IEIE Transactions on Smart Processing and Computing, vol. 10, no. 1, February 2021
https://doi.org/10.5573/IEIESPC.2021.10.1.055

IEIE Transactions on Smart Processing and Computing

An Optimal On-demand Scrubbing Solution for Read Disturbance Errors in Phase-change Memory

Moonsoo Kim[1], Juhan Lee[1], Hyun Kim[2],*, and Hyuk-Jae Lee[1]

[1] Department of Electrical and Computer Engineering, Seoul National University, Seoul, Korea
{kimms213, jhyi, hyuk_jae_lee}@capp.snu.ac.kr
[2] Department of Electrical and Information Engineering and Research Center for Electrical and Information Technology, Seoul National University of Science and Technology, Seoul, Korea hyunkim@seoultech.ac.kr

* Corresponding Author: Hyun Kim

Received May 16, 2020; Revised June 23, 2020; Accepted July 20, 2020; Published February 28, 2021

* *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 is a promising technology due to its attractive properties. However, phase-change memory is difficult to commercialize because of its reliability issues. A read disturbance error, which is the main cause of reliability issues, occurs when a cell is repeatedly read. A conventional solution for read disturbance errors is periodically scrubbing the cells. However, this method requires read counters to count the number of reads per word. This paper proposes an on-demand scrubbing solution that does not require read counters, which significantly reduces resource overhead. The proposed method observes the number of errors in a word using error-correcting code. If the number of errors is larger than a pre-defined threshold, scrubbing is performed to fix the errors. The proposed method removes nearly 1GB of hardware overhead required by read counters, and fixes more than 99.99% of read disturbance errors.

Keywords: Non-volatile memory, Phase-change memory, Read disturbance errors, On-demand scrubbing

1. Introduction

A modern computer system requires large amounts of main memory owing to its multi-core structure and complex applications. In particular, data-intensive applications such as big data and deep learning require a large amount of main memory to support large amounts of data [1-3]. As a result, the need for large-capacity main memory with low power consumption and high reliability has become important [4, 5], and studies on the use of phase-change memory (PCM) as main memory have been actively conducted [6-8]. The cell size in PCM is smaller than DRAM, so the module can be denser, enabling a large memory capacity [9]. Furthermore, owing to the non-volatile characteristics of PCM, it is more advantageous than DRAM in terms of power efficiency and data retention time.

Despite these advantages, PCM suffers from low reliability, which needs to be addressed in order to use PCM as main memory. One of the main causes of reliability issues in PCM is read disturbance errors (RDEs) [10, 11]. An RDE is a phenomenon whereby cells that are repeatedly read are damaged by thermal energy. RDEs occur when the number of reads exceeds a certain threshold. A conventional solution for RDEs is to scrub the cells in a word before the number of reads reaches the threshold. Memory scrubbing first reads a word, corrects bit errors with error-correcting code (ECC), and writes the corrected word back to the same location. Periodically scrubbing a word therefore prevents RDEs in that word.

However, periodical scrubbing requires read counters, which results in significant resource overhead because the number of reads must be counted in order to trigger scrubbing. In this paper, on-demand memory scrubbing that does not require read counters is proposed. Under the given RDE model with ECC, the probability distribution of the number of errors that occur with an additional read is derived. By using the derived probability distribution, the proposed solution suggests whether to scrub or not based on the current number of errors. Because the proposed solution only requires the number of errors, it does not need read counters, thereby eliminating nearly 1GB of resource overhead. The contributions from this paper are summarized as follows.

- A probabilistic model for RDEs is mathematically derived, and the optimal on-demand scrubbing policy is derived from the proposed model.
- Monte-Carlo (MC) simulation is conducted to verify the probabilistic model.
- The proposed on-demand scrubbing eliminates more than 1GB needed for read counters, while fixing more than 99.99% of RDEs.

The remainder of this paper is organized as follows. Section 2 introduces the background, and Section 3 presents the proposed on-demand scrubbing method. In Section 4, experimental results are given. Finally, Section 5 concludes the paper.

2. Background

In this section, the background for RDE error models and RDE mitigating schemes is presented.

2.1 Error Models for RDEs

Typically, a counter-based error model is used for RDEs [9]. Under this model, each cell has an RDE threshold, and when the number of reads reaches the threshold, an RDE occurs. The RDE threshold values follow a Gaussian distribution, $N\left(m,\sigma^2\right)$. For later discussions, m at 3,000 and various σ values are assumed.

The word size for PCM typically ranges from 64B to 256B [7]. In this paper, we assume 128B words. For ECC, 176-21 Reed–Solomon code is assumed so up to 21 symbols can be corrected out of 176 symbols in total. These 1,408 cells in a word are assumed to have independent RDE threshold values modeled as a Gaussian distribution.

2.2 RDE Mitigating Schemes

To mitigate RDE occurrences, a memory scrubbing method is used [12]. In conventional methods, each word utilizes a read counter. If the counter value reaches a certain threshold, the method reads a whole word and checks for errors via ECC. If errors are found, they are corrected and the word is rewritten. This read-and-fix process is called memory scrubbing. Conventional memory scrubbing, which uses a read counter, can remove RDEs effectively as long as the scrubbing threshold value is well chosen. However, as shown in Fig. 1, it requires a read counter per word, which means an extra 2B of storage must be allocated per word. Taking into account that the word size in PCM is typically between 64B and 256B, the extra storage takes about 1/32 to 1/128 of the total PCM capacity. Moreover, these read counters are updated frequently, and thus, DRAM should be used for them. Assuming 512GB of PCM capacity and a 128B word size, nearly 8GB of DRAM is used only for read counters, which is a significantly large overhead.

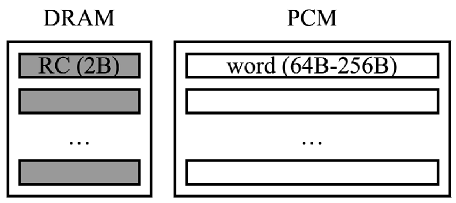

Fig. 1. Diagram of PCM and its read counters.

3. On-demand Memory Scrubbing

In this section, an on-demand memory scrubbing method that effectively eliminates read counters is described. First, the probability distribution under the Gaussian counter-based error model is derived, and then, an efficient on-demand scrubbing policy under the given probability distribution is suggested.

3.1 Probability Distribution for the Number of Errors

Denote as L the number of errors, with K as the number of reads, and T as the RDE threshold. The first probability to derive is e_k, which is the probability that an error occurs when the number of reads is k $\left(K=k\right)$. e_k is derived as follows:

$$e_k = P\left(k \ge T\right) = P\left(k \ge N\left(m,\sigma^2\right)\right). \tag{1}$$

Because errors occur when the number of reads exceeds the RDE threshold, the first equality in (1) stands. The second equality is from the Gaussian modeling of T. It should be noted that $P\left(k \ge N\left(m,\sigma^2\right)\right)$ can be easily calculated from the normal distribution table.

When $K=k$, the probability of L being l is a binomial distribution $B\left(l,176,e_k\right)$. More specifically, for all 176 symbols, the probability of an error is e_k, so they have a binomial distribution as follows:

$$P(l|k) = B\left(l;176,e_k\right) = \binom{176}{l} \cdot e_k^{\,l} \cdot \left(1-e_k\right)^{176-l}. \tag{2}$$

However, we are interested in probability $P(k|l)$, not $P(l|k)$, because the value that can be observed is l, not k. By using Bayes's rule, $P(k|l)$ can be derived as follows:

$$P(k|l) = \frac{P(l\mid k)\cdot P\left(k\right)}{P\left(l\right)} = \frac{P(l\mid k)}{\sum_k P(l\mid k)}. \tag{3}$$

This means that when the observed number of errors is l, the hidden number of reads, k, has the probability distribution derived from (3).

Lastly, probability $P(L'=l'|L=l)$ is derived. L' represents the number of errors when an additional read operation is performed. Therefore, probability $P(l'|l)$ means the number of errors l' from l due to an additional read. It is derived as follows:

$$P(l'|l) = \sum_{k=0}^{\infty} B\left((l'-l);176-l,e_{k+1}\right)\cdot P(k\,|\,l). \qquad (4)$$

$B\left((l'-l);176-l,e_{k+1}\right)$ calculates the probability that the number of additional errors that occur from one more reads is $(l'-l)$. Because the number of reads is now k+1, e_{k+1} is used as the error probability. By summing the term over k, the desired $P(l'|l)$ is calculated.

3.2 On-demand Scrubbing Policy

So far, probability distribution $P(l'|l)$ is calculated with (4). This distribution is used to derive the on-demand scrubbing policy. Assuming that ECC can correct up to 21 symbols, scrubbing must be done before the number of errors exceeds 21. This means that probability $P(L'>21|L=l)$ must be small enough when the scrubbing is done. It can be calculated as follows:

$$P(L'>21|l) = 1-P(L'\le 21|l) = 1-\sum_{l'=l}^{21} P(l'|l). \qquad (5)$$

Table 1 shows the value of $P(L'>21|L=l)$ for various values of σ. When $\sigma=10$ and $L=6$, the probability is 6.5E-7%. This means that when the current number of errors is 6, the probability that the number of errors becomes larger than 21 due to one more read operation is 6.5E-7%. It is obvious that the probability must be very small to prevent an uncorrectable error (UE). Therefore, performing scrubbing at a small value for L is always better in terms of removing UEs. However, a small L value causes too-frequent scrubbing. Table 2 shows the expected value for the number of reads (K) with respect to L. We can see that K increases as L increases. Moreover, the difference in the value is relatively large when L is small (from 0 to 6). On the other hand, when L is relatively large, the difference becomes smaller. To sum up, it is best to delay scrubbing while the UE probability is small enough, but when the L value becomes relatively large, to delay scrubbing does not give much benefit, because the difference in the K value is minor. For example, when the probability goal is 99.999%, which means the UE probability should be less than 0.001%, the selected L values that trigger scrubbing are 7, 10, and 13 when σ is 10, 20, and 50, respectively. The average reads at the selected points are 2984, 2969, and 2920, respectively.

The proposed on-demand scrubbing provides a significant hardware overhead reduction because it does not require read counters. As mentioned above, a read counter for a single word takes up 2B. Assuming a word size of 128B, 1/64 of the total PCM capacity must be allocated to read counters. For example, if the PCM capacity is 64GB, total of 1GB of storage must be allocated to read counters. This means that using on-demand scrubbing significantly reduces storage overhead by about 1/64 of the total capacity.

Table 1. The probability of violation with respect to *L*.

L	$\sigma=10$	$\sigma=20$	$\sigma=50$
0-6	<6.5E-7%	<4.8E-7%	<2.4E-8%
7	2.9E-4%	6.1E-6%	1.6E-7%
8	5.0E-3%	2.0E-5%	1.1E-6%
9	0.03%	8.4E-5%	8.7E-6%
10	0.08%	2.8E-4%	2.2E-5%
11	0.31%	3.4E-3%	9.1E-5%
12	0.82%	0.02%	3.3E-4%

Table 2. The expected value of K with respect to *L*.

L	$\sigma=10$	$\sigma=20$	$\sigma=50$
0-6	2956-2982	2913-2963	2783-2908
7	2983	2965	2912
8	2983	2967	2915
9	2984	2968	2917
10	2984	2969	2918
11	2984	2970	2919
12	2985	2970	2920

4. Simulation Results

In this section, MC simulation results regarding the proposed on-demand memory scrubbing are presented. For MC simulations, RDE threshold value T for each bit in a word was randomly sampled from the Gaussian distribution, $N\left(3000,\sigma^2\right)$. Under the generated T values in a word, the number of errors (L) was followed as the read count (K) increased.

Table 3 shows the MC simulation results with respect to various standard deviation values. The number of MC trials was 1,000,000. The first column of Table 3 shows the number of errors in the current state. From the current state, if one more read causes a violation (*i.e.*, the number of errors exceeds 21), the case is counted as a *violated read*. The second, fifth, and eighth columns in Table 3 show the number of violated reads for standard deviations of 10, 20, and 50, respectively. For example, when the standard deviation was 10, the violated read value was 4 for an L of 7. This means that in these four cases, the error number changed directly from 7 to a value larger than 21. The columns labeled Probability of VR show the ratio of violated reads with respect to the total number of trials. The column Average Read Threshold represents the average read counts among the violations. For example, when the standard deviation was 10, the average read

Table 3. MC simulation results showing violated read and average read threshold values.

L	$N(3000,10^2)$			$N(3000,20^2)$			$N(3000,50^2)$		
	Violated Read	Probability of VR (%)	Average Read Threshold	Violated Read	Probability of VR (%)	Average Read Threshold	Violated Read	Probability of VR (%)	Average Read Threshold
0-6	0	0	0	0	0	0	0	0	0
7	**4**	**0.0004**	**2973**	0	0	0	0	0	0
8	37	0.0037	2973	0	0	0	0	0	0
9	197	0.0197	2973	0	0	0	0	0	0
10	706	0.0706	2973	**3**	**0.0003**	**2952**	0	0	0
11	2462	0.2462	2973	25	0.0025	2949	0	0	0
12	7604	0.7604	2973	160	0.016	2948	0	0	0
13	18692	1.8692	2973	866	0.0866	2948	**3**	**0.0003**	**2881**
14	40413	4.0413	2973	3669	0.3669	2949	47	0.0047	2876
15	73756	7.3756	2973	12916	1.2916	2949	432	0.0432	2875
16	115931	11.5931	2973	38812	3.8812	2949	2880	0.288	2876
17	157267	15.7267	2973	95337	9.5337	2949	17173	1.7173	2876
18	186598	18.6598	2974	187397	18.7397	2949	78581	7.8581	2876
19	198617	19.8617	2974	293767	29.3767	2949	267358	26.7358	2877
20	197716	19.7716	2974	367048	36.7048	2950	633526	63.3526	2877

threshold was 2,973 for an L of 7. This means that the average read count over four violations was 2,973.

From the MC simulation results in Table 3, two observations can be drawn. First, the distribution of violated reads changed with respect to the standard deviation. As the standard deviation of the underlying Gaussian model increased, the violations occurred late. In other words, when the standard deviation was relatively large, more errors could be endured until scrubbing. The other observation is that the average read threshold value remained almost the same while L increased. This means that even if we accumulate more errors before scrubbing, in order to reduce scrubbing frequency, the actual reduction in the scrubbing frequency is negligible. From this observation, the optimal on-demand scrubbing point for L is the point where a violation first occurs. Therefore, the optimal on-demand scrubbing points for each standard deviation value for L are 7, 10, and 13, respectively.

It should be noted that the MC simulation results presented in Table 3 verify the probability distribution described in Section 3.1. The probabilities of violations derived from the probability distributions shown in Table 1 give values similar to those in Table 3. For the average read threshold value drawn from the probability distribution in Table 2, the results verify that the value does not increase significantly when L is greater than 6.

In addition to RDEs, there are other reliability issues regarding PRAM, such as write disturbance error (WDE). That is, for example, if the current number of errors is 8, then it is possible that four RDEs and four WDEs both occur. Under this circumstance, consider a case where the proposed on-demand scrubbing policy initiates scrubbing. The policy thinks that all eight errors were caused by RDEs, but in this case, only four errors were caused by RDEs. Therefore, the probability that an additional read will cause an ECC violation is even less when other errors are considered. In summary, the proposed approach, which only considers RDEs, is a conservative approach, and other errors do not compromise the reliability of the proposed on-demand scrubbing.

5. Conclusion

To efficiently address RDEs, this paper proposes an on-demand scrubbing policy that does not require read counters. Assuming the memory capacity of PCM is 64GB, a read counter per word takes nearly 1GB of storage, which creates a significant resource overhead. In the proposed method, without this resource overhead, more than 99.99% of read disturbance errors can be fixed. From a mathematically derived probability distribution model of RDE occurrence, the optimal on-demand scrubbing policy is drawn up with respect to a certain standard deviation in RDE threshold values. MC simulation results also verified the derived probability distribution model. It should be noted that the probability of preventing violations can be increased by scrubbing more often. This means that there is a trade-off between reliability and performance, and the user can adaptively select the configuration according to the application.

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 the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education under Grant NRF-2019R1A6A1A03032119.

References

[1] 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), Apr. 2020.

[2] D. T. Nguyen, N. H. Hung, H. Kim, and H.-J. Lee, “An Approximate Memory Architecture for Energy Saving in Deep Learning Applications,” in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 67, no. 5, pp. 1588-1601, May 2020.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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*, ser. ISCA ’09.

[8] M. K. Qureshi, V. Srinivasan, and J. A. Rivers, “Scalable high performance main memory system using phase-change memory technology,” in *Proceedings of the 36th Annual International Symposium on High-Performance Comp Architecture* (HPCA), 2012.

[9] H.-S. P. Wong, S. Raoux, S. Kim, J. Liang, J. P. Reifenberg, B. Rajendran, M. Asheghi, and K. E. Goodson, “Phase change memory,” *Proceedings of the IEEE*, vol. 98, no. 12, pp. 2201–2227, 2010.

[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," *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, Burlingame, CA, 2015

[11] S. Rashidi, M. Jalili, and H. Sarbazi-Azad., “Improving MLC PCM Performance through Relaxed Write and Read for Intermediate Resistance Levels,” ACM Trans. Archit. Code Optim. 15, 1, Article 12 (April 2018), 31 pages.

[12] M. Awasthi, M. Shevgoor, K. Sudan, B. Rajendran, R. Balasubramonian and V. Srinivasan, "Efficient scrub mechanisms for error-prone emerging memories," *IEEE International Symposium on High-Performance Comp Architecture (HPCA)*, New Orleans, LA, 2012

Moonsoo Kim received a B.S. and Ph.D. degrees in electrical and computer engineering from Seoul National University, Seoul, Korea, in 2014 and 2020, respectively. In 2020, he joined Inter-University Semiconductor Research Center from Seoul National University, Seoul, Korea as a post-doctoral researcher. His research interests include SoC design of video/image applications, and low-power, reliable design of memory hierarchy.

Joohan Yi received the B.S. degree in electric and electrical engineering from Korea University, Seoul, Korea, in 2018. He is currently working toward the integrated M.S and Ph.D degree in electrical and computer engineering at Seoul National University, Seoul, Korea. His research interests include memory hierarchy, deep neural network processor, and image processing for Robot Systems.

Hyun Kim received the B.S., M.S. and Ph.D. degrees in Electrical Engineering and Computer Science from Seoul National University, Seoul, Korea, in 2009, 2011 and 2015, respectively. From 2015 to 2018, he was with the BK21 Creative Research Engineer Development for IT, Seoul National University, Seoul, Korea, as a Research Professor. In 2018, he joined the Department of Electrical and Information Engineering, Seoul National University of Science and Technology, Seoul, Korea, where he is currently working as an Assistant Professor. His research interests are the areas of algorithm, computer architecture, and SoC design for low-complexity multimedia applications.

Hyuk-Jae Lee received B.S. and M.S. degrees in Electronics Engineering from Seoul National University, Korea, in 1987 and 1989, respectively, and the Ph.D. degree in Electrical and Computer Engineering from Purdue University at West Lafayette, Indiana, in 1996. From 1998 to 2001, he worked at the Server and Workstation Chipset Division of Intel Corporation in Hillsboro, Oregon as a senior component design engineer. From 1996 to 1998, he was on the faculty of the Department of Computer Science of Louisiana Tech University at Ruston, Louisiana. In 2001, he joined the School of Electrical Engineering and Computer Science at Seoul National University, Korea, where he is currently working as a Professor. He is a founder of Mamurian Design, Inc., a fabless SoC design house for multimedia applications. His research interests are in the areas of computer architecture and SoC design for multimedia applications.