Q-ary Asymmetric LOCO Codes: Constrained Codes Supporting Flash Evolution

Ahmed Hareedy, Beyza Dabak, and Robert Calderbank
Electrical and Computer Engineering Department, Duke University, Durham, NC 27708 USA
ahmed.hareedy@duke.edu, beyza.dabak@duke.edu, and robert.calderbank@duke.edu

***Abstract*—Flash memory devices are winning the competition for storage density against magnetic recording devices. This outcome results from advances in physics that allow storage of more than one bit per cell, coupled with advances in signal processing that reduce the effect of physical instabilities. Constrained codes are used in storage to avoid problematic patterns. Recently, we introduced binary symmetric lexicographically-ordered constrained codes (LOCO codes) for data storage and transmission. This paper introduces simple constrained codes that support non-binary physical gates in multi, triple, quad, and the currently-in-development penta-level cell (M/T/Q/P-LC) Flash memories. The new codes can be easily modified if problematic patterns change with time. These codes are designed to mitigate inter-cell interference, which is a critical source of error in Flash devices. The new codes are called q-ary asymmetric LOCO codes (QA-LOCO codes), and the construction subsumes codes previously designed for single-level cell (SLC) Flash devices (A-LOCO codes). QA-LOCO codes work for a Flash device with any number, q, of levels per cell. For $q \geq 4$, we show that QA-LOCO codes can achieve rates greater than $0.95 \log_2 q$ information bits per coded symbol. Capacity-achieving rates, affordable encoding-decoding complexity, and ease of reconfigurability support the growing improvement of M/T/Q/P-LC Flash memory devices, as well as lifecycle management as the characteristics of these devices change with time.**

I. Introduction

Data storage densities are increasing rapidly as modern applications, e.g., internet of things applications, access, process, and store more and more data. In 2015, the storage density of Flash memory devices surpassed that of magnetic recording (MR) devices. This milestone resulted from advances in physics, architecture, and signal processing. The major advance in Flash physics was enabling more than two storage levels per cell, and thus allowing the storage of more than one bit per cell. The major advance in Flash architecture was devising the three-dimensional vertical NAND Flash structure.

Constrained codes are designed to avoid problematic patterns. Run-length-limited (RLL) codes are a class of constrained codes introduced in 1970 [1], that were first used to improve the storage density of early MR devices employing peak detection [2], [3]. Modern storage devices employ sequence estimation rather than peak detection, but constrained codes are still used to improve performance [3], [4]. RLL codes also find application in optical recording [5]. When first introduced in [1], lexicographic indexing was used to encode and decode RLL codes, but this was replaced by methods based on finite-state machines (FSMs) in later work [6]. RLL codes are associated with transition-based signaling.

In level-based signaling, each symbol (or bit) is associated with a distinct level for storage or transmission. A binary symmetric $\mathcal{S}_x$-constrained code is a code that forbids the patterns in the set $\mathcal{S}_x \triangleq \{010, 101, 01^20, 10^21, \ldots, 01^x0, 10^x1\}$ from appearing in any codeword, where the notation $\mathbf{y}^r$ refers to a sequence of r consecutive y's. A binary asymmetric $\mathcal{A}_x$-constrained code is a code that forbids the patterns in the set $\mathcal{A}_x \triangleq \{101, 10^21, \ldots, 10^x1\}$ from appearing in any codeword. $\mathcal{S}_x$-constrained and $\mathcal{A}_x$-constrained codes are associated with level-based signaling, which is natural for Flash.

In Flash devices, inter-cell interference (ICI) is one of the main sources of errors. Parasitic capacitances in and across floating gate transistors result in charge propagation from cells being programmed to the highest charge level to neighboring cells being programmed to lower levels or unprogrammed. Thus, unintentional increases in charge values occur, resulting in errors during reading. The authors of [7] and [8] introduced constrained codes to prevent the level pattern $(q-1)0(q-1)$ from being written in a Flash device with $q \geq 2$ levels per cell.[1] Via extensive experiments, the authors of [9] showed that for multi-level cell (MLC) Flash devices (4 levels), the set of level patterns to be forbidden (contribute the most to ICI) should be $\{303, 313, 323\}$. This set was recently generalized in [10] to $\{(q-1)0(q-1), (q-1)1(q-1), \ldots, (q-1)(q-2)(q-1)\}$ for a Flash device with q levels per cell. We focus on ICI among cells on the same bit line because of its dominance [9].

In previous work [11], we introduced lexicographically-ordered $\mathcal{S}_x$-constrained codes (LOCO codes), that make significant MR density gains possible. LOCO codes are simple and reconfigurable. The $\mathcal{A}_x$-constraint forbids ICI-causing patterns in single-level cell (SLC) Flash devices (2 levels). In [12], we designed capacity-achieving $\mathcal{A}_x$-constrained codes, named asymmetric LOCO codes (A-LOCO codes), that offer a better rate-complexity trade-off than previous codes, and that can be easily reconfigured. We anticipate using a combination of machine learning and analysis of errors collected before the error-correction (EC) decoder to identify new patterns that need to be forbidden as the device ages. We see (A-)LOCO codes as a method of extending device lifetime.

In this paper, we generalize our asymmetric constrained codes in [12] to Flash devices with any number, q, of levels per cell. In particular, we introduce fixed-length q-ary asymmetric LOCO codes (QA-LOCO codes) for all Flash devices. QA-LOCO codes are capacity-achieving, and we devise the encoding-decoding rule for them to offer simplicity. While available literature only focuses on the effect of ICI on adjacent cells, we handle more general constraints for higher reliability. QA-LOCO codes are also reconfigurable because of their encoding-decoding rule. We show that QA-LOCO codes can contribute to significant lifetime gains for the Flash device

[1] Note that charge levels directly translate to threshold voltage levels. For simplicity, levels are defined by their indices $\{0, 1, \ldots, q-1\}$.

with rates greater than $0.95 \log_2 q$ information bits per coded symbol, $q \geq 4$, at affordable complexities. High performance EC codes are also key to such gains since they are necessary to correct errors that do not result from ICI. We suggest that QA-LOCO codes (with EC codes) can significantly improve the performance (increase the lifetime) of multi ($q = 4$) and triple ($q = 8$)-level cell Flash memories, and can remarkably accelerate the evolution of quad ($q = 16$) and penta ($q = 32$)-level cell Flash memories, which are the next generation.

The rest of the paper is organized as follows. In Section II, we define QA-LOCO codes and introduce their cardinality. In Section III, we derive the QA-LOCO encoding-decoding rule. In Section IV, we discuss rates and make comparisons. In Section V, we present the encoding and decoding algorithms and discuss reconfigurability. Section VI concludes the paper.

II. Definition and Cardinality

Denote a Galois field (GF) of size q by GF(q). Let α be a primitive element of GF(q).[2] Consequently,

$$\text{GF}(q) \triangleq \{0, 1, \alpha, \alpha^2, \ldots, \alpha^{q-2}\}.$$

We define δ as an element in GF(q)$\backslash\{\alpha^{q-2}\}$ and also $\boldsymbol{\delta}^r \triangleq \delta_{r-1}\delta_{r-2}\ldots\delta_0$ as a sequence in $\left[\text{GF}(q)\backslash\{\alpha^{q-2}\}\right]^r$. We now formally define QA-LOCO codes, which are $\mathcal{Q}_x^q$-constrained:

Definition 1. *A QA-LOCO code $\mathcal{QC}_{m,x}^q$ with $q \geq 2$, $m \geq 1$, and $x \geq 1$ is defined by the following properties:*

1) *Each codeword $\mathbf{c}$ in $\mathcal{QC}_{m,x}^q$ has its symbols in GF(q) and is of length m symbols.*
2) *Codewords in $\mathcal{QC}_{m,x}^q$ are ordered lexicographically.*
3) *Each codeword $\mathbf{c}$ in $\mathcal{QC}_{m,x}^q$ does not contain any of the patterns in the set $\mathcal{Q}_x^q$, where:*

$$\mathcal{Q}_x^q \triangleq \{\alpha^{q-2}\delta\alpha^{q-2}, \alpha^{q-2}\boldsymbol{\delta}^2\alpha^{q-2}, \ldots, \alpha^{q-2}\boldsymbol{\delta}^x\alpha^{q-2}\}. \quad (1)$$

4) *The code $\mathcal{QC}_{m,x}^q$ contains all codewords satisfying the above three properties.*

Lexicographic ordering of codewords means codewords are ordered in an ascending manner following the rule $0 < 1 < \alpha < \cdots < \alpha^{q-2}$ for any symbol, and the symbol significance reduces from left to right.

Let c be an element in GF(q). Define $a \triangleq \mathcal{L}(c)$ as the Flash charge level equivalent to symbol c, which is given by:

$$a \triangleq \mathcal{L}(c) \triangleq \begin{cases} 0, & c = 0, \\ \text{gflog}_\alpha(c) + 1, & \text{otherwise}, \end{cases} \quad (2)$$

where $\text{gflog}_\alpha(c)$ returns the power of the GF element c with $\text{gflog}_\alpha(1) = 0$. Thus, the set of charge levels equivalent to GF(q) is $\{0, 1, 2, 3, \ldots, q-1\}$, and the set of charge-level patterns equivalent to $\mathcal{Q}_x^q$ in (1) is:

$$\{(q-1)\mu(q-1), (q-1)\boldsymbol{\mu}^2(q-1), \ldots, (q-1)\boldsymbol{\mu}^x(q-1)\}, \quad (3)$$

where $\boldsymbol{\mu}^r \triangleq \mathcal{L}(\delta_{r-1})\mathcal{L}(\delta_{r-2})\ldots\mathcal{L}(\delta_0)$.

Observe that in the case of $x = 1$, the set in (1) reduces to $\mathcal{Q}_1^q = \{\alpha^{q-2}\delta\alpha^{q-2}\} = \{\alpha^{q-2}0\alpha^{q-2}, \alpha^{q-2}1\alpha^{q-2}, \ldots, \alpha^{q-2}\alpha^{q-3}\alpha^{q-2}\}$. The set of level patterns equivalent to $\mathcal{Q}_1^q$ is $\{(q-1)0(q-1), (q-1)1(q-1), \ldots, (q-1)(q-2)(q-1)\}$, which is the exact same set in [10], and also in [9] for $q = 4$. It is clear that for the binary case ($q = 2$), $\mathcal{Q}_x^2$ is simply $\mathcal{A}_x$.

The partition of QA-LOCO codewords into groups is essential to deriving the cardinality and later the encoding-decoding rule. We partition the codewords in $\mathcal{QC}_{m,x}^q$, $m \geq 2$, into three groups according to what they start with from the left, i.e., at the left-most symbols (LMSs), as follows:

Group 1: Codewords starting with δ from the left.
Group 2: Codewords starting with $\alpha^{q-2}\alpha^{q-2}$ from the left.
Group 3: Codewords starting with $\alpha^{q-2}\boldsymbol{\delta}^{x+1}$ from the left.

Observe that given the set of forbidden patterns $\mathcal{Q}_x^q$ in (1), there are no other symbol options for a codeword $\mathbf{c}$ in $\mathcal{QC}_{m,x}^q$ to have at its LMSs. Now, we are ready to enumerate QA-LOCO codewords recursively.

Theorem 1. *The cardinality (size) of a QA-LOCO code $\mathcal{QC}_{m,x}^q$, denoted by $N_q(m, x)$, is given by:*

$$N_q(m,x) = qN_q(m-1,x) - (q-1)N_q(m-2,x) + (q-1)^{x+1}N_q(m-x-2,x), \ m \geq 2, \quad (4)$$

where the defined cardinalities are:

$$N_q(m,x) \triangleq (q-1)^m, \ m \leq 0, \ \textit{and} \ N_q(1,x) \triangleq q. \quad (5)$$

Proof: We use the group structure above to prove Theorem 1. See the long version [13]. ∎

Example 1. *Consider the QA-LOCO codes $\mathcal{QC}_{m,1}^4$ ($q = 4$ and $x = 1$) with $m \in \{2, 3, \ldots, 6\}$. From (5), the defined cardinalities needed here are:*

$$N_4(-1,1) \triangleq 3^{-1}, \ N_4(0,1) \triangleq 1, \ \textit{and} \ N_4(1,1) \triangleq 4.$$

The cardinalities of the aforementioned QA-LOCO codes are:

$$N_4(2,1) = 4N_4(1,1) - 3N_4(0,1) + 9N_4(-1,1) = 16,$$
$$N_4(3,1) = 4N_4(2,1) - 3N_4(1,1) + 9N_4(0,1) = 61,$$
$$N_4(4,1) = 4N_4(3,1) - 3N_4(2,1) + 9N_4(1,1) = 232,$$
$$N_4(5,1) = 4N_4(4,1) - 3N_4(3,1) + 9N_4(2,1) = 889, \ \textit{and}$$
$$N_4(6,1) = 4N_4(5,1) - 3N_4(4,1) + 9N_4(3,1) = 3409.$$

Theorem 1 is a key result in the analysis of QA-LOCO codes. The theorem provides insights regarding how the codewords of a QA-LOCO code of a specific length relate to the codewords of QA-LOCO codes of smaller lengths. As we shall see shortly, Theorem 1 and the insights it provides are fundamental to the derivation of the encoding-decoding rule, to the rate discussion, and to the algorithms.

III. QA-LOCO Encoding-Decoding Rule

Now, we derive a formula that relates the lexicographic index of a QA-LOCO codeword to the codeword itself. We call this formula the encoding-decoding rule of QA-LOCO codes since it is the foundation of the QA-LOCO encoding and decoding algorithms presented in Section V.

We define a QA-LOCO codeword of length m symbols as $\mathbf{c} \triangleq c_{m-1}c_{m-2}\ldots c_0$ in $\mathcal{QC}_{m,x}^q$. The index of a QA-LOCO codeword $\mathbf{c}$ in $\mathcal{QC}_{m,x}^q$ is denoted by $g(m, x, \mathbf{c})$, which is sometimes abbreviated to $g(\mathbf{c})$ for simplicity. Our lexicographic index $g(\mathbf{c})$ is in $\{0, 1, \ldots, N_q(m,x) - 1\}$. For each symbol c_i, we define its level-equivalent $a_i \triangleq \mathcal{L}(c_i)$ as shown in (2),

[2] Our analysis works for any GF size q. However, we focus more on $q = 2^v$, $v \geq 1$, because of the nature of Flash devices. We write one symbol per cell.

with $c_i \triangleq 0$ and $a_i \triangleq 0$ for $i \geq m$. The same notation applies for a QA-LOCO codeword of length $m+1$, $\mathbf{c}'$ in $\mathcal{QC}^q_{m+1,x}$, and a QA-LOCO codeword of length $m-x$, $\mathbf{c}''$ in $\mathcal{QC}^q_{m-x,x}$.

For each codeword symbol c_i, define **Condition (*)** as the condition that $c_{i+k_i} \dots c_{i+2}c_{i+1} = \alpha^{q-2}\boldsymbol{\delta}^{k_i-1}$ for some $k_i \in \{1, 2, \dots, x\}$. For example, for a QA-LOCO code with $q = 4$, $m \geq 7$, and $x = 3$, if we have $c_6c_5c_4c_3 = \alpha^2\alpha 1\alpha$ then, $k_5 = 1$, $k_4 = 2$, and $k_3 = 3$.

The following theorem introduces the encoding-decoding rule of QA-LOCO codes. Observe that indexing is straightforward for the case of $m = 1$.

Theorem 2. *Consider a QA-LOCO code $\mathcal{QC}^q_{m,x}$ with $m \geq 2$. Let $\mathbf{c}$ be a QA-LOCO codeword in $\mathcal{QC}^q_{m,x}$. The relation between the lexicographic index $g(\mathbf{c})$ of this codeword and the codeword itself is given by:*

$$g(\mathbf{c}) = \sum_{i=0}^{m-1} a_i (q-1)^{\gamma_i} N_q(i - \gamma_i, x), \tag{6}$$

where γ_i for symbol c_i is computed as follows:

$$\gamma_i = \begin{cases} x - k_i + 1, & k_i \text{ satisfying (*) exists}, \\ 0, & \text{otherwise}. \end{cases} \tag{7}$$

Starting from the left (LMS), parameter $k_i \in \{1, 2, \dots, x\}$, if exists, represents the backward distance in symbols from c_i to the nearest α^{q-2} symbol. Note that $\gamma_{m-1} = 0$.

Proof: We prove Theorem 2 by induction and using the group structure. See the long version [13]. ∎

Example 2. *We use (6) to compute the index of two QA-LOCO codewords in $\mathcal{QC}^4_{6,2}$ ($q = 4$, $m = 6$, and $x = 2$). Using Theorem 1, the required cardinalities are $N_4(-1, 2) \triangleq 3^{-1}$, $N_4(0, 2) \triangleq 1$, $N_4(1, 2) \triangleq 4$, $N_4(2, 2) = 16$, $N_4(3, 2) = 61$, $N_4(4, 2) = 223$, and $N_4(5, 2) = 817$.*

The first codeword is the 334th codeword $011\alpha^2 0\alpha$. This codeword has $a_5 = 0$, $a_4 = a_3 = 1$, $a_2 = 3$, $a_1 = 0$, and $a_0 = 2$. From (7), we get $\gamma_5 = \gamma_4 = \gamma_3 = \gamma_2 = 0$, $\gamma_1 = x = 2$, and $\gamma_0 = x - 1 = 1$. Thus, from (6):

$$\begin{aligned} g(\mathbf{c}) &= \sum_{i=0}^{5} a_i (3^{\gamma_i}) N_4(i - \gamma_i, 2) \\ &= N_4(4, 2) + N_4(3, 2) + 3N_4(2, 2) + 6N_4(-1, 2) \\ &= 223 + 61 + 3 \times 16 + 6 \times 3^{-1} = 334, \end{aligned}$$

which is the correct index.

The second codeword is the 1850th codeword $\alpha 0\alpha^2\alpha^2\alpha 0$. This codeword has $a_5 = 2$, $a_4 = 0$, $a_3 = a_2 = 3$, $a_1 = 2$, and $a_0 = 0$. From (7), we get $\gamma_5 = \gamma_4 = \gamma_3 = 0$, $\gamma_2 = \gamma_1 = x = 2$, and $\gamma_0 = x - 1 = 1$. Thus, from (6):

$$\begin{aligned} g(\mathbf{c}) &= \sum_{i=0}^{5} a_i (3^{\gamma_i}) N_4(i - \gamma_i, 2) \\ &= 2N_4(5, 2) + 3N_4(3, 2) + 27N_4(0, 2) + 18N_4(-1, 2) \\ &= 2 \times 817 + 3 \times 61 + 27 \times 1 + 18 \times 3^{-1} = 1850, \end{aligned}$$

which is the correct index.

Theorem 2 is the key result behind the simple, reconfigurable QA-LOCO encoding and decoding we offer. The theorem provides one-to-one mapping from an index to the corresponding codeword, which is the encoding, and one-to-one demapping from a codeword to the corresponding index, which is the decoding.

IV. Achievable Rates and Comparisons

Bridging is required in order to prevent forbidden patterns from appearing while transitioning from a codeword into the next one [11]. Consider the QA-LOCO code $\mathcal{QC}^4_{5,1}$ ($q = 4$, $m = 5$, and $x = 1$). Assume that we are about to write the following two consecutive codewords on an MLC Flash device (4 levels): $01\alpha\alpha^2\alpha^2$ and $1\alpha^2001$. The stream containing the two consecutive codewords to be written on ten consecutive cells is $01\alpha\alpha^2\underline{\alpha^2 1\alpha^2}001$, and it does contain the forbidden pattern $\alpha^2 1\alpha^2$. Bridging fixes such a problem.

Let $e \triangleq \alpha^{q-2}$. We perform bridging in a QA-LOCO code $\mathcal{QC}^q_{m,x}$ via adding bridging patterns as follows:

1) If the right-most symbol (RMS) of a codeword and the LMS of the next codeword are both α^{q-2}'s, bridge with $\mathbf{e}^x$, i.e., bridge with x consecutive $e \triangleq \alpha^{q-2}$ symbols (x consecutive cells programmed to level $q-1$).
2) Otherwise, bridge with $\mathbf{0}^x$, i.e., bridge with x consecutive 0 symbols (x consecutive unprogrammed cells).

Applying this bridging method to the above scenario results in the following stream $01\alpha\alpha^2\underline{\alpha^2 01\alpha^2}001$, and the forbidden pattern is prevented from appearing across the codewords.

Our bridging is not only simple, but also optimal in the sense that it provides the maximum protection from ICI for the symbols at the edges of QA-LOCO codewords.

Self-clocking is required in order to maintain calibration of the system [3], [12]. Self-clocked constrained codes do not allow long streams of the same symbol to be written (transmitted). Given our bridging method illustrated above for a QA-LOCO code $\mathcal{QC}^q_{m,x}$, even if we repeat a same-symbol codeword consecutive times in a stream, as long as this symbol is in GF(q)$\backslash\{0, \alpha^{q-2}\}$, bridging will guarantee that two transitions to then from a different symbol (0) occur right before each new codeword in the stream. This does not happen with only two same-symbol codewords, which are $\mathbf{0}^m$ and $\mathbf{e}^m$, $e \triangleq \alpha^{q-2}$. Consequently, these are the only codewords we need to remove from $\mathcal{QC}^q_{m,x}$ to achieve self-clocking.

Definition 2. *Let $\mathcal{QC}^q_{m,x}$ be a QA-LOCO code with $q \geq 2$, $m \geq 1$, and $x \geq 1$. A self-clocked QA-LOCO code (CQA-LOCO code) $\mathcal{QC}^{q,\mathrm{c}}_{m,x}$ is obtained from $\mathcal{QC}^q_{m,x}$ as follows:*

$$\mathcal{QC}^{q,\mathrm{c}}_{m,x} \triangleq \mathcal{QC}^q_{m,x} \setminus \{\mathbf{0}^m, \mathbf{e}^m\},\ e \triangleq \alpha^{q-2}. \tag{8}$$

Therefore, the cardinality of the CQA-LOCO code is:

$$N^{\mathrm{c}}_q(m, x) = N_q(m, x) - 2. \tag{9}$$

Now, we are ready to discuss the achievable rates of QA-LOCO codes. Consider a CQA-LOCO code $\mathcal{QC}^{q,\mathrm{c}}_{m,x}$ with cardinality $N^{\mathrm{c}}_q(m, x)$, which is given in (9). The length, in bits, of the messages $\mathcal{QC}^{q,\mathrm{c}}_{m,x}$ encodes is:

$$s^{\mathrm{c}} = \left\lfloor \log_2 N^{\mathrm{c}}_q(m, x) \right\rfloor = \left\lfloor \log_2 \left(N_q(m, x) - 2 \right) \right\rfloor. \tag{10}$$

The input information message is intentionally selected to be a binary message in order to minimize the number of omitted codewords from $\mathcal{QC}^{q,\mathrm{c}}_{m,x}$, and therefore maximize the rate for $q > 2$. The rate of the CQA-LOCO code $\mathcal{QC}^{q,\mathrm{c}}_{m,x}$ then is:

TABLE I
RATES AND NORMALIZED RATES OF VARIOUS CQA-LOCO CODES $\mathcal{QC}^{q,c}_{m,1}$ WITH $q \in \{4, 8, 16, 32\}$ (FOR M/T/Q/P-LC FLASH) AND $x = 1$.

$q=4$			$q=8$			$q=16$			$q=32$		
m	$R^{c}_{\text{QA-LOCO}}$	$R^{c,n}_{\text{QA-LOCO}}$	m	$R^{c}_{\text{QA-LOCO}}$	$R^{c,n}_{\text{QA-LOCO}}$	m	$R^{c}_{\text{QA-LOCO}}$	$R^{c,n}_{\text{QA-LOCO}}$	m	$R^{c}_{\text{QA-LOCO}}$	$R^{c,n}_{\text{QA-LOCO}}$
14	1.8000	0.9000	18	2.7895	0.9298	18	3.7368	0.9342	19	4.7000	0.9400
26	1.8519	0.9260	26	2.8519	0.9506	27	3.8214	0.9554	29	4.8000	0.9600
49	1.9000	0.9500	44	2.9111	0.9704	45	3.8913	0.9728	49	4.8800	0.9760
77	1.9103	0.9552	71	2.9306	0.9769	66	3.9254	0.9813	70	4.9155	0.9831
97	1.9184	0.9592	103	2.9519	0.9840	111	3.9554	0.9888	117	4.9492	0.9898
Capacity	1.9374	0.9687	Capacity	2.9817	0.9939	Capacity	3.9950	0.9987	Capacity	4.9987	0.9997

TABLE II
RATES AND NORMALIZED RATES OF VARIOUS CQA-LOCO CODES $\mathcal{QC}^{q,c}_{m,2}$ WITH $q \in \{4, 8, 16, 32\}$ (FOR M/T/Q/P-LC FLASH) AND $x = 2$.

$q=4$			$q=8$			$q=16$			$q=32$		
m	$R^{c}_{\text{QA-LOCO}}$	$R^{c,n}_{\text{QA-LOCO}}$	m	$R^{c}_{\text{QA-LOCO}}$	$R^{c,n}_{\text{QA-LOCO}}$	m	$R^{c}_{\text{QA-LOCO}}$	$R^{c,n}_{\text{QA-LOCO}}$	m	$R^{c}_{\text{QA-LOCO}}$	$R^{c,n}_{\text{QA-LOCO}}$
20	1.7273	0.8636	22	2.7083	0.9028	24	3.6538	0.9135	25	4.5926	0.9185
38	1.8000	0.9000	32	2.7941	0.9314	34	3.7500	0.9375	36	4.7105	0.9421
57	1.8305	0.9153	52	2.8519	0.9506	51	3.8302	0.9575	56	4.8103	0.9621
76	1.8462	0.9231	73	2.8800	0.9600	73	3.8800	0.9700	77	4.8608	0.9722
96	1.8571	0.9285	108	2.9091	0.9697	100	3.9118	0.9779	108	4.9000	0.9800
Capacity	1.8947	0.9473	Capacity	2.9675	0.9892	Capacity	3.9906	0.9977	Capacity	4.9975	0.9995

$$R^{c}_{\text{QA-LOCO}} = \frac{s^{c}}{m+x} = \frac{\lfloor \log_2 (N_q(m,x) - 2) \rfloor}{m+x}, \tag{11}$$

where $R^{c}_{\text{QA-LOCO}}$ is measured in information bits per coded symbol. We can normalize this rate as follows:

$$R^{c,n}_{\text{QA-LOCO}} = \frac{\lfloor \log_2 (N_q(m,x) - 2) \rfloor}{(m+x)\log_2 q}. \tag{12}$$

Example 3. *Consider the CQA-LOCO code $\mathcal{QC}^{4,c}_{9,1}$ ($q = 4$, $m = 9$, and $x = 1$). From the recursion in Theorem 1, we can reach that $N_4(9, 1) = 191518$. From (11), we get a rate of:*

$$R^{c}_{\text{QA-LOCO}} = \frac{\lfloor \log_2 (191518 - 2) \rfloor}{9+1} = 1.7$$

information bits per coded symbol. From (12), the normalized rate is $1.7/\log_2 4 = 0.85$.

Now, suppose that we want to encode non-binary messages, with their symbols defined over GF(4) here. The rate in this case becomes:

$$\overline{R}^{c}_{\text{QA-LOCO}} = \frac{\lfloor \log_4 (191518 - 2) \rfloor}{9+1} = 0.8.$$

Clearly, this is a significant rate loss compared with the 0.85 normalized rate achieved by encoding binary information messages.[3] *The reason is the higher number of omitted codewords when messages are non-binary.*

Except only the two codewords $\mathbf{0}^m$ and $\mathbf{e}^m$, $e \triangleq \alpha^{q-2}$, all the codewords satisfying the $\mathcal{Q}^q_x$ constraint are in the CQA-LOCO code $\mathcal{QC}^{q,c}_{m,x}$. Additionally, the number of symbols we add for bridging is constant, which is x. Thus, CQA-LOCO codes are **capacity-achieving** codes, i.e., the asymptotic rate of a CQA-LOCO code matches the capacity.

Tables I and II present the rates and the normalized rates of CQA-LOCO codes $\mathcal{QC}^{q,c}_{m,x}$ with $q \in \{4, 8, 16, 32\}$, various values of m, and $x \in \{1, 2\}$. The capacities are given in the last row of each table. We compute the capacity, in information bits per coded symbol, of a $\mathcal{Q}^q_x$-constrained code from a finite-state transition diagram (FSTD) representing the infinitude of a sequence satisfying this $\mathcal{Q}^q_x$ constraint.

[3]CQA-LOCO code rates that are a lot closer to the capacity of a $\mathcal{Q}^4_1$-constrained code are going to be presented in this section.

Table I (resp., Table II) demonstrates that for all values of q, the rates of CQA-LOCO codes with $x = 1$ (resp., $x = 2$) and moderate lengths reach within only 1% (resp., 2%) from capacity. Most important, the tables show that CQA-LOCO codes for all values of q and x achieve normalized rates > 0.95, i.e., rates $> 0.95 \log_2 q$ information bits per coded symbol, except for the case of $q = 4$ and $x = 2$. In other words, significant ICI mitigation in the Flash device can be achieved with only 5% or less redundancy, even later in the lifetime of the device when x can be raised to 2.

The two tables also show the effect of increasing q on the achievable rates. As q increases, the sufficient rate to protect the Flash device increases. Consider quad-level cell (QLC) and penta-level cell (PLC) Flash devices ($q = 16$ and $q = 32$, respectively). For $x = 1$, Table I shows that only about 1.9% (resp., 1.7%) redundancy is enough at length 66 symbols (resp., 70 symbols) for QLC devices (resp., PLC devices). For $x = 2$, Table II shows that only about 3% (resp., 2.8%) redundancy is enough at length 73 symbols (resp., 77 symbols) for QLC devices (resp., PLC devices). Essentially, this is telling that the ICI mitigation via CQA-LOCO codes is coming **almost for free with respect to redundancy**.

Next, we present brief comparisons between QA-LOCO codes and other codes designed for similar goals:

1) It is already not easy to design FSM-based binary constrained codes with rates close to capacity [2], [11]. This task becomes even more complicated in the non-binary domain. Our QA-LOCO codes offer simple encoding and decoding because of their rule, even with $q > 2$.
2) The authors of [1] introduced q-ary lexicographically-ordered RLL (Q-LO-RLL) codes. However, their constraints impose a minimum number of zeros between each two consecutive non-zero symbols. This results in a significant unneeded rate loss if applied for Flash.
3) The authors of [10] introduced enumerative q-ary $\mathcal{Q}^q_1$-constrained codes for Flash. While their codes are capacity-achieving and efficient, QA-LOCO codes offer simpler encoding and decoding compared with their unrank-rank approach. Additionally, the codes in [10] are only for the case of $x = 1$, which means QA-LOCO codes address more general constraints.

4) Non-binary constrained codes are significantly more efficient, rate-wise, compared with binary codes. From [12], the capacity of a binary $\mathcal{A}_1$-constrained code ($x = 1$) is 0.8114. From Table II, even for $q = 4$, a self-clocked QA-LOCO code of length only 20 symbols achieves about 6.4% rate advantage with respect to the above binary capacity, and at $x = 2$ (more ICI mitigation).

V. Algorithms and Reconfigurability

Now, we introduce the encoding and decoding algorithms of QA-LOCO codes, which are based on their encoding-decoding rule (6) of Theorem 2. The algorithms perform the mapping-demapping between an index and the associated codeword, and thus, they are essential for enumerative techniques to offer simplicity. See [14] for a conceptually connected work in the context of multi-dimensional constellations.

The encoding algorithm of our codes is [13, Algorithm 1], and [13, Example 4] illustrates how it works. The decoding algorithm of our codes is Algorithm 1, and Example 2 illustrates how it works.

Algorithm 1 Decoding CQA-LOCO Codes

Inputs: Incoming stream of q-ary CQA-LOCO codewords, in addition to q, m, x, and s^{c}.
Use (4) and (5) to compute $N_q(i, x)$, $i \in \{2, 3, \ldots, m-1\}$.
for each incoming codeword $\mathbf{c}$ of length m **do**
  Initialize $g(\mathbf{c})$ with 0 and c_i with 0 for $i \geq m$.
  Initialize γ_i with 0 for $i \in \{0, 1, \ldots, m-1\}$.
  for $i \in \{m-1, m-2, \ldots, 0\}$ **do** *(in order)*
    for $k_i \in \{1, 2, \ldots, x\}$ **do**
      if $c_{i+k_i} = \alpha^{q-2}$ **then**
        Set $\gamma_i = x - k_i + 1$.
        break. *(exit current loop)*
      end if
    end for
    Set index $= i - \gamma_i$.
    if $c_i \neq 0$ **then** *(same as $a_i \neq 0$)*
      Set $a_i = \mathcal{L}(c_i)$.
      $g(\mathbf{c}) \leftarrow g(\mathbf{c}) + a_i(q-1)^{\gamma_i} N_q(\text{index}, x)$.
    end if
  end for
  Compute $\mathbf{b} = \text{binary}(g(\mathbf{c}) - 1)$, which has length s^{c}.
  Ignore the next x bridging symbols.
end for
Output: Outgoing stream of binary messages.

In order to reduce complexity, all terms containing multiplications in [13, Algorithm 1] and Algorithm 1, e.g., $a_i(q-1)^{\gamma_i} N_q(\text{index}, x)$, are not computed at runtime. This increases the storage overhead. However, the gain is that the complexity of both algorithms is still mainly governed by the adder size that will perform the comparisons/subtractions and additions. The adder size is itself the message length s^{c}. For example, to achieve a rate of 1.8519 information bits per coded symbol using a CQA-LOCO code with $q = 4$ and $x = 1$, adders of size $1.8519 \times (26 + 1) = 50$ bits are needed (see Table I).

A detailed storage and complexity analysis is in [13].

A Flash device with q levels per cell has $\log_2 q$ pages. In general, the Flash industry prefers to process different pages independently in order to increase access speed. One idea to achieve this goal is to apply the QA-LOCO code only on the parity part of the EC low-density parity-check (LDPC) code as we did in [11] for MR systems. In particular, the idea is to reserve few word lines for parity. We encode parity bits via a QA-LOCO code into symbols over GF(q) before writing them on the parity word lines. The $\mathcal{Q}_x^q$ constraint should be satisfied over all bit lines across the parity word lines. While reading, we read the parity word lines first. Next, the parity bits are decoded via the QA-LOCO decoder. The LDPC decoder then operates independently on the $\log_2 q$ pages to retrieve the $\log_2 q$ codewords for each word line. High performance LDPC codes for Flash can be designed as in [15] and [16].

The fact that the encoding and decoding of QA-LOCO codes are performed through simple adders enables reconfigurability. All that is needed to reconfigure a QA-LOCO code, i.e., change the code parameters such that more (or even different) constraints are supported, is to change the cardinalities that are inputs to the adders at both encoding and decoding sides such that the encoding-decoding rule in (6) supports the new constraints. As the Flash device ages, charges propagate during programming with higher rates and reach further non-adjacent cells. Thus, while QA-LOCO codes with $x = 1$ are sufficient when the device is fresh, reconfiguring to QA-LOCO codes with $x > 1$, i.e., forbidding more patterns, is needed such that the device keeps functioning reliably late in its lifetime.

Aided by machine learning, errors before the LDPC decoder can be collected to identify the set of error-prone patterns that should be forbidden at different stages of the Flash device lifetime. Once this set is found to be bigger than the currently supported set by the QA-LOCO code, we propose to respond via reconfiguring the QA-LOCO code to support the new set as illustrated in the previous paragraph. Therefore, machine learning and reconfigurable constrained codes, along with high performance EC codes, can help increase the lifetime of modern Flash devices significantly, and therefore support the evolution of QLC and PLC Flash memories.

VI. Conclusion

We introduced capacity-achieving q-ary asymmetric LOCO codes (QA-LOCO codes) for Flash devices with any number, q, of levels per cell. We partitioned the codewords of a QA-LOCO code into groups, which we used to recursively compute the cardinality. We devised an encoding-decoding rule for QA-LOCO codes to map from index to codeword and vice versa, which is the key result behind the simple encoding and decoding of these codes. We introduced the achievable rates of QA-LOCO codes, and showed that they need 5% or less redundancy to protect the device. For QLC and PLC devices, we demonstrated that ICI mitigation almost comes for free with respect to redundancy. We presented the encoding and decoding algorithms, and provided an analysis for the storage and complexity growth with q. We suggest that machine learning and reconfigurable QA-LOCO codes (with EC codes) can significantly increase the lifetime of modern Flash devices.

Acknowledgment

This research was supported by NSF under Grant CCF 1717602 and by AFOSR under Grant FA 9550-17-1-0291.

REFERENCES

[1] 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.

[2] P. Siegel, "Recording codes for digital magnetic storage," *IEEE Trans. Magn.*, vol. 21, no. 5, pp. 1344–1349, Sep. 1985.

[3] 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.

[4] R. Karabed and P. H. Siegel, "Coding for higher-order partial-response channels," in *Proc. SPIE Int. Symp. Voice, Video, and Data Commun.*, M. R. Raghuveer, S. A. Dianat, S. W. McLaughlin, and M. Hassner, Eds., Philadelphia, PA, Oct. 1995, vol. 2605, pp. 115–126.

[5] 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.

[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] M. Qin, E. Yaakobi, and P. H. Siegel, "Constrained codes that mitigate inter-cell interference in read/write cycles for flash memories," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 5, pp. 836–846, Apr. 2014.

[8] S. Kayser and P. H. Siegel, "Constructions for constant-weight ICI-free codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Honolulu, HI, USA, Jun.–Jul. 2014, pp. 1431–1435.

[9] 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.

[10] 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.

[11] A. Hareedy and R. Calderbank, "LOCO codes: Lexicographically-ordered constrained codes," *IEEE Trans. Inf. Theory*, to be published, doi: 10.1109/TIT.2019.2943244.

[12] 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.

[13] A. Hareedy, B. Dabak, and R. Calderbank, "Managing device lifecycle: Reconfigurable constrained codes for M/T/Q/P-LC Flash memories," Jan. 2020. [Online]. Available: https://arxiv.org/abs/2001.02325

[14] 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.

[15] A. Hareedy, C. Lanka, and L. Dolecek, "A general non-binary LDPC code optimization framework suitable for dense Flash memory and magnetic storage," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 9, pp. 2402–2415, Sep. 2016.

[16] H. Esfahanizadeh, A. Hareedy, and L. Dolecek, "Finite-length construction of high performance spatially-coupled codes via optimized partitioning and lifting," *IEEE Trans. Commun.*, vol. 67, no. 1, pp. 3–16, Jan. 2019.