國立臺灣大學電機資訊學院資訊工程學系

碩士論文

Department of Computer Science and Information Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

揮發或非揮發？可轉換式相變化記憶體為基礎之記憶體系統

Volatile or Non-Volatile? A Morphable PCM-Based Memory System

沈德鈺

De-Yu Shen

指導教授：楊佳玲 博士

Advisor: Chia-Lin Yang, Ph.D.

中華民國 101 年 7 月

July, 2012

致謝

首先我要感謝我的指導教授-楊佳玲老師。感謝老師的悉心指導，無論是研究方面，對於研究的內容與困難，引導我思考最核心的問題，並以嚴謹的態度點出研究的方向，或是在生活工作方面，與學生分享人生上的經驗，並給予幫助，使我能有更充分的準備來完成本論文。此外，也感謝口試委員王成淵博士、徐慰中老師與蘇雅韻老師提供寶貴的意見，使本篇論文能更加完整。

研究所期間，要特別謝謝呂仁碩學長。學長總是在我研究遇到困難時給予專業的建議，也在研究機制的實作上、論文寫作上給予最大的幫助。還有要謝謝林仲祥學長，能在投影片的製作，計畫報告的撰寫，乃至於研究上的困難給予意見。此外，也要謝謝實驗室的學長和同學，謝謝業俊學長、柏翰學長、靖淳學長、耀慶、呂敏、小剛、希哥、Chopper、振銘的陪伴，互相打氣、打鬧、開玩笑，使這研究生生活不只多了一點而是多了很多的樂趣。

最後，我還要感謝我的家人，感謝你們無怨無悔的付出，感謝你們的包容。在我就學期間，提供我最有力的支持，提供一個溫暖的家，讓我可以專注於學業上。我會好好記住你們的用心，在現在與未來的日子裡，積極的面對各種挑戰。

國立臺灣大學碩士學位論文
口試委員會審定書

揮發或非揮發？ 可轉換式相變化記憶體為基礎之記憶體系統

Volatile or Non-Volatile? A Morphable PCM-Based Memory System

本論文係沈德鈺君（學號 R99922118）在國立臺灣大學資訊工程學系完成之碩士學位論文，於民國 101 年 7 月 25 日承下列考試委員審查通過及口試及格，特此證明

口試委員：

[illegible]
（指導教授）

徐慰中

蘇雅韻

王成淵

系 主 任　許永真

中文摘要

相變化記憶體 (Phase-Change Memory) 為目前新興且廣泛引人注意的記憶體。其具有位元組可編址性 (byte-addressability) 、高存儲密度與與非揮發性等特質，將能夠大大的改善目前對於記憶體階層架構的設計。舉例而言，相變化記憶體之高存儲密度與位元組可編址性的特性，可用於建造大型之記憶體系統，用滿足日趨成長之記憶體容量需求。而相變化記憶體之非揮發性，則可使記憶體能夠如同磁碟一般存取資料或檔案，用以彌補硬碟存取速度較慢的問題。

然而，目前主記憶體並不需要儲存非揮發之資料。當多層式儲存 (multi level cell, MLC) 相變化記憶體僅作為主記憶體使用時，其卻因需要維持記憶體之非揮發性而付出昂貴之寫入代價。因此在本篇論文中，我們提出了一種新的相變化記憶體系統的設計概念，相變化記憶體可以同時支援兩種寫入模式以因應不同的使用方式：傳統之非揮發之寫入模式可用於寫入需要長時間保存之資料與檔案；而快而揮發之寫入模式則可適用於相變化記憶體做為主記憶體時，不需要儲存非揮發資料的情況，藉以增進記憶體系統之效能。

揮發之記憶體單元需要定期的刷新以保持資料的正確性，而傳統的相變化記憶體設計者認為此刷新動作將造成巨大的能量消耗，並可能危及系統之效能。為驗證提出的設計概念，我們實做了以 1000 秒刷新區間為主的主記憶體系統。分析結果顯示刷新動作所產生之能量消耗可控制在一微小的範圍。實驗結果顯示我們所提出之快而揮發之寫入模式能夠提供系統平均 21.5%的效能改善。

關鍵字 - 多層式儲存相變化記憶體, 非揮發性, 資料維持時間

Abstract

Phase-change memory (PCM) is an emerging and appealing technology nowadays. PCM is byte-addressable, high-density, and non-volatile, so-called “universal” memory, which could revolute the memory hierarchy design. The high density and byte-addressability of PCM enable architects to build large-scale memory systems; many works have demonstrated the benefits of supplementing DRAM with PCM in the main memory. To take advantage of its non-volatility, instantly turning on/off computers or maintaining data and files persistent in main memory without accessing disks are also proposed.

The byte addressability and non-volatility of PCM blur the line between memory and storage, and call for a rethinking on architectural design. When PCM is used as part of the main memory, non-volatility is unnecessary. Relaxing non-volatility could significantly reduce the write latency of multi level cell(MLC) PCM. Therefore, in this paper, we advocate a new design concept for PCM-based memory system. That is, PCM cells can be written in two different modes, fast-yet-volatile writes for main-memory usage and conventional non-volatile writes for persistency purpose. Volatile memory cells require refreshing to ensure data are valid over time. Conventional wisdom of PCM-based design thinks PCM is power-hungry and refreshing incurs considerable power overhead. On the contrary, we demonstrate that a refresh interval of 1000 seconds leads to insignificant power overhead. Simulation results show that the proposed techniques can improve overall system performance by 21.5% on average.

***Keywords* -** MLC PCM, non-volatility, data retention time

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Recently, several non-volatile memories (NVMs) are emerging in memory hierarchy design, including phase-change memory (PCM), spin-transfer torque RAM (STT-RAM), memristor, and racetrack memory. Among them, PCM is one of the most mature and promising technologies. PCM exploits the property of chalcogenide alloy of germanium, antimony and tellurium (GeSbTe), a kind of phase-change material, to store data in the form of different crystallization states and resistances. Vendors such as IBM, Intel, Macronix, Numonyx, and Samsung have all announced their PCM prototypes or samples.

The byte-addressability and non-volatility of PCM blur the line between memory and storage. Recently, many researches propose to adopt PCM to supplement DRAM in the main-memory hierarchy [1–3]. PCM has an advantage in density due to its excellent scalability and the multi-level cell (MLC) technology. Therefore, it enables architecting a large main memory system for future CMPs' needs. Currently, 20-nm PCM devices have been fabricated and tested [4, 5]. Devices down to 8 nm are also projected [6]. Another strength of PCM is it can store multiple bits in a cell using multiple resistance levels. Previous works have shown PCM prototypes with two bits to four bits per cell [7]. Up to 10-year retention is also demonstrated available even if 2-bit MLC is adopted [8–10]. To take advantage of the non-volatility feature, several researchers propose to store files and persistent objects in the PCM memory and access them directly using load/store instructions [11–13]. Another interesting application of non-volatility of PCM is instant on-off. With PCM, since most program context can be preserved in the non-volatile main

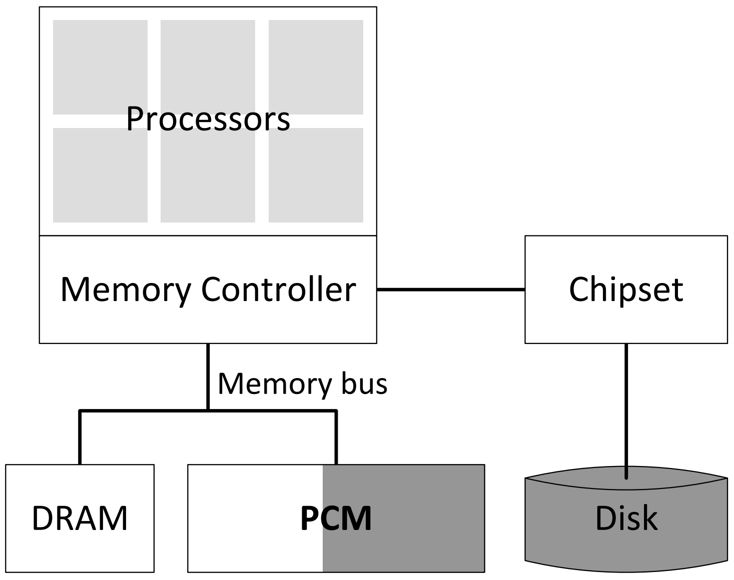


Figure 1.1: PCM in the future computer architecture

memory, a system can be powered on/off with minimum overheads. This enables turning on/off machines on demand in a fine-grained time scale and improve the energy proportionality.

Therefore, we can envision that future PCM is connected to the memory bus, and parts of the memory cells are used as the main memory, and the other parts are used to store persistent objects or files, as illustrated in Figure 1.1. Current studies treat PCM as non-volatile oblivious to its roles in the system (i.e., main memory or storage). However, maintaining non-volatility does not come for free. To achieve a longer retention guarantee, one requires PCM cells to meet a stricter constraint on the resistance distribution, which lengthens the write procedure.

In this thesis, we propose a new design concept for PCM-based memory system. PCM cells can be written in two different modes, fast-yet-volatile writes for main-memory usage or conventional non-volatile writes for persistency purpose. To realize the proposed dual write-mode system, an interface is required to convey memory access types to the memory chips, and volatile memory cells require refreshing to ensure data are valid over time. Conventional PCM-based design takes advantage of PCM's non-volatility to avoid the need of refreshing which incurs both performance and energy overheads. We model MLC PCM and quantitatively analyze the relation between write latency and retention.

We show that volatile writes can achieve up to $1.46\times$ latency speedup if the designed retention is 1000 seconds. Through full-system simulation, we show that refreshing PCM at a 1000-second period incurs insignificant degradation on energy, performance, and lifetime. Experimental results show that a system with 1000-second retention can achieve 22% of IPC improvement.

The rest of this paper begins with some background of PCM. Chapter 3 describes the MLC PCM model. Chapter 4 describes our proposed morphable PCM-based memory system. Chapter 5 evaluates our proposed design. Chapter 6 describes related works and Chapter 7 concludes.

Chapter 2

Background

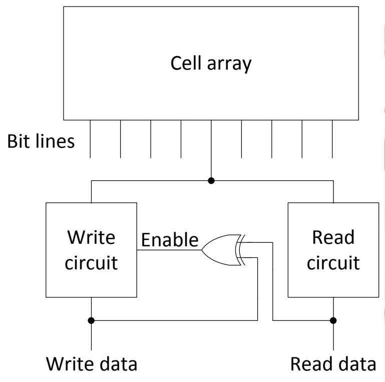


Figure 2.1: Differential write

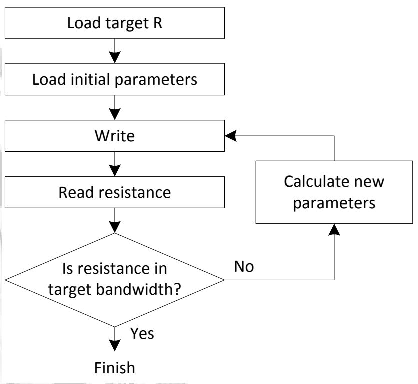


Figure 2.2: Write and verify (WAV) algorithm

Phase-change memory (PCM) is a type of non-volatile memory which exploits the property of chalcogenide alloy of germanium, antimony and tellurium (GeSbTe). PCM can store data by different crystallization states.

The amorphous state has high electrical resistance which represents data 0. And the crystalline state has relative low resistance which represents data 1. Two states can be switched by applying different programming electrical pulses and heating time. For example, the set operation uses small electrical pulse but long time to program cells such that the PCM material changes to crystalline state and has low resistance. On the other hand, if a large electrical pulse but short time programming is applied, the cells would be programmed into amorphous state and have high resistance.

Since the programming pulse hurts the PCM material, the PCM cells have limited write endurance. Typical PCM has 10^6 to 10^8 write cycles before the cell has failed, and it is worse than DRAM cell write endurance (10^{15} write cycle). PCM's failure issue have been studied in several works. Zhou et al. [2] proposed the differential write mechanism. When the data are programmed into the PCM array, only the cells with different value are updated such that the programming pulse and the hurt can be reduced. Figure 2.1 shows the write circuit which implements the differential write. Before the write starts, the circuit reads the data from PCM array and compares them with the written data. By doing so, only the updated cells would be programmed by the write circuit.

The PCM resistance range can be 3 to 5 order of magnitude [7, 14]. Therefore, the large resistance range can be used to represent multiple resistance states and realize the multi-level cell (MLC) PCM. For example, the 2 bits per cell MLC PCM has been proposed in [9], and some researches [7] have shown the possibility of four bits pre cell PCM. By storing multiple bit in a single cells, PCM has ability to provide the high density and to satisfy the high memory capacity demand of applications.

In MLC PCM, due to the tight resistance range of each level, process variation between cells and the target retention requirement, it is hard to use only one programming pulse to program the cells into the target bandwidth. Hence, The studies in [7, 9], have proposed the iterative write-and-verify (WAV) mechanism. Figure 2.2 shows the flow of the WAV programming mechanism. Given the target resistance level, the write circuit decides and initiates the programming parameters first. After each write pulse finishes, the WAV circuit reads the cell resistance and performs verification to check if the cell resistance is located in the target bandwidth. If the target resistance is not achieved, write circuit re-calculates the programming parameters and repeats the WAV iteration. The WAV programming ends when the target resistance is reached.

Chapter 3

Trading Non-Volatility for Improving Write Speed

PCM, like flash, FeRAM, and STT-RAM, is a kind of non-volatile memory (NVM). The non-volatility is an advantage though, it does not come for free: to achieve long data retention, the resistance of PCM cells have to be confined to a tight distribution so that large enough margins are created to tolerate resistance changes over time. This requirement poses difficulty to writes and lengthens write operations. In this section, we model 2-bit MLC PCM to investigate the opportunity of trading non-volatility for write speed. We show that up to $1.46\times$ write speedup is achievable if we set the designed data retention of PCM to 1000 seconds.

3.1 Data Retention vs. Resistance Distribution

Resistance distributions play an important role in determining the data retention of MLC PCM. Due to the property of phase-change material, the resistance of PCM cells tends to increase over time (referred to as *resistance drift*). To guarantee non-volatility, a tight resistance distribution is required because with tighter distribution, it takes longer time for resistance drift to alter the data stored in PCM. For example, Figure 3.1 compares two resistance distributions of 2-bit MLC PCM. Figure 3.1(a) shows a tighter distribution than Figure 3.1(b) does. Consequently, PCM with the former distribution can guarantee longer data retention than the later.

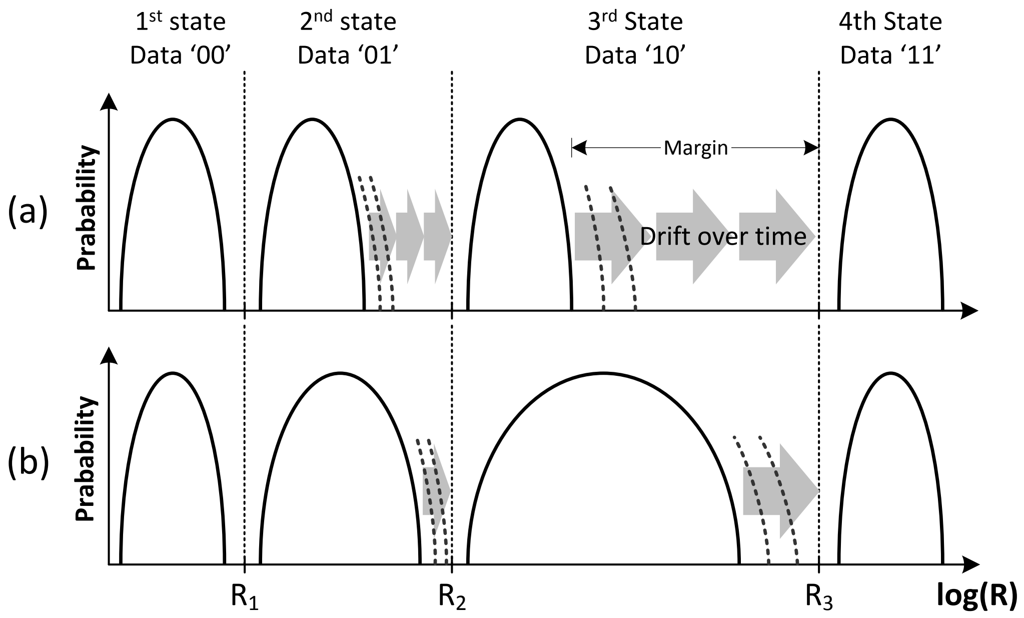


Figure 3.1: Illustration of the resistance distribution

To quantitatively estimate the data retention of PCM, previous works [14–17] suggest using the following resistance-drift model:

$$R(t) = R_0(\frac{t}{t_0})^{\nu} \tag{3.1}$$

Here R_0 is the initial resistance, $R(t)$ is the resistance at time t, t_0 is the time constant, and ν is the drift factor. With this equation, data retention can be found through calculating the time until the resistance of a cell in one state drifts all over the margin to the adjacent state. One thing worth noting here is, for 2-bit MLC PCM, data retention only involves the second and the third states. The reason why the first and the forth states are not affected by resistance drift is because the drift speed factor (ν) of the first state is negligibly small, and the resistance drift is uni-directional (i.e., only toward the high-resistance side), respectively.

Figure 3.2(a) shows the target bandwidth of baseline 2-bit MLC PCM. The resistance window is 3 to 6 in a logarithm scale (i.e., 10^3 to $10^6\Omega$). The target bandwidth of each intermediate data level is 0.458 in width [17]. Table 3.1 shows the draft speed factors we used. Note here that because the drift speed is different for different states, the resistance window is made non-uniform accordingly for maximal retention. According to

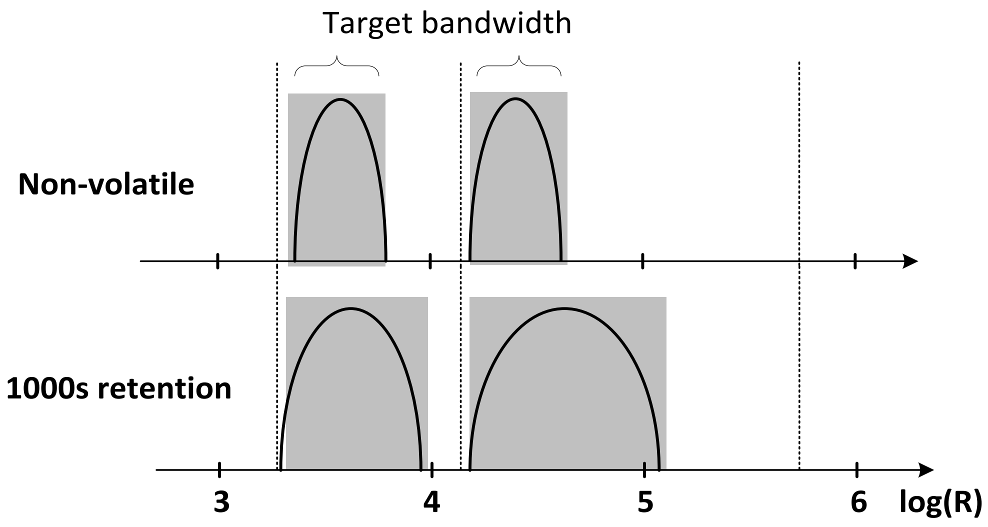


Figure 3.2: Visualization of the resistance window partition for baseline and the MP design

Equation 3.1, the baseline MLC PCM has 10^7 seconds (116 days) of data retention if we take the 4σ values into consideration.

Table 3.1: Drift factors for 2-bit MLC PCM when $t_0 = 1$s

Log(R) Range	ν		
	mean	SDMR	4σ value
< 3.5	0.001	40%	0.0026
3.5 – 4.5	0.02		0.052
4.5 – 5.5	0.06		0.156
> 5.5	0.10		0.26

When PCM is used as part of the main memory, its non-volatility (i.e., the 116-day retention) is unnecessary. Therefore, we see opportunities to trade away the excess non-volatility for improving write performance. In particular, we propose a new design concept named *morphable PCM* which employs tunable resistance distributions. For example, Figure 3.2 shows another distribution whose data retention is 1000 seconds. Below we referred to this mode as MP1000 where MP stands for *morphable PCM*. We can see that with relaxed non-volatility, MP1000 has wide resistance distributions (as well as small margins). More specifically, compared to the baseline, MP1000 allows 150% to

240% wider distributions.

3.2 How Widening Distribution Improves Write Speed

Once the resistance distributions are allowed to be wider, PCM writes can be speeded up because of reducing non-determinism. Writing MLC PCM is non-deterministic due to the variation and randomness of the phase-change material. Even the same cell can respond differently at different time. Previous works show that every time a MLC PCM cell is written, the cell has only 38% and 63% of chance to obtain the resistance conformed to the desired distribution: 38% for the first two trial and 63% for the following trial. Consequently, 8.5 trial-and-error iterations are required on average to complete a write [18,19]. With wider resistance distributions, such non-determinism is mitigated and writing MLC PCM cells becomes easier to success. Therefore, a write can be finished with less trial-and-error iterations.

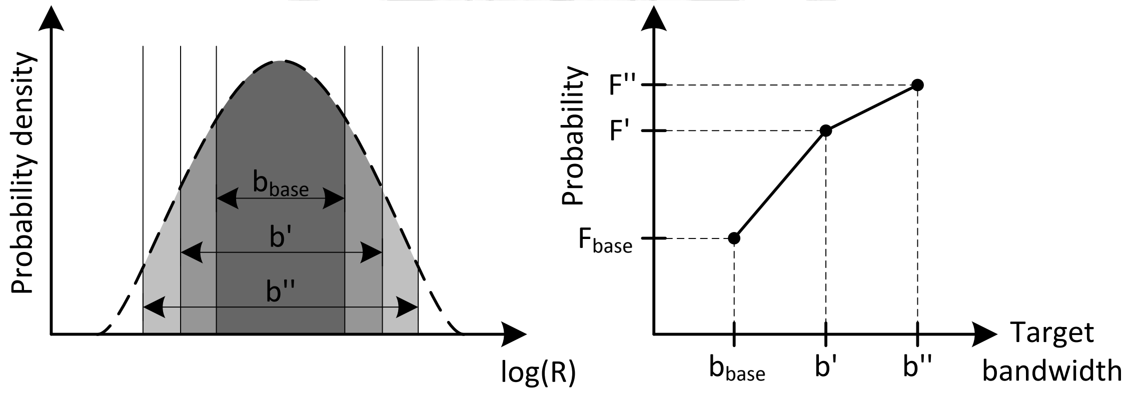


Figure 3.3: Illustration of the probability of successfully programming a cell vs. target bandwidth

Below we quantitatively analyze the opportunity of improving write performance. We refer to the resistance range of the desired distribution as *target bandwidth* and the probability of success per trial as *convergence probability*. As illustrated in Figure 3.3, the resulting resistance of PCM write is typically a normal distribution, therefore, larger target bandwidth means higher convergence probability. The convergence probability can be calculated as follows:

$$F(b) = NormCdf(NormInv(F_{base}) * \frac{b}{b_{base}}) \tag{3.2}$$

Here b_{base} is the baseline target bandwidth, F_{base} is the baseline convergence probability, $NormCdf(z)$ stands for the difference of standard normal cumulative distribution function between $\frac{z}{2}$ and $-\frac{z}{2}$, i.e.:

$$NormCdf(z) = \int_{-\frac{z}{2}}^{\frac{z}{2}} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}t^2} dt \tag{3.3}$$

and $NormInv$ is the inverse of $NormCdf$.

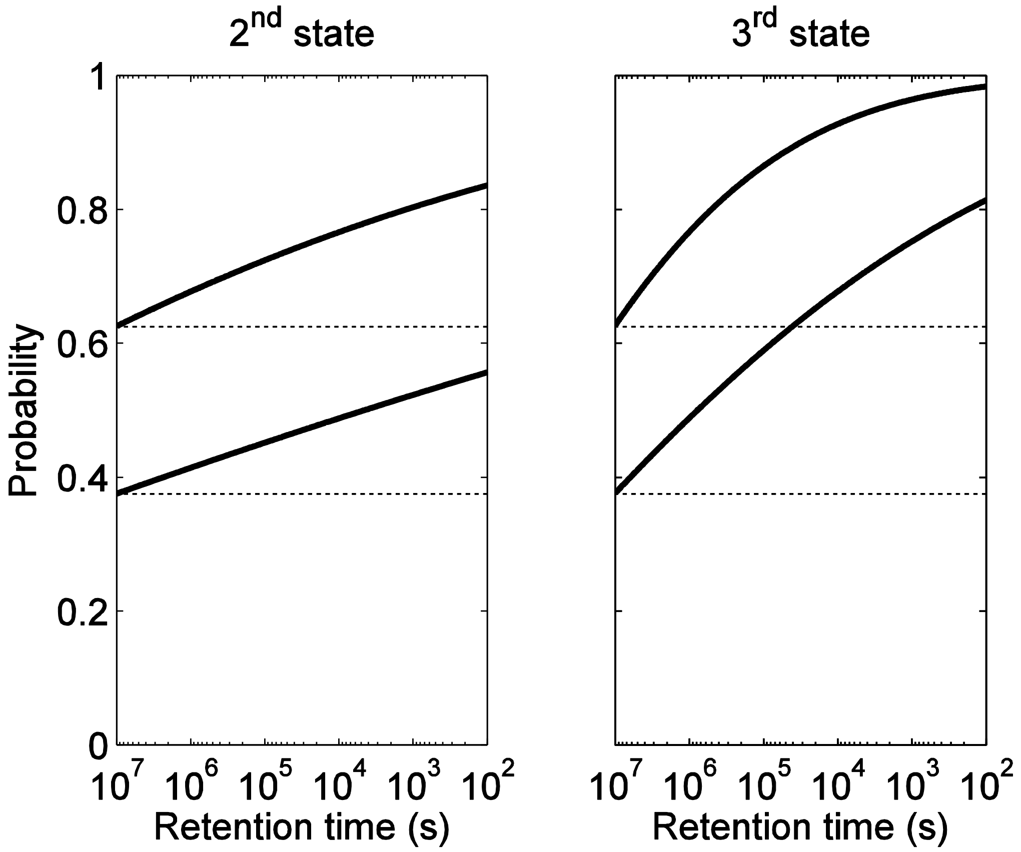


Figure 3.4: Convergence probability given various required data retention

Figure 3.4 sweeps the data retention from 10^7 to 10^3 seconds and plots the convergence probability. We can see that the MP1000 mode significantly increases the convergence probabilities from 38% to 76% and from 63% to 96%, respectively.

The increased convergence probability reduces the average number of write iterations required to write PCM cells. The average number of write iterations can be estimated

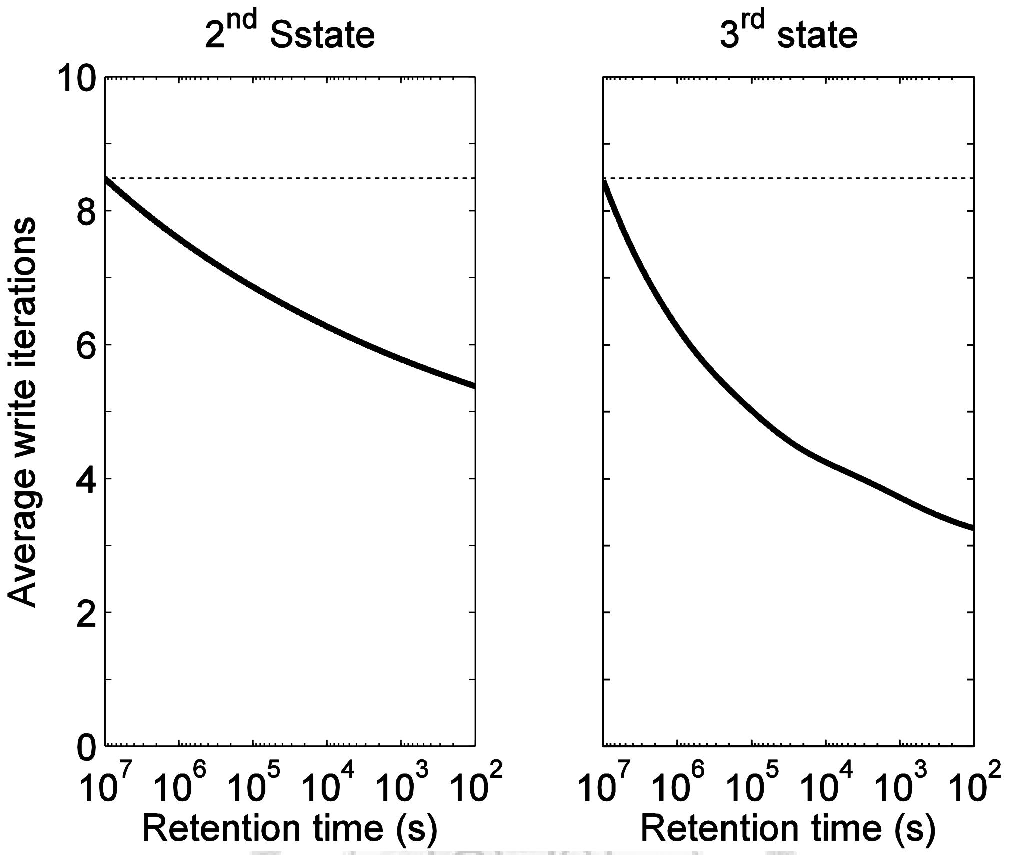


Figure 3.5: Average write iterations given various required data retention

using a Bernoulli process as in [18, 19]. Figure 3.5 shows the average write iteration iteration given various data retention. We can see that MP1000 reduces the average write iteration from 8.5 to 3.7.

Chapter 4

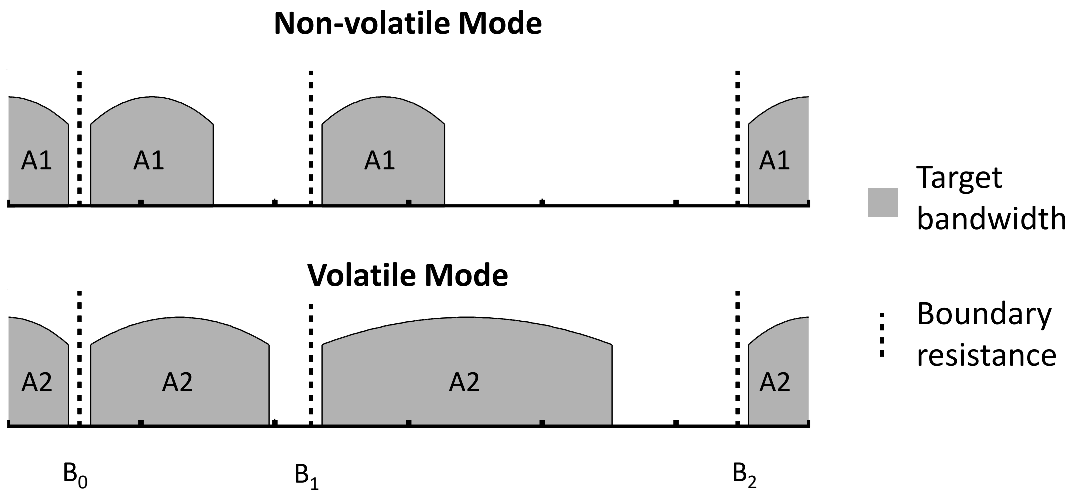
Designing a Morphable PCM-Based Memory System

The fundamental building block of the proposed morphable PCM-based system is PCM chips with dual operating modes. The morphable PCM chip uses the similar command interface as DDR3 DRAM. These PCM chips support commands including Activation, Write and Refresh which directly access the PCM array. Among these commands, the write command can occur in either the volatile mode or the non-volatile mode. The dual-mode PCM chips enables us to design a morphable PCM-based system. We aim at designing a memory management policy for the memory controller with following requirements. First, existing applications can leverage the benefits without code/OS modification. Next, future applications which utilize the persistency of PCM are supported.

4.1 Morphable PCM Commands

Following, we describe how to design the morphable PCM commands. The activation command reads data stored in the PCM cells and move them to the row buffer. The PCM read circuit senses the resistance of PCM cell and compares it with the boundary resistance to identify the resistance level and retrieve the data value. The activation command is used in the general PCM chip and do not require modification in the morphable PCM.

The write command stores the data to the row buffer and also programs the data into the PCM cells simultaneously. The MLC PCM write circuit usually uses the differential-



Command	Mode	Reference Resistances
Write	Non-volatile	A1
	Volatile	A2

Figure 4.1: Dual-mode write command

write and iterative-write procedures to program PCM cells. In morpahble PCM, different modes of write commands use different reference resistances bandwidth during the differential-write and the iterative-write procedures. As shown in Figure 4.1, the non-volatile write command uses the target bandwidth denoted as 'A1' during differential write and iterative write. By doing this, sufficient drift margins are created and data are guaranteed non-volatile. In contrast, the volatile write command uses the target bandwidth denoted as 'A2' so that the iterative write procedure can be speeded up. To support the dual-mode write command, PCM write circuit maintains two forms of reference resistance. And the memory controller can use the RFU (reserve future used) bits which is the reserved bits of current DDR DRAM commands interface to assign the write mode with little modification of current command interface.

The refresh command is to ensure the volatile data are valid over time. Refresh is combination of an array read followed by array write commands. As chapter 3 model, a short retention capability has better write speedup. However, it requires frequenter refresh operation to maintain the data correctness. Due to the expensive write operation of PCM, frequent refresh incurs performance, lifetime and power overhead. Below we use analytic equations to estimate the lifetime and refreshing power given various refresh interval. The power overhead due to refresh PCM cells, $P_{refresh}$, can be calculated as follows:

$$P_{refresh} = \frac{F \cdot N_{cell} \cdot WAV_{avg} \cdot E_{write}}{T_{refresh}} \tag{4.1}$$

here N_{cell} stands for total number of PCM cells in the system, F is the refresh factor which represents the rate of cells requiring refresh, E_{write} stands for energy consumption per cell per write iteration, WAV_{avg} stands for the number of iterations required on average to write a cell, and $T_{refresh}$ stands for the time duration to refresh the entire memory space.The resulting lifetime after adopting refresh can be calculated as follows:

$$L_{refresh} = \frac{N_{endurance}}{(\frac{N_{endurance}}{L_{base}}) + (\frac{F \cdot WAV_{avg}}{T_{refresh}})} \tag{4.2}$$

where $N_{endurance}$ is the cycle endurance of PCM cell, L_{base} is the baseline PCM lifetime. We assume there are 16GB MLC PCM, WAV_{avg} for baseline system is 2.25 iterations, E_{write} is 16.82 pJ, the PCM cycle endurance is 2×10^6 write cycle[1] and the L_{base} is 5 years. F is 0.5[2]. This setting is conservative regarding evaluating the impact of refresh.

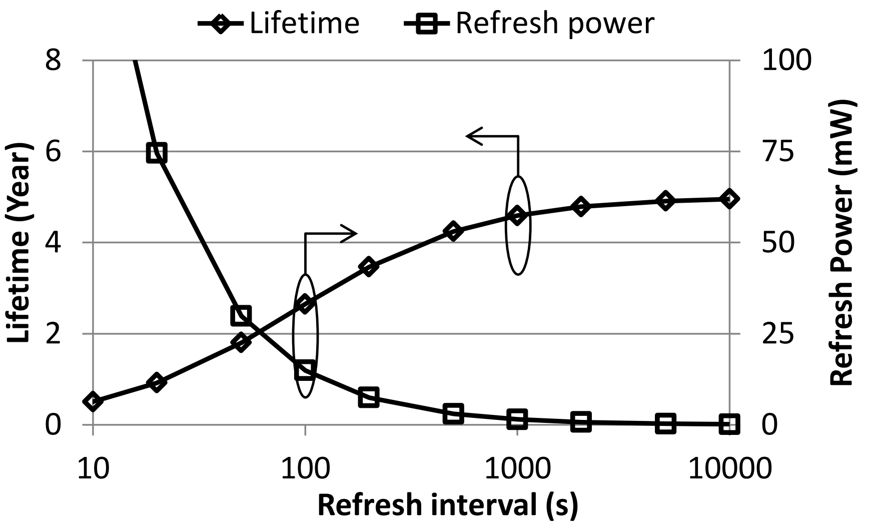


Figure 4.2: Lifetime and refreshing power vs. various refresh interval

Figure 4.2 shows the analyzing results. We can observe that when the refresh interval

[1]Previous wear-leveling studies consider the endurance of PCM to be 10^7 or above

[2]MLC PCM has 4 resistance level, and only two intermediate levels are volatile and should be refreshed

is larger than 1000 seconds, the PCM lifetime is 4.6 years (8% degradation). For refresh power, 1.5mW overhead is slight compared to about 5W of the current DRAM power. On the other hand, if the refresh interval is set to 100 seconds, the lifetime becomes half of baseline lifetime. And by retention mode of chapter 3, the write speedup difference between 1000 seconds and 100 seconds is small. Hence, we decide to set the retention and refresh interval to 1000 seconds based on this analysis.

A naive refresh policy is the DRAM-style refresh which is resemble the auto-refresh command of DRAM. DRAM-style refresh reads a 8KB memory row and writes them into memory array. For DRAM-style refresh, the written back data are volatile and should be refreshed again at next time point. Due to the MLC PCM characteristics like power limitation (e.g. only 512 cells can be programmed concurrently [20]), iterative write operation, refreshing a 8KB row which contains 32768 cells takes long time to complete and blocks other memory requests for a long time.

To mitigate the refresh impact, we propose other mechanisms to optimize the refresh. First, since PCM is non-destructive read which means the data are still stored in the memory array after the target row are activated, the refresh can be issued in a smaller granularity like 64B line instead of needing to program 8KB data back into memory array. Second, morphable PCM supports dual write modes for write operations. Therefore, PCM refresh can choose not only volatile write but also non-volatile write.

Compared to the row refresh, the line refresh mechanism refreshes 64B line at one time. Hence each line refresh refreshes less cell and has short cell programming latency. It can reduce the possibility of normal requests are blocked by refresh operation. However, it incurs additional read operation for refresh. Refresh command can choose either the volatile write or the non-volatile write for programming PCM cells. Volatile refresh allows cells locating in a wide target bandwidth, and therefore it requires less WAV iterations than non-volatile refresh when refreshing same number of cells. However, since the cells are volatile after refresh, those refreshed cells require refreshing again in the next time point. Non-volatile refresh can program cells into non-volatile mode such that those cells do not require refresh again. Non-volatile refresh can reduce the number refreshed cell and benefit from shorter programming latency.

4.2 Morphable PCM-Based Memory System

To allow existing applications to leverage the benefits of volatile PCM without any code or OS modification, the proposed scheme utilizes the volatile mode of commands by default. Refresh commands are issued in a staggered way by the memory controller. Therefore, data do not lose until system powers off. Since existing applications do not store files or persistent objects in the PCM main memory, such design does not lead to any reliability or correctness issues during power-off or a system failure. Future applications may store persistent objects or files in the PCM main memory. These applications inherently require the supporting from language, compiler, libraries, and the OS. That is, programmers have to explicitly declare an object persistent so that the OS knows that the object requires guarantees such as atomicity and consistency and should be allocated to a specific address space for future retrieval. To support these applications, we require the OS also to specify the non-volatile address space through notifying the memory controller the starting and ending addresses of the persistent address space. Through doing so, accesses to the persistent address space always occurs in the non-volatile mode.

Chapter 5

Evaluation

5.1 Experimental Setup

We conduct simulations based on MARSS [21] and DRAMSim2 [22]. MARSS is a cycle-accurate full-system simulator which uses PTLSim [23] for CPU simulation on top of the QEMU emulator [24]. DRAMSim2 is a cycle-accurate DDRx memory system simulator. Table 5.1 lists the configuration of the target system. We simulate a four-core CMP system and each core is modeled as an our-of-order pipeline. The MLC PCM main memory is 16 GB in total which consists of two ranks, and each rank contains eight banks. The system also contains DRAM caches organized as 32 MB pre-core private caches.

We extend MARSS and DRAMSim2 to capture the properties of a system with MLC PCM main memory. First, the latency of write command which programs data to PCM array is calculated according to the data pattern. Each write command take dynamic number of write-and-verify iteration to complete. Power constraints for PCM writes and power budgeting mechanism similar to [20] are implemented in the model. We also prioritize reads over writes by default to improve read responsiveness. For the proposed morphable memory system, a staggered refresh is employed.

We choose five memory-intensive workloads from SPEC CPU2006 [25], including bwave, zeusmp, cactusADM, leslie3d, and lbm. Workloads are executed in both a rate mode (each core executes the same workload) and a mixed mode (four cores execute four different workloads). Table 5.2 lists the 10 combinations of workload we use and their RPKI and WPKI (i.e., main-memory reads/writes per kilo instructions). For each combination, simulation starts from a check point which is 30 seconds after starting the workload. We perform detailed simulation for 2.5 billion of instructions. The first half billion serves as warm-up and the following two billions are for performance evaluation.

Table 5.1: System configuration

System	4-core CMP, 4GHz
Processor Core	4-issue, out-of-order, 32KB iL1, 32KB dL1
L2 (private)	2MB, 4-way, 64B line size, LRU, write back
DRAM cache (private)	32MB, 8-way, 64B line size, 50ns access latency, write back
Main Memory	16GB PCM, 2 ranks of 8-bank each, 8KB row buffer, differential-write support
PCM Latency	Read: 250ns Write: 250ns/iteration Each write operation (to array) takes a variable number of iterations according to the data pattern
PCM Power	Power limitation: 512 cells Read energy: 2.47 pJ/cell Write energy: 16.82 pJ/cell/iteration

Table 5.2: Simulated applications

Benchmark	**Description**	**RPKI**	**WPKI**
bwaves	4 copies of bwaves	5.45	5.12
zeusmp	4 copies of zeusmp	10.75	4.47
cactus	4 copies of cactusADM	10.47	6.20
leslie	4 copies of leslie3d	7.49	3.71
lbm	4 copies of lbm	15.15	4.99
mix_1	bwaves-zeusmp-cactusADM-leslie3d	7.83	4.61
mix_2	bwaves-zeusmp-cactusADM-lbm	9.64	5.86
mix_3	bwaves-zeusmp-leslie3d-lbm	9.05	4.20
mix_4	bwaves-cactusADM-leslie3d-lbm	8.77	5.07
mix_5	zeusmp-cactusADM-leslie3d-lbm	11.28	5.37

The scenarios we evaluate are as follows:

- **Baseline:** This stands for the baseline systems. Data stored into PCM are non-volatile, so no refresh is required.

Refresh policy	Volatile	Non-volatile
Row	MP (DRAM-style)	MP-R1
Line	MP-R2	MP-R3

Table 5.3: Refresh Policy

- **MP:** These stand for morphable MLC PCM systems whose designed data retention is equal to 1000 seconds. A staggered refresh is required in this scheme as PCM is now volatile. In this case, DRAM-style refresh is used to ensure the data correctness.
- **MP-R1, MP-R2, MP-R3:** There are similar to MP except the refresh is issued by different policies. Table 5.3 lists the different refresh policies. MP-R1 means the row and non-volatile refresh. MP-R2 is the line and volatile refresh. MP-R3 uses the line and non-volatile refresh.
- **MP-ideal:** This is similar to MP except the refresh is issued with zero penalty. We use these hypothetical scenarios to present the refresh impact on performance and PCM lifetime.

To compare the performance of the above scenarios, we evaluate several metrics including average write latency, memory utilization, read latency, IPC speedup.

5.2 Experimental Results

5.2.1 Performance

Figure 5.1 compares the write latency of PCM, i.e., brining data from the row buffer to the array of PCM. We normalize the results to the baseline. We observe that for ideal case, MP-ideal can reduce the average write latency to 77% of the baseline. Due to the refresh operation, the write requests may be blocked. For example, cactus has longer write latency than baseline from refresh operation. Therefore, MP case reduce the average write latency to 87% of the baseline.

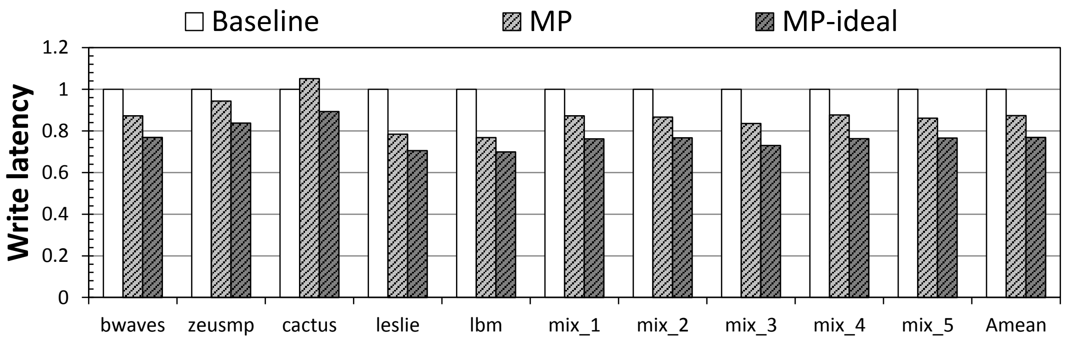


Figure 5.1: Write latency

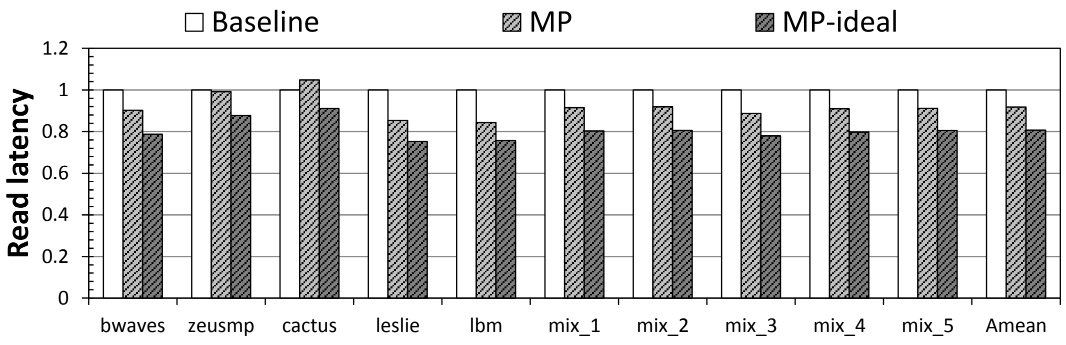


Figure 5.2: Read latency improvement

Reducing write latency helps improving the responsiveness of read. Figure 5.2 compares the average read latency. The results are normalized to the baseline. Generally, we see that the read latency reduction is less than the write latency reduction. On average, MP can reduce read latency to 92% of the baseline. We can see that the difference between MP and MP-ideal is about 12%. Such results mean that dram-style refresh is unsuitable.

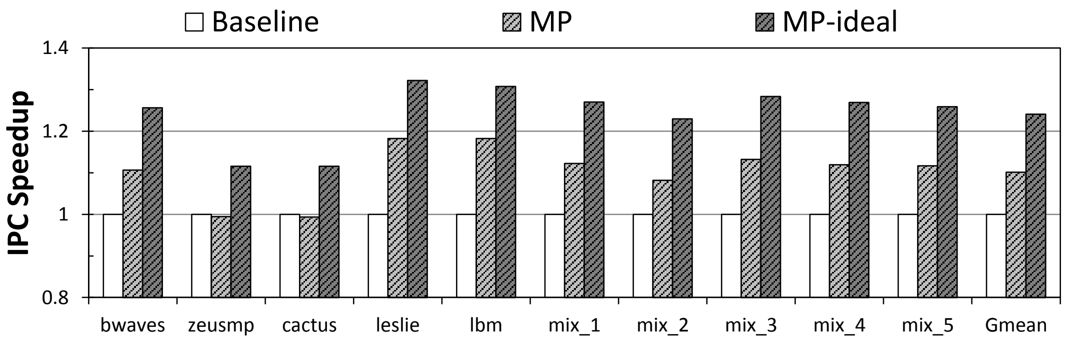


Figure 5.3: Performance (IPC) improvement of DRAM-style refresh

The reduced read latency translates into performance improvement. Figure 5.6 plots the IPC improvement over the baseline. On average, MP can achieve 10% IPC speedup. For lbm, the improvement is as high as 18%. We can see that refresh incurs 14% performance gap between MP and MP-ideal. It means that for the traditional DRAM-style refresh, there are large performance gap between the normal case and ideal case.

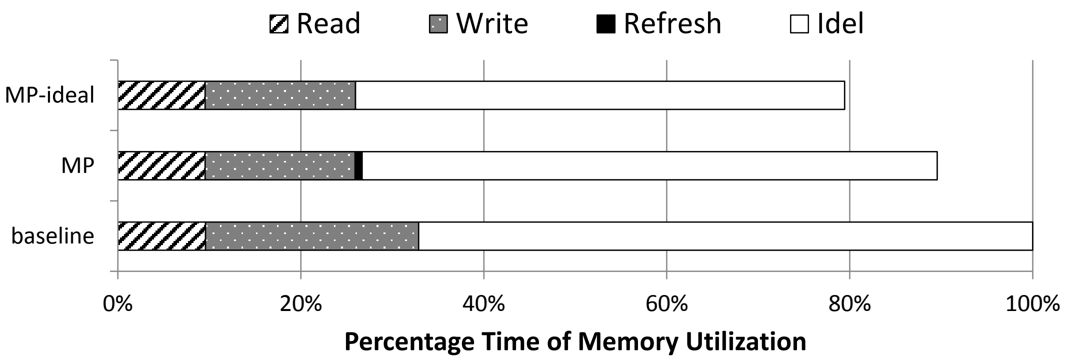


Figure 5.4: Bank utilization

Figure 5.4 shows the results of bank utilization. Results is normalized to the baseline. We can observe that the baseline system spends 10% of the time in read operations. For the MP system, it can reduce the average write utilization from 23% to 16%. We also can see that the refresh only occupies the memory utilization by less than 1%. Hence, performance gap between MP and MP-ideal should comes from the long latency of each refresh operation which blocks normal requests for long time.

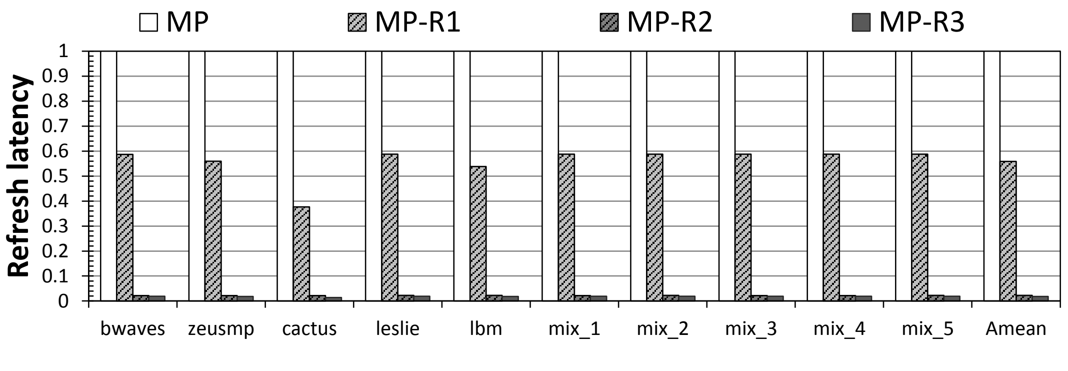


Figure 5.5: Refresh latency under different refresh policies

Below we show the results of different refresh policies. Figure 5.5 shows the averger refresh latency for different refresh policies, and the results are normalized to MP. We can observe that MP has longest latency because each refresh requires refreshing most cells. For MP-R1, it uses the non-volatile write such that it can reduce the refresh cells number significantly. Therefore its latency can be reduced. For line style refresh, its refresh latency is much shorter since the average refresh cell number at one time is less.

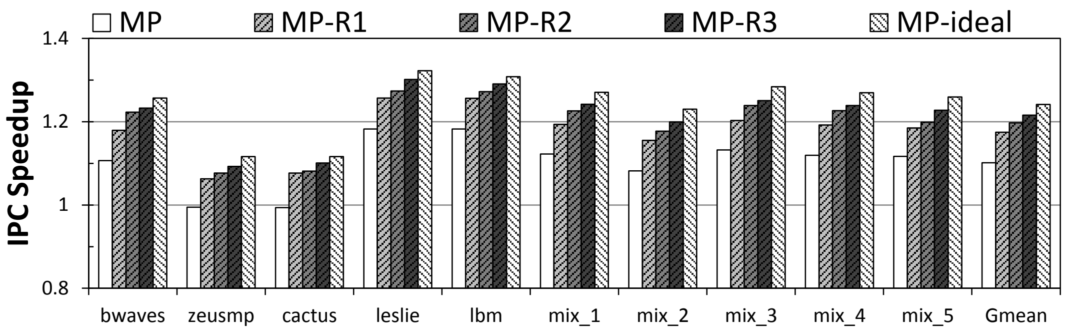


Figure 5.6: IPC speedup of different refresh policies

Figure 5.6 shows the IPC speedup between different refresh policies. Results are normalized to baseline. We can see that MP-R1 provides 17% speedup over baseline mitigate the performance gap between it and ideal case to 7%. MP-R2 and MP-R3 provide 19% and 21.5% speedup over baseline respectively. The performance gap between MP-R3 and ideal case can be reduced to 2.5%.

5.2.2 Lifetime

For each workload we assume perfect wear-leveling and use the following equation to estimate the memory lifetime:

$$Lifetime = \frac{N_{endurance} \times N_{cell}}{N_{stress}} \times T_{base} \tag{5.1}$$

Here $N_{endurance}$ stands for the endurance of a PCM cell, N_{cell} is total number of PCM cells, N_{stress} is the total number of write stress events applied to the PCM, and T_{base} is the execution time of the baseline[1].

Figure 5.7 shows the estimated memory lifetime. We compared the baseline with MP-R2 and MP-R3 since they perform best performance speedup in pervious experiments. On average, the baseline lifetime is about 12 years. Because bwave incurs less PCM stress,

[1]We use T_{base} instead of the execution of individual design for fair comparison. Otherwise, a machine with higher IPC could present worse lifetime than a lower-IPC one.

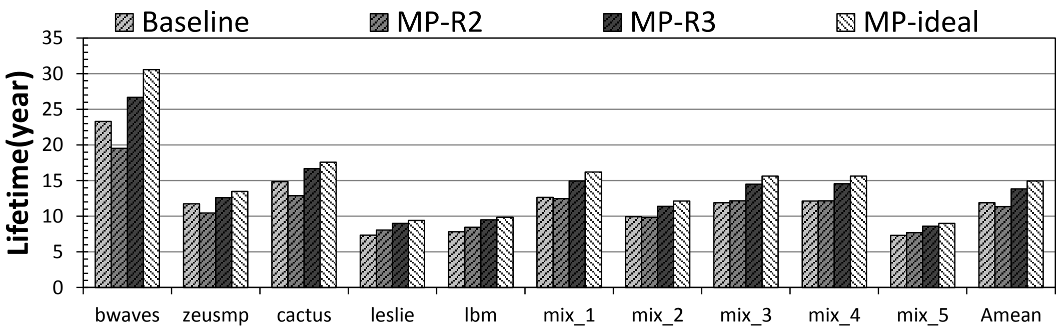


Figure 5.7: Lifetime under different refresh policies

it results in the highest 23 years of lifetime. Since morphable PCM can reduce the write iteration number and the write stress, the ideal case can provide about 3 years lifetime improvement. MP-R2 incurs more write stress from the volatile refresh which requires refreshing more cells. Hence, its lifetime is worse than the baseline. MP-R3 uses the non-volatile refresh to reduce the number of refreshed cells, so it incurs less write stress on PCM array. On average, MP-R3 can improve the lifetime by 2 years compared to the baseline system.

5.3 Sensitivity Study

# σ	Baseline Retention	Speedup over the Baseline (times)			
		MP1000		MP100	
		2nd state	3rd state	2nd state	3rd state
3	6.1 years	1.51	2.52	1.60	2.81
4	3.8 months	1.46	2.37	1.57	2.71
5	2 weeks	1.40	2.21	1.53	2.60

Table 5.4: Speedup sensitivity to the worst variation of the resistance drift speed, ν

The drift velocity coefficient of PCM, ν, follows a normal distribution. Therefore, when estimating the data retention time of PCM memory, one should consider a ν larger than the mean. The criterion of how larger the ν should be considered depends on the strength of ECC – stronger ECC can tolerate higher bit error rate and allow one to only consider a smaller ν. Our baseline design considers a ν which is 4σ larger than the mean.

We vary this criteria from 3σ to 5σ for sensitivity analysis. Table 5.4 shows the retention of the baseline and the write latency speedup of MP1000 and MP100. We consider the case that the number of updated cells is 64 (all to the intermediate states). First, we can see that for the 3σ case (i.e., strong ECC is adopted), the retention of the baseline is 2236 days (about 6.1 years). Under this case, MP1000 and MP100 achieve $1.51\times$ and $1.60\times$ speedup which is slightly higher than the original 4σ case. For the 5σ case (i.e., very weak ECC is adopted), the retention of the baseline becomes 2 weeks. Even so, we can see that MP1000 and MP100 can achieve $1.40\times$ and $1.53\times$ speedup.

5.4 Comparison of Techniques

MP is orthogonal to other write optimizing techniques for MLC PCM such as write cancellation [18], write pausing [18], and write truncation [19]. Write cancellation and write pausing make write operations preemptable. Write truncation exploits the opportunity to omit difficult-to-write cells. In contrast, MP focuses on improving the chance of successfully programming a cell per iteration, the underlying cause which increase the write iterations of MLC PCM. Therefore, combining MP with the above mentioned techniques can further improve system performance.

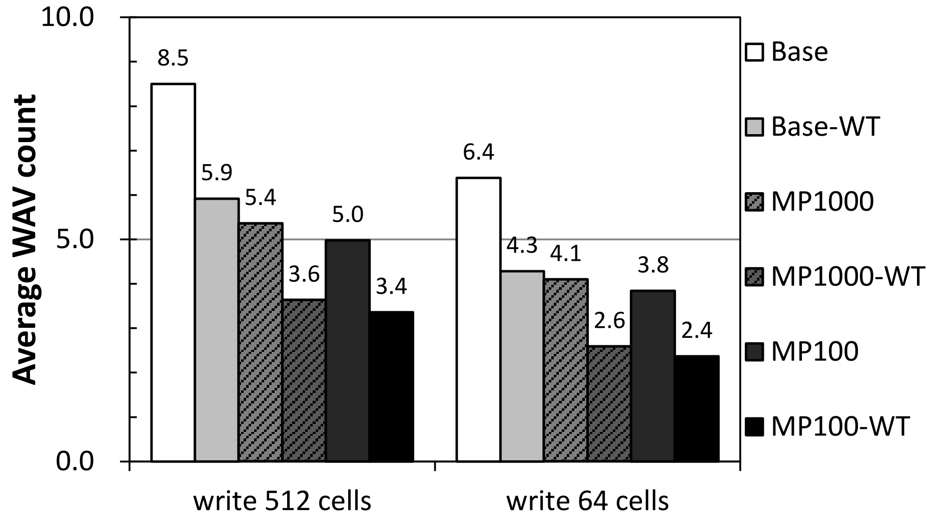


Figure 5.8: Comparing and combining MP with Write Truncation (WT)

Figure 5.8 compares MP with write truncation as well as shows the effect of combining the two. We conduct Monte Carlo simulation of 1 million writes and derive the average write latency. We consider two cases, writing 512 and writing 64 cells (all to the intermediate states). We consider that write truncation can omit writing 4 most difficult-to-write cells as suggested in [19]. Space overhead of ECC parities are omitted for write truncation assuming data compression is applicable. We can see that using write truncation alone can reduce write iterations from 8.5 to 5.9, and 6.4 to 3.8 for the two cases, respectively. In comparison, utilizing MP alone can give more write improvement than write truncation – only 4.1 and 3.3 iterations on average are required for 100-second retention. Equipping MP with write truncation leverages the benefits of the two – the average write iterations can be reduced to as low as 2.9 and 2.0.

Besides being orthogonal, MP incurs low overhead compared to the others. Since write cancellation and write pausing allow preempting writing a row, an additional row buffer (per bank) is required to hold the data of the preempted write, not to mention the buffers to keep those write parameters learned during the learning phase. As previous study [3] implies, additional row buffers are valuable resources and can be otherwise used for improving the performance of PCM main memory. On the other hand, write truncation requires stronger ECC encoding/decoding engines than the baseline does. It also needs compressing and decompressing engines if one wants to avoid space overhead to store ECC parities.

Chapter 6

Related Work

PCM memory is considered a promising technology in computer memory hierarchy due to its non-volatility, high density, byte addressability, and excellent scalability. Zhou et al. [2], Lee et al. [3], and Qureshi et al. [1] propose to architect PCM-based main memory. Coburn et al. [12] and Volos et al. [13] propose methods to utilize the non-volatility of PCM to manage persistent objects in main memory. Condit et al. [11], Sun et al. [26], Akel et al. [27] and Caulfield et al. [28, 29] research PCM-based storage and related OS and architectural optimization.

The limited endurance of PCM is one of the most widely mentioned issues of PCM. When serving as main memory, PCM could present worn-out cell in few minutes if lack of proper protection [30]. Fortunately, low-overhead and effective solutions have already been proposed. Differential write [3] schemes are proposed to eliminate unnecessary writes to PCM. It happens in different granularity including bit-level one [2] and cache line-level one [1]. Wear-leveling techniques such as address rotation [31] and segment swapping [2] are proposed to spread writes across the entire memory space to avoid concentrated wear-out and prolong overall lifetime. To combat malicious attack, Seong et al. [30] propose a randomized address mapping scheme, and Qureshi et al. [32] propose online detecting algorithm. To handle worn-out cells gracefully, Seong et al. [33], Schechter et al. [34], Yoon et al. [35], and Qureshi [36] propose specialized error correcting schemes.

In addition to the limited endurance, the relatively poor write performance of PCM also receives a lot of attention. Qureshi et al. [18] propose schemes named write cancellation and write pausing to allow read operations to abort or pause on-going write operations. Jiang et al. [19] propose a scheme called write truncation which exploits ECC capability to curtail a write operation before all the data are written correctly. Joshi et al. [37] improve write speed and energy through selectively using two write schemes, set-to-reset and reset-to-set, according to data value. Cho and Lee [38], Xu et al. [39], and Wang et al. [40] propose data encoding schemes to improve the write bandwidth and reduce the write energy of PCM. Hay et al. [20] propose a power budgeting approach to improve the write bandwidth of PCM under constrained power.

Relaxing non-volatility is also a strategy to optimize write performance of non-volatile memory. Smullen et al. [41], Sun et al. [42], and Jog et al. [43] analyze the benefit of relaxing the non-volatility of STT-RAM cache on write latency and write energy and proposed corresponding cache management policies. Liu et al. [44] and Pan et al. [45] advocate relaxing the retention of NAND Flash to optimize write performance, ECC cost, and lifetime of SSDs.

Compared to prior work, this paper is the first one on architecting volatile MLC PCM main memory. We analyze the benefit on improving write speed via relaxing the retention capability of MLC PCM. We analyze real applications to reveal the excess non-volatility of PCM when it serves as main memory. We propose design with volatile MLC PCM and conduct full-system simulations to quantitatively demonstrate the performance gain.

Chapter 7

Conclusions

In this thesis, we propose a new design concept for PCM-based memory system. PCM chips can support two write operation modes: non-volatile mode and fast-yet-volatile mode. The fast-yet-volatile mode can be used in the PCM main memory system to speedup the write operation and improve the system performance. And the non-volatile mode can be used to provide the opportunity to store persist object or the file in PCM system to avoid accessing disk. To analyze the benefit of the fast-yet-volatile write mode, we model the MLC PCM and quantify the relation between write latency and data retention. we show the fast-yet-volatile write can speedup the write latency by 1.46X if the data retention is 1000 seconds.

For the PCM reliability, the volatile data should be refreshed to ensure the data are valid over time. Due to the programming current limitation, the refresh operation which programmed a 8KB row size would block a memory bank for long time. Hence, we proposed a fine-grained and non-volatile way to issue refresh commands such that we can mitigate the refresh overhead. By the full-system simulation, we show that refresh PCM with 1000-second interval incurs insignificant degradation on energy, performance, and lifetime. Results show that system with 1000-second retention capability can achieve 21.5% IPC improvement.

Bibliography

[1] Moinuddin K. Qureshi, Vijayalakshmi Srinivasan, and Jude A. Rivers. Scalable high performance main memory system using phase-change memory technology. In *ISCA '09*, 2009.

[2] Ping Zhou, Bo Zhao, Jun Yang, and Youtao Zhang. A durable and energy efficient main memory using phase change memory technology. In *ISCA '09*, 2009.

[3] Benjamin C. Lee, Engin Ipek, Onur Mutlu, and Doug Burger. Architecting phase change memory as a scalable DRAM alternative. In *ISCA '09*, 2009.

[4] S. Raoux, G. W. Burr, M. J. Breitwisch, C. T. Rettner, Y.-C. Chen, R. M. Shelby, M. Salinga, D. Krebs, S.-H. Chen, H.-L. Lung, and C. H. Lam. Phase-change random access memory: a scalable technology. *IBM J. Res. Dev.*, 52(4):465–479, July 2008.

[5] Youngdon Choi, Ickhyun Song, Mu-Hui Park, Hoeju Chung, Sanghoan Chang, Beakhyoung Cho, Jinyoung Kim, Younghoon Oh, Duckmin Kwon, Jung Sunwoo, Junho Shin, Yoohwan Rho, Changsoo Lee, Min Gu Kang, Jaeyun Lee, Yongjin Kwon, Soehee Kim, Jaehwan Kim, Yong-Jun Lee, Qi Wang, Sooho Cha, Sujin Ahn, H. Horii, Jaewook Lee, Kisung Kim, Hansung Joo, Kwangjin Lee, Yeong-Taek Lee, Jeihwan Yoo, and G. Jeong. A 20nm 1.8V 8Gb PRAM with 40MB/s program bandwidth. In *ISSCC '12*, 2012.

[6] 2011 ITRS PIDS chapter. Itrs 2011. Technical report, ITRS, 2011.

[7] T. Nirschl, J.B. Phipp, T.D. Happ, G.W. Burr, B. Rajendran, M.-H. Lee, A. Schrott, M. Yang, M. Breitwisch, C.-F. Chen, E. Joseph, M. Lamorey, R. Cheek, S.-H. Chen, S. Zaidi, S. Raoux, Y.C. Chen, Y. Zhu, R. Bergmann, H.-L. Lung, and C. Lam. Write strategies for 2 and 4-bit multi-level phase-change memory. In *IEDM '07*, 2007.

[8] F. Bedeschi, R. Fackenthal, C. Resta, E.M. Donze, M. Jagasivamani, E. Buda, F. Pellizzer, D. Chow, A. Cabrini, G.M.A. Calvi, R. Faravelli, A. Fantini, G. Torelli,

Duane Mills, R. Gastaldi, and G. Casagrande. A multi-level-cell bipolar-selected phase-change memory. In *ISSCC '08*, 2008.

[9] F. Bedeschi, R. Fackenthal, C. Resta, E.M. Donze, M. Jagasivamani, E.C. Buda, F. Pellizzer, D.W. Chow, A. Cabrini, G. Calvi, R. Faravelli, A. Fantini, G. Torelli, D. Mills, and G. Gastaldi, R.and Casagrande. A bipolar-selected phase change memory featuring multi-level cell storage. *IEEE J. Solid-St. Circ.*, 44(1):217 –227, Jan. 2009.

[10] D-H. Kang, J.-H. Lee, J.H. Kong, D. Ha, J. Yu, C.Y. Um, J.H. Park, F. Yeung, J.H. Kim, W.I. Park, Y.J. Jeon, M.K. Lee, Y.J. Song, J.H. Oh, G.T. Jeong, and H.S. Jeong. Two-bit cell operation in diode-switch phase change memory cells with 90nm technology. In *2008 Symposium on VLSI Technology*, 2008.

[11] Jeremy Condit, Edmund B. Nightingale, Christopher Frost, Engin Ipek, Benjamin Lee, Doug Burger, and Derrick Coetzee. Better I/O through byte-addressable, persistent memory. In *SOSP '09*, 2009.

[12] Joel Coburn, Adrian M. Caulfield, Ameen Akel, Laura M. Grupp, Rajesh K. Gupta, Ranjit Jhala, and Steven Swanson. NV-heaps: making persistent objects fast and safe with next-generation, non-volatile memories. In *ASPLOS '11*, 2011.

[13] Haris Volos, Andres Jaan Tack, and Michael M. Swift. Mnemosyne: lightweight persistent memory. In *ASPLOS '11*, 2011.

[14] Wangyuan Zhang and Tao Li. Helmet: A resistance drift resilient architecture for multi-level cell phase change memory system. In *DSN '11*, 2011.

[15] D. Ielmini, D. Sharma, S. Lavizzari, and A.L. Lacaita. Reliability impact of chalcogenide-structure relaxation in phase-change memory (PCM) cells - Part I: experimental study. 56(5):1070 –1077, May 2009.

[16] Wei Xu and Tong Zhang. Using time-aware memory sensing to address resistance drift issue in multi-level phase change memory. In *ISQED '10*, 2010.

[17] M. Awasthi, M. Shevgoor, K. Sudan, B. Rajendran, R. Balasubramonian, and V. Srinivasan. Efficient scrub mechanisms for error-prone emerging memories. In *HPCA '12*, 2012.

[18] M.K. Qureshi, M.M. Franceschini, and L.A. Lastras-Montano. Improving read performance of phase change memories via write cancellation and write pausing. In *HPCA '10*, 2010.

[19] Lei Jiang, Bo Zhao, Youtao Zhang, Jun Yang, and B.R. Childers. Improving write operations in MLC phase change memory. In *HPCA '12*, 2012.

[20] Andrew Hay, Karin Strauss, Timothy Sherwood, Gabriel H. Loh, and Doug Burger. Preventing PCM banks from seizing too much power. In *MICRO-44*, 2011.

[21] A. Patel, F. Afram, Shunfei Chen, and K. Ghose. MARSS: a full system simulator for multicore x86 CPUs. In *DAC '11*, 2011.

[22] P. Rosenfeld, E. Cooper-Balis, and B. Jacob. DRAMSim2: A cycle accurate memory system simulator. *Computer Architecture Letters*, 10(1):16 –19, 2011.

[23] M.T. Yourst. PTLsim: a cycle accurate full system x86-64 microarchitectural simulator. In *ISPASS '07*, 2007.

[24] Fabrice Bellard. QEMU, a fast and portable dynamic translator. In *USENIX '05*, 2005.

[25] John L. Henning. SPEC CPU2006 benchmark descriptions. *SIGARCH Comput. Archit. News*, 34(4):1–17, September 2006.

[26] Guangyu Sun, Yongsoo Joo, Yibo Chen, Dimin Niu, Yuan Xie, Yiran Chen, and Hai Li. A hybrid solid-state storage architecture for the performance, energy consumption, and lifetime improvement. In *HPCA '10*, 2010.

[27] Ameen Akel, Adrian M. Caulfield, Todor I. Mollov, Rajesh K. Gupta, and Steven Swanson. Onyx: a protoype phase change memory storage array. In *HotStorage '11*, 2011.

[28] Adrian M. Caulfield, Arup De, Joel Coburn, Todor I. Mollow, Rajesh K. Gupta, and Steven Swanson. Moneta: A high-performance storage array architecture for next-generation, non-volatile memories. In *MICRO-43*, 2010.

[29] Adrian M. Caulfield, Todor I. Mollov, Louis Alex Eisner, Arup De, Joel Coburn, and Steven Swanson. Providing safe, user space access to fast, solid state disks. In *ASPLOS '12*, 2012.

[30] Nak Hee Seong, Dong Hyuk Woo, and Hsien-Hsin S. Lee. Security refresh: prevent malicious wear-out and increase durability for phase-change memory with dynamically randomized address mapping. In *ISCA '10*, 2010.

[31] Moinuddin K. Qureshi, John Karidis, Michele Franceschini, Vijayalakshmi Srinivasan, Luis Lastras, and Bulent Abali. Enhancing lifetime and security of PCM-based main memory with start-gap wear leveling. In *MICRO-42*, 2009.

[32] M.K. Qureshi, A. Seznec, L.A. Lastras, and M.M. Franceschini. Practical and secure PCM systems by online detection of malicious write streams. In *HPCA '11*, 2011.

[33] Nak Hee Seong, Dong Hyuk Woo, Vijayalakshmi Srinivasan, Jude A. Rivers, and Hsien-Hsin S. Lee. SAFER: stuck-at-fault error recovery for memories. In *MICRO-43*, 2010.

[34] Stuart Schechter, Gabriel H. Loh, Karin Straus, and Doug Burger. Use ECP, not ECC, for hard failures in resistive memories. In *ISCA '10*, 2010.

[35] Doe Hyun Yoon, N. Muralimanohar, Jichuan Chang, P. Ranganathan, N.P. Jouppi, and M. Erez. FREE-p: protecting non-volatile memory against both hard and soft errorss. In *HPCA '11*, 2011.

[36] Moinuddin K. Qureshi. Pay-as-you-go: low-overhead hard-error correction for phase change memories. In *MICRO-44*, 2011.

[37] M. Joshi, Wangyuan Zhang, and Tao Li. Mercury: A fast and energy-efficient multi-level cell based phase change memory system. In *HPCA '11*, 2011.

[38] Sangyeun Cho and Hyunjin Lee. Flip-N-Write: a simple deterministic technique to improve PRAM write performance, energy and endurance. In *MICRO-42*, 2009.

[39] Wei Xu, Jibang Liu, and Tong Zhang. Data manipulation techniques to reduce phase change memory write energy. In *ISLPED '09*, 2009.

[40] Jue Wang, Xiangyu Dong, Guangyu Sun, Dimin Niu, and Yuan Xie. Energy-efficient multi-level cell phase-change memory system with data encoding. In *ICCD '11*, 2011.

[41] C.W. Smullen, V. Mohan, A. Nigam, S. Gurumurthi, and M.R. Stan. Relaxing non-volatility for fast and energy-efficient STT-RAM caches. In *HPCA '11*, 2011.

[42] Zhenyu Sun, Xiuyuan Bi, Hai (Helen) Li, Weng-Fai Wong, Zhong-Liang Ong, Xiaochun Zhu, and Wenqing Wu. Multi retention level STT-RAM cache designs with a dynamic refresh scheme. In *MICRO-44*, 2011.

[43] Adwait Jog, Asit K. Mishra, Cong Xu, Yuan Xie, Vijaykrishnan Narayanan, Ravishankar Iyer, and Chita R. Das. Cache revive: architecting volatile STT-RAM caches for enhanced performance in CMPs. In *DAC '12*, 2012.

[44] Ren-Shuo Liu, Chia-Lin Yang, and Wei Wu. Optimizing NAND flash-based SSDs via retention relaxation. In *FAST '12*, 2012.

[45] Yangyang Pan, Guiqiang Dong, Qi Wu, and Tong Zhang. Quasi-nonvolatile SSD: trading flash memory nonvolatility to improve storage system performance for enterprise applications. In *HPCA '12*, 2012.