CSC 2224: Parallel Computer Architecture and Programming
Advanced Memory

Prof. Gennady Pekhimenko
University of Toronto

Fall 2019

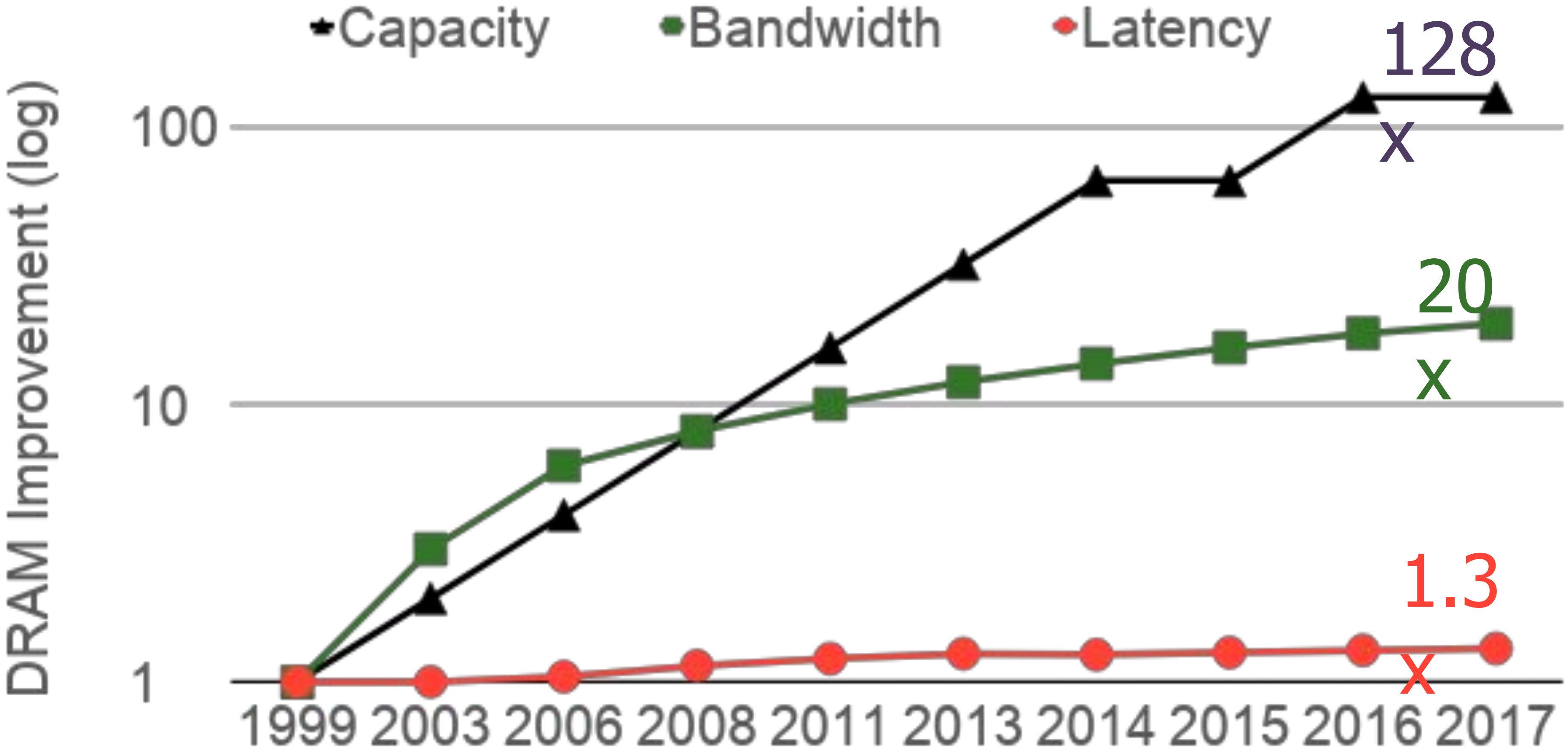
The content of this lecture is adapted from the slides of
Vivek Seshadri, Yoongu Kim,
and lectures of Onur Mutlu @ ETH and CMU

Review #5

Flipping Bits in Memory Without Accessing Them

Yoongu Kim et al., *ISCA 2014*

Review: Memory Latency Lags Behind



Memory latency remains almost constant

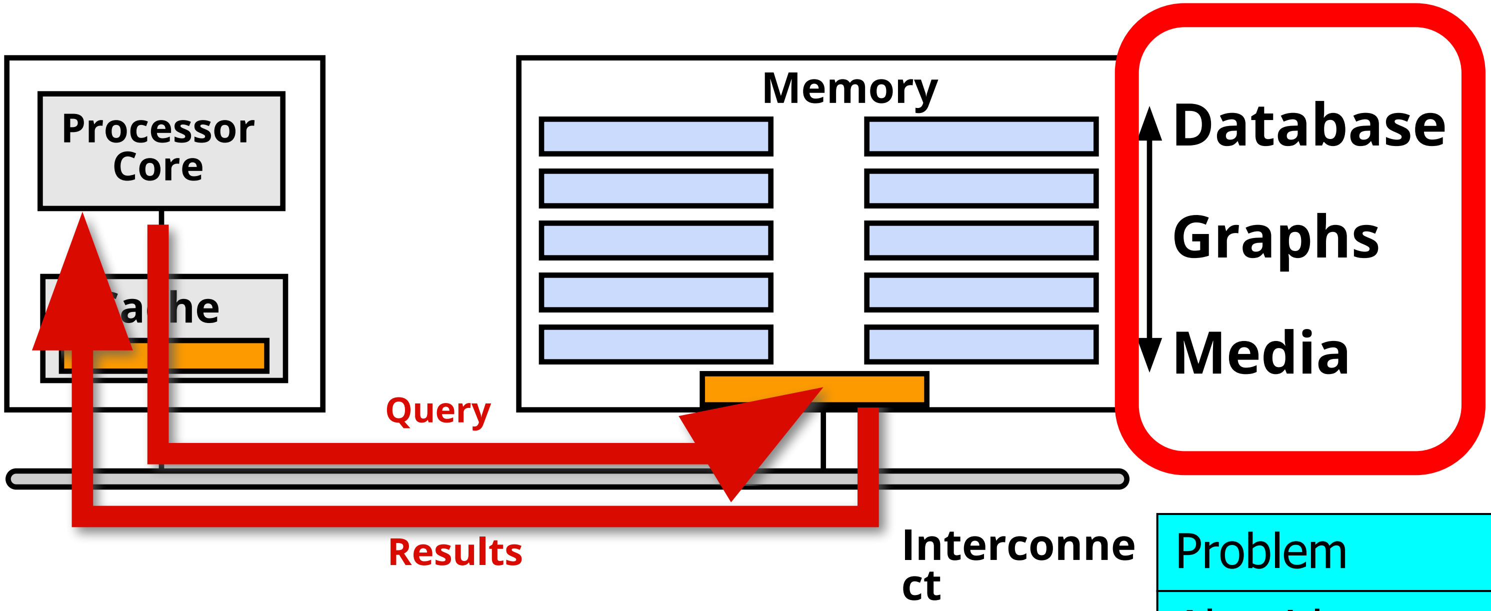
We Need A Paradigm Shift To …

- Enable computation with minimal data movement

- Compute where it makes sense (where data resides)

- Make computing architectures more data-centric

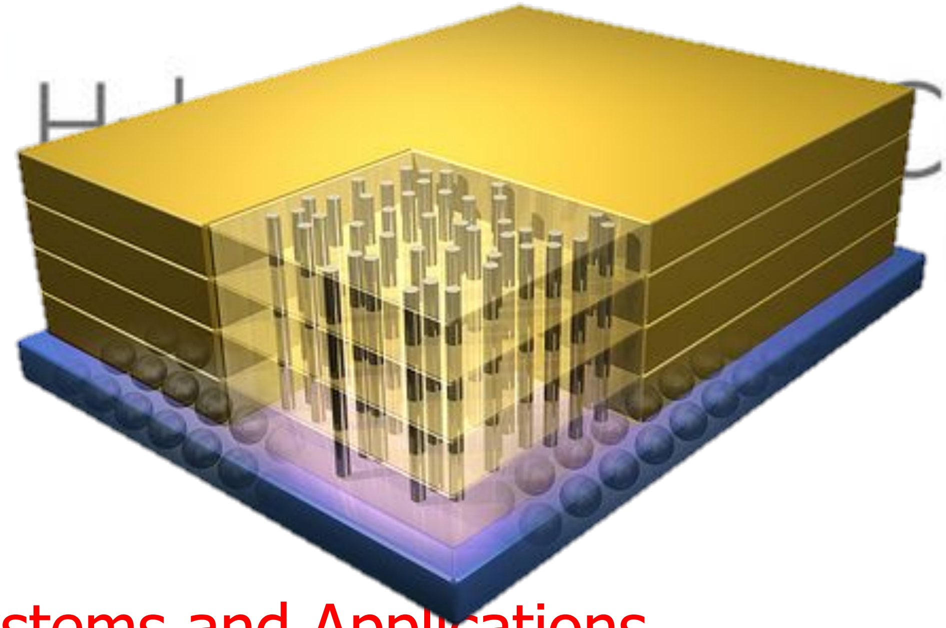
Processing Inside Memory


Processor Core
Cache
Memory
Database
Graphs
Media
Query
Results
Interconnect

- Many questions … How do we design the:
  - compute-capable memory & controllers?
  - processor chip?
  - software and hardware interfaces?
  - system software and languages?
  - algorithms?

Problem
Algorithm
Program/Language
System Software
SW/HW Interface
Micro-architecture
Logic
Devices
Electrons

Why In-Memory Computation Today?



Dally, HiPEAC 2

- Pull from Systems and Applications
  - Data access is a major system and application bottleneck
  - Systems are energy limited
  - Data movement much more energy-hungry than computation

Two Approaches to In-Memory Processing

- 1. Minimally change DRAM to enable simple yet powerful computation primitives
  - 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)

- 2. Exploit the control logic in 3D-stacked memory to enable more comprehensive computation near memory
  - PIM-Enabled Instructions: A Low-Overhead, Locality-Aware Processing-in-Memory Architecture (Ahn et al., ISCA 2015)
  - A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing (Ahn et al., ISCA 2015)
  - Accelerating Pointer Chasing in 3D-Stacked Memory: Challenges, Mechanisms, Evaluation (Hsieh et al., ICCD 2016)

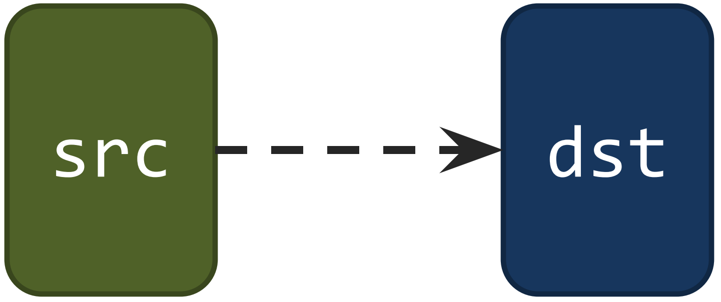
Approach 1: Minimally Changing DRAM

- DRAM has great capability to perform bulk data movement and computation internally with small changes
  - Can exploit internal bandwidth 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)

Starting Simple: Data Copy and Initialization

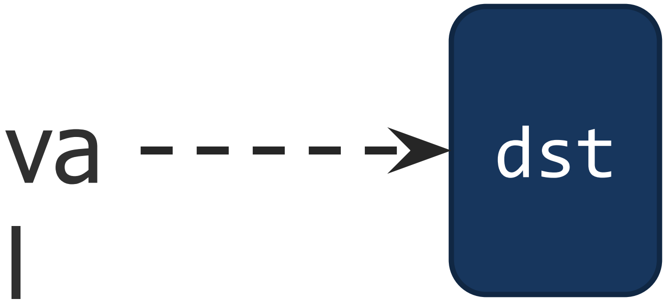
Bulk Data Copy



src

dst

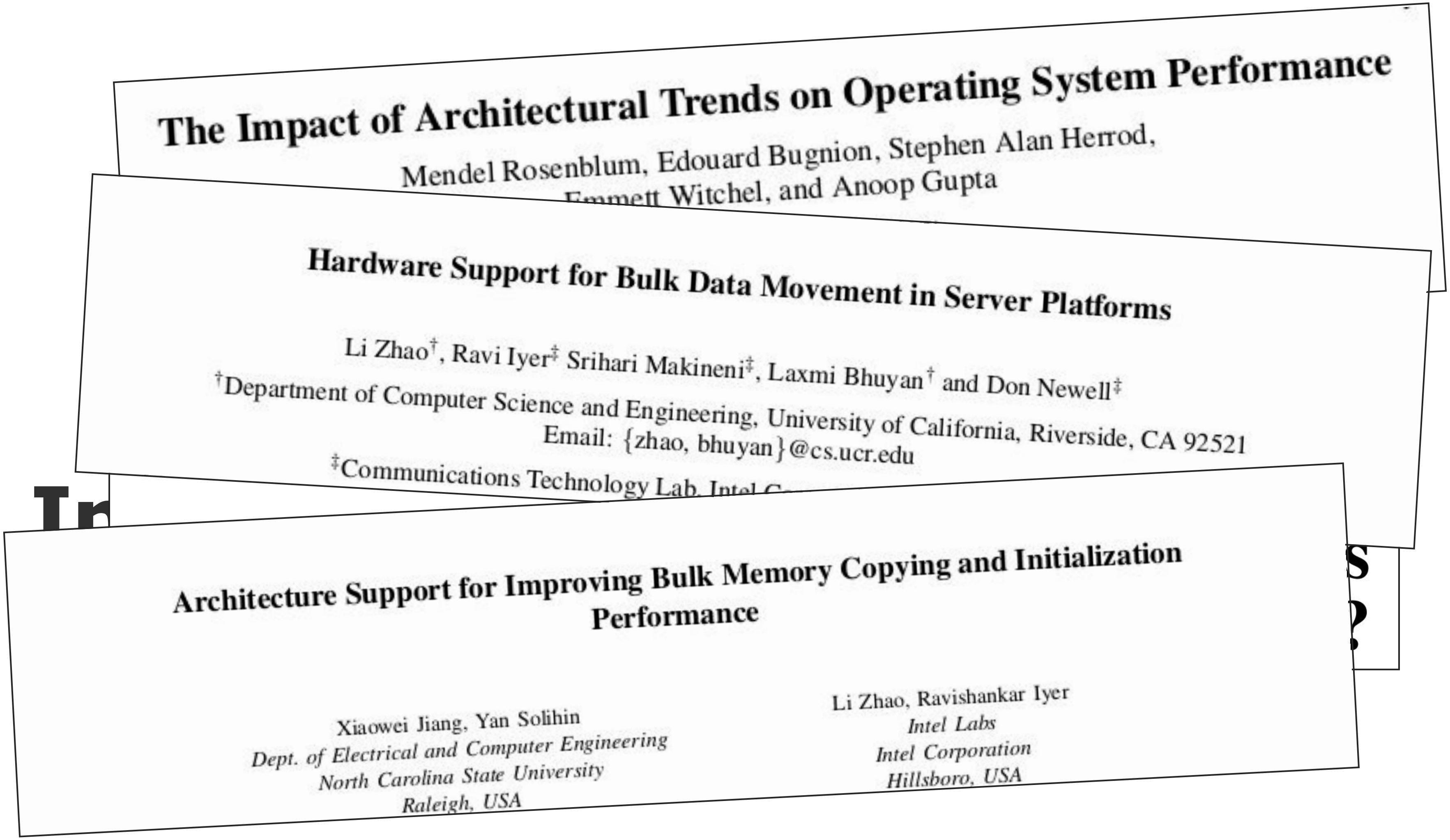
Bulk Data Initialization



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

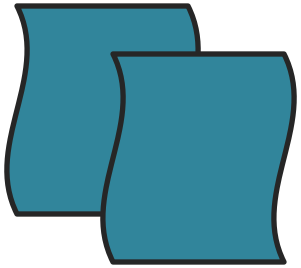
Architecture Support for Improving Bulk Memory Copying and Initialization Performance

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

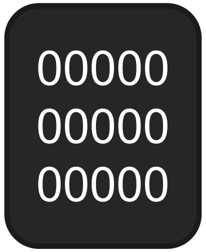
Li Zhao, Ravishankar Iyer
Intel Labs
Intel Corporation
Hillsboro, USA

Bulk Data Copy and Initialization

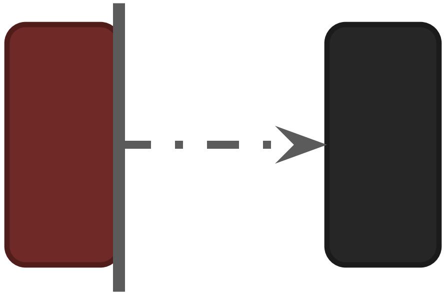
memmove & *memcpy:* 5% cycles in Google's datacenter [Kanev+ ISCA'1



Forking

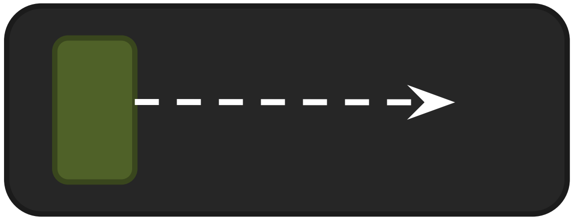


Zero initialization (e.g., security)



Checkpointing

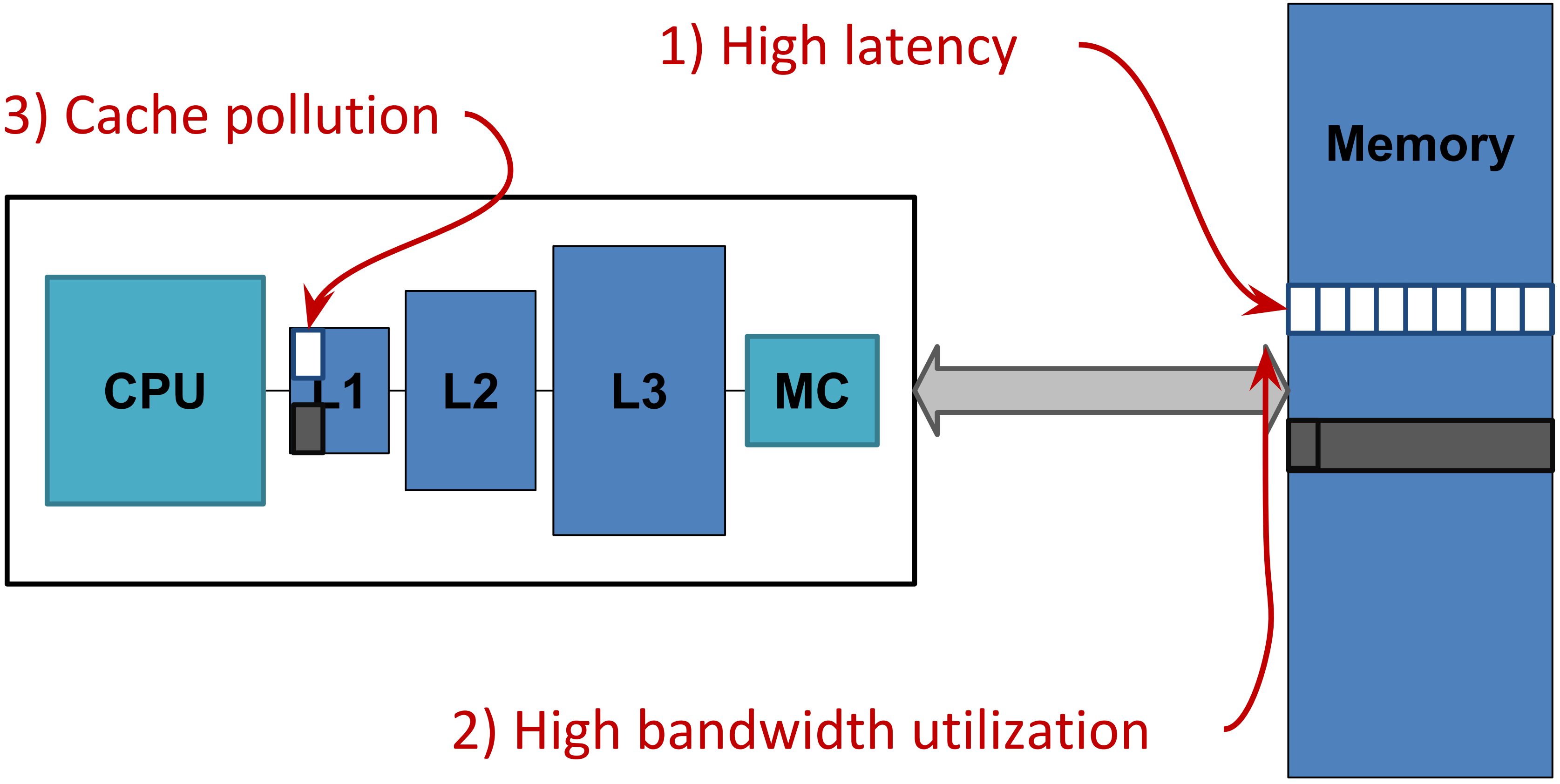
VM Cloning Deduplication



Page Migration

. . .
Many more

Today's Systems: Bulk Data Copy


CPU
L1
L2
L3
MC
Memory

1) High latency

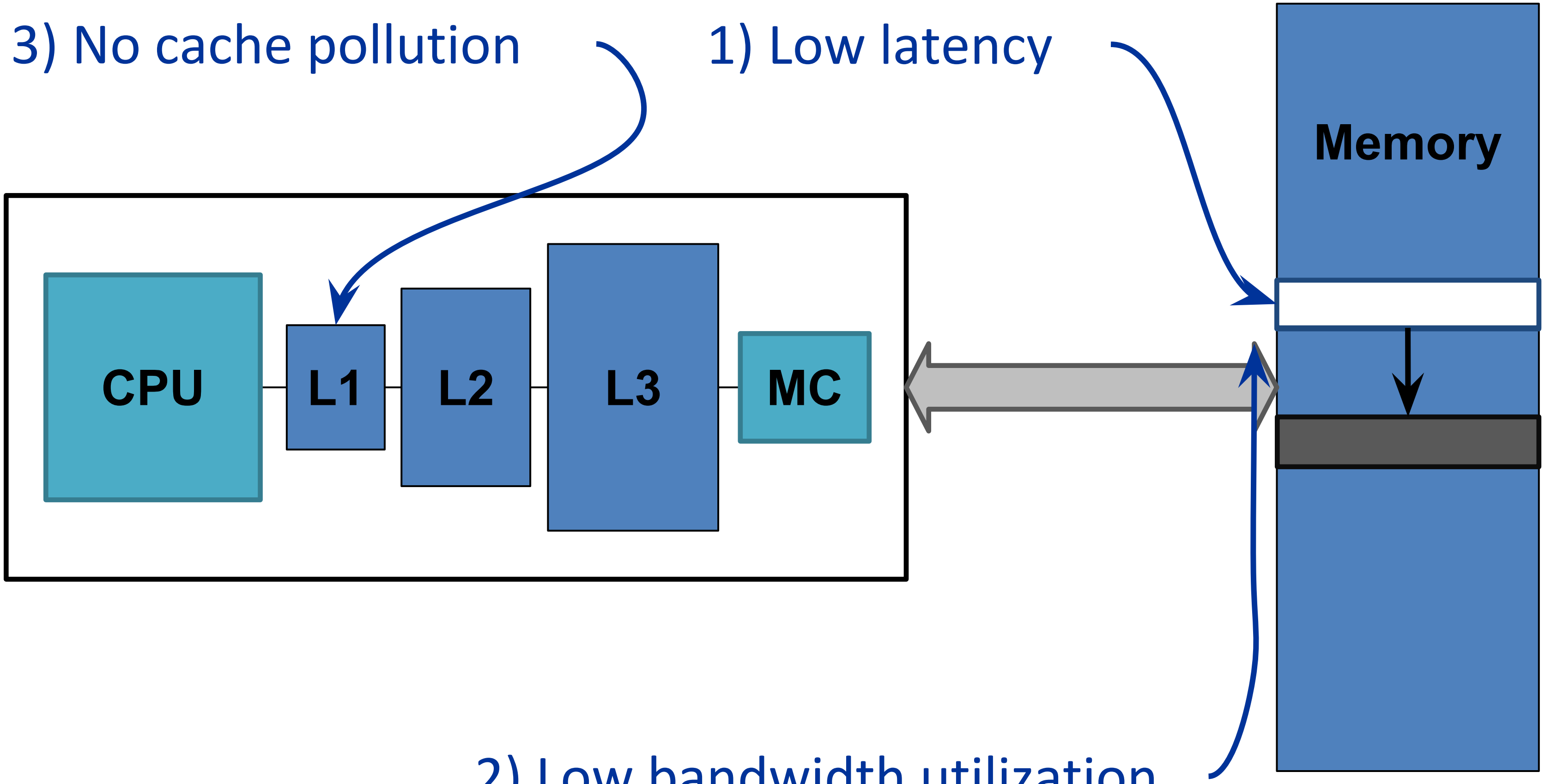
3) Cache pollution

2) High bandwidth utilization

4) Unwanted data movement

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

Future Systems: In-Memory Copy


CPU
L1
L2
L3
MC
Memory

3) No cache pollution

1) Low latency

2) Low bandwidth utilization

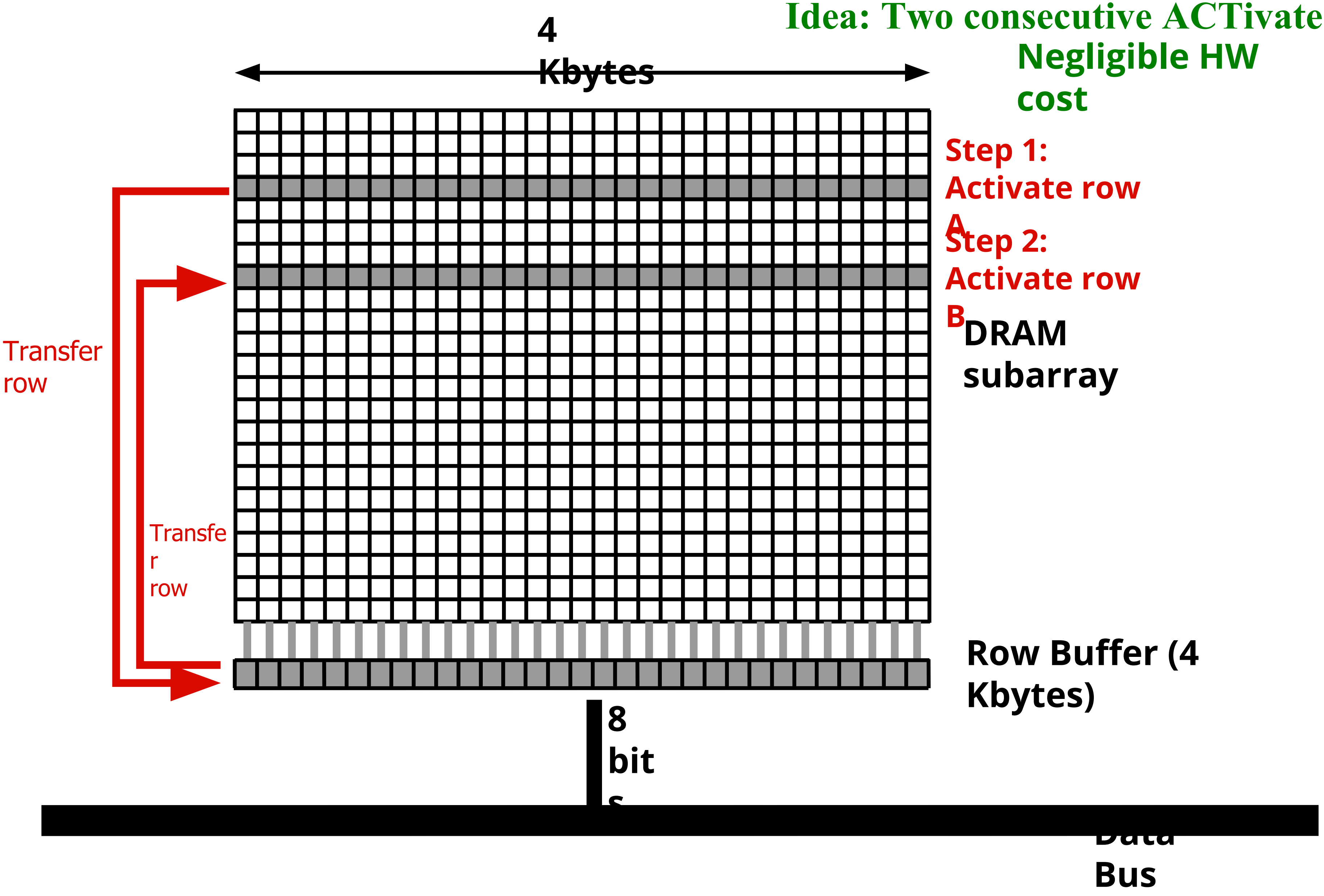
4) No unwanted data movement

1046ns, 3.6uJ 90ns, 0.04uJ

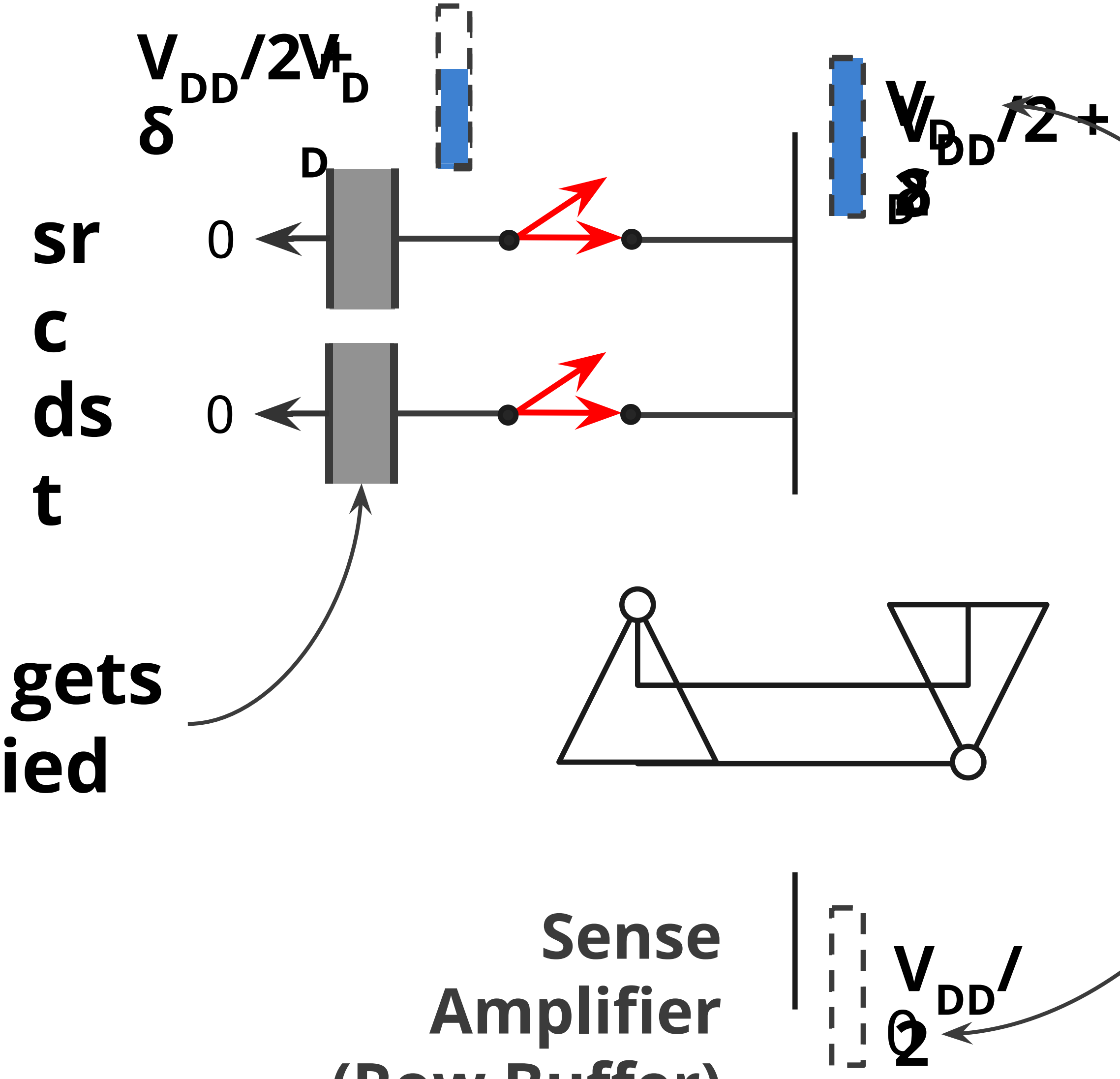
RowClone: In-DRAM Row Copy

Idea: Two consecutive ACTivates

Negligible HW cost


4 Kbytes
Step 1: Activate row A
Step 2: Activate row B
Transfer row
Transfer row
DRAM subarray
Row Buffer (4 Kbytes)
8 bits
Data Bus

RowClone: Intra-Subarray

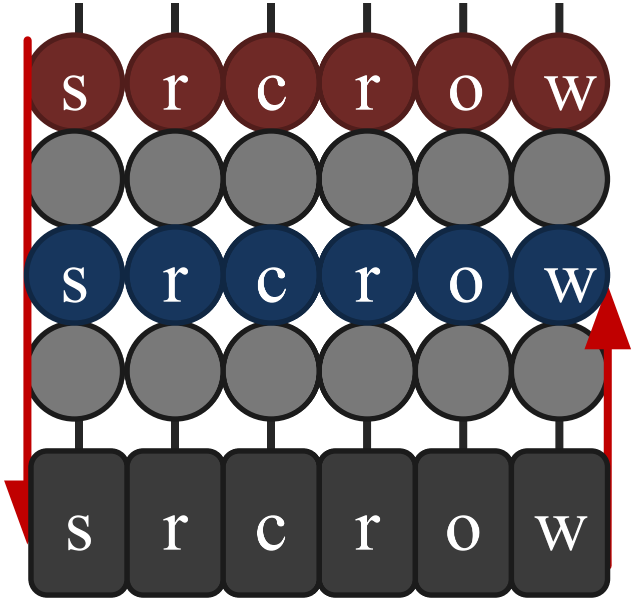

$V_{DD}/2 + \delta$
V_{DD}
sr
c
ds
t
0
0
$V_{DD}/2 + \delta$
V_{DD}
$V_{DD}/2$
0

Data gets copied

Amplify the difference

Sense Amplifier (Row Buffer)

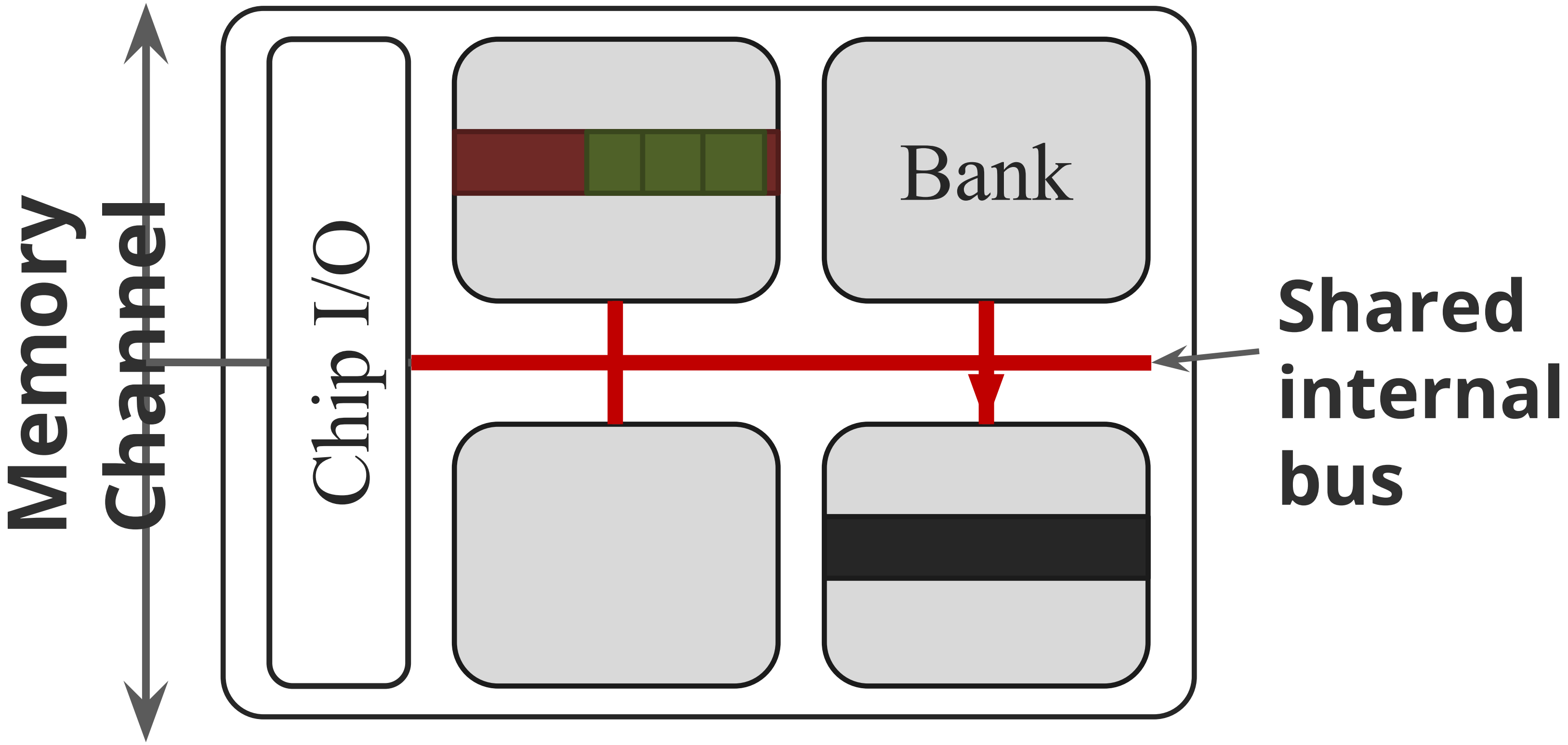
RowClone: Intra-Subarray (II)


s r c r o w
s r c r o w
s r c r o w
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)

RowClone: Inter-Bank

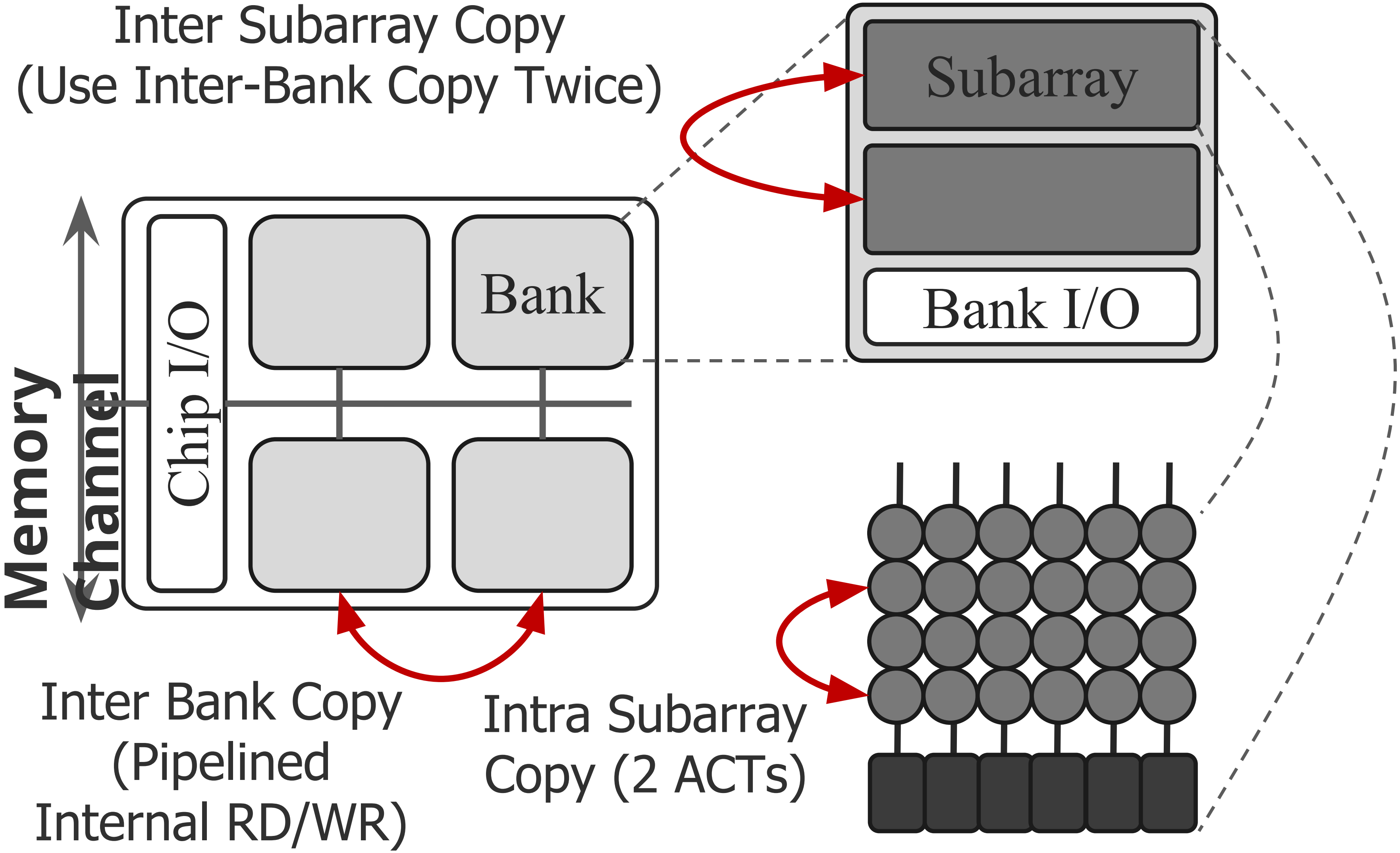

Memory Channel
Chip I/O
Bank
Shared internal bus

Overlap the latency of the read and the write

1.9X latency reduction, 3.2X energy reduction

Generalized RowClone

0.01% area cost

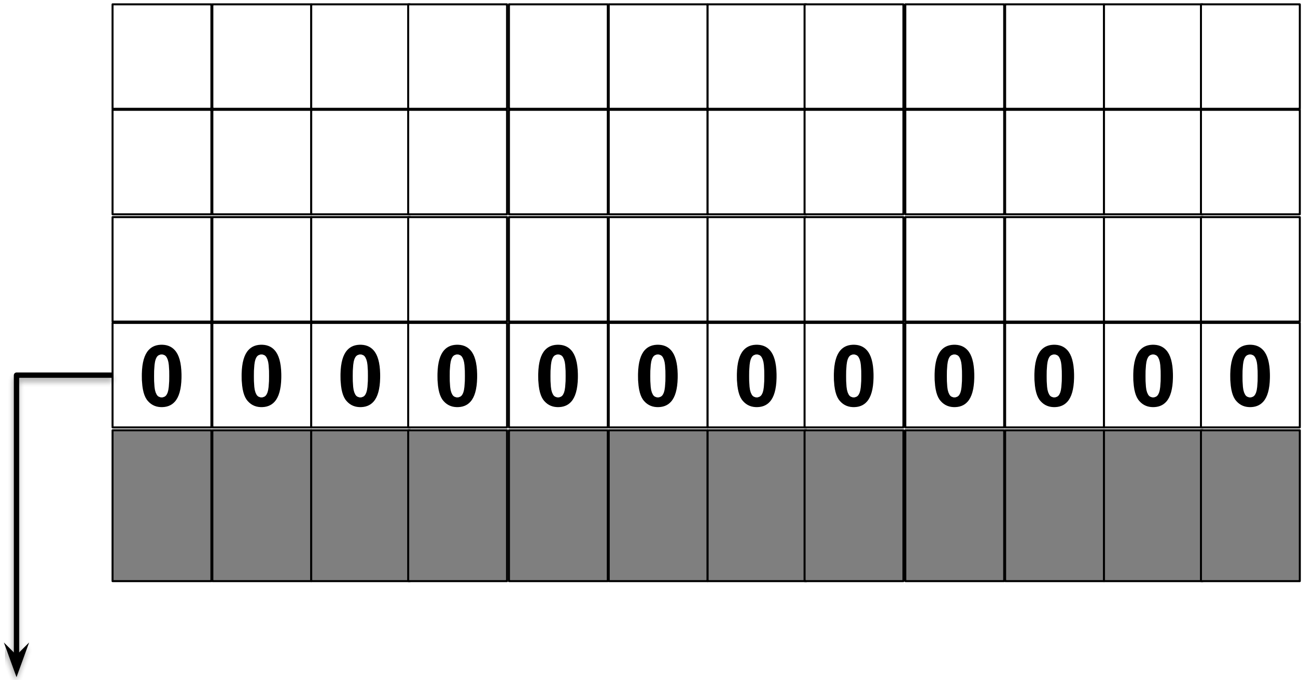

Subarray
Bank I/O
Bank
Chip I/O
Memory Channel

Inter Subarray Copy
(Use Inter-Bank Copy Twice)

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

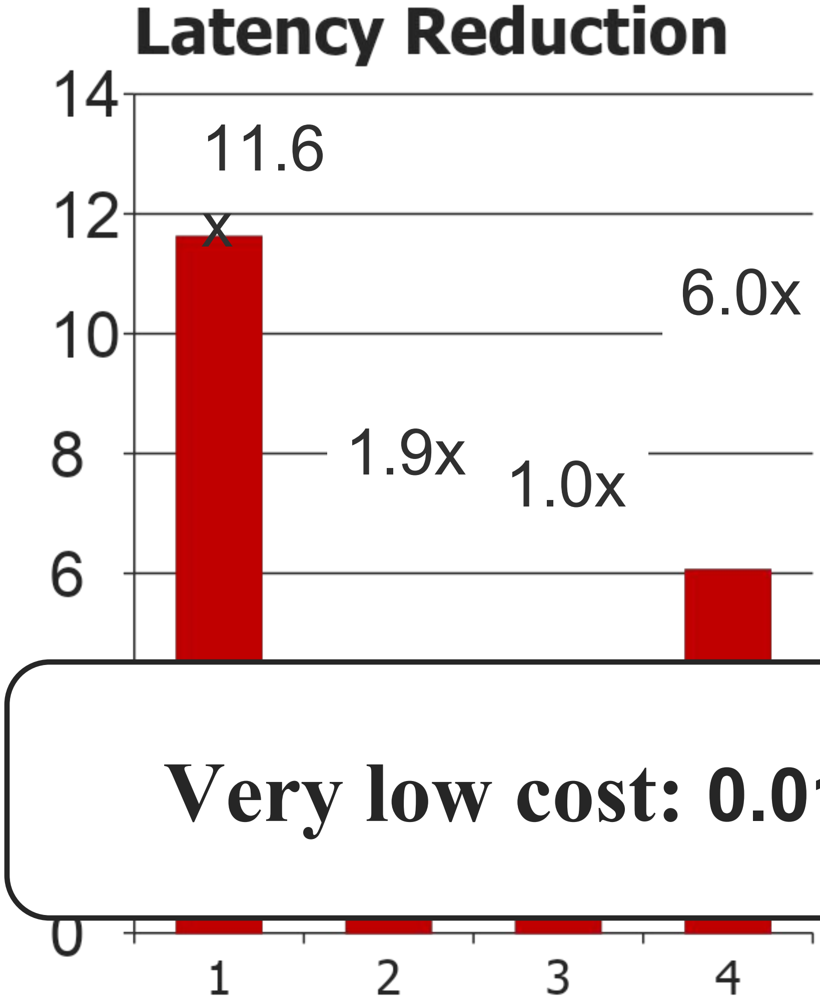
Fix a row at Zero
(0.5% loss in capacity)

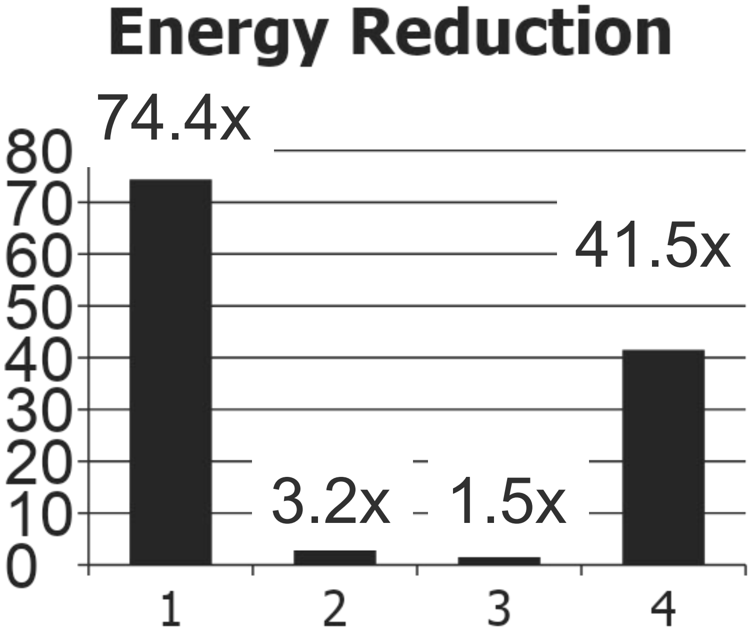
RowClone: Bulk Initialization

- 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

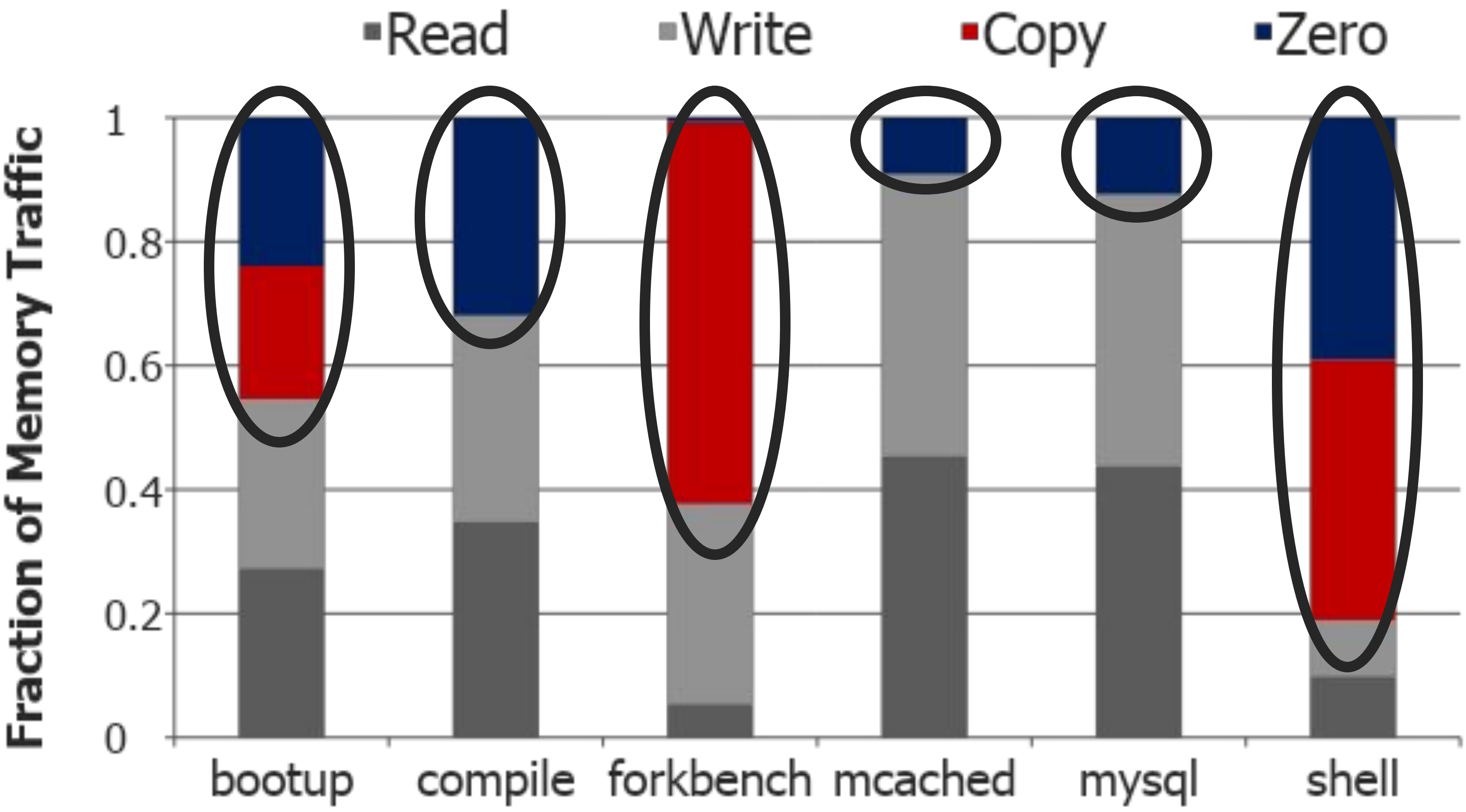
RowClone: Latency & Energy Benefits


Latency Reduction

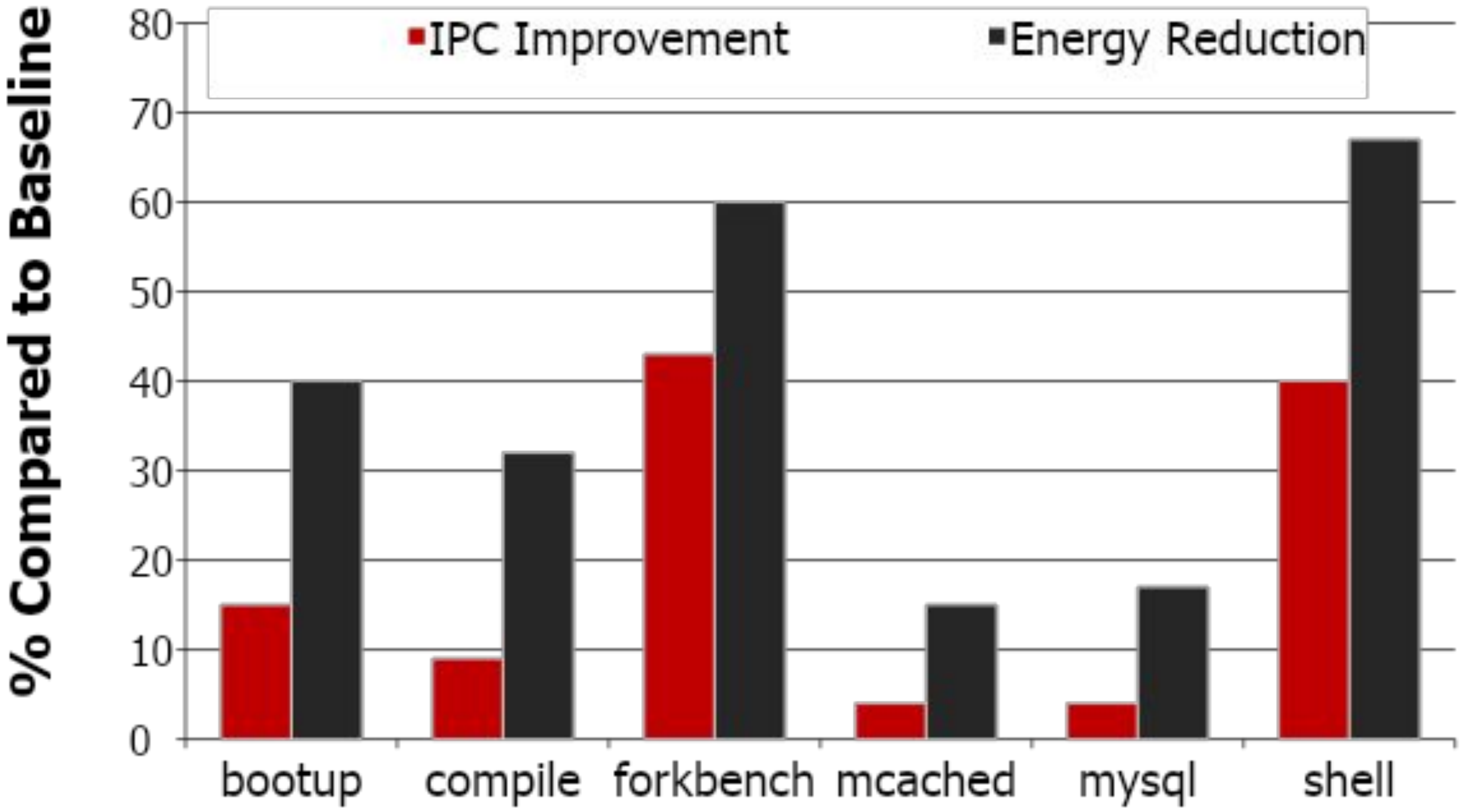

Energy Reduction

Very low cost: 0.01% increase in die area

Copy and Initialization in Workloads



RowClone: Application Performance



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?

Ambit

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

Vivek Seshadri

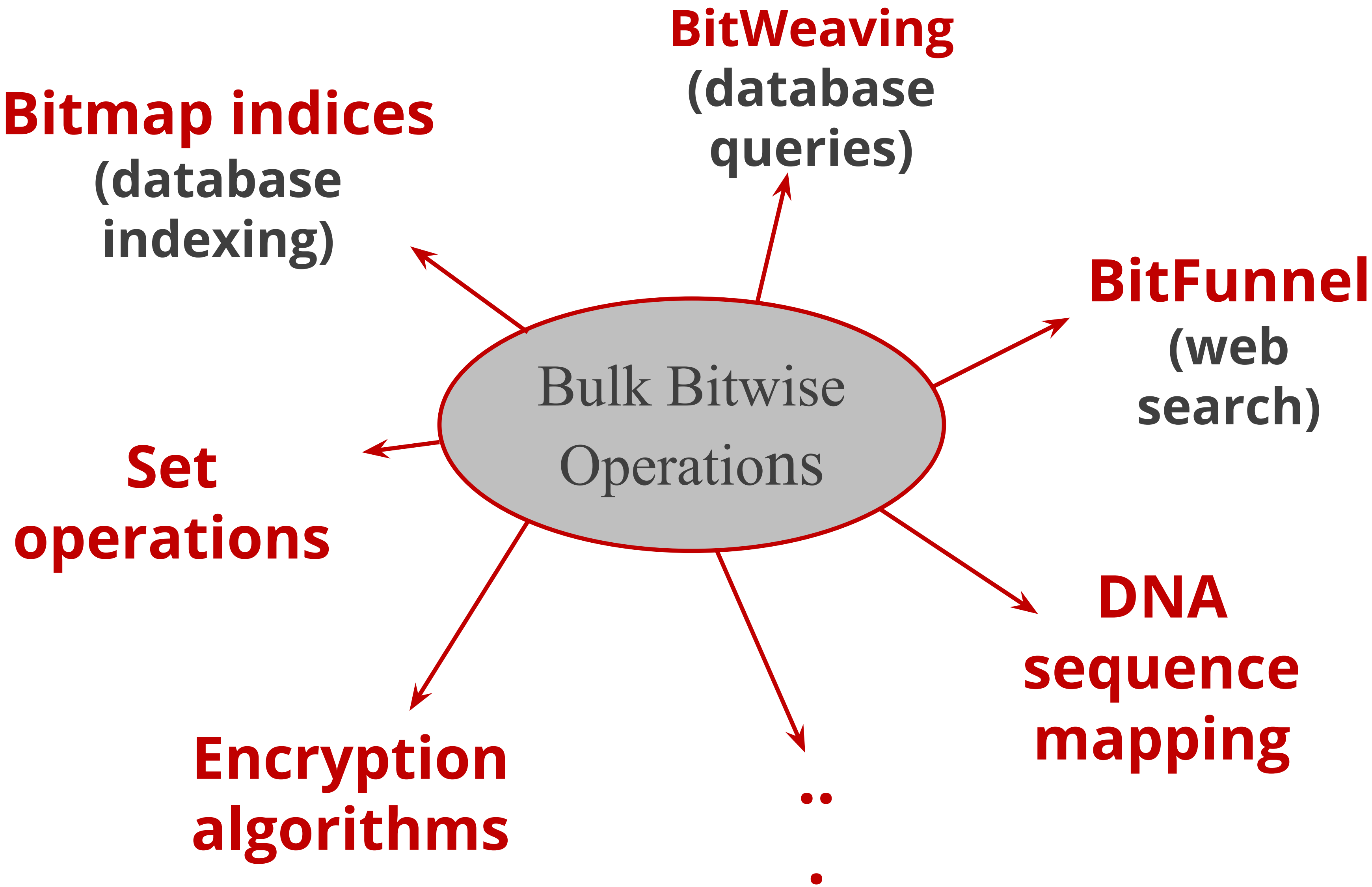
Donghyuk Lee, Thomas Mullins, Hasan Hassan, Amirali Boroumand, Jeremie Kim, Michael A. Kozuch, Onur Mutlu, Phillip B. Gibbons, Todd C. Mowry

SAFARI Carnegie Mellon intel

Microsoft ETH Zürich

Executive Summary

- **Problem: Bulk bitwise operations**
  - present in many applications, e.g., databases, search filters
  - existing systems are memory bandwidth limited
- **Our Proposal: Ambit**
  - perform bulk bitwise operations **completely inside DRAM**
  - **bulk bitwise AND/OR**: simultaneous activation of three rows
  - **bulk bitwise NOT**: inverters already in sense amplifiers
  - less than 1% area overhead over existing DRAM chips
- **Results compared to state-of-the-art baseline**
  - average across seven bulk bitwise operations
    - 32X performance improvement, 35X energy reduction
  - 3X-7X performance for real-world data-intensive applications



Bulk Bitwise Operations

Bitmap indices (database indexing)

BitWeaving (database queries)

BitFunnel (web search)

Set operations

DNA sequence mapping

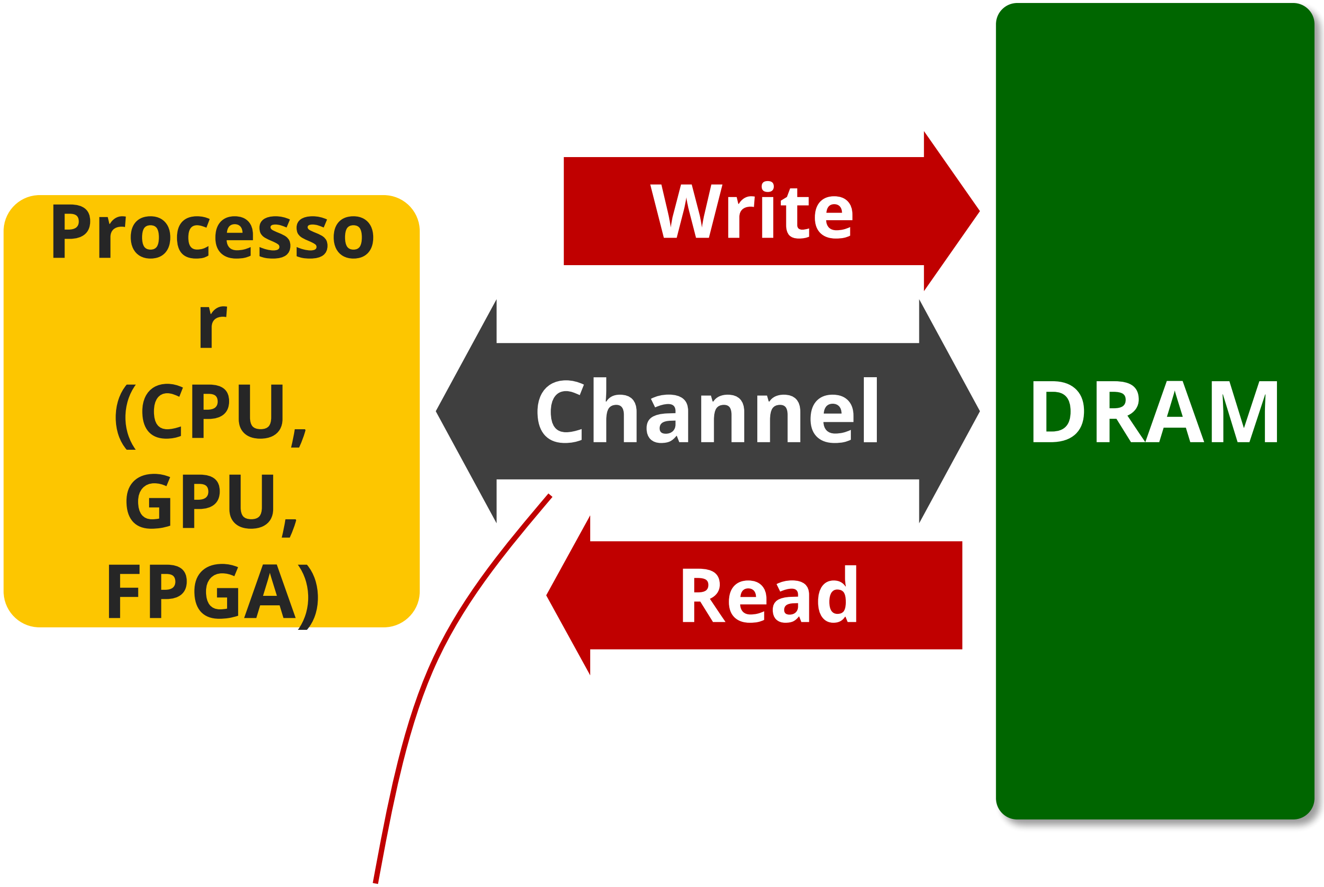
Encryption algorithms

…

[1] Li and Patel, BitWeaving, SIGMOD 2013

[2] Goodwin+, BitFunnel, SIGIR 2017

Today, DRAM is just a storage device!



Processor (CPU, GPU, FPGA)

Write

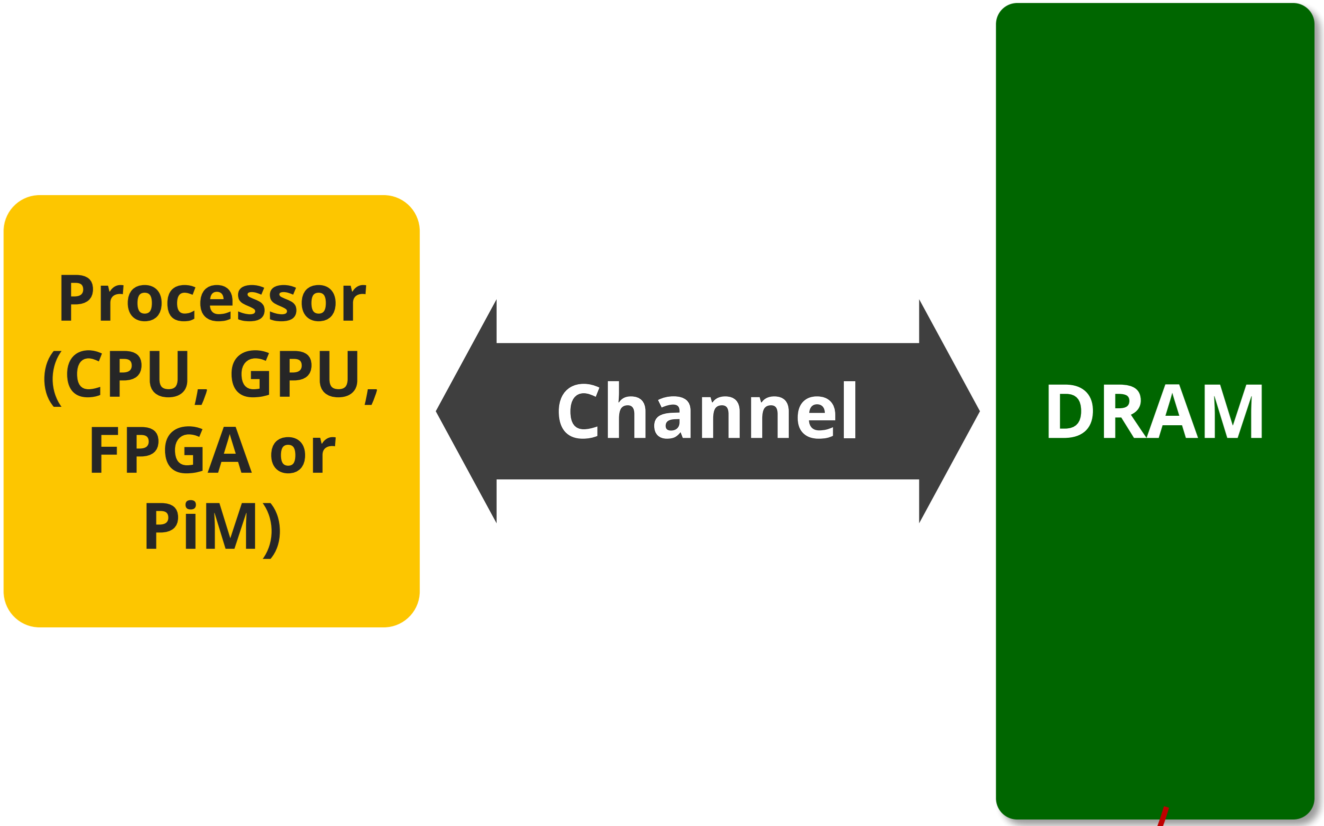
Channel

Read

DRAM

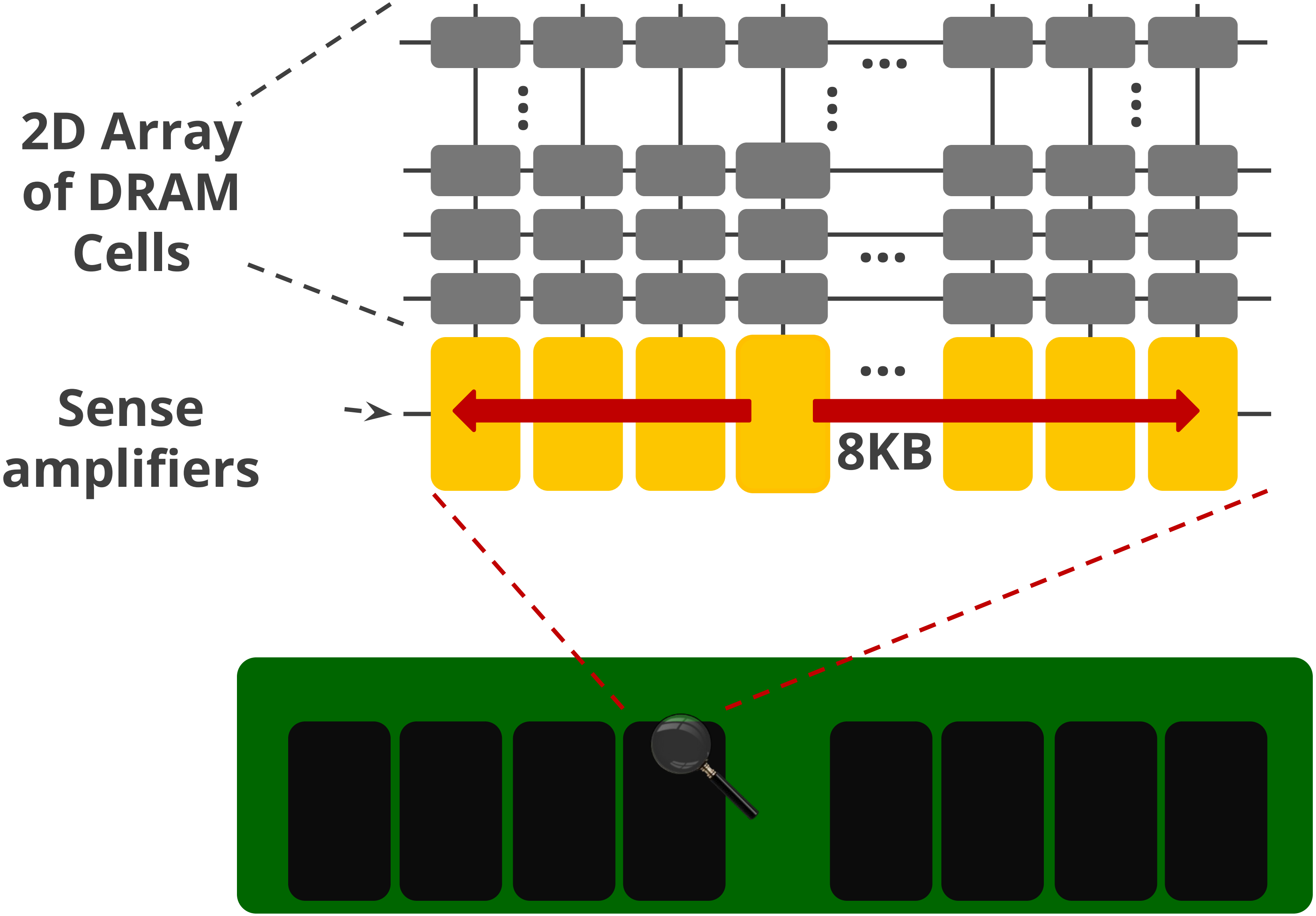
Throughput of bulk bitwise operations limited by available memory bandwidth

Our Approach


Processor (CPU, GPU, FPGA or PiM)
Channel
DRAM

Use analog operation of DRAM to perform bitwise operations completely inside memory!

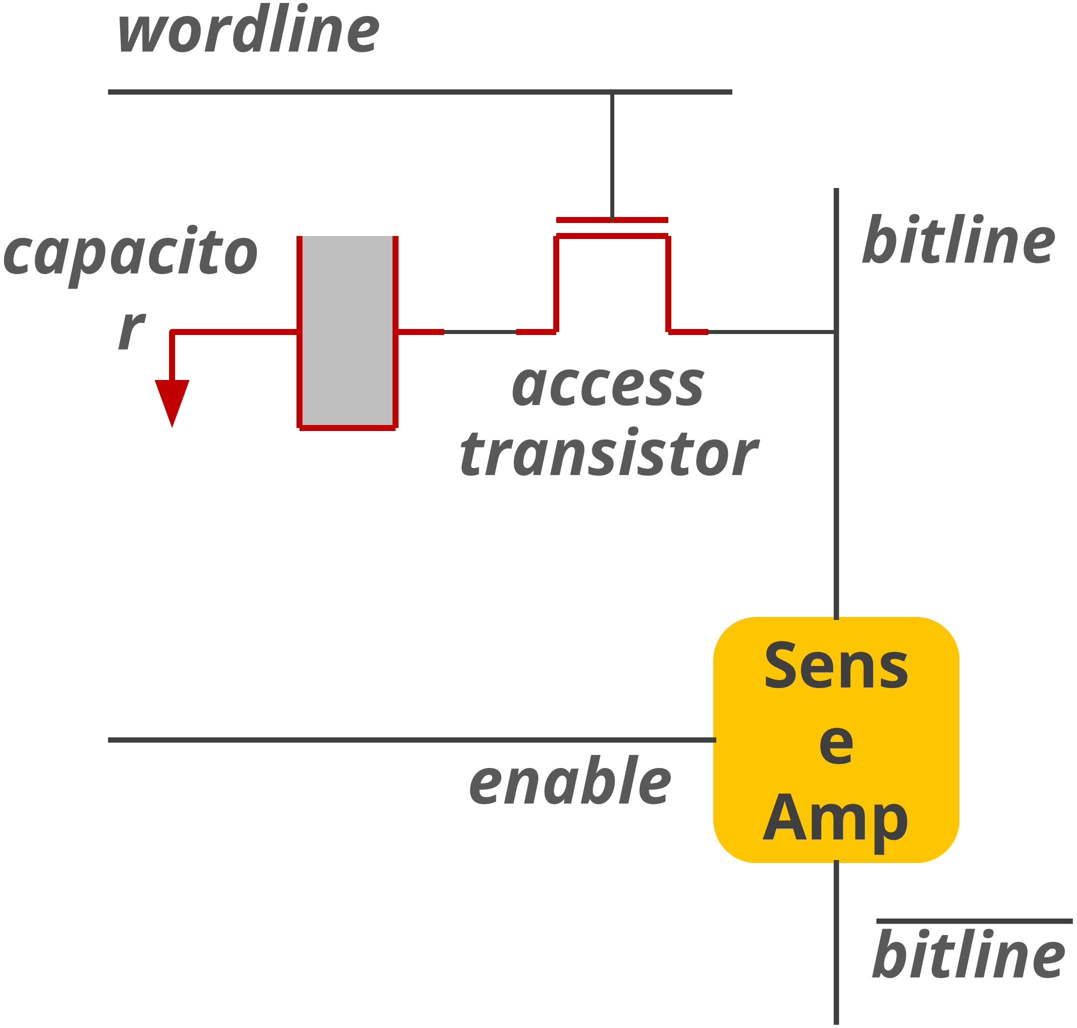
Inside a DRAM Chip



2D Array of DRAM Cells

Sense amplifiers

DRAM Cell Operation



wordline

capacitor

access transistor

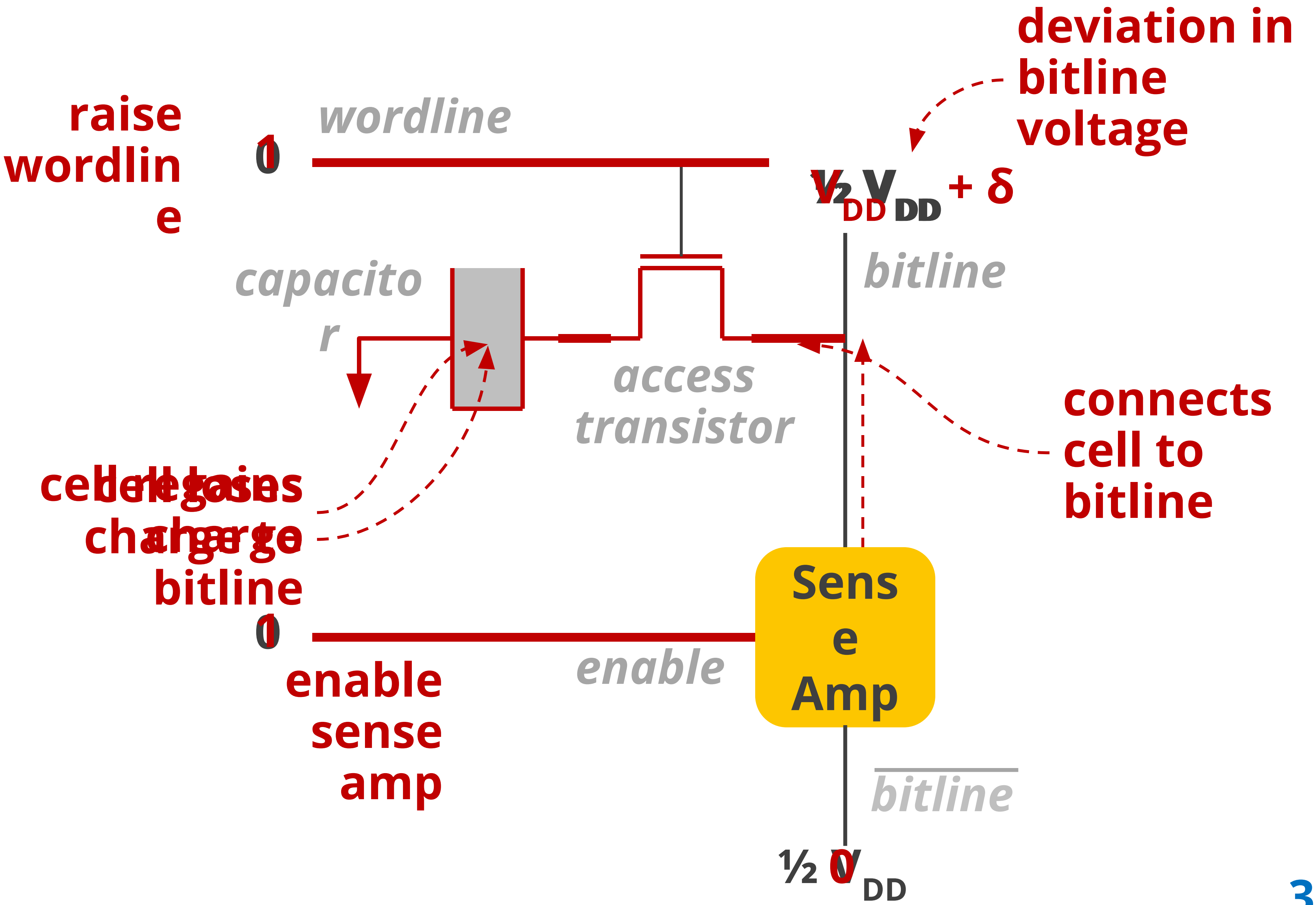
bitline

Sense Amp

enable

$\overline{\text{bitline}}$

DRAM Cell Operation


wordline
capacitor
access transistor
bitline
Sense Amp
enable
$\overline{\text{bitline}}$
0
1
0
1
½ V$_{DD}$
V$_{DD}$
+ δ
½ V$_{DD}$
0

raise wordline

deviation in bitline voltage

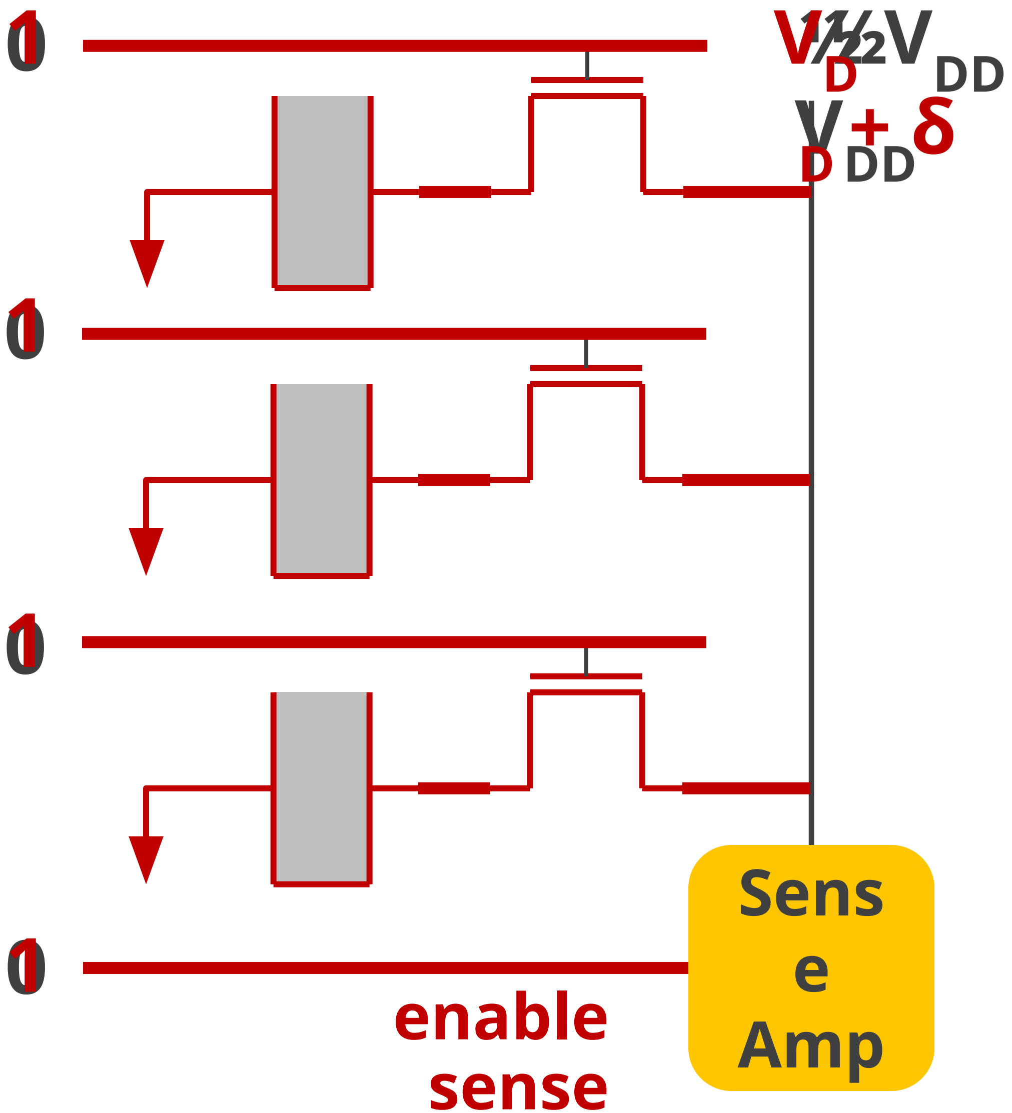
connects cell to bitline

cell loses charge to bitline

cell regains charge

enable sense amp

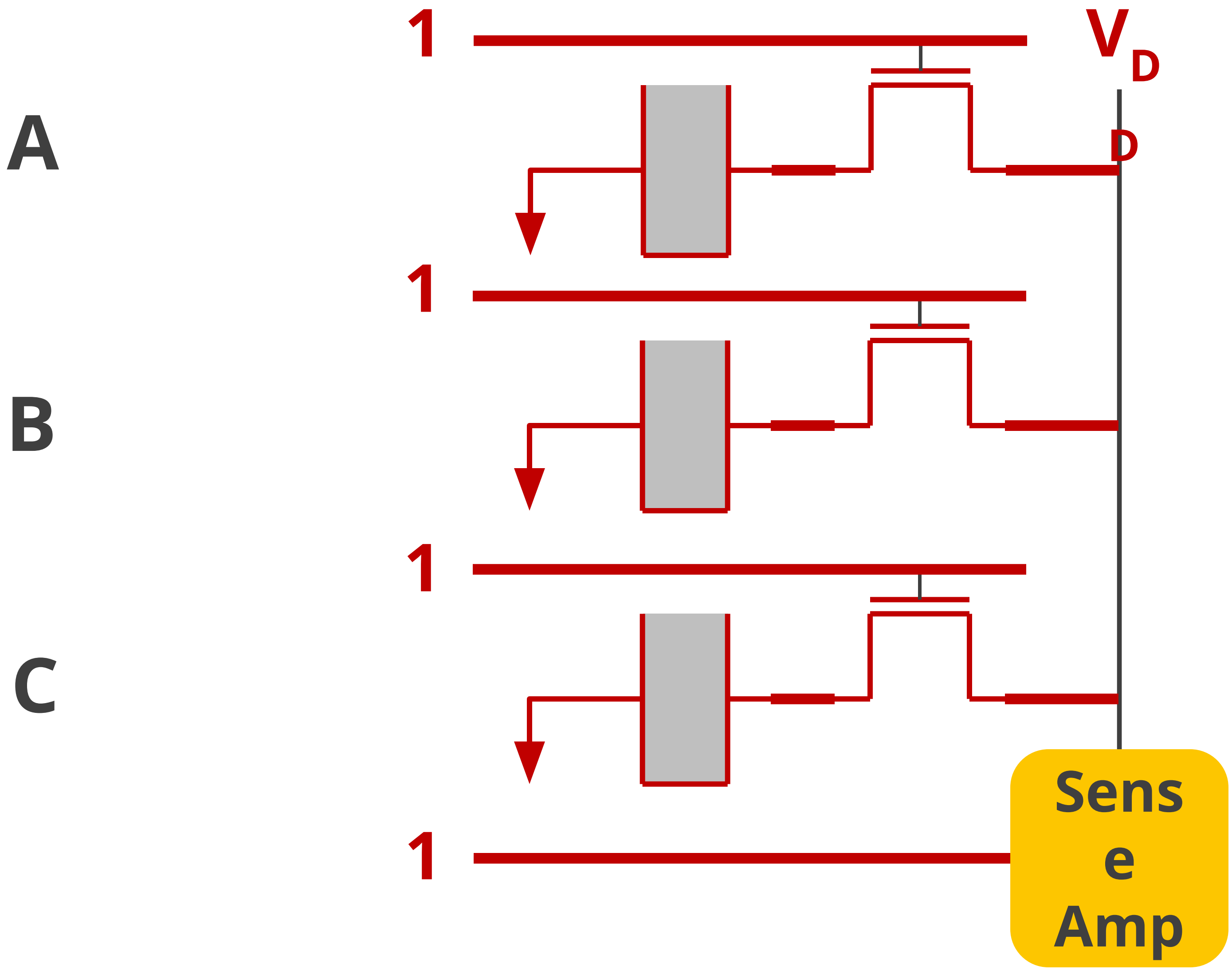
Triple-Row Activation: Majority Function


0
1
½V
DD
V
DD
V
DD
+ δ
0
1
0
1
0
1
Sense Amp

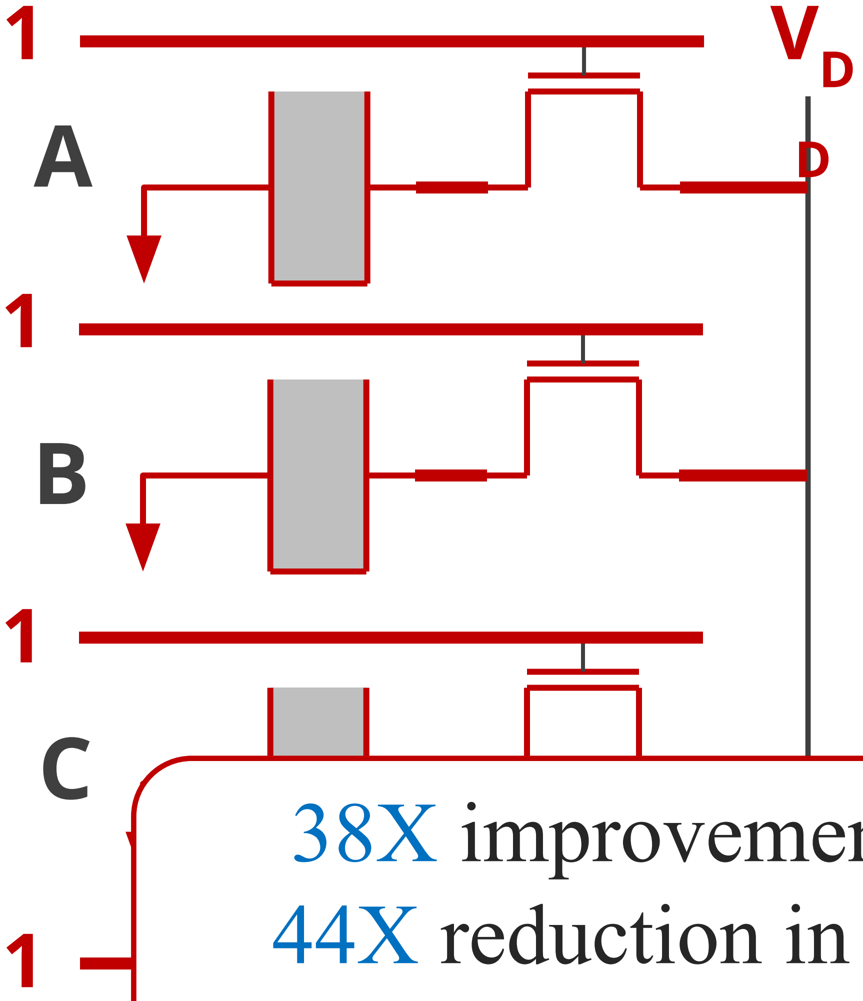
activate all three rows

enable sense amp

Bitwise AND/OR Using Triple-Row Activation


1
V_D
D
A
1
B
1
C
1
Sense Amp

Bitwise AND/OR Using Triple-Row Activation



Output = AB + BC + CA

= C (A **OR** B) + ~C (A **AND** B)

Control the value of C to perform bitwise OR

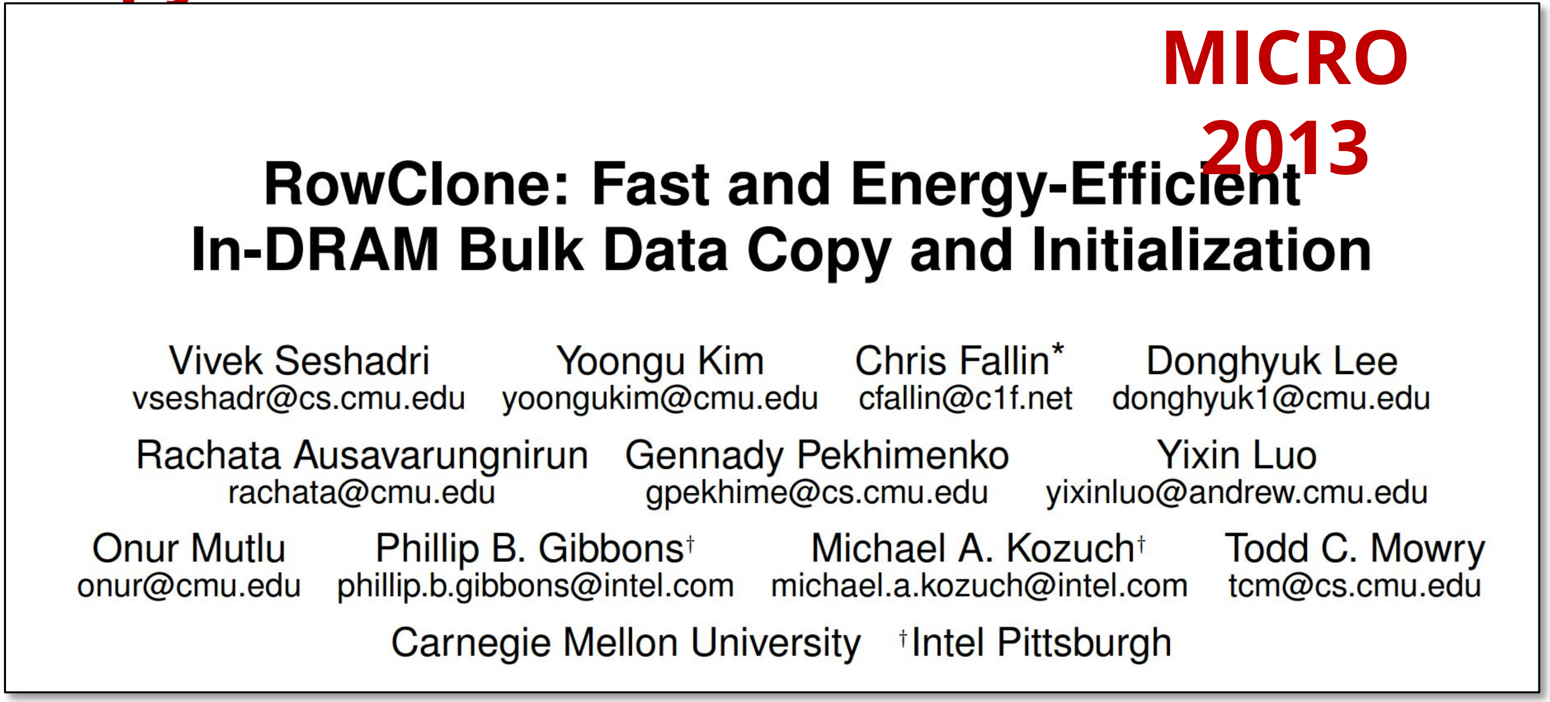
38X improvement in raw throughput
44X reduction in energy consumption
for bulk bitwise AND/OR operations

Bulk Bitwise AND/OR in DRAM

Statically reserve three designated rows **t1**, **t2**, and **t3**

Result = row A **AND/OR** row B

1. **Copy** data of row **A** to row **t1**
2. **Copy** data of row **B** to row **t2**
3.
4.
5.



MICRO 2013

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

Bulk Bitwise AND/OR in DRAM

Statically reserve three designated rows t1, t2, and t3

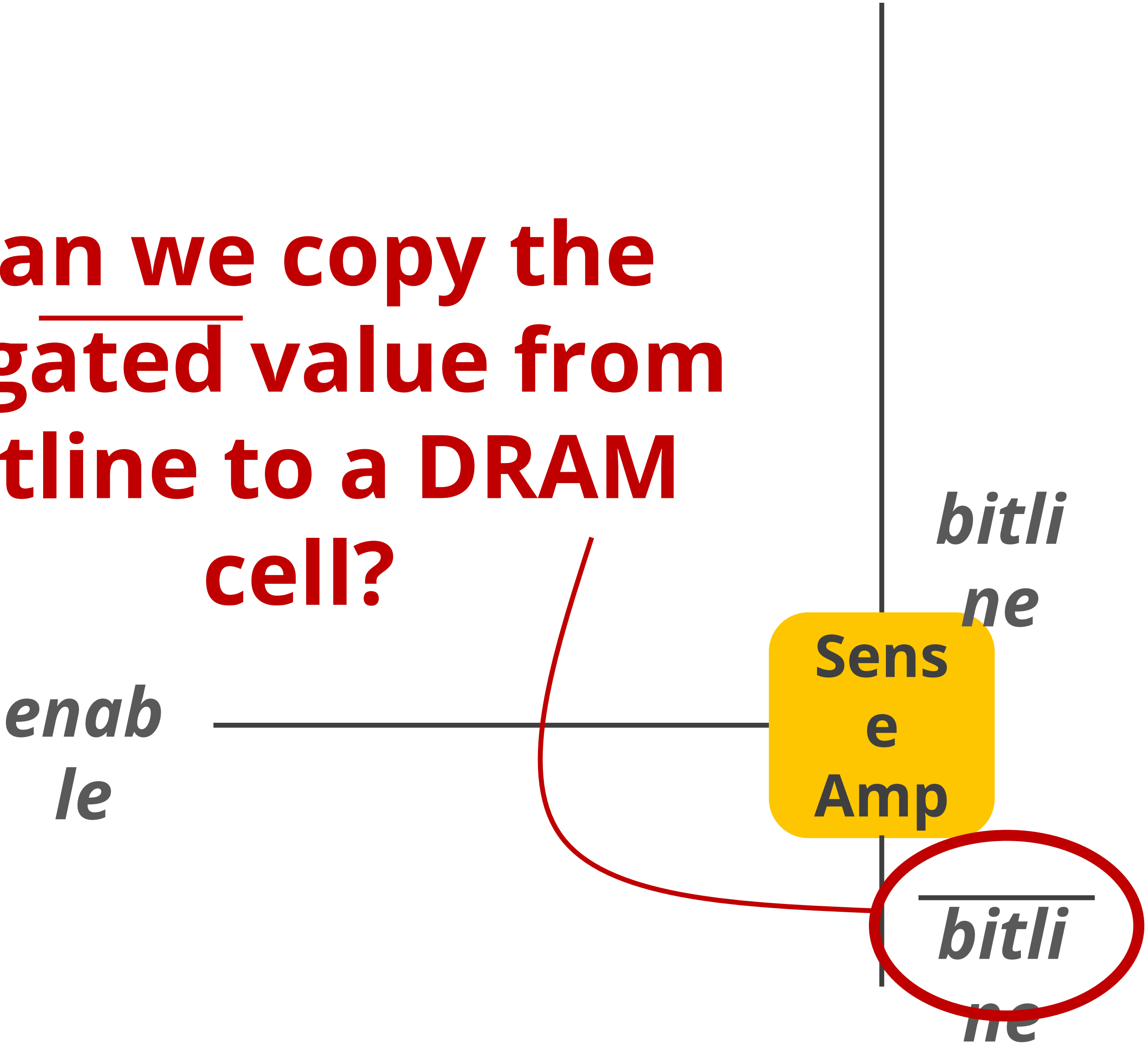
Result = row A AND/OR row B

1. ~~**Copy**~~ **RowClone** data of row **A** to row **t1**
2. ~~**Copy**~~ **RowClone** data of row **B** to row **t2**
3. ~~**Initialize**~~ **RowClone** data of row **t3** to **0/1**
4. **Activate** rows **t1/t2/t3** simultaneously
5. ~~**Copy**~~ **RowClone** data of row **t1/t2/t3** to **Result** row

1. **Copy** data of row **A** to row **t1**
2. **Copy** data of row **B** to row **t2**
3. **Initialize** data of row **t3** to **0/1**
4. **Activate** rows **t1/t2/t3** simultaneously
5. **Copy** data of row **t1/t2/t3** to **Result** row

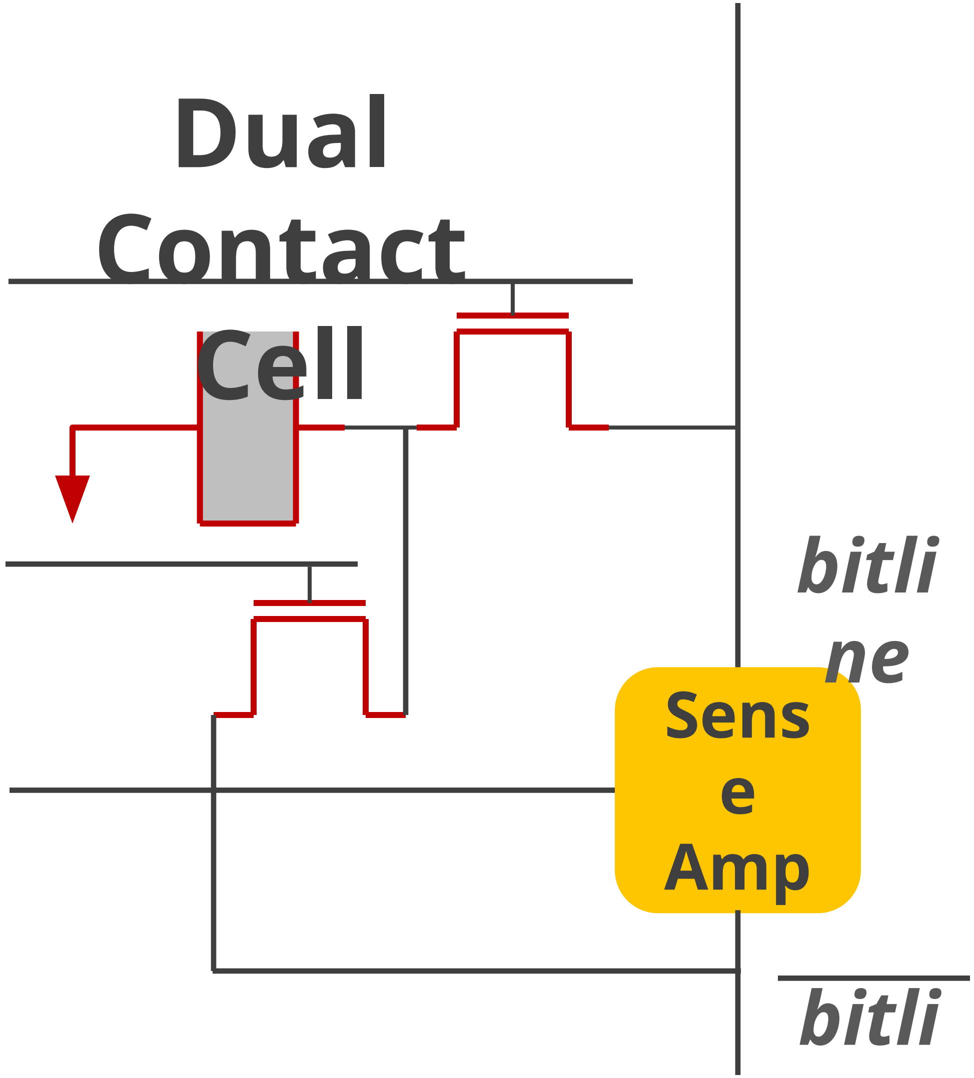
Use RowClone to perform copy and initialization operations completely in DRAM!

Negation Using the Sense Amplifier


Can we copy the $\overline{\text{negated}}$ value from bitline to a DRAM cell?

bitline
Sense Amp
enable
$\overline{\text{bitline}}$

Negation Using the Sense Amplifier



Dual Contact Cell

Regular wordline

Negation wordline

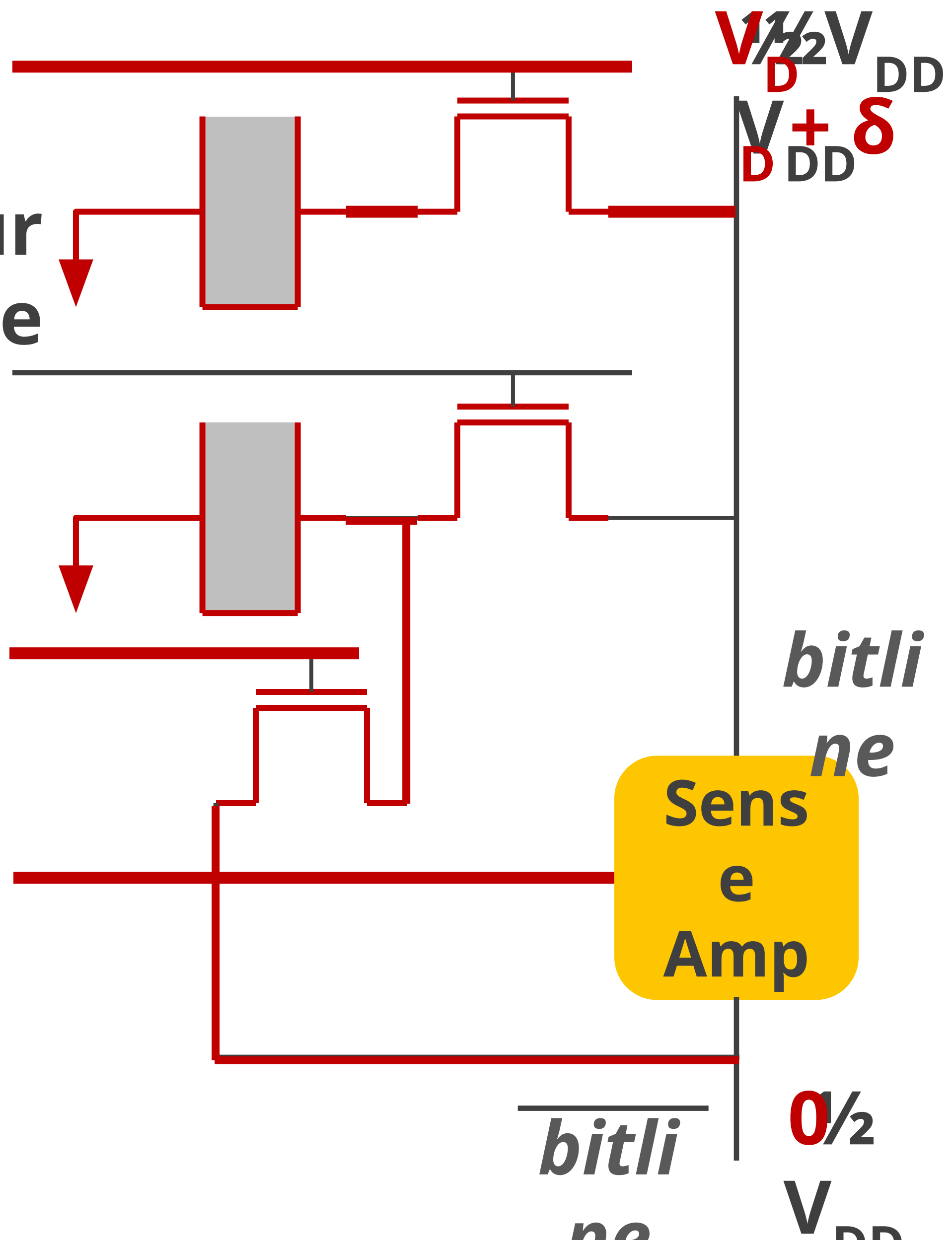
enable

bitline

Sense Amp

$\overline{\textbf{bitline}}$

Negation Using the Sense Amplifier



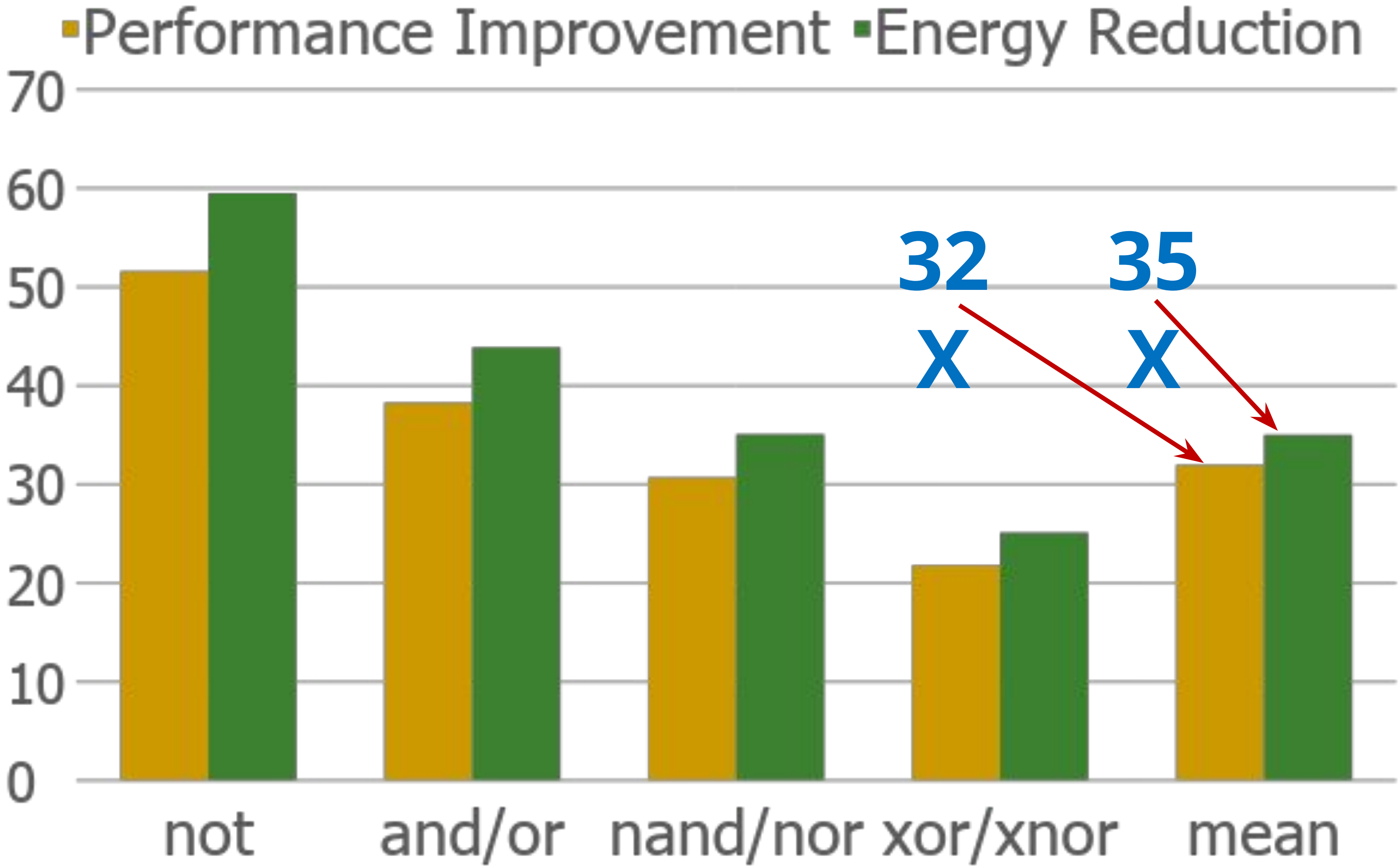
activate source

activate negation wordline

enable sense amp

V_{DD}
$\frac{1}{2}V_{DD}$
$V_{DD}+\delta$
source
bitline
Sense Amp
$\overline{\text{bitline}}$
0
$\frac{1}{2}V_{DD}$

Ambit vs. DDR3: Performance and Energy



32 X

35 X

Integrating Ambit with the System

1. **PCIe device**
   - Similar to other accelerators (e.g., GPU)

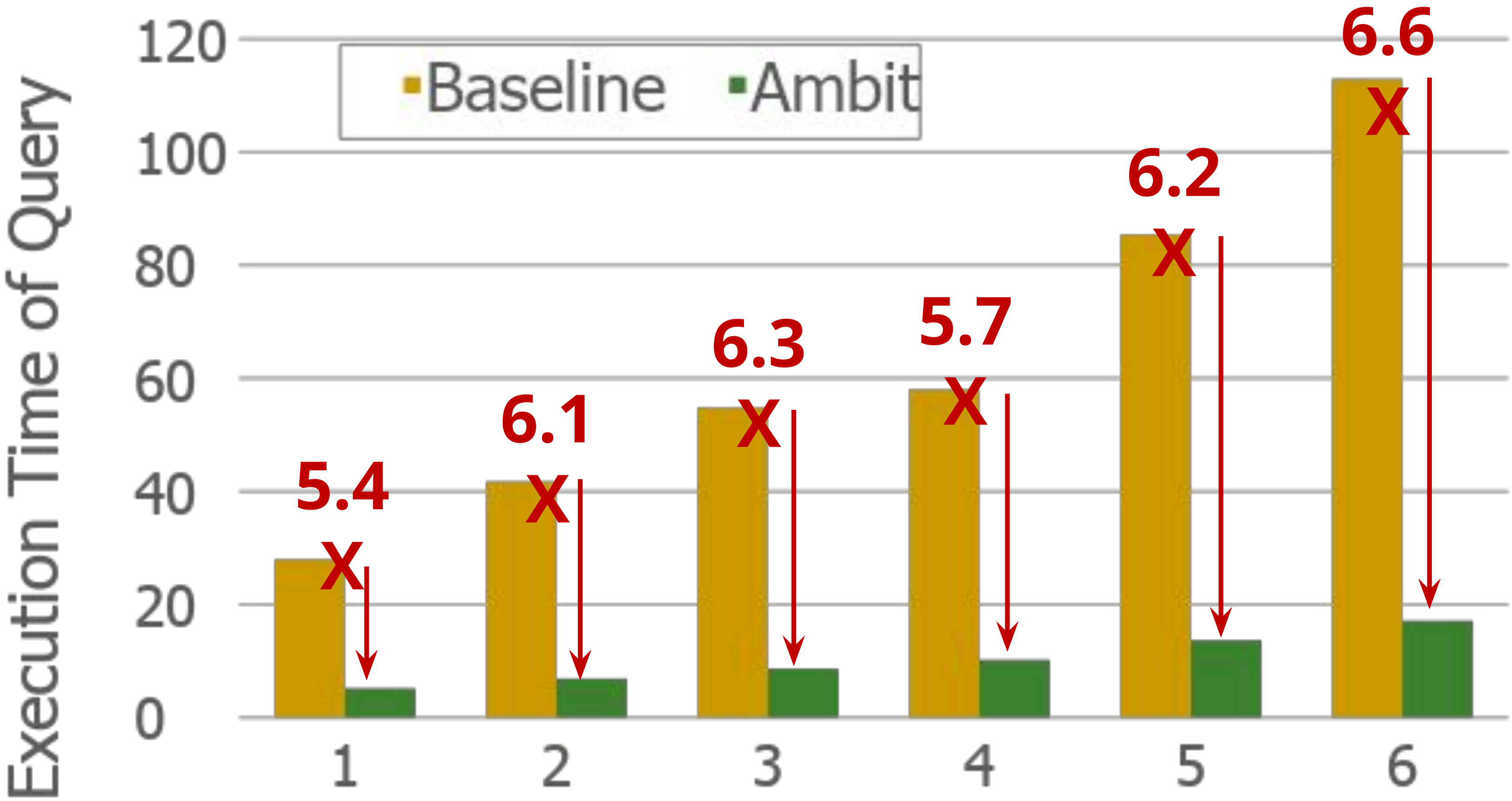
2. **System memory bus**
   - Ambit uses the same DRAM command/address interface

Pros and cons discussed in paper (Section 5.4)

Real-world Applications

- **Methodology** (Gem5 simulator)
  - Processor: x86, 4 GHz, out-of-order, 64-entry instruction queue
  - L1 cache: 32 KB D-cache and 32 KB I-cache, LRU policy
  - L2 cache: 2 MB, LRU policy
  - Memory controller: FR-FCFS, 8 KB row size
  - Main memory: DDR4-2400, 1 channel, 1 rank, 8 bank
- **Workloads**
  - Database bitmap indices
  - BitWeaving –column scans using bulk bitwise operations
  - Set operations – comparing bitvectors with red-black trees

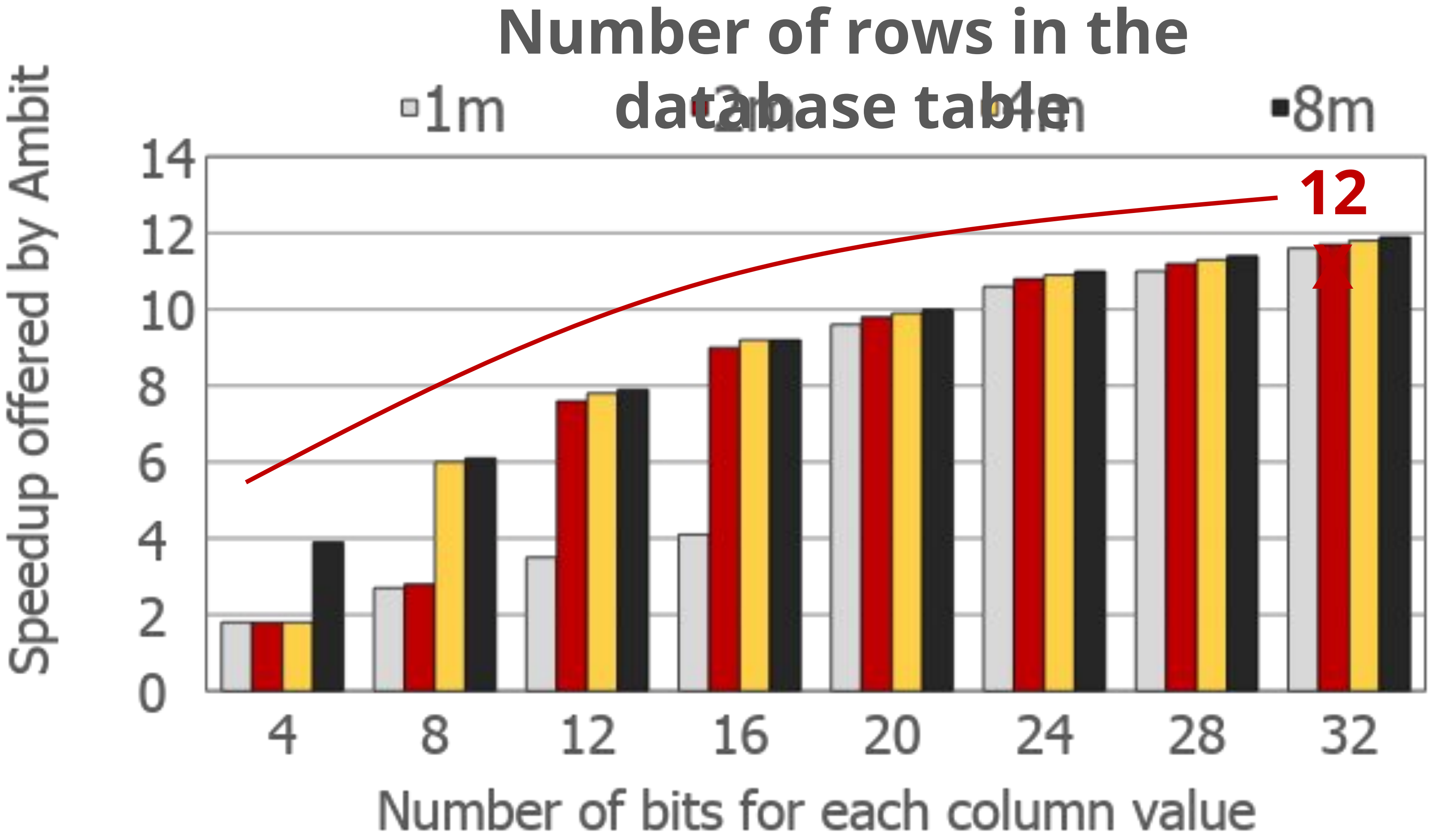
Bitmap Indices: Performance



Consistent reduction in execution time. 6X on average

Speedup offered by Ambit for BitWeaving

`select count(*) where c1 < field < c2`



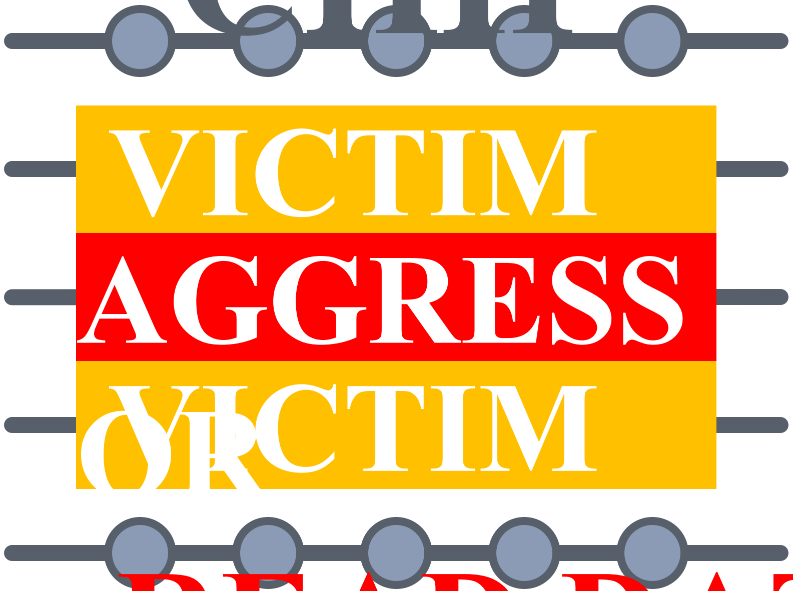
FLIPPING BITS IN MEMORY WITHOUT ACCESSING THEM

ISCA 2014

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

ROW HAMMER

DRAM CHIP



VICTIM

AGGRESSOR

VICTIM

WORDLINE

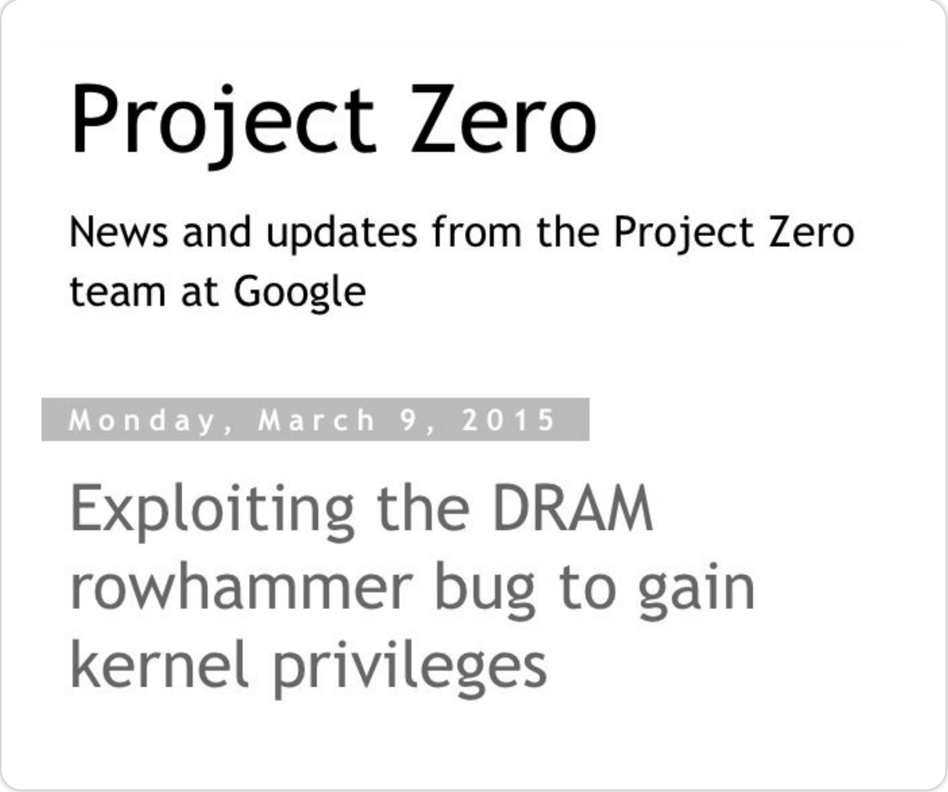
LOW

HIGH

HIGH VOLTAGE

READ DATA FROM HERE,
GET ERRORS OVER THERE

GOOGLE'S EXPLOIT



Project Zero

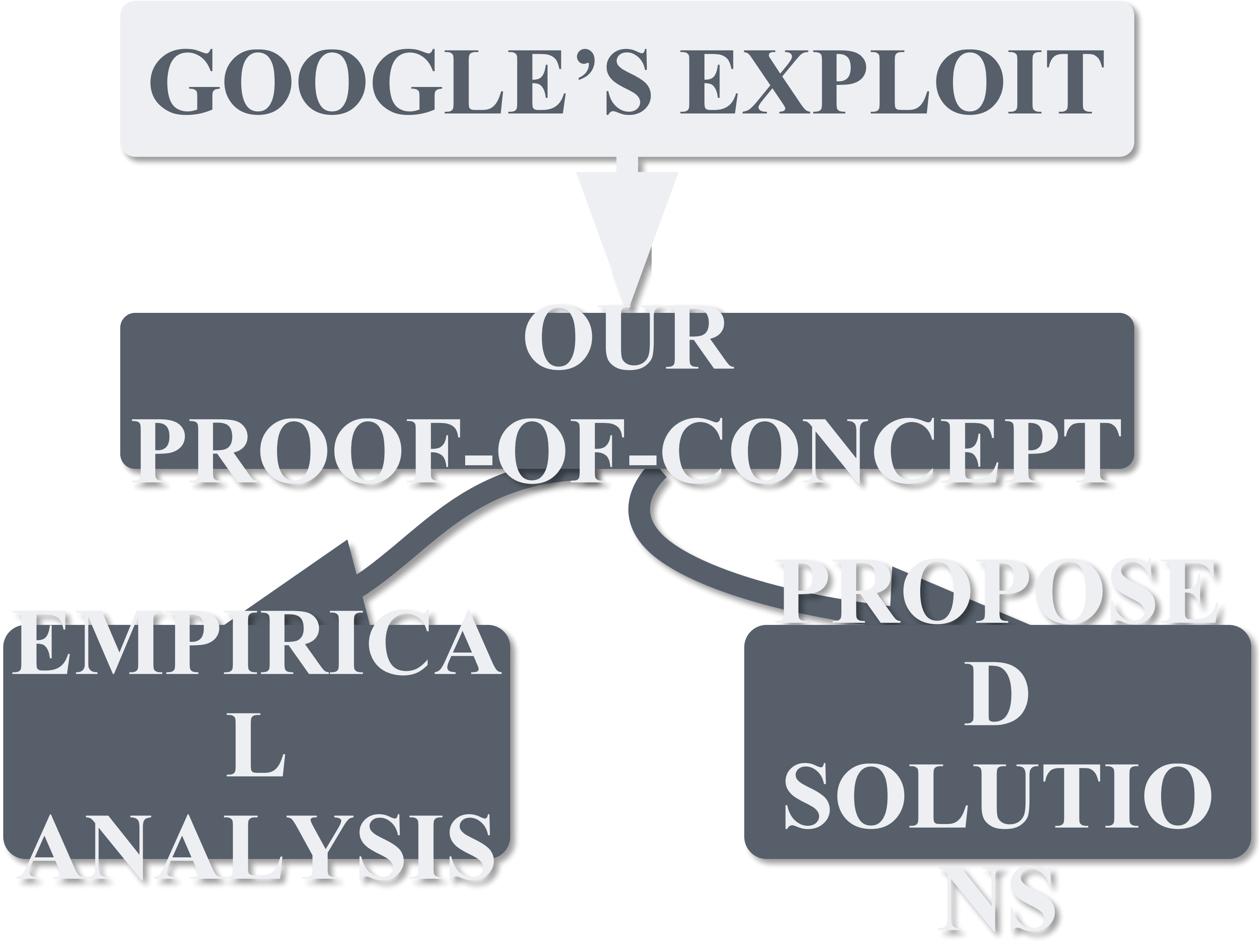
News and updates from the Project Zero team at Google

Monday, March 9, 2015

Exploiting the DRAM rowhammer bug to gain kernel privileges

"We learned about rowhammer from Yoongu Kim et al."

http://googleprojectzero.blogspot.com



GOOGLE’S EXPLOIT

OUR PROOF-OF-CONCEPT

EMPIRICAL ANALYSIS

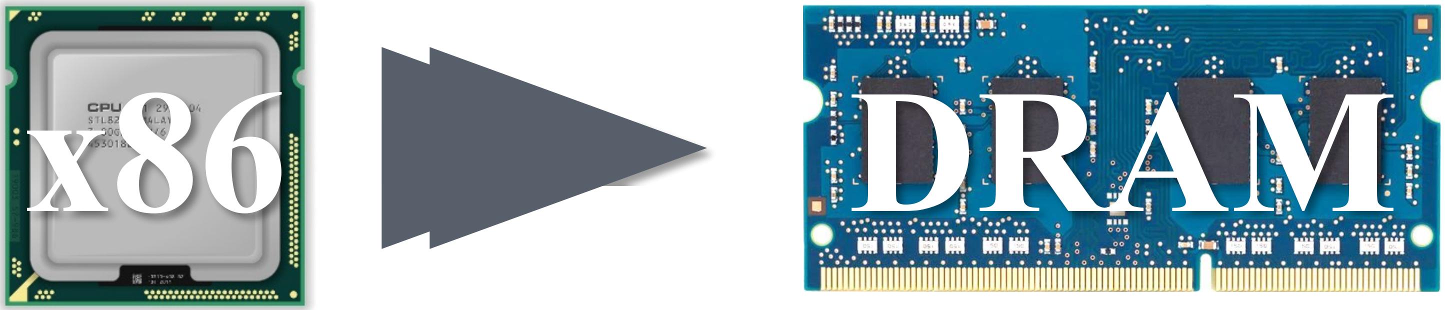
PROPOSED SOLUTIONS

REAL SYSTEM

MANY READS TO SAME ADDRESS

≠

OPEN/CLOSE SAME ROW



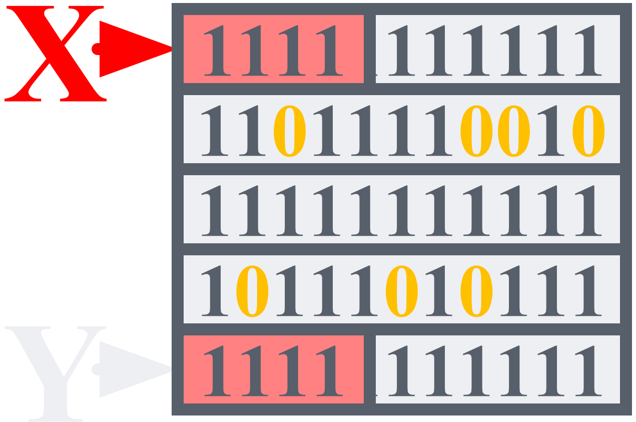
1. CACHE HITS

2. ROW HITS

x86 CPU

DRAM

```
LOOP:
    mov (X), %reg
    mov (Y), %reg
    clflush (X)
    clflush (Y)
    jmp LOOP
```



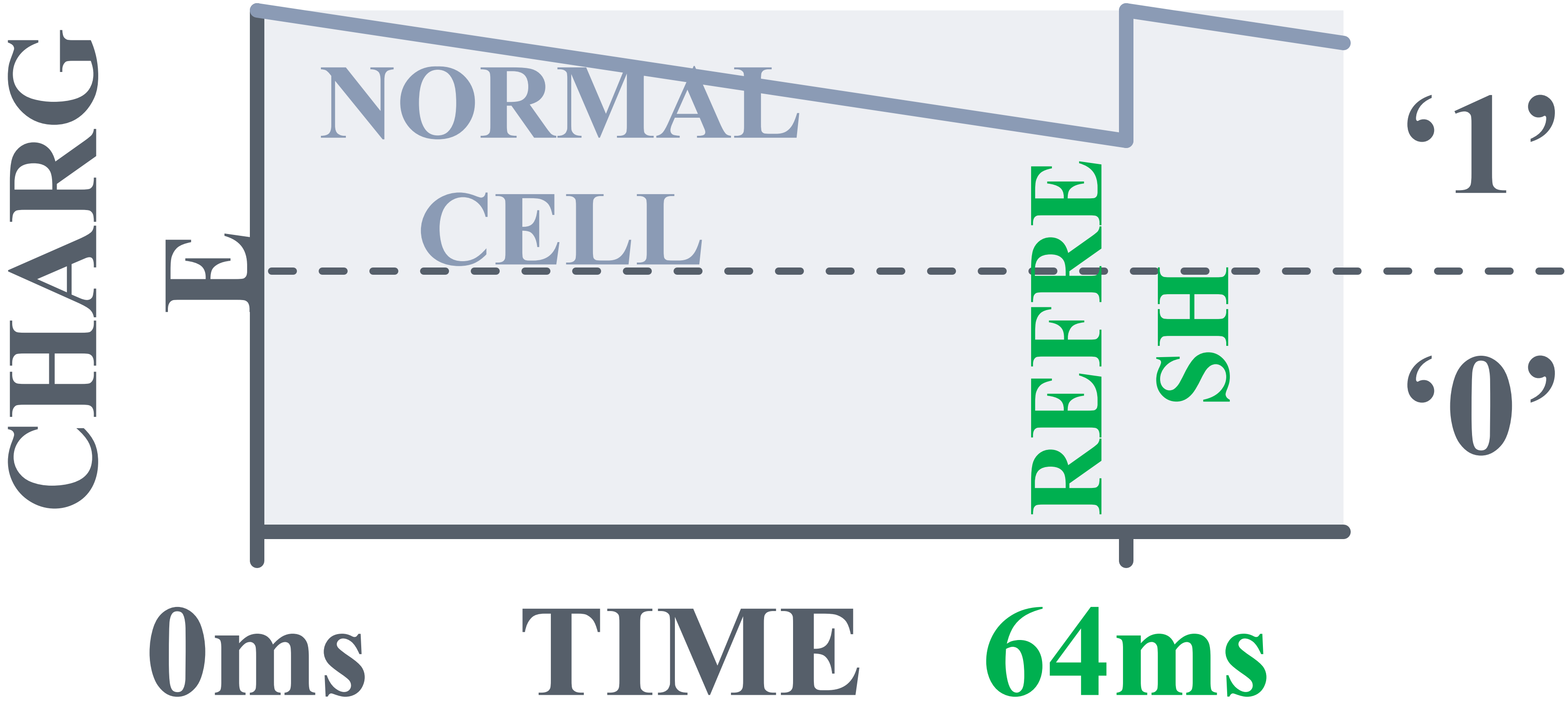
1111 111111
11011110010
11111111111
10111010111
1111 111111

MANY ERRORS!

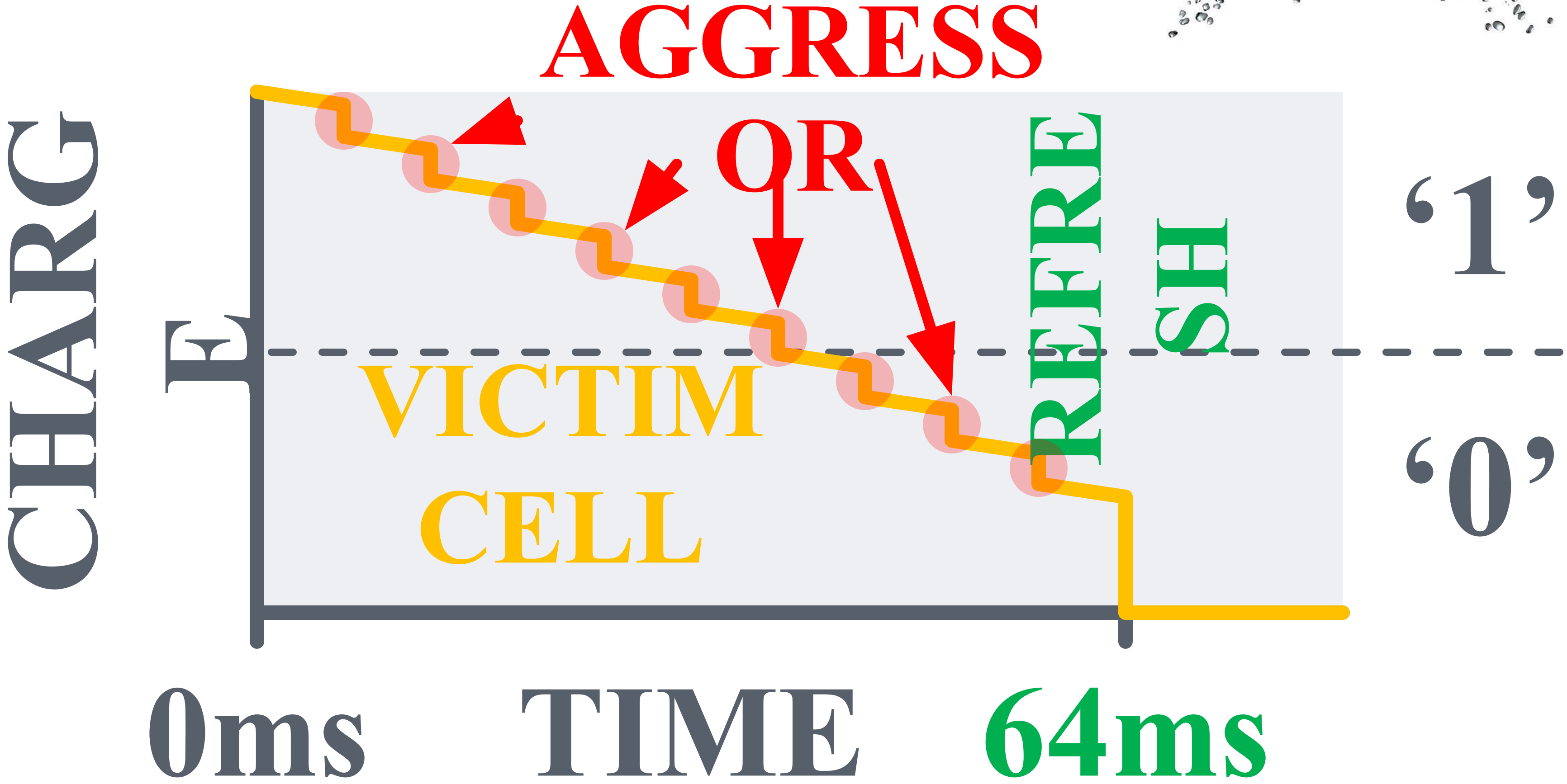
http://www.github.com/CMU-SAFARI/rowhammer

WHY DO THE ERRORS OCCUR?

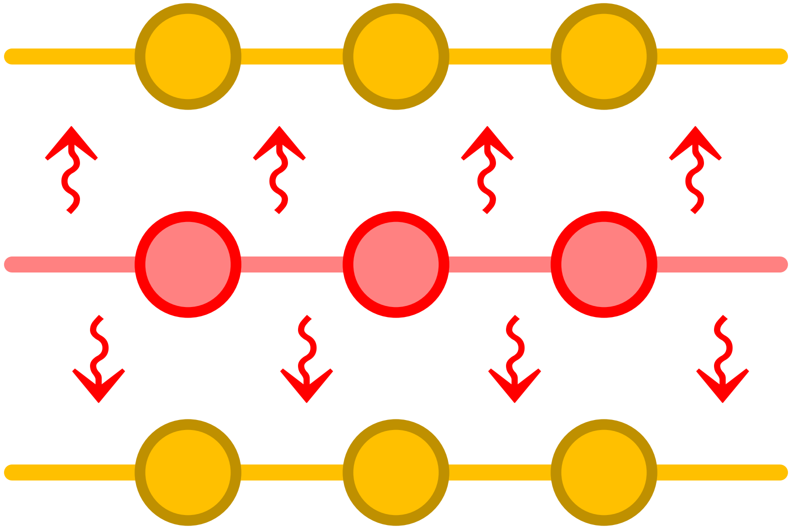
DRAM CELLS ARE LEAKY


CHARGE
NORMAL CELL
REFRESH
'1'
'0'
0ms
TIME
64ms

DRAM CELLS ARE LEAKY


AGGRESS
OR
CHARG
E
VICTIM
CELL
REFRE
SH
'1'
'0'
0ms
TIME
64ms

ROOT CAUSE?



COUPLING

- Electromagnetic
- Tunneling

ACCELERATES CHARGE LOSS

AS DRAM SCALES …

- **CELLS BECOME SMALLER**
  Less tolerance to coupling effects

- **CELLS BECOME PLACED CLOSER**
  Stronger coupling effects

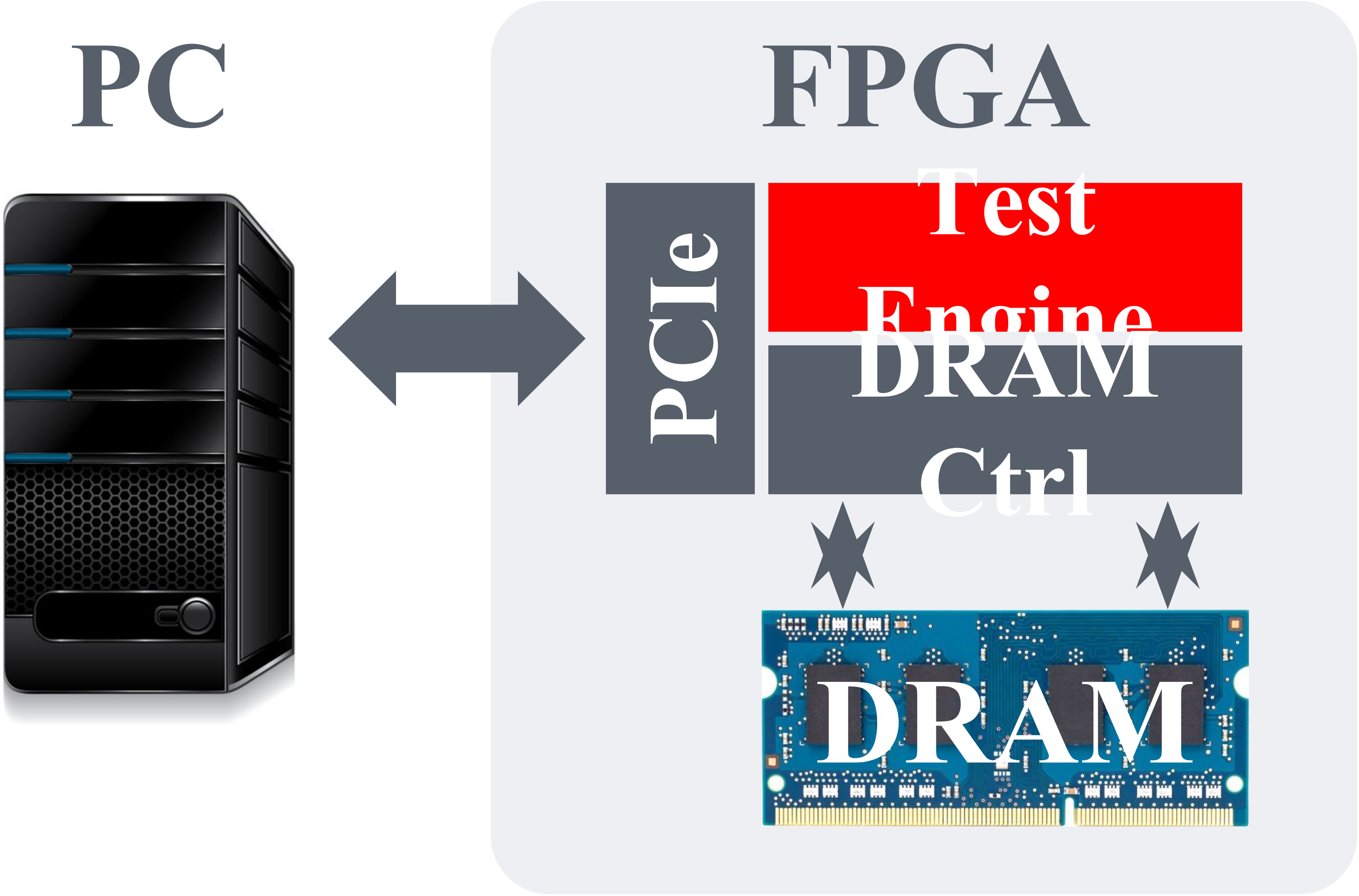
1. **ERRORS ARE RECENT**

   Not found in pre-2010 chips

2. **ERRORS ARE WIDESPREAD**

   >80% of chips have errors

   Up to one error per ~1K cells



PC

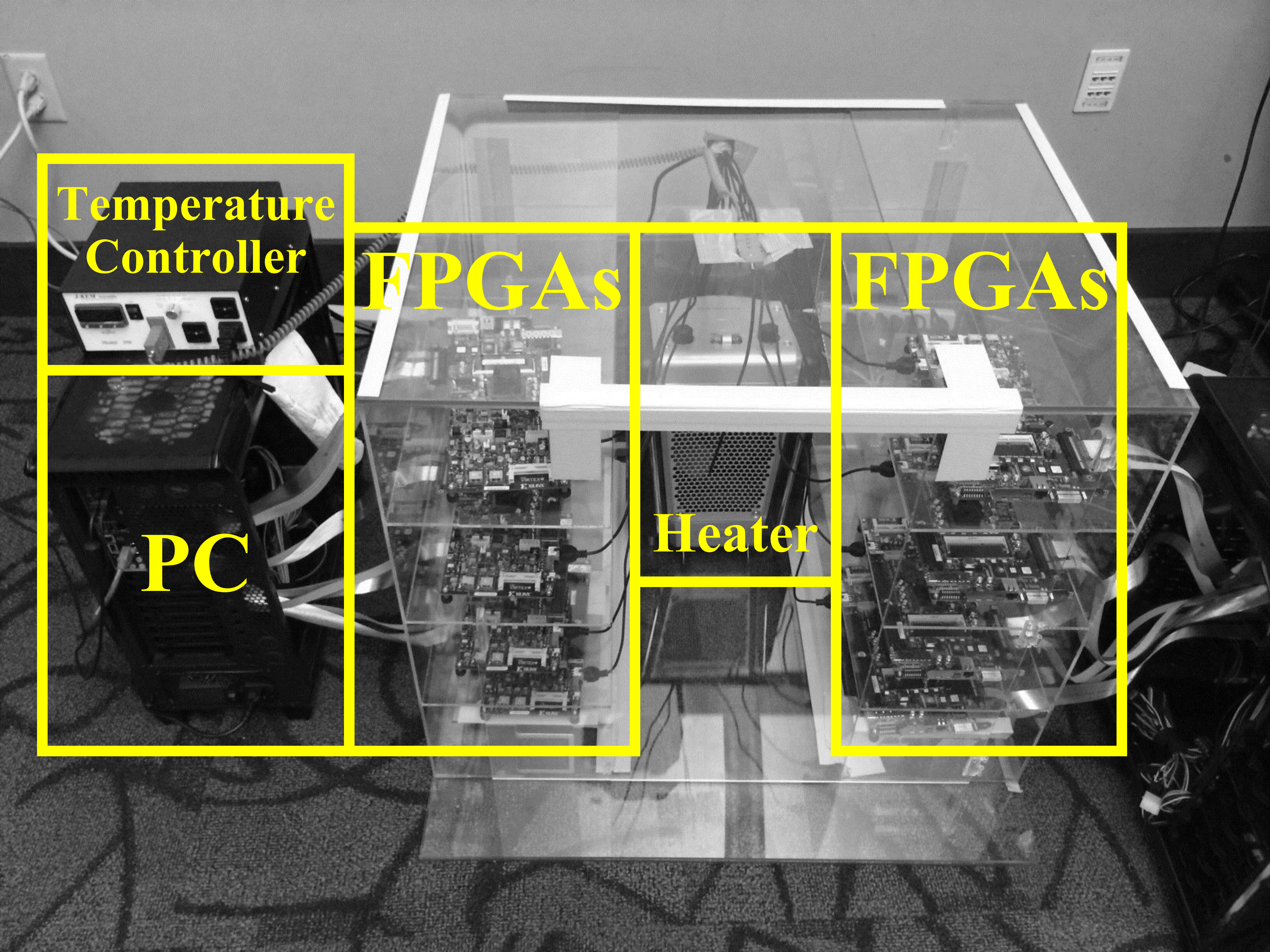
FPGA

PCIe

Test Engine

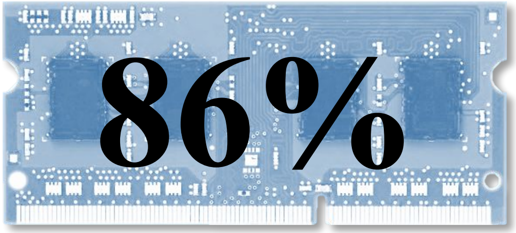
DRAM Ctrl

DRAM


Temperature Controller
FPGAs
FPGAs
Heater
PC

MOST MODULES AT RISK

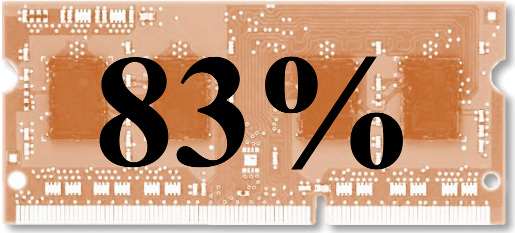
A VENDOR



86%

(37/43)

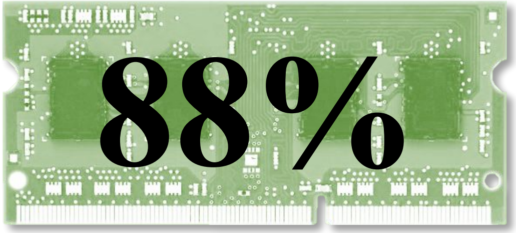
B VENDOR



83%

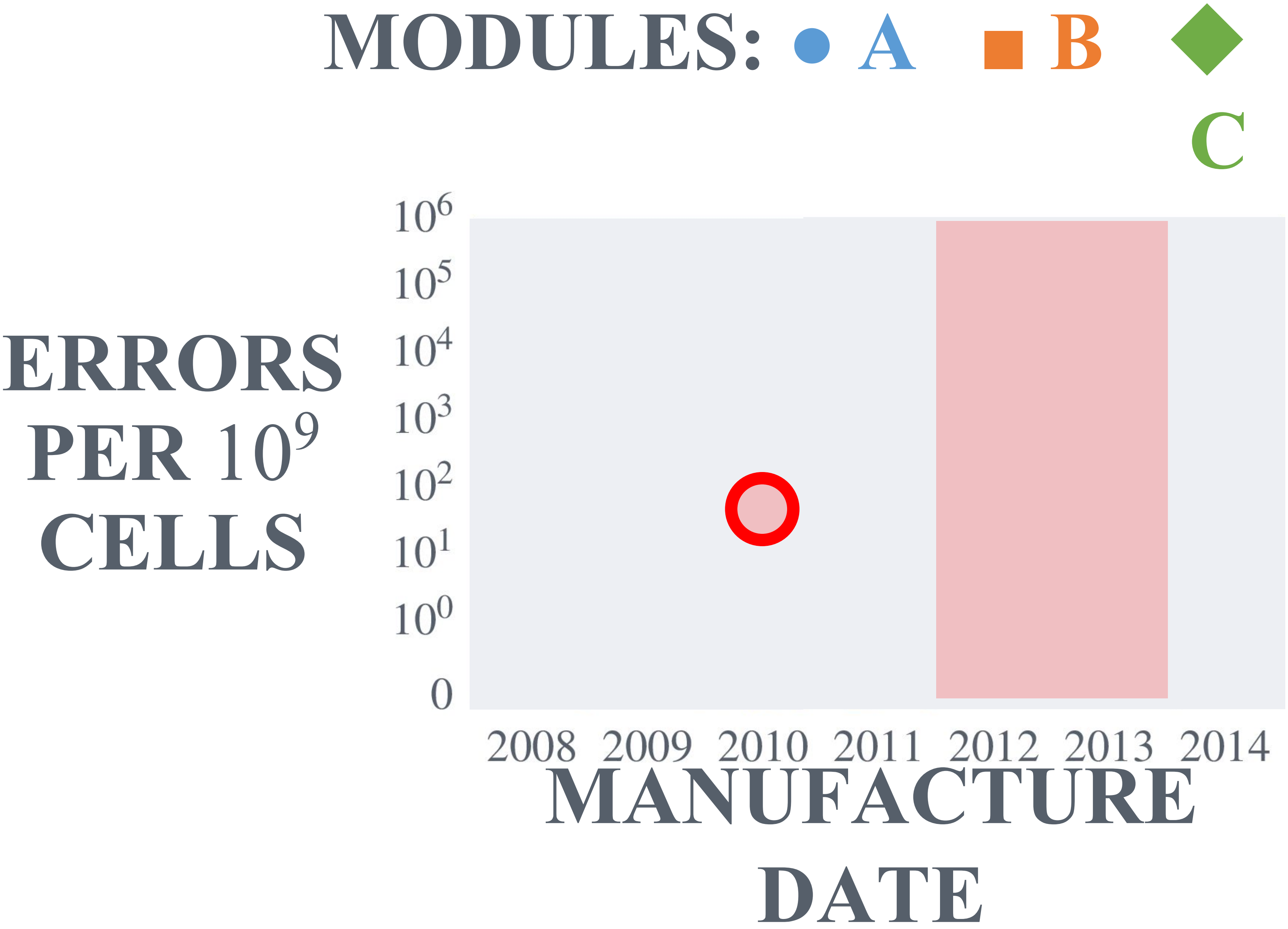
(45/54)

C VENDOR



88%

(28/32)


MODULES: ● A ■ B ◆ C
ERRORS PER 10^9 CELLS
10^6
10^5
10^4
10^3
10^2
10^1
10^0
0
2008
2009
2010
2011
2012
2013
2014
MANUFACTURE DATE

DISTURBING FACTS

- **AFFECTS ALL VENDORS**

  Not an isolated incident
  Deeper issue in DRAM scaling

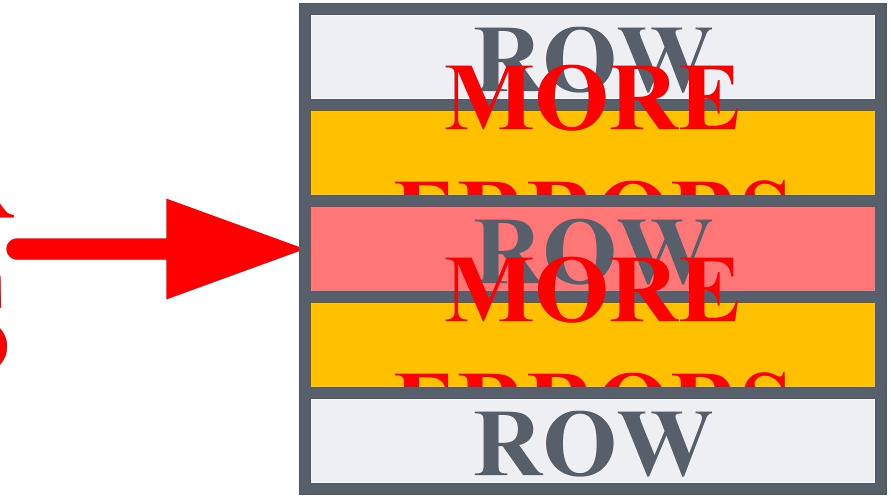
- **UNADDRESSED FOR YEARS**

HOW TO PREVENT COUPLING ERRORS?

Previous Approaches

1. Make Better Chips: Expensive
2. Rigorous Testing: Takes Too Long

FASTER ACCESS



ROW

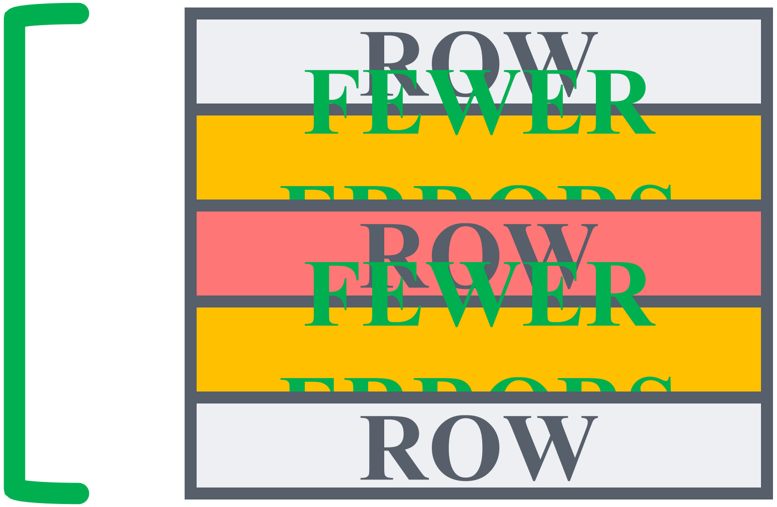
MORE ERRORS

ROW

MORE ERRORS

ROW

FREQUENT REFRESH



ROW

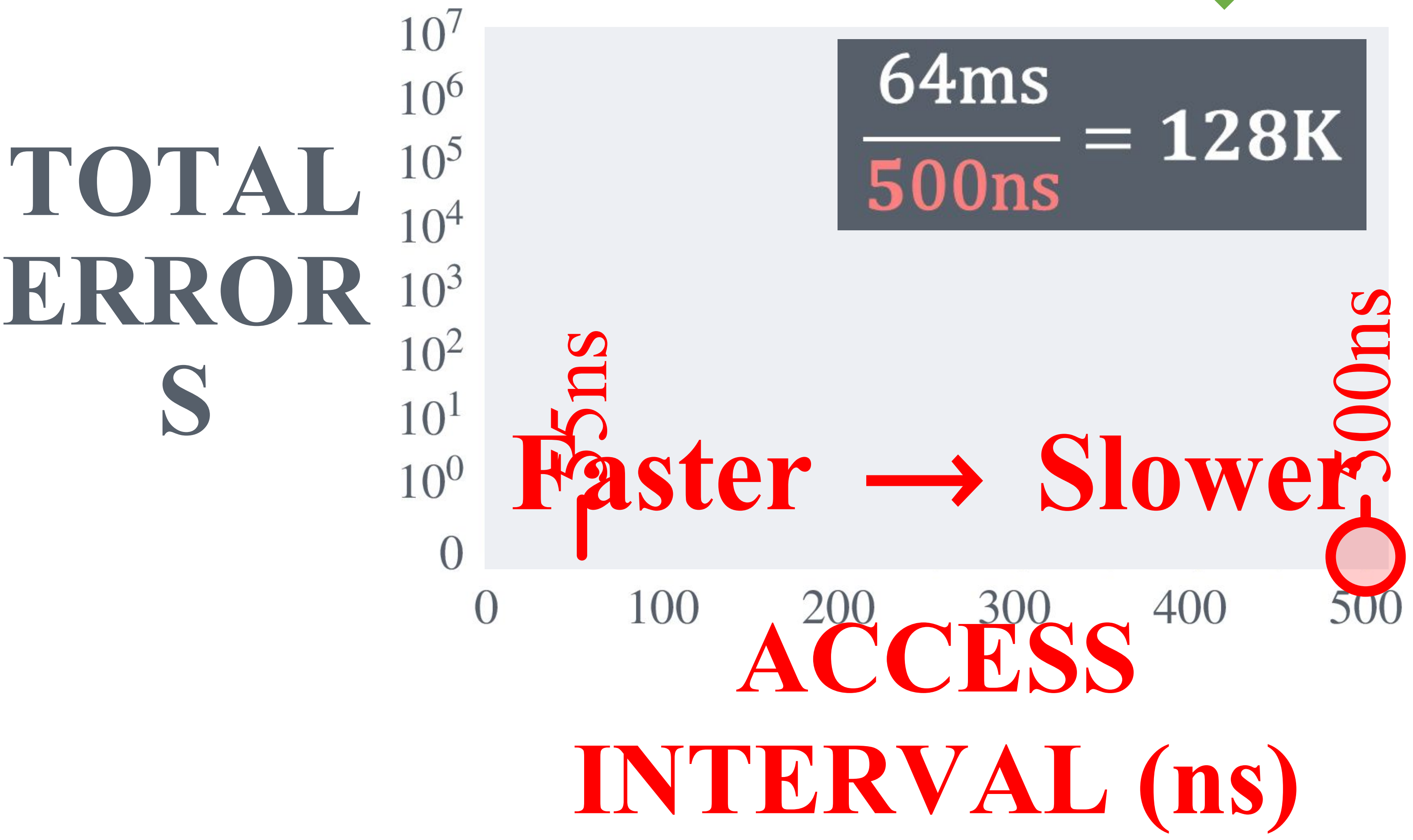
FEWER ERRORS

ROW

FEWER ERRORS

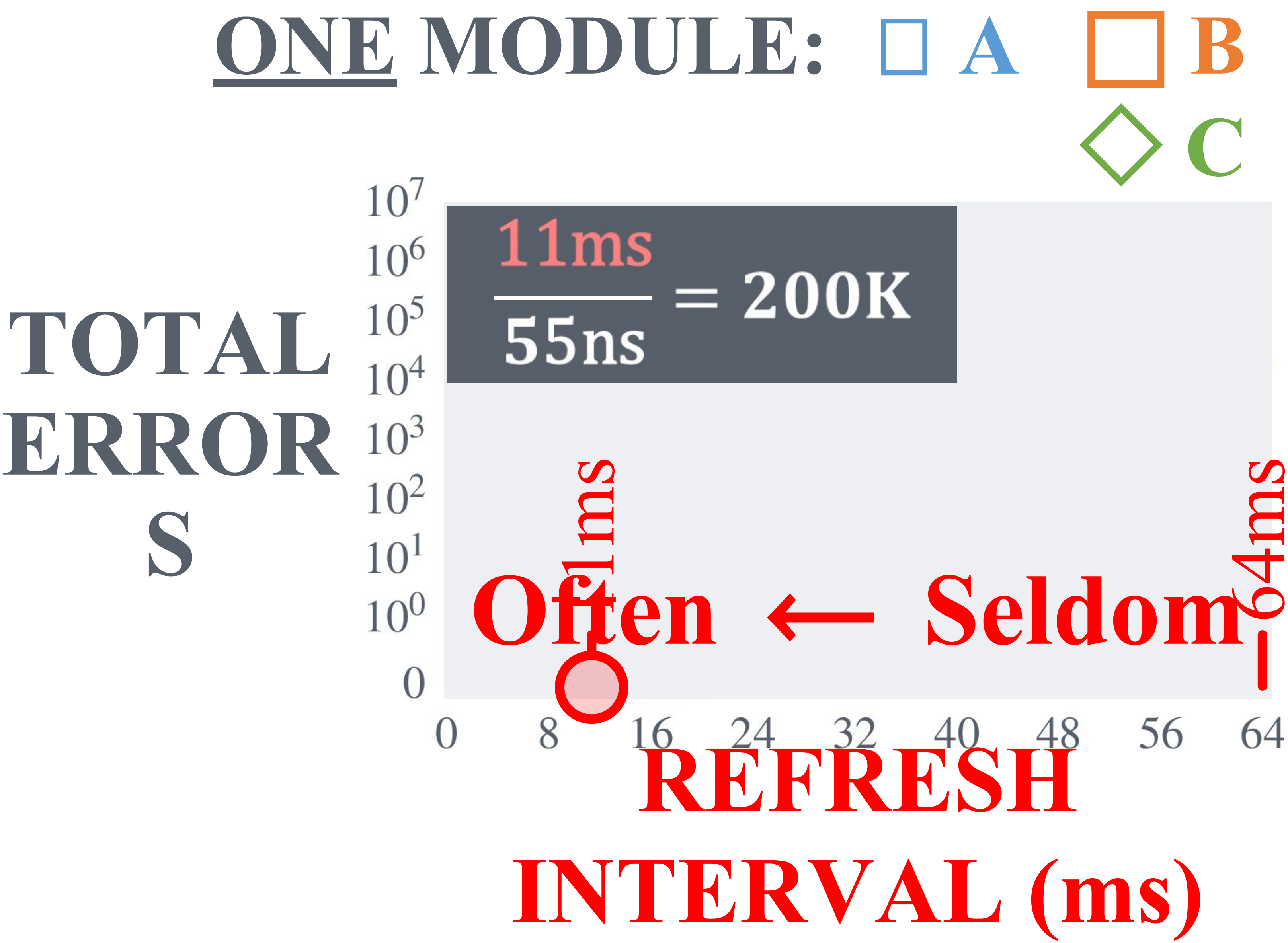
ROW

ONE MODULE: □ A □ B ◇ C



$$\frac{64\text{ms}}{500\text{ns}} = 128\text{K}$$

ONE MODULE:



$$\frac{11ms}{55ns} = 200K$$

TOTAL ERRORS

Often ← Seldom

REFRESH INTERVAL (ms)

TWO NAIVE SOLUTIONS

1. **LIMIT ACCESSES TO ROW**

   Access Interval > 500ns

2. **REFRESH ALL ROWS OFTEN**

   Refresh Interval < 11ms

LARGE OVERHEAD: PERF, ENERGY, COMPLEXITY

OUR SOLUTION: PARR

Probabilistic Adjacent Row Refresh

After closing any row

99.9%

0.1%

Do nothing

Refresh (=Open) adjacent rows

PARR: CHANCE OF ERROR

- **NO REFRESHES IN N TRIALS**

  Probability: 0.999^{N}

- **N=128K FOR ERROR (64ms)**

  Probability: $0.999^{128K} = \mathbf{10^{-56}}$

STRONG RELIABILITY

9.4×10^{-14}
Errors/Year

LOW PERF OVERHEAD

0.20%
Slowdown

NO STORAGE OVERHEAD

0 Bytes

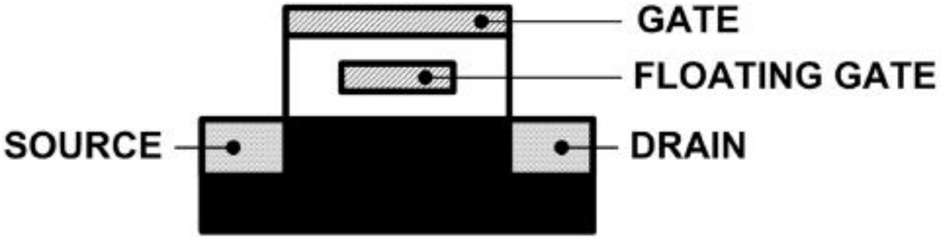
RELATED WORK

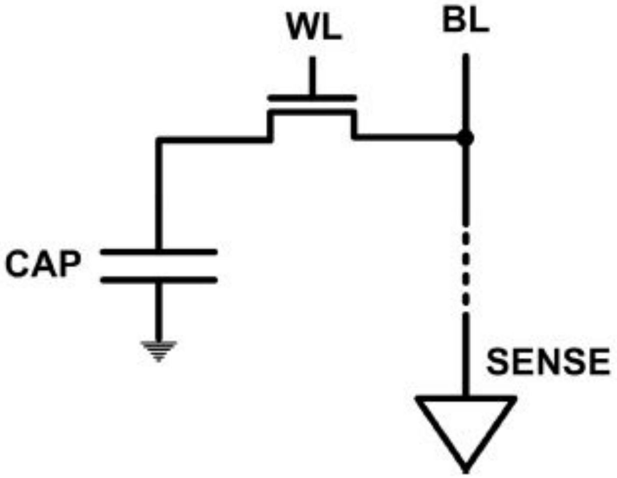
- **Security Exploit** **(Seaborn@Google 2015)**
- **Industry Analysis** **(Kang@SK Hynix 2014)**
  "… will be [more] severe as technology shrinks down."
- **Targeted Row Refresh** **(JEDEC 2014)**
- **DRAM Testing** **(e.g., Van de Goor+**

Emerging Memory Technologies

Limits of Charge Memory

- Difficult charge placement and control
  - Flash: floating gate charge
  - DRAM: capacitor charge, transistor leakage
- Reliable sensing becomes difficult as charge storage unit size reduces


GATE
FLOATING GATE
SOURCE
DRAIN


WL
BL
CAP
SENSE

Charge vs. Resistive Memories

- Charge Memory (e.g., DRAM, Flash)
  - Write data by capturing charge Q
  - Read data by detecting voltage V

- Resistive Memory (e.g., PCM, STT-MRAM, memristors)
  - Write data by pulsing current dQ/dt
  - Read data by detecting resistance R

Promising Resistive Memory Technologies

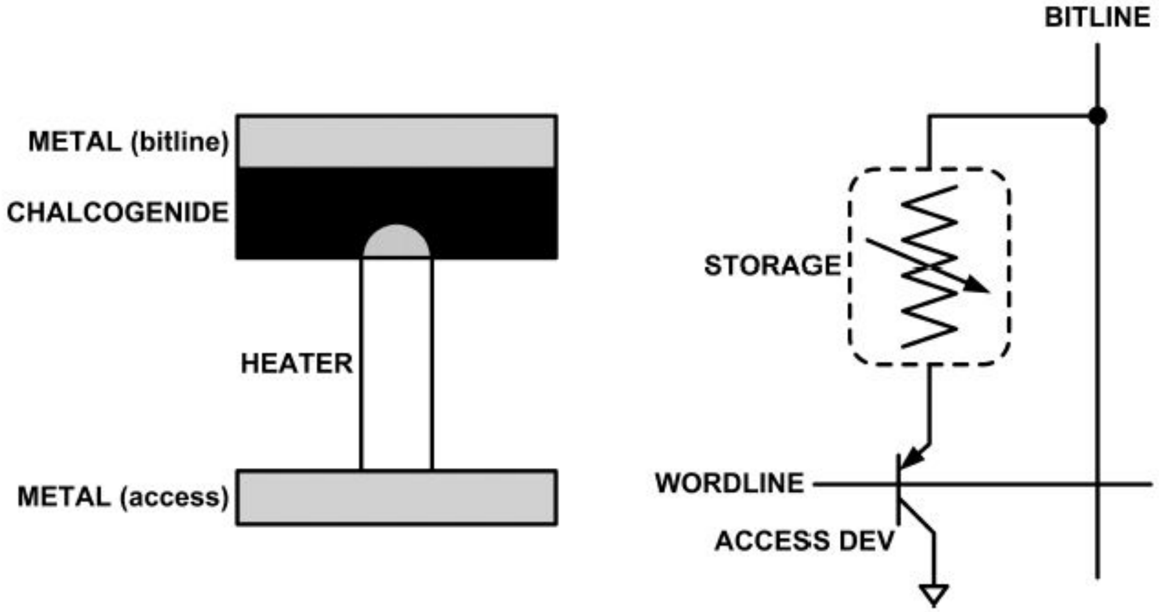
- PCM
  - Inject current to change material phase
  - Resistance determined by phase

- STT-MRAM
  - Inject current to change magnet polarity
  - Resistance determined by polarity

- Memristors/RRAM/ReRAM
  - Inject current to change atomic structure
  - Resistance determined by atom distance

What is Phase Change Memory?

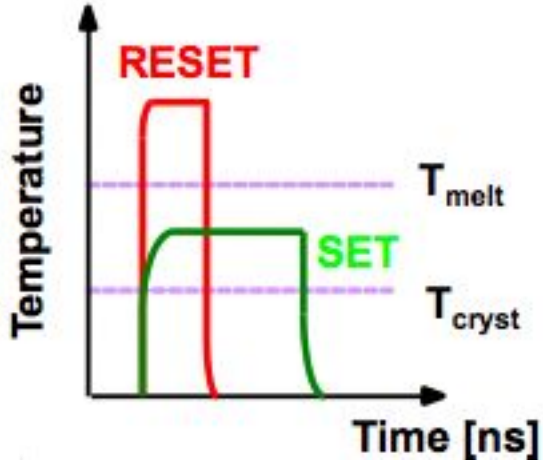
- Phase change material (chalcogenide glass) exists in two states:
  - Amorphous: Low optical reflexivity and high electrical resistivity
  - Crystalline: High optical reflexivity and low electrical resistivity

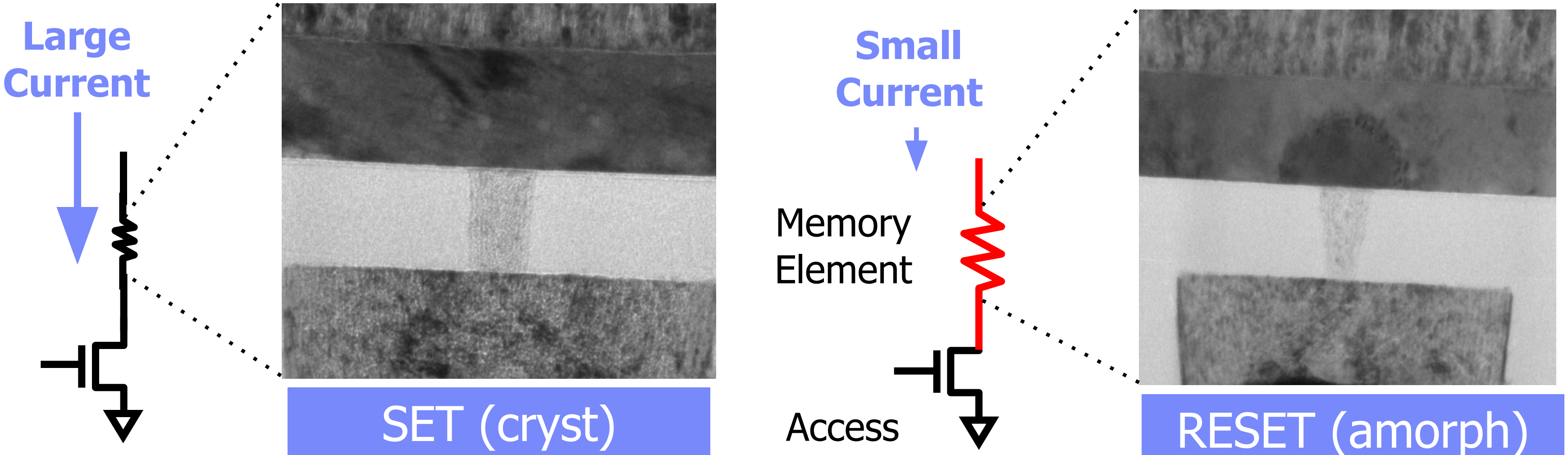

METAL (bitline)
CHALCOGENIDE
HEATER
METAL (access)
BITLINE
STORAGE
WORDLINE
ACCESS DEV

PCM is resistive memory: High resistance (0), Low resistance (1)
PCM cell can be switched between states reliably and quickly

How Does PCM Work?

- Write: change phase via current injection
  - SET: sustained current to heat cell above T*cryst*
  - RESET: cell heated above T*melt* and quenched
- Read: detect phase via material resistance
  - amorphous/crystalline




Large Current
Small Current
Memory Element
Access Device

SET (cryst)
Low resistance

10^3-10^4 Ω

RESET (amorph)
High resistance

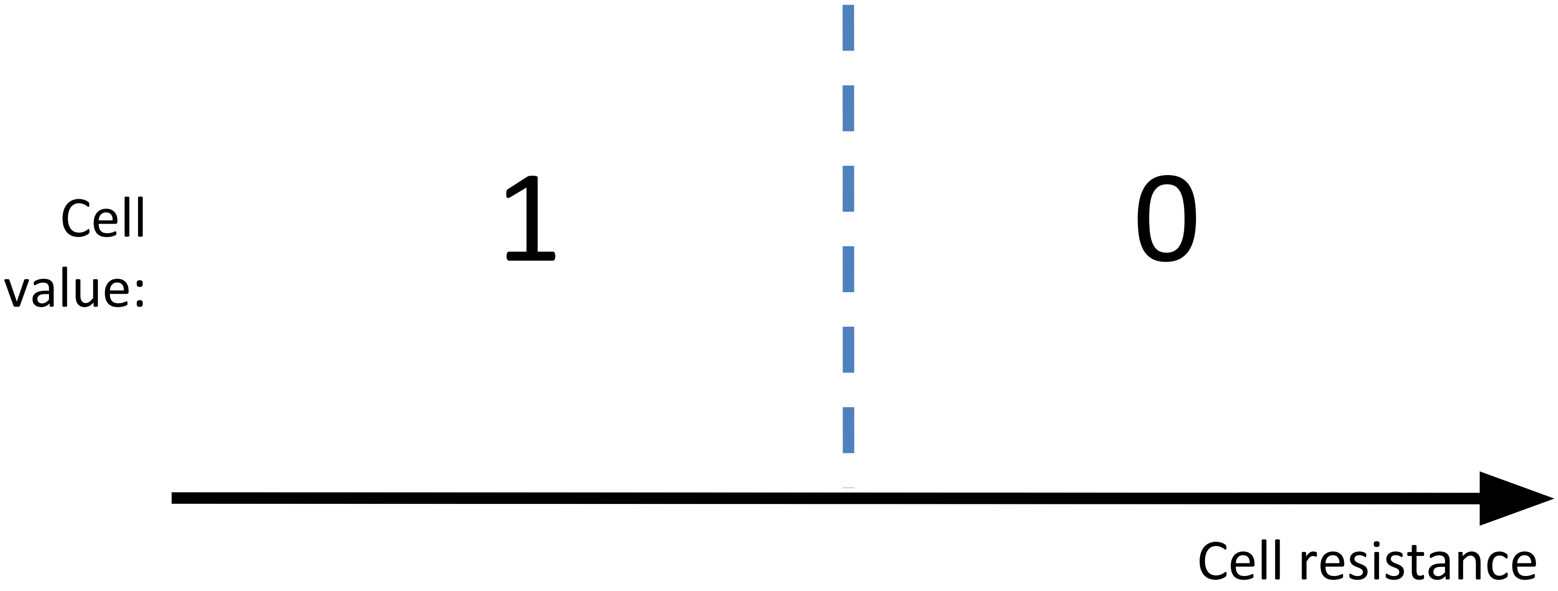
10^6-10^7 Ω

Photo Courtesy: Bipin Rajendran, IBM Slide Courtesy: Moinuddin Qureshi, IBM

Opportunity: PCM Advantages

- Scales better than DRAM, Flash
  - Requires current pulses, which scale linearly with feature size
  - Expected to scale to 9nm (2022 [ITRS])
  - Prototyped at 20nm (Raoux+, IBM JRD 2008)
- Can be denser than DRAM
  - Can store multiple bits per cell due to large resistance range
  - Prototypes with 2 bits/cell in ISSCC'08, 4 bits/cell by 2012
- Non-volatile
  - Retain data for >10 years at 85C
- No refresh needed, low idle power

PCM Resistance → Value



Cell value:

1

0

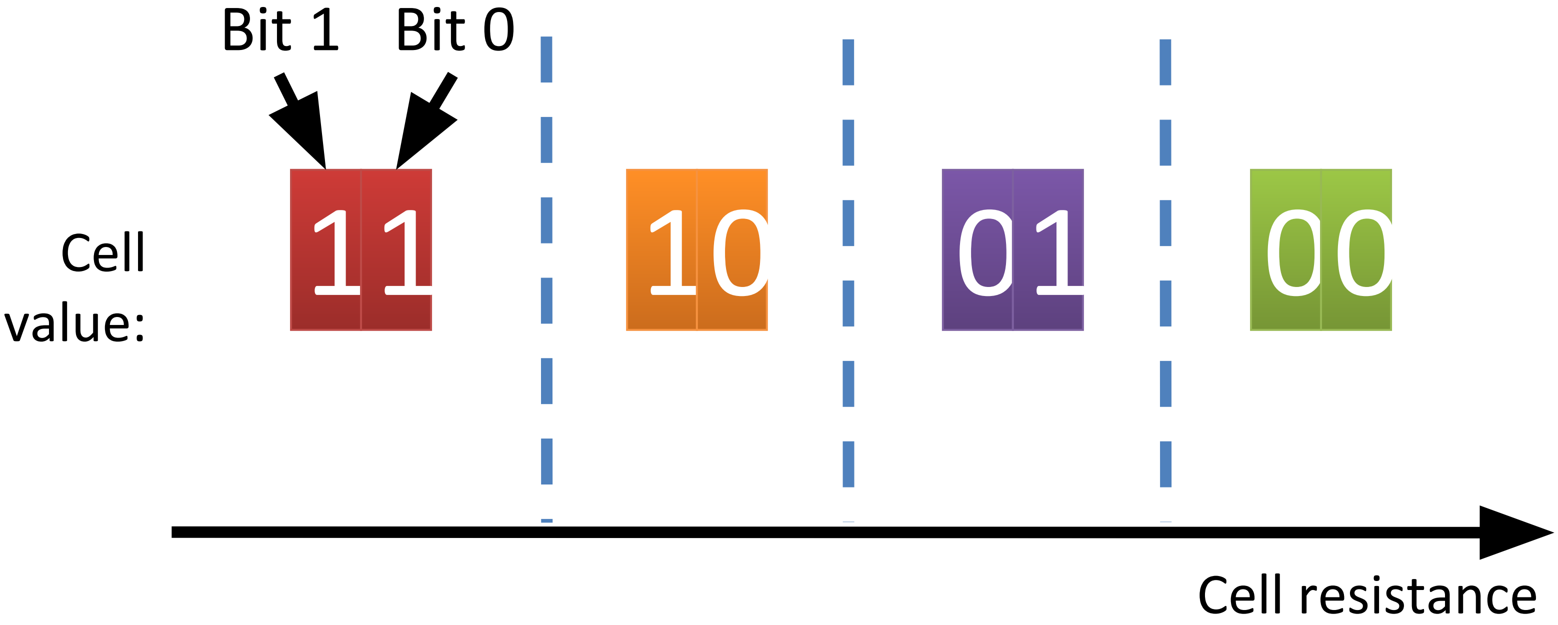
Cell resistance

Multi-Level Cell PCM

- Multi-level cell: more than 1 bit per cell
  - Further increases density by 2 to 4x [Lee+,ISCA'09]

- But MLC-PCM also has drawbacks
  - Higher latency and energy than single-level cell PCM

MLC-PCM Resistance → Value

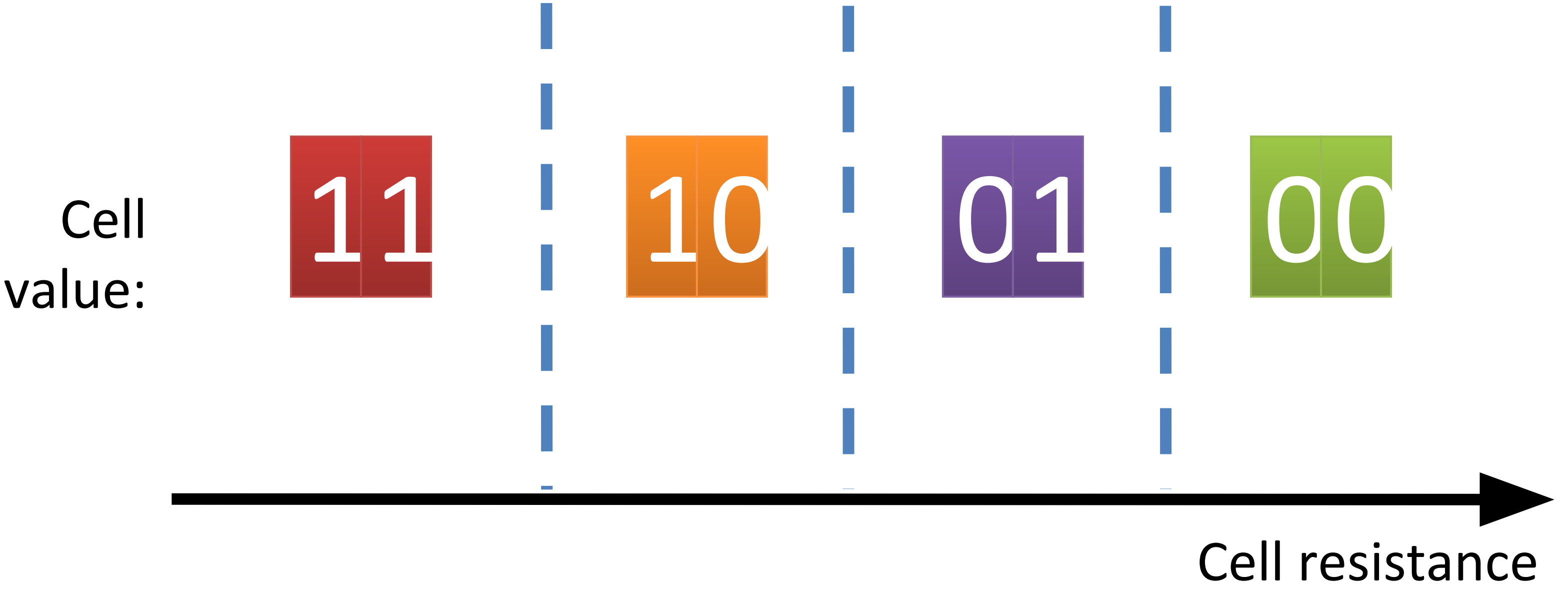

Bit 1
Bit 0
Cell value:
11
10
01
00
Cell resistance

MLC-PCM Resistance → Value

Less margin between values

→ need more precise sensing/modification of cell contents

→ higher latency/energy (~2x for reads and 4x for writes)



Cell value:

11

10

01

00

Cell resistance

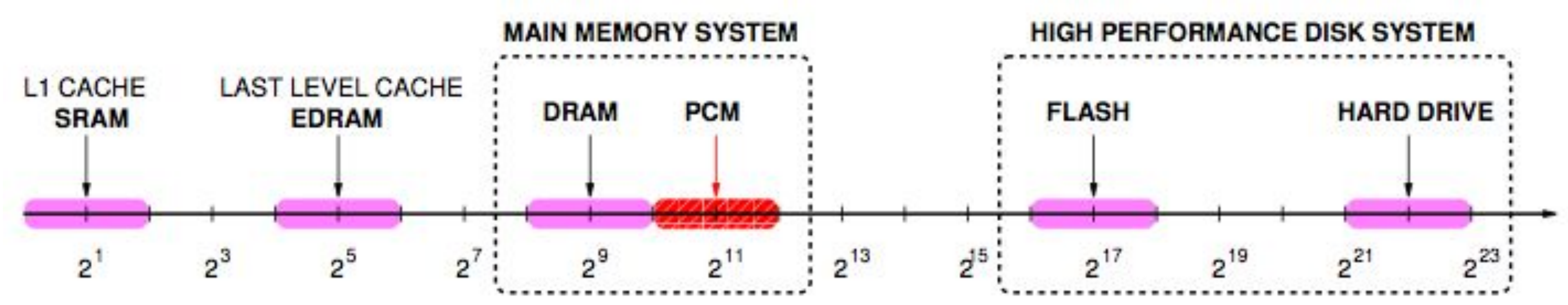
Phase Change Memory Properties

- Surveyed prototypes from 2003-2008 (ITRS, IEDM, VLSI, ISSCC)
- Derived PCM parameters for F=90nm

- Lee, Ipek, Mutlu, Burger, "Architecting Phase Change Memory as a Scalable DRAM Alternative," ISCA 2009.
- Lee et al., "Phase Change Technology and the Future of Main Memory," IEEE Micro Top Picks 2010.

Phase Change Memory Properties: Latency

- Latency comparable to, but slower than DRAM



Typical Access Latency (in terms of processor cycles for a 4 GHz processor)

- Read Latency
  - 50ns: 4x DRAM, 10^{-3}x NAND Flash
- Write Latency
  - 150ns: 12x DRAM
- Write Bandwidth
  - 5-10 MB/s: 0.1x DRAM, 1x NAND Flash

Qureshi+, “Scalable high performance main memory system using phase-change memory technology,” ISCA 2009.

Phase Change Memory Properties

- Dynamic Energy
  - 40 uA Rd, 150 uA Wr
  - 2-43x DRAM, 1x NAND Flash
- Endurance
  - Writes induce phase change at 650C
  - Contacts degrade from thermal expansion/contraction
  - 10^8 writes per cell
  - 10^{-8}x DRAM, 10^3x NAND Flash
- Cell Size
  - 9-12F^2 using BJT, single-level cells
  - 1.5x DRAM, 2-3x NAND (will scale with feature size, MLC)

Phase Change Memory: Pros and Cons

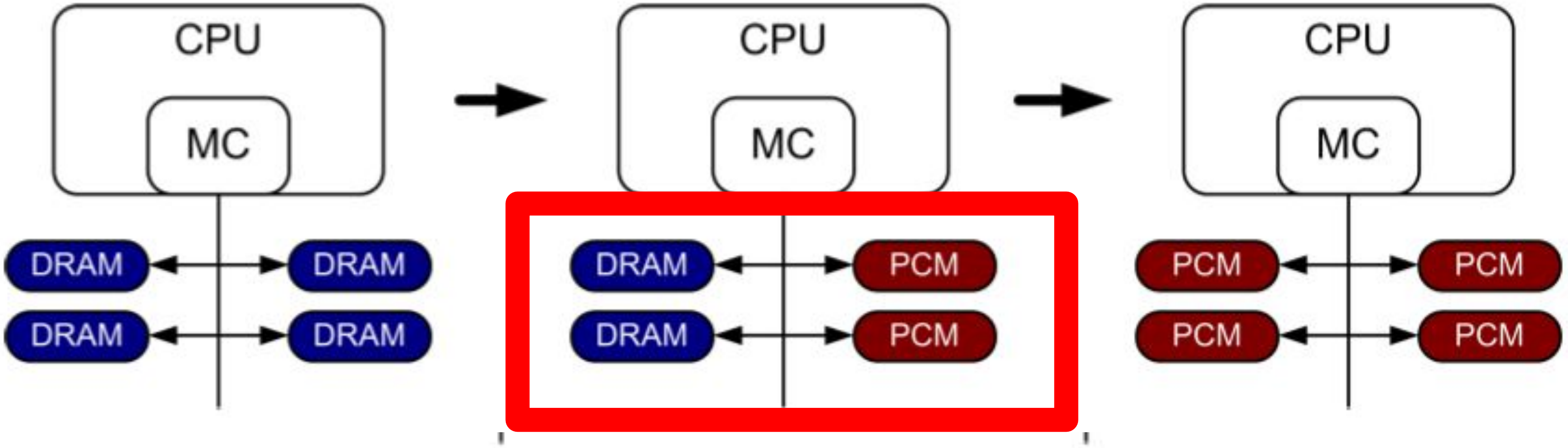
- Pros over DRAM
  - Better technology scaling (capacity and cost)
  - Non volatile ☐ Persistent
  - Low idle power (no refresh)

- Cons
  - Higher latencies: ~4-15x DRAM (especially write)
  - Higher active energy: ~2-50x DRAM (especially write)
  - Lower endurance (a cell dies after ~10^8 writes)
  - Reliability issues (resistance drift)

- Challenges in enabling PCM as DRAM replacement/helper:
  - Mitigate PCM shortcomings
  - Find the right way to place PCM in the system

PCM-based Main Memory (I)

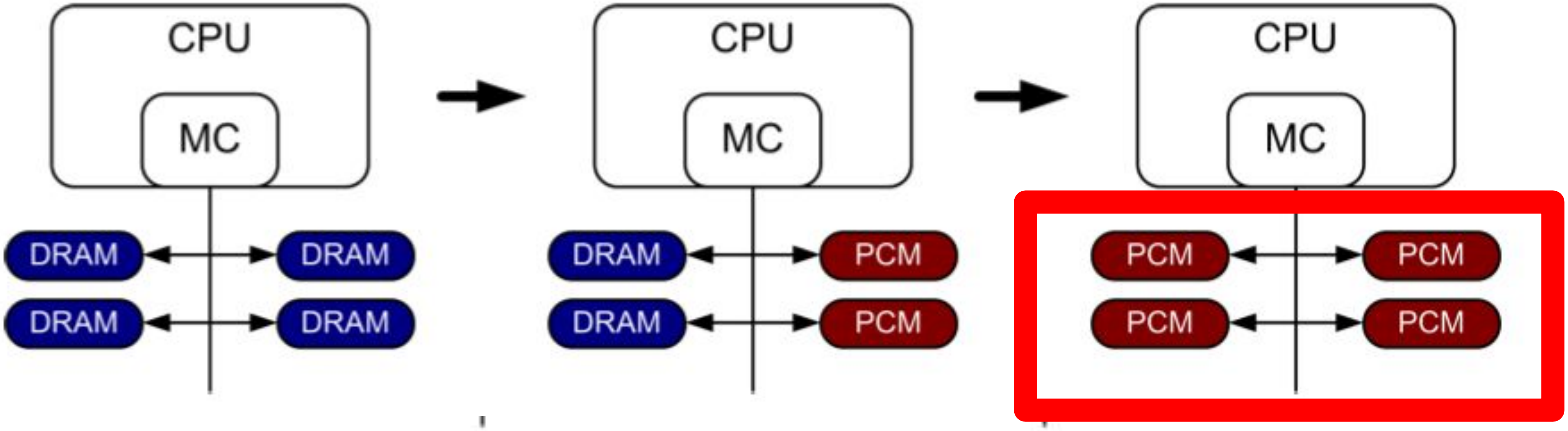
- How should PCM-based (main) memory be organized?


CPU
MC
DRAM
DRAM
DRAM
DRAM
CPU
MC
DRAM
PCM
DRAM
PCM
CPU
MC
PCM
PCM
PCM
PCM

- Hybrid PCM+DRAM [Qureshi+ ISCA'09, Dhiman+ DAC'09]:
  - How to partition/migrate data between PCM and DRAM

PCM-based Main Memory (II)

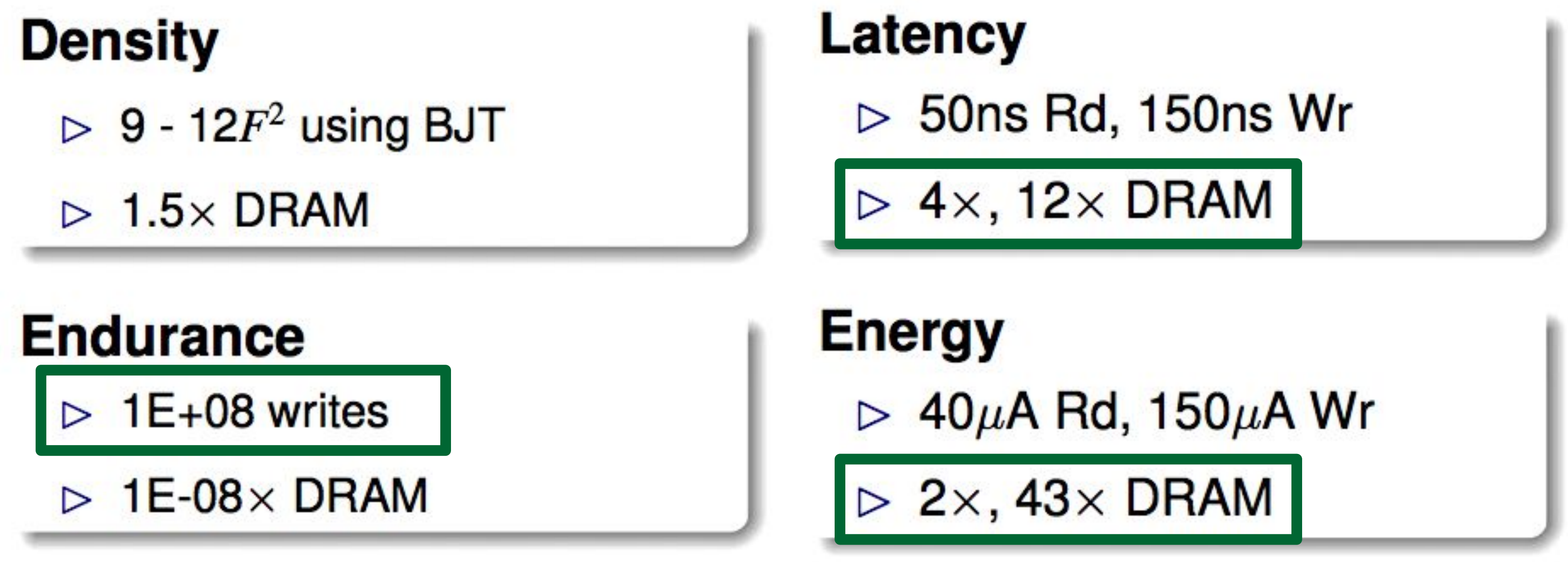
- How should PCM-based (main) memory be organized?


CPU
MC
DRAM
DRAM
DRAM
DRAM
CPU
MC
DRAM
PCM
DRAM
PCM
CPU
MC
PCM
PCM
PCM
PCM

- Pure PCM main memory [Lee et al., ISCA'09, Top Picks'10]:
  - How to redesign entire hierarchy (and cores) to overcome PCM shortcomings

An Initial Study: Replace DRAM with PCM

- Lee, Ipek, Mutlu, Burger, "Architecting Phase Change Memory as a Scalable DRAM Alternative," ISCA 2009.
  - Surveyed prototypes from 2003-2008 (e.g. IEDM, VLSI, ISSCC)
  - Derived "average" PCM parameters for F=90nm



Density

▷ 9 - $12F^2$ using BJT

▷ 1.5× DRAM

Latency

▷ 50ns Rd, 150ns Wr

▷ 4×, 12× DRAM

Endurance

▷ 1E+08 writes

▷ 1E-08× DRAM

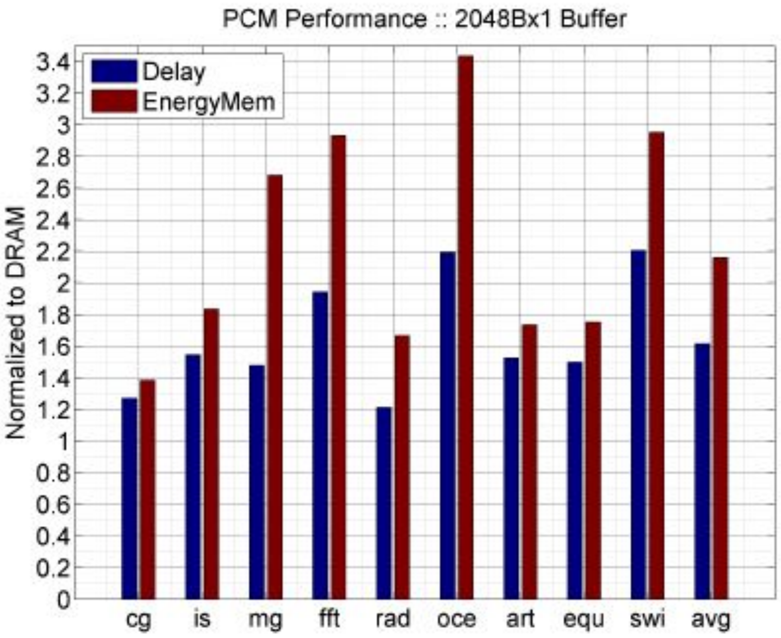
Energy

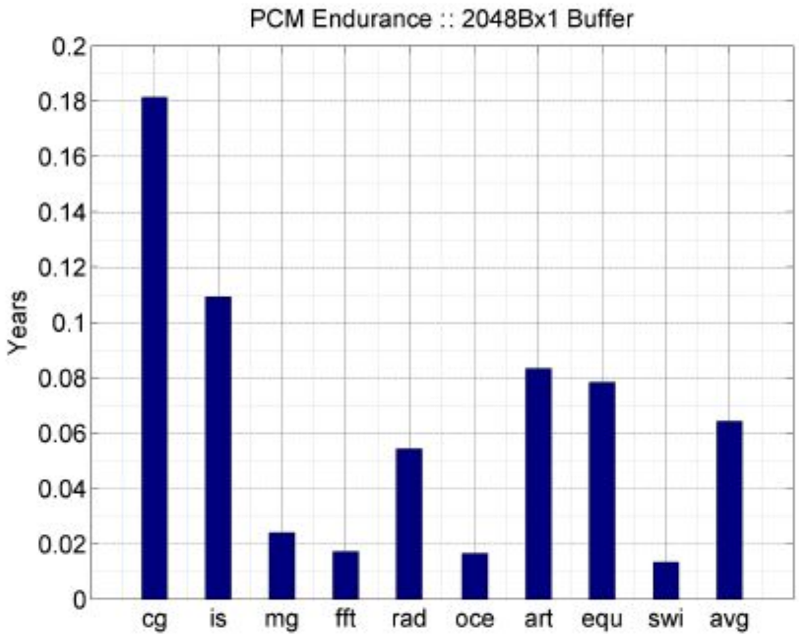
▷ 40μA Rd, 150μA Wr

▷ 2×, 43× DRAM

Results: Naïve Replacement of DRAM with PCM

- Replace DRAM with PCM in a 4-core, 4MB L2 system
- PCM organized the same as DRAM: row buffers, banks, peripherals
- 1.6x delay, 2.2x energy, 500-hour average lifetime



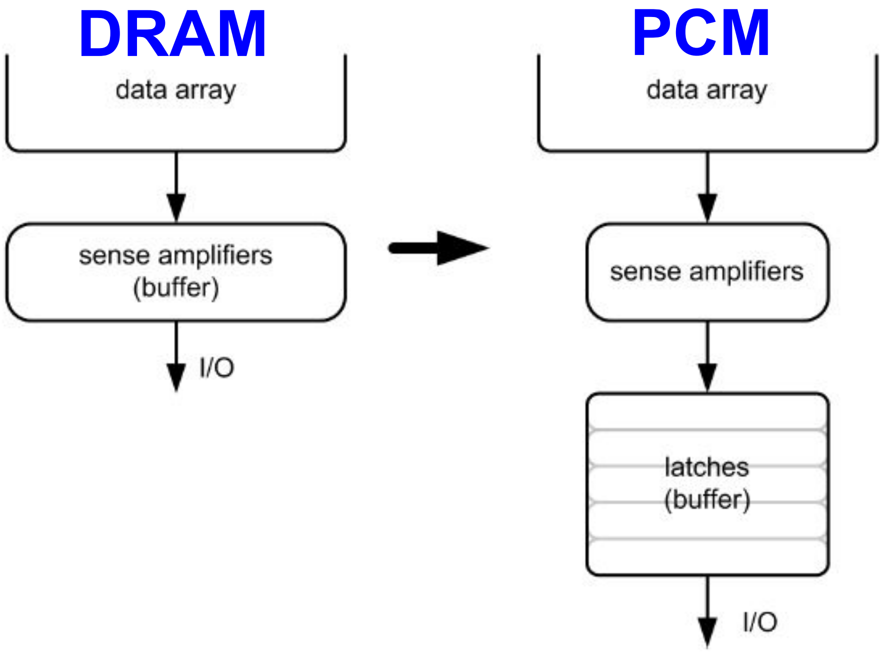


- Lee, Ipek, Mutlu, Burger, "Architecting Phase Change Memory as a Scalable DRAM Alternative," ISCA 2009.

Architecting PCM to Mitigate Shortcomings

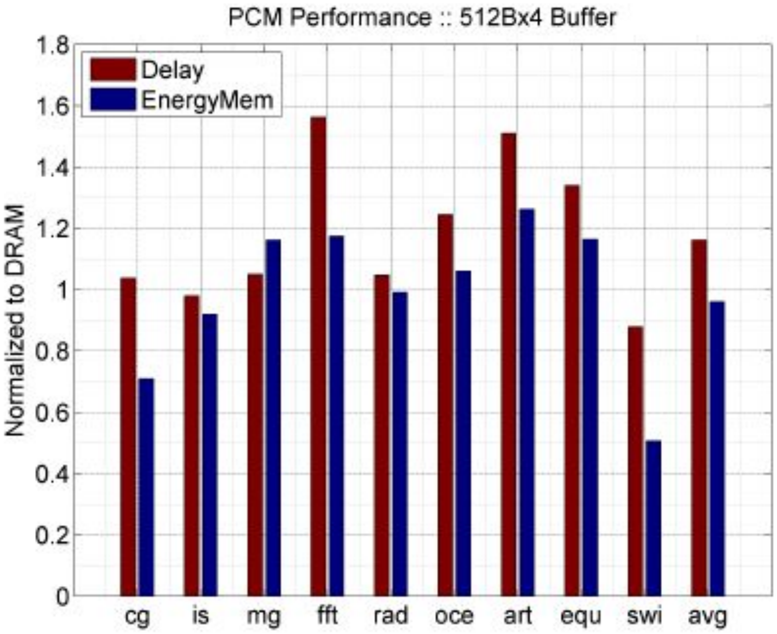
- Idea 1: Use multiple narrow row buffers in each PCM chip
  - ☐ Reduces array reads/writes ☐ better endurance, latency, energy

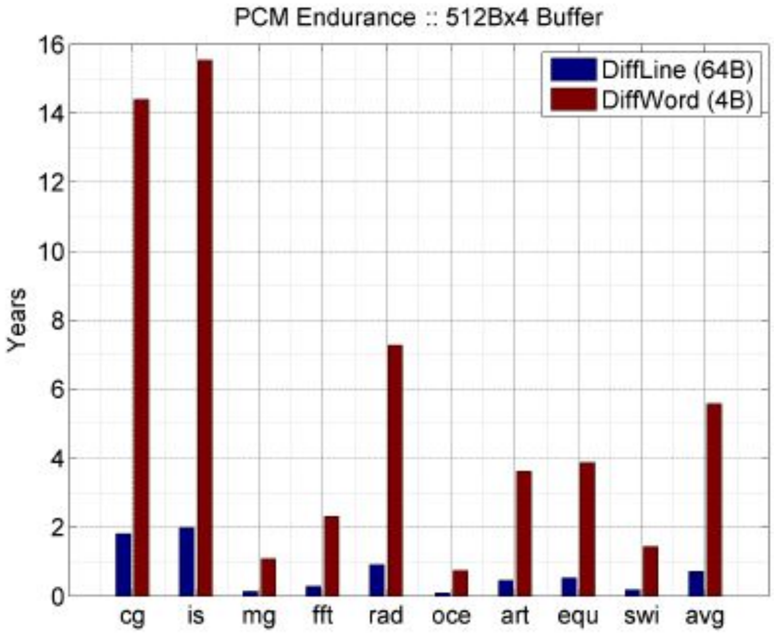
- Idea 2: Write into array at cache block or word granularity
  - ☐ Reduces unnecessary wear


DRAM
data array
sense amplifiers
(buffer)
I/O
PCM
data array
sense amplifiers
latches
(buffer)
I/O

Results: Architected PCM as Main Memory

- 1.2x delay, 1.0x energy, 5.6-year average lifetime
- Scaling improves energy, endurance, density

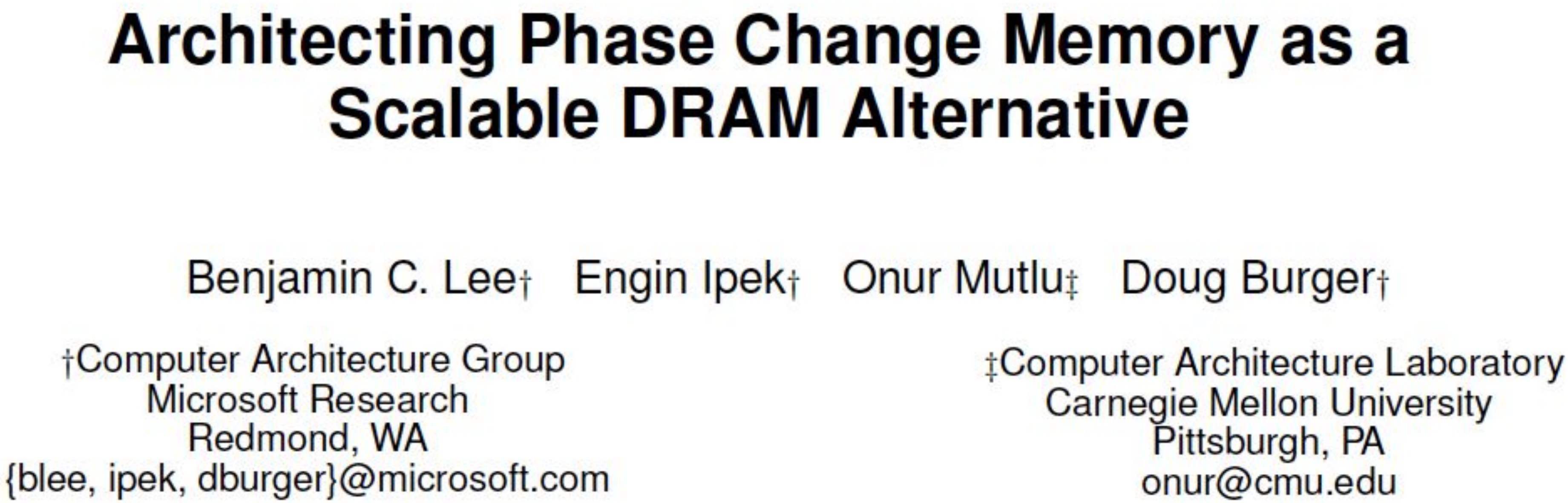




- Caveat 1: Worst-case lifetime is much shorter (no guarantees)
- Caveat 2: Intensive applications see large performance and energy hits
- Caveat 3: Optimistic PCM parameters?

PCM As Main Memory

- Benjamin C. Lee, Engin Ipek, Onur Mutlu, and Doug Burger, **"Architecting Phase Change Memory as a Scalable DRAM Alternative"** *Proceedings of the 36th International Symposium on Computer Architecture* (***ISCA***), pages 2-13, Austin, TX, June 2009. Slides (pdf)



Architecting Phase Change Memory as a Scalable DRAM Alternative

Benjamin C. Lee[†] Engin Ipek[†] Onur Mutlu[‡] Doug Burger[†]

[†]Computer Architecture Group
Microsoft Research
Redmond, WA
{blee, ipek, dburger}@microsoft.com

[‡]Computer Architecture Laboratory
Carnegie Mellon University
Pittsburgh, PA
onur@cmu.edu

Review #5

Flipping Bits in Memory Without Accessing Them

Yoongu Kim et al., *ISCA 2014*

CSC 2224: Parallel Computer Architecture and Programming
Advanced Memory

Prof. Gennady Pekhimenko

University of Toronto

Fall 2019

The content of this lecture is adapted from the slides of
Vivek Seshadri, Yoongu Kim,
and lectures of Onur Mutlu @ ETH and CMU