Seminar in
Computer Architecture
Meeting 2a: Example Review I

Minesh Patel

ETH Zürich

Spring 2020

27 February 2020

Suggested Paper Discussion Format

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

~25 minute Summary

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

~10 min Critique
+
~10 min Discussion

Course Logistics

- Requirements
  - 1: Submit HW0 (due tomorrow at 11:59pm)
  - 2: Choose paper preferences (due 2 March at 11:59pm)

- 2 presentations in each of 11 sessions
  - Max 50 minutes each, including questions and discussions
  - 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 20 minute critique+discussion+brainstorming+feedback
  - Should follow the suggested guidelines

Algorithm for Presentation Preparation

- Study Lecture 1 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 today

Structure of the Presentation

- Background, Problem & Goal
- Novelty
- Key Approach and Ideas
- Mechanisms (in some detail)
- Key Results: Methodology and Evaluation
- Summary

~25 minute Summary

- Strengths
- Weaknesses
- Thoughts and Ideas
- Takeaways
- Open Discussion

**~10 min Critique
+
~10 min 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

Originally presented at MICRO 2013
Today's presenter: Minesh Patel
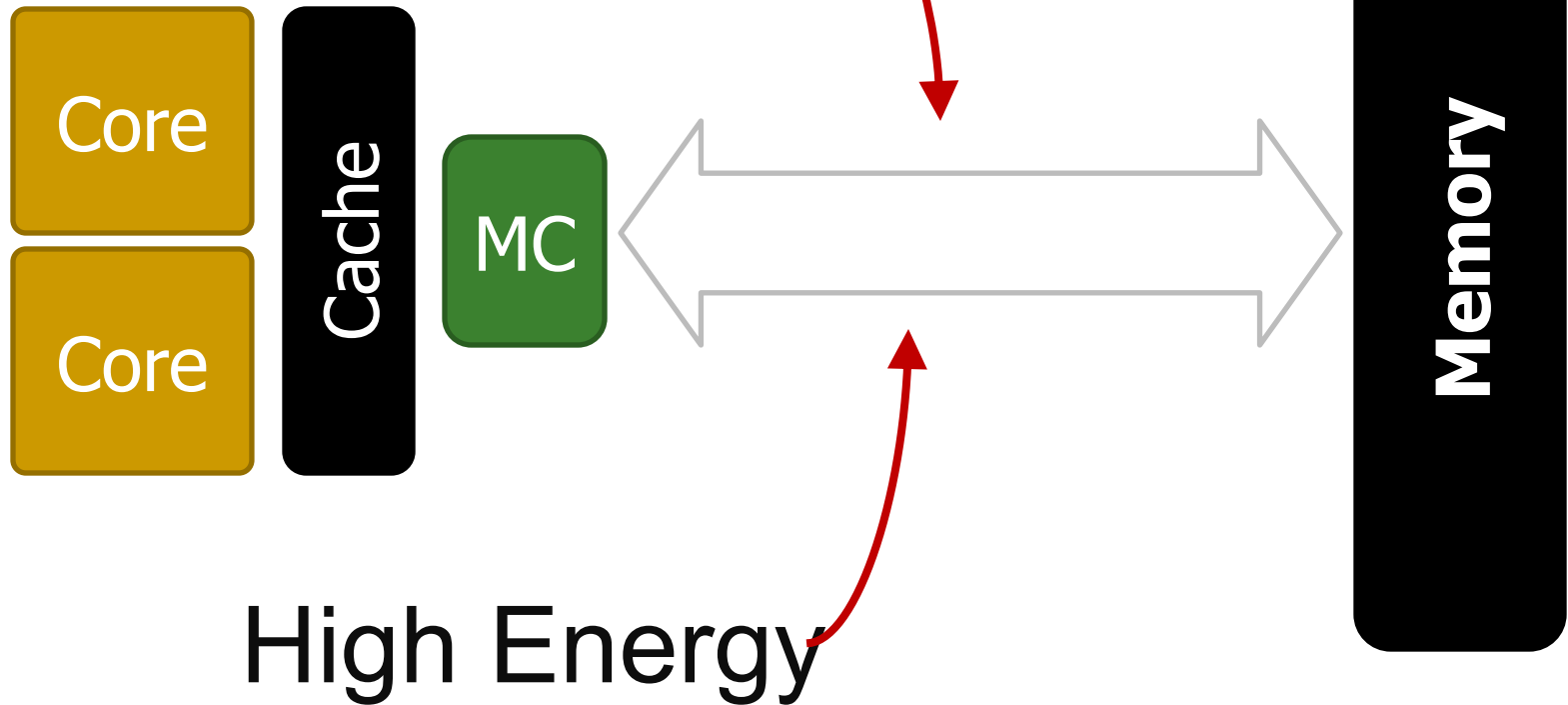
SAFARI   **Carnegie Mellon**   (intel)

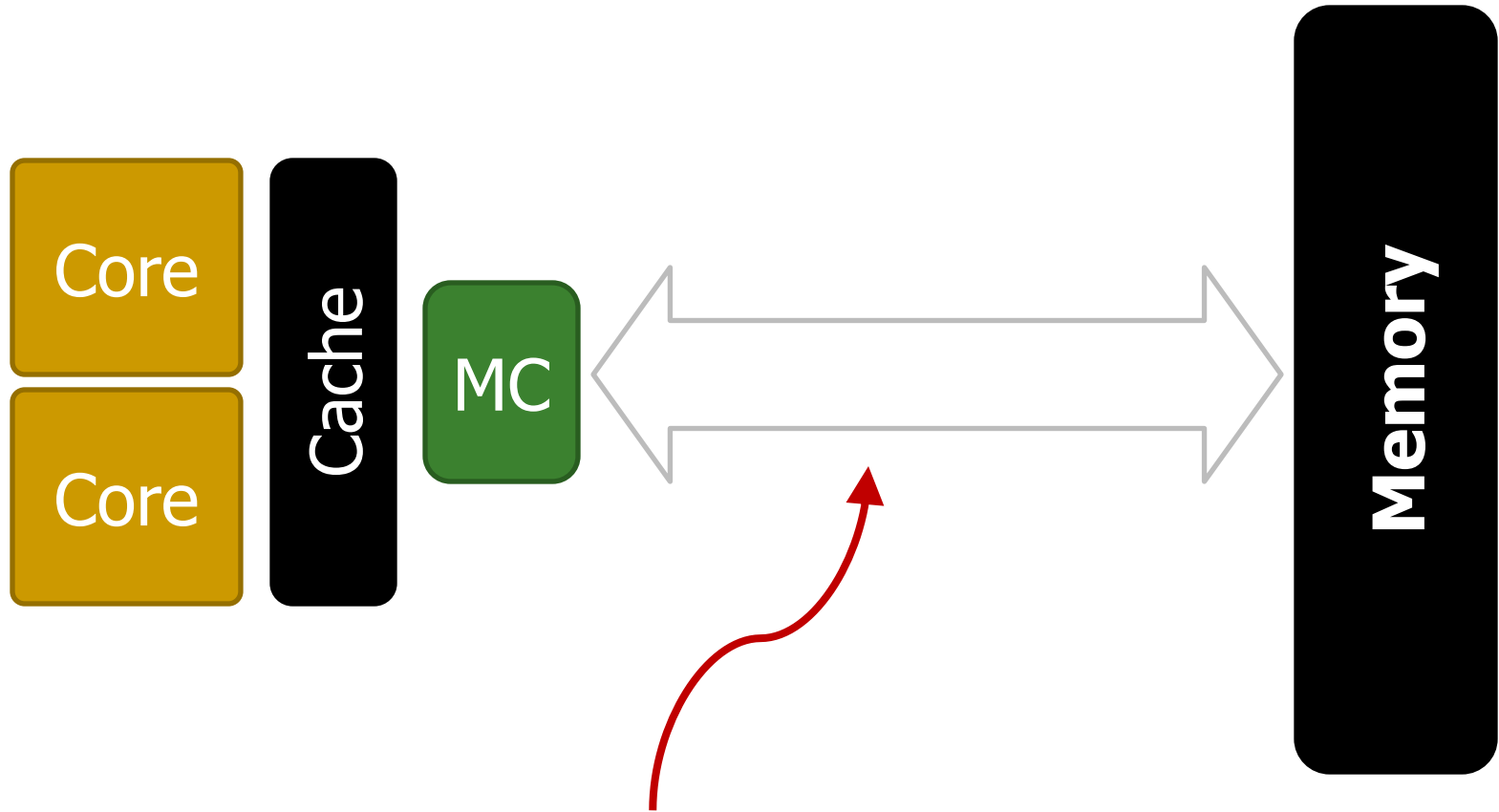# Background, Problem & Goal

Memory Channel – Bottleneck

Goal: Reduce Memory Bandwidth Demand

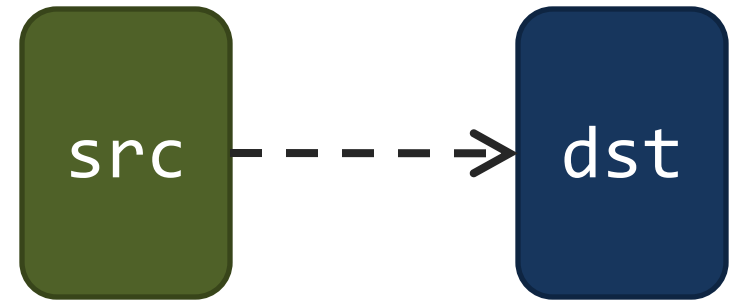**Reduce unnecessary data movement**

Bulk Data Copy and Initialization
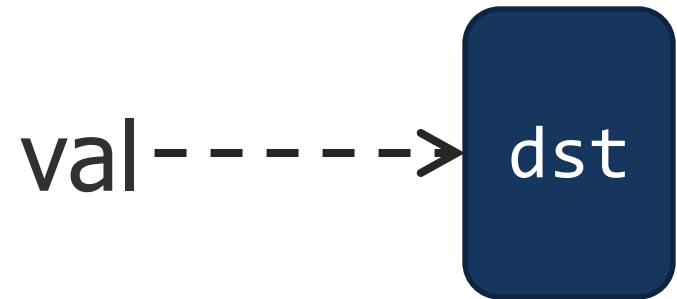
Bulk Data Copy

src - - - - -> dst

Bulk Data Initialization

val - - - - -> dst

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
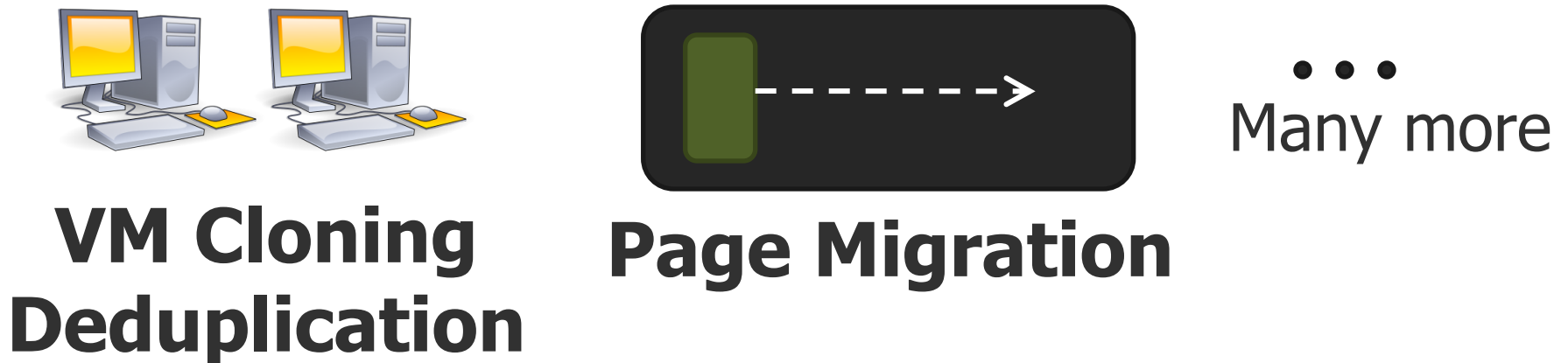Email: {zhao, bhuyan}@cs.ucr.edu
[‡]Communications Technology Lab, Intel C

Architecture Support for Improving Bulk Memory Copying and Initialization Performance

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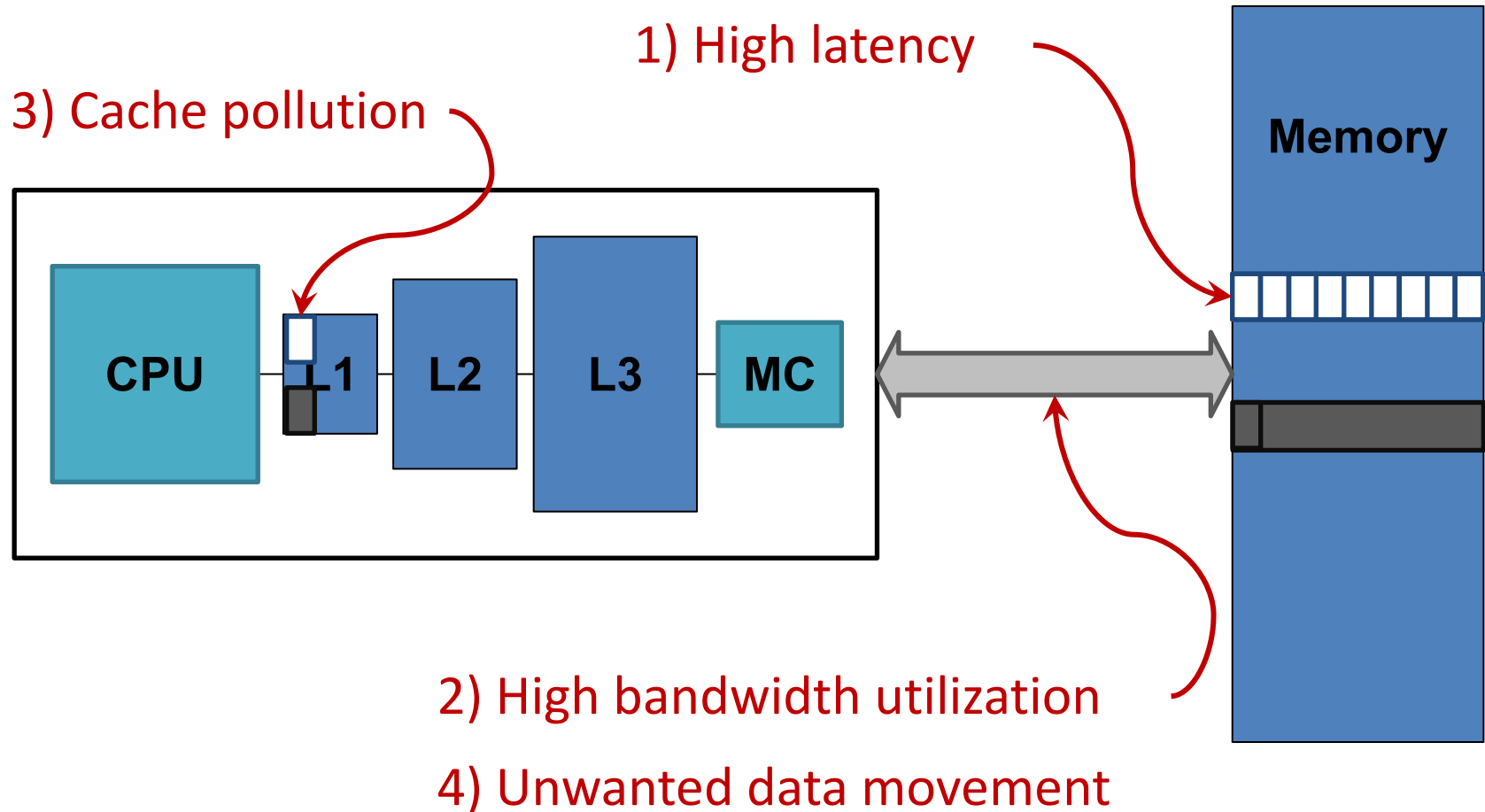North Carolina State University
Raleigh, USA

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

SAFARI

Shortcomings of Today's Systems

1) High latency

3) Cache pollution

Memory

CPU **L1** **L2** **L3** **MC**

2) High bandwidth utilization

4) Unwanted data movement

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

Novelty, Key Approach, and Ideas

RowClone: In-Memory Copy

3) No cache pollution

1) Low latency

Memory

CPU — **L1** **L2** **L3** **MC**

2) Low bandwidth utilization

4) No unwanted data movement
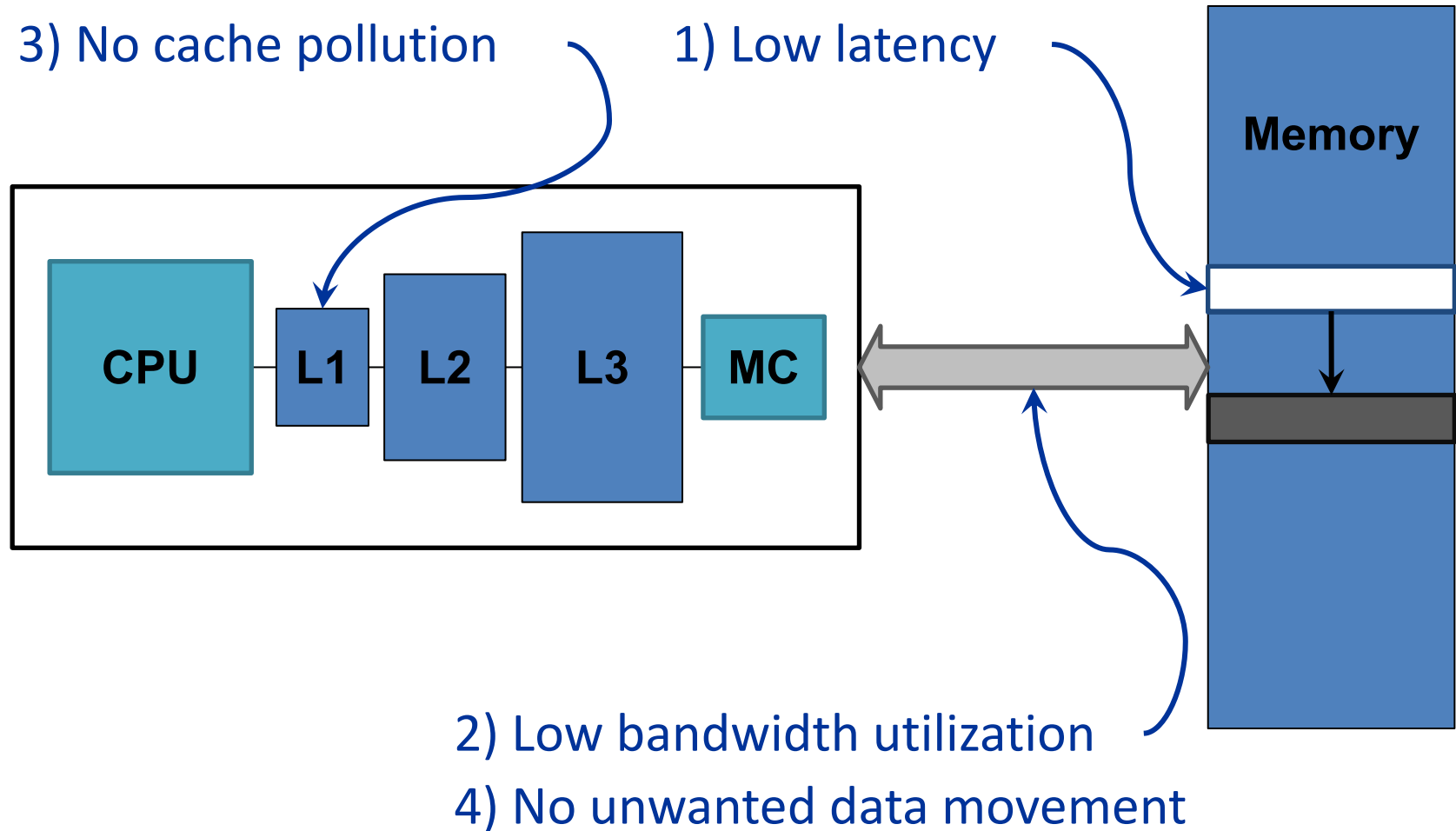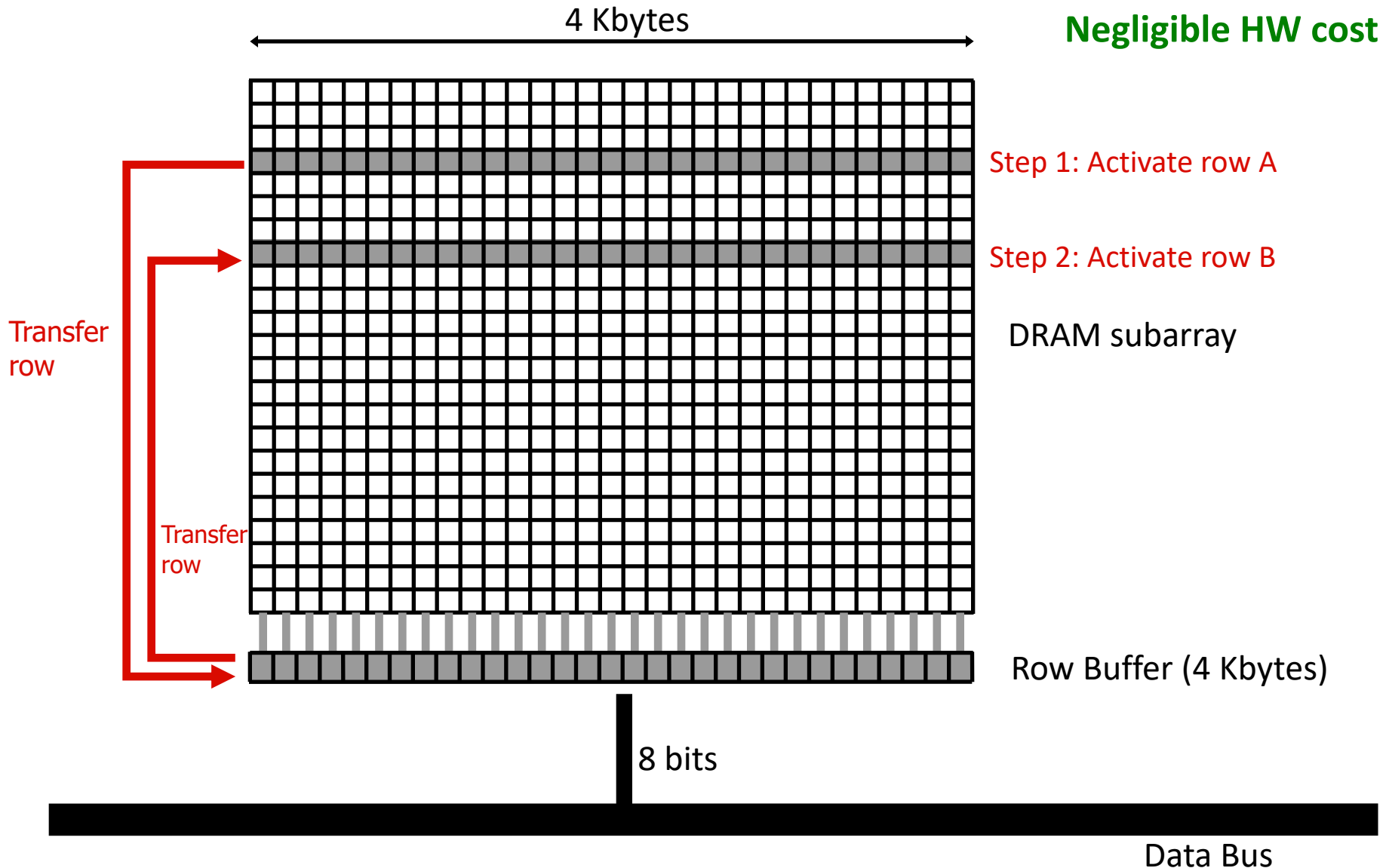
1046ns, 3.6uJ → 90ns, 0.04uJ

RowClone: In-DRAM Row Copy

Idea: Two consecutive ACTivates
Negligible HW cost

4 Kbytes

Step 1: Activate row A

Step 2: Activate row B

DRAM subarray

Transfer row

Transfer row

Row Buffer (4 Kbytes)

8 bits

Data Bus

Mechanisms (in some detail)

DRAM Chip Organization

Memory Channel

Chip I/O

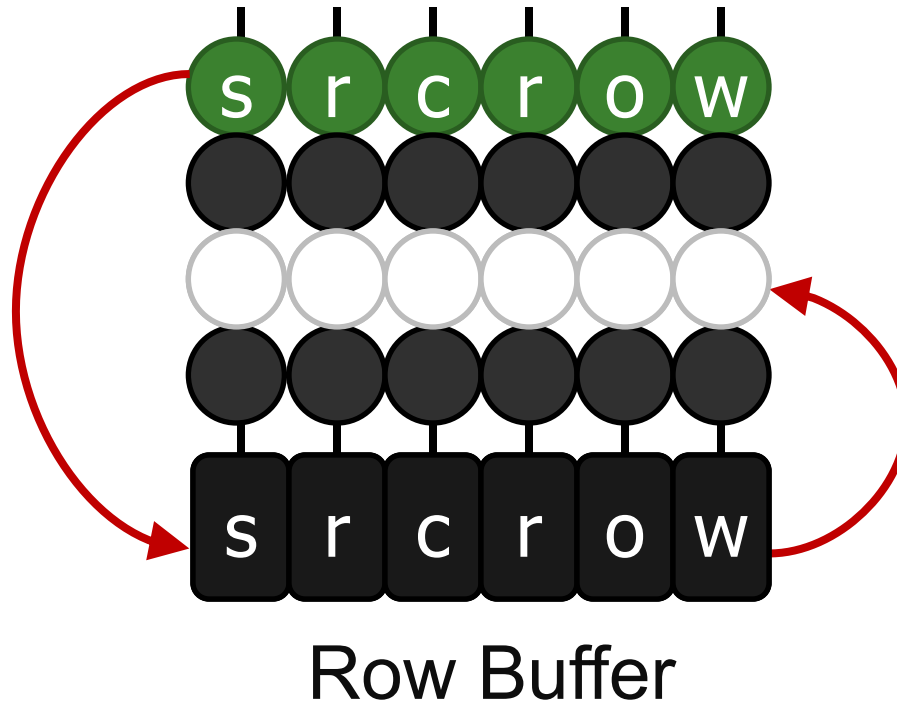Bank

Subarray

Bank I/O

Row of DRAM Cells
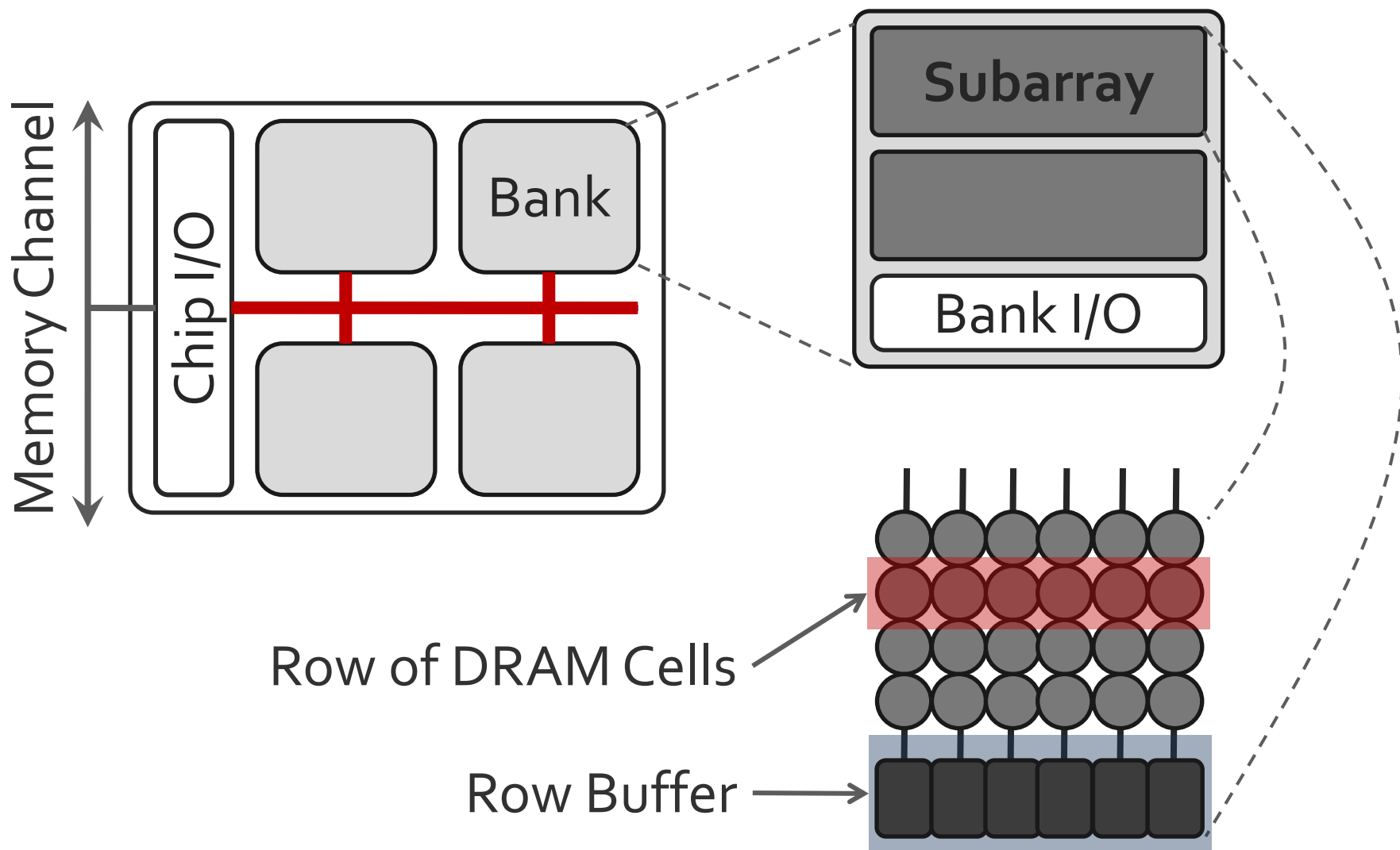
Row Buffer

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)

Row Buffer

✓ **1. Source row to row buffer**
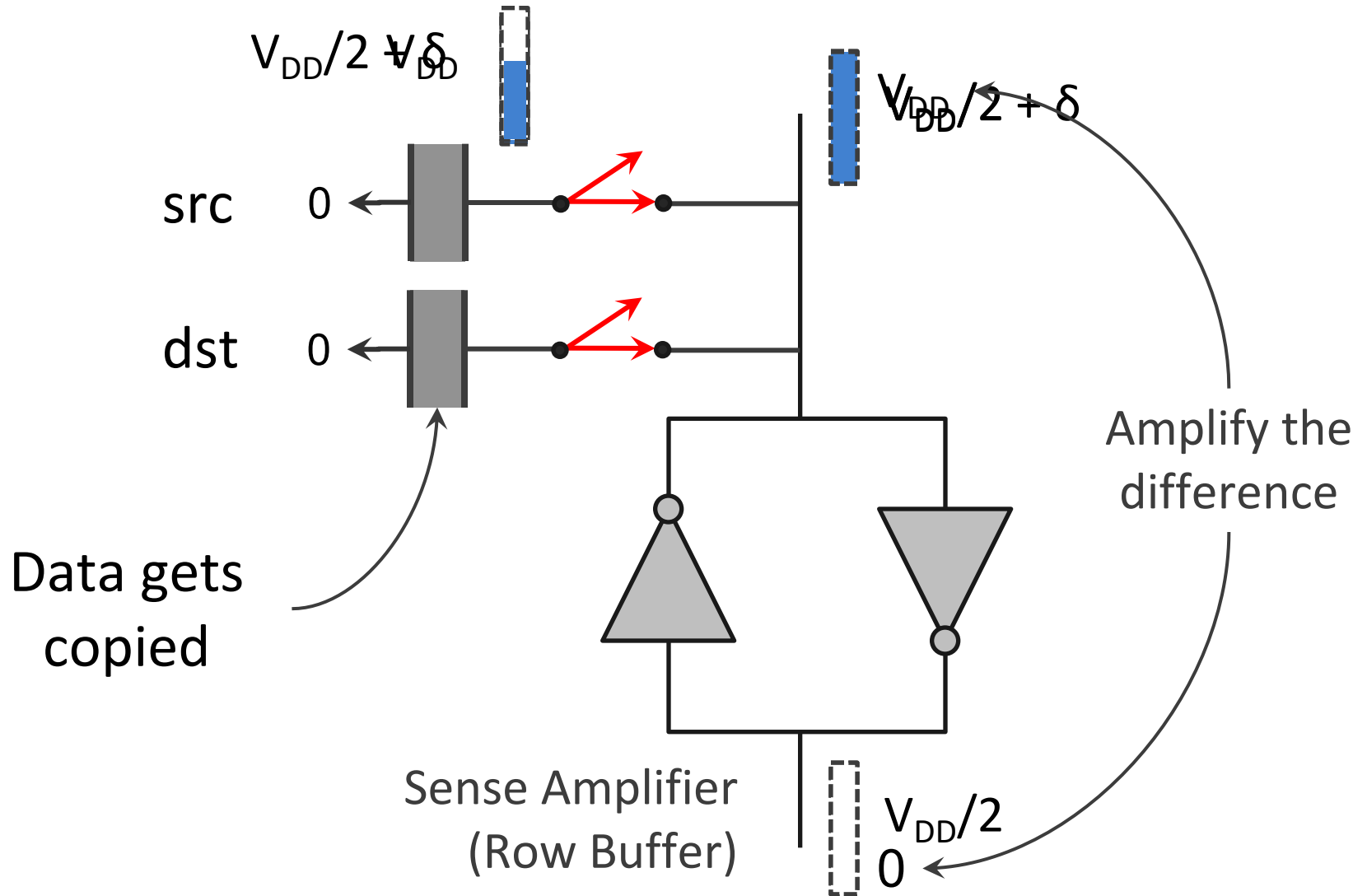
? **2. Row buffer to destination row**

RowClone: Intra-Subarray (I)

$V_{DD}/2 + \delta$ V_{DD}

src   0

dst   0

$V_{DD}/2 + \delta$

Amplify the difference

Data gets copied

Sense Amplifier
(Row Buffer)

$V_{DD}/2$
0

RowClone: Intra-Subarray (II)
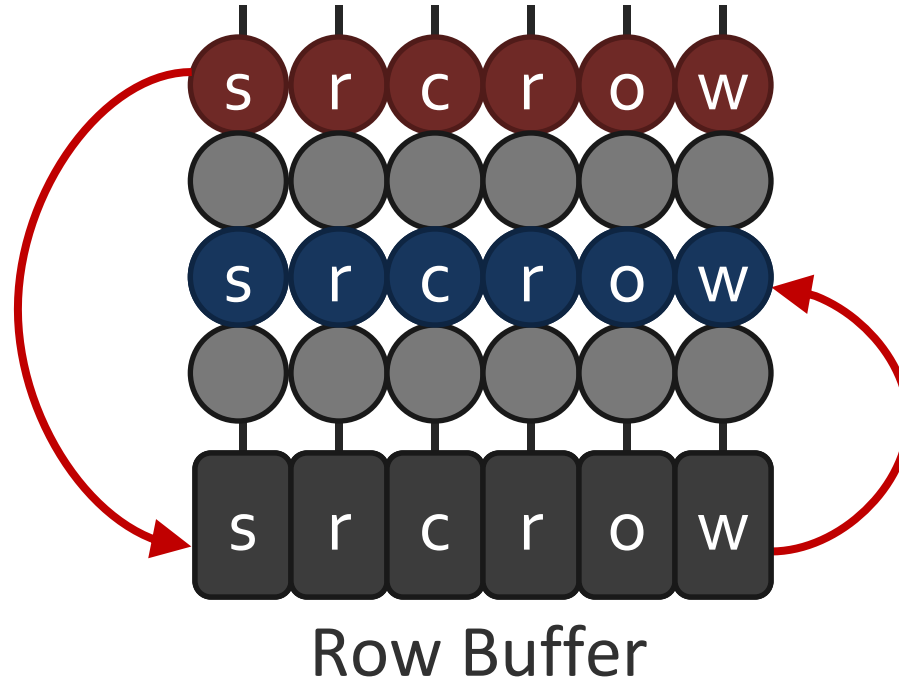
Row Buffer

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

DRAM Energy **74x** ⬇
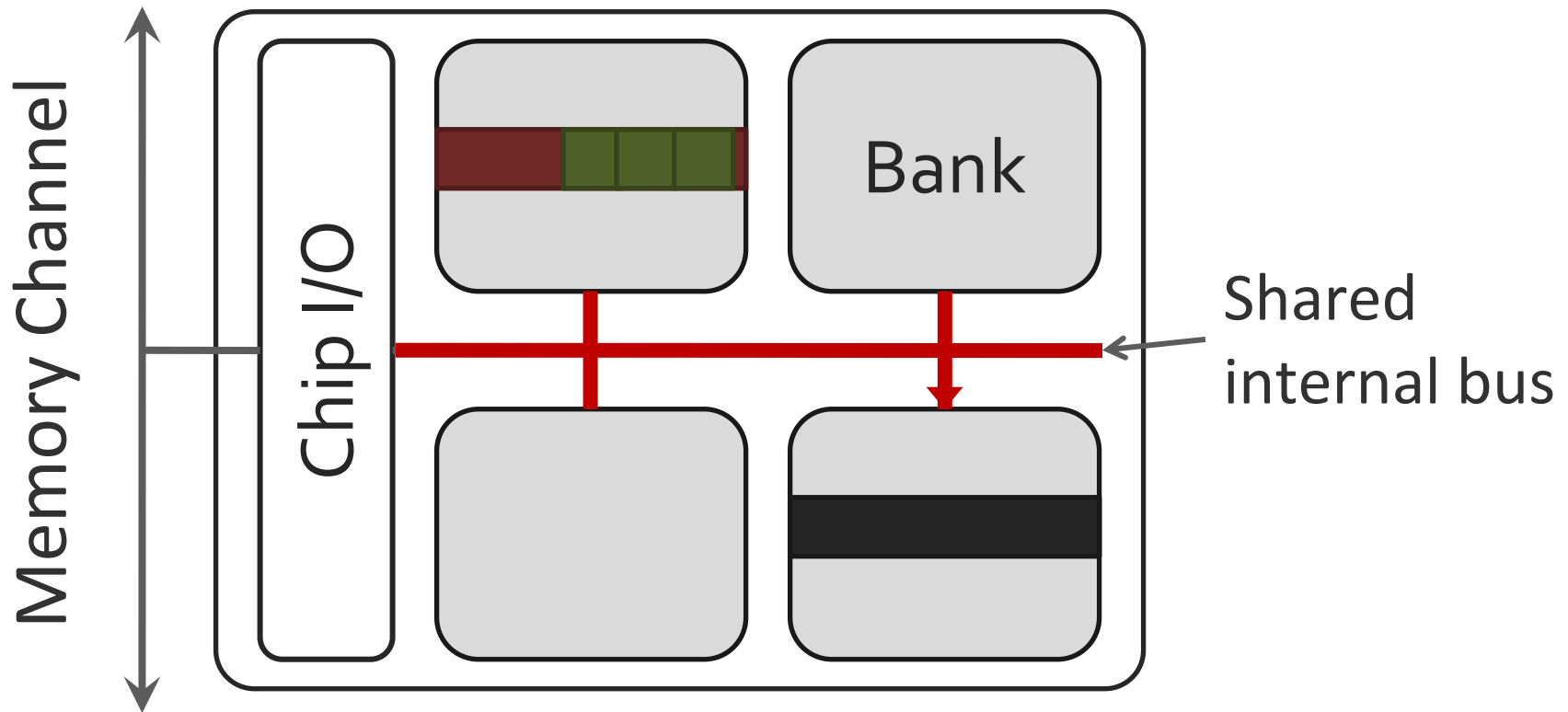
3600nJ to 40nJ

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 *all* 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

Inter Subarray Copy
(Use Inter-Bank Copy Twice)

Subarray

Bank I/O

Memory Channel

Chip I/O

Bank

Inter Bank Copy
(Pipelined
Internal RD/WR)

Intra Subarray
Copy (2 ACTs)
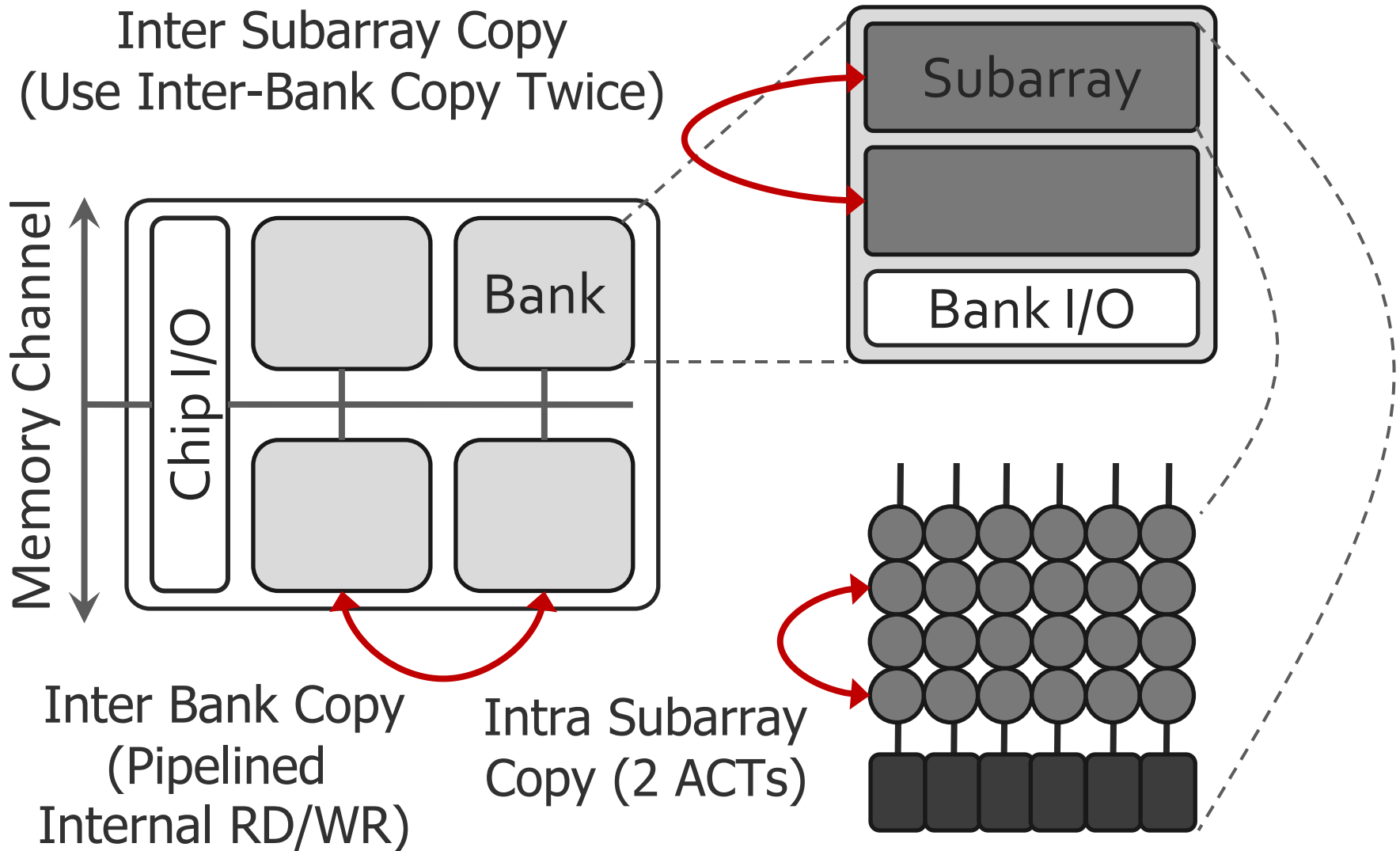
RowClone: Fast Row Initialization

0 0 0 0 0 0 0 0 0 0 0 0
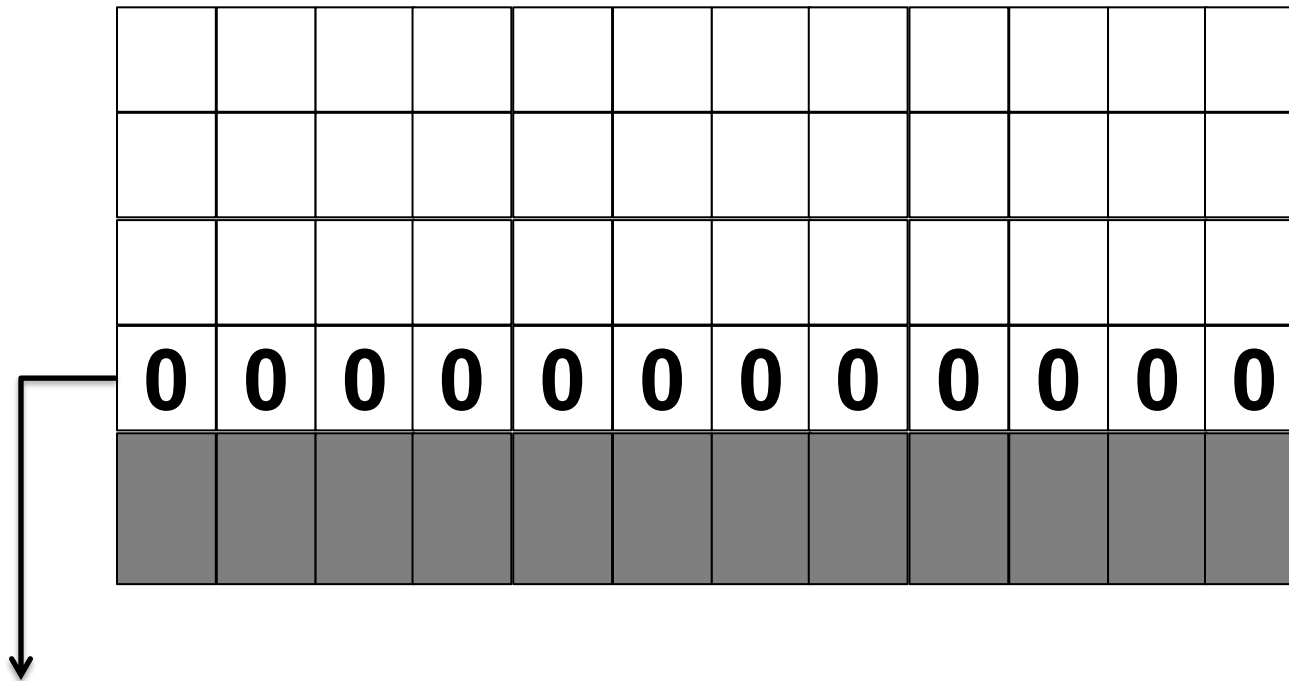
Fix a row at Zero
(0.5% loss in capacity)

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

SAFARI

RowClone: Latency & Energy Benefits

Latency Reduction

11.6x
1.9x
1.0x
6.0x

Copy | Zero

Energy Reduction

74.4x
3.2x
1.5x
41.5x
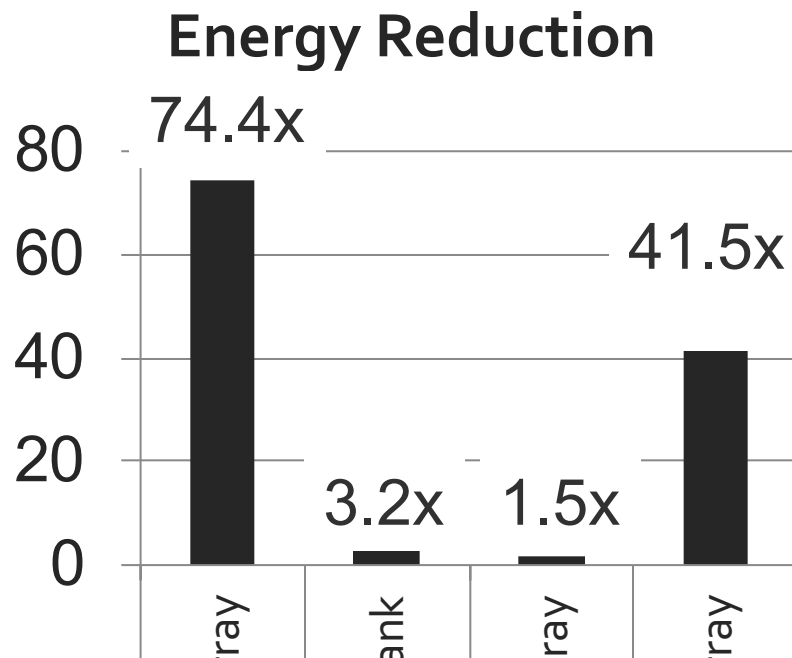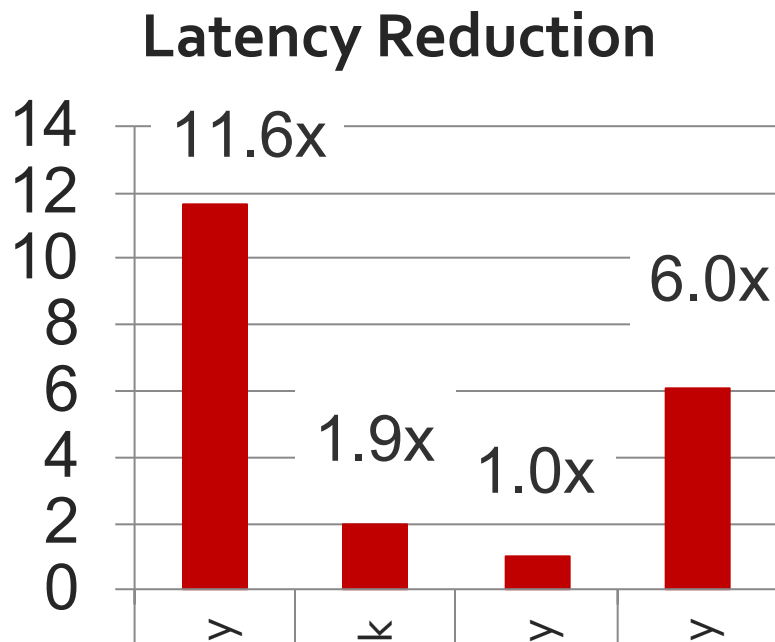
Copy | Zero

Very low cost: 0.01% increase in die area

System Design to
Enable RowClone

End-to-End System Design

Application

Operating System

ISA

Microarchitecture

DRAM (RowClone)

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

How to ensure cache coherence?

How to maximize latency and energy savings?

How to handle data reuse?

1. Hardware/Software Interface

- Two new instructions
  - memcopy 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
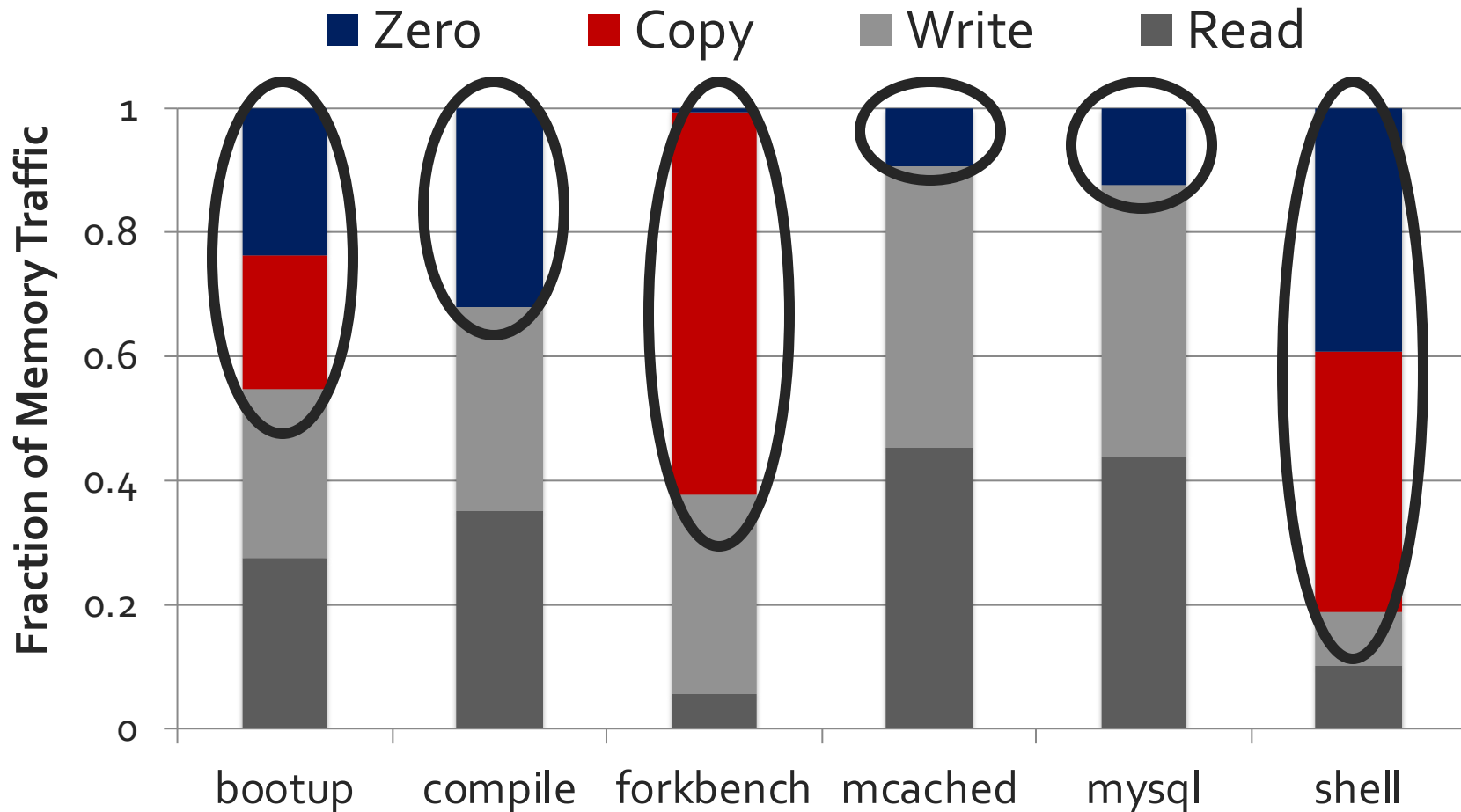
Methodology

- 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
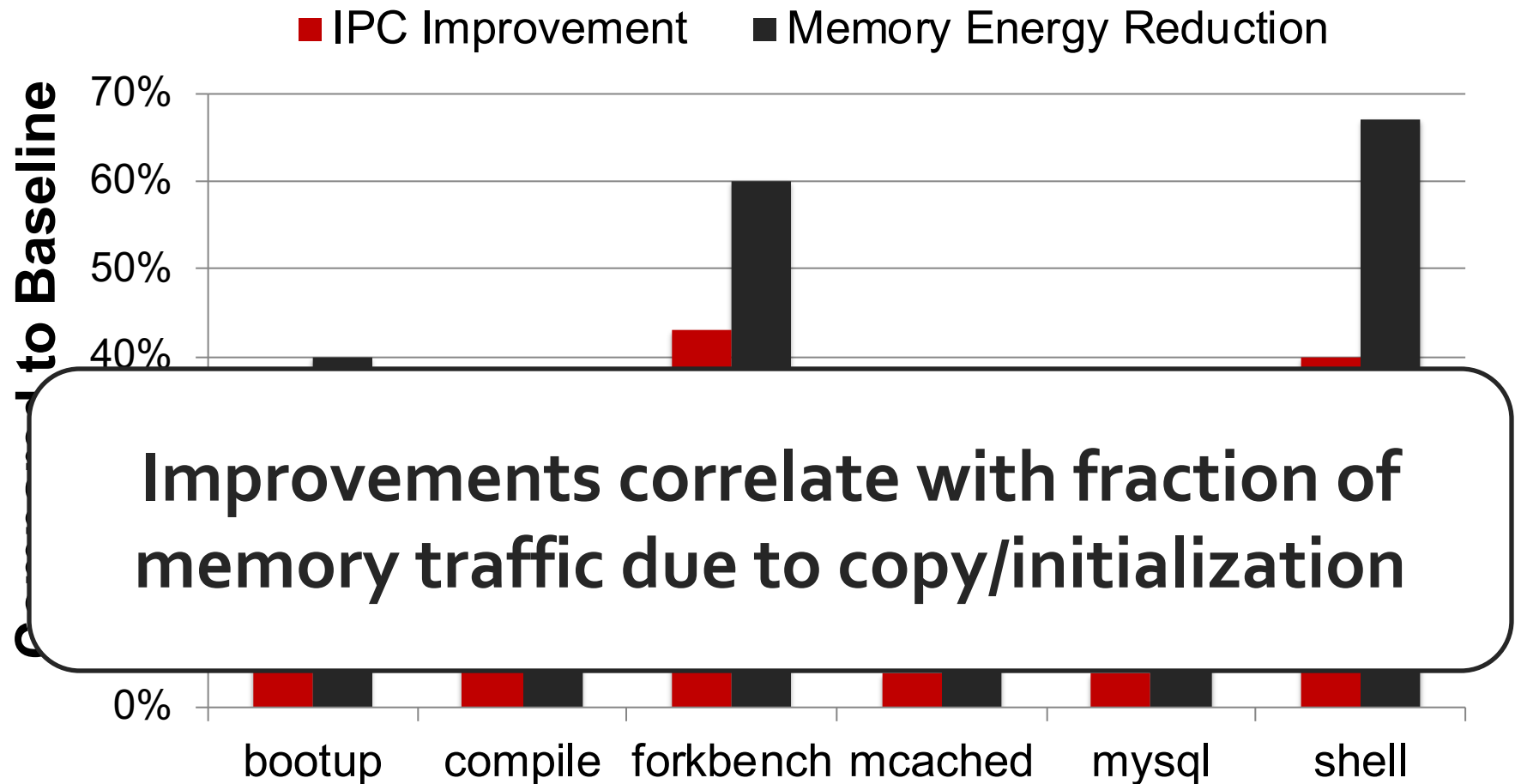
Copy/Initialization Intensive Applications

- **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

SAFARI

Single-Core – Performance and Energy

■ IPC Improvement   ■ Memory Energy Reduction

70%
60%
50%
40%

0%

bootup   compile   forkbench   mcached   mysql   shell

Improvements correlate with fraction of memory traffic due to copy/initialization

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

■ System Performance   ■ Memory Energy Efficiency
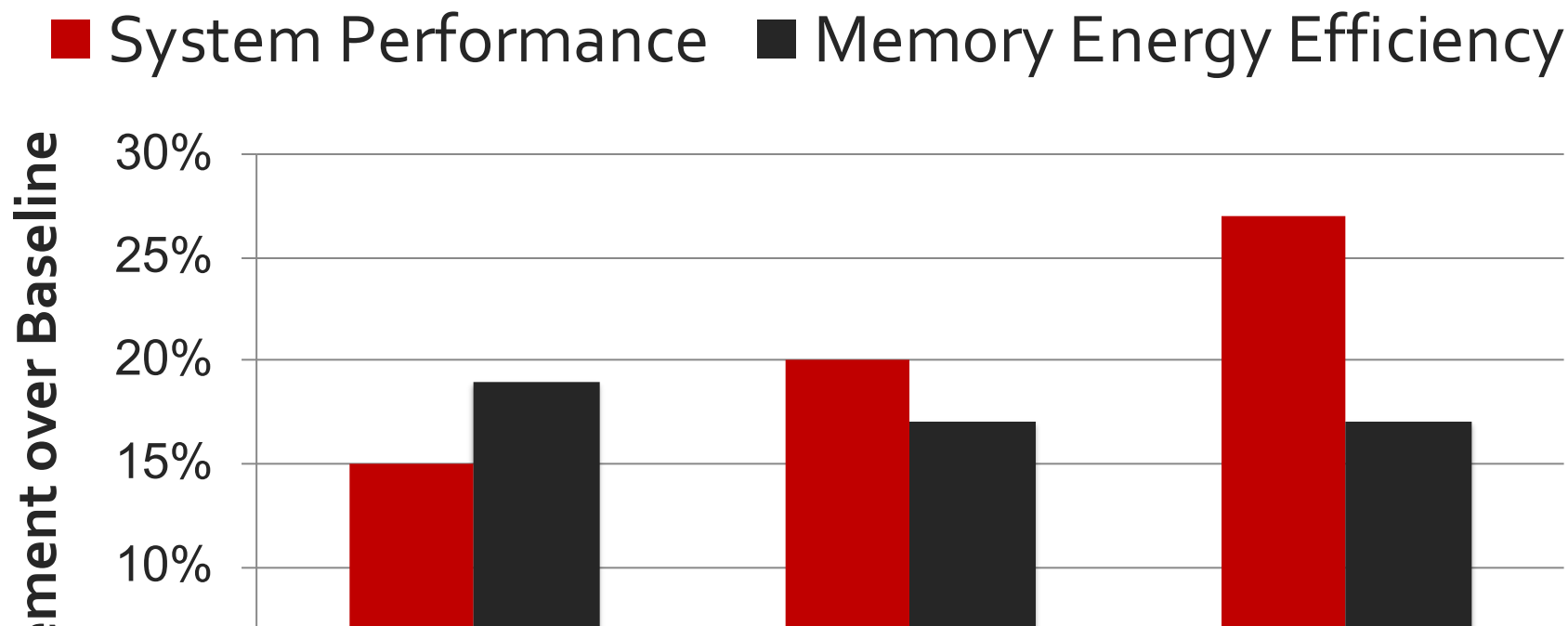
Consistent improvement in energy/instruction

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

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

Performance Analysis Rat Holes

Benchmark

Performance Criteria

Configuration

Specific Details
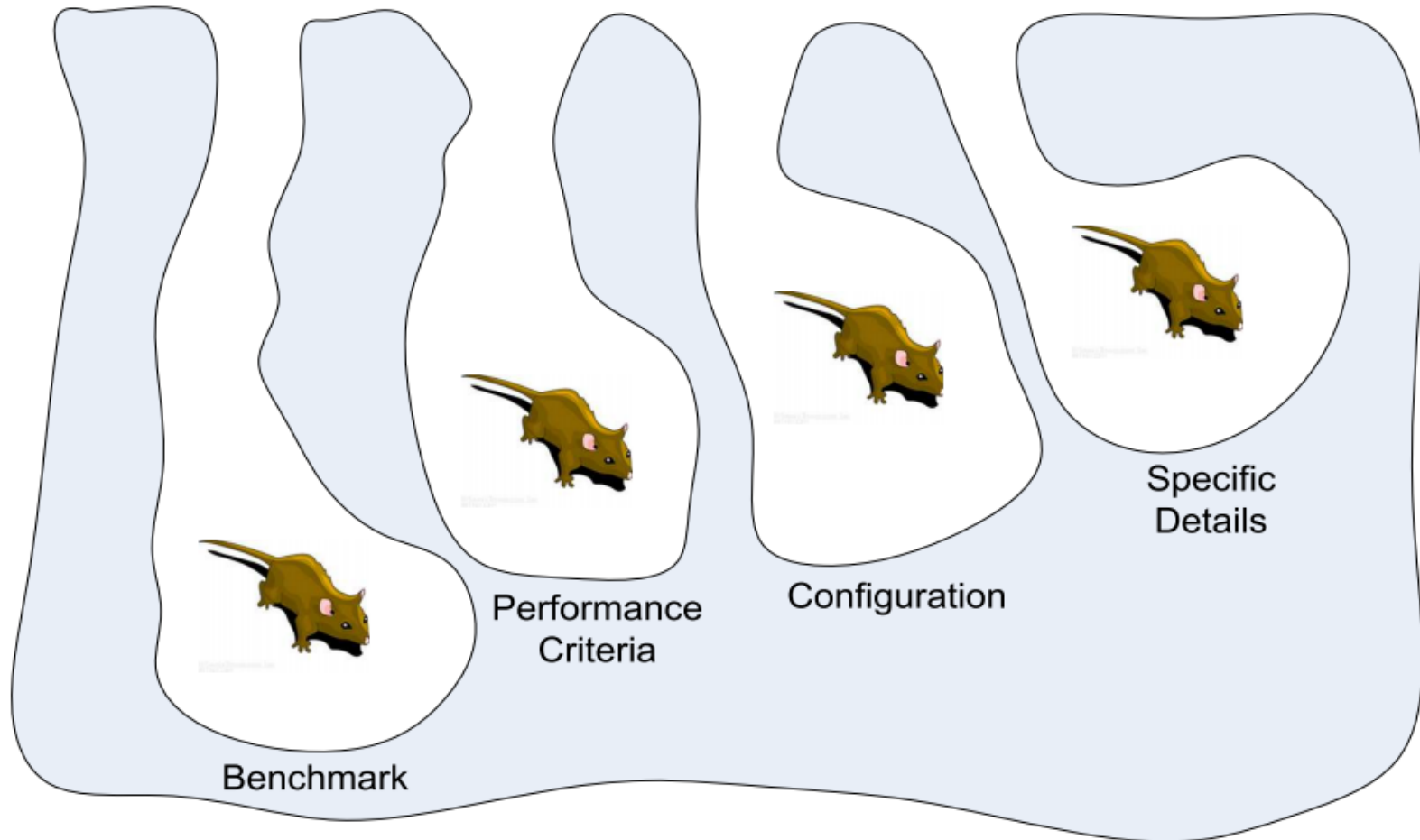
Thoughts and Ideas

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 this be extended to move data at smaller granularities?

- Can we have more efficient solutions to
  - Cache coherence (minimize overhead)
  - Data reuse after copy and initialization

- Can this idea be evaluated on a real system? How?

- 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*

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

Ambit: Bulk-Bitwise in-DRAM Computation

- 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 Hassan[4]   Amirali Boroumand[5]
Jeremie Kim[4,5]   Michael A. Kozuch[3]   Onur Mutlu[4,5]   Phillip B. Gibbons[5]   Todd C. Mowry[5]

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

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 Xu[2], Qiaosha Zou[1,5], Jishen Zhao[3], Yu Lu[4], and Yuan Xie[1]

University of California, Santa Barbara[1], Hewlett Packard Labs[2]
University of California, Santa Cruz[3], Qualcomm Inc.[4], Huawei Technologies Inc.[5]
{shuangchenli, yuanxie}ece.ucsb.edu[1]

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
    - Multiple works have already built on the paper (see LISA, Ambit, and other works in Google Scholar)

- Easy to read and understand paper

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 in some cases?

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
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**   (intel)

Some History

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 uncompelling 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.

Seminar in
Computer Architecture
Meeting 2a: Example Review I

Minesh Patel

ETH Zürich

Spring 2020

27 February 2020