Optimized Cell Programming for Flash Memories with Quantizers

Minghai Qin*, Eitan Yaakobi*†, Paul H. Siegel*
*University of California, San Diego
Electrical and Computer Engineering Department
†California Institute of Technology
Electrical Engineering Department
Emails: {mqin, eyaakobi, psiegel}@ucsd.edu

***Abstract*—Multi-level flash memory cells represent data by the amount of charge stored in them. Certain voltages are applied to the flash memory cells to inject charges when programming and the cell level can be only increased during the programming process as a result of the high cost of block erasures. To achieve a high speed during writing, parallel programming is used, whereby a common voltage is applied to a group of cells to inject charges simultaneously. The voltage sharing simplifies the circuitry and increases the programming speed, but it also affects the precision of charge injection and limits the storage capacity of flash memory cells. Another factor that limits the precision of cell programming is the thermal electronics noise induced in charge injection.**

In this paper, we focus on noiseless parallel programming of multiple cells and noisy programming of a single cell. We propose a new criterion to evaluate the performance of the cell programming which is more suitable for flash memories in practice and then we optimize the parallel programming strategy accordingly. We then proceed to noisy programming and consider the two scenarios where feedback on cell levels is either available during programming or not. We study the optimization problem under both circumstances and present algorithms to achieve the optimal performance.

I. Introduction

Flash memories are a widely-used technology for non-volatile data storage. The basic memory units in flash memories are floating-gate cells, which use charge (i.e., electrons) stored in them to represent data, and the amount of charge stored in a cell determines its *level*. The hot-electron injection mechanism or Fowler-Nordheim tunneling mechanism [2] is used to increase and decrease a cell level by injecting charge into it or by removing charge from it, respectively. The cells in flash memories are organized as blocks, each of which contains about 10^6 cells. One of the most prominent features of programming flash memory cells is its asymmetry; that is, increasing a cell level, i.e., injecting charge into a cell, is easy to accomplish by applying a certain voltage to the cell, while decreasing a cell level, i.e., removing charge from a cell, is expensive in the sense that the block containing the cell must first be erased, i.e., all charge in the cells within the block is totally removed, before reprogramming to their target levels. The erase operation, called a *block erasure*, is not only time consuming, but also degrades the performance and reduces the longevity of flash memories [2].

In order to minimize the number of block erasures, programming flash memories is accomplished very carefully using multiple rounds of charge injection to avoid "overshooting" the desired cell level. Therefore, a flash memory can be modeled as a Write Asymmetric Memory (WAM), for which capacity analysis and coding strategies are discussed in [1], [5], [11].

Parallel programming is a crucial tool to increase the write speed when programming flash memory cells. Two important properties of parallel programming are the use of shared program voltages and the need to account for variation in charge injection [12]. Instead of applying distinct program voltages to different cells, parallel programming applies a common program voltage to many cells simultaneously. Consequently, the complexity of hardware realization is substantially reduced and the write speed is therefore increased. Parallel programming must also account for the fact that cells have different hardness with respect to charge injection [3], [9]. When applying the same program voltage to cells, the amount of charge trapped in different cells may vary. Those cells that have a large amount of trapped charge are called *easy-to-program* cells and those with little trapped charge are called *hard-to-program* cells. Understanding this intrinsic property of cells will allow the programming of cells according to their hardness of charge injection. One widely-used programming method is the *Incremental Step Pulse Programming* (ISPP) scheme [3], [9], which allows easy-to-program cells to be programmed with a lower program voltage and hard-to-program cells to be programmed with a higher program voltage.

In [6], [7], the optimized programming for a single flash memory cell was studied. A programming strategy to optimize the expected precision with respect to two cost functions was proposed, where one of the cost functions is the ℓ_p metric and the other is related to rank modulation [8]. It was assumed that the programming noise follows a uniform distribution and the level increment is chosen adaptively according to the current cell level to minimize the cost function.

In [12], algorithms for parallel programming were studied. The underlying model incorporated shared program voltages and variations in cell hardness, as well as a cost function based upon the ℓ_p metric. The programming problem was formulated as a special case of the Subspace/Subset Selection Problem (SSP) [4] and the Sparse Approximate Solution Problem (SAS) [10], both of which are NP-hard. Then an algorithm with polynomial time complexity was proposed to search for the optimal programming voltages.

We note that flash memories use multiple discrete levels to represent data in real applications [2]. Hence, if the actual cell level is within a certain distance from the target level, it will be quantized to the correct target level even though there is a gap between them. Read errors can be mitigated by use of error correction codes. If the error correction capability is

978-1-4673-2579-0/12/$31.00 ©2012 IEEE

e, then any read error will be totally eliminated as long as the number of read errors is less than e. This motivates us to consider another cost function, which is the number of cells that are not quantized correctly to their target levels.

Assume that $\Theta = (\theta_1, \ldots, \theta_n)$ is the vector of target cell levels and $\boldsymbol{\ell}_t = (\ell_{1,t}, \ldots, \ell_{n,t})$ is a vector of random variables which represent the level of every cell after t programming rounds. Note that in general the value of $\ell_{i,t}$, for $1 \leqslant i \leqslant n$, depends on the applied voltages, the hardness of the cell, and the programming noise. We will evaluate the performance of any programming method by some cost function $\mathcal{C}(\Theta, \boldsymbol{\ell}_t)$ between the target cell levels Θ and the actual cell levels $\boldsymbol{\ell}_t$. Then, the programming problem is to find an algorithm which minimizes the expected value of $\mathcal{C}(\Theta, \boldsymbol{\ell}_t)$. In [12], the ℓ_p metric was considered as the cost function $\mathcal{C}_p(\Theta, \boldsymbol{\ell}_t) = \left(\sum_{i=1}^{n} \left|\theta_i - \ell_{i,t}\right|^p\right)^{\frac{1}{p}}$. Motivated by the nature of quantization of the cell levels, we study in this paper the cost function

$$\mathcal{C}_{\boldsymbol{\Delta}}(\Theta, \boldsymbol{\ell}_t) = \left|\left\{i \in [n] : \left|\theta_i - \ell_{i,t}\right| > \Delta_i\right\}\right|,$$

where Δ_i is the *quantization distance* for the i-th cell. We solve this problem for the special case where the hardness of each cell is known and there is no programming noise. We also study the problem in the presence of noise for a single cell with and without feedback.

The rest of the paper is organized as follows. In Section II, we propose a new cost function and define the parallel programming problem when flash memories quantize the amount of charge to discrete levels. In Section III, we present a polynomial-time algorithm to optimize the noiseless parallel programming with deterministic parameters defined in Section II. In Section IV, single cell programming with noise is studied where there is no feedback information on the cell level. In Section V, noisy cell programming is studied where we can adaptively apply voltages according to the feedback of the current cell level. Due to the lack of space, some proofs will be omitted.

II. Preliminaries

Let $c_1, c_2, \ldots, c_n$ be n flash memory cells, with erased level denoted by 0. Their levels can be increased by injecting electrons, but cannot be decreased. We denote by $[n]$ the set of positive integers less than or equal to n, i.e., $[n] = \{1, 2, \ldots, n\}$. When applying a voltage V to a memory cell c_i, where $i \in [n]$, we assume that the increase of the level of cell c_i is linear with V, that is, the level of c_i will increase by

$$\alpha_i V + \epsilon,$$

where α_i and ϵ are random variables. We call the α_i's, where $\alpha_i > 0, i \in [n]$, the *hardness of charge injection* for cell c_i, and ϵ is the programming noise. (Note that the distribution of ϵ may vary among different cells and different writes.)

We denote by $\theta_i \geqslant 0$, $i \in [n]$, the *target level* of c_i. The programming process consists of t rounds of charge injection achieved by applying a specified voltage to all of the cells. The goal is to program the cell levels to be as close as possible to the target levels. We define the parallel programming problem in detail as follows.

Let $\Theta = (\theta_1, \ldots, \theta_n)$ be the target cell levels and $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$ be the hardness of charge injection. Let $\boldsymbol{V} = (V_1, V_2, \ldots, V_t)^T$ be the voltages applied on the t rounds of programming. Let $b_{i,j} \in \{0, 1\}$ for $i \in [n]$ and $j \in [t]$ indicate whether c_i is programmed on the j-th round; i.e., $b_{i,j} = 1$ if voltage V_j is applied to cell c_i, and $b_{i,j} = 0$, otherwise. Let $\ell_{i,t}$ for $i \in [n]$ be a random variable, representing the level of c_i after t rounds of programming. Then

$$\ell_{i,t} = \sum_{j=1}^{t} (\alpha_i V_j + \epsilon_{i,j})\, b_{i,j},$$

where $\epsilon_{i,j}$, for $i \in [n]$ and $j \in [t]$, is the programming noise of the i-th cell on the j-th programming round. We define $\boldsymbol{\ell}_t = (\ell_{1,t}, \ldots, \ell_{n,t})$ to be the cell-state vector after t rounds of programming and we define the matrix

$$\mathbf{B} = \begin{bmatrix} b_{1,1} & b_{2,1} & \cdots & b_{n,1} \\ b_{1,2} & b_{2,2} & \cdots & b_{n,2} \\ \vdots & \vdots & \ddots & \vdots \\ b_{1,t} & b_{2,t} & \cdots & b_{n,t} \end{bmatrix} \in \{0,1\}^{t \times n}$$

to be the indicator matrix of the programmed cell on each round of programming.

We evaluate the performance of the programming by some cost function $\mathcal{C}(\Theta, \boldsymbol{\ell}_t)$. The programming problem is to minimize the expected value of $\mathcal{C}(\Theta, \boldsymbol{\ell}_t)$ over $\boldsymbol{V}$ and $\mathbf{B}$. That is, given the information of Θ, $\boldsymbol{\alpha}$ and $\{\epsilon_{i,j}\}_{n \times t}$, we seek to solve

$$\text{minimize } \mathbb{E}\left[\mathcal{C}(\Theta, \boldsymbol{\ell}_t)\right], \tag{P1}$$

with $\boldsymbol{V} \in \mathbb{R}_+^t$ and $\mathbf{B} \in \{0,1\}^{t \times n}$, where $\mathbb{E}[X]$ is the expected value of the random variable X.

Remark 1. We use $\mathbb{R}_+$ to denote the set of all non-negative real numbers, i.e., $\mathbb{R}_+ = \{x \in \mathbb{R} : x \geqslant 0\}$.

In [12], the ℓ_p metric is considered as the cost function, i.e.

$$\mathcal{C}_p(\Theta, \boldsymbol{\ell}_t) = \left(\sum_{i=1}^{n} \left|\theta_i - \ell_{i,t}\right|^p\right)^{\frac{1}{p}},$$

and the optimal solution for (P1) was derived for known $\boldsymbol{\alpha}$ in the absence of noise. However, in real applications, flash memories use multiple discrete levels to represent data and if the cell level $\ell_{i,t}$ is within a certain distance from the target level θ_i, it will be quantized to the correct target level even though there is a gap between $\ell_{i,t}$ and θ_i. This motivates us to consider the number of cells that are not correctly quantized to their target levels as the cost function. To be more precise, letting $\boldsymbol{\Delta} = (\Delta_1, \ldots, \Delta_n)$, we define

$$\mathcal{C}_{\boldsymbol{\Delta}}(\Theta, \boldsymbol{\ell}_t) = \left|\left\{i \in [n] : \left|\theta_i - \ell_{i,t}\right| > \Delta_i\right\}\right|$$

to be the cost function, where Δ_i is the *quantization distance* for c_i. Therefore, the cell programming problem is to solve

$$\text{minimize } \mathbb{E}\left[\left|\left\{i \in [n] : \left|\theta_i - \ell_{i,t}\right| > \Delta_i\right\}\right|\right], \tag{P2}$$

with $\boldsymbol{V} \in \mathbb{R}_+^t$ and $\mathbf{B} \in \{0,1\}^{t \times n}$.

III. Noiseless Parallel Programming

In this section, we assume that the cell hardness parameters $(\alpha_1, \ldots, \alpha_n)$ are known and deterministic, and there is no programming noise, i.e., $\epsilon_{i,j} = 0, \forall i \in [n], j \in [t]$. In this scenario, $\ell_{i,t}$ is deterministic so that we can omit the expectation in (P2) and $\ell_{i,t} = \alpha_i \sum_{j=1}^{t} V_j b_{i,j}$. Let $n, t, \Delta_1, \ldots, \Delta_n$ and

$\theta_1, \ldots, \theta_n$ denote the block length, number of programming rounds, quantization distances and target levels, respectively. Our goal is to find an optimal solution to (P2).

Lemma 1. *The solution to Problem* (P2) *is equivalent to the solution of the following:*

$$\text{maximize } f(\boldsymbol{V}, \mathbf{B}), \tag{P3}$$

with $\boldsymbol{V} = (V_1, \ldots, V_t)^T \in \mathbb{R}_+^t$, $\boldsymbol{b}_i = (b_{i,1}, \ldots, b_{i,t})^T \in \{0,1\}^t$ *and* $\mathbf{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$, *where* $u_i = \frac{\theta_i - \Delta_i}{\alpha_i}$, $v_i = \frac{\theta_i + \Delta_i}{\alpha_i}$, $i \in [n]$ *and* $f(\boldsymbol{V}, \mathbf{B}) = \left| \left\{ i \in [n] : u_i \leqslant \boldsymbol{b}_i^T \cdot \boldsymbol{V} \leqslant v_i \right\} \right|$.

Proof: The following chain of equations is easily established.

$$\begin{aligned}
&\min_{\boldsymbol{V},\mathbf{B}} \left| \left\{ i \in [n] : \left| \theta_i - \ell_{i,t} \right| > \Delta_i \right\} \right| \\
=&n - \max_{\boldsymbol{V},\mathbf{B}} \left| \left\{ i \in [n] : \left| \theta_i - \ell_{i,t} \right| \leqslant \Delta_i \right\} \right| \\
=&n - \max_{\boldsymbol{V},\mathbf{B}} \left| \left\{ i \in [n] : \left| \frac{\theta_i}{\alpha_i} - \frac{\ell_{i,t}}{\alpha_i} \right| \leqslant \frac{\Delta_i}{\alpha_i} \right\} \right| \\
=&n - \max_{\boldsymbol{V},\mathbf{B}} \left| \left\{ i \in [n] : \left| \frac{\theta_i}{\alpha_i} - \boldsymbol{b}_i^T \cdot \boldsymbol{V} \right| \leqslant \frac{\Delta_i}{\alpha_i} \right\} \right| \\
=&n - \max_{\boldsymbol{V},\mathbf{B}} \left| \left\{ i \in [n] : \frac{\theta_i}{\alpha_i} - \frac{\Delta_i}{\alpha_i} \leqslant \boldsymbol{b}_i^T \cdot \boldsymbol{V} \leqslant \frac{\theta_i}{\alpha_i} + \frac{\Delta_i}{\alpha_i} \right\} \right| \\
=&n - \max_{\boldsymbol{V},\mathbf{B}} \left| \left\{ i \in [n] : u_i \leqslant \boldsymbol{b}_i^T \cdot \boldsymbol{V} \leqslant v_i \right\} \right|,
\end{aligned}$$

where $\boldsymbol{V} \in \mathbb{R}_+^t, \mathbf{B} \in \{0,1\}^{t \times n}, u_i = \frac{\theta_i - \Delta_i}{\alpha_i}$ and $v_i = \frac{\theta_i + \Delta_i}{\alpha_i}$. This establishes the chain. ∎

Since u_i and v_i, $i \in [n]$ are the boundaries of the correct quantization interval for c_i, we call them the *upper threshold point* and the *lower threshold point* for c_i and we call the interval $[u_i, v_i]$ the *quantization interval* for c_i. Any pair $(\boldsymbol{V}, \mathbf{B})$ that achieves the maximum for (P3) is called an *optimal solution pair*, and $\boldsymbol{V}$ is called *optimal* or an *optimal solution* if there exists $\mathbf{B}$ such that $(\boldsymbol{V}, \mathbf{B})$ is an optimal solution pair.

Definition 1. *Suppose* u_i *and* v_i, $i \in [n]$, *are defined as in Lemma* 1. *Let* $\mathcal{T}_u$ *be the set of upper threshold points and* $\mathcal{T}_v$ *be the set of lower threshold points, i.e.,* $\mathcal{T}_u = \bigcup_{i \in [n]} \{u_i\}$ *and* $\mathcal{T}_v = \bigcup_{i \in [n]} \{v_i\}$. *Let* $\mathcal{T} = \mathcal{T}_u \cup \mathcal{T}_v$ *be the set of all upper and lower threshold points.*

Remark 2. We assume that $|\mathcal{T}| > t$ since otherwise we can easily achieve $\mathcal{C}_{\boldsymbol{\Delta}}(\Theta, \boldsymbol{\ell}_t) = 0$ by setting $\{V_1, \ldots, V_t\} = \mathcal{T}$.

Definition 2. *Suppose* $\boldsymbol{V} = (V_1, \ldots, V_t)^T \in \mathbb{R}_+^t$. *We define* $S_{\boldsymbol{V}}$ *to be*

$$S_{\boldsymbol{V}} = \bigcup_{\boldsymbol{b} \in \{0,1\}^t} \{\boldsymbol{b}^T \cdot \boldsymbol{V}\}.$$

and call it the **attainable set** *of* $\boldsymbol{V}$. *That is,* $S_{\boldsymbol{V}}$ *is the set of voltage values that can be injected by applying* $\boldsymbol{V}$.

Remark 3. For each $i \in [n]$, if there exists $z \in S_{\boldsymbol{V}}$ such that $u_i \leqslant z \leqslant v_i$, then there exists $\boldsymbol{b} \in \{0,1\}^t$ such that $u_i \leqslant \boldsymbol{b}^T \cdot \boldsymbol{V} \leqslant v_i$, and thus c_i can be quantized to the correct target level.

For a fixed $\boldsymbol{V}$, optimizing the cost function over $\mathbf{B}$ is easy to accomplish by checking whether there exists $z \in S_{\boldsymbol{V}}$ such that $u_i \leqslant z \leqslant v_i$ for each i. Intuitively, we can enumerate every possible $\boldsymbol{V}$ and calculate the cost function to search for an optimal solution. However, since $\boldsymbol{V}$ can be any point in $\mathbb{R}_+^t$, there is an uncountably infinite number of choices of $\boldsymbol{V}$ to consider. Lemma 2 states that we can limit the number under consideration to be polynomial in n, and guarantee that an optimal solution can be found. Although this lemma plays the most important role in deriving the algorithm, the proof is too long to present due to space limitations.

Lemma 2. *There exists an invertible matrix* $\mathbf{A} \in \{0,1\}^{t \times t}$, *such that*

$$\mathbf{A} \cdot \boldsymbol{V} = \boldsymbol{p},$$

where $\boldsymbol{V}$ *is an optimal solution for* (P3) *and* $\boldsymbol{p} \in \mathcal{T}^t$.

Remark 4. In Lemma 2 and Algorithm 1 below, the matrix $\mathbf{A}$ has to be invertible over $\mathbb{R}$ instead of $GF(2)$. Therefore, enumerating only the invertible matrices over $GF(2)$ is not sufficient to find an optimal solution.

Next we give an algorithm to search for an optimal solution to (P3), which, as we have shown, is also an optimal solution to (P2). Let $\{p_1, \ldots, p_M\}$ be an arbitrary ordering of the points in $\mathcal{T}$, where $M = |\mathcal{T}|$ is the number of different threshold point values and p_i can be the value of either an upper or a lower threshold point, for $i \in [M]$. Since $\boldsymbol{p}$ is of length t, there are $N = M^t$ choices of $\boldsymbol{p}$ (the entries can be repeated). Let $\{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_N\}$ be an arbitrary ordering of the choices. A matrix $\widetilde{\mathbf{A}} \in \{0,1\}^{t \times t}$ is formed such that no two rows are the same. Thus, the number of different $\widetilde{\mathbf{A}}$'s is $Q = \prod_{k=0}^{t-1} (2^t - k)$. Let $\{\widetilde{\mathbf{A}}_1, \ldots, \widetilde{\mathbf{A}}_Q\}$ be an arbitrary ordering of all possible $\widetilde{\mathbf{A}}$'s. Algorithm 1 will iterate over all choices of $\boldsymbol{p}$ and those $\widetilde{\mathbf{A}}$'s that are invertible.

Algorithm 1 PARALLEL PROGRAMMING

Let $f^* = 0$;
Let $\boldsymbol{V} = \boldsymbol{V}^* = (0, \ldots, 0)$, $\boldsymbol{V}, \boldsymbol{V}^*$ *both have length* t;
Let $\mathbf{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n) \in \{0,1\}^{t \times n}$, $\boldsymbol{b}_i = \mathbf{0}, \forall i \in [n]$;
Let $\mathbf{B}^* \in \{0,1\}^{t \times n}$, $b^*_{i,j} = 0, \forall i \in [t], j \in [n]$;
For $i = 1, 2, \ldots, N$ {
  For $j = 1, 2, \ldots, Q$ {
    If $\widetilde{\mathbf{A}}_j$ *is invertible and* $\widetilde{\mathbf{A}}_j^{-1} \cdot \boldsymbol{p}_i \in \mathbb{R}_+^t$ {
      Let $\boldsymbol{V} = \widetilde{\mathbf{A}}_j^{-1} \cdot \boldsymbol{p}_i$;
      Let $f = 0$;
      For $k = 1, 2, \ldots, n$ {
        If $\exists z \in S_{\boldsymbol{V}}$, *such that* $u_k \leqslant z \leqslant v_k$ {
          Find $\boldsymbol{b} \in \{0,1\}^t$, *such that* $\boldsymbol{b}^T \cdot \boldsymbol{V} = z$;
          Let $\boldsymbol{b}_k = \boldsymbol{b}$;
          $f = f + 1$;
        }}
      If $f > f^*$ {
        $f^* = f$, $\boldsymbol{V}^* = \boldsymbol{V}$, $\mathbf{B}^* = \mathbf{B}$; }
}}}
Output the optimal solution pair $(\boldsymbol{V}^*, \mathbf{B}^*)$ *with maximized* $f(\boldsymbol{V}^*, \mathbf{B}^*) = f^*$.

Remark 5. Since any matrix with two identical rows is not invertible, we only enumerated $\widetilde{\mathbf{A}}$ matrices with distinct rows and then checked their invertibility. Therefore, the number of $\mathbf{A} \in \{0,1\}^{t \times t}$ that we considered is $\prod_{k=0}^{t-1} (2^t - k)$. Furthermore, note that the complexity of the algorithm could be slightly reduced if we enumerated only the set of invertible matrices $\mathbf{A}$ over $\mathbb{R}$.

Theorem 1. *Algorithm* 1 *finds the optimal solution pair* $(\boldsymbol{V}^*, \mathbf{B}^*)$ *and computes the optimal value* $f(\boldsymbol{V}^*, \mathbf{B}^*)$ *for* (P3). *The time complexity of the algorithm is* $O(n^{t+1})$.

Proof: According to Lemma 2, there exists an optimal solution $(\boldsymbol{V}, \mathbf{B})$, an invertible matrix $\mathbf{A} \in \{0,1\}^{t\times t}$, and a threshold-point vector $\boldsymbol{p} \in \mathcal{T}^t$, such that

$$\mathbf{A} \cdot \boldsymbol{V} = \boldsymbol{p}.$$

In Algorithm 1, all possible $\mathbf{A}$'s and $\boldsymbol{p}$'s, have been exhaustively iterated and there is at least one optimal $\boldsymbol{V}$ among all the $\boldsymbol{V}$'s derived from $\mathbf{A}$'s and $\boldsymbol{p}$'s. The algorithm outputs the best $\boldsymbol{V}$ among them. This proves that this algorithm will find an optimal solution to (P3).

The number of iterations of the algorithm is of order NQt^3n2^t, where $N = M^t \leqslant (2n)^t$ and $Q = \prod_{k=0}^{t-1}(2^t - k)$. Therefore, the complexity is $O(n^{t+1})$. ∎

IV. Single Cell Noisy Programming without Feedback

In this section, programming noise is assumed to exist. There is a single cell with injection hardness α, and the number of programming rounds is t. The programming noises $\epsilon_1, \ldots, \epsilon_t$ are assumed to be independently distributed Gaussian random variables with zero mean and variance $\sigma^2_{\epsilon_j}, j \in [t]$, respectively.

Remark 6. Note that according to this model, after every programming round the level of each cell can decrease even though in real applications, the cell level of flash memories can only increase. We choose to study this model while assuming that the variance $\sigma_{\epsilon_j}, j \in [t]$ is much smaller than αV_j, i.e., $\mathbf{P}(\alpha V_j + \epsilon_j < 0)$ is very small, such that the probability of decreasing the cell levels is negligible. This model is a reasonable approximation to a physical cell and it can be studied analytically, as will be seen in this section.

Another reasonable assumption we make is $\sigma_{\epsilon_j} = \sigma V_j, j \in [t]$, where σ is a fixed number; that is, the standard deviation of the programming noise is proportional to the programming voltage. This makes sense since large voltage applied to the cell results in large power of the programming noise. During programming, no feedback information is available, meaning that the actual amount of charge trapped in the cell after each round of programming is not known. The goal is to maximize the probability that after t rounds of programming the final level is in $[\theta - \Delta, \theta + \Delta]$, i.e.,

$$\text{maximize } \mathbf{P}\Big(\theta - \Delta \leqslant \sum_{j=1}^{t}(\alpha V_j + \epsilon_j) \leqslant \theta + \Delta\Big), \qquad \text{(P4)}$$

with $\boldsymbol{V} \in \mathbb{R}_+^t$.

In the rest of the section, (P4) is recast as an optimization problem and the vector of voltages $\boldsymbol{V}$ is written as $\boldsymbol{x}$ in accordance with convention.

Lemma 3. *Assume that* $\sigma_{\epsilon_j} = \sigma V_j$, $j \in [t]$, *where* σ *is a fixed number and feedback information is not available, the cell programming problem* (P4) *is equivalent to*

$$\text{maximize } g(\boldsymbol{x}), \qquad \text{(P5)}$$

with $\boldsymbol{x} \in \mathbb{R}_+^t$, *where*

$$g(\boldsymbol{x}) = \frac{1}{\sqrt{2\pi}} \int_{c(\boldsymbol{x})-\delta(\boldsymbol{x})}^{c(\boldsymbol{x})+\delta(\boldsymbol{x})} e^{-u^2/2}\mathrm{d}u,$$

$c(\boldsymbol{x}) = \frac{\theta - \alpha\sum_{j=1}^t x_j}{\sigma\sqrt{\sum_{j=1}^t x_j^2}}$, *and* $\delta(\boldsymbol{x}) = \frac{\Delta}{\sigma\sqrt{\sum_{j=1}^t x_j^2}}$.

Let $p(y) = \frac{1}{\sqrt{2\pi}} e^{-y^2/2}$ be the $\mathcal{N}(0,1)$ Gaussian probability density function. Then $g(\boldsymbol{x})$ can be interpreted as the area between the curves $p(y)$ and $y = 0$ on the interval determined by $\boldsymbol{x}$, where the interval is centered at $\frac{\theta - \alpha\sum_{j=1}^t x_j}{\sigma\sqrt{\sum_{j=1}^t x_j^2}}$, with radius $\frac{\Delta}{\sigma^2\sqrt{\sum_{j=1}^t x_j^2}}$. For simplicity, $\sum_{j=1}^t(\cdot)$ is written as $\sum(\cdot)$ when the context makes the meaning clear.

The following lemma will be used to determine the optimal solution to (P5) in Theorem 2.

Lemma 4. *If* $\boldsymbol{x}^*$ *is the optimal solution to* (P5), *then* $\forall i, j \in [t], x_i^* = x$, *for some constant* $x \in R^+$.

Theorem 2. *The optimal solution* $\boldsymbol{x}$ *to* (P5) *satisfies the following:* $x_j = x, \forall j \in [t]$ *where* x *is the positive root of the equation*

$$\left(2\ln\frac{b}{a}\right)x^2 + 2(b-a)cx + (a^2 - b^2) = 0,$$

where $a = \frac{-\Delta+\theta}{\sigma\sqrt{t}}, b = \frac{\Delta+\theta}{\sigma\sqrt{t}}, c = \frac{\alpha\sqrt{t}}{\sigma}$.

Proof: According to Lemma 3,

$$\begin{aligned}
g(\boldsymbol{x}) &= \frac{1}{\sqrt{2\pi}} \int_{c(\boldsymbol{x})-\delta(\boldsymbol{x})}^{c(\boldsymbol{x})+\delta(\boldsymbol{x})} e^{-u^2/2}\mathrm{d}u \\
&= \frac{1}{\sqrt{2\pi}} \int_{\frac{-\Delta+\theta-\alpha\sum x_j}{\sigma\sqrt{\sum x_j^2}}}^{\frac{\Delta+\theta-\alpha\sum x_j}{\sigma\sqrt{\sum x_j^2}}} e^{-u^2/2}\mathrm{d}u \\
&\leqslant \frac{1}{\sqrt{2\pi}} \int_{\frac{-\Delta+\theta-\alpha tx}{\sigma\sqrt{tx^2}}}^{\frac{\Delta+\theta-\alpha tx}{\sigma\sqrt{tx^2}}} e^{-u^2/2}\mathrm{d}u \\
&= \frac{1}{\sqrt{2\pi}} \int_{\frac{a-cx}{x}}^{\frac{b-cx}{x}} e^{-\frac{u^2}{2}}\mathrm{d}u \\
&:= \frac{1}{\sqrt{2\pi}} h(x),
\end{aligned}$$

where $a = \frac{-\Delta+\theta}{\sigma\sqrt{t}}, b = \frac{\Delta+\theta}{\sigma\sqrt{t}}, c = \frac{\alpha\sqrt{t}}{\sigma}$. The inequality follows from Lemma 4 and it is satisfied with equality if $x_j = x, \forall j \in [t]$. Differentiating $h(x)$ with respect to x, we have

$$\begin{aligned}
\frac{\mathrm{d}h}{\mathrm{d}x} &= e^{-\frac{1}{2}\left(\frac{b-cx}{x}\right)^2} \cdot \frac{-cx - (b-cx)}{x^2} \\
&\quad - e^{-\frac{1}{2}\left(\frac{a-cx}{x}\right)^2} \cdot \frac{-cx - (a-cx)}{x^2} = 0, \\
&\Leftrightarrow \left(2\ln\frac{b}{a}\right)x^2 + 2(b-a)cx + (a^2 - b^2) = 0.
\end{aligned}$$

It can be seen that there is only one extremal point for $f(x)$ when $x \geqslant 0$ and that $f(x) \geqslant 0, \forall x \geqslant 0$. Meanwhile, $f(x)$ approaches 0 as x approaches $+\infty$. It follows that the extreme point can only be where $f(x)$ achieves its maximum, which completes the proof. ∎

V. Single Cell Noisy Programming with Feedback

In this section, we assume that after every round of programming, we can evaluate the amount of charge that has already been trapped in the cells. That is, we can measure

$\sum_{j=1}^{k}(\alpha V_j + \epsilon_j)$ after the k-th round of programming[1], $\forall k \in [t]$. Therefore, we can adaptively choose the applied voltages according to the current cell level. Similarly, we assume the injection hardness α of the cell is known and fixed, and the programming noise values $\epsilon_1, \ldots, \epsilon_t$ are independent random variables with probability density functions $p_j(x), \forall j \in [t]$.

Our goal is to maximize the probability that after t rounds of programming the final level is in $[\theta - \Delta, \theta + \Delta]$, i.e.,

$$\text{maximize } \mathbf{P}\Big(\theta - \Delta \leqslant \sum_{j=1}^{t}(\alpha V_j + \epsilon_j) \leqslant \theta + \Delta\Big), \qquad \text{(P6)}$$

with $\boldsymbol{V} \in \mathbb{R}_+^t$.

Definition 3. *Let $P(\boldsymbol{V}_1^t, \theta, \Delta, t)$ be the probability that the final cell level after t rounds of programming is in $[\theta - \Delta, \theta + \Delta]$ when the voltages applied are $\boldsymbol{V}_1^t$, where $\boldsymbol{V}_i^j = (V_i, V_{i+1}, \ldots, V_j)$. Let $P(\theta, \Delta, t)$ be the maximum probability over all choices of $\boldsymbol{V}_1^t$, i.e.,*

$$P(\theta, \Delta, t) = \max_{\boldsymbol{V}_1^t \in \mathbb{R}_+^t} P(\boldsymbol{V}_1^t, \theta, \Delta, t),$$

where

$$P(\boldsymbol{V}_1^t, \theta, \Delta, t) = \mathbf{P}\left(\theta - \Delta \leqslant \sum_{j=1}^{t}(\alpha V_j + \epsilon_j) \leqslant \theta + \Delta\right).$$

Suppose the target level and quantization distance are θ and Δ, respectively. Let $P(\theta, \Delta, t)$ be as in Definition 3. Then the following recursion holds:

$$P(\theta, \Delta, t) = \max_{V_1 \in \mathbb{R}_+} \int_{\mathbb{R}_+} p_1(x - \alpha V_1) P(\theta - x, \Delta, t-1) \mathrm{d}x,$$

for $t \geqslant 2$ and

$$P(\theta, \Delta, 1) = \max_{V_1 \in \mathbb{R}_+} \int_{\theta-\Delta}^{\theta+\Delta} p_1(x - \alpha V_1) \mathrm{d}x.$$

We can compute $P(\theta, \Delta, t)$ numerically using the recursion once we know the distribution of the noise $p_j(x), j \in [t]$. However, analytical results are difficult to derive since the noise distribution $p_j(x), j \in [t]$ could be an arbitrary probability distribution. In the sequel, we assume a simple yet nontrivial noise distribution, namely, ϵ_j is uniformly distributed over $[\alpha V_j - \delta_1 V_j, \alpha V_j + \delta_2 V_j]$ for $j \in [t]$, where $0 \leqslant \delta_1 \leqslant \alpha$ and $\delta_2 \geqslant 0$. Thus $p_j(x) = \frac{1}{(\delta_1+\delta_2)V_j} I_{x \in [-\delta_1 V_j, \delta_2 V_j]}$. This assumption is very similar to the one made in [6]. The size of the support set of the noise distribution is proportional to the programming voltage, which is reasonable since larger voltages result in larger deviations of the noise distribution.

Theorem 3. *In Definition* 3,

$$P(\theta, \Delta, 1) = \begin{cases} 1, & \text{if } \frac{\theta-\Delta}{\theta+\Delta} < \frac{\alpha-\delta_1}{\alpha+\delta_2}, \\ \frac{\alpha+\delta_2}{\delta_1+\delta_2}\frac{2\Delta}{\theta+\Delta} & \text{if } \frac{\theta-\Delta}{\theta+\Delta} \geqslant \frac{\alpha-\delta_1}{\alpha+\delta_2}, \end{cases}$$

and the optimal solution is achieved by $V_1 = \frac{\theta+\Delta}{\alpha+\delta_2}$.

Next we would like to find the values of $\boldsymbol{V}_1^t$ that maximize $P(\boldsymbol{V}_1^t, \theta, \Delta, t)$ with feedback information, for arbitrary t.

Lemma 5. *$P(\theta, \Delta, t)$ is a non-increasing function of θ.*

Lemma 6. *$P(\boldsymbol{V}_1^t, \theta, \Delta, t)$ is maximized when $V_1 = \frac{\theta+\Delta}{\alpha+\delta_2}$.*

[1]Measuring the exact amount of charge injected is time consuming for real applications, thus it is common to compare the cell level to certain threshold values and to obtain a range for the cell level. In this work, we follow the assumption that the actual cell level is available, as in [6].

Next we give an algorithm for determining the optimal cell programming for Problem (P6), where feedback information is available.

Algorithm 2 *Suppose the cell voltage is x_j before the j-th write, where $1 \leqslant j \leqslant t$ and $x_1 = 0$. Then on the j-th write, set $V_j = \frac{\theta - x_j + \Delta}{\alpha+\delta_2}$.*

Theorem 4. *Algorithm* 2 *gives an optimal solution for the cell programming problem* (P6).

VI. Conclusion

The study of cell programming for flash memories is an important step toward understanding their storage capacity. Flash memories have a unique property that the cell levels can only increase during programming, which gives rise to a monotonic cell-programming model. In this paper, a new criterion to measure the performance of cell programming is proposed. Both noiseless parallel programming and noisy single cell programming are studied. The potential benefit of using feedback to adaptively choose the programming voltages is considered.

VII. Acknowledgment

This research was supported in part by the ISEF Foundation, the Lester Deutsch Fellowship, the University of California Lab Fees Research Program, Award No. 09-LR-06-118620-SIEP, the National Science Foundation under Grant CCF-1116739, and the Center for Magnetic Recording Research at the University of California, San Diego.

The authors would like to thank Lele Wang for her comments on the statement and proof of Lemma 2.

References

[1] V. Bohossian, A. Jiang, and J. Bruck, "Buffer codes for asymmetric multi-level memory," in *Proc. IEEE Int. Symp. Inform. Theory*, June 2007, pp. 1186–1190.

[2] P. Cappelletti, C. Golla, P. Olivo, and E. Zanoni, *Flash Memories*. Kluwer Academic Publishers, 1st Edition, 1999.

[3] K. D. S. et al, "A 3.3V 32 Mb NAND flash memory with incremental step pulse programming scheme," *IEEE Journal of Solid-State Circuits*, vol. 30, no. 11, pp. 1149–1156, November 1995.

[4] D. Haugland, "A bidirectional greedy heuristic for the subspace selection problem," *Lecture Notes in Computer Science*, vol. 4638, pp. 162–176, August 2007.

[5] A. Jiang, V. Bohossian, and J. Bruck, "Floating codes for joint information storage in write asymmetric memories," in *Proc. IEEE Int. Symp. Inform. Theory*, June 2007, pp. 1166–1170.

[6] A. Jiang and H. Li, "Optimized cell programming for flash memories," in *Proc. IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, August 2009, pp. 914–919.

[7] A. Jiang, H. Li, and J. Bruck, "On the capacity and programming of flash memories," *IEEE Trans. Inform. Theory*, vol. 58, no. 3, pp. 1549 – 1564, March 2012.

[8] A. Jiang, R. Mateescu, M. Schwartz, and J. Bruck, "Rank modulation for flash memories," *IEEE Trans. Inform. Theory*, vol. 55, no. 6, pp. 2659 – 2673, June 2009.

[9] H. T. Lue, T. H. Hsu, S. Y. Wang, E. K. Lai, K. Y. Hsieh, R. Liu, and C. Y. Lu, "Study of incremental step pulse programming (ISPP) and STI edge effiect of BE-SONOS NAND flash," in *Proc. IEEE int. Symp. on Reliability Physiscs*, vol. 30, no. 11, May 2008, pp. 693–694.

[10] B. K. Natarajan, "Sparse approximate solutions to linear systems," *SIAM J. Comput.*, vol. 30, no. 2, pp. 227–234, April 1995.

[11] R. Rivest and A. Shamir, "How to reuse a write-once memory," *Inform. and Contr.*, vol. 55, no. 1-3, pp. 1–19, December 1982.

[12] E. Yaakobi, A. Jiang, P. H. Siegel, A. Vardy, and J. K. Wolf, "On the parallel programming of flash memory cells," in *Proc. IEEE Inform. Theory Workshop*, Dublin, Ireland, August-September 2010, pp. 1–5.