JOURNAL OF SEMICONDUCTOR TECHNOLOGY AND SCIENCE, VOL. 22, NO. 4, AUGUST, 2022
https://doi.org/10.5573/JSTS.2022.22.4.216

ISSN(Print) 1598-1657
ISSN(Online) 2233-4866

A Read Disturbance Tolerant Phase Change Memory System for CNN Inference Workloads

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

Abstract—**Phase-change memory (PCM) garners attention as the most promising nonvolatile memory (NVM). In particular, PCM is suitable for applications that are not memory intensive, and the convolutional neural network (CNN) inference is widely known as a representative computation-intensive model. Therefore, CNN inference seems to be very suitable for a PCM-based system. However, the PCM suffers from the characteristic of being vulnerable to disturbance errors. In particular, read disturbance error (RDE) becomes a serious problem for workloads involving a large number of zeros, and unfortunately, matrices in CNN are sparse, which inevitably incurs a significant amount of RDEs. In this paper, we present an RDE-tolerant PCM-based system for CNN inference workloads. The proposed method restores vulnerable data words by leveraging a dedicated SRAM-based table. Furthermore, we also propose a replacement policy, which detects non-urgent entries, by utilizing the contents (i.e., counters) in the table. As a result, the proposed method significantly reduces RDEs with minor speed degradation.**

Index Terms—**Phase-change memory, read disturbance error, CNN inference, non-volatile memory, reliability**

Manuscript received Mar. 28, 2022; reviewed Jul. 6, 2022; accepted Jul. 26, 2022

[1]Inter-university Semiconductor Research Center, Department of Electrical and Computer Engineering, Seoul National University, Seoul 00826, 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
E-mail : hyunkim@seoultech.ac.kr
Corresponding Author : Hyun Kim

I. Introduction

Owing to its low latency and standby power, phase-change memory (PCM) attracts attention as the next-generation main memory or storage-class memory (SCM) [4, 17, 20]. To become a pioneer in the market, Intel has manufactured PCM-based memory products (e.g., Optane DC PMM/SSD) [2, 7]. However, since the PCM is a picky device due to its high dynamic write energy, researchers try to exhibit its proper applications by analyzing the system performance in many aspects (e.g., energy and latency) [5, 14]. Fortunately, the standby power of PCM devices is much lower than that of DRAM. Since a recent study shows that convolutional neural network (CNN) inference tasks are computationally intensive [6], the CNN inference appears to be a well-suited application for PCM-based systems.

Despite its intriguing characteristics, the PCM shows lower reliability compared to DRAM. In particular, read disturbance error (RDE) is one of the major obstacles for using the PCM as a main memory. An RDE is incurred by frequent read operations, because the higher read current is applied to achieve the lower latency [9, 10]. The current leads to "silent" programming of the cell, shifting the cell state to the low-resistance state [13]. Thus, vulnerable cells suffer from 0-to-1 bit flip errors. Memory traffic characteristics of CNN inference also contribute to RDEs. Fig. 1(a) shows the breakdown analysis of stored bits in the memory, as various CNN inference tasks run on the PCM-based system. In general, the number of RESET cells accounts for 78.6% of all read cells. Fig. 1(b) shows that the access sparsity, which is the ratio of distinct addresses to the totally accessed

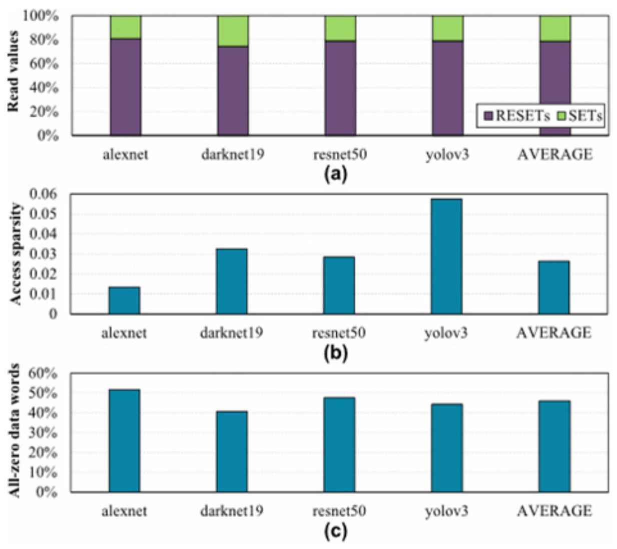

Fig. 1. Memory traffic characteristics regarding various CNN inference models: (a) breakdown of read cells; (b) access sparsity; (c) ratios of all-zero data words.

addresses, is as low as 2.6% for all inference tasks (i.e., high spatial locality). Therefore, the CNN inference inevitably magnifies RDEs in the PCM, requiring an RDE-robust PCM-based system for CNN inference. It should be noted that there are two reasons to map RESET states to the logical 0 and SET states to the logical 1 in this study. First, previous studies generally apply this mapping assumption. Second, the RESET operation has shorter latency than the SET operation. Because CNN inference is 0-domiant, as shown in Fig. 1(a), higher system performance is expected by mapping the logical 0 to the RESET state. To ensure the justification of these reasons, Fig. 1(c) presents the ratios of all-zero words for various networks, where the word size is 32-bit. The experimental results show that the ratio of all-zero data words is 46.03% on average. Moreover, there is no all-one data words in these workloads. Therefore, the ratio of all-zero words generally determines the system performance because programming an all-zero word requires shorter RESET latency. These results denote that mapping RESET states to logical 0 and SET states to logical 1 is more beneficial compared to the opposite case regarding the system performance for CNN inference.

Various techniques, such as error-correcting code (ECC) and scrubbing, can be applied to mitigate RDEs. ECC typically uses Bose-Chaudhuri-Hocquenghem (BCH) codes for multi-bit error correction [16]. Scrubbing is a method that periodically reads and corrects errors; this method often adopts the simple ECC with a short scrubbing period. To reduce the scrubbing overhead, a probabilistic row scrubbing method has been proposed to scrub a row of data in the PCM with a certain probability [10]. However, such approaches have two limitations. First, the PCM has a much higher bit error rate than that of DRAM [10]. ECC schemes, such as BCH or Reed-Solomon code, are necessary; however, such multi-bit ECC schemes incur considerable high encoding/decoding latency and hardware area. Second, the scrubbing may mitigate errors with the low-cost ECC by merely dispatching a read for verification and a write for correction; however, it requires a shorter scrubbing period for reducing errors in PCM devices.

To implement an RDE-tolerant PCM-based system for CNN inference, this study proposes a read disturbance scrubbing assistance (RSA) that utilizes the characteristic of the realistic RDE model and restores vulnerable cells on demand; it is the first approach that mitigates PCM RDEs with a table-based approach. In the realistic RDE model, RDEs occur when the number of read operations on a cell exceeds the specific manufacture number. Within a dedicated small-sized table, the proposed method counts the number of read operations (i.e., read counter) on a data word if bit-0 already exists in that word. Apart from the read counter, the existence of bit-0 is monitored by using another counter (i.e., zero counter). These two counters imply whether the data word is vulnerable to RDEs or not. Thus, it is possible to select the non-urgent entries as eviction candidates in the small-sized table. As a result, our proposed method yields lower error rates compared to the baseline with negligible performance degradation.

The remainder of this paper is organized as follows. In Section 2, the RDE model is explained. In Section 3, the proposed method is presented. In Section 4, experimental results are shown. Finally, Section 5 concludes the paper.

II. Read Disturbance in PCM

RDE is one of the common reliability problems in modern PCM devices. In general, RDE is incurred by frequent and strong reference current of read operations. In particular, the relatively strong reference current, which is used to reduce the read latency [10], gradually shifts the cell resistance over time. The current intensity is much lower than that of write operation; hence, the cell state finally becomes the crystalline state (i.e., 0-to-1 bit flip). The low intensity of the read current may also incur RDEs. A prior study [13] shows that a PCM is vulnerable to resistance drops. That is, an amorphous cell gradually shifts to the crystalline state due to the heat incurred by the read reference current. As a result, 0-to-1 bit-flip errors occur when the resistance of cells trespasses the reference resistance value, as the number of read operations exceeds an *RDE limitation number (RLN)*. Moreover, the idle time between consecutive read operations does not affect the shift phenomenon; the study in [13] explains that a cell can be disturbed in different time frames, even though each frame consists of only one current pulse. Therefore, an RDE would eventually occur regardless of the current intensity (i.e., read latency).

It should be noted that the RLN is a manufacturing dependent number used by manufacturers (e.g., SK Hynix), and there is no exact publicized value; hence, we assume the RLN of 1K in this paper. The number can be referred to as the worst-case number among numerous PCM device samples. Please note that our proposed method is vendor-friendly and is applicable to various RLNs, because the RLN is determined by PCM vendors.

III. RSA: READ DISTURBANCE SCRUBBING ASSISTANCE

1. Architectural Overview

Fig. 2 presents the overall architecture of our system. In conjunction with the multithreading library (e.g., Pthread [11]), CNN inference tasks run on the host processor. The NVM command is issued from the integrated memory controller in the host processor. In the PCM module, the media controller translates the host physical address to the device physical address. Then, the controller schedules NVM commands and generates microcommands (e.g., activation) from one NVM command. The proposed module, RSA, resides between the media controller and PCM media devices.

RSA basically mitigates RDEs by managing read access information in a dedicated SRAM-based table. After a read microcommand goes into the RSA, a new

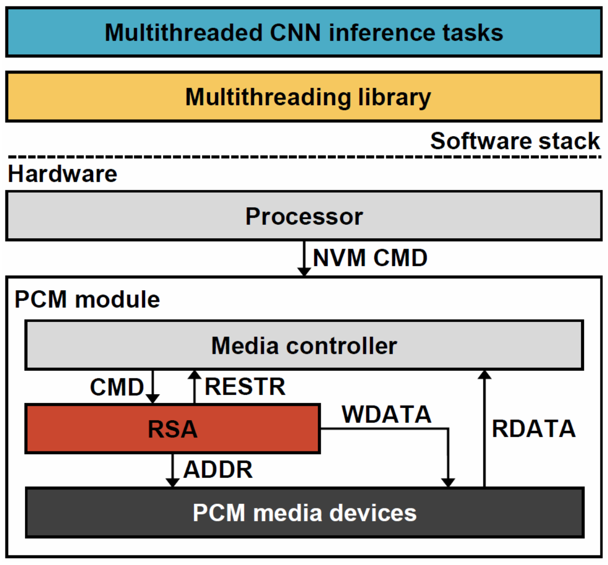

Fig. 2. Architectural overview of the proposed system.

table entry is allocated for that command. The read command updates the number of read operations on the accessed entry. Once the number of read operations exceeds the RLN (see its definition in Section 2), RSA generates the restoration command for the accessed address (*RESTR* in Fig. 2). The restoration command is pushed into the input queue of the media controller and rescheduled by the scheduler. By rewriting the vulnerable PCM cells, the restoration command shifts back the cell resistance to the stable amorphous state, effectively mitigating the RDE before its occurrence. If the table is full on the arrival of the new command, the dedicated replacement policy selects a victim candidate and replaces it with the new command address.

Checking the existence of 0s in data words can improve the efficiency of the restoration command. However, the read command does not accompany the data word; hence, the number of 0s cannot be instantly updated after its allocation. To tackle this challenging point, RSA adopts *late update* by observing write commands as well. That is, the existence of 0s in a read address is checked via the newly written data.

RSA is a novel approach to effectively mitigate RDEs compared to previous schemes. In particular, RSA completely differs from the ECC schemes [16]. Rather than incorporating the high-cost encoding/decoding logic for multi-bit ECC, RSA mitigates RDEs by recording vulnerable status within a low-cost SRAM-based table. In [10], the RDE is assumed as a different model, 1-to-0

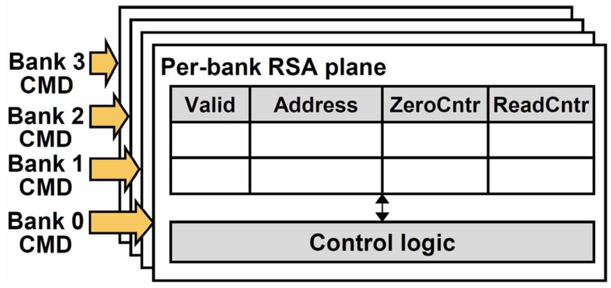

Fig. 3. Internal architecture of the RSA.

bit-flip error, compared to our study. The previous study tries to reduce the read latency by decreasing the latching latency. That is, the logical state value of a cell is latched before the required discharging time (i.e., sensing time) is met, incurring a 1-to-0 bit-flip error. To mitigate such RDEs, [10] proposes a probabilistic row scrubbing scheme using the double-error correcting ECC. In contrast, the proposed RSA assumes a material-level RDE model, and significantly mitigates RDEs by recording vulnerable status within a low-cost SRAM-based table. In [18], an on-demand scrubbing method, which additionally requires a multi-bit ECC scheme, is proposed to significantly mitigate RDEs. This previous study applies a 176-21 Reed-Solomon ECC, which means corrects up to 21 symbols out of 176 symbols. However, the Reed-Solomon ECC incurs significant resource overhead. According to latency and area formulas of multi-bit ECC [19], a 176-21 ECC incurs 1.424E+6 transistors and 8.6 ns of latency (i.e., approximately four cycles at 800 MHz frequency domain). In contrast, RSA requires 22.5 B of SRAM and two cycles of latency (for read and update). Considering a 6-transistor SRAM cell, the 22.5 B-SRAM in RSA is equivalent to 1,080 transistors, which are much fewer than those in [18]. Thus, the proposed RSA becomes more cost-effective than [18].

We observe that recent CNN inference tasks are read dominant and computation intensive; their data are 0-dominant (see Fig. 1(a)). Thus, RSA is essential to the PCM-based system running CNN inference workloads.

2. Implementation Details

Fig. 3 shows the internal architecture of the RSA. The RSA has multiple logic planes, each of which corresponds to one PCM bank. Thus, each RSA plane independently carries out its function, yielding bank-level parallelism. The RSA table in each RSA plane plays a key role in the proposed method, where the entry is updated by the control logic. Particularly, each table entry has four data fields for managing the vulnerable status of a PCM device address:

- *Valid* (1 bit): This bit shows the validity of the entry. It is set as 1, as the control logic allocates an entry for a new command address.
- *Address* (25 bit): It indicates the row and column addresses that are vulnerable to RDEs.
- *ReadCntr* (10 bit): It represents the number of read operations on the corresponding address.
- *ZeroCntr* (9 bit): It shows the existence of 0s in a data word. Although this field is enough to be one bit, we use multi-bits for representing the vulnerability of an address. Because a data word typically has 64 bytes, *ZeroCntr* requires 9 bits for implementation.

As a new command goes into the RSA, the control logic checks the existence of the entry in the corresponding table. The RSA operates in two different cases, depending on the finding result in the RSA table:

- HIT: If the input command hits on the RSA table, the control logic operates differently according to the command type. For the write command, the control logic updates the *ZeroCntr* in the entry. Subsequently, the *ReadCntr* increments if the *ZeroCntr* is updated as 0; otherwise, *ReadCntr* keeps its original value without modification. For the read command, the control logic simply increments the *ReadCntr* if the *ZeroCntr* hold the non-zero value. Finally, the control logic generates the restoration command when the *ReadCntr* exceeds the predefined threshold. The generated command is sent to the write queue in the media controller.
- MISS: If no corresponding entry is found, the table insertion is necessary. However, the RSA does not always insert a new address in the table, because some infrequent accesses may confuse the RSA. To overcome this problem, we apply the probabilistic

Table 1. Simulation configurations

Simulator	Device	Description
Gem5	Cores	Out-of-order, 4-core, 2 GHz
	L1 Cache	I-cache: 2-way set associative D-cache: 4-way set associative The capacity of each is 64 KB
	L2 cache	Shared last-level cache. 16-way set associative, 1MB
NVMain	Media controller	Separated write queue and read queue (64-entry), FR-FCFS
	PCM	Read: 100 ns RESET/SET: 100 ns/150 ns RDE limitation number: 1K Size: 8 GB (2-rank, 2-bank /rank)

insertion method, where the entry allocation carries out with probability *p*. As a result, the replacement and eviction are only performed for frequently accessed addresses. Please note that the restoration command for the replaced entry is also generated but with the probability of 1/16, which does not degrade the performance.

It is noteworthy that the write command is also handled for the hit-case above. This is because the *ZeroCntr* cannot be updated as an entry is allocated. The RSA allocates table entry for the read command; however, the read command does not accompany the data word. Therefore, the late update mechanism is incorporated by observing the number of 0s in the newly written data.

The proposed method has two parameters: (1) threshold for generating the restoration command, and (2) probability *p* for allocating a new table entry. The threshold value is determined as the RLN (see Section 2). Therefore, *ReadCntr* becomes 10 bits if the RLN is assumed as 1K. The other parameter, *p*, determines the insertion probability. Increasing the probability value incurs frequent entry replacement in the table, losing the chance to restore vulnerable addresses. On the other hand, moderately decreasing the probability value can manage vulnerable addresses in the table for a long period. In this study, *p* is assumed as 1/2, because it yields the fewest RDEs.

Based on the characteristics of RDEs, a replacement policy is also required for the RSA table. Thus, we exploit *ZeroCntr* and *ReadCntr* to define the replacement policy. The replaced entry must be the least urgent entry among entities in the table. Because *ReadCntr* shows the

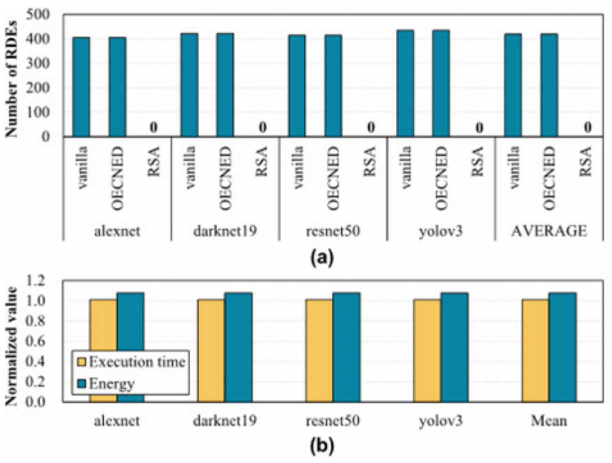

Fig. 4. Evaluations results: (a) the number of RDEs for each scheme (the lower the better); (b) normalized execution time and energy of RSA.

vulnerability of an address, we first select the entries which have the minimum *ReadCntr* values. If multiple candidates are selected, the entry which have minimum *ZeroCntr* value is selected as the final candidate.

In the PCM module, the media controller requires slight modification to support RSA in two aspects. First, acquiring the old data is necessary for the restoration command. Thus, a pre-write read operation is carried out ahead of the restoration command. Particularly, the scheduler confers the medium priority, which is the higher priority than write requests but the lower priority than read requests, to the restoration command. This is because, write requests in the controller primarily drain when the queue is full. The second modification is a merge operation, by which the restoration command can coalesce with write commands heading to the same address.

IV. EVALUATION

1. Evaluation Methodologies

Table 1 shows the simulation environment. We conduct the trace-driven simulation to reduce the simulation time, because running billions of CNN inference instructions requires a long simulation time. On gem5 [1], we run *Darknet*, an open-source CNN framework in C, to extract the memory traces. Please note that four widely used CNN models are chosen for the simulation: *Alexnet*, *Darknet19*, *Resnet50*, and

Yolov3. All memory traces are extracted after fast-forwarding the initialization phase (i.e., weight loading and image loading). An 8 GB PCM system is built upon an NVM simulator, *NVMain* [12]. The read latency is set to 100 ns, and the write latencies of SET and RESET are set to 150 ns and 100 ns, respectively, with the differential write support [15]. The energy per access on the PCM and the SRAM is obtained from NVSim [3] and CACTI [8], respectively, both of which are configured with 22 nm technology, as [21] does. The *baseline* in this paper does not apply any RDE mitigation method.

2. Evaluation Results

Fig. 4(a) compares the number of RDEs for the vanilla (i.e., baseline), OECNED (i.e., eight errors correction and nine errors detection), and RSA. The figure shows the same numbers of RDEs for the baseline and OECNED. The OECNED ECC is capable of correcting up to eight errors at a time. Moreover, we observe that RDEs occur on more than eight cells when a 64 B data word is accessed at a time. As a consequence, an alternative approach to mitigating RDEs rather than traditional ECC is required. In contrast, the proposed method eliminates all RDEs for all inference tasks, because the proposed method restores the vulnerable cells before the RDE occurs by managing vulnerable candidates in an SRAM-based table.

The execution time and energy consumption are crucial metrics for evaluating the practicality of the proposed method. Fig. 4(b) shows the execution time, which is normalized to the baseline, to estimate the performance overhead of the proposed method. As shown in the figure, the proposed method yields 1.011 of the normalized execution time; hence, the overhead of the proposed method is as low as 1.1%. Fig. 4(b) also presents the normalized energy consumption; it is increased by 7.6% compared to the baseline. This is because the RSA table drains more energy from the SRAM. Moreover, the number of entries is set as 4, requiring 4×(1+25+9+10 bits)=22.5 B of SRAM capacity. In conclusion, the proposed method has negligible performance, energy, and area overheads, while notably reducing the number of RDEs.

Next, to show the effectiveness of the RSA, we compare the performance of the RSA for harsher RLNs,

Table 2. Average performance of different RLNs

Metrics	RLN=1K	RLN=512	RLN=256
Number of RDEs	0	0	311
Execution time	1	1	1.000000466
Energy consumption	1	1	1.000045344

such as 256 and 512, as shown in Table 2. It should be noted that the threshold for generating the restoration command is determined as the RLN (see Section III-2), while the number of RSA entries does not change (i.e., four entries). Moreover, the execution time and energy consumption values in the table are normalized to the case of RLN=1K. Table 2 presents that the RSA eliminates all RDEs for the RLN of 512. However, the RSA cannot eliminate all RDEs when the RLN becomes 256 because the baseline of RLN=256 yields four times more RDEs compared to the baseline of RLN=1K. The main reason for the lower mitigation performance of the RSA is the replacement in the RSA table. Entries in the RSA table may be replaced before *ZeroCntr* reaches the threshold of generating restoration commands; such a case is more common for a smaller RLN. Nonetheless, both the time and energy overheads are negligible for all RLNs. As a result, the RLN greater or equal to 512 is a reasonable range for our RSA.

V. CONCLUSIONS

Owing to the low standby power of the PCM, the CNN inference becomes the proper application on the PCM-based computer system. However, RDE hinders the population of PCM devices in the consumer market, because the CNN inference is read intensive. In this study, we propose the read disturbance scrubbing assistance to notably reduce the number of RDEs. By leveraging an SRAM-based table, the proposed method restores the vulnerable cells on demand. Moreover, a dedicated replacement policy that fully utilizes the characteristics of RDEs is proposed. The proposed method is vendor-friendly, because it is implemented in the PCM module without the modification of the PCM module. The evaluation results show that RSA significantly reduces RDEs with negligible performance degradation.

ACKNOWLEDGMENTS

This study was supported by the Research Program funded by the SeoulTech(Seoul National University of Science and Technology).

REFERENCES

[1] N. Binkert et al., "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1-7, Aug. 2011.

[2] J. Choe, "Intel 3d xpoint memory die removed from intel optane pcm," [Online]. Available: https://www.techinsights.com/blog/intel-3d-xpoint-memory-die-removed-intel-optanetm-pcm-phase-change-memory

[3] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi, "Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, pp. 994-1007, July 2012.

[4] K. Doshi, E. Giles, and P. Varman, "Atomic persistence for scm with a non-intrusive backend controller," in *Proceedings of the 22nd International Symposium on High Performance Computer Architecture (HPCA)*, 2016, pp. 77-89.

[5] R. Fisher, "Optimizing nand flash performance," in *Flash Memory Summit*, 2008.

[6] U. Gupta et al. "The architectural implications of facebook's dnn-based personalized recommendation," in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2020, pp. 488-501.

[7] J. Handy. (2019) What's inside and optane dimm. [Online] Available: https://thememoryguy.com/whats-inside-an-optane-dimm

[8] Hewlett Packard Lab. Cacti 6.5. [Online]. Available: https://www.hpl.hp.com/research/ cacti/

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

[10] P. J. Nair, C. Chou, B. Rajendran, and M. K. Qureshi, "Reducing read latency of phase change memory via early read and turbo read," in *Proceedings of the 21st International Symposium on High Performance Computer Architecture (HPCA)*, 2015, pp. 309-319.

[11] Oracle. Multithreaded programming guide. [Online] Available: https://docs.oracle.com/cd/E26502 01/html/E35303/tlib-1.html

[12] M. Poremba, T. Zhang, and Y. Xie, "Nvmain 2.0: A user-friendly memory simulator to model (non-) volatile memory systems," *IEEE Computer Architecture Letters*, vol. 14, no. 2, pp. 140-143, 2015.

[13] U. Russo, D. Ielmini, A. Redaelli, and A. L. Lacaita, "Modeling of programming and read performance in phase-change memories-part ii: Program disturb and mixed-scaling approach," *IEEE Transactions on Electron Devices*, vol. 55, no. 2, pp. 515-522, Feb 2008.

[14] K. Wu et al., "Exploiting intel optane ssd for microsoft sql server," in *Proceedings of the 15th International Workshop on Data Management on New Hardware*. Association for Computing Machinery, 2019.

[15] B. Yang et al., "A low power phase-change random access memory using a data-comparison write scheme," in *Proceeding of the 2007 IEEE International Symposium on Circuits and Systems*, May 2007, pp. 3014-3017.

[16] C. Yang, Y. Emre, Y. Cao, and C. Chakrabarti, "Improving reliability of non-volatile memory technologies through circuit level techniques and error control coding," *EURASIP Journal on Advances in Signal Processing*, vol. 34, no. 211, Oct 2012.

[17] H. Lee et al., "PCMCsim: An Accurate Phase-Change Memory Controller Simulator and its Performance Analysis," in *Proc. 2022 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS 2022)*, May 2022, pp. 299-309.

[18] M. Kim, J. Lee, H. Kim, and H.-J. Lee, "An Optimal On-Demand Scrubbing Solution for Read Disturbance Errors in Phase-Change Memory," *IEIE Transactions on Smart Processing & Computing*, vol. 10, no. 1, pp. 55-60, Feb. 2021.

[19] D. Strukov, "The area and latency tradeoffs of binary bit-parallel BCH decoders for prospective nanoelectronic memories," 2006 Fortieth Asilomar

Conference on Signals, Systems and Computers, 2006, pp. 1183-1187.
[20] M. Kim, H. Lee, H. Kim, and H.-J. Lee, “WL-WD: Wear-Leveling Solution to Mitigate Write Disturbance Errors for Phase-Change Memory,” IEEE Access, vol. 10, pp. 11420-11431, 2022.
[21] H. Lee et al., “An In-Module Disturbance Barrier for Mitigating Write Disturbance in Phase-Change Memory,” IEEE Transactions on Computers, to be published.

Hyokeun Lee received the B.S. and Ph.D. degrees in electrical and computer engineering (ECE) from Seoul National University, Seoul, South Korea, in 2016 and 2021, respectively. He is currently working as a postdoctoral researcher in Inter-university Semiconductor Center (ISRC) at Seoul National University. His current research interests include nonvolatile memory controller design, architecture simulation modeling, and computer architecture.

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.