Coding Schemes for Inter-Cell Interference in Flash Memory

Sarit Buzaglo[*], **Paul H. Siegel**[*†], and **Eitan Yaakobi**[‡]
[*]Center for Magnetic Recording Research, University of California, San Diego, La Jolla, CA 92093-0401 USA
[†]Electrical and Computer Engineering, University of California, San Diego, La Jolla, CA 92093-0407 USA
[‡]Computer Science Department, Technion – Israel Institute of Technology, Haifa 32000, Israel
{*sbuzaglo, psiegel*}*@ucsd.edu*, {*yaakobi*}*@cs.technion.ac.il*

***Abstract*—Inter-cell interference (ICI) is a significant cause of errors in flash memories. In two-level (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 use of conventional constrained codes that forbid the $1\,0\,1$ pattern. This approach does not address the problem of vertical ICI, however. In this work, we present a row-by-row coding technique that eliminates vertical $1\,0\,1$ patterns while preserving the sequential wordline programming order. This scheme, though efficient, necessarily suffers a rate loss of almost 20%. We therefore propose another coding scheme, combining a relaxed constraint on vertical $1\,0\,1$ patterns with a systematic error correcting code, that can mitigate vertical ICI errors while achieving a higher overall code rate, provided that the vertical ICI error probability is sufficiently small.**

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 sizes decrease, however, 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 [10]. This phenomenon occurs in two-level (SLC) flash memory, and is particularly severe when programming multi-level cell (MLC) flash memory [3], [10]. It becomes even more pronounced in the recent designs of 3D flash [9], [13], [16].

It is known that 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 [1] [2], [15]. In fact, experimental results in [15] indicate that bitline ICI can be even more detrimental than wordline ICI in MLC flash, due in part to the sequential wordline programming protocol.

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, [1], [2], [8], [7], [12], [15]. 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 [2], but the encoding is variable-rate, and the decoding must be done in reverse row order.

The main goal of this work is the design of *bitline* ICI-mitigating coding techniques for SLC flash memory[1] that are compatible with the sequential programming order of horizontal wordlines. This is accomplished by allowing the encoding and decoding algorithms to read the previous two wordlines when writing or reading a particular wordline. 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, $C_{ICI} \approx 0.8114$. Our solution can be seen as an embodiment of the design method in [4], [14], where M-track parallel encoding for general 1D constraints is considered.

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 [5], that allows a nonzero probability of occurrence of vertical $1\,0\,1$ patterns along bitlines 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.

The rest of this paper is organized as follows. In Section II we introduce basic notation and definitions used throughout the paper. In Section III we present the efficient row-bv-row coding scheme that forbids the vertical $1\,0\,1$ pattern along bitlines. Section IV presents the hybrid coding scheme that allows a fixed proportion of $1\,0\,1$ patterns along bitlines and uses an error-correcting code to correct the resulting ICI errors. We conclude in Section V.

II. Preliminaries

In this section we present some of the basic definitions and notation used throughout the paper. For integers $a, b \in \mathbb{Z}$ where $a < b$, we define $[a, b] = \{a, a+1, \ldots, b\}$. We represent a block of SLC flash memory as a binary $m \times n$ array, composed of m *wordlines* of length n. For $1 \leq i \leq m$, we define the ith wordline as $WL_i = w_{i,1}w_{i,2}\ldots w_{i,n}$. For $1 \leq j \leq n$, the jth *bitline* is defined as $BL_j = w_{1,j}w_{2,j}\ldots w_{m,j}$.

[1]Although the discussion is framed in the context of SLC flash, the methods are equally applicable to page-oriented coding in MLC flash.

Given a set of indices $J = \{j_1, j_2, \ldots, j_k\}$, $j_1 \leq j_2 \leq \cdots \leq j_k$, and a sequence $\mathbf{x} = x_1 x_2 \ldots x_\ell$, $\ell \geq k$ we denote the restriction of $\mathbf{x}$ to J by $\mathbf{x}_J = x_{j_1} x_{j_2} \ldots x_{j_k}$. Similarly, for every $1 \leq i \leq m$, we denote the restriction of WL_i to the bit indices specified in J by $WL_{i,J} = w_{i,j_1} w_{i,j_2} \ldots w_{i,j_k}$. We assume throughout that the wordlines are programmed sequentially in a monotone increasing order, i.e., for every $1 \leq i_1 < i_2 \leq m$, WL_{i_1} is programmed before WL_{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*.

Let $\mathbb{C}$ be a row-by-row coding scheme. Let $R_1, R_2, \ldots, R_m$ be nonnegative real numbers, such that for every $1 \leq i \leq m$, $\mathbb{C}$ programs one of $2^{R_i n}$ possible messages into WL_i, i.e., R_i is the rate of the ith wordline. $\mathbb{C}$ is said to have a *fixed rate* R if $R_i = R$, for all $1 \leq i \leq m$.

A binary sequence $\mathbf{b} = b_1 b_2 \ldots b_m$ is said to satisfy the *ICI constraint* if $b_{i-1} b_i b_{i+1} \neq 101$, for all $2 \leq i \leq m-1$. Let S be the set of all binary sequences that satisfy the ICI constraint. By abuse of terminology, the set S is also called the ICI constraint where the meaning should be clear from the context. 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 shown in Fig. 1.

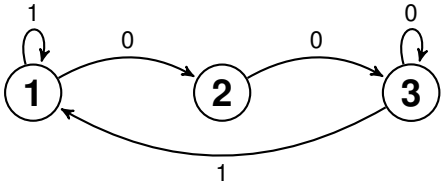

Fig. 1: The graph G representing the ICI constraint.

The capacity, C_{ICI}, of the ICI constraint is defined by

$$C_{ICI} = \lim_{n \to \infty} \frac{\log_2 |S \cap \{0,1\}^n|}{n}.$$

There is a simple expression for the capacity of this constraint in terms of the largest real eigenvalue of the adjacency matrix of G (see, for example, [11]), from which it follows that $C_{ICI} \approx 0.8114$. Conventional techniques can be used to construct efficient encoders and practical decoders for this constraint, and such codes have been proposed for use in magnetic recording; see [6] for a rate-$\frac{4}{5}$, sliding-block decodable, finite-state encoder.

A coding scheme $\mathbb{C}$ for an SLC flash memory is called a *bitline ICI constrained code* if for every $1 \leq j \leq n$, $BL_j \in S$. A *bitline ICI error* is the event where cells $w_{i-1,j}$, $w_{i,j}$, and $w_{i+1,j}$ are successively programmed to 1, 0, and 1, respectively, and due to the bitline ICI, $w_{i,j}$ gets changed to 1. We will assume that bitline ICI is the only source of errors.

Finally, we recall that a length n and weight w, a binary constant-weight code, denoted by $\mathbb{C}(n, w)$, is a subset of $\{0,1\}^n$ such that every $\mathbf{c} = c_1 c_2 \ldots c_n \in \mathbb{C}(n, w)$ has *Hamming weight* w, where the Hamming weight of $\mathbf{c}$ is defined as

$$w_H(\mathbf{c}) = |\{j \ : \ c_j \neq 0\}|.$$

III. Bitline ICI Constrained Codes

The goal of this section is to design row-by-row bitline ICI coding schemes with fixed rates. The key idea is simply that, when programming a given wordline, our encoder will take into account the two previously programmed wordlines. This approach is consistent with the practical requirement that wordlines are programmed consecutively, from top to bottom of the block. It is also compatible with the usual practice of programming all of the cells in a wordline simultaneously.

Specifically, our code design procedure involves two steps. In the first step, we conduct a probabilistic analysis, driven by the choice of a target code rate. In the second step, we give explicit constructions of bitline ICI constrained codes. The constructions make use of constant-weight codes whose parameters are determined by the results of the probabilistic analysis. This design methodology is closely related to the approach proposed in [4], [14] for constructing row-by-row coding schemes for general 1D constraints. It was shown in [4] that there exist such coding schemes that achieve the capacity of the 1D constraint, and an explicit construction, based on constant-weight codes, was given in [14]. We essentially apply the basic ideas of the construction in [14] to the ICI constraint, with some simplifications.

Step 1- Probabilistic Analysis: Let $P_1 : \{0,1\}^2 \to \mathbb{R}$ be a probability mass function, i.e., P_1 satisfies the following two properties:

(a.1) For all $x_1, x_2 \in \{0,1\}$, $P_1(x_1 x_2) \geq 0$.
(a.2) $\sum_{x_1, x_2 \in \{0,1\}} P_1(x_1 x_2) = 1$.

Let $P : \{0,1\}^3 \to \mathbb{R}$ be a conditional probability mass function, i.e., for every $x_1, x_2 \in \{0,1\}$, $0 \leq P(0|x_1 x_2) \leq 1$ and $P(1|x_1 x_2) = 1 - P(0|x_1 x_2)$. The probabilistic functions P_1 and P define a Markov process [11]. Let $\mathbf{b} = b_1 b_2 \ldots b_m$ be a random sequence that is generated by the Markov process. Then the following properties hold:

(b.1) $Prob(b_1 b_2 = x_1 x_2) = P_1(x_1 x_2)$.
(b.2) For all $i \geq 3$,
$Prob(b_i = x_3 | b_{i-2} b_{i-1} = x_1 x_2) = P(x_3 | x_1 x_2)$.
(b.3) For all $i \geq 3$, $Prob(b_i = x) =$
$\sum_{x_1, x_2 \in \{0,1\}} P(x|x_1 x_2) Prob(b_{i-2} b_{i-1} = x_1 x_2)$.
(b.4) $Prob(\mathbf{x}) = P_1(x_1 x_2) \prod_{i=3}^{m} P(x_i | x_{i-2} x_{i-1})$.

Since our goal is to design a fixed rate encoder, we assume that the Markov process is *stationary*, i.e., the sequence $\mathbf{b}$ also satisfies the following property:

(b.5) For all $i \geq 3$, $Prob(b_{i-2} b_{i-1} = x_1 x_2) = P_1(x_1 x_2)$.

From (b.5) we obtain the following system of equations, which are linear in $P_1(x_1 x_2)$:

$$\begin{array}{l} P_1(00)P(0|00) + P_1(10)P(0|10) = P_1(00) \\ P_1(00)P(1|00) + P_1(10)P(1|10) = P_1(01) \\ P_1(01)P(0|01) + P_1(11)P(0|11) = P_1(10) \\ P_1(01)P(1|01) + P_1(11)P(1|11) = P_1(11) \end{array}. \quad (1)$$

Combining (1) and property (a.2) of P_1 we can express $P_1(x_1 x_2)$ in terms of $P(0|\tilde{x}_1 \tilde{x}_2)$, namely

$$\begin{array}{c} P_1(00) = \frac{P(0|10)P(0|11)}{q} \\ P_1(10) = P_1(10) = \frac{(1-P(0|00))P(0|11)}{q} \\ P_1(11) = \frac{(1-P(0|00))(1-P(0|01))}{q}, \end{array} \quad (2)$$

where $q = P(0|10)P(0|11) + 2(1-P(0|00))P(0|11) + (1-P(0|00))(1-P(0|01))$.

The rate of the set of all sequences of length m that are formed by the stationary Markov process is defined by

$$R_m(P) = -\frac{1}{m}\sum_{\mathbf{x}\in\{0,1\}^m}\pi(\mathbf{x})\log_2\pi(\mathbf{x}) = \frac{1}{m}\cdot(H(P_1(x_1x_2)) + (m-2)H(P(x_3|x_1x_2))). \quad (3)$$

The rate of the stationary Markov process is defined by

$$R(P) = \lim_{m\to\infty} R_m(P) = H(P(x_3|x_1x_2)).$$

Substituting (2) in $H(P(x_3|x_1x_2))$, we derive an expression for $R(P)$ in terms of $P(0|x_1x_2)$. Since we wish to avoid the pattern $1\,0\,1$ along bitlines, we set $P(0|10) = 1$. We then choose $P(0|x_1x_2)$ (where $x_1x_2 \neq 10$) to maximize $R(P)$. It is well known (see, for example, [11]) that the conditional probabilities $P(0|x_1x_2)$ can be chosen such that $R(P)$ is the capacity of the one-dimensional ICI constraint. However, for the encoder that is presented in step 2 below we require that $P_1(x_1x_2)$ and $P(x_3|x_1x_2)$ be rational numbers. Since $R(P)$ is a continuous function, we can indeed assume that $P_1(x_1x_2)$ and $P(0|x_1x_2)$ are rational numbers that will yield an encoder with rate close to C_{ICI}, for sufficiently large n. (See [14].) Henceforward, we will assume that $P_1(x_1x_2)n$ and $P(0|x_1x_2)P_1(x_1x_2)n$ are integers.

Step 2- Encoding and Decoding: For $x_1, x_2 \in \{0,1\}$, let $p_{x_1} = Prob(b_1 = x_1) = P_1(x_1 0) + P_1(x_1 1)$, and let $p_{x_1x_2} = Prob(b_1b_2b_3 = x_1x_21) = P(1|x_1x_2)P_1(x_1x_2)$. Recall, that the constant-weight code that consists of all binary sequences of length n and weight w is denoted by $\mathbb{C}(n,w)$. For the encoding process, we will need the following three codes, defined in terms of constant-weight codes:

$$\begin{aligned}\mathbb{C}^1 &= \mathbb{C}(n, p_1 n)\\ \mathbb{C}^2 &= \mathbb{C}(p_0 n, P_1(01)n)\times\mathbb{C}(p_1 n, P_1(11)n)\\ \mathbb{C}^3 &= \mathbb{C}(P_1(00)n, p_{00}n)\times\mathbb{C}(P_1(01)n, p_{01}n)\times\\ &\quad\ \mathbb{C}(P_1(10)n, p_{10}n)\times\mathbb{C}(P_1(11)n, p_{11}n).\end{aligned} \quad (4)$$

Note that, since $P(1|10) = 0$, it follows that $\mathbb{C}(P_1(10)n, p_{10}n) = \{\mathbf{0}\}$. The codes $\mathbb{C}^1$ and $\mathbb{C}^2$ will be used to program the first two wordlines of the memory. The remaining wordlines will be programmed using the code $\mathbb{C}^3$. For every $1 \le i \le 3$, let $\mathcal{E}^i : \mathbb{Z}_{|\mathbb{C}^i|} \to \mathbb{C}^i$, $\mathcal{D}^i : \mathbb{C}^i \to \mathbb{Z}_{|\mathbb{C}^i|}$ be the encoder and decoder for $\mathbb{C}^i$, respectively.

Construction 1. *The encoding process is defined as follows:*

- *Initialize: Given a message $M_1 \in \mathbb{Z}_{|\mathbb{C}^1|}$, write $\mathcal{E}^1(M_1)$ into WL_1. Given a message $M_2 \in \mathbb{Z}_{|\mathbb{C}^2|}$, set $(\mathbf{c}_0, \mathbf{c}_1) = \mathcal{E}^2(M_2)$. For all $x \in \{0,1\}$, set $J_x = \{j : (\mathcal{E}^1(M_1))_j = x\}$. Write $\mathbf{c}_x$ into WL_{2,J_x}. Set $i \leftarrow 3$.*
- *Given a message $M \in \mathbb{Z}_{|\mathbb{C}^3|}$, read WL_{i-2}, WL_{i-1} and for every $x_1x_2 \in \{0,1\}$, set $J_{x_1x_2} = \{j : w_{i-2,j}w_{i-1,j} = x_1x_2\}$. Set $(\mathbf{c}_{00}, \mathbf{c}_{01}, \mathbf{c}_{10}, \mathbf{c}_{11}) = \mathcal{E}^3(M)$. Write $\mathbf{c}_{x_1x_2}$ into $WL_{i,J_{x_1x_2}}$. Set $i \leftarrow i+1$.*

Decoding of WL_i is defined as follows;

- *If $i = 1$, $M_1 = \mathcal{D}^1(WL_i)$. If $i = 2$, read the first line and find the sets of indices J_x, $x \in \{0,1\}$. $M_2 = \mathcal{D}^2(WL_{2,J_0}, WL_{2,J_1})$.*
- *For $i \ge 3$, read WL_{i-2}, WL_{i-1} and find $J_{x_1x_2}$, for all $x_1, x_2 \in \{0,1\}$. $M = \mathcal{D}^3(WL_{i,J_{00}}, WL_{i,J_{01}}, WL_{i,J_{10}}, WL_{i,J_{11}})$.*

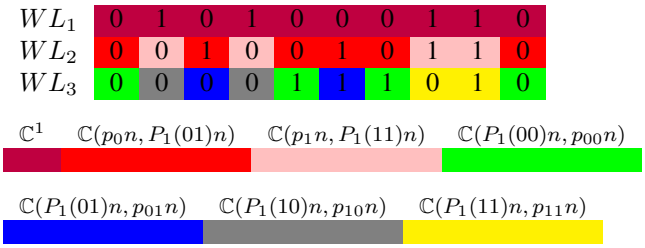

Fig. 2: Programming the first three wordlines with probabilities $P(0|x_1x_2) = 0.5$, for all $x_1x_2 \in \{0,1\}^2$, $x_1x_2 \neq 10$, and asymptotic rate 0.8. The color of a cell indicates to which constant-weight code it belongs. The gray cells must be programmed to 0.

Fig. 2 illustrates, for $n = 10$, the first three programmed wordlines obtained by encoding according to Construction 1, with probabilities $P(0|x_1x_2) = 0.5$, for all $x_1, x_2 \in \{0,1\}$, $x_1x_2 \neq 10$. The asymptotic code rate is 0.8.

Corollary 1. *For every $\epsilon > 0$ and sufficiently large m, n, there exist probability mass functions P_1 and P such that the coding scheme described in Construction 1 produces a row-by-row bitline ICI constrained coding scheme of rate greater than $C_{ICI} - \epsilon$.*

IV. Bitline ICI Error Correcting 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. If we instead consider conditional probability mass functions P that allow $P(1|10)$ to be positive, we can program into the memory an $m \times n$ array such that for every i, the pattern $1\,0\,1$ appears vertically in rows $i-2, i-1, i$ exactly $P(1|10)P_1(10)n$ times. Note that for such P, the rate $R(P)$ can exceed C_{ICI}. Let α be the probability that a bitline pattern $1\,0\,1$ changes to $1\,1\,1$ due to an ICI error (we assume α is a known parameter of the flash memory device.) Our coding technique will guarantee a fixed probability of a bitline ICI error in a cell. By combining it with a systematic error correcting code, we get a coding scheme that mitigates bitline ICI and attains a fixed rate that exceeds the capacity of the one-dimensional ICI constraint.

Let $\mathbb{C}^1, \mathbb{C}^2, \mathbb{C}^3$ be as defined in (4) and let $\mathbb{C}$ be a systematic error-correcting code of length $n + r$ that can correct up to $P_1(10)P(1|10)\alpha n$ errors. Let $\mathcal{E} : \{0,1\}^n \to \mathbb{C}$ and $\mathcal{D} : \mathbb{C} \to \{0,1\}^n$ be an encoder and decoder for $\mathbb{C}$, respectively. As before, we denote the first n cells of the ith row by WL_i. We denote the r parity check bits of the ith row by $PB_i = pb_{i,1}pb_{i,2}\ldots\ldots pb_{i,r}$, i.e., $PB_i = (\mathcal{E}(WL_i))_{[n+1,n+r]}$. For simplicity, we neglect the bitline ICI errors in PB_i. In practice, PB_i could be encoded as in Section III to avoid the pattern $1\,0\,1$ along bitlines. In that case the number of redundancy symbols (the length of PB_i) will be $\tilde{r} \approx \lceil \frac{r}{0.8114} \rceil$. As before, we program WL_i according to the sequences that are programmed into WL_{i-2} and WL_{i-1}. However, upon programming WL_i, the wordline WL_{i-2} may suffer from bitline ICI errors (when WL_{i-1} was programmed). To overcome these errors we can use the error-correcting code on the $(i-2)$-nd wordline to correct the errors. Alternatively, to improve the encoding speed we use an n-bit side memory (for example DRAM

memory) which stores the information that was written on the wordline WL_{i-2} and thus has no ICI errors. The sequence stored in the side memory is denoted by $\mathbf{s} = s_1 s_2 \dots s_n$. Upon programming WL_i, $\mathbf{s}$ stores the sequence that was programmed into WL_{i-2}, before bitline ICI errors had occur.

Construction 2. *Define the encoder $\mathcal{E}_{ICI}$ as follows.*

- *Initialize: Given a message $M_1 \in \mathbb{Z}_{|\mathbb{C}^1|}$, write $\mathcal{E}^1(M_1)$ into WL_1. Given a message $M_2 \in \mathbb{Z}_{|\mathbb{C}^2|}$, set $(\mathbf{c}_0, \mathbf{c}_1) = \mathcal{E}^2(M_2)$. For all $x \in \{0,1\}$, set $J_x = \{j : (\mathcal{E}^1(M_1))_j = x\}$. Write $\mathbf{c}_x$ into WL_{2,J_x} and write $(\mathcal{E}(WL_2))_{[n+1,n+r]}$ into PB_2. Set $\mathbf{s} = WL_2$. Given a message $M_3 \in \mathbb{Z}_{|\mathbb{C}^3|}$, read WL_1, $\mathbf{s}$, and for every $x_1x_2 \in \{0,1\}$, set $J_{x_1x_2} = \{j : w_{1,j}\mathbf{s}_j = x_1x_2\}$. Set $(\mathbf{c}_{00}, \mathbf{c}_{01}, \mathbf{c}_{10}, \mathbf{c}_{11}) = \mathcal{E}^3(M)$. Program $\mathbf{c}_{x_1x_2}$ into $WL_{3,J_{x_1x_2}}$ and $(\mathcal{E}(WL_3))_{[n+1,n+r]}$ into PB_3. Set $i \leftarrow 4$.*
- *Given $M \in \mathbb{Z}_{|\mathbb{C}^3|}$, read $\mathbf{s}, WL_{i-1}$ and for every $x_1x_2 \in \{0,1\}$, set $J_{x_1x_2} = \{j : s_j w_{i-1,j} = x_1x_2\}$. Set $(\mathbf{c}_{00}, \mathbf{c}_{01}, \mathbf{c}_{10}, \mathbf{c}_{11}) = \mathcal{E}^3(M)$. Set $\mathbf{s} = WL_{i-1}$ and then write $\mathbf{c}_{x_1x_2}$ into $WL_{i,J_{x_1x_2}}$. Program $PB_i = (\mathcal{E}(WL_i))_{[n+1,n+r]}$. Set $i \leftarrow i+1$.*

Define the decoder $\mathcal{D}_{ICI}$ as follows.

- *If $i = 1$, return $\tilde{M}_1 = \mathcal{D}^1(WL_i)$. If $i = 2$, set $\mathbf{u}_2 = \mathcal{D}(WL_2PL_2)$ and then read the first line and find the sets of indices J_x, $x \in \{0,1\}$. Return $\tilde{M}_2 = \mathcal{D}^2(\mathbf{u}_{2,J_0}, \mathbf{u}_{2,J_1})$. If $i = 3$, set $\mathbf{u}_2 = \mathcal{D}(WL_2PB_2)$, $\mathbf{u}_3 = \mathcal{D}(WL_3PB_3)$ and for all $x_1, x_2 \in \{0,1\}$, set $J_{x_1x_2} = \{j : w_{1,j}u_{2,j} = x_1x_2\}$. Return $\tilde{M}_3 = \mathcal{D}^3(\mathbf{u}_{3,J_{00}}, \mathbf{u}_{3,J_{01}}, \mathbf{u}_{3,J_{10}}, \mathbf{u}_{3,J_{11}})$*
- *For $i \geq 4$, set $\mathbf{u}_k = \mathcal{D}(WL_kPB_k)$, $k = i-2, i-1, i$. For all $x_1, x_2 \in \{0,1\}$ set $J_{x_1x_2} = \{j : u_{i-2,j}u_{i-1,j} = x_1x_2\}$. Return $\tilde{M} = \mathcal{D}^3(\mathbf{u}_{i,J_{00}}, \mathbf{u}_{i,J_{01}}, \mathbf{u}_{i,J_{10}}, \mathbf{u}_{i,J_{11}})$.*

Fig. 2 illustrates, for $n = 8$, the first four wordlines obtained by programming according to Construction 2, with probabilities $P(0|x_1x_2) = 0.5$, for all $x_1, x_2 \in \{0,1\}$. The rate of the code is $\frac{n}{n+r} = \frac{8}{8+r}$.

For every $i \geq 2$, let $\tilde{\mathbf{c}}_i = \tilde{c}_{i,1}, \tilde{c}_{i,2}, \dots, \tilde{c}_{i,n}$ be the binary sequence that was programmed to WL_i (before bitline ICI errors may have occurred) using the encoder defined in Construction 2 and let

$$N_i = |\{j : \tilde{c}_{i-1,j}\tilde{c}_{i,j}\tilde{c}_{i+1,j} = 101\}|.$$

Lemma 1. *For every $2 \leq i \leq m$,*

$$N_i = P_1(10)P(1|10)n.$$

Proof. Since $\tilde{\mathbf{c}}_i$ is formed by the constant-weight codes defined in (4) according to the stationary Markov process defined by P_1 and P, the claim follows. □

Lemma 2. *For every i, j, $2 \leq i \leq m$ and $1 \leq j \leq n$,*

$$Prob(w_{i,j} \neq \tilde{c}_{i,j}) = P_1(10)P(1|10)\alpha.$$

Proof. Since we program the wordlines according to the stationary Markov process defined by the probability mass functions P_1 and P, it follows that for every $2 \leq i \leq m$, $1 \leq j \leq n$, $Prob(\tilde{c}_{i-1,j}\tilde{c}_{i,j}\tilde{c}_{i+1,j} = 101) = P_1(10)P(1|10)$. We also have that

$$Prob(w_{i,j} \neq \tilde{c}_{i,j}) = Prob(w_{i,j} = 1, w_{i-1,j}\tilde{c}_{i,j}\tilde{c}_{i+1,j} = 101).$$

If $\tilde{c}_{i,j} = 0$ then $w_{i-1,j} = \tilde{c}_{i-1,j}$ and therefore,

$$Prob(w_{i,j} \neq \tilde{c}_{i,j}) =$$
$$Prob(w_{i,j} = 1, \tilde{c}_{i-1,j}\tilde{c}_{i,j}\tilde{c}_{i+1,j} = 101) =$$
$$P_1(10)P(1|10)\alpha.$$

□

Lemma 3. *For every i, j, $2 \leq i \leq m$ and $1 \leq j \leq n$,*

$$Prob(w_{i,j} = 1|\tilde{c}_{i,j} = 0) = \frac{P_1(10)P(1|10)\alpha}{P_1(00) + P_1(10)}.$$

Proof. Since we program the wordlines according to the stationary Markov process defined by the probability mass functions P_1 and P, it follows that for every $2 \leq i \leq m$ and $1 \leq j \leq n$, $Prob(\tilde{c}_{i,j} = 0) = P_1(00) + P_1(10)$. We also have that

$$Prob(w_{i,j} = 1|\tilde{c}_{i,j} = 0) = \frac{Prob(w_{i,j} = 1, \tilde{c}_{i,j} = 0)}{Prob(\tilde{c}_{i,j} = 0)}.$$

By Lemma 2 the claim follows. □

Theorem 1. *For every $2 \leq i \leq m$, let M be the message that was stored in WL_i by $\mathcal{E}_{ICI}$ of Construction 2. If a fraction of at most α of the $P_1(10)P(1|10)n$ appearances of the pattern $1\,0\,1$ along bitlines of $\tilde{c}_{i-1}, \tilde{c}_i$, and $\tilde{c}_{i+1}$ were changed to $1\,1\,1$, then $\mathcal{D}_{ICI}(WL_i) = M$. Moreover, the rate of the code that is defined in Construction 2 is*

$$\frac{R_m(P)n}{n+r},$$

where $R_m(P)$ is as defined in (3).

Proof. For $i = 1$, since no bitline ICI error can occur in WL_1 it follows that $WL_1 = \mathcal{E}^1(M)$ and $\mathcal{D}_{ICI}(WL_1) = \mathcal{D}^1(WL_1) = M$. For every $i \geq 2$, since we assume no more than $\alpha P_1(10)P(1|10)n$ errors occur in WL_i and since $\mathbb{C}$ can correct up to $\alpha P_1(10)P(1|10)n$ errors it follows that $\mathbf{u}_i = \mathcal{D}(WL_iPL_i)$ stores the original sequence that was programmed to WL_i. Hence, for $i = 2$, $(\mathbf{u}_{2,J_0}, \mathbf{u}_{2,J_1}) = \mathcal{E}^2(M)$, and $\mathcal{D}_{ICI}(WL_2) = \mathcal{D}^2(\mathbf{u}_{2,J_0}, \mathbf{u}_{2,J_1}) = M$. For every $3 \leq i \leq m$, $(\mathbf{u}_{i,J_{00}}, \mathbf{u}_{i,J_{01}}, \mathbf{u}_{i,J_{10}}, \mathbf{u}_{i,J_{11}}) = \mathcal{E}^3(M)$, and $\mathcal{D}_{ICI}(WL_i) = \mathcal{D}^3(\mathbf{u}_{i,J_{00}}, \mathbf{u}_{i,J_{01}}, \mathbf{u}_{i,J_{10}}, \mathbf{u}_{i,J_{11}}) = M$.

The rate of the code can be readily verified. □

Let $\epsilon(P, \alpha) = \frac{P_1(10)P(1|10)\alpha}{P_1(00)+P_1(10)}$. From Lemma 3, $\epsilon(P, \alpha) = Prob(w_{i,j} = 1|\tilde{c}_{i,j} = 0)$, for all $2 \leq i \leq m, 1 \leq j \leq n$. Let $\{\mathbb{C}_n\}_{n=1}^{\infty}$ be a collection of binary systematic error correcting codes, where $\mathbb{C}_n$ is of length n, which asymptotically achieves the capacity of the binary symmetric channel (BSC). If we use $\mathbb{C}_n$ in Construction 2, then, as m and n tend to infinity, the rate of our coding scheme tends to

$$R_{symm}(P, \alpha) = \lim_{m\to\infty, n\to\infty} \frac{R_m(P)n}{n+r} = R(P)(1 - H(\epsilon(P, \alpha))) \quad (5)$$

and we conjecture that the average error probability goes to zero.

Since the bitline ICI errors are asymmetric errors, we can use a binary systematic code of length n for the Z-channel in Construction 2. The symmetric information rate (SIR) of the Z-channel, where $P(Y = 1|X = 0) = p$, is given by $H(\frac{1-p}{2}) - \frac{1}{2}H(p)$, which is achievable with random linear

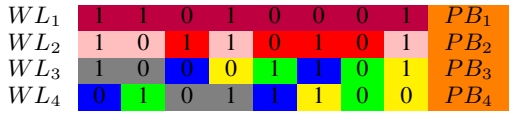

Fig. 3: Programming the first four wordlines with probabilities $P(0|x_1x_2) = 0.5$, for all $x_1x_2 \in \{0,1\}^2$. The color of a cell indicates to which constant-weight code it belongs. Note, that the gray color of $w_{3,1}$ indicates that $w_{2,1}$ was changed to 1 by a bitline ICI error.

codes. Assuming that we have an infinite collection of SIR-achieving binary systematic codes, the rate of our coding scheme tends to

$$R_{asymm}(P,\alpha) = \lim_{m\to\infty, n\to} \frac{R_m(P)n}{n+r} = R(P)\left(H(\tfrac{1-\epsilon(P,\alpha)}{2}) - \tfrac{1}{2}H(\epsilon(P,\alpha))\right) \quad (6)$$

and we again conjecture that the average error probability goes to zero.

Using (2) we can derive expressions for $R_{symm}(P,\alpha)$ and $R_{asymm}(P,\alpha)$ in terms of $P(0|x_1x_2)$ and α. We then define

$$R_{symm}(\alpha) = \max_{P(0|x_1x_2):\ x_1,x_2\in\{0,1\}} \{R_{symm}(P,\alpha)\}$$

and

$$R_{asymm}(\alpha) = \max_{P(0|x_1x_2):\ x_1,x_2\in\{0,1\}} \{R_{asymm}(P,\alpha)\}.$$

Figure 4 shows the graphs of $R_{symm}(\alpha)$ and $R_{asymm}(\alpha)$. Note that both rates usually exceed C_{ICI}. Although $R_{asymm}(\alpha)$ exceeds $R_{symm}(\alpha)$, less is known about construction of good codes for the Z-channel than for the BSC. For the case in which we don't neglect the bitline ICI errors in PB_i, we assumed that the number of redundancy bits is $\tilde{r} \approx \lceil \frac{r}{0.8114} \rceil$ and similarly calculated the corresponding rates $\widetilde{R}_{symm}(\alpha)$ for symmetric codes and $\widetilde{R}_{asymm}(\alpha)$ asymmetric codes. Their plots are also depicted in Figure 4.

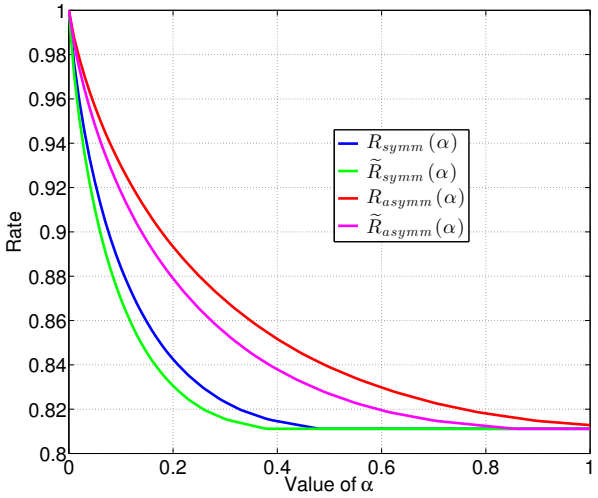

Fig. 4: The graphs of $R_{symm}(\alpha)$, $R_{asymm}(\alpha)$, $\widetilde{R}_{symm}(\alpha)$, and $\widetilde{R}_{asymm}(\alpha)$.

V. Conclusion

In this paper, two row-by-row coding schemes were suggested to handle bitline ICI errors in flash memories. The first coding scheme uses ideas from [14] to eliminate the pattern $1\,0\,1$ along bitlines, and the resulting code rate asymptotically attains the 1D ICI capacity. The second coding scheme maintains a nonzero probability of occurrence of the vertical pattern $1\,0\,1$ along bitlines and uses a systematic error-correcting code on wordlines, allowing an overall coding rate that exceeds the ICI capacity. Although not discussed here, the proposed methods have been use to handle bitline as well as wordline ICI errors in MLC flash memory. Designing efficient, practical row-by-row coding schemes for 2D constraints in flash memory remains a challenging problem. Practical issues such as throughput, latency, and cache lifetime also require careful study.

Acknowledgement

This work was supported in part by NSF Grant CCF-1405119, the Israel Science Foundation (ISF) Grant No. 1624/14, the ISEF Foundation, and the Weizmann Institute of Science -National Postdoctoral Award Program for Advancing Women in Science. The third author was supported in part at the Technion by a fellowship from the Lady Davis Foundation and a Viterbi Leaders Fellowship.

References

[1] A. Berman and Y. Birk, "Constrained flash memory programming," *Proc. IEEE Symp. Inf. Theory,* pp. 2128–2132, St. Petersburg, Russia, July–August, 2011.

[2] A. Berman and Y. Birk, "Low-complexity two-dimensional data encoding for memory inter-cell interference reduction," *Proc. 27th Conv. IEEE Israel (IEEEI)*, Eilat, Israel, November 14-17, 2012.

[3] 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.,* pp. 2718–2728, vol. 57, 2010.

[4] S. Halevy and R. M. Roth, "Parallel constrained coding with application to two-dimensional constraints," *IEEE Trans. Inform. Theory*, vol. 48, pp. 1009–1020, 2002.

[5] K.A. S. Immink, "Weakly constrained codes," *Electron. Lett.*, vol. 33, no. 23, Nov. 1997, pp. 1943-4.

[6] R. Karabed and P. H. Siegel, "Coding for higher order partial response channels," *Proc. 1995 SPIE Int. Symp. on Voice, Video, and Data Communications*, Philadelphia, PA, Oct. 1995, vol. 2605, pp. 115–126.

[7] S. Kayser and P.H. Siegel, "Constructions for constant-weight ICI-free codes," *Proc. IEEE Symp. Inf. Theory,* pp. 1431–1435, Honolulu, Hawaii, June–July, 2014.

[8] Y. Kim, B. Kumar, K. L. Cho, H. Son, J. Kim, J. J. Kong, and J. Lee, Modulation coding for flash memories," in *Proc. ICNC* , San Diego, CA, Jan. 28–21, 2013, pp. 961-967.

[9] Y. Kim, R. Mateescu, S.-H. Song, Z. Bandic, and B.V.K.V. Kumar, "Coding scheme for 3D vertical flash memory," submitted *Proc. IEEE Int'l Conf. Communications*, June 2015.

[10] J.-D. Lee, S.-H. Hur, and J.-D. Choi, "Effects of floating-gate interference on NAND flash memory cell operation," *IEEE Electron Dev. Lett.,* pp. 264–266, vol. 23, 2002.

[11] 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: Elsevier, 1998, pp. 1635–1764.

[12] 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 Comm.,* pp. 836–846, vol. 32, 2014.

[13] K.-T. Park et al., "Three-dimensional 128Gb MLC vertical NAND Flash-memory with 24-WL stacked layers and 50MB/s high-speed programming," *Proc. IEEE Int'l Solid-State Circuits Conf. Dig. Tech. Pap.* pp. 334–335, February 2014.

[14] I. Tal, T. Etzion, and R. Roth, "On row-by-row coding for 2-D constraints," *IEEE Trans. on Inform. Theory*, vol. 55, pp. 3565–3576, 2009.

[15] V. Taranalli, H. Uchikawa, and P.H. Siegel, "Error analysis and inter-cell interference mitigation in multi-level cell flash memories," *IEEE Int'l Conf. Communications*, June 2015.

[16] Y. Wang, L.A.D. Bahten, Z. Shao, and N.D. Dutt, "3D-FlashMap: A physical-location-aware block mapping strategy for 3D NAND flash memory," *Proc. Design, Automation, Test in Europe Conference Exhibition (DATE),* pp. 1307–1312, March, 2012.