Writing on Dirty Memory

Submitted in partial fulfillment of the requirements for

the degree of

Doctor of Philosophy

in

Electrical and Computer Engineering

Yongjune Kim

B.S., Electrical and Computer Engineering, Seoul National University
M.S., Electrical and Computer Engineering, Seoul National University

Carnegie Mellon University
Pittsburgh, PA

July, 2016

Keywords: Memory, Nonvolatile memory, Coding, Side information, Flash memory, Resistive memory

For my wife
Yuri

Abstract

Non-volatile memories (NVM) including flash memories and resistive memories have attracted significant interest as data storage media. Flash memories are widely employed in mobile devices and solid-state drives (SSD). Resistive memories are promising as storage class memory and embedded memory applications.

Data reliability is the fundamental requirement of NVM as data storage media. However, modern nano-scale NVM suffers from challenges of inter-cell interference (ICI), charge leakage, and write endurance, which threaten the reliability of stored data. In order to cope with these adverse effects, advanced coding techniques including soft decision decoding have been investigated actively.

However, current coding techniques do not capture the physical properties of NVM well, so the improvement of data reliability is limited. Although soft decision decoding improves the data reliability by using soft decision values, it degrades read speed performance due to multiple read operations needed to obtain soft decision values.

In this dissertation, we explore coding schemes that use side information corresponding to the physical phenomena to improve the data reliability significantly. The side information is obtained before writing data into memory and incorporated during the encoding stage. Hence, the proposed coding schemes maintain the read speed whereas the write speed performance would be degraded. It is a big advantage from the perspective of speed performance since the read speed is more critical than the write speed in many memory applications.

First, this dissertation investigates the coding techniques for memory with stuck-at defects. The idea of coding techniques for memory with stuck-at defects is employed to handle critical problems of flash memories and resistive memories. For 2D planar flash memories, we propose a coding scheme that combats the ICI, which is a primary challenge of 2D planar flash memories. Also, we propose a coding scheme that reduces the effect of fast detrapping, a degradation factor in 3D vertical flash memories. Finally, we investigate the coding techniques that improve write endurance and power consumption of resistive memories.

Acknowledgments

I am indebted to all those who have helped me finish this dissertation: my advisor, commitee members, collagues, and family.

First of all, I would like to express my deepest gratitude to my advisor B. V. K. Vijaya Kumar (Vijayakumar Bhagavatula). Words are not enough to thank him for his invaluable advice, brilliant insight, and constant encouragement. I was very fortunate to be advised by him. I would also like to thank my thesis committee members Rohit Negi, Pulkit Grover, and Robert Mateescu to their advice and comments for my research.

During my graduate studies, I was fortunate to collaborate with several wonderful researchers: Abhishek A. Sharma, James A. Bain, Xin Li, Euiseok Hwang, Robert Mateescu, Seung-Hwan Song, and Zvonimir Bandic. I learned a lot from them and enjoyed fruitful collaborations.

It has been a great joy to be a member of our research group: Stephen Siena, Zhiding Yu, Eric He, Rick Chang, Andrew Fox, Jon Smereka, Kathy Brigham, Hyunggi Cho, Joseph Fernandez, Can Ye, and Vishnu N. Boddeti. I would like to thank my colleagues at Carnegie Mellon University: Andrew Cheng, Haewon Jeong, Yaoqing Yang, and Manzil Zaheer. I would also like to thank Marilyn Patete and Kara Knickerbocker for their warm support and help.

I would like to thank the Storage Architecture Group at HGST Research. During my internship, I had a great time with friendly members of the Storage Architecture Group. Especially, I would like to thank Robert Mateescu for mentoring me and becoming my committee member.

I would like to thank my parents and parents-in-law for their everlasting love. I would like to thank my sons Hyunsuh and Yoonsuh for brightening my days. Finally, I am sincerely grateful to my love and best friend Yuri for giving so much meaning to my life.

My research was generously supported by the Data Storage Systems Center (DSSC) at Carnegie Mellon University.

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Dissertation Overview . . . 6
- 1.3 Notation . . . 10

2 Coding for Memory with Stuck-at Defects **13**
- 2.1 Introduction . . . 13
- 2.2 Background for Coding for Defect Channel Model . . . 17
  - 2.2.1 Defect Channel Model . . . 17
  - 2.2.2 Capacity of Binary Memory with Stuck-at Defects . . . 20
  - 2.2.3 Additive Encoding . . . 22
  - 2.2.4 Partitioned Linear Block Codes (PLBC) . . . 24
- 2.3 Redundancy Allocation of Finite-Length PLBC . . . 28
  - 2.3.1 Upper Bound on Encoding Failure Probability . . . 29
  - 2.3.2 Redundancy Allocation: BDEC . . . 33
  - 2.3.3 Redundancy Allocation: BDSC . . . 38
- 2.4 Relations between BEC, BDC, BEQ, and WOM . . . 43
  - 2.4.1 Duality between BEC and BDC . . . 43
  - 2.4.2 Relations between BEC, BDC, BEQ, and WOM . . . 49
- 2.5 Conclusion . . . 52
- 2.6 Proof of Proposition 2.4 . . . 52
- 2.7 Proof of Proposition 2.5 . . . 53
- 2.8 Proof of Lemma 2.8 . . . 54
- 2.9 Proof of Lemma 2.9 . . . 55
- 2.10 Proof of Theorem 2.15 . . . 57
- 2.11 Proof of Corollary 2.16 . . . 57
- 2.12 Proof of Theorem 2.18 and Corollary 2.19 . . . 59

3 Coding for Flash Memory **61**
- 3.1 Introduction . . . 61
- 3.2 2D Planar and 3D Vertical Flash Memories . . . 64
  - 3.2.1 Basic Operations and Asymmetry between Write and Erase Operations . . . 64
  - 3.2.2 2D Planar Flash Memories: ICI . . . 66
  - 3.2.3 3D Vertical Flash Memories: ICI and Fast Detrapping . . . 68

3.3 Combating Inevitable Interference for 2D Planar Flash Memory 70
3.3.1 Channel Model of 2D Planar Flash Memory 71
3.3.2 Proposed Scheme for 2D Planar Flash Memories: Combating ICI by Coding 73
3.3.3 Dirty Paper vs. Dirty Flash Memory 78
3.3.4 Simulation Results 78
3.4 Harnessing Intentional Interference for 3D Vertical Flash Memory 84
3.4.1 Channel Model of 3D Vertical Flash Memory 84
3.4.2 Proposed Scheme for 3D Vertical Flash Memory: Harnessing ICI by Coding 85
3.4.3 Simulation Results 89
3.5 Conclusion 91

4 Coding for Resistive Memory **93**
4.1 Introduction 93
4.2 Basics of Resistive Memories 95
4.2.1 Phase Change Memories (PCM) 96
4.2.2 Resistive Random-Access Memories (RRAM) 97
4.3 Writing Cost 99
4.3.1 Power Consumption and Endurance in RRAM Devices 99
4.3.2 Writing Cost: Rewriting Cost and Initial Writing Cost 101
4.4 Rewriting Locality and Locally Rewritable Codes 103
4.4.1 Motivation and a Toy Example 103
4.4.2 Rewriting Locality and Locally Rewritable Codes 105
4.4.3 Upper Bounds on Rewriting Cost and Initial Writing Cost 108
4.5 Constructions of Locally Rewritable Codes 110
4.5.1 Locally Repairable Codes 110
4.5.2 Construction of Locally Rewritable Codes 111
4.5.3 Locally Rewritable Codes with Error Correcting Capability 113
4.6 Conclusion 116
4.7 Proof of Theorem 4.17 116
4.8 Proof of Corollary 4.18 117
4.9 Proof of Theorem 4.19 118
4.10 Proof of Corollary 4.20 120

5 Conclusion **123**
5.1 Thesis Contributions 123
5.2 Future Work 125

Bibliography **127**

List of Figures

1.1 Channel model with side information. A message M is encoded to a codeword $X^n = (X_1, \ldots, X_n)$. The received word $Y^n = (Y_1, \ldots, Y_n)$ is decoded to $\widehat{M}$ (i.e., the estimate of message). $S^n = (S_1, \ldots, S_n)$ represents the side information known to the encoder or the decoder. 3

2.1 Binary defect channel (BDC). 14
2.2 Capacity region of the BDEC derived in Proposition 2.14. Two points of Q_1 and Q_2 in the capacity region represent the pairs of code rates $\left(R_1 = \frac{k}{n}, R_0 = \frac{l}{n}\right)$. Note that the capacity $C_{\text{BDEC}}^{\text{enc}}$ is the supremum of $R_1 = R$. 28
2.3 Comparison of simulation results, upper bounds by (2.41), and calculated values by (2.40) for $P(E = 0 \mid U = u)$. $[n = 31, k, l]$ PBCH codes are used. The code rate is $R = k/n$. 31
2.4 Comparison of simulation results and upper bounds in (2.44) for the probability of encoding failure $P(E = 0)$. We used PBCH codes for the BDC with probability of defect $\beta = 0.1$. 33
2.5 Comparison of simulation results (simul.) and upper bounds (UB) of the probability of recovery failure $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ of BDEC channels in Table 2.2. 38
2.6 Comparison of simulation results and estimates of the probability of recovery failure $P\left(\widehat{\mathbf{m}} \neq \mathbf{m}\right)$. 42
2.7 Probability of failure, i.e., $P(D = 0)$ of the BEC with $\alpha = 0.1$ and $P(E = 0)$ of the BDC with $\beta = 0.1$. 48
2.8 Relations between BEC, BDC, BEQ, and WOM. 49

3.1 2D planar flash memory block where SSL, GSL, and CSL denote string select-line, ground select-line, and common source-line, respectively. 65
3.2 Threshold voltage distribution of flash memory cells. 65
3.3 Inter-cell interference (ICI) between adjacent cells of 2D planar flash memory. . . 67
3.4 3D vertical flash memory cell array in [1]. 68
3.5 Change from the 2D planar flash memory channel with the ICI to the model of memory with defective cells by one pre-read operation. 74
3.6 Vulnerable cells can also be regarded as stuck-at 0 defects by setting a pre-read level such that $\eta_{\text{pre}} < \eta$. 75
3.7 Extension to MLC flash memories. 76
3.8 The improvement of threshold voltage distribution by proposed scheme (SLC, $n = 1023, k = 923, R = 0.90, \alpha = 1.2, \sigma_{Z_{\text{read}}} = 0.25, \eta_{\text{pre}} = -1.4$). 80

3.9 The improvement of raw BER and probability of decoding failure by the proposed scheme (SLC, $n = 1023, k = 923, R = 0.90$, $\alpha = 1.2, \sigma_{Z_{\text{read}}} = 0.25$). If $l = 0$, then the side information is ignored, which is equivalent to the BCH code. 81
3.10 Comparison of $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ (SLC, $n = 1023, k = 923, R = 0.90$). 82
3.11 Comparison of $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ for BCH codes, proposed scheme, and LDPC codes (SLC, $R = 0.89$, $\alpha = 1.4$). 83
3.12 The improvement of threshold voltage distribution and $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ (MLC, $n = 1023, k = 923, R = 0.90$). 85
3.13 Fast detrapping and identifying cells that suffer from fast detrapping. 86
3.14 Intentional ICI for compensating fast detrapping. The cell $C_{(i+1,j,k)}$ will be regarded as stuck-at 0 ("S_1") defect for the intentional ICI. 87
3.15 The improvement of threshold voltage distribution and $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ (SLC, $n = 1023, k = 923, R = 0.90$, $\sigma_{Z_{\text{fast}}} = 0.4$, $\sigma_{Z_{\text{random}}} = 0.2$). 90
3.16 Comparison of $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ for different $\sigma_{Z_{\text{fast}}}$ of fast detrapping and $\sigma_{Z_{\text{random}}}$ (SLC, $n = 1023, k = 923, R = 0.90$). 91

4.1 Cell distribution of resistive memories. Note that $x \in Q = \{0, 1, \ldots, q-1\}$ where $q = 2^b$ represents the channel input of the given channel model. 95
4.2 Principle of PCM. Starting from the amorphous phase with large resistance, a current pulse is applied. After sufficiently long pulse heats the material above the minimum crystallization temperature T_{x} to crystallize the material, the resistance is low (set operation). After the larger and short pulse is applied to heat the material above the melting temperature T_{m}, the material is melt-quenched and returns to the amorphous (reset operation). Different colors represent different atoms (such as Ge, Sb, and Te in the commonly used GeSbTe compounds) in the phase change materials [2]. 97
4.3 Direct current–voltage characteristics of the RRAM device showing forming and switching processes and a physical mechanism of filament formation and dissolution. 98
4.4 Relation between resistance change and write power consumption. The RRAM cells were 200 nm TiN-TaO_x-TiN stack and show reliable resistive switching behavior. These cells were fabricated at Carnegie Mellon University with the same material as in [3]. 99
4.5 Write endurance characteristics of 85 nm RRAM cells. The raw data of these experimental results came from IMEC. 100

List of Tables

2.1 All Possible Redundancy Allocation Candidates of $[n = 1023, k = 923, l]$ PBCH Codes 35
2.2 BDEC with the Same $C_{\text{BDEC}} = 0.96$ 37
2.3 Optimal Redundancy Allocations (l^*, r^*) and Their Estimates $(\widehat{l}, \widehat{r})$ and $(\widetilde{l}, \widetilde{r})$ of BDEC 39
2.4 Several Channel Parameters of the BDSC 42
2.5 Optimal Redundancy Allocations (l^*, r^*) and Their Estimates $(\widehat{l}, \widehat{r})$ of BDSC by (2.63) 43
2.6 Duality between BEC and BDC 46

3.1 Simulation Parameters of 2D Planar Flash Memory 79

4.1 Duality of LRC and LWC 113

Chapter 1

Introduction

1.1 Motivation

Non-volatile memory (NVM) is a type of computer memory that has the capability to retain stored data even when the power is turned off. NVM is typically used for secondary storage or persistent storage. It includes flash memories and emerging memories such as phase change memories (PCM), resistive random access memories (RRAM or ReRAM), ferroelectric RAM (FRAM or FeRAM), magnetic RAM (MRAM), and spin transfer torque RAM (STTRAM).

Flash memory is the most widely employed NVM technology. Due to the rapid growth of mobile devices and solid-state drives (SSD), flash memory market continues to expand at a staggering pace. Obviously, there is a strong demand for higher density flash memories, which has been achieved by scaling down (i.e., using higher resolution techniques such as 1x nm photo-lithography) and multi-level cell (MLC) which represent more than one bit of information in each memory cell. Also, 3D vertical flash memories were proposed to increase flash memory density by stacking up cells vertically.

However, these approaches lead to physical phenomena that degrade the reliability of stored data. Aggressive scaling down of cell size has driven the continuous growth of 2-dimensional (2D) planar flash memory density. However, the scaling down leads to many challenges such as

increased *inter-cell interference (ICI)* and photo-lithography limitation [4, 5]. As the distance between adjacent cells decreases due to scaling down, flash memory cells suffer from higher ICI which causes data retrieval errors, unless methods are put in place to handle ICI [4, 6]. Hence, the ICI is a major challenge for data reliability of high density 2D planar flash memory.

3D vertical flash memories overcome scaling down challenges by stacking up cells in the vertical direction instead of shrinking cells within a 2D plane [7, 8]. Recent 3D vertical flash memory research shows better device characteristics compared to 2D 1x nm planar flash memory [7]. However, 3D vertical flash memory has a problem of *fast detrapping*, which is a rapid charge loss phenomenon resulting in larger threshold voltage variations in programmed cells [7, 9]. Moreover, MLC reduces the noise margin, which makes the stored data more vulnerable to these adverse phenomena because more states should be crammed within the given constrained threshold voltage window [10].

Among emerging NVM technologies, resistive memories including PCM and RRAM have attracted significant research interest. The resistive memories are NVMs that store data by modulating the resistance of each memory cell. The advantages of resistive memories are scalability, fast speed, and rewritability [2, 11, 12, 13, 14]. First, their potential scalability to the nanometer regime is one of the most important advantages of resistive memories over dynamic RAM (DRAM) and flash memories [15, 16]. Unlike DRAM, resistive memories are nonvolatile memories, so refresh operations as in DRAM are unnecessary. Resistive memories can realize the MLC and be stacked in 3D with a compact cross-point architecture for higher density. Moreover, resistive memory technologies are expected to offer better speed performance than flash memories [12, 14]. The resistive memories also allow rewriting the data without erase operation, which is an advantage over flash memories where a number of cells have to be erased to rewrite a single memory cell.

However, resistive memories have challenges of *write endurance* and *write power consumption*. The write endurance refers to the fact that the repeated writes cause resistive memory cells to become unreliable. The write endurance is a critical characteristic because higher en-

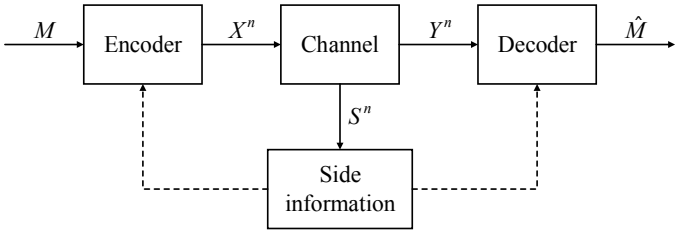


Figure 1.1: Channel model with side information. A message M is encoded to a codeword $X^n = (X_1, \ldots, X_n)$. The received word $Y^n = (Y_1, \ldots, Y_n)$ is decoded to $\widehat{M}$ (i.e., the estimate of message). $S^n = (S_1, \ldots, S_n)$ represents the side information known to the encoder or the decoder.

durance will allow resistive memories to be used in applications where frequent write operations are required [12]. Although the write endurance of resistive memories is better than that of flash memories, their write endurance is worse than that of DRAM. In order to open up many potential applications such as embedded memory, the write endurance needs to be improved substantially [12, 14].

The write operation of resistive memories requires higher power consumption than the read operation in order to change the physical states of memory cells. Especially, the increase of resistance of PCM cell needs substantial power [12, 13]. This inherent write power consumption problem is an important hurdle to increasing write bandwidth because the high write power precludes the writing of many bits in parallel [13].

In order to cope with these adverse effects, traditional approaches have focused on device, circuit, and architecture levels. For example, device level approaches explore new materials and cell structures to improve device characteristics [5, 9]. At circuit and architecture levels, novel operation schemes and architectures such as all bit-line (ABL) were proposed to reduce the adverse effects [7, 17, 18, 19, 20].

Along with these traditional approaches, advanced error control coding (ECC) and signal processing techniques have been investigated [21, 22, 23, 24, 25]. In addition, modulation coding schemes for NVM have been investigated to remove some data patterns which are more vulnerable to adverse phenomena [26, 27, 28, 29, 30].

In this dissertation, we investigate coding schemes that use side information corresponding

to adverse phenomena in NVMs. Our work falls into the topic of *channel coding with side information* at the transmitter (encoder) or receiver (decoder) as shown in Fig. 1.1 [31, 32, 33, 34]. It is well known that the capacity of the channel can be improved, which allows better coding schemes by exploiting the side information at the encoder or decoder.

However, using the side information is not free. Because of the extra steps of obtaining the side information from NVM cells and incorporating this side information into encoding or decoding, the speed performance would be degraded. The write speed would be degraded if the encoder uses the side information. On the other hand, the read speed would be lower if the decoder uses the side information.

In many memory systems, the read speed performance is more critical than the write speed performance [35]. The reason is that the write operation is typically not on the critical path because of write buffers, thus the write latency can be hidden. Also, the read operations are required more often than the write operations in many memory applications. Thus, we focus on the coding schemes using side information at the encoder, which is called *Gelfand-Pinsker problem*.

There are two famous examples in Gelfand-Pinsker problem: *Writing on dirty paper (dirty paper coding)* in [36] and *coding in a memory with defective cells* in [37].

Costa's writing on dirty paper considers the following channel [36]:

$$Y = X + S_{\mathrm{I}} + Z \tag{1.1}$$

where X and Y are the channel input and output, respectively. Also, $S_{\mathrm{I}} \sim N\left(0, \sigma_{S_{\mathrm{I}}}^2\right)$ represents interference and $Z \sim N\left(0, \sigma_Z^2\right)$ denotes additive noise. Also, S_{I} and Z are independent. Assume that the channel input satisfies an average power constraint $\frac{1}{n}\sum_{i=1}^{n} X_i^2 \leq P$ for the channel input vector $X^n = (X_1, \cdots, X_n)$. If neither the encoder nor the decoder knows S_{I}, the capacity is given by [36]

$$C_{\min}^{I} = \frac{1}{2}\log_2\left(1 + \frac{P}{\sigma_{S_{\mathrm{I}}}^2 + \sigma_Z^2}\right). \tag{1.2}$$

If the encoder knows the entire interference vector $S_{\mathrm{I}}^n = (S_{\mathrm{I},1}, \cdots, S_{\mathrm{I},n})$ prior to transmission, the capacity is given by

$$C_{\max}^I = \frac{1}{2}\log_2\left(1 + \frac{P}{\sigma_Z^2}\right) \tag{1.3}$$

where the effect of the interference S_{I} is completely canceled out [36].

The channel model of memory with defective cells (i.e., stuck-at defects) was introduced by Kuznetsov and Tsybakov in 1974 [37]. This channel model has a channel state $S \in \{0, 1, \lambda\}$, which is called defect information. The state $S = 0$ corresponds to a stuck-at 0 defect that always outputs a 0 independent of its input value, the state $S = 1$ corresponds to a stuck-at 1 defect that always outputs a 1, and the state $S = \lambda$ corresponds to a normal cell that outputs the same value as its input. The probabilities of 0, 1, λ states are $\beta/2$, $\beta/2$ (assuming a symmetric defect probability), and $1-\beta$, respectively [34, 37]. We call this channel model the binary defect channel (BDC).

The capacity of the BDC is $1 - \beta$ when both the encoder and the decoder know the defect information. If the decoder is aware of the defect locations in an array of memory cells, then the defects can be treated as erasures so that the capacity is $1 - \beta$ [34, 38]. On the other hand, Kuznetsov and Tsybakov investigated the model where the encoder knows the channel state information (i.e., locations and stuck-at values of defects) and the decoder does not have any information of defects [37]. It was shown that the capacity is also $1 - \beta$ even when only the encoder knows the defect information [37, 38]. Thus, the capacity of the BDC is given by

$$C_{\mathrm{BDC}} = 1 - \beta. \tag{1.4}$$

The common aspect of *writing on dirty paper* and *memory with stuck-at defects* is that the maximum capacities can be achieved if the encoder knows the side information corresponding to the channel state. Note that the side information of writing on dirty paper corresponds to interference and the side inforamtion of coding for memory with stuck-at defects is the defect

information.

In this dissertation, we focus on the model of memory with stuck-at defects because this model can be an appropriate channel model for flash memories and resistive memories. For flash memories, we will show that the problems of combating and harnessing the ICI can be cast as coding for memory with stuck-at defects. The memory with stuck-at defects is also a proper model for resistive memories because a heavily cycled (i.e., rewritten) resistive memory cells can be regarded as stuck-at defects. Based on this model, we investigate the coding techniques that handle physical phenomena of flash memories and resistive memories.

1.2 Dissertation Overview

In Chapter 2, we study coding schemes for the channel model of memory with stuck-at defects in [37]. Afterwards, we formulate the redundancy allocation problem for memory suffering from permanent stuck-at defects and transient errors (erasures or random errors). The coding schemes for memory with permanent defects as well as transient errors have two parts of redundancy: 1) redundancy to deal with permanent defects and 2) redundancy for transient errors. Hence, we investigate the optimum way to allocate redundancy. The objective is to find an optimal allocation between these two parts of redundancy in order to minimize the probability of decoding failure. For an efficient method of finding optimal redundancy allocation, we derive and estimate and an upper bound of decoding failure.

Next, we study the relation between binary memory with stuck-at defects (i.e., binary defect channel (BDC)), binary erasure channel (BEC), binary erasure quantization (BEQ), and write-once memory (WOM). We point out the duality between BDC and BEC, which can be connected to the relation between locally repairable codes (LRC) in distributed storage systems and our proposed locally rewritable codes (LWC) for resistive memories in Chapter 4.

Chapter 2 contains a summary of the material in the following papers:

- Yongjune Kim and B. V. K. Vijaya Kumar, “Coding for memory with stuck-at defects,” in

Proc. IEEE International Conference on Communications (ICC), Jun. 2013.

- Yongjune Kim and B. V. K. Vijaya Kumar, "Redundancy allocation of partitioned linear block codes," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, Jul. 2013.
- Yongjune Kim and B. V. K. Vijaya Kumar, "Duality between erasures and defects," in *Proc. Information Theory and Applications (ITA) Workshop*, Feb. 2016.
- Yongjune Kim and B. V. K. Vijaya Kumar, "Redundancy allocation in finite-length partitioned linear block codes for nonvolatile memories," *submitted.*

The contributions of Chapter 2 to the study of coding techniques for memory with stuck-at defects are as follows:

- Formulating the redundancy allocation problem for memory suffering from permanent stuck-at defects and transient errors.
- Deriving the upper bound on the probability of decoding failure and proposing techniques to determine the optimal redundancy allocation based on this upper bound.
- Investigating the relations between binary erasure channel (BEC), memory with stuck-at defects, binary erasure quantization (BEQ), and write-once memory (WOM).

In Chapter 3, we explore coding problems with side information for 2D planar flash memory and 3D vertical flash memory. For 2D planar flash memories, the encoder obtains the side information of ICI and tries to combat the ICI. We argue that the 2D planar flash memory channel with ICI can be viewed as similar to Costa's *writing on dirty paper (dirty paper coding)* in [36]. We first explain why flash memories are *dirty* due to ICI. We then show that *dirty flash memory* can be mapped into *memory with stuck-at defects* of [37] due to the unique asymmetry property of flash memory between write and erase operations. After this mapping, we can apply coding for memory with stuck-at defects to combat ICI in 2D planar flash memories. It is interesting that the unique property of flash memory bridges writing on dirty paper and memory with stuck-at defects.

Next, we propose a coding scheme for 3D vertical flash memory. Our coding scheme aims to compensate the effect of fast detrapping by using *intentional* ICI. The basic idea comes from the observation that ICI increases the threshold voltage of a cell whereas fast detrapping decreases the threshold voltage of corresponding cell. In order to properly harness the intentional ICI, we formulate the problem of controlling the intentional ICI into coding for memory with stuck-at defects.

We cast both problems of combating and harnessing the ICI of flash memories as coding for memory with stuck-at defects based on the unique properties of flash memories. Thus, we rely on channel coding with side information about defects as a unified solution for both 2D planar and 3D vertical flash memories. This side information of defects identifies and targets the highly interfered cells of 2D planar flash memory and 3D vertical flash memory cells suffering from significant charge loss due to fast detrapping. In addition, we extend the proposed schemes to MLC flash memories by taking into account the multi-page architecture.

Chapter 3 contains a summary of the material in the following papers:

- Yongjune Kim and B. V. K. Vijaya Kumar, "Writing on dirty flash memory," in *Proc. 52nd Annual Allerton Conference on Communication, Control, and Computing*, Oct. 2014.
- Yongjune Kim, Robert Mateescu, Seung-Hwan Song, Zvonimir Bandic, and B. V. K. Vijaya Kumar, "Coding scheme for 3D vertical flash memory," in *Proc. IEEE International Conference on Communications (ICC)*, Jun. 2015.
- Yongjune Kim, Euiseok Hwang, Robert Mateescu, Seung-Hwan Song, Zvonimir Bandic, and B. V. K. Vijaya Kumar, "Writing on dirty flash memory: combating and harnessing inter-cell interference via coding with side information," *submitted*.

The contributions of Chapter 3 to the study of coding schemes for flash memories are as follows:

- Proposing a coding scheme that combats the ICI in 2D planar flash memories.
- Developing a coding scheme that harnesses the intentional ICI to reduce the effect of fast

detrapping in 3D vertical flash memories.

- Based on the unique property of asymmetry between write and erase operations of flash memory, bridging the well-known Gelfand-Pinsker problems: writing on dirty paper and memory with stuck-at defects.

In Chapter 4, we propose locally rewritable codes (LWC)[1] to improve write endurance and power consumption of resistive memories. Inspired by locally repairable codes (LRC) recently introduced for distributed storage systems, we define a novel parameter of *rewriting locality*, which can be connected to *repair locality* of LRC. As small values of repair locality of LRC enable fast repair in distributed storage systems, small values of rewriting locality of LWC are able to reduce the problems of write endurance and write power consumption.

We show how a small value of rewriting locality can improve write endurance and power consumption by deriving the upper bounds on writing cost. Also, we point out the dual relation of LRC and LWC, which indicates that existing construction methods of LRC can be applied to construct LWC. Finally, we investigate the construction of LWC with error correcting capability for random errors.

Chapter 4 contains a summary of the material in the following papers:

- Yongjune Kim, Abhishek A. Sharma, Robert Mateescu, Seung-Hwan Song, Zvonimir Z. Bandic, James A. Bain, and B. V. K. Vijaya Kumar, "Locally rewritable codes for resistive memories," in *Proc. IEEE International Conference on Communications (ICC)*, May 2016. *(Best Paper Award)*

- Yongjune Kim, Abhishek A. Sharma, Robert Mateescu, Seung-Hwan Song, Zvonimir Z. Bandic, James A. Bain, and B. V. K. Vijaya Kumar, "Locally rewritable codes for resistive memories," *submitted*.

The contributions of Chapter 4 to the study of coding schemes for resistive memories are as follows:

[1]LWC instead of LRC is used as the acronym of locally rewritable codes so as to distinguish them from locally repairable codes (LRC).

- Proposing locally rewritable codes (LWC) for resistive memories.
- Defining a novel parameter of rewriting locality and showing that a small value of rewriting locality of LWC is able to reduce the problems of write endurance and write power consumption.
- Showing the relation between LWC and locally repairable codes (LRC), which indicates that existing LRC construction methods can be applied to construct LWC.

1.3 Notation

An alphabet Q of size q is defined as the set of integers modulo q such that $Q = \{0, 1, \ldots, q-1\}$. We use parentheses to construct column vectors from comma separated lists. For an n-tuple column vector $\mathbf{x} \in Q^n$, we have $\mathbf{x} = (x_1, \ldots, x_n) = [x_1 \ \ldots \ x_n]^T$ where superscript T denotes transpose. Note that x_i represents the i-th element of $\mathbf{x}$.

We use the following notation:

1. For integers i and j such that $i < j$, $[i : j] = \{i, i+1, \ldots, j\}$;
2. For an integer n, $[n] = [1 : n] = \{1, 2, \ldots, n\}$;
3. For a vector $\mathbf{x}$ and a set $\mathcal{J} = \{i_1, \ldots, i_j\} \subseteq [n]$, $\mathbf{x}_{\mathcal{J}} = \big(x_{i_1}, \ldots, x_{i_j}\big)$;
4. For a vector $\mathbf{x}$ and integers i and j such that $i < j$, $\mathbf{x}_{[i:j]} = (x_i, x_{i+1}, \ldots, x_j)$ and $\mathbf{x}_{\setminus i} = \mathbf{x}_{[n]\setminus i} = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$;
5. For a binary vector $\mathbf{x} \in \mathbb{F}_2^n$, $\overline{\mathbf{x}}$ denotes the bit-wise complement of $\mathbf{x}$;
6. $\mathbf{0}_n$ and $\mathbf{1}_n$ denote the n-tuple all-zero vector and all-one vector, respectively (i.e., $\mathbf{1}_n = \overline{\mathbf{0}}_n$);
7. $\mathbf{0}_{m,n}$ and $\mathbf{1}_{m,n}$ denote the $m \times n$ all-zero matrix and all-one matrix, respectively;
8. For a vector $\mathbf{x}$, $\mathsf{supp}(\mathbf{x})$ denotes the support of $\mathbf{x}$, i.e., $\mathsf{supp}(\mathbf{x}) = \{i : x_i \neq 0\}$;
9. $\|\mathbf{x}\|$ and $\|\mathbf{x}\|_0$ denote the ℓ_0-norm of $\mathbf{x}$ (i.e., Hamming weight) and $\|\mathbf{x}\|_0 = |\mathsf{supp}(\mathbf{x})|$;
10. $\|\mathbf{x}\|_1$ denotes the ℓ_1-norm of $\mathbf{x}$ and $\|\mathbf{x}\|_1 = \sum_{i=1}^{n} |x_i|$;

11. $\mathbb{F}_q$ denotes the finite field with q elements;

12. $\mathbb{F}_q^n$ denotes the set of all n-tuple vectors over $\mathbb{F}_q$ and $\mathbb{F}_q^{m\times n}$ the denotes the set of all $m \times n$ matrices over $\mathbb{F}_q$;

13. $\mathcal{W}$ is a bijective function $\mathcal{W} : \mathbb{F}_q \to Q$ such that $\mathcal{W}(c) = x$ for $c \in \mathbb{F}_q$ and $x \in Q$;

14. For $\mathbf{c} \in \mathbb{F}_q^n$ and $\mathbf{x} \in Q^n$, $\mathcal{W}(\mathbf{c}) = \mathbf{x}$ represents $\mathcal{W}(c_i) = x_i$ for all $i \in [n]$.

Chapter 2

Coding for Memory with Stuck-at Defects

2.1 Introduction

In this chapter, we focus on the model of memory with stuck-at defects, which is a notable example of coding with side information at the encoder. This model can be an appropriate channel model for flash memories and resistive memories. For flash memories, we will show that the problems of combating and harnessing inter-cell interference (ICI) can be cast as coding for memory with stuck-at defects. The memory with stuck-at defects is also a proper model for resistive memories because a heavily cycled (i.e., rewritten) resistive memory cells can be regarded as stuck-at defects.

The channel model of memory with defective cells (i.e., stuck-at defects) was introduced by Kuznetsov and Tsybakov in 1974 [37]. As shown in Fig. 2.1, this channel model has a channel state $S \in \{0, 1, \lambda\}$, which is called defect information. The state $S = 0$ corresponds to a stuck-at 0 defect that always outputs a 0 independent of its input value, the state $S = 1$ corresponds to a stuck-at 1 defect that always outputs a 1, and the state $S = \lambda$ corresponds to a normal cell that outputs the same value as its input. The probabilities of 0, 1, λ states are $\beta/2$, $\beta/2$ (assuming a symmetric defect probability), and $1 - \beta$, respectively [34, 37]. We call this channel model the binary defect channel (BDC).

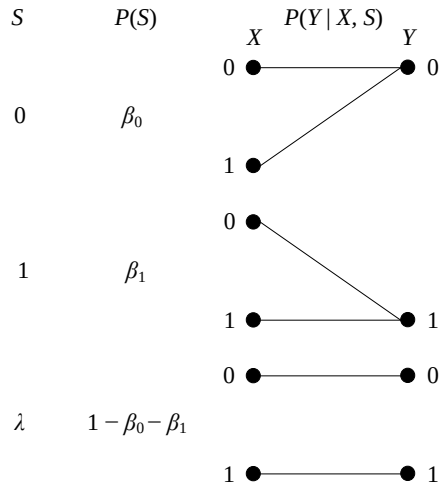


Figure 2.1: Binary defect channel (BDC).

The capacity of the BDC is $1-\beta$ when both the encoder and the decoder know the defect information. If the decoder is aware of the defect locations in an array of memory cells, then the defects can be treated as erasures so that the capacity is $1-\beta$ [34, 38]. On the other hand, Kuznetsov and Tsybakov investigated the model where the encoder knows the channel state information (i.e., locations and stuck-at values of defects) and the decoder does not have any information of defects [37]. It was shown that the capacity is also $1-\beta$ even when only the encoder knows the defect information [37, 38]. Thus, the capacity of the BDC is given by

$$C_{\mathrm{BDC}} = 1-\beta. \tag{2.1}$$

A practical coding scheme for the BDC is *additive encoding* which masks defects by adding a carefully selected binary vector [37, 39]. The goal of masking defects is to make a codeword whose values at the locations of defects match the stuck-at values of corresponding defects.

It was shown in [40] that the capacity can be achieved by an optimal encoding scheme with computational complexity $\mathcal{O}\left(2^n\right)$. In [41], Dumer showed that the capacity can be achieved with encoding complexity $\mathcal{O}\left(n^3\right)$. In addition, capacity-achieving codes with encoding complexity $\mathcal{O}\left(n\log_2^2 n\right)$ were proposed [41].

In regards to the explicit code constructions, it was shown in [40] that cyclic codes such as Bose-Chaudhuri-Hocquenghem (BCH) codes can be used for the BDC. Recently, Mahdavifar and Vardy [42] proposed the explicit code constructions based on polar codes and low-density parity check (LDPC) codes.

Theoretically, channel coding for memory with stuck-at defects can be explained by Gelfand-Pinsker problem. The Gelfand-Pinsker problem assumes that only the encoder knows non-causally the channel state information [32, 34]. It is worth mentioning that another notable example of Gelfand-Pinsker problem is *writing on dirty paper (dirty paper coding)* [34, 36]. Also, the BDC is closely connected to write once memories (WOM), write unidirectional memories (WUM), and some other constrained memories [43, 44].

Recently, the BDC has received renewed attention as a possible channel model for nonvolatile memories such as resistive memories and flash memories [45, 46, 47, 48, 49]. A heavily cycled resistive memory cell's state can not be modulated any more due to unique physical phenomena in resistive memories [12, 13, 14]. Hence, a cell suffering from write endurance problem can be regarded as a stuck-at defect from the perspective of storing data. In flash memories, it has been shown that highly interfered cells can be regarded as stuck-at defects because of the asymmetry between write and erase operations [50].

In [51], the binary defect and symmetric channel (BDSC) model was considered where memory cells suffer from both stuck-at defects and random errors. This channel model is more realistic since random errors happen in resistive memory cells because of write noise, resistance drift and unwanted heating [12, 14]. Similarly, flash memory suffers from random errors due to charge loss, write noise, and random telegraph noise [4].

The channel coding scheme for the BDSC was first explored by Tsybakov [51]. Subsequently, Heegard [40] proposed the partitioned linear block codes (PLBC) that efficiently incorporate the defect information in the encoding process and are capable of correcting both stuck-at errors (due to defects) and random errors.

The PLBC require two generator matrices. One of them is for masking stuck-at defects and

the other generator matrix is for correcting transient errors. Due to these two generator matrices, we can separate the redundancy for masking defects from the redundancy for correcting random errors [40]. We assume that the number of redundant bits for masking defects and correcting random errors are l and r, respectively. Hence, the total redundancy is $l + r = n - k$ (where n is the codeword size and k is the message size). Note that the code rate is $R = k/n = (n-l-r)/n$.

The fact that the redundancy can be divided into two parts leads to the problem of redundancy allocation. The objective is to find, for a fixed total redundancy $n - k$, optimal l and r that minimize the probability of recovery failure. This redundancy allocation problem can be stated as follows:

$$\begin{aligned} (l^*, r^*) = \underset{(l,r)}{\text{argmin}} \quad & P(\widehat{\mathbf{m}} \neq \mathbf{m}) \\ \text{subject to} \quad & l + r = n - k, \quad 0 \leq l \leq n - k, \quad 0 \leq r \leq n - k \end{aligned} \tag{2.2}$$

where $\mathbf{m}$ and $\widehat{\mathbf{m}}$ denote a message and its estimate (recovered message), respectively. $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ denotes the probability of recovery failure.

Not surprisingly, the optimal redundancy allocation (l^*, r^*) depends on the channel. If a channel exhibits only stuck-at defects, we should allot all redundancy to masking stuck-at defects, i.e., $(l^*, r^*) = (n - k, 0)$. It is also expected that the optimal redundancy allocation for a channel with only random errors is $(l^*, r^*) = (0, n - k)$.

In this dissertation, the optimal redundancy allocation for channels that exhibit both stuck-at defects and transient errors (erasures or random errors) will be investigated. First, we investigate the redundancy allocation of the binary defect and erasure channel (BDEC) where transient errors are modeled by erasures. We derive the upper bound on the probability of recovery failure. Based on this upper bound, we can obtain the estimate $(\widehat{l}, \widehat{r})$ for the optimal (l^*, r^*).

For the BDSC, it was shown [40] that PLBC can achieve the channel capacity by the minimum distance encoding (MDE) for masking stuck-at defects and the maximum a posteriori (MAP) decoding for correcting random errors. Since the computational complexities of MDE

and MAP decoding are exponential, we propose a two-step encoding for masking stuck-at defects and consider the standard bounded distance decoding for correcting random errors whose computational complexities are polynomial.

Next, we derive the estimate of the probability of recovery failure for the PLBC with the two-step encoding and the bounded distance decoding. Using this estimate of the probability of recovery failure, we can obtain the estimate $(\widehat{l}, \widehat{r})$ for the optimal redundancy allocation (l^*, r^*). Numerical results show that the estimates $(\widehat{l}, \widehat{r})$ of both the BDEC and the BDSC are well matched with the optimal redundancy allocations.

Next, we study the duality of erasures and defects [52]. This duality can be observed in channel properties, capacities, capacity-achieving schemes, and their failure probability. In [42], it was shown that we can construct capacity-achieving codes for the BDC based on state of the art codes which achieve $C_{\text{BEC}} = 1 - \alpha$. Recently, it was proved that Reed-Muller (RM) codes achieve C_{BEC} under MAP [53]. Based on the duality between the BEC and the BDC, we show that RM codes can achieve C_{BDC} with $\mathcal{O}(n^3)$ complexity.

Also, we extend this duality to the other models such as binary erasure quantization (BEQ) problems [54], and write once memories (WOM) [55]. We review the related literature and describe the relations between these models. From these relations, we can claim that C_{BDC} can be achieved with $\mathcal{O}(n \log n)$ encoding complexity which is better than the best known result in [41], i.e., $\mathcal{O}(n \log^2 n)$ encoding complexity.

2.2 Background for Coding for Defect Channel Model

2.2.1 Defect Channel Model

We summarize the defect channel model (i.e., memory with stuck-at defect) in [37, 40]. Define a variable λ that indicates whether the memory cell is defective or not and $\widetilde{Q} = Q \cup \{\lambda\}$. The channel model of memory with defective cells (i.e., stuck-at defects) in Fig. 2.1 can be described

as follows.

$$Y = X \circ S \tag{2.3}$$

where "$\circ$" denote the operator $\circ : Q \times \widetilde{Q} \to Q$ as in [40].

$$x \circ s = \begin{cases} x, & \text{if } s = \lambda; \\ s, & \text{if } s \neq \lambda. \end{cases} \tag{2.4}$$

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

$$\mathbf{y} = \mathbf{x} \circ \mathbf{s} \tag{2.5}$$

where $\mathbf{x}, \mathbf{y} \in Q^n$ are the channel input and output vectors. Also, the channel state vector $\mathbf{s} \in \widetilde{Q}^n$ represents the defect information in the n-cell memory. Note that $\circ$ is the vector component-wise operator.

If $s_i = \lambda$, this i-th cell is *normal*. If the i-th cell is *defective* (i.e., $s_i \neq \lambda$), its output y_i is stuck-at s_i independent of the input x_i. So, the i-th cell is called stuck-at defect whose stuck-at value is s_i (i.e., stuck-at s_i defect). The probabilities of stuck-at defects and normal cells are given by

$$P(S = s) = \begin{cases} 1 - \beta, & \text{if } s = \lambda; \\ \beta_s, & \text{if } s \neq \lambda \end{cases} \tag{2.6}$$

where the probability of stuck-at defects is $\beta = \sum_{s=0}^{q-1} \beta_s$.

Definition 2.1 (Location and Number of Defects) *The locations of defects $\mathcal{U} \subseteq [n]$ are defined as $\mathcal{U} = \{i \in [n] \mid s_i \neq \lambda\}$. Also, the number of defects u is given by $u = |\mathcal{U}|$.*

Definition 2.2 (Number of Stuck-at Errors) *The number of stuck-at errors (i.e., the number of errors caused by stuck-at defects) is given as $\varepsilon = \|\mathbf{x} \circ \mathbf{s} - \mathbf{x}\|_0$.*

The encoder transforms a message $\mathbf{m} \in \mathbb{F}_q^k$ into a codeword $\mathbf{c} \in \mathbb{F}_q^n$. The channel input

$\mathbf{x} \in Q^n$ is obtained by using a write function $\mathcal{W}$, i.e., $\mathbf{x} = \mathcal{W}(\mathbf{c})$. From the channel output $\mathbf{y} \in Q^n$ of (2.5), the received word $\mathbf{r} \in \mathbb{F}_q^n$ is obtained by

$$\mathbf{r} = \mathcal{R}(\mathbf{y}) = \mathcal{R}(\mathbf{x} \circ \mathbf{s}) \tag{2.7}$$

where $\mathcal{R} = \mathcal{W}^{-1}$. From $\mathbf{r}$, the decoder outputs the estimate of message $\widehat{\mathbf{m}} \in \mathbb{F}_q^k$.

Now we consider the channel models with both permanent stuck-at defects and transient errors. If transient errors come from erasure, $X \circ S$ will be the channel input of the erasure channel with the erasure probability α.

If the transient errors are random errors, the channel model can be given by

$$Y = X \circ S + Z \tag{2.8}$$

where $X \circ S$ comes from (2.3) and $Z \in Q$ denotes random errors.

An n-cell memory with defects and random errors is modeled by

$$\mathbf{y} = \mathbf{x} \circ \mathbf{s} + \mathbf{z} \tag{2.9}$$

where $\mathbf{x} \circ \mathbf{s}$ comes from (2.5) and $\mathbf{z} \in Q^n$ denotes the vector of random errors. Then, the received word $\widetilde{\mathbf{r}} \in \mathbb{F}_q^n$ can be given by

$$\widetilde{\mathbf{r}} = \mathcal{R}(\mathbf{y}) = \mathcal{R}(\mathbf{x} \circ \mathbf{s} + \mathbf{z}) = \mathbf{r} + \mathbf{e} \tag{2.10}$$

where $\mathbf{r} = \mathcal{R}(\mathbf{x} \circ \mathbf{s})$ comes from (2.7) and $\mathbf{e} \in \mathbb{F}_q^n$ denotes the random error vector due to $\mathbf{z} \in Q^n$. By (2.7) and (2.10), $\mathbf{e}$ is given by $\mathbf{e} = \widetilde{\mathbf{r}} - \mathbf{r} = \mathcal{R}(\mathbf{x} \circ \mathbf{s} + \mathbf{z}) - \mathcal{R}(\mathbf{x} \circ \mathbf{s})$.

For binary memory, $X \circ S$ is the input to the binary symmetric channel (BSC) with the

crossover probability p. Hence, Z represents the random error whose probability is given by

$$P(Z=z)=\begin{cases}1-p, & z=0;\\ p, & z=1.\end{cases} \tag{2.11}$$

If stuck-at defects do not suffer from transient errors, the capacity can be easily determined.

Theorem 2.3 *[38] If defective cells (i.e., stuck-at defects) do not suffer from transient errors, then the capacity is given by*

$$\widetilde{C}^{max}=\widetilde{C}^{enc}=P(S=\lambda)\widetilde{C} \tag{2.12}$$

where $\widetilde{C}$ is the capacity of the discrete memoryless channel (DMC) with $P(Y \mid X)=P(Y \mid X, S=\lambda)$. The superscript 'max' in (2.12) represents the capacity when the defect information is fully known to both the encoder and decoder. Also, the superscript 'enc' denotes the capacity when only the encoder knows the defect information. Since $\widetilde{C}^{\text{enc}}$ is the same as $\widetilde{C}^{\text{max}}$, we can omit the superscript if the stuck-at defects do not suffer from transient errors.

2.2.2 Capacity of Binary Memory with Stuck-at Defects

Let $\widetilde{C}_{\text{BDEC}}$ denote the capacity of the binary memory channel when only the normal cells can be erased. Similarly, let $\widetilde{C}_{\text{BDSC}}$ denote the capacity of the binary memory channel when only the normal cells suffer from random errors. By Theorem 2.3, it is clear that

$$\widetilde{C}_{\text{BDEC}}=(1-\beta)(1-\alpha), \tag{2.13}$$

$$\widetilde{C}_{\text{BDSC}}=(1-\beta)(1-h(p)) \tag{2.14}$$

where $h(x)=-x\log_2 x-(1-x)\log_2(1-x)$.

If neither the encoder nor the decoder knows the defect information, the capacity is given

by [38]

$$\widetilde{C}_{\text{BDSC}}^{\text{min}} = 1 - h\left((1-\beta)p + \frac{\beta}{2}\right). \tag{2.15}$$

For the case where both the stuck-at defects and the normal cells suffer from transient errors, it is difficult to derive closed-form expressions for $C_{\text{BDSC}}^{\text{max}}$ and $C_{\text{BDSC}}^{\text{enc}}$. Instead, these capacities can be evaluated numerically [38].

For the BDEC, we can derive $C_{\text{BDEC}}^{\text{max}}$ and $C_{\text{BDEC}}^{\text{enc}}$ due to the simplicity of the channel model. If both the encoder and decoder know the defect information, this channel is equivalent to the BEC with the erasure probability of $\alpha + \beta - \alpha\beta$. Hence,

$$C_{\text{BDEC}}^{\text{max}} = 1 - \alpha - \beta + \alpha\beta = (1-\beta)(1-\alpha) \tag{2.16}$$

which is the same as in (2.13). The capacity $C_{\text{BDEC}}^{\text{enc}}$ is derived as follows.

Proposition 2.4 *If only the encoder knows the defect information, the capacity of the BDEC is given by*

$$C_{BDEC}^{enc} = 1 - \alpha - \beta \tag{2.17}$$

which shows that $C_{BDEC}^{enc} < C_{BDEC}^{max} = \widetilde{C}_{BDEC}$.

Proof: The proof is given in Appendix 2.6. ■

Although $C_{\text{BDEC}}^{\text{enc}} < C_{\text{BDEC}}^{\text{max}} = \widetilde{C}_{\text{BDEC}}$, the difference between $C_{\text{BDEC}}^{\text{enc}}$ and $\widetilde{C}_{\text{BDEC}}$ is only $\alpha\beta$ which is much smaller than α or β for $\alpha, \beta \ll 1$.

For the BDSC, the closed-form expressions for $C_{\text{BDSC}}^{\text{max}}$ and $C_{\text{BDSC}}^{\text{enc}}$ are not known. In [40], only a non-closed-form expression for $C_{\text{BDSC}}^{\text{enc}}$ was derived. Although we cannot obtain a closed-form expression for C_{BDSC}, we can claim the following bound.

Proposition 2.5 *If only the encoder knows the defect information, the capacity of the BDSC is bounded by*

$$C_{BDEC}^{lower} \leq C_{BDSC}^{enc} \leq C_{BDSC}^{upper} \tag{2.18}$$

where

$$C_{BDEC}^{lower} = 1 - \beta - h(p), \tag{2.19}$$

$$C_{BDEC}^{upper} = \widetilde{C}_{BDSC} = (1 - \beta)(1 - h(p)). \tag{2.20}$$

Proof: The proof is given in Appendix 2.7 ■

From (2.19) and (2.20), we can see that the difference between the upper bound and lower bound is only $\beta h(p)$. For $\beta, p \ll 1$, $\beta h(p)$ is much smaller than β and $h(p)$.

Proposition 2.6 *If neither the encoder nor the decoder knows the defect information, the capacity is given by*

$$C_{BDSC}^{min} = \widetilde{C}_{BDSC}^{min} = 1 - h(\widetilde{p}) \tag{2.21}$$

where $\widetilde{C}_{BDSC}^{min}$ is given by (2.15)*, i.e., the capacity of the BSC with the crossover probability of*

$$\widetilde{p} = (1 - \beta)p + \frac{\beta}{2}. \tag{2.22}$$

Proof: Without the defect information, the BDSC is equivalent to the BSC with the crossover probability $\widetilde{p}$. This is true for both cases: random errors happen in 1) only normal cells and 2) both stuck-at defects and normal cells. ■

2.2.3 Additive Encoding

In the defect channel model, it is assumed that the encoder knows the channel state vector $\mathbf{s}$ before writing data to memory [37]. A traditional coding scheme for defect channel is *additive encoding* which masks defects by adding a carefully selected vector [39]. Subsequently, several coding techniques for memory with stuck-at defects were proposed to efficiently mask stuck-at defects [41, 42, 47, 56, 57, 58]. The goal of masking stuck-at defects is to make a codeword whose values at the locations of defects match the stuck-at values of corresponding defects such

that $\varepsilon = \|\mathbf{x} \circ \mathbf{s} - \mathbf{x}\|_0 = 0$. The additive encoding can be described as follows.

Encoding: A message $\mathbf{m} \in \mathbb{F}_q^k$ is encoded to the following codeword:

$$\mathbf{c} = (\mathbf{m}, \mathbf{0}_{n-k}) + \mathbf{c}_0 = (\mathbf{m}, \mathbf{0}_{n-k}) + G_0\mathbf{p} \tag{2.23}$$

where $\mathbf{c}, \mathbf{c}_0 \in \mathbb{F}_q^n$ and $G_0 \in \mathbb{F}_q^{n\times(n-k)}$. Since $\mathbf{c}_0 \in \mathcal{C}_0$ where $\mathcal{C}_0$ is a linear subspace such that $\mathcal{C}_0 \subseteq \mathbb{F}_q^n$, G_0 is the generator matrix of $\mathcal{C}_0$ and $\mathbf{p} \in \mathbb{F}_q^{n-k}$ denotes the parity for additive encoding.

For the given defect information (channel state vector) $\mathbf{s}$ and message $\mathbf{m}$, the encoder should choose $\mathbf{p}$ to mask as many stuck-at defects as possible. If $\varepsilon = \|\mathbf{x} \circ \mathbf{s} - \mathbf{x}\|_0 = 0$, the encoder declares encoding success and $\mathbf{y} = \mathbf{x} \circ \mathbf{s} = \mathbf{x}$ (i.e., $\mathbf{r} = \mathbf{c}$).

Decoding: The decoding can be given by

$$\widehat{\mathbf{m}} = H_0^T\mathbf{r} = H_0^T\mathbf{c} \tag{2.24}$$

where we assume that the additive encoding succeeded. $\widehat{\mathbf{m}}$ represents the recovered message of $\mathbf{m}$. Note that (2.24) is equivalent to the equation of coset codes.

For the systematic codes, G_0 is given by $G_0 = \begin{bmatrix} R^T & I_{n-k} \end{bmatrix}^T$ where $R \in \mathbb{F}_q^{k\times(n-k)}$ and I_{n-k} is the $(n-k)$-dimensional identity matrix [40]. The parity check matrix H_0 of $\mathcal{C}_0$ is given by $H_0 = [I_k \ \ -R]^T \in \mathbb{F}_q^{n\times k}$ such that $H_0^T(\mathbf{m}, \mathbf{0}_{n-k}) = \mathbf{m}$ and $H_0^T G_0 = \mathbf{0}_{k,n-k}$, i.e., $H_0^T\mathbf{c}_0 = \mathbf{0}_k$.

The *minimum distance* of additive encoding is given by

$$d^\star = \min_{\substack{\mathbf{c}\neq\mathbf{0} \\ G_0^T\mathbf{c}=\mathbf{0}}} \|\mathbf{c}\|_0 \tag{2.25}$$

which means that any $d^\star - 1$ rows of G_0 are linearly independent. Thus, additive encoding guarantees masking up to $d^\star - 1$ stuck-at defects [39, 40].

2.2.4 Partitioned Linear Block Codes (PLBC)

Heegard [40] proposed the partitioned linear block codes (PLBC) that efficiently incorporate the defect information in the encoding process and are capable of correcting both stuck-at errors (errors due to stuck-at defects) and random errors. The (n, k) PLBC consists of a pair of linear subspaces $\mathcal{C}_1 \subset \mathbb{F}_q^n$ of dimension k and $\mathcal{C}_0 \subset \mathbb{F}_q^n$ of dimension l such that $\mathcal{C}_1 \cap \mathcal{C}_0 = \{\mathbf{0}\}$. The PLBC requires two generator matrices. The generator matrix of $\mathcal{C}_0$ is aimed at masking stuck-at defects and the generator matrix of $\mathcal{C}_1$ is for correcting random errors.

Due to these two generator matrices, we can separate the parity for masking defects from the parity for correcting random errors. Since the size of total parity part is $n - k$ and the parity size for masking defects is l, the parity size for correcting random errors is $n - k - l$.

The encoding and decoding of the PLBC are as follows [40]:

Encoding: A message $\mathbf{m} \in \mathbb{F}_q^k$ is encoded to a corresponding codeword $\mathbf{c} \in \mathcal{C}$ as follows.

$$\mathbf{c} = \mathbf{c}_1 + \mathbf{c}_0 = G_1\mathbf{m} + G_0\mathbf{p} = \widetilde{G} \begin{bmatrix} \mathbf{m} \\ \mathbf{p} \end{bmatrix} \tag{2.26}$$

where $G_1 \in \mathbb{F}_q^{n\times k}$, and $G_0 \in \mathbb{F}_q^{n\times l}$, and $\widetilde{G} = [G_1 \ G_0] \in \mathbb{F}_q^{n\times(k+l)}$. Also, $\mathbf{c}_1 = G_1\mathbf{m} \in \mathcal{C}_1$ and $\mathbf{c}_0 = G_0\mathbf{p} \in \mathcal{C}_0$. Note that $\mathbf{p} \in \mathbb{F}_q^l$ is the parity (redundancy) for masking defects. Thus, the PLBC $\mathcal{C}$ can be viewed as an $(n, k+l)$ linear block code with the generator matrix $\widetilde{G} = [G_1 \ G_0]$.

After constructing $\mathbf{c}_1 = G_1\mathbf{m}$, the encoder should choose $\mathbf{p}$ carefully to mask as many stuck-at defects as possible by comparing $\mathbf{s}$ and $\mathbf{c}_1$. If $\varepsilon = 0$, then the encoder declares encoding success. i.e., $\mathbf{y} = \mathbf{x} \circ \mathbf{s} = \mathbf{x}$ and $\mathbf{r} = \mathbf{c}$.

Decoding: Receive $\widetilde{\mathbf{r}} = \mathbf{r} + \mathbf{e}$ according to (2.10). If the encoding succeeded, then the syndrome $\mathbf{v} = \widetilde{H}^T\widetilde{\mathbf{r}} = \widetilde{H}^T(\mathbf{c} + \mathbf{e}) = \widetilde{H}^T\mathbf{e}$ where $\widetilde{H} \in \mathbb{F}_q^{n\times r}$ denotes the parity check matrix such that $\widetilde{H}^T\widetilde{G} = \mathbf{0}_{r,k+l}$. From the syndrome $\mathbf{v}$, the decoder guesses the estimate of $\widehat{\mathbf{e}} \in \mathbb{F}_q^n$ and $\widehat{\mathbf{c}} = \widetilde{\mathbf{r}} - \widehat{\mathbf{e}}$. Then $\widehat{\mathbf{m}} = \widetilde{G}_1^T\widehat{\mathbf{c}}$ where $\widetilde{G}_1 \in \mathbb{F}_q^{n\times k}$ denotes the message inverse matrix such that $\widetilde{G}_1^T G_1 = I_k$ and $\widetilde{G}_1^T G_0 = \mathbf{0}_{k,l}$.

In [40], a pair of minimum distances $(d^\star, \widetilde{d}\,)$ of an (n, k) PLBC are defined where

$$\widetilde{d} = \min_{\substack{\widetilde{G}_1^T \mathbf{c} \neq \mathbf{0} \\ \widetilde{H}^T \mathbf{c} = \mathbf{0}}} \|\mathbf{c}\|_0 \tag{2.27}$$

and $d^\star$ was given by (2.25). Note that $\widetilde{d}$ is greater than or equal to the minimum distance of the $(n, k + l)$ linear block code with the parity check matrix $\widetilde{H}$ [40].

In [40], partitioned cyclic codes were proposed. An (n, k) partitioned cyclic code is an (n, k) PLBC such that $\mathcal{C}$ and $\mathcal{C}_0$ are cyclic. A partitioned cyclic code can be described by two generator polynomials: $g(x)$ of degree $n - k - l$ and $g_0(x)$ of degree $n - l$ satisfying $g(x) \mid g_0(x)$, i.e.,

$$g_0(x) = g(x)q(x). \tag{2.28}$$

Encoding: A message polynomial $m(x)$ is encoded to a corresponding codeword

$$c(x) = m(x)g(x) + p(x)g_0(x) \tag{2.29}$$

where $p(x)$ should be chosen carefully in order to mask stuck-at defects.

Decoding: Let the received polynomial be $\widetilde{r}(x) = r(x) + e(x)$. If the encoding succeeded, then $\widetilde{r}(x) = c(x) + e(x)$. The decoder computes the syndrome polynomial $v(x) = \widetilde{r}(x) \mod g(x)$ and chooses $\widehat{e}(x)$ such that $\widehat{e}(x) \mod g(x) = v(x)$. Then,

$$\widehat{m}(x) = \frac{(\widetilde{r}(x) - \widehat{e}(x)) \mod g_0(x)}{g(x)} \tag{2.30}$$

which follows from $c(x) = m(x)g(x) + p(x)g_0(x) = g(x)\{m(x) + p(x)q(x)\}$ due to (2.28).

It is critical to choose the proper $\mathbf{p}$ during the encoding stage of PLBC. For the binary memory

case, the MDE chooses $\mathbf{p}$ as follows.

$$\mathbf{p}^* = \underset{\mathbf{p}}{\operatorname{argmin}} \|\mathbf{x} \circ \mathbf{s} - \mathbf{x}\| = \underset{\mathbf{p}}{\operatorname{argmin}} \|\mathbf{c} \circ \mathbf{s} - \mathbf{c}\| \tag{2.31}$$

$$= \underset{\mathbf{p}}{\operatorname{argmin}} \left\|\mathbf{s}^{\mathcal{U}} + G_0^{\mathcal{U}}\mathbf{p} + G_1^{\mathcal{U}}\mathbf{m}\right\|$$

$$= \underset{\mathbf{p}}{\operatorname{argmin}} \left\|G_0^{\mathcal{U}}\mathbf{p} + \mathbf{b}^{\mathcal{U}}\right\| \tag{2.32}$$

where $\|\mathbf{x} \circ \mathbf{s} - \mathbf{x}\|$ represents the number of errors due to stuck-at defects. Also, $\mathcal{U} = \{i_1, \ldots, i_u\}$ indicates the set of locations of stuck-at defects. We use the notation of $\mathbf{s}^{\mathcal{U}} = (s_{i_1}, \ldots, s_{i_u})^T$, $G_0^{\mathcal{U}} = \left[\mathbf{g}_{0,i_1}^T, \ldots, \mathbf{g}_{0,i_u}^T\right]^T$, and $G_1^{\mathcal{U}} = \left[\mathbf{g}_{1,i_1}^T, \ldots, \mathbf{g}_{1,i_u}^T\right]^T$ where $\mathbf{g}_{0,i}$ and $\mathbf{g}_{1,i}$ are the i-th rows of G_0 and G_1 respectively. Also, note that $\mathbf{b} = G_1\mathbf{m} + \mathbf{s}$ and $\mathbf{b}^{\mathcal{U}} = G_1^{\mathcal{U}}\mathbf{m} + \mathbf{s}^{\mathcal{U}}$.

By solving the optimization problem in (2.32), we can minimize the number of errors due to stuck-at defects. It was shown [40] that the capacity of the BDSC can be achieved by the MDE and the MAP decoding.

However, the computation complexity of the MDE is impractical, i.e., $\mathcal{O}(2^n)$. Hence, we consider the polynomial time encoding [39]. Instead of solving the exponential complexity optimization problem, we just try to solve the following linear equation.

$$G_0^{\mathcal{U}}\mathbf{p} = \mathbf{b}^{\mathcal{U}} \tag{2.33}$$

Gaussian elimination or some other linear equation solution methods can be used to solve (2.33) with $\mathcal{O}\left(n^3\right)$ complexity. If the encoder fails to find a solution of (2.33), then encoding failure is declared. It is clear that (2.33) has at least one solution if and only if

$$\operatorname{rank}\left(G_0^{\mathcal{U}}\right) = \operatorname{rank}\left(G_0^{\mathcal{U}} \mid \mathbf{b}^{\mathcal{U}}\right) \tag{2.34}$$

where $\left(G_0^{\mathcal{U}} \mid \mathbf{b}^{\mathcal{U}}\right)$ denotes the augmented matrix. If $u < d^\star$, $\operatorname{rank}\left(G_0^{\mathcal{U}}\right)$ is always u by (2.25). Hence, (2.34) holds and at least one solution $\mathbf{p}$ exists. If $u \geq d^\star$, the encoder may fail to find a

solution of (2.33).

For convenience, we define a random variable E as follows.

$$E = \begin{cases} 1, & \|\mathbf{c} \circ \mathbf{s} - \mathbf{c}\| = 0 \text{ (encoding success)} \\ 0, & \|\mathbf{c} \circ \mathbf{s} - \mathbf{c}\| \neq 0 \text{ (encoding failure)} \end{cases} \tag{2.35}$$

We can observe that the probability of encoding failure $P(E = 0)$ by solving (2.32) is the same as $P(E = 0)$ by solving (2.33). It is because $G_0^{\mathcal{U}}\mathbf{p} \neq \mathbf{b}^{\mathcal{U}}$ if and only if $\|\mathbf{c} \circ \mathbf{s} - \mathbf{c}\| \neq 0$. Although solving (2.33) is suboptimal in regards to the number of stuck-at errors (i.e., $\|\mathbf{c}\circ\mathbf{s}-\mathbf{c}\|$), C_{BDC} can be achieved by solving (2.33) instead of (2.32), which is easily shown by using the results of [40].

Proposition 2.7 *C_{BDC} can be achieved by solving* (2.33)*.*

Proof: Suppose that each element of G_0 is selected at random with equal probability from $\{0, 1\}$. Then,

$$\begin{aligned} P\left(E = 0\right) &= P\left(E = 0, n(\beta - \epsilon) \leq |\mathcal{U}| \leq n(\beta + \epsilon)\right) + \epsilon' \\ &\leq \sum_{u=n(\beta-\epsilon)}^{n(\beta+\epsilon)} P(E = 0 \mid |\mathcal{U}| = u) + \epsilon' \\ &\leq \sum_{u=n(\beta-\epsilon)}^{n(\beta+\epsilon)} P\left(\text{rank}\left(G_0^{\mathcal{U}}\right) < u \mid |\mathcal{U}| = u\right) + \epsilon' \\ &\leq (2n\epsilon + 1) \cdot \frac{2^{n(\beta+\epsilon)}}{2^{n-k}} + \epsilon' \\ &= (2n\epsilon + 1)2^{n\left(\frac{k}{n} - (1-\beta) + \epsilon\right)} + \epsilon' \end{aligned} \tag{2.36}$$

where we assume that $n(\beta \pm \epsilon)$ are integers without loss of generality. If $R = \frac{k}{n} < C_{\text{BDC}} - \epsilon$, $P\left(E = 0\right)$ converges to zero as $n \to \infty$. ■

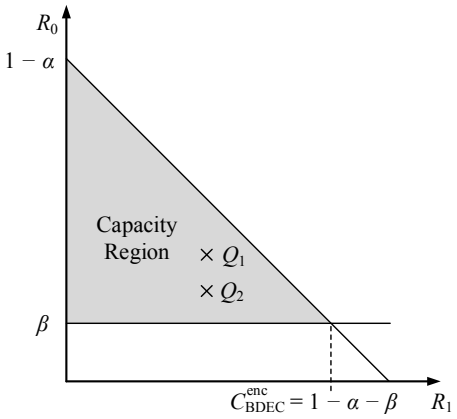


Figure 2.2: Capacity region of the BDEC derived in Proposition 2.14. Two points of Q_1 and Q_2 in the capacity region represent the pairs of code rates $\left(R_1 = \frac{k}{n}, R_0 = \frac{l}{n}\right)$. Note that the capacity $C_{\text{BDEC}}^{\text{enc}}$ is the supremum of $R_1 = R$.

2.3 Redundancy Allocation of Finite-Length PLBC

In this section, we investigate the redundancy allocation for finite-length PLBC. In order to clarify the redundancy allocation problem for finite-length PLBC, we define a pair of code rates (R_1, R_0) where $R_0 = \frac{l}{n}$ is the code rate of $\mathcal{C}_0$. Also, $R_1 = \frac{k}{n}$ is the code rate of $\mathcal{C}_1$, which is equivalent to the actual code rate $R = \frac{k}{n}$. For the given codeword size n, we can readily calculate (l, r) from (R_1, R_0).

For the BDEC, we will derive the capacity region $\mathbb{C}_{\text{BDEC}}$ in Proposition 2.14. If $(R_1, R_0) \in \mathbb{C}_{\text{BDEC}}$, then $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ converges to zero as $n \to \infty$. In Fig. 2.2, Q_1 and Q_2 represent two pairs of code rates in $\mathbb{C}_{\text{BDEC}}$ with the same $R_1 = R$. Then, Q_1 and Q_2 have the same total redundancy $n - k = l + r$ whereas they have the different redundancy allocations (l, r).

Asymptotically, both Q_1 and Q_2 make $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ converge to zero. On the other hand, $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ of Q_1 and Q_2 can be significantly different for the finite-length codes, which leads to the need for formulating and solving the redundancy allocation problem in (2.2).

In order to choose the optimal redundancy allocation (l^*, r^*) of (2.2), we should derive the $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ which depends on the channel parameters (i.e., β, α, p) as well as the code parameters (n, k, l). In the following subsection, we will derive the upper bound on encoding failure

probability for finite-length codes, which is important for deriving the estimate of $P(\widehat{\mathbf{m}} \neq \mathbf{m})$.

2.3.1 Upper Bound on Encoding Failure Probability

Since we focus on the redundancy allocation problem in finite-length codes, we derive the upper bound on $P(E = 0)$ for finite n. During the encoding stage, $\mathbf{p}$ is chosen by solving the linear equation (2.33) instead of the optimization problem (2.32).

Lemma 2.8 *The upper bound on the probability of encoding failure given u defects is given by*

$$P\left(E = 0 \mid U = u\right) \leq \min\left\{\frac{\sum_{w=d^\star}^{u} B_{0,w}\binom{n-w}{u-w}}{\binom{n}{u}}, 1\right\} \tag{2.37}$$

where the random variable U represents the number of stuck-at defects. In addition, $B_{0,w}$ is the weight distribution of $\mathcal{C}_0^\perp$ (i.e., the dual code of $\mathcal{C}_0$).

Proof: The proof is given in Appendix 2.8. ■

Lemma 2.8 supports that $P\left(E = 0 \mid U = u\right) = 0$ for $u < d^\star$. The following Lemma states that $P\left(E = 0 \mid U = u\right)$ can be obtained exactly for $d^\star \leq u \leq d^\star + \left\lfloor \frac{d^\star - 1}{2} \right\rfloor$ where $\lfloor x \rfloor$ is the largest integer not greater than x.

Lemma 2.9 *For $u \leq d^\star + \left\lfloor \frac{d^\star - 1}{2} \right\rfloor$, $P\left(E = 0 \mid U = u\right)$ is given by*

$$P\left(E = 0 \mid U = u\right) = \frac{1}{2} \cdot \frac{\sum_{w=d^\star}^{u} B_{0,w}\binom{n-w}{u-w}}{\binom{n}{u}}. \tag{2.38}$$

Proof: The proof is given in Appendix 2.9. ■

From the definition of $d^\star$ in (2.25), Lemma 2.8, and Lemma 2.9, we can state the following theorem.

Theorem 2.10 *$P\left(E=0 \mid U=u\right)$ is given by*

$$P\left(E=0 \mid U=u\right)=\begin{cases} 0, & \text{for } u<d^{\star} \quad (2.39)\\ \frac{1}{2}\cdot\frac{\sum_{w=d^{\star}}^{u} B_{0,w}\binom{n-w}{u-w}}{\binom{n}{u}}, & \text{for } d^{\star}\le u\le d^{\star}+t^{\star} \quad (2.40)\\ \le \min\left\{\frac{\sum_{w=d^{\star}}^{u} B_{0,w}\binom{n-w}{u-w}}{\binom{n}{u}}, 1\right\}, & \text{for } u>d^{\star}+t^{\star}. \quad (2.41)\end{cases}$$

where $t^{\star}=\left\lfloor\frac{d^{\star}-1}{2}\right\rfloor$.

We compare our upper bounds and simulation results assuming the the number of defects u is given. The $[n=31,k,l]$ PBCH codes are considered and the weight distributions $B_{0,w}$ are calculated using the MacWilliams identity. Fig. 2.3 shows that the upper bounds of (2.41) are close to the simulation results for $P(E=0 \mid U=u)$. In addition, the calculated values of (2.40) are well matched with the simulation results. Fig. 2.3 shows that the upper bounds approach $P(E=0 \mid U=u)$ and meet $P(E=0 \mid U=u)$ as the code rate decreases.

From the bound on $P(E=0 \mid U=u)$ in Theorem 2.10, we derive the following upper bound.

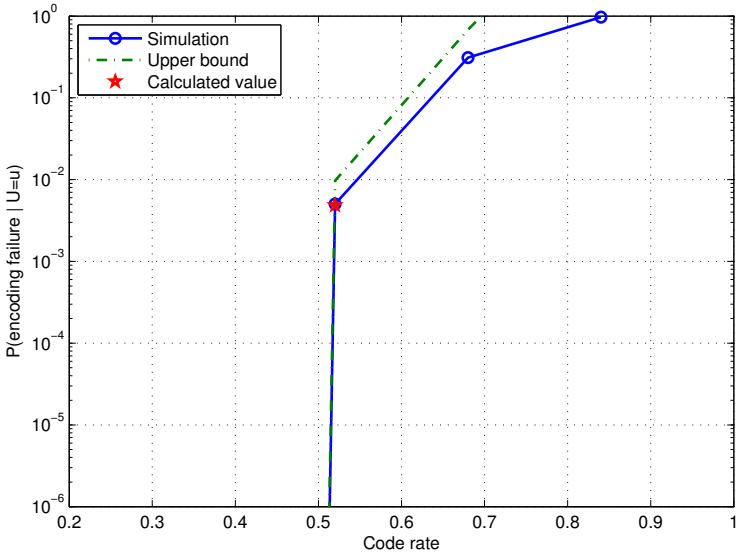
Corollary 2.11 *The upper bound on the probability of encoding failure $P\left(E=0\right)$ is given by*

$$P(E=0)\le\sum_{u=d^{\star}}^{n}\beta^{u}\left(1-\beta\right)^{n-u}\sum_{w=d^{\star}}^{u}B_{0,w}\binom{n-w}{u-w}. \quad (2.42)$$

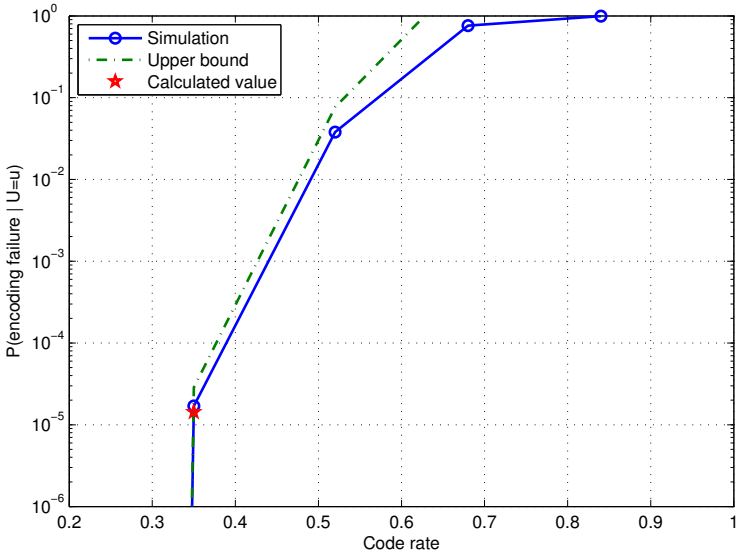
Proof: The proof is straightforward.

$$\begin{aligned} P(E=0) &= \sum_{u=d^{\star}}^{n} P(U=u)P(E=0 \mid U=u)\\ &\le \sum_{u=d^{\star}}^{n}\binom{n}{u}\beta^{u}(1-\beta)^{n-u}\min\left\{\frac{\sum_{w=d^{\star}}^{u} B_{0,w}\binom{n-w}{u-w}}{\binom{n}{u}}, 1\right\}\\ &\le \sum_{u=d^{\star}}^{n}\beta^{u}\left(1-\beta\right)^{n-u}\sum_{w=d^{\star}}^{u}B_{0,w}\binom{n-w}{u-w}\end{aligned}$$

where $P(U=u)=\binom{n}{n}\beta^{u}(1-\beta)^{n-u}$. ■



(a) $u = 10$



(b) $u = 12$

Figure 2.3: Comparison of simulation results, upper bounds by (2.41), and calculated values by (2.40) for $P(E = 0 \mid U = u)$. $[n = 31, k, l]$ PBCH codes are used. The code rate is $R = k/n$.

Since it is intractable to compute $B_{0,w}$ for large n, we consider the following binomial approximation.

$$B_{0,w} \cong 2^{-l} \binom{n}{w} \tag{2.43}$$

For many codes including random linear codes (each element of generator matrix is chosen uniformly at random from $\{0, 1\}$) and BCH codes, it is known that the weight distribution is well

approximated by the binomial distribution [59].

Corollary 2.12 *If the weight distribution $B_{0,w}$ follows the binomial approximation, the upper bound on the probability of encoding failure $P(E=0)$ is given by*

$$P(E=0) \leq 2^{-l}(1+\beta)^n, \tag{2.44}$$

$$\log_2 P(E=0) \leq n\{R-(1-\log_2(1+\beta))\}. \tag{2.45}$$

Proof: From (2.42) and (2.43), the upper bound on $P(E=0)$ can be derived as follows.

$$P(E=0) \leq \sum_{u=d^\star}^{n} \beta^u (1-\beta)^{n-u} \sum_{w=d^\star}^{u} B_{0,w} \binom{n-w}{u-w} \tag{2.46}$$

$$= 2^{-l} \sum_{u=d^\star}^{n} \beta^u (1-\beta)^{n-u} \sum_{w=d^\star}^{u} \binom{u}{w}\binom{n}{u} \tag{2.47}$$

$$\leq 2^{-l} \sum_{u=0}^{n} \binom{n}{u} (2\beta)^u (1-\beta)^{n-u} \tag{2.48}$$

$$= 2^{-l}(1+\beta)^n \tag{2.49}$$

where (2.47) follows from $\binom{n}{w}\binom{n-w}{u-w} = \binom{u}{w}\binom{n}{u}$. Also, (2.48) follows from the binomial theorem $\sum_{w=0}^{u}\binom{u}{w} = 2^u$. (2.45) can be obtained by taking the logarithm. ■

Fig. 2.4 shows that the upper bound is very close to the simulation results when the probability of encoding failure is low. In regards to the simulation results, we used PBCH codes for the BDC with $\beta = 0.1$. The plotted upper bounds are based on the binomial approximation. In spite of this approximation, our upper bound is very close to the simulation results.

Remark 2.13 (2.44) *shows that the probability of encoding failure decreases as l increases, whereas the probability of encoding failure increases as β increases. For infinite-length codes, this upper bound is not tight since $1-\log_2(1+\beta)$ of* (2.45) *is less than $C_{BDC} = 1-\beta$. However, this upper bound is tight for finite-length codes as shown in Fig. 2.4. Moreover, the upper bound in* (2.45) *is the linear function of R where n is the slope and $1-\log_2(1+\beta)$ is the R-intercept,*

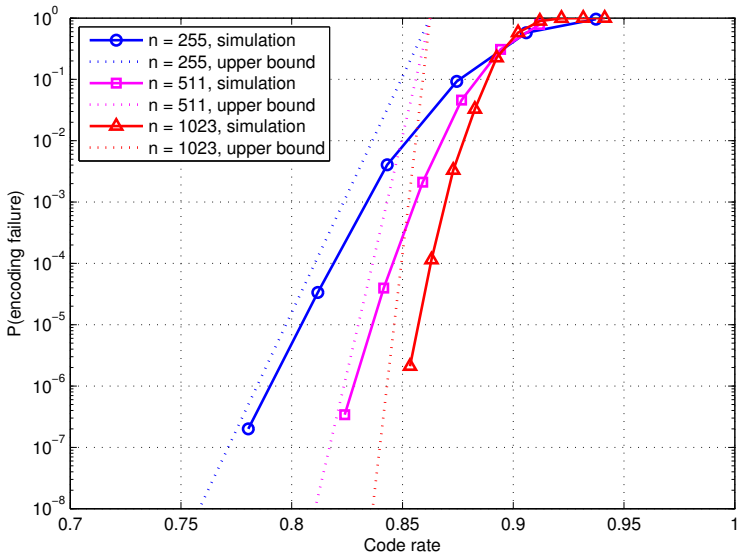


Figure 2.4: Comparison of simulation results and upper bounds in (2.44) for the probability of encoding failure $P(E = 0)$. We used PBCH codes for the BDC with probability of defect $\beta = 0.1$.

which explicitly shows that longer codes improve the probability of encoding failure.

2.3.2 Redundancy Allocation: BDEC

For the BDEC, the encoder should try to mask stuck-at defects and the decoder should correct erasures. The encoding process of the BDEC is the same as the encoding of PLBC. Only the decoding will be modified as follows.

Decoding (BDEC): The MAP decoding is performed by solving the following linear equation.

$$\widetilde{G}^{\mathcal{V}} \begin{bmatrix} \widehat{\mathbf{m}} \\ \widehat{\mathbf{p}} \end{bmatrix} = \mathbf{y}^{\mathcal{V}} \tag{2.50}$$

where $\mathcal{V} = \{j_1, \cdots, j_v\}$ indicates the locations of v unerased bits. We use the notation of $\mathbf{y}^{\mathcal{V}} = (y_{j_1}, \cdots, y_{j_v})^T$ and $\widetilde{G}^{\mathcal{V}} = \left[\widetilde{\mathbf{g}}_{j_1}^T, \cdots, \widetilde{\mathbf{g}}_{j_v}^T\right]^T$ where $\widetilde{\mathbf{g}}_j$ is the j-th row of $\widetilde{G}$. By solving (2.50), we can obtain estimates of the message $\mathbf{m}$ and redundancy $\mathbf{p}$, i.e., $\widehat{\mathbf{m}}$ and $\widehat{\mathbf{p}}$.

We consider the MAP decoding in order to derive the upper bound on $P\left(\widehat{\mathbf{m}} \neq \mathbf{m}\right)$. The MAP decoding can be accomplished by solving the linear equation in (2.50), whose complexity

is $\mathcal{O}(n^3)$.

Proposition 2.14 *If a pair of code rates (R_0, R_1) satisfy the following conditions, then the probability of recovery failure $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ approaches zero as $n \to \infty$.*

$$R_0 > \beta, \quad R_1 + R_0 < 1 - \alpha \tag{2.51}$$

where α is the probability of erasure and β is the probability of defect for the BDEC.

Proof: We show that $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ approaches zero with n if (R_0, R_1) satisfies (2.51). We can claim that $P(\widehat{\mathbf{m}} \neq \mathbf{m}) = P(E = 0) + P(E = 1, D = 0)$ where the random variable D takes on two values as follows.

$$D = \begin{cases} 1, & \text{decoding success} \\ 0, & \text{decoding failure} \end{cases} \tag{2.52}$$

From (2.36), we can claim that $P(E = 0) \leq n(\beta + \epsilon)2^{-n\left(\frac{l}{n} - \beta - \epsilon\right)} + \epsilon'$. Hence, $P(E = 0)$ converges to zero if $R_0 = \frac{l}{n} > \beta$. If the additive encoding succeeds (i.e., $E = 1$), then the corresponding channel is equivalent to the BEC with the erasure probability α. It is because all the stuck-at defects are masked. Note that the code rate of $\widetilde{G}$ of (2.50) is $\frac{k+l}{n} = R_0 + R_1$. Thus, $R_0 + R_1 < 1 - \alpha$. ■

Fig. 2.2 represents the capacity region by Proposition 2.14. The supremum of R_1 in this capacity region is $1 - \alpha - \beta$ which is equal to $C_{\text{BDEC}}^{\text{enc}}$ of (2.17). From (2.51), we can obtain the following conditions for (l, r) which achieve the capacity for infinite n.

$$l > n\beta, \quad r > n\alpha \tag{2.53}$$

As explained earlier, these asymptotic results cannot be directly used to choose the optimal redundancy allocation (l^*, r^*) of (2.2) for finite-length codes. We emphasize that the optimal re-

Table 2.1: All Possible Redundancy Allocation Candidates of $[n = 1023, k = 923, l]$ PBCH Codes

Code	l	r	$d^\star$	$\widetilde{d}$	Notes
0	0	100	0	21	Only correcting transient errors
1	10	90	3	19	
2	20	80	5	17	
3	30	70	7	15	
4	40	60	9	13	
5	50	50	11	11	
6	60	40	13	9	
7	70	30	15	7	
8	80	20	17	5	
9	90	10	19	3	
10	100	0	21	0	Only masking stuck-at defects

dundancy allocation (l^*, r^*) is equivalent to finding the optimal (i.e., minimizing the probability of recovery error) point in the capacity region.

In order to solve the optimization problem of (2.2), we need a closed-form expression for $P(\widehat{\mathbf{m}} \neq \mathbf{m})$. Unfortunately, it is difficult to obtain the exact expression of $P(\widehat{\mathbf{m}} \neq \mathbf{m})$. Instead, we will consider an estimate $(\widehat{l}, \widehat{r})$ which minimizes the upper bound on $P(\widehat{\mathbf{m}} \neq \mathbf{m})$. The upper bound on $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ for finite-length codes is stated as follows.

Theorem 2.15 *The upper bound on $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ of the BDEC is given by*

$$\begin{aligned} P(\widehat{\mathbf{m}} \neq \mathbf{m}) \leq & \sum_{u=d^\star}^{n} \beta^u (1-\beta)^{n-u} \sum_{w=d^\star}^{u} B_{0,w} \binom{n-w}{u-w} \\ & + \sum_{e=\widetilde{d}}^{n} \alpha^e (1-\alpha)^{n-e} \sum_{w=\widetilde{d}}^{e} A_w \binom{n-w}{e-w}. \end{aligned} \tag{2.54}$$

where A_w is the weight distribution of $\mathcal{C}$. If A_w and $B_{0,w}$ follow the binomial distribution,

$$P(\widehat{\mathbf{m}} \neq \mathbf{m}) \leq 2^{-l}(1+\beta)^n + 2^{-r}(1+\alpha)^n \tag{2.55}$$

Proof: The proof is given in Appendix 2.10. ■

From these upper bounds on $P(\widehat{\mathbf{m}} \neq \mathbf{m})$, we can modify (2.2) for the BDEC as follows.

$$\begin{aligned}(\widehat{l}, \widehat{r}) = \underset{(l,r)}{\text{argmin}} \quad & \text{Upper bound on } P(\widehat{\mathbf{m}} \neq \mathbf{m}) \\ \text{subject to} \quad & l + r = n - k, \quad 0 \leq l \leq n - k, \quad 0 \leq r \leq n - k\end{aligned} \tag{2.56}$$

If the code parameters $(n, k, l, d^{\star}, \widetilde{d})$ and the channel parameters (α, β) are given, the solution $(\widehat{l}, \widehat{r})$ of (2.56) can be obtained. To illustrate this, we consider $[n = 1023, k = 923, l]$ PBCH codes. All possible redundancy allocation candidates of PBCH codes are listed in Table 2.1. Since l and r are multiples of 10 (i.e., the degree of the Galois field of $[n = 1023, k]$ BCH codes), there are only 11 redundancy allocation candidates. Hence, we can readily obtain the $(\widehat{l}, \widehat{r})$ that minimizes the objective function of (2.56).

In addition, the objective function is *convex* if we treat l and r as real values, even though we know that they are non-negative integers less than or equal to $n - k$. We can derive the solution $(\widetilde{l}, \widetilde{r})$ of (2.56) satisfying *Karush-Kuhn-Tucker* (KKT) conditions.

Corollary 2.16 *Treating l and r as real values, the solution of* (2.56) *satisfying KKT conditions is given by*

$$(\widetilde{l}, \widetilde{r}) = \begin{cases} (0, n-k), & \textit{for } \frac{1+\alpha}{1+\beta} > 2^{1-R} \quad (2.57) \\ \left(\check{l}, \check{r}\right), & \textit{for } 2^{-(1-R)} \leq \frac{1+\alpha}{1+\beta} \leq 2^{1-R} \quad (2.58) \\ (n-k, 0), & \textit{for } \frac{1+\alpha}{1+\beta} < 2^{-(1-R)} \quad (2.59) \end{cases}$$

where $\left(\check{l}, \check{r}\right)$ is given by

$$\check{l} = \frac{1}{2}\left\{n\left(1 - \log_2 \frac{1+\alpha}{1+\beta}\right) - k\right\}, \tag{2.60}$$

$$\check{r} = \frac{1}{2}\left\{n\left(1 + \log_2 \frac{1+\alpha}{1+\beta}\right) - k\right\}. \tag{2.61}$$

Proof: The proof is given in Appendix 2.11. ■

Table 2.2: BDEC with the Same $C_{\text{BDEC}} = 0.96$

Channel	α	β	Notes
1	0.040	0	BEC
2	0.035	0.005	
3	0.025	0.015	
4	0.020	0.020	
5	0.015	0.025	
6	0.005	0.035	
7	0	0.040	BDC

If α is much larger than β such that $\frac{1+\alpha}{1+\beta} > 2^{1-R}$ for all possible (l, r), then (2.57) shows that we should allot all redundancy for correcting erasures to minimize the upper bound of (2.56). If β is much larger than α such that $\frac{1+\alpha}{1+\beta} < 2^{-(1-R)}$ for all possible (l, r), then (2.59) shows that we should allot all redundancy for masking stuck-at defects to minimize the upper bound. For other α and β, we should allot the redundancy (l, r) such that $2^{-l}(1+\beta)^n = 2^{-r}(1+\alpha)^n$ to minimize the upper bound, which are satisfied by (2.60) and (2.61).

Remark 2.17 *If only the normal cells can be erased, the corresponding channel's erasure probability is* $\widetilde{\alpha} = (1-\beta)\alpha$. *Hence the capacity will be* $1 - \widetilde{\alpha} - \beta = (1-\beta)(1-\alpha)$ *which is the same as* (2.13). *The equivalent results of Proposition 2.14, Theorem 2.15, and Corollary 2.16 can be obtained by replacing* α *by* $\widetilde{\alpha}$.

In order to compare the optimal redundancy and the estimated redundancy allocation based on the derived upper bound, we consider several channels shown in Table 2.2 whose capacities are all equal, i.e., $C^{\text{enc}}_{\text{BDEC}} = 0.96$. For these channels, we compare the performance of PBCH codes in Table 2.1.

Fig. 2.5 shows the simulation results for the channels in Table 2.2. The simulation results of channel 1 (BEC) and channel 7 (BDC) are omitted because their optimal redundancy allocations are obvious. The optimal redundancy allocation for channel 1 (BEC) is $(l^*, r^*) = (0, 100)$. The more stuck-at defects a channel has, the larger l is expected to be for the optimal redundancy

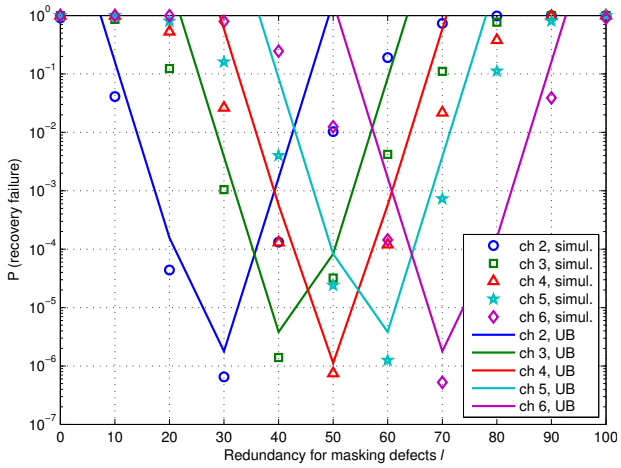


Figure 2.5: Comparison of simulation results (simul.) and upper bounds (UB) of the probability of recovery failure $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ of BDEC channels in Table 2.2.

allocation. Eventually, the optimal redundancy allocation for channel 7 (BDC) will be $(l^*, r^*) = (100, 0)$. The optimal (l^*, r^*) can be obtained from Monte-Carlo simulation results in Fig. 2.5, which are presented in the second column of Table 2.5.

We can readily obtain the $(\widehat{l}, \widehat{r})$ that minimizes the upper bounds in Fig. 2.5. The estimates of redundancy allocation $(\widehat{l}, \widehat{r})$ are shown in the third column of Table 2.3 which shows that $(\widehat{l}, \widehat{r})$ obtained using the upper bounds match very well the (l^*, r^*) determined by Monte-Carlo simulations.

Next, $(\widetilde{l}, \widetilde{r})$ can be calculated from (2.57)–(2.61) by treating l and r as real values. Resulting $(\widetilde{l}, \widetilde{r})$ values are shown in the last column of Table 2.3. Note that $(\widehat{l}, \widehat{r})$ is the nearest one from $(\widetilde{l}, \widetilde{r})$ considering the possible redundancy allocation candidates in Table 2.1.

2.3.3 Redundancy Allocation: BDSC

A similar approach to redundancy allocation of the BDEC will be used for the BDSC. Instead of minimizing the upper bound on $P(\widehat{\mathbf{m}} \neq \mathbf{m})$, we will derive an estimate of $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ for the BDSC and the redundancy allocation (l, r) that minimizes this estimate of $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ will be used.

For the BDC and the BDEC, the number of unmasked defects after encoding failure (i.e.,

Table 2.3: Optimal Redundancy Allocations (l^*, r^*) and Their Estimates $(\widehat{l}, \widehat{r})$ and $(\widetilde{l}, \widetilde{r})$ of BDEC

Channel	(l^*, r^*)	$(\widehat{l}, \widehat{r})$	$(\widetilde{l}, \widetilde{r})$
1	(0, 100)	(0, 100)	(0, 100)
2	(30, 70)	(30, 70)	(28.3, 71.7)
3	(40, 60)	(40, 60)	(42.8, 57.2)
4	(50, 50)	(50, 50)	(50, 50)
5	(60, 40)	(60, 40)	(57.2, 42.8)
6	(70, 30)	(70, 30)	(71.7, 28.3)
7	(100, 0)	(100, 0)	(100, 0)

$\mathbf{x} \circ \mathbf{s} \neq \mathbf{c}$) does not affect $P(\widehat{\mathbf{m}} \neq \mathbf{m})$. On the other hand, the number of unmasked defects is important for $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ of the BDSC. The reason is that the stuck-at errors due to unmasked defects can be treated as random errors and corrected during the decoding stage. Hence, we propose a two-step encoding method (described in Algorithm 1), which reduces the performance gap between (2.32) and (2.33).

Algorithm 1 Two-step Encoding

Step 1: Try to solve (2.33), i.e., $G_0^{\mathcal{U}}\mathbf{p} = \mathbf{b}^{\mathcal{U}}$.
if $u < d^\star$ **then** go to **End**. ▷ A solution $\mathbf{p}$ always exists.
else ▷ A solution $\mathbf{p}$ exists so long as (2.34) holds.
  if $\mathbf{p}$ exists **then** go to **End**.
  else go to **Step 2**.
  end if
end if
Step 2:
- Choose $d^\star - 1$ locations among $\mathcal{U}$ and define $\mathcal{U}' = \{i_1, \ldots, i_{d^\star - 1}\}$.
- Solve the following linear equation: $G_0^{\mathcal{U}'}\mathbf{p} = \mathbf{b}^{\mathcal{U}'}$ ▷ A solution $\mathbf{p}$ always exists.

End

The two-step encoding tries to reduce the number of unmasked defects by using the second step (i.e., Step 2) even though $\mathbf{x} \circ \mathbf{s} \neq \mathbf{c}$. When the encoder fails to solve (2.33), the encoder randomly chooses $d^\star - 1$ defect locations among $\mathcal{U}$ and define $\mathcal{U}' = \{i_1, \ldots, i_{d^\star - 1}\}$. Afterwards, the encoder solves $G_0^{\mathcal{U}'}\mathbf{p} = \mathbf{b}^{\mathcal{U}'}$ where a solution $\mathbf{p}$ always exists by the definition of $d^\star$ in (2.25). If $\mathbf{p}$ is obtained in Step 2, then the number of unmasked defects is $u - (d^\star - 1)$ instead of u.

For the BDC and the BDEC, Step 2 cannot improve $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ because unmasked stuck-at defects cannot be corrected during the decoding stage. However, Step 2 is helpful for the BDSC because the stuck-at errors can be regarded as random errors and corrected at the decoder.

The two-step encoding's complexity is $\mathcal{O}(n^3)$ because both Step 1 and Step 2 are related to solving the linear equations. Also, the bounded distance decoding for estimating $\widehat{\mathbf{z}}$ can be implemented by polynomial decoding algorithms. For PBCH codes, standard algorithms such as Berlekamp-Massey algorithm can be used for decoding. The flow of PLBC's decoding for the BDSC was explained in Section 2.2.4.

We will derive the upper bound on $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ where the two-step encoding and the bounded distance decoding are used.

Theorem 2.18 *The upper bound on $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ is given by*

$$P(\widehat{\mathbf{m}} \neq \mathbf{m}) \leq \left[\sum_{u=d^\star}^{n} \left\{ \binom{n}{u} \beta^u (1-\beta)^{n-u} \min\left\{ \frac{\sum_{w=d^\star}^{u} B_{0,w} \binom{n-w}{u-w}}{\binom{n}{u}}, 1 \right\} \cdot \sum_{t=\widetilde{t}+d^\star-u}^{n} \binom{n}{t} p^t (1-p)^{n-t} \right\}\right] + \sum_{t=\widetilde{t}+1}^{n} \binom{n}{t} p^t (1-p)^{n-t} \tag{2.62}$$

where $\widetilde{t} = \left\lfloor \frac{\widetilde{d}-1}{2} \right\rfloor$ is the error correcting capability of $\mathcal{C}$.

Proof: The proof is given in Appendix 2.12. ■

During the derivation of the upper bound of (2.62), we regard all the unmasked stuck-at defects as random errors. However, on average, only half of the unmasked defects result in error if $P(S=0) = P(S=1) = \frac{\beta}{2}$. Thus, we can derive the following estimate of $P(\widehat{\mathbf{m}} \neq \mathbf{m})$.

Corollary 2.19 *The estimate of $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ is given by*

$$P(\widehat{\mathbf{m}} \neq \mathbf{m}) \simeq \left[\sum_{u=d^\star}^{n} \left\{ \binom{n}{u} \beta^u (1-\beta)^{n-u} \min\left\{ \frac{\sum_{w=d^\star}^{u} B_{0,w} \binom{n-w}{u-w}}{\binom{n}{u}}, 1 \right\} \cdot \sum_{t=\widetilde{t}-\left\lceil \frac{u-d^\star+1}{2} \right\rceil+1}^{n} \binom{n}{t} p^t (1-p)^{n-t} \right\}\right] + \sum_{t=\widetilde{t}+1}^{n} \binom{n}{t} p^t (1-p)^{n-t} \tag{2.63}$$

where $\lceil x \rceil$ is the smallest integer not less than x.

Proof: The proof is given in Appendix 2.12. ■

In order to compare the optimal redundancy (l^*, r^*) and the estimated redundancy allocation, we consider the several channels shown in Table 2.4. Channel 1 and Channel 7 are equivalent to the BSC and the BDC, respectively. For the other channels from Channel 2 to Channel 6, their lower bounds of the capacity $C_{\text{BDSC}}^{\text{lower}}$ of (2.19) and the upper bounds $C_{\text{BDSC}}^{\text{upper}}$ are almost the same. Thus, we can estimate $C_{\text{BDSC}}^{\text{enc}}$ from bounds although the closed-form of $C_{\text{BDSC}}^{\text{enc}}$ is not known.

It is worth mentioning that all the channel parameters are chosen to have almost the same $\widetilde{p} \simeq 4 \times 10^{-3}$, which was given by (2.22). Hence, all the channels show similar $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ for $(l, r) = (0, 100)$ which represents the case when the defect information is not used. On the other hand, each channel has different $C_{\text{BDSC}}^{\text{enc}}$. The larger β, the more defect information we can obtain, which results in the larger $C_{\text{BDSC}}^{\text{enc}}$. We apply $[n = 1023, k = 923, l]$ PBCH codes in Table 2.1.

Fig. 2.6 compares the simulation results and the estimates of $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ for the channels in Table 2.4, which shows that the estimates of $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ given by (2.63) match well with simulation results. Hence, we can choose the redundancy allocation minimizing the estimates instead of the simulation results in spite of the binomial approximation of (2.43). The redundancy allocation that minimizes the estimate of $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ is the estimate of the optimal redundancy allocation, i.e., $(\widehat{l}, \widehat{r})$. The simulation results of $P(\widehat{\mathbf{m}} \neq \mathbf{m}) < 10^{-8}$ are incomplete because of their impractical computational complexity.

Table 2.5 shows that the estimate of the optimal redundancy allocation $(\widehat{l}, \widehat{r})$ is the same as the optimal redundancy allocation (l^*, r^*) for the channels in Table 2.4. Thus, we can accurately estimate the optimal redundancy allocation without simulations. The estimate of optimal redundancy allocation requires much less computations than Monte-Carlo simulations.

Fig. 2.6 also shows that the optimal redundancy allocation significantly improves $P(\widehat{\mathbf{m}} \neq \mathbf{m})$. For example, $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ of channel 6 is 1.00×10^{-7} with the optimal redundancy allocation

Table 2.4: Several Channel Parameters of the BDSC

Channel	p	β	$C_{\text{BDSC}}^{\text{lower}}$	$C_{\text{BDSC}}^{\text{upper}}$
1	4.0×10^{-3}	0	0.9624	
2	3.0×10^{-3}	2.0×10^{-3}	0.9685	0.9686
3	2.5×10^{-3}	3.0×10^{-3}	0.9718	0.9719
4	2.0×10^{-3}	4.0×10^{-3}	0.9752	0.9753
5	1.0×10^{-3}	6.0×10^{-3}	0.9826	0.9827
6	5.0×10^{-4}	7.0×10^{-3}	0.9868	0.9868
7	0	8.0×10^{-3}	0.9920	

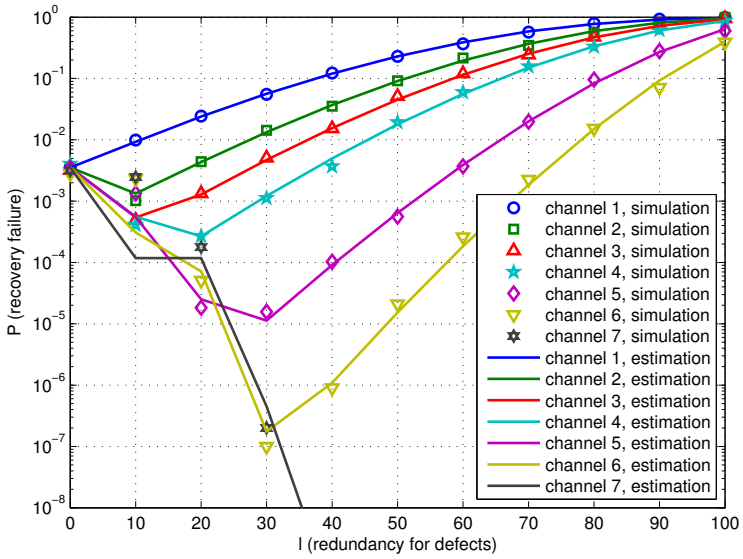


Figure 2.6: Comparison of simulation results and estimates of the probability of recovery failure $P(\widehat{\mathbf{m}} \neq \mathbf{m})$.

$(l^*, r^*) = (30, 70)$ whereas $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ is 2.80×10^{-3} with $(l, r) = (0, 100)$. Thus, it is important to find and use the optimal redundancy allocation for better $P(\widehat{\mathbf{m}} \neq \mathbf{m})$.

In addition, it is worth mentioning that $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ for (l^*, r^*) improves as β increases in Fig. 2.6. $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ of all the channels are the almost same for the redundancy allocation of $(l = 0, r = n - k)$ because all the channel parameters are chosen to have the same $\widetilde{C}_{\text{BDSC}}^{\min}$. As β increases, $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ for (l^*, r^*) improves because the PLBC can exploit more defect information as indicated by the lower and upper bounds on $C_{\text{BDSC}}^{\text{enc}}$ in Table 2.4.

Table 2.5: Optimal Redundancy Allocations (l^*, r^*) and Their Estimates $(\widehat{l}, \widehat{r})$ of BDSC by (2.63)

Channel	(l^*, r^*)	$(\widehat{l}, \widehat{r})$
1	(0, 100)	(0, 100)
2	(10, 90)	(10, 90)
3	(10, 90)	(10, 90)
4	(20, 80)	(20, 80)
5	(30, 70)	(30, 70)
6	(30, 70)	(30, 70)
7	(100, 0)	(100, 0)

2.4 Relations between BEC, BDC, BEQ, and WOM

2.4.1 Duality between BEC and BDC

We summarize the known facts of the BEC to show the dual relation between the BEC and the BDC. The codeword most likely to have been transmitted is the one that agrees with all of received bits that have not been erased. If there is more than one such codeword, the decoding may lead to a failure. Thus, the following simple coding scheme was proposed in [60].

Encoding: A message (information) $\mathbf{m} \in \mathbb{F}_2^k$ is encoded to a corresponding codeword $\mathbf{c} \in \mathcal{C}$ where $\mathcal{C} = \{\mathbf{c} \in \mathbb{F}_2^n \mid \mathbf{c} = G\mathbf{m}, \mathbf{m} \in \mathbb{F}_2^k\}$ where $\mathcal{C}$ is a set of codewords and the generator matrix is $G \in \mathbb{F}_2^{n\times k}$ such that $\mathrm{rank}(G) = k$.

Decoding: Let g denote the decoding rule. If the channel output $\mathbf{y}$ is identical to one and only one codeword on the unerased bits, the decoding succeeds. If $\mathbf{y}$ matches completely with more than one codeword on the unerased bits, the decoder chooses one of them randomly [60].

We will define a random variable D as follows.

$$D = \begin{cases} 0, & \mathbf{c} \neq \widehat{\mathbf{c}} \text{ (decoding failure);} \\ 1, & \mathbf{c} = \widehat{\mathbf{c}} \text{ (decoding success)} \end{cases} \tag{2.64}$$

where $\widehat{\mathbf{c}}$ is the estimated codeword produced by the decoding rule of g.

Elias showed that random codes of rates arbitrarily close to C_{BEC} can be decoded with an exponentially small error probability using the MAP decoding [60, 61, 62]. The MAP decoding rule of g can be achieved by solving the following linear equations [60]:

$$G^{\mathcal{V}}\widehat{\mathbf{m}} = \mathbf{y}^{\mathcal{V}} \tag{2.65}$$

where $\widehat{\mathbf{m}}$ is the estimate of $\mathbf{m}$ and $\mathcal{V} = \{j_1, \cdots, j_v\}$ indicates the locations of the v unerased bits. We use the notation of $\mathbf{y}^{\mathcal{V}} = (y_{j_1}, \cdots, y_{j_v})$ and $G^{\mathcal{V}} = \left[\mathbf{g}_{j_1}^T, \cdots, \mathbf{g}_{j_v}^T\right]^T$ where $\mathbf{g}_j$ is the j-th row of G. Note that $G^{\mathcal{V}} \in \mathbb{F}_2^{(n-e)\times k}$.

The decoding rule g can also be represented by the parity check matrix H instead of the generator matrix G as follows.

$$H^T\widehat{\mathbf{c}} = \left(H^{\mathcal{E}}\right)^T \widehat{\mathbf{c}}^{\mathcal{E}} + \left(H^{\mathcal{V}}\right)^T \widehat{\mathbf{c}}^{\mathcal{V}} = \mathbf{0} \tag{2.66}$$

where the parity check matrix H is an $n \times (n-k)$ matrix such that $H^T G = \mathbf{0}$. Also, $\mathcal{E} = \{i_1, \cdots, i_e\}$ indicates the locations of the e erased bits such that $\mathcal{E} \cup \mathcal{V} = [n]$ and $\mathcal{E} \cap \mathcal{V} = \emptyset$ (i.e., $n = e + v$). Note that $\widehat{\mathbf{c}}^{\mathcal{E}} = (\widehat{c}_{i_1}, \cdots, \widehat{c}_{i_e})$, $\widehat{\mathbf{c}}^{\mathcal{V}} = (\widehat{c}_{j_1}, \cdots, \widehat{c}_{j_v})$, $H^{\mathcal{E}} = \left[\mathbf{h}_{i_1}^T, \cdots, \mathbf{h}_{i_e}^T\right]^T$ and $H^{\mathcal{V}} = \left[\mathbf{h}_{j_1}^T, \cdots, \mathbf{h}_{j_v}^T\right]^T$ where $\mathbf{h}_i$ is the i-th row of H.

The decoder estimates the erased bits $\widehat{\mathbf{c}}^{\mathcal{E}}$ from the unerased bits $\widehat{\mathbf{c}}^{\mathcal{V}} = \mathbf{c}^{\mathcal{V}}$. Thus, (2.66) can be represented by the following linear equations:

$$\left(H^{\mathcal{E}}\right)^T \widehat{\mathbf{c}}^{\mathcal{E}} = \mathbf{q} \tag{2.67}$$

where $\mathbf{q} = \left(H^{\mathcal{V}}\right)^T \mathbf{c}^{\mathcal{V}}$ and $\left(H^{\mathcal{E}}\right)^T \in \mathbb{F}_2^{(n-k)\times e}$.

Remark 2.20 *In* (2.65) *and* (2.67)*, the number of equations is more than or equal to the number of unknowns. Usually, these systems of linear equations are* overdetermined*. The reason is that* $k \leq n - e$ *for correcting* e *erasures. Note that* $G^{\mathcal{V}} \in \mathbb{F}_2^{(n-e)\times k}$ *and* $\left(H^{\mathcal{E}}\right)^T \in \mathbb{F}_2^{(n-k)\times e}$*. Note that*

(2.65) *and* (2.67) *are* consistent *linear systems (i.e., there is at least one solution).*

The minimum distance d of $\mathcal{C}$ is given by

$$d = \min_{\substack{\mathbf{x}\neq\mathbf{0} \\ H^T\mathbf{x}=\mathbf{0}}} \|\mathbf{x}\| \tag{2.68}$$

which shows that any $d-1$ rows of H are linearly independent. So (2.67) has a unique solution when e is less than d.

The following Lemma has been known in coding theory community.

Lemma 2.21 *The upper bound on the probability of decoding failure of the MAP decoding rule is given by*

$$P\left(D=0 \mid |\mathcal{E}|=e\right) \leq \frac{\sum_{w=d}^{e} A_w \binom{n-w}{e-w}}{\binom{n}{e}} \tag{2.69}$$

where A_w is the weight distribution of $\mathcal{C}$.

Proof: The proof was well known, which can be found in [52]. ■

$P\left(D=0 \mid |\mathcal{E}|=e\right)$ can be obtained exactly for $d \leq e \leq d + \left\lfloor \frac{d-1}{2} \right\rfloor$ (where $\lfloor x \rfloor$ represents the largest integer not greater than x) as stated in the following Lemma.

Lemma 2.22 *[52] For $e \leq d+t$ where $t = \left\lfloor \frac{d-1}{2} \right\rfloor$, we can show that*

$$P\left(D=0 \mid |\mathcal{E}|=e\right) = \frac{1}{2} \cdot \frac{\sum_{w=d}^{e} A_w \binom{n-w}{e-w}}{\binom{n}{e}}. \tag{2.70}$$

From the definition of d in (2.68), Lemma 2.21 and Lemma 2.22, we can state the following.

Theorem 2.23 *[52] $P\left(D=0 \mid |\mathcal{E}|=e\right)$ is given by*

$$\begin{cases} 0 & \text{for } e < d, \quad (2.71) \\ \frac{1}{2} \cdot \frac{\sum_{w=d}^{e} A_w \binom{n-w}{e-w}}{\binom{n}{e}} & \text{for } d \leq e \leq d+t, \quad (2.72) \\ \leq \frac{\sum_{w=d}^{e} A_w \binom{n-w}{e-w}}{\binom{n}{e}} & \text{for } e > d+t. \quad (2.73) \end{cases}$$

Table 2.6: Duality between BEC and BDC

	BEC	BDC
Channel property	Ternary output $Y \in \{0, 1, *\}$ (erasure $*$ is neither "0" nor "1")	Ternary state $S \in \{0, 1, \lambda\}$ (defect is either "0" or "1")
Capacity	$C_{\text{BEC}} = 1 - \alpha$	$C_{\text{BDC}} = 1 - \beta$
Channel state information	Locations	Locations and stuck-at values
Correcting / Masking	Decoder corrects erasures	Encoder masks defects
MAP decoding / MDE	$G^{\mathcal{V}}\widehat{\mathbf{m}} = \mathbf{y}^{\mathcal{V}}$ (2.65) (Overdetermined)	$G_0^{\mathcal{U}}\mathbf{p} = \mathbf{b}^{\mathcal{U}}$ (2.33) (Underdetermined)
Solutions	$\widehat{\mathbf{m}}$ (estimate of message)	$\mathbf{p}$ (parity)
Minimum distance	$d = \min\{\|\mathbf{x}\| : H^T\mathbf{x} = \mathbf{0}, \mathbf{x} \neq \mathbf{0}\}$ If $e < d$, e erasures are corrected.	$d^\star = \min\{\|\mathbf{x}\| : G_0^T\mathbf{x} = \mathbf{0}, \mathbf{x} \neq \mathbf{0}\}$ If $u < d^\star$, u defects are masked.
Upper bounds on probability of failure	Theorem 2.23	Theorem 2.10
Probability of failure	If $H = G_0$ and $\alpha = \beta$, then $P(D = 0) = P(E = 0)$ (Theorem 2.24)	

We will discuss the duality between erasures and defects summarized in Table 2.6. In the BEC, the channel input $X \in \{0, 1\}$ is binary and the channel output $Y = \{0, 1, *\}$ is ternary where the erasure $*$ is *neither* 0 nor 1. In the BDC, the channel state $S \in \{0, 1, \lambda\}$ is ternary whereas the channel input and output are binary. The ternary channel state S informs whether the given cells are stuck-at defects or normal cells. The stuck-at value is *either* 0 or 1.

The expressions for capacities of both channels are quite similar as shown in $C_{\text{BEC}} = 1 - \alpha$ and $C_{\text{BDC}} = 1 - \beta$. In the BEC, the *decoder* corrects erasures by using the information of locations of erasures, whereas the *encoder* masks the defects by using the information of defect locations and stuck-at values in the BDC.

The capacity achieving scheme of the BEC can be represented by the linear equations based on the generator matrix G of (2.65) or the linear equations based on the parity check matrix H of (2.67). Both linear equations are usually *overdetermined* as discussed in Remark 2.20. The solution of linear equations based on G is the *estimate of message* $\widehat{\mathbf{m}}$ and there should be only one $\widehat{\mathbf{m}}$ for decoding success.

On the other hand, the capacity achieving scheme of the BDC can be described by the linear equation which are usually *underdetermined* as explained in 2.33. The additive encoding can be represented by the linear equations based on the generator matrix G_0 of (2.33) whose solution is the *parity* $\mathbf{p}$.

We can see the duality between erasures and defects by comparing the solution $\widehat{\mathbf{m}}$ of (2.65) and the solution $\mathbf{p}$ of (2.33), i.e., *message* and *parity*. Note that coding schemes of (2.65) and (2.33) are based on the generator matrix.

In the BEC, the minimum distance d is defined by the *parity check matrix* H, whereas the minimum distance $d^\star$ of the BDC is defined by the *generator matrix* G_0. The upper bound on the probability of decoding failure is dependent on the weight distribution of $\mathcal{C}$ (i.e., A_w), whereas the upper bound on the probability of encoding failure is dependent on the weight distribution of $\mathcal{C}_0^\perp$ (i.e., B_w).

If $A_w = B_w$ and $e = u$, it is clear that the upper bound on $P\left(D = 0 \mid |\mathcal{E}| = e\right)$ is same as the upper bound on $P\left(E = 0 \mid |\mathcal{U}| = u\right)$ by Theorem 2.23 and Theorem 2.10. In particular, the following Theorem shows the equivalence of the failure probabilities (i.e., the probability of decoding failure of erasures and the probability of encoding failure of defects).

Theorem 2.24 *[52] If $H = G_0$ and $\alpha = \beta$, then the probability of decoding failure of MAP decoding for the BEC is the same as the probability of encoding failure of MDE for the BDC (i.e., $P(D = 0) = P(E = 0)$). The computational complexities for both cases are $\mathcal{O}(n^3)$.*

Proof: If $\alpha = \beta$, then it is clear that that $P(\mathcal{E}) = P(\mathcal{U})$ for $\mathcal{E} = \mathcal{U}$. If $\mathcal{E} = \mathcal{U}$ and $H = G_0$, then $H^{\mathcal{E}} = G_0^{\mathcal{U}}$. If $H^{\mathcal{E}}$ and $G_0^{\mathcal{U}}$ are full rank, then it is clear that $P(D = 0) = P(E = 0) = 0$.

Suppose that $\mathrm{rank}(H^{\mathcal{E}}) = \mathrm{rank}(G_0^{\mathcal{U}}) = e - j$ where $\mathcal{E} = \mathcal{U}$ (i.e., $e = u$). For the BEC, there are 2^j codewords that satisfy (2.67) and the decoder chooses one codeword among them randomly. Hence, $P(D = 0 \mid \mathcal{E}) = 1 - \frac{1}{2^j}$.

For the BDC, each element of $\mathbf{b}^{\mathcal{U}}$ in (2.35) is uniform since $P(S = 0 \mid S \neq \lambda) = P(S = 1 \mid S \neq \lambda) = \frac{1}{2}$. (2.35) has at least one solution if and only if $\mathrm{rank}(G_0^{\mathcal{U}}) = \mathrm{rank}(G_0^{\mathcal{U}} \mid \mathbf{b}^{\mathcal{U}})$.

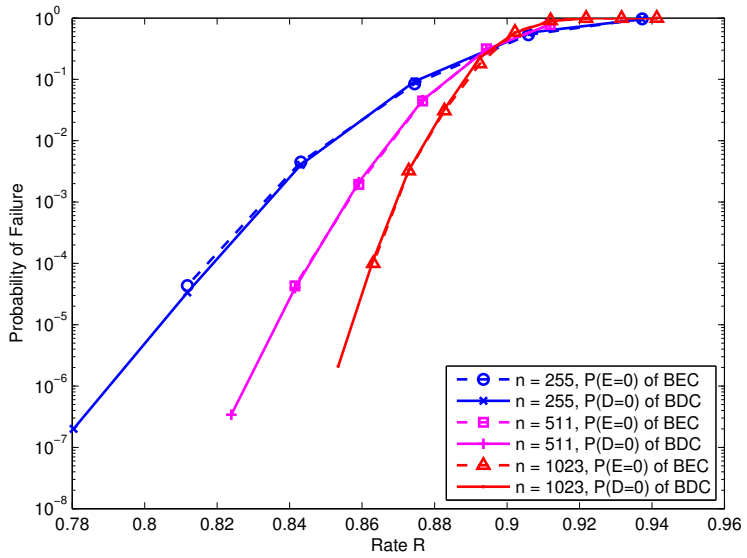


Figure 2.7: Probability of failure, i.e., $P(D = 0)$ of the BEC with $\alpha = 0.1$ and $P(E = 0)$ of the BDC with $\beta = 0.1$.

In order to satisfy this condition, the last j elements of $\mathbf{b}^{\mathcal{U}}$ should be zeros, which means that $P(E = 0 \mid \mathcal{U}) = 1 - \frac{1}{2^j}$. Thus, $P(D = 0 \mid \mathcal{E}) = P(E = 0 \mid \mathcal{U})$ if $\mathcal{E} = \mathcal{U}$ and $H = G_0$.

Since $P(\mathcal{E}) = P(\mathcal{U})$ and $P(D = 0 \mid \mathcal{E}) = P(E = 0 \mid \mathcal{U})$ for $\mathcal{E} = \mathcal{U}$, it is true that $P(D = 0) = P(E = 0)$. ■

Fig. 2.7 compares $P(D = 0)$ of the BEC and $P(E = 0)$ of the BDC when $H = G_0$ and $\alpha = \beta$. The parity check matrices of Bose-Chaudhuri-Hocquenghem (BCH) codes are used for H and G_0. Hence, BCH codes are used for the BEC and the duals of BCH codes are used for the BDC. The numerical results in Fig. 2.7 show that $P(D = 0) = P(E = 0)$ if $H = G_0$ and $\alpha = \beta$, which supports Theorem 2.24.

Recently, it was proved that a sequence of linear codes achieves C_{BEC} under MAP decoding if its blocklengths are strictly increasing, its code rates converge to some $\delta \in (0, 1)$, and the permutation group of each code is doubly transitive [53]. Hence, RM codes and BCH codes can achieve C_{BEC} under MAP decoding. Based on the duality between the BEC and the BDC, we can claim the following Corollary.

Corollary 2.25 *RM codes achieve C_{BDC} with computational complexity $\mathcal{O}(n^3)$.*

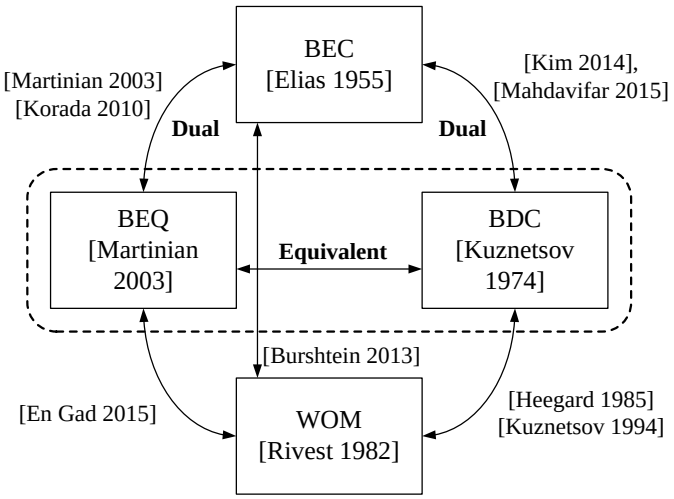


Figure 2.8: Relations between BEC, BDC, BEQ, and WOM.

Proof: In [53], it was shown that RM codes achieve C_{BEC} under MAP decoding whose computational complexity is $\mathcal{O}(n^3)$. By Theorem 2.24, the duals of RM codes achieve C_{BDC} with $\mathcal{O}(n^3)$. Since the duals of RM codes are also RM codes [59, pp. 375–376], RM codes achieve C_{BDC}. ■

We note that the duals of BCH codes also achieve C_{BDC} by the same reason.

In this section, we have demonstrated the duality between the BEC and the BDC from channel properties, capacities, capacity-achieving schemes, and their failure probabilities. This duality implies that the existing code constructions and algorithms of the BEC can be applied to the BDC and *vice versa*.

2.4.2 Relations between BEC, BDC, BEQ, and WOM

We extend the duality between the BEC and the BDC to other interesting models such as binary erasure quantization (BEQ) and write-once memory (WOM) codes. We review the literature on these models and describe the relations between BEC, BDC, BEQ, and WOM.

Martinian and Yedidia [54] considered BEQ problems where the source vector consists of $\{0, 1, *\}$ ($*$ denotes an erasure). Neither ones nor zeros may be changed, but erasures may

be quantized to either zero or one. The erasures do not affect the distortion regardless of the value they are assigned since erasures represents source samples which are missing, irrelevant, or corrupted by noise. The BEQ problem with erasure probability α can be formulated as follows.

$$P(S = s) = \begin{cases} \alpha, & \text{if } s = *; \\ \frac{1-\alpha}{2}, & \text{if } s = 0 \text{ or } 1 \end{cases} \tag{2.74}$$

and the Hamming distortion $d_H(\cdot, \cdot)$ is given by

$$d_H(0, *) = d_H(1, *) = 0, \quad d_H(0, 1) = 1. \tag{2.75}$$

The rate-distortion bound with zero distortion is given by

$$R_{\text{BEQ}} = 1 - \alpha. \tag{2.76}$$

In [54] the duality between the BEC and the BEQ was observed, and the authors showed that low-density generator matrix (LDGM) codes (i.e., the duals of LDPC codes) can achieve the R_{BEQ} by modified message-passing algorithm. The computational complexity is $\mathcal{O}(nd_{\mathcal{G}})$ where $d_{\mathcal{G}}$ denotes the maximum degree of the bipartite graph $\mathcal{G}$ of the low-density generator matrix. In [63], it was shown that polar codes with an successive cancellation encoder can achieve R_{BEQ} with $\mathcal{O}(n \log n)$.

From (2.74)–(2.76), we can claim that the BEQ with erasure probability α is equivalent to the BDC with defect probability β if $\beta = 1 - \alpha$. We can observe that $s = *$ of the BEQ corresponds to $s = \lambda$ of the BDC, which represents normal cells by comparing (2.6) and (2.74). Also, $s = 0$ and $s = 1$ of the BEQ can be regarded as stuck-at 0 defects and stuck-at 1 defects, respectively. In addition, $R_{\text{BEQ}} = C_{\text{BEC}} = 1 - C_{\text{BDC}}$.

Since the BEQ is equivalent to the BDC, we can claim that RM codes achieve R_{BEQ} due to Corollary 2.25. Hence, LDGM codes (duals of LDPC codes), polar codes, and RM codes achieve

R_{BEQ}.

Inversely, the coding scheme for the BEQ can be applied to the BDC. Thus, C_{BDC} can be achieved by LDGM codes and polar codes whose complexities are $\mathcal{O}(nd_{\mathcal{G}})$ and $\mathcal{O}(n \log n)$ respectively. It is important because the best known encoding complexity of capacity achieving scheme for the BDC was $\mathcal{O}(n \log^2 n)$ in [41]. Also, note that the encoding complexity of coding schemes in [42] is $\mathcal{O}(n^3)$.

The model of WOM was proposed for data storage devices where once a one is written on a memory cell, this cell becomes permanently associated with a one. Hence, the ability to rewrite information in these memory cells is constrained by the existence of previously written ones [43, 55]. Recently, the WOM model has received renewed attention as a possible channel model for flash memories [64, 65].

In [43], it was noted that WOM are related to the BDC since the cells storing ones can be considered as stuck-at 1 defects. Moreover, Kuznetsov and Han Vinck [44] showed that additive encoding for the BDC can be used to achieve the capacity of WOM. Burshtein and Strugatski [66] proposed a capacity-achieving coding scheme for WOM with $\mathcal{O}(n \log n)$ complexity, which is based on polar codes and successive cancellation encoding [63]. Recently, En Gad *et al.* [67] related the WOM to the BEQ. Hence, LDGM codes and message-passing algorithm in [54] can be used for WOM. Note that the encoding complexity is $\mathcal{O}(nd_{\mathcal{G}})$.

Fig. 2.8 illustrates the relations between BEC, BDC, BEQ, and WOM. We emphasize that a coding scheme for one model can be applied to other models based on these relations. It is worth mentioning that RM codes, LDPC (or LDGM) codes, and polar codes can achieve the capacities of all these models. Their computational complexities are $\mathcal{O}(n^3)$, $\mathcal{O}(nd_{\mathcal{G}})$, and $\mathcal{O}(n \log n)$, respectively.

2.5 Conclusion

The channel coding for memory with stuck-at defects and redundancy allocation problem for memory with permanent stuck-at defects and transient errors were investigated. We derived the upper bound on the probability of recovery failure for the BDEC and the estimate of the probability of recovery failure for the BDSC. Based on these analytical results, we can efficiently estimate the optimal redundancy allocation. The estimated redundancy allocation matches the optimal redundancy allocation well while requiring much less computation than Monte-Carlo simulations.

In addition, the duality between the BEC and the BDC was investigated. Based on this duality, we showed that RM codes and duals of BCH codes achieve the capacity of the BDC. This duality can be extended to the relations between BEC, BDC, BEQ, and WOM. Based on these relations, we showed that RM codes achieve the capacity of the BDC with $\mathcal{O}(n^3)$ and LDGM codes (duals of LDPC codes) achieve the capacity with $\mathcal{O}(nd_{\mathcal{G}})$. Also, polar codes can achieve the capacity with $\mathcal{O}(n \log n)$ complexity, which beats the best known result of $\mathcal{O}(n \log^2 n)$.

2.6 Proof of Proposition 2.4

By Gelfand-Pinsker theorem [32, 34],

$$C_{\text{BDEC}}^{\text{enc}} = \max_{P(U|S),X(U,S)} (I(U;Y) - I(U;S))$$

where $|\mathcal{U}| \leq \min\{|\mathcal{X}| \cdot |\mathcal{S}|, |\mathcal{Y}| + |\mathcal{S}| - 1\}$. It is clear that

$$I(U;Y) - I(U;S) = H(U \mid S) - H(U \mid Y). \tag{2.77}$$

For $S = \lambda$, set $U \sim \text{Bern}(1/2)$ and $X = U$, i.e., U is a Bernoulli random variable with parameter $\frac{1}{2}$. If $S \neq \lambda$, we set $U = X = S$. Then,

$$H(U \mid S) = P(S = \lambda)H(X \mid S = \lambda) + P(S \neq \lambda)H(X = S \mid S \neq \lambda) = 1 - \beta. \tag{2.78}$$

In addition,

$$H(U \mid Y) = H(X \mid Y) = H(X) - I(X;Y) = 1 - (1 - \alpha) = \alpha \tag{2.79}$$

where $H(X) = 1$ follows from $P(X = 0) = P(X = 1)$ for both stuck-at defects and normal cells. We should minimize $H(U \mid Y)$ in order to maximize (2.77), which can be achieved by setting $I(X;Y) = 1 - \alpha$ (i.e., the capacity of the BEC). By combining (2.78) and (2.79), $C_{\text{BDEC}}^{\text{enc}} = 1 - \alpha - \beta$.

2.7 Proof of Proposition 2.5

The upper bound of (2.18) is easy to see because of (2.14) where we assume that the stuck-at defects do not suffer from random errors.

Unlike the BDEC, the unmasked defects in the BDSC can be corrected by the decoder. Hence, we can allow the encoder to mask a fraction of stuck-at defects. Suppose that η denotes the fraction of unmasked stuck-at defects during encoding. We need to find the optimal η^*, which makes it complicated to derive the capacity in [40].

By setting $\eta = 0$ instead of using the optimal η^*, the lower bound of (2.19) can be derived, which is similar to the proof of Proposition 2.4. For $S = \lambda$, set $U \sim \text{Bern}(1/2)$ and $X = U$. For $S \neq \lambda$, we set $U = X = S$ which means that $\eta = 0$. Then,

$$H(U \mid S) = 1 - \beta \tag{2.80}$$

which is the same as (2.78). In addition,

$$H(U \mid Y) = H(X \mid Y) = H(X) - I(X;Y) = 1 - (1 - h(p)) = h(p) \tag{2.81}$$

which follows from the similar reason of (2.79). By combining (2.80) and (2.81), $C_{\text{BDSC}}^{\text{lower}} = 1 - \beta - h(p)$.

2.8 Proof of Lemma 2.8

First, we will show that

$$P\left(E = 0 \mid U = u\right) = \sum_{j=1}^{u} \left(1 - \frac{1}{2^j}\right) P\left(\text{rank}\left(G_0^{\mathcal{U}}\right) = u - j \mid U = u\right). \tag{2.82}$$

If $\text{rank}\left(G_0^{\mathcal{U}}\right) = u$, (2.33) has at least one solution since (2.34) holds, i.e., $P\left(E = 0 \mid U = u\right) = 0$. If $\text{rank}\left(G_0^{\mathcal{U}}\right) = u - j$ for $1 \leq j \leq u$, the last j rows of the row reduced echelon form of $G_0^{\mathcal{U}}$ are zero vectors. In order to satisfy (2.34), the last j elements of the column vector $\mathbf{b}^{\mathcal{U}}$ should also be zeros. The probability that the last j elements of $\mathbf{b}^{\mathcal{U}}$ are zeros is $\frac{1}{2^j}$ since $P(S = 0 \mid S \neq \lambda) = P(S = 1 \mid S \neq \lambda) = \frac{1}{2}$. Thus, $P\left(E = 0 \mid U = u\right)$ is given by (2.82).

By (2.82), we can claim that

$$\frac{P\left(\text{rank}\left(G_0^{\mathcal{U}}\right) < u \mid U = u\right)}{2} \leq P\left(E = 0 \mid U = u\right) \leq P\left(\text{rank}\left(G_0^{\mathcal{U}}\right) < u \mid U = u\right). \tag{2.83}$$

Suppose that there exists a nonzero codeword $\mathbf{c}^{\perp} \in \mathcal{C}_0^{\perp}$ of Hamming weight w. Note that G_0 is the parity check matrix of $\mathcal{C}_0^{\perp}$. Let $\Psi_w(\mathbf{c}^{\perp}) = \left\{i \mid c_i^{\perp} \neq 0\right\}$ denote the locations of nonzero elements of $\mathbf{c}^{\perp}$ and $\mathcal{U} = \{i_1, \ldots, i_u\}$ denote the locations of u defects.

If $\Psi_w(\mathbf{c}^{\perp}) \subseteq \mathcal{U}$, $\text{rank}\left(G_0^{\mathcal{U}}\right) < u$. Note that $G_0^{\Psi_w(\mathbf{c}^{\perp})}$ is a submatrix of $G_0^{\mathcal{U}}$ and the rows of $G_0^{\Psi_w(\mathbf{c}^{\perp})}$ are linearly dependent since $G_0^T \mathbf{c}^{\perp} = \mathbf{0}$.

For any $\mathbf{c}^{\perp}$ such that $\Psi_w(\mathbf{c}^{\perp}) \subseteq \mathcal{U}$, the number of possible $\mathcal{U}$ is $\binom{n-w}{u-w}$. Due to dou-

ble counting, the number of $\mathcal{U}$ which results in $\operatorname{rank}\left(G_0^{\mathcal{U}}\right) < u$ will be less than or equal to $\sum_{w=d^\star}^{u} B_{0,w}\binom{n-w}{u-w}$. Since the number of all possible $\mathcal{U}$ such that $U = u$ is $\binom{n}{u}$,

$$P\left(\operatorname{rank}\left(G_0^{\mathcal{U}}\right) < u \mid U = u\right) \leq \frac{\sum_{w=d^\star}^{u} B_{0,w}\binom{n-w}{u-w}}{\binom{n}{u}}. \tag{2.84}$$

From (2.83) and (2.84), the upper bound on $P\left(E = 0 \mid U = u\right)$ is given by (2.37). Note that we set $P\left(E = 0 \mid U = u\right) = 1$ if $\frac{\sum_{w=d^\star}^{u} B_{0,w}\binom{n-w}{u-w}}{\binom{n}{u}} \geq 1$.

2.9 Proof of Lemma 2.9

The proof has two parts. First, we will show that

$$P\left(\operatorname{rank}\left(G_0^{\mathcal{U}}\right) < u \mid U = u\right) = \frac{\sum_{w=d^\star}^{u} B_{0,w}\binom{n-w}{u-w}}{\binom{n}{u}} \tag{2.85}$$

for $u \leq d^\star + t^\star$ where $t^\star = \left\lfloor \frac{d^\star - 1}{2} \right\rfloor$, which means that there is no double counting in (2.84). Second, we will prove that

$$P\left(\operatorname{rank}\left(G_0^{\mathcal{U}}\right) < u \mid U = u\right) = P\left(\operatorname{rank}\left(G_0^{\mathcal{U}}\right) = u - 1 \mid U = u\right) \tag{2.86}$$

for $u \leq d^\star + t^\star$, which means that $P\left(\operatorname{rank}\left(G_0^{\mathcal{U}}\right) \leq u - 2 \mid U = u\right) = 0$.

Then, $P\left(E = 0 \mid U = u\right)$ is given by

$$P\left(E = 0 \mid U = u\right) = \frac{1}{2} \cdot P\left(\operatorname{rank}\left(G_0^{\mathcal{U}}\right) = u - 1 \mid U = u\right) \tag{2.87}$$

$$= \frac{1}{2} \cdot \frac{\sum_{w=d^\star}^{u} B_{0,w}\binom{n-w}{u-w}}{\binom{n}{u}} \tag{2.88}$$

where (2.87) follows from (2.82) and (2.86). Also, (2.88) follows from (2.85).

1) Proof of (2.85)

Suppose that there are two nonzero codewords $\mathbf{c}_1^\perp, \mathbf{c}_2^\perp \in \mathcal{C}_0^\perp$ such that $\|\mathbf{c}_1^\perp\| = w_1$ and

$\|\mathbf{c}_2^\perp\| = w_2$. Without loss of generality, assume that $d^\star \leq w_1 \leq w_2$. The locations of nonzero elements in $\mathbf{c}_1^\perp$ and $\mathbf{c}_2^\perp$ are given by

$$\Psi_{w_1}\left(\mathbf{c}_1^\perp\right) = \{i_{1,1}, \ldots, i_{1,w_1}\}, \quad \Psi_{w_2}\left(\mathbf{c}_2^\perp\right) = \{i_{2,1}, \ldots, i_{2,w_2}\}.$$

Let $\Psi_\alpha = \{i_1, \ldots, i_\alpha\}$ denote $\Psi_\alpha = \Psi_{w_1}\left(\mathbf{c}_1^\perp\right) \cap \Psi_{w_2}\left(\mathbf{c}_2^\perp\right)$ where

$$\Psi_{w_1}\left(\mathbf{c}_1^\perp\right) = \Psi_\alpha \cup \left\{i'_{1,1}, \ldots, i'_{1,\beta_1}\right\}, \quad \Psi_{w_2}\left(\mathbf{c}_2^\perp\right) = \Psi_\alpha \cup \left\{i'_{2,1}, \ldots, i'_{2,\beta_2}\right\}$$

where i'_{1,j_1} for $j_1 \in \{1, \ldots, \beta_1\}$ and i'_{2,j_2} for $j_2 \in \{1, \ldots, \beta_2\}$ are the reindexed locations of nonzero elements of $\mathbf{c}_1^\perp$ and $\mathbf{c}_2^\perp$ that are mutually disjoint with Ψ_α. Note that $\left\{i'_{1,1}, \ldots, i'_{1,\beta_1}\right\} \cap \left\{i'_{2,1}, \ldots, i'_{2,\beta_2}\right\} = \emptyset$, $\beta_1 = w_1 - \alpha$ and $\beta_2 = w_2 - \alpha$.

Due to the property of linear codes, $\mathbf{c}_3^\perp = \mathbf{c}_1^\perp + \mathbf{c}_2^\perp$ is also a codeword of $\mathcal{C}_0^\perp$, i.e., $\mathbf{c}_3^\perp \in \mathcal{C}_0^\perp$ and $\|\mathbf{c}_3^\perp\| = \beta_1 + \beta_2$. Also, the following conditions should hold because of the definition of $d^\star$.

$$\alpha + \beta_1 \geq d^\star, \quad \alpha + \beta_2 \geq d^\star, \quad \beta_1 + \beta_2 \geq d^\star$$

Thus, we can claim that $2(\alpha + \beta_1 + \beta_2) \geq 3d^\star$, which results in $\alpha + \beta_1 + \beta_2 \geq d^\star + \left\lfloor \frac{d^\star+1}{2} \right\rfloor = d^\star + t^\star + 1$ since $\alpha + \beta_1 + \beta_2$ has to be an integer.

For double counting to occur in (2.84), there should exist at least two codewords $\mathbf{c}_1^\perp$ and $\mathbf{c}_2^\perp$ such that $\Psi_{w_1}\left(\mathbf{c}_1^\perp\right) \cup \Psi_{w_2}\left(\mathbf{c}_2^\perp\right) \subseteq \mathcal{U}$. It means that double counting occurs only if $u \geq \alpha + \beta_1 + \beta_2 \geq d^\star + t^\star + 1$. Thus, there is no double counting for $u \leq d^\star + t^\star$. For $u \leq d^\star + t^\star$, there exists at most one codeword $\mathbf{c}^\perp$ such that $\Psi_w\left(\mathbf{c}^\perp\right) \subseteq \mathcal{U}$.

2) Proof of (2.86)

It is clear that $\mathrm{rank}\left(G_0^{\mathcal{U}}\right) = u - 1$ if and only if there exists only one nonzero codeword $\mathbf{c}^\perp$ such that $\Psi_w\left(\mathbf{c}^\perp\right) \subseteq \mathcal{U}$. Note that $\mathrm{rank}\left(G_0^{\mathcal{U}}\right) < u - 1$ if and only if $\mathcal{U}$ includes the locations of nonzero elements of at least two nonzero codewords. We have already shown that there exists at most one nonzero codeword $\mathbf{c}^\perp$ such that $\Psi_w\left(\mathbf{c}^\perp\right) \subseteq \mathcal{U}$ for $u \leq d^\star + t^\star$.

2.10 Proof of Theorem 2.15

From $P(\widehat{\mathbf{m}} \neq \mathbf{m}) = P(E=0) + P(E=1, D=0)$, we derive the upper bounds on $P(E=0)$ and $P(E=1, D=0)$ respectively. The upper bounds on $P(E=0)$ was shown in Corollary 2.11 and 2.12.

If the additive encoding succeeds (i.e., $E=1$), then the corresponding channel is equivalent to the BEC with the erasure probability α. The upper bound on $P(E=1, D=0)$ is given by

$$P(E=1, D=0) \leq \sum_{e=\widetilde{d}}^{n} \alpha^e (1-\alpha)^{n-e} \sum_{w=\widetilde{d}}^{e} A_w \binom{n-w}{e-w} \tag{2.89}$$

$$= 2^{-r} \sum_{e=\widetilde{d}}^{n} \alpha^e (1-\alpha)^{n-e} \sum_{w=\widetilde{d}}^{e} \binom{n}{w} \binom{n-w}{e-w} \tag{2.90}$$

$$= 2^{-r} (1+\alpha)^n \tag{2.91}$$

where (2.89) follows from $P(D=0 \mid |\mathcal{E}|=e) \leq \frac{\sum_{w=d_1}^{e} A_w \binom{n-w}{e-w}}{\binom{n}{e}}$ for the BEC, which can be derived in a similar way as for Lemma 2.8. If $A_w = 2^{-r}\binom{n}{w}$, (2.91) can be derived by a similar way as for Corollary 2.12.

2.11 Proof of Corollary 2.16

Suppose that l and r are real values. Since the objective function is convex and other constraints are linear, the optimization problem of (2.56) is convex. The Lagrangian L is given by

$$\begin{aligned} L(l, r, \lambda_1, \lambda_2, \lambda_3, \lambda_4, \nu) = {} & 2^{-l}(1+\beta)^n + 2^{-r}(1+\alpha)^n + \lambda_1(-l) + \lambda_2(-r) \\ & + \lambda_3\{l-(n-k)\} + \lambda_4\{r-(n-k)\} + \nu\{l+r-(n-k)\} \end{aligned} \tag{2.92}$$

where λ_i for $i=1,2,3,4$ are the Lagrange multipliers associated with the inequality constraints and ν is the Lagrange multiplier with the equality constraint.

The KKT conditions can be derived as follows.

$$\nabla L = \begin{bmatrix} -\ln 2 \cdot 2^{-l} (1+\beta)^n \\ -\ln 2 \cdot 2^{-r} (1+\alpha)^n \end{bmatrix} + \begin{bmatrix} -\lambda_1 + \lambda_3 + \nu \\ -\lambda_2 + \lambda_4 + \nu \end{bmatrix} = 0 \tag{2.93}$$

$$-l \leq 0, \quad -r \leq 0, \quad l - (n-k) \leq 0, \quad r - (n-k) \leq 0$$

$$l + r - (n-k) = 0 \tag{2.94}$$

$$\lambda_i \geq 0, \quad i = 1, \ldots, 4 \tag{2.95}$$

$$\lambda_1 l = 0, \quad \lambda_2 r = 0 \tag{2.96}$$

$$\lambda_3 \{l - (n-k)\} = 0, \quad \lambda_4 \{r - (n-k)\} = 0 \tag{2.97}$$

If α is much greater than β such that $\frac{1+\alpha}{1+\beta} > 2^{1-R}$, then we can claim that $2^{-l}(1+\beta)^n < 2^{-r}(1+\alpha)^n$ for any (l, r) where $l + r = n - k$. By (2.93), it is true that

$$2^{-r}(1+\alpha)^n - 2^{-l}(1+\beta)^n = \lambda_1' - \lambda_2' - \lambda_3' + \lambda_4' > 0 \tag{2.98}$$

where $\lambda_i' = \frac{\lambda_i}{\ln 2}$. Thus,

$$\lambda_1 + \lambda_4 > \lambda_2 + \lambda_3 \geq 0. \tag{2.99}$$

It is clear that $(l, r) = (0, n-k)$ satisfies the KKT conditions since $\lambda_2 = \lambda_3 = 0$ due to complement slackness. If β is much greater than α such that $\frac{1+\alpha}{1+\beta} < 2^{-(1-R)}$, then it can be shown that $(l, r) = (n-k, 0)$ satisfies the KKT conditions similarly.

Otherwise, suppose that $0 < l < n-k$ and $0 < r < n-k$. Due to complementary slackness, $\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = 0$. Thus, (2.93) will be as follows.

$$-\ln 2 \cdot 2^{-l}(1+\beta)^n + \nu = 0 \tag{2.100}$$

$$-\ln 2 \cdot 2^{-r}(1+\alpha)^n + \nu = 0 \tag{2.101}$$

By (2.100) and (2.101),

$$2^{-l}\left(1+\beta\right)^{n}=2^{-r}\left(1+\alpha\right)^{n}. \tag{2.102}$$

Then, we can derive (2.60) and (2.61) by (2.94) and (2.102). It is clear that (2.60) and (2.61) satisfy the KKT conditions.

2.12 Proof of Theorem 2.18 and Corollary 2.19

The probability of recovery failure $P\left(\widehat{\mathbf{m}}\neq\mathbf{m}\right)$ is given by

$$P\left(\widehat{\mathbf{m}}\neq\mathbf{m}\right)=P(E=0,\widehat{\mathbf{m}}\neq\mathbf{m})+P(E=1,\widehat{\mathbf{m}}\neq\mathbf{m}).$$

Note that the stuck-at errors due to unmasked defects can be corrected at the decoder even if the encoding fails. First, we will derive the upper bound on $P(E=0,\widehat{\mathbf{m}}\neq\mathbf{m})$. By the chain rule, $P(E=0,\widehat{\mathbf{m}}\neq\mathbf{m})$ is given by

$$P(E=0,\widehat{\mathbf{m}}\neq\mathbf{m})=\sum_{u=1}^{n}P(U=u)P(E=0\mid U=u)P(\widehat{\mathbf{m}}\neq\mathbf{m}\mid E=0,U=u)$$

where $P(U=u)=\binom{n}{u}\beta^{u}\left(1-\beta\right)^{n-u}$. In addition, the upper bound on $P(E=0|U=u)$ is given by Lemma 2.8. Also, $P(\widehat{\mathbf{m}}\neq\mathbf{m}|E=0,U=u)$ is given by

$$P(\widehat{\mathbf{m}}\neq\mathbf{m}|E=0,U=u)\leq P\left(\{u-(d^{\star}-1)\}+t>\widetilde{t}\right)=P\left(t\geq\widetilde{t}-u+d^{\star}\right). \tag{2.103}$$

where t is the number of random errors. Note that $u-(d^{\star}-1)$ represents the number of unmasked defects when two-step encoding fails. Since the number of random errors can be modeled by the binomial random variable, $P(\widehat{\mathbf{m}}\neq\mathbf{m}|E=0,U=u)$ is given by

$$P\left(\widehat{\mathbf{m}}\neq\mathbf{m}|E=0,U=u\right)\leq\sum_{t=\widetilde{t}+d^{\star}-u}^{n}\binom{n}{t}p^{t}\left(1-p\right)^{n-t}.$$

Hence,

$$P(E=0,\widehat{\mathbf{m}}\neq\mathbf{m})\leq\sum_{u=d^{\star}}^{n}\left\{\binom{n}{u}\beta^{u}\left(1-\beta\right)^{n-u}\min\left\{\frac{\sum_{w=d^{\star}}^{u}B_{0,w}\binom{n-w}{u-w}}{\binom{n}{u}},1\right\}\right.$$
$$\left.\cdot\sum_{t=\widetilde{t}+d^{\star}-u}^{n}\binom{n}{t}p^{t}\left(1-p\right)^{n-t}\right\}.$$

Now, we will derive the upper bound on $P(E=1,\widehat{\mathbf{m}}\neq\mathbf{m})$. If the encoding succeeds, the channel is equivalent to the BSC. Thus,

$$P(E=1,\widehat{\mathbf{m}}\neq\mathbf{m})=P(t>\widetilde{t})\leq\sum_{t=\widetilde{t}+1}^{n}\binom{n}{t}p^{t}(1-p)^{n-t}.$$

For the proof of Corollary 2.19, we will change (2.103) by considering $P(S=0)=P(S=1)=\frac{\beta}{2}$. Because only the half of unmasked defects result in stuck-at errors on average,

$$P(\widehat{\mathbf{m}}\neq\mathbf{m}|E=0,U=u)\simeq P\left(t\geq\widetilde{t}-\left\lceil\frac{u-d^{\star}+1}{2}\right\rceil+1\right)$$

Thus, (2.62) will be changed into (2.63).

Chapter 3

Coding for Flash Memory

3.1 Introduction

Flash memory is now the most important nonvolatile memory due to the rapid growth of mobile devices and solid-state drives (SSD). Obviously, there is a strong demand for higher density flash memories. In order to meet this demand, scaling down and multi-level cell (MLC) have driven the continuous growth of flash memory density. Also, vertical stacking has drawn considerable attention in recent years.

Aggressive scaling down of cell size has driven the continuous growth of 2-dimensional (2D) planar flash memory density. However, the scaling down has to deal with many challenges such as increased inter-cell interference (ICI) and photo-lithography limitation [4, 5]. As the distance between adjacent cells decreases due to scaling down, flash memory cells suffer from higher ICI [4, 6]. Hence, the ICI is a major challenge for data reliability of high-density 2D planar flash memory.

In order to cope with the ICI, various approaches have been proposed. Device level approaches such as new materials and novel cell structures try to reduce the parasitic capacitances between adjacent cells [5]. At circuit and architecture levels, several write schemes and all bit-line (ABL) architecture were proposed to deal with the ICI [17, 18, 19, 20]. Also, strong error

control codes (ECC) such as low-density parity check (LDPC) codes and signal processing have been also investigated [21, 22, 23, 24, 25]. The disadvantage of soft decision decoding and signal processing is the degradation of read speed due to multiple reads needed to obtain the soft decision values. In addition, modulation coding has been investigated to remove some data patterns which are more vulnerable to ICI [26, 28, 30]. However, the significant redundancy of modulation coding is a major disadvantage.

Recently, 3D vertical flash memories were proposed to overcome scaling down challenges by stacking up cells in the vertical direction instead of shrinking cells within a 2D plane [7, 8]. The recent 3D vertical flash memory shows better device characteristics compared to 2D 1x nm planar flash memory [7].

However, 3D vertical flash memory has a problem of *fast detrapping*, which is a rapid charge loss phenomenon resulting in larger threshold voltage variations in programmed cells [7, 9]. The fast detrapping usually occurs in charge trap cells rather than floating gate cells [9]. Since 3D vertical flash memory adopts charge trap cells for easier 3D integration [7], fast detrapping is an important problem in 3D vertical flash memory. In contrast, the fast detrapping does not happen in 2D planar flash memory consisting of floating gate cells.

In order to reduce the effect of fast detrapping, several approaches have been proposed. These approaches include cell structure engineering at device level [9] and reprogramming at circuit level [7]. However, the coding and signal processing solutions have not been explored yet.

In this chapter, we propose channel coding schemes for both problems: ICI of 2D planar flash memories and fast detrapping of 3D vertical flash memories. For 2D planar flash memories, the encoder obtains the side information of ICI and tries to combat the ICI. We argue that the 2D planar flash memory channel with ICI is similar to the channel model of Costa's dirty paper coding in [36]. We first explain why flash memories are *dirty* due to ICI. We then show that *dirty flash memory* can be changed into *memory with stuck-at defects* of [37] due to the unique asymmetry property of flash memory between write and erase operations in 3.2. After this transformation, we can apply channel coding for memory with defective cells to combat ICI in 2D planar flash

memories. It is interesting that the unique property of flash memory bridges two notable examples of Gelfand-Pinsker problems: *writing on dirty paper* and *coding for memory with stuck-at defects*.

Next, we propose a channel coding scheme for 3D vertical flash memory. Our scheme aims to compensate the effect of fast detrapping by using intentional ICI. The basic idea comes from the observation that ICI increases the threshold voltage of a cell whereas fast detrapping decreases the threshold voltage of corresponding cell. In order to properly harness the intentional ICI, we formulate the problem of controlling the intentional ICI into coding for memory with defective cells. To the best of our knowledge, our conference paper [68] is the first paper to propose a coding scheme to deal with fast detrapping in 3D vertical flash memory.

Although the proposed coding schemes can improve the data reliability (i.e., decoding failure probability) by using the side information of ICI or fast detrapping, the write speed would be degraded during the obtaining of the side information and incorporating it into encoding.

Note that signal processing and soft decision decoding for flash memory can improve the decoding failure probability at the expense of decreased read speed due to multiple read operations needed to obtain soft decision values. In memory systems, it is well-known that the read speed is more critical than the write speed since the write operation is typically not on the critical path. Due to write buffers in the memory hierarchy, the write latency can be hidden [35, 46]. Also, the read operations are required more often than the write operations in many memory applications. Thus, the coding schemes using side information at the encoder may be preferable over soft decision decoding from the perspective of speed performance. It is worth mentioning that the proposed scheme can be combined with LDPC codes using the technique of additive encoding LDPC codes [69, 70] if we are willing to accept the degradation of read speed.

We cast both problems of combating and harnessing the ICI of flash memories as coding for memory with defective cells based on the unique properties of flash memories. Thus, we rely on channel coding with side information about defects as a unified solution for both 2D planar and 3D vertical flash memories. This side information about defects identifies and targets the

highly interfered cells of 2D planar flash memory and 3D vertical flash memory cells suffering from significant charge loss due to fast detrapping. Moreover, we extend the proposed schemes to MLC flash memories by taking into account the multi-page architecture.

3.2 2D Planar and 3D Vertical Flash Memories

In this section, we explain the basics of 2D planar flash memories and 3D vertical flash memories. Also, we describe the ICI of 2D planar flash memories and fast detrapping in 3D vertical flash memories.

3.2.1 Basic Operations and Asymmetry between Write and Erase Operations

Each cell of 2D planar flash memory is a floating gate transistor whose threshold voltage can be configured by controlling the amount of electron charge in the floating gate. More electrons in the floating gate make the corresponding cell's threshold voltage higher. As shown in Fig. 3.1, each flash memory block of planar flash memory is a 2D cell array where each cell is connected to a word-line (WL) and a bit-line (BL).

In order to store b bits per cell, each cell's threshold voltage is divided into 2^b states, similar to pulse amplitude modulation (PAM). Fig. 3.2 (a) shows the threshold voltage distribution of 1-bit per cell flash memory, which is traditionally called single-level cell (SLC). Initially, all memory cells are erased, so their threshold voltages are in the lowest erase state S_0. In order to store data, some of cells in S_0 should be written (i.e., programmed) into S_1.

For multi-level cell (MLC) flash memories (i.e., $b \geq 2$), some of cells in S_0 (erase state) will be written into $S_1, \ldots, S_{2^b-1}$ (program states) as shown in Fig. 3.2 (b). For b bits per cell flash memory, each WL stores b pages data.

In write operation, the page buffer in Fig. 3.1 is loaded with a unit of page data. Depending on

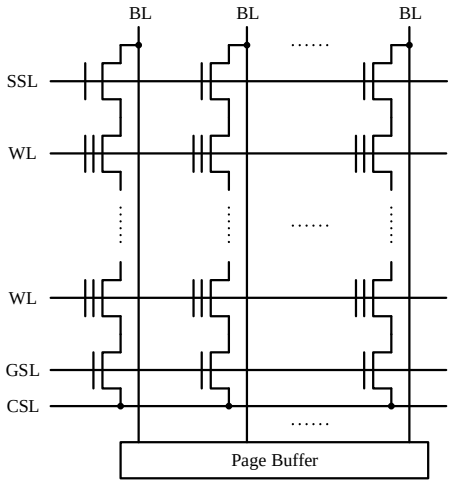
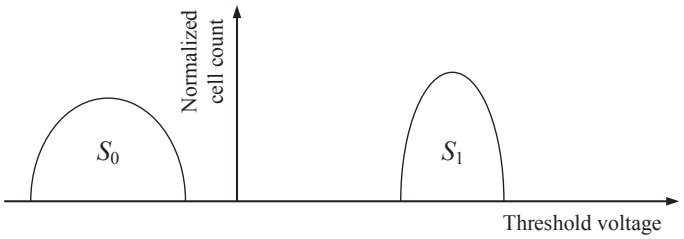
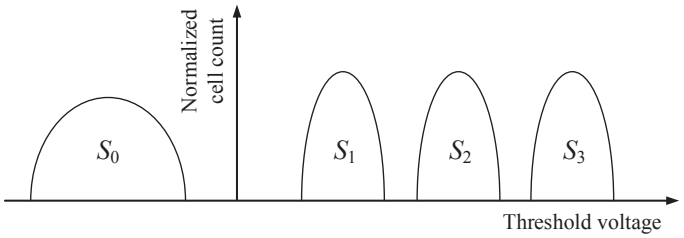


Figure 3.1: 2D planar flash memory block where SSL, GSL, and CSL denote string select-line, ground select-line, and common source-line, respectively.



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



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

Figure 3.2: Threshold voltage distribution of flash memory cells.

the loaded data in the page buffer, some of cells remain in erase state and others are programmed into program states.

The most widely used write operation scheme is the incremental step pulse programming (ISPP) scheme, which was proposed to maintain a tight threshold voltage distribution for high

reliability [71]. The ISPP is based on repeated program and verify cycles with the staircase program voltage V_{pp}. Each program state associates with a verify level that is used in the verify operation. During each program and verify cycle, the floating gate threshold voltage is increased by the incremental step voltage ΔV_{pp} and then compared with the corresponding verify level. If the threshold voltage of the memory cell is still lower than the verify level, the program and verify iteration continues. Otherwise, further programming of this cell is disabled [23, 71].

The positions of program states are determined by verify levels and the tightness of each program state depends on the incremental step voltage ΔV_{pp} [10, 71]. By reducing ΔV_{pp}, the threshold voltage distribution can be made tighter, however the write time increases [71].

In read operation, the threshold voltages of cells in the same WL are compared to a given read level. After a read operation, a page of binary data is transferred to the page buffer in Fig. 3.1. The binary data shows whether the threshold voltage of each cell is lower or higher than the given read level. Namely, the read operation of flash memory is a binary decision. Thus, multiple read operations are required to obtain a soft decision value, which lowers the read speed. The degradation of read speed is an important drawback of soft decision decoding [21].

The threshold voltage of flash memory cell can be reduced by erase operation. In flash memory, all the memory cells in the same flash memory block should be erased at the same time [71]. Note that a page of data (within a WL) can be written or read (generally, a 2D planar flash memory block consists of 64 WLs). In addition, the threshold voltage of cell should be moved into the lowest state S_0 by erase operation whereas a slight increase of threshold voltage is possible by ISPP during write operation [71]. These unique properties of flash memory cause *asymmetry between write and erase operations*.

3.2.2 2D Planar Flash Memories: ICI

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

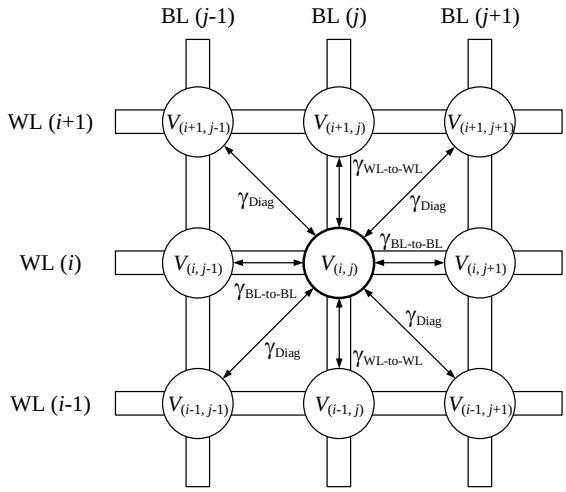


Figure 3.3: Inter-cell interference (ICI) between adjacent cells of 2D planar flash memory.

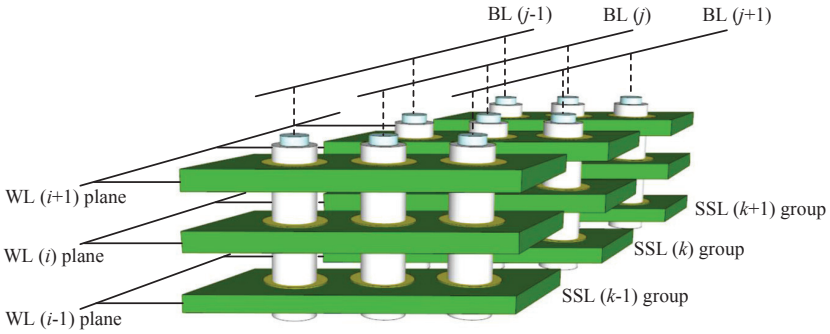
cells [4, 6].

Fig. 3.3 illustrates the ICI between adjacent cells of 2D planar flash memory. $V_{(i,j)}$ is the threshold voltage of (i, j) cell which is situated at i-th WL and j-th BL. $\gamma_{\text{WL-to-WL}}$ is coupling ratio between WL and adjacent WL. Also, $\gamma_{\text{BL-to-BL}}$ is coupling ratio between BL and adjacent BL. Finally, γ_{Diag} is diagonal coupling ratio. These coupling ratios depend on parasitic capacitances between adjacent cells. As the cell size continues to shrink, the distances between cells become smaller, which results in the increase of the parasitic capacitances. The increase of parasitic capacitances causes the increase of coupling ratios [4, 6].

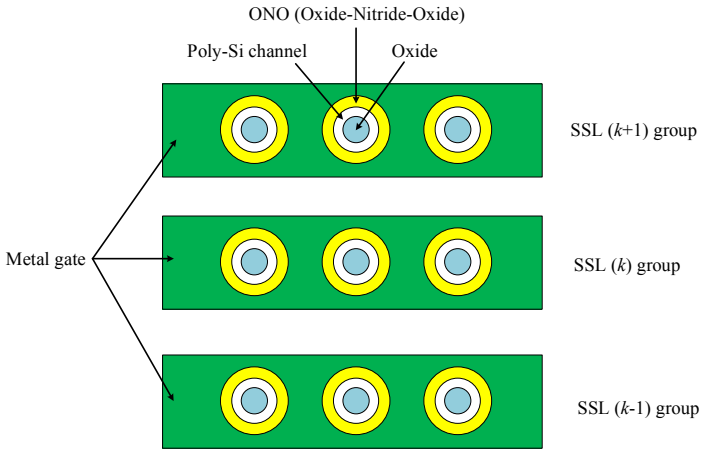
According to [6], the threshold voltage shift $\Delta_{\text{ICI}}V_{(i,j)}$ of (i, j) cell due to the ICI is given by

$$\begin{aligned}\Delta_{\text{ICI}}V_{(i,j)} = {} & \gamma_{\text{WL-to-WL}}\left(\Delta V_{(i-1,j)} + \Delta V_{(i+1,j)}\right) + \gamma_{\text{BL-to-BL}}\left(\Delta V_{(i,j-1)} + \Delta V_{(i,j+1)}\right) \\ & + \gamma_{\text{Diag}}\left(\Delta V_{(i-1,j-1)} + \Delta V_{(i-1,j+1)} + \Delta V_{(i+1,j-1)} + \Delta V_{(i+1,j+1)}\right) \end{aligned} \tag{3.1}$$

where $\Delta V_{(i\pm1,j\pm1)}$ in the right hand of (3.1) side represent the threshold voltage shifts of adjacent cells after the (i, j) cell has been written. The ICI that happens before writing (i, j) cell can be compensated by several write schemes so long as (i, j) cell is in program states [18, 19]. Note that the ICI to (i, j) cell in S_0 cannot be compensated by these write schemes since a cell in S_0



(a) Simplified bird's eye view of 3D vertical flash memory



(b) Cross section diagram along single WL plane

Figure 3.4: 3D vertical flash memory cell array in [1].

is never written (i.e., stays in S_0) [19, 28].

3.2.3 3D Vertical Flash Memories: ICI and Fast Detrapping

Among various 3D flash memory array architectures, the vertical channel type architecture having multiple WL planes, i.e., 3D vertical flash memory has been adopted in [7] for easier 3D integration and better device characteristics. Fig. 3.4 illustrates the simplified bird's eye view of 3D vertical flash memory cell array in [7] and its cross section diagram along single WL plane. Note that a string-select-line (SSL) group in Fig. 3.4 (a) is equivalent to the 2D planar flash memory cell array in Fig. 3.1.

In contrast to the 2D planar flash memory where the floating gate is formed as an electron

storage layer upon single crystal planar silicon channel, the 3D vertical flash memory cell is based on charge trap flash (CTF) technology where a nitride layer inside oxide-nitride-oxide (ONO) stack which is grown as a charge trap layer along the circumference of the thin poly-silicon vertical channel. Note that each charge trap layer in this 3D vertical flash memory is surrounded by the metal gates along WL plane due to its cylindrical geometric structure.

By taking into account the structure of 3D vertical flash memories, we will address the ICI of 3D vertical flash memories. Suppose that $V_{(i,j,k)}$ is the threshold voltage of (i, j, k) cell which is situated at i-th WL, j-th BL, and k-th SSL as shown in Fig. 3.4. The threshold voltage shift $\Delta_{\text{ICI}}V_{(i,j,k)}$ of the (i, j, k) cell due to the ICI can be given by

$$\begin{aligned}\Delta_{\text{ICI}}V_{(i,j,k)} = {} & \gamma_{\text{WL-to-WL}}\left(\Delta V_{(i-1,j,k)} + \Delta V_{(i+1,j,k)}\right) + \gamma_{\text{BL-to-BL}}\left(\Delta V_{(i,j-1,k)} + \Delta V_{(i,j+1,k)}\right) \\ & + \gamma_{\text{SSL-to-SSL}}\left(\Delta V_{(i,j,k-1)} + \Delta V_{(i,j,k+1)}\right) \end{aligned} \tag{3.2}$$

which is an extension of the ICI model of 2D planar flash memories in (3.1). $\Delta V_{(i\pm1,j\pm1,k\pm1)}$ in the right hand side represent the threshold voltage shifts of adjacent cells after the (i, j, k) cell has been written. Note that $\gamma_{\text{WL-to-WL}}$ is coupling ratio between WL plane and adjacent WL plane. Also, $\gamma_{\text{BL-to-BL}}$ is coupling ratio between BL and adjacent BL. Finally, $\gamma_{\text{SSL-to-SSL}}$ is coupling ratio between SSL group and its adjacent SSL group. The diagonal ICI terms are neglected since they are very small due to the longer distance between corresponding cells.

Since each charge trap layer in this 3D vertical flash memory is surrounded by the metal gates along WL plane, the ICI between adjacent BLs typically found in high density 2D planar flash memory is completely absent in the same WL plane. Similarly, the ICI between adjacent SSL groups will be negligible due to the metal gates.

The ICI between adjacent WL planes is also reduced compared to the 2D planar flash memory since the charge trap layer in the 3D vertical flash memory is much thinner than the floating gate layer in the 2D flash memory. However, the ICI between adjacent WL planes increases as the distance between WL planes is reduced for the higher cell density. Thus, it is enough to take into

account only the ICI between adjacent WL planes in the same SSL group and BL. By setting $\gamma_{\text{BL-to-BL}} = \gamma_{\text{SSL-to-SSL}} = 0$, the ICI model of (3.2) can be simplified into

$$\Delta_{\text{ICI}} V_{(i,j,k)} = \gamma_{\text{WL-to-WL}} \left(\Delta V_{(i-1,j,k)} + \Delta V_{(i+1,j,k)} \right). \tag{3.3}$$

We explained that the ICI of 3D vertical flash memory is significantly reduced due to the adoption of CTF technology and the structure of 3D vertical flash memories. However, this charge trap layer results in fast detrapping, which is an important challenge in the 3D vertical flash memory [7].

Fast detrapping is the phenomenon where shallowly trapped electrons in charge trap layers immediately detrap and tunnel out after the programming pulse is terminated [9]. Thus, the charge loss due to fast detrapping quickly decreases the threshold voltage of corresponding cells and degrades the threshold voltage distribution. The fast detrapping occurs immediately during write operation [7, 9].

Device level approaches such as cell structure and material do not completely solve this problem [9]. In [7], a counter-pulse program using self-boosting was proposed at circuit level, which accelerates fast detrapping before a verify operation of ISPP such that fast detrapped cells can be reprogrammed by the subsequent programming pulses. Nevertheless, fast detrapping can happen again in later programming pulses, which requires a different approach at higher levels such as coding level. Since the approaches at different levels can coexist, the proposed scheme at coding level can work with other approaches at lower levels in combating fast detrapping.

3.3 Combating Inevitable Interference for 2D Planar Flash Memory

In this section, we propose a coding scheme for combating ICI of 2D planar flash memories. After introducing the channel model of 2D planar flash memory, we explain our coding scheme

by using coding with side information.

3.3.1 Channel Model of 2D Planar Flash Memory

The channel model of 2D planar flash memory can be given by

$$Y = X + S_{\text{2D}} + Z \tag{3.4}$$

$$= X + Z_{\text{write}} + S_{\text{2D}} + Z_{\text{read}} \tag{3.5}$$

$$= V + S_{\text{2D}} + Z_{\text{read}} \tag{3.6}$$

where X and Y are the channel input and output. Also, S_{2D} represents the ICI from adjacent cells. The additive random noise Z is a sum of Z_{write} and Z_{read} where Z_{write} is the write noise due to the initial threshold voltage distribution after erase operation and the incremental step voltage ΔV_{pp} of ISPP. Z_{read} is the read noise due to other noise sources.

Since the write noise Z_{write} precedes the ICI S_{2D}, we consider a random variable $V = X + Z_{\text{write}}$. As shown in (3.1), the shifts of V in adjacent cells determine the ICI S_{2D}. Thus, we claim that the ICI S_{2D} of (i, j) cell is given by

$$S_{\text{2D}} = \Delta_{\text{ICI}} V_{(i,j)} \tag{3.7}$$

where $\Delta_{\text{ICI}} V_{(i,j)}$ is defined by (3.1). The read noise Z_{read} happens after ICI. The channel model of (3.4) is supported by experimental results from the 2x nm NAND flash memory [72].

It can be seen that (3.4) is equivalent to the interference channel in [36], which is given by

$$Y = X + S_{\text{I}} + Z \tag{3.8}$$

where X and Y are the channel input and output, respectively. Also, $S_{\text{I}} \sim N\left(0, \sigma^2_{S_{\text{I}}}\right)$ represents interference and $Z \sim N\left(0, \sigma^2_Z\right)$ denotes additive noise. Note that S_{I} and Z are independent. We

assume that the channel input satisfies an average power constraint $\frac{1}{n}\sum_{i=1}^{n} X_i^2 \leq P$ for the channel input vector $X^n = (X_1, \cdots, X_n)$. In [36], Costa showed that the interference S_{I} can completely canceled out by dirty paper coding if the entire interference vector $S_{\mathrm{I}}^n = (S_{\mathrm{I},1}, \cdots, S_{\mathrm{I},n})$ is known to the encoder.

However, there are important differences between the flash memory channel of (3.4) and the dirty paper channel of (3.8). First, S_{2D} of (3.4) depends on the adjacent cells' X and Z_{write} whereas S_{I} of (3.8) is independent of X and Z. In addition, the mean of S_{2D} in (3.4) is not zero since the coupling ratios are positive and the threshold voltage shifts of adjacent cells $\Delta V_{(i\pm1,j\pm1)}$ in (3.1) are nonnegative.

Now, we discuss why it is difficult for the encoder to know S_{2D}. First, the encoder has to know the channel input X of adjacent cells in different WLs to know S_{2D}. Since the write operation is performed page by page, it is possible for the encoder to know the channel input of cells in several WLs only in the case where a large number of continuous pages are written at a time.

Even in the case where the encoder knows enough channel input X of several WLs in advance, it is still difficult to know the random variable $V = X + Z_{\mathrm{write}}$ that determines S_{2D} because of the random write noise Z_{write}. In addition, it is much more complicated to know the voltage shift of adjacent cells (i.e., $\Delta V_{(i\pm1,j\pm1)}$ in (3.1)) since flash memory's read operation is inherently binary decision. Hence, multiple read operations are required to know $\Delta V_{(i\pm1,j\pm1)}$, which significantly reduce the read speed performance.

Since it is difficult for the encoder to know the ICI S_{2D} due to these reasons, we change flash memory channel with the ICI into flash memory with defective cells. After this change, the encoder uses the side information of defects rather than the ICI.

3.3.2 Proposed Scheme for 2D Planar Flash Memories: Combating ICI by Coding

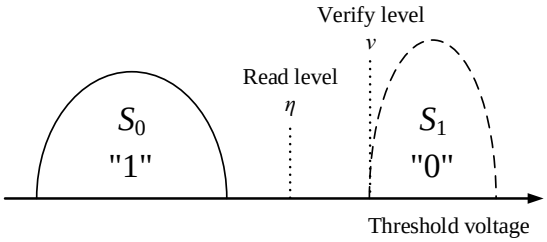
Now we propose our scheme to combat the ICI by coding with side information. First, we describe how to change the 2D planar flash memory channel with ICI into the model of memory with defective cells. After this change, we can apply channel coding schemes for memory with defective cells to combat the ICI of flash memories.

SLC

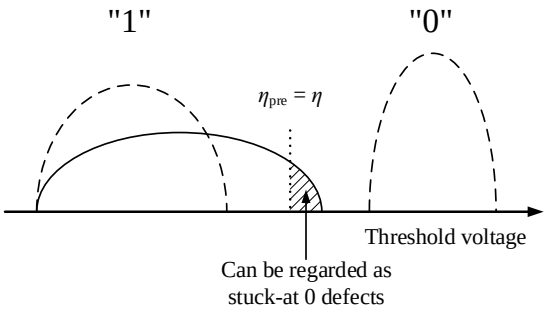
Fig. 3.5 shows the threshold voltage distribution of cells in the i-th WL before writing. Initially, all cells are in the erase state S_0 as shown in Fig. 3.5 (a). However, after writing the adjacent $(i-1)$-th WL, the threshold voltages of cells in the i-th WL will be distorted due to the ICI from the $(i-1)$-th WL as shown in Fig. 3.5 (b). Thus, some of cells' threshold voltages can be higher than the given read level η although the i-th WL is yet to be written.

As explained in Section 3.2.2, the threshold voltage of flash memory cells cannot be reduced during write operation. In order to decrease the threshold voltage of a cell, we have to erase the whole flash memory block. Thus, the cell whose threshold voltage is higher than the read level η will be detected as S_1. Assume that S_0 and S_1 denote the data "1" and "0" respectively. If a "1" is attempted to be written to this cell, an error results. However, "0" can be written into this cell. Thus, these cells can be regarded as stuck-at 0 defects in Fig. 2.1. If some of the cells regarded as stuck-at 0 defects may be "1" due to the read noise Z_{read}, those errors can be regarded as random errors and corrected by the standard ECC.

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



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



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

Figure 3.5: Change from the 2D planar flash memory channel with the ICI to the model of memory with defective cells by one pre-read operation.

information of defects and reduce the errors by highly interfered cells.

Using only one pre-read operation before writing, the flash memory channel with the ICI in (3.4) can be changed into (2.8) of *binary* memory with defective cells. In (2.8) of memory with defective cells, X, Y, and Z are the *binary* vectors. In contrast, X, Y, and Z of (3.4) are *real* values. Note the difference between $X + S_{\text{2D}}$ in (3.4) and $X \circ S$ in (2.8). Now, we can combat the ICI by coding schemes for memory with defective cells.

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

However, the effect of these subsequent ICI can be alleviated by changing the pre-read level. Suppose that the pre-read level η_{pre} is lower than the read level η as shown in Fig. 3.6. A cell

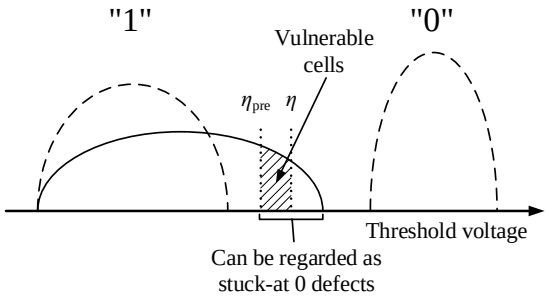


Figure 3.6: Vulnerable cells can also be regarded as stuck-at 0 defects by setting a pre-read level such that $\eta_{\text{pre}} < \eta$.

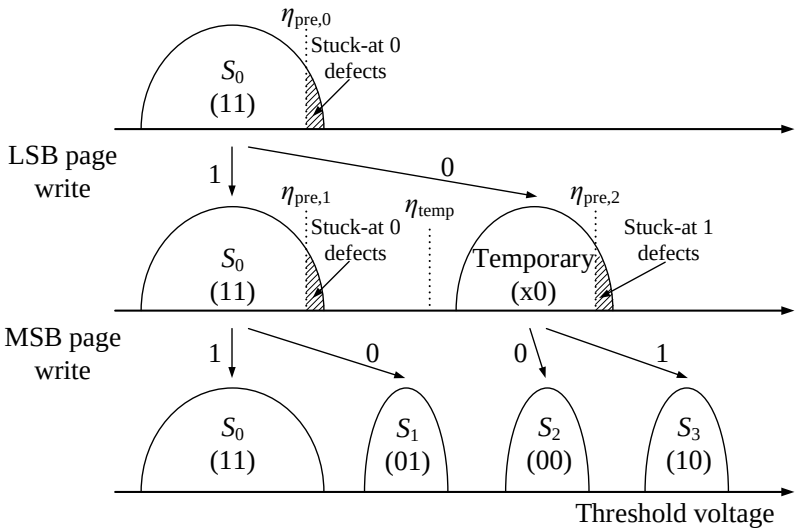
whose threshold voltage is between η_{pre} and η is a vulnerable cell although it is not a stuck-at 0 defect. When the data "1" is written to this cell, the ISPP cannot change the threshold voltage of this cell and its threshold voltage is near η. Thus, it is vulnerable to the subsequent ICI and read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

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

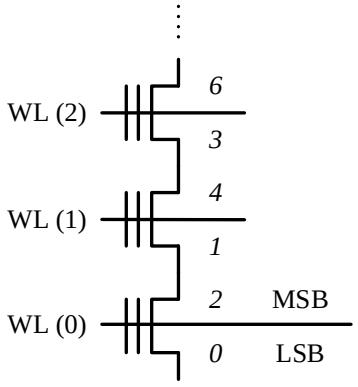
MLC

We can extend our proposed scheme to MLC flash memory. Most MLC flash memories adopt the multi-page architecture [17]. In 2 bits per cell flash memories, the least significant bit (LSB) and most significant bit (MSB) are mapped to two separate pages: LSB page and MSB page. Since one page is the unit of data that is written and read at one time, the ECC should be applied within the same page.

Fig. 3.7 explains the most widely used MLC write operation in [18]. At the LSB page writing stage, the cell can be written from "11" to "x0" as a temporary state just like SLC write operation.



(a) Extension to MLC flash memories



(b) Page address ordering to reduce the ICI from the next WL [18]

Figure 3.7: Extension to MLC flash memories.

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

S_1 and 2) from temporary state to S_2 or S_3.

The ICI from the next WL can be reduced by performing MSB page writing for a selected WL after writing the LSB page of next WL [18]. The ICI from writing the LSB page in $(i+1)$-WL can be compensated during writing the MSB page in i-th WL due to the page address ordering in Fig. 3.7 (b). The details of multi-page architecture and page address ordering can be found in [17, 18].

If the LSB page data detection is erroneous, the finally programmed state can be wrong. Suppose that the LSB page data 1 is misread as 0 because the threshold voltage of corresponding cell is higher than η_{temp}. If the MSB page data is 1, then the final state will be S_3 (10) instead of S_0 (11) [73, 74]. Hence, an error happens in the LSB page. We call this error as *internal error*, which can result in a large magnitude error (e.g., from S_0 to S_3 and vice versa).

However, many more errors than internal errors happen between S_1 and S_2. The reason is that the noise margin between S_1 and S_2 is much smaller than that of erase state S_0 (11) and temporary state (x0). Most LSB page errors between S_1 and S_2 are not related to the side information of stuck-at 0 defects obtained by read at the pre-read level $\eta_{\text{pre},0}$. Instead, they depend on the noise margin between S_1 and S_2 decided during the MSB write operation. Hence, we do not apply the proposed scheme for LSB page although the proposed scheme can reduce the internal errors in the LSB page.

For the MSB page, the proposed scheme can improve the decoding failure probability by using the side information. Before writing the MSB page, two pre-read operations at the pre-read levels of $\eta_{\text{pre},1}$ and $\eta_{\text{pre},2}$ are required to obtain the side information. A cell whose threshold voltage is between $\eta_{\text{pre},1}$ and η_{temp} will be regarded as a stuck-at 0 defect. Also, a cell whose threshold voltage is higher than $\eta_{\text{pre},2}$ is regarded as a stuck-at 1 defect. By using this side information of defects, the proposed scheme can improve the decoding failure probability. The proposed scheme can be extended to three and more bits per cell MLC flash memories by the same way.

3.3.3 Dirty Paper vs. Dirty Flash Memory

We explain that the unique asymmetry property of write and erase operations of flash memory connects two notable examples of Gelfand-Pinsker problem: *writing on dirty paper* and *memory with defective cells*.

Imagine a sheet of *lined* paper (Costa considered a sheet of *blank* paper in [36]). A flash memory block is a sheet of paper and each WL corresponds to a row between lines. If a row between lines is spacious, then the writer can easily write a message between lines.

In order to write more messages on a sheet of paper, the writer tries to narrow the space between lines (i.e., scaling down). However, as the space between lines narrows, it is more difficult to write a message without crossing the lines (i.e., ICI). Eventually, after writing a message in a narrower space, the adjacent rows have more dirty spots (i.e., stuck-at defects) due to the ink marks crossing the line. One way to solve this problem is to erase the dirty spots in a corresponding row before writing. However, erasing a row is not permitted because of the asymmetry between write and erase operations in flash memory.

Now we consider another option instead of erasing a row before writing. Assume that the writer knows the location of the dirty spots, but the reader cannot distinguish between the message and the dirt [36]. Hence, the problem of writing on flash memory with the ICI can be considered as a Costa's writing on dirty paper, i.e., *writing on dirty flash memory*. Since the dirty spots are changed into stuck-at defects by the pre-read operation, writing on dirty flash memory is equivalent to writing on (flash) memory with stuck-at defects. Thus, *writing on dirty paper* and *coding for memory with stuck-at defects* can be bridged in 2D planar flash memory.

3.3.4 Simulation Results

We present the simulation results of proposed scheme for 2D planar flash memories. The simulation parameters are summarized in Table. 3.1. The initial threshold voltage distribution (after erasing a flash memory block) is assumed to be the Gaussian distribution $\mathcal{N}(-4, 1^2)$. ISPP was

Table 3.1: Simulation Parameters of 2D Planar Flash Memory

Parameters	Values
Architecture	All bit-line (ABL)
Initial threshold voltage distribution	$\mathcal{N}\left(-4, 1^2\right)$
Verify levels	For SLC, $\nu_1 = 1$ For MLC, $\nu_1 = 1$, $\nu_2 = 2.5$, $\nu_3 = 4.5$
Incremental step voltage of ISPP	For SLC and LSB of MLC, $\Delta V_{\text{pp}} = 1$ For MSB of MLC, $\Delta V_{\text{pp}} = 0.25$
$\left(\gamma_{\text{WL-to-WL}}, \gamma_{\text{BL-to-BL}}, \gamma_{\text{Diag}}\right)$	$\alpha\left(0.1, 0.08, 0.006\right)$
Z_{read} of (3.6)	$\mathcal{N}\left(0, \sigma^2_{Z_{\text{read}}}\right)$

implemented with the parameters of the verify level for S_i, i.e., ν_i and the incremental step voltage ΔV_{pp}. The variance of initial threshold voltage distribution and the incremental step voltage ΔV_{pp} work for Z_{write} of (3.5), which precedes the ICI.

The ICI S_{2D} is calculated by (3.1) where the coupling ratios are $\left(\gamma_{\text{WL-to-WL}}, \gamma_{\text{BL-to-BL}}, \gamma_{\text{Diag}}\right) = \alpha\left(0.1, 0.08, 0.006\right)$. The scaling factor α represents the ICI strength, and the ratios between $\gamma_{\text{WL-to-WL}}$, $\gamma_{\text{BL-to-BL}}$, and γ_{Diag} are taken from [4]. These ratios can be different for each product of flash memory. The read noise Z_{read} after the ICI is assumed to the $\mathcal{N}\left(0, \sigma^2_{Z_{\text{read}}}\right)$.

Most of our simulation results were based on PBCH codes. The possible parameter sets of $[n = 1023, k = 923, l]$ PBCH codes are presented in Table 2.1. However, the proposed scheme can be combined with additive encoding LDPC codes. After obtaining the defect information corresponding the ICI by the proposed scheme, this defect information can be incorporated by additive encoding LDPC codes [69, 70]. In the two-dimensional magnetic recording (TDMR) channel where the soft decision value is obtained without the read speed degradation, [70] shows that additive encoding LDPC codes with defect information corresponding to the channel state of TDMR improve the decoding failure probability of LDPC codes.

Fig. 3.8 shows that the proposed scheme can improve the threshold distribution by using the

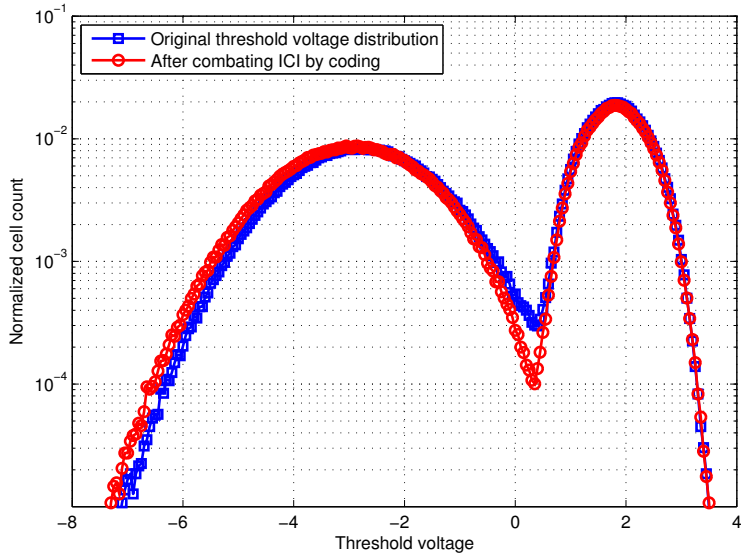


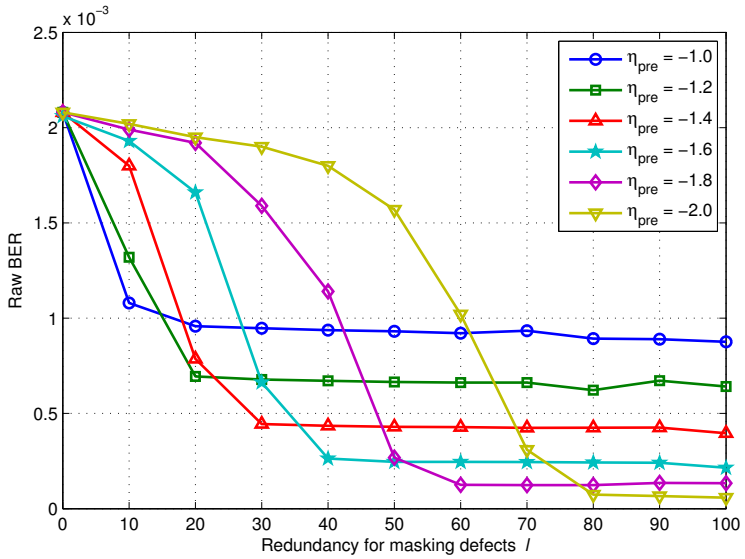
Figure 3.8: The improvement of threshold voltage distribution by proposed scheme (SLC, $n = 1023$, $k = 923$, $R = 0.90$, $\alpha = 1.2$, $\sigma_{Z_{\text{read}}} = 0.25$, $\eta_{\text{pre}} = -1.4$).

side information of highly interfered cells and vulnerable cells, which are regarded as stuck-at defects. In contrast, the standard channel coding schemes cannot improve the threshold voltage distribution.

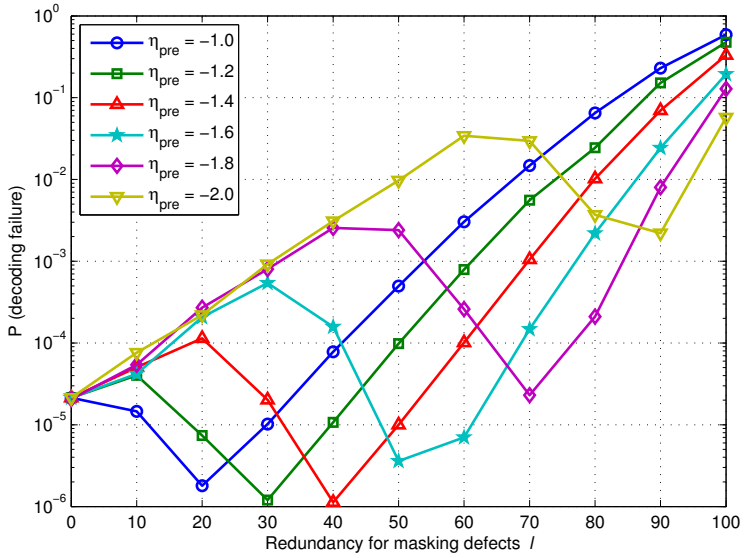
The improvement of threshold voltage distribution depends on the redundancy allocation (l, r) and the pre-read level η_{pre}. Fig. 3.9 (a) shows this relation where raw bit error rates (BER) are plotted instead of threshold voltage are reduced by allocating more redundancy l for masking defects, which leads to the improvement of threshold voltage distributions.

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

Fig. 3.9 (b) shows that $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ is improved by the proposed scheme. If $l > 0$, then



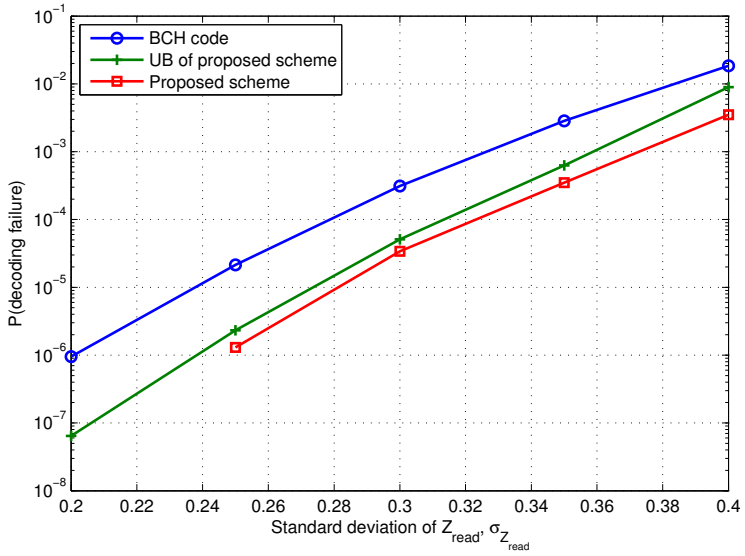
(a) Raw BER



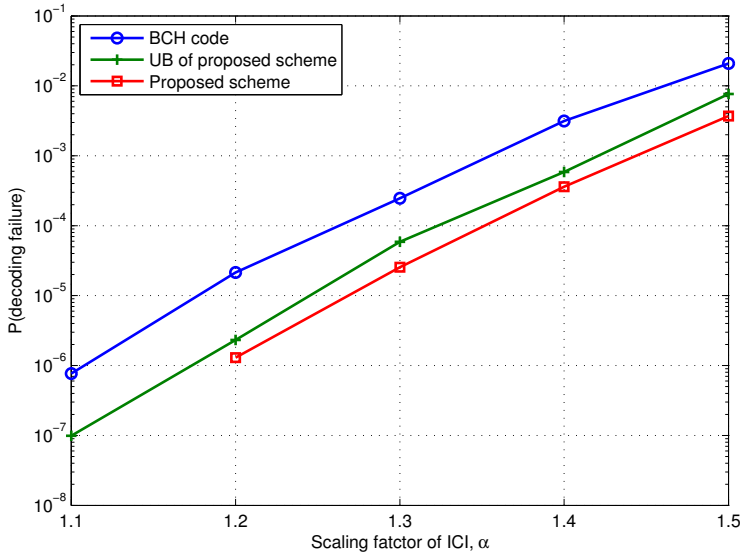
(b) Probability of decoding failure, $P\left(\widehat{\mathbf{m}} \neq \mathbf{m}\right)$

Figure 3.9: The improvement of raw BER and probability of decoding failure by the proposed scheme (SLC, $n = 1023, k = 923, R = 0.90$, $\alpha = 1.2, \sigma_{Z_{\text{read}}} = 0.25$). If $l = 0$, then the side information is ignored, which is equivalent to the BCH code.

the encoder uses the side information of defects to improve $P\left(\widehat{\mathbf{m}} \neq \mathbf{m}\right)$. Note that $P\left(\widehat{\mathbf{m}} \neq \mathbf{m}\right)$ depends on the pre-read level η_{pre} since it controls the number of cells regarded as defects. In order to minimize $P\left(\widehat{\mathbf{m}} \neq \mathbf{m}\right)$, we should choose the optimal redundancy allocation (l^*, r^*). For lower η_{pre}, more cells are regarded as stuck-at defects. More defects require more redundancy for



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



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

Figure 3.10: Comparison of $P\left(\widehat{\mathbf{m}} \neq \mathbf{m}\right)$ (SLC, $n = 1023, k = 923, R = 0.90$).

l. For the given simulation results in Fig. 3.9 (b), the pre-read level $\eta_{\text{pre}} = -1.4$ and $(l^*, r^*) = (40, 60)$ minimizes $P\left(\widehat{\mathbf{m}} \neq \mathbf{m}\right)$, which was improved from 2.1×10^{-5} to 1.1×10^{-6} by using the side information of defects.

Fig. 3.10 compares $P\left(\widehat{\mathbf{m}} \neq \mathbf{m}\right)$ of BCH codes and our proposed scheme based on PBCH codes where l was optimized for each channel parameters α and $\sigma_{Z_{\text{read}}}$. Also, the upper bound on $P\left(\widehat{\mathbf{m}} \neq \mathbf{m}\right)$ by (2.62) is close to the simulation results. The raw BER and the empirical

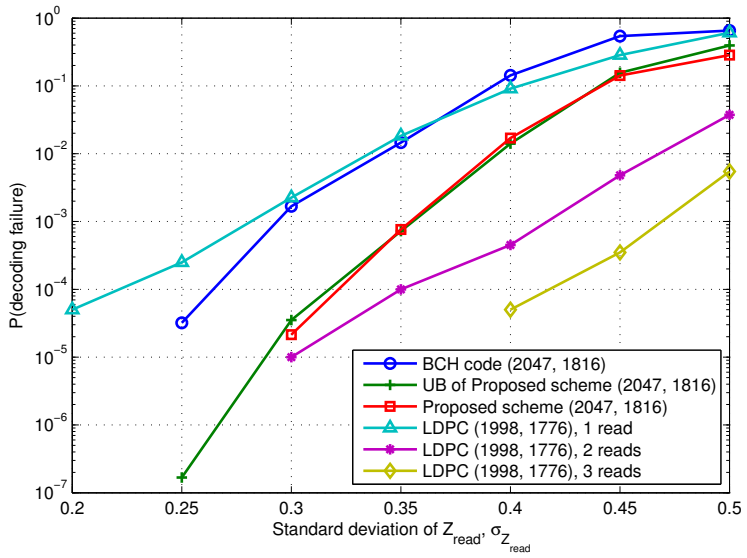


Figure 3.11: Comparison of $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ for BCH codes, proposed scheme, and LDPC codes (SLC, $R = 0.89$, $\alpha = 1.4$).

probability of stuck-at defects are used for p and β in (2.62), respectively. By using this upper bound, we can estimate $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ quickly without the computationally demanding Monte-Carlo simulations especially for very low $P(\widehat{\mathbf{m}} \neq \mathbf{m})$.

Fig. 3.10 (b) shows that the gain of scaling factor α by proposed scheme is around 0.1. We can claim that $\alpha \propto \frac{1}{D}$ where D denotes the distances between flash memory cells because the parasitic capacitances are inversely proportional to D. Since the density of 2D planar flash memory is proportional to $\frac{1}{D^2}$, we can claim that the density gain is about 19%$\left(= \frac{1.2^2}{1.1^2}\right)$ for $P(\widehat{\mathbf{m}} \neq \mathbf{m}) \approx 10^{-6}$ in Fig. 3.10 (b).

In Fig. 3.11, we compare $P(\widehat{\mathbf{m}} \neq \mathbf{m})$ of BCH codes, proposed scheme based on PBCH codes, and LDPC codes. LDPC codes with column weight four came from [75] and the read levels are chosen to maximize the mutual information [22]. For $P(\widehat{\mathbf{m}} \neq \mathbf{m}) \approx 10^{-5}$, the proposed scheme based on PBCH codes is comparable to LDPC code with 2 reads although the read speed performance is the same as LDPC code with 1 read. Since the read speed degradation is more critical than write speed degradation in many memory system applications, the proposed scheme has an advantage over soft decision decoding of LDPC codes. In addition, the proposed scheme

based on PBCH codes does not suffer from error floor and can be estimated by the upper bound in (2.62). If we are willing to accept the read speed degradation, then the proposed scheme can be combined with LDPC codes, i.e., additive encoding LDPC codes.

Fig. 3.12 (a) shows that the proposed scheme can improve the threshold voltage distribution of MLC flash memories. Fig. 3.12 (b) shows the improvement of $P\left(\widehat{\mathbf{m}} \neq \mathbf{m}\right)$ of the MSB page data by the proposed scheme.

3.4 Harnessing Intentional Interference for 3D Vertical Flash Memory

In this section, we propose a coding scheme that reduces the effect of fast detrapping in 3D vertical flash memories. In order to compensate the charge loss by fast detrapping, we take advantage of the intentional ICI controlled by coding with side information corresponding to fast detrapping.

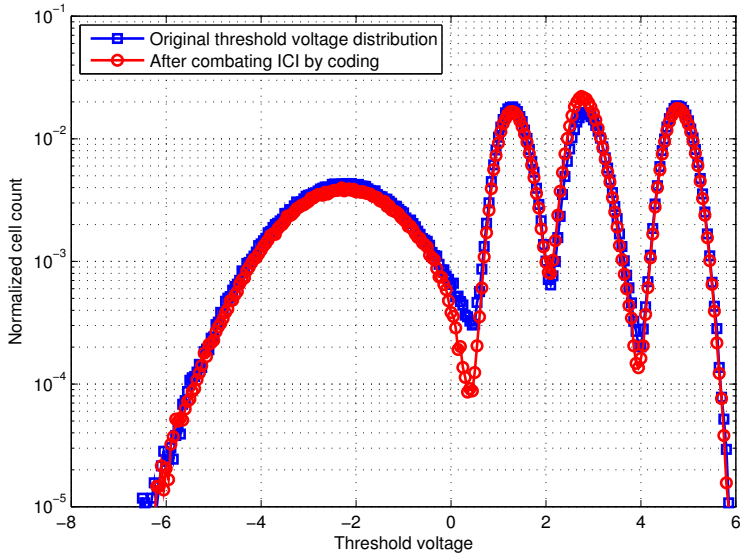
3.4.1 Channel Model of 3D Vertical Flash Memory

By taking into account fast detrapping, the 3D vertical flash memory channel model can be modified from the 2D planar flash memory channel model of (3.6) as follows.

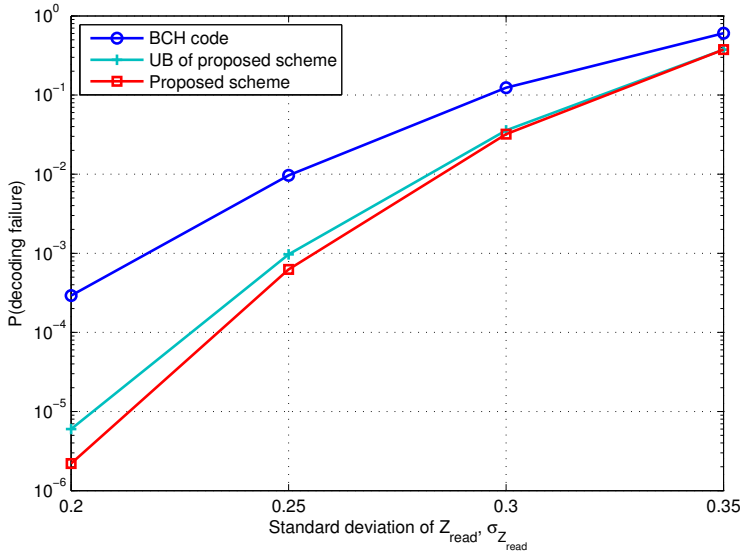
$$Y = V + S_{\text{3D}} + Z_{\text{read}} = V + S_{\text{3D}} + Z_{\text{fast}} + Z_{\text{random}} \tag{3.9}$$

where $Z_{\text{read}} = Z_{\text{fast}} + Z_{\text{random}}$. Z_{fast} denotes the noise due to fast detrapping. All the other read noise sources are represented by Z_{random}. In addition, S_{3D} denotes the ICI of the 3D vertical flash memory. As shown in (3.3), S_{3D} of (i, j, k) cell is given by

$$S_{\text{3D}} = \Delta_{\text{ICI}} V_{(i,j,k)} = \gamma_{\text{WL-to-WL}} \left(\Delta V_{(i-1,j,k)} + \Delta V_{(i+1,j,k)} \right).$$



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

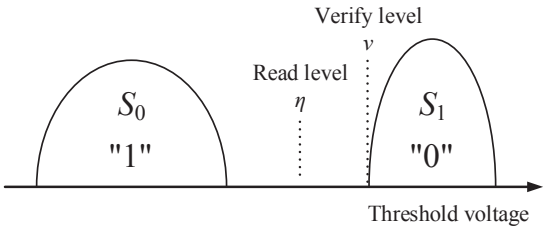


(b) $P\left(\widehat{\mathbf{m}} \neq \mathbf{m}\right)$ of MSB page for different $\sigma_{Z_{\text{read}}}$ ($\alpha = 1.0$)

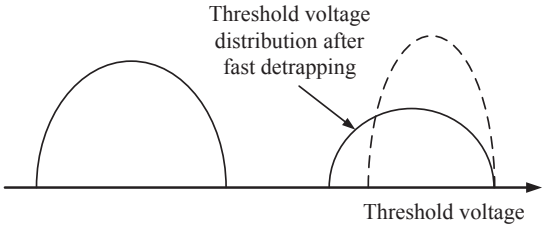
Figure 3.12: The improvement of threshold voltage distribution and $P\left(\widehat{\mathbf{m}} \neq \mathbf{m}\right)$ (MLC, $n = 1023, k = 923, R = 0.90$).

3.4.2 Proposed Scheme for 3D Vertical Flash Memory: Harnessing ICI by Coding

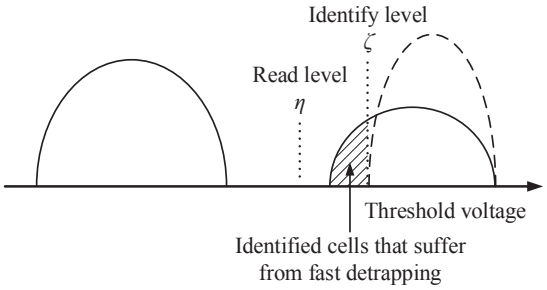
In order to reduce the effect of fast detrapping using coding with side information, we should first obtain the side information corresponding to fast detrapping. Fig. 3.13 (a) shows the threshold



(a) Threshold voltage distribution of cells in the i-th WL before fast detrapping



(b) Threshold voltage distribution of cells in the i-th WL after fast detrapping



(c) Identified cells that suffer from fast detrapping in the i-th WL

Figure 3.13: Fast detrapping and identifying cells that suffer from fast detrapping.

voltage distribution before fast detrapping happens. After writing the i-th WL, some of cells in S_1 immediately suffer from fast detrapping and their threshold voltages decrease as shown in Fig. 3.13 (b).

The cells suffering from fast detrapping can be identified using two read operations at the read level η and the identify level ζ, as shown in Fig. 3.13 (c). If a cell's threshold voltage is between η and ζ, we can claim that this cell suffers from fast detrapping and the encoder obtains the location of this cell, which is the side information about fast detrapping.

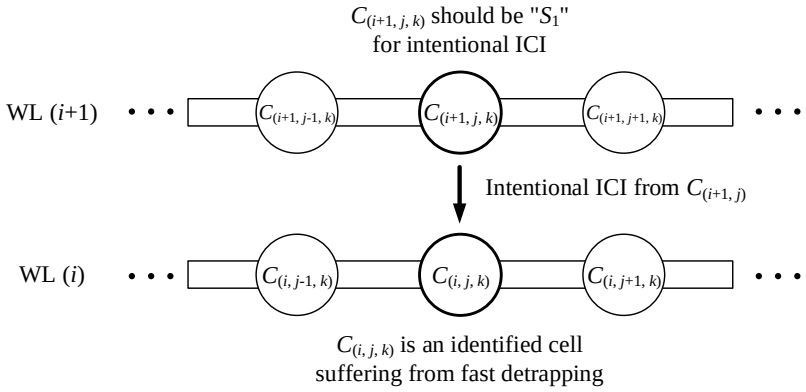


Figure 3.14: Intentional ICI for compensating fast detrapping. The cell $C_{(i+1,j,k)}$ will be regarded as stuck-at 0 ("S_1") defect for the intentional ICI.

If the encoder holds the original data of the i-th WL, we can obtain this side information of fast detrapping by just one read operation since we do not need to apply the read operation at the read level η. By combining the original data and the binary data obtained from the read operation at the identify level ζ, the encoder can identify the cells suffering from fast detrapping.

The identify level ζ does not need to be the same as the verify level ν. We can change the identify level ζ by taking into account the strength of fast detrapping. The number of identified cells depend on the identify level ζ.

Now the encoder knows the side information corresponding to fast detrapping in cells of the i-th WL. The side information represents the locations of identified cells, which suffer from fast detrapping. The next step is to use this side information during encoding.

The key idea is to compensate the effect of the fast detrapping in the i-th WL by controlling the *intentional* ICI from the $(i + 1)$-th WL. The intentional ICI can compensate the decrease of threshold voltage due to fast detrapping since the ICI increases the threshold voltage of the interfered cell. Note that fast detrapping results in charge loss, which decreases the corresponding cells' threshold voltages.

Assume that a cell $C_{(i,j,k)}$ in Fig. 3.14 suffered from fast detrapping, which has been identified by the read operations. If the upper cell $C_{(i+1,j,k)}$ is written into S_1, the ICI from $C_{(i+1,j,k)}$ will increase the threshold voltage of $C_{(i,j,k)}$ during increasing the threshold voltage of $C_{(i+1,j,k)}$ from

S_0 to S_1. If $C_{(i+1,j,k)}$ is written into S_0, the threshold voltage of $C_{(i+1,j,k)}$ remains at the initial erase state S_0, so the ICI from $C_{(i+1,j,k)}$ is absent.

Thus, in order to compensate the effect of fast detrapping, the upper cell $C_{(i+1,j,k)}$ of the identified cell $C_{(i,j,k)}$ should be written into S_1 which is mapped into binary data "0" as shown in Fig. 3.13 (a).

As a technique to control the intentional ICI properly, we formulate the problem of controlling the intentional ICI into memory with defective cells. Since the cell $C_{(i+1,j,k)}$ in Fig. 3.14 should store the data "0" (i.e., S_1) for the intentional ICI, we will regard this cell $C_{(i+1,j,k)}$ as stuck-at 0 defect. Then, the additive encoding tries to make the codeword's element corresponding the cell $C_{(i+1,j,k)}$ become "0." After writing the controlled data into the $(i+1)$-th WL, the ICI from the cell $C_{(i+1,j,k)}$ increases the threshold voltage of the cell $C_{(i,j,k)}$, which compensates the threshold voltage decrease of the cell $C_{(i,j,k)}$ due to fast detrapping.

The proposed scheme can be extended to MLC flash memories. For 2 bits per cell 3D vertical flash memories, suppose that the MSB page write operation is done in the i-th WL. Because of page address ordering in Fig. 3.7 (b), the $(i+1)$-th WL's LSB page is written before the MSB page write operation of i-th WL. Hence, the cells in the $(i+1)$-th WL will be in S_0 (11) or temporary state (x0) as shown in Fig. 3.7 (a). Before writing the MSB page of the $(i+1)$-th WL, we can identify a cell $C_{(i,j,k)}$ suffering from fast detrapping among the program states (S_1, S_2, and S_3) by read operations.

For the intentional ICI, we should increase the threshold voltage of the upper cell $C_{(i+1,j,k)}$. In Fig. 3.7 (a), we can observe that the the threshold voltage of $C_{(i+1,j,k)}$ can be increased from S_0 (11) to S_1 (01) or from temporary state (x0) to S_3 (10). Hence, the MSB page data of $C_{(i+1,j,k)}$ should be regarded as a stuck-at 0 defect if $C_{(i+1,j,k)}$ is in S_0 (11). Also, the MSB page data of $C_{(i+1,j,k)}$ should be regarded as a stuck-at 1 defect if $C_{(i+1,j,k)}$ is in temporary state (x0).

In summary, the encoder obtains the locations of identified cells which are suffering from fast detrapping in the i-th WL before writing the $(i+1)$-th WL. These locations can be obtained by reading at the identify levels, which would degrade the write speed performance. For the

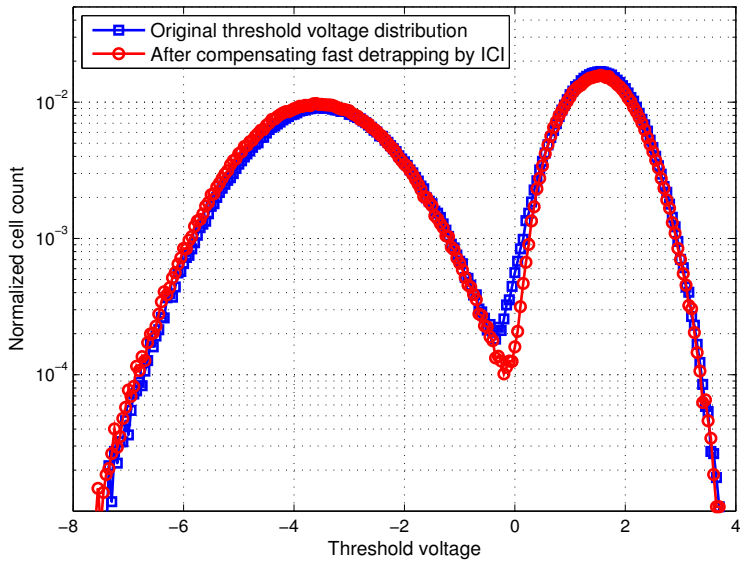
intentional ICI, the upper cells in the $(i+1)$-th WL of the identified cell in the i-th WL will be regarded as stuck-at defects. Note that the side information of *fast detrapping* in the i-th WL is changed into the side information of *defects* in the $(i+1)$-th WL. Hence, we can harness the intentional ICI by a coding technique for memory with defects.

3.4.3 Simulation Results

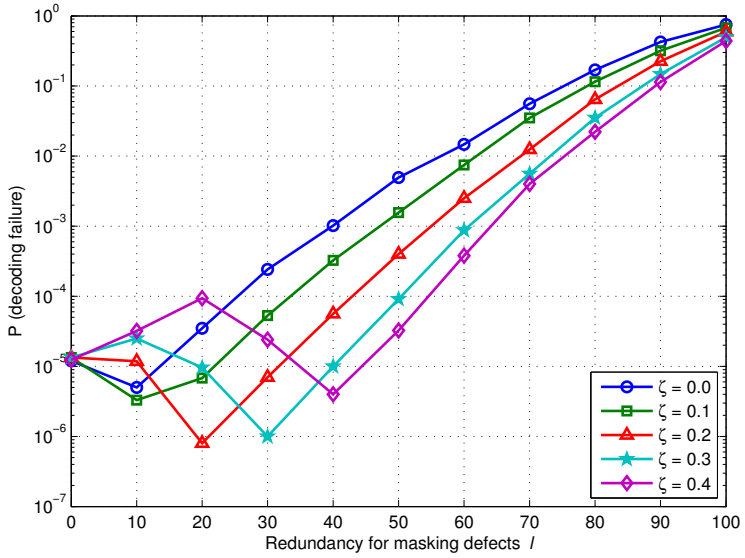
In this section, simulation results are presented. We use the same simulation parameters as in Table 3.1 besides the noise due to fast detrapping and coupling ratio. The noise due to fast detrapping Z_{fast} is assumed to be the Gaussian distribution $\mathcal{N}\left(-0.2, \sigma^2_{Z_{\text{fast}}}\right)$ by taking into account experimental results in [9]. We applied $[n = 1023, k = 923, l]$ PBCH codes with the two-step encoding scheme in Table 2.1.

Fig. 3.15 (a) shows that controlling ICI by coding with side information can compensate the effect of fast detrapping. After compensating the fast detrapping by the intentional ICI, the threshold voltage distribution can be improved. Fig. 3.15 (b) shows that $P\left(\widehat{\mathbf{m}} \neq \mathbf{m}\right)$ can be improved by the proposed scheme. If the redundancy l for masking defects is zero, it means that the side information of fast detrapping is ignored. Otherwise, the encoder uses the side information of fast detrapping to improve $P\left(\widehat{\mathbf{m}} \neq \mathbf{m}\right)$. Note that $P\left(\widehat{\mathbf{m}} \neq \mathbf{m}\right)$ depends on the identify level ζ since it controls the number of identified cells, which is equivalent to the number of cells regarded as stuck-at defects in the upper WL. Also, the optimal redundancy allocation (l^*, r^*) to minimize $P\left(\widehat{\mathbf{m}} \neq \mathbf{m}\right)$ depends on the identify level ζ. For larger ζ, the number of cells identified as defects increases, which requires more redundancy for masking defects. For the given parameters in Fig. 3.15 (b), the identify level of $\zeta = 0.2$ minimizes $P\left(\widehat{\mathbf{m}} \neq \mathbf{m}\right)$.

Fig. 3.16 compares $P\left(\widehat{\mathbf{m}} \neq \mathbf{m}\right)$ of BCH codes and our proposed scheme. In Fig. 3.16 (a), $\sigma_{Z_{\text{fast}}}$ of fast detrapping is varied from 0.4 to 0.5 for the given random noise of $\sigma_{Z_{\text{random}}} = 0.2$. By using the side information of fast detrapping, we can significantly improve $P\left(\widehat{\mathbf{m}} \neq \mathbf{m}\right)$ for different $\sigma_{Z_{\text{fast}}}$. Also, Fig. 3.16 (b) compares $P\left(\widehat{\mathbf{m}} \neq \mathbf{m}\right)$ for different $\sigma_{Z_{\text{random}}}$ for the given $\sigma_{Z_{\text{fast}}} =$



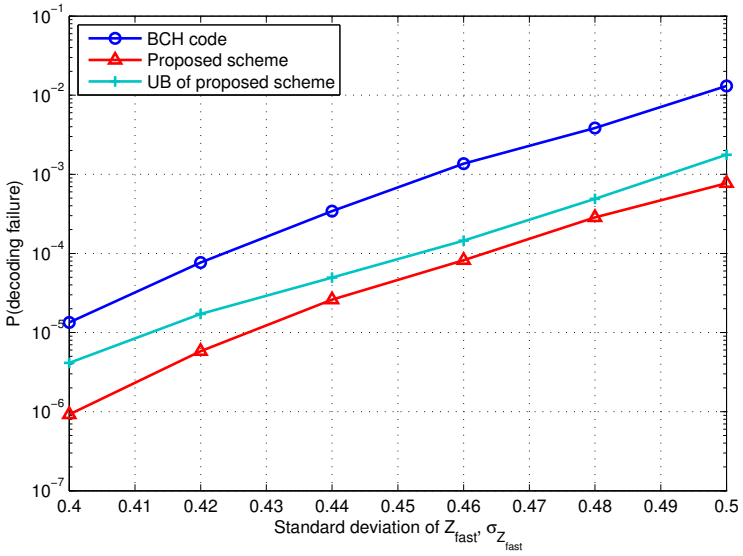
(a) The threshold voltage distribution ($\zeta = 0.4$)



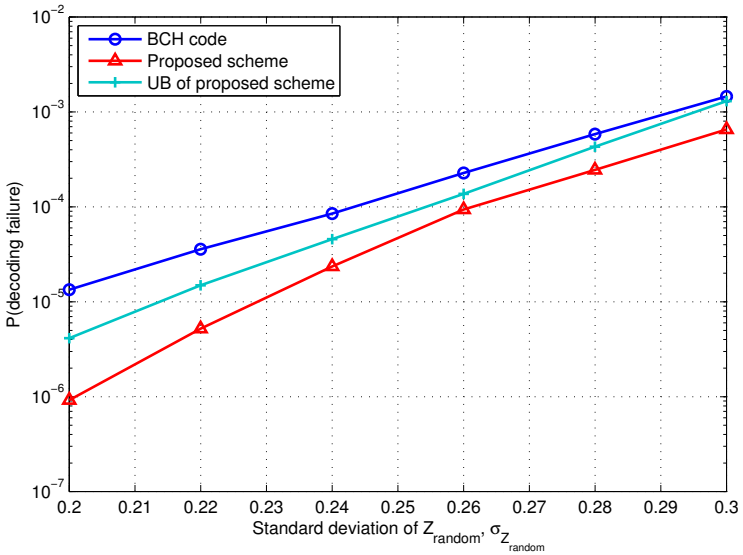
(b) $P\left(\widehat{\mathbf{m}} \neq \mathbf{m}\right)$ for different identify levels ζ

Figure 3.15: The improvement of threshold voltage distribution and $P\left(\widehat{\mathbf{m}} \neq \mathbf{m}\right)$ (SLC, $n = 1023, k = 923, R = 0.90, \sigma_{Z_{\text{fast}}} = 0.4, \sigma_{Z_{\text{random}}} = 0.2$).

0.40. It is worth mentioning that the improvement of $P\left(\widehat{\mathbf{m}} \neq \mathbf{m}\right)$ becomes significant as the fast detrapping Z_{fast} dominates the random noise Z_{random}.



(a) For the given $\sigma_{Z_{\text{random}}} = 0.20$



(b) For the given $\sigma_{Z_{\text{fast}}} = 0.40$

Figure 3.16: Comparison of $P\left(\widehat{\mathbf{m}} \neq \mathbf{m}\right)$ for different $\sigma_{Z_{\text{fast}}}$ of fast detrapping and $\sigma_{Z_{\text{random}}}$ (SLC, $n = 1023, k = 923, R = 0.90$).

3.5 Conclusion

Our proposed schemes can combat and harness the ICI via coding with side information at the encoder. For 2D planar flash memory, the proposed scheme combats the ICI, which is a primary challenge in 2D planar flash memory. For 3D vertical flash memory, we harness the intentional

ICI via coding to compensate the effect of fast detrapping, which is a problem of 3D vertical flash memory. Because of the unique properties of flash memory, both problems of combating and harnessing the ICI were cast as the coding problem of memory with defective cells. Our proposed coding schemes can improve the data reliability at the cost of decrease in write speed performance whereas soft decision decoding schemes degrade the read speed.

Chapter 4

Coding for Resistive Memory

4.1 Introduction

Among several nonvolatile memory technologies, resistive memories have attracted significant research interest recently. The resistive memories are nonvolatile memories that store data by changing the resistance of each memory cell. Phase change memories (PCM) and resistive random access memories (RRAM or ReRAM) are major technologies of resistive memories.

The advantages of resistive memories are scalability, non-volatility, fast speed, and rewritability [2, 11, 12, 13, 14]. First, the potential scalability to the nanometer regime is one of the most important advantages of resistive memories over dynamic random access memories (DRAM) and flash memories [15, 16]. Unlike DRAM, resistive memories are nonvolatile memories, so refresh operations as in DRAM are unnecessary. Resistive memories can realize the MLC and be stacked in 3D with a compact cross-point architecture for higher density. Moreover, resistive memory technologies are expected to offer better speed performance than flash memories [12, 14]. The resistive memories also allow rewriting the data without erase operation, which is an advantage over flash memories where a number of cells should be erased to rewrite a single memory cell.

Two important challenges of resistive memories are *write endurance* and *write power consumption*. The write endurance refers to the fact that the repeated writes cause resistive mem-

ory cells to become unreliable. The write endurance is a critical characteristic because higher endurance broadens the application areas where frequent write operations are required [12]. Although the write endurance of resistive memories is better than that of flash memories, their write endurance is worse than DRAM. In order to open up many potential applications such as embedded memory, the write endurance should be improved substantially [12, 14].

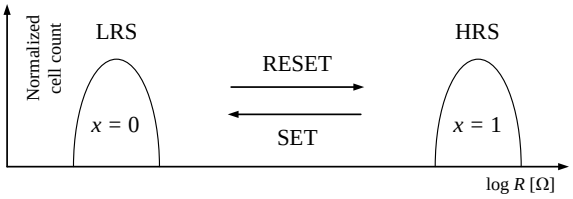
The write operation of resistive memories requires higher power consumption than the read operation so as to change the physical states of memory cells. Especially, the increase of resistance of PCM cell needs a substantial power [12, 13]. This inherent write power consumption problem is an important hurdle for the increase of write bandwidth because the high write power prohibits writing of many bits in parallel [13].

In order to explore coding approaches to improve write endurance and power consumption of resistive memories, a proper channel model is important. We rely on the channel model of memory with stuck-at defects [37] because a heavily cycled (i.e., rewritten) memory cell's resistance can not be modulated any more due to unique physical mechanisms of resistive memories [12, 13, 14]. Hence, a cell suffering from write endurance problem can be regarded as a stuck-at defect from the perspective of storing data.

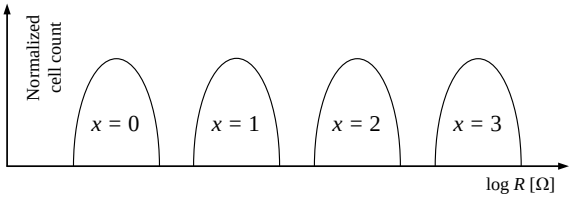
Based on this model of memory with stuck-at defect (i.e., defect channel model), we propose *locally rewritable codes (LWC)*.[1] Inspired by the *repair locality* defined for distributed storage systems, we introduce the *rewriting locality* for resistive memories. When a single stuck-at defect happens, we derive the upper bounds on the writing cost. These upper bounds explicitly show that a small value of rewriting locality can reduce the problems of write endurance and power consumption. Note that a small value of repair locality is preferred for fast repair of single disk node failure.

In addition, we provide the construction of LWC. We show that existing construction methods of locally repairable codes (LRC) can be used to construct LWC due to the *duality* between LRC

[1]LWC instead of LRC is used as the acronym of locally rewritable codes so as to distinguish them from locally repairable codes (LRC).



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



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

Figure 4.1: Cell distribution of resistive memories. Note that $x \in Q = \{0, 1, \ldots, q-1\}$ where $q = 2^b$ represents the channel input of the given channel model.

and LWC. Afterwards, we investigate the LWC with error correcting capability for resistive memories suffering from random errors as well as write endurance problem.

4.2 Basics of Resistive Memories

Resistive memories store data by modulating the resistance of each memory cell. The write operation includes the *set* operation and the *reset* operation. The set operation is the switching event from the high resistance state (HRS) to the low resistance state (LRS). Conversely, the switching event from LRS to HRS is called the reset operation as shown in Fig. 4.1 (a). To read the data from a cell, a small read voltage is applied that does not affect the resistance state of the memory cell to detect whether the cell is lower or higher than the given read level. Hence, the write operation requires much higher power consumption than the read operation [12, 14].

Although the write endurance characteristic of resistive memories is better than flash memories, the write endurance should be improved substantially for the embedded memory applications and storage class memory applications where frequent write operations are required [14].

PCM and RRAM have unique physical mechanisms that result in write endurance problem, which will be explained in the subsequent subsections. However, the outcomes of write endurance problem have commonality that the resistance state of a cell suffering from write endurance problem can not be changed by the set operation or reset operation. From the perspective of storing data, this heavily rewritten cell can be regarded as a stuck-at defect as will be explained in Section 2.2.1.

4.2.1 Phase Change Memories (PCM)

PCM has shown great promise as a storage class memory due to its superior resistance ratio, scalability, and high speed performance. PCM consists of chalcogenide materials like Ge-Sb-Te (GST), which are known to have two stable resistance states [12]. As shown in Fig. 4.2, the LRS corresponds to a crystalline structure of the chalcogenide material, whereas the HRS corresponds to an amorphous structure.

In PCM, the write operations including set and reset operations consume much more power than the read operation. The set operation is brought about by applying a long and low-power heat pulse to the device. The reset operation is done by pulsing the device with a short and high-power heat pulse that melts the chalcogenide, thus amorphizing it. The reset operation consumes the largest power since the cell needs to reach the melting temperature as shown in Fig. 4.2. To read the stored data, the resistance of the cell is measured by passing an electrical current small enough not to disturb the current state [2, 12].

One of the main challenges for the storage class memory application is its limited write endurance. From the point of view of the data, this endurance problems can be interpreted as stuck-at defects. Such stuck-at defects may either appear in as-fabricated devices due to process variations or may be generated during the rewriting (i.e., cycling) process.

The stuck-at defects are classified into: (1) stuck-at LRS defect which corresponds to the cell in LRS being unable to reset to HRS and (2) stuck-at HRS defect which corresponds to the

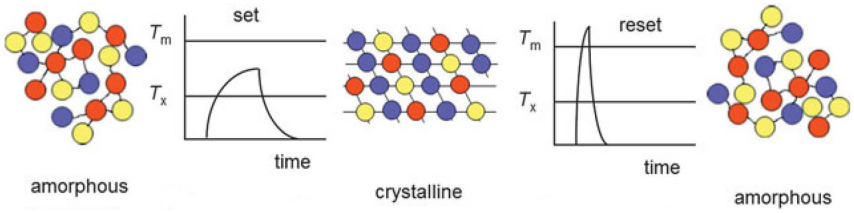


Figure 4.2: Principle of PCM. Starting from the amorphous phase with large resistance, a current pulse is applied. After sufficiently long pulse heats the material above the minimum crystallization temperature T_{x} to crystallize the material, the resistance is low (set operation). After the larger and short pulse is applied to heat the material above the melting temperature T_{m}, the material is melt-quenched and returns to the amorphous (reset operation). Different colors represent different atoms (such as Ge, Sb, and Te in the commonly used GeSbTe compounds) in the phase change materials [2].

cell in HRS incapable of being set to LRS for the same operating conditions. The stuck-at LRS defect is traditionally attributed to the formation of crystallites in the amorphous state that do not melt during reset operation due to local inhomogeneities [76]. This causes the HRS to gradually move towards the LRS with cycling. Similarly, the stuck-at HRS is attributed to the formation of voids in the materials and their eventual agglomeration [77]. This causes the material to become inhomogeneous and often insufficient heating during the set operation.

4.2.2 Resistive Random-Access Memories (RRAM)

RRAM is a resistance change memory that relies on micro-structural change in the material in order to modulate the resistance of each cell. As shown in Fig. 4.3, the RRAM cell consists of a metal–oxide–metal (MOM) stack in which the sub-oxide is typically TaO_x, HfO_x or TiO_x. RRAM can be integrated with conventional complementary metal–oxide–semiconductor (CMOS) in a simple way, using a material set compatible with the conventional CMOS fabrication environment and process temperatures.

The RRAM device has to go through a one-time programming process known as *forming* to become a resistive switching memory. The forming process involves the application of a high voltage pulse that causes the oxide to breakdown and forms a conductive filament shunting the two metal electrodes [14, 78].

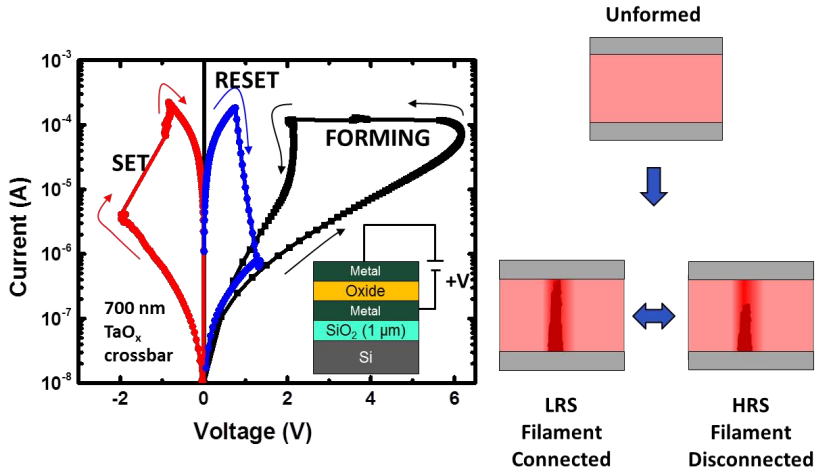


Figure 4.3: Direct current–voltage characteristics of the RRAM device showing forming and switching processes and a physical mechanism of filament formation and dissolution.

The LRS of RRAM memory cell corresponds to the shunted conductive filament. This filament can be disconnected by applying a voltage of the opposite polarity. Once the conductive filament is disconnected, the device resistance increases, and the device is said to be in the HRS. The device can now be cycled between LRS and HRS by applying voltages of opposite polarity as shown in Fig. 4.3.

Similar to PCM, the write operation of RRAM consumes more than an order of magnitude power compared to the read operation. This is due to the read operation being based on capacitive sensing, in which the bitline capacitor is discharged through the resistive memory cell and thus needing only relative references, making read a low-power process.

RRAM also suffers from write endurance problem [14, 79, 80]. The stuck-at LRS defects have been attributed to the widening of the conductive filament [81]. Once the filament widens, the device resistance drops and the reset power is insufficient to disconnect the filament. This causes the cell to be permanently set to LRS. The widening of the filament is thought of as a stochastic increase in the number of oxygen vacancies in the filament during the set and forming operation. It can be explained by an incomplete retraction of oxygen vacancies during the previous reset [82]. Similarly, the device can also suffer from a stuck-at HRS defect if the de-

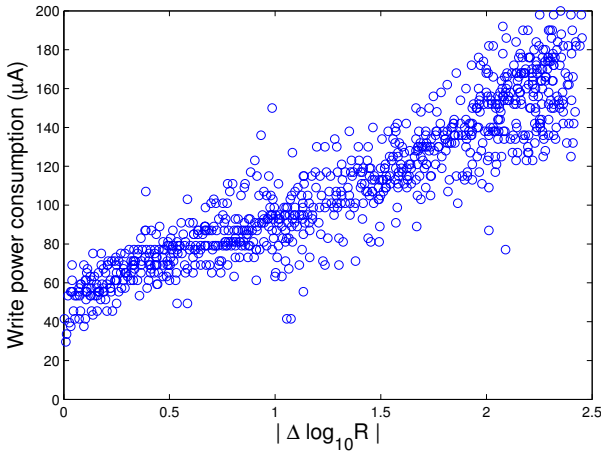


Figure 4.4: Relation between resistance change and write power consumption. The RRAM cells were 200 nm TiN-TaO_x-TiN stack and show reliable resistive switching behavior. These cells were fabricated at Carnegie Mellon University with the same material as in [3].

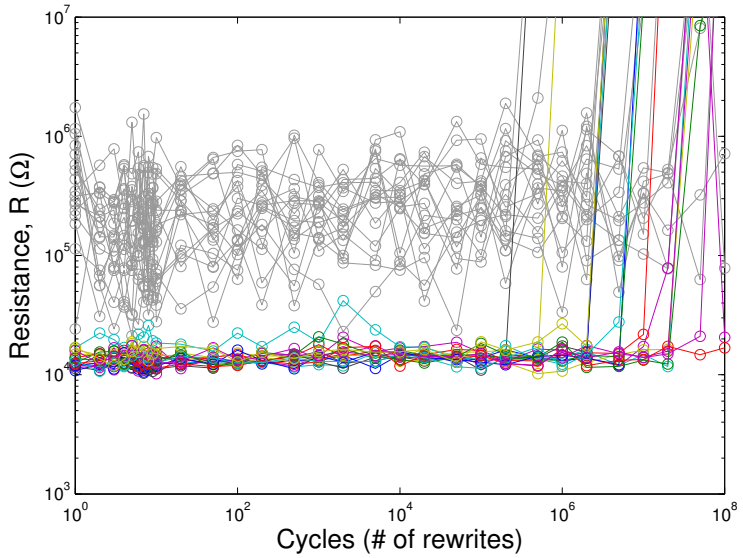
vice undergoes over-reset [83]. In this process, the oxygen vacancies are retracted irreversibly, making the device stuck-at HRS defect.

4.3 Writing Cost

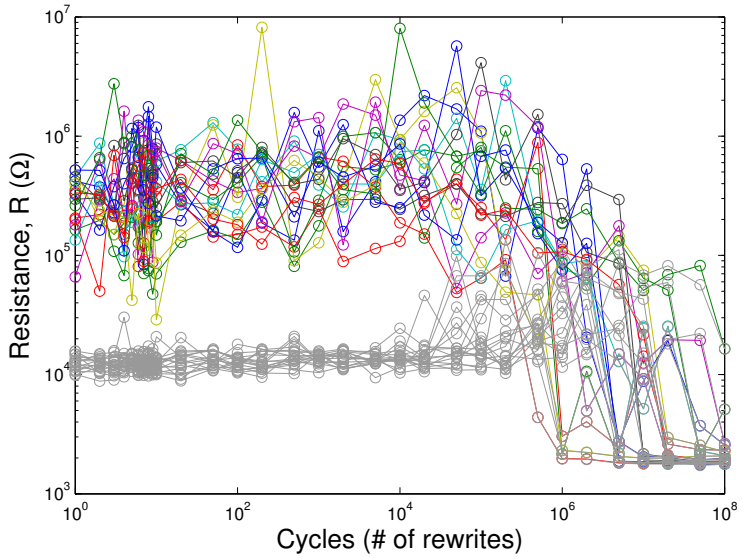
We define two writing costs: *rewriting cost* and *initial writing cost* which are related to power consumption and write endurance. Based on experimental data from real RRAM devices, we explain why high writing costs are harmful to write endurance and increase power consumption.

4.3.1 Power Consumption and Endurance in RRAM Devices

From the experimental data, we investigate the characteristics of write power consumption and write endurance in RRAM devices. First, we evaluated the increase in power as a function of resistance change. The resistance change implies filament completion and then growth. Recent work [84] demonstrates that devices that experience a higher power switching pulse form a larger filament. Since the finding of [84] indicated that the power density remains nearly constant for different power values, it follows that the change in $\log R$ is directly proportional to the power



(a) Stuck-at HRS



(b) Stuck-at LRS

Figure 4.5: Write endurance characteristics of 85 nm RRAM cells. The raw data of these experimental results came from IMEC.

consumption.

Fig. 4.4 shows an increase in the write power as the device gradually goes from HRS to LRS using 100 ns pulses of increasing amplitude. The x-axis shows a change in $\log R$ of the device. Since the states of resistive memories depend on $\log R$ [12, 14], we can claim that the write power consumption is proportional to the state change in Fig. 4.1 where x-axis denotes $\log R$.

Fig. 4.5 shows the write endurance characteristics of 85 nm RRAM cells. IMEC provided these experimental data which were presented in [80]. Only 17 cells were plotted in Fig. 4.5. However, we obtained experimental data from more than 500 cells and observed similar characteristics.

From Fig. 4.5, we can claim that the probability of defect β depends on the number of rewrites. As the number of rewrites increases, more cells will become stuck-at defects. When the number of rewrites is 10^8, the resistances of 16 cells among 17 cells are higher than 10^5 in Fig. 4.5 (a). Also, 16 cells among 17 cells have lower resistance than 10^4 in Fig. 4.5 (b).

One property of write endurance is that the stuck-at defect is *not* directly proportional to the change in $\log R$. The experimental results in [80] show that the different set operation voltages (e.g., 1.5V and 2.5V) result in similar write endurance failures. Hence, we can claim that the number of rewrites is a valid measure of write endurance.

4.3.2 Writing Cost: Rewriting Cost and Initial Writing Cost

We define the rewriting cost using ℓ_0-norm and ℓ_1-norm.

Definition 4.1 (Rewriting Cost) *Suppose that* $\mathbf{m}$ *was stored by* $\mathbf{x}$ *in* n *cells. If* $\mathbf{x}'$ *is rewritten to these* n *cells to store the updated* $\mathbf{m}'$*, the rewriting costs are given by*

$$\Delta_{\ell_0}(\mathbf{m}, \mathbf{m}') = \|\mathbf{x} - \mathbf{x}'\|_0 \tag{4.1}$$

$$\Delta_{\ell_1}(\mathbf{m}, \mathbf{m}') = \|\mathbf{x} - \mathbf{x}'\|_1 \tag{4.2}$$

where we assume that the channel state vector $\mathbf{s}$ *in* (2.5) *does not change and both* $\mathbf{c}$ *and* $\mathbf{c}'$ *mask stuck-at defects.*

Remark 4.2 $\Delta_{\ell_0}(\mathbf{m}, \mathbf{m}')$ *represents the number of cells to be rewritten whereas* $\Delta_{\ell_1}(\mathbf{m}, \mathbf{m}')$ *corresponds to the sum of resistance state changes during rewriting. Since the write power consumption depends on the change in* $\log R$ *as shown in Fig. 4.4, we can claim that* $\Delta_{\ell_1}(\mathbf{m}, \mathbf{m}')$

is a good measure of write power consumption. With regards to write endurance, experimental results in [80] support that $\Delta_{\ell_0}(\mathbf{m}, \mathbf{m}')$ *is a good measure.*

Example 4.3 *Suppose that* $q = 4$*,* $n = 6$*,* $\mathbf{s} = (3, 0, \lambda, 1, \lambda, \lambda)$*, and* $\mathbf{x} = (3, 0, 2, 1, 0, 1)$*. Also, suppose that* $\mathbf{x}' = (3, 0, 1, 1, 3, 1)$ *is rewritten to these* n *cell without an stuck-at error, i.e.,* $\varepsilon = 0$*. Then,* $\Delta_{\ell_0}(\mathbf{m}, \mathbf{m}') = \| (3, 0, 2, 1, 0, 1) - (3, 0, 1, 1, 3, 1) \|_0 = \| (0, 0, 1, 0, -3, 0) \|_0 = 2$*. Also,* $\Delta_{\ell_1}(\mathbf{m}, \mathbf{m}') = \| (0, 0, 1, 0, -3, 0) \|_1 = 4$*.*

Remark 4.4 *For SLC (i.e.,* $q = 2$*),* $\Delta_{\ell_0}(\mathbf{m}) = \Delta_{\ell_1}(\mathbf{m})$ *and* $\Delta_{\ell_0}(\mathbf{m}, \mathbf{m}') = \Delta_{\ell_1}(\mathbf{m}, \mathbf{m}')$*.*

Definition 4.5 (Initial Writing Cost) *Suppose that* $\mathbf{m}$ *was stored by its codeword* $\mathbf{c}$ *in the initial stage of* n *cells where all the normal cells are set to zeros. The initial writing costs are given by*

$$\Delta_{\ell_0}(\mathbf{m}) = \|\mathbf{x}\|_0 - \|\mathbf{s}_{\mathcal{U}}\|_0 \tag{4.3}$$

$$\Delta_{\ell_1}(\mathbf{m}) = \|\mathbf{x}\|_1 - \|\mathbf{s}_{\mathcal{U}}\|_1 \tag{4.4}$$

where the location of defects $\mathcal{U}$ *was defined in Definition 2.1. Also,* $\|\mathbf{s}_{\mathcal{U}}\|_0$ *denotes the number of stuck-at defects whose stuck-at values are nonzero.* $\|\mathbf{s}_{\mathcal{U}}\|_1$ *represents the* ℓ_1*-norm of stuck-at values among* n *cells.*

We cannot write in defective cells (i.e., stuck-at defects) because their stuck-at values are fixed. If the encoder succeeds to mask stuck-at defects, the stuck-at defects do not affect the initial writing cost. $\Delta_{\ell_0}(\mathbf{m})$ represents the number of normal cells to be written whereas $\Delta_{\ell_1}(\mathbf{m})$ corresponds to the sum of resistance state changes in normal cells during initial writing.

Example 4.6 *Suppose that* $q = 4$*,* $n = 6$*,* $\mathbf{s} = (3, 0, \lambda, 1, \lambda, \lambda)$*. If* $\mathbf{x} = (3, 0, 2, 1, 0, 1)$ *was written without a stuck-at error, i.e.,* $\varepsilon = 0$*, then* $\Delta_{\ell_0}(\mathbf{m}) = \|\mathbf{x}\|_0 - \|\mathbf{s}_{\mathcal{U}}\|_0 = 4 - 2 = 2$*, which is the number of normal cells to be written. Also,* $\Delta_{\ell_1}(\mathbf{m}) = \|\mathbf{x}\|_1 - \|\mathbf{s}_{\mathcal{U}}\|_1 = 7 - 4 = 3$*, which is the sum of state changes in normal cells.*

Remark 4.7 *In general, the rewriting cost is more important than the initial writing cost since most of write operations will be of the rewriting type. If a device has a write endurance limit*

of 10000 cycles, 9999 write operations will be of the rewriting type whereas only the first is the initial write operation (i.e., 0.01%). However, there may be some storage applications such as for archival storage, where the initial writing cost is critical.

4.4 Rewriting Locality and Locally Rewritable Codes

4.4.1 Motivation and a Toy Example

As a toy example, suppose that n-cell *binary* memory has a single stuck-at defect. It is easy to see that this stuck-at defect can be handled by the following simple technique [37].

$$\mathbf{c} = (\mathbf{m}, 0) + \mathbf{1}_n \cdot p \tag{4.5}$$

where $\mathbf{c} \in \mathbb{F}_2^n$, $\mathbf{m} \in \mathbb{F}_2^k$, $p \in \mathbb{F}_2$ and $k = n - 1$. Note that $\mathcal{W}(0) = 0$ and $\mathcal{W}(1) = 1$.

Suppose that i-th cell is a defect whose stuck-at value is $s_i \in \{0, 1\}$. If $i \in [n-1]$ and $s_i = m_i$, or if $i = n$ and $s_n = 0$, then p should be 0. Otherwise, $p = 1$. Thus, p decides whether to flip $\mathbf{m}$ or not. This simple coding scheme is optimal since it achieves the following upper bound in [37] with equality.

$$n - t - \left\lceil \log_2 \ln 2^t \binom{n}{t} \right\rceil \leq \log_2 \mathcal{M} \leq n - t \tag{4.6}$$

where $\mathcal{M}$ is the number of codewords and t is the number of stuck-at defects among n cells. For linear block codes, $k = \log_2 \mathcal{M}$, i.e., $k \leq n - 1$.

If there is no stuck-at defect among n cells, then we can store $\mathbf{m}$ by writing $\mathbf{c} = (\mathbf{m}, 0)$ (i.e., $p = 0$). Now, consider the case when the stored message needs to be updated causing $\mathbf{m}$ to become $\mathbf{m}'$. Usually, $\|\mathbf{m} - \mathbf{m}'\| \ll n$, which happens often when updating files. Instead of storing $\mathbf{m}'$ into another group of n cells, it is more efficient to store $\mathbf{m}'$ by rewriting only $\|\mathbf{m} - \mathbf{m}'\|$ cells. For example, suppose that $m'_i \neq m_i$ for an $i \in [k]$ and $m'_j = m_j$ for all other

$j \in [k] \setminus i$. Then, we can store k bits $\mathbf{m}'$ by rewriting only one cell.

An interesting problem arises when a cell to be rewritten is defective. Suppose that i-th cell is a stuck-at defect whose stuck-at value is s_i. If $s_i = m_i \neq m'_i$, then we should write $\mathbf{c} = (\mathbf{m}, 0)$ for storing $\mathbf{m}$. However, in order to store the updated information $\mathbf{m}'$, we should write $\mathbf{c}' = \overline{\mathbf{c}} = (\overline{\mathbf{m}}, 1)$ where $p = 1$. Thus, $n-1$ cells should be rewritten to update one bit data m'_i without a stuck-at error. The same thing happens when $s_i = m'_i \neq m_i$. When considering write endurance and power consumption, rewriting $n-1$ cells is a high price to pay for dealing with one bit stuck-at error.

In order to relieve this burden, we change (4.5) by introducing an additional parity bit as follows.

$$\begin{aligned}\mathbf{c} &= \left(\mathbf{m}_{[1:\frac{k}{2}]}, 0, \mathbf{m}_{[\frac{k}{2}+1:k]}, 0\right) + G_0\mathbf{p} \\ &= \left(\mathbf{m}_{[1:\frac{k}{2}]}, 0, \mathbf{m}_{[\frac{k}{2}+1:k]}, 0\right) + \begin{bmatrix} \mathbf{1}_{\frac{n}{2}} & \mathbf{0}_{\frac{n}{2}} \\ \mathbf{0}_{\frac{n}{2}} & \mathbf{1}_{\frac{n}{2}} \end{bmatrix} (p_1, p_2) \end{aligned} \tag{4.7}$$

where $k = n-2$. For simplicity, we assume that n is even. Then, $\mathbf{1}_{\frac{n}{2}}$ and $\mathbf{0}_{\frac{n}{2}}$ are all-ones and all-zeros column vectors with $n/2$ elements. By introducing an additional parity bit, we can reduce the number of rewriting cells from $n-1$ to $\frac{n}{2}-1$.

This idea is similar to the concept of Pyramid codes which are the early LRC [85]. For n disk nodes, single parity check codes can repair one node failure (i.e., single erasure) by

$$\mathbf{1}_n^T \widehat{\mathbf{c}} = 0 \tag{4.8}$$

where $\widehat{\mathbf{c}}$ represents the recovered codeword from disk node failures. Assuming that c_i is erased due to a node failure, c_i can be recovered by $\widehat{c}_i = c_i = \sum_{j \in [n] \setminus i} c_j$. For this recovery, we should access $n-1$ disk nodes which degrades the repair speed. For more efficient repair process, we

can add a new parity bit as follows (i.e., $k = n - 2$).

$$H^T\widehat{\mathbf{c}} = \begin{bmatrix} \mathbf{1}_{\frac{n}{2}} & \mathbf{0}_{\frac{n}{2}} \\ \mathbf{0}_{\frac{n}{2}} & \mathbf{1}_{\frac{n}{2}} \end{bmatrix}^T \widehat{\mathbf{c}} = \mathbf{0} \tag{4.9}$$

Then, a failed node c_i can be repaired by accessing only $\frac{n}{2} - 1$ nodes. Note that the repair locality of (4.9) is $\frac{n}{2} - 1$ whereas the repair locality of (4.8) is $n - 1$ which is the simple but effective idea of Pyramid codes.

An interesting observation is that G_0 of (4.7) is the same as H of (4.9). In addition, the number of resistive memory cells to be rewritten is equal to the number of disk nodes to be accessed in distributed storage systems. This observation will be further discussed in Section 4.5.

4.4.2 Rewriting Locality and Locally Rewritable Codes

First, we define the write function as follows.

Definition 4.8 (Write Function) *For a bijective function $\mathcal{W} : \mathbb{F}_q \to Q$, $\mathcal{W}$ is called* write function *if*

$$\mathcal{W}(0) = 0. \tag{4.10}$$

Lemma 4.9 *If $\mathcal{W}$ is a write function and $\mathcal{W}(\mathbf{c}) = \mathbf{x}$, then $\|\mathbf{x}\|_0 = \|\mathbf{c}\|_0$.*

Proof: $\|\mathbf{x}\|_0 = \mathsf{supp}(\mathcal{W}(\mathbf{c})) = \mathsf{supp}(\mathbf{c}) = \|\mathbf{c}\|_0$ where $\mathsf{supp}(\mathcal{W}(\mathbf{c})) = \mathsf{supp}(\mathbf{c})$ because of $\mathcal{W}(c) \neq 0$ for any $c \neq 0$ by Definition 4.8. ■

Next, we show the following simple lemma based on Lemma 4.9, Definition 4.5, and Definition 4.1.

Lemma 4.10 *If $\mathbf{m}$ is encoded by* (2.23)*, the initial writing cost $\Delta_{\ell_0}(\mathbf{m}, \mathbf{m}')$ is given by*

$$\Delta_{\ell_0}(\mathbf{m}) = \|(\mathbf{m}, \mathbf{0}) + \mathbf{c}_0\|_0 - \|\mathbf{s}_{\mathcal{U}}\|_0. \tag{4.11}$$

If $\mathbf{m}$ *is updated to* $\mathbf{m}'$ *and both are encoded by* (2.23)*, then the rewriting cost* $\Delta_{\ell_0}(\mathbf{m}, \mathbf{m}')$ *is given by*

$$\Delta_{\ell_0}(\mathbf{m}, \mathbf{m}') = \|(\mathbf{m} - \mathbf{m}', \mathbf{0}) + (\mathbf{c}_0 - \mathbf{c}_0')\|_0 \,. \tag{4.12}$$

Proof: By Lemma 4.9 and Definition 4.5, we can show that the initial writing cost $\Delta_{\ell_0}(\mathbf{m})$ is given by

$$\Delta_{\ell_0}(\mathbf{m}) = \|\mathbf{x}\|_0 - \|\mathbf{s}_{\mathcal{U}}\|_0 = \|\mathbf{c}\|_0 - \|\mathbf{s}_{\mathcal{U}}\|_0 = \|(\mathbf{m}, \mathbf{0}) + \mathbf{c}_0\|_0 - \|\mathbf{s}_{\mathcal{U}}\|_0.$$

The rewriting cost $\Delta_{\ell_0}(\mathbf{m}, \mathbf{m}')$ is given by

$$\Delta_{\ell_0}(\mathbf{m}, \mathbf{m}') = \|\mathbf{x} - \mathbf{x}'\|_0 = \|\mathbf{c} - \mathbf{c}'\|_0 \tag{4.13}$$

$$= \|(\mathbf{m} - \mathbf{m}', \mathbf{0}) + (\mathbf{c}_0 - \mathbf{c}_0')\|_0 \tag{4.14}$$

where (4.13) follows from Definition 4.1 and Lemma 4.9. Also, (4.14) follows from (2.23). ■

Now we introduce the *information rewriting locality* and *parity rewriting locality* which affect initial writing cost and rewriting cost.

Definition 4.11 (Information Rewriting Locality) *Suppose that* m_i *for* $i \in [k]$*, i.e., information (message) part, should be updated to* $m_i' \neq m_i$ *and the corresponding* i*-th cell is a stuck-at defect. If* m_i *can be updated to* m_i' *without a stuck-at error by rewriting at most* $r^\star$ *other cells, then the* i*-th coordinate has* information rewriting locality $r^\star$.

Lemma 4.12 *If the* i*-th coordinate for* $i \in [k]$ *has information rewriting locality* $r^\star$*, then there exists* $\mathbf{c}_0 \in \mathcal{C}_0$ *such that* $i \in \mathsf{supp}(\mathbf{c}_0)$ *and* $\|\mathbf{c}_0\|_0 \leq r^\star + 1$.

Proof: For $\mathbf{m}$ and $\mathbf{m}'$, suppose that $m_i \neq m_i'$ for an $i \in [k]$ and $m_j = m_j'$ for all other $j \in [k] \setminus i$. Note that i-th cell is a stuck-at defect whose stuck-at value is s_i. We should consider the following cases: 1) $\mathcal{W}(m_i) \neq \mathcal{W}(m_i') = s_i$, 2) $\mathcal{W}(m_i') \neq \mathcal{W}(m_i) = s_i$, and 3) $\mathcal{W}(m_i) \neq \mathcal{W}(m_i')$, $\mathcal{W}(m_i) \neq s_i$ and $\mathcal{W}(m_i') \neq s_i$.

For $\mathcal{W}(m_i) \neq \mathcal{W}(m_i') = s_i$, $\mathbf{c} = (\mathbf{m}, \mathbf{0}_{n-k}) + \mathbf{c}_0$ and $\mathbf{c}' = (\mathbf{m}', \mathbf{0}_{n-k})$ where $\mathcal{W}(m_i + c_{0,i}) = \mathcal{W}(m_i') = s_i$ and $i \in \mathsf{supp}(\mathbf{c}_0)$ to mask this i-th coordinate's defect by (2.23). By contraposition, suppose that $\min \|\mathbf{c}_0'\|_0 \geq r^\star + 2$. The information rewriting locality of i is at least $r^\star + 1$ since

$$\Delta_{\ell_0}(\mathbf{m}, \mathbf{m}') = \|(\mathbf{m} - \mathbf{m}', \mathbf{0}) + \mathbf{c}_0\|_0 \tag{4.15}$$

$$= \|(c_{0,1}, \cdots, m_i - m_i' + c_{0,i}, \cdots, c_{0,n})\|_0 \tag{4.16}$$

$$= \left\|\mathbf{c}_{0\setminus i}\right\|_0 \geq r^\star + 1. \tag{4.17}$$

where (4.15) follows from (4.12) and $\mathbf{c}_0' = \mathbf{0}$. (4.16) follows from $m_j = m_j'$ for all other $j \in [k] \setminus i$. (4.17) follows from $m_i + c_{0,i} = m_i'$ and the assumption of $\min \|\mathbf{c}_0\|_0 \geq r^\star + 2$.

For $\mathcal{W}(m_i') \neq \mathcal{W}(m_i) = s_i$, the proof is similar.

For $m_i \neq m_i'$, $\mathcal{W}(m_i) \neq s_i$ and $\mathcal{W}(m_i') \neq s_i$, $\mathbf{c} = (\mathbf{m}, \mathbf{0}_{n-k}) + \mathbf{c}_0$ and $\mathbf{c}' = (\mathbf{m}', \mathbf{0}_{n-k}) + \mathbf{c}_0'$. We can pick $\mathbf{c}_0' = \alpha \mathbf{c}_0$ (where $\alpha \in \mathbb{F}_q$) such that $i \in \mathsf{supp}(\mathbf{c}_0) = \mathsf{supp}(\mathbf{c}_0')$ and $\mathcal{W}(m_i + c_{0,i}) = \mathcal{W}(m_i' + c_{0,i}') = s_i$. By constraposition, suppose that $\min \|\mathbf{c}_0\|_0 = \min \|\mathbf{c}_0'\|_0 \geq r^\star + 2$. Then we can show that the information rewriting locality of i-th coordinate is at least $r^\star + 1$ by the similar method. ■

Definition 4.13 (Parity Rewriting Locality) *Suppose that only one nonzero symbol m_i for $i \in [k]$ should be stored to the initial stage of n cells (i.e., all the cells are set to zeros) and there is a stuck-at defect in the parity location j for $j \in [k+1:n]$ and $s_j \neq 0$. If m_i can be stored without a stuck-at error by writing at most $r^\star + 1$ cells, then the j-th coordinate has* parity rewriting locality $r^\star$.

Lemma 4.14 *If the j-th coordinate for $j \in [k+1:n]$ has parity rewriting locality $r^\star$, then there exists $\mathbf{c}_0 \in \mathcal{C}_0$ such that $j \in \mathsf{supp}(\mathbf{c}_0)$ and $\|\mathbf{c}_0\|_0 \leq r^\star + 1$.*

Proof: Suppose that $\mathbf{m}$ should be stored to the initial stage of n cells where $m_i \neq 0$ for an $i \in [k]$ and $m_{i'} = 0$ for all other $i' \in [k] \setminus i$. By (2.23), $\mathbf{c} = (\mathbf{m}, \mathbf{0}_{n-k}) + \mathbf{c}_0$ such that $j \in \mathsf{supp}(\mathbf{c}_0)$ and $\mathcal{W}(c_{0,j}) = s_j$ to mask this j-th coordinate's defect. Hence, $\left\|\mathbf{c}_{\setminus j}\right\|_0$ represents the number of

cells to be written because of Lemma 4.9 and $\mathcal{W}(c_j) = s_j$.

By contraposition, suppose that $\min \|\mathbf{c}'_0\|_0 \geq r^\star + 2$. If $i \notin \mathsf{supp}(\mathbf{c}_0)$ for $i \in [k]$, then we should write both m_i and $\|\mathbf{c}_{0\setminus j}\|_0 \geq r^\star + 1$, i.e., at least $r^\star + 2$ cells.

For the parity rewriting locality $r^\star$, there should exist $\mathbf{c}_0$ such that $j \in \mathsf{supp}(\mathbf{c}_0)$ and $\|\mathbf{c}_0\|_0 \leq r^\star + 1$ in order to mask this j-th coordinate defect. If $i \in \mathsf{supp}(\mathbf{c}_0)$, then it is possible to store m_i without a stuck-at error by writing $\|\mathbf{c}_{0\setminus j}\|_0 \leq r^\star$ cells because $\mathcal{W}(c_j) = \mathcal{W}(c_{0,j}) = s_j$. If $i \notin \mathsf{supp}(\mathbf{c}_0)$, we should write both m_i and $\|\mathbf{c}_{0\setminus j}\|_0$, i.e., at most $r^\star + 1$ cells. ■

Definition 4.15 (Rewriting Locality) *If both information rewriting locality and parity rewriting locality are $r^\star$, then we define that the rewriting locality is $r^\star$.*

Based on the definition of rewriting locality, locally rewritable codes (LWC) are defined as follows.

Definition 4.16 (Locally Rewritable Codes) *An $(n, k, d^\star, r^\star)$ LWC code is an (n, k) code with minimum distance $d^\star$ and rewriting locality $r^\star$.*

As the repair locality of LRC is meaningful only for single disk failure, the rewriting locality is valid when there is only one stuck-at defect among n cells. In distributed storage systems, the most common case is a single node failure among n nodes [85].

As shown in Fig. 4.5, below a certain number of rewrites (e.g., the write endurance limit recommended by memory manufacturers), we can expect that β is considerably low and the number of defects among n cells is usually zero or one. Hence, a small value of rewriting locality can suppress the increase in the number of rewrites. If multiple stuck-at defects happen after many rewrites, rewriting locality is not a good measure. Nevertheless, the LWC guarantees masking up to $d^\star - 1$ stuck-at defects by (2.25).

4.4.3 Upper Bounds on Rewriting Cost and Initial Writing Cost

We derive the upper bounds on rewriting cost and initial writing cost when a single stuck-at defect occurs. These upper bounds show that rewriting locality $r^\star$ is an important parameter for

write endurance and power consumption.

First, we derive the upper bound on rewriting cost and initial writing cost based on ℓ_0-norm.

Theorem 4.17 *Suppose that* $\mathbf{m}$ *is updated to* $\mathbf{m}'$ *by LWC with rewriting locality* $r^\star$*. If there is a single stuck-at defect in* n *cells, then the rewriting cost* $\Delta_{\ell_0}(\mathbf{m}, \mathbf{m}')$ *is given by*

$$\Delta_{\ell_0}(\mathbf{m}, \mathbf{m}') \leq \|\mathbf{m} - \mathbf{m}'\|_0 + r^\star - 1. \tag{4.18}$$

Proof: The proof is given in Appendix 4.7. ■

Corollary 4.18 *If* $\mathbf{m}$ *is stored in the initial stage of* n *cells with a single stuck-at defect, then the initial writing cost* $\Delta_{\ell_0}(\mathbf{m})$ *is given by*

$$\Delta_{\ell_0}(\mathbf{m}) \leq \|\mathbf{m}\|_0 + r^\star. \tag{4.19}$$

Proof: The proof is given in Appendix 4.8. ■

Now we derive the upper bounds on rewriting cost and initial writing cost based on ℓ_1-norm, which are given by (4.2) and (4.4).

Theorem 4.19 *Suppose that* $\mathbf{m}$ *is updated to* $\mathbf{m}'$ *by LWC with rewriting locality* $r^\star$*. If there is a single stuck-at defect in* n *cells, then the rewriting cost* $\Delta_{\ell_1}(\mathbf{m}, \mathbf{m}')$ *is given by*

$$\Delta_{\ell_1}(\mathbf{m}, \mathbf{m}') \leq \|\mathcal{W}(\mathbf{m}) - \mathcal{W}(\mathbf{m}')\|_1 + (q-1)r^\star - 1. \tag{4.20}$$

Proof: The proof is given in Appendix 4.9. ■

Corollary 4.20 *If* $\mathbf{m}$ *is stored in the initial stage of* n *cells with a single stuck-at defect, then the initial writing cost* $\Delta_{\ell_1}(\mathbf{m})$ *is given by*

$$\Delta_{\ell_1}(\mathbf{m}) \leq \|\mathcal{W}(\mathbf{m})\|_1 + (q-1)r^\star. \tag{4.21}$$

Proof: The proof is given in Appendix 4.10. ■

For $q = 2$, note that (4.20) and (4.21) are equivalent to (4.18) and (4.19), respectively.

We can observe that a smaller value of rewriting locality $r^\star$ reduces these upper bounds on rewriting cost and initial writing cost. From Theorem 4.17 and Corollary 4.18, we can claim that a smaller $r^\star$ can reduce the number of cells to be rewritten (or initially written). Also, Theorem 4.19 and Corollary 4.20 show that a smaller $r^\star$ decreases the sum of resistance state changes during rewriting (or initial writing). Hence, these upper bounds show that the LWC with a small rewriting locality $r^\star$ can improve write endurance and power consumption.

4.5 Constructions of Locally Rewritable Codes

In this section, we investigate the construction of LWC. We point out the *duality* between LRC and LWC, which indicates that existing construction methods of LRC can be used to construct LWC. Afterwards, we investigate the LWC with error correcting capability for resistive memories suffering from random errors as well as write endurance problem.

4.5.1 Locally Repairable Codes

We briefly summarize LRC which is an important new class of codes for distributed storage systems [85, 86]. An (n, k, d, r) LRC is a code of length n with information (message) length k, minimum distance d, and repair locality r. If a symbol in the LRC-coded data is lost due to a single node failure, its value can be repaired (i.e., reconstructed) by accessing at most r other symbols [85, 86].

One way to ensure fast repair is to use low repair locality such that $r \ll k$ at the cost of decreased minimum distance d. The relation between d and r is shown to be [86]

$$d \leq n - k - \left\lceil \frac{k}{r} \right\rceil + 2. \tag{4.22}$$

It is worth mentioning that this upper bound is a generalization of the Singleton bound. The LRC

achieving this bound with equality are called *optimal*. Recently, constructions and properties of LRC have been investigated [85, 86, 87, 88, 89, 90, 91, 92, 93].

4.5.2 Construction of Locally Rewritable Codes

We investigate construction of $(n, k, d^\star, r^\star)$ LWC. We will observe the duality between cyclic LRC and cyclic LWC, which allows us to construct LWC by using the existing construction methods of LRC. First, we define cyclic LWC as follows.

Definition 4.21 *If $\mathcal{C}_0$ is cyclic, then the LWC is called* cyclic.

Lemma 4.22 *Let $\mathcal{C}_0$ denote a cyclic code whose minimum distance is d_0. Then, corresponding cyclic LWC's rewriting locality is $r^\star = d_0 - 1$.*

Proof: Due to the property of cyclic codes, there exists $\mathbf{c}_0 \in \mathcal{C}_0$ such that $i \in \mathsf{supp}(\mathbf{c}_0)$ and $\|\mathbf{c}_0\| = d_0$ for any $i \in [n]$. Thus, the rewriting locality is at most $r^\star = d_0 - 1$. ■

From the definition of $d^\star$ in (2.25), $d^\star = d_0^\perp$ which is the minimum distance of $\mathcal{C}_0^\perp$ (namely, dual code of $\mathcal{C}_0$). Thus, the parameters of cyclic LWC are given by

$$(d^\star, r^\star) = (d_0^\perp, d_0 - 1). \tag{4.23}$$

In [91, 92], an equivalent relation for cyclic LRC was given by

$$(d, r) = (d, d^\perp - 1). \tag{4.24}$$

Comparing (4.23) and (4.24), we observe the duality between LRC and LWC. This duality is important since it indicates that we can construct LWC using existing construction methods of LRC as shown in the following theorem.

Theorem 4.23 *Suppose that $H_{LRC} \in \mathbb{F}_q^{n\times(n-k)}$ is the parity check matrix of cyclic LRC $\mathcal{C}_{LRC}$ with*

$(d, r) = (d, d^{\perp} - 1)$. *By setting* $G_0 = H_{LRC}$*, we can construct cyclic LWC* $\mathcal{C}_{LWC}$ *with*

$$(d^{\star}, r^{\star}) = (d, d^{\perp} - 1). \tag{4.25}$$

Proof: By setting $G_0 = H_{\text{LRC}}$, the LWC's codeword $\mathbf{c} \in \mathcal{C}_{\text{LWC}}$ is given by

$$\mathbf{c} = (\mathbf{m}, \mathbf{0}) + H_{\text{LRC}} \cdot \mathbf{p}. \tag{4.26}$$

From the definition of $d^{\star}$ in (2.25), $d^{\star}$ is given by

$$d^{\star} = \min_{\substack{\mathbf{x} \neq \mathbf{0} \\ H_{\text{LRC}}^{T} \mathbf{x} = \mathbf{0}}} \|\mathbf{x}\|$$

which is equivalent to the minimum distance d of $\mathcal{C}_{\text{LRC}}$. Hence,

$$d^{\star} = d_0^{\perp} = d. \tag{4.27}$$

From (4.23) and (4.27), $r^{\star} = d_0 - 1 = d^{\perp} - 1$. ■

Remark 4.24 (Optimal Cyclic LWC) *Theorem 4.23 shows that the optimal cyclic* (n, k, d, r) *LRC can be used to construct the optimal cyclic* $(n, k, d^{\star}, r^{\star})$ *LWC such that*

$$d^{\star} = n - k - \left\lceil \frac{k}{r^{\star}} \right\rceil + 2. \tag{4.28}$$

Hence, the optimal LWC can be constructed from the optimal LRC.

Remark 4.25 (LWC Bound) *From Theorem 4.23 and Remark 4.24, we can claim the following bound for LWC.*

$$d^{\star} \leq n - k - \left\lceil \frac{k}{r^{\star}} \right\rceil + 2 \tag{4.29}$$

which is equivalent to the bound for LRC given by (4.22).

Table 4.1: Duality of LRC and LWC

	(n, k, d, r) LRC	$(n, k, d^\star, r^\star)$ LWC
Application	Distributed storage systems (system level)	Resistive memories (physical level)
Channel	Erasure channel	Defect channel
Encoding	$\mathbf{c} = G_{\text{LRC}}\mathbf{m}$	$\mathbf{c} = (\mathbf{m}, \mathbf{0}) + H_{\text{LRC}}\mathbf{p}$
Decoding	$H_{\text{LRC}}^T\widehat{\mathbf{c}} = \mathbf{0}$	$G_{\text{LRC}}^T\mathbf{c} = \widehat{\mathbf{m}}$
Bound	$d \leq n - k - \left\lceil \frac{k}{r} \right\rceil + 2$	$d^\star \leq n - k - \left\lceil \frac{k}{r^\star} \right\rceil + 2$
Trade-off	d (reliability) vs. r (repair efficiency)	$d^\star$ (reliability) vs. r^* (rewriting efficiency)

In Table 4.1, the dual properties of LRC and LWC are summarized. For the LRC, there is a trade-off between minimum distance d and repair locality r. In order to obtain a smaller r for fast repair, we should decrease d which degrades the reliability of stored data. For the LWC, this trade-off exists between minimum distance $d^\star$ and rewriting locality $r^\star$. At the cost of $d^\star$, we can improve write endurance and power consumption of resistive memories. Both trade-off relations for LRC and LWC can be described by the same bound due to this duality.

4.5.3 Locally Rewritable Codes with Error Correcting Capability

Although write endurance and power consumption are important problems of resistive memories, we should consider random errors also due to variability and several physical mechanisms, especially for MLC resistive memories.

Definition 4.26 (LWC with Error Correcting Capability) *An $(n, k, d^\star, r^\star, \widetilde{t}\,)$ LWC code is defined as $(n, k, d^\star, r^\star)$ LWC with random-error correcting capability of $\widetilde{t}$.*

We can construct $(n, k, d^\star, r^\star, \widetilde{t}\,)$ LWC codes based on partitioned cyclic codes in Section 2.2.4 and the duality of LRC and LWC as follows:

1. By using the duality between LRC and LWC, set $g_0(x) = h_{\text{LRC}}(x)$ where $h_{\text{LRC}}(x)$ is the parity polynomial of $(n, n - l, d, r)$ LRC. Note that l denotes the redundancy for masking

defects and $\deg(g_0(x)) = \deg(h_{\text{LRC}}(x)) = n - l$.

2. Among generator polynomials of $(n, k + l, \widetilde{d} = 2\widetilde{t} + 1)$ codes, choose $g(x)$ satisfying $g(x) \mid h_{\text{LRC}}(x)$ of (2.28) where $\deg(g(x)) = n - k - l$.

3. From (2.29), $c(x) = m(x)g(x) + p(x)h_{\text{LRC}}(x)$.

Corollary 4.27 *From the above construction, we can construct an* $(n, k, d^\star = d, r^\star \leq \widetilde{d}+r-1, \widetilde{t})$ *LWC where* $\widetilde{d} = 2\widetilde{t} + 1$.

Proof: Due to the properties of partitioned cyclic codes, $d^\star$ depends on only $g_0(x)$. If $g_0(x) = h_{\text{LRC}}(x)$, then $d^\star = d$. Also, $\widetilde{t}$ is decided by $g(x)$ for $g(x) \mid h_{\text{LRC}}(x)$ which means that $g(x) \mid c(x) = m(x)g(x) + p(x)h_{\text{LRC}}(x)$.

The rewriting locality $r^\star$ is decided by $g(x)$ and $h_{\text{LRC}}(x)$. Let $[0 : n-1]$ instead of $[n]$ denote the coordinate of a codeword for polynomial notation. Suppose that m_i for $i \in [0 : k-1]$, i.e., information (message) part, should be updated to $m_i' \neq m_i$ and the corresponding i-th cell is a stuck-at s_i defect and $m_j = m_j'$ for all other $j \in [0 : k-1] \setminus i$. We can claim that

$$\begin{aligned}
\|c(x) - c'(x)\|_0 &= \|(m(x) - m'(x))g(x) + (p(x) - p'(x))h_{\text{LRC}}(x)\|_0 && (4.30)\\
&\leq \|(m_i - m_i')x^i \cdot g(x)\|_0 + \|(p_v - p_v')x^v \cdot h_{\text{LRC}}(x)\|_0 - 2 && (4.31)\\
&= \|g(x)\|_0 + \|h_{\text{LRC}}(x)\|_0 - 2 && (4.32)\\
&\leq \widetilde{d} + r - 1 && (4.33)
\end{aligned}$$

where (4.31) follows from $m_i g_0 x^i + p_v h_{\text{LRC},i-v} x^i = m_i' g_0 x^i + p_v' h_{\text{LRC},i-v} x^i = s_i x^i$ for $i \geq v$ where g_0 denotes the constant term of $g(x)$. Due to the properties of cyclic codes, we can choose $\|g(x)\|_0 = \widetilde{d}$ and $\|h_{\text{LRC}}(x)\|_0 = r + 1$. By Definition 4.11 and (4.33), the information rewriting locality is at most $\widetilde{d} + r - 1$.

Suppose that only one nonzero symbol m_i should be stored to the initial stage of n cells and there is a stuck-at defect in the parity location j for $j \in [k : n-1]$. Then,

$$\|c(x)\|_0 = \|m_i x^i g(x) + p_v x^v h_{\text{LRC}}(x)\|_0 \tag{4.34}$$

$$\leq \|g(x)\|_0 + \|h_{\text{LRC}}(x)\|_0 - 1 \tag{4.35}$$

$$\leq \widetilde{d} + r \tag{4.36}$$

where (4.35) follows from $p_v h_{\text{LRC},j-v} x^j = s_j x^j$. Due to the properties of cyclic codes, we can choose $\|g(x)\|_0 = \widetilde{d}$ and $\|h_{\text{LRC}}(x)\|_0 = r+1$. By Definition 4.13 and (4.36), the parity rewriting locality is at most $\widetilde{d} + r - 1$. ■

Fact 4.28 *From the definitions in* (2.25) *and* (2.27)*, an* $(n, k, d^\star, r^\star, \widetilde{t})$ *LWC code guarantees masking up to* $d^\star - 1$ *stuck-at defects and correcting up to* $\widetilde{t}$ *random errors where* $\widetilde{t} = \left\lfloor \frac{\widetilde{d}-1}{2} \right\rfloor$ *(*$\lfloor x \rfloor$ *is the largest integer not greater than* x*).*

Candidates for $g(x)$ can be constructed by standard code construction methods such as BCH codes and Reed-Solomon (RS) codes. We should choose a proper $g(x)$ among these candidates to satisfy the condition of PLBC, i.e., $g(x) \mid h_{\text{LRC}}(x)$. By using the property of partitioned cyclic codes [40], we can claim the following relation between $g(x)$ and $g_{\text{LRC}}(x)$.

Fact 4.29 *Since* $g_{LRC}(x)h_{LRC}(x) = x^n - 1$ *by definition of parity polynomial, we can claim that* $g(x)$ *and* $g_{LRC}(x)$ *should not share any common roots in order to satisfy* $g(x) \mid h_{LRC}(x)$*.*

Example 4.30 *We can construct a binary* $(n, k, d^\star, r^\star, \widetilde{t}) = (15, 6, 2, 4, 1)$ *LWC where* $l = 5$*. Note that the parity size for correcting random errors is* $n - k - l = 4$*. In [90], the parity polynomial of* $(n = 15, n - l = 10, d = 2, r = 2)$ *LRC is given by* $h_{LRC}(x) = x^{10} + x^5 + 1$*. First set* $g_0(x) = h_{LRC}(x)$*. For* $\widetilde{t} = 1$*, we can choose a generator polynomial* $g(x) = x^4 + x + 1$ *such that* $g(x) \mid h_{LRC}(x)$ *by* $(n = 15, k + l = 11, \widetilde{d} = 3)$ *BCH code construction. Note that* $r^\star \leq \widetilde{d} + r - 1 = 4$*. Then, the codeword of LWC with* $(n, k, d^\star, r^\star, \widetilde{t}) = (15, 6, 2, 4, 1)$ *is given by* $c(x) = (x^4 + x + 1)m(x) + (x^{10} + x^5 + 1)p(x)$*.*

4.6 Conclusion

Inspired by LRC for distributed storage systems, we proposed LWC to improve write endurance and power consumption of resistive memories. We derived the upper bounds on rewriting cost and initial writing cost showing that a small value of rewriting cost is helpful for write endurance and power consumption. Also, we pointed out the duality between LRC and LWC, which makes it possible to construct LWC by using existing construction methods of LRC. In addition, we investigated the constructions of LWC with error correcting capability for random errors.

4.7 Proof of Theorem 4.17

First, suppose that the single defect's coordinate is $i \in [k]$ and its stuck-at value is s_i. If $m_i = m'_i$, then $\mathbf{c} = (\mathbf{m}, \mathbf{0}) + \mathbf{c}_0$ and $\mathbf{c}' = (\mathbf{m}', \mathbf{0}) + \mathbf{c}_0$. By Lemma 4.10 and $\mathbf{c}'_0 = \mathbf{c}_0$, $\Delta_{\ell_0}(\mathbf{m}, \mathbf{m}') = \|\mathbf{m} - \mathbf{m}'\|_0$.

If $\mathcal{W}(m_i) \neq \mathcal{W}(m'_i) = s_i$, then $\mathbf{c} = (\mathbf{m}, \mathbf{0}) + \mathbf{c}_0$ and $\mathbf{c}' = (\mathbf{m}', \mathbf{0})$. Note that $i \in \mathsf{supp}(\mathbf{c}_0)$ to mask the stuck-at defect in i-th coordinate. Then,

$$\begin{aligned}
\Delta_{\ell_0}(\mathbf{m}, \mathbf{m}') &= \|(\mathbf{m} - \mathbf{m}', \mathbf{0}) + \mathbf{c}_0\|_0 = \|(\mathbf{m} - \mathbf{m}', \mathbf{0})_{\setminus i} + (\mathbf{c}_0)_{\setminus i}\|_0 && (4.37)\\
&\leq \|(\mathbf{m} - \mathbf{m}', \mathbf{0})_{\setminus i}\|_0 + \|(\mathbf{c}_0)_{\setminus i}\|_0 && (4.38)\\
&= \|\mathbf{m} - \mathbf{m}'\|_0 + \|\mathbf{c}_0\|_0 - 2 && (4.39)\\
&\leq \|\mathbf{m} - \mathbf{m}'\|_0 + r^\star - 1 && (4.40)
\end{aligned}$$

where (4.37) follows from $m_i - m'_i + c_{0,i} = 0$ (i.e., $\mathcal{W}(m_i + c_{0,i}) = \mathcal{W}(m'_i) = s_i$). Also, (4.39) follows from $m_i \neq m'_i$ and $c_{0,i} \neq 0$ and (4.40) follows from Lemma 4.12.

For $\mathcal{W}(m'_i) \neq \mathcal{W}(m_i) = s_i$, we can show (4.18) by a similar method.

If $m_i \neq m'_i$, $\mathcal{W}(m_i) \neq s_i$ and $\mathcal{W}(m'_i) \neq s_i$, then $\mathbf{c} = (\mathbf{m}, \mathbf{0}) + \mathbf{c}_0$ and $\mathbf{c}' = (\mathbf{m}', \mathbf{0}) + \mathbf{c}'_0$.

Both $\mathbf{c}$ and $\mathbf{c}'$ can mask the defect by setting $\mathbf{c}'_0 = \alpha \mathbf{c}_0$ where $\alpha \in \mathbb{F}_q$ and $\alpha \neq 1$. Then,

$$\Delta_{\ell_0}(\mathbf{m}, \mathbf{m}') = \|(\mathbf{m} - \mathbf{m}', \mathbf{0}) + (1-\alpha)\mathbf{c}_0\|_0 = \|(\mathbf{m} - \mathbf{m}', \mathbf{0})_{\setminus i} + (1-\alpha)(\mathbf{c}_0)_{\setminus i}\|_0 \tag{4.41}$$

$$\leq \|(\mathbf{m} - \mathbf{m}', \mathbf{0})_{\setminus i}\|_0 + \|(\mathbf{c}_0)_{\setminus i}\|_0 \tag{4.42}$$

$$\leq \|\mathbf{m} - \mathbf{m}'\|_0 + r^\star - 1 \tag{4.43}$$

where (4.41) follows from $m_i - m'_i + (1-\alpha)c_{0,i} = 0$ (i.e., $\mathcal{W}(m_i + c_{0,i}) = \mathcal{W}(m'_i + \alpha c_{0,i}) = s_i$).

Next, suppose that the single defect's coordinate is $i \in [k+1 : n]$. Since $\mathbf{c} = (\mathbf{m}, \mathbf{0}) + \mathbf{c}_0$ and $\mathbf{c}' = (\mathbf{m}', \mathbf{0}) + \mathbf{c}_0$, the rewriting cost is $\Delta_{\ell_0}(\mathbf{m}, \mathbf{m}') = \|\mathbf{m} - \mathbf{m}'\|_0$.

4.8 Proof of Corollary 4.18

First, suppose that the single defect's coordinate is $i \in [k]$ and its stuck-at value is s_i. If $\mathcal{W}(m_i) = s_i$, then $\mathbf{c} = (\mathbf{m}, \mathbf{0})$. From (4.11) and $\mathbf{c}_0 = \mathbf{0}$,

$$\Delta_{\ell_0}(\mathbf{m}) = \|\mathbf{m}\|_0 - \|s_i\|_0 \leq \|\mathbf{m}\|_0. \tag{4.44}$$

If $\mathcal{W}(m_i) \neq s_i = 0$ (i.e., $\|\mathbf{s}_{\mathcal{U}}\|_0 = \|s_i\|_0 = 0$), then

$$\Delta_{\ell_0}(\mathbf{m}) = \|(\mathbf{m}, \mathbf{0}) + \mathbf{c}_0\|_0 = \|(\mathbf{m}, \mathbf{0})_{\setminus i} + (\mathbf{c}_0)_{\setminus i}\|_0 \tag{4.45}$$

$$\leq \|\mathbf{m}\|_0 + \|\mathbf{c}_0\|_0 - 2 \tag{4.46}$$

$$\leq \|\mathbf{m}\|_0 + r^\star - 1 \tag{4.47}$$

where (4.45) follows from $\mathcal{W}(m_i + c_{0,i}) = s_i = 0$. Also, (4.46) follows from $m_i \neq 0$ and $c_{0,i} \neq 0$ and (4.47) follows from Lemma 4.12.

If $\mathcal{W}(m_i) \neq s_i \neq 0$ (i.e., $\|\mathbf{s}_{\mathcal{U}}\|_0 = \|s_i\|_0 = 1$), then

$$\Delta_{\ell_0}(\mathbf{m}) = \|(\mathbf{m}, \mathbf{0}) + \mathbf{c}_0\|_0 - 1 \leq \|\mathbf{m}\|_0 + \|\mathbf{c}\|_0 - 1 \leq \|\mathbf{m}\|_0 + r^\star \tag{4.48}$$

where (4.48) follows from Lemma 4.12.

Next suppose that the single defect's coordinate is $j \in [k+1 : n]$. If $s_j = 0$, then $\mathbf{c} = (\mathbf{m}, \mathbf{0})$. Hence, $\Delta_{\ell_0}(\mathbf{m}) = \|\mathbf{m}\|_0$. If $s_j \neq 0$, then $\mathbf{c} = (\mathbf{m}, \mathbf{0}) + \mathbf{c}_0$. We can show that $\Delta_{\ell_0}(\mathbf{m}) \leq \|\mathbf{m}\|_0 + r^\star$ by the similar method of (4.48).

4.9 Proof of Theorem 4.19

Let $\mathcal{U} = \{i\}$ denote the defect location. Also, we define $\mathcal{J} = \{j : m_j \neq m'_j, \text{for } j \in [k]\}$ where $|\mathcal{J}| = \|\mathbf{m} - \mathbf{m}'\|_0$.

First, suppose that the single defect's coordinate is $i \in [k]$ and its stuck-at value is s_i. If $m_i = m'_i$, then $i \notin \mathcal{J}$ and $i \in \mathsf{supp}(\mathbf{c}_0)$. By additive encoding, $\mathbf{c} = (\mathbf{m}, \mathbf{0}) + \mathbf{c}_0$ and $\mathbf{c}' = (\mathbf{m}', \mathbf{0}) + \mathbf{c}'_0 = (\mathbf{m}', \mathbf{0}) + \mathbf{c}_0$ where $\mathbf{c}'_0 = \mathbf{c}_0$, which means that $\mathcal{W}(c_j) = \mathcal{W}(c'_j)$ for any $j \notin \mathcal{J}$. The rewriting cost is given by

$$\Delta_{\ell_1}(\mathbf{m}, \mathbf{m}') = \|\mathbf{x} - \mathbf{x}'\|_1 = \|\mathcal{W}(\mathbf{c}) - \mathcal{W}(\mathbf{c}')\|_1 = \sum_{j \in \mathcal{J}} \left|\mathcal{W}(c_j) - \mathcal{W}(c'_j)\right|. \tag{4.49}$$

If $\mathcal{J} \cap \mathsf{supp}(\mathbf{c}_0) = \emptyset$, then

$$\Delta_{\ell_1}(\mathbf{m}, \mathbf{m}') = \sum_{j \in \mathcal{J}} \left|\mathcal{W}(c_j) - \mathcal{W}(c'_j)\right| = \sum_{j \in \mathcal{J}} \left|\mathcal{W}(m_j) - \mathcal{W}(m'_j)\right| = \|\mathcal{W}(\mathbf{m}) - \mathcal{W}(\mathbf{m}')\|_1 . \tag{4.50}$$

Otherwise (i.e., $\mathcal{J} \cap \mathsf{supp}(\mathbf{c}_0) \neq \emptyset$),

$$\begin{aligned}\Delta_{\ell_1}(\mathbf{m}, \mathbf{m}') &= \sum_{j \in \mathcal{J} \setminus \mathsf{supp}(\mathbf{c}_0)} \left|\mathcal{W}(m_j) - \mathcal{W}(m'_j)\right| + \sum_{j \in \mathcal{J} \cap \mathsf{supp}(\mathbf{c}_0)} \left|\mathcal{W}(m_j + c_{0,j}) - \mathcal{W}(m'_j + c_{0,j})\right| \\ &\leq \left\|\mathcal{W}(\mathbf{m}) - \mathcal{W}(\mathbf{m}')\right\|_1 + (q-1)r^\star - 1 \qquad (4.51)\end{aligned}$$

where (4.51) follows from

$$\sum_{j \in \mathcal{J} \setminus \mathsf{supp}(\mathbf{c}_0)} \left|\mathcal{W}(m_j) - \mathcal{W}(m'_j)\right| \leq \left\|\mathcal{W}(\mathbf{m}) - \mathcal{W}(\mathbf{m}')\right\|_1 - 1 \qquad (4.52)$$

because of $|\mathcal{J} \setminus \mathsf{supp}(\mathbf{c}_0)| < |\mathcal{J}|$ and $\left|\mathcal{W}(m_j) - \mathcal{W}(m'_j)\right| \geq 1$ for $j \in \mathcal{J}$. Also,

$$\sum_{j \in \mathcal{J} \cap \mathsf{supp}(\mathbf{c}_0)} \left|\mathcal{W}(m_j + c_{0,j}) - \mathcal{W}(m'_j + c_{0,j})\right| \leq (q-1)r^\star$$

where $1 \leq |\mathcal{J} \cap \mathsf{supp}(\mathbf{c}_0)| \leq r^\star$ because of Lemma 4.12 and the fact that $i \notin \mathcal{J}$ and $i \in \mathsf{supp}(\mathbf{c}_0)$ for the defect location $i \in [k]$.

If $\mathcal{W}(m_i) \neq \mathcal{W}(m'_i) = s_i$, then $\mathbf{c} = (\mathbf{m}, \mathbf{0}) + \mathbf{c}_0$ and $\mathbf{c}' = (\mathbf{m}', \mathbf{0})$ where $\mathbf{c}'_0 = \mathbf{0}$. It is clear that $i \in \mathcal{J}$ and $i \in \mathsf{supp}(\mathbf{c}_0)$. Hence, $i \in \mathcal{J} \cap \mathsf{supp}(\mathbf{c}_0)$ and $\mathcal{W}(c_i) = \mathcal{W}(c'_i)$, which means that $m_i + c_{0,i} = m'_i$. Then,

$$\begin{aligned}\Delta_{\ell_1}(\mathbf{m}, \mathbf{m}') = \left\|\mathcal{W}(\mathbf{c}) - \mathcal{W}(\mathbf{c}')\right\|_1 &= \sum_{j \in \mathcal{J} \cup \mathsf{supp}(\mathbf{c}_0)} \left|\mathcal{W}(c_j) - \mathcal{W}(c'_j)\right| \\ &= \sum_{j \in \mathcal{J} \setminus \mathsf{supp}(\mathbf{c}_0)} \left|\mathcal{W}(m_j) - \mathcal{W}(m'_j)\right| + \sum_{j \in \mathsf{supp}(\mathbf{c}_0) \setminus \{i\}} \left|\mathcal{W}(c_j) - \mathcal{W}(c'_j)\right| \\ &\leq \left\|\mathcal{W}(\mathbf{m}) - \mathcal{W}(\mathbf{m}')\right\|_1 + (q-1)r^\star - 1 \qquad (4.53)\end{aligned}$$

where (4.53) follows from (4.52) and $|\mathsf{supp}(\mathbf{c}_0) \setminus \{i\}| = r^\star$.

For $\mathcal{W}(m'_i) \neq \mathcal{W}(m_i) = s_i$, the proof is similar to (4.53).

If $\mathcal{W}(m_i) \neq \mathcal{W}(m'_i)$, $\mathcal{W}(m_i) \neq s_i$ and $\mathcal{W}(m'_i) \neq s_i$, then $\mathbf{c} = (\mathbf{m}, \mathbf{0}) + \mathbf{c}_0$ and $\mathbf{c}' =$

$(\mathbf{m}', \mathbf{0}) + \mathbf{c}'_0$. By setting $\mathbf{c}'_0 = \alpha \mathbf{c}_0$ for $\alpha \in \mathbb{F}_q$, $\mathsf{supp}(\mathbf{c}_0) = \mathsf{supp}(\mathbf{c}'_0)$. Note that $i \in \mathcal{J} \cap \mathsf{supp}(\mathbf{c}_0)$ and $\mathcal{W}(c_i) = \mathcal{W}(c'_i)$. Then,

$$\begin{aligned}
\Delta_{\ell_1}(\mathbf{m}, \mathbf{m}') &= \|\mathcal{W}(\mathbf{c}) - \mathcal{W}(\mathbf{c}')\|_1 \\
&= \sum_{j \in \mathcal{J} \setminus \mathsf{supp}(\mathbf{c}_0)} \left|\mathcal{W}(m_j) - \mathcal{W}(m'_j)\right| + \sum_{j \in \mathsf{supp}(\mathbf{c}_0) \setminus \{i\}} \left|\mathcal{W}(c_j) - \mathcal{W}(c'_j)\right| \\
&\leq \|\mathcal{W}(\mathbf{m}) - \mathcal{W}(\mathbf{m}')\|_1 + (q-1)r^\star - 1 \qquad (4.54)
\end{aligned}$$

where (4.54) can be shown by the same way of (4.53).

Next, suppose that the single defect's coordinate is $i \in [k+1 : n]$ where $i \notin \mathcal{J}$ and $i \in \mathsf{supp}(\mathbf{c}_0)$. Note that $\mathbf{c} = (\mathbf{m}, \mathbf{0}) + \mathbf{c}_0$ and $\mathbf{c}' = (\mathbf{m}', \mathbf{0}) + \mathbf{c}_0$. The rewriting cost is given by

$$\Delta_{\ell_1}(\mathbf{m}, \mathbf{m}') = \sum_{j \in \mathcal{J}} \left|\mathcal{W}(c_j) - \mathcal{W}(c'_j)\right| \leq \|\mathcal{W}(\mathbf{m}) - \mathcal{W}(\mathbf{m}')\|_1 + (q-1)r^\star - 1 \qquad (4.55)$$

where (4.55) can be shown by the same claim of (4.50) and (4.51).

From (4.50), (4.51), (4.53), (4.54), and (4.55), we can claim that (4.20) is true.

4.10 Proof of Corollary 4.20

Let $\mathcal{U} = \{i\}$ denote the defect location and we define $\mathcal{J} = \{j : m_j \neq 0 \text{ for } j \in [k]\}$.

First suppose that the single defect's coordinate is $i \in [k]$ and its stuck-at value is s_i. If $\mathcal{W}(m_i) = s_i$, then $\mathbf{c} = (\mathbf{m}, \mathbf{0})$. By (4.4), the initial writing cost is given by

$$\Delta_{\ell_1}(\mathbf{m}) = \|\mathcal{W}(\mathbf{c})\|_1 - s_i = \|\mathcal{W}(\mathbf{m})\|_1 - s_i \leq \|\mathcal{W}(\mathbf{m})\|_1. \qquad (4.56)$$

If $\mathcal{W}(m_i) \neq s_i$, then $\mathcal{W}(c_i) = \mathcal{W}(m_i + c_{i,0}) = s_i$ by additive encoding $\mathbf{c} = (\mathbf{m}, \mathbf{0}) + \mathbf{c}_0$.

$$\begin{aligned}\Delta_{\ell_1}(\mathbf{m}) = \|\mathcal{W}(\mathbf{c})\|_1 - s_i &= \sum_{j \in \mathcal{J} \cup \mathsf{supp}(\mathbf{c}_0)} |\mathcal{W}(c_j)| - s_i \\ &= \sum_{j \in \mathcal{J} \setminus \mathsf{supp}(\mathbf{c}_0)} |\mathcal{W}(m_j)| + \sum_{j \in \mathsf{supp}(\mathbf{c}_0) \setminus \{i\}} |\mathcal{W}(c_j)| \qquad (4.57) \\ &\leq \|\mathcal{W}(\mathbf{m})\|_1 + (q-1)r^\star \qquad (4.58)\end{aligned}$$

where (4.57) follows from $\mathcal{W}(c_i) = \mathcal{W}(m_i + c_{i,0}) = s_i$. Also, (4.58) follows from Lemma 4.14.

Suppose that the single defect's coordinate is $i \in [k+1 : n]$. If $s_i = 0$, then $\mathbf{c} = (\mathbf{m}, \mathbf{0})$ and

$$\Delta_{\ell_1}(\mathbf{m}) = \|\mathcal{W}(\mathbf{c})\|_1 = \|\mathcal{W}(\mathbf{m})\|_1. \qquad (4.59)$$

If $s_i \neq 0$, then $\mathcal{W}(c_i) = \mathcal{W}(c_{i,0}) = s_i$ by additive encoding $\mathbf{c} = (\mathbf{m}, \mathbf{0}) + \mathbf{c}_0$ for $i \in [k+1 : n]$.

$$\begin{aligned}\Delta_{\ell_1}(\mathbf{m}) = \sum_{j \in \mathcal{J} \cup \mathsf{supp}(\mathbf{c}_0)} |\mathcal{W}(c_j)| - s_i &= \sum_{j \in \mathcal{J} \setminus \mathsf{supp}(\mathbf{c}_0)} |\mathcal{W}(m_j)| + \sum_{j \in \mathsf{supp}(\mathbf{c}_0) \setminus \{i\}} |\mathcal{W}(c_j)| \\ &\leq \|\mathcal{W}(\mathbf{m})\|_1 + (q-1)r^\star. \qquad (4.60)\end{aligned}$$

From (4.56), (4.58), (4.59), and (4.60), we can claim that $\Delta(\mathbf{m}) \leq \|\mathcal{W}(\mathbf{m})\|_1 + (q-1)r^\star$.

Chapter 5

Conclusion

5.1 Thesis Contributions

The main contributions of this dissertation are channel coding schemes that use side information corresponding physical phenomena in flash memories and resistive memories. This dissertation included (1) coding techniques for the model of memory with stuck-at defects; (2) coding schemes for combating and harnessing the ICI of flash memories; and (3) coding schemes that improve write endurance and power consumption of resistive memories.

First, we investigated the coding techniques for memory with stuck-at defects. The idea of coding for memory with stuck-at defects was employed to handle the problems of flash memories and resistive memories. The contributions of this dissertation to the study of coding techniques for memory with stuck-at defects are as follows:

- Formulating the redundancy allocation problem for memory suffering from permanent stuck-at defects and transient errors.
- Deriving the upper bound on the probability of decoding failure and proposing techniques to determine the optimal redundancy allocation based on this upper bound.
- Investigating the relations between binary erasure channel (BEC), memory with stuck-at

defects, binary erasure quantization (BEQ), and write-once memory (WOM).

Based on the coding techniques for the model of memory with stuck-at defects, we proposed coding schemes for 2D planar flash memories and 3D vertical flash memories. The coding for memory with stuck-at defects was employed to combat ICI, which is a primary challenge of 2D planar flash memories. Also, we proposed a coding scheme that reduces the effect of fast detrapping, a degradation factor in 3D vertical flash memories. The contributions of this dissertation to the study of coding schemes for flash memories are as follows:

- Proposing a coding scheme that combats the ICI in 2D planar flash memories.
- Developing a coding scheme that harnesses the intentional ICI to reduce the effect of fast detrapping in 3D vertical flash memories.
- Based on the unique property of asymmetry between write and erase operations of flash memory, bridging the well-known Gelfand-Pinsker problems: writing on dirty paper and memory with stuck-at defects.

The final part of this dissertation proposed locally rewritable codes (LWC) to improve write endurance and power consumption of resistive memories. The dissertation contributions in this topic are:

- Proposing LWC for resistive memories.
- Defining a novel parameter of rewriting locality and showing that a small value of rewriting locality of LWC is able to reduce the problems of write endurance and write power consumption.
- Showing the relation between LWC and locally repairable codes (LRC), which indicates that existing LRC construction methods can be applied to construct LWC.

5.2 Future Work

There are many interesting research topics that may come from this dissertation. Here, we discuss some of future research work.

In this dissertation, we focused on the redundancy allocation of algebraic codes such as PBCH codes. The redundancy allocation problem of LDPC codes and polar codes is an important topic. The analysis based on error exponent may be helpful for this redundancy allocation problem.

Write efficient memory (WEM) in [94] is an important rewriting model to handle write endurance problem. WEM has drawn attention for PCM [95, 96]. The WEM model assumes that the encoder knows the previously written data, which is also a Gelfand-Pinsker problem. Although the WEM differs from LWC because WEM does not consider stuck-at defects, the connection between WEM and LWC can be an interesting research topic.

Recently, LRC with multiple repair alternatives (i.e., availability) has been investigated to manage hot data in distributed storage systems [97, 98]. We can consider LWC with multiple rewriting alternatives for resistive memories. When we mask a single defect in resistive memories, LWC with multiple rewriting alternatives can improve write endurance.

Although we focused on flash memories and resistive memories in this dissertation, there are various emerging memories such as ferroelectric RAM (FRAM or FeRAM), magnetic RAM (MRAM), spin transfer torque RAM (STTRAM), carbon nanotube RAM, and polymer memory. The strong demand for denser, faster, and more reliable storage media motivates the exploration of these emerging memories. Since each emerging memory technology stores the data based on different underlying mechanisms, the emerging memories can have uniques advantages as well as challenges to be addressed.

The fundamental idea in this dissertation could be applied to other emerging memory technologies: the side information corresponding to adverse phenomena can provide more efficient coding techniques for memories. Hence, there will be many research opportunities regarding

obtaining the side information efficiently and developing coding techniques to properly handle adverse phenomena in emerging memories.

Bibliography

[1] K.-T. Park *et al.*, "Three-dimensional 128Gb MLC vertical NAND Flash-memory with 24-WL stacked layers and 50MB/s high-speed programming," in *Proc. IEEE Int. Solid-State Circuits Conf. Dig. Tech. Pap. (ISSCC)*, Feb. 2014, pp. 334–335.

[2] S. Raoux, F. Xiong, M. Wuttig, and E. Pop, "Phase change materials and phase change memory," *MRS Bulletin*, vol. 39, pp. 703–710, Aug. 2014.

[3] T. C. Jackson, A. A. Sharma, J. A. Bain, J. A. Weldon, and L. Pileggi, "Oscillatory neural networks based on TMO nano-oscillators and multi-level RRAM cells," *IEEE Trans. Emerg. Sel. Topics Circuits Syst.*, vol. 5, no. 2, pp. 230–241, Jun. 2015.

[4] K. Prall, "Scaling non-volatile memory below 30nm," in *Proc. 22nd IEEE Non-Volatile Semiconductor Memory Workshop*, Aug. 2007, pp. 5–10.

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

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

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

[8] W. Jeong *et al.*, "A 128 Gb 3b/cell V-NAND flash memory with 1 Gb/s I/O rate," *IEEE J. Solid-State Circuits*, vol. 51, no. 1, pp. 204–212, Jan. 2016.

[9] C.-P. Chen, H.-T. Lue, C.-C. Hsieh, K.-P. Chang, and C.-Y. Lu, "Study of fast initial charge loss and it's impact on the programmed states Vt distribution of charge-trapping NAND Flash," in *Proc. IEEE Int. Electron Devices Meet. (IEDM)*, Dec. 2010, pp. 5.6.1–5.6.4.

[10] Y. Kim, J. Kim, J. J. Kong, B. V. K. Vijaya Kumar, and X. Li, "Verify level control criteria for multi-level cell flash memories and their applications," *EURASIP J. on Adv. in Signal Process.*, vol. 2012, no. 1, pp. 1–13, 2012.

[11] S. Raoux *et al.*, "Phase-change random access memory: A scalable technology," *IBM Journal of Research and Development*, vol. 52, no. 4.5, pp. 465–479, Jul. 2008.

[12] H.-S. P. Wong, S. Raoux, S. Kim, J. Liang, J. P. Reifenberg, B. Rajendran, M. Asheghi, and K. E. Goodson, "Phase change memory," *Proc. IEEE*, vol. 98, no. 12, pp. 2201–2227, Dec. 2010.

[13] G. W. Burr *et al.*, "Phase change memory technology," *J. Vac. Sci. Technol. B*, vol. 28, no. 2, pp. 223–262, Mar. 2010.

[14] H.-S. P. Wong, H.-Y. Lee, S. Yu, Y.-S. Chen, Y. Wu, P.-S. Chen, B. Lee, F. T. Chen, and M.-J. Tsai, "Metal–Oxide RRAM," *Proc. IEEE*, vol. 100, no. 6, pp. 1951–1970, Jun. 2012.

[15] M. A. Caldwell, S. Raoux, R. Y. Wang, H.-S. P. Wong, and D. J. Milliron, "Synthesis and size-dependent crystallization of colloidal germanium telluride nanoparticles," *J. Mater. Chem.*, vol. 20, no. 7, pp. 1285–1291, Nov. 2010.

[16] M.-J. Lee *et al.*, "Electrical manipulation of nanofilaments in transition-metal oxides for resistance-based memory," *Nano Lett.*, vol. 9, no. 4, pp. 1476–1481, Mar. 2009.

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

[18] K.-T. Park, M. Kang, D. Kim, S.-W. Hwang, B.-Y. Choi, Y.-T. Lee, C. Kim, and K. Kim, "A zeroing cell-to-cell interference page architecture with temporary LSB storing and parallel

MSB program scheme for MLC NAND flash memories," *IEEE J. Solid-State Circuits*, vol. 43, no. 4, pp. 919–928, Apr. 2008.

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

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

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

[22] J. Wang, K. Vakilinia, T.-Y. Chen, T. Courtade, G. Dong, T. Zhang, H. Shankar, and R. Wesel, "Enhanced precision through multiple reads for LDPC decoding in flash memories," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 5, pp. 880–891, May 2014.

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

[24] D. Park and J. Lee, "Floating-gate coupling canceller for multi-level cell nand flash," *IEEE Trans. Magn.*, vol. 47, no. 3, pp. 624–628, March 2011.

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

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

[27] A. A. Jiang, H. Zhou, J. Bruck, and Z. Wang, "Patterned cells for phase change memories," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2011, pp. 2333–2337.

[28] Y. Kim, B. V. K. Vijaya Kumar, K. L. Cho, H. Son, J. Kim, J. J. Kong, and J. Lee, "Modulation coding for flash memories," in *Proc. IEEE Int. Conf. Comput., Netw. Commun. (ICNC)*,

San Diego, CA, Jan. 2013, pp. 961–967.

[29] M. Qin, E. Yaakobi, and P. H. Siegel, "Time–space constrained codes for phase-change memories," *IEEE Trans. Inf. Theory*, vol. 59, no. 8, pp. 5102–5114, Aug 2013.

[30] ——, "Constrained codes that mitigate inter-cell interference in read/write cycles for flash memories," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 5, pp. 836–846, May 2014.

[31] C. E. Shannon, "Channels with side information at the transmitter," *IBM J. Res. Develop.*, vol. 2, no. 4, pp. 289–293, 1958.

[32] S. I. Gelfand and M. S. Pinsker, "Coding for channel with random parameters," *Probl. Contr. and Inf. Theory*, vol. 9, no. 1, pp. 19–31, 1980.

[33] G. Keshet, Y. Steinberg, and N. Merhav, "Channel coding in the presence of side information," *Found. Trends in Commun. and Inf. Theory*, vol. 4, no. 6, pp. 445–586, 2007.

[34] A. El Gamal and Y.-H. Kim, *Network Information Theory*. Cambridge, U.K.: Cambridge University Press, 2011.

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

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

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

[38] C. Heegard and A. El Gamal, "On the capacity of computer memory with defects," *IEEE Trans. Inf. Theory*, vol. 29, no. 5, pp. 731–739, Sep. 1983.

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

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

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

[42] H. Mahdavifar and A. Vardy, "Explicit capacity achieving codes for defective memories," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2015, pp. 641–645.

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

[44] A. V. Kuznetsov and A. J. Han Vinck, "On the general defective channel with informed encoder and capacities of some constrained memories," *IEEE Trans. Inf. Theory*, vol. 40, no. 6, pp. 1866–1871, Nov. 1994.

[45] L. A. Lastras-Montano, A. Jagmohan, and M. M. Franceschini, "Algorithms for memories with stuck cells," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Austin, TX, USA, Jun. 2010, pp. 968–972.

[46] A. Jagmohan, L. A. Lastras-Montano, M. M. Franceschini, M. Sharma, and R. Cheek, "Coding for multilevel heterogeneous memories," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Cape Town, South Africa, May 2010, pp. 1–6.

[47] Y. Kim and B. V. K. Vijaya Kumar, "Coding for memory with stuck-at defects," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Budapest, Hungary, Jun. 2013, pp. 4347–4352.

[48] A. N. Jacobvitz, R. Calderbank, and D. J. Sorin, "Coset coding to extend the lifetime of memory," in *Proc. 19th IEEE Int. Symp. on High Performance Comput. Architecture (HPCA)*, Shenzhen, China, Feb. 2013, pp. 222–233.

[49] Y. Kim and B. V. K. Vijaya Kumar, "Redundancy allocation of partitioned linear block codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Istanbul, Turkey, Jul. 2013, pp. 2374–2378.

[50] ——, "Writing on dirty flash memory," in *Proc. 52nd Annu. Allerton Conf. Commun., Control, Comput.*, Monticello, IL, USA, Oct. 2014, pp. 513–520.

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

[52] Y. Kim and B. V. K. Vijaya Kumar, "On the duality of erasures and defects," *arXiv preprint arXiv:1403.1897*, vol. abs/1403.1897, 2014. [Online]. Available: http://arxiv.org/abs/1403.1897

[53] S. Kudekar, S. Kumar, M. Mondelli, H. D. Pfister, E. Şaşoğlu, and R. Urbanke, "Reed-Muller codes achieve capacity on erasure channels," *arXiv preprint arXiv:1601.04689*, Jan. 2016. [Online]. Available: http://arxiv.org/abs/1601.04689

[54] E. Martinian and J. S. Yedidia, "Iterative quantization using codes on graphs," in *Proc. 41st Annu. Allerton Conf. Commun., Control, Comput.*, Monticello, IL, USA, Oct. 2003, pp. 1–10.

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

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

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

[58] E. Hwang, B. Narayanaswamy, R. Negi, and B. V. K. Vijaya Kumar, "Iterative cross-entropy encoding for memory systems with stuck-at errors," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Houston, TX, USA, Dec. 2011, pp. 1–5.

[59] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. Amsterdam, The Netherlands: North-Holland, 1977, pp. 282–283.

[60] P. Elias, "Coding for two noisy channels," in *Proc. 3rd London Symp. Inf. Theory*, London, U.K., 1955, pp. 61–76.

[61] A. Shokrollahi, "Raptor codes," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2551–2567,

Jun. 2006.

[62] T. Richardson and R. Urbanke, *Modern coding theory*. New York, NY, USA: Cambridge University Press, 2008.

[63] S. B. Korada and R. L. Urbanke, "Polar codes are optimal for lossy source coding," *IEEE Trans. Inf. Theory*, vol. 56, no. 4, pp. 1751–1768, Apr. 2010.

[64] A. Jiang, "On the generalization of error-correcting WOM codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Nice, France, Jun. 2007, pp. 1391–1395.

[65] E. Yaakobi, S. Kayser, P. H. Siegel, A. Vardy, and J. K. Wolf, "Codes for write-once memories," *IEEE Trans. Inf. Theory*, vol. 58, no. 9, pp. 5985–5999, Sep. 2012.

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

[67] E. En Gad, W. Huang, Y. Li, and J. Bruck, "Rewriting flash memories by message passing," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Hong Kong, Jun. 2015, pp. 646–650.

[68] Y. Kim, R. Mateescu, S.-H. Song, Z. Bandic, and B. V. K. Vijaya Kumar, "Coding scheme for 3D vertical flash memory," in *Proc. IEEE Int. Conf. Commun. (ICC)*, London, UK, Jun. 2015, pp. 1861–1867.

[69] G.-C. Zhu, W. Yu, and F. R. Kschischang, "Finite-geometry low-density parity-check codes for channels with stuck-at defects," in *Canadian Workshop on Inform. Theory*, 2005.

[70] E. Hwang, R. Negi, and B. V. K. Vijaya Kumar, "Additive encoding low-density parity-check (AE-LDPC) codes for two-dimensional magnetic recording (TDMR)," in *Proc. IEEE Int. Conf. Comput., Netw. Commun. (ICNC)*, Maui, HI, Feb. 2012, pp. 481–485.

[71] K.-D. Suh *et al.*, "A 3.3 V 32 Mb NAND flash memory with incremental step pulse programming scheme," *IEEE J. Solid-State Circuits*, vol. 30, no. 11, pp. 1149–1156, Nov. 1995.

[72] J. Moon, J. No, S. Lee, S. Kim, S. Choi, and Y. Song, "Statistical characterization of noise

and interference in NAND flash memory," *IEEE Trans. Circuits Syst. I*, vol. 60, no. 8, pp. 2153–2164, Aug. 2013.

[73] M. Helm *et al.*, "A 128Gb MLC NAND-Flash device using 16nm planar cell," in *Proc. IEEE Int. Solid-State Circuits Conf. Dig. Tech. Pap. (ISSCC)*, San Francisco, CA, USA, Feb. 2014, pp. 326–327.

[74] T. Parnell, N. Papandreou, T. Mittelholzer, and H. Pozidis, "Modelling of the threshold voltage distributions of sub-20nm NAND flash memory," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Austin, TX, USA, Dec. 2014, pp. 2351–2356.

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

[76] S. Lee, G. Kim, S. Hong, S. J. Baik, H. Hori, and D.-h. Ahn, "Enhanced cycling endurance in phase change memory via electrical control of switching induced atomic migration," in *Proc. 14th Annu. Non-Volatile Memory Techn. Symp. (NVMTS)*, Oct. 2014, pp. 1–3.

[77] C.-F. Chen *et al.*, "Endurance improvement of Ge2Sb2Te5-based phase change memory," in *Proc. IEEE Int. Memory Workshop (IMW)*, May 2009, pp. 1–2.

[78] A. A. Sharma, M. Noman, M. Abdelmoula, M. Skowronski, and J. A. Bain, "Electronic instabilities leading to electroformation of binary metal oxide-based resistive switches," *Adv. Functional Mater.*, vol. 24, no. 35, pp. 5522–5529, Jul. 2014.

[79] Y. Y. Chen, L. Goux, S. Clima, B. Govoreanu, R. Degraeve, G. S. Kar, A. Fantini, G. Groeseneken, D. J. Wouters, and M. Jurczak, "Endurance/retention trade-off on HfO_2/metal cap 1T1R bipolar RRAM," *IEEE Trans. Electron Devices*, vol. 60, no. 3, pp. 1114–1121, Mar. 2013.

[80] A. Fantini *et al.*, "Engineering of Hf1–xAlxOy amorphous dielectrics for high-performance RRAM applications," in *2014 IEEE 6th International Memory Workshop (IMW)*, may 2014, pp. 1–4.

[81] E. Yalon, A. A. Sharma, M. Skowronski, J. A. Bain, D. Ritter, and I. V. Karpov, "Thermometry of filamentary RRAM devices," *IEEE Trans. Electron Devices*, vol. 62, no. 9, pp. 2972–2977, Sep. 2015.

[82] J. Kwon, A. A. Sharma, J. A. Bain, Y. N. Picard, and M. Skowronski, "Oxygen vacancy creation, drift, and aggregation in TiO_2-based resistive switches at low temperature and voltage," *Adv. Functional Mater.*, vol. 25, no. 19, pp. 2876–2883, Mar. 2015.

[83] H. Y. Lee *et al.*, "Evidence and solution of over-RESET problem for HfO_x based resistive memory with sub-ns switching speed and high endurance," in *Proc. IEEE Int. Electron Devices Meeting (IEDM)*, Dec. 2010, pp. 19.7.1–19.7.4.

[84] J. Kwon, A. A. Sharma, C.-Y. Chen, A. Fantini, M. Jurczak, A. A. Herzing, J. A. Bain, Y. N. Picard, and M. Skowronski, "Transient thermometry and high-resolution analysis of filamentary resistive switches," *ACS Nano*, 2006, submitted.

[85] C. Huang, M. Chen, and J. Li, "Pyramid codes: Flexible schemes to trade space for access efficiency in reliable data storage systems," in *Proc. IEEE Int. Symp. Netw. Comput. Appl. (NCA)*, Jul. 2007, pp. 79–86.

[86] P. Gopalan, C. Huang, H. Simitci, and S. Yekhanin, "On the locality of codeword symbols," *IEEE Trans. Inf. Theory*, vol. 58, no. 11, pp. 6925–6934, Nov. 2012.

[87] N. Silberstein, A. S. Rawat, O. O. Koyluoglu, and S. Vishwanath, "Optimal locally repairable codes via rank-metric codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2013, pp. 1819–1823.

[88] I. Tamo, D. S. Papailiopoulos, and A. G. Dimakis, "Optimal locally repairable codes and connections to matroid theory," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2013, pp. 1814–1818.

[89] I. Tamo and A. Barg, "A family of optimal locally recoverable codes," *IEEE Trans. Inf. Theory*, vol. 60, no. 8, pp. 4661–4676, Aug. 2014.

[90] S. Goparaju and R. Calderbank, "Binary cyclic codes that are locally repairable," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2014, pp. 676–680.

[91] P. Huang, E. Yaakobi, H. Uchikawa, and P. H. Siegel, "Cyclic linear binary locally repairable codes," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Apr. 2015, pp. 1–5.

[92] I. Tamo and A. Barg, "Cyclic LRC codes and their subfield subcodes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2015, pp. 1262–1266.

[93] N. Silberstein and A. Zeh, "Optimal binary locally repairable codes via anticodes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2015, pp. 1247–1251.

[94] R. Ahlswede and Z. Zhang, "Coding for write-efficient memory," *Inf. Comput.*, vol. 83, no. 1, pp. 80–97, 1989.

[95] L. A. Lastras-Montano, M. Franceschini, T. Mittelholzer, J. Karidis, and M. Wegman, "On the lifetime of multilevel memories," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Seoul, Korea, jun 2009, pp. 1224–1228.

[96] Q. Li and A. A. Jiang, "Polar codes are optimal for write-efficient memories," in *Proc. 51st Annu. Allerton Conf. Commun., Control, Comput.*, Monticello, IL, USA, oct 2013, pp. 660–667.

[97] L. Pamies-Juarez, H. D. L. Hollmann, and F. Oggier, "Locally repairable codes with multiple repair alternatives," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, jul 2013, pp. 892–896.

[98] A. S. Rawat, D. Papailiopoulos, A. Dimakis, and S. Vishwanath, "Locality and availability in distributed storage," *IEEE Trans. Inf. Theory*, 2016, accepted.