Utilizing PCM for Energy Optimization in Embedded Systems

Zili Shao*, Yongpan Liu†, Yiran Chen‡, and Tao Li§
*Department of Computing, The Hong Kong Polytechnic University, cszlshao@comp.polyu.edu.hk
†Department of Electronic Engineering, Tsinghua University, ypliu@tsinghua.edu.cn
‡Department of Electrical and Computer Science, University of Pittsburgh, yic52@pitt.edu
§Department of Electrical and Computer Science, University of Florida, taoli@ece.ufl.edu

***Abstract*—Due to its high density, bit alterability, and low standby power, phase change memory (PCM) is considered as a promising DRAM alternative. In embedded systems, especially battery-driven mobile devices, energy is one of the most important performance metrics. Therefore, it becomes an interesting problem of utilizing PCM for energy optimization in embedded systems. While recently there have been extensive studies on PCM, energy optimization with PCM in embedded systems has not been fully addressed. In this paper, we present a hybrid memory system architecture in which PCM is used to replace DRAM as much as possible so the system energy can be reduced by utilizing the lower standby power of PCM. However, to achieve this, system-level software optimization techniques are required in order to solve problems caused by the three disadvantages of PCM: namely, long write latency, large write energy and limited write endurance. We propose an optimal static data allocation scheme to solve a simplified problem, and discuss how to extend this to solve more complex problems. We also present emerging research issues in compiler optimization, real-time task scheduling and operating systems when utilizing PCM for energy optimization in embedded systems.**

***Keywords*-PCM (Phase Change Memory), Energy, Embedded Systems.**

I. Introduction

Energy optimization is vitally important in embedded systems. As main memory is becoming the major energy contributor to the total energy [1], [2], reducing the energy consumption of main memory becomes critical for energy saving. Phase-change memory (PCM), as an emerging non-volatile memory technique [3], [4], [5], shows a lot of promise for energy optimization in embedded systems. As shown in Table I, compared with DRAM, PCM is much better in the idle power; however, it has larger write energy, longer write latency and less write endurance. Therefore, a hybrid main memory with both DRAM and PCM should be utilized to fully exploit their advantages and simultaneously conceal their constraints for energy optimization in embedded systems.

Hybrid PCM/DRAM main memory architectures have been proposed in the previous work [7], [8]. In these studies, both PCM and DRAM are exposed to the Operating System (OS), and the OS can transfer pages between PCM and DRAM to improve PCM lifetime. Improving PCM lifetime, however, cannot optimize energy. On the contrary, it may increase the energy consumption by introducing extra writes with page transfer. There have been several power management techniques for the hybrid PCM/DRAM main memory. In [9], PCM has been utilized to reduce DRAM refreshes in the hybrid PCM/DRAM main memory. Several compiler optimization techniques have been proposed to reduce writes [10], [11], [12], [13] and energy [14] in the hybrid DRAM/NVM(Non-Volatile Memory) main memory. However, these techniques focus on either specific aspects or specific applications. Indeed, a more general framework is needed in order to fully explore PCM and DRAM for energy optimization in embedded systems.

Table I
A Comparison of PCM with DRAM and NAND flash memory [6].

Attributes	DRAM	PCM	NAND
Non-Volatile	No	Yes	Yes
Erase Required	Bit	Bit	Block
Software	Simple	Simple	Complex
Power	$\sim$W/GB	100$\rightarrow$500mW/die	$\sim$100mW/die
Write Bandwidth	$\sim$GB/s	1$\rightarrow$100+ MB/s/die	10$\rightarrow$100MB/s/die
Write Latency	$\sim$20-50ns	$\sim$1μs	$\sim$100μs
Write Energy	$\sim$0.1nJ/b	$<$1nJ/b	0.1-1nJ/b
Read Latency	50ns	50-100ns	10-25μs
Read Energy	$\sim$0.1nJ/b	$\ll$1nJ/b	$\ll$1nJ/b
idle Power	$\sim$W/GB	$\ll$0.1W	$\ll$0.1W
Endurance	∞	10^8	$10^5 \rightarrow 10^4$
Data Retention	ms	Not f (cycles)	f (cycles)

In this paper, we target at a hybrid memory system architecture in which PCM is used to replace DRAM as much as possible so the system energy can be reduced by utilizing the lower standby power of PCM. As embedded systems are mostly application-oriented, such an application-specific hybrid memory architecture can further help reduce the system energy. To achieve this, we need to solve several problems that have not been addressed in the previous work. One important problem is to determine the best sizes of PCM and DRAM for a specific application so the energy can be minimized with the minimum time overhead. How to allocate data and programs in PCM and DRAM is also important for energy optimization, since different allocations may introduce different write traffic to PCM and DRAM. Furthermore, as many embedded systems are with real-time computing constraints, we need to consider whether timing constraints can be met as utilizing PCM for energy optimization may introduce time overhead.

As several optimization problems involve, in this paper, as a first step, we solve a simplified problem in which we assume that data are statically allocated into PCM without further movement, and there is no time constraint. This is the case for simple embedded systems that do not need OS (Operating Systems) or only require simple real-time kernels. We show the problem can be solved optimally, and the sizes of PCM and DRAM can be obtained. We discuss how to extend this scheme to solve more complex problems. Emerging research issues in compiler optimization, OS, and real-time task scheduling are presented for energy optimization with the application-specific hybrid PCM/DRAM main memory in embedded systems.

The rest of the paper is organized as follows. The previous work is introduced in Section II. Section III presents the background. Our optimal static data allocation scheme is proposed in Section IV. In Section V, we discuss emerging research issues in system-level software optimization. The conclusion is given in Section VI.

II. Related work

PCM management has been intensively studied in the general-purpose computing field. In [15], a memory system with *on-chip DRAM cache and PCM main memory* is proposed, in which the DRAM cache that is invisible to operating systems (OS) is directly managed by the memory controller. In [7], [8], a *hybrid PCM/DRAM* memory system is proposed where both PCM and DRAM are exposed to the OS and the OS can transfer pages between PCM and DRAM for improving PCM lifetime. Various techniques have been proposed to extend PCM life time via write reduction [1], [16], [17], [18], [19] and wear leveling [1], [20], [21], defend against malicious attacks [20], [22], [21], [23], [24], [25], prevent chip failure caused by failed PCM cells [26], [27], [28], [29], [30], and reduce the influence caused by long write latency [31], [32], [33], [34]. In the above work, however, energy optimization has not been fully addressed.

There have been studies for utilizing PCM in embedded systems. At the architecture level, a hybrid PCM/SRAM scratched-pad memory is proposed in [35], and a hybrid PCM/NAND flash storage system is proposed in [36]. At the software level, a file system is designed to support to store metadata of file systems in PCM [37]. In [38], a write-activity-aware page table management scheme is proposed to reduce writes by modifying the page table initialization and page frame allocation modules in Linux kernel. In [39], PCM-FTL, a write-activity-aware NAND flash management scheme, is proposed to enhance the PCM endurance in embedded systems with PCM/NAND flash. The above techniques focus on either reducing writes or improving endurance, and cannot be directly applied in energy optimization.

III. Background

In this section, we first introduce PCM and then present the system architecture and open problems to utilize the application-specific hybrid PCM/DRAM main memory for energy optimization in embedded systems.

A. PCM

PCM is one type of non-volatile memory to store information based on chalcogenide (the phase-change material) [5]. Figure 1 shows a typical PCM cell that contains a layer of chalcogenide such as Ge2Sb2Te5 (GST) and two electrodes. GST has two states, crystalline and amorphous, and the resistivity of the amorphous state is 3-4 orders of magnitude higher than that of crystalline state.

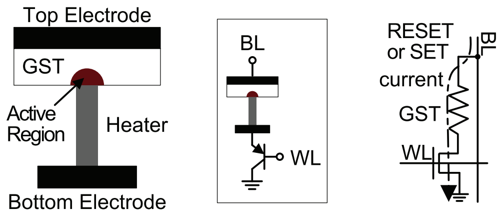

Figure 1. The structure of a PCM cell [5].

By injecting current into the heater, we can thermally induce the phase change of GST. As shown in Figure 2, in the SET operation, with a moderate current pulse, GST is heated to a temperature between the crystallization and melting temperatures (between 300°C and 600°C) for a relatively long period, which induces GST into a crystalline state (logic "1"); in the RESET operation, with a high current, GST is heated to a temperature above the melting temperature for a short period, which induces GST into an amorphous state (logic "0").

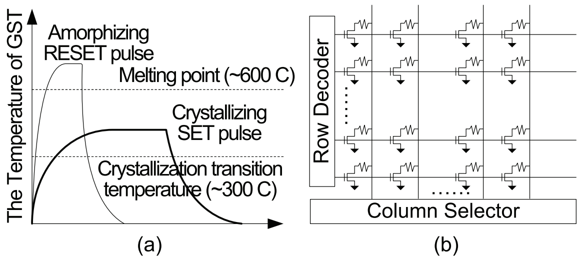

Figure 2. (a) RESET/SET operations; (b) PCM cell array [5].

In a PCM chip, cells are organized in a 2D array (Figure 2). Although the organization is similar to that of DRAM, PCM has some specific design issues [4].

B. System Architecture and Problems

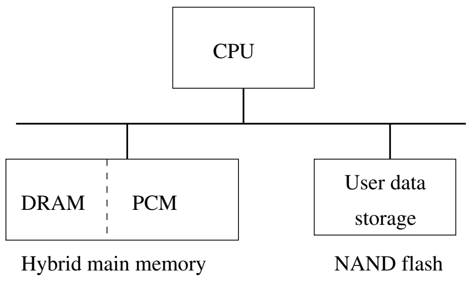

Figure 3. The system architecture with the application-specific hybrid PCM/DRAM main memory in embedded systems.

Figure 3 shows the system architecture with the hybrid PCM/DRAM main memory in embedded systems. NAND flash is used as the second storage to store user data. The hybrid PCM/DRAM memory serves as the main memory. Different from DRAM-based main memory, this hybrid main memory can store code and even user data by utilizing the non-volatility of PCM.

In embedded systems, hardware and software can be specially designed and optimized for specific applications. Thus, to optimize energy for an application, we need to solve various problems. For example, what are the best sizes of PCM and DRAM, how to statically allocate data and program, and how to migrate data between PCM and DRAM. A lot of new optimization problems arise and various system-level optimization techniques need to be developed in order to fully explore this application-specific PCM/DRAM main memory for energy optimization in embedded systems.

IV. Optimal Static Data Allocation

As shown above, several optimization problems are involved in order to fully explore this architecture. Thus, as a first step, we solve a simplified problem, that is, *if there is no time constraint, and data are statically allocated in PCM without further movement, what are the best sizes of PCM and DRAM?* Next, we first present the energy cost model and propose the optimal static data allocation scheme, and then discuss how to extend our scheme to solve more complex problems.

A. Energy Cost Model

The energy consumption of a DRAM chip can be divided into the standby energy, the refresh energy and the access energy, where the standby energy represents the energy consumed when the memory is idle, the refresh energy represents the energy consumed for supplying refresh cycles for data retention, and the access energy represents the energy consumption caused by reads and writes. As PCM is non-volatile, it does not need to be refreshed. So the energy consumption of a PCM chip can be divided into two parts: the standby energy and the access energy. In either PCM or DRAM, the standby energy is mainly determined by the memory and is not related to data accesses. Therefore, given a data or program item, in DRAM, its energy comes from accesses and refreshes; in PCM, it comes from accesses. Based on this, we define the energy cost of item i as follows.

$$Cost(i) = E_{Ref}(i) + E_{DAccess}(i) - E_{PAccess}(i) \quad (1)$$

In Equation 1, $Cost(i)$ represents the energy difference between putting item i into DRAM and PCM; $E_{Ref}(i)$ and $E_{DAccess}(i)$ represents the refresh energy and access energy, respectively, when item i is put in DRAM; $E_{PAccess}$ represent the access energy when i is in PRAM. Let $N_{RD}(i)$ and $N_{WR}(i)$ be the read and write numbers for item i. Then we can obtain $E_{DAccess}(i)$ and $E_{PAccess}(i)$ by the following equations:

$$E_{DAccess}(i) = \sum_{j=1}^{N_{RD}} E_{DRAM}^{RD(j)}(i) + \sum_{j=1}^{N_{WR}} E_{DRAM}^{WR(j)}(i)$$

$$E_{PAccess}(i) = \sum_{j=1}^{N_{RD}} E_{PCM}^{RD(j)}(i) + \sum_{j=1}^{N_{WR}} E_{PCM}^{WR(j)}(i)$$

Here, $E_{DRAM}^{RD(j)}(i)$ and $E_{DRAM}^{WR(j)}(i)$ represent the read DRAM energy and the write DRAM energy of item i caused by jth read and write, respectively.

Put the above equations into Equation 1, and we have the the following:

$$Cost(i) = E_{Ref}(i) + \sum_{j=1}^{N_{RD}} (E_{DRAM}^{RD(j)}(i) - E_{PCM}^{RD(j)}(i)) + \sum_{j=1}^{N_{WR}} (E_{DRAM}^{WR(j)}(i) - E_{PCM}^{WR(j)}(i)) \quad (2)$$

Equation 2 shows that we may not get any energy benefit by putting an item into PCM if it will be written too many times, since PCM write energy is bigger than DRAM write energy. Furthermore, the PCM cells stored an item may be worn out with too many writes.

B. Optimal Data Allocation and PCM/DRAM Sizes

There are two cases in data allocation. First, let us assume that there is no limit on the PCM size. Then item i should be put into PCM if $Cost(i) > 0$ and its write number is less than the allowed write endurance. In this way, we can obtain the data allocation and the best sizes of PCM and DRAM.

If the PCM size is fixed and cannot hold all the above items, a subset of the items should be selected to be put into PCM. In this case, the data allocation problem becomes 0-1 knapsack problem. Let W be the PCM size, and Size(i) be the size of item i. Then the problem becomes:

Given a set of items $I = \{1, 2, \cdots, N\}$ in which for each item $i \in I$, $Cost(i) \in Z^+$ is its value and $Size(i) \in Z^+$ is its weight, and two given positive integers H and W, is there a subset S of I such that $\sum_{i \in S} Cost(i) \geq H$ and $\sum_{i \in S} Size(i) \leq W$?

The knapsack problem is NP-complete [40]. It can be optimally solve by dynamic programming in pseudo polynomial time. Efficient heuristic algorithms should be developed to handle the cases when there are too many items or the PCM size is very big.

C. Extension

The above optimal static data allocation scheme can be extended to solve more complex problems. When time overheads introduced by PCM need are considered, we can perform a procedure to iteratively adjust data allocation and test time constraints. How to efficiently and effectively achieve this is an interesting open problem.

V. Research Issues

System-level software optimization techniques are required in order to fully exploit the potential features of the application-specific hybrid PCM/DRAM main memory architecture for energy optimization. Besides long write latency, large write energy and limited write endurance, PCM introduces large write power consumption. To solve problems caused by these disadvantages, various research issues need to be addressed at instruction-level, task-level and OS-level. Some sample open problems are listed below.

A. Compiler Optimization

Compiler optimization techniques are needed to implement static allocation. The static allocation are particularly useful for simple embedded systems in which the static memory layout is good enough for executing single process or multiple processes.

In order to obtain read and write behaviors and time performances of data and programs, we need to perform *specific program analysis statically and dynamically*. The static analysis may contain data flow analysis, control flow analysis and other classical compiler analysis so as to collect necessary information for memory accesses. The dynamic analysis is needed in order to obtain time and memory behaviors with profiling techniques. For a process, we need the information of the numbers and time/energy performances of reads and writes for the code, data, heap and stack. Then we need an integrated technique to combine the results of the static analysis and dynamic analysis, and generate the estimate information needed in the static allocation such as in Equation 2. How to efficiently perform this estimate and how to evaluate if it is accurate are interesting open problems.

PCM brings some new hardware features which can be utilized by compiler optimization. For example, in PCM writes, there are big differences in both time and energy between a row buffer write and a row array write [7], [8], [15]. It is an interesting open problem of how to maximally utilize a row buffer before a row array write is triggered? Different data layouts may generate different results. Different instruction sequences also cause different behaviors. These should be studied, and they may help reduce PCM writes.

Some existing compiler techniques may need to revised to be "PCM-write-aware" in order to solve problems caused by the disadvantages of PCM writes. For example, in "PCM-write-aware instruction scheduling", PCM writes may be scheduled into positions so as to introduce minimum time delay or minimum power variation. Register spilling may need to be "PCM-write-aware", by which it should use DRAM first in order to reduce PCM writes. It is also possible to combine with DVS (Dynamic Voltage Scaling) to implement a joint DVS and PCM write scheduling to optimize power variation.

Registers, cache, and SPM (Scratch-Pad Memory) should be fully utilized to reduce PCM writes. For example, in order to keep hot data from PCM in cache lines, we can add extra "read". Then we need to study where to insert these "read" and what are tradeoffs. To utilize SPM, instead of using frequency-based allocation policies, we may need to only focus on "PCM write frequency". Based on this, new SPM allocation and eviction schemes can be developed.

B. OS optimization

The static allocation is good enough for simple embedded systems, in which it may not need OS or only requires a simple real-time kernel. For high-end embedded systems such as smartphones, we need OS support to explore the hybrid memory for energy optimization.

Most previous work focuses on wear leveling by migrating pages with frequent-writes from PCM to DRAM, and vice versa. Page migration introduces time and energy overheads. To reduce these overheads, we should provide interfaces for programmers, and let them select correct memory types for specific applications. For example, PCM is for read most data and DRAM for write most data. As programmers understand applications better than OS, this selection should help reduce the number of migrations.

PCM does not need to be refreshed. This can be utilized to optimize energy. For example, a DRAM bank can be turned off for a period of time by storing its data into a PCM area with a small DRAM buffer. By using the small DRAM buffer to cache data, with good buffer management policies, it may not or introduce very few writes in the PCM area. If OS can help group pages that are not write intensive into one bank, this may help save energy.

C. Real-time Task Scheduling

Many embedded systems have real-time constraints. The tradeoff between the energy gain and the time overhead needs to be considered in real-time task scheduling. Several open problems should be studied from energy optimization and WCET (Worst-Case Execution Time) Analysis.

In Energy optimization, by combining the schedulability analysis with the static allocation, scheduling algorithms can be studied based on various task models such as single task or multiple tasks, periodic or aperiodic, dependent or independent, preemptive or non-preemptive, online or offline, and single-processor or multi-processor. It is also interesting to study dynamic allocation when scheduling tasks, where one PCM area can be used by different tasks at different times.

Introducing PCM into the system architecture makes WCET analysis more difficult. Particularly, in PCM writes, there is big time difference between a row buffer write and a row array write. This new feature should be handled in WCET analysis. In an environment with dynamic allocation where there may be data migration between PCM and DRAM, it is an open problem of how to perform WCET analysis.

VI. Conclusion

In this paper, we studied how to utilize PCM for energy optimization in embedded systems. We presented an application-specific hybrid memory system architecture in which PCM is used to replace DRAM as much as possible. We proposed an optimal static data allocation scheme to solve a simplified problem, and discussed how to extend this to solve more complex problems. We presented emerging research issues in compiler optimization, real-time task scheduling and operating systems when utilizing PCM for energy optimization in embedded systems.

VII. Acknowledgement

The work described in this paper is partially supported by the grants from the Innovation and Technology Support Programme of Innovation and Technology Fund of the Hong Kong Special Administrative Region, China (ITS/082/10), and the Hong Kong Polytechnic University (G-YK24).

References

[1] P. Zhou, B. Zhao, J. Yang, and Y. Zhang, "A durable and energy efficient main memory using phase change memory technology," in *Proceedings of the 36th annual international symposium on Computer architecture (ISCA '09)*. New York, NY, USA: ACM, 2009, pp. 14 –23. [Online]. Available: http://doi.acm.org/10.1145/1555754.1555759

[2] W. Tian, J. Li, Y. Zhao, C. J. Xue, M. Li, and E. Chen, "Optimal task allocation on non-volatile memory based hybrid main memory," in *Proceedings of the 2011 ACM Symposium on Research in Applied Computation*, ser. RACS '11. New York, NY, USA: ACM, 2011, pp. 1–6. [Online]. Available: http://doi.acm.org/10.1145/2103380.2103382

[3] Y. Xie, "Modeling, architecture, and applications for emerging memory technologies," *IEEE Design Test of Computers*, vol. 28, no. 1, pp. 44 –51, Jan.-Feb. 2011.

[4] B. Lee, P. Zhou, J. Yang, Y. Zhang, B. Zhao, E. Ipek, O. Mutlu, and D. Burger, "Phase-change technology and the future of main memory," *IEEE Micro*, vol. 30, no. 1, p. 143, Jan.-Feb. 2010.

[5] C. Xue, Y. Zhang, Y. Chen, G. Sun, J. Yang, and H. Li, "Emerging non-volatile memories: Opportunities and challenges," in *2011 Proceedings of the 9th International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS '11),*, Oct. 2011, pp. 325 –334.

[6] S. Eilert, M. Leinwander, and G. Crisenza, "Phase Change Memory: A new memory enables new memory usage models," in *IEEE International Memory Workshop (IMW '09)*, May 2009, pp. 1 –2.

[7] W. Zhang and T. Li, "Exploring Phase Change Memory and 3D die-stacking for power/thermal friendly, fast and durable memory architectures," in *18th International Conference on Parallel Architectures and Compilation Techniques (PACT '09)*, Sept. 2009, pp. 101 –112.

[8] G. Dhiman, R. Ayoub, and T. Rosing, "PDRAM: A hybrid PRAM and DRAM main memory system," in *46th ACM/IEEE Design Automation Conference (DAC '09)*, July 2009, pp. 664 –669.

[9] H. Park, S. Yoo, and S. Lee, "Power management of hybrid dram/pram-based main memory," in *Design Automation Conference (DAC), 2011 48th ACM/EDAC/IEEE*, june 2011, pp. 59 –64.

[10] J. Hu, C. J. Xue, W.-C. Tseng, Y. He, M. Qiu, and E. H.-M. Sha, "Reducing write activities on non-volatile memories in embedded CMPs via data migration and recomputation," in *Proceedings of the 47th Design Automation Conference (DAC '10)*. New York, NY, USA: ACM, 2010, pp. 350–355. [Online]. Available: http://doi.acm.org/10.1145/1837274.1837363

[11] J. Hu, W.-C. Tseng, C. Xue, Q. Zhuge, Y. Zhao, and E.-M. Sha, "Write activity minimization for nonvolatile main memory via scheduling and recomputation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 4, pp. 584 –592, April 2011.

[12] J. Hu, C. J. Xue, W.-C. Tseng, Y. He, M. Qiu, and E. H.-M. Sha, "Reducing write activities on non-volatile memories in embedded CMPs via data migration and recomputation," in *2010 47th ACM/IEEE Design Automation Conference (DAC '10)*, June 2010, pp. 350 –355.

[13] J. Hu, C. Xue, W.-C. Tseng, Q. Zhuge, and E.-M. Sha, "Minimizing write activities to non-volatile memory via scheduling and recomputation," in *IEEE 8th Symposium on Application Specific Processors (SASP '10)*, June 2010, pp. 101 –106.

[14] T. Liu, Y. Zhao, C. Xue, and M. Li, "Power-aware variable partitioning for DSPs with hybrid PRAM and DRAM main memory," in *48th ACM/EDAC/IEEE Design Automation Conference (DAC '11)*, June 2011, pp. 405 –410.

[15] 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 Computer architecture (ISCA '09)*. New York, NY, USA: ACM, 2009, pp. 24–33. [Online]. Available: http://doi.acm.org/10.1145/1555754.1555760

[16] G. Sun, D. Niu, J. Ouyang, and Y. Xie, "A frequent-value based PRAM memory architecture," in *2011 16th Asia and South Pacific Design Automation Conference (ASP-DAC '11)*, Jan. 2011, pp. 211 –216.

[17] W. Zhang and T. Li, "Characterizing and mitigating the impact of process variations on phase change based memory systems," in *42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO '09)*, Dec. 2009, pp. 2 –13.

[18] S. Cho and H. Lee, "Flip-N-Write: A simple deterministic technique to improve PRAM write performance, energy and endurance," in *42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO '09)*, Dec. 2009, pp. 347 –357.

[19] A. Ferreira, M. Zhou, S. Bock, B. Childers, R. Melhem, and D. Mosse, "Increasing PCM main memory lifetime," in *Design, Automation Test in Europe Conference Exhibition (DATE '10)*, March 2010, pp. 914 –919.

[20] M. 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 *42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO '09)*, Dec. 2009, pp. 14 – 23.

[21] 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," in *Proceedings of the 37th annual international symposium on Computer architecture (ISCA '10)*. New York, NY, USA: ACM, 2010, pp. 383–394. [Online]. Available: http://doi.acm.org/10.1145/1815961.1816014

[22] M. Qureshi, A. Seznec, L. Lastras, and M. Franceschini, "Practical and secure PCM systems by online detection of malicious write streams," in *IEEE 17th International Symposium on High Performance Computer Architecture (HPCA '11)*, Feb. 2011, pp. 478 –489.

[23] A. Seznec, "A Phase Change Memory as a secure main memory," *Computer Architecture Letters*, vol. 9, no. 1, pp. 5 –8, Jan. 2010.

[24] S. Chhabra and Y. Solihin, "i-NVMM: a secure non-volatile main memory system with incremental encryption," in *Proceedings of the 38th annual international symposium on Computer architecture (ISCA '11)*. New York, NY, USA: ACM, 2011, pp. 177–188. [Online]. Available: http://doi.acm.org/10.1145/2000064.2000086

[25] J. Kong and H. Zhou, "Improving privacy and lifetime of PCM-based main memory," in *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN '10)*, 28 2010-july 1 2010, pp. 333 –342.

[26] S. Schechter, G. H. Loh, K. Straus, and D. Burger, "Use ECP, not ECC, for hard failures in resistive memories," in *Proceedings of the 37th annual international symposium on Computer architecture (ISCA '10)*. New York, NY, USA: ACM, 2010, pp. 141–152. [Online]. Available: http://doi.acm.org/10.1145/1815961.1815980

[27] D. H. Yoon, N. Muralimanohar, J. Chang, P. Ranganathan, N. Jouppi, and M. Erez, "FREE-p: Protecting non-volatile memory against both hard and soft errors," in *IEEE 17th International Symposium on High Performance Computer Architecture (HPCA '11)*, Feb. 2011, pp. 466 –477.

[28] L. Jiang, Y. Du, Y. Zhang, B. Childers, and J. Yang, "LLS: Cooperative integration of wear-leveling and salvaging for PCM main memory," in *IEEE/IFIP 41st International Conference on Dependable Systems Networks (DSN '11)*, Jun. 2011, pp. 221 –232.

[29] E. Ipek, J. Condit, E. B. Nightingale, D. Burger, and T. Moscibroda, "Dynamically replicated memory: building reliable systems from nanoscale resistive memories," in *Proceedings of the fifteenth edition of ASPLOS on Architectural support for programming languages and operating systems (ASPLOS '10)*. New York, NY, USA: ACM, 2010, pp. 3–14. [Online]. Available: http://doi.acm.org/10.1145/1736020.1736023

[30] J. Chen, Z. Winter, G. Venkataramani, and H. Huang, "rPRAM: Exploring redundancy techniques to improve lifetime of PCM-based main memory," in *International Conference on Parallel Architectures and Compilation Techniques (PACT '11)*, Oct. 2011, pp. 201 –202.

[31] M. Qureshi, M. Franceschini, and L. Lastras-Montano, "Improving read performance of Phase Change Memories via write cancellation and write pausing," in *IEEE 16th International Symposium on High Performance Computer Architecture (HPCA '10)*, Jan. 2010, pp. 1 –11.

[32] G. Sun, X. Dong, Y. Xie, J. Li, and Y. Chen, "A novel architecture of the 3D stacked MRAM L2 cache for CMPs," in *IEEE 15th International Symposium on High Performance Computer Architecture (HPCA '09)*, Feb. 2009, pp. 239 –249.

[33] P. Zhou, Y. Du, Y. Zhang, and J. Yang, "Fine-grained QoS scheduling for PCM-based main memory systems," in *IEEE International Symposium on Parallel Distributed Processing (IPDPS '10)*, April 2010, pp. 1 –12.

[34] M. Joshi, W. Zhang, and T. Li, "Mercury: A fast and energy-efficient multi-level cell based Phase Change Memory system," in *IEEE 17th International Symposium on High Performance Computer Architecture (HPCA '11)*, Feb. 2011, pp. 345 –356.

[35] J. Hu, C. Xue, Q. Zhuge, W.-C. Tseng, and E.-M. Sha, "Towards energy efficient hybrid on-chip Scratch Pad Memory with non-volatile memory," in *Design, Automation Test in Europe Conference Exhibition (DATE '11)*, March 2011, pp. 1 –6.

[36] J. K. Kim, H. G. Lee, S. Choi, and K. I. Bahng, "A PRAM and NAND flash hybrid architecture for high-performance embedded storage subsystems," in *Proceedings of the 8th ACM international conference on Embedded software (EMSOFT '08)*. New York, NY, USA: ACM, 2008, pp. 31–40. [Online]. Available: http://doi.acm.org/10.1145/1450058.1450064

[37] Y. Park and K. H. Park, "High-performance scalable flash file system using virtual metadata storage with Phase-Change RAM," *IEEE Transactions on Computers*, vol. 60, no. 3, pp. 321 –334, March 2011.

[38] T. Wang, D. Liu, Z. Shao, and C. Yang, "Write-activity-aware page table management for PCM-based embedded systems," in *17th Asia and South Pacific Design Automation Conference (ASP-DAC '12)*, Jan. 30 -Feb. 2 2012, pp. 317 –322.

[39] D. Liu, T. Wang, Y. Wang, Z. Qin, and Z. Shao, "PCM-FTL: A write-activity-aware NAND flash memory management scheme for PCM-based embedded systems," in *IEEE 32nd Real-Time Systems Symposium (RTSS '11)*, Nov. 29 -Dec. 2 2011, pp. 357 –366.

[40] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.