Flexible WOM Codes for NAND Flash Memory Based on Raptor-Like Codes

Bohwan Jun, Heeyoul Kwak, Jong-Seon No, and Hosung Park

Abstract: **Write-once memory (WOM) codes aim to extend the lifetime and improve the writing efficiency of storage devices such as NAND flash memory by reducing the number of erase operations. In this paper, a new rewriting scheme for NAND flash memory is proposed, which supports two writes (or only one rewrite) and allows the second write incrementally done multiple times by using raptor-like codes as rate-compatible (RC) low-density generator matrix (LDGM) codes. The proposed scheme improves writing efficiency of the NAND flash memory when combined with a proper page selection method. It is verified via numerical analysis that the proposed WOM codes outperform the conventional WOM codes in terms of writing efficiency.**

Index Terms: **Binary erasure quantization (BEQ) codes, low-density generator matrix (LDGM) codes, NAND flash memory, raptor-like codes, write-once memory (WOM) codes.**

I. INTRODUCTION

NAND flash memory has been widely used in various storage systems from mobile devices to high-end server systems owing to its outstanding features such as high storage density, low power consumption, fast accessibility, and random read/write behavior. In particular, NAND flash-based solid-state drives (SSDs) have become one of the most popular storage devices. By injecting a fixed amount of charge into the floating gate of a cell, a bit information is stored in case of single-level cell (SLC) flash memory. In order to increase the storage density, more than one bit data can be stored in one cell by injecting multiple amounts of charge. Multi-level cell (MLC) and triple-level cell flash memories store two bits and three bits in a cell, respectively. Unfortunately, flash memory suffers from wearing out phenomenon due to frequent program (write) and erase (P/E) operations, which degrades reliability and thus limits the lifetime of the NAND flash memory [1]. In order to mitigate the reliability degradation of the NAND flash memory, error correction codes can be a good solution [2]–[5]. As another approach to improve its reliability, rewriting schemes were recently proposed to reduce the number of erase operations [6]–[9].

Write-once memory (WOM) codes are originally introduced by Rivest and Shamir [10] for WOM devices such as punch cards and compact discs. Their capacities are studied in [11] and they can be used for other applications [12]. NAND flash memory is a kind of WOM device because if the amount of charge in a cell is once increased through write operations, it cannot decrease any more until erase operation is performed. Thus, WOM codes in the NAND flash memory allow multiple writes on the same cells without invoking block erase operation. Recently, low-complexity WOM codes for the NAND flash memory are constructed by using polar codes [13] and low-density generator matrix (LDGM) codes [14]. In particular, Gad *et al.* [14] regarded encoding of WOM code as solving a binary erasure quantization (BEQ) problem by LDGM codes with a simple message passing algorithm. However, the encoding (writing) by the WOM code will fail if the probability that cells in an invalid page remain unprogrammed is lower than the rate of rewriting code, where the invalid page means the one waiting for the rewriting or the block erase operation. In that case, the corresponding page cannot be reused for the second write and it should be erased.

In this paper, a flexible rewriting scheme with WOM codes is proposed for NAND flash memory to support two writes using raptor-like codes as an instance of rate-compatible (RC) LDGM codes to implement BEQ codes. At the second write, invalid pages can be reused more efficiently by adjusting the rate of rewriting code than the conventional WOM code in [14]. The RC-LDGM codes are constructed from protographs with desired rates, each of which is optimized by computer search similar to the method in [15]. In addition, we propose a page selection scheme that distinguishes rewritable pages from the predetermined candidate pages for the second write. Since the encoder of WOM code is able to recognize whether a new message can be written over the selected pages or not, invalid pages in the block can be efficiently reused because combinations of the selected pages and their ordering can be controlled.

The remainder of the paper is organized as follows. Section II overviews the characteristics of NAND flash memory, existing rewriting schemes for flash memory, and the construction of rewriting codes with BEQ codes. Then a new rewriting scheme for NAND flash memory is proposed in Section III. Section IV gives a construction of raptor-like codes for the proposed WOM codes and the performance of the proposed WOM codes is verified via numerical analysis in Section V. Finally, the conclusion is provided in Section VI.

Manuscript received September 29, 2017 approved for publication by Sang-Hyo Kim, Division I Editor, February 20, 2018.

This research was supported by Samsung Electronics, Co., Ltd. (0423-20160054) and also by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2015R1D1A1A01060941).

B. Jun is with the Samsung Electronics, Co., Ltd., Hwasung, Gyeonggi-do, 18448, Korea, email: netjic@gmail.com.

H. Kwak and J.-S. No are with the Department of ECE, INMC, Seoul National University, Seoul 08826, Korea, email: ghy1228@ccl.snu.ac.kr, jsno@snu.ac.kr.

H. Park is with the School of Electronics and Computer Engineering, Chonnam National University, Gwangju 61186, Korea, email: hpark1@jnu.ac.kr.

H. Park is the corresponding author.

Digital Object Identifier: 10.1109/JCN.2018.000020

II. PRELIMINARIES

A. NAND Flash Memory

First, physical characteristics of NAND flash memory are reviewed. In the NAND flash memory, charge can be injected into the floating gate in a cell, called *program operation* and removed from the cell by *erase operation*. When charge is captured in the floating gate, the transistor will be turned on if a voltage above threshold is applied at the control gate. If there is no charge in the floating gate, then the transistor cannot be turned on with the same voltage. Thus, states of the cell can be distinguished by measuring current between source and drain, which is called *read operation*.

The program and erase functions are asymmetrical, which is an inherent property of NAND flash memory. A unit of flash memory for reading and programming operations is called *page*, which is composed of a large number of cells. The size of page may vary from 1 KB to 16 KB. On the other hand, a *block* is a unit of the erase operation and it may contain 16–512 pages. Note that all pages in a block should be erased at once before writing new messages, which causes wearing out problem of the flash memory [16]. Hence, as the number of P/E cycles increases, raw bit error rate of the memory device increases, which results in the limitation of device lifetime. Therefore, reducing the number of erase operations is one of key approaches to extend the lifetime of the flash memory devices.

B. Rewriting Schemes for Flash Memory

Recently, Yadgar *et al.* [8] proposed a rewriting scheme with polar code for the NAND flash memory, called a reusable SSD, in which invalid pages are reused for the second write. At the first write, two messages $\mathbf{m}_{1,1}$ and $\mathbf{m}_{1,2}$ are written in two clean pages without coding. When the messages are updated, the erase operations are not invoked immediately but only the status of the corresponding pages is changed to *invalid*. While the pages wait for block erase operation in the conventional scheme, the reusable SSD takes advantage of invalid pages for rewriting. In other words, the block erasure is not invoked and a new message $\mathbf{m}_2$ of size equal to one page for the second write is encoded by the WOM encoder and rewritten over the two invalid pages. After that, the status of the pair of pages becomes valid again. If we want to update $\mathbf{m}_2$ once again, the status of the pair is changed to 'invalid' and then these pages wait for the block erase operation. Finally, the block is erased when the number of invalid pages exceeds the predetermined value. Hence the length of P/E cycle in a block is increased from 3 to 5, which is shown in Fig. 1, and it is known that the number of erase operations is reduced by 33% in most cases of SLC flash-based SSD.

Generally, the low-high programming [17] is used to avoid programming interference for MLC NAND flash memory, i.e., low page (bit) is programmed first and then high page (bit) is programmed. Unfortunately, it is not easy to adopt WOM codes to both pages in MLC flash memory due to the following two reasons. First, the bit of high page which increases the charge level of cell depends on that of low page. More precisely, if the bit of low page is '1', then '0' in the high page increases the cell level, while '1' in the high page increases the cell level when the bit is '0' in the low page. Second, some transitions

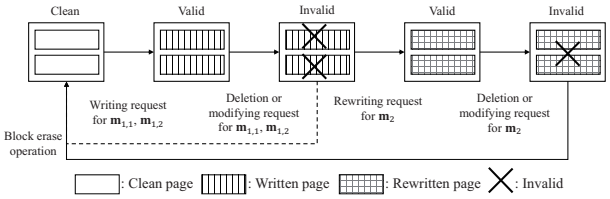

Fig. 1. Cycles of the rewriting scheme in a block consisting of two pages [8].

cannot be implemented in MLC flash memory even though they are not violating the WOM constraint for alphabet size four. To resolve this drawback, the low-low-high (LLH) programming is suggested in [17] and [18], which reuses only low pages. It is shown that the rewriting with LLH programming scheme gives 20% reduction in erase operations and hence this will be considered for the proposed WOM coding scheme.

C. Construction of Rewriting Codes Using BEQ Codes

In this subsection, we overview BEQ codes [19] which are used for the construction of rewriting codes.

Definition 1: A binary erasure channel (BEC) function $BEC : \{0,1\}^n \times \{0,1\}^n \to \{0,1,*\}^n$ is defined as follows. Let $\mathbf{z}$ and $\mathbf{v}$ be a binary vector and an erasure location vector of length n, respectively. For $i = 1, \cdots, n$, the ith component of $\mathbf{r} = BEC(\mathbf{z}, \mathbf{v})$ is given as

$$r_i = BEC(\mathbf{z}, \mathbf{v})_i = \begin{cases} z_i, & \text{if } \mathrm{v_i} = 0; \\ *, & \text{if } \mathrm{v_i} = 1, \end{cases}$$

where $*$ denotes an erasure.

Definition 2: Let $D(\mathbf{r}, \hat{\mathbf{r}})$ be a distortion function between an n-tuple binary vector with erasures $\mathbf{r}$ and another n-tuple binary vector $\hat{\mathbf{r}}$. Then we have

$$D(\mathbf{r}, \hat{\mathbf{r}}) = \sum_{i=1}^{n} d(r_i, \hat{r}_i),$$

where d_i is given as

$$d(r_i, \hat{r}_i) = \begin{cases} 1, & \text{if } r_i \neq *, r_i \neq \hat{r}_i; \\ 0, & \text{otherwise.} \end{cases}$$

In [19], the authors introduced a binary erasure quantizer $Q(\mathbf{r})$ which is defined by a binary linear code C_Q with a generator matrix $\mathbf{G}_Q$ and a parity-check matrix $\mathbf{H}_Q$. The binary erasure quantizer $Q(\mathbf{r})$ compresses the vector $\mathbf{r}$ into the vector $\mathbf{u}$ without any distortion such that $D(\mathbf{r}, \mathbf{u}\mathbf{G}_Q) = 0$, where $\mathbf{u}\mathbf{G}_Q$ is a valid codeword in C_Q. Here, C_Q is called a BEQ code. It is known that an LDGM code, a dual of LDPC code, can be used as a BEQ code. Clearly, BEQ problem with LDGM code is dual of BEC problem with LDPC code, which can be solved by the well-known simple message passing algorithm, called belief propagation (BP) decoding in the erasure channel. Thus, computational complexity of the binary erasure quantizer is linear in the code length.

In [14], the authors proposed a rewriting code C_R which is a collection of all cosets of BEQ code C_Q in $\mathbb{F}_2^n$. In order to rewrite

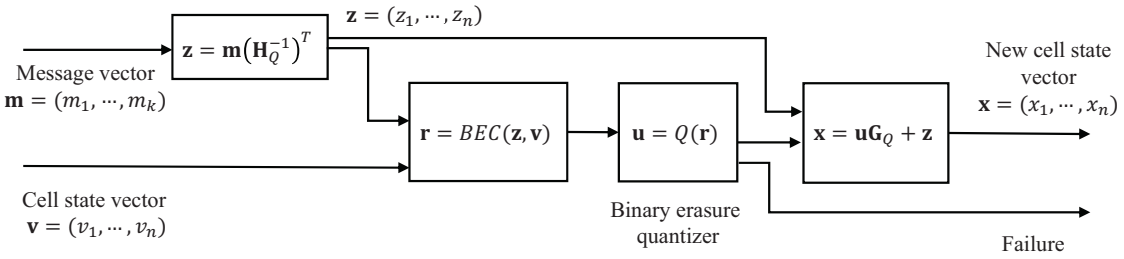

Fig. 2. Block diagram of rewriting (encoding) process by a BEQ code.

an arbitrary k-bit message $\mathbf{m}$ onto the given n cells, it is transformed into n-tuple binary vector $\mathbf{z} = \mathbf{m}\left(\mathbf{H}_Q^{-1}\right)^\intercal$, where $\mathbf{H}_Q^{-1}$ is called a transform matrix which satisfies $\mathbf{H}_Q\mathbf{H}_Q^{-1} = \mathbf{I}_k$ and $(\cdot)^\intercal$ denotes the transpose. Then $\mathbf{z}$ is converted to the vector $\mathbf{r}$ by the BEC function with the given cell state vector $\mathbf{v}$, i.e., erasure location vector. The vector $\mathbf{r}$ is compressed via the binary erasure quantizer $Q(\cdot)$ and then the corresponding codeword $\mathbf{u}\mathbf{G}_Q$ and $\mathbf{z}$ are added to make a rewriting vector (new cell state vector) $\mathbf{x}$, which is a codeword of WOM code. The block diagrams of rewriting and reconstruction are shown in Figs. 2 and 3.

Here, the code rates of the first and the second writes are $R_1 = 1$ and $R_2 = k/n$, respectively. Thus, the sum-rate is $R_{\text{sum}} = R_1 + R_2 = 1 + k/n$. It is known that when R_1 is fixed to 1, the capacity of two-write WOM code is $(1, 0.5)$. Thus, if we use a parity-check matrix of capacity-achieving LDPC codes with half rate over binary erasure channel as a generator matrix of BEQ code, the rewriting capacity can be asymptotically achieved.

III. PROPOSED REWRITING CODES

A. System Model

In this paper, we focus on two writes on SLC flash memory and further extend it to MLC flash memory using the LLH programming scheme. It is considered that there are L logical pages in a block and the size of each page is k bits, which corresponds to a message vector. At the first write, L message vectors are written through L pages without coding. After the first write, the cell state is assumed to be i.i.d. and ϵ denotes the probability that cell is in initial (unprogrammed) state, i.e., $\Pr[v = 1] = \epsilon$. It is also assumed that the second writing is invoked after sufficient time is elapsed from the first write, which means that some preprocessing for the second write can be done before the request of the second write. Moreover, if an LDGM code is applied as a BEQ code in practical systems, there exists performance degradation from the asymptotic results due to its finite length. Thus, some fraction of another page is added to the codeword for the second write, which slightly reduces the rewriting rate. Therefore, two invalid pages and extra α page are used for each rewriting request, where $0 < \alpha < 1$.

B. Multi-rate Rewriting Codes

In [14], only a single LDGM code is used as a linear BEQ code, which implies a fixed rewriting code rate. Thus, when the portion of 1's (initial or unprogrammed states) in a cell state vector is below target $\hat{\epsilon}$, the new message cannot be written over the cell state vector. Therefore, those pages will not be used for rewriting and have to be thrown away. When the binary erasure quantizer in the encoder is not able to find any valid codeword in the BEQ code for a given cell state vector, the encoder of WOM code declares encoding failure. To overcome this, we could modify encoding procedure by extending the codeword length using additional invalid pages, which reduces the rewriting code rate.

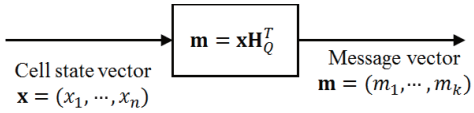

Fig. 3. Block diagram of reconstruction (decoding) process by a BEQ code.

In this paper, a new encoding scheme using $\beta + 1$ multi-rate LDGM codes as BEQ codes is proposed, that is, $C_Q^{(0)}, C_Q^{(1)}, \cdots, C_Q^{(\beta)}$, where β is a nonnegative integer. At first, the WOM encoder tries to rewrite k-tuple binary message vector $\mathbf{m}$ over a given cell state vector $(p^{(0)}, p^{(1)}, p^{(2)})$. Here, $p^{(0)}$ is α fractional invalid page while $p^{(1)}$ and $p^{(2)}$ are the l_1th and the l_2th whole invalid pages, respectively for $1 \le l_1, l_2 \le L$ and $l_1 \ne l_2$. In other words, the encoder searches for a valid codeword in $C_Q^{(0)}$ which does not have any distortion in the $(2 + \alpha)k$-tuple binary vector with erasures formed by the given cell state vector. When the encoder declares failure, we use $C_Q^{(1)}$ instead of $C_Q^{(0)}$ and the length of cell state vector is increased to $(3 + \alpha)k$ by adding another invalid page in the block. After that, we try to encode the same message $\mathbf{m}$ using the BEQ code $C_Q^{(1)}$. If the encoder fails again, then the same procedures are repeated by incrementally lengthening the cell state vector and lowering the rate of BEQ code through $C_Q^{(2)}, \cdots, C_Q^{(\beta)}$. Note that a cell state vector is divided into $\beta + 3$ parts for $C_Q^{(\beta)}$, i.e., a cell state vector can be represented as

$$\mathbf{v} = \left(p^{(0)}, p^{(1)}, p^{(2)}, \cdots, p^{(\beta+2)}\right),$$

where $p^{(0)}$ and $p^{(i)}$ are the α fractional page and the l_ith whole page in the block, respectively for $i = 1, \cdots, \beta + 2$ and $1 \le l_i \le L$. Clearly, the rewriting code rate of $C_Q^{(i)}$ is $R_{2,i} = \frac{1}{2+\alpha+i}$ for $i = 0, \cdots, \beta$. An example of cell state vector with $\alpha = 0.125$ and $\beta = 0$ is shown in Fig. 4. Since the second write of the proposed WOM code is incrementally done with multiple BEQ codes, i.e., the rewriting rate is adjusted flexibly, the invalid pages in the block can be reused more efficiently.

Definition 3: If the encoder succeeds in rewriting over a given cell state vector, then the cell state vector is said to be *rewritable*.

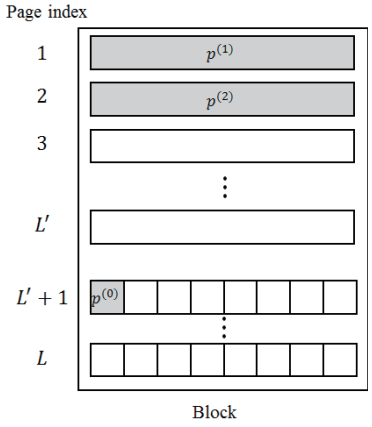

Fig. 4. An example of cell state vector $\mathbf{v} = \left(p^{(0)}, p^{(1)}, p^{(2)}\right)$.

Now, one of performance criteria for NAND flash memory with WOM code, called writing efficiency, is defined as follows.

Definition 4: Assume that L pages are in a block and t writes are allowed. Then *writing efficiency* is defined as

$$\eta = \frac{\sum_{i=1}^{t} W_i}{L},$$

where W_i is the expected number of message vectors written through the whole block at the ith write.

In this scenario, we only consider that two writes are used, i.e., $t = 2$, and $\beta + 1$ multi-rate BEQ codes are applied at the second writing. Thus, we have

$$W_2 = \sum_{i=0}^{\beta} W^{(i)},$$

where $W^{(i)}$ is the expected number of message vectors written by the ith BEQ code $C_Q^{(i)}$ at the second writing. Note that the best case is when all rewritings are successful with the highest rate. In other words, only $2 + \alpha$ pages are required for each message vector. As a result, the maximum number of message vectors for rewriting is $W_{max} = \left\lfloor \frac{L}{2+\alpha} \right\rfloor$ and we have

$$\sum_{i=0}^{\beta} W^{(i)}(2 + i) + \lceil \alpha W_{\max} \rceil \leq L.$$

C. New Rewriting Code with Page Selection

In this subsection, a new page selection scheme in a block is proposed for encoding of the WOM code. Rewritable cell state vectors can be efficiently selected from L invalid pages in a block by the page selection scheme and thus the writing efficiency can be enhanced. Assume that the status of the lth page is changed from 'valid' to 'invalid', for $1 \leq l \leq L'$, where $L' = L - \lceil \alpha W_{\max} \rceil$.

Remark 1: The 1s in a cell state vector $\mathbf{v} = (p^{(0)}, p^{(1)}, p^{(2)}, \cdots, p^{(\beta+2)})$ will be changed to erasures after $\mathbf{r} = BEC(\mathbf{z}, \mathbf{v})$ in Fig. 2. The quantization is done via the message-passing decoding on the bipartite graph constructed from the generator matrix $\mathbf{G}_Q$ of the BEQ code and its failure occurs when the erasure pattern in $\mathbf{r}$ includes any stopping sets in the bipartite graph. Hence, a desirable condition for a cell state vector to be rewritable is that the locations of 1s does not contain any local stopping sets in each part $p^{(i)}$.

It is noted that the condition can be easily checked by the message passing algorithm. For the page selection, we construct a temporary cell state vector $\mathbf{v}_{\langle l \rangle, i}$ for the lth page as

$$\mathbf{v}_{\langle l \rangle, i} = \left(\mathbf{1}_{\alpha k}, p_i^{(1)}, p_i^{(2)}, \cdots, p_i^{(\beta+2)}\right),$$

where $p_i^{(j)}$ is a k-tuple binary vector and

$$p_i^{(j)} = \begin{cases} p_l, & \text{if } j = i; \\ \mathbf{1}_k, & \text{otherwise} \end{cases}$$

for $i, j = 1, \cdots, \beta + 2$ and p_l is the k-tuple binary vector written at the first write in the lth page. Here, the corresponding erased vector is given as

$$\mathbf{r}_{\langle l \rangle, i} = BEC(\mathbf{0}_n, \mathbf{v}_{\langle l \rangle, i}),$$

where $n = (2 + \alpha + \beta)k$ and $\mathbf{1}_n$ and $\mathbf{0}_n$ denote all one and all zero vectors with length n, respectively. Then the quantization algorithm is applied to each erased vector $\mathbf{r}_{\langle l \rangle, i}$. Finally, a $(\beta+2)$-tuple vector $\mathbf{q}_{\langle l \rangle} = (q_{\langle l \rangle, 1}, \cdots, q_{\langle l \rangle, \beta+2})$ is given as

$$q_{\langle l \rangle, i} = \begin{cases} 0, & \text{if } Q(\mathbf{r}_{\langle l \rangle, i}) = \text{failure}; \\ 1, & \text{otherwise} \end{cases}$$

for $i = 1, \cdots, (\beta + 2)$ and the quantization $Q(\mathbf{r}_{\langle l \rangle, i})$ is performed with BEQ code $C_Q^{(\beta+2)}$.

Let θ be the maximum trial number for each BEQ code $C_Q^{(i)}$ for rewriting, where $i = 0, \cdots, \beta$. Initially, i is set to 0 and then $p^{(0)}$ and $p^{(j)}$ are randomly selected from the predetermined fractional page and invalid pages for $j = 1, \cdots, \beta + 2$. It is assumed that the corresponding index of $p^{(0)}$ is l_0, where $1 \leq l_0 \leq W_{\max}$ and the binary vector is denoted by $p_{l_0}^{\star}$. Similarly, we have the page index l_j and the binary vector p_{l_j} for $p^{(j)}$, where $1 \leq l_j \leq L'$. From Remark 1, the selected page $p^{(j)}$ should satisfy $q_{\langle l_j \rangle, j} = 1$. Finally, the BEC function and quantization with $C_Q^{(i)}$, described in Algorithm 1, are applied to the constructed cell state vector $\mathbf{v} = \left(p^{(0)}, p^{(1)}, p^{(2)}\right)$. A valid output $\mathbf{u}$ is returned if the quantization is successful. On the other hand, if the quantization fails, the encoder increases i by one and the previous steps are repeated, that is, fragments of cell state vector are selected again using $\left(\mathbf{q}_{\langle l \rangle}, \cdots, \mathbf{q}_{\langle \beta+2 \rangle}\right)$. The overall procedure of the proposed scheme is described in Algorithm 2.

It is clear that as either θ or β increases, the computational complexity also increases. However, it is reasonably assumed that there is enough time to prepare for the second write. Thus, constructing cell state vector can be done before the second write, which mitigates latency problem for the second write while improving the writing efficiency of the NAND flash memory.

Algorithm 1 Quantization with BEQ code [14]

Input: $\mathbf{m}, \mathbf{v}, C_Q$
1: $\mathbf{z} \leftarrow \mathbf{m}\left(\mathbf{H}_Q^{-1}\right)^{\mathsf{T}}$
2: $\mathbf{r} \leftarrow BEC(\mathbf{z}, \mathbf{v})$
3: $\mathbf{u} \leftarrow Q(\mathbf{r})$
4: **if** $D(\mathbf{r}, \mathbf{u}\mathbf{G}_Q) = 0$ **then**
5: $\mathbf{x} \leftarrow \mathbf{u}\mathbf{G}_Q + \mathbf{z}$
6: **return** $\mathbf{x}$
7: **else**
8: **return** failure
9: **end if**

IV. RAPTOR-LIKE CODES FOR PROPOSED REWRITING CODES

It is well known that a generator matrix of LDGM code is the same as the parity-check matrix of the dual LDPC code. Thus, construction of good LDGM code is equivalent to that of LDPC code, which has been intensively studied for decades. Particularly, a protograph-based construction of a subclass of multi-edge-type LDPC codes has been introduced and shown in [20] to have excellent error correcting performance and low decoding complexity. In order to implement multi-rate structured LDPC codes, RC-LPDC codes have attracted attention. If codewords of higher rate codes are embedded into the codewords of lower rate codes, those codes are said to have RC property. Recently, the construction of protograph-based RC-LDPC codes by extending and puncturing approaches has been studied in [15], [21]–[25]. Generally, it is done by exhaustive computer search, where limiting search space is a major concern. In [15], a raptor-like structure has been adopted to RC-LDPC codes and it is shown that the corresponding codes have outstanding performance while providing wide rate-compatibility.

Similarly, we construct protograph-based raptor-like RC-LDGM codes by computer search with the protograph extrinsic information transfer (PEXIT) analysis over BEC [26] and thus the protograph which has the best BP threshold is selected from all candidates. Note that the required set of code rates for the proposed rewriting scheme is $\left\{\frac{1}{2+\alpha}, \cdots, \frac{1}{2+\alpha+\beta}\right\}$. Since the construction of LDGM codes with those rates is complicating and requires large search space, we decompose the construction into two stages.

Assume that $\beta = 2$ and $\alpha = 0.125$. At the first stage, we start with the base matrix of well designed code such as R4JA code with rate $1/2$ [27] represented by

$$\mathbf{B}_{\text{R4JA}} = \begin{bmatrix} 2 & 2 & 1 & 1 \\ 1 & 1 & 3 & 1 \end{bmatrix}.$$

By setting the maximum parallel edges to p, the base matrix for the desired rate $1/3$ can be given as

$$\mathbf{B}_{1/3} = \left[\begin{array}{c:c} \mathbf{B}_{\text{R4JA}} & \mathbf{0} \\ \hdashline \mathbf{B}_1 & \mathbf{I}_2 \end{array}\right] = \left[\begin{array}{cccc:cc} 2 & 2 & 1 & 1 & 0 & 0 \\ 1 & 1 & 3 & 1 & 0 & 0 \\ \hdashline x_1 & x_2 & x_3 & x_4 & 1 & 0 \\ x_5 & x_6 & x_7 & x_8 & 0 & 1 \end{array}\right],$$

Algorithm 2 Procedure of the second writing

Input: $\mathbf{m}, \alpha, \beta, \theta, L, \left\{C_Q^{(0)}, \cdots, C_Q^{(\beta)}\right\}, (\mathbf{q}_{\langle 1 \rangle}, \cdots, \mathbf{q}_{\langle L' \rangle})$
1: **for** $i = 0$ to β **do**
2: **for** $t = 1$ to θ **do**
3: Randomly select l_0 from the fragment indices and $p^{(0)} \leftarrow p_{l_0}^{\star}$
4: **for** $j = 1$ to $i + 2$ **do**
5: Randomly select l_j satisfying $q_{\langle l_j \rangle, j} = 1$ and $p^{(j)} \leftarrow p_j$
6: **end for**
7: $\mathbf{v} \leftarrow \left(p^{(0)}, p^{(1)}, \cdots, p^{(i+2)}\right)$
8: Do **Algorithm 1** with $\mathbf{m}$, $\mathbf{v}$, $\mathbf{G}_Q^{(i)}$ as inputs
9: **if** **Algorithm 1** is successful **then**
10: $\mathbf{x} \leftarrow$ output of **Algorithm 1**
11: **return** $\mathbf{x}$
12: **end if**
13: **end for**
14: **end for**
15: **return** failure

where $0 \le x_i \le p$ for $i = 1, 2, \cdots, 8$. Using the computer search by PEXIT analysis, $x_1 = 1, x_3 = 3, x_7 = 3$, and other $x_i = 0$ are found when $p = 3$. Similar to the raptor-like structure [15], the protograph of code with the lowest rate for $\beta = 2$, $\mathbf{B}_{1/4}$ can be constructed from $\mathbf{B}_{1/3}$ as

$$\mathbf{B}_{1/4} = \left[\begin{array}{cc:c} \mathbf{B}_{\text{R4JA}} & \mathbf{0} & \mathbf{0} \\ \hdashline \mathbf{B}_1 & \mathbf{I}_2 & \mathbf{0} \\ \mathbf{B}_2 & \mathbf{0}_2 & \mathbf{I}_2 \end{array}\right]$$

because the search space $\mathbf{B}_2$ should be limited such as 4^8. Then, $\mathbf{B}_2$ is obtained as

$$\mathbf{B}_2 = \begin{bmatrix} 1 & 0 & 3 & 0 \\ 1 & 0 & 3 & 0 \end{bmatrix}.$$

Now, the first lifting algorithm [21] is applied to avoid parallel edges in the base matrix $\mathbf{B}_{1/4}$. For example, it becomes 24×32 matrix $\mathbf{B}'_{1/4}$ when lifting factor 4 is used.

At the second stage of the construction, $\mathbf{B}'_{1/4}$ needs to be extended again due to the fractional part α. The protograph of rate $R_{2,0} = \frac{1}{2+\alpha}$ is constructed based on the lifted version of $\mathbf{B}_{\text{R4JA}}$, denoted by $\mathbf{B}'_{\text{R4JA}}$. Then the desired protograph for $\beta = 0$ and $\alpha = 0.125$, i.e., $\mathbf{B}'_{8/17}$, is represented as

$$\mathbf{B}'_{8/17} = \left[\begin{array}{c:ccccc} 1 & x_1 & x_2 & \cdots & x_{16} \\ \hdashline 0 & & & & \\ 0 & & & & \\ \vdots & & \multicolumn{3}{c}{\mathbf{B}'_{\text{R4JA}}} \\ 0 & & & & \end{array}\right],$$

where $0 \le x_i \le 1$ for $i = 1, \cdots, 16$ and the computer search is also used. The other protographs of rates $R_{2,i} = \frac{1}{2+\alpha+i}$, $i = 1, \cdots, \beta$, are also constructed in the same way. Finally, the second lifting algorithm, called circulant progressive edge growth (CPEG) algorithm [28], is applied to guarantee not small girths of the parity check matrices of the RC-LDPC codes, that is, the generator matrices of the RC-LDGM codes.

Table 1. Constructed RC-LDPC codes.

Code	$\alpha = 0.125$			$\alpha = 0.25$		
	C_0	C_1	C_2	C_0	C_1	C_2
Rate	0.471	0.320	0.242	0.444	0.308	0.235
Threshold	0.470	0.648	0.734	0.502	0.652	0.738

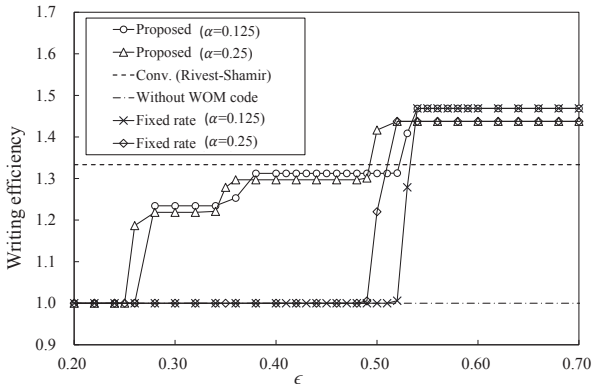

Fig. 5. Comparison of writing efficiency between the proposed and the conventional WOM codes.

V. NUMERICAL ANALYSIS

In this paper, the parameters for the proposed scheme are set to $k = 8192$, $L = 128$, $\alpha = 0.125, 0.25$, and $\beta = 2$. The parity-check matrices of raptor-like codes for each α, $\mathbf{H}_0$, $\mathbf{H}_1$, and $\mathbf{H}_2$, are constructed from Section IV, where the lifting factor for CPEG is set to 1,024 and the girth of each code is larger than or equal to 8. The rates and thresholds of the constructed RC-LDPC codes, C_0, C_1, and C_2, are shown in Table 1. These parity-check matrices are used as the generator matrices of the RC-LDGM codes, that is, BEQ codes, $\mathbf{G}_Q^{(0)} = \mathbf{H}_0$, $\mathbf{G}_Q^{(1)} = \mathbf{H}_1$, and $\mathbf{G}_Q^{(2)} = \mathbf{H}_2$ for the proposed WOM codes.

The performance of writing efficiency of the proposed WOM code is compared to the cases without WOM code and with the conventional WOM codes, i.e., Rivest-Shamir $\langle 4 \rangle^2/3$ WOM code [10] and WOM code with a fixed rewriting rate. Note that the fixed rate WOM code implemented by BEQ code is the same as the proposed rewriting scheme with $\theta = 1$ and $\beta = 0$. The corresponding simulation results are shown in Fig. 5. It can be shown that by adjusting the rate of BEQ code, i.e., from $R_{2,0}$ to $R_{2,1}$ or $R_{2,2}$, the more invalid pages are reusable and thus the number of erase operations can be reduced with the proposed WOM codes. Clearly, the writing efficiency for $\langle 4 \rangle^2/3$ WOM code is fixed to 1.333, whereas those of the proposed scheme vary from 1 to 1.469 and 1.438 for $\alpha = 0.125$ and $\alpha = 0.25$, respectively. Thus, the proposed WOM codes outperform Rivest-Shamir WOM code with respect to η when $\epsilon > 0.51$ and $\epsilon > 0.48$ for $\alpha = 0.125$ and $\alpha = 0.25$, respectively.

Furthermore, it is verified that the page selection method improves rewriting capability. The simulation result of the proposed WOM code with various θ for $\alpha = 0.25$ and $\epsilon = 0.5$ is shown in Table 2. As θ increases from 1 to 20, the writing efficiency also increases from 1.3113 to 1.4427. It is shown that more write requests can be performed before invoking the block erase operation at a cost of increased iteration number, which induces higher computational complexity. However, we assumed

Table 2. Writing efficiency η with various θ when $\alpha = 0.25$ and $L = 128$.

θ	$W^{(0)}$	$W^{(1)}$	Iter.	η
1	19.51	20.33	1.38	1.3113
3	32.38	17.69	2.25	1.3784
5	37.44	15.69	3.17	1.4151
10	42.87	10.26	4.89	1.4335
20	46.08	7.05	7.08	1.4427

* $W^{(2)}$ for all θ is equal to zero

that there is enough time for finding rewritable cell state vector and thus it can be done before the second write. Therefore, we can reduce the erase operation in NAND flash memory with the proposed WOM codes by carefully choosing θ.

In addition, when the proposed WOM code is implemented for MLC using LLH programming, we have $\eta_{\text{MLC}} = 1.234$ and $\eta_{\text{MLC}} = 1.219$ when $\alpha = 0.125$ and $\alpha = 0.25$, respectively for the best cases. Even though the gain for rewriting scheme is decreased, the proposed scheme still has advantage over the conventional scheme without WOM codes.

Now, encoding complexity of the proposed scheme which includes finding valid codeword and accessing complexities is analyzed. Since the proposed WOM code is implemented by multiple BEQ codes with the BP algorithm, the processing complexity is linear to the length of the message, i.e., $O(k)$. Although the encoder of Rivest-Shamir WOM code uses a simple lookup table which takes $O(k)$, the proposed scheme has improved writing efficiency with cost of acceptable complexity.

In the accessing complexity aspect, we define the performance measure $\tau = N_a/N_{tot}$, where N_{tot} and N_a are the numbers of total writing pages and pages accessed to achieve N_{tot}, respectively. For the proposed WOM code with $L = 128$, $\beta = 2$, and $\alpha = 0.125$, 128 pages are accessed and written in the first write and then $60(= 128R_{2,0})$ pages are written by accessing $180(= 60(\beta + 1))$ pages in the second write, which results in $\tau_{\text{prop}} = \frac{128+(60\times3)}{128+60} = 1.63$. For the Rivest-Shamir WOM code, 256 pages are accessed but 128 pages are written in each write, which results in $\tau_{\text{conv}} = \frac{128\times2\times2}{256} = 2$. In addition, while reading and programming latencies become two-fold at the first write in the NAND flash memory with Rivest-Shamir WOM code because $\lceil 3/2 \rceil$ pages are used, those of the proposed rewriting scheme with BEQ code are not increased at the first write.

VI. CONCLUSIONS

While NAND flash memory has many advantages such as high storage density and low latency, it can be worn out due to frequent erase operations. Thus, reducing the number of erase operations can be one of key approaches to extend lifetime of NAND flash memory. In this paper, a flexible two-write WOM code is proposed, which adopts incremental encoding process with raptor-like codes for the second write. Also, by selecting invalid pages and ordering those pages appropriately to make cell state vectors rewritable, writing efficiency of the proposed scheme is improved. It is verified via numerical analysis that the proposed rewriting scheme outperforms the conventional rewriting schemes with respect to writing efficiency.

REFERENCES

[1] L. M. Grupp *et al.,* "Characterizing flash memory: Anomalies, observations, and applications," in *Proc. IEEE/ACM MICRO*, Dec. 2009, pp. 24–33.

[2] H. Choi, W. Liu, and W. Sung, "VLSI implementation of BCH error correction for multilevel cell NAND flash memory," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 18, no. 5, pp. 843–847, May 2010.

[3] J. Kim and W. Sung, "Rate-0.96 LDPC decoding VLSI for soft-decision error correction of NAND flash memory," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 22, no. 5, pp. 1004–1015, May 2014.

[4] S. Tanakamaru, M. Doi, and K. Takeuchi, "A design strategy of error-prediction low-density parity-check (EP-LDPC) error-correcting code (ECC) and error-recovery schemes for scaled NAND flash memories," *IEICE Trans. Electron.*, vol. E98-C, no. 1, pp. 53–61, Jan. 2015.

[5] S. Kotaki and M. Kitakami, "An error correction method for neighborhood-level errors in NAND flash memories," *IEICE Trans. Fundamentals Electron., Commun. Comput. Sci.*, vol. E100-A, no. 2, pp. 653–662, Feb. 2017.

[6] A. Jagmohan, M. Franceschini, and L. Lastras, "Write amplification reduction in NAND flash through multi-write coding," in *Proc. IEEE MSST*, May 2010, pp. 1–6.

[7] E. Yaakobi, H. Mahdavifar, P. H. Siegel, A. Vardy, and J. K. Wolf, "Rewriting codes for flash memories," *IEEE Trans. Inf. Theory*, vol. 60, no. 2, pp. 964–975, Feb. 2014.

[8] G. Yadgar, E.Yaakobi, and A. Schuster "Write once, get 50% free: Saving SSD erase costs using WOM codes," in *Proc. FFAST*, Feb. 2015, pp. 257–271.

[9] W. J. M. Chua, K. Cai, and W. L. Goh, "Efficient two-write WOM-codes for non-volatile memories," *IEEE Commun. Lett.*, vol. 19, no. 10, pp. 1690–1693, Oct. 2015.

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

[11] C. Heegard, "On the capacity of permanent memory," *IEEE Trans. Inf. Theory*, vol. 31, no. 1, pp. 34–42, Jan. 1985.

[12] R. Sekiya and B. M. Kurkoski, "Applying write-once memory codes to binary symmetric asymmetric multiple access channels," *IEICE Trans. Fundamentals Electron., Commun. and Comput. Sci.*, vol. E99-A, no. 12, pp. 2202–2210, Dec. 2016.

[13] D. Burshtein and A. Strugatski, "Polar write once memory codes," *IEEE Trans. Inf. Theory*, vol. 59, no. 8, pp. 5088–5101, Aug. 2013.

[14] E. E. Gad, W. Huang, Y. Li, and J. Bruck, "Rewriting flash memories by message passing," in *Proc. IEEE ISIT*, July 2015, pp. 646–650.

[15] T.-Y. Chen, K. Vakilinia, D. Divsalar, and R. D. Wesel, "Protograph-based raptor-like LDPC codes," *IEEE Trans. Commun.*, vol. 63, no. 5, pp. 1522–1532, May 2015.

[16] T.-S. Chung, D.-J. Park, S. Park, D.-H. Lee, S.-W. Lee, and H.-J. Song, "A survey of flash translation layer," *J. Syst. Archit.*, vol. 55, pp. 332–343, May 2009.

[17] F. Margaglia and A. Brinkmann, "Improving MLC flash performance and endurance with extended P/E cycles," in *Proc. MSST*, May 2015, pp. 1–12.

[18] F. Margaglia *et al.,* "The devil is in the details: Implementing flash page reuse with WOM codes," in *Proc. USENIX FAST*, Feb. 2016, pp. 95–109.

[19] E. Martinian and J. S. Yedidia, "Iterative quantization using codes on graphs," in *Proc. Allerton Conf. Commun., Control, Comput.*, Oct. 2003.

[20] J. Thorpe, "Low-density parity-check (LDPC) codes constructed from protographs," in *Proc. IPN Progr. Rep.*, Aug. 2003, pp. 1–7.

[21] D. Divsalar, S. Dolinar, C. R. Jones, and K. Andrews, "Capacity-approaching protograph codes," *IEEE J. Sel. Areas Commun.*, vol. 27, no. 6, pp. 876–888, Aug. 2009.

[22] S. Choi, S. Yoon, W. Sung, H. Kwon, and J. Heo, "A good puncturing scheme for rate compatible low-density parity-check codes," *J. Commun. Netw.*, vol. 11, no. 5, pp. 455–463, Oct. 2009.

[23] T. Nguyen, A. Nosratinia, and D. Divsalar, "The design of rate-compatible protograph LDPC codes," *IEEE Trans. Commun.*, vol. 60, no. 10, pp. 2841–2850, Oct. 2012.

[24] T. V. Nguyen and A. Nosratinia, "Rate-compatible short-length protograph LDPC codes," *IEEE Commun. Lett.*, vol. 17, no. 5, pp. 948–951, May 2013.

[25] L. Zhou, W.-C. Huang, R. Zhao, and Y.-C. He, "A non-greedy puncturing method for rate-compatible LDPC codes," *J. Commun. Netw.*, vol. 19, no. 1, pp. 32–40, Feb. 2017.

[26] G. Liva and M. Chiani, "Protograph LDPC code design based on EXIT chart analysis," in *Proc. IEEE GLOBECOM*, Nov. 2007, pp. 3250–3254.

[27] D. Divsalar, C. Jones, S. Dolinar, and J. Thorpe, "Protograph based LDPC codes with minimum distance linearly growing with block size," in *Proc. IEEE GLOBECOM*, Nov. 2005, pp. 1152–1156.

[28] Z. Li and B. V. Kumar, "A class of good quasi-cyclic low-density parity check codes based on progressive edge growth graph," in *Proc. Asilomar Conf. Signals, Syst. Comput.*, Nov. 2004, pp. 1990–1994.

Bohwan Jun received the B.S. and Ph.D. degrees in Electrical and Computer Engineering from Seoul National University, Seoul, Korea, in 2011 and 2017, respectively. He is currently a Senior Engineer at Samsung Electronics, Co., Ltd., Gyeonggi-do, Korea. His area of research interests include error-correcting codes, coding theory, and coding for memory.

Heeyoul Kwak received the B. S. degree in Electrical and Computer Engineering from Seoul National University, Seoul, Korea, in 2013, where he is currently pursuing the Ph.D. degree in Electrical Engineering and Computer Science. His area of research interests includes error-correcting codes, coding theory, and coding for memory.

Jong-Seon No received the B.S. and M.S.E.E. degrees in Electronics Engineering from Seoul National University, Seoul, Korea, in 1981 and 1984, respectively, and the Ph.D. degree in Electrical Engineering from the University of Southern California, Los Angeles, CA, USA, in 1988. He was a Senior MTS with Hughes Network Systems from 1988 to 1990. He was an Associate Professor with the Department of Electronic Engineering, Konkuk University, Seoul, from 1990 to 1999. He joined the faculty of the Department of Electrical and Computer Engineering, Seoul National University, in 1999, where he is currently a Professor. His area of research interests includes error-correcting codes, sequences, cryptography, LDPC codes, interference alignment, and wireless communication systems. He was a recipient of the IEEE Information Theory Society Chapter of the Year Award in 2007. From 1996 to 2008, he served as a Founding Chair of the Seoul Chapter of the IEEE Information Theory Society. He was a General Chair of Sequence and Their Applications 2004, Seoul. He also served as a General Co-Chair of the International Symposium on Information Theory and Its Applications 2006 and the International Symposium on Information Theory 2009, Seoul. He has been a Co-Editor-in-Chief of the IEEE JOURNAL OF COMMUNICATIONS AND NETWORKS since 2012.

Hosung Park received the B.S., M.S., and Ph.D. degrees in Electrical Engineering from Seoul National University, Seoul, Korea, in 2007, 2009, and 2013, respectively. He was a Post-Doctoral Researcher with the Institute of New Media and Communications, Seoul National University, in 2013, and the Qualcomm Institute, the California Institute for Telecommunications and Information Technology, University of California at San Diego, La Jolla, CA, USA, from 2013 to 2015. He has been an Assistant Professor with the School of Electronics and Computer Engineering, Chonnam National University, Gwangju, Korea, since 2015. His research interests include channel codes for communications systems, coding for memory, coding for distributed storage, coded computing, communication signal processing, compressed sensing, and network information theory.