

ulm university universität
uulm

Enhancing Reliability and Efficiency in Flash Memory: Readout Methods and Error Correction Algorithms

DISSERTATION

zur Erlangung des akademischen Grades eines

DOKTOR-INGENIEURS

(Dr.-Ing.)

der Fakultät für Ingenieurwissenschaften,
Informatik und Psychologie der Universität Ulm

von

Daniel Nicolas Bailon
aus Singen (Hohentwiel)

Gutachter:	Prof. Dr.-Ing. Jürgen Freudenberger
	Prof. Dr. Stefan ten Brink
	Prof. Dr.-Ing. Martin Bossert
Amtierende Dekanin:	Prof. Dr. Anke Huckauf

Ulm, 10.06.2024

Version from: August 28, 2024

© 2024 Daniel Nicolas Bailon

Teile dieser Dissertation wurden bereits in folgenden Fachartikeln veröffentlicht:

[1] J.-P. Thiers, D. Nicolas Bailon, and J. Freudenberger, "Bit-labeling and page capacities of TLC non-volatile flash memories," in *2020 IEEE 10th International Conference on Consumer Electronics (ICCE-Berlin)*, nov 2020, ©2021 IEEE.
https://doi.org/10.1109/icce-berlin50680.2020.9352190.

[2] D. Nicolas Bailon, J.-P. Thiers, and J. Freudenberger, "Error correction for TLC and QLC NAND flash memories using cell-wise encoding," *Electronics*, vol. 11, no. 10, p. 1585, may 2022. CC BY 4.0, https://creativecommons.org/licenses/by/4.0/
https://doi.org/10.3390/electronics11101585.

[3] J.-P. Thiers, D. Nicolas Bailon, J. Freudenberger, and J. Lu, "Read reference calibration and tracking for non-volatile flash memories," *Electronics*, vol. 10, no. 18, p. 2306, sep 2021. CC BY 4.0, https://creativecommons.org/licenses/by/4.0/
https://doi.org/10.3390/electronics10182306.

[4] D. Nicolas Bailon, S. Shavgulidze, and J. Freudenberger, "Cell-wise encoding and decoding for TLC flash memories," in *2022 IEEE 12th International Conference on Consumer Electronics (ICCE-Berlin)*, sep 2022, ©2022 IEEE.
https://doi.org/10.1109/icce-berlin56473.2022.9937136.

[5] M. Safieh, D. Nicolas Bailon, and J. Freudenberger, "An acceptance criterion for hybrid algebraic and soft-input decoding," in *2020 24th International Conference on Information Technology (IT)*, feb 2020, ©2020 IEEE.
https://doi.org/10.1109/it48810.2020.9070575.

[6] D. Nicolas Bailon, G. Taburet, S. Shavgulidze, and J. Freudenberger, "Neural network aided reference voltage adaptation for NAND flash memory," in *2022 IEEE 12th International Conference on Consumer Electronics (ICCE-Berlin)*, sep 2022, ©2022 IEEE.
https://doi.org/10.1109/icce-berlin56473.2022.9937118.

[7] D. Nicolas Bailon, S. Shavgulidze, and J. Freudenberger, "Read reference voltage adaptation for NAND flash memories with neural networks based on sparse histograms," *IEEE Access*, vol. 11, pp. 56 801–56 811, 2023. CC BY 4.0, https://creativecommons.org/licenses/by/4.0/
https://doi.org/10.1109/access.2023.3283445.

[8] D. Nicolas Bailon, S. Shavgulidze, and J. Freudenberger, "Features for read reference voltage estimation in NAND flash memory," in *2023 IEEE 13th International Conference on Consumer Electronics (ICCE-Berlin).*, sep 2023, ©2023 IEEE.
https://doi.org/10.1109/ICCE-Berlin58801.2023.10375586.

[9] J. Freudenberger, D. Nicolas Bailon, and M. Safieh, "Reduced complexity hard- and soft-input BCH decoding with applications in concatenated codes," *IET Circuits, Devices Systems*,

vol. 15, no. 3, pp. 284–296, mar 2021. CC BY 4.0, https://creativecommons.org/licenses/by/4.0/ https://doi.org/10.1049/cds2.12026.

[10] D. Nicolas Bailon and J. Freudenberger, "List decoding for concatenated codes based on the Plotkin construction with BCH component codes," in *2021 11th International Symposium on Topics in Coding (ISTC)*, aug 2021, ©2021 IEEE.
https://doi.org/10.1109/istc49272.2021.9594194.

[11] D. Nicolas Bailon, M. Bossert, J.-P. Thiers, and J. Freudenberger, "Concatenated codes based on the Plotkin construction and their soft-input decoding," *IEEE Transactions on Communications*, vol. 70, no. 5, pp. 2939–2950, may 2022, ©2022 IEEE.
https://doi.org/10.1109/tcomm.2022.3158329.

[12] D. Nicolas Bailon, J.-P. Thiers, and J. Freudenberger, "Soft-input decoding of concatenated codes based on the Plotkin construction and BCH component codes," in *2020 IEEE 10th International Conference on Consumer Electronics (ICCE-Berlin)*, nov 2020, ©2020 IEEE.
https://doi.org/10.1109/icce-berlin50680.2020.9352179.

Acknowledgements

First and foremost, I would like to express my gratitude to my mentor and supervisor, Jürgen Freudenberger, for granting me the invaluable opportunity to pursue my Ph.D. at the Institute of System Dynamics (ISD) over the past years. I am profoundly grateful for his support, excellent advice, inspiration, and numerous helpful discussions, which significantly contributed to the development of my dissertation and enhanced the quality of my research. Without his guidance and support, this work would not have been possible.

Special thanks go to Stefan ten Brink and Martin Bossert for taking the time to review this dissertation as well as for the colloquium.

I extend my sincere appreciation to my colleagues at the ISD for their engaging discussions, unfailing helpfulness, and valuable advice. Special acknowledgment goes to Johann-Philip Thiers, who accompanied me for a significant part of this journey and served as a great motivator. My thanks also go to Sergo Shavgulidze, with whom I had the privilege of having various helpful discussions.

Lastly, my heartfelt thanks go to my family and friends for their great support. Special gratitude goes to my life partner, who has been a constant pillar of support and encouragement throughout this journey.

Contents

Acknowledgements **V**

Abstract **XI**

Acronyms **XV**

1 Introduction **1**

2 Flash Memory Basics **5**
2.1 Flash cell 5
2.2 Architectures and Technologies 7
2.3 Description of Error Sources 10

3 Flash Channel Models and Measurements **13**
3.1 Approximation with Gaussian Noise 13
3.2 Measured Threshold Voltage Distributions 15
3.3 Model Approximation from Measured Voltage Distributions 16

4 Bit-Labeling and Capacity Considerations **19**
4.1 Bit-Labeling for Flash Memory 20
4.2 Channel Capacity of Flash Memories 25
4.3 Capacity on End-of-Life 27
4.4 Capacity Comparison of Page-Wise and Joint Readout 29
4.5 Summary 30

5 Read Reference Voltage Adaptation **31**
5.1 Readout Methods 32
5.2 Features for Read Reference Voltage Estimation 33
5.3 Read Reference Voltage Adaptation Algorithms 43
5.4 Reference Voltage Adaptation with Neural Networks 51
5.5 Summary 66

6 Fundamentals of Error Correction Procedures **69**
6.1 Preliminaries 70
6.2 Reed-Muller Codes 72
6.3 Bose-Chaudhuri-Hocquenghem Codes 75

6.4 LDPC Codes for Flash Memory 79

7 Soft-Decision Decoding of Short Cyclic Codes 81
7.1 Repetition and Single Parity-Check Codes 81
7.2 Chase Decoding 82
7.3 Ordered Statistic Decoding 83
7.4 Acceptance Criteria for Soft-Input Decoding 84
7.5 Hybrid Ordered Statistics Decoding 88
7.6 Test Pattern Reduction 93

8 Code Concatenation and Complexity Reduction 99
8.1 Plotkin Concatenation with BCH Component Codes 99
8.2 Decoding of Plotkin Concatenated Codes 102
8.3 Simulation and Complexity Results of Plotkin Codes 106
8.4 Generalized Concatenated Codes 113
8.5 Complexity Reduction of GC Decoder 118
8.6 Summary 123

9 Error Correction using Cell-Wise Processing 125
9.1 GC Codes for Cell-Wise Processing 126
9.2 Performance using GC Codes 129
9.3 LDPC Codes for Cell-Wise Processing 133
9.4 Summary 136

10 Conclusion 139

Bibliography XVII

Abstract

Flash memory is essential in modern electronics due to its fast access, high storage density, and cost-effectiveness. As the demand for expanded local storage capacity continues, its appliance is increasing.

This work primarily focuses on enhancing the reliability of flash memories. It begins with a comprehensive characterization of the flash channel, identifying and analyzing various sources of errors. The study delves into different bit-labeling schemes and investigates the achievable capacities associated with them. Additionally, the importance of read reference voltages is explained, particularly in adapting them to the life-cycle condition. The thesis also introduces calibration and adaptation algorithms for this purpose.

The challenges related to error correction codes are addressed extensively, focusing on algorithms designed to reduce decoding complexity. The research delves into low-complexity decoding techniques, particularly for scenarios involving small code lengths. Another area of investigation is concatenated codes based on small cyclic codes.

Furthermore, the thesis explores the advantages of joint processing of NAND flash pages, highlighting improvements in hard-decision decoding to minimize the need for additional read-out operations. This joint processing approach is thoroughly compared with conventional processing methods to assess its effectiveness and potential benefits.

Overall, this thesis contributes to enhancing the reliability of flash memories while proposing optimizations for their decoding processes.

Acronyms

ASK amplitude shift keying.

AWGN additive white Gaussian noise.

BCH Bose-Chaudhuri-Hocquenghem.

BER bit error rate.

BMA Berlekamp-Massey algorithm.

BOP binary operation.

BPSK binary phase shift keying.

CSB center significant bit.

CT charge trap.

DA data augmentation.

DRT data-retention time.

ECC error correction code.

EMG exponentially modified Gaussian.

EOL end-of-life.

FER frame error rate.

FG floating gate.

FLOP floating point operation.

FNN feedforward neural network.

GC generalized concatenated.

GMC generalized multiple concatenation.

GMM Gaussian mixture model.

HD hard-decision.

HOSD hybrid ordered statistics decoding.

LDPC Low-Density Parity-Check.

LLR log-likelihood ratio.

LM Levenberg-Marquardt.

LPD list Plotkin decoding.

LSB least significant bit.

MI mutual information.

ML maximum-likelihood.

MLC multi-level cell.

MMSE minimum mean square error.

MSB most significant bit.

NL Normal-Laplace.

NN neural network.

OSD ordered statistics decoding.

P/E program/erase.

PD Plotkin decoding.

PLC penta-level cell.

PTMP program temperature.

QLC quad-level cell.

RAM random-access memory.

RBER residual bit error rate.

RD read disturb.

RM Reed-Muller.

RRV read reference voltage.

RS Reed-Solomon.

RTMP read temperature.

SB soft bit.

SC successive cancellation.

SCL successive cancellation list.

SD soft-decision.

SLC single-level cell.

SNR signal-to-noise ratio.

SPC single parity-check.

TLC triple-level cell.

TS training set.

TVD threshold voltage distribution.

Chapter 1

Introduction

The ongoing digitalization has resulted in significant shifts in resource requirements. The demand for information and communication technologies has escalated, particularly in terms of transmitting and storing large volumes of data reliably and efficiently. Fast evaluation, robust error-free storage, and efficient data transmission have become increasingly critical. NAND flash memories meet these criteria due to their absence of mechanical parts, their low power consumption, fast access times, and high storage capacity.

The need for faster storage technologies has consequently led to a higher demand for flash memory storage, outpacing the capabilities of traditional options like hard disk drives. Flash controllers play an important role in efficiently reading and writing data on the flash cells while addressing different interfaces and processing data according to specifications. These controllers are very complex and contribute to technological progress.

According to Moore's Law, the number of transistors per unit area on integrated circuits doubles approximately every two years. In the context of NAND flash memory, this phenomenon implies a corresponding increase in memory density at a similar rate [13]. Therefore, flash memory is the focus of cost-effective storage solutions for large data volumes with high data density.

However, this presents challenges in meeting reliability, durability, and error-free requirements [14]. The ultimate objective is to ensure error-free and dependable data transfer throughout the entire lifespan of flash data storage devices. As a result, various algorithms and methods must be integrated into flash controllers to address these challenges effectively. This highlights the critical significance of error correction methods and ongoing research in the area of signal processing.

In the field of signal processing, various topics belonging to transmission systems are under research. Figure 1.1 illustrates a simplified transmission system for reference. Flash memory can be seamlessly integrated into a transmission system.

The source represents the data provided to the flash controller from a device, where the flash controller is responsible for encrypting, encoding, and modulating the data. The channel denotes the physical flash memory, and the flash controller components handle the control of data storage and recall within the memory. Channel disturbances may arise due to physical

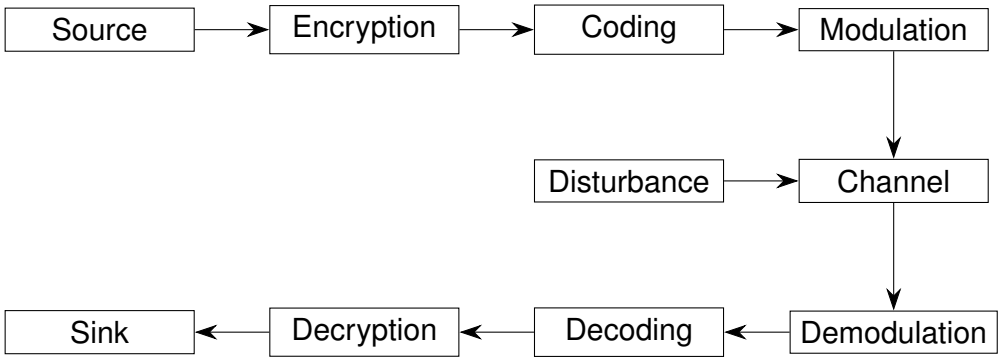


Figure 1.1: General simplified transmission system.

alterations or aging effects. Furthermore, the flash controller is also tasked with demodulating, decrypting, and decoding the data. In the diagram, the sink represents the output of data from the flash controller. For storage devices, reliability must rely exclusively on error correction codes (ECCs) since there is no feedback channel.

Numerous algorithms and sequences are required to address all of the mentioned blocks comprehensively. However, attempting to cover all of them would exceed the scope of this work. Consequently, we will focus on two specific aspects and their interface, namely the readout from the flash channel and error correction mechanisms.

Outline of the Thesis

The first chapters of this thesis highlight the core structure, functionality, and inherent error sources associated with flash memory. Subsequently, we discuss different channel models that are essential for understanding the characteristics of data transfer.

A pivotal element in the discourse of flash memory is the readout concept. As technology has advanced, contemporary NAND flash memories are capable of storing multiple bits per cell, leading to different charge states, as noted in [14, 15]. Our primary focus in this research is placed on triple-level cell (TLC) flash memories, characterized by their capacity to store three bits per flash cell. A well-designed bit-labeling is the basis for the readout methods. Bit-labelings can modify channel capacities depending on the readout type, directly influencing the error correction capability, as discussed by Parnell [16]. Different types of bit-labeling for flash memory are discussed in Chapter 4.

Following this, Chapter 5 explores improvements of the readout accuracy. Given different error sources, the increasing density of memory volumes, smaller form factors, and manufacturing process tolerances, the flash memory channel experiences significant shifts throughout its operational lifetime. Due to the many life-cycle conditions, a universally applicable model becomes non-trivial and is currently unavailable. Given these dynamics and the expected longevity of flash memory, refining and adapting the reading process is critical to minimizing the probability of residual errors.

Even under perfect conditions, an error-free readout of the data is not feasible. Therefore,

integrating robust error correction mechanisms is necessary to maintain the reliability of the data. In flash memory, we primarily distinguish two data streams, namely payload and metadata. These data streams have distinct characteristics requiring special treatment and tailored error correction [17, 18, 19, 14, 20]. Thus, different ECC are needed to address these varying characteristics effectively. Preliminaries on select ECC are elaborated in Chapter 6, setting the stage for Chapter 7, which delves into decoding strategies for short cyclic codes. These subjects are necessary in the subsequent chapters.

Chapter 8 describes concatenated codes incorporating short cyclic codes as components. At this point, we generate codes that are especially suitable for metadata. Short codes with high error correction capability are defining characteristics. Our objective is to enable the utilization of soft decoding to enhance reliability. The primary emphasis lies on achieving near-optimal performance with minimal complexity to ensure efficient decoding.

Subsequently, Chapter 9 considers possible error correction strategies for payload data. Therefore, we explore the cell-wise processing strategy, which maximizes the readout potential of flash memory. This method is compared to the conventional page-wise process, whereby a comparative performance analysis of these strategies using exemplary error-correcting methods completes this chapter.

Finally, Chapter 10 summarizes the main aspects of this thesis.

Chapter 2

Flash Memory Basics

Flash memories comprise many floating gates (FG) or charge trap (CT) MOSFET (metal oxide semiconductor field effect transistor) elements, commonly known as flash cells. These memory cells possess the capability to rapidly store, read, and erase data, which is why they are referred to as "flash" memory. The flash memory technology was invented by Dr. Fujio Masuoko of Toshiba in 1984. This technology is built to retain digital data for extended periods without requiring a power supply, making it a non-volatile storage medium. Data is stored as an electrical charge on the flash cell. The first storage medium based on this technology was produced in 1988. Flash memory has experienced remarkable evolution since then, marked by reductions in manufacturing costs and a significant increase in storage capacity. These advancements have elevated flash memory to play a pivotal role in data storage, making it an indispensable technology of countless electronic devices. Unfortunately, this technology is associated with significant challenges, as flash cells introduce several error sources.

This chapter introduces the key aspects of data storage in a flash cell, explains the underlying flash technologies and architectures in this thesis, and provides an overview of selected error sources. Parts of this chapter have been published in [1, 2].

2.1 Flash cell

As previously mentioned, there are essentially two underlying technologies used as flash cells: FG and CT MOSFET. The charge on an FG MOSFET is saved on a polysilicon layer located between the gate and the channel. NAND flash memory based on CT technology follows a different design and stores the charge on a silicon nitride film. The primary difference between FG and CT technology lies in the materials used in the layers between the control gate and the substrate, as well as between the drain and source. This difference in materials affects conductivity and consequently alters physical properties of the memory [14].

In the following, we will focus on the widely used flash cell based on FG technology, as depicted in Figure 2.1. This FG-based flash cell enables three main operations: writing, erasing and reading data. Essentially, the FG is completely isolated within the flash cell,

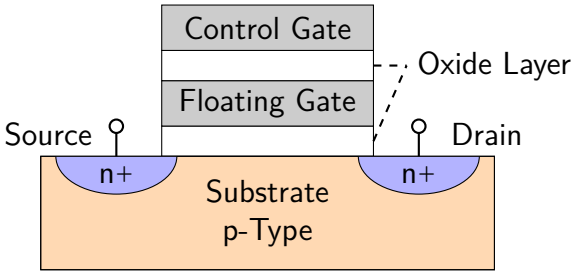


Figure 2.1: Schematic representation of a FG MOSFET.

without any electrical connection to the rest of the cell. The principles of FG technology are comprehensively described in the publication by Kahng [21].

- Write/program: Writing on an FG MOSFET involves the transfer of electrons from the substrate to the FG, effectively storing information as an electrical charge. This process relies on a quantum effect called Fowler-Nordheim tunneling, first discovered in 1928 [22]. This phenomenon describes the probabilistic behavior of particles crossing a barrier. In a flash cell, this phenomenon is enabled when the source is connected to ground potential, a charging voltage is applied to the control gate, and a positive voltage to the drain input. Typical necessary voltages are 9 V on the gate, 4.5 V on the drain, and ground potential on the source input [13].

  The charging process of the cell is precisely controlled in time to ensure that the FG acquires the desired charge. An example of such control is the Program Verify method, which is one feasible implementation [13]. The charging process increases the charge of the FG, which increases the threshold voltage V_{th} of the FG MOSFET.

- Read: To read the data, a read voltage is applied to the gate, while the source is connected to ground, and a read voltage is connected to the drain. This allows the flow of electric current between the drain and the source. An electric charge in the FG changes the required threshold voltage V_{th} that needs to be applied to the gate to activate the FG MOSFET.

  To determine the charge status of a cell, a read reference voltage (RRV) V_r is applied, and the resulting current is subsequently measured. If current is detected, the FG remains uncharged, which represents a logical 1, characterized by a low V_{th}. Conversely, the absence of a detectable current corresponds to a logical 0, associated with a high V_{th}). The RRV is chosen to enable optimal distinction between these two scenarios. For this purpose, the distribution of the threshold voltages in relation to the charge states is investigated. This elementary case corresponds to the single-level cell (SLC).

  Modern flash technologies have evolved to store multiple bits per cell, i.e., they can distinguish multiple charge states, denoted in this work as S. During the readout process, multiple RRVs are necessary to distinguish between the possible charge states. The query whether a current is flowing or not remains a fundamental requirement. Consequently, a

flash cell with S states must be distinguished using a maximum of $S-1$ read voltages.

Thus, the multi-level cell (MLC) is capable of storing 2 bits, i.e. distinguishing $2^2 = 4$ states. The next extension is the triple-level cell (TLC), which can store 3 bits per cell, enabling $2^3 = 8$ distinct memory states. TLCs are widely adopted in today's flash memory landscape. A quad-level cell (QLC) accommodates 4 bits per cell, allowing $2^4 = 16$ memory states [14]. QLC flash is already commercially available, and further advancements such as penta-level cell (PLC) are currently under development [15].

- Erase: Erasing the data from a flash cell involves discharging the FG. The discharge is enabled through the Fowler-Nordheim tunneling effect, similar to the writing process. For discharging the FG, an electric field opposite to that used during writing needs to be applied. This removes the electrons from the FG.

  However, as high negative voltages are often avoided in practice, the gate can be connected to the ground potential, while a high voltage is applied between the source and drain, which generates a negative electric field. Typically, the threshold voltage V_{th} of an erased cell typically falls below the ground potential, which means that an erased cell can have a negative threshold voltage.

  In the case of cells that have more than two different states, the writing and reading of the cell changes. The cells are always completely discharged when they are deleted.

In order to perform this operations, additional circuits are implemented. Therefore, the next section introduces the most common architectures with flash cells.

2.2 Architectures and Technologies

Flash cells are connected in series or parallel, which results in two architectures: NOR or NAND flash. The term "NOR" or "NAND" is derived from the arrangement of the cells within each architecture. In NOR flash, each individual cell (e.g. FG MOSFET) is connected in parallel, meaning that the source terminal of each cell is individually connected to a common ground, while the drain terminal is connected to the output. The parallel connection of cells ensures that they operate independently without mutual interference. This independent connection enables individual control over each cell, allowing individual manipulation and management of each cell.

By contrast, the NAND architecture organizes the basic cells in series, such that multiple cells are connected to a common ground and output line. This is called a NAND string. To perform a read procedure on a NAND flash cell, it is necessary to activate all other cells in the string. This is achieved by applying a throughput voltage to the gate of each cell to ensure that they are switched on. To facilitate the control, transistors are used at the beginning and end of each string, connecting them to the so-called source line and bit line, respectively. The bit line is connected through other NAND strings, while the gates of the flash cells are interconnected

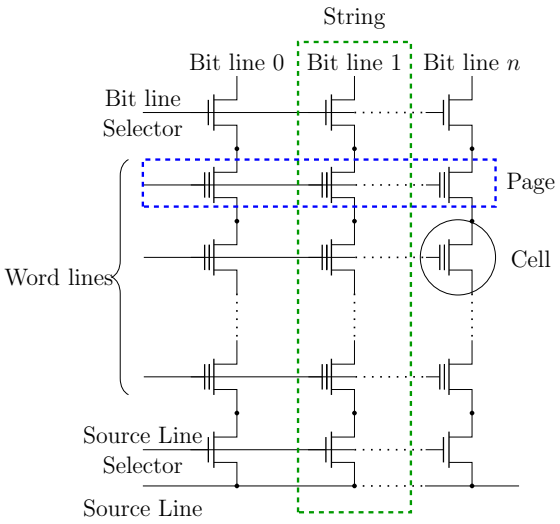


Figure 2.2: Simplified schematic of a NAND flash architecture.

through dedicated word lines. NAND flash memories are organized hierarchically. Figure 2.2 presents a simplified schematic interconnection of the flash cells in a NAND flash block. Please note that components are missing and additional readout design considerations are necessary for cells that store more than one bit.

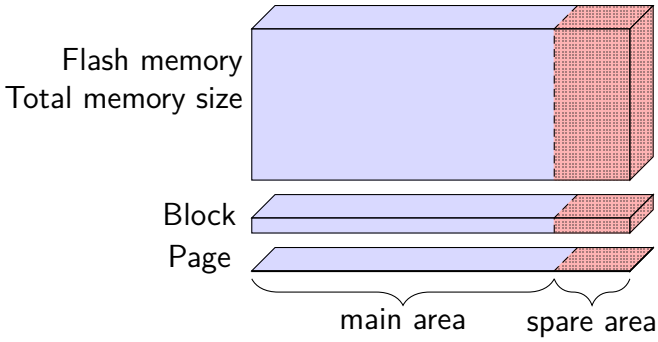


Figure 2.3: Illustration of the NAND flash memory organization hierarchy.

Next we describe the hierarchical organization of a NAND flash memory. A NAND flash die comprises one or more planes, each containing multiple blocks. Each block in turn comprises multiple pages. The smallest erasable unit is a block, while the page is the smallest individual readable and programmable unit in NAND flash. Multiple planes within a die are only employed in modern flash memories, capable of parallel data read and write operations. A page is typically divided into a main area and spare area. The main area contains the payload data and the spare area is allocated for error correction, system pointers and firmware metadata [14]. NAND flash memories are constructed such that the spare area used for the error correction and metadata has a defined size, and each page typically has the same spare area [23]. Thus, using page-wise encoding implies that each page uses the same code rate. For a better understanding,

Figure 2.3 illustrates the hierarchical organization of a NAND flash memory.

The primary advantage of NOR flash is its fast data reading capability, although it is comparatively slower in writing and erasing data compared to NAND flash. Due to the structure of NOR flash, each cell has a connection to a powerline and ground, which leads to a large amount of connections. On the other hand, NAND flash offers the benefit of processing larger amounts of data at once, resulting in increased erase and write speeds. Additionally, since the cells are not individually accessed, the requirement for extensive wiring is eliminated, allowing for smaller structures. Consequently, this enables higher cell density, leading to reduced size, power consumption and costs per cell.

NAND flash is particularly optimized for cost and size, thanks to its denser memory architecture. As a result, NAND flash memory devices find application in a broader range of applications compared to NOR devices. The disadvantage of NAND flash is the extended read time of a flash cell and additional interference issues, which are explained later. NOR flash is primarily used in scenarios where random read operations are essential, such as small firmware memories in embedded devices or critical applications. At present, NAND flash cells are mainly implemented in commercial applications due to their ability to serve as low-cost bulk storage devices [24]. Therefore, this work specifically concentrates on NAND flash memories, as they represent the most prevalent architecture.

The planar NAND architectures are being increasingly replaced by 3D NAND architectures since the storage density is increased, i.e., multiple layers are stacked vertically. Other advantages are lower power consumption and faster read and write speeds [14, 25]. The current technologies are Bit-Cost Scalable (BiCS) technologies first introduced in [26] and the Terabit Cell Array Transistor (TCAT) introduced in [27].

When using a flash cell that can store more than one bit, each bit of the cell is assigned to different pages and different codewords of the ECC. This page-wise encoding minimizes the read latency with hard-input decoding. Depending on the bit-labeling, a different number of read operations is needed for the different pages. Accordingly, the read latency depends on the bit-labeling.

Considering exemplary TLC NAND flash memories, each cell stores three bits, and eight possible charge states are denoted as S_i. Each bit is assigned to a specific page, which is labeled as the most significant bit (MSB) page, the center significant bit (CSB) page, and the least significant bit (LSB) page, based on the associated bit. In the case of MLC flash, there are only two pages: the MSB page and the LSB page. However, for QLC flash, the pages are referred in this thesis as pages 1, 2, 3, and 4.

Depending on the bit mapping strategy, the number of reference voltages required to read a bit of a page varies. The additional extraction of soft information amplifies the total number of readouts per cell depending on the desired resolution, i.e., the number of soft bits (SBs). This information augmentation consequently increases the latency and power consumption. However, this approach increases the channel capacity, and it is more probable to avoid decoding failures and data loss within the flash memory.

The bit-labeling of available flash memories is usually based on Gray codes [28]. Gray codes are binary codes where the codewords of adjacent bit patterns only differ by a single bit position. Such a bit mapping scheme minimizes latency for page-wise random access. However, Gray codes do not consistently achieve an ideal balance between error probability and total cell capacity [1]. Conversely, a multilevel coding strategy generally requires joint processing of all pages, impairs the random read performance but maximizes the cell capacity.

Table 2.1: Gray code bit-labeling for TLC flash cell used in this work (adapted from [7], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/).

page	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
MSB	1	1	1	0	0	0	0	1
CSB	1	1	0	0	1	1	0	0
LSB	1	0	0	0	0	1	1	1

We consider the code presented in Table 2.1 as an exemplary Gray code for TLC flash memory. In this code, both the MSB and LSB pages require two reference voltages, while the CSB page needs three reference voltages to determine the corresponding bit. This labeling supports a page-wise read operation, allowing the independent reading of the LSB, CSB, and MSB pages. However, this labeling does not necessarily balance the error probabilities on the different pages. Profound discussion on bit-labeling is part of Chapter 4.

2.3 Description of Error Sources

Different life-cycle states are reached throughout the flash memory's operating time. The reliable operation of the entire memory strongly relies on the tunnel oxide's ability to withstand the demanding condition of program and erasure. As a result, extensive efforts are dedicated to determining the ideal process for creating the tunnel oxide, considering factors like thickness, material, growth, defectivity, interface, and others. Additionally, practical algorithms are crucial in achieving successful and dependable NAND technology. In the following, some of the most challenging effects that change the physical properties of the flash cell are described in more detail.

- Program/erase (P/E) cycling: The programming and erasing of data on the flash cells gradually damage the insulating layer of the FG, leading to changes in the physical properties of the semiconductor. The number of P/E cycles is the primary property to describe the aging of the flash memory, such that it is an equivalent of the lifetime. Manufacturers specify a maximal number of P/E cycles for the NAND flash memories produced, ensuring a high probability of avoiding uncorrectable errors.

  For SLC, the number of P/E cycles is in the order of 10^5 cycles, and for MLC this number is reduced to 10^4 [23]. These numbers may vary for consumer devices and specific

deployment for enterprise solutions. Other errors may occur when programming/writing or easing data on the cell. These errors may involve failing to reach the desired charge level or the erase state properly. Such errors emerge after repeated P/E cycles due to trapped electrons in the tunnel oxide. This phenomenon is correspondingly called gain degradation and charge trap [13].

- Data-retention time (DRT): Even though the FG is physically isolated, the FG will discharge over time. One source of error is so-called *trap-assisted tunneling* [29, 30, 31]. This phenomenon occurs when the trapped charge within the flash cell creates an electrical tunnel that amplifies the weak tunnel current, which facilitates the charge leakage from the FG. The tunneling effect causes an accelerated discharge due to the internal electric field and reduces the threshold voltage. In addition, this effect is enhanced by the damage to the insulating layer caused by the P/E cycles. A degraded insulating layer causes a faster discharge.

  A critical factor for DRT is the thickness of the insulating layer. A thicker insulating layer reduces the probability of charge leakage. However, this also requires a higher electric field strength to write data effectively, which implies higher energy consumption. On the other hand, if the layer is too thin, it is more vulnerable to damage from writing operations, accelerating its degradation over time and contributing to increased discharge [14]. Flash memory manufacturers guarantee a DRT for the produced NAND flash memories.

- Charge trapping/detrapping: Another physical effect is so-called *charge trapping/detrapping* whereby random positive or negative charges located in the tunnel oxide affect the FG after a time [29, 30, 32, 33]. This effect can increase or decrease the charge level on the FG, causing the threshold voltage to change accordingly.

- Read disturb (RD) / write disturb (WD): A persistent source of interference in NAND flash is the RD effect [14]. Reading a page several times without erasing may cause a charge increase in the cells on the affected pages. The probability of RD increases with the number of P/E cycles, and it is more likely on already damaged cells. Erasing the cells resets this disturbance, as this procedure discharges the cells completely [14]. More details are available in [34, 35] and for specific insights related to 3D NAND flash in [36].

- Temperature effects: Temperature instabilities constitute a significant concern of CMOS circuit reliability [23]. As a result, the temperature may also affect flash devices. A cross-temperature effect arises due to varying or extreme program temperature (PTMP) and read temperature (RTMP) conditions. Different temperatures cause variations of the electrical properties, i.e., the charge distributions of the flash cell are changed. Consequently, significant temperature disparities can lead to critical errors when reading the information. This issue is particularly problematic in cross-temperature scenarios, i.e., writing at one temperature and reading at another. In such cases, the threshold voltage distributions (TVDs) can shift massively.

- Gamma and alpha rays: Radiation problems are frequent in particular environments, such as aircraft, space, or radioactive hazard zones. Gamma and alpha rays can change the charge in the flash memory, resulting in incorrect readouts [23].

The most important properties of a flash memory device lie in its endurance and data retention capabilities. Endurance represents the minimum number of P/E cycles that the memory can endure before encountering significant failures. Data retention refers to the duration for which the memory device can retain data without an external power supply.

All of the effects described in this chapter hold significant importance for the effective management of flash memory. For the appropriate configuration of signal processing from flash memory, it becomes crucial to have access to measurements and accurate models of the communication channel. The next chapter describes the models and measurements used in this thesis. The measurements of the TLC NAND flash memory used in this thesis were provided by the company Hyperstone GmbH (Konstanz, Germany).

Chapter 3

Flash Channel Models and Measurements

The previous chapter described numerous influences on the data stored in flash memory. A precise communication channel description is crucial to appropriately select the signal processing and error correction methods. According to the pioneer of information theory, C. Shannon, the channel provides the scope for the successful transmission of information.

It is challenging to model the physical flash channel precisely. The different error sources in flash memory arise from diverse physical effects, which increase the challenges in modeling the flash channel [37]. Numerous publications deal with this challenges [38, 39, 40, 41, 42, 43]. The influence of the physical effects also depends on the flash type and the manufacturing procedure. Due to the increasingly smaller designs for flash memory, the physical structure and attributes are constantly evolving. Consequently, excessive measurements across various life-cycle states covering flash memories from different distributors become necessary to accurately estimate the channel conditions. This task is demanding and involves a substantial workload.

In this chapter, we will take a closer look at flash channel models and present measurements from flash memories that we will use in the subsequent chapters. We consider two possibilities to approximate the flash channel. Initially, we consider a simplified flash channel approximation, employing amplitude shift keying (ASK) as the modulation scheme and incorporating additive white Gaussian noise (AWGN) as the noise component. Afterwards, we verify the deviations between the idealized model and the measurements of the flash memory. Subsequently, we construct a simplified model to illustrate the fundamental behavior utilizing the measurements from flash memory. However, due to the extensive scope of the problem and the absence of comprehensive coverage across all possible life-cycle states, the applicability of this model is limited. Parts of this chapter have been published in [1, 2, 3].

3.1 Approximation with Gaussian Noise

The threshold voltage is the minimum voltage required to trigger a change in the state of a memory cell. In flash memory, the threshold voltage is used as a reference point to determine

the stored electrical charge. Due to factors such as manufacturing variations, process defects, and error sources (cf. Section 2.3), threshold voltage values differ among flash memory cells. A threshold voltage distribution (TVD) refers to the statistical distribution of threshold voltage values in a group of flash memory cells. The value stored in a flash can be accurately determined using a large scale of threshold voltage measurements. Subsequently, the TVD can be analyzed to gain insights into the characteristics of the flash memory.

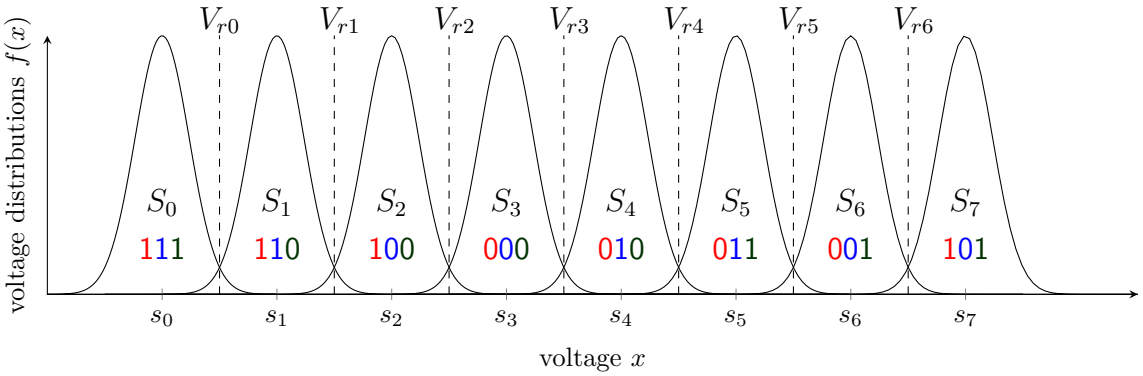


Figure 3.1: TLC channel as a uniform-spaced M-ASK model with equally distributed AWGN, labeled reference voltages V_{th}, and exemplary bit-labeling. (adapted from [3], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/).

Given that the amount of electrical charge corresponds to the information stored in a flash cell, the ASK modulation scheme theoretically approximates the charge distribution. This concept applies to a SLC flash memory with binary ASK (BASK) modulation. In general, M-ASK modulation is applied in flash memory types, where M denotes the number of bits encoded into a single cell. Typically, the ASK modulation is bipolar. However, given that the charges on the flash memory are strictly non-negative, a unipolar ASK modulation is prescribed. In the case of a TLC flash memory, which can store three bits per cell, there are $2^3 = 8$ possible charge states, resulting in a unipolar 8-ASK model.

In the context of the individual charge distributions, the most intuitive approach assumes the widely employed AWGN distribution. Figure 3.1 shows exemplarily the TVD for TLC flash memory using ASK modulation and AWGN error distribution. The AWGN model is based upon the presumption of statistical independence between charge and noise components, coupled with a normal distribution for additive noise. However, these assumptions lack applicability to the majority of flash memories. The physical properties of the flash cell combined with various noise sources, as mentioned in section 2.3, introduce non-Gaussian characteristics into the system. Consequently, the accurate depiction of noise behavior in these systems requires alternative models or more sophisticated noise characterization techniques. The advantage of AWGN modeling lies in its compatibility with standard simulation tools and established calculation formulas. In Chapter 4, this channel model is employed to compute theoretical error probabilities and analyze theoretical channel capacity. The subsequent section will detail

the flash memory measurements, which proves that AWGN modeling is insufficient in terms of providing an accurate description.

3.2 Measured Threshold Voltage Distributions

In the following, we discuss TVD measurements conducted on TLC flash memories. Given that a TLC flash involves eight different charge levels, namely $S_0, .., .S_7$, it requires seven defined reference voltages, $V_{r0}, ..., V_{r6}$, to effectively distinguish all three bits. The three bits of each cell are typically mapped to three different pages called MSB, CSB, and LSB page, respectively, as detailed in Chapter 2.

Figure 3.1 illustrates the conditional distributions $f(V_{th}|S)$ of the threshold voltages V_{th} of the flash cells given the charge levels. The figure also provides a possible bit-labeling for the charge levels, where the colors indicate the corresponding page, i.e., MSB in red, CSB in blue, and LSB in green. Most flash memories support page-wise read operations. To read the MSB page with the bit-labeling in Figure 3.1, for instance, we need to read at the reference voltages V_{r2} and V_{r6}.

The distributions $f(V_{th} \mid S)$ are not static but change over the lifetime of the flash. Two of the most important noise factors for these alterations are the number of P/E cycles and the DRT (cf. Section 2.3). The P/E cycles lead to the wear-out effect of the FG, while the DRT leads to a charge loss. The distributions also depend on effects like the program and RTMP and the charges of adjacent cells, i.e., cell-to-cell interference and much more. Moreover, they can vary over pages and blocks.

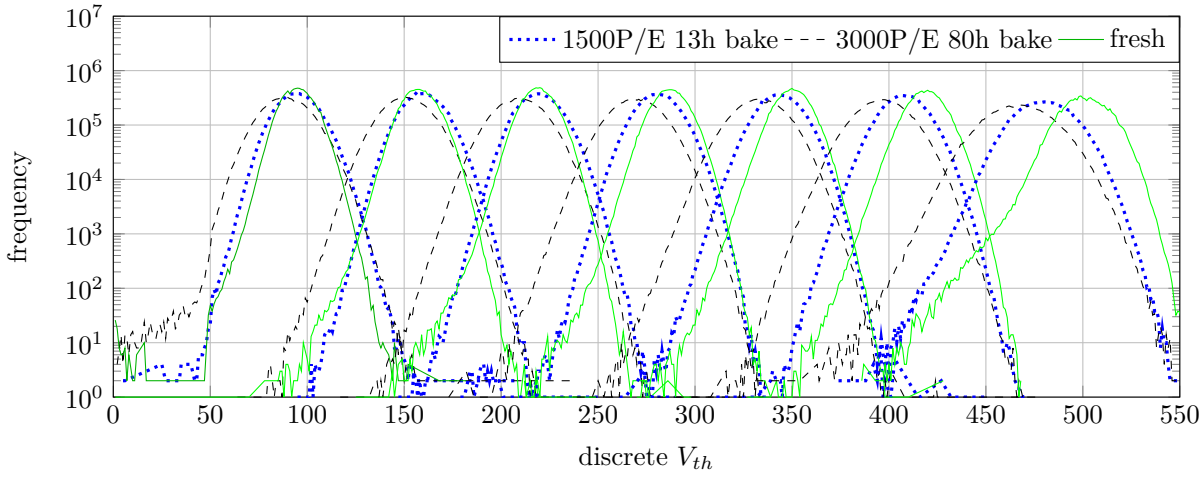


Figure 3.2: Measurements of TLC flash memory in different life-cycle states (adapted from [3], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/).

Figure 3.2 shows the charge loss due to DRT as well as the widening of the distributions due to wear-out. Due to the logarithmic scale of the y-axis, it is apparent that the distributions are not

symmetric, and the voltages are not normally distributed. While the TVDs are often modeled as Gaussian distributions, the measured distributions show an exponential tail toward the lower voltages. To cope with this exponential tail, the exponentially-modified Gaussian distribution was proposed in [44], which is the sum of two independent normal and exponential random variables. A Normal-Laplace (NL) mixture model was proposed in [43]. Similarly, the model in [45] copes with program errors in MLC flash memory. All three bit pages are programmed simultaneously in our measurements, and thus no write errors occur.

The DRT is simulated for the measurements by subjecting the chip to a baking process for a specific duration [46, 47]. Figure 3.2 shows the measured histograms acquired for three life-cycle states. The measurements were conducted on fresh flash cells, cells with $1\,500$ P/E cycles and $13\ h$ of baking, which simulates about two months of data retention, and cells with $3\,000$ P/E cycles and $80\ h$ of baking time, which corresponds to one year DRT, and this life-cycle defines the end-of-life (EOL) provided by the manufacturer. Only the charged states S_1 to S_7 are shown. The charge state S_0 corresponds to the erased state of the cell, which can extend into the negative voltage range concerning $V_{th} = 0$. However, the distribution for this state cannot be determined correctly using commercially available chips. The controller can only adjust the voltages within certain voltage levels, identified by index numbers corresponding to the possible voltage steps of the flash controller.

The measured distributions are additionally dependent on the life-cycle state, including factors such as the number of P/E cycles and DRT, among others. Since the distributions can differ across various pages and blocks, the page and block numbers also contribute to the distribution's characteristics. However, for simplicity, we do not include these conditions in the notation. Measurements of all conceivable life-cycle states and their corresponding a priori probabilities influenced by usage would be necessary to derive the unconditional distributions.

To acquire a TVD, as shown in Figure 3.2, it is necessary to perform exhaustive conditional readings. This means that the written data is known, and all potential threshold voltages are examined during readout. Such measurements are typically reserved for testing due to their inherent complexity.

In Chapter 5, we use an extensive data set of measurements to explain the challenges related to RRVs and propose potential solutions to address these issues. The following section describes a model approximation based on measurements.

3.3 Model Approximation from Measured Voltage Distributions

Due to aging, the TVDs and the channel capacities of a flash memory change over time [16]. This section proposes a method to estimate the voltage distributions from measurement data.

Referring to Figure 3.2 and considering the logarithmic scale, we can see that the parabolic shape of the Gaussian distribution adequately approximates the voltage distribution around the

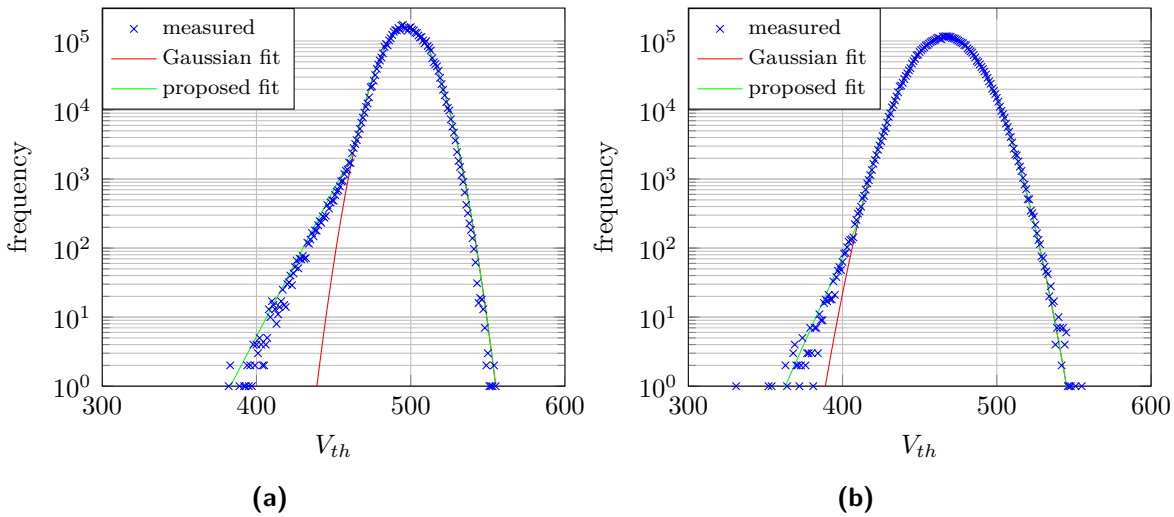


Figure 3.3: Fitted voltage distribution $f(V_{th} \mid S_7)$ of **(a)** fresh and **(b)** EOL flash memory (adapted from [1], ©2020 IEEE).

mean value. However, the measured distribution exhibits tails that deviate from the Gaussian distribution. The observation in [42] suggests that exponential tails in flash TVDs arise due to the influence of random telegraph noise effects, and these tails exhibit an asymmetric behavior at both the left and right extremes of the distribution.

The previous works in [43, 48] show that the NL and exponentially modified Gaussian (EMG) distributions result in good approximations of the MLC measurements. However, these distributions are not consistent with the TLC measurement data used in this thesis. The EMG distribution models the sum of two independent normal and exponential random variables, while the NL distribution describes the sum of independent normal and Laplace random variables. These distributions are obtained derived through the convolution of a normal density with either an exponential or a Laplace density.

Figure 3.3 shows exemplary S_7 for a fresh and EOL measurement, where the measured histograms display an inflection point at the left tail. Such inflection points cannot be accurately represented by the NL and EMG distributions.

We employ a parameter estimation approach presented in [1] to approximate the observed distributions. This involves employing a piecewise-defined function $f(x \mid S_i)$ for each state S_i. This function is divided into two segments, with the left segment representing an exponential distribution and the right segment representing a Gaussian distribution.

$$f(x \mid S_i) = \frac{1}{n_i} \cdot \begin{cases} c_i \cdot e^{\lambda_i (x - x_i)} & x < x_i \\ \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{\frac{-(x-\mu_i)^2}{2\sigma_i^2}} & x \geq x_i \end{cases} \tag{3.1}$$

with

$$c_i = \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{\frac{-(x_i-\mu_i)^2}{2\sigma_i^2}}. \tag{3.2}$$

With the cumulative distribution function $\phi(x)$ of the standard normal distribution, it results in

$$n_i = 1 + \frac{c_i}{\lambda_i} - \phi\left(\frac{x_i - \mu_i}{\sigma_i}\right), \tag{3.3}$$

which ensures the normalization

$$\int_{-\infty}^{\infty} f(x \mid S_i) dx = 1. \tag{3.4}$$

The proposed model with the piecewise function requires four parameters, i.e., $\sigma_i, \mu_i, \lambda_i$, and x_i. The left exponential tail is estimated using an exponential distribution, while the remainder is modeled as a normal distribution. This enables a precise estimation of the measured distributions, which is reflected by the fitted curves in Figure 3.3. This figure shows the measurements of the voltage distribution $f(V_{th} \mid S_7)$ of a fresh memory along with the corresponding fitted distribution. This model enables the estimation of error probabilities to non-adjacent states, e.g., from state S_5 to state S_7. Reliable estimation of such errors directly from the measurement samples is challenging due to the rarity of these errors, which have a very low occurrence probability.

The magnitude of the skewness, i.e., the third standardized moment of the measured distributions, is minimal. Consequently, the parameters σ_i^2 and μ_i can be computed from the sample mean and the sample variance, as the effects of the exponential tail on the mean and variance can be neglected. Working in the logarithmic scale proves most advantageous to identify the tails of the distribution. By employing the logarithm of the histogram, we can estimate the parameter λ_i of the exponential distribution by fitting a least squares regression line to the left side of the histogram. Subsequently, the intersection point x_i with the normal distribution can be determined. It is worth noting that the exponential tail towards higher voltages in the lower states of the measured distribution is not modeled, as its impact is negligible compared to the tail towards lower voltages.

We use the curve fitting to determine the optimal threshold voltage $V_{opt}(V_{ri})$ for each read reference such that the bit error probability is minimized, i.e., the optimal threshold voltage is defined as

$$V_{opt}(V_{ri}) = \underset{V}{\operatorname{argmin}} \int_{V}^{\infty} f(x \mid S_{i-1})\ dx + \int_{-\infty}^{V} f(x \mid S_i)\ dx. \tag{3.5}$$

Different error sources lead to manipulating the electrical charge on the flash cell, leading to threshold voltage shifts. Therefore, V_{opt} is not static; instead, it changes accordingly to the shifts.

Chapter 4

Bit-Labeling and Capacity Considerations

In this chapter, we first discuss various bit-labeling concepts for TLC flash memory, after which we introduce the computation of the channel capacity for flash memories. Subsequently, we analyze the presented bit-labeling schemes concerning their impact on the channel capacity.

As discussed in Chapter 2, the charge states of a flash cell correspond to binary information. The number of possible memory states in the flash cell varies depending on the specific flash technology. Numerous alternatives exist for employing different source codes to map information onto the flash cell. The mapping of a charge state can be arranged in exactly $a!$ ways, where a represents the number of distinguishable states.

The individual bits of the mapping are usually assigned to different pages, whereby the bit-mapping or bit-labeling defines the number of readouts for each page. Therefore, the total readout latency for page-wise read depends on the bit-labeling since a different number of RRVs must be applied for each page. Additionally, bit allocation leads to different page capacities, influencing the redundancy required for error correction. This impacts the resulting bit error probability per page. For instance, in SLC flash memory, which distinguishes two states, only two possible mappings arise. Furthermore, an MLC with four attainable charge states yields $4! = 24$ possibilities, while a TLC entails $40\,320$ possible patterns. This number highlights the complexity of finding good code labelings, especially when the number of distinguishable states is even more extensive. However, only some labelings are relevant for implementation because the benefits become redundant.

Channel capacity holds significant importance in ascertaining error-free data transmission across a communication channel. In this context, different bit-labeling strategies result in different channel capacities. To explore this further, we compare page and cell reading approaches.

The capacity of an MLC was derived and investigated in [16], considering its evolution over time and the impact of word line or page dependencies. Specifically, we focus on exploring bit-labeling and resulting capacities for TLC flash memory. The objective is to mitigate soft-read errors by achieving balanced error probabilities across pages. Due to the physical effects of the

flash cell, the charge distributions and the channel capacities change over the lifetime [16]. In the simplest form, the flash channel can be represented by ASK with Gaussian distribution of the charge states, i.e., additive Gaussian noise as described in Chapter 3. For simplicity, uniformly spaced amplitudes are used in this chapter.

In the following, we first introduce selected bit-labelings. Subsequently, the multi-level labeling is further explained, then we address the error probabilities for page-wise reading, and the calculation for the channel capacity is introduced. Finally, we consider the capacity at the EOL of measured flash memories for the different bit-labelings, followed by the capacity comparison between page-wise and joint processing for Gray codes. Parts of this chapter have been published in [1, 2, 4].

4.1 Bit-Labeling for Flash Memory

The bit-labeling or coding of the cell states has been addressed in the literature with different aims. In [49], a coding scheme was introduced that reduces inter-cell interference for various flash types, including QLC and PLC. These constrained codes exclude problematic bit patterns from the set of possible code patterns. However, these constrained codes significantly reduce the code rate. The achievable code rates are low compared to the spare area provided by today's flash memories. Another coding approach is pursued with write-once memory (WOM) codes [50, 51]. This coding technique attacks the aging due to program and erase cycles by reducing the number of erasures.

Chapter 3 discusses the relationship between charge states and bit-labeling in flash memories. Typically, flash memories utilize Gray codes for their bit-labeling [28]. Gray codes are binary codes in which adjacent charge states differ in only one bit position. This encoding strategy reduces latency for random access at the page level. Nevertheless, it is worth noting that Gray codes for TLC flash memory do not achieve an ideal equilibrium between the error probability and maximum achievable capacity when leveraging the complete information inherent to each flash cell [1].

Other code labelings may provide a better rate balancing than Gray labeling. Such codes can improve the error correction coding with hard-input decoding. This labeling may require more read thresholds than ordinary Gray labeling, although the number of read thresholds is significantly smaller than with soft readouts.

4.1.1 Exemplary Labeling and Partitions

Labeling for flash technologies with small memory sizes is trivial. When dealing with TLC flash memory or larger memory size, bit-labeling becomes complex due to the multitude of potential patterns, specifically for TLC $8! = 40\,320$ patterns are possible. Different strategies

for selecting a good bit-label can be useful. Nonetheless, achieving equal page capacities through bit-labeling is not feasible due to the number of charge states involved.

Table 4.1 presents all considered code labelings in this thesis pursuing different goals. Page no. 1 represents the MSB, page no. 2 is the CSB, and page no. 3 the LSB page. The binary components are associated with both a charge state S_i and a page number. The columns represent the charge states, while the rows represent the page number. Vertical bars separate the binary change within a page across all charge states to clarify the page-wise readout.

This section uses a different notation for the RRV. The reference $V_{ri,j}$ is defined by the intersection of the distributions of charge state S_i and S_j.

Table 4.1: Possible Gray code bit-labeling for TLC flash cell.

	page no.	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
Maximum distance code	1	1	0	1	0	1	0	1	0
	2	1	1	0	0	1	1	0	0
	3	1	1	1	1	0	0	0	0
Gray code 1	1	1	1	1	0	0	0	0	1
	2	1	1	0	0	1	1	0	0
	3	1	0	0	0	0	1	1	1
Gray code 2	1	1	1	1	1	0	0	0	0
	2	1	1	0	0	0	0	1	1
	3	1	0	0	1	1	0	0	1
Gray code 3	1	1	1	1	1	0	0	0	0
	2	1	0	0	1	1	1	0	0
	3	1	1	0	0	0	1	1	0
Multilevel code 1	1	1	1	0	0	0	0	1	1
	2	1	0	0	1	0	1	1	0
	3	1	1	1	1	0	0	0	0
Multilevel code 2	1	1	1	1	0	0	0	0	1
	2	1	0	1	1	0	1	0	0
	3	1	1	0	1	1	0	0	0

An intuitive approach for bit-labeling is to maximize the Euclidean distance of the symbols, i.e., distances between intersections for lapping decisions. Such an 8-amplitude modulation scheme was proposed in [52]. While the first level requires all upper references to differentiate odd and even states, the second level can use the information of the first level and therefore has to distinguish only even or only odd states using $V_{r\{i,j\}}$ with $j = i + 2$. These decisions have a much lower error probability.

The maximum distance code in Table 4.1 is such a code, with Figure 4.3 illustrating the concept. For an example, we assume that a cell is programmed to state S_1 with the label 011. For the first bit, we must distinguish even and odd states using all references $r_{i,j}$ with $j = i+1$. We can infer the second bit from the knowledge that the first bit is zero. Hence, only the odd states must be distinguished to read the second bit using $V_{r1,3}$ and $V_{r3,5}$. Similarly, with

the knowledge that the second bit is one, only the states S_1 and S_5 remain. Hence, only the threshold $V_{r1,5}$ has to be applied for the third bit.

Note that in flash memory, the read operation is performed on a page comprising many cells. Therefore, the same reference voltages are applied to all cells at the same time. Since the cells are programmed to different states, we must apply all possible reference voltages for each level. Hence, for the second level, we have to apply all $V_{ri,j}$ with $j = i + 2$ independent of the value of level 1. When decoding, the relevant thresholds are chosen to determine the values of the bits at each level.

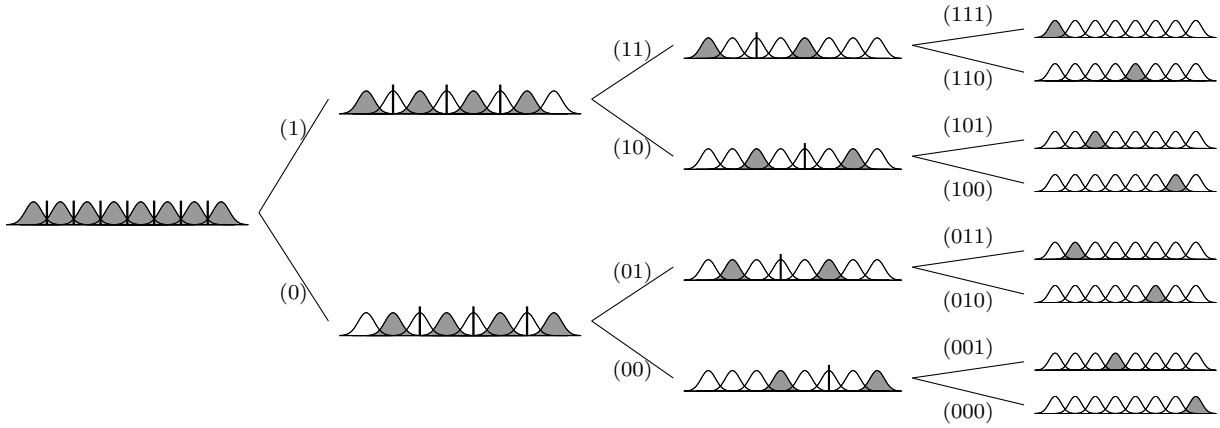


Figure 4.1: Maximum distance code partitioning for TLC flash bit-labeling

Figure 4.1 illustrates a partitioning scheme based on maximum distance coding. The final charge state is only determined in the last step of this scheme. Each V_r is applied at the first level, additional reference voltages are applied at the second decision level, and so on.

Gray code is a code family particularly suitable for page-wise readout of flash memories due to their properties. Adjacent binary codes only differ by one position, which reduces the average number of readouts required to read out individual bits. In addition, the error probability is more balanced.

Figure 4.2 illustrates the decision tree of Gray code 1, where the minimum number of RRV required to read out one bit is shown. Specifically, for the MSB page, $V_{r2,3}$ and $V_{r6,7}$ are necessary. The CSB page requires $V_{r1,2}$, $V_{r3,4}$, and $V_{r5,6}$. Finally, the LSB page needs $V_{r0,1}$ and $V_{r4,5}$.

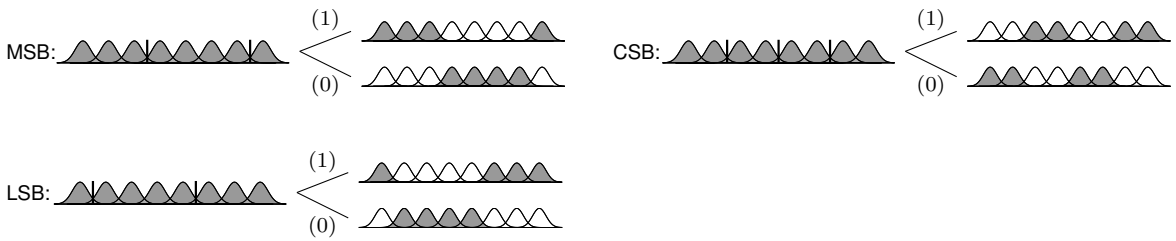


Figure 4.2: Decision tree for TLC flash using Gray code 1 bit-labeling

Gray codes minimize the latency for random access reads. However, in the next section, we will see that Gray codes cannot balance the page capacities. On the other hand, a multilevel coding strategy requires joint processing of all pages (also called cell-wise processing), which impairs the random read performance.

The capacity of a bit page depends on the error probabilities of all required references. Figure 4.3 shows a section of fitted distributions and different references. For Gray labeling, only the references $V_{ri,j}$ with $j = i + 1$ are required to read all three bits. Each bit page requires one to four references, depending on the labeling. Examples are Gray code 1 to Gray code 3 in Table 4.1.

Soft-decoding approaches for flash memories improve the error correction performance [53, 54]. A TLC with Gray labeling requires seven read references for hard-input decoding, whereas 21 read references are required to obtain a single SB and 35 for two SBs. These additional read references increase the channel capacity of the flash channel. However, the threshold voltage sensing operation to obtain the soft information incurs a more significant energy consumption and a latency penalty [55].

Next, we propose a new bit-labeling for the flash states. This labeling results from a multilevel coding strategy where only two pages are processed jointly. The proposed labeling features a better rate balancing than Gray labeling but reduces the latency compared with conventional multilevel coding [52, 56]. This approach provides a trade-off between latency and achievable capacity. It requires more read thresholds than ordinary Gray labeling, but the number of read thresholds is significantly smaller than with soft readouts. The aim is to achieve a capacity gain compared with page-wise reads with only a moderate increase in the number of read references.

Table 4.1 presents all considered codes. With the proposed labeling, only the first page is encoded independently of the other pages. The second and third pages are encoded jointly using a multilevel coding approach. The third level exploits knowledge from the second level and then has to distinguish only non-adjacent states.

In order to reduce the number of read thresholds, we can utilize the proximity of some thresholds. In Figure 4.3, for example the reference $V_{r1,5}$ is close to reference $V_{r2,3}$. Hence, using $V_{r2,3}$ instead of $V_{r1,5}$, the number of necessary threshold voltages and therefore the latency can be reduced. This would lead to an increased error probability for the third level compared with the decision at reference $V_{r1,5}$, although the error probability of this level would be negligible.

4.1.2 Error Probability Considerations

The capacity of the bit pages may significantly differ. The leading cause of these differences is the additional reference voltages required to read a particular page. To provide an insight into this issue, we consider a simplified model demonstrating that the bit error probabilities for the pages mainly depend on the number of reference voltages. Consider, for example, Gray code

2 from Table 4.1. The MSB is one for states S_0 to S_3 and zero for states S_4 to S_7. Hence, only the reference voltage V_{r3} has to be applied to read the MSB page. Assume that each state has the same probability $p(S_i) = 1/8$. The probability of making an error on the MSB page is therefore

$$p_{e,MSB} = \frac{1}{8}\sum_{i=0}^{3}\int_{V_{r3}}^{\infty} f(x|S_i)dx + \frac{1}{8}\sum_{i=4}^{7}\int_{-\infty}^{V_{r3}} f(x|S_i)dx, \tag{4.1}$$

where $f(x|S_i)$ is the probability density for the threshold voltage given that state S_i was programmed.

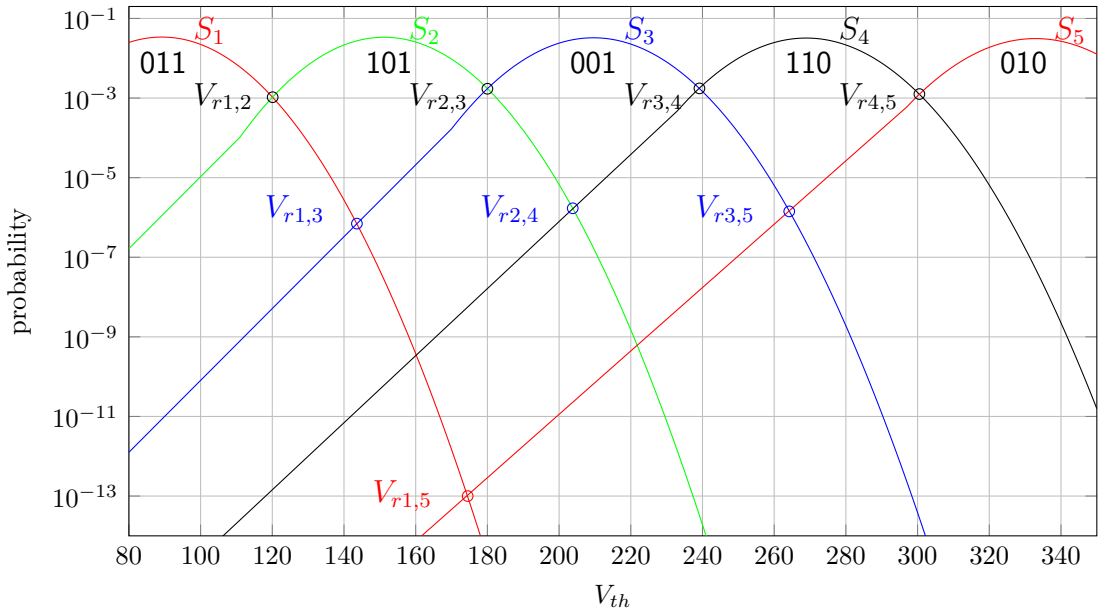


Figure 4.3: Example for distributions and reference voltages (adapted from [1], ©2020 IEEE).

Note that the error probability for each state mainly depends on the distance between the reference V_{r3} and the mean value μ_i for state S_i. The probability of an error to non-adjacent states is extremely low, as illustrated in Figure 4.3. Assuming that non-adjacent states cannot induce an error, we obtain the simplified estimate

$$p_{e,MSB} \approx \frac{1}{8}\int_{V_{r3}}^{\infty} f(x|S_3)dx + \frac{1}{8}\int_{-\infty}^{V_{r3}} f(x|S_4)dx. \tag{4.2}$$

Neglecting the asymmetry of the voltage distributions, we furthermore assume

$$p_e \approx \int_{V_{ri}}^{\infty} f(x|S_i)dx \approx \int_{-\infty}^{V_{ri}} f(x|S_{i+1})dx,\ \forall i. \tag{4.3}$$

This leads to the bit error probability $p_{b,MSB} \approx \frac{1}{4}p_e$ for the MSB page. Under this model, the value p_e is the probability that an error occurs at V_{ri} given that the programmed state is an adjacent charge state, i.e., either S_i or S_{i+1}. The factor $\frac{1}{4}$ is the probability that S_i or S_{i+1}

are used. Consequently, the symbol error probability is $p_s \approx \frac{14}{8} p_e$, because the states S_0 and S_7 have only one neighboring reference, whereas all other states have two.

In this simplified model, the bit error probability for a page depends on p_e and the number of states adjacent to the required reference voltages. The number of states adjacent to a reference voltage is determined by the number of references $N_{R,page}$ for the page. Thus, we have the bit error probability

$$p_{b,page} \approx \frac{2N_{R,page}}{8} p_e. \tag{4.4}$$

Consider, for instance, the LSB page for Gray code 2. The relevant references for the LSB page are V_{r0}, V_{r2}, V_{r4}, and V_{r6}. Hence, all states are adjacent to a reference voltage, which leads to the error probability for the LSB page $p_{b,LSB} \approx p_e$. Accordingly, Gray code 1 has the page error probabilities $p_{b,MSB} = p_{b,LSB} \approx \frac{1}{2} p_e$ and $p_{b,CSB} \approx \frac{3}{4} p_e$ and is therefore better balanced than the other Gray codes from Table 4.1.

While this model is oversimplified to calculate the exact bit error probabilities, it explains the differences of the page capacities in the results of the next section. Considering Gray code 2, the LSB page uses four references, leading to the highest bit error probability and the lowest page capacity. The CSB page uses two references. The highest capacity is obtained for the MSB page with a single reference voltage.

4.2 Channel Capacity of Flash Memories

The channel capacity C is a fundamental property in information theory, representing the maximum achievable communication rate. It establishes the upper limit for an error-free communication rate. This theory was introduced by Claude E. Shannon in his seminal work "A Mathematical Theory of Communication" in 1948 [57]. In the context of forward ECCs, the code rate is defined as $R = k/n$, where k is the code dimension and n is the number of symbols per codeword. Reliable transmission or data storage is only possible for $R < C$, as exceeding this would compromise the data integrity.

To estimate the channel capacity, we consider a discrete memoryless channel, i.e., we neglect inter-cell interference from neighboring cells. In this case, the capacity C can be obtained by means of numerical integration [52]. The channel capacity is also defined as the supremum over all possible input distributions $P_X(x)$ of the mutual information (MI) between two random variables

$$C = \sup_{P_X(x)} I(X;Y), \tag{4.5}$$

where $I(X;Y)$ is the MI between the random variables of the channel input X and the output Y. The MI defines the amount of information exchanged between the random variables and therefore reflects the quality of the communication channel. In the case of flash memory, the channel input X represents the written charge state of the flash cell, and Y represents the readout of the charge state by applying different reference voltages V_r. However, only a subset

of the possible reference voltages may be applied. In the following, we treat MI as a quality measure.

A disadvantage of page-wise reading is the performance loss regarding the achievable channel capacity, which can be explained by the chain rule of MI. Let $Y_0, Y_1, \ldots, Y_{n-1}$ be the binary results from reading at the n references, i.e., the channel output is the binary vector $\mathbf{Y} = (Y_0, Y_1, \ldots, Y_{n-1})$. As a result, we have the following chain rule [58]

$$\begin{aligned} I(\mathbf{X};\mathbf{Y}) &= I(\mathbf{X};Y_0,Y_1,\ldots,Y_{n-1}) \\ &= \sum_{i=0}^{n-1} I(\mathbf{X};Y_i|Y_0,Y_1,\ldots,Y_{i-1}). \end{aligned} \tag{4.6}$$

Hence, omitting some references may result in a reduction of the achievable MI. Note that $\mathbf{Y}$ can also be represented by a vector with $\lceil \log_2(n) \rceil$ bits. However, we use this notation to indicate the difference in read operations using a reduced set of references V_{ri}. For instance, only the references V_{r2} and V_{r6} are used to read the MSB of Gray code 1. Hence, the channel output for the page-wise read of the MSB page is $\tilde{\mathbf{Y}} = (Y_2, Y_6)$. We denote the MI for such a reduced set of references by $I(\mathbf{X};\tilde{\mathbf{Y}})$, i.e., $\tilde{\mathbf{Y}}$ is a binary vector with a length smaller n.

Similarly, we can represent the input $\mathbf{X}$ by l binary random variables $X_1, \ldots, X_l$. In flash memory, these bits are assigned to different pages. For this representation, we have the chain rule [56]

$$\begin{aligned} I(\mathbf{Y};\mathbf{X}) &= I(\mathbf{Y};X_1,X_2,\ldots,X_l) \\ &= \sum_{i=1}^{l} I(\mathbf{Y};X_i|X_1,X_2,\ldots,X_{i-1}). \end{aligned} \tag{4.7}$$

This leads to the definition of the page capacities C_i

$$C_i = I(\mathbf{Y};X_i|X_1,X_2,\ldots,X_{i-1}) \tag{4.8}$$

for fixed a priori probabilities of bits. Note that the page capacities C_i depend on the mapping from bits to cell states and the reading order. According to the chain rule of information theory, the sum of the page capacities C_i results in the total capacity [56]

$$C = \sum_i C_i. \tag{4.9}$$

This capacity is achievable with a multilevel coding approach by adjusting the code rate of each page to its page capacity $R_i = k_i/n_i = C_i$ [16]. However, such a multilevel coding strategy generally requires joint processing of all pages. In contrast to (4.8), we define the page capacity $\tilde{C}_i$ for an independent page-wise read as

$$\tilde{C}_i = I(X_i;\tilde{\mathbf{Y}}_i). \tag{4.10}$$

Note that the set of read references depends on the page. Hence, the channel output $\tilde{\mathbf{Y}}_i$ depends on i. Similar to (4.9), we can define the achievable capacity with page-wise read as $\tilde{C} = \sum_i \tilde{C}_i$.

The overall capacity C does not change with the bit-labeling for a fixed set of reference voltages. However, the page capacities C_l depend on the bit-labeling. For practical purposes, most coding schemes for NAND flash memories use page-wise encoding with the same number of information bits k, number of code bits n, and code rate R on all pages. Such independent processing may incur a capacity loss. In this case, the code rate has to satisfy

$$R = R_l < C_{min} = \min_l C_l. \tag{4.11}$$

Accordingly, the achievable capacity is $C_p = 3C_{min} \leq C$ for a TLC with page-wise read, which restricts the achievable error-correcting performance. To reduce the capacity loss, the bit-labeling should balance the page capacities.

Furthermore, the capacity of a flash can be increased using multiple reads per reference, which leads to additional soft information. All threshold voltages must be applied sequentially to read out a flash memory. Hence, the latency and energy consumption increase with the number of thresholds [55].

In addition to reading flash cells page-wise, it is also possible to perform a joint readout. This approach involves sequentially applying all reference voltages to read a symbol rather than a single bit at a time.

The upcoming sections consider the channel capacity by employing MI calculations for EOL scenarios on flash memory. We compare various bit-labeling strategies and examine page-wise and joint readout methods.

4.3 Capacity on End-of-Life

This section investigates the proposed bit-labeling and the resulting page capacities for a TLC flash. Table 4.2 presents page capacities with different bit-labeling for a TLC at EOL. The values $C_0,\ C_1,\ C_2$ are the page capacities for the MSB, CSB, and LSB page, respectively. C is the total capacity according (4.9). The value C_p indicates the capacity for page-wise reads according to (4.11). The last two columns of Table 4.2 indicate the number of read references and the minimum required spare area. The last value indicates the minimum percentage of the total flash storage capacity required for error correction coding. The spare area is an essential component of the memory region specifically not used as data storage. It is preserved for data integrity and to ensure operational efficiency. Consequently, crucial components like the parity of the ECC are stored here.

The three Gray codes proposed in Table 4.1 use the same seven read references. Hence, these codes result in equal total capacities (with minor numerical differences). However, all

Table 4.2: Page capacities of a TLC at EOL depending on the bit-labeling ([1], ©2020 IEEE)

	C_0 (MSB page)	C_1 (CSB page)	C_2 (LSB page)	C	C_p	read thresholds	spare area (%)
Gray code 1	0.967	0.958	0.982	2.907	2.874	7	4.20
Gray code 2	0.949	0.975	0.983	2.907	2.847	7	5.10
Gray code 3	0.983	0.964	0.959	2.906	2.877	7	4.10
Gray 1 with 1 bit soft	0.981	0.975	0.988	2.944	2.926	21	2.46
Gray 1 with 2 bit soft	0.982	0.978	0.991	2.951	2.934	35	2.21
max. dist.	0.916	0.999	0.999	2.915	2.915	17	2.83
proposed code 1	0.967	0.945	0.999	2.911	2.901	10	3.30
proposed code 2	0.979	0.936	0.999	2.914	2.903	9	3.25
prop. 2 with 1 bit soft	0.984	0.961	0.999	2.944	2.940	23	2.00

Gray codes result in different page capacities. These differences lead to a capacity loss with page-wise read operations. The Gray codes require a spare area of at least 4.1%.

The necessary size of the spare area for error correction can be reduced with special labeling codes. The row denoted by max. dist. in Table 4.2 presents results for a multilevel coding approach [52, 56]. This scheme would be the most efficient coding scheme from a coding perspective, i.e., it achieves the smallest spare area of 2.8%. A capacity-achieving code would be required only to protect the MSB page, whereas the capacities of the two other pages are very high. Hence, simple algebraic codes could be used for error correction in the CSB and LSB pages. However, this maximum distance labeling requires additional read references. A total of seventeen references are required, which increases the read latency compared with Gray labeling. These additional references result in a slight gain in the total capacity compared to page-wise reads.

The two multilevel codes in Table 4.1 use a joint processing approach for the CSB and LSB page, whereas the MSB page can be independently read. This approach reduces capacity loss compared with Gray labeling. The minimum required spare area is 3.25%, and the increase in the number of RRVs remains moderate.

Table 4.2 contains results with one or two bits of soft information using Gray code 1 and multilevel code 2. The soft-decision (SD) decoding with Gray code 1 results show an enhanced capacity, notably reducing the required spare area from 4.2% to 2.46% or 2.21% with one or two SBs, respectively. However, these capacity improvements come at the expense of a high read latency due to the requirement of applying 21 or 35 RRVs for reading the soft information across all read references.

The maximum capacity is achieved by employing only one SB for the proposed code 2, requiring 2% spare area. Simultaneously, reading one SB for the proposed code 2 requires only 23 threshold voltages, resulting in lower latency and energy consumption than Gray coding with two bits of soft information.

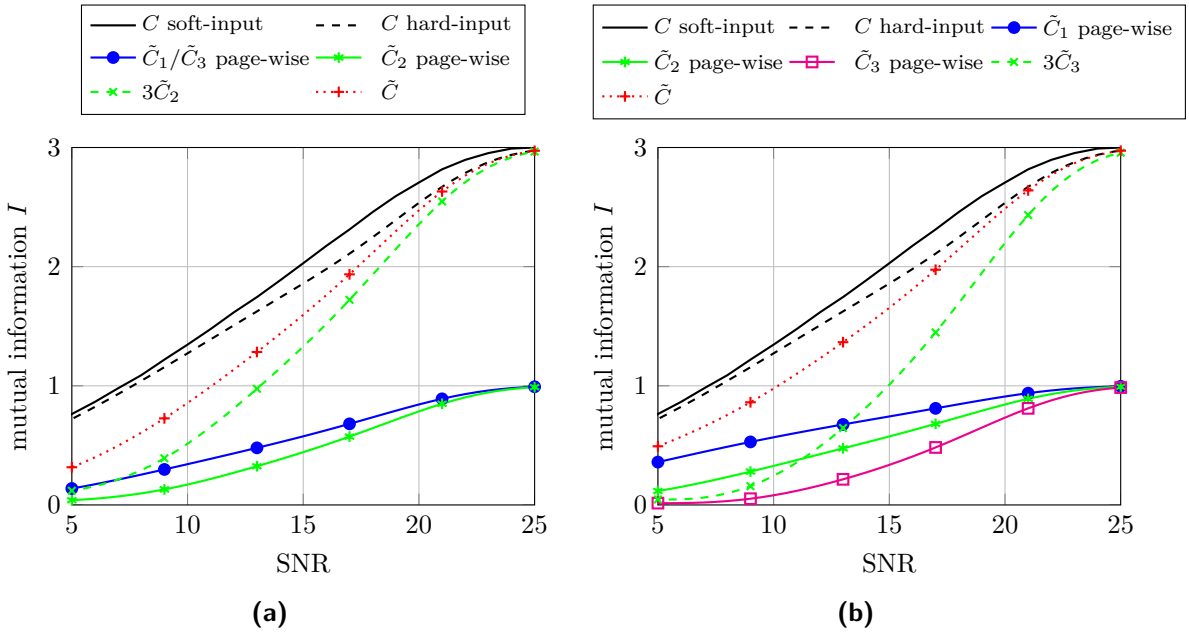


Figure 4.4: MI for the TLC channel model using **(a)** Gray Code 1 and **(b)** Gray Code 2 ([4], ©2022 IEEE).

4.4 Capacity Comparison of Page-Wise and Joint Readout

This section compares the capacity considering page-wise and joint readout scenarios, utilizing specific bit-labeling schemes. We employ the ASK modulation with AWGN for modeling each charge state, as introduced in 3. Due to the NAND architecture, only threshold voltages of one polarity are used to read the flash cell. This context leads to using unipolar ASK modulation to represent the flash channel. Uniformly spaced amplitudes are adopted to simplify the model further. We further consider TLC flash memory, i.e., we employ a unipolar 8-ASK as the modulation alphabet. This channel model is used throughout this section.

Figures 4.4a and 4.4b show the MI for Gray code 1 and 2 from Table 4.1 versus the signal-to-noise ratio (SNR). These codes are used for mapping the information bits to charge states. The solid black lines correspond to the overall capacity C with real-valued soft-input, whereas the dashed black lines correspond to hard-input. Note that the overall capacity C is independent of the mapping. However, the page capacities $\tilde{C}_i$ depend on the Gray code. $\tilde{C}_1, \tilde{C}_2, \tilde{C}_3$ are the page capacities for the MSB, CSB, and LSB page, respectively. Reading soft values increases the MI, but additional readouts are needed, increasing the latency.

In Figure 4.4a, the LSB and MSB page have the same page capacities, whereas the capacity for the CSB page is significantly lower. This results from the different bit error probabilities in the pages, i.e., the bit error probabilities depend on the bit-mapping. For high SNR, the ratio of the average bit error probability P_b to the symbol error probability P_s can be approximated

as [59]

$$P_b \approx \frac{P_s}{\log_2(M)}. \tag{4.12}$$

Section 4.1.2 presents a detailed analysis of the page error probabilities. For Gray code 1, the page probabilities become $P_{b,MSB} = P_{b,LSB} \approx \frac{2}{7}P_s$ and $P_{b,CSB} \approx \frac{3}{7}P_s$. For Gray code 2, the differences in the bit error probabilities are even larger, which results in the large differences in the page capacities shown in Figure 4.4b.

The various page capacities have a direct impact on the coding scheme. Achieving capacity $\tilde{C}$ requires adjusting the code rates of the different pages to match their respective channel capacities, as discussed in [16]. Regardless, there remains a noticeable disparity between the capacity $\tilde{C}$ (indicated by the red dotted line) and the overall capacity C (represented by the black dashed line). This gap primarily results from the page-wise read operation, which uses only a subset of the potential read references. To avoid this capacity gap, it is necessary to implement joint processing of all pages at the decoder.

For practical purposes, most coding schemes for NAND flash memories use page-wise encoding with the same code rate R on all pages. This approach further reduces the achievable capacity because the worst-case page capacity determines the attainable code rate $R < \min_i \tilde{C}_i$. In Figure 4.4a, the CSB page determines this code rate and results in the overall capacity $3\tilde{C}_2$ (green dashed line). The LSB page has the worst-case performance in Figure 4.4b. Hence, the overall rate is limited to $3\tilde{C}_3$ (green dashed line).

4.5 Summary

We have described a multilevel bit-labeling for TLC flash memories that provides a better balancing of the page capacities than ordinary Gray labeling. This labeling requires only two additional read thresholds compared with Gray labeling. Hence, the number of read thresholds is significantly smaller than with soft readouts. The new labeling achieves a significant capacity gain compared with Gray labeling and page-wise read operations. The price for this capacity gain is the joint processing of two pages, which degrades the random access performance.

We have also investigated cell-wise, i.e., joint processing of all pages and page-wise processing for Gray codes for TLC NAND flash memories. This procedure increases the MI, and therefore it would improve the error-correcting capability. Typically, the design strives to maximize the MI at the EOL scenario. For this reason, we evaluated the labeling on the EOL. The ASK modeling approach is oversimplified and does not pass on the physical changes, but it indicates which type of readout provides more information. This simplified modeling serves as a valuable analysis tool without requiring extensive measurements or complex channel specifications.

The next chapter delves into the challenges associated with RRVs and elucidates the necessity for adaptation. We will also present various adaptation methods to address these challenges effectively.

Chapter 5

Read Reference Voltage Adaptation

In flash memory, the information is physically stored on a flash cell as an electrical charge (cf. Chapter 2). This electrical charge changes the threshold voltage of the flash cell. Various flash cells are read simultaneously, considering NAND flash memory. Therefore, to read the data, we apply a certain RRV to several cells and check which charge was stored on the cells. Depending on the technology, multiple voltages must be applied to read the information [13].

For signal detection, the reading method of the data is essential. This process restricts significantly further processing. The behavior of flash cells during readout is described using probability distributions related to the physical processes of writing, erasing, and reading (cf. Chapter 3). Moreover, the error probability of flash cells is influenced by various factors, including the storage density and charge losses caused by the deterioration of the flash cells. Employing a constant RRV cannot sufficiently mitigate the bit error rate (BER), and the elevated number of errors might exceed the capability of the ECC. The RRV must be dynamically adapted based on the current life-cycle state of the flash memory to ensure low residual error probability while maintaining high processing speeds. Ideally, the adaptation aims to attain the optimal threshold voltage V_{opt}, which is the RRV that minimizes the residual error rate. However, achieving this condition is a non-trivial task. Throughout this chapter, we will describe this in more detail.

Compared with HD read operation, the threshold voltage sensing operation to obtain the soft information causes a substantial latency and an increased energy consumption [55]. Consequently, soft-input decoding should only be employed selectively, primarily for blocks where the HD input decoding fails. Algorithms that adapt the RRVs can reduce the latency and energy consumption by avoiding SD readouts. Moreover, many SD input decoding approaches assume reading procedures that maximize the MI between the input and output of flash memory. This requires precise calibration of the RRV for the quantization intervals [60, 61].

Since complete TVDs are usually not available, it cannot be assumed that the RRV, which minimizes the residual error probability and improves the readout method, can be derived solely from measured conditional distributions. For this reason, different RRV adaptation methods have been proposed in several publications [40, 62, 63, 64, 65, 66, 67, 68] based on different features. Methods to estimate the flash channel using neural networks (NNs) have

been proposed in [69, 70, 71]. The flash controller, for instance, can employ a read-retry mechanism to adapt the reference voltages. Initially, the controller reads the data using the default RRV. If the ECC decoder successfully corrects the errors, there is no need to modify the default value. However, in the case of decoding failures, the flash controller performs multiple reads with specific threshold voltages until the decoder manages to correct all errors. Such read-retry mechanisms can be considerably time-consuming.

This chapter outlines various data-driven approaches to adapt the RRV to minimize the residual error rate. The upcoming section describes the conventional hard-decision (HD) and SD readout techniques for flash memory. Subsequently, we will discuss features used for RRV adaptation based on the error statistics and histograms based on the TVD. Therefore, we introduce the information-theoretical calculations employed to analyze and optimize the calibration. Additionally, this chapter describes a calibration algorithm based on the number of errors in the metadata. Accordingly, a tracking algorithm is presented to avoid using this calibration algorithm for many consecutive pages. Moreover, we present an approach involving NN aided RRV adaptation. We will explore the available training data and assess the applicability of this approach across different scenarios. The results of all adaptation methods rely on measurement data of TLC NAND flash. Parts of the content in this chapter have previously been published in [3, 5, 6, 7, 8].

5.1 Readout Methods

Two primary types of readout methods are distinguished: HD readout and SD readout. The former refers to a readout type that distinguishes only between defined states. In the case of an SLC flash memory, a single query is sufficient to distinguish its memory states. However, as the number of possible charge states increases, more readouts are necessary to distinguish the charge states effectively. Specifically, for S charge states, a total of $S-1$ defined reads are required for HD readout.

On the other hand, SD decoding describes the readout of additional reliability information. The soft information is commonly converted into log-likelihood ratios (LLRs), which are exploited by the ECC. Increasing the resolution leads to higher MI, enabling more robust error correction [53, 60, 72].

In flash memory, several read cycles of a cell are necessary to acquire additional information. A high number of readout steps increases the information content of the readout, although this increases latency and enlarges the energy consumption [55]. An increased resolution of the channel output also demands additional hardware since this information must be quickly transferred and buffered. Commonly, it is preferable to employ as few readouts as possible, aiming to minimize the complexity overhead of the readout process.

As the number of readouts increases, both the resolution and the MI rise accordingly, as shown by Wang et al. [72]. However, analyzing different readout resolutions reveals that no

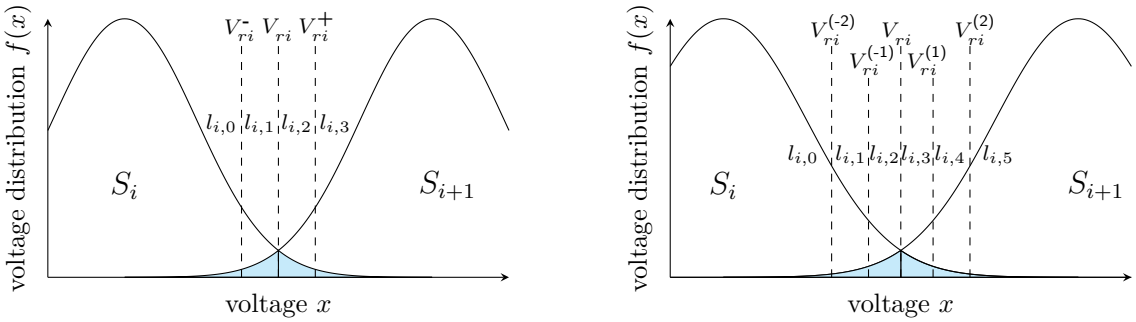


Figure 5.1: Schematic of two soft readout possibilities of adjacent TVD. **(a)** Readout with two additional RRVs, resulting in one SB, and **(b)** Readout with four additional RRVs, resulting in two SBs (adapted from [7], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/).

substantial improvements can be anticipated beyond a certain threshold. Consequently, in this work, we will focus our further analysis exclusively on three and five readouts between charge states, i.e., one SB and two SBs readout methods. These readout methods are shown in Figure 5.1.

Commonly, information is first accessed by HD readouts to process the data as quickly as possible. SD readout and decoding are only performed on poor channel conditions, and the readouts are chosen such that the MI is maximized, which depends on the SNR. The search for the optimal SD RRVs is neglected in the following approaches. The estimation methods focus on optimizing the readout process for HD by identifying the optimal RRV V_{ri}.

5.2 Features for Read Reference Voltage Estimation

The optimal RRV can be derived in the simplest case by analyzing the TVD and utilizing algorithms to identify the minimum of a function. This would ensure the minimum possible readout error. Still, the maximum achievable resolution for readouts is limited to those presented in the previous section. Generally, the assumption is that readouts with a higher resolution than two SBs are not feasible due to the increased complexity. High-resolution conditional TVDs are not available during runtime.

In the past, various approaches utilizing distinct features have been published. In order to obtain statistical information about the cell state, a sensing-directed estimation method was proposed in [40]. This method approximates the TVD as a Gaussian mixture model (GMM). Although, GMMs are not suitable to approximate the TVDs of current TLC NAND flash memories [3, 37, 44]. This scheme does not rely on known pilot bits but employs additional memory-sensing operations similar to the read-retry mechanisms. Such voltage sensing operations incur a larger energy consumption and a latency penalty. The adaptation method from [65] is also based on the assumption of Gaussian voltage distributions. However, we have

shown in Chapter 3 that the cell TVDs of MLC and TLC are actually highly asymmetric with exponential tails [1, 38, 44] and cannot be estimated with Gaussian distributions.

An adaptation algorithm based on an error characteristic-aware RRV acquisition scheme was proposed in [73]. In this approach, the TVD is characterized using a polynomial function fitted to a limited number of measurements. By identifying the minimum point between two neighboring charge states, the optimal RRV is determined. It is worth noting that this method requires a considerable number of measurements and imposes significant latency due to the polynomial calculation process.

Methods that leverage NNs to estimate the flash channel have been proposed in [69, 70]. In [69], an NN-based approach is introduced, exploiting meta-information and ECC decoding. This method employs features such as the number of P/E cycles, the frame error rate (FER), the average iterations of the underlying LDPC decoder, and the bit flip ratio of this particular decoder. Although showing promising results, this technique requires access to decoded metadata, requiring additional readouts and processing.

The features discussed in this section are independent of Gaussian models or any specific knowledge of the life-cycle state. Instead, we employ information-theoretic measures to explore the relation between the optimal read references and the statistical information derived from the features.

5.2.1 Asymmetry of Error Probabilities

Research conducted in [44, 62, 63, 74] indicates that the probabilities of neighboring states in flash memory are asymmetric. This asymmetry is also demonstrated in Chapter 3 through the asymmetric properties of the threshold distributions. We will exploit this observation to adjust the RRVs.

This asymmetry causes an asymmetric number of errors for the neighboring states. Similar to the methods proposed in [64, 66], we exploit the ECCs to obtain information about the cell state. Considering two states, i.e., the binary values 0 and 1, two error ranges define the error probability between the written data w and read data r $P(r = 1|w = 0)$ and $P(r = 0|w = 1)$. These values correspond to the error probabilities of the adjacent states of the RRV. Typically, these probabilities are unavailable as it is unknown which specific information was previously written to the flash cells. We have adopted the terms 0-to-1 and 1-to-0 to describe this particular type of error. In the following context, 1 represents the lower voltage, while 0 denotes the higher threshold voltage condition. In a nutshell, the number of errors provides information about the distance to the optimal threshold, and the asymmetry provides information about the direction, i.e., whether to increase or decrease the reference voltage.

The position of the optimal reference voltage V_{opt}, described in Equation 3.5, depends on the physical effect and asymmetry. Figures 5.2a and 5.2b illustrate the average error count when

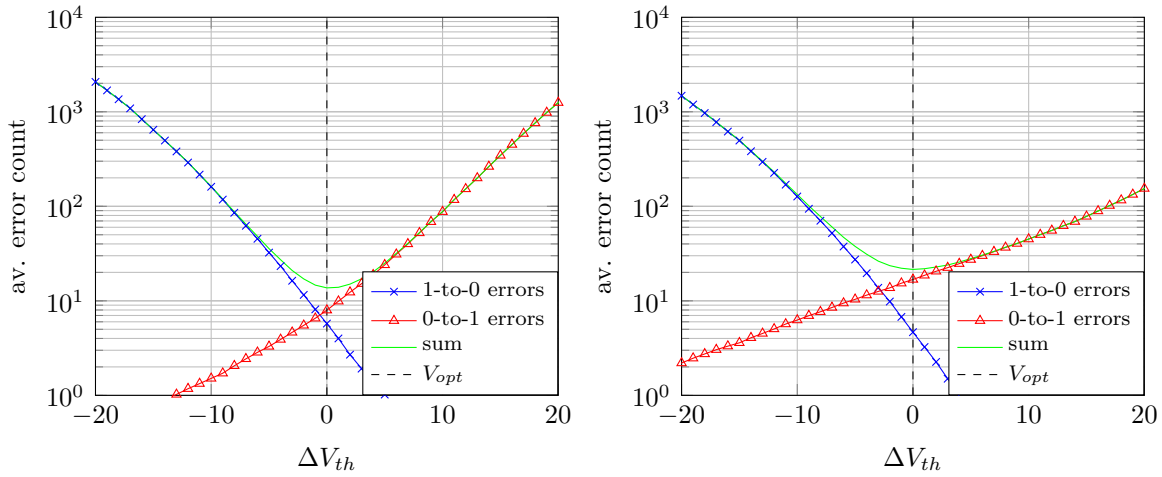


Figure 5.2: Average number of 1-to-0 errors and 0-to-1 errors versus the threshold voltage for **(a)** reference V_{r2} and **(b)** V_{r6} at fresh life-cycle (adapted from [3], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/).

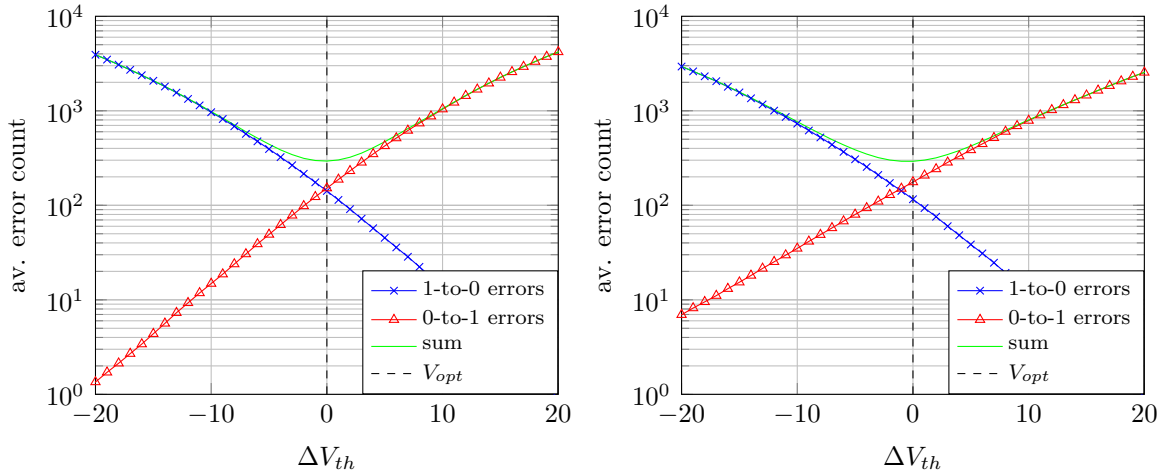


Figure 5.3: Average number of 1-to-0 errors and 0-to-1 errors versus the threshold voltage for **(a)** reference V_{r2} and **(b)** V_{r6} at EOL (adapted from [3], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/).

applying a specific threshold voltage on a page of a fresh TLC NAND flash memory. Similarly, Figures 5.3a and 5.3b show the error count under an EOL scenario. The x-axis describes the possible threshold voltage in discrete voltage steps, while the y-axis represents the average number of errors.

The optimal reference voltage is at $\Delta V_{th} = 0$. A high number of 1-to-0 errors occurs when the threshold voltage is on the negative part of the x axis. This situation arises from misinterpreting the symbol 1 as 0. In Figure 5.2a, the optimal reference voltage is exactly aligned with the intersection of the neighboring distributions. On the other hand, Figure 5.2b shows an example where the optimal point is shifted. When adjusting the reference voltage, these effects must be considered to improve accuracy.

5.2.2 Number of Errors

Two types of data are stored in flash memory, as introduced in Chapter 2: the main area, where user data or payload data is stored, and the spare area, which is used for ECC and firmware metadata. The metadata also contains some essential characteristics of the life-cycle status, e.g., the number of P/E cycles and other information. Different ECCs are typically applied for the user data and metadata. The metadata is protected by a significantly robust ECC, which typically possesses a lower code rate than the ECC used for the payload data. Hence, correcting the errors in the metadata codeword with the meta ECC is possible even at a relatively large offset in the reference voltage, i.e., in poor channel conditions. On the other hand, the length of the metadata is small because it stores few bytes. As the code length is short, this can be processed very quickly. Consequently, the number of errors in the meta ECC has an unavoidably small sample size. However, the number of errors in the metadata can provide valuable insights into the condition of the flash cells that can be used to derive good RRV. In most cases, the observed errors in the metadata are correctable, offering valuable information about the state of the flash cell.

Moreover, there is a noticeable correlation between the number of errors and the life-cycle state across various life states. The metadata and the payload data of a flash memory page undergo the same physical aging process. Consequently, the information obtained from the metadata can be used to infer suitable reference voltages for the payload data. Analyzing the number of errors in the metadata provides information about the life-cycle state of the flash cells.

Figure 5.4 shows the total number of errors (y-axis) in relation to discrete threshold voltage steps (x-axis) for two specific reference voltages: V_{r2} distinguishing S_2 and S_3, and V_{r6} distinguishing S_6 and S_7. These results are presented for two different life-cycle states. We use these RRV in this example because the MSB page of Gray code 1 from Table 4.1 uses them. This analysis considers codes with length $n = 511$ and error correction capability of $t = 21$ errors. Using a BCH code, this translates to a $\mathcal{B}(511, 331, 43)$ code.

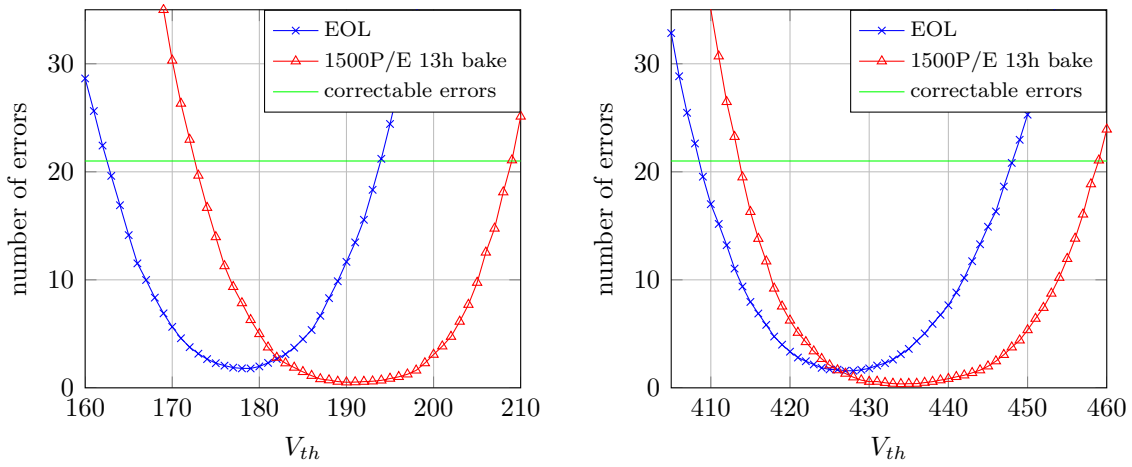


Figure 5.4: Average number of errors in the metadata for different life-cycle states (a) between S_2 and S_3 for V_{r2} (b) between S_6 and S_7 for V_{r6} (adapted from [3], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/).

The yellow horizontal lines in Figures 5.4a and 5.4b indicate the maximum number of correctable errors, i.e., $t = 21$. Information about the life-cycle conditions is obtained based on the error counts. For the discrete threshold voltage $V_{th} = 190$ in Figure 5.4a, the average error number of EOL is $t = 11$ and for the life condition with 1500 P/E cycles and $13\ h$ of baking ($\approx 1.4{\cdot}10^3\ h$ DRT), the average error number is almost zero.

Similarly, it is shown in Figure 5.4a that for constant V_{th} values, differences in average error rates are observed. The error number can distinguish between these two life states in these examples and even predict the corresponding RRV shift. To apply this approximation, it is necessary to read at a position that provides sufficient information about the possible life state and the optimum.

The observation indicates that if V_{th} is far enough from the optimum, additional information value will be gained, while a value near the optimum readout will gain little information about the life state. The idea is to use this information to predict the offset to the optimal readout voltage. If a threshold voltage is above the error limit, the correct error number cannot be detected, which may lead to incorrect decisions and lower information gain.

5.2.3 Histogram Generation

By employing multiple threshold voltages V_{th} for control, it becomes possible to count the number of switching cells and thus generate a distribution function with a defined resolution. The TVD presented in the preceding chapters represent the histogram with the highest possible resolution for flash memory. Conducting measurements during runtime does not permit reading conditional distributions, and measurements are not feasible due to the significant latency

caused by the numerous readout cycles involved.

To enable feasible histogram generation during runtime, the resolution can be reduced, allowing for a minimal number of readout cycles to be performed. As a result, a sparse histogram is obtained. The sparse histogram is generated such that cells are counted, which are turned on by the HD RRV in the interval between two reference voltages V_{rk} and V_{rk+1}. This sparse histogram component is similar to the description of the on-cell ratio from [71]. However, we do not calculate a ratio, but instead we count the active cells between adjacent RRVs, and it is not strictly necessary to use predefined RRVs.

We will consider different histogram resolutions in the further course of this chapter. The simplest histogram comprises the relative frequencies of zeros and ones at the different RRVs using the HD readout. Higher resolution histograms are obtained by considering the quantization levels of SD readout.

Example 5.1. *For a better understanding, consider the sparse histogram generation using Gray code 1 from Table 4.1.*

Under HD readout of the MSB and LSB, two RRVs need to be applied, and for CSB, three RRVs are required. Counting the activated cells triggered by an RRV V_{rk} enables determining the active cells between two consecutive RRVs. Consequently, this approach enables quantifying active cells within each interval.

Accordingly, the application of HD readout on the MSB and LSB pages results in three values, and for the CSB it generates four values. With SD readout, the number of intervals increases. The sparse histogram generation for the MSB and LSB pages using one SB readout results in seven intervals. When considering a two SB readout, the histogram is extended to eleven values. Similarly, the CSB page requires ten intervals for one SB and sixteen intervals for two SBs.

The technical requirement for this process is a read-bit counter. Alternatively, a viable option is to conduct a joint readout of the pages, effectively generating a comprehensive representation of the data.

5.2.4 Mutual Information of Different Features

The features presented are possibilities to obtain information regarding the optimal read reference voltage. We use information-theoretic measures to investigate the relation between the optimal read references and the statistical information obtained by the different features.

In particular, we consider the MI. The MI $I(X;Y)$ of two random variables is a measure for the information in bits gained about one variable when knowing the other variable. Let X and Y be two random variables with $X \in \{x_1, x_2, ..., x_N\}$ and $Y \in \{y_1, y_2, ..., y_M\}$ with probability

distributions $f_X(x_i)$ and $f_Y(y_j)$ and joint probability distribution $f_{XY}(x_i, y_j)$. The MI between X and Y using probability mass functions for discrete distributions is defined as [75]

$$\begin{aligned} I(X;Y) &= H(X) - H(X|Y) \\ I(X;Y) &= \sum_{x\in X}\sum_{y\in Y} f_{XY}(x,y)\cdot\log\left(\frac{f_{X|Y}(x|y)}{f_X(x)}\right). \end{aligned} \tag{5.1}$$

Where $H(X)$ defines the entropy (or uncertainty) of the variable X and $H(X|Y)$ is the conditional entropy. The MI $I(X;Y)$ is always non-negative and upper bounded by the minimum of the entropies of the two variables. Consequently, the inequality $H(X) \geq H(X|Y)$ holds. Note that in our case, continuous distributions are not available because all distributions are measured based on discrete values. For practical applications, it can have advantages using other measures, e.g., the residual error rate of a given decoder. However, those measures are very specific to a given implementation.

When considering the asymmetry and the number of errors, we exclusively consider information derived from the metadata. Specifically, we consider a codeword length of $n = 511$, with the capability to correct up to $t = 21$ errors. Despite the relatively small size of the metadata, this information is also influenced by the various life-cycle states.

For assessing the asymmetry, we introduce a binary random variable A, which indicates the direction of the asymmetry between the error proportions at a reference voltage. If more errors occur on the right side of the distribution, A takes the value of 1, otherwise $A = 0$. The MI between the asymmetry and V_{opt} is denoted as $I(A; V_{opt})$. Next, we consider the MI between the decodable number of errors E and V_{opt}, denoted by $I(E; V_{opt})$. The variable E is limited in the range 0 to 21 because only the number of the exact detected errors is considered.

Another possible feature relies on reading the unconditional payload data without decoding, which enables directly using the read information without further processing. The variable Z represents the number of active cells within each interval of unconditional quantization levels. Specifically, an instance of Z is defined for each interval, resulting in two values in the case of HD readout. Using readout methods to increase the reliability information as shown in Figure 5.1, the interval count increases to four for one SB and six intervals for two SBs.

To quantify the relationship between Z and the optimal threshold voltage V_{opt}, we introduce the MI $I(Z; V_{opt})$. Given this aspect, we describe this MI based on intervals as

$$\begin{aligned} I(\mathbf{Z}; V_{opt}) &= I(Z\{l_0\}, ..., Z\{l_n\}; V_{opt}) \\ &= H(Z\{l_0\}, ..., Z\{l_n\}) - H(Z\{l_0\}, ...|V_{opt}). \end{aligned} \tag{5.2}$$

Figure 5.5 illustrates $I(E; V_{opt})$ and $I(A; V_{opt})$ regarding the metadata, as well as $I(Z; V_{opt})$ corresponding to the payload data. The latter MI is computed concerning the counter of the HD readout and separately for the counter of the one SB readout. Subsequently, as the SD readout is mainly used for worst-case scenarios, all RRVs for the SD readout are chosen such

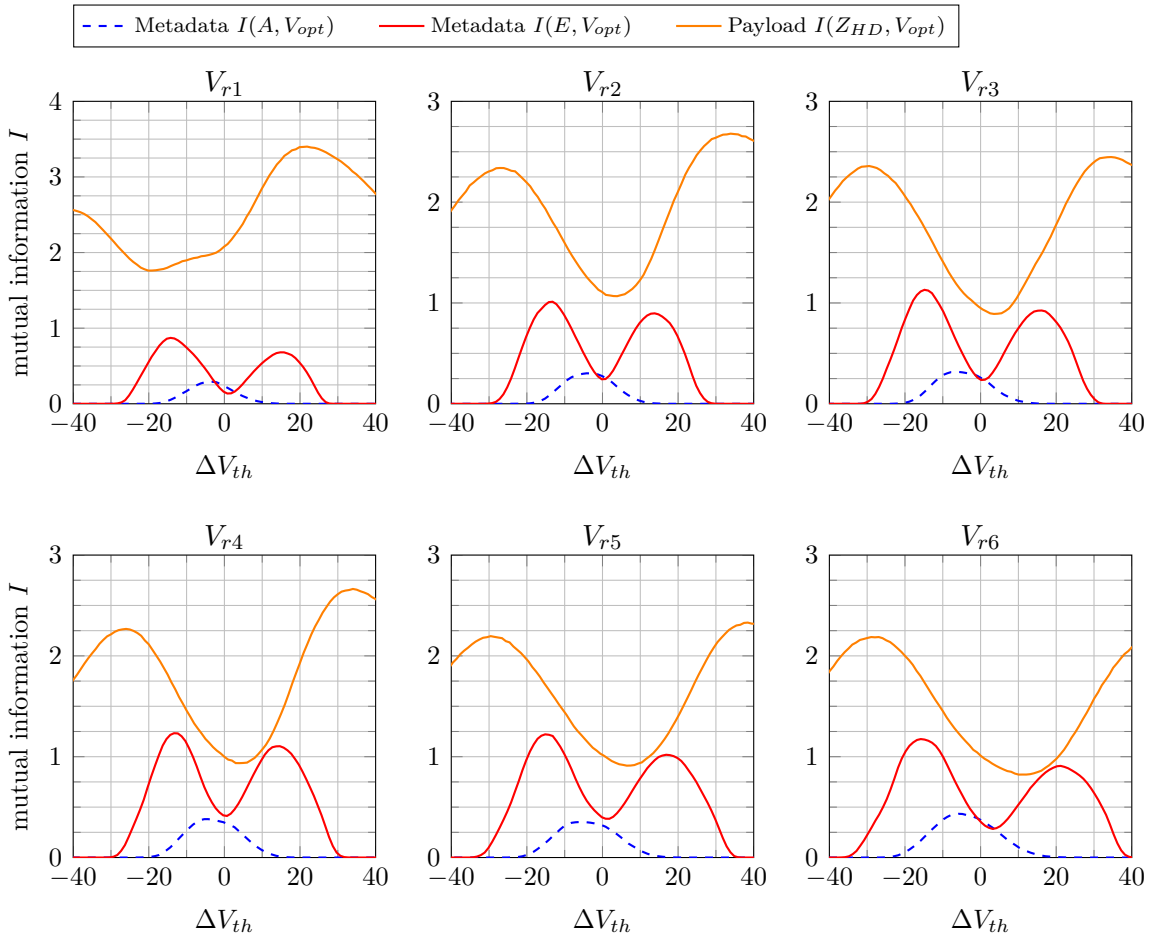


Figure 5.5: MI between asymmetry variable A and optimal RRV $I(A; V_{opt})$ on reference V_{r1} to V_{r6} in respect of metadata.
$I(E; V_{opt})$ - MI between the number of errors E and the optimal RRV of metadata.
$I(Z_{HD}; V_{opt})$ - MI between histogram counter Z and V_{opt}.

that they maximize the MI at the EOL case. This approach avoids adapting RRVs for individual cases, thereby reducing complexity. Simultaneously, suboptimal values are calculated, which replicates some errors in field applications. Employing the presumed measurements, the maximum MI readout at the EOL condition are used to derive the asymmetrical SD RRVs. Note that the calculation of MI to gain information about V_{opt} is presented for all reference voltages, except V_{r0} due to the inability to measure charge state S_0. The calculations consider all available measurements, and the x-axis of the figures are chosen such that $\Delta V_{th} = 0$ corresponds to the mean optimum RRV.

The MI $I(E; V_{opt})$ shows noticeably higher values compared to $I(A; V_{opt})$, underscoring the effectiveness of using the number of correctable errors from the metadata as a feature. When using the number of errors in the metadata, we attain the maximum information about the optimum reference when reading at approximately -14, i.e., a decrement of fourteen voltage steps from the mean optimal reference voltage. We can see that near the optimum, the MI

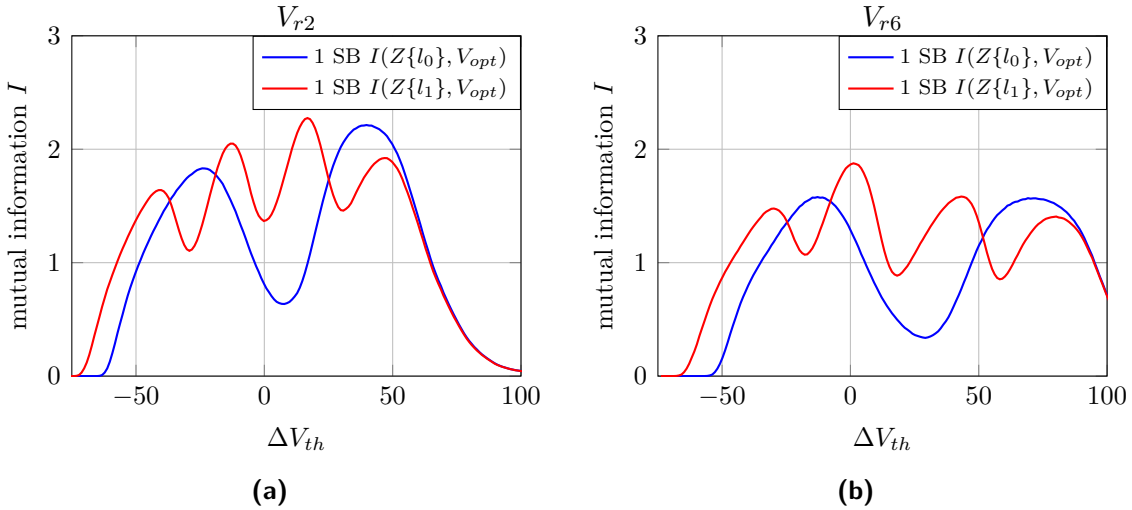


Figure 5.6: MI of histogram levels defined from the different intervals of one SB readout for **(a)** V_{r2} and **(b)** V_{r6} (adapted from [8], ©2023 IEEE).

is relatively low due to the inherent low entropy of the number of errors within that specific range. Reading substantially distant to the optimum ($|\Delta V_{th}| \geq 30$), we obtain no information about V_{opt} because the read data is incorrect. In the case of a decoding failure, the exact number of errors remains unknown. Near the optimum $\Delta V_{th} = 0$, we can see that in almost all cases, the binary variable A, denoting asymmetry, provides more information about V_{opt}, compared to E.

Considering the payload data to extract information about V_{opt}, we can observe in Figure 5.5 that the relationship between HD readout and SD changes regarding information content. Notice that the MI significantly increases, and the area with high information is more expansive. This demonstrates the substantial information gain resulting from increasing the number of histograms in the available features. However, the information gain comes at the cost of escalated readout complexity.

Comparing the MI derived from the payload data to that from the metadata readout, a notable difference lies in the broader range of high MI values. This implies that significant deviations from the mean optimal RRV yield substantial information regarding the precise location of the optimum RRV. In contrast to $I(E; V_{opt})$, specific values of $I(Z; V_{opt})$ for HD readout exhibit lower values, although it is evident that $I(Z; V_{opt})$ using SD readout yields the most substantial information. The analysis demonstrates that the feature derived from the error count in the metadata E is suitable to extract information about V_{opt}. Besides, the feature relying on the count of active cells per RRV interval Z based on the payload data holds significant importance in deducing information about V_{opt}.

In the following, we focus on the observation of the different intervals of the SD readout regarding one SB and two SBs as shown in Figures 5.1a and 5.1b. To discuss the influence of the individual intervals l_i we calculate the MI between the number of active cells in each

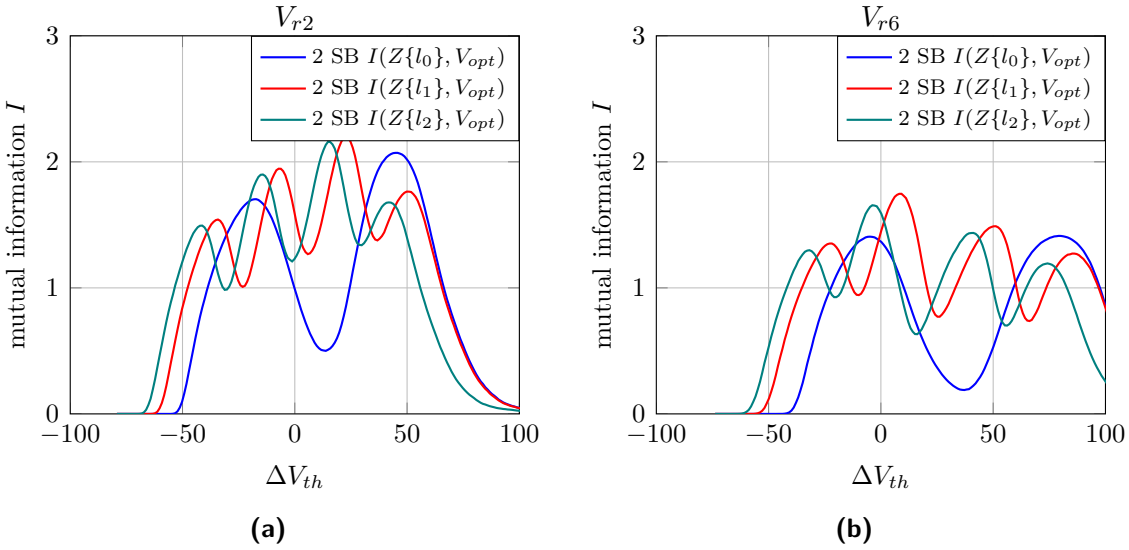


Figure 5.7: MI of histogram levels defined from the different intervals of two SB readout for **(a)** V_{r2} and **(b)** V_{r6} (adapted from [8], ©2023 IEEE).

interval $Z\{l_i\}$ and V_{opt}. Instead of discussing the MI for each RRV, we concentrate on the RRVs necessary to read out the MSB page of Gray Code 1 from Table 4.1, namely V_{r2} and V_{r6}. The MI of the other references has a similar behavior. If the quantization levels used are symmetrical, the MI regarding the left intervals from $\Delta V_{th} = 0$ are symmetrical to the MI regarding the proper intervals. As we use asymmetrical values, the results are not symmetrical to $\Delta V_{th} = 0$ but very similar. Therefore, we only consider the results regarding the left intervals of the mirror axis V_{ri}.

Figure 5.6 illustrates the results of the MI between V_{opt} and the histogram levels defined by the intervals of one SB, specifically $l_{i,0}$ and $l_{i,1}$, which correspond to the counters within the left section of the intervals along the mirror axis V_{ri}. The MI concerning the outer intervals l_0 and l_4, follow a similar trend to the MI derived from the HD readout. By contrast, the MI between counters of the inner intervals and V_{opt} display an oscillating behavior, where the minima have a higher magnitude compared to the minima of the MI of the counters in the outer intervals.

Next, in Figure 5.7 we review the MI results derived from the two SB readouts. To increase the clarity, we consider only the left section of the intervals along the mirror axis V_{ri}. This results in three MI curves. The MI derived from the proper section are mirrored, slightly shifted, and altered due to the asymmetric SD RRVs. As for the one SB case, the inner intervals of the two SB case $I(Z\{l_0\}, V_{opt})$ and its corresponding counterpart follow a similar trend to the $I(Z; V_{opt})$ curve of the HD readout. The MI between the inner counters and V_{opt} have an oscillating behavior as for one SB.

5.3 Read Reference Voltage Adaptation Algorithms

The variation in the optimal RRV V_{opt} across the pages of a flash memory block can have a substantial impact on the residual error rate. In the following, we present two methods for calibrating the RRV that aim to minimize the residual error rate.

The first method involves leveraging the ECC to extract information about the cell state. By analyzing the number of observed errors detected from the ECC, we can determine the number of errors and the asymmetry of the error distribution, i.e., the probabilities of 1-to-0 errors and 0-to-1 errors are typically asymmetric. Using this information, we predict the shift of the reference voltages. It is demonstrated that the observed number of errors is a good feature for the voltage shifts and is utilized for the initial calibration of the read thresholds. This calibration method can effectively minimize the impact of voltage shifts and optimize the read thresholds based on the observed error patterns.

Typically, one RRV set is applied on all pages of a block, due to complexity reduction. Unfortunately, these voltages may vary from page to page of the same block, even when all pages have the same life-cycle condition. It is common to read multiple pages concurrently within a block when utilizing flash memory for user data storage, especially given that the data size within a single page is typically comparatively small, e.g., between 512 and 4096 bytes or more, depending on the manufacturer.

The second method discusses a tracking approach, which reduces the latency associated with threshold voltage adaption when reading multiple pages sequentially from the same flash memory block. This approach exploits the information from the previous page to adjust the RRV, enabling an adaptive adjustment for subsequent pages within the same block.

5.3.1 Calibration of the Read Reference Voltages

This section discusses a data-driven calibration method designed to predict the optimal threshold voltage without prior knowledge of the life-cycle state of the flash memory. This approach exploits the observed number of errors within the codeword for the metadata. Typically, the error correction capability of the metadata is stronger than that of the user data, commonly referred to as payload data. The metadata is constructed to remain correctly readable even if the payload data is unrecoverable. This is achieved by using a lower code rate, ensuring that more readout errors are successfully correctable.

For the calibration method, the read voltage V_{cal} of the initial readout is selected to maximize the MI between the number of errors and the optimal threshold voltage over all data sets. Consider Figure 5.5 for reference V_{r2}, the read voltage V_{cal} that provides the most information is located at $V_{th} = -14$. Since the MI varies with each reference voltage, V_{cal} must be determined for each reference.

The following steps outline the algorithmic process. Initially, the metadata is read at the reference voltage V_{cal}. Subsequently, this information is decoded. If the decoding process

fails, a read-retry mechanism is initiated. At this point, the initial reference voltage V_{cal} is adjusted, and the metadata is read and decoded once again. In this second fitting step, the MI is calculated using the pages where metadata was not correctable using V_{cal}. The reference voltage that maximizes the MI between the number of errors and the optimal RRV is then chosen for the subsequent iteration.

If the decoding procedure is successful, the number of errors is revealed, and this number is used to adjust the RRV of the payload data. The new RRV is obtained from a look-up table, which specifies the voltage offset corresponding to a given number of observed errors. The composition of the table will be discussed later on. For the measurements used in this thesis, a maximum of two readings were sufficient for the calibration method. The maximum number of read operations mostly depends on the error correction capability of the specified ECC.

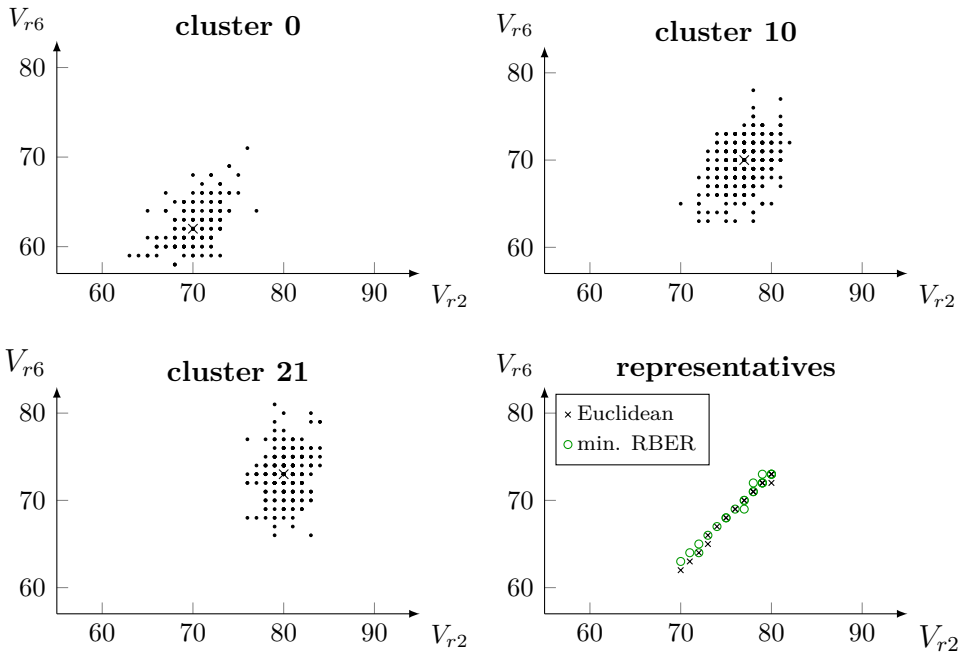


Figure 5.8: Exemplary clusters to analyze the RRV for the calibration table. The axis represents the discrete threshold voltage concerning the lower charge state regarding the reference voltage ([3], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/).

A look-up table is suitable for adjusting the optimum voltage based on the number of observed faults, accelerating the fitting process. The look-up table generation implies clustering the data sets of the measurements based on the number of errors in the metadata when reading at the calibration voltage V_{cal}. This approach is similar to the well-known k-means clustering of [76] but does not require iterative updates as the partition of the data set is based on the observed number of errors on each page. The error correction capability t of the code determines the number of clusters required for the look-up table, resulting in $t + 1$ clusters. To obtain the representatives of each cluster (i.e., the cluster centroid), we minimize the Euclidean distance to all samples in the cluster, similar to the approach used in k-means clustering.

In the subsequent analysis, we adopt the bit-labeling from Gray code 1 detailed in Table 4.1. The MSB page is determined by the reference voltages V_{r2} and V_{r6}, the CSB page by V_{r1}, V_{r3}, and V_{r5}, and the LSB page by V_{r0} and V_{r4}.

Figure 5.8 presents the clustering results for the MSB page, defined by references V_{r2} and V_{r6}, in scenarios with 0, 10, and 21 errors in the metadata. The plot displays the optimal threshold voltage for reference V_{r6} against the optimal threshold for reference V_{r2} for each observed pair of optimal voltages. The points on the plot represent the observed pairs of optimal voltages corresponding to a specific number of errors in the metadata. The small crosses in the figures indicate the representatives of the three clusters resulting from different error counts. As shown in Figure 5.8, the clusters are not distinctly separated, which can be attributed to the limited number of bits in the metadata and the subsequent high variability in the number of errors.

The lower right-hand figure shows the representatives of all clusters obtained by minimizing the Euclidean distance or minimizing the average residual bit error rate (RBER) within the cluster. Both approaches achieve comparable outcomes. Given that V_{cal} is typically lower than V_{opt}, as shown in Figure 5.5, a higher error count results in the prediction of a higher threshold voltage for the representative.

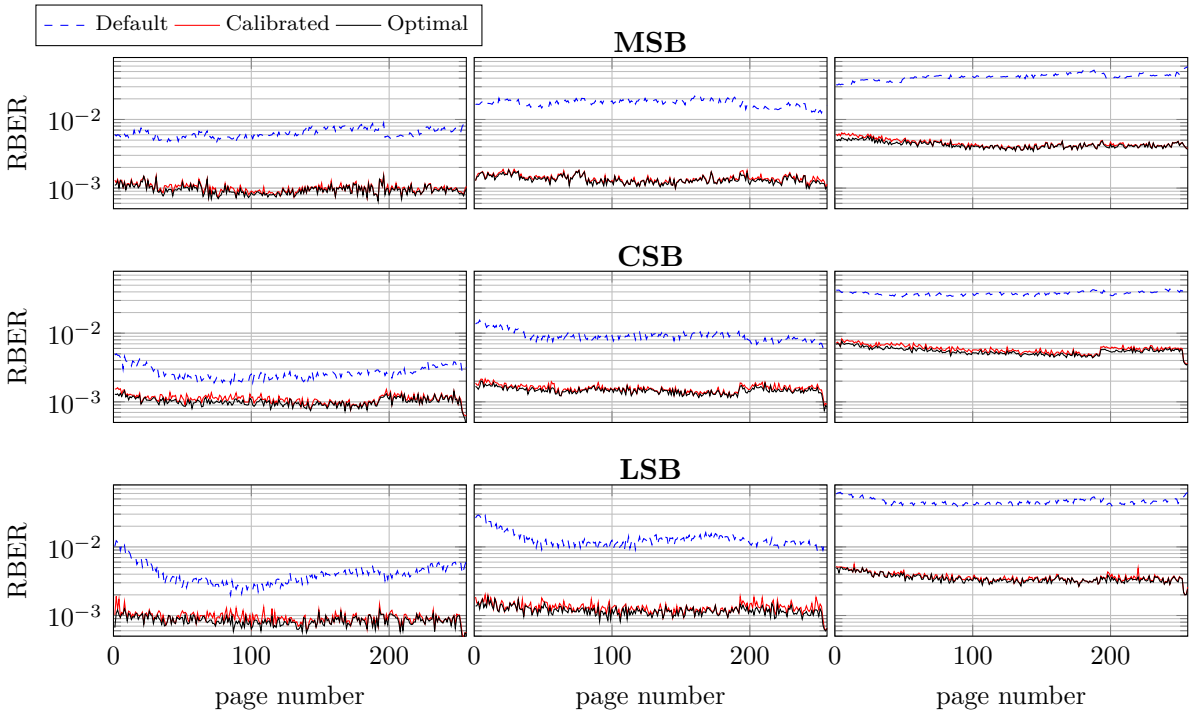


Figure 5.9: RBER of proposed calibration compared to minimum RBER for the bit pages regarding the labeling from Gray Code 1 (Table:4.1) for different life-cycle states. Left: 1500 P/E cycles and $1.4{\cdot}10^3$ h DRT; middle 1500 P/E cycles and $8.2{\cdot}10^3$ h DRT; right 3000 P/E cycles and $8.2{\cdot}10^3$ h DRT (EOL) (adapted from [3], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/).

Subsequently, Figure 5.9 illustrates calibration procedure results for three measured life-cycle states. For calculating the RBER for each page, the Gray code 1 bit-labeling from Table 4.1 is considered. In this context, the x-axis denotes the page number, while the y-axis represents the RBER when reading at the default threshold voltage $V_{default}$, which corresponds to the optimum threshold voltage for each page. It also includes the RBER using the calibration algorithm for each page, and the minimum possible RBER achievable with the optimal RRV. Each data set comprises 256 pages. The left column corresponds to $1\,500$ P/E cycles and $1.4{\cdot}10^3$ h DRT (13 h baking), the figures in the middle consider $1\,500$ P/E cycles and $8.2{\cdot}10^3$ h DRT (80 h baking), and the right figures correspond to $3\,000$ P/E cycles and $8.2{\cdot}10^3$ h DRT (EOL). The default RRV remains constant and is only suitable for fresh flash memories, which is not presented in this figure since calibration is typically unnecessary. The optimal RRV is individually calculated for each page based on the measurements.

Through calibration, the RBER is very close to the minimum possible error rate. Although the calibrated RRV does not necessarily achieve the optimum for each page, this method provides sufficiently accurate calibration. However, it requires reading the metadata up to two times and changing the threshold voltage up to three times, contributing to a relatively high overall latency.

The question emerges concerning which performance is reached by applying an ECC. Therefore, we consider a regular LDPC code with the code rate $R = 0.9$ and dimension $k = 16\,384$ for 2kB payload data. For a RBER below 0.0088, the decoder proposed in [77] achieves a FER of less than 10^{-9} with soft-input decoding employing three SBs (five readout voltages for each reference). Using hard-input decoding FER of 10^{-9} is achieved at a RBER of 0.0038.

At the EOL state, the minimum RBER without calibration is around 0.03, exceeding the error correction capability of the decoder, even with soft-input decoding. However, with calibration, the maximum observed RBER value among all pages in the EOL data is approximately 0.008, ensuring a FER of less than 10^{-9} even under worst-case conditions. Nonetheless, soft reading is time-consuming. Otherwise, if only hard-input decoding is employed, a RBER of 0.006 would lead to a FER of approximately 0.01 due to the decoder, which is unacceptable. Thus, calibration becomes crucial to achieve reliable performance under challenging conditions.

The default threshold voltage remains constant and is only suitable for fresh flash memories. This is not considered here because typically calibration is on fresh flash memories is unnecessary. For instance, consider the first life-cycle state shown in Figure 5.9. Without calibration, around 22% of the pages have a RBER higher than 0.006. After calibration, the highest RBER value in the first life-cycle state is approximately 0.002, leading to a FER below 10^{-9} with hard-input decoding. Thus, calibration reduces the number of pages that require soft-reading in mid-life conditions, subsequently leading to reduced read latency and power consumption for both the read operation and decoding process.

In order to minimize the complexity of the calibration, an algorithm is presented in the next section, which is designed to adapt the threshold voltages of successive pages within a block.

5.3.2 Tracking the Optimal Read Threshold

The variation in the optimal readout threshold voltage V_{opt} across the pages of a flash memory block can substantially affect the residual error rate. Due to complexity considerations, a single set of RRVs is commonly employed for all pages within a flash memory block. However, this approach can result in higher residual error rates due to the varied optimal threshold voltages for different pages.

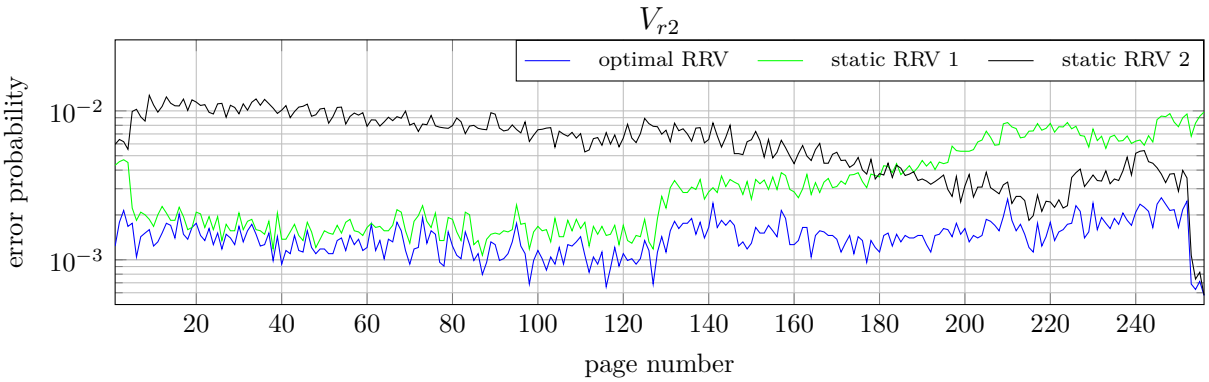


Figure 5.10: Error probability of a TLC block on V_{r2} on a block with the life-cycle ($5.8{\cdot}10^3$ h DRT, 3000 P/E) for optimal readout and two worst-case static readouts (adapted from [3], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/).

Figure 5.10 illustrates the error probability for each page within a single TLC block at the reference V_{r2}, considering a scenario with $3\,000$ P/E-cycles and 55 h of baking, which corresponds to approximately $5.8{\cdot}10^3$ h DRT. Note that the error rate at a particular reference is not equivalent to the page-level RBER, as each page uses additional references for readout.

Each page in the flash block exhibits a specific readout threshold voltage V_{opt} at which the lowest number of errors is observed. Unfortunately, these optimal voltages can vary from page to page within the same block, even when all pages share the same life-cycle condition. This phenomenon is clearly depicted in Figure 5.10.

The curve labeled as *optimal readout* indicates the minimum achievable error probability for each page using the individual V_{opt} as RRV. The other two curves represent static threshold voltages, which are in the specific set of possible V_{opt}. Therefore, the error rate is minimized in a specific block range. However, even if one optimal RRV in the block is found correctly, applying it to the entire block may lead to high residual error rates. This highlights the need for more adaptive methods to minimize errors across the pages of a flash memory block.

For the curve *static 1*, the threshold voltage was calibrated to the end of the block, and for curve 2, the threshold voltage was calibrated to the beginning of the block. We observe that the static threshold voltages lead to significantly higher BERs.

The previous example shows that constant reference voltages for a complete block can cause considerable performance degradation. To cope with this problem, we propose a tracking

procedure for the reference voltages for consecutive pages within a block. As shown in Figure 5.3, the optimal threshold voltage corresponds to a balanced error probability between the possible error conditions given by the left and right error probabilities of the distributions on a read reference voltage. These are described by the error between the written w and the read data r, i.e., with binary data, this corresponds to the probabilities $P(r = 1|w = 0)$ and $P(r = 0|w = 1)$. We can call this type of error 1-to-0 errors and 0-to-1 errors.

The binary variable A indicates the direction of the asymmetry between 0-errors and 1-errors. $A = 1$ indicates more 1-to-0 errors and $A = 0$ indicates more 0-to-1 errors. Hence, A also provides the direction for the voltage change toward the equilibrium of the error probabilities. In theory, the number of binary data on a page is approximately equally distributed. This follows from data randomization algorithms to minimize the error rate [78]. This results in an approximately equal distribution of the data.

Considering the MI between the asymmetry bit A and V_{opt} with respect to the user data, asymmetry is an excellent feature to estimate the optimal voltage. For the proposed algorithm, we assume that we have successfully calibrated the first page of a block.

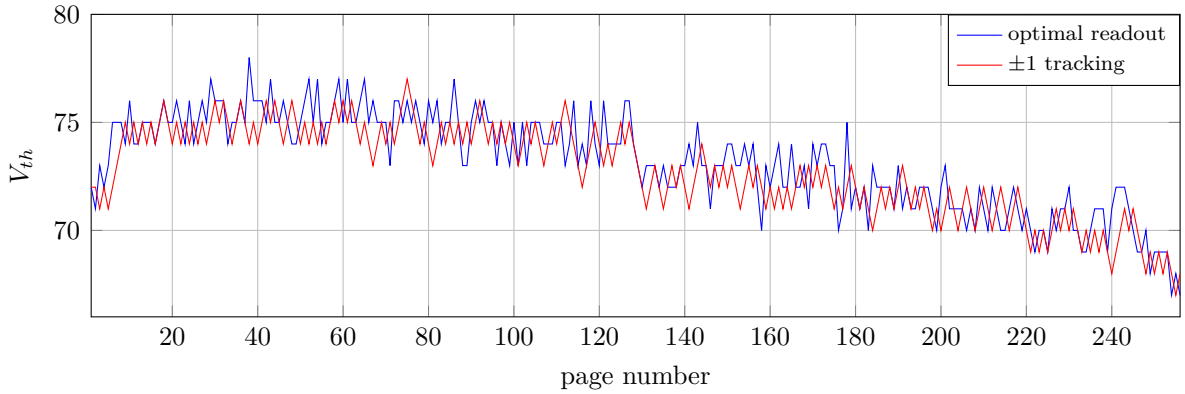


Figure 5.11: Threshold voltage adaptation of a TLC block on reference V_{r2} ($5.8{\cdot}10^3$ h DRT, 3000 P/E) (adapted from [3], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/).

The tracking approach considered in the following adjusts the threshold voltage for the subsequent page by either increasing or decreasing it by a single voltage step, depending on the value of A (asymmetry). This concept is illustrated in Figure 5.11, which depicts the adaptation of threshold voltages across one flash memory block for reference V_{r2}. The blue line represents the optimal threshold voltage values (V_{opt}) for each individual page. The red curve represents the results obtained with the tracking method, where adjustments of ± 1 voltage steps are applied based on the value of A. As we can observe, the tracking approach closely follows the optimal voltage values, indicating that the threshold adaptation effectively tracks changes in the optimal threshold voltages across pages.

The equilibrium of the error probabilities does not always coincide with V_{opt}, as indicated in Figures 5.3 and 5.5. For instance, the equilibrium of the error probabilities corresponds to the value $\Delta V_{th} = -3$ in Figure 5.2b. In such a case, the tracking towards the equilibrium results

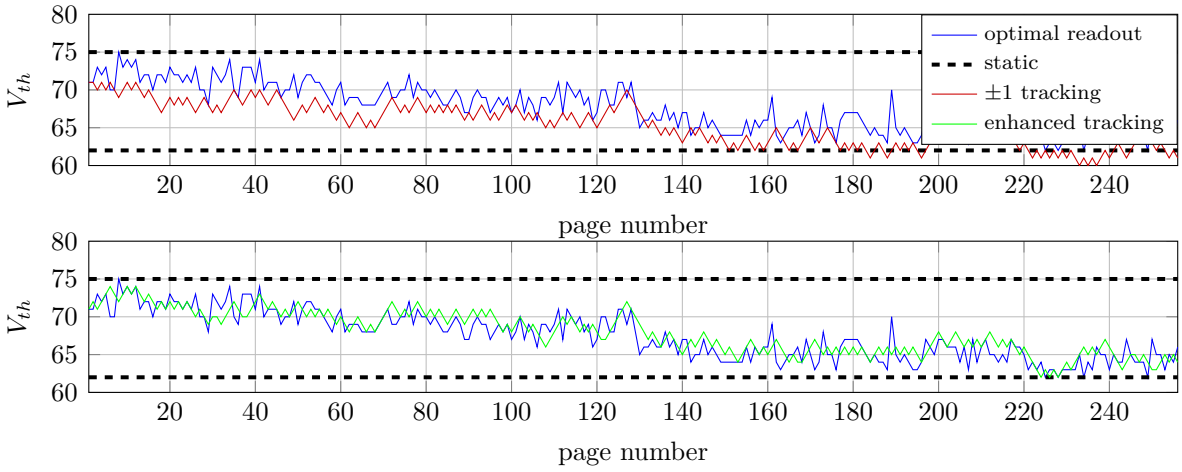


Figure 5.12: Threshold voltage adaptation of a TLC block on reference V_{r6} ($5.8{\cdot}10^3$ h DRT, 3000 P/E) (adapted from [3], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/).

in a bias, i.e., the estimated voltages systematically overestimate or underestimate V_{opt}. This bias effect is displayed in the top part of Figure 5.12, which shows the adaptation of the threshold voltages across one block for reference V_{r6}. The dashed black lines correspond to the static voltages from Figure 5.10, and the blue line shows the values of V_{opt} for each page. The red curve depicts the results with tracking, where ± 1 voltage steps are applied. We can observe that the adaptation follows the optimal voltage values but slightly diverges from the optimal curve, where V_{opt} is mostly underestimated. On average, the bias corresponds to the shift $\Delta V_{th} = -3$ of the equilibrium.

In such a case, an enhanced tracking algorithm can be applied that corrects this bias. Instead of changing the value of A at the equilibrium of the error probabilities, we calculate the rate of error number corresponding to the optimal voltage V_{opt} or the shift $\Delta V_{th} = 0$, respectively. In Figure 5.2b, the number of 1-to-0 errors is about four times the number of 0-to-1 errors. In this case, we can choose the value of A such that $A = 1$ indicates four times more 1-to-0 errors than 0-to-1 errors.

To average the ratio over different life-cycle states, we choose the ratio that corresponds to the value $\Delta V_{th} = 0$ for the MI $I(A; V_{opt})$. The lower part of Figure 5.12 illustrates the threshold voltage adjustments with this approach. As a result, the adaptation becomes closer to the optimal voltage V_{opt}. This enhanced tracking algorithm helps to correct the bias introduced by the equilibrium of error probabilities and further improves the accuracy of the threshold voltage adjustments.

Figures 5.11 and 5.12 show results for one block in a single life-cycle state. Figure 5.13 demonstrates the tracking accuracy averaged over different life-cycle states. In this box plot, the median of the error probability at V_{r6} is represented as the central marker, and a box represents the interquartile range, i.e., the interval for 50% of all measurements when ordered from lowest to highest error probability. The whiskers represent the most extreme data points

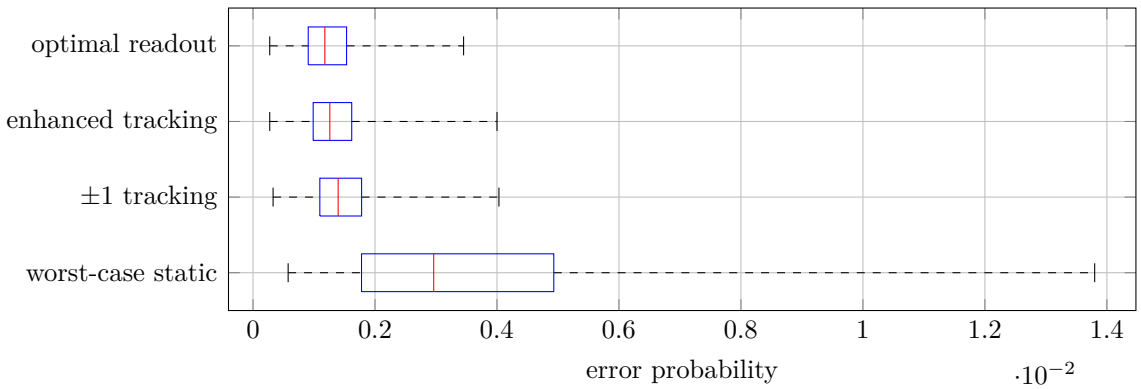


Figure 5.13: Error statistic on reference V_{r6} of all pages of 27 blocks with $5.8{\cdot}10^3$ h DRT 3000 P/E ([3], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/).

without outliers.

The box plots consider the simple ± 1 voltage adaptation, the enhanced adaptation, and the worst-case static readout voltages. For every block in the data set, the worst-case readout voltage was selected within the range of optimal readout voltages. Hence, these values correspond to an optimal readout voltage of one page within a block, which was selected as the static readout voltage for the entire block. The results in Figure 5.13 show that a static readout voltage can significantly increase the error probability compared with the optimal voltage.

In the following, we consider the LDPC code used in Section 5.2.2 for the discussion of the ECC performance. We discuss results for the FER using hard-input decoding. For a RBER of 0.006, the hard-input decoding results in a FER of about 0.01. Therefore, we assume that a soft-read operation is required for all RBER values greater or equal 0.006. For the worst-case static readout voltage, many pages exceed the RBER of 0.006. When using the tracking approach, the maximum RBER is about 0.004, which results in a FER of less than 10^{-9} with hard-input decoding. As a result, the tracking improves the latency and power consumption by reducing the probability of soft-read and soft-input decoding procedures.

The ± 1 adaptation and the enhanced adaptation obtain a similar performance that is close to the optimal read voltage. Consequently, the bias compensation is not critical concerning the achievable error rate. However, the bias compensation may be required for soft-input decoding [60, 61]. Many soft-input decoding algorithms assume quantization intervals that maximize the MI between the input and output of flash memory, which requires an accurate voltage adaptation.

5.3.3 Discussion on Proposed Adaptation Algorithms

The presented adaptation method can be divided into two components. The first part involves a calibration approach, aiming to find a suitable reference threshold with a minimal number of read trials when we have no knowledge about the life-cycle state. In the second part, a tracking

approach is introduced based, leveraging a stepwise channel estimation method to dynamically adjust the reading threshold while iterating over successive pages, i.e., we calibrate the first page of a block and successively read other pages of the same block. This tracking process involves making only minor changes to the RRV, since adjacent pages in a block usually have similar distributions. On the other hand, a tracking algorithm should be as time-efficient as possible, while calibration may require multiple reading and voltage adjustment steps, which is quite time-consuming. Since we have some knowledge about the life-cycle state, we want to use this information to update the reference without additional reads.

The proposed methods are also motivated by observations on MI. We use the number of errors in the metadata for calibration because it provides information about V_{opt} over a relatively wide range of read voltages V_r. On the other hand, we propose a tracking method based on the asymmetry of 0 and 1 errors, because the asymmetry has higher MI for voltages near the optimum, where the total number of errors provides little information.

In this section, we have introduced two strategies to dynamically adjust read thresholds based on the observed errors in the codeword of an ECC. We utilize statistical features derived from the total number of errors in the codeword and the counts of 1-to-0 and 0-to-1 errors. These features are explored through the lens of MI analysis, which quantifies the relationship between the optimal read voltage and the measured error characteristics.

The total error count provides insights into the proximity of the current threshold voltage to the optimal value, while the error asymmetry offers information about the direction of the shift. Leveraging the total error count enables predicting the voltage shift required to calibrate read thresholds. The second method involves a gradual channel estimation approach to accommodate threshold calibration discrepancies and voltage shifts across various pages within the same block. This technique capitalizes on the insights from asymmetric error distributions. By applying these approaches, substantial reductions in the RBER of NAND flash memories are achievable. Our conclusions are substantiated through numerical results based on extensive flash memory measurements.

5.4 Reference Voltage Adaptation with Neural Networks

The RRV calibration method presented in the previous section requires a readout of the metadata. While this algorithm maintains a relatively low complexity, it requires at least one readout of the metadata, which introduces additional complexity. Consequently, executing this algorithm several times results in elevated temporal and energy complexities.

This section describes an alternative approach for adapting the RRV using NNs. The application of NNs for different signal processing problems is particularly suitable. Machine learning techniques have already been applied to various issues related to flash memory, e.g., estimating the life-cycle states of flash devices [79]. Lee et al. [80] investigated an approach to predict process variations of flash devices. Zambelli et al. [81] characterize flash memory in different

life-cycle states to adapt error correction strategies. These methodologies can be trained using empirical data, avoiding the complex task of creating accurate models.

The correct modeling of TLC flash cells is complex since many different physical effects are considered to lead to a multidimensional problem. Moreover, flash cells from different manufacturers exhibit variations in tolerances and manufacturing processes, resulting in various side effects. For these reasons, good models are not available. Various methods for estimating the flash channel using NNs have been proposed, such as in [69, 70]. In [69], an NN approach is proposed based on meta-information and ECC decoding, i.e., the knowledge of the number of P/E cycles, the FER, the average number of iterations of the LDPC decoder, and the bit flip ratio of the decoder are used as features. This method shows promising results but requires metadata and therefore needs additional readouts. Mei et al. propose a different machine-learning approach using a recurrent NN to dynamically estimate the RRV of MLC flash memory [70]. The data for training and verification are generated by a theoretic GMM. The method is based on a deep learning technique in combination with the ECC decoder. The adaption procedure incurs additional read latency and power consumption and is proposed as a background operation when the system is idle.

In order to reduce the average read and decoding latency, [71] presented an approach to predict the optimal decoding method called dynamic error recovery flow. This method does not predict optimal RRVs; instead, it assumes optimal RRVs. The underlying NN is designed to be reusable for different flash memories with different process variations. The chosen flash model is a Gaussian channel and noise model. The input features required for the NN are the on-cell ratio, the P/E cycles, the word line, and the page index. The on-cell ratio defines the ratio between active and total cells at a fixed predefined RRV. Note that for the classification in [71], two on-cell ratios at different RRV are required.

Subsequently, we delve into the basics of NNs, exploring their applications in flash memories and providing pertinent examples, after which the training data set used in this work is presented. Following that, we introduce an initial NN approach to demonstrate its generalization capabilities. This approach leverages data augmentation (DA) on a limited number of measurements to achieve effective adaptation across different life-cycle states. Finally, we introduce a NN approach that exploits sparse histograms.

5.4.1 Basics of Neural Networks

NNs attempt to emulate the function of the human brain by using several interconnected nodes organized in a layered structure to solve complex data-driven problems. NNs have proven to be extremely useful in various fields, especially in solving signal processing problems. Consequently, several network architectures and training functions have been developed [82, 83]. The neurons are trained with certain training methods to react to inputs according to the learned data and produce an output. The word "neuron" is employed due to the analogous nature of this process to that of the human brain.

NNs is particularly suitable for data prediction and classification. One of the classic applications of NNs is image classification, a task in which the necessary training data can be considerably large, depending on the complexity of the distinctions. In this context, the term "feature" is described as characteristic variables or attributes of the data set that are used as inputs for the NN.

The quality of these features plays an essential role in achieving accurate results, and the number of features can influence the computational complexity. Moreover, the complexity of a NN also depends on the number of neurons and layers. The number of features depends on the target, i.e., characteristics needed for accurate differentiation.

The two basic learning methods for machine learning are supervised and unsupervised learning. Supervised learning employs labeled training data to classify input data. This process trains the network to classify new input data into predetermined categories. By contrast, unsupervised learning explores data patterns and relationships independently and without predefined categories or solutions to problems. In this work, we apply the supervised learning approach, i.e., input values are associated with target values during training.

A typical NN comprises an input layer, one or several hidden layers, and an output layer. Each layer contains an independent number of neurons, and the connections between layers differ depending on the network type. In our context, the resulting NNs should provide near-to-optimal RRVs V_r. Various features are possible to accomplish this task. We will analyze different features as input data based on the TVD of TLC flash memories. In the presented cases, NNs learn the complex signal characteristics of the flash cells. Using threshold voltage distributions, accurate predictions of suitable reference voltages are possible.

To address this challenge, we apply a feedforward neural network (FNN). Various analysis showed that this architecture is sufficient for this task, as FNNs are commonly used to solve classification and regression problems. The FNN is characterized by complete connectivity. All neurons between adjacent layers are completely interconnected without using any feedback loops. As input, we use the current RRVs and the sparse histograms that are mapped to the range $[\text{-}1, 1]$. This normalization is a standard practice, ensuring that inputs are compatible with sigmoid activation functions. The normalization prevents rapid saturation and accelerates the training process.

Afterwards, the input vector $\mathbf{x} = [x_1, ..., x_n]$ is manipulated between each layer with the weighting matrix $\mathbf{W}$ and the bias vector $\mathbf{b}$ for each neuron j such that

$$a_j = \sum_{i=1}^{n} W_{i,j} \cdot x_i + b_j. \tag{5.3}$$

Each hidden neuron uses a hyperbolic tangent sigmoid transfer function as activation function

$$z_j = \text{tansig}(a_j) = \frac{2}{1 + \exp(\text{-}2a_j)} - 1, \tag{5.4}$$

where the output value is limited to $z_j : \mathbb{R} \to (-1, 1)$. After the hidden layers, the outputs

are connected to the output layer. This layer uses a linear transfer function to map the results to the required numeric range. As final output of the FNN, we get the estimated RRVs.

The weighting factors, typically determined during training, are commonly established using well-defined training functions. One common training algorithm that we will use in this work is the Levenberg-Marquardt (LM) algorithm described in [84]. The weighting factors of the NN are changed by comparing the target and actual output values of the FNN until the error between these values is minimized. The LM algorithm is a well-suited approach for backpropagation training that combines the gradient descent and the Gauss-Newton method. It is theoretically justified, easy to implement, and has proven robustness and efficiency in practice [85, 86]. Further details about NNs using LM training algorithms are described in [83].

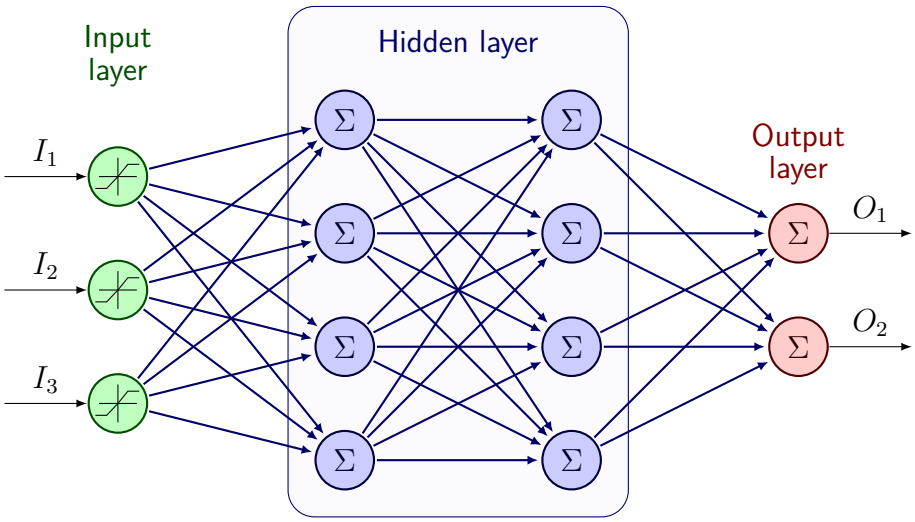


Figure 5.14: Diagram of a FNN with three input neurons, two hidden layers with four hidden neurons each, and two output neurons.

A diagram of such a FNN with three input neurons, two hidden layers with four neurons each, and two output neurons with all connections is shown in Figure 5.14. In general, this FNN approach has a storage overhead of $N(I_n + 1)$, where N is the number of neurons of a layer and I_n is the number of inputs. The specification of the total complexity in terms of floating point operations (FLOPs) strongly depends on the implementation of the mapping, the *tansig*, and the linear transfer function for the corresponding input, hidden and output layer. Ignoring these special functions, the number of floating point additions and multiplications required corresponds to the storage overhead.

Acquiring sufficient data to train and test the NN effectively is crucial. In order to obtain the necessary training data covering all possible life-cycle states, numerous measurements must be performed. However, collecting the data is extraordinarily resource-intensive and time-consuming. This practical constraint underscores the need to enhance the generalization capability of the FNN. The question arises concerning which measurements are required to adequately train the NN threshold adaptation for all relevant life-cycle conditions.

The overarching goal is to proficiently predict RRVs using histograms from life-cycle conditions that were not part of the training data. To achieve this, the FNN should demonstrate robust generalization, performing well on previously unseen data. Finding the right balance is essential, as excessive training data can lead to overfitting. Overfitting implies that the NN fits the training data closely but struggles to perform correctly with other data.

One approach to mitigate overfitting and enhance generalization is to introduce controlled noise into the training data. Conversely, the opposite phenomenon, called underfitting, can occur when the model is too simple to capture the underlying patterns in the data. Rigorous testing and verification of the performance of NN is essential and usually performed with independent testing sets.

In the subsequent section, we will present in detail the available training data for the NNs approach. To mitigate the risk of overfitting and minimize the measurement requirements, we will use as little training data as possible afterwards to demonstrate that specific measurements are a sufficient training basis.

5.4.2 Training and Test Data

The following describes the underlying data set used for testing and training purposes. Numerous physical effects change the TVDs of flash memory. The main disturbance factors and their effects have already been discussed in Chapter 3. We focus on specific disturbance effects. For the analysis in this research, we conducted measurements under different life-cycle conditions. We consider an industrial BiCS TLC NAND flash memory with a defined EOL with $3\,000$ P/E cycles and one-year DRT. For this study, we consider P/E, DRT, RD, and cross-temperature effects. The complete data set comprises 185 measurements over specific life-cycle conditions.

One measurement comprises one flash block with 256 word lines, i.e., 256 pages per block. Data is available for 0 to $3\,000$ P/E cycles in steps of 200 P/E cycles. DRT is simulated with a baking process where 83 h of baking correspond to one year DRT. Measurements with $0\ h$, $13\ h$, $27\ h$, $42\ h$ and $55\ h$ baking time are available, which corresponds to approximately $0\ h$, $1.4{\cdot}10^3\ h$, $2.8{\cdot}10^3\ h$, $4.4{\cdot}10^3\ h$ and $5.8{\cdot}10^3\ h$ data retention. The temperatures used for cross-temperature measurements are -$35\,^\circ$C, $25\,^\circ$C and $85\,^\circ$C, where $25\,^\circ$C is room temperature. This temperature range is nearly compliant with the AEC-Q100 Grade 3 qualification for automotive applications. The measurements without a cross-temperature effect were carried out at room temperature. Most measured sets have a combination of several effects.

Due to physical disorders, the optimal voltages are not identical for each page, block, or cell. Moreover, each life-cycle condition has a different set of ideal reference voltages, i.e., the reference voltages V_{opt}, which lead to the lowest error probability. The voltage setting during readout is performed using discrete voltage steps since the controller can set only discrete voltages. Consequently, all measurements and results will be presented in discrete voltage steps to ensure precision. We will exclude the charge state S_0 from our following

Table 5.1: Average optimal discrete RRV for different life-cycle conditions ([7], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/).

	life-cycle condition					optimal read reference voltage step					
test set	P/E	DRT	RD	PTMP	RTMP	V_{r1}	V_{r2}	V_{r3}	V_{r4}	V_{r5}	V_{r6}
1	800	0	no	25 °C	-35 °C	142	202	261	322	384	449
2	800	0	yes	85 °C	-35 °C	142	202	261	322	384	449
3	1000	0	no	-35 °C	85 °C	154	219	286	356	424	489
4	1400	0	yes	-35 °C	85 °C	152	215	280	349	416	481
5	1600	0	yes	-35 °C	25 °C	158	223	290	357	425	490
6	1800	0	no	-35 °C	85 °C	148	210	272	339	405	472
7	2200	0	no	-35 °C	25 °C	151	214	277	342	408	477
8	2400	0	yes	-35 °C	85 °C	156	222	290	360	429	492
9	2600	0	no	85 °C	-35 °C	139	199	261	322	385	449
10	3000	0	no	25 °C	25 °C	142	204	265	331	397	464
11	3000	$2.8{\cdot}10^3$ h	no	25 °C	25 °C	142	201	260	321	383	451
12	3000	$5.8{\cdot}10^3$ h	no	25 °C	25 °C	136	191	250	310	370	434
	maximal difference					20	31	40	50	59	58

analysis, as it corresponds to the deleted state and typically cannot be measured correctly using commercially available flash devices.

Table 5.1 shows the mean optimal discrete RRVs, i.e., V_{opt}, for exemplary blocks with different life-cycle conditions. V_{opt} leads to the average lowest error probability. Note that V_{r0} is ignored because S_0 is ignored in the analysis.

Figure 5.15 shows the TVD of two life states, according to test set 8 and 12, showing the different variances and mean values. 200 pages are considered for these TVDs. The x-axis represents a discrete voltage step, and the y-axis represents the number of cells switched on. Note that S_0 represents the deleted state. The measurement setup does not allow measuring the distribution of S_0.

5.4.3 Data Generalization through Data Augmentation Techniques

Numerous RRV adaptation methods for flash memory are based on data-driven approaches. In such strategies, model generalization is an essential topic, reflecting the ability to respond correctly to previously unseen data and make accurate predictions. Overfitted NN models generate accurate results within their training data, but their predictions are often faulty when confronted with unseen data.

For the adaptation of RRVs to various life-cycle conditions, we employ a FNN as described in Section 5.4.1. The training and the RRV adaptation are based on histogram measurements. The primary focus is not on implementing the RRV adaptation but on the model generalization. A conventional strategy to avoid overfitting and improve generalization involves DA, which

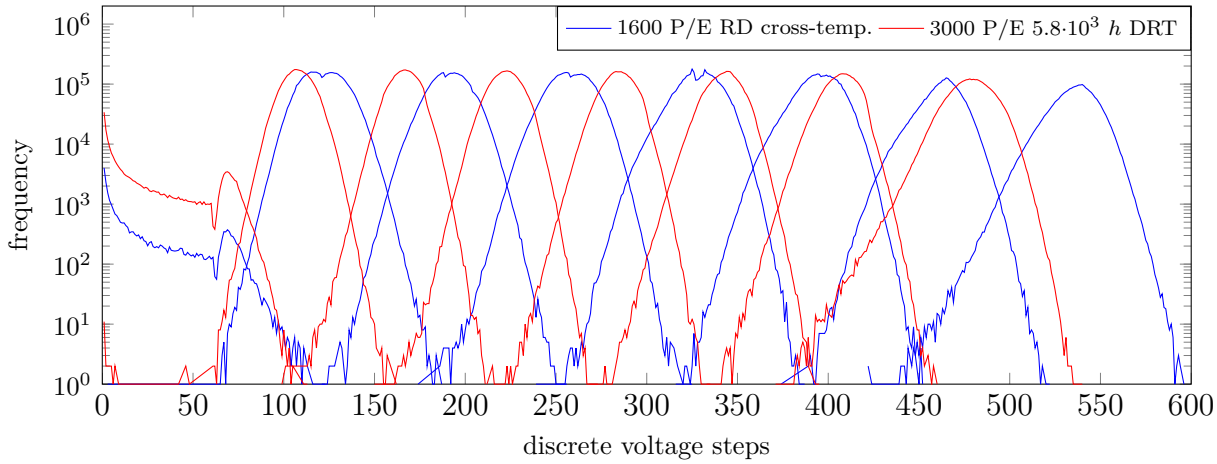


Figure 5.15: TVD for two measurements: blue curves correspond to 2400 P/E cycles, no DRT, RD, -$35\,°$C PTMP, $85\,°$C RTMP; red curves correspond to 3000 P/E $5.8{\cdot}10^3$ h DRT, no RD, $25\,°$C PTMP, $25\,°$C RTMP. 200 pages are considered per TVD (adapted from [7], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/).

generates new data by modifying existing data. We propose a DA approach for training the reference voltage adaptation in the following.

As shown in Figure 5.15, the TVD and the optimal RRVs of flash memory shifts under varying life-cycle conditions. This phenomenon allows us to replicate additional life-cycle conditions through shifting. To expand the training data set, we proportionally shift the TVDs proportional to the shift of the charge states and recalculate the optimal RRVs. This implies that charge state S_1 is shifted less drastically compared to S_7. We aim to reduce the range of required measured life-cycle conditions for training by emulating the data from other conditions through histogram shifting.

The training of the NN is based on noisy histogram measurements. The data set involves page-wise measurements since this corresponds to the smallest readable unit for NAND flash. However, these individual measurements at the page level provide noisy data, which does not allow the straightforward determination of the optimal RRV.

As an example, Figures 5.16a and 5.16b illustrate TVDs depicting S_6 and S_7 under different life-cycle conditions. These figures depict histograms conditioned on the charge state, a measurement that requires knowledge of the stored data. However, the realistic input of the FNN is unconditioned histograms. This corresponds to the sum of the conditioned histograms for each threshold voltage, effectively discarding information about the original data. Thus, this approach remains completely independent of any knowledge regarding the stored data.

The objective is to train a shallow FNN with as few measurements from the data set as possible, including emulated data. This shall show that it is possible to emulate arbitrary life-cycles with a small set of measurements. Experiments with different parametrization reveal that a FNN comprising one hidden layer with three neurons can solve this task effectively.

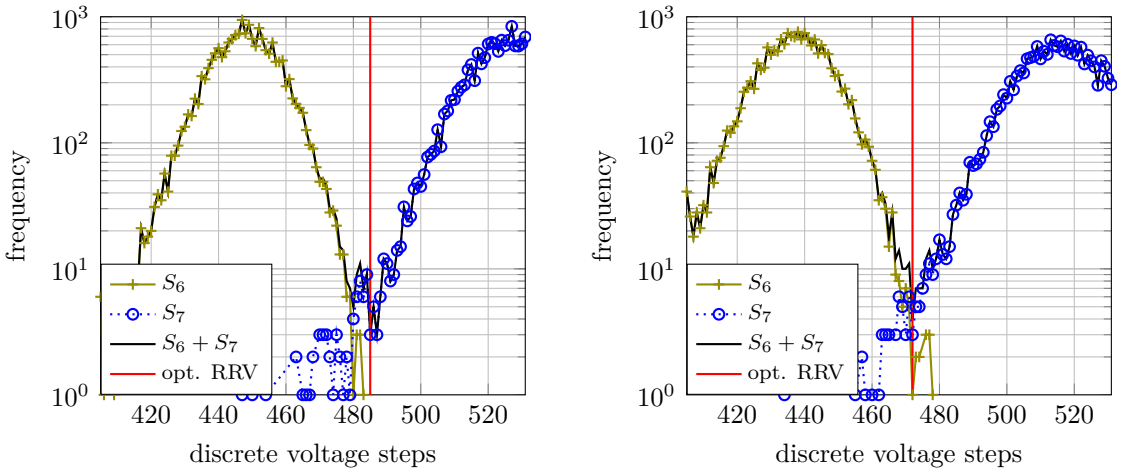


Figure 5.16: TVDs for S_6, S_7, and the sum of both distributions (mixture distribution) for a single page. **(a)** 800 P/E, 0 h DRT, no RD, 25 °C PTMP, - 35 °C RTMP. **(b)** 3000 P/E, 5.4×10^3 h DRT, no RD, 25 °C PTMP, 25 °C RTMP (adapted from [7], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/).

Table 5.2: Average optimal discrete RRV for different life-cycle conditions using averaged TVD (adapted from [6], ©2022 IEEE)

	life-cycle condition				optimal reference voltage step					
test set	P/E	DRT	PTMP	RTMP	V_{r1}	V_{r2}	V_{r3}	V_{r4}	V_{r5}	V_{r6}
1	1000	0	-35 °C	85 °C	149	212	276	342	405	478
2	2200	0	-35 °C	25 °C	149	211	272	337	399	472
3	2600	0	85 °C	-35 °C	141	205	267	332	394	463
4	3000	0	25 °C	25 °C	144	209	270	337	399	468
5	3000	$2.8{\cdot}10^3$ h	25 °C	25 °C	145	205	266	330	392	463
6	3000	$5.8{\cdot}10^3$ h	25 °C	25 °C	138	200	262	326	388	460
	maximal difference				11	12	14	16	17	18

The size of the NN indicates that the complexity of the problem is manageable. For both training and testing purposes, we will employ averaged high-resolution histograms as the input for the FNN. The total TVD of measurements is employed, where ten consecutive pages are averaged to reduce the noise effects and evaluate the optimal reference voltage. The target of the FNN is the optimal RRV. The considerable amount of input data contributes significantly to achieving exceptional, accurate results. Note that acquiring fine-grained TVDs requires applying every possible threshold voltage and a subsequent read operation. While adopting this approach results in a highly precise TVD, which is advantageous for subsequent signal processing, it concurrently leads to a significant increase in latency, adversely influencing the overall processing speed.

Employing averaged TVDs results in different optimal reference voltage steps compared to

those illustrated in Table 5.1. Consequently, Table 5.2 shows the average optimal reference voltage steps for six different test sets using averaged TVD. This averaging process also introduces a level of uncertainty into the learning process. This uncertainty enhances the problem statement and contributes to the model generalization, i.e., increases the robustness.

In the subsequent, we present numerical results that compare the performance of FNN with and without the application of DA. The investigation focuses on assessing the generalization capabilities of the FNN. Specifically, we train these networks using a limited measurement set of life-cycle conditions. The networks are evaluated using completely independent measured training data. The test sets with the same life-cycle conditions as the training data were measured on different flash memory blocks. Each block has unique physical properties due to the electrical properties on the chip and production process variations. The output of the NN provides an estimation of the optimal RRV, which is used to compute the residual error probability corresponding to each measurement.

We established a comparison involving different scenarios (sc.):

1. The initial setup employs two training data sets without any DA. These two sets were selected based on the extreme points concerning the observed mean values. Within the tested life-cycle conditions, it is observed that threshold voltages tend to increase with cross-temperature effects and decrease with high DRT. For this reason, we use a measurement set with the life-cycle condition as test set 1 and 6 from Table 5.2 as these represent the extremes. Note that variations in PTMP and RTMP indicate a cross-temperature effect. (sc. 1)

2. The second test scenario uses only the upper extreme training set from sc. 1 but includes DA. This procedure involves a stepwise shift of the threshold voltages of the measured upper extreme point (equivalent to test set 1) down to the minimum mean value. Consequently, this generates eighteen shifted data sets. (sc. 2)

3. The third test scenario reverses the approach of sc. 2, i.e., we perform a stepwise shift of the minimum measured extreme point (equivalent to test set 6) up to the maximum mean value. (sc. 3)

4. The training set for the final scenario combines the generated data from scenarios 2 and 3. This combination implies that we use additionally 36 training sets generated by DA.

These scenarios were developed to provide a comprehensive evaluation of the performance of the FNNs, both with and without the use of DA. In the subsequent section, we present numerical results that compare the performance of FNN with and without the application of DA.

The left part of Figure 5.17 shows the resulting error probability for different pages on V_{r6} without DA. The achievable error rate is shown for comparison, which is obtained by averaging the conditional histograms over ten consecutive pages. In particular, estimation without DA leads to increased error rates, with some cases more than doubling the error probability. The

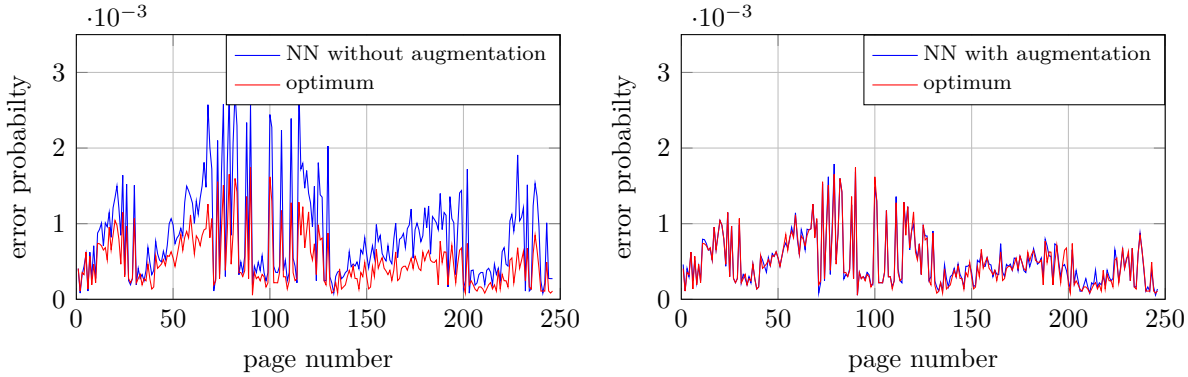


Figure 5.17: Training method 1 (left part) and training method 4 (right part) on test set 4 on V_{r6} (adapted from [6], ©2022 IEEE).

right part of Figure 5.17 depicts the error rate when DA is employed, demonstrating near-optimal performance.

Table 5.3: Comparison of the results of the NN for V_{r1} and V_{r6} (adapted from [6], ©2022 IEEE)

		average error probability				
test set	V_r	target	Set 1 and 2 (sc. 1)	Set 1 shifted (sc. 2)	Set 2 shifted (sc. 3)	Set 1 and 2 shifted (sc. 4)
1	1	$9.11{\cdot}10^{-4}$	$9.21{\cdot}10^{-4}$	$9.14{\cdot}10^{-4}$	$1.01{\cdot}10^{-3}$	$9.21{\cdot}10^{-4}$
	6	$6.83{\cdot}10^{-4}$	$6.92{\cdot}10^{-4}$	$6.88{\cdot}10^{-4}$	$8.71{\cdot}10^{-4}$	$7.02{\cdot}10^{-4}$
2	1	$1.3{\cdot}10^{-3}$	$1.33{\cdot}10^{-3}$	$1.31{\cdot}10^{-3}$	$1.40{\cdot}10^{-3}$	$1.32{\cdot}10^{-3}$
	6	$9.06{\cdot}10^{-4}$	$9.84{\cdot}10^{-4}$	$9.72{\cdot}10^{-4}$	$9.55{\cdot}10^{-4}$	$9.22{\cdot}10^{-4}$
3	1	$9.25{\cdot}10^{-4}$	$1.01{\cdot}10^{-3}$	$9.98{\cdot}10^{-4}$	$9.38{\cdot}10^{-4}$	$9.27{\cdot}10^{-4}$
	6	$1.44{\cdot}10^{-3}$	$1.75{\cdot}10^{-3}$	$1.73{\cdot}10^{-3}$	$1.46{\cdot}10^{-3}$	$1.46{\cdot}10^{-3}$
4	1	$1.51{\cdot}10^{-4}$	$1.67{\cdot}10^{-3}$	$1.62{\cdot}10^{-4}$	$1.79{\cdot}10^{-4}$	$1.85{\cdot}10^{-4}$
	6	$4.97{\cdot}10^{-4}$	$8.37{\cdot}10^{-4}$	$6.19{\cdot}10^{-4}$	$5.16{\cdot}10^{-4}$	$5.26{\cdot}10^{-4}$
5	1	$9.12{\cdot}10^{-4}$	$9.66{\cdot}10^{-4}$	$9.27{\cdot}10^{-4}$	$9.48{\cdot}10^{-4}$	$9.26{\cdot}10^{-4}$
	6	$1.21{\cdot}10^{-3}$	$1.33{\cdot}10^{-3}$	$1.34{\cdot}10^{-3}$	$1.22{\cdot}10^{-3}$	$1.22{\cdot}10^{-3}$
6	1	$5.14{\cdot}10^{-4}$	$5.22{\cdot}10^{-4}$	$6.04{\cdot}10^{-4}$	$5.14{\cdot}10^{-4}$	$5.14{\cdot}10^{-4}$
	6	$1.17{\cdot}10^{-3}$	$1.22{\cdot}10^{-3}$	$1.29{\cdot}10^{-3}$	$1.18{\cdot}10^{-3}$	$1.18{\cdot}10^{-3}$
relative mean deviation			$1.21{\cdot}10^{-1}$	$9.07{\cdot}10^{-2}$	$6.88{\cdot}10^{-2}$	$3.41{\cdot}10^{-2}$

Table 5.3 compares the results with and without DA across different life-cycle conditions and distinct test scenarios. The comparison focuses on the RRV for the MSB page according to the bit-labeling from Gray code 1 within Table 4.1, i.e., V_{r1} and V_{r6}.

While the results attained through the NN approach without DA are already very close to the optimal values, incorporating DA yields further enhancements. The combined test set with DA gives the best average result for all test scenarios. To quantitatively compare the scenarios, the average relative deviation concerning the target value is calculated and shown in the final row of Table 5.3. This indicates that the additional data improves the statistics for the training

algorithm.

In conclusion, this approach effectively addresses the challenge of model generalization in training an NN to adjust the reference voltage. The proposed estimation procedure leverages a shallow NN and histogram measurements to accurately estimate RRVs for different life-cycle conditions. It is demonstrated that a limited number of measurements can cover the essential characteristics of the threshold voltage distributions of flash-based memory.

Additionally, DA using shifted histograms can enhance the training data for the NNs. The NNs trained on measurements of extreme points exhibit strong generalization capabilities for other life-cycle conditions. However, this NN experiment utilizes training sets with full measurement resolution, which is impractical due to excessive complexity. Therefore, quantizing the measurements and using the quantized information as a feature is more convenient and realistic.

In the subsequent section, we reduce the data resolution to develop a more realistic and practical adaptation approach.

5.4.4 Analysis with Sparse Histograms

This section shows that an adaptation employing FNNs relying on sparse histograms of HD and SD readouts leads to nearly optimal performance. However, we do not focus on developing an optimized low-complexity FNN approach for this specific problem. We continue to use fully-connected FNNs as described in Section 5.4.1 for the adaptation process.

Similar to [3], we assume that the data is initially read at a default RRV V_{ri}, i.e., these default RRVs are used to generate the histogram data for the input features. The entire process of histogram generation is described in Section 5.2.3. The default RRVs are also used for one of the evaluations as static references without adaptation.

To determine the default RRVs, we average the optimal RRVs obtained from a measurement with 800 P/E, $0\ h$ DRT, no RD, $25\,^{\circ}$C PTMP and -$35\,^{\circ}$C RTMP. This specific measurement exhibits a minimal residual error rate and closely approximates a fresh condition. Moreover, the selection of the SD intervals to generate one and two SBs at each V_{ri} is based on maximizing the MI at the EOL state. These default values serve as the basis for training and validating the NNs. We compute additional training data sets generating sparse histograms at different RRVs to generalize the NN approach. This generates some uncertainty in the training process and avoids using fixed predefined RRVs.

FNN with different configurations were tested, revealing that a few hidden layers with a small number of neurons are adequate for solving the estimation problem. For the presented numerical results, we consider two cases:

1. NN with one hidden layer and five neurons (NN1)
2. NN with two hidden layers with five neurons each (NN2).

Table 5.4: Size of the NN (number of weights) in the amount of floating point numbers ([7], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/).

	NN1			NN2		
page	HD	1 SB	2 SB	HD	1 SB	2 SB
MSB/LSB	42	62	72	68	92	112
CSB	58	88	103	88	118	148

The storage requirements for the FNNs vary depending on the utilized readout method and the target page. Table 5.4 outlines the storage needed for each readout method corresponding to the presented FNN configurations.

To mitigate overfitting and minimize the requirements for the measurements, we employ a minimal amount of training data to demonstrate that a few specific measurements are sufficient as a training basis. We explore three different training sets (TS) based on Table 5.1 involving different measurements:

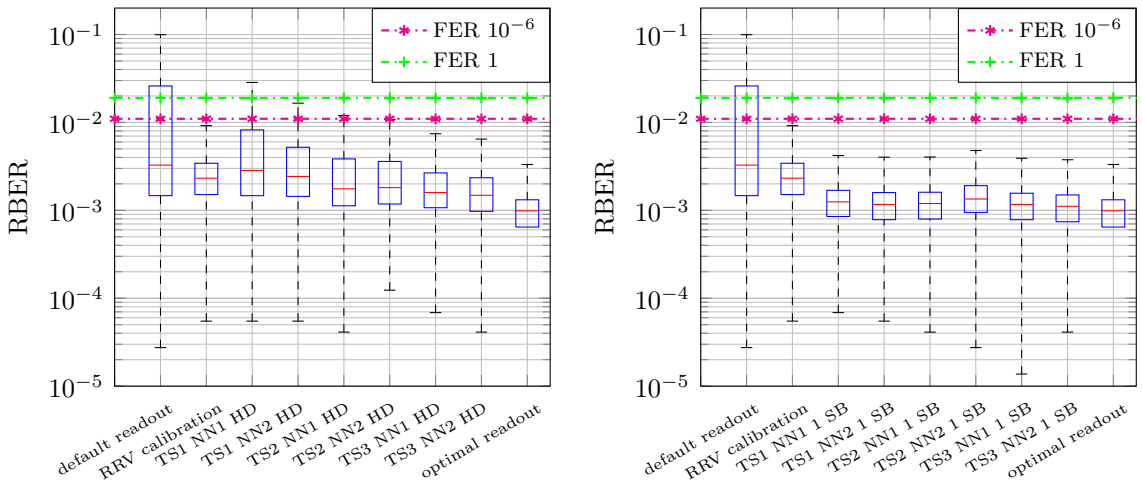
1. training set 1 (TS1): two data sets, two cases of extreme measurements, comprising test set 8 and 12.
2. training set 2 (TS2): four data sets, two cases of extreme measurements, comprising test set 8 and 5; two data sets with condition 12.
3. training set 3 (TS3): six data sets comprising TS2 with two additional data sets (test set 10 and 11).

Moreover, we incorporate eight different RRV sets to generate multiple histograms of each training set. These RRV sets are uniformly distributed across the range between the minimum and maximum mean RRVs of the data sets. This approach generates elements of uncertainty during the training process and eliminates the need for predefined fixed RRVs. Experiments have shown that using this methodology additional DA does not significantly improve the test data. As a result, we ignore DA in the subsequent analysis.

Overall, we used for the training process 256 pages per data set. The test procedure includes independent data sets from the training data, but similar life-cycle conditions are included in the test set.

Next, the results of the proposed methods are presented. A comprehensive comparison between these results and the RRV calibration discussed in Section 5.3.1 is provided. The outcomes of five NNs were averaged for the numerical results to avoid anomalies.

We use boxplot figures to compare the statistical performance of the different training methods. The boxplot is constructed such that the top and bottom of each box correspond to 75% and 25% of the samples, respectively. The red line in the box indicates the median. The top and bottom whisker lengths are adjusted to be three times the length of the interquartile range. Values exceeding this range are considered extreme outliers and are not considered in the boxplot figures. One measurement set comprises 256 pages, i.e., the total number of pages



(a) **(b)**

Figure 5.18: NN based on default readout against RRV calibration: BERs for the MSB page with different readout methods and training strategies compared to RRV calibration strategy based on metadata from Section 5.2.2. **(a)** Quantization based on HD readout. **(b)** Quantization with one SB SD readout (adapted from [7], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/).

for verification is $151 \cdot 256 = 38\,656$. Each page has a payload of 4kB with a code rate $R = 0.9$, resulting in $36\,490$ bits per page. Note that no averaging of the TVD is performed.

We concentrate on applying the NN on the MSB page according to the Gray code 1 bit-labeling on Table 4.1 as an example, specifically predicting V_{r2} and V_{r6}. According to this bit-labeling, we assign charge states S_3 - S_6 to logical 0, and S_0, S_1, S_2, S_7 to logical 1.

In our analysis, the boxplots illustrate the RBERs achieved by employing different approaches for determining the RRV. Additionally, we include two residual error probability curves for an LDPC code with a code rate of $R = 0.91$ and payload length of 1kB using HD decoding from [87]. This code has been standardized for flash memories with a high storage rate, and the shortness of the code enables fast processing. At a RBER of 0.011, this code reaches a FER of about 10^{-6}, whereas for a RBER of 0.019, the decoder fails with a probability close to one. This narrow RBER range highlights the sensitivity of error-correctability.

Using the default RRV, approximately 37% of the measured pages cannot be successfully decoded. Conversely, by employing the optimal RRVs, 98% of the pages achieve a FER below 10^{-6}. However, even with perfect RRV adaptation, some extreme outliers (0.2% of all codewords) are not decodable with hard-input decoding.

We now delve into the results, beginning with the case involving the default RRV. We compare all the results with the RRV calibration algorithm based on metadata from Section 5.2.2, denoted as *RRV calibration*. Figure 5.18b illustrates numerical results using histograms with a single SB for the MSB page. This method results in seven histogram values, i.e., the input features comprise these seven values plus two values for the HD RRVs, resulting in a total of

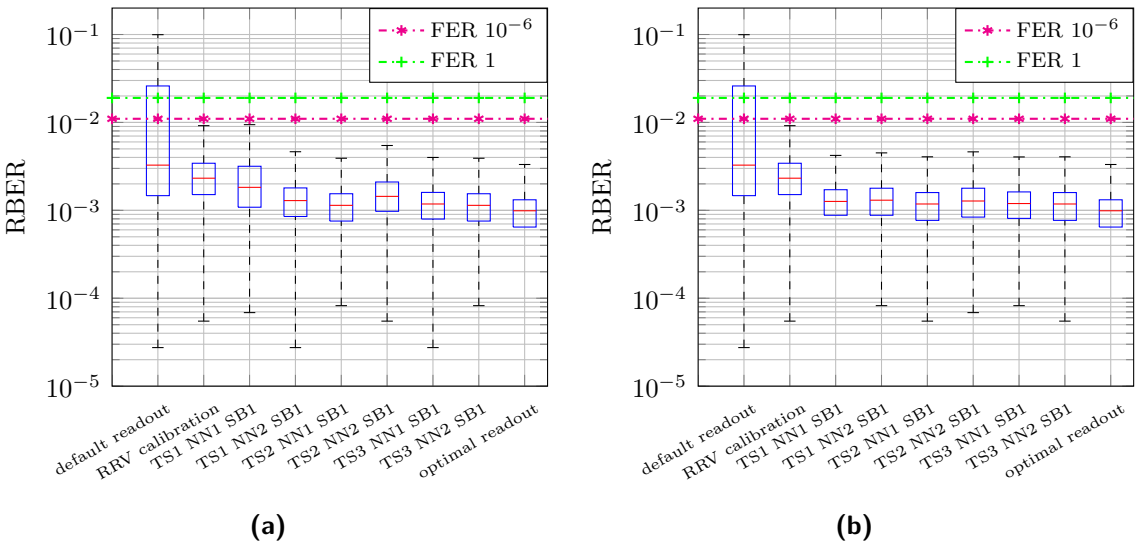


Figure 5.19: Two-step process against RRV calibration: RBERs for the MSB page on NN with different training strategies using one SB SD readout after using RRV calibration compared to RRV calibration strategy based on metadata Section 5.2.2. **(a)** Training without DA. **(b)** Training with DA (adapted from [7], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/).

nine input features for the FNN. For the small training set (TS1), NN2 outperforms the NN1, and with the more extensive data set (TS3), both NNs achieve close to optimal performance. The predicted RRVs from the NNs yield consistently lower error rates than the static default RRVs. On average, the static RRVs produces three times more errors.

The NNs using two SBs readouts demonstrate similar outcomes as using only one SB. Notably, the additional histogram values do not significantly enhance the resulting mean error rate. Consequently, we omit detailed numerical results.

Subsequently, we consider numerical results using a HD readout for the histograms as input to the NN. This means that we generate only five input values for the NN, specifically three values as histogram data and two values for the RRVs for the MSB page. As depicted in Figure 5.18a, estimations based on HD readouts consistently result in some performance degradation, which is particularly evident with smaller training data sets. Depending on the chosen neural network and training method, the HD approach achieves comparable results to the RRV calibration from Section 5.2.2.

Besides this analysis, we consider a two-step process: first, performing the RRV calibration presented in Section 5.2.2, followed by the presented FNN approach to enhance the adaptation. This strategy is particularly relevant when using HD readout as the primary option, and a calibration process is executed before resorting to SD readout. This approach aims to maintain a lower complexity. SD readout inherently involves higher complexity and should therefore only be used in situations where it is unavoidable. The histograms are generated with the RRV from the calibration result, i.e., the RRV suggested by the calibration is used for each page.

For this analysis, we use the same NN and data sets as before. Figure 5.19a shows the results using one SB. The two SB readout approaches show almost the same results in terms of error rate. Thus, detailed numerical results for this scenario are omitted.

To generate more training data, we employ DA. In our approach, shifting the measured histogram sets enables emulating other life-cycle conditions. DA improves the generalization capability of the NNs. However, we limit the application of DA to doubling the data in the training sets. Figure 5.19b displays the impact of DA on enhancing the training sets. In this case, DA significantly improves performance only for TS1, where limited training data is available.

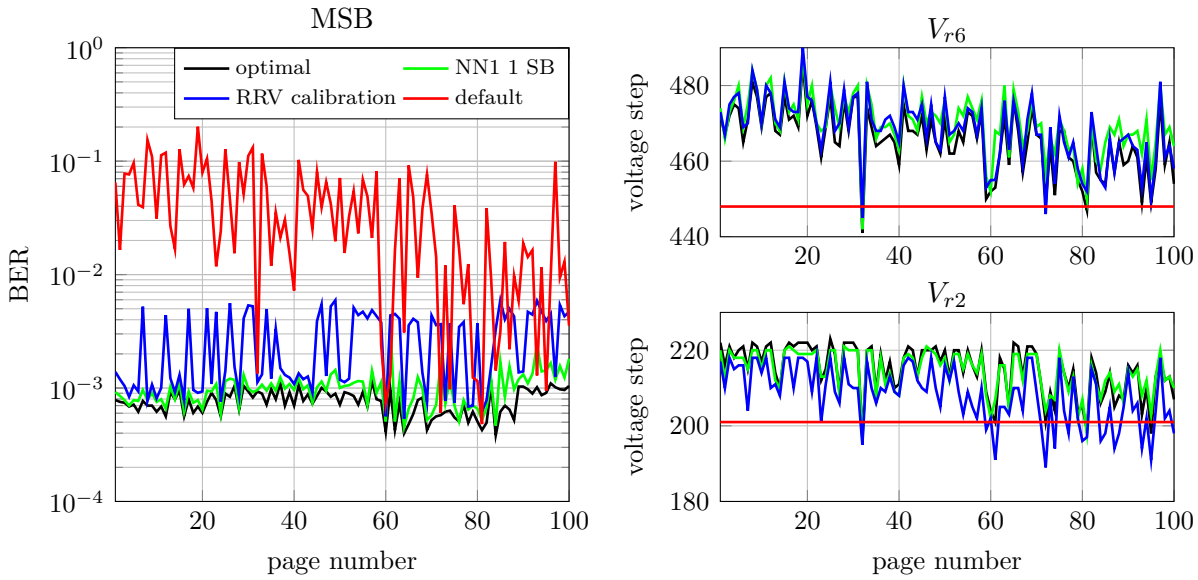


Figure 5.20: Results with one SB SD readout on RRV of the calibration method (two-step process) for using NN1 using training set 2 (TS2) against RRV calibration on pages with 3000 P/E, $1.4{\cdot}10^3$ h DRT, no RD, 25 °C PTMP and 25 °C RTMP (adapted from [7], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/).

The boxplot graphs provide insights into the error rates using different NNs setups. Subsequently, we delve into the performance of an exemplary life-cycle condition measurement, considering the variations across pages. One measurement comprises 256 consecutive pages, although due to their noisy nature, we present only partial sections of a measurement.

Figure 5.20 shows an example of the two-step process. The default readout strategy leads to a high error rate, which is a substantial issue in various scenarios. Consequently, adjusting the RRV would become necessary in such a case. Results for NN1 with TS2 are discussed. This example clearly demonstrates that the two-step process significantly enhances performance, enabling nearly optimal results. Note that the two NN approaches reach almost the same performance in this case. Hence, numerical results for NN2 are unnecessary.

5.5 Summary

This chapter shows that RRV adaptation of non-volatile NAND flash memories is essential to achieve low residual error probability, which ensures storage reliability. The optimal RRVs depends on many aging effects, such as the number of P/E cycles, DRT, RD, and others. Using fixed default RRVs leads to high RBER and eventual decoding failures. Especially the HD decoding requires a low RBER, essential for maintaining high access speeds.

We proposed two methods for low-complexity RRV adaptation. First, a calibration method based on the observable errors in the metadata enables an initial prediction of the voltage shift for threshold calibration. However, a single adaptation step does not guarantee sufficient accuracy. Adaptation errors may lead to decoding failures. Moreover, the RRVs vary from page to page, even in the same flash block. The second adaptation method involves gradual RRV tracking to address threshold calibration errors and voltage shifts across consecutive pages within the same block. This tracking algorithm approach exploits the asymmetric error distribution. Our numerical results based on flash measurements underscore that both methods significantly reduce the RBER of NAND flash memories. These methods rely on metadata or pilot data for estimation.

Therefore, an RRV prediction method was proposed based on shallow FNNs, which uses histograms as features. Many approaches for adapting the RRVs of the flash cells are data-driven. To effectively train and verify such calibration methods and channel models, it is necessary to measure flash cells for many different life-cycle conditions. However, performing a high number of such measurements is costly and time-consuming. Hence, the question arises concerning which life-cycle conditions measurements are relevant for RRV adaptation training.

We first addressed the issue of model generalization for training an NN to adapt RRVs. It was shown that few measurements can cover the essential aspects of the TVDs. DA can enhance the training data for the neural networks by using shifted histograms. Neural networks trained on measurements from extreme points generalize well to other life-cycle conditions. However, these training sets were employed with full measurement resolution.

In response, we proposed an FNN approach relying on sparse histogram features. This approach enables efficient voltage adaptation in the case of a decoding failure or voltage changes, eliminating the need for additional read operations. This approach achieves a nearly optimal readout performance with a modest data set. We also showed that DA improves the results when minimal data is used in the training sets. Additionally, an NN with an HD readout achieves similar results to conventional adaptation algorithms employing pilot or metadata. Compared to the RRV calibration method, NN adaptation with HD readout achieves similar performance, while SD readout yields even superior results. We demonstrated that generating histograms for NN input features with HD and SD readout does not require fixed predefined RRVs, enabling sequential application with other calibration methods.

The pivotal question concerning when an RRV adaptation becomes necessary. If HD decoding fails, the SD decoding is triggered afterwards. At this point, the NN estimates the optimal

RRV, enabling an improved RRV for subsequent readout attempts. This method can lead to reduced residual error rates and successful HD decoding in subsequent attempts.

However, a data-driven estimation method can only consider some possible life-cycle conditions. Future research should consider other methods of DA, such as training data based on more accurate probability density functions for various life-cycle conditions.

Important information about the life-cycle state of the flash device is stored in the metadata. This chapter describes that specific features are extracted from the metadata and processed. This processing may involve tasks such as adjusting the RRVs or determining the current life-cycle of the device. To successfully extract these features, data must be read correctly from the metadata. Therefore, ECCs are used to ensure the integrity of the data.

In the following chapters, we will focus on investigating specific ECCs for metadata. The challenge is constructing reliable codes with low complexity and short code length. Hence, the forthcoming chapter delves into some fundamentals of ECCs that are necessary for the subsequent chapters. In the following chapters, we will focus on investigating specific ECCs for metadata. The challenge is constructing reliable codes with low complexity and short code length. Hence, the forthcoming chapter delves into some fundamentals of ECCs that are necessary for the subsequent chapters.

Chapter 6

Fundamentals of Error Correction Procedures

The reliability of NAND flash memory is inherently limited, and data integrity can only be maintained using ECC [13, 14, 20]. Flash memory for enterprise solutions requires low residual error rates over the entire life-cycle. Various flash memory manufacturers are committed to ensuring very low residual error rates. Meeting the diverse requirements of error correction in NAND flash memory entails employing various error correction methods. Payload data, which demands high data rates due to performance purposes, often relies on long codes to reach a low FER [14, 19, 88]. On the other hand, the metadata area uses short codes with moderate or low code rates [20].

Reading the information stored involves applying reference voltages to the flash cells, in a process that can introduce errors, as we have seen in the previous chapters. The error characteristics depend on many factors, as explained in Chapter 3. To achieve the required access speeds, it is crucial to use low latency ECCs. Consequently, encoding and decoding are often implemented in hardware [89, 90, 13]. In addition, the ECC must have a minimum complexity to minimize the required hardware area.

This chapter provides background information on error correction procedures for the subsequent chapters. First, we start with some preliminaries continued by introducing the well-known Reed-Muller (RM) code and explore their generation using the Plotkin construction, a concept that we will leverage in a forthcoming chapter. Subsequently, we provide a brief introduction to Bose-Chaudhuri-Hocquenghem (BCH) codes, as they are part of the investigations in the subsequent chapters. Notably, we will exploit specific properties of BCH codes in the following chapters to reduce complexity and generate other codes. Additionally, binary BCH codes will be utilized in conjunction with Reed-Solomon (RS) codes in the upcoming chapters. Finally, we offer a very brief introduction to Low-Density Parity-Check (LDPC) codes, a coding scheme extensively employed in flash memory applications. This code family will be used for performance comparison in the subsequent chapters. Parts of this chapter have been published in [4, 9].

6.1 Preliminaries

In this work, we consider linear ECCs as $\mathcal{C}(n, k, d)$, where

- n represents the code length,
- k is the code dimension,
- d denotes the minimum distance.

In this context, each code has $n - k$ parity symbols. The code rate is defined as the ratio between dimension and length expressed as $R = k/n$ and must satisfy the condition $R \leq 1$.

A codeword $\mathbf{c} \in \mathcal{C}$ is represented as $\mathbf{c} = [c_0, c_1, c_2, \ldots, c_{n-1}]$. This code $\mathcal{C} \subset GF\,(2^m)^n$ is defined as a linear block code when for every pair of codewords $\mathbf{c}_1$ and $\mathbf{c}_2 \in \mathcal{C}$, the linear combination $a\mathbf{c}_1 + b\mathbf{c}_2$ with $a, b \in GF(2^m)$ remains within the code $\mathcal{C}$ [91]. Unless otherwise stated, in the following we consider binary codes with the symbols $c_i \in GF(2)$.

The minimum Hamming distance is defined by

$$d = \min_{\substack{\mathbf{c}_1, \mathbf{c}_2 \in \mathcal{C} \\ \mathbf{c}_1 \neq \mathbf{c}_2}} \{\mathrm{dist}(\mathbf{c}_1, \mathbf{c}_2)\}, \tag{6.1}$$

where the minimum distance for linear codes is equal to the minimum Hamming weight [92]

$$d = \min_{\substack{\mathbf{c}_1, \mathbf{c}_2 \in \mathcal{C} \\ \mathbf{c}_1 \neq \mathbf{c}_2}} \{\mathrm{wt}\,(\mathbf{c}_1 - \mathbf{c}_2)\} = \min_{\substack{\mathbf{c}_1, \in \mathcal{C} \\ \mathbf{c}_1 \neq 0}} \{\mathrm{wt}(\mathbf{c}_1)\}. \tag{6.2}$$

An essential property of ECCs is their ability to detect errors or verify correctness. To illustrate this aspect, we consider the single parity-check (SPC) code as an example, which has the capability to detect a single error. In this code, the code length is calculated as $n = k + 1$, indicating the addition of just one extra bit. Consequently, the code rate is given by $R = k/n = n - 1/n$, which implies that larger code lengths result in higher code rates. The minimum Hamming distance is constantly $d = 2$.

The generation rule for the additional bit is based on the summation of all k binary digits modulo 2. If this results in an even number, the parity bit is a binary 0. Otherwise, it is assigned a binary 1.

Example 6.1. *Consider the binary sequence* $\mathbf{i} = [10110]$. *As the summation of the binary digits modulo 2 results in* 1, *the parity bit is also set to* 1. *The codeword becomes* $\mathbf{c} = [101101]$.

In the event that a bit experiences a flip either during transmission over a communication channel or while reading the codeword from memory, the error can be detected. However, correction is not possible in this scenario.

The second important role of ECCs is the ability to correct errors within a codeword. Therefore, we consider the well-known repetition code. As the name suggests, this code involves repeating the binary information until the desired codeword length n is achieved.

Consequently, the minimum distance of the code is $d = n$, and the code rate is $R = 1/n$. After transmission, the codeword is decoded with the majority decision approach. Therefore, it is important to have an odd code length to ensure a unique solution during the decoding process.

Example 6.2. *We consider an exemplary repetition code with a code length of $n = 3$. In this case, the minimum distance becomes $d = 3$, and the code rate is $R = 1/3$. As an exemplary bit sequence, we take $[101]$. The encryption process results in the code sequence comprising three codewords $[111\ 000\ 111]$.*

Now, we suppose one transmission error per single codeword. For instance, the corrupted code sequence could be $[101\ 010\ 101]$. When employing maximum-likelihood (ML) decoding, the three errors would be detected because for the first and third codewords, two out of three bits are 1, i.e., the majority is 1, and for the second codeword, two out of three code parts are 0, i.e., the majority is 0. As a result, the decoded bit sequence is $[101]$.

However, if two errors per codeword occurred, the erroneous code sequence might result in $[100\ 011\ 001]$. After ML decoding, the result would be $[010]$. Consequently, the decoding process is unable to correct the errors, and the original message cannot be recovered accurately.

These examples highlight the inherent limit of error detection and correction capability. This is determined by the minimum Hamming distance of the code.

In general, it is possible to detect $d - 1$ errors, and the error correction capability is expressed as

$$t = \left\lfloor \frac{d-1}{2} \right\rfloor$$

Thus, a linear block code can safely correct up to t errors. SPC codes cannot correct any error and can detect only one. The error correction capability in the example with the repetition code of length $n = 3$ is $t = 1$. Accordingly, it can only correct one error per codeword. In this context, we declare one frame error if there is at least one symbol error per codeword.

The relationship between code length and error correction capability is a fundamental concept described by Shannon's channel coding theorem [57]. This theorem asserts that there always exists a block code with rate R and length n such that the frame error probability P_F satisfies:

$$\lim_{n\to\infty} P_F^n \to 0.$$

In this context, it is essential to consider the channel capacity C, introduced in Section 4.2. Thus, as long as the condition $R < C$ is satisfied for sufficiently long codes, there will always be an ECC capable of correcting all errors. Conversely, if $R \geq C$, the error probability for all codes become $\lim_{n\to\infty} P_F^n = 1$ [92, 88].

Indeed, the length of a code is related to the error correction strength. However, the decoding complexity also increases with the length of an ECC. In the context of the cyclic block codes like RS codes, e.g., the complexity of polynomial arithmetic increases with the size of the finite field. As the code length evolves, both the polynomial degree and the field size expand, leading to corresponding computational complexity during decoding. This trade-off between error correction strength and decoding complexity is a critical consideration in code design and implementation.

Linear codes can be generated using a generator matrix denoted as $\mathbf{G}$ and verified with a parity-check matrix $\mathbf{H}$. These matrices are orthogonal to each other, which means

$$\mathbf{G} \cdot \mathbf{H}^\top = \mathbf{H} \cdot \mathbf{G}^\top = 0.$$

The dimensions of these matrices are $k \times n$ for $\mathbf{G}$ and $(n-k) \times n$ for $\mathbf{H}$. When encoding the vector $\mathbf{i}$, the codeword $\mathbf{c} = \mathbf{i} \cdot \mathbf{G}$ results. A valid codeword $\mathbf{c} \in \mathcal{C}$ satisfies [92]

$$\mathbf{H} \cdot \mathbf{c}^\top = \mathbf{c} \cdot \mathbf{H}^\top = 0.$$

The generator and parity-check matrices can be represented in systematic or unsystematic forms, resulting in corresponding codes. In the systematic form, the information part of the codeword is readable without decoding. This is achieved by defining $\mathbf{G} = [\mathbf{I}_G|\mathbf{A}]$, where $\mathbf{I}_G$ represents an identity matrix of dimension $k \times k$. $\mathbf{A}$ denotes the parity generating part with dimensions $k \times (n-k)$. In this representation, the message bits occupy the k first positions. The parity-check matrix in systematic form is defined as $\mathbf{H} = [-\mathbf{A}^\top|\mathbf{I}_H]$, where $\mathbf{I}_H$ has the dimension $(n-k)\times(n-k)$. By contrast, unsystematic coding mixes the parity and information positions, which do not allow a direct readability of the information bits. The systematic form is a desirable property for a linear block code, and as such we employ this structure in the context of this work [91].

In the following, we will discuss a binary code construction that leverages the simplicity of the SPC and the repetition code. This code generation leads to codes with different rates and error correction capabilities.

6.2 Reed-Muller Codes

Reed-Muller (RM) codes are a well-known class of binary linear codes, first introduced in 1954 by Irving S. Reed and David E. Muller [93, 94]. RM codes offer efficient encoding and decoding methods that require minimal computational resources. Notably, they have recently regained attention due to their relation with polar codes [95, 96, 97, 98]. In fact, it has been demonstrated in [99] that RM codes can achieve capacity on erasure channels.

Binary RM codes are defined by two parameters, r and m, with the constraint $0 \leq r \leq$

m. These parameters are typically represented as $\mathcal{R}(r, m)$. The parameter for the code representation $\mathcal{C}_{\mathcal{R}}(n, k, d)$ are determined using the following relationship such that [92]:

$$n = 2^m \tag{6.3}$$

$$k = \sum_{i=0}^{r} \binom{m}{i} \tag{6.4}$$

$$d = 2^{m-r}. \tag{6.5}$$

The parameter r is often referred to as the order of the code and plays a crucial role in defining its characteristics.

RM codes can be recursively defined using the Plotkin construction, also called $|u \mid u+v|$ construction, introduced in [100]. Essentially, this method involves concatenating two binary component codes of length n to create a code of length $2n$. This process is repeated until the desired final code length and rate are reached.

Two component codes $\mathcal{C}^{(1)}(n, k_1, d_1)$ and $\mathcal{C}^{(2)}(n, k_2, d_2)$ are the basis of the resulting concatenated code $\mathcal{C}(2n, k_1 + k_2, \min(2 \cdot d_1, d_2))$. The codewords $\mathbf{x}^{(1)} = [x_0^{(1)}, \ldots, x_{n-1}^{(1)}] \in \mathcal{C}^{(1)}$ and $\mathbf{x}^{(2)} = [x_0^{(2)}, \ldots, x_{n-1}^{(2)}] \in \mathcal{C}^{(2)}$ of these component codes are combined as follows

$$\begin{aligned} \mathbf{x} &= [\mathbf{x}^{(1)} \mid \mathbf{x}^{(1)} \oplus \mathbf{x}^{(2)}] \\ &= [x_0^{(1)}, \ldots, x_{n-1}^{(1)}, x_0^{(1)} \oplus x_0^{(2)}, \ldots, x_{n-1}^{(1)} \oplus x_{n-1}^{(2)}], \end{aligned} \tag{6.6}$$

where the operator $\oplus$ denotes the modulo-2 sum.

For encoding, generator matrices are commonly employed in various applications. The generator matrix for this Plotkin construction here comprises the component generator matrices. Let $\mathbf{G}_1$ and $\mathbf{G}_2$ be the generator matrices of code $\mathcal{C}^1$ and $\mathcal{C}^2$, respectively. The concatenated generator matrix is then formulated as

$$\mathbf{G} = \begin{bmatrix} \mathbf{G}_1 & \mathbf{G}_2 \\ 0 & \mathbf{G}_2 \end{bmatrix}. \tag{6.7}$$

Note that component codes $\mathcal{C}^{(1)}$ and $\mathcal{C}^{(2)}$ are selected to maximize the minimum Hamming distance. In the case of multiple concatenation, each component may comprise further subcomponents, creating a hierarchical structure.

Using the Plotkin construction to generate an RM code denoted as $\mathcal{R}(r, m)$, the recursion process begins with SPC codes and repetition codes at the first level, which corresponds to either $r = 0$ or $r = m-1$. At the first level, an SPC code has the parameters $\mathcal{C}(2^m, 2^m - 1, 2)$, while a repetition code has the parameters $\mathcal{C}(2^m, 1, 2^m)$.

Figure 6.1 shows the code tree of RM codes using the Plotkin construction, showing possible code parameters up to $m = 5$. The RM code with parameters (r, m) is constructed by iteratively combining the RM codes $\mathcal{C}^{(1)}$ with parameters $(r, m-1)$ and $\mathcal{C}^{(2)}$ with parameters $(r-1, m-1)$. This recursive approach generates RM codes with the desired parameters while

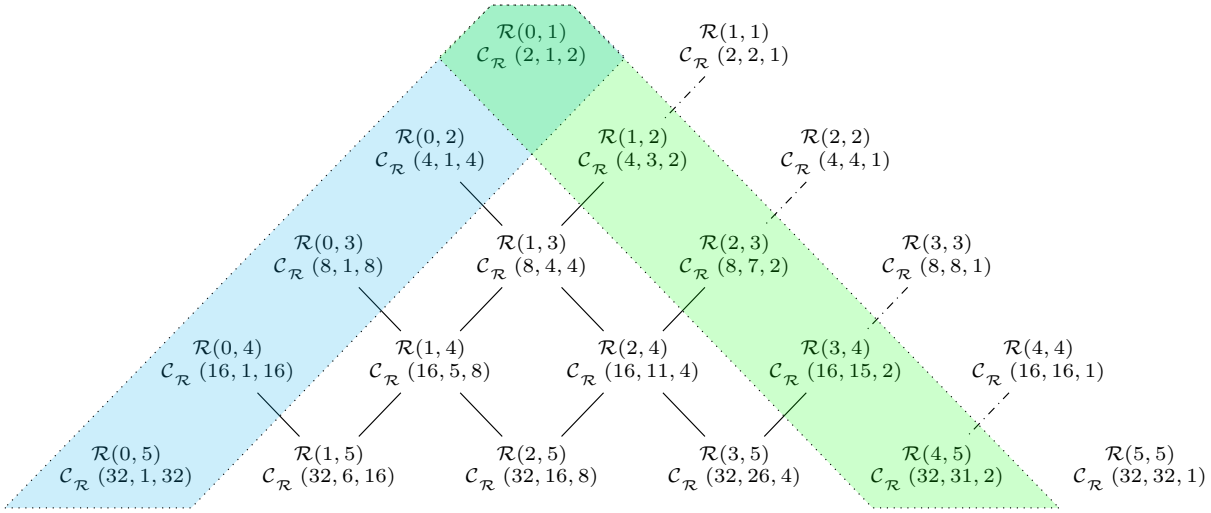


Figure 6.1: RM code tree (adapted from [101], ©1998 IEEE).

consistently maximizing the minimum Hamming distance.

The code tree demonstrates that the maximum dimension for the parameter set is attained when $r = m$, resulting in a code without parity, i.e., $d = 1$.

Characteristic code types can be observed on the left and right sides of the code tree in Figure 6.1. Codes with $r = 0$ and $m > 0$ represent repetition codes, while codes with $r = m - 1$ are SPC codes. These distinct code types are visible within the code tree.

Example 6.3. *Consider exemplary the code generator matrix construction of $\mathcal{R}(1,3)$. In this case, we use $\mathcal{R}(1,2)$ as $\mathcal{C}^{(1)}$ and $\mathcal{R}(0,2)$ as $\mathcal{C}^{(2)}$. The generator matrices of $\mathcal{C}^{(1)}$ and $\mathcal{C}^{(2)}$ are*

$$\mathbf{G}_1 = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \qquad \mathbf{G}_2 = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}$$

such that the code matrix of $\mathcal{R}(1,3)$ results in

$$\mathbf{G} = \begin{bmatrix} \mathbf{G}_1 & \mathbf{G}_2 \\ 0 & \mathbf{G}_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

6.3 Bose-Chaudhuri-Hocquenghem Codes

BCH codes constitute a significant class of cyclic ECCs published by Hocquenghem in 1959 and independently by Bose and Chaudhuri in 1960 [91]. The defining characteristic of BCH codes lies in their algebraic structure, specifically their polynomial description, which enables highly efficient decoding processes.

Binary BCH codes are constructed using polynomials over a Galois field of the form $GF\,(2^m)$. All calculations are performed with elements of the logarithm table of this Galois field, i.e., in extended finite number fields. For the purposes of this discussion, we will focus exclusively on binary BCH codes. The underlying mathematical concepts and principles will not be elaborated upon here.

A binary BCH code is defined by its parameters $\mathcal{B}(n, k, d)$. The following parameters hold for BCH codes [91]

$$\begin{aligned} n &= 2^m - 1 \\ n - k &\leq mt \\ d &\geq 2t + 1. \end{aligned}$$

In the context of MLC NAND flash memory, the use of HD decoding methods based on BCH codes has historically provided sufficient reliability [28, 102]. However, with the advancements of flash memory technology, particularly TLC and QLC flash memory, combined with the increasing miniaturization, stronger ECCs are required.

The subsequent sections provide a brief overview of the encoding and decoding processes of a BCH code. For a comprehensive and detailed discussion about this code family, the interested reader is referred to [92, 91, 103, 104].

6.3.1 BCH code construction

BCH codes are typically defined by a generator polynomial, which can be transformed into a generator matrix. The generator polynomial, denoted as $g(x)$, is generated for the encoding using primitive elements from a Galois field $GF(2^m)$. Each cyclic shift of a valid codeword results in another valid codeword. Next, we will provide a brief introduction to creating generator polynomials.

Cyclotomic cosets
For $n = 2^m - 1$, the cyclotomic cosets are defined as [104]

$$\mathcal{K}_i = \{i2^j \bmod n, \quad j = 0, 1, \ldots, m-1\}. \tag{6.8}$$

The largest possible number in a coset is $n - 1$. The smallest entry of each cyclotomic coset is called the coset representative.

BCH generator polynomial

Let α be a primitive element of the Galois field $GF(2^m)$. The generator polynomial of a cyclic code is determined by the union $\mathcal{M} = \mathcal{K}_j \cup \mathcal{K}_l \cup \ldots$ of cyclotomic cosets as

$$g(x) = \prod_{i \in \mathcal{M}} \left(x - \alpha^i\right) \tag{6.9}$$

.

First, we compute all cyclotomic cosets in $GF\left(2^m\right)$ to generate a generator polynomial with the previous steps. To achieve the desired minimum distance, exactly $d-1$ consecutive numbers need to be present in the union set $\mathcal{M}$, excluding K_0. Afterwards, the generator polynomial is calculated. Note that the actual minimum distance of the resulting code may be larger than initially planned.

Example 6.4. *In $GF(2^4)$, there are five cyclotomic cosets:*

$$K_0 = \{0\} \quad K_1 = \{1, 2, 4, 8\} \quad K_3 = \{3, 6, 9, 12\} \quad K_5 = \{5, 10\} \quad K_7 = \{7, 11, 13, 14\}$$

Suppose that the target minimum distance is $d = 5$. Therefore, the union set should comprise $\mathcal{M} = \mathcal{K}_1 \cup \mathcal{K}_3 = \{1, 2, 3, 4, 6, 8, 9, 12\}$. Applying Equation 6.9, the generator polynomial results in

$$g(x) = 1 + x^4 + x^6 + x^7 + x^8.$$

This generator polynomial belongs to a $\mathcal{B}(15, 7, 5)$ code.

One possibility to apply the encoding process involves the following steps:

1. An information polynomial in the form of:

   $$i\left(x\right) = i_0 + i_1 x + i_2 x^2 + \ldots + i_{k-1} x^{k-1}$$

   is multiplied by x^{n-k} to adjust the coefficients. Hereby, all elements of $i\left(x\right)$ are binary, i.e., $i_j \in GF(2)$.

2. Next, the resulting polynomial modulo the generator polynomial is calculated, and the residual polynomial is determined by [91]

   $$i\left(x\right) \cdot x^{n-k} \; mod \; g\left(x\right) = r\left(x\right)$$

   The remainder of the division $r(x)$ has degree $\leq (n-k-1)$.

3. Finally, the codeword polynomial $c(x)$ is formed from $r(x)$ and $i(x)$

   $$c\left(x\right) = r\left(x\right) + i\left(x\right) \cdot x^{n-k} (16)$$

The polynomial coding approach can be efficiently implemented as a hardware shift register, as described in [103, 92].

Certain applications require a length that is not reached with the standard code length of BCH codes $n = 2^m - 1$. Therefore, a process known as code shortening is performed to change the codeword lengths within the same extension field. Only a subset of the possible code constructions is utilized in this process. For the information polynomials, $i(x)$, it holds that $grad(i(x)) < k^* < k$ and the length of the shortened BCH code becomes $n^* = n - (k - k^*)$. This type of code shortening ensures a constant minimum distance. The unused portion of the code, i.e., the $k - k^*$ most significant digits, are set to zero [92].

Extended BCH codes represent a specific subtype of the BCH code family employed to enhance decoding failure detection by adding a parity digit to the codeword. When the minimum distance of an extended code is initially odd, a new minimum distance of $\hat{d} = d + 1 = 2 \cdot t + 2$ applies, ensuring that $\hat{d}$ is always an even number [92]. Extending to an odd weight is not permitted. The user must adjust the generator polynomial or matrix by including K_0 to the union set to correctly generate the codeword during the encoding process.

6.3.2 Decoding of binary BCH codes

The decoding process aims to verify the correctness of the codeword obtained and, if necessary, identify the error positions and correct the codeword.

BCH decoding involves three fundamental steps. First, the syndrome values are calculated to determine the validity of the received word $y(x)$. Hereby, the syndrome tuple is calculated, such as $\mathbf{yH} = \mathbf{S}$, where the parity-check matrix has the dimension $(n - k) \times n$. The algebraic structure allows simplifying the calculation of the Syndrome elements as [91]

$$S_i = y(\alpha^i) \quad \text{for } i = 0, ..., 2t.$$

If all syndrome elements are equal to 0, either the codeword is correct or no error is detectable. Otherwise, the syndrome elements are used in the subsequent decoding step, where the error-location polynomial $\Lambda(x)$ is calculated. The roots of $\Lambda(x)$ are the error locations, with the degree of the polynomial indicating the number of errors in the codeword obtained $y(x)$. Peterson introduced a procedure to calculate the elements of $\Lambda(x)$ in 1960 [105]. This method is efficient for codes with low error correction capabilities. Based on Peterson's algorithm, decoders have been proposed in [106, 91, 107, 108]. One widely used algorithm to generate $\Lambda(x)$ is the Berlekamp-Massey algorithm (BMA). This code is very efficient even for codes with high error correction capability and has been further enhanced in numerous publications [91, 109, 102, 110].

After defining the error location polynomial, the error positions are identified by finding the roots of $\Lambda(x)$. One efficient and commonly employed method involves conducting an exhaustive search. This method, known as the Chien search [111], involves calculating $\Lambda(\alpha^i)$ for $i = 0, 1, ..., n - 1$. If the calculation results in 0, α^i is recognized as a root of Λ. Finally, the

error positions determined are corrected by flipping the corresponding bits, particularly in the context of binary codes.

For BCH codes with moderate length, it is possible to implement the syndrome calculation and Chien search in parallel structures, ranging from $GF(2^6)$ to $GF(2^{12})$. This allows performing the calculations in only a single clock cycle [112, 113, 9]. However, the computation of the error location polynomial using BMA involves high computational complexity.

The algorithms are not discussed in more detail here, and instead interested readers are referred to [104, 92, 91].

6.3.3 Relation to Reed-Solomon codes

BCH codes were generalized to codes exploiting p^m symbols, where p represents a prime number, resulting in non-binary codes. The most important subclass of non-binary BCH codes is the class of RS codes published 1960 in [114], independently of the publications regarding the BCH code [91].

The decoding process for RS codes is similar to that for BCH codes. The non-binary nature requires some changes and additionally an error-value evaluator to identify the error magnitude. Binary codes can simply flip a bit once the error is located, whereas non-binary codes need to determine the error magnitude.

RS codes are maximum distance separable codes, meaning that they satisfy the condition $d = n - k + 1$. This characteristic contributes to the efficiency in error correction. Similar to BCH codes, RS codes offer the flexibility to truncate the code length.

Moreover, RS codes can handle combinations of errors ν and erasures ε. In this context, erasures ε are known error locations, and ν are unknown random error locations. This code family can correct errors and erasures according to the condition [91]

$$2\nu + \varepsilon < d. \tag{6.10}$$

In terms of the number of error corrections, this can be expressed as

$$\nu + \varepsilon/2 \leq t. \tag{6.11}$$

This means that we can correct twice as many erasures as errors, significantly enhancing their error correction capability.

RS codes are particularly effective at correcting burst errors, making them a suitable choice as ECCs for various communication and storage systems. A detailed explanation of the functionality of RS codes is omitted at this point. More in-depth information is found in [91, 92, 103, 104].

6.4 LDPC Codes for Flash Memory

LDPC codes, originally proposed by Robert Gallager in his PhD thesis [115], have experienced a resurgence of interest since their rediscovery in the 1990s [116]. LDPC codes initially faced challenges competing with other code classes, but over time they have been designed to exhibit exceptional error correction capabilities.

As the name suggests, the special feature of LDPC codes is their sparse parity-check matrix. *Sparse* in this context implies that the matrix contains a relatively small number of ones against the total number of entries. Each one in the matrix means a connection between the i-th check node and the j-th message node. This sparse property enables efficient encoding and decoding processes, making LDPC codes well suited for high-speed and bandwidth-limited applications.

Depending on the type, there are different construction specifications. The first LDPC codes proposed by Gallager belong to the category of regular LDPC codes. Specifically, each row has the same weight w_i, and each column has a constant weight w_j, while no two columns or two rows share the same pattern. By contrast, irregular LDPC codes do not have the same weights for columns or rows. By carefully designing the sparse parity-check matrix, LDPC codes can effectively detect and correct errors induced by noisy communication channels.

One remarkable advantage of LDPC codes is their ability to approach the Shannon limit, representing the maximum achievable data rate over a noisy channel. The use of irregular LDPC codes has enabled the construction of long codes that closely approach the Shannon limit, as demonstrated in [117]. This characteristic makes LDPC codes highly desirable for various communication standards and protocols, including wireless communication, satellite systems, digital video broadcasting, and optical communication.

The decoding of LDPC codes involves iterative algorithms that refine the estimated transmitted codeword by exchanging information between the parity-check equations and the bits received. These iterative decoding algorithms, such as belief propagation or the sum-product algorithm, demonstrate excellent error correction capabilities, enabling LDPC codes to achieve near-optimal performance.

Despite this, it is non-trivial to generate good LDPC codes. Computer searches and simulations are often necessary to find LDPC codes with desirable properties. LDPC codes are susceptible to problems like cycles in the graph, trapping sets, and pseudocodewords, among others, due to the iterative decoding algorithms. These challenges can result in an error floor, making it difficult to predict and guarantee performance [118, 119]. Consequently, computationally complex simulations are frequently employed to assess code performance.

Today, LDPC codes are used for flash systems with both hard- and soft-input decoding [14, 120, 89, 121], where soft-input decoding significantly improves the error correction performance [53, 89]. In particular, a quasi-cyclic LDPC code approach has been standardized for the use with flash memory [87]. In Chapter 8, we will leverage LDPC codes for a particular error correction implementation for flash memory.

Chapter 7

Soft-Decision Decoding of Short Cyclic Codes

Soft-decision (SD) decoding for flash memories represents an advanced decoding approach that leverages the reliability of the data received. This approach substantially improves decoding performance, in particular scenarios with poor channel conditions. Various decoding methods utilizing SD information can be found in the literature. In this chapter, our focus is on soft decoding techniques specifically designed for short cyclic codes.

Short ECCs are commonly employed in flash memory for managing the metadata. Previously, this specific information was solely corrected through HD decoding. However, with the introduction of SD decoding, this information can also be read under significantly degraded channel conditions.

The concept of code concatenation reduces the decoding complexity of long codes to small subproblems, significantly reducing overall decoding complexity. To further improve this coding strategy, we apply short cyclic codes with SD decoding. Code concatenation is introduced in Chapter 8.

We start this chapter with a brief example of how to apply soft-decision decoding to simple block codes. Subsequently, we examine two well-known SD decoding methods known as Chase decoding and ordered statistics decoding (OSD). Afterwards, we explore some methods to reduce the computational complexity of these approaches.

All of the methods presented in this chapter are applied for the ECCs featured in the following chapter. Parts of this chapter have been published in [5, 9, 10, 11].

7.1 Repetition and Single Parity-Check Codes

We start this chapter with the example of SD decoding of simple block codes. In Chapter 6, we have already introduced the definition of repetition and SPC codes. Now, we present how to leverage reliability information for these codes. Consider a binary symmetric channel with AWGN and binary phase shift keying (BPSK), i.e., the transmitted information is modulated in

the sign of the symbols. The vector received with soft information from the channel comprises $\mathbf{y} = [y_0, y_1, ..., y_{n-1}]$.

Considering the repetition code with soft information, we can apply the following formula to the information received [122]

$$\hat{y}_j = \text{sign}\left(\sum_{i=1}^{n-1} y_i\right) \quad \text{for } j = 0, ..., n-1.$$

Where $\hat{\mathbf{y}} = [\hat{y}_0, \hat{y}_1, ...\hat{y}_{n-1}]$ is the estimated HD codeword.

SPC codes using are not suitable for correcting bits with HD. However, with reliability information, a simple query can be applied to correct one digit if the codeword is erroneous. If the SPC code is not valid, we can find position j with $\min(y_j)$ for $(j = 0, ..., n-1)$ and flip the sign of this digit. Thus, we flip the least reliable position of the codeword received, resulting in a valid codeword.

Next, we will delve into enhanced SD algorithms designed for more powerful block codes.

7.2 Chase Decoding

Bit-flipping decoding of binary block codes was pioneered by Chase [123]. Chase introduced reliability information-based decoding procedures that generate a list of candidate codewords by flipping bits in the word received. The test patterns for the bit-flipping are based on the least reliable positions of the word received.

Originally, Chase distinguished three variants of his bit-flipping approach, each generating different test patterns [123, 92].

- Variant 1 generates all possible correction patterns up to weight $\lfloor d/2 \rfloor$, resulting in $\sum_{i=0}^{\lfloor d/2 \rfloor} \binom{n}{i}$ test patterns.
- Variant 2 generates all possible sets from the $\lfloor d/2 \rfloor$ most unreliable bit positions, i.e., there are $2^{\lfloor d/2 \rfloor}$ possible test patterns.
- Variant 3 generates only $\lfloor d/2 + 1 \rfloor$ test patterns, where each error pattern only flips the i unreliable positions of the codeword. If d is even then $i = 0,\ 2,\ \ldots,\ d-1$ and otherwise $i = 1,\ 3,\ \ldots,\ d-1$. Thus, only $\lfloor d/4 \rfloor$ test patterns are generated.

As the variants evolve, the complexity of the Chase algorithm decreases, but the residual error rate also deteriorates.

In the following, we use Variant 2 of the Chase algorithm while further restricting this variant. For instance, the p-Chase algorithm decodes the 2^p binary n-tuples obtained by considering all possible values in the p least reliable positions while $p \leq \lfloor d/2 \rfloor$ [124].

In general, the algebraic hard-input decoding is applied for each test pattern. Ultimately, the most likely codeword is selected from this list of candidates using a suitable metric. It was

shown by Chase in [123] that bit-flipping decoding can achieve near-ML performance and correct up to $d-1$ errors.

7.3 Ordered Statistic Decoding

In this section, we briefly discuss OSD, a bit-flipping procedure introduced in 1995 by Fossorier and Lin [125]. This approach belongs to the most reliable basis algorithm class, similar to the Dorsch decoder [126, 127, 128]. OSD is a generalization of the SD list decoding that can be applied to binary linear block codes $\mathcal{C}(n,k,d)$. We assume an AWGN channel with BPSK modulation.

The procedure starts by ordering the bits received, denoted as $\mathbf{y}=[y_0,y_1,...,y_{n-1}]$, based on their reliability, which is often represented by LLRs or other statistical models. The soft information vector is sorted in ascending order

$$|\tilde{y}_0| \leq |\tilde{y}_1| \leq ... \leq |\tilde{y}_{n-1}|. \tag{7.1}$$

The ordered vector is denoted as

$$\tilde{\mathbf{y}} = [\tilde{y}_0, \tilde{y}_1, ..., \tilde{y}_{n-1}]. \tag{7.2}$$

The smallest values correspond to the most unreliable bits, as they are closest to the decision threshold. This order defines a permutation function such that $\tilde{\mathbf{y}} = \lambda_1(\mathbf{y})$. Next, the generator matrix $\mathbf{G}$ of the code is transformed into a new matrix $\mathbf{G}'$ based on the k most reliable bits among the bits received, according to the soft information. The permutation function is applied to rearrange the generator matrix $\mathbf{G}$

$$\mathbf{G_1} = \lambda_1(\mathbf{G}) \tag{7.3}$$

performing column permutations. If necessary, the matrix $\mathbf{G_1}$ has to be reordered such that the first k columns are independent vectors. This order defines a second permutation function

$$\mathbf{G_2} = \lambda_2(\mathbf{G_1}) = \lambda_2(\lambda_1(\mathbf{G})). \tag{7.4}$$

The final permutation function λ results from the concatenation $\lambda = \lambda_2 \bullet \lambda_1$, i.e., $\mathbf{G}_2 = \lambda\,(\mathbf{G})$ Finally, Gaussian elimination is applied on $\mathbf{G_2}$ to obtain a matrix in systematic form

$$\mathbf{G}' = [\mathbf{I}|\mathbf{A}] \tag{7.5}$$

where $\mathbf{I}$ is an identity matrix and $\mathbf{A}$ comprises $n-k$ columns to determine the parity bits. Afterwards, the $n-k$ least reliable positions of the vector received are removed. The k most reliable bits are used to generate a candidate codeword applying G'.

The next steps of the original OSD algorithm propose flipping each possible combination of the k most reliable bits and calculating a metric for each candidate codeword. The goal is to select the ML codeword from the list of all valid candidate codewords. The bit-flipping procedure can be divided into various decoding sections, each of them flipping exactly θ bits. Order 0 implies that no position is flipped, requiring only one re-encoding step for the k most reliable elements in $\mathbf{y}$. With order 1, exactly k re-encoding steps are necessary because each possible position is flipped once. In general, order θ applies all possible bit-flipping patterns with θ flips, requiring $\binom{k}{\theta}$ re-encoding steps, respectively. In the final decoding step, the best codeword in the list is selected using an ML metric. Thus, the complete OSD algorithm requires 2^k re-encoding steps.

OSD is only applicable to linear block codes, as the fundamental characteristics of this code class are required for the ordering and re-encoding process. Furthermore, knowledge of the generator matrix of the original code is necessary for the procedure. The bits received must be ordered according to the reliability, which demands associated soft information.

According to [129], OSD achieves asymptotically near-ML performance with a maximum order $\theta = \lceil d/4 \rceil$. Consequently, the computational complexity of OSD is of order $O(k^{\frac{d}{4}})$, and the number of re-encoding steps for long codes with high minimum distance becomes immense. As an example, executing OSD up to order $\lceil d/4 \rceil = 6$ for the extended BCH code $\mathcal{B}(128, 64, 22)$ needs

$$\sum_{\theta=0}^{6} \binom{64}{i} = 83\,278\,001$$

re-encoding steps. OSD is suitable for codes with a small dimension or small minimum distance.

7.4 Acceptance Criteria for Soft-Input Decoding

Acceptance criteria can be employed during the decoding procedure to further reduce complexity. The decoding procedure is stopped immediately as soon as such criterion is fulfilled. In the following, a well-known concept is introduced and extended.

Kaneko's rule is a sufficient condition for finding the ML codeword. However, we will demonstrate that Kaneko's rule can be excessively restrictive. First, we briefly introduce notation and briefly review Kaneko's acceptance criterion. Afterwards, we discuss the new acceptance criterion and use it exemplary on OSD to reduce the complexity.

7.4.1 Kaneko's Acceptance Criterion

The complexity of OSD and Chase decoding can be significantly reduced by employing stopping criteria. These criteria allow the decoding process to terminate once a reliable codeword is identified [130, 131, 132, 54, 133]. Kaneko et al. proposed a stopping criterion in their work [130], which serves as a sufficient condition for a codeword to be the ML codeword.

In [132], this criterion was used to stop the OSD decoding process once a codeword satisfies Kaneko's acceptance criterion.

Consider a binary linear code $\mathcal{C}(n, k, d)$, where the Hamming distance between two binary vectors $\mathbf{a}, \mathbf{b}$ is denoted by $\mathrm{dist}(\mathbf{a}, \mathbf{b})$. We assume transmission of codewords $\mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{C}$ over an AWGN channel with BPSK modulation. Let $\mathbf{y}$ be the vector received. The hard decision vector $\hat{\mathbf{y}} = [\hat{y_0}, \hat{y_1}, \ldots, \hat{y}_{n-1}]$ is obtained as follows

$$\hat{y}_i = \begin{cases} 1, \, y_i \leq 0 \\ 0, \, y_i > 0 \end{cases} \tag{7.6}$$

Furthermore, we use the vector of soft values $\mathbf{y}$, which is sorted in ascending order as shown in Equation 7.1, resulting in $\tilde{\mathbf{y}}$ as denoted in Equation 7.2. The bit-flipping metric between a codeword $\mathbf{x}$ and the vector received $\mathbf{y}$ is defined as

$$l(\mathbf{y}, \mathbf{x}) = \sum_{x_i \neq \hat{y}_i} |y_i| \, . \tag{7.7}$$

Let $\mathfrak{m} = \mathrm{dist}\,(\mathbf{x}', \hat{\mathbf{y}})$ be the Hamming distance between a candidate codeword $\mathbf{x}'$ and $\hat{\mathbf{y}}$. Furthermore, let $\mathbf{x}_0$ denote the codeword found from a hard-input decoder with Hamming distance $\mathfrak{m}_\mathfrak{o} = \mathrm{dist}(\mathbf{x}_0, \hat{\mathbf{y}})$. Kaneko's acceptance criterion is defined as

$$l(\mathbf{x}', \mathbf{y}) \leq \sum_{i=1}^{d - \lfloor \frac{\mathfrak{m} + \mathfrak{m}_\mathfrak{o}}{2} \rfloor} |\tilde{y}_i|. \tag{7.8}$$

This criterion serves as a sufficient condition for classifying $\mathbf{x}'$ as an ML codeword [130]. Upon fulfilling the specified inequality, $\mathbf{x}'$ is considered the valid codeword.

7.4.2 Enhanced Acceptance Criterion

The computational complexity associated with Kaneko's acceptance criterion must be further reduced for specific applications. In the upcoming section, we introduce a hybrid decoding approach combining algebraic and soft-input decoding to reduce decoding complexity while reaching an acceptable performance. An improved acceptance criterion, such as a decoding-stopping rule, is essential to enable this. Consequently, we conduct an in-depth analysis of Kaneko's acceptance criterion and examine various properties of the bit-flipping metric. Let $p(\mathbf{y}|\mathbf{x})$ denote the conditional density function for the receiving vector $\mathbf{y}$ after transmitting the codeword $\mathbf{x}$ over the AWGN channel.

For the AWGN channel, we have

$$l(\mathbf{y}, \mathbf{x}'') - l(\mathbf{y}, \mathbf{x}') = \log \frac{p(\mathbf{y}|\mathbf{x}')}{p(\mathbf{y}|\mathbf{x}'')}, \tag{7.9}$$

where $\mathbf{x}'$ and $\mathbf{x}''$ are two codewords (proof in [54, 5]). By leveraging this, we can derive some interesting properties of the bit-flipping metric. In the subsequent proposition, $\mathbf{x}'$ denotes the ML codeword, and $\mathbf{x}''$ is the second best codeword.

Proposition 7.1.

- *a) The bit-flipping metric is an ML metric.*
- *b) If the HD vector $\hat{\mathbf{y}}$ is a codeword, then $\mathbf{x}' = \hat{\mathbf{y}}$ is the ML codeword.*
- *c) Let $\mathfrak{m} = \mathrm{dist}(\mathbf{x}', \hat{\mathbf{y}})$ be the Hamming distance between $\mathbf{x}'$ and $\hat{\mathbf{y}}$. Then, the metrics of the best and second-best codeword satisfy*

$$l(\mathbf{x}', \mathbf{y}) \geq \sum_{i=1}^{\mathfrak{m}} |\tilde{y}_i| \tag{7.10}$$

$$l(\mathbf{x}'', \mathbf{y}) \geq \sum_{i=1}^{d-\mathfrak{m}} |\tilde{y}_i| \tag{7.11}$$

Proposition 7.1 a) follows directly from Equation (7.9). The bit-flipping metric can be used for ML decoding, because $l(\mathbf{y}, \mathbf{x}') \leq l(\mathbf{y}, \mathbf{x}'')$ implies $p(\mathbf{y}|\mathbf{x}') \geq p(\mathbf{y}|\mathbf{x}'')$.

Proposition 7.1 b) follows from the fact that the bit-flipping metric is non-negative and $l(\hat{\mathbf{y}}, \mathbf{y}) = 0$.

To prove Proposition 7.1 c), we first consider the case where $\hat{\mathbf{y}}$ is a codeword, i.e. $\mathfrak{m} = 0$ and $l(\mathbf{x}', \mathbf{y}) = 0$. In this case, the Hamming distance $\mathrm{dist}(\mathbf{x}'', \hat{\mathbf{y}}) = \mathrm{dist}(\mathbf{x}'', \mathbf{x}')$ is at least d. Consequently, the metric $l(\mathbf{x}'', \mathbf{y})$ is the sum over at least d values z_i. This sum is lower bounded by

$$l(\mathbf{x}'', \mathbf{y}) \geq \sum_{i=1}^{d} |\tilde{y}_i|$$

due to the ordering of the soft values.

Consider a codeword $\mathbf{x}'$ with a Hamming distance of $\mathfrak{m}$ to the vector $\hat{\mathbf{y}}$. In this case, the codeword $\mathbf{x}''$ differs in at least $d - \mathfrak{m}$ positions from $\hat{\mathbf{y}}$. Consequently, the lower bound in (7.11) follows. Similarly, (7.10) follows from ordering the soft values.

From inequality (7.11), we can deduce that

$$l(\mathbf{x}', \mathbf{y}) \leq \sum_{i=1}^{d-\mathfrak{m}} |\tilde{y}_i| \tag{7.12}$$

is a sufficient condition for $\mathbf{x}'$ to be the ML codeword. In fact, this is a special case of Kaneko's acceptance criterion according to (7.8). Kaneko's acceptance criterion is a sufficient condition for $\mathbf{x}'$ being the ML codeword. However, (7.8) is not a necessary condition. In some cases, Kaneko's rule is too restrictive, as shown in the following proposition.

Proposition 7.2. *Let* $\mathbf{x}$ *be a candidate codeword with* $\mathfrak{m} = \mathrm{dist}(\mathbf{x}, \hat{\mathbf{y}})$*. Kaneko's acceptance criterion according to (7.8) is satisfied by a codeword* $\mathbf{x}$ *if*

$$\mathfrak{m} < \frac{2d+1}{3}. \tag{7.13}$$

The lower bound in (7.10) depends on the Hamming distance $\mathfrak{m}$ of the ML codeword. From this lower bound it follows that Kaneko's acceptance criterion can only be fulfilled if

$$\mathfrak{m} \leq d - \left\lfloor \frac{\mathfrak{m} + \mathfrak{m}_0}{2} \right\rfloor \tag{7.14}$$

holds. To obtain an upper bound on $\mathfrak{m}$, we assume that $\mathfrak{m}$ takes on the maximum value

$$\mathfrak{m} = d - \left\lfloor \frac{\mathfrak{m} + \mathfrak{m}_0}{2} \right\rfloor. \tag{7.15}$$

Now we bound the term $d - \lfloor \frac{\mathfrak{m}+\mathfrak{m}_0}{2} \rfloor$. Due to $\lfloor \frac{\mathfrak{m}+\mathfrak{m}_0}{2} \rfloor > \frac{\mathfrak{m}+\mathfrak{m}_0}{2} - 1$, we have

$$\mathfrak{m} < d - \frac{\mathfrak{m} + \mathfrak{m}_0}{2} + 1. \tag{7.16}$$

It follows from proposition 7.1 that $\mathfrak{m}_0 \geq 1$ if $\hat{\mathbf{y}}$ is not a codeword. Hence, we have

$$\mathfrak{m} < d - \frac{\mathfrak{m}}{2} + \frac{1}{2}. \tag{7.17}$$

This leads to the upper bound in (7.13).

According to Chase, bit-flipping decoding can achieve near-ML performance while correcting up to $d-1$ errors [123]. Consequently, the metric of the ML codeword can be the sum over $\mathfrak{m} = d-1$ reliability values, which is much larger than indicated in (7.13).

To enhance the detection rate, a modified acceptance criterion proposed in [54, 5] can be employed

$$l(\mathbf{x}', \mathbf{y}) \leq \sum_{i=0}^{M} |\tilde{y}_i| - T. \tag{7.18}$$

In this criterion, M and T serve as adjustable parameters that allow for a trade-off between decoding performance and the number of test patterns used.

The acceptance criterion is based on Yamamoto and Itoh's decision rule [134], which simplifies Forney's optimal criterion regarding the trade-off between erasure and error probability [135]. It has been demonstrated in [54] that this criterion offers a good approximation of Yamamoto and Itoh's original proposal. However, to obtain the best possible trade-off parameters, these parameters must be determined by Monte Carlo simulations.

This method only leads to a relatively small reduction of the computational complexity using OSD. A stopping criterion for OSD has recently been introduced [133], achieving a larger complexity reduction for high OSD orders. The methods proposed in [132] and [133] use at

least order-0 OSD, which has a complexity that is typically much higher than the complexity of algebraic HD decoding due to the costly Gaussian elimination required.

In the following section, we propose a hybrid decoding approach that combines algebraic decoding of binary block codes with soft-input decoding. The stopping criterion indicates that the hard-decoding result is sufficiently reliable for many codewords obtained. Consequently, the costly soft-input decoding can be omitted. The proposed acceptance criterion could be used with bit-flipping decoding and OSD.

7.5 Hybrid Ordered Statistics Decoding

Hybrid ordered statistics decoding (HOSD) is especially usable for low-complexity soft decoding of cyclic codes that provide an efficient HD decoding algorithm.

As explained in Section 7.3, the decoding complexity of OSD increases rapidly with the decoding order, requiring $\binom{k}{\theta}$ test patterns for the θ-th order. Consequently, the complexity scales as $O\left(k^{\theta}\right)$. In contrast to prior approaches [124, 136, 54], HOSD does not use a fixed number of decoding cycles.

The stopping criteria introduced in [132, 133] aim to reduce the decoding order of OSD. However, even order-0 has a decoding complexity that is typically much higher than conventional HD decoders due to the Gaussian elimination required to create the systematic form of $\mathbf{G}'$. The Gaussian elimination has to be applied on each received vector and involves $\min\left(k, n-k\right)^2 n$ elementary binary additions [125], i.e., it has a complexity order $O\left(n^3\right)$.

The computational complexity of standard algebraic decoding of cyclic codes is of order $O\left(nt\right)$ [137], where t is the error correction capability of the code. Thus, the primary goal of the proposed hybrid decoding approach is to avoid the computationally costly soft-input decoding entirely if the estimated codeword from the HD decoder is sufficiently reliable.

Therefore, we need a codeword acceptance criterion. To reduce the search space for optimal parameters for the acceptance criterion, we simplify equation 7.18 to

$$l(\mathbf{x}', \mathbf{y}) \leq \sum_{i=0}^{M} |\tilde{y}_i|, \tag{7.19}$$

where only the parameter M is necessary. OSD with a smaller order may use a larger value of M.

We propose a hybrid decoding approach that combines algebraic HD decoding of binary BCH codes with bit-flipping decoding. Figure 7.1 illustrates this HOSD approach.

In the initial decoding step, we perform a syndrome check to verify the codeword. According to Proposition 7.1, we stop the decoding process if the vector $\hat{\mathbf{y}}$ is a valid codeword.

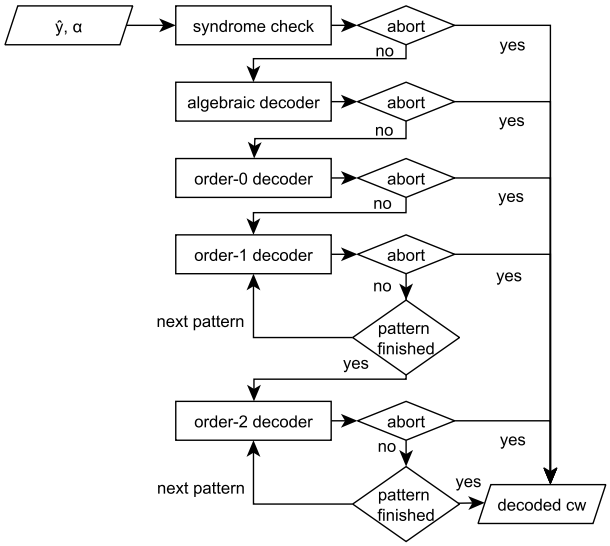


Figure 7.1: Flow chart of the proposed hybrid ordered statistics decoding (HOSD) approach up to order-2 decoding ([5], ©2020 IEEE).

In the second stage, we apply algebraic HD decoding based on the HD vector $\hat{\mathbf{y}}$. This step may result in an estimated codeword or a decoding failure. A failure is identified when, e.g., the error location polynomial indicates an odd number of errors while the parity bit suggests an even number, leading to an inconsistent solution of the error location polynomial. In such cases, we proceed with OSD.

After each decoding step, starting with the algebraic decoder, we assess the reliability of the estimated codeword according to Equation (7.19). The decoding process is stopped if the criterion is fulfilled. Otherwise, the decoding process advances to the next decoding step as indicated in Figure 7.1.

The stopping criterion indicates that the HD result is sufficiently reliable for codewords obtained. This allows us to bypass the resource-intensive bit-flipping decoding. In cases where bit-flipping decoding is necessary, we implement a procedure with a predetermined maximum number of decoding cycles and apply a decision rule to each candidate. The decoder can stop after each decoding round if the codeword satisfies the acceptance criterion. If no reliable codeword is found after the final decoding round, an erasure is declared.

Next we present some simulation results for BCH codes with the parameters $\mathcal{C}_1(63, 45, 7)$, $\mathcal{C}_2(63, 30, 13)$, and $\mathcal{C}_3(127, 99, 9)$. All simulations are considered for transmission over an AWGN channel with BPSK, and the SNR per bit is presented as E_b/N_0[dB]. Moreover, the decoding order-θ is denoted as O.θ in the legends of Figures 7.2 - 7.4.

For $\mathcal{C}_1(63, 45, 7)$ and $\mathcal{C}_2(63, 30, 13)$ codes, we adapt the threshold values M to the E_b/N_0, as provided in Table 7.1. It follows from Chase's results that $M = d - 1$ should achieve near-ML performance. For large E_b/N_0 values, $M = d - 1$ is required to ensure a low residual error rate. However, for lower E_b/N_0 values, we have chosen larger values of M to reduce the

Table 7.1: Threshold values depending on SNR.

E_b/N_0	2	2.5	3	3.5	4	4.5	5
M for $\mathcal{C}_1$	10	10	9	9	8	7	6
M for $\mathcal{C}_2$	16	16	16	15	14	13	12

decoding complexity. For the $\mathcal{C}_3(127, 99, 9)$ BCH code, we consider a combination with the stopping rule proposed in [133] with HOSD.

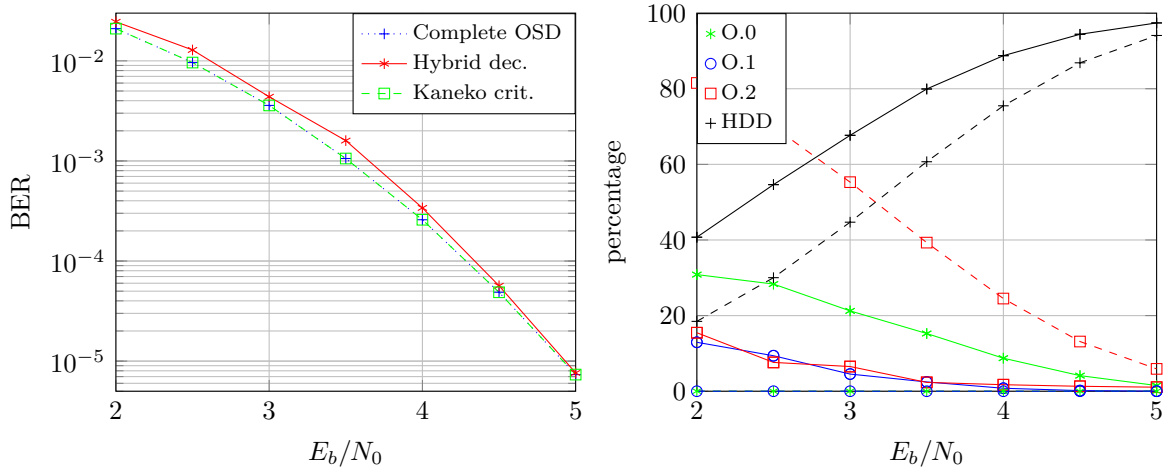


Figure 7.2: Decoding performance and complexity for the $\mathcal{C}_1(63, 45, 7)$ BCH code. Solid lines present results for the proposed threshold, dashed lines for Kaneko's acceptance criterion (7.13), and dots for complete order-2 decoding (adapted from [5], ©2020 IEEE).

For $n = 63$, Figures 7.2 and 7.3 depict the BER complexity, respectively. In the case of the $\mathcal{C}_1(63, 45, 7)$ code, order-2 decoding is sufficient to achieve near-ML performance, as indicated in [92], while order-3 decoding is required for the $\mathcal{C}_2(63, 30, 13)$. Hence, in Figures 7.2 and 7.3, we compare the decoding performance of the proposed decoding approach with complete order-2 and order-3 decoding without stopping, respectively. These results demonstrate that the proposed decoding method only incurs a marginal performance loss. Figure 7.2 illustrates results for the decoding with Kaneko's acceptance criterion.

The proposed hybrid decoding can significantly reduce the decoding complexity. The graphs on the right depict the percentage of decoding attempts that were stopped at the indicated decoding stage. In the case of code $\mathcal{C}_1(63, 45, 7)$, at $E_b/N_0 = 4$ dB, approximately 90% of all decoding attempts are successfully handled using HD decoding. With Kaneko's acceptance criterion, only about 75% of the decoding attempts are stopped after HD decoding, and the remaining 25% require order-2 decoding. The probability that Kaneko's rule stops the decoding after order-0 or order-1 is very low. We observe that Kaneko's acceptance criterion is too restrictive, resulting in notably higher decoding complexity. Evidently, larger decoding orders are required at low E_b/N_0. With the proposed acceptance criterion, 36% to 41%

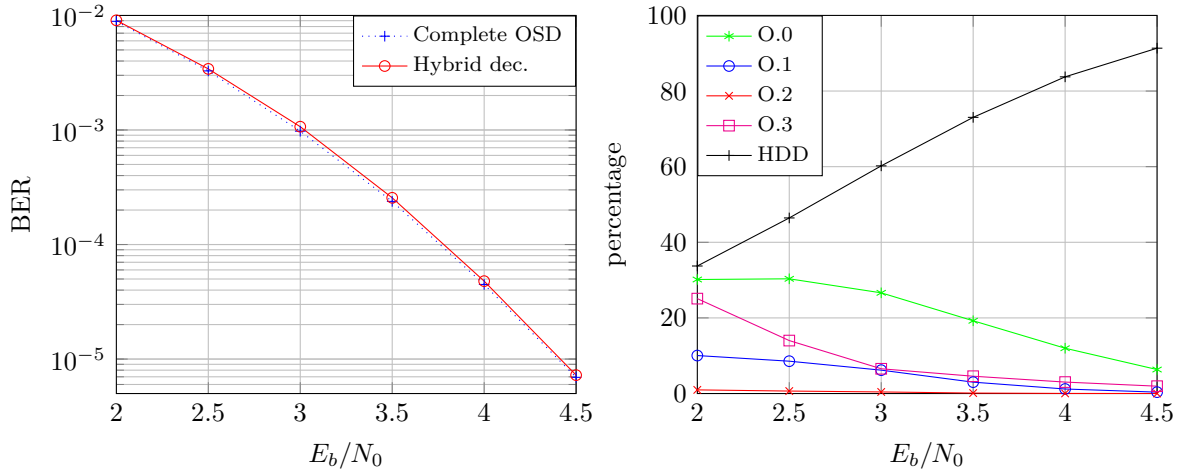


Figure 7.3: Decoding performance and complexity for the $\mathcal{C}_2(63, 30, 13)$ BCH code using the proposed threshold (adapted from [5], ©2020 IEEE).

of all vectors received for $\mathcal{C}_1$ and $\mathcal{C}_2$ can be decoded with HD decoding at $E_b/N_0 = 2$ dB, whereas order-2 decoding is required for 80% of all vectors with Kaneko's acceptance criterion at $E_b/N_0 = 2$ dB.

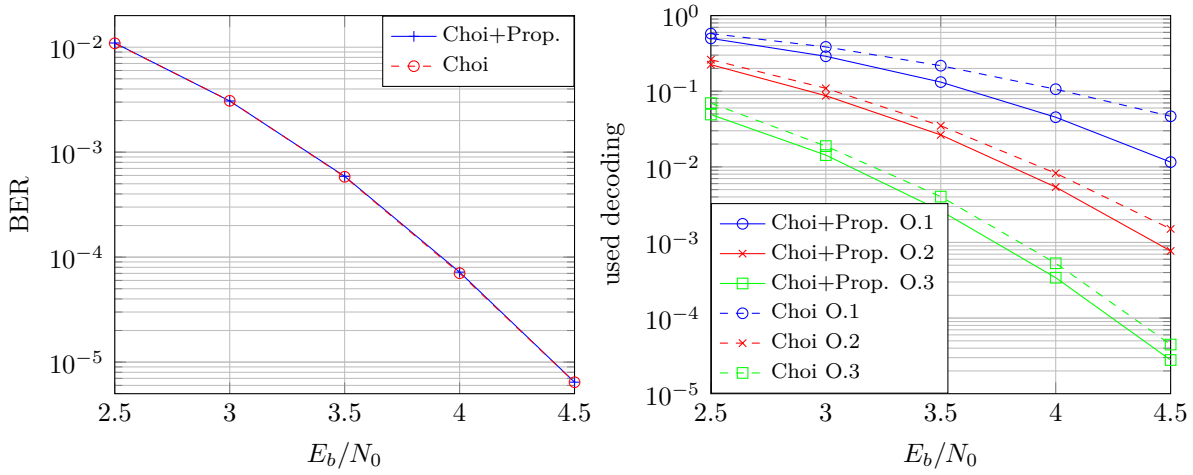


Figure 7.4: Decoding performance and the logarithm of complexity for the $\mathcal{C}_3(127, 99, 9)$ BCH code. Dashed lines depict the decoding procedure of Choi [133] with $\beta = 0.4$, $\tau_E = 4$, and $\tau_{PSC} = 14$. Solid lines illustrate a combination with HOSD with $\beta = 0.5$ and $M = 9$ (adapted from [5], ©2020 IEEE).

For higher decoding orders, combining the proposed decoding algorithm with the stopping rule from [133] is more efficient. In this case, we use probabilistic sufficient condition (PSC) methods with the variable τ_E, which serves as an early termination parameter, checking if the number of errors in the vector received is within the error-correcting capability, and τ_{PSC} is the upper bound to prune out impossible candidate codewords [133]. These parameters can be adjusted empirically. Furthermore, an order skipping rule called the probabilistic necessary

condition (PNC) is introduced in [133], which enables avoiding some particular OSD orders. This lower bound is calculated such

$$s_{PNC,\theta} = \sum_{i=n-k}^{n-k+\theta} |\tilde{y}_i|. \tag{7.20}$$

The trade-off can be further manipulated by the parameter β such that

$$s_{skip,\theta} = s_{PNC,\theta} + \beta(n-k). \tag{7.21}$$

In Figure 7.4, we compare the complete decoding approach from [133] with a combined approach. For this comparison, we used the criterion from [133] for orders $\theta \geq 1$. This criterion involves skipping an order if the Hamming distance for the best candidate is lower than a given threshold, with the threshold being adapted for each order. Once an order-i is entered, the acceptance criterion suggested in Equation (7.19) is applied to accept a codeword. Note that a constant threshold of $M = 9$ was used for this simulation with HD decoding and order-0 decoding, while the stopping rule from [133] is used for higher orders. As illustrated in Figure 7.4, combining this hybrid decoding approach halves the decoding complexity without performance loss.

We use the algebraic decoder on the punctured code to apply this approach with extended BCH codes and RM codes. After the algebraic decoding, a parity bit is added to the resulting codeword. If the algebraic decoding succeeds, this results in a codeword $\mathbf{x}'$ of the extended BCH code or the RM code. Consequently, the acceptance criterion (7.19) can be applied. In cases where the algebraic decoder fails or the acceptance criterion is violated, the decoder resorts to OSD.

To minimize complexity, we will describe a simplified version of the skipping rule inspired by [133]. Let $\mathbf{x}'$ denote the best codeword found after order $\theta = 1$. Order θ is only performed if

$$l(\mathbf{x}', \mathbf{y}) \geq \delta + \sum_{i=n-k}^{n-k+\theta} |\tilde{y}_i| \tag{7.22}$$

Thus, an order is only executed if the metric of the current codeword exceeds the sum of the θ smallest magnitudes on the most reliable basis. Similar to the parameter M, the parameter δ enables a trade-off between complexity reduction and decoding performance. Supposing that the condition in (7.22) is violated. It becomes unlikely that order θ will yield a superior codeword candidate, and thus the order can be skipped without a significant loss in decoding performance. This approach proves particularly advantageous when higher orders are used.

This section has shown that OSD can be entirely omitted with the proposed hybrid decoding approach when the hard-input decoder provides a reliable codeword. The reliability of codeword candidates is assessed against a new acceptance criterion. This criterion is similar to Kaneko's acceptance criterion for bit-flipping decoding but is less restrictive. It does not guarantee that an accepted codeword candidate is the ML codeword, but simulation results indicate that its

reliability is sufficient to achieve near-ML performance. Moreover, the hybrid decoding results in a substantially reduced decoding complexity compared to conventional OSD, where OSD with a large order is only necessary for a small percentage of vectors received.

7.6 Test Pattern Reduction

In the following, we explore the shift-sum decoder, as proposed in [138, 139], to reduce the number of test patterns required for HOSD. This decoding concept involves deriving reliability information for code bits by applying products of the vector of HDs $\hat{\mathbf{y}}$ with cyclic shifts of low-weight codewords from the dual code $\mathcal{B}^{\perp}$. These results are counted symbol-wise.

The concept of the shift-sum decoder is based on the orthogonality of the code and the dual code, i.e., the scalar product of a codeword from the code and a codeword from the dual code is zero. A non-zero scalar product with the vector received of HDs indicates errors in the code symbols corresponding to the support of the codeword from the dual code. The shift-sum decoder determines multiple scalar products and is an efficient method to count the number of non-zero scalar products for each code symbol.

In the following, we represent vectors as polynomials, e.g., $\hat{y}(x)$ represents the vector $\hat{\mathbf{y}}$. The support of a polynomial $b(x) = b_0 + b_1 x + b_2 x^2 + \ldots b_{n-1} x^{n-1}$ is denoted by $\sup b(x)$, reflecting the set of indices i such that $b_i \neq 0$. Let $\mathcal{B}^{\perp}$ be the dual code to the cyclic code $\mathcal{B}$. The minimum Hamming distance in the dual code is denoted by $d^{\perp}$. The product of any codeword $b(x) \in \mathcal{B}$ and any dual codeword $b^{\perp}(x) \in \mathcal{B}^{\perp}$ is zero, i.e.,

$$b(x)b^{\perp}(x) \mod (x^n - 1) = 0. \tag{7.23}$$

The word received $\hat{y}(x) = b(x) + e(x)$ is the sum of a codeword $b(x) \in \mathcal{B}$ and an error polynomial $e(x)$, where the product

$$w(x) = \hat{y}(x)b^{\perp}(x) \mod (x^n - 1) = e(x)b^{\perp}(x) \mod (x^n - 1) \tag{7.24}$$

depends on the error polynomial $e(x)$ and $b^{\perp}(x)$. Consider a codeword of weight $d^{\perp}$ of the form $b^{\perp}(x) = 1 + x^{b_1} + \ldots + x^{b_{d^{\perp}-1}}$. In the case where there is an error in position i, the polynomial $w(x)$ has $d^{\perp}$ non-zero coefficients, among them the error position i. Now, consider the sum over the $d^{\perp}$ cyclic shifts $x^{-j}w(x) \mod (x^n - 1)$, $j \in \sup b^{\perp}(x)$, where the summation of the coefficients is performed as an integer addition and not as an operation in the binary field. This sum

$$\mathbf{W} = \sum_{j \in \sup b^{\perp}(x)} x^{-j} w(x) \mod (x^n - 1) \tag{7.25}$$

defines a vector of length n, where every error position is counted up to $d^{\perp}$ times. A large value in $\mathbf{W} = [W_0, W_1, ..., W_{n-1}]$ indicates a possible error position.

Example 7.1. *To illustrate the reliability information of the vector* $\mathbf{W}$*, we consider the BCH code* $\mathcal{B}(15, 7, 5)$*. The dual code* $\mathcal{B}^{\perp}(15, 8, 4)$ *has a minimum Hamming distance* $d^{\perp} = 4$*. For the decoding, we consider the codeword* $b^{\perp}(x) = 1 + x^8 + x^{12} + x^{14} \in \mathcal{B}^{\perp}$*. Now, assume that the all-zero codeword is transmitted and the vector* $\hat{y}(x) = 1 + x$ *is received, i.e., there are two errors in the first two positions. According to (7.25), we obtain*

$$\mathbf{W} = [3, 3, 2, 0, 2, 1, 1, 0, 2, 2, 1, 1, 2, 2, 2],$$

starting with index 0 on the left side. The two largest values in $\mathbf{W}$ *correspond to the two error positions.*

The previous example illustrates that large values in $\mathbf{W}$ indicate unreliable bit positions. A larger number of codewords from the dual code enlarges the differences between the counter values for error positions and error-free positions. The shift-sum decoder uses D codewords $b^{(l)}(x) \in \mathcal{B}^{\perp}$ of weight $d^{\perp}$ and the form $b^{(l)}(x) = 1 + x^{b_1} + \ldots + x^{b_{d^{\perp}-1}}$. The decoder calculates the D products

$$w^{(l)}(x) = \hat{y}(x) b^{(l)}(x) \mod (x^n - 1),\ l = 1, \ldots, D. \tag{7.26}$$

The polynomials $w^{(l)}(x)$ are used to determine the vector $\mathbf{W}$ of reliability values as

$$w_j = \sum_{l=1}^{D} \sum_{i \in \operatorname{sup} b^{(l)}(x)} w^{(l)}_{j+i \mod n},\ j = 0, 1, \ldots, n-1 \tag{7.27}$$

where again the summation of the coefficients is performed as an integer addition and not as an operation in the binary field. Hence, $\mathbf{W}$ is a vector where each component is an integer between 0 and $Dd^{\perp}$. The $\mathbf{W}$ calculation requires at most $nDd^{\perp}$ integer increments.

The vector $\mathbf{W}$ is used in [138] for a bit-flipping decoder that flips the code bits corresponding to the μ largest values in $\mathbf{W}$. In contrast to [138], we will focus on the smallest values in $\mathbf{W}$, which indicate reliable bit positions. Consider the μ smallest values in $\mathbf{W}$. The corresponding indices define a set Ω, which is not necessarily unique. In Example 7.1, we have $\Omega = \{3, 5, 6, 7, 10, 11\}$ for $\mu = 6$.

We can reduce the number of test patterns for OSD by detecting more reliable bits in the most reliable basis $\mathcal{I}$. In particular, we define the set of reliable bit positions as

$$\mathcal{I}_r = \Omega \cap \mathcal{I}. \tag{7.28}$$

The test patterns for the bit-flipping will concentrate on the relative complement $\mathcal{I}_c = \mathcal{I} \setminus \mathcal{I}_r$. We illustrate this in the following example.

Example 7.2. *Again, we consider the code from Example 7.1 and the all-zero codeword, but for transmission over the AWGN channel with the vector received*

$$\mathbf{y} = [\text{-}1.2, \text{-}0.5, 2.6, 2.1, \text{-}0.1, 0.2, 1.7, 0.2, 1.1, 0.15, 0.8, 1.9, 1.8, 0.6, 0.7]. \tag{7.29}$$

This corresponds to three errors and $\hat{y}(x) = x^0 + x^1 + x^4$. *The most reliable basis is* $\mathcal{I} = \{0, 2, 3, 6, 8, 10, 11, 12\}$. *The set* $\mathcal{I}$ *contains a single error with the index 0. Using the* $\mu = 9$ *smallest reliability values in*

$$\mathbf{W} = [2, 2, 1, 1, 2, 2, 2, 1, 1, 2, 2, 2, 1, 1, 2],$$

we divide the set $\mathcal{I}$ *into the set of reliable positions* $\mathcal{I}_r = \{2, 3, 8, 12\}$ *and the complement* $\mathcal{I}_c = \{0, 6, 10, 11\}$.

To reduce the number of test patterns, we apply different decoding orders to the reliable set and its complement, i.e., we flip at most θ_r bits in the $k_r = |\mathcal{I}_r|$ reliable positions from the set $\mathcal{I}_r$ and up to $\theta_c = \theta - \theta_r > \theta_r$ bits in the remaining $k_c = k - k_r$ positions. To estimate the complexity reduction for the worst-case complexity, we consider the leading term k^θ for the number of encoding steps for order θ OSD. Choosing μ such that $k_r = k_c = k/2$, the leading term is reduced to

$$\left(\frac{k}{2}\right)^{\theta_r} \left(\frac{k}{2}\right)^{\theta_c} = \left(\frac{k}{2}\right)^{\theta} = \left(\frac{1}{2}\right)^{\theta} k^\theta.$$

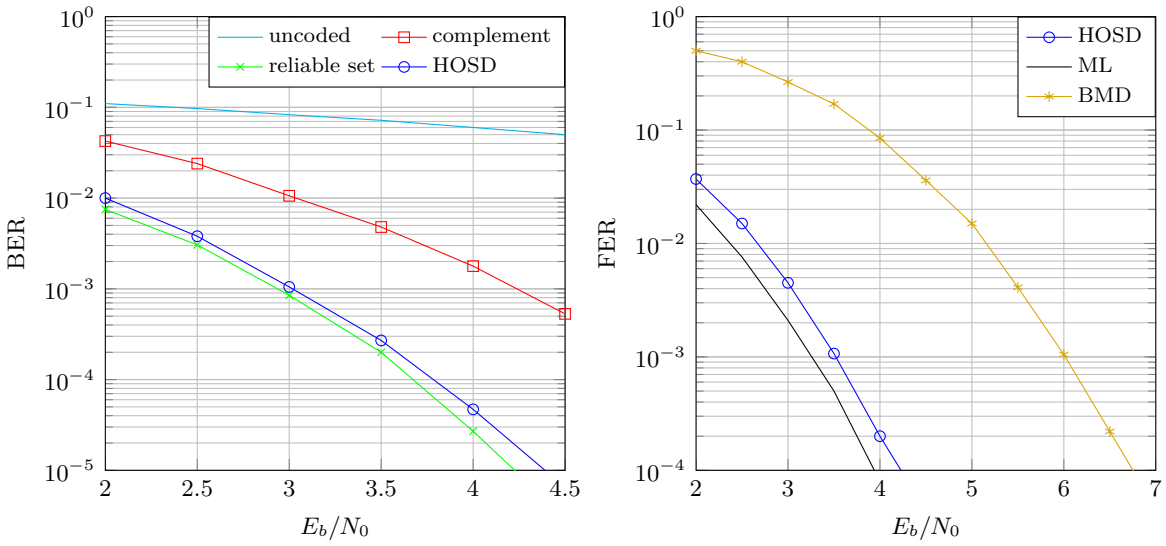


Figure 7.5: Simulation results for the code $\mathcal{B}(63, 30, 13)$ with different decoding methods; HOSD with $\theta_c = 1$, $\theta_c = 2$, $D = 78$ and $d^\perp = 12$ (adapted from [11], ©2022 IEEE).

The HOSD proposed is illustrated in Figure 7.5, which depicts simulation results for the BCH code $\mathcal{B}(63, 30, 13)$ over the AWGN channel. The left-hand figure presents BERs, i.e., the channel error rate (uncoded) as well as the BERs in the sets $\mathcal{I}_r$ (reliable set) and $\mathcal{I}_c$ (complement). Due to the ordering of the channel values, the error probability in the set $\mathcal{I}_c$

is reduced compared to the uncoded transmission. The shift-sum decoding identifies the set $\mathcal{I}_r$ of more reliable bit positions. We use $\theta_r = 1$ and $\theta_c = 2$ for the proposed decoding. This results in a near-ML performance as shown in the right part of Figure 7.5, where FERs are presented. The proposed decoder achieves a gain of 2.5 dB compared to bounded minimum distance decoding (BMD), whereas the difference compared to ML is about 0.25 dB.

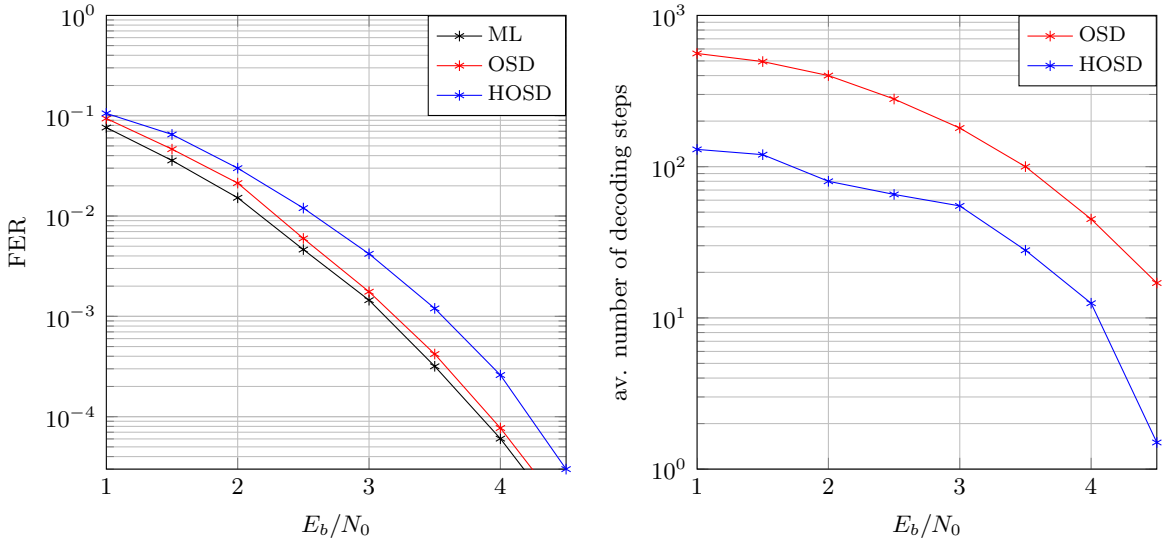


Figure 7.6: Simulation results for the code $\mathcal{B}(63, 24, 15)$ with OSD applying Box-and-Match algorithm with $s = 36$; HOSD with $\theta_r = 1$, $\theta_c = 2$, $D = 35$ and $d^{\perp} = 8$ (adopted from [11], ©2022 IEEE).

Complete order $\theta = 3$ decoding of the code $\mathcal{B}(63, 30, 13)$ requires $4\,526$ encoding steps. In the worst case, the proposed decoding still requires $1\,696$ encoding steps. However, the average complexity is much lower. On average, 122 codewords are considered at $E_b/N_0 = 2$ dB, and only 6.6 encoding steps are required at $E_b/N_0 = 4.5$ dB.

We consider the BCH code $\mathcal{B}(63, 24, 15)$ as a second example for the proposed decoding. Simulation results for the AWGN channel are presented in Figure 7.6. The left-hand figure shows FERs for the proposed decoding, a lower bound for the ML performance derived from [92], and results using OSD with Box-and-Match algorithm. For this code, the proposed decoder achieves a coding gain of about 1dB at FER$=10^{-4}$ compared to the shift-sum decoder proposed in [138], where the HOSD requires only three encoding steps on average.

The Box-and-Match algorithm of order θ can apply test patterns of up to order $(\theta \cdot 2)$, which increases the correction capability [140]. The size of the control band, denoted as $s - k$, determines the required size of the memory and the number of additional test patterns. More details can be found in [140].

Both OSD algorithms considered in Figure 7.6 flip up to $\theta = 3$ bits and use the hybrid decoding approach discussed in Section 7.5. Both algorithms reduce the number of OSD encoding steps compared to full order $\theta = 3$ OSD, which requires $2\,325$ encoding steps for

the code $\mathcal{B}(63, 24, 15)$. However, conducting a fair complexity comparison proves challenging because the Box-and-Match algorithm requires more memory while HOSD demands more computations for the shift-sum algorithm. For instance, the shift-sum algorithm can determine the set of reliable bit positions necessary for the enhanced Box-and-Match algorithm proposed in [141]. In essence, HOSD entails fewer encoding steps and less memory consumption than the Box-and-Match approach, yet it achieves superior performance.

The presented concepts are adaptable for decoding concatenated codes, which will be discussed in the next chapter. Furthermore, it is feasible to combine the two methods.

Chapter 8

Code Concatenation and Complexity Reduction

Code concatenation is the concept of combining small to medium-sized codes to create more extended codes. Forney introduced the concept of code concatenation in 1966 [142], involving an inner and an outer ECC. Longer ECCs typically have higher encoding and decoding complexities. On the other hand, they offer superior error correction capabilities. The benefits of code concatenation include the ability to generate long codes from short ones, reducing the complexity associated with encoding and decoding. Moreover, this technique allows exploiting specific error correction properties [92].

We introduced some fundamentals of ECCs and decoding techniques in the preceding chapters. Flash memories typically require very long codes for storing payload data and a high error-correcting capability. Especially for metadata, short ECCs are necessary due to the limited amount of data involved. These data pose specific requirements because they should be readable even after the EOL state of the flash memory, and the readout process should have very low complexity. Powerful error correction often requires SD decoding, which increases the complexity. In this context, we propose low-complexity concatenated codes with short to medium code lengths designed for metadata.

Various methods can be employed for code concatenation. This chapter will first explore the Plotkin code concatenation to build codes based on extended BCH component codes and describe efficient decoding methods. Afterwards, we will delve into the concept of generalized concatenated (GC) codes, its encoding and decoding procedure, and complexity reduction methods. The notation of this chapter is described in chapter 6. Parts of this chapter were published in [2, 9, 11, 12].

8.1 Plotkin Concatenation with BCH Component Codes

The Plotkin construction, introduced in 1960 [100], offers a recursive method for concatenating binary codes to achieve specific parameters. This construction allows the use of any linear

binary block codes. We represent such codes as $\mathcal{C}(n, k, d)$ in the following. Notably, the Plotkin construction can be employed to generate RM codes, as discussed in Section 6.2.

We employ extended BCH codes as component codes to improve the code parameters compared to RM codes. Notably, for $m \geq 6$, extended BCH codes offer better code parameters than comparable RM codes. These improvements include achieving a higher minimum distance while maintaining the same code length and rate. For instance, we can compare the $\mathcal{C}_{\mathcal{R}}(128, 64, 16)$ with the parameters $(r = 3, m = 7)$ to the extended BCH code $\mathcal{B}(128, 64, 22)$.

Similar to BCH codes, RM codes can also be constructed based on cyclotomic cosets [143, 144]. A punctured RM code of length $n = 2^m - 1$ is a cyclic code. However, for $m \geq 6$, binary BCH codes and RM codes are typically based on different cyclotomic cosets. We illustrate this in the following example.

Example 8.1. *We consider BCH codes and punctured RM codes for $m = 6$ and $n = 2^m - 1 = 63$. There exist thirteen cyclotomic cosets with the coset representatives:*

$$\{0, 1, 3, 5, 7, 9, 11, 13, 15, 21, 23, 27, 31\}$$

for $n = 63$. The code $\mathcal{B}(63, 24, 15)$ results from the coset representatives $\{1, 3, 5, 7, 9, 11, 13\}$, which are consecutive odd integers. To construct the punctured RM code with minimum distance $d = 15$, we have to select all cosets, where the binary representation of a representative has at most Hamming weight three [143]. For instance, the decimal number 13 has the binary representation $(1, 1, 0, 1)$ with Hamming weight three. This construction leads to the coset representatives $\{1, 3, 5, 7, 9, 11, 13, 21\}$, and the punctured RM code $\mathcal{C}_{\mathcal{R}}(63, 22, 15)$. Both codes are cyclic codes and have minimum distance $d = 15$, but the BCH code has the higher dimension.

Table 8.1 displays the code parameters for concatenated codes of length $n = 128$. These codes are constructed from two BCH component codes. To construct a code with minimum a distance d, we select component code 2 with a minimum distance $d_2 = d$ and component code 1 with the largest dimension that satisfies $d_1 \geq d/2$. The codes marked with bold digits in Table 8.1 have a minimum distance that is a power of two, making them comparable to RM codes of the same length. For instance, the code $\mathcal{C}(128, 69, 16)$ can be compared to the RM code $\mathcal{C}_{\mathcal{R}}(128, 64, 16)$ with parameters $(r = 3, m = 7)$. Similarly, the concatenated codes $\mathcal{C}(128, 102, 8)$ and $\mathcal{C}(128, 31, 32)$ have larger dimensions compared to the RM codes $\mathcal{C}_{\mathcal{R}}(128, 99, 8)$ $\mathcal{C}_{\mathcal{R}}(128, 29, 32)$, respectively. Table 8.1 also illustrates that this construction offers more flexibility than the RM construction because it is not limited to codes where the minimum distance is a power of two.

Table 8.2 provides selected examples of multiple concatenations, where we use the codes from Table 8.1 as component codes. The resulting codes have length $n = 256$. Some of these concatenated codes can be compared to RM codes with the same minimum distance. For instance, the code $\mathcal{C}(256, 171, 16)$ has a higher dimension than the RM code $\mathcal{C}_{\mathcal{R}}(256, 163, 16)$.

Table 8.1: Concatenated codes of length $n = 128$ ([11], ©2022 IEEE).

component 1	component 2	concatenated code
$\mathcal{B}(64, 63, 2)$	$\mathcal{B}(64, 57, 4)$	$\mathcal{C}(\mathbf{128}, \mathbf{120}, \mathbf{4})$
$\mathcal{B}(64, 57, 4)$	$\mathcal{B}(64, 51, 6)$	$\mathcal{C}(128, 108, 6)$
$\mathcal{B}(64, 57, 4)$	$\mathcal{B}(64, 45, 8)$	$\mathcal{C}(\mathbf{128}, \mathbf{102}, \mathbf{8})$
$\mathcal{B}(64, 51, 6)$	$\mathcal{B}(64, 39, 10)$	$\mathcal{C}(128, 90, 10)$
$\mathcal{B}(64, 51, 6)$	$\mathcal{B}(64, 36, 12)$	$\mathcal{C}(128, 87, 12)$
$\mathcal{B}(64, 45, 8)$	$\mathcal{B}(64, 30, 14)$	$\mathcal{C}(128, 75, 14)$
$\mathcal{B}(64, 45, 8)$	$\mathcal{B}(64, 24, 16)$	$\mathcal{C}(\mathbf{128}, \mathbf{69}, \mathbf{16})$
$\mathcal{B}(64, 39, 10)$	$\mathcal{B}(64, 18, 22)$	$\mathcal{C}(128, 57, 20)$
$\mathcal{B}(64, 36, 12)$	$\mathcal{B}(64, 18, 22)$	$\mathcal{C}(128, 54, 22)$
$\mathcal{B}(64, 36, 12)$	$\mathcal{B}(64, 16, 24)$	$\mathcal{C}(128, 52, 24)$
$\mathcal{B}(64, 30, 14)$	$\mathcal{B}(64, 10, 28)$	$\mathcal{C}(128, 40, 28)$
$\mathcal{B}(64, 24, 16)$	$\mathcal{B}(64, 7, 32)$	$\mathcal{C}(\mathbf{128}, \mathbf{31}, \mathbf{32})$
$\mathcal{B}(64, 18, 22)$	$\mathcal{B}(64, 1, 64)$	$\mathcal{C}(128, 19, 44)$
$\mathcal{B}(64, 16, 24)$	$\mathcal{B}(64, 1, 64)$	$\mathcal{C}(128, 19, 48)$
$\mathcal{B}(64, 10, 28)$	$\mathcal{B}(64, 1, 64)$	$\mathcal{C}(128, 11, 56)$
$\mathcal{B}(64, 7, 32)$	$\mathcal{B}(64, 1, 64)$	$\mathcal{C}(\mathbf{128}, \mathbf{8}, \mathbf{64})$

Table 8.2: Concatenated codes of length $n = 256$ ([11], ©2022 IEEE).

component 1	component 2	concatenated code
$\mathcal{C}(128, 120, 4)$	$\mathcal{C}(128, 102, 8)$	$\mathcal{C}(\mathbf{256}, \mathbf{222}, \mathbf{8})$
$\mathcal{C}(128, 108, 6)$	$\mathcal{C}(128, 90, 10)$	$\mathcal{C}(256, 198, 10)$
$\mathcal{C}(128, 108, 6)$	$\mathcal{C}(128, 87, 12)$	$\mathcal{C}(\mathbf{256}, \mathbf{195}, \mathbf{12})$
$\mathcal{C}(128, 102, 8)$	$\mathcal{C}(128, 75, 14)$	$\mathcal{C}(256, 177, 14)$
$\mathcal{C}(128, 102, 8)$	$\mathcal{C}(128, 69, 16)$	$\mathcal{C}(\mathbf{256}, \mathbf{171}, \mathbf{16})$
$\mathcal{C}(128, 90, 10)$	$\mathcal{C}(128, 57, 20)$	$\mathcal{C}(256, 147, 20)$
$\mathcal{C}(128, 87, 12)$	$\mathcal{C}(128, 52, 24)$	$\mathcal{C}(256, 139, 24)$
$\mathcal{C}(128, 75, 14)$	$\mathcal{C}(128, 40, 28)$	$\mathcal{C}(256, 115, 28)$
$\mathcal{C}(128, 69, 16)$	$\mathcal{C}(128, 31, 32)$	$\mathcal{C}(\mathbf{256}, \mathbf{100}, \mathbf{32})$
$\mathcal{C}(128, 54, 22)$	$\mathcal{C}(128, 19, 44)$	$\mathcal{C}(256, 73, 44)$
$\mathcal{C}(128, 52, 24)$	$\mathcal{C}(128, 17, 48)$	$\mathcal{C}(256, 69, 48)$
$\mathcal{C}(128, 40, 28)$	$\mathcal{C}(128, 11, 56)$	$\mathcal{C}(256, 51, 56)$
$\mathcal{C}(128, 31, 32)$	$\mathcal{C}(128, 8, 64)$	$\mathcal{C}(\mathbf{256}, \mathbf{39}, \mathbf{64})$

The code examples in Tables 8.1 and 8.2 prioritize optimizing the minimum distance. However, we will later see that codes with the same length and rate can be generated with different component codes. This possibility enables exploring other design paradigms, such as concepts known from multilevel codes [56].

In the upcoming section, we will discuss a soft-input decoding method that leverages the cyclic structure of the component codes. Note that shortened BCH codes are also considered cyclic codes [145]. Therefore, the Plotkin construction and decoding methods can also be applied to shortened BCH codes.

8.2 Decoding of Plotkin Concatenated Codes

This section introduces a soft-input decoding algorithm designed for concatenated codes. Inspired by the Schnabl and Bossert approach [122], we exploit the recursive structure of the Plotkin construction. However, we employ HOSD for soft-input decoding at the first level of concatenation, as proposed in Section 7.5. Before delving into the recursive decoding algorithm using HOSD and further improvements, we review the generalized multiple concatenation (GMC) decoding method from [122].

8.2.1 Generalized Multiple Concatenation Decoding

The GMC decoder introduced in [122] can perform soft decoding of RM codes with moderate computational complexity. It exploits the recursive structure of the concatenated codes. At the first recursion level, ML decoding is exclusively applied to the SPC and repetition codes. A simple combination of the soft channel values and the hard output from the component decoders is utilized at the higher recursion levels. We briefly review this decoding procedure, focusing on the soft value combining, as it will be a crucial component of the forthcoming decoding algorithm.

We consider the decoding of the RM code $\mathcal{C}_{\mathcal{R}}(n,k,d)$ with parameters r and m as described in Section 6.2. At the first level, which corresponds to the states $r=0$ or $r=m-1$, ML decoding is applied. A recursive decoding process is conducted for the vector received $\mathbf{y}$ for all other levels. Consider the channel values represented as $\mathbf{y}=[y_0,...,y_{2n-1}]$, where $n=2^{m-1}$ is the length of the component codes. Initially, we process the RM component code $\mathcal{C}^{(2)}$ with $r^{(2)}=r-1$ and $m^{(2)}=m-1$. To decode $\mathcal{C}^{(2)}$, the values received are combined as

$$y_i^{(2)} = \operatorname{sign}\left(y_i \cdot y_{n+i}\right) \min\left(|y_i|,|y_{n+i}|\right) \quad \text{for } i=0,...,n-1. \tag{8.1}$$

Note that this step is the so-called *minsum* approximation of the *boxplus* operation, a concept from iterative decoding algorithms [146]. The minsum operation in (8.1) can be replaced with the boxplus operation using LLRs for more accurate soft values. The values $y_i^{(2)}$ are processed

with a decoder for the component code $\mathcal{C}^{(2)}$, resulting in the estimated codeword

$$\hat{\mathbf{x}}^{(2)} = \left[\hat{x}_0^{(2)}, \ldots, \hat{x}_{n-1}^{(2)}\right] \in \mathcal{C}^{(2)}.$$

Next, this result is used to calculate the soft values for the other component code $\mathcal{C}^{(1)}$ as

$$y_i^{(1)} = \frac{1}{2}\left(y_i + y_{n+i} \cdot \hat{x}_i^{(2)}\right) \quad \text{for } i = 0, ..., n-1 \tag{8.2}$$

where we assume the mapping $\hat{x}_i^{(2)} \in \{1, \text{-}1\}$. This operation is referred to as *soft-combining* in the following. It is important to note that soft-combining is an approximation. The values y_i and y_{n+i} are treated as noisy observations of two independent transmissions of the symbol x_i. In this scenario, $1/2\,(y_i + y_{n+i})$ represents the minimum mean square error (MMSE) estimate of x_i. However, due to the product $y_{n+i} \cdot \hat{x}_i^{(2)}$, the two terms in (8.2) are not independent.

The vector $\mathbf{y}^{(1)} = \left[y_0^{(1)}, ..., y_{n-1}^{(1)}\right]$ can subsequently be decoded with a decoder for the RM code with $r^{(1)} = r$ and $m^{(1)} = m - 1$, resulting in an estimate codeword

$$\hat{\mathbf{x}}^{(1)} = \left[\hat{x}_0^{(1)}, \ldots, \hat{x}_{n-1}^{(1)}\right] \in \mathcal{C}^{(1)}.$$

Finally, a codeword for the code $\mathcal{C}_{\mathcal{R}}(n, k, d)$ is derived by concatenating the hard output results as

$$\hat{\mathbf{x}} = \left[\hat{\mathbf{x}}^{(1)} \mid \hat{\mathbf{x}}^{(1)} \cdot \hat{\mathbf{x}}^{(2)}\right]. \tag{8.3}$$

Here, an element-wise multiplication is performed instead of $\oplus$, as we assume $\hat{x}_i^{(j)} \in \{1, \text{-}1\}$.

8.2.2 Recursive Decoding Algorithm exploiting BCH Component Codes

We leverage the inherent structure of the Plotkin construction to maintain low complexity for longer ECCs. Therefore, the HOSD is only applied to the lowest recursion level, i.e., the BCH component codes. For higher recursion levels, we employ soft value combining.

The proposed soft-input hard-output decoding procedure is described in Algorithm 8.1. Similar to the order of RM codes, we use r to denote the recursion levels. A code with r recursion levels must decode 2^r component codes, each with equal code lengths. The algorithm **recursive-decoding** is executed recursively until $r = 1$ is reached. Similar to the GMC decoding algorithm [122], soft values for the component codewords are computed in each iteration, whereas the HOSD procedure is only applied at level $r = 1$. The final step of each recursion level returns a candidate codeword of the component code. In the following, this recursive structure is referred to as Plotkin decoding (PD).

Algorithm 8.1 Recursive-decoding

input: received vector $\mathbf{y}$, recursion level r
output: estimated codeword $\hat{\mathbf{x}}$

1: $\mathbf{y}^{(2)} \leftarrow \mathrm{minsum}(\mathbf{y})$ // apply (8.1)
2: **if** $r == 1$ **then**
3: $\hat{\mathbf{x}}^{(2)} \leftarrow \mathrm{HOSD}\left(\mathbf{y}^{(2)}\right)$ // decoding of $\mathcal{C}^{(2)}$
4: $\mathbf{y}^{(1)} \leftarrow \text{soft-combining}\left(\mathbf{y}, \hat{\mathbf{x}}^{(2)}\right)$ // apply (8.2)
5: $\hat{\mathbf{x}}^{(1)} \leftarrow \mathrm{HOSD}\left(\mathbf{y}^{(1)}\right)$ // decoding of $\mathcal{C}^{(1)}$
6: $\hat{\mathbf{x}} = \left[\hat{\mathbf{x}}^{(1)} \mid \hat{\mathbf{x}}^{(1)} \cdot \hat{\mathbf{x}}^{(2)}\right]$ // combine components
7: **return** $\hat{\mathbf{x}}$
8: **else**
9: $\hat{\mathbf{x}}^{(2)} \leftarrow \textbf{recursive-decoding}\left(\mathbf{y}^{(2)}, r-1\right)$
10: $\mathbf{y}^{(1)} \leftarrow \text{soft-combining}\left(\mathbf{y}, \hat{\mathbf{x}}^{(2)}\right)$ // apply (8.2)
11: $\hat{\mathbf{x}}^{(1)} \leftarrow \textbf{recursive-decoding}\left(\mathbf{y}^{(1)}, r-1\right)$
12: $\hat{\mathbf{x}} = \left[\hat{\mathbf{x}}^{(1)} \mid \hat{\mathbf{x}}^{(1)} \cdot \hat{\mathbf{x}}^{(2)}\right]$ // combine components
13: **return** $\hat{\mathbf{x}}$
14: **end if**

8.2.3 Recursive List-Decoding Algorithm exploiting BCH Component Codes

Next, we extend the previously proposed recursive decoding algorithm strategy by incorporating list-decoding techniques for the component codes. OSD inherently includes list-decoding. By storing a list of the Υ best candidates for the code $\mathcal{B}^{(2)}$ and employing multiple attempts for the decoding of $\mathcal{B}^{(1)}$, we can enhance the decoding performance of the concatenated code. This list-decoding increases the number of decoding steps. However, the decoding complexity for the component code $\mathcal{B}^{(1)}$ is typically much lower than for $\mathcal{B}^{(2)}$, and hence, a moderate list size Υ will only marginally increase the overall decoding complexity.

Again, we only consider list-decoding for a single recursion level, as described in Algorithm 8.2. We first decode $\mathcal{B}^{(2)}$ given the concatenated structure. LHOSD refers to the hybrid OSD algorithm that returns a list $\mathcal{L} = \hat{\mathbf{x}}_1^{(2)}, \hat{\mathbf{x}}_2^{(2)}, \ldots$ containing at most Υ candidate codewords, which are the codewords with the smallest metric values. For each candidate in the list, we proceed with decoding the code $\mathcal{B}^{(1)}$. Consequently, the decoding process for $\mathcal{B}^{(1)}$ is carried out at most Υ times. To reduce the decoding complexity, the OSD order is incremented with each iteration until either the maximum order is reached or a codeword satisfying the stopping criterion for the concatenated codeword is found. Due to the stopping criterion (7.19), the list may not be completely filled. To increase the number of candidate codewords in the list $\mathcal{L}$, this criterion should be more restrictive than for HOSD decoding. The list size depends on the SNR. In the following, this approach is referred to as list Plotkin decoding (LPD).

Algorithm 8.2 List-decoding algorithm for one recursion level

input: received vector $\mathbf{y}$, list size Υ, stopping parameters M, maximal order $\theta^{(1)}$ for $\mathcal{B}^{(1)}$, maximal order $\theta^{(2)}$ for $\mathcal{B}^{(2)}$
output: estimated codeword $\hat{\mathbf{x}}$

$\theta_1 = 0$; $\theta_2 = 0$;
$\mathbf{y}^{(2)} = \text{minsum}(\mathbf{y})$ // apply (8.1)
repeat
  $\mathcal{L} = \text{LHOSD}\left(\mathbf{y}^{(2)}, \Upsilon, \theta_2, M^{(2)}\right)$ // dec. $\mathcal{B}^{(2)}$
  for i $= 1$ **to** $|\mathcal{L}|$ **do**
    $\mathbf{y}_i^{(1)} = \text{soft-combining}\left(\mathbf{y}, \hat{\mathbf{x}}_i^{(2)}\right)$ // apply (8.2)
    repeat
      $\hat{\mathbf{x}}^{(1)} = \text{HOSD}\left(\mathbf{y}_i^{(1)}, \theta_1, M^{(1)}\right)$ // dec. $\mathcal{B}^{(1)}$
      $\hat{\mathbf{x}} = \left[\hat{\mathbf{x}}_1^{(1)} \mid \hat{\mathbf{x}}_1^{(1)} \cdot \hat{\mathbf{x}}_i^{(2)}\right]$ // combine components
      if $l\left(\hat{\mathbf{x}}, \mathbf{y}\right) \leq l_{th}$ **then**
        break // criterion accepted
      end if
      θ_1 +=1
    until $\left(\theta_1 > \theta^{(1)}\right)$
  end for
  θ_2 +=1
until $\left(\theta_2 > \theta^{(2)}\right)$
return $\hat{\mathbf{x}}$

8.3 Simulation and Complexity Results of Plotkin Codes

This section presents a comprehensive analysis of the PD and LPD decoding methods. The evaluation includes simulation results and exploration of the decoding complexity. We assume BPSK transmission over an AWGN channel.

The performance is measured in terms of FER versus the normalized SNR in E_b/N_0 (energy per bit to noise power spectral density ratio).

The decoding complexity of the OSD method escalates with the decoding order because $\binom{k}{\theta}$ test patterns are applied for the θ-th order. Consequently, this results in a complexity of order $O\left(k^{\theta}\right)$. According to [129], near-ML performance can be achieved performing OSD up to order $\theta = \lceil d/4 \rceil$ for codes of rates $R \geq 1/2$. This implies a complexity of order $O\left(k^{\lceil d/4 \rceil}\right)$. The Box-and-Match technique from [140] reduces the number of test patterns and results in a worst-case complexity of order $O\left(k^{\lceil d/4 \rceil - 1}\right)$. Applying this decoding concept to the component codes of a Plotkin code with a single recursion level results in the worst-case complexity $O\left(k_2^{\lceil d/4 \rceil - 1}\right)$ with $k_2 \leq k/2$ which is determined by the Box-and-Match algorithm decoding of the component code $\mathcal{C}^{(2)}$ with dimension k_2. For many codes, k_2 is much smaller than $k/2$, and a further reduction in decoding order is feasible for component codes with rates $R \leq 1/2$. Therefore, both PD and LPD exhibit a significantly lower worst-case complexity than the Box-and-Match algorithm for the complete code.

However, the average complexity is typically considerably lower than the worst-case complexity due to the stopping criteria. In the best case, PD and LPD demand only $O\left(n \log(n)\right)$ FLOPs for the metric calculation and $O\left(t^2\right)$ finite field operations for the algebraic decoding, where t is the error correction capability of the component code $\mathcal{C}^{(2)}$. Notably, the average complexity of the considered algorithms varies depending on the SNR and is not expressible in closed form.

In this section, we provide complexity models for the algorithms used. To compare all decoding approaches, we evaluate the complexity in terms of FLOPs and binary operations (BOPs). In this context, we do not consider the additional memory requirements resulting from the size of the control band using the Box-and-Match algorithm. Subsequently, we present and discuss numerical results for the average complexity.

8.3.1 Complexity of Generalized Multiple Concatenated Decoding and Polar Decoding

The GMC algorithm primarily involves a minimal number of BOPs, which can be effectively neglected. The complexity analysis focuses on counting the FLOPs involved in each decoding step, considering component codes of length $n = 2^m$. The soft-input decoding of a repetition code with a length of n needs $n-1$ floating point additions. Similarly, the soft-input decoding of the SPC requires $n-1$ floating point comparisons.

Next, we delve into the soft-combining step. Calculating the soft values for the code $\mathcal{C}^{(2)}$ of length n according to (8.1) demands n floating point additions. For the code $\mathcal{C}^{(1)}$, the soft value calculation as described in (8.2) needs $2n$ FLOPs (n additions and n divisions). Notably, the length $n = 2^m$ of the component codes changes at each recursion level, and the total complexity is the summation across all levels.

It is crucial to recognize that the complexity of the GMC decoder closely aligns with that of polar codes employing successive cancellation (SC) decoding. The primary determinant of complexity in both of these algorithms is the number of computed likelihood ratios [147]. Specifically, SC decoding for polar codes and GMC decoding exhibit a computational complexity of $O\left(N\ log_2(N)\right)$, whereas the complexity of successive cancellation list (SCL) decoding with a list size Υ is $O\left(\Upsilon\ N\ log_2(N)\right)$ [148].

8.3.2 Complexity of Recursive Decoding with BCH Component Codes

The total computational complexity of HOSD comprises the combination of algebraic decoding, OSD, and soft value combining. Due to the stopping rule, the average number of operations required depends on the SNR. For instance, the main algorithms of the BCH decoder are only required only when the syndrome check for the vector received indicates errors. Otherwise, the decoding process can terminate if a correct codeword is received. Consequently, OSD with Box-and-Match algorithm is only executed if the algebraic decoding is unsuccessful or the acceptance criterion is not satisfied. Thus, Gaussian elimination is not required for all vectors received. The average number of operations can be estimated through simulation. In the following, we examine the complexity of the individual components, namely the algebraic BCH decoding method and OSD.

Complexity of BCH Decoding

As introduced in Chapter 6, we consider only a binary BCH decoder in this work. This decoder comprises different components, including syndrome calculation, key equation solving, and error location search algorithm [149, 150]. To simplify the complexity calculation, we focus on the most computationally intensive aspects of the BCH decoder. The syndrome calculation is relatively efficient, and thus we will ignore it in further consideration.

The algebraic algorithms require computations over a finite field. Finite field additions (FFAs) can be implemented as bit-wise XOR, whereas the finite field multiplication (FFM) is the most complex mathematical operation among the necessary calculations [151]. Different FFM implementations are analyzed in [151]. It was shown in [20] that the overall complexity of an entire parallel FFM can be approximated by m^2 XOR operations and m^2 AND operations. Therefore, we estimate the total complexity of an FFM as equivalent to $2m^2$ BOPs.

Table 8.3: Complexity as number of BOPs for BCH decoding algorithms with no parallelization factor ([11], ©2022 IEEE).

Complexity	Multiplication	Addition
TSiBMA	$4t(t+1)m^2$	$m(t+1)$
Chien Search	$ntm^2/2$	$n(t-1)m$

Table 8.4: Complexity for OSD with Box-and-Match algorithm ([11], ©2022 IEEE).

Operations	FLOPs	BOPs
sorting	$n\log_2(n)$	
Gaussian elimination		$n \cdot \min(k, n-k)^2$
re-encoding up to order θ	$d \cdot \sum_{i=1}^{\theta} \binom{k}{i}$	$(n-k) \cdot \sum_{i=1}^{\theta} i\binom{k}{i}$
BMA re-encoding in order θ	$d \cdot \sum_{i=1}^{\theta} \binom{k}{i}$	$(n-k) \cdot \sum_{i=1}^{\theta} 2 \cdot i\binom{k}{i}$

Various algorithms can be employed to solve the key equation. The truncated simplified inversionless Berlekamp-Massey algorithm (TSiBMA) from [110] is optimized for binary BCH decoders, offering lower complexity than other solvers. Importantly, it avoids the computational complex finite field inversion. The TSiBMA requires $2t+2$ FFM and $t+1$ FFA in t cycles to compute the error location polynomial.

The Chien search is the commonly used error location search algorithm for BCH decoding. To find all error locations, the standard implementation needs t multiplications and $(t-1)$ additions per code bit [152]. Hence, a fully parallel implementation would require nt multiplications and $n(t-1)$ additions. Several publications have explored methods to reduce the complexity of the Chien search algorithm [152, 153, 154]. For the complexity estimation, we consider results from [153, 20]. Table 8.3 provides the BOPs needed for the algorithms considered for BCH decoding.

Complexity of Ordered Statistics Decoding

The OSD algorithm involves BOPs and FLOPs, with the latter being considerably more complex than the former. Consequently, considering computational complexity, BOPs are often neglected. However, for OSD, the number of BOPs is significant and affects the complexity. An estimation of the total number of computations for OSD up to order θ was provided in [125]. Table 8.4 shows the complexity calculations for the most dominant components of the OSD algorithm, with computations categorized into FLOPs and BOPs.

In [125], the number of operations for the re-encoding is an upper bound, assuming that all parity positions are changed in every re-encoding step. In practice, only d digits are typically altered per re-encoding step. Moreover, only the soft values of the changed positions are required for the metric calculation of the current candidate codeword.

Performance Comparison of Recursive Decoding with BCH Codes

In the following, we discuss some simulation results and compare the complexity of the proposed codes and decoding methods. The analysis assumes transmission using BPSK over an AWGN channel. The performance evaluation is based on the FER plotted against the normalized SNR, represented in E_b/N_0.

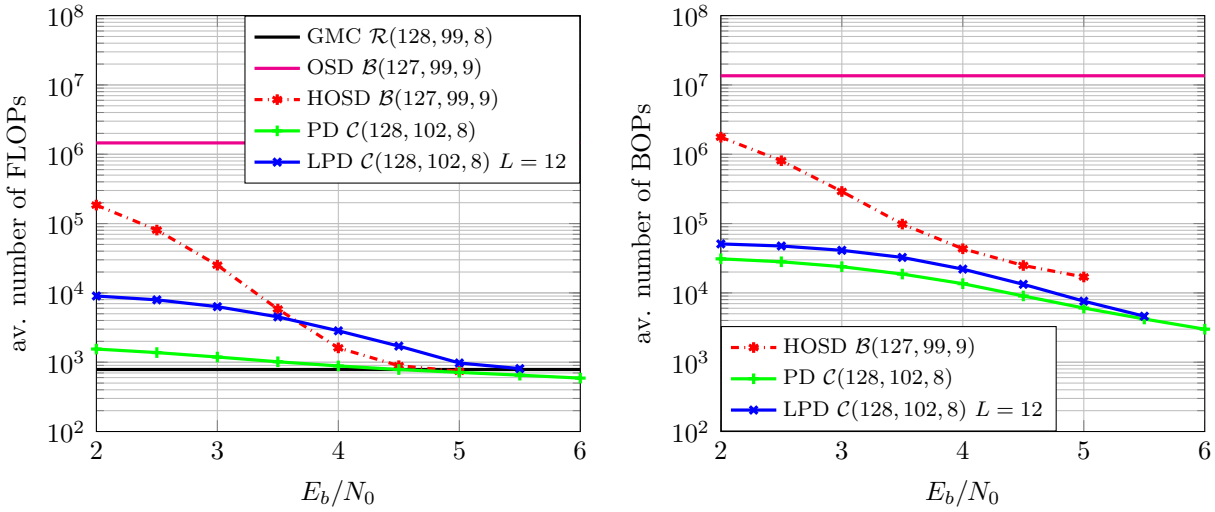


Figure 8.1: Numerical results for the decoding complexity (adapted from [11], ©2022 IEEE).

Figure 8.1 compares the decoding complexity for different decoding methods, including GMC decoding of a RM code $\mathcal{C}_{\mathcal{R}}(128, 99, 8)$, HOSD with maximum order $\theta = 3$ for the BCH code $\mathcal{B}(127, 99, 9)$, the proposed PD decoding for the concatenated code $\mathcal{C}(128, 102, 8)$ from Table 8.1 with one recursion level (with parameters $s = 64$ for $\mathcal{B}^{(1)}$, $s = 50$ for $\mathcal{B}^{(2)}$ max. $\theta = 1$ and $M = d - 1$ for both components), and the LPD decoding for the same code with one recursion level (max. $\theta = 0$, $s = 64$ for $\mathcal{B}^{(1)}$, $\theta = 1$, $s = 47$ for $\mathcal{B}^{(2)}$ and $M = d - 2$ for both components). The decoding complexity is compared in Figure 8.1 in terms of the number of FLOPs and BOPs. The number of BOPs with GMC is negligible and not depicted.

The complexity of complete OSD and GMC decoding remains constant regardless of the SNR. However, in the high SNR region, the HOSD decoding significantly reduces the complexity compared to complete OSD by several magnitudes. As the noise level increases, HOSD tends to approach the complexity of complete OSD.

On the other hand, PD decoding substantially reduces the complexity in the lower SNR region. In the high SNR region, the GMC, HOSD, PD, and LPD decoding algorithms exhibit similar FLOP requirements. Regarding BOPs, the LPD and PD show similar and lower complexity than the HOSD algorithm.

The error correction performance of these codes is shown in Figure 8.2. Complete OSD with $\theta = 3$ for the code $\mathcal{B}(127, 99, 9)$ was not simulated due to its extensive computational complexity. The curves in Figure 8.2 demonstrate that the BCH code with HOSD achieves

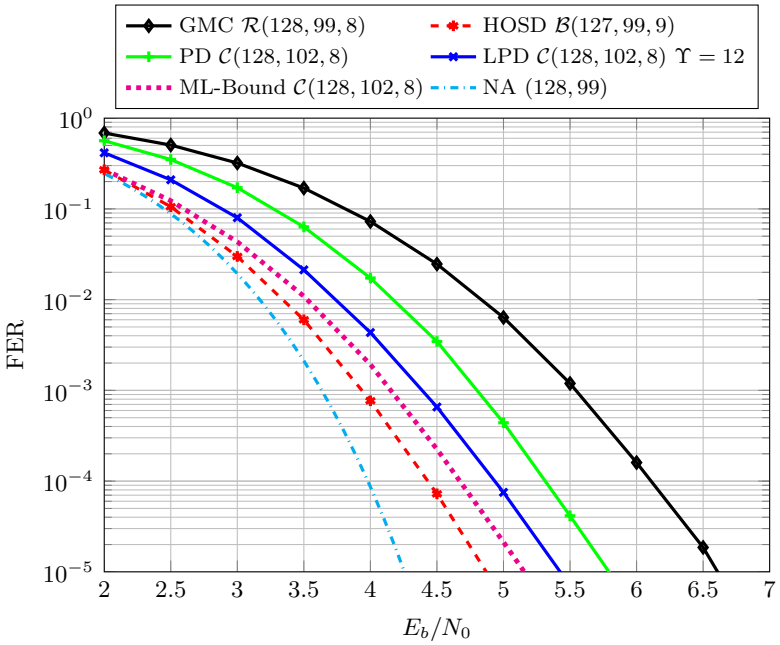


Figure 8.2: Simulation results for the codes $\mathcal{B}(127, 99, 9)$, $\mathcal{C}_{\mathcal{R}}(128, 99, 8)$, and $\mathcal{C}(128, 102, 8)$ with different decoding methods (adapted from [11], ©2022 IEEE).

the best performance. The PD and LPD methods consistently outperform GMC decoding across the entire SNR range. Figure 8.2 also contains a lower bound on the ML performance from [92] for the concatenated code $\mathcal{C}(128, 102, 8)$. LPD achieves a near-ML performance, coming close to the results of the BCH code $\mathcal{B}(127, 99, 9)$ with HOSD. For comparison, we include the normal-approximation (NA), which is an approximation of the random-coding union bound, providing an estimate of the achievable error probability of an $\mathcal{C}(127, 99)$ code [155].

For longer codes and high noise levels, the complexity of HOSD for the complete code becomes prohibitive. By contrast, the complexity of PD and LPD is primarily determined by decoding the component codes. The Plotkin construction demonstrates a more favorable performance-complexity trade-off as the code length increases. The subsequent analysis focuses on concatenated codes.

As previously discussed, it is possible to construct Plotkin codes with similar parameters from different component codes. Figure 8.3 presents different concatenated codes compared to an RM code and the ML bound. All of these codes share a length of $n = 128$ and have similar dimensions.

One highlighted code is $\mathcal{C}(128, 69, 16)$, a concatenated code detailed in Table 8.1, specifically designed to attain the largest minimum distance for a dimension of $k = 69$. Similarly, $\mathcal{C}(128, 69, 12)$ is constructed with one recursion level but employs different component codes, namely $\mathcal{B}^{(1)}(64, 51, 6)$ and $\mathcal{B}^{(2)}(64, 18, 22)$. Despite having a smaller minimum distance compared to $\mathcal{C}(128, 69, 16)$, it performs better within the SNR range considered when decoded with PD. PD achieves near-ML performance for this particular code, rendering list decoding

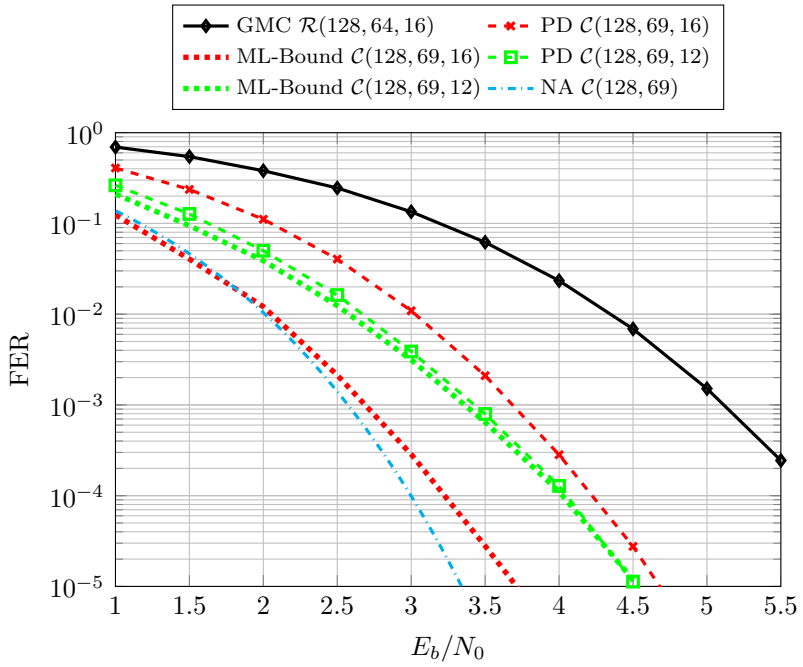


Figure 8.3: Comparison of different concatenated codes with length 128 and similar dimensions (adapted from [11], ©2022 IEEE).

ineffective to attain performance gains. Regarding the $\mathcal{C}(128, 69, 16)$ code, a performance gap of approximately 1dB exists relative to the ML bound when using PD. However, using LPD can further enhance the performance, as demonstrated in Figure 8.4.

In Figure 8.4, we conduct a comparative analysis involving the proposed PD and LPD decoders for codes with $n = 128$ and a code rate of approximately 0.5. These are contrasted with a polar code featuring list decoding with a length of 32 and a CRC of length 6, the BCH-polar code using OSD with order $\theta = 5$ described in [156] and the SC-Flip (SCF) Fano decoder detailed in [157]. For the BCH-polar code, we use the BCH codes $\mathcal{B}(64, 18, 22)$ and $\mathcal{B}(64, 45, 8)$ as outer codes, applying OSD with order $\theta = 5$ to both codes and using the stopping criterion from [125] to reduce complexity.

For the LPD, we employ the parameters for control band $s = 55$ with a maximum order of $\theta = 1$ for $\mathcal{B}^{(1)}$ and $s = 42$ with $\theta = 3$ for $\mathcal{B}^{(2)}$. For the stopping criterion from Equation 7.19, we set $M = d-2$ for both components. On the other hand, for PD we employ the parameters $s = 52$ with $\theta = 1$ for $\mathcal{B}^{(1)}$ and $s = 32$ with $\theta = 2$ for $\mathcal{B}^{(2)}$, while using $M = d - 1$ for the stopping criterion.

The numerical complexity values for polar SC and SCL decoding are calculated according to [147, 148]. The right part of Figure 8.4 displays the computational complexity in terms of FLOPs. The results for polar SCF-Fano decoding are sourced from [157]. This decoding method demonstrates an error correction capability and complexity profile similar to the PD decoder. The list decoders provide substantial performance improvements.

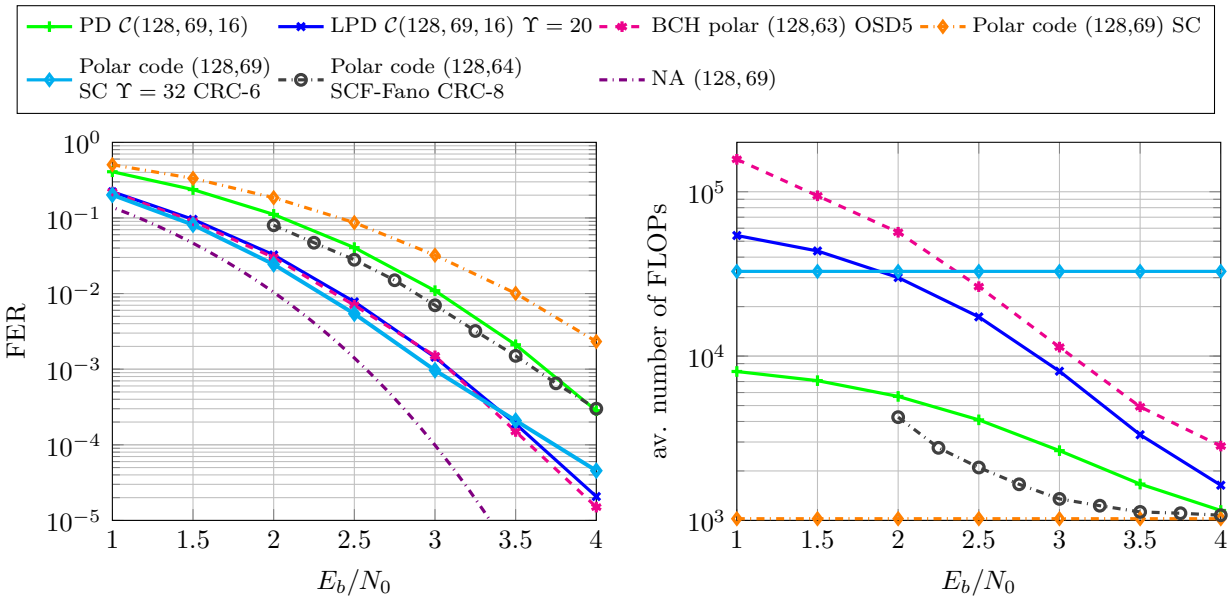


Figure 8.4: Comparison of proposed decoding scheme with list decoding and polar code with rate $R \approx 0.5$ (adapted from [11], ©2022 IEEE).

When evaluating the CRC-polar code, the BCH-polar code from [156], and the proposed Plotkin code $\mathcal{C}(128, 69, 16)$ with LPD, it becomes evident that they exhibit comparable performance. However, LPD stands out as the method with the lowest computation complexity.

Figure 8.5 considers codes of length $n = 256$ and dimension $k = 219$. This analysis compares the proposed coding scheme against polar codes with SCL decoding with list size $\Upsilon = 32$. The SCL decoding of the polar code $(256, 219, 4)$ detailed in [158] is constructed with the improved degrading algorithm and decoded using the SCL decoding algorithm [148, 159]. Additionally, we consider CRC-aided SCL decoding (CA-SCL) with 11 bits.

Turning our attention to the Plotkin construction, we investigate the performance of PD and LPD decoding utilizing HOSD without the Box-and-Match algorithm. The concatenated code $\mathcal{C}(256, 219, 8)$ is configured with one recursion level, employing the corresponding component codes $\mathcal{B}^{(1)}(128, 120, 4)$ and $\mathcal{B}^{(2)}(128, 99, 10)$. For HOSD, we apply $M = d - 1$ for each code. The maximum order for HOSD is set to $\theta = \lceil d/4 \rceil$. The PD decoding attains the error correction capability of the polar code using CA-SCL decoding, while the LPD achieves near-ML performance.

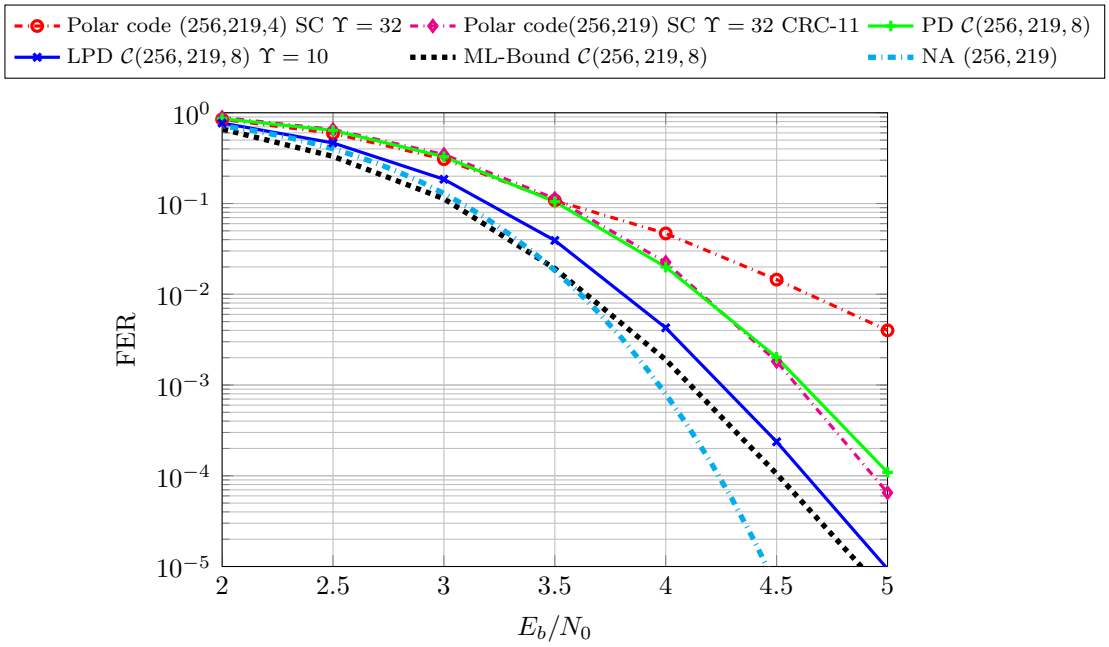


Figure 8.5: Decoding performance of PD and LPD in comparison with polar codes with code length $n = 256$ (adapted from [11], ©2022 IEEE).

8.4 Generalized Concatenated Codes

The decoding performance and complexity of Plotkin concatenated codes are strongly limited by their length and dimension. In applications where long codes with high dimensions are necessary, Plotkin concatenated codes are inappropriate. An alternative is offered by GC codes, which are particularly suitable for long codes with high code rates as for the payload data of flash memory. GC codes represent a robust ECC approach, leveraging the capabilities of multiple component codes. Typically, a GC code comprises outer and inner codes. The flexibility in choosing these component codes enables tailoring the configuration to meet the specific requirements of a given application. Common choices for the component codes of the concatenation include RS, LDPC, BCH, RM, and convolutional codes. Thereby, RS and LDPC frequently serve as outer codes. GC codes employing algebraic codes are particularly well suited for flash memories due to their impressively guaranteed low residual error rates. In this work, our attention is directed towards GC codes that use inner BCH and outer RS codes, as investigated in [89, 136, 54]. This section offers an overview of GC encoding and decoding. Afterwards, we highlight principles of effective code design. For more details, we refer to [20] and [92].

8.4.1 Encoding with GC Codes

In the following, we describe the construction of a GC code composed of outer RS codes and an inner BCH codes. This GC code is constructed from a total of L outer RS codes over the

extension field $GF(2^{m_a})$, in conjunction with L binary inner BCH codes. Each of the outer codes has a length of n_a. We employ the notation $\mathcal{A}^{(l)}(2^{m_a}, n_a, k_a^{(l)}, d_a^{(l)})$ to represent the parameters of the l-th outer code, where $k_a^{(l)}$ indicates the dimension and $d_a^{(l)}$ the minimum Hamming distance. Similarly, we employ $\mathcal{B}^{(l)}(n_b, k_b^{(l)}, d_b^{(l)})$ to denote the l-th inner code. Leveraging extended BCH codes as inner codes is advantageous due to their superior error detection, which improves the erasure-correcting performance of the outer RS codes.

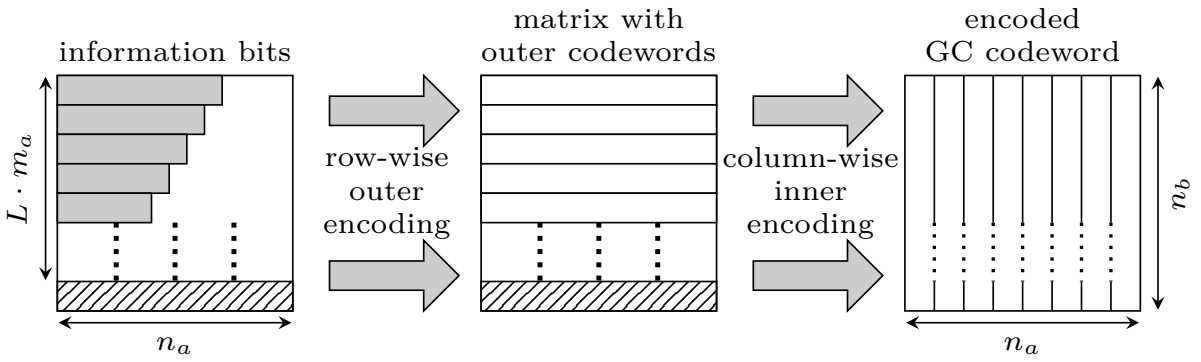


Figure 8.6: Illustration of GC encoding ([2], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/).

Figure 8.6 visually represents the encoding scheme. The concatenated code has a length $N = n_b \cdot n_a$, and each codeword can be represented by an $n_b \times n_a$ binary matrix.

The encoding process starts with the outer codes. In this step, $m_a k_a^{(0)}$ bits are mapped to the first outer codeword, $m_a k_a^{(1)}$ to the second, and so forth, and then encoded by them. This generates a matrix in which each row represents an outer code. Note that the code rate of the outer codes grows from level to level. The outer codes can protect in total Lm_a rows of the matrix. Using every encoding step, the overall dimension becomes

$$K = m_a \sum_{l=0}^{L-1} k_a^{(l)}. \tag{8.4}$$

Following the outer encoding, the columns of the codeword matrix are encoded with the inner BCH codes. Each column of the codeword matrix is the summation of L codewords belonging to nested extended BCH codes of length n_b. This nesting is represented as

$$\mathcal{B}^{L-1} \subset \mathcal{B}^{L-2} \subset \cdots \subset \mathcal{B}^0. \tag{8.5}$$

Simplified, the code $\mathcal{B}^{(0)}$ is partitioned into sub-codes with reduced dimensions but augmented error correction capability. The sub-codes of a BCH code can be generated from the cyclotomic polynomials. Consequently, a higher level code acts as a sub-code of its preceding level, where higher levels generally offer stronger error correction, i.e., $t_{b,L-1} \geq t_{b,L-2} \geq \ldots \geq, t_{b,0}$, where $t_{b,l}$ is the error-correcting capability of level l. The codeword of the j-th column is the sum of

Table 8.5: Parameters of a GC code of length 8188 and rate $R = 0.875$ ([9], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/).

level l	$k_{b,l}$	$d_{b,l}$	$k_{a,l}$	$d_{a,l}$
0	84	4	65	25
1	77	6	81	9
2 - 3	70	8	85	5
4	56	12	87	3
5	49	14	87	3
6-11	42	16	89	1

L codewords

$$\mathbf{b}_j = \sum_{l=0}^{L-1} \mathbf{b}_j^{(l)}. \tag{8.6}$$

These codewords $\mathbf{b}_j^{(l)}$ are constructed by encoding the symbols $a_{j,l}$ with the corresponding sub-code $\mathcal{B}^{(l)}$, where $a_{j,l}$ denotes the j-th symbol (m_a bits) of the outer code $\mathcal{A}^{(l)}$. For this encoding, $(L-l-1)m_a$ zero bits are appended to the symbol $a_{j,l}$.

The dimension of $\mathcal{B}^{(l)}$ is expressed as $k_b^{(l)} = (L-l)m_a$, where $k_b^{(0)} = Lm_a, k_b^{(1)} = (L-1)m_a, \ldots, k_b^{(L-1)} = m_a$. However, only m_a bits are mapped to each codeword $\mathbf{b}_j^{(l)}$. The remaining $n_b - Lm_a$ rows serve as either redundancy for the inner codes or information bits without outer encoding. Note that the j-th column $\mathbf{b}_j$ represents a codeword of $\mathcal{B}^{(0)}$ due to the inherent linearity of the nested codes.

Example 8.2. *In this example, we construct a code to compare it with a polar code [160] in applications related to flash memories. The comparison will be discussed in Section 8.4.2. The polar code is designed for a block length of 1 kilobyte. Similarly, we construct a GC code with a code rate of $R = 0.875$ and a code length of approximately $N = 8192$. This code is designed according to the guidelines proposed in [54] to achieve a FER of 10^{-5} at a channel BER of 0.008. All component codes are constructed over $GF(2^7)$.*

The outer RS codes have a length of $n_a = 89$. The inner codes are binary extended BCH codes with a length of $n_b = 92$. For this GC code, we use $L = 12$ levels, where only the first six levels are protected by outer RS codes. The parameters of the GC code are summarized in Table 8.5, where we use the same RS code from level 6 to level 11. This table provides details about the dimension and minimum Hamming distance of the binary inner codes represented by $k_{b,l}$ and $d_{b,l}$ at level l and the outer RS codes represented by $k_{a,l}$ and $d_{a,l}$ respectively.

Considering Table 8.5, we can calculate the code dimension as $K = m\sum_{l=0}^{L-1} k_{a,j} = 7168$, and the code length of the GC code becomes $N = n_a \cdot n_b = 8188$. This results in the code rate $R = K/N = 0.8754$.

8.4.2 Decoding of GC Codes

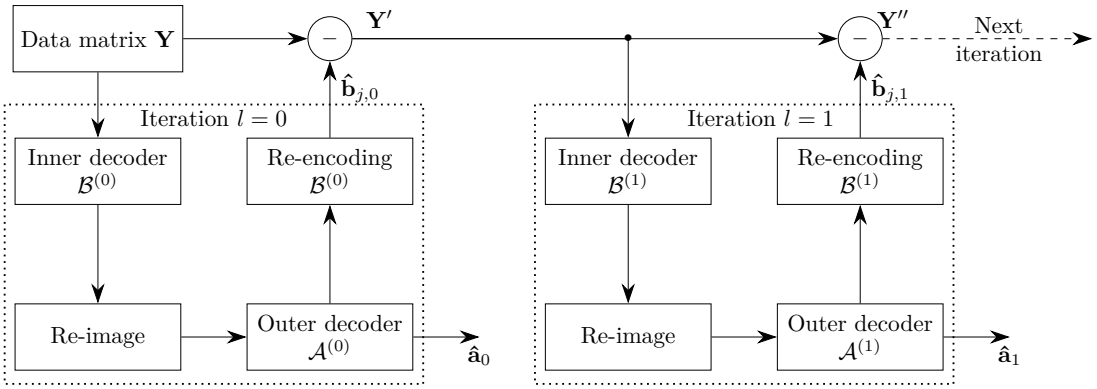


Figure 8.7: GC decoding schemes (adapted from [9], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/).

Decoding a GC code follows a hierarchical structure, where each level undergoes a similar sequence of decoding steps. This iterative decoding process, illustrated in Figure 8.7, begins with the reading of the $n_b \times n_a$ data matrix denoted as $\mathbf{Y}$. At the first level $l = 0$, all columns are decoded concerning the inner BCH code $\mathcal{B}^{(0)}$. This inner decoding operation yields estimates for the outer code symbols $a_{j,0}$ belonging to the outer RS code $\mathcal{A}^{(0)}$, with j denoting the column index.

As we utilize a nested BCH code construction for the inner code, the BCH code $\hat{\mathbf{b}}_{j,0}$ must be re-imaged to obtain the original codeword of the outer code $\mathcal{A}^{(l)}$. This is accomplished by re-encoding the information received $a_{j,i}$ and afterwards subtracting the codeword from the columns with $L > l$. After the successful decoding of all columns with respect to $\mathcal{B}^{(0)}$, the outer RS code $\mathcal{A}^{(0)}$ can be completely decoded. Once this outer decoding is accomplished, an estimated partial decoding result $\hat{\mathbf{a}}_0$ becomes available [20].

However, this result must be re-encoded using the corresponding inner code $\mathcal{B}^{(0)}$ and then subtracted from the matrix received before progressing to the next decoding level.

At the second level $l = 1$, the decoding is performed concerning the inner code $\mathcal{B}^{(1)}$. This code typically has a lower dimension but a larger error-correcting capability compared to the code $\mathcal{B}^{(0)}$. As we proceed to higher levels, the error correction capability of the outer codes can be gradually reduced.

The decoding procedure continues until the last level is reached. When all layer codes have been subtracted from the matrix received, the remaining data matrix contains only the remaining errors. Finally, all outer codes are available, and the total codeword can be reconstructed. Throughout the decoding process, all RS code levels undergo error and erasure decoding throughout the decoding process. For this reason, it is advantageous to employ extended BCH codes, as they provide enhanced failure detection capabilities. This hierarchical decoding approach helps to efficiently correct errors in GC codes [112, 20].

8.4.3 Code Design

Multilevel coding supports a wide range of design rules [56]. To achieve a low residual error rate, we embrace the concept of balanced error probability across the different levels of the GC code. Applying this design rule to GC codes is motivated by the error analysis of the decoder based on the union bound.

.In the context of GC codes, we apply error and erasure decoding on the outer RS codes [92].The defined RS code, represented as $\mathcal{A}^{(l)}(2^{m_a}, n_a, k_a^{(l)}, d_a^{(l)})$ has a minimum Hamming distance

$$d_a^{(l)} = n_a - k_a^{(l)} + 1.$$

With an odd minimum distance, this code is capable of correcting up to

$$t_a^{(l)} = \frac{n_a - k_a^{(l)}}{2}$$

errors and up to $n_a - k_a^{(l)}$ erasures. The probability $P_{a^{(l)}}$ of encountering a decoding failure exploiting error and erasure decoding at the l-th level can be computed as described in [161] as follows

$$P_{a^{(l)}} = \sum_{j=t_a^{(l)}+1}^{n_a} \sum_{i=d_a^{(l)}-2j}^{n_a} \binom{n_a}{j} \binom{n_a - j}{i} \rho_{b^{(l)}}^{j} \lambda_{b^{(l)}}^{i} (1 - \rho_{b^{(l)}} - \lambda_{b^{(l)}})^{n_a - j - i}. \tag{8.7}$$

Here, $\rho_{b^{(l)}}$ represents the error probability for the inner code $\mathcal{B}^{(l)}$, while $\lambda_{b^{(l)}}$ denotes the erasure probability. Using the union bound, the FER P_e of the GC code can be estimated by summing the error probabilities over all levels, i.e.,

$$P_e \leq \sum_{l=0}^{L-1} P_{a^{(l)}}. \tag{8.8}$$

The sum in Equation (8.8) is primarily influenced by the largest probability $P_{a^{(l)}}$. Therefore, the design objective is to balance these error probabilities across all levels to achieve the target channel error rate. These error and erasure probabilities for the inner codes can be estimated using computer simulation or analytical bounds.

Using a cell-wise encoding approach, we estimate the error probabilities for flash memory applications. In this method, we consider the m bits of a flash cell as a code symbol of the inner code. For example, consider a code over a 2^m-ary alphabet with an even minimum distance of $d_b^{(l)}$. The number of symbols with this alphabet becomes $\mathfrak{n}_\mathfrak{b} = n_b/m$. This code is capable of correcting patterns with up to $t_b^{(l)} = \frac{d_b^{(l)}-2}{2}$ errors, and it can detect $t_b^{(l)} + 1$ errors, classifying them as uncorrectable and declaring them as erasures. Given these assumptions,

the erasure probability is bounded by

$$\lambda_{b^{(l)}} \leq \sum_{j=t_b^{(l)}+1}^{\mathfrak{n}_\mathfrak{b}} \binom{\mathfrak{n}_\mathfrak{b}}{j} p_s^j (1-p_s)^{\mathfrak{n}_\mathfrak{b}-j}, \tag{8.9}$$

where p_s denotes the symbol error rate, and $\mathfrak{n}_\mathfrak{b}$ is the code length in 2^m-ary symbols. Similarly, the error probability is bounded by

$$\rho_{b^{(l)}} \leq \sum_{j=t_b^{(l)}+2}^{\mathfrak{n}_\mathfrak{b}} \binom{\mathfrak{n}_\mathfrak{b}}{j} p_s^j (1-p_s)^{\mathfrak{n}_\mathfrak{b}-j}. \tag{8.10}$$

The bounds derived from these equations are not rigorous, but they remain useful for estimating error probabilities in code design and explaining the gain of cell-wise encoding.

8.5 Complexity Reduction of GC Decoder

To ensure the high-speed processing demanded by flash memory, installing the ECC as hardware in a flash controller is essential. In the following, we explore modifications to the decoder described in [136] to minimize complexity. We apply the acceptance criterion outlined in Section 7.4.2 as a decoding stopping criterion after each decoding iteration to reduce computational complexity. Moreover, our objective is to minimize the memory footprint for storing LLR values. As these memory components are integrated into the hardware, this translates into a reduction in hardware requirements.

The decoder architectures in Figure 8.8 illustrate the modifications. The first structure corresponds to the decoder presented in [136], while the second represents the proposed decoder architecture.

Both decoder architectures begin with a mapping function that converts the quantized input value to LLR values based on the estimated channel error probability. These LLR values need to be stored for soft decoding to identify the least reliable positions for a bit-flipping algorithm. We leverage this sorting to minimize the number of soft values.

In [136], LLR values for all code symbols and index values indicating the positions of the least reliable symbols are stored. These positions are required to determine the bit-flipping patterns for the soft-input decoder, as noted in the *pattern generation* block in Figure 8.8. The corresponding LLR values are necessary for metric calculation, which computes the sum of the magnitudes of LLR values corresponding to all altered positions according to Equation (7.7). The positions altered by the bit-flipping patterns can be determined by sorting the soft values. However, the altered positions by the algebraic decoder are only known after each algebraic decoding step. Hence, all LLR values need to be stored in [136].

To reduce random-access memory (RAM) requirements of the controller, we store only the

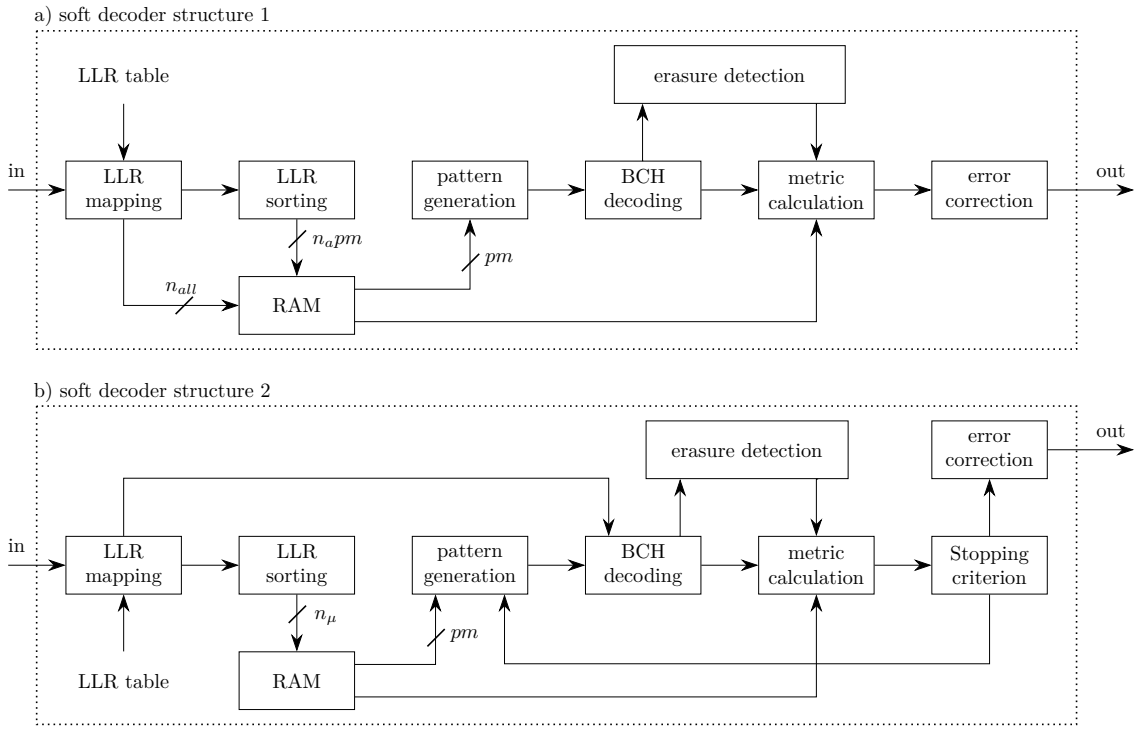


Figure 8.8: Two soft decoder structures (adapted from [9], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/).

positions and LLR values corresponding to the least reliable values while increasing the search depth for the LLR values. This augmentation increases the size of the logic for sorting but enables compression of the soft values. This concept is based on the fact that the algebraic decoder will more likely alter symbols with low reliability. Moreover, quantized input and LLR values reduce the potential variations in LLR values.

The initial BCH codes in the GC code construction exhibit a notably high code rate, and only these specific codes are utilized for soft decoding. Given the coarse resolution of the soft values (see Section 5.1), most input values are transformed into LLR values with maximum magnitudes. Only a small fraction of input symbols will have a small magnitude. In [136], the search depth of the least reliable positions is determined by the maximum number of flipped bit positions p for Chase decoding. The proposed architecture only stores only the LLR values and their positions for the $\mu > p$ least reliable symbols. We can calculate the correct metric value if the algebraic decoder alters symbols within the stored positions. The LLR values of the not stored positions are set to the maximum possible magnitude. With a sufficiently large value of μ, this modification typically results in the correct metric according to Equation (7.7). However, the metric calculation and codeword selection are occasionally affected, leading to a minor performance loss, as we will see in the following.

Example 8.3. *To illustrate the memory size reduction, we consider the BCH codes with a length of $n_b = 92$, as presented in Example 8.2. For these codes, good simulation results*

can be achieved by retaining only the $\mu = 10$ smallest LLR values, assuming quantized soft information with six discrete sensing levels as described in [160]. The quantized input values are converted into LLR values, each LLR represented by $n_{LLR} = 5$ bits: one bit for the HD and four bits indicating the magnitude. To store all soft values without further compression, we calculate

$$\begin{aligned} n_{all} &= n_a \cdot n_b \cdot (n_{LLR} - 1) \\ &= 8188 \cdot 4 = 32\,752 \end{aligned}$$

total number of bits to be stored. Additionally, $n_a \cdot m \cdot p = 1\,869$ bits are required to store symbol positions.

With the proposed approach, we store m bits for each position and $(n_{LLR} - 1)$ bits for the corresponding magnitude. This results in a total of

$$\begin{aligned} n_\mu &= \mu \cdot n_a \cdot (m + (n_{LLR} - 1)) \\ n_\mu &= 10 \cdot 89 \cdot (7 + 4) = 9790 \textit{bits} \end{aligned}$$

In this example, the memory required for storing soft values can be reduced by 72%.

The significant reduction in memory resources shown in the example can substantially affect hardware implementation, especially in resource-constrained environments like flash memory applications.

Moving forward, we present simulation results for the GC code assuming an AWGN communication channel and quantized with six sensing levels, following the approach from [160]. First, we evaluate the performance of the BCH codes according to Example 8.2. In Figure 8.9, the

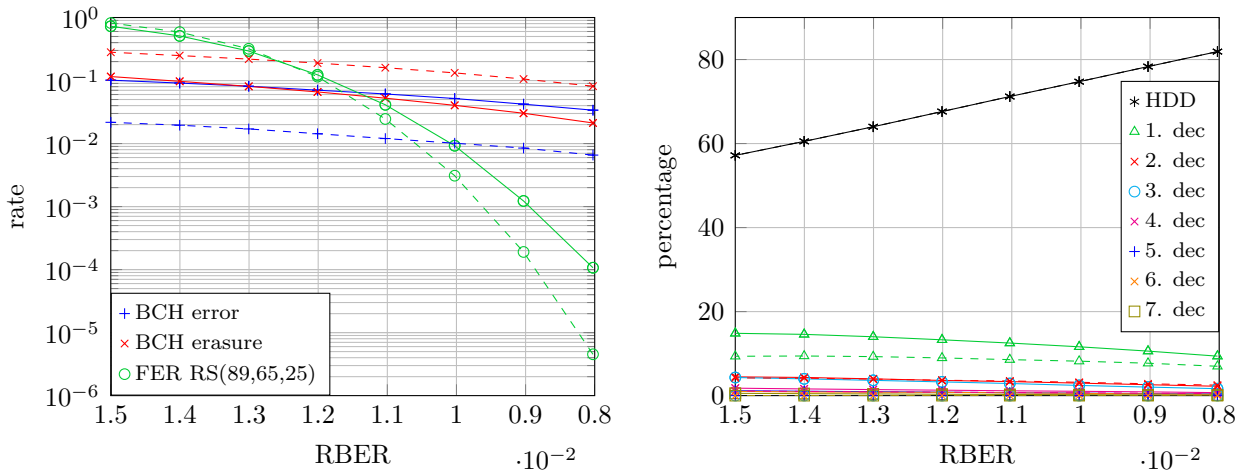


Figure 8.9: Comparison of $\mathcal{B}(92, 84, 4)$ soft decoding using (7.12) (solid line) and the proposed threshold (7.18) (dashed line) as stopping criterion (adapted from [9], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/).

left-hand side displays different error rates for the codes of the first GC level $l = 0$. For the inner BCH code (92,84,4), we illustrate the FER and the erasure rate, i.e., detected decoding failure. We distinguish between two decoding methods: one using Kaneko's stopping rule, represented by solid lines, and the other employing the proposed acceptance criterion from Section 7.4.2, denoted by dashed lines. Additionally, we present FERs for the outer RS code leveraging error and erasure decoding. Applying Kaneko's rule in this context leads to comparable error and erasure probabilities for the BCH code. However, the RS code is capable of correcting more erasures than errors. Hence, it is possible to enhance RS decoding performance by slightly increasing the error rate at the expense of a higher failure (error detection) rate. This improvement in RS code performance is evident in the FER.

Note that the introduction of the new stopping rule has a minimal impact on decoding complexity, as demonstrated on the right-hand side of Figure 8.9. This section of the figure showcases the percentage of decoding iterations that are terminated after the specified iteration.

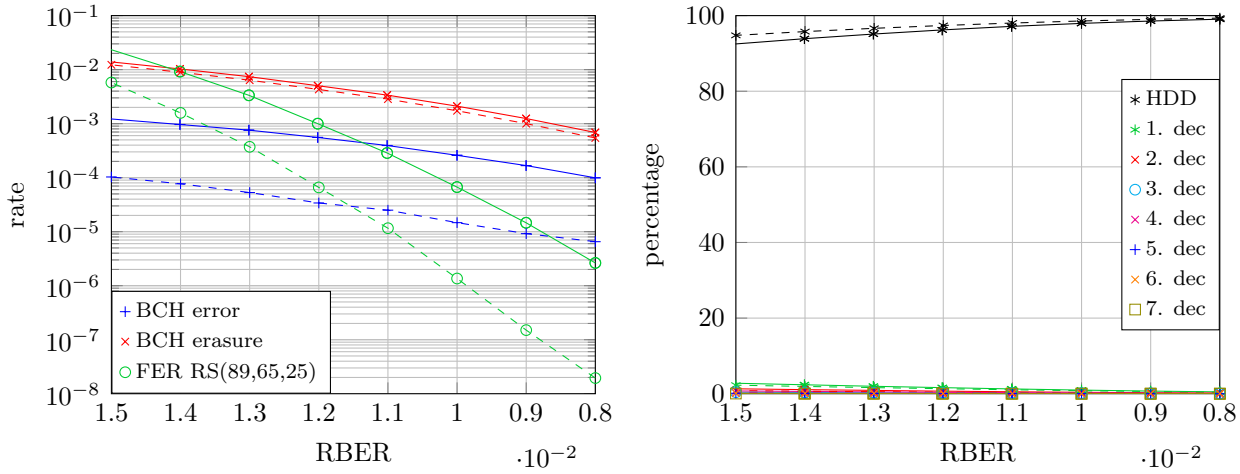


Figure 8.10: Comparison of $\mathcal{B}(92, 70, 8)$ soft decoding using (7.12) (solid line) and the proposed threshold (7.18) (dashed line) as stopping criterion (adapted from [9], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/).

We present similar results for the third decoding level ($l = 2$), as illustrated in Figure 8.10. In this scenario, the performance of the RS code exhibits a significant enhancement of the error correction capability. Thanks to the robust error correction capabilities of the inner code at this level, a remarkable 95% to 99% of all received codewords can be decoded effectively using HD decoding. As we ascend through the hierarchy of levels, with increasing error correction capabilities, the likelihood of employing soft decoding progressively diminishes. Consequently, we omit soft-input decoding for levels beyond $l > 2$. Figure 8.11 provides the overall performance of the GC code using various decoding methods. Specifically, we employ two sets of trade-off parameters as summarized in Table 8.6. The first set is dedicated to decoding without premature stopping. In this scenario, we execute all 2^p iterations of the bit-flipping decoder, and the criterion specified in Equation (7.18) serves as the acceptance criterion for the final candidate codeword, as proposed in [54].

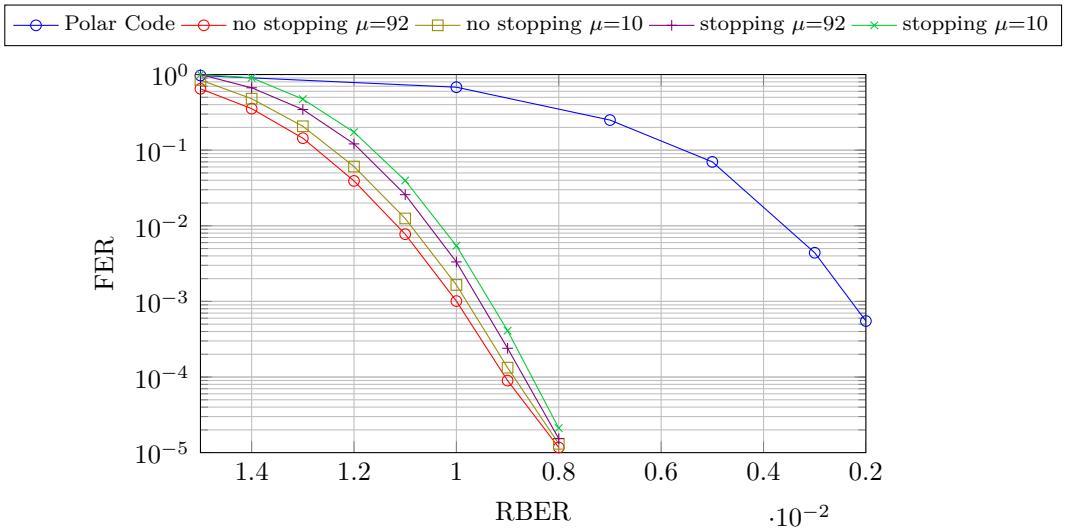


Figure 8.11: Word error rate for different decoding methods for the GC code ($N = 8188$ and $K = 7168$) in comparison with a polar code ($n = 8192$ and $k = 7168$) from [160] (adapted from [9], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/).

Table 8.6: Parameters for acceptance criterion ([9], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/).

level	without stopping	with stopping
0	$M = 4$; $T = 2$	$M = 3$; $T = 1$
1	$M = 6$; $T = 2$	$M = 5$; $T = 0$
2	$M = 8$; $T = 2$	$M = 7$; $T = 1$

On the other hand, the second set of parameters is designed for decoding with premature stopping. These parameter values have been determined through simulation, focused on optimizing the performance of the outer RS code, specifically concerning error and erasure decoding.

We conducted simulations using two distinct storage schemes for the LLR values. The curves labeled as $\mu = 92$ imply that all LLR values were retained, while $\mu = 10$ corresponds to results obtained with limited memory capacity.

As expected, the setup without premature stopping and unrestricted memory exhibits the most favorable performance. However, differences only become noticeable at increased channel BERs. At the target error rate of 0.008, there was only a slight performance loss when employing the proposed decoding methods.

To provide a point of reference, we present results for a polar code with similar parameters and employing soft-input decoding, as reported in [160]. Notably, the GC code significantly outperforms this polar code.

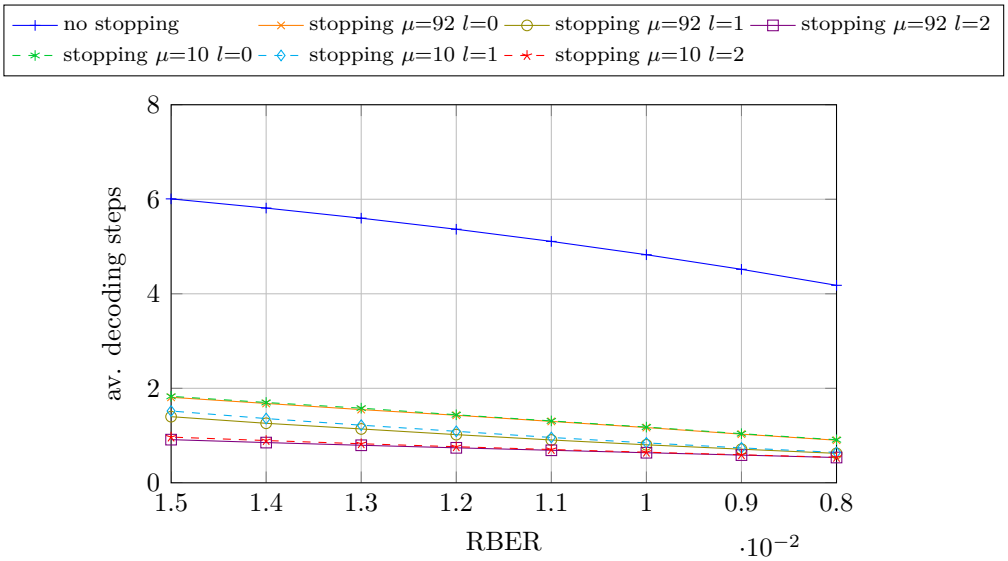


Figure 8.12: Average number of decoding steps for the inner BCH code with and without stopping (adapted from [9], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/).

Finally, we delve into the complexity reduction achieved by our stopping criterion compared to the decoding method proposed in [136]. Figure 8.12 displays the simulation results illustrating this comparison. This figure presents the average number of decoding steps for the inner BCH code, both with and without the stopping criterion, as well as considering memory restrictions, across the initial three GC levels. Note that using the decoding approach from [136], all levels exhibit identical complexity.

With our proposed stopping rule, we observe a significant reduction in the number of decoding iterations, especially for the first level, where it is reduced by a factor of three to four. For the higher levels, the complexity is diminished by a factor ranging from four to six, depending on the channel error rate. Interestingly, the reduction in memory hardly affects the number of iterations. Moreover, the proposed soft-input decoder nearly achieves the computational complexity performance of the hard-input decoder reported in [136]. This translates to an overall soft-input decoding process that is approximately twice as fast as the decoding process presented in [136].

8.6 Summary

In conclusion, this chapter has presented two code concatenation approaches, each suited for various use cases, in our case as ECC for flash memory applications. The primary objective has been to generate longer codes by concatenating shorter ones to mitigate decoding complexity. Note that achieving the same residual error probability with concatenated codes as with non-concatenated codes with the same parameter is a non-trivial task.

First, we delved into the Plotkin construction using BCH component codes. We demonstrated

that binary RM codes can be considered as a special instance of this construction. Nonetheless, employing the construction with BCH components results in ECCs with notably enhanced code parameters when compared to binary RM codes. The GMC soft decoding method, as introduced in [122], was extended to such codes by incorporating HOSD for the component codes. This hybrid decoding approach exploits the recursive code structure to effectively reduce the decoding complexity associated with OSD. The analysis provided valuable insights into the computational efficiency of different decoding methods, highlighting the advantages of the proposed PD and LPD algorithms in terms of complexity reduction, particularly in low SNR scenarios. For higher SNR values, the Plotkin codes demonstrate similar performance to polar codes with SCL decoding while maintaining decoding complexity levels comparable to low-complexity GMC decoding or SC decoding.

Second, we delved into the GC code approach, featuring inner BCH and outer RS codes. This type of ECC exhibits robust capabilities but also offers potential for some enhancements. We have shown that decoding can be accelerated and memory size reduced with minimal impact on overall decoding performance. Further reductions in memory size could be achieved by compressing soft values using probability-based compression techniques. The concepts proposed here are applicable for other concatenated codes, such as product codes, half-product codes, and staircase codes.

The upcoming chapter will employ GC codes for cell-wise code construction designed for flash memory applications.

Chapter 9

Error Correction using Cell-Wise Processing

This chapter presents methods for the joint processing of complete cell data of flash memory. In the case of a joint readout, also referred to as cell-wise readout, code bits from all pages associated with a flash cell must be read together. Depending on the flash technology, this reading procedure may lead to increased read latency. However, it also allows us to leverage the dependencies between different bits stored in a cell, enabling alternative decoding techniques. For instance, various approaches have been introduced to enhance the decoding performance of LDPC codes for cell-wise reads and joint decoding in MLC NAND flash. In [162], it is demonstrated that joint decoding of the two MLC pages improves achievable capacity compared to page-wise decoding.

Furthermore, a progressive read method is considered to acquire additional soft data. Reading the entire cell data and conducting joint decoding improves decoding performance. In [163], a paired page readout technique is presented for MLC flash. However, the mentioned methods require soft-input decoding, which increases computational complexity and consumes more power than hard-input decoding. For SD decoding, reliability models are crucial in error correction by describing the comprehensive factors that influence data accuracy, such as the prevailing noise level and channel conditions. A widely recognized metric used for this purpose is the LLR, which captures the probabilistic information associated with the bits received, affording a refined evaluation of their reliability. By leveraging such reliability models, soft decoding algorithms are able to detect and correct errors more effectively, ensuring robust performance for diverse noise levels and channel environments.

By contrast, we explore hard-input decoding approaches to avoid the additional read operations to gather reliability information. Chapter 4 analyzes the MI of TLC flash and demonstrates that page-wise read operations result in a performance loss regarding achievable cell capacity. To illustrate the practical advantages of cell-wise encoding over page-wise processing, we first introduce a coding approach based on GC codes. This coding approach improves the average error correction performance without increasing the number of readouts. The proposed GC codes exploit the observation that errors in cell-wise reads typically result in a single erroneous bit per cell. This is primarily the case because Gray Codes are commonly used as bit-labeling,

and almost all errors are caused by the incorrect detection of a neighboring charge state.

Afterwards, we explore an approach involving an LDPC code with hard-input considering the joint calculation of the binary combinations based on the selected bit mapping for each charge state. We proceed to calculate the LLRs for each bit, corresponding to the bit mapping of the charge states and the read information. Our research focuses on comparing page-wise and cell-wise processing. Depending on the flash technology and information mapping, page-wise processing may lead to one or more pages with an increased error rate. These pages significantly influence the average error probability and require soft readouts in the early flash memory life-cycle stage. We will show that joint processing enhances the error correction performance for TLC flash memory. Parts of this chapter have been published in [1, 2, 4].

9.1 GC Codes for Cell-Wise Processing

The code design is based on the methods elucidated in Section 8.4.3. In the course of this section, we use Gray labeling as a bit mapping scheme for flash memory. This characteristic results in a particularly low probability of non-adjacent errors. Considering each bit combination of this mapping to a symbol, most symbol errors yield an error pattern where only a single bit among the m bits of a cell is erroneous. We employ binary inner codes for the GC code, interpreting these codes over a 2^m-ary alphabet, where the number of symbols is $\mathfrak{n}_\mathfrak{b} = n_b/m$. In this scenario, the bounds presented in Equation (8.9) and (8.10) may not be rigorous, but they still serve as valuable tools for estimating the error probabilities for the code design and elucidate the advantages of the cell-wise encoding.

We consider exemplary in the following the LSB page for Gray code 2 from Table 4.1, to estimate the page-wise error probability. Under the assumptions from Section 4.1.2, the BER for the LSB page (page no. 3), denoted as $p_{b,LSB}$, is approximately equivalent to the error rate p_e, and consequently the symbol error rate becomes $p_s \approx 1.75 p_e$. Designing a GC code for page-wise encoding, we can leverage Equations (8.9) and (8.10), although we need to substitute the code length $\mathfrak{n}_\mathfrak{b}$ with the length of the binary code n_b, and replace p_s with the BER. This substitution leads to significantly higher error probabilities for the inner code with page-wise encoding. For instance, consider the binary code $\mathcal{B}(120, 112, 4)$ for $p_{b,\mathsf{LSB}} = 0.01$. With cell-wise encoding, we approximate $\lambda_b \approx 0.13$ and $\rho_b \approx 0.025$ according to Equations (8.9) and (8.10), whereas with page-wise encoding, we obtain $\lambda_b \approx 0.29$ and $\rho_b \approx 0.095$.

In this section, we construct ECCs with different sizes and rates tailored to meet the requirements of flash memories. The parameters of the GC codes are designed to ensure a specified FER at specific channel conditions. This involves optimizing the code parameters to balance among the error rates outlined in Equation (8.8), based on the bounds in Equations (8.7), (8.9), and (8.10). Typically, the goal is to achieve a specific FER at a defined channel error rate: the better the ECC used, the higher the admissible channel error rate. At this juncture, we choose GC codes because they enable us to guarantee an exceptionally low residual error

Table 9.1: Exemplary GC code ($N = 36\,414$, $K = 32\,768$, $R \approx 0.9$) for TLC (designed for FER 10^{-15} with an average BER of $3.5{\cdot}10^{-3}$) ([2], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/).

level	inner code	outer code
1	$\mathcal{B}(153, 144, 4)$	$\mathcal{A}(2^8, 238, 148, 91)$
2	$\mathcal{B}(153, 136, 6)$	$\mathcal{A}(2^8, 238, 202, 37)$
3	$\mathcal{B}(153, 128, 8)$	$\mathcal{A}(2^8, 238, 220, 19)$
4	$\mathcal{B}(153, 120, 10)$	$\mathcal{A}(2^8, 238, 226, 13)$
5	$\mathcal{B}(153, 112, 12)$	$\mathcal{A}(2^8, 238, 230, 9)$
6	$\mathcal{B}(153, 104, 14)$	$\mathcal{A}(2^8, 238, 232, 7)$
7	$\mathcal{B}(153, 96, 16)$	$\mathcal{A}(2^8, 238, 234, 5)$
8	$\mathcal{B}(153, 88, 18)$	$\mathcal{A}(2^8, 238, 234, 5)$
9	$\mathcal{B}(149, 80, 20)$	$\mathcal{A}(2^8, 238, 236, 3)$
10	$\mathcal{B}(149, 72, 22)$	$\mathcal{A}(2^8, 238, 236, 3)$
11	$\mathcal{B}(149, 64, 24)$	$\mathcal{A}(2^8, 238, 236, 3)$
12	$\mathcal{B}(149, 56, 26)$	$\mathcal{A}(2^8, 238, 236, 3)$
13	$\mathcal{B}(149, 48, 28)$	$\mathcal{A}(2^8, 238, 236, 3)$
14-18	$\mathcal{B}(149, 40, 30)$	$\mathcal{A}(2^8, 238, 238, 1)$

rate. The codes are developed to achieve the desired code rate with the predefined block length, while the target residual FER is conducted at the highest possible channel error rate. For comparison, the same code is employed for joint decoding and page-wise decoding.

Regarding the bit mapping, the length of the inner code must be divisible by the number m of bits per cell, which equals 3 for TLC and 4 for QLC. We employ extended BCH codes for the inner codes. Consequently, all inner codes have even minimum Hamming distance. The additional parity bit improves the decoding failures detection. In the case of a detected decoding failure, an erasure is declared, which enhances the decoding capability of the outer RS decoder. Our analysis focuses on target error rates of 10^{-15} for TLC and 10^{-16} for QLC to meet stringent reliability standards. Error probabilities of this magnitude are required for distinct flash memory applications [14, 19, 164, 165].

First, we consider a code designed for TLC flash memory with code rate $R \approx 0.9$ for a payload data size of 4kB. The detailed code parameters are shown in Table 9.1. The second code example considers a code rate $R \approx 0.87$ for 2kB payload data. The corresponding code parameters are presented in Table 9.2. Additionally, Table 9.3 details a code designed for QLC flash memory, optimized for 4kB payload data and rate $R \approx 0.9$. Finally, Table 9.4 features a code with 4kB user data with $R \approx 0.85$, which was again specially created for QLC flash memory. The parameters of these codes were selected to align with the ECCs designed for flash memory, as specified in references [87] and [121]. In the header of the code tables 9.1-9.4, the channel error rate at which the target FER is achieved is explicitly stated as the average BER. The next part of this section is dedicated to performance comparisons of these codes regarding page-wise and cell-wise processing.

Table 9.2: Exemplary GC code ($N = 18\,810$, $K = 16\,386$, $R \approx 0.87$) for TLC (designed for FER 10^{-15} with an average BER of 4.2^{-3}) ([2], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/).

level	inner code	outer code
1	$\mathcal{B}(114, 106, 4)$	$\mathcal{A}(2^8, 165, 101, 65)$
2	$\mathcal{B}(113, 98, 6)$	$\mathcal{A}(2^8, 165, 137, 29)$
3	$\mathcal{B}(112, 90, 8)$	$\mathcal{A}(2^8, 165, 151, 17)$
4	$\mathcal{B}(111, 82, 10)$	$\mathcal{A}(2^8, 165, 155, 11)$
5	$\mathcal{B}(110, 74, 12)$	$\mathcal{A}(2^8, 165, 157, 9)$
6	$\mathcal{B}(109, 66, 14)$	$\mathcal{A}(2^8, 165, 159, 7)$
7	$\mathcal{B}(108, 58, 16)$	$\mathcal{A}(2^8, 165, 161, 5)$
8	$\mathcal{B}(114, 50, 22)$	$\mathcal{A}(2^8, 165, 163, 3)$
9	$\mathcal{B}(113, 42, 24)$	$\mathcal{A}(2^8, 165, 163, 3)$
10 - 13	$\mathcal{B}(112, 34, 28)$	$\mathcal{A}(2^8, 165, 165, 1)$

Table 9.3: Exemplary GC code ($N = 36\,300$, $K = 32\,783$, $R \approx 0.9$) for QLC (designed for FER 10^{-16} with an average BER of 3.6^{-3}) ([2], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/).

level	inner code	outer code
1	$\mathcal{B}(220, 211, 4)$	$\mathcal{A}(2^8, 165, 51, 15)$
2	$\mathcal{B}(220, 203, 6)$	$\mathcal{A}(2^8, 165, 113, 53)$
3	$\mathcal{B}(220, 195, 8)$	$\mathcal{A}(2^8, 165, 139, 27)$
4	$\mathcal{B}(220, 187, 10)$	$\mathcal{A}(2^8, 165, 149, 17)$
5	$\mathcal{B}(220, 179, 12)$	$\mathcal{A}(2^8, 165, 155, 11)$
6	$\mathcal{B}(220, 171, 14)$	$\mathcal{A}(2^8, 165, 157, 9)$
7	$\mathcal{B}(220, 163, 16)$	$\mathcal{A}(2^8, 165, 159, 7)$
8	$\mathcal{B}(220, 155, 18)$	$\mathcal{A}(2^8, 165, 161, 5)$
9	$\mathcal{B}(216, 147, 20)$	$\mathcal{A}(2^8, 165, 161, 5)$
10	$\mathcal{B}(216, 139, 22)$	$\mathcal{A}(2^8, 165, 161, 5)$
11 - 15	$\mathcal{B}(216, 131, 24)$	$\mathcal{A}(2^8, 165, 163, 3)$
16-26	$\mathcal{B}(216, 91, 38)$	$\mathcal{A}(2^8, 165, 165, 1)$

Table 9.4: Exemplary GC code ($N = 38\,500$, $K = 32\,773$, $R \approx 0.85$) for QLC (designed for FER 10^{-16} with an average BER of 7^{-3}) ([2], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/).

level	inner code	outer code
2	$\mathcal{B}(220, 203, 6)$	$\mathcal{A}(2^8, 175, 41, 135)$
3	$\mathcal{B}(220, 195, 8)$	$\mathcal{A}(2^8, 175, 101, 75)$
4	$\mathcal{B}(220, 187, 10)$	$\mathcal{A}(2^8, 175, 135, 41)$
5	$\mathcal{B}(220, 179, 12)$	$\mathcal{A}(2^8, 175, 151, 25)$
6	$\mathcal{B}(220, 171, 14)$	$\mathcal{A}(2^8, 175, 159, 17)$
7	$\mathcal{B}(220, 163, 16)$	$\mathcal{A}(2^8, 175, 163, 13)$
8	$\mathcal{B}(220, 155, 18)$	$\mathcal{A}(2^8, 175, 167, 9)$
9	$\mathcal{B}(216, 147, 20)$	$\mathcal{A}(2^8, 175, 167, 9)$
10	$\mathcal{B}(216, 139, 22)$	$\mathcal{A}(2^8, 175, 169, 7)$
11	$\mathcal{B}(216, 131, 24)$	$\mathcal{A}(2^8, 175, 171, 5)$
12	$\mathcal{B}(216, 123, 26)$	$\mathcal{A}(2^8, 175, 171, 5)$
13	$\mathcal{B}(216, 115, 28)$	$\mathcal{A}(2^8, 175, 171, 5)$
14 - 18	$\mathcal{B}(216, 107, 30)$	$\mathcal{A}(2^8, 175, 173, 3)$
19 - 26	$\mathcal{B}(216, 67, 46)$	$\mathcal{A}(2^8, 175, 173, 3)$

9.2 Performance using GC Codes

Next, we present numerical results for the previously introduced GC codes using hard-input decoding. Each page of the flash cell experiences a different channel condition depending on the selected bit-labeling. Typically, we evaluate the performance of an ECC for flash memories in terms of FER over the RBER. However, due to the page-wise readout mechanism, different pages may experience varying RBER under the same channel conditions. Hence, we depict the FER results against the averaged RBER over all pages.

Regarding the TLC flash memory, we consider Gray code 1 and Gray code 2 from Table 4.1. Remember that among the presented TLC codes, Gray code 1 exhibits the most balanced error distribution across the pages, and it is considered the optimal marking for page-wise reading based on the channel capacities.

We compare joint processing with the page-wise processing for the different pages. In the latter case, the bits are interleaved across all pages. In contrast to [163], we employ hard-input decoding. While soft-input decoding would probably achieve higher gains, it would also require increased computational complexity and additional channel estimation to determine the LLRs for the different bits.

As described in Section 3.3, the channel model is based on measurements from a specific NAND flash memory manufacturer. To analyze the codes for various flash types and different code rates, we compute error probabilities using a unipolar ASK modulation with AWGN (cf. Section 3.1). This channel model approximates an ideal flash channel where all states have the same noise variance. Nevertheless, the different pages exhibit varying error rates and channel

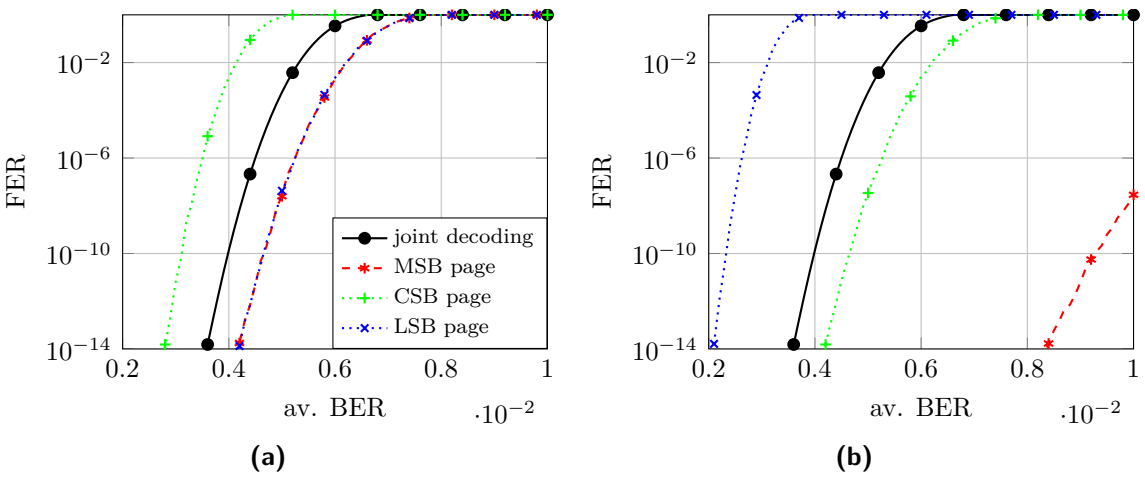


Figure 9.1: GC codes of rate $R = 0.9$ and 4kB payload data for a TLC model using **(a)** Gray code 1 and **(b)** Gray code 2 (adapted from [2], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/).

capacities due to the bit-labeling. The presented numerical results are computed using the formulas provided in Section 4.1.2. In our analysis, we employ the same GC code for the page-wise and cell-wise processing.

The channel analysis procedure involves randomly generating binary symbols, encoding them, and converting them into an m-ary symbol. Subsequently, an average error probability is calculated, and the corresponding variance is determined according to the channel model. This variance is applied to the m-ary symbol, which is then converted into a binary vector using a hard readout. Finally, this binary vector is decoded using the corresponding GC decoder.

Figure 9.1 presents the results for the GC code detailed in Table 9.1 with a rate of $R = 0.9$, employing Gray codes 1 and 2. The curves labeled *joint decoding* represent a scenario with cell-by-cell coding in which all three pages are used equally. Consequently, the RBER reflects the average value for all pages. Looking at Figure 9.1a, it is clear that the LSB and MSB pages have similar performance, while the CSB page has an overall higher error rate. Although joint processing, i.e., cell-wise encoding, would outperform the worst-case scenario, it remains behind the accuracy achieved by the LSB and MSB sides. Testing the worst-case page is essential as it represents one-third of the total memory for TLC flash memory. By contrast, in Figure 9.1b, the worst-case page of Gray code 2, i.e., the LSB page of the code, shows a considerable deterioration. On the other hand, the MSB side maintains a high degree of accuracy, and the CSB page exhibits robust error correction capabilities. Consequently, joint decoding proves to be a viable option in these examples. Using pagewise decoding, Gray code 1 shows a better balance between the individual pages than Gray code 2.

Further, Figure 9.2 compares the GC code with the rate $R = 0.87$ and 2kB from Table 9.2 using a TLC model. Figure 9.2a illustrates the FER for the different pages and the joint decoding using Gray code 1. Figure 9.2b shows the performance with Gray code 2. In general, the analyzed code shows stronger robustness compared to Figure 9.1 due to the use of a lower

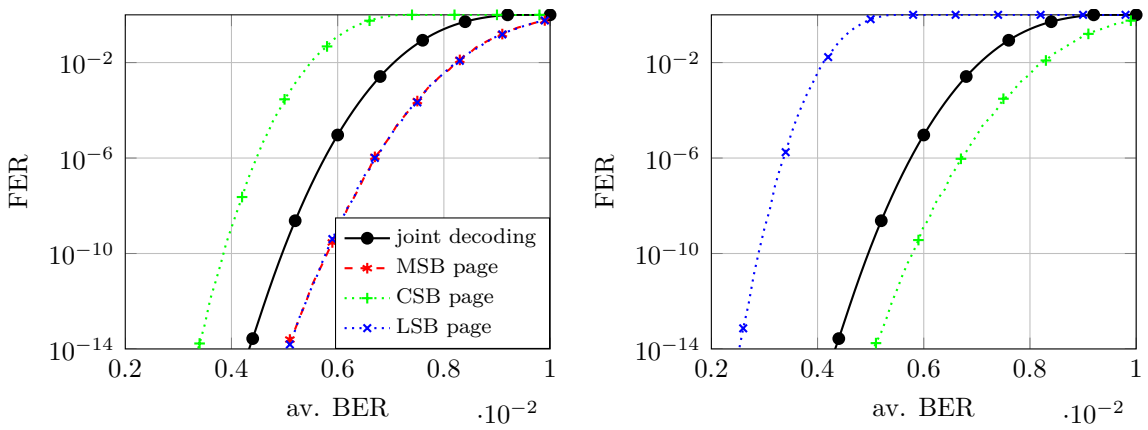


Figure 9.2: GC codes of rate $R = 0.87$ and 2kB payload data code for TLC model with page-wise and joint decoding using **(a)** Gray code 1 and **(b)** Gray code 2 (adapted from [2], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/).

code rate, leading to a shift of the results along the x-axis. The interpretation of the results remains similar: Gray code 1 shows balanced results for page-wise decoding, while Gray code 2 shows significant differences between the pages. Note that the MSB page with Gray code 2 achieves an FER of less than 10^{-15} at an average RBER of 10^{-2} and is therefore not visible in the figure. Moreover, joint decoding achieves a favorable compromise and provides robust results.

Table 9.5: Possible Gray code bit-labeling for QLC flash cell ([2], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/).

Charge state	page no.	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}
Gray code 4	1	1	1	0	0	0	1	1	1	1	1	1	0	0	0	0	0
	2	1	0	0	0	0	0	0	1	1	1	0	0	1	1	1	1
	3	1	1	1	0	0	0	0	0	0	1	1	1	1	0	0	1
	4	1	1	1	1	0	0	1	1	0	0	0	0	0	0	1	1
Gray code 5	1	1	1	1	1	0	0	0	0	0	0	0	1	1	1	1	0
	2	1	1	1	0	0	0	0	1	1	1	1	1	0	0	0	0
	3	1	1	0	0	0	0	1	1	1	0	0	0	0	1	1	1
	4	1	0	0	0	0	1	1	1	0	0	1	1	1	1	0	0

Subsequently, we present results for Gray codes explicitly designed for QLC flash memory. In QLC flash memory, four pages are distinguished and numbered sequentially, starting with the MSB page as page 1 and the LSB page as page 4. Table 9.5 shows two Gray code labelings suitable for QLC flash. Gray code 4 has been previously presented in [166]. Page 1 has three thresholds in this code, while the rest of the pages each employ each four thresholds for bit selection. This labeling approach results in well-balanced error probabilities across the different pages. Specifically, the page error probabilities for good channel conditions are approximately $p_{b,page1} \approx \frac{3}{8} p_e$ and $p_{b,page2} = p_{b,page3} = p_{b,page4} \approx \frac{1}{4} p_e$.

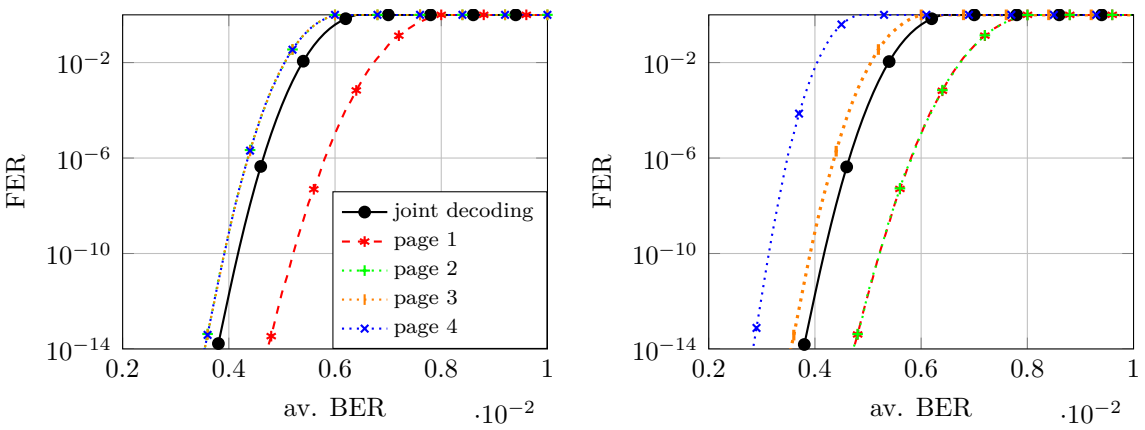


Figure 9.3: GC codes (from Table 9.3) of rate $R = 0.9$ and 4kB payload data for a QLC model using **(a)** Gray code 4 and **(b)** Gray code 5 (adapted from [2], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/).

Figure 9.3a presents numerical results for the GC code from Table 9.3 employing Gray code 4. Notably, pages 2, 3, and 4 exhibit similar FERs, while page 1 demonstrates a lower FER than other pages. With this particular code, joint processing using the GC code yields only a marginal performance improvement when compared to the worst-case pages.

On the other hand, Figure 9.3b displays numerical results employing Gray code 5 in conjunction with the GC code described in Table 9.3. Gray code 5 adopts a configuration with three thresholds for pages 1 and 2, four thresholds for page 3, and five thresholds for page 4. The page error probabilities under favorable channel conditions for Gray code 5 are as follows $p_{b,page1} = p_{b,page2} \approx \frac{3}{8}p_e$, $p_{b,page3} \approx \frac{1}{2}p_e$ and $p_{b,page4} \approx \frac{5}{8}p_e$. In this scenario, the GC code with joint encoding achieves a substantial performance gain when compared to the worst-case pages.

In Figure 9.4, we present results for two Gray codes in conjunction with the GC code detailed in Table 9.4. In Figure 9.4a, the curves for pages 2, 3, and 4 overlap, indicating similar performance. When employing page-wise encoding, Gray code 5 outperforms Gray code 4. Furthermore, joint encoding enhances reliability compared to the worst pages.

These results underscore the significant impact of bit-labeling on page-wise decoding approaches. The worst-case page primarily influences the average error rates of the different Gray codes. While longer codes can improve error correction, they can also increase the random access read latency. Similarly, joint processing also increases latency. However, joint encoding yields more substantial gains than simply increasing code length for a single page.

Since this section is specifically dedicated to GC codes, the subsequent section delves into cell-wise processing employing LDPC codes.

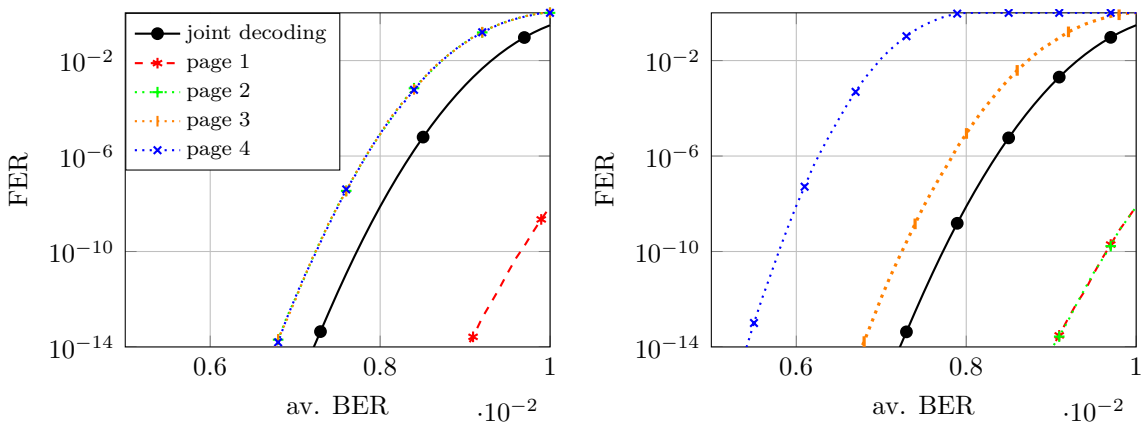


Figure 9.4: GC codes (from Table 9.4) of rate $R = 0.85$ and 4kB payload data for a **(a)** QLC model using Gray code 4 and **(b)** Gray code 5 (adapted from [2], CC BY 4.0, https://creativecommons.org/licenses/by/4.0/).

9.3 LDPC Codes for Cell-Wise Processing

This section investigates and applies methods for cell-wise data processing in TLC flash memory using LDPC codes in conjunction with a soft-input decoder. Our primary focus is to compare page-wise and cell-wise processing. Depending on the specific flash technology and data mapping, page-wise processing can lead to one or more pages with an increased error rate, as demonstrated in the previous section. These pages dominate the average error probability and require soft readouts at an early life-cycle stage. We will show that joint processing improves the error correction performance for TLC flash memory.

9.3.1 Bit-Interleaved Coded Modulation for Flash Memory

The multilevel coding approach requires distinct code rates for the different pages. Furthermore, it requires processing earlier coding stages when a lower level needs decoding. This entails a type of joint processing of all pages at the decoder, which may not be ideal. An alternative approach is Bit-Interleaved Coded Modulation (BICM), where all bits are encoded using the same code, and bit-wise interleaving is employed at the encoder's output [167, 168]. Initially developed for fading channels, BICM has proven effective at approaching the capacity of high-SNR AWGN channels [169]. While BICM has been mainly considered in the context of error correction for MLC flash memories [168, 170], we extend its application to TLC flash memories.

The BICM approach relies on an appropriate soft-decision metric as input to the decoder. This soft-decision metric enables soft-input decoding for TLC memories without requiring additional readings at reference voltages. The demapper leverages the distinct error probabilities associated with the charge states.

Let $\mathcal{C}$ be the Gray code for mapping information bits onto charge states. We partition the code $\mathcal{C}$ into two subsets for the demapping of the ith bit. The set $\mathcal{C}_i^{(0)} \subset \mathcal{C}$ comprises binary vectors with a zero in the i-th position. Likewise, the codewords of $\mathcal{C}_i^{(1)} \subset \mathcal{C}$ feature a one in the i-th position.

The demapper computes bit-wise LLRs as

$$L(x_i) = \log \frac{\sum_{\mathbf{x} \in \mathcal{C}_i^{(0)}} P(\mathbf{y}|\mathbf{x}) P(\mathbf{x})}{\sum_{\mathbf{x} \in \mathcal{C}_i^{(1)}} P(\mathbf{y}|\mathbf{x}) P(\mathbf{x})}, \tag{9.1}$$

using the conditional probability $P(\mathbf{y}|\mathbf{x})$, which represents the conditional probability of observing $\mathbf{y}$ given that the codeword $\mathbf{x}$ was stored, and $P(\mathbf{x})$ is the a priori probability. Typically, we assume that all a priori probabilities $P(\mathbf{x})$ are equal. Furthermore, for TLC flash memories, the maximum terms in the two sums tend to dominate the sums. Therefore, we can employ the max-log approximation, where

$$L(x_i) \approx \log \frac{\max_{\mathbf{x} \in \mathcal{C}_i^{(0)}} P(\mathbf{y}|\mathbf{x})}{\max_{\mathbf{x} \in \mathcal{C}_i^{(1)}} P(\mathbf{y}|\mathbf{x})}. \tag{9.2}$$

Note that this demapping can be efficiently implemented with a simple lookup table, as the channel output $\mathbf{y}$ distinguishes only eight values.

For the error correction coding, we use irregular LDPC codes similar to those used in the DVB-S2 standard [171]. The decoder employs the belief propagation decoding algorithm with a maximum of ten iterations.

In the next section, we will demonstrate how the joint calculation of LLRs can enhance the performance of page-wise coding. This decoding method is similar to the approach proposed in [163] for MLC flash memory. The decoder initially starts with page-wise processing, but it reads the other pages if required to enhance the decoding of the current page.

9.3.2 Comparing Cell-Wise and Page-Wise Processing using LDPC Codes

In this section, we present simulation results for a TLC model to compare the performance of cell-wise and page-wise processing, focusing on HD decoding. Initially, we consider a code for 1kB payload data and code rate of $R = 0.83$, i.e., an LDPC code with a dimension $k = 8192$ and code length $n = 9870$. These code parameters are proposed in the IEEE standard [87].

Figure 9.5 illustrates the page-wise decoding and joint processing for Gray code 1. In this scenario, the CSB page represents the worst-case page. Note that as for the MI consideration, the residual FER of the LSB and MSB page are equal. Using the HD decoding, the joint processing outperforms even the best-case page-wise decoding. This improvement arises from the additional information obtained through joint LLR calculation. To obtain this enhanced

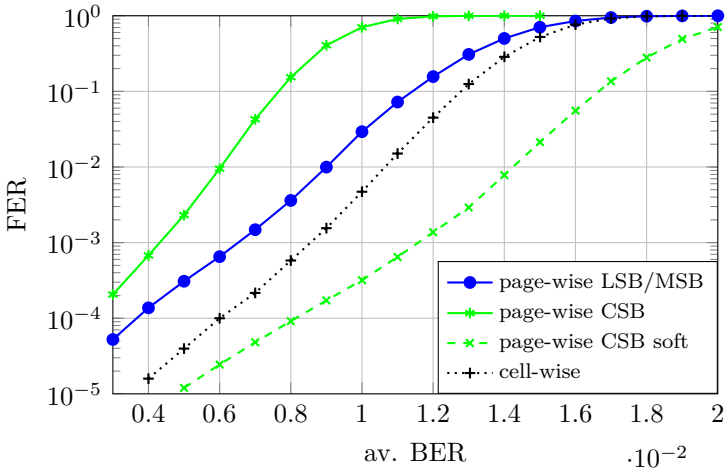


Figure 9.5: LDPC code with parameters $n = 9870$, $k = 8192$, and $R = 0.83$ ([4], ©2022 IEEE).

information, further processing is required to ensure an accurate LLR calculation, which involves mapping the read joint information to the selected bit mapping. This assignment is not possible with a simple HD readout. The incorporation of this additional information is essential for generating LLRs for the individual bits. It is worth noting that the mapping and processing can only take place within the flash controller.

Furthermore, we present results for soft-decision decoding with one SB in Figure 9.5, specifically for the CSB page. The threshold voltages for the soft-input are chosen to maximize the MI. The decoding performance of SD decoding significantly outperforms the HD decoding approaches. This example clearly illustrates the significant benefit of employing SD readout on performance. It can be reasonably assumed that cell-wise processing with SD readout would likewise result in a substantial improvement.

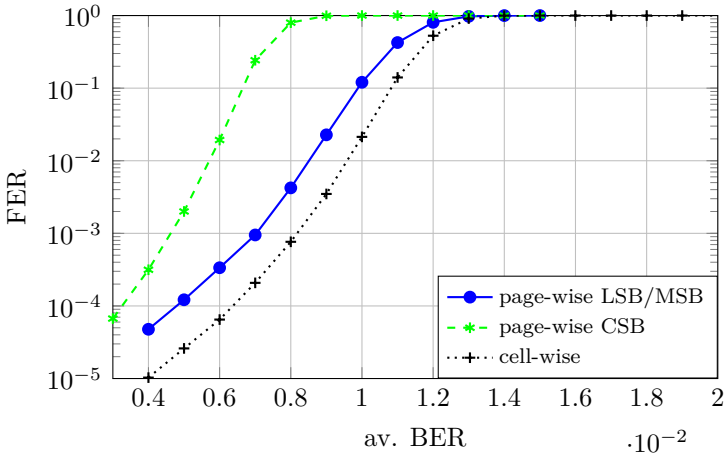


Figure 9.6: LDPC code with parameters $n = 38\,556$, $k = 32\,768$, and $R = 0.85$ ([4] ©2022 IEEE).

Subsequently, we consider a code designed for 4kB payload data with a code rate of $R =$

0.85. This corresponds to the parameters $k = 32\,768$ and $n = 38\,556$. This code length is common for flash memory devices [14]. Figure 9.6 contains the results for this code, which are comparable to those in Figure 9.5. This shows that increasing the code length of the ECC achieves a gain in the achievable FER.

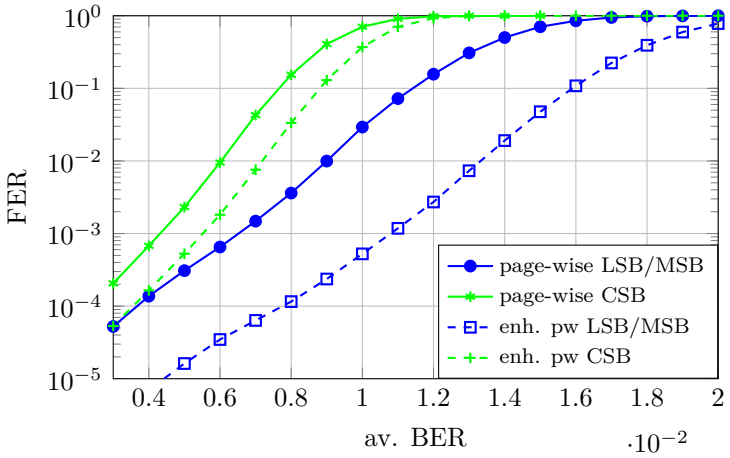


Figure 9.7: LDPC 1kB page-wise read in comparison to progressive read ([4], ©2022 IEEE).

In Figure 9.7, the 1kB LDPC code is used for page-wise encoding, where two different decoding methods are compared. With page-wise processing, only the read reference for the considered page are evaluated. The enhanced page-wise processing evaluates all voltages and uses joint processing to determine the LLR values. The legend denotes the corresponding results as enhanced page-wise (enh. pw).

The results reveal that joint processing leads to a performance improvement for all pages. However, the gain for the LSB and MSB pages is more pronounced compared to the worst-case CSB page.

9.4 Summary

This chapter has focused on error correction approaches for TLC and QLC flash memory, involving a comparison between page-wise and cell-wise processing approaches. The cell-wise processing demonstrates an improvement in achievable channel capacity compared to the conventional method of independent page-wise processing.

First, we considered HD decoding on GC codes. The primary objective of the proposed method is to delay the use of costly soft-input decoding as long as possible. Notably, we have shown that the performance of joint decoding remains consistent across different Gray codes, whereas the specific bit-labeling strongly influences error probabilities in page-wise processing. In page-wise processing, the average error probability across all pages is predominantly determined by the page with the highest error probability.

Next, we focused on implementing a cell-wise processing approach using only HD readout in conjunction with a binary LDPC decoder. Therefore, we applied the BICM technique on the flash memory channel for cell-wise processing to calculate LLRs, which are exploited by the LDPC decoder.

Note that the BICM approach requires the simultaneous reading of all pages associated with a cell, potentially increasing read latency due to the application of multiple read voltages. However, it offers the advantage of eliminating the need for additional voltages for soft-reading. Moreover, the proposed processing can also be applied to page-wise encoding, where information from other pages is only used when HD decoding for a specific page fails. Further improvement of the LDPC code can result in the application of non-binary alphabets for decoding.

The primary objective of this chapter was not to design highly performant LDPC or GC codes but rather to explore the processing techniques for TLC and QLC flash memories.

Chapter 10

Conclusion

This thesis has presented various methods to enhance the reliability of flash memories. It started with a comprehensive characterization of the flash channel, identifying multiple error sources. We explored different bit-labeling schemes and examined achievable capacities.

Additionally, we investigated the importance of adapting read reference voltages (RRVs) to the dynamic life-cycle conditions. We have shown that fixed predefined read thresholds can escalate residual bit error rates (RBERs), potentially leading to decoding failures. The adaptation of RRVs is crucial to extend the reliability of the data on the flash memory. Various adaptation techniques were presented, predominantly centered on statistical features, providing promising results. Of particular interest was an approach driven by neural networks (NN). Remarkably, our research demonstrates that NNs trained using data from life-cycle extremities show impressive generalization across varying life-cycle conditions. Even with perfect readout conditions, it is impossible to eliminate the resulting residual error rate. Therefore, error-correcting techniques remain essential.

Along with this work, challenges associated with error correction codes (ECCs) were addressed, and algorithms aimed at reducing decoding complexity were introduced. Therefore, we presented a hybrid decoding approach based on algebraic decoding combined with ordered statistics decoding (OSD). We used a codeword acceptance criterion to maintain the computational complexity low while achieving excellent error correction performance. Simulation results have confirmed this approach.

Further, we have explored the Plotkin construction using BCH component codes to build larger codes, keeping the decoding complexity low. Binary Reed-Muller (RM) codes can be interpreted as a specialized variant of this construction. However, the larger Plotkin code family promises improved code parameters. To enable soft decoding for these codes, we generalized the GMC soft decoding method proposed by Schnabl et al. in [122]. To keep the complexity low, we employed the hybrid decoding algorithm leveraging the recursive code structure to reduce decoding complexity. Simulation results showed the efficiency of this decoding procedure.

Finally, the thesis provides an error correction approach for TLC and QLC flash memories. This approach relies on joint processing for all bits within a cell, which offers several advantages

over conventional independent page-wise processing. Employing this cell-wise processing made it possible to enhance the achievable channel capacity. While this readout method may lead to increased latency compared to page-wise readout, it potentially eliminates the need for computationally intensive soft readouts. It is important to note that we have focused on using binary codes in this work. However, future investigations should consider employing q-nary codes for joint processing, as this strategy could offer improved performance and additional optimization opportunities.

In conclusion, this work has contributed to improving flash memory reliability and proposed optimizations for decoding processes. The challenges in flash memory storage are expected to persist and evolve with the continued advancement of storage density, as Moore's law dictates. Therefore, ongoing research into innovative processing strategies and error correction methods remains essential.

Bibliography

[1] J.-P. Thiers, D. Nicolas Bailon, and J. Freudenberger, "Bit-labeling and page capacities of TLC non-volatile flash memories," in *2020 IEEE 10th International Conference on Consumer Electronics (ICCE-Berlin)*. IEEE, nov 2020.

[2] D. Nicolas Bailon, J.-P. Thiers, and J. Freudenberger, "Error correction for TLC and QLC NAND flash memories using cell-wise encoding," *Electronics*, vol. 11, no. 10, p. 1585, may 2022.

[3] J.-P. Thiers, D. Nicolas Bailon, J. Freudenberger, and J. Lu, "Read reference calibration and tracking for non-volatile flash memories," *Electronics*, vol. 10, no. 18, p. 2306, sep 2021.

[4] D. Nicolas Bailon, S. Shavgulidze, and J. Freudenberger, "Cell-wise encoding and decoding for TLC flash memories," in *2022 IEEE 12th International Conference on Consumer Electronics (ICCE-Berlin)*. IEEE, sep 2022.

[5] M. Safieh, D. Nicolas Bailon, and J. Freudenberger, "An acceptance criterion for hybrid algebraic and soft-input decoding," in *2020 24th International Conference on Information Technology (IT)*. IEEE, feb 2020.

[6] D. Nicolas Bailon, G. Taburet, S. Shavgulidze, and J. Freudenberger, "Neural network aided reference voltage adaptation for NAND flash memory," in *2022 IEEE 12th International Conference on Consumer Electronics (ICCE-Berlin)*. IEEE, sep 2022.

[7] D. Nicolas Bailon, S. Shavgulidze, and J. Freudenberger, "Read reference voltage adaptation for NAND flash memories with neural networks based on sparse histograms," *IEEE Access*, vol. 11, pp. 56 801–56 811, 2023.

[8] D. Nicolas Bailon, S. Shavgulidze, and J. Freudenberger, "Features for read reference voltage estimation in NAND flash memory," in *2023 IEEE 13th International Conference on Consumer Electronics (ICCE-Berlin)*. IEEE, Sep. 2023.

[9] J. Freudenberger, D. Nicolas Bailon, and M. Safieh, "Reduced complexity hard- and soft-input BCH decoding with applications in concatenated codes," *IET Circuits, Devices Systems*, vol. 15, no. 3, pp. 284–296, mar 2021.

[10] D. Nicolas Bailon and J. Freudenberger, "List decoding for concatenated codes based on the Plotkin construction with BCH component codes," in *2021 11th International Symposium on Topics in Coding (ISTC)*. IEEE, aug 2021.

[11] D. Nicolas Bailon, M. Bossert, J.-P. Thiers, and J. Freudenberger, "Concatenated codes based on the Plotkin construction and their soft-input decoding," *IEEE Transactions on Communications*, vol. 70, no. 5, pp. 2939–2950, may 2022.

[12] D. Nicolas Bailon, J.-P. Thiers, and J. Freudenberger, "Soft-input decoding of concatenated codes based on the Plotkin construction and BCH component codes," in *2020 IEEE 10th International Conference on Consumer Electronics (ICCE-Berlin)*. IEEE, nov 2020.

[13] R. Micheloni, A. Marelli, and R. Ravasio, *Error correction codes for non-volatile memories*. Springer, 2008.

[14] R. Micheloni, Ed., *3D flash memories*. Springer Netherlands, 2016.

[15] A. Goda, "Recent progress on 3D NAND flash technologies," *Electronics*, vol. 10, no. 24, p. 3156, dec 2021.

[16] T. Parnell, C. Dünner, T. Mittelholzer, and N. Papandreou, "Capacity of the MLC NAND flash channel," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 9, pp. 2354–2365, aug 2016.

[17] Y. Wang, L. A. D. Bathen, N. D. Dutt, and Z. Shao, "Meta-cure," in *Proceedings of the 49th Annual Design Automation Conference*. ACM, jun 2012.

[18] M. Huang, Z. Liu, and L. Qiao, "Asymmetric programming: A highly reliable metadata allocation strategy for MLC NAND flash memory-based sensor systems," *Sensors*, vol. 14, no. 10, pp. 18 851–18 877, oct 2014.

[19] J. Spinner, M. Rajab, and J. Freudenberger, "Construction of high-rate generalized concatenated codes for applications in non-volatile flash memories," in *IEEE 8th International Memory Workshop (IMW)*, may 2016, pp. 1–4.

[20] J. Spinner, "Channel coding for flash memories," Ph.D. dissertation, Universität Konstanz, Konstanz, 2019.

[21] D. Kahng and S. M. Sze, "A floating gate and its application to memory devices," *Bell System Technical Journal*, vol. 46, no. 6, pp. 1288–1295, jul 1967.

[22] R. H. Fowler and L. Nordheim, "Electron emission in intense electric fields," *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, vol. 119, no. 781, pp. 173–181, may 1928.

[23] R. Micheloni, L. Crippa, and A. Marelli, *Inside NAND Flash Memories*. Springer Netherlands, 2010.

[24] D. Richter, *Flash memories*. Springer Netherlands, 2014.

[25] H. Kim, S.-J. Ahn, Y. G. Shin, K. Lee, and E. Jung, "Evolution of NAND flash memory: From 2D to 3D as a storage market leader," in *2017 IEEE International Memory Workshop (IMW)*. IEEE, may 2017.

[26] H. Tanaka, M. Kido, K. Yahashi, M. Oomura, R. Katsumata, M. Kito, Y. Fukuzumi, M. Sato, Y. Nagata, Y. Matsuoka, Y. Iwata, H. Aochi, and A. Nitayama, "Bit cost scalable technology with punch and plug process for ultra high density flash memory," in *2007 IEEE Symposium on VLSI Technology*. IEEE, jun 2007.

[27] J. Jang, H.-S. Kim, W. Cho, H. Cho, J. Kim, S. I. Shim, Younggoan, J.-H. Jeong, B.-K. Son, D. W. Kim, Kihyun, J.-J. Shim, J. S. Lim, K.-H. Kim, S. Y. Yi, J.-Y. Lim, D. Chung, H.-C. Moon, S. Hwang, J.-W. Lee, Y.-H. Son, U.-I. Chung, and W.-S. Lee, "Vertical cell array using TCAT(Terabit Cell Array Transistor) technology for ultra high density NAND flash memory," in *2009 Symposium on VLSI Technology*, jun 2009, pp. 192–193.

[28] B. Chen and Xinmiao, "Error correction for multi-level NAND flash memory using Reed-Solomon codes," *IEEE Workshop on Signal Processing Systems*, pp. 94–99, oct 2008.

[29] Y. Cai, S. Ghose, E. F. Haratsch, Y. Luo, and O. Mutlu, "Error characterization, mitigation, and recovery in flash-memory-based solid-state drives," *Proceedings of the IEEE*, vol. 105, no. 9, pp. 1666–1704, sep 2017.

[30] Y. Cai, Y. Luo, E. F. Haratsch, K. Mai, and O. Mutlu, "Data retention in MLC NAND flash memory: Characterization, optimization, and recovery," in *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, feb 2015.

[31] J.-D. Lee, J.-H. Choi, D. Park, and K. Kim, "Degradation of tunnel oxide by FN current stress and its effects on data retention characteristics of 90 nm NAND flash memory cells," in *2003 IEEE International Reliability Physics Symposium Proceedings, 2003. 41st Annual.* IEEE, mar 2003.

[32] R. Degraeve, F. Schuler, B. Kaczer, M. Lorenzini, D. Wellekens, P. Hendrickx, M. van-Duuren, G. Dormans, J. VanHoudt, L. Haspeslagh, G. Groeseneken, and G. Tempel, "Analytical percolation model for predicting anomalous charge loss in flash memories," *IEEE Transactions on Electron Devices*, vol. 51, no. 9, pp. 1392–1400, sep 2004.

[33] R. Yamada, Y. Mori, Y. Okuyama, J. Yugami, T. Nishimoto, and H. Kume, "Analysis of detrap current due to oxide traps to improve flash memory retention," in *2000 IEEE International Reliability Physics Symposium Proceedings. 38th Annual (Cat. No.00CH37059)*. IEEE, apr 2000.

[34] A. Brand, K. Wu, S. Pan, and D. Chin, "Novel read disturb failure mechanism induced by FLASH cycling," in *31st Annual Proceedings Reliability Physics 1993*. IEEE, mar 1993.

[35] M. Kato, N. Miyamoto, H. Kume, A. Satoh, T. Adachi, M. Ushiyama, and K. Kimura, "Read-disturb degradation mechanism due to electron trapping in the tunnel oxide for low-voltage flash memories," in *Proceedings of 1994 IEEE International Electron Devices Meeting*. IEEE, dec 1994.

[36] Y. Zhang, L. Jin, D. Jiang, X. Zou, H. Liu, and Z. Huo, "A novel read scheme for read disturbance suppression in 3D NAND flash memory," *IEEE Electron Device Letters*, vol. 38, no. 12, pp. 1669–1672, dec 2017.

[37] A. S. Spinelli, C. M. Compagnoni, and A. L. Lacaita, "Reliability of NAND flash memories: Planar cells and emerging issues in 3D devices," *Computers*, vol. 6, no. 2, mar 2017. [Online]. Available: http://www.mdpi.com/2073-431X/6/2/16

[38] 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)*, mar 2013, pp. 1285–1290.

[39] S. M. Amoroso, C. M. Compagnoni, A. Ghetti, L. Gerrer, A. S. Spinelli, A. L. Lacaita, and A. Asenov, "Investigation of the RTN distribution of nanoscale MOS devices from subthreshold to on-state," *IEEE Electron Device Letters*, vol. 34, no. 5, pp. 683–685, may 2013.

[40] D. Lee and W. Sung, "Estimation of NAND flash memory threshold voltage distribution for optimum soft-decision error correction," *IEEE Transactions on Signal Processing*, vol. 61, no. 2, pp. 440–449, jan 2013.

[41] Y.-T. Chung, T.-I. Huang, C.-W. Li, Y.-L. Chou, J.-P. Chiu, T. Wang, M. Y. Lee, K.-C. Chen, and C.-Y. Lu, "V_t retention distribution tail in a multitime-program MLC SONOS memory due to a random-program-charge-induced current-path percolation effect," *IEEE Transactions on Electron Devices*, vol. 59, no. 5, pp. 1371–1376, may 2012.

[42] C. Monzio Compagnoni, M. Ghidotti, A. L. Lacaita, A. S. Spinelli, and A. Visconti, "Random telegraph noise effect on the programmed threshold-voltage distribution of flash memories," *IEEE Electron Device Letters*, vol. 30, no. 9, pp. 984–986, aug 2009.

[43] T. Parnell, N. Papandreou, T. Mittelholzer, and H. Pozidis, "Modelling of the threshold voltage distributions of sub-20nm NAND flash memory," in *IEEE Global Communications Conference*, dec 2014, pp. 2351–2356.

[44] J. Freudenberger, M. Rajab, and S. Shavgulidze, "A source and channel coding approach for improving flash memory endurance," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 5, pp. 981–990, may 2018.

[45] Y. Luo, S. Ghose, Y. Cai, E. F. Haratsch, and O. Mutlu, "Enabling accurate and practical online flash channel modeling for modern MLC NAND flash memory," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 9, pp. 2294–2311, aug 2016.

[46] C. Miccoli, G. M. Paolucci, C. M. Compagnoni, A. S. Spinelli, and A. Goda, "Cycling pattern and read/bake conditions dependence of random telegraph noise in decananometer NAND flash arrays," in *IEEE International Reliability Physics Symposium*, may 2015, pp. MY.9.1–MY.9.6.

[47] T. Samadder, S. Kumar, K. Thakor, and S. Mahapatra, "A theoretical framework for trap generation and passivation in NAND flash tunnel oxide during distributed cycling

and retention bake," in *IEEE International Reliability Physics Symposium (IRPS)*, mar 2021, pp. 1–6.

[48] H. Wang, T. Y. Chen, and R. D. Wesel, "Histogram-based flash channel estimation," in *IEEE International Conference on Communications (ICC)*, jun 2015, pp. 283–288.

[49] A. Hareedy, B. Dabak, and R. Calderbank, "Q-ary asymmetric LOCO codes: Constrained codes supporting flash evolution," in *2020 IEEE International Symposium on Information Theory (ISIT)*. IEEE, jun 2020.

[50] R. Gabrys and L. Dolecek, "Constructions of non-binary WOM-codes for multilevel flash memories," *IEEE Transactions on Information Theory*, vol. PP, no. 99, pp. 1–1, oct 2015.

[51] G. Yadgar, E. Yaakobi, F. Margaglia, Y. Li, A. Yucovich, N. Bundak, L. Gilon, N. Yakovi, A. Schuster, and A. Brinkmann, "An analysis of flash page reuse with WOM codes," *ACM Transactions on Storage*, vol. 14, no. 1, pp. 1–39, apr 2018.

[52] G. Ungerboeck, "Channel coding with multilevel/phase signals," *IEEE Trans. on Inform. Theory*, pp. 55–67, jan 1982.

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

[54] M. Rajab, J. Freudenberger, and S. Shavgulidze, "Soft-input bit-flipping decoding of generalized concatenated codes for application in non-volatile flash memories," *IET Communications*, nov 2018.

[55] S. Qi, D. Feng, and J. Liu, "Optimal voltage signal sensing of NAND flash memmory for LDPC code," in *Signal Processing Systems (SiPS), 2014 IEEE Workshop on*, oct 2014, pp. 1–6.

[56] U. Wachsmann, R. Fischer, and J. Huber, "Multilevel codes: theoretical concepts and practical design rules," *IEEE Transactions on Information Theory*, vol. 45, no. 5, pp. 1361–1391, jul 1999.

[57] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, jul 1948.

[58] R. G. Gallager, *Information theory and reliable communication*. New York, NY, USA: John Wiley & Sons, Inc., 1968.

[59] F. Xiong, *Digital modulation techniques, second edition (Artech House Telecommunications Library)*. USA: Artech House, Inc., 2006.

[60] J. Freudenberger, M. Rajab, and S. Shavgulidze, "Estimation of channel state information for non-volatile flash memories," in *IEEE 7th International Conference on Consumer Electronics (ICCE)*, sep 2017.

[61] M. Sandell and A. Ismail, "Machine learning for LLR estimation in flash memory with LDPC codes," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, no. 2, pp. 792–796, aug 2021.

[62] F. Sala, R. Gabrys, and L. Dolecek, "Dynamic threshold schemes for multi-level non-volatile memories," *IEEE Transactions on Communications*, vol. 61, no. 7, pp. 2624–2634, jul 2013.

[63] H. Zhou, A. Jiang, and J. Bruck, "Error-correcting schemes with dynamic thresholds in nonvolatile memories," in *2011 IEEE International Symposium on Information Theory Proceedings*. IEEE, jul 2011.

[64] D. hwan Lee and W. Sung, "Decision directed estimation of threshold voltage distribution in NAND flash memory," *IEEE Transactions on Signal Processing*, vol. 62, no. 4, pp. 919–927, feb 2014.

[65] B. Peleato, R. Agarwal, J. M. Cioffi, M. Qin, and P. H. Siegel, "Adaptive read thresholds for NAND flash," *IEEE Transactions on Communications*, vol. 63, no. 9, pp. 3069–3081, sep 2015.

[66] M. Rajab, J.-P. Thiers, and J. Freudenberger, "Read threshold calibration for non-volatile flash memories," in *2019 IEEE 9th International Conference on Consumer Electronics (ICCE-Berlin)*. IEEE, sep 2019.

[67] W. Zhang, J. Kang, Z. Dong, and Y. Zhu, "RS-LDPC concatenated coding for NAND flash memory: Designs and reduction of short cycles," in *2020 IEEE 3rd International Conference on Information Communication and Signal Processing (ICICSP)*. IEEE, sep 2020.

[68] L. Yang, Q. Wang, Q. Li, X. Yu, J. He, and Z. Huo, "Gradual channel estimation method for TLC NAND flash memory," *IEEE Embedded Systems Letters*, vol. 14, no. 1, pp. 7–10, mar 2022.

[69] H. Choe, J. Jee, S.-C. Lim, S. M. Joe, I. H. Park, and H. Park, "Machine-learning-based read reference voltage estimation for NAND flash memory systems without knowledge of retention time," *IEEE Access*, vol. 8, pp. 176 416–176 429, sep 2020.

[70] Z. Mei, K. Cai, and X. He, "Deep learning-aided dynamic read thresholds design for multi-level-cell flash memories," *IEEE Transactions on Communications*, vol. 68, no. 5, pp. 2850–2862, may 2020.

[71] M. Hwang, J. Jee, J. Kang, H. Park, S. Lee, and J. Kim, "Dynamic error recovery flow prediction based on reusable machine learning for low latency NAND flash memory under process variation," *IEEE Access*, vol. 10, pp. 117 715–117 731, nov 2022.

[72] J. Wang, K. Vakilinia, T.-Y. Chen, T. Courtade, G. Dong, T. Zhang, H. Shankar, and R. Wesel, "Enhanced precision through multiple reads for LDPC decoding in flash memories," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 5, pp. 880–891, may 2014.

[73] M. Zhang, F. Wu, Q. Yu, W. Liu, Y. Wang, and C. Xie, "Exploiting error characteristic to optimize read voltage for 3-d NAND flash memory," *IEEE Transactions on Electron Devices*, vol. 67, no. 12, pp. 5490–5496, dec 2020.

[74] E. Yaakobi, J. Ma, L. Grupp, P. H. Siegel, S. Swanson, and J. K. Wolf, "Error characterization and coding schemes for flash memories," in *2010 IEEE Globecom Workshops*. IEEE, dec 2010.

[75] T. M. Cover and J. A. Thomas, *Elements of information theory*. John Wiley & Sons, 1991.

[76] D. MacKay, *Information theory, inference, and learning algorithms*. Cambridge University Press, 2003.

[77] B. Vasić, D. Declercq, and B. Reynwar, "Iterative decoders robust to threshold voltage uncertainty," in *9th annual Non-Volatile Memories Workshop 2018*, mar 2018.

[78] J. Cha and S. Kang, "Data randomization scheme for endurance enhancement and interference mitigation of multilevel flash memory devices," *ETRI Journal*, vol. 35, no. 1, pp. 166–169, feb 2013.

[79] S. Chattopadhyay, P. Kumari, B. Ray, and R. S. Chakraborty, "Machine learning assisted accurate estimation of usage duration and manufacturer for recycled and counterfeit flash memory detection," in *IEEE 28th Asian Test Symposium (ATS)*, dec 2019, pp. 49–495.

[80] J. K. Lee, K. Ko, and H. Shin, "Analysis on process variation effect of 3D NAND flash memory cell through machine learning model," in *4th IEEE Electron Devices Technology & Manufacturing Conference (EDTM)*, apr 2020, pp. 1–4.

[81] C. Zambelli, G. Cancelliere, F. Riguzzi, E. Lamma, P. Olivo, A. Marelli, and R. Micheloni, "Characterization of TLC 3D-NAND flash endurance through machine learning for LDPC code rate optimization," in *IEEE International Memory Workshop (IMW)*, may 2017, pp. 1–4.

[82] R. Rojas, *Neural networks*. Springer Berlin Heidelberg, 1996.

[83] M. T. Hagan, H. B. Demuth, M. H. Beale, and O. D. Jesús, *Neural network design*, 2nd ed. Martin Hagan, 2009.

[84] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the Society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, jun 1963.

[85] J. J. Moré, "The Levenberg-Marquardt algorithm: Implementation and theory," in *Lecture Notes in Mathematics*. Springer Berlin Heidelberg, jan 1978, pp. 105–116.

[86] S. J. W. J. Nocedal, *Numerical optimization*. Springer New York, 2006.

[87] "IEEE standard for error correction coding of flash memory using low-density parity check codes," *IEEE Std 1890-2018*, pp. 1–51, feb 2019.

[88] M. Bossert, *Einführung in die Nachrichtentechnik*. Oldenbourg Wissenschaftsverlag Verlag, 2012.

[89] J. Spinner, J. Freudenberger, and S. Shavgulidze, "A soft input decoding algorithm for generalized concatenated codes," *IEEE Transactions on Communications*, vol. 64, no. 9, pp. 3585–3595, sep 2016.

[90] T. Tanzawa, T. Tanaka, K. Takeuchi, R. Shirota, S. Aritome, H. Watanabe, G. Hemink, K. Shimizu, S. Sato, Y. Takeuchi, and K. Ohuchi, "A compact on-chip ECC for low cost flash memories," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 5, pp. 662–669, may 1997.

[91] S. Lin and D. J. Costello, *Error control coding*. Upper Saddle River, NJ: Prentice-Hall, 2004.

[92] M. Bossert, *Kanalcodierung*. Oldenbourg Wissenschaftsverlag, 2013. [Online]. Available: https://doi.org/10.1524/9783486755169

[93] D. E. Muller, "Application of Boolean algebra to switching circuit design and to error detection," *Transactions of the I.R.E. Professional Group on Electronic Computers*, vol. EC-3, no. 3, pp. 6–12, sep 1954.

[94] I. Reed, "A class of multiple-error-correcting codes and the decoding scheme," *Transactions of the IRE Professional Group on Information Theory*, vol. 4, no. 4, pp. 38–49, sep 1954.

[95] N. Stolte, "Rekursive Codes mit der Plotkin-Konstruktion und ihre Decodierung," *PhD Thesis, TU Darmstadt*, 2002.

[96] E. Arikan, "A survey of Reed-Muller codes from polar coding perspective," in *IEEE Information Theory Workshop on Information Theory (ITW 2010, Cairo)*, jan 2010, pp. 1–5.

[97] E. Abbe and M. Ye, "Reed-Muller codes polarize," *IEEE Transactions on Information Theory*, vol. 66, no. 12, pp. 7311–7332, jan 2020.

[98] E. Abbe, A. Shpilka, and M. Ye, "Reed-Muller codes: Theory and algorithms," *IEEE Transactions on Information Theory*, pp. 1–1, jun 2020.

[99] S. Kudekar, S. Kumar, M. Mondelli, H. D. Pfister, E. Sasoglu, and R. L. Urbanke, "Reed-Muller codes achieve capacity on erasure channels," *IEEE Transactions on Information Theory*, vol. 63, no. 7, pp. 4298–4316, feb 2017.

[100] M. Plotkin, "Binary codes with specified minimum distance," *IEEE Transactions on Information Theory*, pp. 445–450, sep 1960.

[101] R. Lucas, M. Bossert, and A. Dammann, "Improved list-decoding for reed-muller codes as generalized multiple concatenated (gmc) codes," in *Proceedings. 1998 IEEE International Symposium on Information Theory (Cat. No.98CH36252)*, ser. ISIT-98. IEEE, Aug. 1998.

[102] J. Freudenberger and J. Spinner, "A configurable Bose-Chaudhuri-Hocquenghem codec architecture for flash controller applications," *Journal of Circuits, Systems, and Computers*, vol. 23, no. 2, pp. 1–15, feb 2014.

[103] Y. Jiang, *A practical guide to error-control coding using Matlab*. Artech House, 2010.

[104] H. Schneider-Obermann, "Kanalcodierung," in *Informationstechnik kompakt*. Vieweg+Teubner Verlag, 1999, pp. 201–242.

[105] W. Peterson, "Encoding and error-correction procedures for the Bose-Chaudhuri codes," *IRE Transactions on Information Theory*, vol. 6, no. 4, pp. 459–470, sep 1960.

[106] E.-H. Lu, S.-W. Wu, and Y.-C. Cheng, "A decoding algorithm for triple-error-correcting binary BCH codes," *Information Processing Letters*, vol. 80, no. 6, pp. 299 – 303, dec 2001.

[107] Y. Lee, B. Park, D. Yun, Y. Jeong, P. H. Kim, J. Y. Park, H. C. Yang, M. K. Cho, K.-O. Ahn, and Y. Koh, "The challenges and limitations on triple level cell geometry and process beyond 20 nm NAND flash technology," in *IEEE International Memory Workshop (IMW)*, may 2010, pp. 1–2.

[108] X. Zhang and Z. Wang, "A low-complexity three-error-correcting BCH decoder for optical transport network," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 59, no. 10, pp. 663–667, aug 2012.

[109] W. Liu, J. Rho, and W. Sung, "Low-power high-throughput BCH error correction VLSI design for multi-level cell NAND flash memories," in *IEEE Workshop on Signal Processing Systems Design and Implementation (SIPS)*, oct 2006, pp. 303 –308.

[110] H. Tsai, C. Yang, and H. Chang, "An efficient BCH decoder with 124-bit correctability for multi-channel SSD applications," in *2012 IEEE Asian Solid State Circuits Conference (A-SSCC)*, nov 2012, pp. 61–64.

[111] R. Chien, "Cyclic decoding procedures for Bose-Chaudhuri-Hocquenghem codes," *IEEE Transactions on Information Theory*, vol. 10, no. 4, pp. 357–363, oct 1964.

[112] J. Spinner and J. Freudenberger, "Decoder architecture for generalized concatenated codes," *IET Circuits, Devices & Systems*, vol. 9, no. 5, pp. 328–335, sep 2015.

[113] D. Strukov, "The area and latency tradeoffs of binary bit-parallel BCH decoders for prospective nanoelectronic memories," in *2006 Fortieth Asilomar Conference on Signals, Systems and Computers*, oct 2006, pp. 1183–1187.

[114] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, jun 1960.

[115] R. G. Gallager, *Low-density parity-check codes*. M.I.T. Press, Cambridge, Massachusetts, 1963.

[116] D. MacKay and R. Neal, "Near Shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 32, no. 18, p. 1645, aug 1996.

[117] T. Richardson, M. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 619–637, feb 2001.

[118] T. Richardson, "Error floors of LDPC codes," in *Proceedings of the annual Allerton conference on communication control and computing*, vol. 41, no. 3, 2003, pp. 1426–1435.

[119] S. Chilappagari, S. Sankaranarayanan, and B. Vasic, "Error floors of LDPC codes on the binary symmetric channel," in *2006 IEEE International Conference on Communications*. IEEE, dec 2006.

[120] W. Lin, S.-W. Yen, Y.-C. Hsu, Y.-H. Lin, L.-C. Liang, T.-C. Wang, P.-Y. Shih, K.-H. Lai, K.-Y. Cheng, and C.-Y. Chang, "A low power and ultra high reliability LDPC error correction engine with digital signal processing for embedded NAND flash controller in 40nm CMOS," in *Symposium on VLSI Circuits Digest of Technical Papers*, jun 2014, pp. 1–2.

[121] Y.-C. Liao, C. Lin, H.-C. Chang, and S. Lin, "A (21150, 19050) GC-LDPC decoder for NAND flash applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 3, pp. 1219–1230, nov 2019.

[122] G. Schnabl and M. Bossert, "Soft-decision decoding of Reed-Muller codes as generalized multiple concatenated codes," *IEEE Transactions on Information Theory*, vol. 41, no. 1, pp. 304–308, jan 1995.

[123] D. Chase, "Class of algorithms for decoding block codes with channel measurement information," *IEEE Transactions on Information Theory*, pp. 170–182, jan 1972.

[124] N. Tendolkar and C. Hartmann, "Generalization of Chase algorithms for soft decision decoding of binary linear codes," *IEEE transactions on information theory*, vol. 30, no. 5, pp. 714–721, sep 1984.

[125] M. Fossorier and S. Lin, "Soft-decision decoding of linear block codes based on ordered statistics," *IEEE Transactions on Information Theory*, vol. 41, no. 5, pp. 1379–1396, sep 1995.

[126] B. Dorsch, "A decoding algorithm for binary block codes and J-ary output channels," *Information Theory, IEEE Transactions on*, vol. 20, no. 3, pp. 391–394, may 1974.

[127] M. Tomlinson, C. Tjhai, and M. Ambroze, "Extending the Dorsch decoder towards achieving maximum-likelihood decoding for linear codes," *IET Communications*, vol. 1, no. 3, pp. 479–488, jun 2007.

[128] A. Gortan, R. Jasinski, W. Godoy, and V. Pedroni, "Achieving near-MLD performance with soft information-set decoders implemented in FPGAs," in *2010 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS),*, dec 2010, pp. 312–315.

[129] M. Fossorier and A. Valembois, "Box and match soft decision decoding of linear block codes with iterative information set reduction," in *Proceedings of the IEEE Information Theory Workshop*, oct 2002, pp. 45–48.

[130] T. Kaneko, T. Nishijima, H. Inazumi, and S. Hirasawa, "An efficient maximum-likelihood-decoding algorithm for linear block codes with algebraic decoder," *IEEE Transactions on Information Theory*, vol. 40, no. 2, pp. 320–327, mar 1994.

[131] T. Kaneko, T. Nishijima, and S. Hirasawa, "An improvement of soft-decision maximum-likelihood decoding algorithm using hard-decision bounded-distance decoding," *IEEE Transactions on Information Theory*, vol. 43, no. 4, pp. 1314–1319, jul 1997.

[132] M. P. C. Fossorier and S. Lin, "Complementary reliability-based decodings of binary linear block codes," *IEEE Transactions on Information Theory*, vol. 43, no. 5, pp. 1667–1672, sep 1997.

[133] C. Choi and J. Jeong, "Fast and scalable soft decision decoding of linear block codes," *IEEE Communications Letters*, vol. 23, no. 10, pp. 1753–1756, jul 2019.

[134] H. Yamamoto and K. Itoh, "Viterbi decoding algorithm for convolutional codes with repeat request," *IEEE Transactions on Information Theory*, vol. 26, no. 5, pp. 540–547, sep 1980.

[135] G. Forney, "Exponential error bounds for erasure, list, and decision feedback schemes," *IEEE Transactions on Information Theory*, vol. 14, no. 2, pp. 206–220, mar 1968.

[136] J. Spinner, D. Rohweder, and J. Freudenberger, "Soft input decoder for high-rate generalised concatenated codes," *IET Circuits, Devices Systems*, vol. 12, no. 4, pp. 432–438, mar 2018.

[137] D. Schipani, M. Elia, and J. Rosenthal, "On the decoding complexity of cyclic codes up to the BCH bound," in *IEEE International Symposium on Information Theory (ISIT)*, jul 2011, pp. 835–839.

[138] M. Bossert, "An iterative hard and soft decision decoding algorithm for cyclic codes," in *12th International ITG Conference on Systems, Communications and Coding (SCC)*, feb 2019, pp. 1–6.

[139] ——, "On decoding using codewords of the dual code," *arXiv: 2001.02956*, jan 2020.

[140] A. Valembois and M. Fossorier, "Box and match techniques applied to soft-decision decoding," *IEEE Transactions on Information Theory*, vol. 50, no. 5, pp. 796–810, may 2004.

[141] W. Jin and M. Fossorier, "Efficient box and match algorithm for reliability-based soft-decision decoding of linear block codes," in *2007 Information Theory and Applications Workshop*, jan 2007, pp. 160–169.

[142] G. Forney, *Concatenated codes*. MIT Press Cambridge MA, 1966.

[143] F. MacWilliams and N. Sloane, *The theory of error correcting codes*, 12nd ed. Amsterdam: North-Holland, 2006.

[144] M. Bossert, R. Schulz, and S. Bitzer, "On information set decoding of BCH codes over binary symmetric channel," in *IEEE International Workshop on Mathematics and Coding, Guangzhou, China*, dec 2020.

[145] H. Helgert and R. Stinaff, "Shortened BCH codes," *IEEE Transactions on Information Theory*, vol. 19, no. 6, pp. 818–820, nov 1973.

[146] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Transactions on Information Theory*, vol. 42, no. 2, pp. 429–445, mar 1996.

[147] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Transactions on Information Theory*, jun 2009.

[148] I. Tal and A. Vardy, "How to construct polar codes," *IEEE Transactions on Information Theory*, vol. 59, no. 10, pp. 6562–6582, jul 2013.

[149] M. Bossert, *Channel coding for telecommunications*. Wiley, 1999.

[150] A. Neubauer, J. Freudenberger, and V. Kühn, *Coding theory*. John Wiley & Sons, Ltd, oct 2007.

[151] W. Drescher, *Schaltungsanordnung zur Galoisfeld-Arithmetik*. Dissertation TU-Dresden, 2005.

[152] J. Spinner, J. Freudenberger, C. Baumhof, A. Mehnert, and R. Willems, "A BCH decoding architecture with mixed parallelization degrees for flash controller applications," in *IEEE 26th International SOC Conference (SOCC), Erlangen*, sep 2013, pp. 116–121.

[153] J. Cho and W. Sung, "Strength-reduced parallel Chien search architecture for strong BCH codes," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 55, no. 5, pp. 427–431, may 2008.

[154] M. Zhang, F.Wu, C.Xie, Y. Zhou, and K. Zou, "A novel optimization algorithm for Chien search of BCH codes in NAND flash memory devices," in *2015 IEEE International Conference on Networking, Architecture and Storage (NAS)*, aug 2015, pp. 106–111.

[155] Y. Polyanskiy, H. V. Poor, and S. Verdu, "Channel coding rate in the finite blocklength regime," *IEEE Transactions on Information Theory*, vol. 56, no. 5, pp. 2307–2359, may 2010.

[156] D. Goldin and D. Burshtein, "Performance bounds of concatenated polar coding schemes," *IEEE Transactions on Information Theory*, vol. 65, no. 11, pp. 7131–7148, aug 2019.

[157] M.-O. Jeong and S.-N. Hong, "SC-fano decoding of polar codes," *IEEE Access*, vol. 7, pp. 81 682–81 690, jun 2019.

[158] M. Helmling, S. Scholl, F. Gensheimer, T. Dietz, K. Kraft, S. Ruzika, and N. Wehn, "Database of Channel Codes and ML Simulation Results," www.uni-kl.de/channel-codes, 2017.

[159] I. Tal and A. Vardy, "List decoding of polar codes," *IEEE Transactions on Information Theory*, vol. 61, no. 5, pp. 2213–2226, mar 2015.

[160] H. Song, J.-C. Fu, S.-J. Zeng, J. Sha, Z. Zhang, X. You, and C. Zhang, "Polar-coded forward error correction for MLC NAND flash memory," *Science China Information Sciences*, vol. 61, no. 10, p. 102307, aug 2018.

[161] L. Weiburn and J. Cavers, "Improved performance of Reed-Solomon decoding with the use of pilot signals for erasure generation," in *Vehicular Technology Conference, 1998. VTC 98. 48th IEEE*, vol. 3, may 1998, pp. 1930–1934.

[162] N. Wong, E. Liang, H. Wang, S. V. S. Ranganathan, and R. D. Wesel, "Decoding flash memory with progressive reads and independent vs. joint encoding of bits in a cell," in *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, dec 2019.

[163] S. Lee, D. Kim, and J. Ha, "A paired-page reading scheme for NAND flash memory," in *2016 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, oct 2016.

[164] J. Spinner and J. Freudenberger, "A decoder with soft decoding capability for high-rate generalized concatenated codes with applications in non-volatile flash memories," in *Proceedings of 30th Symposium on Integrated Circuits and Systems Design (SBCCI), Fortaleza, Brazil*, sep 2017.

[165] S. Li and T. Zhang, "Improving multi-level NAND flash memory storage reliability using concatenated BCH-TCM coding," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 10, pp. 1412–1420, oct 2010.

[166] S. Liu and X. Zou, "QLC NAND study and enhanced Gray coding methods for sixteen-level-based program algorithms," *Microelectronics Journal*, vol. 66, pp. 58–66, aug 2017.

[167] G. Caire, G. Taricco, and E. Biglieri, "Bit-interleaved coded modulation," *IEEE Transactions on Information Theory*, vol. 44, no. 3, pp. 927–946, may 1998.

[168] K. Haymaker and C. A. Kelley, "Structured bit-interleaved LDPC codes for MLC flash memory," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 5, pp. 870–879, may 2014.

[169] G. Forney and G. Ungerboeck, "Modulation and coding for linear Gaussian channels," *IEEE Transactions on Information Theory*, vol. 44, no. 6, pp. 2384–2415, oct 1998.

[170] Y. Bu, Y. Fang, G. Han, S. Mumtaz, and M. Guizani, "Design of protograph-LDPC-based BICM-ID for multi-level-cell (MLC) NAND flash memory," *IEEE Communications Letters*, vol. 23, no. 7, pp. 1127–1131, may 2019.

[171] G. Böcherer, F. Steiner, and P. Schulte, "Bandwidth efficient and rate-matched low-density parity-check coded modulation," *IEEE Transactions on Communications*, vol. 63, no. 12, pp. 4651–4665, oct 2015.

Publications of the Author

Journal Papers

Nicolas Bailon, D.; Shavgulidze, S & Freudenberger, J. **(2023)**, "Read Reference Voltage Adaptation for NAND Flash Memories With Neural Networks Based on Sparse Histograms", *IEEE Access*, vol. 11, pp. 56801-56811, doi: 10.1109/ACCESS.2023.3283445.

Nicolas Bailon, D.; Bossert, M.; Thiers, J.-P. & Freudenberger, J. **(2022)** "Concatenated Codes Based on the Plotkin Construction and Their Soft-Input Decoding", *IEEE Transactions on Communications*, vol. 70, no. 5, p. 2939-2950. doi: 10.1109/tcomm.2022.3158329.

Nicolas Bailon, D.; Thiers, J.-P. & Freudenberger, J. **(2022)** "Error Correction for TLC and QLC NAND Flash Memories Using Cell-Wise Encoding", *MDPI Electronics* vol. 11, no.10, p. 1585. doi: 10.3390/electronics11101585.

Kuhn, V; Nicolas Bailon, D.; Freudenberger, J. **(2022)** "Mutual Information Analysis for Generalized Spatial Modulation Systems With Multilevel Coding", *IEEE Open Journal of the Communications Society* vol. 3, p. 1222-1233. doi: OJCOMS.2022.3193068.

Thiers, J.-P.; Nicolas Bailon, D.; Freudenberger, J & Lu, J. **(2021)** "Read Reference Calibration and Tracking for Non-Volatile Flash Memories", *MDPI Electronics* vol. 10, no. 18, p. 2306. doi: electronics10182306.

Freudenberger, J.; Nicolas Bailon, D.; Safieh, M. **(2021)** "Reduced complexity hard- and soft-input BCH decoding with applications in concatenated codes", *IET Circuits, Devices & Systems* vol. 15, p. 284-296. doi: 10.1049/cds2.12026.

Conference Papers

Nicolas Bailon, D.; Shavgulidze, S & Freudenberger, J. **(2023)** "Features for Read Reference Voltage Estimation in NAND Flash Memory", *2023 IEEE 13th International Conference on Consumer Electronics (ICCE-Berlin)*. doi: 10.1109/ICCE-Berlin58801.2023.10375586.

Nicolas Bailon, D., Kuehn, V; Shavgulidze, S & Freudenberger, J. **(2023)** "Estimating Mutual Information for Link Adaptation in Generalized Spatial Modulation Systems with Neural Networks", *WSA & SCC 2023; 26th International ITG Workshop on Smart Antennas and 13th Conference on Systems, Communications, and Coding*, vol. 70, no. 5, p. 2939-2950. ISBN: 978-3-8007-6050-3.

Nicolas Bailon, D.; Shavgulidze, S & Freudenberger, J. **(2022)** "Cell-wise encoding and decoding for TLC flash memories", *2022 IEEE 12th International Conference on Consumer Electronics (ICCE-Berlin)*. doi: 10.1109/ICCE-Berlin56473.2022.9937136.

Nicolas Bailon, D.; Taburet, G; Shavgulidze, S & Freudenberger, J. **(2022)** "Neural network aided reference voltage adaptation for NAND flash memory", *2022 IEEE 12th International Conference on Consumer Electronics (ICCE-Berlin)*. doi: 10.1109/ICCE-Berlin56473.2022.9937118.

Nicolas Bailon, D.; Freudenberger, J. & Kuehn, V. **(2021)** "Constant-Weight Convolutional Codes for Index Modulation", *WSA 2021; 25th International ITG Workshop on Smart Antennas*, pp. 1-6. ISBN: 978-3-8007-5686-5.

Nicolas Bailon, D. & Freudenberger, J. **(2021)** "List decoding for concatenated codes based on the Plotkin construction with BCH component codes", **(2021)** *2021 11th International Symposium on Topics in Coding (ISTC)*, pp. 1-5.
doi: 10.1109/ISTC49272.2021.9594194.

Safieh, M.; Nicolas Bailon, D. & Freudenberger, J. **(2020)**"An Acceptance Criterion for Hybrid Algebraic and Soft-input Decoding", *2020 24th International Conference on Information Technology (IT)*, pp. 1-5. doi: 10.1109/IT48810.2020.9070575.

Thiers, J.-P.; Nicolas Bailon, D. & Freudenberger, J. **(2020)** "Bit-Labeling and Page Capacities of TLC Non-Volatile Flash Memories", *2020 IEEE 10th International Conference on Consumer Electronics (ICCE-Berlin)*, pp. 1-6.
doi: 10.1109/ICCE-Berlin50680.2020.9352190.

Nicolas Bailon, D.; Thiers, J.-P. & Freudenberger, J. **(2020)**"Soft-input decoding of concatenated codes based on the Plotkin construction and BCH component codes", *2020 IEEE 10th International Conference on Consumer Electronics (ICCE-Berlin)*, pp. 1-5. doi: 10.1109/ICCE-Berlin50680.2020.9352179.

Curriculum Vitæ

Zur Person

Nachname	Nicolas Bailon
Vorname	Daniel
Geburtsjahr/ -ort	1992 in Singen am Hohentwiel (Deutschland)

Bildungsgang

03.2020 – 06.2024	Doktorand (Dr.-Ing.) an der Universität Ulm
03.2017 – 02.2019	Masterstudium: Elektrische Systeme (M. Eng.), HTWG Konstanz
09.2013 – 02.2017	Bachelorstudium: Elektro- und Informationstechnik (B. Eng.), HTWG Konstanz
09.2011 – 07.2013	Allgemeine Hochschulreife, Technische Oberschule an der HGS Singen
09.2008 – 07.2011	Ausbildung zum Elektroniker für Geräte und Systeme, *bei ELGO Electronic GmbH & Co. KG*

Beruflicher Werdegang

Seit 11.2023	Software-Testingenieur ZF Group
07.2019 – 10.2023	Wissenschaftlicher Mitarbeiter am Institut für Systemdynamik HTWG Konstanz
03.2019 – 07.2019	Forschungsassistent *bei Hyperstone GmbH*
09.2015 – 01.2016	Praktisches Studiensemester *bei in-tech GmbH*