Memory Module-level Testing and Error Behaviors for Phase Change Memory

Zhe Zhang*, Weijun Xiao†,‡, Nohhyun Park*, and David J. Lilja*
* Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN, 55455
† Department of Electrical and Computer Engineering, Virginia Commonwealth University, Richmond, VA, 23284
‡ Corresponding author: wxiao@vcu.edu
Email: zhan0915@umn.edu, wxiao@vcu.edu,{parkx408, lilja}@umn.edu

***Abstract*—Phase change memory (PCM) is a promising technology to solve energy and performance bottlenecks for memory and storage systems. To help understand the reliability characteristics of PCM devices, we present a simple fault model to categorize four types of PCM errors. Based on our proposed fault model, we conduct extensive experiments on real PCM devices at the memory module level. Numerical results uncover many interesting trends in terms of the lifetime of PCM devices and error behaviors. Specifically, PCM lifetime for the memory chips we tested is greater than 14 million cycles, which is much longer than for flash memory devices. In addition, the distributions for four types of errors are quite different. These results can be used for estimating PCM lifetime and for measuring the fabrication quality of individual PCM memory chips.**

I. Introduction

Phase Change Memory (PCM) is one of the emerging memory technologies that has received a lot of attention from both the academic and industrial communities in recent years. Currently, Samsung Electronics and Micron Technology have made commerical PCM chips available in the market. Architectural efforts show great potential to use PCM as a DRAM alternative for future memory systems because PCM is non-volatile memory and the latency is close to DRAM. Compared to Flash Memory, PCM is byte-addressable with a much longer lifetime and much faster read and write cycles. It is also not necessary to explicitly erase before writing PCM cells. However, most of the previously published work on PCM devices are based on simulations [1], [2], [3]. Little is known in the open literature about how PCM errors behave in real devices and the distributions of the errors.

In this paper, we work on real PCM chips for reliability testing. We develop a simple fault model to characterize PCM errors into four categories: Write Error (Type I and II), Program Interference Error, and Read Error. Based on this fault model, we propose an effective March Test algorithm to detect PCM errors [4]. By measuring and analyzing different types of PCM errors, we characterize error behaviors in terms of the lifetime (i.e., time-sensitive behavior) and error distribution. To the best of our knowledge, this is the first work for PCM testing and error behaviors on the existing PCM devices. Our numerical results reveal many interesting observations for PCM reliability, which could be very helpful for understanding PCM physical properties in terms of performance and reliability and developing error correction coding for PCM. The main contributions of the paper are summarized as follows.

- We characterize the lifetime and error behaviors and find that the life time of PCM is more than 14 million cycles for the chips we tested. In addition, the Write Error (Type I) increases linearly as the program cycle increases. The other three types of errors occur much less frequently than the Write Error (Type I).
- We analyze the distributions of PCM errors and find that the Write Error (Type I) eventually becomes a permanent error and remains stuck at zero forever. It appears as a transient error at first and then gradually becomes permanent as the PCM is continuously cycled. Write Errors (Type II) and Program Interference Errors decrease exponentially in time. Read Errors are transient and typically last for only several cycles.
- The Write Error (Type I) is due to over-cycling while the other three types of errors are not significantly impacted by over-cycling. Based on this observation, the Write Error (Type I) could be used as a lifetime estimator for PCM devices.
- The Write Errors (Type II) provide an indication for estimating the fabrication quality of individual PCM memory chips because PCM cells will have more Type II errors when they contain impurities.

The rest of the paper is organized as follows. Section 2 describes the operation of PCM devices. In Section 3, we develop a simple fault model for PCM devices. Our testing platform and methodology are presented in Section 4 and our experimental results are discussed in Section 5. We conclude the paper in Section 6.

II. Phase Change Memory

Phase Change Memory is a type of non-volatile memory that uses two structural states to represent two logical values. There is a chalcogenide layer (i.e. GST) in a PCM cell. This layer can be suitably changed between two phases (amorphous and crystalline) either through current or plasma heating of the chalcogenide material. Most PCM devices use current heating method.

The crystalline state, which is low resistance for the cells, is interpreted as a logic value '1' while the amorphous state with high resistance in cells is interpreted as a logic value '0' [7]. The amorphous phase is accomplished by heating the

978-1-4673-3052-7/12/$31.00 ©2012 IEEE

material rapidly beyond its melting point (Tm) and cooling it down to the glass transition temperature (Tg) within a very short time. The crystalline phase, on the other hand, is obtained by heating the material to a point between the melting temperature and the glass transition temperature with a relatively large programming duration. The programming pulse lasts for aproximately 10ns for the amorphous phase and 50ns [8] for the crystalline phase. The typical melting temperature of the material is about 600°C and the glass transition is about 300°C [9].

III. PCM Fault Model

During the manufacturing process, spot defects and contaminants between the cell heater and the chalcogende material might result in unreliable data [10]. In this section, we develop a simple fault model that incorporates all of the faults that occurred in our experiments and explain the possible reasons for these faults. We characterize PCM errors into four types: Write Error (Type I) (WE(I)), Write Error (Type II) (WE(II)), Program Interference Error (PI), and Read Error (RE). Although both WE(I) and WE(II) occur due to write operations on PCM, they have different behaviors. We partitioned these errors into two separate types and analyzed them individually. The detailed descriptions of the four types of PCM errors are as follows.

- Write Error (Type I): Writing '1' into a cell but the actual data afterward in the cell is '0'. The reason for this error is that the GST material of the PCM cell might loosen from its top electrode and hence incur an open circuit in the cell [11].
- Write Error (Type II): Writing '0' into a cell but the actual data afterward in the cell is '1'. Two reasons could cause this error. The first reason is that the GST material could be overheated during programming and consequently intermixed with ajacent cells [7]. If this occurs, the cell will exhibit low resistance all the time and never flip to zero again. The second reason is that the contaminants in the cell could result in parallel paths to conductive material in the cell. This current leakage eventually reduces the overall resistance of the cell [12].
- Program Interference Error: Data in a cell is altered because of programming operations in adjacent cells [6]. This failure is caused by thermal crosstalk. Because of defects or bad isolation, when you are programming a cell, the adjacent one might be influenced by thermal leakage, which alters the adjacent cell.
- Read Error (RE): Data in a cell is changed because of reading operations in the current location. This failure can be due to several possible causes. First of all, when a cell is read immediately after it has been programmed [13], there will have a delay for the cell resistance to reach equilibrium state. Because of this delay, the returned value from the read operation is different from the actual data when the cell becomes stable. In our current testing platform, the time interval between a write operation and a read operation is longer than the equilibrium transition delay of the cell, therefore this error never happens for this reason in our system. The second cause of this type of error is that the presence of defects or over-cycling could overheat the cell when running read operations eventually causing the bit to flip [7]. The third cause is that it is possible to mis-activate the current circuit for writing instead of the one for reading [14]. In this case, the generated thermal energy is much larger than from read operation, which can alter a bit in the cell.

IV. Testing Platform and Methodology

Our testing platform consists of three parts: a CPU core, a PCM memory controller, and a PCM daughter board. We use the Xilinx Spartan 6 FPGA to implement the MicroBlaze as the CPU core. The core then connects to the PCM memory controller through a peripheral bus interface (PLB).

A. CPU and PCM memory controller

Xilinx has its own soft core called MicroBlaze, which can run at 100 MHz. We use it as the CPU in our system and attach it to a peripheral through a Peripheral Local Bus (PLB). The PLB is connected to all peripherals such as a UART, Serial Ports as well as We designed a custom custom interfaces. We designed a custom daughter board that is connected to the FPGA board through a 40-pin FMC connector.

The PCM memory controller, which is attached to the PLB, is running at 66.67 MHz. We developed a finite state machine to perform single read and write operations on the PCM chips. Both single read and write operations need 16 bus cycles. A PCM device driver has been implemented for complicated read/write operations based on the datasheet from the manufacturer. Our testing algorithm is run as a standalone application on a Xilinx Embedded Develop Kit (EDK) and we are able to issue multiple PCM commands to the PCM chips.

B. PCM daughter board

We have four PCM chips mounted on a small PCM daughter board that share common address and data buses. There are 8M addresses within a PCM chip and every address has 16 bits, therefore, each PCM has 128 Mbits.

C. Testing Algorithm

Based on the error types we proposed in Section III, we develop a March algorithm that can classify PCM faults. Although there are several proposed diagnostic March tests in the literature [15], [16], we just use basic March algorithm for this preliminary testing and leave these advanced testing approaches as our future work. In our tests, we choose 32 adjacent addresses as a block and each address consists of 16 bits, giving 512 bits per block.

March test algorithm used in this work is shown in Table I. This March test consists of several March elements. Every element applies a set of read/write operations to a given memory address. Our March algorithm runs across consecutive addresses along the direction of the symbol ↘. For example, W0, R0, R0 is a set of operations which are applied to an

TABLE I
PCM ERROR DETECTION MARCH ALGORITHM

b

addr	March 0	March 1	March 2
0	W0,R0,R0	R0	R0
1	W0,R0,R0	R0	R0
⋮	↘	↘	↘
31	W0,R0,R0	R0	R0

addr	March 3	March 4	March 5
0	W1,R1,R1	R1	R1
1	W1,R1,R1	R1	R1
⋮	↘	↘	↘
31	W1,R1,R1	R1	R1

address in memory. First, it writes 0s into the given address. The second and third operations read data from the address and the expected values are 0s. W1 and R1 in the algorithm means writing and reading 1s in a certain address, respectively.

We explain how this March algorithm detects all of the four types of errors as follows:

- WE(I): Write Error (Type I) is detected in March 3. Two read operations obtain two values which are expected to be 1s. However, if a WE(I) occurred at any bit, the returned values from the read operations should be 0s. We can do a Bit-OR operation between two values to detect WE(I). The Bit-OR excludes the RE error.
- WE(II): Similar to WE(I), the Write Error (Type II) can be detected in March 0. Two values from reading are expected to be 0s. We do a Bit-AND operation between them to detect WE(II). The Bit-AND excludes the RE error.
- PI: All PI faults are detected by March 0 and March 3. If a Program Interference error occurs, the data read from March 2 should be different from both of the data read from March 0. March 5 works in the same way. The reason to add March 1 and March 4 is to distinguish RE errors from PI errors. We claim that a PI error occurs only if March 1 and March 2 read the same data from a certain location. March 4 and March 5 works in the same way as March 1 and March 2.
- RE: In March 0 and March 3, a RE occurs if two reads from the same address are different from each other.

We need to point out some exceptions in our algorithm. First, in RE detection, if the first read operation flips some bits in the address, we will never know that a RE happened because the succeeding read data should be the same. However, we did some experiments to clarify our algorithm. We consistently read data from an address. Read errors happened after thousands of read cycles while most of the addresses never had read errors. This experiment was implemented in different blocks and with over 100 million programming cycles. In addition, only a small portion of the read operations can flip the bit in the cell and most of the REs only occurred once. In other words, it did not alter the content in the cell although it exhibited a wrong bit. This behavior suggests that the error was probably a transient problem in the interface circuitry and not in the PCM device itself. We also note that read errors are very rare compared to other errors in PCM devices, as shown in Figure 1. We ignore this error in our experiments. Second, an unexpected read error could mislead us to obtain a WE(I) or a WE(II) in March 3 and March 0. It could also hide a PI error in March 2 and March 5. Still, read error is really uncommon in PCM, so we ignore these errors as well.

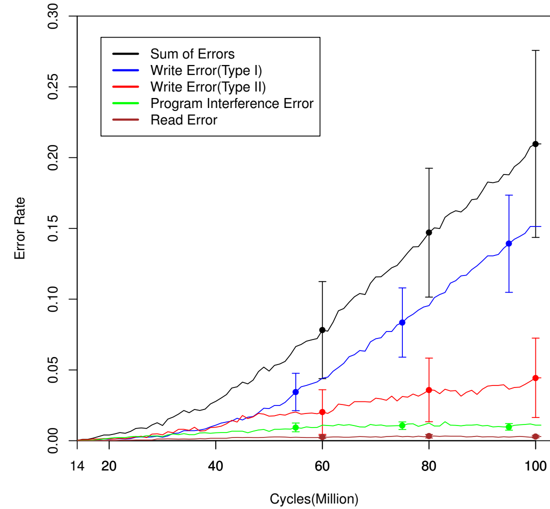


Fig. 1. Overall Error Comparison

TABLE II
MAXIMUM STANDARD DEVIATION

Error Type	max σ_N
Sum	0.068467
WE(I)	0.038458
WE(II)	0.029767
PI	0.005616
RE	0.002051

V. EXPERIMENTAL RESULTS

The lifetime characteristics of the PCM devices and their different error behaviors are presented in this section. These results are obtained from testing eight blocks from two different PCM chips, which four blocks from each chip.

A. Overall PCM Error Comparison

All of the results for the error rates are shown in Figure 1. The x-axis represents how many cycles have already been programmed in a PCM block. The y-axis shows the error rate which is the ratio between the number of bit errors and the total number of bits in the block. The solid lines represent averages of different errors and the vertical bars stand for error ranges of plus-or-minus one standard deviation. We discuss several observations here.

First, all the errors happened after 14 million programming cycles. This means that the PCM cells start to wear out after 14 million programming cycles, which is consistent with the results shown in the literature for PCM lifetime of 10^7 write cycles [3], [2].

Second, WE(I) errors are linearly accumulated and the frequency of this error is much larger than the other three

errors. As shown in Figure 1, WE(I) becomes dominant failure after 50 million programming cycles. The gap between WE(I) and the other three errors becomes larger when the block continues to be extensively programmed. On the other hand, WE(II), PI and RE also tend to accumulate as the number of programming cycles increases. However, from 60 million cycles, PI and RE level out while WE(II) continues to increase, although the slope reduces somewhat. In fact, one can see from the later section that WE(II) eventually becomes constant, too. Since WE(II), PI and RE are not impacted by cycling, we can use the WE(I) rate as a rough lifetime indicator. Furthermore, it is possible to propose a new over-provisioning algorithm for solid-state storage (SSS) based on this behavior. Current storage media with a limited life cycle, such as flash memory, typically include spare memory blocks to compensate for blocks that will fail. The slope of WE(I) in Figure 1 can be used to predict the number of damaged data blocks. Together with the number of spare blocks, this prediction can be used to estimate the number of cycles left on the device. Conversely, the system designer can use this information to determine the number of spare blocks to guarantee a specified lifetime.

Third, Table II shows the standard deviations of the error rates for different types of errors. We choose the maximum value within 100 million programming cycles for each error. From Table II as well as Figure 1, the standard deviations are not large for PI and RE. This indicates that these two types of errors do not vary too much across different blocks or different chips. We will explain later that these two errors are intrinsic properties of PCM chips. On the other hand, WE(I) and WE(II) errors seem very undeterministic in our test results. The reason is because these two write errors are related to the quality of the chips. In later section, we describe how WE(II) has very different behaviors in different chips while it is mostly independent of over-cycling compared to WE(I). Thus, WE(II) can be used as a proxy for the quality of different chips.

B. Error Distribution

In this section, we will discuss the distributions of different errors. The purpose is to discover the relationships between different errors and the number of programming cycles. The experiment runs in several steps: First, we flip all the cells within the block between 0 and 1 and stop flipping after 100 million programming cycles. Second, we run the algorithm shown in Table II 5000 times and print out error messages. Third, we calculate error durations as follows. Once an error occurs at some cell, we initialize a counter for this error and increment it by one if this type of error happens again during next programming cycle. If it does not occur, we stop accumulating and print out the current counter as the error duration. If an error last for 5000 cycles, we consider it as a permanent error which occurs at the first iteration and lasts until the end of 5000 cycles. If an error only lasts for 1 cycle, we consider it a transient error which occurs once and quickly disappears in the next cycle. Data between these two extremes means that the error occurs and lasts for several cycles and then disappears.

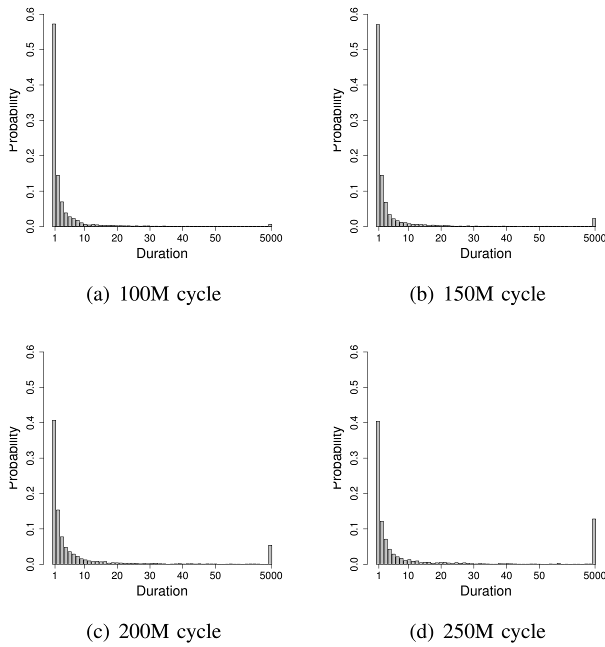
1) Write Error (Type I): Figure 2 shows the WE(I) distribution in a certain block. Figure 2(a) is the distribution after programming 100 million cycles on this block. Figures 2(b), (c), and (d) are the distributions after programming 150 million, 200 million and 250 million cycles, respectively.

Referring to Figure 2, we note that more than half of the errors only last for one cycle and that the distributions from 2 to 10 cycles do not change too much with the number of programming cycles. The error durations from the duration from 50 to 4999 cycles are not shown due to limited graphical resolution. However, the distributions in this range are very low and dispersed across the range. Finally, the distribution of error duration at 5000 cycles and beyond is increasing as the block is being continuously programmed.

As described in Section III, WE(I) occurs due to over-cycling the cell which causes the GST material to separate from the top electrode in the cell. Hence, the cell becomes an open circuit and exhibits high resistance all the time. From our results, we see that wearing out a cell is a gradual process. Figure 2 shows that the probability of one cycle is decreasing as the number of programming cycles increase. Cells appear to be reliable when the chip is young. Even as WE(I) errors begin to occur, the cell tends to recover quickly. However, as the cell is continuously programmed, the connection between the GST and the top electrode apparently becomes looser and the error lasts longer. In Figure 2, the distributions in the range between 2 and 20 cycles are still high and continuous. Finally, the cell reaches a threshold and the error becomes permanent.The probabilities of the duration between 20 and 4999 cycles are very low and the distribution is unexpectedly dispersed within this range. This indicates that the error in the cell is highly likely to become a permanent error if the WE(I) error has already lasted for tens of cycles. This implies the connection between the GST and the electrode is almost broken.

Previously, we suggested using WE(I) as the lifetime indicator of PCM for a new over-provisioning algorithm. In reality, however, it is difficult to obtain lifetime behavior of WE(I) like Figure 1. It requires tracking WE(I) errors at different programming cycles and then predicting the WE(I) slope to estimate the remaining lifetime. Unlike WE(I) lifetime behavior, however, the WE(I) distribution only needs to be examined once to predict the lifetime status. If the distribution is biased towards the permanent error side, we should mark this block as no longer usable. In addition, WE(I) distributions on different blocks are likely to follow the same transition moving from the transient error side to the permanent error side. Therefore, we can record WE(I) distributions from different blocks to develop a global wear-leveling algorithm that balances the lifetime of different blocks. Furthermore, if PCMs are implemented in solid-state storage (SSS), we can also obtain distribution statistics from different blocks and chips and develop a corresponding global wear-leveling algorithm.

2) Write Error(Type II): As shown in Figure 1, the number of WE(II) errors does not dramatically increase with the number of programming cycles. In order to obtain further knowledge of WE(II) behavior, we double the programming



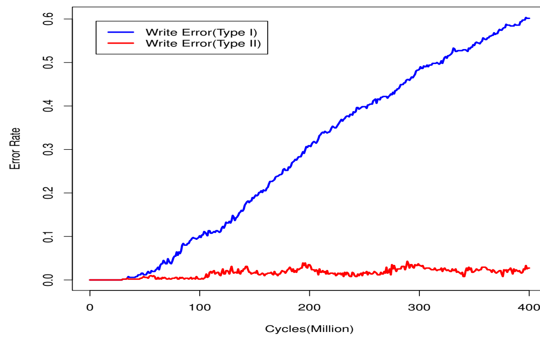
(a) 100M cycle

(b) 150M cycle

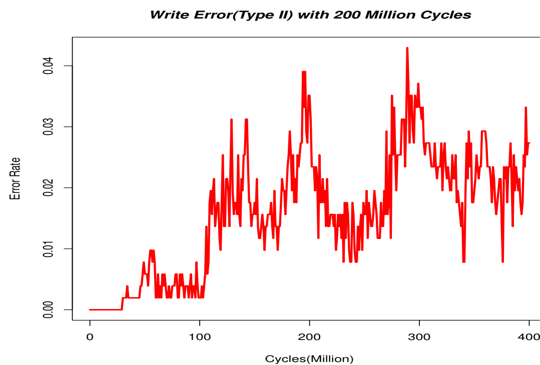
(c) 200M cycle

(d) 250M cycle

Fig. 2. WE(I) Distribution



(a) Two Write Errors



(b) WE(II)

Fig. 3. Write Errors with 200 Million Cycles

cycles to 200 million. Figure 3(a) shows the behaviors of WE(I) and WE(II) with this increased number of write cycles. We also expand WE(II) on a larger scale as shown in Figure 3(b). In Figure 3(a), WE(I) keeps linearly increasing over the 200 million cycles, while WE(II) stays under 0.05.

As we mentioned previously, there are two potential causes of WE(II) errors. The first is due to over-cycling. The chalcogenide material in the cell could intermix with adjacent cells [7] forcing the cell to logic '1'. The second cause can be attributed to impurities in the cell [6]. This is a manufacturing issue not related to over-cycling. Therefore we will separately discuss how these two causes contribute to the observed WE(II) behavior.

From Figure 3(b), WE(II) first occurs at 15 million cycles and slowly increases with the number of programming cycles. This implies over-cycling has impact on WE(II) errors. However, unlike WE(I) errors, WE(II) errors do not linearly increase. After 15 million cycles in Figure 3(b), WE(II) stays below 0.01 until 100 million cycles. At that point, it jumps beyond 0.02 and oscillates between 0.01 and 0.04. This demonstrates that over-cycling could break the physical structure of the cells and force them to stick at '1' all the time. However, compared to WE(I), the lifetime-related WE(II) errors are very rare in PCM. Because most of the cells are completely isolated from adjacent cells, over-cycling these cells will lead to an open circuit which then get categorized as WE(I) errors. Only a small portion of cells would intermix with adjacent cells because of bad isolation when they are extensively programmed. Therefore over-cycling is not the dominant issue for WE(II).

Impurities, which are intrinsic in PCM devices, are a byproduct of the manufacturing process. We found that different PCM chips have different WE(II) behaviors. We selected two chips and picked up four blocks from each chip. We calculated the average of the WE(II) errors for these two chips. From Figure 4, the WE(II) errors in chip1 increase much faster than those in chip2. This suggests different quality levels of the two chips. Since over-cycling does not influence WE(II) very much, we can use WE(II) as a measure of the fabrication quality of PCM chips.

3) PI and RE: Figures 5 and 6 show that the PI and RE errors maintain the same frequency distribution as the number of programming cycles increases. This result indicates that PI and RE are transient errors that are able to heal themselves in the next few cycles.

Recall from section III that PI errors are caused by thermal cross-talk between adjacent cells. Since the PI distribution is independent of the number of programming cycles, it is apparently an intrinsic feature of our PCMs. We speculate that it is the layout of the PCM chip that determines the PI distribution since the layout determines the adjacency between different cells.

Similarly, RE errors are also caused by an intrinsic feature of PCMs. In Section III, we explained that RE errors are caused by defects in the cell or mis-activating the write circuit during a read operation. Both of causes are physical properties of the PCM devices.

Because most PI and RE errors only occur once and then disappear, we can ignore these errors when developing error correcting codes for PCM memory systems. On the other hand, we should put more efforts towards tolerating WE(I) errors

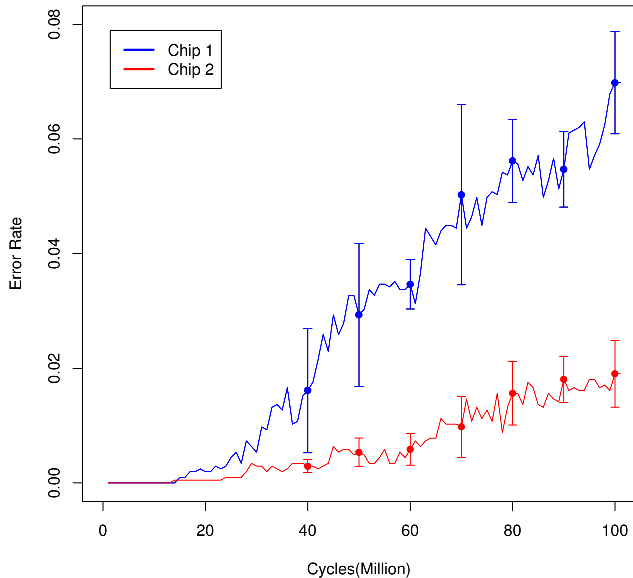
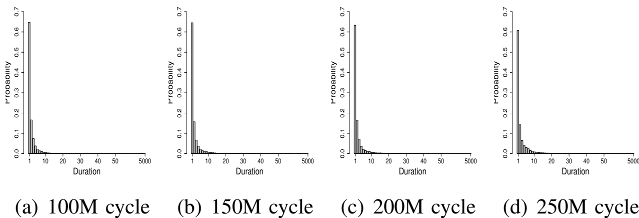
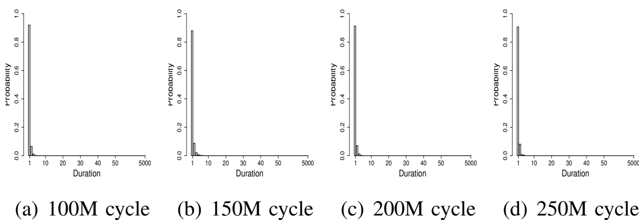


Fig. 4. Chip Comparison for WE(II)



(a) 100M cycle (b) 150M cycle (c) 200M cycle (d) 250M cycle

Fig. 5. PI Distribution



(a) 100M cycle (b) 150M cycle (c) 200M cycle (d) 250M cycle

Fig. 6. RE Distribution

since WE(I) is the dominant error when the PCM devices are extensively programmed.

VI. Conclusion

In this paper, we have developed a simple fault model to characterize PCM errors into four types. Based on our proposed fault model and testing algorithm, we have conducted extensive experiments on real PCM devices at the memory module level. Our measured results have shown many interesting observations in terms of the lifetime of PCM devices and their error distributions. We found that the lifetime of PCM is much longer than that of flash memory and Write Error (Type I) is the dominant type of errors that linearly increases with the number of program cycles. In addition, we have also investigated error distributions for different error types. Our results demonstrated that over-cycling has a significant impact for Write Error (Type I) and negligible effects for the other three error types. In the future, we will use these findings to develop new wear-leveling and ECC algorithms for phase change memory.

Acknowledgment

This material is based upon work supported by the US National Science Foundation (NSF) under Grant ITR-0937060 to the Computing Research Association for the CIFellows Project. This work is also sponsored in part by the Center for Research in Intelligent Storage (CRIS), which is supported by National Science Foundation grant no. IIP-0934396, IIP-1127829, and member companies. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF. The authors also would like to thank Erich Frahm for helping us to build a PCM daughter board.

References

[1] Moinuddin K. Qureshi, Michele Franceschini, and Luis Lastras: "Improving Read Performance of Phase Change Memories via Write Cancellation and Write Pausing", HPCA 2010.

[2] M.K. Qureshi et al.: Morphable Memory System: a Robust Architecture for Exploiting Multi-level Phase ChangeMemories, in ISCA 2010.

[3] S. Raoux et al.: Phase-Change Random Access Memory: a scalable technology, IBM J. Res. Dev., 2008

[4] Magdy S. Abadir and Hassan K. Reghbati.: "Functional Testing of Semiconductor Random Access Memories", ACM Comput. Surv. 15, 3 (September 1983), 175-198.

[5] Yu Cai, Eric F. Haratsch, Onur Mutlu and Ken Mai: "Error Patterns in MLC NAND Flash Memory: Measurement, Characterization and Analysis", DATE12/EDAA, 2012.

[6] M.G.Mohammad: "Fault model and test procedure for phase change memory", IET Computers and Digital Techniques, 2011, 5, (4), pp.263-270.

[7] Pirovano, A., Redaelli, A., Pellizzer, F., et al.: "Reliablibity study of phase-change non-volatile memories", IEEE Trans. Device Mater. Reliab., 2004, 4, (3), pp. 422-427.

[8] Salamon, D., Cockbum, B.F.: "An electrical simulation model for the chalcogenide phase-change memory cell". Int. Workshop on Memory Technology, Design and Testing, July 2003, pp.86-91.

[9] Muller, G., Nagel, N., Pinnow, C.-U., Rohr, T.: "Emerging non-volatile memory technologies". 29th European Solid-State Circuits Conf., September 2003, pp37-44.

[10] Mantegazza, D., Ielmini, D., Pirovano, A., Lacaita, A.: "Anomalous cells with low reset resistance in phase-change memory arrays", IEEE Electron Device Lett., 2007, 28, (10), pp. 865-867.

[11] Lai, S.: "Current status of the phase change memory and its future". IEEE Int. Electron Devices Meeting Technical Digest, December 2003, pp. 10.1.1-10.1.4.

[12] Mantegazza, D., Ielmini, D., Pirovano, A., et al.: "Electrical characterization of anomalous cells in phase change memory arrays". Int. Electron Devices Meeting, December 2006, pp. 1-4.

[13] Ielmini, D., Lacaita, A., Mantegazza, D.: "Recovery and drift dynamics of resistance and threshold voltages in phase-change memories", IEEE Trans. Electron Devices, 2007, 54, (2), pp. 308-315.

[14] Maimon, J.D., Hunt, K.K., Burcin, L., Rodgers, J.: "Chalcogenide memory array: characterization and radiation effects", IEEE Trans. Nucl. Sci., 2003, 50, (6), pp. 1878-1884.

[15] Al-Harbi, S.M., Noor, F., Al-Turjman, F.M.: "March DSS: A New Diagnostic March Test for All Memory Simple Static Faults", IEEE Trans. on CAD of Integrated Circuits and Systems(2007)1713-1720.

[16] Gurgen Harutyunyan, Samvel K. Shoukourian, Valery A. Vardanian, Yervant Zorian: "A New Method for March Test Algorithm Generation and Its Application for Fault Detection in RAMs", IEEE Trans. on CAD of Integrated Circuits and Systems 31(6): 941-949 (2012).