ISSN 2829-4998

Implementation of a new coding scheme for improving the SET operations in Phase Change Memory (PCM)

Milad Mohseni [a,1,*]

[a] Department of Computer Science and Engineering, Islamic Azad University, Tabriz Branch, Tabriz, Iran (Islamic Republic of)
[1] stu.m.mohseni@iaut.ac.ir ;
* corresponding author

ARTICLE INFO

Article history
Received May 3, 2023
Revised June 15, 2023
Accepted July 6, 2023
Available online August 30, 2023

Keywords
PCM
RESET
SET
Write latency
Energy consumption
Non-Volatile Memory

ABSTRACT

Among Non-Volatile Memories (NVMs), PCMs are considered the best alternative to DRAM (dynamic random-access memories). As a result of its superior performance and scalability, there are several advantages over DRAM, including lower leakage and energy consumption, higher cell number, and smaller cells. This kind of memory does, however, suffer from a long write latency. In this article, we present a technique to reduce write latency by reducing the number of SET operations. The proposed method is an improved Write Time Speed-up (WTS) code scheme. In the proposed scheme, a new code based on hamming weight is given, and an appropriate algorithm is written to reduce the number of SET operations. Compared with current methods, the proposed scheme decreased SET and RESET operations by 3.9 percent, SET operations by 3.3 percent, and power consumption by 2.6 percent. Visual Basic 6 and GEM 5 simulations are used to simulate the suggested method.

1. Introduction

Nowadays, a larger main memory is needed because the number of cores is increasing, and complex applications are being developed. Due to its limitations in power and scalability, DRAM has some severe and notable drawbacks. Because of this, it cannot keep up with the growing demand for main memory with a large capacity for saving data. Thus, DRAM will unlikely to become a reliable and promising main memory technology in the near future [1]–[7]. Accordingly, this prompted various emerging memory technologies, including PCM, to be proposed for primary memory use. Many recent research studies have been conducted to determine whether PCM can be used as a primary memory [8], [9]. PCM has been universally reported as an attractive alternative to DRAM due to its scalability, density, and low standby power requirements [10]. PCM devices are made of chalcogenide materials (GST), which modifies the amount of GST resistance with different programming pulses [11]. This type of memory relies on reversible transitions from crystalline (low resistance) to amorphous (high resistance). GST atoms are typically arranged near each other in crystals and have low electrical resistance.In contrast, GST atoms in an amorphous state are irregular, resulting in high resistance [8], [11]. A cell's electrical resistance is determined by the volume rate of an amorphous area. Therefore, at low resistance, amorphous areas have lower mass; at high resistance, they have higher mass. PCM devices with crystalline and amorphous regions are illustrated in Fig. 1. Amorphous regions are formed when a smaller zone of an electrode is surrounded by a more intense electric field and current density [8]. Therefore, PCM chips' electrical resistance indicates how much data they hold; Fig. 1 illustrates how they are built. Every memory cell must be available by linking devices. PCM row lines are enabled or disabled by an access device such as a diode or a transistor. In the following step, after allowing the access device, the column memory can be accessed for writing or reading [8], [11].

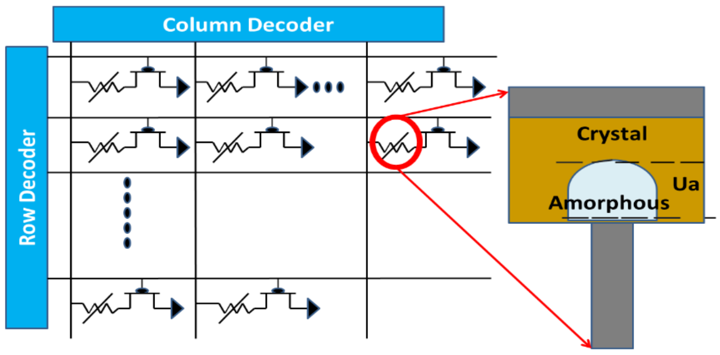

Fig. 1. Structure of Crystal and amorphous

A sandwich of two electrodes is shown in Fig. 2. PCM uses memory lines as the unit for the read/write operations. DRAM and PCM are similar in that each ban k has its own memory lines that can be used independently by the processor. Moreover, multiple banks can be read and written at the same time [12]–[14].

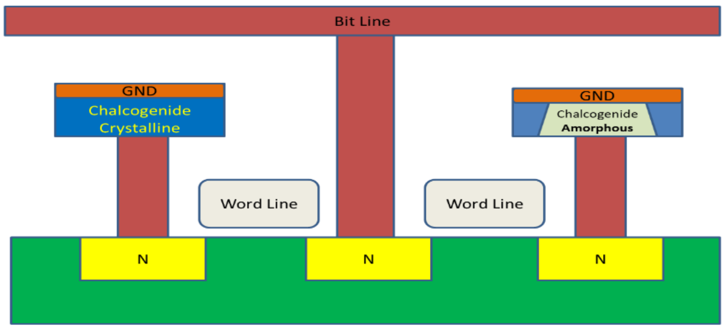

Fig. 2. a PCM cell

Memory cells and phase change materials play a major role in the performance of phase change memories. A PCM cell is specifically designed to improve power efficiency and reduce reset current while simultaneously speeding up processing. A PCM cell's size is limited by its access device. Since they require a relatively high current to switch, Bipolar Junction Transistors (BJT), diodes, and vertical transistors can be essential in reducing PCM cell size. The result is that PCMs can fail because of changes in mass density, disorganized electrode arrangement, and element splitting [15]. Despite the primary goal of replacing NVM with modern memory such as DRAM and flash, there are some limitations, such as cost [16]. Developing high-speed and durable PCM is a significant problem in replacing present memory [15], [17].

2. Related Work

The purpose of this article is to discuss reducing SET operations in order to decrease write latency, thus lowering power consumption. Various strategies are discussed in this section for reducing PCM write latency and power consumption. Each method is then compared for its benefits and drawbacks. It takes a long time to complete a SET operation when there is a long write latency inside a PCM cell. For example, writing "1" into a cell takes somewhere between 6 and 9 times as long as writing "0". Thus, a SET operation creates significant latency during writing. [18] developed a method for diminishing write latency significantly. Write-Once-Memory (WOM) code can reduce write latency and power consumption by lowering the number of SET operations needed. As a result, it improved the average write time. WOM codes are shown in Table 1.

Table 1. $(2^2)^2/3$ WOM code

Information words	**(00)**	**(01)**	**(10)**	**(11)**
Codewords(1st gen)	000	001	010	100
(2nd gen)	111	110	101	011

As in this article, codewords are placed in the table in the first and second generations to decline SET and RESET operations. Next, it performs SET and RESET by using negative logic. In switching from first- to second-generation codewords, there is no need for SET; instead, RESET is required with a little delay between each write operation. Due to encoder and decoder overload, this WOM code is inappropriate for performing intricate computations in hardware. Fig. 3 shows both the WOM code with inverters and the WOM code inverted ((a) with an inverter and (b) without an inverter).

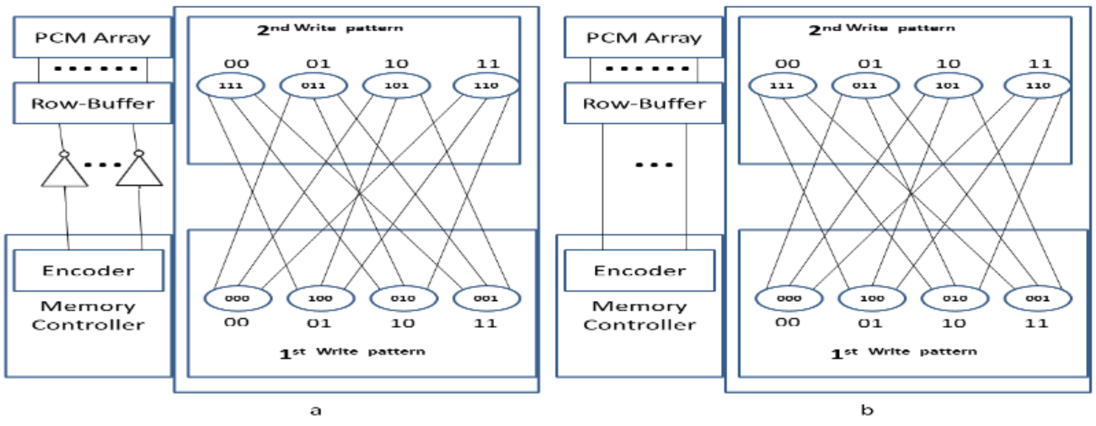

Fig. 3. WOM code with inverters and inverted WOM code

WTS (Write Time Speed-up) is a method similar to the WOM code and the proposed scheme. As a matter of fact, the proposed method optimizes the WTS code. WTS and WOM codes are differentiated by the fact that WTS requires a SET operation to operate non-regularly. Nevertheless, in the WOM code, a SET operation must be performed in every t-the-write operation (where t is the number of codewords corresponding to each information word). This significantly differs between WTS and WOM codes regarding SET operations and codewords. In the WTS code, each information word has multiple codewords. As a result, the codeword would be chosen and encoded so that the procedure for writing operations would require fewer SET operations. In other words, the shortest code word is selected from among its options. Because of this, the WTS code takes less time to write on average. A WTS code example is shown in Table 2.

Table 2. An example of $(2^2)/4$ WTS code

Information words	d_0**=(00)**	d_1**=(01)**	d_2**=(10)**	d_3**=(11)**
Codewords	C_0=(0000)	C_1=(0001)	C_2=(0010)	C_3=(0100)
Codewords	C_4=(1000)	C_5=(0011)	C_6=(0101)	C_7=(0110)
Codewords	C_8=(1001)	C_9=(1010)	C_{10}=(1100)	C_{11}=(0111)
Codewords	C_{12}=(1011)	C_{13}=(1101)	C_{14}=(1110)	C_{15}=(1111)

A practical method is presented by [19], called Data Comparison Write (DCW). When DCW is applied to the writing process, it identifies and filters out unnecessary bits. Using this technique, a memory cell will not be overloaded. Data in a memory cell is compared with new data in DCW. A SET operation is performed when incoming data bits differ from the previous data. Half of the cells in the most desirable statue are written using the mentioned technique. There is a check and update on every cell in the worst statue. Neither a RESET nor SET operation is required when both data have the same bits. On the other hand, it will be necessary to execute the write procedure if both data are different (resulting in both SET and RESET). DCW's SET operations did not appear to be significantly improved over the WTS code scheme.

Flip-N-Write (FNW) is an approach proposed in [20] that is superior to DCW. As a result, redundant writing operations are minimized. Initially, the new data and its inverse are compared, then the hamming distance between them is calculated, and the decision is made whether to write the new data or its inverse. Basically, new data is written after old data has been read out. It is then written to the memory cell when the hamming distance between the new data and its inverse is smaller than the distance between the new and existing data. In all other cases, initial data is written.Additionally, FNW uses a flag bit to recover the original data. FNW produces the best results when the data length is two bits, with an additional flag bit after every two bits. In FNW, simultaneous write operations won't significantly reduce latency. This results in FNW's overhead being much higher than its effectiveness.

According to [21], FNW can be approached in a novel way. SET and RESET operations write the inverse of new data if the total number of ones and zeros is greater than half the total number of bits.

Several strategies for lowering write latency are compared and evaluated in Table 3, including the number of SET operations, the energy consumption in PCM, and the pros and cons of each. However, these approaches overwhelmed and had the write time constraints, even though they diminished SET operations and, consequently, power consumption. In the suggested scheme, we have reduced those limits and the number of SET operations, which should extend PCM lifespans.

Table 3. Methodologies for decreasing the write latency

Methodology	Brief explanation	Pros	Cons
Write Cancellation and Write Pausing [22], [23].	If a read request arrives within a prearranged period of time, the scheduled write request will be cancelled.	Decrease the number of unneeded write operations by concentrating on their removal process.	No significant reduction in write operations has been achieved.
WOM code [24]	Implementing a negative logic to decrease the latency of lengthy write operations.	Reducing power consumption and writing latency.	Incompatible with systems with complicated and complex calculations due to excessive overload when performing encoder and decoder operations.
WTS code [25]	In every information word, lowering the number of SET operations by multiple codewords results in a decrease in the number of SET operations.	SET operations should only be executed when necessary.	The proposed idea has fewer SET operations than the actual number.
PMMA [26]	Energy-efficient, write-cost-effective, and endurance-enhancing memory architecture.	Reducing expensive write operations, balancing read/write latency, and enhancing durability.	Despite the placement policies in the cache, the write operations do not show significant results.
DCW [27]	When the input data and the existing data differ, read operations are performed before writing operations.	Reducing write power consumption by avoiding excessive overhead.	In comparison to current schemes, SET operations occur more frequently.
Flip-N-Write [28]	The simple microarchitectural technique can enhance memory, energy, and endurance.	SET operations are diminishing compared to DCW operations.	Inefficient in its operations because of excessive bits.
D-XOR (Double XOR Mapping) [29]	With a modified bit distribution method, data bits are distributed equally across cell groups.	Analyzing data in smaller segments and finding a balance between them.	Keeping write operations to a minimum.
CAFO [30]	Cost-aware encoding allows the data to be encoded into structures that produce less cost.	A reduction in the cost of writing.	As compared to current methods, it has excessive overhead
DIN	Omitting bit combinations in encoding	The resistance to the write disturbances in high density has decreased.	Comparatively more overhead than current encoding techniques.
Three-stage-write scheme	Makes possible to reduce the delay and change of bits during the writing process by using this method.	Write operations consume less energy and change fewer bits.	Incapable of executing complex instructions due to power and energy limitations.
Clock-Def.	Memory management technique that enhances cell memory performance by optimizing slow write speeds.	Managing memory writes more efficiently.	High-end servers are unable to handle large workloads.

3. Results and Discussion

This article proposes a scheme similar to that offered in [31]. The structure of both is similar. Nevertheless, the table and algorithm in the proposed method differ from those in [31]. There have been some improvements to the table in this scheme, and these improvements have led to fewer SET operations and, subsequently, less power consumption. Also, the algorithm is based on the new structure and order of the codewords.

3.1. The Improved Write Time Speed-up (WTS) code scheme

A table is created using the suggested technique. A hammering weight is used to create the table. The suggested scheme is based on code class and consists of a one-to-multiple function. The table contains several information terms, each of which has multiple codewords.

Assume c is binary code (n is the length of the codeword and k is the information word). Furthermore $C_0, C_1 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots C_{(2^n-1)}$ is an n-bit codeword that is $H(c_i) \preccurlyeq H(c_j)$ for per I and j. $I < j$ whenever H(x) is the hamming length in a vector x. And we assume that $d_0, d_1 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots d_{(2^k-1)}$ are the k-bit information words. Because of these assumptions, a WTS code is defined [31].

Definition: A code in C is called a $(2^k)/n$ WTS code if each information word d_i corresponds to $2^{(n-k)}$ codewords $C(2^k_{j+i})$ $(0 \preccurlyeq j < 2^{(n-k)})$.

Table 4 demonstrates the improved WTS table that decreases the number of SET operations for $(2^2)/4$ k=2 is the information word and n=4 is the codeword. Each k corresponds to $2^{(n-k)} = 4$. For example, d_1 corresponds to C_1, C_5, C_9 and C_{13}. The hamming weight is denoted in the table by the letter I. For C_0 the hamming weight I is zero, one for $C_1 - C_4$,two for $C_5 - C_9$, three for $C_{10} - C_{14}$ and four for c 15. Therefore, $(0 \preccurlyeq j < 2^{(n-k)})$ is true for I and j such that I < j. Furthermore, an information word (d_i) stores a codeword (c') in a memory cell. There are $2^{(n-k)}$ codewords $C(2^k_{j+i})$ $(0 \preccurlyeq j < 2^{(n-k)})$ corresponding to d_i . Thus, one of two $2^{(n-k)}$ codewords is chosen as the encoded word, sent to the designed encoder. Table 4 shows the improved WTS table for $(2^2)/4$.

Table 4. The improved WTS table for $(2^2)/4$

Information words	d_0=(00)	d_1=(01)	d_2=(10)	d_3=(11)
Codewords	C_0=(0000)	C_1=(0010)	C_2=(0100)	C_3=(0001)
Codewords	C_4=(1000)	C_5=(0011)	C_6=(1001)	C_7=(0110)
Codewords	C_8=(0101)	C_9=(1100)	C_{10}=(1010)	C_{11}=(0111)
Codewords	C_{12}=(1101)	C_{13}=(1110)	C_{14}=(1011)	C_{15}=(1111)

The suggested algorithm is carried out in five stages:

- The first step is picking an information word, then selecting the shortest codeword among its codewords and encoding it as a chosen codeword.
- The previously chosen codeword is compared to the codewords of the following information word in the second stage. A SET operation is executed when each bit of the selected codeword changes from one to zero in comparison to the following codeword. Thus, when comparing the selected codeword to the next codeword, any codeword that causes the SET operation is ignored, and the next codeword is checked out. Moreover, when the selected codeword is compared to the next codeword, if there are two or three codewords that do not result in a SET operation, the shortest codeword among those codewords is picked and encoded as a selected codeword.
- As stated in the previous step, a SET operation must be executed when each bit of the chosen codeword changes from one to zero. Therefore, shifting the bits from zero to zero, one to one, and zero to one generates a RESET operation that does not significantly increase the write latency in PCMs.

- Assume that all the codewords allow the SET operation to run (no codeword stops the SET operation from executing). In such a scenario, the shortest of those codewords is chosen and encoded.
- The process would continue until the final information word.

Because the technique involves fewer SET operations in the subsequent stages, the shortest codeword is chosen and encoded. Fig. 4 shows the flow chart of proposed scheme.

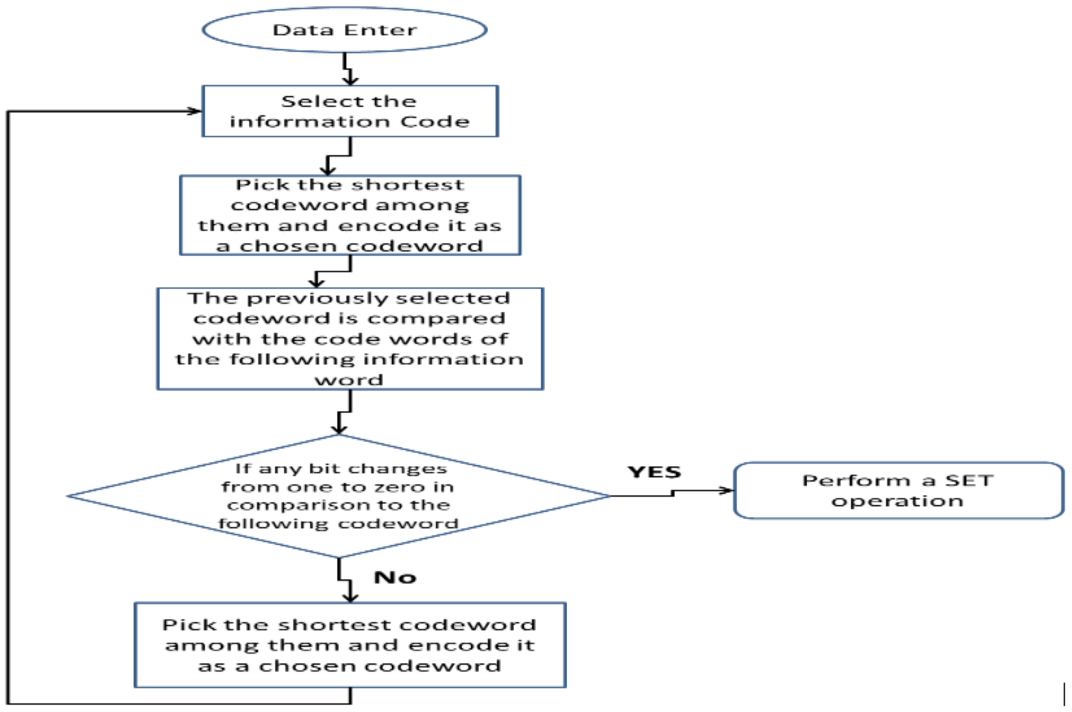
Data Enter
Select the information Code
Pick the shortest codeword among them and encode it as a chosen codeword
The previously selected codeword is compared with the code words of the following information word
If any bit changes from one to zero in comparison to the following codeword
YES
Perform a SET operation
No
Pick the shortest codeword among them and encode it as a chosen codeword

Fig. 4. the flow chart of proposed scheme

Initially, all cells are set to zero and the values d_0= (00), d_1= (01), d_3= (11), and d_2= (10) are written into them. The codewords for d_0= (00) are c_0 = (0000), c_4 = (1000), c_8 = (0101), and c_{12} = (1101). Because all codewords are larger than c_0 = (0000), so c_0 is picked and encoded. The chosen codeword is then compared against codewords in the next information word in the next phase. The codewords for d_1= (01) are c_1 = (0010), c_5 = (0011), c_9 = (1100), and c_{13} = (1110). When the chosen code (c') is compared to the codeword corresponding to d_1= (01), the SET operation is not performed by choosing each of the codewords. As a result, any of the codewords can be chosen as the selected and encoded one; however, to avoid increasing the number of SET operations in subsequent stages, the shortest codeword among those codewords is picked; therefore, c_1 = (0010) is chosen. Following that, the codewords that correspond to d_3= (11) are c_3 = (0001), c_7 = (0110), c_{11} = (0111), and c_{15} = (1111). If c_3 = (0001) is chosen, a SET operation is performed ((the third valuable bit of the selected codeword (c_1 = (0010) is one), and if c_3 = (0001) is chosen as encoded codeword, the bit changes to zero, resulting in a SET action, therefore c_3 = (0001) is skipped and the remaining codewords are checked. If any of the remaining codewords (c_7 = (0110), c_{11} = (0111), and c_{15} = (1111)) is selected as an encoded codeword, no SET operation is done; as a result, the shortest codeword (c_7 = (0110)) is chosen and encoded. In the last step, the codewords that correspond to d_2= (10) are c_2 = (0100), c_6 = (1001), c_{10} = (1010), and c_{14} = (1011). When the picked codeword (c_7 = (0110)) is compared to the existing codewords, it is apparent that if each of the codewords is chosen, a SET operation is performed (the second valuable bit of the (c_7 = (0110) changes to zero). So, the shortest codeword among those codewords c_2 = (0100) is chosen as the selected and encoded codeword. As a result, a SET operation is unavoidably performed at this phase. Therefore, in this procedure, only the step d3 →d2 requires a SET operation to be executed, while the other stages do not. Table 5 demonstrates an example of the proposed scheme's encoding procedure for the $(2^2)/4$.

Table 5. An instance of encoding for $(2^2)/4$

Information words	d_0=(00)	d_1=(01)	d_2=(10)	d_3=(11)
Codewords	C_0=(0000)	C_1=(0010)	C_2=(0100)	C_3=(0001)
Codewords	C_4=(1000)	C_5=(0011)	C_6=(1001)	C_7=(0110)
Codewords	C_8=(0101)	C_9=(1100)	C_{10}=(1010)	C_{11}=(0111)
Codewords	C_{12}=(1101)	C_{13}=(1110)	C_{14}=(1011)	C_{15}=(1111)

3.2. Encoder and decoder for the proposed scheme

Fig. 5 shows the structure of the encoder and decoder's blocks in WTS code, including LUTs [31].

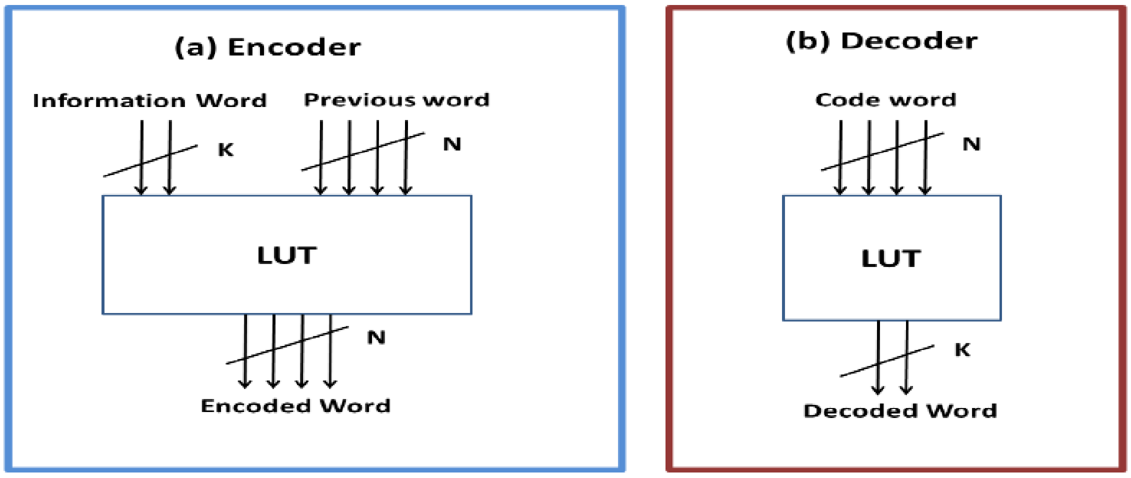

Fig. 5. Encoder and decoder of the WTS code

Decoders and encoders are saved for each entry in the LUT. Each encoded codeword relies on its prior word (c′). As a result, it inserts both the information word and the c into the encoder. A LUT entry is used by the encoder, represented by (k/n). Each decoder executes its actions according to the codewords set in the decoder entrance. The encoder uses a (k/n)-input LUT. Decoders and encoders with small N and K require only a few transistors. In addition to being constructed from hamming weight, the proposed method has fewer SET operations than the WTS code. A suggested algorithm is also developed based on the table in this scheme.

4. The proposed coding scheme

This section summarizes the findings from the evaluation of the proposed plan. We evaluated our proposal by using the gem 5 simulator and visual basic 6. Two separate simulators software are used to simulate the proposed method because gem 5 is a professional PCM simulator that supports 12 distinct workloads. Additionally, we use visual basic 6 to confirm that the assessment findings for the suggested system are near to the actual results. The suggested scheme's outputs are evaluated using fundamental PCM parameters, such as the number of SET and RESET operations in a memory cell and the needed power consumption for write operations. We adapted the gem 5 simulators to use a PCM-based main memory model with a DRAM cache and tested the proposed strategy using 12 PARSEC 2.1 workloads.

4.1. Evaluating the results of the proposed scheme in Visual Basic 6

All simulations were conducted using 1,000 instructions, and the operations performed for $(2^2)/4$ in this software. The number of SET and RESET operations in various transactions in the proposed scheme, WTS code, and WOM code are demonstrated in Fig. 6. As seen in the figure, while WOM code has fewer SET and RESET operations, it suffers from significant overhead in the intricate computation. This code is inappropriate for PCM because it always deals with complex calculations. The proposed strategy, on the other hand, can lessen SET operations in different transactions to 3.3 percent and RESET operations to 2.26 percent.

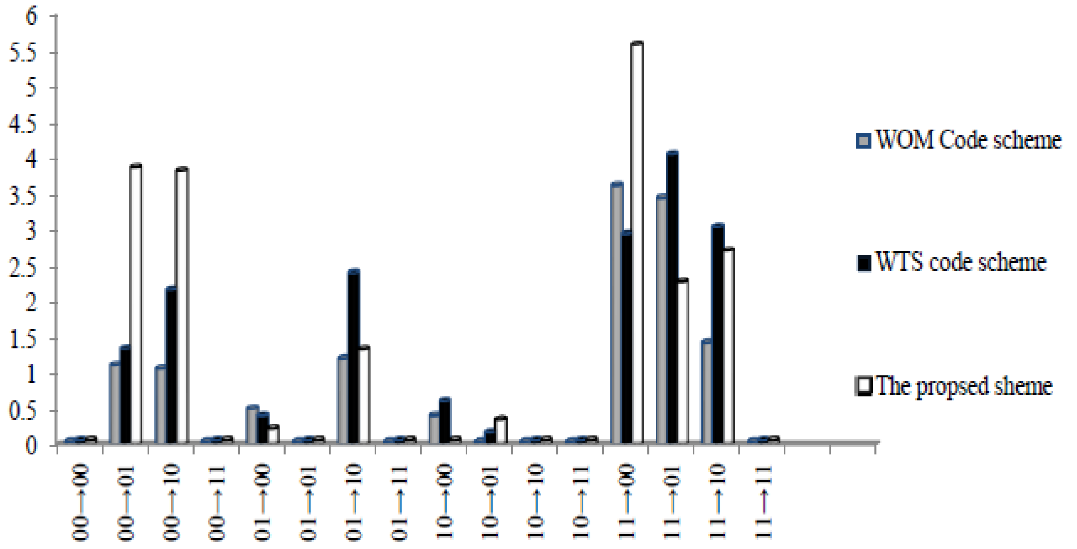

Fig. 6. Number of SET and RESET operations in different transactions for $(2^2)/4$

Fig. 7 demonstrates that the largest numbers of SET operations are executed in the proposed scheme and the WTS code scheme.

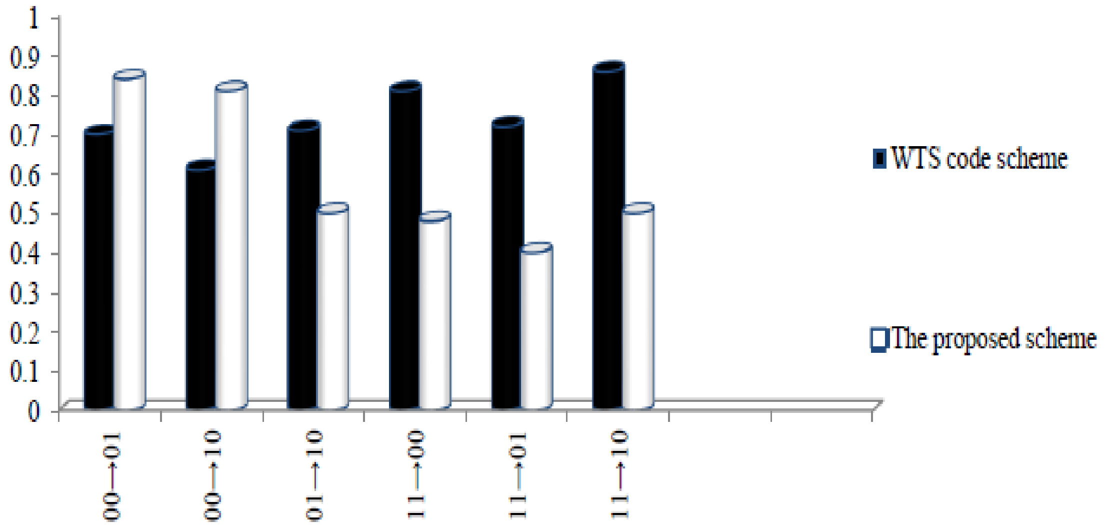

Fig. 7. The largest number of SET operations in different transactions for $(2^2)/4$

When comparing the highest number of SET operations, it is clear in both techniques that transactions 11→00 and 11→10 have the highest number of SET operations among existing transactions. Another significant point in this figure is that the WTS code has more SET operations than other transactions in three transactions 11→00, 11→01, and 11→10. In other words, they comprised 52 % of all SET operations in all the transactions. However, in the proposed scheme, by studying and considering those transactions, not only could we decline the number of SET operations in most of the transactions remarkably, but we also could decrease the number of SET operations in the dominant transactions (11→00, 11→01 and 11→10) in the WTS code scheme to 45%. Fig. 8 illustrates the transactions that contain the fewest SET operations in the proposed scheme and the WTS coding scheme (the transaction that the possibility of executing the SET operation is close to zero in them).

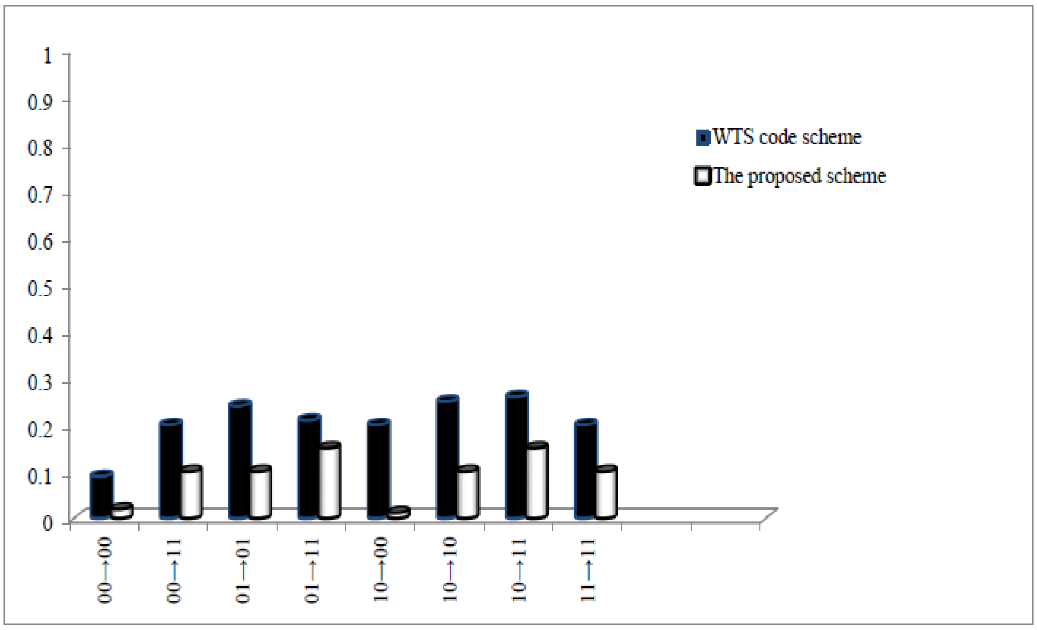

Fig. 8. The smallest number of SET operations in different transactions for $(2^2)/4$

In the proposed scheme, there are eight transactions, whereas, in the WTS code scheme, there are seven transactions that the possibility of executing SET operations is near zero. As a result of the increase in zero transactions, the number of SET operations decreases completely, and in this scenario, the suggested scheme can reduce the number of operations by 7.25 percent as compared to the WTS code scheme.

4.2. Evaluating the results of the proposed scheme in gem 5

All simulations have been executed based on 5 billion instructions, and all operations performed for $(2^2)/4$ in this software. Table 6 contains the detailed baseline parameters. The baseline system has normalized the figures in this section. The number of SET and RESET operations in the WOM code, the WTS code, and the suggested model is shown in Fig. 9 for 12 different workloads. The results show that the proposed technique reduces the number of SET and RESET operations by 3.9 percent compared to the WTS code. Fig. 10 illustrates the number of SET operations necessary in each of the three techniques for various workloads (WOM code, WTS code, and the proposed scheme). According to the graph, the proposed approach can lower the number of SET operations by 3.3 percent when compared to the WTS code.

Table 6. Baseline parameters

System	**4-core CMP، 2 GHz**
Processor core	Single issue in-order، 32KB iL1، 32KB dL1، 4-way، 2-cycle access latency.
L2 cache	2MB، 8-way، LRU، 64B line size، write back، 6-cycle access latency.
Line-State cache	16KB،8-Way، LRU، 64B line size، write back، 6-cycle access latency.
DRAM Cache	32MB ،16-way، 64B cache line size، 50ns (200 cycles) access latency، write back.
Main Memory	LPDDR3-N-1600، 9_8b I/O، 8GB PCM(including 1GB ES)، 4KB page، 1 channel ، 1 rank per channel ، 16 banks per rank، 24 entries read/write queues per bank، read priority scheduling (unless WRQ is >80% full)، with differential write support.
PCM write latency	RESET: 5V، 100_A، 29.7pJ per bit، 50ns operation latency، 125ns iteration latency; SET: 3V، 50_A، 22.5pJ per bit 150ns operation latency، 250ns iteration latency (including Toff [53] & verify)، MLC Write Parameters : 2-bit MLC [5، 14، 21]; MQ: 52.2pJ، 200ns operation latency، 300ns iteration latency [11، 23].
PCM read latency	Tell/Trip=12/2 cycles، traced=200 cycles (250ns) for MLC read / 100 cycles (125ns) for SLC read. 6.3nJ per line.

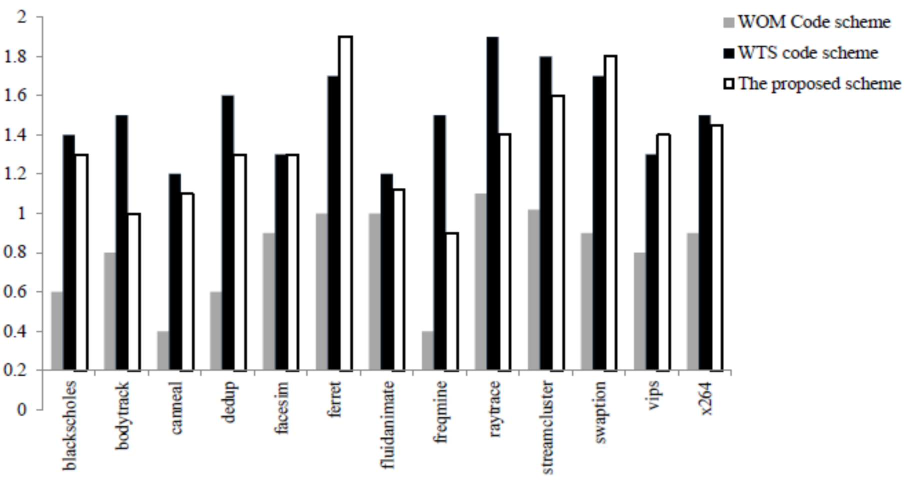

Fig. 9. Number of SET and RESET operations in different workload

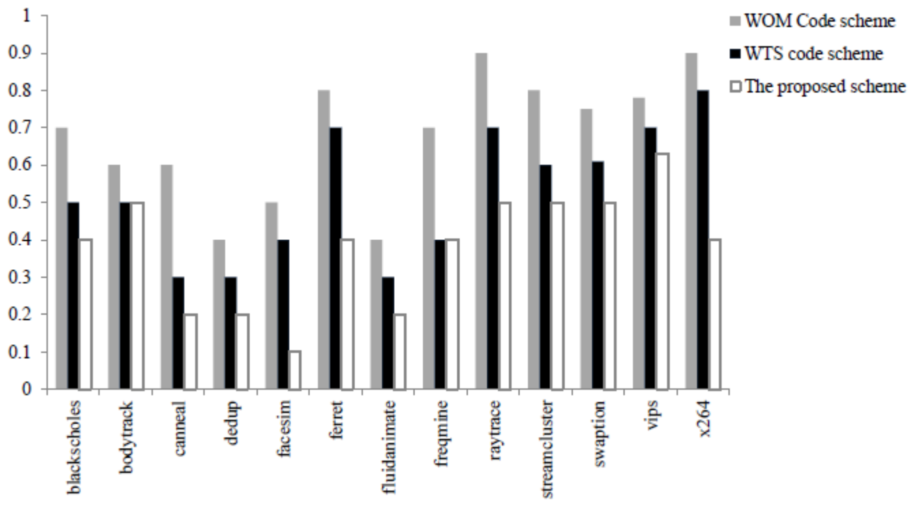

Fig. 10. Number of needful SET operations in different workloads

Equation 1 illustrates the power consumption associated with performing write operations under various workload conditions. We used this equation [19] to determine power consumption.

$$Average\ Power:\ (P_{SET} + P_{RESET})\ /\ 4 \tag{1}$$

P_{RESET}: The amount of power consumption to execute a RESET operation.

P_{SET}: The amount of power consumption to execute a SET operation.

The results indicated that the WOM code consumed less power than the proposed scheme's WTS code scheme due to fewer SET and RESET operations. However, the proposed approach consumed less energy because it reduced the number of SET operations. Overall, it has the potential to reduce power usage by 2.6 percent when compared to the WTS code scheme.

5. Conclusion

NVMs, particularly PCMs, are the leading option to replace DRAM memories due to their multiple advantages over DRAM, including reduced leakage and density, lower standby power consumption, smaller cells, and more scalability. They do, however, have a significant obstacle, which is write latency. In this article, an improved scheme of write time speed-up (WTS) code has been presented.

This scheme relies on the hamming weight in which each information word has several corresponding codewords to decline SET operations. Additionally, an algorithm has been developed to make use of codewords. The proposed scheme was simulated in Gem 5 and Visual Basic 6, and the results indicated that it could reduce the number of SET and RESET operations by 3.9 percent, the SET operation by 3.3 percent, and the power consumption by 2.6 percent when compared to the WTS code.

Declarations

Author contribution. Milad Mohseni introduced and analyzed the coding scheme. Ahmad Habibized Navin simulated the proposed the scheme, and was a major contributor in writing the manuscript. All authors read and approved the final manuscript.

Funding statement. The authors received no financial support for the research, authorship, and/or publication of this article.

Conflict of interest. The authors declare no conflict of interest.

Additional information. No additional information is available for this paper.

References

[1] C. Lefurgy, K. Rajamani, F. Rawson, W. Felter, M. Kistler, and T. W. Keller, “Energy Management for Commercial Servers,” *Computer (Long. Beach. Calif).*, vol. 36, no. 12, pp. 39–48, 2003, doi: 10.1109/MC.2003.1250880.

[2] S. Hong, “Memory technology trend and future challenges,” *2010 International Electron Devices Meeting, San Francisco, CA, USA, 2010, pp. 12.4.1-12.4.4*, doi: 10.1109/IEDM.2010.5703348.

[3] K. Kim, “Technology for sub-50nm DRAM and NAND flash manufacturing,” *Tech. Dig. - Int. Electron Devices Meet. IEDM*, vol. 2005, pp. 323–326, 2005, doi: 10.1109/IEDM.2005.1609340.

[4] B. C. Lee, E. Ipek, O. MutIu, and D. Burger, “Phase change memory architecture and the quest for scalability,” *Commun. ACM*, vol. 53, no. 7, pp. 99–106, Jul. 2010, doi: 10.1145/1785414.1785441.

[5] B. Schroeder, E. Pinheiro, and W. D. Weber, “DRAM errors in the wild: A large-scale field study,” *SIGMETRICS/Performance’09 - Proc. 11th Int. Jt. Conf. Meas. Model. Comput. Syst.*, vol. 37, no. 1, pp. 193–204, 2009, doi: 10.1145/1555349.1555372.

[6] S. Thoziyoor, J. H. Ahn, M. Monchiero, J. B. Brockman, and N. P. Jouppi, “A comprehensive memory modeling tool and its application to the design and analysis of future memory hierarchies,” *Proc. - Int. Symp. Comput. Archit.*, pp. 51–62, 2008, doi: 10.1109/ISCA.2008.16.

[7] W. Zhang and T. Li, “Exploring phase change memory and 3D die-stacking for power/thermal friendly, fast and durable memory architectures,” *Parallel Archit. Compil. Tech. - Conf. Proceedings, PACT*, pp. 101–112, 2009, doi: 10.1109/PACT.2009.30.

[8] M. K. Qureshi, V. Srinivasan, and J. A. Rivers, “Scalable high performance main memory system using phase-change memory technology,” *Proc. - Int. Symp. Comput. Archit.*, pp. 24–33, 2009, doi: 10.1145/1555754.1555760.

[9] M. K. Qureshi, S. Gurumurthi, and B. Rajendran, “Phase Change Memory,” pp. 1-122, 2012, doi: 10.1007/978-3-031-01735-3.

[10] S. Raoux *et al.*, “Phase-change random access memory: A scalable technology,” *IBM J. Res. Dev.*, vol. 52, no. 4–5, pp. 465–479, 2008, doi: 10.1147/RD.524.0465.

[11] S. Rashidi, M. Jalili, and H. Sarbazi-Azad, “Improving MLC PCM Performance through Relaxed Write and Read for Intermediate Resistance Levels,” *ACM Trans. Archit. Code Optim.*, vol. 15, no. 1, p. 12, Mar. 2018, doi: 10.1145/3177965.

[12] L. Jiang, B. Zhao, Y. Zhang, J. Yang, and B. R. Childers, “Improving write operations in MLC phase change memory,” *Proc. - Int. Symp. High-Performance Comput. Archit.*, pp. 201–210, 2012, doi: 10.1109/HPCA.2012.6169027.

[13] Y. Kim, S. Yoo, and S. Lee, “Improving Write Performance by Controlling Target Resistance Distributions in MLC PRAM,” *ACM Trans. Des. Autom. Electron. Syst.*, vol. 21, no. 2, pp. 1-27, Jan. 2016, doi: 10.1145/2820610.

[14] L. Jiang, B. Zhao, J. Yang, and Y. Zhang, "A low power and reliable charge pump design for Phase Change Memories," *Proc. - Int. Symp. Comput. Archit.*, pp. 397–408, 2014, doi: 10.1109/ISCA.2014.6853194.

[15] A. Chen, "A review of emerging non-volatile memory (NVM) technologies and applications," *Solid. State. Electron.*, vol. 125, pp. 25–38, Nov. 2016, doi: 10.1016/J.SSE.2016.07.006.

[16] C. H. Lam, "Phase Change Memory and its intended applications," *Tech. Dig. - Int. Electron Devices Meet. IEDM*, vol. 2015-February, no. February, pp. 29.3.1-29.3.4, Feb. 2015, doi: 10.1109/IEDM.2014.7047133.

[17] H. Y. Cheng *et al.*, "Atomic-level engineering of phase change material for novel fast-switching and high-endurance PCM for storage class memory application," *Tech. Dig. - Int. Electron Devices Meet. IEDM*, pp. 1-30, 2013, doi: 10.1109/IEDM.2013.6724726.

[18] J. Li and K. Mohanram, "Write-once-memory-code phase change memory," *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, Germany*, pp. 1-6, Apr. 2014, doi: 10.7873/DATE.2014.194.

[19] B. Do Yang, J. E. Lee, J. S. Kim, J. Cho, S. Y. Lee, and B. G. Yu, "A low power phase-change random access memory using a data-comparison write scheme," *Proc. - IEEE Int. Symp. Circuits Syst.*, pp. 3014–3017, 2007, doi: 10.1109/ISCAS.2007.377981.

[20] S. Cho and H. Lee, "Flip-N-Write: A simple deterministic technique to improve PRAM write performance, energy and endurance," *2009 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), New York, NY, USA*, pp. 347-357. 2009. Available at: https://ieeexplore.ieee.org/document/5375405.

[21] J. Yue and Y. Zhu, "Accelerating write by exploiting PCM asymmetries," *Proc. - Int. Symp. High-Performance Comput. Archit.*, pp. 282–293, 2013, doi: 10.1109/HPCA.2013.6522326.

[22] M. K. Qureshi, M. M. Franceschini, and L. A. Lastras-Montaño, "Improving read performance of phase change memories via write cancellation and write pausing," *Proc. - Int. Symp. High-Performance Comput. Archit.*, pp. 1-11, 2010, doi: 10.1109/HPCA.2010.5416645.

[23] M. K. Qureshi, M. M. Franceschini, A. Jagmohan, and L. A. Lastras, "PreSET: Improving performance of phase change memories by exploiting asymmetry in write times," *Proc. - Int. Symp. Comput. Archit.*, pp. 380–391, 2012, doi: 10.1109/ISCA.2012.6237033.

[24] A. P. Ferreira, B. Childers, R. Melhem, D. Mossé, and M. Yousif, "Using PCM in next-generation embedded space applications," *Real-Time Technol. Appl. - Proc.*, pp. 153–162, 2010, doi: 10.1109/RTAS.2010.40.

[25] A. P. Ferreira, M. Zhou, S. Bock, B. Childers, R. Melhem, and D. Mossé, "Increasing PCM main memory lifetime," *Proc. -Design, Autom. Test Eur. DATE*, pp. 914–919, 2010, doi: 10.1109/DATE.2010.5456923.

[26] Y. Du, M. Zhou, B. R. Childers, D. Mossé, and R. Melhem, "Bit mapping for balanced PCM cell programming," *Proc. - Int. Symp. Comput. Archit.*, pp. 428–439, 2013, doi: 10.1145/2485922.2485959.

[27] R. Maddah, S. M. Seyedzadeh, and R. Melhem, "CAFO: Cost aware flip optimization for asymmetric memories," *2015 IEEE 21st Int. Symp. High Perform. Comput. Archit. HPCA 2015*, pp. 320–330, Mar. 2015, doi: 10.1109/HPCA.2015.7056043.

[28] Y. Li, X. Li, L. Ju, and Z. Jia, "A three-stage-write scheme with flip-bit for PCM main memory," *20th Asia South Pacific Des. Autom. Conf. ASP-DAC 2015*, pp. 328–333, Mar. 2015, doi: 10.1109/ASPDAC.2015.7059026.

[29] L. Jiang, Y. Zhang, and J. Yang, "Mitigating write disturbance in super-dense phase change memories," *Proc. Int. Conf. Dependable Syst. Networks*, pp. 216–227, Sep. 2014, doi: 10.1109/DSN.2014.32.

[30] S. Lee, H. Bahn, and S. H. Noh, "Characterizing memory write references for efficient management of hybrid PCM and DRAM memory," *IEEE Int. Work. Model. Anal. Simul. Comput. Telecommun. Syst. - Proc.*, pp. 168–175, 2011, doi: 10.1109/MASCOTS.2011.68.

[31] K. Namba and F. Lombardi, “A Coding Scheme for Write Time Improvement of Phase Change Memory (PCM) Systems,” *IEEE Trans. Multi-Scale Comput. Syst.*, vol. 2, no. 4, pp. 291–296, Oct. 2016, doi: 10.1109/TMSCS.2016.2605098.