ETH zürich

D INFK

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

Proceeding of the 52nd International Symposium on Microarchitecture (MICRO), October 2019

Seminar in Computer Architecture
Presented by: Christopher Meier
19 October 2020

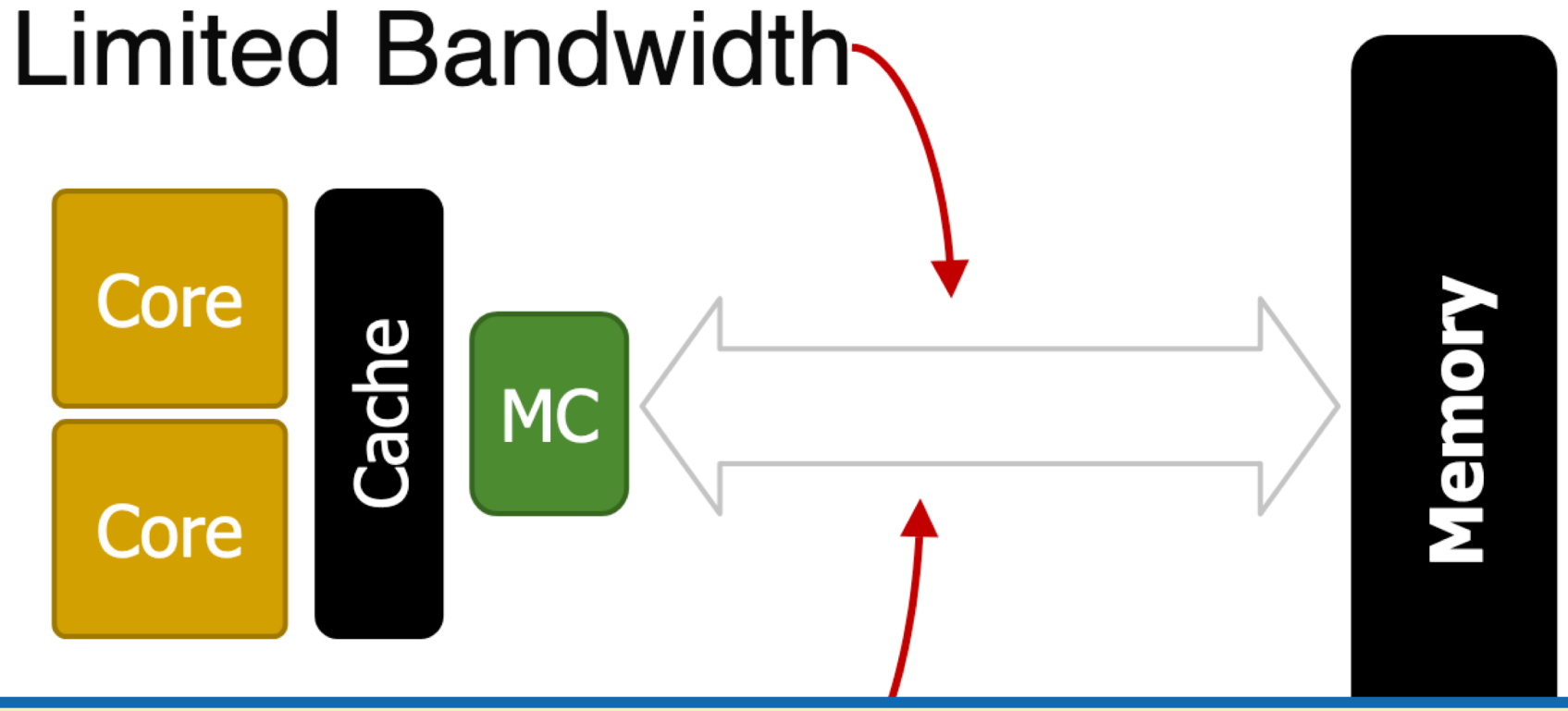
Executive summary

- **Motivation**: Proof that AMBIT and RowClone are usable.
- **Goal**: Demonstrate row copy and bit-wise logical AND and OR in unmodified, commercial, DRAM.
- **Key Idea**: Violate DRAM timing constraints to enable charge sharing across multiple rows in the same sub-array.
- **Mechanism**: Perform operations with DRAM, by carefully violating its timing constraints.
- **Implementation**: Provide an in-memory compute framework to allow arbitrary computation.
- **Results**: Enable high computational throughput, up to 347x more energy efficient than using a vector unit.

Outline

1. Motivation
2. Solution Approaches
3. Recap on DRAM
4. Key Idea
5. Mechanism of ComputeDRAM
6. Operation Reliability
7. Implementation of ComputeDRAM
8. Methodology
9. Evaluation
10. Conclusion

Motivation

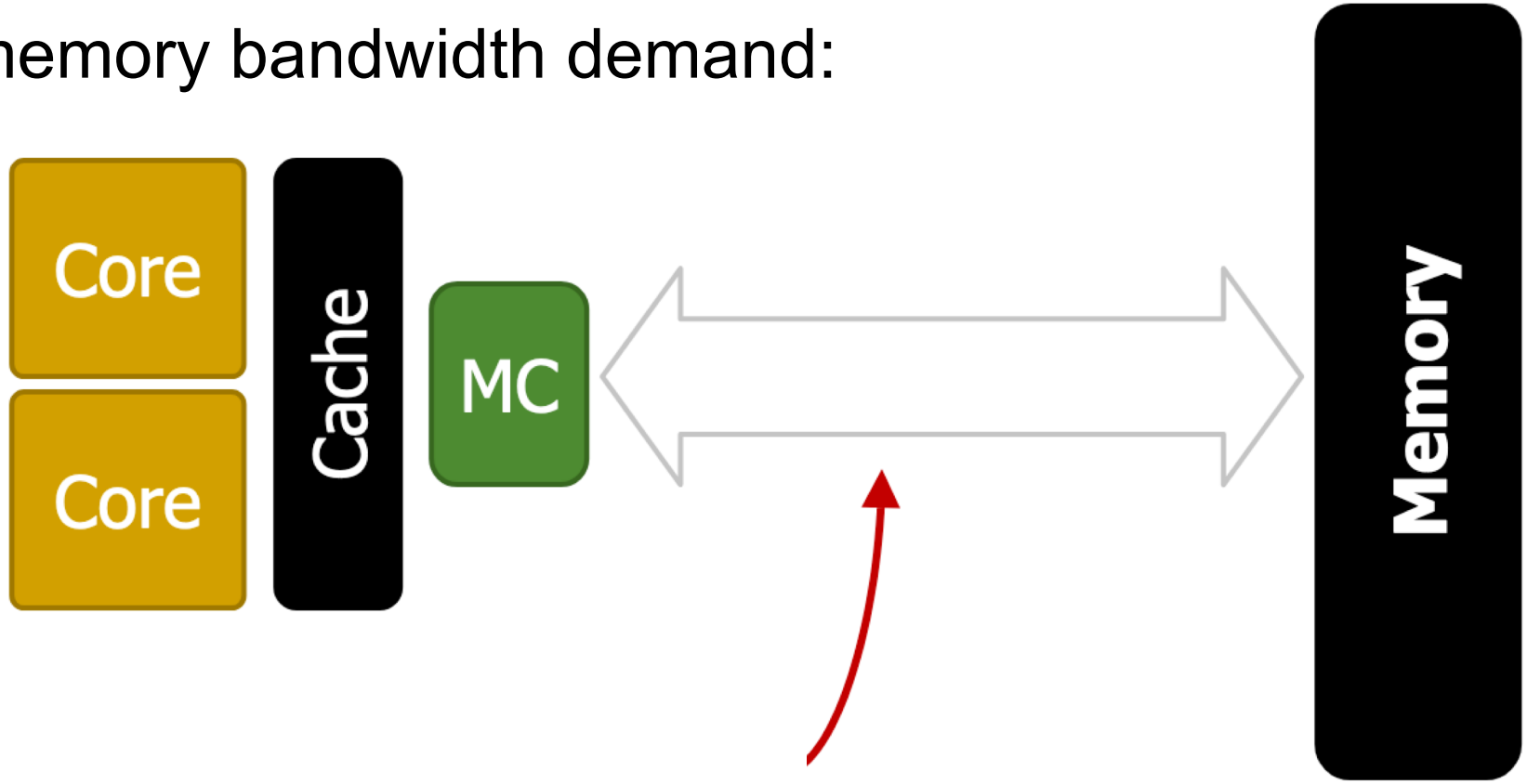


Google consumer workloads[1]:
Data movement contributes to 62.7% of the total energy consumption.

Illustration from Prof. Mutlu's presentation on RowClone, pp. 23
[1]: A. Boroumand et al. 2018. Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks. In ASPLOS '18:

Motivation

Reduce memory bandwidth demand:



Reduce unnecessary data movement

[1]: Illustration from Prof. Mutlu's presentation on RowClone, pp 23.

Solution Approach

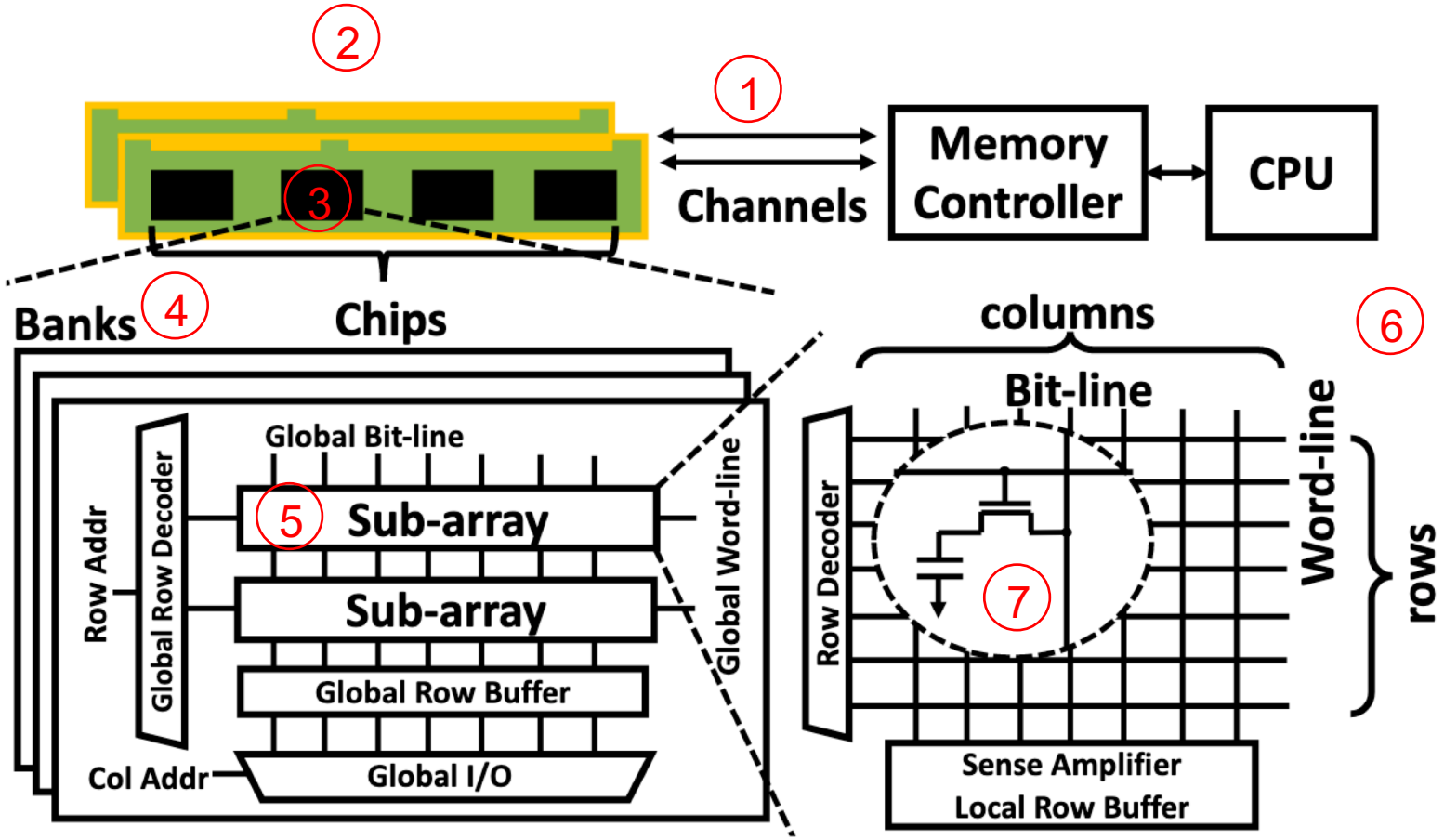
Eliminating data movement by bringing computation closer to memory.

Approach	Enabling Technologies
Processing-Near-Memory	Logic layers in 3D-stacked memory Silicon interposers Logic in memory controllers
Processing-Using-Memory	SRAM DRAM Phase-change memory (PCM) Magnetic RAM (MRAM) Resistive RAM (RRAM)/memristors

S. Ghose, A. Boroumand, J. S. Kim, J. Gómez-Luna and O. Mutlu,
"Processing-in-memory: A workload-driven perspective,"
in *IBM Journal of Research and Development*,
vol. 63, no. 6, pp. 3:1-3:19, 1 Nov.-Dec. 2019, doi: 10.1147/JRD.2019.2934048.

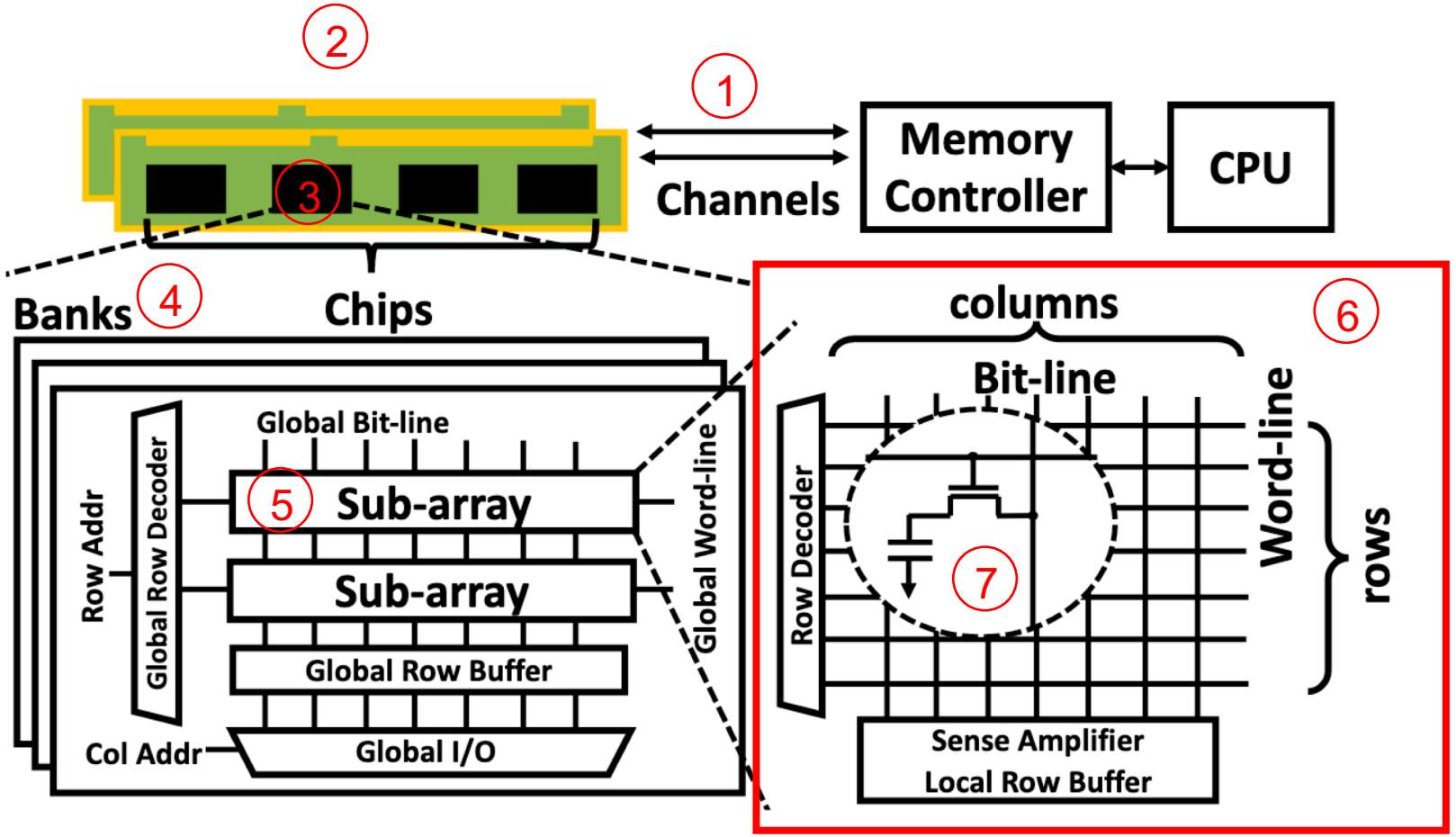
Recap: DRAM Hierarchy

1. Channel
2. Rank
3. Chip
4. Bank
5. Sub-Array
6. Row/Colum
7. Cell



Recap: DRAM Hierarchy

1. Channel
2. Rank
3. Chip
4. Bank
5. **Sub-Array**
6. **Row/Colum**
7. **Cell**

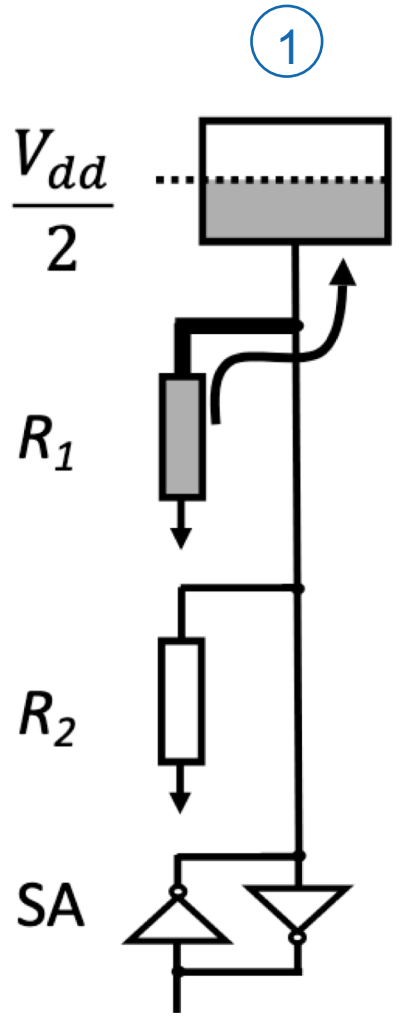


Recap: DRAM Commands

- **Activate:**

  On row level

  1. Open target row

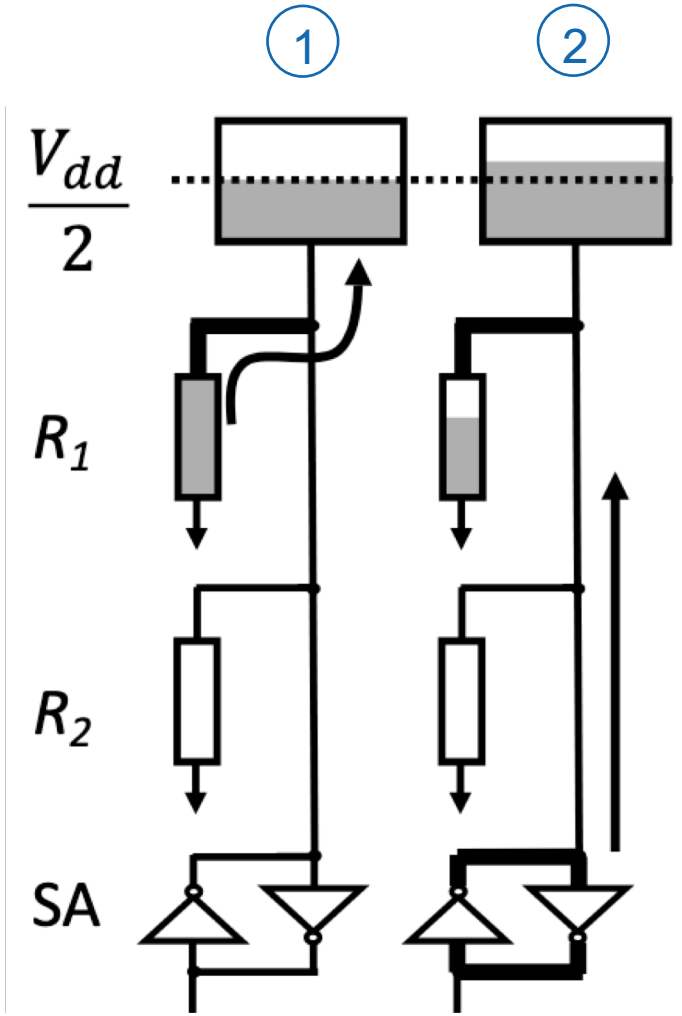


Recap: DRAM Commands

- **Activate:**

  On row level

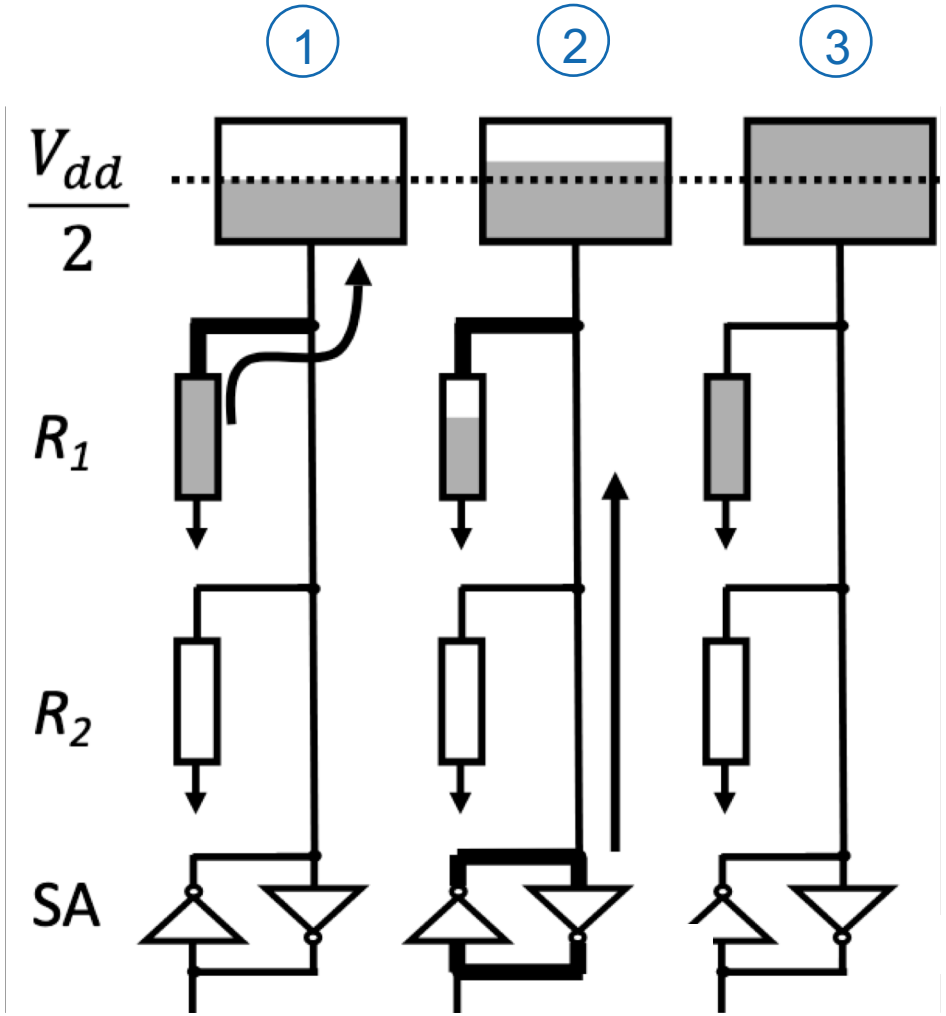
  1. Open target row
  2. Amplify bit-line charge



Recap: DRAM Commands

- **Activate:**
  On row level
  1. Open target row
  2. Amplify bit-line charge

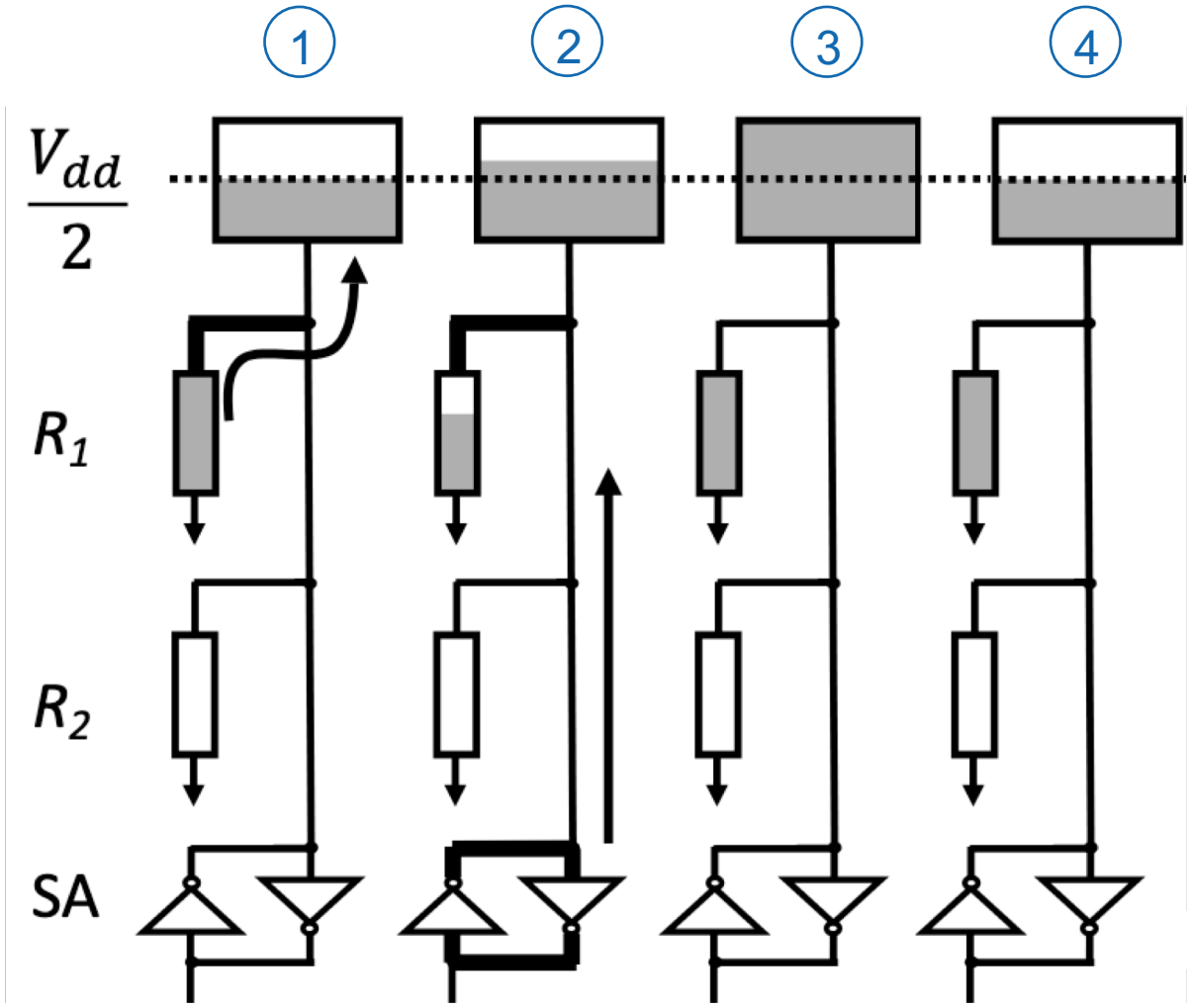
- **Precharge:**
  On bank level
  3. Close all rows



Recap: DRAM Commands

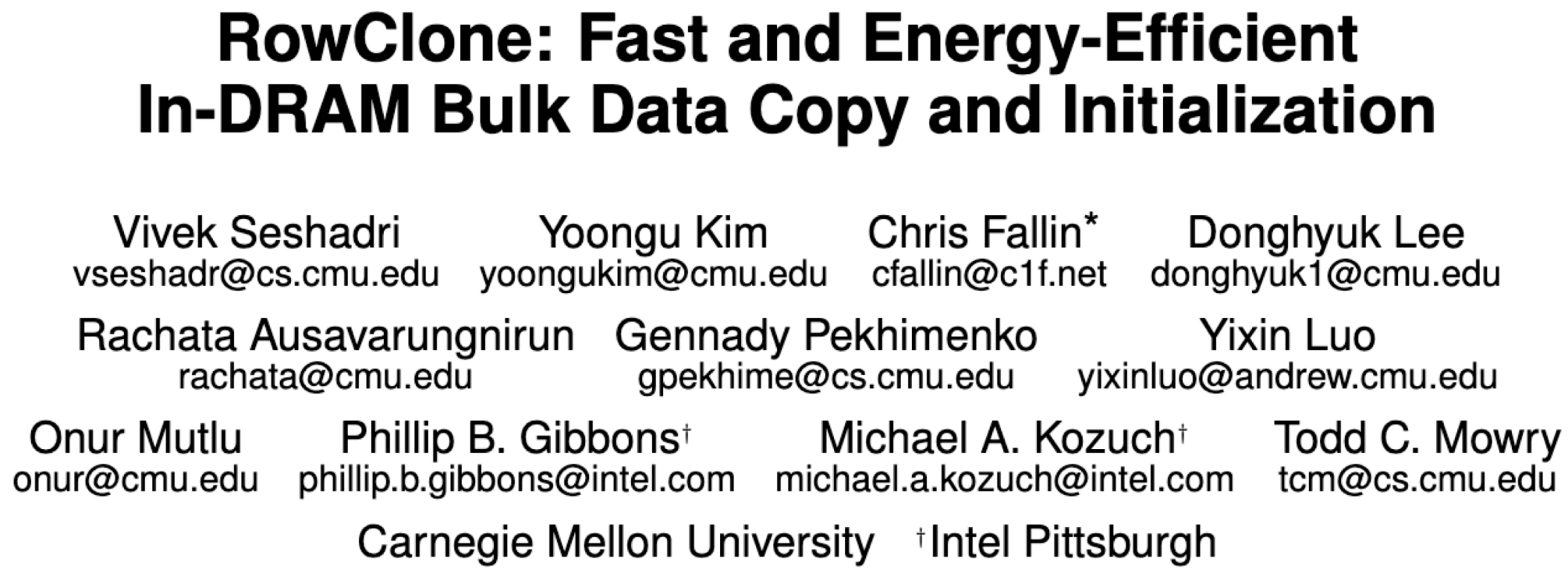
- **Activate:**
  On row level
  1. Open target row
  2. Amplify bit-line charge

- **Precharge:**
  On bank level
  3. Close all rows
  4. Drive bit-lines to $V_{dd}/2$



Motivation

- 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.



RowClone: Fast and Energy-Efficient In-DRAM Bulk Data Copy and Initialization

Vivek Seshadri vseshadr@cs.cmu.edu
Yoongu Kim yoongukim@cmu.edu
Chris Fallin* cfallin@c1f.net
Donghyuk Lee donghyuk1@cmu.edu
Rachata Ausavarungnirun rachata@cmu.edu
Gennady Pekhimenko gpekhime@cs.cmu.edu
Yixin Luo yixinluo@andrew.cmu.edu
Onur Mutlu onur@cmu.edu
Phillip B. Gibbons† phillip.b.gibbons@intel.com
Michael A. Kozuch† michael.a.kozuch@intel.com
Todd C. Mowry tcm@cs.cmu.edu
Carnegie Mellon University †Intel Pittsburgh

RowClone: Intra-Subarray Copy

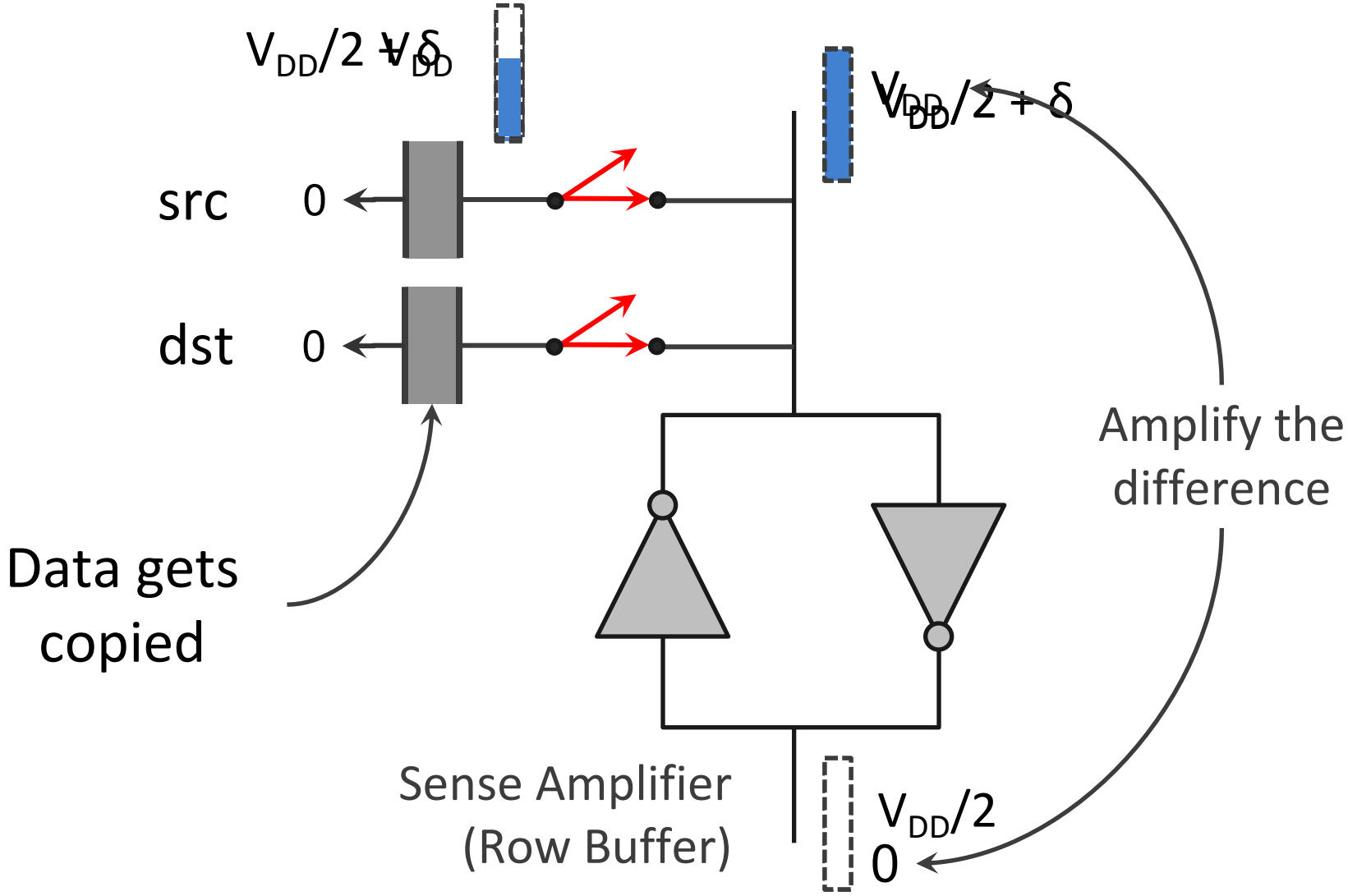
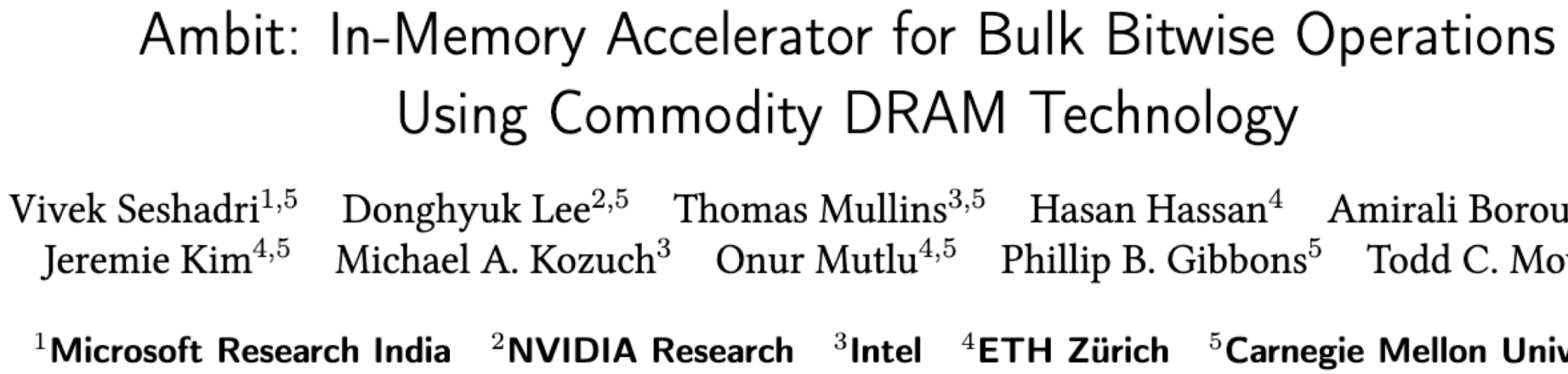


Illustration from Prof. Mutlu's presentation on RowClone, pp 31.

Motivation

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

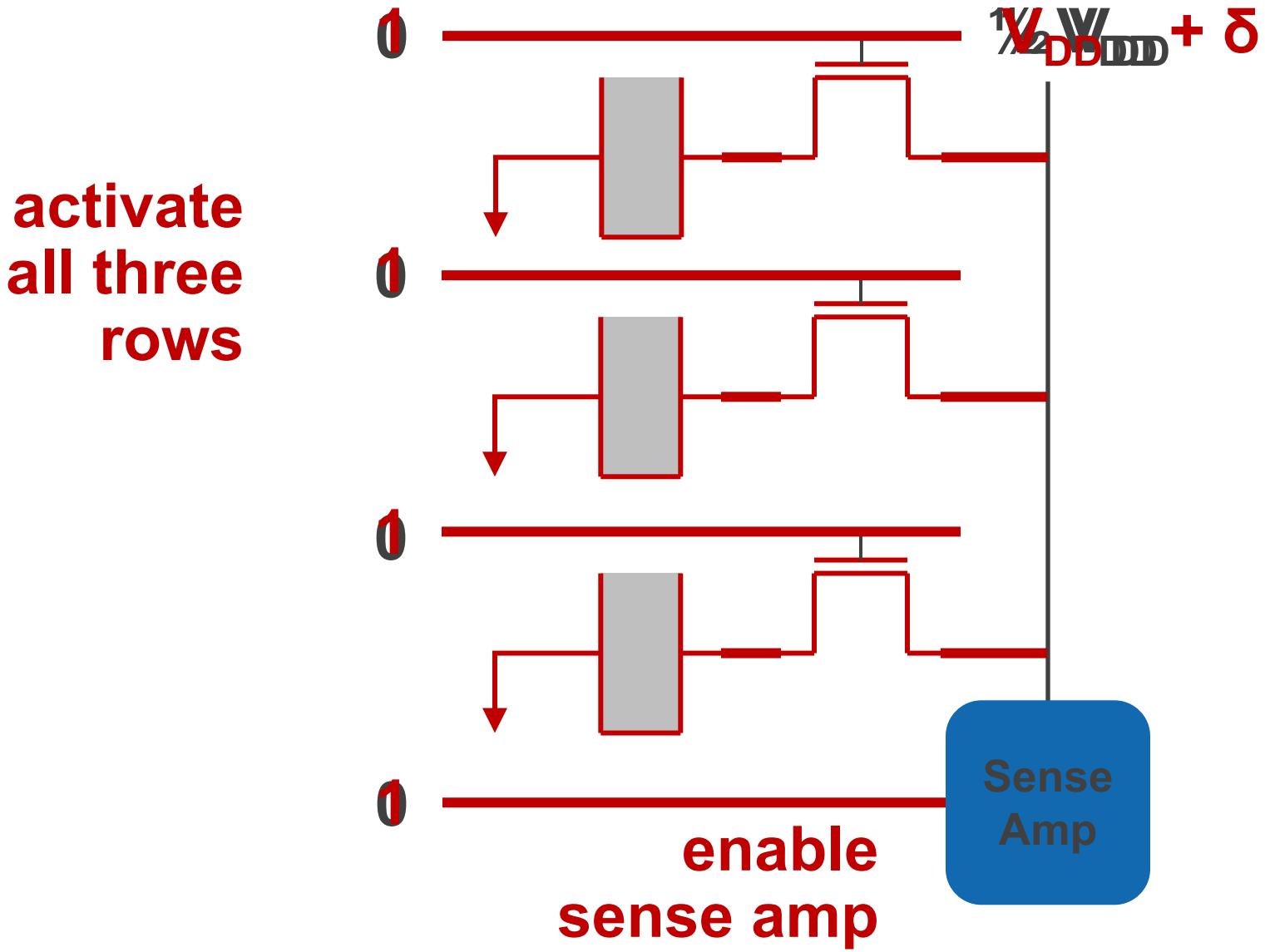


Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology

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

Triple-Row Activation: Majority Function

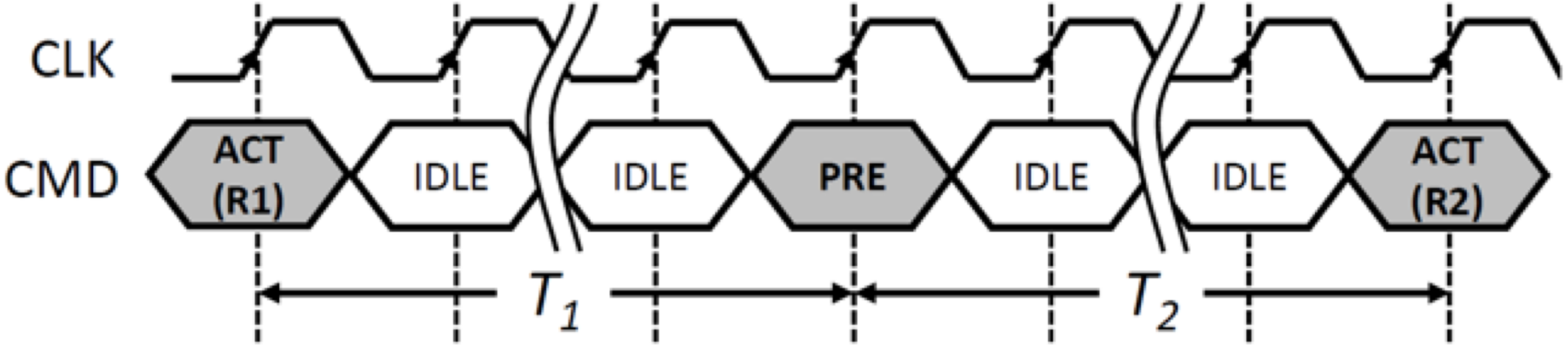


Animation from: https://www.archive.ece.cmu.edu/~safari/pubs/ambit-bulk-bitwise-dram_micro17-talk.pptx

Key Idea

DRAM Operation Timing

- Timing constraints guarantee correctness

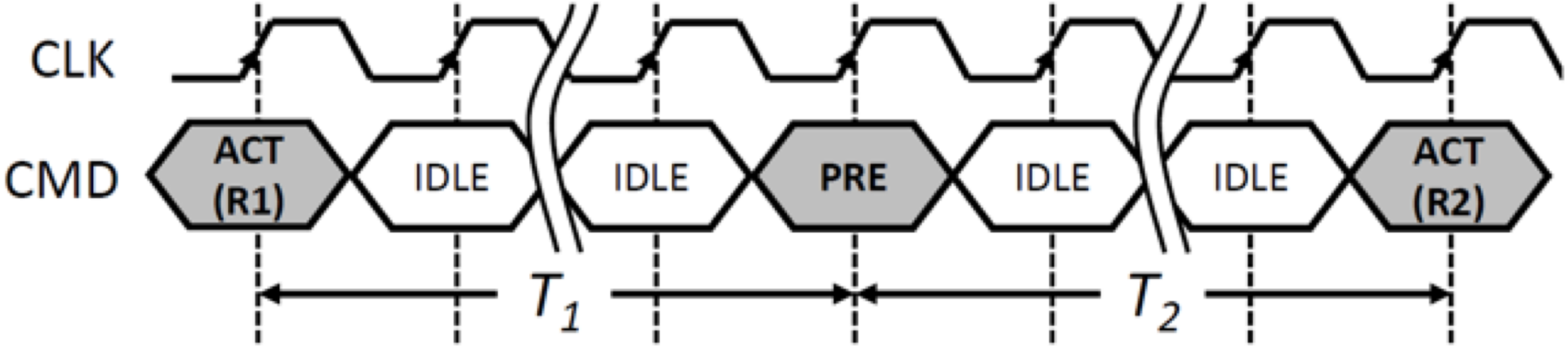


- $T1$, Row Access Strobe t_{RAS} : time to open a row, enable sense amplifier, wait for voltage to reach V_{dd} or GND
- $T2$, Row Precharge t_{RP} : ensures that the previously activated row is closed, and the bit-line voltage has reached $V_{dd}/2$

Key Idea

DRAM Operation Timing

- Timing constraints guarantee correctness



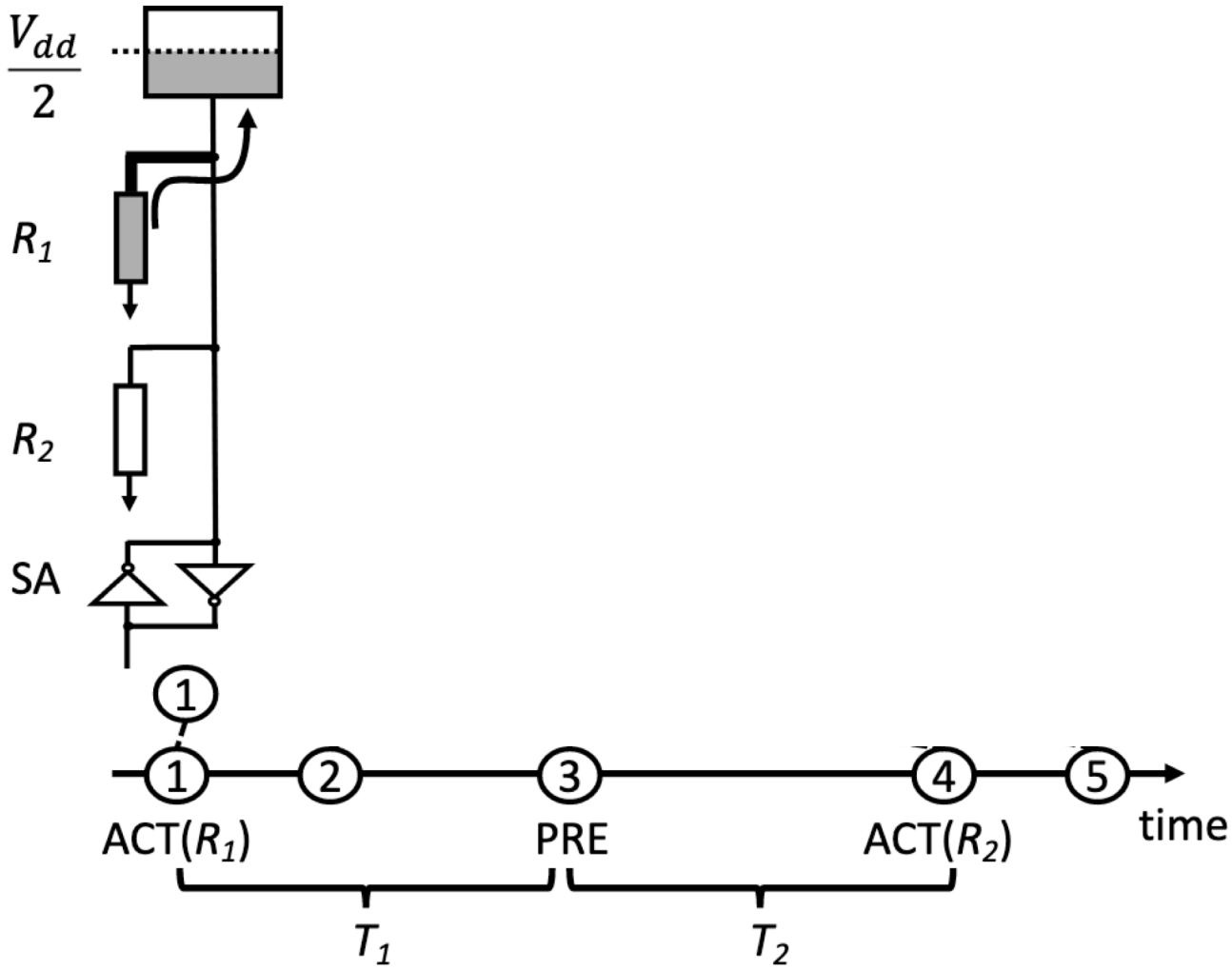
Key Idea:

Violate timing constraints of $T1$ and $T2$ to perform operations.

Mechanism

Performing Row Copy

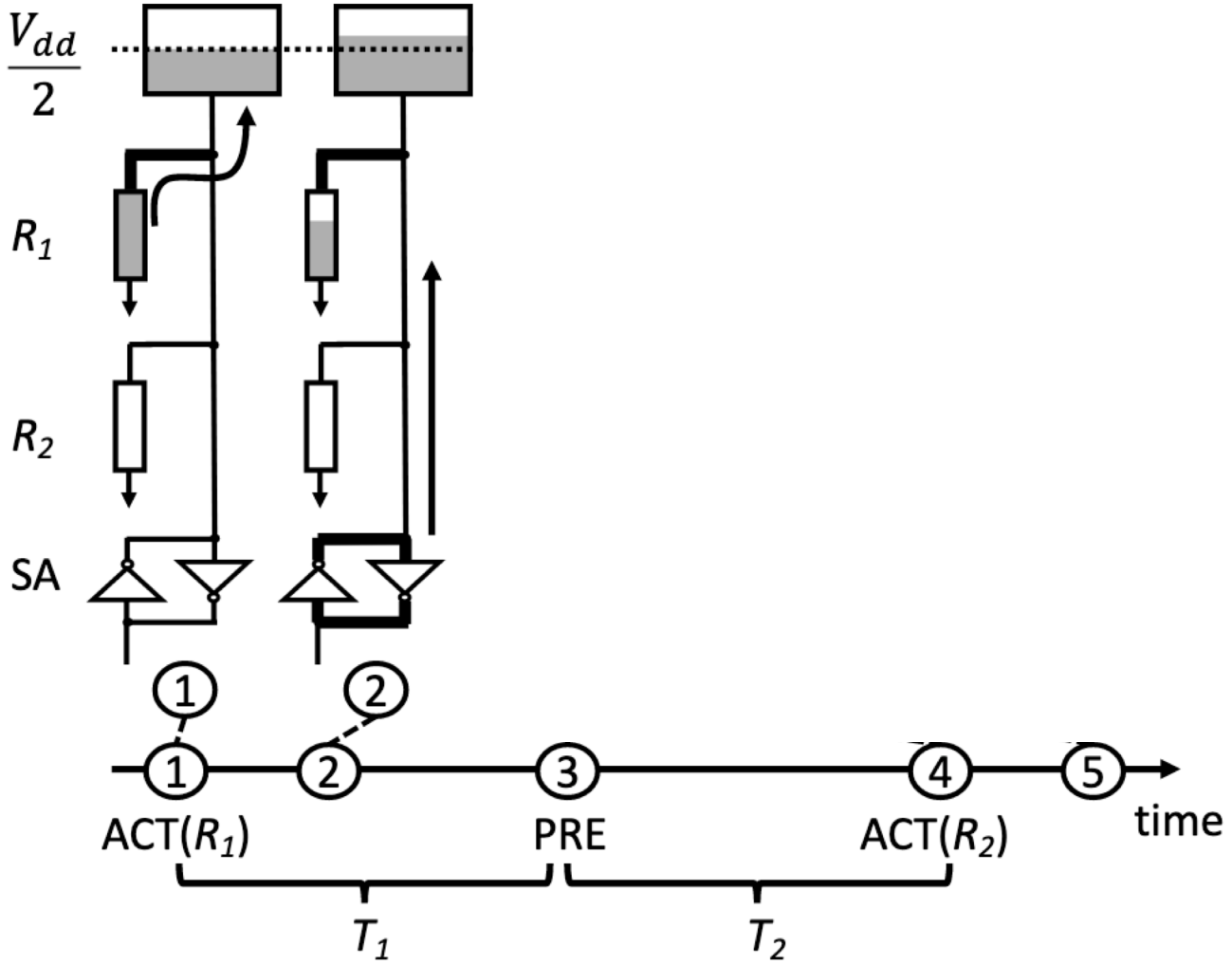
1. Issue Activate R1



Mechanism

Performing Row Copy

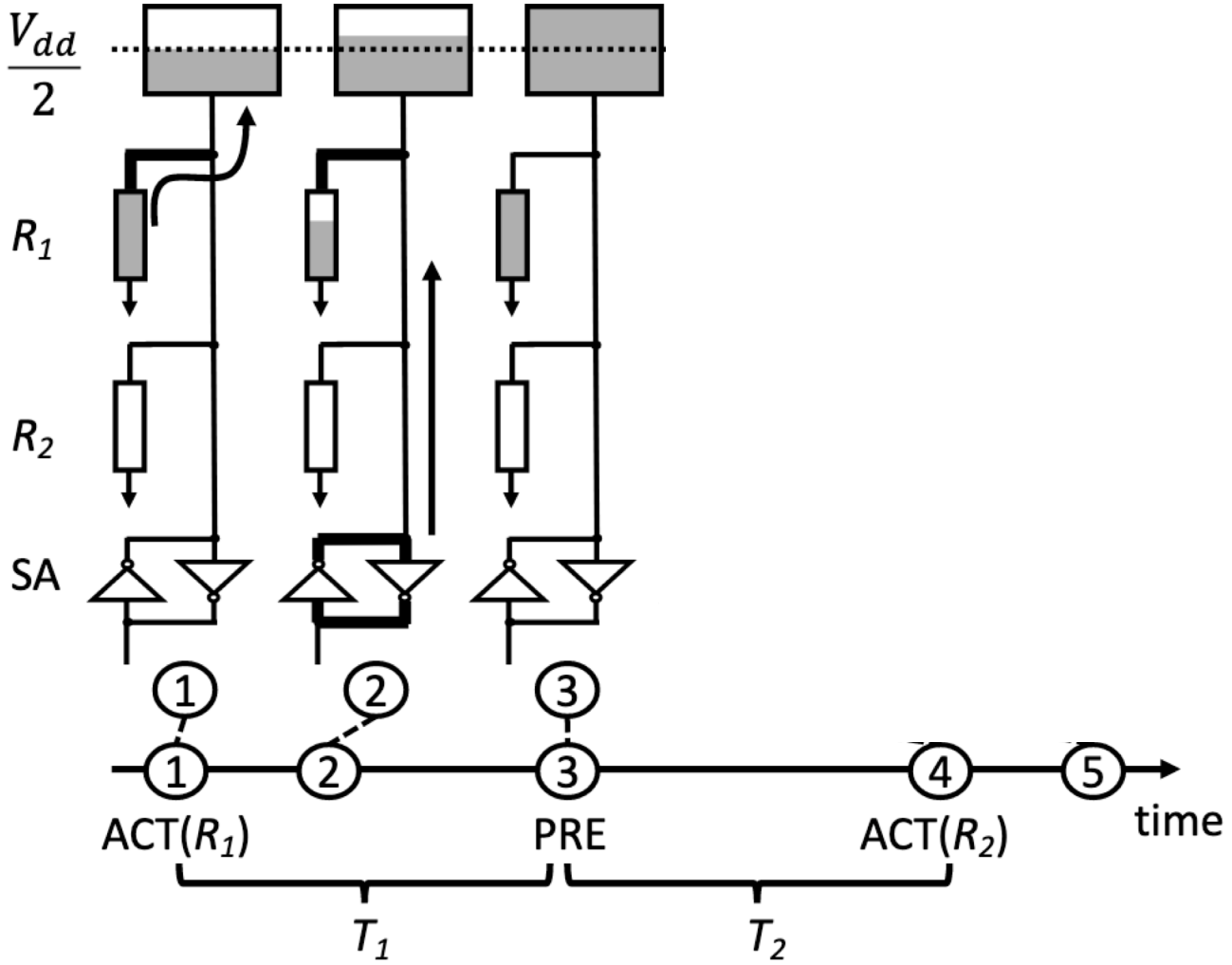
1. Issue Activate R1
2. Bit-line gets amplified



Mechanism

Performing Row Copy

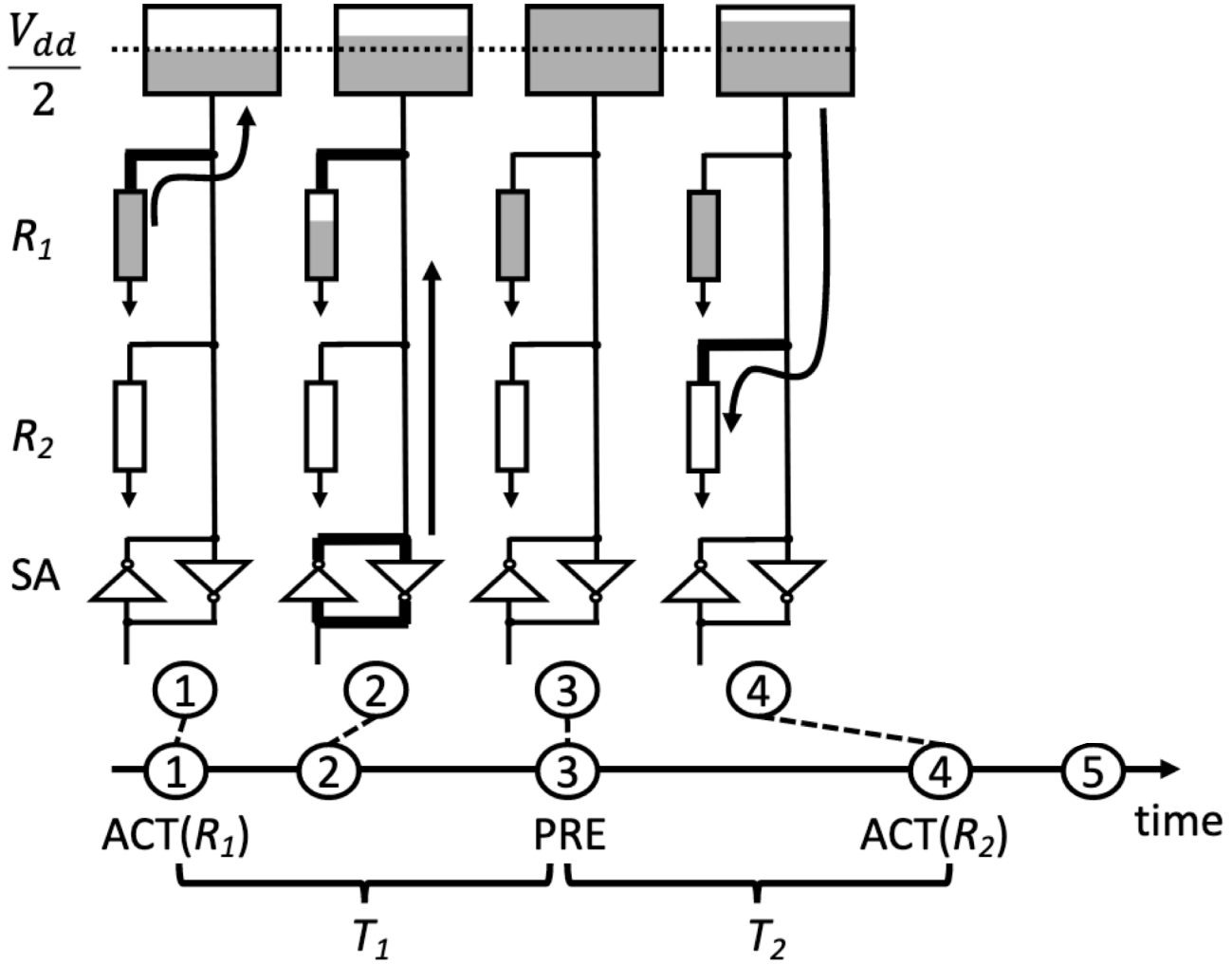
1. Issue Activate R1
2. Bit-line gets amplified
3. Issue Precharge



Mechanism

Performing Row Copy

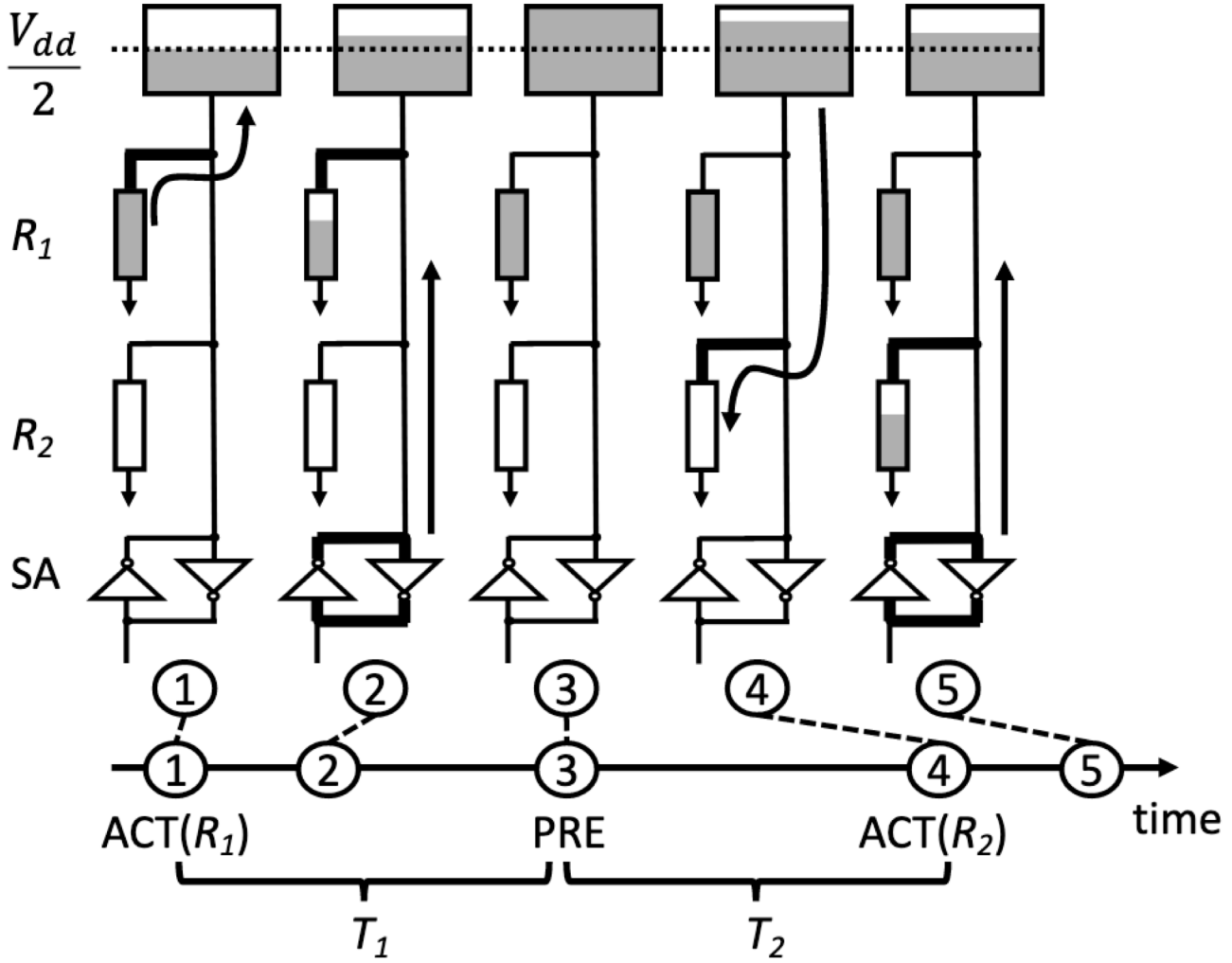
1. Issue Activate R1
2. Bit-line gets amplified
3. Issue Precharge
4.
   - R1 closed, driving $V_{dd}/2$
   - Interrupt Precharge with Activate R2



Mechanism

Performing Row Copy

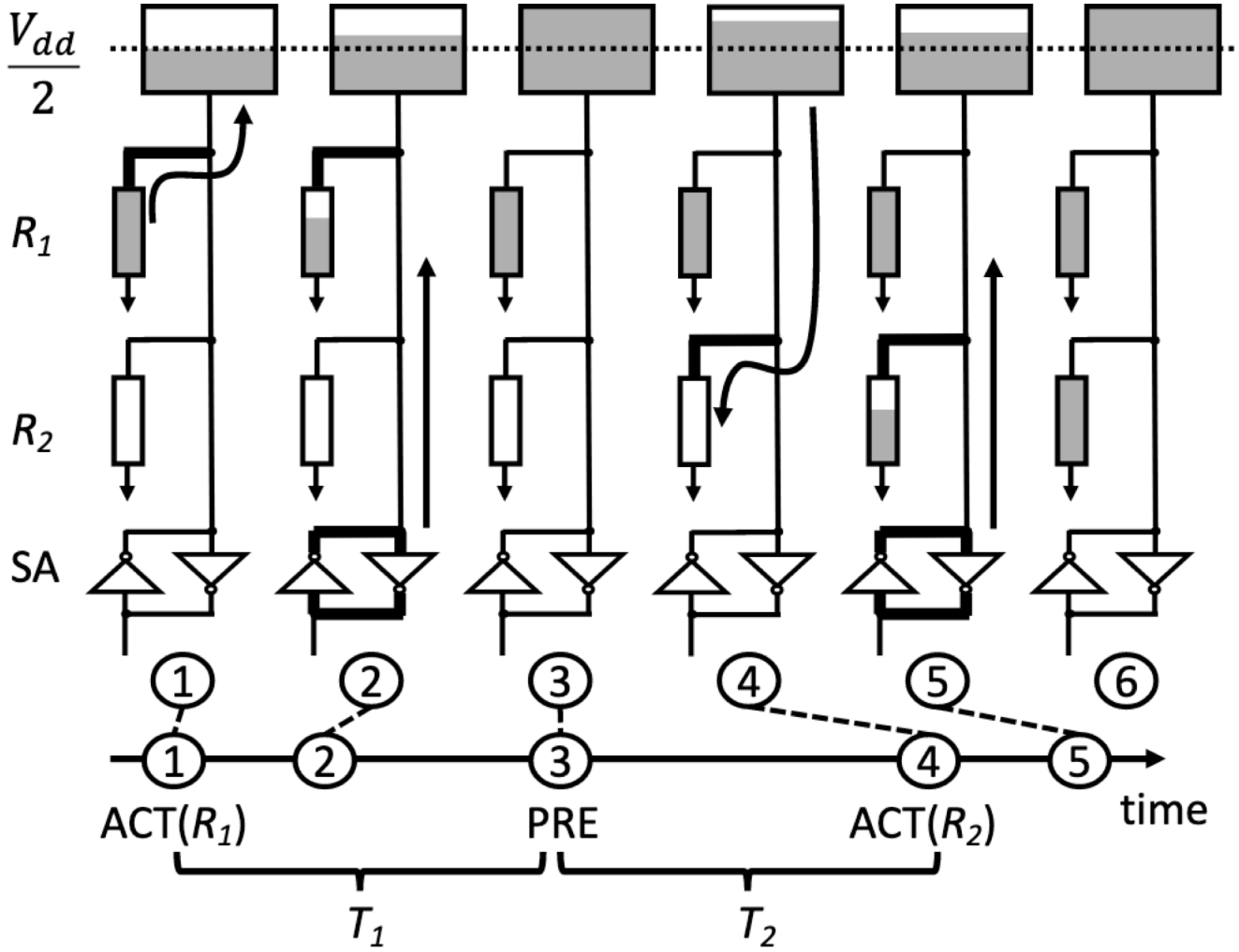
1. Issue Activate R1
2. Bit-line gets amplified
3. Issue Precharge
4.
   - R1 closed, driving $V_{dd}/2$
   - Interrupt Precharge with Activate R2
5. Bit-line and cell of R2 get amplified



Mechanism

Performing Row Copy

1. Issue Activate R1
2. Bit-line gets amplified
3. Issue Precharge
4.
   - R1 closed, driving $V_{dd}/2$
   - Interrupt Precharge with Activate R2
5. Bit-line and cell of R2 get amplified
6. R1 successfully copied to R

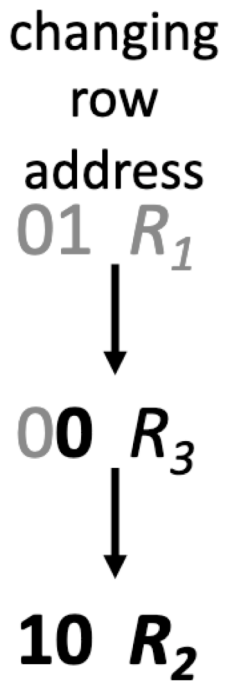


Mechanism

Performing Bulk-Bitwise logical AND/OR

By further reducing $T1$ and $T2$, three different rows can be opened simultaneously.

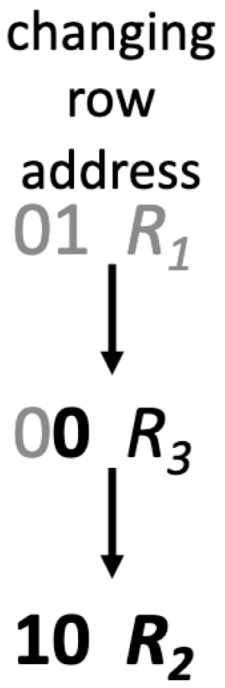
- The second Activate is sent while setting the word-line.
- The word-line according to the value on the row address bus is being driven.
- Intermediate row is being opened as well.



Mechanism

Performing Bulk-Bitwise logical AND/OR

- **Speculation**:
  - Row address is updated from LSB to MSB
- **Note**:
  - The row address update order is dependent on the manufacturer.
  - It will not work the same on every DRAM chip



Operation Reliability

Manufacturing Variations

- Capacitance variations require different timings
- Faulty cells due to manufacturing imperfections
  - Their row addresses are being remapped to another physical location

Implementation

As part of the **proof of concept**, computeDRAM introduces an in-memory compute framework.

In-memory compute framework

- Software interface to perform arbitrary computation using the three basic operations as building blocks.
- Manages the rows, where computations are being executed.
- Addresses the issue of errors due to faulty cells, by introducing an error table.

Implementation

Performing arbitrary computation

- AND and OR are not logically complete on their own, the NOT operation is missing

- Workaround: Save negated values in pairwise fashion with their nominals.

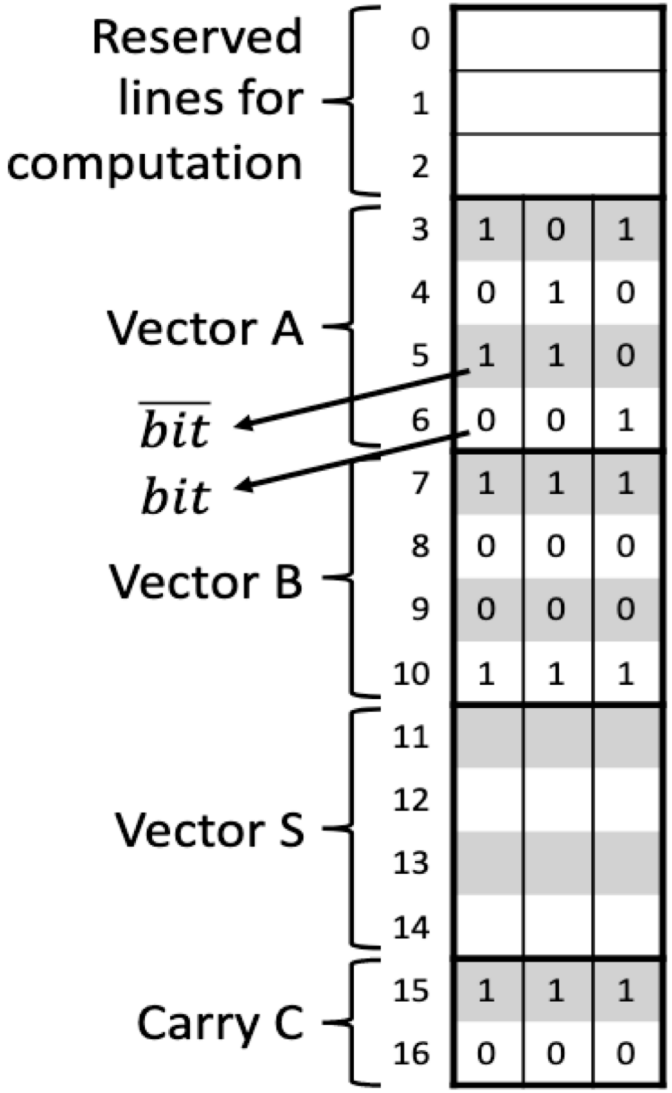
$$\left(A \middle| \overline{A} \right) \quad \left(B \middle| \overline{B} \right) \quad \left(C \middle| \overline{C} \right)$$

- Overhead is quite substantial:
  - Generate the negated pair
  - Double the memory space needed
  - Twice the number of operations needed.

Implementation

Implementation choices

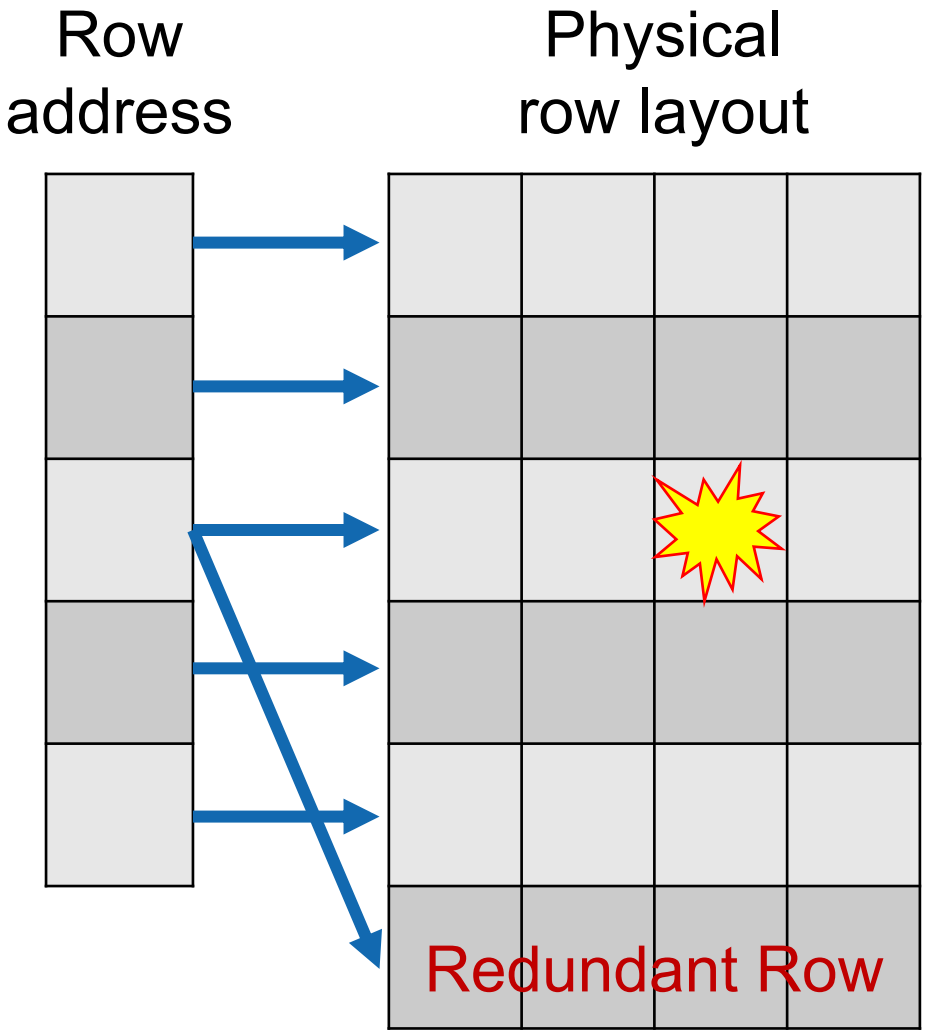
- Computations only performed in the first three rows.
- Operations require a setup:
  1. Copy the operands and the op-constant to these 3 rows
  2. Perform the computation
  3. Copy the result back to the destination row



Implementation

Challenge

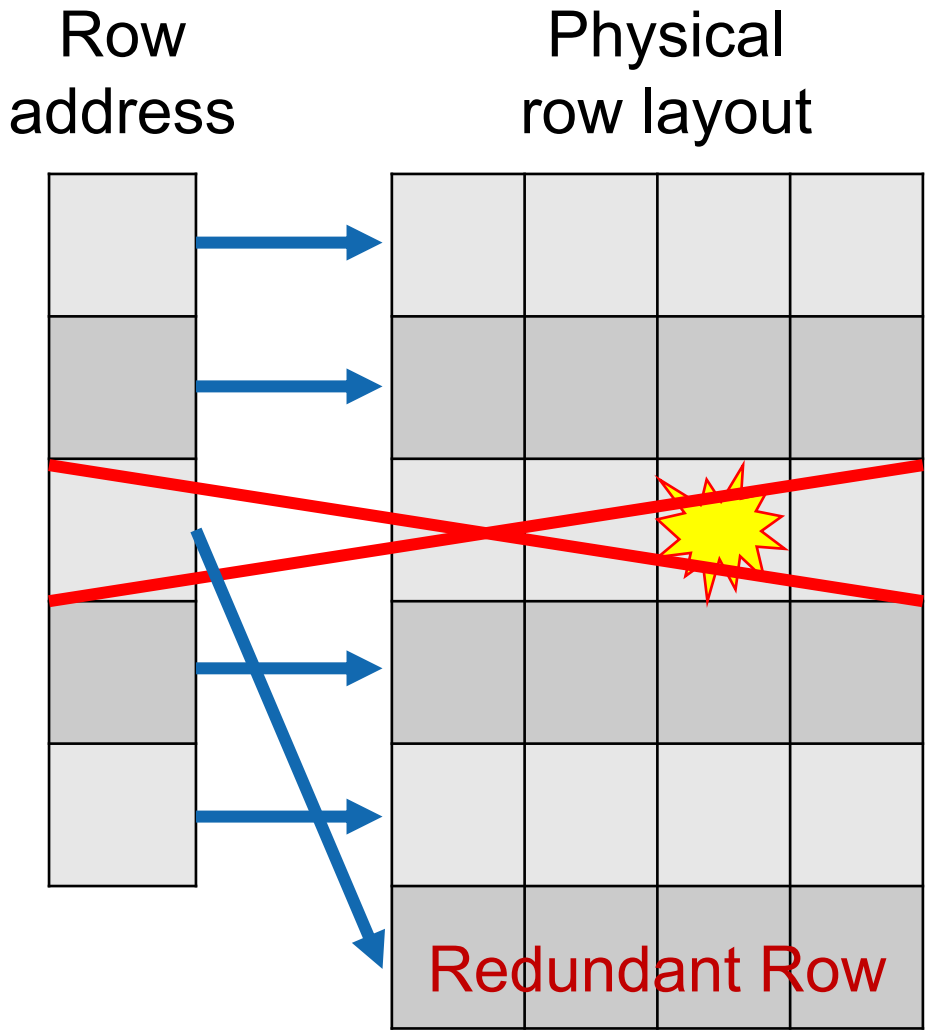
- The library ensures that operand rows are in the same sub-array by checking their address.
- The addresses of remapped rows are not consistent with their physical locations.
- There is no way to guarantee that data is on the same sub-array, as the new row could be anywhere.



Implementation

Solution: Error Table

- Idea: Exclude ”bad” columns and rows from computation with a custom mapping.
- Requires a scanning process to discover ”bad” parts and save them to the error table.
- The error table requires periodical re-scans, due to natural wear out etc.

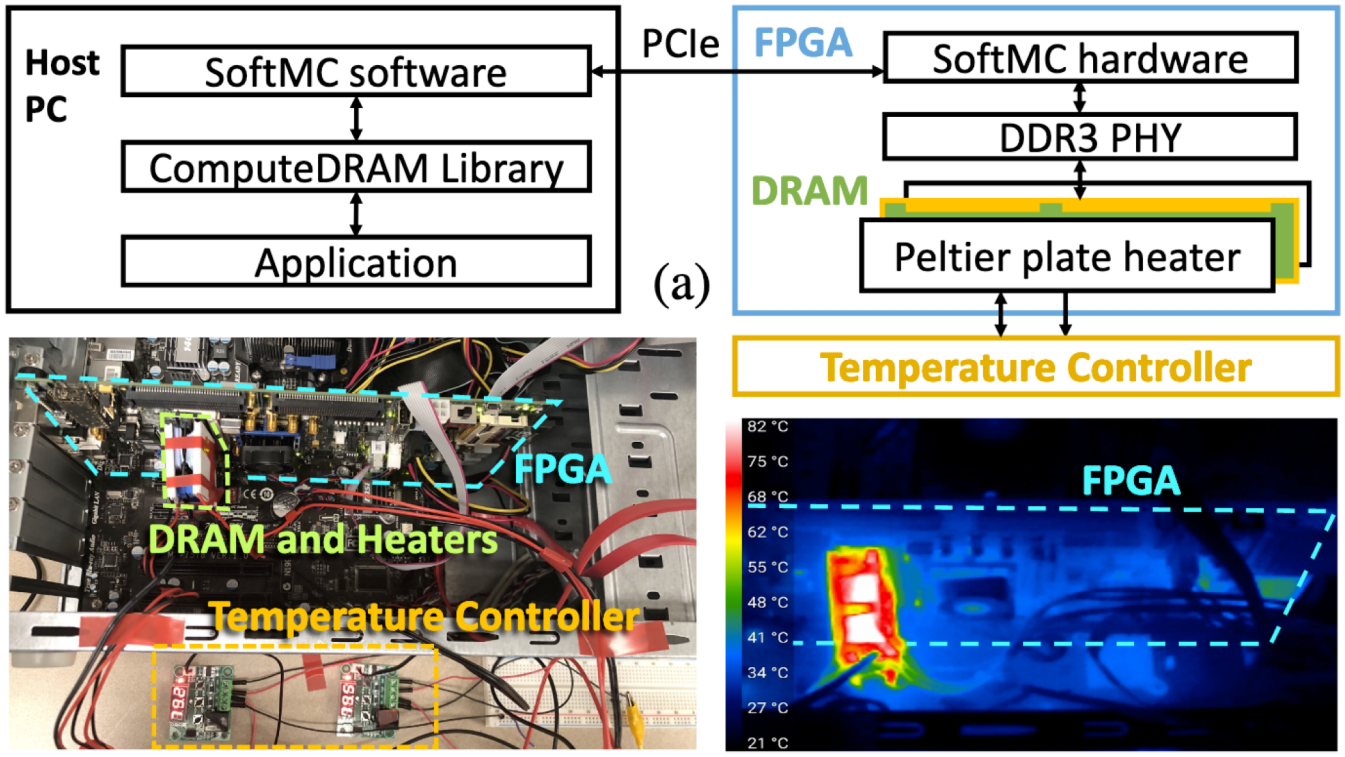


Methodology

- Host system + FPGA running SoftMC to control the DRAM module

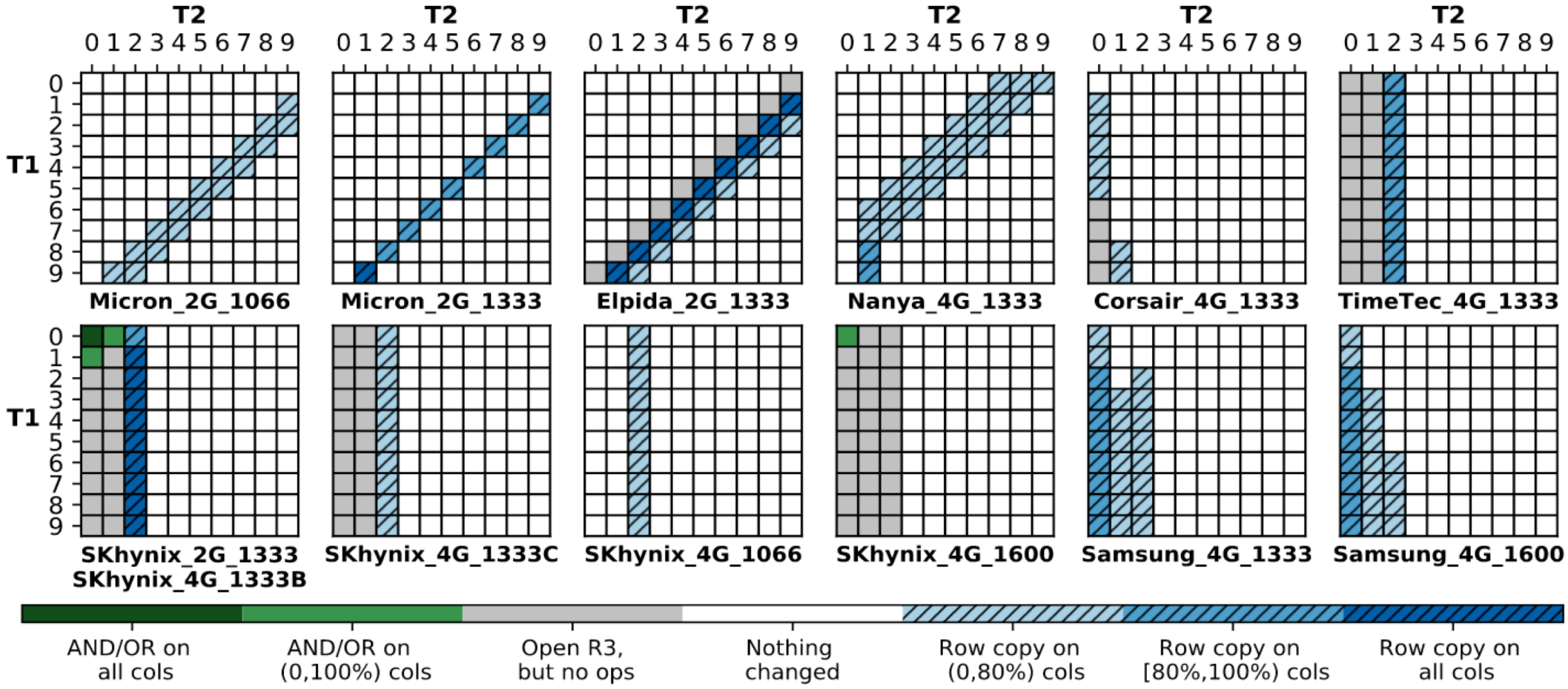
- Limitations:
  - Timing intervals are limited to multiples of 2.5 ns
  - DDR3 chips only

Extensive tests on environment temperature have been made



Evaluation

Which manufacturers work?



Evaluation

Computational Throughput

- Overhead does not change as we move from scalar to vector operations of 64k elements

Energy efficiency

- Eliminates the high energy overhead of transferring data between CPU and main memory.
- 347x more efficient than using a vector unit for row copy.
- 48x more efficient for 8-bit AND/OR
- 9.3x more efficient for 8-bit ADD

Conclusion

- **Motivation**: Proof that AMBIT and RowClone are usable.
- **Goal**: Demonstrate row copy and bit-wise logical AND and OR in unmodified, commercial, DRAM.
- **Key Idea**: Violate DRAM timing constraints to enable charge sharing across multiple rows in the same sub-array.
- **Mechanism**: Perform operations with DRAM, by carefully violating its timing constraints.
- **Implementation**: Provide an in-memory compute framework to allow arbitrary computation.
- **Results**: Enable high computational throughput, up to 347x more energy efficient than using a vector unit.

Strengths

- Working proof of concept
  - No additional hardware required
  - Accessible in form of a library
- Addresses an important problem
- Well written

Weaknesses

- Requirement for pairwise saving of negated values
- Not applicable to every DRAM chip
  - Getting the timings right is substantial
- Requires data to be in the same sub array
- No solution for inter subarray row copy
- Proof of concept
  - No thorough evaluation

Related Work

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*

AlignS: A Processing-In-Memory Accelerator for DNA Short Read Alignment Leveraging SOT-MRAM

Shaahin Angizi[†], Jiao Sun[‡], Wei Zhang[‡] and Deliang Fan[†]

[†] Department of Electrical and Computer Engineering, University of Central Florida, Orlando, FL 32816

[‡] Department of Computer Science, University of Central Florida, Orlando, FL 32816

Related Work

Duality Cache for Data Parallel Acceleration

Daichi Fujiki
dfujiki@umich.edu
University of Michigan

Scott Mahlke
mahlke@umich.edu
University of Michigan

Reetuparna Das
reetudas@umich.edu
University of Michigan

A Case for Exploiting Subarray-Level Parallelism (SALP) in DRAM

Yoongu Kim Vivek Seshadri Donghyuk Lee Jamie Liu Onur Mutlu
Carnegie Mellon University

DRISA: A DRAM-based Reconfigurable In-Situ Accelerator

Shuangchen Li[1] Dimin Niu[2] Krishna T. Malladi[2] Hongzhong Zheng[2]
Bob Brennan[2] Yuan Xie[1]
[1]University of California, Santa Barbara [2]Samsung Semiconductor Inc.

Related Work

DrAcc: a DRAM based Accelerator for Accurate CNN Inference

Quan Deng
College of Computer
National University of Defense Technology
dengquan12@nudt.edu.cn

Lei Jiang
Intelligent Systems Engineering
School of Informatics and Computing
Indiana University Bloomington
jiang60@ie.edu

Youtao Zhang
Computer Science Department
University of Pittsburgh
zhangyt@cs.pitt.edu

Minxuan Zhang
College of Computer
National University of Defense Technology
mxzhang@nudt.edu.cn

Jun Yang
Electrical and Computer Engineering Department
University of Pittsburgh
juy9@pitt.edu

Open Discussion

- Is ComputeDRAM practical for actual use?
  - What overhead is imposed?
  - Do you think the overhead is acceptable?
  - Are there any additional requirements to the system?

- What workloads can benefit from ComputeDRAM?

- Is there a way to enable more general computation?
  - E.g. multiplication, division, floating point arithmetic…
  - Where are the limits in complexity?

Open Discussion

- Will the solution become more important over time?

- What alternatives do you see?

ETH zürich

D INFK

Thank you for your attention!