

Seminar in Computer Architecture

Meeting 3: RowClone (Processing using DRAM)

Prof. Onur Mutlu

ETH Zürich

Fall 2021

7 October 2021

Suggested Paper Discussion Format

- Problem & Goal
- Key Ideas/solution
- Novelty
- Mechanisms & Implementation
- Major Results
- Takeaways/Conclusions

**~20-25 minute
Summary**

- Strengths
- Weaknesses
- Alternatives
- New ideas/problems
- Brainstorming and Discussion

**~10 min Critique
plus
~10 min Discussion**

Presentation Schedule

- We will have ~8 sessions of presentations
- 2 presentations in each of the 11 sessions
 - Max 50 minutes total for each presentation+discussion
 - We will take the entire 2 hours in each meeting
- Each presentation
 - One student presents one paper and leads discussion
 - Max 25 minute summary+analysis
 - Max 10 minute critique
 - Max 10 minute discussion+brainstorming+feedback
 - Should follow the suggested guidelines

Algorithm for Presentation Preparation

- Study Lecture 1b again for presentation guidelines
- Read and analyze your paper thoroughly
 - Discuss with anyone you wish + use any resources
- Prepare a draft presentation based on guidelines
- Meet mentor(s) and get feedback
 - Revise the presentation and delivery
- Meet mentor(s) again and get further feedback
 - Revise the presentation and delivery
- Meetings are mandatory – you have to schedule them with your assigned mentor(s). We may suggest meeting times.
- Practice, practice, practice

Example Paper Presentations

Learning by Example

- A great way of learning
- We will do at least one more today

Structure of the Presentation

- Background, Problem & Goal
- Novelty
- Key Approach and Ideas
- Mechanisms (in some detail)
- Key Results: Methodology and Evaluation
- Summary
- Strengths
- Weaknesses
- Thoughts and Ideas
- Takeaways
- Open Discussion

Background, Problem & Goal

Novelty

Key Approach and Ideas

Mechanisms (in some detail)

Key Results:

Methodology and Evaluation

Summary

Strengths

Weaknesses

Thoughts and Ideas

Takeaways

Open Discussion

Example Paper Presentation

Let's Review This Paper

- Vivek Seshadri, Yoongu Kim, Chris Fallin, Donghyuk Lee, Rachata Ausavarungnirun, Gennady Pekhimenko, Yixin Luo, Onur Mutlu, Michael A. Kozuch, Phillip B. Gibbons, and Todd C. Mowry,
"RowClone: Fast and Energy-Efficient In-DRAM Bulk Data Copy and Initialization"

Proceedings of the 46th International Symposium on Microarchitecture (MICRO), Davis, CA, December 2013. [[Slides \(pptx\)](#)] [[pdf](#)] [[Lightning Session Slides \(pptx\)](#)] [[pdf](#)] [[Poster \(pptx\)](#)] [[pdf](#)]

RowClone: Fast and Energy-Efficient In-DRAM Bulk Data Copy and Initialization

Vivek Seshadri vseshadr@cs.cmu.edu	Yoongu Kim yoongukim@cmu.edu	Chris Fallin* cfallin@c1f.net	Donghyuk Lee donghyuk1@cmu.edu
---------------------------------------	---------------------------------	----------------------------------	-----------------------------------

Rachata Ausavarungnirun rachata@cmu.edu	Gennady Pekhimenko gpekhime@cs.cmu.edu	Yixin Luo yixinluo@andrew.cmu.edu
--	---	--------------------------------------

Onur Mutlu onur@cmu.edu	Phillip B. Gibbons† phillip.b.gibbons@intel.com	Michael A. Kozuch† michael.a.kozuch@intel.com	Todd C. Mowry tcm@cs.cmu.edu
----------------------------	--	--	---------------------------------

Carnegie Mellon University †Intel Pittsburgh

RowClone

**Fast and Energy-Efficient In-DRAM
Bulk Data Copy and Initialization**

Vivek Seshadri

Y. Kim, C. Fallin, D. Lee, R. Ausavarungnirun,
G. Pekhimenko, Y. Luo, O. Mutlu,
P. B. Gibbons, M. A. Kozuch, T. C. Mowry

SAFARI

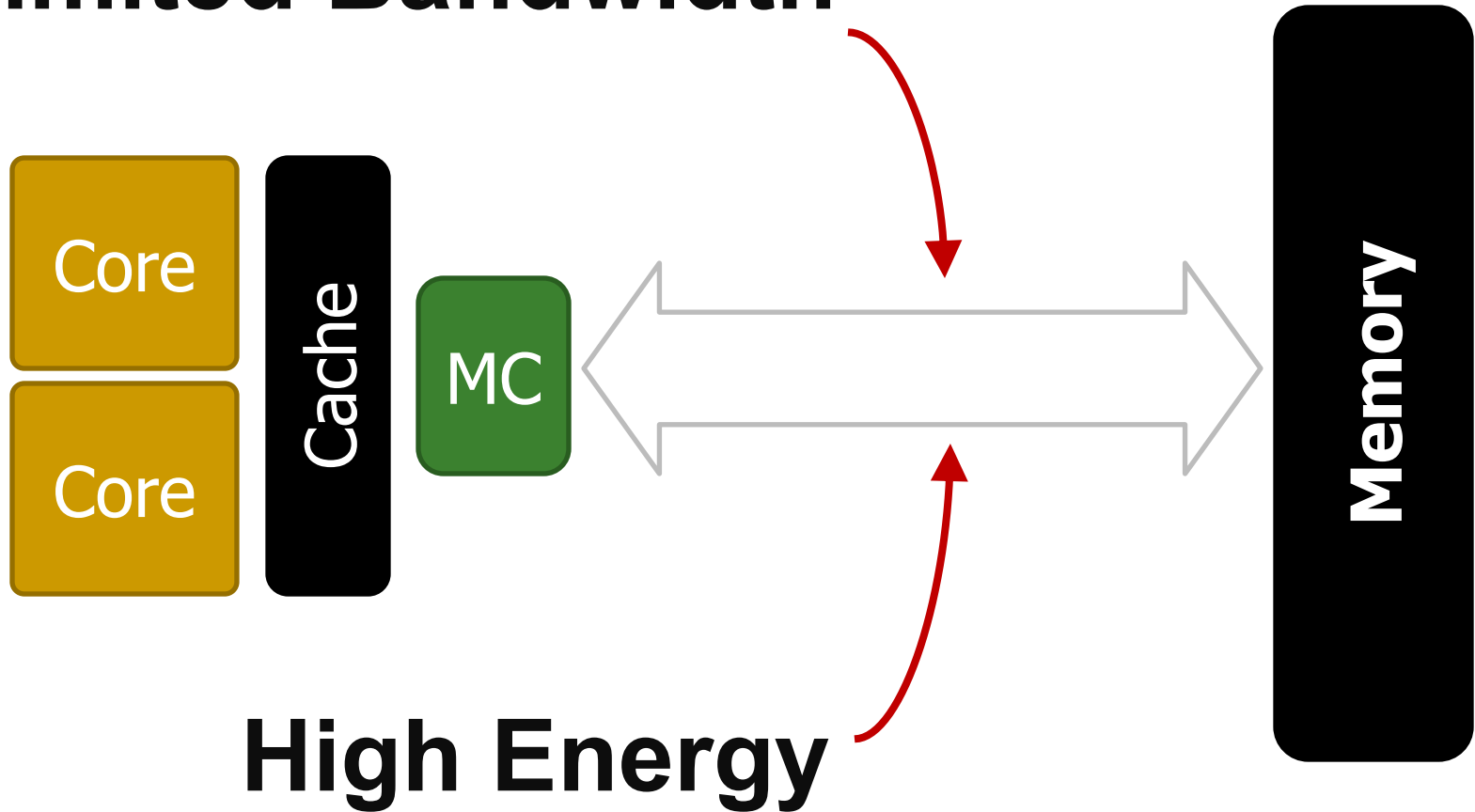
Carnegie Mellon



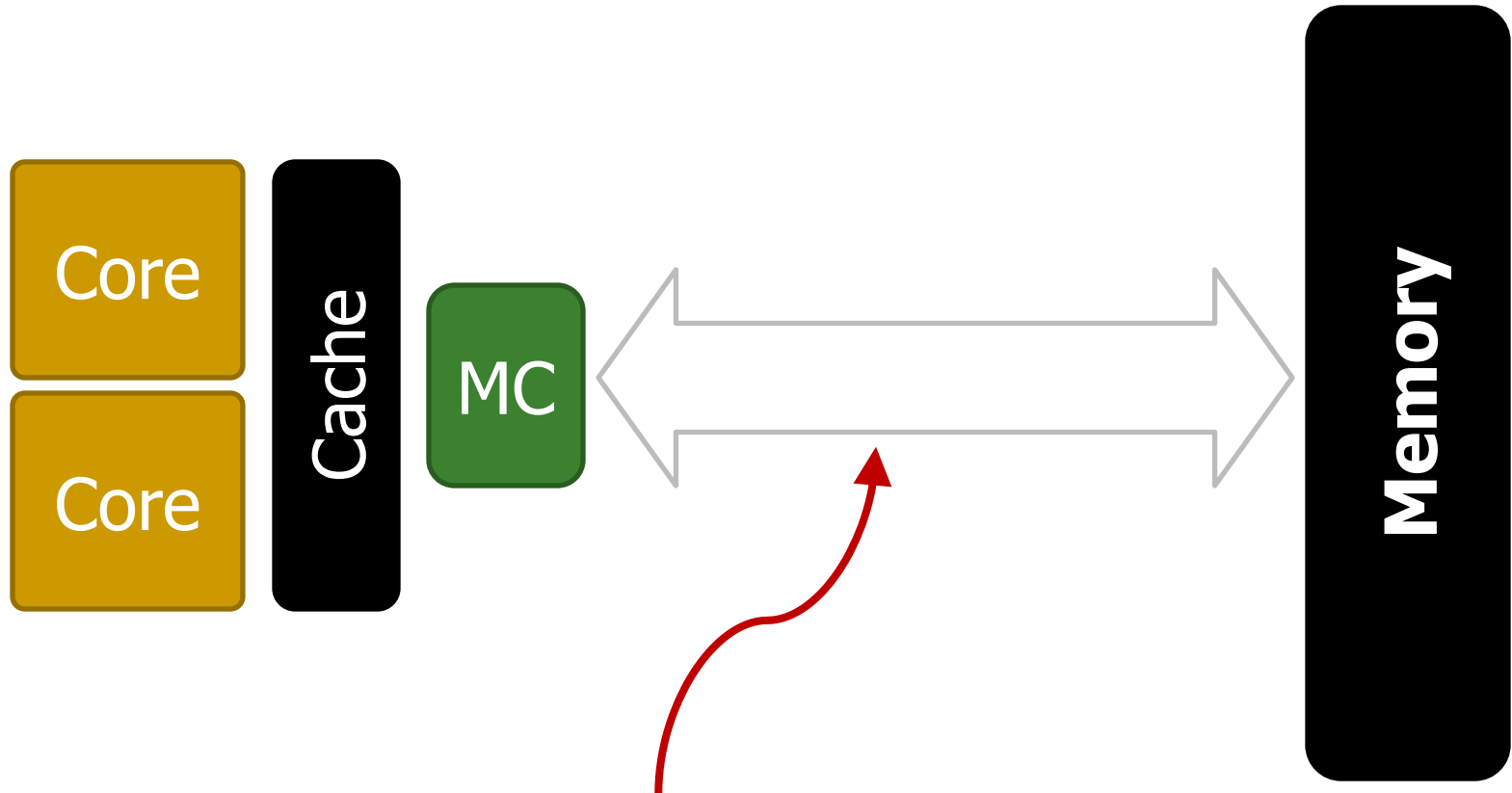
Background, Problem & Goal

Memory Channel – Bottleneck

Limited Bandwidth



Goal: Reduce Memory Bandwidth Demand

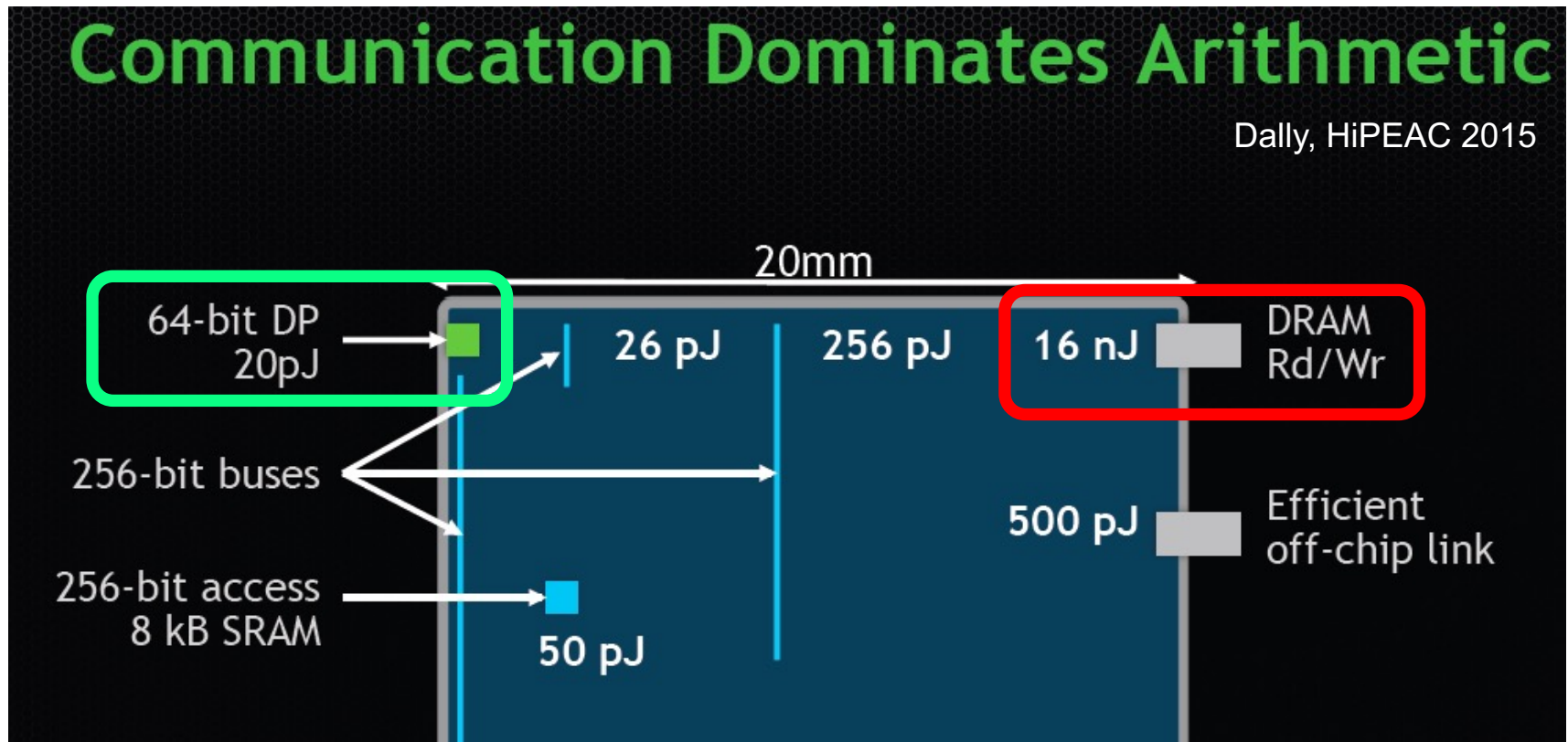


Reduce unnecessary data movement

Data Movement vs. Computation Energy

Communication Dominates Arithmetic

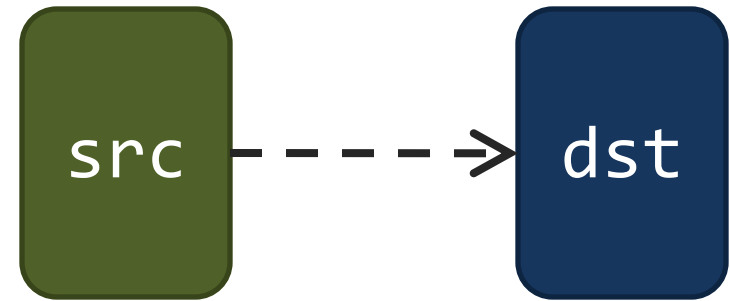
Dally, HiPEAC 2015



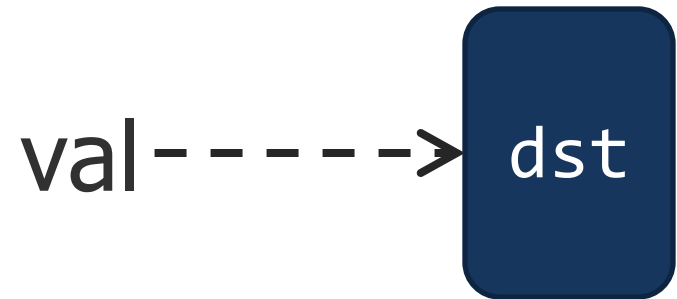
A memory access consumes $\sim 100\text{-}1000\times$ the energy of a complex addition

Bulk Data Copy and Initialization

**Bulk Data
Copy**



**Bulk Data
Initialization**



Bulk Data Copy and Initialization

The Impact of Architectural Trends on Operating System Performance

Mendel Rosenblum, Edouard Bugnion, Stephen Alan Herrod,
Emmett Witchel, and Anoop Gupta

Hardware Support for Bulk Data Movement in Server Platforms

Li Zhao[†], Ravi Iyer[‡], Srihari Makineni[‡], Laxmi Bhuyan[†] and Don Newell[‡]

[†]Department of Computer Science and Engineering, University of California, Riverside, CA 92521
Email: {zhao, bhuyan}@cs.ucr.edu

[‡]Communications Technology Lab, Intel Corp.

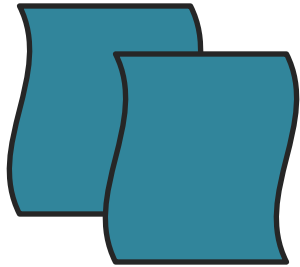
Architecture Support for Improving Bulk Memory Copying and Initialization Performance

Xiaowei Jiang, Yan Solihin
Dept. of Electrical and Computer Engineering
North Carolina State University
Raleigh, USA

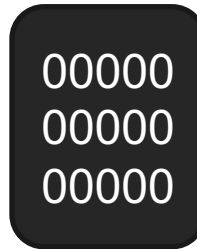
Li Zhao, Ravishankar Iyer
Intel Labs
Intel Corporation
Hillsboro, USA

Bulk Data Copy and Initialization

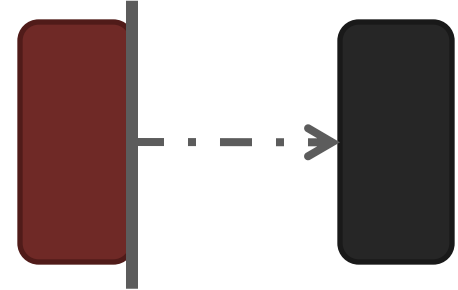
memmove & memcpy: 5% cycles in Google's datacenter [Kanev+ ISCA'15]



Forking



Zero initialization
(e.g., security)



Checkpointing



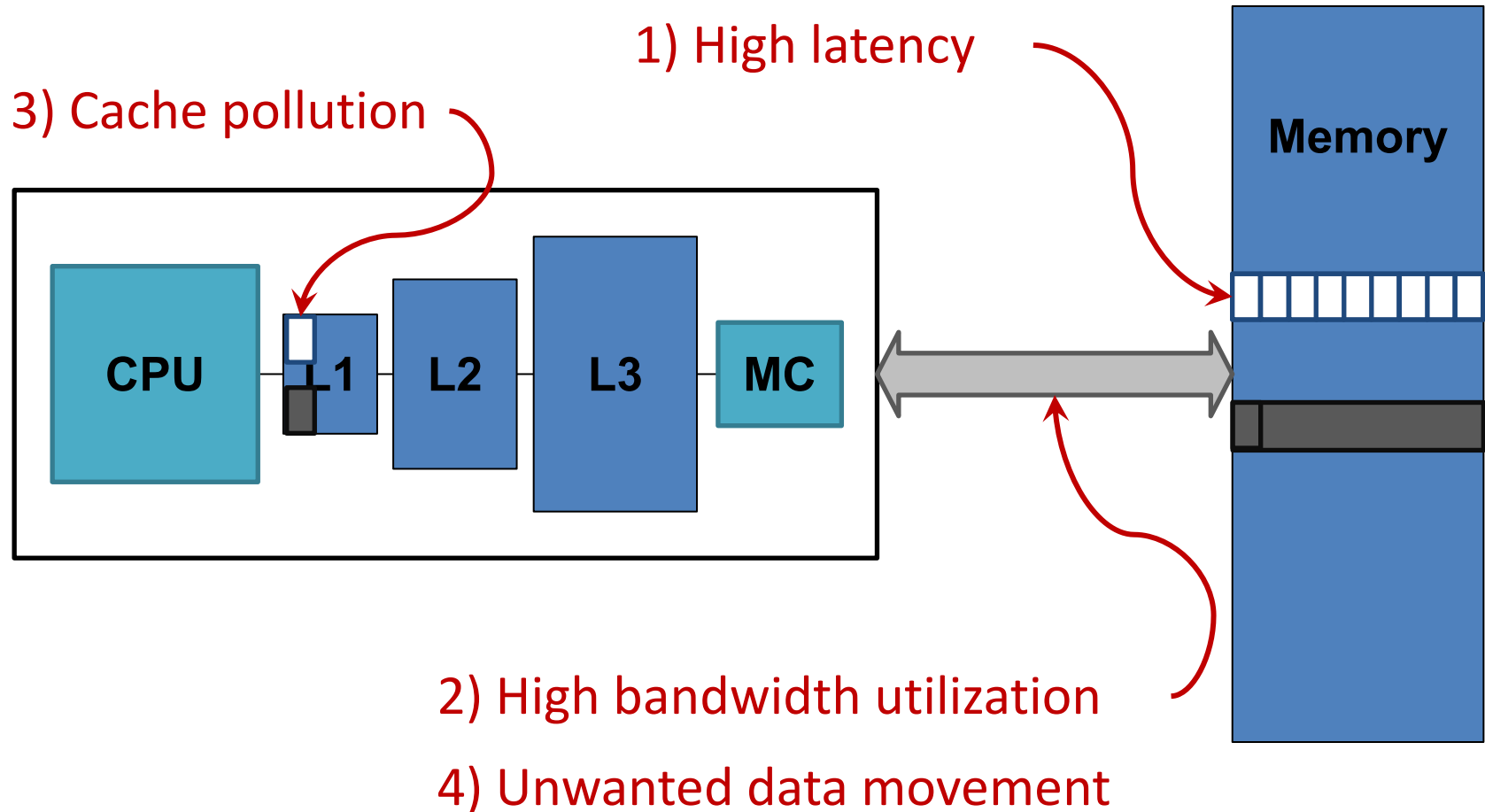
VM Cloning
Deduplication



Page Migration

...
Many more

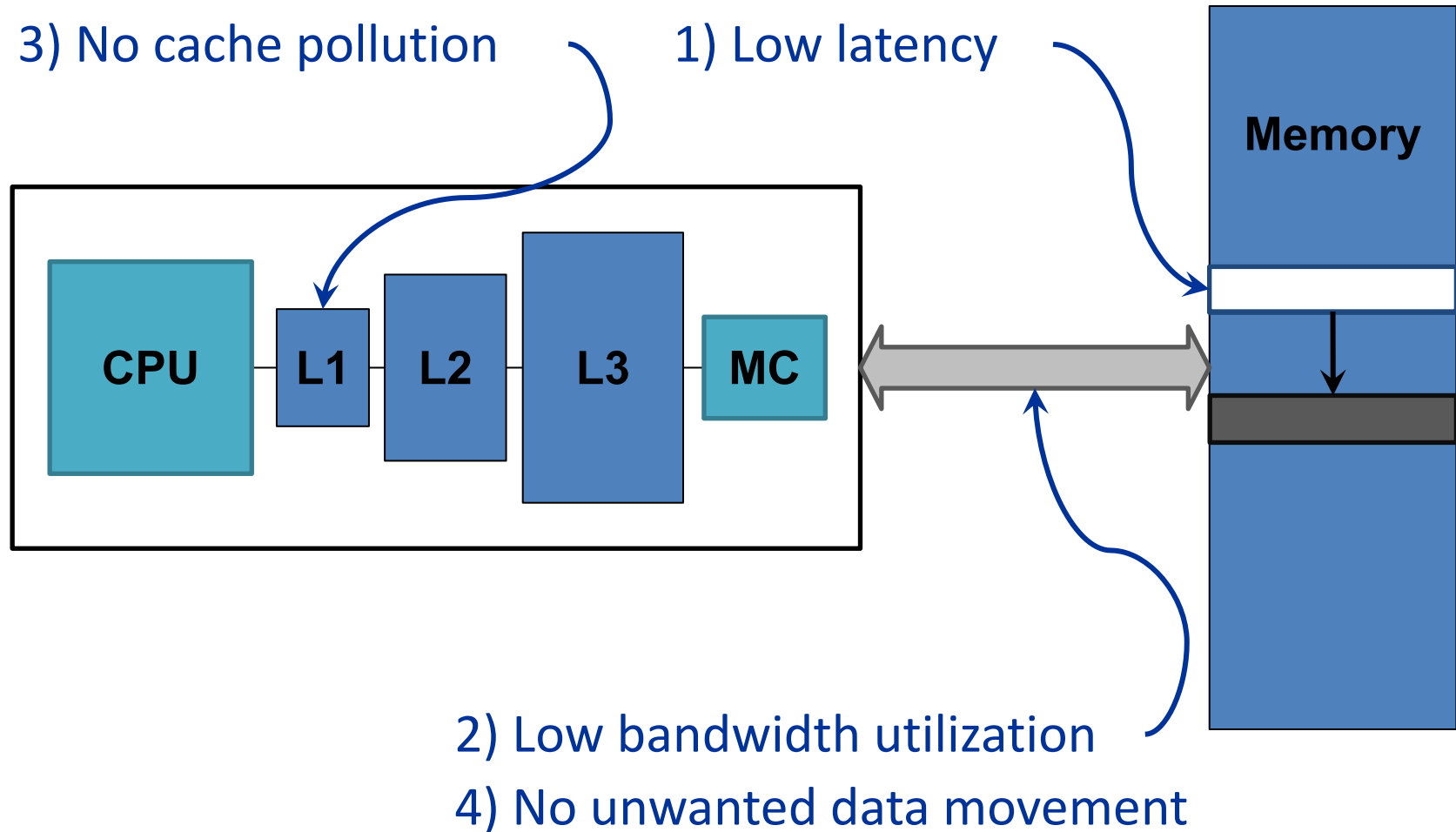
Shortcomings of Today's Systems



1046ns, 3.6uJ (for 4KB page copy via DMA)

Novelty, Key Approach, and Ideas

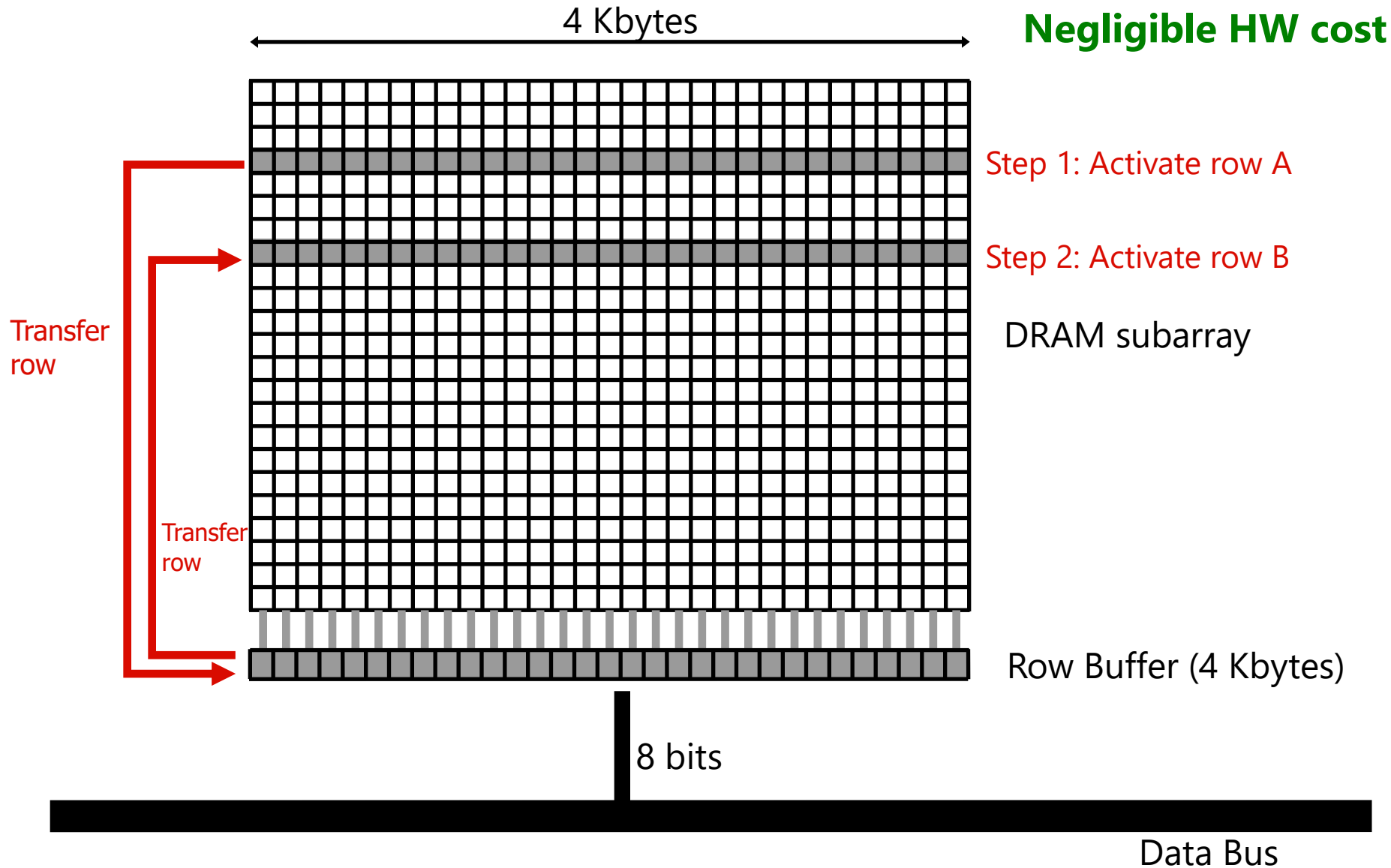
RowClone: In-Memory Copy



1046ns, 3.6uJ → 90ns, 0.04uJ

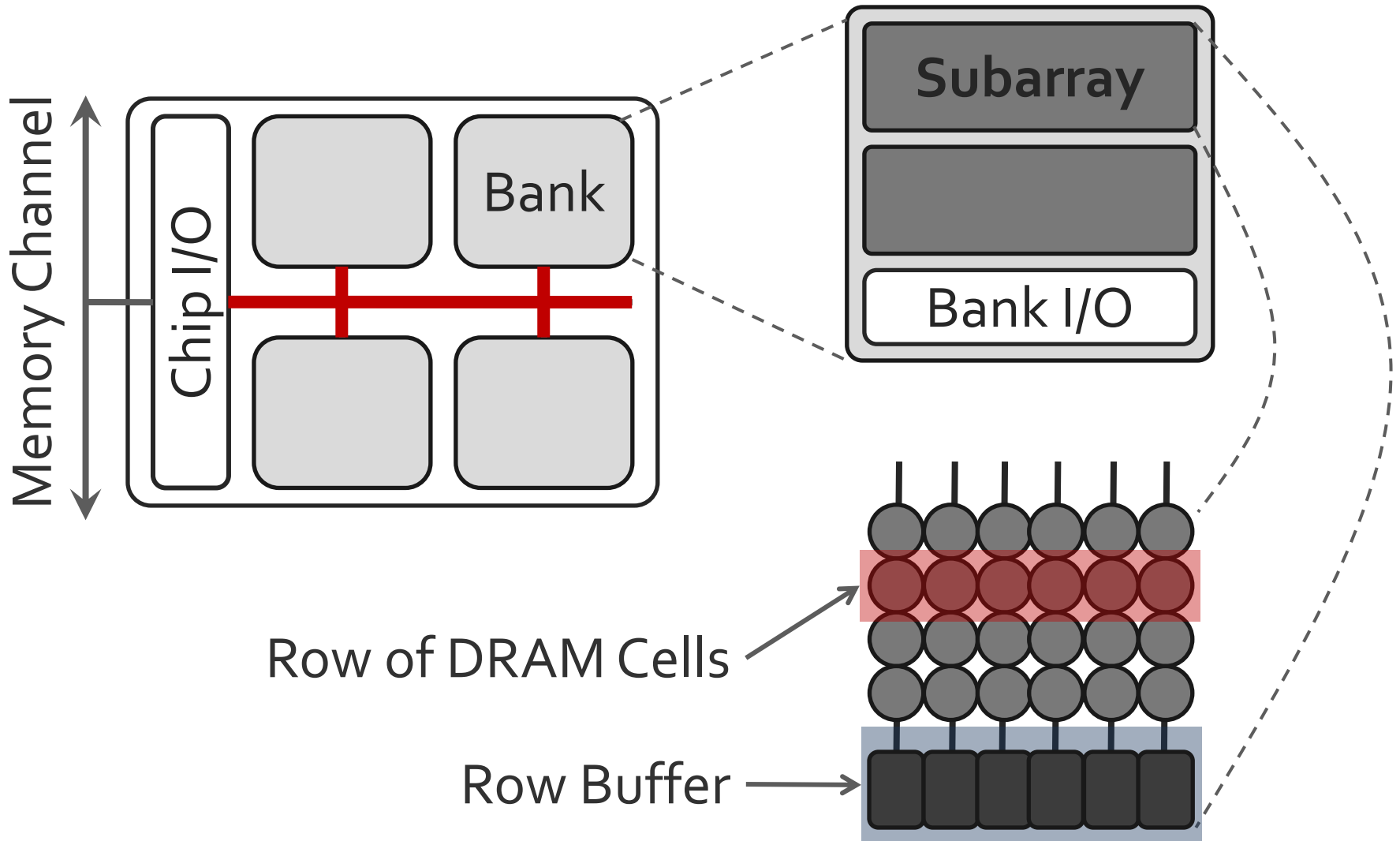
RowClone: In-DRAM Row Copy

**Idea: Two consecutive ACTivates
Negligible HW cost**



Mechanisms (in some detail)

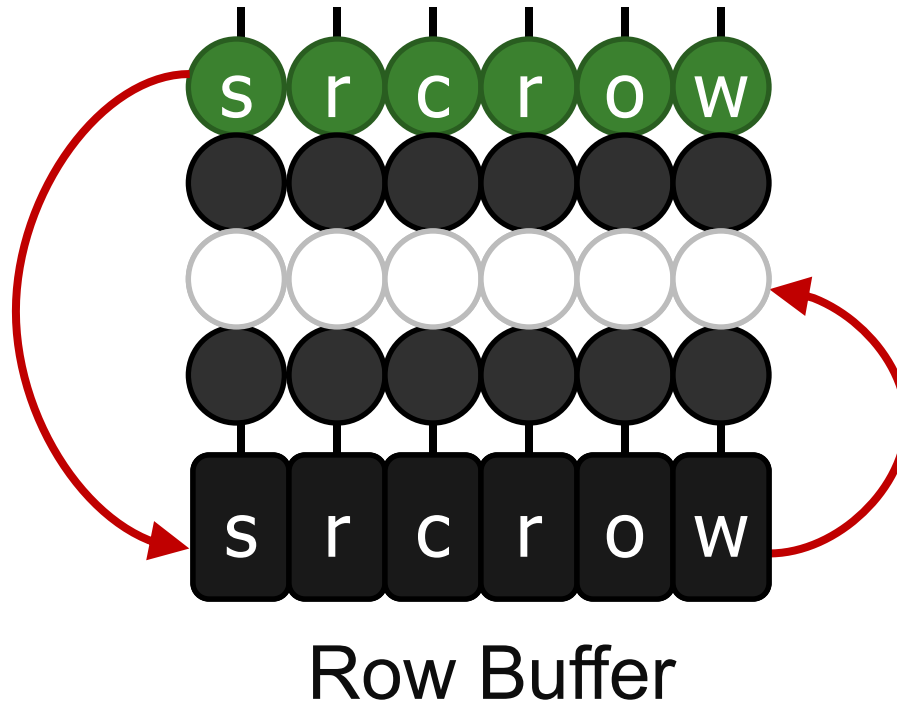
DRAM Chip Organization



RowClone Types

- Intra-subarray RowClone (row granularity)
 - Fast Parallel Mode (FPM)
- Inter-bank RowClone (byte granularity)
 - Pipelined Serial Mode (PSM)
- Inter-subarray RowClone

RowClone: Fast Parallel Mode (FPM)

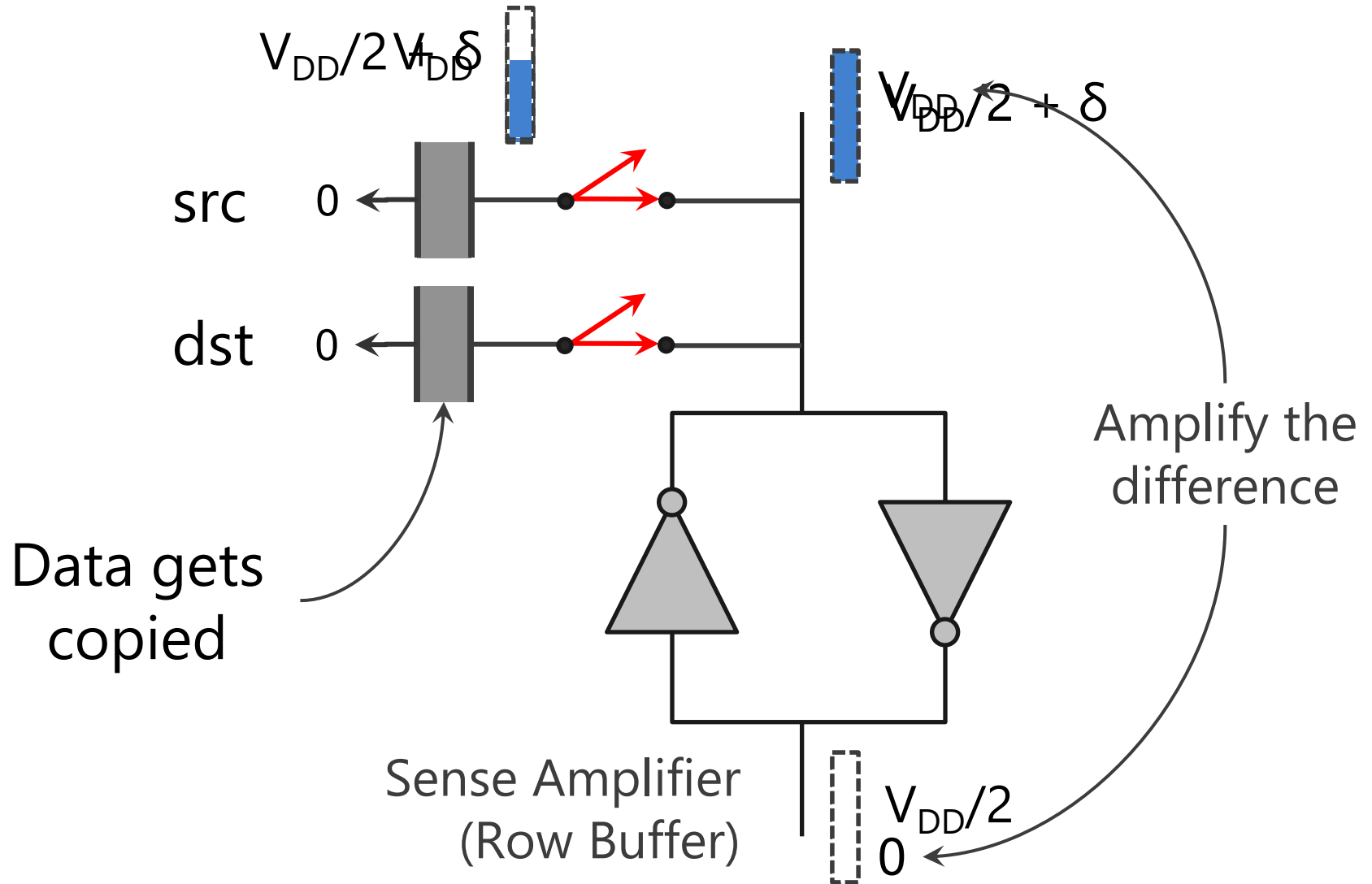


1. Source row to row buffer

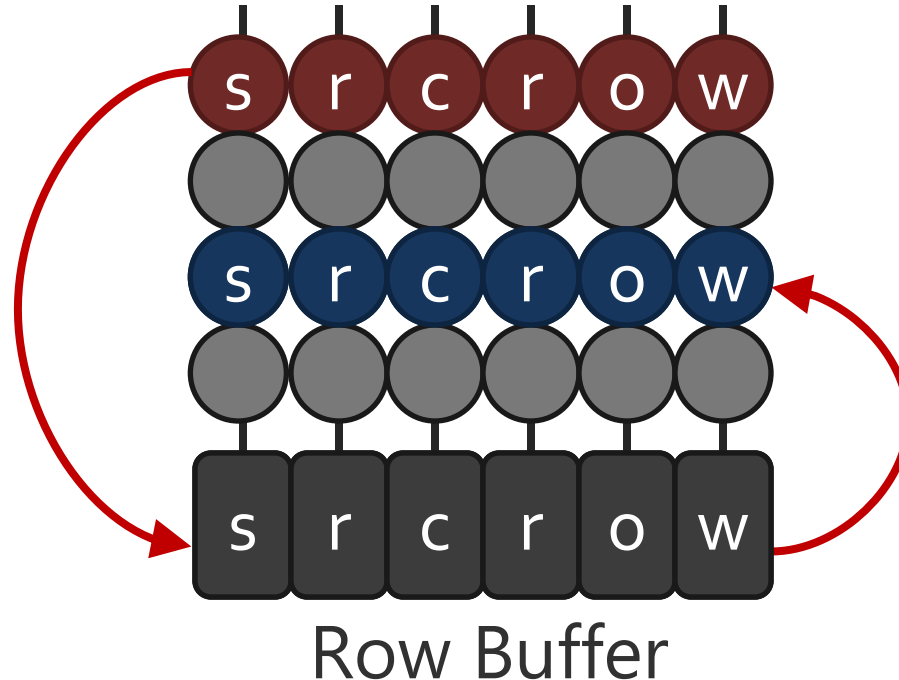


2. Row buffer to destination row

RowClone: Intra-Subarray (I)



RowClone: Intra-Subarray (II)



1. **Activate** src row (copy data from src to row buffer)
2. **Activate** dst row (disconnect src from row buffer, connect dst – copy data from row buffer to dst)

Fast Parallel Mode: Benefits

Bulk Data Copy

Latency **11x** ↓

1046ns to 90ns

Energy **74x** ↓

3600nJ to 40nJ

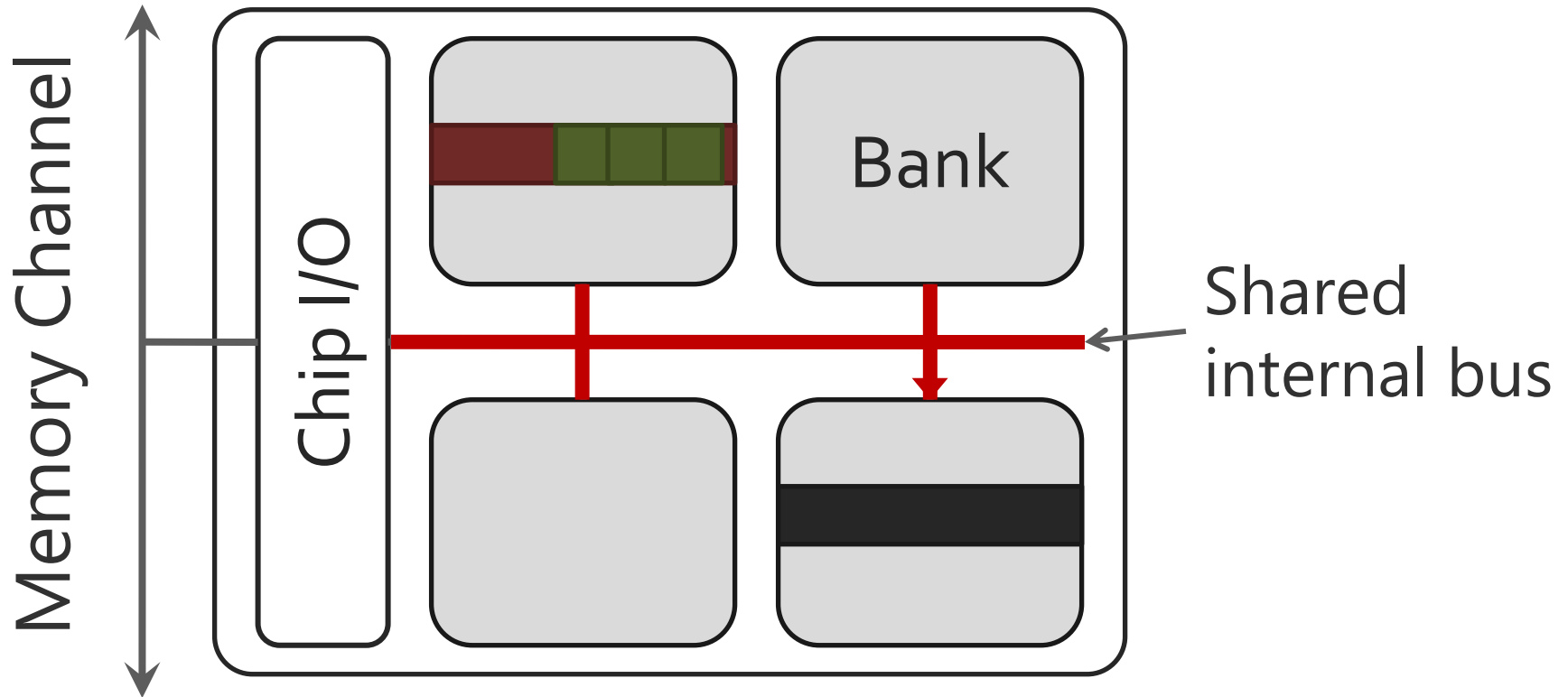
No bandwidth consumption

Very little changes to the DRAM chip

Fast Parallel Mode: Constraints

- Location of source/destination
 - Both should be in the same subarray
- Size of the copy
 - Copies *a//* the data from source row to destination

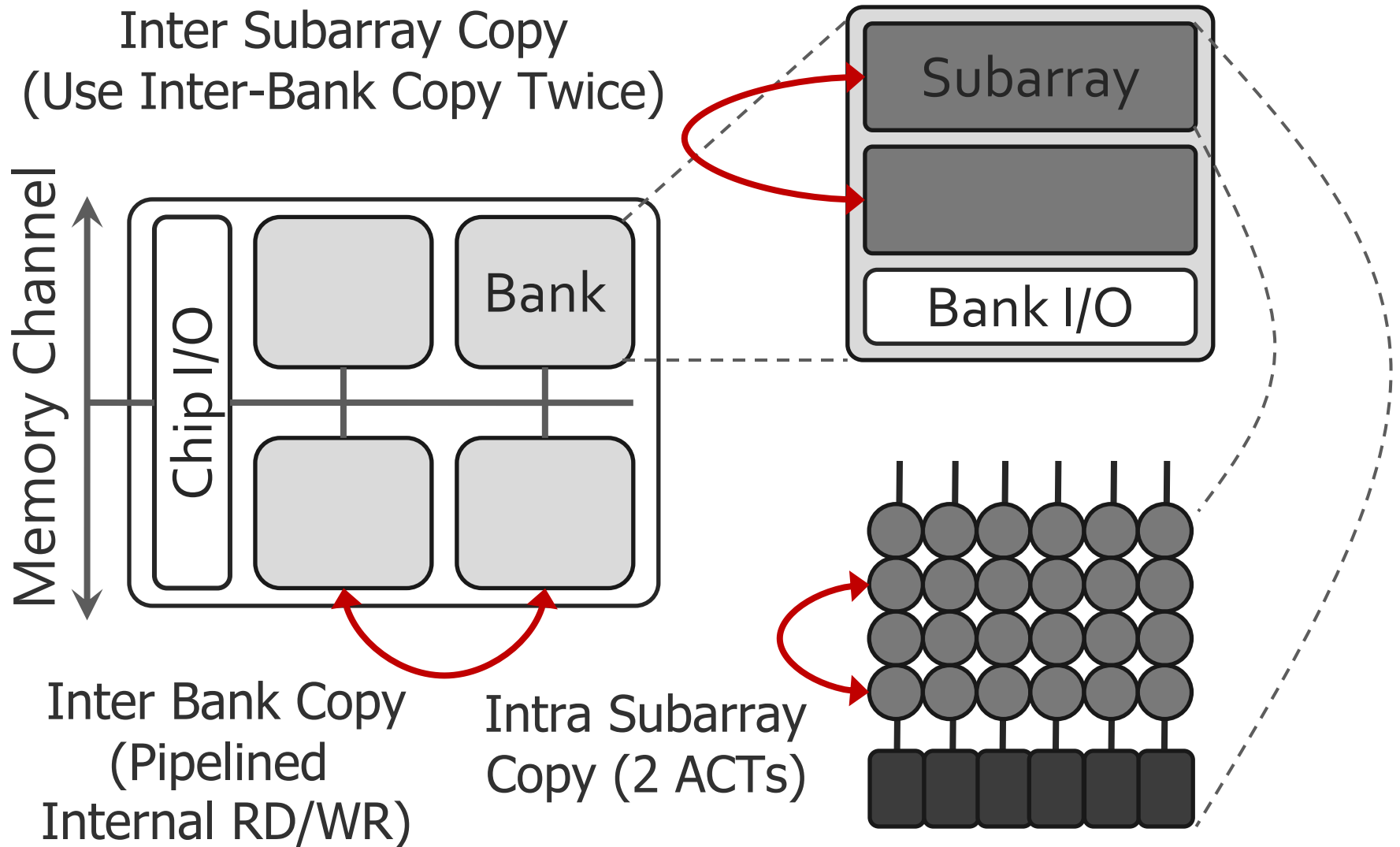
RowClone: Inter-Bank



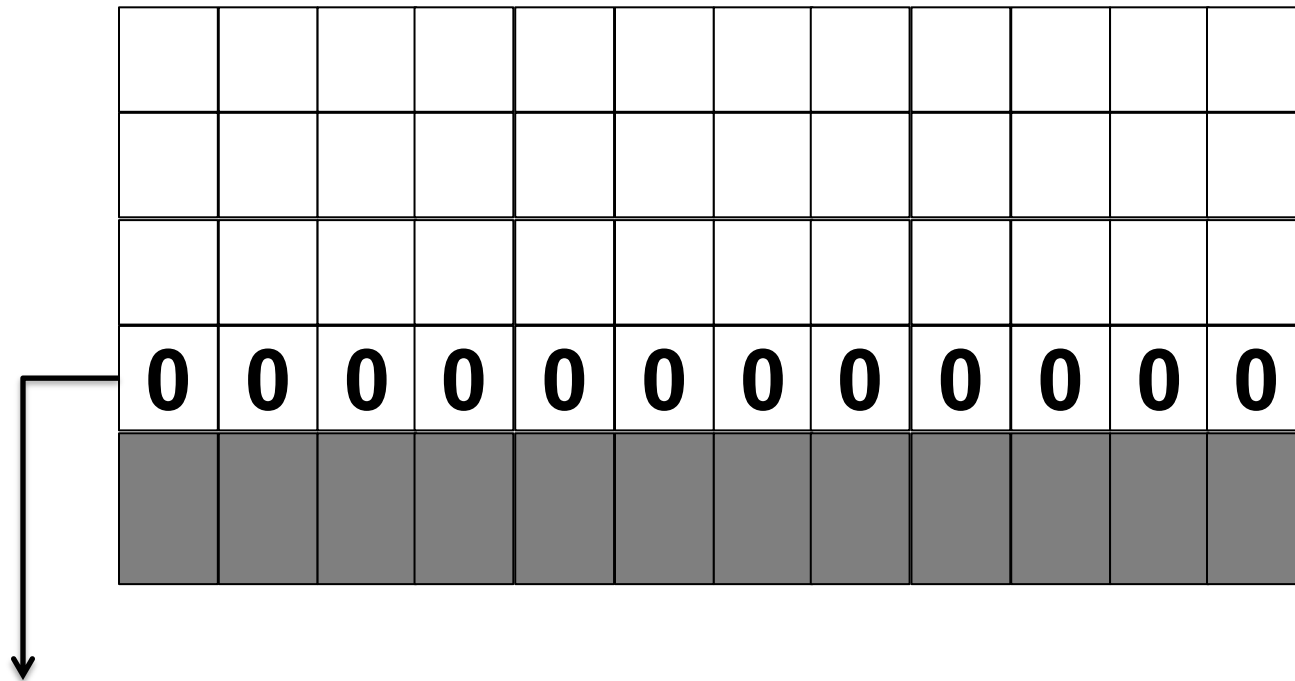
Overlap the latency of the read and the write
1.9X latency reduction, **3.2X** energy reduction

Generalized RowClone

0.01% area cost



RowClone: Fast Row Initialization



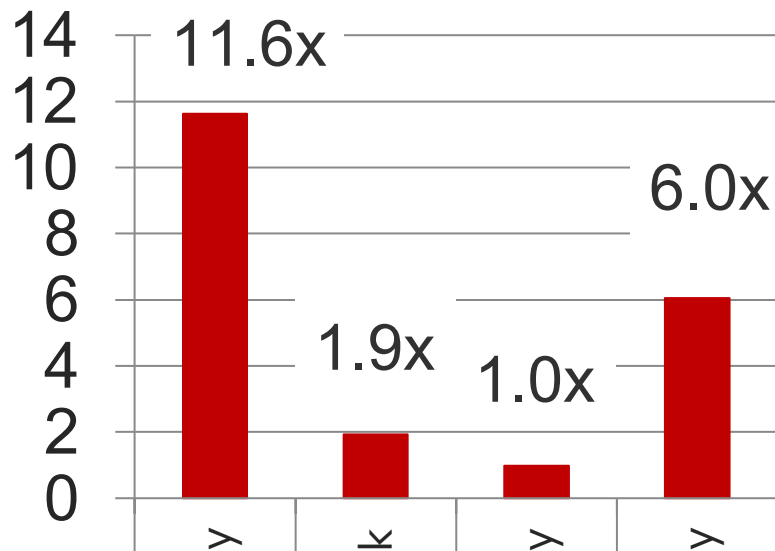
Fix a row at Zero
(0.5% loss in capacity)

RowClone: Bulk Initialization

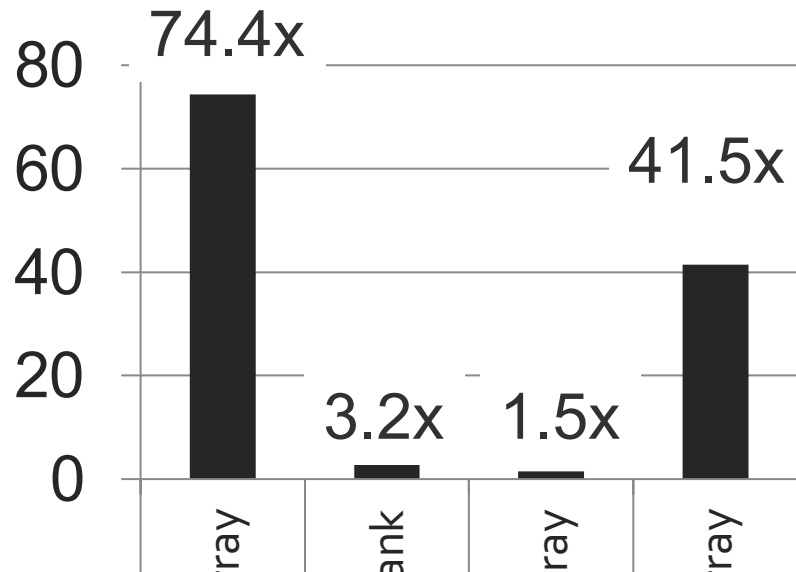
- Initialization with arbitrary data
 - Initialize one row
 - Copy the data to other rows
- Zero initialization (most common)
 - Reserve a row in each subarray (always zero)
 - Copy data from reserved row (FPM mode)
 - **6.0X** lower latency, **41.5X** lower DRAM energy
 - 0.2% loss in capacity

RowClone: Latency & Energy Benefits

Latency Reduction



Energy Reduction



Very low cost: 0.01% increase in die area

System Design to Enable RowClone

End-to-End System Design

Application

How to communicate occurrences of bulk copy/initialization across layers?

Operating System

How to ensure cache coherence?

ISA

Microarchitecture

How to maximize latency and energy savings?

DRAM (RowClone)

How to handle data reuse?

1. Hardware/Software Interface

- Two new instructions
 - memcpy and meminit
 - Similar instructions present in existing ISAs
- Microarchitecture Implementation
 - Checks if instructions can be sped up by RowClone
 - Export instructions to the memory controller

2. Managing Cache Coherence

- RowClone modifies data in memory
 - Need to maintain coherence of cached data
- Similar to DMA
 - Source and destination in memory
 - Can leverage hardware support for DMA
- Additional optimizations

3. Maximizing Use of the Fast Parallel Mode

- Make operating system subarray-aware
- Primitives amenable to use of FPM
 - **Copy-on-Write**
 - Allocate destination in same subarray as source
 - Use FPM to copy
 - **Bulk Zeroing**
 - Use FPM to copy data from reserved zero row

4. Handling Data Reuse After Zeroing

- Data reuse after zero initialization
 - Phase 1: OS zeroes out the page
 - Phase 2: Application uses cachelines of the page
- RowClone
 - Avoids misses in phase 1
 - But incurs misses in phase 2
- **RowClone-Zero-Insert (RowClone-ZI)**
 - Insert clean zero cachelines

Key Results:

Methodology and Evaluation

Methodology

- Out-of-order multi-core simulator
 - 1MB/core last-level cache
 - Cycle-accurate DDR3 DRAM simulator

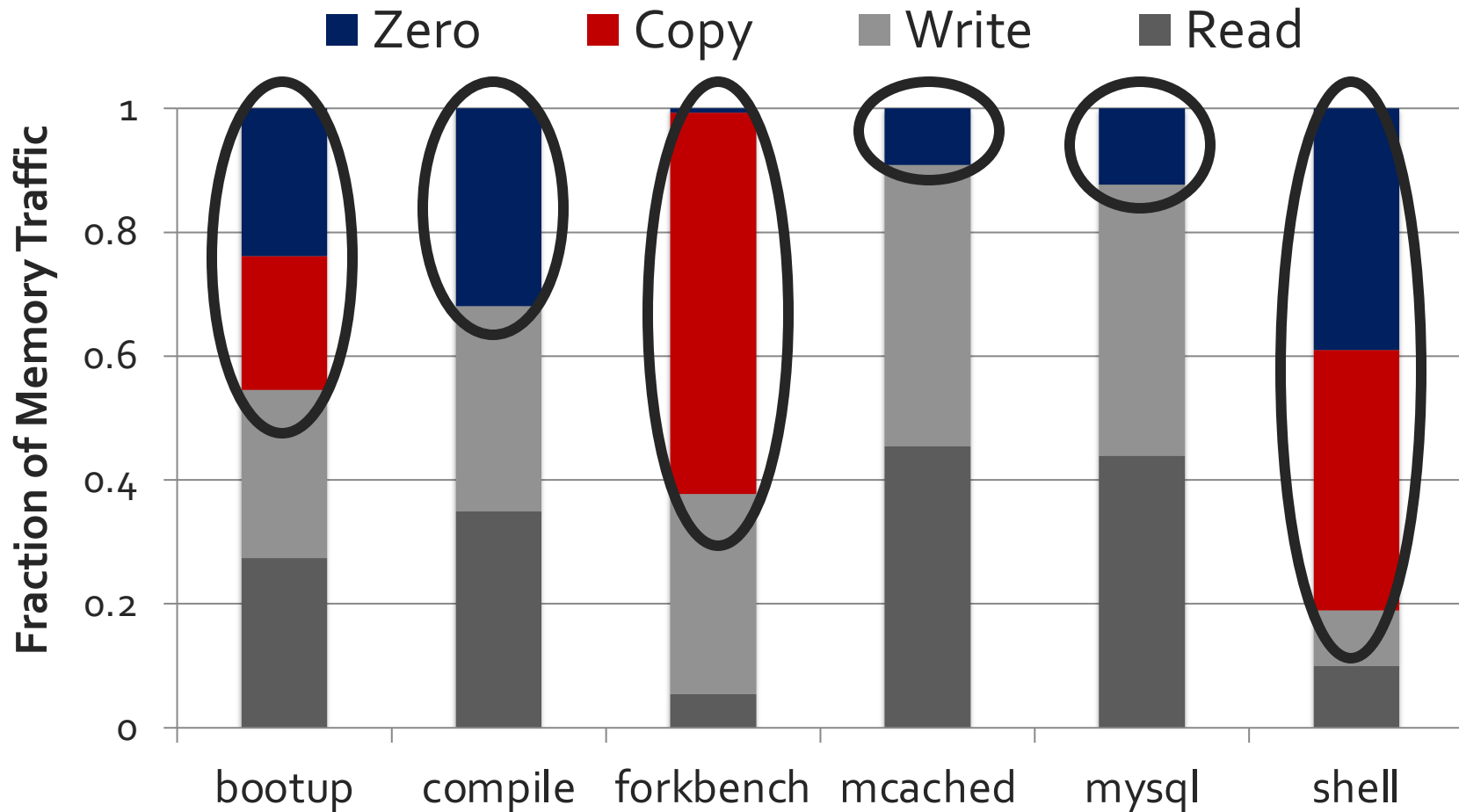
 - 6 Copy/Initialization intensive applications
+SPEC CPU2006 for multi-core

 - Performance
 - Instruction throughput for single-core
 - Weighted Speedup for multi-core
-

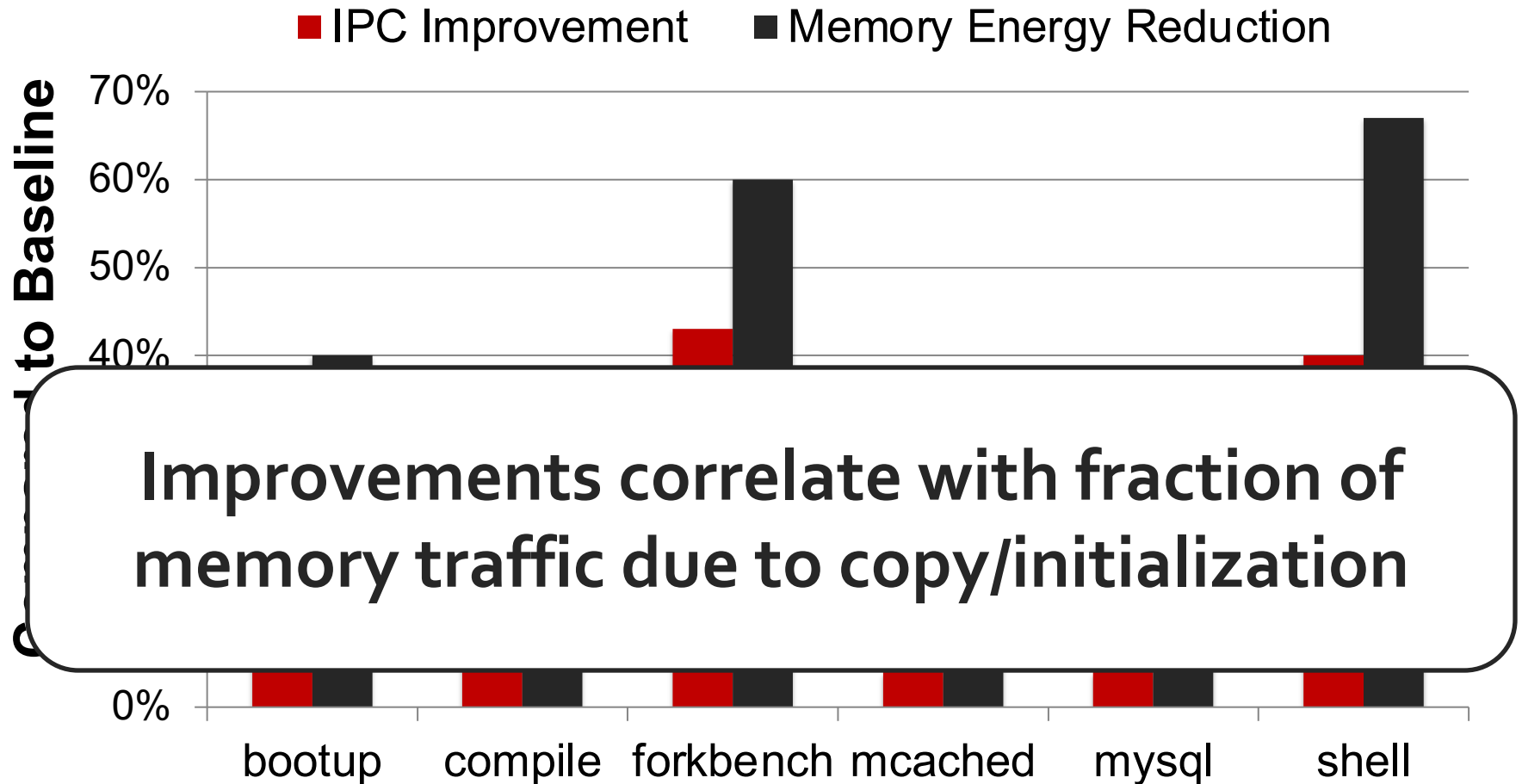
Copy/Initialization Intensive Applications

- **System bootup** (Booting the Debian OS)
 - **Compile** (GNU C compiler – executing cc1)
 - **Forkbench** (A fork microbenchmark)
 - **Memcached** (Inserting a large number of objects)
 - **MySql** (Loading a database)
 - **Shell** script (find with ls on each subdirectory)
-

Copy and Initialization in Workloads



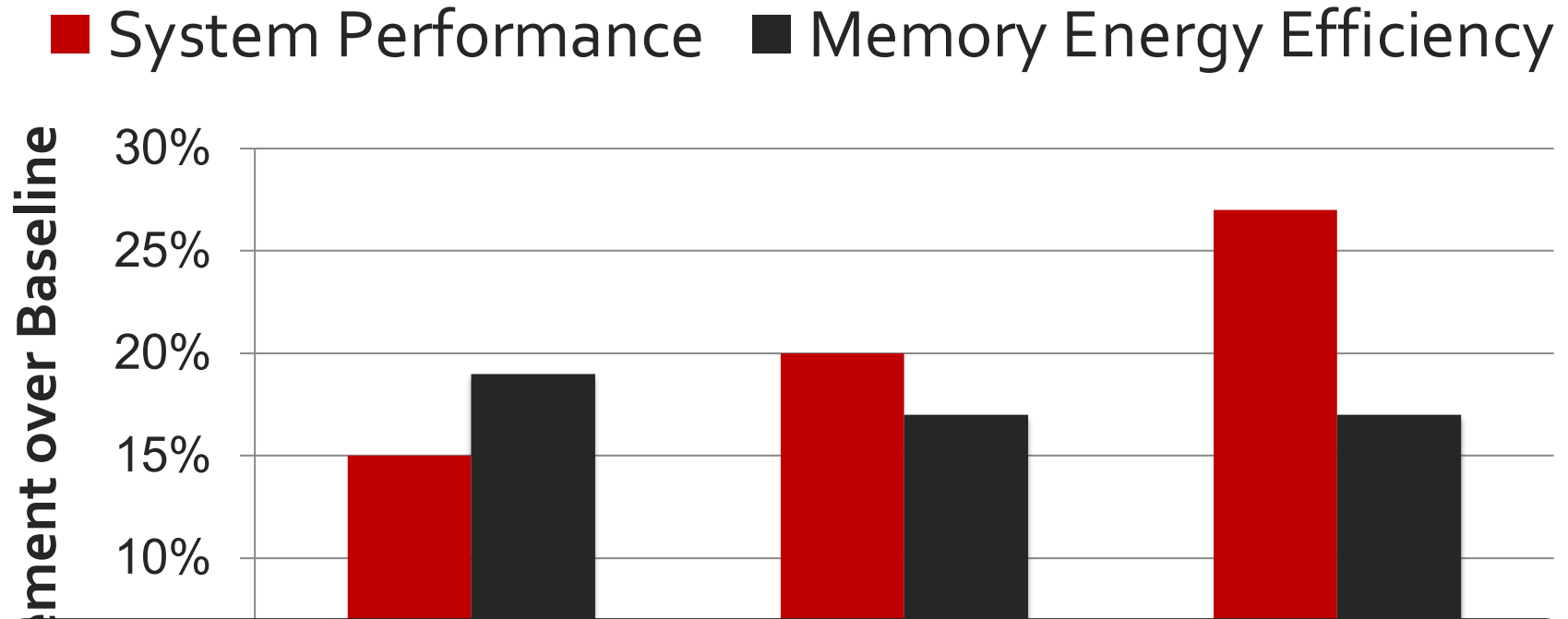
Single-Core – Performance and Energy



Multi-Core Systems

- Reduced bandwidth consumption benefits all applications.
- Run copy/initialization intensive applications with memory intensive SPEC applications.
- Half the cores run copy/initialization intensive applications. Remaining half run SPEC applications.

Multi-Core Results: Summary



**Consistent improvement in
energy/instruction**

Summary

Executive Summary

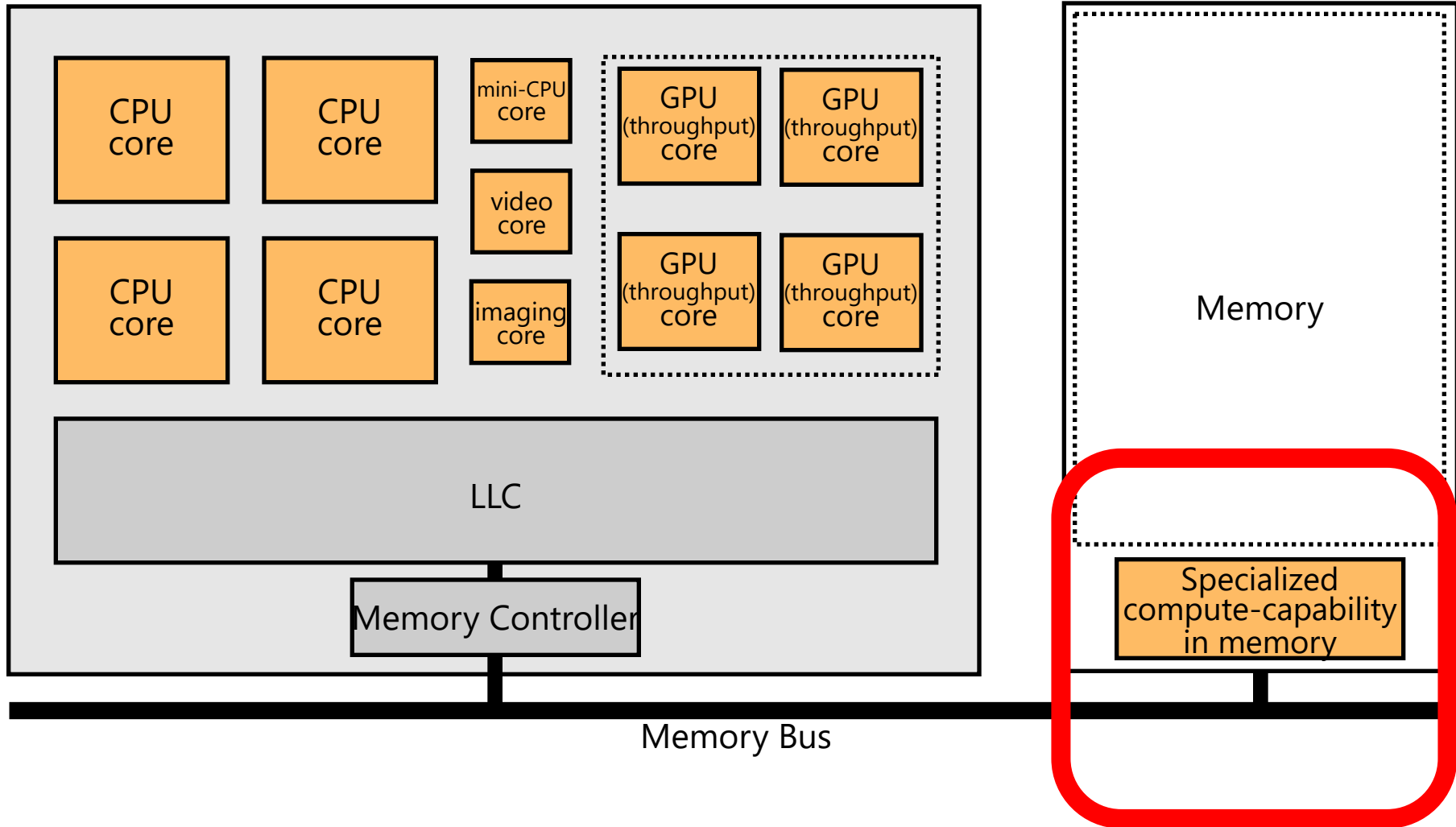
- Bulk data copy and initialization
 - Unnecessarily move data on the memory channel
 - Degrade system performance and energy efficiency
- **RowClone** – perform copy in DRAM with low cost
 - Uses row buffer to copy large quantity of data
 - **Source row → row buffer → destination row**
 - 11X lower latency and 74X lower energy for a bulk copy
- Accelerate Copy-on-Write and Bulk Zeroing
 - Forking, checkpointing, zeroing (security), VM cloning
- Improves performance and energy efficiency at low cost
 - 27% and 17% for 8-core systems (0.01% DRAM chip area)

Strengths

Strengths of the Paper

- Simple, novel mechanism to solve an important problem
- Effective and low hardware overhead
- Intuitive idea!
- Greatly improves performance and efficiency (assuming data is mapped nicely)
- Seems like a clear win for data initialization (without mapping requirements)
- Makes software designer's life easier
 - If copies are 10x-100x cheaper, how to design software?
- Paper tackles many low-level and system-level issues
- Well-written, insightful paper

Mindset: Memory as an Accelerator



Memory similar to a "conventional" accelerator

Mindset: Minimally Changing DRAM

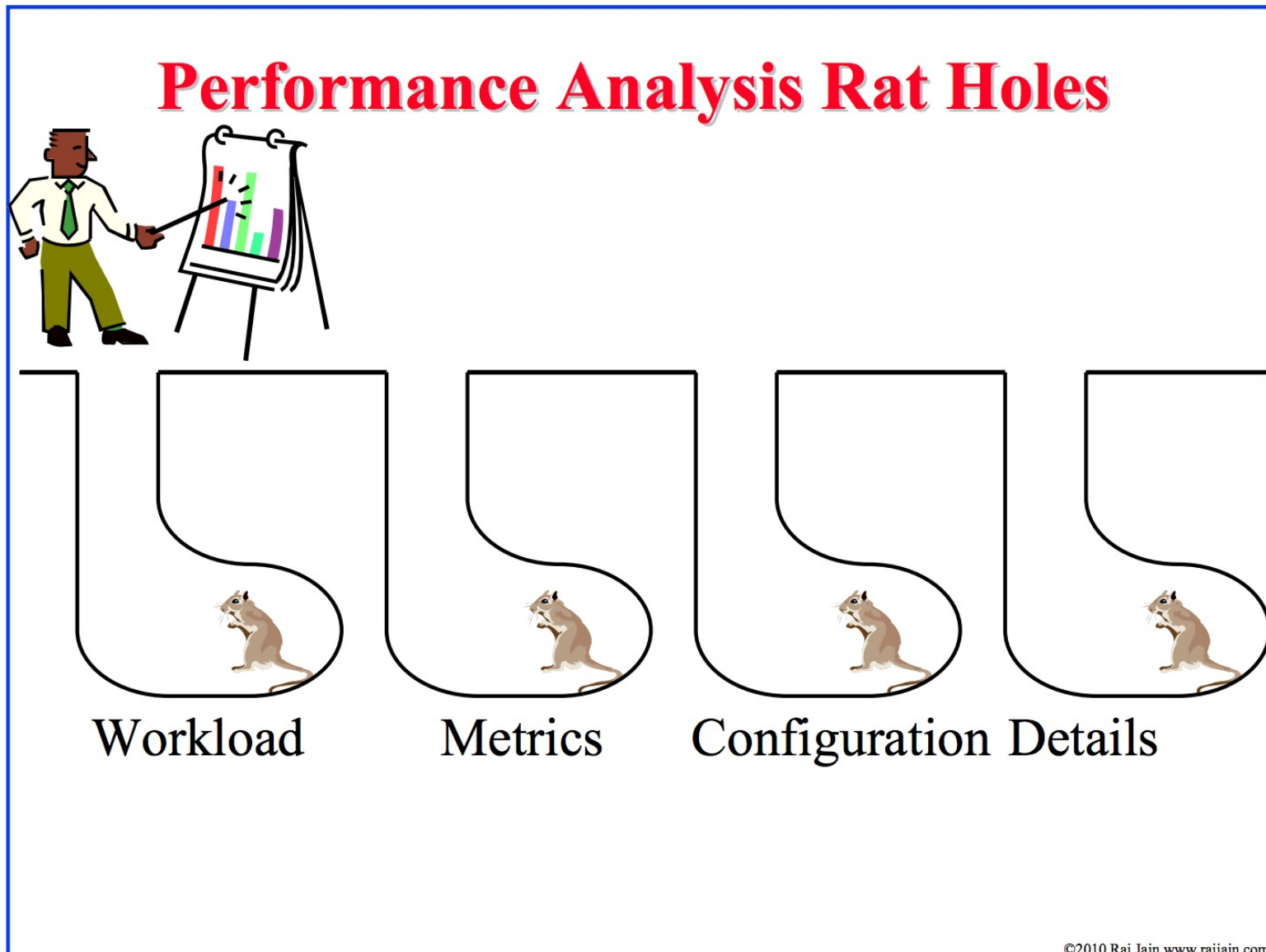
- DRAM has great capability to perform **bulk data movement and computation** internally with small changes
 - Can **exploit internal connectivity** to move data
 - Can **exploit analog computation capability**
 - ...
- Examples: RowClone, In-DRAM AND/OR, Gather/Scatter DRAM
 - RowClone: Fast and Efficient In-DRAM Copy and Initialization of Bulk Data (Seshadri et al., MICRO 2013)
 - Fast Bulk Bitwise AND and OR in DRAM (Seshadri et al., IEEE CAL 2015)
 - Gather-Scatter DRAM: In-DRAM Address Translation to Improve the Spatial Locality of Non-unit Strided Accesses (Seshadri et al., MICRO 2015)
 - "Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology" (Seshadri et al., MICRO 2017)

Weaknesses

Weaknesses

- Requires data to be mapped in the same subarray to deliver the largest benefits
 - Helps less if data movement is not within a subarray
 - Does not help if data movement is across DRAM channels
- Inter-subarray copy is very inefficient
- Causes many changes in the system stack
 - End-to-end design spans applications to circuits
 - Software-hardware cooperative solution might not always be easy to adopt
- Cache coherence and data reuse cause real overheads
- Evaluation is done solely in simulation
- Evaluation does not consider multi-chip systems
- Are these the best workloads to evaluate?

Recall: Try to Avoid Rat Holes



Thoughts and Ideas

Improvements on RowClone

Extensions and Follow-Up Work

- Can this be improved to do faster inter-subarray copy?
 - Yes, [see the LISA paper \[Chang et al., HPCA 2016\]](#)
- Can we enable data movement at smaller granularities within a bank?
 - Yes, [see the FIGARO paper \[Wang et al., MICRO 2020\]](#)
- Can this be improved to do better inter-bank copy?
 - Yes, [see the Network-on-Memory paper \[CAL 2020\]](#)
- Can similar ideas and DRAM properties be used to perform computation on data?
 - Yes, [see the Ambit paper \[Seshadri et al., MICRO 2017\]](#)

LISA: Fast Inter-Subarray Data Movement

- Kevin K. Chang, Prashant J. Nair, Saugata Ghose, Donghyuk Lee, Moinuddin K. Qureshi, and Onur Mutlu,

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

Proceedings of the 22nd International Symposium on High-Performance Computer Architecture (HPCA), Barcelona, Spain, March 2016.

[[Slides \(pptx\)](#) ([pdf](#))]

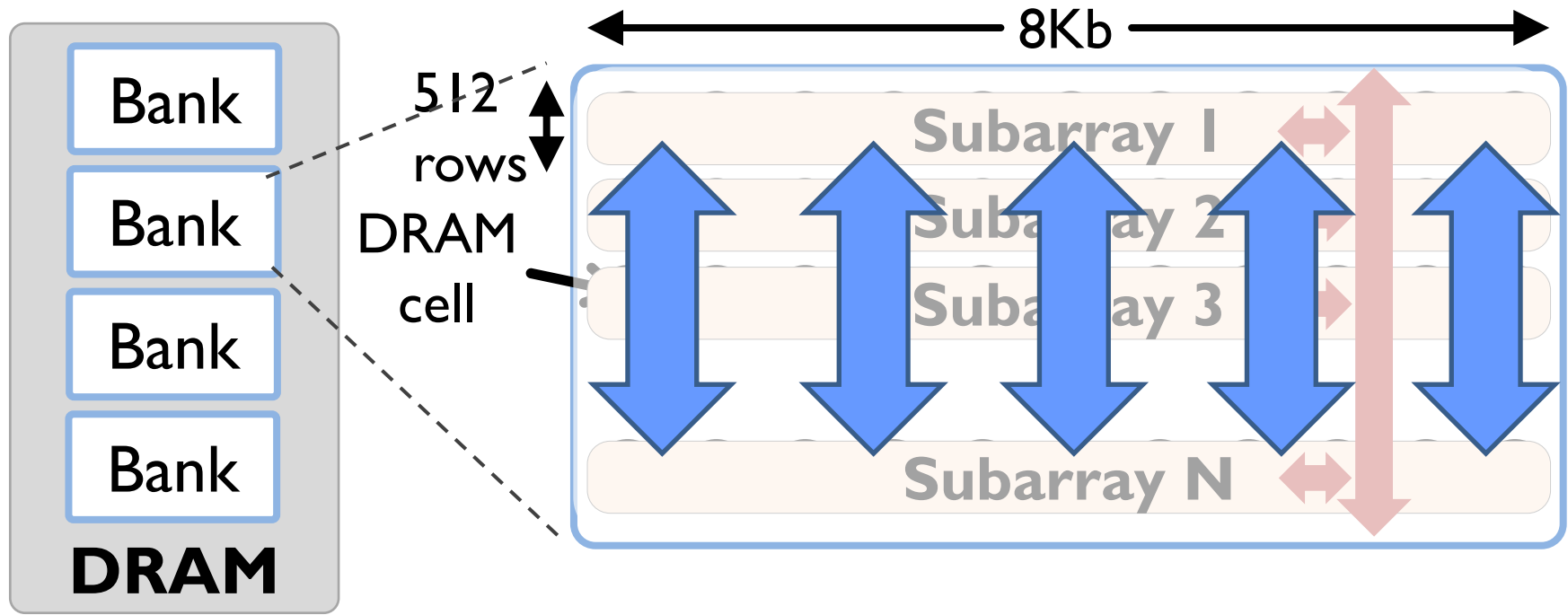
[[Source Code](#)]

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

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

[†]*Carnegie Mellon University* ^{*}*Georgia Institute of Technology*

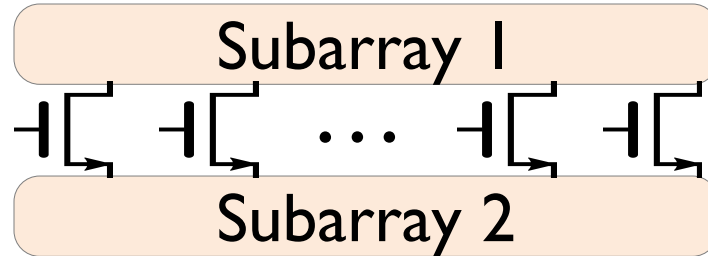
Moving Data Inside DRAM?



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

Key Idea and Applications

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



- LISA is a **versatile substrate** → new applications
 - Fast bulk data copy:** Copy latency 1.363ms→0.148ms (9.2x)
 - 66% speedup, -55% DRAM energy
 - In-DRAM caching:** Hot data access latency 48.7ns→21.5ns (2.2x)
 - 5% speedup
 - Fast precharge:** Precharge latency 13.1ns→5.0ns (2.6x)
 - 8% speedup

More on LISA

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

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

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

[†]Carnegie Mellon University ^{*}Georgia Institute of Technology

FIGARO: Fine-Grained In-DRAM Copy

- Yaohua Wang, Lois Orosa, Xiangjun Peng, Yang Guo, Saugata Ghose, Minesh Patel, Jeremie S. Kim, Juan Gómez Luna, Mohammad Sadrosadati, Nika Mansouri Ghiasi, and Onur Mutlu,
"FIGARO: Improving System Performance via Fine-Grained In-DRAM Data Relocation and Caching"
Proceedings of the 53rd International Symposium on Microarchitecture (MICRO), Virtual, October 2020.

FIGARO: Improving System Performance via Fine-Grained In-DRAM Data Relocation and Caching

Yaohua Wang^{*} Lois Orosa[†] Xiangjun Peng^{⊙*} Yang Guo^{*} Saugata Ghose^{◇‡} Minesh Patel[†]
Jeremie S. Kim[†] Juan Gómez Luna[†] Mohammad Sadrosadati[§] Nika Mansouri Ghiasi[†] Onur Mutlu^{†‡}

^{*}National University of Defense Technology [†]ETH Zürich [⊙]Chinese University of Hong Kong

[◇]University of Illinois at Urbana–Champaign [‡]Carnegie Mellon University [§]Institute of Research in Fundamental Sciences

Network-On-Memory: Fast Inter-Bank Copy

- Seyyed Hossein SeyyedAghaei Rezaei, Mehdi Modarressi, Rachata Ausavarungnirun, Mohammad Sadrosadati, Onur Mutlu, and Masoud Daneshtalab,
"NoM: Network-on-Memory for Inter-Bank Data Transfer in Highly-Banked Memories"
IEEE Computer Architecture Letters (**CAL**), to appear in 2020.

NOm: NETWORK-ON-MEMORY FOR INTER-BANK DATA TRANSFER IN HIGHLY-BANKED MEMORIES

Seyyed Hossein SeyyedAghaei Rezaei¹
Mohammad Sadrosadati³

Mehdi Modarressi^{1,3}
Onur Mutlu⁴

Rachata Ausavarungnirun²
Masoud Daneshtalab⁵

¹University of Tehran

²King Mongkut's University of Technology North Bangkok

³Institute for Research in Fundamental Sciences

⁴ETH Zürich

⁵Mälardalens University

In-DRAM Bulk Bitwise AND/OR

- Vivek Seshadri, Kevin Hsieh, Amirali Boroumand, Donghyuk Lee, Michael A. Kozuch, Onur Mutlu, Phillip B. Gibbons, and Todd C. Mowry,
"Fast Bulk Bitwise AND and OR in DRAM"
IEEE Computer Architecture Letters (***CAL***), April 2015.

Fast Bulk Bitwise AND and OR in DRAM

Vivek Seshadri*, Kevin Hsieh*, Amirali Boroumand*, Donghyuk Lee*,
Michael A. Kozuch†, Onur Mutlu*, Phillip B. Gibbons†, Todd C. Mowry*

*Carnegie Mellon University

†Intel Pittsburgh

Ambit: Bulk-Bitwise in-DRAM Computation

- Vivek Seshadri, Donghyuk Lee, Thomas Mullins, Hasan Hassan, Amirali Boroumand, Jeremie Kim, Michael A. Kozuch, Onur Mutlu, Phillip B. Gibbons, and Todd C. Mowry,
"Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology"
Proceedings of the 50th International Symposium on Microarchitecture (MICRO), Boston, MA, USA, October 2017.
[\[Slides \(pptx\) \(pdf\)\]](#) [\[Lightning Session Slides \(pptx\) \(pdf\)\]](#) [\[Poster \(pptx\) \(pdf\)\]](#)

Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology

Vivek Seshadri^{1,5} Donghyuk Lee^{2,5} Thomas Mullins^{3,5} Hasan Hassan⁴ Amirali Boroumand⁵
Jeremie Kim^{4,5} Michael A. Kozuch³ Onur Mutlu^{4,5} Phillip B. Gibbons⁵ Todd C. Mowry⁵

¹Microsoft Research India ²NVIDIA Research ³Intel ⁴ETH Zürich ⁵Carnegie Mellon University

In-DRAM Bulk Bitwise Execution Paradigm

- Vivek Seshadri and Onur Mutlu,
"In-DRAM Bulk Bitwise Execution Engine"
Invited Book Chapter in Advances in Computers, to appear
in 2020.
[[Preliminary arXiv version](#)]

In-DRAM Bulk Bitwise Execution Engine

Vivek Seshadri
Microsoft Research India
visesha@microsoft.com

Onur Mutlu
ETH Zürich
onur.mutlu@inf.ethz.ch

SIMDRAM Framework for in-DRAM Computing

- Nastaran Hajinazar, Geraldo F. Oliveira, Sven Gregorio, Joao Dinis Ferreira, Nika Mansouri Ghiasi, Minesh Patel, Mohammed Alser, Saugata Ghose, Juan Gomez-Luna, and Onur Mutlu, **["SIMDRAM: An End-to-End Framework for Bit-Serial SIMD Computing in DRAM"](#)** *Proceedings of the 26th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Virtual, March-April 2021.
[[2-page Extended Abstract](#)]
[[Short Talk Slides \(pptx\)](#) ([pdf](#))]
[[Talk Slides \(pptx\)](#) ([pdf](#))]
[[Short Talk Video](#) (5 mins)]
[[Full Talk Video](#) (27 mins)]

SIMDRAM: A Framework for Bit-Serial SIMD Processing using DRAM

*Nastaran Hajinazar ^{1,2}	*Geraldo F. Oliveira ¹	Sven Gregorio ¹	João Dinis Ferreira ¹
Nika Mansouri Ghiasi ¹	Minesh Patel ¹	Mohammed Alser ¹	Saugata Ghose ³
	Juan Gómez-Luna ¹	Onur Mutlu ¹	

¹ETH Zürich

²Simon Fraser University

³University of Illinois at Urbana–Champaign

Extensions and Follow-Up Work (II)

- Can this idea be evaluated on a real system? How?
 - Yes, [see the ComputeDRAM paper \[MICRO 2019\]](#)
- Can similar ideas be used in other types of memories? Phase Change Memory? RRAM? STT-MRAM?
 - Yes, [see the Pinatubo paper \[DAC 2016\]](#)
- Can charge sharing properties be used for other functions?
 - Yes, [see the D-RaNGe \[HPCA 2019\], DL-PUF \[HPCA 2018\], QUAC-TRNG \[ISCA 2021\] works](#)
- Can we have more efficient solutions to
 - Cache coherence (minimize overhead)
 - Data reuse after copy and initialization

Pinatubo: PCM RowClone and Bitwise Ops

Pinatubo: A Processing-in-Memory Architecture for Bulk Bitwise Operations in Emerging Non-volatile Memories

Shuangchen Li^{1*}, Cong Xu², Qiaosha Zou^{1,5}, Jishen Zhao³, Yu Lu⁴, and Yuan Xie¹

University of California, Santa Barbara¹, Hewlett Packard Labs²

University of California, Santa Cruz³, Qualcomm Inc.⁴, Huawei Technologies Inc.⁵
{shuangchenli, yuanxie}@ece.ucsb.edu¹

RowClone Demonstration in Real DRAM Chips

ComputeDRAM: In-Memory Compute Using Off-the-Shelf DRAMs

Fei Gao

feig@princeton.edu

Department of Electrical Engineering
Princeton University

Georgios Tziantzioulis

georgios.tziantzioulis@princeton.edu

Department of Electrical Engineering
Princeton University

David Wentzlaff

wentzlaf@princeton.edu

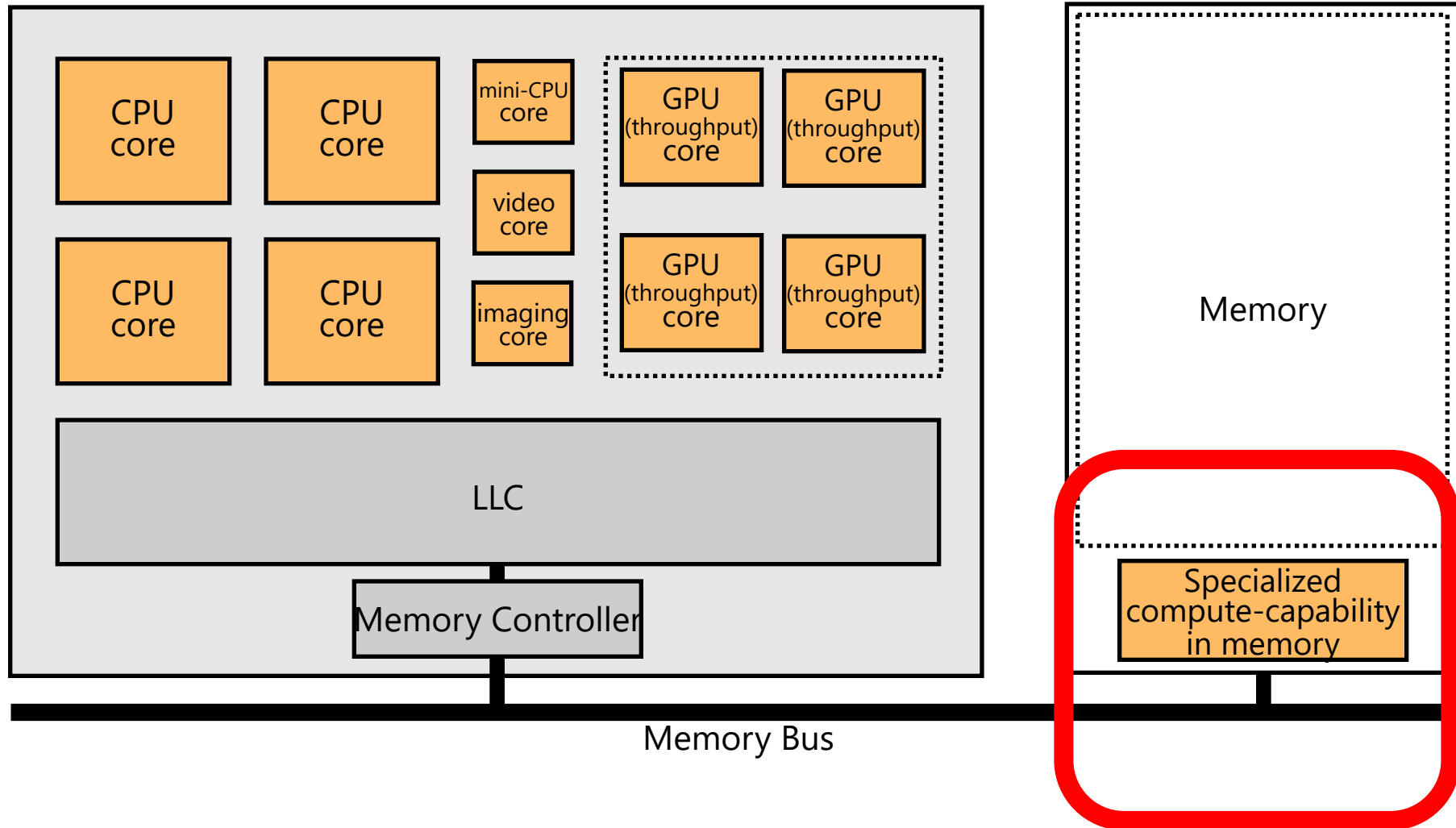
Department of Electrical Engineering
Princeton University

Takeaways

Key Takeaways

- A novel method to accelerate data copy and initialization
- Simple and effective
- Hardware/software cooperative
- Good potential for work building on it to extend it
 - To different granularities
 - To make things more efficient and effective
 - Many works have already built on the paper (see LISA, FIGARO, NoM, Ambit, ComputeDRAM, and other works in Google Scholar)
- Easy to read and understand paper

RowClone: Memory as an Accelerator



Memory similar to a "conventional" accelerator

Mindset: Processing using DRAM

- DRAM has great capability to perform **bulk data movement and computation** internally with small changes
 - Can **exploit internal connectivity** to move data
 - Can **exploit analog computation capability**
 - ...
- Examples: RowClone, In-DRAM AND/OR, Gather/Scatter DRAM
 - RowClone: Fast and Efficient In-DRAM Copy and Initialization of Bulk Data (Seshadri et al., MICRO 2013)
 - Fast Bulk Bitwise AND and OR in DRAM (Seshadri et al., IEEE CAL 2015)
 - Gather-Scatter DRAM: In-DRAM Address Translation to Improve the Spatial Locality of Non-unit Strided Accesses (Seshadri et al., MICRO 2015)
 - "Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology" (Seshadri et al., MICRO 2017)

Open Discussion

Discussion Starters

- Thoughts on the previous ideas?
- How practical is this?
- Will the problem become bigger and more important over time?
- Will the solution become more important over time?
- Are other solutions better?
- Is this solution clearly advantageous or opposite in some cases?

General Issues

- Data mapping and interleaving
- Data coherence between caches and DRAM
- Data reuse
- All are issues with Processing in Memory

PIM Review and Open Problems

A Modern Primer on Processing in Memory

Onur Mutlu^{a,b}, Saugata Ghose^{b,c}, Juan Gómez-Luna^a, Rachata Ausavarungnirun^d

SAFARI Research Group

^a*ETH Zürich*

^b*Carnegie Mellon University*

^c*University of Illinois at Urbana-Champaign*

^d*King Mongkut's University of Technology North Bangkok*

Onur Mutlu, Saugata Ghose, Juan Gomez-Luna, and Rachata Ausavarungnirun,
"A Modern Primer on Processing in Memory"
*Invited Book Chapter in **Emerging Computing: From Devices to Systems - Looking Beyond Moore and Von Neumann**, Springer, to be published in 2021.*

More on RowClone

- Vivek Seshadri, Yoongu Kim, Chris Fallin, Donghyuk Lee, Rachata Ausavarungnirun, Gennady Pekhimenko, Yixin Luo, Onur Mutlu, Michael A. Kozuch, Phillip B. Gibbons, and Todd C. Mowry,
"RowClone: Fast and Energy-Efficient In-DRAM Bulk Data Copy and Initialization"
Proceedings of the 46th International Symposium on Microarchitecture (MICRO), Davis, CA, December 2013. [[Slides \(pptx\)](#)] [[pdf](#)] [[Lightning Session Slides \(pptx\)](#)] [[pdf](#)] [[Poster \(pptx\)](#)] [[pdf](#)]

RowClone: Fast and Energy-Efficient In-DRAM Bulk Data Copy and Initialization

Vivek Seshadri Yoongu Kim Chris Fallin* Donghyuk Lee
vseshadr@cs.cmu.edu yoongukim@cmu.edu cfallin@c1f.net donghyuk1@cmu.edu

Rachata Ausavarungnirun Gennady Pekhimenko Yixin Luo
rachata@cmu.edu gpekhime@cs.cmu.edu yixinluo@andrew.cmu.edu

Onur Mutlu Phillip B. Gibbons† Michael A. Kozuch† Todd C. Mowry
onur@cmu.edu phillip.b.gibbons@intel.com michael.a.kozuch@intel.com tcm@cs.cmu.edu

Carnegie Mellon University †Intel Pittsburgh

RowClone

**Fast and Energy-Efficient In-DRAM
Bulk Data Copy and Initialization**

Vivek Seshadri

Y. Kim, C. Fallin, D. Lee, R. Ausavarungnirun,
G. Pekhimenko, Y. Luo, O. Mutlu,
P. B. Gibbons, M. A. Kozuch, T. C. Mowry

SAFARI

Carnegie Mellon



Some History: RowClone

RowClone: Historical Perspective

- This work is likely the first example of “minimally changing DRAM chips” to perform data movement and computation
 - Surprising that it was done as late as 2013!
- It led to a body of work on in-DRAM (and in-NVM) computation with “hopefully small” changes
- Work building on RowClone still continues
- Initially, it was dismissed by some reviewers
 - Rejected from ISCA 2013 conference

One Review (ISCA 2013 Submission)

PAPER STRENGTHS

The paper includes a well written background on DRAM organization/operation. The proposed technique is simple and elegant; it nicely exploits key circuit-level characteristics of DRAM designs and minimizes the changes necessary to commodity DRAM chips.

PAPER WEAKNESSES

I am concerned on the applicability of the technique and found the evaluation to be unconvincing in terms of motivating the work as well as quantifying the potential benefit. Details on how to efficiently manage the coherence between the cache hierarchy and DRAM to enable the proposed technique are glossed over, but in my opinion are critical to the narrative.

Another Review and Rebuttal

DETAILED COMMENTS

The paper proposes a simple and not new idea, block copy in a DRAM, and the creates a complete

Reviewer B mentions that our idea is "not new". An explicit reference by the reviewer would be helpful here. While the reviewer may be referring to one of the patents that we cite in our paper (citations 2, 6, 25, 26, 27 in the paper), these patents are at a superficial level and do *not* provide a concrete mechanism. In contrast, we propose three concrete mechanisms and provide details on the most important architectural and microarchitectural modifications required at the DRAM chip, the memory controller, and the CPU to enable a system that supports the mechanisms. We also analyze their latency, hardware overhead, power, and performance in detail. We are not aware of any prior work that achieves this.

ISCA 2013 Submission

ISCA40

Paper #295

onur@cmu.edu [Profile](#) | [Help](#) | [Sign out](#)



Main



[Edit](#)

#268 Papers #353

(All)

Search

#295 RowClone: Fast and Efficient In-DRAM Copy and Initialization of Bulk Data

COMMENT

NOTIFICATION

If selected, you will receive email when updated comments are available for this paper.

+ OTHER CONFLICTS

Rejected



1014kB

Thursday 22 Nov 2012 12:11:45am EST

| 0fd459a9adc6194cda028a394d2e4d929f662f32

You are an **author** of this paper.

– ABSTRACT

Many programs initialize or copy large amounts of memory data. Initialization and copying are

+ AUTHORS

V. Seshadri, Y. Kim, D. Lee, C. Fallin, R. Ausavarungnirun, G. Pekhimenko, Y. Luo, O. Mutlu,

[Review #295A](#)

[Review #295B](#)

[Review #295C](#)

OveMer Nov WriQua RevConAnd

3

4

5

3

4

3

4

3

3

4

4

3

Yet Later... in ISCA 2015...

Profiling a warehouse-scale computer

Svilen Kanev[†]
Harvard University

Juan Pablo Darago[†]
Universidad de Buenos Aires

Kim Hazelwood[†]
Yahoo Labs

Parthasarathy Ranganathan
Google

Tipp Moseley
Google

Gu-Yeon Wei
Harvard University

David Brooks
Harvard University

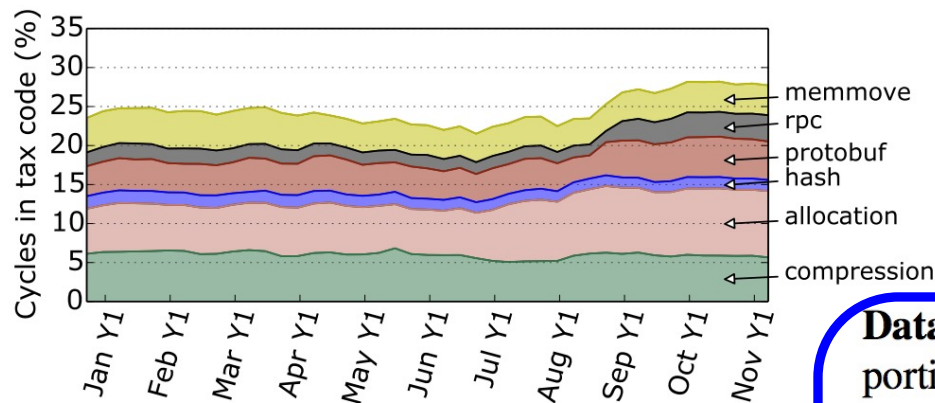


Figure 4: 22-27% of WSC cycles are spent in different components of “datacenter tax”.

we see common building blocks once we aggregate sampled profile data across many applications running in a datacenter. In this section, we quantify the performance impact of the **datacenter tax**, and argue that its components are prime candidates for hardware acceleration in future datacenter SoCs.

Data movement In fact, RPCs are by far not the only code portions that do data movement. We also tracked all calls to the `memcpy()` and `memmove()` library functions to estimate the amount of time spent on explicit data movement (i.e., exposed through a simple API). This is a conservative estimate because it does not track inlined or explicit copies. Just the variants of these two library functions represent 4-5% of datacenter cycles.

Recent work in performing data movement in DRAM [45] could optimize away this piece of tax.

MICRO 2013 Submission

#206 RowClone: Fast and Energy-Efficient In-DRAM Bulk Data Copy and Initialization

ATION
ceive

e for

Accepted



1947kB

Friday 31 May 2013 1:48:46pm PDT |
fd8423acdd9a222280302355899340083e5a40b1

You are an **author** of this paper.

+ **ABSTRACT**

Bulk data copy and initialization operations are frequently triggered by several system level operations in modern systems. Despite the fact that these operations do not require [\[more\]](#)

+ **AUTHORS**

V. Seshadri, Y. Kim, C. Fallin, D. Lee, R. Ausavarungrun, G. Pekhimenko, Y. Luo, O. Mutlu, P. Gibbons, M. Kozuch, T. Mowry
[\[details\]](#)

+ **TOPICS**

	OveMer	Nov	WriQua	RevExp
Review #206A	5	4	4	4
Review #206B	4	2	4	4
Review #206C	3	4	4	4
Review #206D	3	3	4	3
Review #206E	4	3	5	3

More History: Ambit

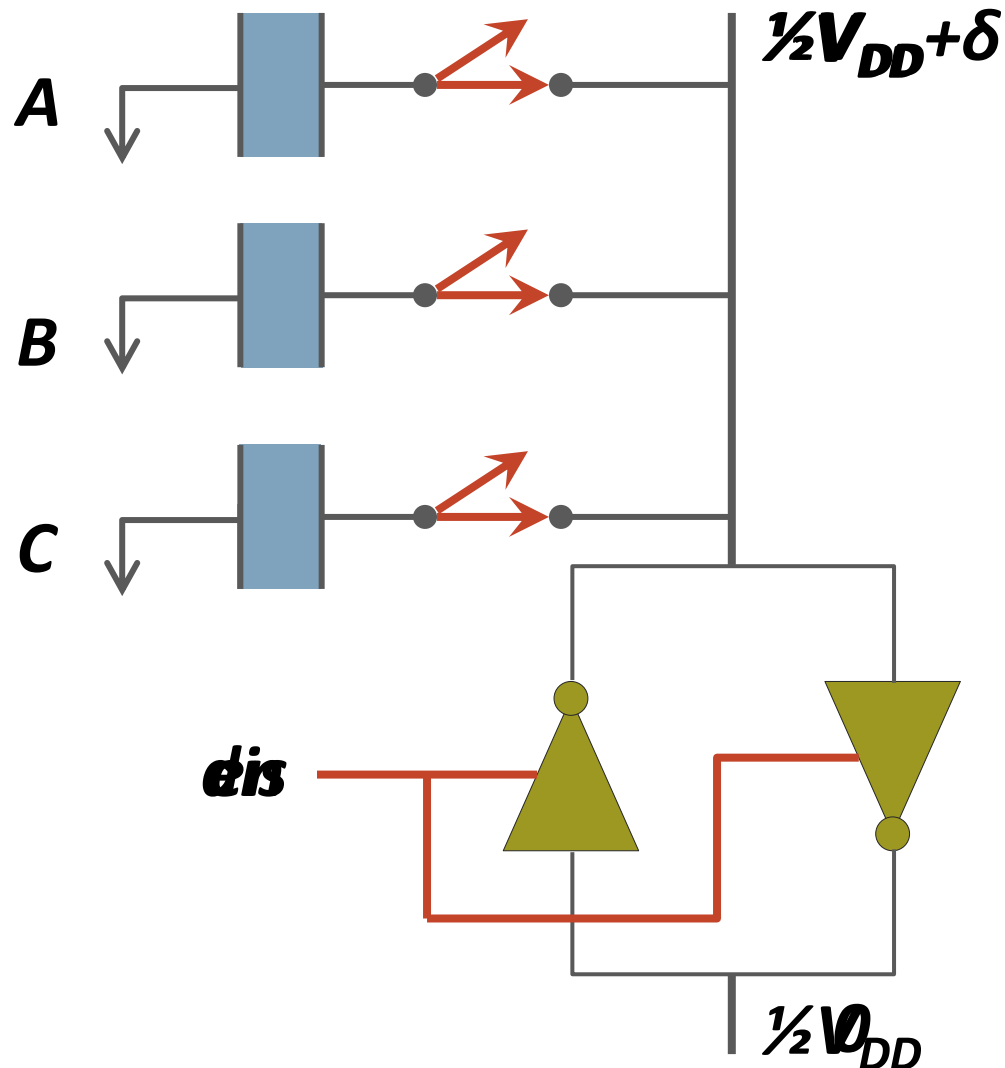
Ambit

- First work on performing bulk bitwise operations in DRAM
 - By exploiting analog computation capability of bitlines
 - Extends and completes our IEEE CAL 2015 paper
- **Disruptive** -- spans algorithms to circuits/devices
 - Requires hardware/software cooperation for adoption
- Led to a large amount of work in similar approaches in DRAM and NVM
 - The work continues to build
- Initially, it was dismissed by many reviewers
 - Rejected from 4 conferences!

Ambit: In-Memory Bulk Bitwise Execution

- We can support in-DRAM COPY, ZERO, AND, OR, NOT, MAJ
- At low cost
- Using analog computation capability of DRAM
 - Idea: activating multiple rows performs computation
- 30-60X performance and energy improvement
 - Seshadri+, “Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology,” MICRO 2017.
- New memory technologies enable even more opportunities
 - Memristors, resistive RAM, phase change mem, STT-MRAM, ...
 - Can operate on data with minimal movement

In-DRAM AND/OR: Triple Row Activation



Final State
 $AB + BC + AC$

**$C(A + B) +$
 $\sim C(AB)$**

In-DRAM Bulk Bitwise AND/OR Operation

- **BULKAND A, B → C**
 - Semantics: Perform a bitwise AND of two rows A and B and store the result in row C
 - R0 – reserved zero row, R1 – reserved one row
 - D1, D2, D3 – Designated rows for triple activation
1. RowClone A into D1
 2. RowClone B into D2
 3. RowClone R0 into D3
 4. ACTIVATE D1,D2,D3
 5. RowClone Result into C

More on In-DRAM Bulk AND/OR

- Vivek Seshadri, Kevin Hsieh, Amirali Boroumand, Donghyuk Lee, Michael A. Kozuch, Onur Mutlu, Phillip B. Gibbons, and Todd C. Mowry,
"Fast Bulk Bitwise AND and OR in DRAM"
IEEE Computer Architecture Letters (***CAL***), April 2015.

Fast Bulk Bitwise AND and OR in DRAM

Vivek Seshadri*, Kevin Hsieh*, Amirali Boroumand*, Donghyuk Lee*,
Michael A. Kozuch†, Onur Mutlu*, Phillip B. Gibbons†, Todd C. Mowry*

*Carnegie Mellon University

†Intel Pittsburgh

In-DRAM NOT: Dual Contact Cell

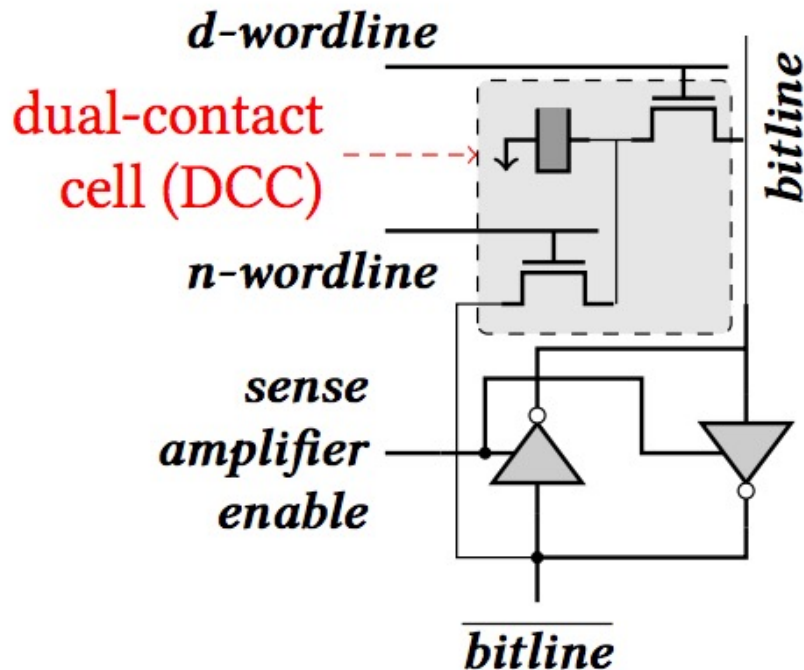


Figure 5: A dual-contact cell connected to both ends of a sense amplifier

Idea:
Feed the
negated value
in the sense amplifier
into a special row

In-DRAM NOT Operation

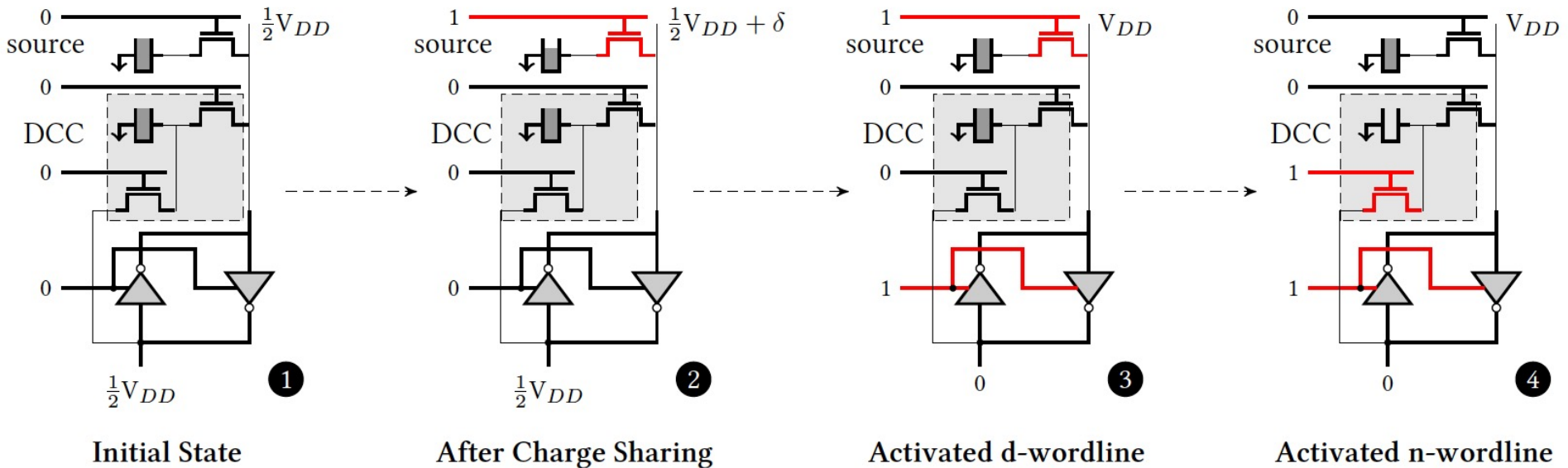


Figure 5: Bitwise NOT using a dual contact capacitor

Seshadri+, "Ambit: In-Memory Accelerator for Bulk Bitwise Operations using Commodity DRAM Technology," MICRO 2017.

Performance: In-DRAM Bitwise Operations

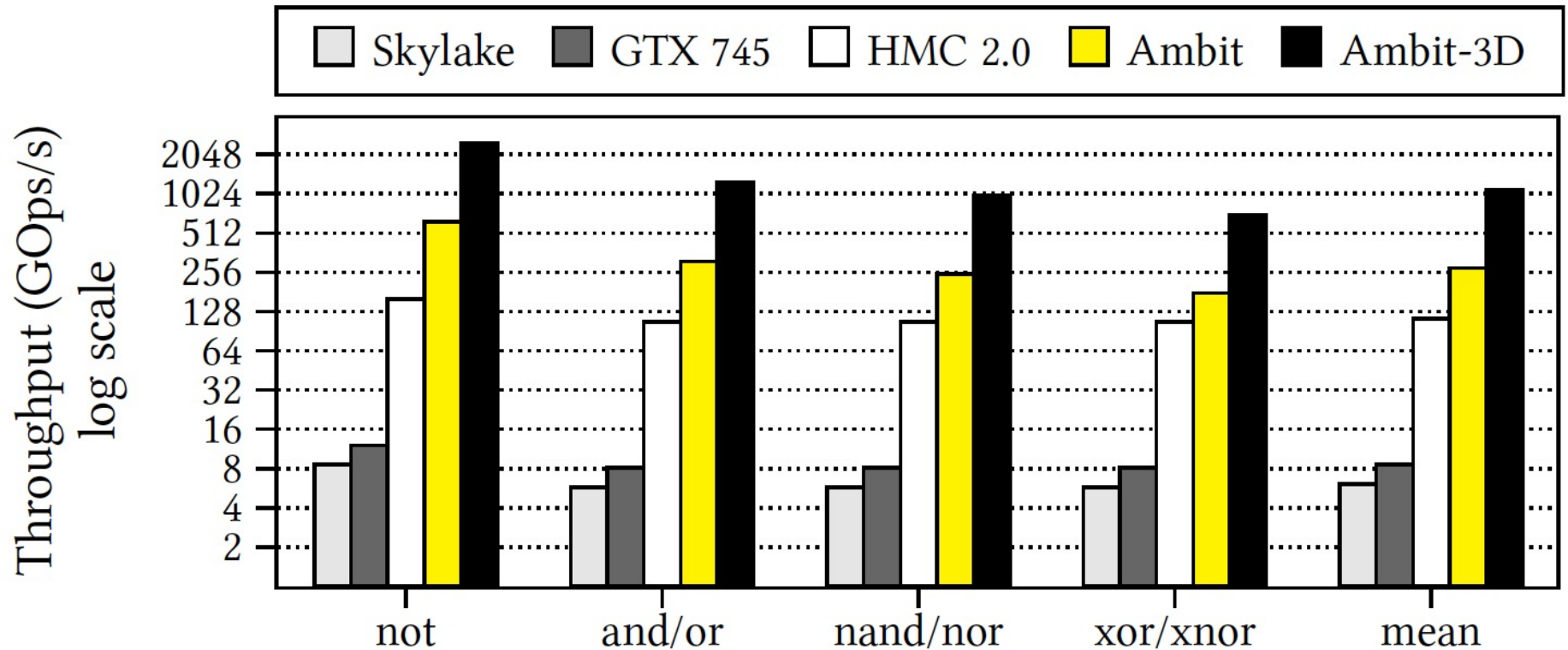


Figure 9: Throughput of bitwise operations on various systems.

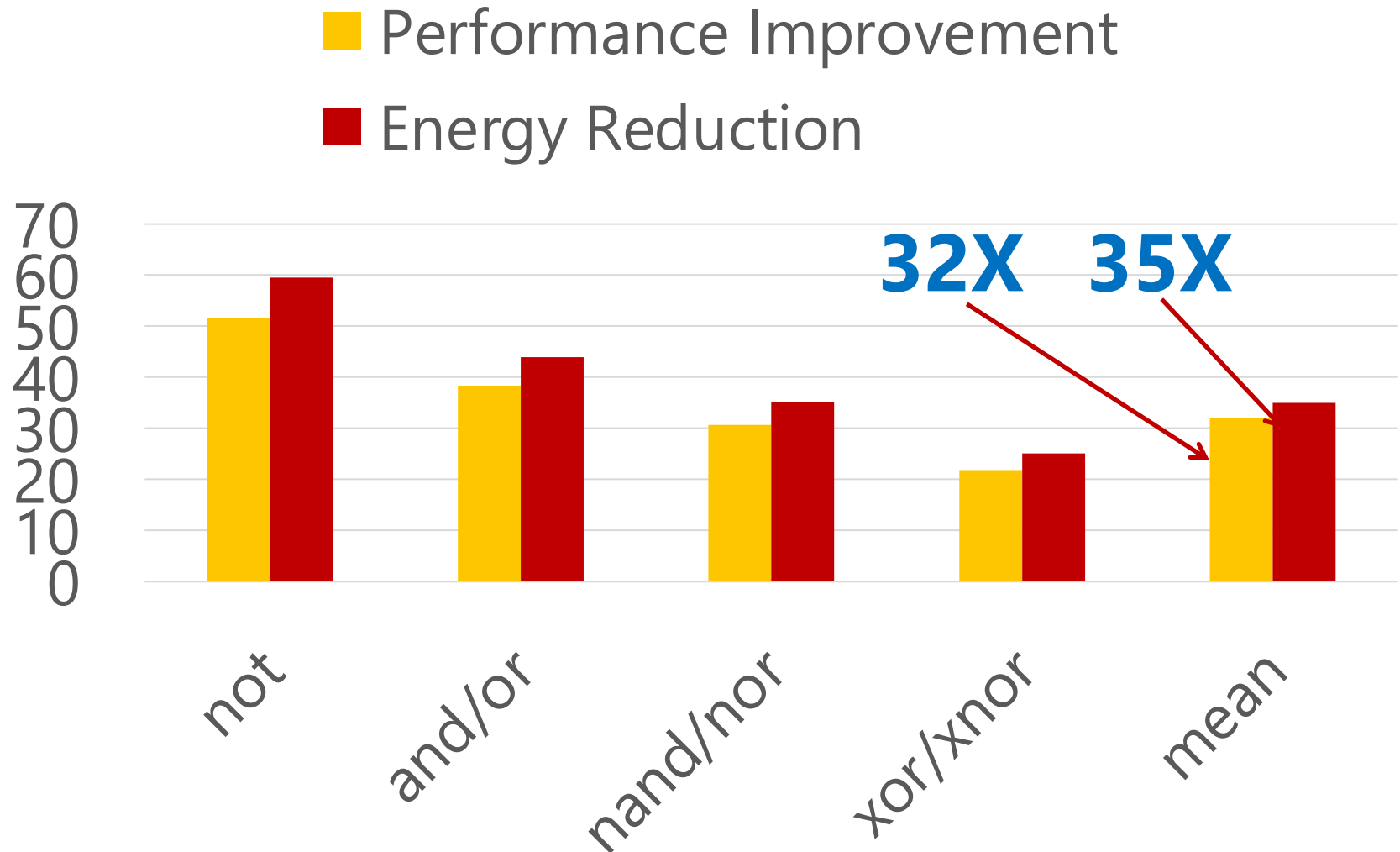
Energy of In-DRAM Bitwise Operations

	Design	not	and/or	nand/nor	xor/xnor
DRAM & Channel Energy (nJ/KB)	DDR3	93.7	137.9	137.9	137.9
	Ambit	1.6	3.2	4.0	5.5
	(↓)	59.5X	43.9X	35.1X	25.1X

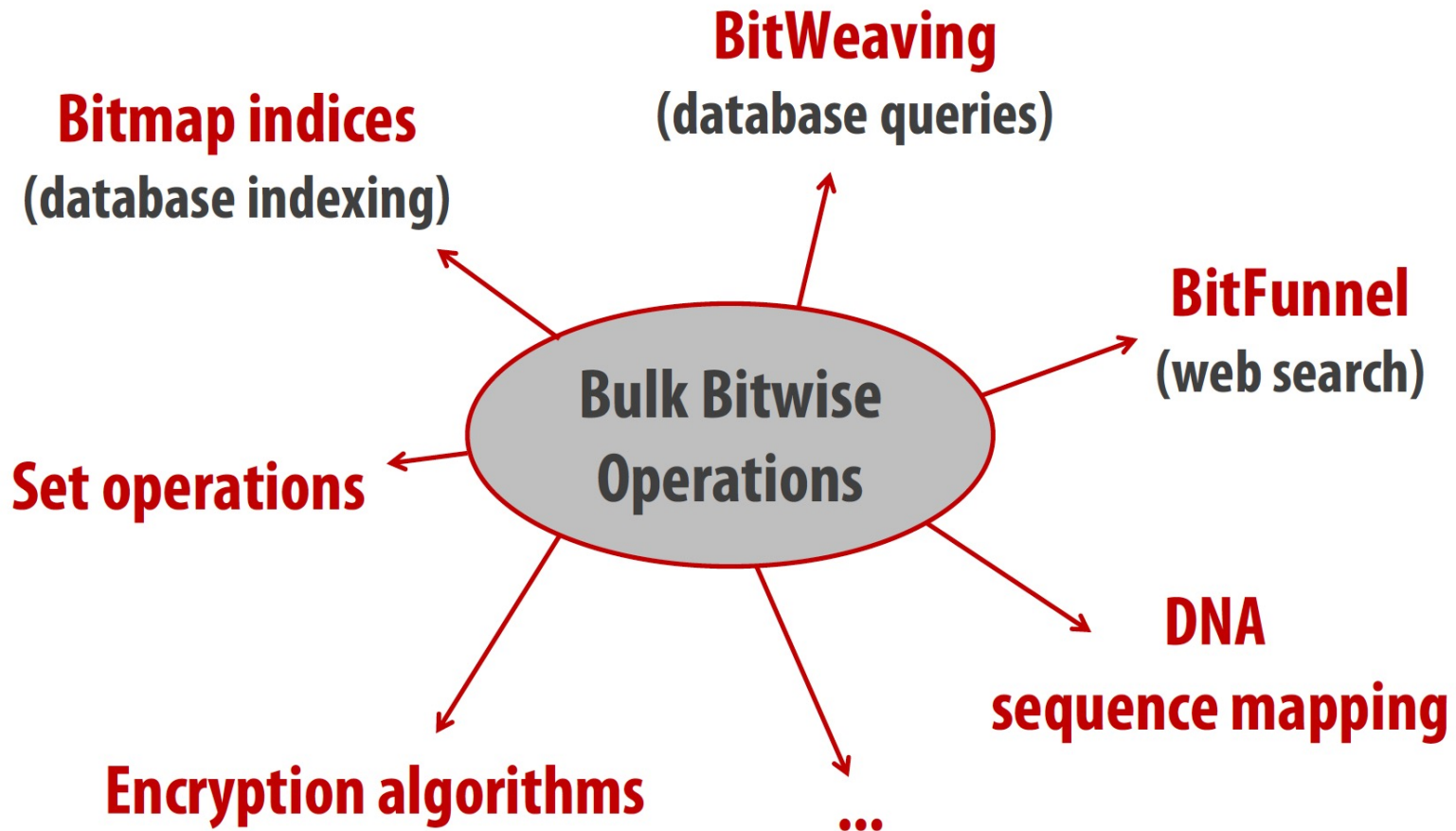
Table 3: Energy of bitwise operations. (↓) indicates energy reduction of Ambit over the traditional DDR3-based design.

Seshadri+, "Ambit: In-Memory Accelerator for Bulk Bitwise Operations using Commodity DRAM Technology," MICRO 2017.

Ambit vs. DDR3: Performance and Energy



Bulk Bitwise Operations in Workloads



Example Data Structure: Bitmap Index

- Alternative to B-tree and its variants
- Efficient for performing *range queries* and *joins*
- **Many bitwise operations to perform a query**

age < 18 18 < age < 25 25 < age < 60 age > 60

Bitmap 1

Bitmap 2

Bitmap 3

Bitmap 4

Performance: Bitmap Index on Ambit

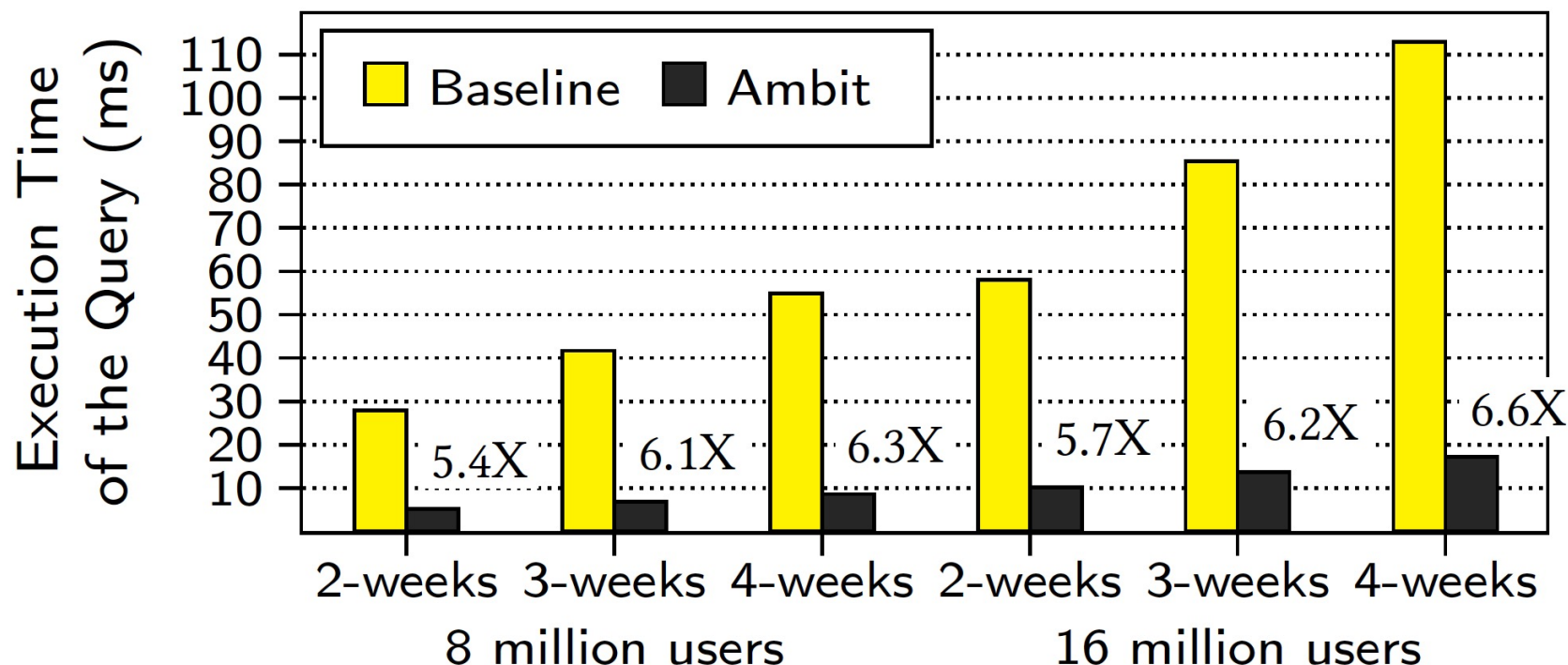


Figure 10: Bitmap index performance. The value above each bar indicates the reduction in execution time due to Ambit.

>5.4-6.6X Performance Improvement

Performance: BitWeaving on Ambit

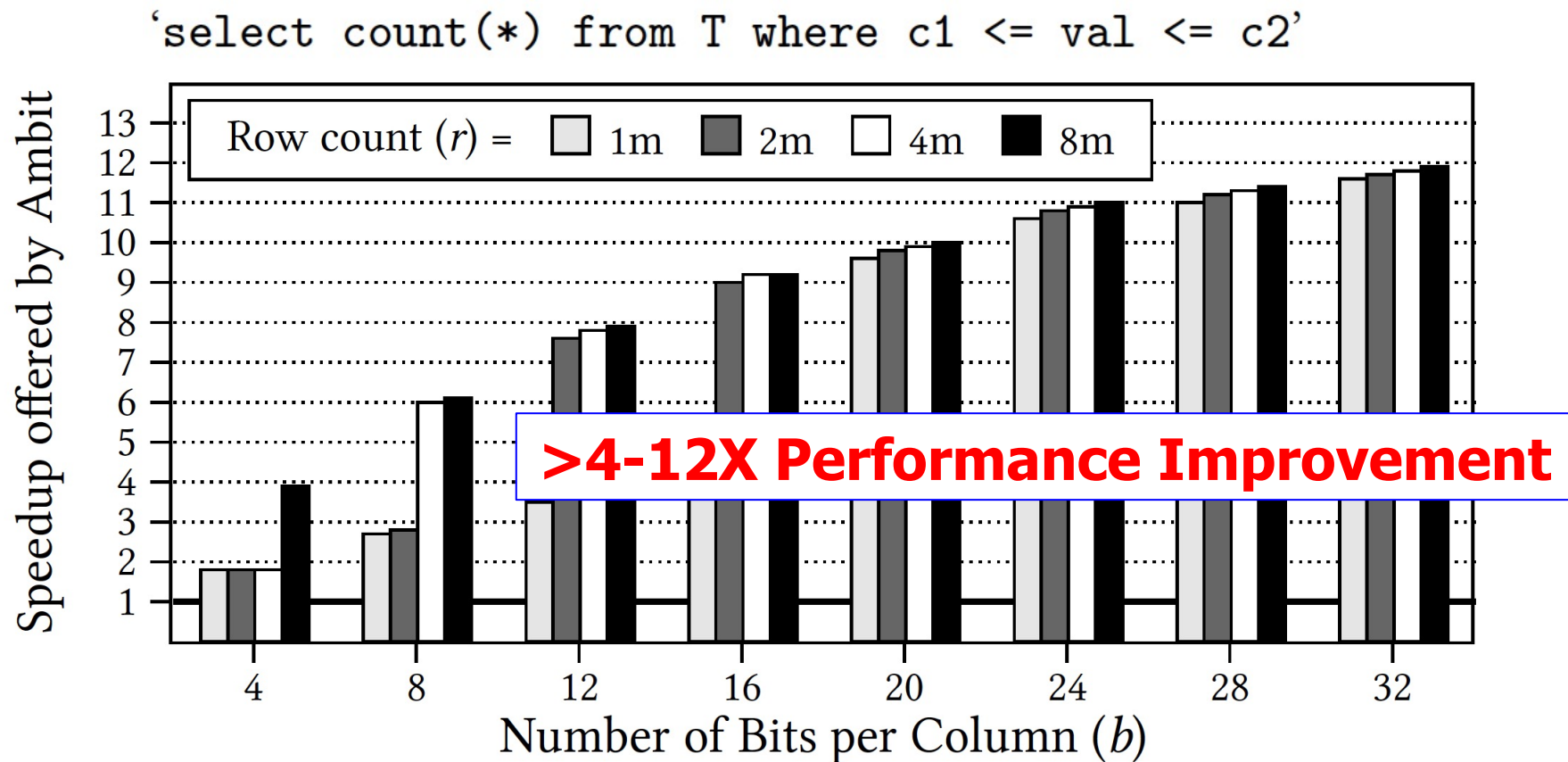


Figure 11: Speedup offered by Ambit over baseline CPU with SIMD for BitWeaving

Seshadri+, "Ambit: In-Memory Accelerator for Bulk Bitwise Operations using Commodity DRAM Technology," MICRO 2017.

More on In-DRAM Bulk AND/OR

- Vivek Seshadri, Kevin Hsieh, Amirali Boroumand, Donghyuk Lee, Michael A. Kozuch, Onur Mutlu, Phillip B. Gibbons, and Todd C. Mowry,
"Fast Bulk Bitwise AND and OR in DRAM"
IEEE Computer Architecture Letters (***CAL***), April 2015.

Fast Bulk Bitwise AND and OR in DRAM

Vivek Seshadri*, Kevin Hsieh*, Amirali Boroumand*, Donghyuk Lee*,
Michael A. Kozuch†, Onur Mutlu*, Phillip B. Gibbons†, Todd C. Mowry*

*Carnegie Mellon University

†Intel Pittsburgh

More on In-DRAM Bitwise Operations

- Vivek Seshadri et al., “[Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology](#),” MICRO 2017.

Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology

Vivek Seshadri^{1,5} Donghyuk Lee^{2,5} Thomas Mullins^{3,5} Hasan Hassan⁴ Amirali Boroumand⁵
Jeremie Kim^{4,5} Michael A. Kozuch³ Onur Mutlu^{4,5} Phillip B. Gibbons⁵ Todd C. Mowry⁵

¹Microsoft Research India ²NVIDIA Research ³Intel ⁴ETH Zürich ⁵Carnegie Mellon University

More on In-DRAM Bulk Bitwise Execution

- Vivek Seshadri and Onur Mutlu,
"In-DRAM Bulk Bitwise Execution Engine"
Invited Book Chapter in Advances in Computers, to appear
in 2020.
[[Preliminary arXiv version](#)]

In-DRAM Bulk Bitwise Execution Engine

Vivek Seshadri
Microsoft Research India
`visesha@microsoft.com`

Onur Mutlu
ETH Zürich
`onur.mutlu@inf.ethz.ch`

SIMDRAM Framework

- Nastaran Hajinazar, Geraldo F. Oliveira, Sven Gregorio, Joao Dinis Ferreira, Nika Mansouri Ghiasi, Minesh Patel, Mohammed Alser, Saugata Ghose, Juan Gomez-Luna, and Onur Mutlu, [**"SIMDRAM: An End-to-End Framework for Bit-Serial SIMD Computing in DRAM"**](#) *Proceedings of the 26th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Virtual, March-April 2021.
[[2-page Extended Abstract](#)]
[[Short Talk Slides \(pptx\)](#) ([pdf](#))]
[[Talk Slides \(pptx\)](#) ([pdf](#))]
[[Short Talk Video](#) (5 mins)]
[[Full Talk Video](#) (27 mins)]

SIMDRAM: A Framework for Bit-Serial SIMD Processing using DRAM

*Nastaran Hajinazar ^{1,2}	*Geraldo F. Oliveira ¹	Sven Gregorio ¹	João Dinis Ferreira ¹
Nika Mansouri Ghiasi ¹	Minesh Patel ¹	Mohammed Alser ¹	Saugata Ghose ³
	Juan Gómez-Luna ¹	Onur Mutlu ¹	

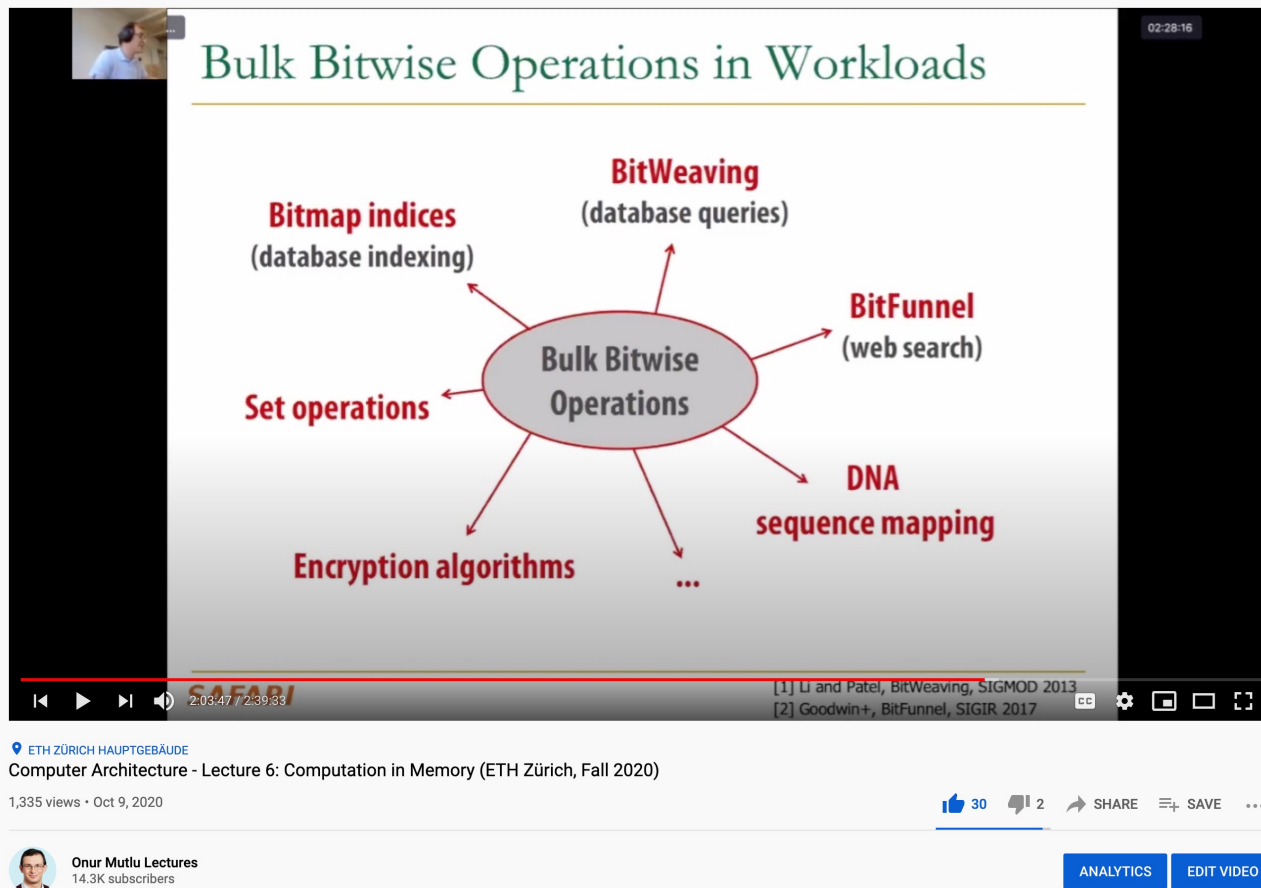
¹ETH Zürich

²Simon Fraser University

³University of Illinois at Urbana–Champaign

More on Ambit and Computation-in-Memory

- Computer Architecture, Fall 2020, Lecture 6
 - ❑ **Computation in Memory** (ETH Zürich, Fall 2020)
 - ❑ <https://www.youtube.com/watch?v=oGcZAGwfEUE&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=12>



The video player displays a slide titled "Bulk Bitwise Operations in Workloads". The slide features a central grey oval labeled "Bulk Bitwise Operations" with arrows pointing to six surrounding text blocks: "Bitmap indices (database indexing)", "BitWeaving (database queries)", "BitFunnel (web search)", "DNA sequence mapping", "Encryption algorithms", and "Set operations". Below the slide, the video player interface shows a progress bar at 2:03:47 / 2:39:33. The video title is "Computer Architecture - Lecture 6: Computation in Memory (ETH Zürich, Fall 2020)" and it has 1,335 views as of Oct 9, 2020. The channel is "Onur Mutlu Lectures" with 14.3K subscribers. The video player also includes a "SAFARI" watermark and a list of references: [1] Li and Patel, BitWeaving, SIGMOD 2013 and [2] Goodwin+, BitFunnel, SIGIR 2017.

Ambit

- First work on performing bulk bitwise operations in DRAM
 - By exploiting analog computation capability of bitlines
 - Extends and completes our IEEE CAL 2015 paper
- **Disruptive** -- spans algorithms to circuits/devices
 - Requires hardware/software cooperation for adoption
- Led to a large amount of work in similar approaches in DRAM and NVM
 - The work continues to build
- Initially, it was dismissed by many reviewers
 - Rejected from 4 conferences!


We Have a Mindset Issue...

- There are many other similar examples from reviews...
 - For many other papers...
- And, we are not even talking about JEDEC yet...
- How do we fix the mindset problem?
- By doing more research, education, implementation in alternative processing paradigms

We need to work on enabling the better future...

ISCA 2016: Rejected

Buddy RAM: Fast and Efficient Bulk Bitwise Operations Using DRAM

Rejected  2006kB 23 Nov 2015 11:30:23pm EST · 7f7234da178e644380275ce12a4f539ef45c4418

You are an **author** of this paper.

► Abstract

Many data structures (e.g., database bitmap indices) rely on fast bitwise operations on large bit vectors to achieve high performance. Unfortunately, the throughput of such bulk [\[more\]](#)

► Authors

V. Seshadri, D. Lee, T. Mullins, A. Boroumand, J. Kim, M. Kozuch, O. Mutlu, P. Gibbons, T. Mowry [\[details\]](#)

► Topics and Options

	RelISC	OveMerPos	RevConAnd	Nov	WriQua
Review #171A	3	4	4	2	3
Review #171B	2	4	3	3	4
Review #171C	3	4	4	2	3
Review #171D	3	5	2	2	3
Review #171E	2	3	2	3	3

MICRO 2016: Rejected



Submission (1662kB) 10 Apr 2016 9:32:31pm EDT ·
e518c6a8916109492574858db80a6184fe61ca0c

► **Abstract**

Certain widely-used data structures
(e.g., bitmap indices) rely on

[\[more\]](#)

▼ **Authors**

Vivek Seshadri (CMU)
<vseshadr@cs.cmu.edu>
Donghyuk Lee (NVIDIA Research)
<donghyuk1@cmu.edu>
Thomas Mullins (Intel)
<thomas.p.mullins@intel.com>
Amirali Boroumand (CMU)
Jeremie Kim (CMU)
Michael A. Kozuch (Intel)
<michael.a.kozuch@intel.com>
Onur Mutlu (CMU/ETH)
<omutlu@gmail.com>
Phillip B. Gibbons (CMU)
<gibbons@cs.cmu.edu>
Todd C. Mowry (CMU) <tc>

► **Topics**

Rejected · You are an **author** of this paper.

	Pos	Reb	Ove	OveMer	RevExp	Nov	WriQua
Review #249A	2	2	4	3	3		
Review #249B	4	4	3	3	5		
Review #249C	2	3	4	2	3		
Review #249D	5	5	2	3	3		
Review #249E	5	5	2	2	3		
Review #249F	3	3	3	3	4		

HPCA 2017: Rejected

1)~significantly improves the performance of queries in applications that use bitmap indices for fast analytics, and
2)~makes bit vectors more attractive than red-black trees to represent sets. We believe Buddy can trigger programmers to redesign applications to use bitwise operations with the goal of achieving high performance and efficiency.

Rejected · You are an **author** of this paper.

	OveMer	RevExp	WriQua	ExpMet	Nov
Review #119A	1	2	3	2	2
Review #119B	4	1	4	4	3
Review #119C	4	4	4	4	4
Review #119D	3	1	4	4	3
Review #119E	3	2	5	4	4

1 Comment: [Response \(V. Seshadri\)](#)

ISCA 2017: Rejected

Rejected



Submission ⌚ 19 Nov 2016 12:03:02am EST ·

📌 3eea263e35e53552851cab5225162776f809eaa

► Abstract

Bitwise operations are an important component of

► Authors

V. Seshadri, D. Lee, T. Mullins, H. Hassan, A. Boroumand, J. Kim, M. Kozuch, O. Mutlu, P. Gibbons, T. Mowry [\[details\]](#)

[\[more\]](#)

► Topics and Options

	PosRebOve	OveMer	Nov	WriQua	RevExp
Review #162A	1	2	2	4	5
Review #162B	2	2	3	3	3
Review #162C	4	4	3	4	4
Review #162D	3	3	3	4	4
Review #162E	4	4	3	4	3

Ambit Sounds Good, No?

Review from ISCA 2016

Paper summary

The paper proposes to extend DRAM to include bulk, bit-wise logical operations directly between rows within the DRAM.

Strengths

- Very clever/novel idea.
 - Great potential speedup and efficiency gains.
-

Weaknesses

- Probably won't ever be built. Not practical to assume DRAM manufacturers will change DRAM in this way.

Very Interesting and Novel, BUT ...

Comments for the authors

I found this idea very interesting and novel. In particular, while there have been many works proposing moving computation closer to memory, I'm not aware of any work which proposes to leverage the DRAM rows themselves to implement the computation. The benefits to this approach are large in that no actual logic is used to implement the logical functions. Further the operation occurs in parallel across the whole row, a huge degree of data parallelism.

... This Will Never Get Implemented

- The biggest problem with the work is that it underestimates the difficulty in modifying DRAM process for benefit in only a subset of applications which do bulk bitwise operations. In particular, I find it hard to believe that the commodity DRAM industry will incorporate this into their standard DRAM process. DRAM process is, at this point, a highly optimized, extremely tuned endeavor. Adding this kind of functionality will have a big impact on DRAM cost. The performance benefit on the subset of applications isn't enough to justify the higher costs this will incur and this will never get implemented.

Another Review

Another Review from ISCA 2016

Strengths

The proposed mechanisms effectively exploit the operation of the DRAM to perform efficient bitwise operations across entire rows of the DRAM.

Weaknesses

This requires a modification to the DRAM that will only help this type of bitwise operation. It seems unlikely that something like that will be adopted.

... This Will Never Get Implemented

Comments for the authors

This paper shows that DRAM could be modified to support bitwise operations directly within the DRAM itself. The performance advantages are compelling for situations in which bulk bitwise operations matter.

However, I am not really convinced that any DRAM manufacturer would really consider modifying the DRAM in this way. It benefits one specific type of operation, and while that is important for some applications, it is not really a general-purpose operation. It is not like the STL library would be changed to use this for its implementation of sets.

Yet Another Review

Yet Another Review from ISCA 2016

Weaknesses

The core novelty of Buddy RAM is almost all circuits-related (by exploiting sense amps). I do not find architectural innovation even though the circuits technique benefits architecturally by mitigating memory bandwidth and relieving cache resources within a subarray. The only related part is the new ISA support for bitwise operations at DRAM side and its induced issue on cache coherence.

This paper suits better to be peer-reviewed and published in a circuit conference or with a fabricated chip in ISSCC.

A Review from HPCA 2017: REJECT

#119 - HPCA23

Review #119A

Paper summary

Paper proposes DRAM technology changes (inverts, etc) to implement bit-wise operations directly on DRAM rows.

Overall merit

1. Reject

Reviewer expertise

2. I have passing familiarity with this area

Experimental methodology

2. Poor

Strengths

Seems like a new idea. Processor-in-Memory (PIM) ideas have resurged.

Weaknesses



* Impractical. Too many implications on ISA, DRAM design, and coherence protocols.

* Unlikely to benefit real-world computations.

* Evaluation did not consider full-program performance.

Comments for author

I am skeptical this would benefit real-world computations. I've never seen real-world program profiles with hot functions or instructions that are bit-wise operations.

On the other hand, I *have* seen system profiles that show non-trivial time zeroing pages. Suggest re-tooling your work to support page zeroing and evaluating that with a full-system simulation. Take a look at when/why the Linux kernel zeroes pages. You might be surprised at the possible impact.

A Review from ISCA 2017

#162 - ISCA 2017

Review #162A Updated 28 Jan 2017 5:16:50am EST

 [Plain text](#)

Post rebuttal overall merit

1. Reject

Overall merit

2. Weak reject

Novelty

2. Incremental improvement

Writing quality

4. Well-written

Reviewer expertise

5. This is my area

Paper summary

This paper proposes in-DRAM bit-wise operations by activating more than one word lines (and cells connected to the wordiness). Basically, it's a charge-based computation where the difference in charge stored cells connected to the same bit line is used for the logic operation.

Strengths

- conceptually a very interesting proposal (but practically not sure).
- consider various aspects including the interaction between

processors and RAM (although there isn't any new contribution and rather use the same proposal as prior work).

Weaknesses

- negative impact on the regularity of DRAM array design (and associated overhead evaluation seems to be very weak.
- significantly increase the testing cost

Comments to authors

This is an interesting proposal and well presented paper. However, I have some concerns regarding the evaluation (especially related to circuit level issues).

Especially, I feel that the variation related modeling and evaluation are weak as there are multiple sources of variations such as access transistors and sense-amp mismatches, minor defects in either access transistors and/or capacitor that can manifest in this particular proposed operation scenarios. That is, the authors oversimplify the variation modeling, which I believe failed to convince me this will work in practice. Also, the area overhead analysis sounds hand-waivy. I totally understand the difficulty of DRAM overhead analysis but also we must pursue more precise ways of evaluating the area impact as DRAM is very cost-sensitive.

Another Review from ISCA 2017

Review #162B Updated 1 Feb 2017 6:50:31pm EST

 [Plain text](#)

Post rebuttal overall merit

2. Weak reject

Overall merit

2. Weak reject

Novelty

3. New contribution

Writing quality

3. Adequate

Reviewer expertise

3. I know the material, but am not an expert

Paper summary

This paper proposes performing bulk bit-wise operations at DRAM. They leverage analog operation of DRAM, and add some extra

#162 - ISCA 2017

circuits to do bit-wise operations at row granularity.

Strengths

The idea of handling bit-wise operations in memory is interesting.

Weaknesses

Not motivated well.

Not convinced the possible gains worth all the complexity.

Not convinced if the proposal is applicable in real world applications that do bit-wise operations on different data granularity.

Comments to authors

* The paper lacks motivation. The authors talk about how common bit-wise operations are. However, they do not provide any stat on how often these operations are being used, and more importantly, on what data granularity.

* Although bit-wise operations are common in some applications, they are not necessarily done at large granularity. For example, many applications do bit-wise operations at small 64-byte (or even smaller) entities. For such cases, this paper requires copying two whole rows to some temporary rows, and doing the operation on those rows. Please explain how you handle such cases, and what the benefits would be.

* What happens if the user does bit-wise operation on two 8-byte data, and want to store it in a third block?

* What happens if both operands are located in one row?

* The main issue with this work is that it requires flushing blocks out of caches to do the bit-wise operations. Imagine you have blocks A and B in the cache, as discussed in section 6.2.3., the proposal would flush them out of caches (not sure how?), writes

ISCA 2017 Summary

@A1 6 Mar 2017

This paper was discussed both online and at the PC meeting. Reviewers were uniformly positive about the novelty of the proposed Buddy-RAM design. However, reviewers were also concerned about the feasibility of the design. During the post-rebuttal and PC discussion, the main concerns raised were (1) the impact of process variation on the design's functional correctness;

#162 - ISCA 2017

(2) the potential reliability issues that arise due to the lack of ECC/CRC mechanisms; and (3) the impact on DRAM testing cost.

Specifically on point (1), some reviewers raised concerns about the limitations of the simulations performed to address variability: "Monte-Carlo cannot capture tail distribution of cell failures. Also Monte-Carlo cannot capture random correlated WID process variation issues (only some random uncorrelated variations)."

Given these concerns, the PC ultimately decided to reject the paper. We hope that this feedback is useful in preparing a future version of the paper.

The Reviewer Accountability Problem

Acknowledgments

We thank the reviewers of ISCA 2016/2017, MICRO 2016/2017, and HPCA 2017 for their valuable comments. We

MICRO 2017: Accepted

Accepted



Submission (1837kB)

4 Apr 2017 11:33:57pm EDT ·

7420f9f02c549bccca0dc6216a5e9887dffe0d422



Revision (1852kB)

14 Jun 2017 4:16am EDT ·

22f0d123a22cf04960928e0ac43d972b5a33a848

▼ Abstract

Many important applications trigger bitwise operations on large bit vectors (bulk bitwise operations). In fact, recent

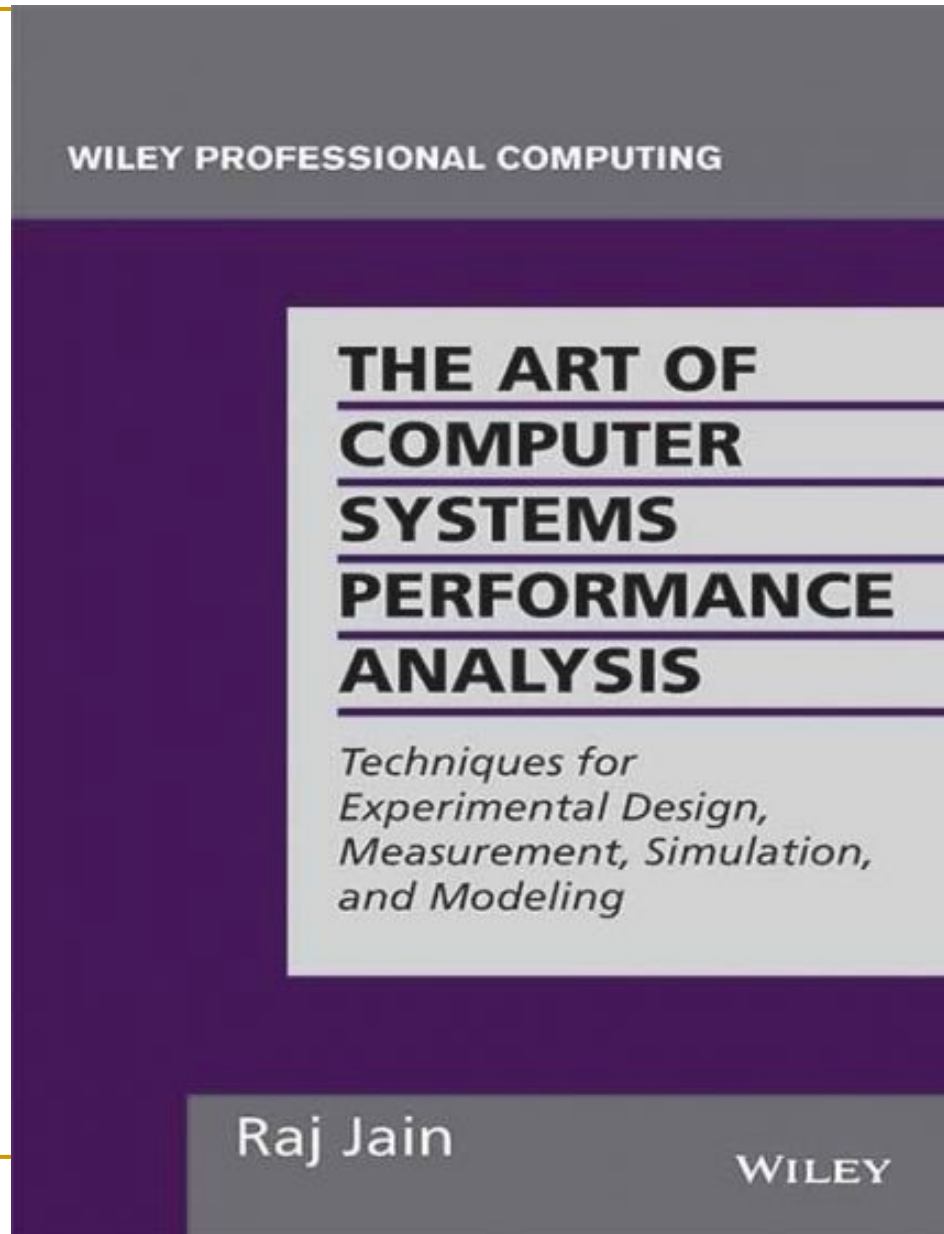
► Authors

V. Seshadri, D. Lee, T. Mullins, H. Hassan, A. Boroumand, J. Kim, M. Kozuch, O. Mutlu, P. Gibbons, T. Mowry [\[details\]](#)

	PosResOve	OveMer	RevExp	Nov	PotImp	WriQua	ImpRev
Review #347A	4	3	5	4	3	4	2
Review #347B	3	4	5	4	3	4	3
Review #347C		2	5	3	2	4	4
Review #347D	3	4	4	4	4	4	2
Review #347E	3	3	4	3	3	3	4
Review #347F	3	2	4	3	3	4	1

1 Comment: [Rebuttal Response \(V. Seshadri\)](#)

Aside: A Recommended Book



Raj Jain, "[The Art of Computer Systems Performance Analysis](#)," Wiley, 1991.

10.8 DECISION MAKER'S GAMES

Even if the performance analysis is correctly done and presented, it may not be enough to persuade your audience—the decision makers—to follow your recommendations. The list shown in Box 10.2 is a compilation of reasons for rejection heard at various performance analysis presentations. You can use the list by presenting it immediately and pointing out that the reason for rejection is not new and that the analysis deserves more consideration. Also, the list is helpful in getting the competing proposals rejected!

There is no clear end of an analysis. Any analysis can be rejected simply on the grounds that the problem needs more analysis. This is the first reason listed in Box 10.2. The second most common reason for rejection of an analysis and for endless debate is the workload. Since workloads are always based on the past measurements, their applicability to the current or future environment can always be questioned. Actually workload is one of the four areas of discussion that lead a performance presentation into an endless debate. These “rat holes” and their relative sizes in terms of time consumed are shown in Figure 10.26. Presenting this cartoon at the beginning of a presentation helps to avoid these areas.

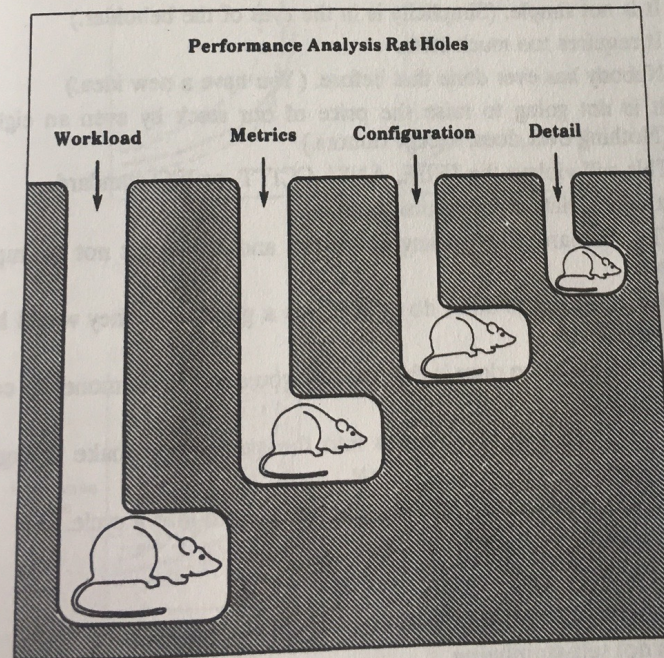


FIGURE 10.26 Four issues in performance presentations that commonly lead to endless discussion.

Raj Jain, “The Art of Computer Systems Performance Analysis,” Wiley, 1991.

Box 10.2 Reasons for Not Accepting the Results of an Analysis

1. This needs more analysis.
2. You need a better understanding of the workload.
3. It improves performance only for long I/O's, packets, jobs, and files, and most of the I/O's, packets, jobs, and files are short.
4. It improves performance only for short I/O's, packets, jobs, and files, but who cares for the performance of short I/O's, packets, jobs, and files; its the long ones that impact the system.
5. It needs too much memory/CPU/bandwidth and memory/CPU/bandwidth isn't free.
6. It only saves us memory/CPU/bandwidth and memory/CPU/bandwidth is cheap.
7. There is no point in making the networks (similarly, CPUs/disks/...) faster; our CPUs/disks (any component other than the one being discussed) aren't fast enough to use them.
8. It improves the performance by a factor of x , but it doesn't really matter at the user level because everything else is so slow.
9. It is going to increase the complexity and cost.
10. Let us keep it simple stupid (and your idea is not stupid).
11. It is not simple. (Simplicity is in the eyes of the beholder.)
12. It requires too much state.
13. Nobody has ever done that before. (You have a new idea.)
14. It is not going to raise the price of our stock by even an eighth. (Nothing ever does, except rumors.)
15. This will violate the IEEE, ANSI, CCITT, or ISO standard.
16. It may violate some future standard.
17. The standard says nothing about this and so it must not be important.
18. Our competitors don't do it. If it was a good idea, they would have done it.
19. Our competition does it this way and you don't make money by copying others.
20. It will introduce randomness into the system and make debugging difficult.
21. It is too deterministic; it may lead the system into a cycle.
22. It's not interoperable.
23. This impacts hardware.
24. That's beyond today's technology.
25. It is not self-stabilizing.
26. Why change—it's working OK.

Raj Jain, "The Art of Computer Systems Performance Analysis," Wiley, 1991.

Suggestions to Reviewers

- Be fair; you do not know it all
- Be open-minded; you do not know it all
- Be accepting of diverse research methods: there is no single way of doing research or writing papers
- Be constructive, not destructive
- Enable heterogeneity, but do **not** have double standards...

Do not block or delay scientific progress for non-reasons

We Need to Fix the Reviewer Accountability Problem

Main Memory Needs Intelligent Controllers

Research Community
Needs

Accountable Reviewers

An Interview on Research and Education

- Computing Research and Education (@ ISCA 2019)
 - https://www.youtube.com/watch?v=8ffSEKZhmvo&list=PL5Q2soXY2Zi_4oP9LdL3cc8G6NIjD2Ydz

- Maurice Wilkes Award Speech (10 minutes)
 - https://www.youtube.com/watch?v=tcQ3zZ3JpuA&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJI&index=15

More Thoughts and Suggestions

- Onur Mutlu,
"Some Reflections (on DRAM)"
*Award Speech for ACM SIGARCH Maurice Wilkes Award, at the **ISCA** Awards Ceremony, Phoenix, AZ, USA, 25 June 2019.*
[Slides (pptx) (pdf)]
[Video of Award Acceptance Speech (Youtube; 10 minutes) (Youku; 13 minutes)]
[Video of Interview after Award Acceptance (Youtube; 1 hour 6 minutes) (Youku; 1 hour 6 minutes)]
[News Article on "ACM SIGARCH Maurice Wilkes Award goes to Prof. Onur Mutlu"]

- Onur Mutlu,
"How to Build an Impactful Research Group"
*57th Design Automation Conference Early Career Workshop (**DAC**), Virtual, 19 July 2020.*
[Slides (pptx) (pdf)]

RowClone & Bitwise Ops in Real DRAM Chips

ComputeDRAM: In-Memory Compute Using Off-the-Shelf DRAMs

Fei Gao

feig@princeton.edu

Department of Electrical Engineering
Princeton University

Georgios Tziantzioulis

georgios.tziantzioulis@princeton.edu

Department of Electrical Engineering
Princeton University

David Wentzlaff

wentzlaf@princeton.edu

Department of Electrical Engineering
Princeton University

RowClone & Bitwise Ops in Real DRAM Chips

MICRO-52, October 12–16, 2019, Columbus, OH, USA

Gao et al.

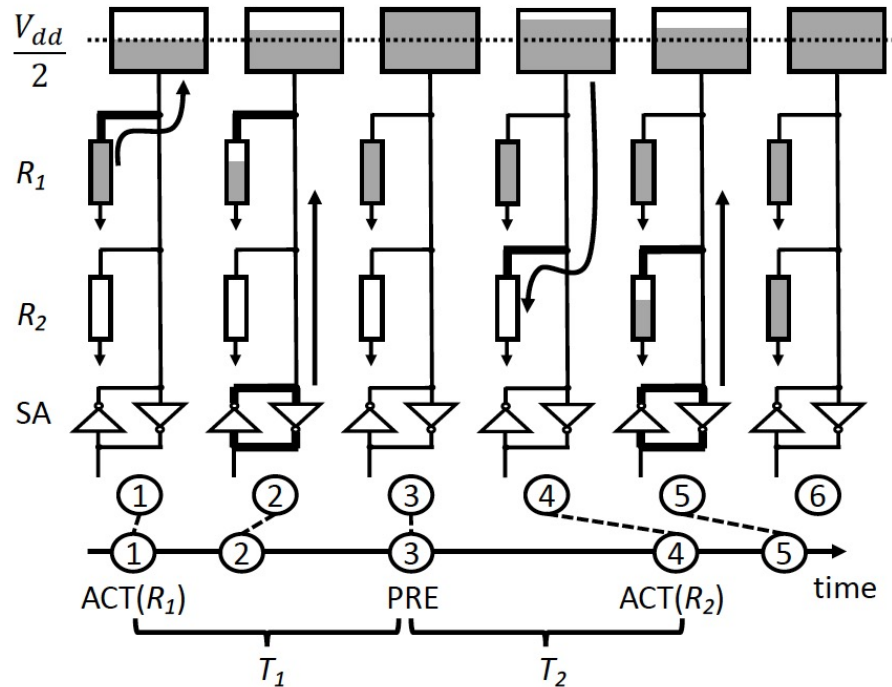


Figure 4: Timeline for a single bit of a column in a row copy operation. The data in R_1 is loaded to the bit-line, and overwrites R_2 .

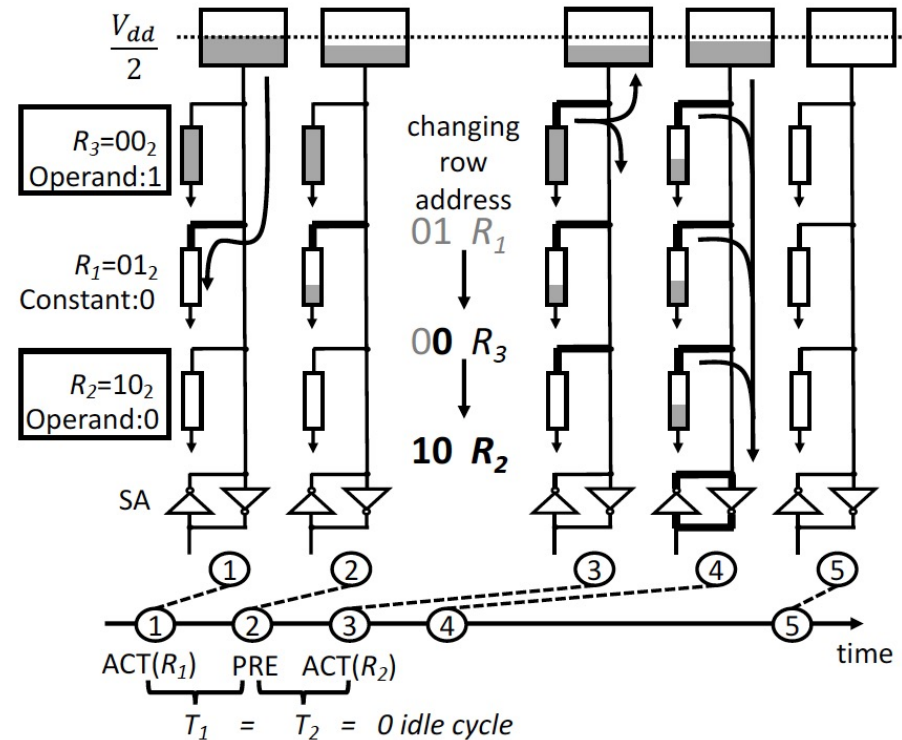
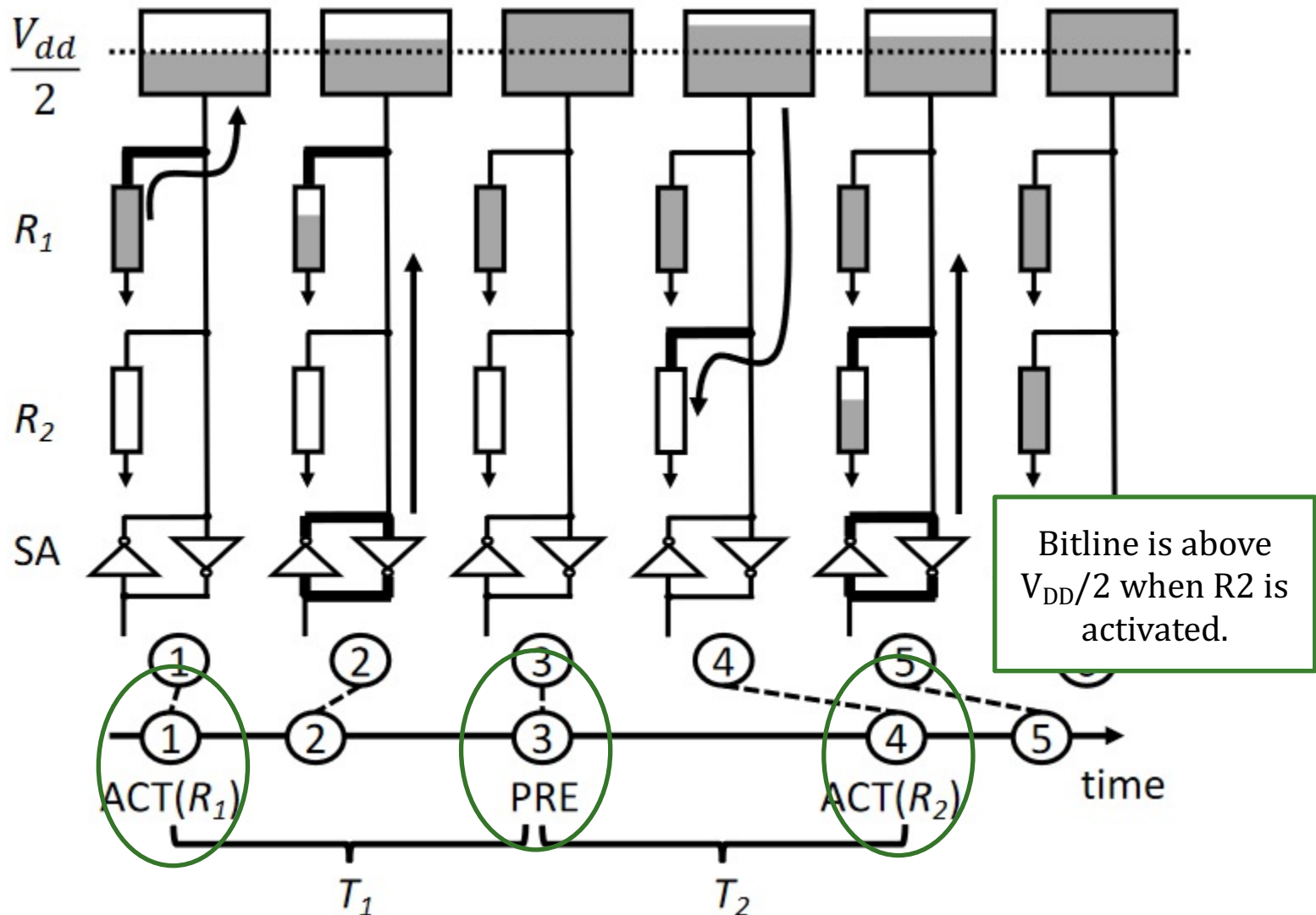
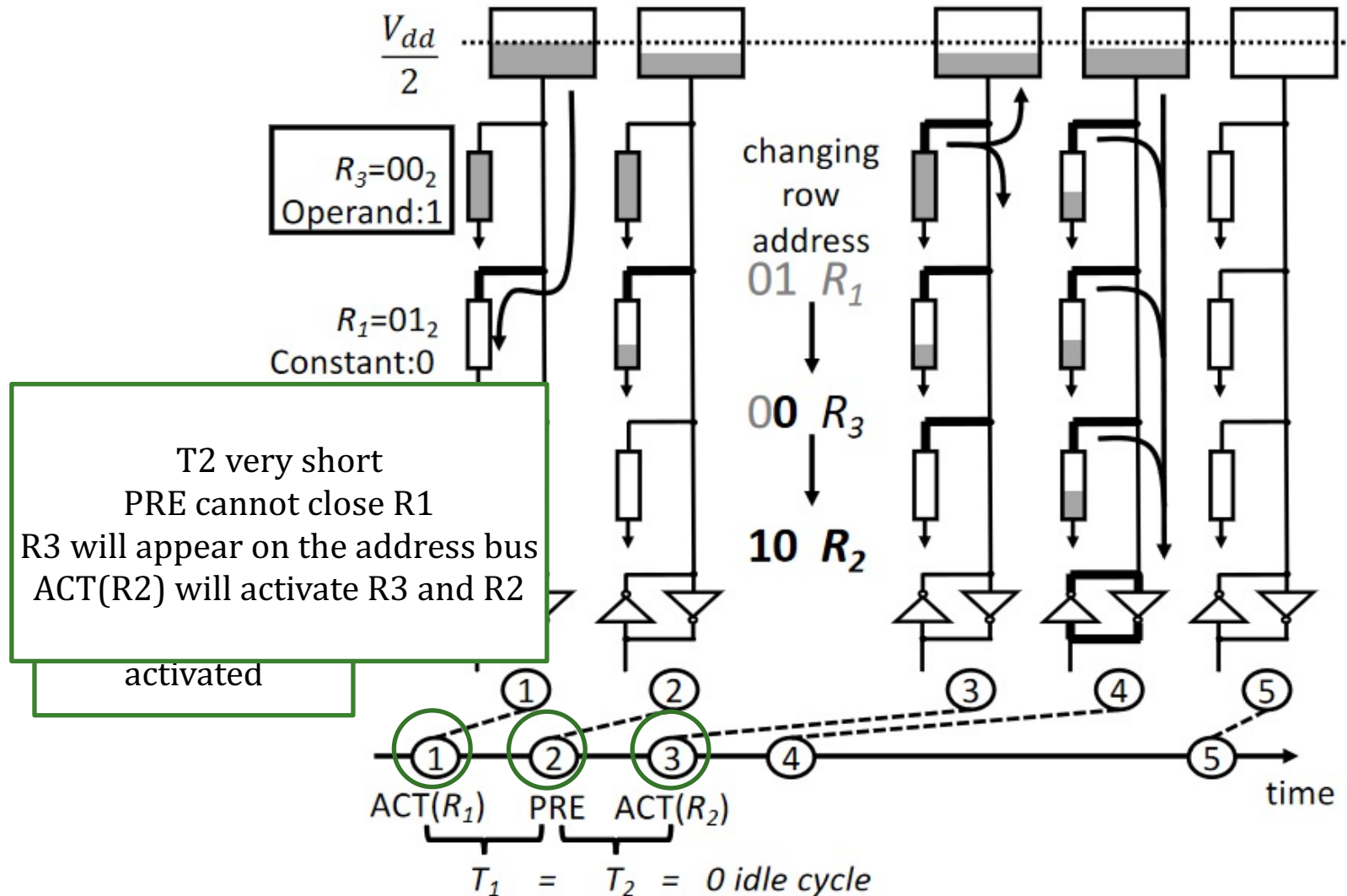


Figure 5: Logical AND in ComputeDRAM. R_1 is loaded with constant zero, and R_2 and R_3 store operands (0 and 1). The result ($0 = 1 \wedge 0$) is finally set in all three rows.

Row Copy in ComputeDRAM



Bitwise AND in ComputeDRAM



Experimental Methodology

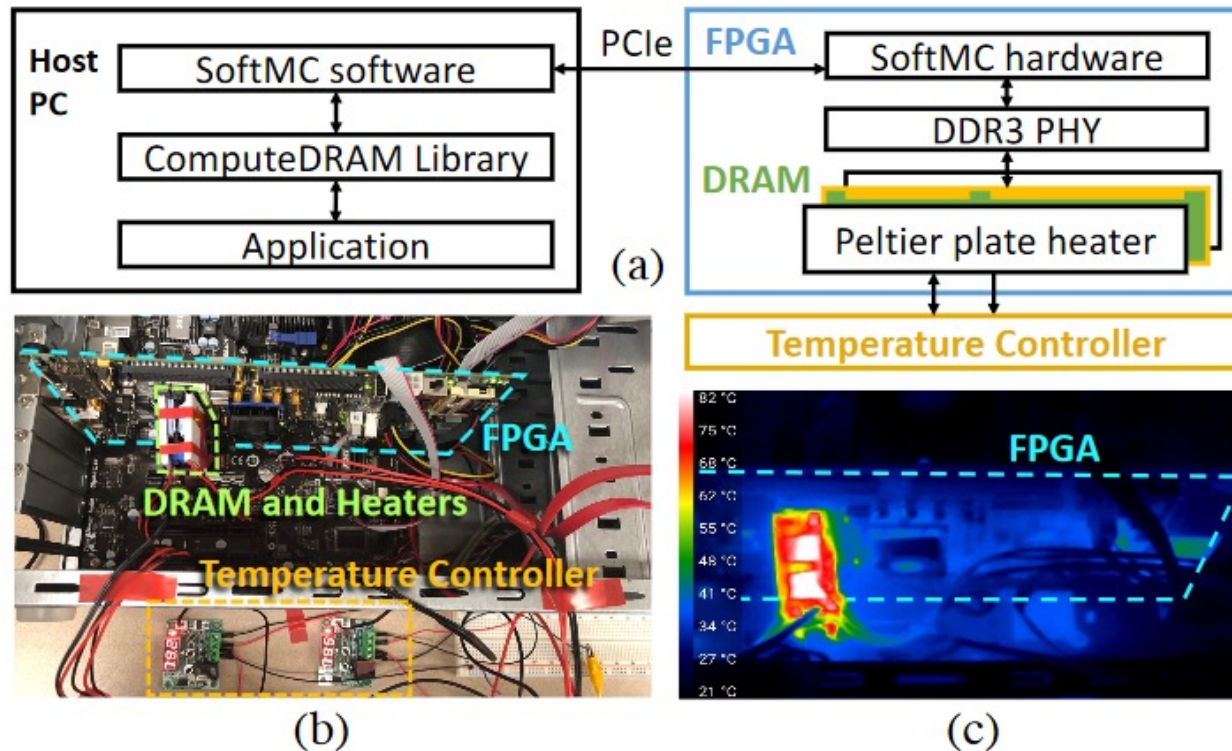


Figure 9: (a) Schematic diagram of our testing framework. (b) Picture of our testbed. (c) Thermal picture when the DRAM is heated to 80 °C.

Experimental Methodology

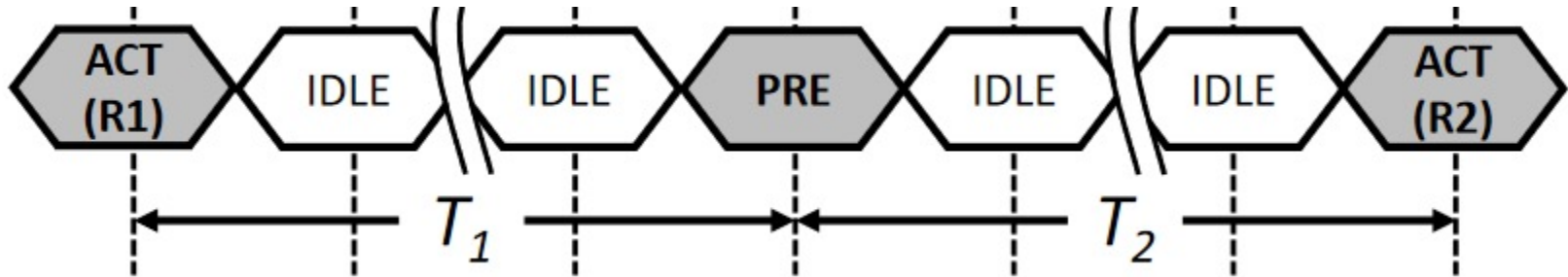
Table 1: Evaluated DRAM modules

Group ID: Vendor_Size_Freq(MHz)	Part Num	# Modules
SKhynix_2G_1333	HMT325S6BFR8C-H9	6
SKhynix_4G_1333		2
SKhynix_4G_1333		2
SKhynix_4G_1333		4
SKhynix_4G_1333		2
Samsung_4G_1333		2
Samsung_4G_1333		2
Micron_2G_1333		2
Micron_2G_1333		2
Elpida_2G_1333	EBJ21UE8BDS0-DJ-F	2
Nanya_4G_1333	NT4GC64B8HG0NS-CG	2
TimeTec_4G_1333	78AP10NUS2R2-4G	2
Corsair_4G_1333	CMSA8GX3M2A1333C9	2

**32 DDR3 Modules
~256 DRAM Chips**

Proof of Concept

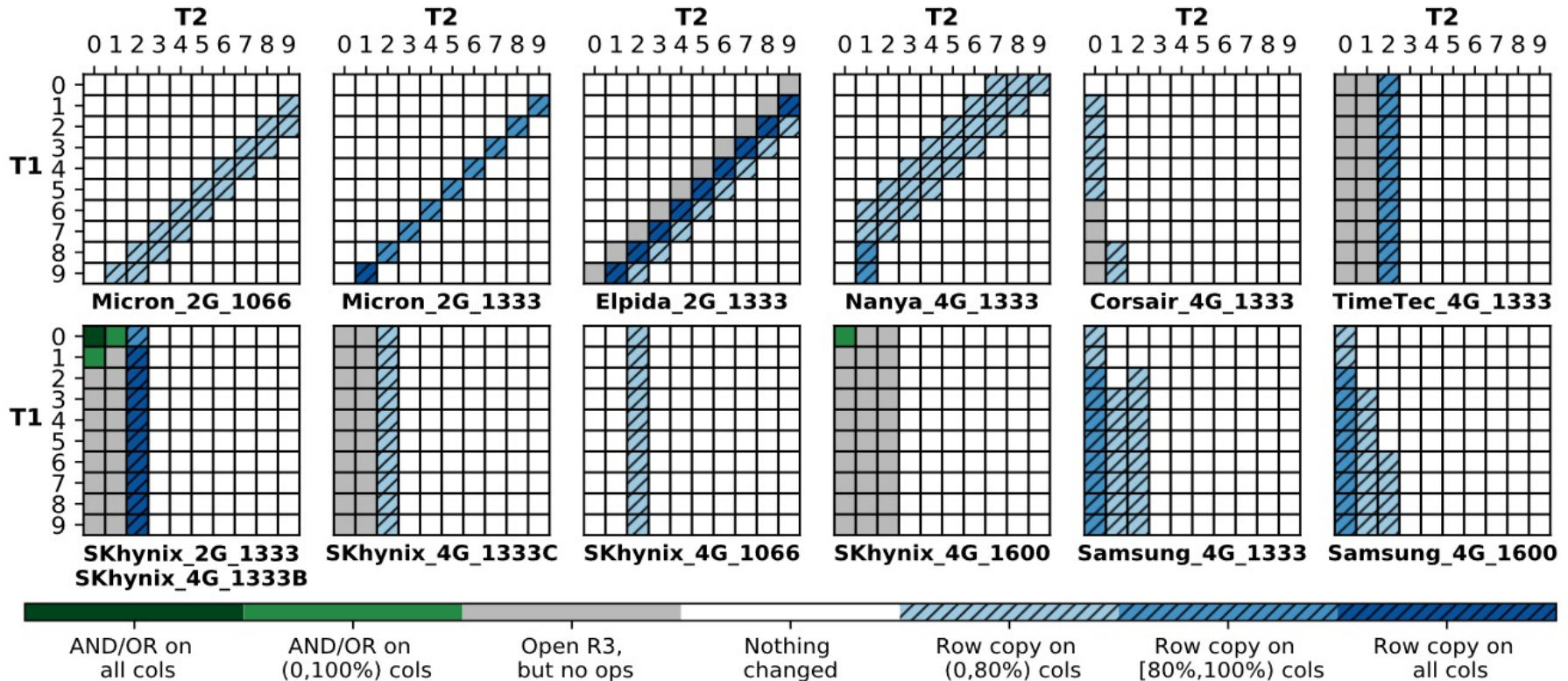
- How they test these memory modules:
 - Vary T_1 and T_2 , observe what happens.



SoftMC Experiment

1. Select a random subarray
2. Fill subarray with random data
3. Issue ACT-PRE-ACTs with given T_1 & T_2
4. Read out subarray
5. Find out how many columns in a row support either operation
 - Row-wise success ratio

Proof of Concept



- Each grid represents the success ratio of operations for a specific DDR3 module.

Pinatubo: RowClone and Bitwise Ops in PCM

Pinatubo: A Processing-in-Memory Architecture for Bulk Bitwise Operations in Emerging Non-volatile Memories

Shuangchen Li^{1*}, Cong Xu², Qiaosha Zou^{1,5}, Jishen Zhao³, Yu Lu⁴, and Yuan Xie¹

University of California, Santa Barbara¹, Hewlett Packard Labs²

University of California, Santa Cruz³, Qualcomm Inc.⁴, Huawei Technologies Inc.⁵
{shuangchenli, yuanxie}@ece.ucsb.edu¹

Pinatubo: RowClone and Bitwise Ops in PCM

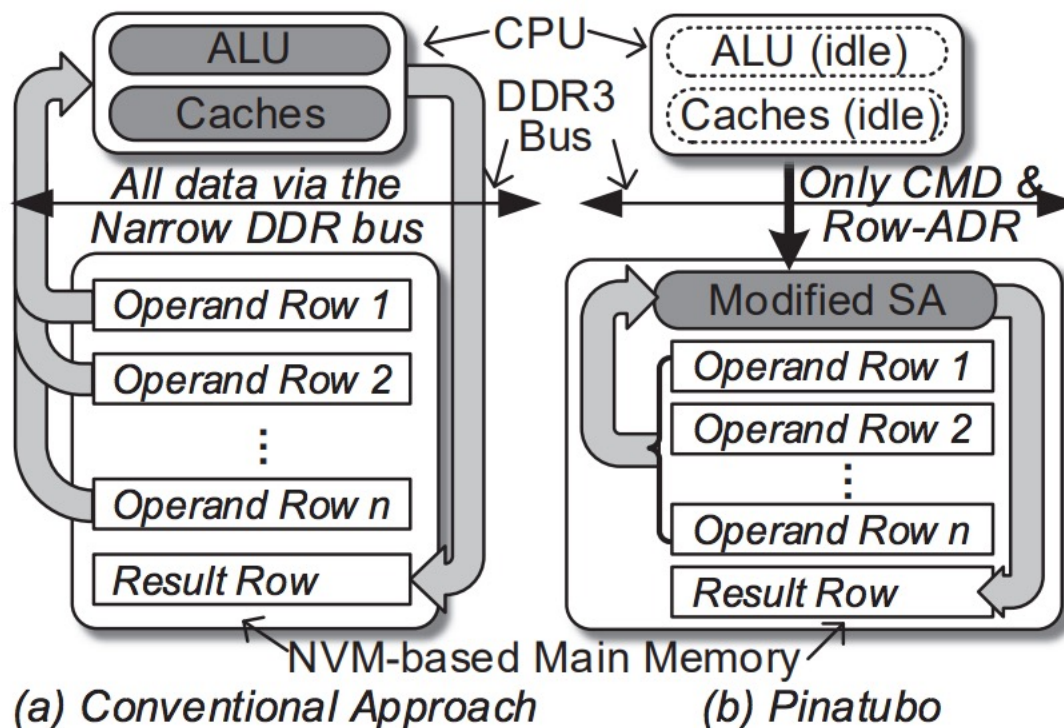


Figure 2: Overview: (a) Computing-centric approach, moving tons of data to CPU and write back. (b) The proposed Pinatubo architecture, performs n -row bitwise operations inside NVM in one step.

Mindset Issues Are Everywhere

- “Why Change? It’s Working OK!” mindset limits progress
- There are many such examples in real life
- Examples of Bandwidth Waste in Real Life
- Examples of Latency and Queueing Delays in Real Life
- Example of Where to Build a Bridge over a River

Suggestion to Researchers: Principle: Passion

Follow Your Passion
**(Do not get derailed
by naysayers)**

Suggestion to Researchers: Principle: Resilience

Be Resilient

Principle: Learning and Scholarship

Focus on
learning and scholarship

Principle: Learning and Scholarship

The quality of your work
defines your impact

Principle: Work Hard

Work Hard to
Enable Your Passion

Principle: Good Mindset, Goals & Focus

You can make a
good impact
on the world

Recommended Interview on Research & Education

- **Computing Research and Education (@ ISCA 2019)**
 - https://www.youtube.com/watch?v=8ffSEKZhmvo&list=PL5Q2soXY2Zi_4oP9LdL3cc8G6NIjD2Ydz

- **Maurice Wilkes Award Speech (10 minutes)**
 - https://www.youtube.com/watch?v=tcQ3zZ3JpuA&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJI&index=15

- Onur Mutlu,
"Some Reflections (on DRAM)"
*Award Speech for ACM SIGARCH Maurice Wilkes Award, at the **ISCA** Awards Ceremony, Phoenix, AZ, USA, 25 June 2019.*
[[Slides \(pptx\)](#)] [[pdf](#)]
[[Video of Award Acceptance Speech \(Youtube; 10 minutes\)](#)] [[Youku; 13 minutes](#)]
[[Video of Interview after Award Acceptance \(Youtube; 1 hour 6 minutes\)](#)] [[Youku; 1 hour 6 minutes](#)]
[[News Article on "ACM SIGARCH Maurice Wilkes Award goes to Prof. Onur Mutlu"](#)]

Recommended Interview



Interview with Onur Mutlu @ ISCA 2019 on computing research & education (after Maurice Wilkes Award)

6,749 views • Oct 19, 2019

👍 195 🗨️ 0 ➦ SHARE ⚙️ ⌵ ⌵ ⌵



Onur Mutlu Lectures
19.1K subscribers

ANALYTICS

EDIT VIDEO

A Talk on Impactful Research & Education



The video player shows a presentation slide with the following content:

Applying to Grad School
& Doing Impactful Research

Onur Mutlu
omutlu@gmail.com
<https://people.inf.ethz.ch/omutlu>
13 June 2020
Undergraduate Architecture Mentoring Workshop @ ISCA 2021

Logos for SAFARI, ETH zürich, and Carnegie Mellon are displayed at the bottom of the slide.

Below the video player, the video title is "Arch. Mentoring Workshop @ISCA'21 - Applying to Grad School & Doing Impactful Research - Onur Mutlu". It has 1,563 views and premiered on Jun 16, 2021. The video is by "Onur Mutlu Lectures" (17.2K subscribers). The description mentions a panel talk at the Undergraduate Architecture Mentoring Workshop at ISCA 2021, with a link to <https://sites.google.com/wisc.edu/uar...>

Video controls at the bottom of the player show a progress bar at 0:27 / 50:31, and buttons for CC, settings, full screen, and other controls.

Richard Hamming

“You and Your Research”

Transcription of the
Bell Communications Research Colloquium Seminar
7 March 1986

<https://safari.ethz.ch/architecture/fall2021/lib/exe/fetch.php?media=youandyourresearch.pdf>

Suggested Reading on Mindset & More

If you really want to be a first-class scientist you need to know yourself, your weaknesses, your strengths, and your bad faults, like my egotism. How can you convert a fault to an asset? How can you convert a situation where you haven't got enough manpower to move into a direction when that's exactly what you need to do? I say again that I have seen, as I studied the history, the successful scientist changed the viewpoint and what was a defect became an asset.

In summary, I claim that some of the reasons why so many people who have greatness within their grasp don't succeed are: they don't work on important problems, they don't become emotionally involved, they don't try and change what is difficult to some other situation which is easily done but is still important, and they keep giving themselves alibis why they don't. They keep saying that it is a matter of luck. I've told you how easy it is; furthermore I've told you how to reform. Therefore, go forth and become great scientists!



Seminar in Computer Architecture

Meeting 3: RowClone (Processing using DRAM)

Prof. Onur Mutlu

ETH Zürich

Fall 2021

7 October 2021