Efficient Data Mapping and Buffering Techniques for Multi-Level Cell Phase-Change Memories

HanBin Yoon, **Justin Meza**,
Naveen Muralimanohar*, Onur Mutlu, Norm Jouppi*†

Carnegie Mellon University * Hewlett-Packard Labs † Google, Inc.

Carnegie Mellon University
SAFARI

Google

Executive Summary

- Phase-change memory (PCM) is a promising emerging technology
  - More scalable than DRAM, faster than flash
  - Multi-level cell (MLC) PCM = multiple bits per cell → high density
- Problem: Higher latency/energy compared to non-MLC PCM
- Observation: MLC bits have ***asymmetric*** read/write characteristics
  - Some bits can be ***read quickly*** but ***written slowly*** and vice versa

Executive Summary

- Goal: Read data from ***fast-read*** bits; write data to ***fast-write*** bits
- Solution:
  - ***Decouple*** bits to expose fast-read/write memory regions
  - ***Map*** read/write-intensive data to appropriate memory regions
  - ***Split*** device row buffers to leverage decoupling for better locality
- Result:
  - Improved performance (+19.2%) and energy efficiency (+14.4%)
  - Across SPEC CPU2006 and data-intensive/cloud workloads

Outline

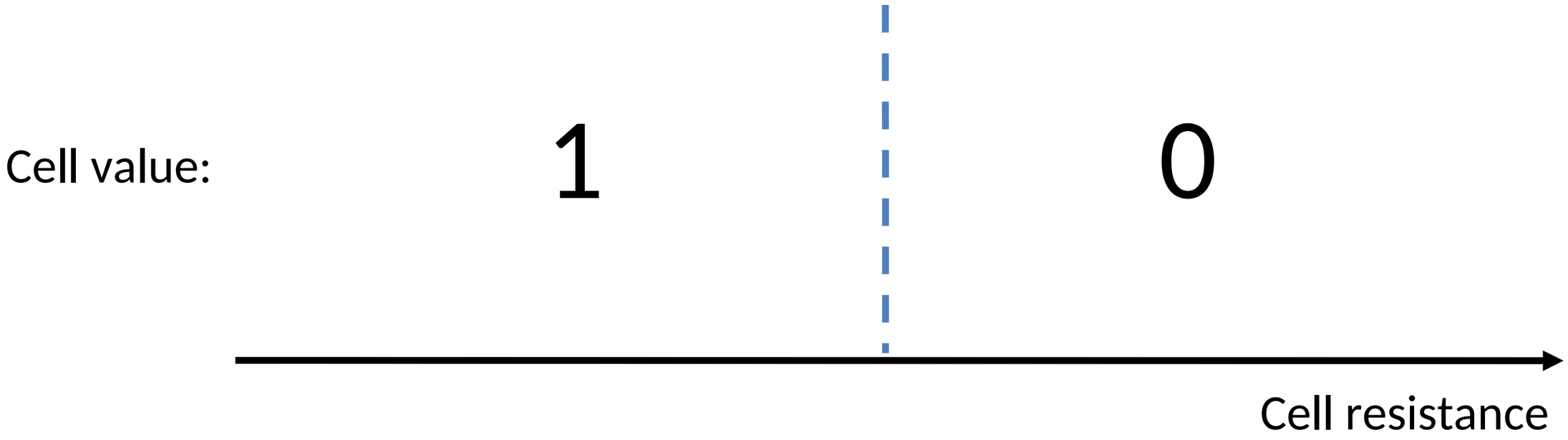
- Background
- Problem and Goal
- Key Observations
  - MLC-PCM cell read asymmetry
  - MLC-PCM cell write asymmetry
- Our Techniques
  - Decoupled Bit Mapping (DBM)
  - Asymmetric Page Mapping (APM)
  - Split Row Buffering (SRB)
- Results
- Conclusions

Background: PCM

- Emerging high-density memory technology
  - Potential for scalable DRAM alternative
    - Projected to be 3 to 12x denser than DRAM
    - Access latency within an order or magnitude of DRAM

- Stores data in the form of ***resistance*** of cell material

PCM Resistance → Value

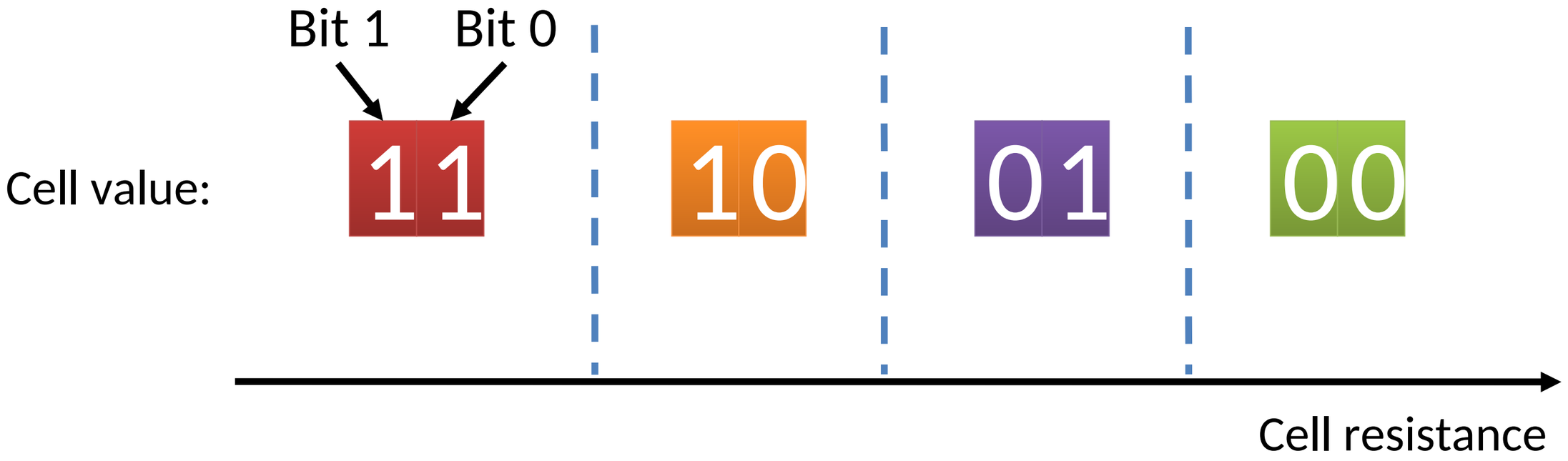


Background: MLC-PCM

- Multi-level cell: more than 1 bit per cell
  - Further increases density by 2 to 4x [Lee+,ISCA'09]

- But MLC-PCM also has drawbacks
  - Higher latency and energy than single-level cell PCM
  - Let's take a look at why this is the case

MLC-PCM Resistance → Value

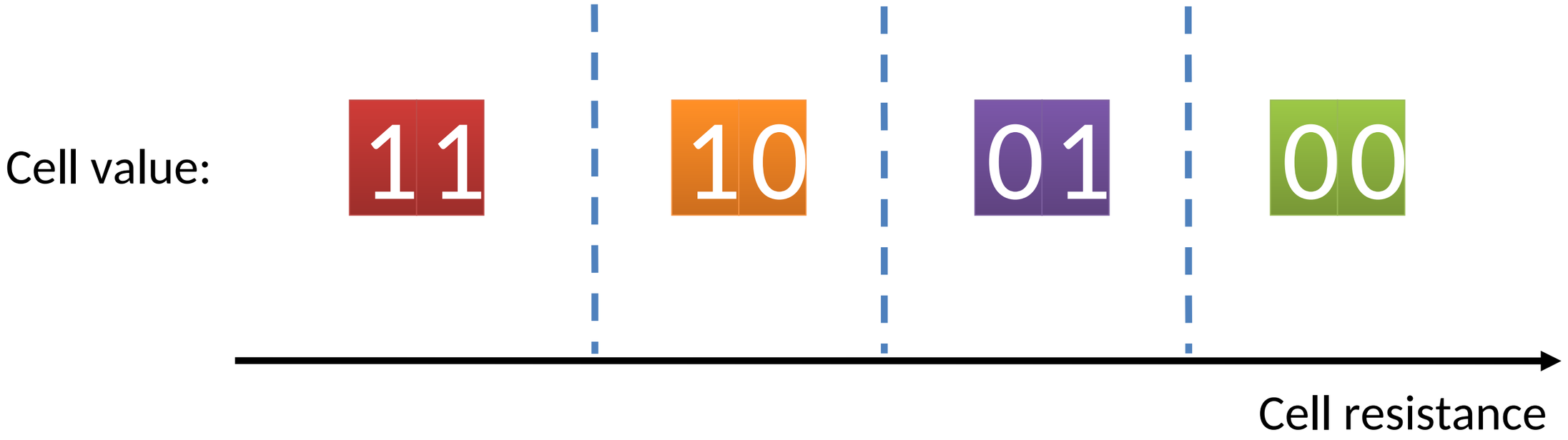


MLC-PCM Resistance → Value

Less margin between values

→ need more precise sensing/modification of cell contents

→ higher latency/energy (~2x for reads and 4x for writes)



Problem and Goal

- Want to leverage MLC-PCM's strengths
  - Higher density
  - More scalability than existing technologies (DRAM)
- But, also want to mitigate MLC-PCM's weaknesses
  - Higher latency/energy
- ***Our goal*** in this work is to design new hardware/software optimizations designed to mitigate the weaknesses of MLC-PCM

Outline

- Background
- Problem and Goal
- Key Observations
  - MLC-PCM cell read asymmetry
  - MLC-PCM cell write asymmetry
- Our Techniques
  - Decoupled Bit Mapping (DBM)
  - Asymmetric Page Mapping (APM)
  - Split Row Buffering (SRB)
- Results
- Conclusions

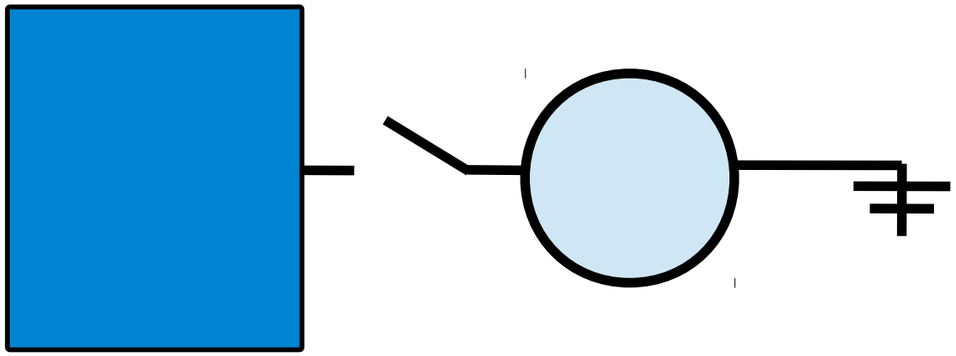
Observation 1: Read Asymmetry

- ***The read latency/energy of Bit 1 is lower than that of Bit 0***

- This is due to how MLC-PCM cells are read

Observation 1: Read Asymmetry

Simplified example

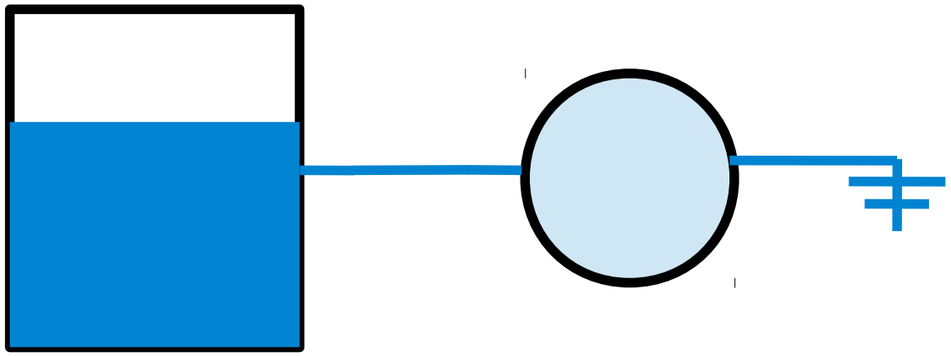


Capacitor filled with reference voltage

MLC-PCM cell with unknown resistance

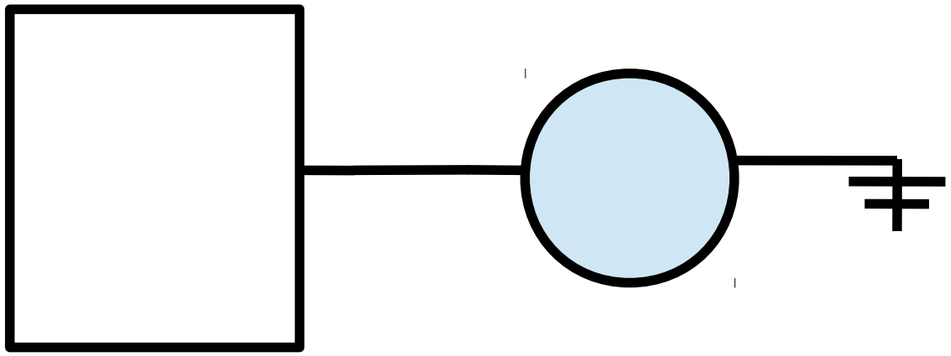
Observation 1: Read Asymmetry

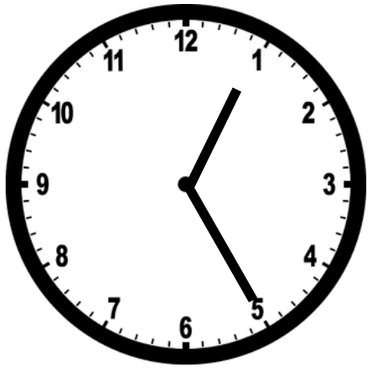
Simplified example



Observation 1: Read Asymmetry

Simplified example

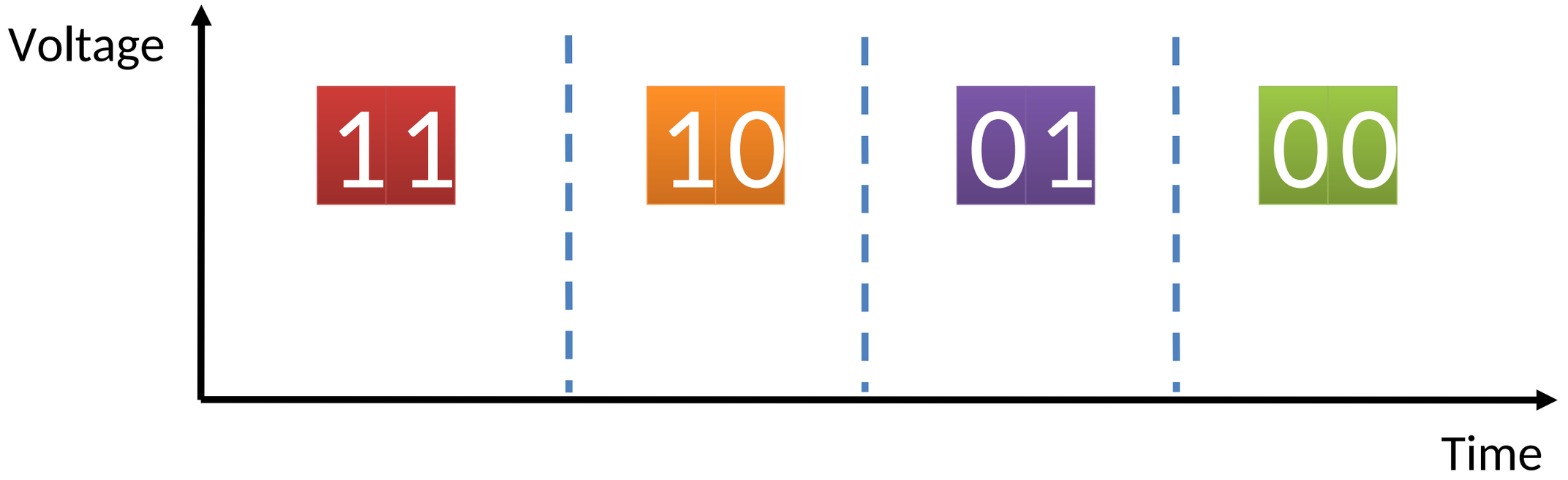




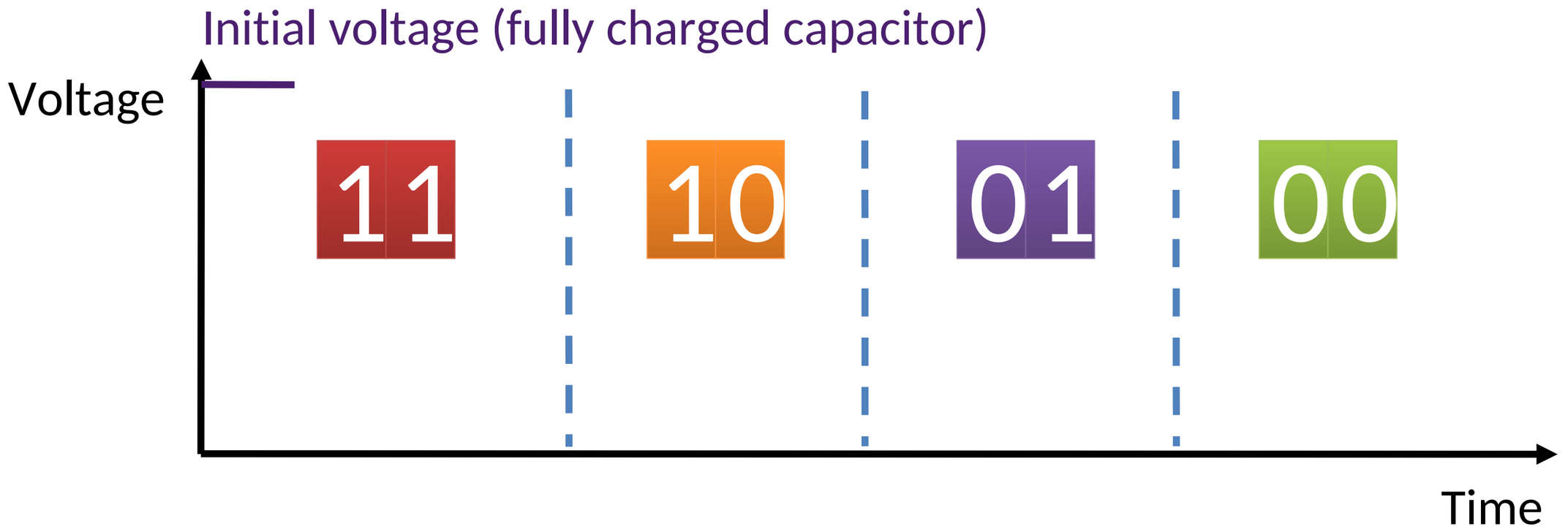
Infer data value

Observation 1: Read Asymmetry

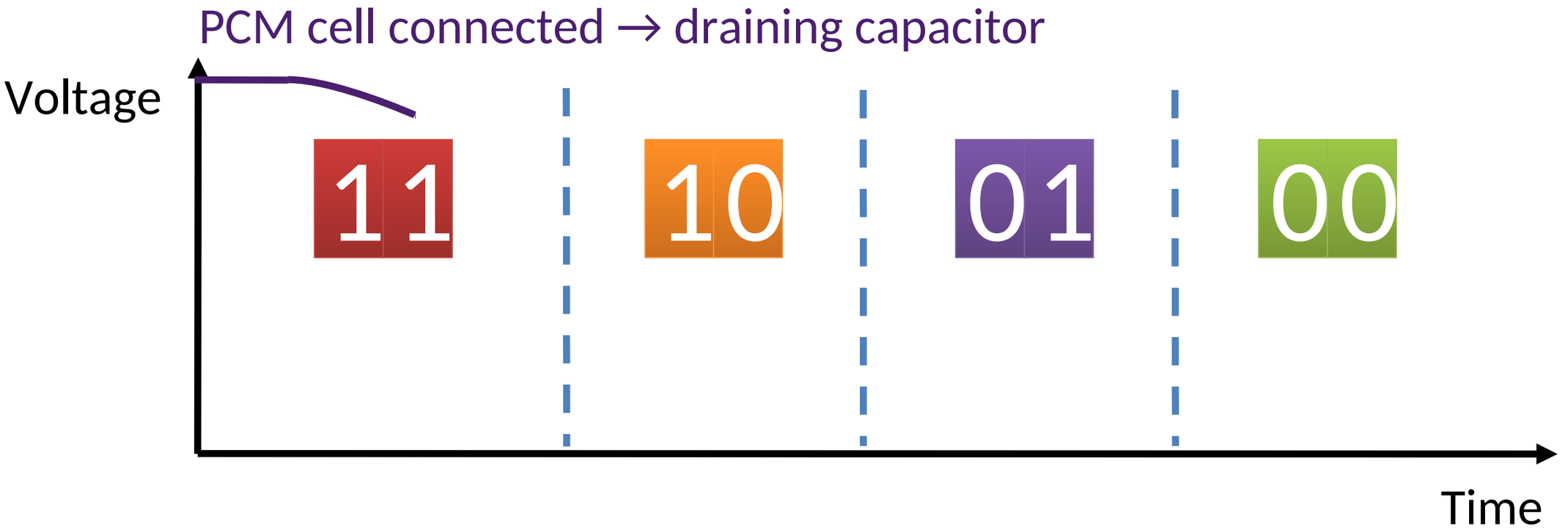
Observation 1: Read Asymmetry



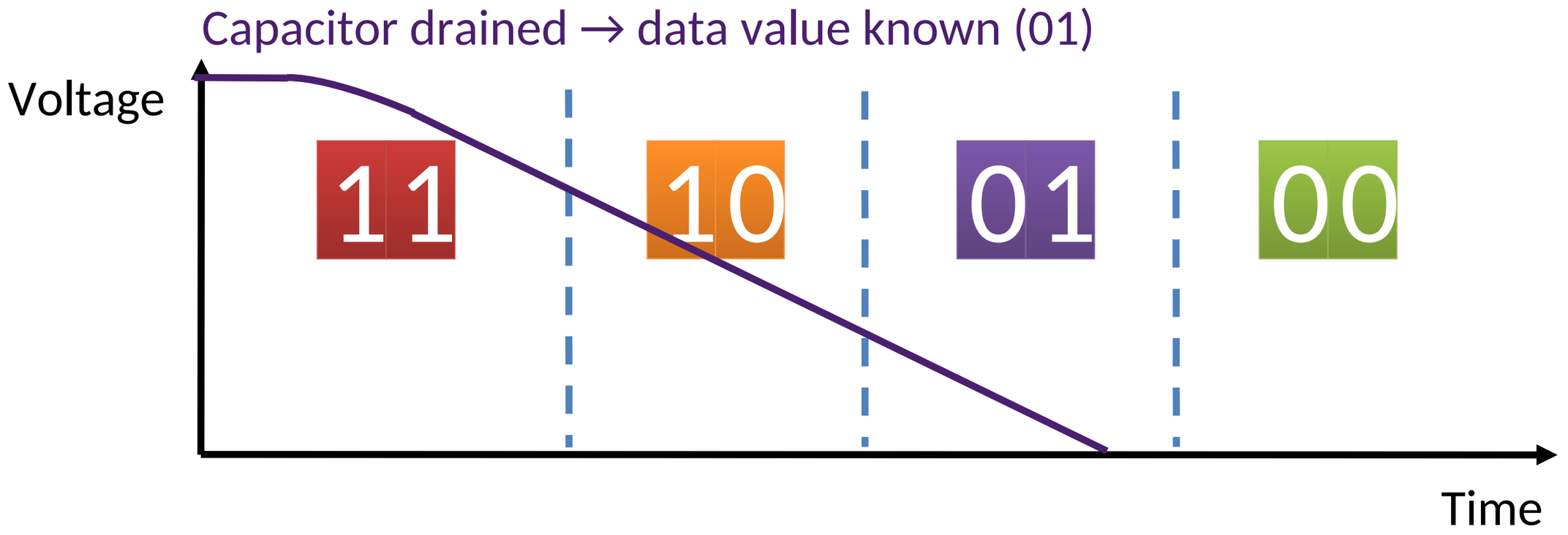
Observation 1: Read Asymmetry



Observation 1: Read Asymmetry



Observation 1: Read Asymmetry

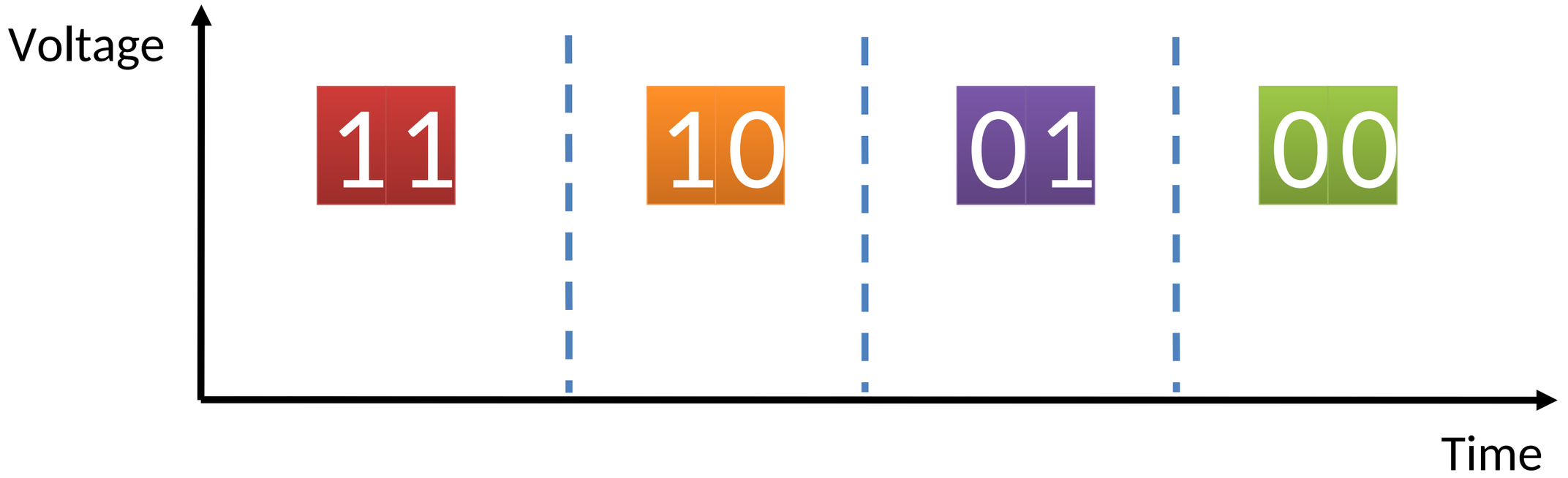


Observation 1: Read Asymmetry

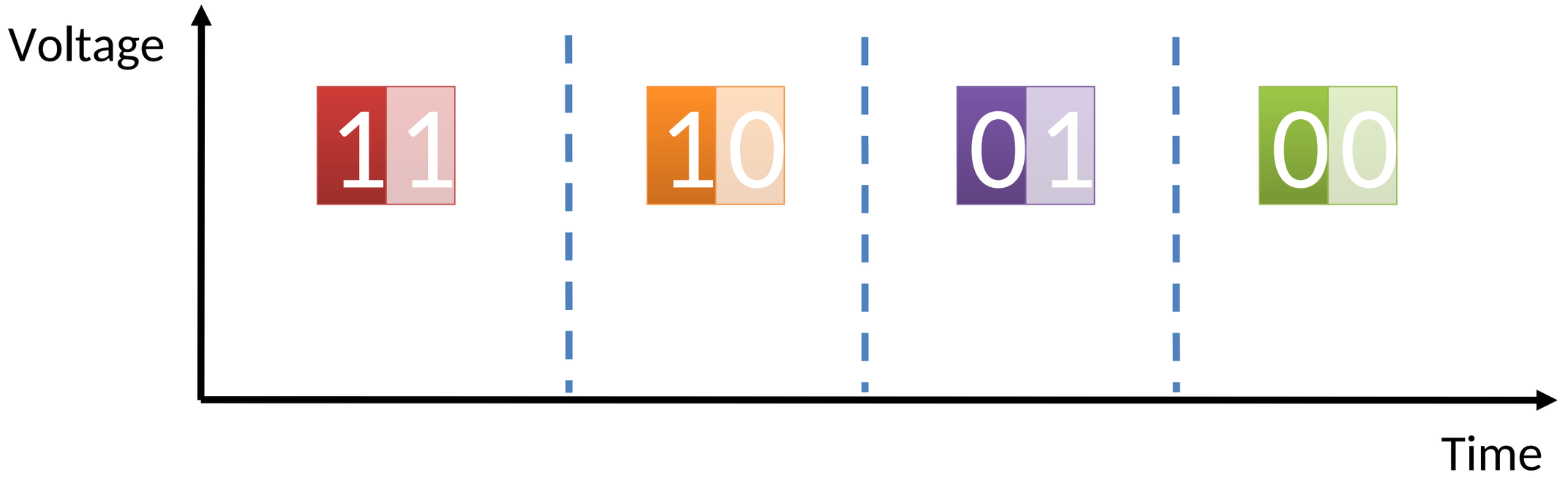
- In existing devices
  - Both MLC bits are read at the same time
  - Must wait ***maximum time*** to read both bits

- However, ***we can infer information about Bit 1 before this time***

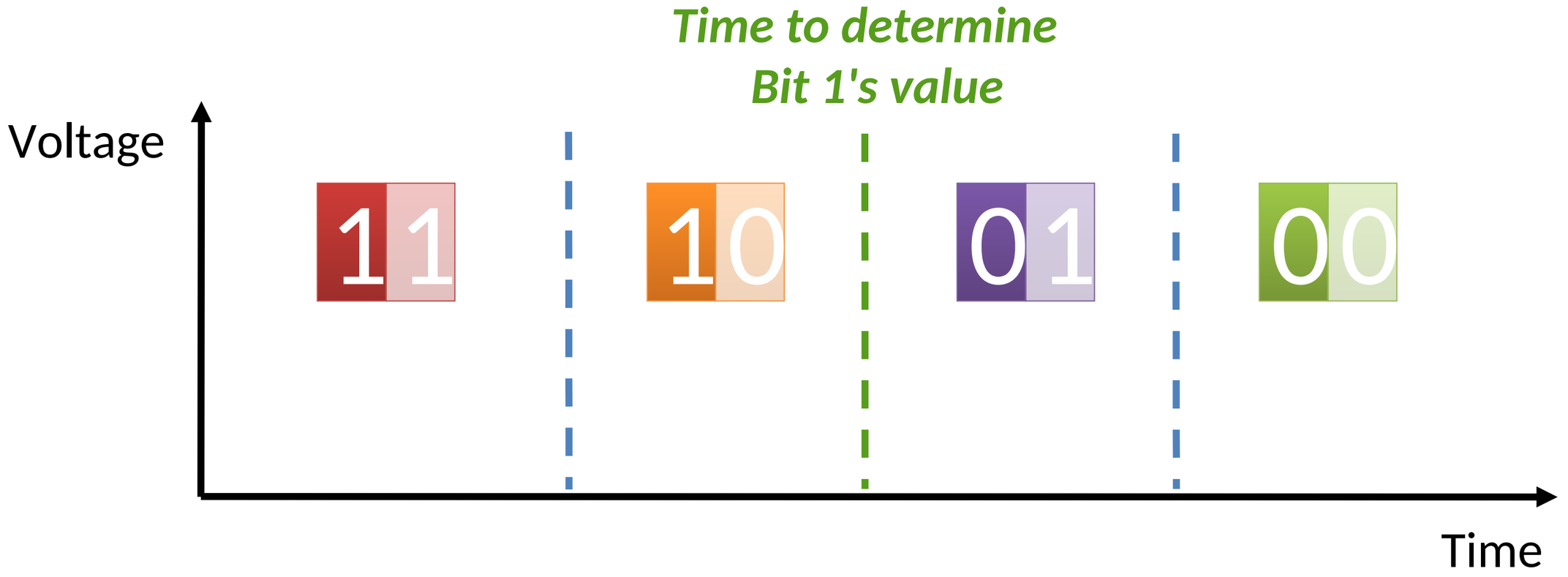
Observation 1: Read Asymmetry



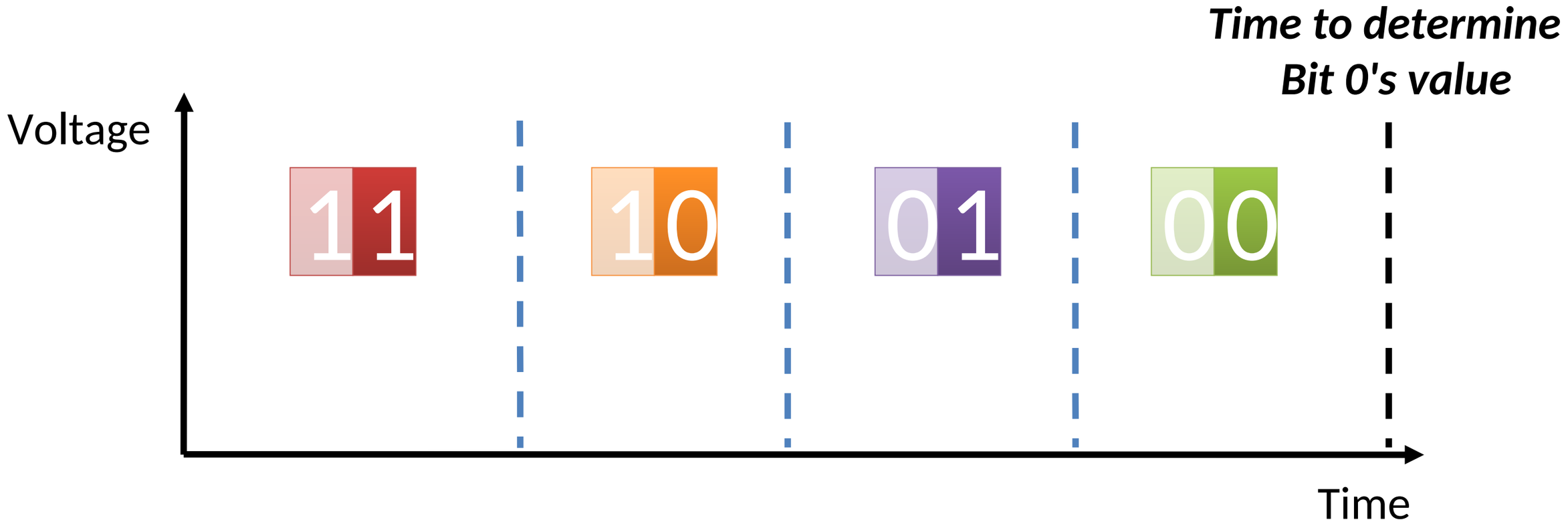
Observation 1: Read Asymmetry



Observation 1: Read Asymmetry



Observation 1: Read Asymmetry



Observation 2: Write Asymmetry

- ***The write latency/energy of Bit 0 is lower than that of Bit 1***

- This is due to how PCM cells are written

- In PCM, cell resistance must physically be changed
  - Requires applying different amounts of current
  - For different amounts of time

Observation 2: Write Asymmetry

- Writing both bits in an MLC cell: ***250ns***
- Only writing Bit 0: ***210ns***
- Only writing Bit 1: ***250ns***

- Existing devices write both bits simultaneously (250ns)

Key Observation Summary

- Bit 1 is faster to ***read*** than Bit 0
- Bit 0 is faster to ***write*** than Bit 1

- We refer to Bit 1 as the *fast-read*/*slow-write* bit (FR)
- We refer to Bit 0 as the *slow-read*/*fast-write* bit (FW)

- We leverage read/write asymmetry to enable several optimizations

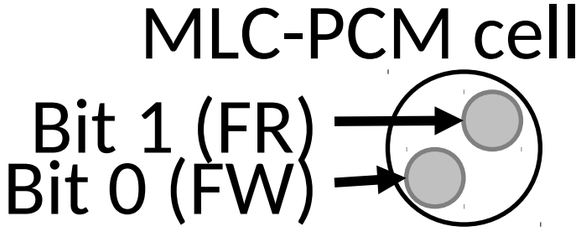
Outline

- Background
- Problem and Goal
- Key Observations
  - MLC-PCM cell read asymmetry
  - MLC-PCM cell write asymmetry
- Our Techniques
  - Decoupled Bit Mapping (DBM)
  - Asymmetric Page Mapping (APM)
  - Split Row Buffering (SRB)
- Results
- Conclusions

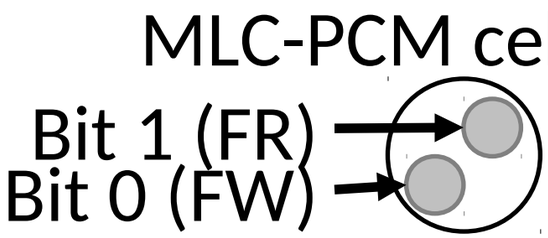
Technique 1: Decoupled Bit Mapping (DBM)

- Key Idea: Logically ***decouple*** FR bits from FW bits
  - Expose FR bits as low-read-latency regions of memory
  - Expose FW bits as low-write-latency regions of memory

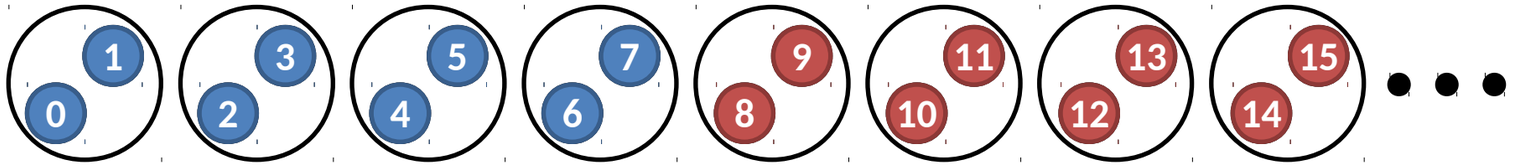
Technique 1: Decoupled Bit Mapping (DBM)



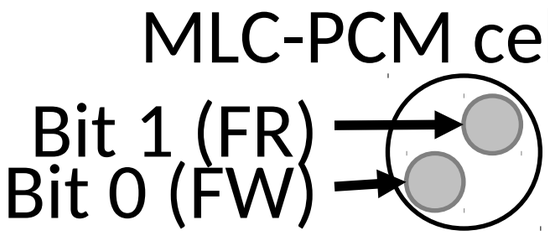
Technique 1: Decoupled Bit Mapping (DBM)



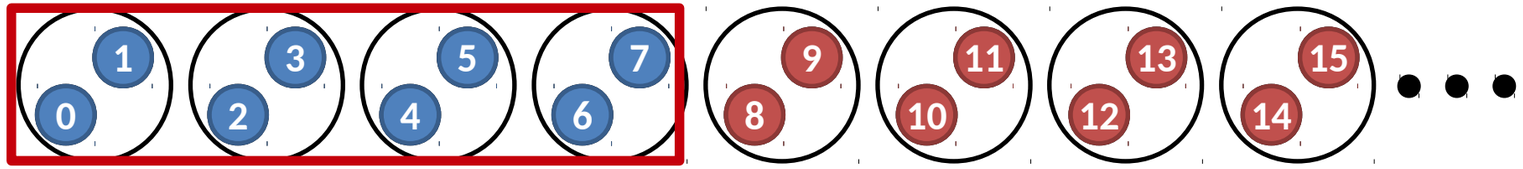
Coupled (baseline): Contiguous ***bits*** alternate between FR and FW



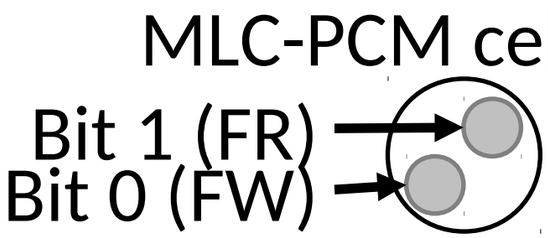
Technique 1: Decoupled Bit Mapping (DBM)



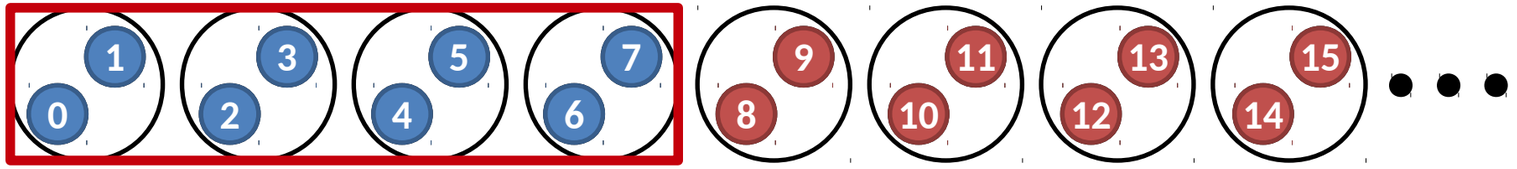
Coupled (baseline): Contiguous ***bits*** alternate between FR and FW



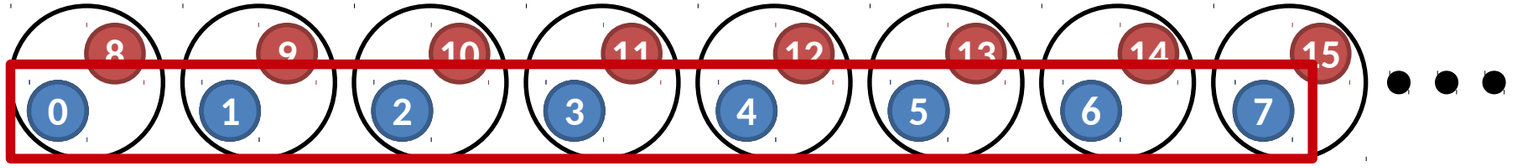
Technique 1: Decoupled Bit Mapping (DBM)



Coupled (baseline): Contiguous ***bits*** alternate between FR and FW



Decoupled: Contiguous ***regions*** alternate between FR and FW



Technique 1: Decoupled Bit Mapping (DBM)

- By decoupling, we've created regions with distinct characteristics
  - We examine the use of 4KB regions (e.g., OS page size)

Fast read page ***Fast write page***

- Want to match *frequently read data to FR pages* and vice versa
- Toward this end, we propose a new ***OS page allocation*** scheme

Technique 2: Asymmetric Page Mapping (APM)

- Key Idea: ***predict*** page read/write intensity and ***map*** accordingly
  - Measure write intensity of instructions that access data
  - If instruction has high write intensity and first touches page
    - »OS allocates FW page, otherwise, allocates FR page
- Implementation (full details in paper)
  - Small hardware cache of instructions that often write data
  - Updated by cache controller when data written to memory
  - New instruction for OS to query table for prediction

Technique 3: Split Row Buffering (SRB)

- *Row buffer* stores contents of currently-accessed data
  - Used to buffer data when sending/receiving across I/O ports
- Key Idea: With DBM, buffer FR bits ***independently*** from FW bits
  - Coupled (baseline): must use large ***monolithic*** row buffer (8KB)
  - DBM: can use two smaller ***associative*** row buffers (2x4KB)
  - Can improve row buffer locality, reducing latency and energy
- Implementation (full details in paper)
  - No additional SRAM buffer storage
  - Requires multiplexer logic for selecting FR/FW buffers

Outline

- Background
- Problem and Goal
- Key Observations
  - MLC-PCM cell read asymmetry
  - MLC-PCM cell write asymmetry
- Our Techniques
  - Decoupled Bit Mapping (DBM)
  - Asymmetric Page Mapping (APM)
  - Split Row Buffering (SRB)
- Results
- Conclusions

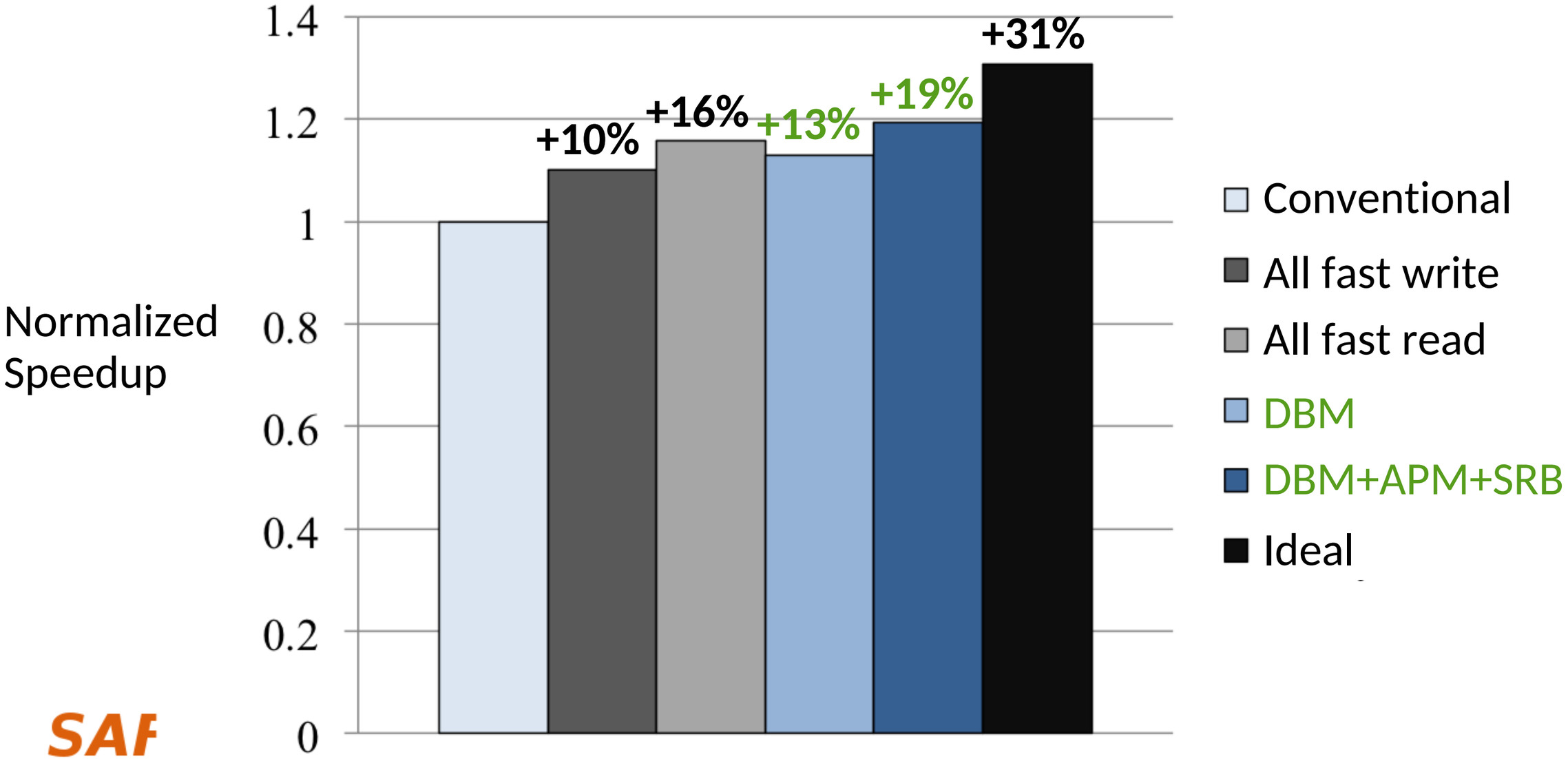
Evaluation Methodology

- Cycle-level x86 CPU-memory simulator
  - CPU: 8 cores, 32KB private L1/512KB private L2 per core
  - Shared L3: 16MB on-chip eDRAM
  - Memory: MLC-PCM, dual channel DDR3 1066MT/s, 2 ranks
- Workloads
  - SPEC CPU2006, NASA parallel benchmarks, GraphLab
- Performance metrics
  - Multi-programmed (SPEC): weighted speedup
  - Multi-threaded (NPB, GraphLab): execution time

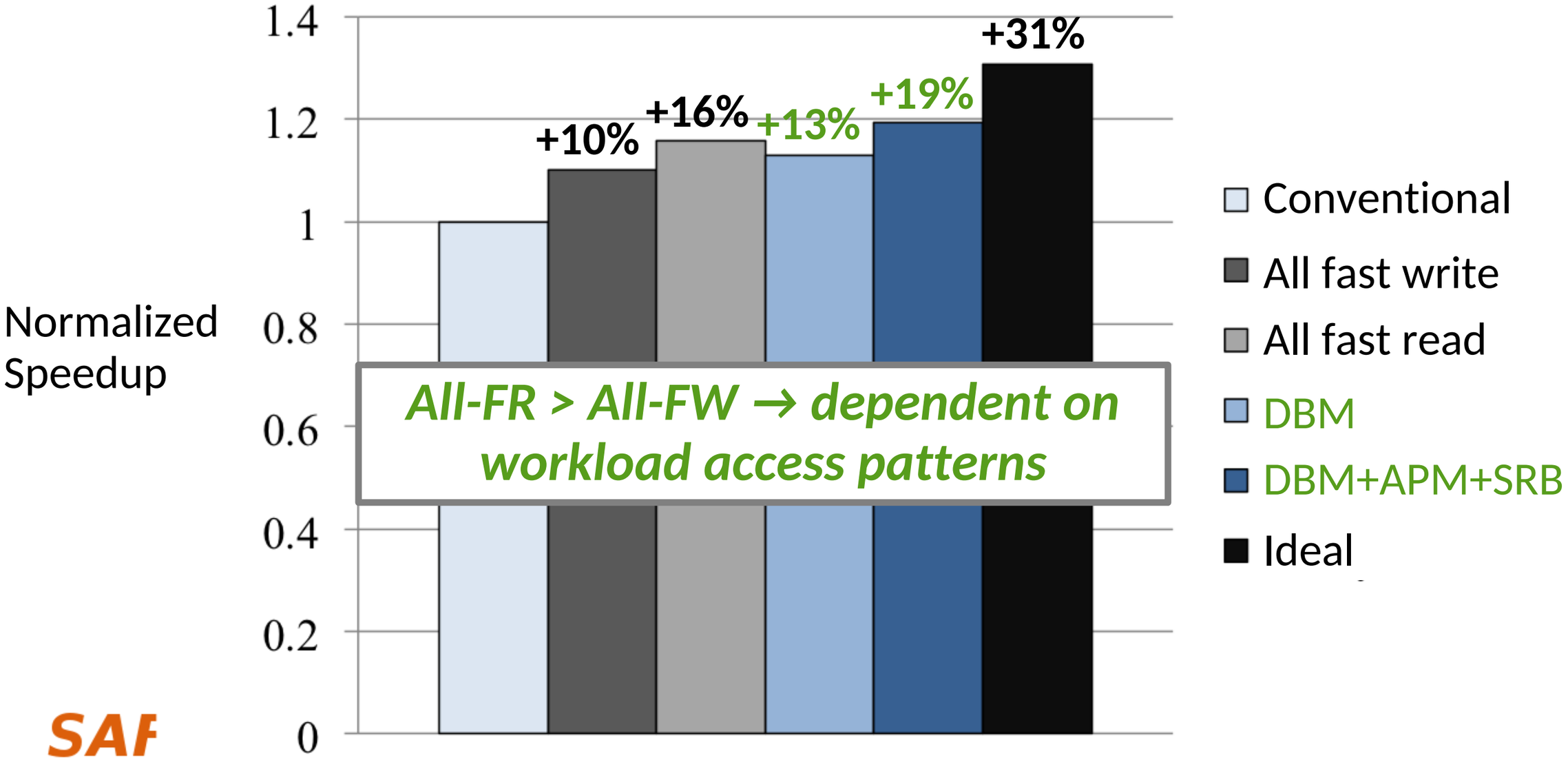
Comparison Points

- ***Conventional***: coupled bits (slow read, slow write)
- ***All-FW***: hypothetical all-FW memory (slow read, fast write)
- ***All-FR***: hypothetical all-FR memory (fast read, slow write)
- ***DBM***: decouples bit mapping (50% FR pages, 50% FW pages)
- ***DBM+***: techniques that leverage DBM (APM and SRB)
- ***Ideal***: idealized cells with best characteristics (fast read, fast write)

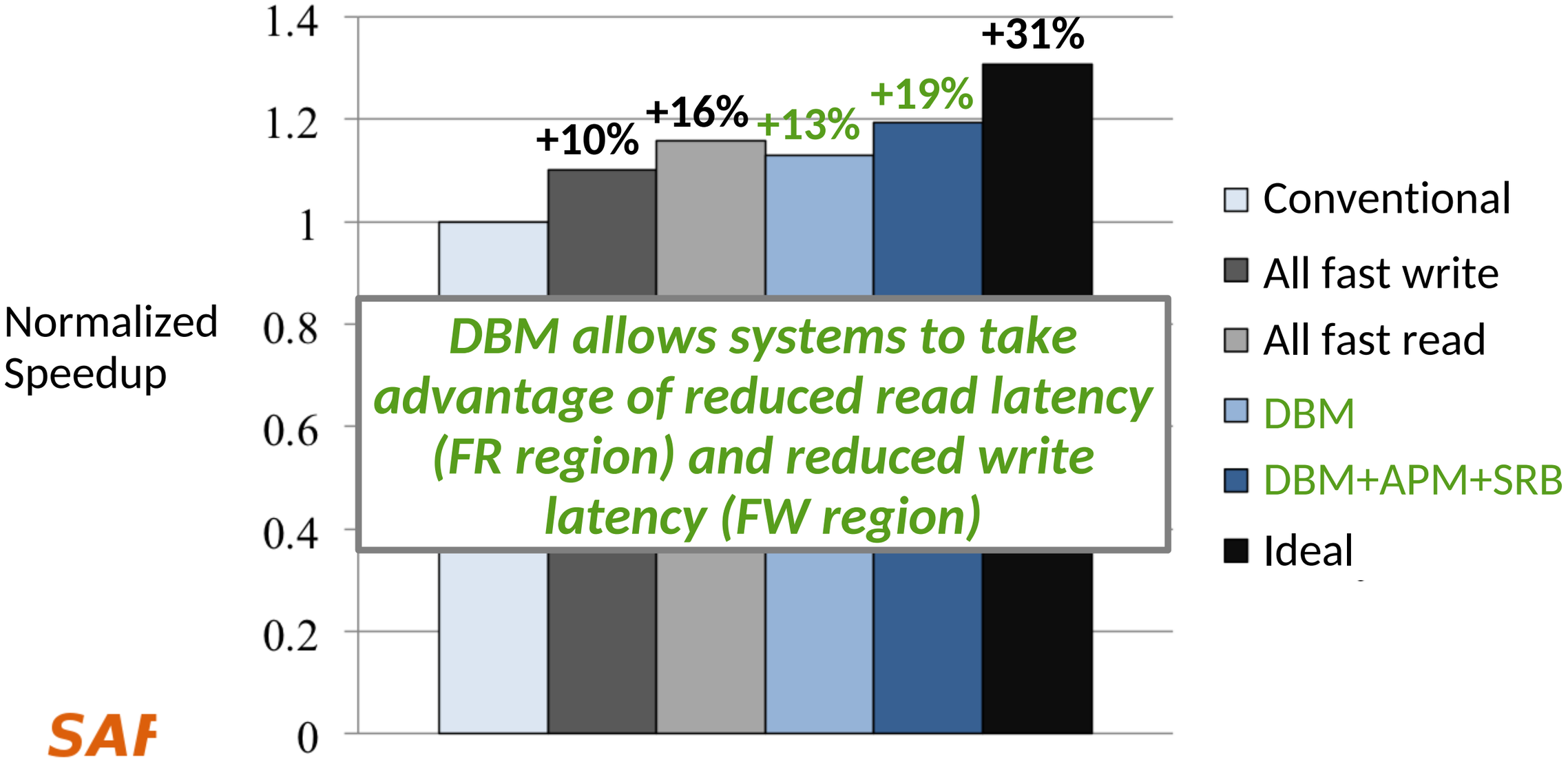
System Performance



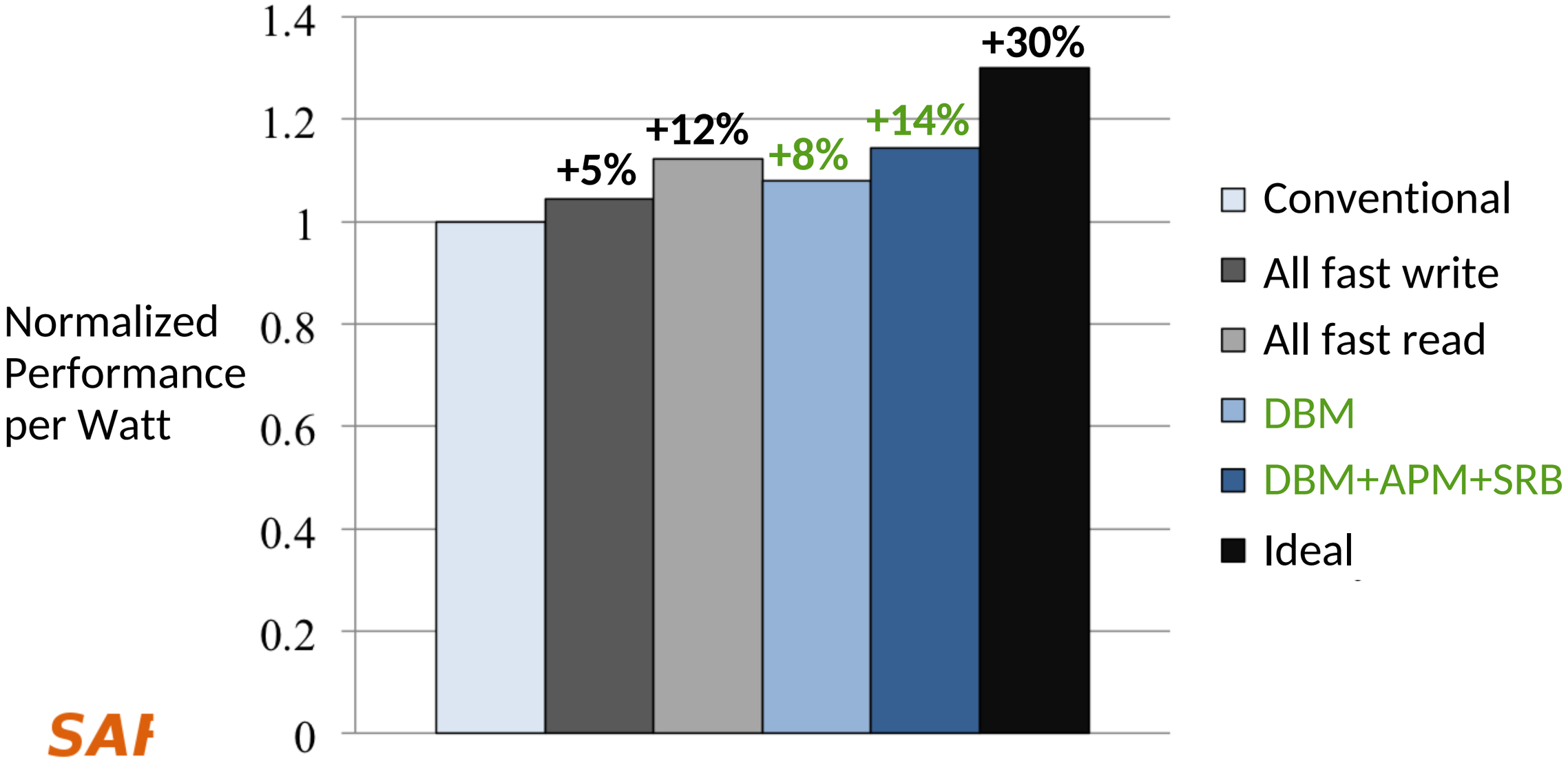
System Performance



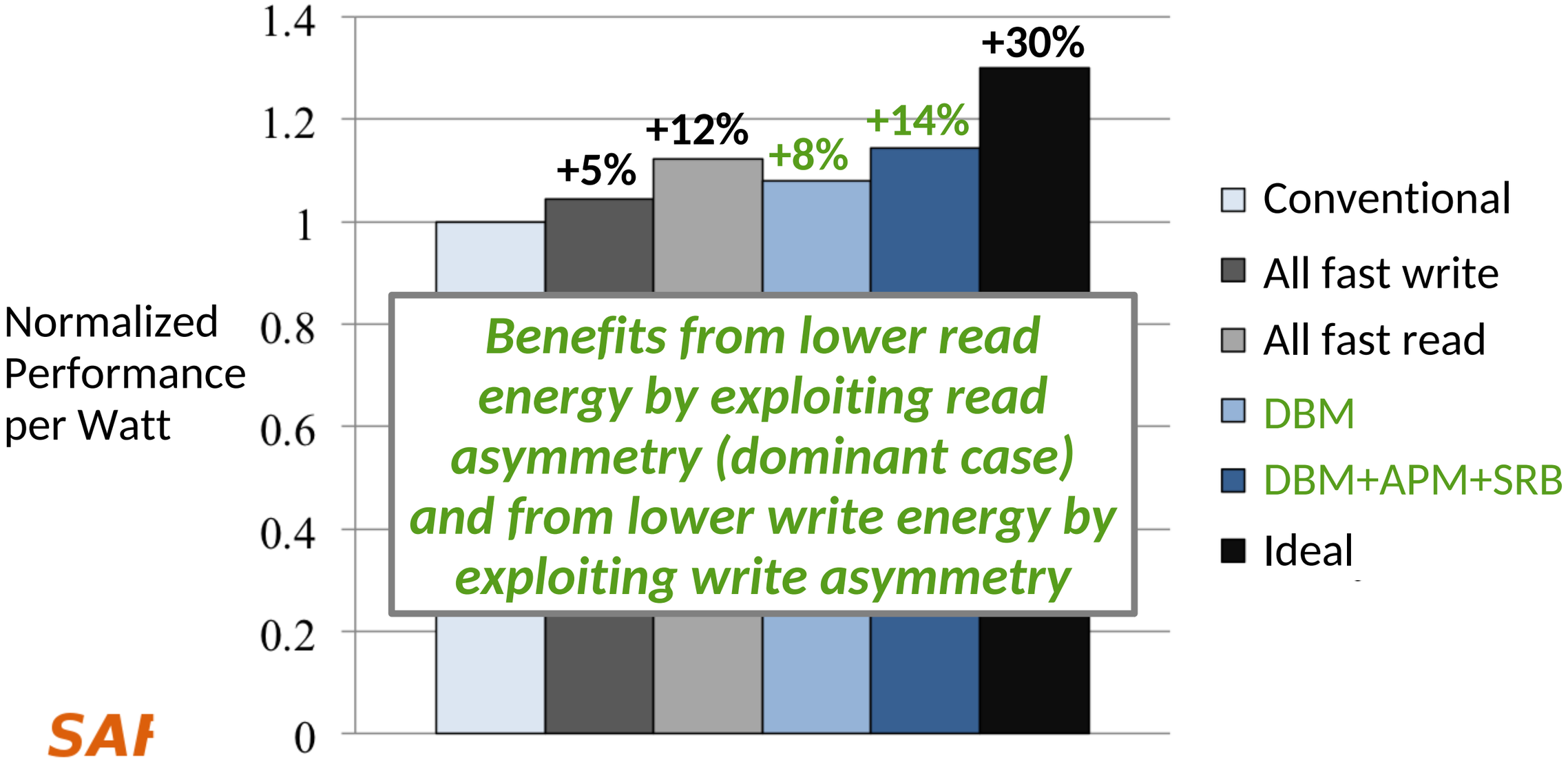
System Performance



Memory Energy Efficiency



Memory Energy Efficiency



Other Results in the Paper

- Improved thread fairness (less resource contention)
  - From speeding up per-thread execution
- Techniques do not exacerbate PCM wearout problem
  - ~6 year operational lifetime possible

Outline

- Background
- Problem and Goal
- Key Observations
  - MLC-PCM cell read asymmetry
  - MLC-PCM cell write asymmetry
- Our Techniques
  - Decoupled Bit Mapping (DBM)
  - Asymmetric Page Mapping (APM)
  - Split Row Buffering (SRB)
- Results
- Conclusions

Conclusions

- Phase-change memory (PCM) is a promising emerging technology
  - More scalable than DRAM, faster than flash
  - Multi-level cell (MLC) PCM = multiple bits per cell → high density
- Problem: Higher latency/energy compared to non-MLC PCM
- Observation: MLC bits have ***asymmetric*** read/write characteristics
  - Some bits can be ***read quickly*** but ***written slowly*** and vice versa

Conclusions

- Goal: Read data from ***fast-read*** bits; write data to ***fast-write*** bits
- Solution:
  - ***Decouple*** bits to expose fast-read/write memory regions
  - ***Map*** read/write-intensive data to appropriate memory regions
  - ***Split*** device row buffers to leverage decoupling for better locality
- Result:
  - Improved performance (+19.2%) and energy efficiency (+14.4%)
  - Across SPEC CPU2006 and data-intensive/cloud workloads

Thank You!

Efficient Data Mapping and Buffering Techniques for Multi-Level Cell Phase-Change Memories

HanBin Yoon, **Justin Meza**,
Naveen Muralimanohar*, Onur Mutlu, Norm Jouppi*†

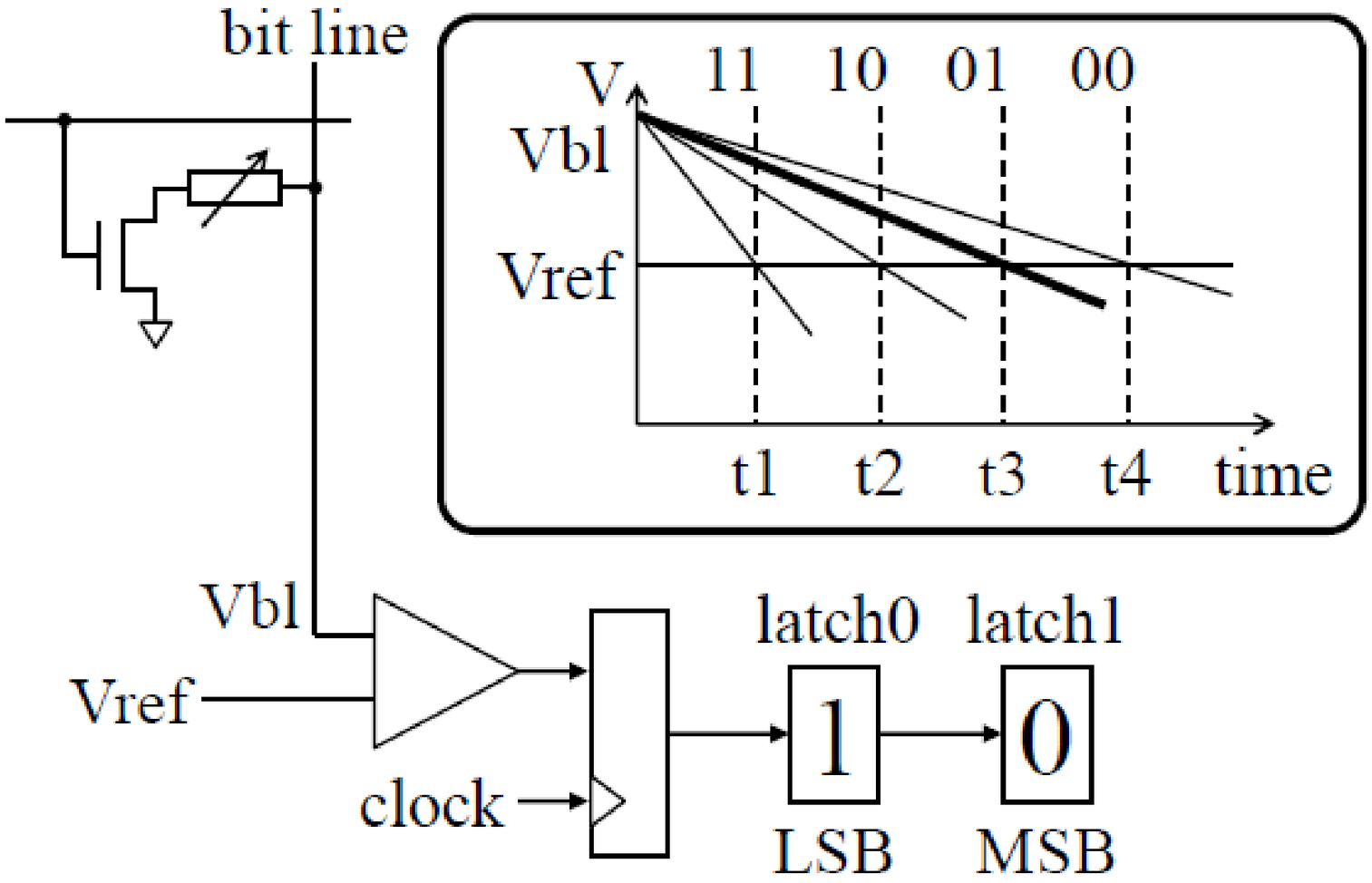
Carnegie Mellon University * Hewlett-Packard Labs † Google, Inc.

Carnegie Mellon University
SAFARI

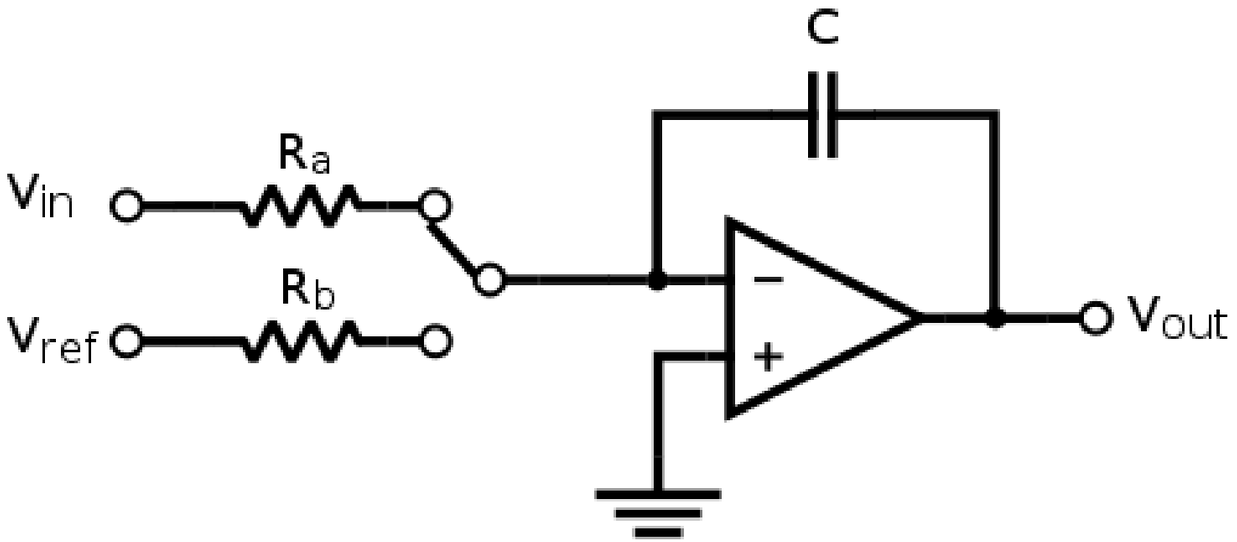
Google

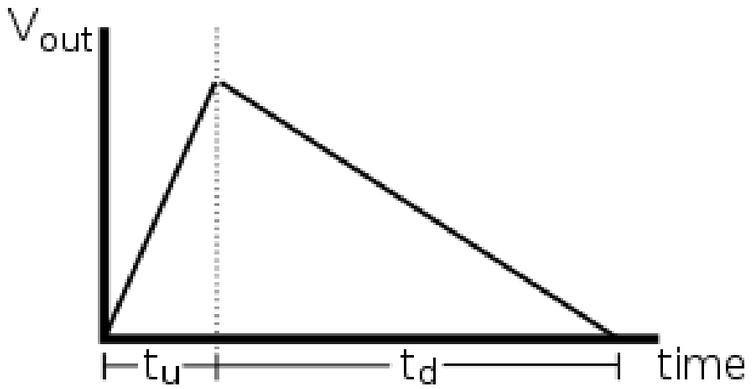
Backup Slides

PCM Cell Operation



Integrating ADC





APM Implementation