Enabling and Exploiting Partition-Level Parallelism (PALP) in Phase Change Memories

Presenter: Dr. Anup K. Das

Shihao Song, Anup Das, Drexel University
Onur Mutlu, ETH Zurich
Nagarajan Kandasamy, Drexel University

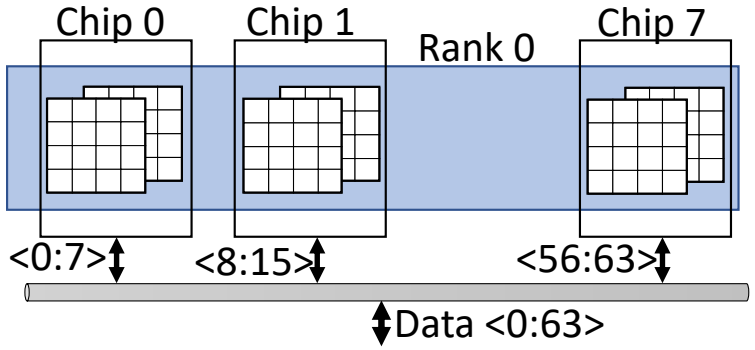
CASES 2019, New York

Executive Summary

- **Observations**
  - Memory bank conflicts reduce system performance by lowering bank utilization, causing CPU cores to stall
  - A PCM bank's peripheral structures allow to read and program 128 PCM cells in parallel, providing opportunity to resolve bank conflicts
- **Idea: PArtition-Level Parallelism (PALP) in PCM**
  - Introduce new mechanism to enable read-write parallelism in PCM banks with minimum changes to PCM interface and its timing
  - Introduce simple circuit modifications to enable read-read parallelism in PCM banks
  - Propose new access scheduling mechanism to exploit read-write and read-read parallelism in PCM banks
- **Design Updates: Analyze internal architecture of PCM banks**
  - New PCM commands
  - New decoupled write driver design
- **Performance Evaluation**
  - Significant bank conflict reduction (28% average system performance improvement for SPEC CPU 2017 and MiBench workloads)

PCM Bank Conflicts

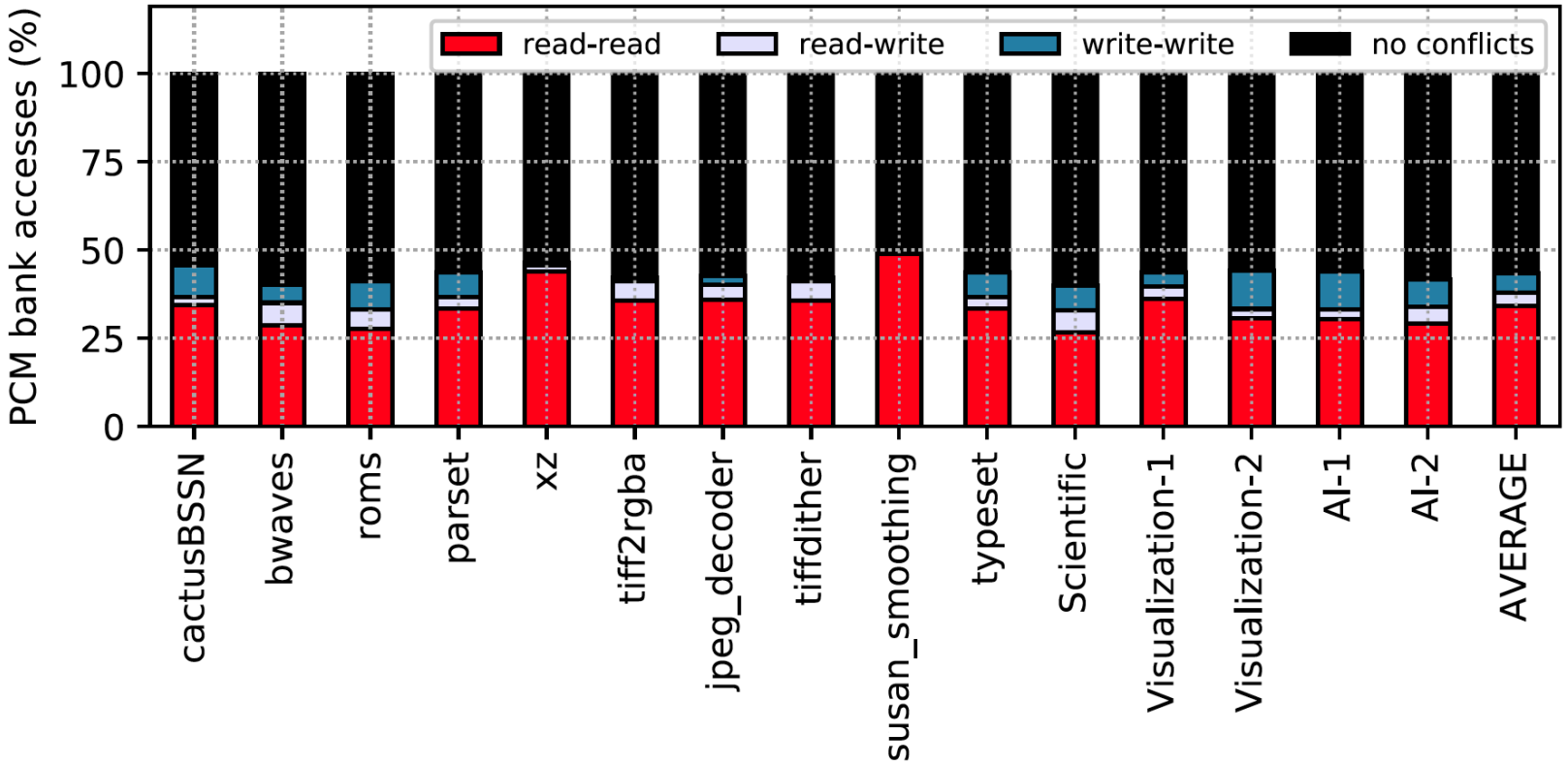
- **PCM can serve requests *in parallel* using multiple banks**



- **Request to the same bank must be served *serially***
  - Called Bank Conflicts
- **Bank conflicts reduce performance by lowering bank utilization, causing CPU cores to stall**
- **Our goal: improve performance by resolving bank conflicts in PCM devices**

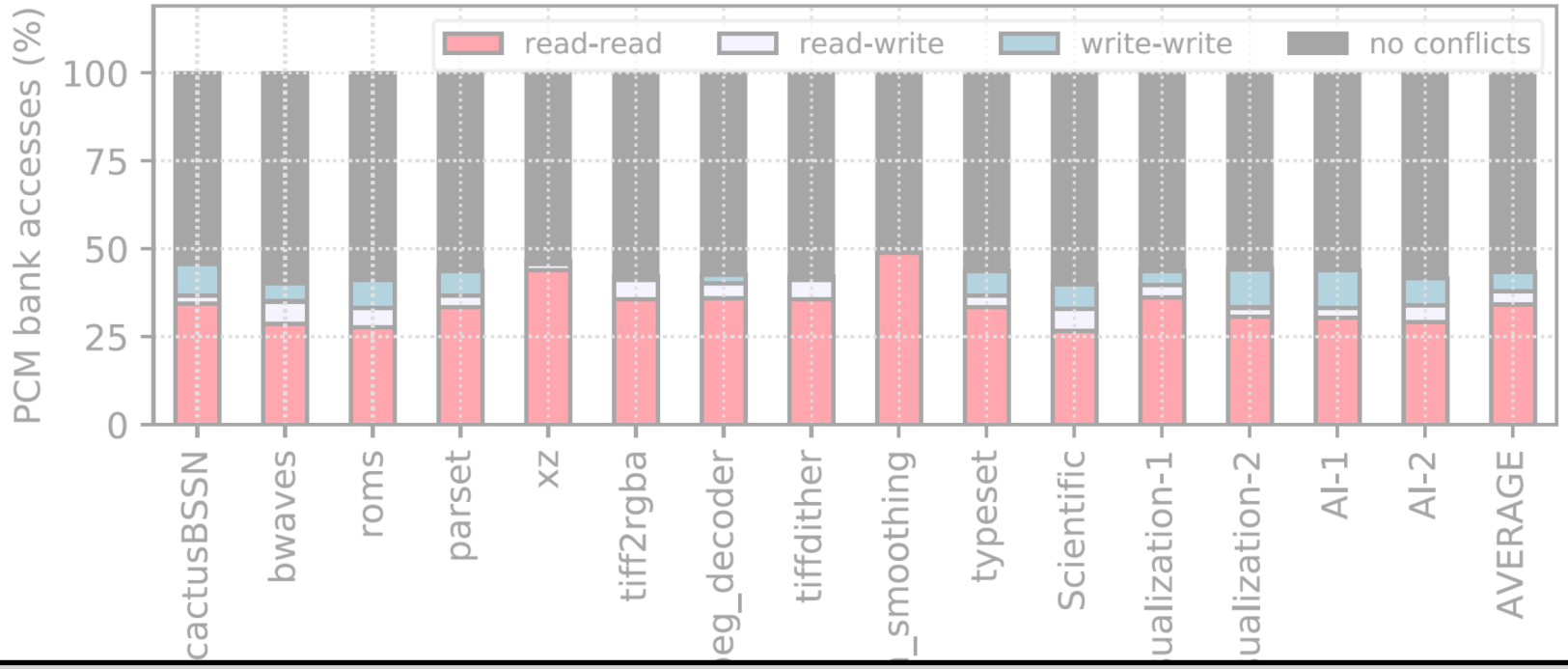
Bank Conflict Related Key Observation

- **Bank conflicts are due to the high temporal and spatial access locality in workloads that lead to repeated access to multiple rows that map to the same bank**



Bank Conflict Related Key Observations

- **Bank conflicts are due to the high temporal and spatial access locality in workloads that lead to repeated access to multiple rows that map to the same bank**



- On average, 43% of PCM requests in these workloads generate bank conflicts
- Read-read bank conflicts outnumber read-write and write-write bank conflicts for all workloads (averaging 79% of all bank conflicts)

Bank Conflict Related Key Observations

- **Bank conflicts are due to the high temporal and spatial access locality in workloads that lead to repeated access to multiple rows that map to the same bank**

Idea: Enable parallelism in PCM banks to resolve read-read bank conflicts first, and then other conflicts (if possible)

- On average, 43% of PCM requests in these workloads generate bank conflicts
- Read-read bank conflicts outnumber read-write and write-write bank conflicts for all workloads (averaging 79% of all bank conflicts)

Bank Conflicts Resolution Techniques

Mechanism	Read-Read Conflicts	Read-Write Conflicts	Write-Write Conflicts
Baseline	No	No	No
MultPartition	No	Yes	No
SALP (DRAM)	Yes	Yes	Yes
PALP (PCM)	Yes	Yes	No

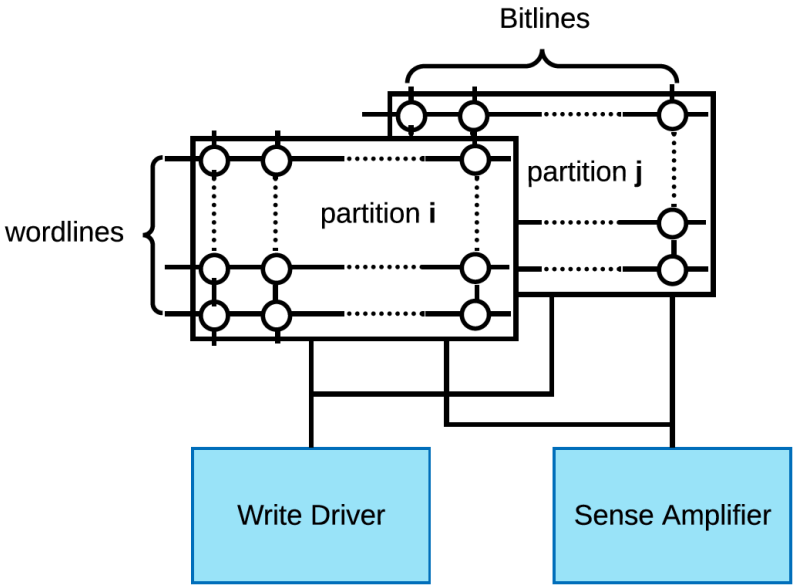
Baseline: Arjomand et al. "*Boosting access parallelism to PCM-based main memory*" in ISCA 2016
MultiPartition: Zhou et al., "An efficient parallel scheduling scheme on multi-partition PCM architecture" in ISLPED 2016
SALP: Kim et al., "A Case for Exploiting Subarray-Level Parallelism (SALP) in DRAM" in ISCA 2012

Outline

- Introduction
- Motivation
- **Background on PCM**
- **Contribution 1: Enabling read-write parallelism**
- **Contribution 2: Enabling read-read parallelism**
- **Contribution 3: Exploiting parallelism**
- **Evaluation**
- **Conclusion**

PCM Bank Structure

- **A PCM bank is implemented as a collection of partitions that operate mostly independently; sharing the sense amplifiers (to read) and the write drivers (to write)**



Peripheral structures in PCM banks allow to read and program 128 PCM cells in parallel

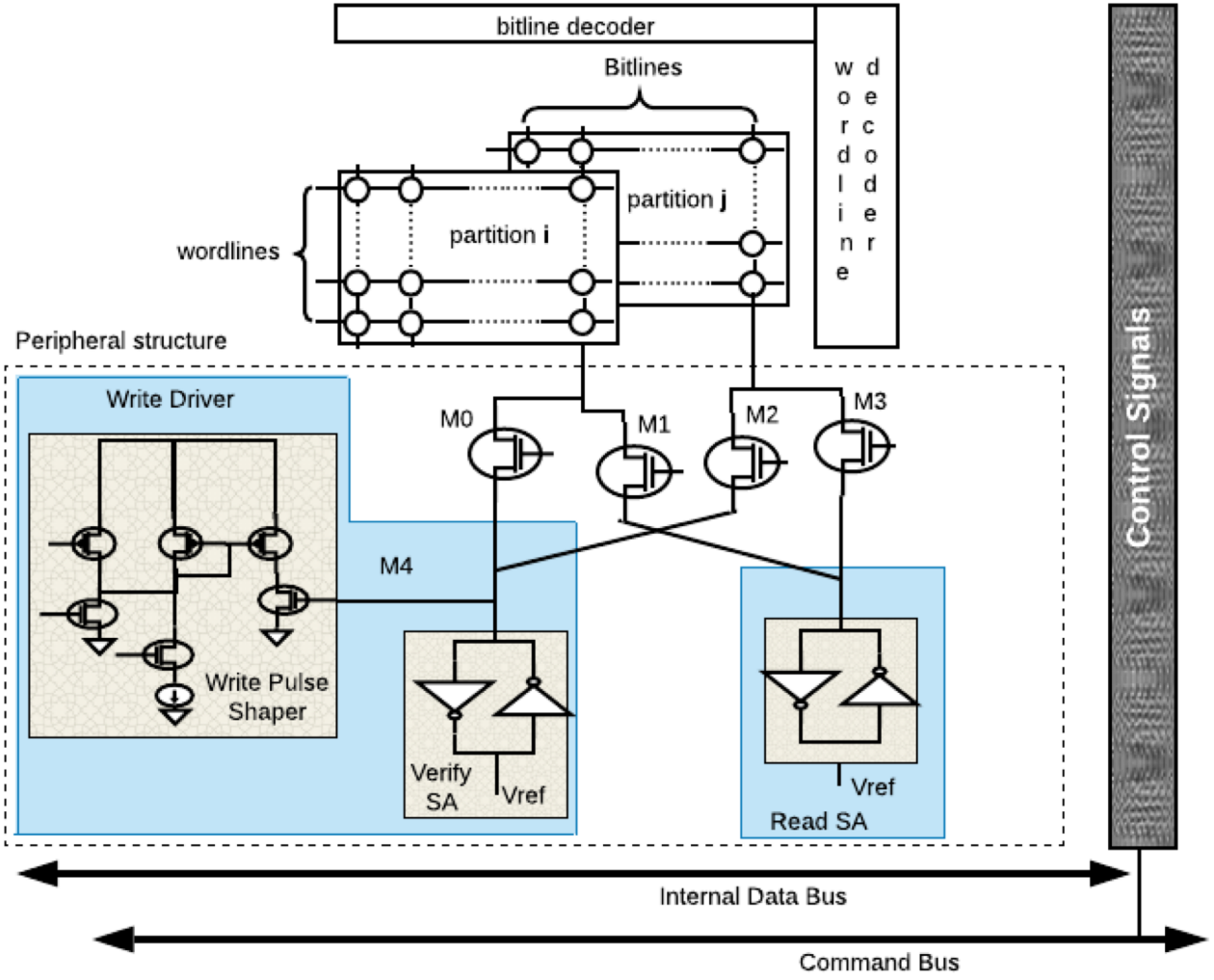
Baseline PCM Commands

- ACTIVATE(A): activate the wordline and enable the access device for the PCM cells to be accessed

- READ(R)/WRITE(W): drive read or write current through the PCM cell. After this command executes, the data stored in the PCM cell is available at the output terminal of the sense amplifier, or the write data is programmed to the PCM cell

- PRECHARGE(P): deactivate the wordline and bitline, and prepare the bank for the next access

Internal Architecture of PCM Banks



Transistor Configuration for Read/Write

Operation	M0	M1	M2	M3
write to partition i	ON	OFF	OFF	OFF
read from partition i	OFF	ON	OFF	OFF
write to partition j	OFF	OFF	ON	OFF
read from partition j	OFF	OFF	OFF	ON

Transistor Configuration for Read/Write

Operation	M0	M1	M2	M3
write to partition i	ON	OFF	OFF	OFF
read from partition i	OFF	ON	OFF	OFF
write to partition j	OFF	OFF	ON	OFF
read from partition j	OFF	OFF	OFF	ON

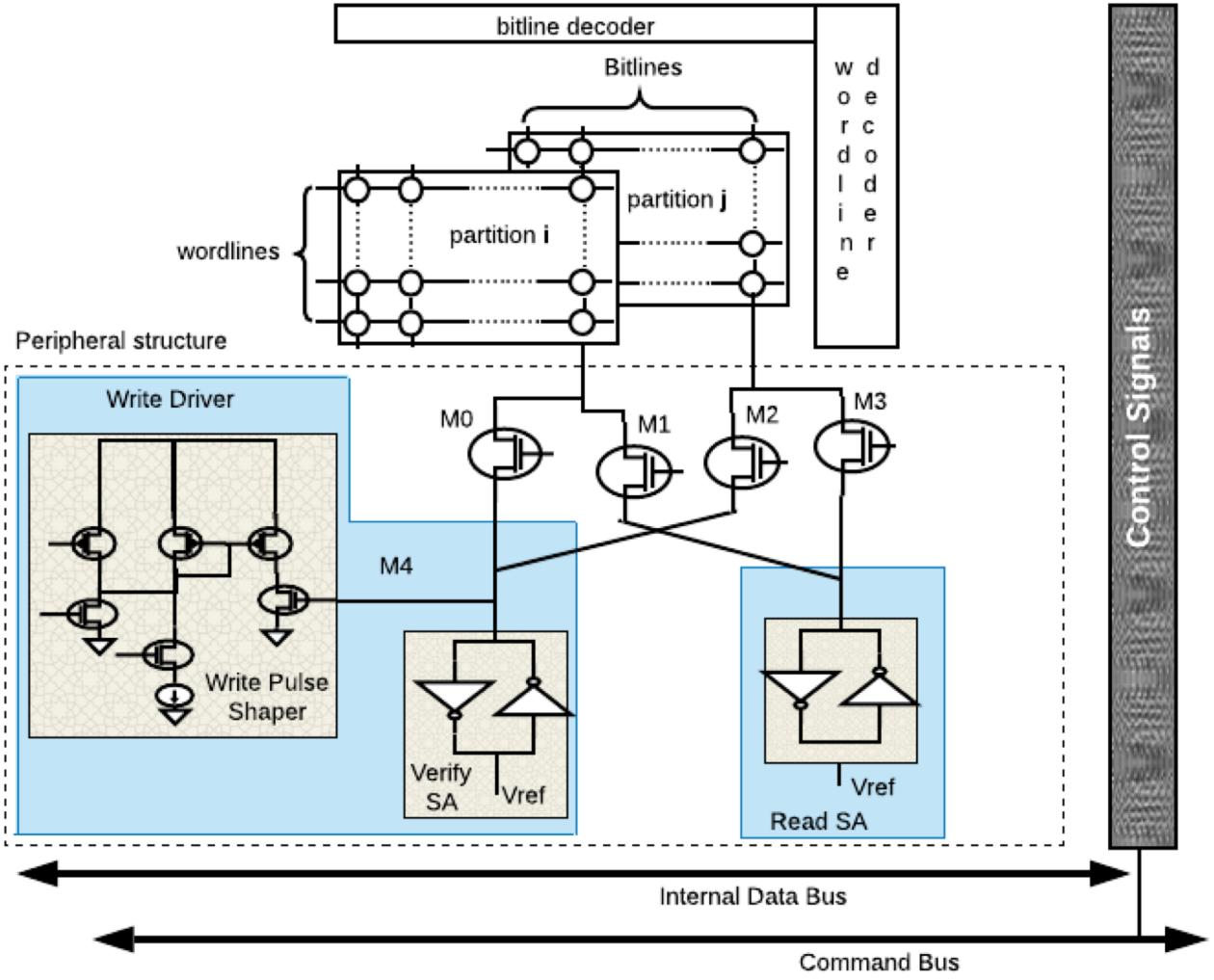
Observation: Only one transistor ON at once for any given operation

Idea: Make two transistors ON to do parallel operations

Outline

- Introduction
- Motivation
- Background on PCM
- **Contribution 1: Enabling read-write parallelism**
- **Contribution 2: Enabling read-read parallelism**
- **Contribution 3: Exploiting parallelism**
- **Evaluation**
- **Conclusion**

Internal Architecture of PCM Banks



New Transistor Configuration

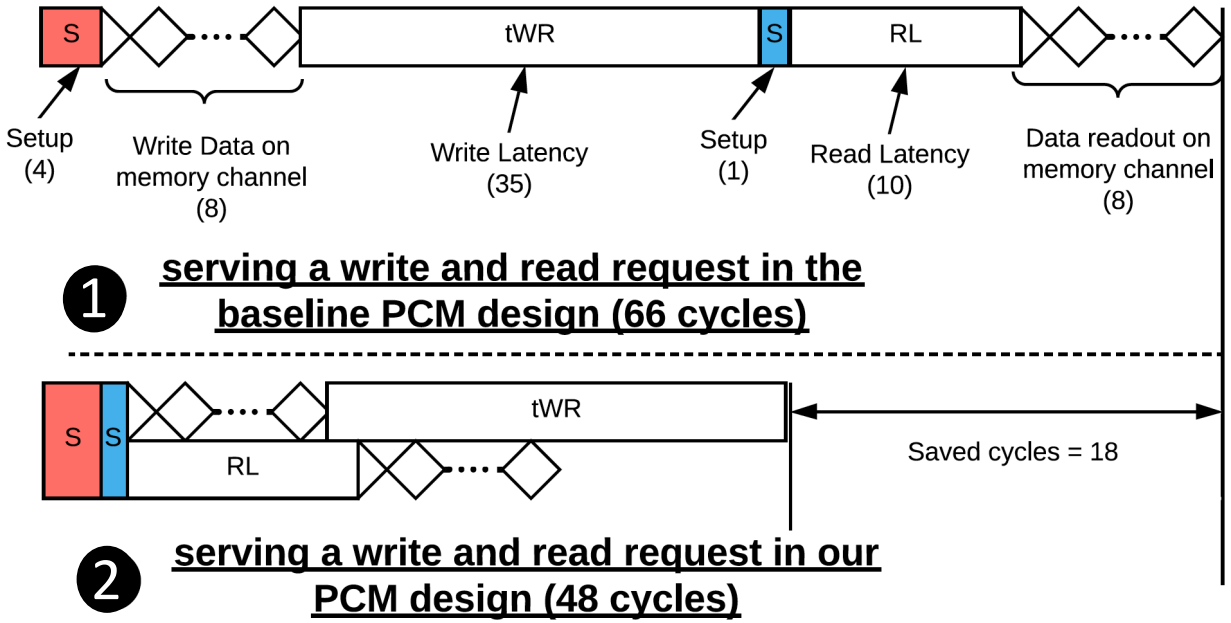
Operation	M0	M1	M2	M3
write to partition i, read from partition j	ON	OFF	OFF	ON
read from partition i, write to partition j	OFF	ON	ON	OFF
invalid	ON	ON	OFF	OFF
invalid	OFF	OFF	ON	ON
invalid	ON	OFF	ON	OFF
invalid	OFF	ON	OFF	ON

Potential to resolve read-write bank conflicts without additional hardware

PCM Interface Changes

- **New PCM command**

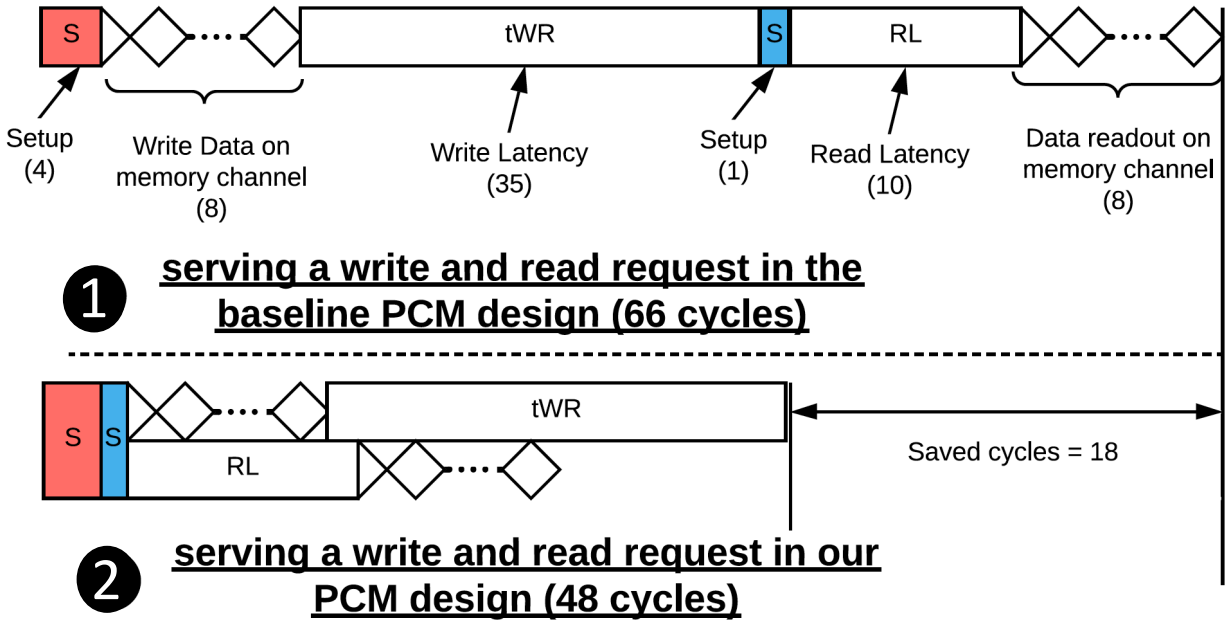
READ-WITH-WRITE (RWW): connect the PCM bank's sense amplifiers and write drivers to the two decoded partitions



PCM Interface Changes

- **New PCM command**

READ-WITH-WRITE (RWW): connect the PCM bank's sense amplifiers and write drivers to the two decoded partitions



Performance improvement 27%

PCM Commands: Baseline vs. New

- **To serve a write and a read request from two different partitions in a PCM bank**

Baseline PCM:

- ACTIVATE address in i
- WRITE
- PRECHARGE
- ACTIVATE address in j
- READ
- PRECHARGE

– Service time = 47 + 19 = 66 cycles

Proposed PCM:

- ACTIVATE address in i
- ACTIVATE address in j
- READ-WITH-WRITE
- PRECHARGE

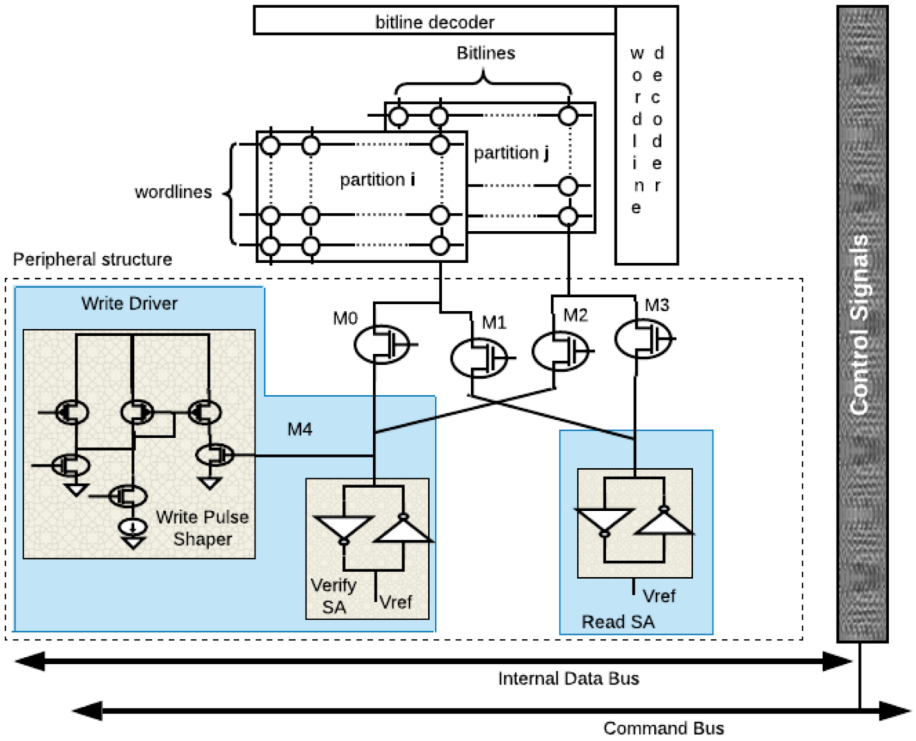
– Service time = 1 + 47 = 48 cycles

Outline

- Introduction
- Motivation
- Background on PCM
- Contribution 1: Enabling read-write parallelism
- **Contribution 2: Enabling read-read parallelism**
- **Contribution 3: Exploiting parallelism**
- **Evaluation**
- **Conclusion**

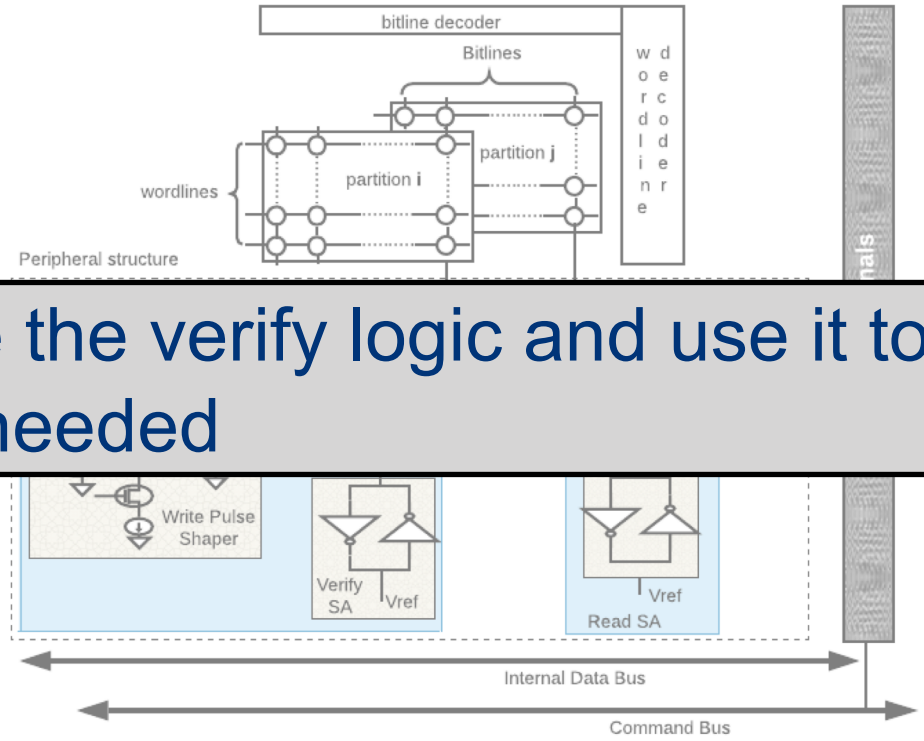
PCM bank Related Key Observation

- **Write drivers internally implement logic to verify the correctness of write operation**
  - Program & Verify Write Scheme in PCM



PCM bank Related Key Observation

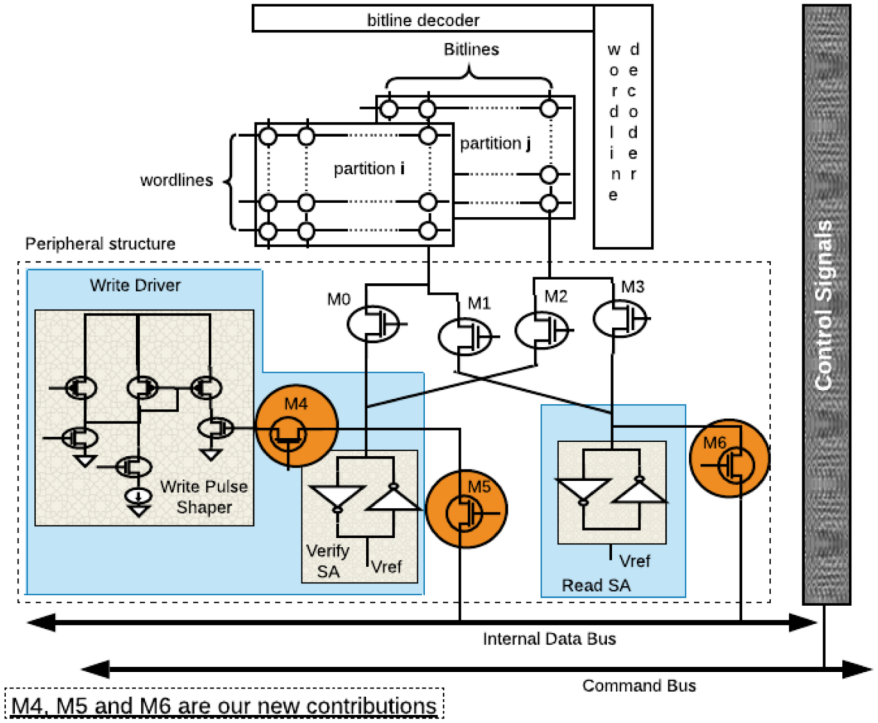
- **Write drivers internally implement logic to verify the correctness of write operation**
  - Program & Verify Write Scheme in PCM



Idea: Decouple the verify logic and use it to serve a read request when needed

PCM bank Related Key Observation

- **Write drivers internally implement logic to verify the correctness of write operation**
  - Program & Verify Write Scheme in PCM



New Operating Modes for Write Driver

- **Introduce two operating modes for the write driver**

Operating Modes	Write driver circuit		Sense Amplifier	Operations Enabled
	Pulse shaper logic	Verify logic		
write	√	√	×	one write request
decoupled	×	√	√	two read requests

- **In the decoupled mode, write driver can serve a read request and the sense amplifier can serve another read request**
  - Resolve read-read bank conflicts in PCM

PCM Interface Changes to Exploit Read-Read Parallelism

- 3 New PCM Commands

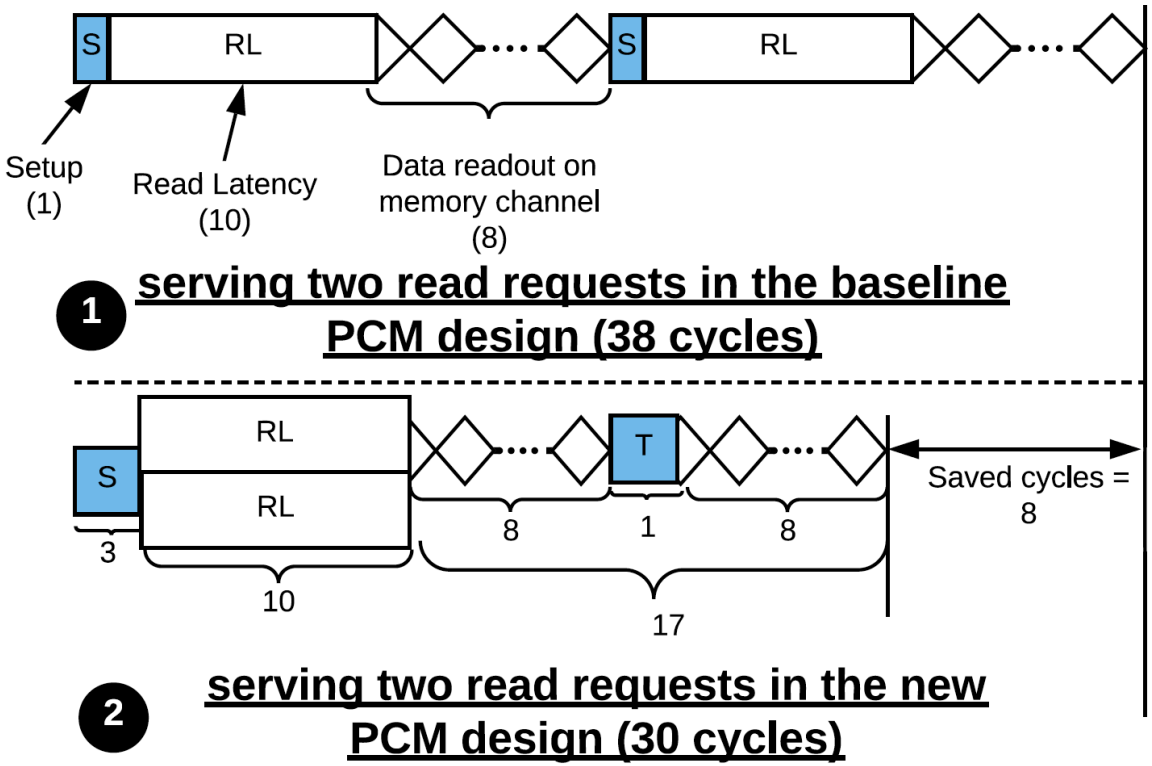
READ-WITH-READ (RWR): connect the sense amplifiers and verify logic of the write drivers to the two decoded partitions

DECOUPLE (D): set M4 = OFF, to enter in decoupled model

TRANSFER (T): sets M5 = ON and M6 = OFF to facilitate arbitration of the data bus to transfer two sets of data to the CPU

PCM Commands: Baseline vs. New

- **To serve two read requests from two different partitions in a PCM bank**



Performance improvement 21%

PCM Commands: Baseline vs. New

- **To serve two read requests from two different partitions in a PCM bank**

Baseline PCM:

- ACTIVATE address in i
- READ
- PRECHARGE
- ACTIVATE address in j
- READ
- PRECHARGE
- Service time = 19 + 19 = 38 cycles

Proposed PCM:

- ACTIVATE address in i
- ACTIVATE address in j
- DECOUPLE
- READ-WITH-READ
- TRANSFER
- PRECHARGE
- Service time = 1 + 1 + 1 + 10 + 8 + 1 + 8 = 30 cycles

Outline

- Introduction
- Motivation
- Background on PCM
- Contribution 1: Enabling read-write parallelism
- Contribution 2: Enabling read-read parallelism
- **Contribution 3: Exploiting parallelism**
- **Evaluation**
- **Conclusion**

Summary of Changes and Impact

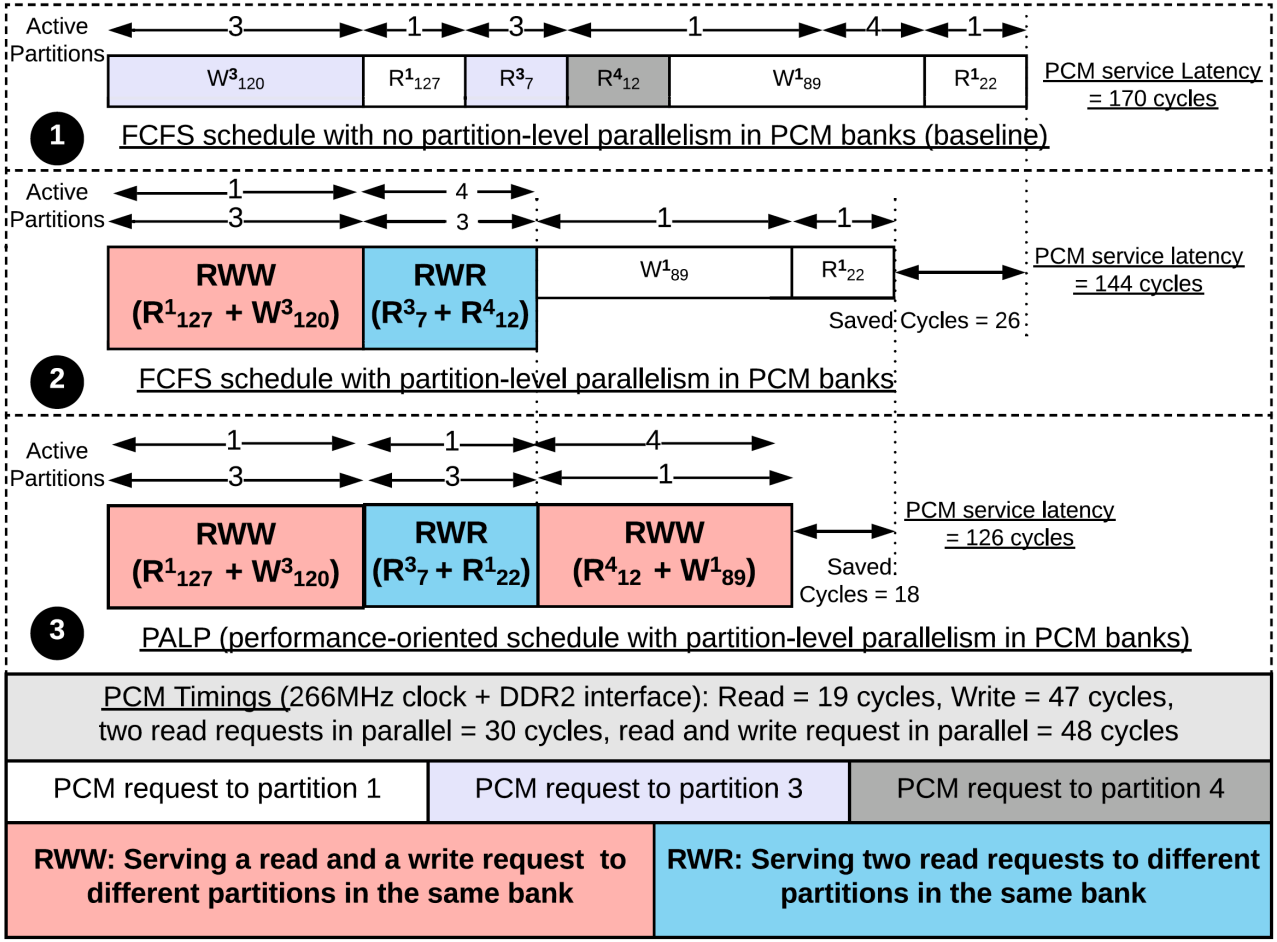
- **Design changes**
  - Four new PCM commands
  - Three new transistors per peripheral structure

- **Impact**
  - Resolve read-write bank conflicts in PCM banks when using different partitions
  - Resolve read-read bank conflicts in PCM banks when using different partitions

- **Contribution 3: Develop a new access scheduler to explicitly prioritize resolving bank conflicts**

Exploiting Partition-Level Parallelism

- **Motivation**

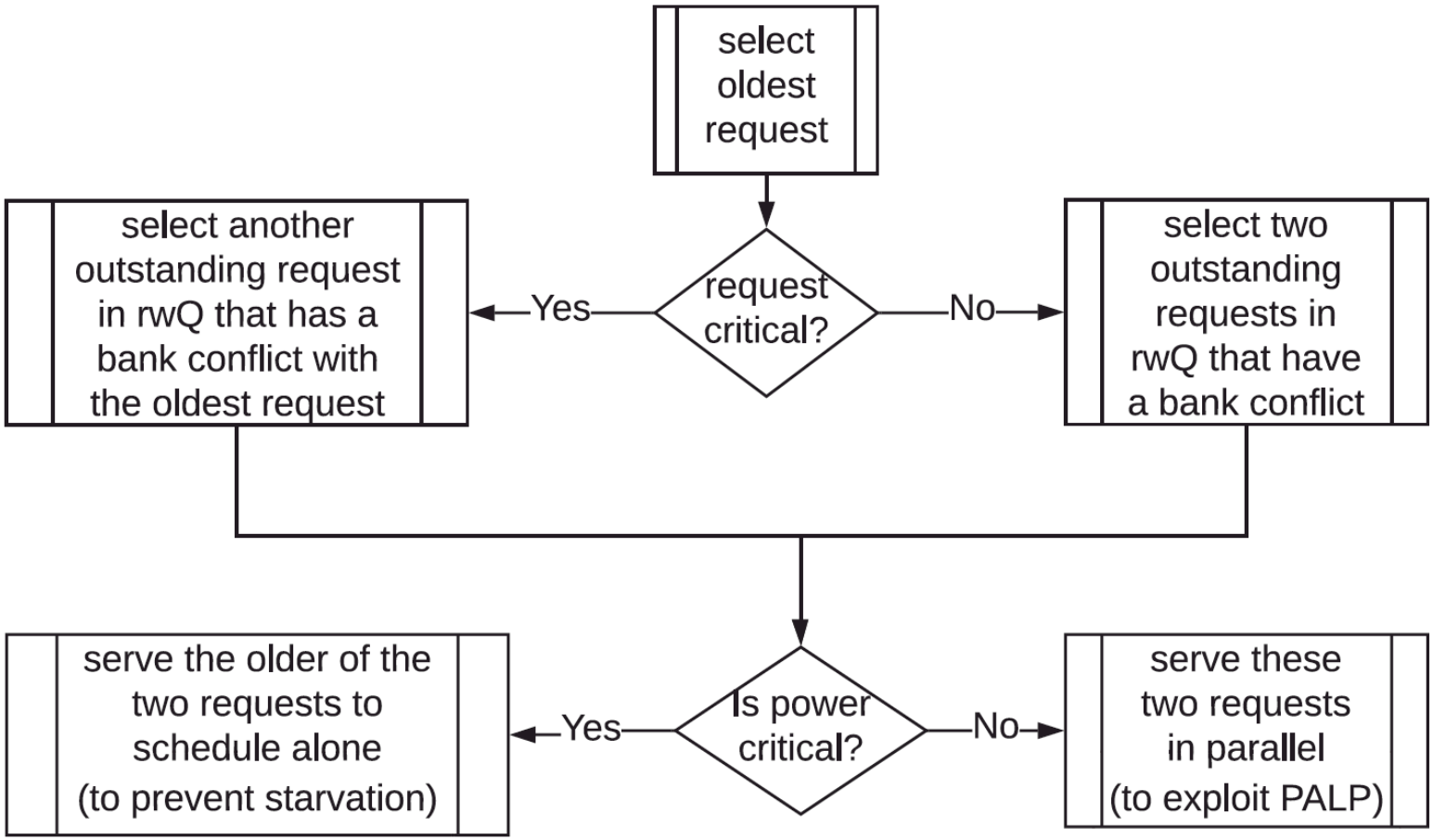


Exploiting Partition-Level Parallelism

- **Key considerations**
  - Ensures that no request is starved, i.e., backlogged excessively
    - Starvation-freedom
  - Ensure power consumption of the active partitions within the bank is not too high
    - Running average power limit (RAPL)

Exploiting Partition-Level Parallelism

- **Algorithm**



Outline

- Introduction
- Motivation
- Background on PCM
- Contribution 1: Enabling read-write parallelism
- Contribution 2: Enabling read-read parallelism
- Contribution 3: Exploiting parallelism
- **Evaluation**
- **Conclusion**

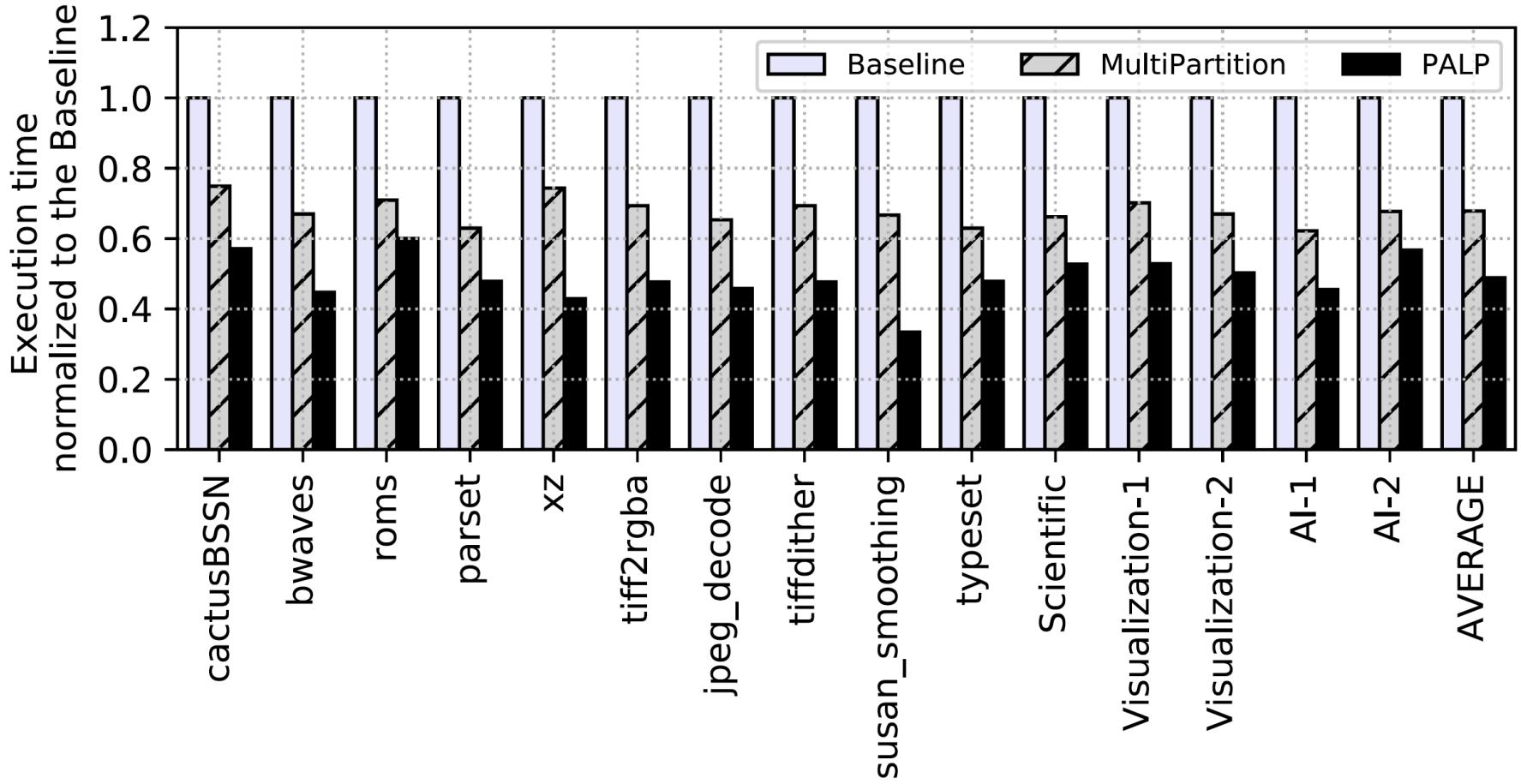
Evaluation Methodology

- **Gem5 frontend to simulate ARMv8-A (aarch64) with 8 cores**
- **Hybrid DRAM-PCM memory system**
- **In-house cycle-level PCM simulator for 8GB, 16GB, and 32GB PCM with DDR4 interface**

MiBench workloads [20]
tiff2rgba, jpeg_decode, tiffdither, susan_smoothing, typeset
SPEC CPU2017 workloads [8]
cactusBSSN, bwaves, roms, parset, xz
Mixed (parallel) applications
AI-1 (4 copies each of deepsjeng, leela), **AI-2** (4 copies each of mcf, exchange2), **Visualization-1** (4 copies each of povray, blender), **Visualization-2** (4 copies each of povray, imagick), **Scientific** (4 copies each of cactusBSSN, bwaves)

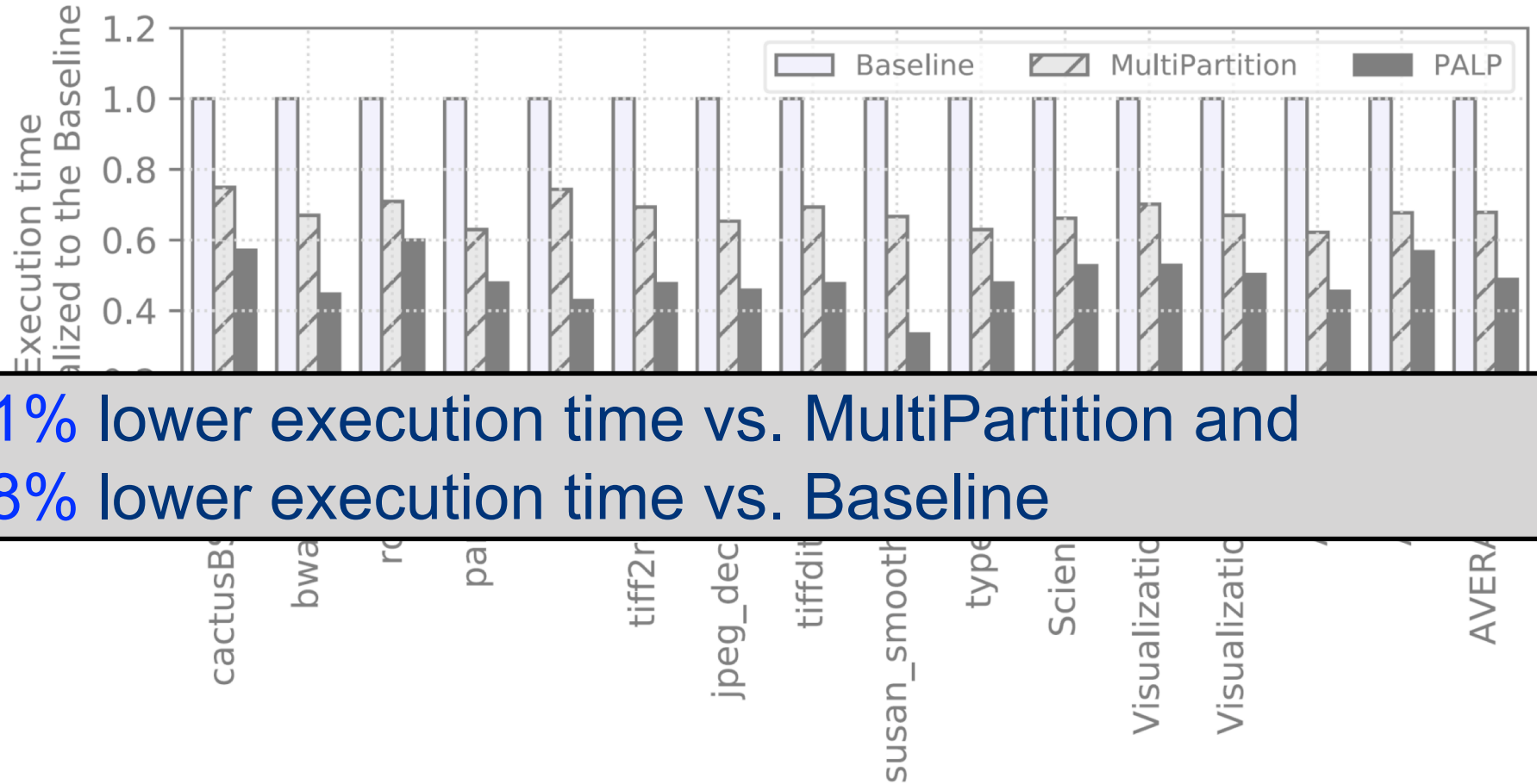
Performance Comparison

- **Execution time (lower is better)**



Performance Comparison

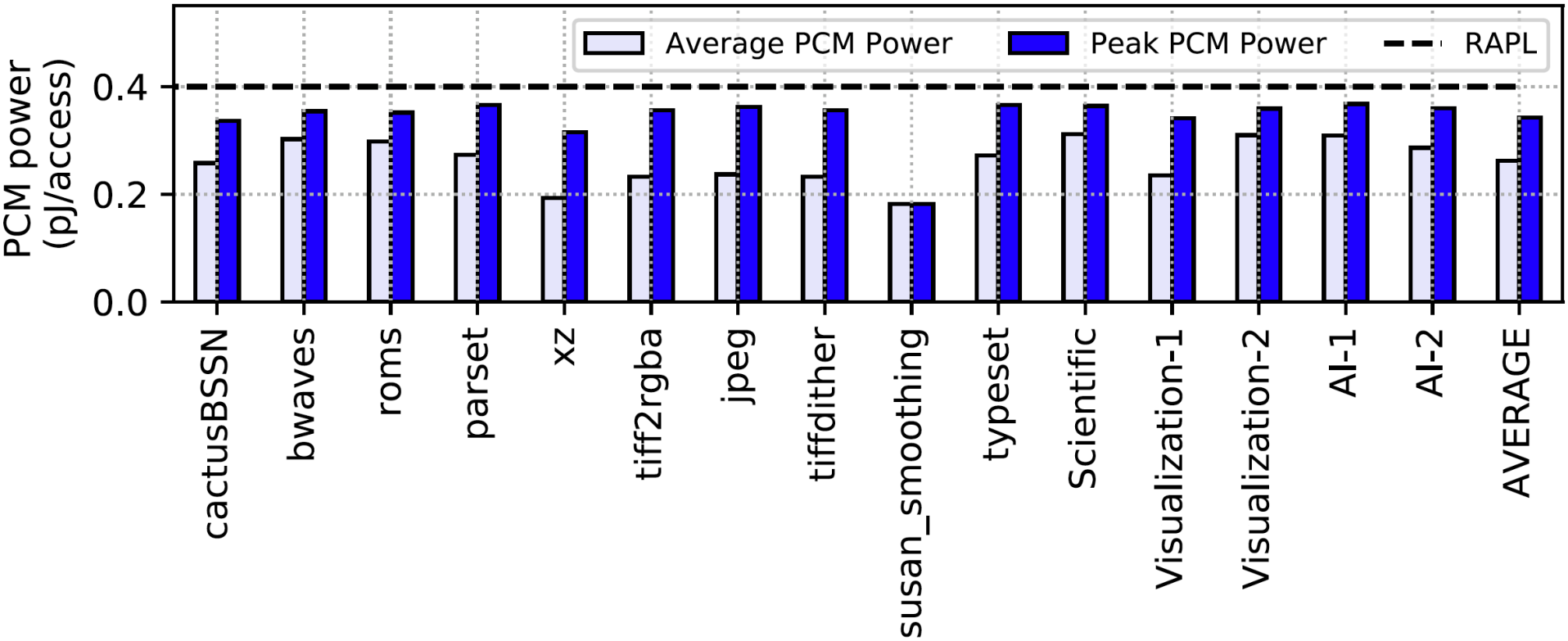
- **Execution time (lower is better)**



51% lower execution time vs. MultiPartition and
28% lower execution time vs. Baseline

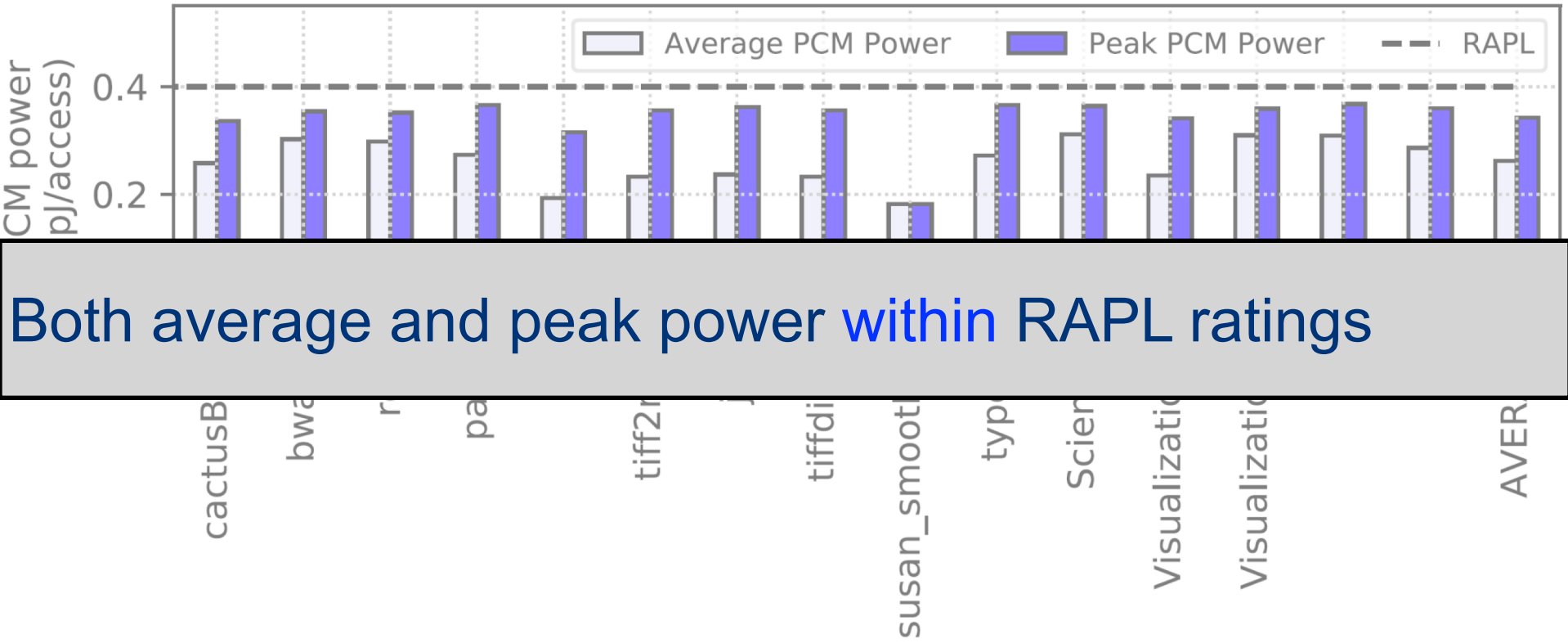
Power Consumption

- **Average Power (lower is better)**
- **Peak Power (lower is better)**



Power Consumption

- **Average Power (lower is better)**
- **Peak Power (lower is better)**



Both average and peak power within RAPL ratings

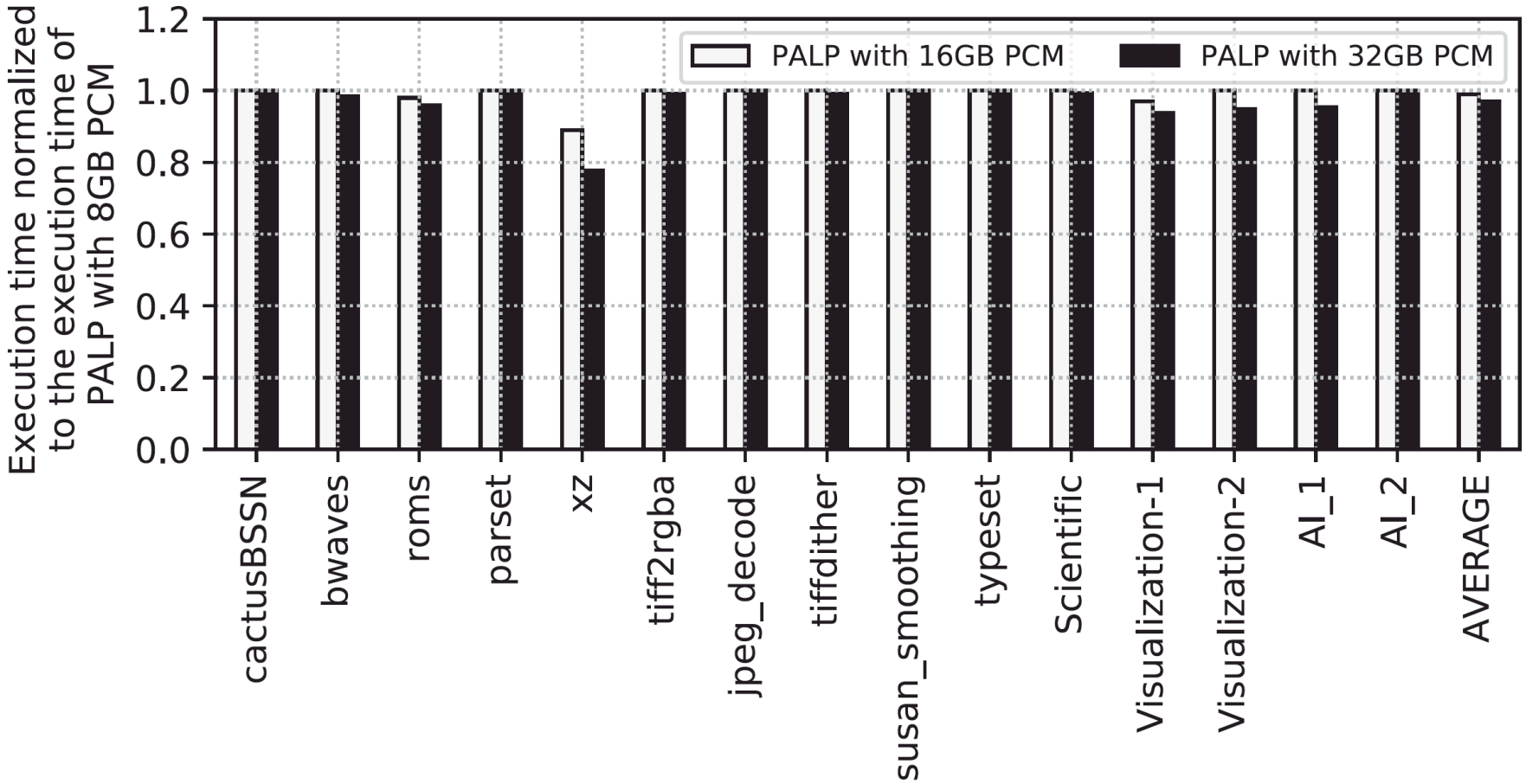
Design Overhead

Design	Technology Type	Vdd (V)	Critical Path Delay (ps)	Power (pJ/access)
Baseline	Double-gate	0.9	1159.2	0.311
Proposed PALP	Double-gate	0.9	1453.2	0.364

On average, critical path delay increases by 25.3%
On average, power consumption increases by 17%

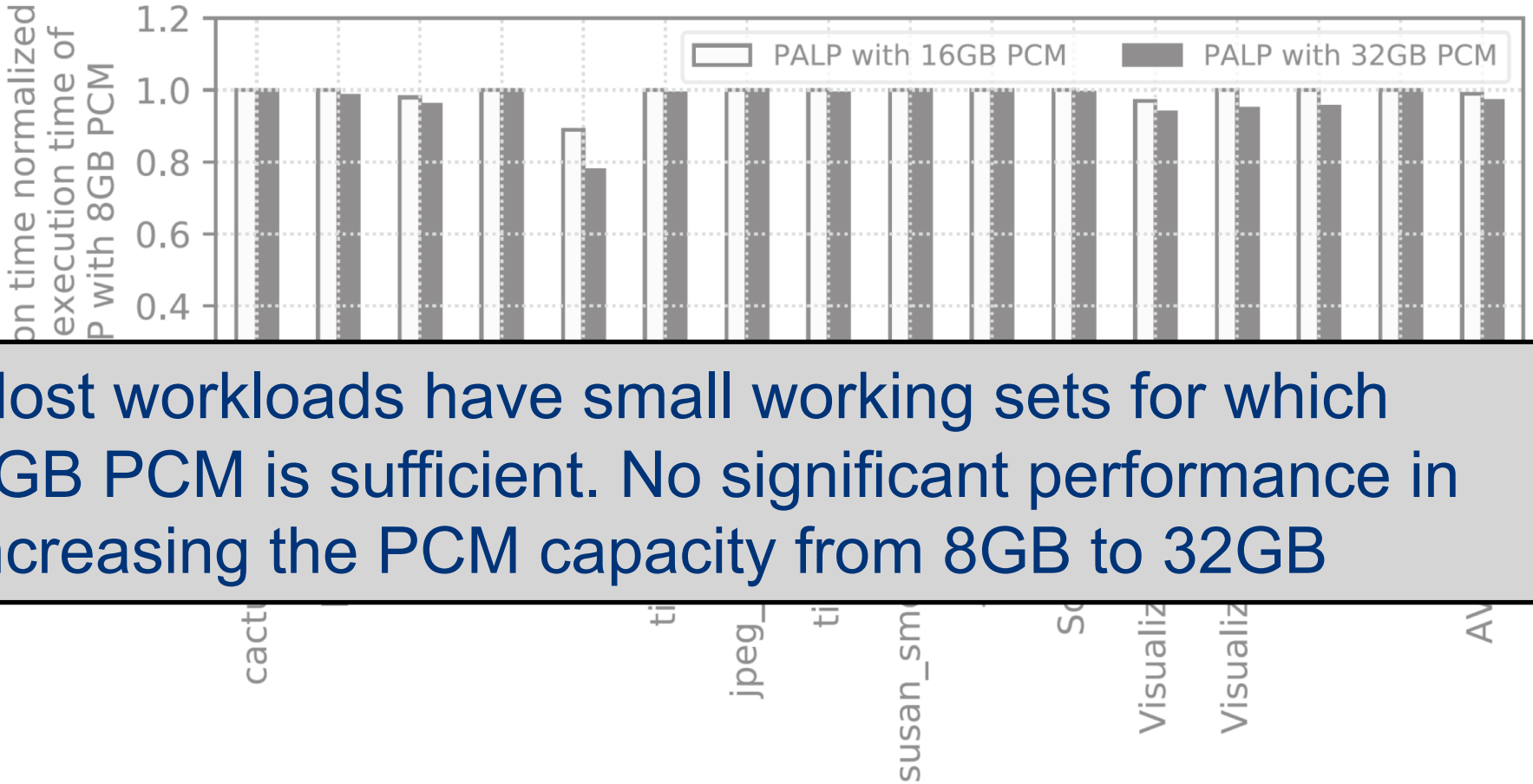
Performance Impact of PCM Capacity

- **8GB, 16GB, and 32GB PCM capacity**



Performance Impact of PCM Capacity

- **8GB, 16GB, and 32GB PCM capacity**



Most workloads have small working sets for which 8GB PCM is sufficient. No significant performance in increasing the PCM capacity from 8GB to 32GB

Outline

- Introduction
- Motivation
- Background on PCM
- Contribution 1: Enabling read-write parallelism
- Contribution 2: Enabling read-read parallelism
- Contribution 3: Exploiting parallelism
- Evaluation
- **Conclusion**

Conclusion

- **Observations**
  - Memory bank conflicts reduce system performance by lowering bank utilization, causing CPU cores to stall
  - A PCM bank's peripheral structures allow to read and program 128 PCM cells in parallel, providing opportunity to resolve bank conflicts
- **Idea: PArtition-Level Parallelism (PALP) in PCM**
  - Introduce new mechanism to enable read-write parallelism in PCM banks with minimum changes to PCM interface and its timing
  - Introduce simple circuit modifications to enable read-read parallelism in PCM banks
  - Propose new access scheduling mechanism to exploit read-write and read-read parallelism in PCM banks
- **Design Updates: Analyze internal architecture of PCM banks**
  - New PCM commands
  - New decoupled write driver design
- **Performance Evaluation**
  - Significant bank conflict reduction (28% average system performance improvement for SPEC CPU 2017 and MiBench workloads)

Open-Source Tool

Open source tool

https://github.com/drexel-DISCO/PALP

Email: shihao.song@drexel.edu

Enabling and Exploiting Partition-Level Parallelism (PALP) in Phase Change Memories

Presenter: Dr. Anup K. Das

Shihao Song, Anup Das, Drexel University
Onur Mutlu, ETH Zurich
Nagarajan Kandasamy, Drexel University

CASES 2019, New York