Spread Programming for NAND flash

Tianqiong Luo and Borja Peleato
Electrical and Computer Engineering
Purdue University
West Lafayette IN 47907 Email: {luo133,bpeleato}@purdue.edu

***Abstract*—The aggressive scaling of NAND flash memories has caused significant degradation in their reliability and endurance. One of the dominant factors in this degradation is the inter-cell-interference (ICI), by which the programming of a cell can affect near-by neighboring cells corrupting the information that they store. This paper proposes a new data representation scheme which increases endurance and significantly reduces the probability of error caused by ICI. The method is based on using an orthogonal code to spread each bit across multiple cells, resulting in a more uniform distribution of voltages being programmed in the cells.**

I. Introduction

NAND Flash is a non-volatile memory technology which offers significantly higher speeds and power efficiency than hard drives, but its higher cost is still an obstacle for its widespread use. Manufacturers have aggressively scaled the technology to pack more cells in the same silicon real state, while they also increased the number of bits stored in each cell. These techniques succeeded in reducing the cost of flash memories to the same order of magnitude as that of hard drives, but they brought other problems, mainly related to the reliability of the stored information.

A flash cell is a floating gate transistor whose threshold voltage can be adjusted by injecting charges into its floating gate. Information is stored by setting this voltage threshold to specific values. In its simplest form, one bit is stored in each cell, depending on whether it is charged or discharged. Memories of this type are known as SLC. In order to increase the capacity (and reduce their cost accordingly) most applications now use MLC memories, which can be programmed to four different voltage levels and store two bits in each cell. Some manufacturers have gone even further, producing memories which store three (TLC) or even four bits in each cell [1].

As flash memory technology scales and more bits are stored in each cell, the signal to noise ratio observed in the programmed voltages decreases. One of the main sources of noise, which is becoming increasingly important as the technology scales and for the forthcoming 3D flash structures, is inter-cell interference (ICI). ICI noise is due to the parasitic capacitance-coupling effect, by which the shift in threshold voltage of one cell can change the threshold voltage of its neighbors [2].

Additionally, flash cells have a limited lifetime. Before data can be written to a page, the block must have been erased[1]. The tunneling of charges into and out of the floating gate causes damage to the dielectric barrier that holds the charges, limiting the range of programmed voltages and the number of times that each cell can be written. The amount of damage that a cell suffers in a single write operation increases super-linearly with the programmed voltage. Hence, writing data patterns that are represented by a lower threshold voltage could prolong the lifetime of the flash [3], [4], [5].

This paper studies a new data representation scheme which, among other features, reduces ICI and extends the lifetime of the memory by reducing the frequency with which the largest voltage levels are programmed. It is based on using an orthogonal code to spread each information symbol across multiple cells, similar to how DS-CDMA is used in wireless communications.

Unfortunately, the voltage of the cells in a Flash memory cannot be measured directly. Reading the cells in a page is done by comparing their stored voltage with an adjustable reference voltage t. The read operation returns a binary vector with one bit for each cell: 1 if the cell has voltage lower than t and 0 otherwise. Section III-B and Fig. 1 will show that the proposed scheme increases the number of voltage levels for each cell, so reading a page might take longer than with the usual data representation scheme. However, many memory manufacturers now incorporate dedicated hardware to perform multiple reads of the same page with different values of t to obtain a finer quantization of the voltages. These commands are generally used to perform soft reads for LDPC decoding, but they can also be used to accelerate the read of additional levels that our scheme proposes. The penalty in terms of read speed might therefore not be as severe as it seems at first.

The rest of the paper is organized as follows. Section II introduces the system model used in the rest of the paper. Section III explains and analyzes the spreading data representation approach, which is then improved in Section IV. Finally, Section V presents simulation results to validate the method and Section VI summarizes and concludes the paper.

II. System model

In order to better illustrate the features of the proposed scheme, this paper will consider multiple scenarios with different noise distributions and memory types. From a high level perspective, it will be assumed that in a write operation the host provides a vector of (possibly encoded) information symbols $\mathbf{b} \in \mathcal{X}^m$ from an alphabet $\mathcal{X}$, which are then mapped to a vector of voltages $\mathbf{v}^0$ to be programmed on the cells.

[1]Cells in a NAND flash are grouped into pages, which is the smallest unit for write and read operations. Pages are grouped into blocks, which is the elementary unit for erase operations

By the time that the cells are read, the voltages $\mathbf{v}^0$ will have suffered some amount of white Gaussian noise, denoted $\mathbf{n}^w$, as well as inter-cell interference (ICI), denoted $\mathbf{n}^{\mathrm{ICI}}$. Therefore, the voltage actually stored in the cells at read time is

$$\mathbf{v} = \mathbf{v}^0 + \mathbf{n}, \qquad \mathbf{n} = \mathbf{n}^w + \mathbf{n}^{\mathrm{ICI}}. \tag{1}$$

The noise due to leakage is also assumed to be Gaussian and is therefore absorbed into the $\mathbf{n}^w$ term.

ICI occurs when a shift in the threshold voltage of one cell changes the threshold voltage of its neighbors due to the parasitic capacitance between cells, known as "floating-gate interference" [6]. Extensive measurements have shown that the change in threshold voltage suffered by the victim cell is proportional to the threshold voltage of the aggressor cell, with a proportionality factor that depends on the parasitic capacitance between the aggressor cell and the victim cell. This factor is commonly known as coupling ratio and will be denoted by γ. Hence,

$$\mathbf{n}^{\mathrm{ICI}} = \gamma \mathbf{v}_{\mathrm{aggressor}}$$

With the usual data representation scheme, each symbol b_i is mapped to a fixed nominal voltage v_i^0. So, for the sake of simplicity, it will be assumed that they both share the same alphabet $\mathcal{X}$ and $b_i = v_i^0$. In SLC memories these symbols are binary, in MLC they can take four values (representing two bits of information), in TLC they take 8 values (3 bits), etc. In general, the number of levels is chosen to be as large as possible while still avoiding potential overlap between the levels and excessive damage when programming the largest voltage.

Damage to flash memory cells can be caused by program/erase (P/E) cycling. According to [3], most of the damage happens when cells are programmed to the largest voltage, so writing data patterns that are represented by a lower threshold voltage could prolong the lifetime of the flash [3], [4], [5].

The proposed data representation scheme will use a different linear mapping between the symbols $\mathbf{b}$ and the nominal voltages $\mathbf{v}^0$, to be described in the next section. This mapping will extend the number of nominal voltages to be programmed, but also reduce the number of cells programmed to large voltages, attenuate the ICI, and increase robustness to impulsive noise. This will result in increased capacity and extended lifetime for the memory.

In practice it is not possible to program NAND flash cells with a negative voltage. The discharged state in which the cells are left after being erased sets a lower limit for the range of programmable voltages and the write procedure can only push the cells towards higher voltages. However, for our derivations it will be useful to shift the reference system so that the range of programmable voltages is symmetric and the voltages $\mathbf{v^0}$ and symbols $\mathbf{b}$ can take both positive and negative values. SLC cells will therefore have their voltage levels relabeled as -0.5 and $+0.5$, while MLC cells will be assumed to take voltage levels -1.5, -0.5, 0.5, and 1.5. In general, if there are $2S$ symbols in the alphabet $\mathcal{X}$, they will be labeled as $\mathcal{X} = \{\pm 0.5, \pm 1.5, \ldots, \pm(S-0.5)\}$. The largest symbol in the alphabet will be denoted by $V_{\max} = S - 0.5$. The rest of the paper assumes that the symbols b_i and voltages v_i^0 have zero mean.

III. The spreading approach

In a traditional flash memory, each cell stores a fixed number of bits. There is usually some redundant bits introduced by the ECC or RAID schemes but, ultimately, each bit is stored in a specific cell. Cells have a fixed number of voltage levels to which they can be programmed and all the levels are written with the same frequency. This section proposes a new data representation scheme which uses orthogonal codes to spread each bit across multiple cells, similar to DS-CDMA transmission in wireless communications. This data representation scheme reduces the variability of the voltages being programmed in the cells, resulting in improved endurance and additional robustness towards impulsive noise and ICI.

Instead of mapping each symbol b_i to a fixed voltage v_i^0, the proposed scheme uses a matrix with orthogonal columns $\mathbf{C}$ (e.g., a Walsh matrix) to map the symbols $\mathbf{b}$ into voltages $\mathbf{v}^0$ to be programmed. For example, when mapping four symbols $b \in \mathcal{X}^4$ into four cells, the voltages to be programmed are:

$$\begin{bmatrix} v_1^0 \\ v_2^0 \\ v_3^0 \\ v_4^0 \end{bmatrix} = \frac{k}{4} \cdot \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix},$$

where k is an adjustable parameter that controls the range of voltages being programmed. Increasing k introduces more separation between the programmed voltage levels, but also increases the damage suffered by the cells and the power consumed. In general, when M symbols are to be programmed into $N \geq M$ cells,

$$\mathbf{v}^0 = \frac{k}{M}\mathbf{C}\mathbf{b}, \tag{2}$$

where $\mathbf{C}$ is a $\{-1, 1\}^{N \times M}$ matrix with orthogonal columns. We will refer to this operation as spreading.

By spreading each information symbol across multiple cells, we increase the number of possible programmed voltages in each cell, so symbols and nominal voltages no longer share the same alphabet. For example, for the SLC case where $b_i \in \{-0.5, 0.5\}$, Eq. 2 yields five possible levels for each cell: $v_i^0 \in \{-\frac{k}{2}, -\frac{k}{4}, 0, \frac{k}{4}, \frac{k}{2}\}$. In general, if $V_{\max}$ is the largest symbol in the alphabet $\mathcal{X}$, the voltage levels after spreading are in the range $[-kV_{\max}, kV_{\max}]$.

When the read operation is performed, the voltages are multiplied by a de-spreading matrix $\mathbf{C}^T$, which is the left inverse of the spreading matrix. Because of the properties of Walsh sequences, the de-spreading matrix is the transpose of the spreading matrix. Continuing with the previous example, when $N = M = 4$

$$\begin{bmatrix} \hat{b_1} \\ \hat{b_2} \\ \hat{b_3} \\ \hat{b_4} \end{bmatrix} = \frac{1}{k} \cdot \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix},$$

where $\hat{b_i},\ i = 1,2,3,4$ represent the information estimates after reading. In general,

$$\hat{\mathbf{b}} = \frac{M}{Nk}\mathbf{C}^T\mathbf{v}, \tag{3}$$

Combining Eqs. (1), (2), and (3), the estimated information symbols can be represented as:

$$\hat{b_i} = b_i + \frac{M}{Nk}\sum_{j=1}^{N} \pm n_j, \quad i = 1,2,\ldots,M. \tag{4}$$

The noise can be arbitrarily attenuated by decreasing $\frac{M}{Nk}$, but that involves sacrificing capacity by decreasing $\frac{M}{N}$ or using a wider range of programmed voltages by increasing k. Since most practical applications are not willing to compromise capacity, the rest of the paper assumes $M = N$, which means that the storage space efficiency is the same as in the regular scheme.

A. Effect of Gaussian noise

For fixed voltage range $(k = 1)$ and signal-independent Gaussian noise, spreading actually decreases the signal-to-noise ratio (SNR) at read time. Assuming independent and identically distributed noise components $n_i \sim \mathcal{N}(0, \sigma^2)$ [7], the SNR of the regular and spreading schemes are:

$$SNR_{\text{regular}} = \frac{P_s}{\sigma^2} \qquad SNR_{\text{spread}} = \frac{P_s}{\frac{N}{k^2}\sigma^2}, \tag{5}$$

where $P_s = E[b_i^2]$ represents the power of the stored symbols. It is easy to increase SNR_{spread} when needed by increasing the scaling constant k, but doing so widens the range of programmed voltages and thus causes more damage and consumes more power. This subsection studies such tradeoff.

One of the advantages of the spreading scheme is that it reduces the probability of programming the maximum voltage, thus reducing the damage to the flash memory. The amount of damage suffered by the memory is approximately proportional to the square of the voltage programmed. As mentioned in Section II, cell voltages must be non-negative so they are shifted to $b_i + V_{\max}$ in the regular scheme and to $v_i^0 + kV_{\max}$ in the spreading scheme. Denote T_{spread} and T_{regular} the damage with the spreading and the regular scheme, respectively. Then,

$$T_{\text{regular}} = a \cdot E\left[(b_i + V_{\max})^2\right] = a(E[b_i^2] + V_{\max}^2)$$
$$T_{\text{spread}} = a \cdot E\left[(v_i^0 + kV_{\max})^2\right] = a\left(\frac{k^2}{N}E[b_i^2] + k^2V_{\max}^2\right),$$

for some constant a.For $k = 1$ (i.e., both schemes have identical programming range), spreading causes less damage than the regular scheme but it lowers the SNR. For $k = \sqrt{N}$ both schemes have the same SNR, but spreading causes more damage. Section IV will elaborate how to choose an optimal k in between.

B. Effect of inter-cell interference

The previous section showed that when the noise is independent from the voltages being programmed in the cells, our spreading scheme does not provide any improvement in terms of SNR unless $k \geq \sqrt{N}$. However, the main source of noise in new memory generations is ICI, which is proportional to the aforementioned voltages.

In most memories, flash cells are organized in an array structure, where all the cells in a wordline are programmed simultaneously and wordlines are programmed in increasing order. The ISPP [6] algorithm used to program wordlines can compensate for the inter-cell interference caused by previously programmed wordlines, but not for the interference of subsequent program operations. Hence, most of the ICI suffered by a specific cell is caused by the direct-above-neighbor. This will be the only ICI component considered in our analysis, but the simulations in Section V will include 3 neighbors.

Assuming $n^{ICI} \gg n^w$, we can write Eq. (4) to be:

$$\hat{b_i} = b_i + \frac{1}{k}\sum_{j=1}^{N} \pm n^{\text{ICI}}, \quad i = 1,2,\ldots,M,$$

where n^{ICI} is proportional to the programmed voltages,

$$n^{\text{ICI}} = \gamma\mathbf{v}^0 = \frac{k\gamma}{N}\sum_{j=1}^{N} \pm b_j,$$

So the estimated symbol can be represented as $\hat{b_i} = b_i + \Delta b_{\text{spread}}$, where

$$\Delta b_{\text{spread}} = \frac{\gamma}{N}\sum_{j=1}^{N^2} \pm b_j.$$

If the distance between the symbols is d, errors happen only when $\Delta b_{\text{spread}} \geq \frac{d}{2}$. The distribution of Δb_{spread} is approximately $\mathcal{N}(0, \gamma^2 E[b_i^2])$ according to the Central Limit Theorem. Then the probability of error for non-extreme symbols is approximately

$$P_e^{\text{spread}} \approx 2\phi\left(\frac{-d}{2\gamma\sqrt{E[b_i^2]}}\right), \tag{6}$$

where $\phi(u) = \int_{-\infty}^{u} \frac{1}{\sqrt{2\pi}} e^{-\frac{y^2}{2}} dy$. Note that in Eq. (6), the scaling parameter k has no effect on ICI.

In the regular scheme, the estimated symbol is:

$$\hat{b_i} = b_i + \gamma b_j,$$

with probability of error for non-extreme symbols

$$P_e^{\text{regular}} = P\left(|b_j| > \frac{d}{2\gamma}\right).$$

The main advantages of our spreading scheme come from the fact that the voltages being programmed in the cells have smaller variance than in the regular scheme. As shown in Fig. 1, spreading leads to a distribution with less variance. As γ increases, the regular scheme introduces much more

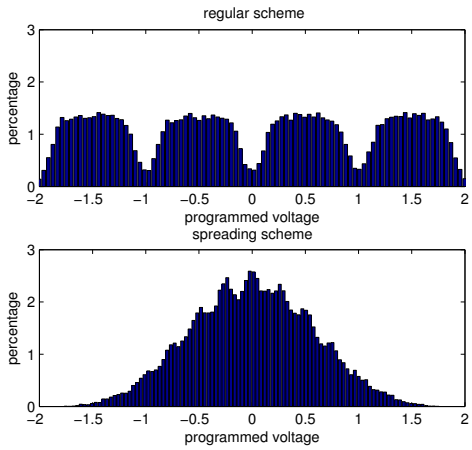

Fig. 1. Distribution of cell voltages for both schemes when M=N=4, k=1, $\sigma = 0.1$, $\gamma = 0.2$. Spreading leads to a distribution with less variance.

probability of error than the spreading scheme. For example, if $\gamma = 0.35$ and $\mathcal{X} = \{\pm 0.5, \pm 1.5\}$ then $d = 1$ and for MLC memories

$$P_{e(MLC)}^{\text{regular}} \simeq 0.5 \qquad P_{e(MLC)}^{\text{spread}} \simeq 0.2 \tag{7}$$

for intermediate (non-extreme) symbols in the absence of Gaussian noise. The lowest and highest symbol would suffer half as much probability of error in both cases.

As γ increases, P_e^{spread} increases slower than P_e^{regular}. So, when γ is large enough, the spreading scheme has a better performance. It is also important to take into account that the grouping of cells into spreading blocks must be done randomly. If the same cells, say $1-4$ were taken as a spreading block in two consecutive wordlines, the ICI would have the form of a scaled codeword, and would therefore not be attenuated by the de-spreading.

C. Effect of impulsive noise

Another important advantage of the spreading approach lies on its increased robustness to impulse noise. Flash memories are currently being used in a wide variety of environments. In most of them they compete with HDD and DRAM but there are some cases in which flash is the only viable option. One of those cases are satellite applications. Hard drives have moving parts, and need a certain air pressure for the head to fly appropriately. DRAM memories are volatile and require frequent refreshing to avoid losing the information. Flash memories, however are perfect for satellite applications. Their lack of moving parts makes them very compact and shock resistant, and they can be powered off for extended periods of time without losing information.

Satellites suffer a significant amount of radiation, constituting one of the leading causes of electrical component failures. A high energy particle impacting on a NAND flash cell usually causes what is known as a stuck-at defect [8]. The cell effectively breaks and will henceforth be read as storing the same voltage value, regardless of what it was meant to be programmed to. In the regular scheme, any bit written to that cell will most likely be lost. The scheme proposed in this paper, on the other hand, spreads each bit across multiple cells, and has a chance to recover the bit even if one of the cells is stuck at a given value.

Broken cells can usually be identified before they are read. The ISPP programming mechanism checks the cell voltages after sending each pulse and, when the controller detects that the cell voltage has not changed after having sent multiple pulses, the cell is marked as broken. If this were known before the programming started, we could just ignore that cell altogether and not store anything in it. Unfortunately, if the broken cell is detected during programming, it is too late to stop the programming of the other cells in the page.

Let P denote the probability of a cell breaking. We assume that the controller knows which cells are broken, and can therefore assign them an arbitrary voltage at read time, independently from the actual state they are in. In order to minimize the resulting noise, broken cells will be read as having a voltage of 0, the average voltage stored by a healthy cell.

Equation (2) shows that the nominal voltage programmed in the i-th cell is $v_i^0 = c_i^T b$, where c_i^T represents the i-th row of the spreading matrix $\mathbf{C}$. If the i-th cell is broken, the controller interprets $v_i = 0$, which is equivalent to replacing i-th column of the de-spreading matrix by zeros when the read operation is performed. The estimated data symbols can then be represented as:

$$\begin{bmatrix} \hat{b_1} \\ \hat{b_2} \\ \vdots \\ \hat{b_N} \end{bmatrix} = \frac{1}{N} \begin{bmatrix} N-1 & \pm 1 & \cdots & \pm 1 \\ \pm 1 & N-1 & \cdots & \pm 1 \\ \vdots & & & \vdots \\ \pm 1 & \pm 1 & \cdots & N-1 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix},$$

where all the noise except the broken cell has been neglected. In other words, the estimated information symbol $\hat{b_i}$ can be represented as:

$$\hat{b_i} = \frac{N-1}{N} b_i + \frac{1}{N} \sum_{j \neq i} \pm b_j \quad i = 1, 2, \ldots, N,$$

where the signs of the error terms depend on the spreading matrix and the signs of the different bits.

In SLC flash memories, an error will occur if the sign of $\hat{b_i}$ is different from the sign of b_i. This can only happen if the signs of the other bits align just right so that the $N-1$ error terms cancel out the correct $\frac{N-1}{N}$ contribution. It happens with probability $\frac{1}{2^{N-1}}$. Moreover, even when the signs align just right, which means $\hat{b_i} = 0$, we still have a 50% chance of guessing the sign correctly. So the probability of error due to broken cells is $\frac{N}{2^N} P(1-P)^{N-1} + O(P^2)$.

However, for the regular scheme, whatever bits were stored in the broken cells are completely lost. The ECC will have to recover them if possible. If a cell is broken, it has a $\frac{1}{2}$ chance of storing the correct value, so the probability of error is $\frac{P}{2}$ for the regular scheme, which is much larger than with the spreading scheme.

In MLC flash memories, however, our scheme does not always offer advantages towards impulsive noise. Since there are more programming voltage levels, the error terms may play a more important role because it may contain some large voltage levels. But space applications generally use SLC memories because they are more reliable than MLC.

IV. Choice of k

Increasing k can improve SNR through noise attenuation, but the range of programmed voltages becomes wider. It was shown in Fig. 1 that the probability of programming a very large or small voltage with the spreading scheme is very low, so it could be helpful to increase k and then crop those extremes. If the gains in terms of noise attenuation obtained by increasing k make up for the cropping noise, the overall SNR will increase.

Instead of increasing k and then cropping the largest voltages, our scheme crops both high and low voltages symmetrically, so as to minimize the cropping noise. Assuming that the desired range of programmed voltages is $[-V_{\max}, V_{\max}]$, the quantization noise introduced by cropping is

$$\mathbf{q} = \min[\mathbf{0}, V_{\max} - \frac{k}{N}\mathbf{Cb}] + \max[\mathbf{0}, -V_{\max} - \frac{k}{N}\mathbf{Cb}].$$

The information symbols read can be represented as:

$$\hat{\mathbf{b}} = \mathbf{b} + \frac{1}{k}\mathbf{C}^T\mathbf{n} + \frac{1}{k}\mathbf{C}^T\mathbf{q},$$

where $\mathbf{q}$ is the quantization noise. In other words, for each estimated information value $\hat{b_i}$:

$$\hat{b_i} = b_i + \frac{1}{k}\sum_{j=1}^{N} \pm(n^{\text{ICI}} + n^w) + \frac{1}{k}\sum_{j=1}^{N} \pm q_j.$$

To minimize the overall distortion introduced by cropping, we want to crop only the largest and smallest voltages in our scheme, $\pm kV_{max}$. These levels are programmed with probability $\frac{2}{L^N}$, where $L = |\mathcal{X}|$ is the number of possible information symbols, hence q can be represented as:

$$q = \begin{cases} \pm(k-1)V_{\max} & \text{with probability } \frac{2}{L^N} \\ 0 & \text{with probability } \frac{L^N-2}{L^N} \end{cases}$$

The Gaussian noise, ICI, and quantization noise are uncorrelated, so the total noise power P_N can be found by a simple sum of the components $P_N = P_w + P_{\text{ICI}} + P_q$, where:

$$\begin{aligned} P_w &= \frac{N}{k^2}\sigma^2 \\ P_{\text{ICI}} &= \gamma^2 E[b_i^2] \\ P_q &= \frac{2N(k-1)^2}{k^2 L^N} V_{\max}^2. \end{aligned} \tag{8}$$

As k increases the write noise decreases but the quantization noise increases. There is a trade-off between quantization noise and write noise. As shown in Fig. 2, the optimal k which minimizes the total noise power and consequently maximizes the SNR is:

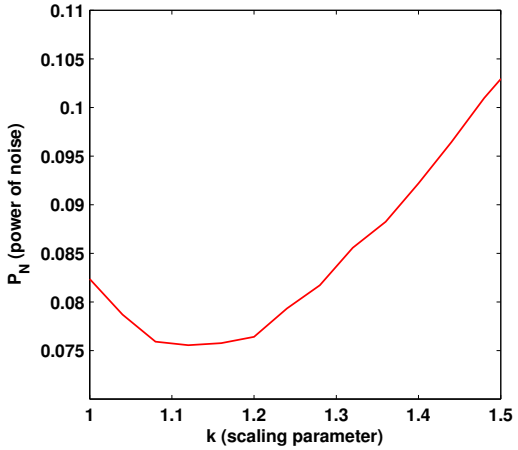

Fig. 2. Quantization noise power as a function of k for a SLC flash memory with M=N=4, $\sigma = 0.1$, and $\gamma = 0.2$

$$k^\star = \arg\min_k \frac{2N(k-1)^2}{k^2 L^N} V_{max}^2 + \frac{N}{k^2}\sigma^2.$$

For some memories, it may be hard to control the small voltage increments between the levels in the spreading scheme, specially if k is small. The over-programming could introduce Gaussian noise, but the total power of this noise would still be lower than that in the regular scheme, since the programming pulses would also be smaller.

V. Simulation results

This section compares the proposed data representation scheme with the regular one through simulations. It evaluates both of them in terms of BER and damage caused to the memory. We simulate 10 memory blocks with 128 pages per block and 8096 cells in each page. Each cell is assumed to suffer ICI from 3 neighbors in the next wordline, with coefficient $\gamma = 0.3\alpha$ for the direct neighbor (the one in the same bitline) and $\gamma = 0.25\alpha$ for the other two. The write noise is assumed to be Gaussian with zero mean and $\sigma = 0.1$ and cells are assumed to break with probability $P_{\text{broken}} = 0.001$.

First, we study how BER increases with ICI when $M = N = 4$, $k = 1.4$, and the voltages v^0 are cropped to be in the range $[-0.5, 0.5]$ for SLC memory and $[-1.5, 1.5]$ for MLC memory. Figure 3 shows the results for an SLC memory (i.e. $\mathbf{b} \in \{-0.5, 0.5\}^4$) and Figure 4 for an MLC memory (i.e. $\mathbf{b} \in \{-1.5, -0.5, 0.5, 1.5\}^4$). When ICI is small the regular scheme performs better in both SLC and MLC cells, but when ICI increases, the spreading scheme provides lower BER.

Then, we study the trade-off curves between damage and BER in both schemes. Gaussian noise, ICI, and breaking probability are kept constant at $\sigma = 0.1$, $\alpha = 1$, and $P_{\text{broken}} = 0.001$ while the programmed voltage levels are scaled. In the regular scheme, the scaling is done by changing

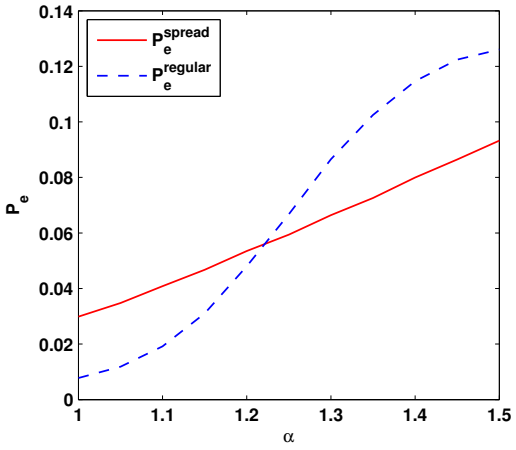

Fig. 3. Evolution of the probability of error as ICI increases for an SLC memory, when M=N=4, k=1.4, $\sigma = 0.1$, $\gamma = (0.25\alpha, 0.25\alpha, 0.3\alpha)$, and $P_{\text{broken}} = 0.001$.

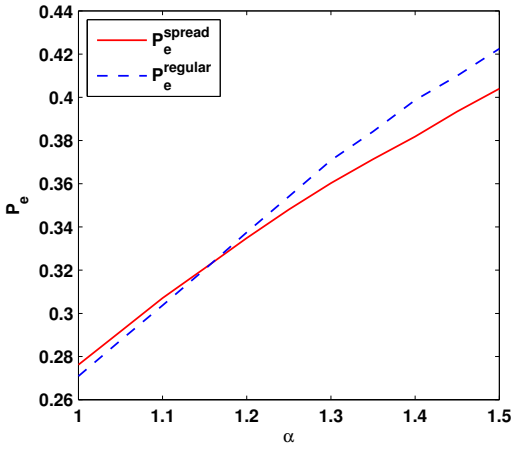

Fig. 4. Evolution of the probability of error as ICI increases for an MLC memory, when M=N=4, k=1.4, $\sigma = 0.1$, $\gamma = (0.25\alpha, 0.25\alpha, 0.3\alpha)$, and $P_{\text{broken}} = 0.001$.

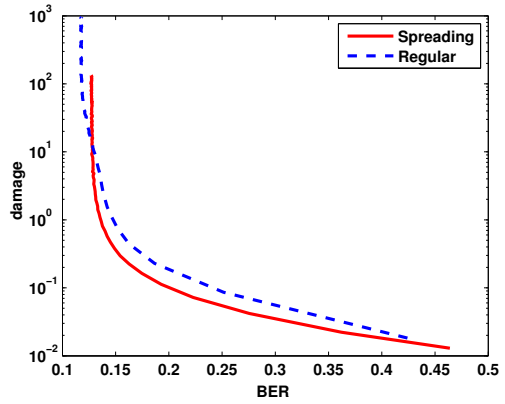

Fig. 5. Trade-off curves between BER and cell damage for an MLC memory when $\sigma = 0.1$, Pbroken=0.001, $\gamma = (0.25, 0.25, 0.3)$, and M=N=4.

the distance between the levels and in the spreading scheme, it is done by means of parameter k. In both cases, the damage is represented by the expected value of the squared programmed voltage. As the distance between levels increases, Gaussian noise becomes comparatively weaker and the probability of error decreases but the memory suffers more damage due to the larger programmed voltages. Figure 5 illustrates the advantages of the spreading scheme: for a fixed probability of error, the spreading scheme causes much less damage and for a fixed damage, the spreading scheme offers a lower probability of error. Only for extreme separations between the levels, when the expected voltage squared is higher than 10, does the regular scheme offer an advantage. Spreading is generally better in terms of both damage and BER.

VI. Conclusion

This paper proposed a novel data representation scheme where orthogonal codes are used to store the information in a NAND flash memory, so that each symbol is spread out over multiple cells. By increasing the number of possible voltage levels in each cell, disregarding the fact that these levels could overlap, the proposed scheme can provide significant gains in terms of robustness towards inter-cell interference and impulsive noise. Additionally, higher levels are used less frequently than in the usual scheme, reducing the damage suffered by the cells and thereby extending the endurance of the memory. The paper provides analytical expressions for the SNR and BER of this spreading scheme under Gaussian noise, ICI, and impulsive noise. Its performance is then studied through simulations.

In future work, we will study the best way to combine this spreading scheme with error correcting codes and will try to find ways to overcome the penalty in terms of read speed that the additional number of levels poses.

References

[1] B. Shin, C. Seol, J.-S. Chung, and J. J. Kong, "Error control coding and signal processing for flash memories," in *Circuits and Systems (ISCAS), 2012 IEEE International Symposium on*. IEEE, 2012, pp. 409–412.

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

[3] H.-W. Tseng, L. Grupp, and S. Swanson, "Understanding the impact of power loss on flash memory," in *Proceedings of the 48th Design Automation Conference*. ACM, 2011, pp. 35–40.

[4] W. Wang, T. Xie, and D. Zhou, "Understanding the impact of threshold voltage on mlc flash memory performance and reliability," in *Proceedings of the 28th ACM international conference on Supercomputing*. ACM, 2014, pp. 201–210.

[5] B. Peleato and R. Agarwal, "Maximizing MLC NAND lifetime and reliability in the presence of write noise," in *Proc. of IEEE International Conf. Comm.*, 2012.

[6] K.-D. Suh, B.-H. Suh, Y.-H. Lim, J.-K. Kim, Y.-J. Choi, Y.-N. Koh, S.-S. Lee, S.-C. Kwon, B.-S. Choi, J.-S. Yum *et al.*, "A 3.3 v 32 mb nand flash memory with incremental step pulse programming scheme," *Solid-State Circuits, IEEE Journal of*, vol. 30, no. 11, pp. 1149–1156, 1995.

[7] Y. Cai, E. F. Haratsch, O. Mutlu, and K. Mai, "Threshold voltage distribution in mlc nand flash memory: Characterization, analysis, and modeling," in *Proceedings of the Conference on Design, Automation and Test in Europe*. EDA Consortium, 2013, pp. 1285–1290.

[8] Y. Kim and B. Kumar, "Coding for memory with stuck-at defects," *arXiv preprint arXiv:1304.4821*, 2013.