Efficient Constrained Codes That Enable Page Separation in Modern Flash Memories

Ahmed Hareedy, *Member, IEEE*, Simeng Zheng, *Student Member, IEEE*,
Paul Siegel, *Life Fellow, IEEE*, and Robert Calderbank, *Life Fellow, IEEE*

***Abstract*—The pivotal storage density win achieved by solid-state devices over magnetic devices in 2015 is a result of multiple innovations in physics, architecture, and signal processing. One of the most important innovations in that regard is enabling the storage of more than one bit per cell in the Flash device, i.e., having more than two charge levels per cell. Constrained coding is used in Flash devices to increase reliability via mitigating inter-cell interference that stems from charge propagation among cells. Recently, capacity-achieving constrained codes were introduced to serve that purpose in modern Flash devices, which have more than two levels per cell. While these codes result in minimal redundancy via exploiting the underlying physics, they result in non-negligible complexity increase and access speed limitation since pages cannot be read separately. In this paper, we suggest new constrained coding schemes that have low-complexity and preserve the desirable high access speed in modern Flash devices. The idea is to eliminate error-prone patterns by coding data either only on the left-most page (binary coding) or only on the two left-most pages (4-ary coding) while leaving data on all the remaining pages uncoded. Our coding schemes work for any number of levels $q \geq 4$ per cell, offer systematic encoding and decoding, and are capacity-approaching. Since the proposed schemes enable the separation of pages, except the two left-most pages in the case of 4-ary coding, we refer to them as *read-and-run (RR)* constrained coding schemes as opposed to schemes adopting *read-and-wait* for other pages. The 4-ary RR coding scheme is introduced in order to limit the rate loss incurred by the binary RR coding schemes, and we show that our 4-ary RR coding scheme is also competitive when it comes to complexity and error propagation. We analyze the new RR coding schemes and discuss their impact on the probability of occurrence of different charge levels. We also demonstrate the performance improvement achieved via RR coding on a practical triple-level cell Flash device.**

***Index Terms*—Constrained codes, lexicographic ordering, LOCO codes, reconfigurable codes, data storage, Flash memories, multi-level technology, reliability, access speed, read and run.**

I. Introduction

The history of constrained coding dates back to 1948, when Shannon represented a constrained sequence via a finite-state transition diagram (FSTD) and derived the capacity under a constraint [2]. Run-length-limited (RLL) codes were introduced by Tang and Bahl in 1970 to support the evolution of magnetic recording at that time [3], and these codes were based on lexicographic indexing. In 1973, Cover presented a result about enumerative coding [4] that will prove fundamental for the design of constrained codes based on lexicographic indexing decades later. Among other researchers, Franaszek developed constrained codes based on finite-state machines (FSMs) derived from FSTDs [5]. In 1983, Adler, Coppersmith, and Hassner introduced a systematic method to develop constrained codes based on FSMs [6]. Details about the history of constrained coding until 1998 are in [7].

Because of their ability to improve performance via eliminating error-prone data patterns and undesirable sequences, constrained codes have a plethora of applications. They find application in one-dimensional (1D) magnetic recording devices, both the old ones, which are based on peak detection, and the modern ones, which are based on sequence detection [8], [9]. They can also be combined with robust signal detection using machine learning [10]. They find application in the emerging two-dimensional (2D) magnetic recording devices as well [11], [12]. Moreover, constrained codes are used to achieve DC balance and self-calibration in optical recording devices [13] in addition to many computer standards for data transmission [14].

In Flash devices, charge propagation from cells programmed to high charge levels into cells programmed to lower charge levels is the main reason behind inter-cell interference (ICI) [15]. This is correct for any number q of charge levels per cell. Mitigating ICI results in remarkable lifetime gains in Flash as demonstrated in [16] for multi-level cell (MLC) Flash ($q = 4$). There are data patterns that are considered usual suspects for contributing most to ICI. Coding to eliminate data patterns resulting in consecutive levels $(q-1)0(q-1)$ was considered in [17] and [18]. Coding to eliminate data patterns resulting in consecutive levels $(q-1)\mu(q-1)$, also called level patterns, for all $\mu < q-1$, was presented in [16], [18], and [19].

A number of recent results revisited [3] and [4] in order to produce efficient constrained codes based on lexicographic indexing, and one example is [22]. Another example is [8], in which we introduced binary symmetric lexicographically-ordered constrained (S-LOCO) codes and demonstrated density gains in a modern 1D magnetic recording system. We extended our result to single-level cell (SLC) Flash memories ($q = 2$) [23] then to Flash memories with any number q of levels per cell [19]. Moreover, we devised a general method to design LOCO codes for any finite set of patterns to forbid

This work was supported in part by the TÜBİTAK 2232-B International Fellowship for Early Stage Researchers and in part by the NSF under Grant CCF-2212437. This article was presented in part at the 2022 IEEE International Conference on Communications (ICC) [1].

Ahmed Hareedy is with the Department of Electrical and Electronics Engineering, Middle East Technical University (METU), 06800 Ankara, Turkey (e-mail: ahareedy@metu.edu.tr).

Simeng Zheng and Paul Siegel are with the Department of Electrical and Computer Engineering, University of California, San Diego (UCSD), La Jolla, CA 92093 USA (e-mail: sizheng@ucsd.edu; psiegel@ucsd.edu).

Robert Calderbank is with the Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708 USA (e-mail: robert.calderbank@duke.edu).

[24], which will be useful in this paper. We studied the power spectra of binary LOCO codes in [25]. LOCO codes are capacity-achieving, simple to encode-decode, and easily reconfigurable [19], [24].

While the constrained codes in [18] and [19] are quite efficient in terms of rate, they require all Flash pages to be processed together, which negatively affects the access speed. In this paper, we propose binary *read-and-run (RR)* constrained coding schemes that allow pages to be accessed separately in modern Flash devices, thus preserving high access speed. Our binary RR coding schemes incur small rate loss and work for any Flash device with $q \geq 4$ levels per cell. The key idea is that the constrained code is applied only on one page, the left-most page, while no coding is applied on the other $\log_2 q - 1$ pages. We present a 2D RR coding scheme as well as a 1D RR coding scheme that is based on LOCO codes, and we name the latter binary RR-LOCO coding. Furthermore, we present a 1D 4-ary RR coding scheme that is based on LOCO codes, which we name 4-ary RR-LOCO coding, in order to further reduce the rate loss without impacting the device reliability. In particular, we apply constrained coding on two pages, the two left-most pages, while no coding is applied on the other $\log_2 q - 2$ pages. Therefore, all pages are separated except the two left-most ones. Our 4-ary RR coding scheme works for any Flash device with $q \geq 8$ levels per cell.[1] We show that our 4-ary RR coding scheme can even outperform the binary RR coding schemes at capacity-approaching rates in terms of both complexity and error propagation. There are techniques in the literature that allow page separation; however, they are either incurring notable rate loss [16] or designed for a specific Flash setup [17]. We study various aspects about the proposed RR coding schemes, including charge-level probabilities. We introduce experimental results in a practical triple-level cell (TLC) Flash device ($q = 8$) that demonstrate notable lifetime gains achieved by our coding schemes.

Signal processing techniques have also been proposed to mitigate ICI effects in Flash memory. Precompensation, or predistortion, methods [20] attempt to predict the ICI that will be experienced by a cell and modify the program level accordingly. Postcompensation, or postprocessing, methods [20], [21] attempt to estimate the ICI distortion after sensing the cell voltages and apply an appropriate correction to the cell read voltage to offset the ICI effect. Both approaches require accurate information about inter-cell coupling ratios over a range of program/erase cycles. They must compute an estimate of the expected ICI for every cell as a function of its program level (or read voltage) and those of its neighboring cells. As pointed out in [20], [21], this comes at a cost of additional processing either during programming or after reading, resulting in extra calculations, additional storage overhead, and added write or read latency. Furthermore, the ICI compensation, whether during programming or after reading, will not be exact, so there may be some residual ICI. On the other hand, results of modeling and simulation in [20], [21] give evidence that these signal processing methods can be very effective in mitigating ICI, and unlike constrained coding methods that introduce redundant data, they do not incur any rate loss penalty.

Given the many issues involved, it is difficult to identify one approach to ICI mitigation as universally superior to another without a careful assessment of the engineering trade-offs. However, further comparison of constrained coding methods with signal processing methods is an interesting topic for future research, as is the consideration of techniques that combine both methods in a complementary fashion.

The rest of the paper is organized as follows. In Section II, we discuss the detrimental patterns, the Flash mapping, and our 2D binary RR coding scheme. In Section III, we introduce our 1D binary RR-LOCO coding scheme. In Section IV, we propose our 1D 4-ary RR-LOCO coding scheme. In Section V, we study the rate, complexity, and error propagation of the new schemes and make comparisons. In Section VI, we present the experimental results on TLC Flash. In Section VII, we conclude the paper.

II. Patterns, Mapping, and 2D RR Coding

As implied in the introduction, literature works do not strictly agree on the set of forbidden patterns to operate on. Additionally, as the Flash device ages, the set of error-prone patterns is expected to expand [19]. According to our recent experimental tests and a machine learning-based ICI characterization [26] of TLC Flash memories, we decided to focus on the set characterized as follows. Let

$$\beta_1, \overline{\beta}_1 \in \mathcal{V}_0 \triangleq \left\{ \frac{q}{2}, \frac{q}{2} + 1, \ldots, q - 1 \right\}, \tag{1}$$

where q is the number of levels per Flash cell (a positive power of 2) and $\mathcal{V}_1 = \{0, 1, \ldots, q-1\} \setminus \mathcal{V}_0$. Then, the set of interest is the set resulting in the high-low-high level patterns in $\mathcal{L}_q$:[2]

$$\mathcal{L}_q \triangleq \{\beta_1 \mu \overline{\beta}_1, \forall \beta_1, \overline{\beta}_1 \mid 0 \leq \mu < \min(\beta_1, \overline{\beta}_1)\}. \tag{2}$$

This set already subsumes all 3-tuple forbidden patterns adopted in the literature for Flash. This set can be relaxed by removing few patterns that have minimal impact on performance as we shall see in Section IV. A block inside the Flash device can be seen as a 2D grid of wordlines and bitlines, with a cell being placed at each intersection [16]. Level patterns in $\mathcal{L}_q$ are detrimental whether they occur on 3 adjacent cells along the same wordline or along the same bitline.

Example 1. *Consider an MLC Flash device, i.e., $q = 4$. In this case, we have $\beta_1, \overline{\beta}_1 \in \{2, 3\}$. Then, the set of interest is the set resulting in:*

$$\mathcal{L}_4 = \{202, 212, 203, 213, 302, 312, 303, 313, 323\}. \tag{3}$$

The last three elements in $\mathcal{L}_4$ are quite known [16], [17], [19].

Next, we discuss how to map from data to charge levels in Flash and vice versa. Since we are interested in page separation throughout this work, the mapping here is from a charge level out of q possible ones to $\log_2 q$ binary bits, one for each page, and vice versa. Gray mapping offers the advantage that there is only one-bit difference between any two adjacent charge

[1] This 4-ary RR coding scheme works for $q = 4$ as well, but with more benign patterns forbidden and with no page separation.

[2] Levels are defined through their indices $\{0, 1, \ldots, q-1\}$ for simplicity.

Algorithm 1 Recursive Alternate Gray Mapping

Input: Number of levels per cell q, and $p = \log_2 q$.
Define map, a binary array of dimensions $q \times p$.
Set $\mathrm{map}(0,:) = \mathbf{1}^p$. *(a sequence of p 1's)*
for $i \in \{0, 1, \ldots, p-1\}$ **do**
  for $j \in \{0, 1, \ldots, 2^i - 1\}$ **do**
    $\mathrm{map}(2^i + j, :) = \mathrm{map}(2^i - 1 - j, :)$.
    Flip the bit $\mathrm{map}(2^i + j, i)$. *(each sequence in* map *is indexed from right to left by* $0, 1, \ldots, p-1$*)*
  end for
end for
Output: Array map that maps each index to binary data.

levels, which is valuable for error performance. We adopt a recursive alternate Gray mapping (RAGM), and Algorithm 1 shows how to produce it for any $q \geq 4$. We highlight that RAGM has already been used in the literature in MLC Flash [16] and TLC Flash [17].

Example 2. *Consider a TLC Flash device, i.e., $q = 8$. In this case, the output of Algorithm 1, which is RAGM, becomes:*

$$\begin{aligned} 0 &\longleftrightarrow 111, & 1 &\longleftrightarrow 110, \\ 2 &\longleftrightarrow 100, & 3 &\longleftrightarrow 101, \\ 4 &\longleftrightarrow 001, & 5 &\longleftrightarrow 000, \\ 6 &\longleftrightarrow 010, & 7 &\longleftrightarrow 011. \end{aligned} \tag{4}$$

Now, we are ready to discuss binary coding schemes. Let us first index the Flash pages the same way the bits in each sequence in the array map are indexed (see Algorithm 1). This means that the left-most page is the one indexed by $p-1$. From (2) and Algorithm 1, the level patterns in $\mathcal{L}_q$ correspond to binary patterns where the left-most page (pages) always has (have) two 0's separated by some bit, i.e., $0x0$. Based on that, forbidding $\{000, 010\}$ on the left-most page (pages) guarantees that no level pattern in $\mathcal{L}_q$ would appear while writing to a Flash device, with any $q \geq 4$, at least in the wordline (bitline) direction. This corresponds to an interleaved RLL $(d, k) = (0, 1)$ constraint [27]. Notably, no coding on any other page is needed. Data will therefore be read from each page independently, and immediately passed to the low-density parity-check (LDPC) decoder to start its processing. This idea is the key idea of our binary RR constrained coding schemes.[3]

RR coding can be performed in the wordline direction only (1D), the bitline direction only (1D), or both directions (2D). Observe that binary RR coding will also prevent some benign level patterns, e.g., 474, 555, and 676 in TLC Flash, resulting in inevitable rate loss. However, as we shall see in Section V, this rate loss is small. Furthermore, some of these benign level patterns will be allowed when we shift from binary to 4-ary coding, which reduces this rate loss, as we shall see in Section IV. RR-LOCO codes are capacity-approaching codes.

We start here with our scheme for 2D binary RR constrained coding. As the name suggests, we want to prevent the patterns in $\mathcal{R}^2 = \{000, 010\}$ from appearing at the left-most pages in both wordline and bitline directions in the Flash device

[3]An equivalent scheme was proposed for MLC Flash, i.e., $q = 4$, in [27].

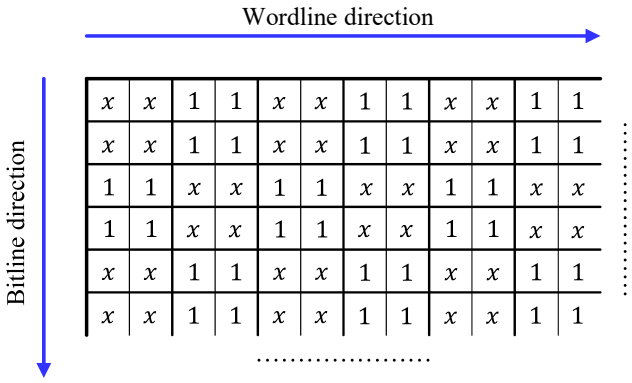

Fig. 1. The left-most pages of a 2D Flash grid with data encoded via the proposed 2D binary RR coding scheme. Symbol x means bit can be 0 or 1 freely.

through simple encoding and decoding. The encoding follows the rules:

1) On wordlines with indices congruent to 0 or 1 (mod 4), you are allowed to write 0's and 1's freely in bit positions congruent to 0 or 1 (mod 4) at the left-most pages.
2) On wordlines with indices congruent to 2 or 3 (mod 4), you are allowed to write 0's and 1's freely in bit positions congruent to 2 or 3 (mod 4) at the left-most pages.
3) In the other bit positions, you can only write 1's on wordlines at the left-most pages.

This 2D binary RR constrained coding scheme is depicted in Fig. 1. It is clear from the figure that the patterns in $\mathcal{R}^2 = \{000, 010\}$ are eliminated on the left-most pages, which forbids all level patterns in $\mathcal{L}_q$, in both directions. Upon encoding, input data bits are freely placed at the positions marked by x for the left-most pages, and they are directly placed (uncoded) at the other pages. Upon decoding, information at the positions marked by 1 is omitted, and data bits at the remaining positions are read with no additional processing and with no correlation between different Flash pages.[4]

This 2D binary scheme is ideal in terms of complexity, access speed, and error propagation (see Section V). It might also seem notably better than any 1D scheme (binary or 4-ary) in terms of performance. However, 1D schemes can achieve almost the same performance with higher rates, which we will discuss in more detail later.

III. RR-LOCO Coding Over GF(2)

In order to design efficient constrained codes, LOCO codes, we adopt the general method in [24]. The steps of this general method are:

1) *Use the forbidden patterns to determine a group structure (partition) for the code.*
2) *Derive the code cardinality (codebook size) formula using the inherent recursion of the groups and subgroups.*

[4]An equivalent 2D scheme forbidding patterns $\{101, 111\}$ on the right-most pages in both wordline and bitline directions in MLC Flash was proposed in [27].

3) *Specify the codeword patterns that represent special cases given the forbidden patterns. Details about these cases will be discussed shortly.*
4) *Find the contribution of a non-zero codeword symbol to the codeword lexicographic index in each special/typical case.*
5) *Merge the contributions for all cases in one codeword-index equation, which is the (LOCO) encoding-decoding rule.*
6) *Develop the encoding and decoding algorithms of the code based on this rule.*

In this section, we introduce a binary RR coding scheme that forbids $\{000, 010\}$ on the left-most pages in either the wordline direction or the bitline direction, while leaving all other pages with no coding, which forbids the level patterns in $\mathcal{L}_q$ and achieves page separation. This scheme is the binary RR-LOCO coding scheme. The constrained code we apply is a binary LOCO code devised according to the aforementioned general method. We start by defining the proposed binary LOCO code.

Definition 1. *A binary LOCO code $\mathcal{RC}_m^2$, where $m \geq 1$, that forbids the patterns in $\mathcal{R}^2 = \{000, 010\}$ is defined by the following properties:*

1) *Codewords in $\mathcal{RC}_m^2$ are defined over GF$(2) = \{0, 1\}$ and are of length m bits.*
2) *Codewords in $\mathcal{RC}_m^2$ are ordered lexicographically.*
3) *Codewords in $\mathcal{RC}_m^2$ do not have patterns in $\mathcal{R}^2$.*
4) *All codewords satisfying 1)–3) are included.*

GF(q) denotes Galois field of order q (the alphabet size). Lexicographic ordering here is ordering codewords ascendingly according to the rule "$0 < 1$", where bit significance reduces from left to right [3], [19]. Table I shows the codebooks of $\mathcal{RC}_m^2$ for all m in $\{1, 2, \ldots, 5\}$. The first step to devise the encoder-decoder of this binary LOCO code is to specify the group structure. Codewords in $\mathcal{RC}_m^2$, $m \geq 2$, can be partitioned into the following groups:

- Group 1: Codewords starting with 0011 from the left.
- Group 2: Codewords starting with 011 from the left.
- Group 3: Codewords starting with 1 from the left.

The second step is to enumerate the codewords in $\mathcal{RC}_m^2$, which is done by Theorem 1. Let $N_2(m) \triangleq |\mathcal{RC}_m^2|$.

Theorem 1. *The cardinality of a binary LOCO code $\mathcal{RC}_m^2$ is given by the recursive formula:*

$$N_2(m) = N_2(m-1) + N_2(m-3) + N_2(m-4), \ m \geq 2, \quad (5)$$

where the defined cardinalities are:

$$N_2(-3) \triangleq 0, \ N_2(-2) = N_2(-1) = N_2(0) \triangleq 1, \\ \text{and } N_2(1) = 2. \quad (6)$$

Proof: We compute the cardinalities of each group then add them all. Let the cardinality of Group i be $N_{2,i}$. As for Group 3 in $\mathcal{RC}_m^2$, there is a bijection between its codewords and the codewords in $\mathcal{RC}_{m-1}^2$ (attach 1 from the left). Thus,

$$N_{2,3}(m) = N_2(m-1). \quad (7)$$

As for Group 2 in $\mathcal{RC}_m^2$, there is a bijection between its codewords and the codewords starting with 1 from the left in $\mathcal{RC}_{m-2}^2$ (attach 01 from the left). Thus using (7),

$$N_{2,2}(m) = N_{2,3}(m-2) = N_2(m-3). \quad (8)$$

As for Group 1 in $\mathcal{RC}_m^2$, there is a bijection between its codewords and the codewords starting with 1 from the left in $\mathcal{RC}_{m-3}^2$ (attach 001 from the left). Thus using (7),

$$N_{2,1}(m) = N_{2,3}(m-3) = N_2(m-4). \quad (9)$$

Adding (7), (8), and (9) gives (5). The defined cardinalities (used for computing $N_2(m)$ for $2 \leq m \leq 4$), other than $N_2(1)$, can be computed by observing that $N_2(1) = 2$, $N_2(2) = 4$, $N_2(3) = 6$, and $N_2(4) = 9$, which sets up four equations. This observation is immediate given the forbidden patterns. ∎

Define a codeword $\mathbf{c}$ in $\mathcal{RC}_m^2$ as $\mathbf{c} \triangleq c_{m-1}c_{m-2}\ldots c_0$, with $c_i \triangleq \zeta$ for $i \geq m$, where ζ represents "out of codeword bounds". This ζ is defined to characterize, only for completeness, the symbols preceding c_{m-1}, which practically do not exist. The integer equivalent of a LOCO codeword bit c_i, $0 \leq i \leq m-1$, is a_i, i.e., a_i is 0 (1) when c_i is 0 (1). Denote the lexicographic index of a codeword $\mathbf{c}$ among all codewords in the LOCO code $\mathcal{RC}_m^2$ by $g_2(m, \mathbf{c})$, which is abbreviated to $g(\mathbf{c})$. In general, $g(\mathbf{c})$ is in $\{0, 1, \ldots, N_2(m) - 1\}$.

Before introducing the third step, we briefly introduce a general definition for the terms "special case" and "typical case". Consider a general LOCO code $\mathcal{C}_m^q$, where the alphabet is GF(q) and the length is m.

Definition 2. *A special case of existence for symbol c_i of codeword $\mathbf{c}$ in $\mathcal{C}_m^q$ is a case where not all codewords starting with any $c_i' < c_i$, according to the lexicographic ordering, from the left in $\mathcal{C}_{i+1}^q$ are allowed to be concatenated from the right to the symbols preceding c_i.*

The typical case of existence for c_i is the case where all codewords starting with any $c_i' < c_i$ from the left in $\mathcal{C}_{i+1}^q$ are allowed to be concatenated from the right to the symbols preceding c_i.

Intuitively, the special cases are the ones implying "jumps" among the codewords of length $i+1$, while the typical case is the one implying "no jumps" among the codewords of length $i+1$. "Typicality" of the typical case follows from the lack of these jumps.

The third step is to specify the special cases of occurence for a 1 inside a codeword in $\mathcal{RC}_m^2$. These cases are:

- Case 2: $c_{i+2}c_{i+1}c_i = 001$.
- Case 3: $c_{i+2}c_{i+1}c_i = 011$.
- Case 4: $c_{i+2}c_{i+1}c_i = 101$ or $c_{i+2}c_{i+1}c_i = \zeta 01$.

The typical or default case, Case 1, is simply the case of "otherwise". In particular, it is the case that $c_{i+2}c_{i+1}c_i = 111$, $c_{i+2}c_{i+1}c_i = \zeta 11$, or $c_{i+1}c_i = \zeta 1$.

The fourth and fifth steps are to find the encoding-decoding rule, which specifies the mapping from index to codeword and vice versa. This rule for $\mathcal{RC}_m^2$ is given in Theorem 2.

TABLE I
ALL THE CODEWORDS OF FIVE BINARY (RR) LOCO CODES, $\mathcal{RC}^2_m$, $m \in \{1, 2, \ldots, 5\}$. THE THREE DIFFERENT GROUPS OF CODEWORDS ARE EXPLICITLY ILLUSTRATED FOR THE CODE $\mathcal{RC}^2_5$.

Codeword index $g(\mathbf{c})$	Codewords of the code $\mathcal{RC}^2_m$					
	$m=1$	$m=2$	$m=3$	$m=4$	$m=5$	
0	0	00	001	0011	00110	Group 1
1	1	01	011	0110	00111	
2		10	100	0111	01100	Group 2
3		11	101	1001	01101	
4			110	1011	01110	
5			111	1100	01111	
6				1101	10011	Group 3
7				1110	10110	
8				1111	10111	
9					11001	
10					11011	
11					11100	
12					11101	
13					11110	
14					11111	
Code cardinality	$N_2(1)=2$	$N_2(2)=4$	$N_2(3)=6$	$N_2(4)=9$	$N_2(5)=15$	

Theorem 2. *The relation between the lexicographic index $g(\mathbf{c})$, $\mathbf{c} \in \mathcal{RC}^2_m$, and the binary codeword $\mathbf{c}$ itself is given by:*

$$g(\mathbf{c}) = \sum_{i=0}^{m-1} a_i \Big[(1-y_{i,1}) N_2(i-2) + (1-y_{i,1}-y_{i,2}) N_2(i-3) \Big], \tag{10}$$

where $y_{i,1}$ and $y_{i,2}$ are specified as follows:

$$\begin{aligned} &y_{i,1} = 1 \text{ if } c_{i+2}c_{i+1}c_i \in \{001, 011\}, \text{ and } y_{i,1} = 0 \text{ otherwise}, \\ &y_{i,2} = 1 \text{ if } c_{i+1}c_i = 01 \text{ s.t. } y_{i,1} = 0, \text{ and } y_{i,2} = 0 \text{ otherwise}. \end{aligned} \tag{11}$$

Proof: We compute the contributions $g_{i,j}(c_i)$ of a bit c_i under Case j, for all $j \in \{1, 2, 3, 4\}$, in a binary LOCO codeword then merge them all. As for the typical case, which we index by 1, this contribution is the number of codewords starting with 0 from the left in $\mathcal{RC}^2_{i+1}$. Thus using (8) and (9),

$$\begin{aligned} g_{i,1}(c_i) &= N_{2,2}(i+1) + N_{2,1}(i+1) \\ &= N_2(i-2) + N_2(i-3). \end{aligned} \tag{12}$$

As for Case 2 (Case 3), this contribution is the number of codewords starting with 000 (010) from the left in $\mathcal{RC}^2_{i+3}$. Note that 000 and 010 are forbidden patterns. Thus,

$$\begin{aligned} &g_{i,2}(c_i) = 0 \text{ and} \\ &g_{i,3}(c_i) = 0. \end{aligned} \tag{13}$$

As for Case 4, this contribution is the number of codewords starting with 00 from the left in $\mathcal{RC}^2_{i+2}$. Thus using (9),

$$g_{i,4}(c_i) = N_{2,1}(i+2) = N_2(i-2). \tag{14}$$

Using $y_{i,1}$ (for Cases 2 and 3) and $y_{i,2}$ (for Case 4) from (11) along with a_i to merge (12), (13), and (14) gives:

$$g_i(c_i) = a_i \Big[(1-y_{i,1}) N_2(i-2) + (1-y_{i,1}-y_{i,2}) N_2(i-3) \Big]. \tag{15}$$

Substituting (15) in $g(\mathbf{c}) = \sum_{i=0}^{m-1} g_i(c_i)$ gives (10). ∎

For brevity, we skip the sixth step, which is to assemble the encoding and decoding algorithms. These algorithms are a direct consequence of the rule in (10), and we refer the reader to [3], [19], [24], and [28] for details. Note that we sometimes refer to $\mathcal{RC}^2_m$ as a *1D binary RR-LOCO code*. The encoding-decoding rule of a LOCO code is the reason behind its low complexity algorithms, where reconfiguration becomes as easy as reprogramming an adder [8], [24].

Example 3. *Consider the binary LOCO code $\mathcal{RC}^2_5$ ($m = 5$). Using (5) and (6), we get $N_2(-3) \triangleq 0$, $N_2(-2) = N_2(-1) = N_2(0) \triangleq 1$, $N_2(1) = 2$, $N_2(2) = 4$, $N_2(3) = 6$, and $N_2(4) = 9$. Consider the codeword $\mathbf{c} = c_4c_3c_2c_1c_0 = 11011$ in $\mathcal{RC}^2_5$. The case indexed by $i_c = 1$ (the typical case) applies to c_4 and c_3, which means $y_{4,1} = y_{4,2} = y_{3,1} = y_{3,2} = 0$. The case indexed by $i_c = 4$ applies to c_1, which means $y_{1,1} = 0$ and $y_{1,2} = 1$. The case indexed by $i_c = 3$ applies to c_0, which means $y_{0,1} = 1$ and $y_{0,2} = 0$. Consequently, and using (10), we get:*

$$\begin{aligned} g(\mathbf{c} = 11011) &= [N_2(2) + N_2(1)] + [N_2(1) + N_2(0)] \\ &\quad + [N_2(-1)] + [0] \\ &= [4+2] + [2+1] + [1] = 10, \end{aligned}$$

which is consistent with the codeword index in Table I. In practice, only $8 = 2^3$ codewords will be used from $\mathcal{RC}^2_5$.

Remark 1. *If the coded bits are complemented before writing to pages, the set of forbidden patterns on the left-most pages becomes $\{101, 111\}$ instead, which appears in [16] as well. In this case, the cardinality of the binary LOCO code remains as in (5), while the encoding-decoding rule becomes exactly that of a binary asymmetric LOCO code in [23] for $x = 1$:*

$$g(\mathbf{c}) = \sum_{i=0}^{m-1} a_i N_2(i - a_{i+1}). \tag{16}$$

Encoding and decoding on the left-most pages are just subtractions and additions. As for the remaining pages, data is written and read directly (uncoded). This guarantees simplicity and maintains high access speed via our 1D binary RR-LOCO coding scheme.

IV. RR-LOCO Coding Over GF(4)

In this section, we propose a 1D RR coding scheme over GF(4), which is also based on LOCO codes. This scheme is our 4-ary RR-LOCO coding scheme. The goal is to limit the rate loss resulting from binary RR coding schemes via coding on the two left-most pages. Finer classification of error-prone patterns, stemming from characterizing them via two bits instead of one, results in allowing some benign or less detrimental patterns, and therefore increasing the rate with negligible effect on performance.

We start by modifying the set of error-prone patterns. Let

$$\begin{aligned}\theta_1, \overline{\theta}_1 \in \mathcal{W}_0 &\triangleq \left\{\frac{3q}{4}, \frac{3q}{4}+1, \ldots, q-1\right\},\\ \theta_2, \overline{\theta}_2 \in \mathcal{W}_1 &\triangleq \left\{\frac{q}{2}, \frac{q}{2}+1, \ldots, \frac{3q}{4}-1\right\},\\ \theta_3 \in \mathcal{W}_2 \cup \mathcal{W}_3,\ \mathcal{W}_2 &\triangleq \left\{\frac{q}{4}, \frac{q}{4}+1, \ldots, \frac{q}{2}-1\right\},\\ \mathcal{W}_3 &\triangleq \left\{0, 1, \ldots, \frac{q}{4}-1\right\}, \end{aligned} \tag{17}$$

where q is the number of levels per Flash cell (a positive power of 2). While mathematically $q \geq 4$, we focus here on the case of $q \geq 8$. Then, the set of interest is the set resulting in the high-low-high level patterns in $\mathcal{L}'_q \subset \mathcal{L}_q$:

$$\begin{aligned}\mathcal{L}'_q \triangleq\ & \{\theta_1 \eta \overline{\theta}_1, \forall \theta_1, \overline{\theta}_1 \mid 0 \leq \eta < \min(\theta_1, \overline{\theta}_1)\}\\ & \cup\ \{\theta_1\theta_3\theta_2, \forall \theta_1, \theta_2, \theta_3\} \cup\\ & \{\theta_2\theta_3\theta_1, \forall \theta_1, \theta_2, \theta_3\}\ \cup\ \{\theta_2\theta_3\overline{\theta}_2, \forall \theta_2, \overline{\theta}_2, \theta_3\}. \end{aligned} \tag{18}$$

This set also subsumes all 3-tuple forbidden patterns adopted in the literature for Flash. The only difference between the set $\mathcal{L}'_q$ and the set $\mathcal{L}_q$ is that in the former, if either the left level is or the right level is or both levels are in $\mathcal{W}_1$, the middle level is always in $\mathcal{W}_2 \cup \mathcal{W}_3$. Our experimental results show that the level patterns in $\mathcal{L}_q \setminus \mathcal{L}'_q$ have very limited contribution to the errors occurring upon reading from the Flash device.

Example 4. *Consider a TLC Flash device, i.e., $q = 8$. In this case, we have $\theta_1, \overline{\theta}_1 \in \{6, 7\}$, $\theta_2, \overline{\theta}_2 \in \{4, 5\}$, and $\theta_3 \in \{0, 1, 2, 3\}$. Then, the difference between the two sets of interest is only one level pattern:*

$$\mathcal{L}_8 \setminus \mathcal{L}'_8 = \{545, 546, 547, 645, 745\}. \tag{19}$$

For mapping from charge levels to binary bits, we adopt the RAGM of Algorithm 1. Moreover, we index the Flash pages the same way the bits in each sequence in the array map are indexed using Algorithm 1. Therefore, we are interested here in the data on the two left-most pages indexed by $p-1$ and $p-2$. We adopt the following binary to 4-ary mapping-demapping, where GF(4) $= \{0, 1, \alpha, \alpha^2\}$, for these two specific Flash pages:

$$\begin{aligned} 11 &\longleftrightarrow 0\ (\mathcal{W}_3), & 10 &\longleftrightarrow 1\ (\mathcal{W}_2),\\ 00 &\longleftrightarrow \alpha\ (\mathcal{W}_1), & 01 &\longleftrightarrow \alpha^2\ (\mathcal{W}_0). \end{aligned} \tag{20}$$

The set of level patterns corresponding to each GF(4) symbol is given between parenthesis.

We can see from (17), (18), and (20) that the set of level patterns in $\mathcal{L}'_q$ can be forbidden in the wordline or the bitline direction by forbidding the 4-ary patterns in the following set $\mathcal{R}^4$ from being written on the two left-most pages indexed by $p-1$ and $p-2$:

$$\begin{aligned}\mathcal{R}^4 = \{&\alpha 0 \alpha, \alpha 1 \alpha, \alpha 0 \alpha^2, \alpha 1 \alpha^2, \alpha^2 0 \alpha, \alpha^2 1 \alpha,\\ &\alpha^2 0 \alpha^2, \alpha^2 1 \alpha^2, \alpha^2 \alpha \alpha^2, \alpha^2 \alpha^2 \alpha^2\}. \end{aligned} \tag{21}$$

Once again, no coding on any other page is needed. Data will therefore be read from each page independently, except the two left-most pages, and immediately passed to the low-density parity-check (LDPC) decoder to start its processing. This idea is the key idea of our 4-ary RR constrained coding scheme.

Consider a TLC Flash device ($q = 8$) once again. Forbidding the patterns in $\mathcal{R}^4$ on the two left-most pages instead of the patterns in $\mathcal{R}^2$ on the left-most page results in allowing many benign patterns that are forbidden if binary RR coding is adopted, e.g., 444, 474, and 555.

Now, we introduce our 4-ary RR coding scheme that forbids the patterns in $\mathcal{R}^4$ on the two left-most pages in either the wordline direction or the bitline direction, while leaving all other pages with no coding. The constrained code we apply is a 4-ary LOCO code devised according to the general method in [24]. We start by defining the proposed 4-ary LOCO code.

Definition 3. *A 4-ary LOCO code $\mathcal{RC}^4_m$, where $m \geq 1$, that forbids the patterns in $\mathcal{R}^4$ is defined by the following properties:*

1) *Codewords in $\mathcal{RC}^4_m$ are defined over GF(4) $= \{0, 1, \alpha, \alpha^2\}$ and are of length m symbols.*
2) *Codewords in $\mathcal{RC}^4_m$ are ordered lexicographically.*
3) *Codewords in $\mathcal{RC}^4_m$ do not have patterns in $\mathcal{R}^4$.*
4) *All codewords satisfying 1)–3) are included.*

Lexicographic ordering here is ordering codewords ascendingly according to the rule "$0 < 1 < \alpha < \alpha^2$", where symbol significance reduces from left to right [3], [19]. The first step to devise the encoder-decoder of this 4-ary LOCO code is to specify the group structure. Let γ_1 and γ_2 be in $\{0, 1\}$. Codewords in $\mathcal{RC}^4_m$, $m \geq 3$, can be partitioned into the following groups:

- Group 1: Codewords starting with γ_1, $\forall \gamma_1$, from the left.
- Group 2: Codewords starting with $\alpha\gamma_1\gamma_2$, $\forall \gamma_1, \gamma_2$, from the left.
- Group 3: Codewords starting with $\alpha\alpha$ or $\alpha\alpha^2$ from the left.
- Group 4: Codewords starting with $\alpha^2\gamma_1\gamma_2$, $\forall \gamma_1, \gamma_2$, from the left.
- Group 5: Codewords starting with $\alpha^2\alpha\gamma_1\gamma_2$, $\forall \gamma_1, \gamma_2$, from the left.
- Group 6: Codewords starting with $\alpha^2\alpha\alpha$ from the left.
- Group 7: Codewords starting with $\alpha^2\alpha^2\gamma_1\gamma_2$, $\forall \gamma_1, \gamma_2$, from the left.
- Group 8: Codewords starting with $\alpha^2\alpha^2\alpha$ from the left.

The second step is to enumerate the codewords in $\mathcal{RC}^4_m$, which is done by Theorem 3. Let $N_4(m) \triangleq |\mathcal{RC}^4_m|$.

Theorem 3. *The cardinality of a 4-ary LOCO code $\mathcal{RC}^4_m$ is given by the recursive formula:*

$$N_4(m) = 3N_4(m-1) - 2N_4(m-2) + 9N_4(m-3) + 7N_4(m-4) + 6N_4(m-5) + 4N_4(m-6),\ m \geq 3, \quad (22)$$

where the defined cardinalities are:

$$N_4(-5) \triangleq \frac{1}{32},\ N_4(-4) \triangleq -\frac{1}{16},\ N_4(-3) \triangleq 0,$$
$$N_4(-2) \triangleq \frac{1}{4},\ N_4(-1) \triangleq \frac{1}{2},\ N_4(0) \triangleq 1,$$
$$\text{and } N_4(1) = 4,\ N_4(2) = 16. \quad (23)$$

Proof: We compute the cardinalities of each group then add them all. Let the cardinality of Group i be $N_{4,i}$. As for Group 1 in $\mathcal{RC}_m^4$, there is a surjection between its codewords and the codewords in $\mathcal{RC}_{m-1}^4$ (attach 0 or 1 from the left). Thus,

$$N_{4,1}(m) = 2N_4(m-1). \quad (24)$$

As for Group 2 in $\mathcal{RC}_m^4$, there is a surjection between its codewords and the codewords in $\mathcal{RC}_{m-3}^4$. Thus,

$$N_{4,2}(m) = (2)(2)N_4(m-3) = 4N_4(m-3). \quad (25)$$

As for Group 3 in $\mathcal{RC}_m^4$, there is a bijection between its codewords and the codewords starting with α or α^2 from the left in $\mathcal{RC}_{m-1}^4$. Thus using (24),

$$N_{4,3}(m) = N_4(m-1) - N_{4,1}(m-1) = N_4(m-1) - 2N_4(m-2). \quad (26)$$

As for Group 4 in $\mathcal{RC}_m^4$, the cardinality is the same as that of Group 2. Thus,

$$N_{4,4}(m) = (2)(2)N_4(m-3) = 4N_4(m-3). \quad (27)$$

As for Group 5 in $\mathcal{RC}_m^4$, it is handled in a way similar to that of Groups 2 and 4. Thus,

$$N_{4,5}(m) = (2)(2)N_4(m-4) = 4N_4(m-4). \quad (28)$$

As for Group 6 in $\mathcal{RC}_m^4$, there is a bijection between its codewords and the codewords starting with α from the left in $\mathcal{RC}_{m-2}^4$. Thus using (25) and (26),

$$N_{4,6}(m) = N_{4,2}(m-2) + N_{4,3}(m-2) = N_4(m-3) - 2N_4(m-4) + 4N_4(m-5). \quad (29)$$

As for Group 7 in $\mathcal{RC}_m^4$, the cardinality is the same as that of Group 5. Thus,

$$N_{4,7}(m) = (2)(2)N_4(m-4) = 4N_4(m-4). \quad (30)$$

As for Group 8 in $\mathcal{RC}_m^4$, there is a bijection between its codewords and the codewords starting with $\alpha^2\alpha$ from the left in $\mathcal{RC}_{m-1}^4$. Thus using (28) and (29),

$$N_{4,8}(m) = N_{4,5}(m-1) + N_{4,6}(m-1) = N_4(m-4) + 2N_4(m-5) + 4N_4(m-6). \quad (31)$$

Adding (24), (25), (26), (27), (28), (29), (30), and (31) gives (22). The defined cardinalities (used for computing $N_4(m)$ for $3 \leq m \leq 6$), other than $N_4(1)$ and $N_4(2)$, can be computed from the cardinalities at small values of m, which set up six equations. ∎

Define a codeword $\mathbf{c}$ in $\mathcal{RC}_m^4$ as $\mathbf{c} \triangleq c_{m-1}c_{m-2}\ldots c_0$, with $c_i \triangleq \zeta$ for $i \geq m$, where ζ represents "out of codeword bounds". The integer equivalent of a LOCO codeword symbol c_i, $0 \leq i \leq m-1$, is a_i, i.e., a_i is 0, 1, 2, or 3 when c_i is 0, 1, α, or α^2, respectively. Denote the lexicographic index of a codeword $\mathbf{c}$ among all codewords in the LOCO code $\mathcal{RC}_m^4$ by $g_4(m, \mathbf{c})$, which is abbreviated to $g(\mathbf{c})$. In general, $g(\mathbf{c})$ is in $\{0, 1, \ldots, N_4(m)-1\}$.

Recall Definition 2 of special and typical cases. The third step is to specify the typical/special cases of occurence for a symbol in GF(4) $\setminus \{0\}$ inside a codeword in $\mathcal{RC}_m^4$. Let γ be in $\{\zeta, 0, 1\}$ and χ be in $\{\alpha, \alpha^2\}$. These cases are:

- Case 1.a: $c_{i+1}c_i = \gamma 1$ or $c_{i+1}c_i = \gamma\alpha$, for all γ.
- Case 1.b: $c_{i+1}c_i = \gamma\alpha^2$, for all γ.
- Case 2: $c_{i+1}c_i = \chi 1$ or $c_{i+1}c_i = \chi\alpha$, for all χ.
- Case 3: $c_{i+1}c_i = \alpha\alpha^2$.
- Case 4: $c_{i+1}c_i = \alpha^2\alpha^2$.

The typical or default case is Case 1 (Case 1.a and Case 1.b combined).

The fourth and fifth steps are to find the encoding-decoding rule, which specifies the mapping from index to codeword and vice versa. This rule for $\mathcal{RC}_m^4$ is given in Theorem 4.

Theorem 4. *The relation between the lexicographic index $g(\mathbf{c})$, $\mathbf{c} \in \mathcal{RC}_m^4$, and the 4-ary codeword $\mathbf{c}$ itself is given by:*

$$g(\mathbf{c}) = \sum_{i=0}^{m-1} \Big[\big[(y_{i,1} + y'_{i,1})a_i + y_{i,3}\big] N_4(i) + \big[2(y_{i,2}a_i + y_{i,3} - y'_{i,1}) + 5y_{i,\mathrm{d}}\big] N_4(i-1) + \big[4(y'_{i,1} + y_{i,3}) + 2y_{i,\mathrm{d}}\big] N_4(i-2) + 4y_{i,\mathrm{d}} N_4(i-3) \Big], \quad (32)$$

where $y_{i,1}$, $y'_{i,1}$, $y_{i,2}$, $y_{i,3}$, and $y_{i,\mathrm{d}}$ are specified as follows:

$$y_{i,1} = 1 \text{ if } c_{i+1}c_i \in \{\gamma 1, \gamma\alpha \mid \forall\gamma\},\ \text{and } y_{i,1} = 0 \text{ otherwise},$$
$$y'_{i,1} = 1 \text{ if } c_{i+1}c_i \in \{\gamma\alpha^2 \mid \forall\gamma\},\ \text{and } y'_{i,1} = 0 \text{ otherwise},$$
$$y_{i,2} = 1 \text{ if } c_{i+1}c_i \in \{\chi 1, \chi\alpha \mid \forall\chi\},\ \text{and } y_{i,2} = 0 \text{ otherwise},$$
$$y_{i,3} = 1 \text{ if } c_{i+1}c_i = \alpha\alpha^2,\ \text{and } y_{i,3} = 0 \text{ otherwise},$$
$$y_{i,\mathrm{d}} = 1 \text{ if } c_{i+1}c_i = \alpha^2\alpha^2,\ \text{and } y_{i,\mathrm{d}} = 0 \text{ otherwise}. \quad (33)$$

Proof: We compute the contributions $g_{i,j}(c_i)$ of a symbol c_i under Case j, for all j in $\{1, 2, 3, 4\}$, in a 4-ary LOCO codeword then merge them all. As for the typical case, Situation a, which we index by $1.a$, this contribution is the number of codewords starting with $c'_i < c_i$, where $c_i \in \{1, \alpha\}$, from the left in $\mathcal{RC}_{i+1}^4$. Thus using (24),

$$g_{i,1.a}(c_i) = a_i N_4(i+1-1) = a_i N_4(i). \quad (34)$$

As for the typical case, Situation b, which we index by $1.b$, this contribution is the number of codewords starting with $c'_i < c_i$, where $c_i = \alpha^2$, from the left in $\mathcal{RC}_{i+1}^4$. Thus using (24), (25), and (26),

$$g_{i,1.b}(c_i) = N_{4,1}(i+1) + N_{4,2}(i+1) + N_{4,3}(i+1) = 3N_4(i) - 2N_4(i-1) + 4N_4(i-2). \quad (35)$$

As for Case 2, this contribution is the number of codewords starting with $c_i'\gamma_1$, $c_i' < c_i$, where $c_i \in \{1, \alpha\}$ and $\gamma_1 \in \{0, 1\}$, from the left in $\mathcal{RC}_{i+1}^4$. Thus using (24),

$$g_{i,2}(c_i) = a_i N_{4,1}(i) = 2a_i N_4(i-1). \tag{36}$$

As for Case 3, this contribution is the number of codewords starting with $\alpha c_i'$, $c_i' < c_i$, where $c_i = \alpha^2$, from the left in $\mathcal{RC}_{i+2}^4$. Those are all the codewords starting with $\gamma_1\gamma_2$, for all γ_1 and γ_2, from the left in $\mathcal{RC}_{i+1}^4$ plus all the codewords starting with α from the left in $\mathcal{RC}_{i+1}^4$. Thus using (24), (25), and (26),

$$\begin{aligned} g_{i,3}(c_i) &= 2N_{4,1}(i) + N_{4,2}(i+1) + N_{4,3}(i+1) \\ &= N_4(i) + 2N_4(i-1) + 4N_4(i-2). \end{aligned} \tag{37}$$

As for Case 4, this contribution is the number of codewords starting with $\alpha^2 c_i'$, $c_i' < c_i$, where $c_i = \alpha^2$, from the left in $\mathcal{RC}_{i+2}^4$. Those are all the codewords starting with $\gamma_1\gamma_2$, for all γ_1 and γ_2, from the left in $\mathcal{RC}_{i+1}^4$ plus all the codewords starting with $\alpha^2\alpha$ from the left in $\mathcal{RC}_{i+2}^4$. Thus using (24), (28), and (29),

$$\begin{aligned} g_{i,4}(c_i) &= 2N_{4,1}(i) + N_{4,5}(i+2) + N_{4,6}(i+2) \\ &= 5N_4(i-1) + 2N_4(i-2) + 4N_4(i-3). \end{aligned} \tag{38}$$

We use $y_{i,1}$, $y_{i,1}'$ (for Case 1), $y_{i,2}$ (for Case 2), $y_{i,3}$ (for Case 3), and $y_{i,\mathrm{d}}$ (for Case 4) from (33) along with a_i to merge (34), (35), (36), (37), and (38). We adopt the following merging functions, where $f_\ell^{\mathrm{mer}}(\cdot)$ is associated with $N_4(i+1-\ell)$:

$$\begin{aligned} f_1^{\mathrm{mer}}(\cdot) &= (y_{i,1} + y_{i,1}')a_i + y_{i,3}, \\ f_2^{\mathrm{mer}}(\cdot) &= 2(y_{i,2}a_i + y_{i,3} - y_{i,1}') + 5y_{i,\mathrm{d}}, \\ f_3^{\mathrm{mer}}(\cdot) &= 4(y_{i,1}' + y_{i,3}) + 2y_{i,\mathrm{d}}, \\ f_4^{\mathrm{mer}}(\cdot) &= 4y_{i,\mathrm{d}}. \end{aligned} \tag{39}$$

Therefore, the general form of the symbol contribution $g_i(c_i)$ is:

$$g_i(c_i) = \sum_{\ell=1}^{4} f_\ell^{\mathrm{mer}}(\cdot) N_4(i+1-\ell). \tag{40}$$

Substituting (39) and (40) in $g(\mathbf{c}) = \sum_{i=0}^{m-1} g_i(c_i)$ gives (32). ∎

Remark 2. *Observe that the number of linearly independent merging variables is always less than the number of final cases [24]. Here, $y_{i,\mathrm{d}}$ is dependent on the other merging variables as it can be written as $y_{i,\mathrm{d}} = \mathbb{1}(a_i)(1 - y_{i,1} - y_{i,1}' - y_{i,2} - y_{i,3})$, where $\mathbb{1}(a_i) = 1$ if $a_i > 0$ and $\mathbb{1}(a_i) = 0$ if $a_i = 0$.*

For brevity, we again skip the sixth step, which is to assemble the encoding and decoding algorithms. These algorithms are a direct consequence of the rule in (32), and we refer the reader to [3], [19], [24], and [28] for details. Note that we sometimes refer to $\mathcal{RC}_m^4$ as a *1D 4-ary RR-LOCO code*.

V. Rate, Complexity, and Error Propagation

We start by calculating asymptotic rates. Unfortunately, deriving the capacity for 2D constrained codes is known to be notoriously hard. Therefore, we will derive the capacity $C_{\mathcal{L}_q}^{\mathrm{1D}}$ only under the 1D constrained coding setup, which is already higher than the capacity under the 2D setup. Thus, $C_{\mathcal{L}_q}^{\mathrm{1D}}$ serves as a ceiling for the highest achievable rate in a device where patterns in $\mathcal{L}_q$ are forbidden at least in one direction. We will shortly show that 1D constrained coding suffices in terms of performance.

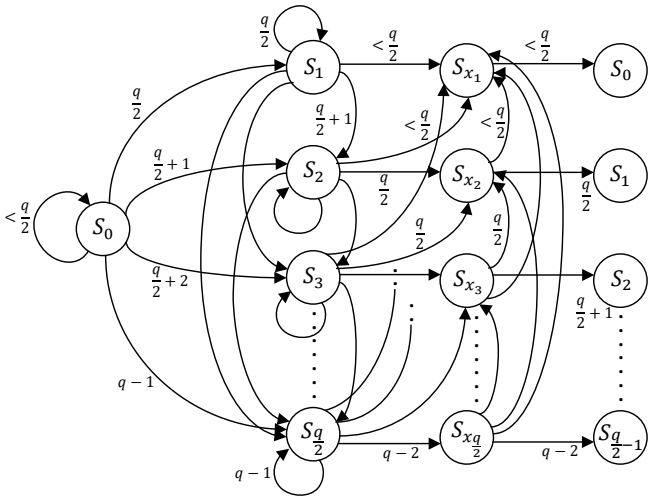

Fig. 2. An FSTD of a 1D constrained sequence forbidding level patterns in $\mathcal{L}_q$, for any q. Here, we operate directly on level patterns for simplicity. The same state could represent multiple 2-tuples, depending on the previous state. Illustrative example: To arrive at S_1, we can a) receive two consecutive levels $< \frac{q}{2}$ then $\frac{q}{2}$ to transition to S_1 via S_0, b) receive two consecutive levels $\frac{q}{2}$ then $\frac{q}{2}$ to self-transition to S_1, or c) receive two consecutive levels $\frac{q}{2}$ then $\frac{q}{2}$ to transition to S_1 via S_{x_2}.

A finite-state transition diagram (FSTD) of a sequence where level patterns in $\mathcal{L}_q$ are forbidden is shown in Fig. 2. This FSTD is designed to have a reduced number of states. Based on this FSTD, the general adjacency matrix is (vectors are row vectors):

$$\mathbf{A}_1 = \left[\begin{array}{c|c|c|c} \frac{q}{2} & \mathbf{1}_{\frac{q}{2}} & 0 & \mathbf{0}_{\frac{q}{2}-1} \\ \hline \mathbf{0}_{\frac{q}{2}}^{\mathrm{T}} & \mathbf{U}_{\frac{q}{2}}^1 & \frac{q}{2}\mathbf{1}_{\frac{q}{2}}^{\mathrm{T}} & \begin{array}{c} \mathbf{0}_{\frac{q}{2}-1} \\ \hline \mathbf{L}_{\frac{q}{2}-1}^1 \end{array} \\ \hline \frac{q}{2} & \mathbf{0}_{\frac{q}{2}} & 0 & \mathbf{0}_{\frac{q}{2}-1} \\ \hline \mathbf{0}_{\frac{q}{2}-1}^{\mathrm{T}} & \mathbf{I}_{\frac{q}{2}-1} \mid \mathbf{0}_{\frac{q}{2}-1}^{\mathrm{T}} & \frac{q}{2}\mathbf{1}_{\frac{q}{2}-1}^{\mathrm{T}} & \begin{array}{c|c} \mathbf{0}_{\frac{q}{2}-1} & \\ \hline \mathbf{L}_{\frac{q}{2}-2}^1 & \mathbf{0}_{\frac{q}{2}-2}^{\mathrm{T}} \end{array} \end{array}\right], \tag{41}$$

where $\mathbf{U}_\delta^1$ ($\mathbf{L}_\delta^1$) is an upper (lower) only-ones triangular matrix of size $\delta \times \delta$. Thus and from [2], the normalized capacity of a 1D constrained code forbidding the level patterns in $\mathcal{L}_q$ is:

$$C_{\mathcal{L}_q}^{\mathrm{1D}} = \frac{\log_2(\lambda_{\max}(\mathbf{A}_1))}{\log_2 q}, \tag{42}$$

where $\lambda_{\max}(\mathbf{A})$ is the maximum real positive eigenvalue of the matrix $\mathbf{A}$.[5]

The capacity of a 2D binary code preventing $\{000, 010\}$ is the capacity of a 2D $(0, 1)$ RLL code, which is ≈ 0.5879 [30]. Thus, the normalized capacity of our 2D RR coding scheme is:

[5]For positive integers $a + b \leq q$, the set H of the a largest levels, and the set L of the b smallest levels in $\{0, 1, \ldots, q-1\}$, a formula for the (count-constrained) capacity of the constrained system forbidding all level patterns in $\{\beta_1\beta_2\overline{\beta}_1 \mid \beta_1, \overline{\beta}_1 \in H, \beta_2 \in L\}$ was derived in [29].

TABLE II
CAPACITY COMPARISON BETWEEN $C^{\text{1D}}_{\mathcal{L}_q}$, 1D BINARY RR CAPACITY $C^{\text{1D}}_{\text{RR2}}$, AND 1D 4-ARY RR CAPACITY $C^{\text{1D}}_{\text{RR4}}$

q	$C^{\text{1D}}_{\mathcal{L}_q}$	$C^{\text{1D}}_{\text{RR2}}$	Capacity gap %	$C^{\text{1D}}_{\text{RR4}}$
4	0.8941	0.8471	5.257%	0.8859
8	0.9235	0.8981	2.750%	0.9239
16	0.9401	0.9235	1.766%	0.9429
32	0.9509	0.9388	1.272%	0.9544

$$C^{\text{2D}}_{\text{RR2}} \approx \frac{0.5879 + \log_2 q - 1}{\log_2 q} = \frac{\log_2 q - 0.4121}{\log_2 q}. \quad (43)$$

As mentioned above, the 1D constrained system where patterns in $\mathcal{R}^2 = \{000, 010\}$ are forbidden can be interpreted as an interleaved RLL $(d, k) = (0, 1)$ constrained system, whose capacity is known to be $\log_2((1+\sqrt{5})/2) \approx 0.6942$. Thus, the normalized capacity of our 1D RR-LOCO coding scheme is:

$$C^{\text{1D}}_{\text{RR2}} = \frac{\log_2((1+\sqrt{5})/2) + \log_2 q - 1}{\log_2 q} \approx \frac{\log_2 q - 0.3058}{\log_2 q}. \quad (44)$$

The capacity gap between $C^{\text{1D}}_{\mathcal{L}_q}$ and $C^{\text{1D}}_{\text{RR2}}$ for different values of q is given in Table II. The table shows that the capacity gap is small, and it gets even smaller as q increases.

The capacity $C^{\text{1D}}_{\mathcal{L}'_q}$ of a 1D constrained system where the level patterns in $\mathcal{L}'_q$ are forbidden is slightly higher than $C^{\text{1D}}_{\mathcal{L}_q}$ since $\mathcal{L}'_q \subset \mathcal{L}_q$. We skip the derivation of $C^{\text{1D}}_{\mathcal{L}'_q}$ for brevity.

An FSTD of a 1D 4-ary constrained system where patterns in $\mathcal{R}^4$ are forbidden is given in Fig. 3. This FSTD is also designed to have a reduced number of states. The adjacency matrix is:

$$\mathbf{A}_2 = \begin{bmatrix} 2 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 1 & 1 \\ 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 1 & 0 \end{bmatrix}.$$

The characteristic polynomial is:

$$\det(x\mathbf{I} - \mathbf{A}_2) = x^6 - 3x^5 + 2x^4 - 9x^3 - 7x^2 - 6x - 4. \quad (45)$$

We can see that if x is replaced by $\lambda_{\text{c}} = \lambda_{\max}(\mathbf{A}_2)$, we get:

$$\lambda_{\text{c}}^m = 3\lambda_{\text{c}}^{m-1} - 2\lambda_{\text{c}}^{m-2} + 9\lambda_{\text{c}}^{m-3} + 7\lambda_{\text{c}}^{m-4} + 6\lambda_{\text{c}}^{m-5} + 4\lambda_{\text{c}}^{m-6}, \quad (46)$$

which is consistent with the cardinality recursion in (22). The capacity of this 4-ary constrained system is $\log_2(\lambda_{\max}(\mathbf{A}_2)) = \log_2(3.4147) = 1.7718$ bits/symbol. Thus, the normalized capacity of our 1D 4-ary RR-LOCO coding scheme is:

$$C^{\text{1D}}_{\text{RR4}} = \frac{1.7718 + \log_2 q - 2}{\log_2 q} \approx \frac{\log_2 q - 0.2282}{\log_2 q}. \quad (47)$$

Table II shows the capacity gain achieved by the 1D 4-ary RR scheme over the 1D binary RR schemes, and we will show that the performance, i.e., the Flash device protection, is nearly the same. An interesting observation is that for $q \in \{8, 16, 32\}$, the capacity of our 1D 4-ary RR scheme $C^{\text{1D}}_{\text{RR4}}$ is slightly higher than $C^{\text{1D}}_{\mathcal{L}_q}$. The reason is that the 1D 4-ary RR scheme is a capacity-approaching constrained coding scheme that forbids the level patterns in $\mathcal{L}'_q \subset \mathcal{L}_q$.

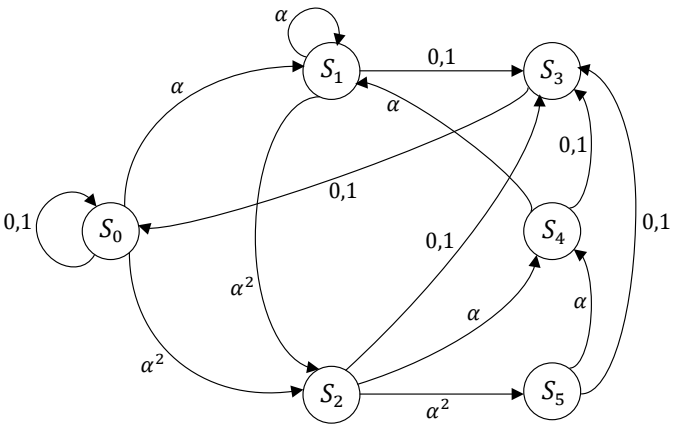

Fig. 3. An FSTD of a 1D 4-ary constrained sequence forbidding patterns in $\mathcal{R}^4$. The same state could represent multiple 2-tuples, depending on the previous state. Illustrative example: To arrive at S_1, we can a) receive two consecutive symbols 0 or 1 then α to transition to S_1 via S_0, b) receive two consecutive symbols α then α to self-transition to S_1, or c) receive two consecutive symbols α then α to transition to S_1 via S_4.

Next, we discuss the finite-length rates. First, the normalized rate of our 2D binary RR constrained coding scheme is:

$$R^{\text{2D}}_{\text{RR2}} = \frac{0.5 + \log_2 q - 1}{\log_2 q} = \frac{\log_2 q - 0.5}{\log_2 q} \quad (48)$$

since the rate of our left-most page coding is 0.5.

There are two differences between the capacity C and the finite-length rate R of a 1D RR-LOCO coding scheme. First, R is characterized by a specific length m of the LOCO code applied on the left-most (two left-most) page(s). Second, bridging bits or symbols are taken into account while computing R.

Regarding our 1D binary RR-LOCO coding scheme, we bridge with the pattern 11 between consecutive codewords in $\mathcal{RC}^2_m$ on the left-most page, and we remove the codeword $\mathbf{1}^m$ for self-clocking [19], [24]. Thus, the rate on the left-most page is $\lfloor \log_2(N_2(m) - 1) \rfloor / (m + 2)$, and the normalized rate of our 1D binary RR-LOCO coding scheme is:

$$R^{\text{1D}}_{\text{RR2}} = \frac{1}{\log_2 q} \left[\frac{\lfloor \log_2(N_2(m) - 1) \rfloor}{m + 2} + \log_2 q - 1 \right]. \quad (49)$$

1D binary RR-LOCO coding schemes are capacity-achieving schemes in the sense that the limit as $m \to \infty$ of $R^{\text{1D}}_{\text{RR2}}$ is $C^{\text{1D}}_{\text{RR2}}$ (see also [19]). Another capacity-achieving 1D RR constrained coding scheme, implementable using enumerative coding without the need for bridging bits, can be obtained by interleaving codewords from an optimal block code for the RLL $(d, k) = (0, 1)$ constraint [31] on the left-most pages. LOCO codes, however, offer simplicity and reconfigurability, which is important as the device ages [19].

Regarding our 1D 4-ary RR-LOCO coding scheme, we cannot bridge with a single GF(4) symbol between consecutive codewords in $\mathcal{RC}^4_m$ on the two left-most pages since any symbol separating α^2 and α^2 generates a forbidden pattern. We propose a novel two-symbol bridging in which *we can*

encode input information bits within the bridging interval as follows:

- For input information bits $00 \in \mathrm{GF}(2)$, bridge with $00 \in \mathrm{GF}(4)$.
- For input information bits $01 \in \mathrm{GF}(2)$, bridge with $01 \in \mathrm{GF}(4)$.
- For input information bits $10 \in \mathrm{GF}(2)$, bridge with $10 \in \mathrm{GF}(4)$.
- For input information bits $11 \in \mathrm{GF}(2)$, bridge with $11 \in \mathrm{GF}(4)$.

While it has no effect on the asymptotic rate, this bridging scheme remarkably reduces the code length at which a specific rate is achieved, significantly reducing the complexity and error propagation in consequence.

To achieve self-clocking, we remove the two codewords $\mathbf{0}^m$ and $\mathbf{1}^m$, which is expected given the bridging above [19], [24]. Thus, the rate on the two left-most pages is $(\lfloor \log_2(N_4(m) - 2) \rfloor + 2)/(m+2)$ bits/symbol, and the normalized rate of our 1D 4-ary RR-LOCO coding scheme is:

$$R_{\text{RR4}}^{\text{1D}} = \frac{1}{\log_2 q} \left[\frac{\lfloor \log_2(N_4(m) - 2) \rfloor + 2}{m+2} + \log_2 q - 2 \right]. \tag{50}$$

1D 4-ary RR-LOCO coding schemes are capacity-achieving schemes in the sense that the limit as $m \to \infty$ of $R_{\text{RR4}}^{\text{1D}}$ is $C_{\text{RR4}}^{\text{1D}}$. RR-LOCO codes offer simplicity and reconfigurability, which is important as the device ages [19].

Remark 3. *The removal of the two codewords $\mathbf{0}^m$ and $\mathbf{1}^m$ from $\mathcal{RC}_m^4$ to achieve self-clocking is addressed in the encoding algorithm by adding to the decimal integer equivalent of the binary message, and in the decoding algorithm by subtracting from the codeword index. This allows avoiding the codewords $\mathbf{0}^m$ and $\mathbf{1}^m$ while covering all possible messages.*

The 2D binary RR constrained coding scheme we propose requires no additional complexity for encoding and decoding since data is written/read directly to/from specific positions on the left-most page and directly to/from all positions on other pages. As for the 1D binary RR-LOCO coding scheme, the complexity is governed by the size of the adder that executes the encoding-decoding rule, which is:

$$s_2 = \lfloor \log_2(N_2(m) - 1) \rfloor \tag{51}$$

bits. Similarly and as for the 1D 4-ary RR-LOCO coding scheme, the complexity is governed by the adder size, which is:

$$s_4 = \lfloor \log_2(N_4(m) - 2) \rfloor \tag{52}$$

bits. For ease of implementation and to avoid affecting the access speed, we prefer to apply the 1D RR-LOCO coding schemes along wordlines instead of bitlines since the performance is very close, as demonstrated by the experimental results in Section VI.

Error propagation is the phenomenon that a single writing error results in multiple errors while reading. The 2D binary RR coding scheme does not incur any error propagation. Thus, the error propagation factor of it is $E_{\text{RR2}}^{\text{2D}} = 1$. As for the 1D binary RR-LOCO coding scheme, there is no codeword-to-codeword error propagation. However, there exists limited error propagation resulting from the codeword-to-message conversion [8], [19] on the left-most page only. This error propagation reaches $s_2/2$ bits on average, where s_2 is the message length as well from (51). Consequently, the error propagation factor averaged over $\log_2 q$ pages is:

$$E_{\text{RR2}}^{\text{1D}} = \frac{1}{\log_2 q} \left[\frac{s_2}{2} + \log_2 q - 1 \right]. \tag{53}$$

As for the 1D 4-ary RR-LOCO coding scheme, again there exists limited error propagation resulting solely from the LOCO codeword-to-message conversion [8], [19] on the two left-most pages. This error propagation reaches $s_4/2$ bits on average, where s_4 is the message length as well from (52). Observe that there is no error propagation for the two additional bits encoded at each bridging interval to specify the two 4-ary bridging symbols. Therefore, the average error propagation on any of these two left-most pages is:

$$\frac{s_4}{2} \cdot \frac{m}{m+2} + 1 \cdot \frac{2}{m+2} = \frac{s_4 m + 4}{2(m+2)}. \tag{54}$$

Consequently, the error propagation factor averaged over $\log_2 q$ pages is:

$$\begin{aligned} E_{\text{RR4}}^{\text{1D}} &= \frac{1}{\log_2 q} \left[2 \cdot \frac{s_4 m + 4}{2(m+2)} + \log_2 q - 2 \right] \\ &= \frac{1}{\log_2 q} \left[\frac{s_4 m + 4}{m+2} + \log_2 q - 2 \right]. \end{aligned} \tag{55}$$

Another metric to compare 1D binary with 1D 4-ary RR-LOCO coding schemes is the amount of coded data at a given rate. As this amount decreases, the code allows achieving the desired rate at a smaller length m, which is an advantage. Since for our 1D binary and 1D 4-ary RR-LOCO coding schemes we use two bits and two symbols for bridging, respectively, these amounts of coded data, $D_{\text{RR2}}^{\text{1D}}$ (binary) and $D_{\text{RR4}}^{\text{1D}}$ (4-ary) are:

$$D_{\text{RR2}}^{\text{1D}} = (m+2)\log_2 q, \ m \text{ is the length of } \mathcal{RC}_m^2, \tag{56}$$

$$D_{\text{RR4}}^{\text{1D}} = (m+2)\log_2 q, \ m \text{ is the length of } \mathcal{RC}_m^4. \tag{57}$$

Table III gives the normalized rates, adder sizes, and error propagation factors of the proposed binary RR schemes under various parameters. The 1D binary RR-LOCO coding scheme has a remarkable rate advantage that reaches 10.147%, 6.096%, and 4.343% for $q = 4$, $q = 8$, and $q = 16$, respectively, over the 2D binary RR constrained coding scheme. The 2D binary RR scheme has a clear advantage in terms of both complexity and error propagation as it requires no processing to encode and decode. Having said that, the error propagation factor of the 1D binary RR scheme decreases notably as q increases. For example, $E_{\text{RR2}}^{\text{1D}} = 2.625$ for $q = 16$ and $m = 21$, which is remarkably small given the code length.

In Table IV, we compare 1D binary with 1D 4-ary RR-LOCO coding schemes in a different way. In particular, we fix the normalized rate, and find the minimum amount of coded data and the minimum complexity (adder size) required to achieve this desired rate for the two coding schemes, in addition to the minimum error propagation associated with them.[6] The sign "$-$" is used in the table whenever the

[6]Achieving a desired rate here means reaching a normalized rate greater than or equal to this desired rate.

TABLE III
COMPARISONS OF RATE, COMPLEXITY, AND ERROR PROPAGATION AT THE SAME LENGTH BETWEEN 2D RR AND 1D BINARY RR CONSTRAINED CODING SCHEMES – THE CAPACITY IS ALSO SHOWN

q	m	$R^{\text{2D}}_{\text{RR2}}$	$R^{\text{1D}}_{\text{RR2}}$	$C^{\text{2D}}_{\text{RR2}}$	$C^{\text{1D}}_{\text{RR2}}$	s_2	$E^{\text{2D}}_{\text{RR2}}$	$E^{\text{1D}}_{\text{RR2}}$
4	7	0.7500	0.7778	0.7939	0.8471	5	1.000	1.750
4	11	0.7500	0.8077	0.7939	0.8471	8	1.000	2.500
4	21	0.7500	0.8261	0.7939	0.8471	15	1.000	4.250
8	7	0.8333	0.8519	0.8626	0.8981	5	1.000	1.500
8	11	0.8333	0.8718	0.8626	0.8981	8	1.000	2.000
8	21	0.8333	0.8841	0.8626	0.8981	15	1.000	3.167
16	7	0.8750	0.8889	0.8970	0.9235	5	1.000	1.375
16	11	0.8750	0.9038	0.8970	0.9235	8	1.000	1.750
16	21	0.8750	0.9130	0.8970	0.9235	15	1.000	2.625

binary coding scheme cannot achieve such a rate. The main conclusions from Table IV are:

- For $q = 8$ and $q = 16$, the 4-ary coding scheme requires less coded data (smaller lengths) than the binary coding scheme does for all desired rates. The difference in favor of the 4-ary coding scheme increases as the rate increases.
- At lower rates, the complexity of the binary coding scheme is lower than that of the 4-ary coding scheme. However, at rates ≥ 0.8900 for $q = 8$ and ≥ 0.9150 for $q = 16$, the 4-ary coding scheme wins the complexity competition.
- As expected, the binary coding scheme incurs less error propagation in general because LOCO coding is performed on one page only. However, at higher rates and higher q, the 4-ary coding scheme becomes quite competitive to the intriguing extent that it already incurs less error propagation at rate 0.9200 and $q = 16$.

The 1D and 2D RR coding schemes can be used in the same device, but at different lifetime stages. A 1D RR-LOCO coding scheme, binary or 4-ary, can be used when the device is relatively fresh or until a moderate number of program/erase (P/E) cycles, while the 2D RR constrained coding scheme can be used when the device ages, where preventing the error-prone patterns in both directions could make a difference and the associated rate loss could be acceptable. However, this performance difference is shown to be small in Section VI, at least for the TLC Flash device we used. The section also shows that the performance difference between 1D binary and 1D 4-ary RR-LOCO coding schemes is negligible.

Remark 4. *An idea that allows page separation for MLC Flash was introduced in [16]. However, the rate offered is only* 0.7500*, which is significantly below the rates offered via our 1D binary RR coding scheme for MLC. Another idea that allows page separation for TLC Flash was introduced in [17]. However, it only heuristically addresses the level pattern* 707.

VI. EXPERIMENTAL RESULTS ON TLC FLASH

To characterize the performance of the proposed RR constrained coding schemes, we conducted program/erase (P/E) cycling experiments on several blocks of a commercial 1X-nm TLC Flash chip, as follows:

1) Erase Flash memory block under test.
2) Program all pages of block under test with data. For uncoded experiments, program pseudo-random data at each P/E cycle. For RR experiments, program prepared data satisfying RR constraints at each P/E cycle.
3) For each successive P/E cycle of RR experiments, "rotate" the data, so the data that was written on the page i is written on the page $(i + 1)$, wrapping around the last page to the first page.
4) Record bit errors and compute channel bit error rate (BER) every 100 P/E cycles.

TABLE IV
COMPARISONS OF MINIMUM CODED DATA, COMPLEXITY, AND ERROR PROPAGATION TO ACHIEVE CERTAIN RATE BETWEEN 1D BINARY RR AND 1D 4-ARY RR CONSTRAINED CODING SCHEMES

q	Rate	$D^{\text{1D}}_{\text{RR2}}$	$D^{\text{1D}}_{\text{RR4}}$	s_2	s_4	$E^{\text{1D}}_{\text{RR2}}$	$E^{\text{1D}}_{\text{RR4}}$
8	0.8500	27	21	5	9	1.500	2.667
8	0.8750	48	24	10	11	2.333	3.250
8	0.8900	138	48	31	25	5.833	7.708
8	0.9000	–	60	–	32	–	10.000
16	0.8900	48	28	7	9	1.625	2.250
16	0.9050	64	32	10	11	2.000	2.688
16	0.9150	144	48	24	18	3.750	4.333
16	0.9200	288	64	49	25	6.875	6.031
16	0.9300	–	100	–	41	–	9.970

The PE cycling experiments were performed at room temperature in a continuous manner with no wait time between the erase-program-read operations.

Gray mappings used in Flash devices may vary between manufacturers and product generations. In our preliminary work [1], we modified the forbidden binary patterns in accordance with the device mapping so that RR coding on one page per wordline would eliminate most of the patterns in $\mathcal{L}_q$ that induce the most severe ICI (see Remark 1).

In this work, the 8-ary encoded level sequences generated by the RR encoders described herein using the RAGM mapping were translated according to the device Gray mapping into the corresponding binary sequences for the lower, middle, and upper pages in the TLC Flash memory. Thus, the 8-ary level sequences stored in the memory are precisely the RR-encoded level sequences (each cell is programmed to a level in $\{0, 1, \ldots, q - 1\}$).

The left subfigure in Fig. 4 shows the channel BER from P/E cycle 0 to P/E cycle $10{,}000$ using pseudo-random data, a rate $24{:}36$ 1D binary RR-LOCO code along wordlines or bitlines, and a rate $20{:}12$ bits/symbol 1D 4-ary RR-LOCO code along wordlines or bitlines. The right subfigure in Fig. 4 shows the channel BER from P/E cycle $4{,}000$ to P/E cycle $10{,}000$ for these cases in more detail. Note that the binary RR code and 4-

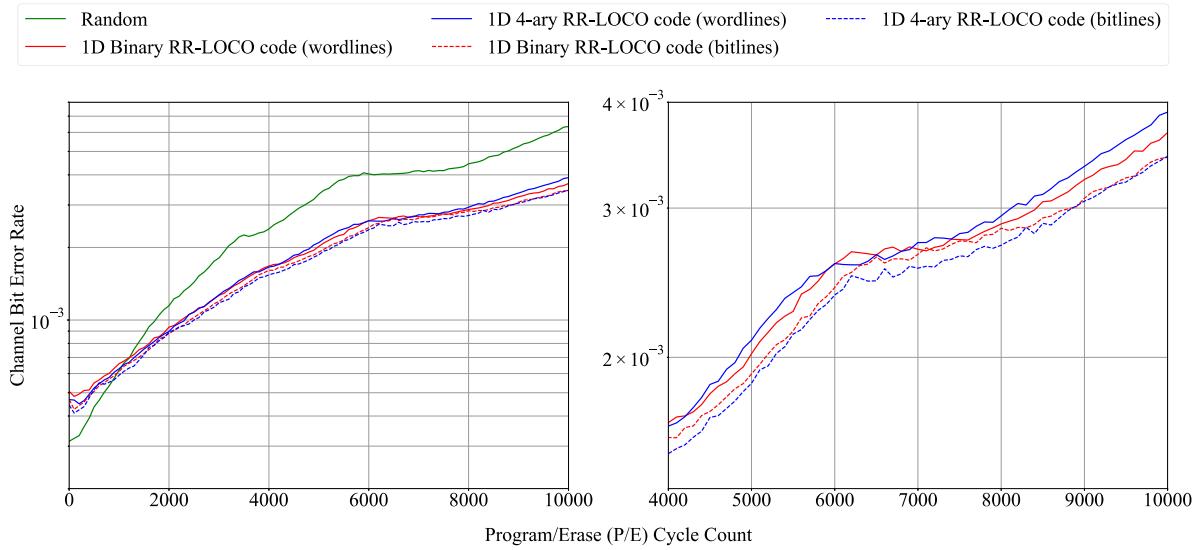

Fig. 4. (Left) Measured average channel BER comparison when all pages are programmed with random data (green curve), 1D binary RR-LOCO coded data (red curves) along wordlines (solid curve) or bitlines (dashed curve), and 1D 4-ary RR-LOCO coded data (blue curves) along wordlines (solid curve) or bitlines (dashed curve) from P/E cycle 0 to P/E cycle 10,000. (Right) Measured average channel BER excluding random data from P/E cycle 4,000 to P/E cycle 10,000.

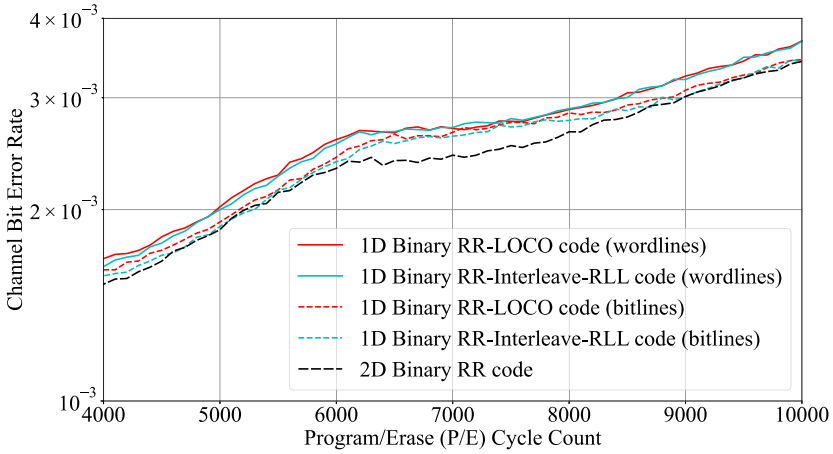

Fig. 5. Measured average channel BER comparison of 1D binary RR-LOCO coded data (red curves) along wordlines (solid curve) or bitlines (dashed curve), 1D binary interleaved RLL-$(0,1)$ coded data (cyan curves) along wordlines (solid curve) or bitlines (dashed curve), and 2D binary RR coded data (black curve) from P/E cycle 4,000 to P/E cycle 10,000.

ary RR code have the same overall rate: $R_{\text{RR2}}^{\text{1D}} = 8/9 \approx 0.8889$ using (49) and $R_{\text{RR4}}^{\text{1D}} = 8/9 \approx 0.8889$ using (50). Therefore, the 1D binary coding scheme achieves about 99% (96%) of the capacity $C_{\text{RR2}}^{\text{1D}}$ ($C_{\mathcal{L}_q}^{\text{1D}}$) and the 1D 4-ary coding scheme achieves about 96% of the capacity $C_{\text{RR4}}^{\text{1D}}$.

As shown in Fig. 4, the uncoded performance is better than that of both binary and 4-ary RR codes up to around 1,200 P/E cycles and is notably worse thereafter. At the later stages of P/E cycling, ICI becomes severe and RR codes outperform the uncoded setting. Specifically, 1D binary RR-LOCO codes along wordlines increase device lifetime by about 1,800 P/E cycles when channel BER is 2×10^{-3}, representing a 57% lifetime gain, and achieve about 3,700 P/E cycles gain when channel BER is 3×10^{-3}, corresponding to a 79% lifetime gain. As shown in the right subfigure of Fig. 4, the BER of 1D binary RR code along wordlines is almost the same as that of the 4-ary RR code between 2,000 and 8,000 P/E cycles. When the P/E cycle count is larger than 8,000, the BER of 1D binary RR code along wordlines is slightly better than that of the 1D 4-ary RR code. In particular, when channel BER is 3×10^{-3}, the 1D binary RR code along wordlines provides a lifetime that is about 300 P/E cycles larger that than obtained with the 1D 4-ary RR code along wordlines. Along the bitline direction, quite intriguingly, the performance of the 1D 4-ary RR-LOCO code is generally better than, though close to, that of the 1D binary RR-LOCO code. The advantage of the 1D 4-ary RR-LOCO is most pronounced from P/E cycle 6,300 to P/E cycle 8,300. For both binary and 4-ary schemes, coding along bitlines generally offers slightly better performance than coding along wordlines.

Fig. 5 compares the BER performance of different implementations of binary RR codes at high P/E cycles: the 24:36 1D binary RR-LOCO code along the wordline or bitline direction, the 1D binary interleaved 12:18 RLL $(d, k) = (0, 1)$ code (which has an overall block length 36 after interleaving) along the wordline or bitline direction, and the 2D binary

RR code. Using (48), we obtain $R^{\text{2D}}_{\text{RR2}} = 5/6 \approx 0.8333$. Therefore, the 2D coding scheme achieves about 93% (90%) of the capacity $C^{\text{2D}}_{\text{RR2}}$ ($C^{\text{1D}}_{\mathcal{L}_q}$).

Referring to Fig. 5, we make the following observations at all P/E cycles. Each 1D RR coding scheme along the bitline direction achieves a slightly better channel BER performance than along the wordline direction; the 1D RR coding schemes along the same direction have similar performance; and the performance of the 2D RR constrained code is better than that of the 1D RR codes along any one direction. For example, when channel BER is 2×10^{-3}, the 2D binary RR coding increases lifetime by 100 P/E cycles over the 1D binary RR-LOCO coding along bitlines and 300 P/E cycles over the 1D binary RR-LOCO coding along wordlines. When channel BER is 3×10^{-3} and the wear condition of the Flash device is severe, the 2D binary RR coding outperforms the 1D binary RR-LOCO coding along bitlines by about 200 P/E cycles and the 1D binary RR-LOCO coding along wordlines by about 600 P/E cycles.

These measurements confirm some of the claimed practical advantages of 4-ary RR codes. The performance results of the 1D binary RR-LOCO code and the 1D 4-ary RR-LOCO code along both wordline and bitline directions are very similar, and the designed codes have the same overall rate (including bridging symbols). The 1D 4-ary RR-LOCO code has a shorter overall block length corresponding to 12 bits per coded page (10 symbols plus 2 bridging symbols) in comparison to the 1D binary RR-LOCO code which has overall block length of 36 bits on the coded page. Moreover, in the code design, the 1D 4-ary RR-LOCO code requires an adder size of 18 bits, while the 1D binary RR-LOCO code requires an adder size of 24 bits. Recall that the adder size governs the LOCO encoding-decoding complexity.

An examination of level probabilities induced by 1D binary and 1D 4-ary RR constraints provides some intuitive insight into the experimental results in Figs. 4 and 5. The probabilities of binary symbols 0 and 1 under the RLL $(d, k) = (0, 1)$ constraint are approximately 0.2764 and 0.7236, respectively [27]. Asymptotically, this leads to probabilities of individual symbols corresponding to levels in $\mathcal{V}_0 = \{4, 5, 6, 7\}$ and $\mathcal{V}_1 = \{0, 1, 2, 3\}$ of about 0.0691 and 0.1809, respectively. From the FSTD of the 4-ary constraint forbidding patterns in $\mathcal{R}_4$, shown in Fig. 3, we find that the probabilities of individual symbols corresponding to levels in $\mathcal{W}_0 = \{6, 7\}$, $\mathcal{W}_1 = \{4, 5\}$, $\mathcal{W}_2 = \{2, 3\}$, and $\mathcal{W}_3 = \{0, 1\}$ are about 0.0787, 0.1030, 0.1591, and 0.1591, respectively. Bridging symbols change these probabilities slightly, further increasing the probabilities of symbols corresponding to levels in $\{0, 1, 2, 3\}$ relative to symbols corresponding to levels in $\{4, 5, 6, 7\}$. These probabilities contrast with those of uncoded random data, where each symbol/level has the same probability of $1/8 = 0.125$.

The modified symbol probabilities help to explain the observed relative performances of the 1D binary RR codes in the wordline and bitline directions along with the 2D RR code. Applying 1D binary RR coding in the wordline direction also indirectly reduces the probability of detrimental patterns in the bitline direction, and vice versa. This reduces the expected advantage of bitline coding over wordline coding in the presence of more severe ICI in the bitline direction. Similarly, the advantage of 2D coding over 1D coding in either direction is less than expected (even without taking into account the rate penalty associated with 2D coding).

We remark that the designed codes are efficient, with rates fairly close to capacity, and the symbol and pattern probabilities observed in the data written to the Flash memory are close to the theoretical values mentioned above.

The cross-over behavior observed in Fig. 4 can be explained if the level patterns eliminated by the code, especially ICI-prone patterns, are not the only significant contributors to error early in the device lifetime. The RR coding significantly changes level probabilities compared with the uncoded setting, possibly increasing the probability of some of the remaining level patterns that cause errors due to other effects, and accordingly increasing their contribution to the BER at low P/E cycles. One way to address this issue is to delay the introduction of coding until later P/E cycles when ICI affects performance. Alternatively, one might apply different constraints at different P/E cycles before and after the cross-over point, much as adaptive error-correction code designs have been proposed to achieve different degrees of protection at various stages of the Flash device lifetime [32], [33]. The reconfigurability feature of LOCO code designs could be exploited, and a machine learning module could be used to identify the device status and direct the transition from one code to another at the appropriate time based on that status. In this regard, we also note that machine learning modeling, as proposed in [34], can be used to characterize the spatio-temporal ICI effects of the Flash memory device and provide a tool for optimizing the offline design of RR-LOCO codes. Another possible approach is to optimize system performance through a combination of signal processing methods [20], [21] and RR coding. These ideas represent intriguing directions for future research.

VII. Conclusion

We introduced read-and-run (RR) constrained coding schemes for modern Flash devices. RR coding schemes eliminate patterns prone to ICI-induced errors while allowing systematic encoder and decoder implementations, high overall rates, and page separation in data recovery. We analyzed properties of 1D binary RR-LOCO codes, 1D 4-ary RR-LOCO codes, and a 2D binary RR code. The three RR coding schemes offer different advantages, and we suggest that system requirements at different stages of the device lifetime should determine the most suitable scheme or schemes to use. Experimental results reveal significant P/E-cycle lifetime gains in a commercial Flash device. Future work includes the incorporation of LDPC codes [35] with RR coding schemes and the development of machine learning-aided, reconfigurable RR coding schemes to maximize Flash device lifetime.

References

[1] A. Hareedy, S. Zheng, P. Siegel, and R. Calderbank, "Read-and-run constrained coding for modern Flash devices," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Seoul, South Korea, May 2022, pp. 1–6.

[2] C. E. Shannon, "A mathematical theory of communication," Bell Sys. Tech. J., vol. 27, Oct. 1948.

[3] D. T. Tang and R. L. Bahl, "Block codes for a class of constrained noiseless channels," *Inf. and Control*, vol. 17, no. 5, pp. 436–461, 1970.

[4] T. Cover, "Enumerative source encoding," *IEEE Trans. Inf. Theory*, vol. 19, no. 1, pp. 73–77, Jan. 1973.

[5] P. A. Franaszek, "Sequence-state methods for run-length-limited coding," *IBM J. Res. Dev.*, vol. 14, no. 4, pp. 376–383, Jul. 1970.

[6] R. Adler, D. Coppersmith, and M. Hassner, "Algorithms for sliding block codes–An application of symbolic dynamics to information theory," *IEEE Trans. Inf. Theory*, vol. 29, no. 1, pp. 5–22, Jan. 1983.

[7] K. A. S. Immink, P. H. Siegel, and J. K. Wolf, "Codes for digital recorders," *IEEE Trans. Inf. Theory*, vol. 44, no. 6, pp. 2260–2299, Oct. 1998.

[8] A. Hareedy and R. Calderbank, "LOCO codes: Lexicographically-ordered constrained codes," *IEEE Trans. Inf. Theory*, vol. 66, no. 6, pp. 3572–3589, Jun. 2020.

[9] B. Vasic and E. Kurtas, *Coding and Signal Processing for Magnetic Recording Systems.* CRC Press, 2005.

[10] S. Zheng, Y. Liu, and P. H. Siegel, "PR-NN: RNN-based detection for coded partial-response channels," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 1967–1982, Jul. 2021.

[11] B. Dabak, A. Hareedy, and R. Calderbank, "Non-binary constrained codes for two-dimensional magnetic recording," *IEEE Trans. Magn.*, vol. 56, no. 11, pp. 1–10, Nov. 2020.

[12] R. Wood, M. Williams, A. Kavcic, and J. Miles, "The feasibility of magnetic recording at 10 terabits per square inch on conventional media," *IEEE Trans. Magn.*, vol. 45, no. 2, pp. 917–923, Feb. 2009.

[13] K. A. S. Immink, "Modulation systems for digital audio discs with optical readout," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, Atlanta, Georgia, USA, Mar.–Apr. 1981, pp. 587–589.

[14] J. Saadé, A. Goulahsen, A. Picco, J. Huloux, and F. Pétrot, "Low overhead, DC-balanced and run length limited line coding," in *Proc. IEEE 19th Workshop on Signal and Power Integrity (SPI)*, Berlin, Germany, May 2015, pp. 1–4.

[15] J.-D. Lee, S.-H. Hur, and J.-D. Choi, "Effects of floating-gate interference on NAND flash memory cell operation," *IEEE Electron Device Lett.*, vol. 23, no. 5, pp. 264–266, May 2002.

[16] V. Taranalli, H. Uchikawa, and P. H. Siegel, "Error analysis and inter-cell interference mitigation in multi-level cell flash memories," in *Proc. IEEE Int. Conf. Commun. (ICC)*, London, UK, Jun. 2015, pp. 271–276.

[17] R. Motwani, "Hierarchical constrained coding for floating-gate to floating-gate coupling mitigation in Flash memory," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, Houston, TX, USA, Dec. 2011, pp. 1–5.

[18] Y. M. Chee, J. Chrisnata, H. M. Kiah, S. Ling, T. T. Nguyen, and V. K. Vu, "Capacity-achieving codes that mitigate intercell interference and charge leakage in Flash memories," *IEEE Trans. Inf. Theory*, vol. 65, no. 6, pp. 3702–3712, Jun. 2019.

[19] A. Hareedy, B. Dabak, and R. Calderbank, "Managing device lifecycle: Reconfigurable constrained codes for M/T/Q/P-LC Flash memories," *IEEE Trans. Inf. Theory*, vol. 67, no. 1, pp. 282–295, Jan. 2021.

[20] G. Dong, S. Li, and T. Zhang, "Using data postcompensation and predistortion to tolerate cell-to-cell interference in MLC NAND flash memory," *IEEE Trans. Circuits Syst. I, Reg. Papers,* vol. 57, no. 10, pp. 2718–2728, Oct. 2010.

[21] C. A. Aslam, Y. L. Guan, and K. Cai, "Detector for MLC NAND flash memory using neighbor-a-priori information," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.,* vol. 24, no. 9, pp. 2827–2836, Sep. 2016.

[22] V. Braun and K. A. S. Immink, "An enumerative coding technique for DC-free runlength-limited sequences," *IEEE Trans. Commun.*, vol. 48, no. 12, pp. 2024–2031, Dec. 2000.

[23] A. Hareedy and R. Calderbank, "Asymmetric LOCO codes: Constrained codes for Flash memories," in *Proc. 57th Annual Allerton Conf. Commun., Control, and Computing*, Monticello, IL, USA, Sep. 2019, pp. 124–131.

[24] A. Hareedy, B. Dabak, and R. Calderbank, "The secret arithmetic of patterns: A general method for designing constrained codes based on lexicographic indexing," *IEEE Trans. Inf. Theory*, vol. 68, no. 9, pp. 5747–5778, Sep. 2022.

[25] J. Centers, X. Tan, A. Hareedy, and R. Calderbank, "Power spectra of constrained codes with level-based signaling: Overcoming finite-length challenges," *IEEE Trans. Commun.*, vol. 69, no. 8, pp. 4971–4986, Aug. 2021.

[26] S. Zheng, C. -H. Ho, W. Peng, and P. H. Siegel, "Spatio-temporal modeling for flash memory channels using conditional generative nets," *Proc. 2023 Design, Automation & Test in Europe Conf. & Exhib. (DATE)*, Antwerp, Belgium, 2023, pp. 1-6.

[27] P. H. Siegel, "Constrained Codes for Multilevel Flash Memory," presented at *North American School of Information Theory (Padovani Lecture)*, La Jolla, California, Aug. 12, 2015. Available: http://cmrr-star.ucsd.edu/static/presentations/Padovani_Lecture_NASIT_Website.pdf. Video: https://www.youtube.com/watch?v=FCv2PJryUr4.

[28] R. Laroia, N. Farvardin, and S. A. Tretter, "On optimal shaping of multidimensional constellations," *IEEE Trans. Inf. Theory*, vol. 40, no. 4, pp. 1044–1056, Jul. 1994.

[29] N. Kashyap, R. M. Roth and P. H. Siegel, "The capacity of count-constrained ICI-free systems," in *IEEE Int. Symp. Inf. Theory*, Paris, France, Jul. 2019, pp. 1592–1596.

[30] A. Kato and K. Zeger, "On the capacity of two-dimensional run-length constrained channels," *IEEE Trans. Inf. Theory*, vol. 45, no. 5, pp. 1527–1540, Jul. 1999.

[31] B. H. Marcus, P. H. Siegel and J. K. Wolf, "Finite-state modulation codes for data storage," in *IEEE J. Sel. Areas Commun.*, vol. 10, no. 1, pp. 5–37, Jan. 1992.

[32] P. Chen, K. Cai and S. Zheng, "Rate-adaptive protograph LDPC codes for multi-level-cell NAND flash memory," *IEEE Commun. Lett.*, vol. 22, no. 6, pp. 1112–1115, Jun. 2018

[33] S.-H. Lim, J.-B. Lee, G.-M. Kim and W. H. Ahn, "A stepwise rate-compatible LDPC and parity management in NAND flash memory-based storage devices," *IEEE Access*, vol. 8, pp. 162491–162506, 2020.

[34] S. Zheng and P. H. Siegel, "Code-aware storage channel modeling via machine learning," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Mumbai, India, Nov. 2022, pp. 196–201.

[35] A. Hareedy, R. Kuditipudi, and R. Calderbank, "Minimizing the number of detrimental objects in multi-dimensional graph-based codes," *IEEE Trans. Commun.*, vol. 68, no. 9, pp. 5299–5312, Sep. 2020.

Ahmed Hareedy (Member, IEEE) is an Assistant Professor with the Department of Electrical and Electronics Engineering at Middle East Technical University (METU), Turkey. He is interested in questions in coding/information theory that are fundamental to opportunities created by the current, unparalleled access to data and computing. He received the Bachelor and M.S. degrees in Electronics and Communications Engineering from Cairo University, Egypt, in 2006 and 2011, respectively. He received the Ph.D. degree in Electrical and Computer Engineering from the University of California, Los Angeles (UCLA), USA, in 2018. He was a Postdoctoral Associate with the Department of Electrical and Computer Engineering at Duke University, USA, between 2018 and 2021. He worked with Mentor Graphics Corporation (currently, Siemens EDA) between 2006 and 2014. He worked as an Error-Correction Coding Architect with Intel Corporation in the summers of 2015 and 2017.

Dr. Hareedy won the 2018–2019 Distinguished Ph.D. Dissertation Award in Signals and Systems from the Department of Electrical and Computer Engineering at UCLA. He is a recipient of the Best Paper Award from the 2015 IEEE Global Communications Conference (GLOBECOM), Selected Areas in Communications, Data Storage Track. He won the 2017–2018 Dissertation Year Fellowship (DYF) at UCLA. He won the 2016–2017 Electrical Engineering Henry Samueli Excellence in Teaching Award for teaching Probability and Statistics at UCLA. He is a recipient of the Memorable Paper Award from the 2018 Non-Volatile Memories Workshop (NVMW) in the area of devices, coding, and information theory. He is a recipient of the 2018–2019 Best Student Paper Award from the IEEE Data Storage Technical Committee (DSTC). He has been awarded the TÜBİTAK 2232-B International Fellowship for Early Stage Researchers in 2022. He is currently a Guest Editor of the Special Issue on Data Storage of the IEEE BITS THE INFORMATION THEORY MAGAZINE.

Simeng Zheng (Student Member, IEEE) received the B.E. degree in Electronic and Information Engineering from Beihang University, Beijing, China, in 2018 and the M.S. degree in Electrical and Computer Engineering from the University of California, San Diego (UCSD), CA, USA, in 2020. He is currently working towards the Ph.D. degree with the Department of Electrical and Computer Engineering, UCSD, where he is also affiliated with the Center for Memory and Recording Research. His current research interests are in data storage and machine learning.

Paul Siegel (Life Fellow, IEEE) received the S.B. and Ph.D. degrees in Mathematics from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 1975 and 1979, respectively. He held a Chaim Weizmann Postdoctoral Fellowship with the Courant Institute, New York University, New York, NY, USA. He was with the IBM Research Division, San Jose, CA, USA, from 1980 to 1995. He joined the faculty at the University of California San Diego (UCSD), La Jolla, CA, USA, in 1995, where he is currently a Distinguished Professor of Electrical and Computer Engineering with the Jacobs School of Engineering. He is affiliated with the Center for Memory and Recording Research where he holds an Endowed Chair and served as Director from 2000 to 2011. His research interests include information theory, coding techniques, and machine learning, with applications to digital data storage and transmission. He is a Member of the National Academy of Engineering. He was a Member of the Board of Governors of the IEEE Information Theory Society from 1991 to 1996 and from 2009 to 2014. He was the 2015 Padovani Lecturer of the IEEE Information Theory Society. He was a co-recipient of the 1992 IEEE Information Theory Society Paper Award, the 1993 IEEE Communications Society Leonard G. Abraham Prize Paper Award, and the 2007 Best Paper Award in Signal Processing and Coding for Data Storage from the Data Storage Technical Committee of the IEEE Communications Society. He served as an Associate Editor of Coding Techniques of the IEEE TRANSACTIONS ON INFORMATION THEORY from 1992 to 1995, and as the Editor-in-Chief from 2001 to 2004. He served as a Co-Guest Editor of the 1991 Special Issue on Coding for Storage Devices of the IEEE TRANSACTIONS ON INFORMATION THEORY. He was also a Co-Guest Editor of the 2001 two-part issue on The Turbo Principle: From Theory to Practice and the 2016 issue on Recent Advances in Capacity Approaching Codes of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS. He is currently the Lead Guest Editor of the Special Issue on Data Storage of the IEEE BITS THE INFORMATION THEORY MAGAZINE.

Robert Calderbank (Life Fellow, IEEE) received the B.S. degree in 1975 from Warwick University, England, the M.S. degree in 1976 from Oxford University, England, and the Ph.D. degree in 1980 from the California Institute of Technology, USA, all in Mathematics.

Dr. Calderbank is a Professor of Electrical and Computer Engineering at Duke University where he directs the Rhodes Information Initiative at Duke (iiD). Prior to joining Duke in 2010, he was a Professor of Electrical Engineering and Mathematics at Princeton University where he directed the Program in Applied and Computational Mathematics. Prior to joining Princeton in 2004, he was the Vice President for Research at AT&T, responsible for directing the first industrial research lab in the world where the primary focus is data at scale. At the start of his career at Bell Labs, innovations by him were incorporated in a progression of voiceband modem standards that moved communications practice close to the Shannon limit. Together with Peter Shor and colleagues at AT&T Labs he developed the mathematical framework for quantum error correction. He is a co-inventor of space-time codes for wireless communication, where correlation of signals across different transmit antennas is the key to reliable transmission.

Dr. Calderbank served as the Editor in Chief of the IEEE TRANSACTIONS ON INFORMATION THEORY from 1995 to 1998, and as an Associate Editor for Coding Techniques from 1986 to 1989. He has recently served as the Editor in Chief of the IEEE BITS THE INFORMATION THEORY MAGAZINE. He was a member of the Board of Governors of the IEEE Information Theory Society from 1991 to 1996 and from 2006 to 2008. He was honored by the IEEE Information Theory Prize Paper Award in 1995 for his work on the $\mathbb{Z}_4$ linearity of Kerdock and Preparata Codes (joint with A.R. Hammons Jr., P.V. Kumar, N.J.A. Sloane, and P. Sole), and again in 1999 for the invention of space-time codes (joint with V. Tarokh and N. Seshadri). He has received the 2006 IEEE Donald G. Fink Prize Paper Award, the IEEE Millennium Medal, the 2013 IEEE Richard W. Hamming Medal, and the 2015 Shannon Award. He was elected to the US National Academy of Engineering in 2005.