Wear-leveling Scheduler for Phase-Change RAM Main Memory for Mobile Consumer Electronics

Sang-Hoon Park, Hyeokjun Seo, Taehee You, Jin-Young Kim and Eui-Young Chung
School of Electrical and Electronic Engineering
Yonsei University
Seoul, Republic of Korea
{soskhong, jjsky7, xoqhd1212, jy0615.kim}@dtl.yonsei.ac.kr, eychung@yonsei.ac.kr

***Abstract*—Main memory systems based on phase-change random access memory (PRAM) have been actively researched due to low static power consumption, which is adequate to mobile consumer electronics. However, PRAMs require wear-leveling, which incur performance overhead, to compensate their limited life-time. Even though many works have discussed the overhead in terms of write latency, read latency is also affected and the degradation is even severer since PRAM read is faster than write. Consequently, this paper proposes a wear-leveling scheduler that reduces the effect of wear-leveling overhead on read latency. The proposed method which can cooperate with various wear-leveling algorithms enables wear-leveling only when no read request is issued for a period longer than the pre-defined threshold. In spite of simplicity, a PRAM-based memory system equipped with the proposed method achieves 50% shorter read latency without affecting wear-leveling performance.**

Keywords—phase-change RAM, main memory, wear-leveling, scheduling

I. INTRODUCTION

Phase-change random access memory (PRAM) is a non-volatile memory which is one of the most promising candidates for non-volatile main memory systems of mobile consumer electronics due to its reasonable performance, dense architecture and low static power consumption [1]. Like most of non-volatile memories, PRAM's cells have a limited life-time which can be represented by the number of given write operations. To maximize the life-time, wear-leveling (WL) schemes, which evenly distributes write operations over the entire cells, have been proposed [2-7]. The schemes equalize write counts by moving data to new physical location periodically. The data movement, which incurs additional read and write operations, is performance overhead since it is not requested by the host. Researchers have analyzed the overhead in the perspective of write performance since WL is triggered by write transactions.

Read performance, however, is also affected since WL may occupy PRAM even when new read transaction is issued by the host. Furthermore, read should be more sensitive than write to the overhead since read latency of PRAM is shorter than that of write.

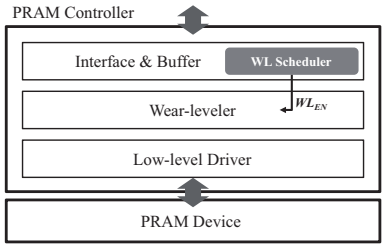

Fig. 1 Block diagram of PRAM controller with the proposed wear-leveling scheduler

In this paper, therefore, we propose a WL scheduler that minimizes overhead which is incurred by WL and degrading read performance. More specifically, the proposed scheduler activates wear-leveling only when negative effect of wear-leveling on read transactions is minimized. At the same time, to maintain WL performance of sophisticated algorithms, the proposed scheduler is designed to generally work with existing WL algorithms.

II. PROPOSED METHOD

Fig.1 shows the architecture of a PRAM controller with the proposed WL scheduler. Transactions to PRAM are fed by interface module first and serviced by its buffer temporarily. Before the low-level driver which actually drives PRAM devices with low-level commands, wear-leveler translates addresses of transactions into PRAM's physical addresses to map the location of moved data by itself.

The proposed WL scheduler can be implemented in the interface module to analyze transactions to schedule WL. It gives a signal, WL_{EN}, to the wear-leveler and wear-leveling is enabled only when the signal is asserted. In other words, WL scheduler decides durations that wear-leveler can move data for wear-leveling with additional read and write operations. Since required change to wear-leveler for the proposed method, which is only en/disabling data movement according to WL_{EN}, is quite simple, the proposed method generally co-work with a variety of existing wear-levelers for PRAM.

The value of WL_{EN} is renewed when a transaction is conveyed to the wear-leveler. To minimize effect of wear-leveling on read latency, WL scheduler enables wear-leveling only if there is no read transaction for a sufficiently long time

TABLE I
USED BENCHMARKS

Name	Description
box	Adopt a box filter (3x3) to an image (1000x1000 pixels)
conv	2D convolution with 600x600 data points
hotspot	Thermal simulation of a chip with a 800x800 grid
matrix	Multiplication of two matrices (300x300)
path	Path finding with a 1400x1400 array

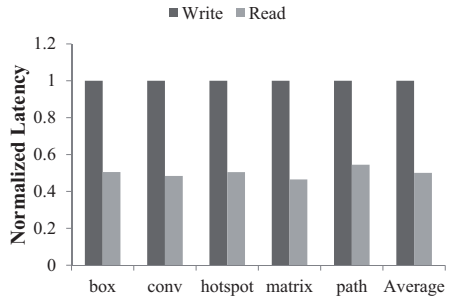

Fig. 2 Normalized write and read latency of PRAM main memory with the proposed scheduler

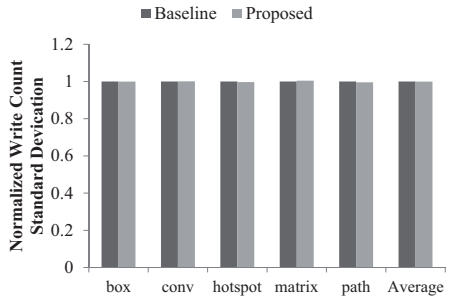

Fig. 3 Normalized write count standard deviation of PRAM main memory

interval, which is longer than a certain time-out threshold. Consequently, the value of WL_{EN} is set to one only the following statement,

$$(I_R > TH_R) \quad (1)$$

, where I_R is the time since the last read request has handled and TH_R is the time-out threshold, is true.

According to (1), WL_{EN} is asserted only when a new transaction is given to wear-leveler and there has been no issued read transaction during the interval longer than TH_R. Although we set TH_R as a pre-defined static value in this paper for lower hardware and timing overheads, it can be easily extended to dynamically varying value.

The strong points of the proposed WL scheduler are quite clear. First, it is simple thus both hardware and timing overhead can be minimized. The hardware overhead is limited to counters to compare I_R and TH_R. Additionally, wear-levelers for PRAM can generally adopt the proposed scheduler with minor modification, which is one additional signal and related logics, to minimize their negative effects on read latency. In spite of its generality and simplicity, the effect of the proposed scheduler on read latency is quite huge as shown in the next section.

III. EVALUATION

In order to evaluate the proposed scheduler, a simulator based on Synopsys's Platform Architect is implemented which includes a single-core ARM9 processor (100MHz) using PRAM as its main memory. Real ARM binaries are executed and evaluation metrics such as latency (read/write) and standard deviation of write count of PRAM cells are measured. The write/read speed of PRAM follows specifications in [8] and TH_R is set to 100us statically. We executed five benchmarks and Table I summarizes their characteristics.

For comparison, the baseline configuration is set as a PRAM controller equipped only with the wear-leveler whereas the proposed PRAM controller also includes the proposed WL scheduler. For the common wear-leveler for both configurations, the start-gap wear-leveler [2] is used since it is one of the representative wear-levelers for PRAM main memory.

Fig. 2 shows write and read latency of the PRAM controller with the proposed WL scheduler which is normalized to the baseline (the PRAM controller only with the start-gap wear-leveler). Read latency is 50% shorter than baseline whereas write latency does not change. In terms of read latency, these results imply that WL scheduler successfully reduces the effect of wear-leveling overheads on read transactions. At the same time, it does not affect write latency at all. According to our analysis, read/write transactions group with the same type of transactions, so that WL scheduler always asserts WL_{EN} during write intensive periods whereas wear-leveler remains disabled during read intensive periods.

Standard deviation of PRAM cell's write count normalized to that of baseline is shown in Fig. 3. Since the purpose of wear-leveling is to distribute write operations evenly over the entire cells, low standard deviation is preferred. Although read latency becomes half, the average variation incurred by WL scheduler is smaller than 0.1% which is negligible. In summary, the proposed method significantly reduces read latency without affecting wear-leveling performance.

IV. CONCLUSION

In this paper, we propose a wear-leveling scheduler which is designed to minimize the effect of PRAM wear-leveling on read transactions. The proposed WL scheduler enables wear-leveling only when there has been no read transaction for a long period, which is longer than TH_R specifically, thus successfully reduces read latency to half compared to the baseline. Furthermore, wear-leveling performance, which is represented by write count standard deviation, and write latency are hardly affected by the WL scheduler. As a future work, we will design an adaptive algorithm to dynamically find the optimal TH_R.

ACKNOWLEDGEMENT

This work (Grant No. C0146555) was supported by Business for Cooperative R&D between industry, Academy,

and Research Institute funded Korea Small and Medium Business Administration in 2013, by IDEC (IC Design Education Center) and by SK-Hynix Semiconductor Inc.

REFERENCES

[1] M. K. Qureshi, V. Srinivasan, and J. A. Rivers, "Scalable high performance main memory system using phase-change memory technology," ACM SIGARCH Computer Architecture News, vol. 37, no. 3, pp. 23-33, 2009.

[2] 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 IEEE/ACM International Symposium on Microarchitecture, pp. 14-23, December 2009.

[3] N. H. Seong, D. H. Woo, and H.-H. S. Lee, "Security refresh: prevent malicious wear-out and increase durability for phase-change memory with dynamically randomized address mapping," ACM SIGARCH Computer Architecture News, vol. 38, no. 3, pp. 383-394, June 2010.

[4] A. Seznec, "A phase change memory as a secure main memory," Computer Architecture Letters, vol. 9, no. 1, pp. 5-8, 2010.

[5] J. Dong, L. Zhang, Y. Han, Y. Wang, and X. Li, "Wear rate leveling: lifetime enhancement of PRAM with endurance variation," In Proceedings of the 48th Design Automation Conference, pp. 972-977, June 2011.

[6] J. Yun, S. Lee, and S. Yoo, "Bloom filter-based dynamic wear leveling for phase-change RAM," In Proceedings of the Conference on Design, Automation and Test in Europe, pp. 1513-1518, March 2012.

[7] D. Liu, T. Wang, Y. Wang, Z. Shao, Q. Zhuge, and E. H. M. Sha, "Curling-PCM: Application-specific wear leveling for phase change memory based embedded systems," In Proceeding of ASP-DAC 2013, pp. 279-284, 2013.

[8] Y. Choi, I. Song, M. H. Park, H. Chung, S. Chang, B. Cho, et al., "A 20nm 1.8 V 8Gb PRAM with 40MB/s program bandwidth," In Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012, pp. 46-48, February 2012.