CROW: A Low-Cost Substrate for Improving DRAM Performance, Energy Efficiency, and Reliability

Hasan Hassan Minesh Patel Jeremie S. Kim A. Giray Yaglikci
Nandita Vijaykumar Nika Mansouri Ghiasi Saugata Ghose Onur Mutlu

ETH zürich
Carnegie Mellon University

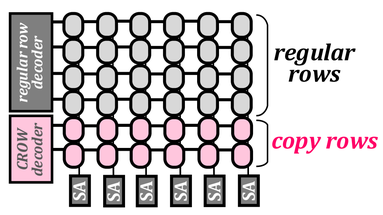
1: Summary

Challenges of DRAM scaling:

- **High access latency** → bottleneck for improving system performance/energy
- **Refresh overhead** → reduces performance and consume high energy
- **Exposure to vulnerabilities** (e.g., RowHammer)

Copy-Row DRAM (CROW)

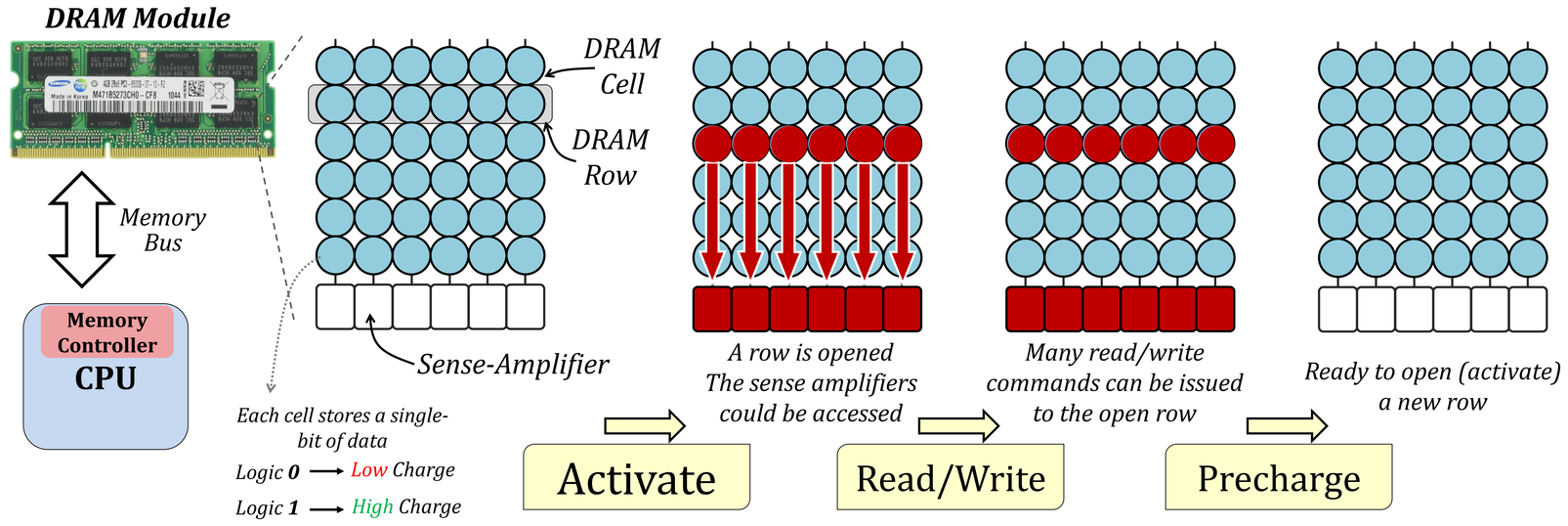
- Introduces copy rows into a subarray
- The benefits of a copy row:
  - Efficiently duplicating data from regular row to a copy row
  - Quick access to a duplicated row
  - Remapping a regular row to a copy row



Use cases:

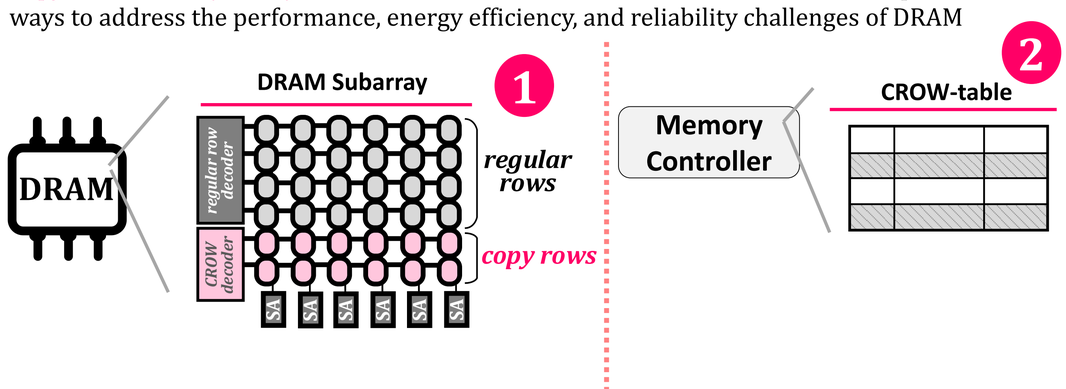
- CROW-cache & CROW-ref **(20% speedup and 22% less DRAM energy)**
- Mitigating RowHammer
- We hope CROW enables many other use cases going forward

2: DRAM Operation Basics



3: The Components of CROW

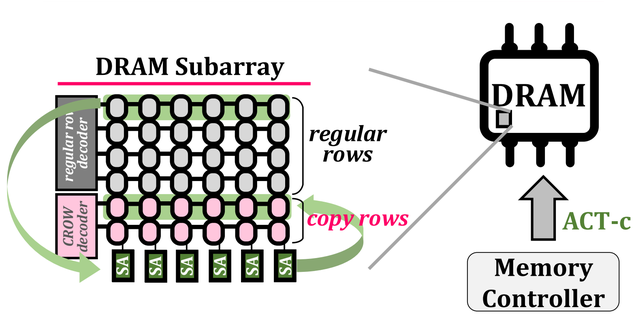
Copy-Row DRAM (CROW): a flexible in-DRAM substrate that can be used in multiple different ways to address the performance, energy efficiency, and reliability challenges of DRAM



4: CROW Operations

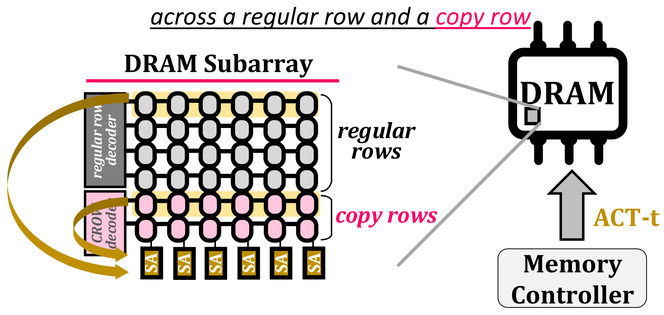
Row Copy

Enables quickly copying a regular row into a copy row



Two-Row Activation

Enables fast access to data that is duplicated across a regular row and a copy row



5: CROW-cache

Problem: High access latency

Key idea: Use copy rows to enable low-latency access to most-recently-activated regular rows in a subarray

CROW-cache combines:

- row copy → copy a newly activated regular row to a copy row
- Two-row activation → activate the regular row and copy row together on next access

Reduces activation latency by **38%**

6: CROW-ref

Problem: Refresh has high overheads. Weak rows lead to high refresh rate

- **weak row:** at least one of the row's cells cannot retain data correctly when refresh interval is increased

Key idea: Avoid storing data in a weak regular row by remapping it to a strong copy row

CROW-ref uses:

- row copy → copy a weak regular row to a strong copy row

CROW-ref **eliminates more than half of the refresh** requests

7: Mitigating RowHammer

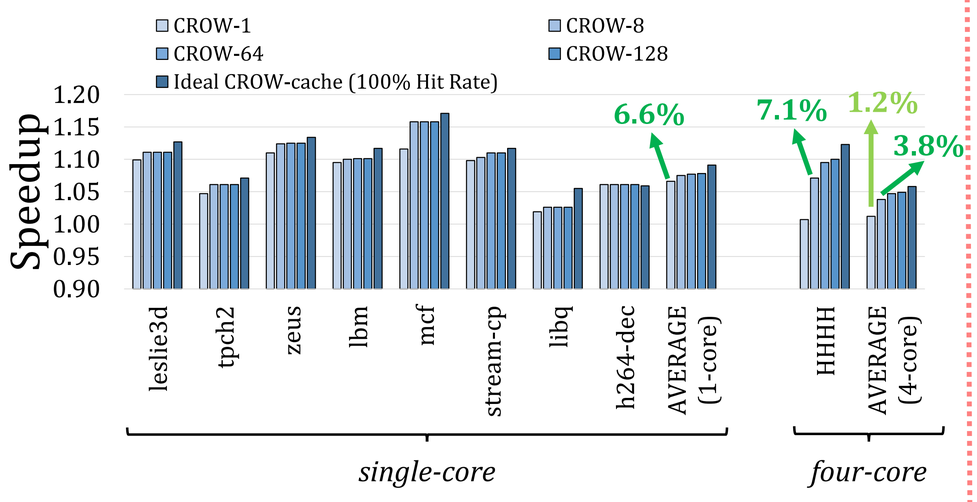
Key idea: remap victim rows to copy rows

8: Evaluation

Methodology

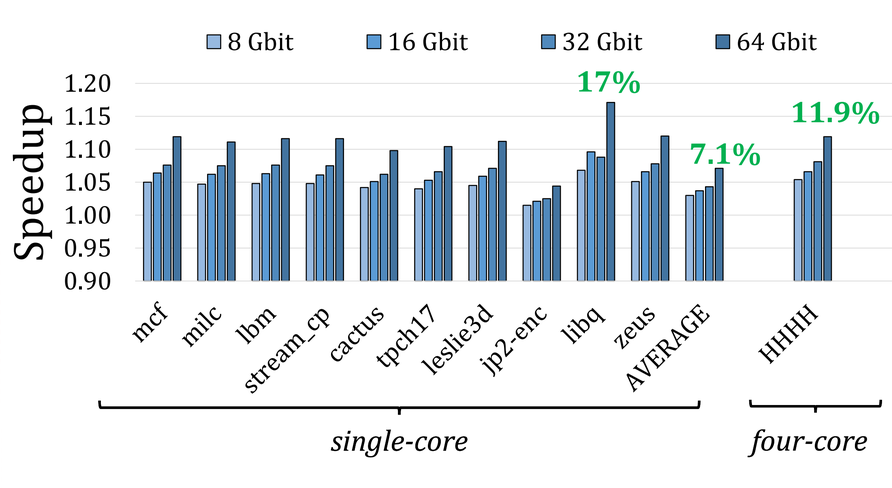
- **Simulator**
  - DRAM Simulator (Ramulator *[Kim+, CAL'15]*)
    https://github.com/CMU-SAFARI/ramulator
- **Workloads**
  - 44 single-core workloads
    - *SPEC CPU2006, TPC, STREAM, MediaBench*
  - 160 multi-programmed four-core workloads
    - *By randomly choosing from single-core workloads*
  - Execute at least 200 million representative instructions per core
- **System Parameters**
  - 1/4 core system with 8 MiB LLC
  - LPDDR4 main memory
  - 8 copy rows per 512-row subarray

CROW-cache Performance



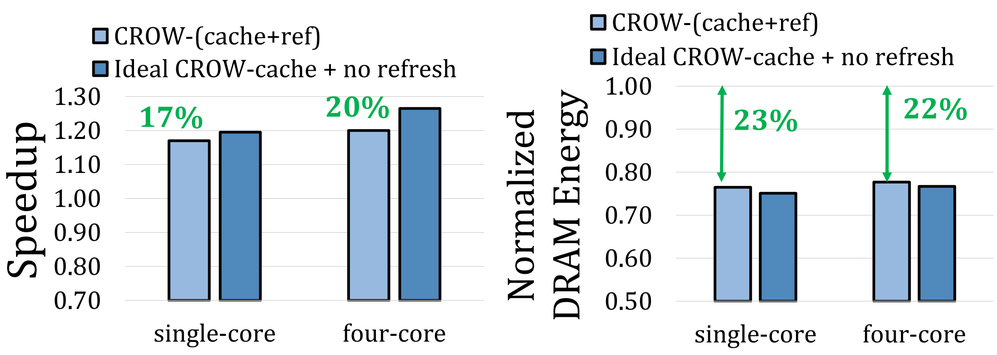
CROW-cache **improves** single-/four-core system **performance**

CROW-ref Performance



CROW-ref significantly reduces the performance overhead of DRAM refresh

Combining CROW-cache and CROW-ref



CROW-(cache+ref) provides more performance and DRAM energy benefits than each mechanism alone

Hardware Overhead

For 8 copy rows:

- 0.5% DRAM chip area
- 1.6% DRAM capacity
- 11.3 KiB memory controller storage

Other Results in the Paper

- Sensitivity to:
  - Number of copy-rows per subarray
  - Chip density
  - Last-level cache capacity
- CROW-cache with prefetching
- CROW-cache against other in-DRAM caching mechanisms:
  - TL-DRAM *[Lee+, HPCA'13]*
  - SALP *[Kim+, ISCA'12]*

Available in July:

github.com/CMU-SAFARI/CROW

SAFARI