UNIVERSITY OF CALIFORNIA SAN DIEGO

Optimization of Read Thresholds in NAND Flash Memory for LDPC Codes

A thesis submitted in partial satisfaction of the
requirements for the degree
Master of Science

in

Electrical and Computer Engineering
(Communication Theory and Systems)

by

Yi-Shen Yeh

Committee in charge:

Professor Paul H. Siegel, Chair
Professor Young-Han Kim
Professor Alexander Vardy

2020

The thesis of Yi-Shen Yeh is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

Chair

University of California San Diego

2020

DEDICATION

To my beloved parents

EPIGRAPH

For the ideal that I hold dear to my heart,

I'd not regret a thousand time to die.

— Qu Yuan, "Li Sao"

Translated to English by Lu Zhang

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

To the building of this thesis, I have many people I'd like to express my gratitude towards. Without their help, it wouldn't have been possible for me to complete this work.

First,I would like to express my sincerest gratitude towards my advisor Prof. Paul H. Siegel for his encouragement, patience, guidance and kindness. Throughout the time that I've been working with him, apart from his professional expertise and invaluable insights unequal to any, he showed great tenderness and care from all aspects and details that not only made me felt comfortable expressing new and sometimes errouneous ideas, but also more confident in my own work. He was also very understanding when it comes to conflicts in schedules, and are able to change the meeting times to adapt to my trip in another time zone. I am grateful for all his help, advices and thoughtfulness. I am surely very fortunate to have Prof. Siegel as my advisor.

Secondly, I would like to thank Arman Fazeli, a postdoc at UCSD and a brilliant mentor. When I first encountered the field of coding theory, I was very troubled and frustrated by the subject. This changed completely after I met him, first in a class where he had to substitute for, and another where he was the main instructor. Patience is his greatest personality and is the reason why I go to him whenever I am in doubt. Not only was he the one that guided my into the field of LDPC codes and eventually recommended me to my advisor Prof. Siegel, but also whenever I had an upcoming job interview or a non-coding related problem, he never hesitated to lend my a helping hand. Furthermore, in our time working together, he always have this super power of explaining a concept, which was foreign to me, in a simplified way so that I'd understand. Repeatedly, he'd given me numerous insightful and monumental suggestions that boosted the content of our work. Repeatedly, he's shown great consideration in the sense of surfing through my English writings, teaching my Latex and have even been there with me debugging my program when I needed a fresh pair of eyes. There is not a chance I would have completed this thesis without his help and I consider myself more than lucky to have worked along side him.

Throughout the journey of my life, there are two people whom I am forever indebted to, my

beloved parents. My mom and dad have always been my number one advocate, unconditionally. Needless to say, their support was paramount and unequivocal. They backed my tuition knowing that they have to live on the breadline and this means the world to me. Calling them at the end of each week recharges my mind and gives me the strength to pursue my dreams further. This thesis is entirely dedicated to them.

Next, I would like to thank Prof. Alexander Vardy and Prof. Young-Han Kim to agreeing to be a part of my thesis committee, for the time and effort and the insightful comments that helped improve this work significantly.

I would like to thank Che-Yu Huang and Yu-Hsin Lin for the informative discussions and debates with them. They're with me throughout the process of understanding new concepts and dissecting complicated mathematical equations. Having their input stimulated my thought process, provided my with a new perspective and helped me a great deal in pushing my work forward. I would also like to extend my gratitude towards my roommate Henry Chang. With his company, I have someone to share my daily life and the progress of thesis with, and needn't fight through the COVID-19 pandemic alone.

Finally, I would like to thank Chao-Hsuan Chen for her continuous devotion to our relationship. Not only is her loyal and tireless support invaluable, but she has also been a constant motivation and a true inspiration for me. She is the best friend one can have and an excellent partner in crime. I am a better person because of her.

Chapter 3, in part, contains materials from [12]. Y. Yeh, A. Fazeli, and P.H. Siegel, "Optimization of Read Thresholds in MLC NAND Memory for LDPC Codes," in *Proc. 11th Annu. Non-Volatile Memories Workshop (NVMW)*, La Jolla, CA, USA, Mar. 2020. [Online]. Available: http://nvmw.ucsd.edu/program/. The thesis author was the primary investigator and author of this paper.

Chapter 5, in part, contains materials from [12]. Y. Yeh, A. Fazeli, and P.H. Siegel, "Optimization of Read Thresholds in MLC NAND Memory for LDPC Codes," in *Proc. 11th Annu. Non-Volatile Memories Workshop (NVMW)*, La Jolla, CA, USA, Mar. 2020. [Online]. Available: http://nvmw.ucsd.edu/program/. The thesis author was the primary investigator and author of this paper.

VITA

2018	B. S. in Communication Engineering, National Taipei University, Taipei
2020	M. S. in Electrical and Computer Engineering, University of California San Diego

PUBLICATIONS

Y. Yeh, A. Fazeli, and P.H. Siegel, "Optimization of Read Thresholds in MLC NAND Memory for LDPC Codes," in *Proc. 11th Annu. Non-Volatile Memories Workshop (NVMW)*, La Jolla, CA, USA, Mar. 2020. [Online]. Available: http://nvmw.ucsd.edu/program/

FIELDS OF STUDY

Major Field: Electrical Engineering

Studies in Communication Theory and Systems

Advisor: Paul H. Siegel

ABSTRACT OF THE THESIS

Optimization of Read Thresholds in NAND Flash Memory for LDPC Codes

by

Yi-Shen Yeh

Master of Science in Electrical and Computer Engineering
(Communication Theory and Systems)

University of California San Diego, 2020

Professor Paul H. Siegel, Chair

As conventional hard decision error correction codes (ECCs), such as BCH codes, become inadequate as the capacity of flash memory increases, soft decision ECCs such as LDPC codes have become a desirable option. However, fully utilizing the potential of soft decision based codes demands higher precision memory sensing with the tradeoff of memory read latency. To that extent, this thesis explores and compares two approaches to optimizing the positioning as well as the number of read (word-line) voltages for a specified program/erase (PE) cycle.

The first approach is optimization subject to maximizing the mutual information (MMI) of the equivalent discrete memoryless channel using gradient descent. We show that we can

get a near optimal performance by only using ~ 20 reads since additional reads beyond this point provides less than 0.05% of additional mutual information and less than 5% of additional performance gain.

The second approach is the optimization of read thresholds taking the code structure of LDPC codes into account. We use discretized density evolution as a proxy for bit error rate, and to serve as our performance measure in the gradient descent search. This optimization is code-dependent and gives lower BER; however, it is subject to be trapped in secondary local minimum. As a result, we propose a two-step optimization: first utilize the MMI approach for coarse read threshold optimization, then follow by the DE-based method for fine optimization.

Chapter 1

Introduction

Flash memories are used widely in multi-media and consumer device storage systems for their capability of storing large quantities of data without having to maintain power (non-volatility). They use the charge (or voltage levels) stored in floating-gate cells to represent data symbols. The naming of flash memories corresponds to the number of bits per symbols represented by this charge, e.g., single-level-cell (SLC), multi-level-cell (MLC) and triple-level-cell (TLC) corresponds to two levels, four levels and eight levels. However, as technologies continue to scale down, NAND flash memory continues to require stronger error correction codes (ECCs) to maintain data storage integrity [1]. For the longest time, BCH codes with hard-decision decoding (HDD) have been the dominant choice as ECCs in flash memory. As industry has continued to push the technology-scaling envelope, soft-decision decoders (such as the Chase BCH decoders [13] [14]) have been developed for BCH codes to prolong their viability and to forestall the introduction of more complex ECCs requiring iterative soft-decision decoding. However in recent years, BCH codes, even with soft-decision decoding, have been found to be insufficient and an alternative choice has been required.

Low-density parity-check (LDPC) codes are well-known for their capacity-achieving abilities for AWGN channels [15] and have become the main choice of ECCs in flash memories

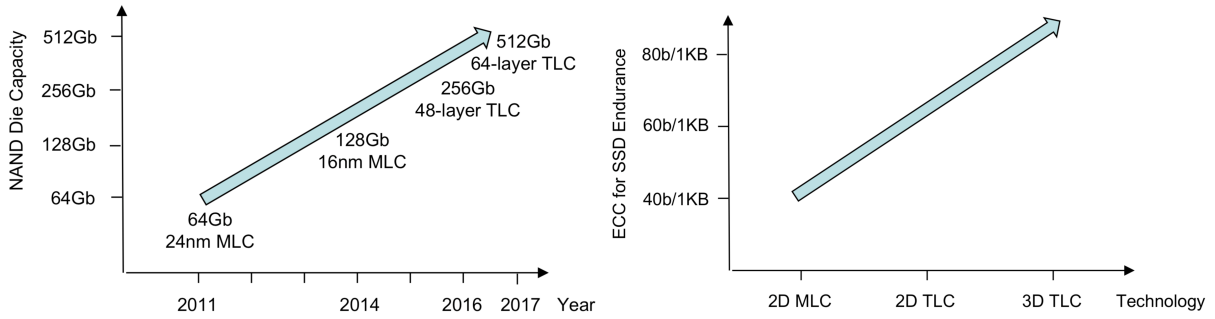


Figure 1.1: NAND flash memory scaling trends [1].

nowadays. The decoding gains from employing soft-decision LDPC codes compared with traditional HDD BCH codes are shown in [6]. However, to fully exploit the advantage of soft-decision decoding demands higher precision of memory sensing in the floating-gate transistors that feeds the soft information to the ECC decoders with the trade-off of memory read latency. Approaches such as rank modulation that eliminates the need for discrete cell levels have been discussed [16] [17], however in this thesis, we focus on the architecture of applying a constant voltage throughout a word-line. Therefore, the grey area between hard-decision reads and fully soft-decision reads is worth exploring, including both the placement and the number of these reads. In [18], the optimization of read threshold positions that maximizes the mutual information (MMI) of the NAND flash discrete equivalent channel is proposed. Another approach, in [19], uses density evolution (DE) as a tool to analyze the error probability of the channel of flash memory channel, which serves as the cost function in the optimization problem.

This thesis compares the two approaches (MMI and DE) towards optimizing the read thresholds for NAND flash memories. In Chapter 2, we present in detail the required background knowledge about NAND flash memory. We first discuss the structure and the read/write operations of this technology, followed by a discussion of noise sources and their mathematical models. In Chapter 3, we discuss the underlying equivalent discrete channel model for NAND flash memory. We reproduce the results in [18] and extend them to a higher number of reads. Also, we establish a binary labeling of cell-levels called the *symmetric channel labeling* (SCL) specifically

for use when looking into the MSB and LSB of MLC flash memory channels separately. We provide the corresponding threshold optimization results from this section in Appendix A and Appendix B. In Chapter 4, we present a review of required background knowledge on LDPC codes. We will explain the fundamental aspects of the code structure including the Tanner graph, cycles/girth, degree distributions, and code rate, as well as some of the common encoding and decoding methods. We will introduce two specific LDPC codes used in our analysis, a toy code which is a (3, 4)-regular QCLDPC code of girth 8 and rate 1/3 (which we will refer to as "Code A"), and an irregular LDPC code of rate 2/3 that is defined in 802.11n standards [11] and widely adopted for practical purposes (which we will refer to as "Code B"). Finally in Chapter 5, we will discuss the refinement of read thresholds using a density evolution approach, extending the ideas in [19]. We introduce a new quantization method (the *split ratio quantization* (SRQ)) on top of the standard quantizations to compensate for the fluctuations in the BER curves. We also discuss the discretized DE [20] method in detail. In the end, we provide our simulation results and discussion of our unique findings.

Throughout this thesis, we will assume no pre-existing knowledge of NAND flash memories and low-density parity check (LDPC) codes from the reader, but a basic understanding of linear algebra and communication theory is required. In terms of notation, we will use overbar ($\bar{v}$) or lowercase boldfaced letters ($\mathbf{v}$), hat ($\hat{w}$) and uppercase boldfaced letters ($\mathbf{H}$) to denote a row vector, a quantized version of the original message w and a matrix, respectively. $(\cdot)^T$ denotes the transpose operation. We will use $\circledast$ and $\otimes$ to denote the standard convolution and the discretized convolution, respectively. The $\mathbb{R}$, $\mathbb{Z}$, $\mathbb{Z}^+$ and $\mathbb{Z}^{++}$ symbols denote the set of all real values, the set of all integers, the set of all non-negative integers and the set of positive integers respectively. We use $\mathcal{N}(\mu, \sigma^2)$ to denote the Gaussian probability density function with mean μ and variance σ^2. The $\mathbb{E}(\cdot)$ denotes the expected value operation.

Table 1.1: List of common acronyms used in this thesis.

AWGN	additive white Gaussian noise
BAWGNC	binary AWGN channel
BCH	Bose-Chaudhuri-Hocquenghem
BER	bit error rate
BP	belief propagation
BSMC	binary symmetric memoryless channel
CCI	cell-to-cell interference
CN	check node
CPM	circular permutation matrix
DE/DDE	density evolution/discretized DE
DMC	discrete memoryless channel
ECC	error correction code
FER	frame error rate
FFT	fast Fourier transform
GD	gradient descent
ISPP	incremental step pulse program
LDPC	low-density parity-check
LDPCL	LDPC locality
LLR	log-likelihood ratio
LSB	least significant bit
MI	mutual information
MLC	multi-level cell
MMI	maximum/maximize MI
MS	min-sum
MSB	most significant bit
NR	new radio
P/E	program/erase
PCM	parity-check matrix
PDF	probability density function
PMF	probability mass function
QC	quasi-cyclic
RTN	random-telegraph noise
SCL	symmetric channel labeling
SLC	single-level cell
SNR	signal-to-noise ratio
SRQ	split ratio quantization
SSD	solid-state drive
TLC	triple-level cell
VN	variable node

Chapter 2

NAND Flash Memory

Solid-state drives (SSDs) are widely used in computer systems nowadays as a primary method of data storage and NAND flash memory technology is the main contributor. Flash memory is a non-volatile solid-state storage medium that relies on electric circuits to store and retrieve data. It has no moving parts, unlike most magnetic storage devices, thus reducing mechanical errors, and is resilient against physical impacts. Moreover, it provides significantly higher read and write performance while requiring less power and much smaller latencies. For the past two decades, the cost per bit for flash memories has maintained a steady reduction allowing a widespread adoption and adhering to the increasing demand from data centers, enterprise and personal computing, and mobile consumer devices.

The term *non-volatile* refers to the capability of maintaining data without power. This is due to the use of a floating-gate transistor as the basic memory element, referred to as a cell (see Figure 2.1). A NAND flash memory uses different voltage levels in cells to represent different information symbols stored. The number of bits in those cells also determine the naming of the flash. For example, 1, 2 and 3 bits per symbols are called single-level cell (SLC), multi-level cell (MLC) and triple-level cell (TLC), respectively. In this thesis, we will focus mainly on MLC NAND flash memory setup (i.e. two bits per symbol). However, our methods can be easily

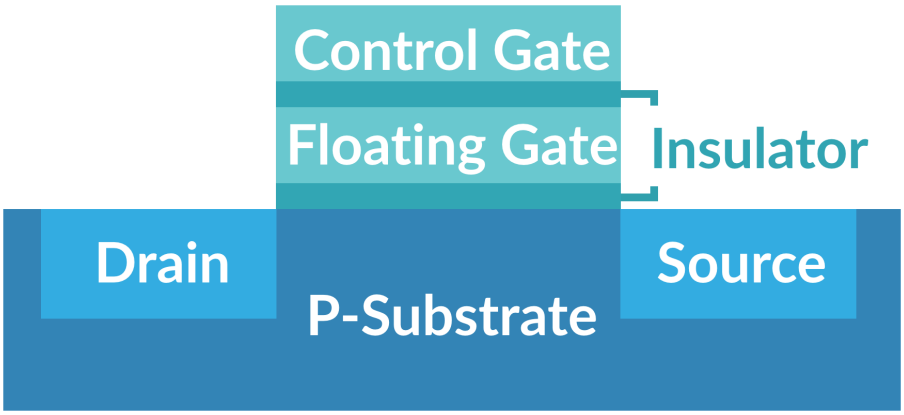


Figure 2.1: A floating gate transistor.

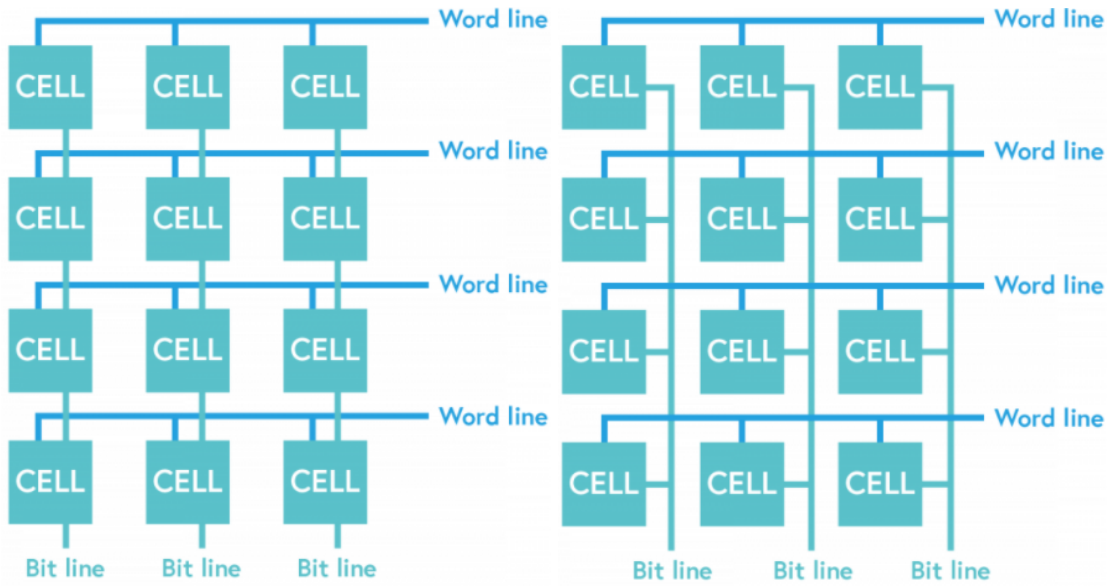
integrated into other types as well.

2.1 Structure

A *NAND* flash memory is named so because the cells are connected in a way that resembles a NAND gate. Its counterpart is the NOR Flash. Both configurations organize cells into rectangular arrays called a *block*, with *word-lines* connecting the cells row-wise plugging into the transistor's control gate and *bit-lines* connecting the cells column-wise linking each cell's source and drain. The main difference is that for NOR flash, the bit-lines run along side the cells whereas in NAND flashes the bit-lines run *through* the cells, bypassing the need for an extra bit-line connector (see Figure 2.2) [2]. Due to their wiring configurations, NOR offers better read speeds and random access capabilities; NAND offers better write/erase capabilities with a cheaper cost per bit, so it is better suited for the data demands of SSDs, Flash Drives, and Flash Memory Cards.

In a broader view, the hierarchy of flash memory is:

$$\text{Chip} \rightarrow \text{Die} \rightarrow \text{Plane} \rightarrow \text{Block} \rightarrow \text{Page} \rightarrow \text{Cell}$$



(a) NAND (b) NOR

Figure 2.2: Configuration of NAND and NOR flash memories [2].

as shown in Figure 2.3. A chip can have as many as 16 dies, each of which in turn contains between one and four planes. Each plane contains hundreds to thousands of flash blocks, and each block is a 2-D array that contains hundreds of rows of flash cells [3]. In a MLC flash memory, the two bits in the cells connected along a word-line are assigned to two separate *pages*. In practice, page size ranges from 512-byte to 8K-byte data. The most significant bit (MSB) is assigned to the lower page while the least significant bit (LSB) is assigned to the upper page. Also, in our simulations, we let voltage levels in each cell that correspond to the 2-bit symbols, denoted $\{'11','10','00','01'\}$ or $\{1,2,3,4\}$, be:

$$\{\mu_{11},\mu_{10},\mu_{00},\mu_{01}\} \equiv \{\mu_1,\mu_2,\mu_3,\mu_4\} = \{-3,-1,1,3\}(\text{V}) \tag{2.1}$$

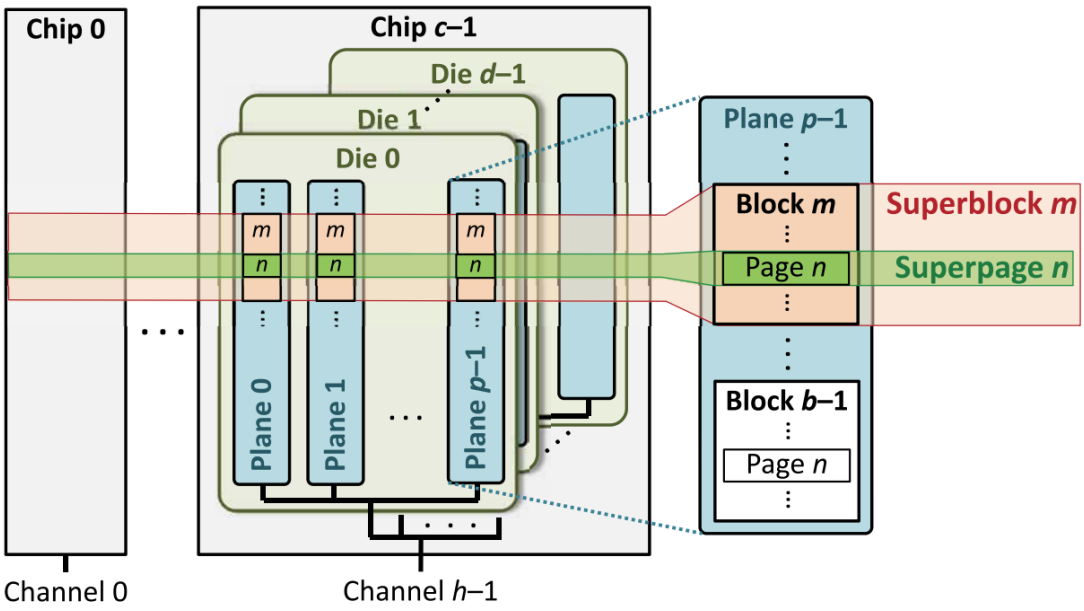


Figure 2.3: Flash memory organization [3].

2.2 NAND Flash Operations

2.2.1 Program and Erase Operations

The measurement of a solid state drive's lifespan is called *program/erase cycles* (PE cycles), where one write and erase operation counts as one cycle, because flash chips suffer a small amount of wear each time they are written and erased. All read and write operations are performed at the granularity of a page whereas the erase operation is done on a block level. However, before we could write to a cell, it must first be erased which is done by applying high voltage to the substrate to remove electrons from the floating gate. The write operation, known as programming, is done by applying high voltage at the control gate to trap electrons inside the floating gate through Fowler-Nordheim tunneling [21] and is conducted in multiple steps. In the case of MLC, it is conducted in two steps. The lower page bit of a cell is first programmed followed by programming the corresponding upper page bit as depicted in Figure 2.4.

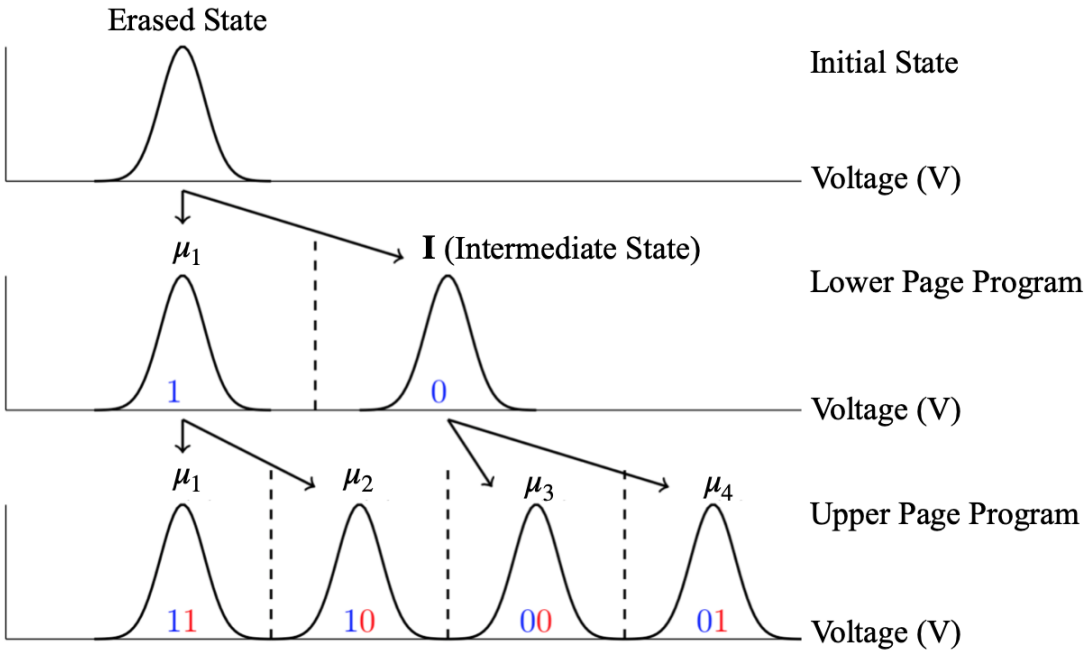


Figure 2.4: Two-step programming of lower page and upper page in MLC flash memory.

2.2.2 Read Operation

The read operation is done by applying a word-line reference voltage (we call it the read voltage, *read threshold* or a *read* denoted as $\gamma(V)$) at the control gate. A sense amp comparator compares the drain current to a threshold. If the drain current is above this threshold, then the reference voltage was sufficient to turn on the transistor indicating that the charge written to the floating gate is below a certain value. Conversely, if a charge in the floating gate has accumulated above a certain level, the resistance between source and drain is high, so that the current flow between source and drain is below the threshold [22]. This behavior resembles a binary output where we will know either the stored voltage level is higher or lower than the reference voltage $\gamma(V)$ or *read* we applied. Consequently, as we increase the number of reads, we will get softer information at the cost of read latency. Note that the stored voltage is often referred to as the *threshold voltage*. However, to avoid confusion with the read thresholds, for this thesis we will call it the *stored voltage*.

2.3 Noise Models

Throughout the life cycle of a NAND flash, no matter whether a cell is used repeatedly or scarcely, it will always experience a variety of distortions, or noise as we call it, such as programming noise, cell-to-cell interference (CCI), random-telegraph noise (RTN), and charge leakage. Here we will briefly discuss some of the NAND flash channel models that reflect the device operations as well as the influence of various factors such as program/erase (PE) cycling and retention period [23].

2.3.1 Erase Operation Model

As we've previously mentioned, a flash memory cell must be erased before it can be programmed. The threshold voltage distribution for erased cells can be modeled with Gaussian distribution given by [4] [24]:

$$f_e(x) = f_{11}(x) = \frac{1}{\sigma_e\sqrt{2\pi}} e^{-\frac{(x-\mu_e)^2}{2\sigma_e^2}}$$

where $\mu_e = \mu_1$ and σ_e are the mean and standard deviation of the erased state.

2.3.2 Program Operation Model

When programming the voltages into the floating gates, a tight threshold voltage control is typically realized by using *incremental step pulse program* (ISPP) [25], i.e., memory cells on the same word-line are recursively programmed using a program-and-verify approach. Denote V_p and V_{pp} as the target programmed state and program step voltage respectively. The stored voltage

of the programmed state tends to have a uniform distribution over $[V_p, V_p + \Delta V_{pp}]$ [4]:

$$f_p = \begin{cases} \frac{1}{\Delta V_{pp}}, & \text{for } V_p \leq x \leq V_p + \Delta V_{pp} \\ 0 & \text{otherwise.} \end{cases}$$

Meanwhile, programmed cells are also affected by programming noise f_{pn} , which can be modeled with Gaussian distribution [26] with zero mean and standard deviation σ_{pn}. Thus the overall distribution of the programmed cells can be given by:

$$f_{\text{prog}} = f_p \circledast f_{pn}$$

where $f_{\text{prog}} \in \{f_{10}, f_{00}, f_{11}\}$ and $V_p \in \{\mu_2, \mu_3, \mu_4\}$.

2.3.3 Cell-to-Cell Interference (CCI)

The cell-to-cell interference (CCI) is a result of parasitic capacitive-coupling between neighboring cells. The threshold voltage shift due to CCI can be given as [4]:

$$V_{CCI} = \sum_k \Delta V_k \eta_k$$

where ΔV_k is change in threshold voltage of the interfering (neighboring) cell and η_k is the capacitive coupling ratio defined as:

$$\eta_k = \frac{C_k}{C_{total}}$$

where C_k is the parasitic capacitance between the interfering cell and the victim cell, and C_{total} is the total capacitance of the victim cell. In an *even-odd bit-line* architecture, where even bits are written before odd bits in a single word-line, the number of interfering cells are five and three for even and odd bits, respectively, as shown in Figure 2.5.

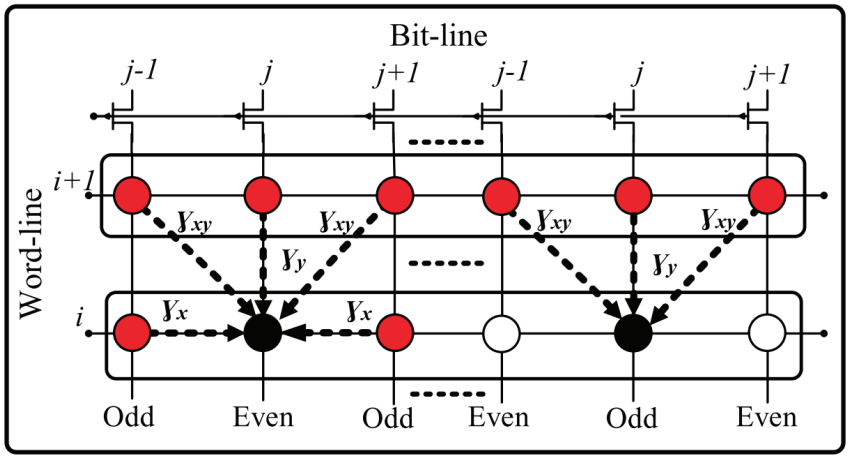


Figure 2.5: Illustration of cell-to-cell interference in an even-odd bit-line architecture [4].

In an *all bit-line* architecture, on the other hand, where all bits in a word-line are written simultaneously, all cells have three interfering cells (i.e., in the next word-line). [4] claims that CCI can be averted using precoding with the exception of the erased state and therefore models μ_1 using Gaussian distribution function with the shifted mean $\tilde{\mu}_1$ given by:

$$\tilde{\mu}_1^{even} = \mu_1 + \Delta\mu_{avg}(2\gamma_x + 2\gamma_{xy} + \gamma_y)$$

$$\tilde{\mu}_1^{odd} = \tilde{\mu}_1^{all} = \mu_1 + \Delta\mu_{avg}(2\gamma_{xy} + \gamma_y)$$

where $\Delta\mu = (\mu_1 + \mu_4)/2 - \mu_1$.

2.3.4 Random Telegraph Noise (RTN)

One effect of PE cycles is the *random telegraph noise* that occurs in semiconductors and ultra-thin gate oxide films. More specifically, it is when electron capture and emission events at charge trap sites near the interface over the course of PE cycling directly result in memory cell threshold voltage fluctuation [23]. In [25], the probability density function (pdf) of RTN-induced

voltage fluctuation is modeled as a symmetric exponential function:

$$f_{\text{RTN}}(x) = \frac{1}{2\lambda_n} e^{-\frac{|x|}{\lambda_n}}$$

where λ_n is the decay constant. However, [4] approximates RTN distribution by a zero mean Gaussian distribution given by:

$$f_{\text{RTN}}(x) = \frac{1}{\sigma_n \sqrt{2\pi}} e^{-\frac{x^2}{2\sigma_n^2}}$$

where the RTN variance σ_n^2 is a non-stationary parameter which varies with respect to PE cycles. Note that the effect of RTN on threshold voltage signal is less significant compared to other noise components.

2.3.5 Data Retention Noise

Another distortion effects of PE cycles is when interface *trap recovery* and electron *detrapping* gradually reduce memory cell threshold voltage, leading to the data retention limitation. It is also modeled as a Gaussian distribution $\mathcal{N}(\mu_{dr}, \sigma_{dr}^2)$ [4]:

$$\begin{cases} \mu_{dr} = (V_s - x_0) \cdot \left[A_t \cdot N^{\alpha_i} + B_t \cdot N^{\alpha_0}\right] \cdot \log(1+T) \\ \sigma_{dr} = 0.4|\mu_{dr}| \end{cases}$$

where $V_s \in \{\tilde{\mu}_1, \mu_2, \mu_3, \mu_4\}$, T is the data retention time measured in years, N is the number of P/E cycles and A_t, B_t, α_i, α_0, x_0 are other constants.

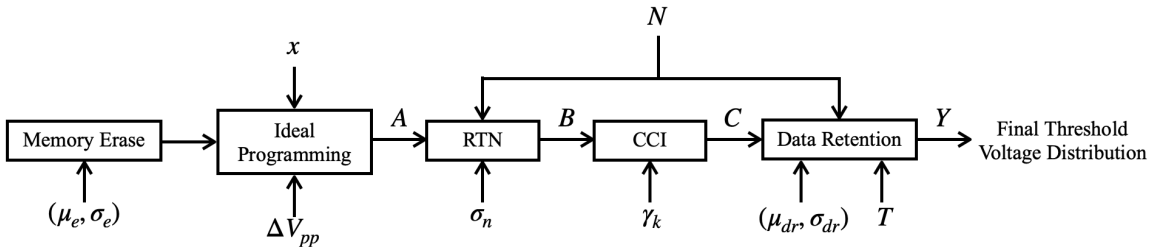


Figure 2.6: Illustration of the approximate NAND flash memory device channel model consisting of major distortion sources.

2.3.6 NAND Flash Channel Model

Overall, we an approximately model NAND flash memory device characteristics, as shown in Figure 2.6, to simulate the memory cell stored voltage distribution. However, the focus of this thesis does not require us to model the flash memory channel with such detail. Therefore, for simplicity, we will model the overall voltage distribution as a mixture of equal-variance Gaussian distributions with fixed, equi-distant means (corresponding to the cell levels in equation 2.1).

2.4 Read Retry Algorithm

Reading a page in flash memory is typically done with a *read retry* algorithm as shown in Figure 2.7, where an additional read with a different reference voltage is applied if the initial page read was inconclusive or had a high error rate. Apart from the obvious read latency such retry operation introduces, sensed data should be temporarily stored in an on-chip page buffer, so additional read will result in higher silicon cost. Therefore it is critical to optimize the placement of these reference voltages in order to minimize the sensing precision (i.e., number of memory sensing levels) required. This serves as the motivation of this thesis, as we will go into the optimization problem in the following chapters.

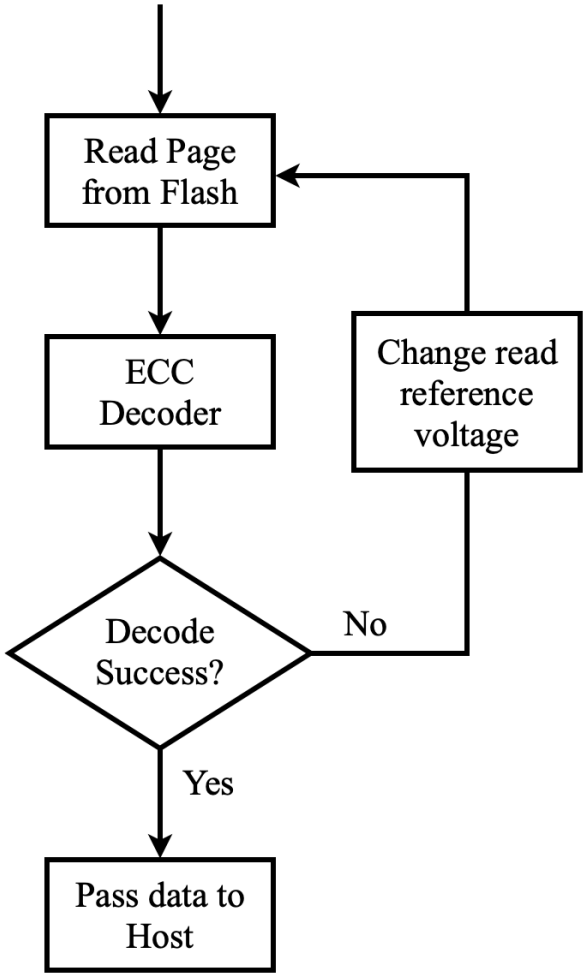


Figure 2.7: Read retry algorithm for NAND flash memory.

Chapter 3

Read Threshold and Channel Capacity

In Chapter 2 we discussed the fundamentals of NAND flash memories. We saw that the stored voltage level can be erroneous due to fluctuations that arise for a variety of reasons. However, it was shown in [18] that, because of the characteristics of the read operation discussed in Section 2.2.2, adjusting the read (word-line) voltages and, thereby, maximizing the capacity of the channel model can reduce the error probability. In this chapter, we first introduce the read voltage setup, revisit the read operations with mathematical descriptions, and reproduce the threshold optimization results given in [18]. In the last section, we extend the methods of [18] to a higher number of reads and state our conclusions about the incremental performance gains associated with the use of additional reads.

3.1 Channel Model

A channel model for a flash memory can be viewed as a simplified representation of the underlying physical mechanisms which induce errors in stored data. Although we've discussed some of the models in Section 2.3, for simplicity, we model the NAND flash channel fluctuations as an additive white Gaussian noise (AWGN).

We define the relationship between the signal-to-noise ratio (SNR) and the noise variance

of the channel $\sigma^2 = N_0$ (for simplicity in writing) with:

$$\text{SNR} = 10\log_{10}\frac{E_s}{N_0}(\text{dB}) \tag{3.1}$$

where $E_s = \left(\sum_{i=1}^{4}\mu_i^2\right)/4$ denotes the energy per symbol. If one wishes to use the traditional definition where $\sigma^2 = N_0/2$, then simply reduce the SNR value by 3dB (as shown in Figure 3.1). Let $\{f_{11}, f_{10}, f_{00}, f_{01}\}$ denote the probability density functions of Gaussian random variables centered at $\{\mu_1, \mu_2, \mu_3, \mu_4\}$ as shown in Figure 3.3, where μ_is, $i = 1, \ldots, 4$, are given by equation (2.1).

These functions represent the conditional densities of the read signal given the programmed cell level. The variance of each of these conditional densities is set equal to the noise variance N_0. In practice, flash memory channel quality is often measured in terms of the number of program/erase (P/E) cycles, rather than as an SNR value. Therefore, channel error-rates are usually plotted as a function of P/E cycle count. However, in this thesis, we will consider performance as a function of SNR. In order to convert between the two channel quality measures, one can use results in [5], where a model is developed that accurately predicts changes in the threshold voltage distribution as the number of P/E cycles increases. The model can be used to establish a one-to-one correspondence between SNR and an equivalent number of P/E cycles, as illustrated in Figure 3.2.

3.1.1 Read Threshold and Gaps

Assume that there are m read thresholds $\bar{\gamma} \triangleq \{\gamma_1, \ldots, \gamma_m\}$ that divide the voltages into $m+1$ regions $\{R_1, \ldots, R_{m+1}\}$. Performing voltage reads at all these thresholds allows the decoder to narrow down the stored voltage to one of these regions. For each region, we can calculate the channel likelihood ratios as

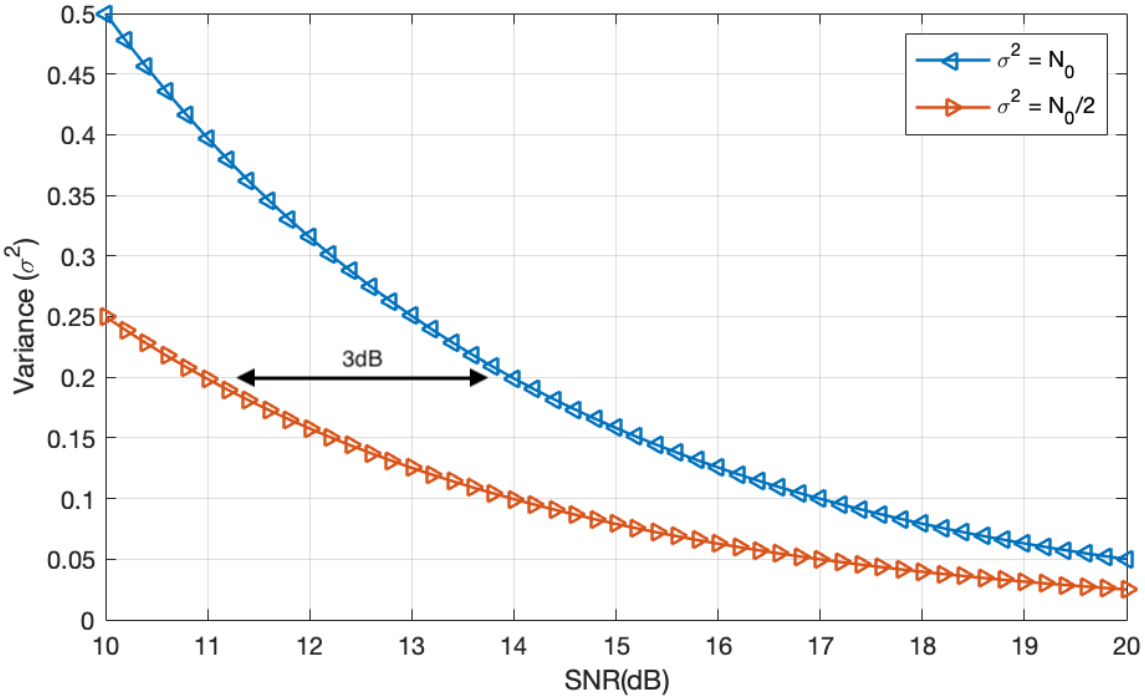


Figure 3.1: The 3dB difference in the two definition of the SNR to noise variance conversion.

$$\begin{aligned} \ell_{\text{MSB}}^{\text{ch}} &= \ln\left(\frac{\int_{R_i}(f_{01}(\nu)+f_{00}(\nu))d\nu}{\int_{R_i}(f_{10}(\nu)+f_{11}(\nu))d\nu}\right) \\ \ell_{\text{LSB}}^{\text{ch}} &= \ln\left(\frac{\int_{R_i}(f_{10}(\nu)+f_{00}(\nu))d\nu}{\int_{R_i}(f_{11}(\nu)+f_{01}(\nu))d\nu}\right), \end{aligned} \tag{3.2}$$

which are then fed to the decoder. For the special case of $m = 6$, we define the notion of *gaps*. Suppose we set the thresholds initially at positions $[\gamma_1, \gamma_2, ..., \gamma_6] = [-2, -2, 0, 0, 2, 2]$. The gaps are defined as the shifts in the read voltage thresholds away from these initial positions. A positive gap corresponds to a decrease in voltages for odd thresholds and an increase for even thresholds (see Figure 3.3). For example, if all gaps equal 0.4(V), the read thresholds will be $[\gamma_1, \gamma_2, ..., \gamma_6] = [-2.4, -1.6, -0.4, 0.4, 1.6, 2.4]$.

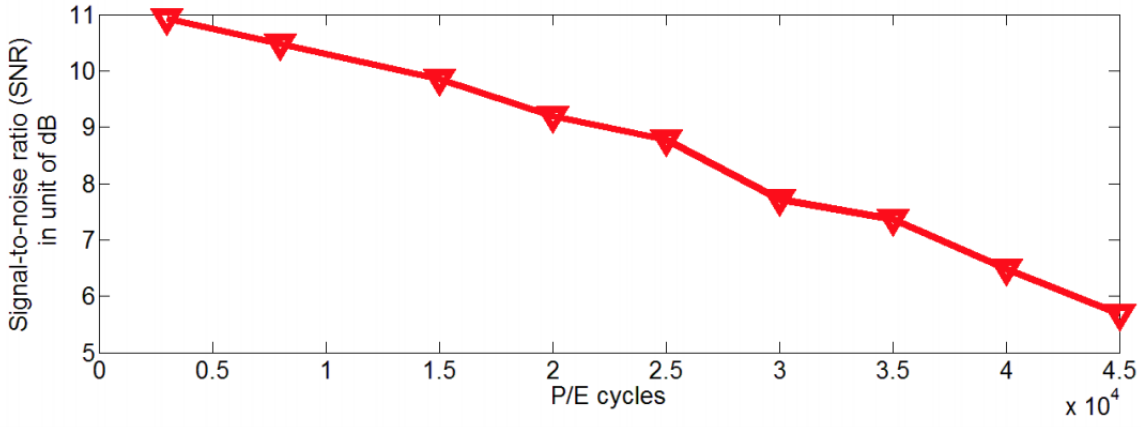


Figure 3.2: Signal to noise ratio (SNR) vs. P/E cycles [5].

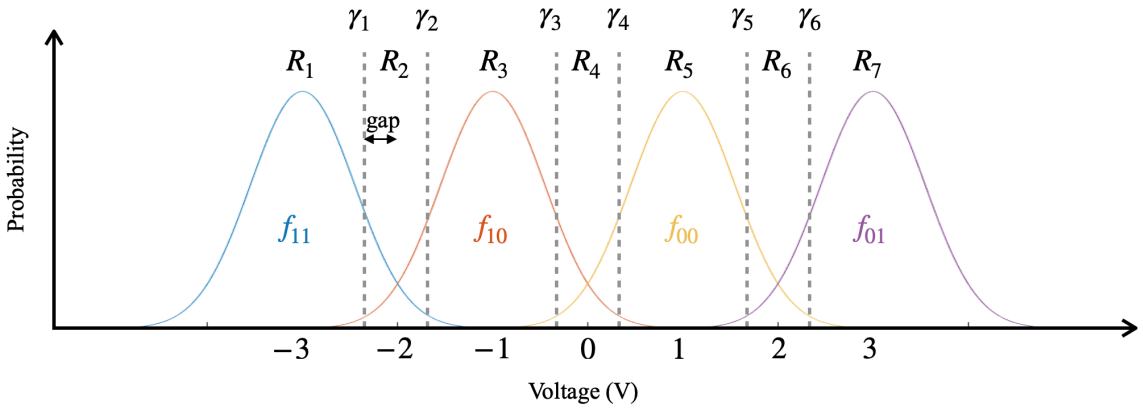


Figure 3.3: Read voltage thresholds and voltage distributions for an MLC NAND flash memory with $m = 6$ read thresholds.

3.1.2 Equivalent Discrete Memoryless Channel

For MLC with $m = 6$ read thresholds, the voltage range is divided into 7 regions as depicted in Figure 3.3. This quantization effectively represents a discrete memoryless channel (DMC) shown in Figure 3.4, where input and output are random variables denoted as $X \in \{\mu_1, \mu_2, \mu_3, \mu_4\}$ and $Y \in \{R_1, \ldots, R_7\}$ respectively, and probabilities $p_{ij} = \Pr\{Y \in R_j | X = \mu_i\}$ denote the crossover probabilities for $i = \{1,2,3,4\}$ and $j = \{1,2,\cdots,7\}$.

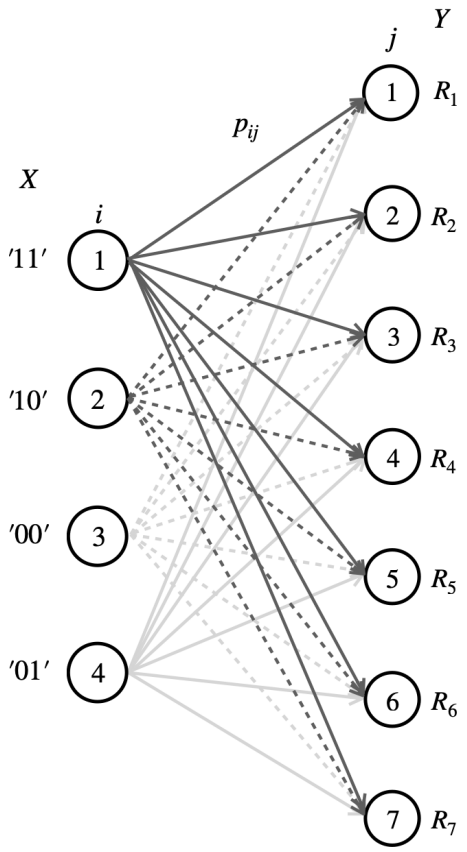


Figure 3.4: Equivalent discrete memoryless channel (DMC) model for MLC with $m = 6$ read thresholds.

3.2 Maximizing Mutual Information

Mutual Information is a measure of the mutual dependence between the two random variables. It quantifies the *amount of information* obtained about one random variable through observing the other. Therefore it is sometimes considered an indicator for the capacity of memory storage. Thus our objective, naturally, is to maximize the mutual information between our input and output random variables X and Y. Assuming X in our MLC discrete memoryless channel (Section 3.1.2) is a uniform random variable, that is, X is equally likely to be any of the input symbols, the mutual information can be given by [18], [27]:

$$
\begin{aligned}
I(X;Y) &= H(Y) - H(Y|X) \\
&= H\left(\frac{p_{11}+p_{21}+p_{31}+p_{41}}{4}, \frac{p_{12}+p_{22}+p_{32}+p_{42}}{4}, \cdots, \frac{p_{17}+p_{27}+p_{37}+p_{47}}{4}\right) \\
&- \frac{1}{4}\Big[H(p_{11},p_{12},\cdots,p_{17}) + H(p_{21},p_{22},\cdots,p_{27}) + \cdots + H(p_{71},p_{72},\cdots,p_{77})\Big]
\end{aligned} \tag{3.3}
$$

where $H(\cdot)$ denotes the entropy function [27].

Constraining ourselves to the MLC scenario, given a specified SNR value and a fixed number of read thresholds m, we use gradient descent (technically *ascent* since we are *maximizing* the mutual information) algorithm to adjust the read thresholds to the position where it would give us the maximum mutual information. With the cost function $g(\bar{\mu}, N_0, \bar{\gamma}) = I(X;Y)$, which we write as $g(\bar{\gamma})$ when the input parameters $\bar{\mu}$ and N_0 are fixed, the gradient descend algorithm gives us:

$$
\bar{\gamma}^{(l+1)} = \bar{\gamma}^{(l)} + \delta \cdot \nabla g(\bar{\gamma}^{(l)}) \tag{3.4}
$$

where δ is the step size (usually denoted by μ in the literature, but renamed here to avoid confusion with our voltage means μ_i), $\bar{\gamma}^{(l)}$ denotes the vector of read thresholds at iteration l and $\nabla(\cdot)$ denotes the gradient operation. In program simulations, the gradient is obtained by choosing two points at distance ε apart along the read voltage dimension and taking the slope, where $\varepsilon \to 0$. We continue with this equation to update the position of read thresholds until the increase in cost function $g(\bar{\gamma})$ per iteration no longer exceeds a certain amount. We call this incremental gain the termination criterion, denoted as ζ.

3.3 Simulation and Results

In this section, we discuss our simulation results done with MATLAB. We set number of reads $m = 6$, step size $\delta = 0.5$, gradient distance $\varepsilon = 1 \times 10^{-3}$ and $\zeta = 1 \times 10^{-8}$. The algorithm

is given in Algorithm 1. As shown in Figure 3.5, we can graphically visualize the evolution of mutual information achieved during each gradient descent iteration as well as how the read threshold positions are adjusted until the mutual information gain per iteration $\partial g < \zeta$. We see that the algorithm quickly converges even with randomly selected initial threshold positions (no symmetry constraint). One observation we make here is that the final positions seem to have equal *gaps*. This observation holds for all $m = 6$ cases with accuracy within 10^{-3} volts. Thus we constrain the read thresholds to have equal gaps and conduct an exhaustive search across gaps for different SNR values. The results are shown in Figure 3.6. We cross reference our results with that of Figure 3.5 and see that the results are consistent. Both figures show an optimal gap of 0.3528 (V) and attain a maximum mutual information of 1.5147.

3.3.1 Higher Number of Reads

For the special case $m = 6$ reads that we have simulated, the results showing that optimal positions should be placed around the intersection points of the probability density functions are intuitive. However, this qualitative behavior is not observed when going beyond $m = 6$ reads.For example, with $m = 30$ reads we have the results shown in Figure 3.8. We see that the read thresholds are more densely distributed the center and gradually spread out outwards. This makes intuitive sense since errors are prone to exist where probability density functions overlap the most, therefore requiring more precise soft information.

Finally, in Figure 3.9, we show the maximum mutual information achievable for a practical range of SNRs and number of reads. We conclude that after $m = 20$ reads, mutual information gain per additional read is below 0.05% and thus attains near optimal performance. All simulation data are provided in Appendix A.

Algorithm 1: Gradient Descent to Maximize Mutual Information

input : The vector of mean stored voltages $\bar{\mu}$ given by equation (2.1), the variance of AWGN N_0 and number of read thresholds m.

output : The optimal read thresholds positions $\bar{\gamma}$ that attains the maximum mutual information achievable.

```
/* Initialization                                                    */
```

Iteration number $l = 0$;

Initialize $\bar{\gamma}^{(0)}$ to any starting position ;

Initialize gain $\partial g = 10$ (a very large number) ;

$\mathrm{MI}^{(l)} = \mathrm{g}(\,\bar{\gamma}\,)$; `// g(·) is the function that calculates the MI with given parameters`

```
/* Main Loop                                                         */
```

while $\partial g > \zeta$ **do**

```
    /* ε̄_i denotes a zero vector of length m with only ε at the i^th
       entry                                                         */
```

  for $i = 1 \rightarrow m$ **do**

    $\nabla g(\bar{\gamma}^{(l)})_i = \frac{g(\bar{\gamma}^{(l)} + \bar{\varepsilon}_i)}{\varepsilon}$;

  end

  $\bar{\gamma}^{(l+1)} = \bar{\gamma}^{(l)} + \delta \cdot \nabla g(\bar{\gamma}^{(l)})$;

  $\mathrm{MI}^{(l+1)} = \mathrm{g}(\,\bar{\gamma}^{(l+1)}\,)$;

  $\partial g = \mathrm{MI}^{(l+1)} - \mathrm{MI}^{(l)}$;

  $l{+}{+}$;

end

Result: Optimal read threshold positions are $\bar{\gamma}^{(l)}$

3.3.2 Symmetric Channel Labeling

Provided with the equivalent DMC, we see that when we look at the MSB channel and the LSB channel individually, not only do they have different crossover probabilities, but also their symmetry properties differ. In particular, the MSB channel is symmetric, whereas the LSB channel is asymmetric. This will preclude the use of conventional density evolution on the LSB channel in the later sections. Thus, putting traditional Gray code labeling aside, we employ a second labeling which we will refer to as the *symmetric channel labeling* for MLC NAND flash memory (or MLC-SCL). This labeling uses a different mapping of distribution means to symbols

mapping :

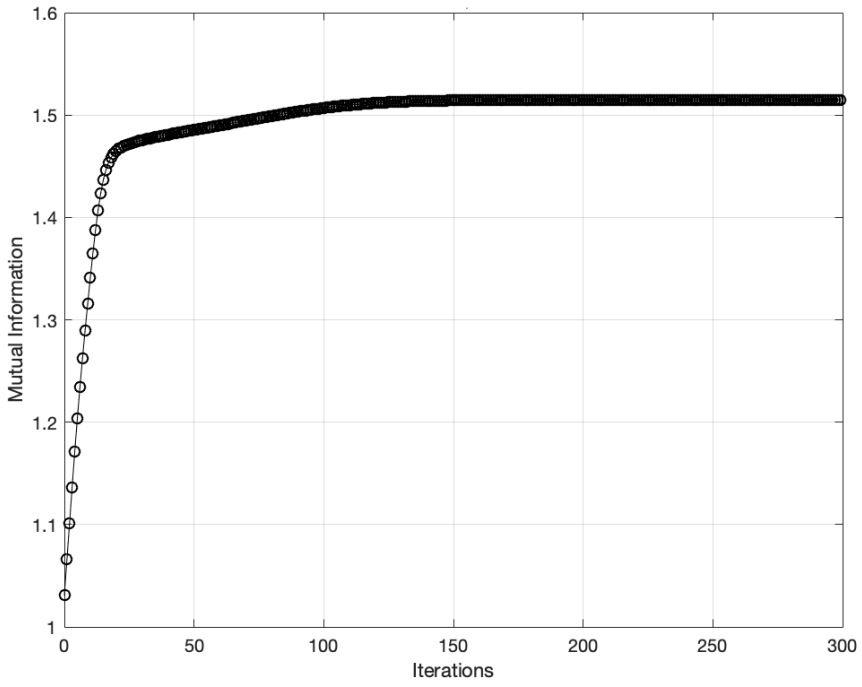
$$X \in \{\mu_1, \mu_2, \mu_3, \mu_4\} \equiv \{11, 10, 01, 00\} \tag{3.5}$$

that allows both MSB and LSB channels to be symmetric. It can be extended to TLC case where the labeling sequence becomes $\{'111', '110', '101', '100', '011', '010', '001', '000'\}$. This specific labeling method can also be found in [28] where they refer to as the *natural order*(NO) labeling. Figures 3.10 and 3.11 shows the MMI achieved for $m = [3, 30]$ reads and SNR $= [10, 15]$(dB) for LSB and MSB channels, respectively, using MLC-SCL. We can see that the LSB channel is clearly more degraded than its counterpart. The complete simulation data, i.e., the optimal read threshold positions as well as the maximum mutual information attained, are provided in Appendix B.

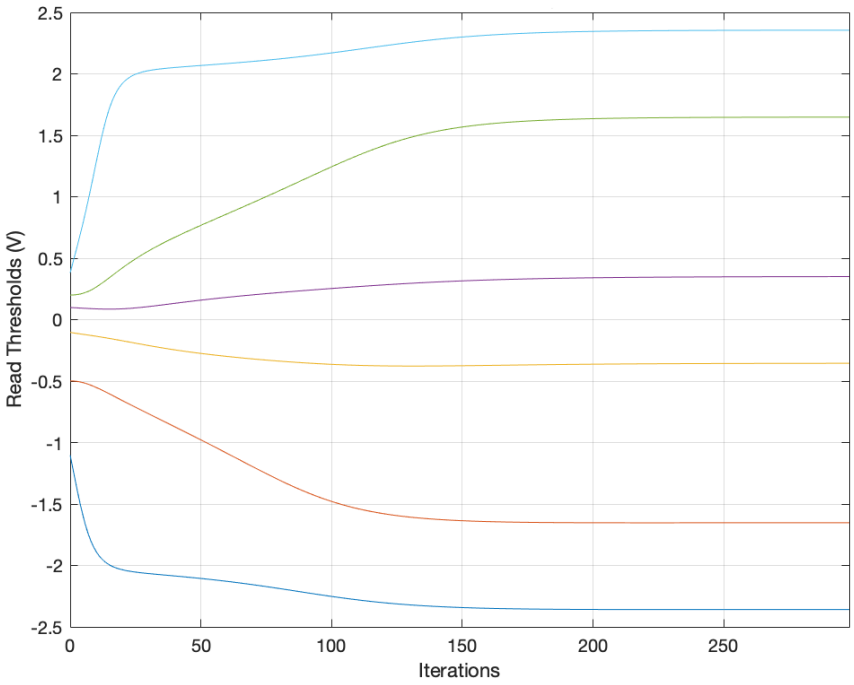
Chapter 3, in part, contains material from [12]

- Y. Yeh, A. Fazeli, and P.H. Siegel, "Optimization of Read Thresholds in MLC NAND Memory for LDPC Codes," in *Proc. 11th Annu. Non-Volatile Memories Workshop (NVMW)*, La Jolla, CA, USA, Mar. 2020. [Online]. Available: http://nvmw.ucsd.edu/program/

The thesis author was the primary investigator and author of this paper.



(a) Evolution of mutual information.



(b) Evolution of read threshold positions.

Figure 3.5: The mutual information achieved and evolution of position of read thresholds throughout gradient descent iteration in $m = 6$, $SNR = 10$ (dB) case. Final mutual information attained is 1.5147.

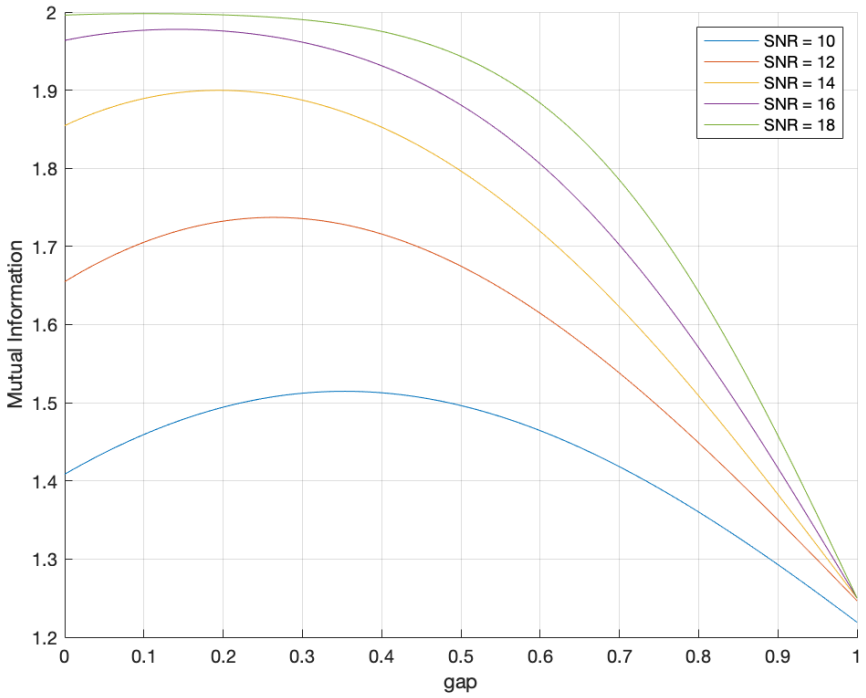


Figure 3.6: Exhaustive search across gaps for special $m = 6$ case and for SNR = 10 to 18 dB.

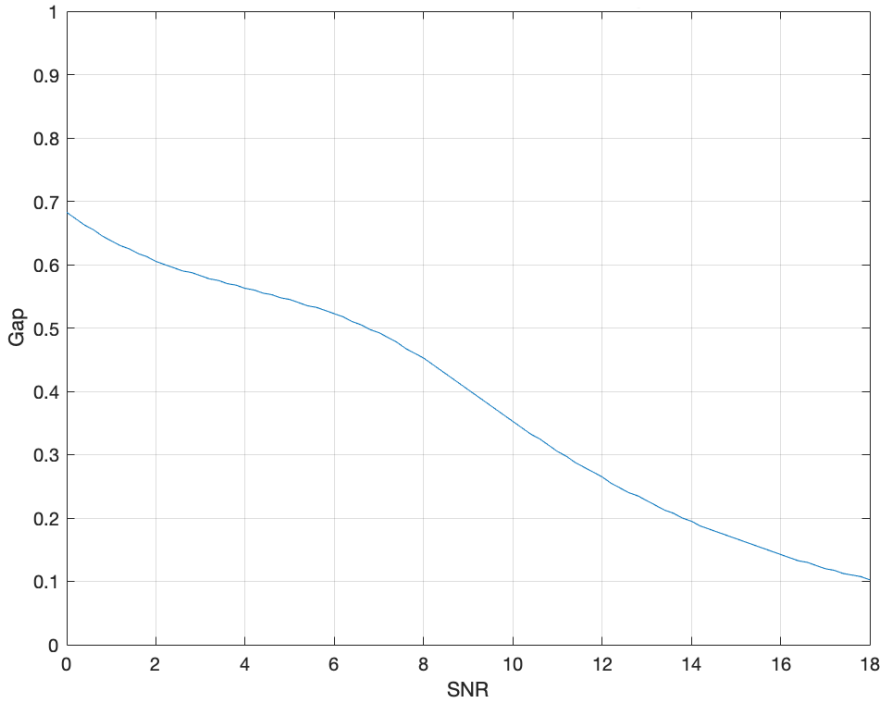


Figure 3.7: Optimal read threshold positions for $m = 6$ case with the constraint of equal gaps and for SNR = 0 to 18 dB.

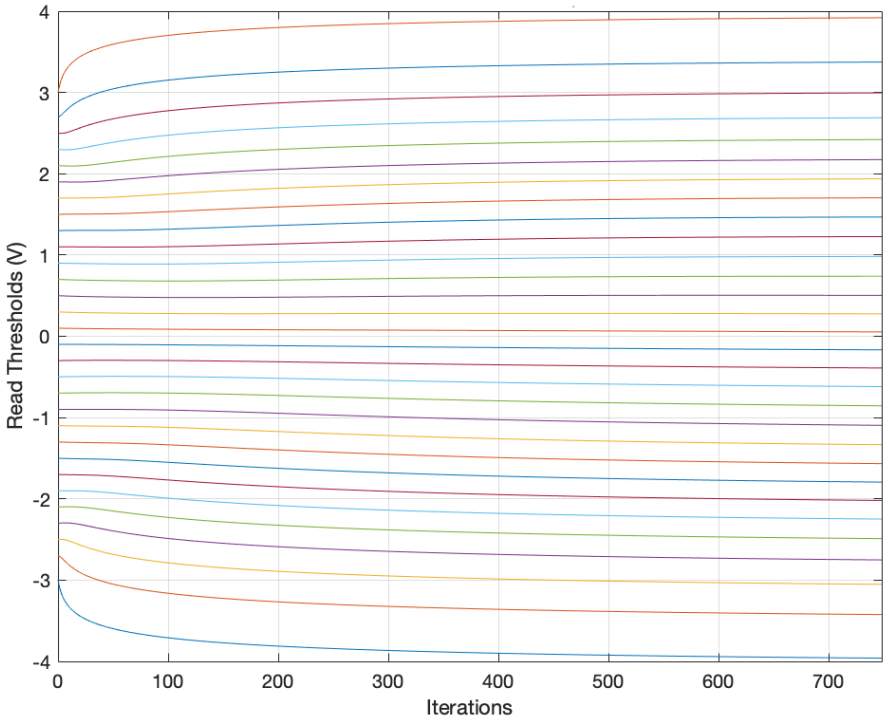


Figure 3.8: The evolution of positions of read thresholds throughout gradient descent iteration in $m = 30$, $SNR = 10$ (dB) case. Final mutual information attained is 1.5781.

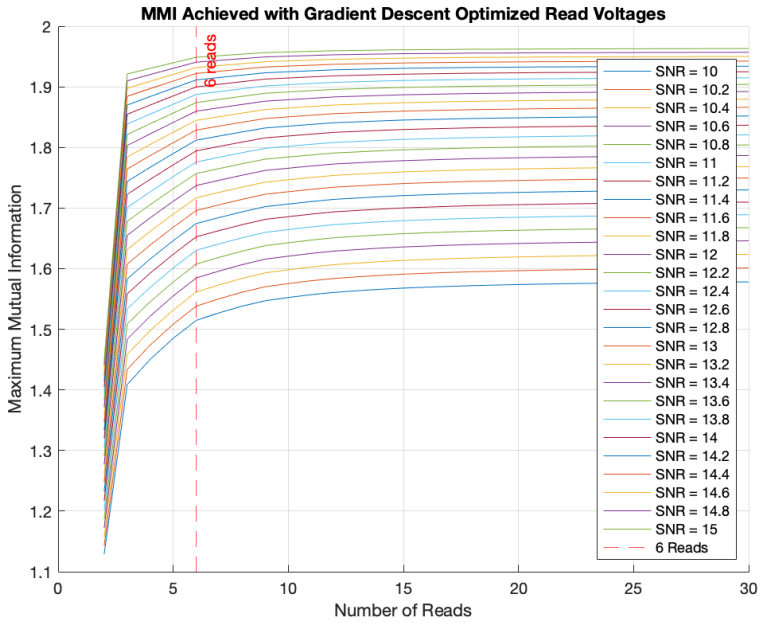


Figure 3.9: Maximum mutual information achievable for $m = [2, 30]$ and SNR $= [10, 15]$(dB).

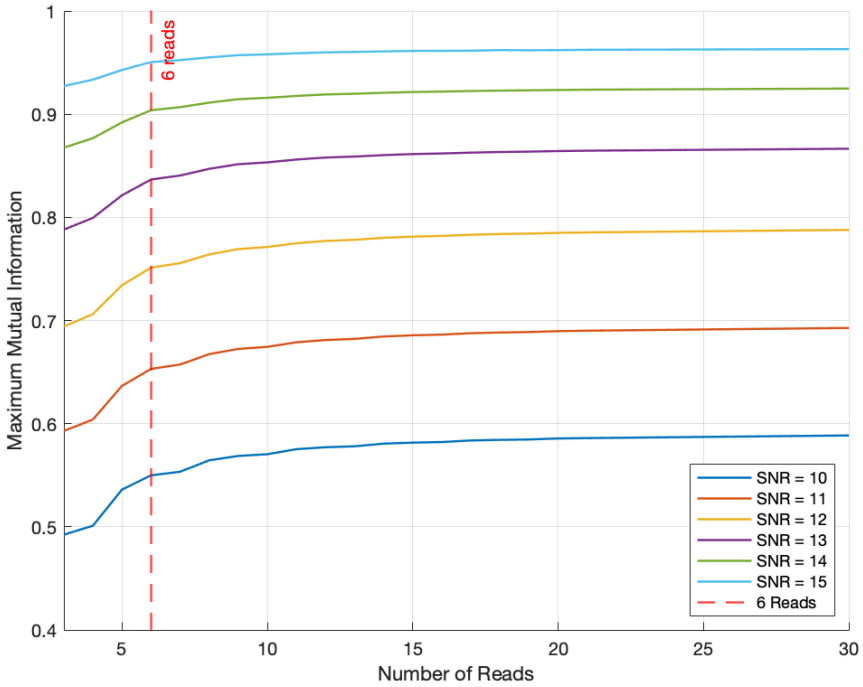


Figure 3.10: Maximum mutual information achievable for the LSB channel using MLC-SCL with $m = [3, 30]$ and SNR $= [10, 15]$(dB).

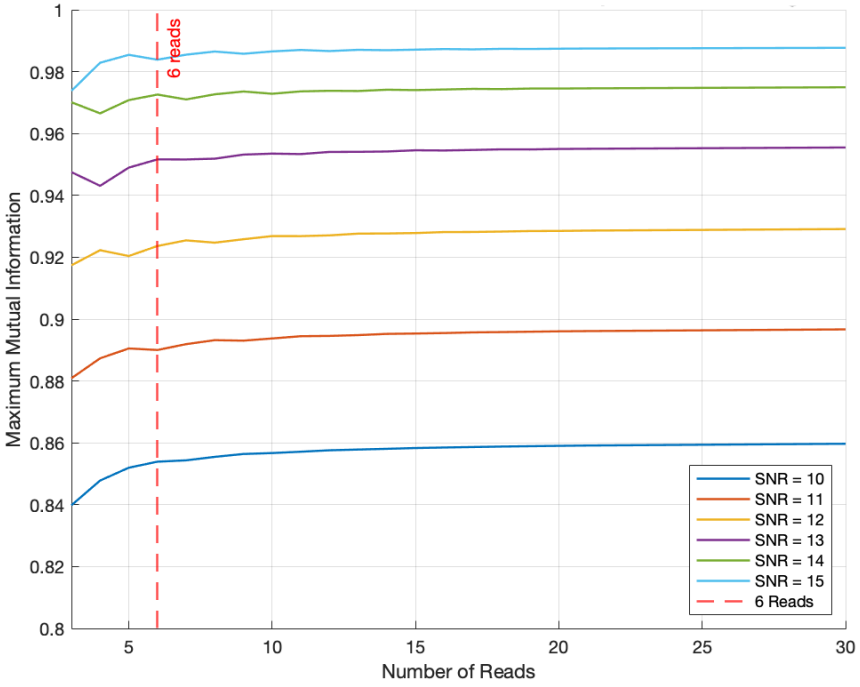


Figure 3.11: Maximum mutual information achievable for the MSB channel using MLC-SCL with $m = [3, 30]$ and SNR $= [10, 15]$(dB).

Chapter 4

LDPC Codes

Error control coding for Flash memory is becoming more important as the capacity of flash memory increases. The increasing number of levels increases sensitivity to variations in signal-to-noise ratio (SNR) from cell to cell and over time due to program/erase (P/E) cycles. This makes stronger error-correction codes, e.g., low-density parity-check (LDPC) codes, necessary. While hard-decision LDPC codes have high error correcting capabilities, they sometimes can be overwhelmed by an inordinate number of errors. That is where iterative soft-decision decoding of LDPC codes, a more analog-based correction algorithm, comes into play. In [6], the performance advantage of soft-decision LDPC decoding over hard-decision LDPC decoding and hard-decision BCH decoding is presented (as shown in Figure 4.1). Therefore, in this chapter, we will give a thorough background review in LDPC codes and iterative soft-decision decoding.

4.1 Overview

Low-density parity-check (LDPC) code [29] is a type of linear error-correcting code (ECC) used to transmit messages through a noisy channel. The term *low-density* refers to the fact that the *parity-check matrix* $\mathbf{H} \in \mathrm{GF}(q)^{(n-k)\times n}$ that characterizes the codewords of the code $\mathcal{C}$ is a sparse matrix. We use GF(q) to denote a *finite* (Galois) field of size q. Throughout the thesis,

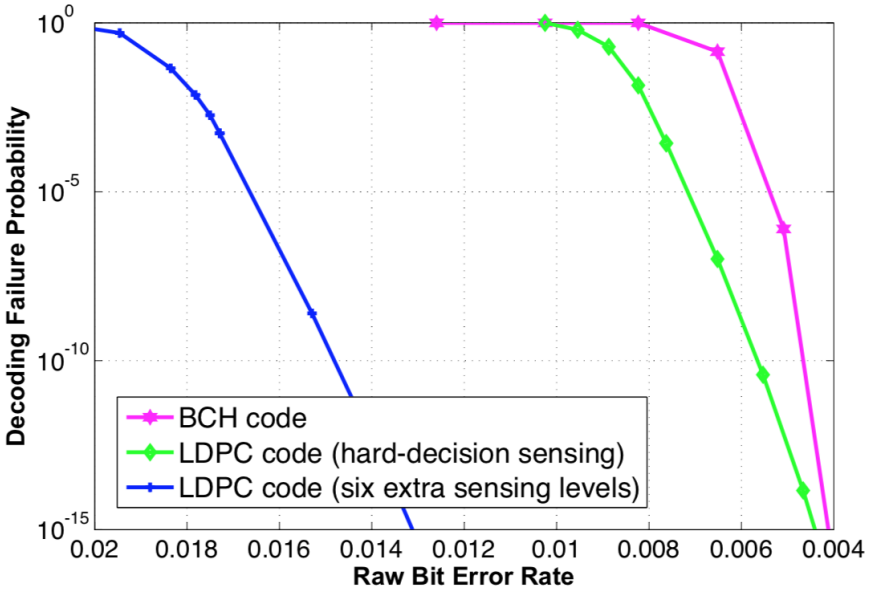


Figure 4.1: Simulated decoding failure probability vs. NAND flash memory raw bit error rate of hard-decision and soft-decision LDPC decoding and BCH code decoding. Both LDPC code and BCH code protect each 4KB user data with 512B coding redundancy [6].

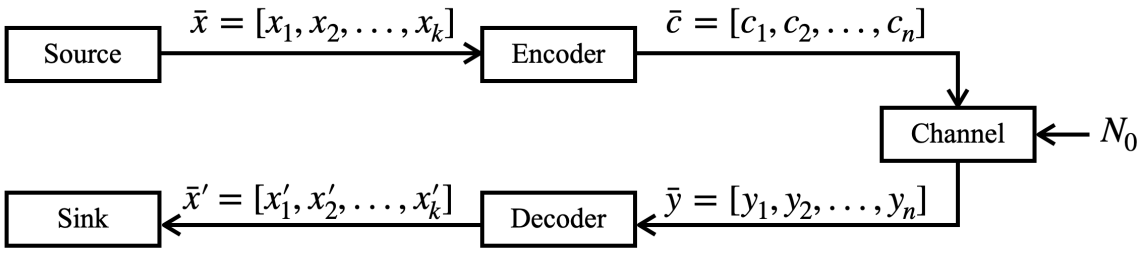


Figure 4.2: Schematic diagram of an error correcting code with rate $R = k/n$.

we will only consider binary ($q = 2$) LDPC codes; therefore the codewords are all of the binary vectors $\mathbf{c} \in \mathrm{GF}(2)^n$ that satisfy $\mathbf{H} \cdot \mathbf{c}^T = \mathbf{0}^T$. In other words, the set of codewords is the null space of $\mathbf{H}$. We call n the length of a codeword and k the dimension of the code (i.e., there are a total of 2^k codewords). Hence the rate of the code is $R = k/n$.

Generator Matrix

A *generator matrix* $\mathbf{G}$ of an LDPC code can be obtained directly (typically not uniquely) from the parity-check matrix. It is a $k \times n$ matrix whose rows form a basis of the code. Obviously the rank of the generator matrix $\mathbf{G}$ is equal to the dimension k of the code $\mathcal{C}$ and it is orthogonal to $\mathbf{H}$, i.e., $\mathbf{G}\mathbf{H}^T = \mathbf{0}$. It is used to encode a message vector $\mathbf{x} \in \mathrm{GF}(2)^k$:

$$\mathbf{x} \mapsto \mathbf{x}\mathbf{G} = \mathbf{c}$$

and since it is full rank, it can be view as a one-to-one mapping: $\mathrm{GF}(2)^k \to \mathrm{GF}(2)^n$. In the special case when the parity-check matrix can be written in the form $\mathbf{H} = \left[-\mathbf{A}^T | \mathbf{I}_{(n-k)\times(n-k)}\right]$ ($\mathbf{I}$ is the identity matrix), the corresponding generator matrix of the form $\mathbf{G} = \left[\mathbf{I}_{k\times k} | \mathbf{A}\right]$ is called *systematic* [30].

Tanner Graph

The $(n-k) \times n$ parity-check matrix can be represented by a Tanner graph (i.e., a bipartite graph), with $(n-k)$ check nodes in one subset of vertices and n variable (or message) nodes in the other subset (see Figure 4.3). The *degree* of a node is the number of edges connected to it. If all variable nodes have degree `l`and all check node has degree `r`, then we call such a code an (`l`, `r`)-regular LDPC code. In contrast, an *irregular* LDPC code is an LDPC code where the degrees of nodes are chosen according to some variable node distribution $\lambda(x)$ and check node distribution $\rho(x)$:

$$\begin{aligned} \lambda(x) &= \sum_{i=1}^{d_v} \lambda_i x^{i-1} \\ \rho(x) &= \sum_{i=1}^{d_c} \rho_i x^{i-1}, \end{aligned} \tag{4.1}$$

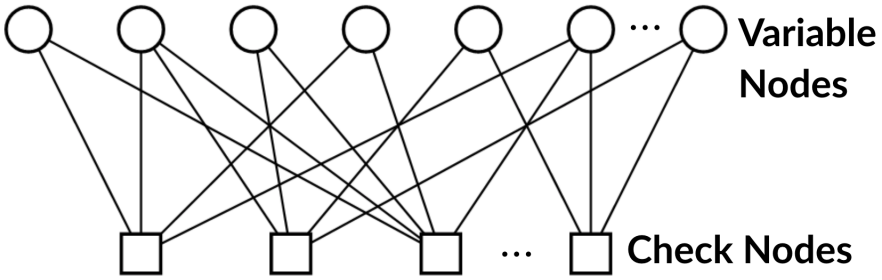


Figure 4.3: A Tanner graph where the circle and the square nodes denote the variable and the check nodes, respectively.

where λ_i and ρ_i denote the fraction of edges connected to variable nodes and check nodes with degree i, respectively, and d_v and d_c denote the maximum degree of variable nodes and check nodes, respectively.

Cycles and Girth

An important property of a Tanner graph is its *cycle* lengths, since these relate directly to the independence of messages passed along edges during iterative decoding. In Section 4.5, we use a technique for analyzing the decoding performance of LDPC codes called density evolution, which assumes the Tanner graph is cycle-free to ensure independence of messages. The *girth* and, more generally, the distribution of cycle lengths of an LDPC code have implications for the accuracy of density evolution as a proxy for actual decoding performance.

First, a *path* is a finite sequence of edges connecting a subset of nodes. A path is called a *cycle* if the first node of the path is the same as the last node with no other nodes being repeated. Thus we define the length of a cycle as the number of edges within that cycle. By definition, a Tanner graph which is a bipartite graph contains no cycles of odd length. This is easily understandable since no two variable nodes nor two check nodes are connected with one edge. The *girth* is the minimum cycle length in a graph. A method of counting the number of cycles of short lengths is presented in [31].

4.2 Code Design

Although following Gallager's [29] introduction of LDPC codes, MacKay and Neal proved that they can achieve near capacity performance [32] over binary-input memoryless symmetric channels (BMSC), it is an ongoing challenge to design the actual code structure that attains such performance. Richardson *et al.* investigated the design of LDPC codes by optimizing the degree structure of the underlying graph using density evolution [15]. There are a few notable codes such as concatenated tree (CT) codes [33], (generalized irregular) repeat-accumulate (RA) codes [34], [35] and codes in [36], [37] that led to the formalism of *multi-edge type* (MET) LDPC codes by Richardson and Urbanke [38]. The structure of MET-LDPC codes is different from standard LDPC codes in the way that they contain diverse edge types which are permuted with a uniformly chosen interleaver. The nodes are characterized by a sequence of degrees, each of which expresses the number of connected edges of a certain type. A specific sequence of degrees is called *edge degree vector* (EDV). The MET-LDPC codes not only outperform the standard LDPC codes, but also possess lower encoding complexity. They are considered an intermediate class of codes that bridges between the standard LDPC codes and protograph codes [39].

Protograph Code

Protograph codes, although not used in this thesis, are worth mentioning because they are a family of codes that exhibit simpler structures, lower complexity and better performance than MET-LDPC codes. A protograph code is defined as an LDPC codes whose Tanner graph is a *derived graph*. First introduced in [7], a *protograph* $\mathcal{G} = (\mathcal{V}, \mathcal{C}, \mathcal{E})$ consist of three sets: N_P variable nodes $v_j \in \mathcal{V}$, M_P check node $c_i \in \mathcal{C}$ and edges $e_{ij} \in \mathcal{E}$. It can be fully specified by an $M_P \times N_P$ *base matrix* $\mathbf{B} = (b_{ij})$ where b_{ij} represents the number of connecting edges from v_j to c_i. Different from conventional LDPC codes, parallel edges are allowed in a protograph, i.e., $b_{ij} \in \mathbb{Z}^+$. Through the "copy-and-permute" operation, we can obtain a larger graph, i.e., the

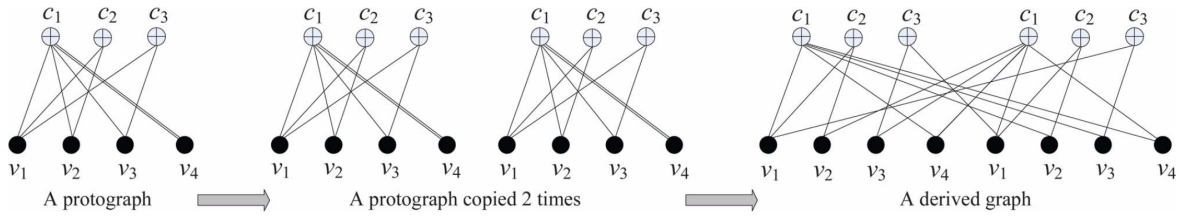


Figure 4.4: A simple example of generating the derived graph from a protograph [7].

derived graph, from a protograph. A simple illustration in [7], which is shown in Figure 4.4, serves as a good explanation of generating a derived graph. The corresponding base matrix of the protograph is:

$$\mathbf{B} = \begin{array}{c} \begin{matrix} v_1 & v_2 & v_3 & v_4 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 1 & 2 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix} \end{array} \begin{matrix} \\ c_1 \\ c_2 \\ c_3 \end{matrix}$$

The derived graph is of size $M \times N$ where $M = zM_P, N = zN_p$, and z is the so-called lifting factor [40]. In practice, one can use the progressive-edge-growth (PEG) algorithm [41] to implement the lifting operation.

4.2.1 QC LDPC Codes

An important category of LDPC codes that is widely adopted in current standards and practices is called the quasi-cyclic (QC) LDPC codes [42]. They not only have competitive performance under iterative decoding, but also can be implemented efficiently because of their underlying structure. A QC-LDPC code has an $m_b L \times n_b L$ parity-check matrix specified by an $m_b \times n_b$ *exponent matrix* (or *base matrix*) we denote as $\mathbf{E}(\mathbf{H})$:

$$\mathbf{E}(\mathbf{H}) = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n_b} \\ a_{21} & a_{22} & \cdots & a_{2n_b} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m_b1} & a_{m_b2} & \cdots & a_{m_bn_b} \end{bmatrix} \tag{4.2}$$

with entries $a_{ij} \in \{-1, 0, 1, ..., L-1\}$. The corresponding parity-check matrix $\mathbf{H}$ is given by:

$$\mathbf{H} = \begin{bmatrix} P^{a_{11}} & P^{a_{12}} & \dots & P^{a_{1n_b}} \\ P^{a_{21}} & P^{a_{22}} & \dots & P^{a_{2n_b}} \\ \vdots & \vdots & \ddots & \vdots \\ P^{a_{m_b1}} & P^{a_{m_b2}} & \dots & P^{a_{m_bn_b}} \end{bmatrix} \tag{4.3}$$

where P is an $L \times L$ circulant permutation matrix (CPM) defined by:

$$\mathbf{P}_{ij}^{a_{ij}} = \begin{cases} 1 & \text{if } i + a_{ij} \equiv j \bmod L \\ 0 & \text{otherwise.} \end{cases} \tag{4.4}$$

In other words, $\mathbf{P}^{a_{ij}}$ is an $L \times L$ binary square matrix that circularly shifts the identity matrix $\mathbf{I}$ to the right by a_{ij} times for any integer $0 \leq a_{ij} < L$. Also, $\mathbf{P}^{-1}$ ($\mathbf{P}^{\infty}$ in some papers) denotes the $L \times L$ zero matrix. We call L the sub-block size, which is denoted as Z in the standards [11].

If we wish for a parity-check matrix with larger girth, we can replace some CPMs in $\mathbf{H}$ with zero matrices of the same size to reduce the number of short cycles and possibly enlarge the girth value. This replacement is called *masking*. Ways of designing QC-LDPC codes with high girths can be found in [43], [42] and [44]. An extension of QC-LDPC codes called lifting, presented in [45], offers a trade-off between performance and memory efficiency.

We use two QC-LDPC codes throughout this thesis. The first is a $(3,4)$-regular QC-LDPC

code of length $n = 36$ and $L = 9$ specified by base matrix:

$$\mathbf{E}(\mathbf{H}) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 3 & 7 \\ 0 & 2 & 6 & 5 \end{bmatrix} \tag{4.5}$$

(see [44]), which we refer to as "Code A". The second one is an irregular QC-LDPC code of length $n = 1944$, rate $2/3$ and $L = 81$ used in the 802.11n Wi-Fi standard [11], which we refer to as "Code B". It's base matrix is specified by:

$$\begin{bmatrix}
61 & 75 & 4 & 63 & 56 & - & - & - & - & - & - & 8 & - & 2 & 17 & 25 & 1 & 0 & - & - & - & - & - & - \\
56 & 74 & 77 & 20 & - & - & - & 64 & 24 & 4 & 67 & - & 7 & - & - & - & - & 0 & 0 & - & - & - & - & - \\
28 & 21 & 68 & 10 & 7 & 14 & 65 & - & - & - & 23 & - & - & - & 75 & - & - & - & 0 & 0 & - & - & - & - \\
48 & 38 & 43 & 78 & 76 & - & - & - & - & 5 & 36 & - & 15 & 72 & - & - & - & - & - & 0 & 0 & - & - & - \\
40 & 2 & 53 & 25 & - & 52 & 62 & - & 20 & - & - & 44 & - & - & - & - & 0 & - & - & - & 0 & 0 & - & - \\
69 & 23 & 64 & 10 & 22 & - & 21 & - & - & - & - & - & 68 & 23 & 29 & - & - & - & - & - & - & 0 & 0 & - \\
12 & 0 & 68 & 20 & 55 & 61 & - & 40 & - & - & - & 52 & - & - & - & 44 & - & - & - & - & - & - & 0 & 0 \\
58 & 8 & 34 & 64 & 78 & - & - & 11 & 78 & 24 & - & - & - & - & - & 58 & 1 & - & - & - & - & - & - & 0
\end{bmatrix} \tag{4.6}$$

where we replace the (-1) elements indicating the zero matrices with dash marks for better visualization. Notice that it has a double diagonal structure at the end of the matrix. It is for the purpose of low encoding complexity which we will discuss further in Section 4.3.2.

4.2.2 LDPC Codes for NAND flash

There are other special types of LDPC code designs that aim for specific purposes for NAND flash, one of which is the *rate-compatible* (RC) LDPC code [46], [47]. For flash memories, it is well known that the raw BER increases with repeated P/E cycles, thus requiring a more powerful (i.e., lower rate) code as PE cycle count increases. One approach is by *puncturing*

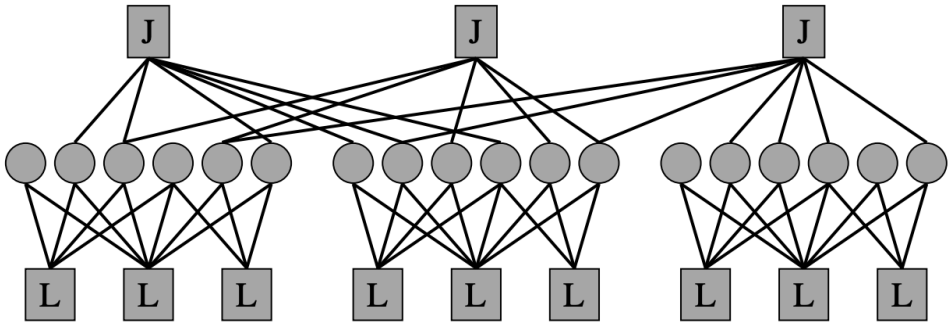


Figure 4.5: Example of a sub-blocked Tanner graph with 3 sub-blocks of length 6 interconnected by 3 joint check nodes [8].

the code, which is starting off with a low-rate code, then gradually discarding the parity bits to achieve higher rates [48]. Another approach is by *extending* the code, that is, to start with a good high-rate code and then successively add more parity-check bits to generate lower-rate codes [49], [50].

Another code design motivated by the storage applications is the *LDPCL* codes (suffix L represents locality) [8]. These codes are designed to have the capability of decoding a sub-block (locally) independent of other sub-blocks for decoding speed and fall back to global decoding (i.e., decoding the entire codeword) when local decoding fails. An example of the graph of such a code is presented in Figure 4.5 where 'L' and 'J' labels denote the local and joint check nodes, respectively.

4.3 Encoding

Recall in Section 4.1 we addressed the basic encoding method using a generator matrix $\mathbf{G}$, as well as the fact that codewords comprise the null space of a parity check matrix $\mathbf{H}$. Thus all codewords $\mathbf{c}$ in codebook $\mathcal{C}$ must satisfy $\mathbf{H}\mathbf{c}^T = \mathbf{0}$. In this section, we discuss some of the encoding methods for LDPC codes.

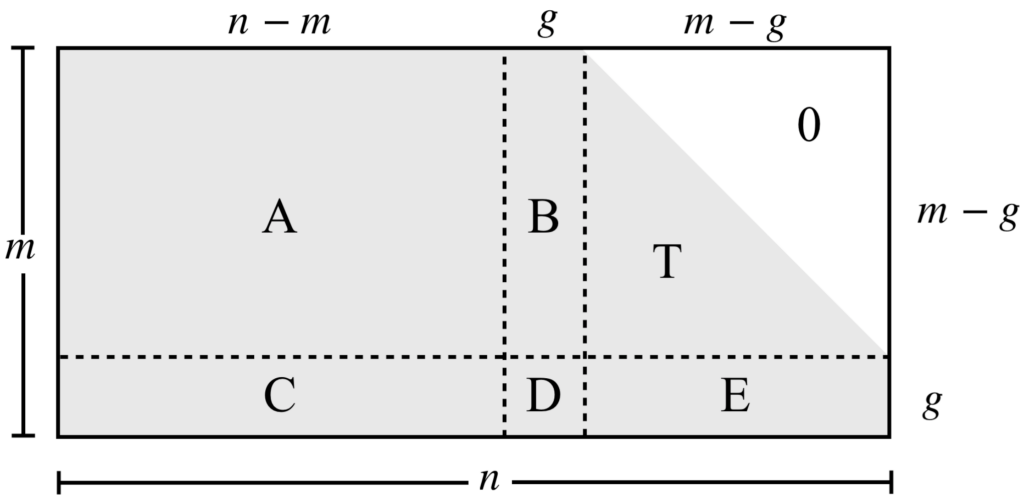


Figure 4.6: The approximate lower triangular form of a parity check matrix where $m = n - k$ and g is typically much less than n [9].

4.3.1 Efficient Encoding

One efficient encoding method applicable to all parity-check matrices is the efficient encoding in [9]. The number of operations required is upper bounded by $0.017^2 n^2 + O(n)$ where n is the code length. The idea is to divide the encoding into two steps: *preprocessing*, which only needs to be done once, and *encoding*. The preprocessing step brings the parity check matrix into an *approximate lower triangular form* as shown in Figure 4.6. Then the encoding step can be done as described in Algorithm 2. The codeword $\mathbf{c}$ can be written as $\mathbf{c} = [\mathbf{s}, \mathbf{p_1}, \mathbf{p_2}]$ where $\mathbf{s}$ is the data vector and $\mathbf{p_1}, \mathbf{p_2}$ are given in equation (4.8).

4.3.2 Double Diagonal Encoding

Another encoding method that is widely used in the existing communication standards is the double/dual diagonal encoding which exploits the double diagonal structure in the parity-check matrix of a QC-LDPC code. A double diagonal LDPC code in a narrow sense has the $m_b \times n_b$ base matrix of the form:

$$\mathbf{E}(\mathbf{H}) = \left[\begin{array}{c|c} \mathbf{A} & \mathbf{D} \end{array} \right]$$

Algorithm 2: Efficient Encoding for LDPC Codes

STEP I : Preprocessing

Input : Full rank parity check matrix $\mathbf{H}$.

Output : An equivalent matrix in approximate lower triangular form such that $-ET^{-1}B+D$ is non-singular

1 [Triangulation] Perform row and column permutations to bring the PCM H into approximate lower triangular form:

$$\mathbf{H} = \begin{bmatrix} A & B & T \\ C & D & E \end{bmatrix} \tag{4.7}$$

2 [Check Singularity] Check if $\phi = -ET^{-1}B+D$ is invertible (non-singular). Perform column permutations to make it so.

STEP II : Encoding

Input : Input the PCM generated form the previous step and a data vector $\mathbf{s} \in \mathrm{GF}(2)^{n-m}$.

Output : Codeword $\mathbf{c} = [\mathbf{s}, \mathbf{p_1}, \mathbf{p_2}], \mathbf{p_1} \in \mathrm{GF}(2)^g, \mathbf{p_2} \in \mathrm{GF}(2)^{n-g}$.

3

$$\begin{cases} \mathbf{p_1} = [-\phi^{-1}(-ET^{-1}A+C)s^T]^T \\ \mathbf{p_2} = [-T^{-1}(A\mathbf{s}^T + B{\mathbf{p_1}}^T)]^T \end{cases} \tag{4.8}$$

where $m = Lm_b = n-k$, $n = Ln_b$ and the square matrix $\mathbf{D}_{m_b \times m_b}$ is the double diagonal part:

$$\mathbf{D}_{m_b \times m_b} = \left[\begin{array}{c|c} \mathbf{d}^T_{1 \times m_b} & \mathbf{D}'_{m_b \times (m_b-1)} \end{array} \right] = \left[\begin{array}{c|ccccccc} d & 0 & & & & & & \\ & 0 & 0 & & & & & \\ & & 0 & \ddots & & & & \\ 0 & & & \ddots & \ddots & & & \\ & & & & \ddots & 0 & & \\ & & & & & 0 & 0 & \\ d & & & & & & 0 & \end{array} \right]$$

where $\mathbf{d}^T$ is a column vector of weight 3 and $d \in \mathbb{Z}^{++}$. IEEE 802.11n LDPC codes are structured this way and support codeword lengths of $n = 648$, 1296, and 1944 with sub-block sizes $L =$ 27, 54, and 81, respectively. The supported code rates are $1/2$, $2/3$, $3/4$ and $5/6$ and the corresponding base matrices are shown in Table D.1 for the length $n = 1944$ code.

In a broader sense, a double diagonal LDPC code can have a base matrix of the form:

$$\mathbf{E}(\mathbf{H}) = \begin{bmatrix} \mathbf{A} & \mathbf{D} & \mathbf{0} \\ \mathbf{B} & \mathbf{C} & \mathbf{I} \end{bmatrix}$$

where **D** has the double diagonal structure discussed previously, **0** is an all-zero matrix and **I** is the identity matrix (this type is common in the 5G NR standards). Columns **A** and **B** are the information columns; Columns **D** and **C** are the core parity columns; Columns **0** and **I** are considered as the extension parity columns. Note that, given such a base matrix, one can perform the *rate matching* operation to obtain a sub-PCM of another desired rate, but this is a topic reserved for another time. Below, in Algorithm 3, we will only discuss the encoding for the first type but it can be easily extended to encode the second type. We will use it throughout the rest of this thesis to encode our Code B (equation (4.6)). One can also refer to [51] [52] and [53] for more hardware-efficient encoding methods.

4.4 Decoding

In this section, we discuss some decoders for LDPC codes in general. We will first go through the standard LDPC decoding scheme, the belief-propagation (BP) decoding, in detail because this decoding method is closely related to our discussion of density evolution in the later sections. Next, we will discuss an approximated BP decoding called the min-sum algorithm which is a more hardware-efficient decoding method that is widely used in practice. We will describe its decoding process in pseudo-code due to its importance. Besides these two decoders, there exists a more recently proposed decoder called the *finite alphabet iterative decoder* (FAID) which is not directly relevant to this thesis but is worth mentioning briefly here due to its performance properties. On finite length codes, BP decoding in the low error-rate region can exhibit what is called the *error floor*, i.e., a flattening of the performance, due to the presence of cycles forcing

Algorithm 3: Double Diagonal Encoding Algorithm

Input : A data vector $\mathbf{s} = [s_1, s_2, \cdots, s_k] \in \mathrm{GF}(2)^{(1\times k)}$, L, and a $m_b \times n_b$ base matrix of the form $\mathbf{E}(\mathbf{H}) = \left[\ \mathbf{A} \mid \mathbf{d^T} \mid \mathbf{D'}\ \right]$ with elements e_{ij}, where $k = L(n_b - m_b)$

Output : The codeword vector $\mathbf{c} \in \mathrm{GF}(2)^{(1\times n)}$

/* ----- Main Code ----- */

$\mathbf{c} = \mathbf{0}_{(1\times n_b \cdot L)}$

$\mathbf{c}\big[1 : (n_b - m_b)L\big] = \mathbf{s}$

for $i = 1 \rightarrow m_b$ **do**

  for $j = 1 \rightarrow n_b - m_b$ **do**

    $\mathbf{t}_{1\times L} = \mathrm{mod2}\left\{ \mathbf{t} + \mathrm{shift}\Big(\mathbf{c}\big[(j-1)L+1 : jL\big], e_{ij} \Big) \right\}$

  end

end

$r =$ the index of the 0 element in $\mathbf{d}^T$

$p = e_{r,(n_b - m_b + 1)}$

$\mathbf{c}\ \left[\ (n_b - m_b)L + 1 : (n_b - m_b + 1)L\ \right] = \mathrm{shift}\ \Big(\ \mathbf{t}, L - p\ \Big)$

for $i = 1 \rightarrow m_b$ **do**

  $\mathbf{t} = \mathbf{0}_{(1\times L)}$

  for $j = 1 \rightarrow n_b - m_b + i$ **do**

    $\mathbf{t} = \mathrm{mod2}\left\{ \mathbf{t} + \mathrm{shift}\Big(\mathbf{c}\big[(j-1)\cdot L + 1 : j \cdot L\big], e_{ij} \Big) \right\}$

  end

  $\mathbf{c}\big[(n_b - m_b + i)L + 1 : (n_b - m_b + i + 1)L\big] = \mathbf{t}$

end

/* ----- Functions ----- */

Function $\mathbf{y} =$ `shift` $(\mathbf{x}, k)$ **:**

  if *if k==-1* **then**

    $\mathbf{y} = \mathbf{0}_{(1\times length(\mathbf{x}))}$

  else

    $\mathbf{y} = \left[\ \mathbf{x}(k+1 : \mathrm{end})\ \mathbf{x}(1:k)\ \right]$

  end

  return

the decoding process to converge to *trapping sets*. FAIDs were introduced to overcome this problem [54], [55]. They are designed to optimize the error-correcting capability in the error floor region [56].

4.4.1 Belief Propagation

A common decoding scheme for LDPC codes is the Belief Propagation (BP) decoding [29], also known as the *sum-product* algorithm and a sub-class of *message passing* algorithms. It is a soft-input soft-output iterative algorithm that passes probability messages, or "beliefs", along the edges of a Tanner graph (see Figure 4.3).

In the case of decoding LDPC codes specified by the $m \times n$ PCM $\mathbf{H}$, each variable node (VN) takes real values (*a priori* information) from the channel along with the latest communicated values from its connecting check nodes (CNs). It then computes updated information to send to each of its connecting CNs. Each CN, which represents a parity-check constraint of the code, then computes updated information from its received messages to return to its connecting VNs. This process make up one decoding iteration.

Let us use $\mathtt{m}_{\mathtt{v}_j\mathtt{c}_i}^{(l)}$ and $\mathtt{m}_{\mathtt{c}_i\mathtt{v}_j}^{(l)}$ to denote the messages passed from VN $\mathtt{v}_j$ to CN $\mathtt{c}_i$ and from CN $\mathtt{c}_i$ to VN $\mathtt{v}_j$, respectively, at iteration l where $i \in \{1,2,\cdots,m\}$ and $j \in \{1,2,\cdots,n\}$. Also let $\mathbf{x} = (x_1, x_2, \cdots, x_n) \in \mathrm{GF}(2)^{1\times n}$ denote the codeword and $\mathbf{y} = (y_1, y_2, \cdots, y_n) \in \mathbb{R}^{1\times n}$ be the received vector. Initially at iteration 0, each VN $\mathtt{v}_j$ calculates the log-likelihood ratios (LLRs) defined as $\ell^{(ch)}$:

$$\ell_j^{(ch)} = \ln\left(\frac{\Pr(x_j = 0|y_j)}{\Pr(x_j = 1|y_j)}\right) \tag{4.9}$$

where $\Pr(x_j = 0|y_j)$ and $\Pr(x_j = 1|y_j)$ are the conditional probabilities of $x_j = 0$ and $x_j = 1$ given y_j, respectively. Intuitively, we can think of the magnitude of LLRs as the amount of confidence or certainty that a bit is 0 or 1. If the LLR > 0, then the decision of the bit is 0. Otherwise, the decision of the bit is 1. The job of the BP decoder is to iteratively update these LLRs by passing them between the VNs and CNs to obtain the output.

VN Update Rule

Let's first discuss the VN update rule, i.e., the calculation of the message passed from VN to CNs ($\mathtt{m}_{\mathtt{v}_j\mathtt{c}_i}^{(l)}$). One important feature of BP decoding is that only *extrinsic* information is being passed, e.g., when calculating $\mathtt{m}_{\mathtt{v}_j\mathtt{c}_i}^{(l)}$, $\mathtt{m}_{\mathtt{c}_i\mathtt{v}_j}^{(l-1)}$ is excluded from consideration and vice versa with the superscripts of the message swapped. To give an example, let us consider the case depicted in Figure 4.7a where VN $\mathtt{v}_j$ is of degree 3 with edges $(e_1, e_2, e_3) \in \mathcal{E}_j$ connecting CNs $\mathtt{c}_i$, $\mathtt{c}_6$, and $\mathtt{c}_7$. The message $\mathtt{v}_j$ passes to $\mathtt{c}_i$ is:

$$\mathtt{m}_{\mathtt{v}_j\mathtt{c}_i}^{(l)} = \ell_j^{(ch)} + \ell_{e_2}^{(l-1)} + \ell_{e_3}^{(l-1)}$$

As you can see, we are excluding $\ell_{e_1}^{(l-1)}$ from the summation. Note that in our notation, $\ell_{e_2}^{(l-1)} \equiv \ell_6^{(l-1)}$ and $\ell_{e_3}^{(l-1)} \equiv \ell_7^{(l-1)}$. Extending to general cases, the VN messages are given by:

$$\mathtt{m}_{\mathtt{v}_j\mathtt{c}_i}^{(l)} = \ell_j^{(ch)} + \sum_{e_d \in \mathcal{E}_j \setminus \{e_d = i\}} \ell_{e_d}^{(l-1)} \tag{4.10}$$

where $\ell^{(0)}$ is initialized to 0.

CN Update Rule

To explain the CN update rule for message $\mathtt{m}_{\mathtt{c}_i\mathtt{v}_j}^{(l)}$, let us consider another simple example depicted in Figure 4.7b where a CN $\mathtt{c}_5$ is connected to three other VNs $\mathtt{v}_1$,$\mathtt{v}_6$ and $\mathtt{v}_7$ via edges $(e_1, e_2, e_3) \in \mathcal{E}_5$ and we wish to compute $\mathtt{m}_{\mathtt{c}_5\mathtt{v}_1}^{(l)}$. For simplicity, all messages are expressed as likelihood ratios (LRs) denoted as $\mathcal{L}$, rather than LLRs. We have:

$$\begin{cases} \mathcal{L}_6 = \frac{\Pr(x_6=0|y_6)}{\Pr(x_6=1|y_6)} \triangleq \frac{p6_0}{p6_1} \\ \mathcal{L}_7 = \frac{\Pr(x_7=0|y_7)}{\Pr(x_7=1|y_7)} \triangleq \frac{p7_0}{p7_1} \end{cases}$$

and

$$\begin{aligned}
\mathcal{L}_6 \oplus \mathcal{L}_7 &\equiv \mathcal{L}_{e_2} \oplus \mathcal{L}_{e_3} \\
&= \frac{p6_1 p7_1 + p6_0 p7_0}{p6_0 p7_1 + p6_1 p7_0} = \frac{1 + \frac{p6_0 p7_0}{p6_1 p7_1}}{\frac{p6_0}{p6_1} + \frac{p7_0}{p7_1}} = \frac{1 + \mathcal{L}_6 \mathcal{L}_7}{\mathcal{L}_6 + \mathcal{L}_7} = \frac{2 + 2\mathcal{L}_6 \mathcal{L}_7}{2(\mathcal{L}_6 + \mathcal{L}_7)} \\
&= \frac{(\mathcal{L}_6 + 1)(\mathcal{L}_7 + 1) + (\mathcal{L}_6 - 1)(\mathcal{L}_7 - 1)}{(\mathcal{L}_6 + 1)(\mathcal{L}_7 + 1) - (\mathcal{L}_6 - 1)(\mathcal{L}_7 - 1)} \\
&= \frac{1 + \frac{\mathcal{L}_6 - 1}{\mathcal{L}_6 + 1} \frac{\mathcal{L}_7 - 1}{\mathcal{L}_7 + 1}}{1 - \frac{\mathcal{L}_6 - 1}{\mathcal{L}_6 + 1} \frac{\mathcal{L}_7 - 1}{\mathcal{L}_7 + 1}}
\end{aligned} \tag{4.11}$$

where $\oplus$ denotes the addition operation in modulo 2, equivalent to the XOR operation. Thus in general, when the CN $\mathtt{c}_i$ has d_c connecting edges $\mathcal{E}_i = (e_1, e_2, ...e_{d_c})$ connected to VNs $\mathtt{v}_{e_d}$, the message $\mathtt{m}^{(l)}_{\mathtt{c}_i\mathtt{v}_j}$ is given by:

$$\mathtt{m}^{(l)}_{\mathtt{c}_i\mathtt{v}_j} = \frac{1 + \prod_{e_d \in \mathcal{E}_i \setminus \{e_d = j\}} \frac{\mathcal{L}_{e_d}^{(l-1)} - 1}{\mathcal{L}_{e_d}^{(l-1)} + 1}}{1 - \prod_{e_d \in \mathcal{E}_i \setminus \{e_d = j\}} \frac{\mathcal{L}_{e_d}^{(l-1)} - 1}{\mathcal{L}_{e_d}^{(l-1)} + 1}} \tag{4.12}$$

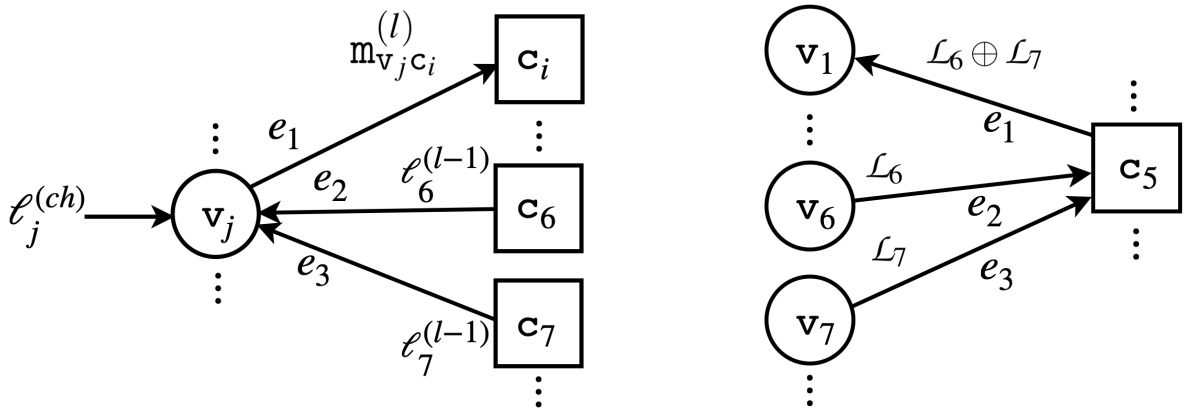
Moreover, coming back to working with LLRs (i.e., $\ell = \ln \mathcal{L}$), we have:

$$\frac{\mathcal{L} - 1}{\mathcal{L} + 1} = \tanh(\frac{\ell}{2})$$

and finally, the LLR update rule at CNs is given by:

$$\begin{aligned}
\mathtt{m}^{(l)}_{\mathtt{c}_i\mathtt{v}_j} &= 2\tanh^{-1}\Big(\prod_{e_d \in \mathcal{E}_i \setminus \{e_d = j\}} \tanh(\frac{\ell_{e_d}^{(l-1)}}{2})\Big) \\
&= 2\tanh^{-1}\Big[\exp\Big(\sum_{e_d \in \mathcal{E}_i \setminus \{e_d = j\}} \ln\big[\tanh(\frac{\ell_{e_d}^{(l-1)}}{2})\big]\Big)\Big]
\end{aligned} \tag{4.13}$$

Given equations (4.10) and (4.13), the iterations are repeated accordingly, usually until either the maximum number of iterations $l_{\max}$ is reached or the average magnitude of LLR values



(a) VN update

(b) CN update

Figure 4.7: A simple example of message passing at a (a) VN and (b) CN, both of degree 3. We use e_d to denote the d^{th} edge in our consideration. ℓ and $\mathcal{L}$ are the LLR and LR messages passed along the edges, respectively. $\ell^{(ch)}$ denotes the observed channel values.

exceeds a certain threshold. As such, we show the complete decoding process in Algorithm 4.

4.4.2 Min-sum decoding

Although BP decoding gives good performance, it is considered too computationally expensive for hardware implementation due to the floating point calculations. An approximated method that deals only with positive real numbers and XOR operations, the *min-sum* (MS) algorithm greatly reduces the complexity of BP [57] [58]. Thus we will briefly present the essence of this algorithm below (Algorithm 5). The reader should acknowledge the fact that while MS is hardware efficient, its ultimate performance is often much worse than that of BP. To compensate for this degradation, a linear post processing (normalization) of the check node messages can be effective [59] [60]. The optimal normalization factor can be found using density evolution methods such that it achieves near BP decoding capability. In [61], an improved 2D normalization method for MS decoding is presented that gives considerably better performance than standard MS and 1D normalized MS decoding.

In the min-sum algorithm, we use **S** to denote a storage matrix of the same size as the

Algorithm 4: Belief Propagation Decoding Algorithm

Input : The received LLR vector $\mathbf{y} = [y_1, y_2, \cdots, y_n] \in \mathbb{R}^{(1\times n)}$ given by the channel, the $m \times n$ PCM $\mathbf{H}$ and $l_{\max}$.

Output : The decoded codeword $\mathbf{c}' \in \mathrm{GF}(2)^{(1\times n)}$

/* ----- Initialization ----- */

Set iteration count $l = 0$;

Initialize two $m \times n$ storage matrix $\mathbf{V} = \mathbf{C} = \mathbf{H}$;

Set $\tilde{v}_{ij}$, the non-zero entries of $\mathbf{V}$, to be y_j for all $j \in [1, n]$;

/* ----- Message Passing Loop ----- */

while $l < l_{max}$ **do**

  /* ----- VN update $\mathbf{C}$ ----- */

  for *all non-zero entry* (i, j) *pairs* **do**

    $\tilde{c}_{ij} = \mathtt{m}^{(l)}_{\mathtt{v}_j\mathtt{c}_i}$ given by equation (4.10);

  end

  /* ----- CN update $\mathbf{V}$ ----- */

  for *all non-zero entry* (i, j) *pairs* **do**

    $\tilde{v}_{ij} = \mathtt{m}^{(l)}_{\mathtt{c}_i\mathtt{v}_j}$ given by equation (4.13);

  end

  $l{+}{+}$;

end

/* ----- Decode LLR to Binary ----- */

for $j = 1 \to n$ **do**

  c'_j = avg(non-zero entries of column j in $\mathbf{C}$);

end

$\mathbf{c}' = \mathbf{c}' < 0$;

parity-check matrix of the code $\mathbf{H} \in \mathrm{GF}(2)^{(n-k)\times n}$. We let h_{ij} and s_{ij} denote the entries in $\mathbf{H}$ and $\mathbf{S}$ respectively. Also we let $\tilde{s}_{ij}$ denote the non-zero entries in $\mathbf{S}$. The main loop of the algorithm generally consist of two parts, row operation and column operation, and is fully described in Algorithm 5.

4.5 Density Evolution

Density evolution, proposed in [62], is the most powerful analytical asymptotic tool for determining the capacity for LDPC codes under message passing decoding algorithms. It relies

Algorithm 5: Min-sum Decoding Algorithm

Input : The received vector $\mathbf{r} = [r_1, r_2, \cdots, r_n]$ given by the channel.
Output : The decoded codeword $\mathbf{c}' \in \mathrm{GF}(2)^{(1\times n)}$

/* ----- Initialization ----- */

Set iteration count $l = 0$ and maximum iteration number $l_{\max}$;

Initialize a $(n-k)\times n$ storage matrix $\mathbf{S} = \begin{cases} s_{ij} = r_j & \text{if } h_{ij} = 1 \\ s_{ij} = 0 & \text{otherwise} \end{cases}$;

while $l < l_{max}$ **do**

  /* ----- Row Operation ----- */

  for $i = 1 \to (n-k)$ **do**

    $\min_1$ = minimum absolute value of all non-zero entries in row i;

    $\min_2$ = the next minimum absolute value;

    sign = $\begin{cases} +1 & \text{if } \prod_{j=1}^{n} \tilde{s}_{ij} > 0 \\ -1 & \text{otherwise} \end{cases}$;

    Set magnitude of all $\tilde{s}_{ij} = \min_1$;

    Set magnitude of the original $\min_1$ value = $\min_2$;

    Element-wise multiplication of row i with sign;

  end

  /* ----- Column Operation ----- */

  for $j = 1 \to n$ **do**

    $c'_j = r_j + \sum_{i=1}^{n-k} s_{ij}$;

    // Update codeword

    for $i = 1 \to (n-k)$ **do**

      $\hat{s}_{ij} = c'_j - \hat{s}_{ij}$;

    end

  end

  $l{+}{+}$;

end

/* ----- Decode LLR to Binary ----- */

$\mathbf{c}' = \mathbf{c}' < 0$;

on the assumption of message independence, which can be satisfied as the codeword length tends to infinity such that the corresponding graph is close to cycle-free. Another important assumption is the channel symmetry. This is the reason for Section 3.3.2 where we established a symmetric channel labeling (SCL) specifically for MLC LSB channels in the flash memory scenario. In this section, we will first introduce the fundamentals of the standard (continuous) density evolution,

followed by an approximated method called the *Gaussian approximation* method that greatly simplifies the process for BAWGNC [63].

4.5.1 Standard Density Evolution

Given the VN and CN update rules (equations 4.10 and 4.13), suppose we let $p_{\mathtt{v}}^{(l)}$ and $p_{\mathtt{c}}^{(l)}$ denote the probability density functions (pdf) of VNs and CNs, respectively at iteration l. Then we have the density function $p_{\mathtt{v}}^{(l)}$ of the outgoing LLRs at the variable node $\mathtt{v}$:

$$p_{\mathtt{v}}^{(l)} = p^{(ch)} \circledast \left[p_{\mathtt{c}}^{(l-1)}\right]^{\circledast(d_v-1)} \tag{4.14}$$

where $p^{\circledast n}$ is a short hand for the pdf $\mathtt{p}$ convolved with itself $n-1$ times (e.g., $p^{\circledast 3} \equiv p \circledast p \circledast p$), $p^{(ch)}$ denotes the pdf given by the channel LLRs and d_v denote the degree of the target VN $\mathtt{v}$. On the other hand, the density $p_{\mathtt{c}}^{(l)}$ of the outgoing LLRs at the check node $\mathtt{c}$ is given by:

$$p_{\mathtt{c}}^{(l)} = \Gamma^{-1}\left(\left[\Gamma(p_{\mathtt{v}}^{(l-1)})\right]^{\circledast(d_c-1)}\right) \tag{4.15}$$

where $\Gamma(\cdot)$ is an invertible operator on probability densities defined in [10]. Furthermore, with:

$$\lambda(p) \triangleq \sum_{i\geq 2} \lambda_i (p)^{\circledast(i-1)}$$

$$\rho(p) \triangleq \sum_{i\geq 2} \rho_i (p)^{\circledast(i-1)}$$

where λ_i and ρ_i are as defined in equation (4.1), we have the updating equation of VN densities:

$$p_{\mathtt{v}}^{(l)} = p^{(ch)} \circledast \lambda\left[\Gamma^{-1}\left(\rho\left[\Gamma(p_{\mathtt{v}}^{(l-1)})\right]\right)\right] \tag{4.16}$$

While performing density evolution analysis, we usually assume that all-0 codeword is being transmitted. Naturally, with the conventional BPSK mapping where symbols '0' and '1'

get mapped to positive values and negative values, respectively, we have the probability of error after l iterations given by:

$$P_e^{(l)} = \int_{-\infty}^{0} p_{\mathrm{v}}^{(l)}(z)dz. \tag{4.17}$$

A graphical example that visualizes the density evolution process borrowed from [10] is given in Figure 4.8 for a code with degree distributions:

$$\begin{aligned}\lambda(x) =& 0.212332x + 0.197596x^2 + 0.0142733x^4 + 0.0744898x^5 + 0.0379457x^6 \\ &+ 0.0693008x^7 + 0.086264x^8 + 0.00788586x^{10} + 0.0168657x^{11} + 0.283047x^{30} \\ \rho(x) =& x^8\end{aligned} \tag{4.18}$$

Since the channel is a BAWGNC with noise variance $\sigma^2 = 0.93^2$, the channel density function $p^{(ch)} \sim \mathcal{N}(\frac{2}{\sigma^2}, \frac{4}{\sigma^2})$ as depicted in the upper left corner of the figure $p_{\mathrm{v}}^{(0)} = p^{(ch)}$. The $p_{\mathrm{c}}^{(0)}$ is always initialized as a unit-impulse with all its mass at 0 because the initial message from any CN is 0. We can observe that, with the assumption of all-0 codeword being transmitted, both p_{v} and p_{c} densities gradually shift towards the right as iteration progresses. Intuitively, this indicates that the decoder becomes more confident in its decision, while simultaneously having larger fraction of bits being decided as the '0' symbol, i.e., reducing P_e.

4.5.2 Gaussian Approximation

A method that greatly simplifies the process of density evolution is called *Gaussian approximation* proposed in [63]. It treats the messages exchanged in BP decoding as Gaussian random variables with mean μ and variance 2μ. By doing so, we only need to track the mean throughout each iteration since it fully specifies the entire LLR density function. Equation (4.10) simply becomes

$$\mu_{\mathrm{v}}^{(l)} = \mu^{(ch)} + (d_v - 1)\mu_{\mathrm{c}}^{(l-1)} \tag{4.19}$$

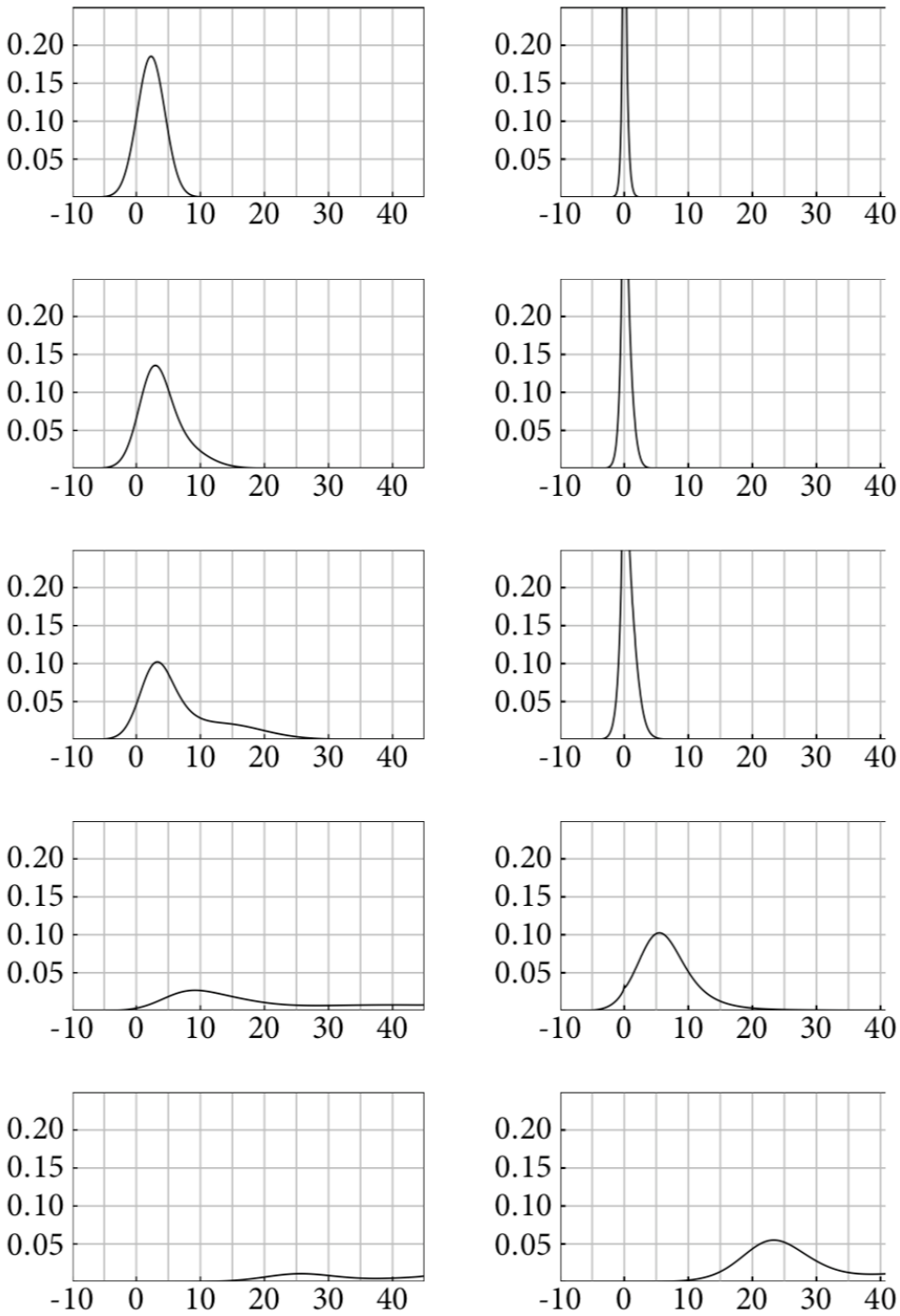


Figure 4.8: The evolution of the LLR densities at the VNs (left column) and CNs (right column) for iterations $l = 0, 5, 10, 50$, and 140 corresponding to each row from top to bottom, for BAWGNC with noise variance $\sigma^2 = 0.93^2$ with a code specified by the degree distribution in equation (4.18) [10].

where $\mu_{\mathtt{v}}^{(l)}$, $\mu_{\mathtt{c}}^{(l)}$ and $\mu^{(ch)}$ denote the mean of the density functions $p_{\mathtt{v}}^{(l)}$, $p_{\mathtt{c}}^{(l)}$, and $p^{(ch)}$, respectively. The indices j are omitted because $\mathtt{v}_j$'s are i.i.d. for $1 \leq i < d_v$. And in general, the VN update rule becomes:

$$\mu_{\mathtt{v}}^{(l)} = \sum_i \lambda_i \left[\mu^{(ch)} + i \cdot \mu_{\mathtt{c}}^{(l-1)} \right] \tag{4.20}$$

The update mean $\mu_{\mathtt{c}}^{(l)}$ can be calculated by taking the expected value on both sides of equation (4.13)

$$\mathbb{E}\left[\tanh \frac{\mathtt{m}_{\mathtt{cv}}^{(l)}}{2} \right] = \mathbb{E}\left[\tanh \frac{\mathtt{m}_{\mathtt{vc}}^{(l)}}{2} \right]^{d_c - 1} \tag{4.21}$$

and since we're approximating $\mathtt{m}_{\mathtt{cv}}$ as Gaussian $\mathcal{N}(\mu_{\mathtt{c}}, 2\mu_{\mathtt{c}})$, the expectation $\mathbb{E}\left[\tanh \frac{\mathtt{m}_{\mathtt{cv}}}{2} \right]$ depends only on the mean $\mu_{\mathtt{c}}$:

$$\mathbb{E}\left[\tanh \frac{\mathtt{m}_{\mathtt{cv}}}{2} \right] = \frac{1}{\sqrt{4\pi\mu_{\mathtt{c}}}} \int_{\mathbb{R}} \tanh \frac{\mathtt{m}_{\mathtt{cv}}}{2} \exp\left[-\frac{(\mathtt{m}_{\mathtt{cv}} - \mu_{\mathtt{c}})^2}{4\mu_{\mathtt{c}}} \right] d\mathtt{m}_{\mathtt{cv}} \tag{4.22}$$

If we let $f(\mu)$ be:

$$f(\mu) = \begin{cases} 1 - \frac{1}{\sqrt{4\pi\mu}} \int_{\mathbb{R}} \tanh \frac{u}{2} \exp\left[-\frac{(u-\mu)^2}{4\mu} \right] du, & \text{if } \mu > 0 \\ 1, & \text{if } \mu = 0 \end{cases}$$

then $f(\mu)$ can be approximated by $\phi(\mu)$ [63]:

$$\phi(\mu) = \begin{cases} e^{-0.4527\mu^{0.86} + 0.0218}, & \mu < 10 \\ \sqrt{\frac{\pi}{\mu}} e^{-\frac{\mu}{4}} (1 - \frac{20}{7\mu}), & \mu \geq 10 \end{cases}$$

The corresponding CN update rule at one specific CN with degree d_c is:

$$\mu_{\mathrm{c}}^{(l)} = \phi^{-1}\left(1 - \left[1 - \phi\left(\mu_{\mathrm{v}}^{(l-1)}\right)\right]^{d_c - 1}\right) \tag{4.23}$$

Averaging over all check node degrees, we have the CN update rule as:

$$\mu_{\mathrm{c}}^{(l)} = \sum_j \rho_j \phi^{-1}\left(1 - \left[1 - \phi\left(\mu_{\mathrm{v}}^{(l-1)}\right)\right]^{j-1}\right) \tag{4.24}$$

Finally, given degree distribution of the code $[\lambda(x), \rho(x)]$, with equations (4.20) and (4.24) we have the following update equation:

$$\mu_{\mathrm{c}}^{(l)} = \sum_j \rho_j \phi^{-1}\left(1 - \left[1 - \sum_i \lambda_i \phi\left(\mu^{(ch)} + (i-1)\mu_{\mathrm{c}}^{(l-1)}\right)\right]^{j-1}\right) \tag{4.25}$$

Figure 4.9 shows an example of the evolution of P_e throughout each message-passing iteration using Gaussian approximation with a (3, 6)-regular LDPC code and different noise parameters σ, where P_e is given by $\Phi\left(\frac{-\mu_{\mathrm{v}}}{\sigma}\right)$ and $\Phi(\cdot)$ is the cumulative distribution function (CDF) of the standard normal distribution ($\mathcal{N}(0,1)$). Overall, Gaussian approximation serves as an alternative analysis method for the additive white Gaussian noise channel reducing what used to be the evolution of infinite-dimensional density space into the evolution in a single parameter. However, the MSB and LSB flash memory channels are not AWGN channels, so Gaussian approximation is not applicable. Therefore, in the next chapter, we discuss a more relevant computational method, discretized density evolution.

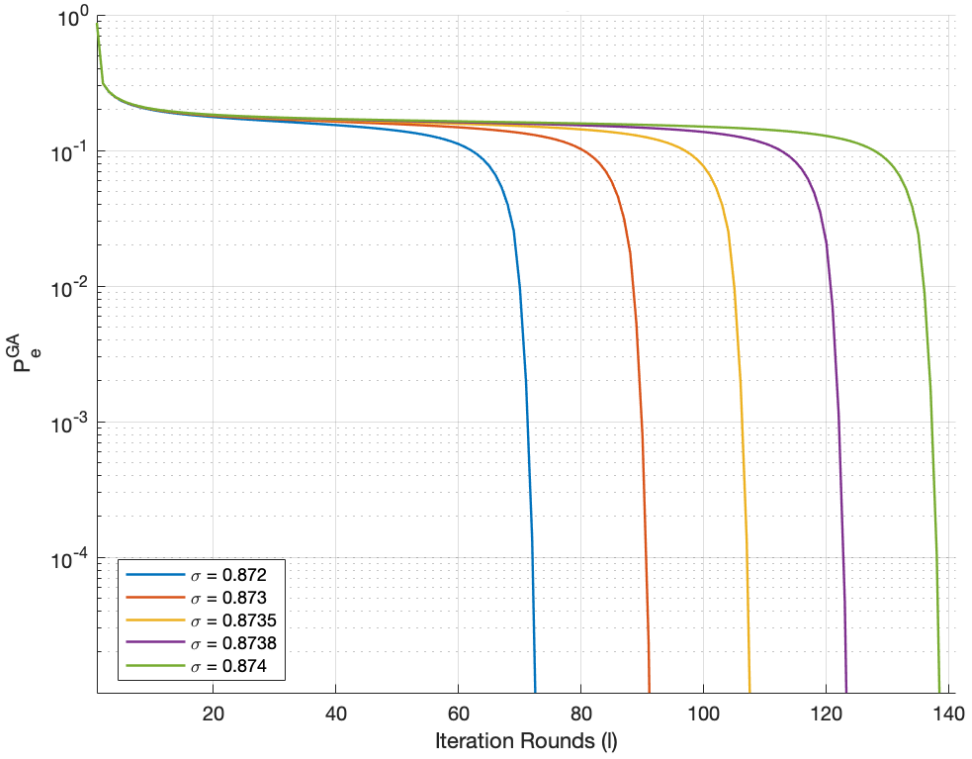


Figure 4.9: Evolution of probability of errors P_e given by Gaussian approximation methods of a (3, 6)-regular LDPC code with different noise parameters σ.

Chapter 5

Read Threshold and LDPC Codes

In this chapter, we discuss the approach of using density evolution as an analytical method of refining the read threshold positions such that it gives the minimum BER possible, as shown in Figure 5.1. This approach was proposed in [19], where a brute force search was used to identify optimal read thresholds for use with an LDPC code. In an attempt to reproduce the results in [19], we developed an alternative, algorithmic approach to to the problem and systematically explored several technical issues that arise in the application of density evolution in this context. We will discuss the individual components of our approach in succession from a system point of view. We begin with the read signal noise variance $N_0 = \sigma^2$, represented by an SNR value. This is the input to the channel quantizer, where the finite discrete channel LLR pmfs, denoted as $\mathrm{p}^{(ch)}$, are computed using a split-ratio quantization method, to be discussed in more detail below. Then we use the *discretized density evolution* (DDE) method introduced in [20] as a proxy for bit error rate (BER), followed by the implementation of gradient descent to gradually adjust the read positions. This process is repeated until it reaches the near-optimal positions, i.e., when the gradient computed is close to zero, indicating that a local minimum has been reached. Towards the end of this chapter, we will provide our simulation parameters as well as results and discussions.

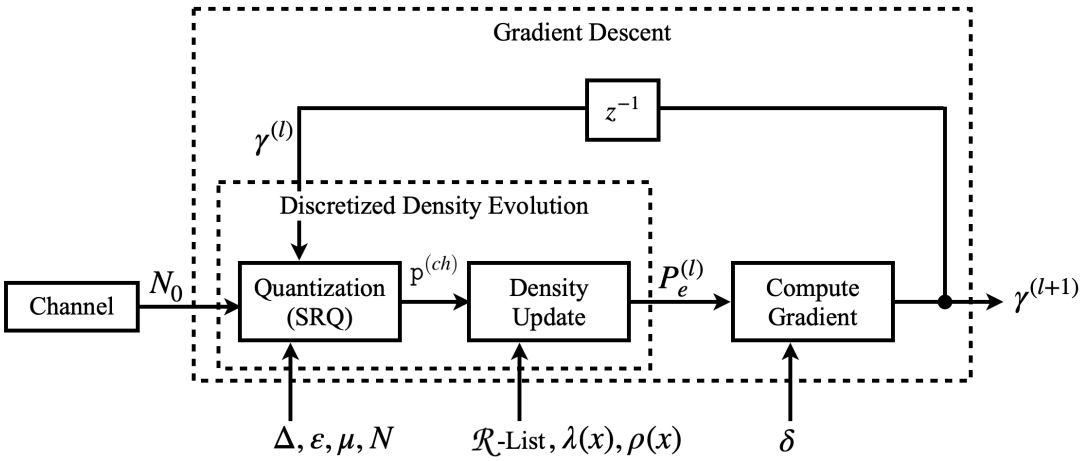


Figure 5.1: Block diagram of read threshold optimization using density evolution.

5.1 Discretized Density Evolution

In [20], an improved implementation of density evolution is developed called the *discretized density evolution* (DDE) which is able to model the exact behavior of discretized sum-product decoding on quantized AWGN channels. The input to the DDE are the channel LLR pmfs which are dictated by both SNR (i.e., noise variance σ^2) and the position of read thresholds γ. We use the MLC NAND flash scenario, 2 bits (MSB and LSB) per symbol, while using the symmetric channel labeling (SCL) as discussed in Section 3.3.2 (see Figure 5.2). The LLRs are computed as

$$\ell_{\text{MSB}}^{(\text{ch})} = \ln\left(\frac{\int_{R_i}(f_{01}(\nu)+f_{00}(\nu))d\nu}{\int_{R_i}(f_{10}(\nu)+f_{11}(\nu))d\nu}\right)$$

$$\ell_{\text{LSB}}^{(\text{ch})} = \ln\left(\frac{\int_{R_i}(f_{10}(\nu)+f_{00}(\nu))d\nu}{\int_{R_i}(f_{11}(\nu)+f_{01}(\nu))d\nu}\right),$$

represent the messages being passed along the edges of the graph.

Note that the LSB channel and MSB channel are entirely distinct and and hence should be analyzed separately. Howver, we will mainly deal with the LSB channel since it is shown in Figure 3.11 that the LSB channel is more degraded than MSB, which makes it the bottleneck in improving the BER.

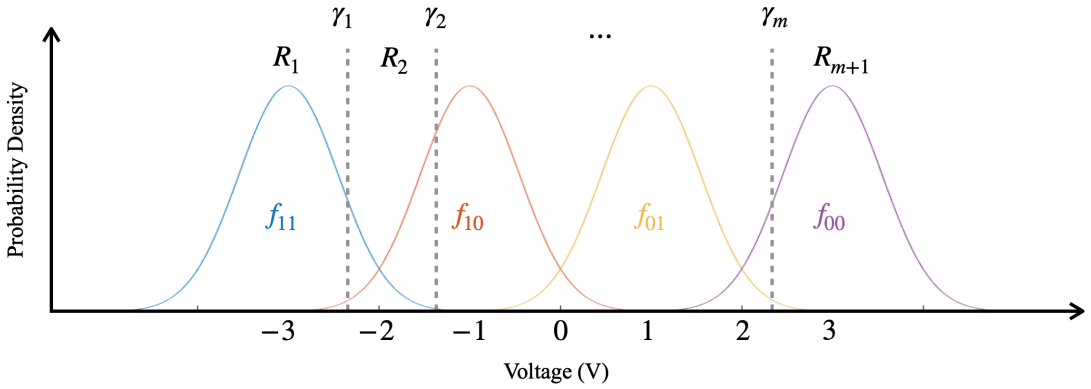


Figure 5.2: The MLC-SCL channel probability distributions with m read thresholds.

5.1.1 Split Ratio Quantization

The first step is quantization. As we know, the core of density evolution revolves around keeping track of the LLR densities at the CNs and VNs. However for hardware implementations and computer simulations, we could never store continuous and unbounded data without some modifications. Thus the quantization is used regularly to convert continuous LLR pdfs into discrete and finite LLR pmfs. For a message w, the standard quantization generates the quantized message $Q(w) = \hat{w}$, defined as:

$$\hat{w} \equiv Q(w) = \begin{cases} \left\lfloor \frac{w}{\Delta} + \frac{1}{2} \right\rfloor \cdot \Delta, & \text{if } w \geq \frac{\Delta}{2} \\ \left\lceil \frac{w}{\Delta} - \frac{1}{2} \right\rceil \cdot \Delta, & \text{if } w \leq \frac{\Delta}{2} \\ 0, & \text{otherwise} \end{cases} \tag{5.1}$$

where Δ is the quantization interval. The pmf value $P(w)$ associated with w is then assigned to $Q(w)$. However, in our DDE implementation, we observed fluctuations in the DDE output due to the use of this quantization. Therefore, we introduced a modified quantization method, which we call split ratio quantization (SRQ), whereby the pmf value associated to w is split between adjacent quantization points in inverse proportion to the distance of w to those points. We refer to Figure 5.3, where original data point is at position w with value $P(w)$. The quantized pmf

under SRQ has two components at positions $Q_l(w)$ and $Q_r(w)$ with values $P_l(w) = \frac{b}{(a+b)}P(w)$ and $P_r(w) = \frac{a}{(a+b)}P(w)$, respectively.

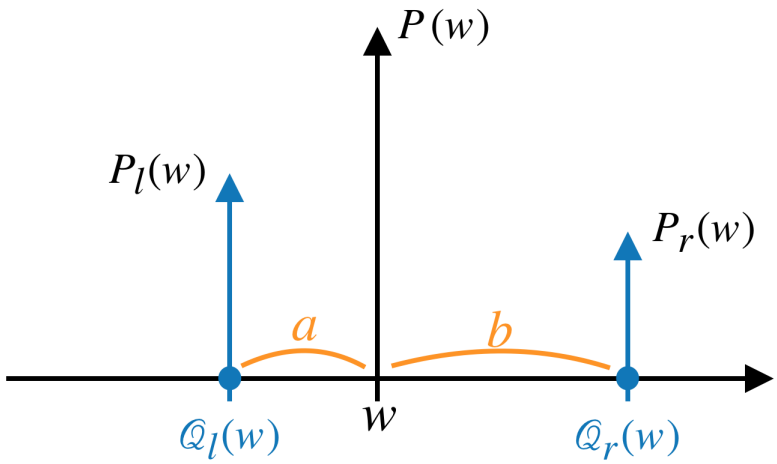
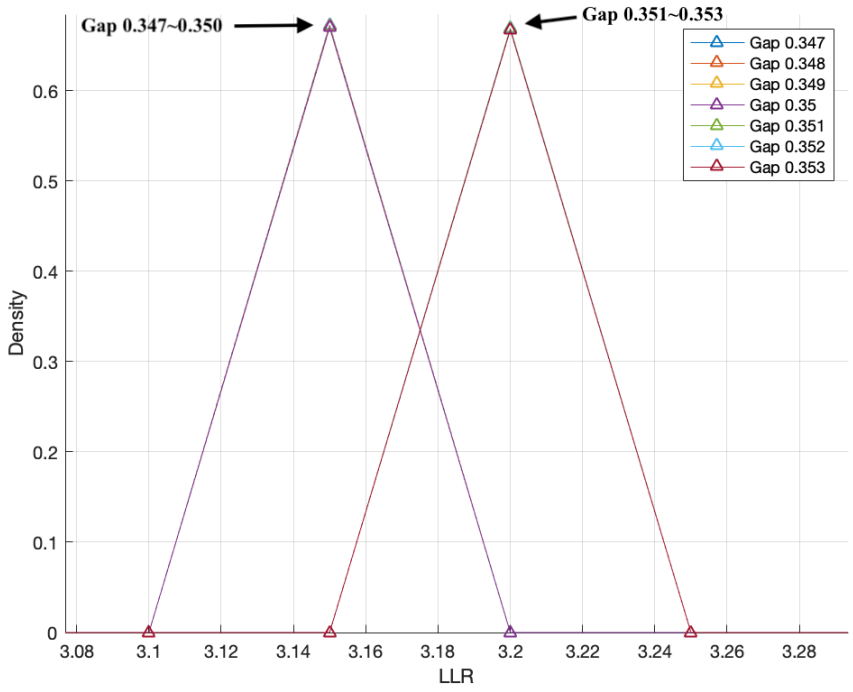


Figure 5.3: Split ratio quantization (SRQ).

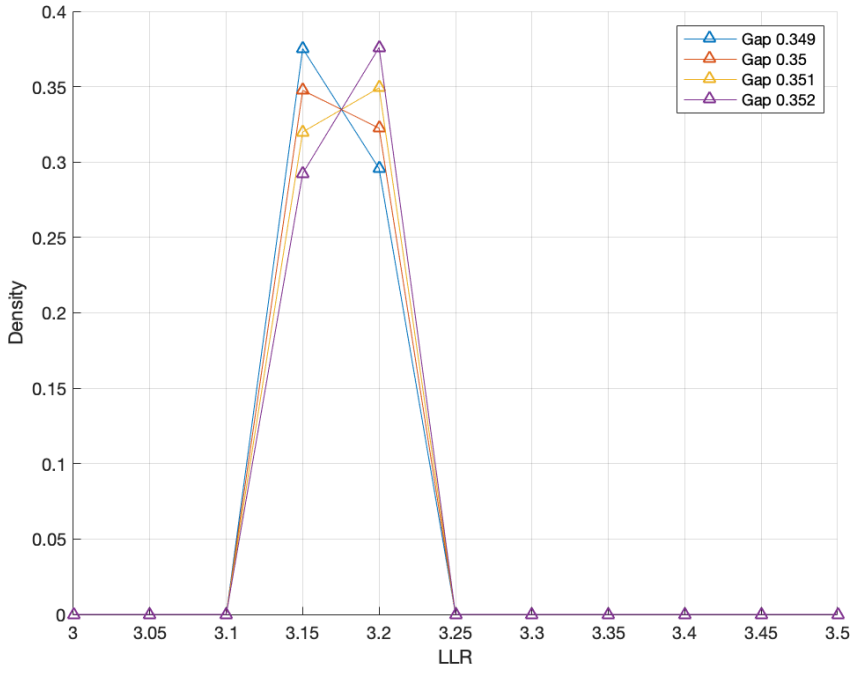
The effect of this method can be observed in Figure 5.4 where we show the LSB channel pmfs before and after applying SRQ. In Figure 5.4a, the LLR pmfs for gaps = 0.347 to 0.350 have almost identical non-zero value at LLR $= 3.15$ while the LLR pmfs for gap = 0.351 to 0.353 have almost identical non-zero value at the next quantization point LLR $= 3.20$. This sharp transition between the gap $= 0.350$ and the gap $= 0.351$ is the source of the observed fluctuations. After applying SRQ, as shown in Figure 5.4b, the transition is smoothed out and fluctuations are significantly reduced. This is important because later on, when we implement gradient descent on top of this, even the slightest fluctuations will result in the creation of local minima traps and prevent us from converging to the global minimum we want.

5.1.2 Finite LLR-pmf Operations

With the quantization method established for the discretized LLR pmfs, we now need to address the issue of representing the pmfs using only a finite number of values. We follow the recommendation of [10] where we only store the LLR range of $\Delta[-N,N]_{\mathbb{Z}}$. We use `p` to



(a) Before SRQ.



(b) After SRQ.

Figure 5.4: The effect of SRQ is to smooth out the transition of LSB channel LLR densities, where $\Delta = 0.05$.

denote this pmf of length $2N+1$. If we convolve two such pmfs, $\mathtt{p}_a$ and $\mathtt{p}_b$, the size of these pmfs are usually really large, therefore the use of an efficient convolution method is a must. We use the FFT-based convolution where the output pmf $\mathtt{p}_c = \mathtt{p}_a \otimes \mathtt{p}_b = \text{IFFT}\big\{\text{FFT}\{\mathtt{p}_a\}\text{FFT}\{\mathtt{p}_b\}\big\}$ of length $4N+1$ supported on $\Delta[-2N, 2N]_{\mathbb{Z}}$. This pmf $\mathtt{p}_c$ can be divided into three parts: $\Delta[-2N, -(N+1)]_{\mathbb{Z}}$, $\Delta[-N, N]_{\mathbb{Z}}$ and $\Delta[N+1, 2N]_{\mathbb{Z}}$. The first part ($\Delta[-2N, -(N+1)]_{\mathbb{Z}}$) can be neglected if the pmfs are symmetric (in the sense of [10]) and if $(N+1)\Delta$ is sufficiently large, which are both satisfied in our case. It is recommended that the probability mass on the last part be stored either at ΔN or separately; we choose the first approach. The resulting convolution operation on finite pmfs produces an output with the same length as the inputs.

5.1.3 Discretized Density Update

The core of DDE [20] is updating the discretized LLR densities, i.e., the finite LLR pmfs discussed previously. In Chapter 4, Section 4.4.1 we discussed the VN update rule under BP decoding. Those rules can be converted to the discrete version:

$$\mathtt{p}_{\mathtt{v}} = \mathtt{p}^{(ch)} \otimes \left(\mathtt{p}_{\mathtt{c}}^{\otimes (d_v - 1)} \right) \tag{5.2}$$

where $\mathtt{p}_{\mathtt{v}}$ and $\mathtt{p}_{\mathtt{c}}$ are the pmf of VN and CN respectively.

As for the CN update equation, we recall the definition of $\mathcal{R}$-operator from [20] for two quantized messages $\hat{w_1}$ and $\hat{w_2}$:

$$\mathcal{R}(\hat{w_1}, \hat{w_2}) = Q\left[2\tanh^{-1}\left(\tanh\frac{\hat{w_1}}{2} \tanh\frac{\hat{w_2}}{2} \right) \right]$$

Then if $\hat{w_3} = \mathcal{R}(\hat{w_1}, \hat{w_2})$, then the pmf associated with this message is given by:

$$\mathtt{p}_{\hat{w_3}}[k] = \sum_{(i,j):k\Delta = \mathcal{R}(i\Delta, j\Delta)} \mathtt{p}_{\hat{w_1}}[i]\mathtt{p}_{\hat{w_2}}[j] \tag{5.3}$$

which we further denote as $\mathtt{p}_{\hat{w_3}} = \mathcal{R}\big(\mathtt{p}_{\hat{w_1}}, \mathtt{p}_{\hat{w_2}}\big)$. For a CN with degree d_c, we have $\mathtt{p_c} = \mathcal{R}\big(\mathtt{p_v}, \mathcal{R}\big(\mathtt{p_v}, \cdots, \mathcal{R}\big(\mathtt{p_v}, \mathtt{p_v}\big), \cdots\big)\big) \equiv \mathcal{R}^{(d_c-1)}\mathtt{p_v}$ since all $\mathtt{p}_{\mathtt{v}_i}$'s are i.i.d for $1 \leq i < d_c$. By defining:

$$\lambda(\mathtt{p}) \triangleq \sum_{i \geq 2} \lambda_i [\mathtt{p}^{\otimes(i-1)}]$$

$$\rho(\mathtt{p}) \triangleq \sum_{j \geq 2} \rho_j [\mathcal{R}^{(j-1)}\mathtt{p}]$$

we can describe the DDE update rule as

$$\mathtt{p}_{\mathtt{c}}^{(l+1)} = \rho\big[\mathtt{p}^{(ch)} \otimes \lambda(\mathtt{p}_{\mathtt{c}}^{(l)})\big] \tag{5.4}$$

where $\mathtt{p}_{\mathtt{c}}^{(0)}$ is a unit impulse at LLR = 0.

The corresponding probability of error at iteration l, is given by:

$$P_e^{(l)} = \sum_{k=-N}^{0} \mathtt{p}_{\mathtt{v}}^{(l)}[k] \tag{5.5}$$

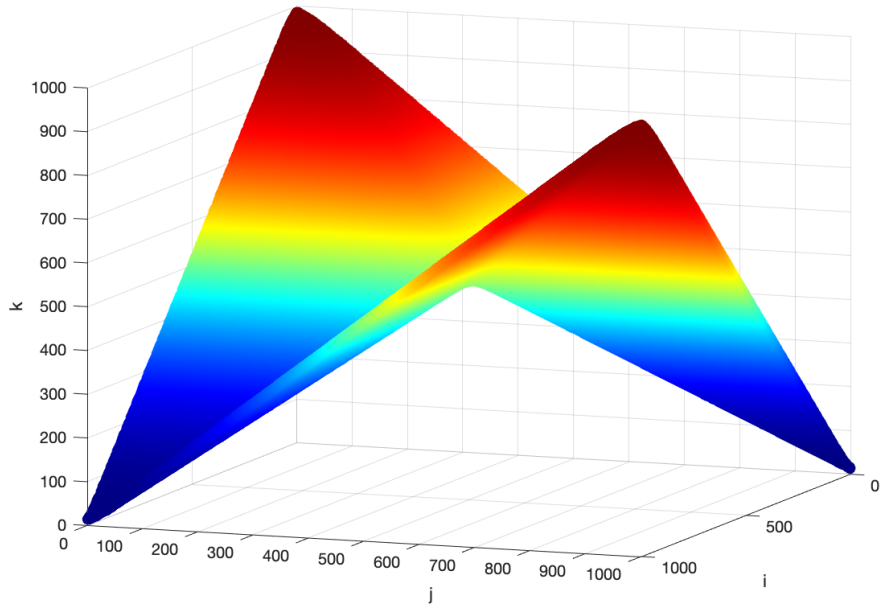
5.1.4 Pre-computing R-List

In equation 5.3 we defined the $\mathcal{R}$-operation [20] produces a new pmf from two input pmfs. This is a key computational element of DDE. However, computing it on the go requires $O(n^3)$ computational complexity since we have to run a 3-layered for-loop of indices k, i and j, where each index runs over the interval $\in [0, 2N+1]$. This can be done in a more efficient manner by pre-computing all the 3-tuple indices that satisfy

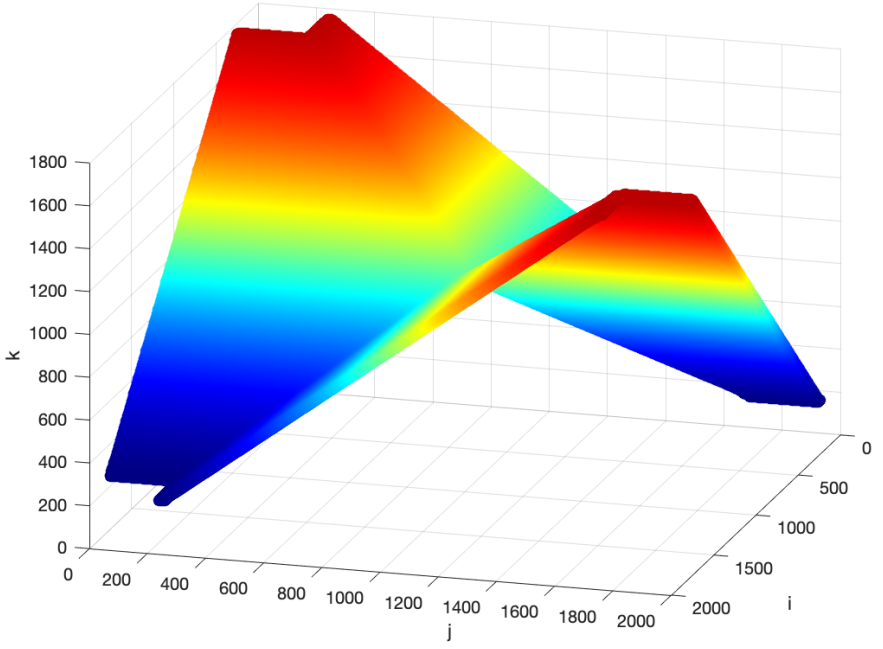
$$(i, j) : k\Delta = \mathcal{R}(i\Delta, j\Delta) \tag{5.6}$$

and store them in a list, which we refer to as the $\mathcal{R}$-List. Figure 5.5 shows all the 3-tuple indices that satisfy equation (5.6). Therefore, in exchange for a little bit of additional memory, we can

reduce the complexity by an entire dimension and significantly reduce computational cost.



(a) $\Delta N = 25$.



(b) $\Delta N = 50$.

Figure 5.5: Illustration of the 3-tuple indices that satisfy equation (5.6) for $\Delta = 0.05$ and different values of ΔN.

5.2 Read Threshold Optimization

Now that the DDE process has been fully explained, we can define a function $g(\cdot)$, which takes $\mu, \Delta N, \lambda(x), \rho(x), \mu, N_0$, and ℓ as input parameters, uses the computational tricks described earlier, and outputs a the BER estimated after ℓ iterations of massage passing. For simplicity, we denote this function by $g(\gamma)$ since the rest of the input parameters remain constant throughout an entire optimization lifetime. We use the gradient descent again for our optimization algorithm and continuously update the read threshold positions with

$$\gamma^{(t+1)} = \gamma^{(t)} + \delta \cdot \nabla g\left(\gamma^{(t)}\right), \tag{5.7}$$

as we did for MMI-based optimization, where δ is the step size. We use t here to indicate gradient descent iterations, not to be confused with l, the message passing iterations. The gradient is calculated as before, i.e., by picking a small offset value ε and calculating the slope of P_e when each element in γ is offset by ε.

5.3 Simulation Results and Discussion

In this section we apply our DDE-based threshold optimization method to Code A and Code B. We first present results for Code A and then make several observations pertaining to the choice of parity-check matrix, the advantage of DDE over MMI, the initialization of the read thresholds, and the BP performance with more iterations. Finally, we present DDE results for Code B and compare them to the MMI results.

5.3.1 Simulation Results for Code A

The simulation parameters are shown in Table 5.1 if not otherwise specified, and the overall process is given in Algorithm 6. We use Code A (introduced in Section 4.2.1) of rate 1/3

in most of our simulation studies of the MLC-SCL LSB channel because it is a regular code with a sparse degree distribution, thereby requiring fewer computations and less run-time. We choose to work with the LSB channel because as shown in Figures 3.10 and 3.11, the LSB channel is more degraded. However, our simulations can easily be adapted for the MSB channel by simply changing the channel LLR densities provided as input to the DDE. The general generator matrix encoding method is used for Code A and the double-diagonal encoding is used for Code B while we use the standard BP decoding for both cases. Figure 5.7 depicts the evolution of error probability and read thresholds in gradient descent iterations.

Table 5.1: Parameters used in our simulations if not otherwise specified.

Parameter	Value
μ	[-3, -1, 1, 3](V)
Δ	0.05
ε	1×10^{-3}
δ	0.1
N	800
ζ	1×10^{-14}
R (# of BP/DDE iterations)	2
Channel	MLC-SCL LSB
LDPC Code	Code A (Section: 4.2.1)

5.3.2 Comparison of MMI and DDE Based Optimization

We observed that DDE optimization approach does indeed perform better than the MMI approach in the mid-SNR regions (as shown in Figure 5.8 and Table 5.2). The final read threshold positions given by DE eventually converge back to the thresholds given by MMI in high-SNR regions. The optimal threshold positions given by MMI and DE are provided in Appendix B and C, respectively.

Algorithm 6: Gradient Descent to Minimize Probability of Error given by Density Evolution

input : The vector of mean stored voltages μ given by equation (2.1), the variance of AWGN N_0 and the number of read thresholds m.

output : The optimal read threshold positions γ that attain the minimum probability of error given by density evolution.

`/* Initialization */`

Iteration number $l = 0$;

Initialize $\gamma^{(0)}$ to a certain starting position ;

Initialize gain $\partial g = 10$ (a very large number) ;

$P_e^{(l)} = \mathrm{g}(\,\gamma^{(l)}\,)$; `// g(·) is the function that produces the` P_e `as discussed in Section 5.2`

`/* Main Loop */`

while $\partial g > \zeta$ **do**

  `/*` ε_i `denotes a vector of length m with` ε `in the` i^{th} `entry and zero elsewhere */`

  for $i = 1 \rightarrow m$ **do**

    $\nabla g(\gamma^{(l)})_i = \frac{g(\gamma^{(l)}+\varepsilon_i)}{\varepsilon}$;

  end

  $\gamma^{(l+1)} = \gamma^{(l)} + \delta \cdot \nabla g(\gamma^{(l)})$;

  Constrain γ to be symmetric ;

  $P_e^{(l+1)} = \mathrm{g}(\,\gamma^{(l+1)}\,)$;

  $\partial g = P_e^{(l+1)} - P_e^{(l)}$;

  $l++$;

end

Result: Optimal read threshold positions are $\gamma^{(l)}$

5.3.3 Initialization of Read Thresholds for LSB

When implementing the DDE-GD approach, it is possible for GD to get stuck in a local minimum. Referring to Figure 5.9, we explore a two-dimensional (2D) exhaustive search for $m = 5$ reads. This is possible for all $m \leq 5$ cases because of the symmetric constraint imposed on the read thresholds γ. In this case, the search becomes a function of two variables γ_4 and γ_5. We can see from Figure 5.9a that there are two local minima, around $(\gamma_4, \gamma_5) = (0.25, 2)$ and $(1.85, 2.25)$. This is due to the labeling for the LSB. Therefore we propose to initialize the read thresholds γ to MMI positions and then refine them using DDE-GD. By doing so, we can

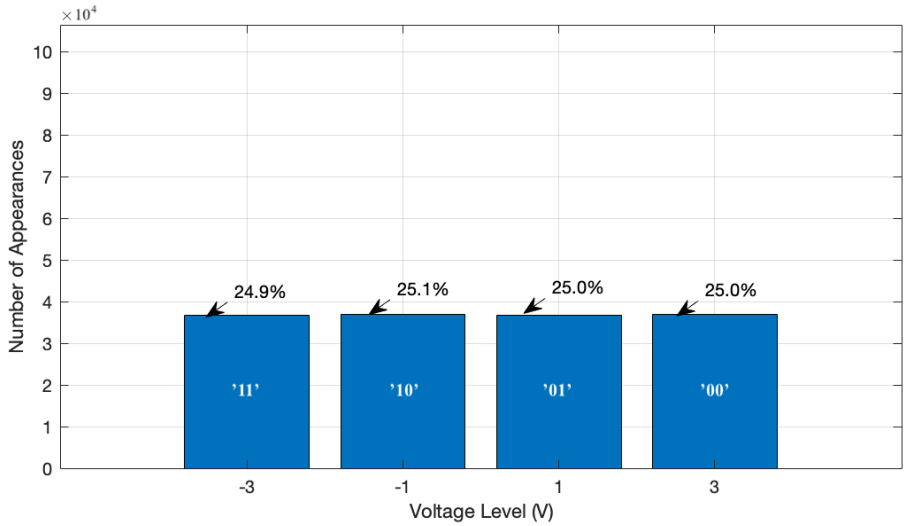
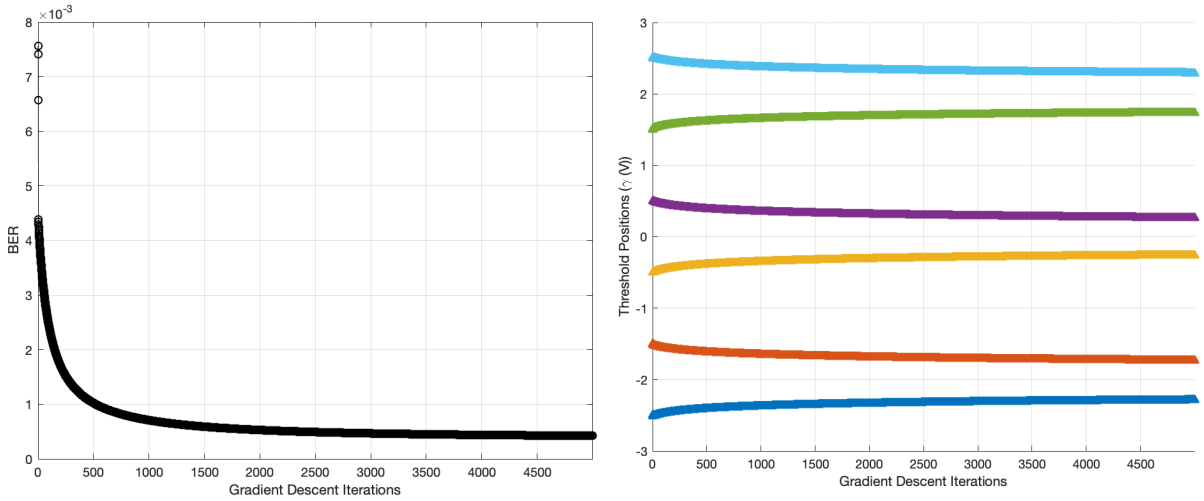


Figure 5.6: Histogram of symbol occurrences in the MLC-SCL channel (LSB and MSB channels independently LDPC-coded).



(a) Evolution of error probability.

(b) Evolution of read threshold positions.

Figure 5.7: Illustration of the evolution of optimizing read threshold positions using DDE criteria for MLC-SCL LSB channel with Code A, SNR = 13(dB), $m = 6$, $R = 2$. The threshold positions are initialized to equal spacing.

guarantee that we can achieve better BER than MMI. In our simulations, we do always end up at the global minimum using this 2-step method, however, there are no mathematical proofs that guarantees this to always be the case. The cost of being trapped at the wrong local minimum is very high. We show in Figure 5.10 and in Table 5.3 the performance comparison between the

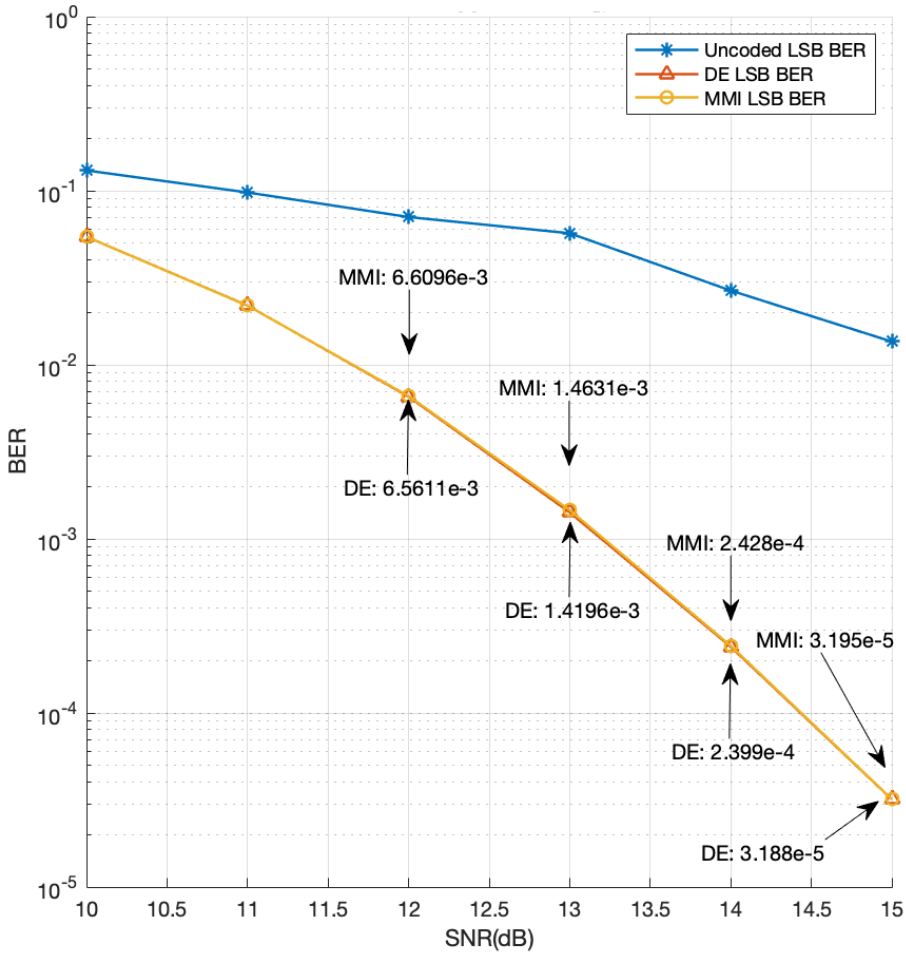


Figure 5.8: BER performance comparison between MMI thresholds and DDE thresholds for MLC-SCL LSB channel with $m = 6, R = 2$.

global minimum the secondary local minimum.

In MLC-SCL MSB channel, we don't have to worry about this problem, as shown in Figure 5.9b. To intuitively explain why the MSB channel have only one local minimum where LSB has two is because the MSB for MLC-SCL are in the sequence of $[1,1,0,0]$. The placement of thresholds are obvious, i.e., around the center. In comparison, the LSB sequence in MLC-SCL is $[1,0,1,0]$. When we have, for example, $m = 5$ reads, the placement of the first 3 reads are obvious, i.e., at around $-2,0$ and 2. However the placement of the last two reads are not so straight forward. One can place them wither at around 0 or around ± 2. This is our explanation towards the phenomenon of having two local minima for LSB and not MSB

Table 5.2: BER performance comparison with BP decoding between MMI and DE given optimal threshold positions. Results are shown for $m = 6, R = 2$ using (3,4)-QCLDPC code (Code A).

	Optimization Criteria	SNR(dB)			
		12	13	14	15
BER (BP)	MMI	6.6096×10^{-3}	1.4631×10^{-3}	2.428×10^{-4}	3.195×10^{-5}
	DE	6.5611×10^{-3}	1.4196×10^{-3}	2.399×10^{-4}	3.188×10^{-5}

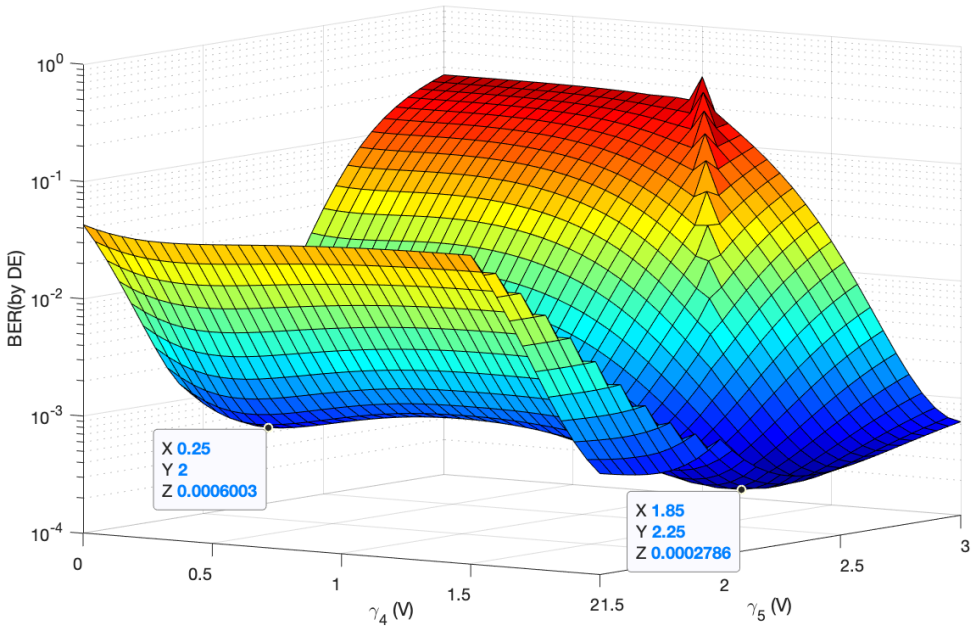
Table 5.3: Comparing the DE optimal thresholds for Code A, with $m = 5$ MLC-SCL LSB channel, when gradient descent is initialized to different values. We see that there are two local minima with a significant error probability difference.

# of Reads	$m = 5$						
Initialization	MMI Optimal γ				Secondary Local Minimum		
	SNR	γ_4	γ_5	BER(BP)	γ_4	γ_5	BER(BP)
	10	1.8640	2.4302	5.0447×10^{-2}	0.2804	2.0612	6.4084×10^{-2}
	11	1.8596	2.3924	1.9865×10^{-2}	0.2612	2.0453	2.7807×10^{-2}
	12	1.8740	2.3614	5.5126×10^{-3}	0.2563	2.0187	8.9167×10^{-3}
	13	1.8481	2.2848	1.1762×10^{-3}	0.2799	2.0107	2.3613×10^{-3}
	14	1.8572	2.2408	2.3101×10^{-4}	0.2438	2.0144	5.6512×10^{-4}
	15	1.8777	2.1901	5.1858×10^{-5}	0.2371	2.0051	1.1230×10^{-4}

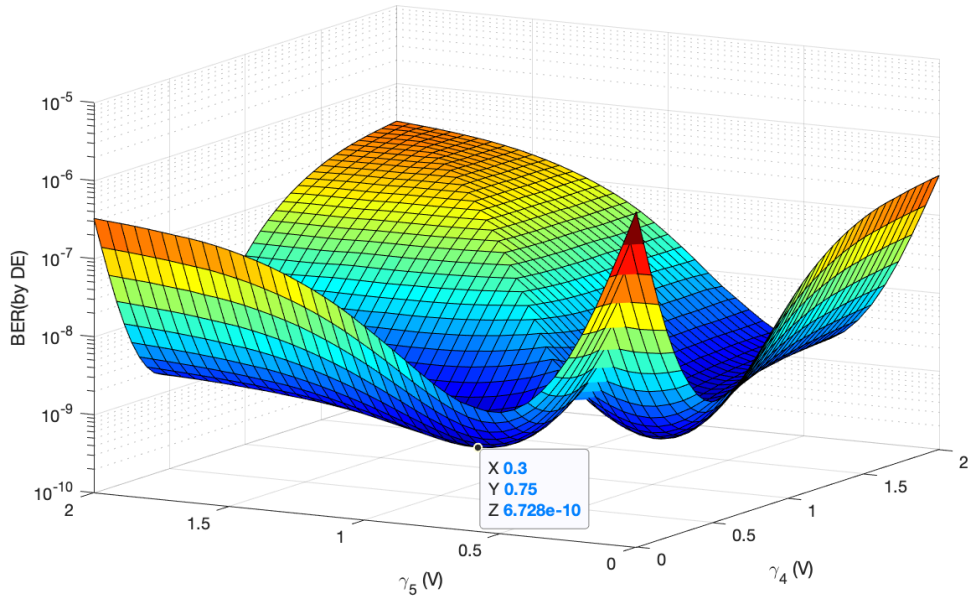
channel in MLC-SCL. Note that the 3D plot of MSB is symmetric about $\gamma_4 = \gamma_5$ because reversing the order of γ_4 and γ_5 does not change the regions the thresholds divide into, e.g., $[-2,-1,0,1,2]$ and $[-1,-2,0,2,1]$ are equivalent since they both split the voltage regions into the set $R_i \in \{[-\infty,-2],[-2,-1],[-1,0],[0,1],[1,2],[2,\infty]\}$.

5.3.4 Higher Number of Iterations in BP Decoding

Using a higher number of BP decoding iterations may also yield a different relative performance using the DDE-optimized and MMI-optimized thresholds. To study this, we use the set of read threshold positions given in Table C.1 for $R = 2$ and $R = 20$ rounds of BP decoding iterations under the MLC-SCL LSB channel. As shown in Figure 5.11, there is not much difference in the sense that the DE curve is still close to the MMI curve. Specific simulation numbers at SNR = 13dB are given in Table 5.4.



(a) LSB channel.



(b) MSB channel.

Figure 5.9: A 2D DE exhaustive search for $m = 5$ reads, SNR = 13dB MLC-SCL, and $\boldsymbol{\gamma} = [\gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5]$ where $\gamma_1 = -\gamma_5$, $\gamma_2 = -\gamma_4$ and $\gamma_3 = 0$.

5.3.5 Simulation Results for Code B

Finally, we implemented DDE-based read threshold optimization for Code B and evaluated the resulting performance under BP decoding. We find that the optimal threshold positions differ

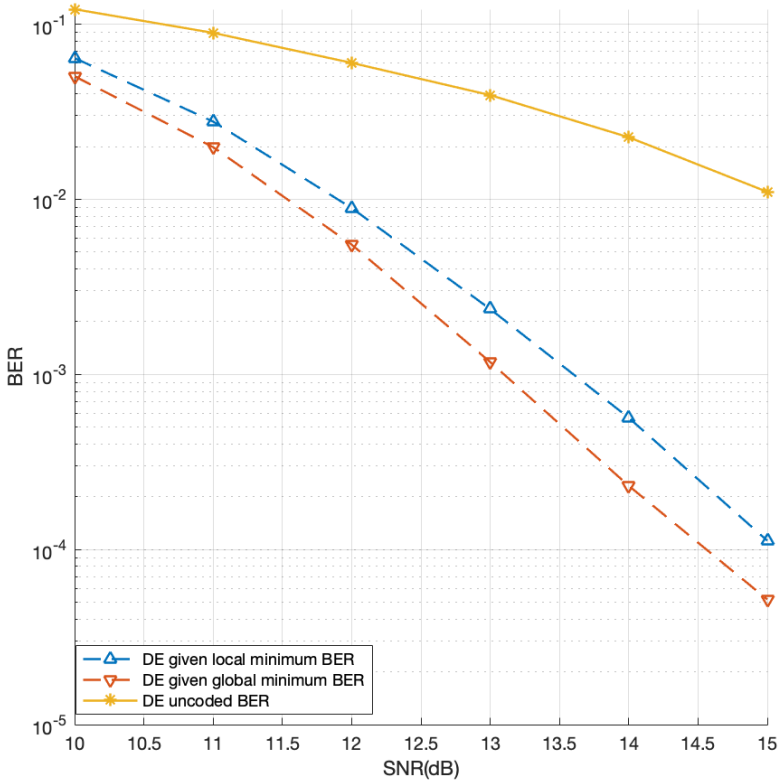
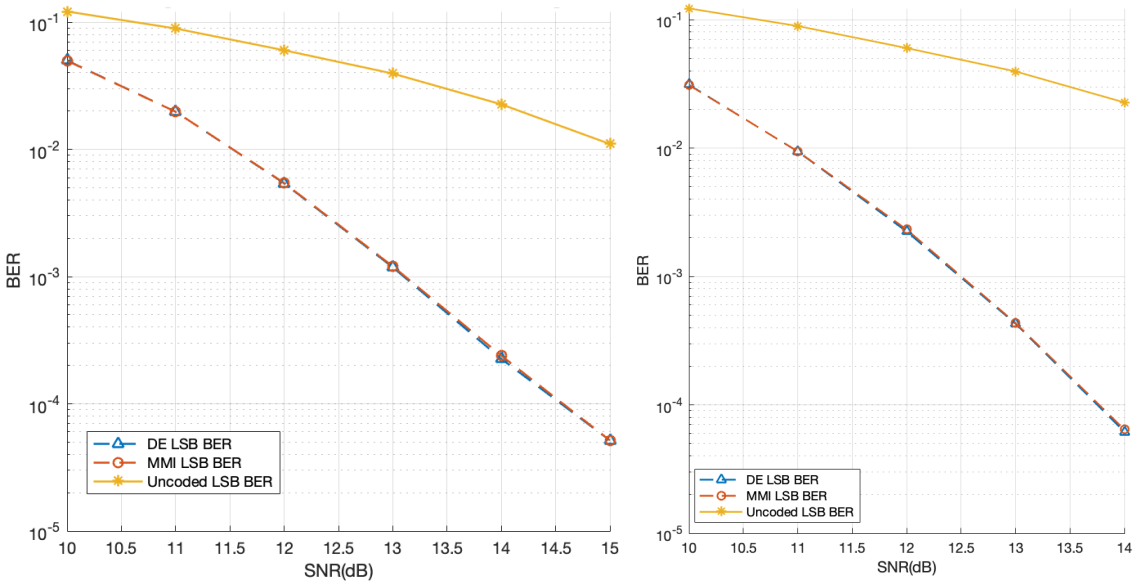


Figure 5.10: BER performance comparison between read thresholds at local minimum and global minimum given by density evolution.

from those found for Code A, as well as those found by MMI-based optimization. We give an example in Table 5.5, which shows the DDE thresholds for Code A and Code B corresponding to SNR = 14dB, $m = 5$, $R = 2$, the MMI thresholds, and the BP decoding performance of Code B using the different sets of thresholds. We conclude that, in contrast to the MMI approach, the DDE approach is indeed code-dependent. It provides read threshold positions that take into account the structure of the Tanner graph of the LDPC code and performs better than MMI optimization approach, which is independent of any code properties.

5.4 Summary

In this thesis, we compared two approaches towards optimizing the read thresholds for NAND flash memories, the MMI approach (Chapter 3) and the DE approach (Chapter 5). For the



(a) $R = 2$. (b) $R = 20$.

Figure 5.11: BER performance comparison between different number of iterations in BP decoding for $m = 5$ reads.

Table 5.4: BER performance comparison given by BP decoding between MMI and DE with different BP decoding iterations. Results are shown for SNR = 13dB, $m = 5, R = 2, 20$ using (3,4)-QCLDPC code (Code A).

Curve	R	
	2	20
MMI BER	1.203×10^{-3}	4.344×10^{-4}
DE BER	1.182×10^{-3}	4.286×10^{-4}

Table 5.5: BER performance comparison of Code B under BP decoding for MLC-SCL LSB channel with read thresholds optimized for SNR = 14dB, $m = 5$, $R = 2$ and various criteria .

	Optimization Criteria		
	DE Code A Degree Distribution	DE Code B Degree Distribution	MMI
γ	-2.2408 -1.8572 0 1.8572 2.2408	-2.2366 -1.8684 0 1.8684 2.2366	-2.2284 -1.8679 0 1.8679 2.2284
BER (Code B BP Decoding)	1.860308×10^{-3}	1.858656×10^{-3}	1.860759×10^{-3}

former approach, proposed in [18], we extended the application to a higher number of reads and provided results for an alternative, symmetric binary labeling of the MLC cell levels. Detailed results are presented in Appendix A and Appendix B.

For the latter approach, originally proposed in [19], we provide a more thorough exploration of several aspects relevant to accurate and efficient implementation of DE-based optimization, including LLR quantization, finite pmf convolution, and channel symmetry. We also investigate code-related issues such as dependence on degree distributions, Tanner graph selection, and existence of secondary local minima. Detailed results are presented in Appendix C. We conclude that the DE approach yields better performance than the MMI approach. However, for the symmetric MLC LSB channel, better results are obtained by using a two-step optimization procedure: first, use the MMI approach for coarse optimization of the read threshold positions and, then, use the DE approach for fine optimization. This helps to avoid the problem of falling into a secondary local minimum that can arise when using DE alone.

Chapter 5, in part, contains materials from [12]

- Y. Yeh, A. Fazeli, and P.H. Siegel, "Optimization of Read Thresholds in MLC NAND Memory for LDPC Codes," in *Proc. 11th Annu. Non-Volatile Memories Workshop (NVMW)*, La Jolla, CA, USA, Mar. 2020. [Online]. Available: http://nvmw.ucsd.edu/program/

The thesis author was the primary investigator and author of this paper.

Appendix A

MMI Optimal Read Thresholds

In this section, we provide the result of our simulations when optimizing for maximum mutual information using gradient descent algorithm (as described in Chapter 3). We include the optimal read voltage positions for $m = \{2, 3, \cdots, 8, 10\}$ with SNR $= [10, 15]$(dB) from Table A.1 to A.8. Also, we show the optimal read voltage positions for SNR $= \{10, 13, 15\}$(dB) with $m = [2, 30]$ in Tables A.9, A.10 and A.11.

Table A.1: Read threshold positions $\bar{\gamma}$ that attain the maximum mutual information for $m = 2$ at the given SNR value.

SNR(dB)	m	$\bar{\gamma}$	MMI
10	2	-1.7306 1.7306	1.1289
10.2	2	-1.7545 1.7545	1.1432
10.4	2	-1.7763 1.7763	1.1577
10.6	2	-1.7962 1.7962	1.1724
10.8	2	-1.8144 1.8144	1.1872
11	2	-1.8309 1.8309	1.2022
11.2	2	-1.8459 1.8459	1.2172
11.4	2	-1.8595 1.8595	1.2323
11.6	2	-1.8719 1.8719	1.2473
11.8	2	-1.8831 1.8831	1.2621
12	2	-1.8934 1.8934	1.2769
12.2	2	-1.9027 1.9027	1.2914
12.4	2	-1.9111 1.9111	1.3056
12.6	2	-1.9189 1.9189	1.3196
12.8	2	-1.9259 1.9259	1.3332
13	2	-1.9323 1.9323	1.3463
13.2	2	-1.9381 1.9381	1.359
13.4	2	-1.9434 1.9434	1.3712
13.6	2	-1.9483 1.9483	1.3829
13.8	2	-1.9527 1.9527	1.3941
14	2	-1.9567 1.9567	1.4046
14.2	2	-1.9604 1.9604	1.4145
14.4	2	-1.9638 1.9638	1.4239
14.6	2	-1.9669 1.9669	1.4325
14.8	2	-1.9697 1.9697	1.4406
15	2	-1.9722 1.9722	1.448

Table A.2: Read threshold positions $\bar{\gamma}$ that attain the maximum mutual information for $m = 3$ at the given SNR value.

SNR(dB)	m	$\bar{\gamma}$	MMI
10	3	-1.9847 0 1.9847	1.4087
10.2	3	-1.9869 0 1.9869	1.4336
10.4	3	-1.9888 0 1.9888	1.4586
10.6	3	-1.9904 0 1.9904	1.4837
10.8	3	-1.9919 0 1.9919	1.5087
11	3	-1.9931 0 1.9931	1.5337
11.2	3	-1.9942 0 1.9942	1.5585
11.4	3	-1.9951 0 1.9951	1.5831
11.6	3	-1.9959 0 1.9959	1.6074
11.8	3	-1.9966 0 1.9966	1.6314
12	3	-1.9972 0 1.9972	1.655
12.2	3	-1.9977 0 1.9977	1.6781
12.4	3	-1.9981 0 1.9981	1.7006
12.6	3	-1.9985 0 1.9985	1.7226
12.8	3	-1.9988 0 1.9988	1.7439
13	3	-1.999 0 1.999	1.7644
13.2	3	-1.9993 0 1.9993	1.7842
13.4	3	-1.9994 0 1.9994	1.8032
13.6	3	-1.9996 0 1.9996	1.8213
13.8	3	-1.9997 0 1.9997	1.8384
14	3	-1.9997 0 1.9997	1.8547
14.2	3	-1.9999 0 1.9999	1.87
14.4	3	-1.9999 0 1.9999	1.8843
14.6	3	-1.9999 0 1.9999	1.8976
14.8	3	-1.9999 0 1.9999	1.9099
15	3	-2 0 2	1.9212

Table A.3: Read threshold positions $\bar{\gamma}$ that attain the maximum mutual information for $m = 4$ at the given SNR value.

SNR(dB)	m	$\bar{\gamma}$	MMI
10	4	-2.0272 -0.40788 0.40788 2.0272	1.4503
10.2	4	-2.0234 -0.39141 0.39141 2.0234	1.4741
10.4	4	-2.0201 -0.37582 0.37582 2.0201	1.4979
10.6	4	-2.0172 -0.36098 0.36098 2.0172	1.5217
10.8	4	-2.0148 -0.34694 0.34694 2.0148	1.5455
11	4	-2.0126 -0.3336 0.3336 2.0126	1.5692
11.2	4	-2.0108 -0.32103 0.32103 2.0108	1.5927
11.4	4	-2.0092 -0.30908 0.30908 2.0092	1.616
11.6	4	-2.0078 -0.29777 0.29777 2.0078	1.639
11.8	4	-2.0066 -0.28709 0.28709 2.0066	1.6617
12	4	-2.0056 -0.27686 0.27686 2.0056	1.6839
12.2	4	-2.0047 -0.26718 0.26718 2.0047	1.7056
12.4	4	-2.004 -0.25797 0.25797 2.004	1.7268
12.6	4	-2.0033 -0.24914 0.24914 2.0033	1.7474
12.8	4	-2.0028 -0.24071 0.24071 2.0028	1.7673
13	4	-2.0023 -0.2327 0.2327 2.0023	1.7865
13.2	4	-2.0019 -0.22498 0.22498 2.0019	1.8049
13.4	4	-2.0016 -0.21754 0.21754 2.0016	1.8224
13.6	4	-2.0013 -0.21045 0.21045 2.0013	1.8392
13.8	4	-2.0011 -0.20365 0.20365 2.0011	1.855
14	4	-2.0009 -0.19706 0.19706 2.0009	1.8699
14.2	4	-2.0007 -0.19072 0.19072 2.0007	1.8839
14.4	4	-2.0006 -0.18463 0.18463 2.0006	1.8969
14.6	4	-2.0005 -0.17876 0.17876 2.0005	1.909
14.8	4	-2.0004 -0.17307 0.17307 2.0004	1.9202
15	4	-2.0003 -0.16758 0.16758 2.0003	1.9304

Table A.4: Read threshold positions $\bar{\gamma}$ that attain the maximum mutual information for $m = 5$ at the given SNR value.

SNR(dB)	m	$\bar{\gamma}$	MMI
10	5	-2.2908 -1.5111 0 1.5111 2.2908	1.4851
10.2	5	-2.2889 -1.5393 0 1.5393 2.2889	1.5084
10.4	5	-2.2865 -1.5654 0 1.5654 2.2865	1.5317
10.6	5	-2.2835 -1.5892 0 1.5892 2.2835	1.5549
10.8	5	-2.2799 -1.611 0 1.611 2.2799	1.578
11	5	-2.2758 -1.6309 0 1.6309 2.2758	1.601
11.2	5	-2.2713 -1.6491 0 1.6491 2.2713	1.6237
11.4	5	-2.2664 -1.6657 0 1.6657 2.2664	1.6461
11.6	5	-2.2612 -1.681 0 1.681 2.2612	1.6682
11.8	5	-2.2557 -1.6951 0 1.6951 2.2557	1.6899
12	5	-2.2501 -1.7081 0 1.7081 2.2501	1.7111
12.2	5	-2.2443 -1.7201 0 1.7201 2.2443	1.7317
12.4	5	-2.2385 -1.7314 0 1.7314 2.2385	1.7518
12.6	5	-2.2326 -1.7419 0 1.7419 2.2326	1.7712
12.8	5	-2.2267 -1.7517 0 1.7517 2.2267	1.7899
13	5	-2.2207 -1.761 0 1.761 2.2207	1.8078
13.2	5	-2.2149 -1.7697 0 1.7697 2.2149	1.8249
13.4	5	-2.209 -1.7779 0 1.7779 2.209	1.8413
13.6	5	-2.2032 -1.7858 0 1.7858 2.2032	1.8567
13.8	5	-2.1975 -1.7932 0 1.7932 2.1975	1.8713
14	5	-2.1918 -1.8003 0 1.8003 2.1918	1.8849
14.2	5	-2.1863 -1.807 0 1.807 2.1863	1.8977
14.4	5	-2.1808 -1.8135 0 1.8135 2.1808	1.9095
14.6	5	-2.1755 -1.8196 0 1.8196 2.1755	1.9204
14.8	5	-2.1702 -1.8255 0 1.8255 2.1702	1.9305
15	5	-2.1651 -1.8312 0 1.8312 2.1651	1.9396

Table A.5: Read threshold positions $\bar{\gamma}$ that attain the maximum mutual information for $m = 6$ at the given SNR value.

SNR(dB)	m	$\bar{\gamma}$	MMI
10	6	-2.3575 -1.6501 -0.35284 0.35284 1.6501 2.3575	1.5147
10.2	6	-2.3469 -1.6593 -0.34315 0.34315 1.6593 2.3469	1.5382
10.4	6	-2.3366 -1.6684 -0.33358 0.33358 1.6684 2.3366	1.5615
10.6	6	-2.3267 -1.6775 -0.32417 0.32417 1.6775 2.3267	1.5847
10.8	6	-2.3171 -1.6865 -0.31494 0.31494 1.6865 2.3171	1.6076
11	6	-2.3077 -1.6953 -0.3059 0.3059 1.6953 2.3077	1.6302
11.2	6	-2.2987 -1.704 -0.29707 0.29707 1.704 2.2987	1.6525
11.4	6	-2.2899 -1.7125 -0.28844 0.28844 1.7125 2.2899	1.6744
11.6	6	-2.2813 -1.7208 -0.28 0.28 1.7208 2.2813	1.6959
11.8	6	-2.2729 -1.7289 -0.27181 0.27181 1.7289 2.2729	1.7168
12	6	-2.2648 -1.7368 -0.26383 0.26383 1.7368 2.2648	1.7372
12.2	6	-2.2569 -1.7445 -0.25603 0.25603 1.7445 2.2569	1.7569
12.4	6	-2.2492 -1.752 -0.24844 0.24844 1.752 2.2492	1.776
12.6	6	-2.2418 -1.7594 -0.24105 0.24105 1.7594 2.2418	1.7943
12.8	6	-2.2345 -1.7665 -0.23387 0.23387 1.7665 2.2345	1.8119
13	6	-2.2274 -1.7734 -0.22688 0.22688 1.7734 2.2274	1.8287
13.2	6	-2.2205 -1.7802 -0.22009 0.22009 1.7802 2.2205	1.8447
13.4	6	-2.2139 -1.7867 -0.21348 0.21348 1.7867 2.2139	1.8598
13.6	6	-2.2074 -1.7932 -0.20703 0.20703 1.7932 2.2074	1.874
13.8	6	-2.201 -1.7994 -0.20075 0.20075 1.7994 2.201	1.8874
14	6	-2.1949 -1.8055 -0.19466 0.19466 1.8055 2.1949	1.8998
14.2	6	-2.1889 -1.8114 -0.18874 0.18874 1.8114 2.1889	1.9113
14.4	6	-2.1831 -1.8171 -0.18298 0.18298 1.8171 2.1831	1.922
14.6	6	-2.1775 -1.8227 -0.1774 0.1774 1.8227 2.1775	1.9318
14.8	6	-2.1721 -1.8281 -0.17195 0.17195 1.8281 2.1721	1.9407
15	6	-2.1668 -1.8334 -0.16667 0.16667 1.8334 2.1668	1.9488

Table A.6: Read threshold positions $\bar{\gamma}$ that attain the maximum mutual information for $m = 7$ at the given SNR value.

SNR(dB)	m	$\bar{\gamma}$	MMI
10	7	-2.3948 -1.7213 -0.63829 0 0.63829 1.7213 2.3948	1.5272
10.2	7	-2.3788 -1.7205 -0.61408 0 0.61408 1.7205 2.3788	1.5503
10.4	7	-2.3638 -1.7208 -0.59069 0 0.59069 1.7208 2.3638	1.5733
10.6	7	-2.3498 -1.7223 -0.56829 0 0.56829 1.7223 2.3498	1.5961
10.8	7	-2.3366 -1.7246 -0.54693 0 0.54693 1.7246 2.3366	1.6187
11	7	-2.3242 -1.7277 -0.52664 0 0.52664 1.7277 2.3242	1.6409
11.2	7	-2.3126 -1.7315 -0.50747 0 0.50747 1.7315 2.3126	1.6628
11.4	7	-2.3016 -1.7358 -0.48932 0 0.48932 1.7358 2.3016	1.6843
11.6	7	-2.2912 -1.7405 -0.47209 0 0.47209 1.7405 2.2912	1.7054
11.8	7	-2.2813 -1.7456 -0.45581 0 0.45581 1.7456 2.2813	1.7259
12	7	-2.2718 -1.7509 -0.44037 0 0.44037 1.7509 2.2718	1.7458
12.2	7	-2.2628 -1.7564 -0.42568 0 0.42568 1.7564 2.2628	1.7651
12.4	7	-2.2542 -1.7621 -0.4117 0 0.4117 1.7621 2.2542	1.7838
12.6	7	-2.2459 -1.7678 -0.39836 0 0.39836 1.7678 2.2459	1.8017
12.8	7	-2.2379 -1.7736 -0.3856 0 0.3856 1.7736 2.2379	1.8188
13	7	-2.2303 -1.7794 -0.3734 0 0.3734 1.7794 2.2303	1.8352
13.2	7	-2.2229 -1.7852 -0.36169 0 0.36169 1.7852 2.2229	1.8507
13.4	7	-2.2158 -1.791 -0.35046 0 0.35046 1.791 2.2158	1.8654
13.6	7	-2.209 -1.7967 -0.33963 0 0.33963 1.7967 2.209	1.8792
13.8	7	-2.2023 -1.8023 -0.32922 0 0.32922 1.8023 2.2023	1.8921
14	7	-2.1959 -1.8079 -0.31918 0 0.31918 1.8079 2.1959	1.9042
14.2	7	-2.1898 -1.8134 -0.30945 0 0.30945 1.8134 2.1898	1.9153
14.4	7	-2.1838 -1.8188 -0.30011 0 0.30011 1.8188 2.1838	1.9256
14.6	7	-2.178 -1.8241 -0.29109 0 0.29109 1.8241 2.178	1.935
14.8	7	-2.1724 -1.8293 -0.28238 0 0.28238 1.8293 2.1724	1.9436
15	7	-2.167 -1.8344 -0.27393 0 0.27393 1.8344 2.167	1.9513

Table A.7: Read threshold positions $\bar{\gamma}$ that attain the maximum mutual information for $m = 8$ at the given SNR value.

SNR(dB)	m	$\bar{\gamma}$	MMI
10	8	-2.5294 -1.9464 -1.3245 -0.31301 0.31301 1.3245 1.9464 2.5294	1.5382
10.2	8	-2.5169 -1.9517 -1.3506 -0.30892 0.30892 1.3506 1.9517 2.5169	1.5612
10.4	8	-2.505 -1.9572 -1.3772 -0.30444 0.30444 1.3772 1.9572 2.505	1.5841
10.6	8	-2.4934 -1.9627 -1.4034 -0.2995 0.2995 1.4034 1.9627 2.4934	1.6067
10.8	8	-2.482 -1.9677 -1.4285 -0.29413 0.29413 1.4285 1.9677 2.482	1.629
11	8	-2.4707 -1.9723 -1.4524 -0.28839 0.28839 1.4524 1.9723 2.4707	1.651
11.2	8	-2.4594 -1.9764 -1.4748 -0.28234 0.28234 1.4748 1.9764 2.4594	1.6726
11.4	8	-2.4482 -1.9799 -1.4957 -0.27606 0.27606 1.4957 1.9799 2.4482	1.6938
11.6	8	-2.437 -1.9829 -1.5153 -0.26961 0.26961 1.5153 1.9829 2.437	1.7144
11.8	8	-2.4258 -1.9855 -1.5336 -0.26306 0.26306 1.5336 1.9855 2.4258	1.7346
12	8	-2.4147 -1.9877 -1.5507 -0.25645 0.25645 1.5507 1.9877 2.4147	1.7542
12.2	8	-2.4038 -1.9895 -1.5668 -0.24984 0.24984 1.5668 1.9895 2.4038	1.7731
12.4	8	-2.393 -1.9911 -1.582 -0.24324 0.24324 1.582 1.9911 2.393	1.7913
12.6	8	-2.3823 -1.9924 -1.5963 -0.23669 0.23669 1.5963 1.9924 2.3823	1.8089
12.8	8	-2.3718 -1.9935 -1.6098 -0.2302 0.2302 1.6098 1.9935 2.3718	1.8256
13	8	-2.3614 -1.9944 -1.6227 -0.2238 0.2238 1.6227 1.9944 2.3614	1.8415
13.2	8	-2.3513 -1.9951 -1.6349 -0.21749 0.21749 1.6349 1.9951 2.3513	1.8567
13.4	8	-2.3414 -1.9958 -1.6467 -0.21129 0.21129 1.6467 1.9958 2.3414	1.8709
13.6	8	-2.3317 -1.9963 -1.6579 -0.2052 0.2052 1.6579 1.9963 2.3317	1.8843
13.8	8	-2.3222 -1.9967 -1.6686 -0.19923 0.19923 1.6686 1.9967 2.3222	1.8968
14	8	-2.3129 -1.9971 -1.6789 -0.19338 0.19338 1.6789 1.9971 2.3129	1.9085
14.2	8	-2.3038 -1.9974 -1.6888 -0.18766 0.18766 1.6888 1.9974 2.3038	1.9193
14.4	8	-2.2949 -1.9976 -1.6983 -0.18206 0.18206 1.6983 1.9976 2.2949	1.9292
14.6	8	-2.2863 -1.9978 -1.7075 -0.1766 0.1766 1.7075 1.9978 2.2863	1.9382
14.8	8	-2.2778 -1.9979 -1.7163 -0.17127 0.17127 1.7163 1.9979 2.2778	1.9465
15	8	-2.2696 -1.998 -1.7248 -0.16607 0.16607 1.7248 1.998 2.2696	1.9539

Table A.8: Read threshold positions $\bar{\gamma}$ that attain the maximum mutual information for $m = 10$ at the given SNR value.

SNR(dB)	m	$\bar{\gamma}$	MMI
10	10	-2.6185 -2.0767 -1.5785 -0.79421 -0.22671 0.22671 0.79421 1.5785 2.0767 2.6185	1.5524
10.2	10	-2.5949 -2.0656 -1.5748 -0.76739 -0.21867 0.21867 0.76739 1.5748 2.0656 2.5949	1.5754
10.4	10	-2.5725 -2.0556 -1.5725 -0.74003 -0.21067 0.21067 0.74003 1.5725 2.0556 2.5725	1.5981
10.6	10	-2.5514 -2.0468 -1.5717 -0.71287 -0.20284 0.20284 0.71287 1.5717 2.0468 2.5514	1.6206
10.8	10	-2.5315 -2.0393 -1.5725 -0.68651 -0.19528 0.19528 0.68651 1.5725 2.0393 2.5315	1.6427
11	10	-2.5127 -2.0328 -1.5749 -0.66126 -0.18804 0.18804 0.66126 1.5749 2.0328 2.5127	1.6644
11.2	10	-2.4951 -2.0274 -1.5787 -0.63732 -0.18116 0.18116 0.63732 1.5787 2.0274 2.4951	1.6857
11.4	10	-2.4784 -2.0229 -1.5837 -0.61465 -0.17462 0.17462 0.61465 1.5837 2.0229 2.4784	1.7066
11.6	10	-2.4625 -2.0191 -1.5896 -0.59321 -0.16839 0.16839 0.59321 1.5896 2.0191 2.4625	1.7269
11.8	10	-2.4475 -2.016 -1.5964 -0.57298 -0.16248 0.16248 0.57298 1.5964 2.016 2.4475	1.7466
12	10	-2.4331 -2.0133 -1.6038 -0.55384 -0.15686 0.15686 0.55384 1.6038 2.0133 2.4331	1.7657
12.2	10	-2.4194 -2.0112 -1.6117 -0.53572 -0.1515 0.1515 0.53572 1.6117 2.0112 2.4194	1.7841
12.4	10	-2.4063 -2.0093 -1.6199 -0.51843 -0.14636 0.14636 0.51843 1.6199 2.0093 2.4063	1.8019
12.6	10	-2.3936 -2.0078 -1.6283 -0.50198 -0.14144 0.14144 0.50198 1.6283 2.0078 2.3936	1.8188
12.8	10	-2.3815 -2.0065 -1.637 -0.48632 -0.13674 0.13674 0.48632 1.637 2.0065 2.3815	1.835
13	10	-2.3698 -2.0054 -1.6457 -0.47128 -0.13221 0.13221 0.47128 1.6457 2.0054 2.3698	1.8504
13.2	10	-2.3585 -2.0045 -1.6545 -0.45687 -0.12785 0.12785 0.45687 1.6545 2.0045 2.3585	1.865
13.4	10	-2.3476 -2.0038 -1.6632 -0.443 -0.12365 0.12365 0.443 1.6632 2.0038 2.3476	1.8787
⋮	⋮	⋮	⋮
15	10	-2.2723 -2.0007 -1.73 -0.3484 -0.094817 0.094817 0.3484 1.73 2.0007 2.2723	1.9575

Table A.9: Read threshold positions $\bar{\gamma}$ that attain the maximum mutual information for given number of reads m and SNR $= 10$(dB).

SNR(dB)	m	$\bar{\gamma}$	MMI
10	2	-1.7306 1.7306	1.1289
10	3	-1.9847 0 1.9847	1.4087
10	4	-2.0272 -0.40788 0.40788 2.0272	1.4503
10	5	-2.2908 -1.5111 0 1.5111 2.2908	1.4851
10	6	-2.3575 -1.6501 -0.35284 0.35284 1.6501 2.3575	1.5147
10	7	-2.3948 -1.7213 -0.63829 0 0.63829 1.7213 2.3948	1.5272
10	8	-2.5294 -1.9464 -1.3245 -0.31301 0.31301 1.3245 1.9464 2.5294	1.5382
10	9	-2.5778 -2.0193 -1.4751 -0.54305 0 0.54305 1.4751 2.0193 2.5778	1.5471
10	10	-2.6185 -2.0767 -1.5785 -0.79421 -0.22671 0.22671 0.79421 1.5785 2.0767 2.6185	1.5524
10	11	-2.696 -2.1853 -1.7577 -1.2242 -0.47464 0 0.47464 1.2242 1.7577 2.1853 2.696	1.5571
10	12	-2.735 -2.2378 -1.8363 -1.3759 -0.66151 -0.19688 0.19688 0.66151 1.3759 1.8363 2.2378 2.735	1.5609
10	13	-2.7696 -2.2839 -1.9023 -1.4911 -0.8824 -0.37733 0 0.37733 0.8824 1.4911 1.9023 2.2839 2.7696	1.5636
10	14	-2.8233 -2.3525 -1.9948 -1.6368 -1.1721 -0.5764 -0.17586 0.17586 0.5764 1.1721 1.6368 1.9948 2.3525 2.8233	1.566
10	15	-2.8554 -2.3936 -2.0489 -1.7167 -1.3149 -0.74044 -0.32936 0 0.32936 0.74044 1.3149 1.7167 2.0489 2.3936 2.8554	1.5679
⋮	⋮	⋮	⋮
10	19	-2.9787 -2.5471 -2.24 -1.9721 -1.7012 -1.3842 -0.95802 -0.55971 -0.26062 0 0.26062 0.55971 0.95802 1.3842 1.7012 1.9721 2.24 2.5471 2.9787	1.5729
10	20	-3.0069 -2.5815 -2.2814 -2.0238 -1.7703 -1.4865 -1.1222 -0.69961 -0.38319 -0.12254 0.12254 0.38319 0.69961 1.1222 1.4865 1.7703 2.0238 2.2814 2.5815 3.0069	1.5737
10	21	-3.0297 -2.6096 -2.315 -2.065 -1.824 -1.5626 -1.2414 -0.83143 -0.49829 -0.23521 0 0.23521 0.49829 0.83143 1.2414 1.5626 1.824 2.065 2.315 2.6096 3.0297	1.5745
⋮	⋮	⋮	⋮
10	30	-3.1722 -2.7868 -2.5262 -2.3174 -2.1338 -1.9607 -1.7873 -1.6032 -1.3955 -1.1482 -0.86881 -0.62381 -0.42099 -0.2438 -0.079891 0.079891 0.2438 0.42099 0.62381 0.86881 1.1482 1.3955 1.6032 1.7873 1.9607 2.1338 2.3174 2.5262 2.7868 3.1722	1.5781

Table A.10: Read threshold positions $\bar{\gamma}$ that attain the maximum mutual information for given number of reads m and SNR $= 13$(dB).

SNR(dB)	m	$\bar{\gamma}$	MMI
13	2	-1.9323 1.9323	1.3463
13	3	-1.999 0 1.999	1.7644
13	4	-2.0023 -0.2327 0.2327 2.0023	1.7865
13	5	-2.2207 -1.761 0 1.761 2.2207	1.8078
13	6	-2.2274 -1.7734 -0.22688 0.22688 1.7734 2.2274	1.8287
13	7	-2.2303 -1.7794 -0.3734 0 0.3734 1.7794 2.2303	1.8352
13	8	-2.3614 -1.9944 -1.6227 -0.2238 0.2238 1.6227 1.9944 2.3614	1.8415
13	9	-2.3674 -2.0015 -1.6369 -0.36429 0 0.36429 1.6369 2.0015 2.3674	1.8477
13	10	-2.3698 -2.0054 -1.6457 -0.47128 -0.13221 0.13221 0.47128 1.6457 2.0054 2.3698	1.8504
13	11	-2.4591 -2.1261 -1.8637 -1.526 -0.35816 0 0.35816 1.526 1.8637 2.1261 2.4591	1.8531
13	12	-2.4662 -2.133 -1.8724 -1.5424 -0.45952 -0.12979 0.12979 0.45952 1.5424 1.8724 2.133 2.4662	1.8557
13	13	-2.4682 -2.1365 -1.8781 -1.5546 -0.54638 -0.22693 0 0.22693 0.54638 1.5546 1.8781 2.1365 2.4682	1.8571
13	14	-2.5334 -2.2197 -1.9951 -1.7694 -1.4515 -0.45024 -0.12781 0.12781 0.45024 1.4515 1.7694 1.9951 2.2197 2.5334	1.8585
13	15	-2.5425 -2.2278 -2.0035 -1.7803 -1.4711 -0.53153 -0.22248 0 0.22248 0.53153 1.4711 1.7803 2.0035 2.2278 2.5425	1.8599
⋮	⋮	⋮	⋮
13	19	-2.6061 -2.3063 -2.1017 -1.9209 -1.7211 -1.4375 -0.66401 -0.36496 -0.16616 0 0.16616 0.36496 0.66401 1.4375 1.7211 1.9209 2.1017 2.3063 2.6061	1.8628
13	20	-2.6374 -2.3458 -2.1504 -1.9854 -1.8186 -1.6172 -1.3114 -0.56541 -0.28568 -0.088536 0.088536 0.28568 0.56541 1.3114 1.6172 1.8186 1.9854 2.1504 2.3458 2.6374	1.8633
13	21	-2.6582 -2.3649 -2.1689 -2.0051 -1.8423 -1.6498 -1.3675 -0.63596 -0.3538 -0.16183 0 0.16183 0.3538 0.63596 1.3675 1.6498 1.8423 2.0051 2.1689 2.3649 2.6582	1.8638
⋮	⋮	⋮	⋮
13	30	-2.8012 -2.516 -2.3315 -2.1882 -2.0641 -1.9461 -1.8235 -1.6838 -1.5079 -1.2489 -0.75241 -0.49399 -0.31873 -0.17984 -0.058206 0.058206 0.17984 0.31873 0.49399 0.75241 1.2489 1.5079 1.6838 1.8235 1.9461 2.0641 2.1882 2.3315 2.516 2.8012	1.8661

Table A.11: Read threshold positions $\bar{\gamma}$ that attain the maximum mutual information for given number of reads m and SNR $= 15$(dB).

SNR(dB)	m	$\bar{\gamma}$	MMI
15	2	-1.9722 1.9722	1.448
15	3	-2 0 2	1.9212
15	4	-2.0003 -0.16758 0.16758 2.0003	1.9304
15	5	-2.1651 -1.8312 0 1.8312 2.1651	1.9396
15	6	-2.1668 -1.8334 -0.16667 0.16667 1.8334 2.1668	1.9488
15	7	-2.167 -1.8344 -0.27393 0 0.27393 1.8344 2.167	1.9513
15	8	-2.2696 -1.998 -1.7248 -0.16607 0.16607 1.7248 1.998 2.2696	1.9539
15	9	-2.2726 -2.0002 -1.728 -0.27214 0 0.27214 1.728 2.0002 2.2726	1.9564
15	10	-2.2723 -2.0007 -1.73 -0.3484 -0.094817 0.094817 0.3484 1.73 2.0007 2.2723	1.9575
15	11	-2.3413 -2.091 -1.9026 -1.65 -0.27061 0 0.27061 1.65 1.9026 2.091 2.3413	1.9585
15	12	-2.3467 -2.0947 -1.9061 -1.6546 -0.34568 -0.094266 0.094266 0.34568 1.6546 1.9061 2.0947 2.3467	1.9595
15	13	-2.3456 -2.0949 -1.9072 -1.658 -0.40654 -0.16392 0 0.16392 0.40654 1.658 1.9072 2.0949 2.3456	1.9601
15	14	-2.3831 -2.1441 -1.9799 -1.8097 -1.5496 -0.3418 -0.093454 0.093454 0.3418 1.5496 1.8097 1.9799 2.1441 2.3831	1.9605
15	15	-2.4042 -2.1635 -2.0005 -1.8378 -1.598 -0.40245 -0.16266 0 0.16266 0.40245 1.598 1.8378 2.0005 2.1635 2.4042	1.9611
⋮	⋮	⋮	⋮
15	19	-2.4952 -2.247 -2.0894 -1.9564 -1.8094 -1.5902 -0.80544 -0.34451 -0.14567 0 0.14567 0.34451 0.80544 1.5902 1.8094 1.9564 2.0894 2.247 2.4952	1.962
15	20	-2.5433 -2.2737 -2.1099 -1.9766 -1.8356 -1.6382 -1.165 -0.42575 -0.20885 -0.063818 0.063818 0.20885 0.42575 1.165 1.6382 1.8356 1.9766 2.1099 2.2737 2.5433	1.9621
15	21	-2.5965 -2.3012 -2.1322 -2.0004 -1.8688 -1.7004 -1.4085 -0.59167 -0.29981 -0.13152 0 0.13152 0.29981 0.59167 1.4085 1.7004 1.8688 2.0004 2.1322 2.3012 2.5965	1.9624
⋮	⋮	⋮	⋮
15	30	-2.8808 -2.5054 -2.2996 -2.1612 -2.051 -1.949 -1.8388 -1.7005 -1.4949 -1.1214 -0.87864 -0.50513 -0.29948 -0.1612 -0.051001 0.051001 0.1612 0.29948 0.50513 0.87864 1.1214 1.4949 1.7005 1.8388 1.949 2.051 2.1612 2.2996 2.5054 2.8808	1.9631

Appendix B

MMI Optimal Thresholds of MLC-SCL LSB Channel

Table B.1: Read threshold positions $\bar{\gamma}$ of MLC-SCL LSB that attain the maximum mutual information for $m = 3$ at the given SNR value.

SNR(dB)	m	$\bar{\gamma}$	MMI
10	3	-2.1088 0 2.1088	0.49234
10.2	3	-2.1007 0 2.1007	0.51208
10.4	3	-2.0931 0 2.0931	0.53207
10.6	3	-2.0861 0 2.0861	0.55225
10.8	3	-2.0796 0 2.0796	0.57256
11	3	-2.0735 0 2.0735	0.59296
11.2	3	-2.0678 0 2.0678	0.61339
11.4	3	-2.0625 0 2.0625	0.63378
11.6	3	-2.0577 0 2.0577	0.65408
11.8	3	-2.0532 0 2.0532	0.67422
12	3	-2.049 0 2.049	0.69414
12.2	3	-2.0451 0 2.0451	0.71377
12.4	3	-2.0415 0 2.0415	0.73304
12.6	3	-2.0382 0 2.0382	0.75191
12.8	3	-2.0351 0 2.0351	0.77029
13	3	-2.0323 0 2.0323	0.78815
13.2	3	-2.0297 0 2.0297	0.80541
13.4	3	-2.0273 0 2.0273	0.82203
13.6	3	-2.0251 0 2.0251	0.83795
13.8	3	-2.023 0 2.023	0.85315
14	3	-2.0211 0 2.0211	0.86757
14.2	3	-2.0194 0 2.0194	0.88119
14.4	3	-2.0178 0 2.0178	0.89398
14.6	3	-2.0163 0 2.0163	0.90593
14.8	3	-2.015 0 2.015	0.91702
15	3	-2.0137 0 2.0137	0.92726

Table B.2: Read threshold positions $\bar{\gamma}$ of MLC-SCL LSB that attain the maximum mutual information for $m = 4$ at the given SNR value.

SNR(dB)	m	$\bar{\gamma}$	MMI
10	4	-2.0848 -0.20608 0.20608 2.0848	0.50106
10.2	4	-2.077 -0.2054 0.2054 2.077	0.52133
10.4	4	-2.0698 -0.20452 0.20452 2.0698	0.54182
10.6	4	-2.0632 -0.2033 0.2033 2.0632	0.56247
10.8	4	-2.0572 -0.2019 0.2019 2.0572	0.58321
11	4	-2.0517 -0.20024 0.20024 2.0517	0.60399
11.2	4	-2.0466 -0.19838 0.19838 2.0466	0.62474
11.4	4	-2.0421 -0.19632 0.19632 2.0421	0.6454
11.6	4	-2.0379 -0.19409 0.19409 2.0379	0.66591
11.8	4	-2.0341 -0.19166 0.19166 2.0341	0.68618
12	4	-2.0307 -0.18909 0.18909 2.0307	0.70617
12.2	4	-2.0276 -0.18639 0.18639 2.0276	0.7258
12.4	4	-2.0248 -0.18357 0.18357 2.0248	0.745
12.6	4	-2.0223 -0.18066 0.18066 2.0223	0.76373
12.8	4	-2.02 -0.17762 0.17762 2.02	0.78191
13	4	-2.018 -0.17451 0.17451 2.018	0.79949
13.2	4	-2.0161 -0.17139 0.17139 2.0161	0.81642
13.4	4	-2.0145 -0.16817 0.16817 2.0145	0.83264
13.6	4	-2.013 -0.16487 0.16487 2.013	0.84813
13.8	4	-2.0116 -0.16159 0.16159 2.0116	0.86284
14	4	-2.0104 -0.15825 0.15825 2.0104	0.87674
14.2	4	-2.0094 -0.15495 0.15495 2.0094	0.8898
14.4	4	-2.0084 -0.15156 0.15156 2.0084	0.90202
14.6	4	-2.0075 -0.14823 0.14823 2.0075	0.91338
14.8	4	-2.0067 -0.14492 0.14492 2.0067	0.92387
15	4	-2.0061 -0.14159 0.14159 2.0061	0.93351

Table B.3: Read threshold positions $\bar{\gamma}$ of MLC-SCL LSB that attain the maximum mutual information for $m = 5$ at the given SNR value.

SNR(dB)	m	$\bar{\gamma}$	MMI
10	5	-2.4829 -1.8737 0 1.8737 2.4829	0.53611
10.2	5	-2.4649 -1.8703 0 1.8703 2.4649	0.5561
10.4	5	-2.4475 -1.8673 0 1.8673 2.4475	0.5762
10.6	5	-2.4308 -1.8648 0 1.8648 2.4308	0.59636
10.8	5	-2.4147 -1.8627 0 1.8627 2.4147	0.61653
11	5	-2.3994 -1.861 0 1.861 2.3994	0.63663
11.2	5	-2.3845 -1.8596 0 1.8596 2.3845	0.65662
11.4	5	-2.3703 -1.8587 0 1.8587 2.3703	0.67644
11.6	5	-2.3566 -1.858 0 1.858 2.3566	0.69601
11.8	5	-2.3435 -1.8576 0 1.8576 2.3435	0.71528
12	5	-2.3309 -1.8576 0 1.8576 2.3309	0.7342
12.2	5	-2.3187 -1.8578 0 1.8578 2.3187	0.7527
12.4	5	-2.3071 -1.8582 0 1.8582 2.3071	0.77072
12.6	5	-2.2958 -1.8588 0 1.8588 2.2958	0.78821
12.8	5	-2.285 -1.8597 0 1.8597 2.285	0.80512
13	5	-2.2747 -1.8607 0 1.8607 2.2747	0.82141
13.2	5	-2.2647 -1.8619 0 1.8619 2.2647	0.83702
13.4	5	-2.2551 -1.8632 0 1.8632 2.2551	0.85192
13.6	5	-2.2458 -1.8647 0 1.8647 2.2458	0.86608
13.8	5	-2.2369 -1.8663 0 1.8663 2.2369	0.87946
14	5	-2.2284 -1.8679 0 1.8679 2.2284	0.89206
14.2	5	-2.2201 -1.8697 0 1.8697 2.2201	0.90384
14.4	5	-2.2122 -1.8716 0 1.8716 2.2122	0.91481
14.6	5	-2.2046 -1.8736 0 1.8736 2.2046	0.92497
14.8	5	-2.1972 -1.8756 0 1.8756 2.1972	0.93431
15	5	-2.1901 -1.8777 0 1.8777 2.1901	0.94285

Table B.4: Read threshold positions $\bar{\gamma}$ of MLC-SCL LSB that attain the maximum mutual information for $m = 6$ at the given SNR value.

SNR(dB)	m	$\bar{\gamma}$	MMI
10	6	-2.4504 -1.8198 -0.24672 0.24672 1.8198 2.4504	0.5499
10.2	6	-2.4327 -1.8168 -0.24496 0.24496 1.8168 2.4327	0.57055
10.4	6	-2.4157 -1.8143 -0.24291 0.24291 1.8143 2.4157	0.59126
10.6	6	-2.3996 -1.8125 -0.24057 0.24057 1.8125 2.3996	0.61197
10.8	6	-2.3842 -1.8112 -0.23796 0.23796 1.8112 2.3842	0.6326
11	6	-2.3695 -1.8104 -0.23509 0.23509 1.8104 2.3695	0.6531
11.2	6	-2.3554 -1.8102 -0.232 0.232 1.8102 2.3554	0.67339
11.4	6	-2.342 -1.8104 -0.22869 0.22869 1.8104 2.342	0.69341
11.6	6	-2.3291 -1.8109 -0.22519 0.22519 1.8109 2.3291	0.7131
11.8	6	-2.3169 -1.812 -0.22151 0.22151 1.812 2.3169	0.73239
12	6	-2.3051 -1.8133 -0.21769 0.21769 1.8133 2.3051	0.75123
12.2	6	-2.2939 -1.8149 -0.21374 0.21374 1.8149 2.2939	0.76955
12.4	6	-2.2832 -1.8168 -0.20969 0.20969 1.8168 2.2832	0.78731
12.6	6	-2.273 -1.819 -0.20553 0.20553 1.819 2.273	0.80445
12.8	6	-2.2631 -1.8214 -0.20132 0.20132 1.8214 2.2631	0.82092
13	6	-2.2537 -1.8239 -0.19704 0.19704 1.8239 2.2537	0.83668
13.2	6	-2.2446 -1.8267 -0.19273 0.19273 1.8267 2.2446	0.8517
13.4	6	-2.236 -1.8296 -0.18838 0.18838 1.8296 2.236	0.86595
13.6	6	-2.2276 -1.8326 -0.18403 0.18403 1.8326 2.2276	0.87941
13.8	6	-2.2196 -1.8357 -0.17968 0.17968 1.8357 2.2196	0.89204
14	6	-2.212 -1.8388 -0.17534 0.17534 1.8388 2.212	0.90385
14.2	6	-2.2046 -1.8421 -0.17102 0.17102 1.8421 2.2046	0.91483
14.4	6	-2.1975 -1.8454 -0.16672 0.16672 1.8454 2.1975	0.92498
14.6	6	-2.1906 -1.8487 -0.16246 0.16246 1.8487 2.1906	0.93431
14.8	6	-2.184 -1.852 -0.15825 0.15825 1.852 2.184	0.94283
15	6	-2.1776 -1.8554 -0.15409 0.15409 1.8554 2.1776	0.95056

Table B.5: Read threshold positions $\bar{\gamma}$ of MLC-SCL LSB that attain the maximum mutual information for $m = 7$ at the given SNR value.

SNR(dB)	m	$\bar{\gamma}$	MMI
10	7	-2.4405 -1.8026 -0.35483 0 0.35483 1.8026 2.4405	0.5534
10.2	7	-2.4228 -1.7995 -0.35316 0 0.35316 1.7995 2.4228	0.57423
10.4	7	-2.406 -1.7971 -0.35105 0 0.35105 1.7971 2.406	0.5951
10.6	7	-2.3899 -1.7953 -0.34859 0 0.34859 1.7953 2.3899	0.61596
10.8	7	-2.3746 -1.7941 -0.34577 0 0.34577 1.7941 2.3746	0.63671
11	7	-2.36 -1.7935 -0.34255 0 0.34255 1.7935 2.36	0.65731
11.2	7	-2.3461 -1.7934 -0.33901 0 0.33901 1.7934 2.3461	0.67769
11.4	7	-2.3328 -1.7939 -0.33514 0 0.33514 1.7939 2.3328	0.69777
11.6	7	-2.3202 -1.7947 -0.33099 0 0.33099 1.7947 2.3202	0.71749
11.8	7	-2.3082 -1.7961 -0.32656 0 0.32656 1.7961 2.3082	0.73679
12	7	-2.2967 -1.7978 -0.32188 0 0.32188 1.7978 2.2967	0.7556
12.2	7	-2.2857 -1.7998 -0.317 0 0.317 1.7998 2.2857	0.77388
12.4	7	-2.2753 -1.8022 -0.31191 0 0.31191 1.8022 2.2753	0.79156
12.6	7	-2.2653 -1.8048 -0.30665 0 0.30665 1.8048 2.2653	0.8086
12.8	7	-2.2557 -1.8076 -0.30129 0 0.30129 1.8076 2.2557	0.82495
13	7	-2.2466 -1.8107 -0.29576 0 0.29576 1.8107 2.2466	0.84058
13.2	7	-2.2379 -1.814 -0.29017 0 0.29017 1.814 2.2379	0.85544
13.4	7	-2.2295 -1.8174 -0.28452 0 0.28452 1.8174 2.2295	0.86952
13.6	7	-2.2215 -1.8209 -0.27879 0 0.27879 1.8209 2.2215	0.88278
13.8	7	-2.2138 -1.8245 -0.27304 0 0.27304 1.8245 2.2138	0.89522
14	7	-2.2064 -1.8282 -0.2673 0 0.2673 1.8282 2.2064	0.90683
14.2	7	-2.1994 -1.832 -0.26151 0 0.26151 1.832 2.1994	0.9176
14.4	7	-2.1925 -1.8358 -0.25579 0 0.25579 1.8358 2.1925	0.92753
14.6	7	-2.186 -1.8396 -0.25005 0 0.25005 1.8396 2.186	0.93664
14.8	7	-2.1797 -1.8434 -0.24439 0 0.24439 1.8434 2.1797	0.94495
15	7	-2.1736 -1.8473 -0.23874 0 0.23874 1.8473 2.1736	0.95248

Table B.6: Read threshold positions $\bar{\gamma}$ of MLC-SCL LSB that attain the maximum mutual information for $m = 8$ at the given SNR value.

SNR(dB)	m	$\bar{\gamma}$	MMI
10	8	-2.6664 -2.1458 -1.696 -0.26165 0.26165 1.696 2.1458 2.6664	0.56446
10.2	8	-2.6431 -2.135 -1.6946 -0.25949 0.25949 1.6946 2.135 2.6431	0.58516
10.4	8	-2.6206 -2.1248 -1.6938 -0.25701 0.25701 1.6938 2.1248 2.6206	0.60588
10.6	8	-2.599 -2.1154 -1.6936 -0.25425 0.25425 1.6936 2.1154 2.599	0.62654
10.8	8	-2.5783 -2.1067 -1.6939 -0.25121 0.25121 1.6939 2.1067 2.5783	0.64707
11	8	-2.5584 -2.0985 -1.6948 -0.24791 0.24791 1.6948 2.0985 2.5584	0.66742
11.2	8	-2.5392 -2.0909 -1.6961 -0.24437 0.24437 1.6961 2.0909 2.5392	0.68751
11.4	8	-2.5208 -2.0839 -1.6978 -0.24062 0.24062 1.6978 2.0839 2.5208	0.70729
11.6	8	-2.5031 -2.0774 -1.7001 -0.23667 0.23667 1.7001 2.0774 2.5031	0.72668
11.8	8	-2.486 -2.0713 -1.7026 -0.23255 0.23255 1.7026 2.0713 2.486	0.74563
12	8	-2.4696 -2.0657 -1.7056 -0.22828 0.22828 1.7056 2.0657 2.4696	0.76407
12.2	8	-2.4539 -2.0606 -1.7088 -0.22388 0.22388 1.7088 2.0606 2.4539	0.78196
12.4	8	-2.4387 -2.0558 -1.7124 -0.21938 0.21938 1.7124 2.0558 2.4387	0.79925
12.6	8	-2.4241 -2.0514 -1.7161 -0.21479 0.21479 1.7161 2.0514 2.4241	0.81588
12.8	8	-2.4101 -2.0474 -1.7202 -0.21014 0.21014 1.7202 2.0474 2.4101	0.83182
13	8	-2.3965 -2.0436 -1.7244 -0.20544 0.20544 1.7244 2.0436 2.3965	0.84702
13.2	8	-2.3834 -2.0401 -1.7287 -0.20071 0.20071 1.7287 2.0401 2.3834	0.86146
13.4	8	-2.3708 -2.0369 -1.7332 -0.19597 0.19597 1.7332 2.0369 2.3708	0.87512
13.6	8	-2.3587 -2.0339 -1.7378 -0.19122 0.19122 1.7378 2.0339 2.3587	0.88797
13.8	8	-2.347 -2.0312 -1.7426 -0.18648 0.18648 1.7426 2.0312 2.347	0.9
14	8	-2.3356 -2.0286 -1.7473 -0.18177 0.18177 1.7473 2.0286 2.3356	0.9112
14.2	8	-2.3247 -2.0263 -1.7522 -0.17709 0.17709 1.7522 2.0263 2.3247	0.92158
14.4	8	-2.3142 -2.0241 -1.757 -0.17245 0.17245 1.757 2.0241 2.3142	0.93114
14.6	8	-2.3039 -2.022 -1.7619 -0.16787 0.16787 1.7619 2.022 2.3039	0.93989
14.8	8	-2.294 -2.0201 -1.7668 -0.16334 0.16334 1.7668 2.0201 2.294	0.94786
15	8	-2.2845 -2.0184 -1.7716 -0.15887 0.15887 1.7716 2.0184 2.2845	0.95506

Table B.7: Read threshold positions $\bar{\gamma}$ of MLC-SCL LSB that attain the maximum mutual information for $m = 10$ at the given SNR value.

SNR(dB)	m	$\bar{\gamma}$	MMI
10	10	-2.6496 -2.1219 -1.6566 -0.45039 -0.14179 0.14179 0.45039 1.6566 2.1219 2.6496	0.57037
10.2	10	-2.6263 -2.1111 -1.6548 -0.44861 -0.14076 0.14076 0.44861 1.6548 2.1111 2.6263	0.59136
10.4	10	-2.604 -2.1012 -1.6537 -0.44634 -0.13956 0.13956 0.44634 1.6537 2.1012 2.604	0.61233
10.6	10	-2.5826 -2.0919 -1.6533 -0.44366 -0.13821 0.13821 0.44366 1.6533 2.0919 2.5826	0.63321
10.8	10	-2.562 -2.0834 -1.6535 -0.44052 -0.1367 0.1367 0.44052 1.6535 2.0834 2.562	0.65394
11	10	-2.5423 -2.0755 -1.6543 -0.43693 -0.13504 0.13504 0.43693 1.6543 2.0755 2.5423	0.67444
11.2	10	-2.5234 -2.0682 -1.6558 -0.43294 -0.13326 0.13326 0.43294 1.6558 2.0682 2.5234	0.69465
11.4	10	-2.5053 -2.0616 -1.6578 -0.42847 -0.13132 0.13132 0.42847 1.6578 2.0616 2.5053	0.7145
11.6	10	-2.4879 -2.0556 -1.6603 -0.42367 -0.12929 0.12929 0.42367 1.6603 2.0556 2.4879	0.73392
11.8	10	-2.4712 -2.05 -1.6633 -0.41844 -0.12713 0.12713 0.41844 1.6633 2.05 2.4712	0.75285
12	10	-2.4552 -2.045 -1.6668 -0.41289 -0.12488 0.12488 0.41289 1.6668 2.045 2.4552	0.77124
12.2	10	-2.4398 -2.0404 -1.6707 -0.40704 -0.12256 0.12256 0.40704 1.6707 2.0404 2.4398	0.78903
12.4	10	-2.4251 -2.0363 -1.675 -0.40086 -0.12015 0.12015 0.40086 1.675 2.0363 2.4251	0.80618
12.6	10	-2.4109 -2.0326 -1.6796 -0.39446 -0.11769 0.11769 0.39446 1.6796 2.0326 2.4109	0.82263
12.8	10	-2.3974 -2.0292 -1.6845 -0.38784 -0.11519 0.11519 0.38784 1.6845 2.0292 2.3974	0.83835
13	10	-2.3843 -2.0262 -1.6897 -0.38104 -0.11264 0.11264 0.38104 1.6897 2.0262 2.3843	0.8533
13.2	10	-2.3718 -2.0234 -1.6951 -0.37406 -0.11007 0.11007 0.37406 1.6951 2.0234 2.3718	0.86747
13.4	10	-2.3597 -2.021 -1.7007 -0.36696 -0.10748 0.10748 0.36696 1.7007 2.021 2.3597	0.88082
⋮	⋮	⋮	⋮
15	10	-2.2779 -2.0085 -1.7486 -0.3088 -0.087074 0.087074 0.3088 1.7486 2.0085 2.2779	0.95803

Table B.8: Read threshold positions $\bar{\gamma}$ of MLC-SCL LSB that attain the maximum mutual information for given number of reads m and SNR $= 10$(dB).

SNR(dB)	m	$\bar{\gamma}$	MMI
10	3	-2.1088 0 2.1088	0.49234
10	4	-2.0848 -0.20608 0.20608 2.0848	0.50106
10	5	-2.4829 -1.8737 0 1.8737 2.4829	0.53611
10	6	-2.4504 -1.8198 -0.24672 0.24672 1.8198 2.4504	0.5499
10	7	-2.4405 -1.8026 -0.35483 0 0.35483 1.8026 2.4405	0.5534
10	8	-2.6664 -2.1458 -1.696 -0.26165 0.26165 1.696 2.1458 2.6664	0.56446
10	9	-2.6544 -2.1291 -1.6689 -0.38175 0 0.38175 1.6689 2.1291 2.6544	0.56869
10	10	-2.6496 -2.1219 -1.6566 -0.45039 -0.14179 0.14179 0.45039 1.6566 2.1219 2.6496	0.57037
10	11	-2.8041 -2.3292 -1.9646 -1.59 -0.39536 0 0.39536 1.59 1.9646 2.3292 2.8041	0.57526
10	12	-2.7981 -2.3215 -1.954 -1.5734 -0.46853 -0.14662 0.14662 0.46853 1.5734 1.954 2.3215 2.7981	0.57712
10	13	-2.7955 -2.3176 -1.9484 -1.5641 -0.51674 -0.24246 0 0.24246 0.51674 1.5641 1.9484 2.3176 2.7955	0.57803
10	14	-2.911 -2.4652 -2.1408 -1.8446 -1.5178 -0.47956 -0.14963 0.14963 0.47956 1.5178 1.8446 2.1408 2.4652 2.911	0.58062
10	15	-2.9076 -2.4608 -2.1353 -1.8372 -1.5064 -0.52965 -0.24734 0 0.24734 0.52965 1.5064 1.8372 2.1353 2.4608 2.9076	0.5816
⋮	⋮	⋮	⋮
10	19	-2.994 -2.5676 -2.2662 -2.0061 -1.7476 -1.4505 -0.60392 -0.37698 -0.18217 0 0.18217 0.37698 0.60392 1.4505 1.7476 2.0061 2.2662 2.5676 2.994	0.58463
10	20	-3.0616 -2.6526 -2.3694 -2.1337 -1.9136 -1.6858 -1.4216 -0.58314 -0.32574 -0.10524 0.10524 0.32574 0.58314 1.4216 1.6858 1.9136 2.1337 2.3694 2.6526 3.0616	0.58562
10	21	-3.0663 -2.6564 -2.3722 -2.1355 -1.914 -1.6844 -1.4169 -0.61122 -0.38021 -0.18342 0 0.18342 0.38021 0.61122 1.4169 1.6844 1.914 2.1355 2.3722 2.6564 3.0663	0.58599
⋮	⋮	⋮	⋮
10	30	-3.1778 -2.7967 -2.541 -2.3383 -2.1626 -2.0003 -1.8423 -1.6813 -1.5089 -1.3103 -0.69926 -0.5138 -0.35428 -0.20798 -0.068597 0.068597 0.20798 0.35428 0.5138 0.69926 1.3103 1.5089 1.6813 1.8423 2.0003 2.1626 2.3383 2.541 2.7967 3.1778	0.58855

Table B.9: Read threshold positions $\bar{\gamma}$ of MLC-SCL LSB that attain the maximum mutual information for given number of reads m and SNR = 13(dB).

SNR(dB)	m	$\bar{\gamma}$	MMI
13	3	-2.0323 0 2.0323	0.78815
13	4	-2.018 -0.17451 0.17451 2.018	0.79949
13	5	-2.2747 -1.8607 0 1.8607 2.2747	0.82141
13	6	-2.2537 -1.8239 -0.19704 0.19704 1.8239 2.2537	0.83668
13	7	-2.2466 -1.8107 -0.29576 0 0.29576 1.8107 2.2466	0.84058
13	8	-2.3965 -2.0436 -1.7244 -0.20544 0.20544 1.7244 2.0436 2.3965	0.84702
13	9	-2.3878 -2.0315 -1.7008 -0.31434 0 0.31434 1.7008 2.0315 2.3878	0.85153
13	10	-2.3843 -2.0262 -1.6897 -0.38104 -0.11264 0.11264 0.38104 1.6897 2.0262 2.3843	0.8533
13	11	-2.4863 -2.1614 -1.9161 -1.6336 -0.32365 0 0.32365 1.6336 1.9161 2.1614 2.4863	0.85604
13	12	-2.4822 -2.1556 -1.9074 -1.6166 -0.39536 -0.11587 0.11587 0.39536 1.6166 1.9074 2.1556 2.4822	0.85797
13	13	-2.481 -2.1533 -1.9035 -1.6077 -0.44505 -0.19514 0 0.19514 0.44505 1.6077 1.9035 2.1533 2.481	0.85893
13	14	-2.5558 -2.2482 -2.0325 -1.8271 -1.5669 -0.40368 -0.11773 0.11773 0.40368 1.5669 1.8271 2.0325 2.2482 2.5558	0.86034
13	15	-2.5551 -2.2458 -2.0284 -1.8203 -1.5536 -0.45628 -0.19869 0 0.19869 0.45628 1.5536 1.8203 2.0284 2.2458 2.5551	0.86135
⋮	⋮	⋮	⋮
13	19	-2.6162 -2.3185 -2.1165 -1.9407 -1.7521 -1.4992 -0.53693 -0.3133 -0.14604 0 0.14604 0.3133 0.53693 1.4992 1.7521 1.9407 2.1165 2.3185 2.6162	0.86371
13	20	-2.6586 -2.3726 -2.1826 -2.0258 -1.8739 -1.7024 -1.4722 -0.50987 -0.26488 -0.082891 0.082891 0.26488 0.50987 1.4722 1.7024 1.8739 2.0258 2.1826 2.3726 2.6586	0.86424
13	21	-2.6645 -2.3758 -2.184 -2.0257 -1.8721 -1.698 -1.463 -0.54396 -0.31562 -0.14681 0 0.14681 0.31562 0.54396 1.463 1.698 1.8721 2.0257 2.184 2.3758 2.6645	0.86462
⋮	⋮	⋮	⋮
13	30	-2.802 -2.5177 -2.3341 -2.1919 -2.0692 -1.9536 -1.8349 -1.7027 -1.5432 -1.3323 -0.66936 -0.46037 -0.3025 -0.17215 -0.055915 0.055915 0.17215 0.3025 0.46037 0.66936 1.3323 1.5432 1.7027 1.8349 1.9536 2.0692 2.1919 2.3341 2.5177 2.802	0.86651

Table B.10: Read threshold positions $\bar{\gamma}$ of MLC-SCL LSB that attain the maximum mutual information for given number of reads m and SNR $= 15$(dB).

SNR(dB)	m	$\bar{\gamma}$	MMI
15	3	-2.0137 0 2.0137	0.92726
15	4	-2.0061 -0.14159 0.14159 2.0061	0.93351
15	5	-2.1901 -1.8777 0 1.8777 2.1901	0.94285
15	6	-2.1776 -1.8554 -0.15409 0.15409 1.8554 2.1776	0.95056
15	7	-2.1736 -1.8473 -0.23874 0 0.23874 1.8473 2.1736	0.95248
15	8	-2.2845 -2.0184 -1.7716 -0.15887 0.15887 1.7716 2.0184 2.2845	0.95506
15	9	-2.2803 -2.0116 -1.7557 -0.25066 0 0.25066 1.7557 2.0116 2.2803	0.95721
15	10	-2.2779 -2.0085 -1.7486 -0.3088 -0.087074 0.087074 0.3088 1.7486 2.0085 2.2779	0.95803
15	11	-2.3519 -2.1047 -1.9236 -1.6981 -0.25615 0 0.25615 1.6981 1.9236 2.1047 2.3519	0.95907
15	12	-2.3528 -2.1031 -1.9194 -1.6865 -0.31844 -0.088987 0.088987 0.31844 1.6865 1.9194 2.1031 2.3528	0.95995
15	13	-2.35 -2.1008 -1.9166 -1.6805 -0.36289 -0.15152 0 0.15152 0.36289 1.6805 1.9166 2.1008 2.35	0.96038
15	14	-2.3953 -2.16 -2.0016 -1.8461 -1.6309 -0.32503 -0.090253 0.090253 0.32503 1.6309 1.8461 2.0016 2.16 2.3953	0.9609
15	15	-2.4091 -2.1701 -2.0095 -1.853 -1.6333 -0.37071 -0.15369 0 0.15369 0.37071 1.6333 1.853 2.0095 2.1701 2.4091	0.96135
⋮	⋮	⋮	⋮
15	19	-2.4953 -2.247 -2.0894 -1.9564 -1.8092 -1.5897 -0.78274 -0.34127 -0.14473 0 0.14473 0.34127 0.78274 1.5897 1.8092 1.9564 2.0894 2.247 2.4953	0.96198
15	20	-2.5435 -2.274 -2.1103 -1.9772 -1.8366 -1.6405 -1.1808 -0.42603 -0.20893 -0.063837 0.063837 0.20893 0.42603 1.1808 1.6405 1.8366 1.9772 2.1103 2.274 2.5435	0.96216
15	21	-2.5969 -2.3018 -2.133 -2.0017 -1.8709 -1.7052 -1.4267 -0.57389 -0.2956 -0.1302 0 0.1302 0.2956 0.57389 1.4267 1.7052 1.8709 2.0017 2.133 2.3018 2.5969	0.96244
⋮	⋮	⋮	⋮
15	30	-2.8808 -2.5054 -2.2996 -2.1612 -2.051 -1.949 -1.8388 -1.7006 -1.4951 -1.1225 -0.87746 -0.50491 -0.29941 -0.16117 -0.050993 0.050993 0.16117 0.29941 0.50491 0.87746 1.1225 1.4951 1.7006 1.8388 1.949 2.051 2.1612 2.2996 2.5054 2.8808	0.9631

Appendix C

DE Optimal Thresholds of MLC-SCL LSB Channel

Table C.1: DE-optimized read threshold positions $\bar{\gamma}$ of MLC-SCL LSB for Code A $m = 5$ at the given SNR value. Thresholds are found using gradient descent initialized to MMI optimal positions.

SNR(dB)	m	$\bar{\gamma}$	BER(BP)
10	5	-2.4302 -1.8640 0 1.8640 2.4302	5.0447×10^{-2}
11	5	-2.3924 -1.8596 0 1.8596 2.3924	1.9865×10^{-2}
12	5	-2.3614 -1.8740 0 1.8740 2.3614	5.5126×10^{-3}
13	5	-2.2848 -1.8481 0 1.8481 2.2848	1.1762×10^{-3}
14	5	-2.2408 -1.8572 0 1.8572 2.2408	2.3101×10^{-4}
15	5	-2.1901 -1.8777 0 1.8777 2.1901	5.1858×10^{-5}

Table C.2: DE-optimized read threshold positions $\bar{\gamma}$ of MLC-SCL LSB for Code A $m = 6$ at the given SNR value.

SNR(dB)	m	$\bar{\gamma}$
10	6	-2.4300 -1.8352 -0.2379 0.2379 1.8352 2.4300
11	6	-2.3607 -1.8087 -0.2341 0.2341 1.8087 2.3607
12	6	-2.3143 -1.8044 -0.2261 0.2261 1.8044 2.3143
13	6	-2.2687 -1.8109 -0.2086 0.2086 1.8109 2.2687
14	6	-2.2122 -1.8386 -0.1755 0.1755 1.8386 2.2122
15	6	-2.1776 -1.8554 -0.15409 0.15409 1.8554 2.1776

Table C.3: DE-optimized read threshold positions γ of MLC-SCL LSB for Code B, $m = 5$ at the given SNR value.

SNR(dB)	m	γ
10	5	-2.0059 -2.0052 0 2.0052 2.0059
11	5	-2.1065 -1.9134 0 1.9134 2.1065
12	5	-2.2172 -1.8640 0 1.8640 2.2172
13	5	-2.2379 -1.8534 0 1.8534 2.2379
14	5	-2.2366 -1.8684 0 1.8684 2.2366
15	5	-2.2239 -1.8769 0 1.8769 2.2239

Appendix D

Double Diagonal LDPC Code Base Matrix in 802.11n

Table D.1: The base matrices of LDPC codes used in the Wi-Fi standards 802.11n of code length $n = 1944$ bits, sub-block length $Z = 81$ of various rates [11].

(a) Coding rate R = 1/2.

57	-	-	-	50	-	11	-	50	-	79	-	1	0	-	-	-	-	-	-	-	-	-	-
3	-	28	-	0	-	-	-	55	7	-	-	-	0	0	-	-	-	-	-	-	-	-	-
30	-	-	-	24	37	-	-	56	14	-	-	-	-	0	0	-	-	-	-	-	-	-	-
62	53	-	-	53	-	-	3	35	-	-	-	-	-	-	0	0	-	-	-	-	-	-	-
40	-	-	20	66	-	-	22	28	-	-	-	-	-	-	-	0	0	-	-	-	-	-	-
0	-	-	-	8	-	42	-	50	-	-	8	-	-	-	-	-	0	0	-	-	-	-	-
69	79	79	-	-	-	56	-	52	-	-	-	0	-	-	-	-	-	0	0	-	-	-	-
65	-	-	-	38	57	-	-	72	-	27	-	-	-	-	-	-	-	-	0	0	-	-	-
64	-	-	-	14	52	-	-	30	-	-	32	-	-	-	-	-	-	-	-	0	0	-	-
-	45	-	70	0	-	-	-	77	9	-	-	-	-	-	-	-	-	-	-	-	0	0	-
2	56	-	57	35	-	-	-	-	-	12	-	-	-	-	-	-	-	-	-	-	-	0	0
24	-	61	-	60	-	-	27	51	-	-	16	1	-	-	-	-	-	-	-	-	-	-	0

(b) Coding rate R = 2/3.

61	75	4	63	56	-	-	-	-	-	-	8	-	2	17	25	1	0	-	-	-	-	-	-
56	74	77	20	-	-	-	64	24	4	67	-	7	-	-	-	-	0	0	-	-	-	-	-
28	21	68	10	7	14	65	-	-	-	23	-	-	-	75	-	-	-	0	0	-	-	-	-
48	38	43	78	76	-	-	-	-	5	36	-	15	72	-	-	-	-	-	0	0	-	-	-
40	2	53	25	-	52	62	-	20	-	-	44	-	-	-	-	0	-	-	-	0	0	-	-
69	23	64	10	22	-	21	-	-	-	-	-	68	23	29	-	-	-	-	-	-	0	0	-
12	0	68	20	55	61	-	40	-	-	-	52	-	-	-	44	-	-	-	-	-	-	0	0
58	8	34	64	78	-	-	11	78	24	-	-	-	-	-	58	1	-	-	-	-	-	-	0

(c) Coding rate R = 3/4.

48	29	28	39	9	61	-	-	-	63	45	80	-	-	-	37	32	22	1	0	-	-	-	-
4	49	42	48	11	30	-	-	-	49	17	41	37	15	-	54	-	-	-	0	0	-	-	-
35	76	78	51	37	35	21	-	17	64	-	-	-	59	7	-	-	32	-	-	0	0	-	-
9	65	44	9	54	56	73	34	42	-	-	-	35	-	-	-	46	39	0	-	-	0	0	-
3	62	7	80	68	26	-	80	55	-	36	-	26	-	9	-	72	-	-	-	-	-	0	0
26	75	33	21	69	59	3	38	-	-	-	35	-	62	36	26	-	-	1	-	-	-	-	0

(d) Coding rate R = 5/6.

13	48	80	66	4	74	7	30	76	52	37	60	-	49	73	31	74	73	23	-	1	0	-	-
69	63	74	56	64	77	57	65	6	16	51	-	64	-	68	9	48	62	54	27	-	0	0	-
51	15	0	80	24	25	42	54	44	71	71	9	67	35	-	58	-	29	-	53	0	-	0	0
16	29	36	41	44	56	59	37	50	24	-	65	4	65	52	-	4	-	73	52	1	-	-	0

Bibliography

[1] Erich F. Haratsch. NAND flash media management algorithms, 2017.

[2] Zachary Painter. NAND flash memory technology: The basics of a flash memory cell, June 2018.

[3] Y. Cai, S. Ghose, E. F. Haratsch, Y. Luo, and O. Mutlu. Error characterization, mitigation, and recovery in flash-memory-based solid-state drives. *Proceedings of the IEEE*, 105(9):1666–1704, 2017.

[4] C. A. Aslam, Y. L. Guan, and K. Cai. Read and write voltage signal optimization for multi-level-cell (MLC) NAND flash memory. *IEEE Transactions on Communications*, 64(4):1613–1623, 2016.

[5] Y. Cai, E. F. Haratsch, O. Mutlu, and K. Mai. Threshold voltage distribution in MLC NAND flash memory: Characterization, analysis, and modeling. In *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1285–1290, 2013.

[6] Kai Zhao, Wenzhe Zhao, Hongbin Sun, Tong Zhang, Xiaodong Zhang, and Nanning Zheng. LDPC-in-SSD: making advanced error correction codes work effectively in solid state drives. In *FAST*, 2013.

[7] Jeremy Thorpe. Low-density parity-check (LDPC) codes constructed from protographs. In *Proc. IPN Progr. Rep.*, pages 1–7, Aug 2003.

[8] Eshed Ram and Yuval Cassuto. LDPC codes with local and global decoding. *CoRR*, abs/1801.03951, 2018.

[9] T. J. Richardson and R. L. Urbanke. Efficient encoding of low-density parity-check codes. *IEEE Transactions on Information Theory*, 47(2):638–656, 2001.

[10] Tom Richardson and Ruediger Urbanke. *Modern coding theory*. Cambridge university press, 2008.

[11] IEEE standard for information technology—telecommunications and information exchange between systems local and metropolitan area networks—specific requirements - part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)*, pages 1–3534, 2016.

[12] Yishen Yeh, Arman Fazeli, and Paul H. Siegel. Optimization of read thresholds in MLC NAND memory for LDPC codes. In *Proc. 11th Annu. Non-Volatile Memo-*

ries Workshop (NVMW), La Jolla, California, USA, March 2020. [Online]. Available: http://nvmw.ucsd.edu/program/.

[13] X. Zhang, J. Zhu, and Y. Wu. Efficient one-pass Chase soft-decision BCH decoder for multi-level cell NAND flash memory. In *2011 IEEE 54th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 1–4, 2011.

[14] X. Zhang. An efficient interpolation-based Chase BCH decoder. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 60(4):212–216, 2013.

[15] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke. Design of capacity-approaching irregular low-density parity-check codes. *IEEE Transactions on Information Theory*, 47(2):619–637, 2001.

[16] F. Zhang, H. D. Pfister, and A. Jiang. LDPC codes for rank modulation in flash memories. In *2010 IEEE International Symposium on Information Theory*, pages 859–863, 2010.

[17] Anxiao Jiang, M. Schwartz, and J. Bruck. Error-correcting codes for rank modulation. In *2008 IEEE International Symposium on Information Theory*, pages 1736–1740, 2008.

[18] J. Wang, K. Vakilinia, T. Chen, T. Courtade, G. Dong, T. Zhang, H. Shankar, and R. Wesel. Enhanced precision through multiple reads for LDPC decoding in flash memories. *IEEE Journal on Selected Areas in Communications*, 32(5):880–891, 2014.

[19] C. Duangthong, W. Phakphisut, and P. Supnithi. Read voltage optimization in MLC NAND flash memory via the density evolution. In *2019 26th International Conference on Telecommunications (ICT)*, pages 361–365, 2019.

[20] Sae-Young Chung, G. D. Forney, T. J. Richardson, and R. Urbanke. On the design of low-density parity-check codes within 0.0045 db of the Shannon limit. *IEEE Communications Letters*, 5(2):58–60, 2001.

[21] R. Bez, E. Camerlenghi, A. Modelli, and A. Visconti. Introduction to flash memory. *Proceedings of the IEEE*, 91(4):489–502, 2003.

[22] How data are read from a NAND type flash memory cell. *SHG2A Series | TECH JOURNAL | TDK Product Center*.

[23] Xueqiang Wang, Guiqiang Dong, Liyang Pan, and Runde Zhou. *Error Correction Codes and Signal Processing in Flash Memory, Flash Memories*. IntechOpen, Sep 2011.

[24] Quan Xu, Pu Gong, Thomas M. Chen, John Michael, and Shancang Li. Modelling and characterization of NAND flash memory channels. *Measurement*, 70:225 – 231, 2015.

[25] Kang-Deog Suh, Byung-Hoon Suh, Young-Ho Lim, Jin-Ki Kim, Young-Joon Choi, Yong-Nam Koh, Sung-Soo Lee, Suk-Chon Kwon, Byung-Soon Choi, Jin-Sun Yum, Jung-Hyuk Choi, Jang-Rae Kim, and Hyung-Kyu Lim. A 3.3 V 32 Mb NAND flash memory with incremental step pulse programming scheme. *IEEE Journal of Solid-State Circuits*, 30(11):1149–1156, 1995.

[26] G. Dong, N. Xie, and T. Zhang. Enabling NAND flash memory use soft-decision error correction codes at minimal read latency overhead. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 60(9):2412–2421, 2013.

[27] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, USA, 2006.

[28] P. Huang, P. H. Siegel, and E. Yaakobi. Performance of multilevel flash memories with different binary labelings: A multi-user perspective. *IEEE Journal on Selected Areas in Communications*, 34(9):2336–2353, 2016.

[29] R. Gallager. Low-density parity-check codes. *IRE Transactions on Information Theory*, 8(1):21–28, 1962.

[30] Ron Roth. *Introduction to Coding Theory*. Cambridge University Press, USA, 2006.

[31] J. Fan and Y. Xiao. A method of counting the number of cycles in LDPC codes. In *2006 8th international Conference on Signal Processing*, volume 3, 2006.

[32] D. J. C. MacKay and R. M. Neal. Near Shannon limit performance of low density parity check codes. *Electronics Letters*, 32(18):1645–, 1996.

[33] Li Ping and K. Y. Wu. Concatenated tree codes: a low-complexity, high-performance approach. *IEEE Transactions on Information Theory*, 47(2):791–799, 2001.

[34] Dariush Divsalar, Hui Jin, and Robert J. McEliece. Coding theorems for "turbo-like" codes. *Proceedings of the 1998 Allerton Conference*, page 210, 1998.

[35] Hui Jin, Aamod Khandekar, and Robert J. McEliece. Irregular repeat-accumulate codes. *2nd International Symposium on Turbo Codes and Related Topics*, pages 1–8, 11 2000.

[36] MacKay, David J. C. and Neal, Radford M. Good codes based on very sparse matrices. In Colin Boyd, editor, *Cryptography and Coding*, pages 100–111, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.

[37] Ido Kanter and David Saad. Error-correcting codes that nearly saturate Shannon's bound. *Phys. Rev. Lett.*, 83:2660–2663, Sep 1999.

[38] Tom Richardson and Rüdiger Urbanke. Multi-edge type LDPC codes. *ISIT talk*, 01 2002.

[39] Y. Fang, G. Bi, Y. L. Guan, and F. C. M. Lau. A survey on protograph LDPC codes and their applications. *IEEE Communications Surveys Tutorials*, 17(4):1989–2016, 2015.

[40] D. Divsalar, S. Dolinar, C. R. Jones, and K. Andrews. Capacity-approaching protograph codes. *IEEE Journal on Selected Areas in Communications*, 27(6):876–888, 2009.

[41] Xiao-Yu Hu, E. Eleftheriou, and D. M. Arnold. Regular and irregular progressive edge-growth tanner graphs. *IEEE Transactions on Information Theory*, 51(1):386–398, 2005.

[42] M. P. C. Fossorier. Quasicyclic low-density parity-check codes from circulant permutation matrices. *IEEE Transactions on Information Theory*, 50(8):1788–1793, 2004.

[43] C. Sun, H. Xu, D. Feng, and B. Bai. (3, l) quasi-cyclic LDPC codes: Simplified exhaustive search and designs. In *2016 9th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, pages 271–275, 2016.

[44] A. Kalsi, A. Bajpai, L. Wuttisittikulkij, and P. Kovintaewat. A base matrix method to construct column weight 3 quasi-cyclic LDPC codes with high girth. In *2016 International Conference on Electronics, Information, and Communications (ICEIC)*, pages 1–4, 2016.

[45] Seho Myung and Kyeongcheol Yang. Extension of quasi-cyclic LDPC codes by lifting. In *Proceedings. International Symposium on Information Theory, 2005. ISIT 2005.*, pages 2305–2309, 2005.

[46] D. Divsalar, S. Dolinar, J. Thorpe, and C. Jones. Constructing LDPC codes from simple loop-free encoding modules. In *IEEE International Conference on Communications, 2005. ICC 2005. 2005*, volume 1, pages 658–662 Vol. 1, 2005.

[47] George I. Davida and Sudhakar M. Reddy. Forward-error correction with decision feedback. *Information and Control*, 21(2):117 – 133, 1972.

[48] M. El-Khamy, J. Hou, and N. Bhushan. Design of rate-compatible structured LDPC codes for hybrid ARQ applications. *IEEE Journal on Selected Areas in Communications*, 27(6):965–973, 2009.

[49] P. Huang, Y. Liu, X. Zhang, P. H. Siegel, and E. F. Haratsch. Syndrome-coupled rate-compatible error-correcting codes: Theory and application. *IEEE Transactions on Information Theory*, 66(4):2311–2330, 2020.

[50] P. Chen, K. Cai, and S. Zheng. Rate-adaptive protograph LDPC codes for multi-level-cell NAND flash memory. *IEEE Communications Letters*, 22(6):1112–1115, 2018.

[51] C. Yoon, E. Choi, M. Cheong, and S. Lee. Arbitrary bit generation and correction technique for encoding QC-LDPC codes with dual-diagonal parity structure. In *2007 IEEE Wireless Communications and Networking Conference*, pages 662–666, 2007.

[52] C. Yoon, J. Oh, M. Cheong, and S. Lee. A hardware efficient LDPC encoding scheme for exploiting decoder structure and resources. In *2007 IEEE 65th Vehicular Technology Conference - VTC2007-Spring*, pages 2445–2449, 2007.

[53] Chia-Yu Lin, Chih-Chun Wei, and Mong-Kai Ku. Efficient encoding for dual-diagonal structured LDPC codes based on parity bit prediction and correction. In *APCCAS 2008 - 2008 IEEE Asia Pacific Conference on Circuits and Systems*, pages 1648–1651, 2008.

[54] S. K. Planjery, D. Declercq, L. Danjean, and B. Vasic. Finite alphabet iterative decoders for LDPC codes surpassing floating-point iterative decoders. *Electronics Letters*, 47(16):919–921, 2011.

[55] Shiva Kumar Planjery, David Declercq, Ludovic Danjean, and Bane V. Vasic. Finite alphabet iterative decoders, part I: decoding beyond belief propagation on BSC. *CoRR*, abs/1207.4800, 2012.

[56] F. Cai, X. Zhang, D. Declercq, S. K. Planjery, and B. Vasić. Finite alphabet iterative decoders for LDPC codes: Optimization, architecture and analysis. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 61(5):1366–1375, 2014.

[57] J. Hagenauer, E. Offer, and L. Papke. Iterative decoding of binary block and convolutional codes. *IEEE Trans. Inf. Theor.*, 42(2):429–445, September 2006.

[58] M. P. C. Fossorier, M. Mihaljevic, and H. Imai. Reduced complexity iterative decoding of low-density parity check codes based on belief propagation. *IEEE Transactions on Communications*, 47(5):673–680, 1999.

[59] Jinghu Chen and M. P. C. Fossorier. Near optimum universal belief propagation based decoding of low-density parity check codes. *IEEE Transactions on Communications*, 50(3):406–414, 2002.

[60] Jinghu Chen, A. Dholakia, E. Eleftheriou, M. P. C. Fossorier, and Xiao-Yu Hu. Reduced-complexity decoding of LDPC codes. *IEEE Transactions on Communications*, 53(8):1288–1299, 2005.

[61] Juntan Zhang, M. Fossorier, Daqing Gu, and Jinyun Zhang. Improved min-sum decoding of LDPC codes using 2-dimensional normalization. In *GLOBECOM '05. IEEE Global Telecommunications Conference, 2005.*, volume 3, pages 6 pp.–, 2005.

[62] T. J. Richardson and R. L. Urbanke. The capacity of low-density parity-check codes under message-passing decoding. *IEEE Transactions on Information Theory*, 47(2):599–618, 2001.

[63] Sae-Young Chung, T. J. Richardson, and R. L. Urbanke. Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation. *IEEE Transactions on Information Theory*, 47(2):657–670, 2001.