IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Iterative Pseudo-Soft-Reliability-based Majority-Logic Decoding for NAND Flash Memory

KYEONG BIN PARK[1], (Student Member, IEEE,) and KI-SEOK CHUNG, (Member, IEEE)
Department of Electronic and Computer Engineering, Hanyang University, 04763 Seoul, Korea (e-mail: lay1523@naver.com[1])

Corresponding author: Ki-Seok Chung (e-mail: kchung@hanyang.ac.kr).

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. 2020R1A4A4079177).

ABSTRACT This paper proposes a decoding algorithm for nonbinary low-density parity-check (NB-LDPC) codes, aiming to improve the error rate performance for NAND flash memory. Several NB-LDPC decoding methods for NAND flash memory have been studied. Some approaches rely on hard decisions, and these are relatively simple but do not have a good error rate performance. Others are based on soft decisions that require multiple reads for each flash memory cell, leading to significant memory throughput degradation. To improve the error rate performance without suffering performance degradation owing to multiple reads, an iterative pseudo-soft-reliability-based decoding algorithm is proposed. Using Galois field addition to calculate the Hamming distance at the initialization, the proposed algorithm not only improves the error rate performance but also reduces the average number of iterations compared with those of conventional hard-decision-based decoding algorithms.

INDEX TERMS Error correction codes, Hamming distance, Hard decision, Iterative hard-reliability-based majority-logic decoding algorithm, NAND flash memory, Nonbinary low-density parity-check codes

I. INTRODUCTION

NON-BINARY low-density parity-check (NB-LDPC) codes over GF(q) ($q > 2$) are known to have a better error rate performance than binary LDPC codes when the code length is moderate and the code rate is high [1]–[3]. Despite their remarkable error rate performance, the decoding complexity of NB-LDPC codes is unacceptably high. For instance, a well-known NB-LDPC decoding method called q-ary sum-product algorithm (QSPA) has a decoding complexity of $O(q^2)$ for a single check node (CN) update. To reduce this high decoding complexity, numerous decoding algorithms have been proposed. In particular, for NAND flash memory, high data throughput is crucial. Therefore, quite a few schemes have focused on reducing the decoding complexity of the NB-LDPC codes for NAND flash memory [4]–[10]. Moreover, the reliability of flash memory cells continues to degrade owing to the rapid increase in storage density via multi-level data cells. Therefore, a reduction in the decoding complexity should not aggravate the error rate performance.

The information stored in a flash memory cell is determined by the amount of electron charge trapped in the cell. In the case of a single-level cell, a single read is required to determine the logic value, 0 or 1, which is called hard decision. Obviously, it is impossible to determine whether this hard decision is true or whether the information is corrupted. To overcome the drawback of the hard decision and obtain probabilistic information on whether the corresponding bit is 0 or 1, which is called soft reliability, previous studies have proposed the use of *multiple read* with multiple levels of read threshold voltages [11]–[13]. However, this multiple-read operation causes significant throughput degradation. Meanwhile, the authors in [6]–[8] proposed the application of NB-LDPC codes for MLC NAND flash memory, and the soft-reliability-based QSPA was employed for decoding. The QSPA, however, suffered a significant throughput degradation owing to high decoding complexity. Therefore, a decoding algorithm that has a good error rate performance and does not suffer high decoding throughput degradation is necessary for NAND flash memory. Further, the code rate of the NAND flash memory is significantly higher than that of wireless communication because the memory space allowed

for the parity bits is relatively small. Therefore, the decoding algorithm for NAND flash memory should have a practically decent error rate performance, even at a high code rate.

In general, there are three types of algorithms for reducing the decoding complexity of NB-LDPC codes: simplified belief propagation decoding (SBPD) [14]–[17], symbol flipping decoding (SFD) [4], [18]–[22], and majority-logic decoding (MLgD) [23]–[26] algorithms. The SBPD algorithms outperform the other two types, but their computational complexity is higher than that of the other two. In contrast, the complexities of the SFD and MLgD algorithms are considerably lower than that of the SBPD algorithms, and both algorithms have a slightly inferior error rate performance to the SBPD algorithms. The iterative hard-reliability-based MLgD (IHRB-MLgD) algorithm [23] updates the reliability based on hard decisions. Thus, the IHRB-MLgD algorithm has a significantly lower computational complexity at the cost of losing some error rate performance.

In this paper, we propose an iterative pseudo-soft-reliability-based MLgD (IPSRB-MLgD) algorithm that uses the Hamming distance. Several studies [24]–[26] have proposed the use of soft reliability, but they are not suitable for NAND flash memory because fine-grained soft reliability can only be derived by multiple reads. In contrast, the IPSRB-MLgD algorithm calculates the Hamming distances between the hard-decision symbol and the other symbols and uses them as the soft reliability of each symbol at the initialization. Accordingly, the IPSRB-MLgD algorithm does not require multiple reads and improves the error rate performance of the hard-decision-based decoding algorithm. For brevity, the term "MLgD" will be omitted when we mention the related algorithms for the rest of the paper. To validate the effectiveness of the proposed algorithm, in this study, we evaluated the error rate performance in two different types of channels: an additive white Gaussian noise (AWGN) channel modeled as a general communication channel and a binary symmetric channel (BSC) modeled as a NAND flash memory channel [27]–[30]. In addition, we compared only the hard-decision-based decoding algorithms with a high code rate (0.89) to preserve the system throughput and minimize the parity bit length. Compared with the IHRB algorithm, the proposed IPSRB algorithm has a better bit error rate (BER) performance by nearly 1 dB at a BER of 10^{-5} with a decreased average number of iterations.

II. PRELIMINARIES

The NB-LDPC code C is defined by an ultra sparse parity-check matrix $\mathbf{H}$ or a graphical representation of the matrix called the Tanner graph [31], [32]. Fig. 1 shows the parity-check matrix and its Tanner graph representation. Nonzero elements in $\mathbf{H}$ belong to Galois field GF(q), where $q = 2^p$ for some positive integer p, and the elements construct the interconnections between CNs and variable nodes (VNs) in the Tanner graph. In the Tanner graph representation, there are N VNs (columns in $\mathbf{H}$) and M CNs (rows in $\mathbf{H}$). The degree of a node in a Tanner graph, which is the number

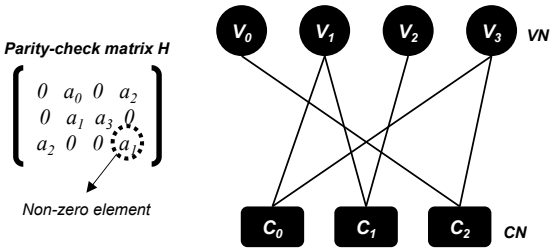


FIGURE 1. Tanner-graph representation of parity-check matrix H with GF(4).

of its adjacent nodes, corresponds to the number of nonzero elements in a column or a row in $\mathbf{H}$, and we will call it the weight in the matrix. This study considers only regular NB-LDPC codes, the $\mathbf{H}$ of which has a constant column weight d_v and a constant row weight d_c. For NB-LDPC code with a length of N, let $\mathbf{c} = [c_0, c_1, ..., c_{N-1}]$ be a codeword of C. Let $\mathbf{x} = [x_0, x_1, ..., x_{N-1}]$ and $\mathbf{y} = [y_0, y_1, ..., y_{N-1}]$ are channel input vector and received vector, respectively. Thus, the vector $\mathbf{y}$ is noisy vector of $\mathbf{x}$. Based on the $\mathbf{y}$, let $\mathbf{z}^{(k)} = [z_0^{(k)}, z_1^{(k)}, ..., z_{N-1}^{(k)}]$ be the hard-decision vector of the received symbols in the k^{th} decoding iteration. Accordingly, $\mathbf{z}^{(0)}$ consists of the hard-decision symbols from the channel output and is the input of the decoder. The goal of the decoding iteration is to make $\mathbf{z}^{(k)}$ satisfies the syndrome check equation $\mathbf{z}^{(k)} \times H^T = 0$, which means that $\mathbf{z}^{(k)}$ is codeword.

Given the initial hard-decision vector $\mathbf{z}^{(0)}$, the authors in [19] proposed two iterative hard-decision algorithms called generalized Gallager's Algorithm B (AlgB) and weighted Gallager's Algorithm B (wtd-AlgB). At the k^{th} iteration in the AlgB, hard-decision symbol $z_j^{(k)}$ is passed from the j^{th} VN to its neighboring CNs. The *extrinsic information sum* (EXI) passed from the i^{th} CN to the j^{th} VN is denoted as $\sigma_{i,j}^{(k)}$, and $\sigma_{i,j}^{(k)}$ is derived as

$$\sigma_{i,j}^{(k)} = h_{i,j}^{-1} \sum_{u \in N_i \backslash j} z_u^{(k)} h_{i,u} \tag{1}$$

In (1), N_i is the set of VNs that are connected to the i^{th} CN, and $N_i \setminus j$ is the subset of N_i without the j^{th} VN ($0 \leq i < M, 0 \leq j < N$). Based on the derived $\sigma_{i,j}^{(k)}$, the VN updates hard-decision vector $\mathbf{z}^{(k)}$ as follows:

$$z_j^{(k+1)} = \begin{cases} \underset{\sigma_{i,j}^{(k)}, i \in M_j}{argmax}\ n(\sigma_{i,j}^{(k)}), & \underset{i \in M_j}{max}\ n(\sigma_{i,j}^{(k)}) \geq T \\ z_j^{(k)}, & \underset{i \in M_j}{max}\ n(\sigma_{i,j}^{(k)}) < T \end{cases} \tag{2}$$

where $n(\sigma_{i,j}^{(k)})$ denotes the number of occurrences of $\sigma_{i,j}^{(k)}$, and T is a predetermined threshold value.

The wtd-AlgB is a modified version of the AlgB, and its error rate performance is improved by employing the Hamming distance. The VN update in the wtd-AlgB is modified as

$$z_j^{(k+1)} = \begin{cases} \underset{\sigma_{i,j}^{(k)}, i \in M_j}{argmax}\ I(\sigma_{i,j}^{(k)}), & \underset{i \in M_j}{max}\ I(\sigma_{i,j}^{(k)}) \geq T \\ z_j^{(k)}, & \underset{i \in M_j}{max}\ I(\sigma_{i,j}^{(k)}) < T \end{cases} \quad (3)$$

where $d(a, b)$ denotes the Hamming distance between two symbols a and b, and $\theta_{d(a,b)}$ denotes the weighting factor that corresponds to $d(a, b)$. $I(\sigma_{i,j}^{(k)})$ is defined as $\theta_{d(z_j^{(k)}, \sigma_{i,j}^{(k)})} \times n(\sigma_{i,j}^{(k)})$.

Recently, SFD algorithms that based on prediction [21], [22] improves error rate performances but it is not considered in this study because they are soft reliability based algorithms. Meanwhile, the authors in [4] proposed a method called the decision-symbol-reliability-based SFD (DRB-SFD) algorithm, which aims to improve the error rate performance for NAND flash memory. The DRB-SFD algorithm uses the *crossover probability* derived from the *program/erase (P/E) cycle* of flash memory, in addition to the hard-decision outputs from memory cells [33], where the crossover probability is the probability that a transmitted bit is flipped under the BSC. The crossover probability, however, does not increase monotonically with the P/E cycle count [34]. Therefore, the DRB-SFD algorithm, which requires the crossover probability in advance, is not practically applicable to commercial products.

In contrast, the MLgD algorithms consider only the most reliable symbol in the CN update. Thus, the MLgD algorithms are computationally simpler than the SBPD algorithms. In the MLgD algorithms, the reliability of a symbol is decided by voting for a VN based on the messages from adjacent CNs. There are two types of MLgD algorithms: the iterative soft-reliability-based MLgD algorithm and the IHRB-MLgD algorithm. The main difference between these two types of algorithms is that they use the soft or hard reliability from the channel as the initial reliability. The enhanced IHRB (EIHRB) and improved EIHRB (IEIHRB) algorithms in [25], [26] introduce soft reliability at the initialization and have improved the error rate performance.

Among the previously proposed SFD and MLgD algorithms, some used the Hamming distance to improve the error rate performance. In the *weighted bit-reliability-based* decoding algorithm [24], the Hamming distance between the EXI and the hard-decision symbol is used as the reliability of the EXI. In the SFD algorithm based on prediction [21], a method that employed the Hamming distance and a plurality logic as a flipping metric was proposed. In these two algorithms, the error rate performance is improved by using the Hamming distance as the reliability at the node-updating step.

III. PROPOSED ALGORITHM

The IHRB algorithm [23] is an iterative decoding algorithm that starts with the given initial hard-decision vector $\mathbf{z}^{(0)}$, and the reliability of the received symbols is updated based on the majority logic. $\mathbf{R}_j^{(k)} = [R_{j,0}^{(k)}, R_{j,1}^{(k)}, ..., R_{j,q-1}^{(k)}]$ is the reliability vector of the j^{th} received symbols. It indicates the probabilities that the j^{th} received symbol is equal to each Galois field element and defines a set $\{a_0, a_1, ..., a_{q-1}\}$ that consists of all the elements of GF(q). At the *initialization*, as described in Algorithm 1, the reliability of hard-decision symbol $R_{j,l}^{(0)}$ is set to γ and the reliabilities of the other symbols are set to zero, where γ is a predetermined positive integer. In other words, the IHRB algorithm creates a reliability biased toward hard-decision symbols.

Algorithm 1: IHRB algorithm

1: *initialization*
$\mathbf{R}_{j,l}^{(0)} = \gamma$ if $z_j^{(0)} = a_l$;
$\mathbf{R}_{j,l}^{(0)} = 0$ otherwise, $0 \leq l \leq q-1$
//iterative decoding
$k = 0 : I_{max}$
2: *syndrome check*
Stop if $z^{(k)} \times H^T = 0$
3: *CN update*
for $i = 0 : m-1$
for $j \in N_i$
$\sigma_{i,j} = h_{i,j}^{-1} \sum_{u \in N_i \backslash j} z_u^{(k)} h_{i,u}$
if $\sigma_{i,j} = a_l$
$R_{j,l}^{(k+1)} = R_{j,l}^{(k)} + 1$
4: *VN update*
for $j = 0 : n-1$
$z_j^{(k+1)} = argmax_l R_{j,l}^{(k+1)}$

One of the reasons why the error rate performance of the soft-decision algorithm is higher than that of the hard-decision algorithm is that the soft-decision algorithm considers the possibility of all the symbols from the initialization. Therefore, if the reliability of other symbols, except for the hard-decision symbols, is initialized to 0, as in the IHRB algorithm, the error rate performance will be inevitably poor. This weakness of the IHRB algorithm could be overcome by employing soft reliability at the initialization, as in the EIHRB and IEIHRB algorithms. However, the cells in the NAND flash memory essentially derive only hard reliability without multiple reads. To generate soft reliability without conducting multiple reads in the flash memory, we propose the use of the Hamming distance between the hard-decision symbol and the other symbols. We call this reliability the pseudo-soft reliability in this paper.

The NB-LDPC codes consist of symbols that are Galois field elements. The Galois field elements can be represented by binary numbers. Hence, the Hamming distances between elements can be easily computed. Because the Hamming distance between two symbols indicates the correlation between the two symbols, the greater the Hamming distance from the hard-decision symbol, the lower the correlation of the corresponding symbol and the hard-decision symbol. Based on this observation, the proposed IPSRB algorithm initializes

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3079939, IEEE Access

the reliability of not only the hard-decision symbol but also the other symbols at the initialization.

The improved IHRB (IIHRB) algorithm [35] also uses the Hamming distance at the initialization, but the IIHRB algorithm considers only the symbols the Hamming distance with the hard-decision symbol of which is 1. However, the higher the code rate, the worse the error rate performance; hence, the reliability of all symbols should be assigned at initialization to improve the error rate performance. In the NAND flash memory, especially, a high-rate code is used; therefore, the IIHRB algorithm may not have a good error rate performance. Furthermore, the larger the Galois field size, the more the reliability information that will be ignored in the IIHRB algorithm because the number of symbols, the Hamming distance to an arbitrary symbol of which is not 1, increases rapidly. In addition, the IHRB and IIHRB algorithms need to find the optimal γ through a simulation, but the proposed algorithm does not require such a process. The proposed algorithm exhibits a better error rate performance than the IIHRB algorithm presented in [35], and the details will be reported in the experimental results.

Algorithm 2: IPSRB algorithm

1: *initialization*
$\mathbf{R}_{j,l}^{(0)} = d_v \times (p - d(z_j^{(0)}, a_l)), 0 \leq l \leq q-1$
//iterative decoding
$k = 0 : I_{max}$
2: *syndrome check*
Stop if $z^{(k)} \times H^T = 0$
3: *CN update*
for $i = 0 : m-1$
for $j \in N_i$
$\sigma_{i,j} = h_{i,j}^{-1} \sum_{u \in N_i \backslash j} z_u^{(k)} h_{i,u}$
if $\sigma_{i,j} == a_l$
$R_{j,l}^{(k+1)} = R_{j,l}^{(k)} + 1$
4: *VN update*
for $j = 0 : n-1$
$z_j^{(k+1)} = argmax_l R_{j,l}^{(k+1)}$

Algorithm 2 describes the proposed IPSRB algorithm in detail. Unlike the IHRB algorithm described in **Algorithm 1**, $d_v \times p$ denotes the highest reliability among all symbols at the initialization of the IPSRB algorithm, where p is computed as $log_2 q$. The reliability of all the other symbols is set to $d_v \times (p - d(z_j^{(0)}, a_l))$, implying that the closer the Hamming distance from the hard-decision symbol, the higher the reliability of the symbol. The Hamming distance can be easily derived by the binary XOR operation between symbols; the addition of such a simple operation improves the error rate performance of the IPSRB algorithm significantly.

Consequently, the difference between the IHRB and IPSRB algorithms is only the reliability vector at initialization. Fig. 2, for example, presents the difference in $R_{j,l}^{(0)}$ between the IHRB (left) and IPSRB (right) algorithms when the j^{th} hard-decision symbol $z_j^{(0)}$ is a_0. Based on the $R_{j,l}^{(0)}$, if symbol

a_l	$R_{j,l}^{(0)}$
a_0	γ
a_1	0
a_2	0
a_3	0
a_4	0
a_5	0
a_6	0
a_7	0

a_l	$R_{j,l}^{(0)}$
a_0	$d_v(p - d(a_0, a_0))$
a_1	$d_v(p - d(a_0, a_1))$
a_2	$d_v(p - d(a_0, a_2))$
a_3	$d_v(p - d(a_0, a_3))$
a_4	$d_v(p - d(a_0, a_4))$
a_5	$d_v(p - d(a_0, a_5))$
a_6	$d_v(p - d(a_0, a_6))$
a_7	$d_v(p - d(a_0, a_7))$

FIGURE 2. Initialization difference between the IHRB and IPSRB algorithms over GF(8).

$\sigma_{i,j}$ from the i^{th} CN, adjacent to the j^{th} VN, is equal to a_1, $R_{j,1}^{(0)}$ will be voted in the next iteration, which indicates that $R_{j,1}^{(1)} = R_{j,1}^{(0)} + 1$.

Here, it should be noted that d_v is multiplied by $(p - d(z_j^{(0)}, a_l))$ in the proposed algorithm. In the VN update, voting is performed d_v times. If d_v is not multiplied, the difference in $R_{j,l}^{(0)}$ between each symbol will be small. Therefore, if the votes are evenly distributed, three or more symbols with the same number of votes may appear. For example, assuming that a hard-decision symbol with GF(8) at initialization in the j^{th} VN is a_0, $\mathbf{R}_{j,l}^{(0)}$ is set to [3, 2, 2, 1, 2, 1, 1, 0]. In the next iteration, if $d_v = 4$ and the neighboring CNs vote for a_1, a_2, a_5, and a_5, $\mathbf{R}_{j,l}^{(1)}$ will be changed to [3, 3, 3, 1, 2, 3, 1, 0]. In the case of the aforementioned example, it is troublesome to choose the $z_j^{(k+1)}$ among the four symbols. To prevent such a situation and give a bias to the channel information, as in the IHRB algorithm, d_v is multiplied. Owing to the multiplication, only one symbol with the same number of votes as $z_j^{(k)}$ will appear. Because this means that the corresponding symbol is voted from all CNs neighboring the j^{th} VN, the corresponding symbol is chosen as $z_j^{(k+1)}$ in the proposed algorithm.

IV. PERFORMANCE COMPARISONS

The BER performance and the computational complexity of the proposed algorithm are compared with those of the hard-decision algorithms: AlgB, wtd-AlgB, IHRB, and IIHRB [19], [23], [35]. The QSPA is also implemented as a performance limit in the BER performances.

A. BIT ERROR RATES

The parameters for the previous hard-decision algorithms, which would maximize the BER performance, were determined through a brute-force search. Threshold T for the AlgB and that for the wtd-AlgB were set to $d_v - 1$ and 1, respectively. The $\theta_{d(z_j^{(k)}, \sigma_{i,j}^{(k)})}$ for the wtd-AlgB was set to $[1, 0.75, 0.5]$, thereby indicating that $\theta_{d(z_j^{(k)}, \sigma_{i,j}^{(k)})}$ was 1, 0.75, and 0.25 if $d(z_j^{(k)}, \sigma_{i,j}^{(k)})$ was 0, 1, and 2, respectively.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3079939, IEEE Access

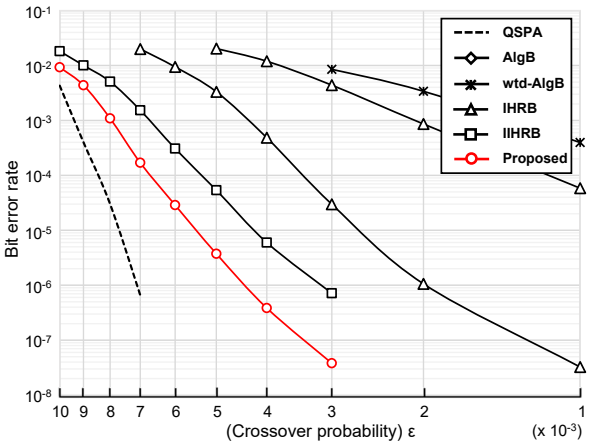


FIGURE 3. BERs under BSC for a (999, 888) code over GF(32), with $d_v = 3$ and $d_c = 27$.

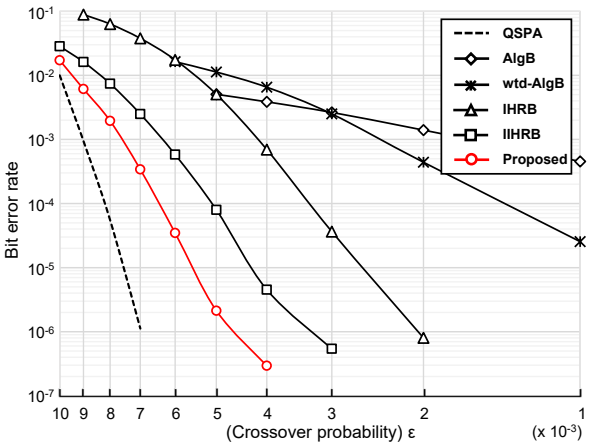


FIGURE 4. BERs under BSC for a (999, 888) code over GF(64), with $d_v = 3$ and $d_c = 27$.

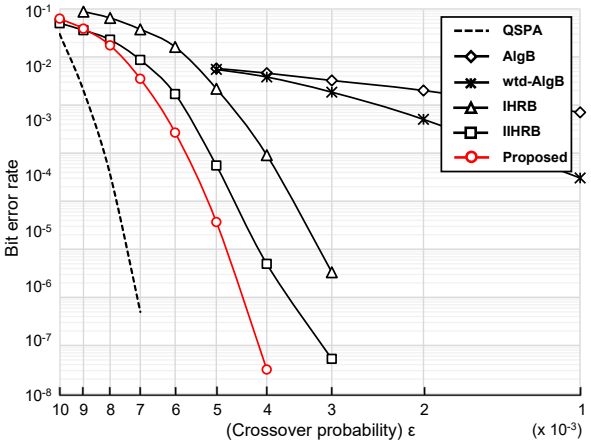


FIGURE 5. BERs under BSC for a (1908, 1696) code over GF(32), with $d_v = 4$ and $d_c = 36$.

Otherwise, $\theta_{d(z_j^{(k)},\sigma_{i,j}^{(k)})}$ was set to 0. The γ values for the IHRB and IIHRB algorithms were set to 6. The reliability information of the symbols, with a Hamming distance of 1 and with the hard-decision symbol, was set to 3 for the IIHRB algorithm [35]. The maximum number of iterations for all the algorithms used in the experiments was fixed at 20.

The channel of the NAND flash memory is often modeled as a channel with asymmetric noise because the error rates for bit values "0" and "1" are observed differently. To overcome the asymmetry of the error rates and reduce the raw BERs, storage device manufacturers have proposed a solution called *voltage optimization* [29]. Because of *voltage optimization*, the NAND flash channel can be modeled as the BSC, and many previous studies on the error correction code for NAND flash storage adopted the BSC [27]–[30]. Therefore, to examine whether the proposed algorithm is suitable for NAND flash memory, the channel was modeled as a BSC, assuming *voltage optimization*. The error rate performance in the AWGN channel with binary phase shift keying modulation $1 \rightarrow +1$ and $0 \rightarrow -1$ were also evaluated, where the AWGN channel is commonly modeled for a telecommunication channel.

Because the proposed decoding algorithm mainly focuses on error correction codes for NAND flash memory, the parity-check matrices were chosen as they were considered to be suitable for the NAND flash memory. The error correction engine in the flash memory decodes stored data per read operation, and the page is the unit of a read operation, where the page size varies from 4K bits to 16K bits. We simulated high-rate codes that had a length of more than 4K bits. The simulated codes were chosen to measure the performances based on different column weights, Galois field sizes, and code lengths. Thus, for performance evaluation, the (999, 888) and (1908, 1696) NB-LDPC codes over GF(32) and the (999, 888) NB-LDPC code over GF(64) were simulated.

We first evaluated the BER of the decoding algorithms under the BSC. Under the BSC, the smaller the *crossover probability* ε, the better the channel reliability, and the more the slope shifts to the left, the better the BER performance. Accordingly, as shown in Fig.3, Fig.4, and Fig.5, the proposed algorithm outperforms all the other hard-decision algorithms in the simulated NB-LDPC codes. Further, the smaller the crossover probability, the larger the difference. At the crossover probability of 5×10^{-3}, only the proposed algorithm achieved a BER that was lower than 10^{-5} in the simulated NB-LDPC codes except QSPA. The BER performances of the AlgB and wtd-AlgB are relatively worse than those of the MLgD algorithms.

We also evaluated the BER of the decoding algorithms under the AWGN, and the results are presented in Fig.6, Fig.7, and Fig.8. Under the AWGN, similar to that under the BSC, the proposed algorithm has the best BER performance among all the hard-decision algorithms. At a BER of 10^{-5} for the (999, 888) codes over GF(32), the proposed algorithm outperforms the IHRB and IIHRB algorithms by approximately 0.8 dB and 0.4 dB, respectively. In addition, the proposed algorithm outperforms the AlgB and wtd-AlgB by more than 2 dB, at the BER of 10^{-5} with all the simulated

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3079939, IEEE Access

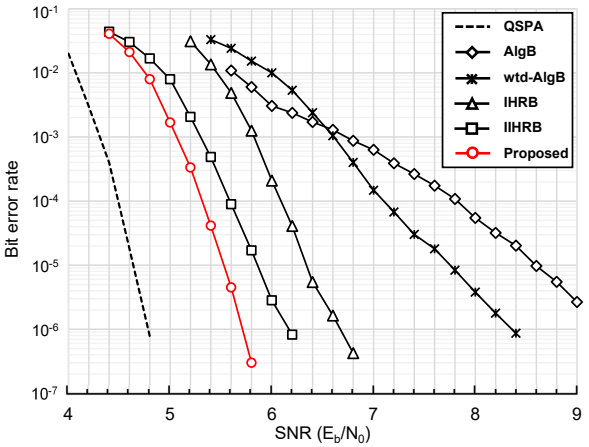


FIGURE 6. BERs under AWGN for a (999, 888) code over GF(32), with $d_v = 3$ and $d_c = 27$.

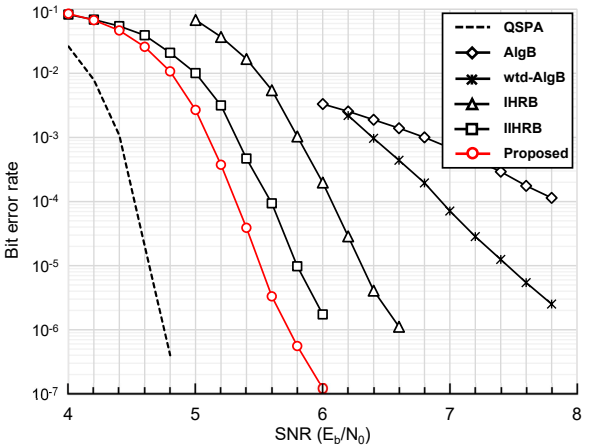


FIGURE 7. BERs under AWGN for a (999, 888) code over GF(64), with $d_v = 3$ and $d_c = 27$.

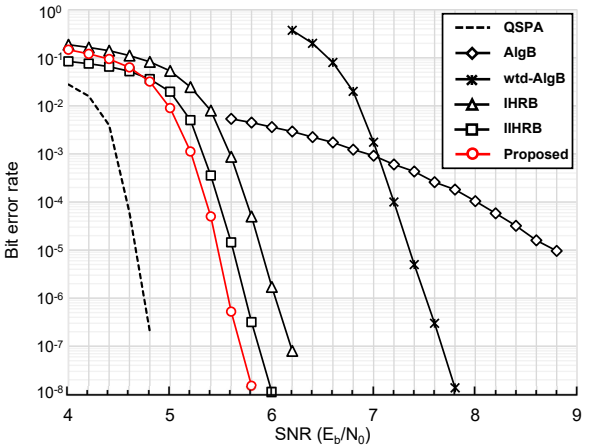


FIGURE 8. BERs under AWGN for a (1908, 1696) code over GF(32), with $d_v = 4$ and $d_c = 36$.

codes. For the rest of the simulated codes, the proposed algorithm achieves an improvement in the BER performance of at least 0.2 dB.

B. COMPUTATIONAL COMPLEXITY

We compared the computational complexity in terms of the number of operations. The comparison results for the extended min-sum (EMS) algorithm [14], wtd-AlgB, IHRB, IIHRB, and IPSRB algorithms are summarized in Table 1. As shown in Table 1, the EMS algorithm, a soft-reliability-based algorithm, needs much more operations than the other algorithms. The proposed algorithm and the IIHRB algorithm perform the same number of operations as the IHRB algorithm at the node-updating stage, but additional operations are required at the initialization to calculate the Hamming distance. The proposed algorithm requires $N(q-1)$ bit-wise XOR operations and $N(q-1)$ integer multiplications (IMs) at the initialization, whereas the IIHRB algorithm requires $N(q-1)$ bit-wise XOR operations and $N(q-1)$ integer comparisons (ICs). It should be noted that the bit-wise XOR operation and the Galois field addition (GA) are identical operations.

Fig.9, Fig.10, and Fig.11 present the average number of iterations for the simulated algorithms. The AlgB and wtd-AlgB were excluded because of unstable error rate performances. The proposed algorithm achieved the lowest average number of iterations among the three algorithms under the same BER for all the simulated codes. The IIHRB algorithm, however, had the highest average number of iterations

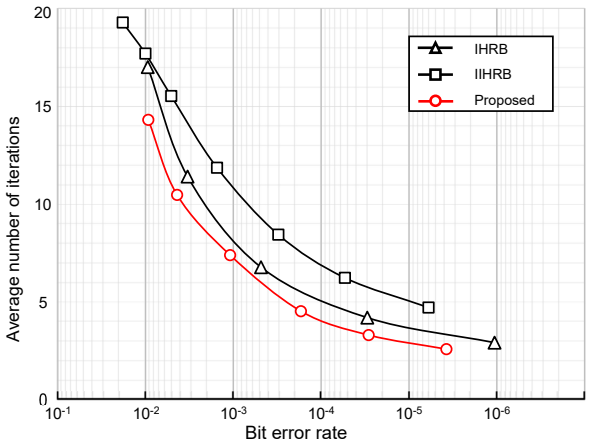


FIGURE 9. Average number of iterations versus BER for a (999, 888) code over GF(32).

Given the number of operations in Table 1 and the average number of iterations, the computational complexities of the simulated algorithms at the BER of 10^{-5} are summarized in Table 2. In the proposed algorithm, the number of GA and Galois multiplication operations is approximately three times higher than that in the IHRB algorithm, but the GA is significantly simpler than integer and real number additions; moreover, the proposed algorithm requires a lower average number of iterations than the IHRB algorithm. Therefore, the throughput of the two algorithms was nearly the same.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3079939, IEEE Access

TABLE 1. Number of operations per iteration of various hard-decision-based decoding algorithms with the (N, M) NB-LDPC code over GF($q = 2^p$)

Decoding algorithm	GA	GM	IA/RA	IM/RM	IC/RC
EMS [14]	$9n_m(\delta - 2M)$	$2\delta n_m$	$n_m(20\delta - 18M - 12N)$	-	$n_m log_2 n_m(9\delta - 12M - 4N)$
wtd-AlgB [19]	$3\delta - M$	2δ	δ	δ	δ
IHRB [23]	$2\delta - M$	2δ	$\delta + Nq$	$2Nq - 2N$	-
IIHRB [35]	$(2\delta - M) + N(q-1)$	2δ	$\delta + Nq$	$2Nq - 2N$	$N(q-1)$
IPSRB	$(2\delta - M) + N(q-1)$	2δ	$\delta + Nq$	$2Nq - 2N + N(q-1)$	-

GA: Galois field addition GM: Galois field multiplication
IA/IM/IC: Integer addition/multiplication/comparison RA/RM/RC: Real addition/multiplication/comparison
n_m: message truncation size [14] $\delta = N \times d_v$ (number of edges in the Tanner graph)

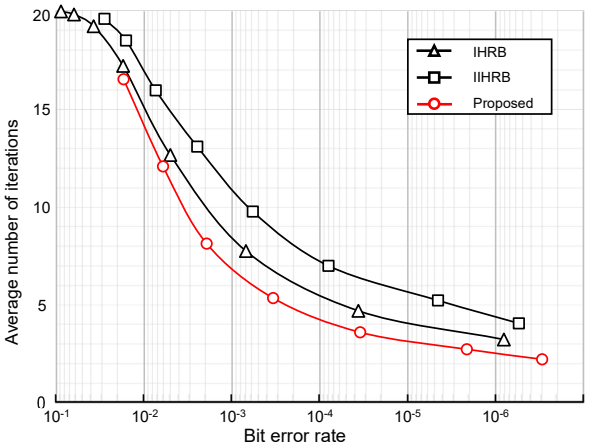


FIGURE 10. Average number of iterations versus BER for a (999, 888) code over GF(64).

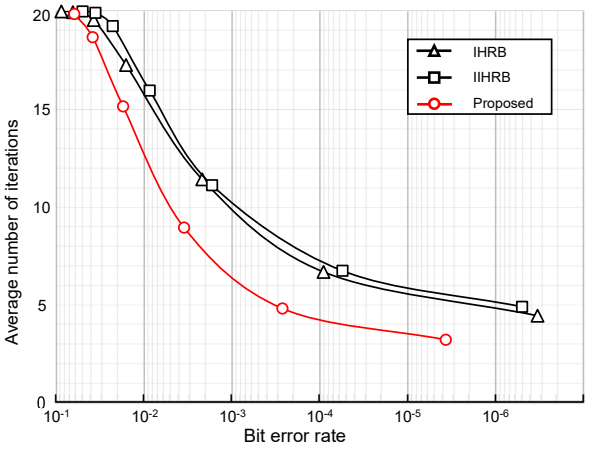


FIGURE 11. Average number of iterations versus BER for a (1908, 1696) code over GF(32).

Furthermore, the IM operations at initialization could be replaced by a look-up table. In this case, the memory space required to store the table is $q \times 64$ bits based on 64-bit integer-type data.

V. CONCLUSION

In this paper, we presented a novel hard-decision-based decoding algorithm using the Hamming distance for NAND flash memory. By introducing the Hamming distance as the reliability information of symbols at the initialization, the proposed algorithm outperforms previous hard-decision based algorithms for various NB-LDPC codes under both the BSC for NAND flash memory and the AWGN channel for communication. We also demonstrated that the proposed algorithm achieved the lowest average number of iterations among all the IHRB-MLgD-based algorithms. Compared with the IHRB-MLgD algorithm, the proposed algorithm requires more GA and IM operations at initialization, but it is offset by the smaller average number of iterations. In the future, based on this study, we will investigate a decoding algorithm for NAND flash memory that can decode a channel assuming asymmetric noise without voltage optimization.

TABLE 2. Computation complexity of various hard-decision-based decoding algorithms at the BER of 10^{-5} for the (1908, 1696) code over GF(32)

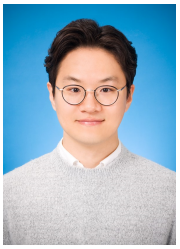
Algorithm	Avg.iter	GA/GM	IA/RA	IM/RM	IC/RC
IHRB [23]	5.1	28832	68688	118296	-
IIHRB [35]	5.2	87980	68688	118296	59148
IPSRB	3.4	87980	68688	177444	-

Avg.iter: Average number of iterations

REFERENCES

[1] Y. Kou, S. Lin, and M. P. Fossorier, "Low-density parity-check codes based on finite geometries: a rediscovery and new results," IEEE Transactions on Information theory, vol. 47, no. 7, pp. 2711–2736, 2001.
[2] C. Poulliat, M. Fossorier, and D. Declercq, "Design of regular (2, d/sub c/)-ldpc codes over gf (q) using their binary images," IEEE Transactions on Communications, vol. 56, no. 10, pp. 1626–1635, 2008.
[3] B. Zhou, J. Kang, S. Song, S. Lin, K. Abdel-Ghaffar, and M. Xu, "Construction of non-binary quasi-cyclic ldpc codes by arrays and array dispersions-[transactions papers]," IEEE Transactions on Communications, vol. 57, no. 6, pp. 1652–1662, 2009.
[4] J. Oh, S. Han, and J. Ha, "An improved symbol-flipping algorithm for nonbinary ldpc codes and its application to nand flash memory," IEEE Transactions on Magnetics, vol. 55, no. 9, pp. 1–13, 2019.
[5] Y. Toriyama and D. Markovic, "A 2.267 gbps, 93.7pj/b non-binary ldpc decoder for storage applications," in 2017 Symposium on VLSI Circuits, pp. C334–C335, 2017.
[6] L. Qiao, H. Wu, D. Wei, and S. Wang, "A joint decoding strategy of non-binary ldpc codes based on retention error characteristics for mlc nand flash memories," in 2016 Sixth International Conference on Instrumentation & Measurement, Computer, Communication and Control (IMCCC), pp. 183–188, IEEE, 2016.
[7] Y. Maeda and H. Kaneko, "Error control coding for multilevel cell flash memories using nonbinary low-density parity-check codes," in 2009 24th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, pp. 367–375, IEEE, 2009.
[8] C. A. Aslam, Y. L. Guan, and K. Cai, "Non-binary ldpc code with multiple memory reads for multi-level-cell (mlc) flash," in Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2014 Asia-Pacific, pp. 1–9, IEEE, 2014.

[9] A. Hareedy, C. Lanka, and L. Dolecek, "A general non-binary ldpc code optimization framework suitable for dense flash memory and magnetic storage," IEEE Journal on Selected Areas in Communications, vol. 34, no. 9, pp. 2402–2415, 2016.
[10] K. Vakilinia, D. Divsalar, and R. D. Wesel, "Optimized degree distributions for binary and non-binary ldpc codes in flash memory," in 2014 International Symposium on Information Theory and its Applications, pp. 6–10, IEEE, 2014.
[11] J. Wang, K. Vakilinia, T.-Y. Chen, T. Courtade, G. Dong, T. Zhang, H. Shankar, and R. Wesel, "Enhanced precision through multiple reads for ldpc decoding in flash memories," IEEE Journal on Selected Areas in Communications, vol. 32, no. 5, pp. 880–891, 2014.
[12] L. Dolecek, "Making error correcting codes work for flash memory," Flash Memory Summit, vol. 3, no. 3.1, pp. 3–3, 2014.
[13] K. Zhao, W. Zhao, H. Sun, X. Zhang, N. Zheng, and T. Zhang, "Ldpc-in-ssd: Making advanced error correction codes work effectively in solid state drives," in Presented as part of the 11th {USENIX} Conference on File and Storage Technologies ({FAST} 13), pp. 243–256, 2013.
[14] D. Declercq and M. Fossorier, "Decoding algorithms for nonbinary ldpc codes over gf (q)," IEEE transactions on communications, vol. 55, no. 4, pp. 633–643, 2007.
[15] A. Voicila, D. Declercq, F. Verdier, M. Fossorier, and P. Urard, "Low-complexity decoding for non-binary ldpc codes in high order fields," IEEE transactions on communications, vol. 58, no. 5, pp. 1365–1375, 2010.
[16] V. Savin, "Min-max decoding for non binary ldpc codes," in 2008 IEEE International Symposium on Information Theory, pp. 960–964, IEEE, 2008.
[17] E. Li, D. Declercq, and K. Gunnam, "Trellis-based extended min-sum algorithm for non-binary ldpc codes and its hardware structure," IEEE Transactions on Communications, vol. 61, no. 7, pp. 2600–2611, 2013.
[18] B. Liu, J. Gao, G. Dou, and W. Tao, "Weighted symbol-flipping decoding for nonbinary ldpc codes," in 2010 Second International Conference on Networks Security, Wireless Communications and Trusted Computing, vol. 1, pp. 223–226, IEEE, 2010.
[19] K. Jagiello and W. E. Ryan, "Iterative plurality-logic and generalized algorithm b decoding of q-ary ldpc codes," in Proc. IEEE Inf. Theory App. Workshop, pp. 1–7, 2011.
[20] F. Garcia-Herrero, D. Declercq, and J. Valls, "Non-binary ldpc decoder based on symbol flipping with multiple votes," IEEE Communications Letters, vol. 18, no. 5, pp. 749–752, 2014.
[21] S. Wang, Q. Huang, and Z. Wang, "Symbol flipping decoding algorithms based on prediction for non-binary ldpc codes," IEEE Transactions on Communications, vol. 65, no. 5, pp. 1913–1924, 2017.
[22] W. Ullah, L. Cheng, and F. Takawira, "Low complexity bit reliability and predication based symbol value selection decoding algorithms for non-binary ldpc codes," IEEE Access, vol. 8, pp. 142691–142703, 2020.
[23] C.-Y. Chen, Q. Huang, C.-c. Chao, and S. Lin, "Two low-complexity reliability-based message-passing algorithms for decoding non-binary ldpc codes," IEEE Transactions on Communications, vol. 58, no. 11, pp. 3140–3147, 2010.
[24] Q. Huang, M. Zhang, Z. Wang, and L. Wang, "Bit-reliability based low-complexity decoding algorithms for non-binary ldpc codes," IEEE Transactions on Communications, vol. 62, no. 12, pp. 4230–4240, 2014.
[25] X. Zhang, F. Cai, and S. Lin, "Low-complexity reliability-based message-passing decoder architectures for non-binary ldpc codes," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 20, no. 11, pp. 1938–1950, 2011.
[26] C. Xiong and Z. Yan, "Improved iterative hard-and soft-reliability based majority-logic decoding algorithms for non-binary low-density parity-check codes," IEEE Transactions on Signal Processing, vol. 62, no. 20, pp. 5449–5457, 2014.
[27] S.-g. Cho, D. Kim, J. Choi, and J. Ha, "Block-wise concatenated bch codes for nand flash memories," IEEE Transactions on Communications, vol. 62, no. 4, pp. 1164–1177, 2014.
[28] D. Kim and J. Ha, "Quasi-primitive block-wise concatenated bch codes with collaborative decoding for nand flash memories," IEEE Transactions on Communications, vol. 63, no. 10, pp. 3482–3496, 2015.
[29] Y. Cai, S. Ghose, E. F. Haratsch, Y. Luo, and O. Mutlu, "Error characterization, mitigation, and recovery in flash-memory-based solid-state drives," Proceedings of the IEEE, vol. 105, no. 9, pp. 1666–1704, 2017.
[30] J. Wang, T. Courtade, H. Shankar, and R. D. Wesel, "Soft information for ldpc decoding in flash: Mutual-information optimized quantization," in 2011 IEEE Global Telecommunications Conference-GLOBECOM 2011, pp. 1–6, IEEE, 2011.
[31] R. Gallager, "Low-density parity-check codes," IRE Transactions on information theory, vol. 8, no. 1, pp. 21–28, 1962.
[32] R. Tanner, "A recursive approach to low complexity codes," IEEE Transactions on information theory, vol. 27, no. 5, pp. 533–547, 1981.
[33] C. A. Aslam, Y. L. Guan, and K. Cai, "Read and write voltage signal optimization for multi-level-cell (mlc) nand flash memory," IEEE transactions on communications, vol. 64, no. 4, pp. 1613–1623, 2016.
[34] J. Meza, Q. Wu, S. Kumar, and O. Mutlu, "A large-scale study of flash memory errors in the field," ACM SIGMETRICS Performance Evaluation Review, vol. 10, pp. 2796314–2745848, 2015.
[35] S. Yeo and I.-C. Park, "Improved hard-reliability based majority-logic decoding for non-binary ldpc codes," IEEE Communications Letters, vol. 21, no. 2, pp. 230–233, 2016.



KYEONGBIN PARK received his B.S. in Electronics & Communication Engineering from Hanyang University, Seoul, Korea in 2014, and He is currently working toward Ph.D. degree in Electronics and Computer Engineering from Hanyang University, Seoul, Korea. His interest research includes algorithms and hardware implementation of error correction codes.

KI-SEOK CHUNG received his B.S. in Computer Engineering from Seoul National University, Seoul, Korea in 1989, and Ph.D. in Computer Science from University of Illinois at Urbana-Champaign in 1998. He was a Senior R&D Engineer at Synopsys, Inc. in Mountain View, CA from 1998 to 2000, and was a Staff Engineer at Intel Corp. in Santa Clara, CA from 2000 to 2001. He also worked as an Assistant Professor at Hongik University, Seoul, Korea from 2001 to 2004. Since 2004, he has been a professor at Hanyang University, Seoul, Korea. His research interests include low power embedded system design, multi-core architecture, image processing, reconfigurable processor and DSP design, SoC-platform based verification and system software for MPSoC.