IEIE Transactions on Smart Processing and Computing, vol. 9, no. 5, October 2020
https://doi.org/10.5573/IEIESPC.2020.9.5.413

Bit-width Reduction in Write Counters for Wear Leveling in a Phase-change Memory System

Hyokeun Lee[1], Hyunmin Jung[1], Hyuk-Jae Lee[1], and Hyun Kim[2,*]

[1] Inter-university Semiconductor Research Center (ISRC), Department of Electrical and Computer Engineering, Seoul National University / Seoul 08826, Korea
[2] Research Center for Electrical and Information Technology, Department of Electrical and Information Engineering, Seoul National University of Science and Technology / Seoul 01811, Korea

* Corresponding Author: Hyun Kim, hyunkim@seoultech.ac.kr

Received April 22, 2020; Accepted May 21, 2020; Published October 30, 2020

** Regular Paper*

** Extended from a Conference: Preliminary results of this paper were presented at the IEEE International Conference on Electronics, Information, and Communication (ICEIC) 2020. This paper has been accepted by the editorial board through the regular reviewing process that confirms the original contribution.*

***Abstract*:** Phase-change memory (PCM) has garnered attention as a next-generation memory owing to its non-volatility and scalability. However, PCM wears out under excessive write accesses; hence, it must be supported by wear-leveling algorithms to uniformly distribute the number of accesses across the entire address space. Table-based wear leveling is one of the representative algorithms that stores a write counter for each address region for remapping frequently accessed addresses with lower overhead; however, write counters consume resources in a PCM system. In this study, a bit-width reduction method in write counters for wear leveling is proposed, where the method utilizes a stochastic finite-state machine to probabilistically count the number of write accesses. The proposed method shows only a 1.2% lifetime degradation using six bits for each counter, with 40% fewer resources spent on write counters when the endurance of a 4KB block is 1E+06.

***Keywords*:** Non-volatile memory, Phase-change memory, Wear leveling, Finite-state machine

1. Introduction

As technological processes scale down to nanometers, non-volatile memory devices such as phase-change memory (PCM) are expected to be deployed in the commercial market. Within the modern computer architecture, PCM has gained attention as a promising next-generation memory device owing to its non-volatility, scalability, low power consumption, and low latency [1-3]. These impressive characteristics allow PCM to be placed as storage-class memory between the main dynamic random-access memory (DRAM) and NAND flash-based storage [4-7], because PCM is slower than DRAM but faster than NAND flash. Accordingly, PCM is expected to provide power reduction and performance improvement in applications such as deep neural networks that intensively access memory in conjunction with the parallelization of graphics processing units (GPUs) [8, 9]. It should be noted that reducing the power consumption of main memory is very important in modern computer architectures [10].

However, PCM is not being deployed due to the cell endurance problem. PCM cell programming requires repetitive joule heating to change the resistance of the cell material [11-13]; hence, excessive write operations on a PCM cell incur stuck-at fault errors. Unlike hard-disk drive (HDD) storage or DRAM, the lifetime of a PCM-based main memory device is primarily limited by the cell or the page that is the worst damaged. Consequently, a wear-leveling algorithm, which remaps frequently updated data to infrequently written cells, must be embedded in the PCM controller to uniformly distribute the number of write operations to each cell, and thereby enhance the device lifetime [14-18].

Table-based wear leveling (TWL) manages address mapping information in a table-based data structure [19, 20]. TWL has to store a write counter for each address block to trigger address remapping with lower performance overhead by mapping mostly hot data to cold

physical addresses. Specifically, the write counters take up more memory resources with the technological scaling of PCM. A 512GB PCM device requires 8B×512GB/4KB =1GB in write counters if the memory region is divided into 4KB units. Therefore, it is crucial to reduce the resource overhead of the write counters. In [21], Yun et al. declared that write counters can be roughly maintained for wear leveling instead of keeping a precise number in each write counter. Since spatial locality exists in mutually adjacent logical addresses, previous work maintained a lot of write counter information in a single counter by adopting Bloom filters. However, the bit width of write counters increases as the number of maintained addresses grows; hence, reducing the bit width of each counter is also critical for reducing the overall overhead incurred by write counters.

On the other hand, algebraic mapping–based wear leveling (AWL) performs an algebraic operation on the logical address to get the physical address in PCM; hence, the mapping table is not required. However, the write counter of each address is still required as metadata for further management by system software, such as operating systems [19]. In summary, reducing the resource overhead incurred by write counters is crucial for both TWL and AWL.

This paper proposes a bit-width reduction method of write counters for wear leveling in a PCM-based system. The proposed method manages each counter as a stochastic finite-state machine (FSM), which has a shorter bit width than the baseline, to probabilistically count the number of write commands in each address region. The proposed work increases the counter number by tossing an *unfair coin* instead of increasing the number for each access. Thus, a shorter bit width can be maintained within a single counter. Our experiment results indicate that the proposed method reduces the bit width from 10-bit to 6-bit when the endurance of a 4KB address region is 1E+06. Nevertheless, the proposed method shows only 1.2% degradation in memory lifetime.

The remainder of this paper is organized as follows. Section 2 explains the background, including the system architecture and the existing algorithms used in this study. In Section 3, the proposed method is described. Subsequently, the evaluation results are presented in Section 4. Finally, the last section concludes the paper.

2. Background

2.1 System Layout

Fig. 1 depicts the memory controller architecture adopted in this paper. As shown in the figure, input commands are stored in the command buffer, waiting for command scheduling. The write counters are stored in DRAM cache, where the counter data are loaded into the write counter register if the scheduled write command is ready to be processed. Hot address detector table-based wear leveling (HAD-TWL) is not triggered for every write command in order to prevent significant performance degradation. Instead, HAD-TWL launches once it receives

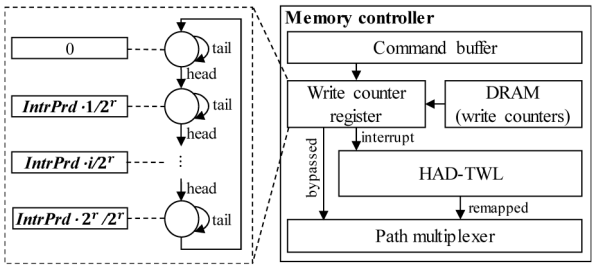

Fig. 1. System layout, right, and the proposed method to reduce write counters, left.

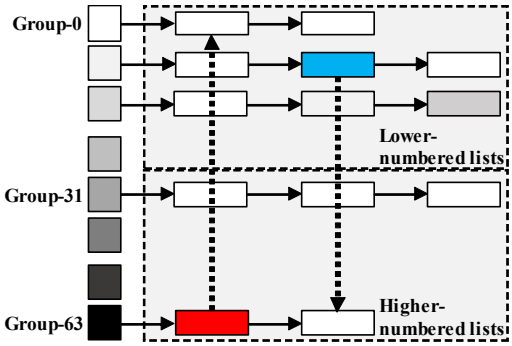

Fig. 2. Block diagram of the Rejuvenator wear-leveling algorithm, which has 64 write count groups.

an interrupt signal from the write counter register when the number reaches a predefined threshold value (e.g., 1024). The hot address detector determines whether the processed command is hot (i.e., frequently accessed) or not, and signals wear leveling if it is hot. Consequently, wear leveling is performed, and the hot data are remapped to a cold physical address.

2.2 Hot Address Detector

The TWL algorithm fundamentally remaps a logically hot address to a physically cold address by leveraging write counter information. However, it induces significant performance overhead if the address space range is considerably large, because searching a target requires linear search among a number of counters. Therefore, a hot address detector is required in order to reduce such overhead. Qureshi et al. proposed a practical address detector to manage frequently accessed write addresses within a tiny table, instead of searching the counter [14]. Kim et al. proposed HAD-TWL to perform TWL with the support of a variable-sized table [19]. In this study, wear leveling is assumed to be HAD-TWL, as shown in Fig. 1.

2.3 Rejuvenator

As shown in Fig. 2, TWL in this paper, namely Rejuvenator, is a wear-leveling algorithm designed for NAND flash memory [22]. The algorithm maintains a fixed number of linked lists as a moving window, in which each group of lists roughly corresponds to one erase count group. Each node in the list maintains a physical block address number; hence, it can be considered a *memory*

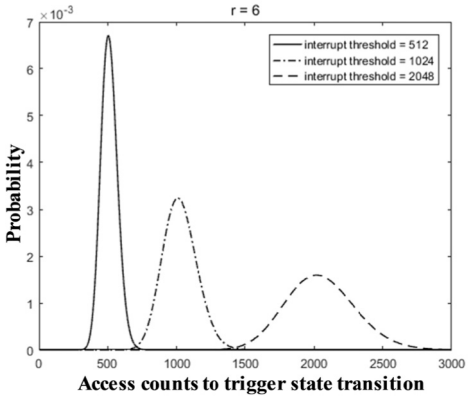

Fig. 3. Distribution of access counts triggering a state transition with regard to different interrupt threshold values when *r* = 6.

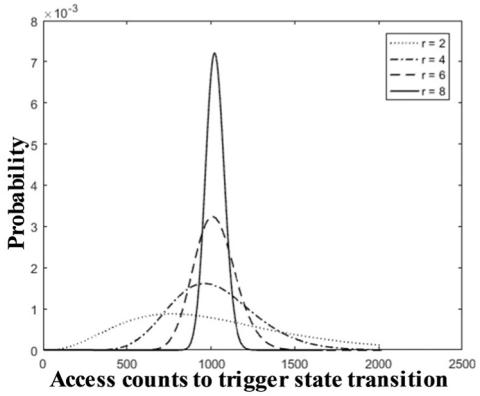

Fig. 4. Distribution of access counts triggering state transition with regard to different bit widths in the proposed method when *n* = 10.

pool. The objective of this algorithm is to map a logically hot address and a logically cold address on the lower-numbered list and higher-numbered list, respectively. Consequently, all erase count groups slide as remapping is performed. Since the memory tested in this paper is PCM, we use the term *write count* rather than *erase count*. Furthermore, the linked lists in Rejuvenator are assumed to be stored in DRAM cache, as mentioned in Section 2.1.

3. Proposed Method

3.1 A Stochastic FSM in Write Counters

In this study, a method is proposed to significantly reduce the bit width of a write counter by modeling each write counter as a stochastic FSM, since a precise write counter is not required. As shown in the dashed box on the left side of Fig. 1, the write counter acts as a cyclic FSM, where the meaning of each state corresponds to the number of write accesses, as depicted in the rectangular box. Besides, there are two parameters in the proposed method. They are:

- ***IntrPrd***: the period value that incurs the HAD-TWL interrupt, which is the pre-defined threshold value used by the normal write counter.
- ***r***: shows the bit width of the stochastic FSM that directly determines the precision of the counter. Since 2^{r} indicates the maximum value that stochastic FSM can physically represent, it determines the number of FSM states.

As mentioned above, the number stored in the counter can be represented as $\boldsymbol{IntrPrd}*i/2^{r}$, where the currently stored number, namely i in Fig. 1, represents the state number ranging from 0 to $(2^{r}-1)$. Therefore, each state means that an address is written approximately $\boldsymbol{IntrPrd}*i/2^{r}$ times. Even though the formula has a division operation, the approximate write count can be obtained with a mere shift operation, because the operation is a power of 2. After continuously transferring states, the FSM interrupts HAD-TWL when the final state is achieved, and the state reverts to the initial state. On the other hand, the state transition model is described in the next subsection.

3.2 A Probabilistic State Transition Model

The transition between two states is performed probabilistically. In this study, the transition is considered a *coin toss* model. The transition occurs if the coin is flipped and comes up *heads*. According to the predefined parameters above, the state transition probability is defined in Eq. (1):

$$P(head) \equiv \frac{2^{r}}{\boldsymbol{IntrPrd}} \tag{1}$$

Here, $P(head)$ means the probability that heads come up during the coin toss. Therefore, the counter remains in the same state, probabilistically, for approximately $(\boldsymbol{IntrPrd}/2^{r}-1)$ times, and transits right after that moment.

Bit width ***r*** in the state transition model determines not only the overall resource overhead of the write counters but also the lifetime and the reliability of the overall system. Thus, discussing the confidence of the proposed method is crucial to obtaining a reliable PCM-based memory system. If it is assumed that ***n*** is the original bit width of the write counter, ***IntrPrd*** can be defined as 2^{n}. Thus, the probabilistic formula (1) becomes $P(head)=1/2^{n-r}$. Furthermore, since the access number of an address triggering the state transition (i.e., the number of trials that turn up heads) follows a negative binomial distribution [23], the probability density function of the access number and its true mean, respectively, can be modeled as seen in Eqs. (2) and (3). Moreover, if Eq. (2) is substituted into (3), the true mean becomes 2^{n}.

$$P(access) = \binom{access-1}{access-2^{r}} P(head)^{2^{r}} \left(1-P(head)\right)^{access-2^{r}} \tag{2}$$

$$\text{true mean} = \frac{2^{r}\left(1-P(access)\right)}{P(access)} + 2^{r} \tag{3}$$

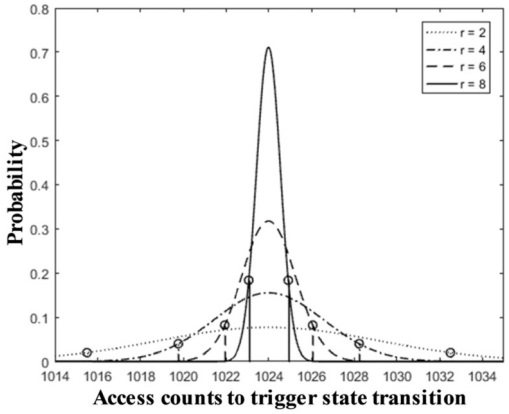

Fig. 5. Distribution of the sample mean with regard to different bit widths (*r*) when *n* = 10, where the vertical line in the figure shows the 90% confidence interval of each graph.

Fig. 3 shows the distribution of Eq. (2) when ***IntrPrd***= 512, 1024, and 2048 when ***r***=6. As shown in the figure, a smaller interrupt threshold (i.e., ***IntrPrd***) provides better reliability in the memory system, because ***n*** is closer to ***r*** as ***n*** becomes smaller. Therefore, a moderate value for ***IntrPrd*** of 1024 was selected in this study. On the other hand, Fig. 4 shows the distribution of (2) when ***r*** varies from 2 to 8, and ***n*** is fixed as 10. As shown in the figure, a larger ***r*** provides a more reliable system; however, greater resource overhead accompanies the enhanced reliability.

3.3 Confidence of the Proposed Method

Fig. 4 shows that the variance is significant if considering a single interrupt as ***r*** increases. However, there would be a number of interrupts in HAD-TWL during the execution of an application in the system. A late interrupt causes late data remapping, which may degrade memory lifetime. Conversely, an early interrupt causes early data remapping, which may prevent the degradation in lifetime. In other words, these two effects compensate for each other; hence, it is necessary to discuss the 90% confidence interval of the ***average*** (i.e., true mean) access number that triggers state transition. According to the central limit theorem [24], the 90% confidence interval of the average access number can be expressed in Eq. (4), where $\overline{access}$ is the sample mean of $P(access)$, s is the sample variance, and ρ is the sample number, which is defined in Eq. (5):

$$P\left(\overline{access}-1.645\frac{s}{\sqrt{\rho}}\le 2^n\le \overline{access}+1.645\frac{s}{\sqrt{\rho}}\right)=90\% \tag{4}$$

$$\rho\approx\frac{total\ write\ number}{2^n} \tag{5}$$

Eq. (4) explains that there is a 90% probability that the sample-mean value (i.e., 2^n in the equation) will lie on the interval $\left[\overline{access}-1.645\frac{s}{\sqrt{\rho}},\overline{access}+1.645\frac{s}{\sqrt{\rho}}\right]$. Fig. 5 shows the 90% confidence interval when ***n*** =10. As shown in the figure, the confidence interval becomes narrower as the value of ***r*** reaches ***n***. Furthermore, it shows that the system reliability is high enough even for a small bit width for the write counter. The experimental verification of this model is described in the next section.

Table 1. System configuration.

Configuration	Description
HAD	Initially 128 entries, frequency-based entry eviction policy
TWL (Rejuvenator)	Number of list groups: 16
PCM	16GB, SLC, 4KB block, Endurance: 1E+05, 1E+06
Write counter	Interrupt HAD for every 1024 accesses. Allocated for 4KB block address.
Workloads	31 application traces from SNIA Nexus 5, each application is scheduled by the round robin scheduler.

Table 2. Theoretical values and experiment results of the 90% confidence interval.

R	Theoretical values	Experimental results
2	1.6418	1.5944
4	0.8160	0.8105
6	0.3982	0.3974
8	0.1781	0.1779

4. Evaluations

4.1 Configurations

As shown in Table 1, an in-house simulation was conducted where the system architecture was designed as seen in Fig. 1. In particular, PCM was configured as a 16GB single-level cell (SLC) array. Endurances of a 4KB block were selected at 1E+05 and 1E+06 for evaluation. Each write counter was allocated a 4KB block address. Furthermore, 31 workload traces of SNIA Nexus 5 were executed in the simulation environment, where a round-robin scheduling policy was used for executing these 31 workloads in a single system to simulate a practical server system.

4.2 Verification of Modeling

Fig. 6 illustrates the distribution of an access number triggering state transition from a real workload, which shows that Eq. (2) is well modeled as a mathematical form, compared with Fig. 3. Moreover, Table 2 shows a theoretical 90% confidence interval for the true mean, and the experiment results obtained 90% confidence in the sample mean. For all values of ***r*** shown in the tables, the 90% confidence intervals from the experiment are similar

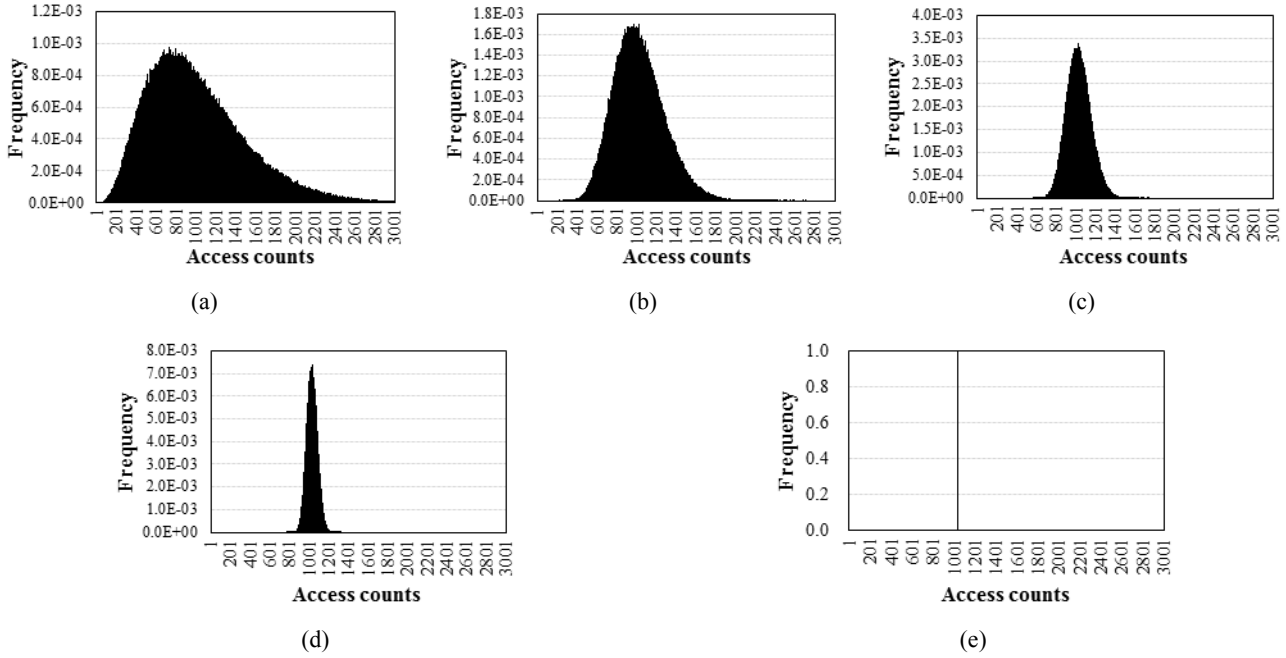

Fig. 6. Experimentally obtained access counts triggering a state transition with regard to different bit widths in the proposed method (a) r = 2, (b) r = 4, (c) r = 6, (d) r = 8, and (e) r = n = 10.

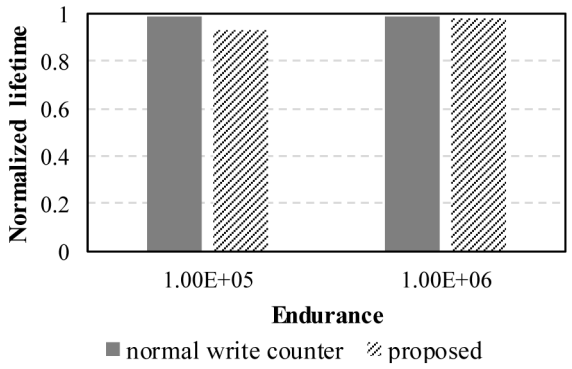

Fig. 7. Normalized lifetime with respect to different endurances.

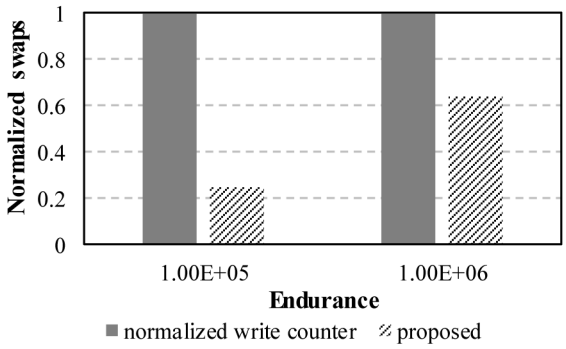

Fig. 8. Normalized swap operations with respect to different endurances.

to the theoretical values. For r=6, the confidence interval is 0.3974, whereas the theoretical value is 0.3982, which shows a 2% error. Therefore, this paper assumes r=6 in the following section, since it provides a negligible error.

4.3 Lifetime and Swap Overhead

Normalized lifetime is defined as the ratio of the total write counts just before system failure to the maximum total write counts if all blocks are written uniformly in an ideal case. Fig. 7 shows the normalized lifetime of the system when the 31 server workloads are executed concurrently, and when the bit width of the proposed write counter is fixed at r=6. When the block endurance is 1E+05, the normalized lifetime is degraded by 5.7%, compared with the baseline write counter (i.e., n=10) from Section 3.2. For a block endurance of 1E+06, the normalized lifetime degradation is as small as 1.2%.

On the other hand, Fig. 8 shows the normalized swap overhead when the proposed method is adopted. The normalized swap overhead is defined as the ratio of the number of swaps adopting the proposed method to the number of swaps adopting the baseline write counter. As shown in the figure, the number of swaps is reduced by 75.1% and 36.4% when the endurance values are 1E+05 and 1E+06, respectively. The figure shows that HAD-TWL is triggered late in most cases, and thereby, the lifetime is degraded (see Fig. 7). However, a 1.2% lifetime degradation can be negligible, considering that the resource overhead of the write counters is reduced by 40%.

5. Conclusion

In this paper, a stochastic FSM for reducing the bit width of write counters in a PCM-based memory system is proposed and modeled with mathematical equations to

verify the system reliability. The proposed method shows a 40% reduction in memory resources consumed by write counters, along with only 5.7% and 1.2% lifetime degradation when the endurance values are 1E+05 and 1E+06, respectively. Since the proposed method is verified in an environment simulating more than 31 workloads concurrently, it is expected to be adopted not only for PCM-based memory systems but also for NAND flash storage systems.

Acknowledgement

This paper was supported in part by the Technology Innovation Program (10080613, DRAM/PRAM heterogeneous memory architecture and controller IC design technology research and development) funded by the Ministry of Trade, Industry & Energy (MOTIE), Korea, and in part by a National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2019R1F1A1057530).

References

[1] E. Kultursay, M. Kandemir, A. Sivasubramaniam, and O. Mutlu, "Evaluating stt-ram as an energy-efficient main memory alternative," in IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), 2013. Article (CrossRef Link)

[2] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger, "Architecting phase change memory as a scalable dram alternative," in Proceedings of the 36th Annual International Symposium on Computer Architecture (ISCA), 2009. Article (CrossRef Link)

[3] S. Mittal, "A survey of soft-error mitigation techniques for non-volatile memories," Computers, vol. 6, no. 1, 2017. Article (CrossRef Link)

[4] Intel. (2015) Intel and micron produce breakthrough memory technology. [Online]. Available: https://newsroom.intel.com/news-releases/intel-andmicron-produce-breakthrough-memory-technology

[5] S. Sundararaman, N. Talagala, D. Das, A. Mudrankit, and D. Arteaga, "Towards software defined persistent memory: Rethinking software support for heterogeneous memory architectures," in Proceedings of the 3rd Workshop on Interactions of NVM/FLASH with Operating Systems and Workloads (INFLOW), 2015, pp. 6:1–6:10. Article (CrossRef Link)

[6] S. R. Dulloor, S. Kumar, A. Keshavamurthy, P. Lantz, D. Reddy, R. Sankaran, and J. Jackson, "System software for persistent memory," in Proceedings of the 9th European Conference on Computer Systems (EuroSys), 2014, pp. 15:1–15:15. Article (CrossRef Link)

[7] B. Bhattacharjee, M. Canim, C. A. Lang, G. A. Mihaila, and K. A. Ross, "Storage class memory aware data management," Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, vol. 33, no. 4, pp. 35–40, 2010. Article (CrossRef Link)

[8] B. Kim et al., "PCM: Precision-Controlled Memory System for Energy Efficient Deep Neural Network Training," 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE), 2020. Article (CrossRef Link)

[9] C. Lee and H. Lee, "Effective Parallelization of a High-Order Graph Matching Algorithm for GPU Execution," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 2, pp. 560-571, Feb. 2019. Article (CrossRef Link)

[10] M. Kim, I. Chang and H. Lee, "Segmented Tag Cache: A Novel Cache Organization for Reducing Dynamic Read Energy," in *IEEE Transactions on Computers*, vol. 68, no. 10, pp. 1546-1552, 2019. Article (CrossRef Link)

[11] S. Chen, P. B. Gibbons, and S. Nath, "Rethinking database algorithms for phase change memory," 04 2011, pp. 21–31. Article (CrossRef Link)

[12] H. Lee, M. Kim, H. Kim, H. Kim, and H. Lee, "Integration and boost of a read-modify-write module in phase change memory system", IEEE Transactions on Computers, pp. 1772–1784, vol. 68, no. 12, 2019. Article (CrossRef Link)

[13] P. Zhou, J. Y. B. Zhao, and Y. Zhang, "Throughput enhancement for phase change memories," IEEE Transactions on Computers, vol. 63, pp. 2080–2093, 2014. Article (CrossRef Link)

[14] M. K. Qureshi, J. Karidis, M. Franceschini, V. Srinivasan, L. Lastras, and B. Abali, "Enhancing lifetime and security of pcm-based main memory with start-gap wear leveling," in Proceedings of the 42nd Annual International Symposium on Microarchitecture (MICRO), 2009. Article (CrossRef Link)

[15] N. H. Seong, D. H. Woo, and H. H. Lee, "Security refresh: Protecting phase-change memory against malicious wear out," IEEE Micro, vol. 31, no. 1, pp. 119–127, 2011. Article (CrossRef Link)

[16] F. Huang, D. Feng,W. Xia,W. Zhou, Y. Zhang, M. Fu, C. Jiang, and Y. Zhou, "Security rbsg: Protecting phase change memory with security-level adjustable dynamic mapping," in Proceedings of the 30th IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2016, pp. 1081–1090. Article (CrossRef Link)

[17] H. Yu and Y. Du, "Increasing endurance and security of phase change memory with multi-way wear-leveling," IEEE Transactions on Computers, vol. 63, no. 5, pp. 1157–1168, 2014. Article (CrossRef Link)

[18] M. Kim, J. Choi, H. Kim and H. Lee, "An Effective DRAM Address Remapping for Mitigating Rowhammer Errors," in *IEEE Transactions on Computers*, vol. 68, no. 10, pp. 1428-1441, 1 Oct. 2019. Article (CrossRef Link)

[19] S. Kim, H. Jung, W. Shin, H. Lee and H. Lee, "HAD-TWL: Hot Address Detection-Based Wear Leveling for Phase-Change Memory Systems with Low Latency," in *IEEE Computer Architecture Letters*,

2019. Article (CrossRef Link)
[20] Y. Chang, P. Hsiu, Y. Chang, C. Chen, T. Kuo, and C. M. Wang, "Improving PCM Endurance with a Constant-Cost Wear Leveling Design", ACM Transactions on Design Automation of Electronic Systems, vol. 22, no. 1, 2016. Article (CrossRef Link)
[21] J. Yun, S. Lee, and S. Yoo, "Dynamic wear leveling for phase change memories with endurance variations," IEEE Transactions on Very Large-Scale Integration (VLSI) Systems, vol. 23, no. 9, pp. 1604–1615, 2015. Article (CrossRef Link)
[22] M. Murugan and D. H. C. Du, "Rejuvenator: A static wear leveling algorithm for NAND flash memory with minimized overhead," 2011 IEEE 27th Symposium on Mass Storage Systems and Technologies (MSST), Denver, CO, 2011, pp. 1-12. Article (CrossRef Link)
[23] Negative binomial distribution, April, 2020, Article (CrossRef Link)
[24] Central limit theorem, April, 2020, Article (CrossRef Link)

Hyokeun Lee received a BSc in electrical and computer engineering from Seoul National University, Seoul, South Korea, in 2016, where he is currently working toward an integrated MSc and PhD in electrical and computer engineering. His current research interests include non-volatile memory controller design, hardware persistent models for non-volatile memory, and computer architectures.

Hyunmin Jung received a BSc in electric engineering from Kyung Hee University, Yongin, South Korea, in 2014, and an MSc in electrical and computer engineering from Seoul National University, Seoul, in 2016, where he is currently working toward a PhD in electrical and computer engineering. His research interests include immersive media, virtual reality, augmented reality, and light field processing.

Hyuk-Jae Lee received a BSc and an MSc in electronics engineering from Seoul National University, Seoul, South Korea, in 1987 and 1989, respectively, and received a PhD in electrical and computer engineering from Purdue University, West Lafayette, IN, USA, in 1996. From 1998 to 2001, he was a Senior Component Design Engineer in the Server and Workstation Chipset Division of Intel Corporation, Hillsboro, OR, USA. From 1996 to 1998, he was a Faculty Member in the Department of Computer Science, Louisiana Tech University, Ruston, LA, USA. In 2001, he joined the School of Electrical Engineering and Computer Science, Seoul National University, where he is a Professor. He is the Founder of Mamurian Design, Inc., Seoul, a fabless SoC design house for multimedia applications. His current research interests include computer architectures and SoCs for multimedia applications.

Hyun Kim received a BSc, an MSc, and a PhD in electrical engineering and computer science from Seoul National University, Seoul, South Korea, in 2009, 2011, and 2015, respectively. From 2015 to 2018, he was a BK Assistant Professor for BK21 Creative Research Engineer Development for IT, Seoul National University. In 2018, he joined the Department of Electrical and Information Engineering, Seoul National University of Science and Technology, Seoul, where he is an Assistant Professor. His current research interests include algorithm, computer architecture, memory, and SoC design for low-complexity multimedia applications and deep neural networks.

Copyrights © 2020 The Institute of Electronics and Information Engineers