Optimized Cell Programming for Flash Memories With Quantizers

Minghai Qin, *Student Member, IEEE*, Eitan Yaakobi, *Member, IEEE*, and Paul H. Siegel, *Fellow, IEEE*

***Abstract*— Multilevel flash memory contains blocks of cells that represent data by the amount of charge stored in them. The cell writing—or *programming*—process applies specified voltages in a sequential manner, injecting charge to achieve a desired level. Reducing a cell level requires a costly block erasure, so programming only increases cell levels. Parallel programming, whereby a common voltage is applied to a group of cells to inject charge simultaneously, simplifies circuitry and increases programming speed. However, cell-to-cell variations and limited programming round can adversely affect its precision. In this paper, we consider algorithms for efficient cell programming. Since cell levels are quantized to a discrete set of values, our objective is to minimize the number of cells that are not quantized to their target levels. For a specified number of programming rounds, we derive an optimal parallel programming algorithm with complexity that is polynomial in the number of cells. We extend the algorithm to account for intercell interference, where the voltage applied to a cell can affect the level of adjacent cells. We then consider noisy programming of a single cell, with and without feedback about the cell level. In both scenarios, we present an algorithm that, for a given number of programming rounds, minimizes the probability of an incorrect cell level.**

***Index Terms*— Parallel programming, flash memory programming, intercell interference.**

I. Introduction

FLASH memories are a widely-used technology for nonvolatile 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; the amount of charge stored in a cell determines its *level*. The hot-electron injection mechanism or Fowler-Nordheim tunneling mechanism [3] 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 in programming and erasing. That is, increasing a cell level (injecting charge into a cell) is easy to accomplish by applying a certain voltage to the cell. However, decreasing a cell level (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 them to their target levels. The erase operation, called a *block erasure*, is not only time-yconsuming, but also degrades the performance and reduces the longevity of the flash memory [3].

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 [2], [6], and [15].

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 [17]. Instead of applying distinct program voltages to different cells, parallel programming applies a common program voltage to many cells simultaneously. Consequently, the write speed is increased and the complexity of hardware realization is substantially reduced. Parallel programming must also account for the fact that cells have different hardness with respect to charge injection [13], [16]. 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 [13], [16], 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 [7] and [8], optimized programming for a single flash memory cell was studied. A programming strategy to optimize the precision with respect to two cost functions was proposed,

Manuscript received October 9, 2012; revised August 13, 2013; accepted October 11, 2013. Date of publication November 26, 2013; date of current version April 17, 2014. This work was supported in part by the International Sephardic Education Foundation, the Lester Deutsch Fellowship, the University of California Laboratory Fees Research Program under Award 09-LR-06-118620-SIEP, in part by the National Science Foundation under Grant CCF-1116739, and in part by the Center for Magnetic Recording Research at the University of California, San Diego. This paper was presented at the 2012 IEEE International Symposium on Information Theory.

M. Qin and P. H. Siegel are with the Department of Electrical and Computer Engineering and the Center for Magnetic Recording Research, University of California, San Diego, La Jolla, CA 92093 USA (e-mail: mqin@ucsd.edu; psiegel@ucsd.edu).

E. Yaakobi is with the Department of Electrical Engineering, California Institute of Technology, Pasadena, CA 91125 USA, and also with the Department of Electrical and Computer Engineering and the Center for Magnetic Recording Research, University of California, San Diego, La Jolla, CA 92093 USA (e-mail: eyaakobi@ucsd.edu).

Communicated by E. Arikan, Associate Editor for Coding Theory.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIT.2013.2292819

where one of the cost functions is the ℓ_p metric and the other is related to rank modulation [9]. It was assumed that the programming noise, which is the difference between the ideal and actual trapped charge in the cell, follows a uniform distribution and the level increment is chosen adaptively according to the current cell level to minimize the cost function.

In [17], 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) [5] and the Sparse Approximate Solution Problem (SAS) [14], 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 [3]. 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 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.

To formulate this more precisely, let $\Theta = (\theta_1, \ldots, \theta_n)$ be the vector of target cell levels and let $\boldsymbol{\ell}_t = (\ell_{1,t}, \ldots, \ell_{n,t})$ be 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 $\mathcal{C}(\Theta, \boldsymbol{\ell}_t)$ involving 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 [17], the cost function was based upon the ℓ_p metric, i.e., $\mathcal{C}_p(\Theta, \boldsymbol{\ell}_t) = \left(\sum_{i=1}^{n} \left|\theta_i - \ell_{i,t}\right|^p\right)^{\frac{1}{p}}$. In this paper, we study a cost function motivated by the quantization of cell levels, namely

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

where Δ_i is the *quantization distance* for the i-th cell. We analytically solve the corresponding problem of finding an optimal parallel programming algorithm for the special case where the hardness of each cell is known and there is no programming noise. We also derive optimal programming algorithms for a single cell in the presence of noise, both with and without the availability of feedback during the programming process. We focus upon these scenarios because of their amenability to analysis; the solution to the general cell-programming problem remains open.

Another factor that limits the precision of flash programming is inter-cell interference (ICI), which can cause the level of a cell to increase when its neighboring cells are programmed. The ICI is caused by the parasitic capacitance between neighboring cells, and it is of particular concern in multilevel cell programming [4], [11]. Constrained codes that mitigate the effect of ICI have been studied in [1] and [12]. In this paper, we consider a simple model of ICI and, by application of dynamic programming in the form of the Viterbi algorithm, we derive an optimal, linear-time algorithm for parallel programming with ICI in the absence of noise.

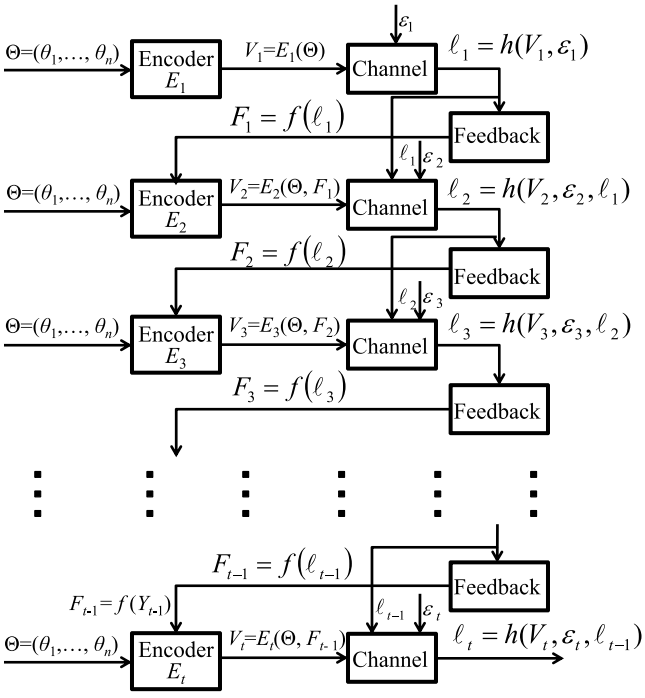

Fig. 1. The information-theoretic framework of the cell programming model.

The rest of the paper is organized as follows. In Section II, we formulate the parallel programming problem with the cost function that reflects the quantization of cell levels. In Section III, we derive a polynomial-time algorithm for optimal parallel programming in the absence of noise, assuming known, deterministic cell-hardness parameters. In Section IV, we extend this to a polynomial-time, optimal parallel programming algorithm for the case where ICI is present. In Section V, we study the problem of programming a single cell in the presence of noise, but with no feedback on the cell level during programming. In Section VI, we then consider noisy cell programming with applied voltages chosen adaptively using feedback about the current cell level.

II. Preliminaries

Let us define the cell programming problem in a general information-theoretic framework, such as a cascade channel described in Fig. 1, where the number of channels is the number of programming rounds, t. We assume that there are n cells, denoted by $c_1, c_2, \ldots, c_n$, whose initial *levels* (i.e., charge levels) are all 0. Each cell is characterized by some *target level* $\theta_i \geqslant 0$ and the target-level vector is $\Theta \triangleq (\theta_1, \ldots, \theta_n)$. Each round of programming is first described by an encoder E_i, $1 \leqslant i \leqslant t$. The input to the first encoder is the target-level vector. For the other encoders, the input also includes feedback on the cell levels after the previous round of programming. The output of encoder E_i is the vector V_i

which includes information about the programming voltage of the i-th round and the set of cells that are programmed with this voltage. The output of the i-th channel, which is the outcome of the i-th round of programming, is a function of V_i and ϵ_i as well as $\boldsymbol{\ell}_{i-1}$ if $i > 1$. The vector $\boldsymbol{\epsilon}_i$ represents the noise in each cell and any other property of the cell that will affect its level. For $i > 1$, the vector $\boldsymbol{\ell}_i$ represents the value of cells after the i-th round. For round $i+1$, the outcome of the i-th round of programming $\boldsymbol{\ell}_i$ is used to generate a feedback vector F_i on the cell levels to be used in the next round of programming. The goal is to minimize a cost function that measures the difference between the channel output $\boldsymbol{\ell}_t$ after t rounds of programming and the target level vector Θ.

Remark 1. In practice, electrons trapped in the oxide layer can cause transient charge leakage during programming, which leads to a slight decrease of cell levels. Since the leakage and other detrimental factors are typically significantly smaller than the programming noise and inter-cell interference discussed in Section IV, we simply assume the cell levels can only increase during programming.

Feedback information on cell levels after a particular write can be used to adaptively choose the programming voltage of the next round, thus increasing the precision of programming. However, obtaining the feedback information is time- and energy-consuming since reading the cell level is accomplished by comparing it to a sequence of reference values.

In the remainder of the paper, we denote by $[m : n]$ the integer set $\{i \in \mathbb{Z} : m \leqslant i \leqslant n\}$. We will sometimes shorten $[1 : n]$ to $[n]$ when the meaning is clear from the context. We will use $\mathbb{R}_+$ to denote the set of all non-negative real numbers, i.e., $\mathbb{R}_+ = \{x \in \mathbb{R} : x \geqslant 0\}$.

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

$$\alpha V + \epsilon,$$

where α and ϵ are random variables that might be different for each cell c_i and each round of programming. Each cell c_i is associated with an α and we call it 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 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 and let $\boldsymbol{V} = (V_1, V_2, \ldots, V_t)^T \in \mathbb{R}_+^t$ be the vector of voltages applied on the t rounds of programming. Define the *indicator matrix*

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

where, for $i \in [n]$ and $j \in [t]$, the entry $b_{i,j} \in \{0, 1\}$ indicates whether the cell 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. Denote the programming noise of the i-th cell on the j-th programming round by $\epsilon_{i,j}$, for $i \in [n]$ and $j \in [t]$. For $i \in [n]$, let $\ell_{i,t}$ be the random variable representing the level of cell c_i after t rounds of programming; that is,

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

We define $\boldsymbol{\ell}_t = (\ell_{1,t}, \ldots, \ell_{n,t})$ to be the cell-state vector after t rounds of programming.

We evaluate the performance of the programming by reference to a 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 Θ, $\boldsymbol{\alpha}$ and $\{\epsilon_{i,j}\}_{n\times t}$, we seek to solve the optimization problem

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

over $\boldsymbol{V} \in \mathbb{R}_+^t$ and $\mathbf{B} \in \{0,1\}^{t\times n}$, where, for a random variable X, $\mathbb{E}[X]$ denotes its expected value.

In [17], 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 as the cost function the number of cells that are not correctly quantized to their target levels. 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 $\mathbf{B} \in \{0,1\}^{t\times n}$.

Remark 2. Problem (P2) is the most general form of the cell programming problem and, therefore, difficult to solve analytically. In the following sections of the paper, we consider four special cases, of both theoretical and practical interest, for which analytical solutions can be found.

Remark 3. Another concern in programming is the writing speed, which strongly depends on the number of programming rounds. Therefore, an alternative criterion for evaluating the performance of a programming method is the minimum number of programming rounds needed to achieve a specified level of programming accuracy, as described by the expected cost. That is, given the values of Θ, $\boldsymbol{\alpha}$ and $\{\epsilon_{i,j}\}_{n\times t}$, we seek to determine

$$\min_{\boldsymbol{V}\in\mathbb{R}_+^t, \mathbf{B}\in\{0,1\}^{t\times n}} t, \text{ subject to } \mathbb{E}[\mathcal{C}_{\boldsymbol{\Delta}}(\Theta, \boldsymbol{\ell}_t)] \leqslant \gamma, \tag{P2$'$}$$

where γ is the maximum allowable expected cost.

If t can be bounded above by a finite number $t_{\max}$, Problem (P2$'$) can be translated to Problem (P2) through a binary

search for t between 0 and $t_{\max}$. If there exists an algorithm with time complexity $O(f(n))$ for Problem (P2), then there exists an algorithm with time complexity $O(\log(t_{\max})f(n))$ for (P2′). As $t_{\max}$ is usually a small number between 6 and 10 in practice [16], [17], we focus on solving Problem (P2) throughout this paper.

Remark 4. In practice, quantization is performed by comparing to predetermined voltage levels. The number of such reference levels may therefore affect the read latency, as well as the chip area in a circuit implementation. The trade-off between storage capacity – a function of the number of levels – and quantization speed is determined by the cell quantization distances, $\{\Delta_i\}$. In the case of a uniform quantizer with quantization distance Δ, if the maximum cell level is $\theta_{\max}$, then the number of levels equals $\lfloor \frac{\theta_{\max}}{2\Delta} \rfloor + 1$.

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 $n, t, \mathbf{\Delta} = \{\Delta_1, \ldots, \Delta_n\}$ and $\Theta = \{\theta_1, \ldots, \theta_n\}$ denote the block length, the number of programming rounds, the set of quantization distances, and the set of target levels, respectively. Our goal is to find a solution to (P2).

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

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

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 lemma. ∎

The quantities u_i and v_i defined in the statement of Lemma 1 represent the boundaries of the correct quantization interval for cell c_i, $\forall i \in [n]$. We call them the *upper threshold point* and the *lower threshold point* for cell c_i and we call the interval $[u_i, v_i]$ the *quantization interval* for cell 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.

Example 1. Suppose the target levels are $\Theta = (10, 13, 8, 5, 10)$, the quantization distances are $\mathbf{\Delta} = (2, 2, 2, 3, 1)$, and the cell hardness parameters are $\boldsymbol{\alpha} = (0.5, 0.5, 1, 1, 0.5)$. According to Lemma 1, $(u_1, \ldots, u_5) = (16, 22, 6, 2, 18)$ and $(v_1, \ldots, v_5) = (24, 30, 10, 8, 22)$. Then $\mathcal{T} = \{2, 6, 8, 10, 16, 18, 22, 24, 30\}$.

Remark 5. We assume that $|\mathcal{T}| > t$ since otherwise we can easily achieve $\mathcal{C}_{\mathbf{\Delta}}(\Theta, \boldsymbol{\ell}_t) = 0$ by using the set of threshold points, $\mathcal{T}$, as the set of programming voltages $\{V_1, \ldots, V_t\}$.

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 achieved by applying $\boldsymbol{V}$.

Remark 6. Note that, for any $i \in [n]$, 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 the level of c_i can be quantized to the correct target level.

For a fixed $\boldsymbol{V}$, maximizing the function $f(\boldsymbol{V}, \mathbf{B})$ over $\mathbf{B}$ is easy to accomplish by checking whether there exists, for each i, an attainable voltage level $z \in S_{\boldsymbol{V}}$ such that $u_i \leqslant z \leqslant v_i$. If one could enumerate all possible vectors $\boldsymbol{V}$, one could use this approach to exhaustively search for an optimal solution. However, since $\boldsymbol{V}$ can be, in principle, any vector in $\mathbb{R}_+^t$, there is an uncountably infinite number of possible choices of $\boldsymbol{V}$ to consider. Nevertheless, Lemma 2 states that we can limit the number of vectors $\boldsymbol{V}$ under consideration to be polynomial in n, and still guarantee that an optimal solution will be found.

Lemma 2. There exists a matrix $\mathbf{A} \in \{0,1\}^{t \times t}$, invertible over $\mathbb{R}$, such that

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

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

Proof: See Appendix A. ∎

Remark 7. In Lemma 2 and Algorithm 1 below, the binary matrix $\mathbf{A}$ has to be invertible over $\mathbb{R}$, not necessarily over $GF(2)$. Therefore, enumerating only the binary matrices invertible over $GF(2)$ may not be sufficient to find an optimal solution.

Algorithm 1 Parallel Programming

Function $(f^*, \boldsymbol{V}^*, \mathbf{B}^*) = ParallelProgramming\left(t, u_1^n, v_1^n\right)$.
Input:
t, $(u_1, \ldots, u_n)$, and $(v_1, \ldots, v_n)$.
Output:
f^*: maximum value of Problem (P3);
$(\boldsymbol{V}^*, \mathbf{B}^*)$: the optimal solution pair.

1. Let $f^* = 0$;
2. Let $\boldsymbol{V} = \boldsymbol{V}^* = (0, \ldots, 0) \in \mathbb{R}_+^t$,
3. Let $\mathbf{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n) \in \{0,1\}^{t\times n}$, $\boldsymbol{b}_i = \mathbf{0}, \forall i \in [n]$;
4. Let $\mathbf{B}^* \in \{0,1\}^{t\times n}$, $b^*_{i,j} = 0, \forall i \in [t], j \in [n]$;
5. For $i = 1, 2, \ldots, N$ {
6. For $j = 1, 2, \ldots, Q$ {
7. If $\widetilde{\mathbf{A}}_j$ is invertible and $\widetilde{\mathbf{A}}_j^{-1} \cdot \boldsymbol{p}_i \in \mathbb{R}_+^t${
8. Let $\boldsymbol{V} = \widetilde{\mathbf{A}}_j^{-1} \cdot \boldsymbol{p}_i$;
9. Let $f = 0$;
10. For $k = 1, 2, \ldots, n$ {
11. If $\exists z \in S_{\boldsymbol{V}}$, such that $u_k \leqslant z \leqslant v_k$ {
12. Find $\boldsymbol{b} \in \{0,1\}^t$, such that $\boldsymbol{b}^T \cdot \boldsymbol{V} = z$;
13. Let $\boldsymbol{b}_k = \boldsymbol{b}$;
14. $f = f + 1$;
15. }
16. }
17. If $f > f^*$
18. $f^* = f$, $\boldsymbol{V}^* = \boldsymbol{V}$, $\mathbf{B}^* = \mathbf{B}$;
19. }}}

Output the optimal solution pair $(\boldsymbol{V}^*, \mathbf{B}^*)$ with maximized $f(\boldsymbol{V}^*, \mathbf{B}^*) = f^*$.

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). First, 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. Now, let $\widetilde{\mathbf{A}} \in \{0,1\}^{t\times t}$ be a binary matrix with distinct rows; the number of such matrices 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 these matrices. Algorithm 1 will iterate over all choices of $\boldsymbol{p}$ and those matrices $\widetilde{\mathbf{A}}$ that are invertible.

Example 2. Suppose $\boldsymbol{\alpha}$, Θ, $\boldsymbol{\Delta}$, u_i and v_i, $1 \leqslant i \leqslant 5$, are given as in Example 1. Suppose the number of programming rounds is $t = 2$. If $\boldsymbol{p}_i = (30, 8)^T \in \mathcal{T}^2$ and

$$\widetilde{\mathbf{A}}_j = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

as we iterate through Line 5 to Line 19, then

$$\boldsymbol{V} = (V_1, V_2)^T = \widetilde{\mathbf{A}}_j^{-1} \cdot \boldsymbol{p}_i = (8, 22)^T,$$

and

$$S_{\boldsymbol{V}} = \{0, 8, 22, 30\}.$$

By choosing the indicator matrix as

$$\mathbf{B} = \begin{bmatrix} b_{1,1} & b_{2,1} & b_{3,1} & b_{4,1} & b_{5,1} \\ b_{1,2} & b_{2,2} & b_{3,2} & b_{4,2} & b_{5,2} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix},$$

the final cell levels are $\boldsymbol{\ell}_2 = (11, 15, 8, 8, 11)$, where

$$\ell_{i,2} = \alpha_i \sum_{j=1}^{2} V_j b_{i,j} = \alpha_i \left(V_1 b_{i,1} + V_2 b_{i,2}\right), \forall 1 \leqslant i \leqslant 5.$$

It can be easily checked that

$$\theta_i - \Delta_i \leqslant \ell_{i,2} \leqslant \theta_i + \Delta_i, \forall 1 \leqslant i \leqslant 5,$$

implying all cells are correctly quantized and $\boldsymbol{V} = (8, 22)^T$ is an optimal solution.

Theorem 1. Algorithm 1 finds an optimal solution pair $(\boldsymbol{V}^*, \mathbf{B}^*)$ and computes the optimal value $f(\boldsymbol{V}^*, \mathbf{B}^*)$ for Problem (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 considered 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 $O(n^{t+1})$. ∎

Remark 8. The efficiency of the algorithm could be improved if, rather than iterating over the $Q = \prod_{k=0}^{t-1}(2^t - k)$ binary matrices with distinct rows, we instead iterated only over the set of binary matrices that are invertible over $\mathbb{R}$.

IV. Noiseless Parallel Programming With Inter-Cell Interference

In this section, we consider the scenario where cell density has increased to the point that inter-cell interference (ICI) exists. The phenomenon of ICI in flash memories was studied in [4] and [11] and constrained codes that mitigate the effect of ICI were presented in [1] and [12]. In this section, we extend the results of Sections II and III, formulating the cell programming problem with ICI as an optimization problem and providing an efficient polynomial time algorithm to solve it.

Suppose the cell layout is a one-dimensional array. When a cell is programmed by applying a voltage to it, the levels of the left and right neighboring cells will also increase. Those cells that cause the ICI are called *interfering cells* and those cells whose levels are increased unexpectedly because of ICI are called *victim cells*. Since a large programming voltage will result in a more severe ICI, we make the further assumption that the ICI of the victim cell is proportional to the voltage applied to the interfering cell. We define a sequence

of parameters $\beta_{i,i+1} \in \mathbb{R}_+$ and $\beta_{i+1,i} \in \mathbb{R}_+, i \in [n-1]$ to describe the effect of ICI from c_i to c_{i+1} and from c_{i+1} to c_i, respectively.

To be more precise, suppose the flash memory cells are $\boldsymbol{c} = (c_1, \ldots, c_n)$ with injection hardness parameters $(\alpha_1, \ldots, \alpha_n)$. There are t rounds of charge injection, corresponding to a set of applied voltages $(V_1, \ldots, V_t)$. There is no programming noise, i.e., $\epsilon_{i,j} = 0, \forall i \in [n], j \in [t]$. If the voltage applied to the cell c_i in round j is V_j, then

- the cell level of c_i is increased by $\alpha_i b_{i,j} V_j$, for all $i \in [n]$
- the cell level of c_{i+1} is increased by $\alpha_{i+1} b_{i+1,j} \beta_{i,i+1} V_j$, for all $i \in [n-1]$
- the cell level of c_{i-1} is increased by $\alpha_{i-1} b_{i-1,j} \beta_{i,i-1} V_j$, for all $i \in [2:n]$.

We represent the indicator matrix as $\mathbf{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n) \in \{0,1\}^{t \times n}$, where the vector $\boldsymbol{b}_j$, for $j \in [n]$, denotes the j-th column of $\mathbf{B}$. For convenience, we also define $\beta_{0,1} = \beta_{n+1,n} = 0$, reflecting the fact that the leftmost cell c_1 and rightmost c_n have only one neighboring cell each. Similarly, we define $\boldsymbol{b}_0 = (b_{0,1}, \ldots, b_{0,t})^T = \mathbf{0}, \boldsymbol{b}_{n+1} = (b_{n+1,1}, \ldots, b_{n+1,t})^T = \mathbf{0}$.

Now, let $\boldsymbol{\beta}$ denote the vector $(\beta_{0,1}, \beta_{1,2}, \beta_{2,3}, \ldots, \beta_{n,n+1}, \beta_{2,1}, \ldots, \beta_{n,n-1}, \beta_{n+1,n})$. Assuming there is no programming noise, the final cell level of $c_i, i \in [n]$ after t rounds of programming can be written as

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

The cell programming problem is the same as Problem (P2), which is to solve

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

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

Lemma 3. The solution to Problem *(*P2*)* is equivalent to the solution of the following:

$$\textit{maximize } \hat{f}(\boldsymbol{V}, \mathbf{B}), \tag{P3$'$}$$

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

$$\hat{f}(\boldsymbol{V}, \mathbf{B}) = \left|\left\{i \in [n] : u_i \leqslant (\boldsymbol{b}_i^T + \beta_{i+1,i} \boldsymbol{b}_{i+1}^T + \beta_{i-1,i} \boldsymbol{b}_{i-1}^T) \cdot \boldsymbol{V} \leqslant v_i\right\}\right|.$$

Proof: The following chain of equations is easily established.

$$\begin{aligned}
&\min_{\boldsymbol{V},\mathbf{B}} \left|\{i : |\theta_i - \ell_{i,t}| > \Delta_i, i \in [n]\}\right| \\
&= n - \max_{\boldsymbol{V},\mathbf{B}} \left|\{i : |\theta_i - \ell_{i,t}| \leqslant \Delta_i, i \in [n]\}\right| \\
&= n - \max_{\boldsymbol{V},\mathbf{B}} \left|\left\{i \in [n] : \left|\frac{\theta_i}{\alpha_i} - \frac{\ell_{i,t}}{\alpha_i}\right| \leqslant \frac{\Delta_i}{\alpha_i}\right\}\right| \\
&= n - \max_{\boldsymbol{V},\mathbf{B}} \left|\left\{i \in [n] : \right.\right. \\
&\qquad \left.\left. \left|\frac{\theta_i}{\alpha_i} - (\boldsymbol{b}_i^T + \beta_{i+1,i} \boldsymbol{b}_{i+1}^T + \beta_{i-1,i} \boldsymbol{b}_{i-1}^T) \cdot \boldsymbol{V}\right| \leqslant \frac{\Delta_i}{\alpha_i}\right\}\right| \\
&= n - \max_{\boldsymbol{V},\mathbf{B}} \left|\left\{i \in [n] : \right.\right. \\
&\qquad \left.\left. \frac{\theta_i - \Delta_i}{\alpha_i} \leqslant (\boldsymbol{b}_i^T + \beta_{i+1,i} \boldsymbol{b}_{i+1}^T + \beta_{i-1,i} \boldsymbol{b}_{i-1}^T) \cdot \boldsymbol{V} \leqslant \frac{\theta_i + \Delta_i}{\alpha_i}\right\}\right| \\
&= n - \max_{\boldsymbol{V},\mathbf{B}} \left|\left\{i \in [n] : \right.\right. \\
&\qquad \left.\left. u_i \leqslant (\boldsymbol{b}_i^T + \beta_{i+1,i} \boldsymbol{b}_{i+1}^T + \beta_{i-1,i} \boldsymbol{b}_{i-1}^T) \cdot \boldsymbol{V} \leqslant v_i\right\}\right|,
\end{aligned}$$

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 $v_i = \frac{\theta_i + \Delta_i}{\alpha_i}$. ■

As was the case for Problem (P3) in Section III, the optimization in Problem (P3$'$) is over the applied voltage vector $\boldsymbol{V}$ and the binary indicator matrix $\mathbf{B}$. In Section III, however, there was no ICI and the cells could be treated independently. Consequently, for a fixed voltage vector $\boldsymbol{V}$, we could maximize $f(\boldsymbol{V}, \mathbf{B})$ over $\mathbf{B}$ simply by checking for each $i \in [n]$ individually whether there exists an attainable value $z \in S_{\boldsymbol{V}}$ such that $u_i \leqslant z \leqslant v_i$. The time complexity of this procedure is $O(n)$.

Unfortunately, this procedure is not applicable in the presence of ICI because the level of a given cell depends on the voltage increments applied to neighboring cells. Therefore, to solve Problem (P3$'$), we first develop an efficient algorithm with time complexity $O(n)$ to maximize $\hat{f}(\boldsymbol{V}, \mathbf{B})$ over $\mathbf{B}$ for a fixed $\boldsymbol{V}$. We then prove a generalization of Lemma 2 that limits the number of candidate voltage vectors $\boldsymbol{V}$ to be polynomial in n, and, finally, present an efficient algorithm to search for the optimal solution pair $(\boldsymbol{V}, \mathbf{B})$.

We call $s_i = (\boldsymbol{b}_{i-1}, \boldsymbol{b}_i) \in \{0,1\}^{t \times 2}$ the ***state*** of cell c_i, $\forall i \in [n+1]$. Note that the last column of s_{n+1} is all zeros. Let $s_0 \in \{0,1\}^{t \times 2}$ be the all-zero matrix. For example, if $t = 1$, then there are 4 states, corresponding to all binary vectors of length 2, i.e., $(0,0), (0,1), (1,0), (1,1)$. We can relate Problem (P3$'$) to an optimization problem over a trellis $\mathbf{T}$, which we define as follows [10].

Definition 3. A trellis $\mathbf{T}$ of depth n is a triplet $(\mathbf{S}, \mathbf{E}, \mathbf{L})$, where $\mathbf{S} = \mathbf{S}_0 \cup \mathbf{S}_1 \cup \mathbf{S}_2 \cup \cdots \cup \mathbf{S}_n$ denotes the set of states; $\mathbf{E} = \mathbf{E}_1 \cup \mathbf{E}_2 \cup \cdots \cup \mathbf{E}_n$ denotes the set of edges, where each edge $e \in \mathbf{E}_i$ has initial state $\sigma(e) \in \mathbf{S}_{i-1}$ and terminal state $\tau(e) \in \mathbf{S}_i$; and $\mathbf{L} : \mathbf{E} \to \Sigma_1$ denotes a label function that assigns to each edge a value in the set Σ_1.

For our cell programming problem, we construct a trellis as follows.

Construction 1 Suppose the number of cells is n and the number of programming rounds is t. We define a trellis $\mathbf{T}$ of depth $n+1$, where $\mathbf{S}_0 = \{\mathbf{0}\}$ and $\mathbf{S}_i$ is the set of states of cell c_i, for all $i \in [n+1]$. There exists as edge $e \in \mathbf{E}_i$ from state $s_i \in \mathbf{S}_i$ to state $s_{i+1} \in \mathbf{S}_{i+1}$ if and only if the last column of s_i is the same as the first column of s_{i+1}. In that case, $\sigma(e) = s_i$ and $\tau(e) = s_{i+1}$.

We will make use of two label functions. The **terminal state label function** is $\mathbf{L} : \mathbf{E} \to \{0,1\}^t$, where for all $e \in \mathbf{E}$, $\mathbf{L}(e)$ equals the last column of $\tau(e)$. The **branch metric label functions** are $\mathbf{L}_i^b : \mathbf{E}_i \to \{0,1\}, \forall i \in [2 : n+1]$, where $\mathbf{L}_i^b(e) = 1$ if and only if $e \in \mathbf{E}_i$ and the cell c_{i-1} can be quantized correctly by the voltage vector $\boldsymbol{V}$ and the submatrix

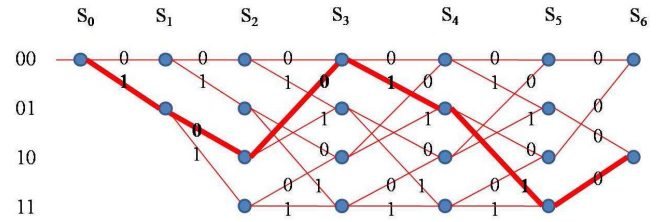

Fig. 2. Illustration of a terminal state label function of a trellis.

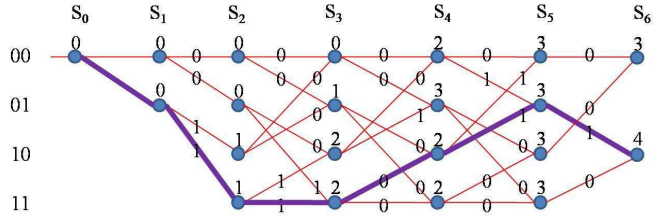

Fig. 3. Path with maximum metric in a trellis with $t = 1$.

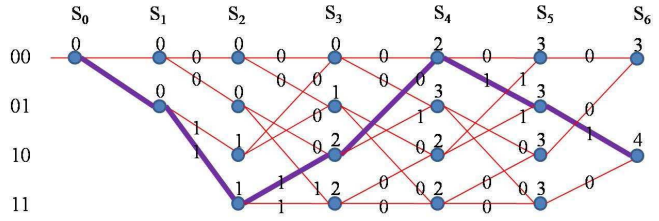

Fig. 4. Another path with maximum metric in a trellis with $t = 1$.

$(\sigma(e_i), \mathbf{L}(e_i)) = (\boldsymbol{b}_{i-2}, \boldsymbol{b}_{i-1}, \boldsymbol{b}_i)$ of the indicator matrix, i.e.,

$$u_{i-1} \leqslant (\boldsymbol{b}_{i-1}^T + \beta_{i,i-1}\boldsymbol{b}_i^T + \beta_{i-2,i-1}\boldsymbol{b}_{i-2}^T) \cdot \boldsymbol{V} \leqslant v_{i-1}.$$

Since the construction of the trellis $\mathbf{T}$ depends upon $t, \boldsymbol{V}, \boldsymbol{\beta}, u_1^n$ and v_1^n, we denote the trellis by $\mathbf{T}(t, \boldsymbol{V}, \boldsymbol{\beta}, u_1^n, v_1^n)$.

A path $\boldsymbol{e} = (e_1, e_2, \ldots, e_n)$ in $\mathbf{T}$ is a sequence of edges, where $\sigma(e_1) \in \mathbf{S}_0, \sigma(e_{i+1}) = \tau(e_i), \forall i \in [1 : n-1]$. The associated *path metric* is defined as $m(\boldsymbol{e}) = \sum_{i=1}^n \mathbf{L}_i^b(e_i)$. A path $\boldsymbol{e}$ also defines an indicator matrix $\mathbf{B}(\boldsymbol{e})$, which is obtained by reading off and concatenating the column vectors corresponding to the terminal state labels of its edges. The path metric $m(\boldsymbol{e})$ can be interpreted as the number of cells being quantized correctly when the voltage tuple is $\boldsymbol{V}$ and the indicator matrix is $\mathbf{B}(\boldsymbol{e})$, i.e., $m(\boldsymbol{e}) = \hat{f}(\boldsymbol{V}, \mathbf{B}(\boldsymbol{e}))$. Therefore, for a fixed $\boldsymbol{V}$, solving Problem (P3′) is equivalent to finding a path $\boldsymbol{e}$ from $\mathbf{S}_0$ to $\mathbf{S}_n$ that maximizes $m(\boldsymbol{e})$, and thus finding the indicator matrix $\mathbf{B}(\boldsymbol{e})$.

Example 3. Fig. 2 shows an example of a trellis $\mathbf{T}$ with terminal state label function for $n = 5$ cells and $t = 1$. The number of states for each cell is $2^{2t} = 4$. The number of paths emanating from each state is $2^t = 2$. For the highlighted path $\boldsymbol{e}$, the corresponding indicator matrix $\mathbf{B}(\boldsymbol{e}) = (1, 0, 0, 1, 1)$.

Next we state the principle of optimality underlying the technique of dynamic programming and, in particular, the well-known Viterbi algorithm.

Algorithm 2 Viterbi Algorithm

Function $(m_i(s), q_i(s)) = \mathit{Viterbi}\big(\mathbf{T}(t, \boldsymbol{V}, \boldsymbol{\beta}, u_1^n, v_1^n)\big)$.
Input:
Trellis $\mathbf{T}(t, \boldsymbol{V}, \boldsymbol{\beta}, u_1^n, v_1^n)$ in Construction 1.
Output:
$m_i(s), \forall i \in [n], s \in \mathbf{S}$: the maximum path metric from s_0 to $s \in \mathbf{S}_i$;
$q_i(s) \in \mathbf{S}_i, \forall i \in [n], s \in \mathbf{S}$: the state sequence corresponding to the path from s_0 to s with maximum metric.
The algorithm has 4 steps.

1) Initialize.
Let $m_0(s) = 0, \forall s \in \mathbf{S}$. Let $q_0(s_0) = s_0$.
2) Add.
For each state $s \in \mathbf{S}_i$, and edge $e \in \mathbf{E}_i$ such that $\tau(e) = s$, let
$$\widetilde{m}_i(e) = m_{i-1}(\sigma(e)) + \mathbf{L}_i^b(e)$$
3) Compare.
For each state $s \in \mathbf{S}_i$, determine edge e^* with $\tau(e^*) = s$, such that $\widetilde{m}_i(e^*) \geqslant \widetilde{m}_i(e), \forall e$ such that $\tau(e) = s$.
4) Select.
Let $m_i(s) = m_{i-1}(\sigma(e^*)) + \mathbf{L}_i^b(e^*)$ and $q_i(s) = (q_{i-1}(\sigma(e^*)), s)$.

Theorem 2. [Principal of Optimality] Let $\boldsymbol{e} = (e_1, e_2, \ldots, e_{i-1}, e_i)$ be a path from state $s_0 \in \mathbf{S}_0$ to state $s_i \in \mathbf{S}_i$ with maximum path metric $m(\boldsymbol{e})$. Let $s_{i-1} \in \mathbf{S}_{i-1}$ be the terminal state of e_{i-1}, i.e., $s_{i-1} = \tau(e_{i-1})$. Then, the path $\widetilde{\boldsymbol{e}} = (e_1, \ldots, e_{i-1})$ from state s_0 to s_{i-1} has the maximum path metric over all paths from s_0 to s_{i-1}.

In Algorithm 2, we present the Viterbi algorithm as applied to the search for the maximum path metric from $\mathbf{S}_0$ to $\mathbf{S}_n$.

Example 4. Let $\boldsymbol{\alpha}, \Theta, \boldsymbol{\Delta}, u_i$ and $v_i, 1 \leqslant i \leqslant 5$, be specified as in Example 1. Suppose $t = 1$, $V_1 = 20$, $\beta_{i+1,i} = \beta_{i,i+1} = 0.2$, $\forall i \in [n-1]$. Figs. 3 and 4 show the trellis structure along with the value of the branch metric label function on each edge. Recall that $\boldsymbol{u} = (16, 22, 6, 2, 18)$ and $\boldsymbol{v} = (24, 30, 10, 8, 22)$. The values $m_i(s), s \in \mathbf{S}_i$ are also shown. The highlighted paths have the maximum path metric from s_0 to any state $s_6 \in \mathbf{S}_6$, namely $m_6(10) = 4$, where the indicator matrices are $(1, 1, 1, 0, 1)$ and $(1, 1, 0, 0, 1)$, respectively.

Theorem 3. Algorithm 2 finds the path with the maximum path metric with time complexity $O(n)$.

Proof: The correctness follows from Theorem 2 and the linear complexity follows from the properties of the Viterbi Algorithm. ■

So far we have constructed an algorithm with linear complexity to determine $\mathbf{B}$ that maximizes $\hat{f}(\boldsymbol{V}, \mathbf{B})$ for a fixed $\boldsymbol{V}$. It is left to determine $\boldsymbol{V}$ that maximizes $\hat{f}(\boldsymbol{V}, \mathbf{B})$.

As in Section III, we define a threshold-point vector $\boldsymbol{p} = (p_{k_1}, \ldots, p_{k_t}) \in \mathcal{T}^t$, where $\mathcal{T}$ is given as in Definition 1, such that p_{k_j} is a threshold point for the k_j-th cell, for $j \in [t]$ and $k_j \in [n]$. For a fixed $\boldsymbol{p}$, we define a finite set of matrices $\mathbf{A}(\boldsymbol{p})$ of size $t \times t$, such that $a_{i,j} \in \{0,\ 1,\ \beta_{k_i+1,k_i},$

Algorithm 3 Parallel Programming With ICI

Function $(f^*, \boldsymbol{V}^*, \mathbf{B}^*) = \mathit{ParallelProgrammingICI}\big(t, u_1^n, v_1^n, \boldsymbol{\beta}\big)$.
Input:
t, $(u_1, \ldots, u_n)$, $(v_1, \ldots, v_n)$ and $\boldsymbol{\beta}$;
Output:
f^*: maximum value of Problem (P3′);
$(\boldsymbol{V}^*, \mathbf{B}^*)$: optimal solution pair.
1. Let $f^* = 0$;
2. Let $\boldsymbol{V} = \boldsymbol{V}^* = (0, \ldots, 0) \in \mathbb{R}_+^t$;
4. Let $\mathbf{B}^* \in \{0,1\}^{t \times n}$, $b^*_{i,j} = 0, \forall i \in [t], j \in [n]$;
5. For $i = 1, 2, \ldots, N$ {
6. For $j = 1, 2, \ldots, Q(\boldsymbol{p}_i)$ {
7. If $\widetilde{\mathbf{A}}_j(\boldsymbol{p}_i)$ is invertible and $\widetilde{\mathbf{A}}_j(\boldsymbol{p}_i)^{-1} \cdot \boldsymbol{p}_i \in \mathbb{R}_+^t$ {
8. Let $\boldsymbol{V} = \widetilde{\mathbf{A}}_j(\boldsymbol{p}_i)^{-1} \cdot \boldsymbol{p}_i$;
8. Construct the trellis $\mathbf{T}(t, \mathbf{V}, \boldsymbol{\beta}, u_1^n, v_1^n)$ according to Construction 1;
9. Let $(m_k(s), q_k(s)) = \mathit{Viterbi}\big(\mathbf{T}(t, \mathbf{V}, \boldsymbol{\beta}, u_1^n, v_1^n)\big)$, for $k \in [n], s \in \mathbf{S}$;
10. Let $s^* = \arg\max_{s \in \mathbf{S}_n} m_n(s)$ and $f = m_n(s^*)$;
11. If $f > f^*$ {
12. $f^* = f$, $\boldsymbol{V}^* = \boldsymbol{V}$;
13. Let the path $\boldsymbol{e} = (s_0, q_1(s^*), q_2(s^*), \ldots, q_n(s^*))$;
14. Let $\mathbf{B}^* = \mathbf{B}(\boldsymbol{e})$;}
15. }}}
Output the optimal solution pair $(\boldsymbol{V}^*, \mathbf{B}^*)$ with maximized $\hat{f}(\boldsymbol{V}^*, \mathbf{B}^*) = f^*$.

$1 + \beta_{k_i+1,k_i}$, $\beta_{k_i-1,k_i}, 1 + \beta_{k_i-1,k_i}$, $\beta_{k_i+1,k_i} + \beta_{k_i-1,k_i}$, $1 + \beta_{k_i+1,k_i} + \beta_{k_i-1,k_i}\}$. To determine the optimal $\boldsymbol{V}$, we make use of the following modified version of Lemma 2.

Lemma 4. There exists a threshold-point vector $\boldsymbol{p}$ and an invertible matrix $\mathbf{A}(\boldsymbol{p})$ in the corresponding finite set of matrices such that

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

where $\boldsymbol{V} \in \mathbb{R}_+^t$ is an optimal solution.

Proof: See Appendix A. ∎

Remark 9. Note that, in contrast to Lemma 2, when ICI is present the matrices that we consider for a given threshold vector $\boldsymbol{p}$ are defined in terms of $\boldsymbol{p}$.

Finally, we give an algorithm to search for an optimal solution to Problem (P3′), which is also an optimal solution to Problem (P2) when ICI exists. 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. 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. For a fixed $\boldsymbol{p}_i, i \in [M]$, a sequence of matrices $\widetilde{\mathbf{A}}(\boldsymbol{p}_i)^{t \times t}$ is formed such that no two rows are the same. Thus, the number of different $\widetilde{\mathbf{A}}(\boldsymbol{p}_i)$'s is $Q(\boldsymbol{p}_i) = \prod_{k=0}^{t-1}(8^t - k)$. Let $\{\widetilde{\mathbf{A}}_1(\boldsymbol{p}_i), \ldots, \widetilde{\mathbf{A}}_{Q(\boldsymbol{p}_i)}(\boldsymbol{p}_i)\}$ be an arbitrary ordering of all possible $\widetilde{\mathbf{A}}(\boldsymbol{p}_i)$'s. Algorithm 3 will iterate over all choices of $\boldsymbol{p}_i$ and those $\widetilde{\mathbf{A}}(\boldsymbol{p}_i)$'s that are invertible.

The proof of the following theorem is similar to that of Theorem 1, so we omit the details.

Theorem 4. Algorithm 3 finds the optimal solution pair $(\boldsymbol{V}^*, \mathbf{B}^*)$ and computes the optimal value $\hat{f}(\boldsymbol{V}^*, \mathbf{B}^*)$ for Problem (P3′). The time complexity of the algorithm is $O(n^{t+1})$.

V. Single Cell Noisy Programming Without Feedback

In this section, programming noise is assumed to exist. To carry out our analysis, we must restrict to the case of programming a single cell, with injection hardness α. The number of programming rounds is again denoted by t, and the programming noise vector $\epsilon_1, \ldots, \epsilon_t$ consists of independently distributed Gaussian random variables with zero means and variances σ_j^2, $j \in [t]$, respectively.

Remark 10. Note that according to this model, after every programming round the level of each cell could decrease because ϵ_j could be negative. We choose to study this model while assuming that the variance σ_j, $j \in [t]$ is much smaller than αV_j, i.e., $\mathbf{P}\left(\alpha V_j + \epsilon_j < 0\right)$ is very small. Thus, 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 that $\sigma_j = \sigma V_j$, $j \in [t]$, 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 larger voltage applied to the cell results in larger power of the programming noise [7]. We further assume that 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 $\boldsymbol{V} \in \mathbb{R}_+^t$.

Lemma 5. The cell programming problem (P4) is equivalent to

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

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

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

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

Proof: We rewrite the probability in (P4) as

$$\mathbf{P}\Big(-\Delta + \theta \leqslant \sum_{j=1}^{t}(\alpha V_j + \epsilon_j) \leqslant \Delta + \theta\Big)$$
$$= \mathbf{P}\Big(\frac{-\Delta + \theta - \alpha \sum_{j=1}^t V_j}{\sqrt{\sum_{j=1}^t (\sigma_j^2)}} \leqslant X \leqslant \frac{\Delta + \theta - \alpha \sum_{j=1}^t V_j}{\sqrt{\sum_{j=1}^t (\sigma_j^2)}}\Big),$$

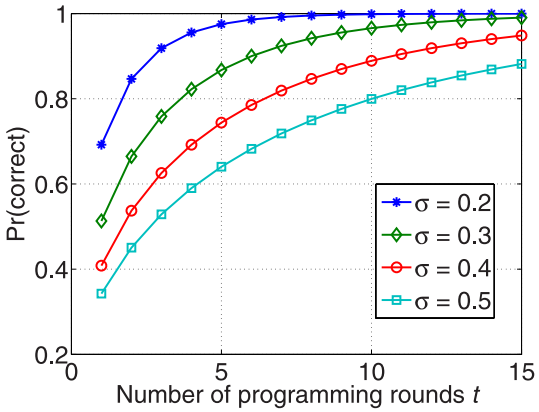

Fig. 5. Probability of correct quantization as a function of the number of programming rounds.

where $X = \frac{\sum_{j=1}^{t}(\alpha V_j+\epsilon_j)-\alpha\sum_{j=1}^{t}V_j}{\sqrt{\sum_{j=1}^{t}\sigma_j^2}} \sim \mathcal{N}(0,1)$. Under the assumption that $\sigma_j = \sigma V_j$, we have

$$\begin{aligned}\mathbf{P}\Big(\frac{-\Delta+\theta-\alpha\sum V_j}{\sigma\sqrt{\sum V_j^2}} \leqslant X \leqslant \frac{\Delta+\theta-\alpha\sum V_j}{\sigma\sqrt{\sum V_j^2}}\Big)\\ = \frac{1}{\sqrt{2\pi}}\int_{\frac{-\Delta+\theta-\alpha\sum V_j}{\sigma\sqrt{\sum V_j^2}}}^{\frac{\Delta+\theta-\alpha\sum V_j}{\sigma\sqrt{\sum V_j^2}}} e^{-u^2/2}\mathrm{d}u,\\ = g(\boldsymbol{V}).\end{aligned}$$

■

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

Remark 11. In the remainder of the paper, we will on occasion simplify notation by writing summations of the form $\sum_{j=1}^{t}(\cdot)$ as $\sum(\cdot)$, provided that the meaning is clear from the context.

Lemma 6. If $\boldsymbol{V}^*$ is the optimal solution to (P5), then $\forall j \in [t]$, $V_j^* = x$, for some constant $x \in R_+$.

Proof: See Appendix B. ■

Theorem 5. The optimal solution $\boldsymbol{V}^*$ to (P5) satisfies the following: $V_j^* = x^*, \forall j \in [t]$, where x^* is the positive root of the equation

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

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

Proof: According to Lemmas 5 and 6, the optimal solution to (P5) is achieved by a sequence of programming voltages $\boldsymbol{V}^* = (V_1,\ldots,V_t)$, where $V_j = x, \forall j \in [t]$, for some $x \in R_+$. Referring to the definition of $g(\boldsymbol{V})$, we must therefore find

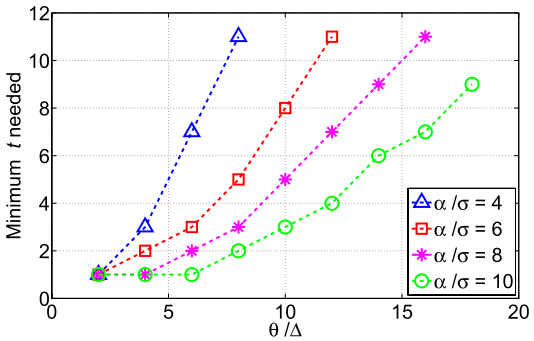

Fig. 6. Minimum number of rounds required to ensure 90% probability of correct programming.

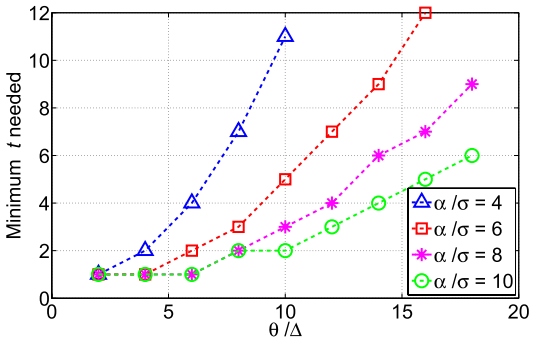

Fig. 7. Minimum number of rounds to ensure 80% probability of correct programming.

$x \in R_+$ that maximizes

$$h(x) = \int_{\frac{a-cx}{x}}^{\frac{b-cx}{x}} e^{-\frac{u^2}{2}}\mathrm{d}u$$

where $a = \frac{-\Delta+\theta}{\sigma\sqrt{t}}, b = \frac{\Delta+\theta}{\sigma\sqrt{t}}$, and $c = \frac{\alpha\sqrt{t}}{\sigma}$. Note that $h(x) \geqslant 0, \forall x \geqslant 0$. Moreover, $h(0) = 0$ and $h(x) \to 0$ as $x \to 0$. To determine a value of x that maximizes $h(x)$, we examine the points where $h'(x)$, the first derivative of $h(x)$, vanishes. A simple calculation shows that

$$\begin{aligned}h'(x) = e^{-\frac{1}{2}\left(\frac{b-cx}{x}\right)^2}\cdot\frac{-cx-(b-cx)}{x^2}\\ -e^{-\frac{1}{2}\left(\frac{a-cx}{x}\right)^2}\cdot\frac{-cx-(a-cx)}{x^2}.\end{aligned}$$

The condition $h'(x) = 0$ translates to

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

Since $\left(2\ln\frac{b}{a}\right)(a^2-b^2) < 0$, this equation has two real solutions, one of which is positive. We denote this solution by x^*. Noting that $h(x)$ is clearly positive for *some* $x \in R_+$, we conclude that the maximum value of $h(x)$ must be achieved when $x = x^*$. This completes the proof. ■

Example 5. Using Theorem 5, a simple calculation shows that the probability of the cell being quantized correctly is a function of three parameters: the number of programming rounds t, the ratio between θ and Δ, and the ratio between α and σ.

Fig. 5 shows the probability of correct programming as a function of the number of programming rounds t for different σ's, where $\alpha = 1, \theta = 1$, and $\Delta = 0.2$. Figs. 6 and 7 show the minimum number of programming rounds t, for different θ/Δ and α/σ, such that the probability of correct quantization is above 90% and 80%, respectively.

VI. 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.[1] That is, we can measure the value $\sum_{j=1}^{k}(\alpha V_j + \epsilon_j)$ after the k-th round of programming, $\forall k \in [t]$. 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 4. 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 4. Then we have

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

Suppose V_1 is the voltage applied on the first programming round. Then

$$P(\boldsymbol{V}_1^t, \theta, \Delta, t) = \int_{\mathbb{R}_+} p_1(x - \alpha V_1) P(\theta - x, \Delta, t-1)\mathrm{d}x.$$

Since feedback information is available, the recursion

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

holds for $t \geqslant 2$. It follows that the problem of finding $P(\theta, \Delta, t)$ can be reduced to the problem of finding $P(\theta - x, \Delta, t-1)$.

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 non-trivial 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 similar to the one made in [7] except that we do not constrain V_j to be integer-valued. 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.

Lemma 7. In Definition 4,

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

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

Proof: See Appendix C. ∎

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

Proof: See Appendix C. ∎

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

Proof: The proof consists of two parts. First we prove that for any $\widehat{\boldsymbol{V}}_1^t \stackrel{\text{def}}{=} (\widehat{V}_1, \ldots, \widehat{V}_t)$ such that $\widehat{V}_1 < V_1 = \frac{\theta+\Delta}{\alpha+\delta_2}$, $\max_{\widehat{\boldsymbol{V}}_2^t} P(\widehat{\boldsymbol{V}}_1^t, \theta, \Delta, t) \leqslant \max_{\boldsymbol{V}_2^t} P(\boldsymbol{V}_1^t, \theta, \Delta, t)$. Next, we prove that for any $\widetilde{\boldsymbol{V}}_1^t \stackrel{\text{def}}{=} (\widetilde{V}_1, \ldots, \widetilde{V}_t)$ such that $\widetilde{V}_1 > V_1 = \frac{\theta+\Delta}{\alpha+\delta_2}$, $\max_{\widetilde{\boldsymbol{V}}_2^t} P(\widetilde{\boldsymbol{V}}_1^t, \theta, \Delta, t) \leqslant \max_{\boldsymbol{V}_2^t} P(\boldsymbol{V}_1^t, \theta, \Delta, t)$.

Case (1): Suppose $\widehat{V}_1 < V_1 = \frac{\theta+\Delta}{\alpha+\delta_2}$.

First we provide a sketch of the proof. If the first voltage applied is V_1 (resp. $\widehat{V}_1$), then the cell level after the first programming round is uniformly distributed over $\mathbb{F} = [(\alpha - \delta_1)V_1, (\alpha + \delta_2)V_1]$ (resp. $\widehat{\mathbb{F}} = [(\alpha - \delta_1)\widehat{V}_1, (\alpha + \delta_2)\widehat{V}_1]$). We will divide $\mathbb{F}$ (resp. $\widehat{\mathbb{F}}$) into non-overlapping intervals and prove that in each interval applying V_1 yields higher probability of correct programming than applying $\widehat{V}_1$.

Let $\ell = \lceil \frac{(\delta_1+\delta_2)\widehat{V}_1}{(\alpha-\delta_1)(V_1-\widehat{V}_1)} \rceil$ and divide $\mathbb{F}$ (resp. $\widehat{\mathbb{F}}$) evenly into ℓ non-overlapping intervals. That is, let $\mathbb{F}_i = \big[(\alpha - \delta_1)V_1 + (i-1)\frac{(\delta_1+\delta_2)V_1}{\ell}, (\alpha - \delta_1)V_1 + i\frac{(\delta_1+\delta_2)V_1}{\ell}\big]$ (resp. $\widehat{\mathbb{F}}_i = \big[(\alpha - \delta_1)\widehat{V}_1 + (i-1)\frac{(\delta_1+\delta_2)\widehat{V}_1}{\ell}, (\alpha - \delta_1)\widehat{V}_1 + i\frac{(\delta_1+\delta_2)\widehat{V}_1}{\ell}\big]$), for $i \in [\ell]$. Note that if $x \in \mathbb{F}_i$ and $\hat{x} \in \widehat{\mathbb{F}}_i$, then $x > \hat{x}, \forall i \in [\ell]$. Then

$$\begin{aligned}\max_{\boldsymbol{V}_2^t} P(\boldsymbol{V}_1^t, \theta, \Delta, t) &= \int_{\mathbb{F}} p_1(x - \alpha V_1) P(\theta - x, \Delta, t-1)\mathrm{d}x \\ &= \int_{\bigcup_{i=1}^{\ell}\mathbb{F}_i} \frac{1}{(\delta_1+\delta_2)V_1} P(\theta - x, \Delta, t-1)\mathrm{d}x \\ &= \sum_{i=1}^{\ell} \int_{\mathbb{F}_i} \frac{1}{(\delta_1+\delta_2)V_1} P(\theta - x, \Delta, t-1)\mathrm{d}x\end{aligned}$$

[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 [7].

and

$$\max_{\widehat{\boldsymbol{V}}_2^t} P(\widehat{\boldsymbol{V}}_1^t, \theta, \Delta, t) = \int_{\mathbb{F}} p_1(x - \alpha \widehat{V}_1) P(\theta - x, \Delta, t-1) \mathrm{d}x$$
$$= \int_{\bigcup_{i=1}^{\ell} \widehat{\mathbb{F}}_i} \frac{1}{(\delta_1 + \delta_2)\widehat{V}_1} P(\theta - x, \Delta, t-1) \mathrm{d}x$$
$$= \sum_{i=1}^{\ell} \int_{\widehat{\mathbb{F}}_i} \frac{1}{(\delta_1 + \delta_2)\widehat{V}_1} P(\theta - x, \Delta, t-1) \mathrm{d}x.$$

According to Lemma 8, $P(\theta - x, \Delta, t-1)$ is a non-decreasing function of x; therefore, for each element in the summation, we have

$$\int_{\mathbb{F}_i} \frac{1}{(\delta_1 + \delta_2)V_1} P(\theta - x, \Delta, t-1) \mathrm{d}x$$
$$\geqslant \frac{|\mathbb{F}_i| P(\theta - ((\alpha - \delta_1)V_1 + i\frac{(\delta_1+\delta_2)V_1}{\ell}), \Delta, t-1)}{(\delta_1 + \delta_2)V_1}$$
$$\geqslant \frac{|\widehat{\mathbb{F}}_i| P(\theta - ((\alpha - \delta_1)\widehat{V}_1 + (i-1)\frac{(\delta_1+\delta_2)\widehat{V}_1}{\ell}), \Delta, t-1)}{(\delta_1 + \delta_2)\widehat{V}_1}$$
$$\geqslant \int_{\widehat{\mathbb{F}}_i} \frac{1}{(\delta_1 + \delta_2)\widehat{V}_1} P(\theta - x, \Delta, t-1) \mathrm{d}x,$$

for all $i \in [\ell]$. This proves $\max_{\widehat{\boldsymbol{V}}_2^t} P(\widehat{\boldsymbol{V}}_1^t, \theta, \Delta, t) \leqslant \max_{\boldsymbol{V}_2^t} P(\boldsymbol{V}_1^t, \theta, \Delta, t)$.

Case (2): Suppose $\widetilde{V}_1 > V_1 = \frac{\theta+\Delta}{\alpha+\delta_2}$.

If the first voltage applied is $\widetilde{V}_1$, then the voltage of the cell after the first round of programming is uniformly distributed over $\widetilde{\mathbb{F}} = [(\alpha - \delta_1)\widetilde{V}_1, (\alpha + \delta_2)\widetilde{V}_1]$. Once the voltage is in $(\theta + \Delta, (\alpha + \delta_2)\widetilde{V}_1]$, the probability that after t rounds of programming the final cell level is within the interval $[\theta - \Delta, \theta + \Delta]$ is 0, since the cell level cannot be decreased in our model of flash cell programming.

Now, since $\widetilde{V}_1 > V_1$, we have

$$\max_{\widetilde{\boldsymbol{V}}_2^t} P(\widetilde{\boldsymbol{V}}_1^t, \theta, \Delta, t) = \int_{\widetilde{\mathbb{F}}} p_1(x - \alpha \widetilde{V}_1) P(\theta - x, \Delta, t-1) \mathrm{d}x$$
$$= \int_{(\alpha-\delta_1)\widetilde{V}_1}^{\theta+\Delta} \frac{1}{(\delta_1 + \delta_2)\widetilde{V}_1} P(\theta - x, \Delta, t-1) \mathrm{d}x$$
$$\leqslant \int_{(\alpha-\delta_1)\widetilde{V}_1}^{\theta+\Delta} \frac{1}{(\delta_1 + \delta_2)V_1} P(\theta - x, \Delta, t-1) \mathrm{d}x$$
$$\leqslant \int_{(\alpha-\delta_1)V_1}^{\theta+\Delta} \frac{1}{(\delta_1 + \delta_2)V_1} P(\theta - x, \Delta, t-1) \mathrm{d}x$$
$$= \max_{\boldsymbol{V}_2^t} P(\boldsymbol{V}_1^t, \theta, \Delta, t).$$

Noting that

$$\max_{\boldsymbol{V}_1^t} P(\boldsymbol{V}_1^t, \theta, \Delta, t) = \max_{V_1} \max_{\boldsymbol{V}_2^t} P(\boldsymbol{V}_1^t, \theta, \Delta, t),$$

we conclude that $P(\boldsymbol{V}_1^t, \theta, \Delta, t)$ is maximized when $V_1 = \frac{\theta+\Delta}{\alpha+\delta_2}$. ∎

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

Corollary 1. Algorithm 4 gives an optimal solution for the cell programming problem (P6).

Algorithm 4 Noisy Programming With Feedback

The voltage V_j on the j-th round of programming, where $1 \leqslant j \leqslant t$, is set as follows.

Let x_j denote the feedback representing the cell level before the j-th write, where, for $j = 1$, we set $x_1 = 0$.

Set $V_j = \frac{\theta - x_j + \Delta}{\alpha + \delta_2}$.

Proof: According to Theorem 6, if we need to reach the level θ, then the voltage applied on the first round is $\frac{\theta+\Delta}{\alpha+\delta_2}$. Thus, after the $(j-1)$-st round, if we know that the current cell level is x_j, then the voltage applied on the j-th round is $\frac{\theta - x_j + \Delta}{\alpha+\delta_2}$, which completes the proof. ∎

VII. Conclusion

Accurate and efficient cell programming is critical to the enhancement of flash memory functionality and storage capacity. Programming techniques must take into account the asymmetric nature of the write process, the manner in which discrete data values are represented within the range of cell levels, the presence or absence of noise, and the reduction in write latency that parallel programming can provide.

In this paper, we make the realistic assumption that cell levels are quantized to a discrete set of levels to represent digital data. The programming of a cell is considered to be successful if the programmed cell level is correctly quantized to the desired target level. For several scenarios, we present programming algorithms that, for a specified number of programming rounds, achieve optimality with respect to this figure of merit. Specifically, when cells have known hardness to charge injection and the programming process is noiseless, we derive an optimal parallel programming algorithm whose complexity is polynomial in the number of cells. We also modify the algorithm to take into account the presence of inter-cell interference from adjacent cells.

We also consider techniques for programming a single cell in the presence of noise. Assuming that no feedback on the cell level is available during the write process, we present a programming algorithm that, for a given number of programming rounds, maximizes the probability of attaining a cell level corresponding to the desired target level. We then address the situation where feedback is available and present an optimal strategy for adaptively choosing the programming voltages.

Appendix A

Proof of Lemma 2: We prove the lemma by induction.

For $t = 1$, it is equivalent to prove that there exists an optimal V_1 such that $V_1 \in \mathcal{T}$. So, suppose V_1^* is an optimal solution. If $V_1^* \in \mathcal{T}$, then the lemma holds for $t = 1$; if not, define $\delta_{\min}$ to be the smallest distance from V_1^* to an element of $\mathcal{T}$, i.e.,

$$\delta_{\min} = \min_{p \in \mathcal{T}} \{|V_1^* - p|\}.$$

If $\delta_{\min}$ is achieved by choosing an upper threshold point, set $\widehat{V}_1 = V_1^* + \delta_{\min}$; otherwise, set $\widehat{V}_1 = V_1^* - \delta_{\min}$. That is, $\widehat{V}_1$ is the closest threshold point to V_1^*. By the definition of

$\delta_{\min}$, any cell that can be quantized correctly using V_1^* can be quantized correctly using $\widehat{V}_1$; thus, $\widehat{V}_1$ is also an optimal solution. Meanwhile, $\widehat{V}_1 \in \mathcal{T}$. This proves that there always exists an optimal solution $V_1 \in \mathcal{T}$.

Suppose the lemma holds if the number of programming rounds is $t-1$. That is, for $t \geqslant 2$, assume there exists an invertible matrix $\mathbf{A} \in \{0,1\}^{(t-1)\times(t-1)}$, such that

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

where $\boldsymbol{V} \in \mathbb{R}_+^{t-1}$ is an optimal solution for (P3) and $\boldsymbol{p} \in \mathcal{T}^{(t-1)}$. We are going to prove by contradiction that the lemma holds if the number of programming rounds is t.

Suppose the opposite is true. Then, for any $\boldsymbol{p}' \in \mathcal{T}^t$, there does not exist an invertible matrix $\mathbf{A} \in \{0,1\}^{t\times t}$, such that

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

where $\boldsymbol{V}$ is an optimal solution for (P3). Let t' be the largest number, $0 \leqslant t' < t$, such that there exists a matrix $\mathbf{A}' \in \{0,1\}^{t'\times t}$ with full row rank, such that

$$\mathbf{A}'\boldsymbol{V}^* = \boldsymbol{p}',$$

where $\boldsymbol{V}^*$ is an optimal solution for (P3) and $\boldsymbol{p}' \in \mathcal{T}^{t'}$.

Let $\boldsymbol{V} \in \mathbb{R}_+^t$ satisfy $\mathbf{A}'\boldsymbol{V} = \boldsymbol{p}'$. Since $\text{rank}(\mathbf{A}') = t' < t$, the solution space for $\boldsymbol{V}$ is a nonempty polytope $\mathcal{P}$ consisting of the non-negative vectors in a t'-dimensional subspace. That is,

$$\mathcal{P} = \left\{\boldsymbol{V} \in \mathbb{R}_+^t | \mathbf{A}'\boldsymbol{V} = \boldsymbol{p}'\right\}.$$

(Note that if $t' = 0$, then $\mathcal{P}$ is the space of non-negative t-dimensional vectors.)

Claim 1. There exists a $\widehat{\boldsymbol{V}}$ in $\mathcal{P}$ such that $\widehat{\boldsymbol{V}}$ is on the boundary of $\mathcal{P}$, i.e., $\exists \widehat{\boldsymbol{V}} \in \mathcal{P}$ and $k \in [t]$, such that $\widehat{V}_k = 0$.

Proof of Lemma 2: Since $\mathcal{P}$ is a nontrivial polytope, there exists $\boldsymbol{X} \in \mathbb{R}_+^t$ in $\mathcal{P}$ such that $\boldsymbol{X} \neq \boldsymbol{V}^*$. If there exists $j \in [t]$ such that $X_j < V_j^*$, let

$$z_{\min} = \arg \min_{1 \leqslant j \leqslant t} \frac{X_j}{V_j^*}.$$

If there exists more than one index $j \in [t]$ that minimizes $\frac{X_j}{V_j^*}$, then $z_{\min}$ is chosen arbitrarily from among these indices. Let

$$y_{\min} = \min_{1 \leqslant j \leqslant t} \frac{X_j}{V_j^*} = \frac{X_{z_{\min}}}{V_{z_{\min}}^*} < 1,$$

and set

$$\widehat{\boldsymbol{V}} = \frac{\boldsymbol{X} - y_{\min}\boldsymbol{V}^*}{(1 - y_{\min})}.$$

Since $\widehat{\boldsymbol{V}}$ is a linear combination of $\boldsymbol{X}$ and $\boldsymbol{V}^*$, we have $\mathbf{A}'\widehat{\boldsymbol{V}} = \boldsymbol{p}'$. Then

$$\widehat{V}_{z_{\min}} = \frac{X_{z_{\min}} - y_{\min}V_{z_{\min}}^*}{(1 - y_{\min})} = \frac{X_{z_{\min}} - \frac{X_{z_{\min}}}{V_{z_{\min}}^*}V_{z_{\min}}^*}{(1 - y_{\min})} = 0,$$

and

$$\widehat{V}_j = \frac{X_j - y_{\min}V_j^*}{(1 - y_{\min})} \geqslant 0, \quad \forall j \in [t].$$

Therefore $\widehat{\boldsymbol{V}} \in \mathcal{P}$ and $\widehat{V}_{z_{\min}} = 0$.

If there does not exist $j \in [t]$ such that $X_j < V_j^*$, then there exists $j \in [t]$ such that $V_j^* < X_j$ since $\boldsymbol{X} \neq \boldsymbol{V}^*$. Following similar reasoning, we can prove that there exists $\widehat{\boldsymbol{V}} \in \mathcal{P}$ such that $\widehat{V}_k = 0$, for some $k \in [t]$. ■

Now, there are two different cases to consider for the $\widehat{\boldsymbol{V}}$ of Claim 1.

Case (1): $\widehat{\boldsymbol{V}}$ is an optimal solution of (P3).

Claim 2. If $\widehat{\boldsymbol{V}}$ is optimal, then all of the n cells can be quantized correctly using $t-1$ rounds of programming.

Proof: Suppose the opposite is true, and there exists c_i with quantization interval $[v_i, u_i]$ that is not quantized correctly by $\widehat{\boldsymbol{V}}$. Set $\widetilde{V}_j = \widehat{V}_j$ for all j such that $1 \leqslant j \neq k \leqslant t$ and set $\widetilde{V}_k = v_i$. Then the number of cells that are quantized correctly by $\widetilde{\boldsymbol{V}}$ is larger than $\widehat{\boldsymbol{V}}$, contradicting the assumption that $\widehat{\boldsymbol{V}}$ is optimal. ■

By the induction assumption, if the number of programming rounds is $t-1$, there exists an invertible matrix $\mathbf{A} \in \{0,1\}^{(t-1)\times(t-1)}$, such that

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

where $\boldsymbol{V} \in \mathbb{R}_+^{(t-1)}$ is an optimal solution and $\boldsymbol{p} \in \mathcal{T}^{(t-1)}$. Note that in this case, according to Claim 2, all of the n cells can be quantized correctly. We form another invertible matrix $\widetilde{\mathbf{A}} \in \{0,1\}^{t\times t}$ by adding one column and one row to $\mathbf{A}$, where all added entries are 0 except that $\widetilde{a}_{t,t} = 1$. Let $\widetilde{\boldsymbol{V}} = (V_1, \ldots, V_{t-1}, p_t)^T \in \mathbb{R}_+^t$ where $p_t \in \mathcal{T}$ is any threshold point. Let $\widetilde{\boldsymbol{p}} = (\boldsymbol{p}^T, p_t)^T = (p_1, \ldots, p_{t-1}, p_t)^T$ be a threshold-point vector. Then we have

$$\widetilde{\mathbf{A}}\widetilde{\boldsymbol{V}} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0^T} & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{V} \\ p_t \end{bmatrix} = \begin{bmatrix} \boldsymbol{p} \\ p_t \end{bmatrix} \in \mathcal{T}^t.$$

and $\widetilde{\boldsymbol{V}}$ is optimal since all of the n cells are quantized correctly. The existence of $\widetilde{\mathbf{A}}$, $\widetilde{\boldsymbol{V}}$ and $\widetilde{\boldsymbol{p}}$ contradicts the assumption that t' is the maximum row rank.

Case (2): $\widehat{\boldsymbol{V}}$ is not an optimal solution of (P3).

Note that because $\boldsymbol{V}^*$ is optimal while $\widehat{\boldsymbol{V}}$ is not optimal, the following claims hold.

Claim 3. There exists $\boldsymbol{b} \in \{0,1\}^t$ such that $\boldsymbol{b}^T\boldsymbol{V}^* \in [v_i, u_i]$ and $\boldsymbol{b}^T\widehat{\boldsymbol{V}} \notin [v_i, u_i]$, for some $i \in [n]$.

Proof: Since $\boldsymbol{V}^*$ is optimal while $\widehat{\boldsymbol{V}}$ is not optimal, at least one cell can be quantized correctly by $\boldsymbol{V}^*$ but not by $\widehat{\boldsymbol{V}}$. Suppose the cell is c_i. Then there exists $\boldsymbol{b} \in \{0,1\}^t$ such that $\boldsymbol{b}^T\boldsymbol{V}^* \in [v_i, u_i]$ and $\boldsymbol{b}^T\widehat{\boldsymbol{V}} \notin [v_i, u_i]$. ■

Claim 4. Every $\boldsymbol{b} \in \{0,1\}^t$ satisfying the property in Claim 3 is linearly independent with respect to the set of row vectors of $\mathbf{A}'$.

Proof: Suppose the opposite is true. That is, $\boldsymbol{b}^T = \boldsymbol{x}\mathbf{A}'$, for some $\boldsymbol{x} \in \mathbb{R}^{t'}$. Then

$$\boldsymbol{b}^T\boldsymbol{V}^* = \boldsymbol{x}\mathbf{A}'\boldsymbol{V}^* = \boldsymbol{x}\boldsymbol{p}' = \boldsymbol{x}\mathbf{A}'\widehat{\boldsymbol{V}} = \boldsymbol{b}^T\widehat{\boldsymbol{V}}.$$

This contradicts the fact that $\boldsymbol{b}^T\boldsymbol{V}^* \neq \boldsymbol{b}^T\widehat{\boldsymbol{V}}$. ■

Suppose the number of triplets $(\boldsymbol{b}, v_i, u_i)$ for $\boldsymbol{b} \in \{0,1\}^t$ and $i \in [n]$ in Claim 3 is K. We list all such triplets and label them by $(\boldsymbol{b}_k, w_k, y_k)$, $k \in [K]$. Note that one and only one of w_k and y_k is between $\boldsymbol{b}_k^T \boldsymbol{V}^*$ and $\boldsymbol{b}_k^T \widehat{\boldsymbol{V}}$. Without loss of generality, we assume that w_k is between $\boldsymbol{b}_k^T \boldsymbol{V}^*$ and $\boldsymbol{b}_k^T \widehat{\boldsymbol{V}}$. In particular, $\boldsymbol{b}_k^T \widehat{\boldsymbol{V}} \leqslant w_k \leqslant \boldsymbol{b}_k^T \boldsymbol{V}^*$. Define

$$\delta_{\min} = \min_{1 \leqslant k \leqslant K} \frac{\boldsymbol{b}_k^T \boldsymbol{V}^* - w_k}{\boldsymbol{b}_k^T \boldsymbol{V}^* - \boldsymbol{b}_k^T \widehat{\boldsymbol{V}}} \in [0,1]$$

and

$$k_{\min} = \arg \min_{1 \leqslant k \leqslant K} \frac{\boldsymbol{b}_k^T \boldsymbol{V}^* - w_k}{\boldsymbol{b}_k^T \boldsymbol{V}^* - \boldsymbol{b}_k^T \widehat{\boldsymbol{V}}}.$$

Consider the convex combination of $\boldsymbol{V}^*$ and $\widehat{\boldsymbol{V}}$ given by

$$\widetilde{\boldsymbol{V}} = \boldsymbol{V}^* - \delta_{\min}(\boldsymbol{V}^* - \widehat{\boldsymbol{V}}) = (1 - \delta_{\min})\boldsymbol{V}^* + \delta_{\min}\widehat{\boldsymbol{V}}.$$

Claim 5. $\widetilde{\boldsymbol{V}}$ is an optimal solution of (P3).

Proof: Suppose c_i can be programmed into its quantization interval $[v_i, u_i], i \in [n]$ by $\boldsymbol{V}^*$ but not by $\widehat{\boldsymbol{V}}$. According to Claim 3, let $\boldsymbol{b} \in \{0,1\}^t$ be a vector such that $\boldsymbol{b}^T \boldsymbol{V}^* \in [v_i, u_i]$, but $\boldsymbol{b}^T \widehat{\boldsymbol{V}} \notin [v_i, u_i]$. We will prove that $v_i \leqslant \boldsymbol{b}^T \widetilde{\boldsymbol{V}} \leqslant u_i$.

Without loss of generality, we assume $\boldsymbol{b}^T \widehat{\boldsymbol{V}} < v_i \leqslant \boldsymbol{b}^T \boldsymbol{V}^* \leqslant u_i$. Then

$$\begin{aligned} u_i \geqslant \boldsymbol{b}^T \boldsymbol{V}^* &\overset{①}{\geqslant} \boldsymbol{b}^T \widetilde{\boldsymbol{V}} = \boldsymbol{b}^T \left(\boldsymbol{V}^* - \delta_{\min}(\boldsymbol{V}^* - \widehat{\boldsymbol{V}})\right) \\ &\geqslant \boldsymbol{b}^T \boldsymbol{V}^* - \boldsymbol{b}^T \frac{\boldsymbol{b}^T \boldsymbol{V}^* - v_i}{\boldsymbol{b}^T \boldsymbol{V}^* - \boldsymbol{b}^T \widehat{\boldsymbol{V}}}(\boldsymbol{V}^* - \widehat{\boldsymbol{V}}) \\ &= \boldsymbol{b}^T \boldsymbol{V}^* - \boldsymbol{b}^T (\boldsymbol{V}^* - \widehat{\boldsymbol{V}}) \frac{\boldsymbol{b}^T \boldsymbol{V}^* - v_i}{\boldsymbol{b}^T (\boldsymbol{V}^* - \widehat{\boldsymbol{V}})} \\ &= v_i, \end{aligned}$$

where ① follows from the fact that

$$\begin{aligned} \boldsymbol{b}^T \widetilde{\boldsymbol{V}} &= \boldsymbol{b}^T \left(\boldsymbol{V}^* - \delta_{\min}(\boldsymbol{V}^* - \widehat{\boldsymbol{V}})\right) \\ &= \boldsymbol{b}^T \boldsymbol{V}^* - \delta_{\min} \boldsymbol{b}^T (\boldsymbol{V}^* - \widehat{\boldsymbol{V}}) \\ &\leqslant \boldsymbol{b}^T \boldsymbol{V}^*. \end{aligned}$$

If c_i can be programmed into its quantization interval $[v_i, u_i], i \in [n]$ by $\boldsymbol{V}^*$ and $\widehat{\boldsymbol{V}}$, then it can be programmed into $[v_i, u_i]$ by $\widetilde{\boldsymbol{V}}$ as well, since $\widetilde{\boldsymbol{V}}$ is a convex combination of $\boldsymbol{V}^*$ and $\widehat{\boldsymbol{V}}$. Thus, each cell that can be quantized correctly by $\boldsymbol{V}^*$ can also be quantized correctly by $\widetilde{\boldsymbol{V}}$, implying that $\widetilde{\boldsymbol{V}}$ is optimal. ∎

Let

$$\widetilde{\mathbf{A}} = \begin{bmatrix} \mathbf{A}' \\ \boldsymbol{b}_{k_{\min}}^T \end{bmatrix}.$$

Claim 6. $\widetilde{\mathbf{A}}$ has row rank $t' + 1$ and $\widetilde{\mathbf{A}}\widetilde{\boldsymbol{V}} \in \mathcal{T}^{(t'+1)}$.

Proof: According to Claim 4, each $\boldsymbol{b}^T$ is linearly independent of the set of row vectors of $\mathbf{A}'$, implying that $\operatorname{rank}(\widetilde{\mathbf{A}}) = t' + 1$.

Consider

$$\widetilde{\mathbf{A}}\widetilde{\boldsymbol{V}} = \begin{bmatrix} \mathbf{A}' \\ \boldsymbol{b}_{k_{\min}}^T \end{bmatrix} \widetilde{\boldsymbol{V}} = \begin{bmatrix} \mathbf{A}'\widetilde{\boldsymbol{V}} \\ \boldsymbol{b}_{k_{\min}}^T \widetilde{\boldsymbol{V}} \end{bmatrix} \overset{\text{def}}{=} \widetilde{\boldsymbol{p}},$$

Since $\widetilde{\boldsymbol{V}}$ is a convex combination of $\boldsymbol{V}^*$ and $\widehat{\boldsymbol{V}}$, it is in the polytope $\mathcal{P}$, thus $\mathbf{A}'\widetilde{\boldsymbol{V}} = \boldsymbol{p}' \in \mathcal{T}^{t'}$.

Now,

$$\begin{aligned} \boldsymbol{b}_{k_{\min}}^T \widetilde{\boldsymbol{V}} &= \boldsymbol{b}_{k_{\min}}^T \left(\boldsymbol{V}^* - \delta_{\min}(\boldsymbol{V}^* - \widehat{\boldsymbol{V}})\right) \\ &= \boldsymbol{b}_{k_{\min}}^T \left(\boldsymbol{V}^* - \frac{\boldsymbol{b}_{k_{\min}}^T \boldsymbol{V}^* - w_{k_{\min}}}{\boldsymbol{b}_{k_{\min}}^T \boldsymbol{V}^* - \boldsymbol{b}_{k_{\min}}^T \widehat{\boldsymbol{V}}}(\boldsymbol{V}^* - \widehat{\boldsymbol{V}})\right) \\ &= \boldsymbol{b}_{k_{\min}}^T \boldsymbol{V}^* - \boldsymbol{b}_{k_{\min}}^T (\boldsymbol{V}^* - \widehat{\boldsymbol{V}}) \frac{\boldsymbol{b}_{k_{\min}}^T \boldsymbol{V}^* - w_{k_{\min}}}{\boldsymbol{b}_{k_{\min}}^T (\boldsymbol{V}^* - \widehat{\boldsymbol{V}})} \\ &= w_{k_{\min}} \in \mathcal{T}. \end{aligned}$$

Therefore, $\widetilde{\boldsymbol{p}} \in \mathcal{T}^{(t'+1)}$. ∎

For both Case (1) and Case (2), the existence of $\widetilde{\mathbf{A}}$, $\widetilde{\boldsymbol{V}}$ and $\widetilde{\boldsymbol{p}}$ contradicts the assumption that t' is the maximum row rank of a matrix $\mathbf{A}$ such that $\mathbf{A}\boldsymbol{V}^* = \boldsymbol{p}'$. Therefore, there exists an invertible matrix $\mathbf{A} \in \{0,1\}^{t \times t}$ such that

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

where $\boldsymbol{V} \in \mathbb{R}_+^t$ is an optimal solution for (P3) and $\boldsymbol{p} \in \mathcal{T}^t$ is a threshold-point vector. ∎

Proof of Lemma 4: The proof is based on induction and is very similar to the proof of Lemma 2. Therefore, we prove the initial step of the induction and omit the remaining details.

For $t = 1$, we prove that there exist a threshold-point $p_i \in \mathcal{T}$ of the cell i and a real number

$$\begin{aligned} a \in \{&0, 1, \beta_{i-1,i}, \beta_{i+1,i}, 1 + \beta_{i-1,i}, 1 + \beta_{i+1,i}, \beta_{i-1,i} \\ &+ \beta_{i+1,i}, 1 + \beta_{i-1,i} + \beta_{i+1,i}\} \\ = &\bigcup_{\boldsymbol{b} \in \{0,1\}^3} \{(\beta_{i-1,i}, 1, \beta_{i+1,i}) \cdot \boldsymbol{b}\} \end{aligned}$$

such that

$$aV = p_i,$$

where $V \geqslant 0$ is optimal.

Suppose V^* is optimal. If $\exists a \in \{0, 1, \beta_{i-1,i}, \beta_{i+1,i}, 1 + \beta_{i-1,i}, 1 + \beta_{i+1,i}, \beta_{i-1,i} + \beta_{i+1,i}, 1 + \beta_{i-1,i} + \beta_{i+1,i}\}$ and $p_i \in \mathcal{T}$ such that

$$aV^* = p_i,$$

then the statement holds for $t = 1$. Otherwise, let

$$\delta = \min_{i \in [n], \boldsymbol{b} \in \{0,1\}^3} \left(V^*(\beta_{i-1,i}, 1, \beta_{i+1,i}) \cdot \boldsymbol{b} - p_i\right)^+,$$

where, for $x \in \mathbb{R}$, $x^+ = x$ if $x \geqslant 0$ and $x = +\infty$ if $x < 0$. Suppose the minimum is achieved for $i = i_{\min} \in [n]$ and $\boldsymbol{b} = \boldsymbol{b}_{\min}$. That is,

$$\delta = (\beta_{i_{\min}-1,i}, 1, \beta_{i_{\min}+1,i}) \cdot \boldsymbol{b}_{\min} V^* - p_{i_{\min}}.$$

Let

$$\widehat{V} = \frac{p_{i_{\min}}}{p_{i_{\min}} + \delta} V^*.$$

Then, by setting $a = (\beta_{i_{\min}-1,i}, 1, \beta_{i_{\min}+1,i}) \cdot \boldsymbol{b}_{\min}$, we have

$$\begin{aligned} a\widehat{V} &= (\beta_{i_{\min}-1,i}, 1, \beta_{i_{\min}+1,i}) \cdot \boldsymbol{b}_{\min} \widehat{V} \\ &= (\beta_{i_{\min}-1,i}, 1, \beta_{i_{\min}+1,i}) \cdot \boldsymbol{b}_{\min} \frac{p_{i_{\min}}}{p_{i_{\min}} + \delta} V^* \\ &= p_{i_{\min}}. \end{aligned}$$

In addition, the number of cells quantized correctly does not decrease in going from V^* to $\widehat{V}$ since, by definition, $\widehat{V} \leqslant V^*$

is chosen such that if a cell is quantized correctly by the voltage V^*, the cell can be also quantized correctly by $\widehat{V}$. This completes the proof of the case where $t = 1$. ■

APPENDIX B

Proof of Lemma 6: We first prove the following claim as it is used to establish the inequalities in the proof of Lemma 6.

Claim 7. Let c_1, c_2 and $\delta_1, \delta_2 > 0$ be real numbers. Let

$$p_1 = \frac{1}{\sqrt{2\pi}} \int_{c_1-\delta_1}^{c_1+\delta_1} e^{-u^2/2} \mathrm{d}u \text{ and } p_2 = \frac{1}{\sqrt{2\pi}} \int_{c_2-\delta_2}^{c_2+\delta_2} e^{-u^2/2} \mathrm{d}u.$$

Then the following three statements hold:

1) If $c_1 = c_2$ and $\delta_1 > \delta_2$, then $p_1 > p_2$.
2) If $\delta_1 = \delta_2$ and $|c_1| < |c_2|$, then $p_1 > p_2$.
3) If $|c_1| \leqslant |c_2|$ and $\delta_1 > \delta_2$, then $p_1 > p_2$.

Proof:

1) If $\delta_1 > \delta_2$, then $[c_1-\delta_1, c_1+\delta_1] \supset [c_2-\delta_2, c_2+\delta_2]$; thus $p_1 > p_2$ since the integrand is strictly positive on $\mathbb{R}$.
2) Let $\delta_1 = \delta_2 = \delta$. We prove the result for the case where $0 \leqslant c_1 < c_2$ and $[c_1 - \delta, c_1 + \delta] \bigcap [c_2 - \delta, c_2 + \delta] = \emptyset$. Other cases can be reduced to this case by subtracting the integration over the intersection of the intervals. In the case considered, $\xi_1 \leqslant \xi_2, \forall \xi_1 \in [c_1 - \delta, c_1 + \delta]$, $\xi_2 \in [c_2 - \delta, c_2 + \delta]$.
   Note that $f(u) = e^{-u^2/2}$ is symmetric with respect to $u = 0$ and $f(u)$ is a continuous strictly decreasing function for $u \geqslant 0$. Therefore, there exists $\xi_i \in [c_i - \delta, c_i + \delta]$ such that
$$p_i = \int_{c_i-\delta}^{c_i+\delta} e^{-u^2/2} \mathrm{d}u = 2\delta e^{-\xi_i^2/2}, i = 1, 2.$$
   It follows that $p_1 > p_2$ since $\xi_1 < \xi_2$.
3) Let $p_3 = \int_{c_3-\delta_3}^{c_3+\delta_3} e^{-u^2/2} \mathrm{d}u$, where $c_3 = c_1$ and $\delta_3 = \delta_2$. Then from 1) we have $p_1 \geqslant p_3$, and from 2) we have $p_3 > p_2$. Therefore, $p_1 > p_2$.

■

Now we proceed to the proof of Lemma 6. Suppose the opposite is true; that is, for some i and j in $[t]$, $V_i^* \neq V_j^*$. Consider another vector $\hat{\boldsymbol{V}} = (\hat{V}_1, \ldots, \hat{V}_t)$, where $\hat{V}_k = \overline{V} \stackrel{\text{def}}{=} \sqrt{\frac{\sum_{j=1}^t V_j^{*2}}{t}}, \forall k \in [t]$. Then $\sqrt{\sum_{j=1}^t \hat{V}_j^2} = \sqrt{\sum_{j=1}^t V_j^{*2}}$. Furthermore, we have

$$\sum_{j=1}^t \hat{V}_j = t\overline{V} = t\sqrt{\frac{\sum_{j=1}^t V_j^{*2}}{t}} > \sum_{j=1}^t V_j^*.$$

This is a special case of the Cauchy-Schwartz Inequality, and the inequality is strict due to the assumption that $V_i^* \neq V_j^*$ for some $i, j \in [t]$. Therefore, $\theta - \alpha \sum_{j=1}^t \hat{V}_j < \theta - \alpha \sum_{j=1}^t V_j^*$.

We want to show that $\boldsymbol{V}^*$ is not optimal, in particular, $g(\hat{\boldsymbol{V}}) > g(\boldsymbol{V}^*)$. We consider two cases.

Case (1): Suppose $\theta - \alpha \sum_{j=1}^t \hat{V}_j = \theta - \alpha t \overline{V} \geqslant 0$.

Let

$$p_1 = g(\boldsymbol{V}^*) = \frac{1}{\sqrt{2\pi}} \int_{c(\boldsymbol{V}^*)-\delta(\boldsymbol{V}^*)}^{c(\boldsymbol{V}^*)+\delta(\boldsymbol{V}^*)} e^{-\frac{u^2}{2}} \mathrm{d}u$$

and

$$p_2 = g(\hat{\boldsymbol{V}}) = \frac{1}{\sqrt{2\pi}} \int_{c(\hat{\boldsymbol{V}})-\delta(\hat{\boldsymbol{V}})}^{c(\hat{\boldsymbol{V}})+\delta(\hat{\boldsymbol{V}})} e^{-\frac{u^2}{2}} \mathrm{d}u,$$

where $c(\boldsymbol{V})$ and $\delta(\boldsymbol{V})$ are as defined in Lemma 5.

Recall that $c(\hat{\boldsymbol{V}}) = \frac{\theta - \alpha \sum \hat{V}_j}{\sigma\sqrt{\sum \hat{V}_j^2}}$ and $c(\boldsymbol{V}^*) = \frac{\theta - \alpha \sum V_j^{*2}}{\sigma\sqrt{\sum V_j^{*2}}}$, and let $s = \sqrt{\sum_{j=1}^t \hat{V}_j^2} = \sqrt{\sum_{j=1}^t V_j^{*2}}$. Then we have

$$\begin{aligned} 0 \leqslant c(\hat{\boldsymbol{V}}) &= \frac{\theta - \alpha \sum_{j=1}^t \hat{V}_j}{\sigma\sqrt{\sum_{j=1}^t \hat{V}_j^2}} \\ &= \frac{\theta - \alpha \sum_{j=1}^t \hat{V}_j}{s\sigma} \\ &< \frac{\theta - \alpha \sum_{j=1}^t V_j^{*2}}{s\sigma} \\ &= \frac{\theta - \alpha \sum_{j=1}^t V_j^{*2}}{\sigma\sqrt{\sum_{j=1}^t V_j^{*2}}} \\ &= c(\boldsymbol{V}^*). \end{aligned}$$

Recall that $\delta(\hat{\boldsymbol{V}}) = \frac{\Delta}{\sigma\sqrt{\sum \hat{V}_j^2}}$ and $\delta(\boldsymbol{V}^*) = \frac{\Delta}{\sigma\sqrt{\sum V_j^{*2}}}$. Then $\delta(\hat{\boldsymbol{V}}) = \delta(\boldsymbol{V}^*)$. Set $c_1 = c(\hat{\boldsymbol{V}}), c_2 = c(\boldsymbol{V}^*), \delta_1 = \delta(\hat{\boldsymbol{V}})$, and $\delta_2 = \delta(\boldsymbol{V}^*)$. According to 2) in Claim 7, we conclude that $p_1 < p_2$. Therefore, $g(\boldsymbol{V}^*) < g(\hat{\boldsymbol{V}})$, implying $\boldsymbol{V}^*$ is not optimal.

Case (2): Suppose $\theta - \alpha \sum_{j=1}^t \hat{V}_j = \theta - \alpha t \overline{V} < 0$.

Consider another vector $\tilde{\boldsymbol{V}} = (\tilde{V}_1, \ldots, \tilde{V}_t)$ where $\tilde{V}_j = \frac{\theta}{\alpha t}$, $\forall j \in [t]$. Then $\theta - \alpha \sum_{j=1}^t \tilde{V}_j = 0$ and we have

$$\sqrt{\sum_{j=1}^t \tilde{V}_j^2} = \sqrt{\left(\frac{\theta}{\alpha t}\right)^2 t} < \sqrt{\overline{V}^2 t} = \sqrt{\sum_{j=1}^t \hat{V}_j^2} = \sqrt{\sum_{j=1}^t V_j^{*2}}.$$

It is easy to see that

$$|c(\tilde{\boldsymbol{V}})| = \left|\frac{\theta - \alpha \sum_{j=1}^t \tilde{V}_j}{\sigma\sqrt{\sum_{j=1}^t \tilde{V}_j^2}}\right| = 0 \leqslant |c(\boldsymbol{V}^*)|,$$

and

$$\delta(\tilde{\boldsymbol{V}}) = \frac{\Delta}{\sigma\sqrt{\sum_{j=1}^t \tilde{V}_j^2}} > \frac{\Delta}{\sigma\sqrt{\sum_{j=1}^t V_j^{*2}}} = \delta(\boldsymbol{V}^*).$$

Let $p_1 = g(\tilde{\boldsymbol{V}})$ and $p_2 = g(\boldsymbol{V}^*)$. Set $c_1 = c(\tilde{\boldsymbol{V}}), c_2 = c(\boldsymbol{V}^*), \delta_1 = \delta(\tilde{\boldsymbol{V}})$, and $\delta_2 = \delta(\boldsymbol{V}^*)$. According to 3) in Claim 7, we conclude that $p_1 > p_2$. Therefore, $g(\tilde{\boldsymbol{V}}) > g(\boldsymbol{V}^*)$, implying that $\boldsymbol{V}^*$ is not optimal.

These contradictions of the optimality of $\boldsymbol{V}^*$ arose from the assumption that $V_i^* \neq V_j^*$ for some $i, j \in [t]$. Therefore, we conclude that $V_i^* = V_j^*, \forall i, j \in [t]$. ■

APPENDIX C

Proof of Lemma 7: Recall that when applying voltage V_j, the cell-level increment is uniformly distributed in $[(\alpha-\delta_1)V_j, (\alpha+\delta_2)V_j]$. Consider the following two cases.

Case (1): Suppose $\frac{\theta-\Delta}{\theta+\Delta} < \frac{\alpha-\delta_1}{\alpha+\delta_2}$. Setting $V_1 = \frac{\theta+\Delta}{\alpha+\delta_2}$, we have

$$\begin{aligned}
P(V_1,\theta,\Delta,1) &= \int_{\theta-\Delta}^{\theta+\Delta} p_1(x-\alpha V_1)\mathrm{d}x \\
&= \int_{\theta-\Delta}^{\theta+\Delta} \frac{1}{(\delta_1+\delta_2)V_1} I_{x-\alpha V_1\in[-\delta_1 V_1,\delta_2 V_1]}\mathrm{d}x \\
&= \int_{\theta-\Delta}^{\theta+\Delta} \frac{1}{(\delta_1+\delta_2)V_1} I_{x\in[(\alpha-\delta_1)V_1,(\alpha+\delta_2)V_1]}\mathrm{d}x \\
&= \int_{(\alpha-\delta_1)V_1}^{(\alpha+\delta_2)V_1} \frac{1}{(\delta_1+\delta_2)V_1}\mathrm{d}x \\
&= 1.
\end{aligned}$$

Therefore,

$$1 \geqslant P(\theta,\Delta,1) = \max_{V_1} P(V_1,\theta,\Delta,1) \geqslant 1.$$

Case (2): Suppose $\frac{\theta-\Delta}{\theta+\Delta} \geqslant \frac{\alpha-\delta_1}{\alpha+\delta_2}$. Then

$$\begin{aligned}
P(V_1,\theta,\Delta,1) &= \int_{\theta-\Delta}^{\theta+\Delta} p_1(x-\alpha V_1)\mathrm{d}x \\
&= \int_{\theta-\Delta}^{\theta+\Delta} \frac{1}{(\delta_1+\delta_2)V_1} I_{x-\alpha V_1\in[-\delta_1 V_1,\delta_2 V_1]}\mathrm{d}x \\
&= \int_{\mathbb{R}} \frac{1}{(\delta_1+\delta_2)V_1} I_{x\in[(\alpha-\delta_1)V_1,(\alpha+\delta_2)V_1]\bigcap[\theta-\Delta,\theta+\Delta]}\mathrm{d}x.
\end{aligned}$$

There are two possibilities to consider.

1) If $V_1 \leqslant \frac{\theta+\Delta}{\alpha+\delta_2}$, then $(\alpha+\delta_2)V_1 \leqslant \theta+\Delta$ and $(\alpha-\delta_1)V_1 \leqslant \theta-\Delta$. Therefore

$$\begin{aligned}
&P(V_1,\theta,\Delta,1) \\
&\quad= \int_{\mathbb{R}} \frac{I_{x\in[(\alpha-\delta_1)V_1,(\alpha+\delta_2)V_1]\bigcap[\theta-\Delta,\theta+\Delta]}}{(\delta_1+\delta_2)V_1}\mathrm{d}x \\
&\quad= \int_{\theta-\Delta}^{(\alpha+\delta_2)V_1} \frac{1}{(\delta_1+\delta_2)V_1}\mathrm{d}x \\
&\quad= \frac{\alpha+\delta_2}{\delta_1+\delta_2} - \frac{\theta-\Delta}{(\delta_1+\delta_2)V_1} \\
&\quad\leqslant \frac{\alpha+\delta_2}{\delta_1+\delta_2} - \frac{\theta-\Delta}{(\delta_1+\delta_2)\frac{\theta+\Delta}{\alpha+\delta_2}} \\
&\quad= \left(\frac{\alpha+\delta_2}{\delta_1+\delta_2}\right)\left(\frac{2\Delta}{\theta+\Delta}\right),
\end{aligned}$$

where equality holds if $V_1 = \frac{\theta+\Delta}{\alpha+\delta_2}$.

2) If $V_1 > \frac{\theta+\Delta}{\alpha+\delta_2}$, then $(\alpha+\delta_2)V_1 > \theta+\Delta$. Therefore

$$\begin{aligned}
&P(V_1,\theta,\Delta,1) \\
&\quad= \int_{\mathbb{R}} \frac{I_{x\in[(\alpha-\delta_1)V_1,(\alpha+\delta_2)V_1]\bigcap[\theta-\Delta,\theta+\Delta]}}{(\delta_1+\delta_2)V_1}\mathrm{d}x \\
&\quad\leqslant \int_{\theta-\Delta}^{\theta+\Delta} \frac{1}{(\delta_1+\delta_2)V_1}\mathrm{d}x \\
&\quad= \frac{2\Delta}{(\delta_1+\delta_2)V_1} \\
&\quad< \frac{2\Delta}{(\delta_1+\delta_2)\frac{\theta+\Delta}{\alpha+\delta_2}} \\
&\quad= P\left(\frac{\theta+\Delta}{\alpha+\delta_2},\theta,\Delta,1\right)
\end{aligned}$$

It can be seen that under both circumstances, $P(V_1,\theta,\Delta,1)$ is maximized when $V_1 = \frac{\theta+\Delta}{\alpha+\delta_2}$. Consequently,

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

■

Proof of Lemma 8: We first prove the following two claims as they serve as the basis for the proof of Lemma 8.

Claim 8. For $\beta \neq 0$, $P(\beta\theta,\beta\Delta,t) = P(\theta,\Delta,t)$.

Proof: We proceed by induction. For $t=1$, we have

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

For notational convenience, let $a \bigwedge b \overset{\text{def}}{=} \min\{a,b\}$. Now, suppose that $P(\beta\theta,\beta\Delta,t-1) = P(\theta,\Delta,t-1)$. Then

$$\begin{aligned}
&P(\beta\theta,\beta\Delta,t) \\
&= \max_{V_1}\int_{\mathbb{R}_+} p_1(x-\alpha V_1)P(\beta\theta-x,\beta\Delta,t-1)\mathrm{d}x \\
&\overset{(1)}{=} \max_{V_1}\int_{\mathbb{R}_+} \frac{I_{x\in[(\alpha-\delta_1)V_1,((\alpha+\delta_2)V_1)\bigwedge\beta\theta]}}{(\delta_1+\delta_2)V_1} \\
&\qquad\qquad \cdot P(\beta\theta-x,\beta\Delta,t-1)\mathrm{d}x \\
&\overset{(2)}{=} \max_{V_1}\int_{\mathbb{R}_+} \frac{I_{\beta y\in[(\alpha-\delta_1)V_1,((\alpha+\delta_2)V_1)\bigwedge\beta\theta]}}{(\delta_1+\delta_2)V_1} \\
&\qquad\qquad \cdot P(\beta\theta-\beta y,\beta\Delta,t-1)\mathrm{d}\beta y \\
&\overset{(3)}{=} \max_{V_1}\int_{\mathbb{R}_+} \frac{I_{\beta y\in[(\alpha-\delta_1)V_1,((\alpha+\delta_2)V_1)\bigwedge\beta\theta]}}{(\delta_1+\delta_2)V_1} \\
&\qquad\qquad \cdot P(\theta-y,\Delta,t-1)\mathrm{d}\beta y \\
&\overset{(4)}{=} \max_{V_1}\int_{\mathbb{R}_+} \frac{I_{y\in[(\alpha-\delta_1)\frac{V_1}{\beta},\left((\alpha+\delta_2)\frac{V_1}{\beta}\right)\bigwedge\theta]}}{(\delta_1+\delta_2)\frac{V_1}{\beta}} P(\theta-y,\Delta,t-1)\mathrm{d}y \\
&\overset{(5)}{=} \max_{V_1'}\int_{\mathbb{R}_+} \frac{I_{y\in[(\alpha-\delta_1)V_1',((\alpha+\delta_2)V_1')\bigwedge\theta]}}{(\delta_1+\delta_2)V_1'} P(\theta-y,\Delta,t-1)\mathrm{d}y \\
&\overset{(6)}{=} \max_{V_1'}\int_{\mathbb{R}_+} p_1(y-\alpha V_1')P(\theta-y,\Delta,t-1)\mathrm{d}y \\
&= P(\theta,\Delta,t).
\end{aligned}$$

Equation (1) follows from the definition of $p_1(x)$. Equation (2) follows from the change of variables $x = \beta y$. Equation (3) follows from the induction hypothesis that $P(\beta\theta,\beta\Delta,t-1) = P(\theta,\Delta,t-1)$. Equation (4) follows from the linearity of the indicator function and the min operator. That is $I_{\beta y\in[a,b]} = I_{y\in[a/\beta,b/\beta]}$, and $\frac{1}{y}\min\{a,b\} = \min\{\frac{a}{y},\frac{b}{y}\}$. Equation (5) follows from the change of variables $V_1' = \frac{V_1}{\beta}$. Equation (6) holds for the same reason as Equation (1). ■

Claim 9. For $\beta > 1$, $P(\beta\theta, \beta\Delta, t) \geqslant P(\beta\theta, \Delta, t)$.

Proof: Since the interval $[\beta\theta - \Delta, \beta\theta + \Delta] \subseteq [\beta\theta - \beta\Delta, \beta\theta + \beta\Delta]$ for $\beta > 1$, the event "the cell level is in $[\beta\theta - \Delta, \beta\theta + \Delta]$ after t rounds of programming" is included in the event "the cell level is in $[\beta\theta - \beta\Delta, \beta\theta + \beta\Delta]$ after t rounds of programming." Thus, $P(\beta\theta, \beta\Delta, t) \geqslant P(\beta\theta, \Delta, t)$. ■

It follows from Claim 8 and Claim 9 that, for $\beta > 1$,

$$P(\theta, \Delta, t) = P(\beta\theta, \beta\Delta, t) \geqslant P(\beta\theta, \Delta, t),$$

which proves that $P(\theta, \Delta, t)$ is non-increasing in θ. ■

ACKNOWLEDGMENT

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

REFERENCES

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

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

[3] P. Cappelletti, C. Golla, P. Olivo, and E. Zanoni, *Flash Memories*, 1st ed. Norwell, MA, USA: Kluwer, 1999.

[4] 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.*, vol. 57, no. 10, pp. 2718–2728, Oct. 2010.

[5] D. Haugland, "A bidirectional greedy heuristic for the subspace selection problem," in *Lecture Notes in Computer Science*, vol. 4638. New York, NY, USA: Springer-Verlag, Aug. 2007, pp. 162–176.

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

[7] A. Jiang and H. Li, "Optimized cell programming for flash memories," in *Proc. IEEE PACRIM*, Victoria, BC, Canada, Aug. 2009, pp. 914–919.

[8] A. Jiang, H. Li, and J. Bruck, "On the capacity and programming of flash memories," *IEEE Trans. Inf. Theory*, vol. 58, no. 3, pp. 1549–1564, Mar. 2011.

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

[10] J. Lafferty and A. Vardy, "Ordered binary decision diagrams and minimal trellises," *IEEE Trans. Comput.*, vol. 48, no. 9, pp. 971–986, Sep. 1999.

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

[12] Q. Li, "WOM codes against inter-cell interference in NAND memories," in *Proc. 49th Annu. Allerton Conf. Commun., Control Comput.*, Monticello, IL, USA, Sep. 2011, pp. 1416–1423.

[13] 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 effect of BE-SONOS NAND flash," in *Proc. IEEE Int. Symp. Rel. Phys.*, vol. 30, May 2008, no. 11, pp. 693–694.

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

[15] R. Rivest and A. Shamir, "How to reuse a write-once memory," *Inf. Control*, vol. 55, nos. 1–3, pp. 1–19, Dec. 1982.

[16] K.-D. Suh, B.-H. Suh, Y.-H. Lim, Y.-J. C. J.-K. Kim, Y.-N. Koh, S.-S. Lee, S.-C. Kwon, B.-S. Choi, J.-S. Yum, J.-H. Choi, J.-R. Kim, and H.-K. Lim, "A 3.3 V 32 Mb NAND flash memory with incremental step pulse programming scheme," *IEEE J. Solid-State Circuits*, vol. 30, no. 11, pp. 1149–1156, Nov. 1995.

[17] E. Yaakobi, A. Jiang, P. H. Siegel, A. Vardy, and J. K. Wolf, "On the parallel programming of flash memory cells," in *Proc. IEEE ITW*, Dublin, Ireland, Sep. 2010, pp. 1–5.

Minghai Qin (S'11) received the B.E. degree in electronic and electrical engineering from Tsinghua University, Beijing, China, in 2009. He is currently pursuing the Ph.D. degree in electrical engineering from the Department of Electrical and Computer Engineering at the University of California, San Diego, where he is with the Center for Magnetic Recording Research.

Eitan Yaakobi (S'07–M'12) received the B.A. degrees in computer science and mathematics, and the M.Sc. degree in computer science from the Technion-Israel Institute of Technology, Haifa, Israel, in 2005 and 2007, respectively, and the Ph.D. degree in electrical engineering from the University of California, San Diego, in 2011.

He is a Postdoctoral Researcher in Electrical Engineering at the California Institute of Technology, Pasadena. His current research interests include coding theory, algebraic error-correction coding, and their applications for digital data storage and in particular for non-volatile memories.

Dr. Yaakobi received the Marconi Society Young Scholar in 2009 and the Intel Ph.D. Fellowship in 2010-2011.

Paul H. Siegel (M'82–SM'90–F'97) received the S.B. and Ph.D. degrees in mathematics from the Massachusetts Institute of Technology (MIT), Cambridge, in 1975 and 1979, respectively.

He held a Chaim Weizmann Postdoctoral Fellowship at the Courant Institute, New York University. He was with the IBM Research Division in San Jose, CA, from 1980 to 1995. He joined the faculty at the University of California, San Diego in July 1995, where he is currently Professor of Electrical and Computer Engineering in the Jacobs School of Engineering. He is affiliated with the Center for Magnetic Recording Research where he holds an endowed chair and served as Director from 2000 to 2011. His current research interests include information theory and communications, particularly coding and modulation techniques, with applications to digital data storage and transmission.

Prof. Siegel was a member of the Board of Governors of the IEEE Information Theory Society from 1991 to 1996 and from 2009 to 2011. He was re-elected for another 3-year term in 2012. He served as Co-Guest Editor of the May 1991 Special Issue on Coding for Storage Devices of the IEEE TRANSACTIONS ON INFORMATION THEORY. He served the same Transactions as Associate Editor for Coding Techniques from 1992 to 1995, and as Editor-in-Chief from July 2001 to July 2004. He was also Co-Guest Editor of the May/September 2001 two-part issue on The Turbo Principle: From Theory to Practice of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS.

Prof. Siegel was co-recipient, with R. Karabed, of the 1992 IEEE Information Theory Society Paper Award and shared the 1993 IEEE Communications Society Leonard G. Abraham Prize Paper Award with B.H. Marcus and J.K. Wolf. With J.B. Soriaga and H.D. Pfister, he received the 2007 Best Paper Award in Signal Processing and Coding for Data Storage from the Data Storage Technical Committee of the IEEE Communications Society. He holds several patents in the area of coding and detection, and was named a Master Inventor at IBM Research in 1994. He is a member of the National Academy of Engineering.