Reducing DRAM Latency via Charge-Level-Aware Look-Ahead Partial Restoration

Yaohua Wang[1,2] Arash Tavakkol[1] Lois Orosa[1,4] Saugata Ghose[3] Nika Mansouri Ghiasi[1]
Minesh Patel[1] Jeremie S. Kim[1] Hasan Hassan[1] Mohammad Sadrosadati[1] Onur Mutlu[1,3]

[1] ETH zürich [2] National University of Defense Technology [3] Carnegie Mellon [4] University of Campinas

1: Summary

DRAM latency is a major bottleneck:

1) Mainly consists of: activation and restoration
2) Partial restoration on soon-to-be refreshed cells can help to reduce restoration latency

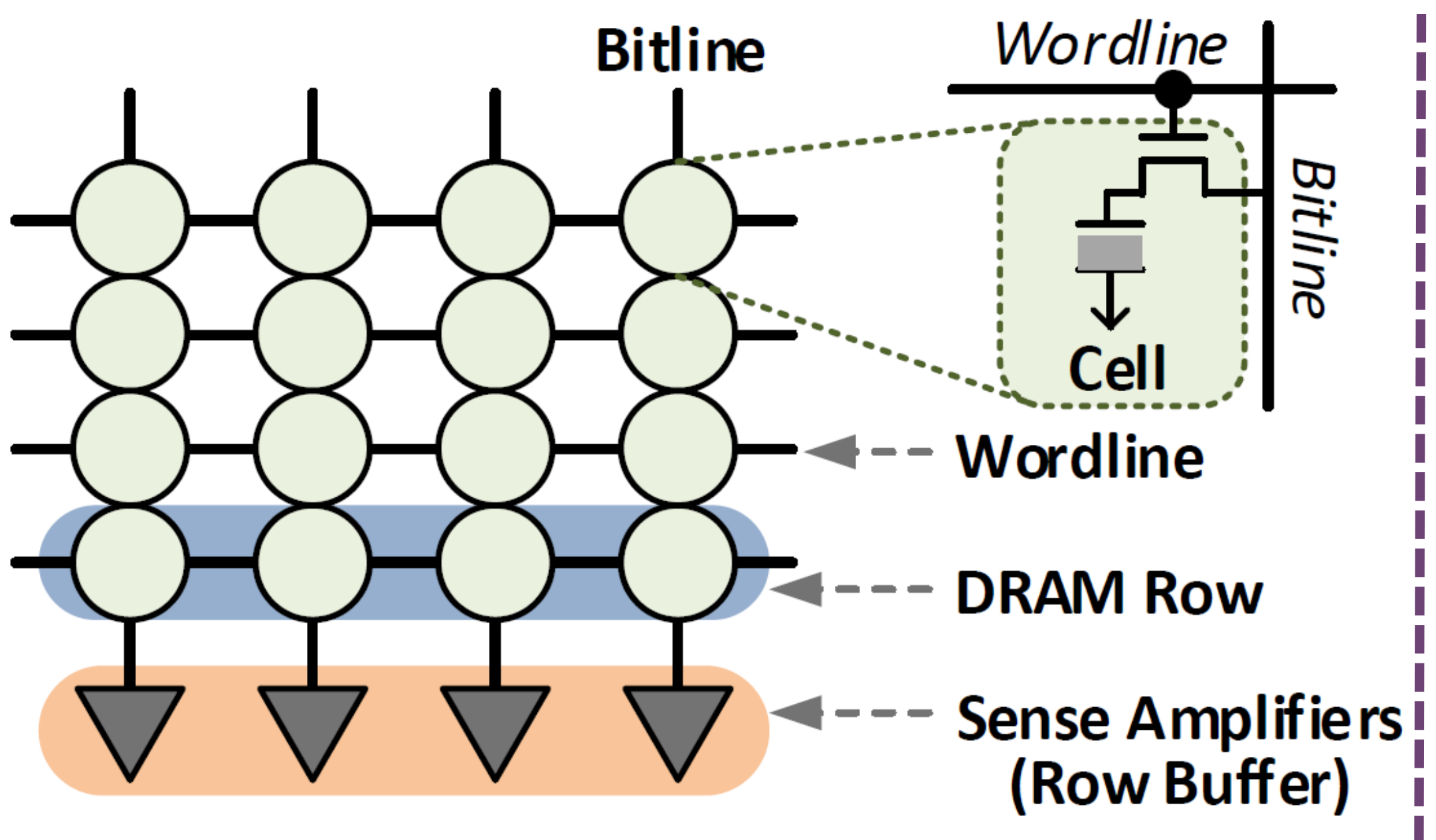
Motivation:

1) The potential of partial restoration can be greatly improved when applied on soon-to-be-reactivated cells
2) We can trade-off restoration and activation latency reductions to maximize the overall benefit

Charge-level-aware look-ahead Partial restoration (CAL):

- Accurately predict the next access-to-access interval
- Carefully apply partial restoration according to the prediction and next scheduled refresh
- Greatly improve overall performance and energy efficiency at low cost

2: Background and Key Ideas

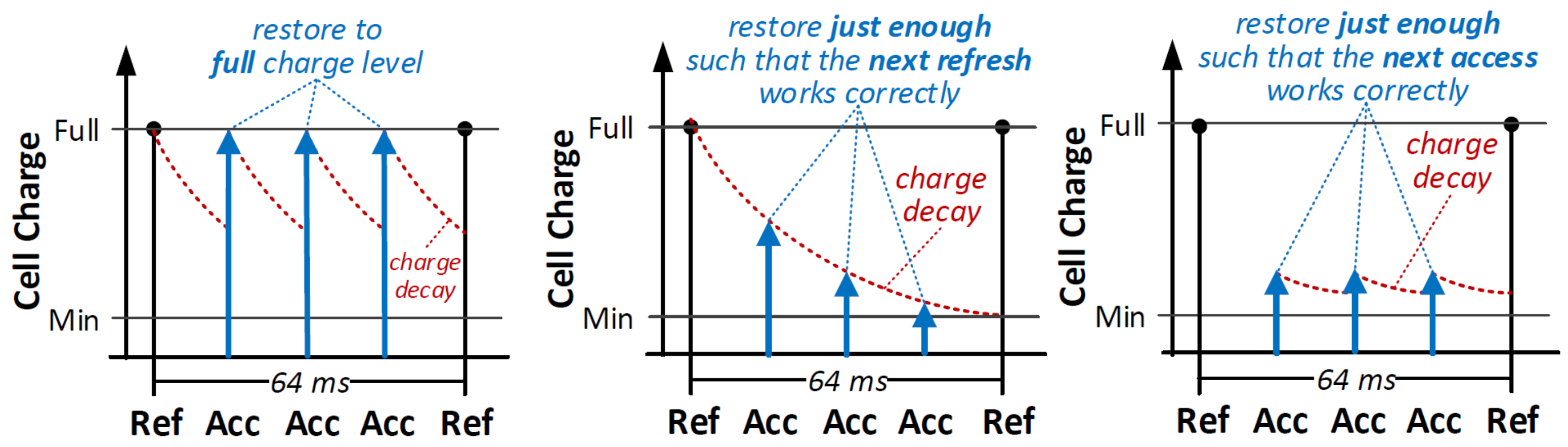


DRAM subarray structure

Two ciritial parts of DRAM access latency:

1) Activation: Sensing and amplifying the charge of cell (tRCD)

2) Restoration: Restoring the charge of cell after access (tRAS)

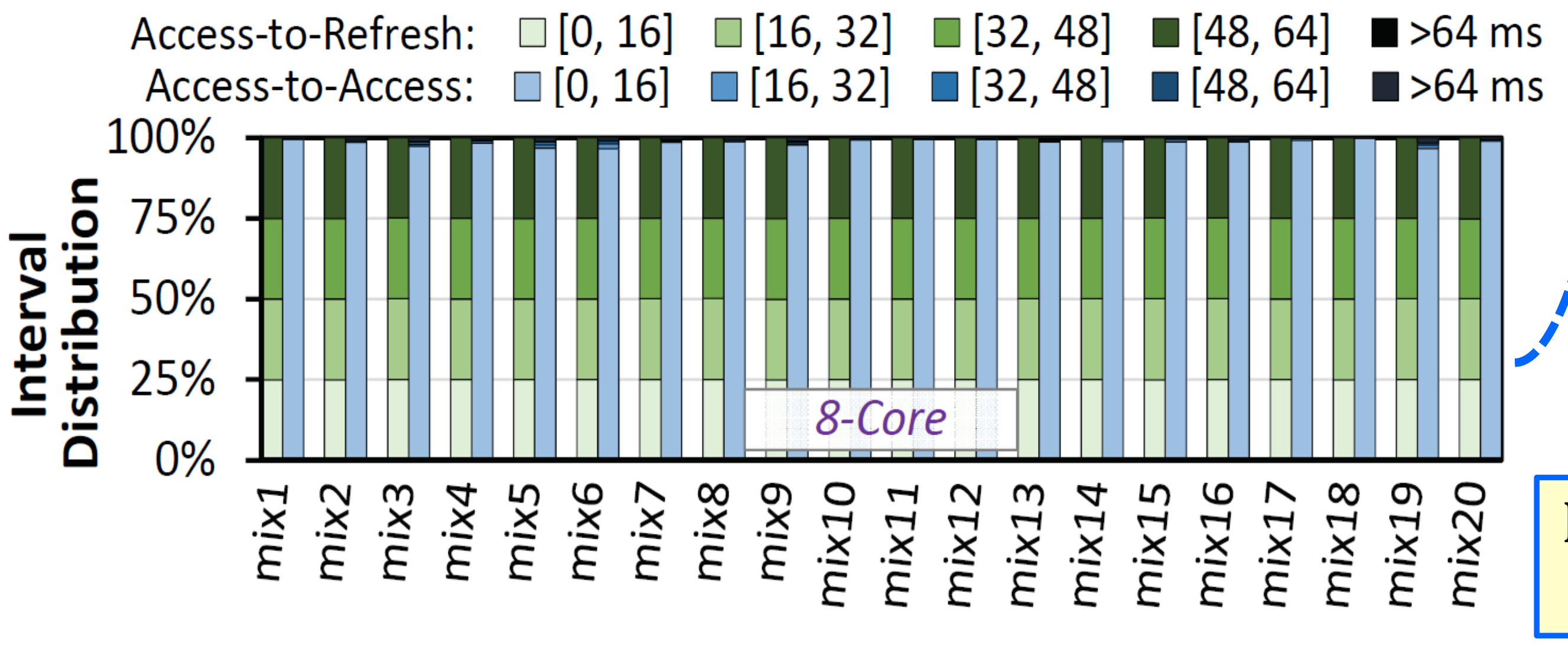


(a) Normal Restoration (b) Refresh-Based Partial Restoratior (c) CAL Partial Restoration

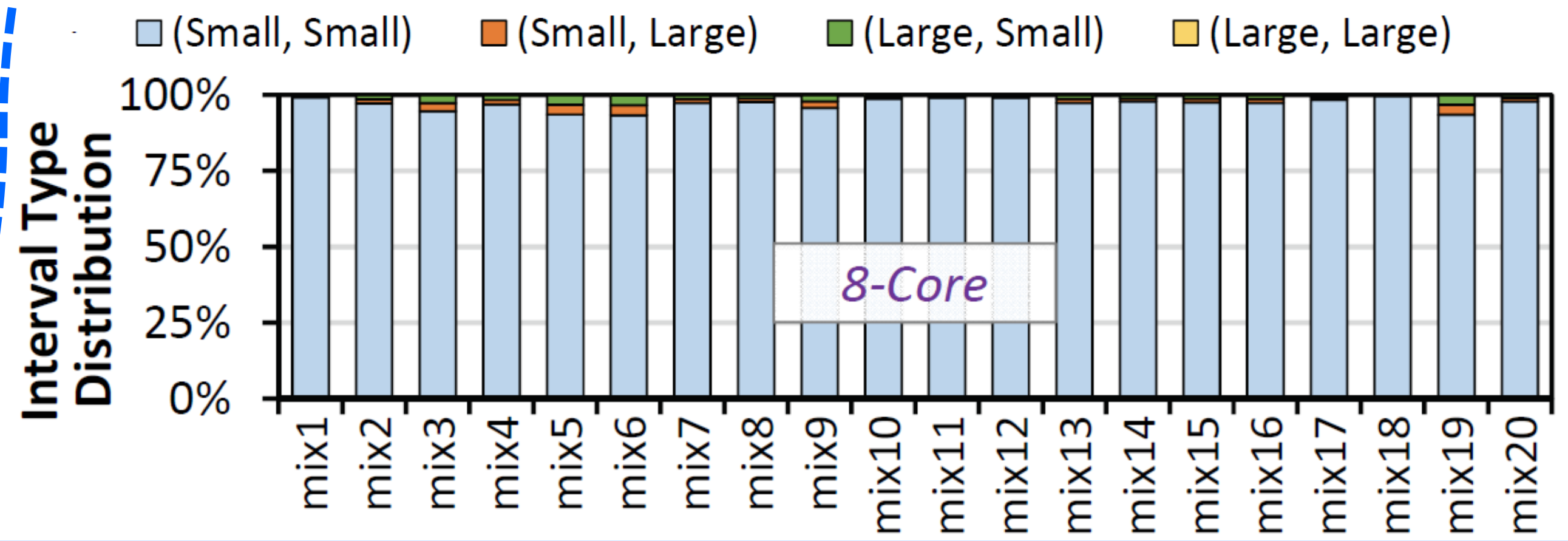
(a) To compensate for the charge depletion and avoid data loss, DRAM fully restores the charge level of the cell during access; **(b)** Restore Truncation[Zhang+,HPCA 2016] partially restores the charge of soon-to-be-refreshed cell to a level, such that the amount of charge is just enough to ensure that the refresh operation can still correctly read the data; **(c)** CAL effectively enable partial restoration for both soon-to-be-refreshed and soon-to-be-reactivated cells, while still effectively exploiting the benefits of activation latency reduction

3: Motivations

Applying partial restoration on soon-to-be-reactivated DRAM cells can provide larger benefit 1

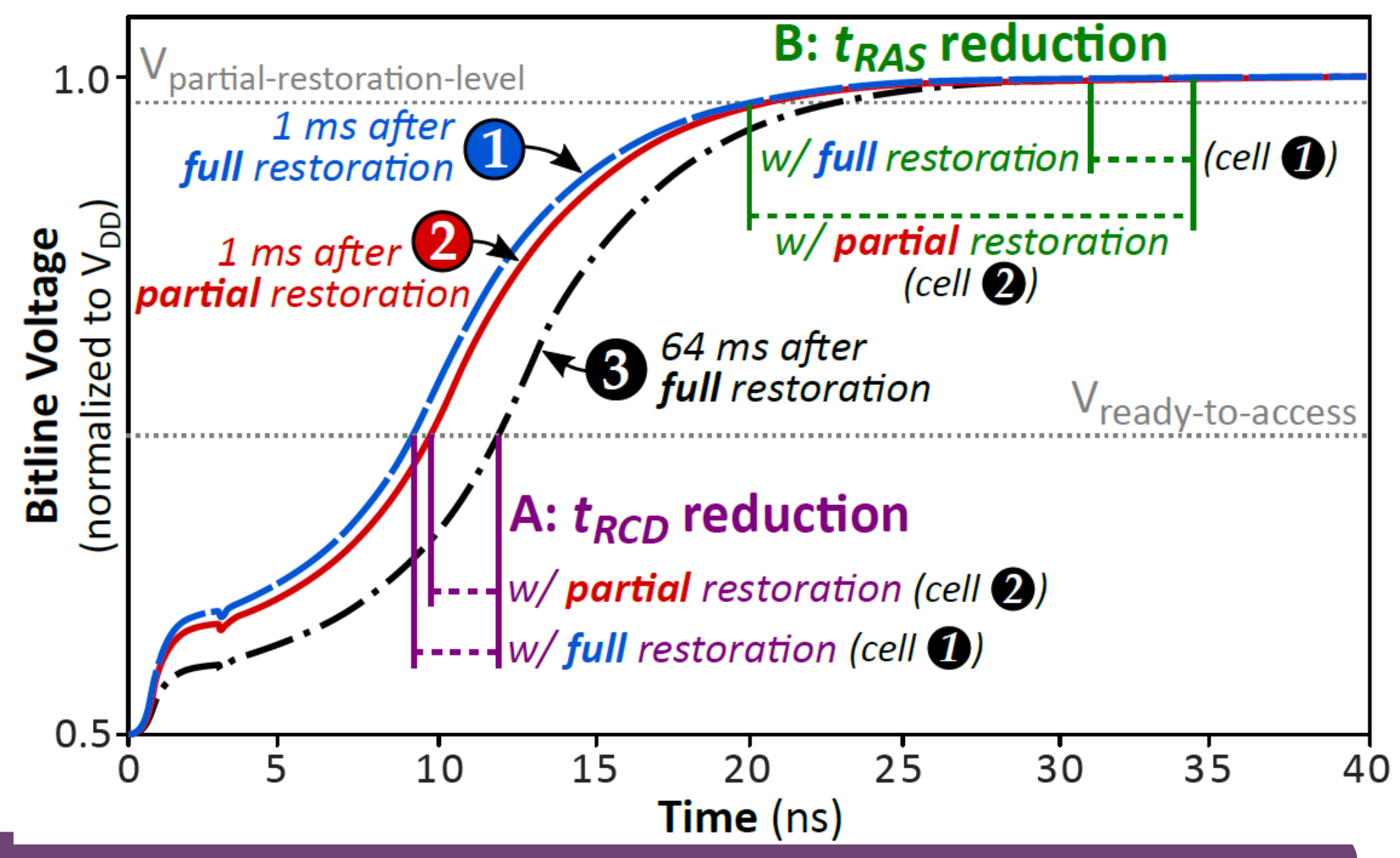


How to know the next access-to-access interval?



If the last access-to-access interval is small (e.g., <16ms), the next one would highly likely to be small (i.e., 98%)

A charge level aware partial restoration can trade a smaller tRCD reduction for a larger tRAS reduction. 2



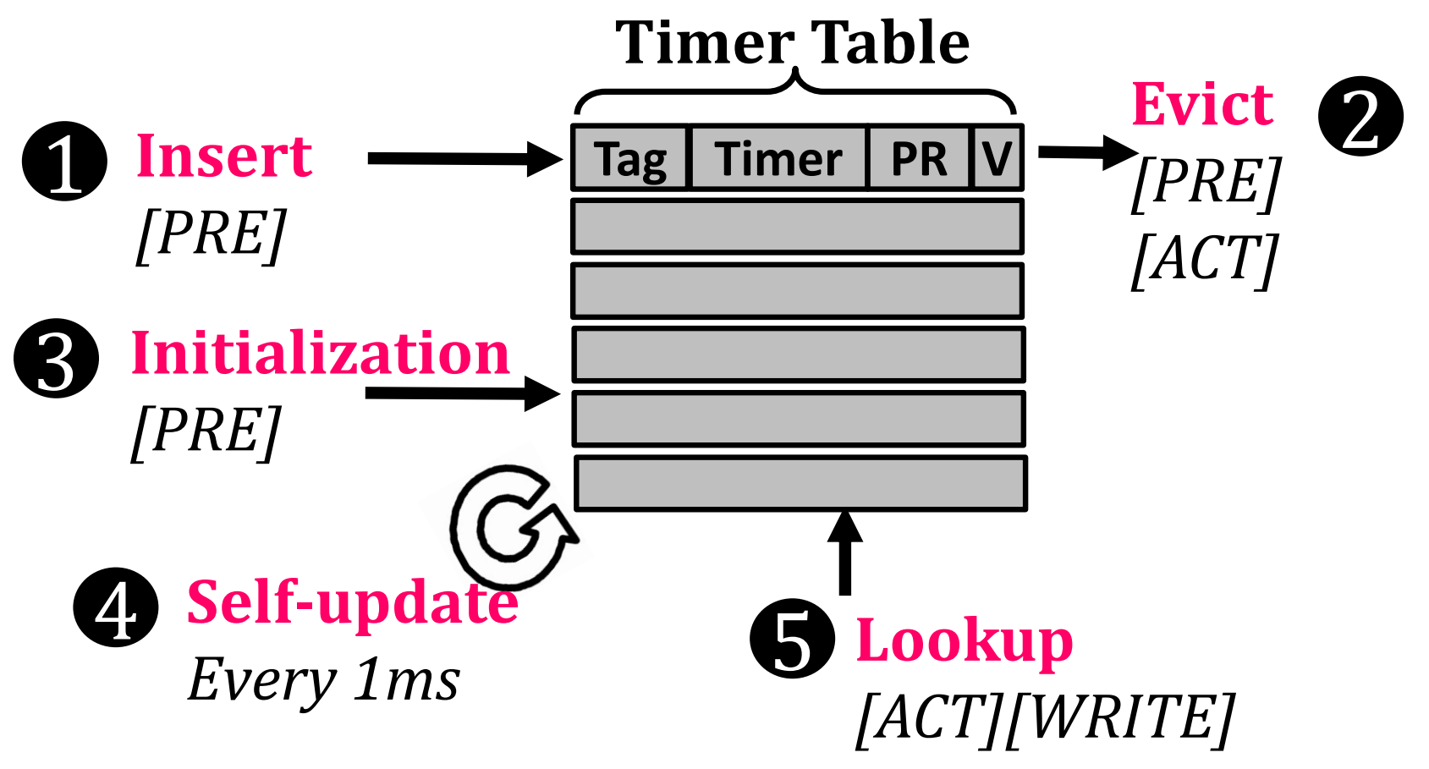
4: Charge-Level-Aware Look-Ahead Partial Restoration

Key Mechanism

1. Uses the last access-to-access interval of a row to **predict** whether the row will be reactivated again soon

2. Decides by **how much** the restoration latency should be reduced, based on the prediction and trade-off

Hardware Structure



Tag: row number, **Timer**: 4-bit timer, **PR**: partially restored, **V**: valid

High Level Operations

- ➢ *Insertion: New items are inserted upon PRE command, potentially evicting other items, and issuing ACT and PRE commands*
- ➢ *Initialization: The timer value is initialized upon PRE command*
- ➢ *Update: The timer table performs an update every 1ms to record access intervals*
- ➢ *Lookup: Each ACT/WRITE incurs a lookup in the timer table to reduce timing parameters accordingly*

Reduced Timing Parameters

Three cases to reduce tRCD/tRAS/tWR according to the timer value (T):

➢T == 15: < 1ms since last restoration 11.2/16.1/6.8ns

➢0 < T < 15: 1-15ms since last restoration 13.75/19.4/8.4ns

➢T == 0: >15ms passed (or lookup miss) Default parameters

5: Evaluation

Methodology

Simulator

DRAM Simulator (Ramulator *[Kim+, CAL'15]*)
https://github.com/CMU-SAFARI/ramulator

Workloads

20 single-core workloads
SPEC CPU2006, TPC, BioBench

20 multi-programmed 8-core workloads
By randomly choosing from single-core workloads

Mechanism Parameters

8-way cache-like set-associative timer table

DDR4 timing tRCD/tRAS/tWR: 13.75/35/15ns

Mechanisms Evaluated

- ➢ **CAL**
- ➢ **ChargeCache (CC)** *[Hassan+, HPCA'16]*

reduces tRCD and tRAS for highly-charged rows

- ➢ **Restore Truncation (RT)** *[Xian+, HPCA16]*

reduces tRAS and tWR for soon-to-be-refreshed rows

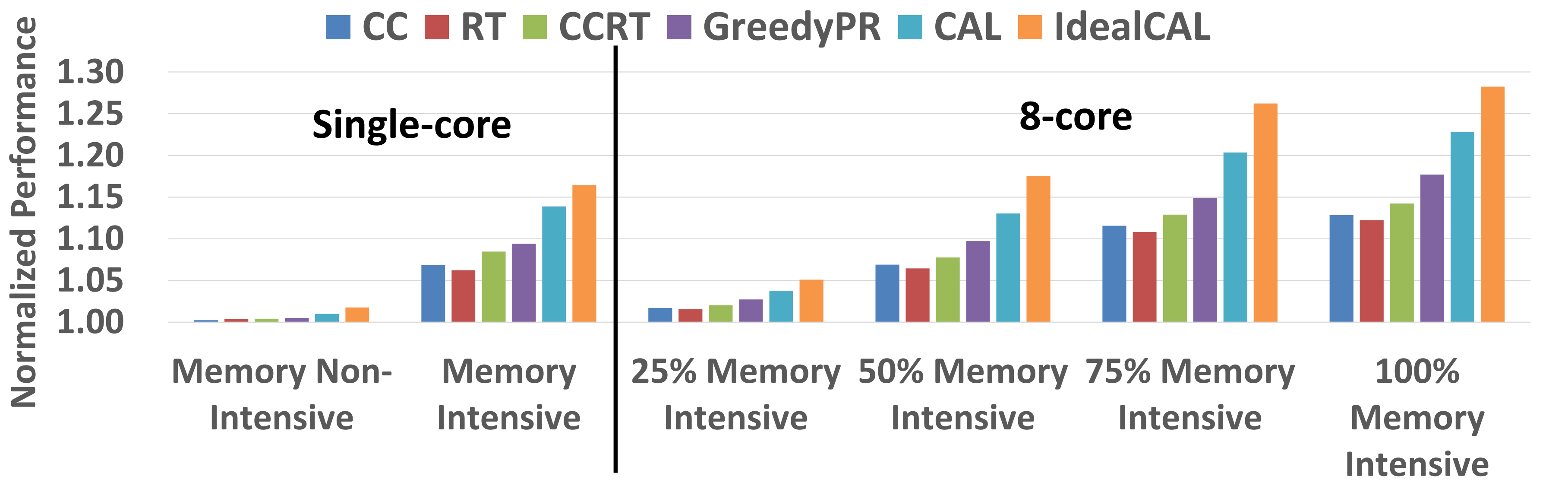
- ➢ **Combinations of activation and restoration latency reductions**

CCRT: a simple combination of CC and RT

GreedyPR: similar to CAL, but unaware of future charge level

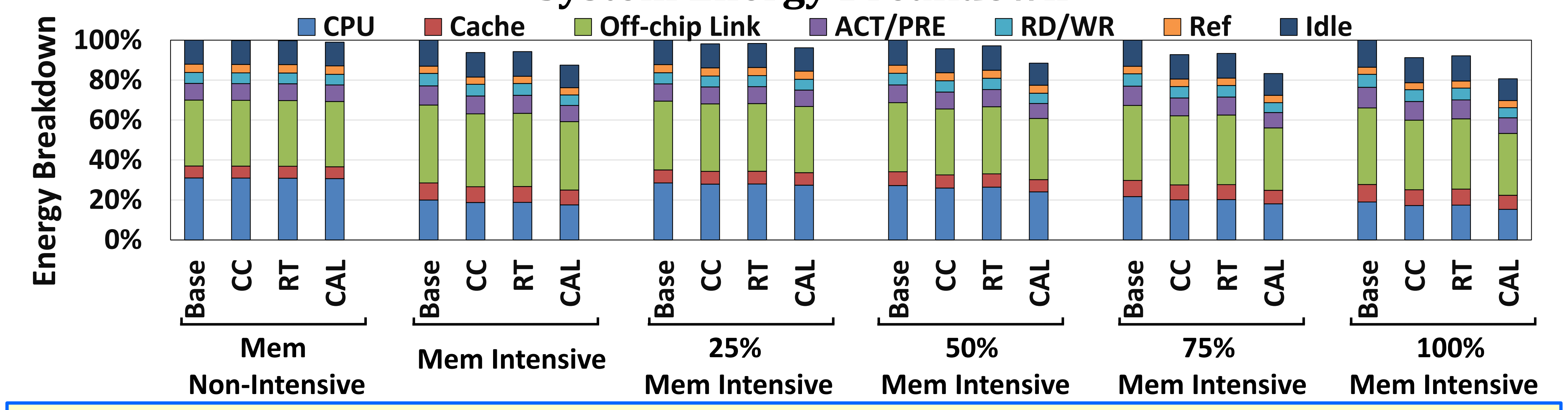
- ➢ **Idealized CAL (IdealCAL)**

Performance Improvement



CAL always outperforms the other mechanisms By an average of 7.4%(14.7%) for single-core (8-core) workloads

System Energy Breakdown



On average, CAL reduces system energy by 10.1% and 18.9% for memory intensive single-core and 8-core workloads

6: Hardware Overhead & Sensitivity Analysis

- ➢ **Area: 0.034mm2, 0.11% of 16MB LLC**
- ➢ **Power: 0.202mW, 0.08% of 16MB LLC**

- ➢ **CAL's performance is robust across various configurations**
  - TC table size
  - Page management policy
  - Address mapping policy

- ➢ **Partial restoration level plays an important role, trading tRCD with tRAS reduction can provide opportunities for performance gain**

- ➢ **CAL's is still effective to high temperatures, where the refresh rate is increased**

SAFARI