

Computer Architecture

Lecture 7: Computation in Memory II

Prof. Onur Mutlu

ETH Zürich

Fall 2019

10 October 2019

Sub-Agenda: In-Memory Computation

- Major Trends Affecting Main Memory
- The Need for Intelligent Memory Controllers
 - Bottom Up: Push from Circuits and Devices
 - Top Down: Pull from Systems and Applications
- Processing in Memory: Two Directions
 - Minimally Changing Memory Chips
 - Exploiting 3D-Stacked Memory
- How to Enable Adoption of Processing in Memory
- Conclusion

Processing in Memory: Two Approaches

1. Minimally changing memory chips
2. Exploiting 3D-stacked memory

Recall: 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

Recall: End-to-End System Design

Application

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

Operating System

How to ensure cache coherence?

ISA

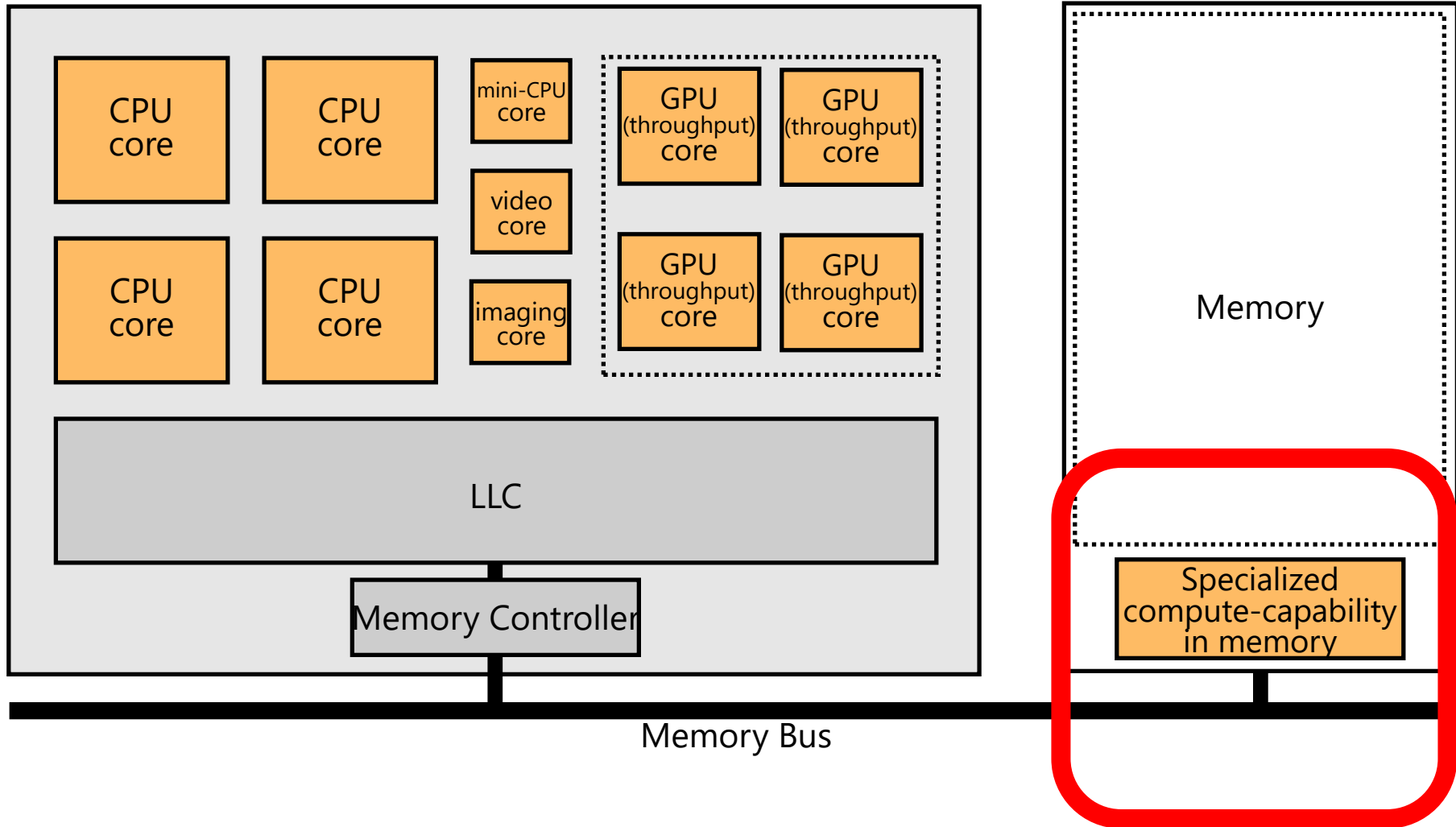
Microarchitecture

How to maximize latency and energy savings?

DRAM (RowClone)

How to handle data reuse?

Memory as an Accelerator

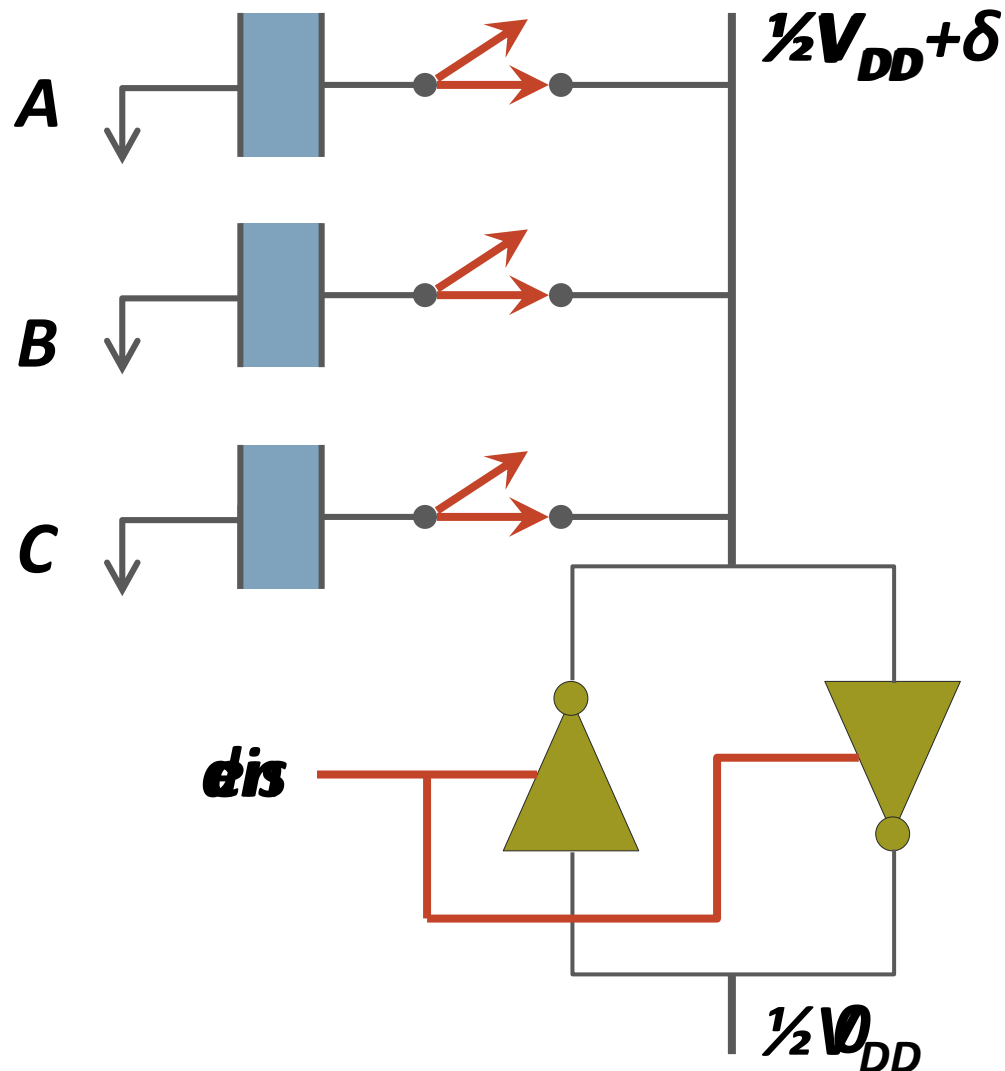


Memory similar to a "conventional" accelerator

In-Memory Bulk Bitwise Operations

- 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

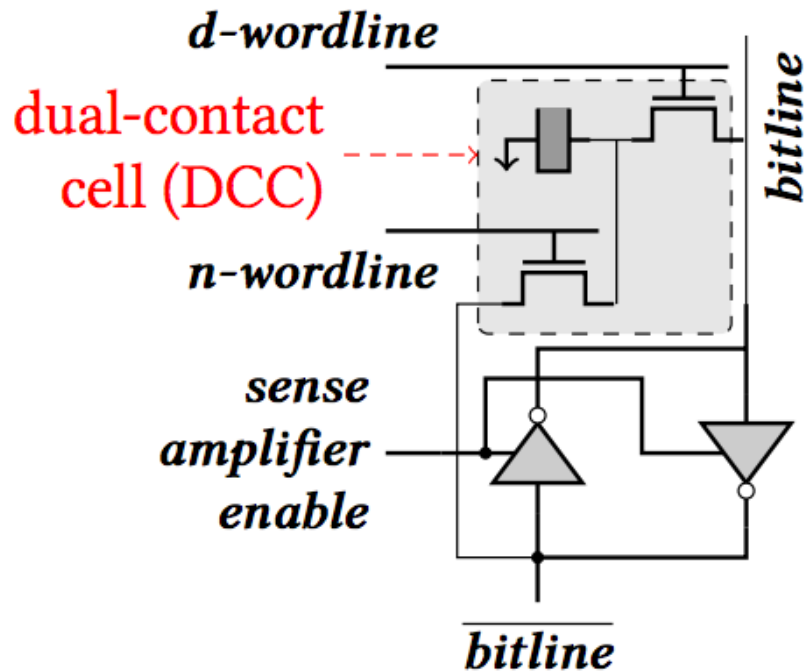


Figure 5: A dual-contact cell connected to both ends of a sense amplifier

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

In-DRAM NOT Operation

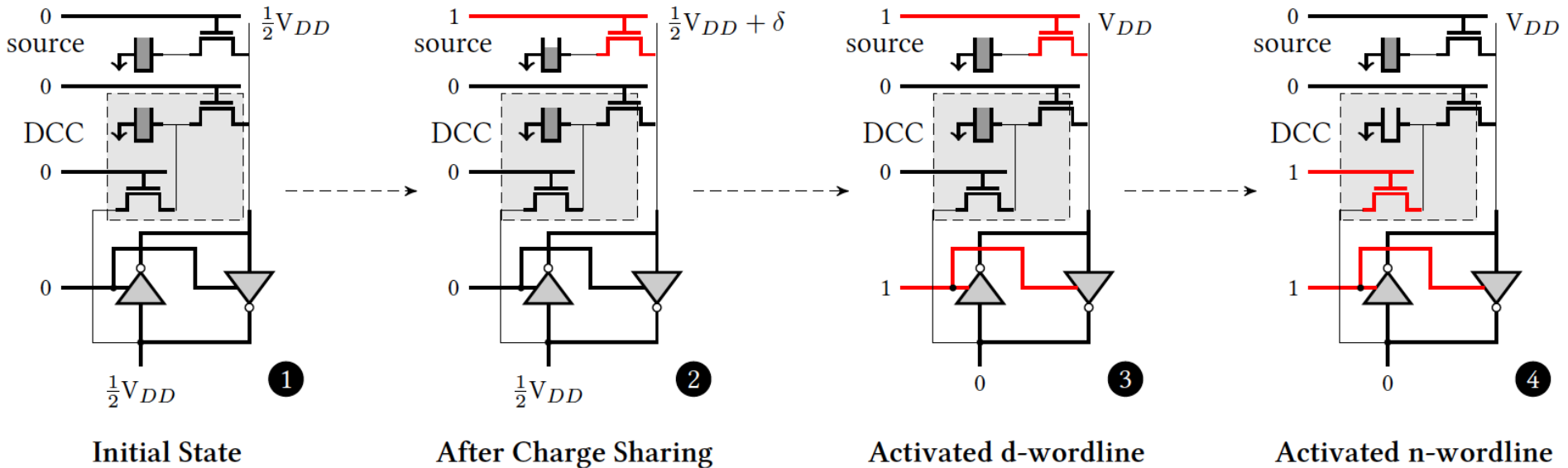


Figure 5: Bitwise NOT using a dual contact capacitor

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

Performance: In-DRAM Bitwise Operations

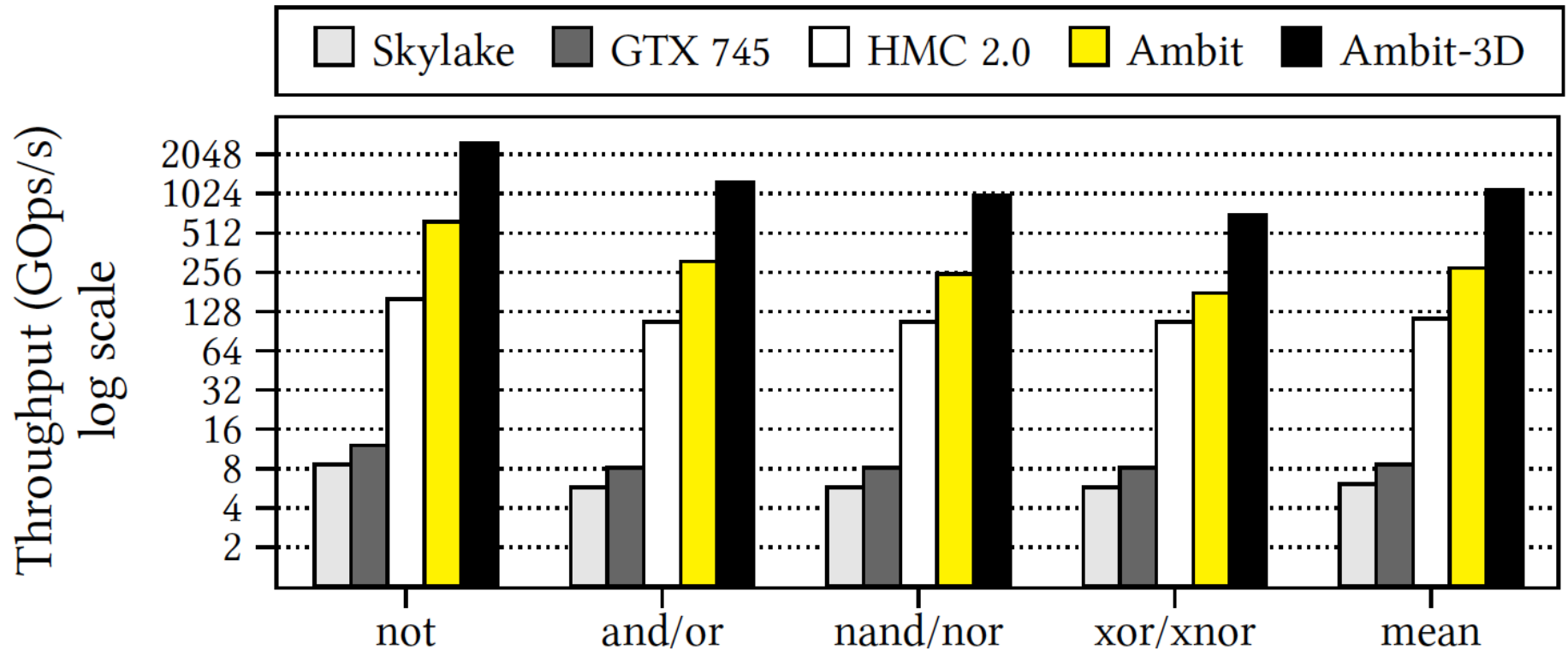


Figure 9: Throughput of bitwise operations on various systems.

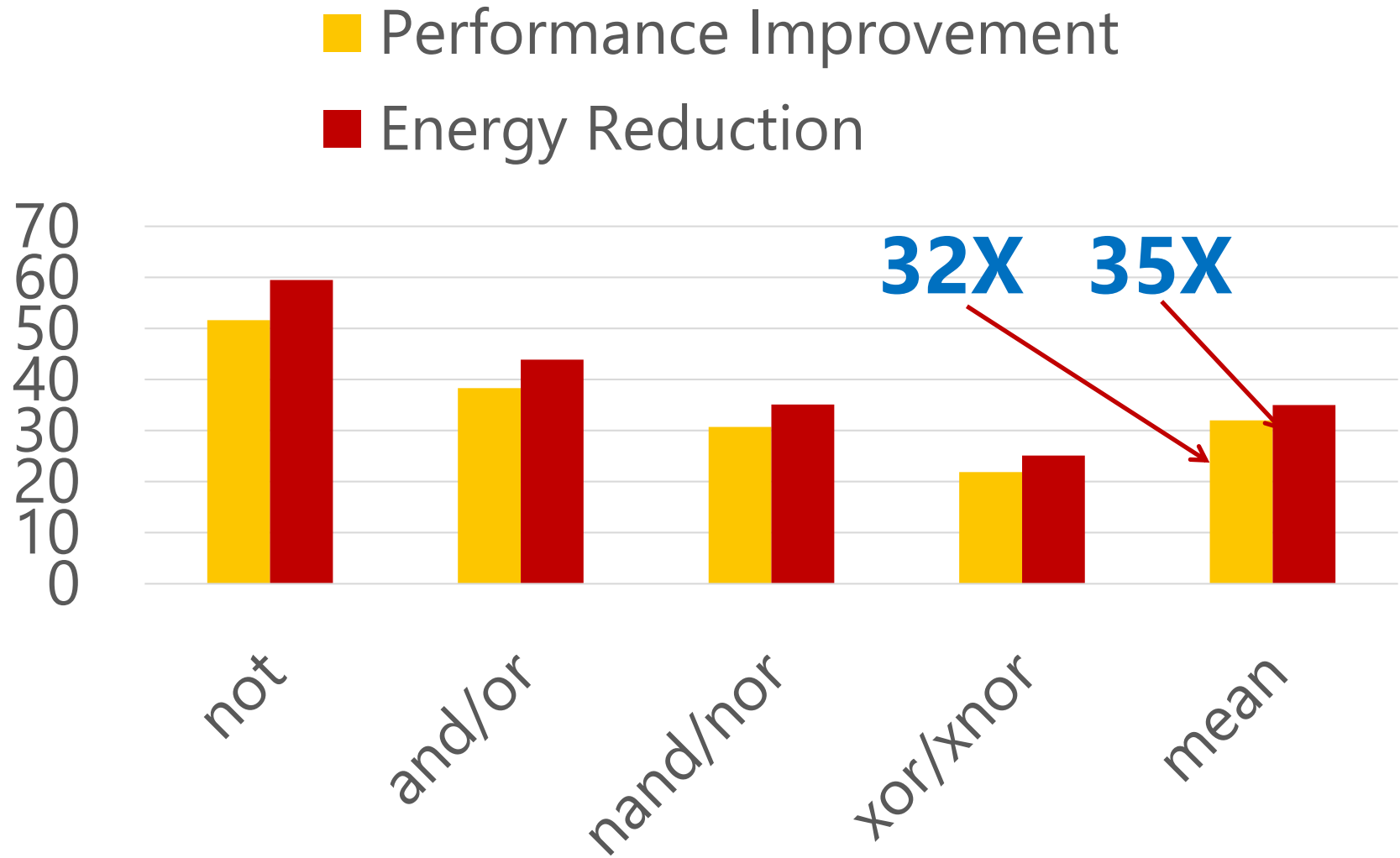
Energy of In-DRAM Bitwise Operations

	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

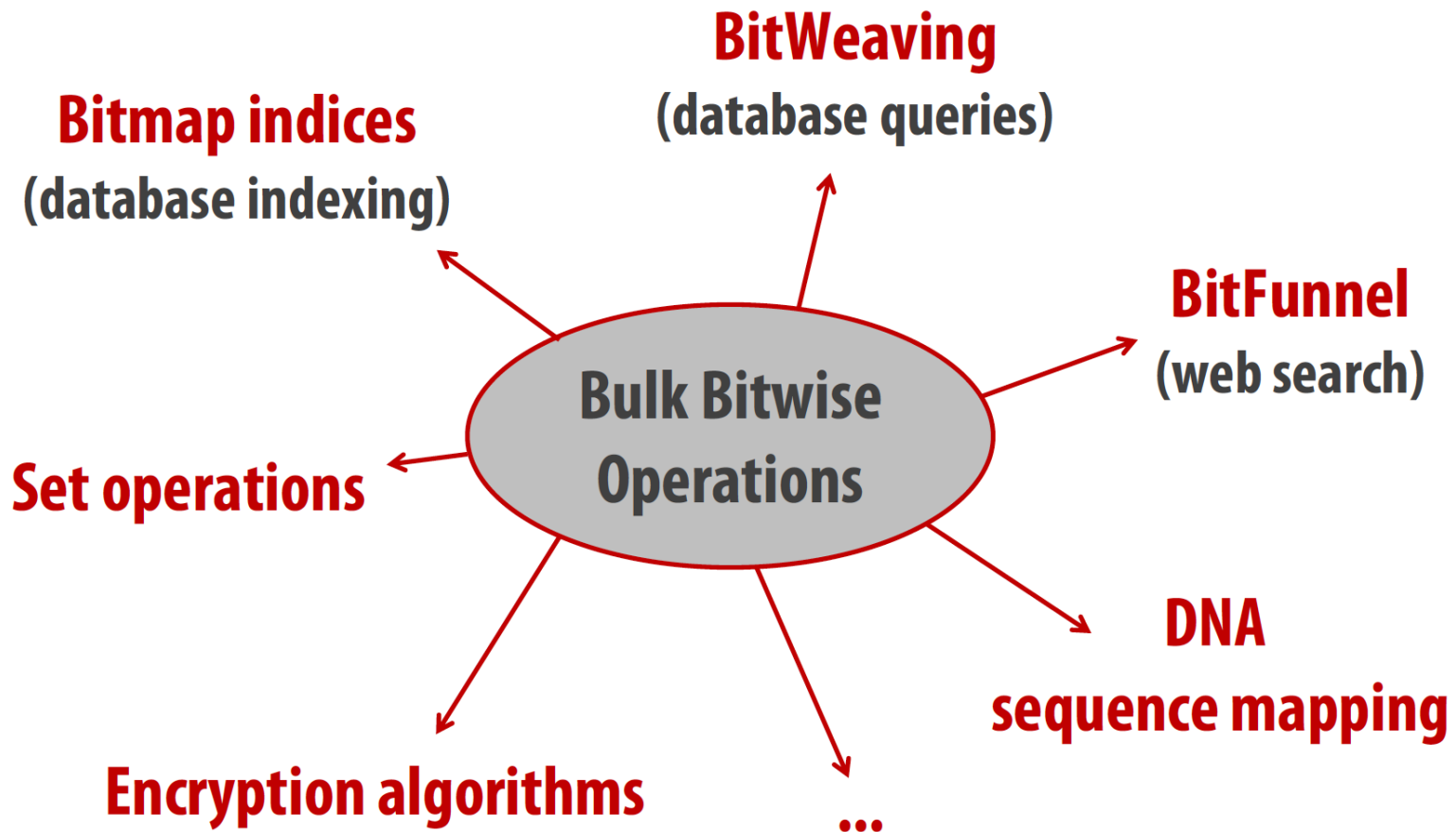
Table 3: Energy of bitwise operations. (↓) indicates energy reduction of Ambit over the traditional DDR3-based design.

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

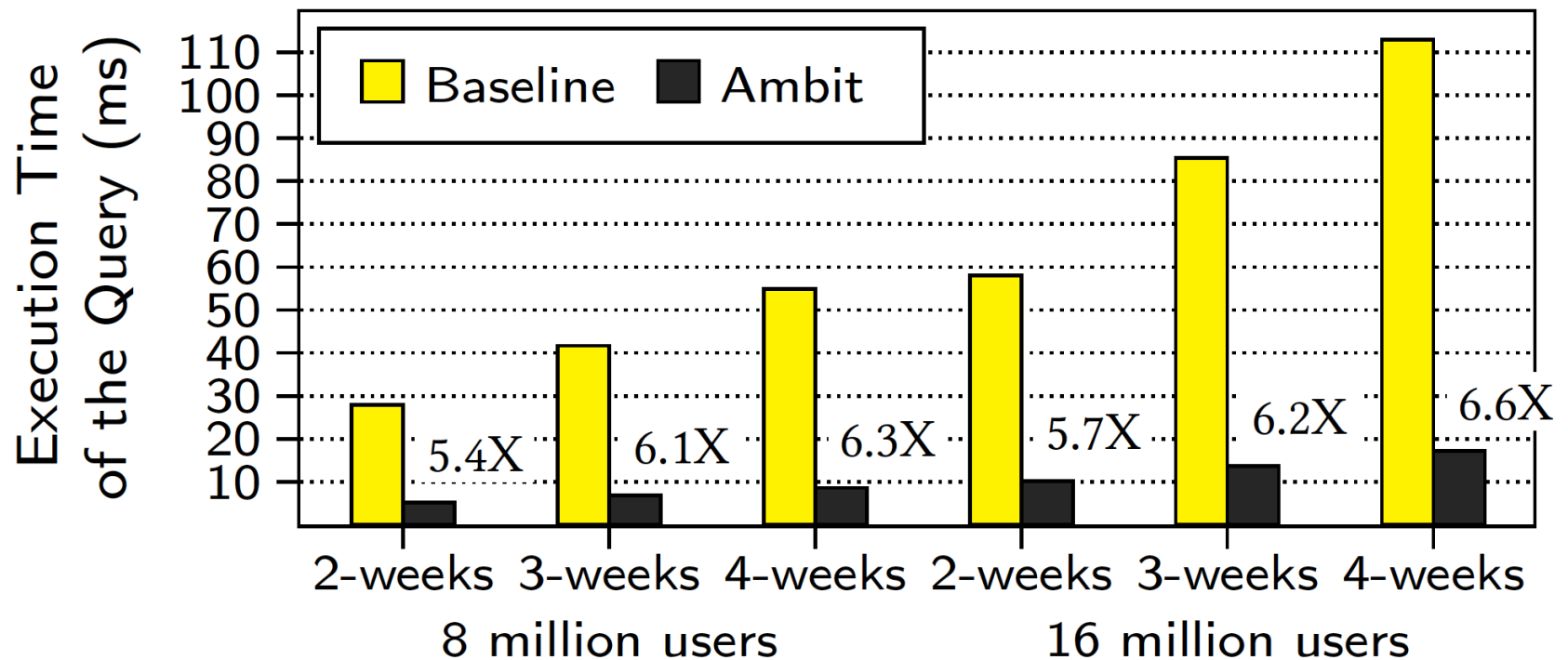


Figure 10: Bitmap index performance. The value above each bar indicates the reduction in execution time due to Ambit.

>5.4-6.6X Performance Improvement

Performance: BitWeaving on Ambit

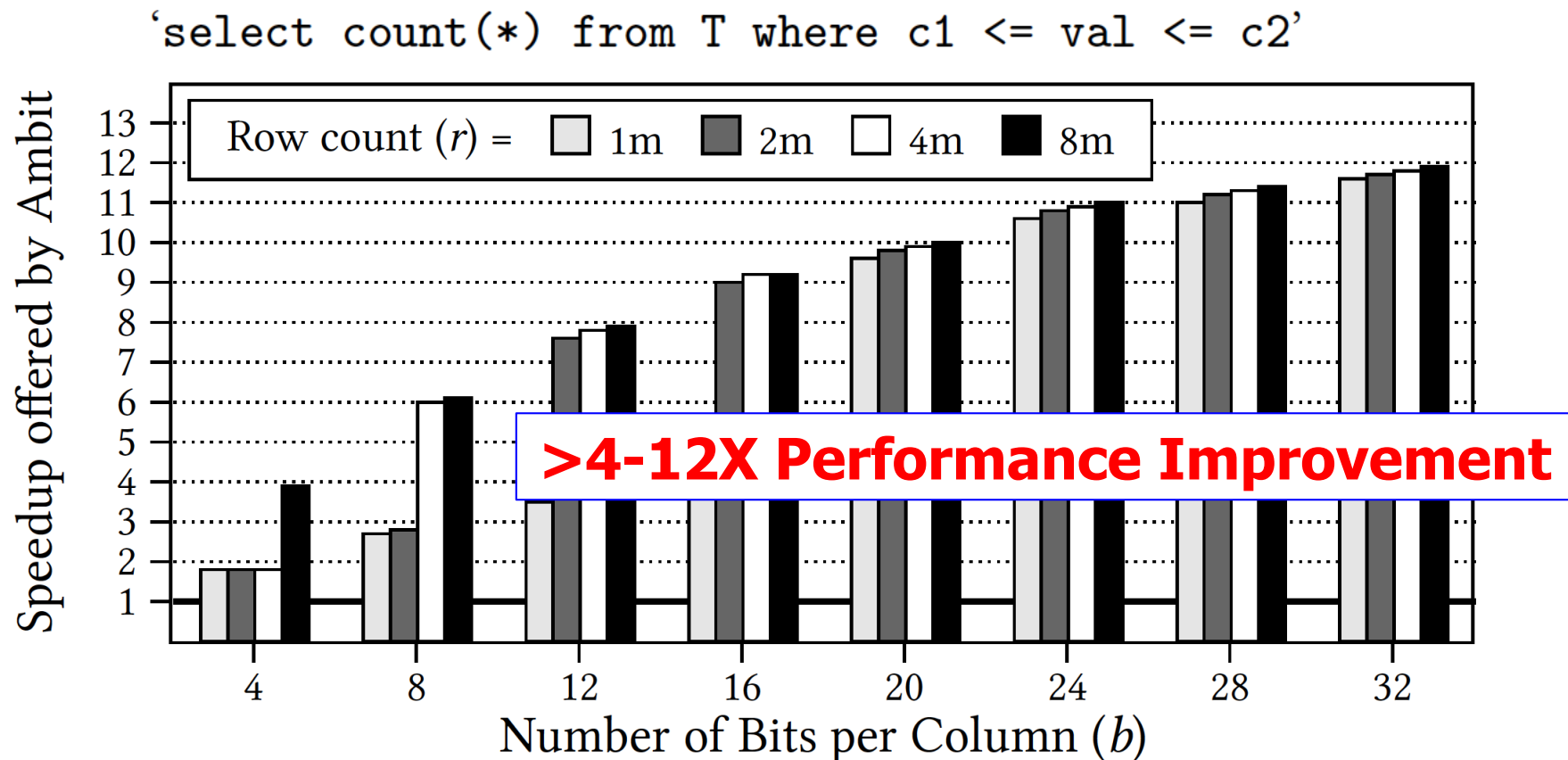


Figure 11: Speedup offered by Ambit over baseline CPU with SIMD for BitWeaving

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`

Challenge: Intelligent Memory Device

Does **memory**
have to be
dumb?

Computing Architectures with Minimal Data Movement

A Detour on the Review Process

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.

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.

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.

The Reviewer Accountability Problem

Acknowledgments

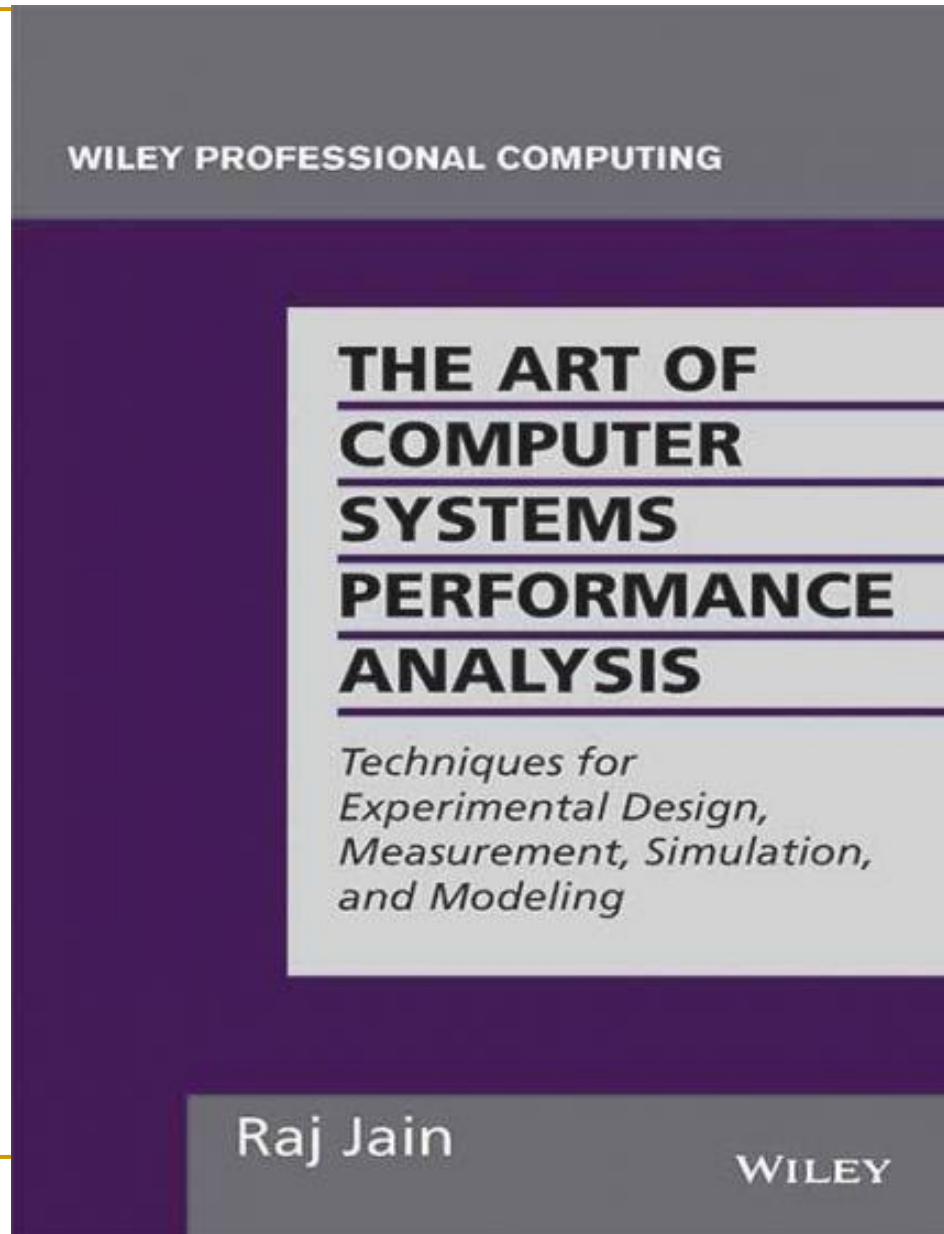
We thank the reviewers of ISCA 2016/2017, MICRO 2016/2017, and HPCA 2017 for their valuable comments. We

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



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.

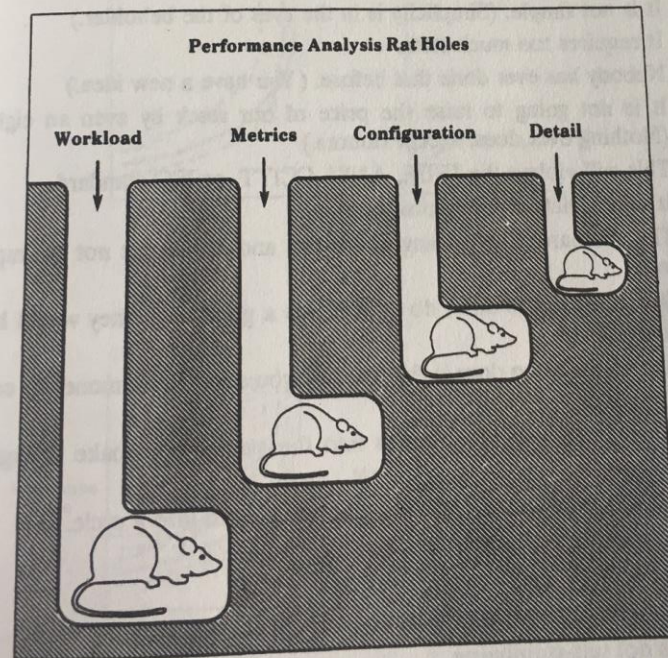


FIGURE 10.26 Four issues in performance presentations that commonly lead to endless discussion.

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.

We Need to Fix the Reviewer Accountability Problem

Main Memory Needs Intelligent Controllers

Research Community
Needs

Accountable Reviewers

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
- Be constructive, not destructive
- Do not have double standards...

Do not block or delay scientific progress for non-reasons

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

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

Other Examples of “Why Change? It’s Working OK!”

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 on the Road

Another Example

Initial RowHammer Reviews

Disturbance Errors in DRAM: Demonstration, Characterization, and Prevention

Rejected (R2)

 863kB

Friday 31 May 2013 2:00:53pm PDT

b9bf06021da54cddf4cd0b3565558a181868b972

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

+ ABSTRACT

+ AUTHORS

	OveMer	Nov	WriQua	RevExp
Review #66A	1	4	4	4
Review #66B	5	4	5	3
Review #66C	2	3	5	4
Review #66D	1	2	3	4
Review #66E	4	4	4	3
Review #66F	2	4	4	3

Missing the Point **Reviews from Micro 2013**

PAPER WEAKNESSES

This is an excellent test methodology paper, but there is no micro-architectural or architectural content.

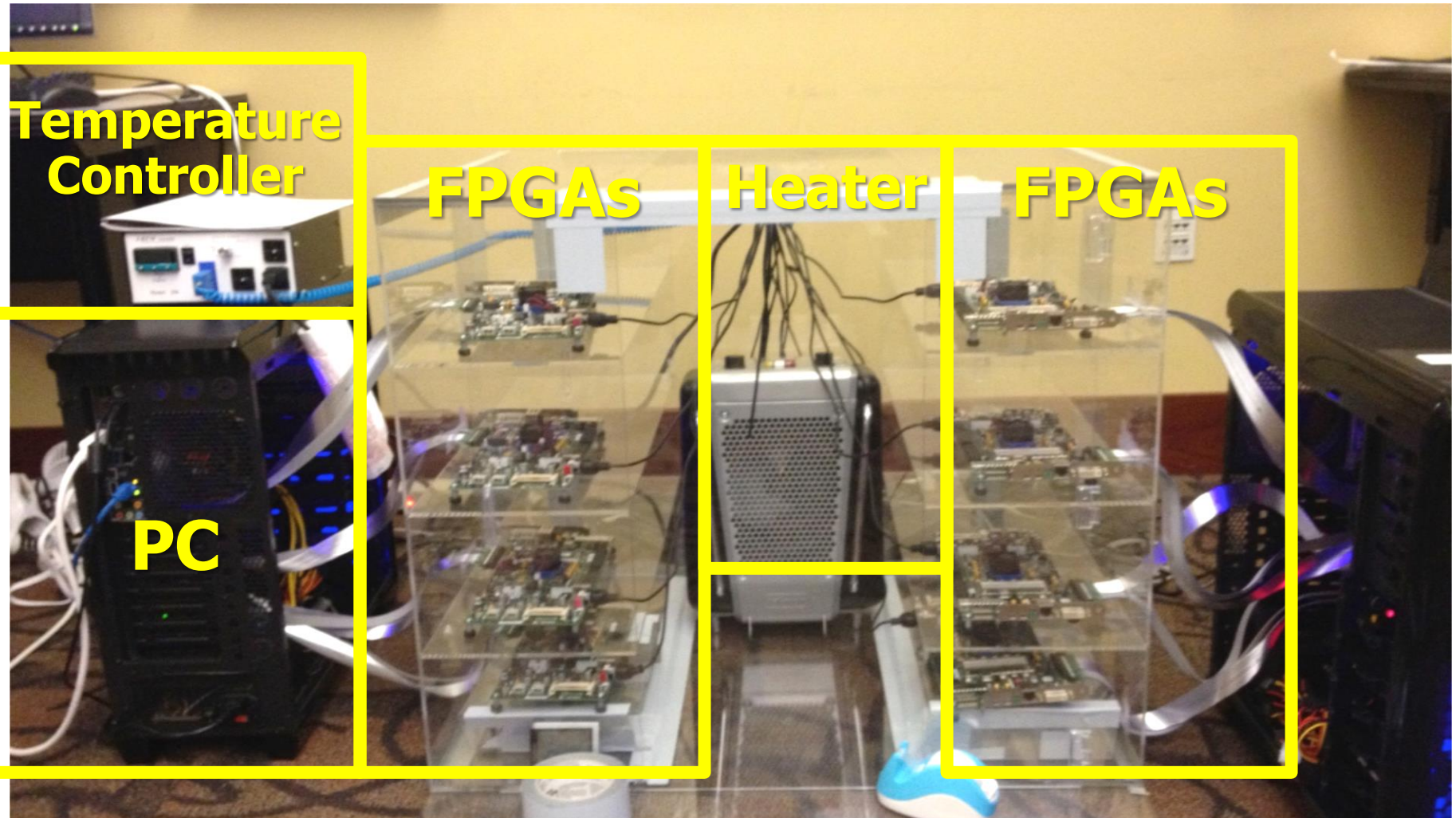
PAPER WEAKNESSES

- Whereas they show disturbance may happen in DRAM array, authors don't show it can be an issue in realistic DRAM usage scenario
- Lacks architectural/microarchitectural impact on the DRAM disturbance analysis

PAPER WEAKNESSES

The mechanism investigated by the authors is one of many well known disturb mechanisms. The paper does not discuss the root causes to sufficient depth and the importance of this mechanism compared to others. Overall the length of the sections restating known information is much too long in relation to new work.

Experimental DRAM Testing Infrastructure



Tested DRAM Modules (129 total)

Manufacturer	Module	Date*	Timing [†]		Organization		Chip		Victims-per-Module			RI _{th} (ms)	
		(yy-ww)	Freq (MT/s)	t _{RC} (ns)	Size (GB)	Chips	Size (Gb) [‡]	Pins	DieVersion [§]	Average	Minimum	Maximum	Min
Total of 43 Modules	A ₁	10-08	1066	50.625	0.5	4	1	×16	B	0	0	0	–
	A ₂	10-20	1066	50.625	1	8	1	×8	F	0	0	0	–
	A ₃₋₅	10-20	1066	50.625	0.5	4	1	×16	B	0	0	0	–
	A ₆₋₇	11-24	1066	49.125	1	4	2	×16	D	7.8 × 10 ¹	5.2 × 10 ¹	1.0 × 10 ²	21.3
	A ₈₋₁₂	11-26	1066	49.125	1	4	2	×16	D	2.4 × 10 ²	5.4 × 10 ¹	4.4 × 10 ²	16.4
	A ₁₃₋₁₄	11-50	1066	49.125	1	4	2	×16	D	8.8 × 10 ¹	1.7 × 10 ¹	1.6 × 10 ²	26.2
	A ₁₅₋₁₆	12-22	1600	50.625	1	4	2	×16	D	9.5	9	1.0 × 10 ¹	34.4
	A ₁₇₋₁₈	12-26	1600	49.125	2	8	2	×8	M	1.2 × 10 ²	3.7 × 10 ¹	2.0 × 10 ²	21.3
	A ₁₉₋₃₀	12-40	1600	48.125	2	8	2	×8	K	8.6 × 10 ⁶	7.0 × 10 ⁶	1.0 × 10 ⁷	8.2
	A ₃₁₋₃₄	13-02	1600	48.125	2	8	2	×8	–	1.8 × 10 ⁶	1.0 × 10 ⁶	3.5 × 10 ⁶	11.5
	A ₃₅₋₃₆	13-14	1600	48.125	2	8	2	×8	–	4.0 × 10 ¹	1.9 × 10 ¹	6.1 × 10 ¹	21.3
	A ₃₇₋₃₈	13-20	1600	48.125	2	8	2	×8	K	1.7 × 10 ⁶	1.4 × 10 ⁶	2.0 × 10 ⁶	9.8
	A ₃₉₋₄₀	13-28	1600	48.125	2	8	2	×8	K	5.7 × 10 ⁴	5.4 × 10 ⁴	6.0 × 10 ⁴	16.4
	A ₄₁	14-04	1600	49.125	2	8	2	×8	–	2.7 × 10 ⁵	2.7 × 10 ⁵	2.7 × 10 ⁵	18.0
	A ₄₂₋₄₃	14-04	1600	48.125	2	8	2	×8	K	0.5	0	1	62.3
Total of 54 Modules	B ₁	08-49	1066	50.625	1	8	1	×8	D	0	0	0	–
	B ₂	09-49	1066	50.625	1	8	1	×8	E	0	0	0	–
	B ₃	10-19	1066	50.625	1	8	1	×8	F	0	0	0	–
	B ₄	10-31	1333	49.125	2	8	2	×8	C	0	0	0	–
	B ₅	11-13	1333	49.125	2	8	2	×8	C	0	0	0	–
	B ₆	11-16	1066	50.625	1	8	1	×8	F	0	0	0	–
	B ₇	11-19	1066	50.625	1	8	1	×8	F	0	0	0	–
	B ₈	11-25	1333	49.125	2	8	2	×8	C	0	0	0	–
	B ₉	11-37	1333	49.125	2	8	2	×8	D	1.9 × 10 ⁶	1.9 × 10 ⁶	1.9 × 10 ⁶	11.5
	B ₁₀₋₁₂	11-46	1333	49.125	2	8	2	×8	D	2.2 × 10 ⁶	1.5 × 10 ⁶	2.7 × 10 ⁶	11.5
	B ₁₃	11-49	1333	49.125	2	8	2	×8	C	0	0	0	–
	B ₁₄	12-01	1866	47.125	2	8	2	×8	D	9.1 × 10 ⁵	9.1 × 10 ⁵	9.1 × 10 ⁵	9.8
	B ₁₅₋₃₁	12-10	1866	47.125	2	8	2	×8	D	9.8 × 10 ⁵	7.8 × 10 ⁵	1.2 × 10 ⁶	11.5
	B ₃₂	12-25	1600	48.125	2	8	2	×8	E	7.4 × 10 ⁵	7.4 × 10 ⁵	7.4 × 10 ⁵	11.5
	B ₃₃₋₄₂	12-28	1600	48.125	2	8	2	×8	E	5.2 × 10 ⁵	1.9 × 10 ⁵	7.3 × 10 ⁵	11.5
B ₄₃₋₄₇	12-31	1600	48.125	2	8	2	×8	E	4.0 × 10 ⁵	2.9 × 10 ⁵	5.5 × 10 ⁵	13.1	
B ₄₈₋₅₁	13-19	1600	48.125	2	8	2	×8	E	1.1 × 10 ⁵	7.4 × 10 ⁴	1.4 × 10 ⁵	14.7	
B ₅₂₋₅₃	13-40	1333	49.125	2	8	2	×8	D	2.6 × 10 ⁴	2.3 × 10 ⁴	2.9 × 10 ⁴	21.3	
B ₅₄	14-07	1333	49.125	2	8	2	×8	D	7.5 × 10 ³	7.5 × 10 ³	7.5 × 10 ³	26.2	
Total of 32 Modules	C ₁	10-18	1333	49.125	2	8	2	×8	A	0	0	0	–
	C ₂	10-20	1066	50.625	2	8	2	×8	A	0	0	0	–
	C ₃	10-22	1066	50.625	2	8	2	×8	A	0	0	0	–
	C ₄₋₅	10-26	1333	49.125	2	8	2	×8	B	8.9 × 10 ²	6.0 × 10 ²	1.2 × 10 ³	29.5
	C ₆	10-43	1333	49.125	1	8	1	×8	T	0	0	0	–
	C ₇	10-51	1333	49.125	2	8	2	×8	B	4.0 × 10 ²	4.0 × 10 ²	4.0 × 10 ²	29.5
	C ₈	11-12	1333	46.25	2	8	2	×8	B	6.9 × 10 ²	6.9 × 10 ²	6.9 × 10 ²	21.3
	C ₉	11-19	1333	46.25	2	8	2	×8	B	9.2 × 10 ²	9.2 × 10 ²	9.2 × 10 ²	27.9
	C ₁₀	11-31	1333	49.125	2	8	2	×8	B	3	3	3	39.3
	C ₁₁	11-42	1333	49.125	2	8	2	×8	B	1.6 × 10 ²	1.6 × 10 ²	1.6 × 10 ²	39.3
	C ₁₂	11-48	1600	48.125	2	8	2	×8	C	7.1 × 10 ⁴	7.1 × 10 ⁴	7.1 × 10 ⁴	19.7
	C ₁₃	12-08	1333	49.125	2	8	2	×8	C	3.9 × 10 ⁴	3.9 × 10 ⁴	3.9 × 10 ⁴	21.3
	C ₁₄₋₁₅	12-12	1333	49.125	2	8	2	×8	C	3.7 × 10 ⁴	2.1 × 10 ⁴	5.4 × 10 ⁴	21.3
	C ₁₆₋₁₈	12-20	1600	48.125	2	8	2	×8	C	3.5 × 10 ³	1.2 × 10 ³	7.0 × 10 ³	27.9
	C ₁₉	12-23	1600	48.125	2	8	2	×8	E	1.4 × 10 ⁵	1.4 × 10 ⁵	1.4 × 10 ⁵	18.0
	C ₂₀	12-24	1600	48.125	2	8	2	×8	C	6.5 × 10 ⁴	6.5 × 10 ⁴	6.5 × 10 ⁴	21.3
	C ₂₁	12-26	1600	48.125	2	8	2	×8	C	2.3 × 10 ⁴	2.3 × 10 ⁴	2.3 × 10 ⁴	24.6
	C ₂₂	12-32	1600	48.125	2	8	2	×8	C	1.7 × 10 ⁴	1.7 × 10 ⁴	1.7 × 10 ⁴	22.9
	C ₂₃₋₂₄	12-37	1600	48.125	2	8	2	×8	C	2.3 × 10 ⁴	1.1 × 10 ⁴	3.4 × 10 ⁴	18.0
	C ₂₅₋₃₀	12-41	1600	48.125	2	8	2	×8	C	2.0 × 10 ⁴	1.1 × 10 ⁴	3.2 × 10 ⁴	19.7
	C ₃₁	13-11	1600	48.125	2	8	2	×8	C	3.3 × 10 ⁵	3.3 × 10 ⁵	3.3 × 10 ⁵	14.7
	C ₃₂	13-35	1600	48.125	2	8	2	×8	C	3.7 × 10 ⁴	3.7 × 10 ⁴	3.7 × 10 ⁴	21.3

* We report the manufacture date marked on the chip packages, which is more accurate than other dates that can be gleaned from a module.

† We report timing constraints stored in the module's on-board ROM [33], which is read by the system BIOS to calibrate the memory controller.

‡ The maximum DRAM chip size supported by our testing platform is 2Gb.

§ We report DRAM die versions marked on the chip packages, which typically progress in the following manner: $\mathcal{M} \rightarrow \mathcal{A} \rightarrow \mathcal{B} \rightarrow \mathcal{C} \rightarrow \dots$.

Table 3. Sample population of 129 DDR3 DRAM modules, categorized by manufacturer and sorted by manufacture date

Fast Forward 6 Months

More Reviews... **Reviews from ISCA 2014**

PAPER WEAKNESSES

1) The disturbance error (a.k.a coupling or cross-talk noise induced error) is a known problem to the DRAM circuit community.

2) What you demonstrated in this paper is so called DRAM row hammering issue - you can even find a Youtube video showing this! - <http://www.youtube.com/watch?v=i3-gQSnBcdo>

2) The architectural contribution of this study is too insignificant.

PAPER WEAKNESSES

- Row Hammering appears to be well-known, and solutions have already been proposed by industry to address the issue.

- The paper only provides a qualitative analysis of solutions to the problem. A more robust evaluation is really needed to know whether the proposed solution is necessary.

Final RowHammer Reviews

Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors

Accepted



639kB

21 Nov 2013 10:53:11pm CST |

f039be2735313b39304ae1c6296523867a485610

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

	OveMer	Nov	WriQua	RevConAnd
Review #41A	8	4	5	3
Review #41B	7	4	4	3
Review #41C	6	4	4	3
Review #41D	2	2	5	4
Review #41E	3	2	3	3
Review #41F	7	4	4	3

RowHammer: Hindsight & Impact (I)

Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors

Abstract. Memory isolation is a key property of a reliable and secure computing system — an access to one memory address should not have unintended side effects on data stored in other addresses. However, as DRAM process technology

Project Zero

Flipping Bits in Memory Without Accessing Them:
An Experimental Study of DRAM Disturbance Errors
(Kim et al., ISCA 2014)

News and updates from the Project Zero team at Google

Exploiting the DRAM rowhammer bug to
gain kernel privileges (Seaborn, 2015)

Monday, March 9, 2015

Exploiting the DRAM rowhammer bug to gain kernel privileges

RowHammer: Hindsight & Impact (II)

- Onur Mutlu and Jeremie Kim,
"RowHammer: A Retrospective"
IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD) Special Issue on Top Picks in Hardware and Embedded Security, 2019.
[[Preliminary arXiv version](#)]

RowHammer: A Retrospective

Onur Mutlu^{§‡} Jeremie S. Kim^{‡§}
§ETH Zürich ‡Carnegie Mellon University

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

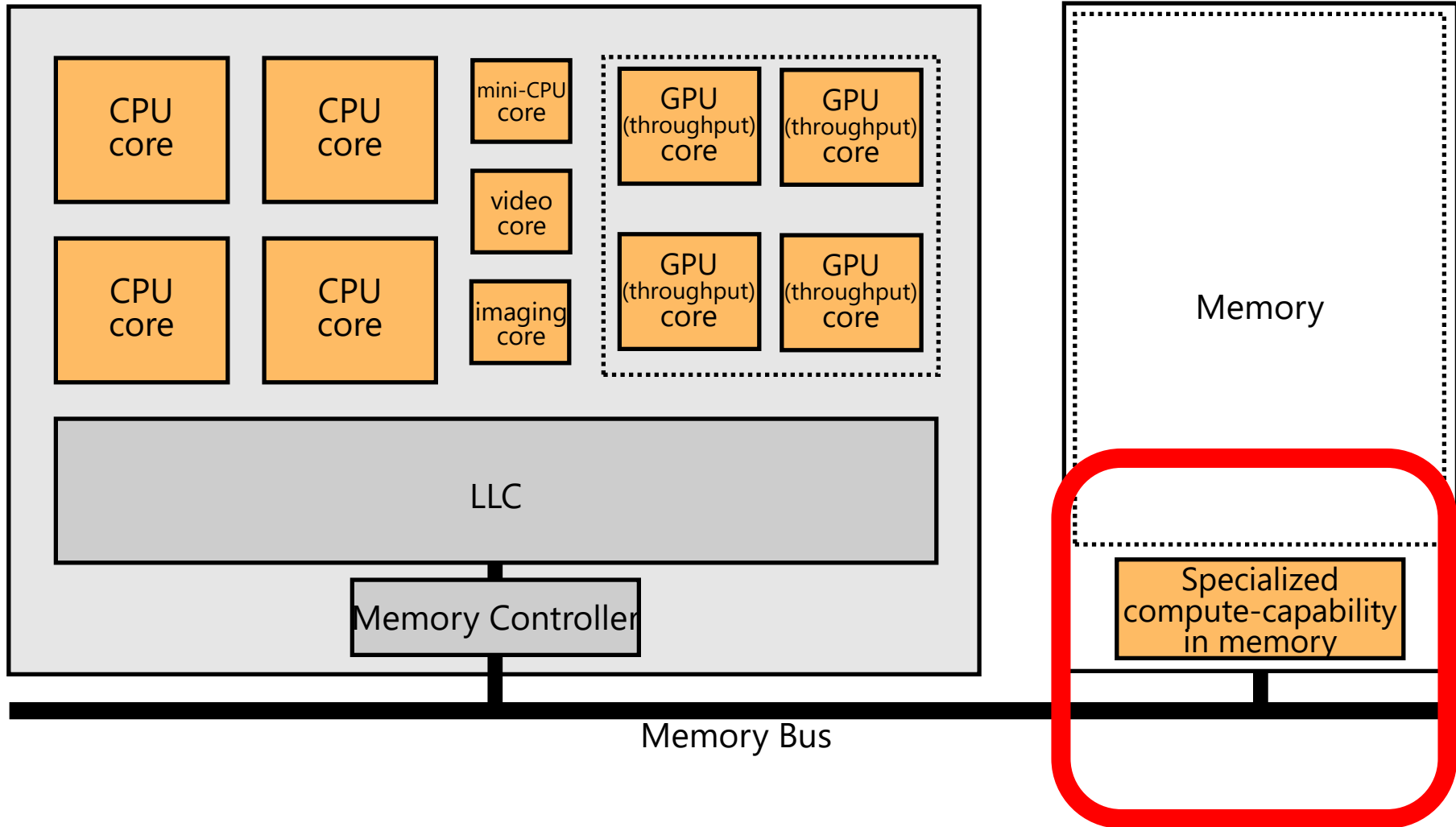
The quality of your work
defines your impact

Sub-Agenda: In-Memory Computation

- Major Trends Affecting Main Memory
- The Need for Intelligent Memory Controllers
 - Bottom Up: Push from Circuits and Devices
 - Top Down: Pull from Systems and Applications
- Processing in Memory: Two Directions
 - Minimally Changing Memory Chips
 - Exploiting 3D-Stacked Memory
- How to Enable Adoption of Processing in Memory
- Conclusion

We Need to Think Differently
from the Past Approaches

Memory as an Accelerator



Memory similar to a "conventional" accelerator

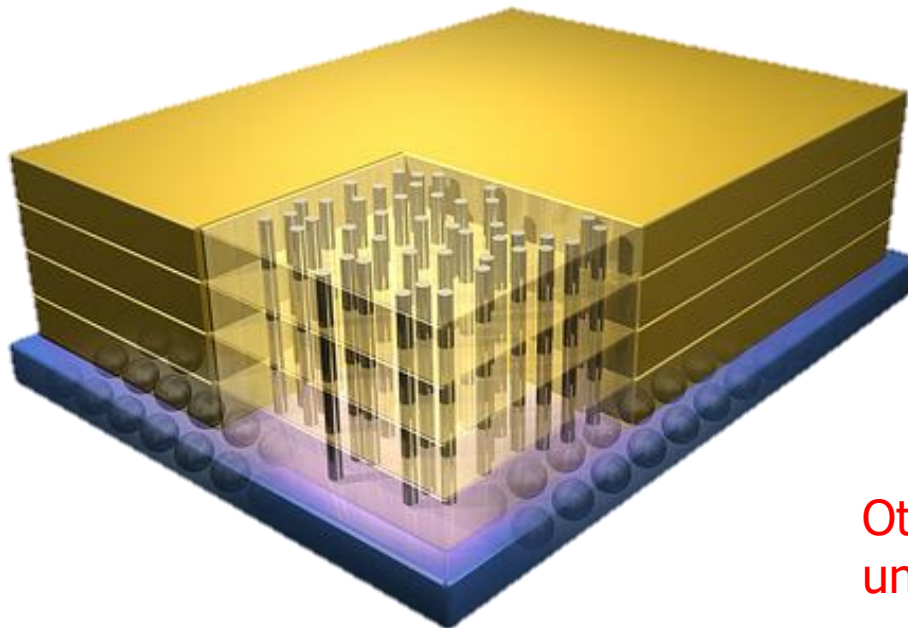
Processing in Memory: Two Approaches

1. Minimally changing memory chips
2. Exploiting 3D-stacked memory

Opportunity: 3D-Stacked Logic+Memory



Hybrid Memory Cube
C O N S O R T I U M



Memory

Logic

Other "True 3D" technologies
under development

DRAM Landscape (circa 2015)

<i>Segment</i>	<i>DRAM Standards & Architectures</i>
Commodity	DDR3 (2007) [14]; DDR4 (2012) [18]
Low-Power	LPDDR3 (2012) [17]; LPDDR4 (2014) [20]
Graphics	GDDR5 (2009) [15]
Performance	eDRAM [28], [32]; RLDram3 (2011) [29]
3D-Stacked	WIO (2011) [16]; WIO2 (2014) [21]; MCDRAM (2015) [13]; HBM (2013) [19]; HMC1.0 (2013) [10]; HMC1.1 (2014) [11]
Academic	SBA/SSA (2010) [38]; Staged Reads (2012) [8]; RAIDR (2012) [27]; SALP (2012) [24]; TL-DRAM (2013) [26]; RowClone (2013) [37]; Half-DRAM (2014) [39]; Row-Buffer Decoupling (2014) [33]; SARP (2014) [6]; AL-DRAM (2015) [25]

Table 1. Landscape of DRAM-based memory

Kim+, “[Ramulator: A Flexible and Extensible DRAM Simulator](#)”, IEEE CAL 2015.

Several Questions in 3D-Stacked PIM

- What are the performance and energy benefits of using 3D-stacked memory as a coarse-grained accelerator?
 - By changing the entire system
 - By performing simple function offloading

- What is the minimal processing-in-memory support we can provide?
 - With minimal changes to system and programming

Another Example: In-Memory Graph Processing

- Large graphs are everywhere (circa 2015)



36 Million
Wikipedia Pages



1.4 Billion
Facebook Users

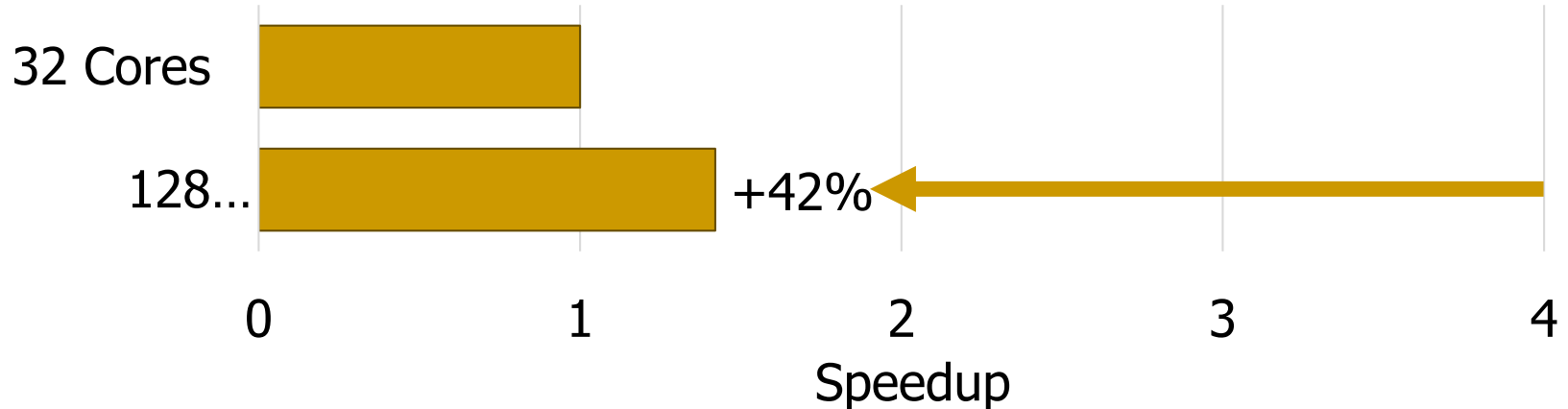


300 Million
Twitter Users



30 Billion
Instagram Photos

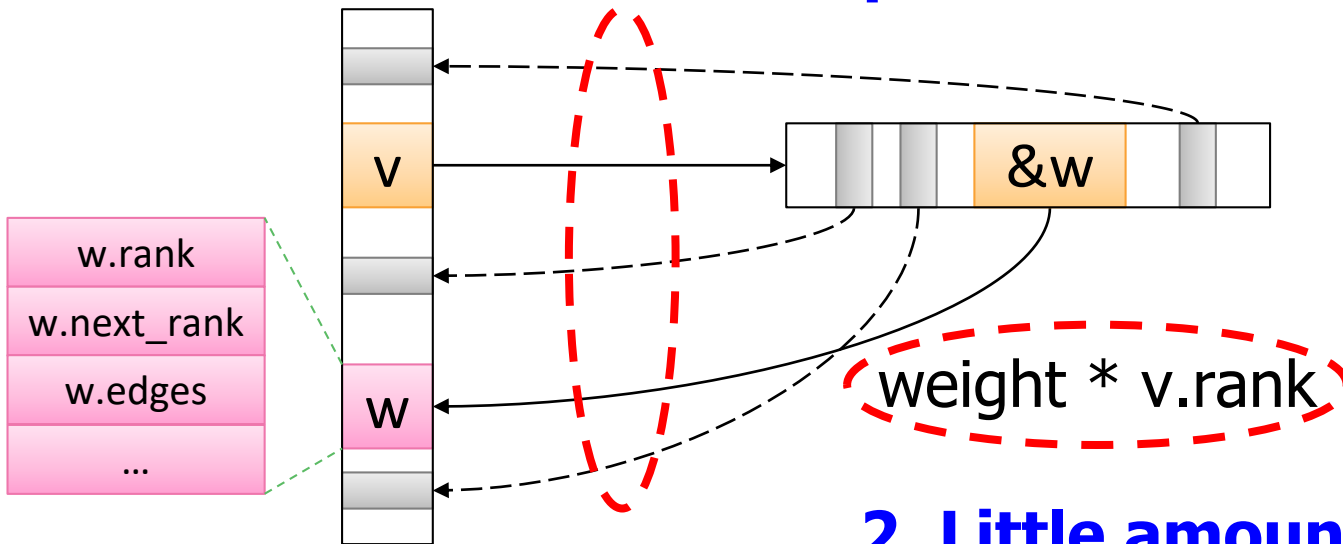
- Scalable large-scale graph processing is challenging



Key Bottlenecks in Graph Processing

```
for (v: graph.vertices) {  
  for (w: v.successors) {  
    w.next_rank += weight * v.rank;  
  }  
}
```

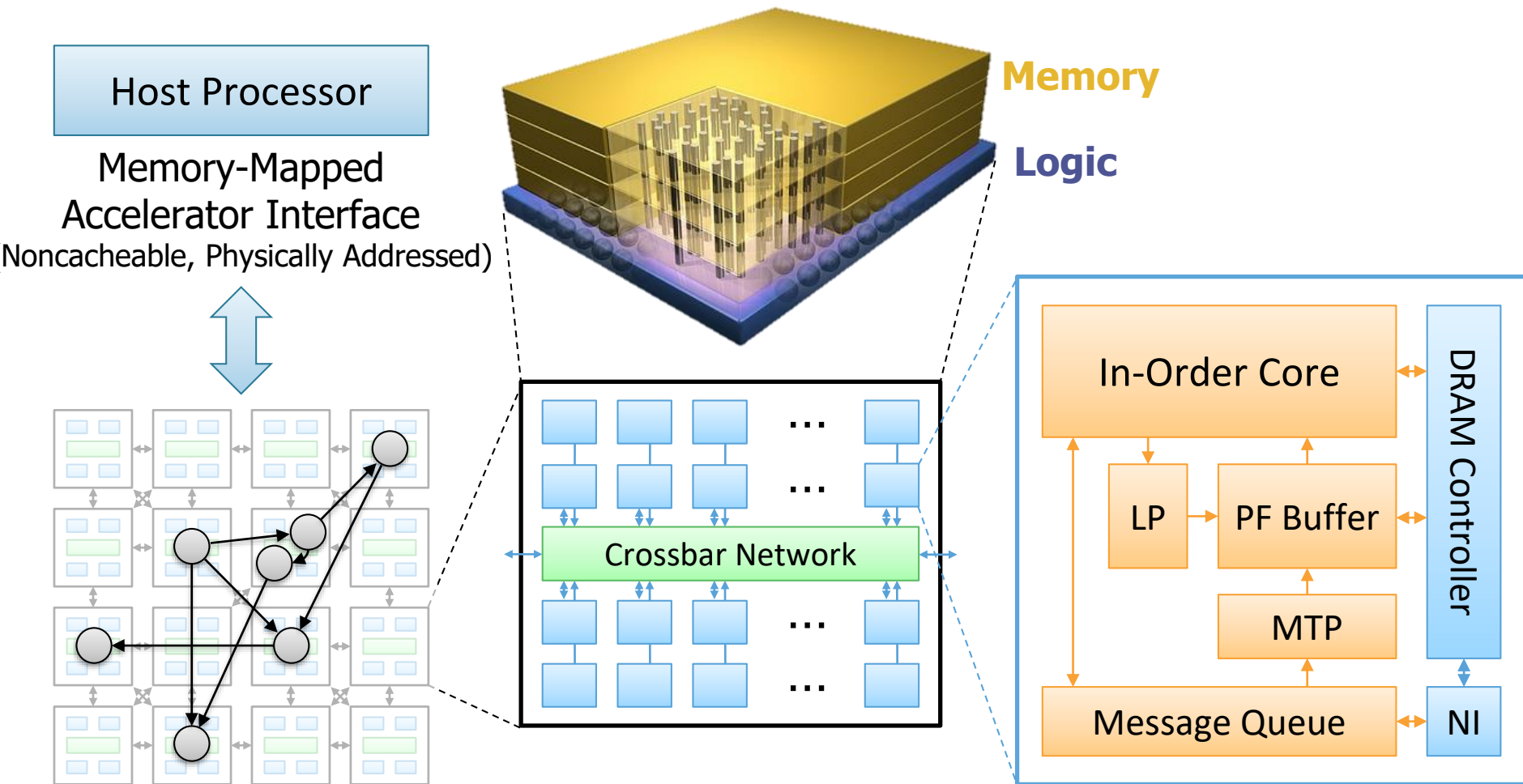
1. Frequent random memory accesses



2. Little amount of computation

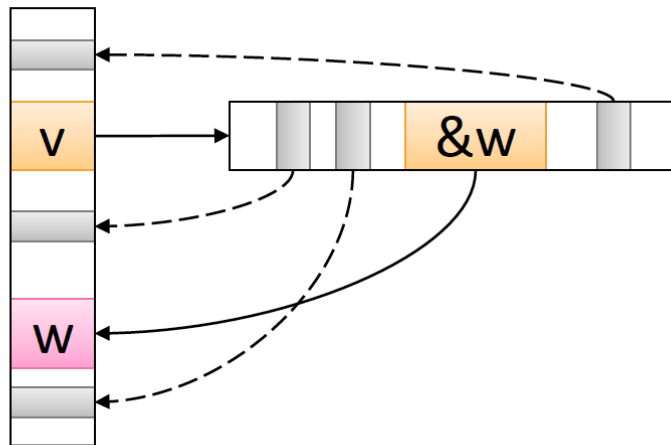
Tesseract System for Graph Processing

Interconnected set of 3D-stacked memory+logic chips with simple cores



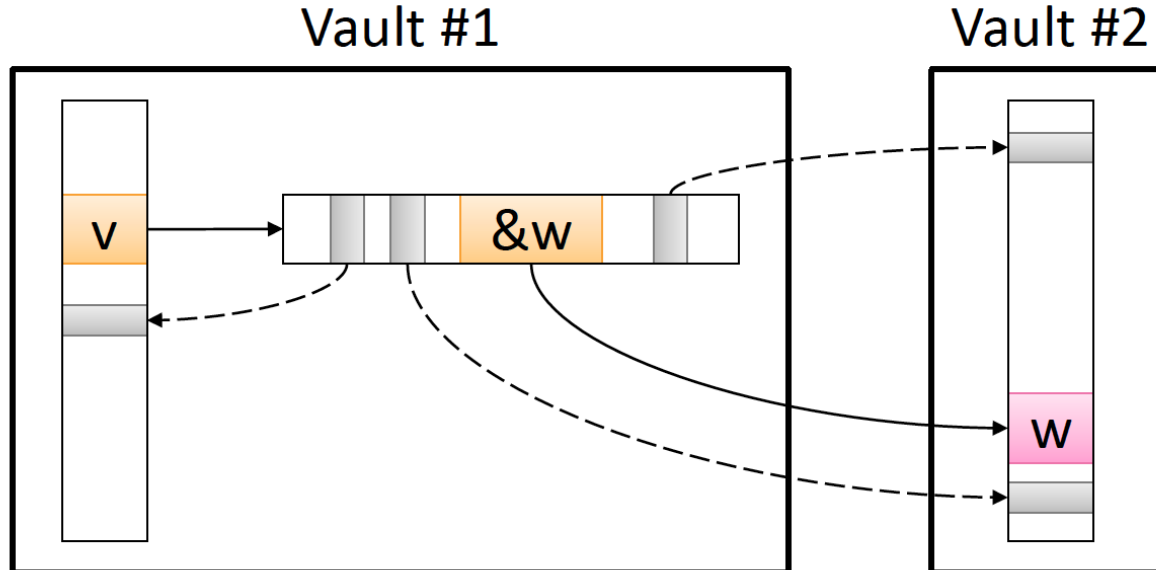
Communications In Tesseract (I)

```
for (v: graph.vertices) {  
  for (w: v.successors) {  
    w.next_rank += weight * v.rank;  
  }  
}
```



Communications In Tesseract (II)

```
for (v: graph.vertices) {  
  for (w: v.successors) {  
    w.next_rank += weight * v.rank;  
  }  
}
```

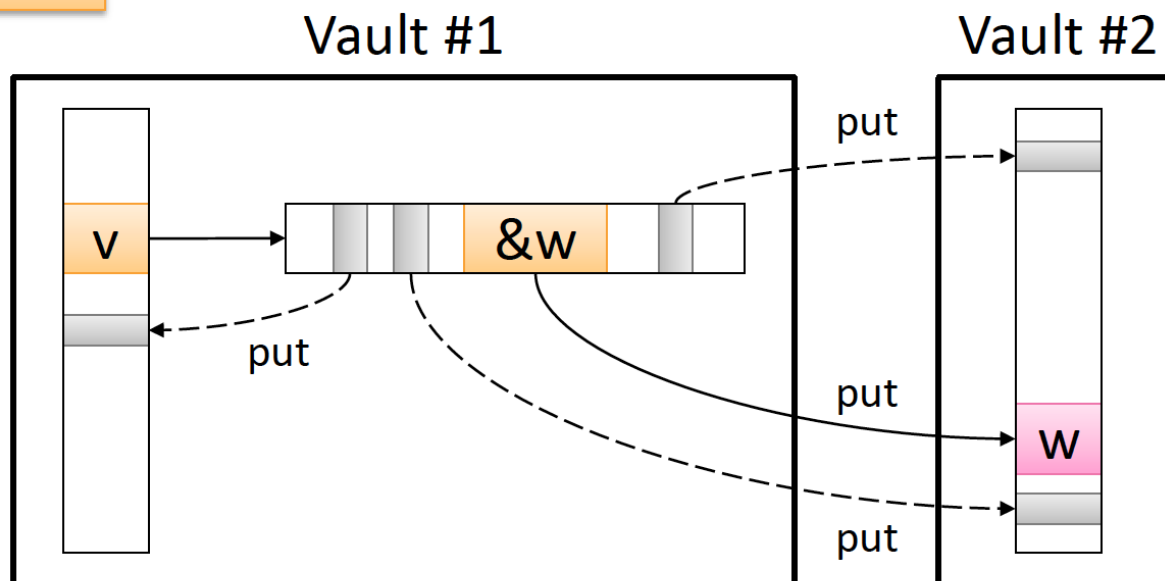


Communications In Tesseract (III)

```
for (v: graph.vertices) {  
  for (w: v.successors) {  
    put(w.id, function() { w.next_rank += weight * v.rank; });  
  }  
}  
barrier();
```

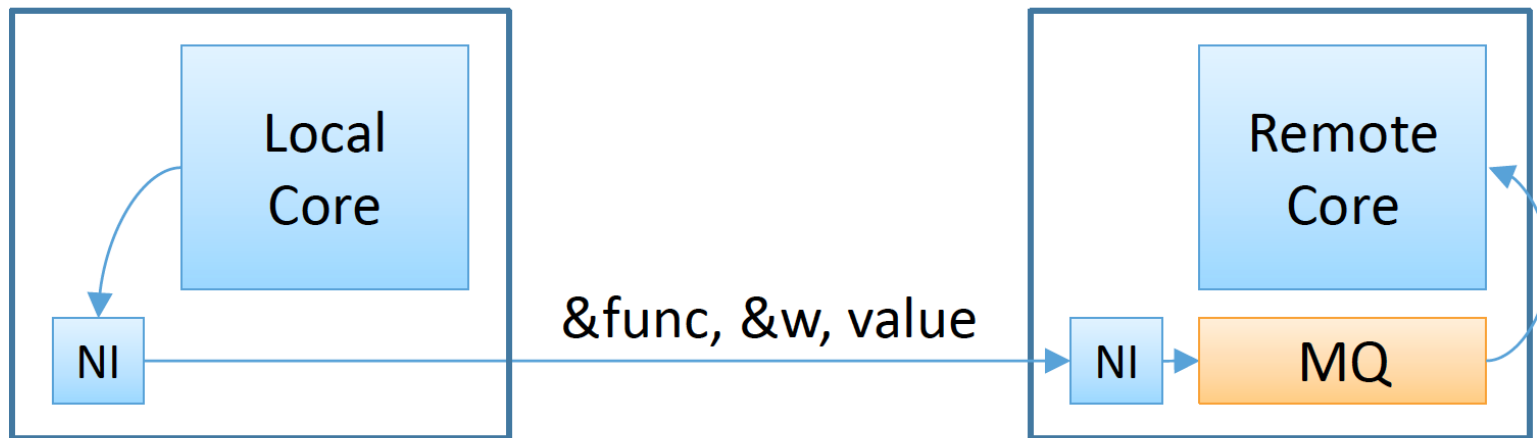
Non-blocking Remote Function Call

Can be **delayed**
until the nearest barrier



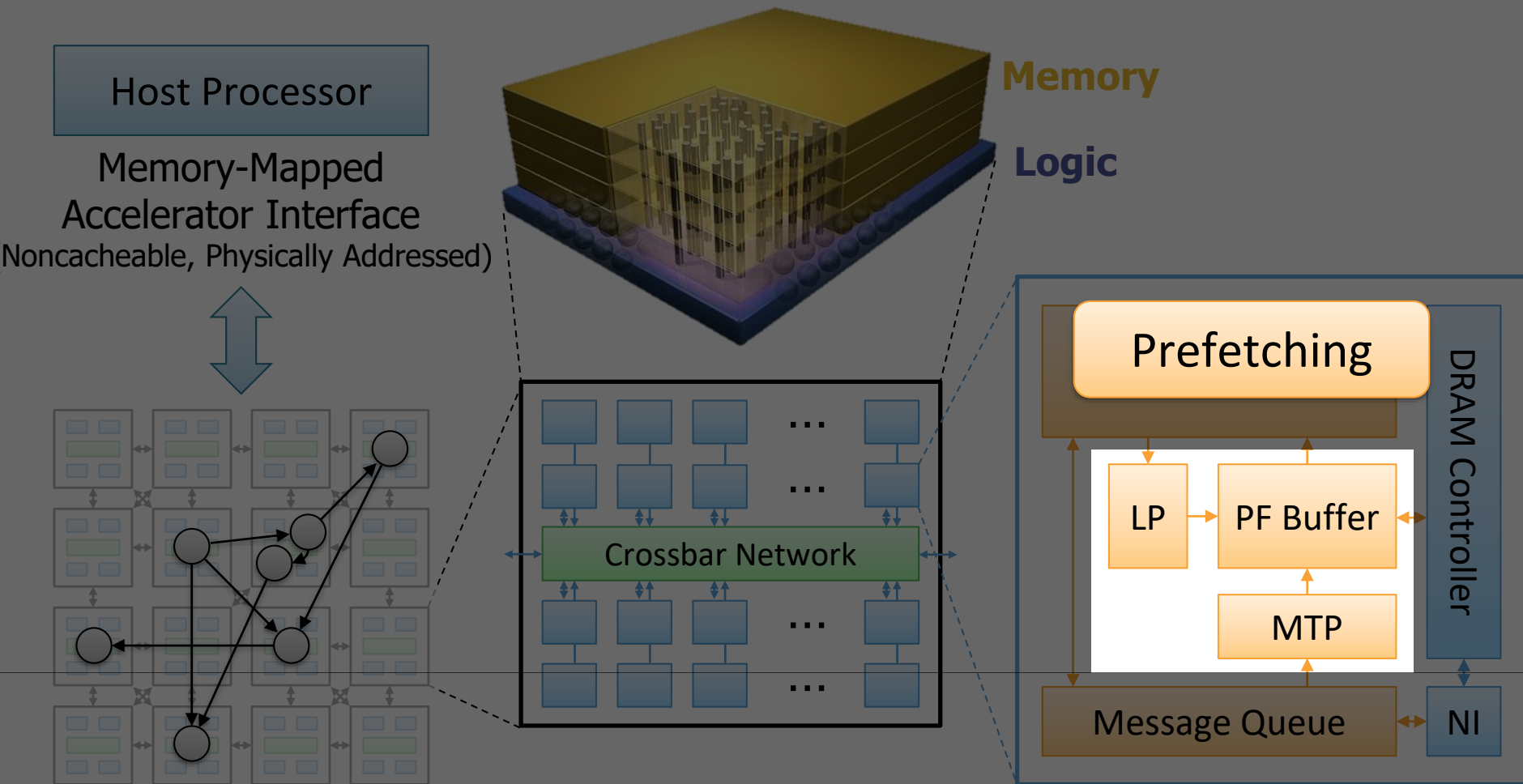
Remote Function Call (Non-Blocking)

1. Send function address & args to the remote core
2. Store the incoming message to the message queue
3. Flush the message queue when it is full or a synchronization barrier is reached



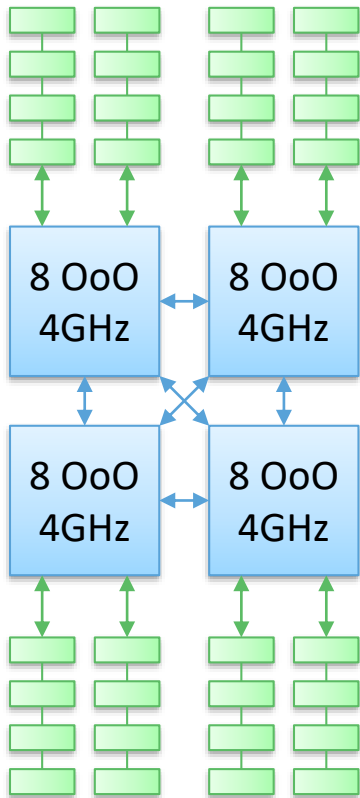
```
put(w.id, function() { w.next_rank += value; })
```

Tesseract System for Graph Processing



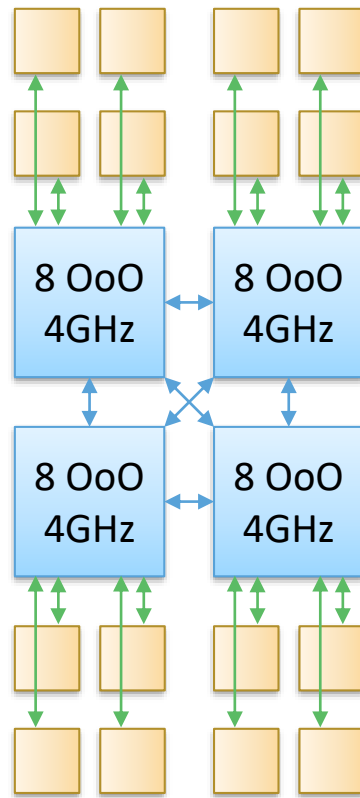
Evaluated Systems

DDR3-OoO



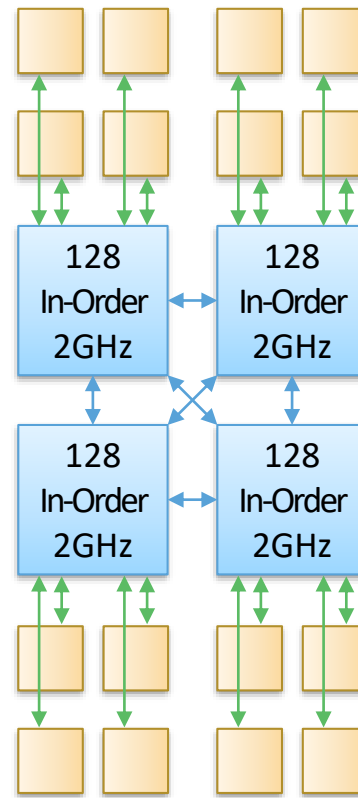
102.4GB/s

HMC-OoO



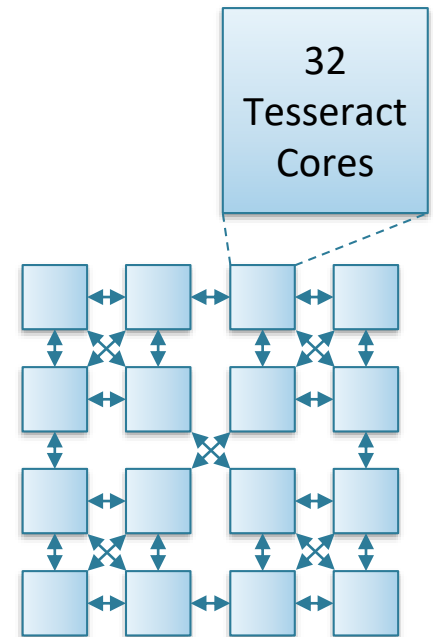
640GB/s

HMC-MC



640GB/s

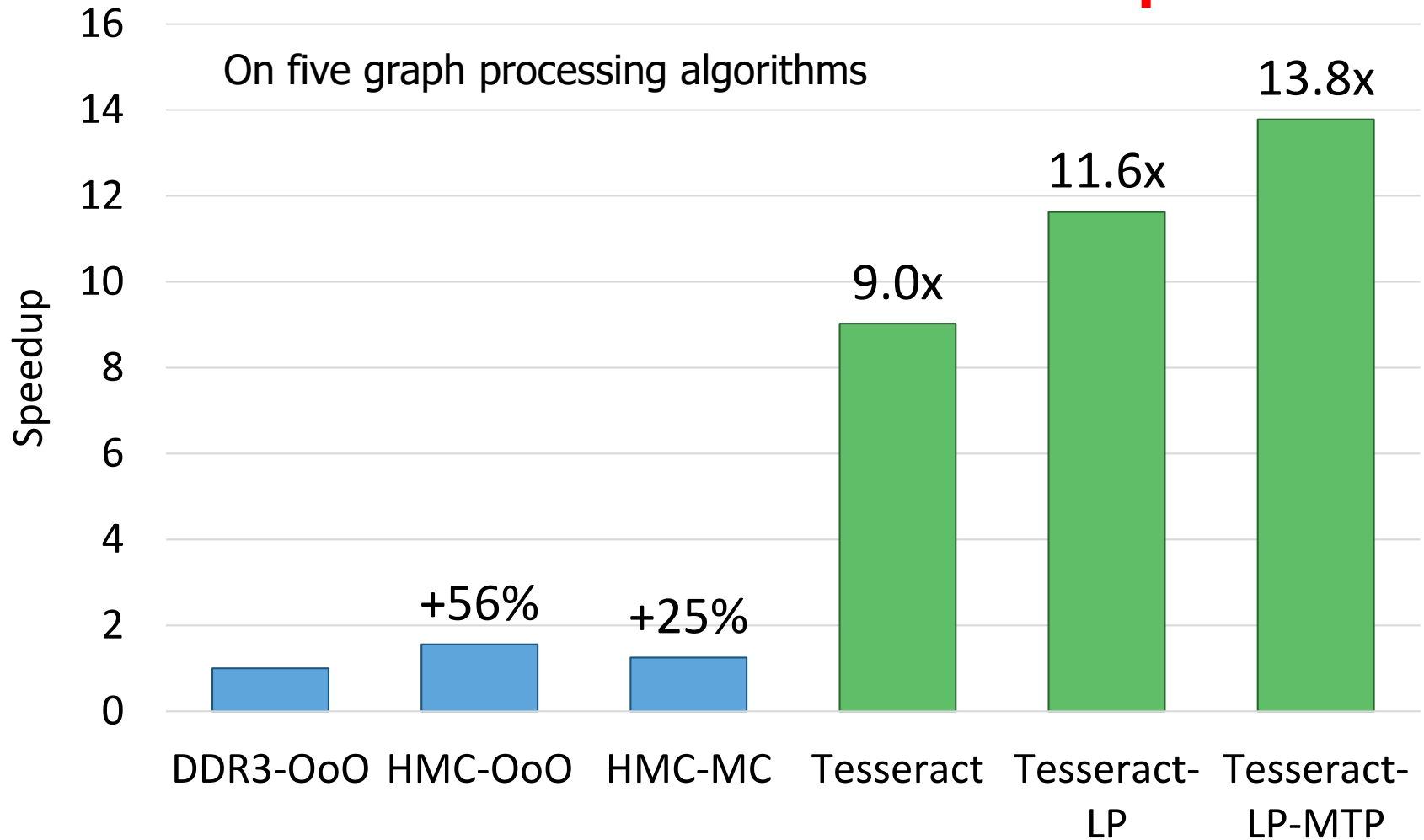
Tesseract



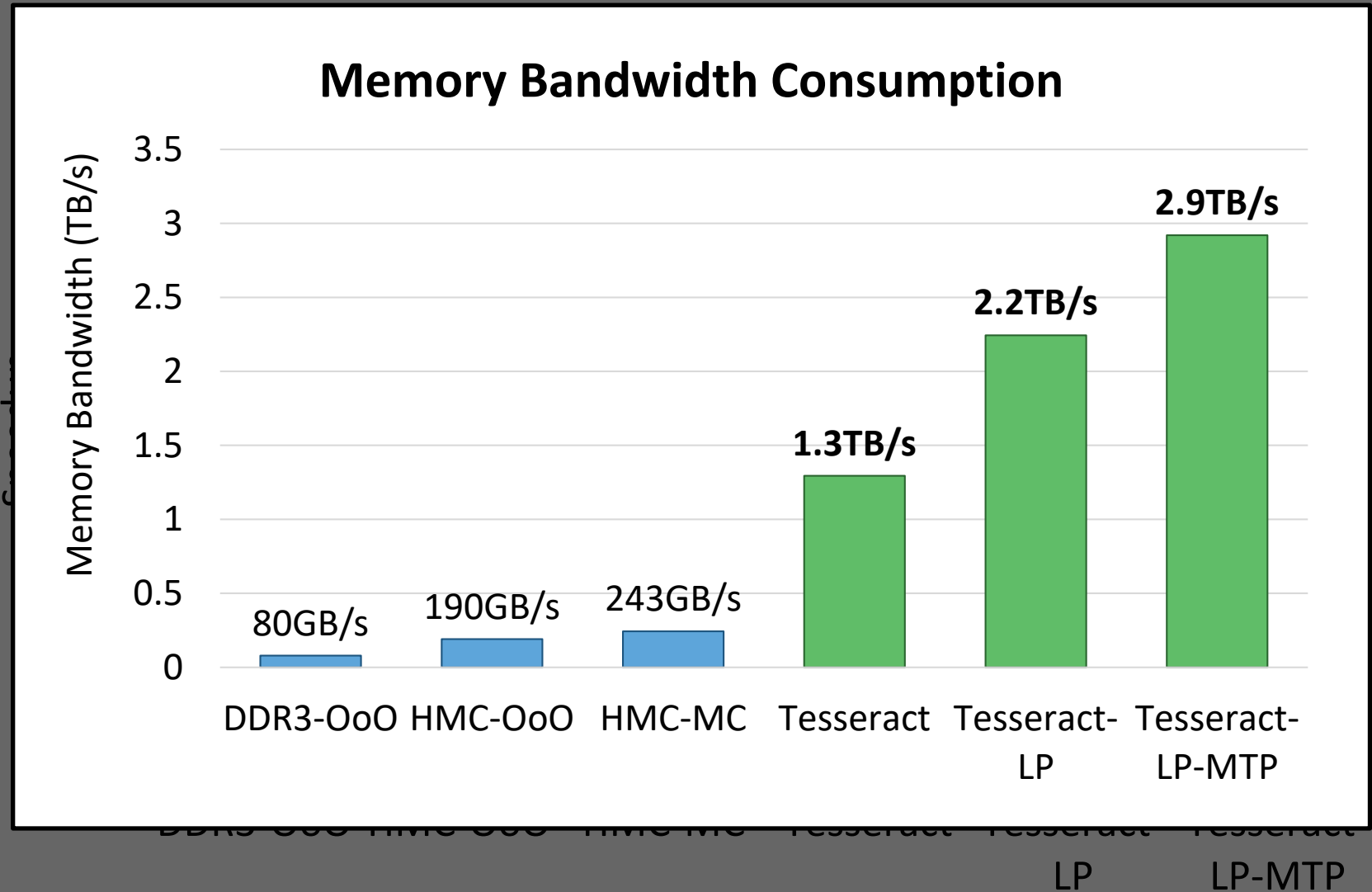
8TB/s

Tesseract Graph Processing Performance

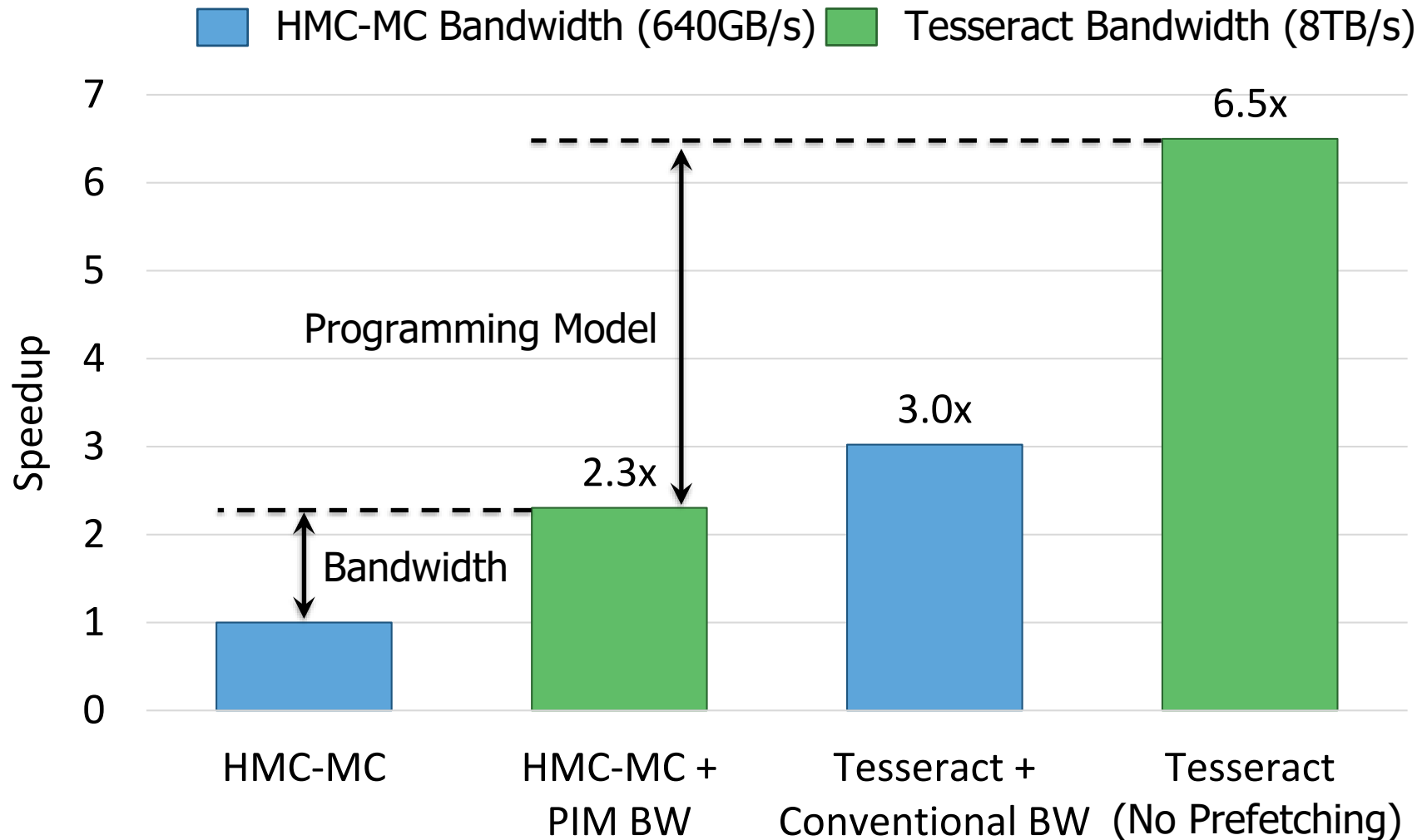
>13X Performance Improvement



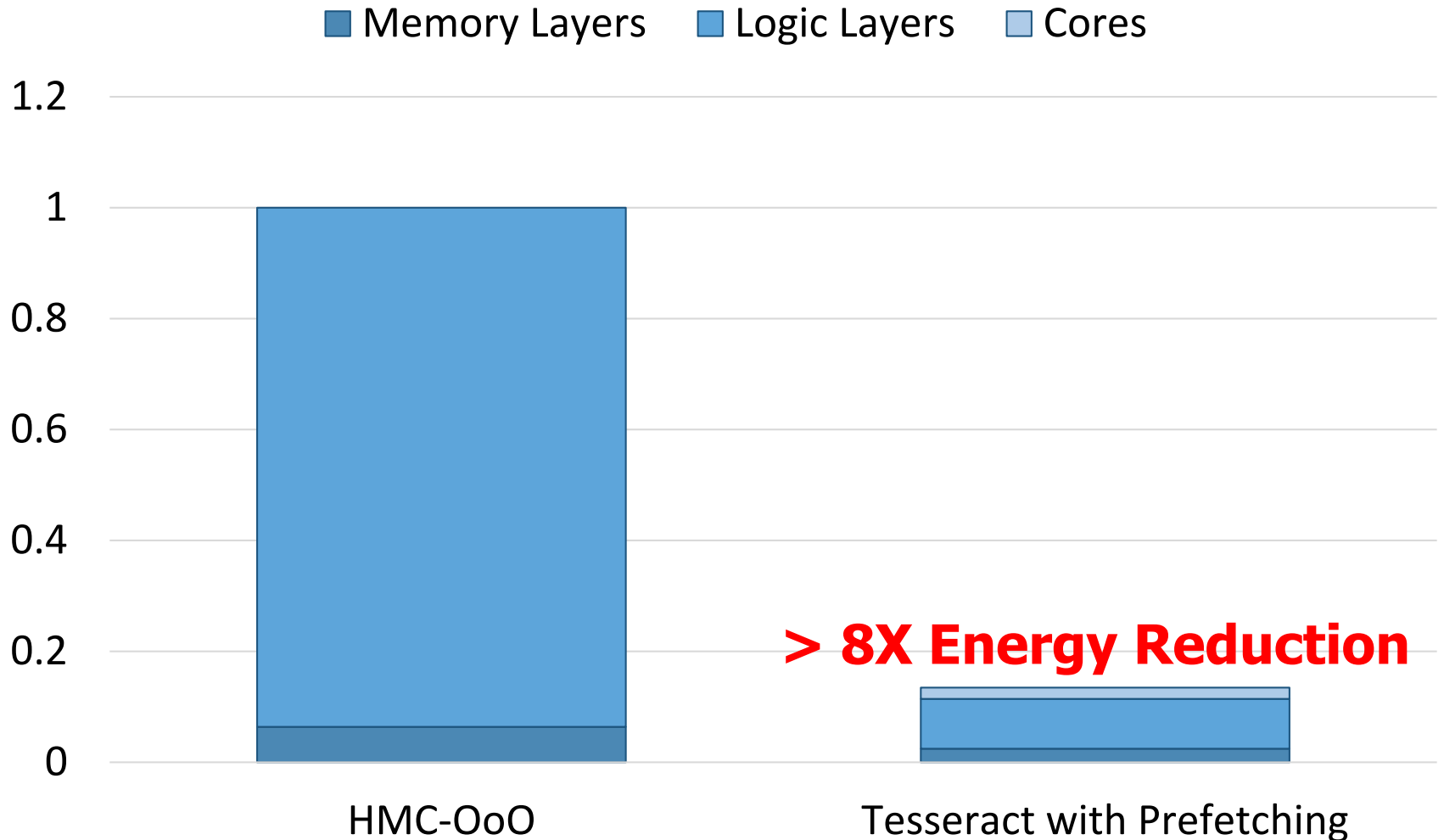
Tesseract Graph Processing Performance



Effect of Bandwidth & Programming Model



Tesseract Graph Processing System Energy



Tesseract: Advantages & Disadvantages

■ Advantages

- + Specialized graph processing accelerator using PIM
- + Large system performance and energy benefits
- + Takes advantage of 3D stacking for an important workload
- + More general than just graph processing

■ Disadvantages

- Changes a lot in the system
 - New programming model
 - Specialized Tesseract cores for graph processing
- Cost
- Scalability limited by off-chip links or graph partitioning

More on Tesseract

- Junwhan Ahn, Sungpack Hong, Sungjoo Yoo, Onur Mutlu, and Kiyoungh Choi,
"A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing"
Proceedings of the 42nd International Symposium on Computer Architecture (ISCA), Portland, OR, June 2015.
[[Slides \(pdf\)](#)] [[Lightning Session Slides \(pdf\)](#)]

A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing

Junwhan Ahn Sungpack Hong[§] Sungjoo Yoo Onur Mutlu[†] Kiyoungh Choi
junwhan@snu.ac.kr, sungpack.hong@oracle.com, sungjoo.yoo@gmail.com, onur@cmu.edu, kchoi@snu.ac.kr

Seoul National University

[§]Oracle Labs

[†]Carnegie Mellon University

Computer Architecture

Lecture 7: Computation in Memory II

Prof. Onur Mutlu

ETH Zürich

Fall 2019

10 October 2019