LADDER: Architecting Content and Location-aware Writes for Crossbar Resistive Memories

Md Hafizul Islam Chowdhuryy[1], Muhammad Rashedul Haq Rashed[1], Amro Awad[2], Ricard Ewetz[1] and Fan Yao[1]

[1]Department of ECE
University of Central Florida
Florida, USA

[2]Department of ECE
North Carolina State University
North Carolina, USA

IEEE/ACM International Symposium on Microarchitecture (**MICRO'21**)
October 18–22, 2021

ReRAM: Emerging Memory Technology

- Increasing need for efficient and scalable memory systems.
- DRAM technologies do not match the capacity demand.
- **Resistive memory (ReRAM) is promising for main-memory integration.**
  - High cell-density and scalability
  - Low static power

ReRAM suffers from varying latency requirements for write, which could limit system performance if not utilized properly.

❖ Increasing demand for efficient and scalable memory system.

❖ Traditional memory (i.e., DRAM) cannot match the capacity demand.

❖ R

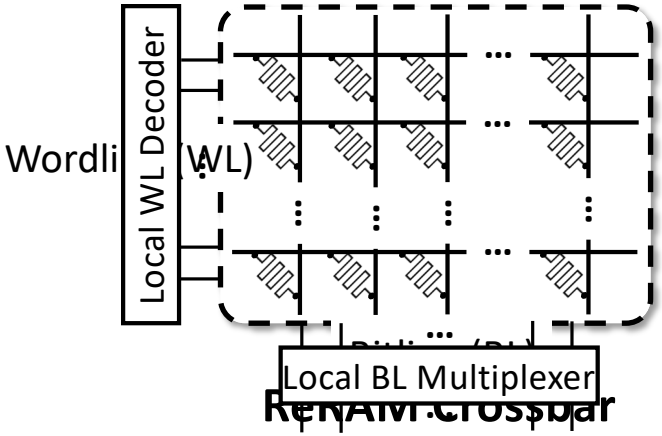
This work

LADDER **- Processor-side low-overhead framework to improve ReRAM performance by exploiting variable write latency.**

ReRAM suffer from *location/content-dependent* variable write latency, limiting system performance.

ReRAM Memory Organization

- ReRAM cells are arranged in dense array structures (called **Crossbar**).
  - Increase area efficiency
- ReRAM cell arrays form a **MAT** (i.e., crossbar size of 512x512 cells).
  - Contains peripheral circuitry to support read/write
  - The basic unit to form *banks, ranks and channels*

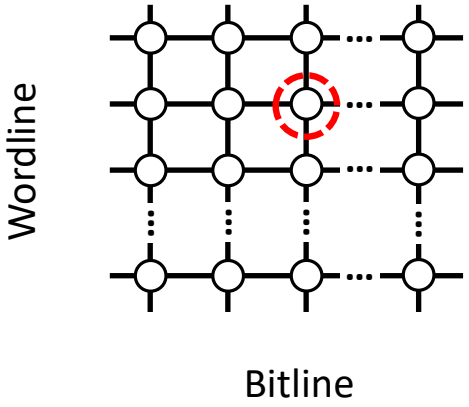


ReRAM MAT

UCF

ReRAM Write Operations

- ReRAM write operation involves two phases.
  - **RESET:** transition from ***low-resistive state*** to ***high-resistive state*** (i.e., '1' → '0')
  - **SET:** transition from ***high-resistive state*** to ***low-resistive state*** (i.e., '0' → '1')

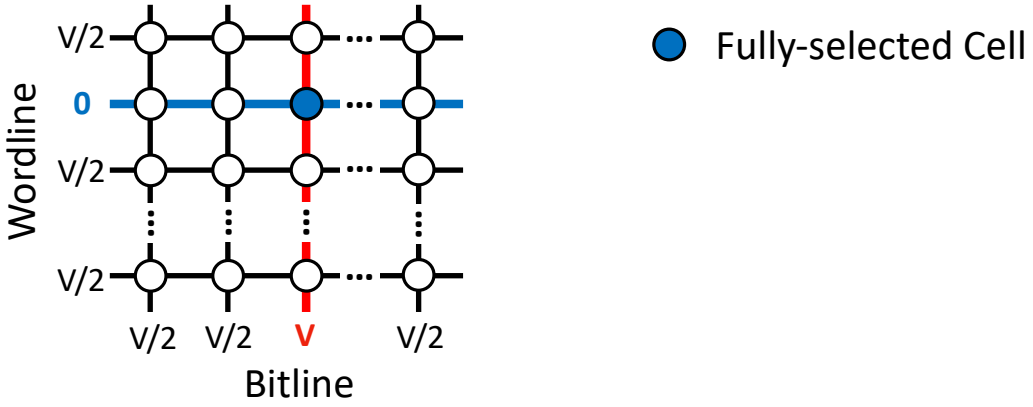


ReRAM Crossbar

UCF

ReRAM Write Operations

- ReRAM write operation involves two phases.
  - **RESET:** transition from ***low-resistive state*** to ***high-resistive state*** (i.e., '1' → '0')
  - **SET:** transition from ***high-resistive state*** to ***low-resistive state*** (i.e., '0' → '1')

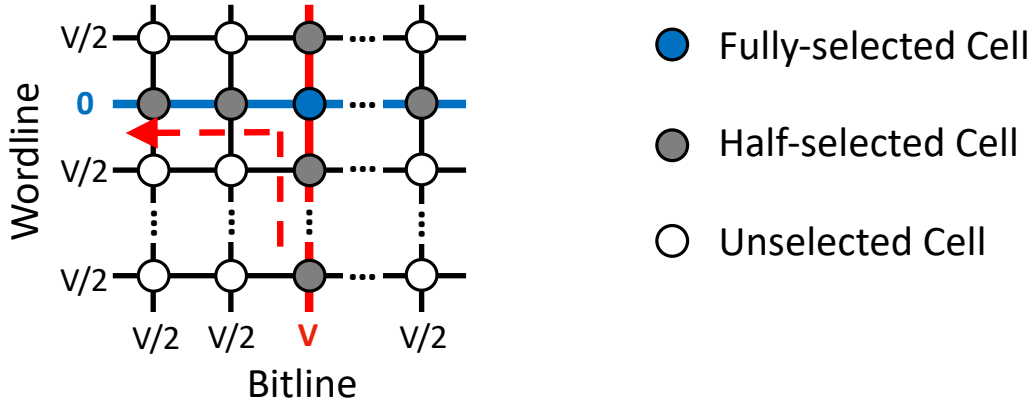


ReRAM Crossbar

UCF

ReRAM Write Operations

- ReRAM write operation involves two phases.
  - **RESET:** transition from ***low-resistive state*** to ***high-resistive state*** (i.e., '1' → '0')
  - **SET:** transition from ***high-resistive state*** to ***low-resistive state*** (i.e., '0' → '1')



ReRAM Crossbar

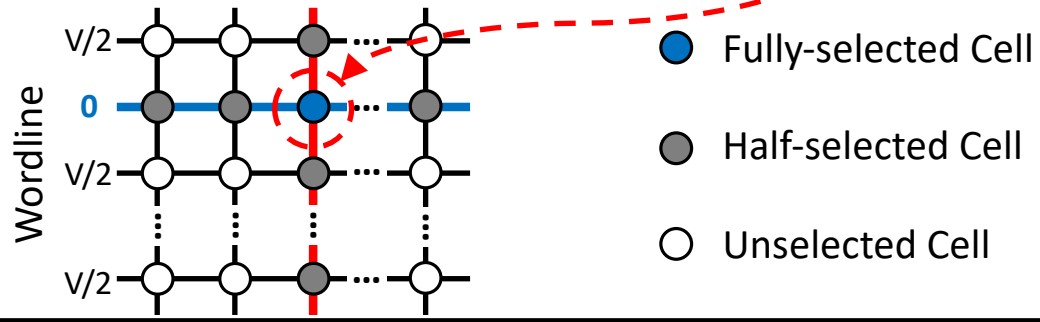
UCF

ReRAM Write Operations

- ReRAM write operation involves two phases.
  - **RESET:** transition from ***low-resistive state*** to ***high-resistive state*** (i.e., '1' → '0')
  - **SET:** transition from ***high-resistive state*** to ***low-resistive state*** (i.e., '0' → '1')

- RESET latency, $t = C * e^{-k|V_d|}$

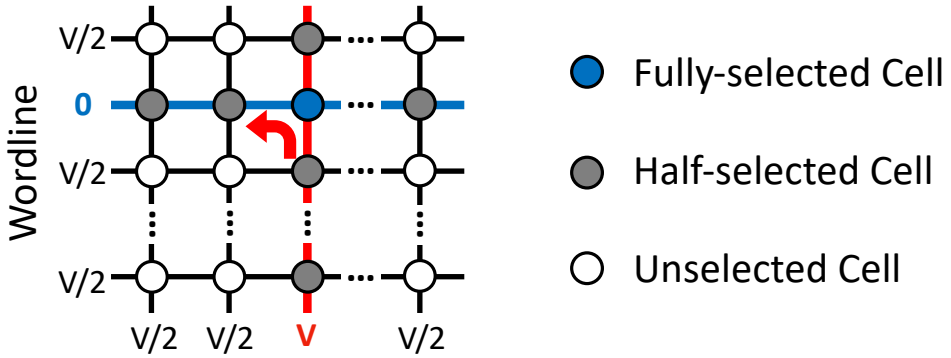
V_d = Voltage drop across target cell



RESET latency of a ReRAM cell is *exponentially proportional* to the voltage drop across the target cell.

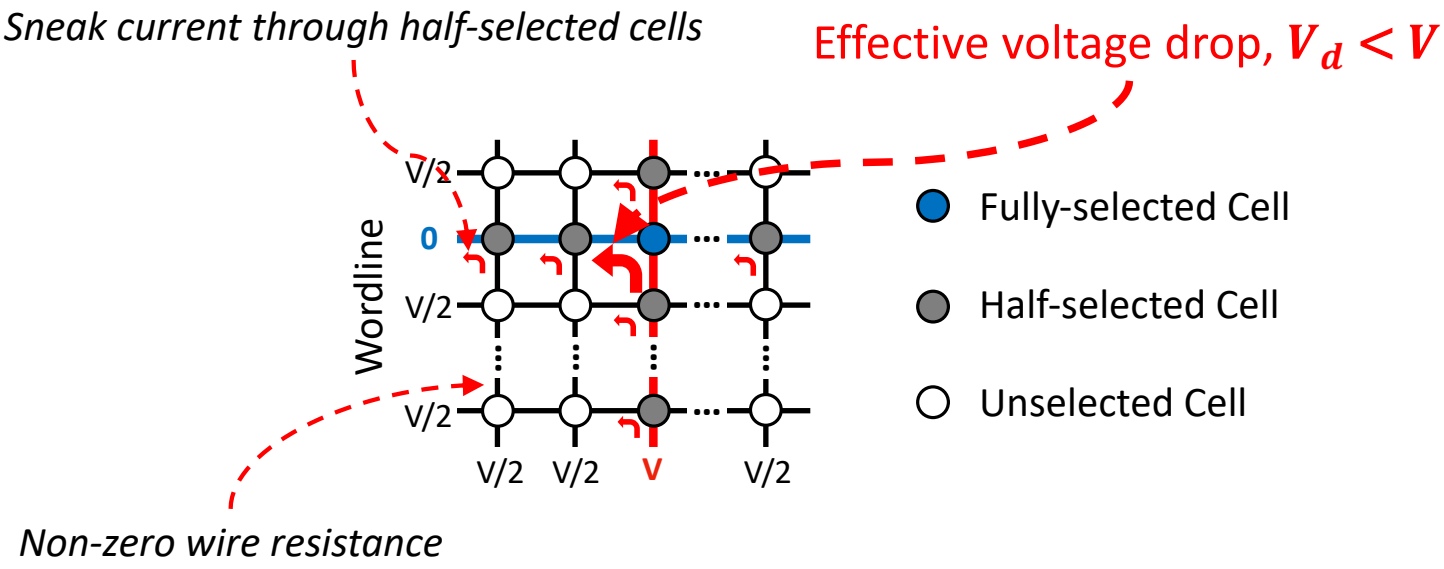
UCF

Variable RESET Latency



UCF

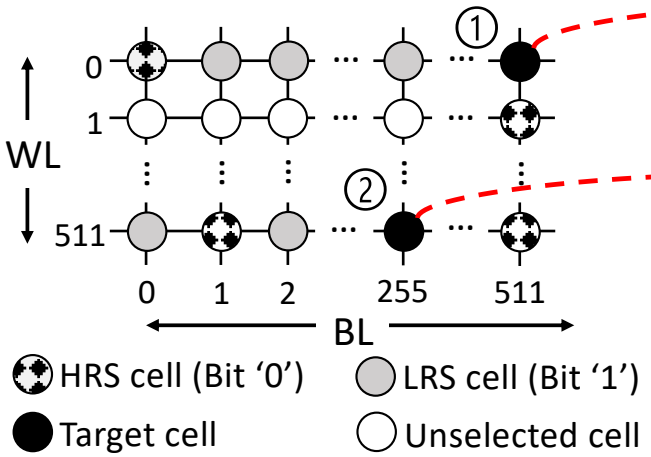
Variable RESET Latency



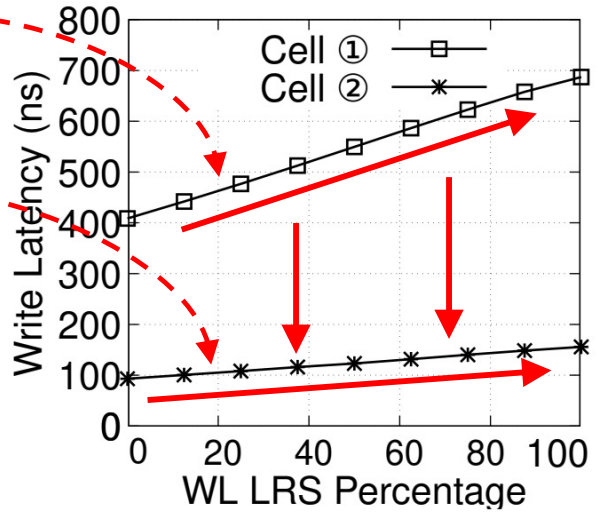
RESET latency, $t = C * e^{-k|V_d|}$

UCF

Location/Content Dependence of RESET



Crossbar showing two RESETs



Location/Content dependency of RESET latency

- RESET latency varies significantly based on **data-pattern and location**.
- Using worst case reset latency can significantly degrade system performance.

UCF

Prior Works and Motivation

❖ **Architecture and circuit-level techniques:**

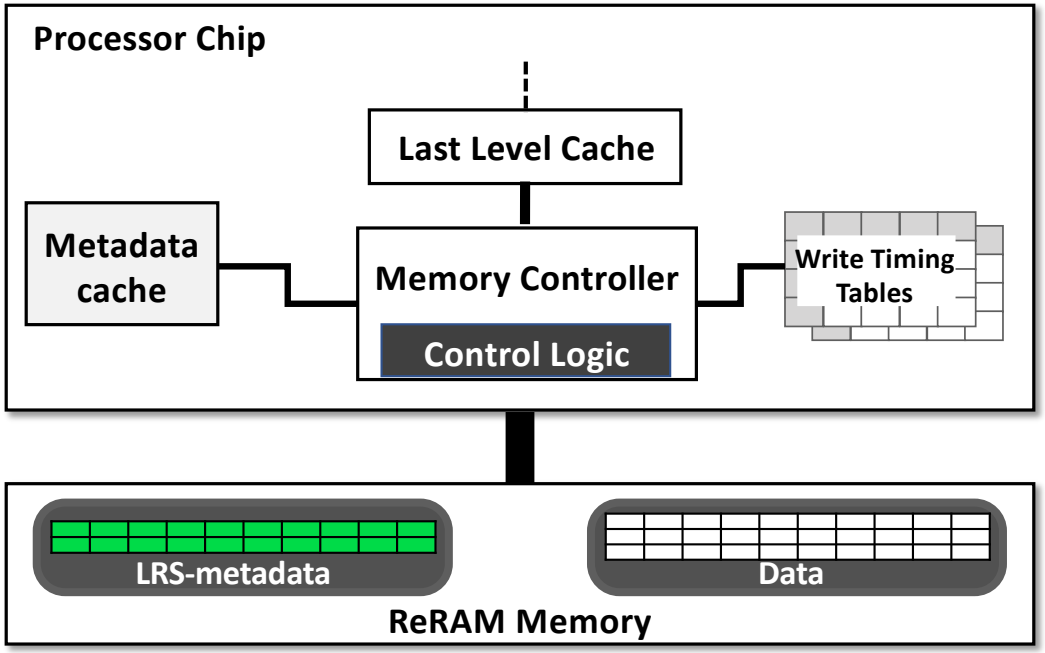
- Dynamic reset voltage to compensate the IR-drop (e.g., **Zokaee et al. HPCA'20**)
- Track *bitline data patterns* with custom profiling circuitry (e.g., **Wen et al. TCAD'19**)
- Model *location dependent* RESET latency (e.g., **Zhang et al. DATE'16**)
- Limit sneak current at crossbar level (e.g., **Xu et al. HPCA'15**)

❖ Issues:

- Additional circuitry increases design complexity of memory subsystems
- Do not harness the full potential gains

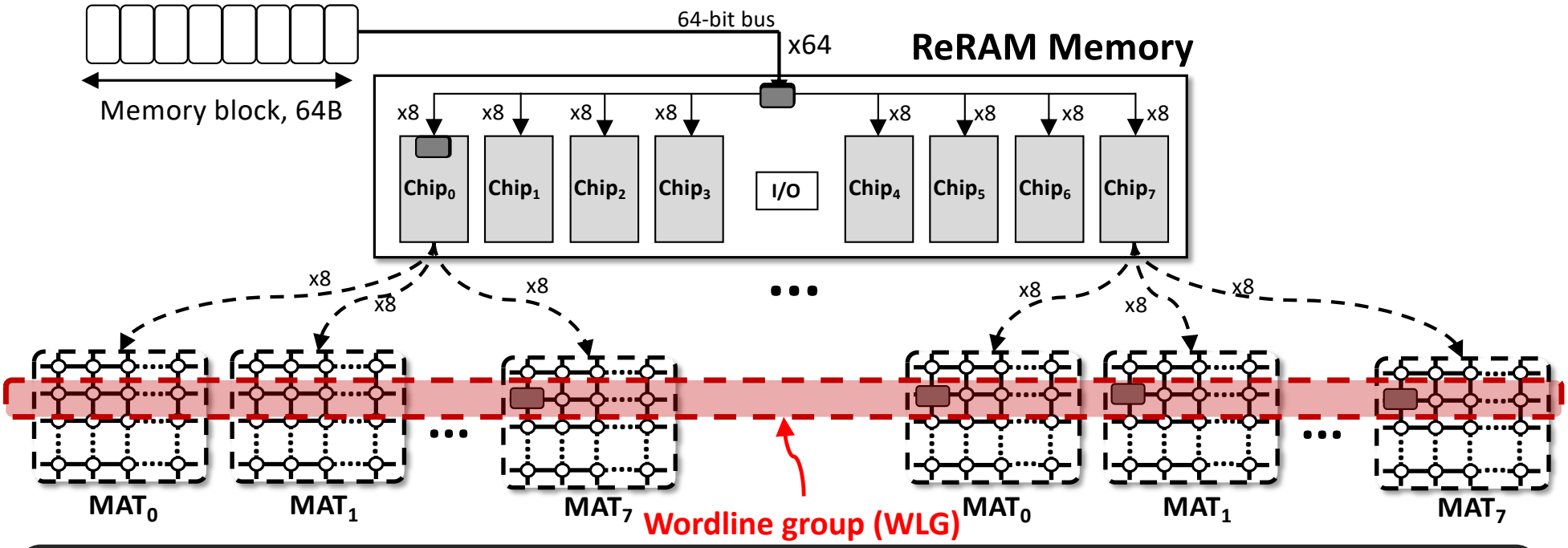
A processor-side scheme that utilizes variable RESET latency, without adversely impacting the complexity of commodity ReRAM devices.

Overview of LADDER



UCF

Location & Data-Content Aware Latency Model

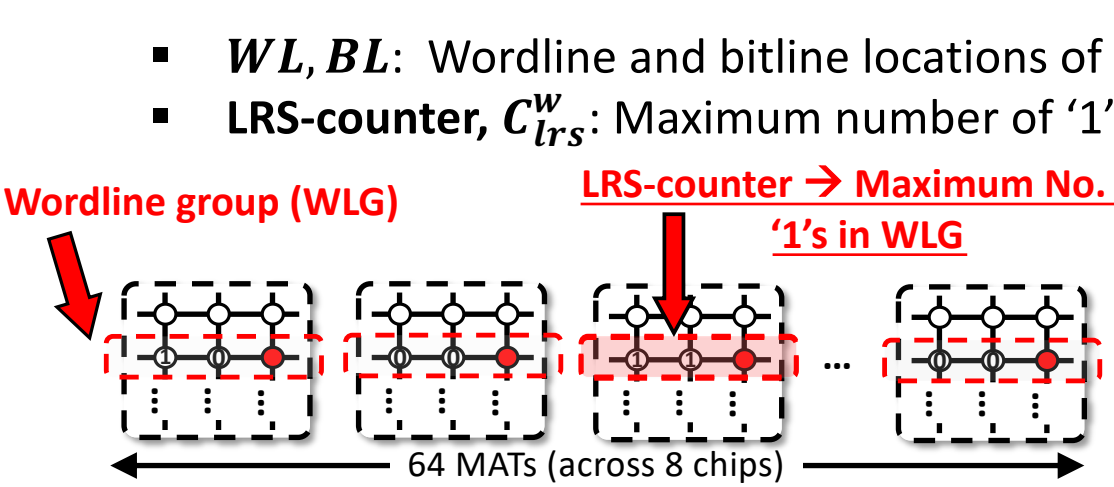


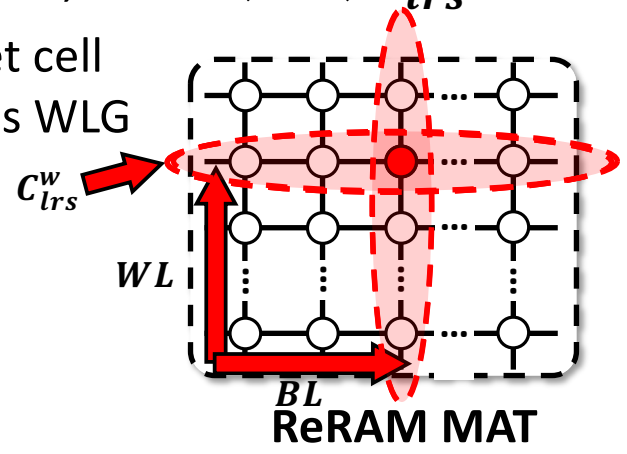
- Each MAT stores 1 byte (8-bit) per memory block.
- 8 MATs in each chip are used to store one memory block (64 WLs total).

UCF

Location & Data-Content Aware Latency Model

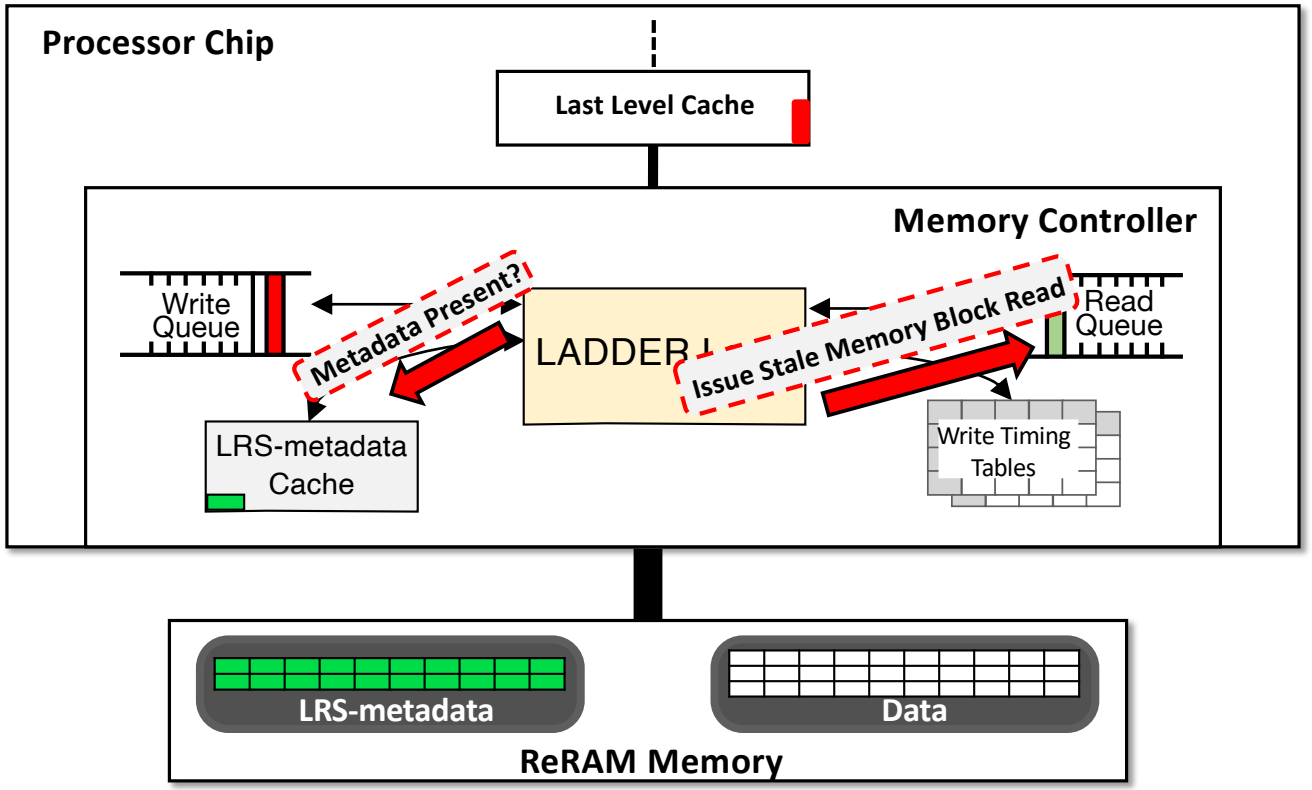
- Ideally, both WL and BL data pattern should be monitored.
  - i.e., keep the counter of '1's in each row and column
  - Profiling BL content (i.e., column) in memory controller is prohibitively expensive
- Tradeoff: Track number of '1's in ***WL (i.e., row) only*** (*LRS metadata*)
- WL and BL location and WL content of target cell, $< \boldsymbol{WL}, \boldsymbol{BL}, \boldsymbol{C_{lrs}^{w}} >$.
  - $\boldsymbol{WL}, \boldsymbol{BL}$: Wordline and bitline locations of a target cell
  - **LRS-counter,** $\boldsymbol{C_{lrs}^{w}}$: Maximum number of '1's across WLG





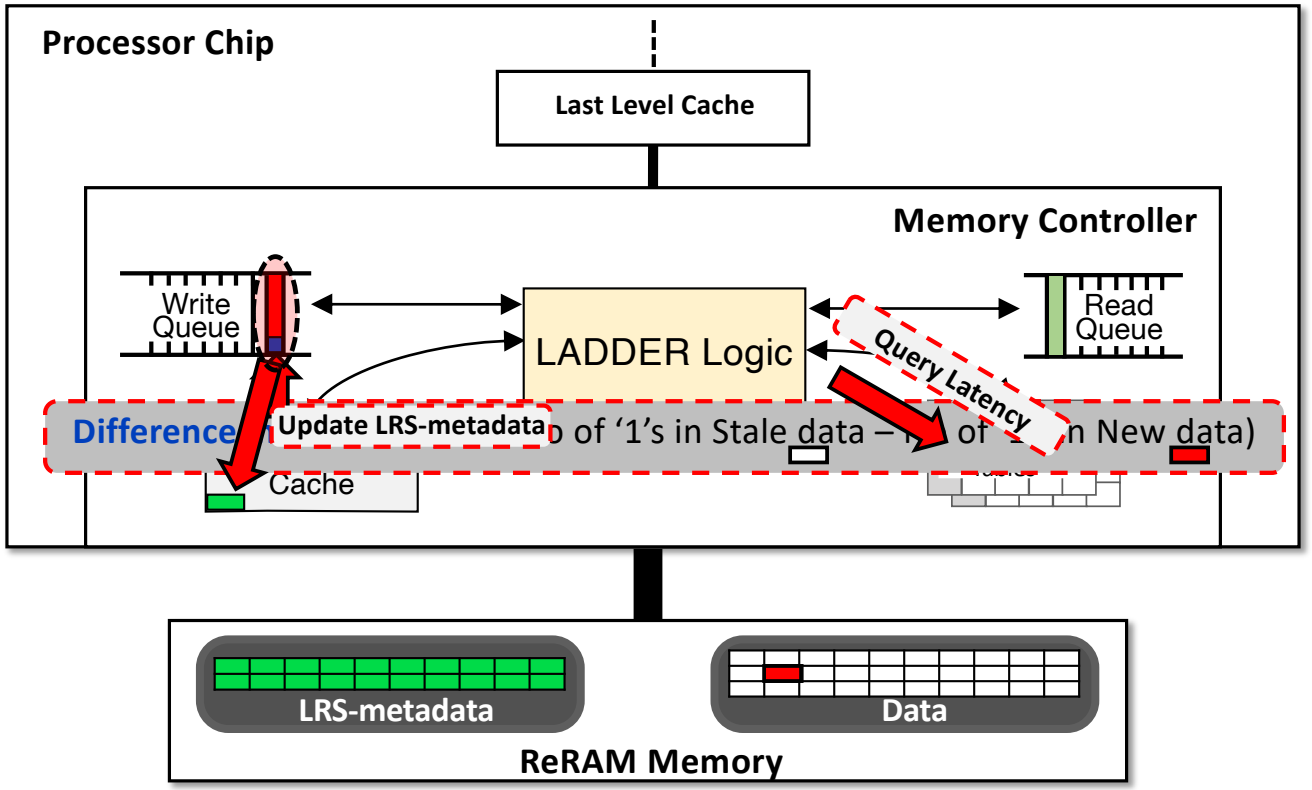
UCF

LADDER-Basic Data Write Operation



UCF

LADDER-Basic Data Write Operation



UCF

Optimization Opportunities with LADDER-Basic

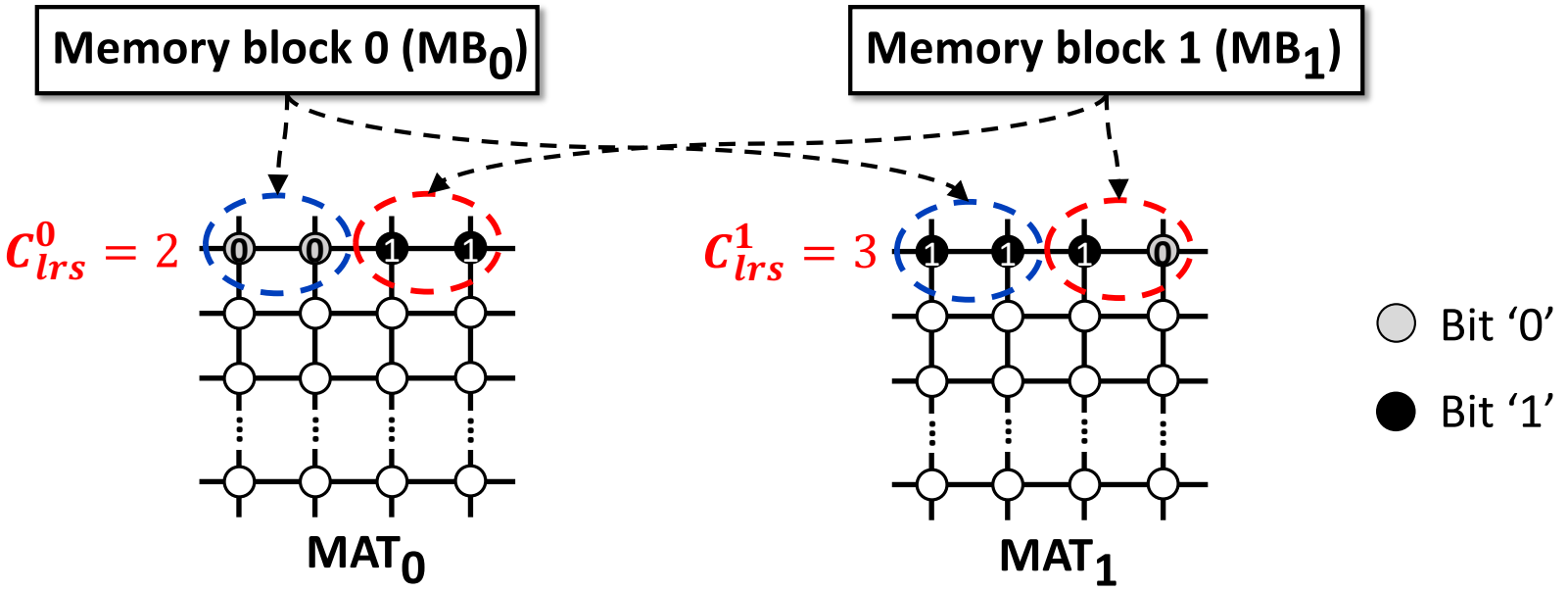
- Each data write requires a Stale Memory Block (SMB) read.
- Non-trivial metadata maintenance overhead.
  - Additional reads and writes for LRS-metadata
  - Metadata storage overhead

- How to get rid of Stale Memory Block (SMB) Reads?
- How to reduce LRS-metadata maintenance overhead?

LADDER-Est: Eliminating SMB Reads

- RESET latency depends **only** on maximum no. of ‘1’s in WLG.
- LRS-metadata estimation scheme:
  - Tracks only the worst-case byte in a newly-written data as metadata
  - Memory controller can update no. of ‘1’s in the worst-case byte *without old data*

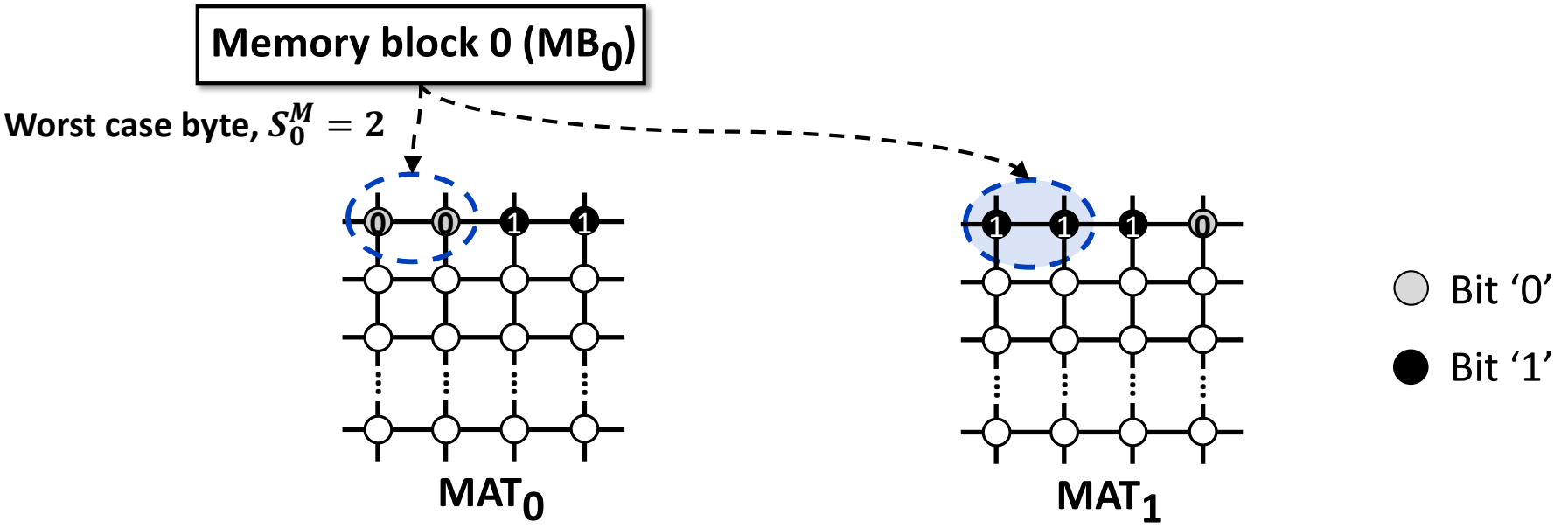
LRS-metadata Estimation Principle



Accurate LRS-counter (LADDER-Basic), $C^w_{lrs} = 3$

UCF

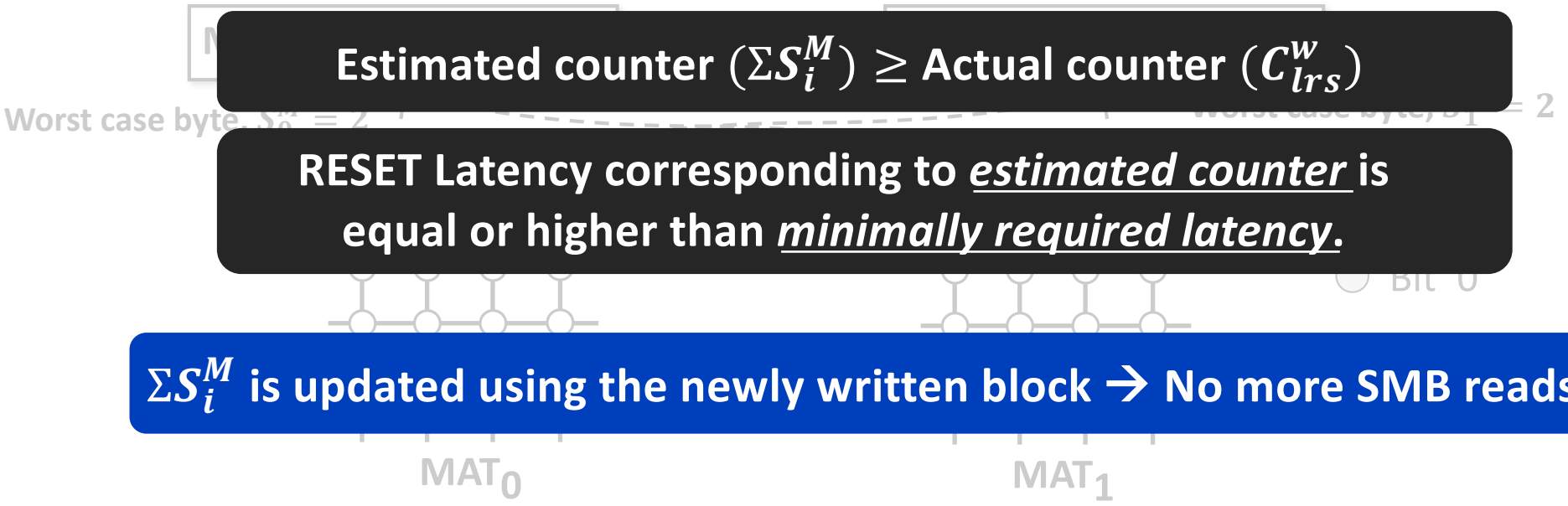
LRS-metadata Estimation Principle



Accurate LRS-counter (LADDER-Basic), $C_{lrs}^{w} = 3$

UCF

LRS-metadata Estimation Principle

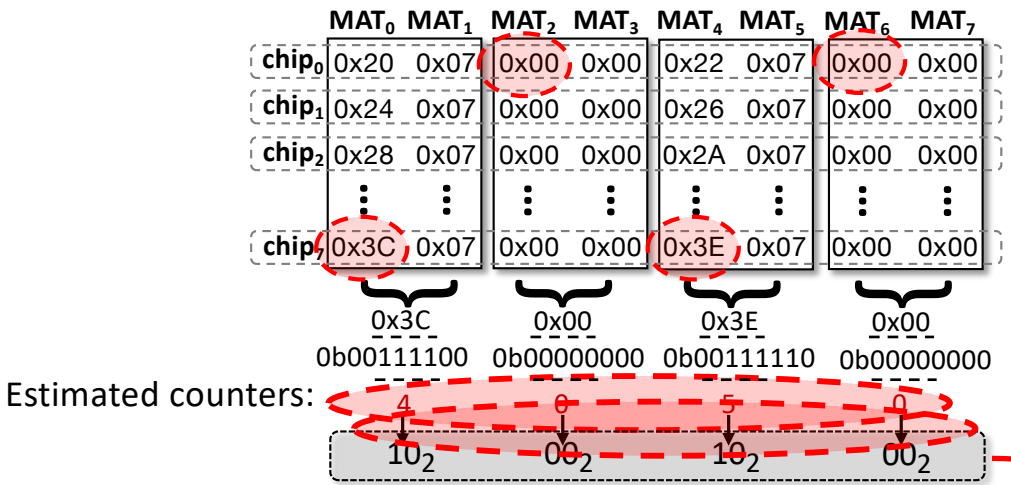


Accurate LRS-counter (LADDER-Basic), $C_{lrs}^w = 3$

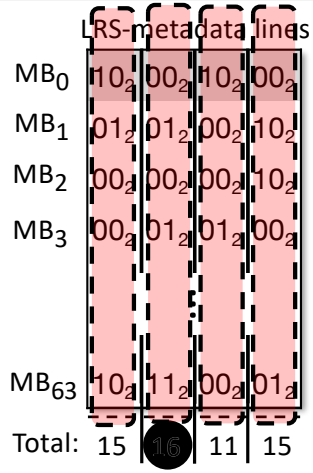
Estimated LRS-counter, $\sum S_i^M = 4$

UCF

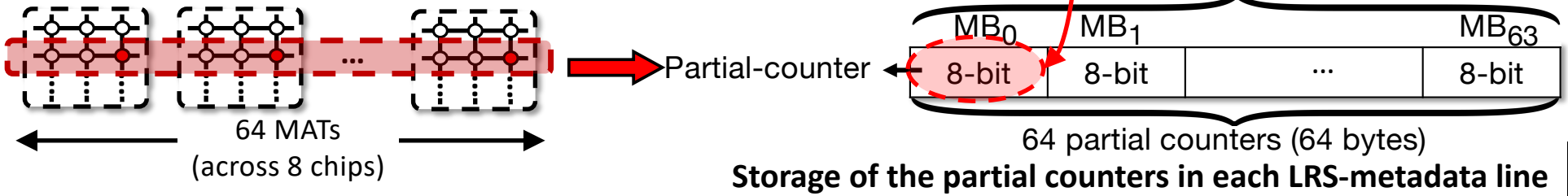
LADDER-Estimate Scheme



Example memory block



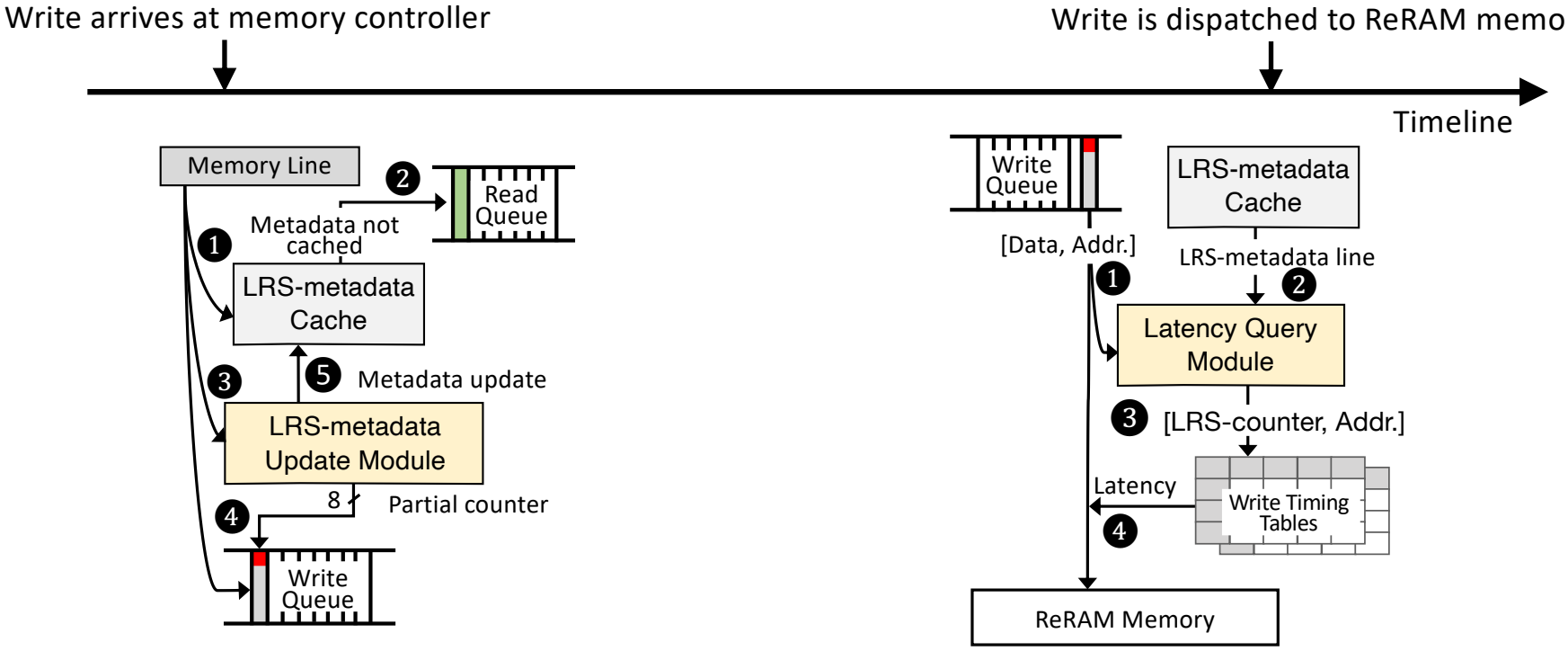
LRS-counter (C^{w}_{lrs}) generation from partial counter



Storage of the partial counters in each LRS-metadata line

UCF

LADDER-Est: Optimized LADDER Logic



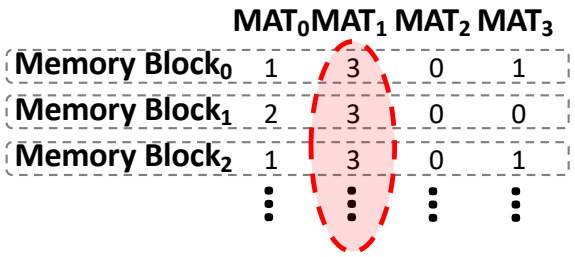
Partial counter generation and LRS-metadata line update

LRS-counter generation and the RESET latency determination

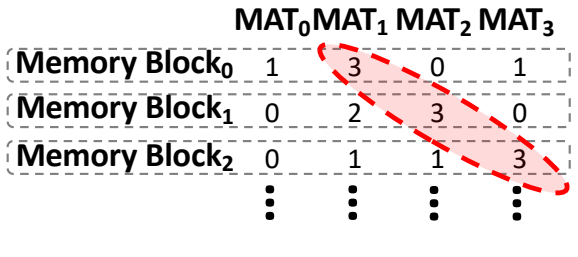
UCF

Improving Estimation Performance with Shifting

- **Observation:** Logical value ‘1’s are clustered together in a few mats.
- Each data block in the WLG leverages a distinct shift based on its position in WL.
- During read, reverse shift is performed to restore original bit order.



LRS cell count in each MAT corresponding to different memory blocks

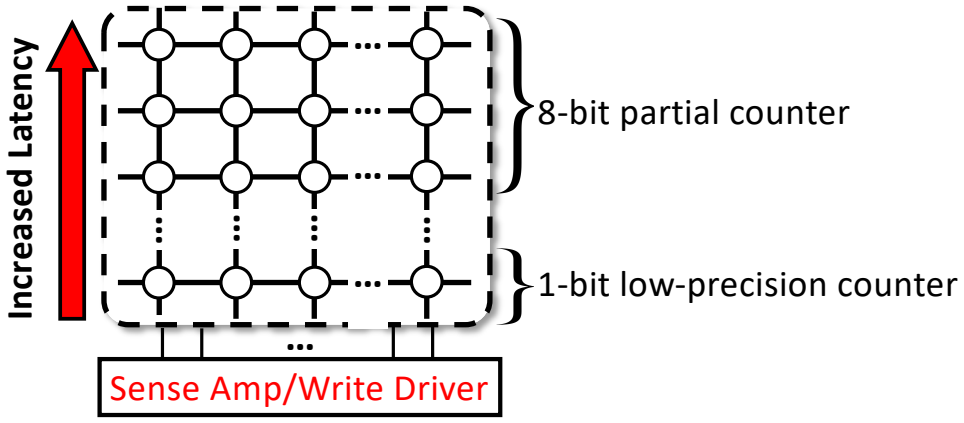


Intra-line bit-level shifting in data lines

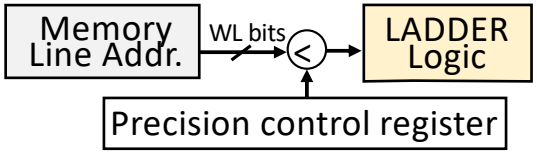
UCF

LADDER-Hybrid: Reducing LRS-metadata Overhead

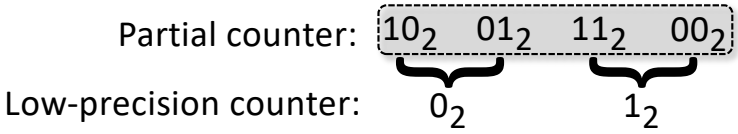
- WLs closer to the write driver are **relatively insensitive** to WL-contents.
- **Multi-Granularity LADDER Counters:** Selectively reduce LRS-counter precision.



ReRAM MAT



LADDER architecture with multi-granularity counter

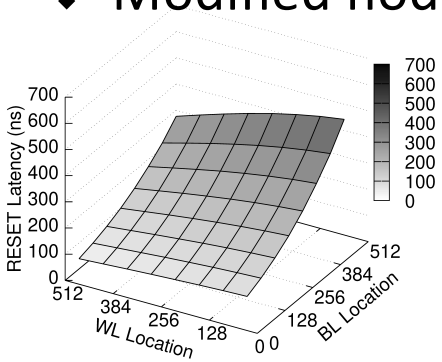


Partial counter to Low-precision counter conversion

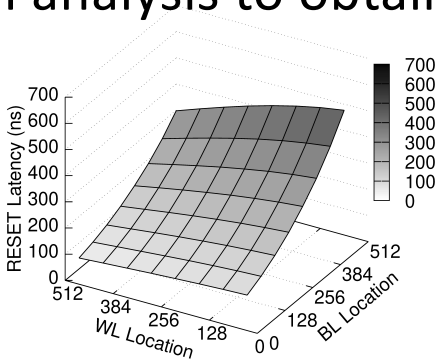
UCF

LADDER Latency Model

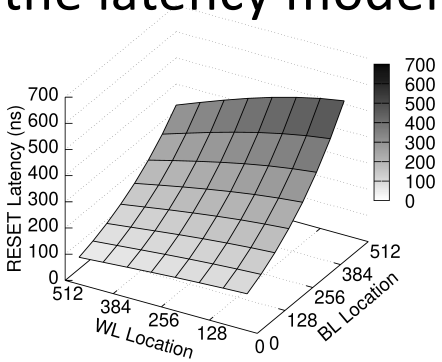
- Modified nodal analysis to obtain the latency model.



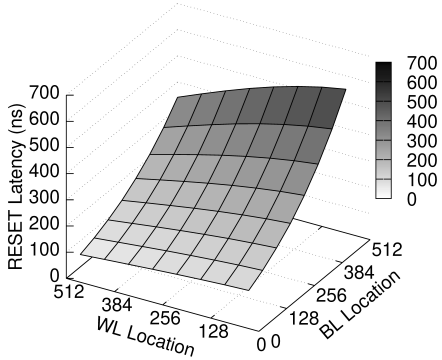
C^{w}_{lrs} range $< 0, 63 >$



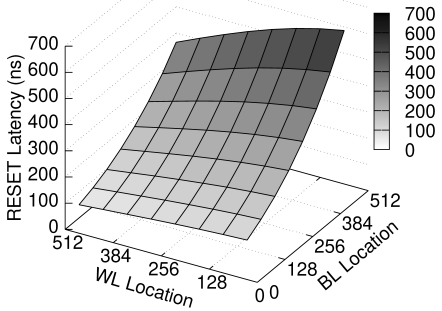
C^{w}_{lrs} range $< 64, 127 >$



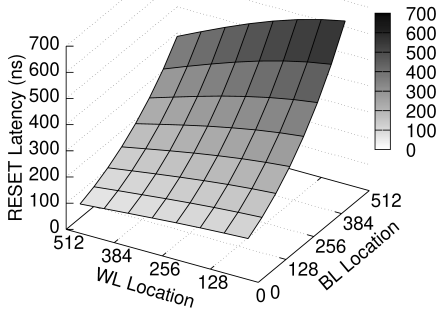
C^{w}_{lrs} range $< 128, 191 >$



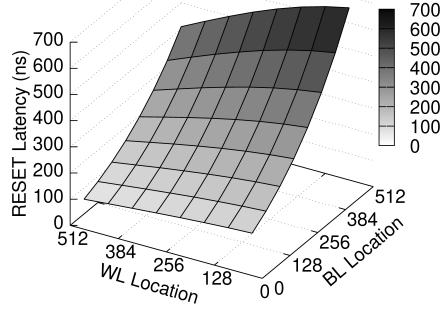
C^{w}_{lrs} range $< 192, 255 >$



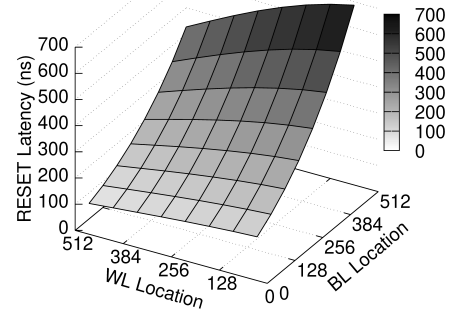
C^{w}_{lrs} range $< 256, 319 >$



C^{w}_{lrs} range $< 320, 383 >$



C^{w}_{lrs} range $< 384, 447 >$



C^{w}_{lrs} range $< 448, 512 >$

Derived latency tables for different C^{w}_{lrs} (**data pattern**) at varying $\frac{WL}{BL}$(**Location**)

UCF

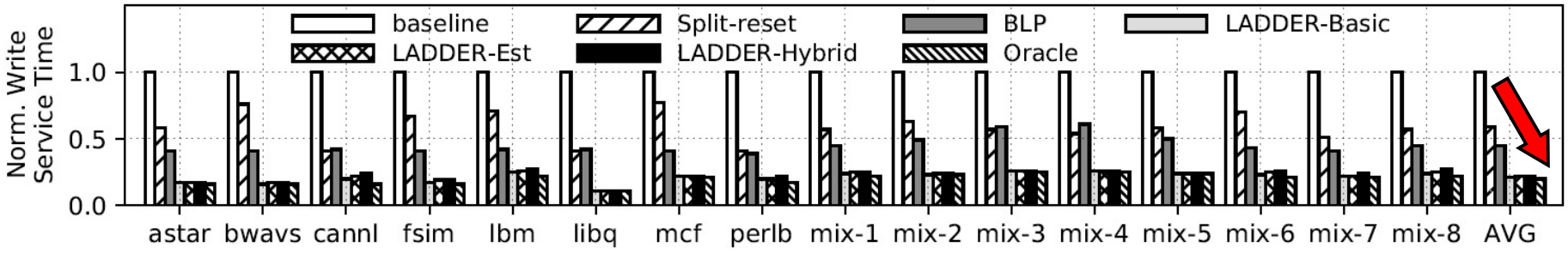
Evaluation Methodology

- **Simulator:** Gem5 full-system simulation (**Kernel:** Linux 3.4).
  - **CPU:** 4-core, Out-of-order, x86
  - **Cache:** Private L1/L2 cache, shared L3 cache
  - **Memory:** 16GB dual channel ReRAM memory, 256 MATs/bank, 512x512 crossbar
- **Workloads:** SPEC2006 (reference input), PARSEC2 (sim-large input).
- **Schemes:**
  - **Baseline:** Worst case fixed latency
  - **Prior state-of-the-art schemes:** Bitline data pattern (BLP)[1], Split-reset[2]
  - **LADDER:** LADDER-Basic, LADDER-Estimation with shifting, LADDER-Hybrid
  - **Oracle:** Location and content-aware writes with no overhead (theoretical)

1. *Exploiting In-memory Data Patterns for Performance Improvement on Crossbar Resistive Memory*, **IEEE TCAD'19**
2. *Overcoming the challenges of crossbar resistive memory architectures*, **HPCA'15**

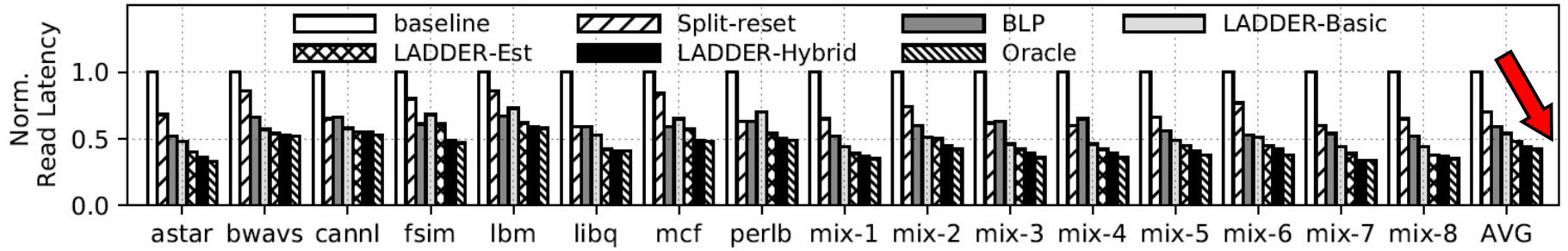
Evaluation: Read/Write Performance

79% write latency reduction compared to baseline **37% and 23% reduction compared to Split-reset and BLP**



Average write service time to ReRAM memory

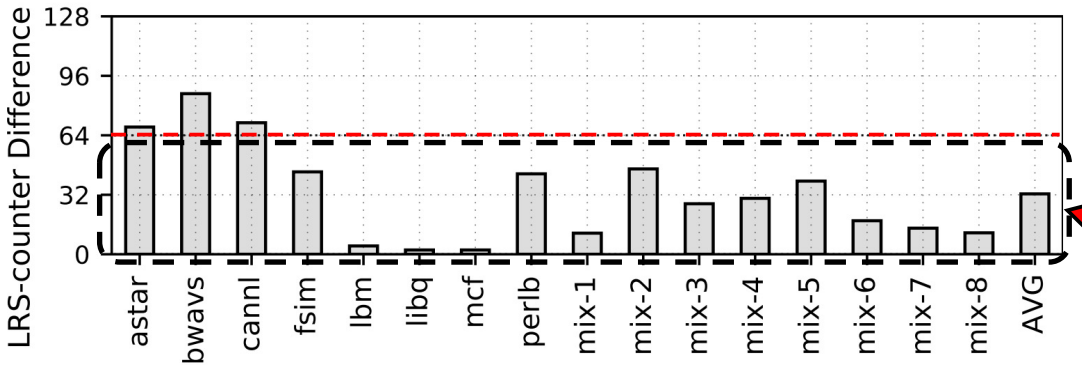
56% read latency reduction compared to baseline **37% and 16% reduction compared to Split-reset and BLP**



Average latency for processor data reads

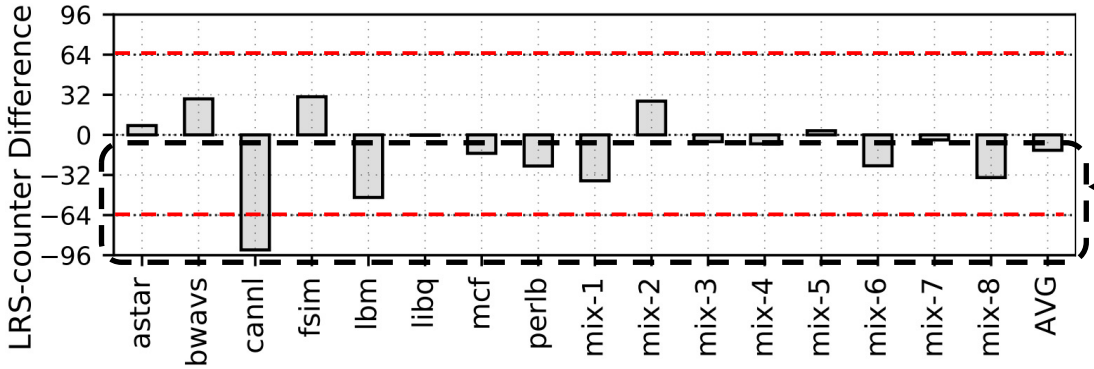
UCF

Evaluation: LRS-metadata Estimation Effectiveness



Typically, RESET latency **does not change**

LRS-counter differences between LADDER-basic and LADDER-Est without Shifting



Improved LRS-counter (compared to accurate counting)

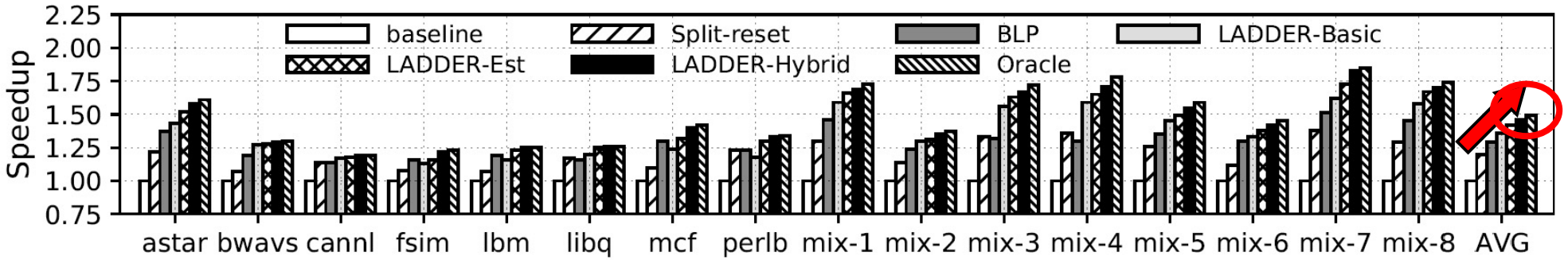
LRS-counter differences between LADDER-basic and LADDER-Est with Shifting

UCF

Evaluation: Overall Speedup

On average, 22% and 13% speedup over Split-reset and BLP respectively

98% of the performance of the ideal scheme



UCF

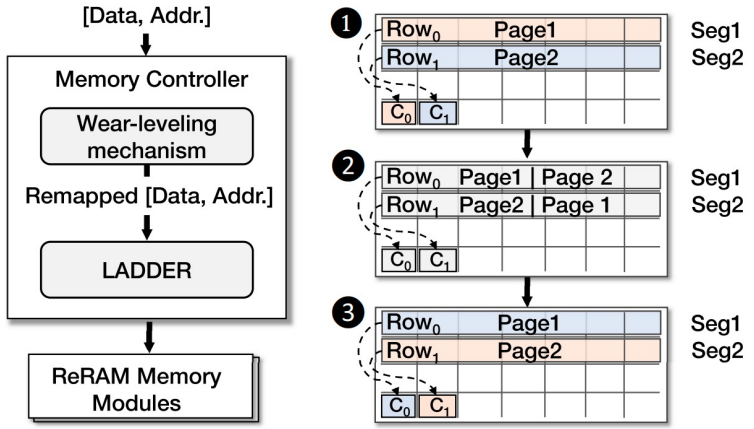
Evaluation: Hardware Overhead Analysis

- ReRAM storage overhead: **1.56%** for **LADDER-Est** and **0.97%** for **LADDER-Hybrid.**
- On-chip storage for LADDER latency table: **512 Bytes** (8x8x8).
- LADDER logic and LRS-metadata cache overhead.
  - Implemented LADDER logic using Verilog
  - Synthesized using Synopsis Design Compiler with 45nm
  - Used CACTI7 to *LRS-metadata Cache*

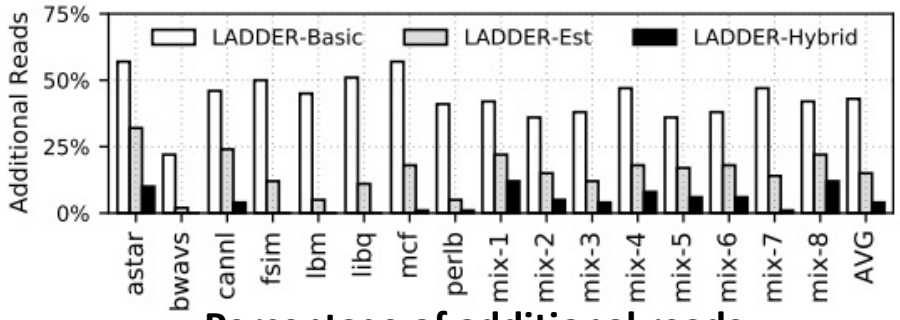
Module	**Area** (mm^2)	**Power** (mW)	**Latency** (ns)
LRS-metadata Update Module	0.0061	3.71	0.17
Latency Query Module	0.0047	6.57	0.32
LRS-metadata Cache (64KB)	0.2442	48.83	0.81

More on Paper

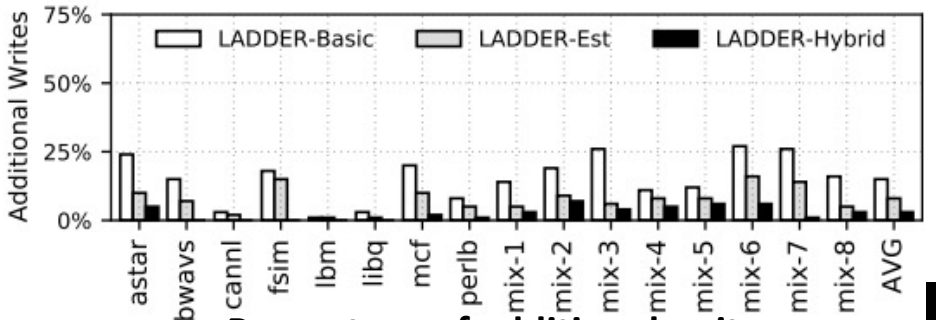
- **Additional details of LADDER designs.**
- **Overhead analysis of LADDER:**
  - Overhead of metadata maintenance
  - Dynamic energy consumption overheads
- **LADDER with wear-leveling techniques.**
- **Crash-consistency of LADDER.**
- **And more…**



Wear-leveling in LADDER



Percentage of additional reads



Percentage of additional writes

UCF

Conclusions

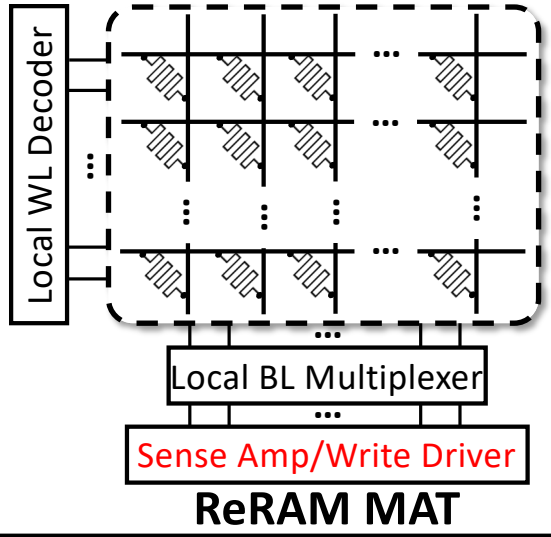
- **LADDER –** a *processor-side* scheme improving write performance for ReRAM crossbar main memories.
- A counter-based mechanism (LRS-metadata) to track WL data pattern.
- Several novel optimizations to further enhance LADDER.
- Achieves considerable performance and energy efficiency improvements over state-of-the-art techniques with minimal overheads.

Thanks! Questions?

Md Hafizul Islam Chowdhuryy
Email: reyad@knights.ucf.edu

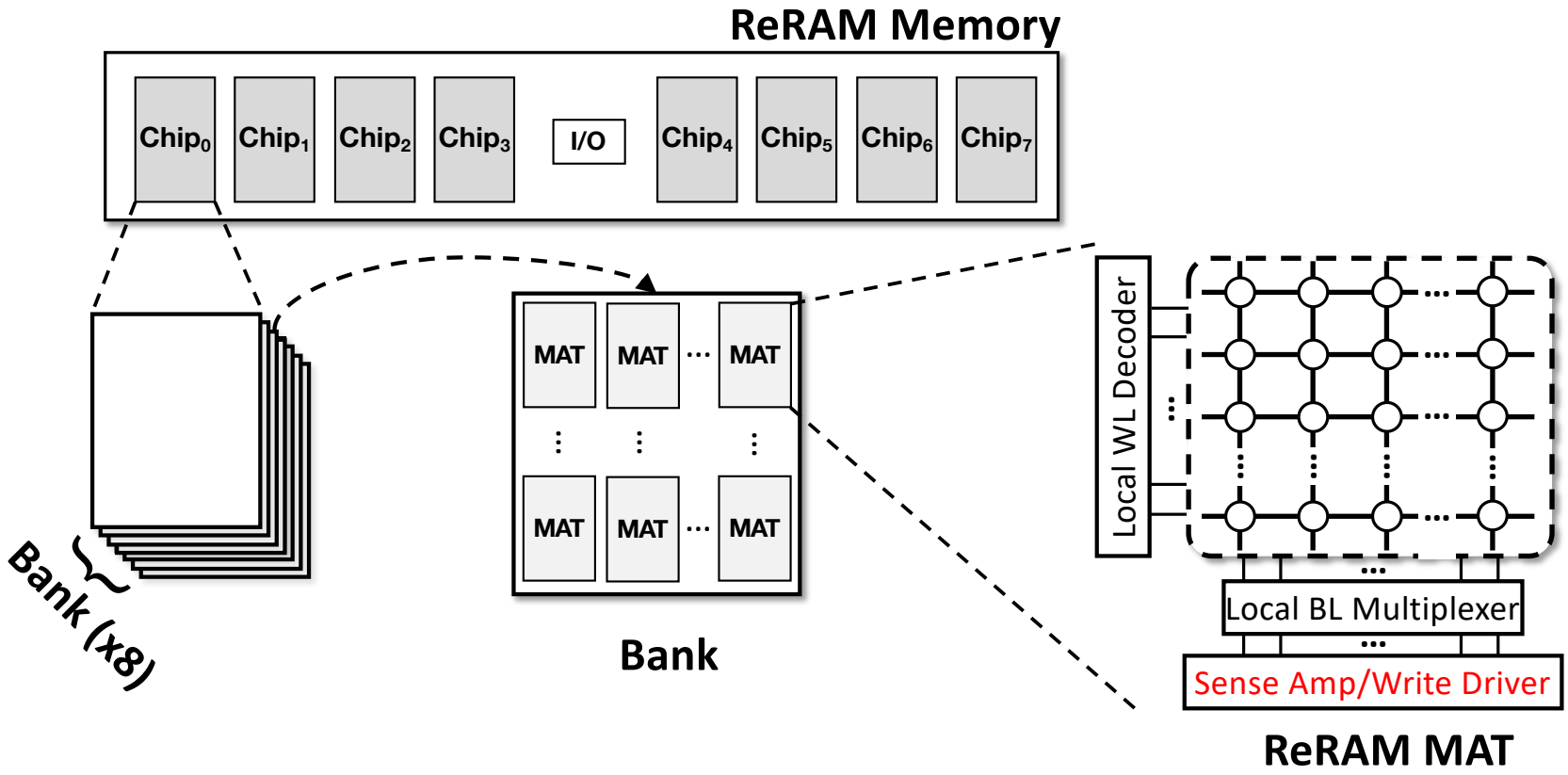
ReRAM Memory Organization

- ReRAM cells are arranged in dense array structure (called **Crossbar**).
  - Increases area efficiency
- ReRAM cell arrays form ***MAT*** (i.e., crossbar size of 512x512 cells).
  - MAT contains peripheral circuitry to support read/write
  - MAT is the basic unit, which forms *banks, chips* and *ranks* of a ReRAM module



UCF

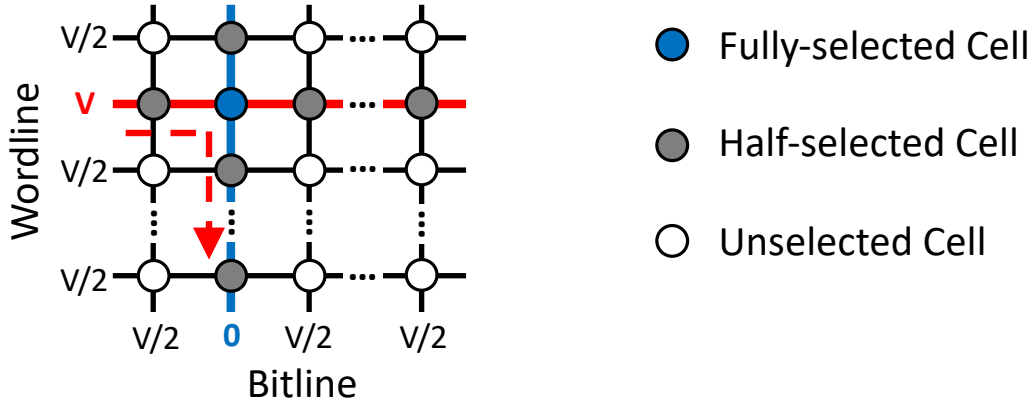
ReRAM Memory Organization



UCF

ReRAM Write Operation

- ReRAM write operation involves two phases.
  - **RESET:** Transition from *low-resistive state* to *high-resistive state* (i.e., '1' → '0')
  - **SET:** Transition from *high-resistive state* to *low-resistive state* (i.e., '0' → '1')

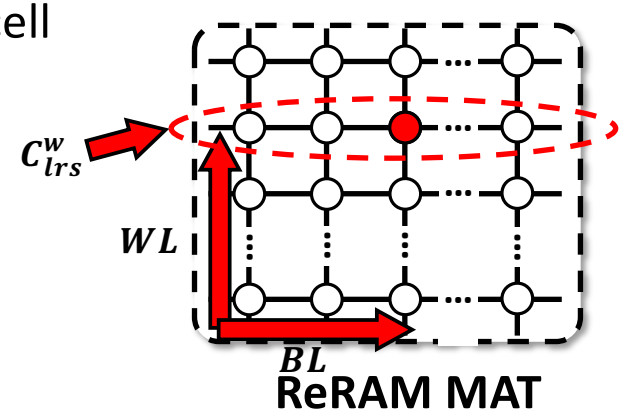


ReRAM Crossbar

UCF

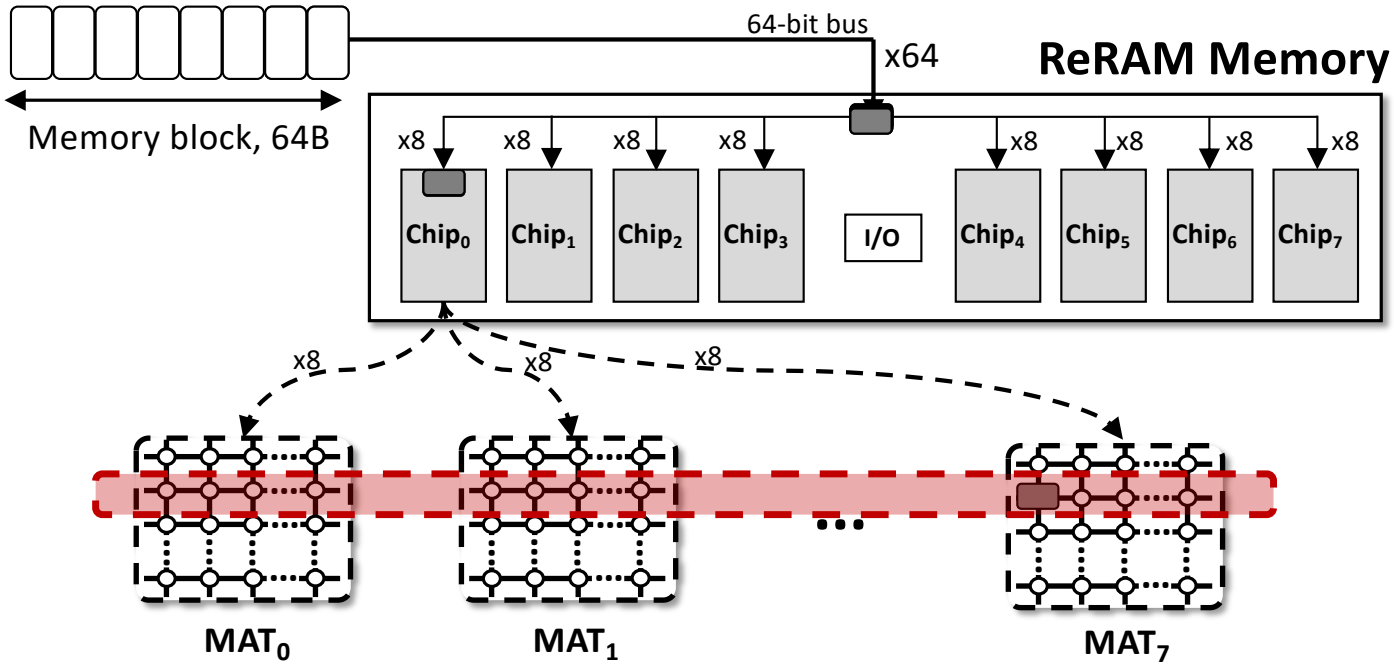
Location & WL-Content Aware Latency Model

- Bitline content-aware scheme requires either:
  - Specialized circuit support in memory to count LRS cell, or
  - Non-trivial overhead of tracking LRS count in many bitlines
- **Tradeoff:** Use LRS cell count in WL only (called *LRS metadata*)
- WL and BL location and WL content of target cell, $< \boldsymbol{WL}, \boldsymbol{BL}, \boldsymbol{C_{lrs}^{w}} >$.
  - $\boldsymbol{WL}, \boldsymbol{BL}$: Wordline and bitline location of target cell
  - $\boldsymbol{C_{lrs}^{w}}$: Worst case LRS-count across selected WLs



UCF

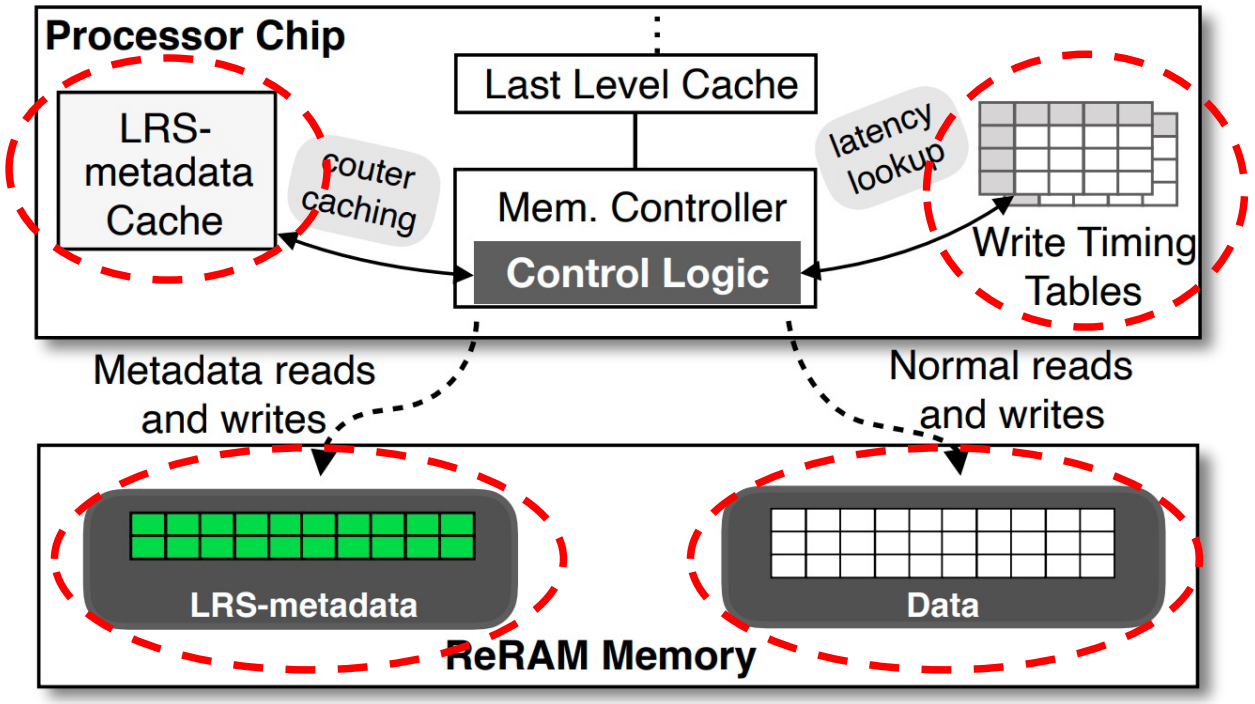
Physical Mapping of Memory Block



- Each MAT stores 1 byte (8-bit) per memory block.
- 8 MATs in each chips are used to store one memory block (64 WLs total).

UCF

Overview of LADDER

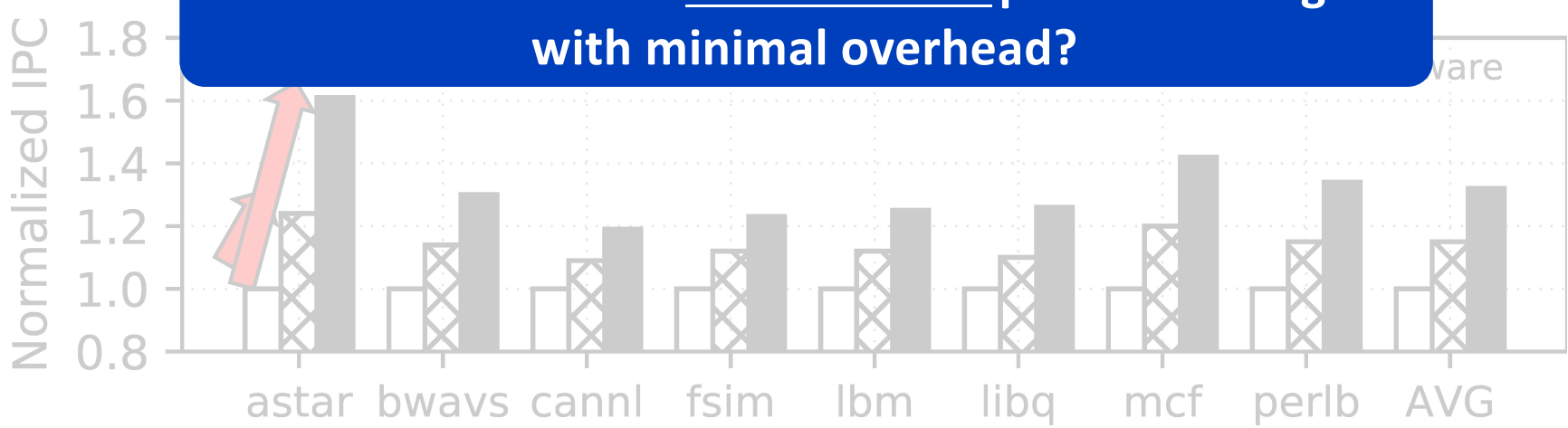


UCF

Potential Speedup using Variable RESET

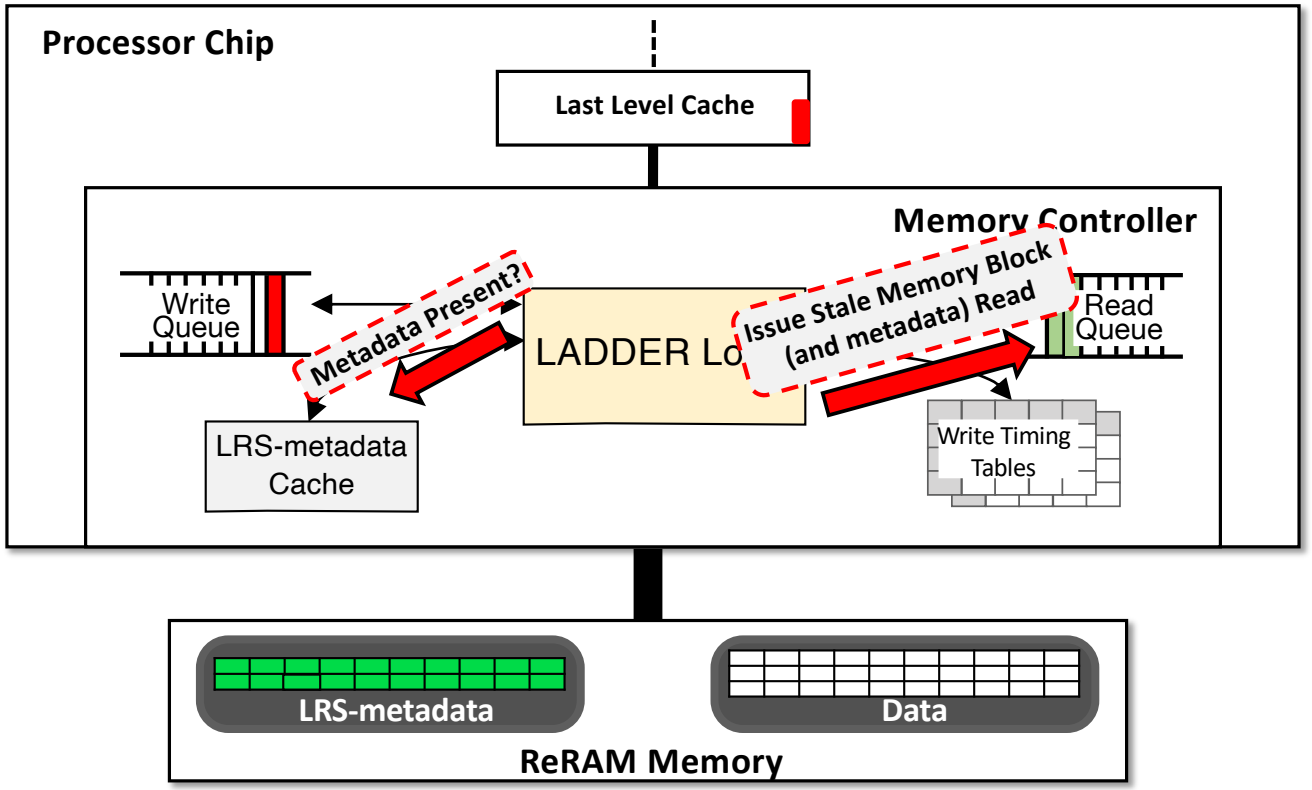
- Ideal speedup using *minimally required* RESET latency compared to worst-case latency.
- Up to **1.24x** performance improvement using location dependence.
- Up to **1.6x** performance improvement using location and content dependence.

***Goal:* Can we realize close-to-ideal performance gain with minimal overhead?**



UCF

LADDER-Basic Data Write Operation



UCF