Enhancing lifetime of Phase Change Memory by write variation-aware address remapping

MTP Phase-II Presentation

Deep Bhuinya (204101021)
Supervisor: Prof. Hemangee K. Kapoor

CSE Department, IIT Guwahati

May 19, 2022

Outline

1. DRAM Drawback
2. Non-Volatile Technology
3. Key Challenges
4. State-of-the-art Wear Leveling Algorithms
5. Proposed Method
6. Evaluation
7. Conclusion & Future Work

Outline

1. DRAM Drawback
2. Non-Volatile Technology
3. Key Challenges
4. State-of-the-art Wear Leveling Algorithms
5. Proposed Method
6. Evaluation
7. Conclusion & Future Work

Outline

1. DRAM Drawback
2. Non-Volatile Technology
3. Key Challenges
4. State-of-the-art Wear Leveling Algorithms
5. Proposed Method
6. Evaluation
7. Conclusion & Future Work

Outline

1. DRAM Drawback
2. Non-Volatile Technology
3. Key Challenges
4. State-of-the-art Wear Leveling Algorithms
5. Proposed Method
6. Evaluation
7. Conclusion & Future Work

Outline

1. DRAM Drawback
2. Non-Volatile Technology
3. Key Challenges
4. State-of-the-art Wear Leveling Algorithms
5. Proposed Method
6. Evaluation
7. Conclusion & Future Work

Outline

1. DRAM Drawback
2. Non-Volatile Technology
3. Key Challenges
4. State-of-the-art Wear Leveling Algorithms
5. Proposed Method
6. Evaluation
7. Conclusion & Future Work

Outline

1. DRAM Drawback
2. Non-Volatile Technology
3. Key Challenges
4. State-of-the-art Wear Leveling Algorithms
5. Proposed Method
6. Evaluation
7. Conclusion & Future Work

DRAM Drawback

- As much Leakage energy as Dynamic Energy
  - Nearly 40% in a mid-level IBM eServer[a]
- No way to scale down DRAM below 22nm[b]

[a] Charles Lefurgy et al. "Energy Management for Commercial Servers". In: *Computer* (2003).

[b] "The ITRS report 2009". In: *http://www.itrs2.net/* (2009).

DRAM Drawback

- As much Leakage energy as Dynamic Energy
  - Nearly 40% in a mid-level IBM eServer[a]
- No way to scale down DRAM below 22nm[b]

[a] Charles Lefurgy et al. “Energy Management for Commercial Servers”. In: *Computer* (2003).

[b] “The ITRS report 2009”. In: *http://www.itrs2.net/* (2009).

DRAM Drawback

- As much Leakage energy as Dynamic Energy
  - Nearly 40% in a mid-level IBM eServer[a]
- No way to scale down DRAM below 22nm[b]

[a] Charles Lefurgy et al. “Energy Management for Commercial Servers”. In: *Computer* (2003).

[b] “The ITRS report 2009”. In: http://www.itrs2.net/ (2009).

DRAM Drawback

- As much Leakage energy as Dynamic Energy
  - Nearly 40% in a mid-level IBM eServer[a]
- No way to scale down DRAM below 22nm[b]

[a] Charles Lefurgy et al. "Energy Management for Commercial Servers". In: *Computer* (2003).

[b] "The ITRS report 2009". In: http://www.itrs2.net/ (2009).

Non-Volatile Technology

Exceptionally low leakage energy

- NAND Flash
- STT-RAM
- PCM

Non-Volatile Technology

Exceptionally low leakage energy

- NAND Flash
- STT-RAM
- PCM

Non-Volatile Technology

Exceptionally low leakage energy

- NAND Flash
- STT-RAM
- PCM

Non-Volatile Technology

Exceptionally low leakage energy

- NAND Flash
- STT-RAM
- PCM

Non-Volatile Technology

Exceptionally low leakage energy

- NAND Flash
- STT-RAM
- PCM

PCM: Among the best

High lifetime (than NAND Flash)
Scalability
High density

Key Challenges

- Low Lifetime: $(10^6 \sim 10^8)$ vs 10^{16}
- High write energy consumption and longer access time: Due to the nature of Phase Change Material
- Write Disturbance: Heat produced by a write to one PCM cell may alter the value stored in nearby cells

Key Challenges

- Low Lifetime: $(10^6 \sim 10^8)$ vs 10^{16}
- High write energy consumption and longer access time: Due to the nature of Phase Change Material
- Write Disturbance: Heat produced by a write to one PCM cell may alter the value stored in nearby cells

Key Challenges

- Low Lifetime: $(10^6 \sim 10^8)$ vs 10^{16}
- High write energy consumption and longer access time: Due to the nature of Phase Change Material
- Write Disturbance: Heat produced by a write to one PCM cell may alter the value stored in nearby cells

Industry Manufacturing

Intel, STMicroelectronics, Samsung, IBM, Western Digit, Micron. . .

Industry Manufacturing

Intel, STMicroelectronics, Samsung, IBM, Western Digit, Micron. . .

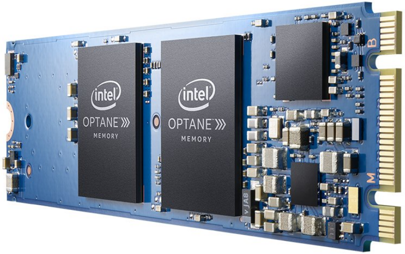


Figure: Intel Optane Memory 16GB PCIe M.2

Improve Lifetime

Lifetime of a cell: $10^6 \sim 10^8$. PCM worn out when several cells are worn out

- Write Reduction
- Wear Leveling
  - Segment Swapping[1]
  - Start-Gap[2]
  - Security Refresh[3]

[1] Ping Zhou et al. "A Durable and Energy Efficient Main Memory Using Phase Change Memory Technology". In: *Proceedings of the 36th Annual International Symposium on Computer Architecture*. 2009.

[2] Moinuddin K. Qureshi et al. "Enhancing Lifetime and Security of PCM-Based Main Memory with Start-Gap Wear Leveling". In: *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*. 2009.

[3] Nak Hee Seong, Dong Hyuk Woo, and Hsien-Hsin S. Lee. "Security Refresh: Prevent Malicious Wear-out and Increase Durability for Phase-Change Memory with Dynamically Randomized Address Mapping". In: *Proceedings of the 37th Annual International Symposium on Computer Architecture*. 2010.

Improve Lifetime

Lifetime of a cell: $10^6 \sim 10^8$. PCM worn out when several cells are worn out

- Write Reduction
- Wear Leveling
  - Segment Swapping[1]
  - Start-Gap[2]
  - Security Refresh[3]

[1] Ping Zhou et al. "A Durable and Energy Efficient Main Memory Using Phase Change Memory Technology". In: *Proceedings of the 36th Annual International Symposium on Computer Architecture*. 2009.

[2] Moinuddin K. Qureshi et al. "Enhancing Lifetime and Security of PCM-Based Main Memory with Start-Gap Wear Leveling". In: *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*. 2009.

[3] Nak Hee Seong, Dong Hyuk Woo, and Hsien-Hsin S. Lee. "Security Refresh: Prevent Malicious Wear-out and Increase Durability for Phase-Change Memory with Dynamically Randomized Address Mapping". In: *Proceedings of the 37th Annual International Symposium on Computer Architecture*. 2010.

Improve Lifetime

Lifetime of a cell: $10^6 \sim 10^8$. PCM worn out when several cells are worn out

- Write Reduction
- Wear Leveling
    - Segment Swapping[1]
    - Start-Gap[2]
    - Security Refresh[3]

[1] Ping Zhou et al. "A Durable and Energy Efficient Main Memory Using Phase Change Memory Technology". In: *Proceedings of the 36th Annual International Symposium on Computer Architecture*. 2009.

[2] Moinuddin K. Qureshi et al. "Enhancing Lifetime and Security of PCM-Based Main Memory with Start-Gap Wear Leveling". In: *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*. 2009.

[3] Nak Hee Seong, Dong Hyuk Woo, and Hsien-Hsin S. Lee. "Security Refresh: Prevent Malicious Wear-out and Increase Durability for Phase-Change Memory with Dynamically Randomized Address Mapping". In: *Proceedings of the 37th Annual International Symposium on Computer Architecture*. 2010.

Improve Lifetime

Lifetime of a cell: $10^6 \sim 10^8$. PCM worn out when several cells are worn out

- Write Reduction
- Wear Leveling
  - Segment Swapping[1]
  - Start-Gap[2]
  - Security Refresh[3]

[1] Ping Zhou et al. “A Durable and Energy Efficient Main Memory Using Phase Change Memory Technology”. In: *Proceedings of the 36th Annual International Symposium on Computer Architecture.* 2009.

[2] Moinuddin K. Qureshi et al. “Enhancing Lifetime and Security of PCM-Based Main Memory with Start-Gap Wear Leveling”. In: *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture.* 2009.

[3] Nak Hee Seong, Dong Hyuk Woo, and Hsien-Hsin S. Lee. “Security Refresh: Prevent Malicious Wear-out and Increase Durability for Phase-Change Memory with Dynamically Randomized Address Mapping”. In: *Proceedings of the 37th Annual International Symposium on Computer Architecture.* 2010.

Improve Lifetime

Lifetime of a cell: $10^6 \sim 10^8$. PCM worn out when several cells are worn out

- Write Reduction
- Wear Leveling
  - Segment Swapping[1]
  - Start-Gap[2]
  - Security Refresh[3]

[1] Ping Zhou et al. “A Durable and Energy Efficient Main Memory Using Phase Change Memory Technology”. In: *Proceedings of the 36th Annual International Symposium on Computer Architecture*. 2009.

[2] Moinuddin K. Qureshi et al. “Enhancing Lifetime and Security of PCM-Based Main Memory with Start-Gap Wear Leveling”. In: *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*. 2009.

[3] Nak Hee Seong, Dong Hyuk Woo, and Hsien-Hsin S. Lee. “Security Refresh: Prevent Malicious Wear-out and Increase Durability for Phase-Change Memory with Dynamically Randomized Address Mapping”. In: *Proceedings of the 37th Annual International Symposium on Computer Architecture*. 2010.

Improve Lifetime

Lifetime of a cell: $10^6 \sim 10^8$. PCM worn out when several cells are worn out

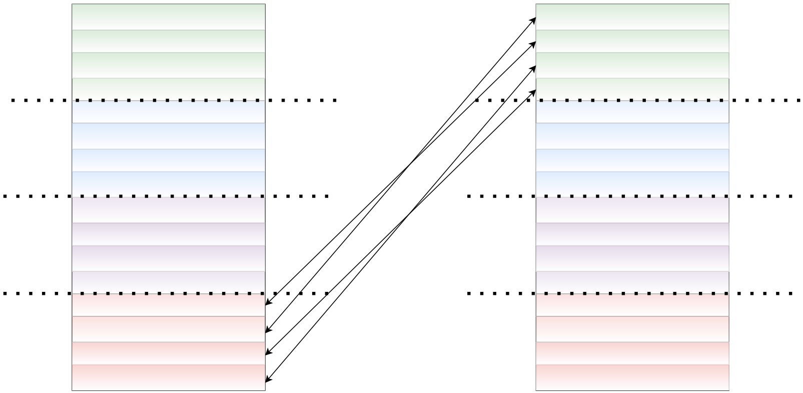
- Write Reduction
- Wear Leveling
  - Segment Swapping[1]
  - Start-Gap[2]
  - Security Refresh[3]

[1] Ping Zhou et al. "A Durable and Energy Efficient Main Memory Using Phase Change Memory Technology". In: *Proceedings of the 36th Annual International Symposium on Computer Architecture.* 2009.

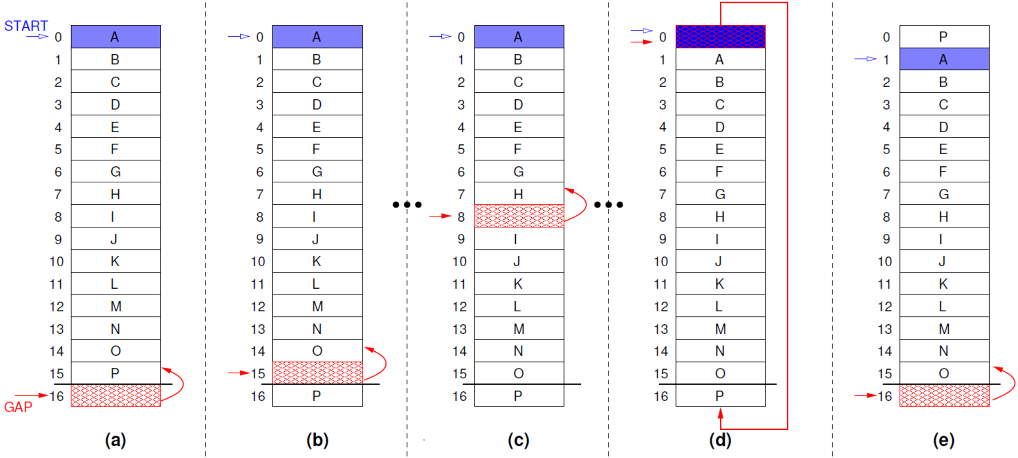
[2] Moinuddin K. Qureshi et al. "Enhancing Lifetime and Security of PCM-Based Main Memory with Start-Gap Wear Leveling". In: *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture.* 2009.

[3] Nak Hee Seong, Dong Hyuk Woo, and Hsien-Hsin S. Lee. "Security Refresh: Prevent Malicious Wear-out and Increase Durability for Phase-Change Memory with Dynamically Randomized Address Mapping". In: *Proceedings of the 37th Annual International Symposium on Computer Architecture.* 2010.

Segment Swapping

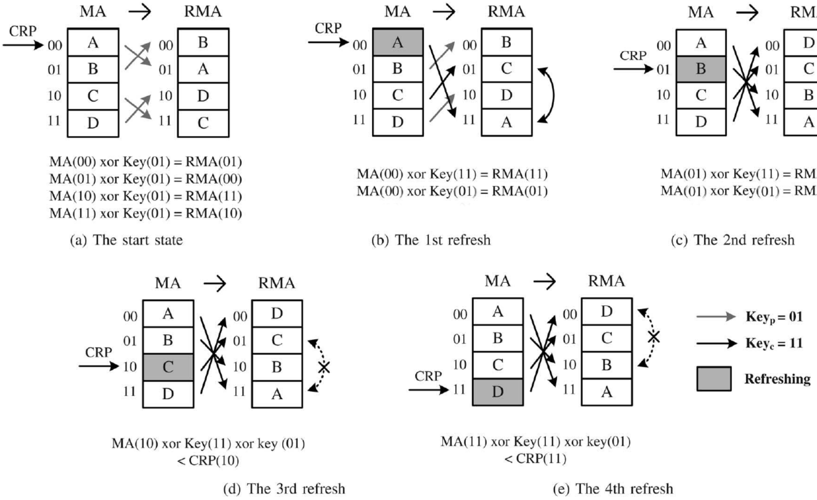


Start-Gap



	(a)	(b)	(c)	(d)	(e)
0	A	A	A		P
1	B	B	B	A	A
2	C	C	C	B	B
3	D	D	D	C	C
4	E	E	E	D	D
5	F	F	F	E	E
6	G	G	G	F	F
7	H	H	H	G	G
8	I	I		H	H
9	J	J	I	I	I
10	K	K	J	J	J
11	L	L	K	K	K
12	M	M	L	L	L
13	N	N	M	M	M
14	O	O	N	N	N
15	P		O	O	O
16		P	P	P	

Security Refresh



MA(00) xor Key(01) = RMA(01)
MA(01) xor Key(01) = RMA(00)
MA(10) xor Key(01) = RMA(11)
MA(11) xor Key(01) = RMA(10)

(a) The start state

MA(00) xor Key(11) = RMA(11)
MA(00) xor Key(01) = RMA(01)

(b) The 1st refresh

MA(01) xor Key(11) = RMA(10)
MA(01) xor Key(01) = RMA(00)

(c) The 2nd refresh

MA(10) xor Key(11) xor key (01)
< CRP(10)

(d) The 3rd refresh

MA(11) xor Key(11) xor key(01)
< CRP(11)

(e) The 4th refresh

Proposed Method: Write variation-aware address remapping

- Memory space is divided in **banks**
- Banks are further divided into **segments**
- Maintain two sets **high** and **low** for every segment
- Initially all addresses are **considered as low**
- A **write counter** for each address is maintained
- Whenever the write count of an address reaches **ADDTHRSLD**, it is **moved to high** set

Proposed Method: Write variation-aware address remapping

- Memory space is divided in **banks**
- Banks are further divided into **segments**
- Maintain two sets **high** and **low** for every segment
- Initially all addresses are **considered as low**
- A **write counter** for each address is maintained
- Whenever the write count of an address reaches **ADDTHRSLD**, it is **moved to high** set

Proposed Method: Write variation-aware address remapping

- Memory space is divided in **banks**
- Banks are further divided into **segments**
- Maintain two sets **high** and **low** for every segment
- Initially all addresses are **considered as low**
- A **write counter** for each address is maintained
- Whenever the write count of an address reaches **ADDTHRSLD**, it is **moved to high** set

Proposed Method: Write variation-aware address remapping

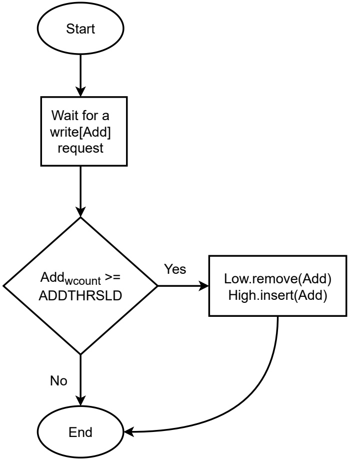
- Memory space is divided in **banks**
- Banks are further divided into **segments**
- Maintain two sets **high** and **low** for every segment
- Initially all addresses are **considered as low**
- A **write counter** for each address is maintained
- Whenever the write count of an address reaches **ADDTHRSLD**, it is **moved to high** set

Proposed Method: Write variation-aware address remapping

- Memory space is divided in **banks**
- Banks are further divided into **segments**
- Maintain two sets **high** and **low** for every segment
- Initially all addresses are **considered as low**
- A **write counter** for each address is maintained
- Whenever the write count of an address reaches **ADDTHRSLD**, it is **moved to high** set

Proposed Method: Write variation-aware address remapping

- Memory space is divided in **banks**
- Banks are further divided into **segments**
- Maintain two sets **high** and **low** for every segment
- Initially all addresses are **considered as low**
- A **write counter** for each address is maintained
- Whenever the write count of an address reaches **ADDTHRSLD**, it is **moved to high** set



Proposed Method: Write variation-aware address remapping

- A **write counter** for each bank is maintained
- Whenever the write count of a bank reaches **THRSLD**, it will go through a complete address mapping round for all of its segments
- For each segment, whatever the high addresses accumulated till that point, mapped to any random low addresses of the same segment
- Once again, all the addresses of that segment will be **considered as low**
- Write count for all the addresses of that segment will be **reset to 0**

Proposed Method: Write variation-aware address remapping

- A **write counter** for each bank is maintained
- Whenever the write count of a bank reaches **THRSLD**, it will go through a complete address mapping round for all of its segments
- For each segment, whatever the high addresses accumulated till that point, mapped to any random low addresses of the same segment
- Once again, all the addresses of that segment will be **considered as low**
- Write count for all the addresses of that segment will be **reset to 0**

Proposed Method: Write variation-aware address remapping

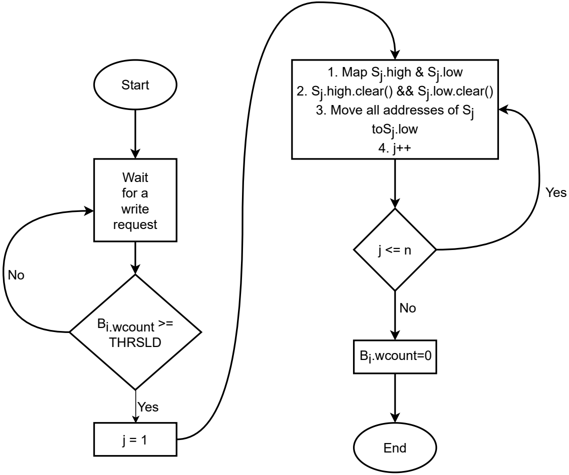
- A **write counter** for each bank is maintained
- Whenever the write count of a bank reaches **THRSLD**, it will go through a complete address mapping round for all of its segments
- For each segment, whatever the high addresses accumulated till that point, mapped to any random low addresses of the same segment
- Once again, all the addresses of that segment will be **considered as low**
- Write count for all the addresses of that segment will be **reset to 0**

Proposed Method: Write variation-aware address remapping

- A **write counter** for each bank is maintained
- Whenever the write count of a bank reaches **THRSLD**, it will go through a complete address mapping round for all of its segments
- For each segment, whatever the high addresses accumulated till that point, mapped to any random low addresses of the same segment
- Once again, all the addresses of that segment will be **considered as low**
- Write count for all the addresses of that segment will be **reset to 0**

Proposed Method: Write variation-aware address remapping

- A **write counter** for each bank is maintained
- Whenever the write count of a bank reaches **THRSLD**, it will go through a complete address mapping round for all of its segments
- For each segment, whatever the high addresses accumulated till that point, mapped to any random low addresses of the same segment
- Once again, all the addresses of that segment will be **considered as low**
- Write count for all the addresses of that segment will be **reset to 0**


Start
Wait for a write request
B_i.wcount >= THRSLD
No
Yes
j = 1
1. Map S_j.high & S_j.low
2. S_j.high.clear() && S_j.low.clear()
3. Move all addresses of S_j toS_j.low
4. j++
j <= n
Yes
No
B_i.wcount=0
End

Proposed Method: Write variation-aware address remapping

Address Remapping

(1,5)

1	5
5	1

Proposed Method: Write variation-aware address remapping

Address Remapping

(1,5)

1	5
5	1

(1,5), (2,4)

2 → 4 ✗

4 → 2 ✗

1	5
5	1

1	5
5	1
2	4
4	2

Swap(M[2], M[4])

Proposed Method: Write variation-aware address remapping

Address Remapping

(1,5)

1	5
5	1

(1,5), (2,4)

2 → 4 ✗

4 → 2 ✗

1	5
5	1

1	5
5	1
2	4
4	2

Swap(M[2], M[4])

(1,5), (2,4), (5,10)

5 → 10 ✓

10 → 5 ✗

1	5
5	1
2	4
4	2

1	~~5~~ 10
5	1
2	4
4	2
10	5

Swap(M[5], M[10])

Proposed Method: Write variation-aware address remapping

Address Remapping

(1,5)

1	5
5	1

(1,5), (2,4)

2 → 4 ✗

4 → 2 ✗

1	5
5	1

1	5
5	1
2	4
4	2

Swap(M[2], M[4])

(1,5), (2,4), (5,10)

5 → 10 ✓

10 → 5 ✗

1	5
5	1
2	4
4	2

1	~~5~~ 10
5	1
2	4
4	2
10	5

Swap(M[5], M[10])

(1,5), (2,4), (5,10), (7,2)

7 → 2 ✗

2 → 7 ✓

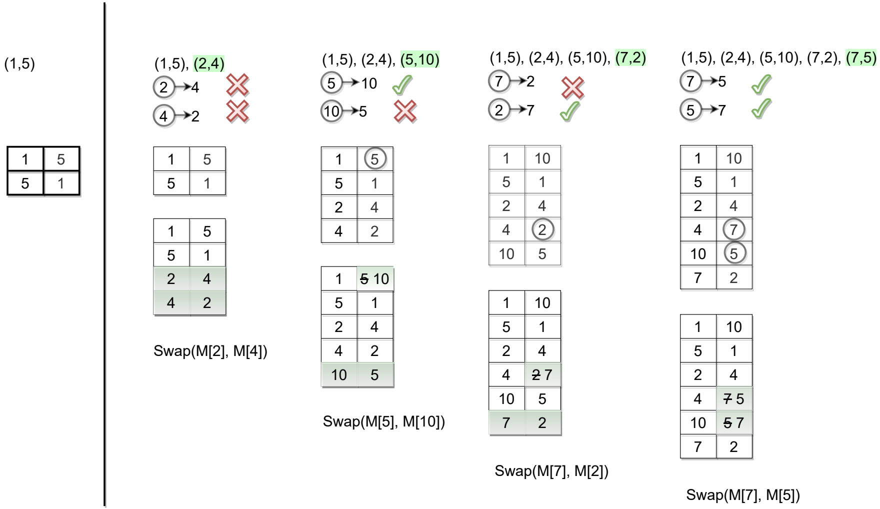
1	10
5	1
2	4
4	2
10	5

1	10
5	1
2	4
4	~~2~~ 7
10	5
7	2

Swap(M[7], M[2])

Proposed Method: Write variation-aware address remapping

Address Remapping



(1,5)

1	5
5	1

(1,5), (2,4)

2 → 4 ✗

4 → 2 ✗

1	5
5	1

1	5
5	1
2	4
4	2

Swap(M[2], M[4])

(1,5), (2,4), (5,10)

5 → 10 ✓

10 → 5 ✗

1	5
5	1
2	4
4	2

1	~~5~~ 10
5	1
2	4
4	2
10	5

Swap(M[5], M[10])

(1,5), (2,4), (5,10), (7,2)

7 → 2 ✗

2 → 7 ✓

1	10
5	1
2	4
4	2
10	5

1	10
5	1
2	4
4	~~2~~ 7
10	5
7	2

Swap(M[7], M[2])

(1,5), (2,4), (5,10), (7,2), (7,5)

7 → 5 ✓

5 → 7 ✓

1	10
5	1
2	4
4	7
10	5
7	2

1	10
5	1
2	4
4	~~7~~ 5
10	~~5~~ 7
7	2

Swap(M[7], M[5])

Proposed Method: Write variation-aware address remapping

Address Remapping

(1,5)

1	5
5	1

(1,5), (2,4)

2 → 4 ✗

4 → 2 ✗

1	5
5	1

1	5
5	1
2	4
4	2

Swap(M[2], M[4])

(1,5), (2,4), (5,10)

5 → 10 ✓

10 → 5 ✗

1	5
5	1
2	4
4	2

1	~~5~~ 10
5	1
2	4
4	2
10	5

Swap(M[5], M[10])

(1,5), (2,4), (5,10), (7,2)

7 → 2 ✗

2 → 7 ✓

1	10
5	1
2	4
4	2
10	5

1	10
5	1
2	4
4	~~2~~ 7
10	5
7	2

Swap(M[7], M[2])

(1,5), (2,4), (5,10), (7,2), (7,5)

7 → 5 ✓

5 → 7 ✓

1	10
5	1
2	4
4	7
10	5
7	2

1	10
5	1
2	4
4	~~7~~ 5
10	~~5~~ 7
7	2

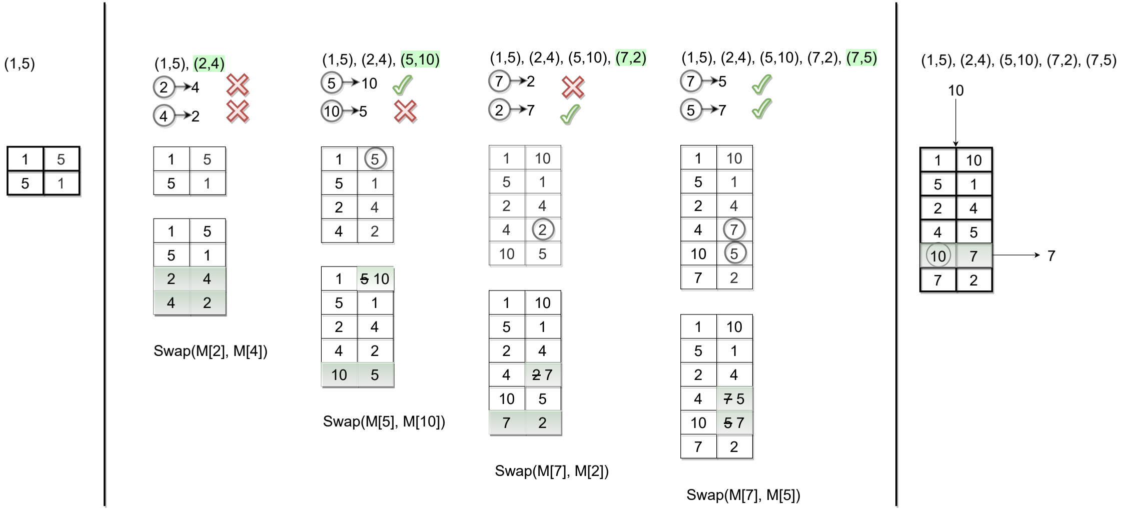
Swap(M[7], M[5])

(1,5), (2,4), (5,10), (7,2), (7,5)

1	10
5	1
2	4
4	5
10	7
7	2

Proposed Method: Write variation-aware address remapping

Address Translation



(1,5)

1	5
5	1

(1,5), (2,4)

2 → 4 ✗

4 → 2 ✗

1	5
5	1

1	5
5	1
2	4
4	2

Swap(M[2], M[4])

(1,5), (2,4), (5,10)

5 → 10 ✓

10 → 5 ✗

1	5
5	1
2	4
4	2

1	~~5~~ 10
5	1
2	4
4	2
10	5

Swap(M[5], M[10])

(1,5), (2,4), (5,10), (7,2)

7 → 2 ✗

2 → 7 ✓

1	10
5	1
2	4
4	2
10	5

1	10
5	1
2	4
4	~~2~~ 7
10	5
7	2

Swap(M[7], M[2])

(1,5), (2,4), (5,10), (7,2), (7,5)

7 → 5 ✓

5 → 7 ✓

1	10
5	1
2	4
4	7
10	5
7	2

1	10
5	1
2	4
4	~~7~~ 5
10	~~5~~ 7
7	2

Swap(M[7], M[5])

(1,5), (2,4), (5,10), (7,2), (7,5)

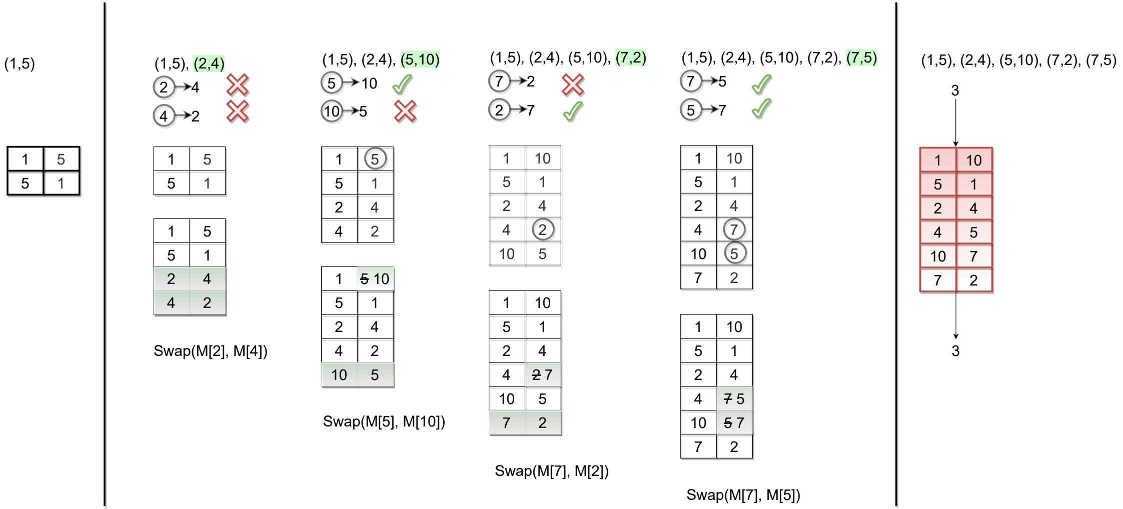
10

1	10
5	1
2	4
4	5
10	7
7	2

→ 7

Proposed Method: Write variation-aware address remapping

Address Translation



(1,5)

1	5
5	1

(1,5), (2,4)

2 → 4 ✗

4 → 2 ✗

1	5
5	1

1	5
5	1
2	4
4	2

Swap(M[2], M[4])

(1,5), (2,4), (5,10)

5 → 10 ✓

10 → 5 ✗

1	5
5	1
2	4
4	2

1	~~5~~ 10
5	1
2	4
4	2
10	5

Swap(M[5], M[10])

(1,5), (2,4), (5,10), (7,2)

7 → 2 ✗

2 → 7 ✓

1	10
5	1
2	4
4	2
10	5

1	10
5	1
2	4
4	~~2~~ 7
10	5
7	2

Swap(M[7], M[2])

(1,5), (2,4), (5,10), (7,2), (7,5)

7 → 5 ✓

5 → 7 ✓

1	10
5	1
2	4
4	7
10	5
7	2

1	10
5	1
2	4
4	~~7~~ 5
10	~~5~~ 7
7	2

Swap(M[7], M[5])

(1,5), (2,4), (5,10), (7,2), (7,5)

3

1	10
5	1
2	4
4	5
10	7
7	2

3

Evaluation

Table: System Parameters

Components	**Parameters**
Processor	ALPHA
L1 Cache	Private, 32 kB SRAM split I/D caches, 2-way associative, 64 B block
L2 Cache	Private, 512 kB SRAM, 64 B block, 8-way associative
Main Memory	PCM: 4 GB, Memory Controller: FRFCFS
Memory Latency	PCM:: Row hit (read miss, write miss) = 40 (120, 150) ns DRAM:: Row hit (miss) = 40 (80) ns

Evaluation

- Gem5 full system simulator + NVMain to generate the memory traces
- In-house simulator for better flexibility
- Benchmarks: SPEC2006

Evaluation

- Gem5 full system simulator + NVMain to generate the memory traces
- In-house simulator for better flexibility
- Benchmarks: SPEC2006

Evaluation

- Gem5 full system simulator + NVMain to generate the memory traces
- In-house simulator for better flexibility
- Benchmarks: SPEC2006

Evaluation

Lifetime Improvement

Formula 1

$$LI = \frac{maximumWriteCount_{base}}{maximumWriteCount_{proposed}}$$

Evaluation

Lifetime Improvement

Formula 1

$$LI = \frac{maximumWriteCount_{base}}{maximumWriteCount_{proposed}}$$

Formula 2

$$LI = \frac{AvgWrite_{base} * (1 + IntraV_{base})}{AvgWrite_{proposed} * (1 + IntraV_{proposed})}$$

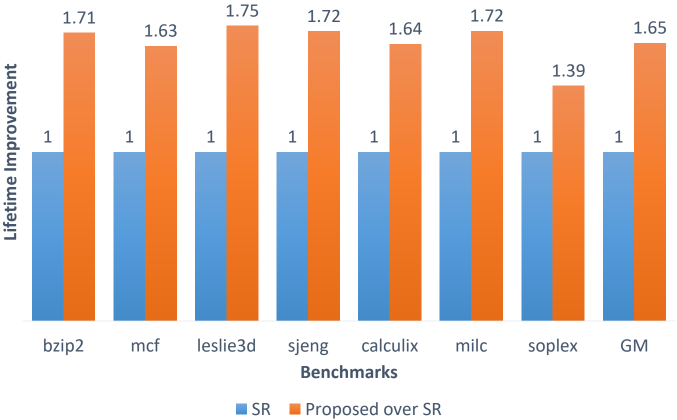
$IntraV$ is the coefficient of variation of writes in a bank

$$IntraV = \frac{\sqrt{\frac{\sum_{i=0}^{N}(AvgWrite - W_i)^2}{N-1}}}{AvgWrite}$$

and, $AvgWrite$ is the average number of writes in a bank

Evaluation

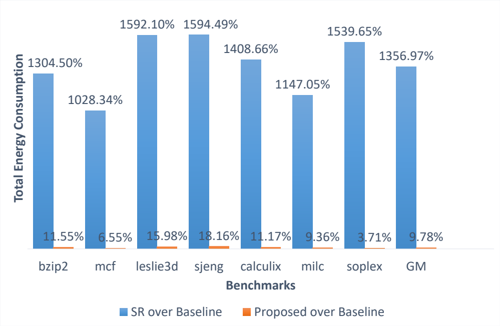
Lifetime Improvement



Evaluation

Total Energy Consumption

- Read, Write: $0.2nJ/bit, 1nJ/bit$
- $\frac{E_{SR}-E_{Base}}{E_{Base}} * 100\%, \frac{E_{Proposed}-E_{Base}}{E_{Base}} * 100\%$



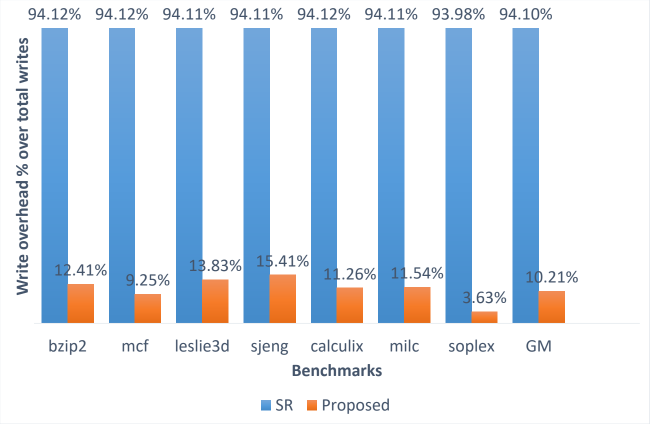
Total Energy Consumption

Table: Total number of reads and writes

	bzip2	mcf	leslie3d	sjeng	calculix	milc	soplex
Baseline	355489	454235	113789	185170	2476284	82421	3967
Proposed	434893	536667	140429	228490	2994624	101621	4439
SR	11725985	14986843	3751037	6099794	81709308	2703861	118655

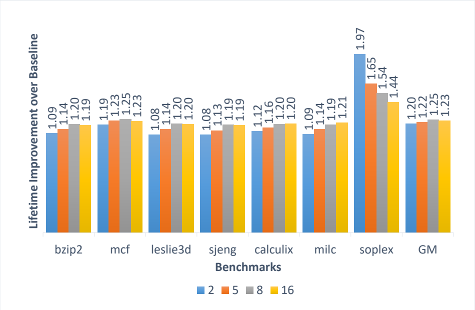
Evaluation

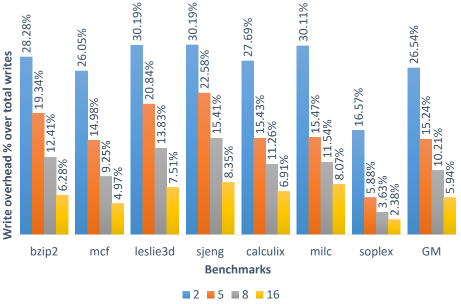
$$\frac{OverheadWrite}{TotalWrite} * 100\%$$



Evaluation

Sensitivity Analysis of *ADDTHRSLD*





- Lifetime improves till 8
- Lower the *ADDTHRSLD*, higher the swap overhead
- We choose **8**

Conclusion & Future Work

- DRAM Drawback
- Introduced Non-volatile memories, PCM: Best candidate to replace DRAM
- State-of-the-art wear leveling algorithms
- Write variation-aware address remapping
  - Based on the write counts of addresses
  - Only between high and low addresses
  - Achieved 1.65 times better lifetime than SR with just 10.21% write overhead
- MLC PCM
  - Increased device capacity
  - Decreased Cell Endurance

Conclusion & Future Work

- DRAM Drawback
- Introduced Non-volatile memories, PCM: Best candidate to replace DRAM
- State-of-the-art wear leveling algorithms
- Write variation-aware address remapping
  - Based on the write counts of addresses
  - Only between high and low addresses
  - Achieved 1.65 times better lifetime than SR with just 10.21% write overhead
- MLC PCM
  - Increased device capacity
  - Decreased Cell Endurance

Conclusion & Future Work

- DRAM Drawback
- Introduced Non-volatile memories, PCM: Best candidate to replace DRAM
- State-of-the-art wear leveling algorithms
- Write variation-aware address remapping
  - Based on the write counts of addresses
  - Only between high and low addresses
  - Achieved 1.65 times better lifetime than SR with just 10.21% write overhead
- MLC PCM
  - Increased device capacity
  - Decreased Cell Endurance

Conclusion & Future Work

- DRAM Drawback
- Introduced Non-volatile memories, PCM: Best candidate to replace DRAM
- State-of-the-art wear leveling algorithms
- Write variation-aware address remapping
  - Based on the write counts of addresses
  - Only between high and low addresses
  - Achieved 1.65 times better lifetime than SR with just 10.21% write overhead
- MLC PCM
  - Increased device capacity
  - Decreased Cell Endurance

Conclusion & Future Work

- DRAM Drawback
- Introduced Non-volatile memories, PCM: Best candidate to replace DRAM
- State-of-the-art wear leveling algorithms
- Write variation-aware address remapping
  - Based on the write counts of addresses
  - Only between high and low addresses
  - Achieved 1.65 times better lifetime than SR with just 10.21% write overhead
- MLC PCM
  - Increased device capacity
  - Decreased Cell Endurance

Conclusion & Future Work

- DRAM Drawback
- Introduced Non-volatile memories, PCM: Best candidate to replace DRAM
- State-of-the-art wear leveling algorithms
- Write variation-aware address remapping
  - Based on the write counts of addresses
  - Only between high and low addresses
  - Achieved 1.65 times better lifetime than SR with just 10.21% write overhead
- MLC PCM
  - Increased device capacity
  - Decreased Cell Endurance

Conclusion & Future Work

- DRAM Drawback
- Introduced Non-volatile memories, PCM: Best candidate to replace DRAM
- State-of-the-art wear leveling algorithms
- Write variation-aware address remapping
  - Based on the write counts of addresses
  - Only between high and low addresses
  - Achieved 1.65 times better lifetime than SR with just 10.21% write overhead
- MLC PCM
  - Increased device capacity
  - Decreased Cell Endurance

Conclusion & Future Work

- DRAM Drawback
- Introduced Non-volatile memories, PCM: Best candidate to replace DRAM
- State-of-the-art wear leveling algorithms
- Write variation-aware address remapping
  - Based on the write counts of addresses
  - Only between high and low addresses
  - Achieved 1.65 times better lifetime than SR with just 10.21% write overhead
- MLC PCM
  - Increased device capacity
  - Decreased Cell Endurance

Conclusion & Future Work

- DRAM Drawback
- Introduced Non-volatile memories, PCM: Best candidate to replace DRAM
- State-of-the-art wear leveling algorithms
- Write variation-aware address remapping
  - Based on the write counts of addresses
  - Only between high and low addresses
  - Achieved 1.65 times better lifetime than SR with just 10.21% write overhead
- MLC PCM
  - Increased device capacity
  - Decreased Cell Endurance

Conclusion & Future Work

- DRAM Drawback
- Introduced Non-volatile memories, PCM: Best candidate to replace DRAM
- State-of-the-art wear leveling algorithms
- Write variation-aware address remapping
  - Based on the write counts of addresses
  - Only between high and low addresses
  - Achieved 1.65 times better lifetime than SR with just 10.21% write overhead
- MLC PCM
  - Increased device capacity
  - Decreased Cell Endurance

Thank You!

Open to **questions** now