Seminar in Computer Architecture
Meeting 2: Example Review: RowClone

Prof. Onur Mutlu

ETH Zürich
Spring 2021
4 March 2021
Suggested Paper Discussion Format

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

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

- Summary
- Critique

- ~20-25 minute Summary
- ~10 min Critique plus
- ~10 min Discussion
Presentation Schedule

- We will have 11 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 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
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

Background, Problem & Goal
Memory Channel – Bottleneck

Limited Bandwidth

Core
Cache
MC
Memory

High Energy
Goal: Reduce Memory Bandwidth Demand

Reduce unnecessary data movement
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 Corporation

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

\textit{memmove} & \textit{memcpy}: 5\% cycles in Google’s datacenter \cite{Kanev+ISCA’15}

- Forking
- Zero initialization (e.g., security)
- Checkpointing
- VM Cloning
- Deduplication
- Page Migration
- Many more
Shortcomings of Today’s Systems

1) High latency
2) High bandwidth utilization
3) Cache pollution
4) Unwanted data movement

1046ns, 3.6uJ (for 4KB page copy via DMA)
Novelty, Key Approach, and Ideas
RowClone: In-Memory Copy

1) Low latency
2) Low bandwidth utilization
3) No cache pollution
4) No unwanted data movement

1046ns, 3.6uJ → 90ns, 0.04uJ
RowClone: In-DRAM Row Copy

Idea: Two consecutive ACTivates
Negligible HW cost

Step 1: Activate row A
Transfer row

Step 2: Activate row B
Transfer row

DRAM subarray

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)

1. Source row to row buffer
2. Row buffer to destination row
RowClone: Intra-Subarray (I)

Data gets copied

Sense Amplifier (Row Buffer)

Amplify the difference

\[ V_{DD}/2 \rightarrow V_{DD} - \delta \rightarrow 0 \]

\[ V_{DD}/2 + \delta \rightarrow V_{DD}/2 \rightarrow 0 \]
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
1046ns to 90ns

Energy
3600nJ to 40nJ

11x

74x

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)

Inter Bank Copy
(Pipelined Internal RD/WR)

Intra Subarray Copy (2 ACTs)
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

- Intra-Subarray: 11.6x
- Inter-Bank: 1.9x
- Inter-Subarray: 1.0x
- Intra-Subarray: 6.0x

Energy Reduction

- Intra-Subarray: 74.4x
- Inter-Bank: 3.2x
- Inter-Subarray: 1.5x
- Intra-Subarray: 41.5x

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 bootstrap** (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

Fraction of Memory Traffic

Zero  Copy  Write  Read

bootup  compile  forkbench  mcached  mysql  shell
Single-Core – Performance and Energy

![Bar chart showing IPC Improvement and Memory Energy Reduction for different applications.]

**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

Workload  Metrics  Configuration  Details

Source: https://www.cse.wustl.edu/~jain/iu-cee/ftp/k_10adp.pdf
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 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"
[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
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

![Diagram of subarrays with arrows indicating data movement]

• 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"
  [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"
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"

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"

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: 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.

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
Vivek Seshadri and Onur Mutlu, "In-DRAM Bulk Bitwise Execution Engine",
[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
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 we have more efficient solutions to
  - Cache coherence (minimize overhead)
  - Data reuse after copy and initialization
Pinatubo: A Processing-in-Memory Architecture for Bulk Bitwise Operations in Emerging Non-volatile Memories

Shuangchen Li¹, Cong Xu², Qiaosha Zou¹,⁵, 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 Tziortziotis
georgios.tziortziotis@princeton.edu
Department of Electrical Engineering
Princeton University

David Wentzlaff
wentzlaff@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
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
A Modern Primer on Processing in Memory

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

SAFARI Research Group

\textsuperscript{a}ETH Zürich
\textsuperscript{b}Carnegie Mellon University
\textsuperscript{c}University of Illinois at Urbana-Champaign
\textsuperscript{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"

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


SAFARI Carnegie Mellon Intel
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.
RowClone: Fast and Efficient In-DRAM Copy and Initialization of Bulk Data

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

Review #295A 3 4 5 3
Review #295B 4 3 4 3
Review #295C 3 4 4 3
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

Accepted 1947kB Friday 31 May 2013 1:48:46pm PDT

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

+ Topics

<table>
<thead>
<tr>
<th>OveMer</th>
<th>Nov</th>
<th>WriQua</th>
<th>RevExp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Review #206A</td>
<td>5</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Review #206B</td>
<td>4</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Review #206C</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Review #206D</td>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Review #206E</td>
<td>4</td>
<td>3</td>
<td>5</td>
</tr>
</tbody>
</table>
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!
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](https://www.youtube.com/watch?v=oGcZAGwfEUE&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=12)
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...
Aside: A Recommended Book

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 it provoked needs for 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.
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/disk/...) faster; our CPUs/disk (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.
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
Suggestion to Community

We Need to Fix the Reviewer Accountability Problem
Main Memory Needs
Intelligent Controllers
Takeaway

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](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](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 Tzialazioulis
georgios.tzialazioulis@princeton.edu
Department of Electrical Engineering
Princeton University

David Wentzlauff
wentzlauff@princeton.edu
Department of Electrical Engineering
Princeton University

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 \& 0) is finally set in all three rows.
Bitline is above $V_{DD}/2$ when R2 is activated.
Bitwise AND in ComputeDRAM

- \( V_{dd}/2 \)
- \( R_3 = 00_2 \)  
  Operand: 1
- \( R_1 = 01_2 \)  
  Constant: 0

T2 very short  
PRE cannot close R1  
R3 will appear on the address bus  
ACT(R2) will activate R3 and R2

ACT(R1)  
PRE  
ACT(R2)

\( T_1 = T_2 = 0 \) idle cycle
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

<table>
<thead>
<tr>
<th>Group ID: Vendor_Size_Freq(MHz)</th>
<th>Part Num</th>
<th># Modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>SKhynix_2G_1333</td>
<td>HMT325S6BFR8C-H9</td>
<td>6</td>
</tr>
<tr>
<td>SKhynix_4G_1333</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>SKhynix_4G_1333</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>SKhynix_4G_1333</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>SKhynix_4G_1333</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>Samsung_4G_1333</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>Samsung_4G_1333</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>Micron_2G_1333</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>Micron_2G_1333</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>Elpida_2G_1333</td>
<td>EBJ21UE8BDS0-DJ-F</td>
<td>2</td>
</tr>
<tr>
<td>Nanya_4G_1333</td>
<td>NT4GC64B8HG0NS-CG</td>
<td>2</td>
</tr>
<tr>
<td>TimeTec_4G_1333</td>
<td>78AP10NUS2R2-4G</td>
<td>2</td>
</tr>
<tr>
<td>Corsair_4G_1333</td>
<td>CMSA8GX3M2A1333C9</td>
<td>2</td>
</tr>
</tbody>
</table>

**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
Each grid represents the success ratio of operations for a specific DDR3 module.
Pinatubo: A Processing-in-Memory Architecture for Bulk Bitwise Operations in Emerging Non-volatile Memories

Shuangchen Li\textsuperscript{1,*}, Cong Xu\textsuperscript{2}, Qiaosha Zou\textsuperscript{1,5}, Jishen Zhao\textsuperscript{3}, Yu Lu\textsuperscript{4}, and Yuan Xie\textsuperscript{1}

University of California, Santa Barbara\textsuperscript{1}, Hewlett Packard Labs\textsuperscript{2}
University of California, Santa Cruz\textsuperscript{3}, Qualcomm Inc.\textsuperscript{4}, Huawei Technologies Inc.\textsuperscript{5}
{shuangchenli, yuanxie}@ece.ucsb.edu\textsuperscript{1}
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
Focus on learning and scholarship
The quality of your work defines your impact.
An Interview on Research and Education

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

SAFARI  https://www.youtube.com/onurmutlulectures
Seminar in Computer Architecture
Meeting 2: Example Review: RowClone

Prof. Onur Mutlu

ETH Zürich
Spring 2021
4 March 2021