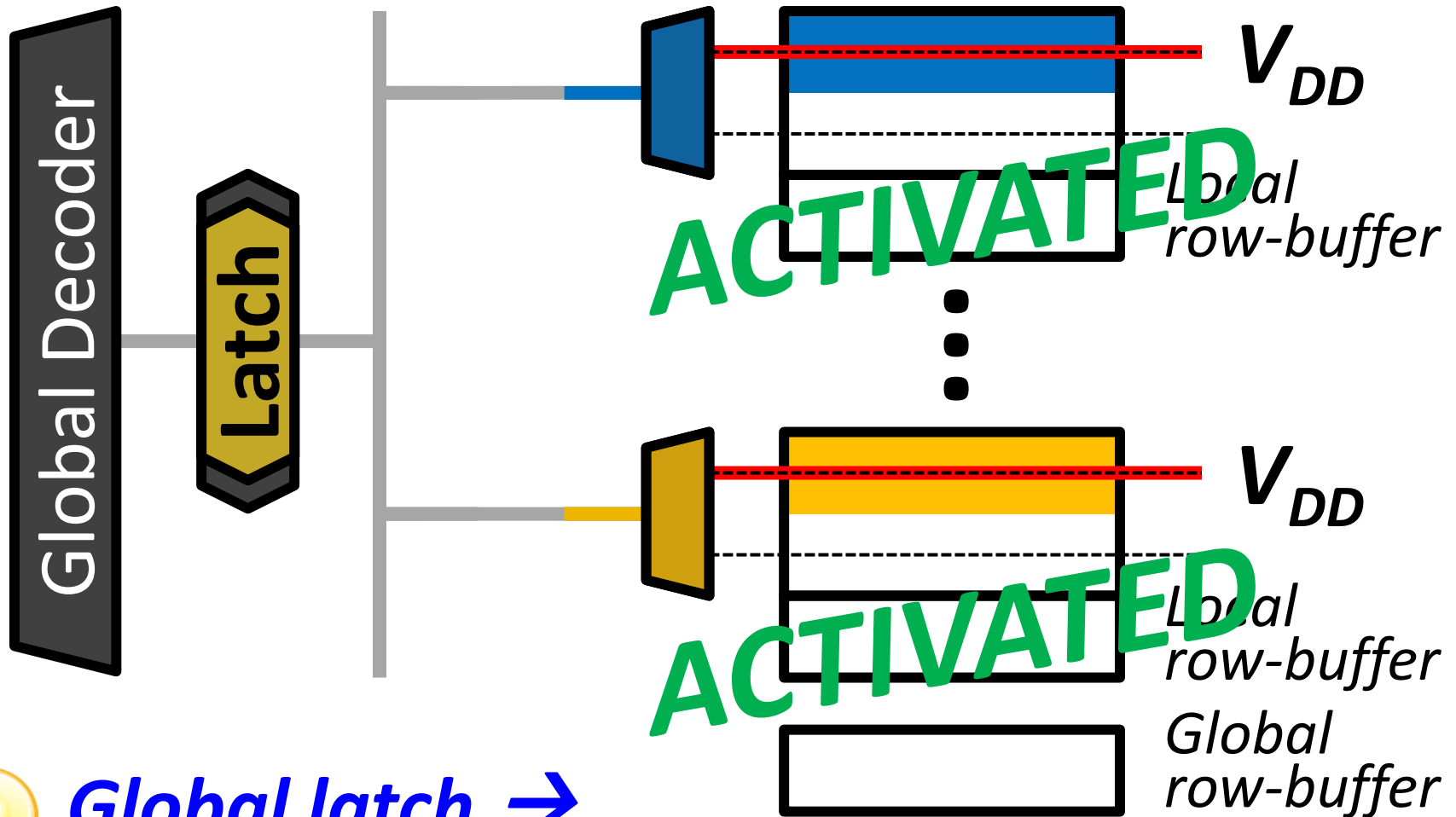
1. Global Address Latch

2. Global Bitlines

Challenge #1. Global Address Latch



Solution #1. Subarray Address Latch



Global latch →
local latches

Challenges: Global Structures

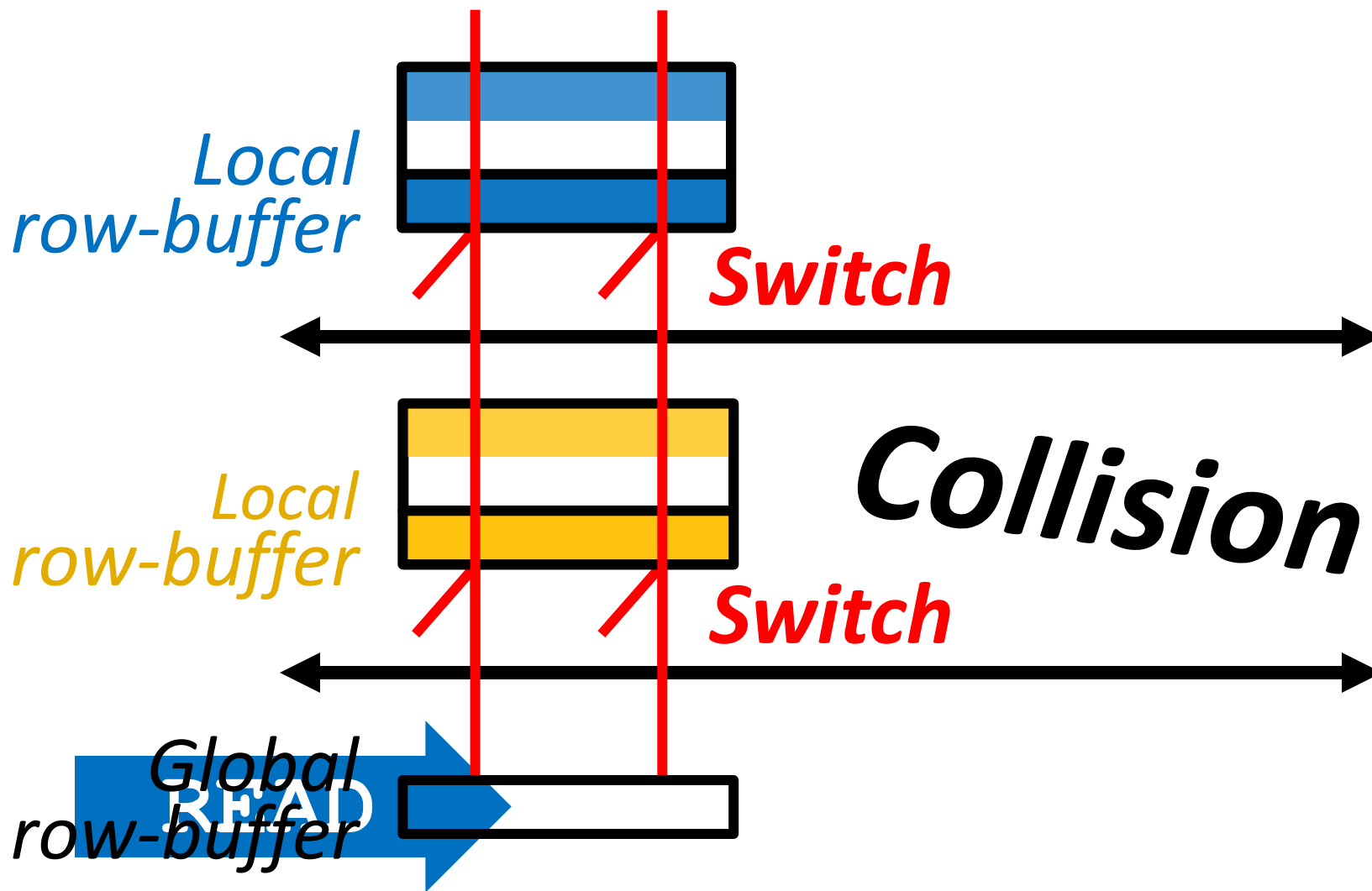
1. Global Address Latch

- Problem: Only one raised wordline
- Solution: **Subarray Address Latch**

2. Global Bitlines

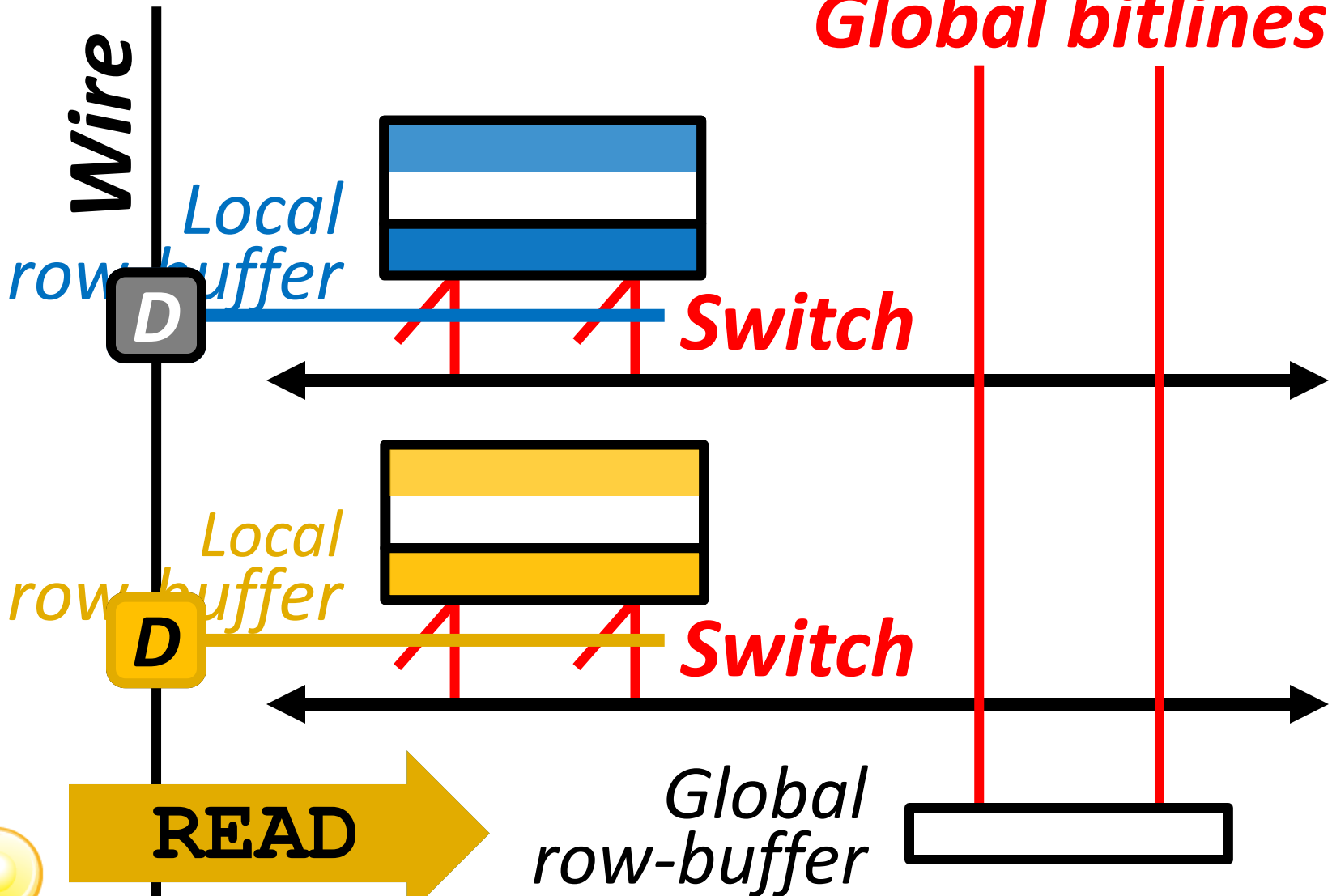
Challenge #2. Global Bitlines

Global bitlines



Solution #2. Designated-Bit Latch

Global bitlines



Selectively connect local to global



Challenges: Global Structures

1. Global Address Latch

- Problem: Only one raised wordline
- Solution: **Subarray Address Latch**

2. Global Bitlines

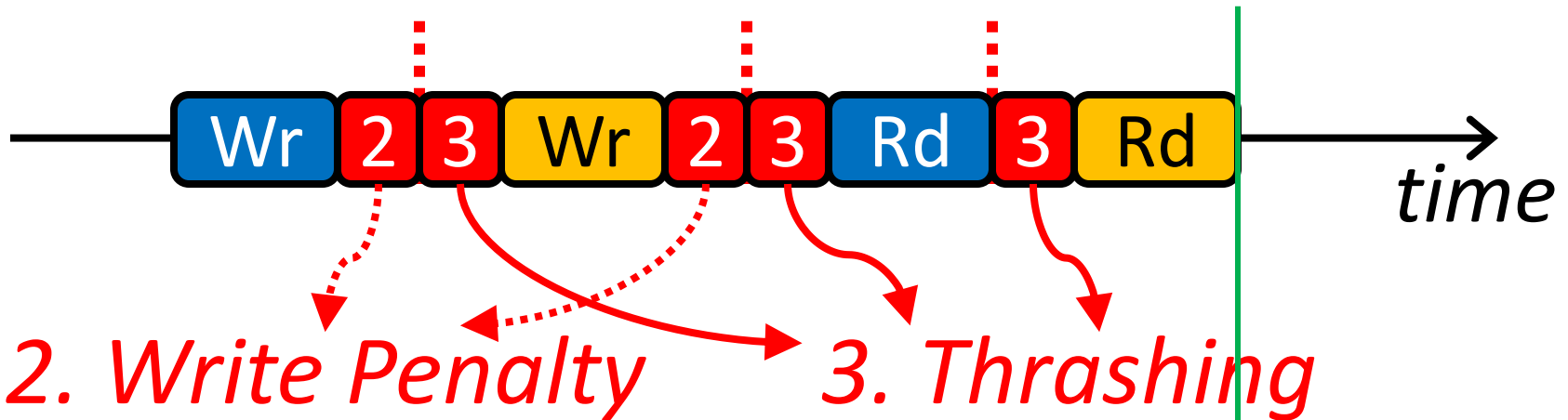
- Problem: Collision during access
- Solution: **Designated-Bit Latch**

MASA (Multitude of Activated Subarrays)

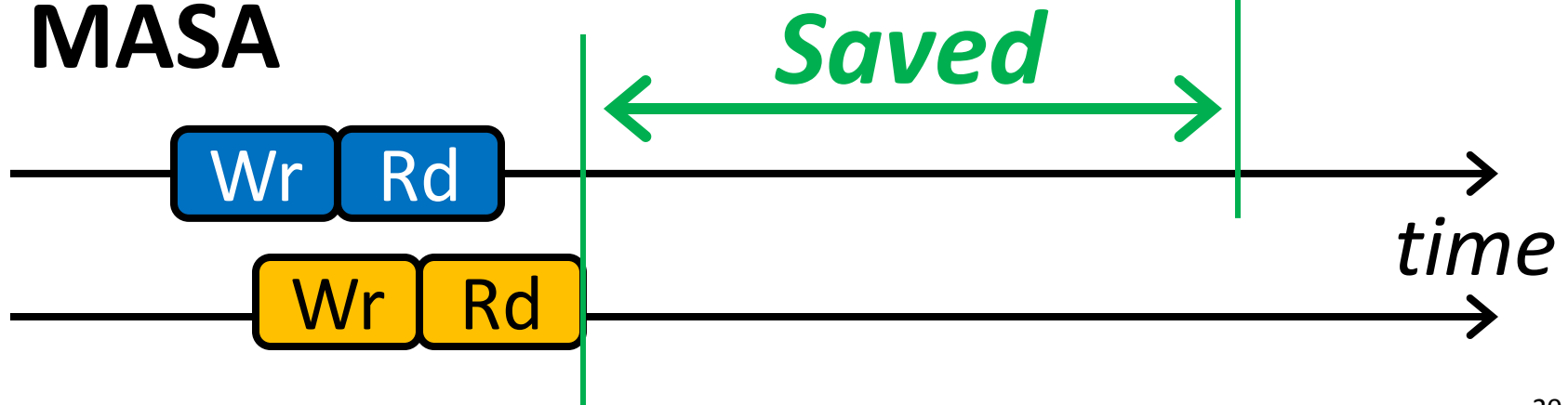
MASA: Advantages

- Baseline (Subarray-Oblivious)

1. Serialization



- MASA

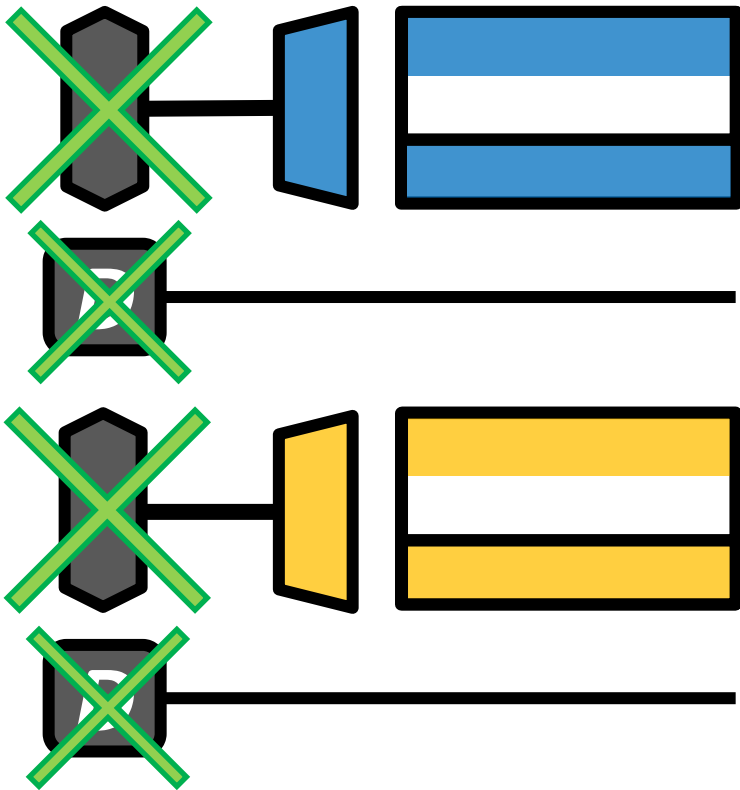


MASA: Overhead

- **DRAM Die Size: 0.15%** increase
 - Subarray Address Latches
 - Designated-Bit Latches & Wire
- **DRAM Static Energy:** Small increase
 - **0.56mW** for each activated subarray
 - *But saves dynamic energy*
- **Controller:** Small additional storage
 - Keep track of subarray status (< **256B**)
 - Keep track of new timing constraints

Cheaper Mechanisms

Latches



1. Serialization

2. Wr-Penalty

3. Thrashing

MASA

SALP-2

SALP-1

System Configuration

- **System Configuration**

- CPU: 5.3GHz, 128 ROB, 8 MSHR
- LLC: 512kB per-core slice

- **Memory Configuration**

- DDR3-1066
- **(default) 1 channel, 1 rank, 8 banks, 8 subarrays-per-bank**
- *(sensitivity)* 1-8 chans, 1-8 ranks, 8-64 banks, 1-128 subarrays

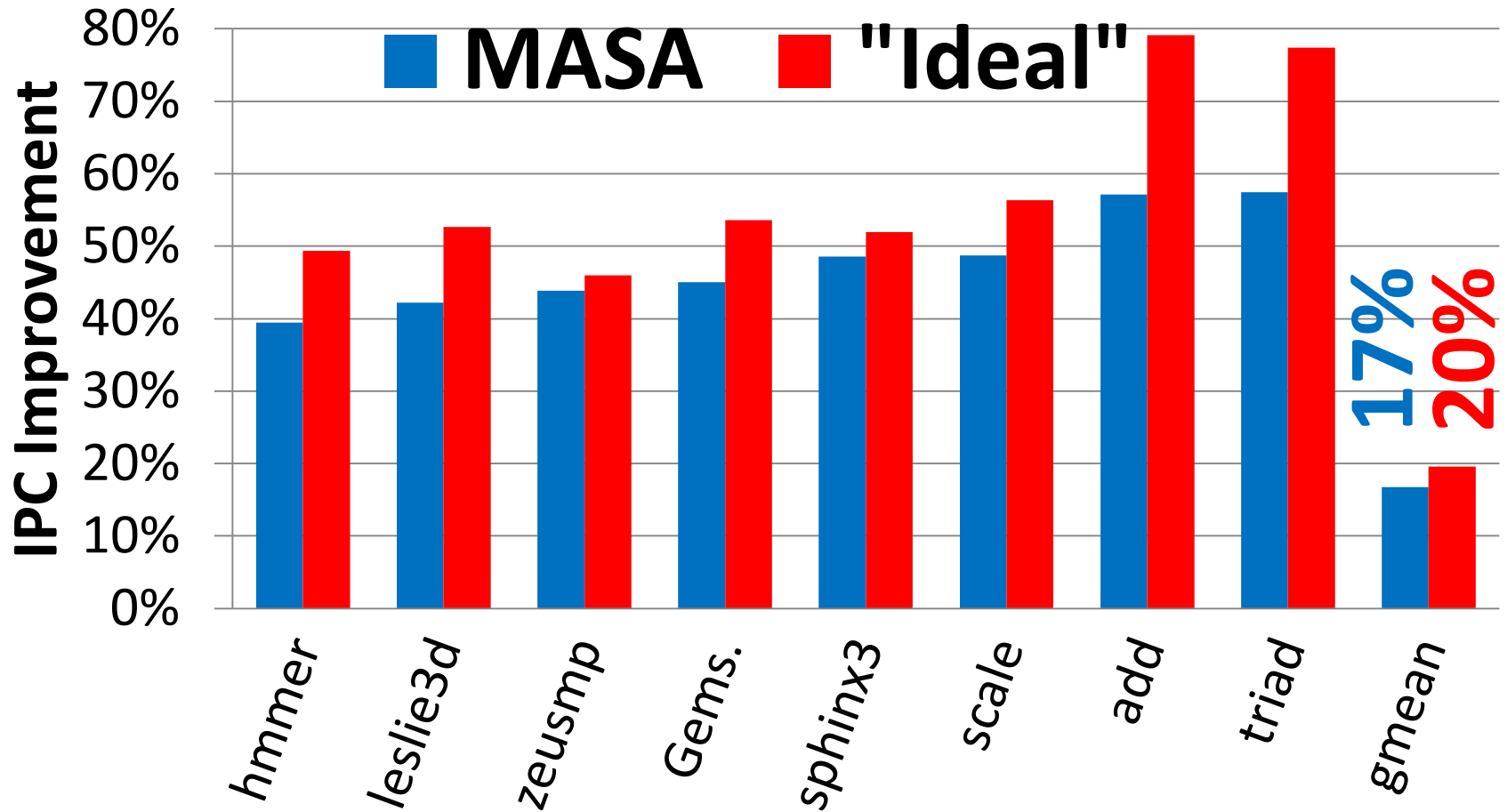
- **Mapping & Row-Policy**

- **(default) Line-interleaved & Closed-row**
- *(sensitivity)* Row-interleaved & Open-row

- **DRAM Controller Configuration**

- 64-/64-entry read/write queues per-channel
- FR-FCFS, batch scheduling for writes

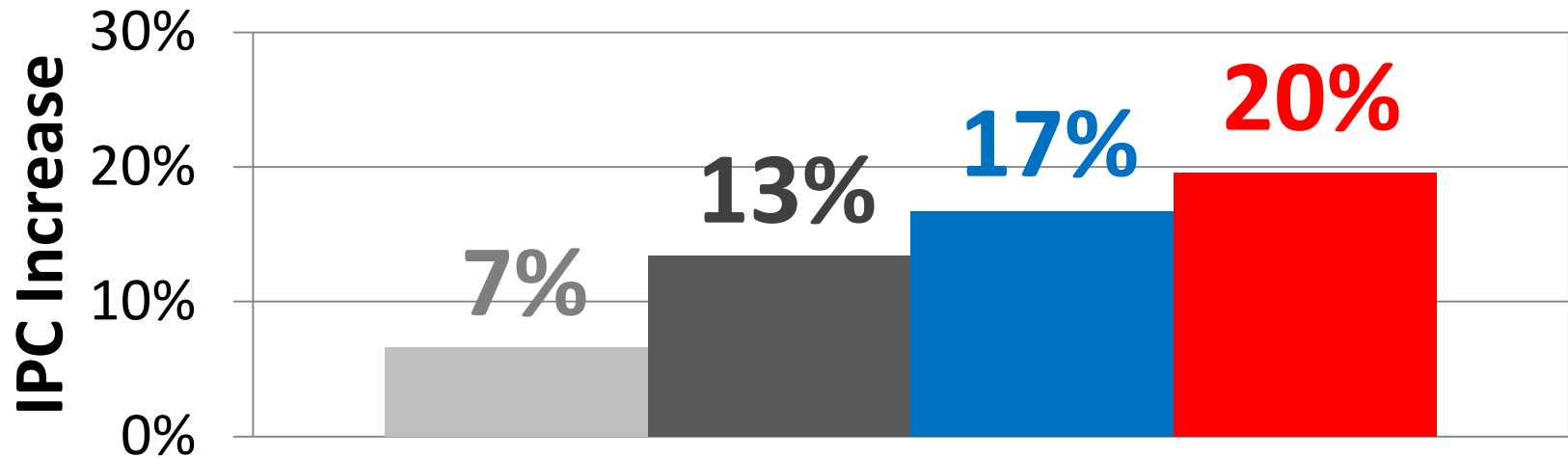
SALP: Single-core Results



***MASA** achieves most of the benefit of having more banks ("Ideal")*

SALP: Single-Core Results

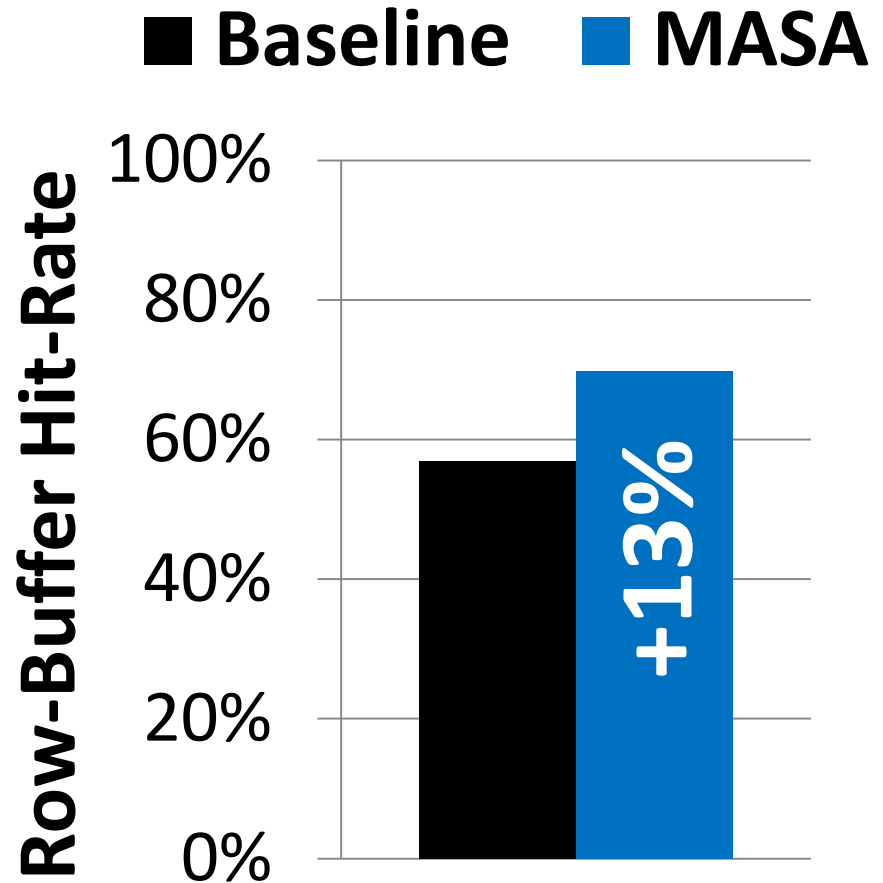
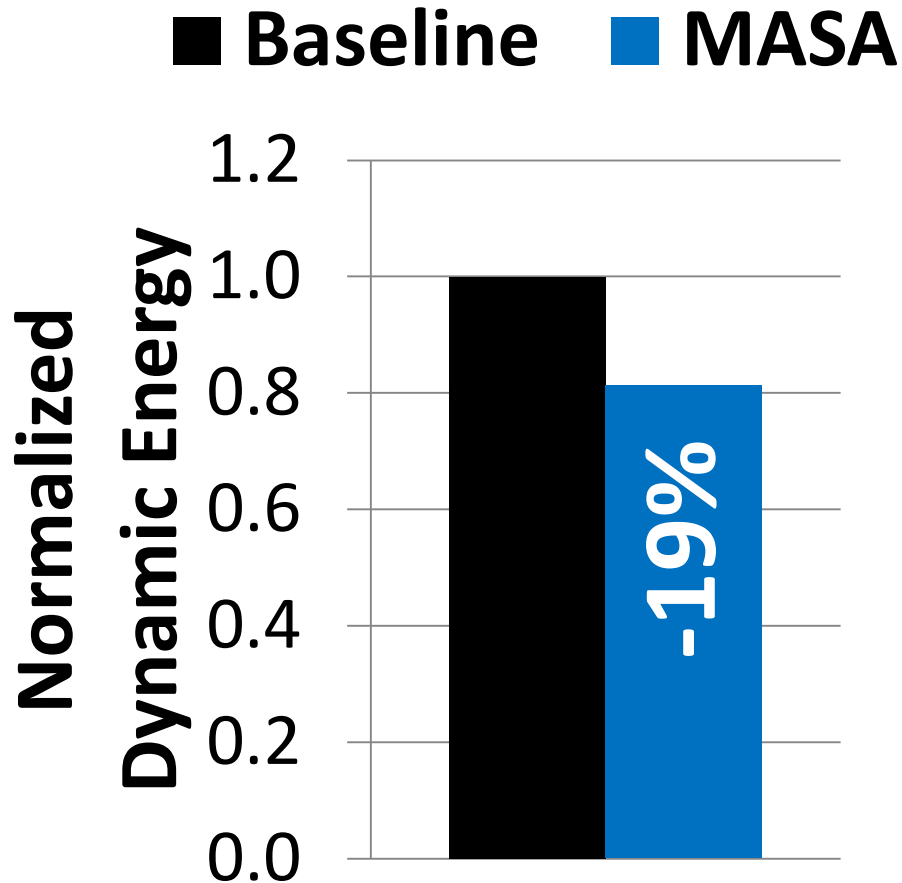
■ SALP-1 ■ SALP-2 ■ MASA ■ "Ideal"



DRAM Die Area	< 0.15%	0.15%	36.3%
----------------------	-------------------	--------------	--------------

SALP-1, SALP-2, MASA improve performance at low cost

Subarray-Level Parallelism: Results



MASA increases energy-efficiency

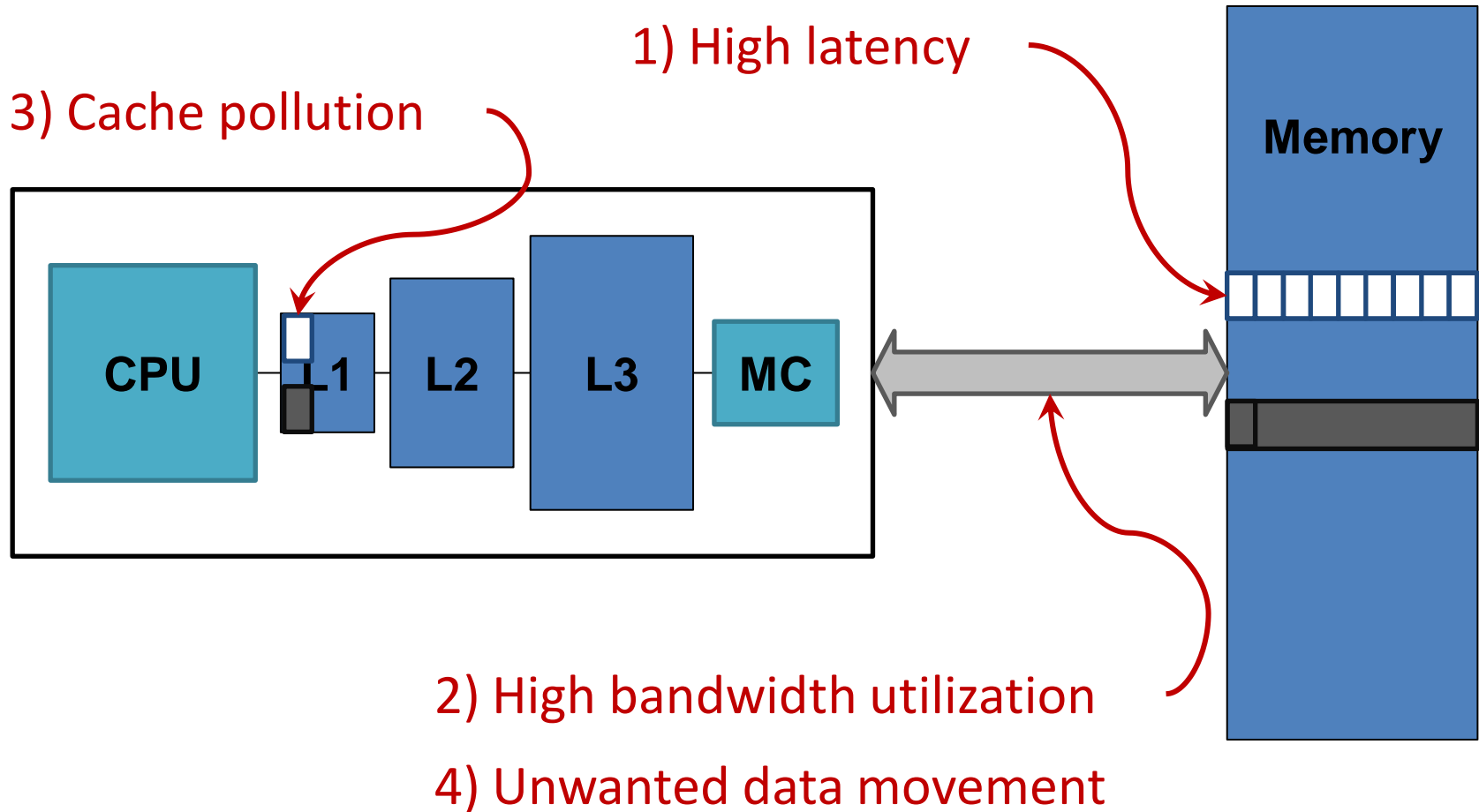
New DRAM Architectures

- RAIDR: Reducing Refresh Impact
- TL-DRAM: Reducing DRAM Latency
- SALP: Reducing Bank Conflict Impact
- RowClone: Fast Bulk Data Copy and Initialization

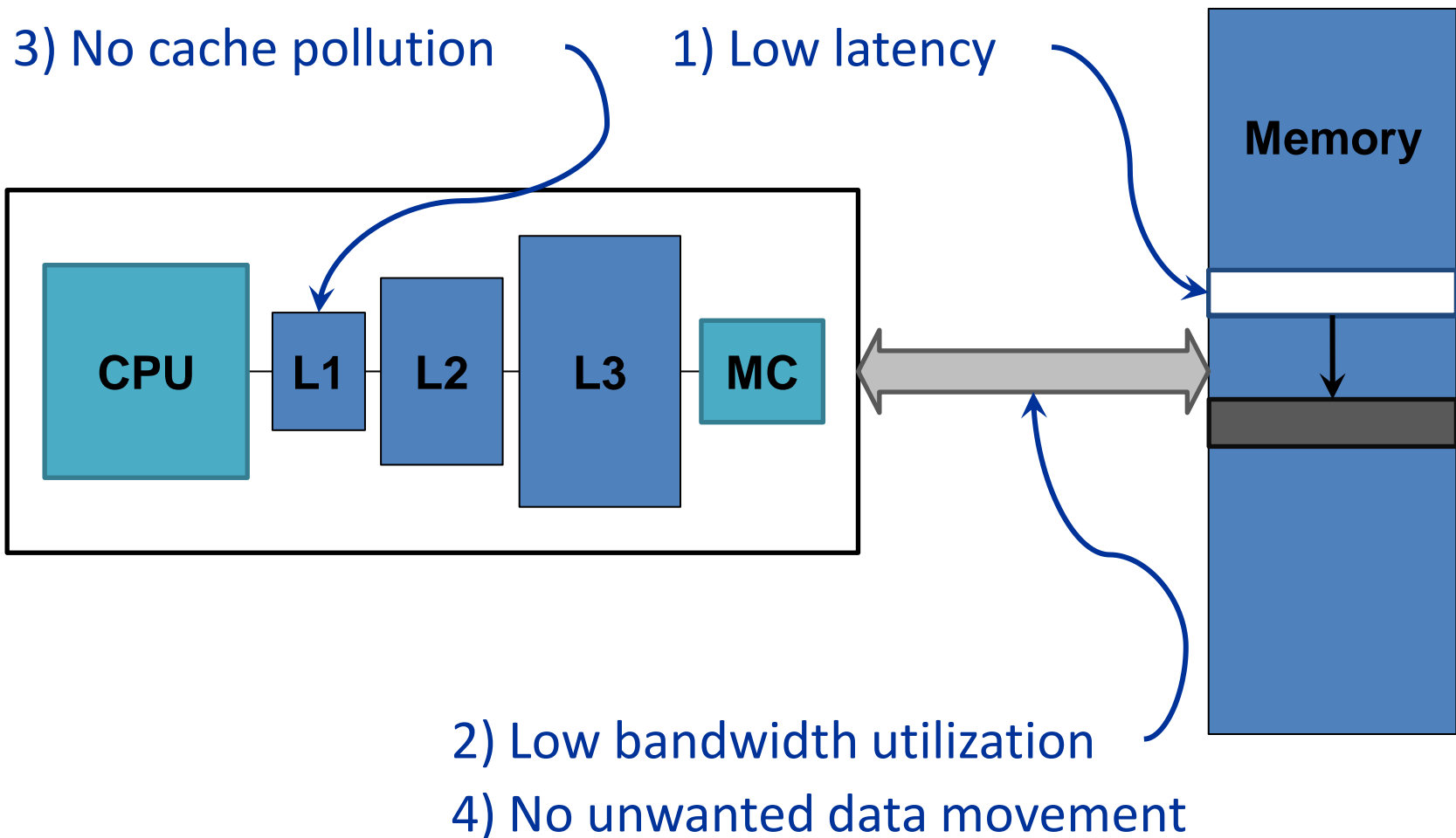
RowClone: Fast Bulk Data Copy and Initialization

Vivek Seshadri, Yoongu Kim, Chris Fallin, Donghyuk Lee, Rachata Ausavarungnirun,
Gennady Pekhimenko, Yixin Luo, Onur Mutlu, Phillip B. Gibbons, Michael A. Kozuch, Todd C. Mowry,
"RowClone: Fast and Efficient In-DRAM Copy and Initialization of Bulk Data"
CMU Computer Science Technical Report, CMU-CS-13-108, Carnegie Mellon University, April 2013.

Today's Memory: Bulk Data Copy

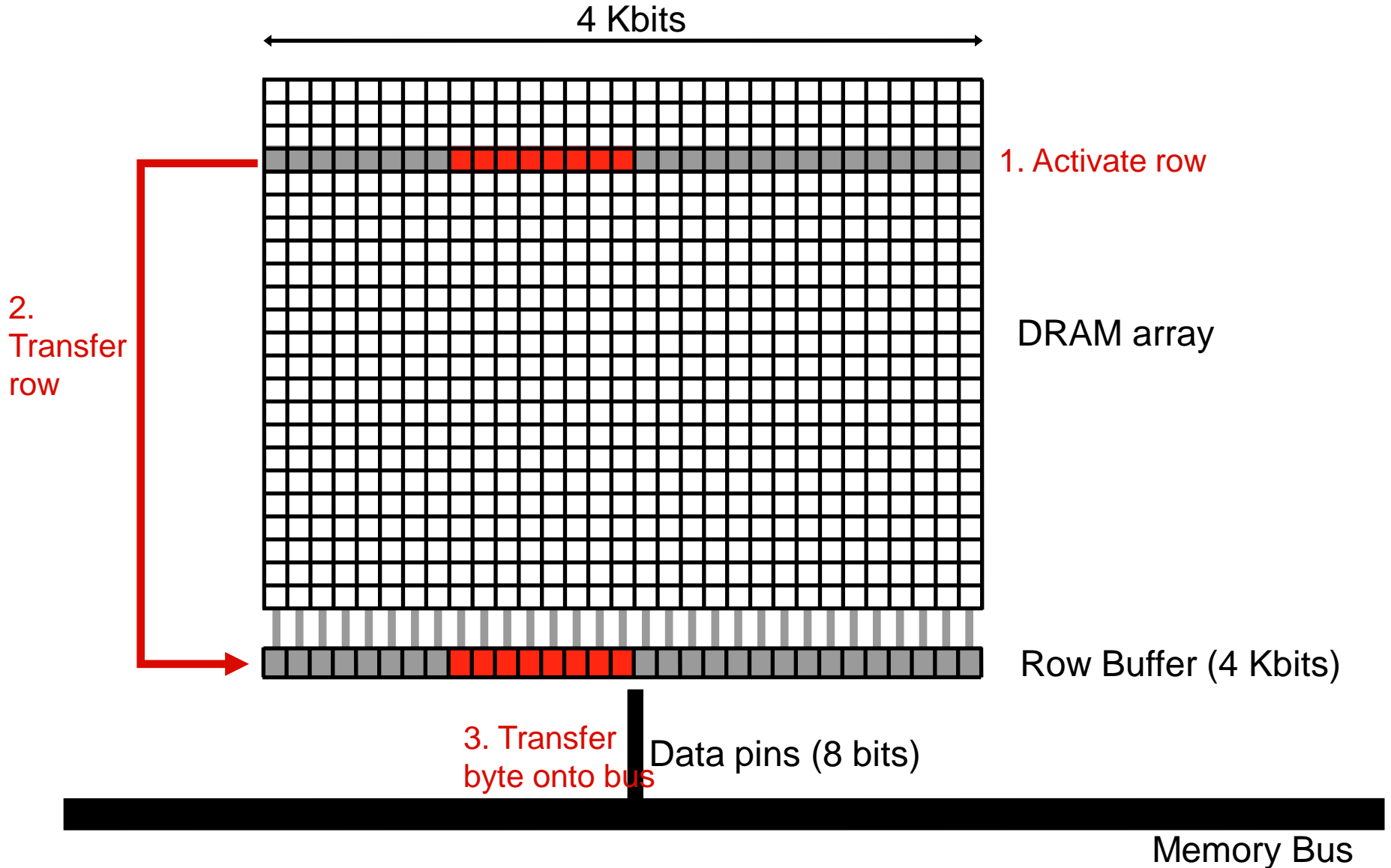


Future: RowClone (In-Memory Copy)

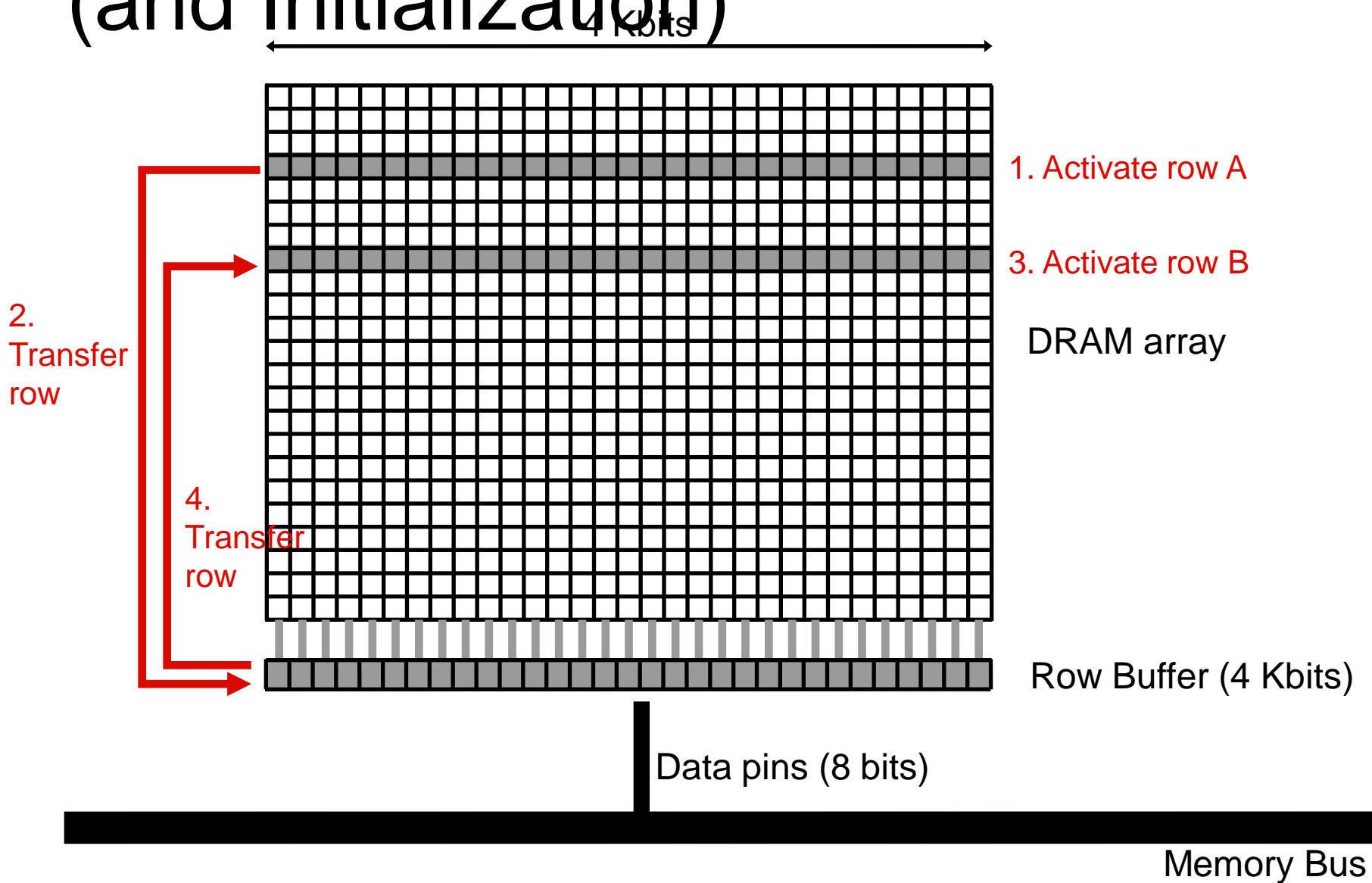


Seshadri et al., "RowClone: Fast and Efficient In-DRAM Copy and Initialization of Bulk Data," CMU Tech Report 2013.

DRAM operation (load one byte)



RowClone: in-DRAM Row Copy (and Initialization)

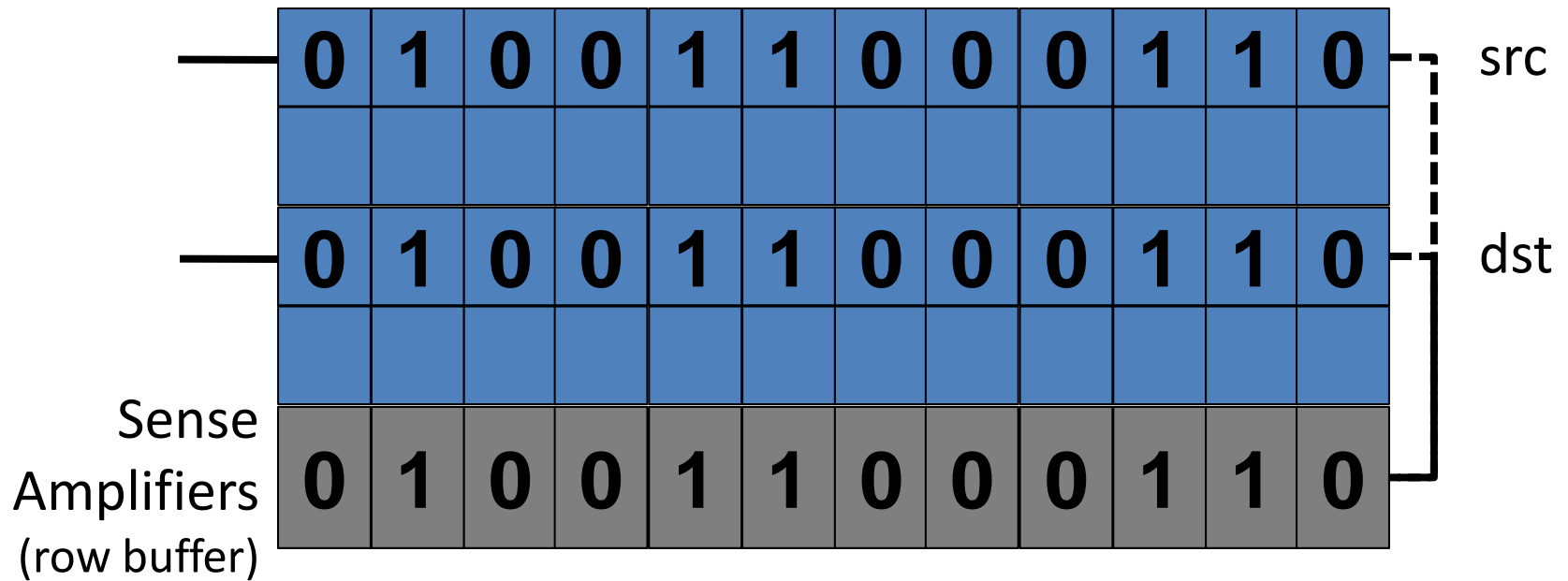


RowClone: Key Idea

- DRAM banks contain
 1. Multiple rows of DRAM cells – row = 8KB
 2. A row buffer shared by the DRAM rows
- Large scale copy
 1. Copy data from source row to row buffer
 2. Copy data from row buffer to destination row

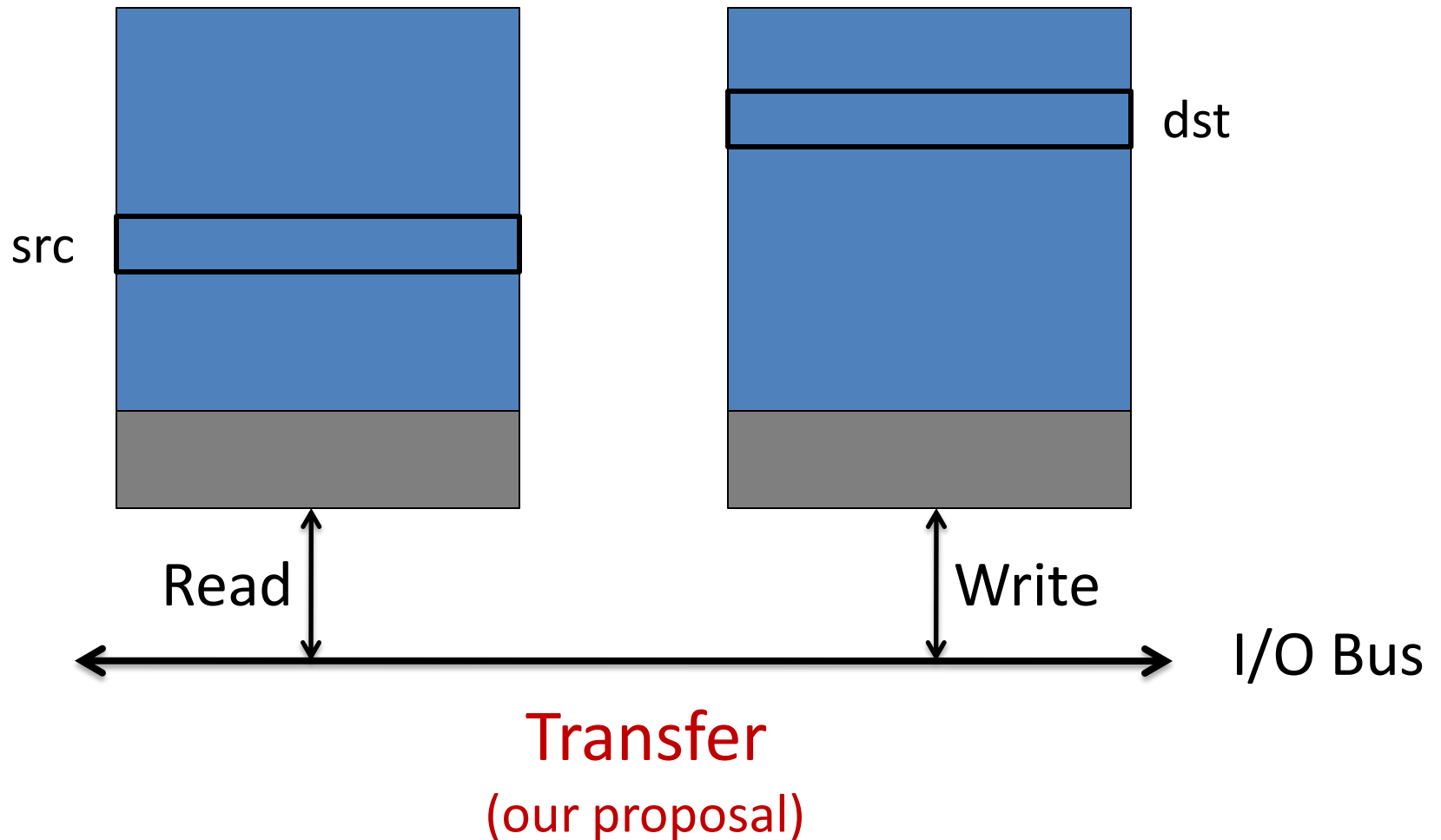
**Can be accomplished by two consecutive ACTIVATES
(if source and destination rows are in the same subarray)**

RowClone: Intra-subarray Copy

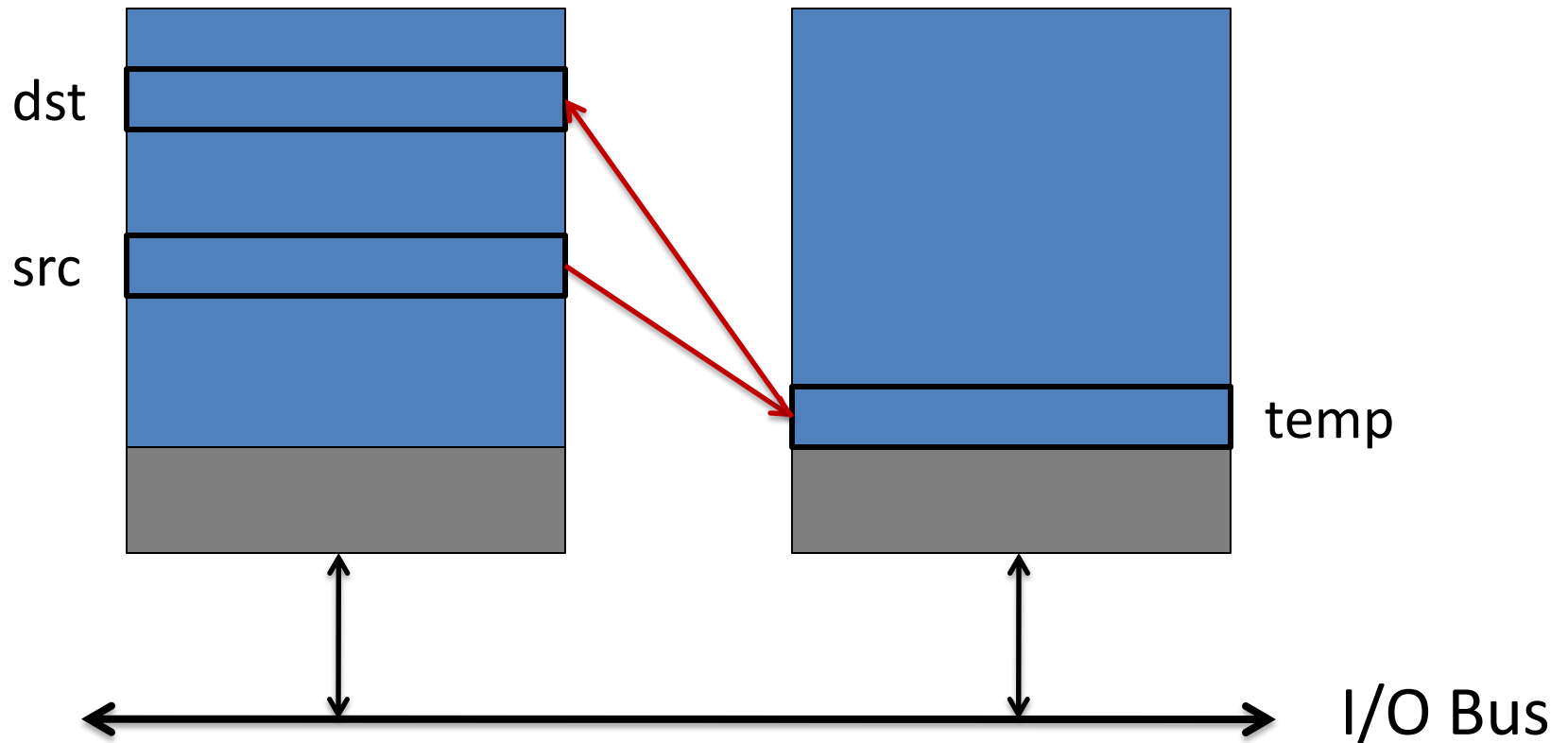


Activate (src) → Deactivate (our proposal) → Activate (dst)

RowClone: Inter-bank Copy

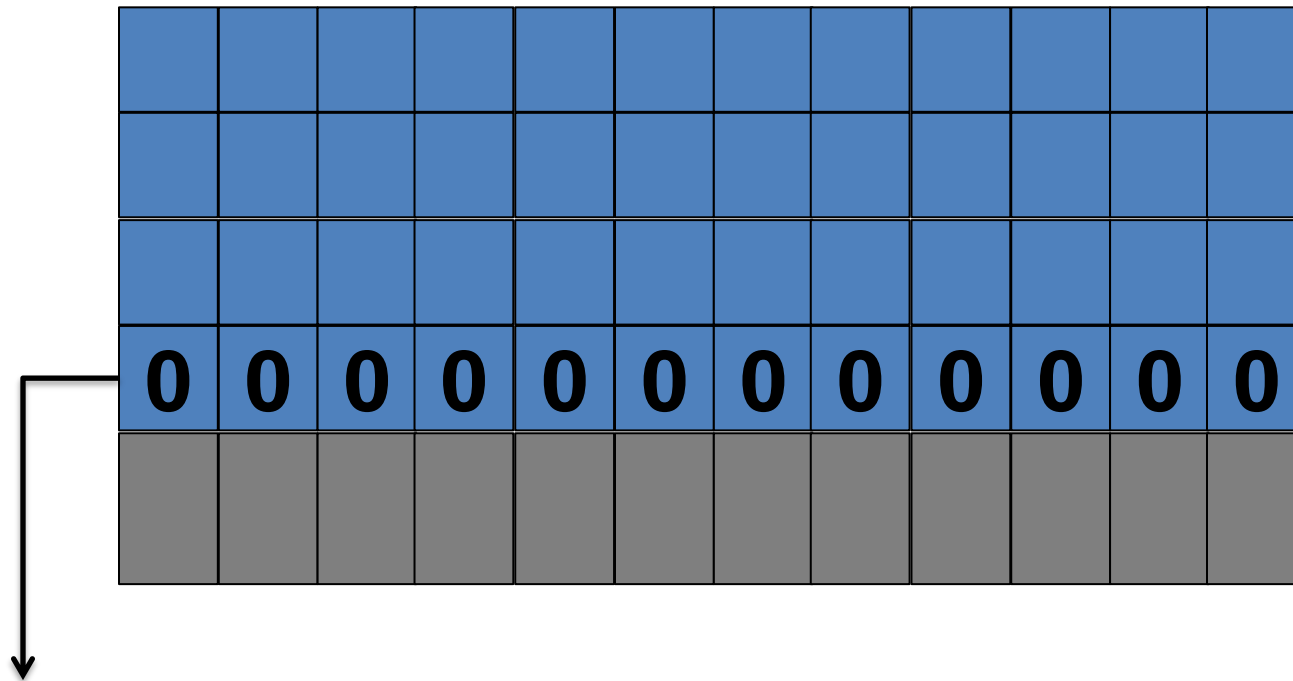


RowClone: Inter-subarray Copy



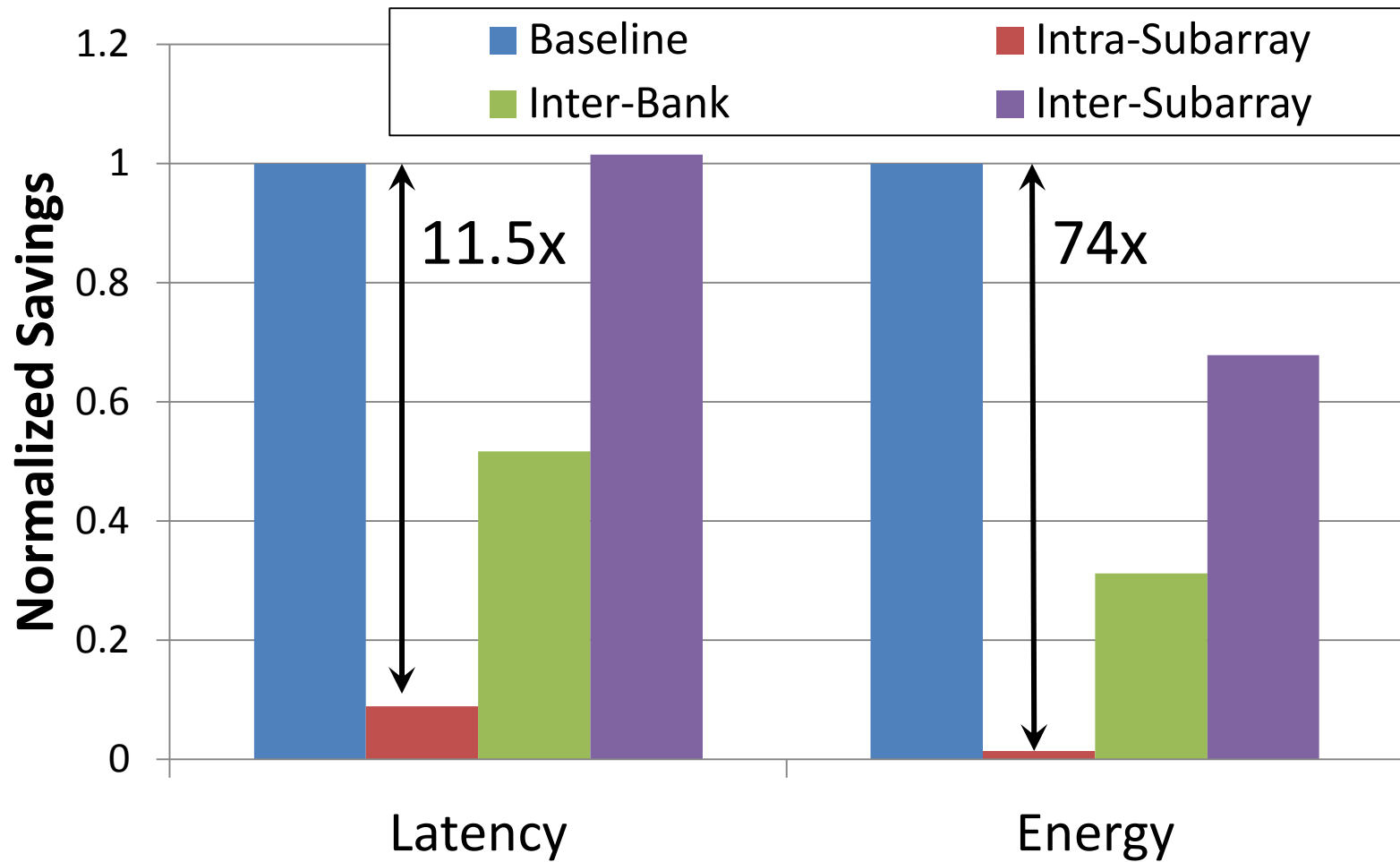
1. Transfer (src to temp)
2. Transfer (temp to dst)

Fast Row Initialization



Fix a row at Zero
(0.5% loss in capacity)

RowClone: Latency and Energy Savings



Seshadri et al., "RowClone: Fast and Efficient In-DRAM Copy and Initialization of Bulk Data," CMU Tech Report 2013.

RowClone: Latency and Energy Savings

Mechanism	Absolute		Reduction	
	Latency (ns)	Energy (μ J)	Latency	Energy
4KB Copy				
Baseline	1046	3.6	1.00	1.0
Intra-subarray	90	0.04	11.62	74.4
Inter-Bank - PSM	540	1.1	1.93	3.2
Intra-Bank - PSM	1050	2.5	0.99	1.5
4KB Zeroing				
Baseline	546	2.0	1.00	1.0
Intra-subarray	90	0.05	6.06	41.5

Table 3: Latency and energy reductions due to RowClone

RowClone: Overall Performance

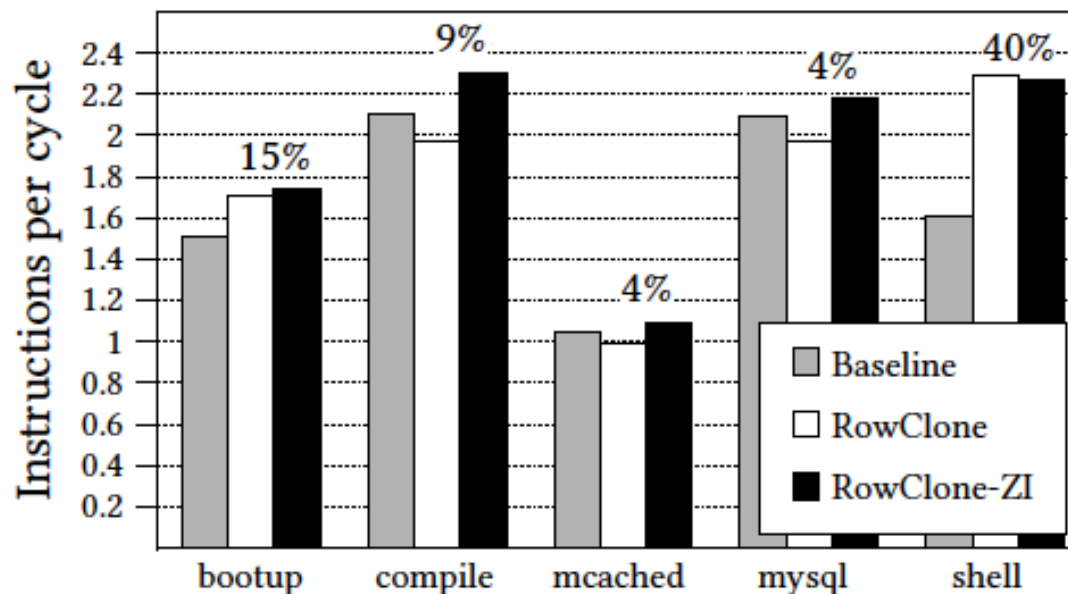


Figure 10: Performance improvement of RowClone-ZI. Value on top indicates percentage improvement compared to baseline.

Application	<i>bootup</i>	<i>compile</i>	<i>mcached</i>	<i>mysql</i>	<i>shell</i>
Energy Reduction	40%	32%	15%	17%	67%

Number of Cores	2	4	8
Number of Workloads	138	50	40
Weighted Speedup Improvement	15%	20%	27%
Energy per Instruction Reduction	19%	17%	17%

Agenda for Topic 1 (DRAM Scaling)

- What Will You Learn in This Mini-Lecture Series
- Main Memory Basics (with a Focus on DRAM)
- Major Trends Affecting Main Memory
- DRAM Scaling Problem and Solution Directions
- Solution Direction 1: System-DRAM Co-Design
- [Ongoing Research](#)
- Summary

Sampling of Ongoing Research

- Online retention time profiling
 - Preliminary work in ISCA 2013
 - Jamie Liu, Ben Jaiyen, Yoongu Kim, Chris Wilkerson, and Onur Mutlu, **"An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms"**
Proceedings of the 40th International Symposium on Computer Architecture (ISCA), Tel-Aviv, Israel, June 2013. [Slides \(pptx\)](#) [Slides \(pdf\)](#)
- More computation in memory and controllers
- Refresh/demand parallelization

Agenda for Topic 1 (DRAM Scaling)

- What Will You Learn in This Mini-Lecture Series
- Main Memory Basics (with a Focus on DRAM)
- Major Trends Affecting Main Memory
- DRAM Scaling Problem and Solution Directions
- Solution Direction 1: System-DRAM Co-Design
- Ongoing Research
- **Summary**

Summary

- Major problems with DRAM scaling and design: high refresh rate, high latency, low parallelism, bulk data movement
- Four new DRAM designs
 - RAIDR: Reduces refresh impact
 - TL-DRAM: Reduces DRAM latency at low cost
 - SALP: Improves DRAM parallelism
 - RowClone: Reduces energy and performance impact of bulk data copy
- All four designs
 - Improve both performance and energy consumption
 - Are low cost (low DRAM area overhead)
 - Enable new degrees of freedom to software & controllers
- Rethinking DRAM interface and design essential for scaling
 - Co-design DRAM with the rest of the system

Further Reading: Data Retention and Power

■ Characterization of Commodity DRAM Chips

- Jamie Liu, Ben Jaiyen, Yoongu Kim, Chris Wilkerson, and Onur Mutlu, **"An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms"**
Proceedings of the 40th International Symposium on Computer Architecture (ISCA), Tel-Aviv, Israel, June 2013. [Slides \(pptx\)](#) [Slides \(pdf\)](#)

■ Voltage and Frequency Scaling in DRAM

- Howard David, Chris Fallin, Eugene Gorbатов, Ulf R. Hanebutte, and Onur Mutlu, **"Memory Power Management via Dynamic Voltage/Frequency Scaling"**
Proceedings of the 8th International Conference on Autonomic Computing (ICAC), Karlsruhe, Germany, June 2011. [Slides \(pptx\)](#) [\(pdf\)](#)

Scalable Many-Core Memory Systems

Topic 1: DRAM Basics and DRAM Scaling

Prof. Onur Mutlu

<http://www.ece.cmu.edu/~omutlu>

onur@cmu.edu

HiPEAC ACACES Summer School 2013

July 15-19, 2013

Carnegie Mellon

Computer Architecture: Main Memory (Part III)

Prof. Onur Mutlu
Carnegie Mellon University

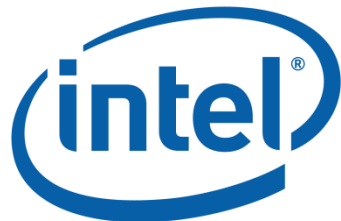
Additional Material (Not Covered in Lecture)

Three Papers

- Howard David, Chris Fallin, Eugene Gorbatoov, Ulf R. Hanebutte, and Onur Mutlu, **"Memory Power Management via Dynamic Voltage/Frequency Scaling"**
Proceedings of the 8th International Conference on Autonomic Computing (ICAC), Karlsruhe, Germany, June 2011. [Slides \(pptx\)](#) [\(pdf\)](#)
- Jamie Liu, Ben Jaiyen, Yoongu Kim, Chris Wilkerson, and Onur Mutlu, **"An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms"**
Proceedings of the 40th International Symposium on Computer Architecture (ISCA), Tel-Aviv, Israel, June 2013. [Slides \(pptx\)](#) [Slides \(pdf\)](#)
- Yu Cai, Gulay Yalcin, Onur Mutlu, Erich F. Haratsch, Adrian Cristal, Osman Unsal, and Ken Mai, **"Error Analysis and Retention-Aware Error Management for NAND Flash Memory"**
Intel Technology Journal (ITJ) Special Issue on Memory Resiliency, Vol. 17, No. 1, May 2013.

Memory Power Management via Dynamic Voltage/Frequency Scaling

Howard David (Intel)
Eugene Gorbatov (Intel)
Ulf R. Hanebutte (Intel)

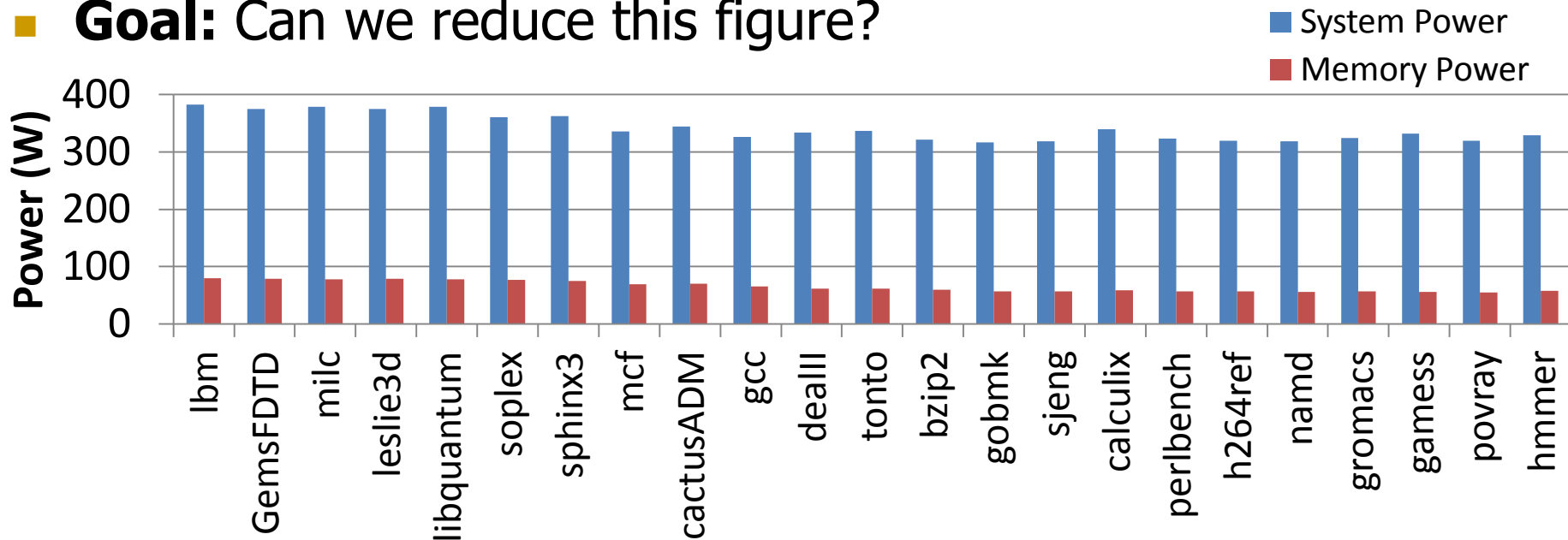


Chris Fallin (CMU)
Onur Mutlu (CMU)

SAFARI
Carnegie Mellon

Memory Power is Significant

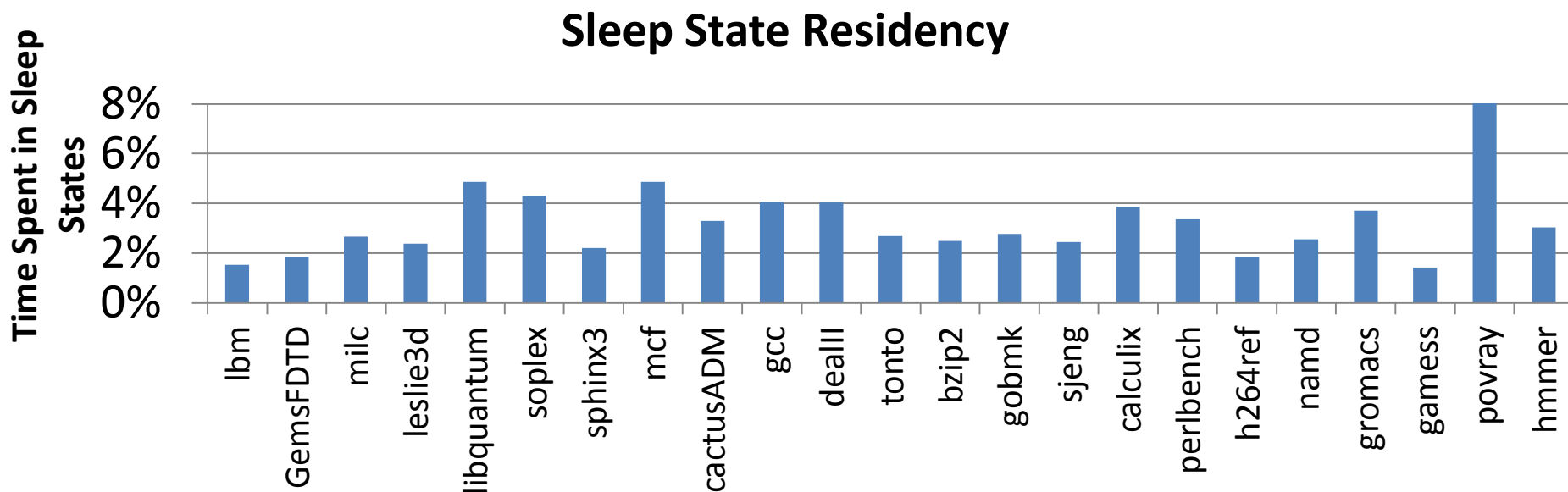
- **Power consumption** is a primary concern in modern servers
- Many works: **CPU**, **whole-system** or **cluster-level** approach
- But **memory power** is largely unaddressed
- Our server system*: **memory** is 19% of system power (avg)
 - Some work notes up to 40% of total system power
- **Goal:** Can we reduce this figure?



*Dual 4-core Intel Xeon®, 48GB DDR3 (12 DIMMs), SPEC CPU2006, all cores active.
Measured AC power, analytically modeled memory power.

Existing Solution: Memory Sleep States?

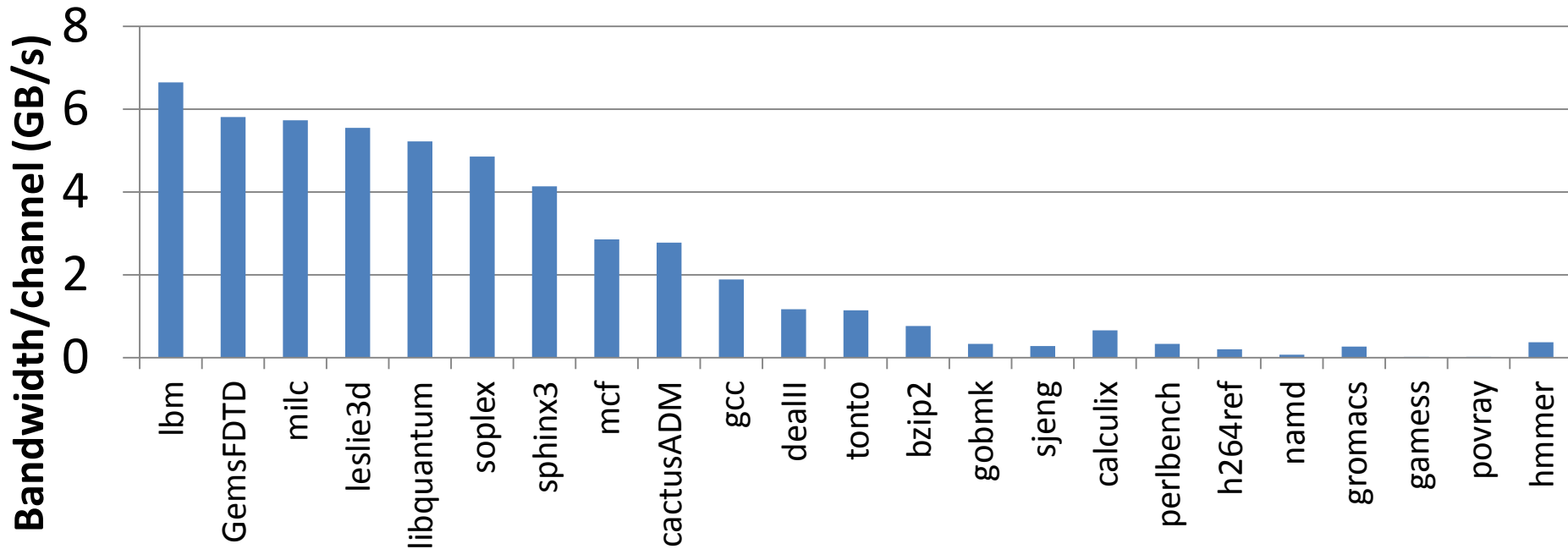
- Most memory energy-efficiency work uses **sleep states**
 - Shut down DRAM devices when no memory requests active
- But, even low-memory-bandwidth workloads keep memory awake
 - Idle periods between requests diminish in multicore workloads
 - CPU-bound workloads/phases rarely completely cache-resident



Memory Bandwidth Varies Widely

- Workload **memory bandwidth requirements** vary widely

Memory Bandwidth for SPEC CPU2006



- Memory system is provisioned for peak capacity
→ often **underutilized**

Memory Power can be Scaled Down

- DDR can operate at multiple frequencies → **reduce power**
 - Lower frequency directly **reduces switching power**
 - Lower frequency allows for **lower voltage**
 - **Comparable to CPU DVFS**

CPU Voltage/Freq.	System Power
↓ 15%	↓ 9.9%

Memory Freq.	System Power
↓ 40%	↓ 7.6%

- Frequency scaling increases latency → **reduce performance**
 - Memory storage array is asynchronous
 - But, **bus transfer depends on frequency**
 - When bus bandwidth is bottleneck, performance suffers

Observations So Far

- **Memory power** is a significant portion of total power
 - 19% (avg) in our system, up to 40% noted in other works
- **Sleep state residency** is low in many workloads
 - Multicore workloads reduce idle periods
 - CPU-bound applications send requests frequently enough to keep memory devices awake
- **Memory bandwidth demand** is very low in some workloads
- Memory power is reduced by **frequency scaling**
 - And **voltage scaling** can give further reductions

DVFS for Memory

- **Key Idea:** observe memory bandwidth utilization, then adjust memory frequency/voltage, to **reduce power** with **minimal performance loss**
 - **Dynamic Voltage/Frequency Scaling (DVFS) for memory**
- **Goal in this work:**
 - Implement **DVFS** in the memory system, by:
 - Developing a **simple control algorithm** to exploit opportunity for reduced memory frequency/voltage by observing behavior
 - Evaluating the proposed algorithm on a **real system**

Outline

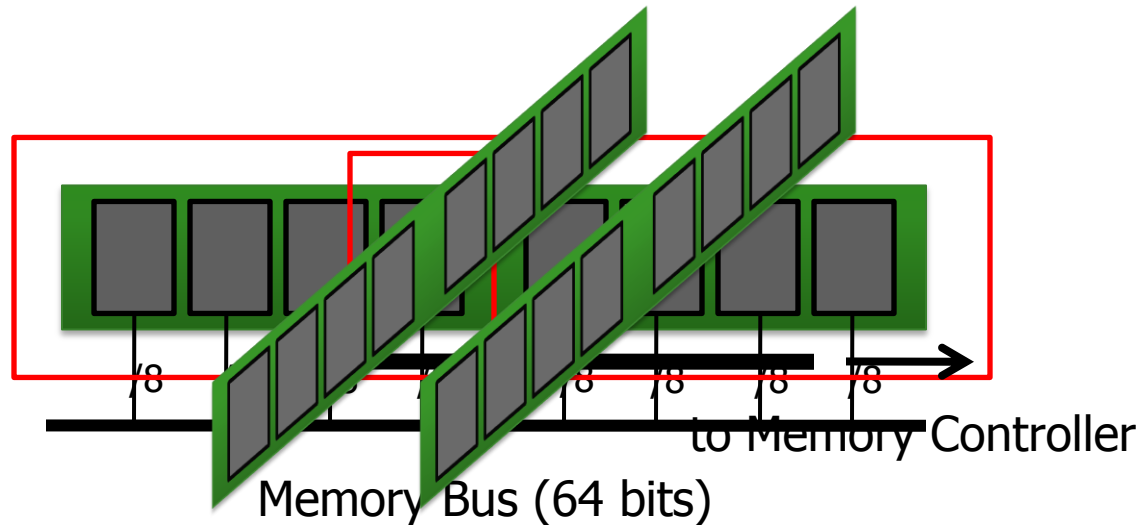
- Motivation
- Background and Characterization
 - DRAM Operation
 - DRAM Power
 - Frequency and Voltage Scaling
- Performance Effects of Frequency Scaling
- Frequency Control Algorithm
- Evaluation and Conclusions

Outline

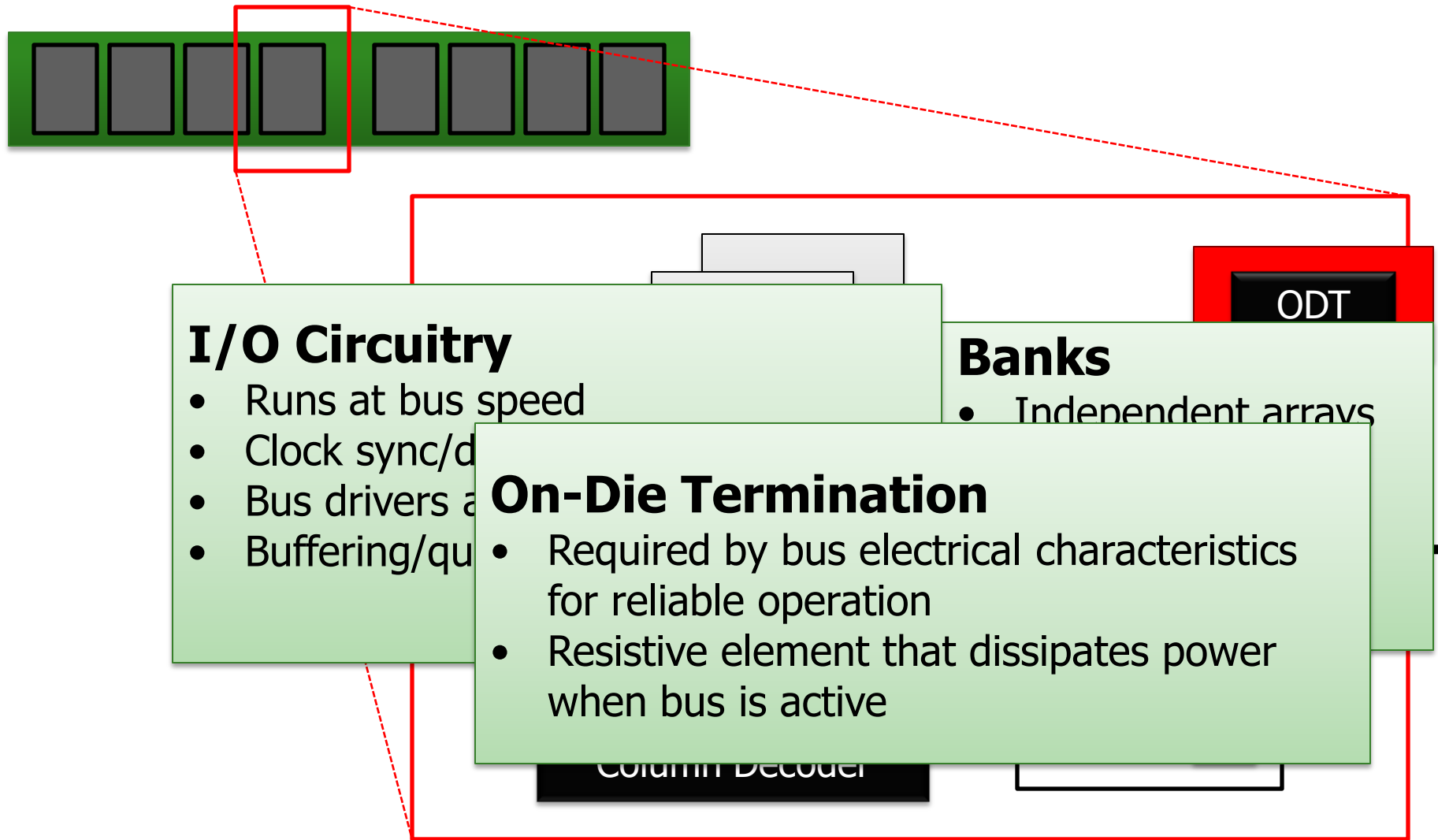
- Motivation
- Background and Characterization
 - DRAM Operation
 - DRAM Power
 - Frequency and Voltage Scaling
- Performance Effects of Frequency Scaling
- Frequency Control Algorithm
- Evaluation and Conclusions

DRAM Operation

- Main memory consists of DIMMs of DRAM devices
- Each DIMM is attached to a memory bus (channel)
- Multiple DIMMs can connect to one channel



Inside a DRAM Device



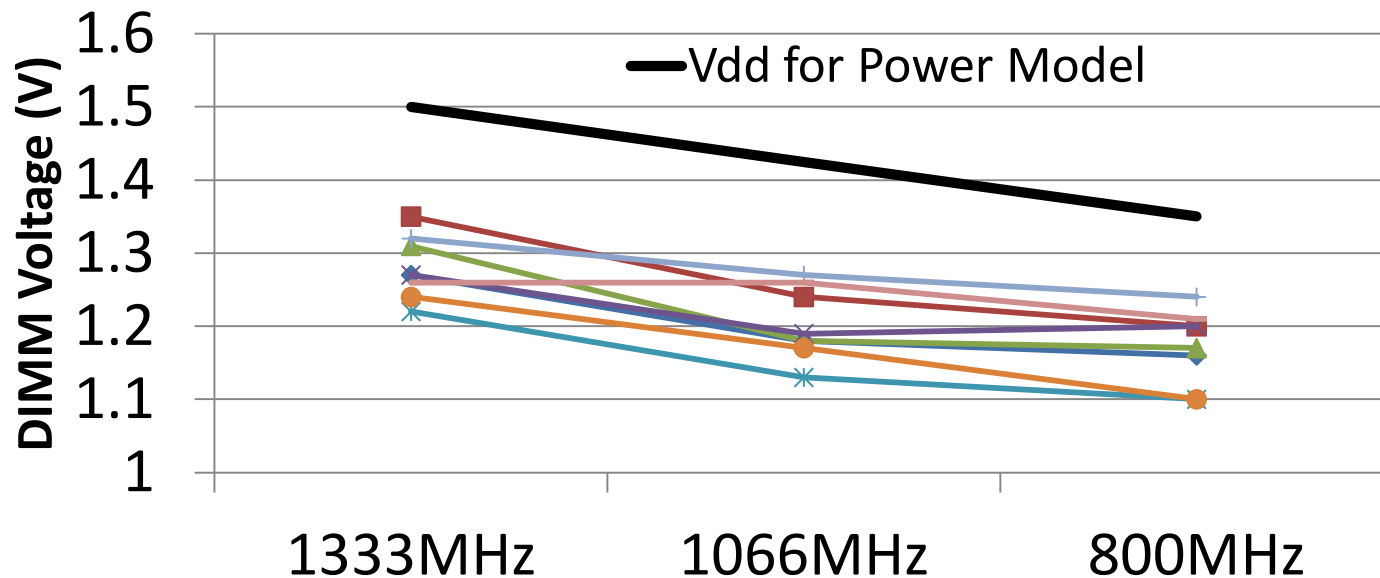
Effect of Frequency Scaling on Power

- **Reduced memory bus frequency:**
- **Does not affect bank power:**
 - ❑ Constant energy per operation
 - ❑ Depends only on utilized memory bandwidth
- **Decreases I/O power:**
 - ❑ Dynamic power in bus interface and clock circuitry reduces due to less frequent switching
- **Increases termination power:**
 - ❑ Same data takes longer to transfer
 - ❑ Hence, bus utilization increases
- **Tradeoff between I/O and termination results in a net power reduction at lower frequencies**

Effects of Voltage Scaling on Power

- Voltage scaling further reduces power because all parts of memory devices will draw **less current (at less voltage)**
- Voltage reduction is possible because stable operation requires **lower voltage at lower frequency**:

Minimum Stable Voltage for 8 DIMMs in a Real System

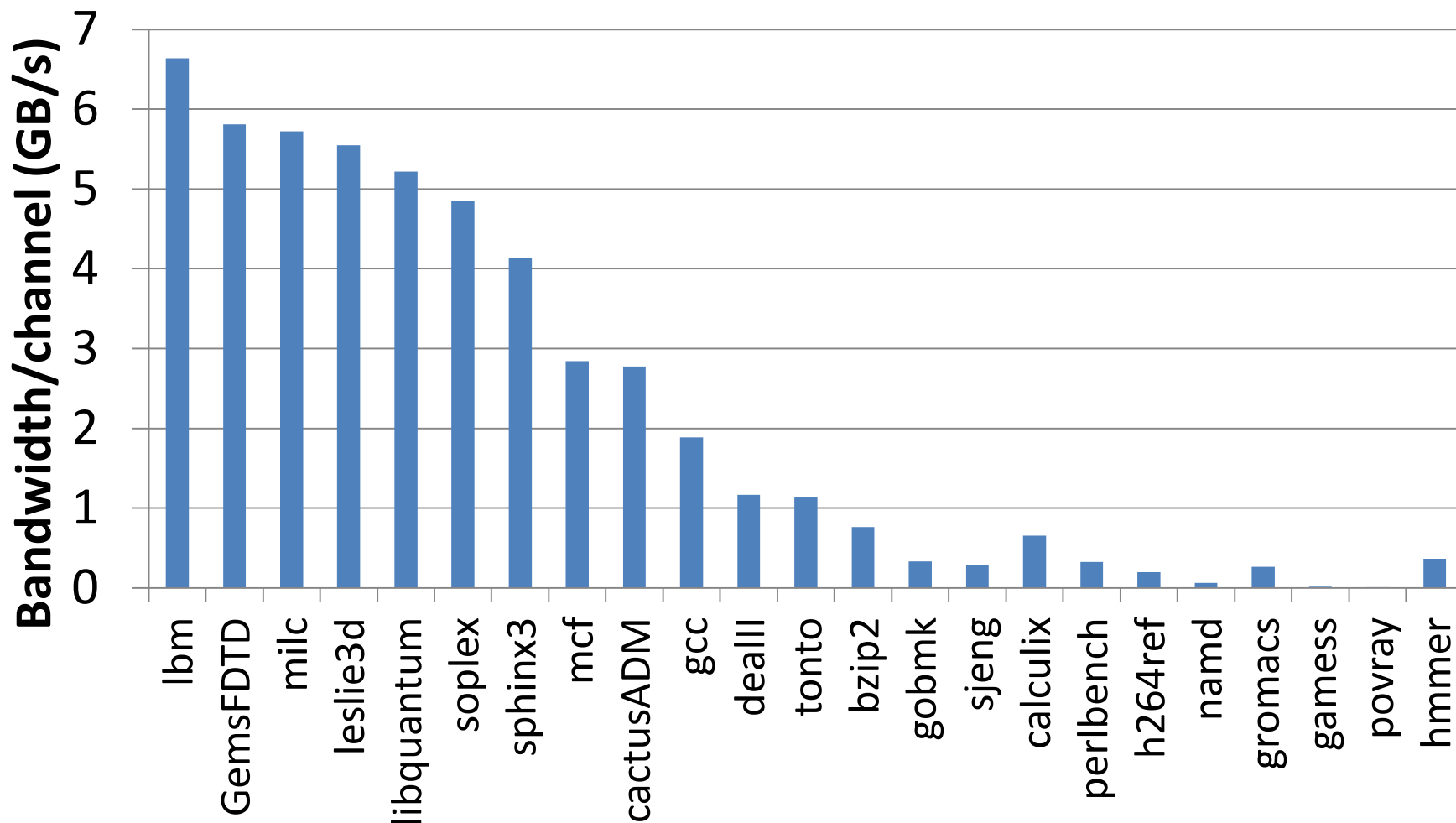


Outline

- Motivation
- Background and Characterization
 - DRAM Operation
 - DRAM Power
 - Frequency and Voltage Scaling
- **Performance Effects of Frequency Scaling**
- Frequency Control Algorithm
- Evaluation and Conclusions

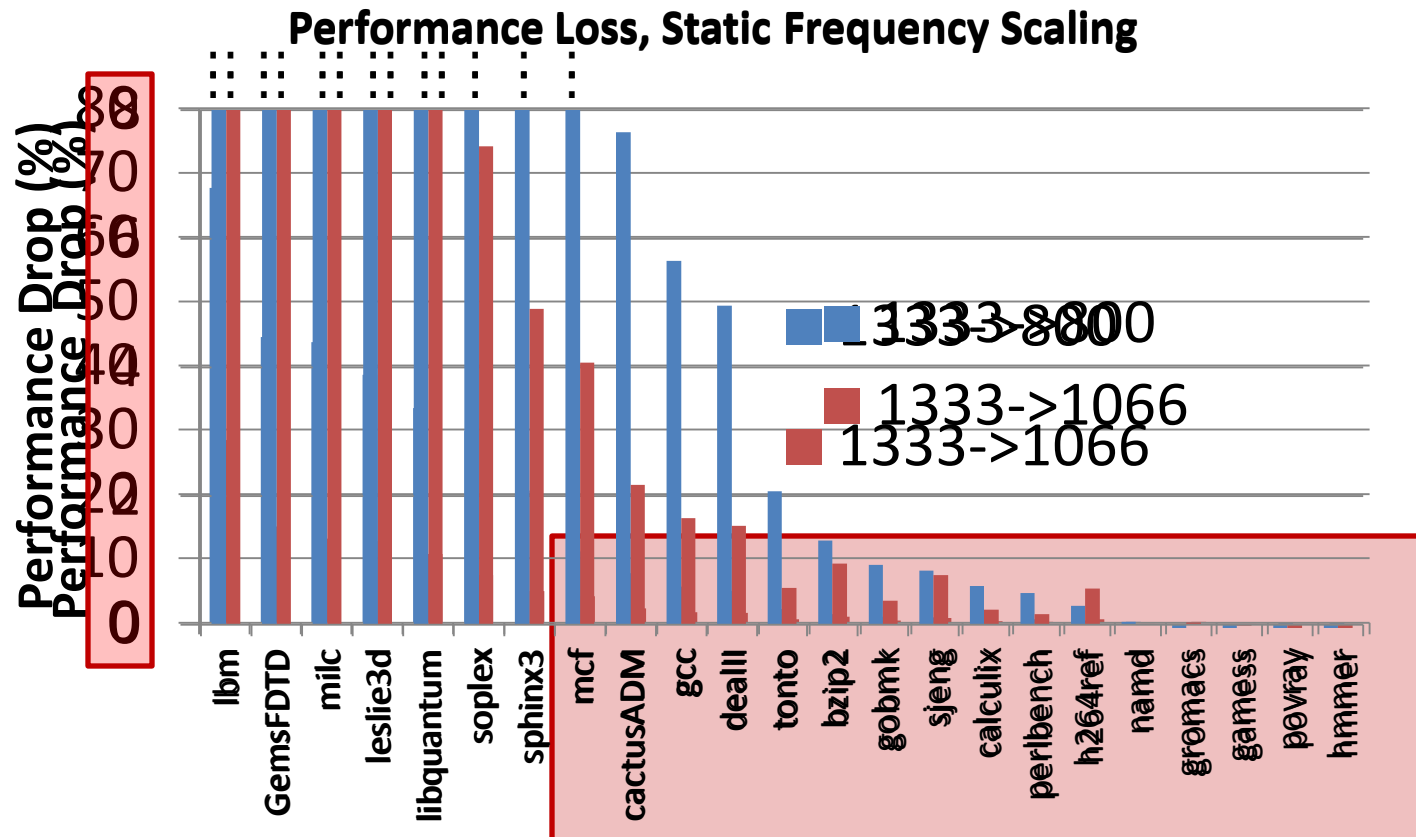
How Much Memory Bandwidth is Needed?

Memory Bandwidth for SPEC CPU2006



Performance Impact of Static Frequency Scaling

- Performance impact is proportional to bandwidth demand
- Many workloads tolerate lower frequency with minimal performance drop



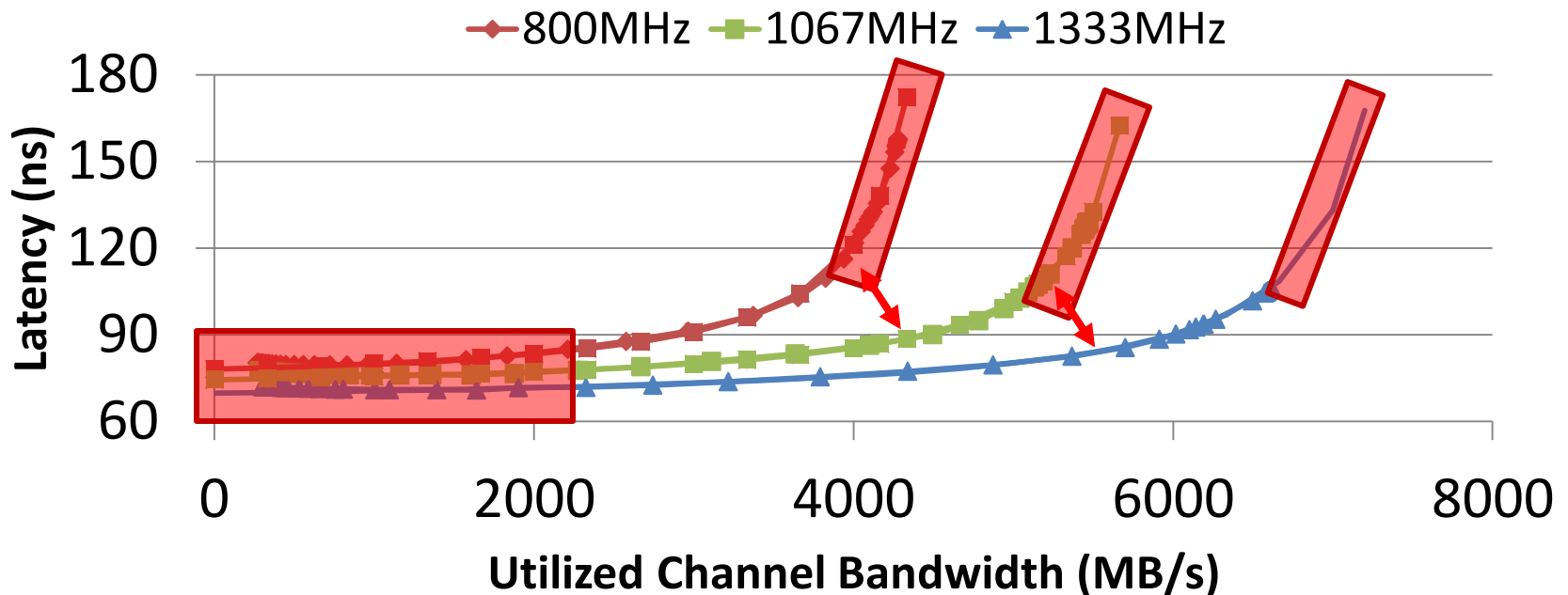
Outline

- Motivation
- Background and Characterization
 - DRAM Operation
 - DRAM Power
 - Frequency and Voltage Scaling
- Performance Effects of Frequency Scaling
- **Frequency Control Algorithm**
- Evaluation and Conclusions

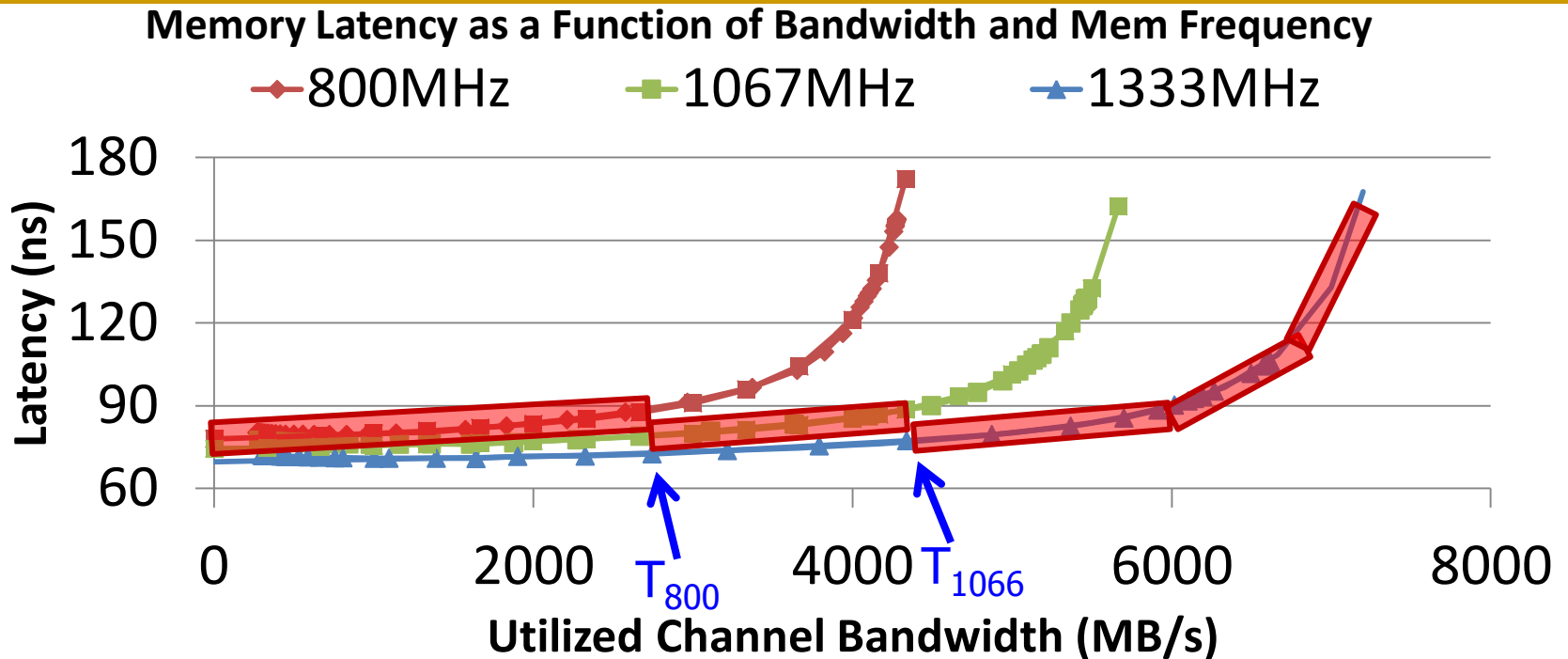
Memory Latency Under Load

- At **low load**, most time is in array access and bus transfer
→ **small constant offset** between bus-frequency latency curves
- As **load increases**, queueing delay begins to dominate
→ bus frequency **significantly affects latency**

Memory Latency as a Function of Bandwidth and Mem Frequency



Control Algorithm: Demand-Based Switching



After each epoch of length T_{epoch} :

Measure per-channel bandwidth BW

if $BW < T_{800}$: switch to 800MHz

else if $BW < T_{1066}$: switch to 1066MHz

else : switch to 1333MHz

Implementing V/F Switching

- **Halt Memory Operations**
 - Pause requests
 - Put DRAM in Self-Refresh
 - Stop the DIMM clock
- **Transition Voltage/Frequency**
 - Begin voltage ramp

- 👍 Memory frequency already adjustable statically
- 👍 Voltage regulators for CPU DVFS can work for memory DVFS
- 👍 Full transition takes $\sim 20\mu\text{s}$

Outline

- Motivation
- Background and Characterization
 - DRAM Operation
 - DRAM Power
 - Frequency and Voltage Scaling
- Performance Effects of Frequency Scaling
- Frequency Control Algorithm
- Evaluation and Conclusions

Evaluation Methodology

■ **Real-system evaluation**

- ❑ Dual 4-core Intel Xeon®, 3 memory channels/socket
- ❑ 48 GB of DDR3 (12 DIMMs, 4GB dual-rank, 1333MHz)

■ **Emulating memory frequency for performance**

- ❑ Altered memory controller **timing registers** (tRC, tB2BCAS)
- ❑ Gives performance equivalent to slower memory frequencies

■ **Modeling power reduction**

- ❑ Measure baseline system (AC power meter, 1s samples)
- ❑ Compute reductions with an analytical model (see paper)

Evaluation Methodology

■ Workloads

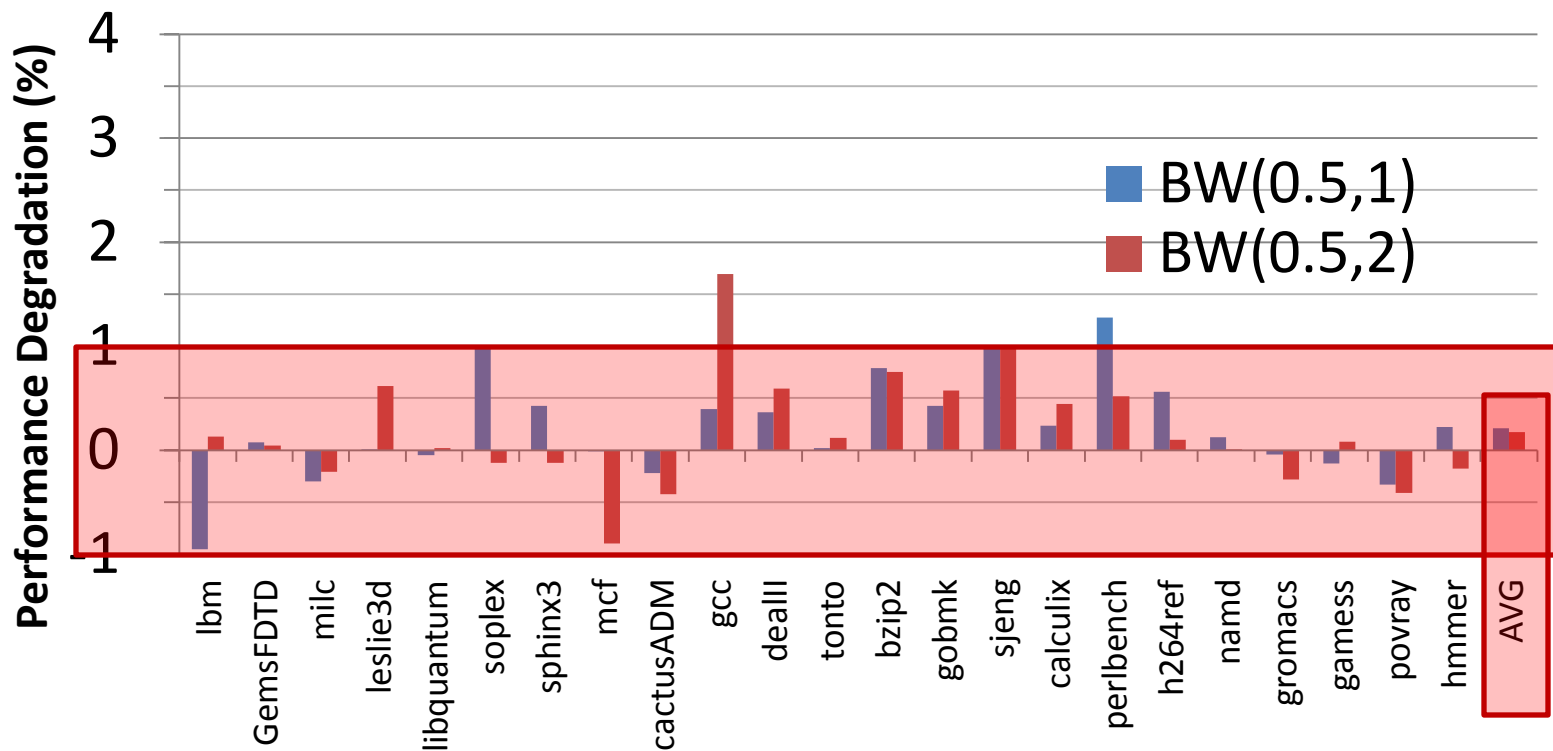
- ❑ SPEC CPU2006: CPU-intensive workloads
- ❑ All cores run a copy of the benchmark

■ Parameters

- ❑ $T_{\text{epoch}} = 10\text{ms}$
- ❑ Two variants of algorithm with different switching thresholds:
- ❑ BW(0.5, 1): $T_{800} = 0.5\text{GB/s}$, $T_{1066} = 1\text{GB/s}$
- ❑ BW(0.5, 2): $T_{800} = 0.5\text{GB/s}$, $T_{1066} = 2\text{GB/s}$
 - More aggressive frequency/voltage scaling

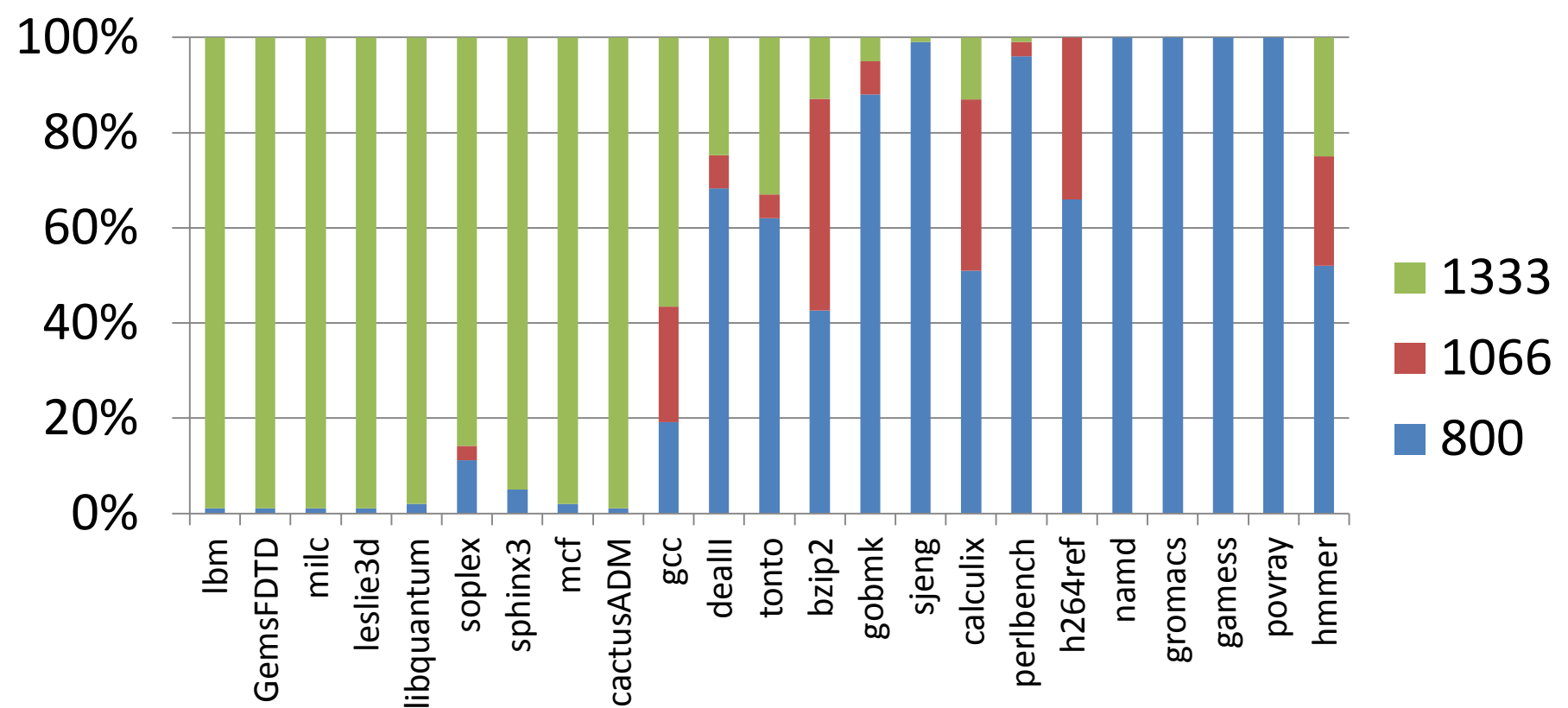
Performance Impact of Memory DVFS

- Minimal performance degradation: 0.2% (avg), 1.7% (max)
- Experimental error $\sim 1\%$



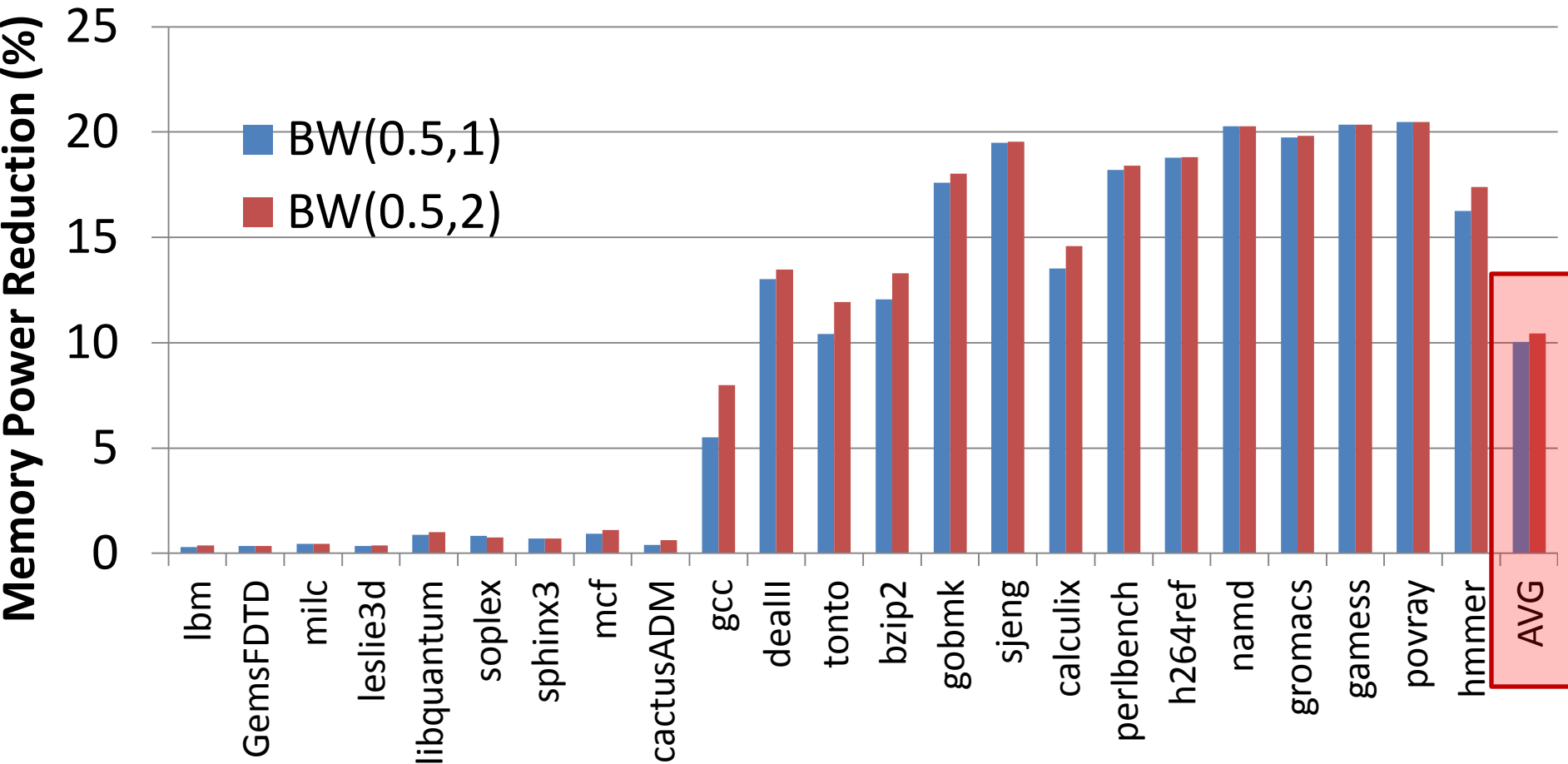
Memory Frequency Distribution

- Frequency distribution shifts toward higher memory frequencies with more memory-intensive benchmarks



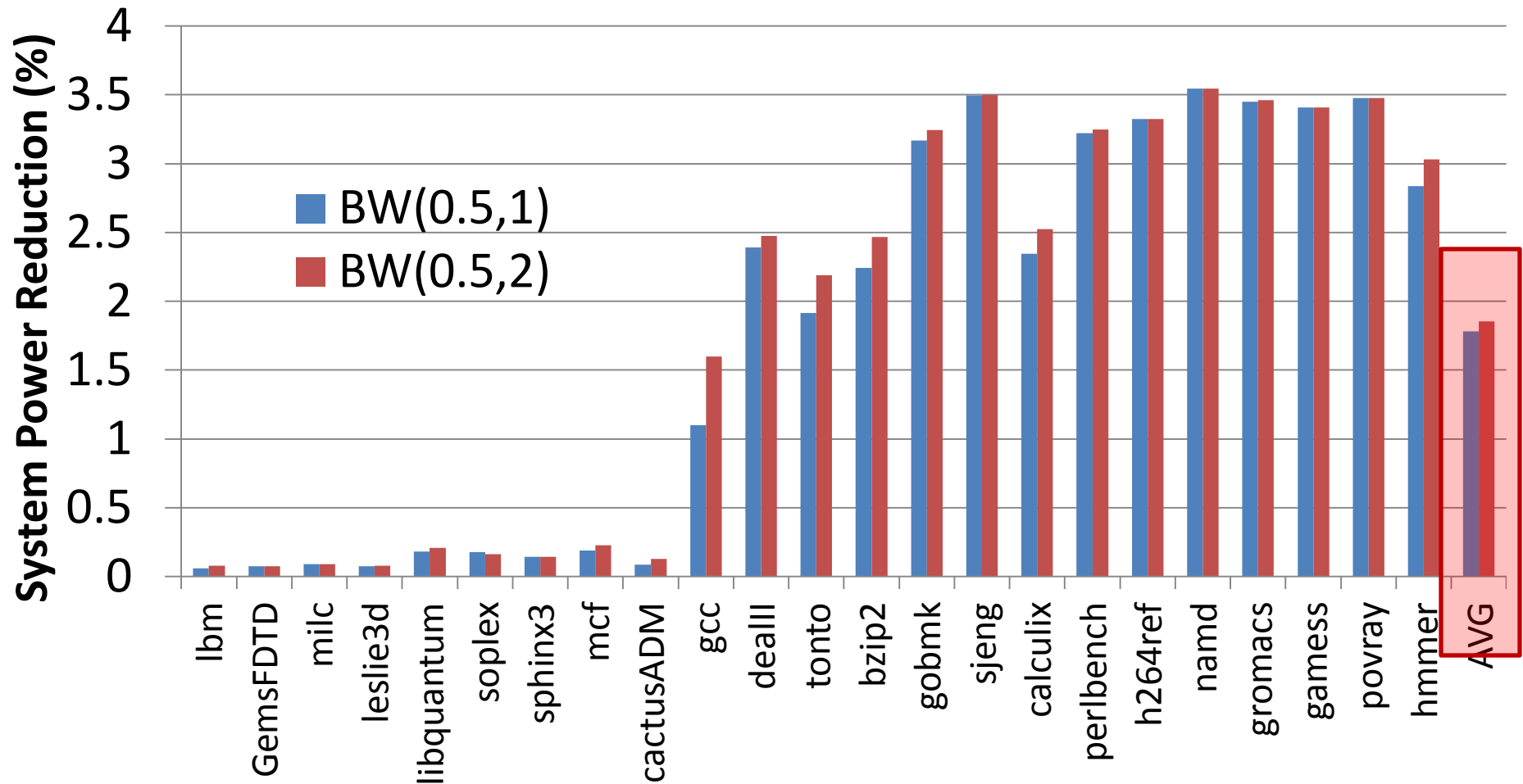
Memory Power Reduction

- Memory power reduces by 10.4% (avg), 20.5% (max)



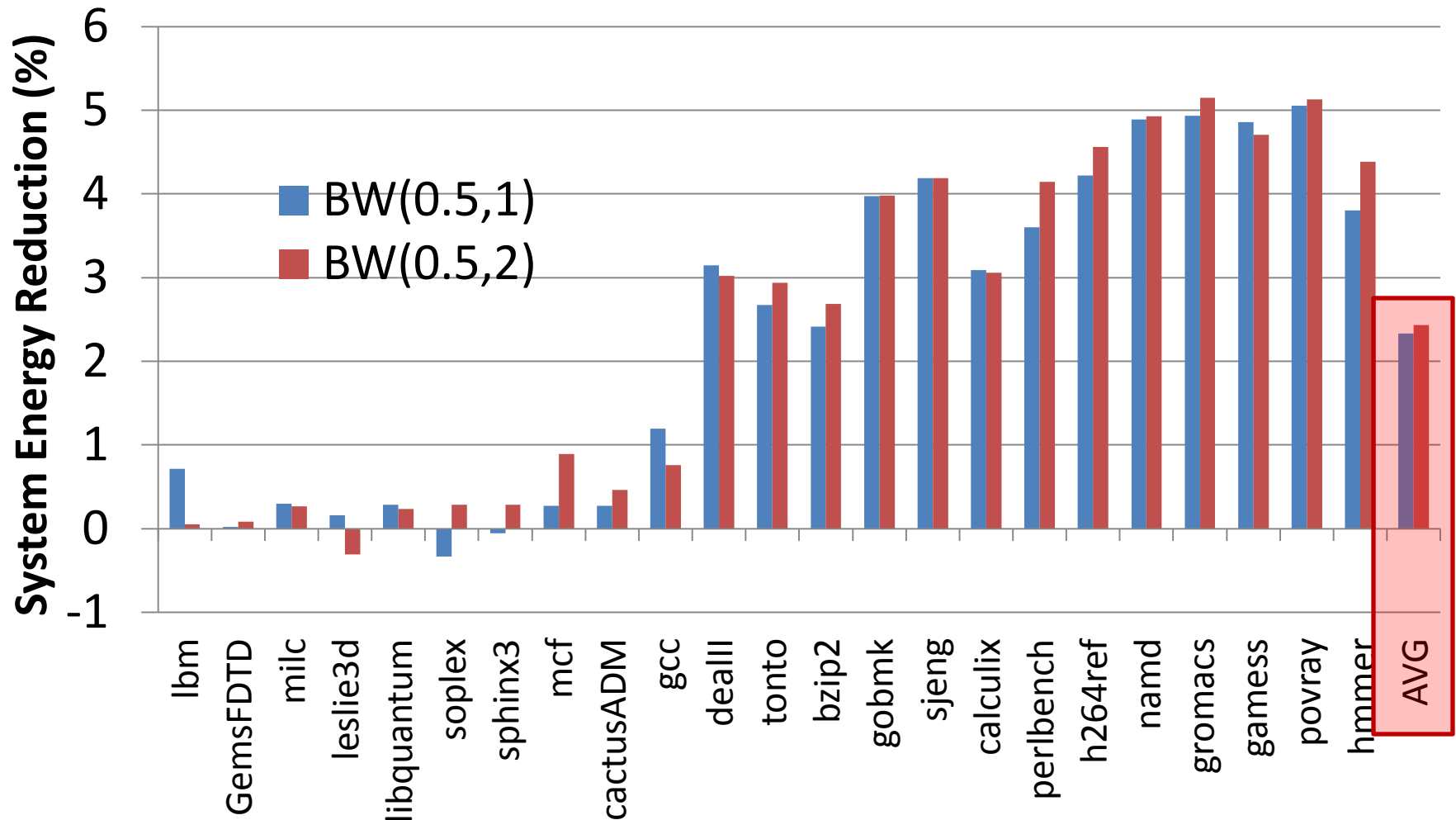
System Power Reduction

- As a result, system power reduces by 1.9% (avg), 3.5% (max)



System Energy Reduction

- System energy reduces by 2.4% (avg), 5.1% (max)



Related Work

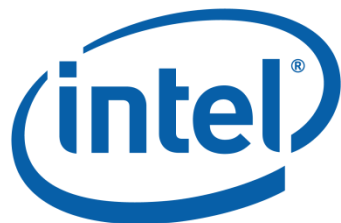
- **MemScale** [Deng11], concurrent work (ASPLOS 2011)
 - Also proposes Memory DVFS
 - **Application performance impact model to decide voltage and frequency**: requires specific modeling for a given system; our bandwidth-based approach avoids this complexity
 - **Simulation-based evaluation**; our work is a real-system proof of concept
- **Memory Sleep States** (Creating opportunity with data placement [Lebeck00,Pandey06], OS scheduling [Delaluz02], VM subsystem [Huang05]; Making better decisions with better models [Hur08,Fan01])
- **Power Limiting/Shifting** (RAPL [David10] uses memory throttling for thermal limits; CPU throttling for memory traffic [Lin07,08]; Power shifting across system [Felter05])

Conclusions

- Memory power is a **significant component** of system power
 - 19% average in our evaluation system, 40% in other work
 - Workloads often keep memory **active** but **underutilized**
 - Channel bandwidth demands are highly variable
 - Use of memory sleep states is often limited
 - Scaling **memory frequency/voltage** can reduce memory power with minimal system performance impact
 - 10.4% average memory power reduction
 - Yields 2.4% average system energy reduction
 - Greater reductions are possible with wider frequency/voltage range and better control algorithms
-

Memory Power Management via Dynamic Voltage/Frequency Scaling

Howard David (Intel)
Eugene Gorbatov (Intel)
Ulf R. Hanebutte (Intel)



Chris Fallin (CMU)
Onur Mutlu (CMU)

SAFARI
Carnegie Mellon

An Experimental Study of Data Retention Behavior in Modern DRAM Devices

Implications for Retention Time Profiling Mechanisms

Jamie Liu¹ Ben Jaiyen¹ Yoongu Kim¹

Chris Wilkerson² Onur Mutlu¹

¹ Carnegie Mellon University

² Intel Corporation

Summary (I)

- DRAM requires periodic refresh to avoid data loss
 - Refresh wastes energy, reduces performance, limits DRAM density scaling
- Many past works observed that different DRAM cells can retain data for different times without being refreshed; proposed reducing refresh rate for strong DRAM cells
 - **Problem:** These techniques require an accurate profile of the retention time of all DRAM cells
- Our goal: To analyze the retention time behavior of DRAM cells in modern DRAM devices to aid the collection of accurate profile information
- Our experiments: We characterize 248 modern commodity DDR3 DRAM chips from 5 manufacturers using an FPGA based testing platform
- Two Key Issues:
 1. **Data Pattern Dependence:** A cell's retention time is heavily dependent on data values stored in itself and nearby cells, which cannot easily be controlled.
 2. **Variable Retention Time:** Retention time of some cells change unpredictably from high to low at large timescales.

Summary (II)

■ Key findings on Data Pattern Dependence

- There is no observed single data pattern that elicits the lowest retention times for a DRAM device → very hard to find this pattern
- DPD varies between devices due to variation in DRAM array circuit design between manufacturers
- DPD of retention time gets worse as DRAM scales to smaller feature sizes

■ Key findings on Variable Retention Time

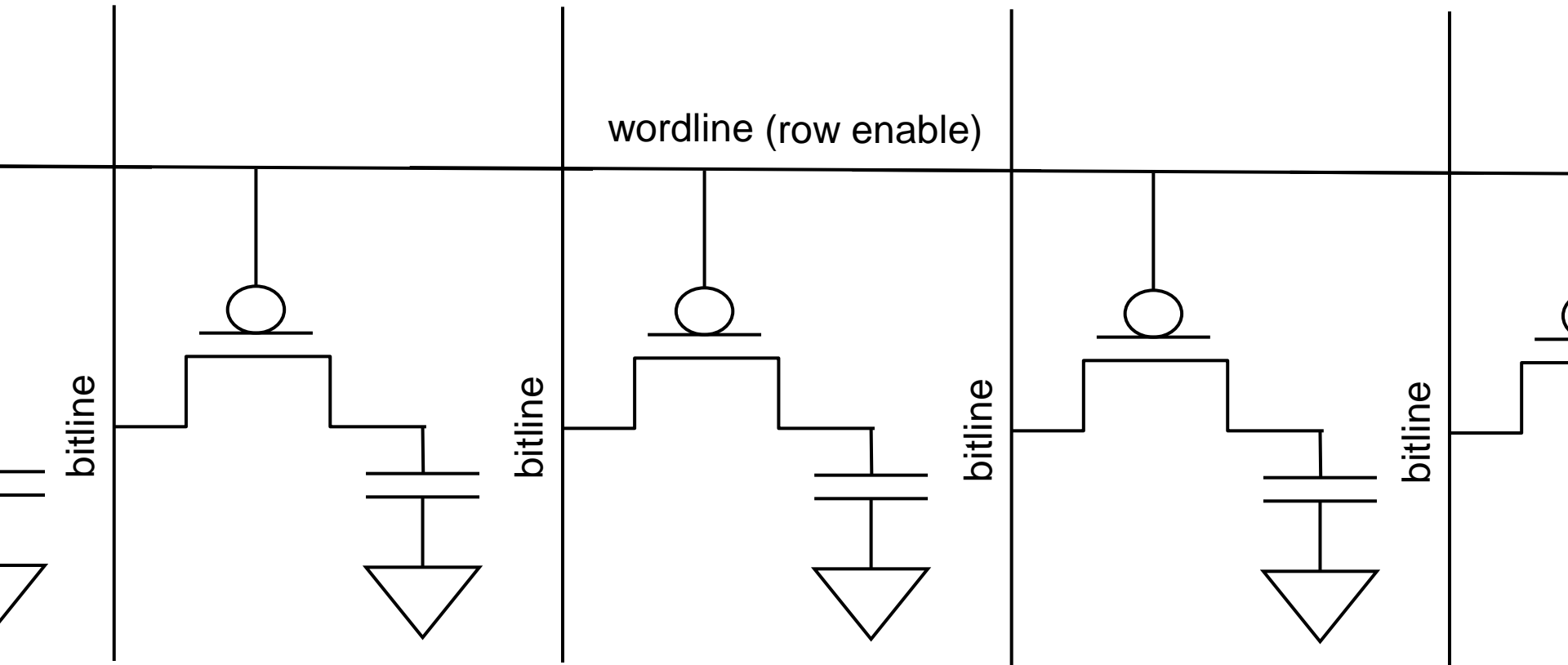
- VRT is common in modern DRAM cells that are weak
- The timescale at which VRT occurs is very large (e.g., a cell can stay in high retention time state for a day or longer) → finding minimum retention time can take very long

- Future work on retention time profiling must address these issues

Talk Agenda

- **DRAM Refresh: Background and Motivation**
- Challenges and Our Goal
- DRAM Characterization Methodology
- Foundational Results
 - Temperature Dependence
 - Retention Time Distribution
- Data Pattern Dependence: Analysis and Implications
- Variable Retention Time: Analysis and Implications
- Conclusions

A DRAM Cell

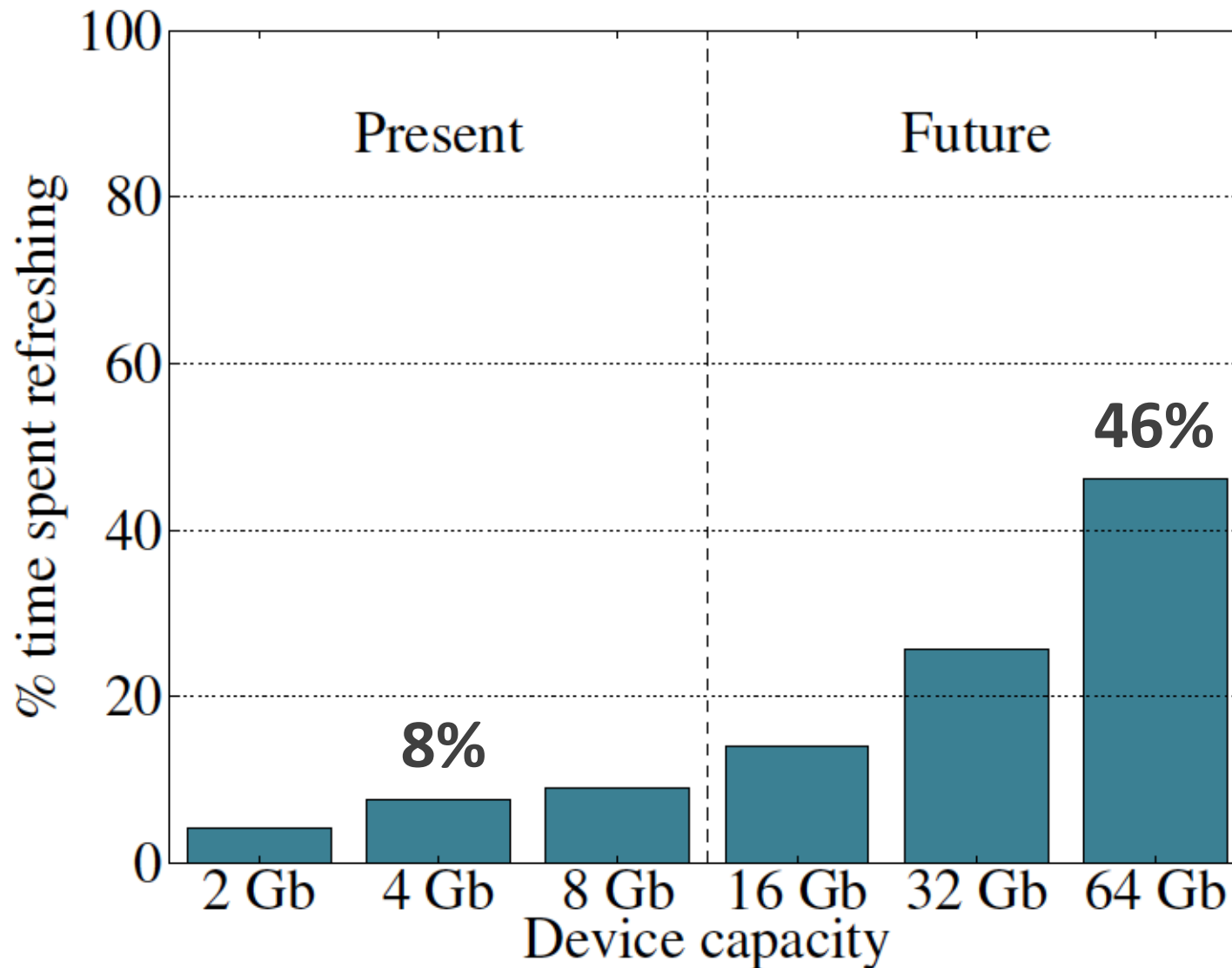


- A DRAM cell consists of a capacitor and an access transistor
- It stores data in terms of charge in the capacitor
- A DRAM chip consists of (10s of 1000s of) rows of such cells

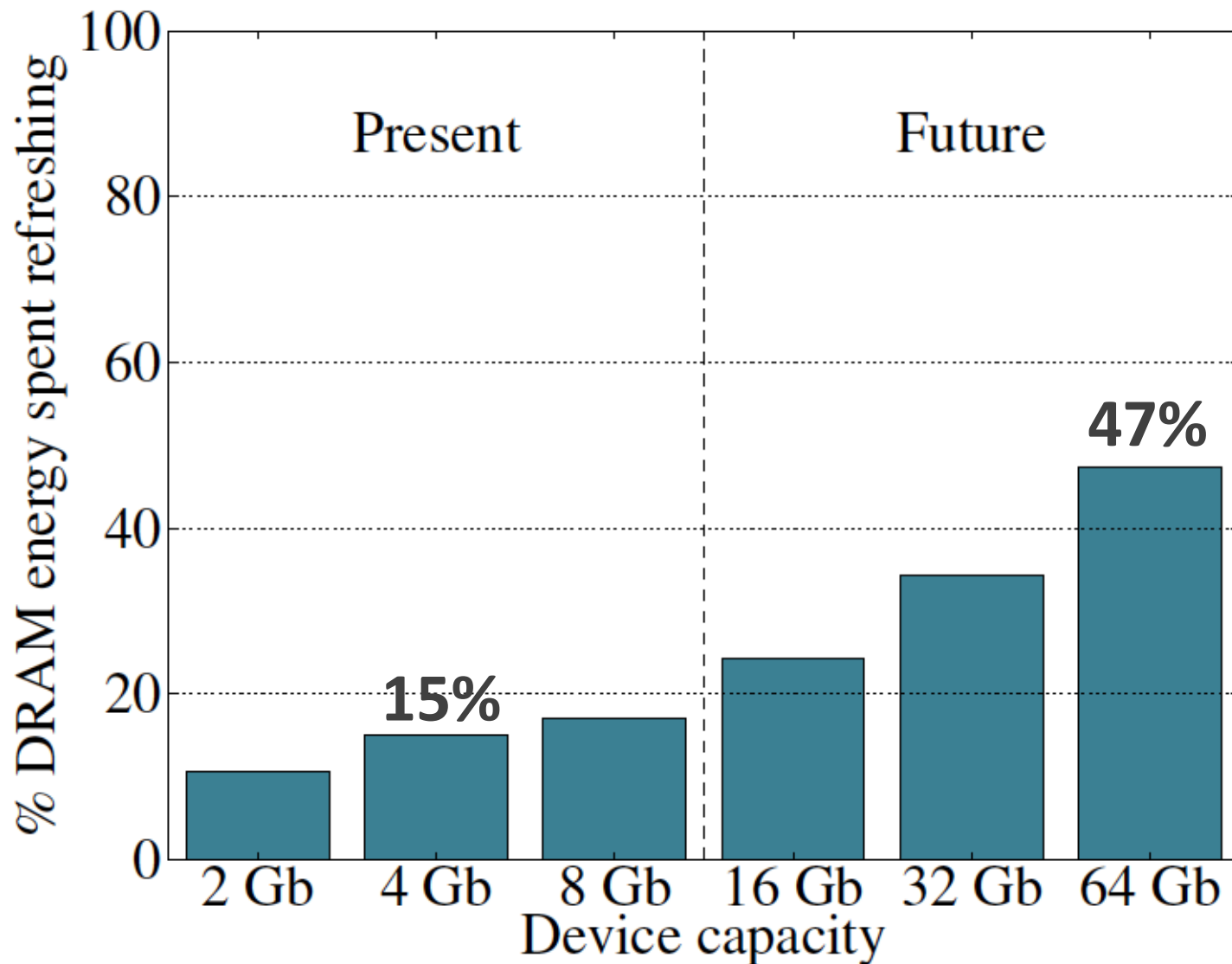
DRAM Refresh

- DRAM capacitor charge leaks over time
- Each DRAM row is periodically refreshed to restore charge
 - Activate each row every N ms
 - Typical $N = 64$ ms
- Downsides of refresh
 - **Energy consumption**: Each refresh consumes energy
 - **Performance degradation**: DRAM rank/bank unavailable while refreshed
 - **QoS/predictability impact**: (Long) pause times during refresh
 - **Refresh rate limits DRAM capacity scaling**

Refresh Overhead: Performance



Refresh Overhead: Energy

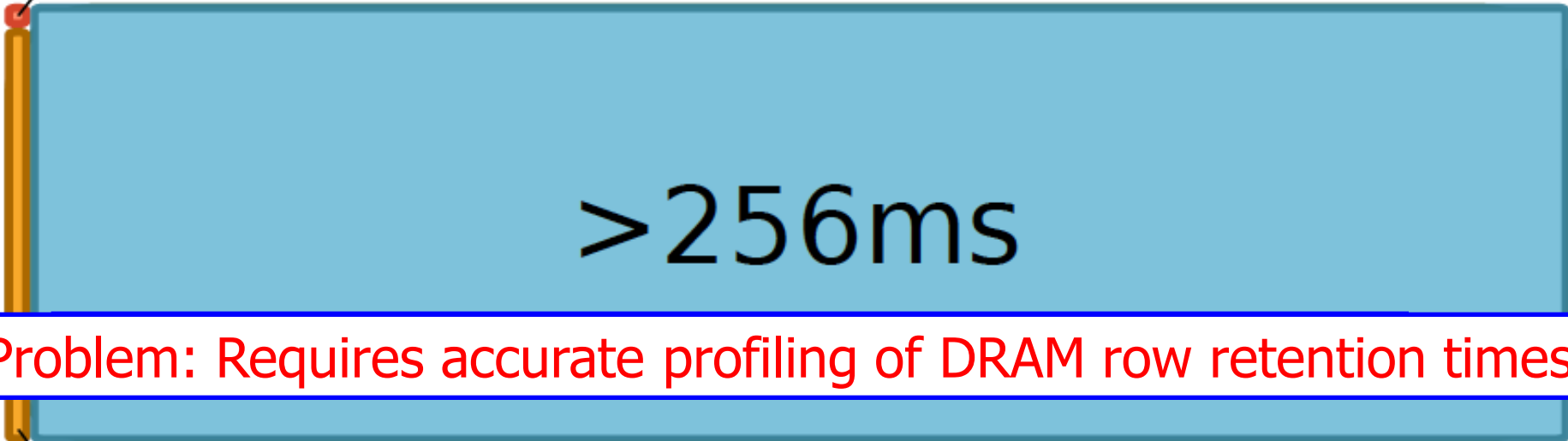


Previous Work on Reducing Refreshes

- Observed significant variation in data retention times of DRAM cells (due to manufacturing process variation)
 - **Retention time:** maximum time a cell can go without being refreshed while maintaining its stored data
- Proposed methods to take advantage of widely varying retention times among DRAM rows
 - Reduce refresh rate for rows that can retain data for longer than 64 ms, e.g., [Liu+ ISCA 2012]
 - Disable rows that have low retention times, e.g., [Venkatesan+ HPCA 2006]
- Showed large benefits in energy and performance

An Example: RAIDR [Liu+, ISCA 2012]

64-128ms



>256ms

Problem: Requires accurate profiling of DRAM row retention times

128-256ms

Can reduce refreshes by $\sim 75\%$
→ reduces energy consumption and improves performance

Motivation

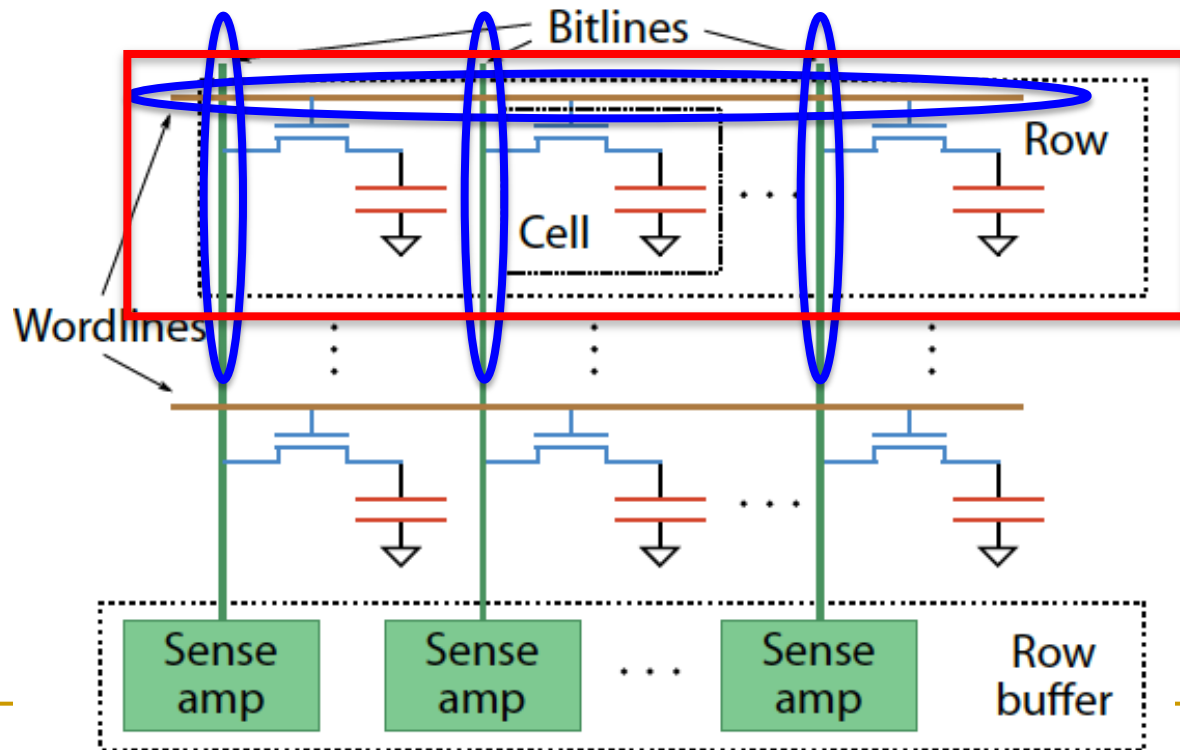
- Past works require **accurate and reliable measurement of retention time of each DRAM row**
 - To maintain data integrity while reducing refreshes
- **Assumption: worst-case retention time of each row can be determined and stays the same at a given temperature**
 - Some works propose writing all 1's and 0's to a row, and measuring the time before data corruption
- **Question:**
 - Can we reliably and accurately determine retention times of all DRAM rows?

Talk Agenda

- DRAM Refresh: Background and Motivation
- Challenges and Our Goal
- DRAM Characterization Methodology
- Foundational Results
 - Temperature Dependence
 - Retention Time Distribution
- Data Pattern Dependence: Analysis and Implications
- Variable Retention Time: Analysis and Implications
- Conclusions

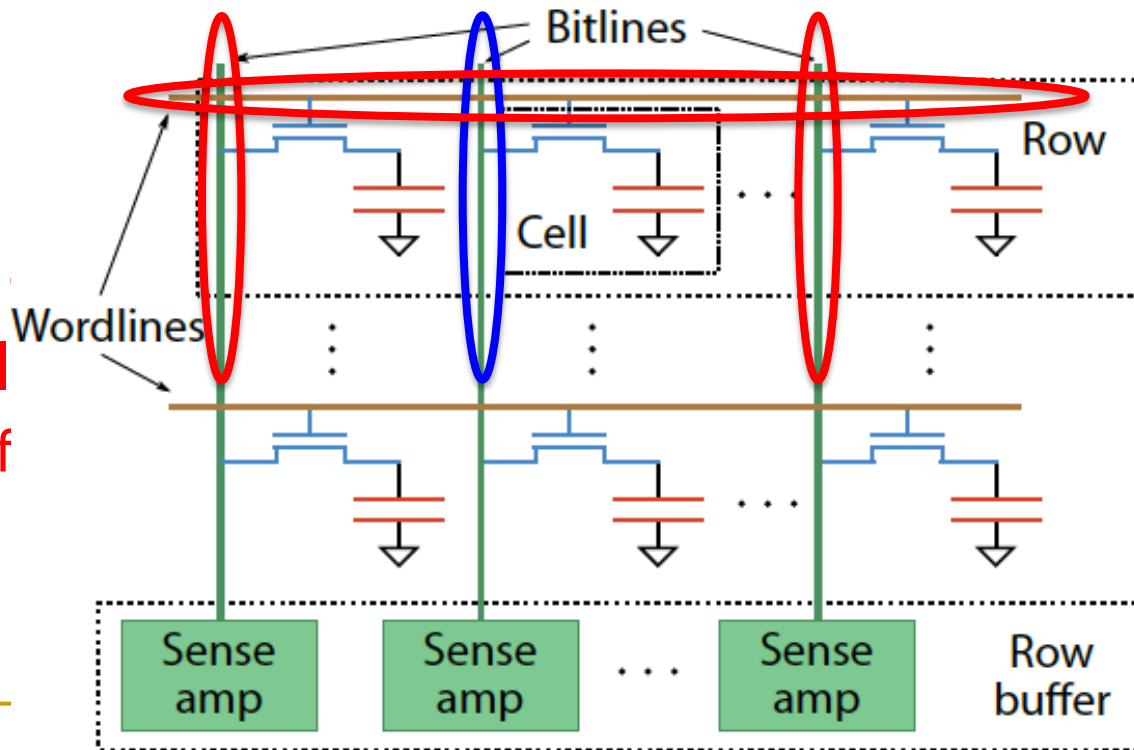
Two Challenges to Retention Time Profiling

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



Data Pattern Dependence

- Electrical noise on the bitline affects reliable sensing of a DRAM cell
- The magnitude of this noise is affected by values of nearby cells via
 - Bitline-bitline coupling → electrical coupling between adjacent bitlines
 - Bitline-wordline coupling → electrical coupling between each bitline and the activated wordline



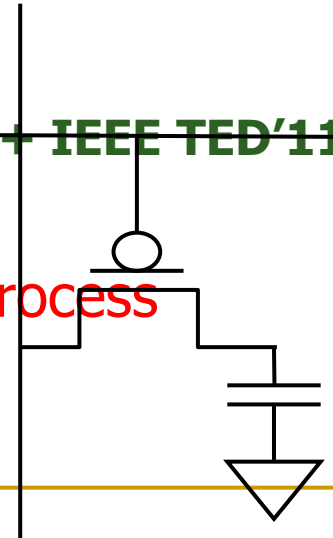
- Retention nearby cell → need to f

tored in attention time

Two Challenges to Retention Time Profiling

■ Challenge 2: Variable Retention Time (VRT)

- Retention time of a DRAM cell changes randomly over time
 - a cell alternates between multiple retention time states
- Leakage current of a cell changes sporadically due to a charge trap in the gate oxide of the DRAM cell access transistor
- When the trap becomes occupied, charge leaks more readily from the transistor's drain, leading to a short retention time
 - Called *Trap-Assisted Gate-Induced Drain Leakage*
- This process appears to be a random process [Kim+ IEEE TED'11]
- Worst-case retention time depends on a random process
 - need to find the worst case despite this



Our Goal

- Analyze the retention time behavior of DRAM cells in modern commodity DRAM devices
 - to aid the collection of accurate profile information
- Provide a comprehensive empirical investigation of two key challenges to retention time profiling
 - Data Pattern Dependence (DPD)
 - Variable Retention Time (VRT)

Talk Agenda

- DRAM Refresh: Background and Motivation
- Challenges and Our Goal
- DRAM Characterization Methodology
- Foundational Results
 - Temperature Dependence
 - Retention Time Distribution
- Data Pattern Dependence: Analysis and Implications
- Variable Retention Time: Analysis and Implications
- Conclusions

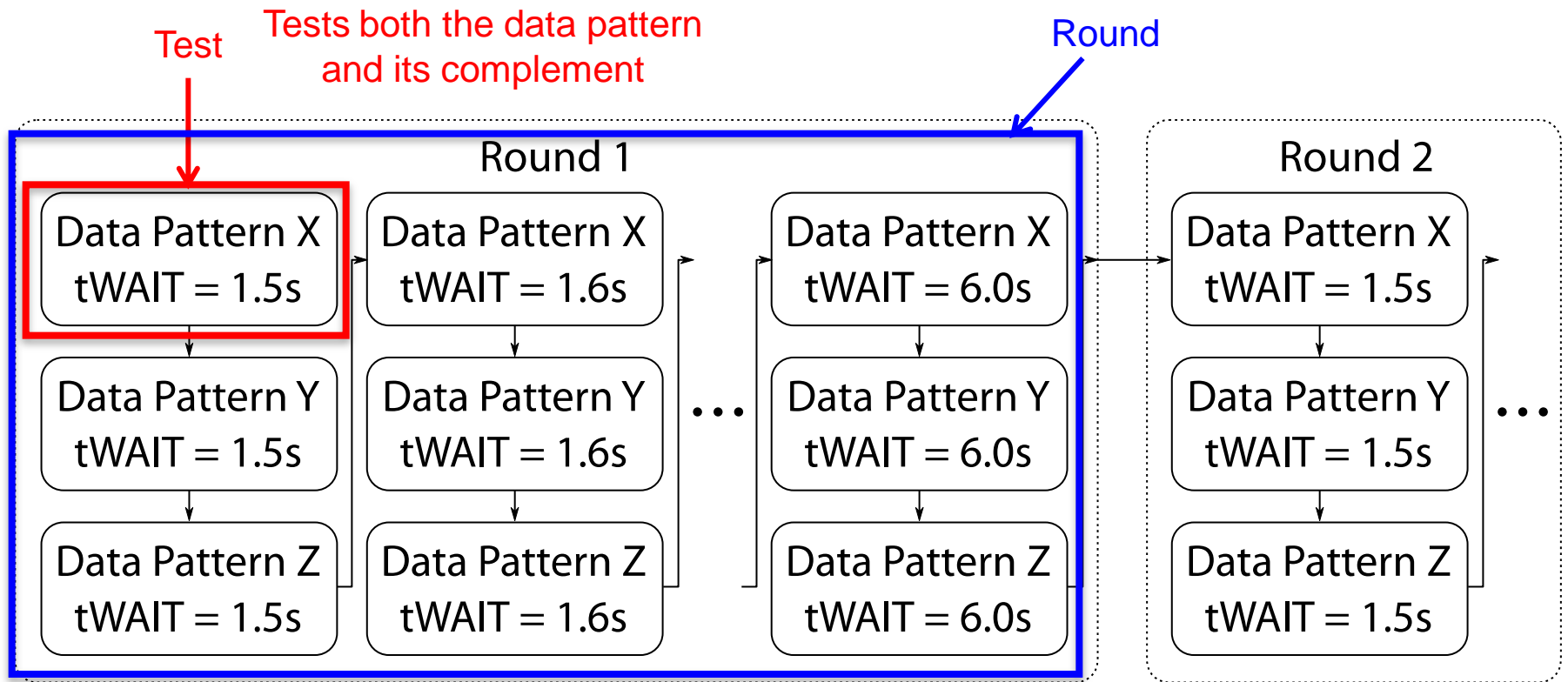
DRAM Testing Platform and Method

- **Test platform:** Developed a DDR3 DRAM testing platform using the Xilinx ML605 FPGA development board
 - Temperature controlled
- **Tested DRAM chips:** 248 commodity DRAM chips from five manufacturers (A,B,C,D,E)
- Seven families based on equal capacity per device:
 - A 1Gb, A 2Gb
 - B 2Gb
 - C 2Gb
 - D 1Gb, D 2Gb
 - E 2Gb

Experiment Design

- Each module tested for multiple ***rounds*** of ***tests***.
- Each test searches for the set of cells with a retention time less than a threshold value for a particular data pattern
- High-level structure of a test:
 - Write data pattern to rows in a DRAM bank
 - Prevent refresh for a period of time t_{WAIT} , leave DRAM idle
 - Read stored data pattern, compare to written pattern and record corrupt cells as those with retention time $< t_{WAIT}$
- Test details and important issues to pay attention to are discussed in paper

Experiment Structure



Experiment Parameters

- Most tests conducted at 45 degrees Celsius
- No cells observed to have a retention time less than 1.5 second at 45°C
- Tested *tWAIT* in increments of 128ms from 1.5 to 6.1 seconds

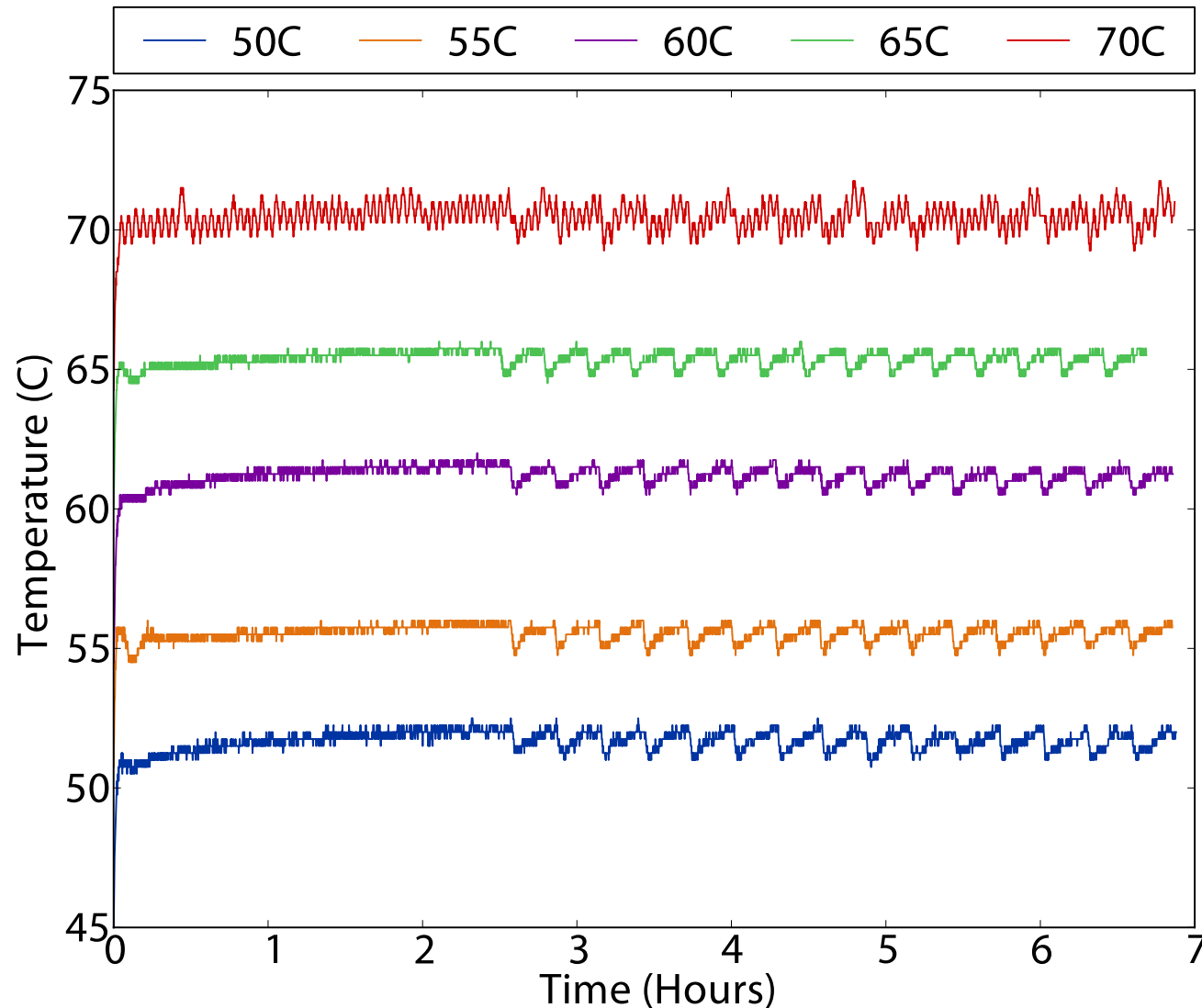
Tested Data Patterns

- **All 0s/1s:** Value 0/1 is written to all bits **Fixed patterns**
 - Previous work suggested this is sufficient
- **Checkerboard:** Consecutive bits alternate between 0 and 1
 - Coupling noise increases with voltage difference between the neighboring bitlines → May induce worst case data pattern (if adjacent bits mapped to adjacent cells)
- **Walk:** Attempts to ensure a single cell storing 1 is surrounded by cells storing 0
 - This may lead to even worse coupling noise and retention time due to coupling between *nearby* bitlines [**Li+ IEEE TCSI 2011**]
 - Walk pattern is permuted in each round to exercise different cells
- **Random:** Randomly generated data is written to each row
 - A new set of random data is generated for each round

Talk Agenda

- DRAM Refresh: Background and Motivation
- Challenges and Our Goal
- DRAM Characterization Methodology
- **Foundational Results**
 - Temperature Dependence
 - Retention Time Distribution
- Data Pattern Dependence: Analysis and Implications
- Variable Retention Time: Analysis and Implications
- Conclusions

Temperature Stability



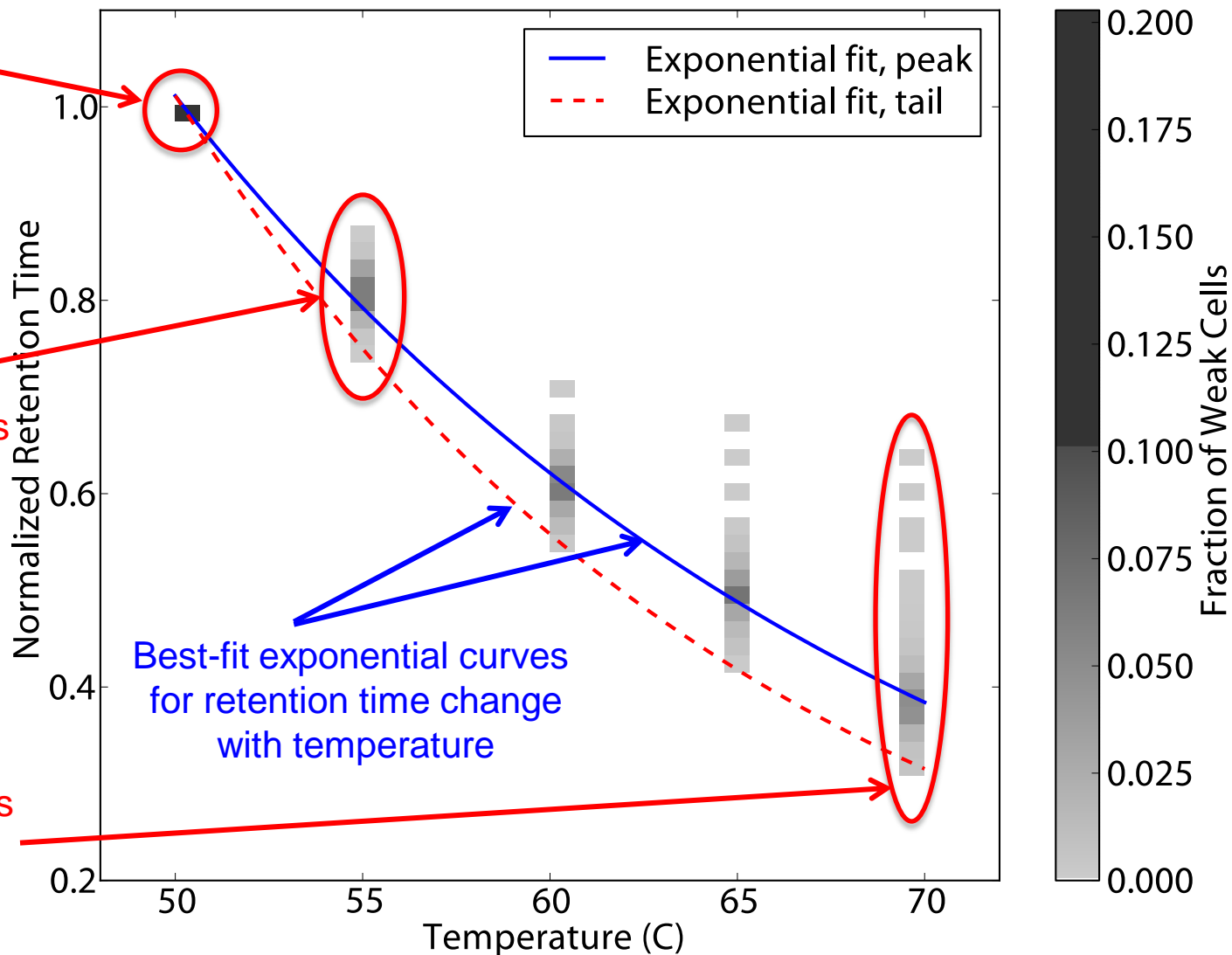
Tested chips at five different stable temperatures

Dependence of Retention Time on Temperature

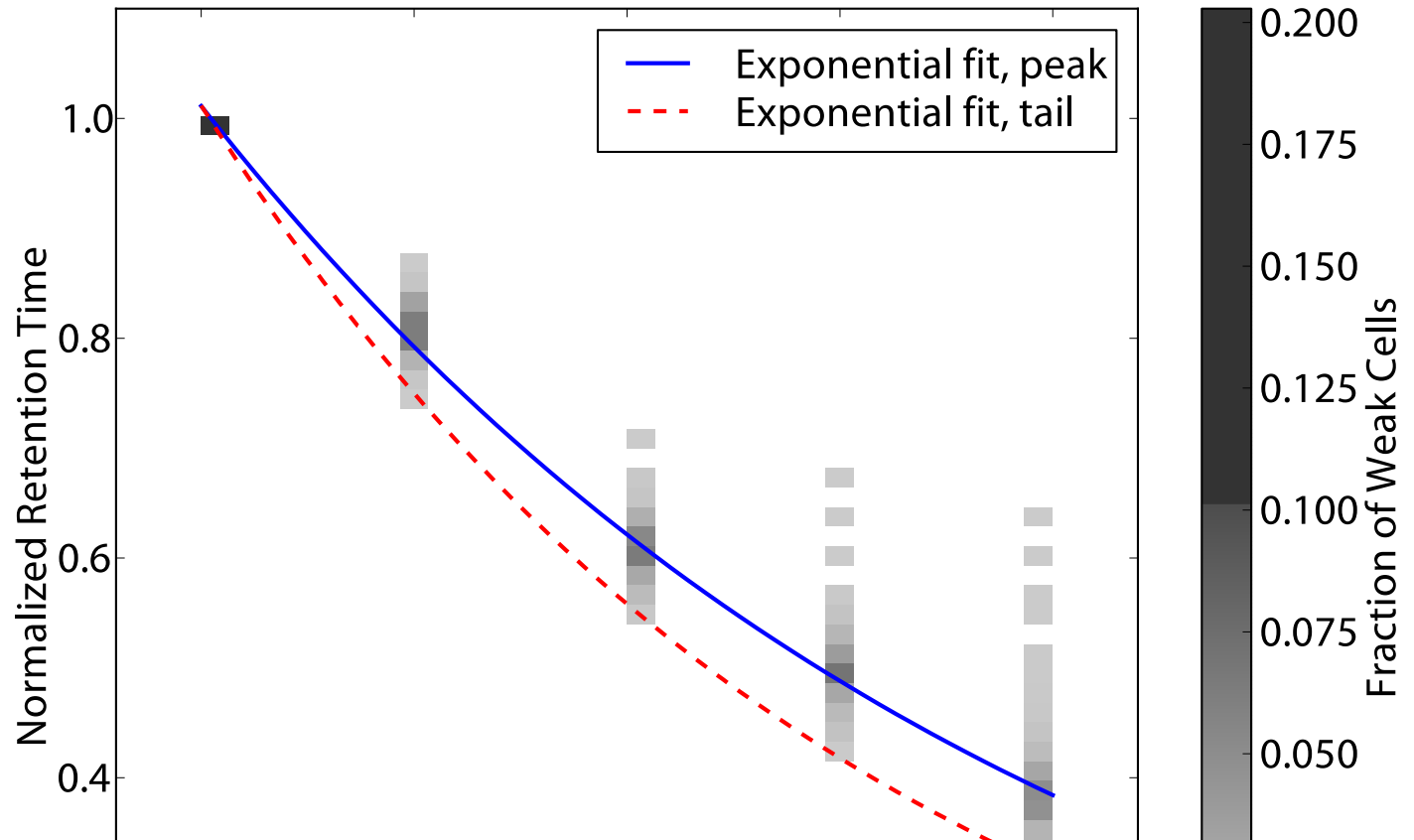
Fraction of cells that exhibited retention time failure at any t_{WAIT} for any data pattern at 50°C

Normalized retention times of the same cells at 55°C

Normalized retention times of the same cells At 70°C



Dependence of Retention Time on Temperature

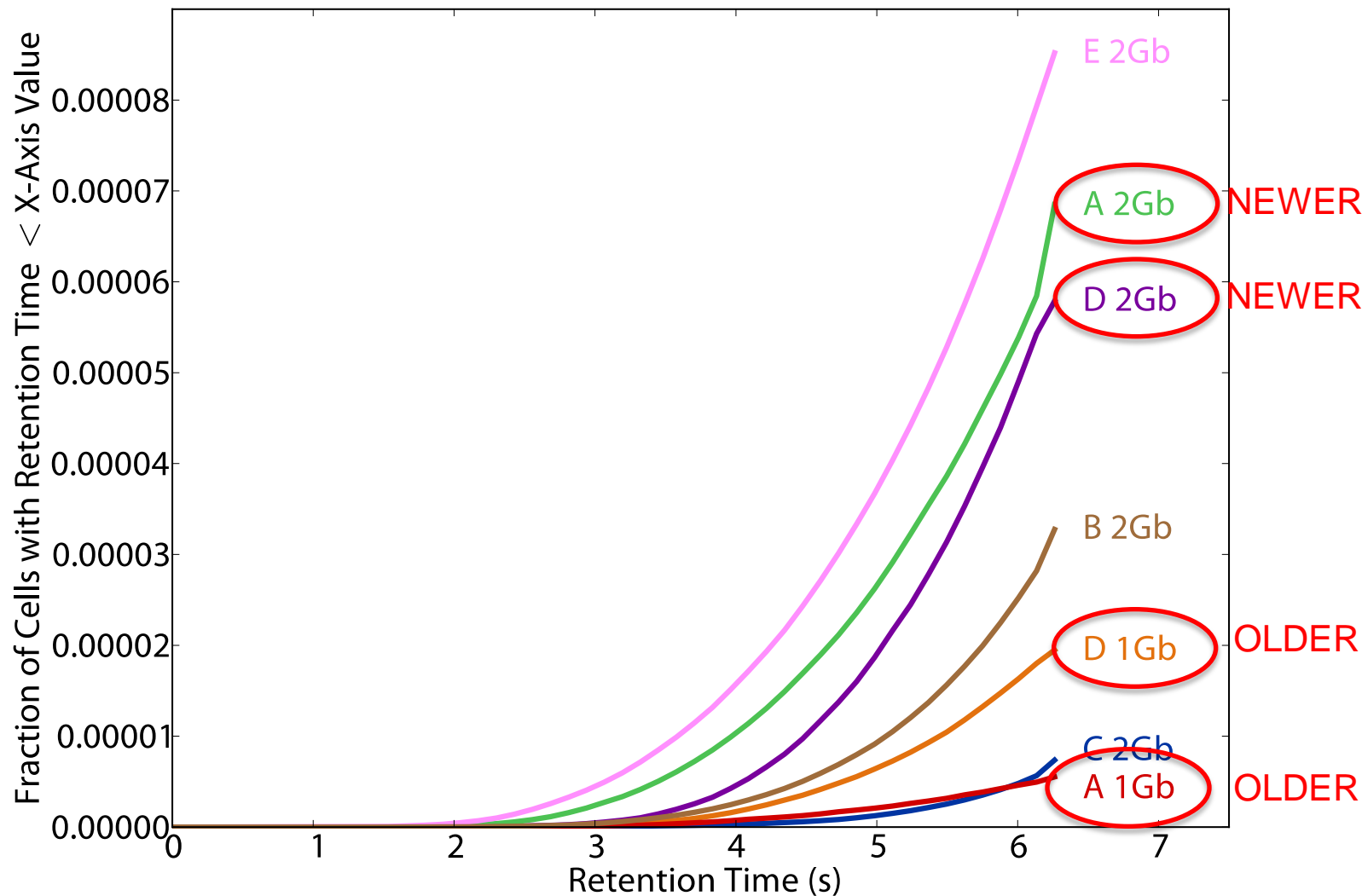


Relationship between retention time and temperature is consistently bounded (predictable) within a device

Every 10⁰C temperature increase

→ 46.5% reduction in retention time in the worst case

Retention Time Distribution



**Newer device families have more weak cells than older ones
Likely a result of technology scaling**

Talk Agenda

- DRAM Refresh: Background and Motivation
- Challenges and Our Goal
- DRAM Characterization Methodology
- Foundational Results
 - Temperature Dependence
 - Retention Time Distribution
- Data Pattern Dependence: Analysis and Implications
- Variable Retention Time: Analysis and Implications
- Conclusions

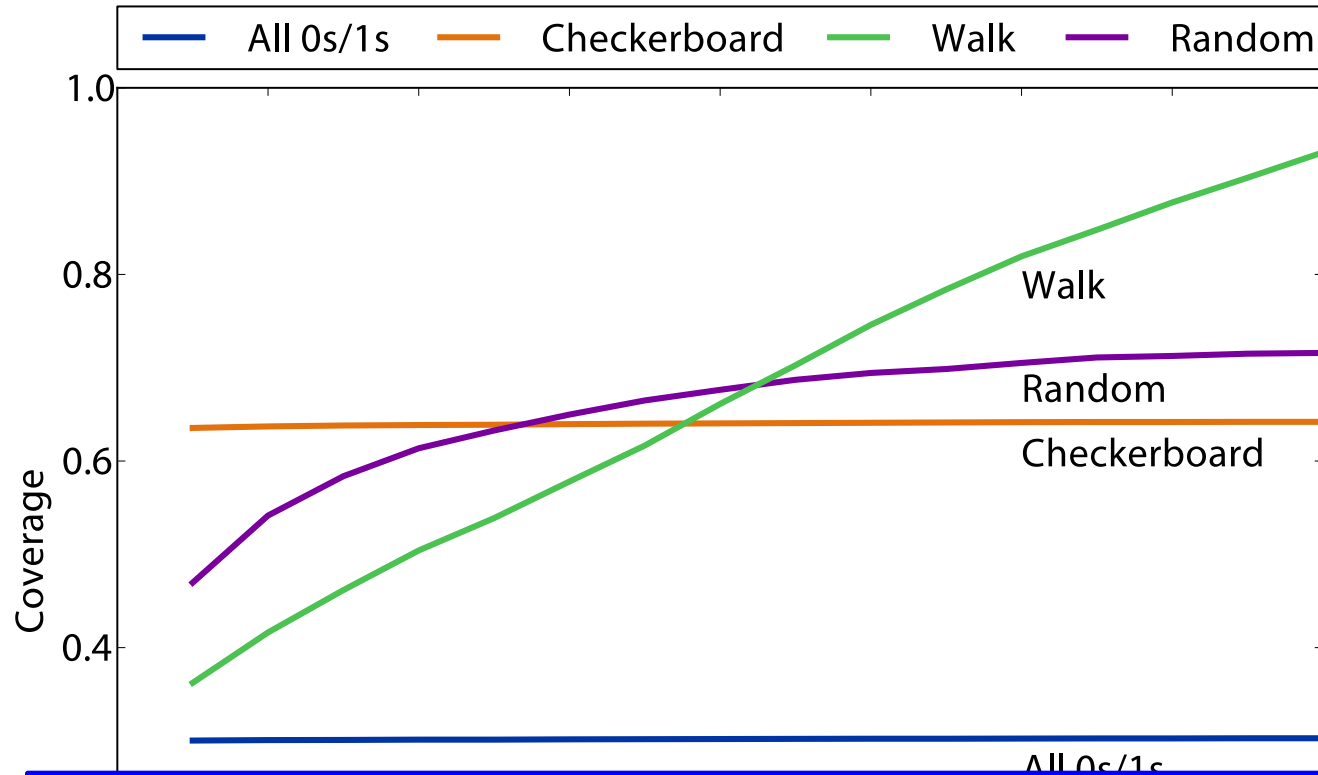
Some Terminology

- **Failure population of cells with Retention Time X:** The set of all cells that exhibit retention failure in any test *with any data pattern* at that retention time (t_{WAIT})
- **Retention Failure Coverage of a Data Pattern DP:** Fraction of cells with retention time X that exhibit retention failure with that *particular* data pattern DP
- If retention times are not dependent on data pattern stored in cells, we would expect
 - Coverage of any data pattern to be 100%
 - In other words, if one data pattern causes a retention failure, any other data pattern also would

Recall the Tested Data Patterns

- **All 0s/1s**: Value 0/1 is written to all bits **Fixed patterns**
- **Checkerboard**: Consecutive bits alternate between 0 and 1
- **Walk**: Attempts to ensure a single cell storing 1 is surrounded by cells storing 0
- **Random**: Randomly generated data is written to each row

Retention Failure Coverage of Data Patterns



A 2Gb chip family

6.1s retention time

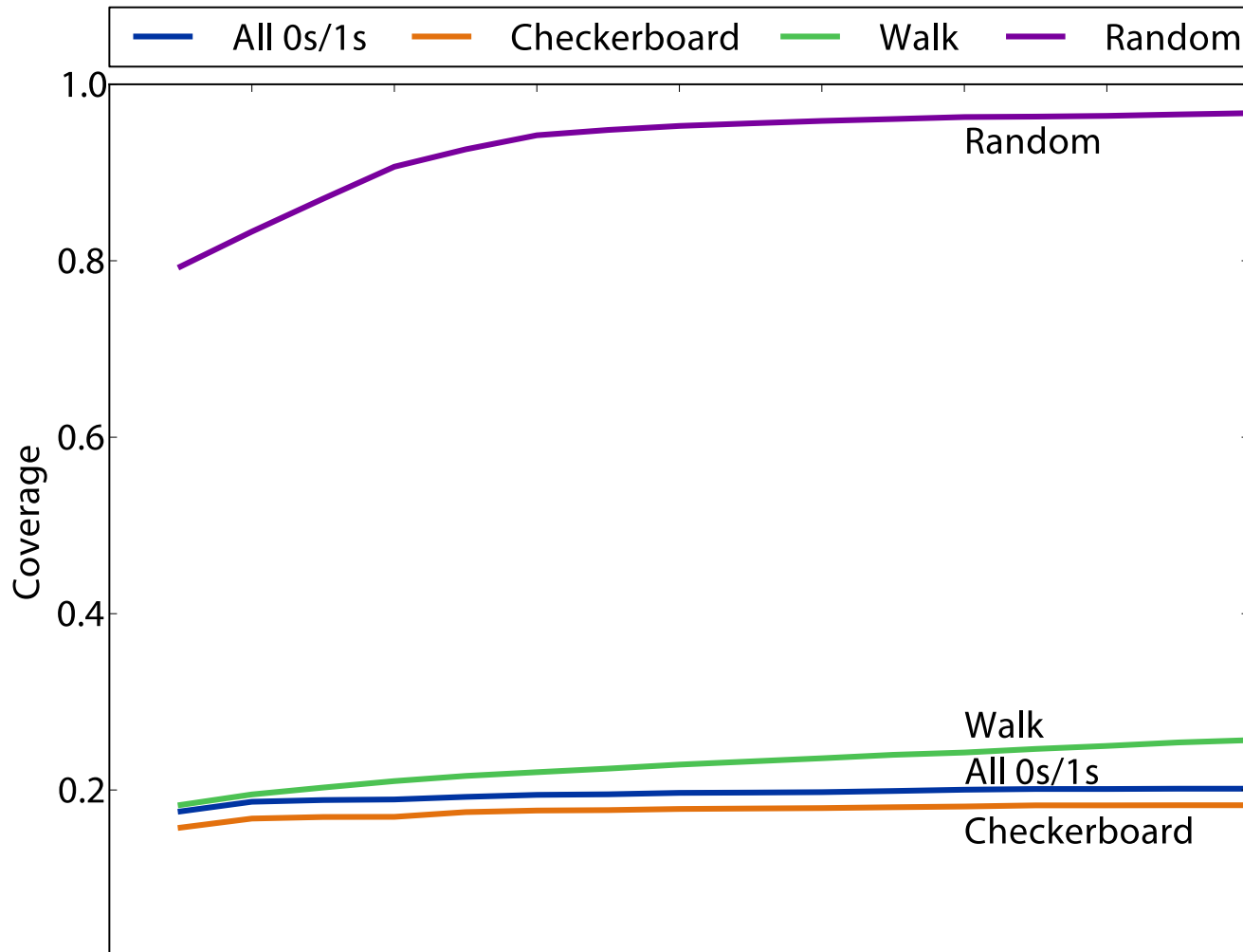
**Different data patterns have widely different coverage:
Data pattern dependence exists and is severe**

Coverage of fixed patterns is low: ~30% for *All 0s/1s*

***Walk* is the most effective data pattern for this device**

No data pattern achieves 100% coverage

Retention Failure Coverage of Data Patterns



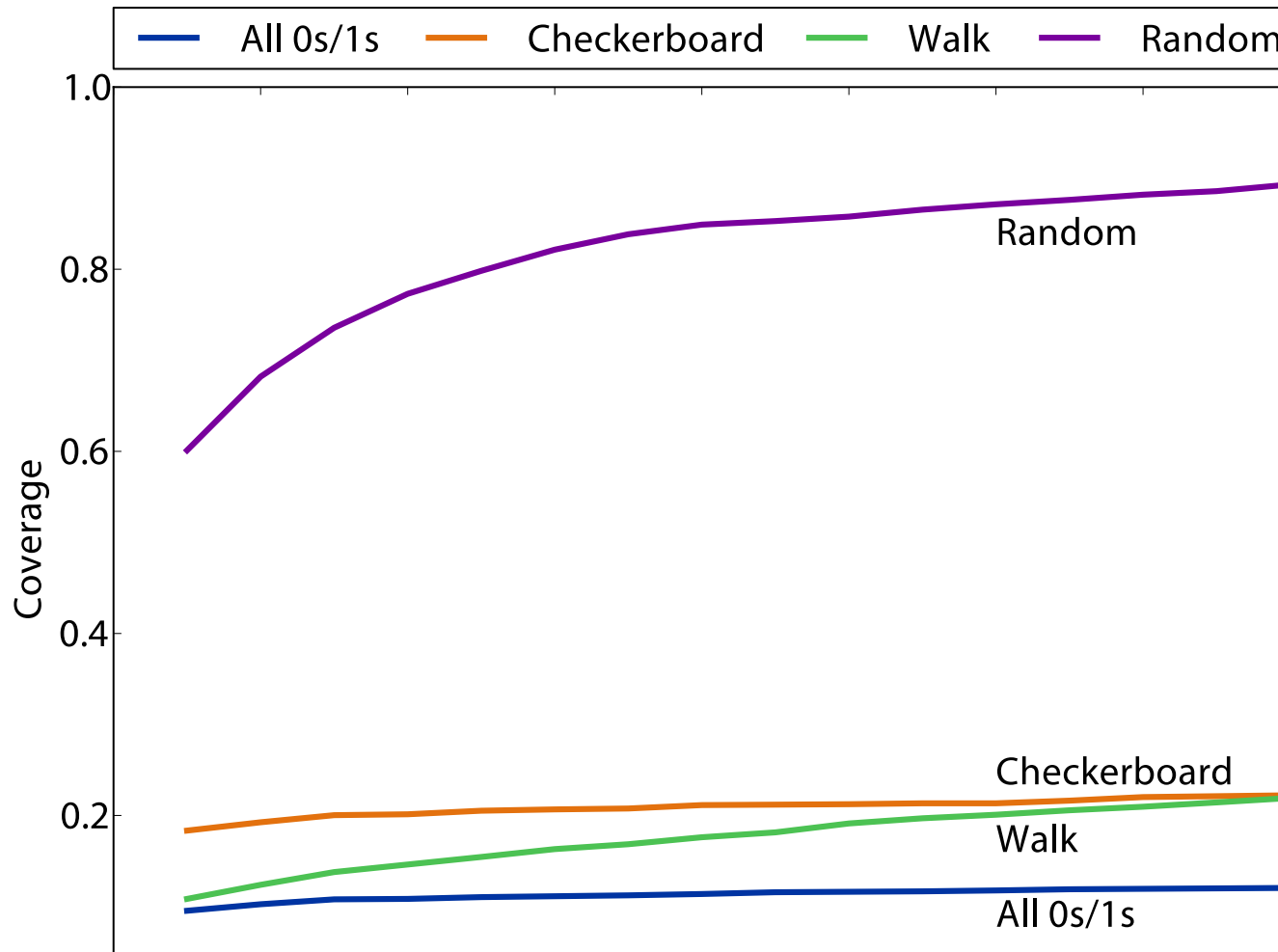
B 2Gb chip family

6.1s retention time

Random is the most effective data pattern for this device

No data pattern achieves 100% coverage

Retention Failure Coverage of Data Patterns



C 2Gb chip family

6.1s retention time

Random is the most effective data pattern for this device

No data pattern achieves 100% coverage

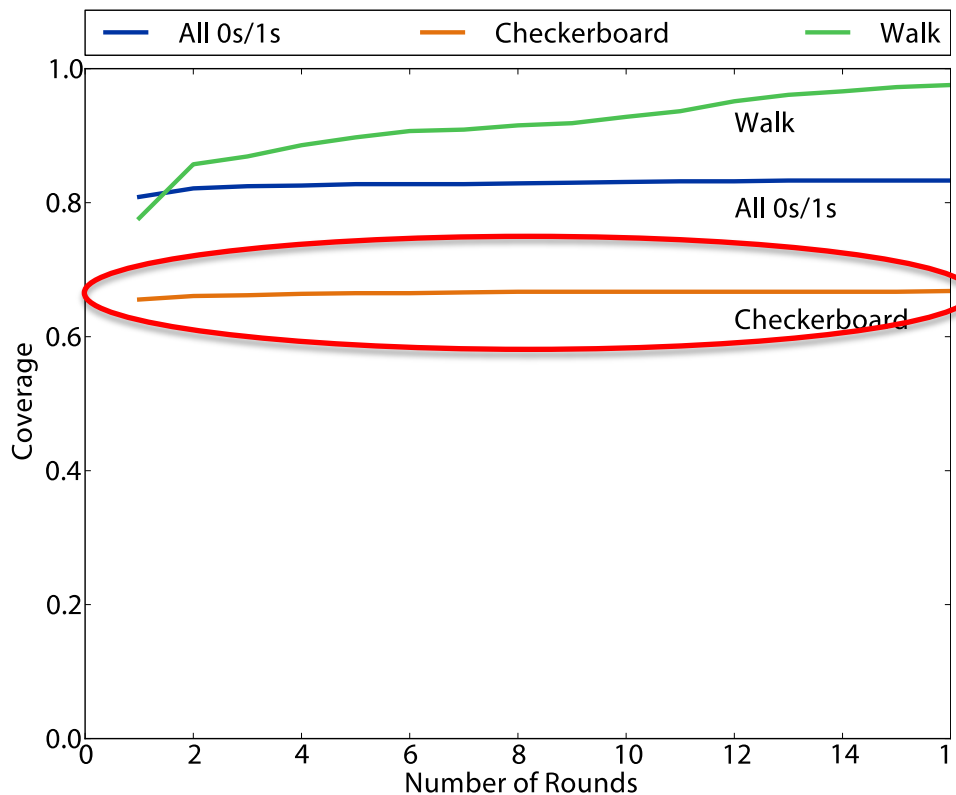
Data Pattern Dependence: Observations (I)

- A cell's retention time is heavily influenced by data pattern stored in other cells
 - Pattern affects the coupling noise, which affects cell leakage
- No tested data pattern exercises the worst case retention time for all cells (no pattern has 100% coverage)
 - No pattern is able to induce the worst-case coupling noise for every cell
 - Problem: **Underlying DRAM circuit organization is *not* known to the memory controller** → very hard to construct a pattern that exercises the worst-case cell leakage
 - Opaque mapping of addresses to physical DRAM geometry
 - Internal remapping of addresses within DRAM to tolerate faults
 - Second order coupling effects are very hard to determine

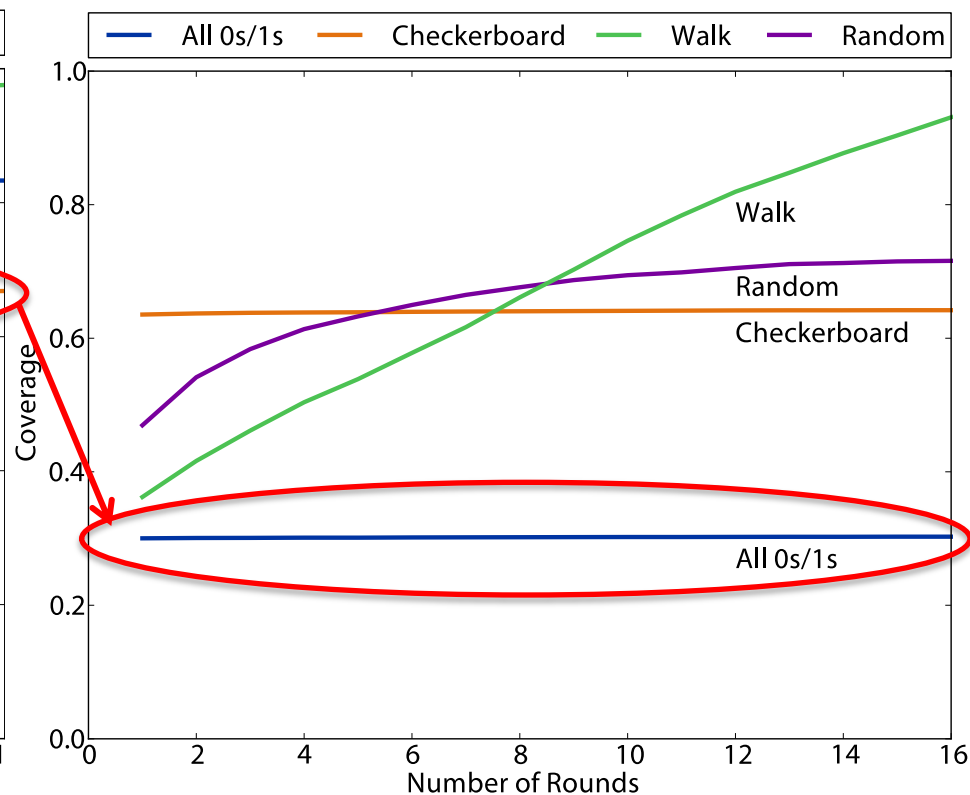
Data Pattern Dependence: Observations (II)

- Fixed, simple data patterns have low coverage
 - They do not exercise the worst-case coupling noise
- The effectiveness of each data pattern varies significantly between DRAM devices (of the same or different vendors)
 - Underlying DRAM circuit organization likely differs between different devices → patterns leading to worst coupling are different in different devices
- Technology scaling appears to increase the impact of data pattern dependence
 - Scaling reduces the physical distance between circuit elements, increasing the magnitude of coupling effects

Effect of Technology Scaling on DPD



A 1Gb chip family



A 2Gb chip family

The lowest-coverage data pattern achieves much lower coverage for the smaller technology node

DPD: Implications on Profiling Mechanisms

- Any retention time profiling mechanism must handle data pattern dependence of retention time
- Intuitive approach: Identify the data pattern that induces the worst-case retention time for a particular cell or device
- Problem 1: Very hard to know at the memory controller which bits actually interfere with each other due to
 - Opaque mapping of addresses to physical DRAM geometry → logically consecutive bits may not be physically consecutive
 - Remapping of faulty bitlines/wordlines to redundant ones internally within DRAM
- Problem 2: Worst-case coupling noise is affected by non-obvious second order bitline coupling effects

DPD: Suggestions (for Future Work)

- A mechanism for identifying worst-case data pattern(s) likely requires support from DRAM device
 - DRAM manufacturers might be in a better position to do this
 - But, the ability of the manufacturer to identify and expose the entire retention time profile is limited due to VRT
- An alternative approach: Use random data patterns to increase coverage as much as possible; handle incorrect retention time estimates with ECC
 - Need to keep profiling time in check
 - Need to keep ECC overhead in check

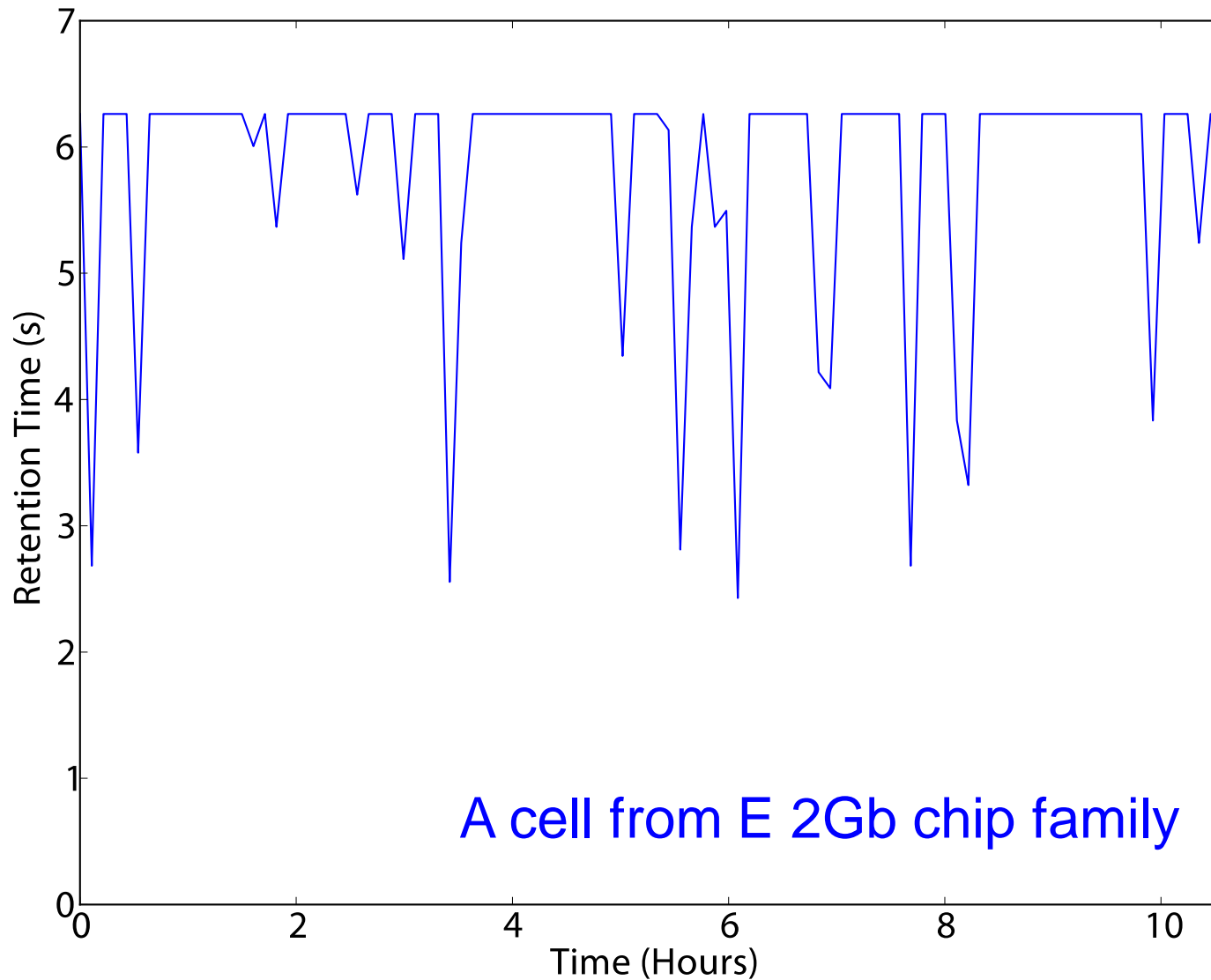
Talk Agenda

- DRAM Refresh: Background and Motivation
- Challenges and Our Goal
- DRAM Characterization Methodology
- Foundational Results
 - Temperature Dependence
 - Retention Time Distribution
- Data Pattern Dependence: Analysis and Implications
- Variable Retention Time: Analysis and Implications
- Conclusions

Variable Retention Time

- Retention time of a cell can vary over time
- A cell can randomly switch between multiple leakage current states due to *Trap-Assisted Gate-Induced Drain Leakage*, which appears to be a random process
[Yaney+ IEDM 1987, Restle+ IEDM 1992]

An Example VRT Cell



VRT: Questions and Methodology

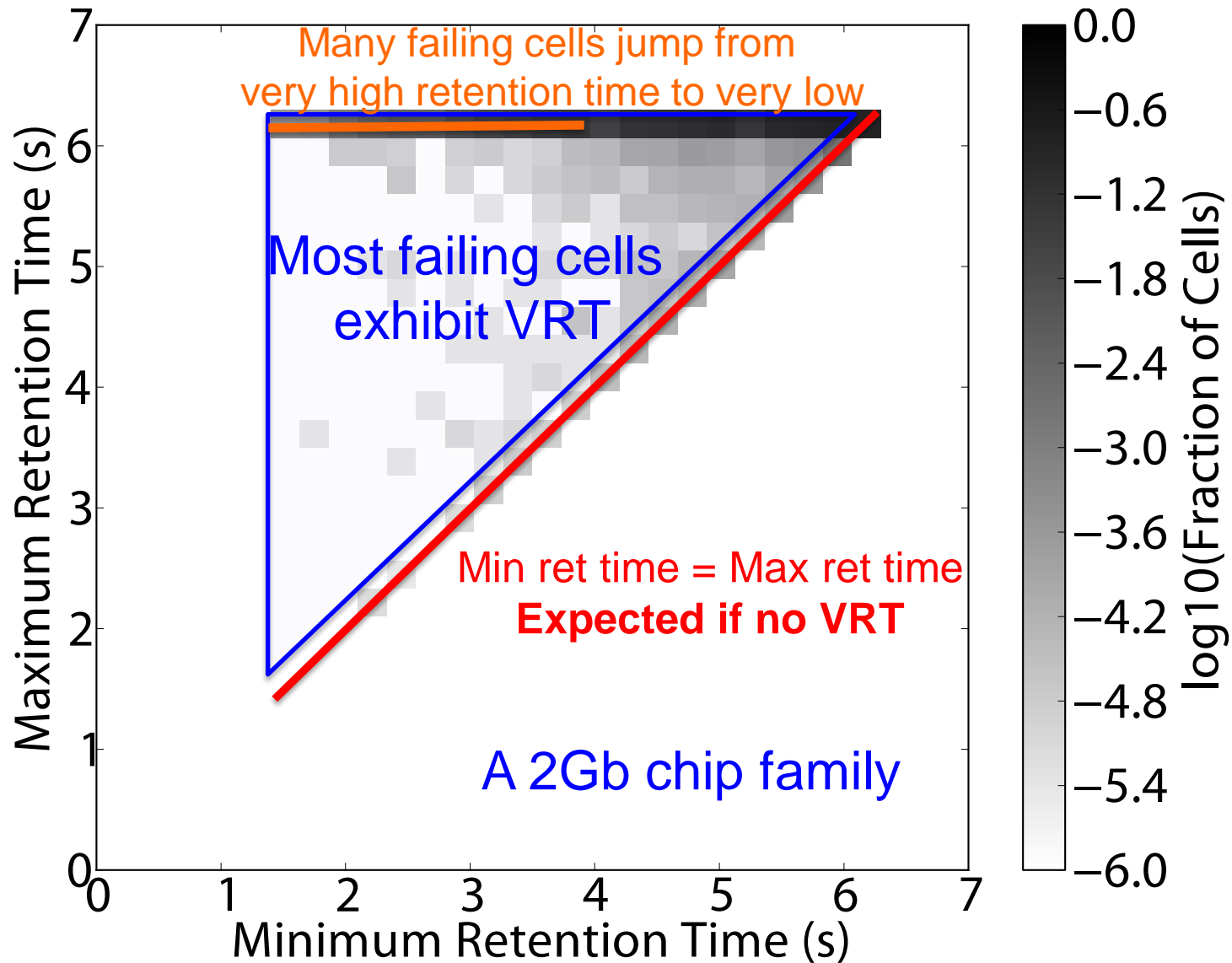
■ Key Questions

- ❑ How prevalent is VRT in modern DRAM devices?
- ❑ What is the timescale of observation of the lowest retention time state?
- ❑ What are the implications on retention time profiling?

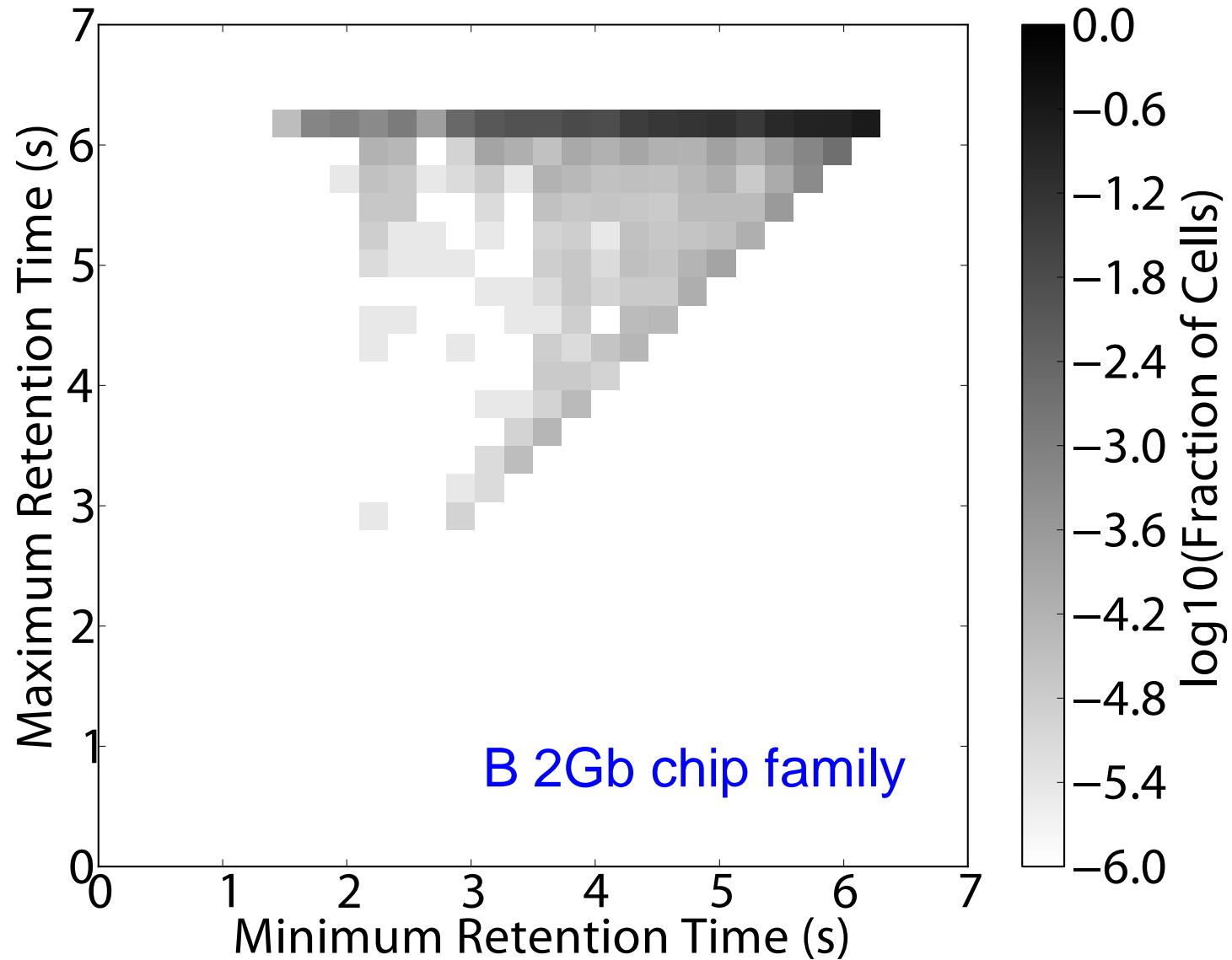
■ Test Methodology

- ❑ Each device was tested for at least 1024 rounds over 24 hours
- ❑ Temperature fixed at 45°C
- ❑ Data pattern used is the most effective data pattern for each device
- ❑ For each cell that fails at any retention time, we record the minimum and the maximum retention time observed

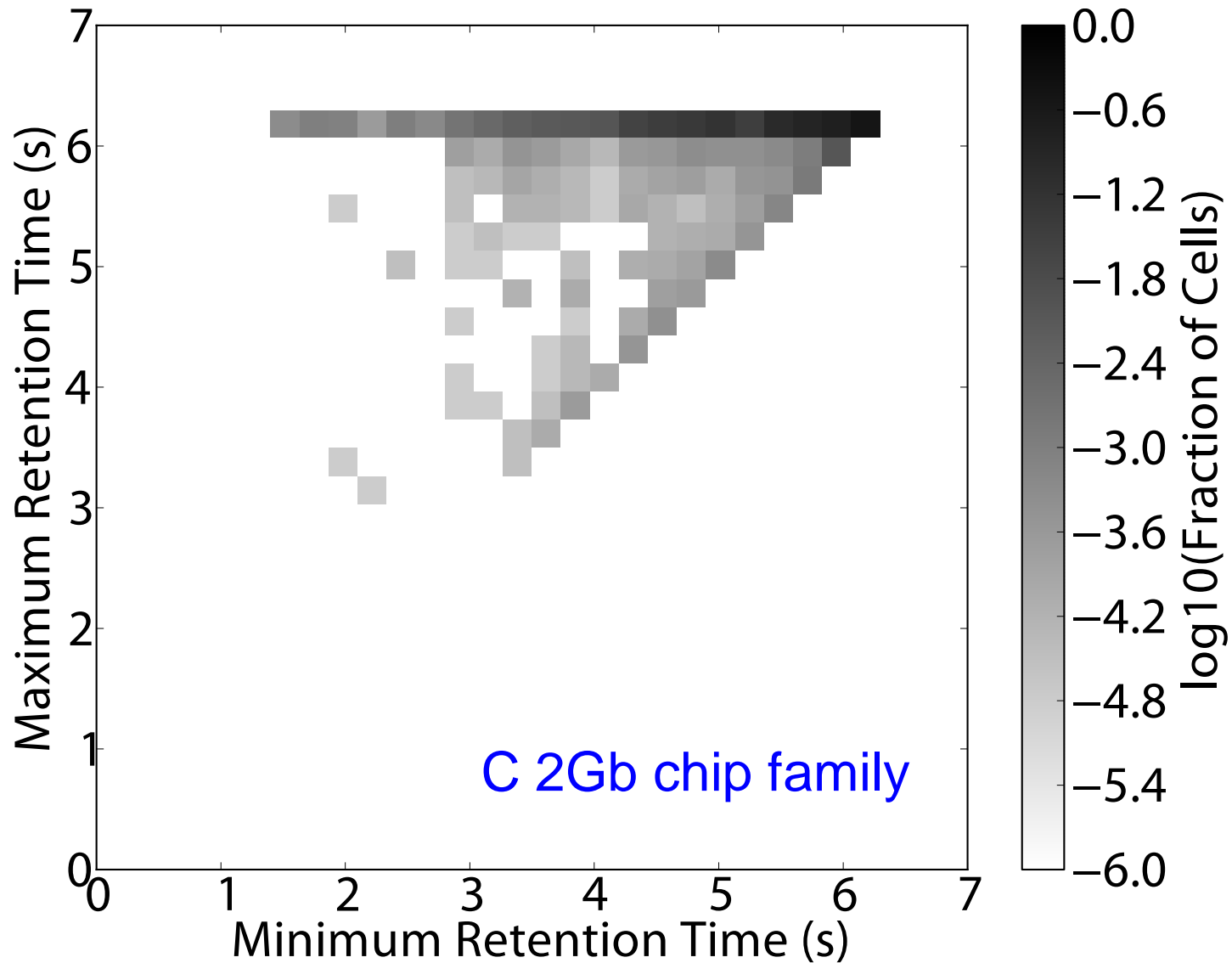
Variable Retention Time



Variable Retention Time



Variable Retention Time



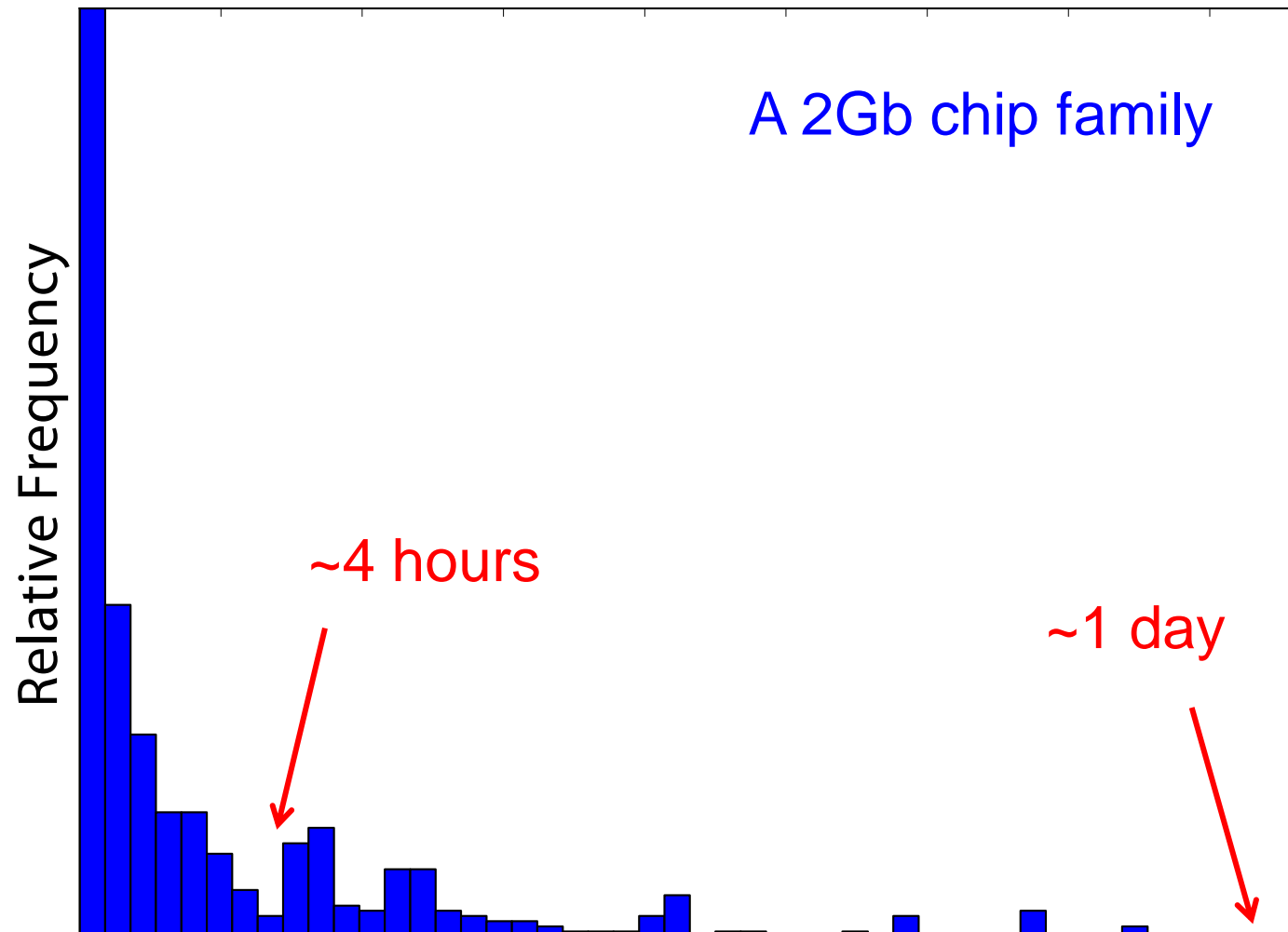
VRT: Observations So Far

- VRT is common among weak cells (i.e., those cells that experience low retention times)
- VRT can result in significant retention time changes
 - Difference between minimum and maximum retention times of a cell can be more than 4x, and may not be bounded
 - **Implication:** Finding a retention time for a cell and using a guardband to ensure minimum retention time is “covered” requires a large guardband or may not work
- Retention time profiling mechanisms must identify lowest retention time in the presence of VRT
 - **Question:** How long to profile a cell to find its lowest retention time state?

Time Between Retention Time State Changes

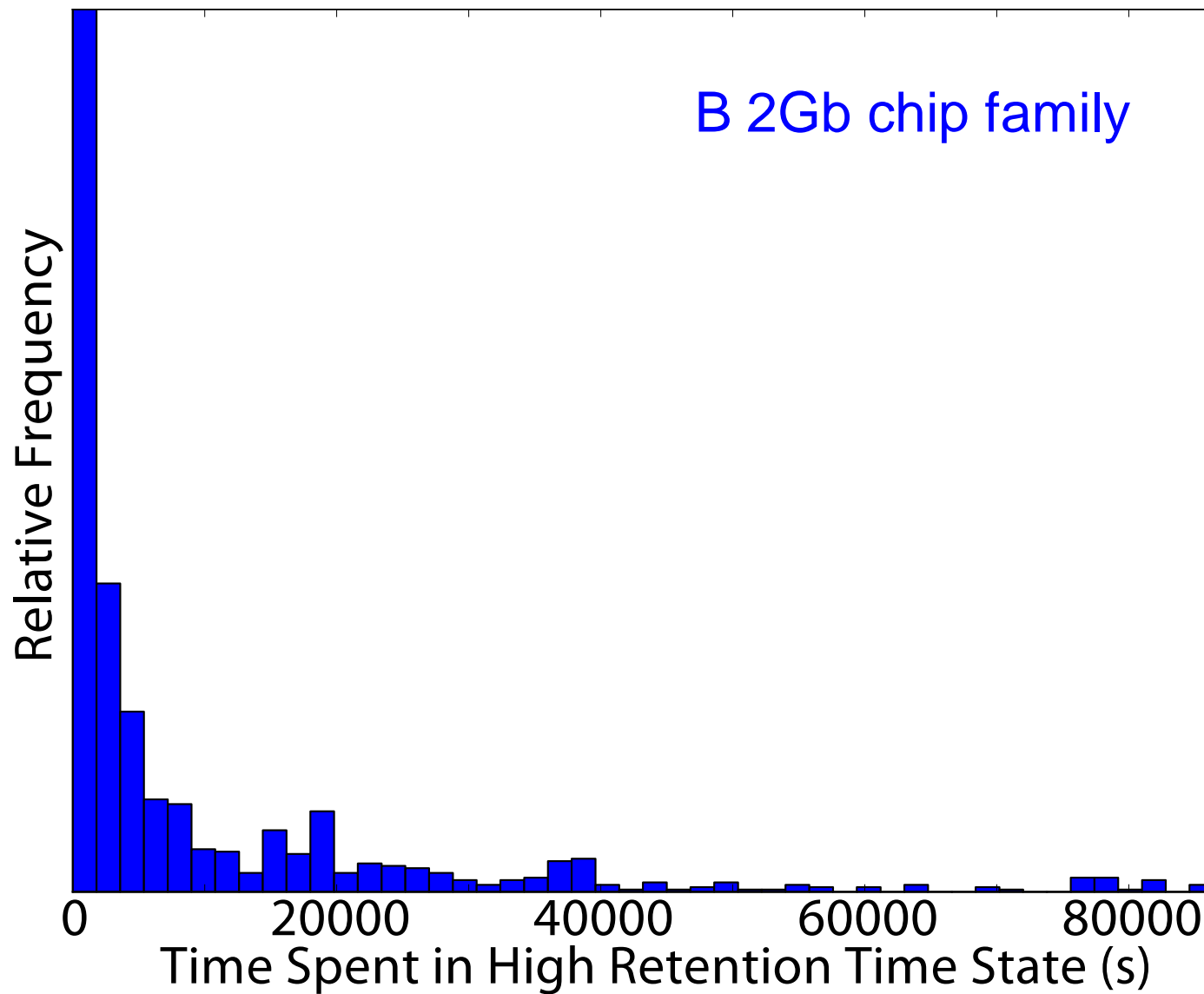
- How much time does a cell spend in a high retention state before switching to the minimum observed retention time state?

Time Spent in High Retention Time State

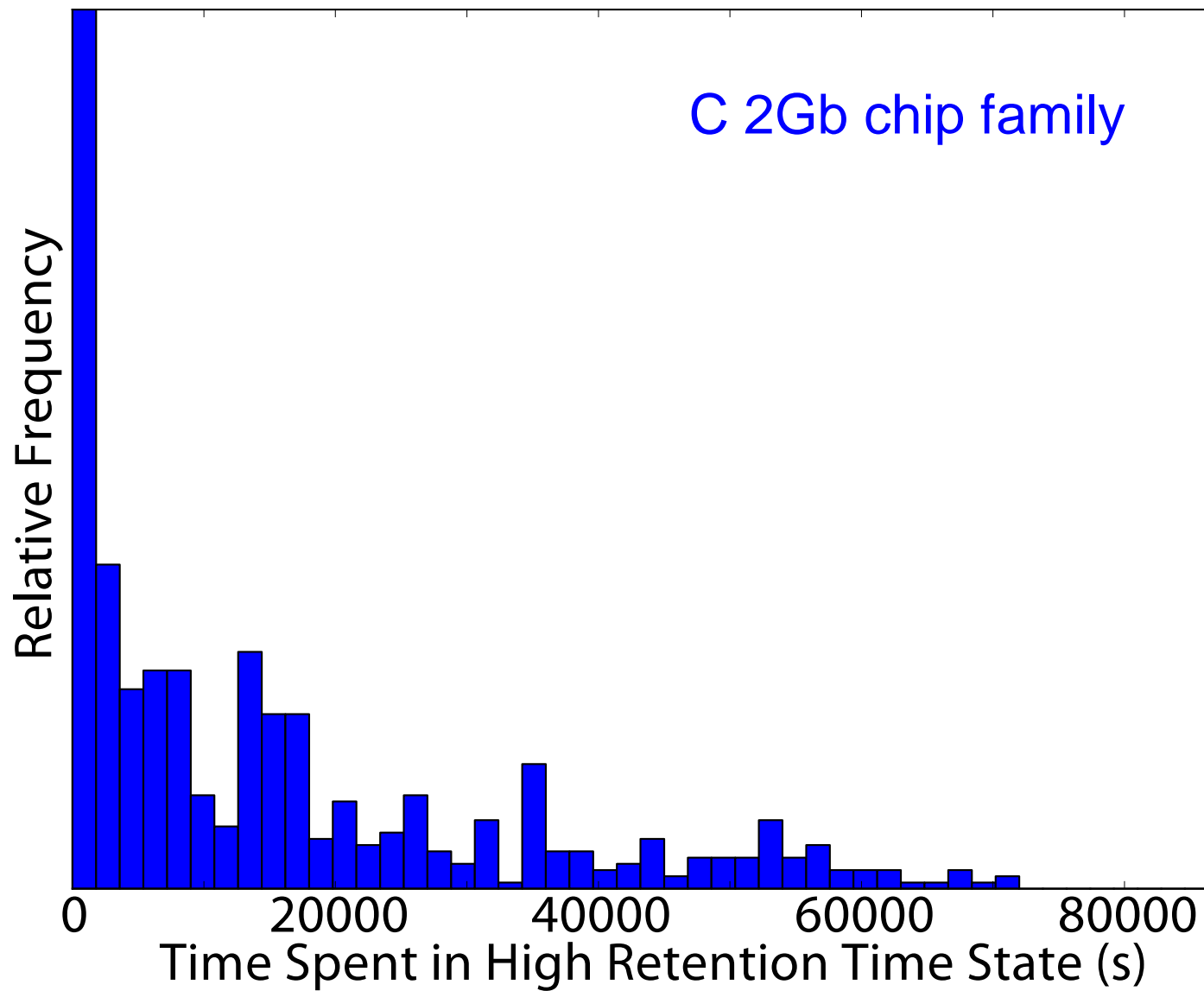


Need to profile for a long time to
get to the minimum retention time state

Time Spent in High Retention Time State



Time Spent in High Retention Time State



VRT: Implications on Profiling Mechanisms

- Problem 1: There does not seem to be a way of determining if a cell exhibits VRT without actually observing a cell exhibiting VRT
 - VRT is a memoryless random process [Kim+ JJAP 2010]
- Problem 2: VRT complicates retention time profiling by DRAM manufacturers
 - Exposure to very high temperatures can induce VRT in cells that were not previously susceptible
 - can happen during soldering of DRAM chips
 - manufacturer's retention time profile may not be accurate
- One option for future work: Use ECC to continuously profile DRAM online while aggressively reducing refresh rate
 - Need to keep ECC overhead in check

Talk Agenda

- DRAM Refresh: Background and Motivation
- Challenges and Our Goal
- DRAM Characterization Methodology
- Foundational Results
 - Temperature Dependence
 - Retention Time Distribution
- Data Pattern Dependence: Analysis and Implications
- Variable Retention Time: Analysis and Implications
- Conclusions

Summary and Conclusions

- DRAM refresh is a critical challenge in scaling DRAM technology efficiently to higher capacities and smaller feature sizes
- Understanding the retention time of modern DRAM devices can enable old or new methods to reduce the impact of refresh
 - Many mechanisms require accurate and reliable retention time profiles
- We presented the first work that comprehensively examines data retention behavior in modern commodity DRAM devices
 - Characterized 248 devices from five manufacturers
- Key findings: Retention time of a cell significantly depends on data pattern stored in other cells (**data pattern dependence**) and changes over time via a random process (**variable retention time**)
 - Discussed the underlying reasons and provided suggestions
- Future research on retention time profiling should solve the challenges posed by the DPD and VRT phenomena

An Experimental Study of Data Retention Behavior in Modern DRAM Devices

Implications for Retention Time Profiling Mechanisms

Jamie Liu¹ Ben Jaiyen¹ Yoongu Kim¹

Chris Wilkerson² Onur Mutlu¹

¹ Carnegie Mellon University

² Intel Corporation

Flash Memory Scaling

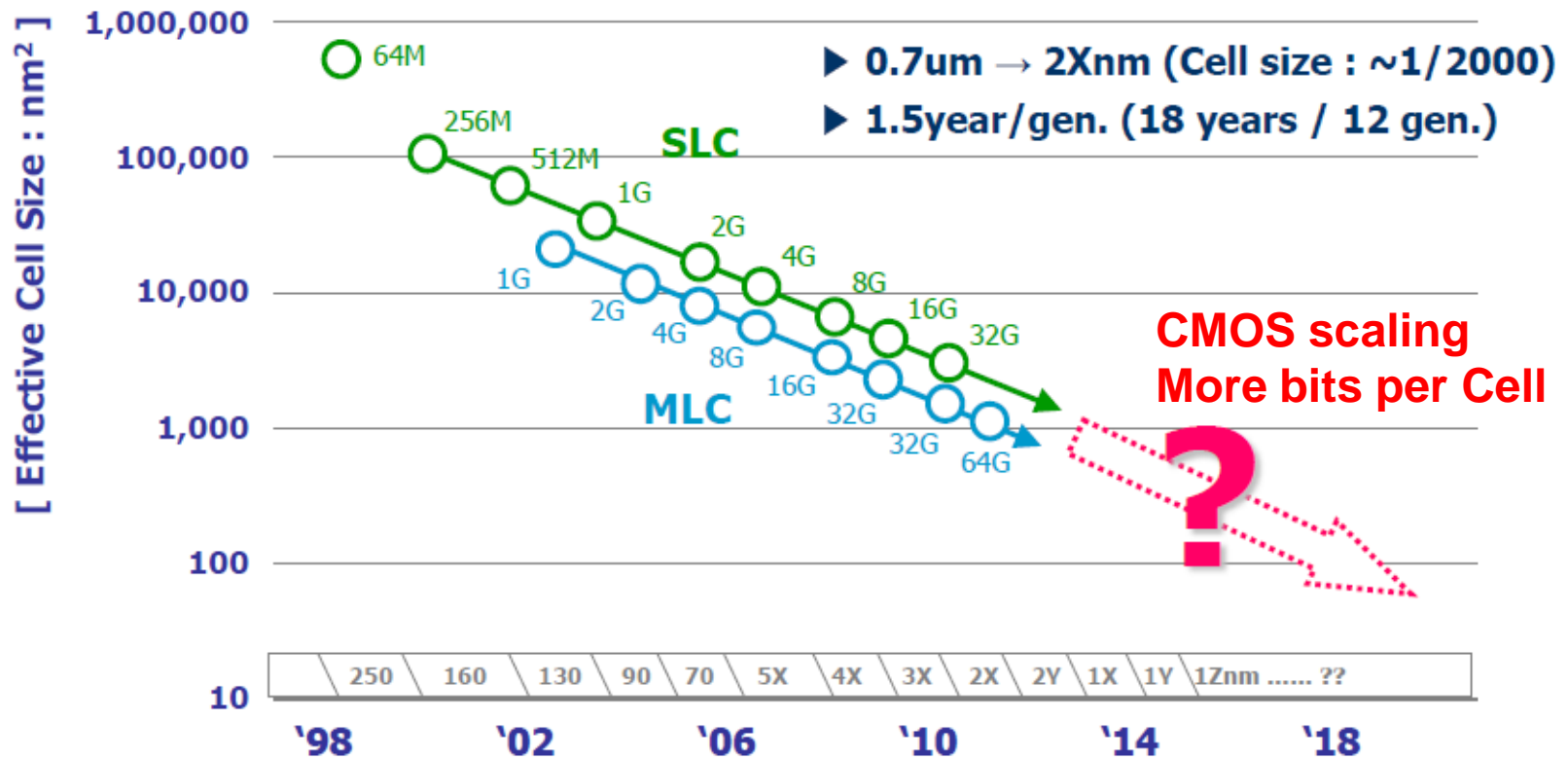
Aside: Scaling Flash Memory [Cai+, ICCD'12]

- NAND flash memory has low endurance: a flash cell dies after 3k P/E cycles vs. 50k desired → Major scaling challenge for flash memory
- Flash error rate increases exponentially over flash lifetime
- **Problem:** Stronger error correction codes (ECC) are ineffective and undesirable for improving flash lifetime due to
 - diminishing returns on lifetime with increased correction strength
 - prohibitively high power, area, latency overheads
- **Our Goal:** Develop techniques to tolerate high error rates w/o strong ECC
- **Observation:** Retention errors are the dominant errors in MLC NAND flash
 - flash cell loses charge over time; retention errors increase as cell gets worn out
- **Solution:** Flash Correct-and-Refresh (FCR)
 - Periodically read, correct, and reprogram (in place) or remap each flash page before it accumulates more errors than can be corrected by simple ECC
 - Adapt “refresh” rate to the severity of retention errors (i.e., # of P/E cycles)
- **Results:** FCR improves flash memory lifetime by 46X with no hardware changes and low energy overhead; outperforms strong ECCs

Readings in Flash Memory

- Yu Cai, Gulay Yalcin, Onur Mutlu, Erich F. Haratsch, Adrian Cristal, Osman Unsal, and Ken Mai, **"Error Analysis and Retention-Aware Error Management for NAND Flash Memory"** *Intel Technology Journal (ITJ) Special Issue on Memory Resiliency*, Vol. 17, No. 1, May 2013.
- Yu Cai, Erich F. Haratsch, Onur Mutlu, and Ken Mai, **"Threshold Voltage Distribution in MLC NAND Flash Memory: Characterization, Analysis and Modeling"** *Proceedings of the Design, Automation, and Test in Europe Conference (DATE)*, Grenoble, France, March 2013. [Slides \(ppt\)](#)
- Yu Cai, Gulay Yalcin, Onur Mutlu, Erich F. Haratsch, Adrian Cristal, Osman Unsal, and Ken Mai, **"Flash Correct-and-Refresh: Retention-Aware Error Management for Increased Flash Memory Lifetime"** *Proceedings of the 30th IEEE International Conference on Computer Design (ICCD)*, Montreal, Quebec, Canada, September 2012. [Slides \(ppt\)](#) [\(pdf\)](#)
- Yu Cai, Erich F. Haratsch, Onur Mutlu, and Ken Mai, **"Error Patterns in MLC NAND Flash Memory: Measurement, Characterization, and Analysis"** *Proceedings of the Design, Automation, and Test in Europe Conference (DATE)*, Dresden, Germany, March 2012. [Slides \(ppt\)](#)

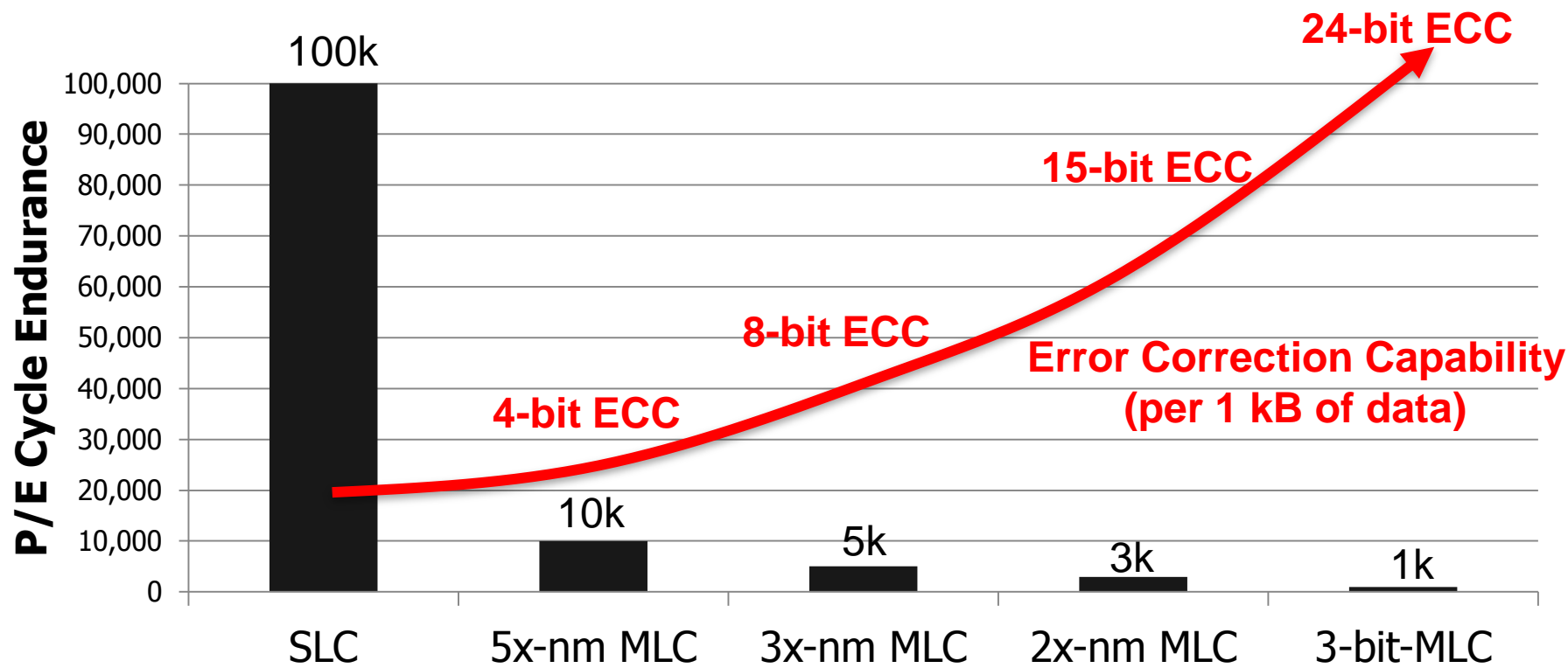
Evolution of NAND Flash Memory



Seaung Suk Lee, "Emerging Challenges in NAND Flash Technology", Flash Summit 2011 (Hynix)

- Flash memory widening its range of applications
 - Portable consumer devices, laptop PCs and enterprise servers

Decreasing Endurance with Flash Scaling

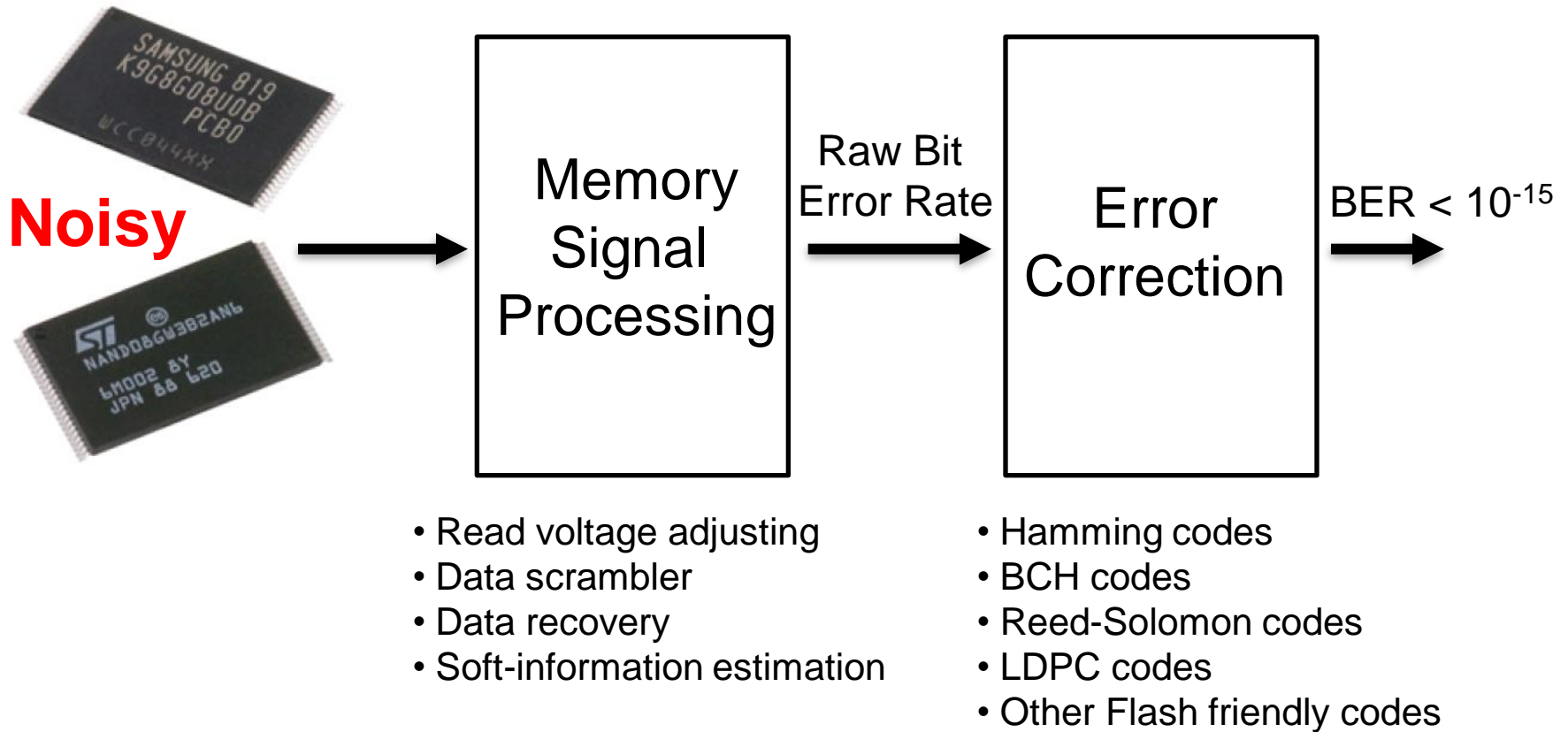


Ariel Maislos, "A New Era in Embedded Flash Memory", Flash Summit 2011 (Anobit)

- Endurance of flash memory decreasing with scaling and multi-level cells
- Error correction capability required to guarantee storage-class reliability (UBER < 10^{-15}) is increasing exponentially to reach less endurance

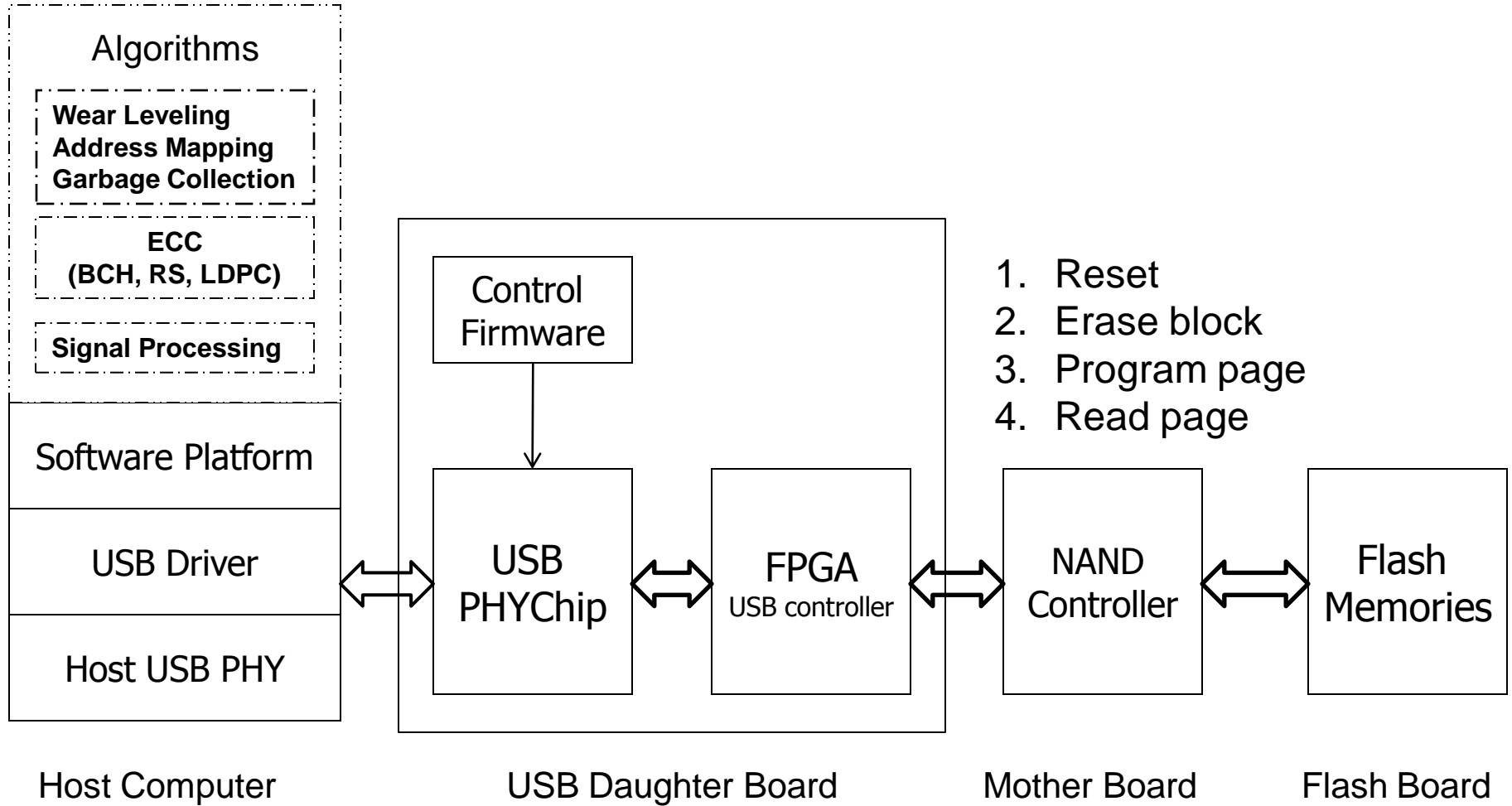
UBER: Uncorrectable bit error rate. Fraction of erroneous bits after error correction.

Future NAND Flash Storage Architecture

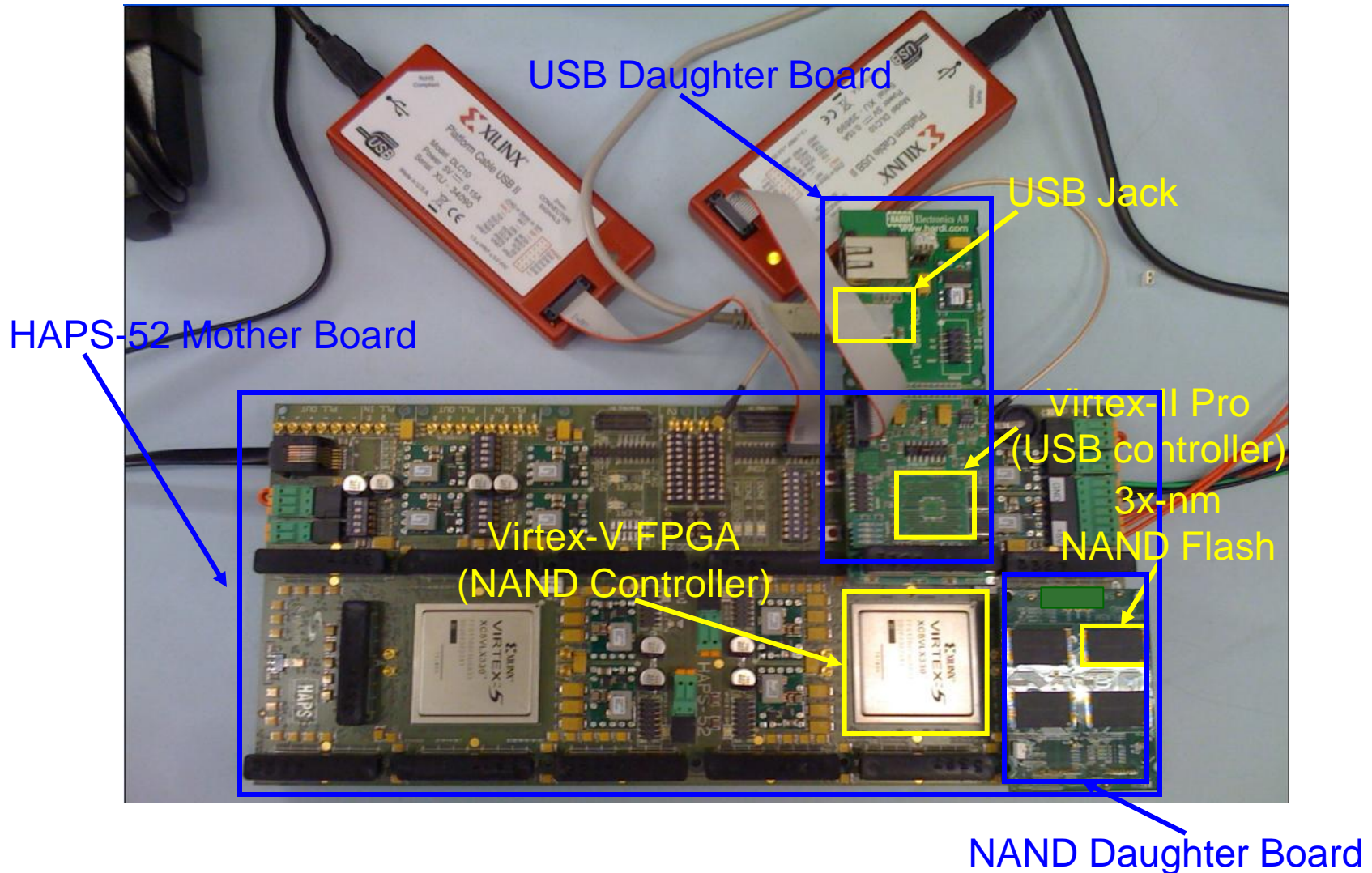


Need to understand NAND flash error patterns

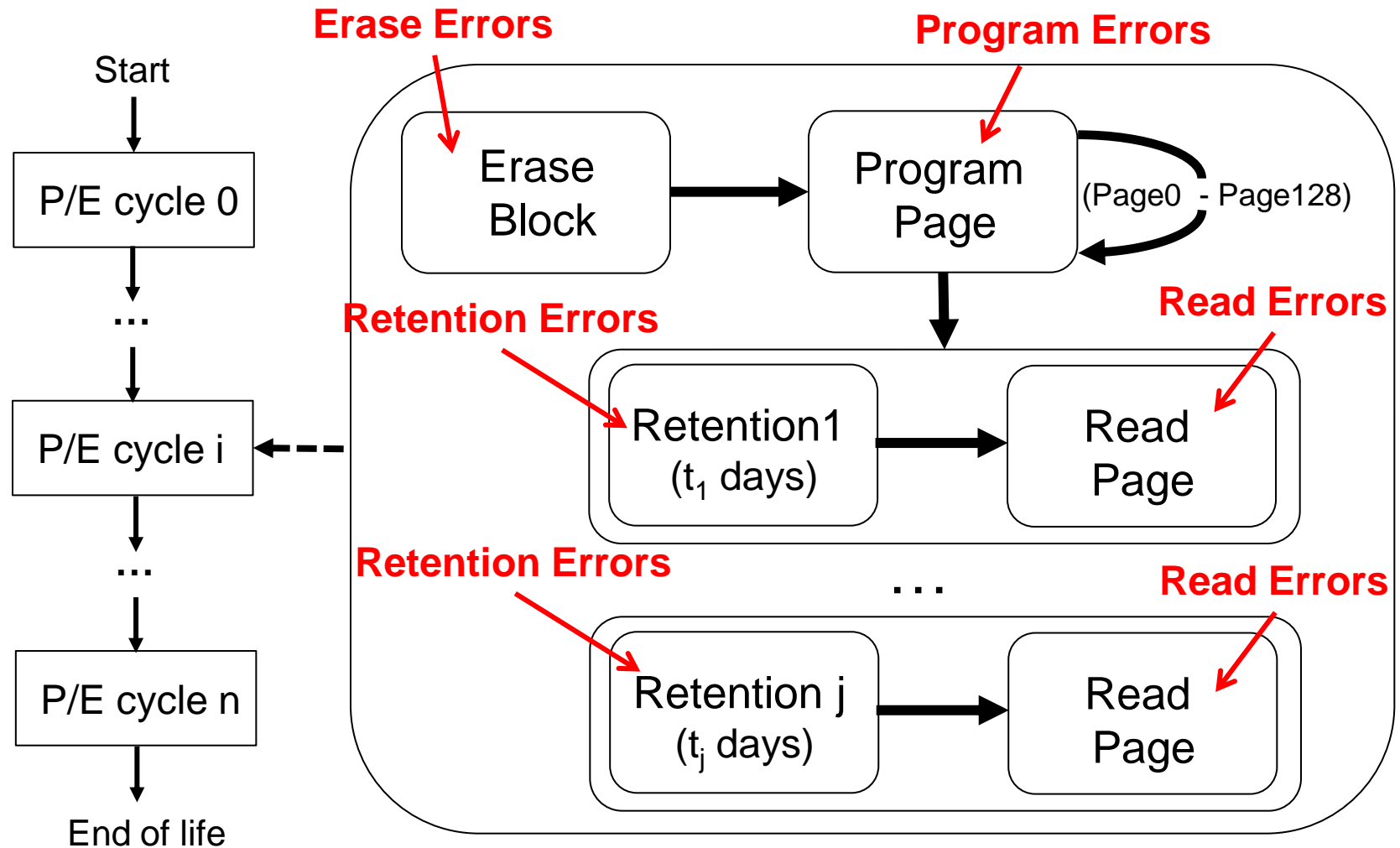
Test System Infrastructure



NAND Flash Testing Platform



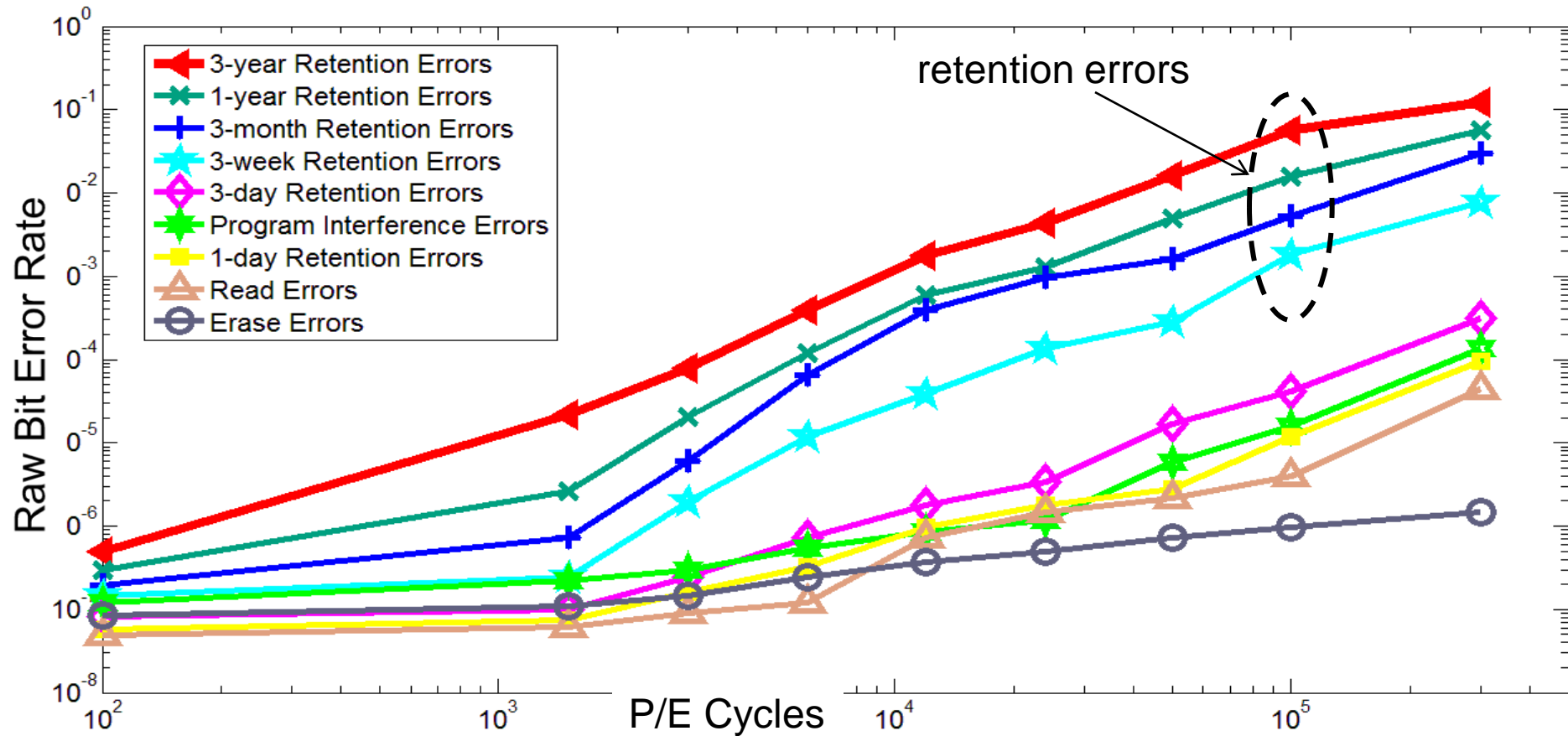
NAND Flash Usage and Error Model



Error Types and Testing Methodology

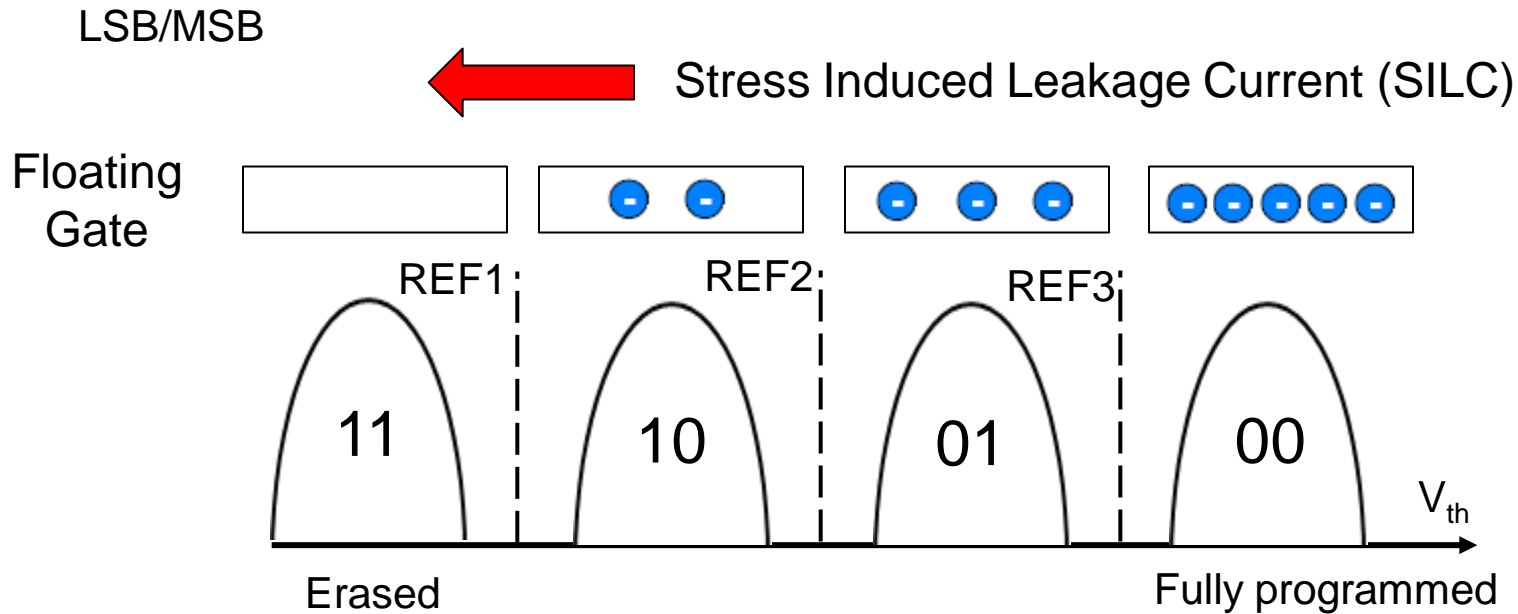
- Erase errors
 - Count the number of cells that fail to be erased to “11” state
- Program interference errors
 - Compare the data immediately after page programming and the data after the whole block being programmed
- Read errors
 - Continuously read a given block and compare the data between consecutive read sequences
- Retention errors
 - Compare the data read after an amount of time to data written
 - Characterize short term retention errors under room temperature
 - Characterize long term retention errors by baking in the oven under 125°C

Observations: Flash Error Analysis



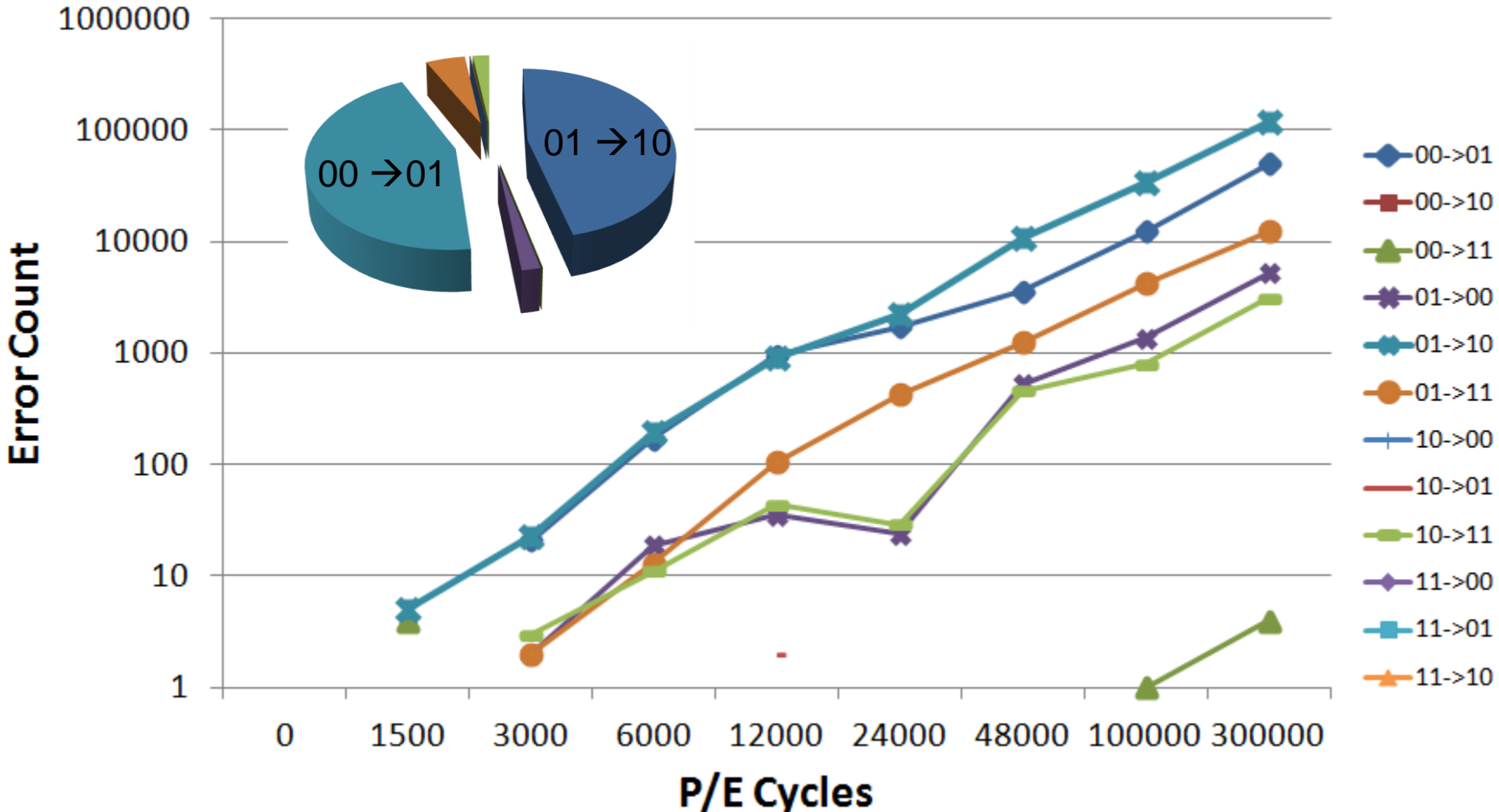
- Raw bit error rate increases exponentially with P/E cycles
- Retention errors are dominant (>99% for 1-year ret. time)
- Retention errors increase with retention time requirement

Retention Error Mechanism



- Electron loss from the floating gate causes retention errors
 - ❑ Cells with more programmed electrons suffer more from retention errors
 - ❑ Threshold voltage is more likely to shift by one window than by multiple

Retention Error Value Dependency

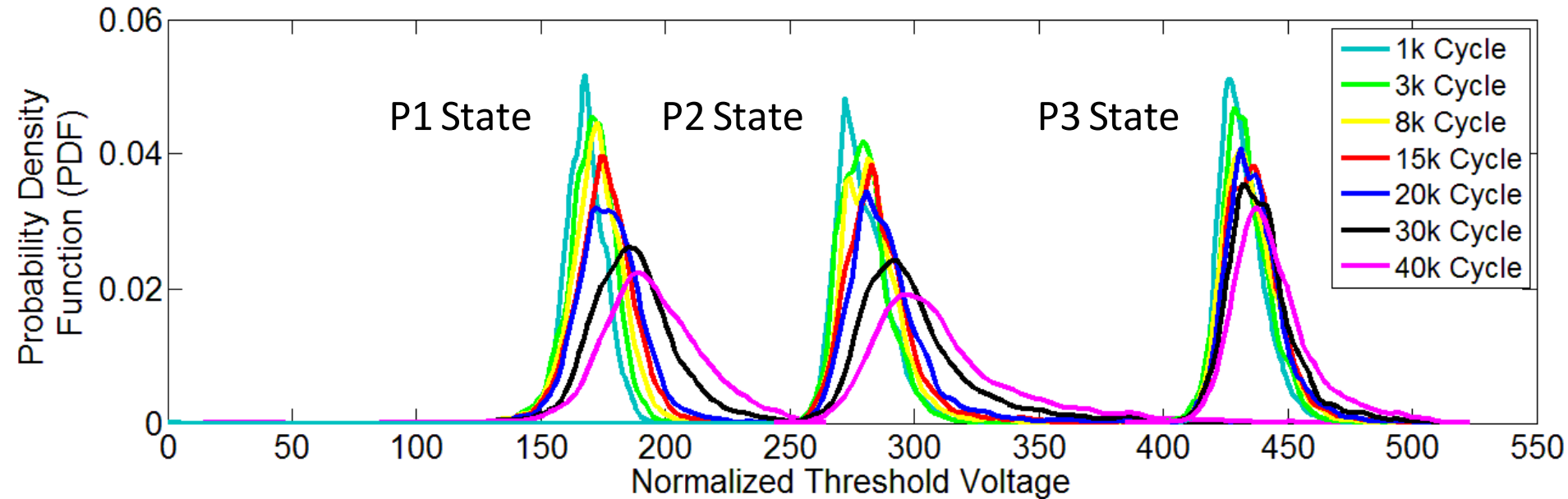


- Cells with more programmed electrons tend to suffer more from retention noise (i.e. 00 and 01)

More Details on Flash Error Analysis

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

Threshold Voltage Distribution Shifts



As P/E cycles increase ...

- Distribution shifts to the right
- Distribution becomes wider

More Detail

- Yu Cai, Erich F. Haratsch, Onur Mutlu, and Ken Mai, **"Threshold Voltage Distribution in MLC NAND Flash Memory: Characterization, Analysis and Modeling"** *Proceedings of the Design, Automation, and Test in Europe Conference (**DATE**)*, Grenoble, France, March 2013. Slides (ppt)

Flash Correct-and-Refresh

Retention-Aware Error Management for Increased Flash Memory Lifetime

Yu Cai¹ Gulay Yalcin² Onur Mutlu¹ Erich F. Haratsch³
Adrian Cristal² Osman S. Unsal² Ken Mai¹

¹ Carnegie Mellon University

² Barcelona Supercomputing Center

³ LSI Corporation



SAFARI

Carnegie Mellon

Executive Summary

- NAND flash memory has low endurance: a flash cell dies after 3k P/E cycles vs. 50k desired → Major scaling challenge for flash memory
 - Flash error rate increases exponentially over flash lifetime
 - **Problem:** Stronger error correction codes (ECC) are ineffective and undesirable for improving flash lifetime due to
 - diminishing returns on lifetime with increased correction strength
 - prohibitively high power, area, latency overheads
 - **Our Goal:** Develop techniques to tolerate high error rates w/o strong ECC
 - **Observation:** Retention errors are the dominant errors in MLC NAND flash
 - flash cell loses charge over time; retention errors increase as cell gets worn out
 - **Solution:** Flash Correct-and-Refresh (FCR)
 - Periodically read, correct, and reprogram (in place) or remap each flash page before it accumulates more errors than can be corrected by simple ECC
 - Adapt “refresh” rate to the severity of retention errors (i.e., # of P/E cycles)
 - **Results:** FCR improves flash memory lifetime by 46X with no hardware changes and low energy overhead; outperforms strong ECCs
-

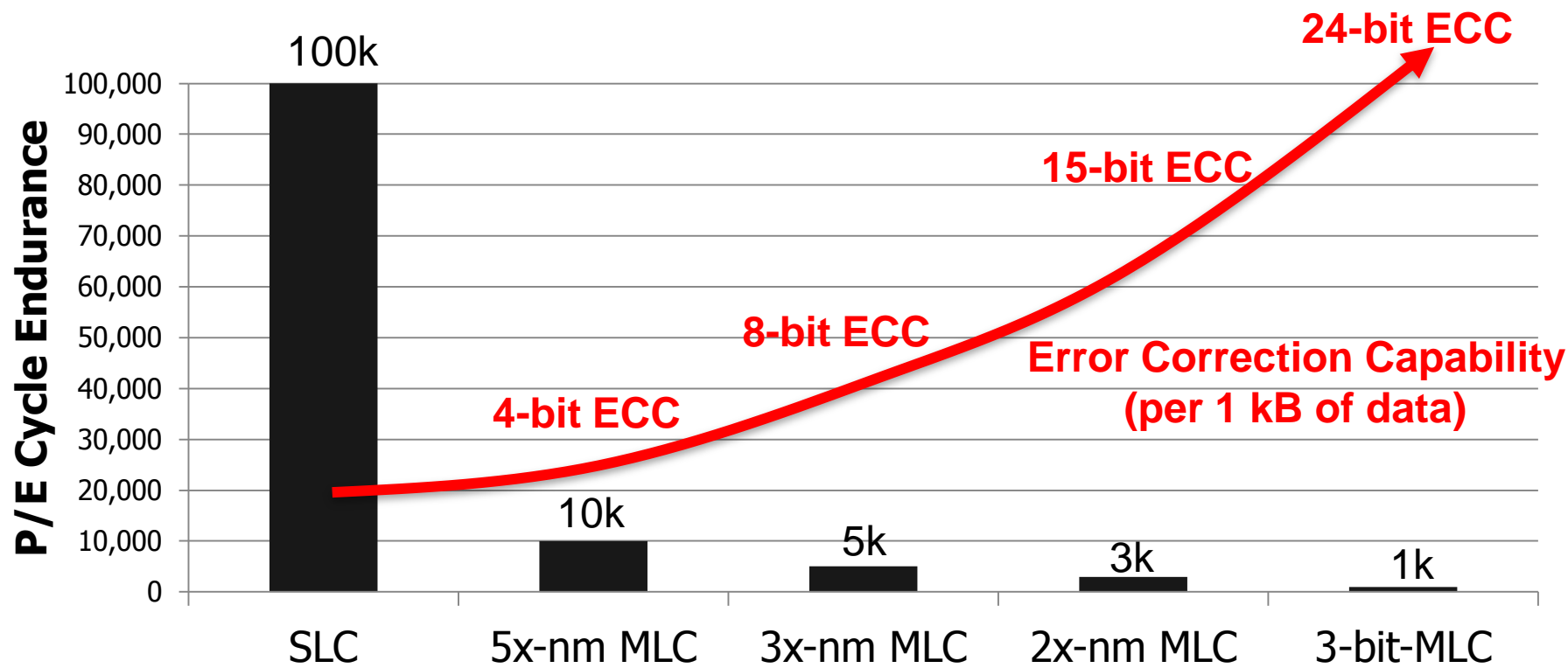
Outline

- Executive Summary
- **The Problem: Limited Flash Memory Endurance/Lifetime**
- Error and ECC Analysis for Flash Memory
- Flash Correct and Refresh Techniques (FCR)
- Evaluation
- Conclusions

Problem: Limited Endurance of Flash Memory

- NAND flash has limited endurance
 - A cell can tolerate a small number of Program/Erase (P/E) cycles
 - 3x-nm flash with 2 bits/cell → 3K P/E cycles
 - Enterprise data storage requirements demand very high endurance
 - >50K P/E cycles (10 full disk writes per day for 3-5 years)
 - Continued process scaling and more bits per cell will reduce flash endurance
 - One potential solution: stronger error correction codes (ECC)
 - Stronger ECC not effective enough and inefficient
-

Decreasing Endurance with Flash Scaling



Ariel Maislos, "A New Era in Embedded Flash Memory", Flash Summit 2011 (Anobit)

- Endurance of flash memory decreasing with scaling and multi-level cells
- Error correction capability required to guarantee storage-class reliability (UBER < 10^{-15}) is increasing exponentially to reach less endurance

UBER: Uncorrectable bit error rate. Fraction of erroneous bits after error correction.

The Problem with Stronger Error Correction

- Stronger ECC detects and corrects more raw bit errors → increases P/E cycles endured

- Two shortcomings of stronger ECC:

1. High implementation complexity

→ Power and area overheads increase super-linearly, but correction capability increases sub-linearly with ECC strength

2. Diminishing returns on flash lifetime improvement

→ Raw bit error rate increases exponentially with P/E cycles, but correction capability increases sub-linearly with ECC strength

Outline

- Executive Summary
- The Problem: Limited Flash Memory Endurance/Lifetime
- Error and ECC Analysis for Flash Memory
- Flash Correct and Refresh Techniques (FCR)
- Evaluation
- Conclusions

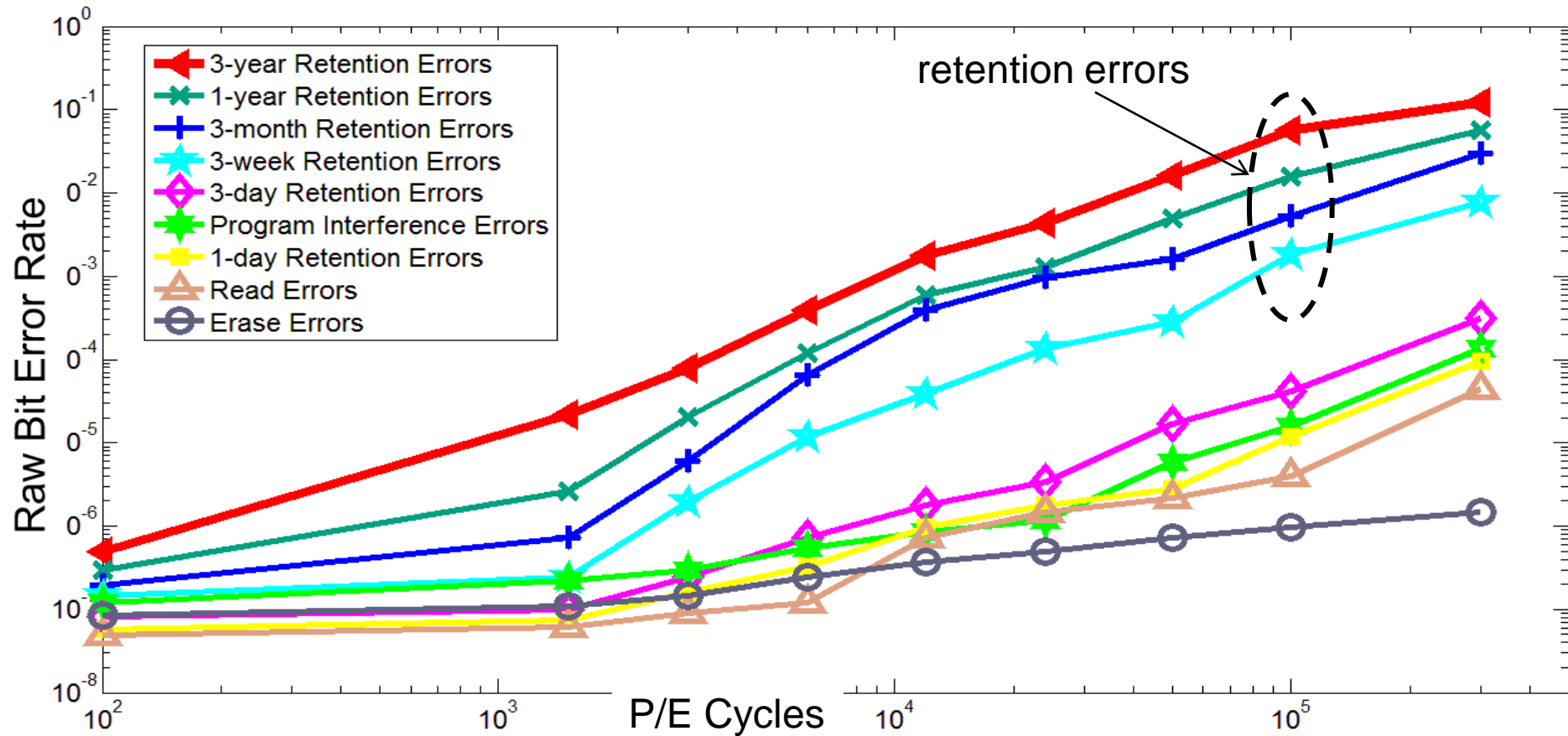
Methodology: Error and ECC Analysis

- **Characterized errors and error rates** of 3x-nm MLC NAND flash using an experimental FPGA-based flash platform
 - Cai et al., "Error Patterns in MLC NAND Flash Memory: Measurement, Characterization, and Analysis," DATE 2012.
- **Quantified Raw Bit Error Rate (RBER) at a given P/E cycle**
 - Raw Bit Error Rate: Fraction of erroneous bits without any correction
- **Quantified error correction capability** (and area and power consumption) of various BCH-code implementations
 - Identified how much RBER each code can tolerate
 - how many P/E cycles (flash lifetime) each code can sustain

NAND Flash Error Types

- Four types of errors [Cai+, DATE 2012]
- Caused by **common flash operations**
 - **Read** errors
 - **Erase** errors
 - **Program** (interference) errors
- Caused by flash **cell losing charge over time**
 - **Retention** errors
 - Whether an error happens depends on required retention time
 - Especially problematic in MLC flash because voltage threshold window to determine stored value is smaller

Observations: Flash Error Analysis



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

Methodology: Error and ECC Analysis

- **Characterized errors and error rates** of 3x-nm MLC NAND flash using an experimental FPGA-based flash platform
 - Cai et al., "Error Patterns in MLC NAND Flash Memory: Measurement, Characterization, and Analysis," DATE 2012.
- **Quantified Raw Bit Error Rate (RBER) at a given P/E cycle**
 - Raw Bit Error Rate: Fraction of erroneous bits without any correction
- **Quantified error correction capability** (and area and power consumption) of various BCH-code implementations
 - Identified how much RBER each code can tolerate
 - how many P/E cycles (flash lifetime) each code can sustain

ECC Strength Analysis

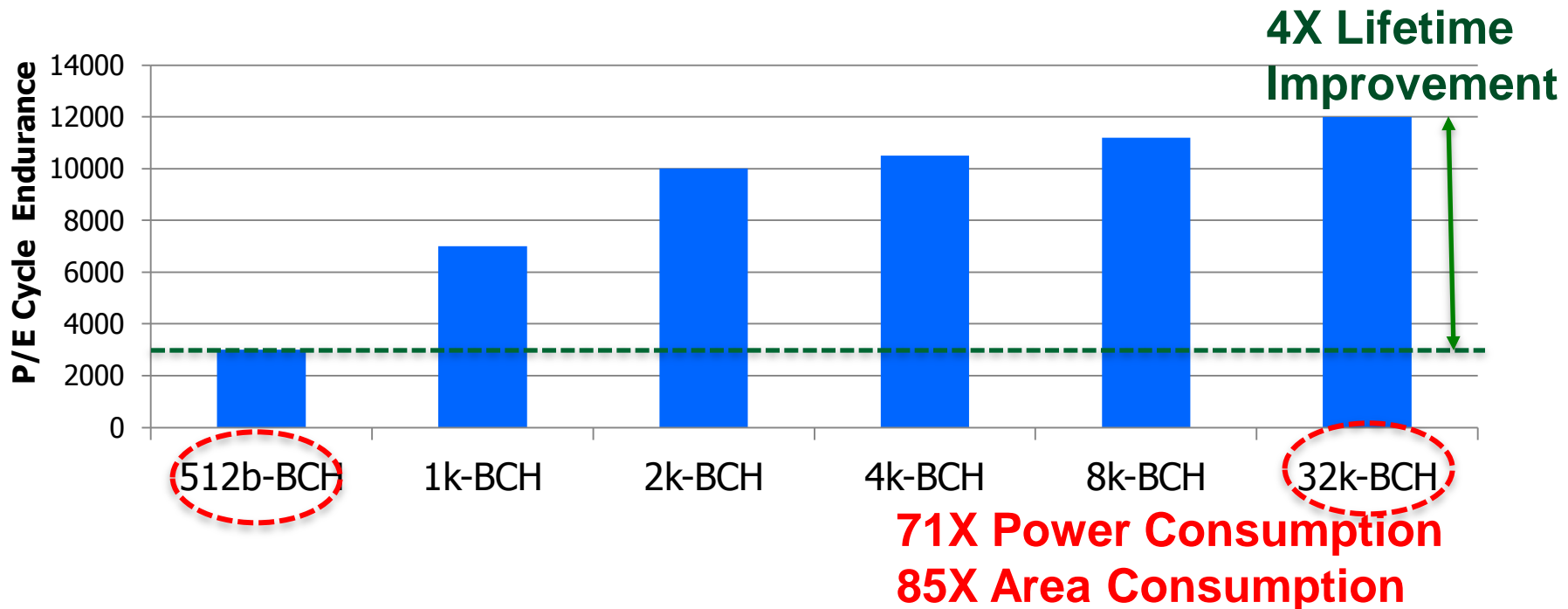
Error correction capability increases sub-linearly

Power and area overheads increase super-linearly

Code length (n)	Correctable Errors (t)	Acceptable Raw BER	Norm. Power	Norm. Area
512	7	1.0×10^{-4} (1x)	1	1
1024	12	4.0×10^{-4} (4x)	2	2.1
2048	22	1.0×10^{-3} (10x)	4.1	3.9
4096	40	1.7×10^{-3} (17x)	8.6	10.3
8192	74	2.2×10^{-3} (22x)	17.8	21.3
32768	259	2.6×10^{-3} (26x)	71	85

Resulting Flash Lifetime with Strong ECC

- Lifetime improvement comparison of various BCH codes



Strong ECC is very inefficient at improving lifetime

Our Goal

Develop new techniques
to improve flash lifetime
without relying on stronger ECC

Outline

- Executive Summary
- The Problem: Limited Flash Memory Endurance/Lifetime
- Error and ECC Analysis for Flash Memory
- Flash Correct and Refresh Techniques (FCR)
- Evaluation
- Conclusions

Flash Correct-and-Refresh (FCR)

- Key Observations:

- Retention errors are the dominant source of errors in flash memory [Cai+ DATE 2012][Tanakamaru+ ISSCC 2011]
→ limit flash lifetime as they increase over time
- Retention errors can be corrected by “refreshing” each flash page periodically

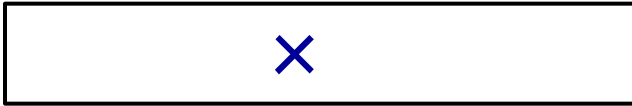
- Key Idea:

- Periodically read each flash page,
 - Correct its errors using “weak” ECC, and
 - Either remap it to a new physical page or reprogram it in-place,
 - Before the page accumulates more errors than ECC-correctable
 - Optimization: Adapt refresh rate to endured P/E cycles
-

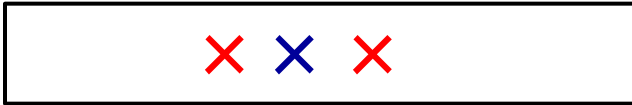
FCR Intuition

Errors with
No refresh

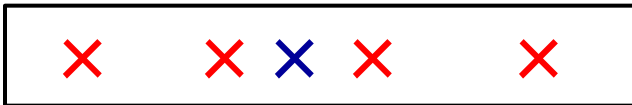
Program
Page



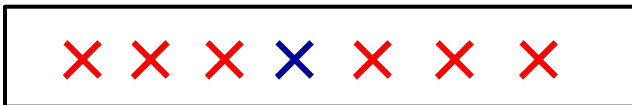
After
time T



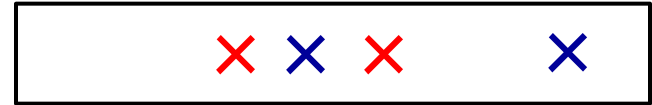
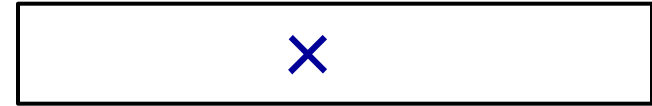
After
time 2T



After
time 3T



Errors with
Periodic refresh



X Retention Error

X Program Error

FCR: Two Key Questions

- How to refresh?
 - Remap a page to another one
 - Reprogram a page (in-place)
 - Hybrid of remap and reprogram

- When to refresh?
 - Fixed period
 - Adapt the period to retention error severity

Outline

- Executive Summary
- The Problem: Limited Flash Memory Endurance/Lifetime
- Error and ECC Analysis for Flash Memory
- Flash Correct and Refresh Techniques (FCR)
 1. Remapping based FCR
 2. Hybrid Reprogramming and Remapping based FCR
 3. Adaptive-Rate FCR
- Evaluation
- Conclusions

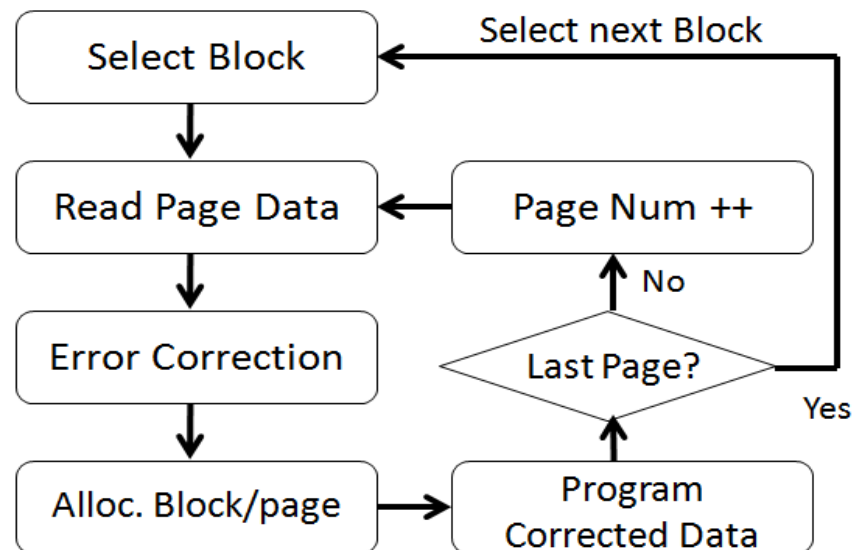
Outline

- Executive Summary
- The Problem: Limited Flash Memory Endurance/Lifetime
- Error and ECC Analysis for Flash Memory
- Flash Correct and Refresh Techniques (FCR)
 1. Remapping based FCR
 2. Hybrid Reprogramming and Remapping based FCR
 3. Adaptive-Rate FCR
- Evaluation
- Conclusions

Remapping Based FCR

- Idea: Periodically remap each page to a different physical page (after correcting errors)

- Also [Pan et al., HPCA 2012]
- FTL already has support for changing logical → physical flash block/page mappings
- Deallocated block is erased by garbage collector



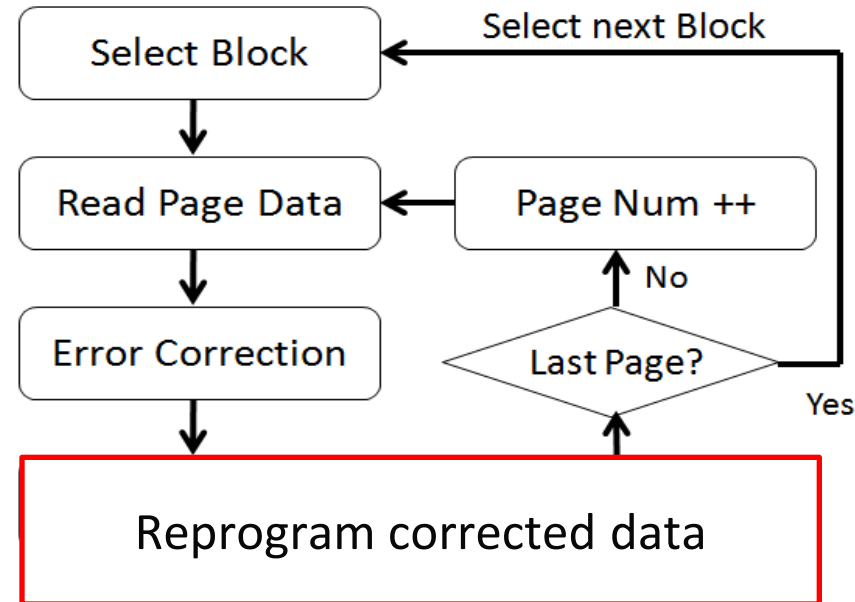
- Problem: Causes additional erase operations → more wearout
 - Bad for read-intensive workloads (few erases really needed)
 - Lifetime degrades for such workloads (see paper)

Outline

- Executive Summary
- The Problem: Limited Flash Memory Endurance/Lifetime
- Error and ECC Analysis for Flash Memory
- Flash Correct and Refresh Techniques (FCR)
 1. Remapping based FCR
 2. Hybrid Reprogramming and Remapping based FCR
 3. Adaptive-Rate FCR
- Evaluation
- Conclusions

In-Place Reprogramming Based FCR

- Idea: Periodically reprogram (in-place) each physical page (after correcting errors)
 - Flash programming techniques (ISPP) can correct retention errors in-place by recharging flash cells



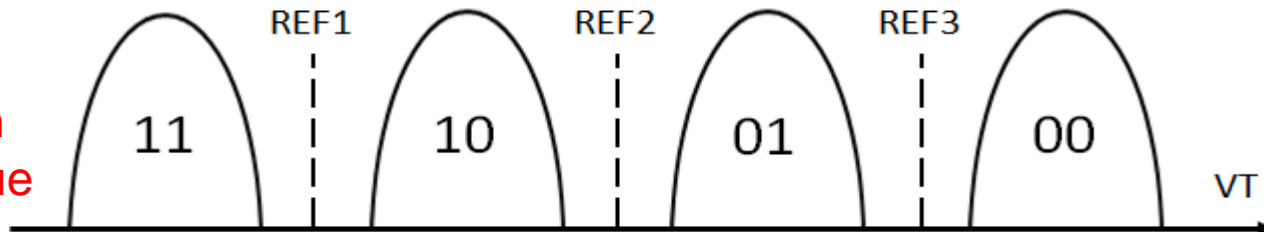
- Problem: Program errors accumulate on the same page → may not be correctable by ECC after some time

In-Place Reprogramming of Flash Cells

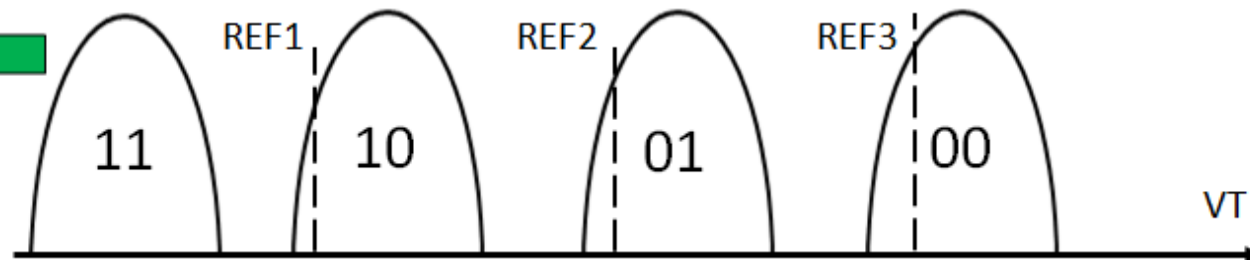
Floating Gate



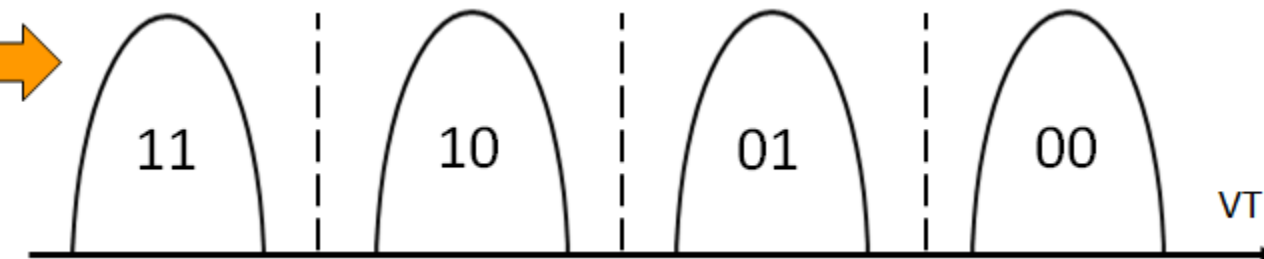
Floating Gate
Voltage Distribution
for each Stored Value



Retention errors are
caused by cell voltage
shifting to the left



ISPP moves cell
voltage to the right;
fixes retention errors

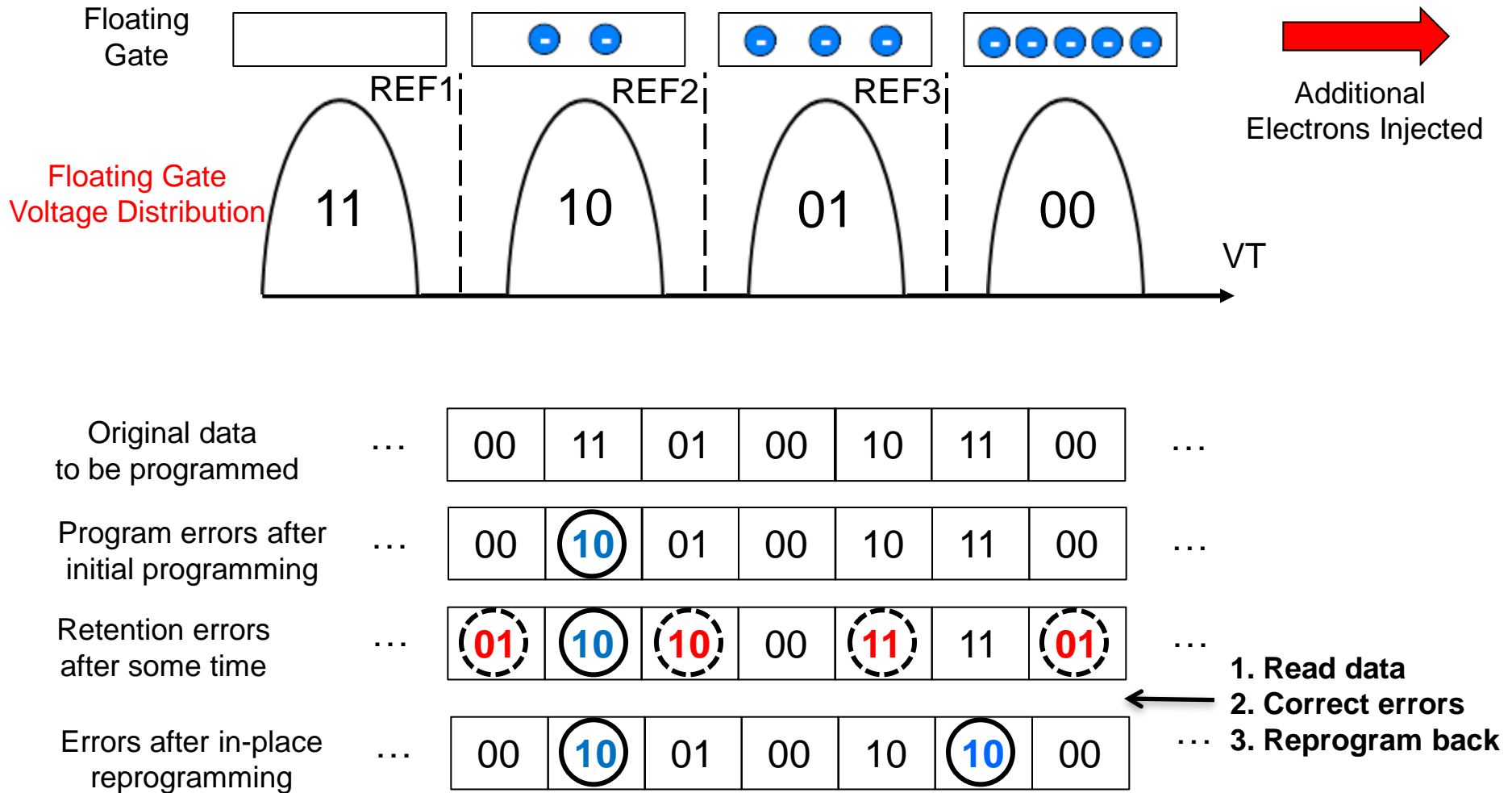


- Pro: No remapping needed → no additional erase operations
- Con: Increases the occurrence of program errors

Program Errors in Flash Memory

- When a cell is being programmed, **voltage level of a neighboring cell changes** (unintentionally) due to parasitic capacitance coupling
 - **can change the data value stored**
- Also called program interference error
- Program interference causes neighboring cell voltage to shift to the right

Problem with In-Place Reprogramming



Problem: Program errors can accumulate over time

Hybrid Reprogramming/Remapping Based FCR

- Idea:
 - Monitor the count of right-shift errors (after error correction)
 - If count < threshold, in-place reprogram the page
 - Else, remap the page to a new page
- Observation:
 - Program errors much less frequent than retention errors → Remapping happens only infrequently
- Benefit:
 - Hybrid FCR greatly reduces erase operations due to remapping

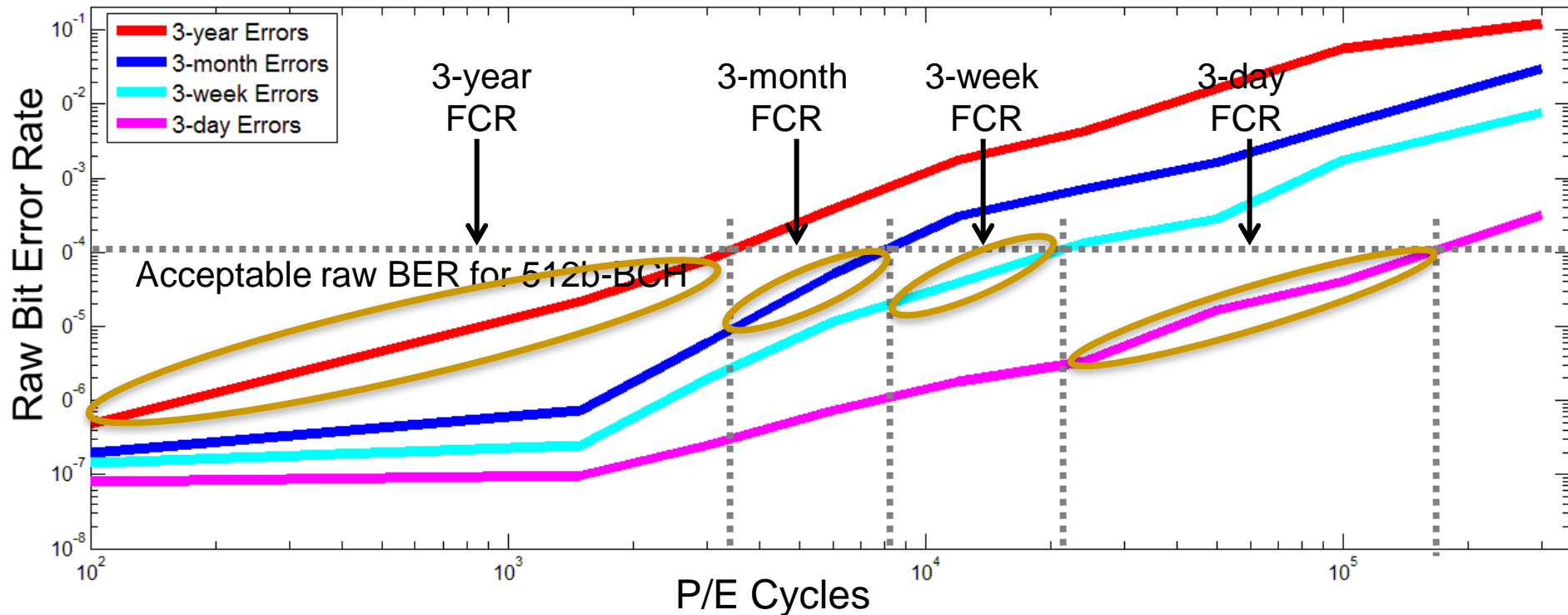
Outline

- Executive Summary
- The Problem: Limited Flash Memory Endurance/Lifetime
- Error and ECC Analysis for Flash Memory
- Flash Correct and Refresh Techniques (FCR)
 1. Remapping based FCR
 2. Hybrid Reprogramming and Remapping based FCR
 3. Adaptive-Rate FCR
- Evaluation
- Conclusions

Adaptive-Rate FCR

- Observation:
 - Retention error rate strongly depends on the P/E cycles a flash page endured so far
 - No need to refresh frequently (at all) early in flash lifetime
 - Idea:
 - Adapt the refresh rate to the P/E cycles endured by each page
 - Increase refresh rate gradually with increasing P/E cycles
 - Benefits:
 - Reduces overhead of refresh operations
 - Can use existing FTL mechanisms that keep track of P/E cycles
-

Adaptive-Rate FCR (Example)



Select refresh frequency such that error rate is below acceptable rate

Outline

- Executive Summary
- The Problem: Limited Flash Memory Endurance/Lifetime
- Error and ECC Analysis for Flash Memory
- Flash Correct and Refresh Techniques (FCR)
 1. Remapping based FCR
 2. Hybrid Reprogramming and Remapping based FCR
 3. Adaptive-Rate FCR
- Evaluation
- Conclusions

FCR: Other Considerations

- Implementation cost
 - No hardware changes
 - FTL software/firmware needs modification
- Response time impact
 - FCR not as frequent as DRAM refresh; low impact
- Adaptation to variations in retention error rate
 - Adapt refresh rate based on, e.g., temperature [Liu+ ISCA 2012]
- FCR requires power
 - Enterprise storage systems typically powered on

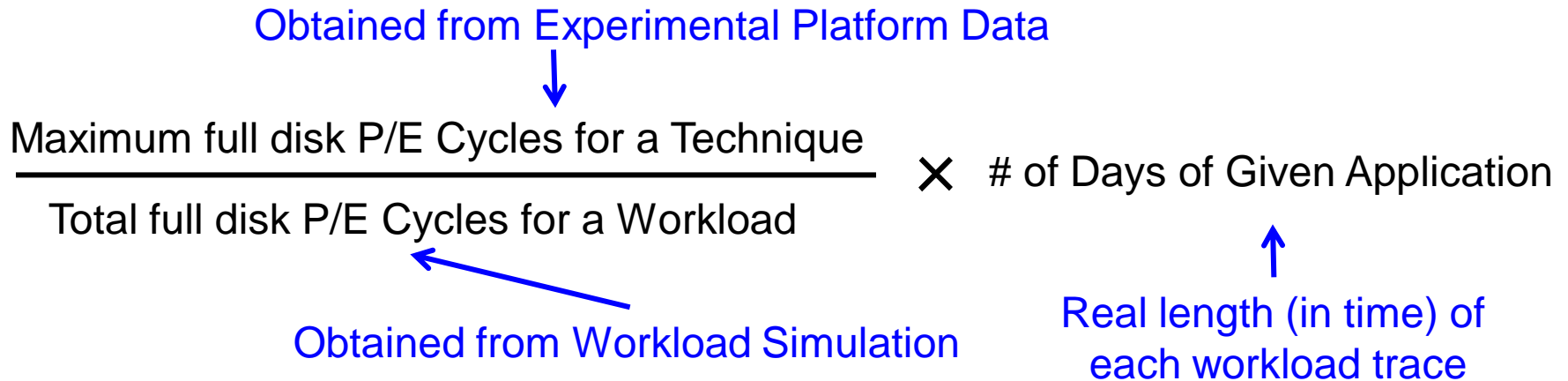
Outline

- Executive Summary
- The Problem: Limited Flash Memory Endurance/Lifetime
- Error and ECC Analysis for Flash Memory
- Flash Correct and Refresh Techniques (FCR)
- Evaluation
- Conclusions

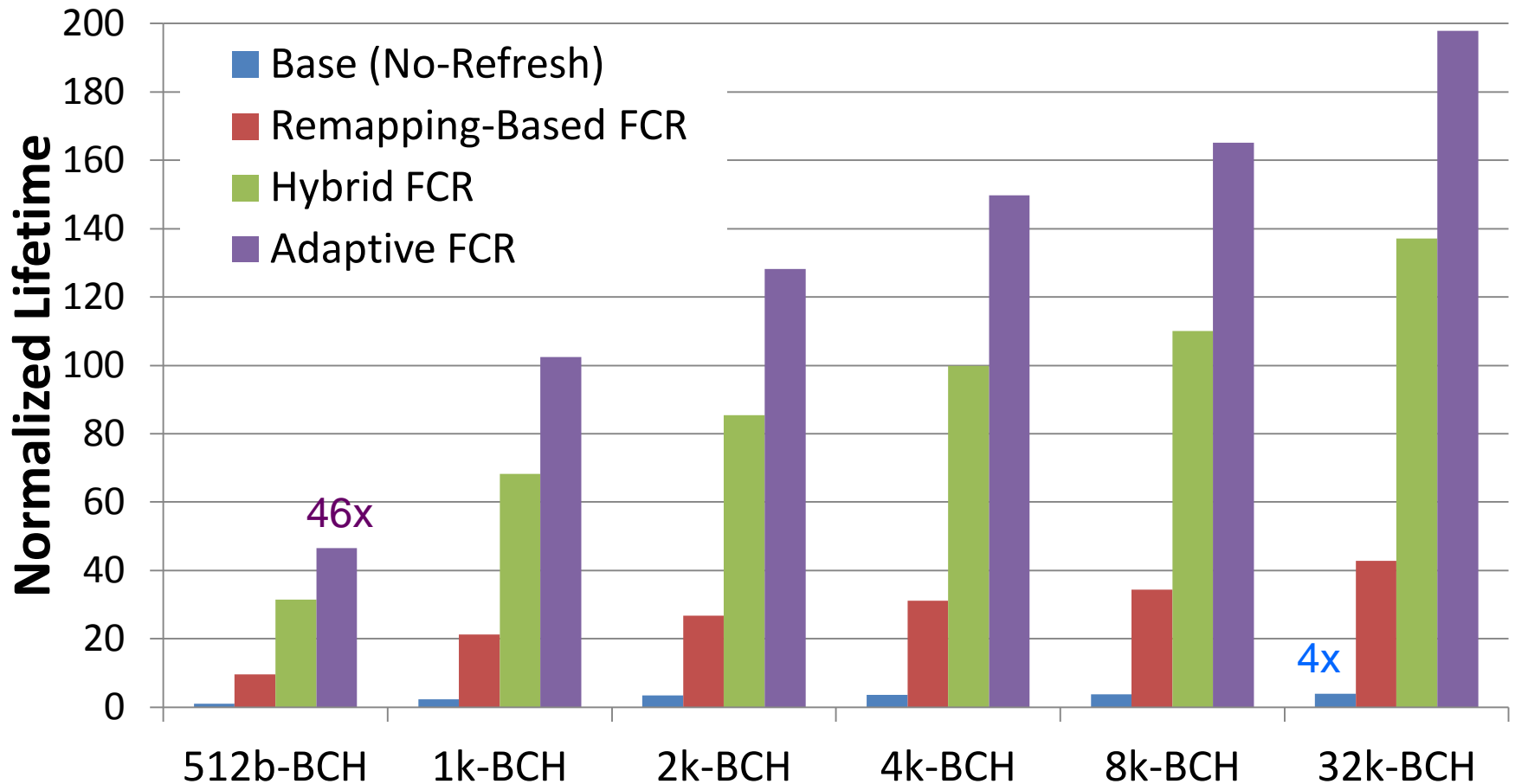
Evaluation Methodology

- Experimental flash platform to obtain error rates at different P/E cycles [Cai+ DATE 2012]
- Simulation framework to obtain P/E cycles of real workloads: DiskSim with SSD extensions
- Simulated system: 256GB flash, 4 channels, 8 chips/channel, 8K blocks/chip, 128 pages/block, 8KB pages
- Workloads
 - File system applications, databases, web search
 - Categories: Write-heavy, read-heavy, balanced
- Evaluation metrics
 - Lifetime (extrapolated)
 - Energy overhead, P/E cycle overhead

Extrapolated Lifetime



Normalized Flash Memory Lifetime

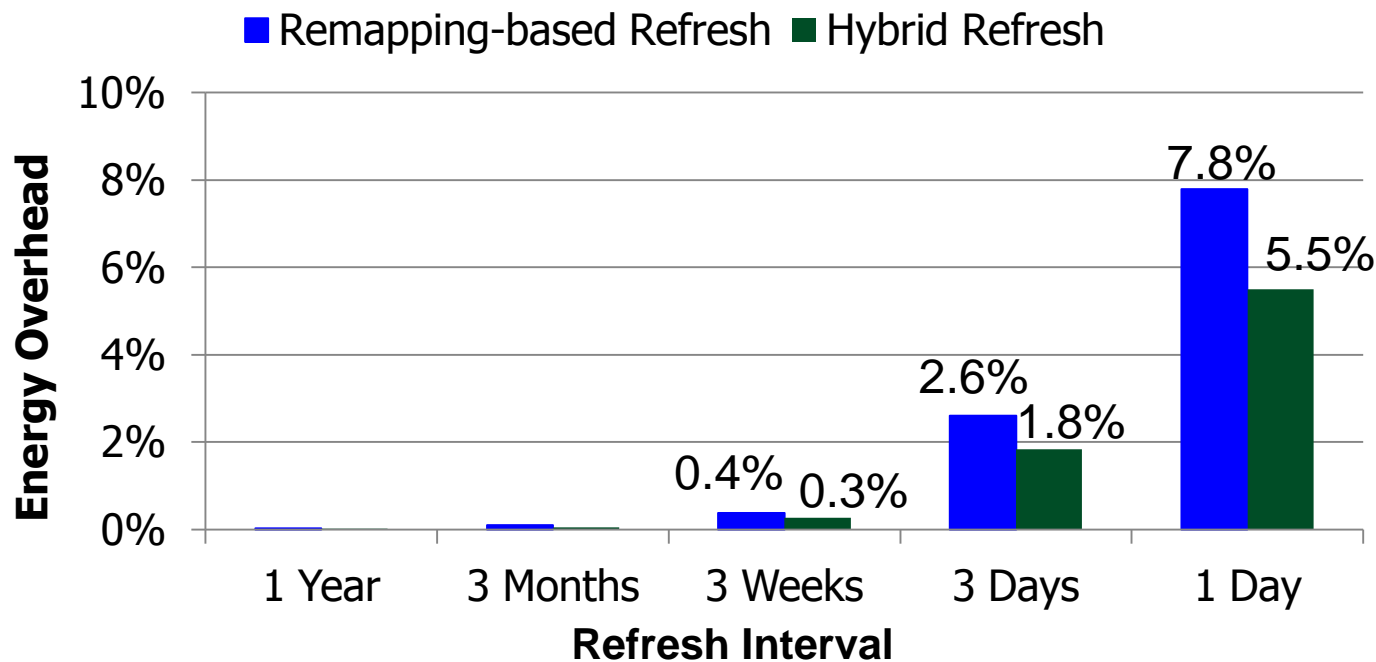


Lifetime of FCR much higher than lifetime of stronger ECC

Lifetime Evaluation Takeaways

- Significant average lifetime improvement over no refresh
 - Adaptive-rate FCR: 46X
 - Hybrid reprogramming/remapping based FCR: 31X
 - Remapping based FCR: 9X
- FCR lifetime improvement larger than that of stronger ECC
 - 46X vs. 4X with 32-kbit ECC (over 512-bit ECC)
 - FCR is less complex and less costly than stronger ECC
- Lifetime on all workloads improves with Hybrid FCR
 - Remapping based FCR can degrade lifetime on read-heavy WL
 - Lifetime improvement highest in write-heavy workloads

Energy Overhead



- Adaptive-rate refresh: <1.8% energy increase until daily refresh is triggered

Overhead of Additional Erases

- Additional erases happen due to remapping of pages
- Low (2%-20%) for write intensive workloads
- High (up to 10X) for read-intensive workloads
- Improved P/E cycle lifetime of all workloads largely outweighs the additional P/E cycles due to remapping

More Results in the Paper

- Detailed workload analysis
- Effect of refresh rate

Outline

- Executive Summary
- The Problem: Limited Flash Memory Endurance/Lifetime
- Error and ECC Analysis for Flash Memory
- Flash Correct and Refresh Techniques (FCR)
- Evaluation
- Conclusions

Conclusion

- NAND flash memory lifetime is limited due to uncorrectable errors, which increase over lifetime (P/E cycles)
- **Observation: Dominant source of errors in flash memory is retention errors** → retention error rate limits lifetime
- **Flash Correct-and-Refresh (FCR) techniques reduce retention error rate to improve flash lifetime**
 - **Periodically read, correct, and remap or reprogram each page** before it accumulates more errors than can be corrected
 - Adapt refresh period to the severity of errors
- **FCR improves flash lifetime by 46X at no hardware cost**
 - More effective and efficient than stronger ECC
 - Can enable better flash memory scaling

Flash Correct-and-Refresh

Retention-Aware Error Management for Increased Flash Memory Lifetime

Yu Cai¹ Gulay Yalcin² Onur Mutlu¹ Erich F. Haratsch³
Adrian Cristal² Osman S. Unsal² Ken Mai¹

¹ Carnegie Mellon University

² Barcelona Supercomputing Center

³ LSI Corporation



SAFARI

Carnegie Mellon