Phase Change Memory Write Cost Minimization by Data Encoding

Azalia Mirhoseini, *Student Member, IEEE*, Miodrag Potkonjak, *Member, IEEE*, and Farinaz Koushanfar, *Senior Member, IEEE*

***Abstract*—Phase change memory (PCM) is a promising next generation nonvolatile memory. Despite the currently popular charge-based storage techniques, PCM leverages a much more scalable thermal-resistive mechanism that enables sub-10 nm feature sizes. To realize PCM's potential, there are a number of technical challenges that need to be addressed, including limited wear endurance and high energy consumption of bit writes. Our work introduces a novel set of tools and methodologies for encoding data on PCM that optimizes its write performance. We develop a framework which exploits asymmetries in PCM read/write. We show that this coding problem is NP-complete. To provide a tractable solution, we propose two different methods: the first uses integer linear programming, and the second leverages dynamic programming to find an approximation of the optimal solution. Our methods target both single and multi-level cell PCM and can be directly applied to any asymmetric nonvolatile memory with bit-level accessibility. We further optimize our codes by leveraging data distributions. We devise a low-overhead architecture for the encoder module which can be easily integrated within the existing computer memory architecture. We demonstrate the applicability, low overhead, and efficiency of our proposed framework with extensive evaluations.**

***Index Terms*—Encoding, memory management, optimization, phase change memory.**

I. Introduction

In THE design of digital integrated circuits and systems, memory often significantly impacts the system's implementation cost, overhead, and performance. Currently, there is an ever increasing performance gap between emerging (multi) processor families and memory. For portable devices and embedded systems with constrained energy sources, minimizing memory energy dissipation becomes excessively important [9]. Improving and scaling the currently used storage technologies would have a limited effectiveness in the long run, especially as the miniaturized technologies reach the limits of the silicon [21]. The newer resistive memory technologies enable an alternative or a hybrid solution that could bridge the growing performance gap between processing and storage.

The data storage mechanism for resistive memories is built upon the large electrical resistance discrepancy between the states of a *phase change material*. In one phase (state), the material is amorphous and has a very high resistance. In another phase, the same material is crystalline and is highly conductive. Extensive research in the field of phase change material has developed new nonvolatile memory cell structures with improved performance, integration, endurance, retention, and yield properties [20], [29].

This paper aims at minimizing the number of bit transitions required for writing data on phase change memory (PCM) by developing encoding methods. Minimizing the number of bit transitions provides energy and endurance enhancement. Our general optimization can be easily integrated with the processor architecture and memory interface and incurs very low overhead. The method is largely transparent and orthogonal to other energy saving transformations and methods. The proposed resistive memory encoding utilizes bitwise manipulation ability during the word overwrites; only the bits that are changing for the new word compared to the existing word in the memory location would require overwriting. The encoding ensures that the number of required overwrites is minimized. Our optimizations capture asymmetries in energy cost of read, set, and reset operations. We provide a general optimization/coding framework that not only works for PCM but can be directly applied to any bit-accessible asymmetric memory.

A special case of data encoding for minimizing the unidirectional transitions in the memory is the write-once memory (WOM) coding that was originally proposed by Rivest and Shamir [30]. They assumed a memory model where the bits could only be set (and could not be reset). Their the goal was to increase the number of effective cycles for rewriting to the memory. Subsequent interesting work has followed, mostly in information theory and coding, with the goal of estimating the capacity and finding more efficient WOM codes. Applications and extensions of this model for improving flash memory lifetime have been studied [18], [24], [36]. These methods however cannot be directly applied to PCM since PCM has distinctive energy characteristics. Other research works have specifically focused on improving PCM endurance and energy consumption. Reducing the number of cell programming for data updates or flipping data to minimize reprogramming redundant bits are prominent approaches [13], [17], [33], [38]. Our work generalizes the above methods by devising codes for

Manuscript received March 29, 2014; revised August 16, 2014; accepted November 12, 2014. Date of publication February 25, 2015; date of current version March 09, 2015. This paper was recommended by Guest Editor H. Hunter.

A. Mirhoseini and F. Koushanfar are with the Department of Electrical and Computer Engineering, Rice University, Houston, TX 77005 USA (e-mail: azalia@rice.edu; farinaz@rice.edu).

M. Potkonjak is with the Department of Computer Science, University of California, Los Angeles, CA 90095 USA (e-mail: miodrag@cs.ucla.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JETCAS.2015.2398211

minimizing the cost of bi-directional bit transitions for PCM write operations.

The large space of possibilities provided by the ability to perform bit-level operations and the asymmetry in setting and resetting transitions have encouraged us to develop new coding schemes to improve PCM's energy consumption. The complicating factors for this problem are the new degrees of freedom and the curse-of-dimensionality due to the exponential number of plausible code combinations. To address these challenges, we present a novel formal handling of the energy minimization for resistive memory and other storage technologies with bit-level operations and simultaneous consideration of the set and reset transition energy costs. Our contributions are as follows.

- A formal treatment and formulation of PCM coding, with the goal of minimizing the cost of bit transitions required for writes. We show that the problem is NP-complete.
- A methodology for deriving the optimal bounds for codes that yield minimum bit-transition cost.
- An integer linear programming (ILP) formulation that can find the optimal solution to the problem. Our ILP framework can integrate both symmetric and asymmetric set/reset costs for different code sizes.
- A new alternative fast and efficient algorithms for addressing the problem for runtime and efficiency reasons. The method builds upon the smaller optimal codes using dynamic programming (DP).
- An efficient distribution-aware data encoding method for nonuniformly distributed data.
- A novel write cost minimization coding technique for multi-level cell PCM.
- An efficient and highly integrable architecture for the coder module.
- Evaluation of the proposed encoding methods on a diverse set of data including image, audio, and text files.

An earlier version of this work has appeared in [25]. The new aspects of the current manuscript include 1) proof for NP-completeness of our problem, 2) detailed discussion of our ILP method in Section VI-A, 3) new studies of our codings from the perspective of memory durability in Section VII, 4) proposing a new energy-aware coding for multi-level cell PCM in Section VIII, 5) devising a new architecture for our encoding module presented in Section IX, lastly 6) providing additional experiments to evaluate the performance of the new schemes in Section X.

The remainder of this paper is organized as follows. The relevant literature is surveyed in Section II. PCM operation and model is discussed in Section III. An overview of our approach is presented in Section IV. In Section V, we formally define the energy saving data encoding problem and discusses its complexity. Our method for finding the optimum bounds on the codes is also presented. In Section VI, we provide our solutions for the write-efficient coding problem and analyze its complexity. In Section VII, we study how our method affects PCM from the memory wear perspective. We present our novel write-efficient coding for multi-level cell phase change memory in Section VIII. We provide the architecture of our coding algorithm in Section IX. Evaluations of the methods on several benchmark data sets are presented in Section X. We conclude in Section XI.

II. Related Work

Recent advances in resistive memory material and device technology have paved the way for building PCM devices that are comparable or better than conventional solid state memory and DRAM in terms of certain properties. The field has been rapidly growing in recent years both in research and in terms of industrial prototypes, making PCM among the most viable emerging technology for the next generation storage devices [23], [34].

The concept of using the resistance alteration in phase-change material for storage has been known for more than forty years now [32]. Historically, the performance of resistive memories was not on par with the contemporary solid state and DRAM storage alternatives. During the past 15 years, there has been an unprecedented growth in this technology driven by its desirable characteristics and extensive research in the field. A number of recent work have shown significant improvements in memory performance by integrating PCM within the hybrid storage hierarchy [14], [22], [35], [39].

Previous PCM research introduced methods for rewriting to the memory cells such that the writes to all bits have a uniform distribution. The heavily used written lines are remapped to the less frequently utilized locations by the memory management unit [28]. It has been demonstrated that the PCM endurance, reliability, and energy consumption would greatly improve if the redundant writes are avoided, i.e., by reading the existing contents of the bits and only programming those bits that must be changed. The method is called data-comparison-write (DCW), [38].

Flip-N-Write (FNW) is a protocol that adds an indicator bit to each word to determine if the word is inverted or not [13]. PCM controller can write the data in an inverted form if it requires less number of bit changes to reduce bit transition costs. Due to the noticeable asymmetry in cell-write energies of multi-level cell memories such as PCM and PRAM, a few encoding schemes also suggested bit-flip-based approaches to reduce the number of read and writes on expensive memory cells [17], [33]. Our paper formalizes, provides proofs and generalizes the bit-flip based methods beyond adding a single bit/cell by devising codes of length $N+K$ for words of length N, where $K \geq 1$. We consider the asymmetric set and reset energy costs. We will show that significant improvements in write energy and endurance are achieved at the expense of allowing a few extra storage bits.

Some of the existing digital storage mechanisms, including the optical storage, only allow for one directional transition of the bits. Write-once memory (WOM) encoding was introduced in a classic paper by Rivest and Shamir [30] to increase the number of writes to such memories with one directional bit setting (in an irreversible fashion). A flurry of subsequent research have centered on improving and generalizing the WOM codes and to extend its reach to other memory models. For example, NAND flash memory has been modeled as a one-way transitional memory. Thus, generalizations of the WOM codes have been applied to this class of memories [18], [24], [36].

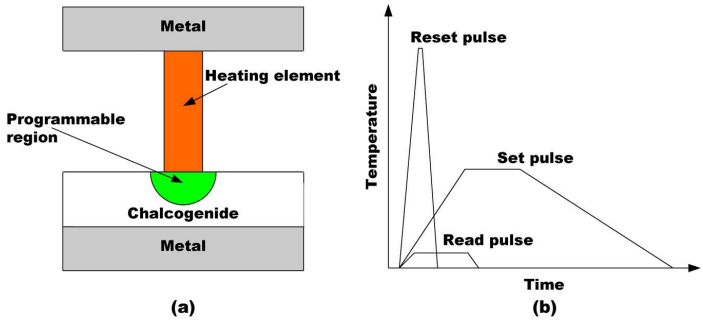

Fig. 1. (a) Cross section of a PCM cell. (b) Electric current amplitude and duration control the set, reset, and read operations.

For the PCM devices, the WOM model and the flash encoding methods do not correctly capture the specifics of the technology. One reason is that the energy discrepancy ratio between the set and reset commands on the PCM is much less subtle when compared to the NAND flash memory devices. The other (perhaps more important) reason, is the ability to perform bit-level manipulation on the PCM, as opposed to block-level operations on NAND flash. The bit-level operations for PCM have been used earlier for error correcting codes [31]. The work in [31] focused on developing error correction for PCM. Since the faulty bits are rather static, they have demonstrated that error correcting pointers (ECP) that include the knowledge of the fault location, are much more efficient than classic error correcting codes (ECCs). Error correction is orthogonal to our write efficient data encoding method.

Write-efficient memory or WEM is an extension of WOM that has been introduced in [8]. The objective of WEM codes is to minimize the overall number of transitions, and therefore, its goal is close to our encoding case. However, WEM's underlying assumption is that the costs of a set and reset are equal. Also, to the best of our knowledge, the few papers available on WEM have mainly focused on developing bounds but did not provide a coding method, or an optimality guarantee, or they centered on constructing suitable error correcting codes, e.g., [16], [26]. Aside from the loose bounds and the error correction, we have not been able to find WEM codes that are applicable to the PCM. Besides, we did not find a transform or proof of the problem's NP-completeness in the earlier literature, even though the concept was suggested.

III. PCM Operation and Energy Model

Single-Level Cell PCM: A key challenge for nonvolatile memory technology, in particular flash, is the high energy cost of writes [34]. The speed of writing and reading from the caches and from the DRAM is often high, and therefore, the number of transitions is higher than the external memories. Since resistive memory is suggested for replacing and complementing various storage units in the memory hierarchy, saving the energy cost of set and reset transitions is of a high value [28], [34].

As shown in Fig. 1(a), the current flows through the phase change material (chalcogenide) from the electrode/metal to the heater. This current is provided as a pulse, and its duration and amplitude controls the temperature needed for the set and reset operations. Heating the phase change material above a crystallization temperature by applying an average current with a wide duration pulse results in the set operation. A very high current (melt quenching) pulse with a short duration resets the device to its amorphous state. The read is done by applying a very low amplitude and low power pulse that senses the device resistance. The shape of the three pulses used for set, reset, and read commands are plotted in Fig. 1(b) ([34]). The energy discrepancy between the PCM set and reset operations has been experimentally demonstrated and quantified, e.g., [10].

Multi-Level Cell (MLC) PCM: The large difference between set and reset resistances in PCM has enabled devising multi-level cell PCM. As opposed to the conventional single-level PCM, the randomness and variability in MLC PCM structure makes it impossible to have a universal pulse shape to attain intermediate resistance levels. Instead, program and verify (P&V) is the technique that is used to obtain different resistance distributions for PCM [11]. P&V applies partial program pulses iteratively and then verifies if the desired cell level is achieved. The iterative approach causes MLC PCM to incur an order of magnitude more write energy than the single-level PCM.

One of the main challenges in prototyping MLC PCM is the relaxation effect that induces resistance drift in the phase-change material over time. The drift is particularly important in MLC PCM due to the high sensitivity of the cell state to resistance. Different memory sensing and error correction techniques have been proposed to develop more robust MLC PCM systems [12], [19], [37]. IBM has announced implementing the first drift-tolerant 2-bit cell PCM in 2011 [27].

IV. Overall Flow of Our Coding Method

Our coding scheme improves performance by reducing the cost incurred by bit transitions for writing data on PCM. We divide the input stream of data into N-bit portions and refer to them as N-bit words. There is a total of 2^N of such N-bit words. Our objective is to design codes such that the cost of writing codes is less than the cost of writing words. We develop multiple codes for each word. Instead of writing a new word on the memory, we write one of its codes. For choosing the best code, we first read the data which is going to be overwritten by the new word. Next among all the codes of the new word, we choose the one that incurs minimum bit transition cost for overwriting the existing data. The following example sheds light on how the coding scheme works and why it is beneficial.

A Word Encoding/Decoding Example: In this example, we provide the optimal solution for encoding 2-bit words with 3-bit codes. We denote the words by $A = 00$, $B = 01$, $C = 10$, and $D = 11$; and denote the codes corresponding to the word A by A_1 and A_2, the codes for B by B_1 and B_2, and so on. We denote by E_S and E_R, the energy spent on a bit set and a bit reset, respectively.

Fig. 2 shows our developed codes. The coding improves the energy consumption in the following way. Assume that we want to overwrite word B by C. The energy cost of a direct overwrite is $E_S + E_R$. However, if we apply the coding scheme, B is written on the memory (encoded) either as $B_1 = 010$ or as $B_2 = 101$. In case B is encoded as B_1, between the the two possible codes for C, we choose $C_2 = 011$ to overwrite B_1,

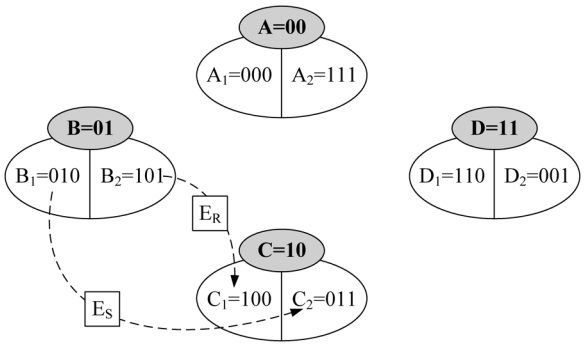

Fig. 2. A 3-bit encoding for 2-bit words. Words are A, B, C, and D; Each word has two codes/representations. The codes for A are A_1 and A_2, the codes for B are B_1 and B_2 and so on. E_S and E_R are set and reset energies. Cost of overwriting C by B is reduced from $E_S + E_R$ (without any coding) to either E_S or E_R (using our encoding).

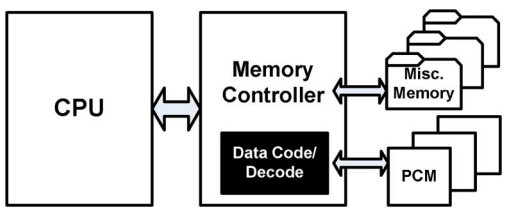

Fig. 3. Data encoder/decoder is shown as an embedded module in the memory controller.

since the overwrite costs only a bit reset or E_R. Note that we do not change the similar bits of B_1 and C_2 while overwriting. On the other hand, if B is currently coded as B_2, then we encode C by $C_1 = 100$, since the overwrite cost would be a bit reset or E_R. Thus, we have reduced the overwrite cost from $E_S + E_R$ to either E_S or E_R.

Now we provide an example that shows how the coding improves the energy cost over a cycle of data overwrites. The following is a chain of overwriting words: $A \rightarrow B \rightarrow C \rightarrow D \rightarrow A$. Assume that A is coded as A_1; the best encoding of the words are as follows: $A_1 \rightarrow B_1 \rightarrow C_2 \rightarrow D_2 \rightarrow A_1$. The cost associated with the code overwrites is $E_S + E_S + E_R + E_R = 2.E_S + 2.E_R$. However, the cost for overwriting the words directly and without coding is $E_S + (E_S + E_R) + E_S + (2.E_R) = 3.E_S + 3.E_R$. Thus, the coding reduces the overwrite cost by one-third.

All the codes have equal lengths. Since we assign more than one code to each word, the size of the codes are more than the words. Thus, the efficiency is achieved at the expense of memory overhead. For a given budget for memory overhead, our algorithm develops the codes offline and the complexity of designing the codes does not affect the realtime performance of the system. The resulting codes from our algorithms are then saved in the memory controller which interfaces with the PCM on one side and with the processing unit on the other side. Fig. 3 presents an abstract view of the placement of the data encoding/decoding module for our method. The details of the architecture of the encoder/decoder module and its overhead will be discussed in Section IX.

V. Problem Formulation, Complexity, and Bounds

We provide an encoding schemes that assign multiple representations (or codes) to each word in the data set. We can formally define our problem as follows.

Problem: Minimize the transition cost of PCM writes.

Given: The word and the code (symbol) lengths in bits denoted by N and $N + K$ respectively, where $K \geq 1$. Each word is represented by 2^K symbols. The read, set, and reset costs of a memory cell are denoted by C_{read}, and C_S, and C_R, respectively.

Objective: Find the best codes for each word such that they minimize the average energy cost of overwrites. We refer to this problem as $\mathcal{P}(N, K)$.

A. Problem Formulation

We denote the words by $W_1, W_2, \ldots, W_{2^N}$ and denote the codes corresponding to word W_i by Z_{li}, where $1 \leq l \leq 2^K$. Function Φ finds the energy required to overwrite a currently written symbol by a symbol of the next word that would incur the minimum transition cost

$$\phi\left(Z_{li}, W'_l\right) = \min\left\{C(Z_{li}, Z_{l'i'}),\ \forall 1 \leq i' \leq 2^K\right\}. \quad (1)$$

The cost function C measures the cost of overwriting a symbol by another one. To overwrite Z_{li} with $Z_{l'i'}$, if N_S number of bit sets and N_R number of bit resets are needed, then C would be

$$C(Z_{li}, Z_{l'i'}) = (N + K).C_{\text{read}} + (N_S).C_S + (N_R).C_R. \quad (2)$$

In the above equation, the first term shows the cost for reading the bits of the existing symbol in the memory (Z_{li}). We ignore this cost as it is two orders of magnitude less than the cost of a reset. The next two terms show the cost for the overwrite process. Similar bits in the two symbols remain untouched. The objective function (OF) can be written as follows:

$$\min\left\{\mathcal{C}(N, K) = \frac{1}{2^{2N+K}} \sum_{1 \leq l, l' \leq 2^N} \sum_{1 \leq i \leq 2^K} \phi(Z_{li}, W_{l'})\right\}. \quad (3)$$

Function $\mathcal{C}(N, K)$ represents the average transition cost for all possible overwrites. The goal is to assign symbols/codes to words such that this cost is minimized.

B. Problem Complexity

We have expressed the energy minimizing coding problem as an instance of a distance-based graph clustering problem; each cluster corresponds to a word and the nodes that belong to a cluster are different codes for the cluster's associated word. The goal is to minimize the inter-cluster distances. In the energy minimizing encoding scenario, the inter-cluster distance is the average distance between the code symbols in one cluster and the closest code symbol in every other cluster. Our example demonstrates the interpretation of the data coding as a graph problem. Extensive prior work on distance-based graph clustering have shown that this problem is NP-complete [7]. In the following we prove that we can reduce the well-known

graph clustering to our encoding problem, and conclude that our problem is NP-complete as well.

We analyze the complexity of creating a PCM code that results in the minimum energy consumption for given costs of writing 0 after 1 and writing 1 after 0.

There are many degrees of freedom in the solution space that impact the complexity of the problem including uncertainty about the sequence of words that are written, the relative cost of recording 0 and 1, the number of bits used for each word, and how many symbols are used for each word. In order to make the problem tractable from the computation complexity point of view we assume that the input sequence of words is known, and that the number of used bits and the number of symbols per word are specified. Furthermore, to enable a clear mapping from a known NP-complete problem to our minimum energy problem, we consider (without loss of generality) a special case of this problem that we denote as the minimum energy (ME) problem.

We start by introducing the notation that follows from Fiat and Shamir [15]. The number of bits per word is denoted by k. Each word can be written a total of t times, corresponding to the number of generations, and there are v values (symbols) for each word. Fiat and Shamir's paper asks for $w(\langle v \rangle^t)$, i.e., for the minimum number of bits sufficient to write each of v values t times. Their main result is that the problem is NP-complete if v is at least three or the number of generations is least two. Otherwise the problem can be solved in polynomial time.

The relevant special case of our problem is the one in which the cost of writing 0 after 1 (setting), C_S, is at least somewhat more expensive than the energy cost of writing 1 after 0 (resetting), C_R, kt times. In addition, we restrict our instances of the input sequences to ones in which each symbol is written at most t times. The question in our energy minimization problem would become: is there a write once memory code for which the given sequence takes at most C_R energy.

The essence of any NP-completeness problem is to map it to a known NP-complete problem. In our case we use the generalized write once memory coding (WOM) problem in which we write v words t times using k bits. We map the WOM problem to our problem by keeping input parameters k, t, and v identical. It is easy to see that if our new problem is solved optimally than we have a solution for the WOM problem. The key observation is that we can accomplish the writings in ME only if we never write 0 after 1 at any position in any word. And that is exactly the condition that is required by the WOM problem. Therefore, ME is an NP-complete problem.

C. Optimal Bounds on the OF

In this part, we provide a lower bound for the OF. The average cost of overwriting each symbol Z_{li} with the other words is determined by the following formulation: $(1/(2^N-1))\sum_{l'} \phi(Z_{li}, W'_l)$ for $l' \neq l$ and $1 \geq l' \leq 2^N$. An optimal code assignment is the one that assigns each of the closest 2^N-1 symbols to Z_{li} to one of the words $W'_l \neq W_l$. This assignment gives the minimum average overwrite cost of the symbols.

We provide a lower bound for the OF as follows. First, we calculate the distances from each code Z_{li} to all the other $2^{N+K}-1$ possible codes. Next, the resulting distances are sorted and the average sum of the smallest 2^N-1 distances are calculated for each node. We compare our DP algorithm result with the optimal bound in our evaluations.

VI. Solving Energy Efficient Coding Problem

We propose two different approaches for addressing the problem formulated earlier. Our first solution is based on mapping the problem to an instance of an integer linear programming (ILP). The method finds the optimal coding for any given word/code width. The approach is discussed in details in the Section VI-A. Due to the complexity of the ILP approach which grows exponentially with the size of the coding problem (i.e., the word and code widths), we introduce another solution based on dynamic programming (DP) paradigm. The solution is designed for both uniform and stochastic data. It is presented in the Section VI-B.

A. Optimal Coding via Integer Linear Programming

An ILP problem formulation requires linear representation of the objective function and the constraints. To the best of our knowledge, ILP has not been used for addressing similar coding problems before. The variables in ILP take integer values. There is a combinatorial complexity associated with assigning values to the variables of our NP-complete problem. The OF represented in (3) is not linear since the function $\phi(.,.)$ is a distance minimization function. To map the OF into a linear format, we define variables to indicate the distance of each symbol in a cluster from its closest symbol in every other clusters. The OF is equivalent to the average of all these variables. Certain linear constrains are applied to ensure the variable meets the minimum distance criteria. The ILP method finds the optimal solution at the expense of runtimes exponentially increasing with the code size.

To formulate our objective function in a linear form, we define an index variable that for each symbol, keeps track of the index of the element (in each of the other clusters) with the minimum distance to the symbol. The following set of variables were used in our ILP formulation.

l, l'	Word indexes W_l or W'_l for $1 \leq l, l' \leq 2^N$.
i, i'	Code indexes within each cluster, $1 \leq i, i' \leq 2^K$.
Z_{li}	The ith code $\in W_l$ for all i.
$\Phi_{ll'i}$	$\phi(Z_{l'i}, W_l)$ for all l, l', i, and i'.
$w_{ll'ii'}$	$w(Z_{li}, Z_{l'i'})$ for all l, l', i, and i'.
$\Delta_{ll'ii'}$	$w_{ll'ii'} - \Phi_{ll'i}$ for all l, l', i, and i'.
X_{lij}	jth significant bit of Z_{li} for $1 \leq j \leq (N+K)$.
$F_{ll'ii'j}$	$w(X_{lij}, X_{l'i'j})$ for all l, l', i, i', and j.
$Id_{ll'ii'}$	An indicator binary; $=0$ iff $\Delta_{ll'ii'} = 0$ for all l, l', i, and i'.

The codes representing a word W_l are shown by Z_{li}; $\Phi_{ll'i}$ denotes the cost of overwriting Z_{li} by a code in $W_{l'}$ that requires the minimum overwrite energy; $w_{ll'ii'}$ is the cost of overwriting

two codes U_{li} and $U_{l'i'}$. Thus, $\Phi_{ll'i} = \min_{i'} w_{ll'ii'}$. Variable $Id_{ll'ii'}$ is an indicator binary variable that indicates if the closest code to Z_{il} in cluster l' is $Z_{i'l'}$ or not. Each code Z_{li} consists of $N+K$ bits and can be written as $(X_{liN+K}, \ldots, X_{li2}, X_{li1})$. The parameter $F_{ll'ii'j}$ is defined to be the cost of overwriting X_{lij} with $X_{l'i'j}$ and its range of values is as follows:

X_{lij}	$X_{l'i'j}$	$F_{ll'ii'j}$
0	0	0
0	1	C_S
1	0	C_R
1	1	0

Using the above variables, we define our OF and provide constraints to our problem in a way that conforms to the ILP format. Our OF, as written in (3), minimizes the average cost of overwriting the codes for all possible overwrites

$$OF: \min \frac{1}{2^N.2^N.2^K} \sum \Phi_{l'li} \text{ for all } l',\ l \text{ and } i \text{ variables.} \tag{4}$$

The following constraints define $\Phi_{ll'i}$.

C1. $\Delta_{ll'i} \geq 0$ for all l, l' and i variables;

C2. $\Sigma_{i' \in 1,\ldots,2^k} Id_{ll'ii'} \leq 2^K - 1$;

C3. $Id_{ll'ii'} \leq \Delta_{ll'ii'}$; and

C4. $C_R.(N+K).Id_{ll'ii'} \geq \Delta_{ll'ii'}$.

Constraints C1 and C2 set $\Phi_{ll'i}$ not greater than each distance $\Delta_{ll'i}$ and equal to at least one of them, respectively; Constraints C3 and C4 define the indicator variable based on the fact that $C_R.(N+K)$ is always grater than $\Delta_{ll'ii'}$.

The below linear constraints set $F_{ll'ii'j}$ to the desired value.

C5. $(1/(C_R + C_S))F_{ll'ii'j} + X_{lij} + X_{l'i'j} \leq 2$;

C6. $F_{ll'ii'j} - C_R.X_{lij} - C_S.X_{l'i'j} \leq 0$;

C7. $F_{ll'ii'j} - C_R.X_{lij} - C_R.X_{l'i'j} \geq 0$; and

C8. $F_{ll'ii'j} - C_S.X_{lij} - C_S.X_{l'i'j} \geq 0$.

The following constraint defines the distance $w_{ll'ii'}$.

C9. $w_{ll'ii'} = \Sigma_{1 \leq j \leq N+K} F_{ll'ii'j}$.

The next constraint is used to ensure that no code is assigned to more than one word. C_S is the minimum cost of overwriting two different codes.

C10. $w_{ll'ii'} \geq C_S$.

The output of the above ILP is the values of X_{lij} that constructs the codes U_{il}. The above constraints are all in linear format and can be readily implemented by any ILP solver. The complexity and runtime for solving the instances of the ILP for our NP-complete problem exponentially increases with the instance size. In our experiments, we have been able to find the optimal solution by using a limited version of an ILP solver licensed to one user for N and K ($N = 2, 3, 4$, $K = 1, 2$). If one has access to the commercial ILP solvers that run on the cloud or supercomputers, it is likely possible to find the optimal codes for the practical problems of longer code sizes. The longer runtimes can be tolerated since the ILP solution needs to be computed only once and off-line.

B. Coding Via Dynamic Programming

In this section, we present our fast and efficient algorithms for solving the energy efficient coding problem. We first explain our method for coding uniform data, where all words have the same frequency of occurrence. Next, we describe our data-aware technique that can be applied to scenarios where the distribution of the words is uneven and skewed.

1) Coding for Uniform Data: Here we first show the optimal coding for solving the $\mathcal{P}(N, 1)$. Next, we show how to devise the codes for any given $\mathcal{P}(N, k)$ based on the coding solutions for the smaller instances of N and K.

Coding For $\mathcal{P}(N, 1)$.

Claim: Optimal coding of $\mathcal{P}(N, 1)$, for any $N \geq 1$ is achieved by assigning the complement pairs to the words.

Proof: The optimal coding finds $2^K = 2$ symbols, each of size $N+1$, for each word. For now, let us assume that the cost of set and reset is equal. This makes the overwrite cost proportional to the number of bitwise differences for the codes. The average transition cost from each code Z_{li} to all the other words satisfies the following inequality:

$$\frac{1}{2^N - 1} \sum_{l'} \phi\left(Z_{li}, W'_l\right) \leq 0.\binom{N+1}{0} + \binom{N+1}{1} + \ldots + \frac{N-1}{2}\binom{N+1}{\left[\frac{N-1}{2}\right]} + O.\frac{N+1}{2}\binom{N+1}{\left[\frac{N+1}{2}\right]},\ for\ 1 \leq l' \leq 2^N$$

where $O = 1$ if N is odd and $O = 0$ otherwise. The right side of the inequality equals $(N+1)2^{N-1}$. The proof of the inequality is as follows. The nearest 2^N codes to Z_{li} should contain all the codes that have zero distance from it (that is Z_{li} itself). The number of such codes is $\binom{N+1}{0}$. It should also include all the codes that are in just one bit different from Z_{li}; the number of such codes is $\binom{N+1}{1}$. The next closest set of codes are the ones that are in different from Z_{li} in 2 bits and so on. We continue until we reach to the first closest 2^N codes to Z_{li}. In that case, the number of bit differences reach to $(N-1)/2$ when N is even and $(N+1)/2$ when N is odd. This is because the following equation holds:

$$\binom{N+1}{0} + \binom{N+1}{1} + \ldots + \binom{N+1}{\left[\frac{N-1}{2}\right]} + O\binom{N+1}{\left[\frac{N+1}{2}\right]} = 2^N$$

where O is the same as defined before.

Now, we show that the complement-pair coding assigns all the above 2^N codes to different words. In this case, the average transition cost for each code Z_{li} will be equal to its optimal value and thus the optimal OF is achieved. The sum of bitwise differences of Z_{li} from any complement pair $(Z_{l'1}, Z_{l'2})$, is equal to $N+1$. This is because each bit of Z_{li} is equal to exactly one of the bits of the complement pair. Thus, one symbol of each word has a distance of less than $(N+1)/2$ bits and the other symbol has a distance of more than $(N+1)/2$ bits from Z_{li}. This means that all the $2^N - 1$ closest codes to Z_{li} belong to different words. ■

Note that our complement results for the $K = 1$ case also apply to the asymmetric set/reset costs. The number of sets and resets for traversing from a code to its complement is not symmetric for most of the code words. Recall that our objective is to minimize the average costs over all possible transitions. It can be readily shown that for achieving the mean cost, the average inter-complement distance can replace the two disparate transition costs between the complements.

Algorithm 1. Dynamic Programming for cost-aware coding

Inputs: Word length: N; Desired code length N+K; $\mathcal{C}(N,1)$; optimal codes for $\mathcal{P}(N,1)$ (Section VI-B1).

⋆ **Finding $\mathcal{C}(n,k)$ and the partitioning index $index(n,k,1:2)$:**

1 for (n=1 to n=N)
2 for (k=1 to k=K)
3 if (k==1)
4 $\mathcal{C}(n,k) = \mathcal{C}(n,1)$;
5 else
6 for (i=1 to i=n-1)
7 for (j=1 to j=k-1)
8 if ($\mathcal{C}(n,k) \geq \mathcal{C}(n-i,k-j) + \mathcal{C}(i,j)$)
9 $\mathcal{C}(n,k) = \mathcal{C}(n-i,k-j) + \mathcal{C}(i,j)$;
10 $index(N,K,1:2)=(i,j)$;

⋆ **Building the codes for $\mathcal{P}(N,K)$:**

11 for (n=1 to n=N)
12 for (k=1 to k=K)
13 if (k==1)
14 $\mathcal{P}(n,k) = \mathcal{P}(n,1)$ from Section VI-B1;
15 else
16 $\mathcal{P}(n,k)$ = code combinations from $\mathcal{P}(n - index(n,k,1),\ k - index(n,k,2))$ and $\mathcal{P}(index(n,k,1), index(n,k,2))$;

We introduce a DP-based algorithm for solving the general $\mathcal{P}(N,K)$ problem. Our algorithm uses the coding results for $\mathcal{P}(p,q)$ and $\mathcal{P}(r,s)$ to construct the codes for $P(p+r,q+s)$ such that the following bounds can be achieved:

$$\mathcal{C}(p+r,q+s) = \mathcal{C}(p,q) + \mathcal{C}(r,s). \tag{5}$$

The code construction is as follows. The word W_i of length $p+r$ is partitioned into two words, W_i^1 and W_i^2. The first word is the first p bits and the second word is the last r bits of W_i. There are 2^q, $p+q$-bit symbols for W_i^1 and 2^s, $r+s$-bit symbols for W_i^2 that are obtained from solving $\mathcal{P}(p,q)$ and $\mathcal{P}(r,s)$, respectively. We construct the codes for W_i by concatenating all the possible combinations of these two set of symbols which provides a total of $2^q.2^s = 2^{q+s}$ codes (of length $p+q+r+s$) for W_i. It can be easily seen that the codes satisfy (5). Based on the above code construction, the DP method breaks N into smaller values and selects the best partitioning to minimize

$$\mathcal{C}(N,K) = \min_{i \leq N} \left\{ \min_{j \leq i} \mathcal{C}(N-i,K-j) + \mathcal{C}(i,j) \right\}. \tag{6}$$

Algorithm 1 provides the details of the DP method. The optimal coding for $\mathcal{P}(N,1)$ is given from the previous part and the algorithm iteratively traverses over all the possible partitions to improve the write cost minimization objective (Lines 1–10). The index vector $index(n,k,1:2)$ is used to store the optimal partitioning of (n,k). After finding all the indexes, the algorithm builds the codes (Lines 11–16). The complexity of the algorithm is $O(N^2K^2)$, but recall that this algorithm is run offline.

2) Coding for Stochastic Data: In Section V, the OF (3) minimizes the average write cost for all the possible word overwrites. Here, we discuss how the inherent stochastic properties for real data scenarios can be exploited to further improve the memory's write performance. An important fact is that different words occur with differing frequencies. To benefit from this property, instead of weighting all the rewrite energy costs equally, we aggressively optimize our encoding for the rewrites that are more prevalent by assigning different number of codes to the words based on their frequency of occurrence.

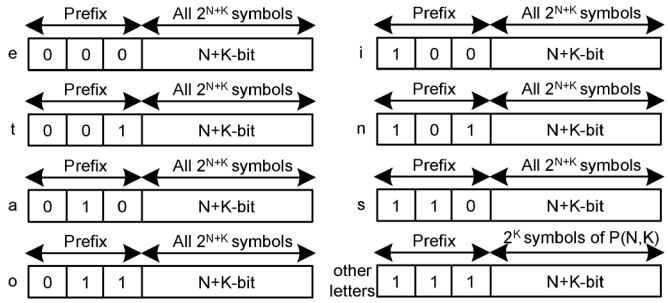

Fig. 4. Data-aware alphabet letter codings. Prefixes are set based on data distribution.

Variable-length and fixed-length coding are two statistical compression techniques. In the variable-length method, shorter codes are assigned to the more frequent words to better improve the compression. However, this adds to the decoding complexity and since our main goal is to minimize the write cost, decoding efficiency is very important. Thus, we use a fixed-length coding method. We describe our method on text files that contain English alphabet letters. The method can be generalized to other data sets with nonuniform frequencies. Our data consists of the lower-case alphabet letters: $W_1 = a$, $W_2 = b, \ldots, W_{26} = z$. Since there are 26 letters, we present W_i's with 5-bit words.

Let us consider the first seven most frequent letters of the table, e, t, a, o, i, n, and s. The probability that an overwrite occurs on any of these letters (by any other letter) plus the probability that these letters overwrite any other letter accounts for almost 60% of all probable overwrites. Thus, we can benefit a lot by optimizing our coding for these seven letters. To do so, we assign a different prefix to each of these letters such that only the prefixes determine the letter. Since there are 7 letters, the prefixes are 3-bit each and are shown in Fig. 4. The prefixes can be interpreted as dictionary indexes. The remaining $N+K$ bits of these letters take all the possible 2^{N+K} states. Thus, an overwrite to/by any of these letters requires only updating the prefix that is of length 3. The other 19 letters have the prefix (111) as shown in the figure. The remaining $N+K$ bits for the less frequent letters are filled with the codes obtained from solving $\mathcal{P}(N,K)$. By this coding, we assign 2^{N+K} symbols to the highly frequent letters and 2^K codes to the rest of the letters. All the symbols are of length $\text{prefix} - \text{length} + N + K$.

VII. Effect of the Encoding on Memory Wear

In this section, we study our encoding scheme from the memory wear perspective. The write endurance of PCM, although orders of magnitude higher than Flash memories, is still limited and considerably less than DRAM. Wear leveling is a technique that is widely used to diminish the limited number of memory write cycles by managing data writes such that they are distributed uniformly across the memory. Wear leveling is performed by memory controller. The encoding scheme can be used along with any conventional wear leveling technique. After the memory controller decides the address to write the data based on the wear leveling method, the encoding module steps in and performs the encoding by first reading the memory at those addresses and then accordingly finding the best codes for the new data.

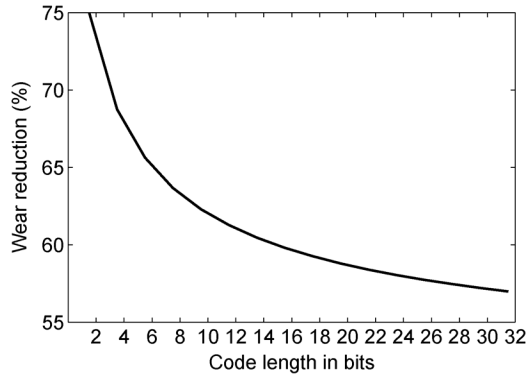

Fig. 5. PCM write endurance normalized to DCW method. Each word of length N (bits) is coded by codes of length N + 1.

The encoding improves the endurance of the memory by reducing the total number of writes (sets and resets). For example, for $\mathcal{P}(2,1)$ the total number of resets and sets per code write is 0.39. However, the average number of writes per word write if no coding is used is 0.5. In general, the average number of bit flips for overwriting the N + 1 − bit codes is the following:

$$\frac{1}{N+1}\frac{\Sigma_{1\le k\le \lfloor \frac{N+1}{2}\rfloor} k.\binom{N+1}{k}}{2^N} = \frac{1}{2} - \frac{1}{2}\frac{\binom{N}{\lfloor \frac{N}{2}\rfloor}}{2^N}. \quad (7)$$

The numerator represents the total number of bit flips required to write an arbitrary code by the closest code of each of the other 2^N words. As mentioned in the proof of optimal coding for $\mathcal{P}(N,1)$, our coding is designated such that it limits the number of bit flips in an overwrite to the following range: $1 \le k \le (N+1)/2$. Dividing by the denominator yields the average number bit-flips during a code overwrite. For the N-bit word data, without coding, the average number of bit flips is 1/2. The proof is straightforward due to the symmetry.

Fig. 5 shows the improved write endurance with our coding method compared to DCW method [38]. The words are of length N and the codes are of length $N+1$. For example, for a 2-bit word with 3-bit codes, the memory wear is improved by 50%.

Note that the wear efficiencies of codes of length $2N-1$ and $2N$ are equal. However, the memory overhead of an $2N$-bit code is $1/2N$ which is less than that of an $2N-1$-bit code that is $1/(2N-1)$. Thus, it is more efficient to use codes with even lengths for saving the memory capacity. Our dynamic programming algorithm takes this property into account.

Uniformity of Bit-Flips Per Single Bit Position: We claim that the number of bit-flips per single bit position is the same in our coding method. Let us consider a $\mathcal{P}(N,1)$ problem. The pair of codes (or representatives) for every word W_i are shown by $(Z_i, \bar{Z}_i)$, where according to our method $\bar{Z}_i$ is the ones' complement of Z_i. To overwrite a code Z_i by a representative of every other word w_j, our encoder chooses the best one between Z_j and $\bar{Z}_j$ which we denote by $Z_j^\star$. Now if apply a circular bit-shift on Z_i to get Z_i^{-1}, the same best code which is shifted by 1 (i.e., $Z_j^{\star -1}$) would be chosen to overwrite Z_i^{-1}. Thus the number of flips per bit will be shifted by one. Based on the same inference, on average, all the bits are flipped equal number of times. Since the solution to the general problem $\mathcal{P}(N,k)$ is created from a collection of smaller $\mathcal{P}(N_i,1)$ problems, the above claim applies to all problem sizes.

TABLE I
REQUIRED ENERGY FOR PROGRAMMING 2-CELL PCMS

Cell level	Energy (pJ)
00	36
01	307
10	547
11	20

TABLE II
2-CELL PCM WORDS AND THEIR CORRESPONDING 3-CELL CODES. USING OUR ENCODING, THE TOTAL NUMBER OF COSTLY INTERMEDIATE CELLS (01 AND 10) IS REDUCED FROM 16 IN THE 2-CELL WORDS TO 8 IN THE 3-CELL CODES

2-cell Words	3-cell Codes
00,00	00,00,00
00,**01**	00,00,11
00,**10**	00,11,00
00,11	00,11,11
01,00	**01**,00,00
01,**01**	**01**,00,11
01,**10**	**01**,11,00
01,11	**01**,11,11
10,00	**10**,00,00
10,**01**	**10**,00,11
10,**10**	**10**,11,00
10,11	**10**,11,11
11,00	11,00,00
11,**01**	11,00,11
11,**10**	11,11,00
11,11	11,11,11

VIII. MULTI-LEVEL CELL PCM

The properties of the Multi-level cell PCM varies from that of a Single-level cell PCM. Table I shows the average required energies for programming different levels of a 4-level cell PCM [11], [33].

Since the required energy for programming the intermediate levels, 01 and 10, is significantly higher than that of 00 and 11, we decided to encode the data such that the number of 01 and 10 cells are minimized. Our method assigns $N+1$-cell ($2N+2-$ bit) codes to N-cell (2N-bit) data. There are 2^{2N+2} different $N+1$-cell data. Our coding selects data of length $N+1$ cells that have the minimum number of intermediate levels and uses them to code the words of length N cells. Each word has only one code. Here, we provide an example for 2-cell word, 3-cell codes in Table II.

As it can be observed in the table, the total number of intermediate levels for the words is 16, whereas there is only 8 intermediate levels for the corresponding codes. To observe the energy efficiency of the code, we compare the average energy cost for writing uniform data on the memory before and after coding. We denote the four levels 00, 01, 10, and 11 with $L1$, $H1$, $H2$, and $L2$, respectively. For simplicity and due to the symmetry of the problem, we consider the write energies of L1 and L2 to be equal to the average of their individual energies; $(36+20)/2 = 28$. Likewise, $H1$ and $H2$ write energies are considered to be $(307+547)/2 = 427$. The average energy for writing the words is $8(L1+H1+H2+L2)/16 = 455$ and the average energy for writing the codes is $(20(L1+L2)+4(H1+H2))/16 = 183.75$. Thus, on average, the energy is reduced by almost 60%.

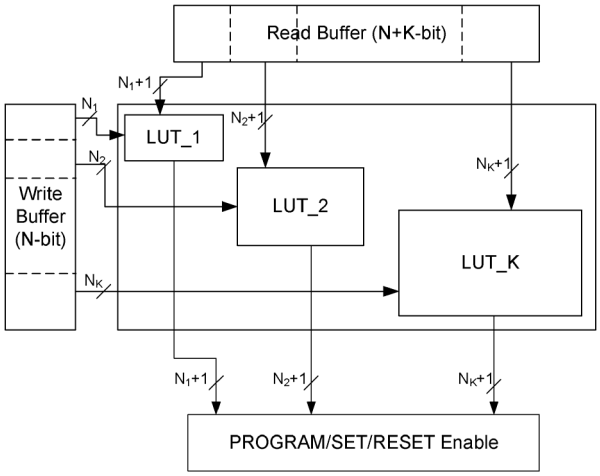

Fig. 6. Architecture of the encoder module. Read buffer contains the existing data from PCM and write buffer contains the new data that is going to replace the existing data. LookUp Tables (LUTs) are used for storing predetermined codes for $\mathcal{P}(N,1)$.

In the above example, despite the significant energy savings, the 33% reduction in memory capacity (3-cell codes are used to store 2-cell data) is not desirable. To address this problem, we propose coding N-cell data with N + 1-cell codes for larger N values. In this case, the memory capacity is reduced by a factor of $1/(N+1)$. The coding uses the same technique as the example and reduces the number of intermediate levels to save energy. We begin with assigning all the codes with zero intermediate levels to the words, then we assign all the codes with one intermediate levels to the words and so on until all the words have been assigned to a code.

Let us calculate how many N+1 cell codes with a given number of intermediate levels, say $0 \leq m$, are available. The answer is equal to the number of N + 1-cell data with exactly m, $H1$ and $H2$ levels and $N+1-m$, $L1$ and $L2$ levels. From combinatorics, this number is equal to the following:

$$\binom{N+1}{m}.2^{N+1} \text{ for } 1 \leq m \leq N+1. \tag{8}$$

We find the minimum m, such that all the 2^{2N} words are assigned to the codes

$$\Sigma_{0\leq m}\binom{N+1}{m}.2^{N+1} \leq 2^{2N}. \tag{9}$$

We denote the answer by $m_{\min}$. Given the answer, the average energy for a code write is $(1/2^{2N})\Sigma_{0\leq m\leq m_{\min}}\binom{N+1}{m}.2^{N+1}.(m \times 427 + (N+1-m)\times 28)$. The average energy for the corresponding word write is $(\lfloor N \rfloor/2)(427+28)$.

IX. Data Encoder/Decoder Architecture and Overhead

Fig. 6 shows the architecture of the encoding unit. Existing data in the current memory address is enqueued in the read buffer. New data that is going to overwrite the data in the current address is read and enqueued in the write buffer. A lookup table is employed for storing the matching codes; given the data in the write buffer and the data in the read buffer, the lookup table finds the code that incurs minimum energy for the overwrite.

TABLE III
Encoder Overhead for Various $\mathcal{P}(N,1)$'s on 45 nm

N	2	3	7
Critical Path delay (ns)	2.47	2.95	6.85
Energy (pJ)	0.11	0.16	3.00
Area (μm^2)	88.4	323.4	1243.2

To measure the encoder overhead, we model its performance in terms of area, delay, and energy. Let us first consider a $\mathcal{P}(N,1)$ problem. The encoder's timing and energy overheads are the sum of the following costs: 1) the cost of reading an existing code from the memory which is approximately $(N+1)T_{\text{read}}$ and $(N+1)E_{\text{read}}$, where T_{read} and E_{read} are the time and energy to read a bit of the PCM memory, receptively; 2) the cost of the lookup table.

We use the following PCM characteristics from [22]: $T_{\text{write}} = T_S = T_R = 95$ ns[1] and $T_{\text{read}} = 48$ ns; $E_{\text{write}} = E_S = E_R = 16.35$ pJ and $E_{\text{read}} = 2$ pJ.

For a $\mathcal{P}(N,1)$ problem, the encoder module is a lookup table with 2N+1 inputs (N bits for the word and N+1 bits for the code that is being written onto) and N+1 bits output (for the code that yields minimum cost). We have synthesized an ASIC design of the encoder module using Synopsis's Design Compiler tool and NanGate 45 nm library for N = 2, 3, and 7. The clock frequency is 500 MHz. Table III provides the synthesis performance report of our design. Given the defined parameters the overall time and energy overhead of $P(N,1)$ encoding method can be written as $T_{ov}(N) = T_{en}(N) + (N+1)T_{\text{read}}$ and $E_{ov}(N) = E_{en}(N) + (N+1)E_{\text{read}}$, respectively. It can be seen the read overhead dominates the total overhead for both energy and time.

Based on (7) and the fact that without any encoding the average amount of flips per bit is 1 per write operation (since all the bits are programmed), the average ratio of the time per write in our method to that of the no-coding scheme for $\mathcal{P}(N,1)$ can be written as

$$\frac{\frac{1}{2}\left(1-\frac{\binom{N}{\frac{N}{2}}}{2^N}\right)NT_{\text{write}}+T_{ov}(N)}{T_{\text{write}}}. \tag{10}$$

The ratio for energy is

$$\frac{\frac{1}{2}\left(1-\frac{\binom{N}{\frac{N}{2}}}{2^N}\right)E_{\text{write}}+E_{ov}(N)}{E_{\text{write}}}. \tag{11}$$

Algorithm 1 breaks down a $\mathcal{P}(N,K)$ problem into a number of $\mathcal{P}(N_i,1)$ sub-problems by finding the set of N_i's such that the following objective holds:

$$\min \sum_{1\leq i\leq K}\mathcal{P}(N_i,1),\ s.t.,\ \sum_{1\leq i\leq K}N_i = N. \tag{12}$$

[1]For simplicity, here we assume the costs of set and reset are equal. This assumption only lowers our encoding benefits since our method can take advantage of the asymmetry between set and reset costs.

Thus, we can write the **time ratio** for the general $\mathcal{P}(N,K)$ problem compared to the no-coding scheme as

$$\frac{\sum_{1\le i\le K}\left(\frac{1}{2}\left(1-\frac{\binom{N_i}{\frac{N_i}{2}}}{2^{N_i}}\right)N_iT_{\text{write}}+T_{ov}(N_i)\right)}{NT_{\text{write}}} \quad (13)$$

and the **energy ratio** as

$$\frac{\sum_{1\le i\le K}\left(\frac{1}{2}\left(1-\frac{\binom{N_i}{\frac{N_i}{2}}}{2^{N_i}}\right)N_iE_{\text{write}}+E_{ov}(N_i)\right)}{NE_{\text{write}}}. \quad (14)$$

In Section X-B we provide numerical values to better demonstrate the low overhead of our encoding method in terms of time, energy, and memory.

X. Evaluation Results

We perform system level evaluations of our methods on a variety of real world data sets. The effect of different word widths, different set and reset costs, and the memory overhead on the efficiency of our method are examined.

A. ILP Results

We used the latest version of Gurobi ILP solver [3]. Gurobi provides free access for academic purposes. The runtime of the solver for solving $\mathcal{P}(4,2)$ is about 30 h on a computer with an Intel dual core 2.80 GHz processor and 4 GB RAM. Due to the constraint of the academic version, we did not attempt to solve the objective function for larger problems. The python ILP code is available upon request to the interested readers. The advantage of the ILP method is that it provides optimal coding. Since defining the codes is an offline one-time process, it is feasible to employ more powerful supercomputing systems with full ILP version to run our solution for bigger code sizes.

B. Performance of DP-Based Algorithm on Uniform Data

The area overhead of a $\mathcal{P}(N,K)$ problem (other than the area of the encoder module) is K/N. Using (7), the endurance in terms of average flip-per-bit is

$$\frac{\sum_{1\le i\le K}\frac{1}{2}\left(1-\frac{\binom{N_i}{\frac{N_i}{2}}}{2^{N_i}}\right)N_i}{N}. \quad (15)$$

Based on (13), (14), and (15) which provide the overall efficiency of our method in terms of time, energy and endurance and also the PCM and encoder module specifications provided in Section IX, we can find the numerical values for performance in terms of time, energy, and area, Table IV.

Using the provided formulas for estimating the overhead and efficiency, one can generate Table IV for many pairs of (N,K) and choose the best encoding method that fits to their design goals. Note that the time improvement due to the reduction in the number of bit-flips almost offsets the time overhead of the encoder.

1) Performance Comparison With Other Encoding Methods: We analyze dynamic programming based encoding method in Algorithm 1 for different memory overheads. We compare our results with DCW and FNW algorithms [13], [38] that are described in Section II.

TABLE IV
Performance Improvement as a Result of Our Encoding, the Encoding Process Overhead, and the Net Improvement in Terms of Energy, Time, Endurance, and Area. All Values are Normalized to the Baseline No-Encoding Scheme (%)

(N,K)	(2,1)	(3,1)	(4,1)	(7,1)	(7,2)	(14,2)	(7,3)
Energy improvement	75.0	68.7	68.7	63.7	68.7	64.5	72.3
Encoder overhead	18.7	16.6	18.4	16.6	17.5	15.3	17.8
Net improvement	56.3	52.1	50.3	47.1	51.2	49.2	54.5
Time improvement	75.0	68.7	68.7	63.7	68.7	64.5	72.3
Encoder overhead	77.0	68.3	64.2	58.8	66.0	58.7	73.2
Net improvement	-2.0	0.4	4.5	4.9	2.7	5.8	-0.9
Endurance improvement	400.0	319.5	319.5	275.5	319.5	281.7	361.0
Area overhead=$\frac{K}{N+K}$	33.3	25.0	20.0	12.5	22.2	12.5	30.0

TABLE V
Comparing the ASCII Distribution-Aware Coding With the Uniform Coding. Costs are Normalized to the No-Coding Scheme. Energy Cost is Reduced by an Extra 8% in Distribution-Aware Coding

Coding scheme	distribution-aware	$\mathcal{P}(7,1)$	$\mathcal{P}(7,2)$	$\mathcal{P}(7,3)$
Normalized cost (%)	81.6	94.6	91.3	89.3

Here, we show the efficiencies for uniform data where all the word writes occur with the same frequency. Our metric is the average (per word write) cost for all possible word combination overwrites.

Fig. 7(a)–(c) shows the energy improvements of our method and the FNW method over the conventional DCW method for 8-bit, 16-bit, and 32-bit systems, respectively. The lined graph shows the results of our method and the black circles show the result of FNW. For example, a 25% memory overhead for a 32-bit system means that 8 extra bits is used for the each code. The results for FNW system are shown by circles since they can only take memory overheads of type $1/(N+1)$. Thus, in a 32-bit system, FNW only accept data-overhead sizes of 31-1, 30-2, 28-4, and 24-8.

Our method performs up to 16% better than FNW. There are two main reasons for the better performance. The first reason is the ability of our method to accept different overheads. For example, our dynamic programming algorithm partitions $\mathcal{P}(30,2)$ into $\mathcal{P}(14,1)+\mathcal{P}(16,1)$ as apposed to the FNW solution that is $2\times\mathcal{P}(15,1)$. Our partition, as we discussed in Section VII delivers better efficiency. Note that as the memory overhead increases, our methods become more efficient.

The second reason is our focus on cost-efficient selection of the codes to overwrite; FNW always flips the data if the number of bit-flips required to write the original data is more than half of the word's size. However, considering the asymmetric cost of set and reset, we do not count the number of bit-flips; instead, we count the total energy of set and resets as our metric to choose a code. Note that if for a new technology

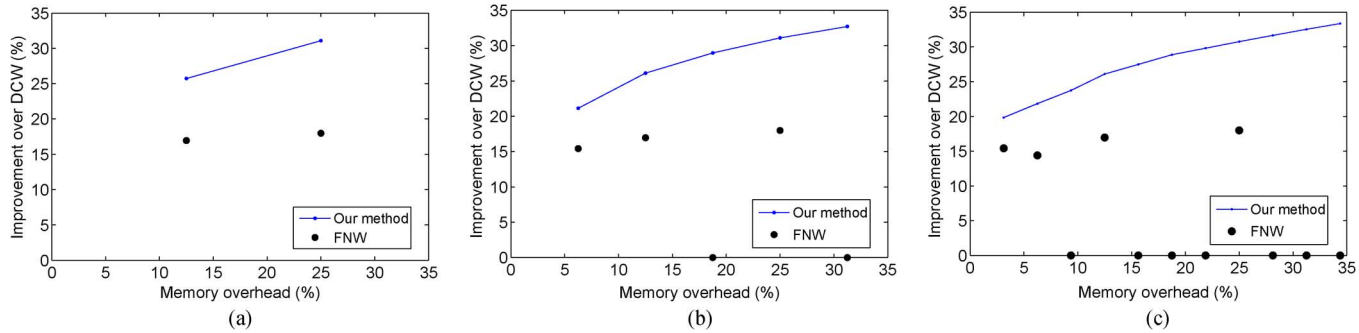

Fig. 7. Comparison of our encoding with FNW method in terms of average energy cost for all possible N-bit word overwrites. Values are normalized to those of DCW scheme. Results are shown for 3 N-*bit* systems, i.e., 3 $\mathcal{P}(N,1)$ problems. For the same memory overhead, our method can improve the energy cost up to 16% more in comparison with FNW. (a) 8-bit system. (b) 16-bit system. (c) 32-bit system.

the relation between the set and reset energies changes: e.g., $E_S > E_R$, then our coding can be easily modified to handle such behavior. For example, let us assume that a new word with codes $= (00000000, 11111111)$ is to overwrite 11110000. Our method (with the assumption $(E_R/E_S) > 1$) encodes the word with 00000000 and the overwrite energy consumption would become $4 \times E_S$. If $(E_R/E_S) < 1$, then our method encodes the word with 11111111 to reduce the overwrite energy consumption to $4 \times E_R$.

2) Performance Comparison With the Optimal Coding: We compare the performance of our dynamic programming-based algorithm with the optimal bounds (Section V-C). The following table provides the ratio of the overwrite costs in our scheme compared to the optimal coding cost (for different coding sizes). The results show the average energy cost for all possible data overwrites when $E_R = 2E_S$. According to the table, the performance gap increases with the memory overhead increment. However, our algorithm results are still close and in some cases equal to the optimal bound.

(N,K)	(4,1)	(4,2)	(8,1)	(8,2)	(8,3)	(8,4)
Ratio	1.0	1.0	1.0	1.01	1.07	1.11

3) Effect of the Asymmetric Set/Reset Energy Ratios: Here we look at the effect of different E_R/E_S ratios on the efficiency of our method. Fig. 8 shows the normalized energy costs for various memory overheads. As the ratio increases, more energy savings are achieved; for example, for a memory overhead of 30%, if the cost of a reset is four times the cost of a set $(E_R/E_S = 4)$ the efficiency is 8% more in comparison with the case where the costs are similar $(E_R/E_S = 1)$. The reason behind this behavior is that the new coding scheme aims to optimize the energy consumption by minimizing the overall cost of the overwrites (including all set and reset operations). When resets inuce a higher energy cost, the minimization impact will also be higher.

We always consider memory overheads of up to 33.34% (or equivalently Ks up to $N/2$). For $K > (N/2)$, our dynamic programming algorithm breaks the $\mathcal{P}(N,K)$ problem into at least one $\mathcal{P}(1,1)$. Such coding does not provide any improvements and only incurs overhead.

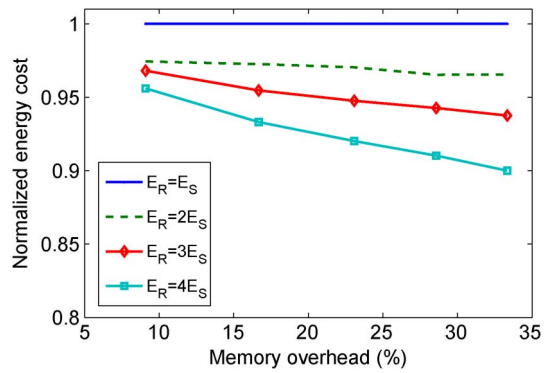

Fig. 8. Energy cost reduction by data-aware coding. As the gap between energies of set (E_S) and reset (E_R) increases, our encoding achieves more savings in comparison with the no-encoding scheme.

C. Performance on Audio and Image Data

We use the encoding method for storage of audio and image data on PCM. Our benchmark data were taken from Columbia University audio and Caltech Vision image databases, [2] and [1], respectively. Four audio and four image files were selected. The audio data are msmn1.wav, msmv1.wav, mssp1.wav, and msms1.wav and are denoted by $a1$, $a2$, $a3$, and $a4$ in and the image files are *dcp-2897.jpg*, *dcp-2898.jpg*, and *dcp-2899.jpg*, and *dcp-2830.jpg*.

We show the normalized average energy cost of overwriting all the audio files in Fig. 9 and the image files in Fig. 10. There are 12 possible overwrites for each file type. The costs are shown for 32-bit codings with various memory overheads with $E_R = 2E_S$. The cost of applying our coding method and FNW method are presented. The costs are normalized to the DCW method's cost. For some memory capacities, FNW cannot be applied and the in such cases its cost is set to be equal to the DCW cost. i.e., 100%. As expected, the gap between the performances become wider as the memory overhead increases. On the benchmark data sets our method outperforms the FNW method by up to 14% and 16%.

D. Performance of Stochastic Data Coding

We first provide evaluation results for the English alphabet coding as described in Section VI-B2. Then, we provide coding and evaluations for the ASCII characters. We used two text

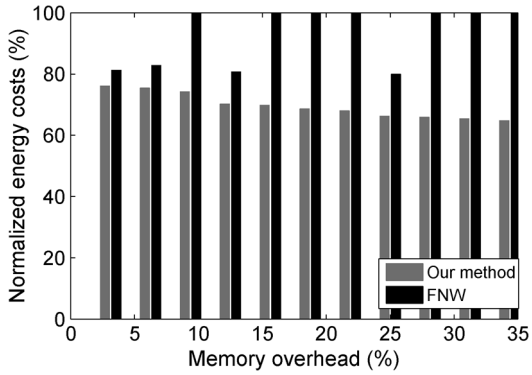

Fig. 9. Comparison of our method with FNW method in terms of energy costs. Encoding is done audio data, on a 32-bit system, and $E_R/E_S = 2$.

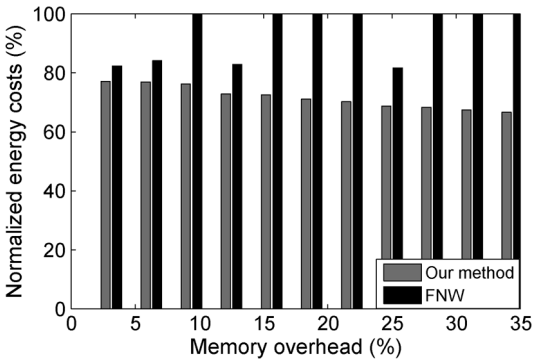

Fig. 10. Comparison of our method with FNW method in terms of energy costs. Encoding is done image data, on a 32-bit system, and $E_R/E_S = 2$.

benchmarks, the 31 MB *text8.txt* file from [6], for alphabet (excluding spaces) evaluations; and the 4.8 MB *KJV.txt* file from [4] for ASCII evaluations.

1) Alphabet Letters: We encode the alphabet letters with the data-aware encoding. Since there are 26 alphabet letters, $N = 5$; we set $K = 1$, and $\text{Prefix} - \text{length} = 3$. The codes are of length $\text{Prefix} - \text{length} + N + K = 9$. We evaluate the method on *Text8.txt* data for different test trials. For each trial, 100 pairs of vectors are created by randomly reading the data from the text file. Each vector has 1000 letters. We overwrote the vectors of each pair and computed the average overwrite cost for $E_R/E_S = 2$. The results demonstrate an average of 9.3% reduction compared to the uniform data coding of $\mathcal{P}(5, 2)$.

2) ASCII Characters: According to the frequencies of ASCII characters from [5], 59% of all the possible rewrites are to/by one of the first 15 most frequent characters out of the total 127 characters. Thus, we optimize our coding for these characters by assigning separate prefixes to them.

The first 15 most frequent characters are: space, e, t, a, o, i, n, s, h, r, d, l, u, m, c. We assigned the following 4-it prefixes to them, respectively: (0000), (0001), (0010), (0100), (1000), (1001), (1010), (0110), (0111), (1011), (1101). The prefix for all the other characters is (1111). Since there are 2^7 ASCII characters, $N = 7$ and we set $K = 1$. Thus, the codes will be of length $4 + N + K = 12$. The encoding method is the same as described for alphabet letters.

We evaluated the ASCII coding scheme on the *KJV.txt* file. We created 100 pairs of vectors, each of length 1000 from the file. The first vector in each pair was overwritten by the second vector. We assumed $E_R = 2E_S$. To compare this method with

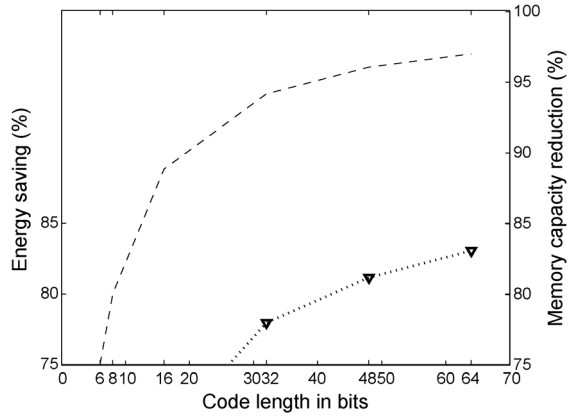

Fig. 11. Energy saving results for the proposed coding for 2-cell PCM. Dashed line shows the (normalized) average required energy for writing data compared to no-codeing method. Dotted line shows the capacity of the MLC PCM compared to no-coding method. For example, coding a 30-bit word with 32-bit codes results in 0.78 reduction in energy while it reduces the memory capacity by $1 - (30/32) = .07$.

the uniform coding, we encoded the ASCII characters with the codes from $\mathcal{P}(7, 1)$, $\mathcal{P}(7, 2)$, and $\mathcal{P}(7, 3)$ and report the corresponding average costs in the following.

We see that the ASCII data-aware coding, on average, reduces the energy cost more than the best uniform data coding. For overwriting each ASCII character, the energy cost is reduced by almost 8% in data-aware coding compared to the the uniform encoding results. This improvement is at the expense of assigning two extra code bits per character.

E. Performance of Multi-Level Cell Coding

Fig. 11 shows the result of our Multi-Level cell coding. The average energy reduction of the coded data for different code widths compared to the noncoded data is shown. Each N-cell word is coded with $N+1$-cell codes. Thus the overhead for such codes is $1/(N+1)$. The figure also shows the memory capacity usage of the coded data. It can be seen there is a tradeoff between the energy reduction and the capacity usage. The reductions are significant even for a small capacity losses. For example, the 16-cell codes incur only a 7% memory overhead in comparison with the no coding scheme but they reduce the write energy by 22%.

XI. Conclusion

We propose a novel data coding methodology for minimizing the write cost of PCM. Our approach improves both energy consumption and endurance of PCM. It flexibly enables trading off memory capacity with performance gain. The new data is encoded such that overwriting on the existing data incurs minimum cost. To address the coding problem, we develop 1) an ILP-based solution with a high combinational complexity that finds the codes optimally; 2) a dynamic programming-based approach that combines the smaller optimal codes to find near-optimal codes; and 3) an independent coding approach for multi-level cell PCM that reduces the number of costly intermediate level transitions to improve the performance. For cases where the distribution of the is *a priori* known, we create a new data-aware algorithm that incorporates those information for further optimizations. We also propose a low overhead encoder module.

Evaluations on a diverse set of text, image, and audio benchmark data demonstrate the applicability and effectiveness of our new methods. Our codings result in up to 16% energy improvement over the existing techniques for the single-level cell PCM and 22% improvement for multi-level cell PCM.

REFERENCES

[1] Caltech Computational Vision Data Repository [Online]. Available: http://www.vision\\.caltech.edu/html-files/archive.html

[2] Columbia University Sound Examples Repository [Online]. Available: http://labrosa.ee\\.columbia.edu/sounds

[3] Gurobi ILP Solver [Online]. Available: http://www.gurobi.com/

[4] The King James Bible (KJV) [Online]. Available: http://patriot.net/bmcgin/kjvpage.html

[5] Letter Frequency Counter [Online]. Available: http://millikeys.sourceforge.net/freqanalysis.html

[6] Text File Test Data [Online]. Available: http://mattmahoney.net/dc/textdata/

[7] T. Gonzalez, "Clustering to minimize the maximum intercluster distance," *Theor. Comput. Sci.*, vol. 38, pp. 293–306, 1985.

[8] R. Ahlswede and Z. Zhang, "Coding for write-efficient memory," *Inf. Comput.*, vol. 83, no. 1, pp. 80–97, 1989.

[9] R. Balasubramonian, D. Albonesi, A. Buyuktosunoglu, and S. Dwarkadas, "Memory hierarchy reconfiguration for energy and performance in general-purpose processor architectures," in *Proc. Int. Symp. Microarchit.*, 2000, pp. 245–257.

[10] F. Bedeschi *et al.*, "4-Mb MOSFET-selected µtrench phase-change memory experimental chip," *IEEE J. Solid-State Circuits*, vol. 40, no. 7, pp. 1557–1565, Jul. 2005.

[11] F. Bedeschi *et al.*, "A bipolar-selected phase change memory featuring multi-level cell storage," *IEEE J. Solid-State Circuits*, vol. 44, no. 1, pp. 217–227, Jan. 2009.

[12] S. Braga, A. Sanasi, A. Cabrini, and G. Torelli, "Voltage-driven partial-reset multilevel programming in phase-change memories," *IEEE Trans. Electron Devices*, vol. 57, no. 10, pp. 2556–2563, Oct. 2010.

[13] S. Cho and H. Lee, "Flip-N-Write: A simple deterministic technique to improve PRAM write performance, energy and endurance," in *Proc. Int. Symp. Microarchit.*, 2009, pp. 347–357.

[14] G. Dhiman, R. Ayoub, and T. Rosing, "PDRAM: A hybrid PRAM and DRAM main memory system," in *Design Automat. Conf.*, 2009, pp. 664–669.

[15] A. Fiat and A. Shamir, "Generalized 'write-once' memories," *IEEE Trans. Inf. Theory*, vol. 30, no. 3, pp. 470–480, May 1984.

[16] F.-W. Fu and R. Yeung, "On the capacity and error-correcting codes of write-efficient memories," *IEEE Trans. Inf. Theory*, vol. 46, no. 7, pp. 2299–2314, Nov. 2000.

[17] H. Hajimiri, P. Mishra, S. Bhunia, B. Long, Y. Li, and R. Jha, "Content-aware encoding for improving energy efficiency in multi-level cell resistive random access memory," in *Proc. IEEE/ACM Int. Symp. Nanoscale Archit.*, 2013, pp. 76–81.

[18] A. Jiang, M. Langberg, M. Schwartz, and J. Bruck, "Universal rewriting in constrained memories," in *Int. Symp. Inf. Theory*, 2009, pp. 1219–1223.

[19] M. Joshi, W. Zhang, and T. Li, "Mercury: A fast and energy-efficient multi-level cell based phase change memory system," in *Proc. IEEE Int. Symp. High Performance Comput. Archit.*, 2011, pp. 345–356.

[20] I. Kim *et al.*, "High performance pram cell scalable to sub-20 nm technology with below 4f2 cell size, extendable to dram applications," in *Symp. VLSI Technol.*, 2010, pp. 203–204.

[21] B. Lee, P. Zhou, J. Yang, Y. Zhang, B. Zhao, E. Ipek, O. Mutlu, and D. Burger, "Phase-change technology and the future of main memory," *IEEE Micro*, vol. 30, no. 1, p. 143, Jan./Feb. 2010.

[22] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger, "Architecting phase change memory as a scalable dram alternative," in *Int. Proc. Symp. Comput. Archit.*, 2009, pp. 2–13.

[23] K. Lee *et al.*, "A 90 nm 1.8 v 512 mb diode-switch pram with 266 mb/s read throughput," in *Proc. IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, 2007, pp. 472–616.

[24] H. Mahdavifar, P. Siegel, A. Vardy, J. Wolf, and E. Yaakobi, "A nearly optimal construction of flash codes," in *Proc. Int. Symp. Inf. Theory*, 2009, pp. 1239–1243.

[25] A. Mirhoseini, M. Potkonjak, and F. Koushanfar, "Coding-based energy minimization for phase change memory," in *Proc. Design Automat. Conf.*, 2012, pp. 68–76.

[26] T. Mittelholzer, L. Lastras-Montaando, M. Sharma, and M. Franceschini, "Rewritable storage channels with limited number of rewrite iterations," in *Proc. Int. Symp. Inf. Theory*, 2010, pp. 973–977.

[27] N. Papandreou *et al.*, "Drift-tolerant multilevel phase-change memory," in *Proc. IEEE Int. Memory Workshop*, 2011, pp. 1–4.

[28] M. K. Qureshi, J. Karidis, M. Franceschini, V. Srinivasan, L. Lastras, and B. Abali, "Enhancing lifetime and security of PCM-based main memory with start-gap wear leveling," in *Proc. Int. Symp. Microarchit.*, 2009, pp. 14–23.

[29] S. Raoux *et al.*, "Phase-change random access memory: A scalable technology," *IBM J. Res. Develop.*, vol. 52, no. 4–5, pp. 465–479, 2008.

[30] R. L. Rivest and A. Shamir, "How to reuse a write—Once memory (preliminary version)," in *Symp. Theory Comput.*, 1982, pp. 105–113.

[31] S. Schechter, G. H. Loh, K. Straus, and D. Burger, "Use ECP, not ECC, for hard failures in resistive memories," in *Proc. Int. Symp. Comput. Archit.*, 2010, pp. 141–152.

[32] C. Sie, "Memory devices using bistable resistivity in amorphous As-Te-Ge films," Ph.D. dissertation, Iowa State Univ., Ames, IA, 1969.

[33] J. Wang, X. Dong, G. Sun, D. Niu, and Y. Xie, "Energy-efficient multi-level cell phase-change memory system with data encoding," in *IEEE Int. Conf. Comput. Design*, 2011, pp. 175–182.

[34] H. Wong *et al.*, "Phase change memory," *Proc. IEEE*, vol. 98, no. 12, pp. 2201–2227, Dec. 2010.

[35] X. Wu, J. Li, L. Zhang, E. Speight, R. Rajamony, and Y. Xie, "Design exploration of hybrid caches with disparate memory technologies," *ACM Trans. Archit. Code Optimizat.*, vol. 7, pp. 1501–1534, 2010.

[36] Y. Wu and A. Jiang, "Position modulation code for rewriting write-once memories," *IEEE Trans. Inf. Theory*, vol. 57, no. 6, pp. 3692–3697, Jun. 2011.

[37] W. Xu and T. Zhang, "Using time-aware memory sensing to address resistance drift issue in multi-level phase change memory," in *Int. Symp. Qual. Electron. Design*, 2010, pp. 356–361.

[38] B.-D. 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," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2007, pp. 3014–3017.

[39] P. Zhou, B. Zhao, J. Yang, and Y. Zhang, "A durable and energy efficient main memory using phase change memory technology," in *Proc. Int. Symp. Comput. Archit.*, 2009, pp. 14–23.

Azalia Mirhoseini (S'12) is currently pursuing the Ph.D. degree in electrical and computer engineering at Rice University, Houston, TX, USA.

Her research interests include various aspects of machine learning, optimization, signal processing, and their applications to big data and emerging technologies.

Ms. Mirhoseini is the recipient of National Gold Medal in Iran Mathematics Olympiad (2004), Microsoft Women Graduate Student Scholarship (2010), IBM Ph.D. Student Scholarship (2012), and Schlumberger Ph.D. Student Fellowship (2013).

Miodrag Potkonjak (M'03) received the Ph.D. degree in electrical engineering and computer science from University of California, Berkeley, CA, USA, in 1991.

He is a Professor with Computer Science Department at the University of California, Los Angeles, CA, USA. He created first watermarking, fingerprinting, and metering techniques for integrated circuits as well as first remote trusted sensing and trusted synthesis approaches, compilation using untrusted tools, and public physical unclonable functions.

Farinaz Koushanfar (SM'13) received the Ph.D. degree in electrical engineering and computer science and the M.A. degree in statistics from the University of California, Berkeley, CA, USA, in 2005.

She is currently an Associate Professor with the Department of Electrical and Computer Engineering, Rice University, Houston, TX, USA. Her research interests include adaptive and low power embedded systems design, hardware security, and design intellectual property protection.

Dr. Koushanfar is a recipient of several awards and honors, including the Presidential Early Career Award for Scientists and Engineers, the ACM SIGDA Outstanding New Faculty Award, the NAS Kavli Foundation Fellowship, and the Young Faculty (or CAREER) Awards from the Army Research Office (ARO), Office of Naval Research (ONR), Defense Advanced Research Projects Agency (DARPA), and National Science Foundation (NSF).