

# Handling PCM Resistance Drift with Device, Circuit, Architecture, and System Solutions

Manu Awasthi<sup>†</sup>, Manjunath Shevgoor<sup>†</sup>, Kshitij Sudan<sup>†</sup>, Rajeev Balasubramonian<sup>†</sup>,  
Bipin Rajendran<sup>‡</sup>, Viji Srinivasan<sup>‡</sup>  
<sup>†</sup>University of Utah, <sup>‡</sup>IBM T.J. Watson Research Center

## Abstract

*Phase Change Memory (PCM) is expected to be incorporated in future memory and storage hierarchies. In recent research, several efforts have been undertaken to overcome PCM limitations of write endurance and write energy. In this paper, we explore a lesser known challenge for PCM – Resistance Drift. Studies have shown that the resistance of a PCM cell increases over time, possibly yielding a wrong state upon read. This is expected to be a frequent source of “soft errors” in PCM devices. The problem is especially significant in multi-level cells (MLCs), where the resistance ranges for each state are more closely spaced. The effect of drift can be negated with a refresh operation that re-writes the contents of a line, similar to refreshes in DRAM devices. However, PCM writes are very expensive in terms of energy, delay, and their effect on endurance, thus making frequent writes infeasible. This paper proposes device, circuit, architecture, and system-level techniques to balance the overhead of refresh with error rate. At the device level, write mechanisms that favor write precision are likely more effective at minimizing the overall efficiency of writes. At the architecture level, we introduce a light-read operation and combine it with ECC support to reduce the need for expensive writes. This basic operation can be augmented with circuit, architectural, and system-level techniques to achieve various trade-offs between energy, latency, soft error rates, and hard error rates. We thus lay the framework for a large family of viable soft error management policies that span the entire system stack.*

## 1 Introduction

Phase change memory (PCM) has emerged as a front-runner among emerging NVM technologies because of its relative maturity in terms of manufacturability [1]. The architecture community has recently made several efforts to identify the role that PCM can play within the memory hierarchy (for example, [9]) and overcome some of its drawbacks [3, 7]. The primary drawback in PCM is the impact of writes on memory latency, energy, and endurance. Various optimizations have been proposed recently to minimize the number of written bits [4] and to efficiently handle the emergence of wearout-based hard errors [10]. However, to date, there has been no architecture evaluation on the impact of resistance drift in PCM. Many papers in the device community have articulated the significance of this problem [3] and it appears that a combination of device, architecture, circuit, and system-level solutions will be required to overcome the problem and realize the potential of MLC PCM. In this abstract, we provide a brief summary of this problem, its significance, and the solution space. The abstract distills the major insight in a detailed paper that is currently under review at ISCA.

## 2 Problem of Resistance Drift

PCM devices are built out of chalcogenide alloy composed of **Ge**, **Sb**, and **Te** which can exist in either low resistance crystalline (SET) or high resistance amorphous state (RESET) [3]. Hybrid states between the amorphous and

crystalline extremes can also be constructed which in turn yields materials with varying percentages of the crystalline state, and hence varying resistance levels. This is the basis for a multi-level cell (MLC), in which a single cell can represent  $n$  bits with  $2^n$  different states (levels) corresponding to different resistance values between the two extremes. As the number of levels is increased, each state is represented by a narrower resistance range.

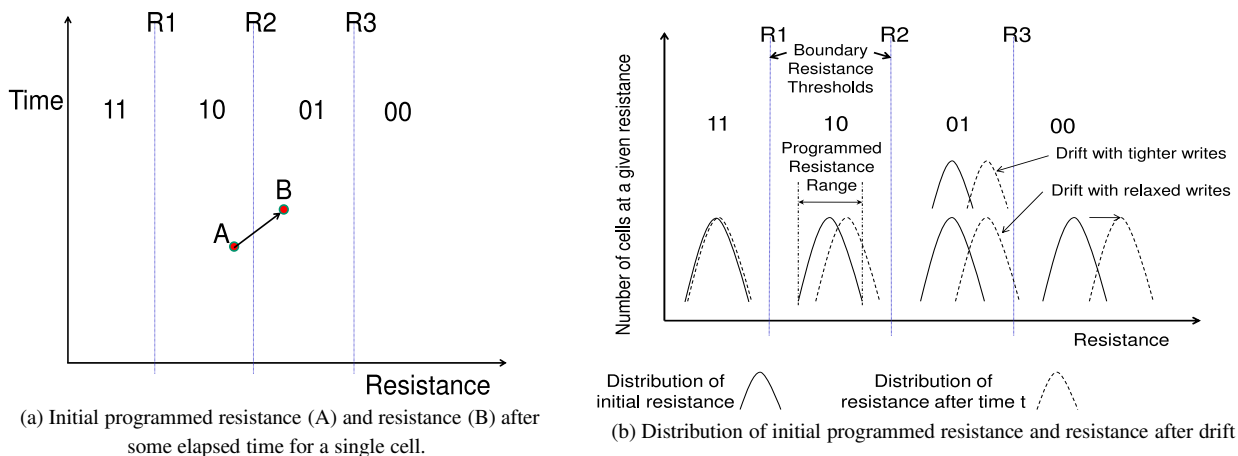
Each state in an MLC has “boundary resistance thresholds”, i.e., if a cell has a resistance between these thresholds, it is said to represent the corresponding state. Due to either mechanical relaxation forced by the reduced density in the amorphous state, or the existence of a large concentration of low mobility states [3], the resistance of a cell tends to increase with time, leading to the phenomenon of *resistance drift*. As represented in Figure 1(a), a given cell may initially have a resistance  $A$  that eventually drifts to a higher resistance value  $B$ . If the cell’s initial resistance represented the value 10, due to drift, its eventual resistance may represent the value 01, thus introducing a fault. Such drift-based faults are expected to be the primary and a frequent source of soft errors in future multi-level PCM cells [3].

In a recently filed patent application, Jeong et al. [5] use the following equation to measure resistance drift. Given an initial resistance of  $R_0$ , the resistance  $R_{drift}(t)$  can be calculated as:  $R_{drift}(t) = R_0 \times t^\alpha$ , where  $\alpha$  represents the drift exponent and  $t$  is the time elapsed (in seconds) since programming the cell. The range of initial programmed resistance  $R_0$  for a state usually follows a Gaussian distribution (solid lines in Figure 1(b)) [2, 6, 11]<sup>1</sup>. Similarly,  $\alpha$  also follows a normal distribution for each state, with the mean being higher for more amorphous states. In other words, states with a higher resistance tend to drift faster; in Figure 1, the 01 state is considered the most drift-prone as it has the highest resistance and is in danger of drifting into the 00 state. Each write results in a normally distributed  $R_0$  and  $\alpha$ , so every cell drifts into the neighboring state after a variable and hard-to-predict time interval. These unique drift properties require novel solutions to tolerate this source of PCM soft errors.

## 3 Proposed Solutions

We first consider a simple “scrubbing” operation (similar to refresh in DRAM) to tackle the drift problem. Unfortunately, such scrubbing would be limited by the behavior of the worst-case cell. For example, if we assume that the write process is precise enough that  $R_0$  lies within  $2.75 \sigma$  of its mean for the 01 state, the worst-case cell can drift in about 1.8 seconds [5, 11]. To prevent errors, every line must be scrubbed within 1.8 seconds, requiring that 600+ cache lines be simultaneously refreshed in a 64 GB system, and exhausting a cell’s lifetime in well under five years. We clearly need to improve upon this impractical baseline model and we describe solutions that can be applied at different levels of the system stack.

<sup>1</sup>In future discussions,  $\sigma$  is assumed to be the standard deviation for this distribution.



**Figure 1. Illustration of the resistance drift phenomenon**

**Device Level Solutions:** If a write can be made more precise, i.e., if the worst-case initial resistance is forced to lie closer to the mean, the worst-case cell will take longer to drift to the adjacent state. However, a precise write requires that the cell be programmed with many pulses of low intensity, so that each pulse causes a relatively small decrement in resistance (resistance change at higher resolution) [8]. This means that the write process is slower and causes greater wearout. This higher cost of the write must be balanced against the lower scrubbing frequency. It also turns out that a precise write is more energy-efficient (since energy is a function of  $I^2R$ ). Alternatively, instead of splitting the resistance range equally among all states, the resistance range can be split unequally, so that drift-prone states have a wider acceptable resistance range.

**Architecture Level Solution:** We introduce a refresh mechanism that is based on periodic “*Light Array Reads for Drift Detection (LARDDs)*” and ECC support. The line is re-written only when the LARDD reveals that the line has errors. The ECC support allows us to recover from the error(s). This makes the mechanism *reactionary*, instead of having the worst-case cell dictate the refresh frequency for every line. As a further optimization, we can add stronger support for multi-bit ECC, and introduce a “*headroom*” scheme that triggers a refresh operation well before the maximum number of correctable errors is reached. Depending on the LARDD frequency, the number of correctable errors, and the provided headroom, a variety of solutions with different error rates, energy overheads, storage overheads, and endurance can be created. For example, more headroom implies lower soft error rates, but it increases the frequency of writes, and hence accelerates the onset of hard errors – this represents a new trade-off for PCM system design.

**Circuit Level Support:** While strong ECC support is required to recover from multiple errors in a line, it would be very expensive to use ECC algorithms to detect errors on every LARDD. Hence, we adopt an approximate circuit solution to detect the onset of a large number of errors and trigger a scrub. The approximate solution simply maintains a count of the drift-prone cells in each word of a cache line, similar to a parity field. Every parity mis-match implies at least one error and a scrub is invoked well before we reach the maximum number of correctable errors for that line. The headroom is required to tolerate the approximate nature of this error detection circuitry.

**System Level Solutions:** As mentioned earlier, there are several parameters in the basic solution involving LARDDs and ECC support. As the LARDD frequency and headroom are varied, we see varying energy, latency, soft error rates, and cell lifetimes. System level solutions will be required to monitor error rates, wearout, and energy budgets, and set the parameters for the refresh mechanisms. This is especially crucial as the drift times and error rates may vary as a function of temperature and number of hard errors in the cache line.

Our results show that each approach has individual merit. These solutions can allow up to 39% reduction in energy, yield four orders of magnitude fewer errors, and achieve up to 102-fold increase in lifetime, all with tolerable overheads for ECC support.

## References

- [1] Micron Omneo P5Q Phase Change Memory <http://numonyx.com/en-US/MemoryProducts/PCM/Pages/P5Q.aspx>.
- [2] M. Boniardi and et al. Statistical and scaling behavior of structural relaxation effects in phase-change memory (PCM) devices. 2009.
- [3] G. W. Burr and et al. Phase Change Memory Technology, 2010. <http://arxiv.org/abs/1001.1164v1>.
- [4] S. Cho and H. Lee. Flip-N-Write: A Simple Deterministic Technique to Improve PRAM Write Performance, Energy, and Endurance. In *Proceedings of MICRO*, 2009.
- [5] C.-W. Jeong and et al. Multiple Level Cell Phase-Change Memory Devices Having Controlled Resistance Drift Parameter, Memory Systems Employing Such Devices and Methods of Reading Memory Devices, 2008. United States Patent Application, Number US 2008/0316804 A1.
- [6] S. Kang and et al. A 0.1- $\mu\text{m}$  1.8-V 256-Mb Phase-Change Random Access Memory (PRAM) With 66-MHz Synchronous Burst-Read Operation. *Solid-State Circuits, IEEE Journal of*, 42(1), jan. 2007.
- [7] B. Lee and et al. Architecting Phase Change Memory as a Scalable DRAM Alternative. In *Proceedings of ISCA*, 2009.
- [8] J.-T. Lin and et al. Design optimization in write speed of multi-level cell application for phase change memory. In *Intl. Conf. of Electron Devices and Solid-State Circuits*, 2009.
- [9] M. Qureshi and et al. Scalable High performance Main Memory System Using Phase-Change Memory Technology. In *Proceedings of ISCA*, 2009.
- [10] S. Schechter and et al. Use ECP, not ECC, for hard Failures in Resistive Memories. In *Proceedings of ISCA*, 2010.
- [11] W. Xu, J. Liu, and T. Zhang. Data Manipulation Techniques to Reduce Phase Change Memory Write Energy. In *Proceedings of ISLPED*, 2009.