Can Multi-Level Cell PCM Be Reliable and Usable? Analyzing the Impact of Resistance Drift

Sungkap Yeo
sungkap@gatech.edu

Nak Hee Seong
nhseong@ece.gatech.edu

Hsien-Hsin S. Lee
leehs@gatech.edu

School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332

ABSTRACT

There are several emerging memory technologies looming on the horizon to compensate the physical scaling challenges of DRAM. Phase change memory (PCM) is one of such candidates proposed for being part of the main memory in computing systems. One salient feature of PCM is its multi-level cell (MLC) property which can be used to multiply the memory capacity at the cell level. However, due to the nature of PCM that the value written to the cell can drift over time, PCM is prone to a unique type of soft errors, posing a great challenge for their practical deployment. To address this reliability issue, many researchers proposed material-based or architectural solutions. In this paper, we analyze the resistance drift problem using both analytical models and Monte Carlo simulation and show the fundamental limit in prior architectural solutions. According to our findings, four-level PCM is unusable given its soft error rate and scrubbing time needed.

1. INTRODUCTION

Phase-change memory (PCM) is a promising alternative memory technology for future computing systems. Based on chalcogenide compound made of Ge, Sb, and Te (GST), the value of stored data is represented using its material state indicated by its current resistance level. When a PCM cell is heated up to a temperature over the melting point and cooled down within several tens of nanoseconds, the cell becomes a high resistive amorphous state. On the other hand, the PCM cell becomes a low resistive crystalline state when it is exposed to a temperature lower than the melting point and cooled down slowly. The resistance of a PCM cell is known to be 10^3 Ohms in the crystalline state and 10^6 Ohms in the amorphous state. Moreover, when we alter the temperature and the duration induced to the PCM cell, researchers found that the resistance can be anywhere in between these two states. Multi-level cell (MLC) PCM exploits these intermediate states in-between the crystalline and amorphous states to store more data per cell.

Although the MLC PCM increases information density, this technique requires a finer-grain control over the resistance of a cell. To make the resistance of a cell within a predefined range, The MLC PCM requires an iterative-writing mechanism, which reads the resistance immediately followed by a write to check whether the cell needs to be rewritten or not. This iterative-writing compromises the write latency. Recent studies show that the write latency of a four-level cell is about 4x $\sim$ 8x slower than that of a Single Level Cell (SLC) PCM [11]. Besides the performance issue, a far more critical problem of making MLC PCM practical is its reliability concern caused by the resistance drift. The resistance drift is the phenomenon that the resistance of a PCM cell increases over time. Such drifting was not a problem in SLC PCM because the rate of resistance drift is proportional to the initial resistance of the cell and is nearly-zero for the crystalline state. However, researchers found that the resistance of cells at the the intermediate states written in the MLC PCM can cross the state boundary and lead to undesirable errors due to state changes. This new type of soft errors caused by resistance drift, if left unaddressed, will make MLC PCM completely useless. In this paper, we present the first attempt to mathematically formulate the soft error rates of MLC PCM. With this analytical model, we evaluate the previously proposed ideas for reducing errors and show that four-level PCM is an infeasible solution as main memory.

Table 1: Configuration Variables of Four-Level Cell PCM When $t_0 = 1$ s.

Storage Level	Data	$\log_{10} R$		α	
		μ_R	σ_R	μ_α	σ_α
0	01	3.0	$\frac{1}{6}$	0.001	$0.4 \times \mu_\alpha$
1	11	4.0		0.02	
2	10	5.0		0.06	
3	00	6.0		0.10	

2. ANALYTICAL ERROR MODEL AND VALIDATION

By measuring the resistance drift of PCM reset and set states from iterative experiments, Ielmini *et al.* [7, 8] found that the drift can be represented with a power-law model shown as the following:

$$R_{drift}(t) = R \times \{\frac{t}{t_0}\}^\alpha \qquad (1)$$

where R and t_0 are normalization constants and α is a drift exponent. Because the main cause of the drift is the structural relaxation of the amorphous state, the drift exponent of the reset state is much larger than that of the set state in the experiments. In other words, the drift exponent will increase as a portion of the amorphous state in a PCM cell increases.

As mentioned earlier, the resistance drift causes soft errors in the multi-level-cell (MLC) PCM. To estimate the reliability impact of resistance drift, we make the following assumptions for the normalization constants and a drift exponent for each storage level. According to the experiments of Nirschl *et al.* [9], the iterative write-and-verify sequence adjusts the programmed resistance R_p to be located within a desired resistance range for a given storage level, where $\log_{10} R_p$ follows a normal (Gaussian) distribution. In this paper, we assume that the logarithm of a normalization resistance, $\log_R$ will follows a normal distribution of $N(\mu_R, \sigma_R^2)$. In addition, a desired programmed resistance range for a given state is set to the range within $10^{\mu_R \pm 2.75\times\sigma_R}$ Ω and the upper and lower sensing boundaries for the state are set to $10^{\mu_R \pm 3\times\sigma_R}$ Ω. The value of a drift exponent is also assumed to follow a normal distribution of $N(\mu_\alpha, \sigma_\alpha^2)$. The parameters we use in our drift analysis are based on the previous works [1, 15] and described in Table 1.

In MLC PCM, a transient (soft) error can occur when the resistance of a cell is drifted above the upper boundary of its programmed state. From the state-boundary settings described above, the condition of a soft error can be represented as follows.

$$R_{drift}(t) > 10^{\mu_R + 3\times\sigma_R} \tag{2}$$

In other words, when considering the values in Table 1, the target resistance values for the four storage levels are 10^3, 10^4, 10^5, and $10^6\Omega$, respectively, and the three sensing boundaries between two adjacent levels are $10^{3.5}$, $10^{4.5}$, and $10^{5.5}\Omega$. For instance, when the resistance of a cell that was programmed for the storage level 2 becomes larger than $10^{5.5}$ Ω, the cell is sensed as the next storage level, which generates a soft error.

By using the assumption that $\log_{10} R$ and α follow normal distributions as shown in Table 1, we can calculate the probability of such soft error type. First, we define two more variables, $m = \log_{10} R$ and $n = \log_{10} t$. By substituting Equation (1) with m and n, we obtain the following.

$$\log_{10}(R_{drift}(t)) = \log_{10} R + \alpha \log_{10} t = m + n\alpha. \tag{3}$$

Thus, the condition of a soft error can be rewritten as follows.

$$\begin{aligned} m + n\alpha &> \mu_R + 3\sigma_R \\ n\alpha &> \mu_R + 3\sigma_R - m \end{aligned}$$

As α follows $N(\mu_\alpha, \sigma_\alpha^2)$, $n\alpha$ follows $N(n\mu_\alpha, (n\sigma_\alpha)^2)$. The probability for $n\alpha$ to be more than $\mu_R + 3\sigma_R - m$ can be calculated as follows.

$$\begin{aligned} &(\text{Probability of soft error for a given } m) \\ &= 1 - \Phi\left(\frac{\mu_R + 3\sigma_R - m - n\mu_\alpha}{n\sigma_\alpha}\right) \\ &\text{where } \Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-x^2/2} dx \end{aligned} \tag{4}$$

Here, we also take the effect of the iterative writing into account. As mentioned earlier, cell programming iterates a write-and-verify sequence until $\log_{10} R$ is less than $\mu_R + 2.75\sigma_R$ or larger than $\mu_R - 2.75\sigma_R$. It means the probability density function of a random variable m, $f(m)$ is as follows.

$$\begin{aligned} &f(m) = \begin{cases} \frac{1}{K}\phi(\frac{m-\mu_R}{\sigma_R}) & \mu_R - 2.75\sigma_R < m < \mu_R + 2.75\sigma_R \\ 0 & \text{otherwise,} \end{cases} \\ &\text{where } K = \int_{\mu_R+2.75\sigma_R}^{\mu_R-2.75\sigma_R} \phi(\frac{m-\mu_R}{\sigma_R}) dm, \\ &\text{and } \phi(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \end{aligned} \tag{5}$$

Therefore, we can obtain the probability of soft error as a function of time ($t = 10^n$) by integrating Equation (4) with a random variable m for $\mu_R - 2.75\sigma_R < m < \mu_R + 2.75\sigma_R$.

$$\begin{aligned} &(\text{Probability of soft error}) \\ &= \int_{\mu_R+2.75\sigma_R}^{\mu_R-2.75\sigma_R} (1 - \Phi(\frac{\mu_R + 3\sigma_R - m - n\mu_\alpha}{n\sigma_\alpha}))f(m) dm \end{aligned} \tag{6}$$

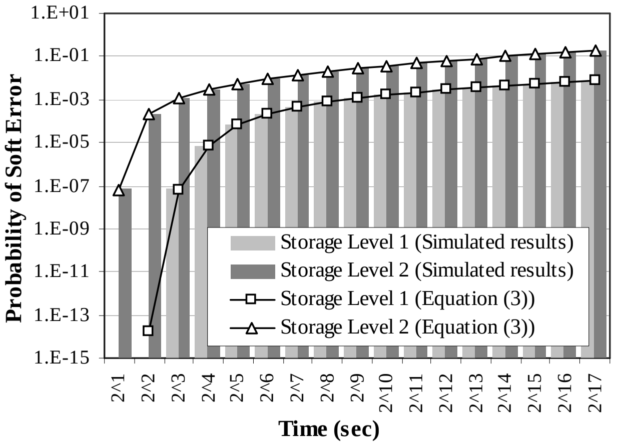

Figure 1: Probability of Soft Error of Four-level Cell PCM Over Time

We evaluate Equation (6) and also run Monte Carlo simulations to verify these equations. In the simulator, we first implement a random number generator, which draws normally distributed random numbers at given mean and variance. The main loop starts by picking R and α from their corresponding normal distributions in Table 1. If $\log_{10} R$ is less than $\mu_R - 2.75\sigma_R$ or larger than $\mu_R + 2.75\sigma_R$, the simulator picks R and α again for emulating write-and-verify (iterative writing) process. Once R and α are picked, the simulator calculates $R_{drift}(t)$ by using Equation (1). Finally, $R_{drift}(t)$ is evaluated whether $\log_{10} R_{drift}(t)$ is larger than $\mu_R + 3\sigma_R$ or not. By definition, $\log_{10} R_{drift}(t)$ larger than $\mu_R + 3\sigma_R$ generates a soft error. The simulator repeats the main loop for one billion times and collects the number of soft errors. For example, if the simulator finds ten soft errors out of one billion trials, the soft error rate becomes 10^{-8}. Figure 1 and Table 2 show the results side by side. We omit the soft error rates for set and reset states, *i.e.,* , the storage level 0 and 3, because (i) resistance drift in level-3 states does not lead to a soft error, and (ii) the error rates of level-0 states are too small to be evaluated and can be ignored. For example, Mathematica 8.0 shows the first non-zero error rate for level-0 states when $t = 2^{35}$ or 1090 years, and the error rate is 2.3×10^{-18}. Similarly, note that three data points for intermediate states (level-1 and level-2 states) are missing and marked as "too small" because either (i) Mathematica 8.0 cannot evaluate Equation (6) or (ii) Monte Carlo simulation found no error in one billion trials. By comparing results from two independent sources, we validate the accuracy of our theoretically derived Equation (6) by simulation. When the soft error rate is larger than 0.01%, the difference is negligible. For the lower soft error rates, the difference comes from insufficient trials in the Monte Carlo simulations. One billion trials were too small to detect such low error rates.

3. REVISITING FOUR-LEVEL CELL PCM

Given the soft error rates in Table 2, it is clear that without any

Table 2: Probability of Soft Error of Four-Level Cell PCM

	Storage Level 1		Storage Level 2	
Elapsed Time (sec)	Equation (6)	Simulation	Equation (6)	Simulation
2	(too small)	(too small)	5.85E-06%	7.40E-06%
2^2	1.59E-12%	(too small)	0.02%	0.02%
2^3	5.85E-06%	7.40E-06%	0.12%	0.12%
2^4	7.45E-04%	7.57E-04%	0.28%	0.29%
2^5	0.01%	0.01%	0.52%	0.53%
2^6	0.02%	0.02%	0.85%	0.86%
2^7	0.05%	0.05%	1.30%	1.31%
2^8	0.08%	0.08%	1.90%	1.91%
2^9	0.12%	0.12%	2.67%	2.68%
2^{10}	0.17%	0.17%	3.64%	3.66%
2^{11}	0.22%	0.22%	4.84%	4.87%
2^{12}	0.28%	0.29%	6.29%	6.32%
2^{13}	0.35%	0.36%	7.99%	8.04%
2^{14}	0.43%	0.44%	9.95%	10.01%
2^{15}	0.52%	0.53%	12.16%	12.24%
2^{16}	0.62%	0.63%	14.61%	14.70%
2^{17}	0.73%	0.74%	17.27%	17.38%

mechanism for reducing the soft error rates, 4LC-PCM is infeasible to be used as a main memory. Thus, researchers have proposed several drift-tolerant approaches such as error correction schemes [1, 16, 10, 15], data encoding schemes using relative resistance difference [10, 16], a reference cell scheme [6], a time-aware drift estimation scheme [15], and most recently an efficient scrubbing scheme [1]. Among them, we focus on the most recent work by Awasthi *et al.* They proposed an architectural mechanism combining a memory scrubbing scheme with a strong error-correction scheme, which achieves lower soft error rates than others aiming to use PCM for main memory in systems. However, as we will show subsequently, even with the most efficient scrubbing mechanism, the soft error rate of 4LC-PCM is still much higher than that of DRAM. DRAM experiences soft errors mainly induced by particle strikes and its soft error rate (SER) is known to be an average of $25,000 \sim 75,000$ FIT (failures in time per billion hours of operation) per Mbit, *i.e.*, $25 \times 10^{-12} \sim 75 \times 10^{-12}$ per bit-hour [12].

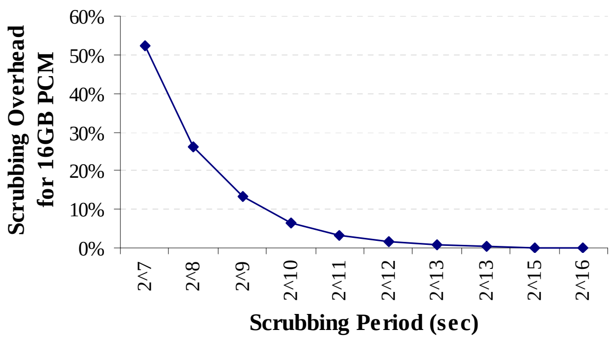

Figure 2: Scrubbing Period Versus Scrubbing Overhead

3.1 Estimating Scrubbing Overhead

In this section, we compare the SER of 4LC-PCM to that of contemporary DRAM technology and argue that 4LC-PCM is not feasible for main memory due to reliability concern. First, we assume a 16GB PCM main memory using a 256B data block[1] as a basic access unit as assumed in the prior literature [14, 13]. According to the recent paper of Choi *et al.* [3], the read and write latencies in SLC PCM are $120ns$ and $150ns$, respectively. Considering iterative write-and-verify steps are required for MLC PCM, however, we assume that scrubbing one cache line takes at least $1\mu s$. Also, we assume that each storage level has the same probability of occurrences.

Figure 2 shows the scrubbing overhead as the scrubbing period increases. Here, the scrubbing overhead is defined as (Time used for scrubbing)/(Scrubbing period). The 16GB PCM has 64M cachelines. Thus, 67.1 seconds ($= 64M \times 1\mu s$) are required for scrubbing the entire physical PCM. If we use the scrubbing period of 45 minutes as in the DRAM memory system for real servers [12], the SER of a PCM cell programmed to storage level 2 becomes around 5%, which is significantly higher than that of DRAM. New memory technologies must have the similar level of SER as that of DRAM. However, here we show that 4LC-PCM with scrub mechanisms cannot satisfy such conditions. As shown in Table 2, even when the memory controller performs nothing but scrubbing (100% overhead, in other words, the memory controller will not have time to respond to any memory request), the SER of storage level 2 in 4LC-PCM is about 0.9% which is still significantly high. To main a very low SER and reduce the scrubbing overhead simultaneously, the maximum PCM capacity must be limited. Our next section will show the largest capacity of 4LC-PCM the scrubbing mechanism can support for different combinations of target SER and scrubbing overhead.

3.2 Reducing Capacity to Achieve Low Soft Error Rates

Another way of lowering SER of 4LC-PCM is to limit the maximum capacity. We assume the capacity of 4LC-PCM as 16GB in Section 3.1 when estimating the scrubbing overhead. Because the scrubbing overhead proportionally increases with the capacity, assuming 8GB of capacity results in halving the overhead. If we fur-

[1]A large last-level DRAM cache is typically used to compensate for the relatively slower PCM access latencies. Its cacheline size is assumed to be 256B

Table 3: Maximum Capacity of Four-Level Cell PCM by Soft Error Rates and Scrubbing Overhead

		Scrubbing Overhead		
Scrubbing Period (sec)	$SER_{combined}$	100.0%	12.5%	1.0%
2	1.46E-06%	488MB	61.0MB	4.88MB
2^2	0.005%	977MB	122MB	9.77MB
2^3	0.030%	1.95GB	244MB	19.5MB
2^4	0.071%	3.91GB	488MB	39.1MB
2^5	0.132%	7.81GB	977MB	78.1MB

ther reduce the capacity, we can achieve lower SER. Table 3 shows the results. In Table 3, we calculate the maximum capacity of 4LC-PCM for different combinations of SER and scrubbing overhead. The leftmost column represents the scrubbing period for each 256B memory block. The next column represents the combined SER, which is an average of SER of all four states in 4LC-PCM. However, because the third storage level shows significantly larger SER than the other levels, this combined SER is close to one fourth of the third storage level's SER. In addition, we show the maximum capacity by each given scrubbing overhead. When the overhead is 100%, the memory controller cannot service any request from the upper memory hierarchy. Since 100% scrubbing overhead is impractical, the third column of Table 3 can be viewed as an upper bound. The table also shows the maximum capacity when the scrubbing overhead are set to 12.5% and 1.0%, respectively. For example, if we design 4LC-PCM with the scrubbing overhead of 1.0%, leaving 99% of the time for servicing memory requests, the maximum PCM capacity will be merely 4.88MB for achieving an average of 1.46E-06% SER. Note that when 4LC-PCM comprises multiple ranks or banks, scrubbing can be performed in parallel. Thus, when one bank is being scrubbed, the other banks can respond to requests from the CPU. However, even with four ranks with four banks each, the maximum capacity amounts to 78.1MB, which is still substantially below the main memory capacity required in any computing system. In sum, although lower SER can be achieved by reducing the capacity of 4LC-PCM, then the maximum capacity becomes totally unusable.

3.3 Using Error-Correcting Codes

Error-correcting codes (ECC) can be applied to compensate the SER of 4LC-PCM. For example, the industry standard (72,64) Hamming code [4] can correct single bit errors by adding 8 redundant bits on top of 64 bits data.[2] This scheme is commonly found in main memory of server systems because of the simplicity in encoding and decoding. Moreover, stronger ECC can also be used to protect data from multiple bit errors. For example, BCH codes [2, 5] correct 8, 16, 24, or 40 bits errors from 256, 512, 1024 bytes of data depending on the size of the redundant bits. Because decoding BCH codes require more computing power and time than (72,64) Hamming code, these codes are not frequently used for latency-sensitive devices such as main memory but commonly used in slower devices such as NAND-based storage. With the combined SER for each cell of 4LC-PCM developed in previous sections, we calculate the error rates after applying (72,64) Hamming code and various BCH codes. Note that for every ECC evaluated in this section, we fix the data size as 256 bytes.

(72,64) Hamming code corrects one bit error, and thus, having more than two bit errors among 72 bits is incorrectable. In addition, since storing 72 bits requires 36 4LC-PCM cells, the probability of having more than two bit errors out of 36 cells can be calculated as follows. Note that by using grey codes as in Table 1, one step change in storage levels can be limited to affect only one bit in two-bit data. Thus, two bit errors can happen only when two 4LC-PCM cells are changed due to resistance drift.

$$\begin{aligned}&\text{Probability of having at least two bit errors}\\ =&P_{error}(64b) = 1 - P(\text{no errors}) - P(\text{one bit error})\\ =&1-(1-SER_{combined})^{36}\\ &-\binom{36}{1}(1-SER_{combined})^{35}(SER_{combined})\end{aligned} \tag{7}$$

Now we calculate the probability of incorrectable errors in 256 bytes. 256 bytes comprises 32 of 64 bits data, therefore, to reconstruct the entire 256 bytes, all 32 blocks should not generate any error. If we define the result of Equation (7) as $P_{error}(64b)$, then the probability of incorrectable error for 256 bytes is defined as follows.

$$P_{error}(256B) = 1-(1-P_{error}(64b))^{32} \tag{8}$$

The fourth column in Table 4 shows the results. In Table 4, we take the scrubbing period, scrubbing overheads, and SER from Table 2 and calculate the probability of incorrectable errors. When the error rates are compared to that without ECC, (72,64) Hamming code reduces the error rates, but those rates are still too high for practical use. The results indicate that 4LC-PCM must use stronger ECC that requires more redundant bits and higher computational overheads.

Now we calculate the probability of incorrectable errors with stronger ECC. On top of 256 bytes of data, BCH-8 corrects up to 8 bits errors by adding 12 redundant bytes, and BCH-16 corrects up to 16 bits errors by adding 24 redundant bytes.[3] We generalize Equation (7) for calculating the probability of having at least n bit errors out of m bits as follows.

$$\begin{aligned}&\text{Probability of having at least } n \text{ bit errors out of } m \text{ bits}\\ &= 1-\sum_{k=0}^{n-1}\binom{m}{k}(1-SER_{combined})^{m-k}(SER_{combined})^{k}\end{aligned} \tag{9}$$

Table 4 also shows the results from Equation (9). When the scrubbing period is 2^7 seconds, the scrubbing overhead is 52.4%, and $P_{error}(256B)$ is 0.949% for BCH-8 and 2.96×10^{-5}% for BCH-16. These error rates are significantly smaller than that of 4LC-PCM with (72,64) Hamming code; however, still $10^5 \sim 10^8$ times higher than $P_{error}(256B)$ of DRAM without ECC support.

This section shows that 4LC-PCM requires ECC schemes stronger than BCH-16. Implementing BCH-24 or BCH-32 is a common practice in some applications with high soft error rates. For example, MLC-NAND based devices implement ECC stronger than BCH-16. However, MLC-NAND based devices could implement rich ECC since they are not latency sensitive and only transfer a few tens of mega bytes per second. On the contrary, 4LC-PCM is latency sensitive and transfers more than a few giga bytes per second as main memory of a system. All in all, such requirements for rich ECC prevent 4LC-PCM from being used as main memory of commodity systems because of the high cost and performance issues. Firstly, implementing a memory controller with complex error-correcting mechanisms is expensive. Since the current trend is to integrate memory controllers on the same die with the processor cores, vendors need to design and fabricate a separate CPU for

[2]The capacity overhead is 12.5%.

[3]The capacity overheads are 4.7% and 9.4%, respectively.

Table 4: **Probability of Incorrectable Errors by ECC and $SER_{combined}$ for 16GB 4LC-PCM**

		Probability of Incorrectable Errors for 256 Bytes $= P_{error}(256B)$					
Scrubbing Period (Overheads)	$SER_{combined}$	No ECC	$(72, 64)$	BCH-8 (256B+12B)	BCH-16 (256B+24B)	BCH-24 (256B+36B)	BCH-32 (256B+48B)
2^7 seconds (52.4%)	0.325%	96.4%	18.0%	0.949%	2.96E-5%	4.11E-11 %	(too small)
2^8 seconds (26.2%)	0.475%	99.2%	33.7%	7.38%	4.00E-3%	1.09E-7%	6.24E-12%
2^9 seconds (13.1%)	0.668%	99.9%	54.3%	29.2%	0.184%	6.68E-5%	3.65E-9%
2^{10} seconds (6.6%)	0.91%	100%	75.1%	64.0%	3.08%	1.09E-2%	6.17E-6%
2^{11} seconds (3.3%)	1.21%	100%	90.3%	90.0%	20.5%	0.53%	2.43E-3%
2^{12} seconds (1.6%)	1.57%	100%	97.6%	98.7%	58.9%	7.83%	0.22%

supporting 4LC-PCM. A complex memory controller requires chip area and design effort, which increases the chip cost. Secondly, the higher computational overhead in decoding increases the memory latency and degrades the performance. As a result, the majority of commodity systems with DRAM as main memory do not even implement (72,64) Hamming code. We argue that suggesting stronger ECC mechanism only limits the application of PCM.

4. CONCLUSION

This paper reveals that achieving error rates of DRAM with 4L-PCM is infeasible in practice. We first present the analytical model for calculating SER of MLC PCM. Our model takes the following things into account; (1) the effect of the resistance drift, (2) the distribution functions of the resistance at $t_0 = 1s$, (3) the distribution functions of the rate of resistance drift, and (4) the effect of iterative writing. The model is verified by comparing the theoretically derived results to the results from Monte Carlo simulations. In addition, we use mean and deviation of distribution functions from other studies to show the relationship among the SER, scrubbing periods, and scrubbing overheads for 4LC-PCM. Further analysis shows that 4L-PCM cannot be used as main memory given its high error rates and scrubbing overheads. The most critical problem of 4L-PCM is high SER of the third storage level, which is about $10^9 \sim 10^{11}$ times higher than that of DRAM. With all our in-depth analysis, due to resistance drift, 4L-PCM is either unreliable for practical deployment or one has to limit its capacity to some unreasonable small size, both indicating that main memory based on 4L-PCM (or PCM with more levels) cannot be reliable and usable at the same time. More research is called for to investigate other novel alternatives to exploit the use of intermediate resistance states and take advantage of them beyond SLC PCM.

5. ACKNOWLEDGMENT

This research was sponsored by NSF CAREER Award CNS-0644096, an IBM Faculty Award and gift from Intel Corporation. The authors would also like to thank the support of Dr. Jude Rivers and Dr. Viji Srinivasan from IBM Research.

6. REFERENCES

[1] M. Awasthi, M. Shevgoor, K. Sudan, B. Rajendran, R. Balasubramonian, and V. Srinivasan. Efficient scrub mechanisms for error-prone emerging memories. In *Proceedings of the International Symposium on High Performance Computer Architecture*, 2012.

[2] R. Bose and D. Ray-Chaudhuri. On a class of error correcting binary group codes. *Information and control*, 3(1):68–79, 1960.

[3] Y. Choi, I. Song, M.-H. Park, H. Chung, S. Chang, B. Cho, J. Kim, Y. Oh, D. Kwon, J. Sunwoo, J. Shin, Y. Rho, C. Lee, M. G. Kang, J. Lee, Y. Kwon, S. Kim, J. Kim, Y.-J. Lee, Q. Wang, S. Cha, S. Ahn, H. Horii, J. Lee, K. Kim, H. Joo, K. Lee, Y.-T. Lee, J. Yoo, and G. Jeong. A 20nm 1.8V 8Gb PRAM with 40MB/s Program Bandwidth. In *Proceedings of the 2012 IEEE International Solid-State Circuits Conference*, 2012.

[4] R. Hamming. Error detecting and error correcting codes. *Bell System Technical Journal*, 29(2):147–160, 1950.

[5] A. Hocquenghem. Codes correcteurs d'erreurs. *Chiffres*, 2(2):147–156, 1959.

[6] Y. Hwang, C. Um, J. Lee, C. Wei, H. Oh, G. Jeong, H. Jeong, C. Kim, and C. Chung. Mlc pram with slc write-speed and robust read scheme. In *VLSI Technology (VLSIT), 2010 Symposium on*, pages 201–202. IEEE, 2010.

[7] D. Ielmini, A. Lacaita, and D. Mantegazza. Recovery and drift dynamics of resistance and threshold voltages in phase-change memories. *Electron Devices, IEEE Transactions on*, 54(2):308–315, 2007.

[8] D. Ielmini, S. Lavizzari, D. Sharma, and A. Lacaita. Physical interpretation, modeling and impact on phase change memory (pcm) reliability of resistance drift due to chalcogenide structural relaxation. In *Electron Devices Meeting, 2007. IEDM 2007. IEEE International*, pages 939–942. IEEE, 2007.

[9] T. Nirschl, J. Phipp, T. Happ, G. Burr, B. Rajendran, M. Lee, A. Schrott, M. Yang, M. Breitwisch, C. Chen, et al. Write strategies for 2 and 4-bit multi-level phase-change memory. In *IEEE International Electron Devices Meeting, 2007. IEDM 2007*, pages 461–464, 2007.

[10] N. Papandreou, H. Pozidis, T. Mittelholzer, G. Close, M. Breitwisch, C. Lam, and E. Eleftheriou. Drift-tolerant multilevel phase-change memory. In *Memory Workshop (IMW), 2011 3rd IEEE International*, pages 1–4. IEEE.

[11] M. Qureshi, M. Franceschini, and L. Lastras-Montano. Improving read performance of phase change memories via write cancellation and write pausing. In *Proceedings of the International Symposium on High Performance Computer Architecture*, 2010.

[12] B. Schroeder, E. Pinheiro, and W. Weber. Dram errors in the wild: a large-scale field study. In *Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems*, pages 193–204. ACM, 2009.

[13] N. H. Seong, D. H. Woo, and H.-H. S. Lee. Security Refresh: Protecting Phase-Change Memory against Malicious Wear Out. *IEEE Micro*, 31(1):119–127, 2011.

[14] N. H. Seong, D. H. Woo, V. Srinivasan, J. A. Rivers, and H.-H. S. Lee. SAFER: Stuck-at-fault error recovery for memories. In *Proceedings of the 43rd IEEE/ACM International Symposium on Microarchitecture*, 2010.

[15] W. Xu and T. Zhang. A time-aware fault tolerance scheme to improve reliability of multilevel phase-change memory in the presence of significant resistance drift. *IEEE transactions on very large scale integration (VLSI) systems*, 19(8):1357–1367, 2011.

[16] W. Zhang and T. Li. Helmet: A resistance drift resilient architecture for multi-level cell phase change memory system. In *IEEE/IFIP International Conference on Dependable Systems & Networks*, pages 197–208, 2011.