Constrained Codes for Phase-change Memories

Anxiao (Andrew) Jiang
Computer Science and Eng. Dept.
Texas A&M University
College Station, TX 77843
Email: ajiang@cse.tamu.edu

Jehoshua Bruck
Electrical Engineering Department
California Institute of Technology
Pasadena, CA 91125
Email: bruck@caltech.edu

Hao Li
Computer Science and Eng. Dept.
Texas A&M University
College Station, TX 77843
Email: hao@cse.tamu.edu

***Abstract*—Phase-change memories (PCMs) are an important emerging non-volatile memory technology that uses amorphous and crystalline cell states to store data. The cell states are switched using high temperatures. As the semi-stable states of PCM cells are sensitive to temperatures, scaling down cell sizes can bring significant challenges. We consider two potential thermal-based interference problems as the cell density approaches its limit, and study new constrained codes for them.**

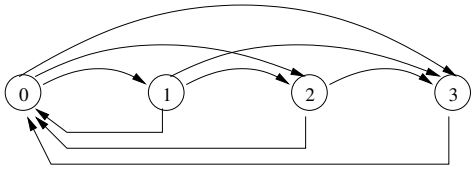

Fig. 1. The transitions among q cell levels. (Here $q = 4$.) The forward and backward edges represent the SET and RESET operations, respectively.

I. Introduction

Phase-change memories (PCMs) are an important emerging non-volatile memory (NVM) technology. Its basic storage unit, a PCM cell, has at least two states: the *amorphous state* and the *crystalline state*. To achieve higher storage capacity, multi-level cells (MLCs) are being developed, where additional *partially crystalline states* are used [1]. We model the $q \geq 2$ states of a PCM cell by q levels – levels $0, 1, \ldots, q-1$ – where level 0 is the amorphous state, level $1, \ldots, q-2$ are the partially crystalline states, and level $q-1$ is the crystalline state. As a cell becomes more crystallized, its level increases.

The level of a PCM cell is switched using high temperatures. A cell can be heated by a high cell-melting temperature (about $600^oC \sim 700^oC$) to change to level 0 (amorphous state), or be heated by a more moderate temperature to increase its level (i.e., to a more crystallized state). To model the direct switching of states, we use the diagram in Fig. 1 (for $q = 4$ as an example) [3]. We see that the cell can be changed from any level $i \in \{1, 2, \ldots, q-1\}$ directly to level 0 (called a *RESET* operation), and from any level i directly to level j for $0 \leq i < j \leq q-1$ (called a *SET* operation). However, for $0 < j < i \leq q-1$, to change it from level i to level j, both the RESET and SET operations are needed.

PCMs are under active study and development due to their very attractive potentials. Compared to the widely used flash memories, PCMs can potentially scale to much smaller cell sizes and achieve higher storage capacity. They can also have substantially better endurance, data retention and read/write speed [1]. However, scaling down cell sizes can also bring significant challenges, and solving them will be key to the PCM development [1]. A major and widely acknowledged challenge is the thermal issue, because the amorphous and partially-crystalline states are only semi-stable states, and high environmental temperatures can further crystalize the cell, i.e., unintentionally increase the cell level [1], [5].

We consider two potential challenges when the PCM's cell density scales toward its limit. The first one is the thermal crosstalk problem, namely, when a cell is RESET (to level 0) by the high melting temperature, the heat affects its adjacent cell and makes it further crystalized [1], [5]. Note that this may happen both when the adjacent cell is not being programmed and when it is being SET, unless it is already in the fully crystalized state (level $q-1$). It is because in both cases, the semi-stable cell state is sensitive to high temperatures.

The second problem is the local thermal accumulation problem. When cells are repeatedly programmed, the heat can accumulate in the area [5]. This residual heat can be a major factor that limits the writing bandwidth of PCMs, because the writing accuracy is sensitive to temperature [1], [5]. When the cell density scales toward its limit, relative to the high I/O speed, it can take nontrivial time for the locally generated heat to spread out uniformly in the memory chip. So if an application repeatedly writes a cluster of adjacent cells at very high speed, the accumulated heat may appear localized [5]. In that case, it is worth considering whether there exist schemes that can make the thermal accumulation more balanced.

The above problems have been considered in [1], [5] from the device and system perspectives. In this paper, we consider coding techniques for these potential challenges. For the thermal crosstalk problem, we use a scheme that removes the crosstalk interference, and then study coding techniques that reduce the programming cost (measured by the number of RESET operations). For the local thermal accumulation problem, we study coding techniques that impose time and space constraints on writing, to help the heat generated by programming be more balanced spatially.

Constrained coding is an important area of coding techniques, and has been applied successfully to both magnetic recording and optical recording [4]. Compared to conventional constrained coding, the codes studied in this paper are for a different setting and sometimes require very different coding

978-1-4244-8263-4/10/$26.00 ©2010 IEEE

techniques. It is also notable that for PCMs, the constraints are not limited to just within a codeword. They are introduced by the difference between old codewords and the new codewords that overwrite them, because only cells that are programmed can generate heat, which may affect other cells.

The rest of the paper is organized as follows. In Section II, we present symbol-constrained codes for the thermal crosstalk problem. In Section III, we present space-time constrained codes for the local thermal accumulation problem. In Section IV, we present the concluding remarks. Due to space limitation, we leave some detailed analysis in the full paper [2].

II. Symbol-constrained Codes

In this section, we study coding techniques for the thermal crosstalk problem. Let $c_1, c_2, \ldots, c_n$ be n cells. For $i \in \{1, 2, \ldots, n\} \triangleq [n]$, let $\ell_i \in \{0, 1, \ldots, q-1\}$ denote the level of c_i. In this paper, we consider the cells as a one-dimensional array. (The concepts can be extended to higher dimensions, too.) Two cells c_i and c_j are neighbors if and only if $|i-j| = 1$. Let $\gamma \in \{1, 2, \ldots, q-1\}$ be a parameter.

We consider the following simple model for thermal crosstalk: *When a cell c_i is RESET to level 0, the thermal crosstalk from c_i (at that moment) will increase its neighboring cell' level ℓ_j (for $j = i \pm 1$) by at most γ, unless the neighboring cell c_j is also being RESET at that moment.* (However, a cell level cannot exceed $q-1$, the stable fully-crystalline state. And a SET operation does not affect the neighboring cells due to its considerably lower temperature.)

Let $\mathcal{C} \subseteq \{0, 1, \ldots, q-1\}^n$ be a code, whose rate $R(\mathcal{C})$ is defined as $\frac{\log_2 |\mathcal{C}|}{n}$. (Clearly, $R(\mathcal{C}) \leq \log_2 q$.) A *rewrite* is to change the cell levels from the current codeword $X = (x_1, \ldots, x_n) \in \mathcal{C}$ to a new codeword $Y = (y_1, \ldots, y_n) \in \mathcal{C}$. For $i \in [n]$, if $x_i > y_i$, then the rewrite needs to RESET c_i (and then SET c_i if $y_i > 0$). For $j = i \pm 1$, if c_i is RESET and $y_j < \min\{x_j + \gamma, q-1\}$, then the rewrite needs to RESET c_j as well, because otherwise the thermal crosstalk from c_i may make ℓ_j greater than y_j. Therefore, a RESET operation applied to a cell can trigger the RESET of its neighboring cell, and this effect can propagate to many cells. Let us define a *RESET segment* in codeword Y as a maximum run of symbols $y_i, y_{i+1}, \ldots, y_j$ (where $1 \leq i \leq j \leq n$) such that: (1) $\forall\, i' \in \{i, \ldots, j\}$, $y_{i'} < \min\{x_{i'} + \gamma, q-1\}$; (2) $\exists\, i'' \in \{i, \ldots, j\}$ such that $x_{i''} > y_{i''}$. By our above analysis, the rewrite must RESET all the cells in a RESET segment (before setting them).

To rewrite cells using parallel programming, it is natural to use the following two-step procedure: *First, RESET all the cells in RESET segments of the new codeword; then, SET all the cells whose levels are still lower than their values in the new codeword.* (The second step has no crosstalk effect.)

Example 1. *Let $n = 11$, $q = 4$ and $\gamma = 3$. Assume the cells need to change from an old codeword $(1, 3, 2, 2, 2, 2, 2, 2, 1, 1, 1)$ to a new codeword $(0, 3, 2, 2, 1, 2, 2, 2, 3, 1, 2)$. First, we RESET the cells $\{c_1, c_3, c_4, c_5, c_6, c_7, c_8\}$. (After this step, the cell levels will be $(0, 3, 0, 0, 0, 0, 0, 0, \ell_9, 1, 1)$, where $\ell_9 \in \{1, 2, 3\}$. (Since cell c_8 is RESET, the thermal crosstalk from c_8 may make ℓ_9 be greater than its original value 1.) Then, we SET the cells $\{c_3, c_4, c_5, c_6, c_7, c_8, c_9, c_{11}\}$, to increase the cell levels to $(0, 3, 2, 2, 1, 2, 2, 2, 3, 1, 2)$.*

We define the *cost* of a rewrite operation as the number of cells that are RESET during rewriting. (In Example 1, the cost is 7.) The number of RESETs is a very important cost measurement because PCM cells have a limited longevity: PCM cells can endure about $10^6 \sim 10^8$ RESETs (or SET-RESET cycles) before becoming non-functional [1], [3]. Note that for the rewrite, for every cell that needs to decrease its level, the whole RESET segment containing it is forced (triggered) to be RESET, too. This motivates us to study constrained codes where RESET segments have *limited lengths*. Given this constraint, we seek capacity-achieving codes.

In this paper, we focus on the case $\gamma = q - 1$ (the worst-case scenario for thermal crosstalk). We define an *unstable segment* in a codeword $X = (x_1, \ldots, x_n)$ as a maximum run of symbols $x_i, x_{i+1}, \ldots, x_j$ (where $1 \leq i \leq j \leq n$) such that for $i' \in \{i, \ldots, j\}$, $x_{i'} < q - 1$. The *length* of this unstable segment is $j - i + 1$. When the memory writes X, an unstable segment in it will become a RESET segment if any of the cells in that unstable segment needs to decrease its level (compared to the old codeword). For a code $\mathcal{C}$, if in all its codewords the unstable segments' lengths are at most k, then during rewriting, the length of every RESET segment is at most k.

Definition 2. Symbol-constrained codes

Let k be a positive integer. A code $\mathcal{C} \subseteq \{0, 1, \ldots, q-1\}^n$ is k-limited if in every codeword of $\mathcal{C}$, every unstable-segment's length is at most k. (It is also called a symbol-constrained code.)

The k-limited codes are a constrained system $\mathcal{S}$ over alphabet $\Sigma \triangleq \{0, 1, \ldots, q-1\}$. An example for $q = 4, k = 3$ is shown in Fig. 2. We see that it generalizes the $(d = 0, k)$-run-length-limited (RLL) codes [4] from the binary alphabet to the q-ary alphabet. Its Shannon capacity is $cap(\mathcal{S}) = \lim_{n \to \infty} \sup \frac{1}{n} \log N(n; \mathcal{S})$, where $N(n; \mathcal{S})$ is the number of words of length n in $\mathcal{S}$. We have constructed a k-limited code for $q = 4$ and $k = 1$, which has a rate $6 : 5$ finite-state encoder. Its rate is 1.2 bits/cell, close to the Shannon capacity (which can be shown to be 1.203). When the code is used for storing data, assuming that the input information bits have a uniform i.i.d. distribution, for every rewrite, the ratio of the average number of RESET operations to the number of information bits is 0.228. This compares favorably with the no-coding method (i.e., storing 2 bits per cell), which has a higher ratio of 0.345. So symbol-constrained codes can reduce RESETs. (For more details of the code, see [2].)

We note that rewriting codes for reducing the RESET operations for PCMs have been studied in [3], where interesting WOM-like codes have been used. However, the study in [3] did not consider any thermal interference problem. We also stress that the codes in [3] and the codes we study are two drastically different approaches. While the codes in [3] always RESET

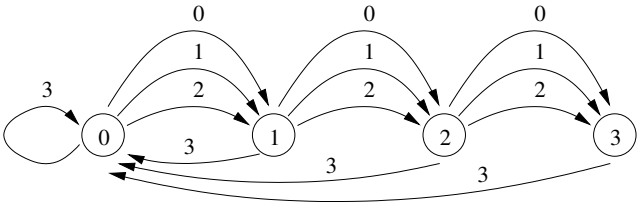

Fig. 2. Shannon cover of the 3-limited codes (constrained system), for $q = 4$.

TABLE I
SHANNON CAPACITY (BITS PER CELL) OF k-LIMITED CODES

$q\backslash k$	1	2	3	4	5	6
2	0.694	0.879	0.947	0.975	0.988	0.994
3	1.000	1.303	1.432	1.496	1.531	1.552
4	1.203	1.585	1.756	1.846	1.899	1.931
5	1.357	1.797	2.000	2.110	2.176	2.218
6	1.481	1.968	2.196	2.322	2.399	2.449
7	1.585	2.111	2.360	2.499	2.585	2.642
8	1.675	2.234	2.501	2.651	2.745	2.807
9	1.754	2.342	2.624	2.785	2.885	2.953
10	1.824	2.438	2.734	2.904	3.010	3.082
11	1.888	2.525	2.834	3.011	3.123	3.199
12	1.946	2.604	2.924	3.108	3.225	3.305
13	2.000	2.677	3.008	3.198	3.319	3.402
14	2.050	2.745	3.084	3.281	3.406	3.492
15	2.096	2.807	3.156	3.358	3.487	3.576
16	2.139	2.866	3.223	3.430	3.563	3.654

all cells at the same time (to get a fresh start for rewriting), we use constrained codes that are based on local constraints, and cells are most likely RESET in different rewrites. And with our constrained-coding approach, slide-block decoders can be built to locally decode information bits efficiently.

The following theorem presents the Shannon capacity of the symbol-constrained codes, for arbitrary q and k. Due to the space limitation, we present its full proof in [2].

Theorem 3. *Let $q \geq 2$ and $k \geq 1$ be integers. Let*

$$f(\lambda) = \lambda^{k+2} - q\lambda^{k+1} + (q-1)^{k+1}.$$

The equation $f(\lambda) = 0$ has at most three real-valued solutions, one of which is $q-1$. Among those real-valued solutions, if $q = k+2$, let λ^* be the solution with the greatest absolute value; otherwise, among the (at most two) real-valued solutions unequal to $q-1$, let λ^* be the solution with the greater absolute value. Then the Shannon capacity of k-limited codes is $\log_2 |\lambda^*|$ bits per cell.

Based on Theorem 3, the Shannon capacity of symbol-constrained codes for different q and k are shown in Table I.

III. SPACE-TIME CONSTRAINED CODES

In this section, we study coding techniques for a different interference problem: the local thermal accumulation problem. It is known that when cells are repeatedly programmed, adjacent cells can be crystallized/disturbed [5]. We seek codes for rewriting data that can balance heat better. This motivates us to study the space-time constrained codes defined below.

Let $c_1, \ldots, c_n$ be n PCM cells, whose levels are denoted by $\ell_1, \ldots, \ell_n \in \{0, \ldots, q-1\}$. Let $V = \{0, 1, \ldots, v-1\}$ be an alphabet of size v. The data stored in the n cells takes its value from the alphabet V. A code $\mathcal{C}$ is a mapping from the cell levels $L \triangleq (\ell_1, \ldots, \ell_n) \in \{0, \ldots, q-1\}^n$ to the data values V. We allow it to be a many-to-one mapping (instead of a one-to-one mapping). The code $\mathcal{C}$ has two associated functions: a *decoding function* F_d and an *update function* F_u. The decoding function $F_d : \{0, \ldots, q-1\}^n \to V$ tells us that the cell levels L represent the data $F_d(L) \in V$. The update function $F_u : \{0, \ldots, q-1\}^n \times V \to \{0, \ldots, q-1\}^n$ tells us that if the old cell levels are L and we want to write the new data $s \in V$ into the cells, we will change the cell levels to $F_u(L, s)$. (Clearly, we should have $F_d(F_u(L, s)) = s$.) A rewrite can change the data to any value in V. Here we do not consider the thermal crosstalk problem. So when a rewrite changes an old codeword $X = (x_1, \ldots, x_n) \in \{0, \ldots, q-1\}^n$ to a new codeword $Y = (y_1, \ldots, y_n)$, for $i \in [n]$, a cell c_i needs to be programmed only if $x_i \neq y_i$. We define the *rewrite cost* as the number of cells that are programmed, $|\{i \in [n] \mid x_i \neq y_i\}|$, which is the Hamming distance between X and Y. To balance programming-generated heat, we study the following code.[1]

Definition 4. SPACE-TIME CONSTRAINED CODES

Let α, β, p be positive integers. A code is (α, β, p)-constrained if for any α consecutive rewrites and for any segment of β cells – namely, $c_i, c_{i+1}, \ldots, c_{i+\beta-1}$ for some $i \in [n]$ – the total rewrite cost of those β cells (over those α rewrites) is at most p. (It is also called a space-time constrained code.)

We note that the space-time constrained codes are interesting because although the system can keep moving data that are frequently rewritten to balance heat, such an approach may cause substantial overhead for file-system/compiler design and their optimization. And for content-addressable systems, where the address of data is determined by the content of the data (e.g., by using a hash function) for fast data retrieval, relocating data can also be very challenging. In this paper, as the starting point of understanding space-time constrained codes, we study the time and space constraints separately.

A. Time-constrained Codes

We first study time-constrained codes with $\alpha \geq 1, \beta = 1, p = 1$. This is the simple case where every cell can be programmed at most once in every α consecutive rewrites. Note that the rate of the code $\mathcal{C}$ is defined as $\frac{\log_2 v}{n}$ bits per cell. It is easy to see that a simple idea based on time division can give us a code of rate $\frac{\log_2 q}{\alpha}$ bits per cell, as follows: Let $n = \alpha\lceil \log_q v \rceil$, and divide the n cells evenly into α groups (call them the 0th, 1st, 2nd, $\ldots$, $(\alpha-1)$-th cell groups); for $i = 1, 2, 3 \cdots$, for the i-th rewrite we write the data into the $(i \mod \alpha)$-th cell group. When $n \to \infty$ (which also means

[1]The model can be generalized by differentiating the cost of RESET and SET operations. In PCMs, the RESET operation uses a higher temperature that the SET operation, but has a shorter duration of time.

$v \to \infty$), the code rate approaches $\frac{\log_2 q}{\alpha}$ bits per cell. So the question is if there exist codes of higher rates.

Note that the challenge for designing time-constrained codes is that we cannot afford to remember for every cell how long ago the cell was programmed for the last time (up to α past rewrites), because that alone will cost $\log_2 \alpha$ bits of storage space for every cell. (Consider the case $q = 2$.) So the programming of cells needs to be synchronized in some way so that this information cost can be reduced. We now present a general time-constrained code construction for $q = 2$ that uses the write-once memory (WOM) codes [6] as sub-codes.

Let $\mathcal{D}$ be a WOM code that stores data of alphabet size w in m cells of $q = 2$ levels. Denote the alphabet of the stored data by $W = \{0, 1, \ldots, w-1\}$. The code $\mathcal{D}$ also has a *decoding function* $F_{d(\mathcal{D})} : \{0,1\}^m \to W$ and an *update function* $F_{u(\mathcal{D})} : \{0,1\}^m \times W \to \{0,1\}^m$. WOM codes have a unique property: *with every rewrite, the cell levels can only increase, not decrease* [6]. Let t denote the number of rewrites the code $\mathcal{D}$ can guarantee to support. (Let the initial cell levels all be zero.) Clearly, due to the unique property of WOM codes, t is a finite number.

Example 5. *Let $w = 4$, $m = 3$, $q = 2$. Let $L' \triangleq (\ell'_1, \ell'_2, \ell'_3) \in \{0,1\}^3$ denote the three cell levels. The following WOM code $\mathcal{D}$ was presented by Rivest and Shamir [6] with $t = 2$:*

L'	000	100	010	001	110	101	011	111
$F_{d(\mathcal{D})}(L')$	0	1	2	3	3	2	1	0

If the $t = 2$ rewrites first write the data as 2, the rewrite it as 1, the code will first let L' be $(0,1,0)$, the change it to $(0,1,1)$.

Let $\mathcal{E}$ be an "elevator code" that mimics $\mathcal{D}$ but allows the cell levels to increase and decrease in a synchronized way, described as follows. $\mathcal{E}$ also stores data of alphabet size w in m cells of $q = 2$ levels. Plainly speaking, for the first α rewrites, $\mathcal{E}$ rewrites data in the same way as $\mathcal{D}$; then it pushes all the m cell levels to $q - 1 = 1$; for the next α rewrites, $\mathcal{E}$ rewrites data by decreasing cell levels, in exactly the opposite way of $\mathcal{D}$; then it pushes all the m cell levels to 0; then the third batch of α rewrites are implemented in the same way as $\mathcal{D}$ again; and so on. We now formally define the *decoding function* $F_{d(\mathcal{E})}$ and the *update function* $F_{u(\mathcal{E})}$ of $\mathcal{E}$. Let us call a sequence of rewrites the 0th, 1st, 2nd, 3rd $\cdots$ rewrites. For $i = 0, 1, 2 \ldots$, let L'_i denote the cell levels after the i-th rewrite, and let $e_i \in W$ denote the data that the i-th rewrite writes into the cells. (Clearly, we should have $F_{d(\mathcal{E})}(L'_i) = e_i$.) Then if $0 \le (i \mod 2\alpha) \le \alpha - 1$, $F_{d(\mathcal{E})}(L'_i) = F_{d(\mathcal{D})}(L'_i)$; otherwise, $F_{d(\mathcal{E})}(L'_i) = F_{d(\mathcal{D})}((q-1, \cdots, q-1) - L'_i)$. If $i \equiv 0 \mod 2\alpha$, $F_{u(\mathcal{E})}(L'_{i-1}, e_i) = F_{u(\mathcal{D})}((0, \cdots, 0), e_i)$. If $1 \le (i \mod 2\alpha) \le \alpha - 1$, $F_{u(\mathcal{E})}(L'_{i-1}, e_i) = F_{u(\mathcal{D})}(L'_{i-1}, e_i)$. If $i \equiv \alpha \mod 2\alpha$, $F_{u(\mathcal{E})}(L'_{i-1}, e_i) = (q-1, \ldots, q-1) - F_{u(\mathcal{D})}((0, \cdots, 0), e_i)$. If $\alpha + 1 \le (i \mod 2\alpha) \le 2\alpha - 1$, $F_{u(\mathcal{E})}(L'_{i-1}, e_i) = (q-1, \ldots, q-1) - F_{u(\mathcal{D})}((q-1, \cdots, q-1) - L'_{i-1}, e_i)$.

Example 6. *Let $\mathcal{D}$ be the WOM code in Example 5, and let $\mathcal{E}$ be the "elevator code" defined as above. Then when the rewrites change the data as $1 \to 2 \to 3 \to 2 \to 3 \to 1 \to \cdots$, the code $\mathcal{E}$ changes the cell levels as $(0,0,0) \to (1,0,0) \to (1,0,1) \to (1,1,1) \to (1,1,0) \to (0,1,0) \to (0,0,0) \to (0,0,1) \to (0,1,1) \to (1,1,1) \to \cdots$*

We now construct the code $\mathcal{C}$ that stores data of alphabet size v in n cells of $q = 2$ levels. Let $v = w^{\frac{t}{\gcd(t,\alpha)}}$, where $\gcd(t, \alpha)$ is the greatest common divisor of t and α. Let $n = \frac{m(t+\alpha)}{\gcd(t,\alpha)}$. We see the stored data as a vector $X = \left(x_1, x_2, \ldots, x_{\frac{t}{\gcd(t,\alpha)}}\right) \in \{0, 1, \ldots, w-1\}^{\frac{t}{\gcd(t,\alpha)}}$. For $i = 0, 1, 2 \cdots$, let X_i denote the data (vector) that the i-th rewrite writes into the n cells. We divide the n cells evenly into $\frac{t+\alpha}{\gcd(t,\alpha)}$ groups, and call them the 0th, 1st, $\cdots$, $(\gcd(t,\alpha) - 1)$-th groups. (Every cell group has m cells.) We implement a sequence rewrites as follows. For $i = 0, 1, 2 \ldots$, let $g(i) = \lfloor \frac{i}{\gcd(t,\alpha)} \rfloor$. Then for the i-th rewrite, the $\frac{t}{\gcd(t,\alpha)}$ elements of X_i are, respectively, written into the $\left(g(i) \mod \frac{t+\alpha}{\gcd(t,\alpha)}\right)$-th, $\left(g(i)+1 \mod \frac{t+\alpha}{\gcd(t,\alpha)}\right)$-th, $\cdots$, $\left(g(i) + \frac{t}{\gcd(t,\alpha)} - 1 \mod \frac{t+\alpha}{\gcd(t,\alpha)}\right)$-th cell groups. (After the rewrite, the data can be decoded from those cell groups as well.) Every cell group uses the "elevator code" $\mathcal{E}$ to rewrite data. (For a cell group, after it is used for t consecutive rewrites, all its cell levels will be pushed to zero or $q-1$ when the next rewrite comes. The it will rest for $\alpha - 1$ rewrites.)

We can see that every cell group will be programmed in $t+1$ consecutive rewrites, then not programmed for another $\alpha - 1$ consecutive rewrites, and then repeat this process. In such a period of $t+\alpha$ rewrites, the cell levels are either all increasing or all decreasing; since $q = 2$, every cell can be programmed only once. So every cell is programmed at most once for every α consecutive rewrites. So $\mathcal{C}$ is a time-constrained code.

The only detail left to specify is how to know the value of $i \mod 2(t+\alpha)$ when the i-th rewrite happens, which is needed in the above coding process. (It is used to compute $g(i)$ and to implement the "elevator code.") This value can be obtained by using a simple "counter" of $2(t+\alpha)$ cells of $q = 2$ levels. Let $\ell'_1, \ell'_2, \ldots, \ell'_{2(t+\alpha)}$ denote their levels. We cyclically program the cells; for every rewrite, we change the level of one cell. We see $\sum_{j=1}^{2(t+\alpha)-1} \left|\ell'_j - \ell'_{2(t+\alpha)}\right|$ equals $i \mod 2(t+\alpha)$. So we can get the wanted value, and every cell in the counter is programmed exact once for every $2(t+\alpha) > \alpha$ rewrites.

Let $w \to \infty$, fix t as a constant, and we choose the smallest m such the WOM code $\mathcal{D}$ exists. By the known results on WOM codes [6], when $t = 2$, $m \approx 1.294 \log_2 w$; when $t = 3$, $m \approx 1.549 \log_2 w$; $\cdots$; for sufficiently large t, $m \approx \frac{t}{\log_2 t} \cdot \log_2 w$. The rate of the time-constrained code $\mathcal{C}$ is $\lim_{w\to\infty} \frac{\log_2 v}{n+2(t+\alpha)} = \lim_{w\to\infty} \frac{\frac{t}{\gcd(t,\alpha)} \log_2 w}{\frac{m(t+\alpha)}{\gcd(t,\alpha)}} = \frac{t}{t+\alpha} \cdot \frac{\log_2 w}{m}$. By the known values of $\frac{\log_2 w}{m}$ [6], we show the rate of $\mathcal{C}$ in the following table, and compare it with $\frac{1}{\alpha}$ (the rate of the code using time sharing). We see that the code $\mathcal{C}$ can achieve a higher rate.

α		4	5	6	7	8
$1/\alpha$		0.250	0.200	0.167	0.143	0.125
rate of $\mathcal{C}$	$t=2$	0.258	0.221	0.193	0.172	0.155
	$t=3$	0.277	0.242	0.215	0.194	0.176
	$t=4$	0.280	0.249	0.224	0.204	0.187
	$t=5$	0.277	0.250	0.227	0.208	0.192

The following theorem presents an upper bound to the rate of time-constrained codes. Due to the space limitation, we present its sketched proof. For its full proof, please refer to [2].

Theorem 7. *Define* $v_{\max}$ *as*

$$v_{\max} \triangleq \max_{\Delta=1,2,\ldots,\lfloor \frac{n}{\alpha} \rfloor} \sum_{i=0}^{\Delta} \binom{n-(\alpha-1)\Delta}{i} (q-1)^i.$$

Then the rate of $(\alpha, 1, 1)$*-constrained codes that use* n *cells of* q *levels is upper bounded by* $(\log_2 v_{\max})/n$ *bits per cell.*

Proof: We present the sketch of the proof here. Given a time-constrained code $\mathcal{C}$ that stores data of alphabet size v in n cells of q levels, we greedily select a sequence of rewrites such that at each rewrite, the number of programmed cells is locally maximized. For $i = 1, 2, 3 \cdots$, let δ_i denote the number of cells programmed by the i-th rewrite. Construct a sequence of positive integers $\Delta_{-\alpha+2}, \ldots, \Delta_{-1}, \Delta_0, \Delta_1, \Delta_2, \Delta_3, \ldots$ this way. First, let $\Delta_0 = \Delta_{-1} = \cdots = \Delta_{-\alpha+2} = 0$; then, for $i = 1, 2, 3 \cdots$, let Δ_i be the smallest integer such that $v \leq \sum_{k=0}^{\Delta_i} \binom{n-\sum_{j=i-\alpha+1}^{i-1} \Delta_j}{k} (q-1)^k$. Based on the greedily chosen rewrite sequence, it can be proved by induction that for $i = 1, 2, 3 \ldots$, $\delta_i \geq \Delta_i$. It can also be shown that for $i > j \geq 1$, $\Delta_i \geq \Delta_j$. Since the sequence $\Delta_1, \Delta_2, \Delta_3 \cdots$ cannot keep strictly monotonically increasing, there must exist positive integers i^* and Δ such that when $i \geq i^*$, $\Delta_i = \Delta$. Based on the way that $\Delta_1, \Delta_2, \Delta_3 \cdots$ are defined, we see that the theorem holds. ∎

B. Space-constrained Codes

We now study space-constrained codes with $\alpha = 1$, $\beta \geq 1$ and $p = 1$. This is the simple case where for every segment of β cells – namely, $c_i, c_{i+1}, c_{i+\beta-1}$ for some $i \in [n]$ – a rewrite will program at most one cell in the segment. Note that the code uses n cells of q levels to store data of alphabet size v.

We derive an upper bound for the rate of space-constrained codes. Let $\vec{x} = (x_1, x_2, \ldots, x_n) \in \{0, 1, \ldots, q-1\}^n$ be a vector that is not equal to $(0, 0, \ldots, 0)$. We call $\vec{x}$ a *β-constrained vector* if for any two non-zero entries x_i and x_j in $\vec{x}$, we have $|i-j| \geq \beta$. Let $M_{n,\beta}$ be the set of all β-constrained vectors. We see that with a space-constrained code, if the current cell levels are $L' = (\ell'_1, \ell'_2, \ldots, \ell'_n)$, a rewrite can change it only to the cell-level states in the set $\{L' + \vec{x} \mid \vec{x} \in M_{n,\beta}\}$. (For all the entries in the vector $L' + \vec{x}$, take modulo q.) Since the stored data have v distinct values, and a rewrite can change the data from any value to any other value, we have $v \leq |M_{n,\beta}| + 1$. So for space-constrained codes that use n cells of q levels, the code rate is upper bounded by $\frac{\log_2(|M_{n,\beta}|+1)}{n}$ bits per cell.

We now compute the value of $|M_{n,\beta}|$. When $n \leq \beta$, $|M_{n,\beta}| = n(q-1)$ because only one entry in a vector $\vec{x} \in M_{n,\beta}$ can be non-zero. Now consider the case $n \geq \beta + 1$. Let $\vec{x} = (x_1, \ldots, x_n)$ be a generic vector in $M_{n,\beta}$. If $x_{n-1} = x_{n-2} = \cdots = x_{n-\beta+1} = 0$, then there are $|M_{n-\beta,\beta}| \cdot q + (q-1)$ ways to choose the values of $x_1, \ldots, x_{n-\beta}$ and x_n. If one of the elements in $\{x_{n-1}, x_{n-2}, \ldots, x_{n-\beta+1}\}$ is not zero, then x_n must be zero; and it is not hard to see that in this case, the number of choices for $\vec{x}$ is $|M_{n-1,\beta}| - |M_{n-\beta,\beta}|$. So we have the recursion

$$\begin{aligned} & |M_{n,\beta}| \\ = \; & |M_{n-\beta,\beta}| \cdot q + (q-1) + (|M_{n-1,\beta}| - |M_{n-\beta,\beta}|) \\ = \; & |M_{n-1,\beta}| + (q-1)\,|M_{n-\beta,\beta}| + q - 1 \end{aligned}$$

Along with the β initial values $|M_{n,\beta}| = n(q-1)$ for $n = 1, 2, \ldots, \beta$, we can use the above recursion to solve for $|M_{n,\beta}|$.

Note that when $q = 2$, the vectors in $M_{n,\beta}$ correspond to the codewords of length n in the $(d = \beta - 1, k = \infty)$-RLL constrained system [4], except that $M_{n,\beta}$ does not contain the all-zero codeword. Therefore, when $q = 2$ and $n \to \infty$, the rate of the space-constrained code is upper bounded by the capacity of the $(\beta - 1, \infty)$-RLL constrained system.

IV. Conclusion

In this paper, we consider thermal interference problems for PCMs, which can be challenging when the cell density scales toward its limit [1], [5]. We consider the thermal crosstalk problem and the local thermal accumulation problem, and propose new constrained codes for solving them. We have studied the capacity of the constrained codes and some code constructions. For the symbol-constrained codes, the capacity and code construction for small γ (corresponding to less serious crosstalk between cells) still need to be studied. For space-time constrained codes, the capacity and construction of codes with both space and time constraints still need to be understood. They remain as our future research topics.

Acknowledgment

This work was supported in part by the NSF CAREER Award CCF-0747415, the NSF grant ECCS-0802107, and by an NSF-NRI award.

References

[1] G. W. Burr *et al.*, "Phase change memory technology," *Journal of Vacuum Science and Technology*, vol. 28, no. 2, pp. 223–262, Mar. 2010.

[2] A. Jiang, J. Bruck and H. Li, "Constrained codes for phase-change memories," online: http://faculty.cse.tamu.edu/ajiang/PCM.pdf.

[3] L. A. Lastras-Montano, M. Franceschini , T. Mittelholzer, J. Karidis and M. Wegman, "On the lifetime of multilevel memories," in *Proc. IEEE International Symposium on Information Theory*, Seoul, Korea, 2009, pp. 1224–1228.

[4] B. H. Marcus, R. M. Roth and P. H. Siegel, *An introduction to coding for constrained systems*, 5th Edition, 2001, online: http://www.math.ubc.ca/marcus/Handbook/index.html.

[5] A. Pirovano *et al.*, "Reliability study of phase-change nonvolatile memories," *IEEE Trans. Device and Materials Reliability*, vol. 4, no. 3, pp. 422–427, Sep. 2004.

[6] R. L. Rivest and A. Shamir, "How to reuse a 'write-once' memory," *Information and Control*, vol. 55, pp. 1–19, 1982.