RAIDR: Retention-Aware Intelligent DRAM Refresh

Jamie Liu
jamiel@cmu.edu

Ben Jaiyen
bjaiyen@cmu.edu

Richard Veras
rveras@cmu.edu

Onur Mutlu
onur@cmu.edu

Carnegie Mellon University
Presented at ISCA 2012

Presented by Sabria Karim
ETH Zürich
31 October 2019

Executive Summary

- Motivation: DRAM refresh operations waste energy and the performance overhead is high.
- Problem: Not all DRAM cells need same refresh rate.
- Goals: to minimize the number of refresh operations performed without increasing hardware or software much. RAIDR categorizes cells and only refreshes those more often which also require it.
- Evaluation: RAIDR achieves in a 32 GB DRAM: 74.6% refresh reduction, an average DRAM power reduction of 16.1% and an average system performance improvement of 8.6% over existing system.

  And the memory controller only needs 1.25KB additional.

Outline

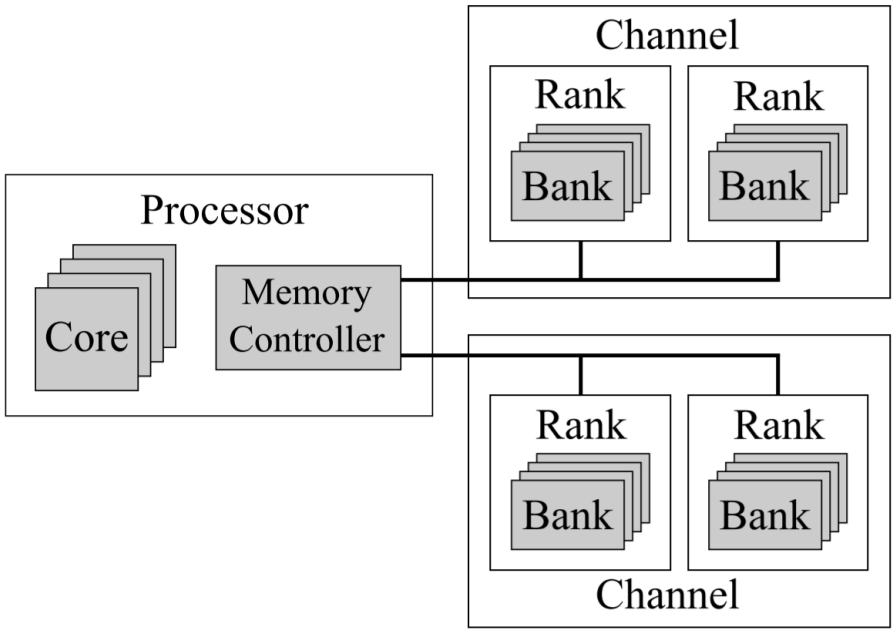
- Executive Summary
- Background & Motivation
- Key Approach, Ideas & Mechanisms
- Key Results: Methodology and Evaluation
- Summary
- Novelty
- Strength
- Weaknesses
- Thoughts and Ideas
- Takeaways
- Open Discussion

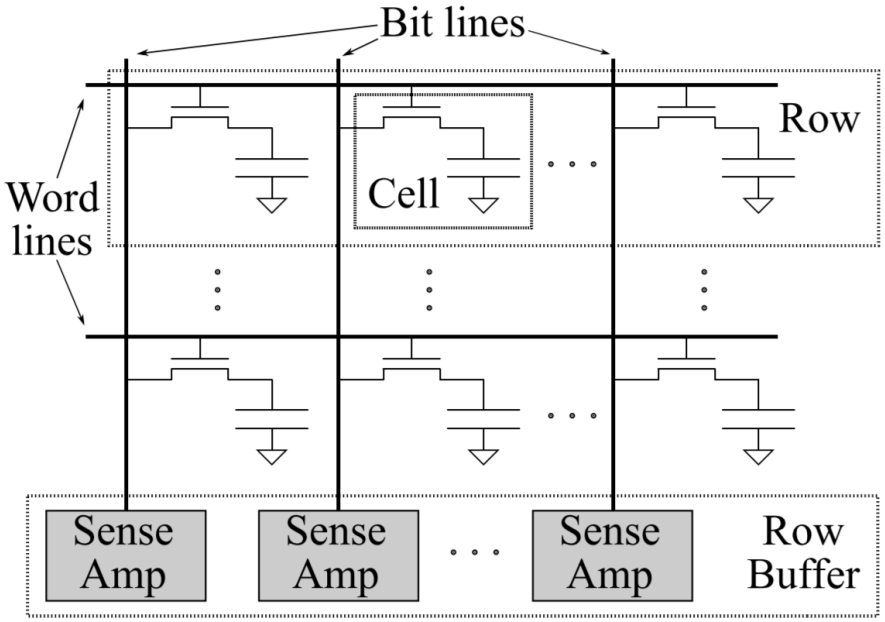
Outline

- Executive Summary
- Background & Motivation
- Key Approach, Ideas & Mechanisms
- Key Results: Methodology and Evaluation
- Summary
- Novelty

- Strength
- Weaknesses
- Thoughts and Ideas
- Takeaways
- Open Discussion

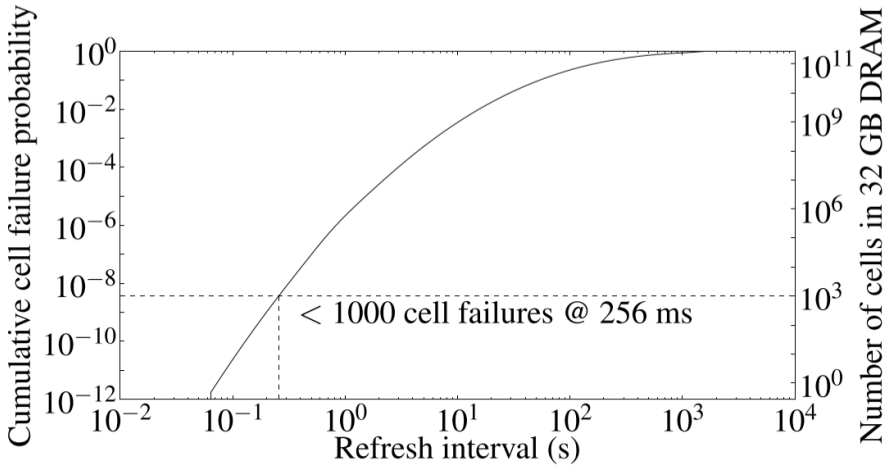
DRAM hierarchy & bank structure





Retention Time & DRAM Refresh

- Retention time := max. time a DRAM cell can keep its stored data without being refreshed



Refresh operation degrades performance:

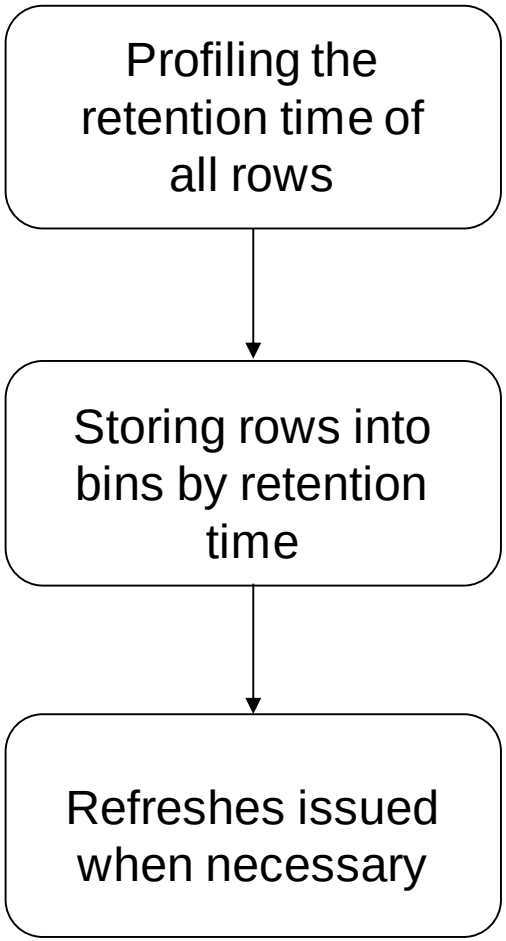
- Loss of bank level parallelism
- Increased memory access latency
- Decreased row hit rate

Outline

- Executive Summary
- Background & Motivation
- Key Approach, Ideas & Mechanisms
- Key Results: Methodology and Evaluation
- Summary
- Novelty

- Strength
- Weaknesses
- Thoughts and Ideas
- Takeaways
- Open Discussion

RAIDR Overview

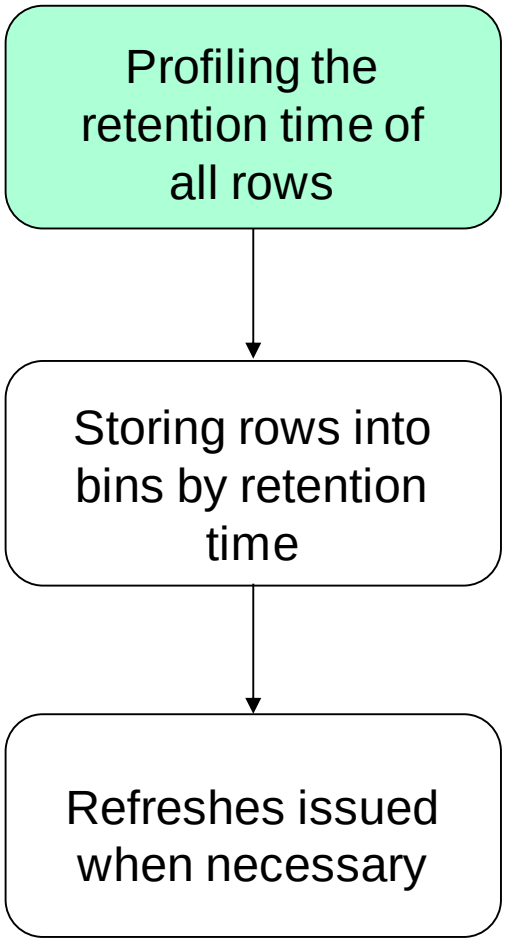


Retention time of each row is measured by deciding which refresh rate is needed.

The memory controller stores through a Bloom filter each row into bins by retention time.

Every 64ms the memory controller decides if the row of cell needs to be refreshed according to its bin.

RAIDR Overview



Retention time of each row is measured by deciding which refresh rate is needed.

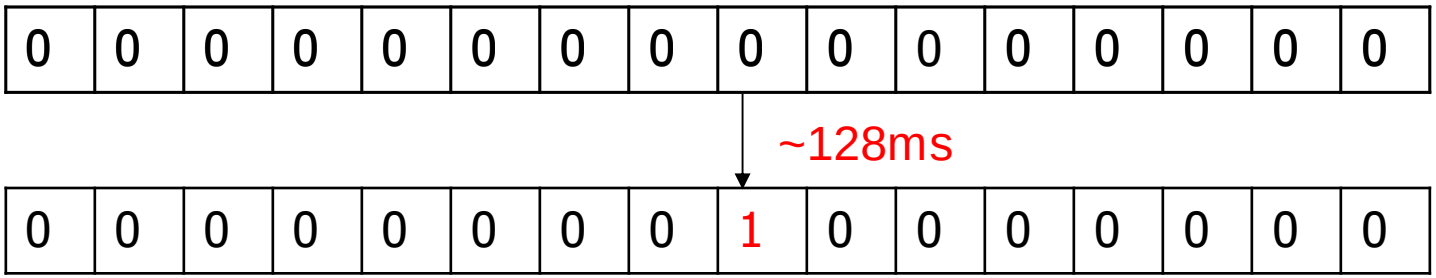
The memory controller stores through a Bloom filter each row into bins by retention time.

Every 64ms the memory controller decides if the row of cell needs to be refreshed according to its bin.

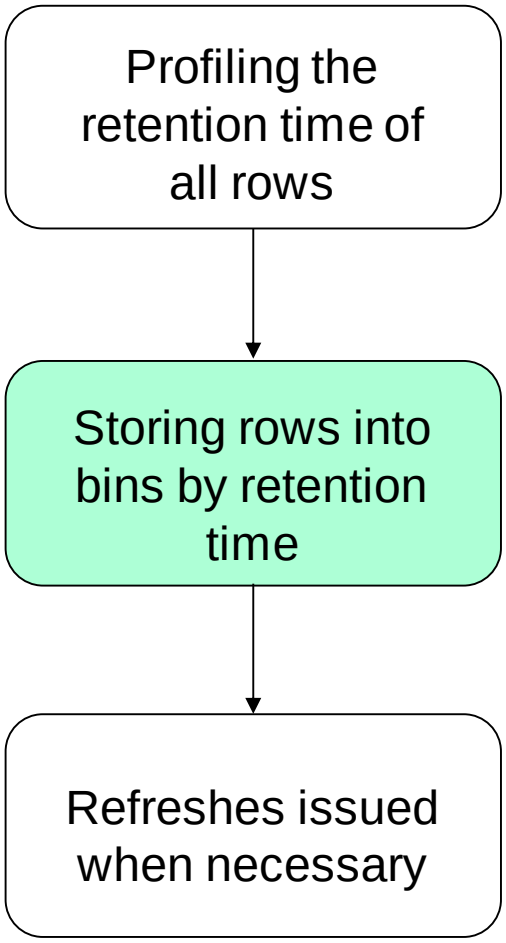
Profiling the retention time of all rows

Simple method:

1. Write all 0's or 1's into the row
2. Don't refresh
3. Observe the first bit change



RAIDR Overview



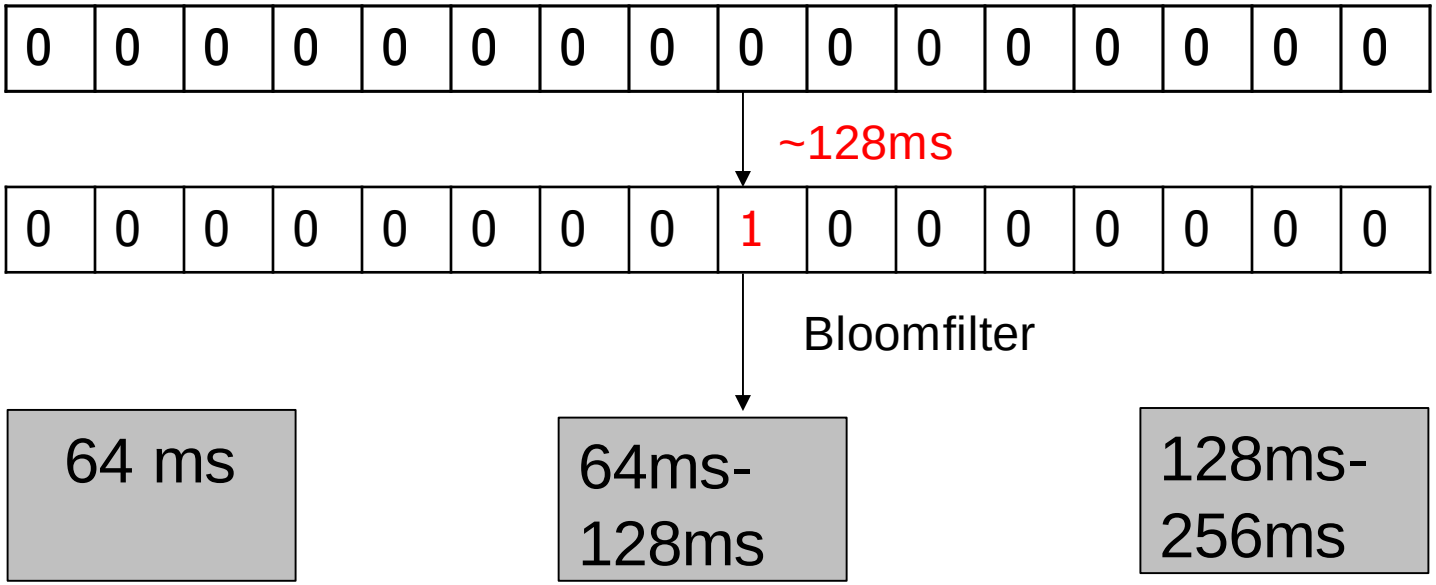
Retention time of each row is measured by deciding which refresh rate is needed.

The memory controller stores through a Bloom filter each row into bins by retention time.

Every 64ms the memory controller decides if the row of cell needs to be refreshed according to its bin.

Storing DRAM rows into the bins

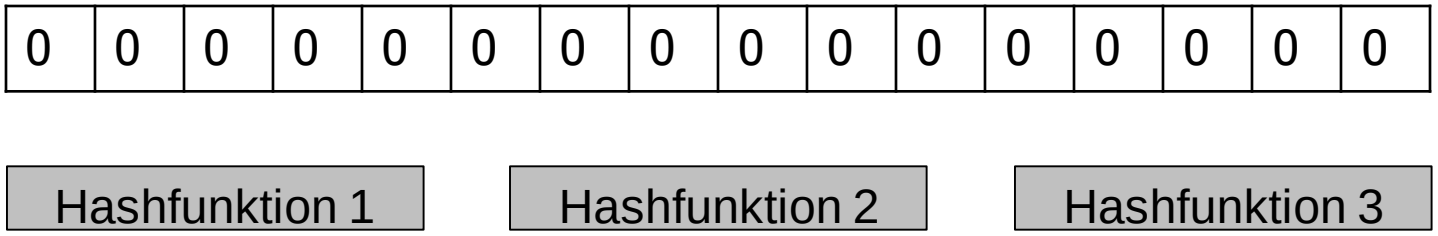
- The memory controller stores each row into one bin based on their profiled retention time.
- Bloomfilter: data structure which is used to determine if an element is part of a set. It consists of a bit-array and some hashfunctions.



Example: storing rows into bins

Example: for 64-128ms bin, 3 hash functions over 16bit

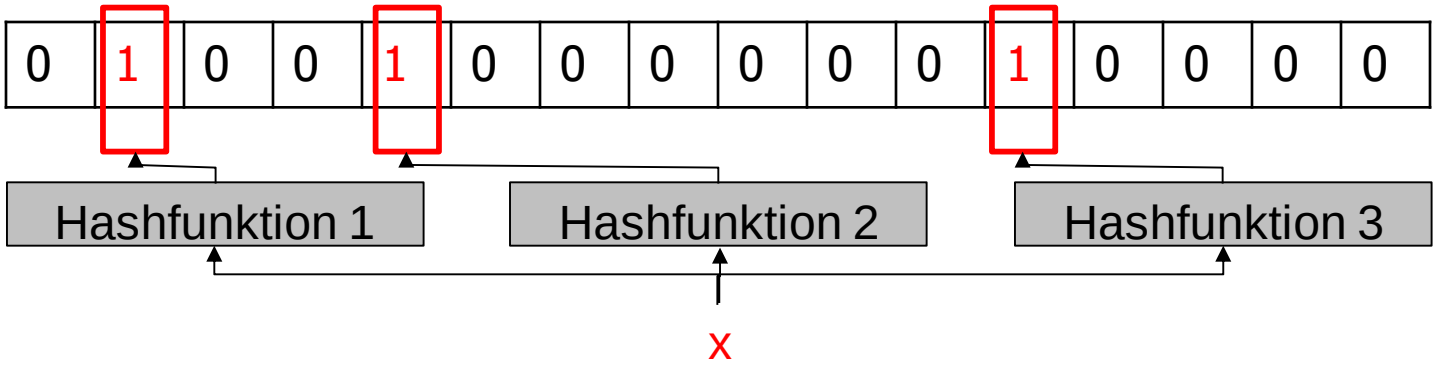
- Initialize the bit array to 0



Example: storing rows into bins

Example: for 64-128ms bin, 3 hash functions over 16bit

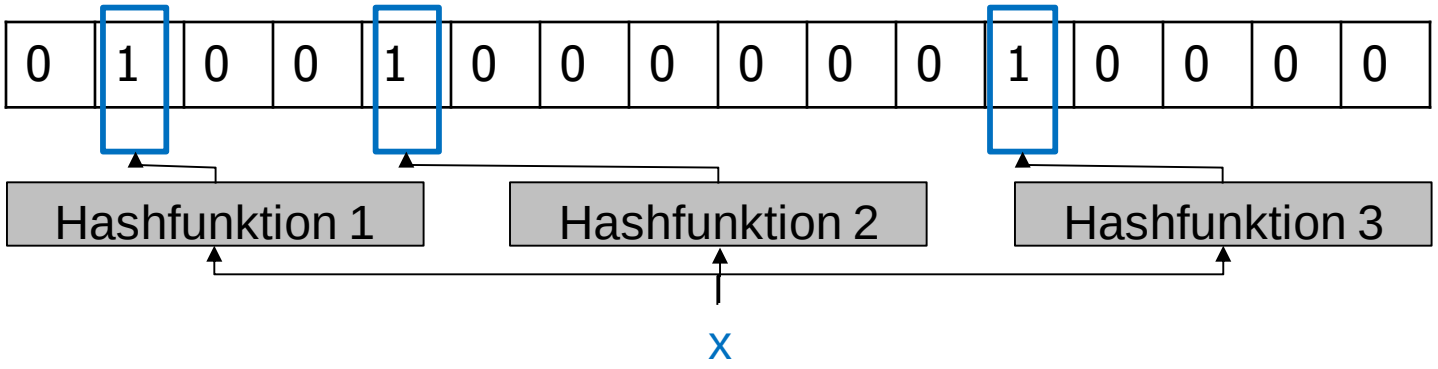
- Initialize the bit-array to 0
- Insert address x into the bin



RAIDR: storing rows into bins

Example: for 64-128ms bin, 3 hash functions over 16bit

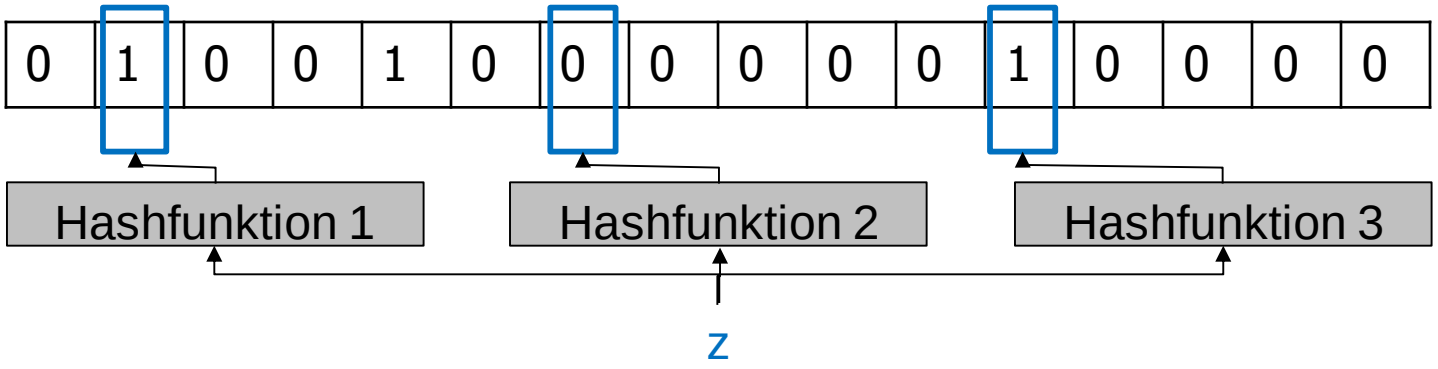
- Initialize the bit array to 0
- Insert address x into the bin
- Test address x: 1 & 1 & 1 = 1 (present)



Example: storing rows into bins

Example: for 64-128ms bin, 3 hash functions over 16bit

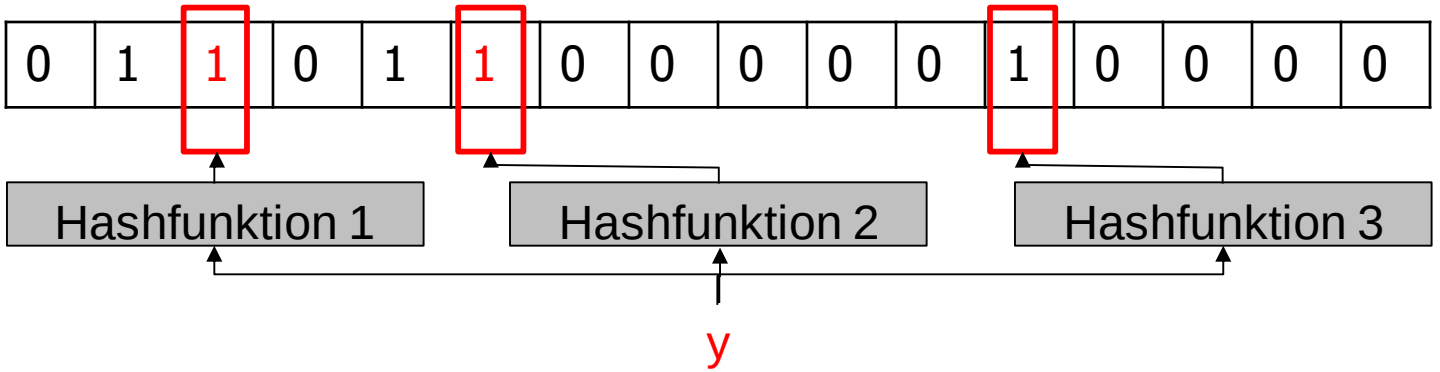
- Initialize the bit array to 0
- Insert address x into the bin
- Test address x: 1 & 1 & 1 = 1 (present)
- Test address z: 1 & 0 & 1 = 0 (not present)



Example: storing rows into bins

Example: for 64-128ms bin, 3 hash functions over 16bit

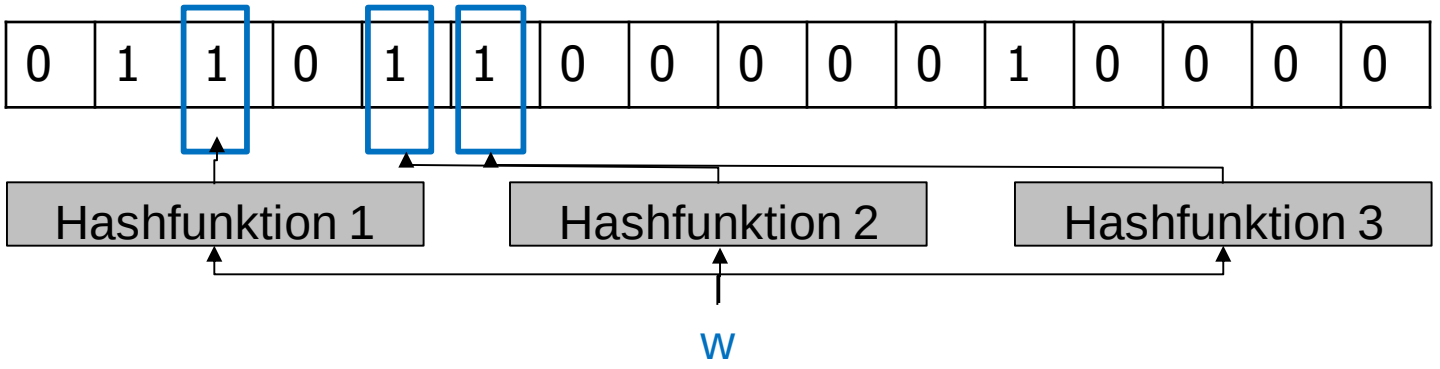
- Initialize the bit array to 0
- Insert address x into the bin
- Test address x: 1 & 1 & 1 = 1 (present)
- Test address z: 1 & 0 & 1 = 0 (not present)
- Insert address y into the bin



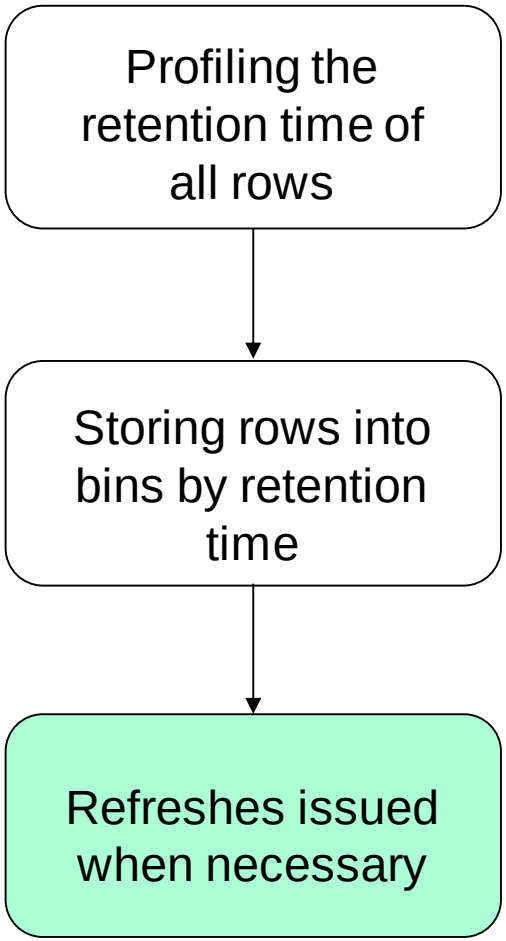
Example: storing rows into bins

Example: for 64-128ms bin, 3 hash functions over 16bit

- Initialize the bit array to 0
- Insert address x into the bin
- Test address x: 1 & 1 & 1 = 1 (present)
- Test address z: 1 & 0 & 1 = 0 (not present)
- Insert address y into the bin
- Test address w: 1 & 1 & 1 = 1 (False Positive, No false negative!)



Overview: RAIDR

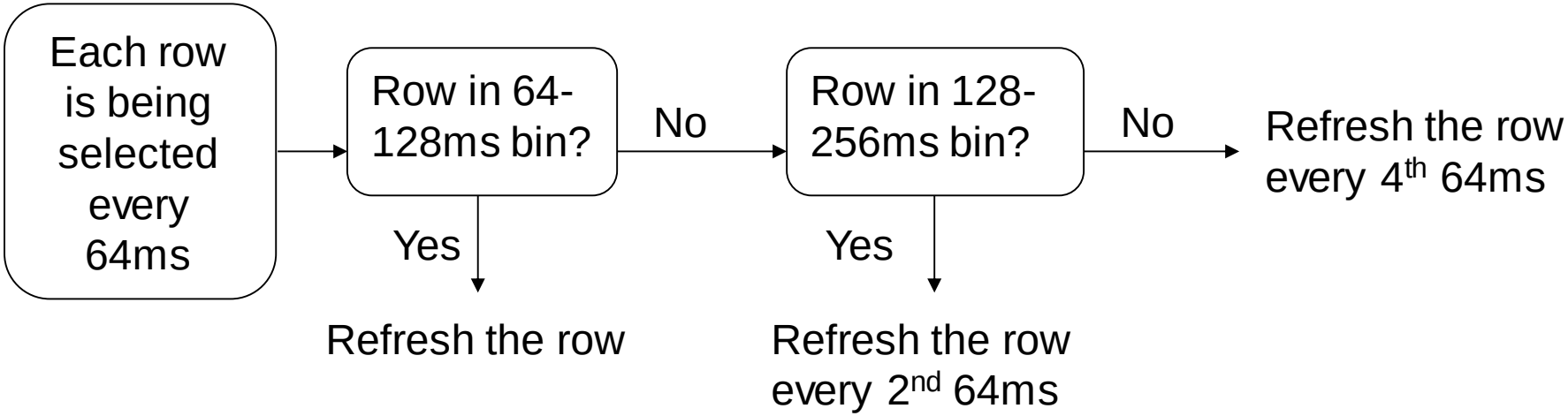


Retention time of each row is measured by deciding which refresh rate is needed.

The memory controller stores through a Bloom filter each row into bins by retention time.

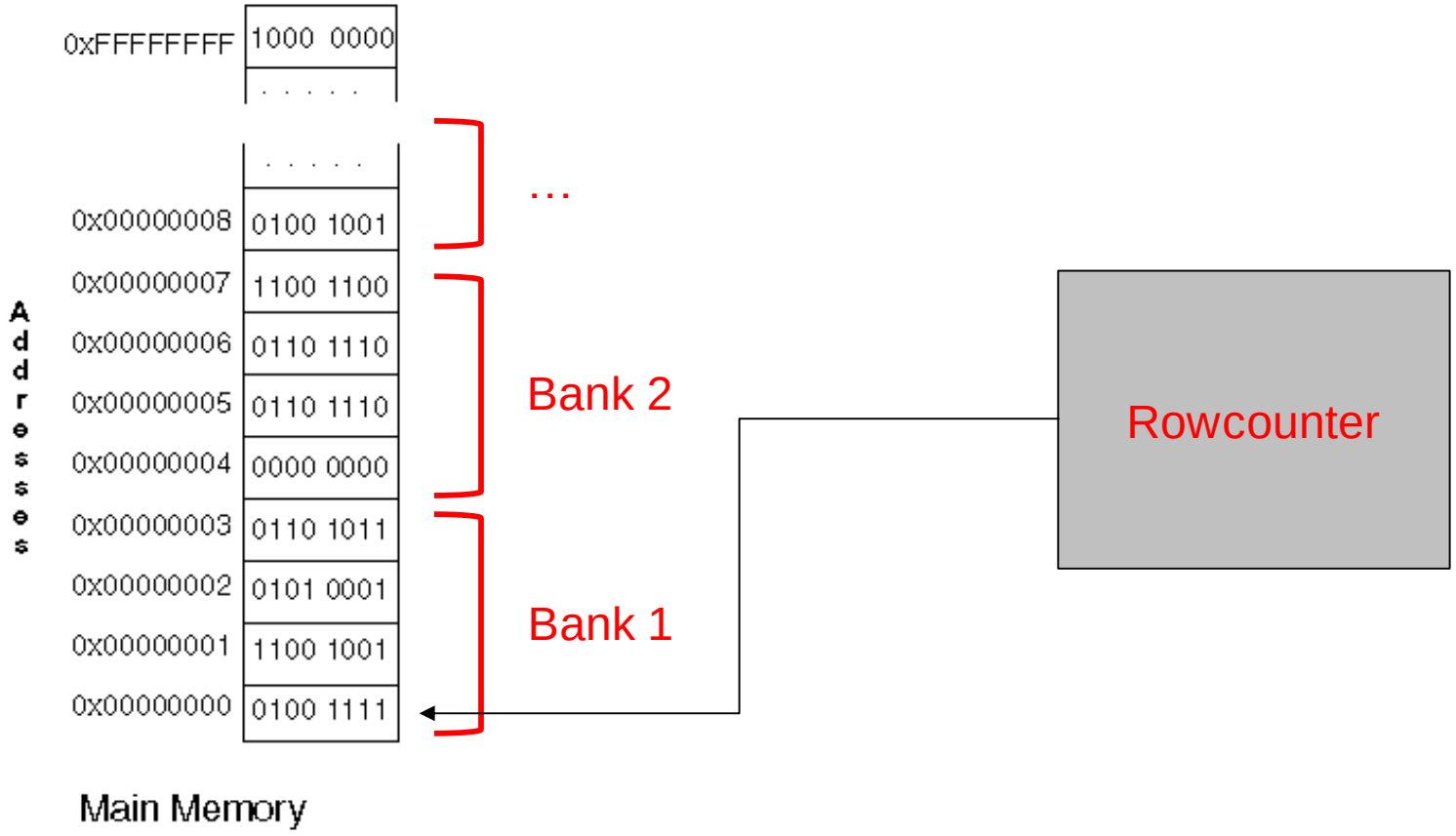
Every 64ms the memory controller decides if the row of cell needs to be refreshed according to its bin.

Overview: Refreshing rows



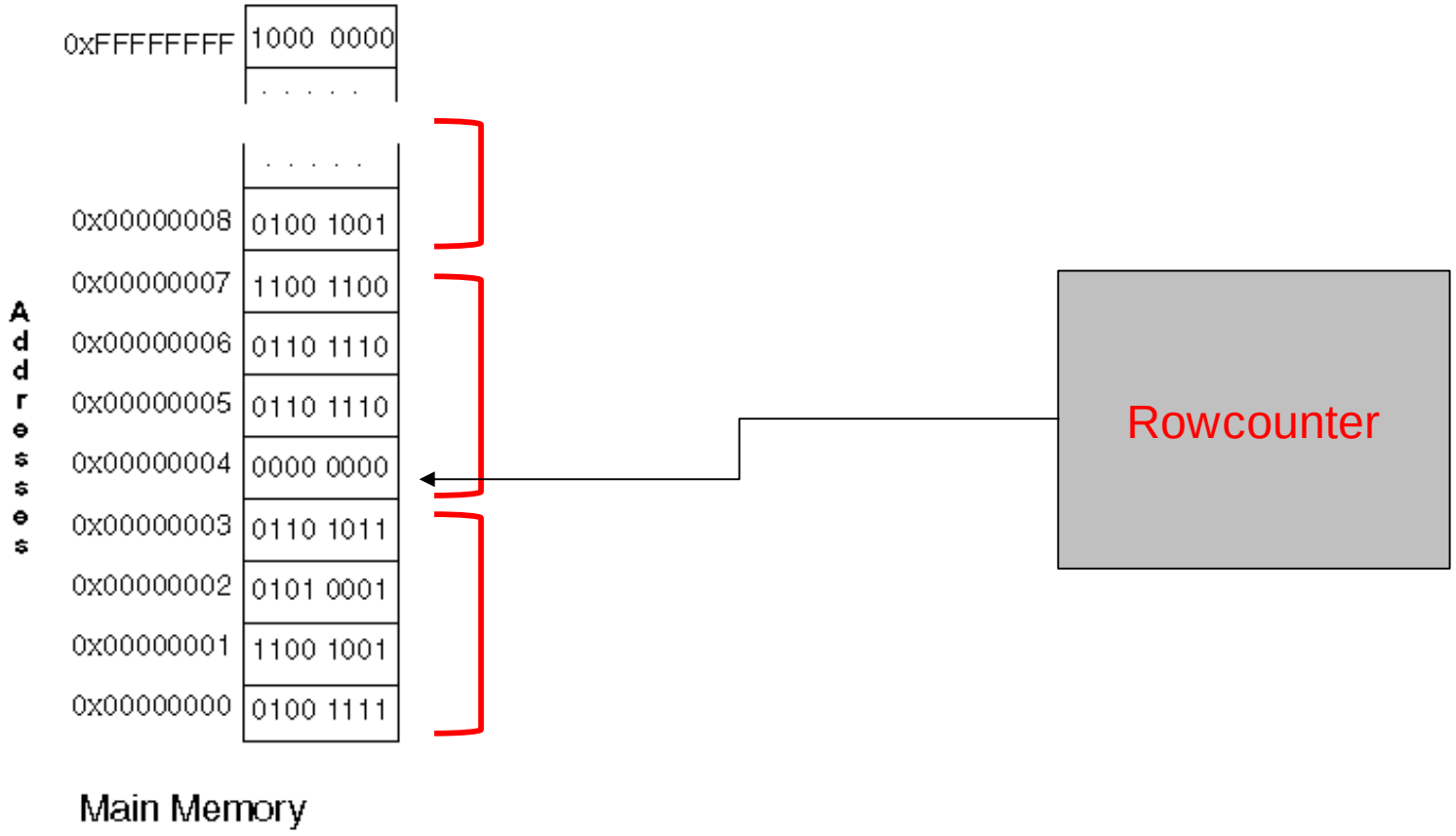
Selecting rows with Rowcounter

Rowcounter: counts through every row sequentially



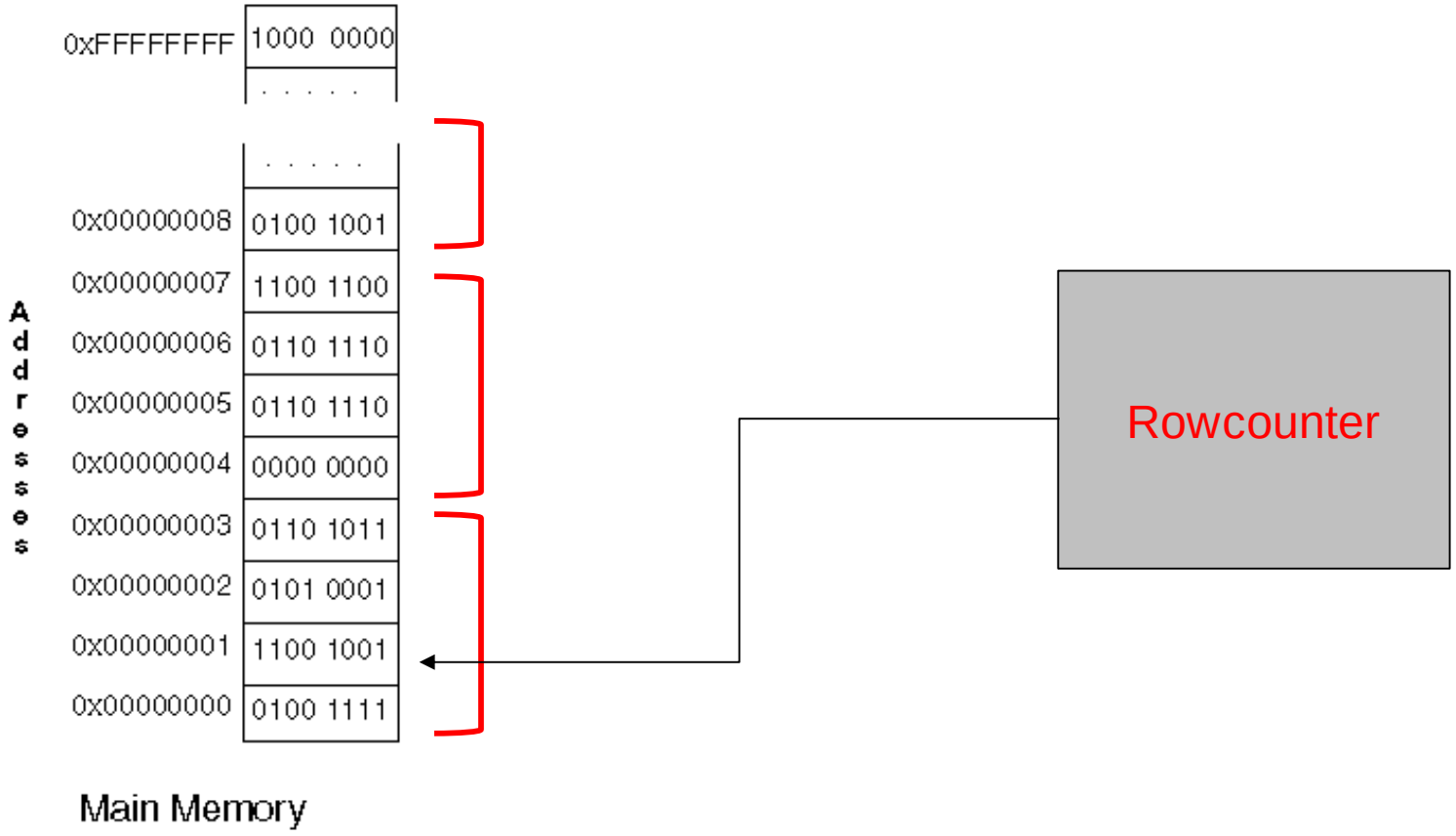
Selecting rows with Rowcounter

Rowcounter: counts through every row sequentially



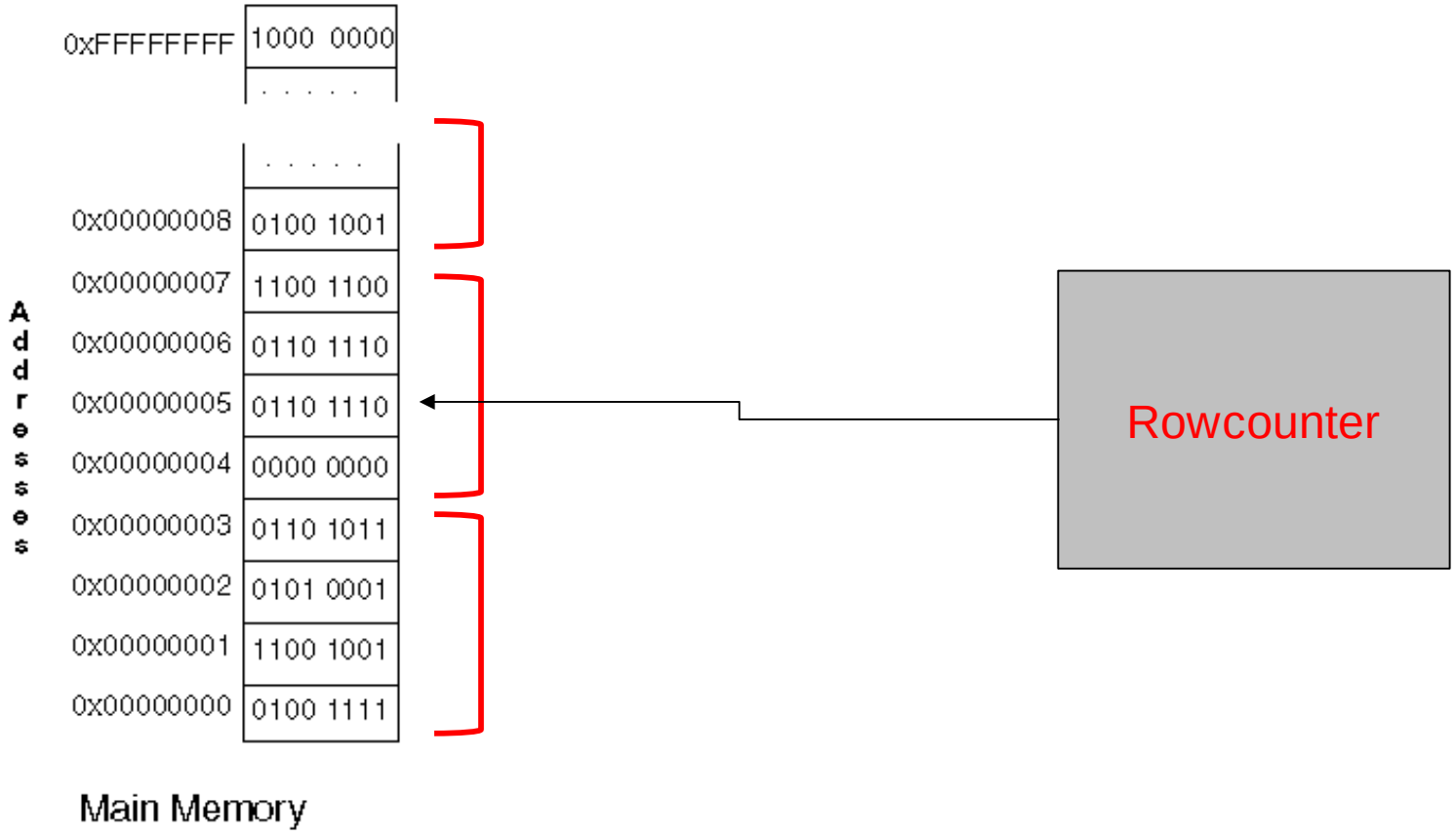
Selecting rows with Rowcounter

Rowcounter: counts through every row sequentially



Selecting rows with Rowcounter

Rowcounter: counts through every row sequentially

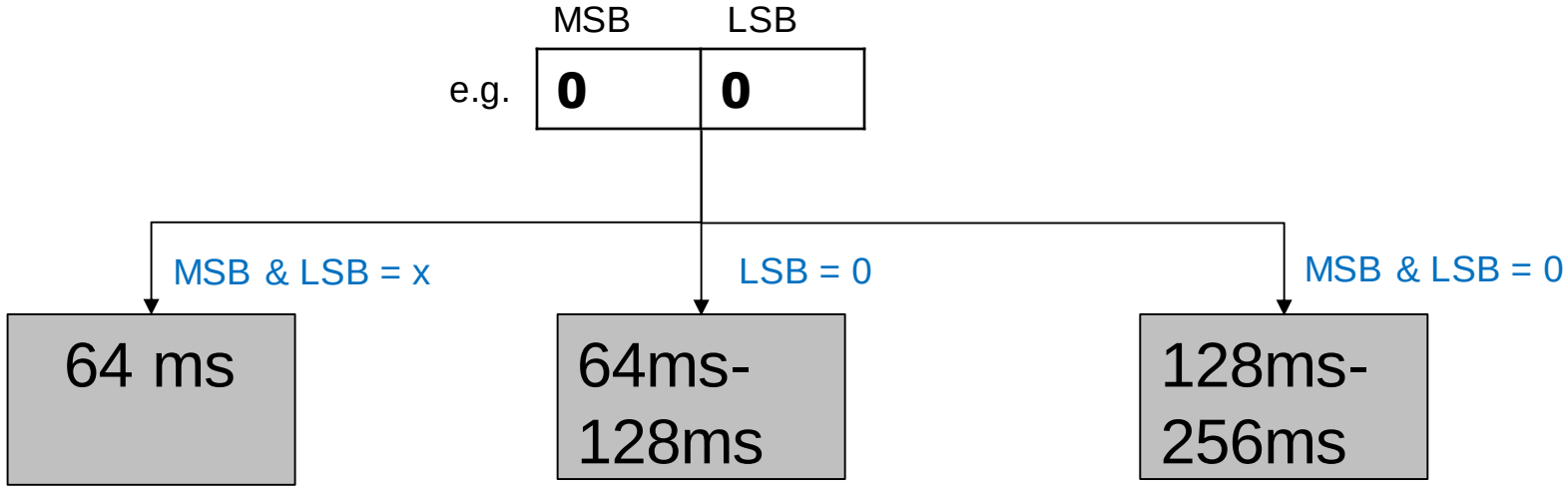


Determining Time since last refresh

Period counter: increments when the row counter resets, it counts to the shortest retention time not covered in any bins divided by 64ms and then rolls over.

Example: 3 bins of 64ms, 64-128ms and 128-256ms

- Shortest retention time not covered: 256ms = 4 x 64ms
- Period counter is 2bits and counts from 0 to 3



Issues with high temperature

- High temperature causes DRAM retention time to decrease.
- RAIDR implements a refresh rate scaling mechanism.
- Refresh rate scaler: a counter which helps to increases the refresh rate the higher the temperature is

Outline

- Executive Summary
- Background
- Key Approach, Ideas & Mechanisms
- Key Results: Methodology and Evaluation
- Summary
- Novelty

- Strength
- Weaknesses
- Thoughts and Ideas
- Takeaways
- Open Discussion

Evaluated system configuration

Table 1: Evaluated system configuration

Component	Specifications
Processor	8-core, 4 GHz, 3-wide issue, 128-entry instruction window, 16 MSHRs per core
Per-core cache	512 KB, 16-way, 64 B cache line size
Memory controller	FR-FCFS scheduling [41, 54], line-interleaved mapping, open-page policy
DRAM organization	32 GB, 2 channels, 4 ranks/channel, 8 banks/rank, 64K rows/bank, 8 KB rows
DRAM device	64x Micron MT41J512M8RA-15E (DDR3-1333) [33]

Table 2: Bloom filter properties

Retention range	Bloom filter size m	Number of hash functions k	Rows in bin	False positive probability
64 ms – 128 ms	256 B	10	28	$1.16 \cdot 10^{-9}$
128 ms – 256 ms	1 KB	6	978	0.0179

Evaluated Methodology

- Each benchmark gets classified as memory-intensive or non-memory-intensive based on its last level cache misses per 1000 instructions (MPKI).

  MPKI > 5 => memory-intensive

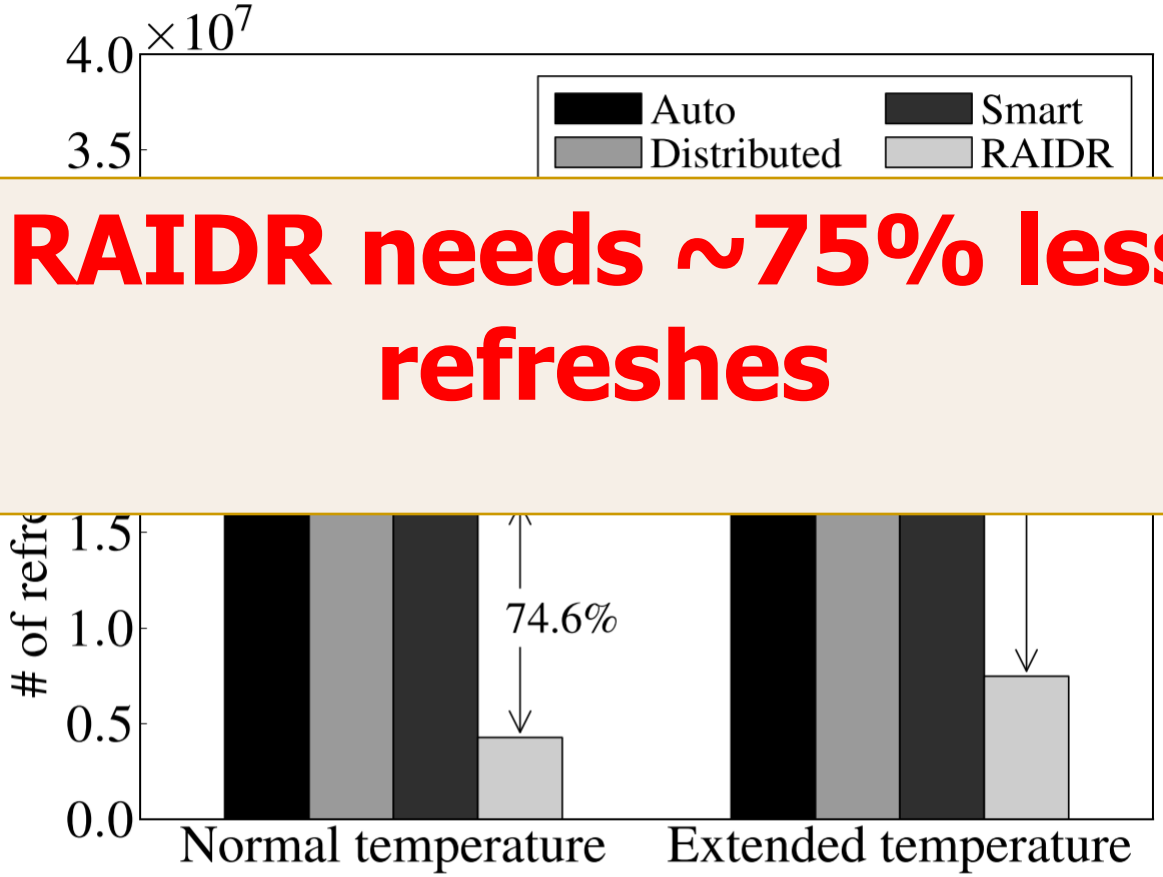
  MPKI < 5 => non-memory-intensive

- DRAM system power = energy per memory access serviced

RAIDR vs. other mechanisms

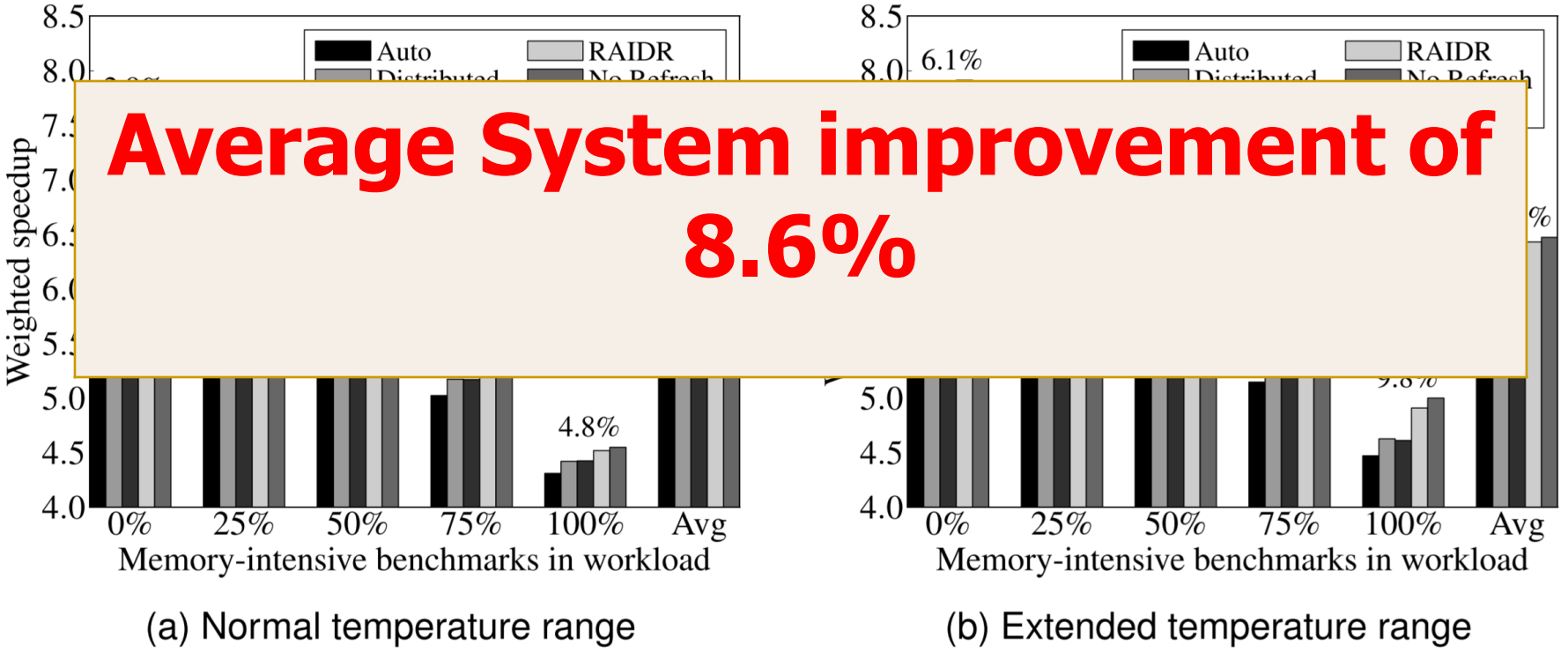
- **Auto-refresh:**
  - Memory controller send out auto-refresh command to refresh several rows per command
  - This is used in existing systems today.
- **Distributed refresh:**
  - Memory controller sends out address row by row that are going to be refreshed.
  - Makes use of bank-level parallelism.
- **Smart Refresh:**
  - Timeout counter for each row which is reset when the row is accessed or refreshed. →refresh when counter expires
- **No refresh:**
  - Ideal scheme, doesn't exist today.

Refresh Reduction



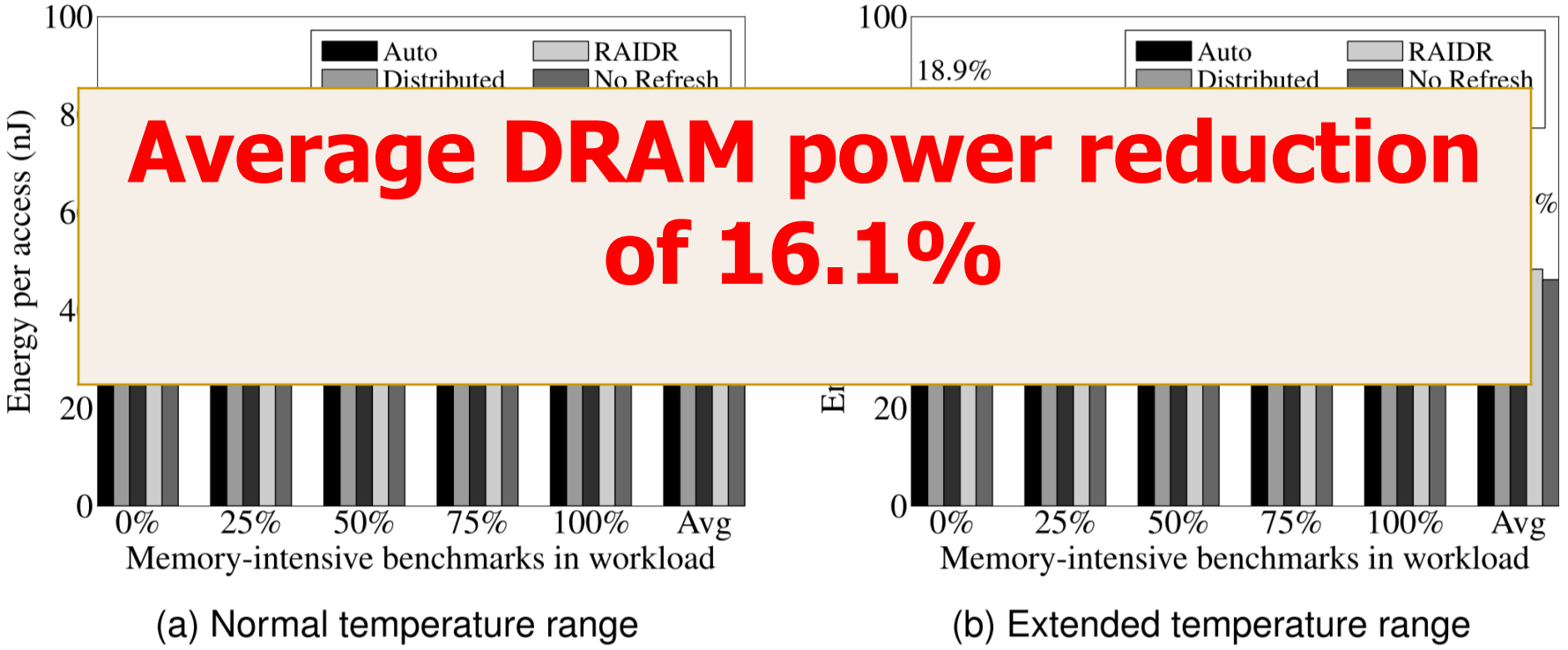
RAIDR needs ~75% less refreshes

Performance Analysis



Average System improvement of 8.6%

Energy Analysis



Average DRAM power reduction of 16.1%

Related works

- Paper: A Case for Exploiting Subarray-Level Parallelism (SALP) in DRAM
- Paper: Improving DRAM Performance by Parallelizing Refreshes with Accesses
- Paper: An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms
- Paper: Smart Refresh: An Enhanced Memory Controller Design for Reducing Energy in Conventional and 3D Die-Stacked DRAMs

Outline

- Executive Summary
- Background & Motivation
- Key Approach, Ideas & Mechanisms
- Key Results: Methodology and Evaluation
- Summary
- Novelty

- Strength
- Weaknesses
- Thoughts and Ideas
- Takeaways
- Open Discussion

Summary

- Motivation: DRAM refresh operations waste energy and performance overhead is high.
- Problem: Not all DRAM cells need same refresh rate.
- Goals: to minimize the number of refresh operations performed without increasing hardware or software much. RAIDR categorizes cells and only refreshes those more often which also require it.
- Evaluation: RAIDR achieves in a 32 GB DRAM: 74.6% refresh reduction, an average DRAM power reduction of 16.1% and an average system performance improvement of 8.6% over existing system.

  And the memory controller only needs 1.25KB additional.

Outline

- Executive Summary
- Background & Motivation
- Key Approach, Ideas & Mechanisms
- Key Results: Methodology and Evaluation
- Summary
- Novelty

- Strength
- Weaknesses
- Thoughts and Ideas
- Takeaways
- Open Discussion

Novelty

- First to group each row of DRAM cells into bins by their retention time and use the difference in retention time like this.
- A low-cost mechanism which require no DRAM modifications and small changes to the memory controller.

Outline

- Executive Summary
- Background & Motivation
- Key Approach, Ideas & Mechanisms
- Key Results: Methodology and Evaluation
- Summary
- Novelty

- Strength
- Weaknesses
- Thoughts and Ideas
- Takeaways
- Open Discussion

Strength

- Smaller refresh rate of the cells in the DRAM are achieved at a small cost, leading to performance enhancement
- Small modification to the memory controller
- No changes needed to the DRAM
- Simple and intuitive key approach
- The paper was easy to read and understand

Outline

- Executive Summary
- Background & Motivation
- Key Approach, Ideas & Mechanisms
- Key Results: Methodology and Evaluation
- Summary
- Novelty

- Strength
- Weaknesses
- Thoughts and Ideas
- Takeaways
- Open Discussion

Weaknesses

- The profiling step of RAIDR: when is this issued and how often is it repeated?

  →retention time of DRAM cells do change over time

- If a row has one cell with low retention time, the whole row has to be refreshed at this rate.

Outline

- Executive Summary
- Background & Motivation
- Key Approach, Ideas & Mechanisms
- Key Results: Methodology and Evaluation
- Summary
- Novelty

- Strength
- Weaknesses
- Thoughts and Ideas
- Takeaways
- Open Discussion

Thoughts and Ideas

- Dynamic profiling needed → AVATAR can be used maybe?

  →Paper: AVATAR: A Variable-Retention-Time (VRT) Aware Refresh for DRAM Systems

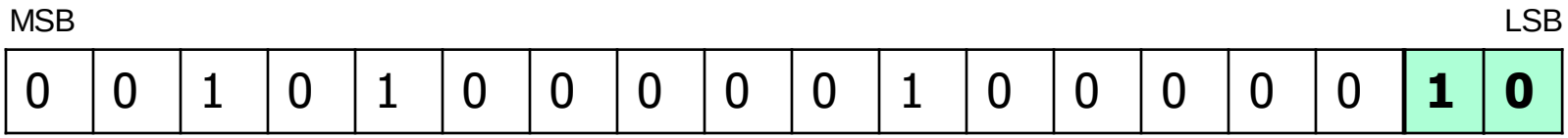
Thoughts and Ideas

- Refresh half of the row if the cells that have the lower retention time is on one side of the row.

  Profiling & binning phase: determine where the "weak cell" is, if it's on the right side of the row, mark the LSB with 1 bit in the bit-array with a 1, and vice versa on the 2nd LSB.

  Refreshing phase: check the address as usual but only refresh half of the row according to the bits set if only one of the two bits are set.

Benefit: don't need to activate half of the bitlines → energy saved.



Outline

- Executive Summary
- Background & Motivation
- Key Approach, Ideas & Mechanisms
- Key Results: Methodology and Evaluation
- Summary
- Novelty

- Strength
- Weaknesses
- Thoughts and Ideas
- Takeaways
- Open Discussion

Main Takeaways

- A good way to make use of different retention time of DRAM cells and increase performance
- Small modifications needed to implement the idea
- Easy to read & understand paper

Outline

- Executive Summary
- Background & Motivation
- Key Approach, Ideas & Mechanisms
- Key Results: Methodology and Evaluation
- Summary
- Novelty

- Strength
- Weaknesses
- Thoughts and Ideas
- Takeaways
- Open Discussion

Open Discussion

- Have you heard about other solutions regarding inefficient DRAM refresh operations?
- What do you think about RAIDR?
- Can you think of other places where we can implement this RAIDR system or parts from it?
- Do you have any other ideas on how to make the profiling step of RAIDR better?
- What are your thoughts on the paper? Negative/Positive

Thank you for your attention!

Additional sides