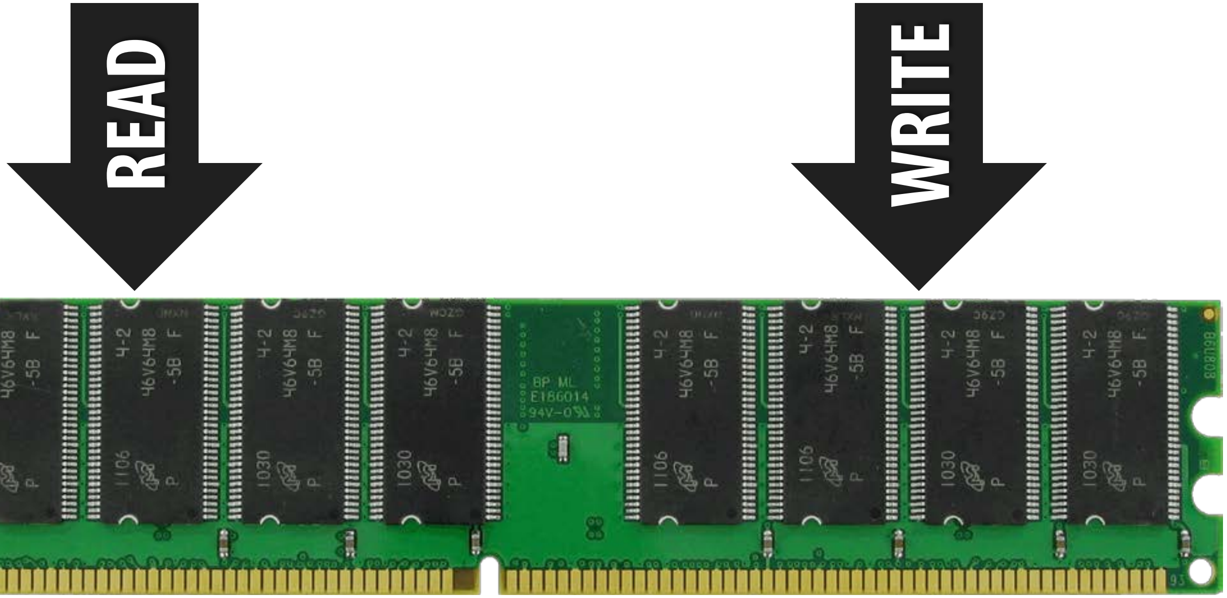
Can DRAM Do More Than Just Store Data?

Vivek Seshadri
Final Year Ph.D. Student
Onur Mutlu, Todd C. Mowry
Phillip B. Gibbons, Michael A. Kozuch

http://www.istc-cc.cmu.edu/

Intel Science & Technology
Center for Cloud Computing

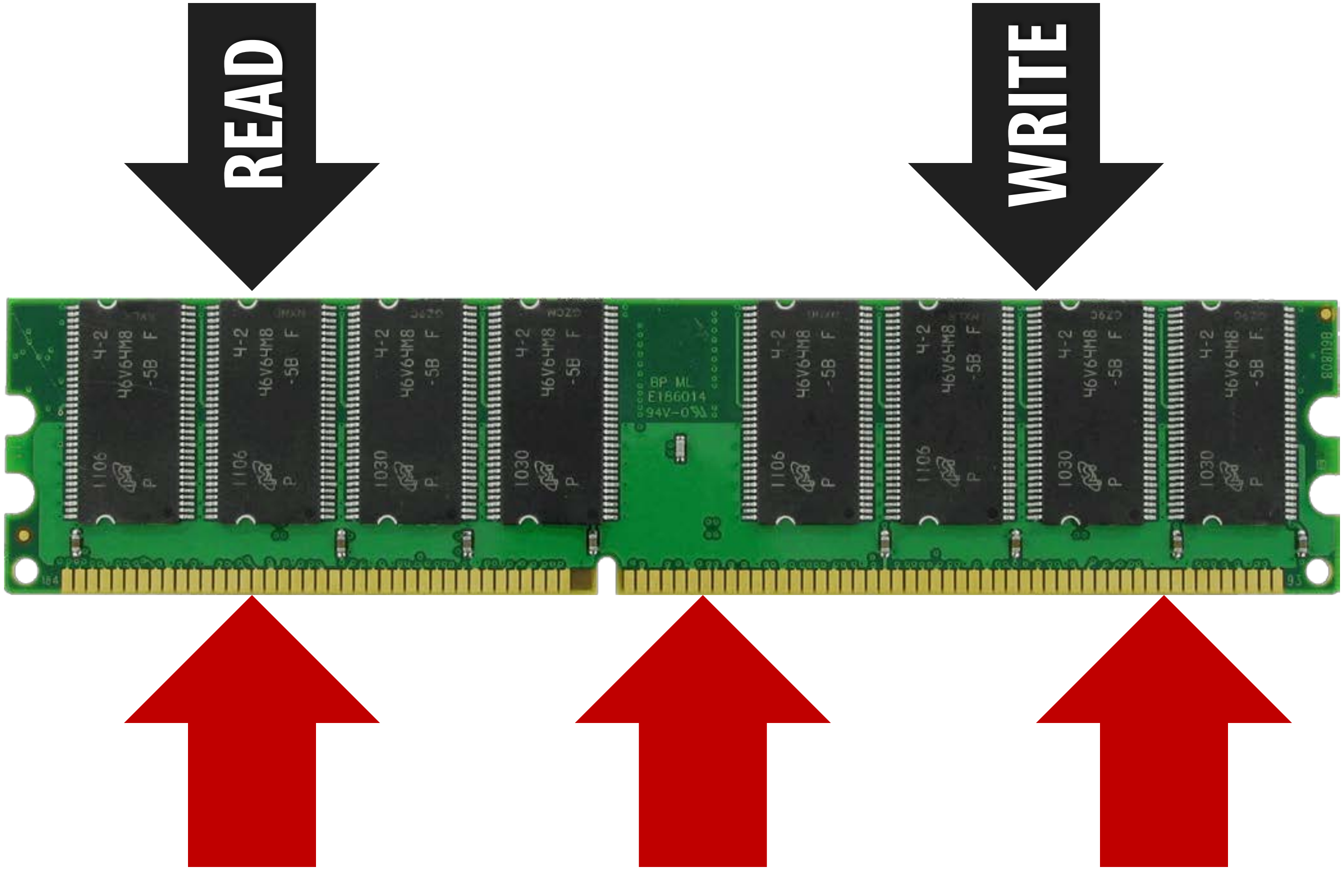


READ

WRITE

Can DRAM do more?

Yes!



Copy/Init

Gather/Scatter
strided access patterns

Bitwise AND/OR

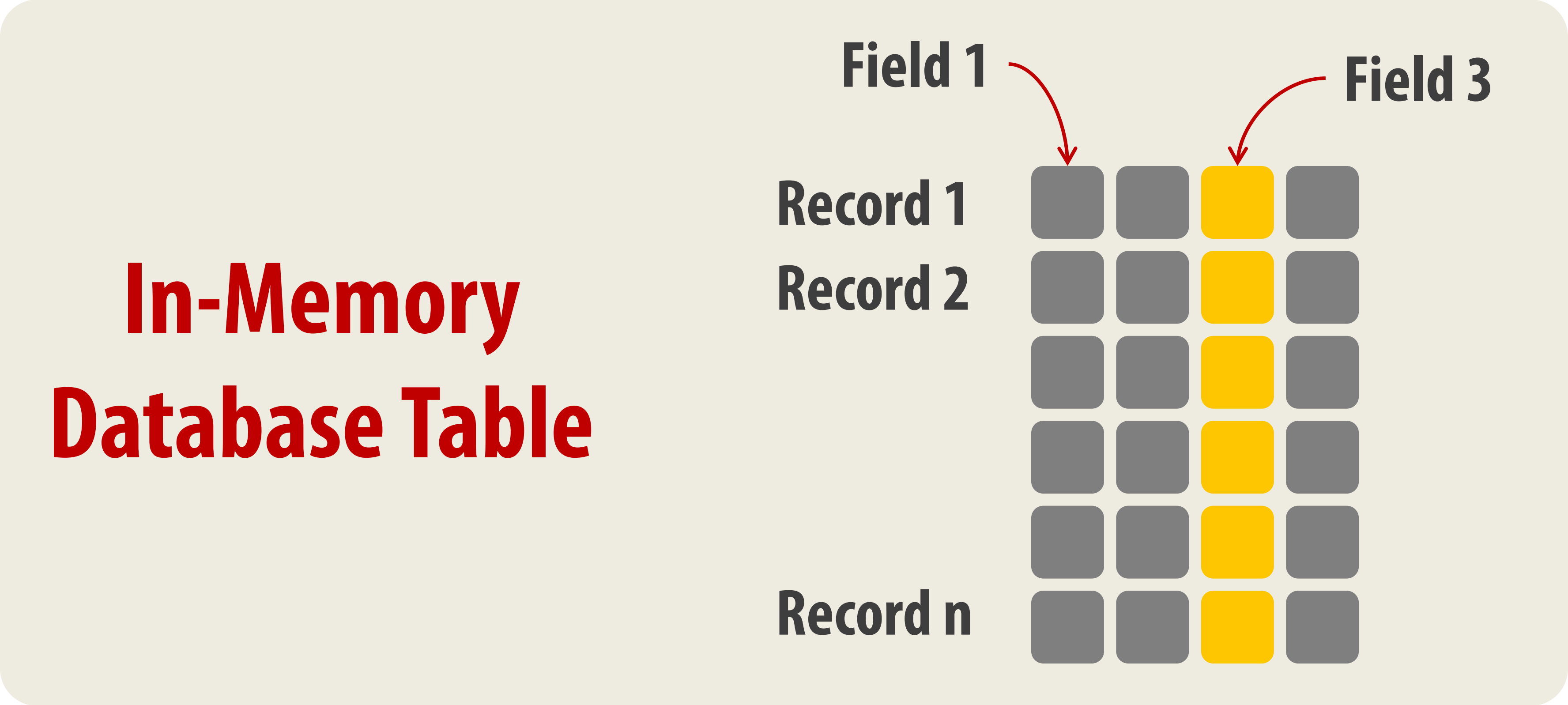
Outline of the Talk

1. Gather-Scatter DRAM
   – accelerating strided access patterns

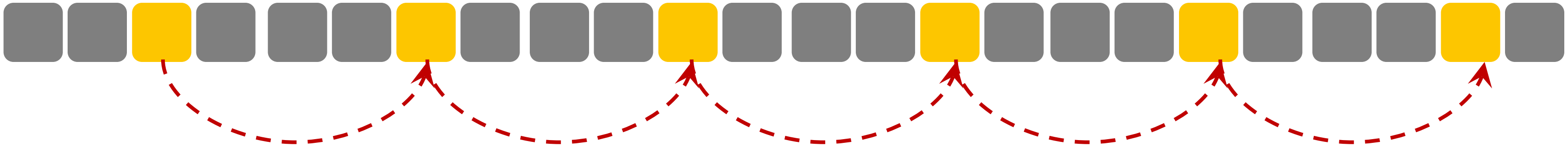
2. RowClone
   – bulk data copy/initialization in DRAM

3. Bulk bitwise AND/OR in DRAM

Strided Access Pattern



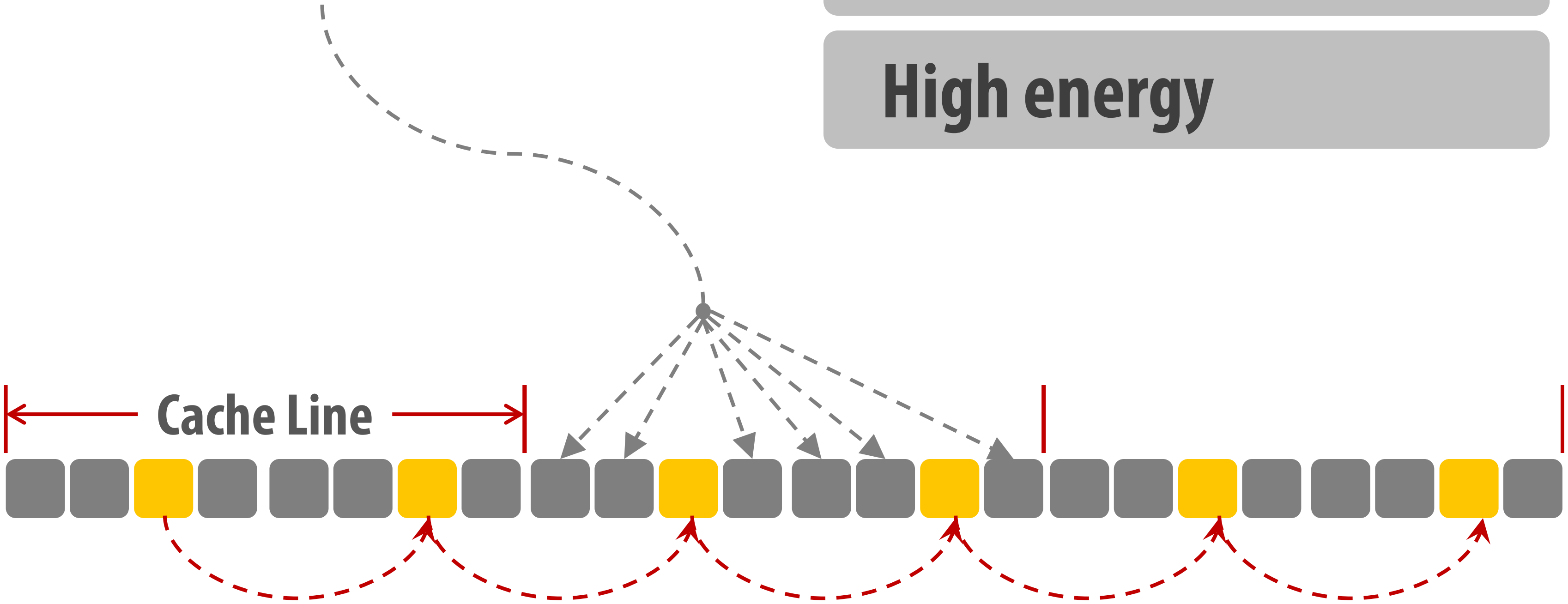
Physical layout of the data structure (row store)



Shortcomings of Existing Systems

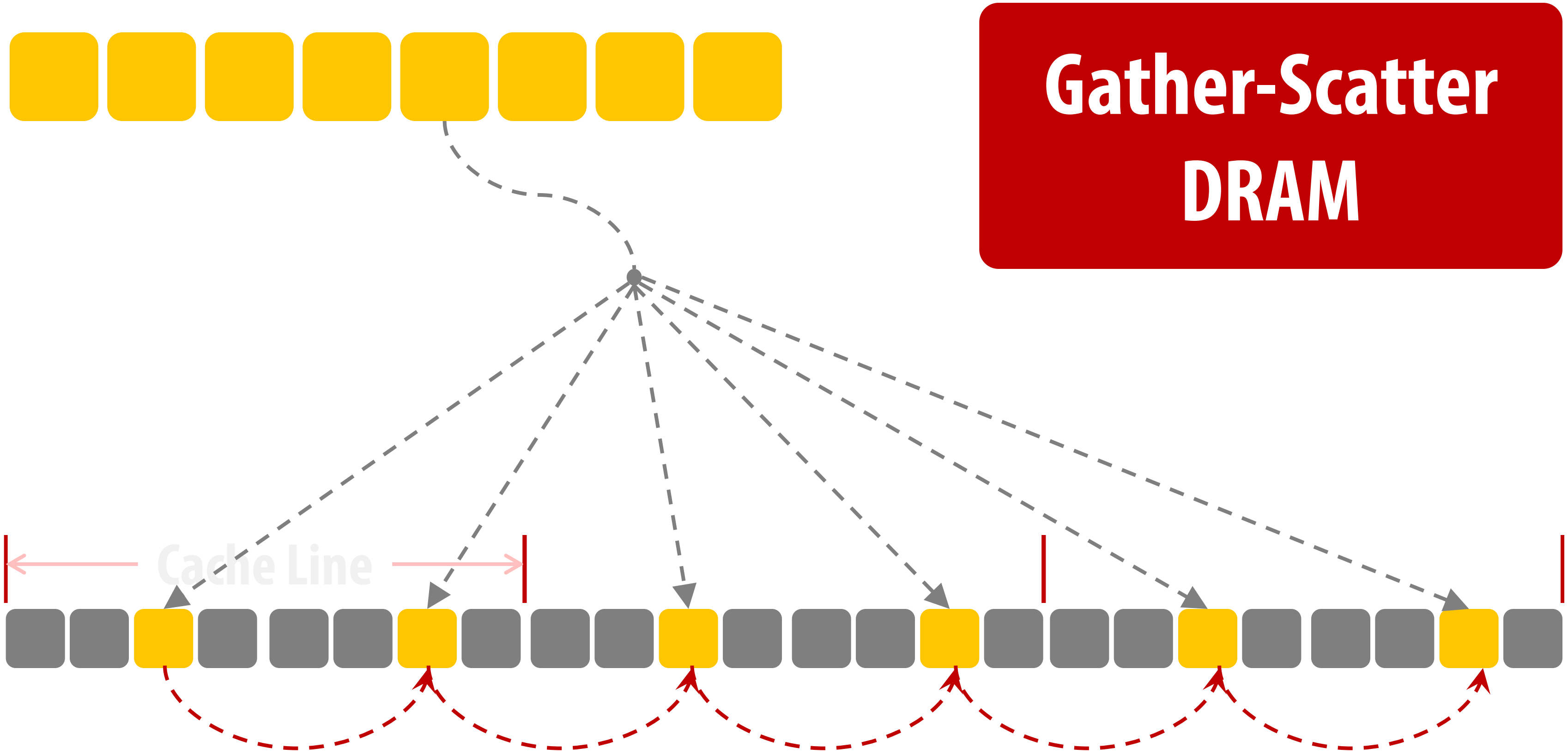
Data **unnecessarily** transferred on the **memory channel** and stored in **on-chip cache**

- High Latency
- Wasted bandwidth
- Wasted cache space
- High energy



Goal: Eliminate Inefficiency

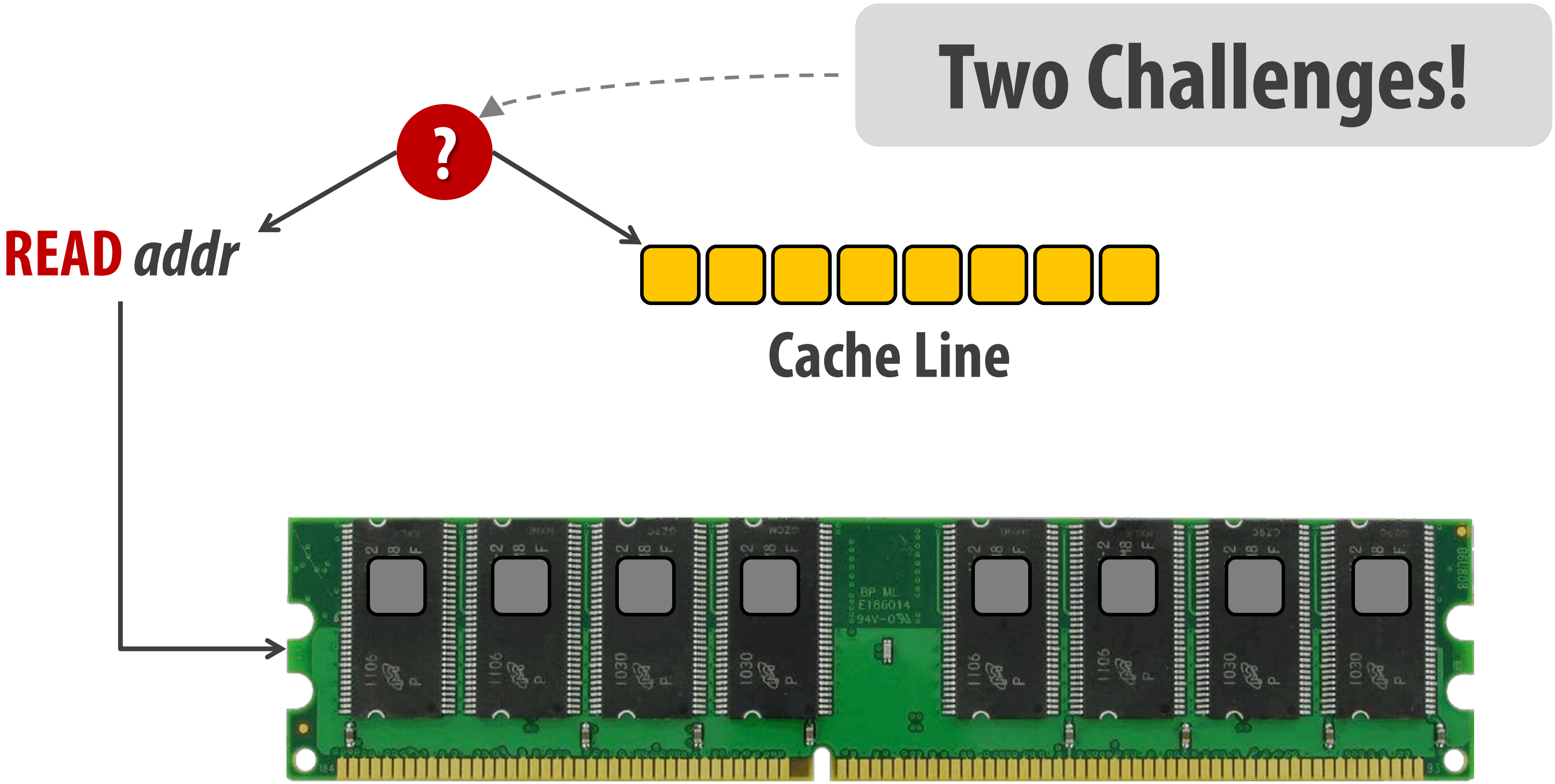
Can we retrieve a cache line with only useful data?



Gather-Scatter DRAM

DRAM Modules of Multiple Chips

All chips within a "rank" operate in unison!



Two Challenges!

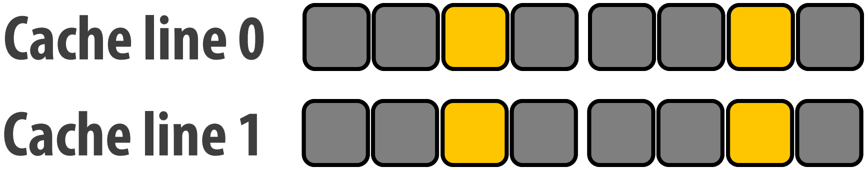
?

READ *addr*

Cache Line

Challenge 1: Chip Conflicts

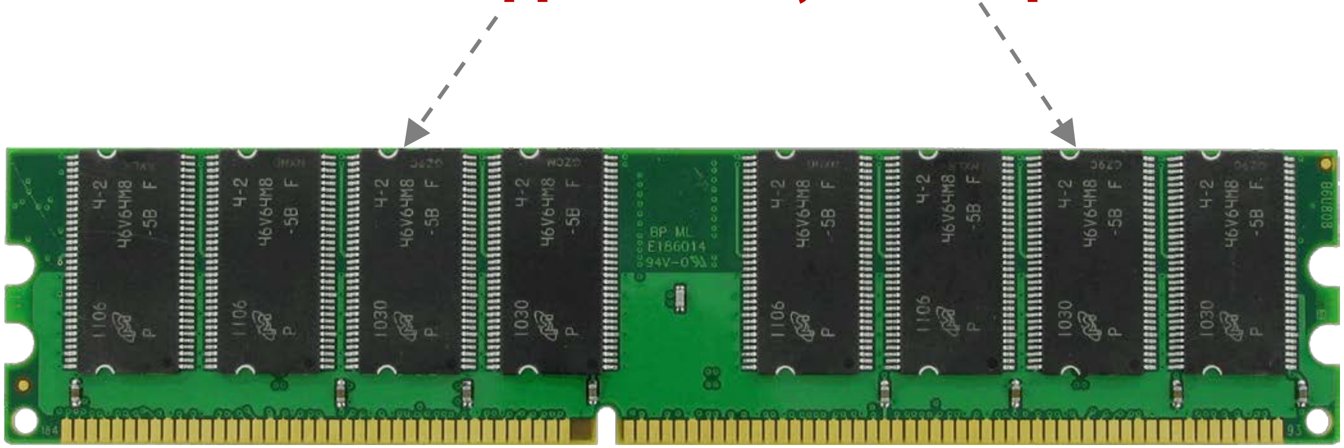
Data of each cache line is spread across all the chips!



Cache line 0

Cache line 1

Useful data mapped to only two chips!

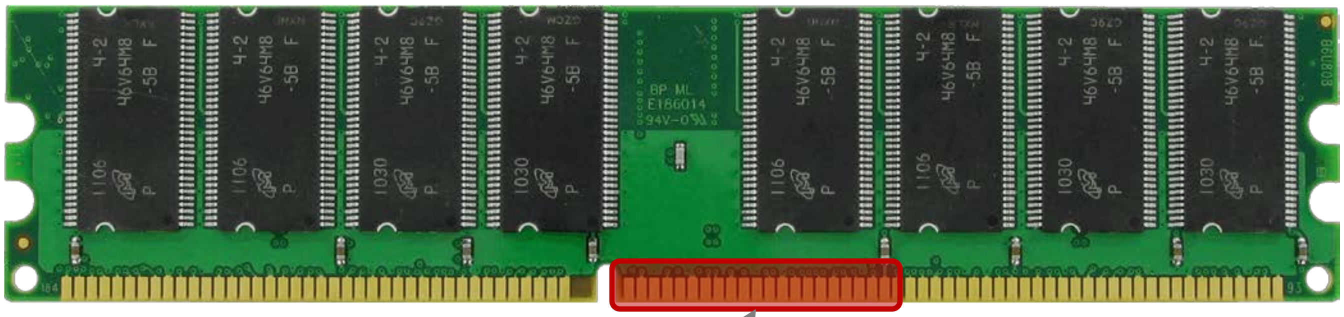


Challenge 2: Shared Address Bus

All chips share the same address bus!

No flexibility for the memory controller to read different addresses from each chip!

One address bus for each chip is costly!



Gather-Scatter DRAM

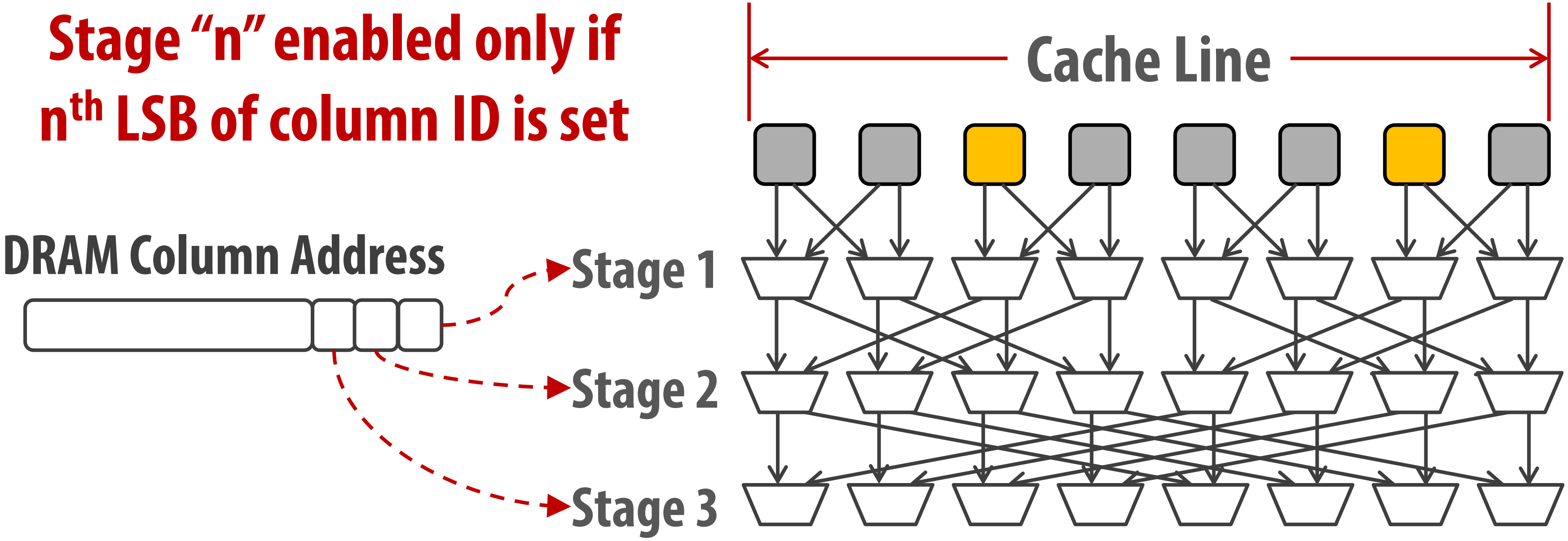
Column-ID-based Data Shuffling

(minimizing chip conflicts – for 2^n strides)

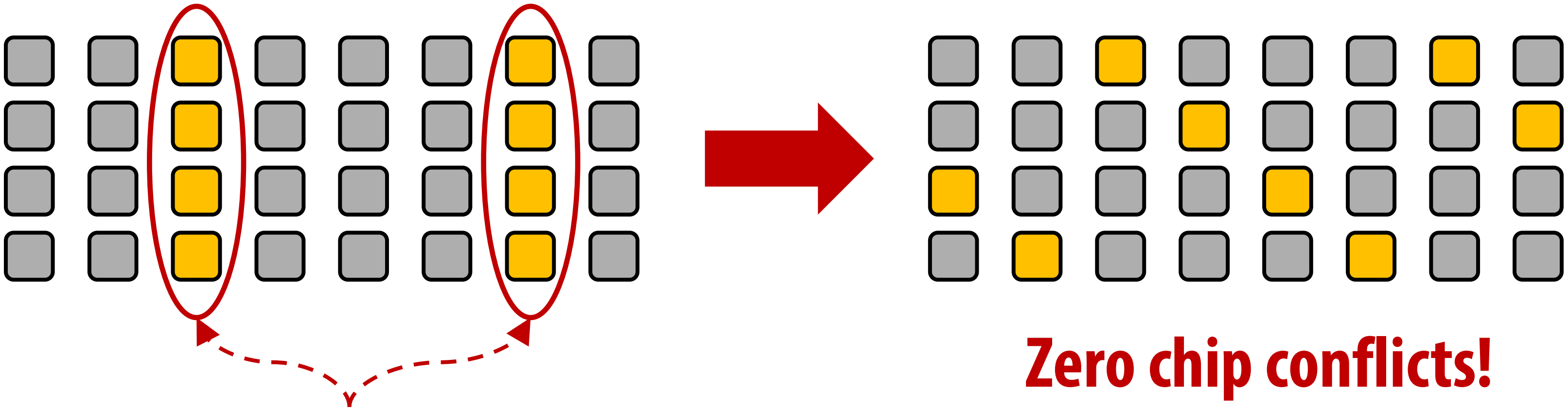
Pattern ID – In-DRAM Address Translation

(flexible column selection for different strides)

Column-ID-based Data Shuffling



Stage “n” enabled only if nth LSB of column ID is set



Chip conflicts

Zero chip conflicts!

Gather-Scatter DRAM

Column-ID-based Data Shuffling

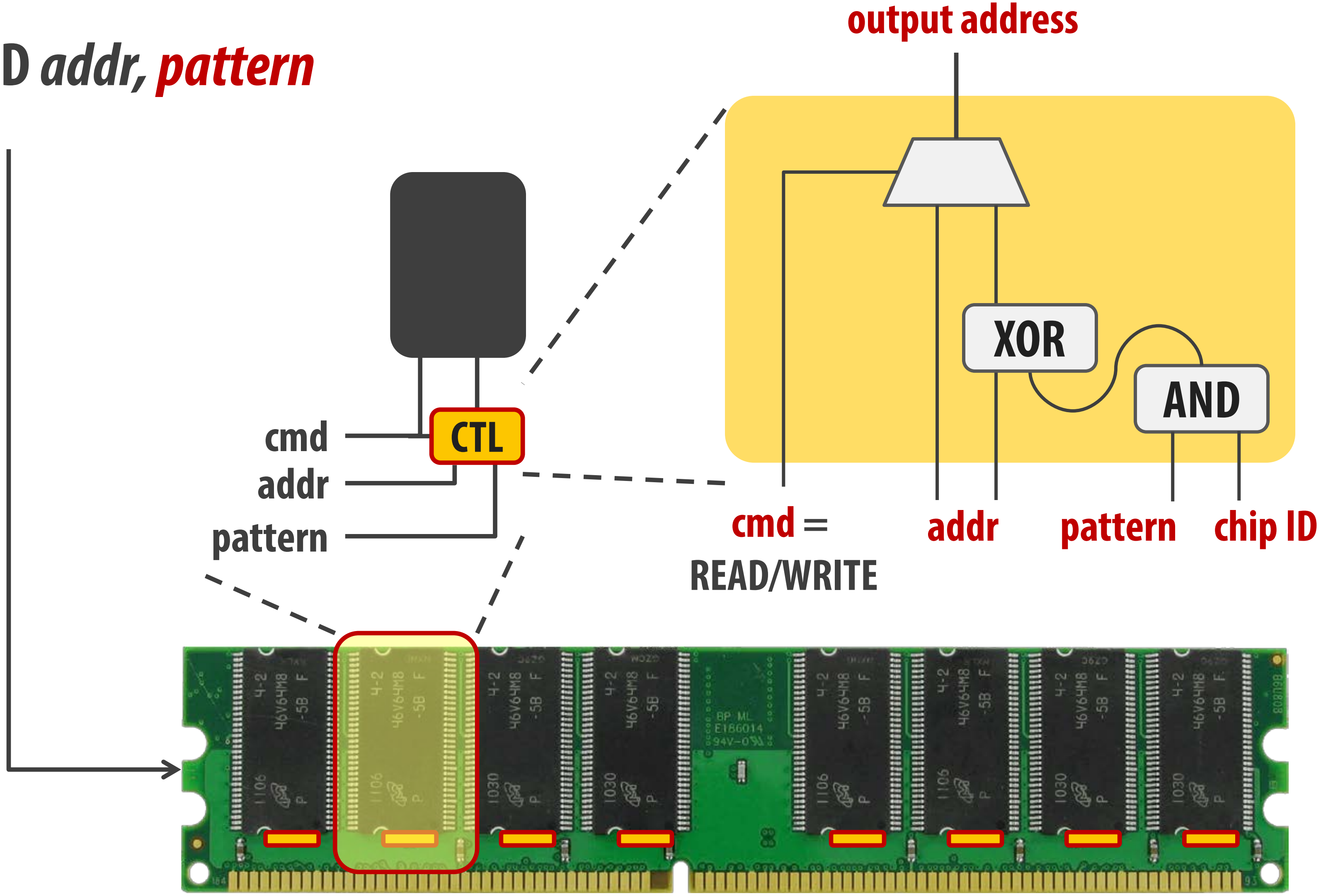
(minimizing chip conflicts – for 2^n strides)

Pattern ID – In-DRAM Address Translation

(flexible column selection for different strides)

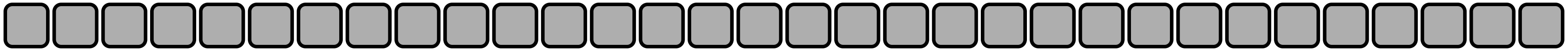
Per-Chip Column Translation Logic

READ *addr*, ***pattern***

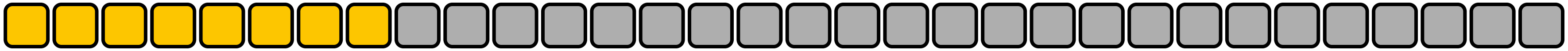


Gather-Scatter DRAM (GS-DRAM)

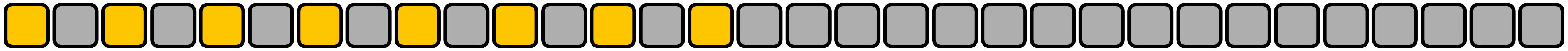
32 values contiguously stored in DRAM (at the start of a DRAM row)



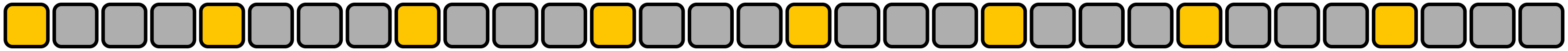
read addr **0**, pattern **0** (stride = 1, default operation)



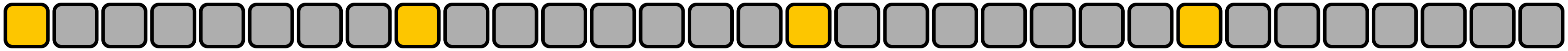
read addr **0**, pattern **1** (stride = 2)



read addr **0**, pattern **3** (stride = 4)



read addr **0**, pattern **7** (stride = 8)



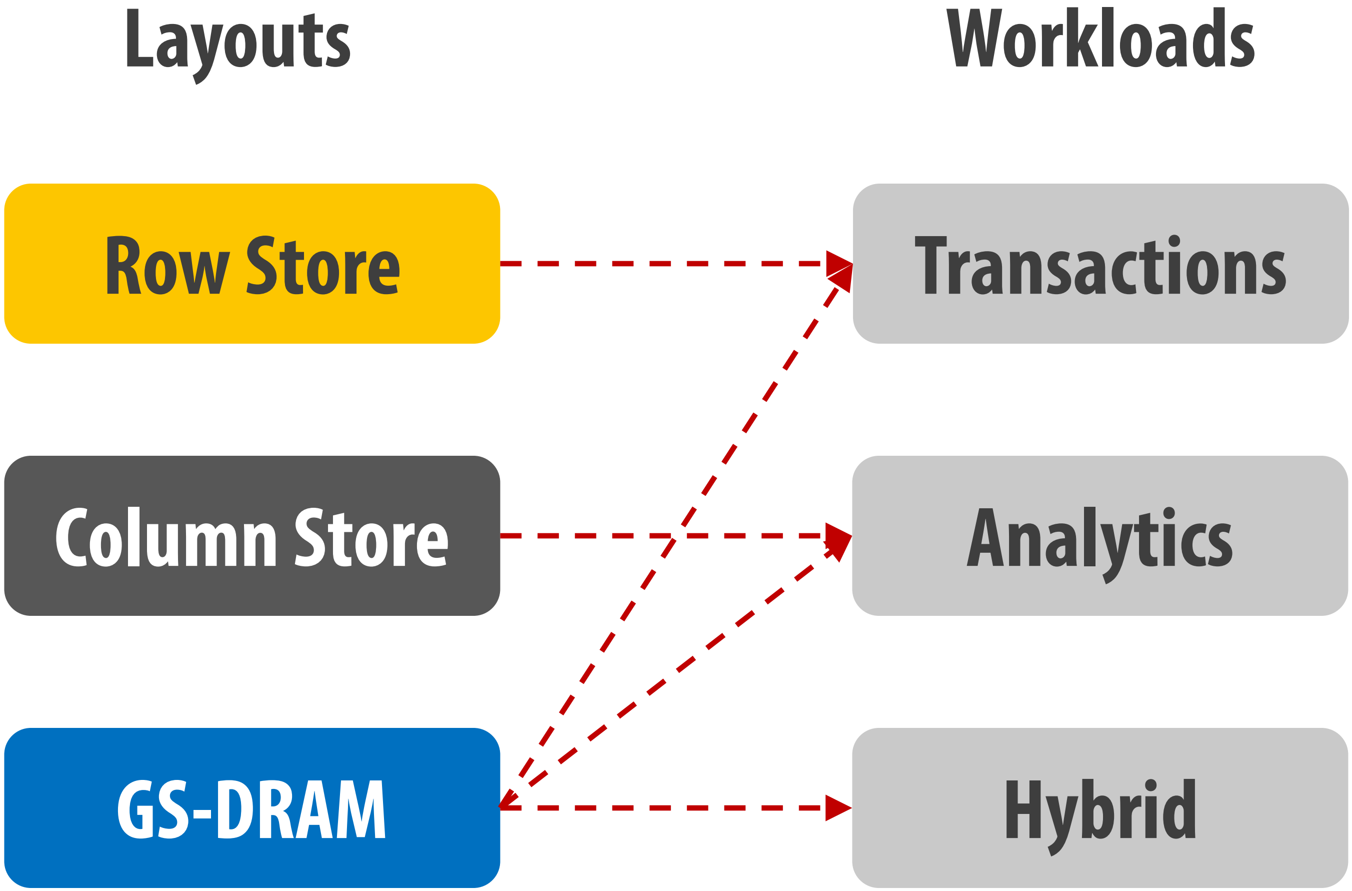
Leveraging GS-DRAM

- On-chip Cache Support
  - identifying non-contiguous cache lines
  - maintaining cache coherence
- ISA support
  - new load/store instructions with pattern
  - x86 (new address mode with pattern)
- Software support
  - specify data structures (pages) that require shuffling
  - convey pattern ID to processor using new load/store instructions

Methodology

- Gem5 x86 simulator
- Use “prefetch” instruction to implement pattern load
- 32KB L1 D/I cache
- 2MB shared L2 cache
- Main Memory: DDR3-1600, 1 channel, 1 rank, 8 banks
- FR-FCFS scheduling policy
- GS-DRAM: support for stride of 8

Evaluation: In-memory Databases



Layouts

Workloads

Row Store

Column Store

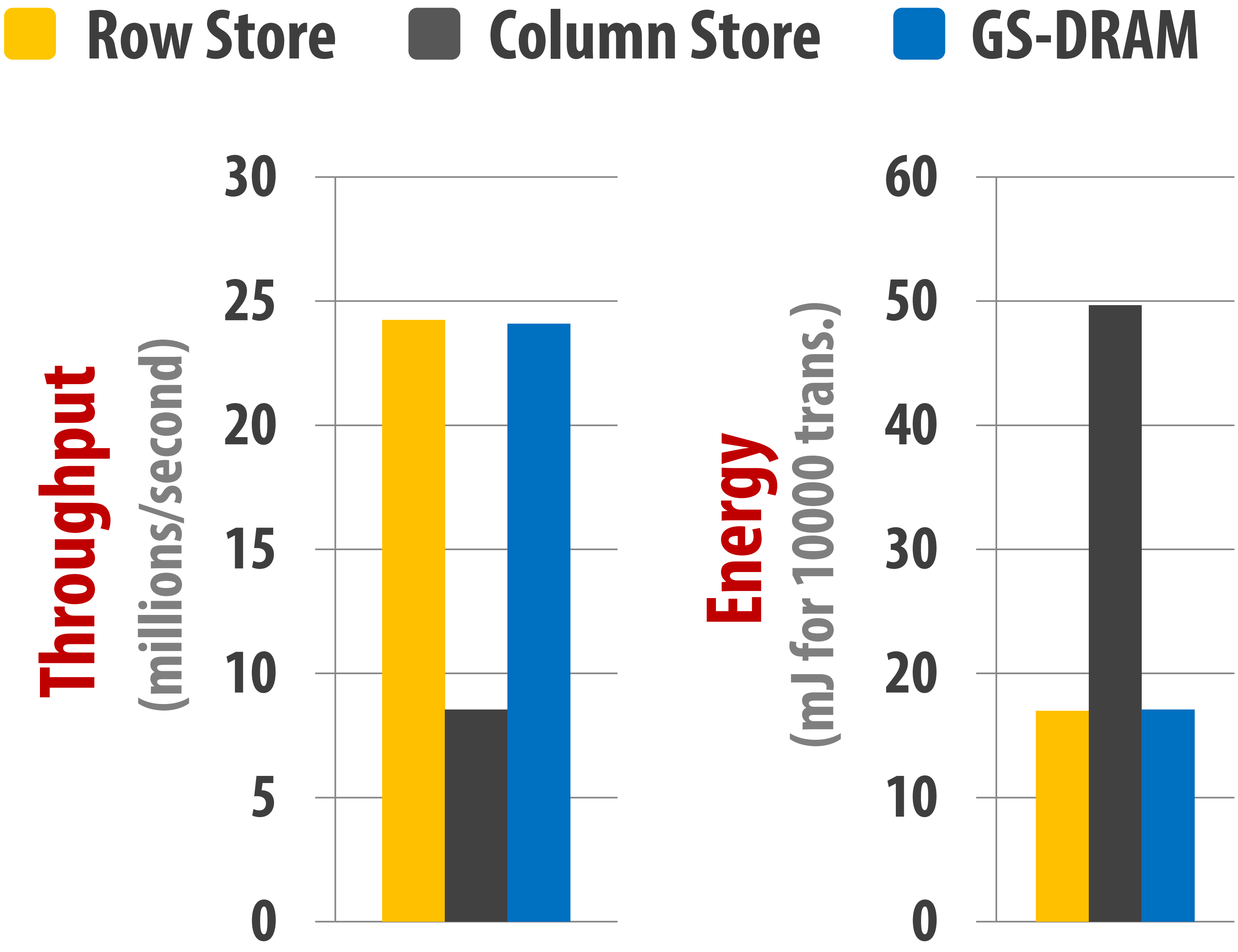
GS-DRAM

Transactions

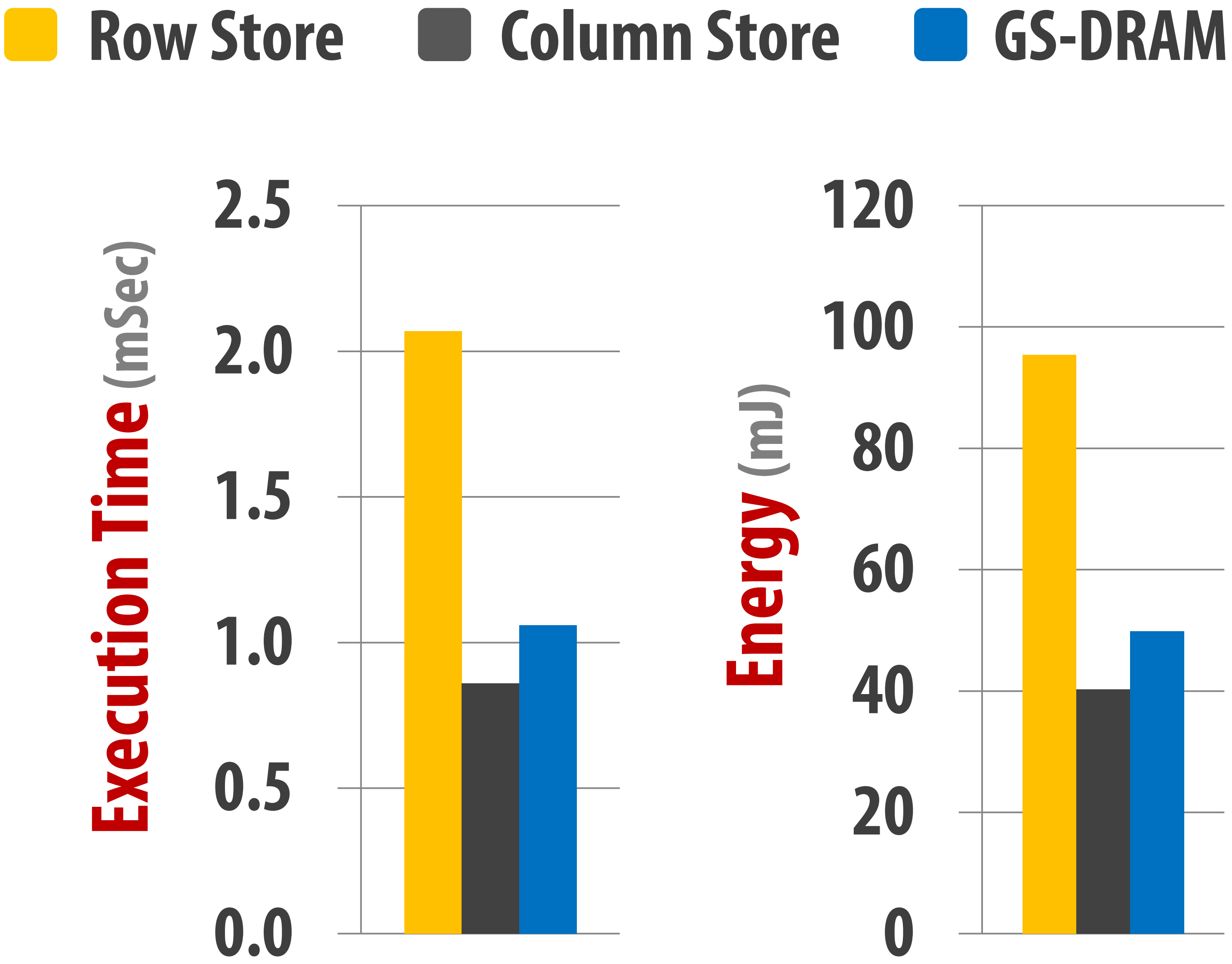
Analytics

Hybrid

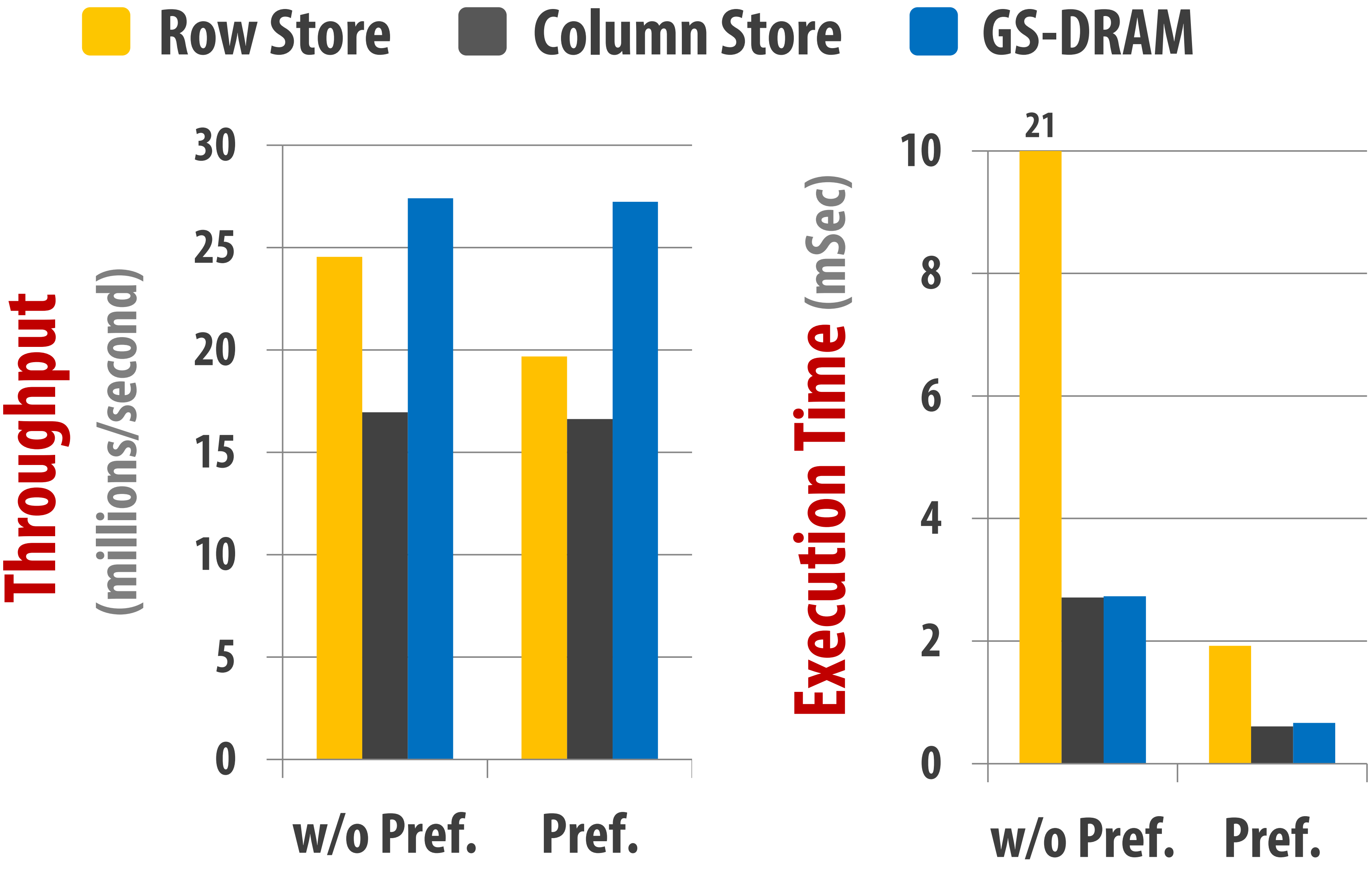
Transaction Throughput and Energy



Analytics Performance and Energy



Hybrid Transactions/Analytical Processing



Gather-Scatter DRAM: Summary

- Many data structures exhibit multiple access patterns
  - Only one access pattern has good spatial locality
- Gather-Scatter DRAM
  - memory controller gathers/scatters strided accesses
  - near ideal bandwidth/cache utilization for 2^n strides
- In-memory databases
  - GS-DRAM provides the best of both a row store and column store

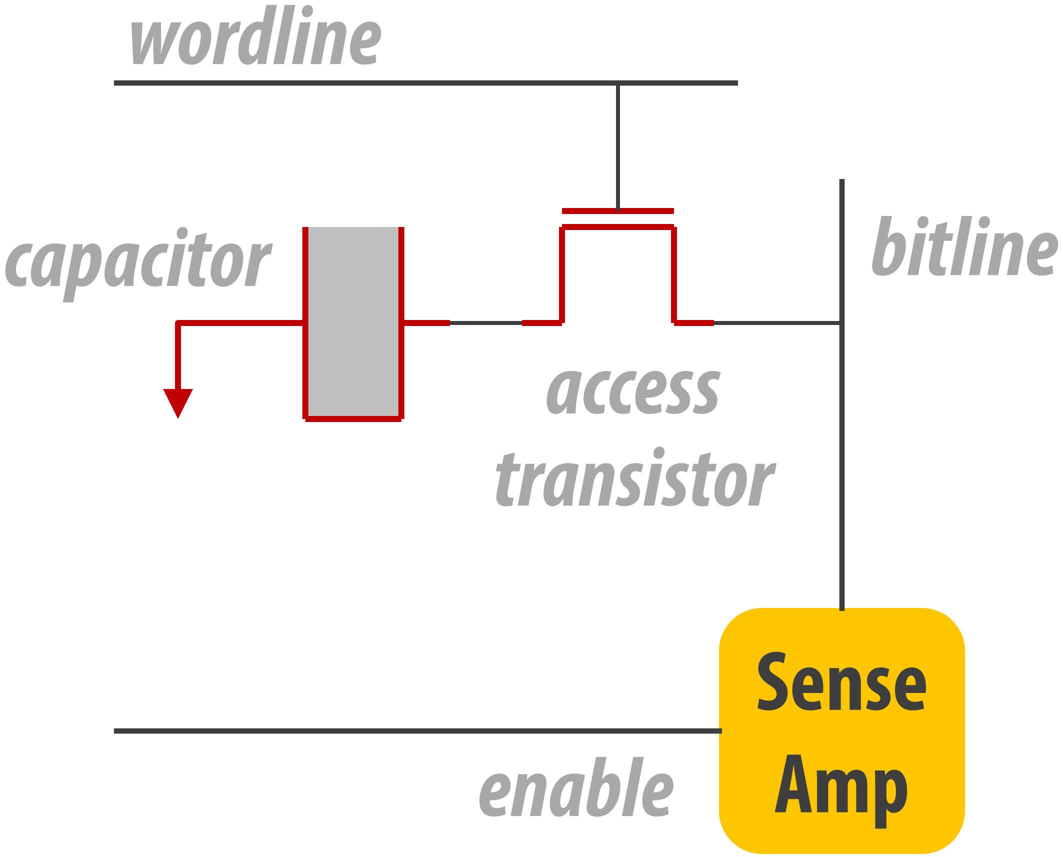
Outline of the Talk

1. Gather-Scatter DRAM
   - accelerating strided access patterns

2. RowClone
   - bulk data copy/initialization in DRAM

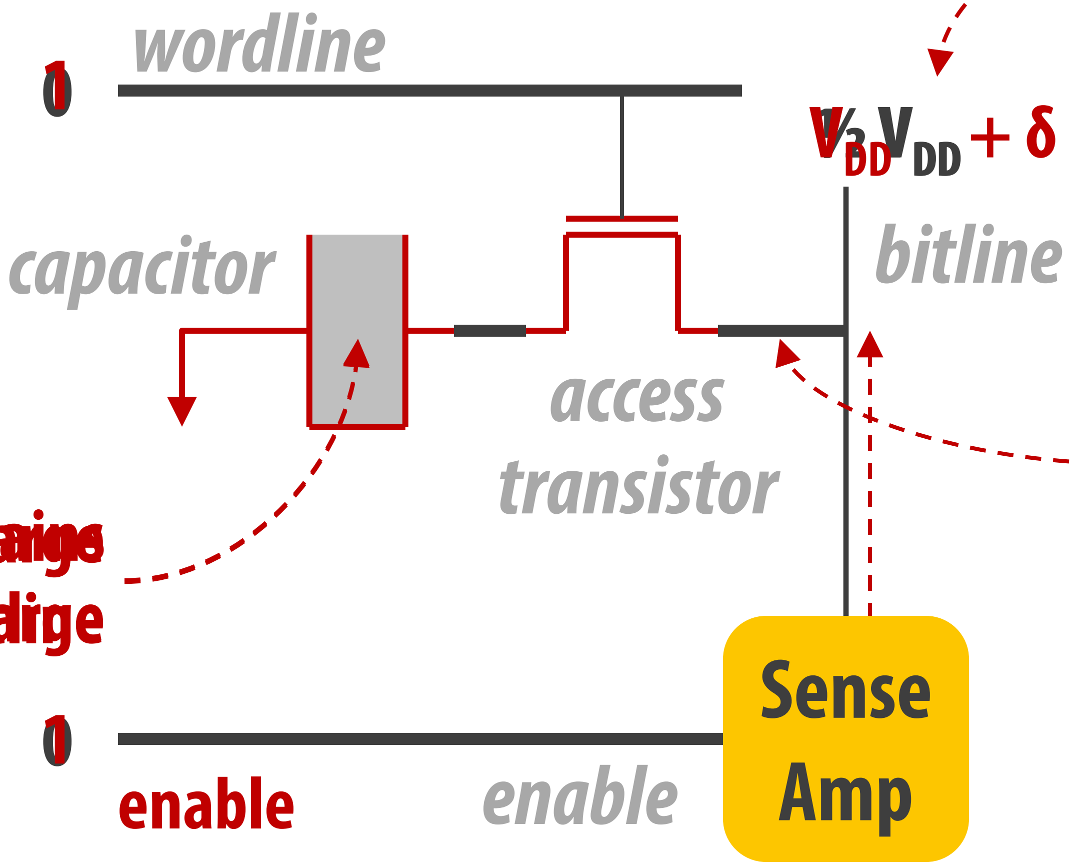
3. Bulk bitwise AND/OR in DRAM

DRAM Cell Operation



DRAM Cell Operation

raise
wordline



deviation in
bitline voltage

connects cell
to bitline

cell loses charge
to bitline

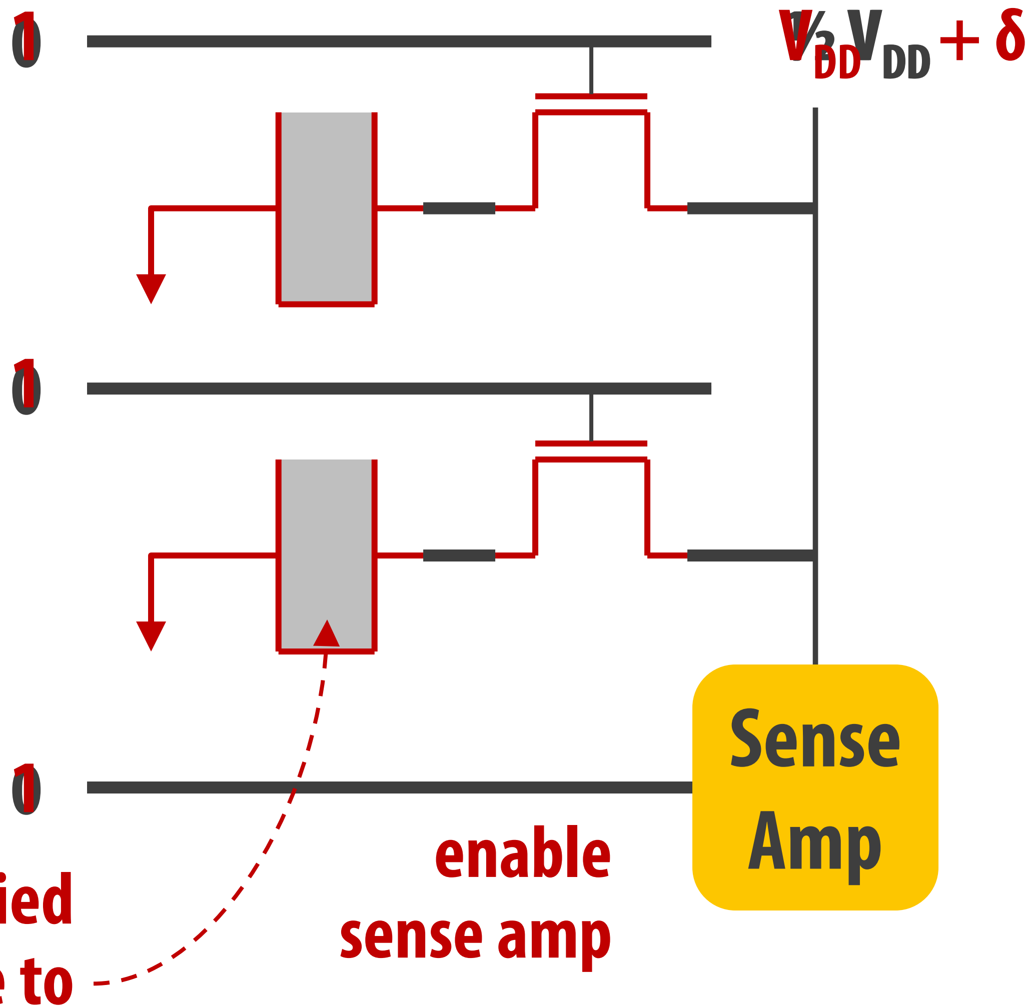
cell regains charge

enable
sense amp

RowClone: In-DRAM Bulk Data Copy

activate
source

activate
destination

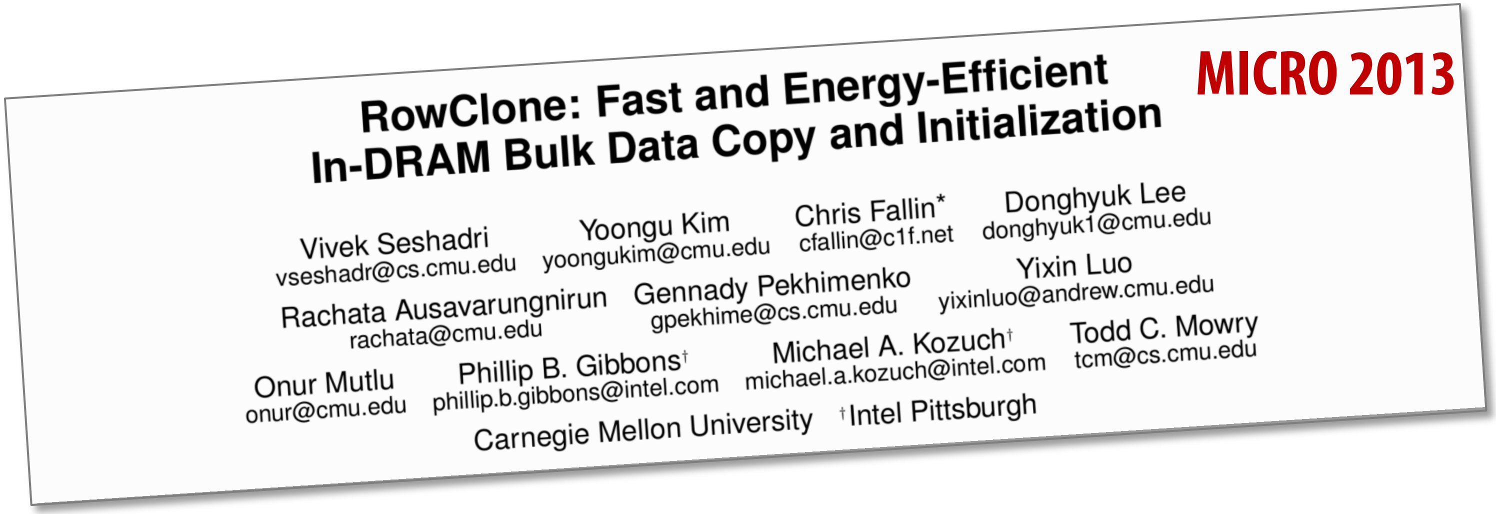


enable
sense amp

data gets copied
source to
destination

RowClone: Summary of Results

- Bulk Row-to-Row Copy (8KB)
  - 11X reduction in latency
  - 74X reduction in energy
- 8-core systems
  - 27% performance
  - 17% memory energy efficiency



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

Outline of the Talk

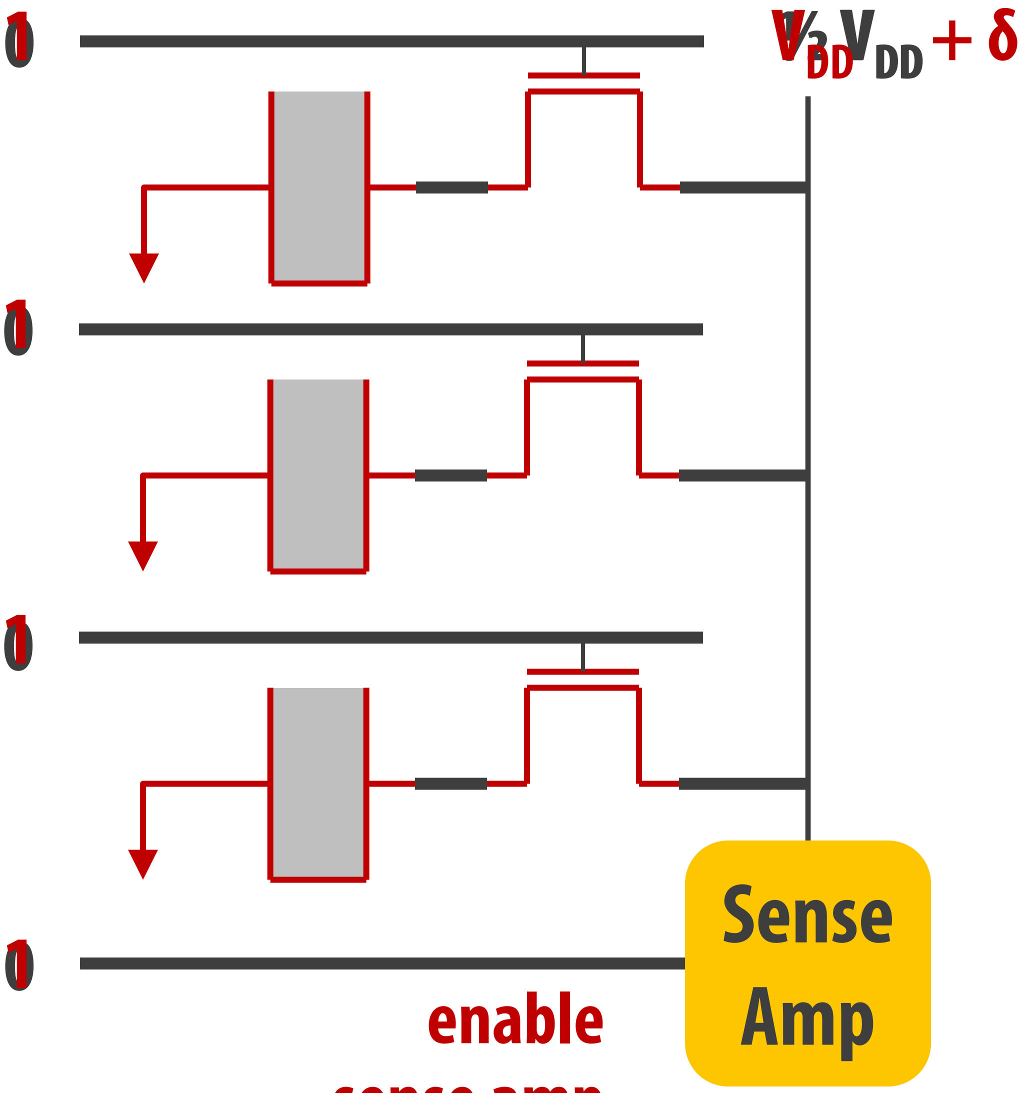
1. Gather-Scatter DRAM
   - accelerating strided access patterns

2. RowClone
   - bulk data copy/initialization in DRAM

3. Bulk bitwise AND/OR in DRAM

Triple-Row Activation

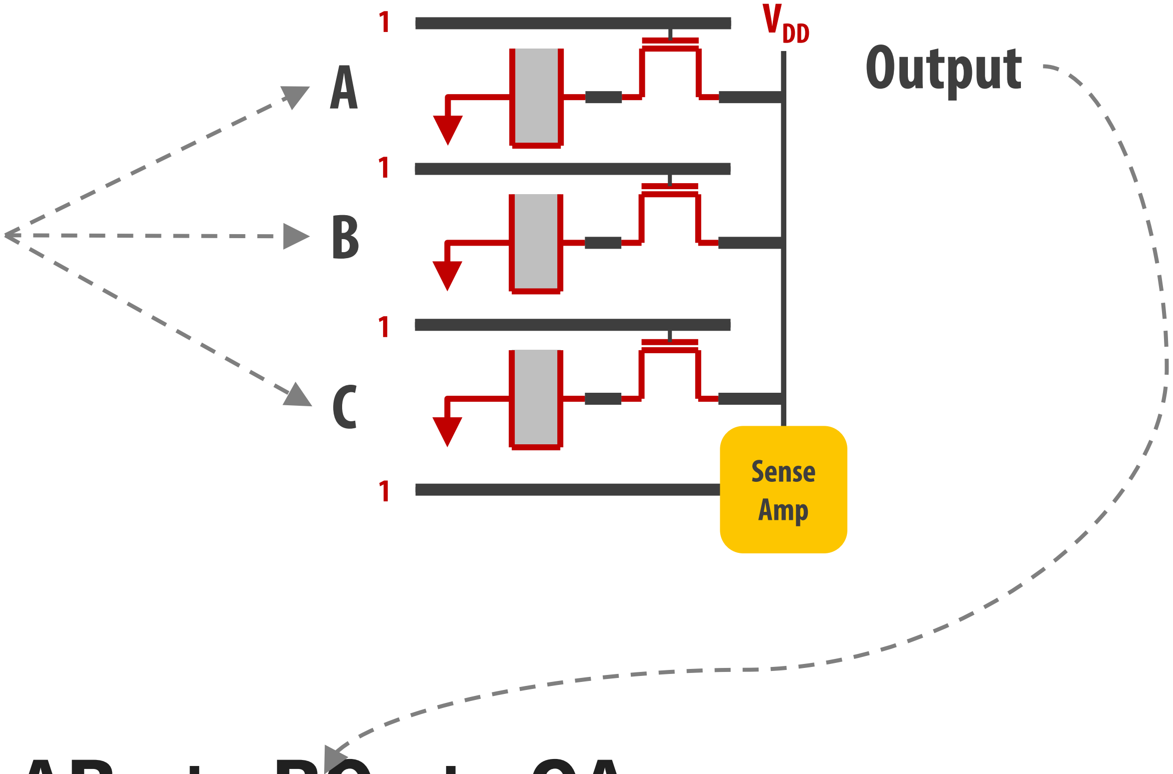
activate all three rows



enable sense amp

Triple-Row Activation

Values of all cells get overwritten



Output = AB + BC + CA

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

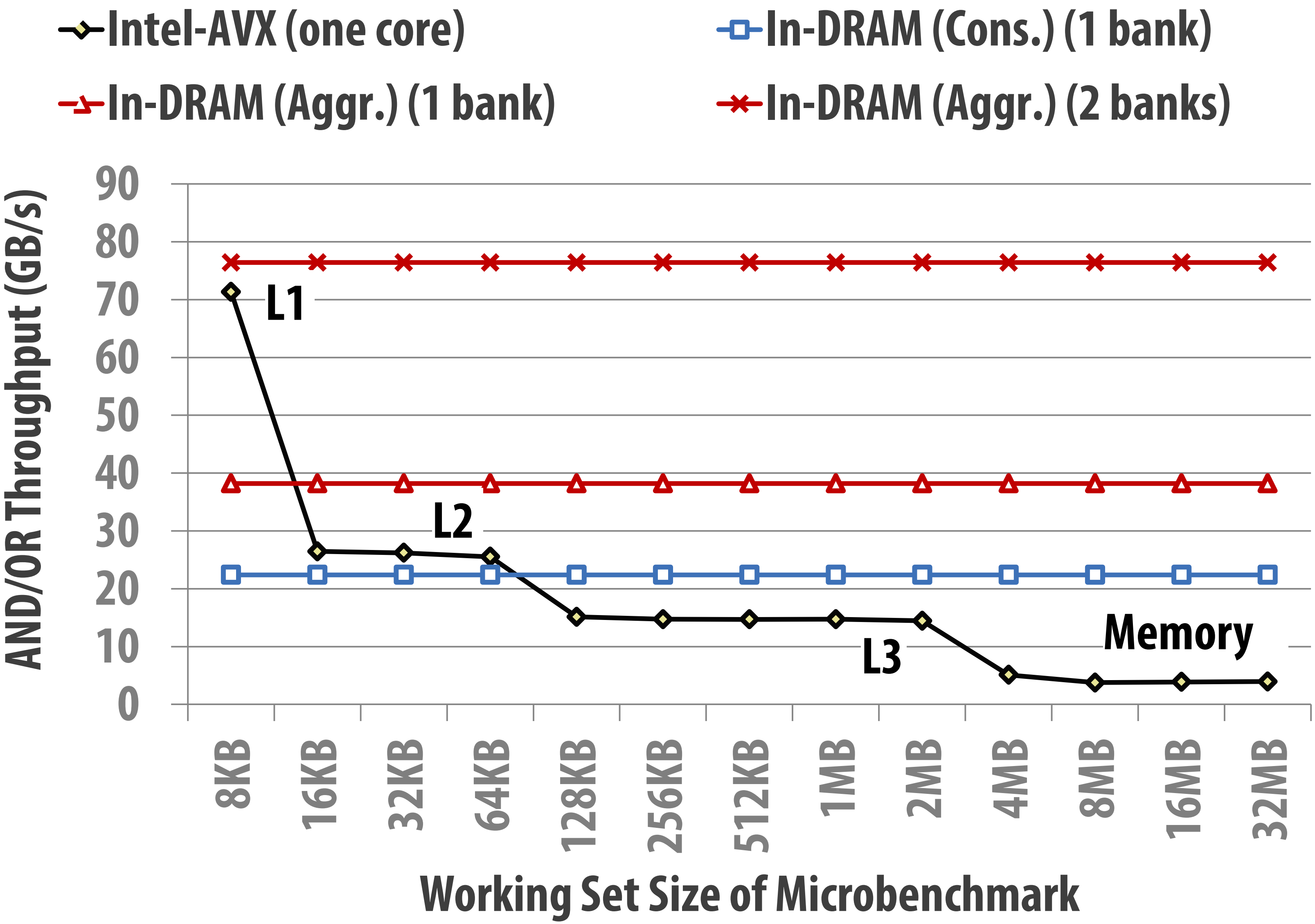
Bulk Bitwise AND/OR in DRAM

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. **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*

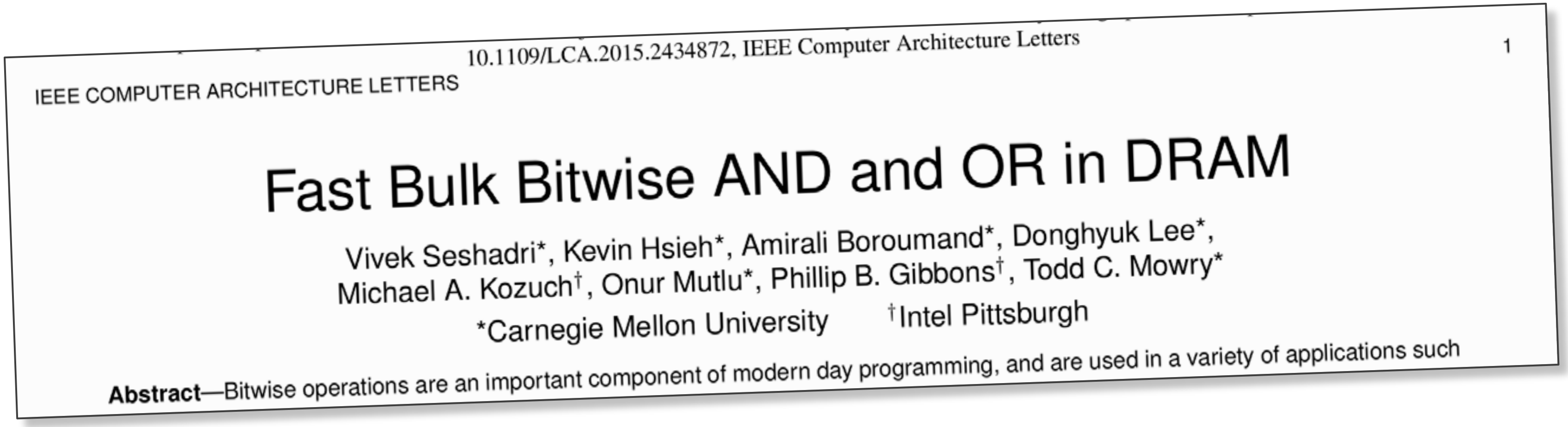
Use RowClone to perform copy and initialization operations completely in DRAM!

Throughput of Bitwise Operations



In-memory Bitmap Indices

- Predicates for records stored as bitmaps (e.g., age > 18)
- Bitwise operations to evaluate query conditions
- FastBit: real-world bitmap implementation
  - Bitwise OR: 33% of execution time for some queries
  - In-DRAM mechanism: 30% performance improvement



10.1109/LCA.2015.2434872, IEEE Computer Architecture Letters

IEEE COMPUTER ARCHITECTURE LETTERS 1

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

Abstract—Bitwise operations are an important component of modern day programming, and are used in a variety of applications such

Summary

- DRAM can be more than a storage device
- In-DRAM techniques
  - Bulk data copy/initialization
  - Gather/Scatter strided access patterns
  - Bulk bitwise AND/OR operations
- High performance at low cost
  - Order of magnitude improvement in performance
  - Changes only to the peripheral logic

Can DRAM Do More Than Just Store Data?

Vivek Seshadri
Final Year Ph.D. Student
Onur Mutlu, Todd C. Mowry
Phillip B. Gibbons, Michael A. Kozuch

http://www.istc-cc.cmu.edu/

Intel Science & Technology
Center for Cloud Computing