Row-by-Row Coding Schemes for Inter-Cell Interference in Flash Memory

Sarit Buzaglo and Paul H. Siegel, *Fellow, IEEE*

***Abstract*— Inter-cell interference (ICI) is a significant cause of errors in flash memories. In single-level cell (SLC) flash memory, ICI arises when $1\,0\,1$ patterns are programmed either in the horizontal or vertical directions. Since data pages are written sequentially in horizontal wordlines, one can mitigate the effects of horizontal ICI by applying conventional constrained codes that forbid the $1\,0\,1$ pattern. This approach does not address the problem of vertical ICI, however. In this paper, a row-by-row coding technique that eliminates vertical $1\,0\,1$ patterns while preserving the sequential wordline programming order is presented. This scheme, though efficient, necessarily suffers a rate loss of almost 20%. We therefore propose another coding scheme, combining a weak constraint on vertical $1\,0\,1$ patterns with a systematic error-correcting code, that can mitigate vertical ICI errors while achieving a higher overall coding rate, provided that the vertical ICI error probability is sufficiently small. Some extensions for multi-level cell (MLC) flash memory are discussed as well.**

***Index Terms*— Decoding, encoding, error-correction coding, Markov processes, memories.**

I. Introduction

FLASH memories are, by far, the most important type of non-volatile memory (NVM) in use today. Their low cost and steadily increasing storage capacity make them attractive for many NVM applications. As memory cell size decreases, inter-cell interference (ICI) can severely degrade the device performance. In the simplest model of ICI, parasitic capacitance can cause an undesirable increase in the voltage level of a *victim* cell when high voltages are applied to some of its neighboring cells [21]. This phenomenon occurs in single-level cell (SLC) flash memory, where each cell stores one bit, and is particularly severe when programming multi-level cell (MLC) flash memory [9], [21], in which a cell level is represented by two bits that are stored in two logical units of programming called *pages*. It becomes even more pronounced in the recent designs of 3-dimensional flash [20], [27], [34].

ICI-induced errors are data dependent and correlated with the data patterns in the neighborhood of the victim cell, in both the horizontal (wordline) and vertical (bitline) directions [2], [3], [33]. In fact, experimental results in [33] indicate that bitline ICI can be even more detrimental than wordline ICI in MLC flash, due in part to the row-by-row programming protocol which requires wordlines to be programmed in a sequential order.

A number of approaches have been proposed to combat ICI effects, including the use of constrained codes that prevent the appearance of ICI-prone cell-level patterns in both one-dimensional (1D) and two-dimensional (2D) configurations; see, for example [2], [3], [6], [18], [19], [28]. Whereas the implementation of 1D constrained codes in wordline pages to address wordline ICI is relatively straightforward, for 2D constraints it remains a challenge to design efficient, fixed-rate encoding and decoding algorithms that are compatible with the conventional sequential programming and reading of independent wordlines. One notable example of 2D row-by-row constrained coding for flash memories appears in [3], but the encoding is variable-rate.

The main goal of this work is the design of row-by-row bitline ICI-mitigating coding techniques for flash memory. This is accomplished by allowing the encoding and decoding algorithms to read the previous two wordlines when writing or reading a particular wordline. For SLC flash memory, we present a row-by-row coding scheme which eliminates the ICI-inducing vertical $1\,0\,1$ patterns along bitlines, while still asymptotically achieving the capacity of the corresponding *ICI constraint,* $C_{ICI} \approx 0.8114$. The proposed scheme is an embodiment of the design method in [15] and [31], where M-track parallel encoding for general 1D constraints is considered. The method is extended for MLC flash memory, as well, to produce coding schemes which prevent a certain ICI error-prone pattern from appearing horizontally, vertically, or in both directions, and which are compatible with the independent programming of pages protocol.

The rate loss incurred by eliminating *all* $1\,0\,1$ patterns in the vertical direction reduces storage capacity. We therefore propose an alternative coding approach, in the spirit of *weakly constrained codes*, suggested in [16] and also studied in [10] and [11], that allows a nonzero probability of occurrence of vertical $1\,0\,1$ patterns and uses a systematic error-correcting code to correct the resulting ICI errors. The rate of this scheme can be much higher than the capacity of the 1D ICI constraint, provided that the probability of bitline ICI-induced errors is not too large. On top of its potential to increase capacity, the suggested scheme becomes handy when other types of errors may occur, since it combines a

Manuscript received October 6, 2016; revised February 27, 2017; accepted April 9, 2017. Date of publication April 24, 2017; date of current version October 16, 2017. This work was supported in part by NSF Grants CCF-1405119 and CCF-1619053, the ISEF Foundation, and the Weizmann Institute of Science-National Postdoctoral Award Program for Advancing Women in Science. This paper was presented at the 2015 IEEE International Symposium on Information Theory, Hong Kong, June 2015. The associate editor coordinating the review of this paper and approving it for publication was L. Dolecek. *(Corresponding author: Sarit Buzaglo.)*

The authors are with the Center for Memory and Recording Research, University of California, San Diego, La Jolla, CA 92093-0401 USA (e-mail: sbuzaglo@ucsd.edu; psiegel@ucsd.edu).

Digital Object Identifier 10.1109/TCOMM.2017.2696530

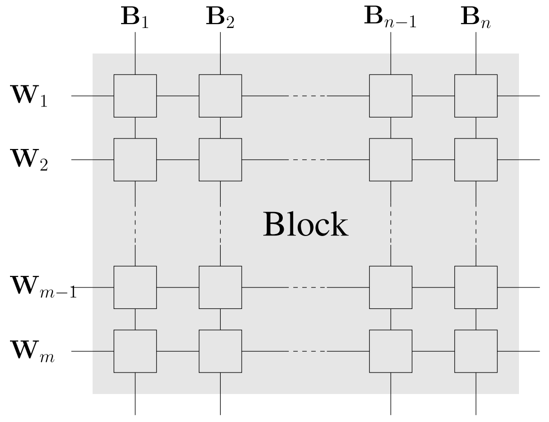

Fig. 1. Structure of an SLC flash memory block.

bitline ICI mitigating coding scheme with an error-correcting code.

The rest of this paper is organized as follows. In Section II we introduce basic notations and definitions used throughout the paper. In Section III we present a row-by-row coding scheme that forbids the vertical 1 0 1 pattern along bitlines for SLC flash memory. This coding scheme is combined with error-correcting codes in Section IV to mitigate ICI errors and achieve higher rates. The frame-error-rate (FER) of this approach is estimated via a simulation result. Extensions for MLC flash are discussed in Section V. We conclude in Section VI.

II. Preliminaries

In this section we present some of the basic definitions and notations used throughout the paper for the design of coding schemes that mitigate the inter-cell interference error mechanism in SLC flash memory. In subsection II-A we introduce the relevant notation and definitions regarding coding schemes for flash memory. In subsection II-B we present notation and definitions for the ICI constraint that avoids the pattern 1 0 1.

A. Coding Schemes for Flash Memory

Denote by $[n]$ the set of n integers $\{1, 2, \ldots, n\}$. For two integers $a, b \in \mathbb{Z}$, where $a < b$, denote by $[a, b]$ the set of $b-a+1$ integers $\{a, a+1, \ldots, b\}$. We use upper-case letters to denote random variables and lower-case letters to denote their realizations. Moreover, we use boldface to denote vectors, both of random variables and of deterministic values. For example, $\mathbf{X} = X_1X_2$ is a vector of two random variables X_1 and X_2, and its realization $\mathbf{x} = x_1x_2$ is a vector of two deterministic values x_1 and x_2, which are the realizations of X_1 and X_2, respectively.

We represent a block of SLC flash memory as an $m \times n$ array of cells, composed of m *wordlines* of length n (see Figure 1), where each cell stores a symbol in $\{0, 1\}$. For $i \in [m]$ and $j \in [n]$, the cell in the ith row and the jth column of the array is denoted by $W_{i,j}$. We consider $W_{i,j}$ as a binary random variable, where its realization, denoted by $w_{i,j}$, is the symbol stored in the cell. The *programming state* of $W_{i,j}$, denoted by $w_{i,j}^{(ps)}$, is the (binary) symbol to which $W_{i,j}$ was programmed. In the event of an error in the cell $W_{i,j}$, the stored symbol and the programming state of the cell do not agree, i.e., $w_{i,j} \neq w_{i,j}^{(ps)}$. The ith *wordline* is the ith row of the array and it is denoted by $\mathbf{W}_i \overset{\text{def}}{=} W_{i,1}W_{i,2}\ldots W_{i,n}$, while its realization is denoted by $\mathbf{w}_i \overset{\text{def}}{=} w_{i,1}w_{i,2}\ldots w_{i,n}$ and its programming state is denoted by $\mathbf{w}_i^{(ps)} \overset{\text{def}}{=} w_{i,1}^{(ps)}w_{i,2}^{(ps)}\ldots w_{i,n}^{(ps)}$. Similarly, the jth *bitline* is the jth column of the array and it is denoted by $\mathbf{B}_j \overset{\text{def}}{=} W_{1,j}W_{2,j}\ldots W_{m,j}$, while its *realization* is denoted by $\mathbf{b}_j \overset{\text{def}}{=} w_{1,j}w_{2,j}\ldots w_{m.j}$ and its programming state is denoted by $\mathbf{b}_j^{(ps)} \overset{\text{def}}{=} w_{1,j}^{(ps)}w_{2,j}^{(ps)}\ldots w_{m.j}^{(ps)}$.

Given a vector $\mathbf{X} = X_1X_2\ldots X_\ell$ of random variables (respectively, a vector $\mathbf{x} = x_1x_2\ldots x_\ell$) and a set of indices $J = \{j_1, j_2, \ldots, j_k\}$, where $1 \leq j_1 < j_2 < \cdots < j_k \leq \ell$, denote the *restriction* of $\mathbf{X}$ to J (respectively, of $\mathbf{x}$ to J) by $\mathbf{X}|_J \overset{\text{def}}{=} X_{j_1}X_{j_2}\ldots X_{j_k}$ (respectively, by $\mathbf{x}|_J \overset{\text{def}}{=} x_{j_1}x_{j_2}\ldots x_{j_k}$). In particular, for every $i \in [m]$ and for every set of indices $J = \{j_1, j_2, \ldots, j_k\}$, where $1 \leq j_1 < j_2 < \cdots < j_k \leq n$, the restriction of $\mathbf{W}_i$ to J is defined as $\mathbf{W}_i|_J \overset{\text{def}}{=} W_{i,j_1}W_{i,j_2}\ldots W_{i,j_k}$.

We assume throughout that wordlines are programmed sequentially in a monotone increasing order, i.e., for every $1 \leq i_1 < i_2 \leq m$, $\mathbf{W}_{i_1}$ is programmed before $\mathbf{W}_{i_2}$. We will refer to a coding scheme in which the wordlines are programmed in such a monotone order as a *row-by-row coding scheme*. A binary row-by-row coding scheme $\mathcal{C}$ has a *fixed rate* R if every row can store any message out of 2^{Rn} possible messages. In Section III we will design a fixed-rate row-by-row coding scheme[1] such that the pattern 1 0 1 does not appear along bitlines. This constraint on the bitlines is motivated by the ICI phenomenon, and therefore is called the *bitline ICI constraint*. In general, a binary sequence $\mathbf{x} = x_1x_2\ldots x_\ell$ is said to satisfy the *ICI constraint* if $x_{i-2}x_{i-1}x_i \neq 1\,0\,1$, for all $i \in [3, \ell]$. We denote by S the set of all binary sequences of finite length that satisfy the ICI constraint. A coding scheme $\mathcal{C}$ for an SLC flash memory is called a *wordline ICI constrained code* if for every $i \in [m]$, $\mathbf{w}_i^{(ps)} \in S$. Similarly, a coding scheme $\mathcal{C}$ for an SLC flash memory is called a *bitline ICI constrained code* if for every $j \in [n]$, $\mathbf{b}_j^{(ps)} \in S$. A *wordline ICI error* in the cell $W_{i,j}$ is the event that the cells $W_{i,j-1}$, $W_{i,j}$, and $W_{i,j+1}$ were programmed to 1, 0, and 1, respectively, and due to the ICI phenomenon, $W_{i,j}$ stores the symbol 1, i.e., $w_{i,j-1}^{(ps)}w_{i,j}^{(ps)}w_{i,j+1}^{(ps)} = 1\,0\,1$, while $w_{i,j-1}w_{i,j}w_{i,j+1} = 1\,1\,1$. Similarly, a *bitline ICI error* in the cell $W_{i,j}$ is the event that the cells $W_{i-1,j}$, $W_{i,j}$, and $W_{i+1,j}$ were programmed to 1, 0, and 1, respectively, and due to the ICI phenomenon, $W_{i,j}$ stores the symbol 1, i.e., $w_{i-1,j}^{(ps)}w_{i,j}^{(ps)}w_{i+1,j}^{(ps)} = 1\,0\,1$, while $w_{i-1,j}w_{i,j}w_{i+1,j} = 1\,1\,1$. We will assume throughout the paper that the only sources of errors are wordline ICI or bitline ICI, and specify in each section the sources of errors in consideration. Notice that, under the assumption that only bitline ICI errors may occur, bitline ICI constrained codes prevent the occurrence of errors entirely.

[1]The first two rows of our row-by-row coding scheme may have different rate than the others, yet for the ease of terminology, we refer to it as a fixed-rate coding scheme.

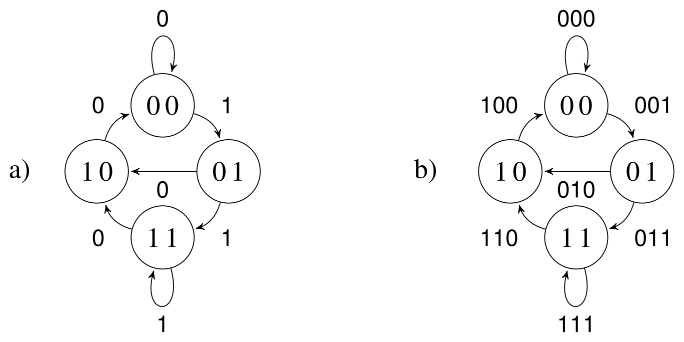

Fig. 2. (a) The edge-labeled graph G representing the ICI constraint. (b) The edge in G from xy to yz is represented by xyz, where z is also its label.

B. The ICI Constraint

The rate of a bitline ICI constrained code is clearly upper bounded by the capacity of the ICI constraint defined by

$$Cap(ICI) \overset{\text{def}}{=} \lim_{n\to\infty} \frac{\log_2 |S \cap \{0,1\}^n|}{n}.$$

There is a one-to-one correspondence between S and the sequences produced by reading the labels of paths in the directed edge-labeled graph $G(V, E, L)$ shown in Fig. 2 (a), where $V = \{0,1\}^2$, $E = \{0,1\}^3 \setminus \{1\,0\,1\}$, and for every edge $e = xyz \in E$, e is an edge from xy to yz (see Fig. 2 (b)) labeled by z. The *adjacency matrix* of the graph G is given by

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix},$$

where $A_{i,j}$ represents the number of edges from the vertex $u = xy$ to the vertex $v = wz$ in the graph G, $i = 2x + y$ and $j = 2w + z$. Let $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ be the eigenvalues of A. Notice that $A_{2x+y,2w+z} = 0$ if $y \neq w$. The *spectral radius* of A is defined by $\lambda \overset{\text{def}}{=} \max\{|\lambda_1|, |\lambda_2|, |\lambda_3|, |\lambda_4|\}$. The spectral radius λ is an eigenvalue of A and it admits a positive right eigenvector as well as a positive left eigenvector, as guaranteed by the Perron-Frobenius Theorem [13, Ch. 8].

There is a simple expression for the capacity of this constraint in terms of λ, namely $Cap(ICI)$ is equal to $\log_2 \lambda \approx 0.8114$ (see, for example [23]). Alternatively, $Cap(ICI)$ can be represented in terms of *stationary Markov chains* on the graph G. A *Markov chain* on G is simply a probability mass function $\mathcal{P}$ on the edges of G, i.e., a mapping $\mathcal{P} : E \to \mathbb{R}$ such that $\mathcal{P}(xyz) \geq 0$, for all $xyz \in E$, and $\sum_{xyz\in E} \mathcal{P}(xyz) = 1$. The *state probability vector* $\pi^T = (\pi_0, \pi_1, \pi_2, \pi_3)$ of the Markov chain $\mathcal{P}$ on G is defined by $\pi_{2x+y} \overset{\text{def}}{=} \pi(xy)$, where $\pi(xy)$ is the marginal probability $\sum_{z:\ xyz\in E} \mathcal{P}(xyz)$, for all $xy \in V$. The *transition matrix* associated with $\mathcal{P}$ is the 4×4 matrix Q, such that $Q_{2x+y,2y+z} \overset{\text{def}}{=} Q(z|xy)$, where $Q(z|xy)$ is the conditional probability defined by $Q(z|xy) \overset{\text{def}}{=} \mathcal{P}(xyz)/\pi(xy)$ if $\pi(xy) > 0$ and $Q(z|xy) \overset{\text{def}}{=} 0$, otherwise. As for the adjacency matrix A, $Q_{2x+y,2w+z} \overset{\text{def}}{=} 0$ if $y \neq w$. We will use the notations $Q_{2x+y,2y+z}$ and $Q(z|xy)$ (respectively, π_{2x+y} and $\pi(xy)$) interchangeably. A Markov chain $\mathcal{P}$ is called *stationary* if $\pi^T Q = \pi^T$, or equivalently if $\pi(xy) = \sum_{z:\ xyz\in E} \mathcal{P}(xyz) = \sum_{z:\ zxy\in E} \mathcal{P}(zxy)$, for all $xy \in V$. The *entropy rate* of $\mathcal{P}$ is defined by

$$H(\mathcal{P}) = -\sum_{xyz\in E} \mathcal{P}(xyz) \log_2\ Q(z|xy).$$

Note that if $Q(z|xy) = 0$ then $\mathcal{P}(xyz) = 0$ and we regard $0 \log_2 0$ as zero.

From [23] we have that

$$Cap(ICI) = \sup_{\mathcal{P}} H(\mathcal{P}),$$

where the supremum is taken over all the stationary Markov chains on G. Moreover, a capacity-achieving stationary Markov chain $\widehat{\mathcal{P}}$ on G can be obtained as follows. Let $\mathbf{v}^T$ and $\mathbf{u}$ be positive left and right eigenvectors of A associated with λ, respectively, normalized such that $\mathbf{v}^T\mathbf{u} = 1$. For all $xyz \in E$,

$$\widehat{\mathcal{P}}(xyz) \overset{\text{def}}{=} \frac{u_j v_i A_{i,j}}{\lambda}, \quad i = 2x + y,\ j = 2y + z. \tag{1}$$

The state probability vector for $\widehat{\mathcal{P}}$, $\widehat{\pi}$, is given by

$$\widehat{\pi}_i \overset{\text{def}}{=} u_i v_i, \quad i \in [0, 3], \tag{2}$$

and the transition matrix for $\widehat{\mathcal{P}}$, $\widehat{Q}$, is given by

$$\widehat{Q}_{i,j} \overset{\text{def}}{=} \frac{A_{i,j} u_j}{\lambda u_i}, \quad \forall\ i, j \in [0, 3]. \tag{3}$$

For our coding schemes, we need one more definition on Markov chains. A Markov chain $\mathcal{P}$ on G is called *n-integral* if, for all $xyz \in E$, we have that $\mathcal{P}(xyz)n$ is an integer.

III. Row-by-Row Bitline ICI Constrained Codes

The goal of this section is to design a row-by-row bitline ICI coding scheme with a fixed rate. As mentioned above, the rate of such a coding scheme is upper bounded by $Cap(ICI)$. The presented construction produces coding schemes of rates that are arbitrarily close to $Cap(ICI)$, provided that the block dimensions n and m are large enough. The key idea behind the construction is simply that, when programming a given wordline, the encoder takes into account the two previously programmed wordlines. This approach is consistent with the practical requirement for wordlines to be programmed sequentially, in a row-by-row manner. It is also compatible with the usual practice of programming all the cells of a wordline simultaneously. We remark that the design methodology described in this section is closely related to the approach proposed in [15] and [31] for constructing row-by-row coding schemes for general constraints. It was shown in [15] that there exist such coding schemes that achieve the capacity of a constraint, and an explicit construction, based on constant-weight codes, was given in [31]. We essentially apply the basic ideas of the construction in [31] to the ICI constraint, with some simplifications.

Specifically, our code design procedure uses an n-integral stationary Markov chain $\mathcal{P}$, i.e., $\mathcal{P}(xyz)n$ is an integer, for all $xyz \in E$, to construct a code $\mathcal{C}^{(wl)}$ of asymptotic rate $H(\mathcal{P})$. A simple technique to obtain an n-integral stationary Markov chain on G from an arbitrary stationary Markov chain on G with a negligible loss in the entropy rate is described in Appendix A. The code $\mathcal{C}^{(wl)}$ is a cartesian product of constant-weight codes with parameters that are determined by the Markov chain $\mathcal{P}$. Our coding scheme $\mathcal{C}$ will program every wordline (with the exception of the first two wordlines) to a codeword of $\mathcal{C}^{(wl)}$. Moreover, it will have the property that $|\{j : w^{(ps)}_{i-1,j} w^{(ps)}_{i,j} = xy\}| = \pi(xy)n$, for every $i \in [2, m]$, and $xy \in V$. This stationarity property of the code will be crucial for the encoding process. The code will also imitate the statistics of patterns of the form $xyz \in E$ in a sequence that is created by reading the labels of a random path in G, according to the Markov chain $\mathcal{P}$. Concretely, it will satisfy the equality $|\{j : w^{(ps)}_{i-2,j} w^{(ps)}_{i-1,j} w^{(ps)}_{i,j} = xyz\}| = \mathcal{P}(xyz)n$, for every $i \in [3, m]$ and $xyz \in E$. In particular, $|\{j : w^{(ps)}_{i-2,j} w^{(ps)}_{i-1,j} w^{(ps)}_{i,j} = 1\,0\,1\}| = 0$, for every $i \in [3, m]$.

Let $p(x) = \pi(x0) + \pi(x1)$ be the probability of an initial state to be of the form $x*$, where $*$ stands for an arbitrary symbol in $\{0, 1\}$, according to the stationary Markov chain $\mathcal{P}$. A binary *constant-weight code* of length n and weight w is a subset of $\{0, 1\}^n$ which consists only of codewords of *Hamming weight* w, where the Hamming weight of $\mathbf{x} = x_1 x_2 \ldots x_n \in \{0, 1\}^n$ is defined by

$$w_H(\mathbf{x}) \overset{\text{def}}{=} |\{j : x_j \neq 0\}|.$$

Concretely, $w_H(\mathbf{x})$ counts the number of ones in $\mathbf{x}$. Denote by $\mathbb{C}(n, w)$ the set of all binary sequences of length n and Hamming weight w, i.e.,

$$\mathbb{C}(n, w) \overset{\text{def}}{=} \{\mathbf{x} \in \{0, 1\}^n : w_H(\mathbf{x}) = w\}.$$

For the encoding process, we will need the following three codes, defined in terms of constant-weight codes:

$$\begin{aligned}
\mathcal{C}^{(1)} &\overset{\text{def}}{=} \mathbb{C}(n, p(1)n) \\
\mathcal{C}^{(2)} &\overset{\text{def}}{=} \mathbb{C}(p(0)n, \pi(01)n) \times \mathbb{C}(p(1)n, \pi(11)n) \\
\mathcal{C}^{(wl)} &\overset{\text{def}}{=} \mathbb{C}(\pi(00)n, \mathcal{P}(001)n) \times \mathbb{C}(\pi(01)n, \mathcal{P}(011)n) \\
&\quad \times \mathbb{C}(\pi(10)n, \mathcal{P}(101)n) \times \mathbb{C}(\pi(11)n, \mathcal{P}(111)n). \quad (4)
\end{aligned}$$

The first wordline will be programmed by the constant weight code $\mathcal{C}^{(1)}$. Then, the second wordline is partitioned into two parts of sizes $p(0)n$ and $p(1)n$, respectively. The first part contains the cells below the $p(0)n$ cells of the first wordline that were programmed to 0. This part is programmed by the $\mathbb{C}(p(0)n, \pi(01)n)$ portion of $\mathcal{C}^{(2)}$. Similarly, the second part contains the cells below the $p(1)n$ cells of the first wordline that were programmed to 1 and is programmed by the $\mathbb{C}(p(1)n, \pi(11)n)$ portion of $\mathcal{C}^{(2)}$. The remaining wordlines are programmed by the code $\mathcal{C}^{(wl)}$. For every $i \in [3, m]$, the cells of the ith wordline are partitioned into four parts, according to the programmed states of the two cells above them in $\mathbf{W}_{i-2}$ and $\mathbf{W}_{i-1}$. The stationarity of $\mathcal{P}$ guarantees that the part that corresponds to cells for which the two cells above them were programmed to xy is of size $\pi(xy)n$. This part is programmed by the $\mathbb{C}(\pi(xy)n, \mathcal{P}(xy1)n)$ portion of the code $\mathcal{C}^{(wl)}$. Figure 3 describes the architecture of this encoding process. Notice that, since $\mathcal{P}(101) = 0$, it follows that $\mathbb{C}(\pi(10)n, \mathcal{P}(101)n) = \{\mathbf{0}\}$, where $\mathbf{0}$ is the all-zero vector.

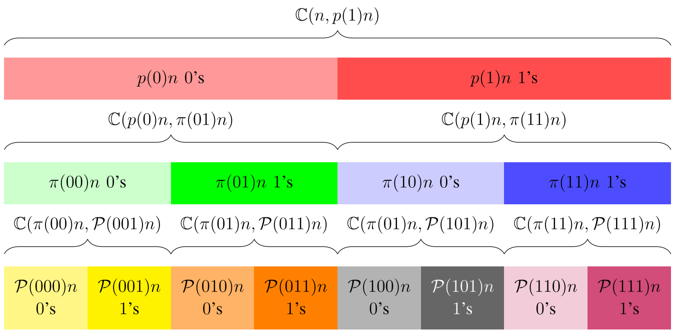

Fig. 3. The architecture of the encoding process. A codeword of $\mathbb{C}(p(x)n, \pi(x1)n)$ is stored in the positions in which x appears in $\mathbb{C}(n, p(1)n)$. Similarly, a codeword of $\mathbb{C}(\pi(xy)n, \mathcal{P}(xy1)n)$ is stored in the positions in which both x appears in $\mathbb{C}(n, p(1)n)$ and y appears in $\mathbb{C}(p(x)n, \pi(x1)n)$.

In the next section we will allow $\mathcal{P}(101)$ to be positive, in which case the constant-weight code $\mathbb{C}(\pi(10)n, \mathcal{P}(101)n)$ will no longer be a trivial code.

For $k \in \{1, 2\}$, let $\mathcal{E}^{(k)} : \mathbb{Z}_{|\mathcal{C}^{(k)}|} \to \mathcal{C}^{(k)}$ be an encoder for $\mathcal{C}^{(k)}$ and let $\mathcal{D}^{(k)} : \mathcal{C}^{(k)} \to \mathbb{Z}_{|\mathcal{C}^{(k)}|}$ be the corresponding decoder. Likewise, let $\mathcal{E}^{(wl)} : \mathbb{Z}_{|\mathcal{C}^{(wl)}|} \to \mathcal{C}^{(wl)}$ and $\mathcal{D}^{(wl)} : \mathcal{C}^{(wl)} \to \mathbb{Z}_{|\mathcal{C}^{(wl)}|}$ be an encoder and its corresponding decoder for $\mathcal{C}^{(wl)}$.

Remark 1: We remark that encoders and decoders for $\mathcal{C}^{(k)}$, $k \in \{1, 2\}$ and for $\mathcal{C}^{(wl)}$ can be readily derived from encoders and decoders of constant-weight codes. Encoders and decoders for constant-weight codes were proposed in [17] and later improved in [31]. Both these papers present efficient encoders that are based on enumerative encoding [7] and suffer from negligible loss in coding rates. In particular, the running time of the encoding and decoding procedures proposed in [31] is $O(n \log_2^2 n)$, where n is the length of the code.

The programming process is formulated in Algorithm 1. Note that we assume no other error sources in this section but the bitline ICI. To show that this programming process produces a row-by-row bitline ICI constrained coding scheme we first prove its validity in the next two lemmas that assume no error sources, including the bitline ICI.

Lemma 1: If there are no error sources, including the bitline ICI, then the programming process described in Algorithm 1 is properly defined, i.e., the following hold.

1) *For all $x \in \{0, 1\}$, the size of J_x, defined in line 10 of Algorithm 1 is equal to the length of $\mathbf{c}_x$, $p(x)n$, where $\mathbf{c}_x$ is defined in line 8 of Algorithm 1.*
2) *For every $i \in [3, m]$ and for all $x, y \in \{0, 1\}$, the size of J_{xy}, defined in line 17 of Algorithm 1 for the programming of the ith wordline, is equal to the length of $\mathbf{c}_{xy}$, $\pi(xy)n$, where $\mathbf{c}_{xy}$ is defined in line 15 of Algorithm 1.*

Proof: To prove item (1), let $x \in \{0, 1\}$. Since we assume no error sources, it follows that $\mathbf{w}_1 = \mathbf{w}^{(ps)}_1 \in \mathbb{C}(n, p(1)n)$ and $|J_x| = |\{j : w^{(ps)}_{1,j} = x\}| = p(x)n$.

For ease of presentation we define $J^{(i)}_{xy} = J_{xy}$, where J_{xy} is defined in line 17 of Algorithm 1 for the programming of the ith wordline. We will prove item (2) by induction on i. Since we assume no error sources, it follows that for all

Algorithm 1 Row-by-Row Bitline ICI Constrained Coding Scheme: Programming Process

```
1: **Initialize:** $i \leftarrow 1$.
2: **Input:** A message $M$ to be programmed into the current wordline $\mathbf{W}_i$.
3: **if** $i = 1$ **then**                                   $\triangleright\ M \in \mathbb{Z}_{|C^{(1)}|}$
4:     Set $\mathbf{c} \leftarrow \mathcal{E}^{(1)}(M)$.
5:     Program $\mathbf{c}$ into $\mathbf{W}_1$.
6:     Set $i \leftarrow 2$.
7: **else if** $i = 2$ **then**                              $\triangleright\ M \in \mathbb{Z}_{|C^{(2)}|}$
8:     Set $(\mathbf{c}_0, \mathbf{c}_1) \leftarrow \mathcal{E}^{(2)}(M)$.
9:     **for all** $x \in \{0, 1\}$ **do**
10:        Set $J_x \leftarrow \{j \in [n] : w_{1,j} = x\}$.
11:        Program $\mathbf{c}_x$ into $\mathbf{W}_2|_{J_x}$.
12:    **end for**
13:    Set $i \leftarrow 3$.
14: **else if** $i \geq 3$ **then**                          $\triangleright\ M \in \mathbb{Z}_{|C^{(wl)}|}$
15:    Set $(\mathbf{c}_{00}, \mathbf{c}_{01}, \mathbf{c}_{10}, \mathbf{c}_{11}) \leftarrow \mathcal{E}^{(wl)}(M)$.
16:    **for all** $x, y \in \{0, 1\}$ **do**
17:        Set $J_{xy} \leftarrow \{j \in [n] : w_{i-2,j} w_{i-1,j} = xy\}$.
18:        Program $\mathbf{c}_{xy}$ into $\mathbf{W}_i|_{J_{xy}}$.
19:    **end for**
20:    Set $i \leftarrow i + 1$.
21: **end if**
```

$i \in [3, m]$, $J_{xy}^{(i)} = \{j \in [n] : w_{i-2,j}^{(ps)} w_{i-1,j}^{(ps)} = xy\}$. For the basis of the induction, we will prove the case $i = 3$. Let $x, y \in \{0, 1\}$. By the definition of $J_{xy}^{(3)}$ it follows that $J_{xy}^{(3)} = \{j \in J_x : w_{2,j}^{(ps)} = y\}$ and from lines 8 and 11 of Algorithm 1 we have that $\mathbf{w}_2^{(ps)}|_{J_x} \in \mathbb{C}(p(x)n, \pi(x1)n)$. Hence, there are exactly $\pi(xy)n$ indices j for which $w_{1,j}^{(ps)} w_{2,j}^{(ps)} = xy$.

For the induction hypothesis we assume that for every $i \in [3, k]$, $|J_{xy}^{(i)}| = \pi(xy)n$. We now prove the induction step for $i = k + 1$. Let $x, y \in \{0, 1\}$. By the definition of $J_{xy}^{(i)}$, it follows that $J_{xy}^{(i)} = \{j \in J_{0x}^{(i-1)} \cup J_{1x}^{(i-1)} : w_{i-1,j}^{(ps)} = y\}$. By the induction hypothesis, we have that $|J_{ux}^{(i-1)}| = \pi(ux)n$, for all $u \in \{0, 1\}$ and from lines 15 and 18 of Algorithm 1 we have that $\mathbf{w}_{i-1}^{(ps)}|_{J_{ux}^{(i-1)}} \in \mathbb{C}(\pi(ux)n, \mathcal{P}(ux1)n)$. Hence, there are exactly $\sum_{u\in\{0,1\}} \mathcal{P}(uxy)n$ indices j for which $w_{i-2,j}^{(ps)} w_{i-2,j}^{(ps)} = xy$. By the stationarity of $\mathcal{P}$ it follows that $\sum_{u\in\{0,1\}} \mathcal{P}(uxy)$ $n = \pi(xy)n$. □

Lemma 2: If there are no error sources, including the bitline ICI, then the programming process described in Algorithm 1 produces a bitline ICI constrained coding scheme.

Proof: Let $j \in [n]$ and $i \in [3, m]$ be such that $w_{i-2,j}^{(ps)} w_{i-1,j}^{(ps)} = 1\,0$. Then, $j \in J_{1\,0}^{(i)}$ and since $\mathbf{w}_i^{(ps)}|_{J_{10}^{(i)}} \in \mathbb{C}(\pi(1\,0)n, \mathcal{P}(1\,0\,1)n) = \{\mathbf{0}\}$, we conclude that $w_{i,j}^{(ps)} = 0$. Thus, there does not exist a $j \in [n]$ and $i \in [3, m]$ for which $w_{i-2,j}^{(ps)} w_{i-1,j}^{(ps)} w_{i,j}^{(ps)} = 1\,0\,1$. □

The proof of the next lemma can be found in Appendix B.

Lemma 3:

$$\frac{\log_2 |C^{(wl)}|}{n} \xrightarrow[n\to\infty]{} H(\mathcal{P})$$

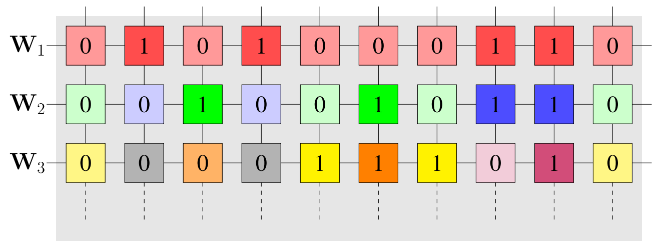

Fig. 4. Programming the first three wordlines, for $n = 10$, according to $\mathcal{P}(xyz) = 0.2$, for all $xyz \in \{0\,0\,0, 0\,0\,1, 1\,0\,0\}$, and $\mathcal{P}(xyz) = 0.1$ for all $xyz \in \{0\,1\,0, 0\,1\,1, 1\,1\,0, 1\,1\,1\}$. Although the rate of the code is approximately 0.458, the rate of the coding scheme for the Markov chain $\mathcal{P}$ approaches the entropy rate $H(\mathcal{P}) = 0.8$ as n goes to infinity. The color of a cell indicates both the symbol it stores and to which constant-weight code it belongs, according to the color index described in Figure 3. The gray cells, i.e., the cells bellow the bitline pattern $1\,0$, must be programmed to 0.

Theorem 1: If there are no error sources, excluding the bitline ICI, then the programming procedure described in Algorithm 1 is properly defined and it produces a row-by-row bitline ICI constrained coding scheme of fixed rate (starting from the third wordline). The rate of the coding scheme is given by

$$R = \frac{\log_2 |C^{(wl)}|}{n} \xrightarrow[n\to\infty]{} H(\mathcal{P})$$

Proof: By Lemmas 1 and 2, it follows that the programming procedure described in Algorithm 1 is properly defined and that it produces a bitline ICI constrained coding scheme, assuming that there are no error sources. Since the bitline ICI error mechanism affects only patterns of the form $1\,0\,1$ along bitlines, it follows that it cannot affect this programming procedure. Otherwise, we will have that the pattern $1\,0\,1$ must appear along a bitline, before any error occurred, in contradiction to Lemma 2.

The last statement of the theorem follows from Lemma 3. □

An example of programming the first three wordlines according to Algorithm 1 is shown in Fig. 4.

The decoding of $\mathbf{w}_i$ is done by first reading the two previous wordlines, $\mathbf{W}_{i-2}$ and $\mathbf{W}_{i-1}$, and then partitioning $\mathbf{w}_i$ into four subsequences, $\widehat{\mathbf{c}}_{xy}^{(i)}$, $xy \in \{0, 1\}^2$, where $w_{i,j}$ belongs to $\widehat{\mathbf{c}}_{xy}$ if the two cells above it, $W_{i-2,j}$ and $W_{i-1,j}$, store the symbols x and y, respectively. The decoding of $\mathbf{w}_i$ is the message $\widehat{M} = \mathcal{D}^{(wl)}(\widehat{\mathbf{c}}_{00}, \widehat{\mathbf{c}}_{01}, \widehat{\mathbf{c}}_{10}, \widehat{\mathbf{c}}_{11})$. In the next theorem we formulate the decoding process in Algorithm 2 and prove that it correctly decodes every wordline.

Theorem 2: For $i \in [m]$, let M be the message that was programmed into $\mathbf{W}_i$, according to Algorithm 1, and let $\widetilde{M}$ be the message that was read from $\mathbf{W}_i$, according to Algorithm 2. If there are no other error sources but bitline ICI then $\widetilde{M} = M$.

Proof: By Theorem 1 it follows that the programming procedure that is described in Algorithm 1 produces a bitline ICI constrained coding scheme. Since we assume no error sources but the bitline ICI, it follows that $\mathbf{w}_i = \mathbf{w}_i^{ps}$ for all $i \in [m]$. This implies that for all $i \in [3, m]$ and for all $x, y \in \{0, 1\}$, $J_{xy} = \widehat{J}_{xy}$ (J_{xy} and $\widehat{J}_{xy}$ are defined in line 17 of Algorithm 1 and in line 12 of Algorithm 2, respectively, when the ith wordline is to be programmed or to be read) and that $\widetilde{M} = \mathcal{D}^{(wl)}(\mathbf{w}_i|_{\widehat{J}_{00}}, \mathbf{w}_i|_{\widehat{J}_{01}}, \mathbf{w}_i|_{\widehat{J}_{10}}, \mathbf{w}_i|_{\widehat{J}_{11}}) =$

Algorithm 2 Row-by-Row Bitline ICI Constrained Coding Scheme: Decoding Process

```
1: Input: An index $i$ of the wordline to be read.
2: Output: A message $\widetilde{M}$.
3: if $i = 1$ then                                   ▷ $\widetilde{M} \in \mathbb{Z}_{|\mathcal{C}^{(1)}|}$
4:     Set $\widetilde{M} \leftarrow \mathcal{D}^{(1)}(\mathbf{w}_i)$.
5: else if $i = 2$ then                              ▷ $\widetilde{M} \in \mathbb{Z}_{|\mathcal{C}^{(2)}|}$
6:     for all $x \in \{0, 1\}$ do
7:         Set $\widehat{J}_x \leftarrow \{j : w_{1,j} = x\}$.
8:     end for
9:     Set $\widetilde{M} \leftarrow \mathcal{D}^{(2)}(\mathbf{w}_2|_{\widehat{J}_0}, \mathbf{w}_2|_{\widehat{J}_1})$.
10: else if $i \geq 3$ then                          ▷ $\widetilde{M} \in \mathbb{Z}_{|\mathcal{C}^{(wl)}|}$
11:     for all $x, y \in \{0, 1\}$ do
12:         Set $\widehat{J}_{xy} \leftarrow \{j : w_{i-2,j}w_{i-1,j} = xy\}$.
13:     end for
14:     Set $\widetilde{M} \leftarrow \mathcal{D}^{(wl)}(\mathbf{w}_i|_{\widehat{J}_{00}}, \mathbf{w}_i|_{\widehat{J}_{01}}, \mathbf{w}_i|_{\widehat{J}_{10}}, \mathbf{w}_i|_{\widehat{J}_{11}})$.
15: end if
```

$\mathcal{D}^{(wl)}(\mathcal{E}^{(wl)}(M)) = M$. Similarly, the theorem holds for $i \in \{1, 2\}$. □

As mentioned above, a technique to obtain an n-integral stationary Markov chain on G from a stationary Markov chain on G with close entropy rates is described in Appendix A. Applying this technique on the Markov chain $\widehat{\mathcal{P}}$ defined in (1) for which $H(\widehat{\mathcal{P}}) = Cap(ICI)$, we obtain an n-integral stationary Markov Chain on G, $\mathcal{P}$, such that $H(\mathcal{P}) = Cap(ICI) - O(1/n)$. Together with Theorems 1 and 2, this implies the following corollary.

Corollary 1: For every $\epsilon > 0$ and sufficiently large n, there exists an n-integral stationary Markov chain on G, $\mathcal{P}$, such that the coding scheme described in Algorithms 1 and 2 produces a row-by-row bitline ICI constrained coding scheme of rate at least $Cap(ICI) - \epsilon$.

IV. Weakly Bitline ICI Constrained Codes

By preventing all bitline ICI errors, the coding scheme suggested in Section III may impose a too severe rate penalty if the probability of ICI-induced errors is not very large. In this section we will adapt our coding scheme to Markov chains $\mathcal{P}$ that allow $\mathcal{P}(101)$ to be positive. Immink [16] coined the term *weakly constrained code* for codes that violate a certain constraint with some fixed probability q. We thereby define *weakly bitline ICI constrained coding scheme* for an SLC flash memory to be a coding scheme in which the probability that the pattern 1 0 1 appears along a bitline is fixed, whereas the probability of every other pattern in $\{0, 1\}^3$ to appear along a bitline is not restricted to a certain value. We will design a row-by-row weakly bitline ICI constrained coding scheme by combining the row-by-row coding scheme from the previous section with an error-correcting code. The rate of the resulting coding scheme may exceed $Cap(ICI)$, if the probability of bitline ICI errors is small. Moreover, this approach can handle errors that are caused by other error mechanisms as well.

Before we can present the encoding and decoding processes for weakly bitline ICI constrained codes we need to slightly modify the graph G. Since we wish to allow the appearance of the pattern 1 0 1, we can no longer consider Markov chains on the graph G, and we define the graph D that is obtained from the graph G by adding the edge 1 0 1, which starts in 1 0, terminates in 0 1, and is labeled by 1. Note that D is just the binary 2-dimensional De-Bruijn graph [8]. Throughout this section, a Markov chain should be understood as a Markov chain on the graph D.

Our coding technique uses some part of the memory for error-correction. Thus, every wordline is partitioned into two parts. The first part, consisting of the first $k \leq n$ cells, is called the *systematic part* of the wordline. The second part, consisting of the last $r = n - k$ cells, is called the *parity check part* of the wordline. Specifically, we introduce the following notations. For every $i \in [m]$, denote the systematic part of $\mathbf{W}_i$ by $\mathbf{S}_i = W_{i,1}W_{i,2}\ldots W_{i,k}$, and the parity check part of $\mathbf{W}_i$ by $\mathbf{P}_i = W_{i,k+1}W_{i,k+2}\ldots W_{i,n}$. As before, we distinguish between the current stored symbols and the programmed states of the cells for the systematic part and use the notations $\mathbf{s}_i \overset{\text{def}}{=} w_{i,1}w_{i,2}\ldots w_{i,k}$ and $\mathbf{s}_i^{(ps)} \overset{\text{def}}{=} w_{i,1}^{(ps)}w_{i,2}^{(ps)}\ldots w_{i,k}^{(ps)}$ to refer to these sequences.

Let $\mathcal{P}$ be a Markov chain on D and let α be the probability that a vertical occurrence of the pattern 1 0 1 changes to 1 1 1 due to a bitline ICI error (we assume α is a known parameter of the flash memory device). Our coding technique will produce a weakly bitline ICI constrained code with $|\{j : w_{i-2,j}^{(ps)}w_{i-1,j}^{(ps)}w_{i,j}^{(ps)} = 1\,0\,1\}| = \mathcal{P}(1\,0\,1)n$, for all $i \in [3, m]$ and $j \in [k]$, and thus the probability of a bitline ICI error in a cell becomes $\mathcal{P}(1\,0\,1)\alpha$. By choosing a suitable value for $\mathcal{P}(1\,0\,1)$ we will be able to control this error probability as we desire.

Let $\mathcal{C}^{(ec)} \subset \{0, 1\}^n$ be a code of dimension k, i.e., $|\mathcal{C}^{(ec)}| = 2^k$. Assume that $\mathcal{C}^{(ec)}$ admits a systematic encoder $\mathcal{E}^{(ec)} : \{0, 1\}^k \to \{0, 1\}^n$, i.e., $\mathcal{E}^{(ec)}(\mathbf{u})|_{[k]} = \mathbf{u}$, for all $\mathbf{u} \in \{0, 1\}^k$, and that $\mathcal{D}^{(ec)} : \{0, 1\}^n \to \{0, 1\}^k$ is the corresponding decoder of $\mathcal{C}^{(ec)}$. The superscript (ec) in the code's notation stands for error-correction. The code $\mathcal{C}^{(ec)}$ will be used to correct bitline ICI-induced errors and may be chosen to guarantee that the frame-error-rate (FER) does not exceed a certain threshold in exchange for reduction in the overall coding rate.

Let $\mathcal{C}^{(1)}, \mathcal{C}^{(2)}, \mathcal{C}^{(wl)}$ be the length-k codes as defined in (4) with respect to the Markov chain $\mathcal{P}$. Notice that, since the definition of these codes depends on the Markov chain $\mathcal{P}$, changing the Markov chain $\mathcal{P}$ results in changing these codes accordingly. In particular, since $\mathcal{P}(1\,0\,1) > 0$, the code $\mathbb{C}(\pi(1\,0)k, \mathcal{P}(1\,0\,1)k)$ is no longer trivial.

As before, we program $\mathbf{S}_i$ according to the sequences that were programmed into $\mathbf{S}_{i-2}$ and $\mathbf{S}_{i-1}$. However, before programming $\mathbf{S}_i$, $\mathbf{S}_{i-2}$ may have suffered from bitline ICI errors in cells $W_{i,j}$, $j \in [3, k]$, for which $w_{i-3,j}^{(ps)}w_{i-2,j}^{(ps)}w_{i-1,j}^{(ps)} = 1\,0\,1$ (as in Section III, we assume the bitline ICI-error mechanism to be the only source of errors). Therefore, we use a side memory (e.g. DRAM memory or a designated part of the memory) which stores the information that was written on the wordline $\mathbf{S}_{i-2}$ reliably. The sequence stored in the side memory is denoted by $\mathbf{v} = v_1v_2\ldots v_k$. Upon programming

$\mathbf{W}_i$, $\mathbf{v}$ stores the programming state of $\mathbf{S}_{i-2}$ (before any errors had occurred) and once $\mathbf{v}$ is used to process the programming state of $\mathbf{W}_i$, $\mathbf{v}$ is reprogrammed to store the programming state of $\mathbf{S}_{i-1}$.

To overcome errors we use the error-correcting code $\mathcal{C}^{(ec)}$. Hence, in each step we also encode $\mathbf{S}_i$ by the systematic encoder $\mathcal{E}^{(ec)}$ and store the parity check bits of the resulting codeword in $\mathbf{P}_i$. This programming process is formulated in Algorithm 3 and is proved to produce a weakly bitline ICI constrained code in the next theorem.

Remark 2: The weakly bitline ICI constraint is enforced only on the systematic part $\mathbf{S}_i$ *and not on the parity part* $\mathbf{P}_i$*. As a result, the probability of a bitline ICI error in the parity part may be higher and the proposed coding scheme relies on the error-correcting code to correct the corresponding errors. This approach is commonly used in reverse concatenation architecture for constrained systems [4], [5], [22].*

Algorithm 3 Row-by-Row Weakly Bitline ICI Constrained Coding Scheme: Programming Process

```
1:  Initialize: $i \leftarrow 1$.
2:  Input: A message $M$ to be programmed into the current
        wordline $\mathbf{W}_i$.
3:  if $i = 1$ then                                    ▷ $M \in \mathbb{Z}_{|\mathcal{C}^{(1)}|}$
4:      Program $\mathcal{E}^{(ec)}(\mathcal{E}^{(1)}(M))$ into $\mathbf{W}_1$ and $\mathbf{v}$.
5:      Set $i \leftarrow 2$.
6:  else if $i = 2$ then                               ▷ $M \in \mathbb{Z}_{|\mathcal{C}^{(2)}|}$
7:      Set $(\mathbf{c}_0, \mathbf{c}_1) \leftarrow \mathcal{E}^{(2)}(M)$.
8:      for all $x \in \{0,1\}$ do
9:          Set $J_x \leftarrow \{j \in [k] \ : \ w_{1,j} = x\}$.
10:         Set $\mathbf{u}|_{J_x} \leftarrow \mathbf{c}_x$.              ▷ $\mathbf{u} \in \{0,1\}^k$
11:     end for
12:     Program $\mathcal{E}^{(ec)}(\mathbf{u})$ into $\mathbf{W}_2$.
13:     Set $i \leftarrow 3$.
14: else if $i \geq 3$ then                            ▷ $M \in \mathbb{Z}_{|\mathcal{C}^{(3)}|}$
15:     Set $(\mathbf{c}_{00}, \mathbf{c}_{01}, \mathbf{c}_{10}, \mathbf{c}_{11}) \leftarrow \mathcal{E}^{(wl)}(M)$.
16:     for all $x, y \in \{0,1\}$ do
17:         Set $J_{xy} \leftarrow \{j \in [k] \ : \ v_j w_{i-1,j} = xy\}$.
18:         Set $\mathbf{u}|_{J_{xy}} \leftarrow \mathbf{c}_{xy}$.            ▷ $\mathbf{u} \in \{0,1\}^k$
19:     end for
20:     $\mathbf{v} \leftarrow \mathbf{s}_{i-1}$.
21:     Program $\mathcal{E}^{(ec)}(\mathbf{u})$ into $\mathbf{W}_i$.
22:     Set $i \leftarrow i + 1$.
23: end if
```

Theorem 3: If there are no error sources, excluding the bitline ICI, then the programming procedure described in Algorithm 3 is properly defined and it produces a row-by-row weakly bitline ICI coding scheme of fixed rate (starting from the third wordline). The rate of the coding scheme is given by

$$R = \log_2 |\mathcal{C}^{(wl)}|/n \approx \frac{H(\mathcal{P})k}{n}.$$

Proof: First, we must show that the programming process defined in Algorithm 3 is properly defined. Since only the programming of the systematic part is dependent on (the systematic parts of) the two previous wordlines, it is sufficient to show that the systematic part of the memory is properly defined. Note that, Lemma 1 holds regardless of whether or not $\mathcal{P}(1\,0\,1) = 0$. Since at each step $i \in [3, m]$, the programming state of $\mathbf{S}_{i-2}$ is available through the side memory, occurrences of bitline ICI errors do not affect the programming state of $\mathbf{S}_i$ and the validity of Algorithm 3 follows immediately from Lemma 1.

The last statement of the theorem follows from Lemma 3. □

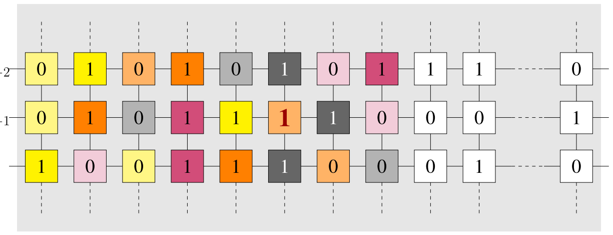

Fig. 5. Example of three consecutive wordlines that were programmed by Algorithm 3 with $\mathcal{P}(xyz) = 0.125$, for all $xyz \in \{0,1\}^3$. The color of a cell indicates both its programmed states and to which constant-weight code it belongs, according to the color index described in Figure 3. The white cells represent the parity check part of the memory. The sixth cell in $\mathbf{W}_{i-1}$ was originally programmed to 0, but due to a bitline ICI error, it stores 1.

An example of programming three wordlines according to Algorithm 3 is shown in Fig. 5.

The decoding of $\mathbf{w}_i$ is done by first reading $\mathbf{W}_i$ and the two previous wordlines, $\mathbf{W}_{i-2}$, $\mathbf{W}_{i-1}$, and applying the decoder of the error-correcting code on each of these three wordlines. If the error-correcting code can correct all the errors, then by doing so we obtain $\mathbf{s}_{i-2}^{(ps)}$, $\mathbf{s}_{i-1}^{(ps)}$, and $\mathbf{s}_i^{(ps)}$. We then proceed with decoding $\mathbf{s}_i^{(ps)}$, similarly to the decoding process described in Algorithm 2, i.e., we partition $\mathbf{s}_i^{(ps)}$ to four subsequences, $\widehat{\mathbf{c}}_{xy}$, $xy \in \{0,1\}^2$, where $s_{i,j}^{(ps)}$ belongs to $\widehat{\mathbf{c}}_{xy}$ if $s_{i-2,j}^{(ps)} s_{i-1,j}^{(ps)} = xy$. The decoding of $\mathbf{w}_i$ is the message $\widehat{M} = \mathcal{D}^{(wl)}(\widehat{\mathbf{c}}_{00}, \widehat{\mathbf{c}}_{01}, \widehat{\mathbf{c}}_{10}, \widehat{\mathbf{c}}_{11})$. We formulate the decoding process in Algorithm 4 and prove that it correctly decodes $\mathbf{w}^{(i)}$, given that $\mathcal{D}^{(ec)}$ successfully decodes $\mathbf{w}^{i-2}$, $\mathbf{w}^{(i-1)}$, and $\mathbf{w}^{(i)}$, in the next theorem.

Theorem 4: For $i \in [3, m]$*, let* M *be the message that was programmed into* $\mathbf{W}_i$ *according to Algorithm 3 and let* $\widetilde{M}$ *be the message that was read from* $\mathbf{W}_i$ *according to Algorithm 4. If* $\mathcal{D}^{(ec)}$ *can correctly decode* $\mathbf{w}_{i-2}, \mathbf{w}_{i-1}, \mathbf{w}_i$ *then* $\widetilde{M} = M$.

Proof: Since $\mathcal{D}^{(ec)}$ successfully decodes $\mathbf{w}_{i-2}, \mathbf{w}_{i-1}, \mathbf{w}_i$, it follows that $\widehat{\mathbf{s}}_b = \mathbf{s}_b^{(ps)}$, for all $b \in \{i-2, i-1, i\}$. This implies that for all $x, y \in \{0,1\}$, $J_{xy} = \widehat{J}_{xy}$ (J_{xy} and $\widehat{J}_{xy}$ are defined in line 17 of Algorithm 3 and in line 16 of Algorithm 4, respectively, when the ith wordline is to be programmed or to be read) and that $\widetilde{M} = \mathcal{D}^{(wl)}(\mathbf{s}_i^{(ps)}|_{\widehat{J}_{00}}, \mathbf{s}_i^{(ps)}|_{\widehat{J}_{01}}, \mathbf{s}_i^{(ps)}|_{\widehat{J}_{10}}, \mathbf{s}_i^{(ps)}|_{\widehat{J}_{11}}) = \mathcal{D}^{(wl)}(\mathcal{E}^{(wl)}(M)) = M$. □

To illustrate the usefulness of row-by-row weakly bitline ICI constrained codes we consider the case where $\alpha = 0.05$. Again, we assume that the bitline ICI is the only error mechanism. Fix a target coding rate $R = 0.9$ and a code length $n = 9102$. A weakly bitline ICI constrained code uses the first k bits to encode the information bits to the systematic part of the memory while controlling the number of vertical

Algorithm 4 Row-by-Row Weakly Bitline ICI Constrained Coding Scheme: Decoding Process

```
1: Input: An index $i$ of the wordline to be read.
2: Output: A message $\widetilde{M}$.
3: if $i = 1$ then                                        $\triangleright\ \widetilde{M} \in \mathbb{Z}_{|C^{(1)}|}$
4:     Set $\widetilde{M} \leftarrow \mathcal{D}^{(1)}(\mathcal{D}^{(ec)}(\mathbf{w}_i))$.
5: else if $i = 2$ then                                   $\triangleright\ \widetilde{M} \in \mathbb{Z}_{|C^{(2)}|}$
6:     Set $\widehat{\mathbf{s}}_1 \leftarrow \mathcal{D}^{(ec)}(\mathbf{w}_1)$ and $\widehat{\mathbf{s}}_2 \leftarrow \mathcal{D}^{(ec)}(\mathbf{w}_2)$.
7:     for all $x \in \{0, 1\}$ do
8:         Set $\widehat{J}_x \leftarrow \{j \ :\ \widehat{s}_{1,j} = x\}$.
9:     end for
10:    Set $\widetilde{M} \leftarrow \mathcal{D}^{(2)}(\widehat{\mathbf{s}}_2|_{\widehat{J}_0}, \widehat{\mathbf{s}}_2|_{\widehat{J}_1})$.
11: else if $i \geq 3$ then                               $\triangleright\ \widetilde{M} \in \mathbb{Z}_{|C^{(wl)}|}$
12:    for all $b \in \{i-2, i-1, i\}$ do
13:        Set $\widehat{\mathbf{s}}_b \leftarrow \mathcal{D}^{(ec)}(\mathbf{w}_b)$.
14:    end for
15:    for all $x, y \in \{0, 1\}$ do
16:        Set $\widehat{J}_{xy} \leftarrow \{j \ :\ \widehat{s}_{i-2,j}\widehat{s}_{i-1,j} = xy\}$.
17:    end for
18:    Set $\widetilde{M} \leftarrow \mathcal{D}^{(wl)}(\widehat{\mathbf{s}}_i|_{\widehat{J}_{00}}, \widehat{\mathbf{s}}_i|_{\widehat{J}_{01}}, \widehat{\mathbf{s}}_i|_{\widehat{J}_{10}}, \widehat{\mathbf{s}}_i|_{\widehat{J}_{11}})$.
19: end if
```

occurrences of the pattern 1 0 1 and it uses $r = n - k$ parity check bits to encode the systematic part using some systematic error-correcting code. For this illustration we use a BCH code as the systematic error-correcting code. Let $R_{sys} = nR/k$ be the rate of encoding the information into the systematic part of the memory. Figure 6 illustrates the frame-error-rate as a function of R_{sys}. For a given value of R_{sys} the corresponding row-by-row weakly bitline ICI constrained code was designed to minimize the probability of error in the first k bits by minimizing the occurrence of the pattern 1 0 1. For the parity check bits we assume the probability of error is $\alpha/8$. In the extreme case where $R_{sys} = 0.9$, i.e., $n = k$ and no error-correcting code is being used, the FER is close to one, since even a single bit error leads to a frame error and the probability of not observing any bit error is extremely small. On the other extreme we have the case where $R_{sys} = 1$ and all the parity check bits are dedicated to error correction. For every $0.9 < R_{sys} < 1$ we have a non-trivial row-by-row weakly bitline ICI constrained code that combines both the encoding to the systematic part and error-correction using a BCH code. Fig 6 indicates that in the range $0.94 \leq R_{sys} < 1$, the non-trivial row-by-row weakly-bitline ICI constrained codes outperform the BCH code (the case where $R_{sys} = 1$). The best FER was measured for $R_{sys} = 0.98$ and is equal to 0.00298 which is roughly 44 times lower than the FER for the BCH code.

V. Extensions for MLC Flash Memory

In this section we consider several extensions for MLC flash memory, where the goal is to mitigate ICI errors in the horizontal direction, in the vertical direction, or in both directions. The suggested methods can also be extended for 3-dimensional three-level cell (TLC) flash memory for ICI error mitigation in one or more of the three directions.

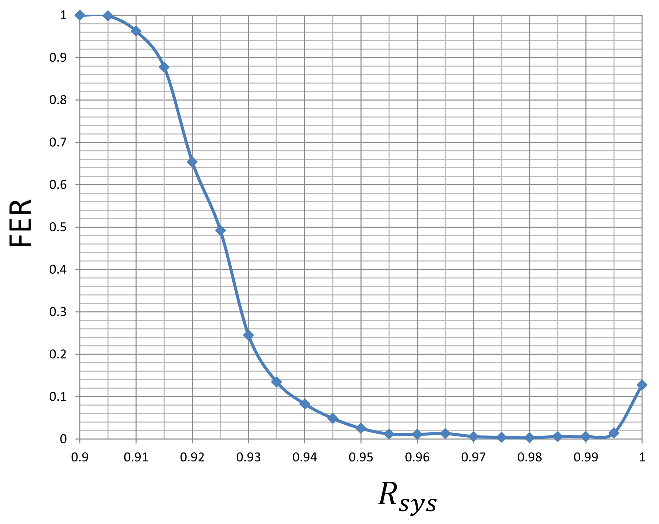

Fig. 6. Frame-error-rate (FER) as a function of R_{sys} for weakly bitline ICI constrained code of length $n = 9102$ with a BCH code as the systematic error-correcting code. The probability that a vertical occurrence of the pattern 1 0 1 changes to 1 1 1 due to a bitline ICI error is $\alpha = 0.05$.

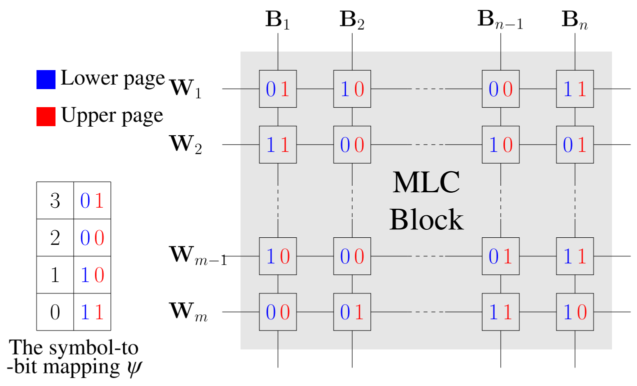

Fig. 7. Structure of a MLC flash memory block.

A cell in MLC flash memory stores one of the symbols of the four-element alphabet $\Sigma = \{0, 1, 2, 3\}$. These symbols are represented by two bits according to the gray mapping $\psi : \Sigma \rightarrow \{0, 1\}^2$ defined by $\psi(0) = 1\,1$, $\psi(1) = 1\,0$, $\psi(2) = 0\,0$, and $\psi(3) = 0\,1$. The two bits of a cell are programmed into two logical units of programming that are called the *lower page* and the *upper page*. The most significant bit is programmed to the lower page while the least significant bit is programmed to the upper page. The structure of a block in a MLC flash memory is depicted in Figure 7.

The experimental studies in [33] indicate that the pattern that is most likely to cause ICI errors (horizontally or vertically) in MLC flash memory is of the form 3 0 3. We therefore suggest methods to eliminate the pattern 3 0 3 horizontally, vertically, or in both directions, while programming the lower page and the upper page independently.

We start with the horizontal direction. Since the pattern 3 0 3 is represented by the pattern 0 1 0 in the lower page and 1 1 1 in the upper page, the pattern 3 0 3 can be eliminated horizontally by preventing the appearance of the pattern 1 1 1 along wordlines in the upper page and applying no coding on the lower page. Hence, we program the upper page with a length-n constrained code in which no codeword contains the pattern 1 1 1. Such a constrained code is known as a (d, k)-

RLL (run-length-limited) code for the parameters $d = 0$ and $k = 2$ (see [12] and references therein). A *run* in a codeword is a maximum substring (a sequence of consecutive entries) of a repeated symbol. For example, the runs in the codeword $0\,0\,1\,1\,0\,1\,0\,0$ are $0\,0$, $1\,1$, 0, 1, and $0\,0$. In a (d, k)-RLL code,[2] every codeword satisfies the property that the run length of 1 is at least d and at most k. For $d = 0$ and $k = 2$, this is equivalent to the property that no codeword contains the pattern $1\,1\,1$. The capacity of the $(0, 2)$-RLL constraint is approximately 0.8791. Adler *et al.* [1] presented an algorithm for generating rate-p/q (d, k)-RLL codes with arbitrary parameters p, q, d, and k, where p/q is upper bounded by the capacity of the (d, k)-RLL constraint. Hence, for large n, we achieved a method for preventing the pattern $3\,0\,3$ horizontally with rate approximately $0.8791 + 1 = 1.8791$.

Remark 3: In the suggested method the pattern $3\,0\,3$ *is eliminated horizontally by preventing the pattern* $1\,1\,1$ *along wordlines in the upper page entirely, when in fact this pattern should not appear horizontally in the upper page only if the pattern* $0\,1\,0$ *appears in the lower page simultaneously. Such a scenario in which two sources operate independently but never transmit a certain pair of patterns simultaneously was studied in [24], where numerical methods to estimate the sum rate capacity of the system were presented. Using one of these methods, we computed the sum rate capacity when* $1\,1\,1$ *and* $0\,1\,0$ *never appear simultaneously and found it to be* 1.8791. *Thus, the suggested method is an optimal scheme to prevent* $3\,0\,3$ *horizontally while programming the two pages independently.*

For the vertical direction, we use a row-by-row programming method as in Section III to avoid the pattern $1\,1\,1$ along bitlines in the upper page. When n and m are large, this method again achieves a rate which is approximately $0.8791 + 1 = 1.8791$, only this time the pattern $3\,0\,3$ is prevented along bitlines.

To prevent $3\,0\,3$ from appearing both horizontally and vertically we apply the row-by-row programming method from Section III on the lower page to prevent the pattern $0\,1\,0$ along bitlines and use a length-n $(0, 2)$-RLL code for the upper page to prevent the pattern $1\,1\,1$ from appearing along wordlines. For large n and m, this method prevents the pattern $3\,0\,3$ both horizontally and vertically and has rate approximately $0.8791 + 0.8114 = 1.6905$.

Another method to prevent the pattern $3\,0\,3$ from appearing both horizontally and vertically is by applying a 2-dimensional $(0, 2)$-RLL constrained code on the upper page, which prevents the pattern $1\,1\,1$ from appearing both horizontally and vertically. Despite the extensive study of 2-dimensional (d, k)-RLL constrained codes [14], [25], [26], [29], [30], [32], the capacity of these codes remains an open problem for most values of d and k and in particular for 2-dimensional $(0, 2)$-RLL constrained codes. The best known 2-dimensional $(0, 2)$-RLL constrained code was given in [30] and its rate is 0.816. Thus, by applying this code on the upper page and apply no coding on the lower page, we can eliminate the pattern $3\,0\,3$ in both directions with rate 1.816. However, this 2-dimensional $(0, 2)$-RLL constrained code may be inapplicable for flash memory programming and the reason for this is twofold. One aspect is that it induces a variable-rate coding as opposed to a fixed-rate. The second aspect is that the encoding cannot be conducted in a row-by-row fashion. Hence, for future work it will be interesting to design a fixed rate row-by-row coding scheme that avoids the pattern $1\,1\,1$ both horizontally and vertically.

Finally, we remark that all the patterns of the form $3\,x\,3$, where $x \in \{0, 1, 2, 3\}$, can be avoided in the vertical direction by excluding the patterns $1\,0\,1$ and $1\,1\,1$ along the bitlines of the lower page. The row-by-row coding technique presented in this paper can be applied for any constraint, given an n-integral stationary Markov chain on the corresponding graph. The simple technique for obtaining an n-integral stationary Markov chain with high entropy rate, presented in Appendix A, can be adapted for the graph associated with avoiding the two patterns $1\,0\,1$ and $1\,1\,1$. The rate in this case is approximately 1.6942. Alternatively, one can apply the row-by-row coding technique on the set of wordlines of odd index and the set of wordlines of even index, independently, to guarantee that $w_{i-2,j}w_{i,j} \neq 1\,1$ for all $i \in [3, m]$ and $j \in [n]$. While this approach will result in the same rate, the row-by-row coding scheme in this case is simpler and its programming requires the reading of only one wordline.

VI. Conclusion

In this paper, row-by-row coding schemes were suggested to handle bitline ICI errors in flash memories. The first coding scheme adapts the technique from [31] to eliminate the pattern $1\,0\,1$ along bitlines and the resulting coding rate asymptotically attains the capacity of the ICI constraint. The second coding scheme maintains a nonzero probability of vertical occurrence for the pattern $1\,0\,1$ and in addition uses a systematic error-correcting code. This scheme is useful when the overall designated coding rate exceeds the ICI capacity as well as in the scenario that other error sources apply. A simulation result which depicted a significant performance improvement using this approach was presented. Extensions for MLC flash memory were also discussed where the objective is to design a fixed rate row-by-row coding scheme that mitigates the pattern $3\,0\,3$ horizontally, vertically, or in both directions, and yet programs the lower and upper pages independently.

Appendix A
Obtaining n-Integral Stationary Markov Chains

In this subsection we will describe a simple process that takes as an input a stationary Markov chain on D and an integer n and outputs an n-integral stationary Markov chain on D, with a very close entropy rate.[3] Recall that D is the graph obtained from the graph G by adding the edge $1\,0\,1$, which starts in $1\,0$, terminates in $0\,1$, and is labeled by 1. The same process works also for stationary Markov chains on G and

[2]Conventionally, in a (d, k)-RLL code every run length of 0 must be at least d and at most k. For convenience, we choose to apply the run length limitation on the symbol 1 instead of 0, where the two definitions are obviously equivalent.

[3]We remark that a general and more complicated process that achieves this goal for general graphs was given in [31].

achieves a similar result, namely, that the difference between the entropy rates of the two Markov chains is bounded (in absolute value) by $\frac{c}{n}$, for some positive constant c and for sufficiently large n.

A Markov chain $\mathcal{P}$ on D (or on G) can be represented by a 4×4 matrix, also denoted by $\mathcal{P}$, where $\mathcal{P}_{2x+y,2y+z} \overset{\text{def}}{=} \mathcal{P}(xyz)$ and $\mathcal{P}_{2x+y,2w+z} \overset{\text{def}}{=} 0$, if $y \neq w$ (in the same way that the transition matrix is obtained from the conditional probability $Q(z|xy)$ in subsection II-B).

Lemma 4: A mapping $\mathcal{P} : E \to \mathbb{R}$ is a stationary Markov chain on D if and only if the matrix $\mathcal{P}$ satisfies the following properties.

(a) *$\mathcal{P}_{i,j} \geq 0$ for all $i, j \in [0, 3]$.*
(b) *$\mathcal{P}_{i,j} = 0$ if $i = 2x + y$ and $j \notin \{2y, 2y + 1\}$.*
(c) *$\sum_{i=0}^{3} \sum_{j=0}^{3} \mathcal{P}_{i,j} = 1$.*
(d) *For all $i \in [0, 3]$, $\sum_{j=0}^{3} \mathcal{P}_{i,j} = \sum_{j=0}^{3} \mathcal{P}_{j,i}$.*

Notice that $\mathcal{P}$ is a stationary Markov chain on G if and only if $\mathcal{P}_{2,1} = 0$ (in addition to the four properties of Lemma 4).

Proof: Items (a), (b), and (c) of Lemma 4 state the necessary and sufficient conditions for $\mathcal{P}$ to be a Markov chain on D. By definition, $\mathcal{P}$ is stationary if and only if item (d) of the lemma is satisfied. □

Construction 1: Let $n \geq 1$ be an integer and let $\mathcal{P}^{(1)}$ be a stationary Markov chain on D. Define the 4×4 matrix $M^{(1)} \overset{\text{def}}{=} \lfloor n\mathcal{P}^{(1)} \rfloor$, i.e., $M^{(1)}_{i,j} \overset{\text{def}}{=} \lfloor n\mathcal{P}^{(1)}_{i,j} \rfloor$, for all $i, j \in [0, 3]$, and define

$$s \overset{\text{def}}{=} M^{(1)}_{0,1} + M^{(1)}_{2,1} - M^{(1)}_{1,2} - M^{(1)}_{1,3}$$

Let

$$\widetilde{n} \overset{\text{def}}{=} \sum_{i=0}^{3} \sum_{j=0}^{3} M^{(1)}_{i,j} + |s| \quad \text{and} \quad d \overset{\text{def}}{=} n - \widetilde{n}.$$

Define the 4×4 matrix $M^{(2)}$ as follows. If $s \geq 0$ then,

$$M^{(2)} \overset{\text{def}}{=} \begin{pmatrix} M^{(1)}_{0,0} + \lceil \frac{d}{2} \rceil & M^{(1)}_{0,1} & 0 & 0 \\ 0 & 0 & M^{(1)}_{1,2} + s & M^{(1)}_{1,3} \\ M^{(1)}_{2,0} & M^{(1)}_{2,1} & 0 & 0 \\ 0 & 0 & M^{(1)}_{3,2} & M^{(1)}_{3,3} + \lfloor \frac{d}{2} \rfloor \end{pmatrix}.$$

and if $s < 0$ then,

$$M^{(2)} \overset{\text{def}}{=} \begin{pmatrix} M^{(1)}_{0,0} + \lceil \frac{d}{2} \rceil & M^{(1)}_{0,1} & 0 & 0 \\ 0 & 0 & M^{(1)}_{1,2} & M^{(1)}_{1,3} \\ M^{(1)}_{2,0} & M^{(1)}_{2,1} - s & 0 & 0 \\ 0 & 0 & M^{(1)}_{3,2} & M^{(1)}_{3,3} + \lfloor \frac{d}{2} \rfloor \end{pmatrix}.$$

Finally, define $\mathcal{P}^{(2)}$ by the 4×4 matrix $n^{-1}M^{(2)}$, i.e., for all $x, y, z \in \{0, 1\}$,

$$\mathcal{P}^{(2)}(xyz) \overset{\text{def}}{=} \frac{M^{(2)}_{2x+y,2y+z}}{n}.$$

Theorem 5: For every stationary Markov chain $\mathcal{P}^{(1)}$ on the graph D (or G) and for every integer $n \geq 1$, the mapping $\mathcal{P}^{(2)}$ that is obtained from n and $\mathcal{P}^{(1)}$ according to Construction 1 is an n-integral stationary Markov chain on D (respectively, on G).

Moreover, there exists a constant c, depending only on $\mathcal{P}^{(1)}$, such that $|H(\mathcal{P}^{(2)}) - H(\mathcal{P}^{(1)})| \leq c/n$ for sufficiently large n.

Proof: First, we use Lemma 4 to show that $\mathcal{P}^{(2)}$ is a stationary Markov chain on D. Items (b) and (c) of Lemma 4 are immediate. Since $\mathcal{P}^{(1)}$ is stationary, we have that $\mathcal{P}^{(1)}_{0,1} + \mathcal{P}^{(1)}_{2,1} = \mathcal{P}^{(1)}_{1,2} + \mathcal{P}^{(1)}_{1,3}$ and therefore, $s \in \{-1, 0, 1\}$. Clearly, $\widetilde{n} - |s| = \sum_{i=0}^{3} \sum_{j=0}^{3} M^{(1)}_{i,j} \leq n$. If $\widetilde{n} - |s| < n$ then since $\widetilde{n} - |s| \leq n - 1$ and $|s| \leq 1$, it follows that $d \geq 0$. If $\widetilde{n} - |s| = n$ then $M^{(1)} = \mathcal{P}^{(1)}n$ and therefore $s = 0$ and $d = 0$. In any case, $0 \leq d$. This shows item (a) of Lemma 4. Since $\mathcal{P}^{(1)}$ is stationary it follows that $M^{(1)}_{0,1} = M^{(1)}_{2,0}$, $M^{(1)}_{1,3} = M^{(1)}_{3,2}$. Together with the definitions of s and $M^{(2)}$, we have that $\mathcal{P}^{(2)}$ also satisfies item (d) of Lemma 4. Thus, $\mathcal{P}^{(2)}$ is a stationary Markov chain on D. Moreover, if $\mathcal{P}^{(1)}$ is a stationary Markov chain on G then $M^{(1)}_{2,1} = 0$ and therefore $s \neq -1$ and $M^{(2)}_{2,1} = 0$. Hence, $\mathcal{P}^{(2)}$ is a stationary Markov chain on G.

Since all the entries of the matrix $M^{(2)}$ are integers, it follows that $\mathcal{P}^{(2)}(xyz)n$ is an integer, for all $x, y, z \in \{0, 1\}$.

It remains to show that $|H(\mathcal{P}^{(2)}) - H(\mathcal{P}^{(1)})| \leq c/n$ for some constant c and for sufficiently large n. Notice that since $d + |s| = \sum_{i,j} n\mathcal{P}^{(1)}_{i,j} - \left\lfloor n\mathcal{P}^{(1)}_{i,j} \right\rfloor$, $d + |s|$ is strictly less than the number of positive entries in $\mathcal{P}^{(1)}$ and therefore $d \leq 7 - |s| \leq 7$. Since $0 \leq d \leq 7$ and by the definition of $M^{(2)}$, it follows that

$$\left\lfloor \mathcal{P}^{(1)}(xyz)n \right\rfloor \leq \mathcal{P}^{(2)}(xyz)n \leq \left\lfloor \mathcal{P}^{(1)}(xyz)n \right\rfloor + 4.$$

Therefore,

$$|\mathcal{P}^{(2)}(xyz) - \mathcal{P}^{(1)}(xyz)| \leq \frac{4}{n}. \tag{A.1}$$

For $r \in \{1, 2\}$, let $\pi^{(r)}$ and $Q^{(r)}$ be the state probability vector and the transition matrix of $\mathcal{P}^{(r)}$, respectively. Then for all $x, y \in \{0, 1\}$

$$\begin{aligned} |\pi^{(2)}(xy) - \pi^{(1)}(xy)| &= |\mathcal{P}^{(2)}(xy0) + \mathcal{P}^{(2)}(xy1) \\ &\quad - (\mathcal{P}^{(1)}(xy0) + \mathcal{P}^{(1)}(xy1))| \leq \frac{8}{n}. \end{aligned} \tag{A.2}$$

Inequality (A.1) implies that for sufficiently large n, $\mathcal{P}^{(1)}(xyz) > 0$ if and only if $\mathcal{P}^{(2)}(xyz) > 0$. Moreover, if $\mathcal{P}^{(2)}(xyz) > 0$ then $\pi^{(2)}(xy) > 0$ and $\pi^{(1)}(xy) > 0$. In particular, $\pi^{(1)}(xy) > \frac{8}{n}$, for sufficiently large n and hence, by inequality A.2 it follows that $\pi^{(2)}(xy) \geq \pi^{(1)}(xy) - \frac{8}{n} > 0$. Then, for sufficiently large n and for all $x, y, z \in \{0, 1\}$ for which $\mathcal{P}^{(1)}(xyz) \neq 0$, we have that

$$\begin{aligned} Q^{(2)}(z|xy) = \frac{\mathcal{P}^{(2)}(xyz)}{\pi^{(2)}(xy)} &\leq \frac{\mathcal{P}^{(1)}(xyz) + \frac{4}{n}}{\pi^{(1)}(xy) - \frac{8}{n}} \\ &= Q^{(1)}(z|xy) \left(\frac{1 + \frac{4}{\mathcal{P}^{(1)}(xyz)n}}{1 - \frac{8}{\pi^{(1)}(xy)n}} \right) \\ &\leq Q^{(1)}(z|xy) \left(1 + \frac{c_1}{n} \right), \end{aligned}$$

for some constant $c_1 > 0$, depending only on $\mathcal{P}^{(1)}$. The last inequality follows from the fact that if a and b are two positive constants and n is sufficiently large then $n-b \geq n/2$ and $\frac{1+a/n}{1-b/n} = 1+\frac{a+b}{n-b} \leq 1+\frac{c}{n}$, for $c=2(a+b)$. Similarly,

$$\begin{aligned} Q^{(2)}(z|xy) &= \frac{\mathcal{P}^{(2)}(xyz)}{\pi^{(2)}(xy)} \geq \frac{\mathcal{P}^{(1)}(xyz)-\frac{4}{n}}{\pi^{(1)}(xy)+\frac{8}{n}} \\ &= Q^{(1)}(z|xy)\left(\frac{1-\frac{4}{\mathcal{P}^{(1)}(xyz)n}}{1+\frac{8}{\pi^{(1)}(xy)n}}\right) \\ &\geq Q^{(1)}(z|xy)\left(1-\frac{c_2}{n}\right), \end{aligned}$$

for some constant $c_2 > 0$, depending only on $\mathcal{P}^{(1)}$. As before, the last inequality follows from the fact that if a and b are two positive constants and n is sufficiently large then $\frac{1-a/n}{1+b/n} = 1-\frac{a+b}{n-b} \geq 1-\frac{c}{n}$, for $c=2(a+b)$.

Then,

$$\begin{aligned} H(\mathcal{P}^{(2)}) &= -\sum_{\substack{x,y,z\in\{0,1\}:\\ \mathcal{P}^{(2)}(xyz)>0}} \mathcal{P}^{(2)}(xyz)\log_2 Q^{(2)}(z|xy) \\ &\leq -\sum_{\substack{x,y,z\in\{0,1\}:\\ \mathcal{P}^{(1)}(xyz)>0}} \left(\mathcal{P}^{(1)}(xyz)+\frac{4}{n}\right)\log_2 Q^{(1)}(z|xy) \\ &\quad -\sum_{\substack{x,y,z\in\{0,1\}:\\ \mathcal{P}^{(1)}(xyz)>0}} \left(\mathcal{P}^{(1)}(xyz)+\frac{4}{n}\right)\log_2\left(1-\frac{c_2}{n}\right) \\ &\leq H(\mathcal{P}^{(1)}) - \frac{4}{n}\left(\sum_{\substack{x,y,z\in\{0,1\}:\\ \mathcal{P}^{(1)}(xyz)>0}} \log_2 Q^{(1)}(z|xy)\right) \\ &\quad +(1+\frac{32}{n})\frac{\ln(2)c_2}{n}, \end{aligned}$$

where the last inequality follows from the fact that the sum of $\mathcal{P}^{(1)}(xyz)$ over all $x,y,z\in\{0,1\}$ for which $\mathcal{P}^{(1)}(xyz)>0$ is one and from $\log_2(1-u) \geq -\ln\ (2)u$, for $u>0$. Then,

$$H(\mathcal{P}^{(2)}) \leq H(\mathcal{P}^{(1)}) + \frac{c_3}{n},$$

for sufficiently large n and for some constant $c_3 > 0$. Similarly,

$$\begin{aligned} H(\mathcal{P}^{(2)}) &= -\sum_{\substack{x,y,z\in\{0,1\}:\\ \mathcal{P}^{(2)}(xyz)>0}} \mathcal{P}^{(2)}(xyz)\log_2 Q^{(2)}(z|xy) \\ &\geq -\sum_{\substack{x,y,z\in\{0,1\}:\\ \mathcal{P}^{(1)}(xyz)>0}} \left(\mathcal{P}^{(1)}(xyz)-\frac{4}{n}\right)\log_2 Q^{(1)}(z|xy) \\ &\quad -\sum_{\substack{x,y,z\in\{0,1\}:\\ \mathcal{P}^{(1)}(xyz)>0}} \left(\mathcal{P}^{(1)}(xyz)-\frac{4}{n}\right)\log_2\left(1+\frac{c_1}{n}\right) \\ &\geq H(\mathcal{P}^{(1)}) + \frac{4}{n}\left(\sum_{\substack{x,y,z\in\{0,1\}:\\ \mathcal{P}^{(1)}(xyz)>0}} \log_2 Q^{(1)}(z|xy)\right) \\ &\quad -(1-\frac{32}{n})\frac{\ln(2)c_1}{n}, \end{aligned}$$

where in the last inequality we used the inequality $\log_2(1+u) \leq \ln(2)u$, that holds for all $u>0$. Then,

$$H(\mathcal{P}^{(2)}) \geq H(\mathcal{P}^{(1)}) - \frac{c_4}{n},$$

for sufficiently large n and for some constant c_4. $\square$

Example 1: Let $n=100$ and let $\mathcal{P}^{(1)}$ be the ICI constraint capacity-achieving stationary Markov chain, $\widehat{\mathcal{P}}$, defined in (1). Then[4]

$$\mathcal{P}^{(1)} = \begin{pmatrix} 0.2345 & 0.177 & 0 & 0 \\ 0 & 0 & 0.0761 & 0.1009 \\ 0.177 & 0 & 0 & 0 \\ 0 & 0 & 0.1009 & 0.1336 \end{pmatrix}.$$

and

$$M^{(1)} = \begin{pmatrix} 23 & 17 & 0 & 0 \\ 0 & 0 & 7 & 10 \\ 17 & 0 & 0 & 0 \\ 0 & 0 & 10 & 13 \end{pmatrix}.$$

Since $M^{(1)}_{0,1} = M^{(1)}_{1,2} + M^{(1)}_{1,3}$ it follows that $s=0$ and $d = 100 - \sum_{i=0}^{3}\sum_{j=0}^{3} M^{(1)}_{i,j} = 3$. Then,

$$M^{(2)} = \begin{pmatrix} 25 & 17 & 0 & 0 \\ 0 & 0 & 7 & 10 \\ 17 & 0 & 0 & 0 \\ 0 & 0 & 10 & 14 \end{pmatrix}$$

and

$$\mathcal{P}^{(2)} = \begin{pmatrix} 0.25 & 0.17 & 0 & 0 \\ 0 & 0 & 0.07 & 0.1 \\ 0.17 & 0 & 0 & 0 \\ 0 & 0 & 0.1 & 0.14 \end{pmatrix}$$

The entropy rate of $\mathcal{P}^{(2)}$, $H(\mathcal{P}^{(2)})$, is equal to 0.8103, i.e., $H(\widehat{\mathcal{P}}) - H(\mathcal{P}^{(2)}) \leq 0.0011$.

Appendix B
Proof of Lemma 3

In this appendix we prove Lemma 3 which states that

$$\frac{\log_2 |C^{(wl)}|}{n} \xrightarrow[n\to\infty]{} H(\mathcal{P}).$$

Proof: Recall that

$$\begin{aligned} C^{(wl)} &\overset{\text{def}}{=} \mathbb{C}(\pi(00)n, \mathcal{P}(001)n) \times \mathbb{C}(\pi(01)n, \mathcal{P}(011)n) \\ &\quad \times \mathbb{C}(\pi(10)n, \mathcal{P}(101)n) \times \mathbb{C}(\pi(11)n, \mathcal{P}(111)n) \end{aligned}$$

and therefore

$$|C^{(wl)}| = \binom{n\pi(00)}{n\mathcal{P}(001)}\binom{n\pi(01)}{n\mathcal{P}(011)}\binom{n\pi(10)}{n\mathcal{P}(101)}\binom{n\pi(11)}{n\mathcal{P}(111)}.$$

Since

$$\binom{n}{\alpha n} \approx 2^{nH(\alpha)},$$

where $H(\alpha) = -\alpha\log_2\alpha - (1-\alpha)\log_2(1-\alpha)$, we have that

$$\binom{n\pi(xy)}{n\mathcal{P}(xy1)} \approx 2^{n\pi(xy)H(Q(1|xy))}$$

[4]The matrix $\widehat{\mathcal{P}}$ was computed using MATLAB and hence its entries are slightly rounded.

and hence

$$\begin{aligned}\frac{\log_2 |C^{(wl)}|}{n} \xrightarrow[n\to\infty]{} &- \sum_{xy1\in E} \pi(xy) Q(1|xy) \log_2 Q(1|xy) \\ &- \sum_{xy0\in E} \pi(xy) Q(0|xy) \log_2 \ Q(0|xy)) \\ = \ &- \sum_{xyz\in E} \mathcal{P}(xyz) \log_2 \ Q(z|xy) = H(\mathcal{P}).\end{aligned}$$

□

Acknowledgement

The authors would like to thank Osamu Torii for his useful suggestions regarding the simulation of the row-by-row weakly bitline ICI coding scheme. They would also like to thank the anonymous reviewers for many insightful comments which contributed to the discussion in the paper.

References

[1] R. L. 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.

[2] A. Berman and Y. Birk, "Constrained flash memory programming," in *Proc. IEEE Symp. Inf. Theory*, St. Petersburg, Russia, Jul./Aug. 2011, pp. 2128–2132.

[3] A. Berman and Y. Birk, "Low-complexity two-dimensional data encoding for memory inter-cell interference reduction," in *Proc. 27th Conv. IEEE Israel (IEEEI)*, Eilat, Israel, Nov. 2012, pp. 1–5.

[4] W. G. Bliss, "Circuitry for performing error correction calculations on baseband encoded data to eliminate error propagation," *IBM Tech. Discl. Bull.*, vol. 23, pp. 4633–4634, 1981.

[5] A. Bhatia, S. Yang, and P. H. Siegel, "Precoding mapping optimization for magnetic recording channels," *IEEE Trans. Mag.*, vol. 50, no. 11, Nov. 2014, Art. no. 3102304.

[6] S. Buzaglo, E. Yaakobi, and P. H. Siegel, "Coding schemes for intercell interference in flash memory," in *Proc. IEEE Symp. Inf. Theory*, Hong Kong, Jun. 2015, pp. 1736–1740.

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

[8] N. G. deBruijn, "A combinatorial problem," *Roy. Netherlands Acad. Sci.*, vol. 49, pp. 758–764, Jun. 1946.

[9] 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.

[10] O. Elishco, T. Meyerovitch, and M. Schwartz, "Semiconstrained systems," in *Proc. IEEE Symp. Inf. Theory*, Hong Kong, Jun. 2015, pp. 246–250.

[11] O. Elishco, T. Meyerovitch, and M. Schwartz, "Semiconstrained systems," *IEEE Trans. Inf. Theory*, vol. 62, no. 4, pp. 1688–1702, Apr. 2016.

[12] P. A. Franaszek, "Sequence-state methods for run-length-limited coding," *IBM J. Res. Develop.*, vol. 14, pp. 376–383, Jul. 1970.

[13] F. R. Gantmacher *Matrix Theory, Vol. 2*. New York, NY, USA: Chelsea, 1960.

[14] S. Halevy, J. Chen, R. M. Roth, P. H. Siegel, and J. K. Wolf, "Improved bit-stuffing bounds on two-dimensional constraints," *IEEE Trans. Inf. Theory*, vol. 50, no. 5, pp. 824–838, May 2004.

[15] S. Halevy and R. M. Roth, "Parallel constrained coding with application to two-dimensional constraints," *IEEE Trans. Inf. Theory*, vol. 48, no. 5, pp. 1009–1020, May 2002.

[16] K. A. S. Immink, "Weakly constrained codes," *Electron. Lett.*, vol. 33, no. 23, pp. 1943–1944, Nov. 1997.

[17] K. A. S. Immink, "A practical method for approaching the channel capacity of constrained channels," *IEEE Trans. Inf. Theory*, vol. 43, no. 8, pp. 1389–1399, Sep. 1997.

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

[19] Y. Kim *et al.*, "Modulation coding for flash memories," in *Proc. IEEE Int Conf. Comput. Netw. Commun.*, San Diego, CA, USA, Jan. 2013, pp. 961–967.

[20] Y. Kim, R. Mateescu, S. H. Song, Z. Bandic, and B. V. K. V. Kumar, "Coding scheme for 3D vertical flash memory," in *Proc. IEEE Int. Conf. Commun.*, London, U.K., Jun. 2015, pp. 264–270.

[21] 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.

[22] M. Mansuripur, "Enumerative modulation coding with arbitrary constraints and post-modulation error correction coding and data storage systems," *Proc. SPIE* vol. 1499, pp. 72–86, Jul. 1991.

[23] B. H. Marcus, R. M. Roth, and P. H. Siegel, "Constrained systems and coding for recording channels," in *Handbook of Coding Theory*, V. Pless and W. Huffman, Eds. Amsterdam, The Netherland: Elsevier, 1998, pp. 1635–1764.

[24] B. E. Moision, A. Orlitsky, and P. H. Siegel, "On codes with local joint constraints," *Linear Algebra Appl.*, vol. 422, pp. 442–454, Apr. 2007.

[25] Z. Nagy and K. Zeger, "Bit-stuffing algorithms and analysis for run-length constrained channels in two and three dimensions," *IEEE Trans. Inf. Theory*, vol. 50, no. 12, pp. 3146–3169, Dec. 2004.

[26] E. Ordentlich and R. M. Roth, "Capacity lower bounds and approximate enumerative coding for 2-D constraints," in *Proc. IEEE Symp. Inf. Theory*, Nice, France, Jun. 2007, pp. 1681–1685.

[27] K. T. Park *et al.*, "Three-dimensional 128Gb MLC vertical NAND flash-memory with 24-WL stacked layers and 50MB/s high-speed programming," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Pap.*, Feb. 2014, pp. 334–335.

[28] 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, May 2014.

[29] R. M. Roth, P. H. Siegel, and J. K. Wolf, "Efficient coding schemes for the hard-square model," *IEEE Trans. Inf. Theory*, vol. 47, no. 3, pp. 1166–1176, Mar. 2001.

[30] A. Sharov and R. M. Roth, "Two-dimensional constrained coding based on tiling," *IEEE Trans. Inf. Theory*, vol. 56, no. 4, pp. 1800–1807, Apr. 2010.

[31] I. Tal, T. Etzion, and R. M. Roth, "On row-by-row coding for 2-D constraints," *IEEE Trans. Inf. Theory*, vol. 55, no. 8, pp. 3565–3576, Aug. 2009.

[32] I. Tal and R. M. Roth, "Bounds on the rate of 2-D bit-stuffing encoders," in *Proc. IEEE Symp. Inf. Theory*, Toronto, ON, Canada, Jul. 2008, pp. 1463–1467.

[33] 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.*, Jun. 2015, pp. 1868–1873.

[34] Y. Wang, L. A. D. Bathen, Z. Shao, and N. D. Dutt, "3D-FlashMap: A physical-location-aware block mapping strategy for 3D NAND flash memory," in *Proc. DATE*, Mar. 2012, pp. 1307–1312.

Sarit Buzaglo was born in Israel in 1983. She received the B.Sc. and M.Sc. degrees from the Department of Mathematics, Technion–Israel Institute of Technology, Haifa, Israel, in 2007 and 2010, respectively, and the Ph.D. degree from the Department of Computer Science, Technion–Israel Institute of Technology. She is currently a Post-Doctoral Researcher with the Center for Memory and Recording Research, at University of California, San Diego. Her research interests include coding theory, algebraic error-correction coding, coding for advanced storage devices and systems, and combinatorics. She received an award from the Weizmann Institute of Science–National Postdoctoral Award Program for Advancing Women in Science, in 2014.

Paul H. Siegel (M'82–SM'90–F'97) 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 Chair Weizmann Post-Doctoral 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, University of California at San Diego, La Jolla, CA, USA, in 1995, where he is currently a Professor of Electrical and Computer Engineering with the Jacobs School of Engineering. He is currently with the Center for Memory and Recording Research, where he holds an Endowed Chair and served as the Director from 2000 to 2011. His research interests include information theory and communications, particularly coding and modulation techniques, 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 a recipient of 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 was the co-recipient of the 1992 IEEE Information Theory Society Paper Award and the 1993 IEEE Communications Society Leonard G. Abraham Prize Paper Award. He was the 2015 Padovani Lecturer of the IEEE Information Theory Society. 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 served as an Associate Editor of Coding Techniques of the IEEE TRANSACTIONS ON INFORMATION THEORY from 1992 to 1995, and as Editor-in-Chief from 2001 to 2004. 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.