Ph.D. Dissertation

Bidirectional Error Correcting Codes and Interference Mitigation for Flash Memories

플래시 메모리를 위한 양방향 비대칭 오류 정정 부호 및 간섭 완화 기법

By

Myeongwoon Jeon

February 2014

Department of Electrical Engineering and Computer Science
College of Engineering
Seoul National University

Abstract

Recently, NAND multi-level cell (MLC) flash memories are now widely used due to low cost and high capacity. However, when the number of cell levels increases, cell-to-cell interference (C2CI) which shifts threshold voltage may degrades the error rate in reading process. There are several approaches to alleviate the errors caused by the threshold voltage shift and we discuss error correcting codes and message encoding schemes.

First, we propose error correcting codes that are effective for multi-level cell flash memory and non-binary WOM (write once memory) codes. In particular, we focus on bidirectional error correction codes. The errors in MLC flash memories tend to be directional and limited-magnitude. Many related works focus on asymmetric errors, but bidirectional errors also occur because of the bidirectional interference and the adjustment of the hard-decision reference voltages. The code treats both upward and downward errors when the error magnitude in each direction differs. The maximum magnitudes of the upward error and downward error are l_u and l_d, respectively. One of proposed codes extends the technique of the distinct sum sets to the bidirectional error correction codes. The other code is bidirectional limited magnitude error correction codes based on modulo operation and uses non-binary conventional error correction codes. These proposed codes can reduce the parity size, and have better error correction performance than the conventional error correction codes when the code rate is equal. Furthermore, error correcting schemes for non-binary WOM codes are

discussed. WOM codes is a coding scheme that allows information to be written in a memory cell multiple times without erasure, and conventional error correction codes cannot be directly applied to WOM codes. The advantages of the proposed methods are that these are practical and systematic codes, and the complexity of encoding and decoding processes are low. We also introduce effective error locating limited-magnitude parity check error correction codes for the MLC flash memory error with lower complexity.

Second, we introduce coding schemes to lower the generated interferences by cell to cell interference. It is known that C2CI is caused by the threshold voltage change of neighbor cells in writing operation. The amount of threshold voltage change is proportional to the magnitude. To minimize the generated interference, the average magnitude needs to be decreased. We propose two new C2CI reduction coding schemes that adjust the average magnitude to reduce C2CI. The proposed coding scheme deals with q-ary message codes, and generates fixed length codes. Message codewords are divided into several blocks, and are modified by modulo addition with proper values to minimize the average magnitude. We also propose low energy Huffman codes based on entropy coding when the frequency of symbols is not distributed uniformly. This scheme produces variable-length codes without redundancy. We modified Huffman codes to minimize average number of high bits ('1' bits). We show that proposed codes generate optimal codewords which have minimum high bits with minimum average codeword length.

Keywords: multi-level cell flash memory, error correction code, cell to cell interference,

WOM code, bidirectional error.

Student number: 2009-30210

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Backgrounds

NAND flash memory has been used widely because of its non-volatility, portability and high capacity. Recently, multi-level cell (MLC) flash memories have been studied for improving memory capacity [1] [2]. Multi level cell flash memories use 4 or more levels, and store several bits in a single cell. High density storage of data can be obtained by using a high number of levels in a MLC flash memory cell. A cell of the NAND flash memory is a floating gate transistor, and its threshold voltage can be programmed by injecting certain amount of charges into the floating gate [3]. The threshold voltage (V_T) is used to distinguish data levels in MLC memory. Several factors may change the distribution of the floating-gate threshold-voltage. These factors include cell-to-cell interference, cell leakage, temperature, program voltage (V_{pgm}) disturbance, the pass voltage (V_{pass}) applied to uns-

elected word-lines, etc [4] [5]. One of the dominant factors is the cell-to-cell interference, which is caused by the V_T change of the neighbor cells in the programming (writing) operation. The cell-to-cell interference is approximately proportional to the voltage change of neighbor cells in the programming operation, but it is also affected by the structure of flash memories, the program order, and the number of levels (MLC) in a cell. Another problem related to degradation of the retention characteristics (a retention problem) also occurs with an increasing number of program/erasure cycles in the MLC flash memory.

There are several approaches to alleviate the errors caused by the threshold voltage shift such as the error correcting codes, signal processing methods and data encoding schemes. The conventional error correction codes can be inefficient for multi-level cell (MLC) flash memories because these codes are constructed for all possible error types where error magnitude and direction are random. Therefore, modified ECCs for MLC flash memories have been studied to increase efficiency. The V_T shift which is caused by the cell-to-cell interference is known to be upward (unidirectional). For these asymmetric interference factors, the error correction codes for the asymmetric channel can be useful. Asymmetric channels have been studied for several decades [6]. The topic was studied initially for a binary asymmetric channel (Z-channel). In a Z-channel, the input and the output are binary, and 1 can be changed to 0 with probability p, but 0 cannot be changed to 1. Recently, many error correction codes for asymmetric error with limited-magnitude error are discussed [7] [8] [9] [17]. Although the cell-to-cell interference which leads to upward errors is the dominant factor

in MLC flash memories, there are also bidirectional (random-telegraph noise) and downward (retention noise) interference [11]. The hard-decision reference voltages (V_{read}) for reading flash memory cells are determined based on the V_T distribution after the cell-to-cell interference, not before the cell-to-cell interference, which means the hard-decision reference voltages for reading is already near optimal. After adjusting V_{read} to be near optimal, the number of errors decreases, but the number of downward errors increases. Therefore, bidirectional errors should be considered in order to improve the BER performance. Even if the errors are bidirectional, the magnitude of the errors is still limited. The magnitudes of downward error and upward error can be different. The upward error magnitude can be larger than the downward magnitude in general since the dominant interference effect is still upward even if the optimal V_{read} is used. [16] proposed the systematic optimal codes for all symmetric errors of limited-magnitude, but it is not practical in that its code rate is too low. [10] introduced the symmetric limited error correction codes which can correct only one single error, so it is not practical for flash memories, either.

Another approach is to use interference cancellation algorithms or low interference memory structures. To mitigate the cell to cell interference, [11] proposes a page architecture including LSB and MSB program schemes. There are also other approaches based on signal processing techniques to compensate communication-channel inter-symbol interference [3] [20] [18]. The signal processing methods can be efficient because they do not require redundancy to improve the error rate. [20] proposes the post-compensation algo-

rithms. It is possible to estimate the amount of the V_T shift based on the neighbor cells' programming (writing) progress, and V_{read} is adapted accordingly.

The retention problem was also addressed by using the coding schemes. Much related research has been conducted, such as WOM (write once memory) code [29], floating code [30], and rank modulation code [31] [32]. WOM codes, which was introduced by Rivest and Shamir first [29], is a coding scheme that allows information to be written in a memory cell multiple times without erasure. The optical disc is an example of WOM, and present-day flash memory can be also considered as WOM. When erasure of each cell is required, the flash memory erases one whole page by constraint of the erasure process. Therefore, the modification of a cell message is inefficient, and it can be improved by WOM codes. The capacity bounds of WOM codes are discussed in [33], while [39] and [34] address two write WOM codes and non-binary WOM codes, respectively. Generally, WOM codes do not have error correction ability and conventional error correction codes cannot be directly applied to WOM codes. Although the original messages are WOM codes, the codes that are encoded by general error correction codes lose the WOM property, and are no longer WOM codes. Therefore, new error correction codes for WOM codes are required. Error-correcting WOM-codes were first studied in [37] for a single error in a binary case, while [36] proposed triple error correction codes for binary WOM codes. [38] discussed the generalization of error-correcting WOM-codes model for the non-binary case. However, the error correcting code is not systematic code and it requires large alphabet size (the number of levels in a cell).

1.2 Scope and Organization

In this dissertation, we discuss the error correction codes and encoding schemes for reliability of NAND multi-level cell flash memories.

At first, we introduce the discussions of error models in multi-level cell flash memory and provide an overview of the factors that contribute to the MLC interference and the cell to cell interference model in chapter 2. The signal processing methods are discussed to reduce the errors caused by the interference and related codes for reliability of flash memory are also introduced such as WOM codes and asymmetric error correction codes.

In chapter 3, we propose error correcting codes that are effective for non-binary WOM codes. In particular, we focus on bidirectional error correction codes. The code treats both upward and downward errors when the error magnitude in each direction differs. One of proposed codes extends the technique of the distinct sum sets [10] to the bidirectional error correction codes. The code uses the parity check matrix which is generated from the distinct sum sets and has low encoding and decoding complexity. The other code is bidirectional limited magnitude error correction codes based on modulo operation [8] [22] [35], and extends the technique of the asymmetric error correction codes [8] to the bidirectional error correction codes. The code uses conventional non-binary error correction codes as base error correction codes. Furthermore, error correcting schemes for non-binary WOM codes are discussed, and a parity splitting method is introduced as the WOM error correcting code schemes. The advantages of the proposed methods are that these are practical and systematic

codes, and their encoding and decoding processes have low complexity. Also we introduce effective error locating limited-magnitude parity check error correction codes for the MLC flash memory error with lower complexity [41].

In chapter 4, we discuss coding schemes to lower cell-to-cell interference (C2CI). In the flash memory process, the write (programming) operation is performed only after the erase operation, and the amount of threshold voltage change is proportional to the magnitude. Therefore, to minimize the generated interference, the average magnitude needs to be decreased. Conventional minimum energy (ME) coding is related to this problem [23], because the goal of the code is to reduce the average energy, and it can be used to generate less interference. However, ME coding causes significant redundancy for uniform symbol frequency, and it results in higher costs for flash storage devices. Therefore, we propose a new coding scheme to lower the magnitude and minimize redundancy. The proposed coding scheme deals with q-ary message codes, and generates fixed length codes. Message codewords are divided into several blocks, and are modified by modulo addition with some constant to minimize the average magnitude. We also propose low energy Huffman codes based on entropy coding when the frequency of symbols is not distributed uniformly. This scheme produces variable-length codes without redundancy. We modified Huffman codes to minimize average number of high bits ('1' bits). We show that proposed codes generate optimal codewords which have minimum high bits with minimum average codeword length.

Finally, a conclusion is given in chapter 5.

Chapter 2

MLC Flash Memory Interference and Mitigation Techniques for Reliability

In this chapter, an overview of the interfernece model in multi-level cell flash memory is provided. Signal processing methods and coding schemes such as WOM codes and asymmetric error correction codes for reliability of flash memory are also introduced.

2.1 MLC flash memory and interference

A cell of the NAND flash memory is a floating gate transistor, and its threshold voltage can be programmed by injecting certain amount of charges into the floating gate [3]. The threshold voltage (V_T) is used to distinguish data levels in MLC memory. Fig. 2.1 shows the V_T distribution of 4-level multi-level cell flash memory. Several factors may change the distribution of the floating-gate threshold-voltage. These factors include cell-to-cell interference, cell leakage, temperature, program voltage (V_{pgm}) disturbance, the pass voltage

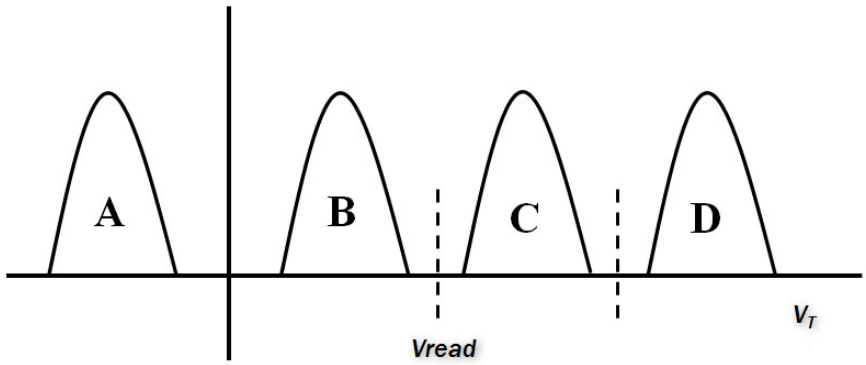

Figure 2.1 Threshold voltage distribution of 4 level multi-level cell

(V_{pass}) applied to unselected word-lines, etc [5]. One of the dominant factors is the cell-to-cell interference, which is caused by the V_T change of the neighbor cells in the programming (writing) operation. If the data of neighbor cells change, the cell-to-cell coupling interference occurs and it is shown in Fig. 2.2. In this case, the V_T shift is known to be upward (unidirectional). The cell-to-cell interference is approximately proportional to the voltage change of neighbor cells in the programming operation, but it is also affected by the structure of flash memories, the program order, and the number of levels (MLC) in a cell. The quantitative interference can be estimated by measurements and simulations.

Fig. 2.3 shows an interference model based on the parasitic capacitance between neighbor cells [20] [18]. Suppose that V_h, V_v, and V_d are the cell voltages of the horizontal, the vertical, and the diagonal neighbor cells, respectively. BL_{even} and BL_{odd} stand for the even and the odd bit lines, respectively, and WL_n stands for the nth word line.

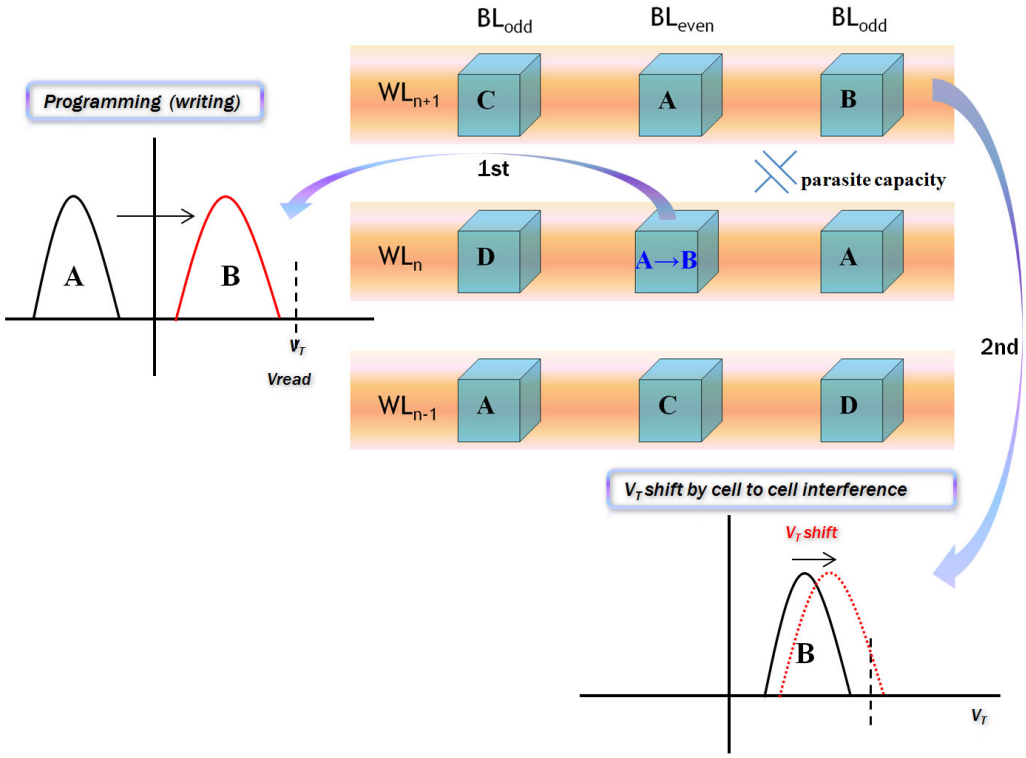

Figure 2.2 Cell to cell interference by the parasitic capacitances

The interference in terms of threshold voltage shift (ΔV_I) is given by

$$\Delta V_I = \alpha_h(\beta_{h1}\Delta V_{h1} + \beta_{h2}\Delta V_{h2}) + \alpha_v(\beta_{v1}\Delta V_{v1} + \beta_{v2}\Delta V_{v2})$$
$$+ \alpha_d(\beta_{d1}\Delta V_{d1} + \beta_{d2}\Delta V_{d2} + \beta_{d3}\Delta V_{d3} + \beta_{d4}\Delta V_{d4}) \quad (2.1)$$

where α_h, α_v, and α_d are the coupling coefficients for the horizontal, the vertical, and the diagonal neighbor cells, respectively. One cell can only be interfered by its neighbor cells which are programmed after this cell has been programmed. β_i has the binary value of '0' or '1', and it indicates whether the cell is interfered by the ith cell or not. If we assume that the

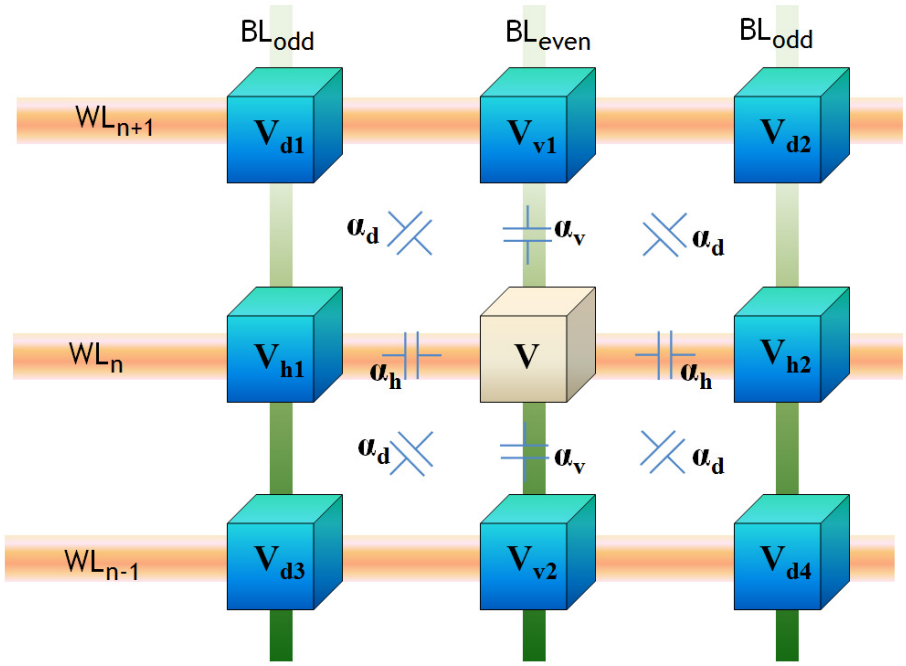

Figure 2.3 Interference model based on parasitic capacitances in a NAND flash array.

full-sequence programming strategy is being used, only after all the cells on one word line have been programmed can the next word line cells be programmed. By using an even/odd bit line structure, the cell-to-cell interference can be reduced. With the even/odd bit line structure, the even bit line cells are programmed first, and the odd bit line cells later [3]. For example, if we use a full-sequence programming strategy with the even/odd bit line structure, and the cell V (the red cell) belongs to an even bit line in Fig.1, the programming (writing) is performed in the order of $\text{WL}_{n-1}(V_{v2}, V_{d3}, V_{d4})$, $\text{WL}_n(V, V_{h1}, V_{h2})$, and $\text{WL}_{n+1}(V_{v1}, V_{d1}, V_{d2})$. V is affected by the cell to cell interference caused by its 5 neighboring cells, $V_{h1}, V_{h2}, V_{v1}, V_{d1}$, and V_{d2}. More specifically, β_{d3}, β_{v2}, and β_{d4} are 0's, and

the others are 1's. With this programming strategy, the interference in terms of the threshold voltage shift of an even bit line cell is given by

$$\begin{aligned}\Delta V_I =&\alpha_h(\Delta V_{h1} + \Delta V_{h2}) + \alpha_v(\Delta V_{v1}) \\ &+ \alpha_d(\Delta V_{d1} + \Delta V_{d2}).\end{aligned} \tag{2.2}$$

If the cell V belongs to an odd bit line, the programming order is $\text{WL}_{n-1}(V_{d3}, V_{d4}, V_{v2})$, $\text{WL}_n(V_{h1}, V_{h2}, V)$, and $\text{WL}_{n+1}(V_{d1}, V_{d2}, V_{v1})$. In this case, ΔV is affected by its 3 neighboring cells, V_{d1}, V_{d2}, and V_{v1}. In other words, β_{h1}, β_{h2}, β_{d3}, β_{v2}, and β_{d4} are 0's, and the others are 1's. The interference in terms of the threshold voltage shift of an odd bit line cell is given by

$$\Delta V_I =\alpha_v(\Delta V_{v1}) + \alpha_d(\Delta V_{d1} + \Delta V_{d2}). \tag{2.3}$$

A more realistic cell-to-cell interference model depends on the program order, page architecture, and the conventional LSB/MSB techniques [11].

For these asymmetric interference factors, the error correction codes for the asymmetric channel can be useful. Asymmetric channels have been studied for several decades [6]. The topic was studied initially for a binary asymmetric channel (Z-channel). In a Z-channel, the input and the output are binary, and 1 can be changed to 0 with probability p, but 0 cannot be changed to 1. Recently, many error correction codes for asymmetric error with limited-magnitude error are discussed [7] [8] [9]. Although the cell-to-cell interference which leads to upward errors is the dominant factor in MLC flash memories, there are also bidirec-

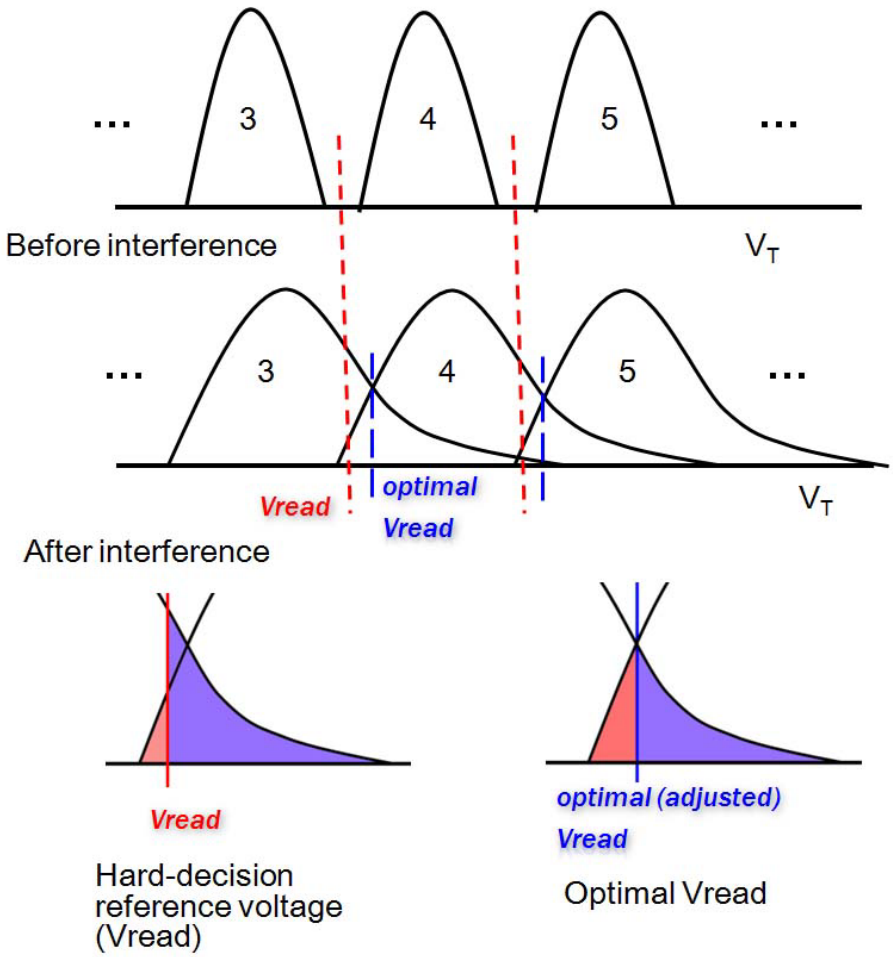

Figure 2.4 V_T shift and bidirectional errors with adjusted V_{read}.

tional (random-telegraph noise) and downward (retention noise) interference [11]. The hard-decision reference voltages for reading flash memory cells are determined based on the V_T distribution after the cell-to-cell interference, not before the cell-to-cell interference, which means the hard-decision reference voltages for reading is already near optimal. Fig. 2.4 illustrates the threshold voltage shift, and the adjusted V_{read}. After adjusting V_{read} to be

near optimal, the number of errors decreases, but the number of downward errors increases. Therefore, bidirectional errors should be considered in order to improve the BER performance. Even if the errors are bidirectional, the magnitude of the errors is still limited. The magnitudes of downward error and upward error can be different. The upward error magnitude can be larger than the downward magnitude in general since the dominant interference effect is still upward even if the optimal V_{read} is used as in Fig. 2.4.

2.2 Signal processing based interference mitigation in MLC flash memories

The signal processing methods are discussed to reduce the errors caused by the interference in this subsection [20] [18]. If we know the exact cell to cell interference values caused by the voltage changes of neighbor cells, and the exact voltage of the current cell, the controller can cancel the interference, and make a less erroneous decision. To estimate the cell to cell interference, the data of the neighbor cells need to be read first. In other words, we need to read twice to estimate the interference. However, to know the exact voltage of a cell is difficult. Since a controller can only decide whether the cell (threshold) voltage is larger or less than the read voltage (V_{read}), only a quantized version of the cell voltage instead of the precise cell voltage is available. This quantization is considered in proposing the following 2 signal processing based interference mitigation techniques.

At first, we discuss the *adaptive read voltage (ARV)* method. Most cells in flash memo-

ries are affected by the cell to cell interference.

$$V_T' = V_T + \Delta V_I + I_G \tag{2.4}$$

V_T' is the threshold voltage, V_T is the original threshold voltage, ΔV_I is cell to cell interference and I_G means the voltage shift by other interferences. Each cell has different interference due to different neighbor cell voltages, but we can apply the average value of the interference to every cell. Though it is not optimal, we can expect performance improvement. For example, if the average value of the cell to cell interference is $\mathcal{E}(\Delta V_I)$, and $V_{read} + \mathcal{E}(\Delta V_I)$ as the read voltage instead of V_{read}, some (if not all) of the cells will be corrected especially when they have large inter-cell interference. $\mathcal{E}(\Delta V_I)$ can be treated as a constant if there are a large number of cells and the data values are random. Let $\mathcal{D}(V_{read})$ be the data decision function. Each cell has q-level and V_{max} is maximum threshold voltage of the maximum level.

$$\mathcal{D}(V_{read}') = \mathcal{D}(V_{read} + \mathcal{E}(\Delta V_I)) \tag{2.5}$$

The interference in terms of the threshold voltage shift of an even bit line cell is given by $\Delta V_I = \alpha_h(\Delta V_{h1} + \Delta V_{h2}) + \alpha_v(\Delta V_{v1}) + \alpha_d(\Delta V_{d1} + \Delta V_{d2})$ and

$$\begin{aligned}
\mathcal{E}(\Delta V_I) &= \mathcal{E}\Big(\alpha_h(\Delta V_{h1} + \Delta V_{h2}) + \alpha_v(\Delta V_{v1}) + \alpha_d(\Delta V_{d1} + \Delta V_{d2})\Big) \\
&= \mathcal{E}(\Delta V)(2\alpha_h + \alpha_v + 2\alpha_d) \\
&= \frac{1 + 2 + \cdots + q - 1}{q} \cdot \frac{V_{max}}{q-1}(2\alpha_h + \alpha_v + 2\alpha_d) \\
&= \frac{V_{max}}{2}(2\alpha_h + \alpha_v + 2\alpha_d).
\end{aligned} \tag{2.6}$$

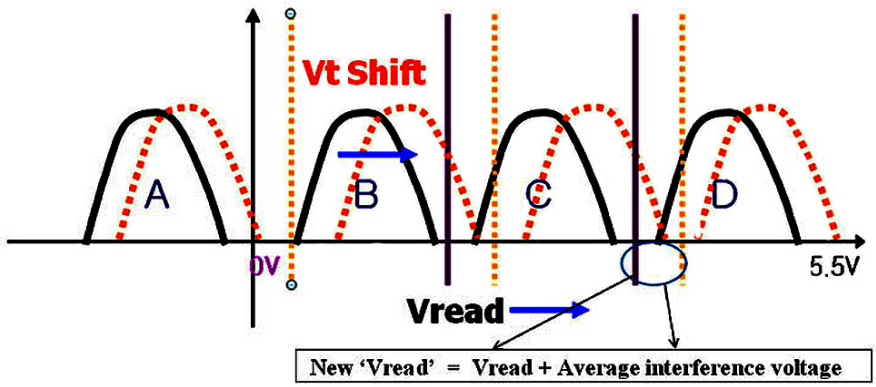

Figure 2.5 Adaptive read voltage algorithm.

As for implementation, we can used fixed $\mathcal{E}(\Delta V_I)$ for all cells, and the complexity of this method is minimal if the $\mathcal{E}(\Delta V_I)$ value is pre-computed. Fig. 2.5 shows how the ARV algorithm works.

Next, we introduce *careful cell compensation (CCC)* method. This is a kind of the post-compensation algorithm. It is possible to estimate the amount of the V_T shift based on the neighbor cells' programming (writing) progress, and V_{read} is adapted accordingly. This algorithm consists of two steps. The first step is to check whether a cell has high possibility for large V_T shift, and the 2nd step is to check whether the inter-cell interference for those cells detected in the first step is large enough, and we correct those cells which pass the tests of the two steps. We define a 'careful cell' as the cell which is expected to be erroneous. The cell to cell coupling interference is always upward (unidirectional), and a cell affected by a large interference tend to have erroneous decisions. In other words, we classify a cell with the threshold voltage near the V_{read} as a 'careful cell'. The distance from the read voltages

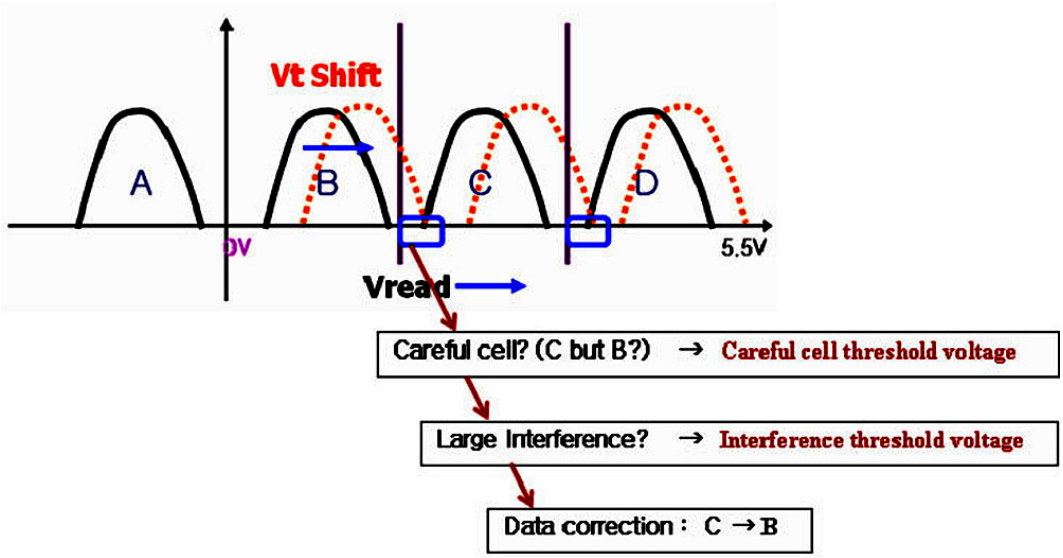

Figure 2.6 The careful cell compensation (CCC) method.

is determined by the 'careful cell threshold'. To decide whether a cell is a careful cell, we read the cell twice with the read voltage of V_{read} and $V_{read} + \tau_{cc}$ where τ_{cc} is the careful cell threshold. $\mathcal{E}(\Delta V_I)$ can be used as τ_{cc}. If two decisions are are not equal, we declare that the cell is a *careful cell*. If a cell is declared as a careful cell, we estimate the interference for the cell. Since we already obtained the data of neighbor cells in the first step (careful cell decision), the cell to cell interference can be estimated from the neighborhood data. If the interference is large enough, the cell data is likely to be in error. The threshold to determine whether the interference is large enough is called *interference threshold voltage*. Let $I_{th} = \tau_{cc} + I'_G$ be interference threshold voltage and I'_G means estimated voltage shift by other interferences. If the interference exceeds the interference threshold, $\Delta V_I > I_{th}$, this cell is assumed to be in error, and is corrected by one downward level. An example is shown in Fig.

2.6. Assume that the data of the cell is detected as C with V_{read}. If it is decided as a careful cell and the estimated interference is larger than the interference threshold, the data is corrected to B. The careful cell threshold and the interference threshold can be determined analytically from the distribution model, but empirical values obtained from simulations may work better. In this algorithm, we need to read all the cells twice, and each reading operation causes delay. There is a trade-off between bit error rate and delay. The careful cell compensation algorithm (CCC) takes the following steps.

Careful Cell Compensation (CCC) Algorithm

1) *Read the cell data twice with V_{read} and $V_{read} + \tau_{cc}$.*

1-1) *If two data values are not equal, then it is a careful cell and go to step 2.*

2) *Estimate the cell to cell interference using the neighbor data distribution of the cell.*

2-1) *If the cell to cell interference ΔV_I exceeds the interference threshold I_{th}, go to step 3.*

3) *Adjust the data value of the cell by one downward level.*

We compare the performance of two proposed algorithms, the adaptive read voltage (ARV) algorithm, the careful cell compensation (CCC) algorithm algorithm. Bit error rate (BER) is used to measure the interference mitigation performance. In the comparison of the

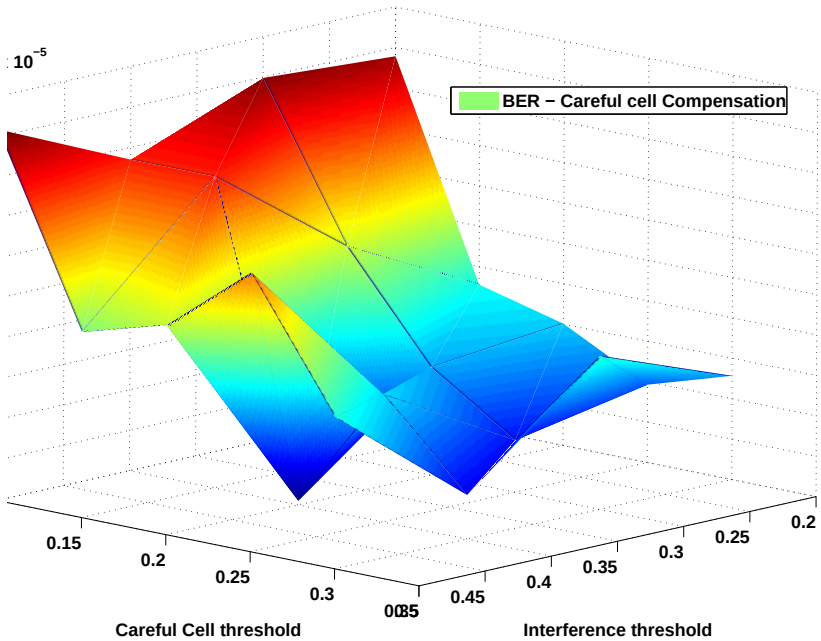

Figure 2.7 BER with respect to the two thresholds for the CCC algorithm.

ARV algorithm with the CCC algorithm, we use 4 levels with equal distribution model. The equal distribution model assumes that the every level has equal width. Before simulating the CCC algorithm, we find the careful cell threshold and the interference threshold numerically. We use an exhaustive search for the two thresholds to yield the best BER performance. In the simulation of the CCC algroithm, it is assumed that the ARV algorithm is combined with the CCC algorithm. It should be noted that we use either the no-interference case (the perfect case) or the case without interference cancellation as a benchmark. Fig. 2.7 is the 3D plot of the BER performance with respect to the careful cell threshold and the interference threshold. When the careful cell threshold voltage is 0.2V and the interference threshold

voltage is 0.4V, the best BER is achieved. To compare the BER performance of the CCC algorithm, we simulate with a fixed interference threshold of 0.4V, and the careful cell threshold is varied. In Fig. 2.8, the "perfect interference cancellation" means the case where cell to cell interference effects are removed. In this case, a threshold voltage V_T is adjusted by

$$V_T' - \Delta V_I = V_T + I_G. \tag{2.7}$$

It is observed that the BER performance of the careful cell compensation algorithm is between that of the ARV algorithm and that of the perfect interference cancellation. The CCC algorithm shows the best performance at the careful cell threshold of 0.2 V, and it is close to the no interference case.

In summary, to mitigate the interference, we propose two signal processing based algorithms: the adaptive read voltage (ARV) algorithm, the careful cell compensation (CCC) algorithm. It was shown by simulations that these algorithms are effective to reduce the effects of the interference. The ARV algorithm does not require extra data read, but its performance is not as good as the other one. The other algorithm CCC requires extra data read stages, but its performance is better than the ARV algorithm in general. If the cell data can be read only twice, the CCC algorithm appears to be promising.

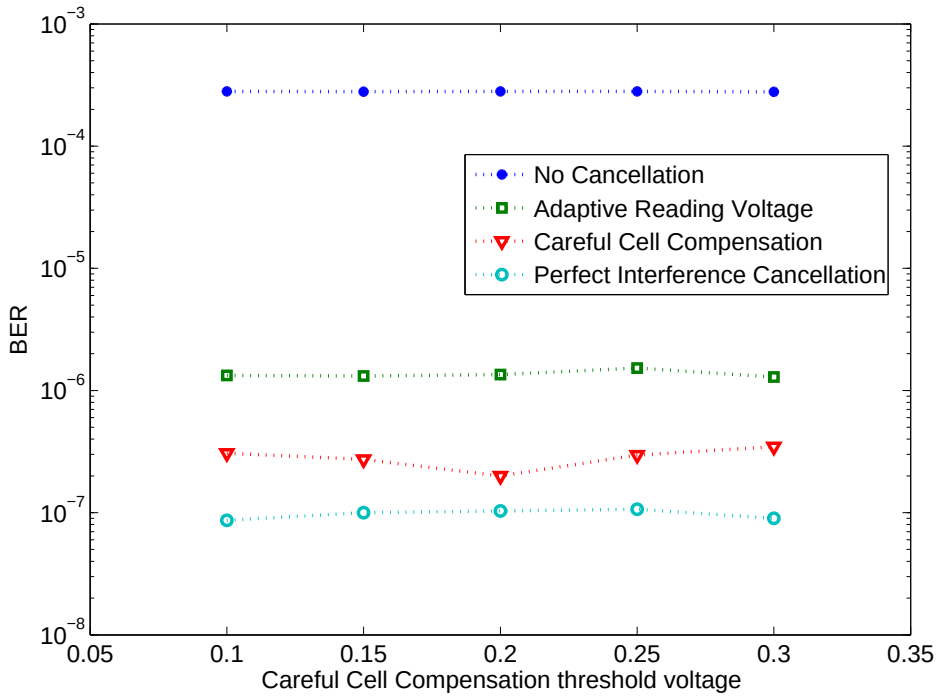

Figure 2.8 BER performance of the CCC and the ARV algorithms.

2.3 WOM codes

The longevity problem related with program/erasure cycle of flash memory has been addressed and much related codes are considered, such as WOM (write once memory) code [29], floating code [30], and rank modulation code [31]. WOM codes among them, which was introduced by Rivest and Shamir first [29], is a coding scheme that allows information to be written in a memory cell multiple times without erasure and the goal of designing WOM codes is to maximize the total amount of information written and to achieve high sum-rate [29]. In t-write WOM codes, the number of message data sets that can be written

without erasure is t. Table 2.1 introduces the example of 2-write binary WOM code [29].

Table 2.1 2-write binary WOM code example

Data	1^{st} write	2^{st} write
00	000	111
01	100	011
10	010	101
11	001	110

In the first write, 2-bit words are encoded using the 1^{st} write codebook. For example, the first messages are 01, so 100 is written in the three cells, and the second messages are 11, then 110 is written. Therefore there are no $1 \rightarrow 0$ cases, but only $0 \rightarrow 1$. If the second 2-bit word is the same as the first, there is no change to the written codeword. Non-binary WOM codes construction generates codes that have a large alphabet (q-ary). We define non-binary WOM codes construction based on [34].

Definition 1. *The t-write non-binary q-ary WOM codes $\mathcal{W}$ is specified by t pairs of encoding and decoding maps $\mathbb{E}_i$, $\mathbb{D}_i$, for $1 \leq i \leq t$.*

$\mathbb{E}_1 : \{1, \cdots, M_1\} \rightarrow \{0, \cdots, q-1\}^n$,

$\mathcal{W}_i = \mathbb{E}_i(v_i, \mathcal{W}_{i-1}) \geq \mathcal{W}_{i-1}, (i \geq 2)$.

$\mathbb{D}_1 : \{0, \cdots, q-1\}^n \rightarrow \{1, \cdots, M_i\}$,

$\mathbb{D}_1(\mathbb{E}_1(v_1)) = v_1$ *and* $\mathbb{D}_i(\mathbb{E}_i(v_i, \mathcal{W}_{i-1})) = v_i$.

$\mathbb{E}_i, \mathbb{D}_i$ are i_{th} encoding and decoding functions respectively, and v_i is the message. M_i is the alphabet size of i_{th} message v_i.

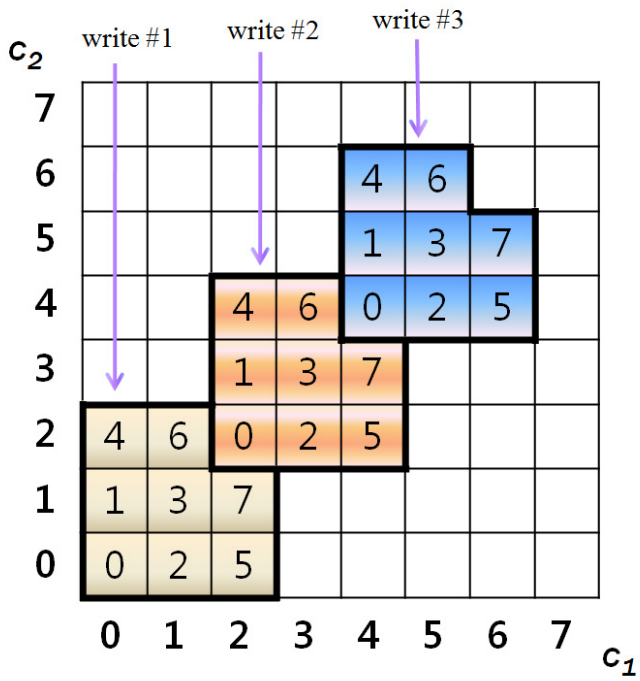

Figure 2.9 A WOM code that stores 3 bits in 2 cells writes [42].

We introduce a non-binary WOM code construction algorithm in [42]. [42] discusses the problem of 2 cell q-ary WOM codes which is addressed with a construction that uses lattice tilings. The resulting codes in [42] are shown to be within a small additive constant from the capacity. We assume that k_m input bits are written t times to n physical cells with q levels where the cell levels cannot decrease between writes in the WOM model. In an $n = 2$ code (2 cell q-ary WOM codes), the content of the memory is described by a pair $(c_1, c_2) \in \{0, \cdots, q-1\}^2$ of cell levels. By stacking 2-dimensional shapes along the main diagonal of the (c_1, c_2) plane, the code in Fig. 2.10 guarantees the re-writes. The rest of the plane outside the stack remains unused and better WOM code can be made by the

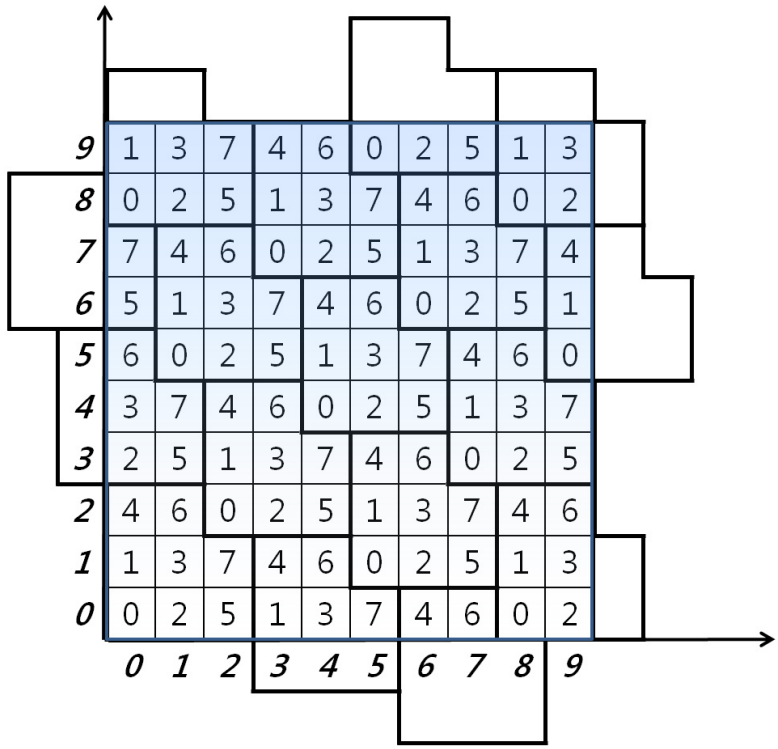

Figure 2.10 The tiling used for an improved WOM code [42].

construction to follow by utilizing the remaining cell states. To get more writes, we can take the following steps [42].

1) Tile the plane with the same basic shape from Fig. 2.9.

2) Specify update functions that traverse the tiling in a way that a certain number of writes is guaranteed for any sequence of input-value updates.

Then Fig. 2.10 is generated from Fig. 2.9 by the construction. The generated code guarantees four writes in the example.

Conventional correction codes cannot be directly applied to WOM codes, but error cor-

recting WOM codes have been conducted. [40] discusses the error correcting codes for flash coding. Flash coding schemes are related with WOM codes strongly and have been developed to maximize the number of writes before a block-erasure is needed. [40] proposes a new flash coding scheme based on error correction codes which minimize the frequency of block-erasures by using some controllable errors. The idea of the scheme is that cells whose levels are higher than others need not be increased and introduces errors which can be corrected within the error-correcting capability of the ECC. The code has also capable of additional errors or erasures. The encoding process of the idea is as follows. The encoding function is characterized by a integer $0 \leq \varrho \leq \lfloor (d-1)/2 \rfloor$. $\varkappa = (\varkappa_1, \varkappa_2, \cdots, \varkappa_n)$ is the current cell-state vector and $\vartheta = E_C(\mathbf{m}) = (\vartheta_1, \vartheta_2, \cdots, \vartheta_n)$ is the codeword in C corresponding to the message $\mathbf{m}$. $\mathcal{F}_0(\vartheta, \varkappa) = \{i : i = 1, 2, \cdots, n, \vartheta_i \neq (\varkappa_i)_2\}$. For the reduced binary vector of the new cell-state vector to be equal to the codeword $\vartheta = E_C(\mathbf{m})$, all cells with indices in $\mathcal{F}_0(\vartheta, \varkappa)$ increase the levels by 1. The scheme of [40] is not to increase the levels of those cells in which the levels are already the highest among all cells with indices in $\mathcal{F}_0(\vartheta, \varkappa)$. The paper introduce errors, ϱ which is the number of controllable errors (CE). By the error-correcting capability of the error-correcting code C, the decoder can recover the message correctly. For this purpose, Let $\mathcal{F}(\vartheta, \varkappa) \subseteq \mathcal{F}_0(\vartheta, \varkappa)$ be a subset of size min $\{|\mathcal{F}_0(\vartheta, \varkappa)|, \varrho\}$ such that for all $i \in \mathcal{F}(\vartheta, \varkappa)$ and $i' \in \mathcal{F}_0(\vartheta, \varkappa) \setminus \mathcal{F}(\vartheta, \varkappa)$, we have $\varkappa_i \geq \varkappa_{i'}$. $\mathcal{F}(\vartheta, \varkappa)$ includes indices of cells in which the levels are highest among all cells with indices belonging to $\mathcal{F}_0(\vartheta, \varkappa)$. The levels of cells with indices in $\mathcal{F}(\vartheta, \varkappa)$ will

not be increased. [40] define $t(q, \varrho)$ to be the number of guaranteed block-writes. Let d be the maximum Hamming distance between a pair of codewords in C. Then for $d > 2\varrho$ and $q \geq 2$,

$$t(q, \varrho) = 2\lfloor \frac{(q-2)\varrho}{d-2\varrho} + q - 1 \rfloor \tag{2.8}$$

From the (2.8), we notice that for [40] scheme to have more than $q-1$ writes in the worst case.

2.4 Asymmetric limited-magitude error correction codes based on distinct sum set

The V_T shift which is caused by the cell-to-cell interference is known to be upward and the error correction codes for the asymmetric channel can be useful for these asymmetric interference factors. Recently, many error correction codes for asymmetric error with limited-magnitude error are discussed [7] [8] [9] [10].

[10] proposes error correction codes which correct single asymmetric limited magnitude errors, that is, l-asymmetric error correcting codes(l-AEC). The proposed codes achieve better performance than the ones given in [8] for the single error case, and it is based on distinct sum sets. For integer i, j, where $i \leq j$, we let $[i, j] = \{i, i+1, i+2, \ldots, j\}$. [10] defines that a $B_\lambda[l](q)$ sequence of length m is a sequence of m distinct positive integers $b_0, b_1, \ldots, b_{m-1}$ such that all sums $\left(\sum_{j=1}^{\lambda} a_j b_{i_j} \right)$ mod q are distinct, where $0 \leq i_1 < i_2 < \cdots < i_\lambda \leq m-1$ and $a_j \in [0, l]$. This sequence or a set is also called distinct sum se-

quence or set in this dissertation. [10] discusses only $\lambda = 1$ case. Based on $B[l](q)$, the error correction codes can be constructed. Let $\mathbf{H}$ be the $r \times n$ parity check matrix whose columns are all possible r length q-ary vectors whose first nonzero element belongs to $B[l](q)$. Let $\mathcal{C}(B[l](q))$ be the null space of $\mathbf{H^T}$.

Theorem 1. *If gcd* $(q, l!) = 1$, $\mathcal{C}(B[l](q))$ *can correct a single asymmetric error limited magnitude* l *[10].*

Proof and related discussions are shown in [10]. Let $\mathbf{c} \in \mathcal{C}$ and $\boldsymbol{\epsilon}$ be a vector of errors with i_{th} component equal to nonzero integer and all other components equal to 0.

$$(\mathbf{c} + \boldsymbol{\epsilon})\mathbf{H^T} = \mathbf{c}\mathbf{H^T} + \boldsymbol{\epsilon}\mathbf{H^T} = \boldsymbol{\epsilon}\mathbf{H^T} \tag{2.9}$$

The syndromes $\boldsymbol{\epsilon}\mathbf{H^T}$ are all distinct and the error can be corrected.

Chapter 3

Error Correction Codes for Flash Memories

3.1 Introduction

The error increases with the number of levels in the cell [2] [4] and another problem related to degradation of the retention characteristics (a retention problem) also occurs with many cycles of program/erasure in MLC flash memories. Conventional error correcting codes have been used for solving reliability problems. Error correction codes for asymmetric or symmetric channels with limited-magnitude error were discussed in [7], [8], [16], and [10] for flash memories. There are also bidirectional (random-telegraph noise) and downward interference [11] and the discussion of bidirectional error correcting codes can be meaningful for reliability of flash memory. Another issue is the retention problem and WOM codes, which was initially introduced by Rivest and Shamir [29], is a coding scheme that allows

information to be written in a memory cell multiple times without erasure. However, conventional error correction codes cannot be combined with WOM codes directly and new error correction codes for WOM codes are required.

In this chapter, error correcting codes that are suitable to practical flash memory devices and non-binary WOM codes are discussed. We deal with bidirectional error correction codes and these research are not conducted much as asymmetric errors and symmetric errors. One of proposed codes extends the technique of the distinct sum sets [10] to bidirectional error correction codes, and the other code is bidirectional limited magnitude error correction codes based on modulo operation [8] [22] [35]. The parity code constructions for systematic WOM codes are also discussed. The proposed codes have encoding and decoding process with low complexity, which is efficient for non-binary WOM codes. Furthermore, we discuss asymmetric and symmetric error locating limited-magnitude parity check error correction codes for the MLC flash memory error with lower complexity encoding [41].

3.2 Bidirectional error correction codes for non-binary WOM codes based on distinct sum sets

3.2.1 Bidirectional error correction codes based on distinct sum sets

As described in the previous chapter, bidirectional error correction codes are efficient for practical systems. In the notation of (l_u, l_d), l_u and l_d represent the maximum upward error magnitude and the maximum downward error magnitude, respectively. Fig. 3.1 (a), (b),

and (c) illustrate the difference of various error types. The asymmetric limited magnitude errors of (a) are discussed in [8] [10], and the symmetric limited magnitude errors of (b) are considered in [16] [10]. However, little has been studied for the bidirectional limited magnitude errors of (c) [22] [35].

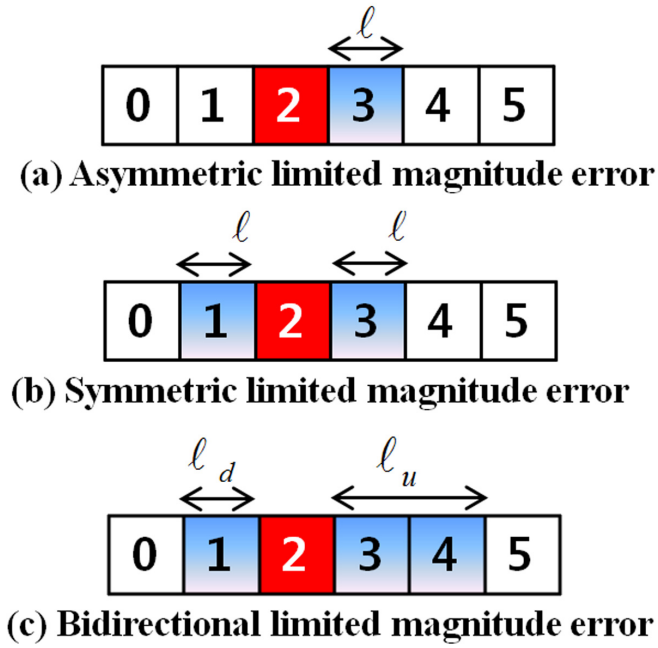

Figure 3.1 Various limited magnitude error types and (l_u, l_d) bidirectional error channel.

The bidirectional error correction codes which extend the technique of the limited magnitude error correction codes [10] to bidirectional error correction codes will be introduced. We define that a *bidirectional distinct sum set* $\Phi_\lambda(l_u, l_d)$ of length m is a sequence of m distinct positive integers $\phi_0, \phi_1, \ldots, \phi_{m-1}$ such that all sums

$$\left(\sum_{j=1}^{\lambda} l_j \phi_{i_j} \right) \bmod q \tag{3.1}$$

are distinct, where $0 \leq i_1 < i_2 < \cdots < i_\lambda \leq m-1$ and $-l_d \leq l_j \leq l_u$. $\lambda = 1$ case will be discussed at first.

Definition 2. *When l_u, l_d are given, we define a set*

$$\Phi_{(l_u,l_d)} = \{1\} \cup \{\omega^i | \textit{max}(l_u, l_d) < \omega^i \leq \omega \cdot \textit{max}(l_u, l_d), \\ \omega : \textit{prime}, i\textit{: positive integer}\}, \tag{3.2}$$

and a sorting function $\Omega(X, m) = \{x_k | x_1 < x_2 < \cdots < x_m, x_1, \cdots, x_m$ are m smallest elements in X, $k = 1, \cdots, m\}$ for $x_i \in X$ and $|\Omega(X, m)| = m$.

We then obtain a set of m elements, which is given by

$$\Phi^m_{(l_u,l_d)} = \Omega(\Phi_{(l_u,l_d)}, m) = \{\phi_1, \phi_2, \cdots, \phi_m\} \tag{3.3}$$

for $\phi_1 < \phi_2 < \cdots < \phi_m$.

The set $\Phi^m_{(l_u,l_d)}$ is modified by the following conditions.

i) Let τ_1 and τ_2 be the two prime factors, which are not included in $\Phi^m_{(l_u,l_d)}$ and less than max(l_u, l_d). If there exist $\tau_1^\alpha \tau_2^\beta < \phi_m$, $\tau_1^\alpha \tau_2^\beta$ replaces ϕ_m (α, β : positive integers).

ii) When $\tau^\alpha \in \Phi^m_{(l_u,l_d)}$ with prime τ and τ^αmax$(l_u, l_d) < \tau^\beta < \phi_m$, τ^β replaces ϕ_m.

iii) When $\tau_1^\alpha \in \Phi^m_{(l_u,l_d)}$ with prime τ_1, let τ_2 be the prime factor which is not included in $\Phi^m_{(l_u,l_d)}$ and less than max(l_u, l_d). If max$(l_u, l_d) < \tau_1^\beta \tau_2^\zeta < \phi_m$ and τ_1^βmax$(l_u, l_d) < \tau_1^\chi < \phi_m$, $\tau_1^\beta \tau_2^\zeta$ and τ_1^χ replace the τ_1^α and ϕ_m ($\alpha, \beta, \zeta, \chi$: positive integers).

For example, when $l_u = 7, l_d = 3, m = 6$, $\Phi^m_{(l_u,l_d)} = \{1, 8, 9, 11, 13, 17\}$ at first, and we get ϕ_m=17. With case iii) of Definition 2, the prime factor 5 is not included in $\Phi^m_{(l_u,l_d)}$, and $5 < \max(l_u, l_d)$, so $\tau_2 = 5$. $2^3 \in \Phi^m_{(l_u,l_d)}$ and $\tau_1 = 2$. $\max(l_u, l_d) < 2 \cdot 5 < \phi_m$ and $2\max(l_u, l_d) < 2^4 < 17$. Then, $2 \cdot 5 = 10$ and $2^4 = 16$ replace 8 and 17. Finally, $\Phi^m_{(l_u,l_d)} = \{1, 9, 10, 11, 13, 16\}$.

Definition 3. *If ϕ_m in distinct sum set $\Phi^m_{(l_u,l_d)}$ which satisfies (3.1) is the minimum value, the set is defined as* a minimum magnitude distinct sum set.

Theorem 2. *$\Phi^m_{(l_u,l_d)} = \{\phi_1, \phi_2, \cdots, \phi_m\}$ in Definition 2 are (l_u, l_d) distinct sum sets which satisfy (3.1) for all $q \geq \phi_m(l_u + l_d) + 1$.*

Proof. Let $1 \leq v, w \leq l_u$ or $-l_d \leq v, w \leq -1$ (v, w : integer, $v \neq w$), $\phi_i, \phi_j \in \Phi^m$, $\phi_i \neq \phi_j$, $1 \leq i, j \leq m$, and each of ϕ_i and ϕ_j has only one prime factor. Let us prove by contradiction. We assume that there exist v, w, ϕ_i, ϕ_j which satisfy $v\phi_i = w\phi_j$. $v\phi_i = w\phi_j$ leads to $\frac{v\phi_i}{w\phi_j} = 1$. ϕ_i and ϕ_j are relatively prime, so $|v| = \phi_j$ and $|w| = \phi_i$. According to the assumption in Definition 2, ϕ_i or $\phi_j > \max(l_u, l_d) > |v|, |w|$, so $|v| \neq \phi_j$ or $|w| \neq \phi_i$, which is a contradiction. Therefore $v\phi_i \neq w\phi_j$. Both $v\phi_i$, $w\phi_j < \phi_m \max(l_u, l_d) < \phi_m(l_u + l_d) + 1$, $v\phi_i \neq w\phi_j$ leads to $v\phi_i \neq w\phi_j \bmod q$ if $q \geq \phi_m(l_u + l_d) + 1$.

When ϕ_i have two prime factor for the case i) of Definition 2, $\tau_1^\alpha \tau_2^\beta$ will be the element.

$v\tau_1^{\alpha}\tau_2^{\beta} \neq w\phi_j$ for any j except i, because both prime factor τ_1, τ_2 are not included in any ϕ_i. For the case ii), if ϕ_i and ϕ_j have a common prime factor, ϕ_i and ϕ_j become τ^{α} and τ^{β}. Because $\tau^{\alpha}\max(l_u, l_d) < \tau^{\beta} < \phi_m$, $|v|\tau^{\alpha} < |w|\tau^{\beta}$ and $v\phi_i \neq w\phi_j$. For the case iii), if $\tau_1^{\beta}\tau_2^{\zeta}$ and τ_1^{χ} have a common prime factor τ_1, $\tau_1^{\beta}\max(l_u, l_d) < \tau_1^{\chi}$ leads to $v\tau_1^{\beta}\tau_2^{\zeta} \neq w\tau_1^{\chi}$. Let $-l_d \leq v \leq -1, 1 \leq w \leq l_u$. $q - l_d\phi_m \leq (v\phi_i \bmod q) < q$ and $0 < w\phi_j \leq l_u\phi_m$. According to the assumption of $q \geq \phi_m(l_u + l_d) + 1$, we have $q - l_d\phi_m > l_u\phi_m$. Because $\{0 < w\phi_j \leq l_u\phi_m < q - l_d\phi_m \leq v\phi_i < q\} \bmod q$, $v\phi_i \neq w\phi_j \bmod q$ is always valid.

Remark 1. *It was shown by exhaustive computer search that $\Phi_{(l_u,l_d)}$ given by Definition 2 is* a minimum magnitude distinct sum set *when $m \leq 7$ and max$(l_u, l_d) \leq 8$.*

Table 3.1 Modification cases in generating minimum magnitude distinct sum sets

max(l_u, l_d)	m=3	m=4	m=5	m=6	m=7
2				ii	ii
3			iii		
4	i	i			
5	i				iii
6					iii
7				iii	iii
8	i	i	i		

As the parameters of $\max(l_u, l_d), m$ vary, the modification case of Definition 2 to generate the minimum magnitude distinct sum set is shown in Table 3.1. i,ii,iii represent the case

i), case ii), and case iii) of Definition 2, respectively. The empty entries in Table II represent a case where minimum magnitude distinct sum sets are obtained by (3.2) and (3.3) only without any additional modification with the max(l_u, l_d) and m.

Bidirectional error correction codes can be constructed by the $\Phi^m_{(l_u,l_d)}$. Let $\mathbf{H}$ be the $r \times n$ parity check matrix, the columns of which are all possible r length q-ary vectors where the first nonzero element belongs to $\Phi^m_{(l_u,l_d)}$. If $\mathcal{C}(\Phi^m_{(l_u,l_d)})$ be the null space of $\mathbf{H^T}$, $\mathcal{C}(\Phi^m_{(l_u,l_d)})$ can correct a bidirectional error. If m and (l_u, l_d) are given and $q \geq \tilde{q}$ in the

Table 3.2 The parameters of $\Phi^m_{(l_u,l_d)}$

	$\tilde{q}$				
(l_u, l_d)	m=2	m=3	m=4	m=5	m=6
(1,0)	3	4	6	8	12
(1,1)	5	7	11	15	23
(2,0)	7	9	11	15	23
(2,1)	10	13	16	22	34
(2,2)	13	17	21	29	45
(3,0)	13	16	22	28	34
(3,1)	17	21	29	37	45
(3,2)	21	26	36	46	56
(3,3)	25	31	43	55	67
(4,0)	21	25	29	37	45
(4,1)	26	31	36	46	56
(4,2)	31	37	43	55	67
(4,3)	36	43	50	64	78
(4,4)	41	49	57	73	89

Table 3.2, $\Phi^m_{(l_u,l_d)}$ generated by Theorem 2 can correct a bidirectional error.

The set $\Phi^m_{(l_u,l_d)}$ generated by Theorem 2 is optimal for special q and (l_u, l_d). An optimal

set in this chapter means that m, the number of element in $\Phi^m_{(l_u,l_d)}$ is maximized with given parameters q and (l_u, l_d), and it can be obtained by an exhaustive search. If q equals to

Table 3.3 The parameters which proposed method generates the optimal set

(l_u, l_d)	$\tilde{q}$
(2,0)	12
(2,1)	13, 14, 15, 17, 18, 19, 21
(2,2)	20, 21, 24, 28
(3,0)	16, 23
(3,1)	21, 22
(3,2)	26, 28, 29, 31, 36
(4,0)	29
(4,1)	31, 33, 37, 39
(4,2)	38
(4,3)	43, 51, 53, 54

$\tilde{q}$ in the Table 3.3, the proposed method generates the optimal set. $\Phi^m_{(l_u,l_d)}$ generated by Theorem 2 can be used for the construction of symmetric error correction codes by setting $l_u = l_d$. [10] also proposed following method generating a B-set which can be used for the symmetric error. Note that

$$B = \{i(2l+1)+1 | i \in [0, m-1]\} \tag{3.4}$$

is a $B([-l, l])(q)$ set for $q = 2p(l+1)$ [10]. $B[-l, l](q)$ is a sequence such that all sums are distinct in (3.1), where $l_j \in [-l, l]$. Our method for a symmetric error is compared to a method in [10] in Table 3.4. The proposed method can construct the symmetric error correction code for all $q \geq \tilde{q}$, while the method in [10] generate the symmetric code when only $q = \tilde{q}$. In addition to that, a lower $\tilde{q}$ is more efficient clearly, and the proposed method

Table 3.4 A comparison of two methods for symmetric error correction code

Proposed		$\tilde{q}$			
	m	(1,1)	(2,2)	(3,3)	(4,4)
	2	5	13	25	41
	3	7	17	31	49
	4	11	21	43	57
	5	15	29	55	73
[10]	m	(1,1)	(2,2)	(3,3)	(4,4)
	2	9	25	49	99
	3	9	25	49	99
	4	15	25	49	99
	5	15	25	49	99

produces lower $\tilde{q}$ in most cases than the method in [10]. This shows the advantage of the proposed method.

There is discussion only of single error correction codes in [10], and bidirectional double error correction codes based on distinct sum sets can be considered. A new set $\Psi = \{\psi_1, \psi_2, \cdots, \psi_m\}$ is defined for double error correction.

Remark 2. *For double error correction, there is a constraint that both* $(\alpha_x\psi_i + \alpha_y\psi_j \bmod q)$ *and* $(\alpha_z\psi_k \bmod q)$ *should be all distinct for any* $\psi_i, \psi_j, \psi_k \in \Psi, i \neq j \neq k$, *and* $-l_d \leq \alpha_x, \alpha_y, \alpha_z \leq l_u$ *with integers* $i, j, k, \alpha_x, \alpha_y, \alpha_z$, *and* l_u, l_d.

Theorem 3. *If* $X = \{\omega^i | \omega : prime, i = 0, \cdots, m-1, \omega > max(l_u, l_d)\}$, X *is a* Ψ *set if* $q > (l_u + l_d)\omega^{m-2}(\omega + 1)$.

Proof. At first, we prove $\alpha_x\psi_i + \alpha_y\psi_j \neq \alpha_z\psi_k + \alpha_w\psi_o$ when all α are positive integers ($0 < \alpha \leq l_u < \omega$) and $k > o, i > j, i \neq j \neq k \neq o$. The prime factor of ψ is ω, and let $\psi_i = \omega^i \in \Psi$. If we assume $\alpha_x\psi_i + \alpha_y\psi_j = \alpha_z\psi_k + \alpha_w\psi_o$, then $\alpha_x\omega^i + \alpha_y\omega^j = \alpha_z\omega^k + \alpha_w\omega^o$, and $\omega^j(\alpha_x\omega^{i-j} + \alpha_y) = \omega^o(\alpha_z\omega^{k-o} + \alpha_w)$. $\alpha_x\omega^{i-j} + \alpha_y < \alpha_x\omega^{i-j} + \omega \leq \alpha_x\omega^{i-j} + \omega^{i-j} = (\alpha_x + 1)\omega^{i-j} \leq \omega^{i-j+1}$. Then $\omega^i < \omega^j(\alpha_x\omega^{i-j} + \alpha_y) < \omega^{i+1}$ and $\omega^k < \omega^o(\alpha_z\omega^{k-o} + \alpha_w) < \omega^{k+1}$. Because $\omega^j(\alpha_x\omega^{i-j} + \alpha_y) = \omega^o(\alpha_z\omega^{k-o} + \alpha_w)$, i should be equal to k, which is a contradiction to the assumption of $i \neq j \neq k \neq o$, so we have

$$\alpha_x\psi_i + \alpha_y\psi_j \neq \alpha_z\psi_k + \alpha_w\psi_o. \tag{3.5}$$

Let us consider the case that all α are negative integers. From (3.5), $-\alpha_x\psi_i - \alpha_y\psi_j \neq -\alpha_z\psi_k - \alpha_w\psi_o$. Let $\tilde{\alpha}$ be $-\alpha$, then $\tilde{\alpha_x}\psi_i + \tilde{\alpha_y}\psi_j \neq \tilde{\alpha_z}\psi_k + \tilde{\alpha_w}\psi_o$, and the constraint is also valid in this case. In the case that two of four α are negative and the others are positive, $\alpha_x\psi_i - \alpha_w\psi_o \neq \alpha_z\psi_k - \alpha_y\psi_j$ from (3.5) and we have $\alpha_x\psi_i + \tilde{\alpha_w}\psi_o \neq \alpha_z\psi_k + \tilde{\alpha_y}\psi_j$. For the case that three α are positve and the other is negative, we prove $\alpha_x\omega^i + \alpha_y\omega^j \neq \alpha_z\omega^k + \tilde{\alpha_w}\omega^o$. If we assume $\alpha_x\omega^i + \alpha_y\omega^j = \alpha_z\omega^k + \tilde{\alpha_w}\omega^o$, $\alpha_x\omega^i + \alpha_y\omega^j + \alpha_w\omega^o = \alpha_z\omega^k$. k should be larger than i, j, o and assume $i > j > o$ (the order can be changed). We get $\alpha_z\omega^k - \alpha_x\omega^i - \alpha_y\omega^j = \alpha_w\omega^o$. Dividing by ω^o, we have $\alpha_z\omega^{k-o} - \alpha_x\omega^{i-o} - \alpha_y\omega^{j-o} = \alpha_w$, which is factorized to

$$\omega^{j-o}(\alpha_z\omega^{k-j} - \alpha_x\omega^{i-j} - \alpha_y) = \alpha_w, \tag{3.6}$$

and $\omega^{j-o}(\omega^{i-j}(\alpha_z\omega^{k-i} - \alpha_x) - \alpha_y) = \alpha_w$. $\alpha_z\omega^{k-i} \geq \omega > \alpha_x$, so $\alpha_z\omega^{k-i} - \alpha_x \geq 1$ and $\omega^{i-j}(\alpha_z\omega^{k-i} - \alpha_x) - \alpha_y \geq 1$. Consequently, $\omega^{j-o}(\alpha_z\omega^{k-j} - \alpha_x\omega^{i-j} - \alpha_y) > \alpha_w$ which is contradictory to (3.6). Therefore, $\alpha_x\psi_i + \alpha_y\psi_j \neq \alpha_z\psi_k + \tilde{\alpha_w}\psi_o$.

Next, we prove $\alpha_x\psi_i + \alpha_y\psi_j \neq \alpha_z\psi_k$ ($0 < \alpha \leq l_u < \omega$). Let us assume that $\alpha_x\psi_i + \alpha_y\psi_j = \alpha_z\psi_k$, $k > i > j > 0$, $i \neq j \neq k$, and $\alpha_x\omega^i + \alpha_y\omega^j = \alpha_z\omega^k$. Dividing by ω^j, we have $\alpha_z\omega^{k-j} - \alpha_x\omega^{i-j} += \alpha_y$, which is factorized to

$$\omega^{i-j}(\alpha_z\omega^{k-i} - \alpha_x) = \alpha_y. \tag{3.7}$$

Because $\omega^{i-j} \geq \omega$ and $\alpha_z\omega^{k-i} \geq \omega > \alpha_x$, we have $\alpha_z\omega^{k-i} - \alpha_x \geq 1$. It leads to $\omega^{i-j}(\alpha_z\omega^{k-i} - \alpha_x) \geq \omega > \alpha_y$, we then have $\omega^{i-j}(\alpha_z\omega^{k-i} - \alpha_x) \neq \alpha_y$, which is contradictory to (3.7). Thus, we conclude that

$$\alpha_x\psi_i + \alpha_y\psi_j \neq \alpha_z\psi_k. \tag{3.8}$$

In the case that two of three α are negative and the other is positive, $\alpha_x\psi_i + \alpha_y\psi_j \neq \alpha_z\psi_k$ from 3.8 and $\alpha_x\psi_i \neq \alpha_z\psi_k - \alpha_y\psi_j$. Then we have $\alpha_x\psi_i \neq \alpha_z\psi_k + \tilde{\alpha_y}\psi_j$.

Because $\max(\alpha_x\psi_i + \alpha_y\psi_j) < l_u\omega^{m-1} + l\omega^{m-2} = l_u\omega^{m-2}(\omega + 1)$ and $\min(\alpha_x\psi_i + \alpha_y\psi_j) > -l_d\omega^{m-1} - l_d\omega^{m-2} = -l_d\omega^{m-2}(\omega + 1)$, we have $(\alpha_x\psi_i + \alpha_y\psi_j \neq \alpha_z\psi_k + \alpha_w\psi_o) \bmod q$ when $q > (l_u + l_d)\omega^{m-2}(\omega + 1)$.

The construction of $\mathbf{H}$ for single error correction is shown in [10], but it is not possible for double error correction, and it needs to consider new parity check matrix construction

$$\mathbf{H} = \begin{bmatrix} \omega & \cdots & \omega & \omega^2 & \cdots & \omega^2 & \omega^3 & \cdots & \omega^m \\ 1 & \cdots & k & k+1 & \cdots & k(k+1) & k^2+k+1 & \cdots & k(k^{m-1}+\cdots+1) \end{bmatrix} \quad (3.11)$$

---

for double error correction codes. Let us consider the case where the length of parity is $r = 2$. For l_1 and l_2 which are the magnitudes of double errors, we have

$$\mathbf{H} \cdot (\mathbf{c} + \mathbf{e})^{\mathbf{T}} = \mathbf{H} \cdot \mathbf{e}^{\mathbf{T}}$$

$$= \begin{bmatrix} \cdots & \psi_j & \psi_j & \cdots & \psi_k & \psi_k & \cdots \\ \cdots & \iota_{j1} & \iota_{j2} & \cdots & \iota_{k1} & \iota_{k2} & \cdots \end{bmatrix} \begin{bmatrix} \vdots \\ l_1 \\ \vdots \\ l_2 \\ \vdots \end{bmatrix},$$

where $0 \leq l_1, l_2 \leq l$, $\psi \in \Psi$, and ι is an integer. For all distinct syndromes,

$$l_1\iota_{j1} + l_2\iota_{k2} \neq l_1\iota_{j2} + l_2\iota_{k1} \quad (3.9)$$

$$l_1(\iota_{j1} - \iota_{j2}) \neq l_2(\iota_{k1} - \iota_{k2}), \quad (3.10)$$

where ι_{ji} is an integer element of a *distinct distance set*. If $l_1 = \pm 1$ and $l_2 = \pm 1$, we have $\iota_{j1} - \iota_{j2} \neq \pm(\iota_{k1} - \iota_{k2})$. For instance, let $|\Psi| = m, l_1 = \pm 1, l_2 = \pm 1, q >max(2\omega^{m-1}(\omega + 1), k^m + k^{m-1} + \cdots + k^2 + k)$, it is possible to construct a parity check matrix of $2 \times km$ for double error correction based on distinct sum set, which is given in (3.11).

3.2.2 Error correction coding schemes for WOM codes based on distinct sum sets

WOM codes do not have error correction ability generally and conventional error correction codes cannot be applied directly to the WOM codes. Although the original messages are WOM codes, the codes that are encoded by general error correction codes lose the WOM property, and are no longer WOM codes. Even if systematic error correction codes are used to keep the property of WOM codes, the parity code part does not guarantee t-write WOM. Therefore, we proposed new methods for error correction codes for WOM codes based on distinct sets. To construct the codes, a parity splitting method is introduced.

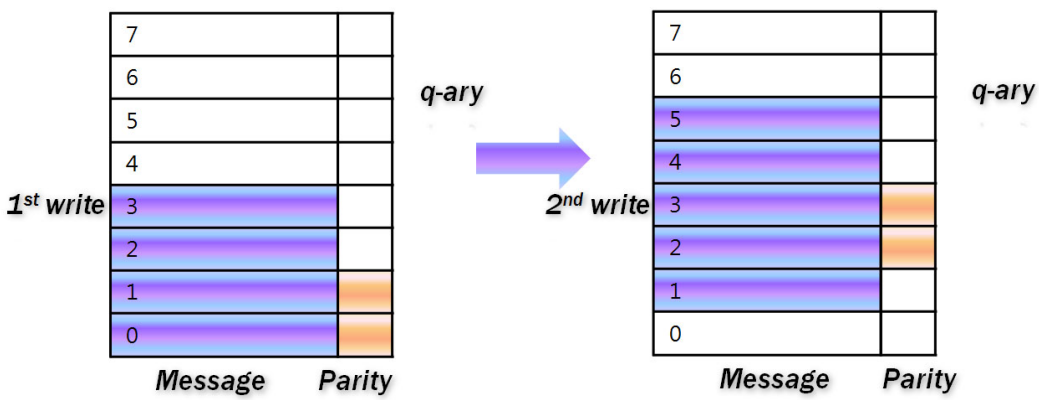

Figure 3.2 Systematic error correction codes structure for WOM code.

Let $\mathcal{C}(\Phi)$ be the null space of $\mathbf{H}^{\mathbf{T}}$ and $\mathbf{c} \in \mathcal{C}(\Phi)$. $\mathbf{c}$ is systematic code, and can be divided by a WOM message part $\mathcal{W}$ and a parity part $\mathcal{P} = [p_1\ p_2 \cdots\ p_r]$. Because $\mathcal{P}$ is not WOM code, it should be converted $\hat{\mathcal{P}}$ which satisfies the constraint of WOM code. If $\mathbb{E}_i^p, \mathbb{D}_i^p$ are WOM encoding and decoding function for parity parts, $\hat{\mathcal{P}}_i = \mathbb{E}_i^p(\mathcal{P}_i, \hat{\mathcal{P}}_{i-1}) \geq \hat{\mathcal{P}}_{i-1}$ should

be valid. Furthermore, if there is u errors within (l_u, l_d) in $\hat{\mathcal{P}}_i$, $\mathcal{P}_i$ obtained by $\mathbb{D}_i^p(\hat{\mathcal{P}}_i)$ should have u or less errors within (l_u, l_d). A simple example of $\mathbb{E}_i^p$ can be introduced for a single error. The parity codes $\mathcal{P}$ can be converted to a sum of s symbols with lower alphabet size, $p_i = p_i^1 + p_i^2 + \cdots + p_i^s$. $\Lambda(p_i^k)$ represents the alphabet size of p_i^k. The parameters s and $\Lambda(p_i)$ of each cell should be determined with $\Lambda(p_i) \leq \Lambda(p_i^1) + \cdots + \Lambda(p_i^s)$. If an (l_u, l_d) error occurs at the j_{th} parity symbol, $p_i - l_d \leq p_i^1 + p_i^2 + \cdots + (p_i^j + l) + \cdots + p_i^s = p_i + l \leq p_i + l_u$, and an error can be corrected by (l_u, l_d) bidirectional error correction codes. Let us assume $\Lambda(p_1) = \cdots = \Lambda(p_s) = \mu$.

Remark 3. $s \geq \lceil \frac{q}{\lfloor q/t \rfloor} \rceil$, *if* $\Lambda(p_1) = \cdots = \Lambda(p_s) = \mu$.

$\mu t \leq q$ leads to $\mu \leq \lfloor \frac{q}{t} \rfloor$. Because $s\mu$ represents the total alphabet size of μ-ary s symbol, and $s\mu$ should be larger than q, so $s \geq \lceil \frac{q}{\lfloor q/t \rfloor} \rceil$. Fig.3.2 shows the example of systematic error correction codes for non-binary WOM codes. The original message does not lose its WOM property due to the systematic codes, and the parity code part can guarantee t-write WOM. This method can only be used for single error correction.

The encoding and the decoding algorithms of the proposed codes based on distinct sum sets are described as follows.

Bidirectional distinct sum set-based non-binary WOM error correction codes (BD-WECC)

Encoding

Input : the i_{th} message codeword $c_i, i = 1, \cdots, t$.

Output : $[\mathcal{W}_i \ \hat{\mathcal{P}}_i], i = 1, \cdots, t$.

$\mathcal{W}_i$: the i_{th} encoded WOM codeword of c_i message

$\hat{\mathcal{P}'}_i$: the i_{th} parity code for WOM codeword

(Initialization) $i = 1$.

1) *Generate the parity check matrix $\mathbf{H}$ based on distinct-set $\Phi^m_{(l_u,l_d)}$*

2) *Obtain non-binary WOM codes using encoding maps.*

If $i = 1$, $\mathcal{W}_1 = \mathbb{E}_1(c_1)$.

or $i \geq 2$, $\mathcal{W}_i = \mathbb{E}_i(c_i, \mathcal{W}_{i-1})$

3) *Generate codes which is null space of $\mathbf{H^T}$ based on proposed bidirectional codes using $\Phi^m_{(l_u,l_d)}$ and obtain r length parity codes $\mathcal{P}_i$.*

4) *Encode r length parity codes into rs length WOM parity codes by $\hat{\mathcal{P}}_i = \mathbb{E}^p_i(\mathcal{P}_i, \hat{\mathcal{P}}_{i-1})$.*

5) *Systematic encoded codewords $[\mathcal{W}_i \ \hat{\mathcal{P}}]$ are written to the q-ary memory cell.*

6) *If $i = t$, go to step 7, else $i \leftarrow i + 1$ and go to step 1.*

7) *Erase the cells, $i \leftarrow 1$ and go to step 1.*

Decoding

(Initialization) Received codeword $[\mathcal{W}_i' \ \hat{\mathcal{P}}_i'] = [\mathcal{W}_i \ \hat{\mathcal{P}}] + \boldsymbol{\epsilon}$*, where* $\boldsymbol{\epsilon} = (\epsilon_1, \ldots, \epsilon_n)$ *is the error vector with each integer component within* (l_u, l_d)*,*

1) *Obtain the decoded parity codes by* $\mathcal{P}_i' = \mathbb{D}_i^p(\hat{\mathcal{P}}_i')$.

2) *Generate the syndrome by* $\mathbf{H} \cdot [\mathcal{W}_i' \ \mathcal{P}_i']$.

3) *Estimate* $\boldsymbol{\epsilon'}$*, the location and the magnitude of an error by syndrome.*

4) *The corrected WOM encoded message* $\mathcal{W}_i$ *is obtained by* $\mathcal{W}_i = \mathcal{W'}_i - \boldsymbol{\epsilon'}$.

5) *The original message is decoded by* $c_i = \mathbb{D}_i(\mathcal{W}_i)$

3.3 Bidirectional error correction codes for WOM codes based on modulo operation

3.3.1 Bidirectional error correction codes based on modulo operation

We introduce t bidirectional (l_u, l_d) limited-magnitude error correction codes, which can reduce errors more effectively. The proposed code is systematic, and can correct t bidirectional errors with upward and downward magnitude of l_u and l_d, respectively. We call the codes the *'bidirectional limited magnitude error correction code based on modulo operation(BLM-ECC)'*. Bidirectional limited-magnitude error correction codes [22] [35] extend the technique of the asymmetric error correction codes [8] to the bidirectional error correction codes. For example in $(2_u, 1_d)$ channel, if cell data is '2' in a 6-ary cell, an

error within $-1 \leq \epsilon \leq 2$ can be added to the data value of '2', so the cell data can be $1 \leq 2 + \epsilon \leq 4$. The threshold voltage (V_T) of the cell is not integer, but it is assumed that the cell data and the error values are integer. Since we do not know the exact threshold voltage or the interference voltage by memory reading operation, only the integer decision after hard-decision is possible. Therefore, the cell data of 2 in the $(2_u, 1_d)$ channel can be changed to $1, 2, 3$, or 4 with an error. $t - (l_u, l_d)$ BLM-ECC can correct the codeword with t errors of (l_u, l_d) magnitude. The code construction is as follows [35] [8]. Let Ω be a q'-ary code and $q' = l_u + l_d + 1$. The q-ary code $\mathcal{C}$ ($q > q'$) is defined as

$$\mathcal{C} = \{\mathbf{c} = (c_1, ..., c_n) \mid \mathbf{c} \bmod (l_u + l_d + 1) \in \Omega\} \tag{3.12}$$

$\mathcal{C}$ correct t bidirectional (l_u, l_d) limited-magnitude errors if Ω corrects t symmetric errors. The process of encoding and decoding of the proposed codes is described as follows.

Let $\mathbf{x} = \{x_1, \ldots, x_k\}$ be a q-ary message codeword, and q-ary multi-level cell memory is assumed to be used. We get the q'-ary remainder of the q-ary message $\mathbf{x}$ by modular q' operation ($q' = l_u + l_d + 1, q' < q$). The q'-ary remainder codes are called *base codes*. In order to encode by the *base codes*, conventional p-ary t symmetric error correction codes are used, which is called *base error correction codes*. With $\mathbf{x}$ mod q' codeword, the p-ary parity codes can be obtained using base error correction codes. A p-ary parity codeword needs to be converted to a q-ary codeword $\mathbf{p} = \{p_1, \ldots, p_r\}$ in order to be stored in a q-ary memory cell. The systematic encoded codeword is then $\mathbf{c} = [\mathbf{x}\ \mathbf{p}] = \{c_1, c_2, \ldots, c_n\}$

and $n = k + r$. 'Systematic' means that the original message part and the parity part are separated in the encoded codeword. The code rate is defined by k/n. For every k symbols of useful information, the code generates total n symbols of data, of which $n - k$ are parity codes. Since $q > q'$, the base code size is smaller than than the original message, and the parity code size can be also reduced. Therefore, the code rate of the BLM-ECC is larger than that of conventional error correction code, and this is the key advantage of the proposed code.

However, the above encoding method can cause the error count mismatch problem. The problem means one erratic cell usually causes two or more errors. There are two kinds of the problem, one is a *message correction problem* when $p < q'$, $l \geq 2$, and the other is a *parity code writing problem* when $p < q$, $l \geq 2$. Let us discuss the message correction problem first. One error in a q-ary cell can cause two or more errors in a p-ary message codeword if $p < q'$. For example, in a $(2_u, 1_d)$ memory channel, let us assume that $q = 8, q' = 4$, and $p = 2$ (binary) are the parameters for the base error correction codes. Note that a_b means b-ary a value for convenience. A message code 1_8 is 1_4 and 01_2. If $l = 2$ error occurs in the q-ary cell, 1_8 is changed to 3_8 which is 3_4 and 10_2 when the gray code $(000, 001, 011, 010, 110, \dots)$ is used. Two bits are different between 01_2 and 10_2, and one cell error in the message cannot be corrected by $t = 1$ binary base error correction codes in this example. If $p \geq q'$, we can avoid the problem although $p < q$.

Next, let us describe the parity code writing problem when $p < q$, $l \geq 2$. The proposed

codes are systematic codes, and a q-ary message is written in a q-ary cell. However, p-ary base error correction codes are used, and a p-ary parity codeword needs to be written in a q-ary cell. For example, $q = 16$ and $p = 4$, a parity code 13_4 is written in the cell as $1 \times 4^1 + 3 \times 4^0 = 7_{16}$. If $l = 2$ error is added to the cell, the cell value becomes '9_{16}' which is '21_4', but the original parity code is '13_4'. Therefore, one q-ary cell parity code error with $l = 2$ causes two p-ary parity code errors. This problem can be avoided when $p \geq q$. Consequently, p should be larger than q and q' ($p \geq q \geq q'$) to avoid the error count mismatch problem. Thus, non-binary p-ary error correction codes such as Reed-Solomon(RS) codes can be used. In order to achieve the maximum code rate, p, q, and q' are two to the power of k (k is integer), and $\log_2 p$ is a multiple of LCM($\log_2 q$,$\log_2 q'$).

Decoding of the proposed code is also based on the modular q' operation, and p-ary base error correcting codes. $\mathbf{c}' = (c'_1, \ldots, c'_n)$ is the received codeword, and $\boldsymbol{\epsilon} = (\epsilon_1, \ldots, \epsilon_n)$ is the error vector where each component is an integer. It is assumed that its magnitude is limited to $-l_d \leq \epsilon \leq l_u$ where l_d and l_u are positive integers. The received codeword is $\mathbf{c}' = \mathbf{c} + \boldsymbol{\epsilon} = [\mathbf{x}\ \mathbf{p}] + \boldsymbol{\epsilon} = [\mathbf{y}\ \mathbf{p}']$. $\mathbf{y} = (y_1, \ldots, y_k)$ is the received message part in the received codeword. At first, modular q' is performed on the received message $\mathbf{y}$, which is similar to the encoding process. We then have $\boldsymbol{\varphi} = \mathbf{y} \bmod q'$. The received q-ary parity part $\mathbf{p}'$ and $\boldsymbol{\varphi}$ need to be converted a p-ary codeword to be decoded by the base error correction codes. We can correct t symmetric errors for the codeword with $\boldsymbol{\varphi}$ and the parity code, and the corrected q'-ary message can be obtained if the t errors are within the (l_u, l_d)

bound. We can then estimate the error codeword $\boldsymbol{\epsilon}'$ by the difference between the corrected message code and the received message code. However, the estimated error may exceed the error bound due to the modular operation, although the error codeword is within the (l_u, l_d) bound, Fortunately, the estimated error can be recovered by a simple shift, adding or subtracting q'. The procedure is described as follows.

We define $\mathbf{x}, \mathbf{y}$, and $\boldsymbol{\epsilon}$ by a transmitted codeword, a received codeword, and a (l_u, l_d) error codeword, respectively. We have

$$\begin{aligned} \boldsymbol{\varphi} &= \mathbf{y} \bmod q' \\ &= (\mathbf{x} + \boldsymbol{\epsilon}) \bmod q' \\ &= (\boldsymbol{\eta} + \boldsymbol{\xi}) \bmod q' \end{aligned} \tag{3.13}$$

where $\boldsymbol{\eta} = \mathbf{x} \bmod q'$ and $\boldsymbol{\xi} = \boldsymbol{\epsilon} \bmod q'$. Since the modular operation with a negative integer may be confusing, we deal with the downward error and the upward error separately. We have

Case I. downward error (if $\epsilon_i = \epsilon_\downarrow$ $(-l_d \leq \epsilon_\downarrow \leq -1)$

$$\varphi_i = \begin{cases} \eta_i + \epsilon_\downarrow + q' & (0 < \eta_i + \epsilon_\downarrow + q' < q') \\ \eta_i + \epsilon_\downarrow & (q' \leq \eta_i + \epsilon_\downarrow + q' < 2q') \end{cases} \tag{3.14}$$

Case II. upward error.(if $\epsilon_i = \epsilon_\uparrow$ $(0 \le \epsilon_\uparrow \le l_u)$)

$$\varphi_i = \begin{cases} \eta_i + \epsilon_\uparrow & (0 < \eta_i + \epsilon_\uparrow < q') \\ \eta_i + \epsilon_\uparrow - q' & (q' \le \eta_i + \epsilon_\uparrow < 2q'). \end{cases} \tag{3.15}$$

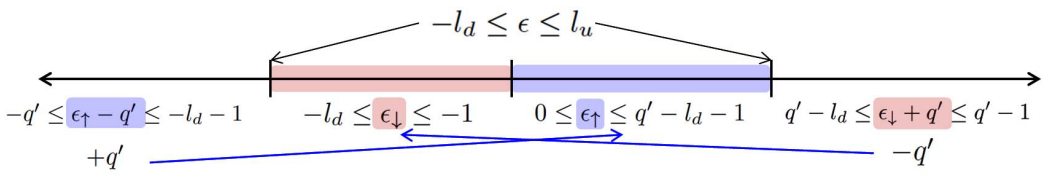

Figure 3.3 Adjustment of estimated error to be in the bound ($l_u = l_d$ is assumed).

It was assumed that $\boldsymbol{\eta} \in \Omega$ and Ω corrects t symmetric errors. Therefore, t symmetric errors of $\boldsymbol{\varphi}$ can be corrected. $\boldsymbol{\varphi}'$ is the corrected codeword of $\boldsymbol{\varphi}$, so $\boldsymbol{\varphi}' = \boldsymbol{\eta}$. The estimated error ϵ_i' is $\varphi_i - \varphi_i'$. As for case I with downward errors, we have $\epsilon_i' = \epsilon_\downarrow + q'$ or $\epsilon_\downarrow$. As for case II with upward errors, we have $\epsilon_i' = \epsilon_\uparrow$ or $\epsilon_\uparrow - q$. Therefore, four types of error show up in the estimated error codeword, and $-q' \le \epsilon_i' < q' - 1$. The original error is in the range of $-l_d \le \epsilon \le l_u$, but the estimated error may exceed the bound (range). However, the four types of error have their own distinct ranges as $-q' \le \epsilon_\uparrow - q' \le -l_d - 1$, $-l_d \le \epsilon_\downarrow \le -1$, $0 \le \epsilon_\uparrow \le q' - l_d - 1$, and $q' - l_d \le \epsilon_\downarrow + q' \le q' - 1$ where we used only q' and l_d ($l_u = q' - l_d - 1$). Thus, we can distinguish them with the range, and recover the estimated error by adding or subtracting q'. The adjustment of estimated error is illustrated in Fig. 3.3.

The encoding and the decoding algorithms of bidirectional limited magnitude error correction codes based on modulo operation (BLM-ECC) are described as follows.

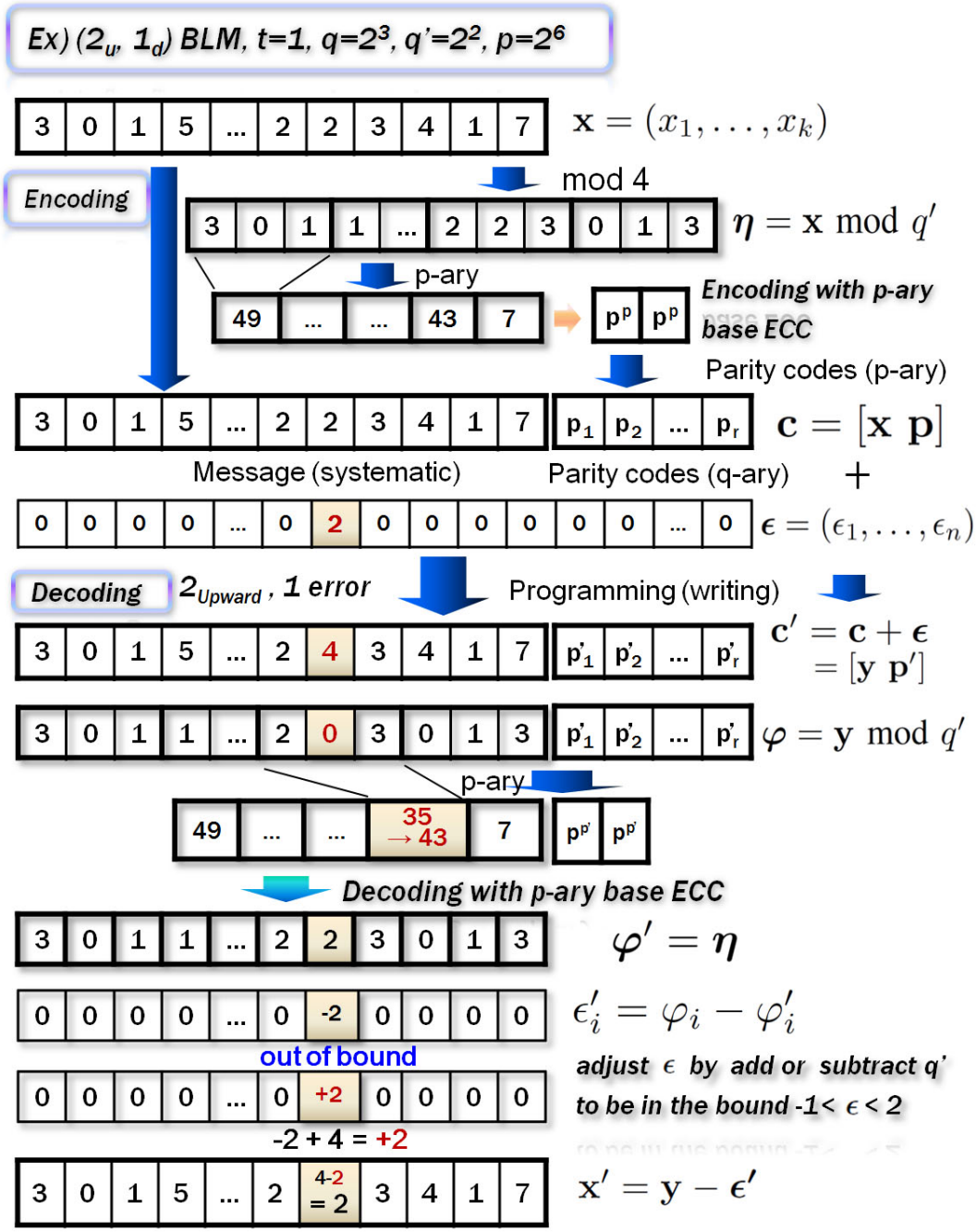

Figure 3.4 An example of BLM-ECC encoding and decoding.

Bidirectional Limited Magnitude Error Correction Codes Based on Modulo Operation Algorithm

Encoding

(Initialization) q-ary message codeword $\mathbf{x}$

$q' = l_u + l_d + 1$, *where* l_u *and* l_d *are upward and downward error magnitude bounds, respectively.*

1) *Get the remainder of message* $\mathbf{x}$ *by* $\mod q'$.

$\boldsymbol{\eta} = \mathbf{x}$ *mod* q'.

2) *Generate the* p*-ary parity codes for* $\boldsymbol{\eta}$ *using* p*-ary base error correction codes and convert the codes to* q*-ary codes,* $\mathbf{p}$ *(*$p \geq q \geq q'$*).*

3) *A systematic encoded codeword is* $\mathbf{c} = [\mathbf{x}\ \mathbf{p}]$. *Write the codeword to the* q*-ary memory cell.*

Decoding

(Initialization) Received codeword $\mathbf{c}' = [\mathbf{x}\ \mathbf{p}] + \boldsymbol{\epsilon} = [\mathbf{y}\ \mathbf{p}']$, *where* $\boldsymbol{\epsilon} = (\epsilon_1, \ldots, \epsilon_n)$ *is the error vector with each integer component within* (l_u, l_d), $\mathbf{y}$ *is the received message code, and* $\mathbf{p}'$ *is the received parity code.*

1) *Get the remainder of received message* $\mathrm{mod}\ q'$.

$\boldsymbol{\varphi} = \mathbf{y}$ *mod* q'.

2) *Convert* $\boldsymbol{\varphi}$ *and* $\mathbf{p}'$ *to* p*-ary codes for base ECC decoding. Corrected* q'*-ary message is* $\boldsymbol{\varphi}'$.

3) *Estimate the error by* $\boldsymbol{\epsilon'} = \boldsymbol{\varphi} - \boldsymbol{\varphi'}$, $\boldsymbol{\epsilon'} = (\epsilon'_1, \ldots, \epsilon'_k)$ *is estimated error of the message.*

4) *If the estimated error exceeds the bound* ($\epsilon'_i > l_u$ *or* $\epsilon'_i < -l_d$)*, let* $\epsilon'_i = \epsilon'_i + q'$ *or* $\epsilon'_i = \epsilon'_i - q'$ *to be in the range of* $-l_d \leq \epsilon'_i \leq l_u$.

5) *The corrected message* $\mathbf{x}'$ *is obtained by* $\mathbf{x}' = \mathbf{y} - \boldsymbol{\epsilon'}$.

The example of BLM-ECC encoding and decoding process is illustrated in Fig. 3.4. $(2_u, 1_d)$ bidirectional limited magnitude codes are assumed and $t = 1$, $q = 8$, $q' = l_u + l_d + 1 = 4$, and $p = 64$.

The number of codewords of $\mathcal{C}$ in (3.12) is bounded by the following inequalities [35] [8].

$$\left\lfloor \frac{q}{l_u + l_d + 1} \right\rfloor^n \cdot |\Omega| \leq |\mathcal{C}| \leq \left\lceil \frac{q}{l_u + l_d + 1} \right\rceil^n \cdot |\Omega| \tag{3.16}$$

$\boldsymbol{\eta} = (\eta_1, \ldots, \eta_n)$ is considered to be a codeword of Ω in (3.12). The codewords of $\mathcal{C}$ can be obtained by replacing each η by the element of the set $\Lambda = \{\lambda | \lambda \bmod q' = \eta_i, \lambda \in \{0, 1, \ldots, q-1\}\}$. The size of Λ is $\lceil q/q' \rceil^n$ if $\eta_i < (q \bmod q')$, and $\lfloor q/q' \rfloor^n$ otherwise. In (3.16), the upper bound is $|\mathcal{C}| \leq \left\lceil \frac{q}{l_u + l_d + 1} \right\rceil^n \cdot |\Omega|$. Ω is q'-ary, and can correct t symmetric

errors, and we have

$$|\Omega| \cdot \sum_{k=0}^{t} \binom{n}{k} (q'-1)^k \leq q'^n. \tag{3.17}$$

Substituting the latter into the former, the following is obtained. If $\mathcal{C}$ is a (l_d, l_u) limited-magnitude t-error correcting codes of length n over an alphabet of size q, we have

$$|\mathcal{C}| \leq \frac{q^n}{\sum_{k=0}^{t} \binom{n}{k} (l_d + l_u)^k}. \tag{3.18}$$

The code rate of the BLM-ECC depends on the p-ary base error correction codes. If (n, k) error correction codes are used as the base ECC, BLM-ECC generates $((n-k)\log_q q' + k, k)$ codes. If $(2^m - 1, 2^m - 1 - 2t)$ Reed-Solomon (RS) codes are used as the p-ary base ECC, the code rate is given by

$$\begin{aligned} R_{BLM} &\leq \frac{(2^m - 1 - 2t)\log_2 q}{(2^m - 1 - 2t)\log_2 q + 2t\log_2 q'} \\ &= \frac{(2^m - 1 - 2t)}{(2^m - 1) + 2t(\log_q q' - 1)} \end{aligned} \tag{3.19}$$

The RS codes encodes the p-ary message with $p = 2^m$. The equality can be achieved when $\log_2 p$, $\log_2 q$, and $\log_2 q'$ are positive integers with $p \geq q \geq q'$, and $\log_2 p$ is a multiple of LCM($\log_2 q$,$\log_2 q'$). The code rate of $(2^m - 1, 2^m - 1 - 2t)$ Reed-Solomon (RS) codes is

$$R_{RS} = \frac{(2^m - 1 - 2t)}{(2^m - 1)} \tag{3.20}$$

If $q > q'$ and we have equality in (3.20), the parity size of BLM-ECC is less than that of the RS codes, and $2t(\log_q q' - 1)$ in the R_{BLM} expression can be negative, giving

$R_{BLM} > R_{RS}$. Let us discuss asymmetric error correction codes. $1A1M$ denotes systematic asymmetric error correction codes with $t = 1$ and $l = 1$ as defined in [8]. The code has the same error correction capability as the $(1_u, 0_d)$ BLM-ECC if the base error correction codes are identical. When we use the Reed-Solomon codes as the p-ary base error correction codes in the asymmetric error correction codes, the code rate is given by $R_{ASY} \leq \frac{2^m-1-2t}{(2^m-1-2t)+2t\log_q 2}$. The parity sizes of RS code, BLM-ECC, and asymmetric-ECC are $P_{RS} = 2t$, $P_{BLM} = 2t\log_q(l_u+l_d+1)$, and $P_{ASY} = 2t\log_q 2$, respectively. For example, if $(l_u, l_d) = (2, 1)$ for BLM-ECC, and $q = 8$, the parity size ratios are $P_{BLM}/P_{RS} = 2/3$ and $P_{ASY}/P_{RS} = 1/3$, which results in $R_{RS} < R_{BLM} < R_{ASY}$. If most errors are in the (l_u, l_d) bound, the BLM-ECC (based on RS) can have better error rate performance than the conventional RS codes with equal code rate when $q > q'$, which is verified in simulations.

3.3.2 Performance simulation of bidirectional error correction codes based on modulo operation

We simulated the bit error rate performance of the proposed bidirectional limited magnitude error correction codes (BLM-ECC). In simulations, the multi-level cell model of flash memories with interference is used. One of the dominant factors of the interference is the cell to cell interference as described in Section 2. The cell-to-cell interference model depends on the program order, the page architecture, and the conventional LSB/MSB techniques, and these factors are considered in simulations. To simulate flash memories with interference, we need not only an interference model, but also a threshold voltage distribution model. To

write data onto flash memories inherently involves errors due to the noise in the physical process. We can approximate the cell threshold voltage distribution as Gaussian [19]. It should be noted that the empirical distribution obtained from measurements is not exactly a Gaussian distribution, but rather a kind of truncated Gaussian distribution. An 8-level flash memory model (3 bits in a cell) is used with an equal noise distribution model. The equal noise distribution model assumes that each level has equal threshold voltage distribution which is clipping Gaussian. Clipiing Gaussian means that tail parts of Gaussian distribution are removed, and it is more realistic.

In our simulations, the center threshold voltages of the 8-levels range from $-0.57V$ to $3.42V$, and there is 0.57V gap between adjacent levels. The hard-decision reference voltages (V_{read}) for reading NAND flash memories are determined based on the V_T distribution after the interference. It means the hard-decision reference voltages for reading are already near-optimal in practical systems. We use both original hard-decision reference voltages and the adjusted near-optimal reference voltages in the Fig. 3.5 and 3.6 simulations in order to verify the performance of asymmetric error correction codes. $(2_u, 1_d)$ BLM-ECC and $1A1M$ asymmetric error correction code simulations are performed. $1A1M$ means asymmetric ECC with $t = 1$ and $l = 1$ as defined in [8]. We use the Reed-Solomon codes as the p-ary base error correction codes in the asymmetric codes and the $(2_u, 1_d)$ BLM-ECC. In the $(2_u, 1_d)$ BLM-ECC, we use $q' = 2 + 1 + 1 = 2^2$ and $q = 8 = 2^3$. The maximum code rate of the proposed algorithm can be achieved when $\log_2 p = m = LCM(2, 3) = 6$,

and $(2^6 - 1, 2^6 - 1 - 2t) = (63, 63 - 2t)$. To compare the performance, the conventional $(63, 63 - 2t)$ RS codes are also simulated.

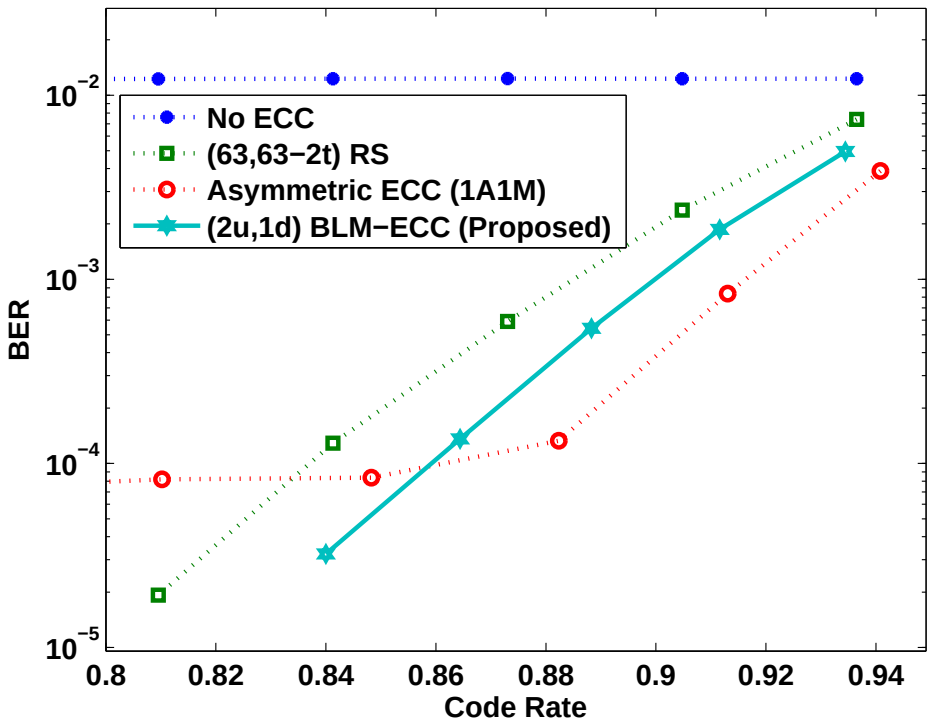

Figure 3.5 BER performance with original V_{read} for an asymmetric channel.

Fig. 3.5 shows the BER performance of the BLM-ECC and the asymmetric error correction codes with the original hard-decision reference voltage(V_{read}). The code rate of the horizontal axis is determined as t changes. The 'No ECC' plot shows the bit error rate without any error correction codes, and the curve is flat. As the code rate decreases for other ECC's, the parity size of the ECC and t increase, and the bit error rate decreases. At the high code rate range, asymmetric error correction codes show the best performance. Be-

cause most errors (over 99%) are upward when original V_{read} is used as the hard-decision reference voltage. However at the low code rate range, asymmetric ECC cannot correct the out of bound errors, especially downward errors. Therefore, there is an error floor for code rate lower than 0.88, and the BLM-ECC has better performance than asymmetric error correction codes for this range. The $(2_u, 1_d)$ BLM-ECC has better performance than the original RS codes for all code rate because the BLM-ECC parity size is reduced. In Fig.

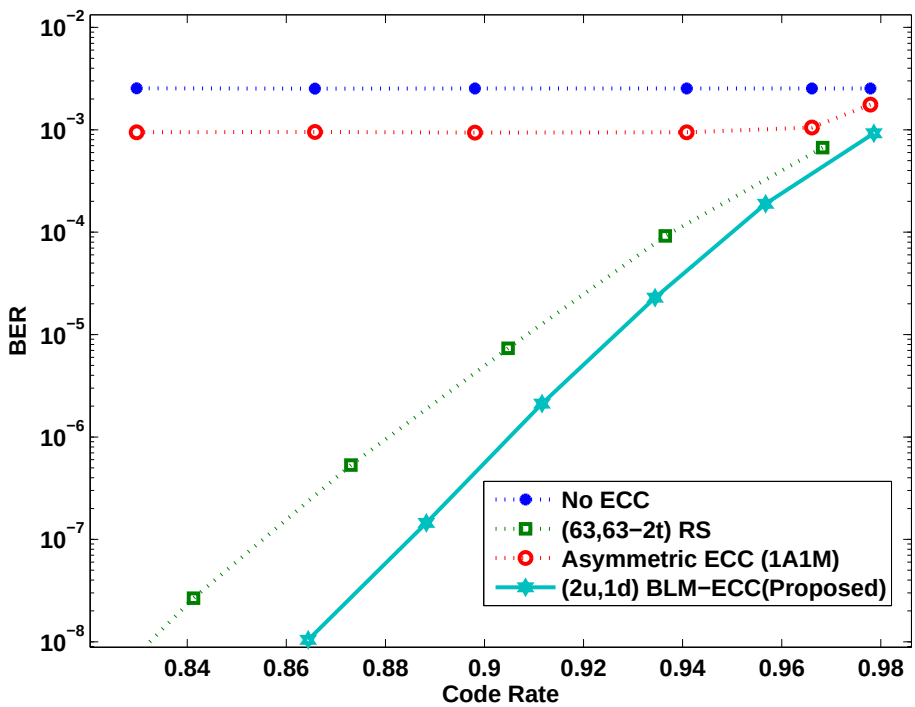

Figure 3.6 BER performance with adjusted V_{read} for a bidirectional channel.

3.6, the hard-decision reference voltage (V_{read}) is already adjusted based on the the average interference quantity with writing random data. The average cell-to-cell interference

assuming random data is estimated to be about $0.07V$ by simulations, and it is added to the original hard-decision reference voltage(V_{read}). The adjusted hard-decision reference voltage compensates the upward V_T shift by the cell-to-cell interference, which makes the errors more symmetric rather than asymmetric. Therefore, the performance of asymmetric error correction codes gets worse. The $(2_u, 1_d)$ BLM-ECC algorithm shows the best BER performance at all code rate, and it is shown that the proposed algorithm is efficient for the MLC flash memory model.

3.3.3 Error correction coding schemes for WOM codes based on modulo operation

The code construction of bidirectional error correction code based on modulo operation for non-binary WOM codes is discussed.

To construct error correction codes for non-binary WOM codes, we get the q'-ary remainder of the i_{th} q-ary WOM codeword $\mathcal{W}_i$ by modular q' operation $(q' = l_u + l_d + 1, q' \leq q)$ first. The q'-ary remainder codes are called *base codes*. In order to encode by the *base codes*, conventional g-ary u symmetric error correction codes are used, which is called *base error correction codes*. With $\mathcal{W}_i$ mod q' codeword, the g-ary parity codes can be obtained using base error correction codes. g should be larger than q and q' $(g \geq q \geq q')$ to avoid the error count mismatch problem [35], which means an erroneous cell may cause two or more errors. Non-binary g-ary error correction codes such as Reed-Solomon(RS) codes can be used [22] [35]. A g-ary parity codeword needs to be converted to a ρ-ary codeword γ_i in

order to be stored in a q-ary memory cell with the t-write WOM code property. Therefore $\rho t \leq q$ and $\rho \leq \lfloor \frac{q}{t} \rfloor$.

Decoding of the proposed code is also based on the modular q' operation, and g-ary base error correcting codes. At first, modular q' is performed on the received message, which is similar to the encoding process. The received q-ary parity part need to be converted a g-ary codeword to be decoded by the base error correction codes. u symmetric errors can be corrected with the parity code, and the corrected q'-ary message can be obtained if the u errors are within the (l_u, l_d) bound. We can then estimate the error codeword by the difference between the corrected message codeword and the received message codeword. The estimated error may exceed the error bound due to the modular operation, although the error codeword is within the (l_u, l_d) bound, In this case, the estimated error can be recovered by a simple shift, adding or subtracting q'.

The encoding and the decoding algorithms of the modular operation-based error correction codes for WOM codes are described as follows.

Bidirectional limited magnitude non-binary WOM error correction codes based on modulo operation(BLM-WECC)

Encoding

Input : the i_{th} message $c_i, i = 1, \cdots, t$.

Output : $[\mathcal{W}_i\ \varphi_i], i = 1, \cdots, t.$

$\mathcal{W}_i$: *the* i_{th} *encoded WOM codeword of* c_i *message*

φ_i : *the* i_{th} *parity codeword for WOM*

(Initialization) $\lfloor \frac{q}{t} \rfloor = \rho, i = 1.$

$q' = l_u + l_d + 1$

1) *Generate non-binary WOM codes using encoding maps.*

If $i = 1$, $\mathcal{W}_1 = \mathbb{E}_1(c_1)$.

or $i \geq 2$, $\mathcal{W}_i = \mathbb{E}_i(c_i, \mathcal{W}_{i-1})$

2) *Get the remainder of WOM codes* $\mathcal{W}_i$ *by* $\mathrm{mod}\ q'$.

$\boldsymbol{\eta} = \mathcal{W}_i$ *mod* q'.

3) *Generate a* g*-ary parity codeword for* $\boldsymbol{\eta}$ *using* g*-ary base error correction codes, and convert the codeword to a* ρ*-ary codeword,* γ_i $(g \geq q \geq q')$.

4) *Get the parity codeword for WOM* $\varphi_i = \gamma_i + (i-1)\rho \cdot \mathbf{1}$

5) *Systematic encoded codewords* $[\mathcal{W}_i\ \varphi_i]$ *are written to the* q*-ary memory cell.*

6) *If* $i = t$*, go to step 7, else* $i \leftarrow i + 1$ *and go to step 1.*

7) *Erase the cells,* $i \leftarrow 1$ *and go to step 1.*

Decoding

(Initialization) Received codeword $[\mathcal{W}_i' \ \varphi_i'] = [\mathcal{W}_i \ \varphi_i] + \boldsymbol{\epsilon}$*, where* $\boldsymbol{\epsilon} = (\epsilon_1, \ldots, \epsilon_n)$ *is the error vector with each integer component within* (l_u, l_d)*,*

1) *Get the remainder of received message* $\bmod q'$*.*

$\boldsymbol{\eta'} = \mathcal{W}_i'$ *mod* q'*.*

2) *Get the* i_{th} *parity codeword for WOM* $\gamma_i' = \varphi_i' - (i-1)\rho \cdot \mathbf{1}$

3) *Convert* $\mathcal{W}'$ *and* γ_i' *to a* g*-ary codeword for base ECC decoding. The corrected* q'*-ary remainder is* $\tilde{\boldsymbol{\eta}}$*.*

4) *Estimate the error by* $\boldsymbol{\epsilon'} = (\epsilon_1', \ldots, \epsilon_k') = \boldsymbol{\eta'} - \tilde{\boldsymbol{\eta}}$*.*

5) *If the estimated error exceeds the bound (*$\epsilon_i' > l_u$ *or* $\epsilon_i' < -l_d$*), let* $\epsilon_i' = \epsilon_i' + q'$ *or* $\epsilon_i' = \epsilon_i' - q'$ *to be in the range of* $-l_d \leq \epsilon_i' \leq l_u$*.*

6) *The corrected WOM encoded message* $\mathcal{W}_i$ *is obtained by* $\mathcal{W}_i = \mathcal{W'}_i - \boldsymbol{\epsilon'}$*.*

7) *The original message is decoded by* $c_i = \mathbb{D}_i(\mathcal{W}_i)$

3.4 Performance of error correction coding schemes for WOM code

The code rate of the two proposed codes for WOM codes will be discussed in this section. The code rate is defined by k/n. For every k symbols of useful information, the code generates total n symbols of data, of which $n-k$ are parity codes. We call bidirectional single error correction codes based on distinct sum sets for non-binary WOM codes in section

3.2 BD-WECC code for short. The column size of parity check matrix H is $m\frac{q^r-1}{q-1}$ and it means the code length n [10]. Let us consider the case that the each alphabet size of WOM parity code symbols are equal for each write, which is shown in Section 3.2. Then the parity code length r changes $r\lceil\frac{q}{\lfloor q/t\rfloor}\rceil$. Therefore, the code rate $R_{BD-WECC}$ is given as follows.

$$\begin{aligned} R_{BD-WECC} &= \frac{m\frac{q^r-1}{q-1}-r}{m\frac{q^r-1}{q-1}+r(\lceil\frac{q}{\lfloor q/t\rfloor}\rceil-1)} \\ &= \frac{m(q^r-1)-r(q-1)}{m(q^r-1)+r(q-1)(\lceil\frac{q}{\lfloor q/t\rfloor}\rceil-1)} \end{aligned} \tag{3.21}$$

Note that $m\frac{q^r-1}{q-1}$ corresponds to the message size in q-ary symbols. The bidirectional limited magitude error correction codes based on modulo operation for non-binary WOM codes in section 3.3 is called BLM-WECC. The code rate of the BLM-WECC depends on the g-ary base error correction codes. Let $(2^i-1, 2^i-1-2u)$ Reed-Solomon (RS) codes be used as the g-ary base error correction codes. The RS codes encodes the g-ary message with $g=2^i$ and each parity block has q-ary $\frac{(2^i-1-2u)\log_2 g}{\log_2 q'}$ cells message symbols. The code rate $R_{BLM-WECC}$ is give by

$$\begin{aligned} R_{BLM-WECC} &\leq \frac{(2^i-1-2u)\frac{\log_2 g}{\log_2 q'}}{(2^i-1-2u)\frac{\log_2 g}{\log_2 q'}+2u\frac{\log_2 q'}{\log_2\lfloor\frac{q}{t}\rfloor}\frac{\log_2 g}{\log_2 q'}} \\ &= \frac{(2^i-1-2u)}{2^i-1+2u(\log_{\lfloor\frac{q}{t}\rfloor}q'-1)} \end{aligned} \tag{3.22}$$

The equality can be achieved when $\log_2 g$, $\log_2 q$, and $\log_2 q'$ are positive integers with $g \geq q \geq q'$, and $\log_2 g$ is a multiple of LCM($\log_2 q$,$\log_2 q'$) [35].

Fig. 3.7 show comparison of the code rate performance of proposed schemes without WOM code schemes. BD-ECC and BLM-ECC stands for BD-WECC and BLM-WECC

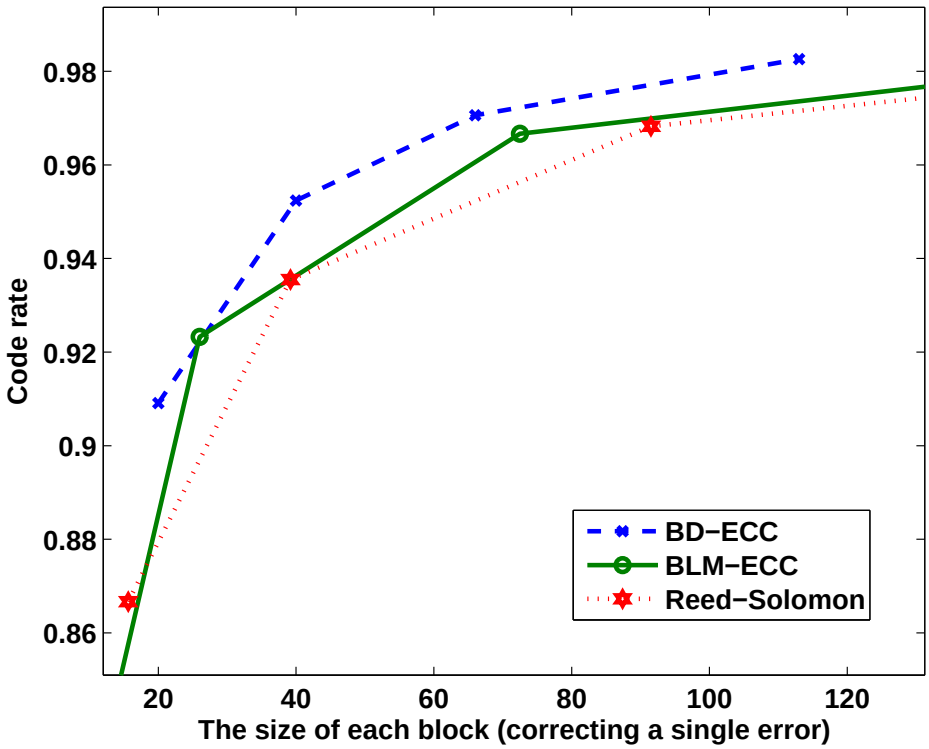

Figure 3.7 The code rate of proposed codes without WOM code schemes

with $t = 1$, which means that the schemes write message only once and do not have WOM codes property. The parameters are $q = [10, 13, 16, 22, 34, 40]$, $(l_u, l_d) = (2, 1)$. $m = [2, 3, 4, 5, 6, 7]$, $r = 2$ for BD-ECC schemes, $i = [3, 4, 5, 6, 7, 8]$ for BLM-ECC schemes and Reed-Solomon codes. Two proposed codes deal with limited magnitude errors and show better performance than conventional Reed-Solomon codes. In this case, the code rate of BD-ECC is larger than BLM-ECC.

Fig. 3.8 shows the code rate performance of the two proposed schemes with respect to

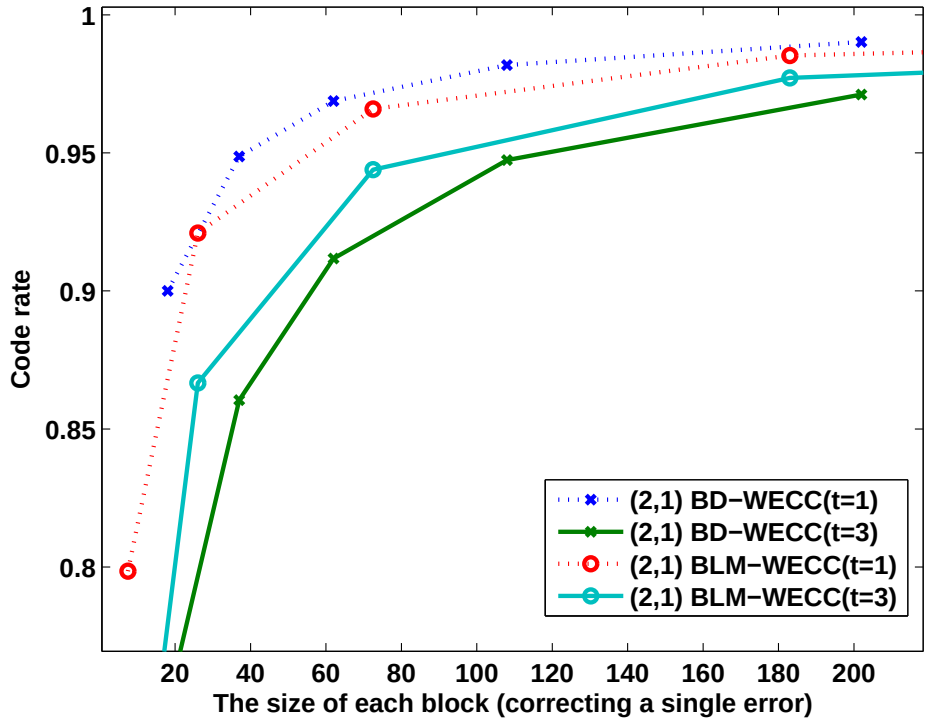

Figure 3.8 The code rate of two proposed codes for WOM codes with varying a block size

the block size. The parameters are $q = [10, 13, 16, 22, 34, 40]$ and $(l_u, l_d) = (2, 1)$. We use $m = [2, 3, 4, 5, 6, 7]$, $r = 2$ for the BD-WECC scheme, and $i = [3, 4, 5, 6, 7, 8]$ for the BLM-WECC scheme. Although the code rate of BD-WECC is the larger than that of BLM-WECC with $t = 1$, BLM-WECC shows better code rate performance than BD-WECC when $t = 3$. This is because of the $\lfloor \frac{q}{t} \rfloor$ term in the code rate equations of (3.21) and (3.22). The code rate of the proposed schemes with respect to t (with fixed block size) is shown in Fig. 3.9. The parameters are $q = 12$, and $(l_u, l_d) = (2, 1)$ for both schemes. We use $m = 3, r = 2$ and $k_{BD} = 37$ (block size) for the BD-WECC scheme, and $i = 4$ and $k_{BLM} = 26$

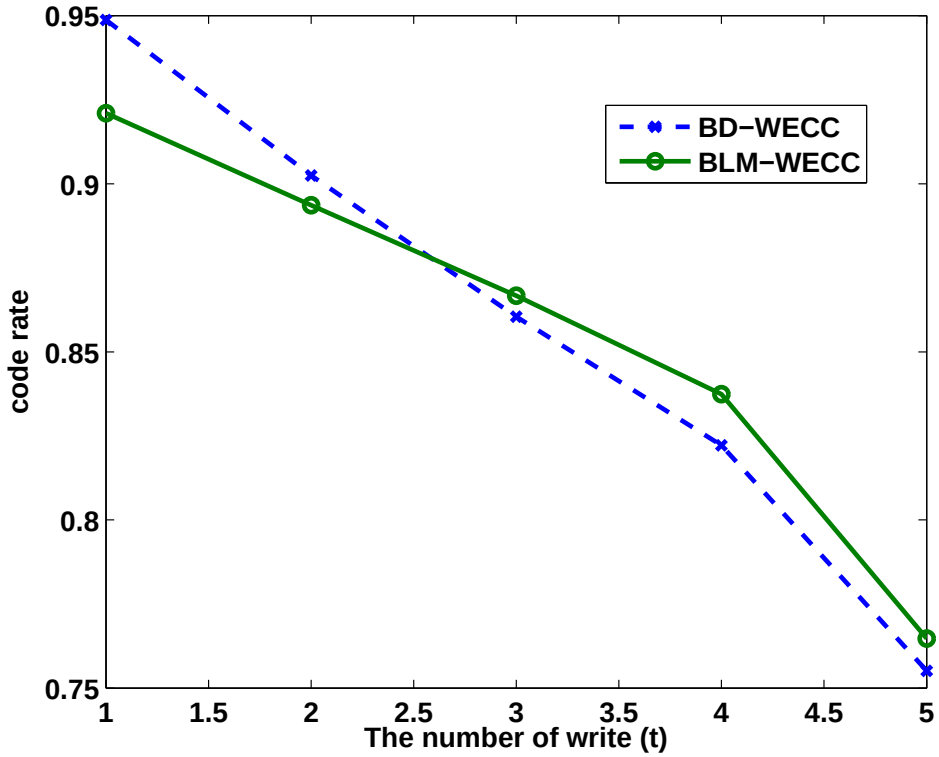

Figure 3.9 The code rate of two proposed codes with varying t

for the BLM-WECC scheme. As the number of write of WOM codes t increases, BLM-WECC gets more efficient than BD-WECC. Furthermore, the error correction capability of BLM-WECC can be adjusted according to the base error correction codes, but BD-WECC can correct only one single error. However, the encoding and decoding complexity of BD-WECC is much smaller than that of BLM-WECC, because BLM-WECC uses the base error correction codes which have high decoding complexity such as RS codes.

The proposed codes have advantages compared to other error correction codes for

WOM codes. The error correction codes in [37] and [36] are appropriate for binary WOM codes, but the proposed methods can be applied to general non-binary WOM codes. The codes proposed in [38] are for limited-magnitude-error non-binary WOM codes. However, the codes are non-systematic error correction codes, and conventional non-binary WOM code construction cannot be used. A large alphabet size is also required for the codes to increase error correction capability. The number of cell levels is limited in multi-l evel memory cells, so this can lead to inefficiency.

Let us compare the performance of proposed error correcting schemes combined with WOM codes in [42] with the performance of algorithms of [40]. Assume that a flash coding scheme guarantees t successive writes and encodes an arbitrary message from a set of M_i messages in the i-th write. The memory rate is defined as

$$\mathcal{R} = \sum_{i=1}^{t} \log_2 \frac{M_i}{n}. \tag{3.23}$$

The memory rate $\mathcal{R}$ is upper bounded by the capacity [33] [40]

$$\mathcal{C}_W = \log_2 \binom{t+q-1}{q-1}. \tag{3.24}$$

$\mathcal{C}_W$ is the maximum total number of information bits stored in one cell during the t successive block-writes. The codes should be allowed to store different number of messages in each write to achieve the capacity [33] [40]. We assume that $q = 8$, single error correcting schemes, and (7,5,3) Reed-Solomon code is used.

Each $\mathcal{R}$ means the rate of each write, and total $\mathcal{R}$ means the rate of sum of total

Table 3.5 The performance comparison of error correcting schemes for WOM codes

	t writes	ϱ	(l_u, l_d)	each $\mathcal{R}$	total $\mathcal{R}$	$\mathcal{C}_W$	$\mathcal{R}/\mathcal{C}_W$
Huang11 [40]	7	0	-	0.43	3	11.74	25.5%
Huang11 [40]	11	1	-	0.43	4.71	14.96	31.4%
BLM-WECC + Cassuto12 [42]	4	-	all	0.68	2.72	8.37	32.5%
BLM-WECC + Cassuto12 [42]	4	-	(2,1)	0.94	3.75	8.37	44.8%
BLM-WECC + Cassuto12 [42]	4	-	(1,0)	1.07	4.2	8.37	50.1%

writes. Table 3.5 show the performance comparison of error correcting schemes for non-binary WOM codes. 'each $\mathcal{R}$' is $\log_2 \frac{M_i}{n}$ which means the rate of each write. 'total $\mathcal{R}$'is $\sum_{i=1}^{t} \log_2 \frac{M_i}{n}$ which means the total sum rate. Proposed algorithm shows smaller gap between $\mathcal{R}$ and $\mathcal{C}_W$ than that of [40].

The proposed codes are systematic and practical because they can be applied to the conventional non-binary WOM codes. The code construction, the encoding process, and the decoding process are also efficient. It is difficult to compare the proposed methods with other error correction schemes for non-binary WOM codes because of different characteristics and structure. Nevertheless, the proposed codes appear to have significant potential advantages.

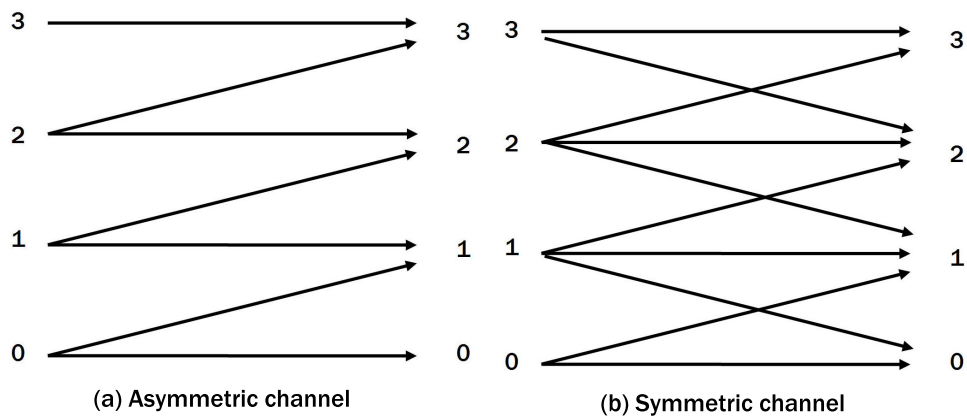

Figure 3.10 Asymmetric and symmetric limited-magnitude channel.

3.5 Error locating parity check codes for errors with limited magnitude

In this subsection, we propose new error correcting codes which are effective for the interference [41]. We assume that there is at most one error in each code block, and the codes are designed to correct one single error in a code block. Because the errors have limited magnitude, the remainder values which are generated with modulo N operation still contains the error information. By taking advantages of the characteristics, we introduce effective asymmetric and symmetric limited-magnitude parity error correction codes for the MLC flash memory error with lower redundancy. At first, asymmetric limited-magnitude errors are considered. Fig. 3.10 (a) shows an example of 4-level (X2) asymmetric limited-magnitude channel, where the error magnitude is limited 1. Let us consider a case where the error is asymmetric. This is a reasonable assumption because the cell-to-cell interference can only

increase the threshold voltage.

Suppose that Ω and Ξ is the q-ary input and output codeword, respectively. The codewords can be represented by $\Omega = (\omega_1, \omega_2, \cdots, \omega_n)$ and $\Xi = (\xi_1, \xi_2, \cdots, \xi_n)$. The codewords can be modified by a modulus operation, i.e., $\bar{\omega}_i = \omega_i \bmod N$ and $\bar{\xi}_i = \xi_i \bmod N$. Note that N is determined by the number of error types. If the maximum error magnitude is ℓ and $N > \ell$, we have $\xi - \omega = (\xi - \omega) \bmod N$. We also have

$$\epsilon_i = \xi_i - \omega_i = (\xi_i - \omega_i) \bmod N \tag{3.25}$$

$$= (\xi_i \bmod N - \omega_i \bmod N) \bmod N \tag{3.26}$$

$$= (\bar{\xi}_i - \bar{\omega}_i) \bmod N. \tag{3.27}$$

In other words, the error ϵ can be obtained from $\bar{\omega}_i$ and $\bar{\xi}_i$, so the error information does not change by the modulo technique. The q-ary data can be converted into N-ary data by the modulo N technique, and the redundancy for error correction can be reduced [14]. The limited magnitude error parity check (LMEPC) algorithm for asymmetric error using modulo 2 is described as follows.

LMEPC Mod 2 Algorithm for Asymmetric Errors

Encoding

(Initialization) The q-ary $m \times n$ data:

$D = \{d_{1,1}, d_{1,2}, \ldots, d_{1,n}, d_{2,1}, \ldots, d_{m,n}\}$.

1) *Get the remainder of message by modulo 2.*

$\bar{D} = \{\bar{d}_{1,1}, \bar{d}_{1,2}, \ldots, \bar{d}_{m,n}\}$ *where* $\bar{d}_{i,j} = d_{i,j} \bmod 2$.

2) *Generate the parity codes for each row and column.* $P_r = \{p_{r_k}\}$ *and* $P_c = \{p_{c_k}\}$ *where*

$p_{r_k} = (\sum_{i=1}^{n} \bar{d}_{k,i}) \bmod 2$ *and* $p_{c_k} = (\sum_{i=1}^{m} \bar{d}_{i,k}) \bmod 2$.

3) *Store the message data and parity check codes.*

Parity codes are converted to q-ary data,

$P_r \rightarrow \hat{P}_r, P_c \rightarrow \hat{P}_c$ *where* $\hat{P}_r$ *and* $\hat{P}_c$ *are q-ary version of* P_r *and* P_c.

D, $\hat{P}_r$, *and* $\hat{P}_c$ *are stored separately.*

Decoding

(Initialization) The q-ary received data and parity information with interference and hard decision:

$D' = \{d'_{1,1}, d'_{1,2}, \ldots, d'_{m,n-1}, d'_{m,n}\}$, $\hat{P}'_r$, $\hat{P}'_c$.

The q-ary parity cell data is converted back to binary data: $\hat{P}'_r \rightarrow P'_r = \{p'_{r_k}\}$, $\hat{P}'_c \rightarrow P'_c = \{p'_{c_k}\}$.

1) *Get the remainder of received message by* $\bmod 2$.

$\bar{D}' = \{\bar{d}'_{1,1}, \bar{d}'_{1,2}, \ldots, \bar{d}'_{m,n}\}$ *where* $\bar{d}'_{i,j} = d'_{i,j} \bmod 2$.

2) *Generate the parity codes of the received data for each row and column.*

$\tilde{P}_r = \{\tilde{p}_{r_k}\}$ *and* $\tilde{P}_c = \{\tilde{p}_{c_k}\}$ *where* $\tilde{p}_{r_k} = (\sum_{i=1}^{n} \bar{d'}_{k,i}) \bmod 2$ *and* $\tilde{p}_{c_k} = (\sum_{i=1}^{m} \bar{d'}_{i,k}) \bmod 2$.

3) *Check the parity and obtain the location of error.*

If $\tilde{p}_{r_i} \neq p'_{r_i}$ *&* $\tilde{p}_{c_j} \neq p'_{c_j}$*, the error location is* $d'_{i,j}$.

4) *Correct the asymmetric upward error,* $d'_{i,j} = d'_{i,j} - 1$.

The original message set is denoted by D, which consists of q-ary $m \times n$ cell messages $(d_{i,j})$. The remainder set of $d_{i,j}$ is denoted by $\bar{D}$. The row parity p_{r_k} and the column parity p_{c_k} are generated from D. The received message data, the row parity check codes, and the column parity check codes are denoted by D', $\hat{P'}_r$, and $\hat{P'}_c$, respectively. It is assumed that at most one asymmetric error of magnitude one occurs. When an error occurs in the data set D, the error location can be identified by the given parity check algorithm. Suppose an error occurs in one of the cells storing the check bits. Since the decoding algorithm will make a correction if it detects an error in both of the row and the column parity, the data bits will not be affected by the decoding algorithm. In this case, we need to store the row check bits and the columns check bits in different cells.

Errors usually occur in the programming (writing) operation, and the stored parity check codes can be used after the reading operation to locate the errors. The following is an example of 8-level (X3) limited-magnitude parity check codes. This parity check codes correct

asymmetric errors using the $\mathrm{mod}\ 2$ technique. The row parity bits and the column parity bits are efficiently stored after being converted to q-ary data. However, to prevent an error on parity cells from causing a message error, the row parity bits and the column parity bits need to be stored in different cells. The code rate can be adjusted continuously by adjusting the data block size. If there are m rows and n columns in a data block, there are mn message cells for 2^k-level MLC. The code rate R_c is given by

$$R_c = \frac{mn}{mn + \lceil \frac{m}{k} \rceil + \lceil \frac{n}{k} \rceil}. \tag{3.28}$$

If m equals n, then the code rate is maximal. One parity block can correct one error, so the code rate can be determined considering error correction capability. We assume that there is only single error in a code block, so the proposed algorithm is developed with the assumption. However, the algorithm can be easily extended to handle more than one error by using more parity bits.

Although the cell-to-cell interference which produces upward errors is a dominant factor for the errors in MLC flash memories, there are other types of noise such as symmetric (random-telegraph noise) and downward (retention noise and stress induced leakage current) interference, which may be less significant [11]. In practice, the read voltage for NAND flash memories is made based on the V_T distribution after (not before) the cell-to-cell interference takes effect, which means the read voltage is near-optimal. Therefore to improve the BER performance, symmetric errors need to be considered. Even if the errors are symmetric, the magnitude of the errors is still limited. Fig. 3.10 (b) shows the 4-level

(X2) MLC channel with symmetric errors. We use a modulo-3 technique which can correct two types of errors (upward and downward). In theory, it is possible to correct $N-1$ error types by using a modulo-N technique. The modulo-3 based encoding and decoding methods by the limited magnitude error parity check (LMEPC) algorithm are given as follows.

LMEPC Mod 3 Algorithm for Symmetric Errors

Encoding

(Initialization) The q-ary $m \times n$ data:

$$D = \{d_{1,1}, d_{1,2}, \ldots, d_{1,n}, d_{2,1}, \ldots, d_{m,n}\}$$

1) *Get the remainder of message by modulo 3.*

$\bar{D} = \{\bar{d}_{1,1}, \bar{d}_{1,2}, \ldots, \bar{d}_{m,n}\}$ *where* $\bar{d}_{i,j} = d_{i,j} \bmod 3$.

2) *Generate the parity codes for each row and column.*

$P_r = \{p_{r_k}\}$ *and* $P_c = \{p_{c_k}\}$ *where* $p_{r_k} = (\sum_{i=1}^{n} \bar{d}_{k,i}) \bmod 3$ *and* $p_{c_k} = (\sum_{i=1}^{m} \bar{d}_{i,k}) \bmod 3$.

3) *Store the message data and the parity check codes.*

Parity codes are converted to q-ary data,

$P_r \to \hat{P}_r, P_c \to \hat{P}_c$.

Decoding

(Initialization) The q-ary received data and parity codes with interference and hard decision:

$D' = \{d'_{1,1}, d'_{1,2}, \ldots, d'_{m,n-1}, d'_{m,n}\}$, $\hat{P'}_r$, $\hat{P'}_c$.

The q-ary parity cell data is converted back to binary data:

$\hat{P'}_r \rightarrow P'_r = \{p'_{r_k}\}$, $\hat{P'}_c \rightarrow P'_c = \{p'_{c_k}\}$.

1) *Get the remainder of received message by* $\mod 3$.

$\bar{D'} = \{\bar{d'}_{1,1}, \bar{d'}_{1,2}, \ldots, \bar{d'}_{m,n}\}$ *where* $\bar{d'}_{i,j} = d'_{i,j} \mod 3$.

2) *Generate the parity codes of the received message data for each row and each column.*

$\tilde{P}_r = \{\tilde{p}_{r_k}\}$ *and* $\tilde{P}_c = \{\tilde{p}_{c_k}\}$ *where* $\tilde{p}_{r_k} = (\sum_{i=1}^{n} \bar{d'}_{k,i}) \mod 3$ *and* $\tilde{p}_{c_k} = (\sum_{i=1}^{m} \bar{d'}_{i,k}) \mod 3$.

3) *Check the parity and obtain the location of error.*

If $\tilde{p}_{r_i} \neq p'_{r_i}$ *&* $\tilde{p}_{c_j} \neq p'_{c_j}$, *the error location is* $d'_{i,j}$.

4) *Decide the type of error and make a correction.*

If $(p'_{r_i} - \tilde{p}_{r_i}) \mod 3 = 2$, *an upward error occurs and* $d'_{i,j} \leftarrow d'_{i,j} - 1$.

If $(p'_{r_i} - \tilde{p}_{r_i}) \mod 3 = 1$, *a downward error occurs and* $d'_{i,j} \leftarrow d'_{i,j} + 1$.

In the symmetric error casem, we can determine whether the error is upward or downward by using $\mod 3$ parity check methods. If we use entropy coding, a ternary parity

symbol $\{0, 1, 2\}$ can be represented by $\log_2 3$ bits which is optimal. If there are m rows and n columns in a data block, there are mn message cells for 2^k-level MLC flash memories, and the corresponding code rate R_c is upperbounded by

$$R_c \leq \frac{mn}{mn + \lceil \log_2 3^{\frac{m}{k}} \rceil + \lceil \log_2 3^{\frac{n}{k}} \rceil}. \tag{3.29}$$

If m is equal to n, the code rate is maximal as expected.

We simulated the two proposed error correction codes. One is the asymmetric limited magnitude parity check codes $(\mathrm{mod}\ 2)$ for asymmetric errors, and the other is the symmetric limited magnitude parity check codes for symmetric errors $(\mathrm{mod}\ 3)$. Bit error rate (BER) is used as the performance measure. We simulate the algorithm for a 8-level flash memory model (3 bits in a cell) with an equal noise distribution model. The equal noise distribution model assumes that each level has equal threshold voltage distribution which is Gaussian. In our simulations, the center threshold voltages of the 8-levels range from -0.57 V to 3.42 V, and there is 0.57 V gap between adjacent levels. Each level has a Gaussian distribution with 3 σ of 0.46 V. The cell-to-cell interference model in [11] is also applied. The average cell-to-cell interference assuming random data is estimated to be about 0.07 V. One block consists of 67,384 cells, and 2000 blocks (about 135 million cells) are used for simulations. The hard-decision read voltages are assumed to be adjusted already based on the the average interference level. The modified program (writing) order with the LSB/MSB method in [11] is applied to reduce the cell to cell interference, which is commonly used in industry.

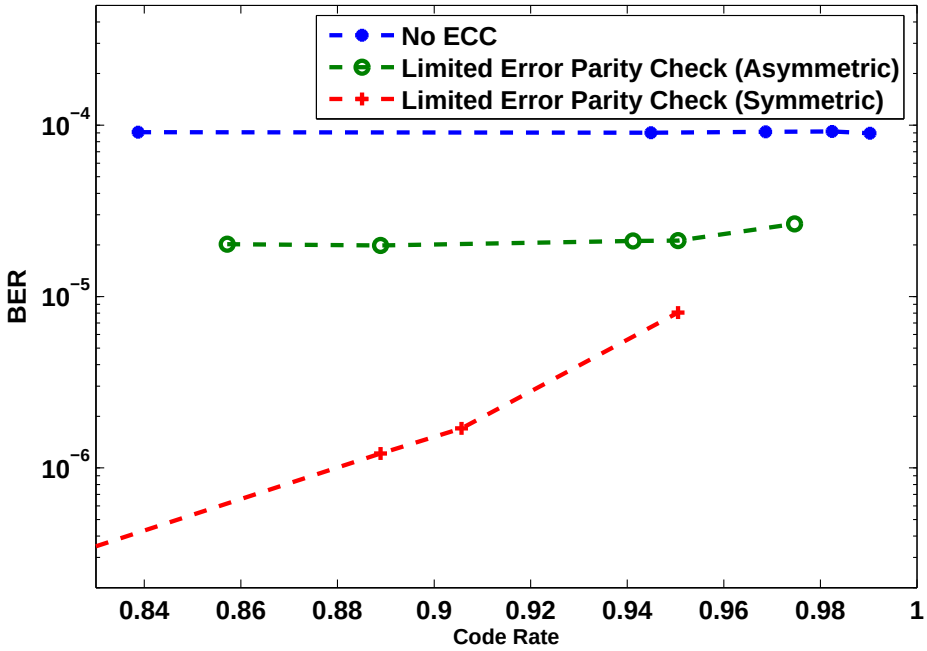

Figure 3.11 BER plot of the asymmetric and symmetric limited-magnitude error parity check codes.

Fig. 3.11 shows the BER performance of limited-magnitude parity check codes for asymmetric and symmetric errors. As the code rate decreases, the data block size gets smaller and the error correction improves. However the bit error rate of the asymmetric codes stays almost the same when the code rate is below 0.95. At a low code rate, the BER of the symmetric parity check codes is better than that of the asymmetric parity check codes because the BER performance at a low code rate is affected by symmetric errors rather than asymmetric errors. These symmetric codes reduce the bit error rate by a factor of about 10^3 compared to the no ECC case, which depends on the interference and the noise model used

in the simulations.

Although the code rate of LMEPC is lower than that of BCH, the proposed LMEPC methods have advantages in terms of computational complexity of encoding and decoding processes. The decoder complexity (X_{BCH}) of the BCH codes has the order of $O(mn(\log_2(mn))^2)$ [28], and it can be easily shown that the decoder complexity (X_{LMEPC}) of the proposed LMEPC codes has the order of $O(mn)$. For a large block size ($mn \gg 1$), we have $X_{\text{LMEPC}} \ll X_{\text{BCH}}$. The LMEPC-symmetric code have a lower code rate than the asymmetric code because a larger number of error types can be corrected. With 2^k-level algorithms, the code rate gets larger as k increases. In other words, the LMEPC codes get more efficient in terms of code rate with higher number of levels for each cell.

3.6 Summary

Error correction codes that are effective for multi-level cell flash memory and non-binary WOM codes are discussed. We discussed the potential problems of existing error correction codes, and show that proposed bidirectional limited-magnitude codes are more suitable to practical flash memory devices in simulations and analyses. A single bidirectional error correction code with the distinct sum set shows the optimal performance based on a minimum magnitude distinct sum set with practical parameters. A double error correction code based on distict sum set is also introduced. Bidirectional limited mgatnitude error correction codes have the practical error correcting capability based on conventional non-binary error

correction codes. Key advantages of these bidirectional error correction codes are that it can reduce the parity size, and that it has better error correction performance than the conventional error correction codes when the code rate is equal. Practical issues of encoding and decoding for the proposed method are discussed, and efficient methods are proposed.

Furthermore, two error correction codes for non-binary WOM codes are discussed in this chapter. The proposed codes deal with parity splitting methods for the WOM code property. The advantages of the proposed methods are that these are practical and systematic codes, and their encoding and decoding processes have low complexity.

We also discuss effective asymmetric and symmetric error locating limited-magnitude parity check error correction codes with lower redundancy. One of the key advantages of the proposed method is that it has low encoding/decoding complexity compared to conventional correcting codes such as the BCH codes. Another notable advantage is the flexibility in choosing the code rate and the block size.

Chapter 4

On Interference Mitigating Codes for Multi-level Flash Memories

4.1 Introduction

As the number of levels is increased in the multi-level cell flash memory, the number of errors tends to increase because of the voltage shift by inter-cell coupling, charge leakage, temperature, program/read disturbance, etc [1] [2] [4]. V_T distribution is disturbed by three well-known major parasitic effects, which are cell to cell interference, background pattern dependency, and noise [11]. Among them, the cell to cell interference is known to be a dominant factor. It is known that the cell to cell interference is mainly caused by floating gate coupling with parasitic capacitance [5].

We discuss coding schemes to lower cell-to-cell interference (C2CI) in this chapter. C2CI is known to be proportional to the threshold voltage change of neighbor cells. The

write operation is performed only after the erase operation, and the amount of threshold voltage change is proportional to the symbol magnitude. Therefore, to minimize the generated interference, the average magnitude of message symbols needs to be decreased. This goal is related to conventional minimum energy coding (ME coding) [23], because the objective of the code is to reduce the average energy, and it can be used to generate less interference. It is originally used in wireless communications and networks for energy efficiency. However, minimum energy coding causes significant redundancy for uniform symbol frequency, and it leads in higher costs for flash storage devices. Therefore, we propose a new coding scheme to lower the magnitude and minimize redundancy. The proposed coding scheme deals with q-ary message codes, and generates fixed length codes. Message codewords are divided into several blocks, and are modified by modulo addition with some constant to minimize the average magnitude. We also propose low energy Huffman codes based on entropy coding when the frequency of symbols is not distributed uniformly. This scheme produces variable-length codes without redundancy. We modified Huffman codes to minimize average number of high bits ('1' bits). We show that proposed codes generate optimal codewords which have minimum high bits with minimum average codeword length.

4.2 The modeling of generated interference in flash memory

Let us describe the C2CI and the sum of interference generated. Fig. 4.1 shows an interference model based on the parasitic capacitance between neighbor cells [20]. Suppose that

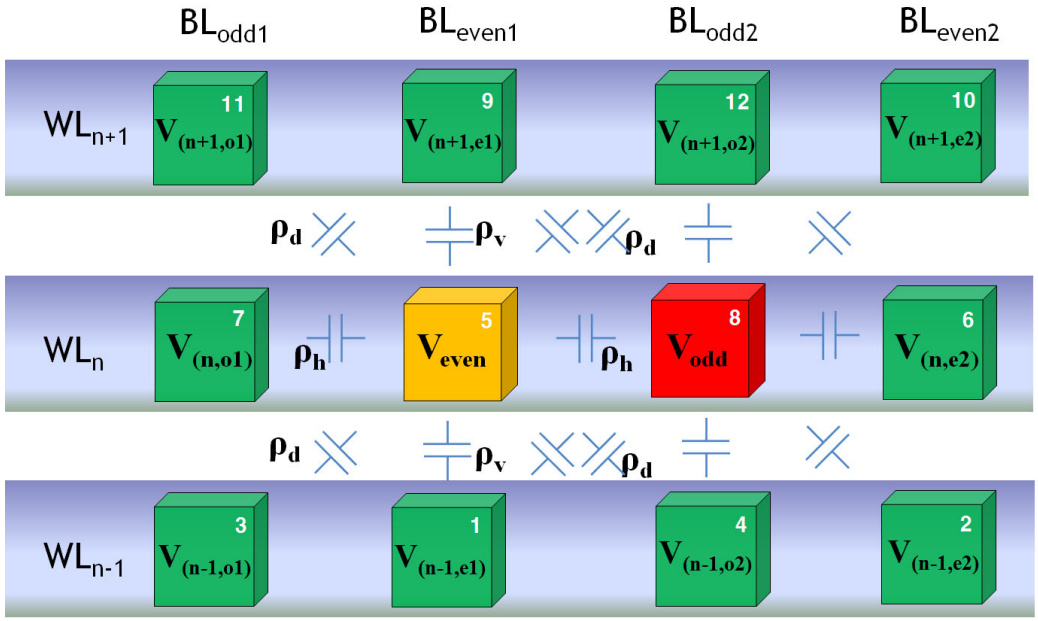

Figure 4.1 Interference model based on parasitic capacitances in a NAND flash array.

ρ_h, ρ_v, and ρ_d are the coupling coefficients for the horizontal, the vertical, and the diagonal neighbor cells, respectively. V is the threshold voltage of the cell. If we assume that the full-sequence programming strategy is being used, only after all the cells on a word line have been programmed can the next word line cells be programmed.

There are two kinds of interference estimation :

(a) total interference received by 'defense' cell

(b) total interference generated by 'offense' cell

All cells can be both defensive and offensive at the same time. First, we discuss the received interference model (case (a)). The interference in terms of threshold voltage shift

(ΔV) is given by

$$\begin{aligned}\sum \mathcal{I}^{\mathcal{R}}{}_{even} &= \rho_h(\Delta V_{(n,o1)} + \Delta V_{n,o2}) \\ &\quad +\rho_d(\Delta V_{(n+1,o1)} + \Delta V_{(n+1,o2)}) \\ &\quad +\rho_v(\Delta V_{(n+1,e1)})\end{aligned} \tag{4.1}$$

$$\begin{aligned}\sum \mathcal{I}^{\mathcal{R}}{}_{odd} &= \rho_d(\Delta V_{(n+1,e1)} + \Delta V_{(n+1,e2)}) \\ &\quad +\rho_v(\Delta V_{(n+1,o2)}).\end{aligned} \tag{4.2}$$

where the superscript $\mathcal{R}$ stands for reception. For example, the 5^{th} written cell V_{even} is interfered with the 7^{th}, the 8^{th}, the 9^{th}, the 11^{th}, and the 12^{th} written cells. The 8_{th} written cell V_{odd} is interfered with the 9^{th}, the 10^{th}, and the 12^{th} written cells.

In case (b), The interference sum $\Sigma\mathcal{I}^{\mathcal{G}}$ generated by an one cell can be modeled as

$$\sum \mathcal{I}^{\mathcal{G}}{}_{even} = \Delta V_{even}(\rho_v + 2\rho_d) \tag{4.3}$$

$$\sum \mathcal{I}^{\mathcal{G}}{}_{odd} = \Delta V_{odd}(2\rho_h + \rho_v + 2\rho_d). \tag{4.4}$$

For instance, the 5^{th} written cell V_{even} interferes to the 1^{st}, the 3^{rd}, and, the 4^{th} written cells. The 8_{th} written cell V_{odd} interferes with the 1^{st}, the 2^{nd}, the 4^{th}, the 5^{th}, and the 6^{th} written cells. These expressions show that a cell level (magnitude) change is related to the interference quantity. In the flash memory process, the write (programming) operation is performed only after the erase operation. Therefore, the amount of threshold voltage change is proportional to the magnitude. For example, if the message alphabet is $\{0, 1, \ldots, 7\}$ in a 8-ary cell, writing the symbol '5' on a cell causes the threshold voltage change amount

$\Delta V = 5k$ with constant k. Therefore, the minimization of the generated interference sum $\Sigma\mathcal{I}^{\mathcal{G}}$ is related to the reduction of the average magnitude.

4.3 Coding schemes for interference mitigation

4.3.1 Minimum energy coding

We now focus on the generated interference sum $\Sigma\mathcal{I}^{\mathcal{G}}$, and propose new coding schemes to mitigate the interference in this section. As mentioned above, average magnitude is connected to lowering C2CI, and this goal is related to minimum energy coding (ME coding) [23]. This is a simple coding scheme which will be used as a benchmark. The objective of ME coding is to reduce the average number of high bits ('1') in a codeword, and it is used in wireless communications and networks for energy efficiency [24] [25].

ME coding consists of two steps, coding optimality and codebook optimality. Let $\mathcal{Q}'$ be a source alphabet with message probabilities $p = \{p_1 \geq p_2 \geq \cdots \geq p_{q'}\}$. Let the codeword w_i of the codebook $W_q = \{w_1, w_2, \cdots, w_{q'}, \cdots, w_q\}$, which is in the ascending order of magnitude. A minimum codebook of q'-codewords, W_{min}, consists of the first q' codewords of the whole codeword set W_q having the least magnitude, $W_{min} = \{w_1, w_2, \cdots, w'_q\} \in W_q$. The codes that minimize the average magnitude are given by the minimum codebook with each codeword assigned to a symbol. We assumed that q-level flash memories are used. Therefore, high level (magnitude) symbols will be used instead of high bits. For example, let us assume that there are five messages ($q' = 5$) to encode with ME codes. We have

the following codebook, $W_q = \{0, 1, 2, 3, 4, 5, 6, 7\}$. Next, we choose the five codewords which have the lowest magnitude to form the ME codebook: $W_{min} = \{0, 1, 2, 3, 4\}$. Finally, we assign these code words to our five messages. We assign the most frequent message to 0 the next four most frequent messages to 1, 2, 3, and 4.

There is also extended ME coding to minimize the number of high bits. The authors of [24] consider an extreme approach where they use at most one high bit in a codeword. This results in the maximum reduction of total number of ones in the transmitted codeword bit sequence. For mapping of M source symbols, we need a codeword of length $M - 1$. This is because an all-zero source symbol is mapped to an all-zero codeword sequence, and the remaining $M - 1$ source symbols can be mapped to codewords with $M - 1$ bits in it, where each codeword has only one high bit. For example, we assume that the 8 symbols $\{a, b, c, d, e, f, g, h\}$ are considered. When we use the 8-ary code, it can be mapped to $\{0, 1, 2, 3, 4, 5, 6, 7\}$ codewords. Otherwise, the codes in [24] show that it can be mapped to $\{0000000, 0000001, 0000010, 0000100, 0001000, 0010000, 0100000, 1000000\}$. However, this extreme form of [24] make the code length too long, and it can be inefficient for storage devices.

In summary, ME coding schemes in [23] [24] are simple and can be efficient when the symbol frequency is already known, and when the distribution of the frequency is not uniform. However, the symbols are generated randomly in general, and the symbol frequency is assumed to have a uniform distribution. In this case, ME coding significantly increases

the redundancy, thereby resulting in higher cost for flash storage devices.

4.3.2 Module shift coding

We propose a new coding scheme to lower the magnitude of the message, and minimize redundancy. The proposed coding scheme deals with q-ary message codes, and generates a fixed length code. Message codewords are divided into several blocks and are modified by modulo addition with some constants to minimize the average magnitude. We call this method '*module shift (MS) coding*'. If $\mathcal{C}'$ is an encoded codeword matrix, and $\mathcal{C}$ is a message codeword matrix, $\mathcal{C}' = |\mathcal{C} + \mathcal{M}|_q$. $\mathcal{M}$ consist of arbitrary integer constants. Note that $|\mathcal{C} + \mathcal{M}|_q$ means modulo q addition. For example, we have

$$\begin{aligned}
\mathcal{C}' &= \left| \begin{pmatrix} 2 & 8 & 9 & 0 \\ 1 & 11 & 8 & 3 \\ 5 & 2 & 3 & 4 \\ 4 & 9 & 4 & 7 \end{pmatrix} + \begin{pmatrix} 0 & 4 & 4 & 0 \\ 0 & 4 & 4 & 0 \\ 8 & 4 & 0 & 8 \\ 8 & 4 & 0 & 8 \end{pmatrix} \right|_{12} \\
&= \begin{pmatrix} 2 & 0 & 1 & 0 \\ 1 & 3 & 0 & 3 \\ 1 & 6 & 3 & 0 \\ 0 & 1 & 4 & 3 \end{pmatrix}.
\end{aligned}$$

An average magnitude of the message codeword matrix $\mathcal{C}$ is 5. but the average magnitude of the encoded codeword matrix $\mathcal{C}'$ is only 1.75. Therefore, an encoded codeword has a

reduced magnitude, and results in lowering the generated interference sum.

The code construction is described as follows. A part of message $\mathcal{C}$ is $\Lambda = \{\lambda_1, ..., \lambda_n\}$ where λ_i is the ith q-ary message symbol, and $\lambda_i \in \mathcal{Q} = \{0, 1, \cdots, q-1\}$. The encoded symbols are given by $\lambda_i' = |\lambda_i + \sigma|_q$. If the sum of λ_i' is smaller than that of λ_i, ΔV_{even} and ΔV_{odd} in (4.3) and (4.4) can be reduced. We find the integer σ which satisfies

$$\hat{\sigma} = \arg\min_{\sigma \in \mathcal{Q}} \sum_{i=1}^{n} |\lambda_i + \sigma|_q. \tag{4.5}$$

Finding an optimal σ leads to significant extra complexity and redundancy. To get around this problem, σ is chosen from only a few choices. The q-ary symbols in $\mathcal{Q}$ are divided into several block called '*module*'. The number of modules is η, and the i_{th} module is defined as $\Omega_i = \{\omega | \theta_{i-1} \leq \omega < \theta_i, \omega \in \mathcal{N}\}$ where $i = 1, \cdots, \eta$, and $\mathcal{N}$ is the set of natural numbers. θ_i is the i_{th} threshold to distinguish modules with $\theta_0 = 0$ and $\theta_p = q$. Let us assume that the symbol frequency is distributed uniformly, q is a multiple of η, and each module has the same number of elements, $\Omega_i = \{\omega_0^i, \omega_1^i, \cdots, \omega_{\frac{q}{\eta}-1}^i\}$. We then have $\omega_k^i = \frac{q}{\eta}(i-1) + k$, the uniform symbol probability of $p(\omega_k^i) = \frac{\eta}{q}$, and an average magnitude of the ith module, $E(\Omega_i)$, is given by

$$\begin{aligned} E(\Omega_i) &= \sum_{k=0}^{\frac{q}{\eta}-1} \omega_k^i \cdot p(\omega_k^i) \\ &= (i - \frac{1}{2})\frac{q}{\eta} - \frac{1}{2}. \end{aligned} \tag{4.6}$$

For example, let us assume that $q = 12$, $\mathcal{Q} = \{0, 1, 2, \cdots, 10, 11\}$, $\eta = 3$, $\Omega_1 = \{0, 1, 2, 3\}$, $\Omega_2 = \{4, 5, 6, 7\}$, and $\Omega_3 = \{8, 9, 10, 11\}$. We then have $E(\Omega_1) = (1 - \frac{1}{2})(12/3) - \frac{1}{2} =$

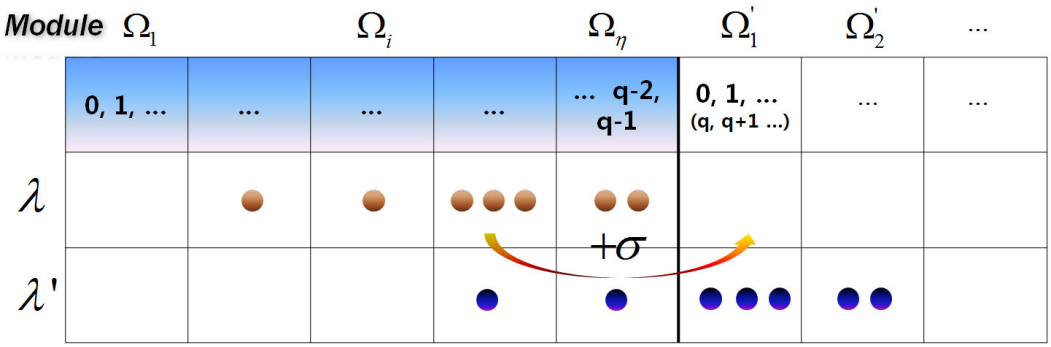

Figure 4.2 Module shift of MS coding

$1.5, E(\Omega_2) = 5.5$, and $E(\Omega_3) = 9.5$.

The symbols in the message block λ can be classified into several modules. If most symbols are included in the first (lowest) module, there is no need to shift levels. However, if most symbols are included in higher module, they should be shifted by modulo addition to the lower module. We define the module selection parameter ζ which means the number of shift levels of the message block to minimize the average magnitude. When $n_i = |\{\lambda_k | \lambda_k \in \Omega_i, \forall k\}|$, $n = n_1 + n_2 + \cdots + n_\eta$, and $\Xi = \{0, 1, \cdots, \eta - 1\}$, the module selection parameter ζ is

$$\begin{aligned} \zeta &= \arg\min_{k \in \Xi} \sum_{i=1}^{\eta} E(\Omega_i) \cdot n_{|i+k-1|_\eta + 1} \\ &= \arg\min_{k \in \Xi} \sum_{i=1}^{\eta} \left(\frac{q}{\eta} \cdot i - \frac{q}{2\eta} - \frac{1}{2}\right) \cdot n_{|i+k-1|_{\eta+1}} \\ &= \arg\min_{k \in \Xi} \sum_{i=1}^{\eta} i \cdot n_{|i+k-1|_{\eta+1}}. \end{aligned} \tag{4.7}$$

With this criterion, we find the module shift level and the module shift constant $\sigma = \frac{q}{\eta}\zeta$,

where $\frac{q}{\eta}$ is the size of a module.

The module shift encoding process is described as follows.

Module Shift Encoding

(Initialization) A part of message codeword $\mathcal{C}$ is

$\Lambda = \{\lambda_1, ..., \lambda_n\}$, $\Omega_i = \{\omega | \theta_{i-1} \leq \omega < \theta_i, \omega \in \mathcal{N}\}$, *$\theta_i$ is module threshold,*

$\Xi = \{0, 1, \cdots, \eta - 1\}$.

1) *Find the* $n_i = |\{\lambda_k | \lambda_k \in \Omega_i, \forall k\}|, \forall i$

2) *Estimate the module selection parameter*

$$\zeta = \underset{k \in \Xi}{arg\ min} \sum_{i=1}^{\eta} i \cdot n_{|i+k-1|_{\eta+1}}$$

3) $\sigma = \frac{q}{\eta}\hat{\zeta}$ *and* $\Lambda' = |\Lambda + \boldsymbol{\sigma}|_q$, *where* $\boldsymbol{\sigma} = \sigma \cdot \mathbf{1_n}$.

4) *Encoded codeword Λ' replaces Λ, and ζ is stored in q-ary cells separately.*

where $\mathbf{1_n}$ means a vector $(1, 1, \cdots, 1)$ of length n.

Example 4.1. let us assume that a part of codeword $\Lambda = \{8, 11, 2, 9\}$, $q = 12$, $\eta = 3$, $\Omega_1 = \{0, 1, 2, 3\}$, $\Omega_2 = \{4, 5, 6, 7\}$, $\Omega_3 = \{8, 9, 10, 11\}$. By (4.7), we find that $\zeta = 1$ and $\sigma = \frac{12}{3} \cdot 1 = 4$. We then have $\Lambda' = |\{8 + 4, 11 + 4, 2 + 4, 9 + 4\}|_{12} = \{0, 3, 6, 1\}$.

4.3.3 Low energy Huffman code

We considered the fixed-length code with a little redundancy in previous subsections, and will now discuss the variable-length code without redundancy. In source coding theory, the Huffman codes are the optimal codes for lossless data compression [21]. In the conventional Huffman encoding process, assigning 0 and 1 to the edge of the tree is not fixed, because it does not affect the average length of codewords or compression performance. However, it can affect the average magnitude, which is discussed here. We propose modified Huffman coding, the *Low Energy Huffman (LE-H) code* to minimize average high bit numbers. Let us assume that $\mathcal{Q} = \{0, 1, \ldots, q-1\}$. $f(\alpha)$ is the probability of symbol 'α'. $g(\alpha)$ is defined by the value of the branch from α, which is either '0' or '1'.

Low Energy Huffman Encoding

1) *Choose two letters α_k, β_k from $\mathcal{Q}$ with the smallest frequencies, and create a subtree that has these two characters. Label the root of this sub tree as γ.*

2) *Set the probability to $f(\gamma) = f(\alpha_k) + f(\beta_k)$. Remove α_k, β_k and add γ creating new $\mathcal{Q} = \mathcal{Q} \cup \{\gamma\} - \{\alpha_k, \beta_k\}$.*

3) *If $f(\alpha_k) \geq f(\beta_k)$, assign '0', '1' to $g(\alpha_k)$ and $g(\beta_k)$, respectively. Otherwise, assign '1' and '0' to these symbols.*

4) *If $|\mathcal{Q}| > 1$, k=k+1 and go to step 1.*

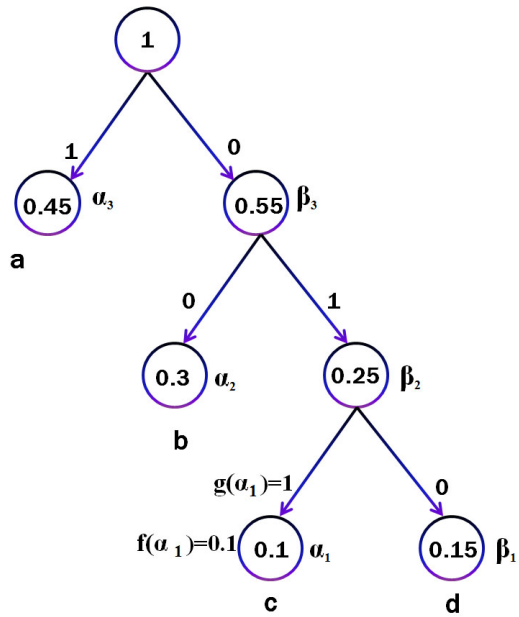

Figure 4.3 Low energy Huffman coding example

Using the algorithm above, an LE-H coding tree $\mathcal{T}$ is generated. There are $(q-1)$ pairs of symbols which have the same root, and '0' or '1' is assigned to the k_{th} pair of symbols by Step 3 of the algorithm. $\mathcal{P}_0$ and $\mathcal{P}_1$ are the average number of '0' and '1' per encoded codeword, and $\mathcal{P}_0 + \mathcal{P}_1$ is the average length of encoded codeword. We have

$$\mathcal{P}_1 = \sum_{k=1}^{q-1} \{g(\alpha_k) f(\alpha_k) + g(\beta_k) f(\beta_k)\} \tag{4.8}$$

$$\mathcal{P}_0 = \sum_{k=1}^{q-1} \{\tilde{g}(\alpha_k) f(\alpha_k) + \tilde{g}(\beta_k) f(\beta_k)\} \tag{4.9}$$

where $\tilde{g}(\alpha_k)$ means reversal of $g(\alpha_k)$, $0 \to 1$ or $1 \to 0$.

Theorem 4. Low energy Huffman code generates optimal codeword which has minimum

high bits with minimum average codeword length.

Proof. Let us assume that the i_{th} couple of symbols in tree $\mathcal{T}$ which have the same root are α_i and β_i. The average number of '1', $\mathcal{P}_1 = \sum_{i=1}^{q-1}\{g(\alpha_i)f(\alpha_i) + g(\beta_i)f(\beta_i)\}$. If we exchanges $g(\alpha_j)$ for $g(\beta_j)$, $\mathcal{P}_1' = \sum_{i=1\setminus j}^{q-1}\{g(\alpha_i)f(\alpha_i) + g(\beta_i)f(\beta_i)\} + g(\beta_j)f(\alpha_j) + g(\alpha_j)f(\beta_j)$. Then $\mathcal{P}_1' - \mathcal{P}_1 = \Big(g(\alpha_j) - g(\beta_j)\Big)\Big(f(\beta_j) - f(\alpha_j)\Big) \geq 0$ due to Step 3 of the algorithm. The inequality $\mathcal{P}_1' \geq \mathcal{P}_1$ is always true, and $\mathcal{P}_1$ is the minimum. Therefore, the codes generated by LE-H coding have minimum high bit numbers with optimal average codeword length.

For example, in Fig 4.3, alphabets a, b, c, and d are encoded. $\mathcal{P}_0 = 0.55{+}0.3{+}0.15 = 1$ and $\mathcal{P}_1 = 0.45 + 0.25 + 0.1 = 0.8$. The encoded codewords for alphabets 'a','b','c', and 'd' are '1', '00', '011', and '010', respectively. The average length of codeword is $\mathcal{P}_0 + \mathcal{P}_1 = 1.8$ and the entropy $\mathcal{H} = 1.7822$ [21].

4.4 Performance analysis of proposed coding schemes

4.4.1 Performance analysis of ME codes

To begin with, we consider three kinds of parameters, $\mathcal{R}$, $\mathcal{A}$, and $\mathcal{L}$ to estimate the performance. $\mathcal{R}$ refers to the code rate, and it is the proportion of information bits. When the code rate is k/n, for every k bits of useful information, the codes generate n bits of data, of which

$n-k$ bits are redundant. As redundancy increases, $\mathcal{R}$ decreases. $\mathcal{A}$ denotes the magnitude reduction ratio of a codeword (redundancy included). $\mathcal{L}$ is the average magnitude of a cell. That is, $\mathcal{L}$ is related to the generated interference in (4.3) and (4.4). As the redundancy of the codes increases, $\mathcal{L}$ decreases but $\mathcal{A}$ may or may not be reduced. Although the average magnitude of each cell decreases, the sum of the magnitude of all the cells may increase due to the large amount of redundancy. The goals of the proposed algorithm are to minimize the redundancy (maximizing $\mathcal{R}$), to maximize $\mathcal{A}$, and to minimize $\mathcal{L}$.

Redundancy is required in ME coding to reduce the average magnitude of the cell. If q-ary m length messages are written to the q'-level m' length memory cell, the parameters should satisfy the inequality.

$$m' \log_2 q' \geq m \log_2 q \tag{4.10}$$

The q-ary m message symbols ($m\log_2 q$ bits) should be represented by q'-level m' symbols ($m'\log_2 q'$ bits). In the example of Sec. III-A, $m=4, q=8, q'=5, W_q=\{0,1,2,3,4,5,6,7\}$, and $W_{min}=\{0,1,2,3,4\}$. Therefore, $4\log_2 8 = 12$ bits are needed to store the message, $m'\log_2 5$ should be larger than 12. We have $12/\log_2 5 = 5.1681 < 6$, so that m' needs to be 6. The code rate of ME coding $\mathcal{R}_{ME}$ is given by

$$\mathcal{R}_{ME} = \frac{m}{m'} \leq \log_q q'. \tag{4.11}$$

The total sum of average magnitude of q-ary m length code is $\frac{m(q-1)}{2}$, and the sum of average magnitude of encoded codeword by ME coding is $\frac{m'(q'-1)}{2}$ The magnitude reduction

ratio of generated codeword $\mathcal{A}_{ME}$ is given by

$$\begin{aligned}\mathcal{A}_{ME} &= \frac{\frac{m(q-1)}{2} - \frac{m'(q'-1)}{2}}{\frac{m(q-1)}{2}} \\ &\leq \frac{\frac{m(q-1)}{2} - \frac{m(q'-1)}{2}\log_{q'}q}{\frac{m(q-1)}{2}} \\ &= 1 - \frac{q'-1}{q-1}\log_{q'}q. \end{aligned} \tag{4.12}$$

The average magnitude of each cell of ME code $\mathcal{L}$ is

$$\mathcal{L}_{ME} = \frac{q'-1}{2}. \tag{4.13}$$

4.4.2 Performance analysis of MS codes

To analyze the performance of MS codes, we assume that $n = \sum_{k=1}^{\eta} n_{k,i}^{\zeta}$. n is the length of message symbols. $n_{k,i}^{\zeta}$ denotes the number of symbols which are included in the k_{th} module among n message symbols, and corresponds to the criterion in (4.7) when a module selection parameter is ζ. Note that i is the index for the combination. We define $N_i^{\zeta} = (n_{1,i}^{\zeta}, n_{2,i}^{\zeta}, \ldots, n_{\eta,i}^{\zeta})$. There are $\binom{n+\eta-1}{\eta-1}\frac{1}{\eta}$ combinations of N_i^{ζ} for each ζ if $q|\eta$, and $N^{\zeta} = \{N_1^{\zeta}, N_2^{\zeta}, \ldots, N_{\binom{n+\eta-1}{\eta-1}\frac{1}{\eta}}^{\zeta}\}$, which is the collection of all such combinations.

For example, let us assume that length of codeword $|\Lambda| = n = 5$, $q = 4$, $\mathcal{Q} = \{0,1,2,3\}$, $\eta = 2$, $\Omega_1 = \{0,1\}$, $\Omega_2 = \{2,3\}$. When $\zeta = 0$ (no shift case), $N_1^0 = (n_{1,1}^0, n_{2,1}^0) = (3,2)$, $N_2^0 = (4,1), N_3^0 = (5,0)$. When $\zeta = 1$ (shift by one module case), $N_1^1 = (0,5)$, $N_2^1 = (1,4), N_3^1 = (2,3)$. $N^0 = \{N_1^0, N_2^0, N_3^0\} = \{(3,2),(4,1),(5,0)\}$, and $N^1 = \{N_1^1, N_2^1, N_3^1\} = \{(0,5),(1,4),(2,3)\}$

When ζ is given, the frequency of the jth module element $\mathcal{F}_j^{\zeta}$ in a codeword is of length n given by

$$\mathcal{F}_j^{\zeta} = \sum_{i=1}^{\binom{n+\eta-1}{\eta-1}^{\frac{1}{\eta}}} \binom{n}{n_{1,i}^{\zeta}} \binom{n-n_{1,i}^{\zeta}}{n_{2,i}^{\zeta}} \cdots \binom{n-\sum_{s=1}^{\eta-1} n_{s,i}^{\zeta}}{n_{\eta,i}^{\zeta}} \cdot \left(\frac{1}{\eta}\right)^n \cdot n_{|j-\zeta-1|_\eta+1,i}^{\zeta}. \tag{4.14}$$

If n is not a multiple of η, the probability of the j_{th} module element is estimated by

$$p(\Omega_j) = \frac{1}{n} \sum_{\zeta'=0}^{\eta-1} \mathcal{F}_j^{\zeta'}. \tag{4.15}$$

An average magnitude of encoded cell $E(\lambda')$ is

$$\begin{aligned} E(\lambda') &= \sum_{j=1}^{\eta} p(\Omega_j) E(\Omega_j) \\ &= \sum_{j=1}^{\eta} \left(\frac{1}{n} \sum_{\zeta'=0}^{\eta-1} \mathcal{F}_j^{\zeta'}\right) \left((j-\frac{1}{2})\frac{q}{\eta} - \frac{1}{2}\right) \\ &= \sum_{j=1}^{\eta} \left\{ \frac{1}{n} \sum_{\zeta'=0}^{\eta-1} \left(\sum_{i=0}^{\binom{n+\eta-1}{\eta-1}^{\frac{1}{\eta}}} \binom{n}{n_{1,i}^{\zeta'}} \binom{n-n_{1,i}^{\zeta'}}{n_{2,i}^{\zeta'}} \right.\right. \\ &\quad \left. \cdots \binom{n-\sum_{s=1}^{\eta-1} n_{s,i}^{\zeta'}}{n_{\eta,i}^{\zeta'}} \cdot \left(\frac{1}{\eta}\right)^n \cdot n_{|j-\zeta'-1|_\eta+1,i}^{\zeta'} \right) \\ &\quad \left. \cdot \left((j-\frac{1}{2})\frac{q}{\eta} - \frac{1}{2}\right) \right\} \end{aligned} \tag{4.16}$$

where $E(\Omega_j) = (j-\frac{1}{2})\frac{q}{\eta} - \frac{1}{2}$. For example, $\eta = 2$ and n = odd, the average encoded magnitude $E(\lambda')$ is

$$E(\lambda') = \frac{q-2}{4} + \left(\frac{1}{2}\right)^n \frac{q}{n} \sum_{i=0}^{\frac{n-1}{2}} \binom{n}{i} i. \tag{4.17}$$

The details are given in Appendix A.

The average magnitude without encoding is given by $E(\lambda) = \frac{q-1}{2}$. To achieve the magnitude reduction ratio of the generated codeword $\mathcal{A}_{MS}$, the average redundancy magnitude should be estimated first. If we know the module selection parameter ζ of each block, decoding is possible, and ζ is an integer in $[0, \eta - 1]$. Accordingly, $\log_2\eta$ redundancy bits ($\frac{\log_2\eta}{\log_2 q}$ redundancy cells) are needed for each block. That is, each message cell generates $\frac{\log_2\eta}{\log_2 q} \cdot \frac{1}{n}$ redundancy cells. $E(\mathcal{G})$, the average redundancy magnitude is given by

$$E(\mathcal{G}) = \frac{q-1}{2} \cdot \frac{\log_2\eta}{\log_2 q} \cdot \frac{1}{n}. \tag{4.18}$$

$\mathcal{A}_{MS}$ is given by

$$\begin{aligned} \mathcal{A}_{MS} &= \frac{E(\lambda) - (E(\lambda') + E(\mathcal{G}))}{E(\lambda)} \\ &= \frac{\frac{q-1}{2} - (E(\lambda') + \frac{q-1}{2n} \cdot \frac{\log_2\eta}{\log_2 q})}{\frac{q-1}{2}} \\ &= 1 - \frac{\{2E(\lambda') + (q-1)\log_q\eta\}}{n(q-1)} \end{aligned} \tag{4.19}$$

The average magnitude of one cell $\mathcal{L}_{MS}$ is given by

$$\mathcal{L}_{MS} = E(\lambda') \tag{4.20}$$

The redundancy is $\log_2\eta$ bits for each block Λ. Each block of length n has $n\log_2 q$ bits. The code rate $\mathcal{R}_{MS}$ is given by

$$\mathcal{R}_{MS} \leq \frac{n\log_2 q}{n\log_2 q + \log_2\eta}. \tag{4.21}$$

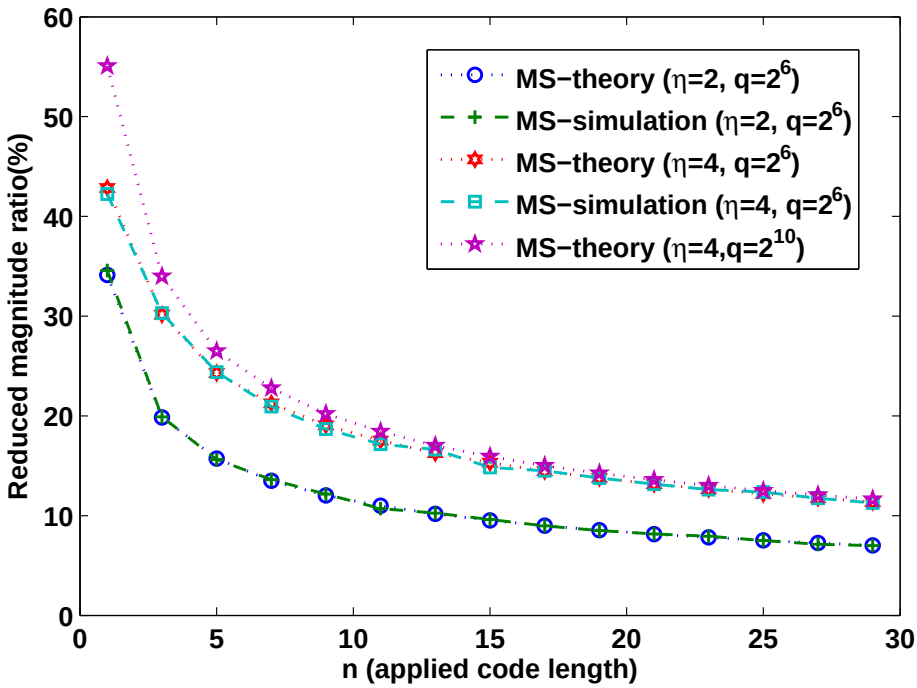

Figure 4.4 Reduced magnitude ratio of the module shift (MS) algorithms

Fig. 4.4 shows the reduced average magnitude ratio with varying code lengths of each block n. It can be seen that the plots of the simulation with 10,000 message blocks correspond to the theoretical plots. It is observed that the ratio increases as the size of q increases.

To compare the performance of ME codes and MS codes, we simulated the performance with equal redundancy, and $\mathcal{A}_{ME}$ and $\mathcal{A}_{MS}$ were plotted with $\mathcal{R}_{ME}$ and $\mathcal{R}_{MS}$. Fig. 4.5 shows the average magnitude reduction ratio with respect to code rate. The MS codes showed a better performance than the ME codes in the high code rate range. When the number of module η increases in the MS codes plot, the average reduced magnitude is improved, but the redundancy also increases rapidly, which is indicated by the crossing of the ME and

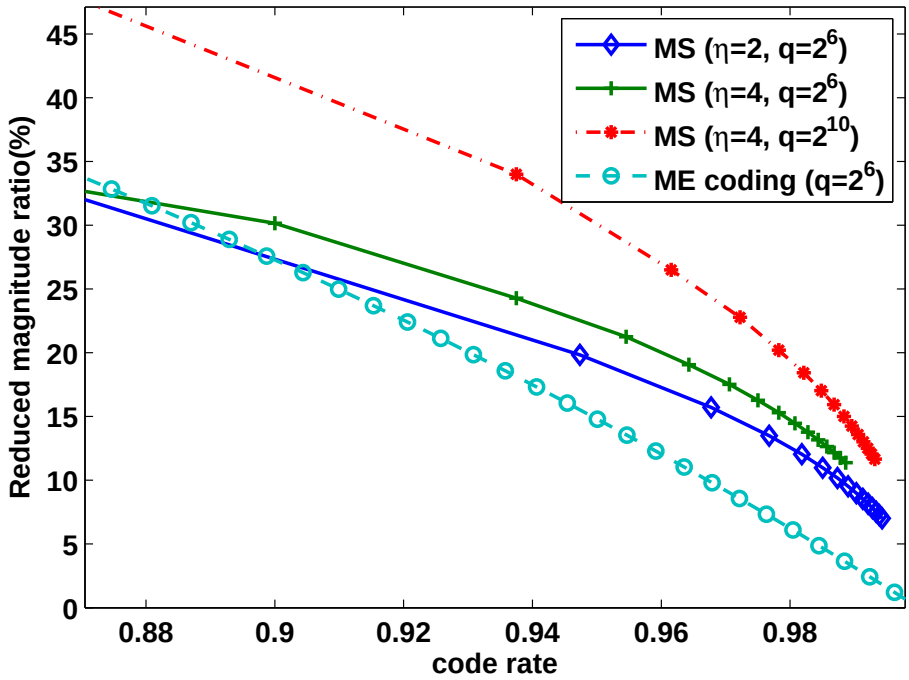

Figure 4.5 Magnitude reduction ratio of encoded codeword

the MS curves.

4.4.3 Performance of low-energy Huffman codes

MS codes generate fixed length code for a uniform symbol frequency with some redundancy, and the LE-H codes generates variable length codewords without redundancy, which is effective only for a non-uniform symbol frequency. LE-H codes are not channel codes, but a kind of source codes. Therefore, the encoder of LE-H generates codewords with reduced codeword length, and the code rate $\mathcal{R}$ is not a meaningful figure of merit. Average magnitude of a cell for a block of length n is $\frac{q-1}{2}$. We define κ is the compression ratio of

Huffman code, and $\kappa = \frac{\text{average length}}{\log_2 q}$. Each encoded symbol has high bit with probability $\frac{\mathcal{P}_1}{\mathcal{P}_0+\mathcal{P}_1}$, and an average level of encoded symbol become $\frac{\mathcal{P}_1}{\mathcal{P}_0+\mathcal{P}_1} \cdot (q-1)$. $\mathcal{A}_{LE-H}$ is given by

$$\begin{aligned} \mathcal{A}_{LE-H} &= \frac{\frac{q-1}{2} - \kappa \frac{\mathcal{P}_1}{\mathcal{P}_0+\mathcal{P}_1} \cdot (q-1)}{\frac{q-1}{2}} \\ &= 1 - \kappa \frac{2\mathcal{P}_1}{\mathcal{P}_0+\mathcal{P}_1}. \end{aligned} \tag{4.22}$$

The average magnitude of one cell $\mathcal{L}_{LE-H}$ is given by

$$\mathcal{L}_{LE-H} = \frac{(q-1)\mathcal{P}_1}{\mathcal{P}_0+\mathcal{P}_1}. \tag{4.23}$$

MS coding is efficient only with an uniform symbol frequency, but LE-H code is efficient only with non-uniform symbol frequency. Therefore, LE-H cannot be compared with MS and ME coding directly, but we show several examples of the performance of the LE-H codes. Table 4.1 shows the performance with different parameters. The ratio κ decreases as the variance of symbol frequency increases, and $\mathcal{A}_{LE-H}$ also increases. $\mathcal{A}_{LE-H}$ is independent of alphabet size q.

q	symbol frequency	$\mathcal{A}_{LE-H}$	κ
4	(0.15, 0.01, 0.80, 0.04)	74 %	0.625
4	(0.05, 0.6, 0.1, 0.25)	40 %	0.775
4	(0.25, 0.4, 0.15, 0.2)	20 %	0.975

Table 4.1 Magnitude reduction ratio of LE-H codes

4.4.4 C2CI reduction performance

Let us discuss the performance of the proposed algorithm in terms of the reduction in C2CI. $\mathcal{L}_x$ is the average magnitude of one cell without encoding, and $E(\epsilon)$ is the average generated interference of each cell without encoding. The coupling coefficients for generated interference are $\rho_v + 2\rho_d$ for even cells in (4.3) and $2\rho_h + \rho_v + 2\rho_d$ for odd cells in (4.4). The coupling coefficients will be averaged out if we assume that the numbers of the even cells and the odd cells are equal. $E(\epsilon)$ is given by

$$\begin{aligned} E(\epsilon) &= \mathcal{L}_x \frac{(2\rho_h + \rho_v + 2\rho_d) + (\rho_v + 2\rho_d)}{2} \\ &= \mathcal{L}_x(\rho_h + \rho_v + 2\rho_d) \end{aligned} \tag{4.24}$$

where $\mathcal{L}_x = \frac{q-1}{2}$. $\mathcal{L}_e$ is the average magnitude of encoded cell, such as $\mathcal{L}_{ME}$ or $\mathcal{L}_{MS}$. The performance is better when every cell is encoded, but there is a trade-off between the redundancy and the performance. On the other hand, when only the cells that generate the most interference are encoded by the proposed codes, there is a trade-off between interference reduction and code rate.

When ΔV_{odd} and ΔV_{even} are equal, ${\mathcal{I}^{\mathcal{G}}}_{odd}$ is larger than ${\mathcal{I}^{\mathcal{G}}}_{even}$, because $(2\rho_h + \rho_v + 2\rho_d) > (\rho_v + 2\rho_d)$ in (4.3), (4.4). That is, the threshold voltage change of the odd cells plays a more important role than that of the even cells. Therefore, we compare the case where all the cells are encoded with the case where only the odd cells are encoded. $E(\epsilon_a)$ and $E(\epsilon_o)$ are the average generated interference of encoded cells when all the cells are encoded and

only the odd cells are encoded, respectively. We have

$$E(\epsilon_a) = E(\epsilon) = \mathcal{L}_e(\rho_h + \rho_v + 2\rho_d) \tag{4.25}$$

$$E(\epsilon_o) = \frac{\mathcal{L}_e(2\rho_h + \rho_v + 2\rho_d) + \mathcal{L}_x(\rho_v + 2\rho_d)}{2}. \tag{4.26}$$

Due to $\mathcal{L}_x > \mathcal{L}_e$, $E(\epsilon_o) - E(\epsilon_a) = (\rho_v + 2\rho_d)(\mathcal{L}_x - \mathcal{L}_e) > 0$. It means that $E(\epsilon_o)$ is always larger than $E(\epsilon_a)$. However, in the case when only the odd cells are encoded, the redundancy is smaller, and the performance can be different with equal redundancy. When only the odd cells are encoded, $m/2$ odd cells are encoded by the ME code among m cells, and the others (even cells) are not encoded. Due to $m' \geq m(\log_2 q/\log_2 q')$ in (4.10), the length of the encoded codeword is $m/2 + m'/2 \geq m/2 + m/2(\log_2 q/\log_2 q') = m/2 + m(\log_{q'} q)$, and the message length is m. Therefore, the code rate of ME coding is

$$\mathcal{R}_{ME_o} \leq \frac{m}{\frac{m}{2} + \frac{m\log_{q'} q}{2}} = \frac{2}{1 + \log_{q'} q}. \tag{4.27}$$

In the MS coding case, if one block has $2n$ cells, n odd cells are encoded by the MS code, and the others (n even cells) are not encoded. The redundancy for each block is $\log_2 \eta$ bit. The encoded codeword bit size is $2n\log_2 q + \log_2 \eta$ bits. The code rate of MS coding when only odd cells are encoded is given by

$$\mathcal{R}_{MS_o} = \frac{2n\log_2 q}{2n\log_2 q + \log_2 \eta}. \tag{4.28}$$

The C2CI reduction ratio of one cell are shown in Fig. 4.6 and 4.7. In the simulation, $\rho_h = 0.11, \rho_v = 0.07$, and $\rho_d = 0.02$. The MS codes have better performance than ME

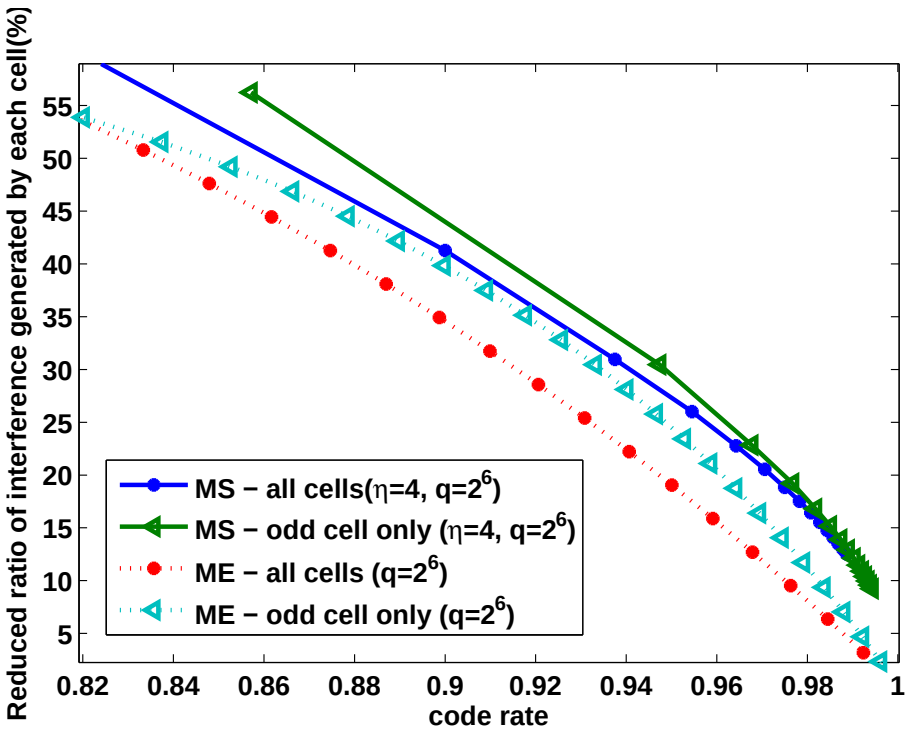

Figure 4.6 C2CI reduction ratio of the proposed algorithms

codes for all code rate. When only odd cells are encoded, it has better performance than the case when all cells are encoded. This is due to the fact that the odd cells play more important roles than the even cells to reduce interference.

We now consider the effects of the coupling coefficient ρ. Fig. 4.8 shows the average interference generated by each cell with two $\boldsymbol{\rho}$ parameters. In the simulation, $\boldsymbol{\rho_1} = (\rho_h, \rho_v, \rho_d) = (0.11, 0.07, 0.02)$, $\boldsymbol{\rho_2} = (0.14, 0.05, 0.01)$, $\eta = 4$, and $q = 2^6$. The generated interference for all cells encoding case and the no encoding case are equal when two $\boldsymbol{\rho_1}, \boldsymbol{\rho_2}$ parameters are used. When only the odd cells are encoded, there is a large difference in ρ_h, ρ_v and ρ_d giving rise to differences in performance. This means that the horizontal

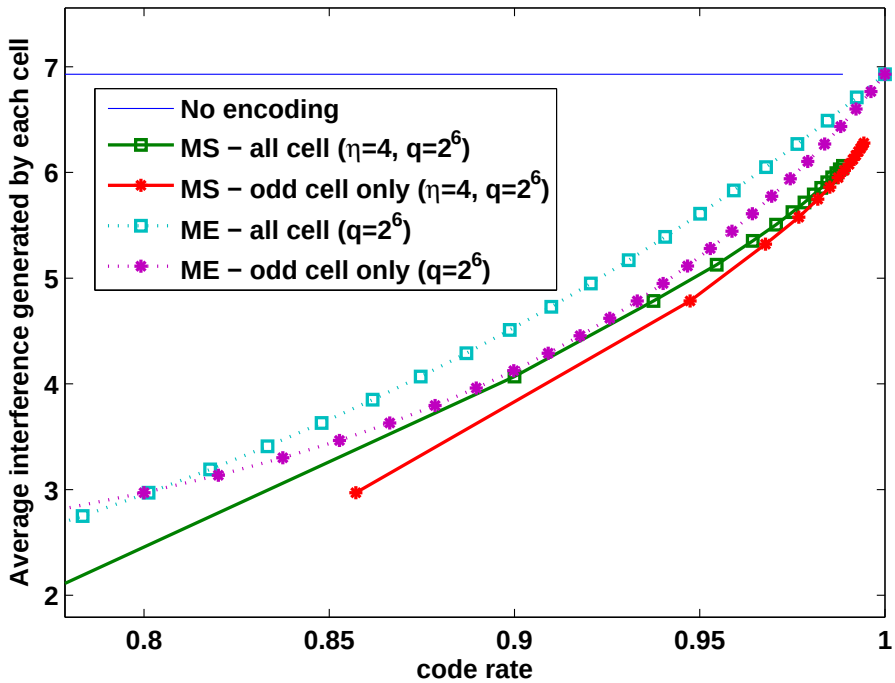

Figure 4.7 C2CI reduction performance of the two algorithms

coupling coefficient ρ_h plays a more critical role for odd cell encoding schemes.

4.5 Summary

New code construction methods are proposed for interference reduction in flash memory devices to mitigate cell-to-cell interference. The proposed schemes use fixed length code with small extra redundancy, and the other uses variable length entropy coding with no extra redundancy. In summary, the minimum energy codes are the simplest, but it increases redundancy. The module shift codes show better performance than the minimum energy

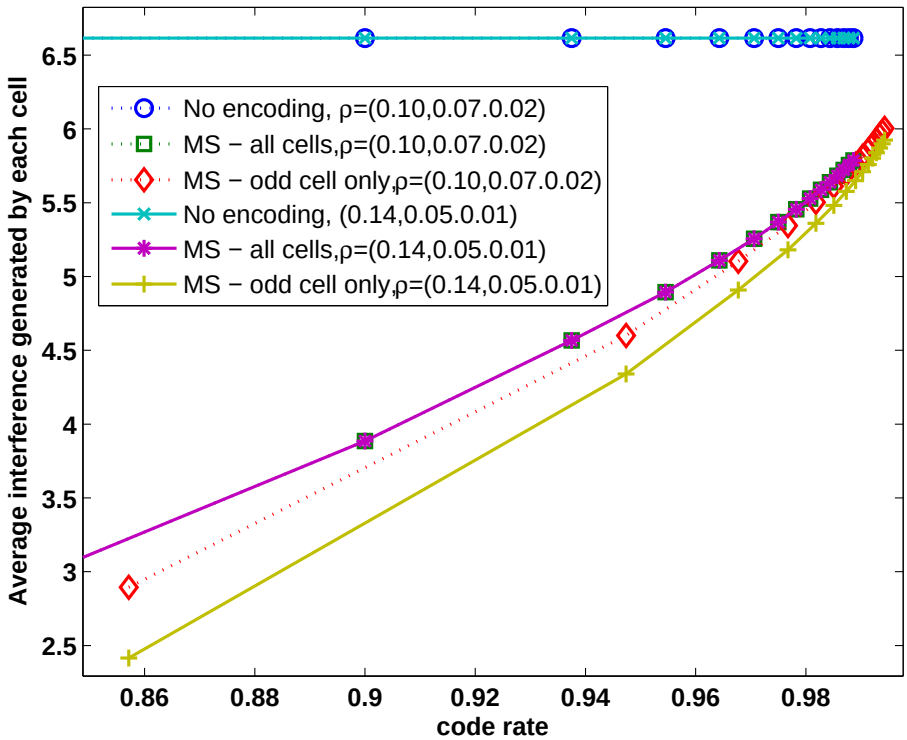

Figure 4.8 C2CI reduction performance of the algorithms with the coupling coefficient

cods at the same code rate. The low energy Huffman coding is optimal with minimum average codeword length for non-uniform symbol frequency. The proposed coding schemes can improve reliability of memory devices by lowering cell-to-cell interference. They can also reduce the power consumption by lowering average magnitude of writing voltage. A key contribution of this chpater is the introduction of new kinds of codes dedicated to reduce cell-to-cell interference in flash memories.

Chapter 5

Conclusions

In this dissertation, we introduce the error correction codes and encoding schemes for reliability of NAND multi-level cell flash memories.

We discussed error correcting codes that are effective for bidirectional and limited-magnitude erorrs and non-binary WOM codes. Key advantages of the bidirectional error correction codes are that it can reduce the parity size, and that it has better error correction performance than the conventional error correction codes. Bidirectional error correction code based on distinct sum set shows the optimal performance based on a minimum magnitude distinct sum set with practical parameters and better performance than that of conventional symmetric limited magnitude error correction codes. A bidirectional double distinct sum set and a code construction are also introduced for double error correction code based on distict sum set. Bidirectional limited magnitude error correction codes based on modulo operation have the practical error correcting capability based on conventional non-binary

error correction codes. Practical issues of encoding and decoding for the proposed method are also discussed, and efficient methods are proposed. We discussed the potential problems of existing error correction codes, and show that proposed bidirectional limited-magnitude code is more suitable to practical flash memory devices in simulations. Two error correcting coding schemes for non-binary WOM codes are also discussed. The proposed codes deal with bidirectional error correction based on distinct sum sets or modulo operation, and parity splitting methods for the WOM code property. The advantages of the proposed methods are that these are practical and systematic codes, and their encoding and decoding processes have low complexity. We also discuss effective asymmetric and symmetric error locating limited-magnitude parity check error correction codes for the MLC flash memory error with lower redundancy.

Furthermore, we propose new code construction methods for interference reduction in flash memory devices. The minimum energy codes are the simplest, but it increases redundancy. The module shift codes show better performance than the minimum energy code at the same code rate. The low energy Huffman coding is optimal with minimum average codeword length for non-uniform symbol frequency. The proposed coding schemes can improve reliability of memory devices by lowering cell-to-cell interference. They can also reduce the power consumption by lowering average magnitude of writing voltage.

A key contribution of these error correction codes and encoding schemes are the introductions of new kinds of efficient coding schemes dedicated to reduce the effects of

interference and improve reliability of memories.

Appendix A

A.1 Performance analysis of MS coding with η=2 case in chap. 4.4.2.

The average encoded magnitude $E(\lambda')$ in (4.17) will be estimated when $\eta = 2$ and n = odd. The module selection parameter is given by $\zeta = \arg\min_{k\in\Xi} \sum_{i=1}^{\eta} i \cdot n_{|i+k-1|_\eta+1}$ in (4.7). We define $\Psi_\zeta = \sum_{i=1}^{\eta} i \cdot n_{|i+\zeta-1|_\eta+1}$. When $\eta = 2$ and $\Xi = \{0, 1\}$, $\Psi_0 = \sum_{i=1}^{2} i \cdot n_{|i+0-1|_2+1} = n_1 + 2n_2$ and $\Psi_1 = \sum_{i=1}^{2} i \cdot n_{|i+1-1|_2+1} = n_2 + 2n_1$. If $\Psi_0 \leq \Psi_1$, ζ becomes 0. Due to $n_1 + 2n_2 \leq n_2 + 2n_1$, $\zeta = 0$ when $n_1 \geq n_2$. On the other hand, if $\Psi_0 > \Psi_1$, that is, $n_1 < n_2$, ζ becomes 1. Due to $n = n_1 + n_2$, when n is an odd number, $0 \leq n_1 \leq \frac{n-1}{2}$ leads to $\zeta = 0$, $\frac{n+1}{2} \leq n_1 \leq n$ results in $\zeta = 1$.

Next, we find the $N_i^\zeta = (n_{1,i}^\zeta, n_{2,i}^\zeta, \cdots, n_{\eta,i}^\zeta)$ according to the criterion in (4.7). $N^\zeta = \{N_1^\zeta, N_2^\zeta, \cdots, N_{\binom{n+\eta-1}{\eta-1}\frac{1}{\eta}}^\zeta\}$ and we get

$N^0 = \{N_1^0, N_2^0, \cdots, N_{\frac{n+1}{2}}^0\} = \{(n, 0), (n-1, 1), \cdots, (\frac{n+1}{2}, \frac{n-1}{2})\}$,

$N^1 = \{N_1^1, N_2^1, \cdots, N_{\frac{n+1}{2}}^1\} = \{(0,n),(1,n-1),\cdots,(\frac{n-1}{2},\frac{n+1}{2})\}$.

We have $n_{1,i}^0 \in \{n, n-1, \cdots, \frac{n+1}{2}\}$, $n_{2,i}^0 \in \{0,1,\cdots,\frac{n-1}{2}\}$, $n_{1,i}^1 \in \{0,1,\cdots,\frac{n-1}{2}\}$ and $n_{2,i}^1 \in \{n, n-1, \cdots, \frac{n+1}{2}\}$.

Our goal is to compute $E(\lambda') = \sum_{j=1}^{\eta} p(\Omega_j)E(\Omega_j)$, and the probability of the j_{th} module element is estimated as $p(\Omega_j) = \frac{1}{n}\sum_{\zeta=0}^{\eta-1} \mathcal{F}_j^{\zeta}$. At first, $\mathcal{F}_j^{\zeta}$ need to be estimated. When ζ is given, an appearance frequency of the jth module element $\mathcal{F}_j^{\zeta}$ in n-length codeword is given by (4.17). Then $\mathcal{F}_1^0$ and $\mathcal{F}_1^1$ are given by

$$\begin{aligned}
\mathcal{F}_1^0 &= \sum_{i=0}^{\frac{n-1}{2}} \binom{n}{n_{1,i}^0}\binom{n-n_{1,i}^0}{n_{2,i}^0}\left(\frac{1}{2}\right)^n n_{1,i}^0 \\
\mathcal{F}_1^1 &= \sum_{i=0}^{\frac{n-1}{2}} \binom{n}{n_{1,i}^1}\binom{n-n_{1,i}^1}{n_{2,i}^1}\left(\frac{1}{2}\right)^n n_{2,i}^1.
\end{aligned}$$

where $n - n_{1,i}^0 = n_{2,i}^0$, $n - n_{1,i}^1 = n_{2,i}^1$, due to $n = \sum_{k=1}^{\eta} n_{k,i}^{\zeta}$.

$$\begin{aligned}
p(\Omega_1) &= \frac{1}{n}\sum_{\zeta=0}^{1} \mathcal{F}_j^{\zeta} = \frac{1}{n}\left(\mathcal{F}_1^0 + \mathcal{F}_1^1\right) \\
&= \frac{1}{n}\left(\frac{1}{2}\right)^n \left(\sum_{i=0}^{\frac{n-i}{2}} \binom{n}{n_{1,i}^0} n_{1,i}^0 + \sum_{i=0}^{\frac{n-1}{2}} \binom{n}{n_{1,i}^1} n_{2,i}^1\right) \\
&= \frac{1}{n}\left(\frac{1}{2}\right)^n \left(\sum_{i=0}^{\frac{n-1}{2}} \binom{n}{n-i}(n-i) + \sum_{i=0}^{\frac{n-1}{2}} \binom{n}{i}(n-i)\right) \\
&= \frac{1}{n}\left(\frac{1}{2}\right)^n \sum_{i=0}^{\frac{n-1}{2}} \left(\binom{n}{n-i}(n-i) + \binom{n}{i}(n-i)\right) \\
&= \frac{2}{n}\left(\frac{1}{2}\right)^n \sum_{i=0}^{\frac{n-1}{2}} \binom{n}{i}(n-i)
\end{aligned}$$

Similarly, we have $\mathcal{F}_2^0 = \sum_{i=0}^{\frac{n-1}{2}} \binom{n}{n_{1,i}^0} (\frac{1}{2})^n n_{2,i}^0$, $\mathcal{F}_2^1 = \sum_{i=0}^{\frac{n-1}{2}} \binom{n}{n_{1,i}^1} (\frac{1}{2})^n n_{1,i}^1$.

$$\begin{aligned} p(\Omega_2) &= \frac{1}{n}\left(\frac{1}{2}\right)^n \left(\sum_{i=0}^{\frac{n-1}{2}} \binom{n}{n_{1,i}^0} n_{2,i}^0 + \sum_{i=0}^{\frac{n-1}{2}} \binom{n}{n_{1,i}^1} n_{1,i}^1 \right) \\ &= \frac{1}{n}\left(\frac{1}{2}\right)^n \sum_{i=0}^{\frac{n-1}{2}} \left(\binom{n}{n-i} i + \binom{n}{i} i \right) \\ &= \frac{2}{n}\left(\frac{1}{2}\right)^n \sum_{i=0}^{\frac{n-1}{2}} \binom{n}{i} i \end{aligned}$$

Finally, an average magnitude of encoded cell $E(\lambda')$ is estimated. $E(\Omega_j) = (j-\frac{1}{2})\frac{q}{\eta} - \frac{1}{2}$ in (4.6), $E(\Omega_1) = \frac{q}{4} - \frac{1}{2} = \frac{q-2}{4}$ and $E(\Omega_2) = \frac{3q}{4} - \frac{1}{2} = \frac{3q-2}{4}$.

$$\begin{aligned} E(\lambda') &= \sum_{j=1}^{\eta} p(\Omega_j) E(\Omega_j) \\ &= p(\Omega_1)E(\Omega_1) + p(\Omega_2)E(\Omega_2) \\ &= \frac{2}{n}\left(\frac{1}{2}\right)^n \sum_{i=0}^{\frac{n-1}{2}} \binom{n}{i} (n-i) E(\Omega_1) \\ &\quad + \frac{2}{n}\left(\frac{1}{2}\right)^n \sum_{i=0}^{\frac{n-1}{2}} \binom{n}{i} i E(\Omega_2) \\ &= \frac{2}{n}\left(\frac{1}{2}\right)^n \sum_{i=0}^{\frac{n-1}{2}} \left(\binom{n}{i} ((n-i)E(\Omega_1) + iE(\Omega_2)) \right) \\ &= \frac{2}{n}\left(\frac{1}{2}\right)^n \sum_{i=0}^{\frac{n-1}{2}} \binom{n}{i} \left(\frac{(q-2)(n-i) + (3q-2)i}{4} \right) \\ &= \frac{2}{n}\left(\frac{1}{2}\right)^n \left(\frac{nq-2n}{4} 2^{n-1} + \frac{q}{2} \sum_{i=0}^{\frac{n-1}{2}} \binom{n}{i} i \right) \\ &= \frac{q-2}{4} + \left(\frac{1}{2}\right)^n \frac{q}{n} \sum_{i=0}^{\frac{n-1}{2}} \binom{n}{i} i \end{aligned}$$

Bibliography

[1] B. Ricco, G. Torelli, M. Lanzoni, A. Manstretta, H. E. Maes, D. Montanari, and A. Modelli, "Nonvolatile multilevel memories for digital applications," *Proceedings of IEEE*, vol. 86, no. 12, pp. 2399-2421, Dec. 1998.

[2] A. Modelli, R. Bez, and A. Visconti, "Multi-level flash memory technology," *in Proc. International Conference on Solid State Devices and Materials*, pp. 516-517, Tokyo, Japan, Sep. 2001.

[3] G. Dong, S. Li, and T. Zhang, "Using data post-compensation and pre-distortion to tolerate cell-to-cell interference in MLC NAND flash memory," *IEEE Transactions on Circuits and Systems I*, vol. 57, no. 10, pp. 2718-2728, Oct. 2010.

[4] P. Cappelletti and A. Modelli, "Flash memory reliability," *Flash Memories*, P. Cappelletti, C. Golla, P. Olivo, and E. Zanoni, Eds. Amsterdam, The Netherlands: Kluwer, pp. 399-441 , 1999.

[5] Jae-Duk Lee, Sung-Hoi Hur, and Jung-Dal Choi, "Effects of floating-gate interference on NAND flash memory cell operation," *IEEE Electron Device Letters*, vol. 23, no. 5, pp. 264-266, May 2002.

[6] W. H. Kim and C. V. Freiman, "Multi-error correcting codes for a binary asymmetric channels," *IRE Transactions on Circuits Theory*, vol. 6, no.5. pp. 71-78, May 1959.

[7] R. Ahlswede, H. Aydinian, L. Khachatrian, and L. Tolhuizen, "On q-ary codes correcting all unidirectional errors of a limited magnitude," *in Proc. International Workshop on Algebraic and Combinatorial Coding Theory*, pp. 20-26, Kranevo, Bulgaria, Jun. 2004.

[8] Y. Cassuto, M. Schwartz, V. Bohossian, and J. Bruck, "Codes for asymmetric limited-magnitude errors with application to multi-level flash memories," *IEEE Transactions on Information Theory*, vol. 56, no. 4, pp. 1582-1595, Apr. 2010.

[9] T. Klove, B. Bose, and N. Elarief, "Systematic single limited magnitude asymmetric error correcting codes," *in Proc. IEEE Information Theory Workshop*, Cairo, Egypt, Jan. 2010.

[10] T. Klove, B. Bose, and N. Elarief, "Systematic, single limited magnitude error correcting codes for flash memories," *IEEE Transactions on Information Theory*, vol. 57, no. 7, pp.4477-4487, Jul. 2011.

[11] 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 Journal of Solid-State Circuits*, vol. 43, no. 4, pp. 919-928, Apr. 2008.

[12] R. Bez, E. Camerlenghi, A. Modelli, and A. Visconti, "Introduction to flash memory," *Proceedings of IEEE*, vol. 91, no.4, pp. 489-502, Apr. 2003.

[13] C. Calligaro, A. Manstretta, A. Modelli, and G. Torelli, "Technological and design constraints for multilevel flash memories," *in Proc. IEEE International Conference on Electronics, Circuits, and Systems*, pp. 1005-1008, Rodos, Greece, Oct. 1996.

[14] Y. Cassuto, M. Schwartz, V. Bohossian, and J. Bruck, "Codes for multi-level flash memories: correcting asymmetric limited-magnitude errors," *in Proc. IEEE International Symposium on Information Theory*, pp. 1176-1180, Nice, France, Jun. 2007.

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

[16] N. Elarief and B. Bose, "Optimal, systematic, q-ary codes correcting all asymmetric and symmetric errors of limited magnitude," *IEEE Transactions on Information Theory*, vol. 56, no. 3, pp. 979-983, Mar. 2010.

[17] H. Zhou, A. Jiang and J. Bruck, "Nonuniform Codes for Correcting Asymmetric Errors in Data Storage", *IEEE Transactions on Information Theory*, vol. 59, no. 5, pp. 2988-3002, May 2013.

[18] M. Jeon, K. Kim, B. Shin and J. Lee, "Interference Compensation Technique for Multilevel Flash Memory", *in Proc. IEEE Midwest Symposium on Circuits and Systems*, Seoul, Korea, Aug. 2011.

[19] K. Takeuchi, T. Tanaka, and H. Nakamura, "A double-level-Vth select gate array architecture for multilevel NAND flash memories," *IEEE Journal of Solid-State Circuits*,

vol. 31, no. 4, pp. 602–609, Apr. 1996.

[20] M. Jeon, K. Kim, S. Chung, S. Chung, B. Shin, and J. Lee, "Adaptive interference mitigation for multilevel flash memory devices", *IEICE Transactions Fundamentals of Electronics, Communications and Computer Sciences*, vol. E94-A, no. 11, pp.2453-2457, Nov. 2011.

[21] T.M. Cover and J.A. Thomas, *Elements of Information Theory, 2nd Edition* John Wiley & Sons, Hoboken, NJ, USA, 2006.

[22] M. Jeon and J. Lee, "On codes correcting bidirectional limited-magnitude errors for flash memories", *in Proc. International Symposium on Information Theory and its Applications*, pp.96-100, Honolulu, USA, Oct. 2012.

[23] C. Erin, H. H. Asada, and K.-Y. Siu. "Minimum Energy Coding for RF Transmission." *in Proc. IEEE Wireless Communications and Networking Conference*, vol. 2, pp. 621-625, New Orleans, USA, 1999.

[24] Y. Prakash and S. K. S. Gupta, "Energy Efficient Source Coding and Modulation for Wireless Applications," *in Proc. IEEE Wireless Communications and Networking Conference*, vol. 1, pp. 212–217, New Orleans, USA, 1999

[25] J. Kim and J. G. Andrews, "An energy efficient source coding and modulation scheme for wireless sensor networks," *in Proc. IEEE Workshop on Signal Processing Advances in Wireless Communications*, pp. 710-714, New York, USA, Jun, 2005

[26] S. Gregori, A. Cabrini, O. Khouri, and G. Torelli, "On-chip error correcting techniques

for new-generation Flash memories," *Proceedings of the IEEE*, vol. 91, no. 4, pp. 602-616, 2003.

[27] E. Yaakobi, J. Ma, L. Grupp, P. H. Siegel, S. Swanson, and J. K. Wolf, "Error Characterization and Coding Schemes for Flash Memories," *in Proc. IEEE GLOBECOM Workshop on Application of Communication Theory to Emerging Memory Technologies*, pp. 1856-1860, Miami, USA, Dec. 2010.

[28] B.G. Bajoga and W.J.Walbesser, "Decoder complexity for BCH codes," *Proceedings of IEE*, vol. 120, no. 4, pp. 429-431, Apr. 1973.

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

[30] A. Jiang, V. Bohossian, and J. Bruck, "Floating codes for joint information storage in write asymmetric memories," *in Proc. IEEE International Symposium on Information Theory*, pp. 1166-1170, Nice, France, Jun. 2007.

[31] A. Jiang, R. Mateescu, M. Schwartz, and J. Bruck,"Rank modulation for flash memories," *IEEE Transactions on Information Theory*, vol. 55, no. 6, pp. 2659-2673, Jun. 2009.

[32] A. Jiang, M. Schwartz and J. Bruck, "Error-Correcting Codes for Rank Modulation," *in Proc. IEEE International Symposium on Information Theory*, pp. 1736-1740, Toronto, Canada, Jul. 2008.

[33] F. Fu and A. J. Han Vinck, "On the capacity of generalized writeonce memory with

state transitions described by an arbitrary directed acyclic graph," *IEEE Transactions on Information Theory*, vol. 45, no. 1, pp. 308–313, Sep. 1999.

[34] R. Gabrys, E. Yaakobi, L. Dolecek, P. H. Siegel, A. Vardy, J. K. Wolf, "Non-binary WOM-Codes for Multilevel Flash Memories", *in Proc. IEEE Information Theory Workshop*, pp. 40-44, Paraty, Brazil, Oct. 2011.

[35] M. Jeon, and J. Lee "Bidirectional Limited-Magnitude Error Correction Codes for Flash Memories," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E96-A, no. 7, pp. 1602-1608, Jul. 2013.

[36] E. Yaakobi, P. H. Siegel, A. Vardy, and J. K. Wolf, "Multiple error-correcting WOM codes", *in Proc. IEEE International Symposium on Information Theory*, Austin, USA, May 2010.

[37] G. Z´emor and G.D. Cohen, "Error-correcting WOM-codes," *IEEE Transactions on Information Theory* vol. 37, no. 3, pp. 730–734, May 1991.

[38] A. Jiang, "On the generalization of error correcting WOM codes," *in Proc. IEEE International Symposium on Information Theory*, pp. 1391–1395, Nice, France, Jun. 2007.

[39] E. Yaakobi, S, Kayser, P. H. Siegel, A. Vardy, and J. K. Wolf, "Efficient two-write WOM-codes," *in Proc. IEEE Information Theory Workshop*, Dublin, Ireland, Aug. 2010.

[40] Q. Huang, S. Lin and K . A. S. Abdel-Ghaffar, "Error-Correcting Codes for Flash Coding," *IEEE Transactions on Information Theory*, vol. 57, no. 9, pp. 6097–6108,

Sep. 2011.

[41] M. Jeon, S. Chung, B. Shin and J. Lee, "Limited Magnitude Error Locating Parity Check Codes for Flash Memories," *in Proc. IEEE Midwest Symposium on Circuits and Systems*, pp. 29-32, Boise, USA, Aug. 2012.

[42] Y. Cassuto, and E. Yaakobi, "Short Q-ary WOM Codes with Hot/Cold Write Differentiation," *in Proc. IEEE International Symposium on Information Theory*, pp. 1391-1395, Cambridge, USA, Jul. 2012.

한글초록

최근 비휘발성 메모리는 무게, 휴대성과 속도 등 여러 장점으로 플래시 메모리 저장장치, SSD (solid state disk)등에 점차 널리 사용되고 있으며, 최근에는 용량 확장을 위해 한 셀에 여러 bit 의 정보를 저장하는 multi-level cell (MLC) 플래시 메모리가 사용되고 있다. 하지만 셀 당 저장되는 정보가 늘어날수록 셀 간 간섭(cell to cell interference) 및 오버 프로그래밍, 잦은 쓰기/삭제 반복으로 인한 retention 문제등 여러 요소로 인한 간섭에 의해 오류가 급격히 늘어나는 신뢰성 문제가 발생한다. 이를 해결하기 위하여, 신호 처리적 방법, 오류 정정 부호, 데이터의 부호화 저장 방식 알고리즘등의 시도가 제안되어 왔다. 본 논문에서는 이러한 비휘발성 메모리의 신뢰성 문제 향상을 위한 오류 정정 부호와 데이터 부호화 기록 방식을 제안한다.

먼저 플래시 메모리의 오류 특성을 이용한 효율적인 오류 정정 부호를 제안한다. 플래시 메모리의 오류는 간섭으로 인한 이웃 레벨로의 문턱 전압 변화에 기인하므로 크기가 제한 되어있는 특성이 있다. 주요 간섭인 셀 간 간섭의 크기는 이웃 셀들의 문턱 전압 변화량에 비례하는 일방향적 특성이 있지만, 동시에 retention 문제와 최적 읽기 전압 재설정 등으로 인하여 하강방면의 오류도 발생하므로 양방향 비대칭 오류(bidirectional error) 에 대한 고려가 필요하다. 양방향 비대칭 오류 정정 부호는 상승,하강 방향으로 각각 l_u, l_d 의 크기를 가지는 오류를 정정하며, 두 방향의 크기는 다를수 있다. 이러한 양방향 비대칭 크기 제한 오류를 정정 하기 위하여 먼저 distinct

sum set에 기반한 오류 정정 부호를 제안한다. 제안된 방식에 의해 생성된 정수 집합에 기반하여 동일 부호율시 전통적인 오류 정정 부호보다 더 좋은 오류 정정 성능을 보이면서도 부호화, 복호화 복잡도가 낮은 코드를 생성한다. 그리고 데이터에 modulo operation을 이용하여 또 다른 효율적인 양방향 비대칭 크기제한 오류 정정 부호를 제안한다. 이 방식은 기존의 전통적인 오류 정정 부호를 기본 코드로 사용하면서도 성능 측면에서 효율적인 코드를 생성할 수 있고 오류 정정 능력의 설정 등의 측면에서 실용적이다. 또한 제안된 양방향 비대칭 오류 정정 부호들을 WOM (write once memory) 부호를 위한 오류 정정 부호를 만드는데 적용한다. WOM 부호는 플래시 메모리의 삭제 횟수 감소를 통해 retention 문제 해결에 도움이 되는 코드로 널리 연구되어 왔다. 하지만 오류정정 부호를 사용하고자 할때 전통적인 오류 정정 부호를 직접적으로 사용하기 어렵고, 기존에 제안된 WOM 부호를 위한 오류 정정 부호의 경우 실용적인 systematic한 부호를 얻기 어려웠으므로, systematic하고 효율적인 부호화 방식을 제안한다. 또한 오류 위치 기반 크기 제한 일방향, 양방향 패리티 검사 부호도 소개하며, 낮은 복잡도를 가진다.

다음으로, 셀 간 간섭의 영향을 줄이기 위한 데이터 부호화 기록 방식을 제안한다. 셀 간 간섭은 이웃셀의 문턱전압 변화량에 비례하고, 셀별로 기록되는 심볼의 크기와 셀이 가지는 문턱전압이 비례한다고 가정하면, 높은 심볼 크기 값을 가지면 주변에 많은 간섭을 일으킨다고 할 수 있다. 따라서 각 셀에 저장되는 심볼의 크기 값을 최소화 하면 셀 간 간섭의 양이 줄어든다. 이를 위해 정보를 기록할 때 최소한의 데이터 심볼 크기를 갖는 코드로 변환하여 기록하는 module shift (MS) coding 방식을 제안한다.

제안하는 방식은 q-ary의 심볼 메시지를 다루며 고정 길이 부호(fixed length code)를 사용한다. 저장할 메시지를 여러 개의 블록으로 나누고, 최적의 값과 modulo addition을 통해 평균적인 메시지 심볼 크기가 최소화 되도록 한다. 또한 데이터 압축을 위해서 소스 코딩의 일종인 허프만(Huffman) 코드를 사용할 때 최소한의 데이터 심볼 크기를 갖는 코드로 변환하여 기록하는 low energy Huffman (LE-H) coding 방식을 제안한다. 제안하는 방식은 길이가 고정되지 않은 코드(variable-length code) 를 다루며, 부호화 과정중 트리 구조에서 심볼 할당 방식을 통해 최소한의 평균길이(minimum average codeword length)를 가질 때 최소한의 메시지 심볼 크기를 가지는 코드를 생성할 수 있다.

주요어: 다중레벨 셀 플래시 메모리, 오류 정정 부호, 셀 간 간섭, WOM 부호, 양방향 비대칭 오류

학번: 2009-30210