35th International Conference on Massive Storage Systems and Technology (MSST 2019)

Tiered-ReRAM: A Low Latency and Energy Efficient TLC Crossbar ReRAM Architecture

Yang Zhang, Dan Feng, Wei Tong, Jingning Liu, Chengning Wang, Jie Xu

Huazhong University of Science & Technology

Outline

- Background
- Related Work and Motivation
- Design
- Evaluation
- Conclusion

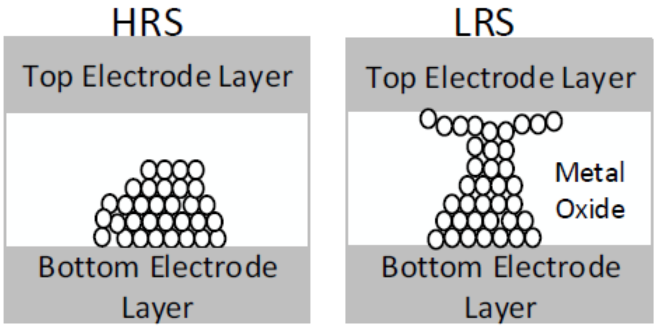
Background

- TLC crossbar ReRAM (Resistive Random Access Memory) is promising to be used as high density storage-class memory

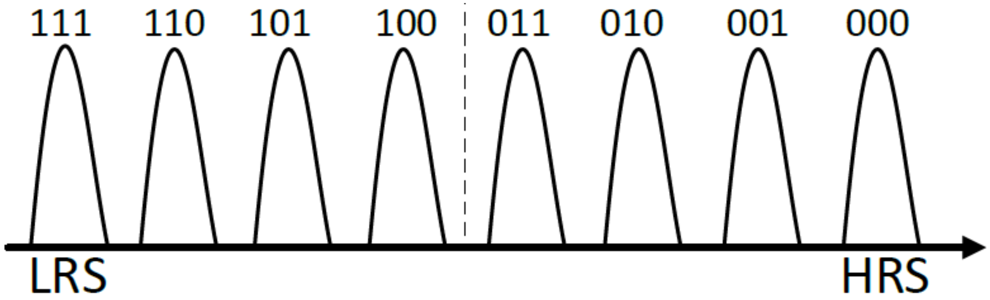
- Advantages
  - Extremely high density
  - High scalability
  - Low standby power
  - Non-volatility

- Disadvantages
  - High write latency and energy
  - IR drop issue
  - Iterative program-and-verify procedure

ReRAM Cell Structure



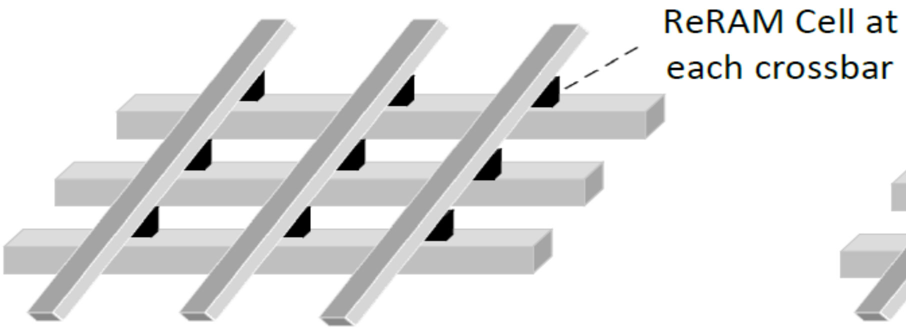
Cell structure



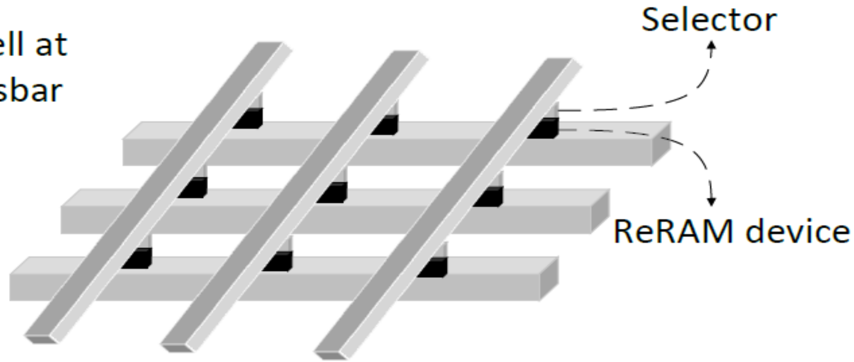
TLC resistance distribution

- **Sandwiched**
- **SLC ReRAM**
  - **HRS(High Resistance State)->0, LRS(Low Resistance State)->1**
  - **RESET (1->0), SET(0->1), RESET latency >> SET latency**
- **TLC ReRAM**
  - **Large resistance differences between HRS and LRS (Ratio can exceed 1000)**
  - **Store three bits into a single cell**

ReRAM Array Structure



0T1R crossbar

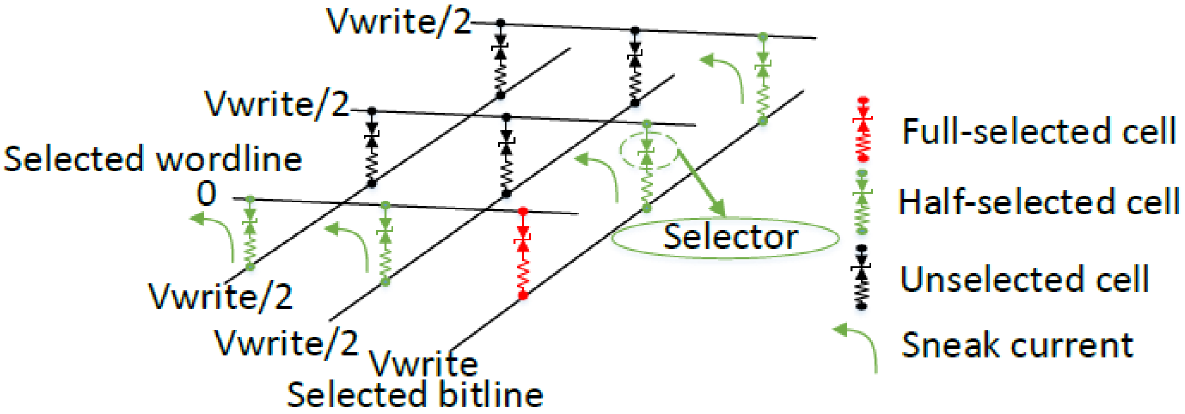


1S1R crossbar

1S1R crossbar structure is more suitable

- **Crossbar**
  - ✓ Smallest planar cell size ($4F^2$)
  - ✓ Better scalability
  - ✓ Lower fabrication cost

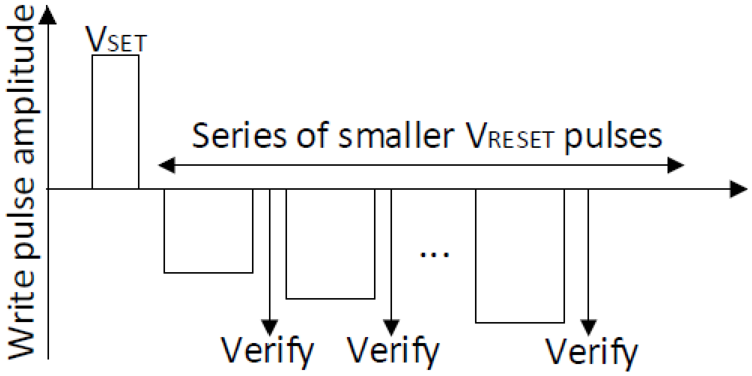
IR Drop Issue



RESET operation in 1S1R crossbar array

- **Sneak currents and wire resistance lead to IR drop issue**
  - ✗ **Significantly increase the RESET latency**
  - ✗ **97% of the total energy is dissipated by the sneak currents of LRS half-selected cells [Lastras et al'HPCA16]**

Iterative Program-and-Verify Procedure



Iterations, Latency and Energy of programming TLC states

Target states	111	110	101	100	011	010	001	000
Iterations	1.21	5.27	10.1	15	14.3	9.83	4.68	1.52
Latency (ns)	14.2	95.4	192	290	383	338.3	286.8	255.2
Energy (pJ)	1.8	13.4	24.3	46.8	94	66.4	41.1	33.6

High write latency and energy have become the greatest design concerns

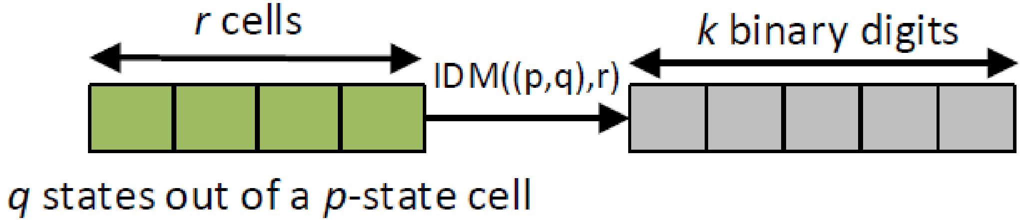
- Program-and-verify (P&V) is commonly used for TLC ReRAM programming
  - Result in high write latency and energy
  - TLC writes with VRESET (e.g., 000) lead to higher latency/energy

Outline

- Background
- Related Work and Motivation
- Design
- Evaluation
- Conclusion

Related Work

- Double-Sided Ground Biasing (DSGB) [Xu et al'HPCA15]
  - ✓Significantly mitigate the IR drops along wordline
  - ×Long length bitlines still result in large IR drops along bitlines
- Incomplete Data Mapping (IDM) [Niu et al'ICCD13]
  - ✓Eliminate certain high-latency and high-energy states of TLC ReRAM
  - ×Sacrifice the capacity of TLC ReRAM
- 0-Dominated Flip Scheme (0-DFS) [Zhang et al'TACO18]
  - ✓Increase the number of high resistance cells ("0" MSB) in crossbar arrays
  - ✓Reduce the leakage energy
  - ×Flip flag bits also sacrifice the capacity of TLC ReRAM



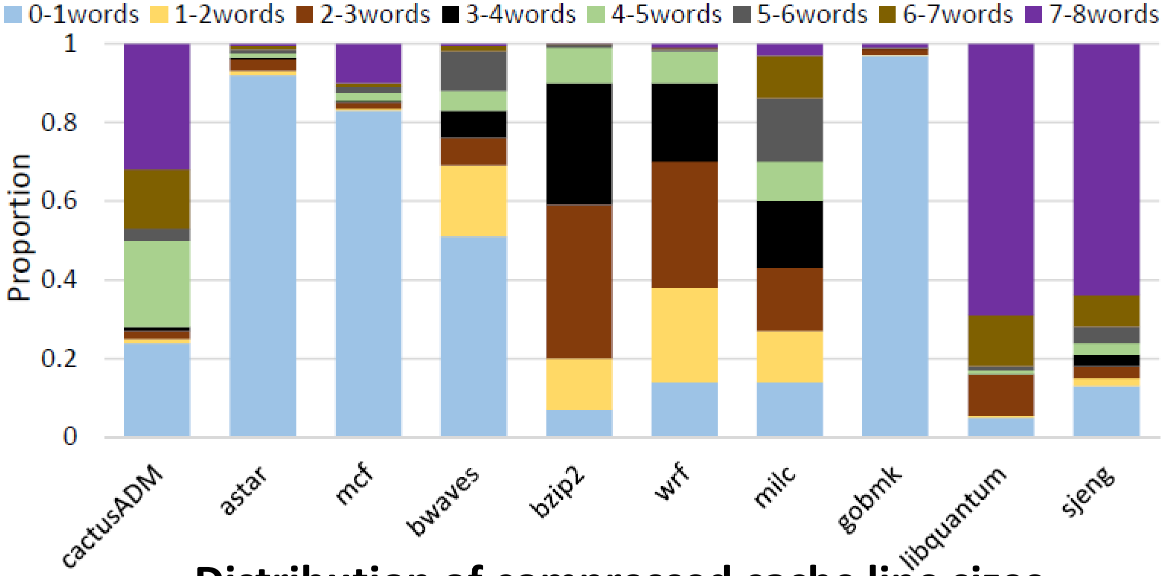
Key Observations

TABLE II: The 64-Bit FPC Patterns with 3-Bit Prefix (Indicated in Red)

Prefix	Pattern Encoded	Example	Compressed Example	Encoded Size	Saved Space
000	Zero run	0x0000000000000000	0x0	3 bits	61 bits
001	8-bit sign extended	0x000000000000007F	0x17F	11 bits	53 bits
010	16-bit sign extended	0xFFFFFFFFFFFFB6B6	0x2B6B6	19 bits	45 bits
011	Half-word sign extended	0x0000000076543210	0x376543210	35 bits	29 bits
100	Half-word, padded with a zero half-word	0x7654321000000000	0x476543210	35 bits	29 bits
101	Two half-words, each a byte sign extended	0xFFFFBEEF00003CAB	0x5BEEF3CAB	35 bits	29 bits
110	Word consisting of four repeated double bytes	0xCAFECAFECAFECAFE	0x6CAFE	19 bits	45 bits

- **Compression techniques can be used to save the storage space**
  - **Frequent Pattern Compression (FPC)**
  - **Saved space of a cache line (eight 64-bit words) may range from 0 to 488 bits**

Key Observations



Distribution of compressed cache line sizes

- **The compressed cache line sizes vary greatly**
  - **Some cache lines can be compressed to smaller than one word**
  - **While some cache lines have more than seven words after compression**

Key Observations

Binary data	111	110	101	100	011	010	001	000
CDM	S7	S6	S5	S4	S3	S2	S1	S0

Write latency= 383ns, Write energy= 322.4pJ, TLC cells= 8

IDM((8, 4), 1)	S7	S7	S6	S6	S7	S0	S5	S6	S6	S0	S6	S0

Write latency= 255.2ns, Write energy= 197.5pJ, TLC cells= 12

IDM((8, 2), 1)	S7	S7	S7	S7	S7	S6	S7	S6	S7	S7	S6	S6	S6	S7	S7	S6	S7	S6	S6	S6	S7	S6	S6	S6

Write latency= 95.4ns, Write energy= 182.4pJ, TLC cells= 24

- **Different IDMs have different tradeoffs in space overhead and write latency/energy**
  - **The IDM that eliminates more states to encode can sacrifice more capacity for more write latency/energy reduction**

Key Observations

- Flip scheme can increase the number of "0" MSBs to reduce the sneak currents and leakage energy
  - 0-Dominated Flip scheme (0-DFS)
- Different word-size 0-DFSs have different tradeoffs in effects and space overhead
  - The 0-DFS that uses smaller word size can achieve more '0' MSBs with higher space overhead

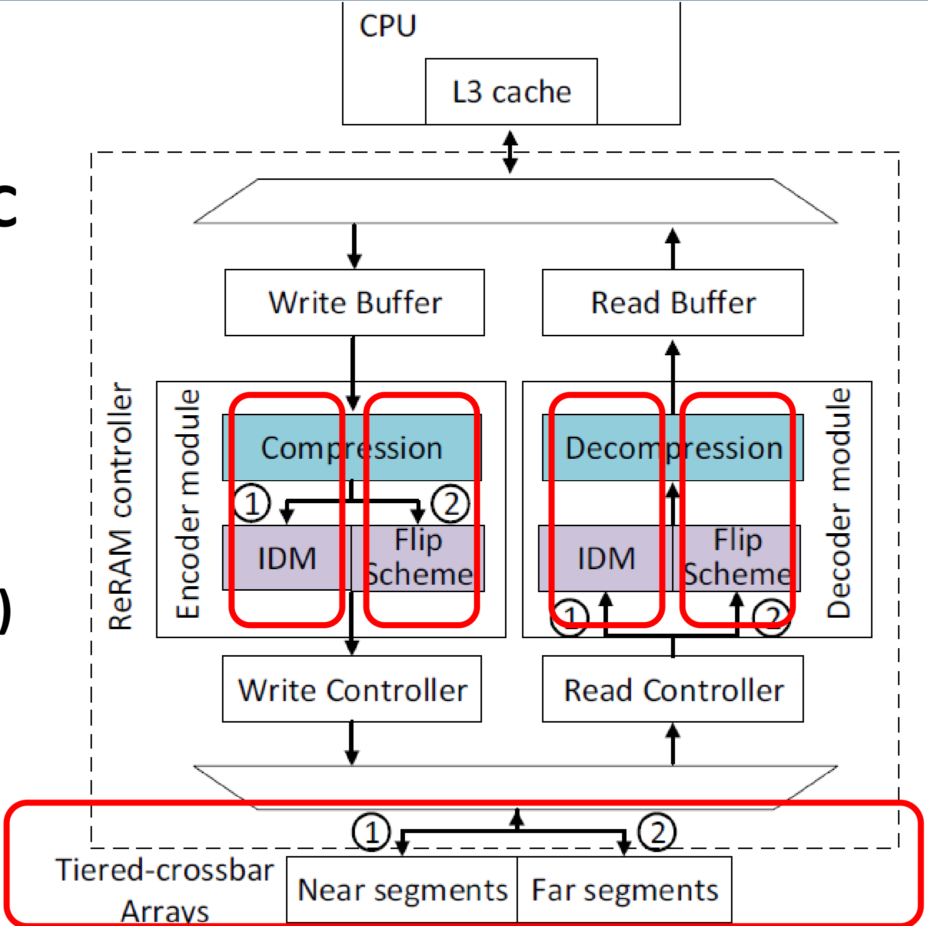
Our idea: Subtly combine the compression technique with IDM and flip scheme

Outline

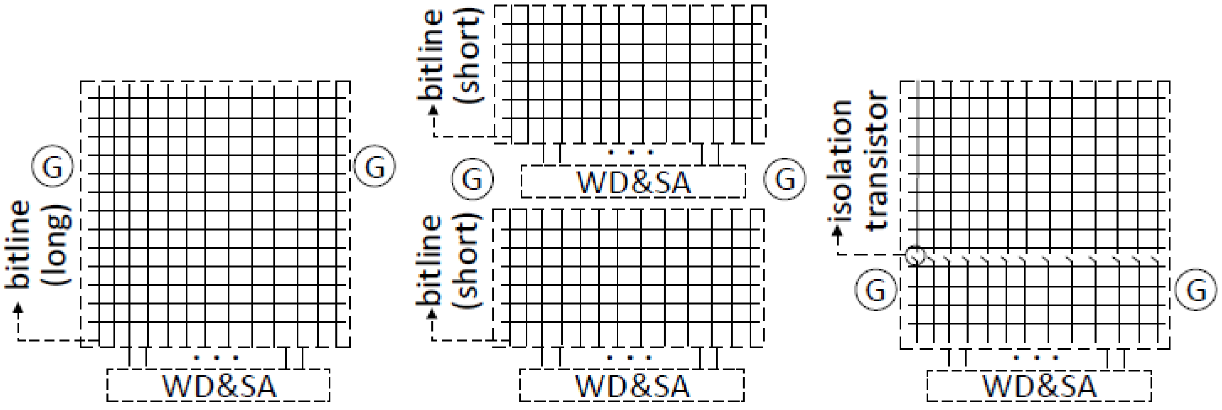
- Background
- Related Work and Motivation
- Design
- Evaluation
- Conclusion

Tiered-ReRAM Architecture

- **Propose Tiered-ReRAM to reduce the write latency and energy of TLC crossbar ReRAM**
- **Three components**
  - **Tiered-crossbar design**
  - **Compression-based IDM (CIDM)**
  - **Compression-based Flip Scheme (CFS)**



Tiered-crossbar Design

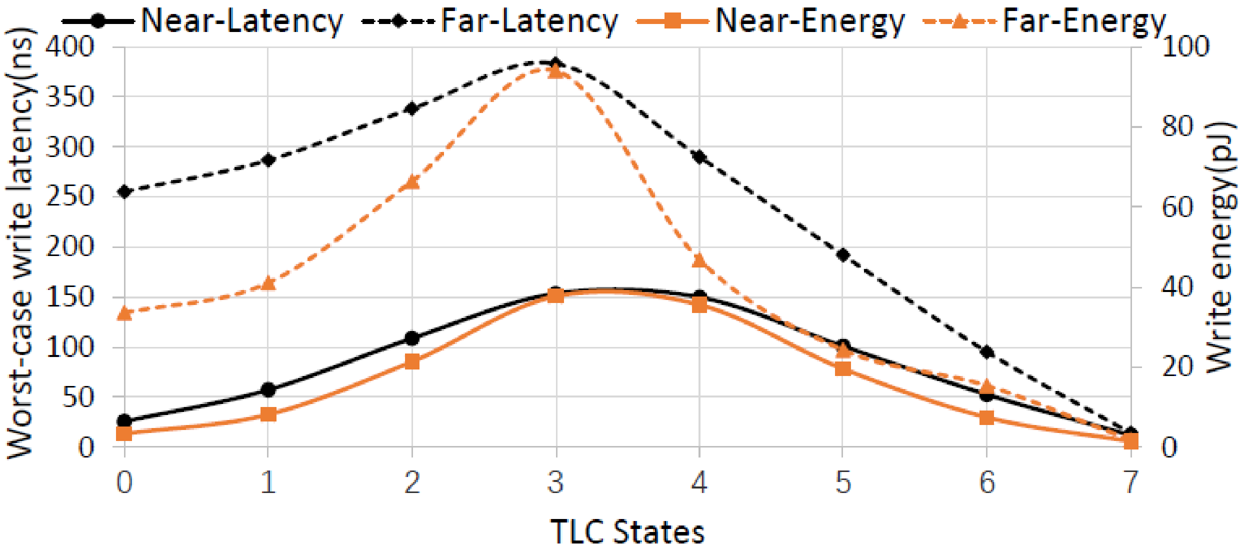


(a) DSGB (b) Latency Opt. (c) Tiered-crossbar

Comparison among different crossbar designs

- **Tiered-crossbar splits each long bitline into two shorter segments using an isolation transistor : *near segment* and *far segment***
  - **To access a ReRAM cell in the near segment (Turn off isolation transistor)**
  - **To access a ReRAM cell in the far segment (Turn on isolation transistor)**
  - **Decrease the additional transistors by 90.9% compared to Latency Opt.**

Tiered-crossbar Design



- **Compared to the far segments, the near segments can achieve 60% write latency reduction and 58% write energy reduction (Near:Far = 1:3)**
- **Remaps hot data to the near segments and cold data to the far segments**

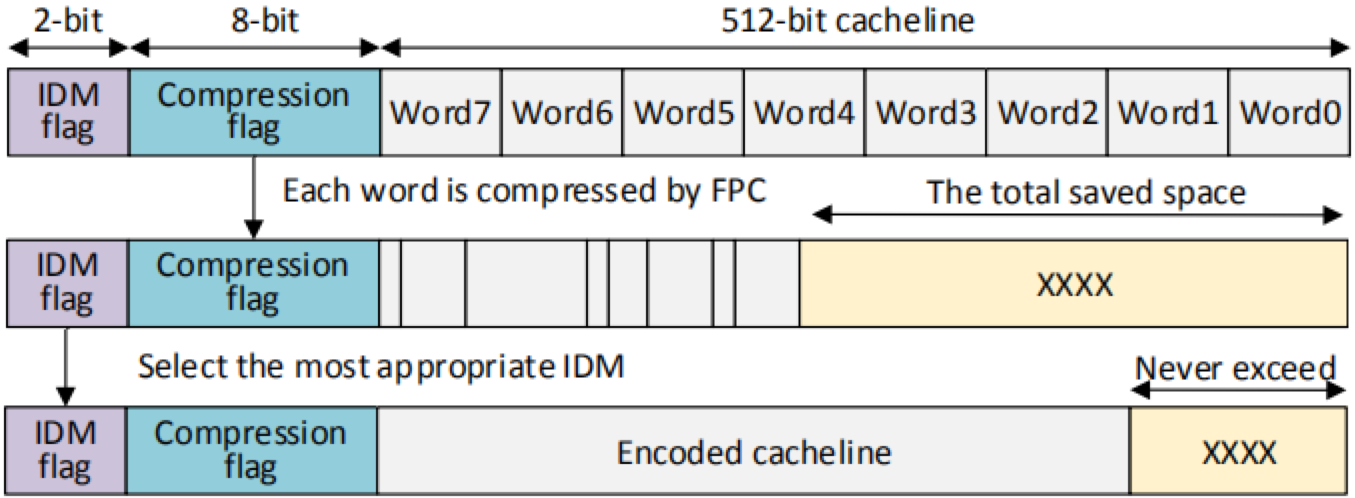
Compression-based IDM (CIDM)

The Most Appropriate IDM

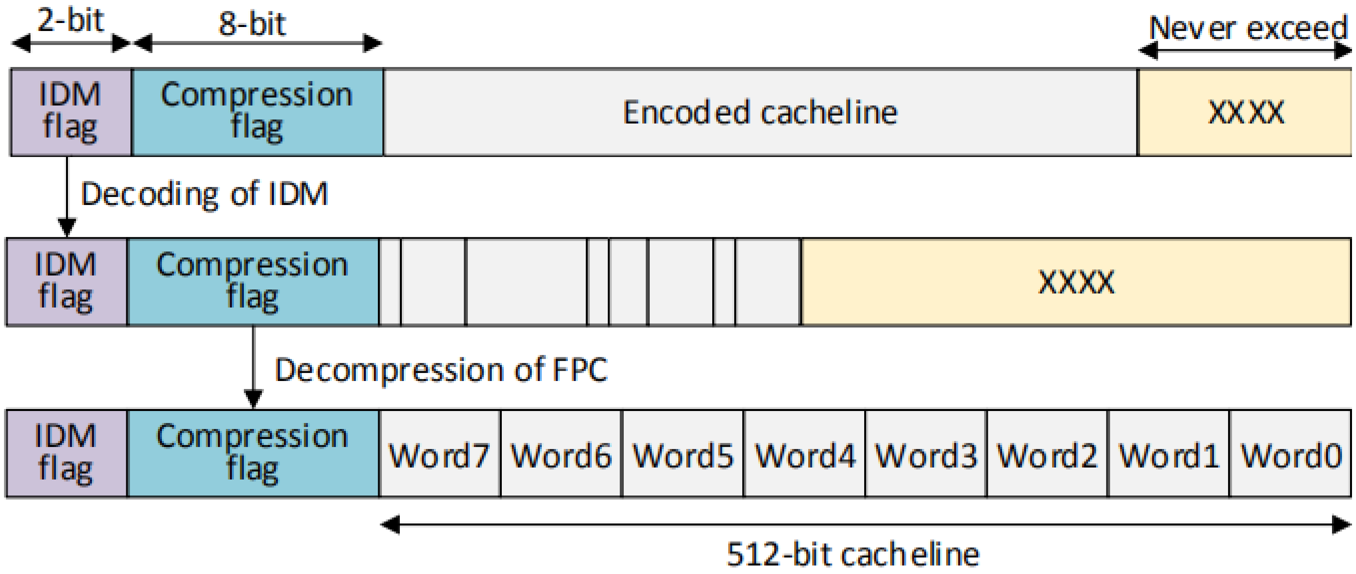
Saved space (bit)	Encoding method	2-bit IDM flag
[341, 488]	IDM((8,2),1)	11
[170, 341)	IDM((8,4),1)	10
[85, 170)	IDM((8,6),2)	01
[0, 85)	CDM	00

- **Dynamically select the most appropriate IDM for each cache line according to the saved space by compression**
- **Implement CIDM in performance-sensitive near segments**
- **Further reduce the write latency/energy**

CIDM Encoder



CIDM Decoder



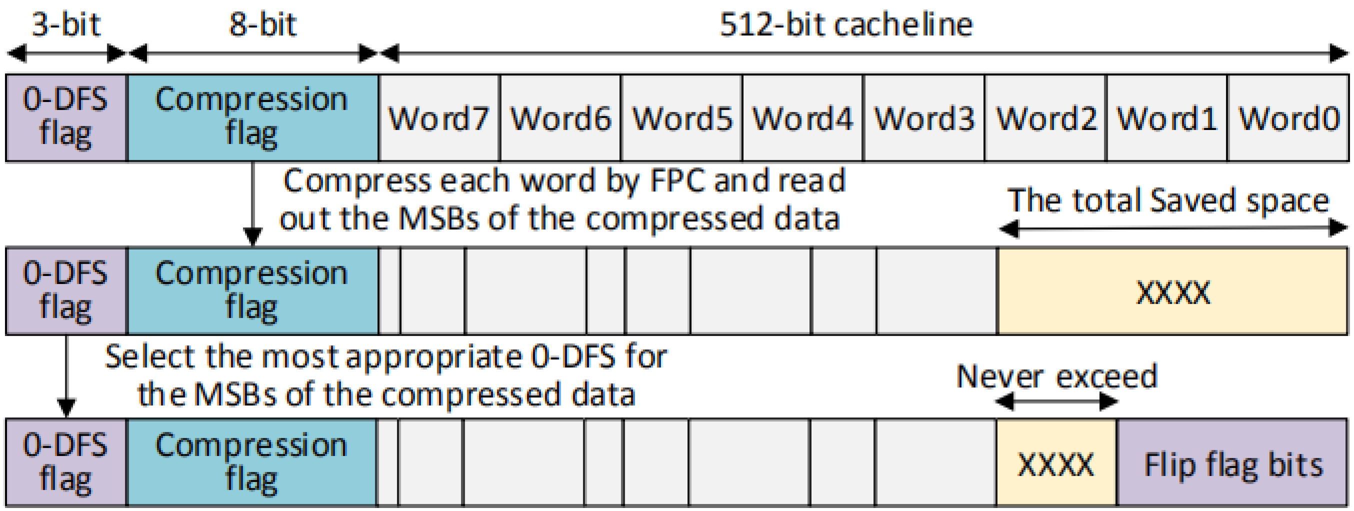
Compression-based Flip Scheme (CFS)

The Most Appropriate 0-DFS

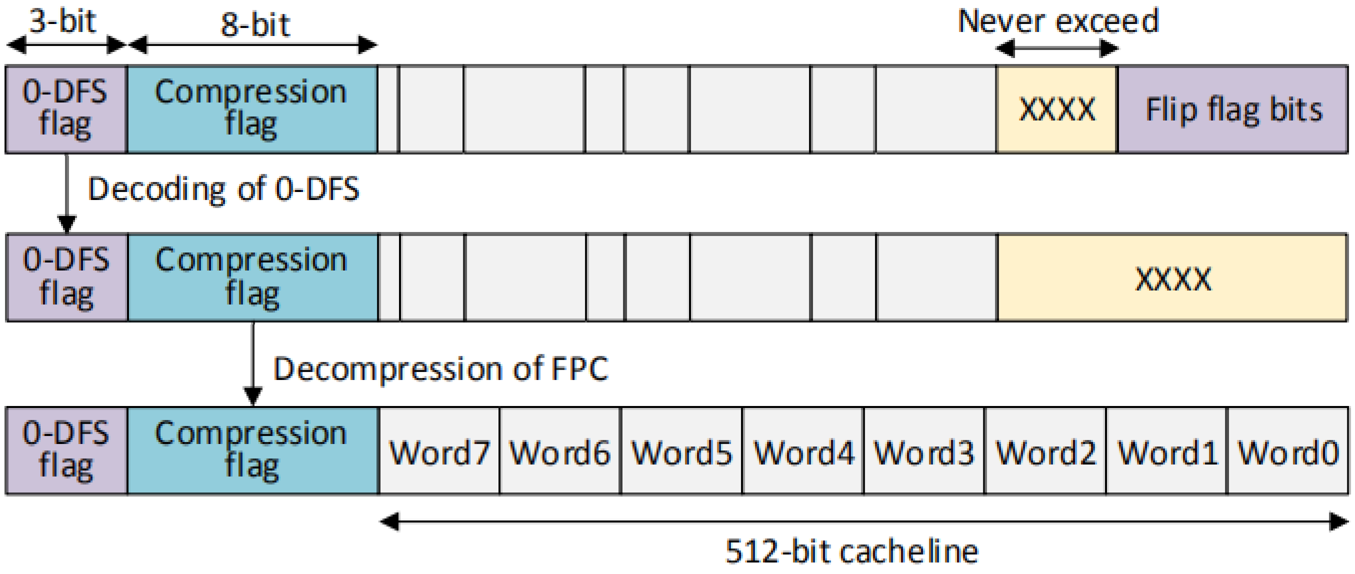
Saved space (bit)	Encoding method	3-bit 0-DFS flag
[74, 488]	2-bit word-size 0-DFS	000
[40, 74)	4-bit word-size 0-DFS	001
[21, 40)	8-bit word-size 0-DFS	010
[11, 21)	16-bit word-size 0-DFS	011
[0, 11)	Without 0-DFS	100

- **Dynamically select the most appropriate 0-DFS for each cache line according to the saved space by compression**
- **Implement CFS in performance-insensitive far segments**
- **Reduce the sneak currents and leakage energy**

CFS Encoder


3-bit
8-bit
512-bit cacheline
0-DFS flag
Compression flag
Word7
Word6
Word5
Word4
Word3
Word2
Word1
Word0
Compress each word by FPC and read out the MSBs of the compressed data
The total Saved space
0-DFS flag
Compression flag
XXXX
Select the most appropriate 0-DFS for the MSBs of the compressed data
Never exceed
0-DFS flag
Compression flag
XXXX
Flip flag bits

CFS Decoder


3-bit
8-bit
Never exceed
0-DFS flag
Compression flag
XXXX
Flip flag bits
Decoding of 0-DFS
0-DFS flag
Compression flag
XXXX
Decompression of FPC
0-DFS flag
Compression flag
Word7
Word6
Word5
Word4
Word3
Word2
Word1
Word0
512-bit cacheline

Outline

- Background
- Related Work and Motivation
- Design
- Evaluation
- Conclusion

Experimental Methodologies

- **Circuit level**
  - Latency/energy parameters from our ReRAM circuit model and NVsim
- **Architecture level**
  - Gem5+NVMain
  - SPEC CPU2006 benchmarks
- **Compared schemes**
  - baseline: DSGB[Xu et al'HPCA15]+IDM((8,6),2)[Niu et al'ICCD13]
  - Tiered-crossbar: Apply the Tiered-crossbar design
  - CIDM: Apply CIDM in the whole crossbar array based on Tiered-crossbar
  - Tiered-ReRAM: Apply CIDM in the near segments and CFS in the far segments based on Tiered-crossbar

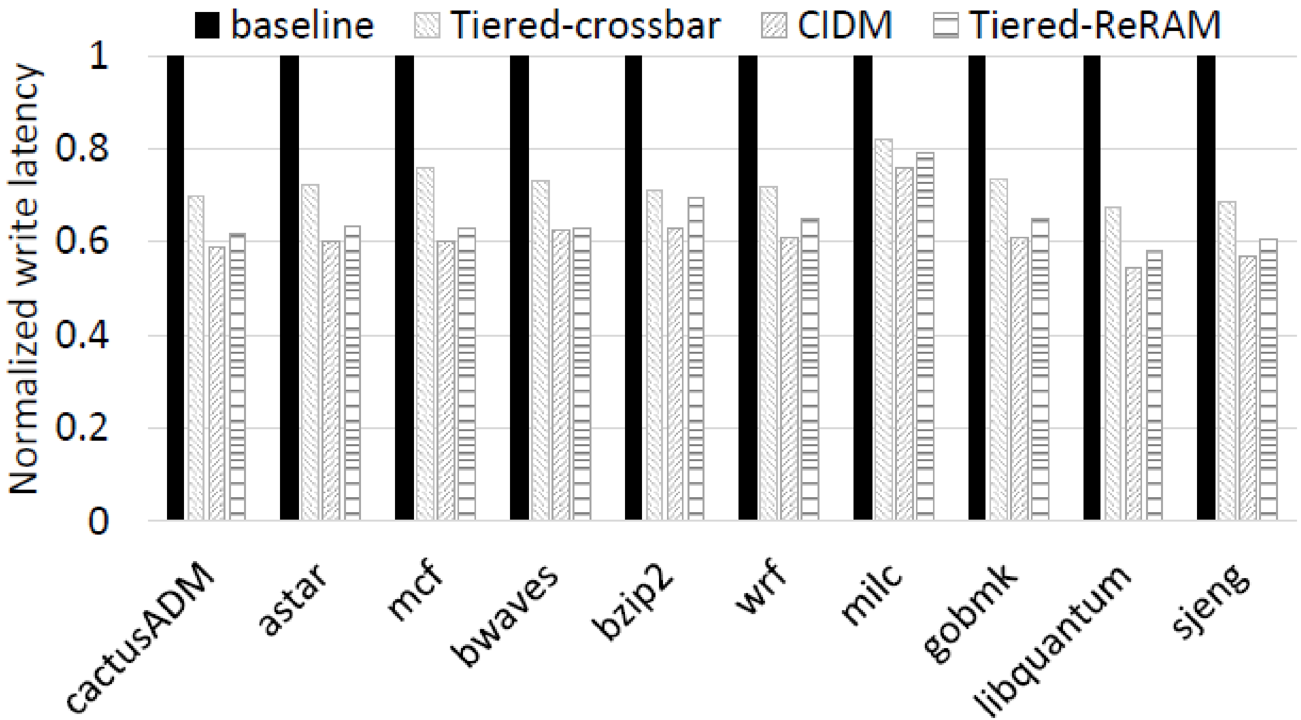
Parameter	Value
CPU	4-Core, out of order, 3GHz, 192-entry recoder buffer, 8 issue width
L1 Cache	Private, 16KB I-cache, 16KB D-cache, 2-way assoc, 2-cycle access latency
L2 Cache	Private, 1MB, 64B cache line, 8-way assoc, 20-cycle access latency
L3 cache	Shared, 16MB, 64B cache line, 16-way assoc, 50-cycle access latency
Main memory	8GB, DDR3-1333, 4 channel, 2 ranks/channel, 32 banks/rank, 1024 crossbar arrays/bank
ReRAM Timing(ns)	tRCD(18), tCL(15), tCWD(13), tFAW(30), tWTR(7.5), tWR(refer to Figure 9)

Simulation Results

- **Improve IPC by 30.6% compared to baseline**

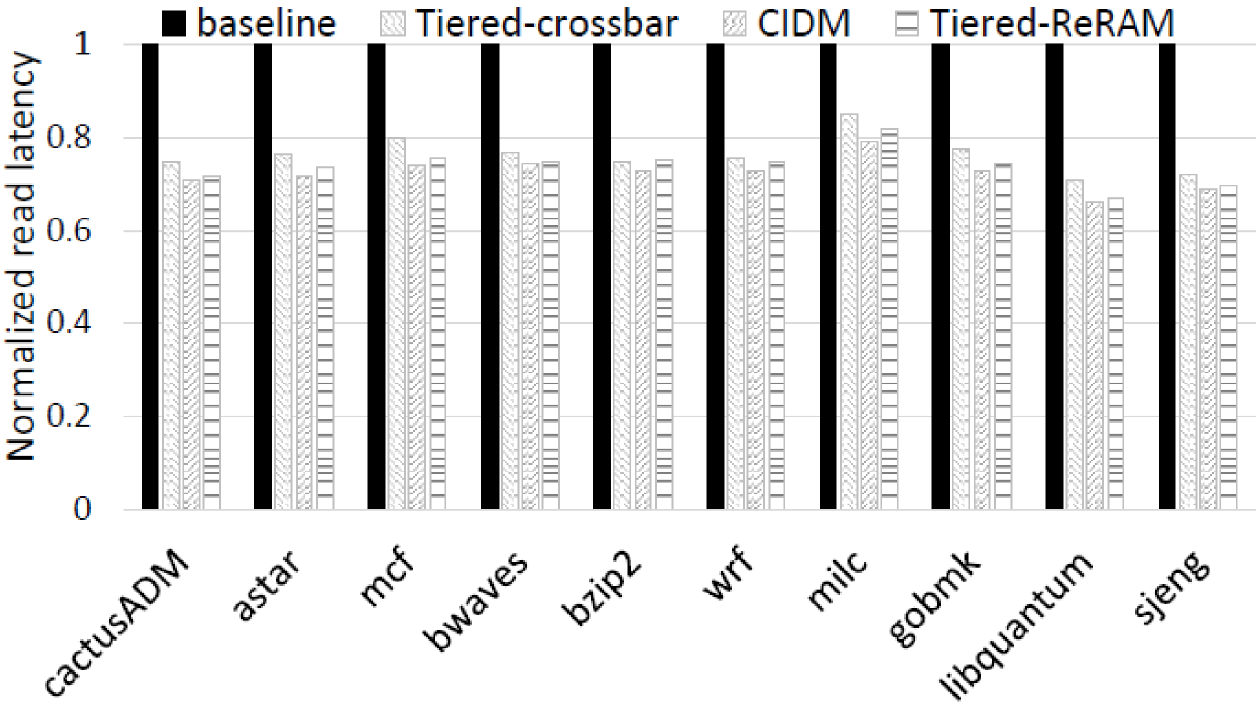
Simulation Results

- **Reduce write latency by 35.2% compared to baseline**



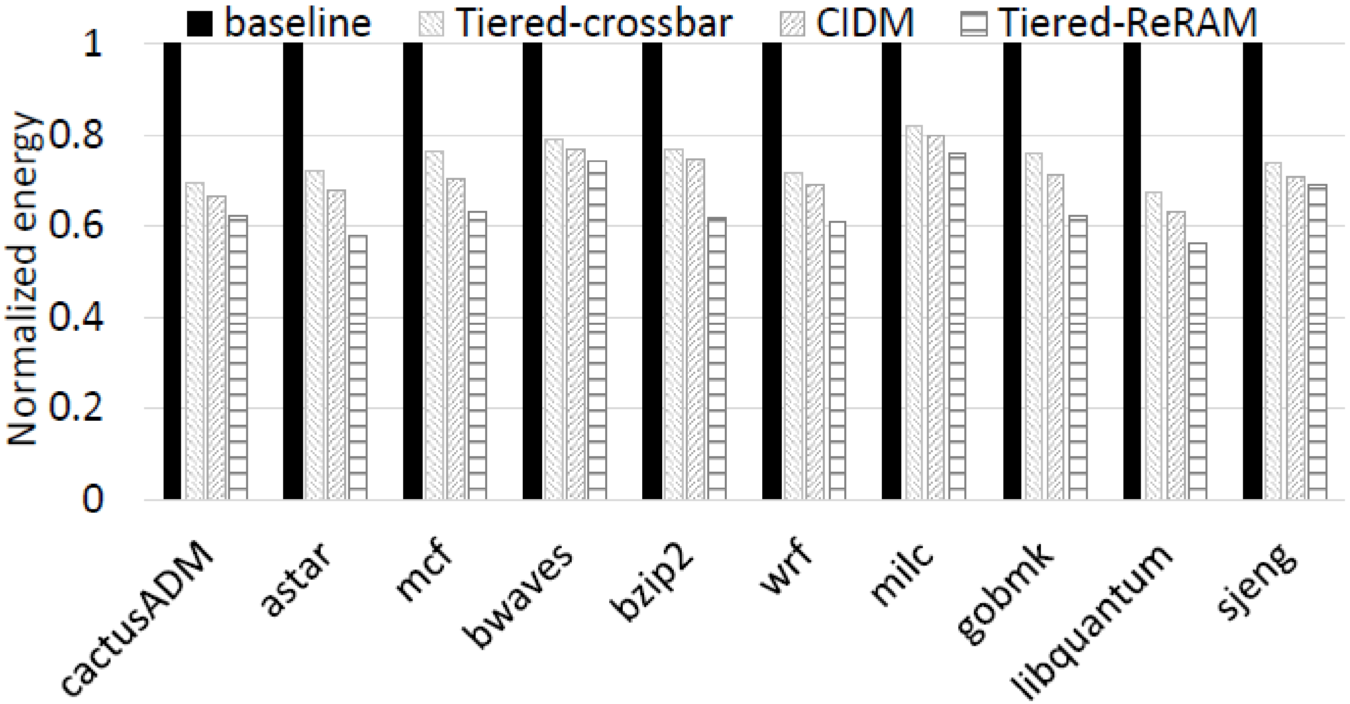
Simulation Results

- **Reduce read latency by 26.1% compared to baseline**



Simulation Results

- **Reduce energy consumption by 35.6% compared to baseline**



Outline

- Background
- Related Work and Motivation
- Design
- Evaluation
- Conclusion

Conclusion

- Challenges
  - IR drop issue
  - Iterative program-and-verify procedure
- Tiered-ReRAM
  - Tiered-crossbar design → Split each long bitline into the near and far segments by an isolation transistor
  - CIDM in the near segments→ Dynamically select the most appropriate IDM for each cache line according to the saved space by compression
  - CFS in the far segments→ Dynamically select the most appropriate flip scheme for each cache line according to the saved space by compression
  - Improve system performance by 30.5% and reduce the energy consumption by 35.6%

Thanks for listening
Q&A