Writing on Dirty Flash Memory: Combating Inter-Cell Interference via Coding with Side Information

Yongjune Kim, Euiseok Hwang, and B. V. K. Vijaya Kumar

***Abstract*—High-density flash memories suffer from inter-cell interference (ICI) which threatens the reliability of stored data. In order to cope with the ICI problem, we propose a channel coding scheme with channel state information of flash memories (i.e., side information of ICI). This side information is obtained before writing data into flash memories and incorporated during the encoding stage. We show that flash memories under ICI problem can be transformed into the model of *memory with defective cells* due to the unique asymmetry property between write (page write) and erase (block erase) operations. Then, the channel coding for memory with defective cells is employed to combat ICI. Simulation results support that the proposed scheme with the ICI side information can effectively improve the decoding failure probability.**

***Index Terms*—Coding with side information, flash memories, intercell interference, partitioned linear block codes, stuck-at defects.**

I. INTRODUCTION

FLASH memory is the most important nonvolatile memory due to the rapid growth of mobile devices and solid-state drives (SSD). Aggressive scaling down of cell size has driven the continuous growth of flash memory density. However, flash memory cells suffer from higher inter-cell interference (ICI) [3]–[5] as the distance between adjacent cells decreases due to scaling down.

In order to cope with the ICI, various approaches have been proposed. Device level approaches such as new materials and novel cell structures try to reduce the parasitic capacitance between adjacent cells [5], [6]. At circuit and architecture levels, several write schemes and all bit-line (ABL) architecture were proposed to deal with the ICI [7]–[10]. Also, strong error control codes (ECC) such as low-density parity check (LDPC) codes and signal processing techniques have also been investigated [11]–[14]. The disadvantage of soft decision decoding and signal processing is the degradation of read speed due to multiple reads needed to obtain the soft decision values. In addition, constrained coding has been investigated to remove (or reduce) some data patterns which are more vulnerable to ICI [15]–[18]. However, the significant redundancy of constrained coding is a critical drawback. We note that a constrained coding scheme to address vertical ICI as well as horizontal ICI was proposed in [19]. Recently, a constrained coding scheme with minor rate loss was proposed for multi-level cell (MLC) flash memories [20].

In this paper, we propose a channel coding scheme to combat the ICI of flash memories. Instead of removing vulnerable data patterns as constrained coding does, we focus on the cells whose threshold voltages are already high before programming. Hence, these vulnerable cells could become erroneous even for a weak ICI. The encoder obtains the side information before writing data into flash memories and incorporates the obtained information during the encoding stage. We show that the flash memory channel with ICI is in line with the channel model of Costa's dirty paper coding in [21]. We first explain why flash memories are *dirty* due to ICI and then show that *dirty flash memory* can be transformed into *memory with defective cells* in [22]. The unique asymmetry property of flash memories (i.e., page write and block erase operations of flash memories) plays a pivotal role in this transformation. Then, we adopt channel coding for memory with defective cells to combat ICI effectively. It is a new finding that the unique asymmetry property of flash memories bridges *writing on dirty paper* and *coding for memory with defective cells.*

The proposed coding scheme would degrade the write speed instead of the read speed, which is an important advantage of the proposed scheme. Soft decision decoding and signal processing techniques for flash memories improve the decoding failure probability at the expense of decreased read speed due to multiple read operations to obtain soft decision values. In memory systems, it is well-known that the read speed is more critical than the write speed since the write operation is typically not on the critical path. Due to write buffers in the memory hierarchy, the write latency can be hidden [23], [24]. Also, the read operations are required more often than the write operations in many memory applications. Thus, the proposed scheme using side information at the encoder would be preferable over soft decision decoding from the perspective of speed performance. It is worth mentioning that

Manuscript received January 22, 2022, revised Junly 22, 2022; approved for publication by Alberto Tarable, Division I Editor, October 6, 2022.

This work was presented in part at the Allerton Conference on Communication, Control, and Computing, Monticello, IL, Oct. 2014 [1] and is based on a thesis submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Electrical and Computer Engineering at Carnegie Mellon University [2].

Y. Kim is with the Department of Electrical Engineering at Pohang University of Science and Technology (POSTECH), Pohang 37673, South Korea, email: yongjune@postech.ac.kr.

E. Hwang is with the School of Electrical Engineering and Computer Science, Gwangju Institute of Science and Technology (GIST), Gwangju 61005, Korea, email: euiseokh@gist.ac.kr.

B. V. K. Vijaya Kumar is with the Department of Electrical and Computer Engineering, Carnegie Mellon University, PA 15213, USA, email: kumar@ece.cmu.edu.

Y. Kim is the corresponding author.

Digital Object Identifier: 10.23919/JCN.2022.000045

the proposed scheme can be combined with LDPC codes using the technique of additive encoding LDPC codes [25], [26] if we are willing to accept the degradation of read speed.

The new contributions in this paper relative to our earlier conference paper [1] include the proposed coding scheme for MLC flash memories. We leveraged the widely used multi-page architecture and the page address ordering of flash memories [7], [8] to develop the coding scheme for MLC flash memories. The proposed scheme can be extended to three and more bits per cell MLC flash memories. In addition, we provide an analysis of the upper bound on the decoding failure probability, which theoretically supports that the proposed coding scheme with side information of ICI can improve the decoding failure probability. We also include extensive simulation results for both single-level cell (SLC) and MLC flash memories.

The rest of this paper is organized as follows. Section II explains the background and unique properties of flash memories. Section III introduces coding for memory with defective cells, which will be used to combat the ICI of flash memories. Section IV presents the proposed coding scheme to combat the ICI of flash memories by using the channel state information at the encoder. Section V provides simulation results and Section VI concludes the paper.

II. Basics of Flash Memories and ICI

In this section, we explain the unique asymmetry property of flash memories and the ICI.

A. Basic Operations and Asymmetry between Write and Erase Operations

Each cell of flash memories is a floating gate transistor whose threshold voltage can be configured by controlling the amount of electron charge in the floating gate. More electrons in the floating gate make the corresponding cell's threshold voltage higher. As shown in Fig. 1, each flash memory block is a cell array where each cell is connected to a word-line (WL) and a bit-line (BL).

In order to store b bits per cell, each cell's threshold voltage is divided into 2^b states, similar to pulse amplitude modulation (PAM). Fig. 2(a) shows the threshold voltage distribution of 1-bit per cell flash memory, which is traditionally called single-level cell (SLC). Initially, all memory cells are erased, so their threshold voltages are in the lowest erase state S_0. In order to store data, some of the cells in S_0 should be written (i.e., programmed) into S_1.

For multi-level cell (MLC) flash memories (i.e., $b \geq 2$), some of cells in S_0 (erase state) will be written into $S_1, \cdots, S_{2^b-1}$ (program states) as shown in Fig. 2(b). For b bits per cell flash memory, each WL stores b pages of data.

In write operation, the page buffer in Fig. 1 is loaded with a unit of page data. Depending on the loaded data in the page buffer, some of cells remain in erase state and others are programmed into program states.

The most widely used write operation scheme is the incremental step pulse programming (ISPP) scheme, which was proposed to maintain a tight threshold voltage distribution for high reliability [27]. The ISPP is based on repeated program and verify cycles with the staircase program voltage V_{pp}. Each program state S_i associates with a verify level ν_i that is used in the verify operation. During each program and verify cycle, the floating gate threshold voltage is increased by the incremental step voltage ΔV_{pp} and then compared with the corresponding verify level. If the threshold voltage of the memory cell is still lower than the verify level, the program and verify iterations continue. Otherwise, further programming of this cell is disabled [13], [27].

The positions of program states are determined by verify levels and the tightness of each program state depends on the incremental step voltage ΔV_{pp}. By reducing ΔV_{pp}, the threshold voltage distribution can be made tighter, however the write time increases [27].

In read operation, the threshold voltages of cells in the same WL are compared to a given read level. After a read operation, a page of binary data is transferred to the page buffer in Fig. 1. The binary data shows whether the threshold voltage of each cell is lower or higher than the given read level. Namely, the

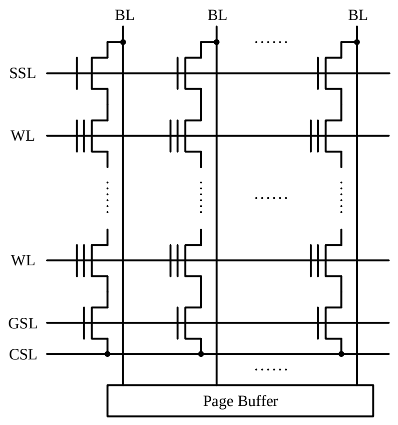
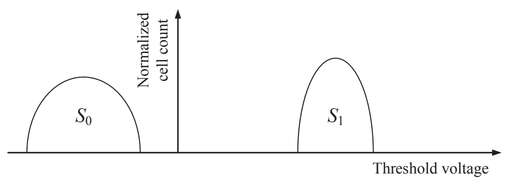
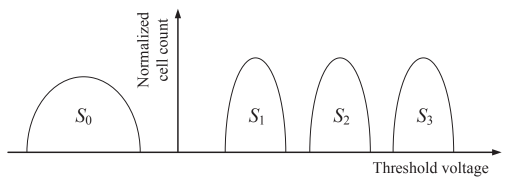


Fig. 1. A flash memory block where SSL, GSL, and CSL denote string select-line, ground select-line, and common source-line, respectively [27].



(a) Single-level cell (SLC) for $b = 1$



(b) Multi-level cell (MLC) for $b = 2$

Fig. 2. Threshold voltage distribution of flash memory cells.

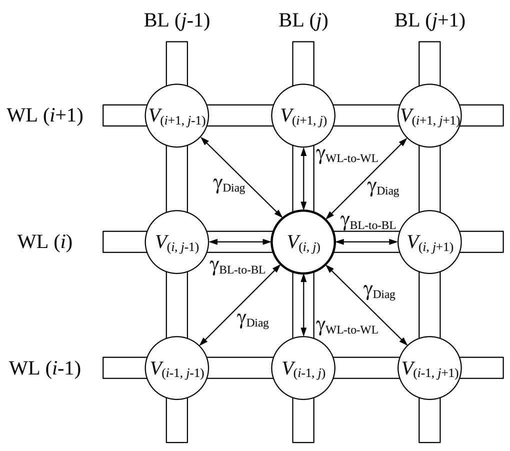


Fig. 3. Inter-cell interference (ICI) between adjacent cells of flash memories.

read operation of flash memory is a binary decision. Thus, multiple read operations are required to obtain a soft decision value, which lowers the read speed. The degradation of read speed is an important drawback of soft decision decoding [11].

The threshold voltage of the flash memory cell can be reduced by erase operation. In current NAND flash memory architectures, all the memory cells in the same flash memory block should be erased at the same time [27]. Note that a page of data (within a WL) can be written or read (generally, a flash memory block consists of 64 WLs). In addition, the threshold voltage of the cell should be moved into the lowest state S_0 by erase operation whereas a slight increase of threshold voltage is possible by ISPP during write operation [27]. These unique properties of flash memory cause *asymmetry between write and erase operations*.

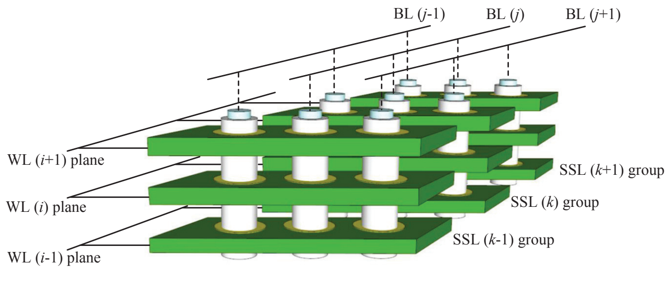
B. ICI

In flash memory, the threshold voltage shift of one cell affects the threshold voltage of its adjacent cell because of the ICI. The ICI is mainly attributed to parasitic capacitance between adjacent cells [3]–[5].

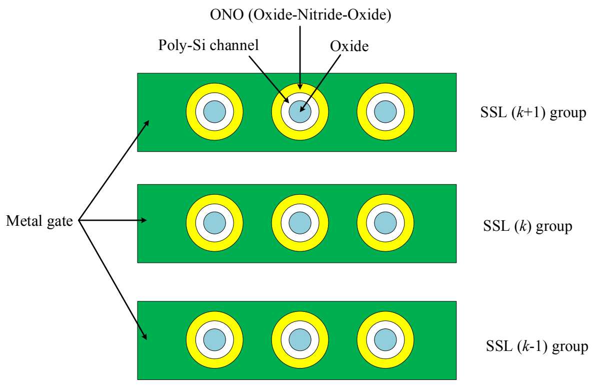
Fig. 3 illustrates the ICI between adjacent cells. $V_{(i,j)}$ is the threshold voltage of (i,j) cell which is situated at ith WL and jth BL. $\gamma_{\text{WL-to-WL}}$ is the coupling ratio between WL and adjacent WL. Also, $\gamma_{\text{BL-to-BL}}$ is the coupling ratio between BL and adjacent BL. Finally, γ_{Diag} is diagonal coupling ratio. These coupling ratios depend on parasitic capacitances between adjacent cells. As the cell size continues to shrink, the distances between cells become smaller, which results in the increase of parasitic capacitances. The increase of parasitic capacitances causes the increase of coupling ratios [3], [4].

According to [4], the threshold voltage shift $\Delta_{\text{ICI}}V_{(i,j)}$ of (i,j) cell due to the ICI is given by

$$\begin{aligned}\Delta_{\text{ICI}}V_{(i,j)} = {} & \gamma_{\text{WL-to-WL}}\left(\Delta V_{(i-1,j)} + \Delta V_{(i+1,j)}\right) \\ & + \gamma_{\text{BL-to-BL}}\left(\Delta V_{(i,j-1)} + \Delta V_{(i,j+1)}\right) \\ & + \gamma_{\text{Diag}}\left(\Delta V_{(i-1,j-1)} + \Delta V_{(i-1,j+1)}\right. \\ & \left. + \Delta V_{(i+1,j-1)} + \Delta V_{(i+1,j+1)}\right), \qquad (1)\end{aligned}$$



(a) The structure of 3D flash memories



(b) Cross section diagram along single WL plane

Fig. 4. 3D flash memory cell array in [28].

where $\Delta V_{(i\pm1,j\pm1)}$ in the right hand side of (1) represent the threshold voltage shifts of adjacent cells after the (i,j) cell has been written. Note that Δ_{ICI} and Δ denote the voltage shift by the ICI and the voltage shift by writing (programming), respectively. It is well known that the ICI that happens before writing (i,j) cell can be compensated by several write schemes so long as (i,j) cell is in program states [8], [9]. Note that the ICI to (i,j) cell in S_0 cannot be compensated by these write schemes since a cell in S_0 is never written (i.e., stays in S_0) [9], [16]. Hence, cells in S_0 are most vulnerable to the ICI.

By taking into account the structure of 3D flash memories, we can address the ICI of 3D flash memories. Suppose that $V_{(i,j,k)}$ is the threshold voltage of (i,j,k) cell which is situated at ith WL, jth BL, and kth SSL as shown in Fig. 4. The threshold voltage shift $\Delta_{\text{ICI}}V_{(i,j,k)}$ of the (i,j,k) cell due to the ICI can be given by

$$\begin{aligned}\Delta_{\text{ICI}}V_{(i,j,k)} = {} & \gamma_{\text{WL-to-WL}}\left(\Delta V_{(i-1,j,k)} + \Delta V_{(i+1,j,k)}\right) \\ & + \gamma_{\text{BL-to-BL}}\left(\Delta V_{(i,j-1,k)} + \Delta V_{(i,j+1,k)}\right) \\ & + \gamma_{\text{SSL-to-SSL}}\left(\Delta V_{(i,j,k-1)} + \Delta V_{(i,j,k+1)}\right), \qquad (2)\end{aligned}$$

which is an extension of the ICI model of 2D planar flash memories in (1). $\Delta V_{(i\pm1,j\pm1,k\pm1)}$ in the right-hand side represents the threshold voltage shifts of adjacent cells after the (i,j,k) cell has been written. Note that $\gamma_{\text{WL-to-WL}}$ is the coupling ratio between WL plane and the adjacent WL plane. Also, $\gamma_{\text{BL-to-BL}}$ is the coupling ratio between BL and adjacent BL. Finally, $\gamma_{\text{SSL-to-SSL}}$ is coupling ratio between string-select-line (SSL) group and its adjacent SSL group. The diagonal

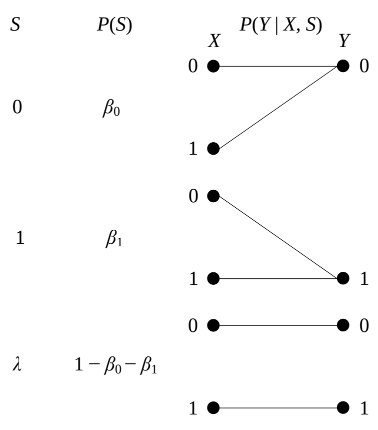


Fig. 5. Memory with defective cells [22].

ICI terms are neglected since they are very small due to the long distance between corresponding cells.

III. Coding for Memory with Defective Cells

Since the problem of combating the ICI is transformed into coding for memory with defective cells of [22], the channel model of memory with defective cells and relevant coding techniques are summarized in this section. The channel model of memory with defective cells (i.e., stuck-at defects) in Fig. 5 can be described as follows.

$$Y = X \circ S, \tag{3}$$

where "$\circ$" denotes the operator $\circ : \{0,1\} \times \{0,1,\lambda\} \to \{0,1\}$ as in [29]

$$x \circ s = \begin{cases} x, & \text{if } s = \lambda; \\ s, & \text{if } s \neq \lambda, \end{cases} \tag{4}$$

where $s \in \{0,1\}$ indicates that the cell's output is fixed to the stuck-at value s whereas $s = \lambda$ indicates that the cell is not defective (i.e., normal).

By using the operator $\circ$, an n-cell memory with defects is modeled by

$$\mathbf{y} = \mathbf{x} \circ \mathbf{s} = \mathbf{c} \circ \mathbf{s}, \tag{5}$$

where $\mathbf{x}, \mathbf{y} \in \{0,1\}^{n \times 1}$ are the channel input vector, and the channel output vector, respectively. Note that we use the column vector representation. We suppose that the $\mathbf{x} = \mathbf{c}$ where $\mathbf{c}$ denotes the codeword. Also, the channel state vector $\mathbf{s} \in \{0,1,\lambda\}^{n \times 1}$ represents the defect information in the n-cell memory. Note that $\circ$ is the vector component-wise operator. As shown in Fig. 5, the probabilities of stuck-at defects and normal cells are given by

$$P(S = s) = \begin{cases} \beta_i, & \text{if } s = i; \\ 1 - \beta_0 - \beta_1, & \text{if } s = \lambda, \end{cases} \tag{6}$$

where the defect probability is $P(S \neq \lambda) = \beta_0 + \beta_1 = \beta$.

If we include random errors, then (3) can be changed into

$$Y = X \circ S + Z, \tag{7}$$

where $X \circ S$ is the input to the binary symmetric channel (BSC) with the crossover probability p. Hence, Z represents the random error whose probability is given by

$$P(Z = z) = \begin{cases} 1 - p, & z = 0; \\ p, & z = 1. \end{cases} \tag{8}$$

An n-cell memory with defects and random errors is modeled by $\mathbf{y} = \mathbf{c} \circ \mathbf{s} + \mathbf{z}$ where $\mathbf{z} \in \{0,1\}^{n \times 1}$ is the random error vector. The number of defects in n cells is equal to the number of non-λ components in $\mathbf{s}$. The number of errors due to defects is given by

$$\|\mathbf{c} \circ \mathbf{s} - \mathbf{c}\|, \tag{9}$$

where $\|\cdot\|$ is the Hamming weight of the vector. The number of errors in a codeword due to defects is less than or equal to the number of defects since some of the stuck-at values are the same as the codeword values and thus do not cause errors.

Tsybakov proposed the *additive encoding* approach which masks defects by adding a carefully selected binary vector [30], [31]. Masking defects aims to produce a codeword whose values at the locations of defects match the stuck-at values at those locations, which reduces the number of errors due to defects. Subsequently, several coding techniques were proposed [32]–[37].

Heegard extended the additive encoding and defined the $[n,k,l]$ partitioned linear block codes (PLBC) that mask stuck-at defects and correct random errors [29]. The $[n,k,l]$ PLBC consists of a pair of linear subspaces $\mathcal{C}_1 \subset \{0,1\}^{n \times 1}$ of dimension k and $\mathcal{C}_0 \subset \{0,1\}^{n \times 1}$ of dimension l such that $\mathcal{C}_1 \cap \mathcal{C}_0 = \{\mathbf{0}\}$. Note that the code rate is $R = k/n$.

The PLBC can be constructed based on several linear block codes. The partitioned Bose-Chaudhuri-Hocquenghem (PBCH) codes can be designed by the construction of BCH codes [29]. Also, PLBC includes additive encoding LDPC codes which combine additive encoding and LDPC decoding [25], [26].

The encoding and decoding of PLBC [29] are summarized as follows.

Encoding: A message $\mathbf{m} \in \{0,1\}^{k \times 1}$ is encoded to a corresponding codeword $\mathbf{c} \in \mathcal{C}$ as follows.

$$\mathbf{c} = G_1 \mathbf{m} + G_0 \mathbf{d} = \widetilde{G} \begin{bmatrix} \mathbf{m} \\ \mathbf{d} \end{bmatrix}, \tag{10}$$

where $\widetilde{G} = [G_1 \; G_0]$. Also, $\mathbf{c}_1 = G_1 \mathbf{m} \in \mathcal{C}_1$ and $\mathbf{c}_0 = G_0 \mathbf{d} \in \mathcal{C}_0$. Note that the optimal $\mathbf{d} \in \{0,1\}^{l \times 1}$ is chosen to minimize $\|\mathbf{c} \circ \mathbf{s} - \mathbf{c}\|$. The generator matrix G_1 is an $n \times k$ matrix and the generator matrix G_0 is an $n \times l$ matrix. Thus, $\mathcal{C}$ can be regarded as an $[n, k+l]$ linear block code with the generator matrix $\widetilde{G} = [G_1 \; G_0]$. The construction of G_1 and G_0 are described in [29]. It is a key problem to determine $\mathbf{d}$ judiciously, which will be explained later.

Decoding: Receive $\mathbf{y} = \mathbf{c} \circ \mathbf{s} + \mathbf{z}$. Compute the syndrome $\mathbf{v} = \widetilde{H}^T \mathbf{y} = \widetilde{H}^T (\mathbf{c} + \mathbf{z}) = \widetilde{H}^T \mathbf{z}$ (superscript T denotes transpose) and estimate $\widehat{\mathbf{z}} \in \{0,1\}^{n \times 1}$ from $\widetilde{H}^T \mathbf{z} = \mathbf{v}$, which can be done by standard BCH decoding algorithms for PBCH codes. Then $\widehat{\mathbf{m}} = \widetilde{G}_1^T \widehat{\mathbf{c}}$ where $\widehat{\mathbf{c}} = \mathbf{y} + \widehat{\mathbf{z}}$.

The parity check matrix $\widetilde{H}$ is an $n \times r$ matrix such that $\widetilde{H}^T \widetilde{G} = 0_{r,k+l}$ (the $r \times (k+l)$ zero matrix) and $k+l+r = n$. The message inverse matrix $\widetilde{G}_1$ is defined as an $n \times k$ matrix such that $\widetilde{G}_1^T G_1 = I_k$ (the k-dimensional identity matrix) and $\widetilde{G}_1^T G_0 = 0_{k,l}$ [29].

In [29], the minimum distances (d_0, d_1) of an $[n,k,l]$ PLBC are defined as

$$d_0 = \min_{\substack{\mathbf{c} \neq \mathbf{0} \\ G_0^T \mathbf{c} = \mathbf{0}}} \|\mathbf{c}\|, \quad d_1 = \min_{\substack{\widetilde{G}_1^T \mathbf{c} \neq \mathbf{0} \\ \widetilde{H}^T \mathbf{c} = \mathbf{0}}} \|\mathbf{c}\|, \tag{11}$$

where d_1 is greater than or equal to the minimum distance of the $[n, k+l]$ code with the parity check matrix $\widetilde{H}$, while d_0 is the minimum distance of the $[n, k+r]$ code with the parity check matrix G_0. An $[n,k,l]$ PLBC with (d_0, d_1) guarantees to mask up to $d_0 - 1$ defects and correct $\lfloor \frac{d_1 - 1}{2} \rfloor$ random errors (see [29, Theorem 1]). These (d_0, d_1) parameters will play a critical role in the following encoding algorithm (Algorithm 1) and theoretical analysis.

The encoder should choose $\mathbf{d}$ judiciously by considering both $\mathbf{c}_1$ and $\mathbf{s}$. The optimal $\mathbf{d}$ is chosen to minimize the number of errors due to defects, i.e., $\|\mathbf{c} \circ \mathbf{s} - \mathbf{c}\|$ [29].

$$\begin{aligned} \mathbf{d}^* &= \underset{\mathbf{d}}{\operatorname{argmin}} \|\mathbf{c} \circ \mathbf{s} - \mathbf{c}\| = \underset{\mathbf{d}}{\operatorname{argmin}} \left\|\mathbf{s}^{\mathcal{U}} - G_0^{\mathcal{U}} \mathbf{d} - G_1^{\mathcal{U}} \mathbf{m}\right\| \\ &= \underset{\mathbf{d}}{\operatorname{argmin}} \left\|G_0^{\mathcal{U}} \mathbf{d} - \mathbf{b}^{\mathcal{U}}\right\|, \end{aligned} \tag{12}$$

where $\mathcal{U} = \{i_1, \cdots, i_u\}$ indicates the set of locations of stuck-at defects. We use the notation of $\mathbf{s}^{\mathcal{U}} = (s_{i_1}, \cdots, s_{i_u})^T$, $G_0^{\mathcal{U}} = \left[\mathbf{g}_{0,i_1}^T, \cdots, \mathbf{g}_{0,i_u}^T\right]^T$, and $G_1^{\mathcal{U}} = \left[\mathbf{g}_{1,i_1}^T, \cdots, \mathbf{g}_{1,i_u}^T\right]^T$ where $\mathbf{g}_{0,i}$ and $\mathbf{g}_{1,i}$ are the ith rows of G_0 and G_1, respectively. Also, note that $\mathbf{b} = \mathbf{s} - G_1 \mathbf{m}$ and $\mathbf{b}^{\mathcal{U}} = \mathbf{s}^{\mathcal{U}} - G_1^{\mathcal{U}} \mathbf{m}$.

Since the computational complexity for finding the following optimal $\mathbf{d}$ is exponential, we use the *two-step encoding* for determining $\mathbf{d}$ which was proposed in [36] (see Algorithm 1). In Step 1 encoding, we just try to solve the following linear equation.

$$G_0^{\mathcal{U}} \mathbf{d} = \mathbf{b}^{\mathcal{U}}. \tag{13}$$

Gaussian elimination or some other linear equation solution methods can be used to solve (13) with $\mathcal{O}\left(n^3\right)$ complexity. It is clear that (13) has at least one solution if and only if

$$\operatorname{rank}\left(G_0^{\mathcal{U}}\right) = \operatorname{rank}\left(G_0^{\mathcal{U}} \mid \mathbf{b}^{\mathcal{U}}\right), \tag{14}$$

where $\left(G_0^{\mathcal{U}} \mid \mathbf{b}^{\mathcal{U}}\right)$ denotes the augmented matrix. If $u < d_0$, $\operatorname{rank}\left(G_0^{\mathcal{U}}\right)$ is always u by (11). Hence, (14) holds and at least one solution $\mathbf{d}$ exists. If $u \geq d_0$, the encoder may fail to find a solution of (13).

If the encoder fails to solve (13), the encoder sets $\mathcal{U}' = \{i_1, \cdots, i_{d_0 - 1}\}$ by randomly choosing $d_0 - 1$ defect locations among $\mathcal{U}$. Afterwards, the encoder solves $G_0^{\mathcal{U}'} \mathbf{d} = \mathbf{b}^{\mathcal{U}'}$ where a solution $\mathbf{d}$ always exists by the definition of d_0 in (11). If $\mathbf{d}$ is obtained in Step 2 encoding, then the number of unmasked defects becomes $u - (d_0 - 1)$ instead of u. Since the unmasked defects can be regarded as random errors, the reduction of unmasked defects by Step 2 encoding can improve the decoding failure probability of PLBC [38].

Algorithm 1 Two-step encoding [36], [38]

```
1: Step 1 encoding: Try to solve (13), i.e., $G_0^{\mathcal{U}}\mathbf{d} = \mathbf{b}^{\mathcal{U}}$.
2: if $u < d_0$ then go to End. ▷ A solution $\mathbf{d}$ always exists.
3: else                ▷ A solution $\mathbf{d}$ exists so long as (14) holds.
4:     if $\mathbf{d}$ exists then go to End.
5:     else go to Step 2.
6:     end if
7: end if
8: Step 2 encoding:
     • Choose $d_0 - 1$ locations among $\mathcal{U}$ and define $\mathcal{U}' = \{i_1, \cdots, i_{d_0-1}\}$.
     • Solve the following linear equation: $G_0^{\mathcal{U}'}\mathbf{d} = \mathbf{b}^{\mathcal{U}'}$.
       ▷ A solution $\mathbf{d}$ always exists.
9: End
```

In [39], the upper bound on the probability of decoding failure of PBCH codes was derived for the channel where defective cells do not suffer from random errors.

$$\begin{aligned} & P\left(\widehat{\mathbf{m}} \neq \mathbf{m}\right) \\ & \leq \sum_{u=d_0}^{n} \left\{ \binom{n}{u} \beta^u (1-\beta)^{n-u} \min\left\{ \frac{\sum_{w=d_0}^{u} B_{0,w} \binom{n-w}{u-w}}{\binom{n}{u}}, 1 \right\} \right. \\ & \quad \times \left. \sum_{t=\max\{0, t_1 + d_0 - u\}}^{n} \binom{n-u}{t} p^t (1-p)^{n-t} \right\} \\ & \quad + \sum_{u=0}^{n} \left\{ \binom{n}{u} \beta^u (1-\beta)^{n-u} \right. \\ & \quad \times \left. \sum_{t=t_1+1}^{n-u} \binom{n-u}{t} p^t (1-p)^{n-u-t} \right\}. \end{aligned} \tag{15}$$

If the defective cells as well as normal cells suffer from random errors, then the upper bound can be modified as follows [38].

$$\begin{aligned} & P\left(\widehat{\mathbf{m}} \neq \mathbf{m}\right) \\ & \leq \sum_{u=d_0}^{n} \left\{ \binom{n}{u} \beta^u (1-\beta)^{n-u} \min\left\{ \frac{\sum_{w=d_0}^{u} B_{0,w} \binom{n-w}{u-w}}{\binom{n}{u}}, 1 \right\} \right. \\ & \quad \times \left. \sum_{t=\max\{0, t_1 + d_0 - u\}}^{n} \binom{n}{t} p^t (1-p)^{n-t} \right\} \\ & \quad + \sum_{t=t_1+1}^{n} \binom{n}{t} p^t (1-p)^{n-t}, \end{aligned} \tag{16}$$

where $B_{0,w}$ is the weight distribution of $\mathcal{C}_0^{\perp}$ (i.e., the dual code of $\mathcal{C}_0$). Also, $t_1 = \lfloor \frac{d_1 - 1}{2} \rfloor$ is the error correcting capability of $\mathcal{C}$. In particular, the upper bound (16) is useful to estimate the probability of decoding failure of the proposed schemes since the cells regarded as defects would suffer from random errors in the proposed schemes.

IV. Combating Inevitable ICI for Flash Memories

In this section, we propose a coding scheme for combating the ICI. After introducing the channel model of flash memories, we explain the proposed coding scheme that reduces the negative impact of ICI by using coding with side information.

A. Channel Model of Dirty Flash Memories

The channel model of flash memories can be given by

$$\begin{aligned} Y &= X + S_I + Z \\ &= X + Z_{\text{write}} + S_I + Z_{\text{read}} \\ &= V + S_I + Z_{\text{read}}, \end{aligned} \tag{17}$$

where X and Y are the channel input and output. Also, S_I represents the ICI from adjacent cells. The additive random noise Z is a sum of Z_{write} and Z_{read} where Z_{write} is the write noise due to the initial threshold voltage distribution after erase operation and the incremental step voltage ΔV_{pp} of ISPP. Z_{read} is the read noise due to other noise sources.

Since the write noise Z_{write} precedes the ICI S_I, we consider a random variable $V = X + Z_{\text{write}}$. As shown in (1), the shifts of V in adjacent cells determine the ICI S_I. Thus, we claim that the ICI S_I of (i, j) cell is given by

$$S_I = \Delta_{\text{ICI}} V_{(i,j)}, \tag{18}$$

where $\Delta_{\text{ICI}} V_{(i,j)}$ is defined by (1). The read noise Z_{read} happens after ICI. The channel model of (17) is supported by experimental results from the 2x nm NAND flash memories [40].

Notice that (17) can be considered as the interference channel in [21], which is given by

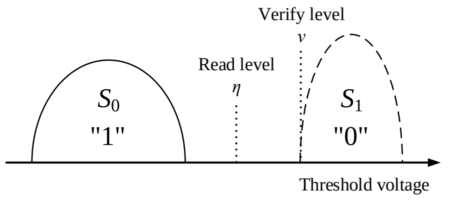
$$Y = X + S_{\text{I}} + Z, \tag{19}$$

where X and Y are the channel input and output, respectively. Also, $S_{\text{I}} \sim N\left(0, \sigma_{S_{\text{I}}}^2\right)$ represents interference and $Z \sim N\left(0, \sigma_Z^2\right)$ denotes additive noise. Note that S_{I} and Z are independent and the channel input satisfies an average power constraint $\frac{1}{n}\sum_{i=1}^{n} X_i^2 \le P$ for the channel input vector $X^n = (X_1, \cdots, X_n)$.

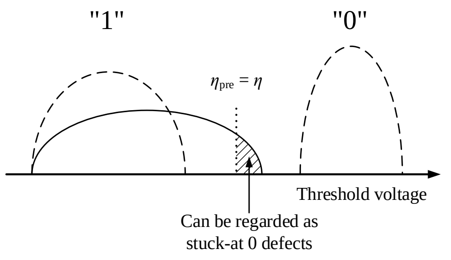
Now, we discuss why it is difficult for the encoder to know S_I of (17). First, the encoder has to know the channel input X of adjacent cells in different WLs to know S_I (17). Since the write operation is performed page by page, it is possible for the encoder to know the channel input of cells in several WLs only in the case where a large number of continuous pages are written at a time. Even in the case where the encoder knows enough channel input X of several WLs in advance, it is still difficult to know the random variable $V = X + Z_{\text{write}}$ that determines S_I because of the random write noise Z_{write}.

In addition, it is much more complicated to know the voltage shifts of adjacent cells (i.e., $\Delta V_{(i\pm1,j\pm1)}$ in (1)) since flash memory's read operation is an inherently binary decision. Hence, multiple read operations are required to know $\Delta V_{(i\pm1,j\pm1)}$, which significantly reduces the read speed performance.

Since it is difficult for the encoder to know the ICI S_I due to these reasons, we change *the flash memory channel with ICI* into the problem of *flash memory with defective cells*. After this change, the encoder uses the side information of defects rather than the ICI.



(a) Threshold voltage distribution of cells in the ith WL before writing the $(i-1)$th WL



(b) Threshold voltage distribution of cells in the ith WL after writing the $(i-1)$th WL (before writing the ith WL)

Fig. 6. Change from the flash memory channel with the ICI to the model of memory with defective cells by one pre-read operation.

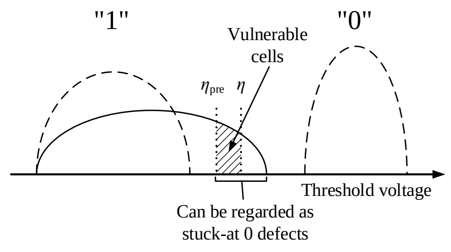


Fig. 7. Vulnerable cells can also be regarded as stuck-at 0 defects by setting a pre-read level such that $\eta_{\text{pre}} < \eta$.

B. Proposed Scheme for Flash Memories: Combating ICI by Coding with Side Information

Now we propose our scheme to combat the ICI by coding with side information. First, we describe how to transform the flash memory channel with ICI into the model of memory with defective cells. After this transformation, we adopt the coding scheme in Section III to combat the ICI of flash memories.

1) SLC: Fig. 6 shows the threshold voltage distribution of cells in the ith WL before writing. Initially, all cells are in the erase state S_0 as shown in Fig. 6(a). However, after writing the adjacent $(i-1)$th WL, the threshold voltages of cells in the ith WL will be distorted due to the ICI from the $(i-1)$th WL as shown in Fig. 6(b). Thus, some of the cells' threshold voltages can be higher than the given read level η although the ith WL is yet to be written.

As explained in Section II-B, the threshold voltage of flash memory cells cannot be reduced during write operation. In order to decrease the threshold voltage of a cell, we have to erase the whole flash memory block. Thus, the cell whose threshold voltage is higher than the read level η will be detected as S_1. Assume that S_0 and S_1 denote the data "1"

Algorithm 2 Proposed scheme for SLC

```
1: Get the message $\mathbf{m}$ for the $i$th WL.
2: Read operation with the pre-read level $\eta_{\text{pre}}$ for the $i$th WL.
3: Set the side information $\mathbf{s}$ where $s_j \in \{\lambda, 0\}^{n\times 1}$ by the following rule:
   $$s_j = \begin{cases} 0, & \text{if } V_{(i,j)} \geq \eta_{\text{pre}}; \\ \lambda, & \text{otherwise.} \end{cases} \quad (20)$$
4: Determine $\mathbf{d}$ by Algorithm 1.
5: PLBC encoding by (10).
6: Write the codeword $\mathbf{c}$ into the $i$th WL.
```

and "0" respectively. If a "1" is attempted to be written to this cell, an error results. However, "0" can be written into this cell. Thus, these cells can be regarded as stuck-at 0 defects in Fig. 5. If some of the cells regarded as stuck-at 0 defects may be "1" due to the read noise Z_{read}, those errors can be regarded as random errors and corrected by the standard ECC.

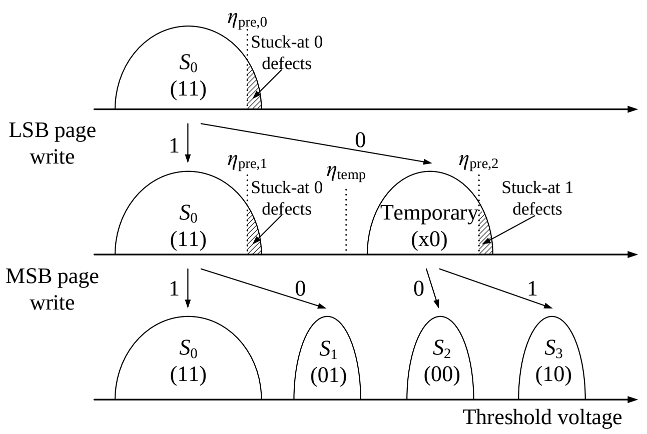
The defect information of stuck-at 0 defects can be obtained by just one read operation before writing the ith WL. Before writing the ith WL, the read operation is performed at the given read level, i.e., *pre-read operation*. When the read level for the pre-read operation, i.e., *pre-read level* η_{pre} is the same as the read level η, the cells whose threshold voltages are higher than the read level η can be identified by the pre-read operation. Thus, the encoder can incorporate the side information of defects and reduce the errors by highly interfered cells.

The proposed scheme for SLC flash memories is described in Algorithm 2. Using only one pre-read operation before writing, the flash memory channel with the ICI in (17) can be changed into (7) of *binary* memory with defective cells. Now, we can combat the ICI by the proposed coding scheme for memory with defective cells.

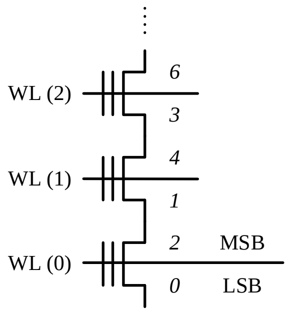
It is worth mentioning that S does not reveal the BL-to-BL ICI from the same WL and the ICI from the $(i+1)$th WL, which are subsequent ICI since the pre-read operation is done before the write operations of ith and $(i+1)$th WLs.

However, the effect of these subsequent ICI can be alleviated by changing the pre-read level. Suppose that the pre-read level η_{pre} is lower than the read level η as shown in Fig. 7. A cell whose threshold voltage is between η_{pre} and η is a vulnerable cell although it is not a stuck-at 0 defect. When the data "1" is written to this cell, the ISPP cannot change the threshold voltage of this cell and its threshold voltage is near η. Thus, it is vulnerable to the subsequent ICI and read noise. On the other hand, the cell's threshold voltage will be higher than a verify level of S_1 by the ISPP if the data "0" is written to this cell. Note that the verify level of S_1 is higher than the read level η.

Thus, by setting a pre-read level such that $\eta_{\text{pre}} < \eta$, we can regard all the cells whose threshold voltages are higher than the pre-read level η_{pre} as stuck-at 0 defects. Using coding with side information of defects, only the data "0" will be written to these cells. Thus, we can obtain more noise margin between S_0 and S_1 and prevent the subsequent ICI and read noise.



(a) Extension to MLC flash memories



(b) Page address ordering to reduce the ICI from the next WL [8]

Fig. 8. Extension to MLC flash memories.

2) MLC: We can extend our proposed scheme to MLC flash memory. Most MLC flash memories adopt the multi-page architecture [7]. In 2 bits per cell flash memories, the least significant bit (LSB) and most significant bit (MSB) are mapped to two separate pages: LSB page and MSB page. Since one page is the unit of data that is written and read at one time, the ECC should be applied within the same page.

Fig. 8 explains the most widely used MLC write operation in [8]. At the LSB page writing stage, the cell can be written from "11" to "x0" as a temporary state just like SLC write operation. Before writing the MSB page, a read operation is required to detect the LSB page data. If the threshold voltage of a cell is lower than η_{temp}, then we decide that this cell is in the erase state S_0 (11) and the corresponding LSB data is 1. During the MSB page writing stage, this cell is programmed to either S_0 or S_1 depending on the MSB page data. Otherwise, the cell is in the temporary state (x0) and its LSB data is 0. This cell is programmed to either S_2 or S_3 corresponding to the MSB page data. It is worth mentioning that the MSB page write operation can be separated by two sets of SLC write depending on the LSB page data: 1) from S_0 to S_0 or S_1 and 2) from a temporary state to S_2 or S_3.

The ICI from the next WL can be reduced by performing MSB page writing for a selected WL after writing the LSB page of next WL [8]. The ICI from writing the LSB page in $(i+1)$th WL can be compensated during writing the MSB page

Algorithm 3 Proposed scheme for MLC

```
1: Get the message $\mathbf{m}$ for the $i$th WL's MSB page.
2: Read operation with pre-read level $\eta_{\text{pre},1}$ for the $i$th WL.
3: Read operation with pre-read level $\eta_{\text{pre},2}$ for the $i$th WL.
4: Set the side information $\mathbf{s}$ where $s_j \in \{\lambda, 0\}^{n\times 1}$ by the following rule:
   $$s_j = \begin{cases} 0, & \text{if } \eta_{\text{pre, 1}} \le V_{(i,j)} < \eta_{\text{temp}}; \\ 1, & \text{if } V_{(i,j)} \ge \eta_{\text{pre},2}; \\ \lambda, & \text{otherwise.} \end{cases} \quad (21)$$
5: Determine $\mathbf{d}$ by Algorithm 1.
6: PLBC encoding by (10).
7: Write the codeword $\mathbf{c}$ into the $i$th WL's MSB page.
```

in ith WL due to the page address ordering in Fig. 8(b). The details of multi-page architecture and page address ordering can be found in [7] and [8].

If the LSB page data detection is erroneous, the finally programmed state can be wrong. Suppose that the LSB page data 1 is misread as 0 because the threshold voltage of corresponding cell is higher than η_{temp}. If the MSB page data is 1, then the final state will be S_3 (10) instead of S_0 (11) [41], [42]. Hence, an error happens on the LSB page. We call this error as *internal error*, which can result in a large magnitude error (e.g., from S_0 to S_3 and vice versa). These internal errors can be reduced by the proposed Algorithm 2 with the pre-read level of $\eta_{\text{pre},0}$.

However, much more random errors than internal errors happen between S_1 and S_2. The reason is that the noise margin between S_1 and S_2 is much smaller than that of erase state S_0 (11) and temporary state (x0) [42]. LSB page errors occur between S_1 and S_2 and most of LSB errors are not related to the side information of stuck-at 0 defects. Instead, they depend on the noise margin between S_1 and S_2 decided during the MSB write operation. Hence, we do not apply the proposed scheme for LSB page although the proposed scheme can reduce the internal errors in the LSB page. We note that the proposed scheme for LSB page could be advantageous for other MLC architectures which are different from the architecture of [7].

Algorithm 3 describes the proposed scheme for two bits per cell MLC flash memories. For the MSB page, the proposed scheme can improve the decoding failure probability by using the side information. Before writing the MSB page, two pre-read operations at the pre-read levels of $\eta_{\text{pre},1}$ and $\eta_{\text{pre},2}$ are required to obtain the side information. A cell whose threshold voltage is between $\eta_{\text{pre},1}$ and η_{temp} will be regarded as a stuck-at 0 defect. Also, a cell whose threshold voltage is higher than $\eta_{\text{pre},2}$ is regarded as a stuck-at 1 defect. By using this side information of defects, the proposed scheme can improve the decoding failure probability. The proposed scheme can be extended to three and more bits per cell MLC flash memories by the same way if their programming schemes are extensions of two-phase MLC programming of Fig. 8. We note that the proposed coding schemes for MLC flash memories should be tailored by considering the programming schemes.

TABLE I
SIMULATION PARAMETERS OF FLASH MEMORIES.

Parameters	Values
Initial threshold voltage distribution	$\mathcal{N}\left(-4, 1^2\right)$
Verify levels	For SLC, $\nu_1 = 1$ For MLC, $\nu_1 = 1$, $\nu_2 = 2.5$, $\nu_3 = 4.5$
Incremental step voltage of ISPP	For SLC and LSB of MLC, $\Delta V_{\text{pp}} = 1$ For MSB of MLC, $\Delta V_{\text{pp}} = 0.25$
$(\gamma_{\text{WL-to-WL}}, \gamma_{\text{BL-to-BL}}, \gamma_{\text{Diag}})$	$(0.1 \times \alpha, 0.08 \times \alpha, 0.006 \times \alpha)$
Z_{read} of (17)	$\mathcal{N}\left(0, \sigma^2_{Z_{\text{read}}}\right)$

C. Dirty Paper vs. Dirty Flash Memory

In this subsection, we briefly explain the connections between *writing on dirty paper* [21] and *coding for memory with defective cells* [22]. Imagine a sheet of *lined* paper (Costa considered a sheet of *blank* paper in [21]). A flash memory block is a sheet of paper and each WL corresponds to a row between lines. If a row between lines is spacious, then the writer can easily write a message between lines. In order to write more messages on a sheet of paper, the writer tries to narrow the space between lines (i.e., scaling down). However, as the space between lines narrows, it is more difficult to write a message without crossing the lines (i.e., ICI). Eventually, after writing a message in a narrower space, the adjacent rows have more dirty spots (i.e., stuck-at defects) due to the ink marks crossing the line. One way to solve this problem is to erase the dirty spots in a corresponding row before writing. However, erasing a row is not permitted because of the asymmetry between write and erase operations in flash memory.

Now we consider another option instead of erasing a row before writing. Assume that the writer knows the location of the dirty spots, but the reader cannot distinguish between the message and the dirt [21]. Hence, the problem of writing on flash memory with the ICI can be considered as a Costa's writing on dirty paper, i.e., *writing on dirty flash memory*. Since the dirty spots are changed into stuck-at defects by the pre-read operation, writing on dirty flash memory is equivalent to writing on (flash) memory with defective cells. Thus, *writing on dirty paper* and *memory with defective cells* can be bridged in NAND flash memories.

V. SIMULATION RESULTS

We present the simulation results of the proposed scheme for flash memories. The simulation parameters are summarized in Table I. The initial threshold voltage distribution (after erasing a flash memory block) is assumed to be the Gaussian distribution $\mathcal{N}\left(-4, 1^2\right)$. ISPP was implemented with the parameters of the verify level for S_i, i.e., ν_i and the incremental step voltage ΔV_{pp} (see Section II-A). The variance of initial threshold voltage distribution and the incremental step voltage ΔV_{pp} work for Z_{write} of (17), which precedes the ICI. For MLC flash memories, we adopted the programming scheme

TABLE II
ALL POSSIBLE REDUNDANCY ALLOCATION CANDIDATES OF $[n = 1023, k = 923, l]$ PBCH CODES.

Code	l	r	d_0	d_1	Notes
0	0	100	0	21	Only correcting random errors
1	10	90	3	19	
2	20	80	5	17	
3	30	70	7	15	
4	40	60	9	13	
5	50	50	11	11	
6	60	40	13	9	
7	70	30	15	7	
8	80	20	17	5	
9	90	10	19	3	
10	100	0	21	0	Only masking defects

in [8] and [42] (see Fig. 8), which is one of the widely used programming schemes.

The ICI S_I is calculated by (1) where the coupling ratios are $(\gamma_{\text{WL-to-WL}}, \gamma_{\text{BL-to-BL}}, \gamma_{\text{Diag}}) = (0.1 \times \alpha, 0.08 \times \alpha, 0.006 \times \alpha)$. The scaling factor α represents the ICI strength, and the ratios between $\gamma_{\text{WL-to-WL}}$, $\gamma_{\text{BL-to-BL}}$, and γ_{Diag} are taken from [3]. These ratios can be different for each product of flash memory. The read noise Z_{read} after the ICI is assumed to the $\mathcal{N}\left(0, \sigma^2_{Z_{\text{read}}}\right)$. The read noise would depend on the program/erase (P/E) cycles, the retention time, and the read disturbance. For a given read noise, we can adaptively estimate the threshold distributions (or the corresponding read levels) by leveraging prior work, e.g., [43], and then apply the proposed coding schemes.

Most of our simulation results were based on PBCH codes. The possible parameter sets of $[n = 1023, k = 923, l]$ PBCH codes are presented in Table II. However, the proposed scheme can be combined with additive encoding LDPC codes. After obtaining the defect information corresponding to the ICI by the proposed scheme, this defect information can be incorporated by additive encoding LDPC codes [25], [26]. In the two-dimensional magnetic recording (TDMR) channel where the soft decision value is obtained without the read speed degradation, [26] shows that additive encoding LDPC codes with defect information corresponding to the channel state of TDMR improve the decoding failure probability.

In Fig. 9, we compare the threshold distributions by the traditional coding scheme and the proposed coding scheme for the SLC flash memories. The threshold distributions are obtained by the normalized histograms. Fig. 9 shows that the proposed scheme can improve the threshold distribution by using the side information of highly interfered cells and vulnerable cells, which are regarded as stuck-at defects. In contrast, the standard channel coding schemes cannot improve the threshold voltage distribution. In particular, the threshold distribution of the erase state S_0 is improved since the vulnerable cells in S_0 are encoded into "0" and then programmed into the state S_1 (see Figs. 6 and 7). Since the width of the threshold voltage distribution of S_0 is reduced, the corresponding raw BER would decrease.

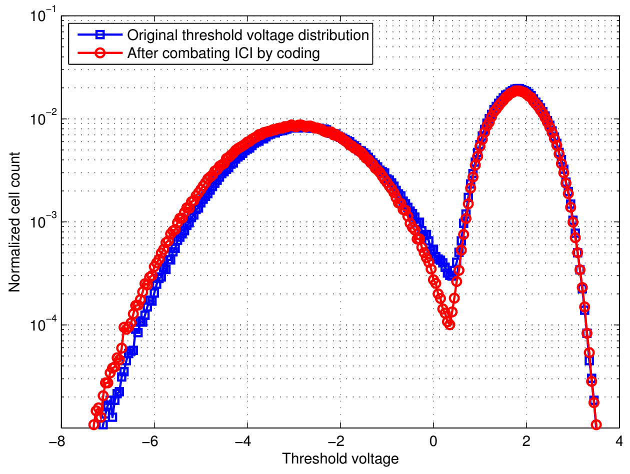
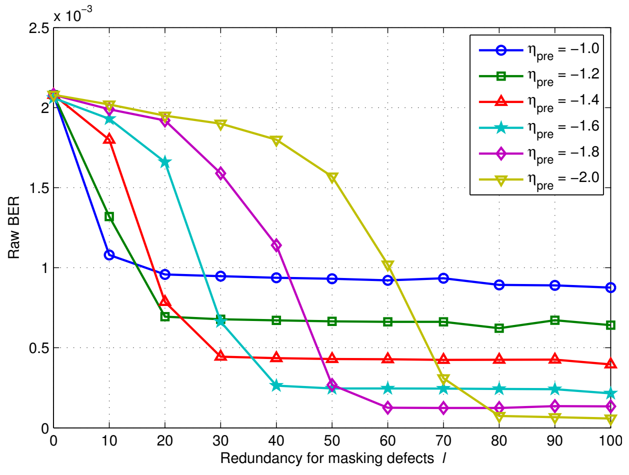
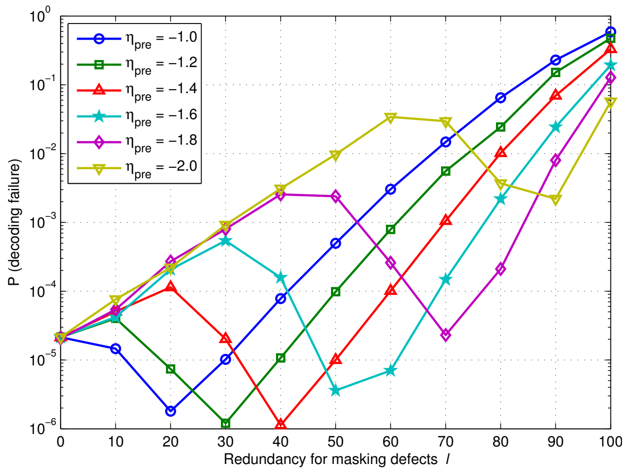


Fig. 9. The improvement of threshold voltage distribution by the proposed scheme (SLC, $n = 1023, k = 923, R = 0.90, \alpha = 1.2, \sigma_{Z_{\text{read}}} = 0.25$, and $\eta_{\text{pre}} = -1.4$).



(a) Raw BER



(b) Probability of decoding failure, $P(\widehat{\mathbf{m}} \neq \mathbf{m})$

Fig. 10. The improvement of raw BER and probability of decoding failure by the proposed scheme (SLC, $n = 1023, k = 923, R = 0.90, \alpha = 1.2$, and $\sigma_{Z_{\text{read}}} = 0.25$). If $l = 0$, then the side information is ignored, which is equivalent to the BCH code.

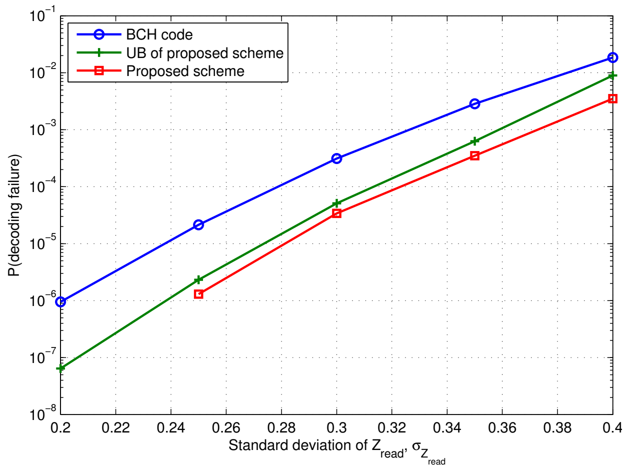
The improvement of threshold voltage distribution depends on the redundancy allocation (l, r) and the pre-read level η_{pre}. Fig. 10(a) shows this relation where raw bit error rates (BER) are plotted instead of threshold voltage distributions for clear comparison. We can observe that the raw BERs are reduced by allocating more redundancy l for masking defects. As shown in Fig. 9, the more redundancy l can lead to the improvement of threshold distributions.

Note that the improvement of raw BER levels off if the coding masks most of the cells regarded as stuck-at defects. Once the raw BER levels off, we should not waste more redundancy l for masking defects because the total redundancy $l + r = n - k$ is fixed. If r is too small, we cannot correct random errors due to the read noise Z_{read} well. For lower pre-read level η_{pre}, more cells are regarded as stuck-at defects, so more redundancy l for masking defects is required to mask stuck-at defects. If $l = 0$, then the pre-read level does not affect the raw BER. It is because the coding ignores the side information of defects for $l = 0$. Hence, the decoding failure probabilities $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ for $l = 0$ are the same for different pre-read levels.

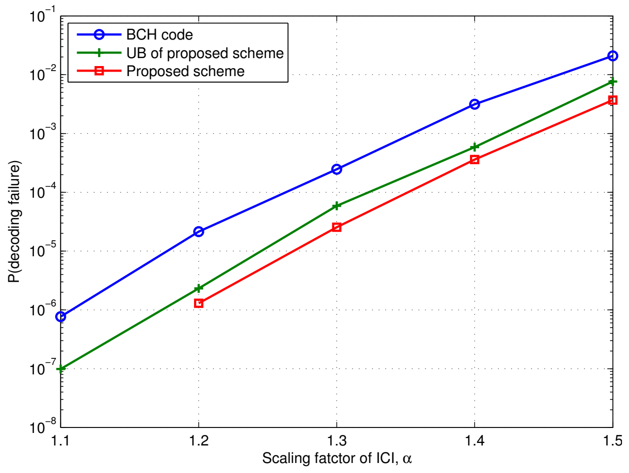
Fig. 10(b) shows that $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ is improved by the proposed scheme for the SLC flash memories. If $l > 0$, then the encoder uses the side information of defects to improve $P(\widehat{\mathbf{m}} \neq \mathbf{m})$. Note that $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ depends on the pre-read level η_{pre} since it controls the number of cells regarded as defects. In order to minimize $P(\widehat{\mathbf{m}} \neq \mathbf{m})$, we should choose the optimal redundancy allocation (l^*, r^*). For lower η_{pre}, more cells are regarded as stuck-at defects. More defects require more redundancy for l. For the given simulation results in Fig. 10(b), the pre-read level $\eta_{\text{pre}} = -1.4$ and $(l^*, r^*) = (40, 60)$ minimizes $P(\widehat{\mathbf{m}} \neq \mathbf{m})$, which was improved from 2.1×10^{-5} to 1.1×10^{-6} by using the side information of defects. It is worth mentioning that the pre-read levels and the redundancy allocations should be jointly optimized to minimize $P(\widehat{\mathbf{m}} \neq \mathbf{m})$.

Fig. 11 compares $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ of BCH codes and our proposed scheme based on PBCH codes where l was optimized for each channel parameters α and $\sigma_{Z_{\text{read}}}$ of SLC flash memories. Also, the upper bound on $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ by (16) is close to the simulation results. The raw BER and the empirical probability of stuck-at defects are used for p and β in (16), respectively. By using this upper bound, we can estimate $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ quickly without the computationally demanding Monte-Carlo simulations especially for very low $P(\widehat{\mathbf{m}} \neq \mathbf{m})$. Fig. 11(b) shows that the gain of scaling factor α by proposed scheme is around 0.1. We can claim that $\alpha \propto 1/D$ where D denotes the distances between flash memory cells because the parasitic capacitances are inversely proportional to D. Since the density of flash memories is proportional to $1/D^2$, we can claim that the density gain is about $19\% (= 1.2^2/1.1^2)$ for $P(\widehat{\mathbf{m}} \neq \mathbf{m}) \approx 10^{-6}$ in Fig. 11(b).

In Fig. 12, we compare $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ of BCH codes, proposed scheme based on PBCH codes, and LDPC codes for the SLC flash memories. LDPC codes with column weight four came from [44] and the read levels are chosen to maximize the mutual information [12]. For $P(\widehat{\mathbf{m}} \neq \mathbf{m}) \approx 10^{-5}$, the proposed scheme based on PBCH codes is comparable to LDPC code with 2 reads although the read speed performance is the same as LDPC code with 1 read. Since the read speed degradation is more critical than write speed degradation in many memory system applications, the proposed scheme has



(a) $\alpha = 1.2$ and different $\sigma_{Z_{\text{read}}}$



(b) $\sigma_{Z_{\text{read}}} = 0.25$ and different α

Fig. 11. Comparison of $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ (SLC, $n = 1023, k = 923$, and $R = 0.90$).

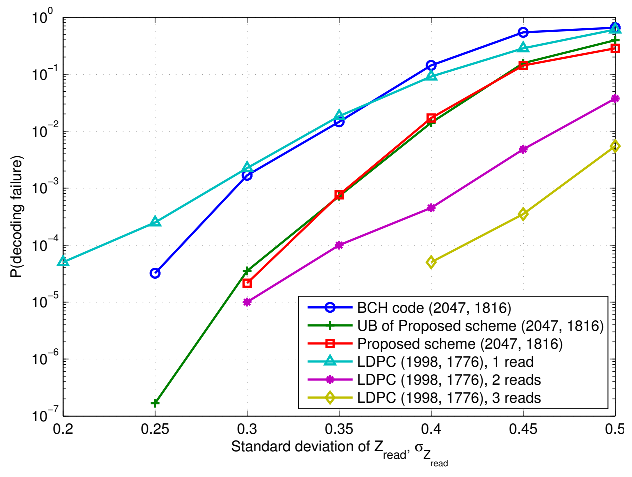
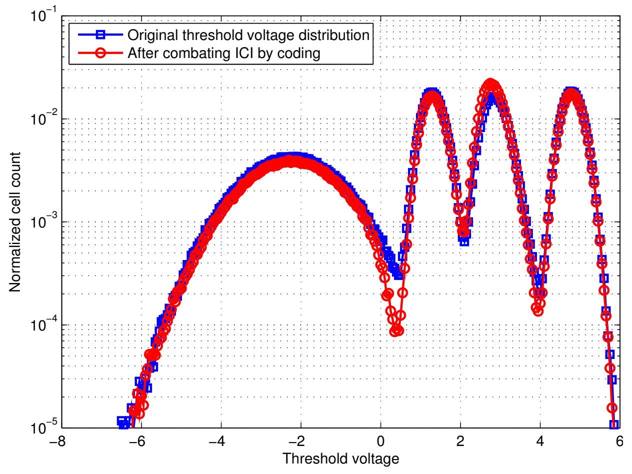
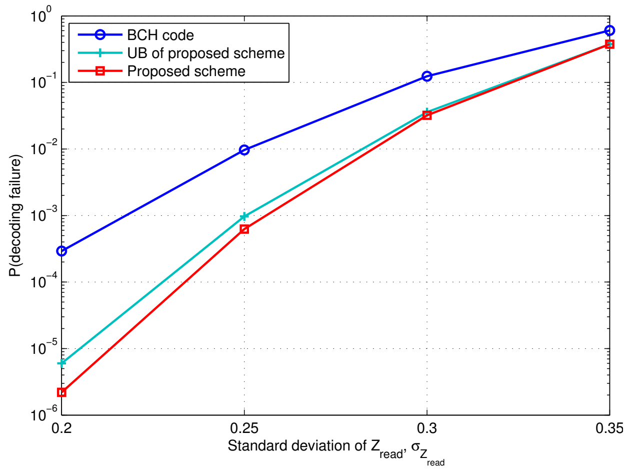


Fig. 12. Comparison of $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ for BCH codes, proposed scheme, and LDPC codes (SLC, $R = 0.89$, and $\alpha = 1.4$).



(a) The threshold voltage distribution ($\alpha = 1.0$ and $\sigma_{Z_{\text{read}}} = 0.2$)



(b) $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ of MSB page for different $\sigma_{Z_{\text{read}}}$

Fig. 13. The improvement of threshold voltage distribution and $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ (MLC, $n = 1023, k = 923, R = 0.90, \eta_{\text{pre},1} = -0.5$, and $\eta_{\text{pre},2} = 1.5$).

an advantage over soft decision decoding of LDPC codes. In addition, the proposed scheme based on PBCH codes does not suffer from error floor and can be estimated by the upper bound in (16). If we are willing to accept the read speed degradation, then the proposed scheme can be combined with LDPC codes, i.e., additive encoding LDPC codes.

Fig. 13(a) shows that the proposed scheme can improve the threshold voltage distribution of MLC flash memories, especially between S_0 and S_1 as shown in Fig. 8. Since this distribution improvement affects the MSB page data, the proposed coding scheme can effectively improve the decoding failure probability of the MSB page. Fig. 13(b) shows the improvement of $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ of the MSB page data by the proposed scheme.

VI. Conclusion

We proposed a coding scheme with side information to cope with the ICI of flash memories. Because of the unique properties of flash memory, the problem of combating the ICI can be cast as the coding problem of memory with defective cells. The proposed coding scheme improves the data reliability at the cost of decrease in write speed performance whereas soft decision decoding schemes degrade the read speed, which is practically useful in many memory applications. Theoretically, we bridge historic coding problems of writing on dirty paper and coding for memory with defective cells by leveraging the unique asymmetry property of flash memories. Although we focus on the ICI of flash memories, it could be important future work to take into account the program/erase cycles and the retention time and adaptively decide the read levels and pre-read levels.

References

[1] Y. Kim and B. V. K. Vijaya Kumar, "Writing on dirty flash memory," in *Proc. Allerton Conf. Commun., Control, Comput.*, Oct. 2014.

[2] Y. Kim, "Writing on dirty memory," Ph.D. dissertation, Dept. Elect. Comput. Eng., Carnegie Mellon Univ., Pittsburgh, PA, 2016.

[3] K. Prall, "Scaling non-volatile memory below 30nm," in *Proc. IEEE NVSMW*, Aug. 2007.

[4] 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.

[5] S. K. Park and J. Moon, "Characterization of inter-cell interference in 3d NAND flash memory," *IEEE Trans. Circuits Syst. I*, vol. 68, pp. 1183–1192, Mar. 2021.

[6] K. Prall and K. Parat, "25nm 64Gb MLC NAND technology and scaling challenges," in *IEDM Tech. Dig.*, Dec. 2010, pp. 5.2.1–5.2.4.

[7] K. Takeuchi, T. Tanaka, and T. Tanzawa, "A multipage cell architecture for high-speed programming multilevel NAND flash memories," *IEEE J. Solid-State Circuits*, vol. 33, no. 8, pp. 1228–1238, Aug. 1998.

[8] K.-T. Park *et al.*, "A zeroing cell-to-cell interference page architecture with temporary LSB storing and parallel MSB program scheme for MLC NAND flash memories," *IEEE J. Solid-State Circuits*, vol. 43, no. 4, pp. 919–928, Apr. 2008.

[9] N. Shibata *et al.*, "A 70 nm 16 Gb 16-level-cell NAND flash memory," *IEEE J. Solid-State Circuits*, vol. 43, no. 4, pp. 929–937, Apr. 2008.

[10] Y. Li *et al.*, "A 16 Gb 3-bit per cell (x3) NAND flash memory on 56 nm technology with 8 MB/s write rate," *IEEE J. Solid-State Circuits*, vol. 44, no. 1, pp. 195–207, Jan. 2009.

[11] G. Dong, N. Xie, and T. Zhang, "On the use of soft-decision error-correction codes in NAND flash memory," *IEEE Trans. Circuits Syst. I*, vol. 58, no. 2, pp. 429–439, Feb. 2011.

[12] J. Wang *et al.*, "Enhanced precision through multiple reads for LDPC decoding in flash memories," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 5, pp. 880–891, May 2014.

[13] G. Dong, S. Li, and T. Zhang, "Using data postcompensation and predistortion to tolerate cell-to-cell interference in MLC NAND flash memory," *IEEE Trans. Circuits Syst. I*, vol. 57, no. 10, pp. 2718–2728, Oct. 2010.

[14] M. Asadi, X. Huang, A. Kavcic, and N. P. Santhanam, "Optimal detector for multilevel NAND flash memory channels with intercell interference," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 5, pp. 825–835, May 2014.

[15] A. Berman and Y. Birk, "Constrained flash memory programming," in *Proc. IEEE ISIT*, St. Petersburg, 2011, pp. 2128–2132.

[16] Y. Kim *et al.*, "Modulation coding for flash memories," in *Proc. IEEE ICNC*, Jan. 2013.

[17] M. Qin, E. Yaakobi, and P. H. Siegel, "Constrained codes that mitigate inter-cell interference in read/write cycles for flash memories," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 5, pp. 836–846, May 2014.

[18] Y. M. Chee *et al.*, "Capacity-achieving codes that mitigate intercell interference and charge leakage in flash memories," *IEEE Trans. Inf. Theory*, vol. 65, pp. 3702–3712, Jun. 2019.

[19] S. Buzaglo and P. H. Siegel, "Row-by-row coding schemes for inter-cell interference in flash memory," *IEEE Trans. Commun.*, vol. 65, no. 10, pp. 4101–4113, Oct. 2017.

[20] A. Hareedy, S. Zheng, P. Siegel, and R. Calderbank, "Read-and-run constrained Coding for modern flash devices," 2021. [Online]. Available: https://arxiv.org/abs/2111.07415

[21] M. H. M. Costa, "Writing on dirty paper," *IEEE Trans. Inf. Theory*, vol. 29, no. 3, pp. 439–441, May 1983.

[22] A. V. Kuznetsov and B. S. Tsybakov, "Coding in a memory with defective cells," *Probl. Peredachi Inf.*, vol. 10, no. 2, pp. 52–60, Apr.–Jun. 1974.

[23] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 3rd ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002.

[24] A. Jagmohan, L. A. Lastras-Montano, M. M. Franceschini, M. Sharma, and R. Cheek, "Coding for multilevel heterogeneous memories," in *Proc. IEEE ICC*, May 2010.

[25] G.-C. Zhu, W. Yu, and F. R. Kschischang, "Finite-geometry low-density parity-check codes for channels with stuck-at defects," in *Proc. CWIT*, Jun. 2005.

[26] E. Hwang, R. Negi, and B. V. K. Vijaya Kumar, "Additive encoding low-density parity-check (AE-LDPC) codes for two-dimensional magnetic recording (TDMR)," in *Proc. IEEE ICNC*, Feb. 2012.

[27] K.-D. Suh *et al.*, "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.

[28] K.-T. Park *et al.*, "Three-dimensional 128 Gb MLC vertical NAND flash memory with 24-WL stacked layers and 50 MB/s high-speed programming," *IEEE J. Solid-State Circuits*, vol. 50, no. 1, pp. 204–213, Jan. 2015.

[29] C. Heegard, "Partitioned linear block codes for computer memory with "stuck-at" defects," *IEEE Trans. Inf. Theory*, vol. 29, no. 6, pp. 831–842, Nov. 1983.

[30] B. S. Tsybakov, "Additive group codes for defect correction," *Probl. Peredachi Inf.*, vol. 11, no. 1, pp. 111–113, Jan.–Mar. 1975.

[31] Tsybakov and B. Solomonovich, "Defect and error correction," *Probl. Peredachi Inf.*, vol. 11, no. 3, pp. 21–30, Jul.–Sep. 1975.

[32] A. V. Kuznetsov, T. Kasami, and S. Yamamura, "An error correcting scheme for defective memory," *IEEE Trans. Inf. Theory*, vol. 24, no. 6, pp. 712–718, Nov. 1978.

[33] J. M. Borden and A. J. Vinck, "On coding for 'stuck-at' defects," *IEEE Trans. Inf. Theory*, vol. 33, no. 5, pp. 729–735, Sep. 1987.

[34] I. I. Dumer, "Asymptotically optimal linear codes correcting defects of linearly increasing multiplicity." *Probl. Peredachi Inf.*, vol. 26, no. 2, pp. 93–104, Apr.–Jun. 1990.

[35] E. Hwang, B. Narayanaswamy, R. Negi, and B. V. K. Vijaya Kumar, "Iterative cross-entropy encoding for memory systems with stuck-at errors," in *Proc. IEEE GLOBECOM*, Dec. 2011.

[36] Y. Kim and B. V. K. Vijaya Kumar, "Coding for memory with stuck-at defects," in *Proc. IEEE ICC*, Jun. 2013.

[37] H. Mahdavifar and A. Vardy, "Explicit capacity achieving codes for defective memories," in *Proc. IEEE ISIT*, Jun. 2015.

[38] Y. Kim and B. V. K. Vijaya Kumar, "Redundancy allocation in finite-length nested codes for nonvolatile memories," *J. Commun. Netw.*, vol. 20, no. 4, pp. 354–365, Aug. 2018.

[39] Y. Kim and B. V. K. Vijaya Kumar, "Redundancy allocation of partitioned linear block codes," in *Proc. IEEE ISIT*, Jul. 2013.

[40] J. Moon *et al.*, "Statistical characterization of noise and interference in NAND flash memory," *IEEE Trans. Circuits Syst. I*, vol. 60, no. 8, pp. 2153–2164, Aug. 2013.

[41] M. Helm *et al.*, "A 128 Gb MLC NAND-flash device using 16 nm planar cell," in *Proc. IEEE ISSCC*, Feb. 2014.

[42] T. Parnell, N. Papandreou, T. Mittelholzer, and H. Pozidis, "Modelling of the threshold voltage distributions of sub-20nm NAND flash memory," in *Proc. IEEE GLOBECOM*, Dec. 2014.

[43] B. Peleato, R. Agarwal, J. M. Cioffi, M. Qin, and P. H. Siegel, "Adaptive read thresholds for NAND flash," *IEEE Trans. Commun.*, vol. 63, no. 9, pp. 3069–3081, Sep. 2015.

[44] D. J. C. Mackay, (2009). *Encyclopedia of sparse graph codes* [Online]. Available: http://www.inference.phy.cam.ac.uk/mackay/codes/data.html

Yongjune Kim received the B.S. (Hons.) and M.S. degrees in Electrical and Computer Engineering from Seoul National University, Seoul, South Korea, in 2002 and 2004, respectively, and the Ph.D. degree in Electrical and Computer Engineering from Carnegie Mellon University, Pittsburgh, PA, USA, in 2016. He is currently an Assistant Professor in the Department of Electrical Engineering at Pohang University of Science and Technology (POSTECH), Pohang, South Korea. From 2020 to 2022, he was an Assistant Professor in the Department of Electrical Engineering and Computer Science at Daegu Gyeongbuk Institute of Science and Technology (DGIST), Daegu, South Korea. From 2018 to 2020, he was a Researcher at Western Digital Research, Milpitas, CA, USA. From 2016 to 2018, he was a Postdoctoral Scholar in the Coordinated Science Laboratory at the University of Illinois at Urbana-Champaign, Urbana, IL, USA. His research interests include machine learning, coding theory, and information theory. He was a recipient of the IEEE Data Storage Best Student Paper Award, the Best Paper Award of the 2016 IEEE International Conference on Communications (ICC), the Best Paper Award (honorable mention) of the 2018 IEEE International Symposium on Circuits and Systems (ISCAS), and the Best Paper Award of the 31st Samsung Semiconductor Technology Symposium.

Euiseok Hwang received his B.S. and M.S. degrees from School of Engineering at Seoul National University, Seoul, Korea in 1998 and 2000, and also received M.S. and Ph.D. degrees in Electrical and Computer Engineering from Carnegie Mellon University (CMU), Pittsburgh, PA in 2010 and 2011. He worked for Digital Media Research Center at Daewoo Electronics Co., Ltd. in Korea from 2000 to 2006 and for Channel Architecture Group at LSI Corp. (now Broadcom) in San Jose, CA, from 2011 to 2014. Since 2015, he has been an Assistant/Associate Professor of School of Electrical Engineering and Computer Science/AI Graduate School/School of Mechatronics at Gwangju Institute of Science and Technology (GIST), Korea. From 2021 to 2022, he worked as a Visiting Scholar in the Department of Computer Science and Engineering at University of Michigan, Ann Arbor. His research interests include statistical signal processing, machine learning and channel coding for data storage and communication systems, and their emerging ICT applications such as IoT and smart grids.

B. V. K. Vijaya Kumar is the U.A. & Helen Whitaker Professor of Electrical and Computer Engineering at Carnegie Mellon University (CMU), Pittsburgh, USA. He was the Director of CMU-Africa (Kigali, Rwanda) during 2018-2021. Prior to going to Rwanda, Prof. Kumar was the Associate Dean as well as the interim Dean for the CMU College of Engineering. Prof. Kumar has made pioneering research contributions to computer vision and pattern recognition methods, with applications in biometric recognition and autonomous driving. He has received many research awards including the 2008 Outstanding Faculty Research Award from the College of Engineering at CMU. His publications include a book, 24 book chapters, 15 patents, more than 410 conference papers and 210 journal papers. In recognition of his technical and professional contributions, he has been elected as a Fellow of Optica (formerly Optical Society of America), SPIE (International Society for Optical Engineering), IEEE (Institute of Electrical and Electronics Engineering), IAPR (International Association of Pattern Recognition), NAI (National Academy of Inventors) and AAAS (American Association for Advancement of Science).