UNIVERSITY OF CALIFORNIA

Los Angeles

Optimizing Flash-Based Storage Systems

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy
in Electrical and Computer Engineering

by

Haobo Wang

2018

ABSTRACT OF THE DISSERTATION

Optimizing Flash-Based Storage Systems

by

Haobo Wang

Doctor of Philosophy in Electrical and Computer Engineering

University of California, Los Angeles, 2018

Professor Richard D. Wesel, Chair

This dissertation proposes mathematical algorithms for improving Flash-based storage system's four key performance metrics: lifetime, reliability, latency and throughput.

The first part of the dissertation presents the novel concept of dynamically voltage allocation (DVA) for Flash memory. Flash memory suffers reduced reliability as the number of program/erase (P/E) cycles increases, thus has a limited lifetime. DVA scales the write threshold voltages of Flash memory adaptively, using lower voltages at the beginning of the lifetime, and gradually increases the scaling to combat the effect of accumulated wear-out from P/E cycling. The proposed algorithm significantly increases the lifetime of the device.

The second part of the dissertation introduces the novel design of error correction using incremental redundancy without feedback. Modern storage systems often require high throughput, high reliability and low latency. Traditional variable-length (VL) codes with feedback have demonstrated to provide high throughput and reliability. The new design reinterprets the results for VL codes with feedback using ergodicity, by encoding the incremental redundancy of multiple VL codewords to a common pool of redundancy. The removal of feedback allows storage systems to benefit from the performance of a feedback scheme with a feedforward design. The decoder of the new design exploits the low complexity of short-blocklength decoders and the parallelization structure to reduce latency. The proposed error correction scheme approaches the throughput of corresponding VL codes with feedback.

Relying on information theory and coding theory, the proposed algorithms provide new approaches to optimize the Flash-based storage systems.

The dissertation of Haobo Wang is approved.

Adnan Youssef Darwiche

Babak Daneshrad

Lieven Vandenberghe

Richard D. Wesel, Committee Chair

University of California, Los Angeles

2018

To my parents.

Table of Contents

1 Overview **1**

2 Using Dynamic Allocation of Write Voltage to Extend Flash Memory Lifetime **5**

2.1 Introduction 5

2.2 Modeling Channel Parameters & Degradation 9

2.2.1 Channel Model with Additive Components 10

2.2.2 Channel Parameter Degradation Model 15

2.2.3 Models Used in This Chapter 18

2.3 Dynamic Voltage Allocation with Ideal Channel Information 19

2.4 Channel Parameter Estimation 26

2.4.1 Channel Parameter Estimation Problem Formation 27

2.4.2 Least Squares Algorithms 28

2.4.3 Binning Strategy 31

2.5 Dynamic Voltage Allocation with Model/Channel Mismatch 36

2.5.1 Model 1 37

2.5.2 Model 2 39

2.6 Comparison of Dynamic Voltage Allocation and Dynamic Threshold Assignment 40

2.7 Dynamic Voltage Allocation when Voltage Placements are Quantized 42

2.8 Complexity Analysis 44

2.9 Conclusion 47

2.10 Channel Parameters Used in This Chapter 48
2.11 Acknowledgment 48

3 Coding with Shared Incremental Redundancy 49
3.1 Introduction 49
3.2 Architecture 51
3.3 Generalized Peeling Decoder Analysis 54
3.3.1 Generalized Peeling Decoder (GPD) 55
3.3.2 Computation of $r_1(t)$ a.k.a. $r_1(x)$ 56
3.3.3 Computing Probability of Left Node Failure ϵ_{FF} 60
3.3.4 Throughput of Feedback and Feedback-Free Systems 61
3.3.5 Throughput Loss in the Feedback-Free System 62
3.4 Design Methods 64
3.4.1 VL Code Design 64
3.4.2 Design the Inter-Frame Degree Distribution 72
3.4.3 Bipartite Graph Design 75
3.5 Results 76
3.5.1 Convolutional VL Code Based System 76
3.5.2 NB-LDPC VL Code based System 80
3.6 Conclusion 85
3.7 Acknowledgment 86

References 87

List of Figures

1.1 Common structure of a P-well Flash memory cell. 2

1.2 NAND Flash memory cell grid. 3

2.1 Estimation-based DVA framework. 7

2.2 Flash read channel PDFs illustrating how the probability density of voltage thresholds is affected by various noise components. 11

2.3 Spacial relationship between the interfering cells (red) and interfered cells (green). Flash memory is written wordline by wordline, i.e., wordline $i+1$ is written after wordline i. For the interference shown above, within a wordline even cells are written before odd cells. 13

2.4 DVA performance with ideal channel information versus fixed allocation's performance. (Ground truth channel is Model 2. DVA targets even channel. Channel parameters are listed in the Sec. 2.10.) 23

2.5 DVA performance with ideal channel information and the performance of the fixed allocation. (The ground truth channel is Model 2. The writing order of pages is switched every 100 P/E cycles. Joint DVA is use to adjust both scaling factors for even and odd cells. Channel parameters are listed in Sec. 2.10.) 24

2.6 Scaling factors generated by DVA with ideal channel information. (Result corresponds to Fig. 2.4 and 2.5. Ground truth model is Model 2. Channel parameters are listed in Sec. 2.10. The starting value of α is 0.37.) 25

2.7 Estimation result versus ground truth for γ_{μ_r} using 10-bin equal-probability histogram. 30
2.8 Iteration count statistics using Levenberg-Marquardt algorithm. 32
2.9 Squared Euclidean distance between the channel distributions and corresponding histograms (10 bins). 34
2.10 Effective resolution of different histograms (10 bins). 35
2.11 DVA's performance with multi-modal Gaussian model. (Ground truth model is Model 1. Channel parameters are listed in Sec. 2.10.) 37
2.12 DVA performance with multi-modal Gaussian model. (The Ground truth model is Model 2. Channel parameters are listed in Sec. 2.10. The lifetime of even cells is extended by 70.5% from 2136 P/E cycles to 3642 P/E cycles. The lifetime of odd cells is extended by 41.6% from 2517 P/E cycles to 3564 P/E cycles. The overall lifetime is extended by 66.9% from 2136 P/E cycles to 3564 P/E cycles.) 38
2.13 DTA and DVA's performance. (Even and odd cells switch positions when using DVA. The legend follows the format *write voltage allocation algorithm - read voltage allocation algorithm*. Comparing the result from DVA - DTA with Fixed - DTA, The lifetime of even cells is extended by 66.5% from 2282 P/E cycles to 3800 P/E cycles. The lifetime of odd cells is extended by 41.4% from 2654 P/E cycles to 3754 P/E cycles. The overall lifetime is extended by 64.5% from 2282 P/E cycles to 3754 P/E cycles.) 41
2.14 DVA performance with quantized placements. (Quantization provides 64 possible locations. Ground truth model is Model 2. Channel parameters are listed in Sec. 2.10.) 42

2.15 DVA's performance with quantized placements. (Quantization provides 128 possible locations. Ground truth model is Model 2. Channel parameters are listed in Sec. 2.10. The lifetime of even cells is extended by 72.1% from 2136 P/E cycles to 3676 P/E cycles. The lifetime of odd cells is extended by 34.8% from 2517 P/E cycles to 3393 P/E cycles. The overall lifetime is extended by 58.8% from 2136 P/E cycles to 3393 P/E cycles.) 43

3.1 Inter-frame encoder structure. $\boldsymbol{W}_i$ is a k-bit message, $\boldsymbol{X}_0^{(i)}$ is a length-ℓ_0 vector, and $\{\boldsymbol{X}_1^{(i)}, \ldots, \boldsymbol{X}_4^{(i)}\}$ and $\boldsymbol{I}_j$ are length-ℓ_Δ vectors. 52

3.2 Decoding process of an inter-frame coding system with the inter-frame layer represented as a bipartite graph. 53

3.3 Example $r_1(x)$ from density evolution and equation (3.8) for $\lambda(x) = x^3$ and the irregular $\rho_{PEG}(x)$ in Section 3.5.2. The VL code characterization PMF $\boldsymbol{\delta} = [0.30909, 0.464, 0.1939, 0.02934, 0.00318, 0.00049]$. 60

3.4 Probability of failure mechanisms of right nodes for the irregular $\rho_{PEG}(x)$ in Section 3.5.2. 63

3.5 Example right degree distribution $\rho_{DE}(x)$ from differential evolution as described in Sec. 3.5.2. 74

3.6 Probability of failure versus the number of iterations of the convolutional VL Code based feedback-free system. 78

3.7 $r_1(x)$ versus x for Table 3.5. The curves are generated using equation (3.8). Circles indicate iteration points determined through density evolution. 79

3.8 Probability of failure versus iterations for $\rho_{QR}(x)$ distributions in Table 3.5. The curves are generated using density evolution. 79

3.9 Probability of failure versus the number of iterations for the designs in Table 3.7 from density evolution and genie-aided simulations. 83

3.10 Probability of failure versus the number of iterations for the $\alpha = 0.597$ 1000-left-node design in Table 3.8 from density evolution, simulations with a genie-aided decoder and decoders using CRCs. 84

List of Tables

2.1 Converge counts of least square algorithms (over 14 cases). 31

2.2 Computational complexity of read threshold voltage allocation. 45

2.3 Computational complexity of Levenberg-Marquardt algorithm. 45

2.4 Computational complexity of scaling factor adjustment. 46

2.5 Channel parameters used in this chapter. 48

3.1 Performance comparison between the $k = 64$ 1024-state TBCC VL code's VI and CI designs. (2 dB BI-AWGN, $m = 5$, target $\epsilon_{FB} = 10^{-3}$ for the first two rows, target $\epsilon_{FB} = 5 \times 10^{-4}$ for the third row) 67

3.2 The first 5 combinations for the first symbol of the rate 0.75 NB-LDPC code. 71

3.3 Performance comparison between the $K = 24$ NB-LDPC VL code with ACK/NACK feedback's VI and CI designs. (2dB BI-AWGN, $m = 5$, target $\epsilon_{FB} = 10^{-3}$ for the first two rows, target $\epsilon_{FB} = 5 \times 10^{-4}$ for the third row) 71

3.4 Performance of the convolutional VL code based feedback-free systems. . . . 77

3.5 Density evolution performance characteristics of quasi-regular $\rho_{QR}(x) = \alpha x^2 + (1-\alpha)x^3$ for the convolutional VL code based feedback-free system. $\lambda(x) = x^3$ in all cases. 79

3.6 Density evolution performance characteristics of the highest $R_t^{(FF)}$ (lowest-β_{FF}) quasi-regular right degree distributions for the convolutional VL code based feedback-free system. $\lambda(x) = x^3$ in all cases. 80

3.7 Density evolution performance characterization of quasi-regular $\rho_{QR}(x) = \alpha x^2 + (1-\alpha)x^3$ for the NB-LDPC VL code based feedback-free system. $\lambda(x) = x^3$ in all cases. 82

3.8 Right degree distributions of the PEG generated inter-frame bipartite graphs in Fig. 3.9. 82

3.9 Probability of failure of 1000-left-node $\alpha = 0.597$ code from Table 3.8 simulated with CRCs. 84

Acknowledgments

This dissertation summarizes my multi-year research effort as a graduate student at University of California, Los Angeles (UCLA). This long march would not have been possible without the support and guidance from my family, friends, colleagues and advisors.

I would like to express my deepest gratitude to my advisor and chairman of my doctoral committee Prof. Richard D. Wesel. I first met Prof. Wesel as a student attending his channel coding course. It is through his deep insights and excellent teaching that I discovered the importance of information theory and coding theory in today's information technology landscape. Through his encouragement, I joined his research group to work on important theoretical challenges that have many practical manifestations. His deep knowledge on a vast amount of subjects and tireless guidance enabled my transition from a student to a researcher. From resolving simple logistics issues in the research group, to honing my writing and presenting skills, to teaching me creative ways to identify and solving theoretical problems, his efforts made my doctoral tenure not only productive, but also fun. His deep connections in the industry provided me with opportunities to gain first-hand experience in solving real engineering problems, which also contributed to my research efforts. As a researcher, as a teacher, and as a friend, Prof. Wesel demonstrated what an exceptional advisor should be. I am honored to have the chance to work with him.

I would also like to thank Prof. Dariush Divsalar. I benefited greatly from his immense experience and insights in both theory and engineering. I am greatly thankful to the rest of my committee members: Prof. Adnan Youssef Darwiche, Prof. Babak Deneshrad and Prof. Lieven Vandenberghe, for their contribution in this dissertation and my doctoral tenure.

As a member of UCLA Communications Systems Laboratory (CSL), I learned greatly from my colleagues in the research group that have worked with me: Dr. Kasra Vakilinia, Dr. Sudarsan V.S. Ranganathan, Nathan Wong, Alexander M. Baldauf, Christopher K. Bachelor, Adam Belhouchat, and Ethan Liang. As the senior student in the group for the

majority of my doctoral tenure, Kasra helped me gain a foothold, and showed to me how to be an exceptional doctoral student. Working with Nathan has always been productive and fun. Alexander, Christopher, Adam and Ethan amazed me how much they can achieve through research as undergraduate students. Especially, I want to thank Sudarsan. Sudarsan and I joined the group at around the same time. I deeply cherish my discussions and arguments with him ranging from research problems to politics. His hardworking attitude and passion towards both research and life will always inspire me in my future endeavors. I also want to thank my colleagues that I did not get a chance to work with: Dr. Adam Williamson, Dr. Chung-Yu Lou, Tong Mu, Chris Miller, Gourav Khadge, Will Chuang, Hengjie Wang and Linfang Wang.

Outside of UCLA, I want to thank Mr. Aldo Cometti at Western Digital. I gained valuable engineering-problem-solving skills through my internship under the guidance of Mr. Cometti. I am grateful for all the time and effort he put into my internship.

Most importantly, I want to dedicate this dissertation to my parents. It is your love, support and great sacrifice that have lifted my life trajectory to this day. I would not have achieved so much without you.

Finally, I want to thank UCLA for providing the institutional support for my doctoral tenure. I express great appreciation to UCLA and National Science Foundation for their financial support.

Vita

2014 Master of Science, Electrical Engineering
University of California, Los Angeles

2015 PhD Candidate, Electrical Engineering
University of California, Los Angeles

Selected Publications

H. Wang, T. Chen, and R. D. Wesel, "Histogram-based Flash channel estimation", in *Proceedings of 2015 IEEE International Conference on Communications (ICC)*, London, UK, June 2015, pp. 283–288.

H. Wang, N. Wong, and R. D. Wesel, "Dynamic voltage allocation with quantized voltage levels and simplified channel modeling",in *Proceedings of 2015 49th Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, November 2015, pp. 834–838.

H. Wang, N. Wong, T. Chen, and R. D. Wesel, "Using dynamic allocation of write voltage to extend Flash memory lifetime", *IEEE Transactions on Communications*, vol. 64, no. 11, pp. 4474–4486, November 2016.

H. Wang, N. Wong, A. M. Baldauf, C. K. Bachelor, S. V. S. Ranganathan, D. Divsalar and R. D. Wesel, "An information density approach to analyzing and optimizing incremental redundancy with feedback", in *Proceedings of 2017 IEEE International Symposium on Information Theory (ISIT)*, Aachen, Germany, June 2017, pp. 261–265.

H. Wang, S. V. S. Ranganathan and R. D. Wesel, "Approaching capacity using incremental redundancy without feedback", in *Proceedings of 2017 IEEE International Symposium on Information Theory (ISIT)*, Aachen, Germany, June 2017, pp. 161–165.

H. Wang and R. D. Wesel, "Channel code analysis and design using multiple variable-length in parallel without feedback", in *Proceedings of 2018 IEEE Global Communications Conference (GLOBECOM)*, Abu Dhabi, UAE, December 2018, to be published.

H. Wang and R. D. Wesel, "Coding with shared incremental redundancy: design methods and a non-binary LDPC example", *IEEE Transactions on Communications*, September 2018, submitted for publication.

CHAPTER 1

Overview

A data storage system usually consists of two components: the storage media and its controller. The storage media is the physical device that stores data, such as optical disks, magnetic disks, and Flash memory cells. The controller is usually the software and hardware implementation of logics that manages the data storage and retrieval process on the storage media, and provides an interface to store and retrieve data to other systems. This dissertation proposes two types of algorithms for the controller which improves the performance of Flash-based storage systems.

The performance of a storage system is usually measured with four metrics: lifetime, reliability, latency and throughput. Lifetime is measured as the amount of time from the point that data is initially stored to the point that the stored data is not be recoverable. Reliability is measured as the probability that data retrieved from the system is not the same as the originally stored data. Latency is measured as the amount of time the system takes from issuing the data retrieval/storage command to the completion of the action. Throughput is measured as the average amount of data the system can store and retrieve in an unit amount of time. An ideal storage system should have a long lifetime, a high reliability, a low latency and a high throughput. However, the fundamental physical characteristics of the storage media dictates the baseline performance metrics of the storage system. Also, physical limitations constrain the ability to improve all four metrics simultaneously. So the controller of practical storage systems need to balance the four metrics, and depending on

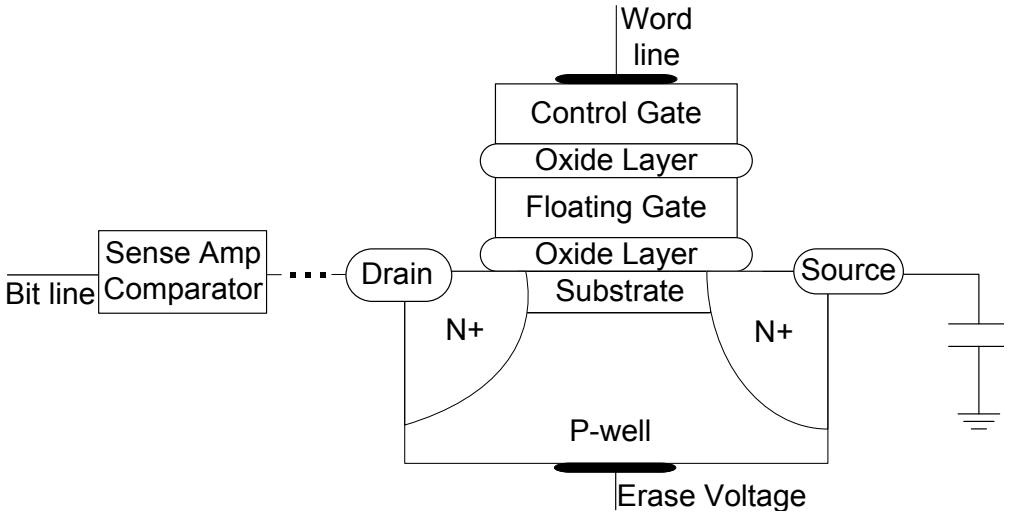


Figure 1.1: Common structure of a P-well Flash memory cell.

application, strengthen certain performance metrics at a cost of others.

For a Flash-memory-based storage system, the storage media, i.e. the Flash memory cells, have unique physical characteristics [1, 2, 3, 4]. Fig. 1.1 shows the basic physical structure of a P-well Flash memory cell. The threshold voltage of a Flash cell is largely determined by the amount of charge in the floating gate and the device's intrinsic (charge-neutral) threshold voltage. Electrons are programmed to and erased from the floating gate through the oxide layer to control the threshold voltage. This program and erase (P/E) operation is used to storage data. The threshold voltage level can be obtained by applying a wordline voltage and reading the sense amplifier output to determine if the threshold voltage has been surpassed. This process can be modeled as a point-to-point communication system with a discrete memoryless channel. Due to different types of channel degradation, the measured value of the threshold voltage often differs from the originally stored threshold voltage.

The Flash memory cells are arranged in a dense grid in a Flash memory system to form the basic storage media [1, 3, 4]. Fig. 1.2 shows the cell grid of a typical NAND Flash memory. Each row of cells are connected to the same wordline, and each column of cells are connected head-to-toe through the bitline. Program and erase operation on each row

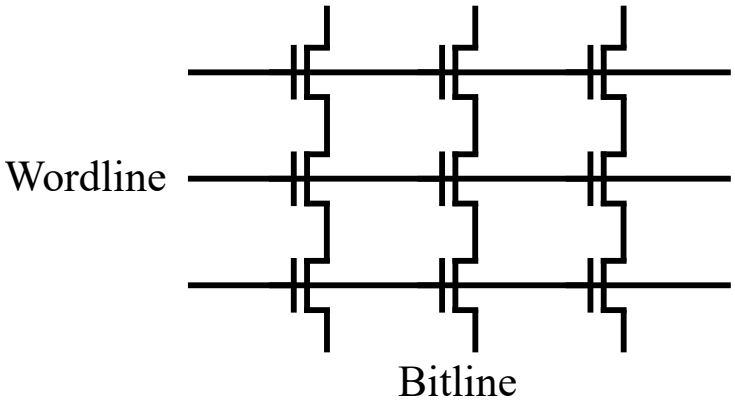


Figure 1.2: NAND Flash memory cell grid.

or column of cells will impact the stored charge in adjacent rows and columns, and cause additional channel degradation.

In Chapter 2, the concept of dynamically voltage allocation (DVA) is presented as a novel method to combat Flash memory channel degradation. This chapter first introduces Flash channel degradation mechanisms and their corresponding models, and provides a technique to estimate the parameters in the models. From the models it can be concluded that the reliability of Flash memory suffers as the number of P/E cycles increases, thus has a limited lifetime. Then the chapter presents the DVA framework which scales the write threshold voltages of Flash memory adaptively, using lower voltages at the beginning of the lifetime, and gradually increases the scaling to combat the effect of accumulated degradation from P/E cycling. Simulation results show that DVA can significantly increase the lifetime of the system.

Through DVA and other methods, the effect of channel degradation can be greatly mitigated. However, certain amount of degradation will still manifest, causing the retrieved data to contain errors. Error correction code (ECC) is usually used to correct these errors. The ECC algorithm encodes the data to be stored into codewords which contain redundant information, and recovers the stored data using retrieved codewords. The design of the ECC has significant impact on the lifetime, reliability, latency and throughput of the storage system.

In Chapter 3, the design of error correction using shared incremental redundancy is intro-

duced as a novel ECC solution for Flash-based storage systems. Traditional variable-length (VL) codes with feedback have demonstrated to provide high throughput and reliability. The new design reinterprets the results for VL codes with feedback using ergodicity, by encoding the incremental redundancy of multiple VL codewords to a common pool of redundancy [5, 6, 7]. This chapter first introduces the inter-frame coding as the structure of the proposed design. The parallel structure and low complexity of the component short-blocklength codes in our encoder and decoder design can help improve latency. Then the chapter provides a convergence analysis of the proposed generalized peeling decoder for the system. The chapter also provides design demonstrations employing two different types of VL codes. Simulation results show that our designs without feedback approach the throughput and error performance of the VL codes with feedback, which can lead to improved throughput, lifetime and reliability.

CHAPTER 2

Using Dynamic Allocation of Write Voltage to Extend Flash Memory Lifetime

2.1 Introduction

Flash memory has been widely employed in both consumer electronic devices and industrial electronic systems because of its ability to support high-throughput and low-latency memory access. However, a fundamental issue with Flash technology is that its read channel experiences significant degradation over time which eventually produces unacceptable reliability. As modern Flash solutions provide more storage capacity in smaller form factors, the resulting increase of physical cell density and signal constellation density amplifies the degradation problem.

The degradation can be addressed in different layers in the Flash system. At the device layer three-dimensional cell structures improve durability [8, 9, 10]. At the system level channel codes such as Bose-Chaudhuri-Hocquenghem (BCH) codes [11, 12, 13] and more recently low density parity check (LDPC) codes [14, 15, 16, 17] add redundancy to protect the stored information. In [18], the authors write to cells with a lower voltage but for a longer time to cause less damage at the expense of increased write time. The resulting scheme provides lifetime extension while still guaranteeing a desired write throughput.

The degradation over time is often modeled as a function of the number of program and erase (P/E) cycles, so a direct solution is reducing the number of P/E cycles needed. Write-

once memory (WOM) codes [19, 20, 21, 22] provide one approach to reduce the number of P/E cycles required to store information by permitting multiple writes before an erase cycle is needed. Another way to reduce the number of P/E cycles is rank modulation [23, 24, 25], which stores information in the cell using the relative value (or ordering) of cell charge levels rather than the absolute value. Thus a block of cells can be rewritten without erasing by adding charge to properly re-order the cells.

Several papers have explored dynamically adjusting to the degrading read channel [26, 27, 28]. In [26], read thresholds are progressively adjusted to minimize hard decoding BER or provide better log-likelihood for soft decoding based on previous reads. In [27, 28], dynamic threshold assignment (DTA) adjusts the read thresholds to match the shifting and widening threshold voltage distributions of the read channel, significantly improving bit error rate (BER) performance.

In contrast to the P/E-cycle-based degradation model, this chapter models the degradation as a function of the cumulative effect of the charge written and erased from the cell, which we call the accumulated voltage V_{acc}. The accumulated voltage model reveals the opportunity to improve lifetime by minimizing the V_{acc} required to store a given amount of information. In particular, this chapter explores dynamically adjusting the target threshold voltage levels by using lower target write threshold voltage levels at the beginning of the device lifetime. This approach is called dynamic voltage allocation (DVA) [29, 2, 3, 4]. As the read channel becomes more degraded, the threshold voltages are gradually increased to what would be the nominal values in a standard device not employing DVA.

Fig. 2.1 illustrates the basic structure of the DVA approach. Periodically (every 100 P/E cycles in the figure) multiple reads using different read thresholds produce a histogram of the threshold voltages of the cells on the page or pages considered. A parameter-based least-squares channel estimation determines the quality of the read channel from this histogram. Based on the channel estimation, write threshold voltages are set to be as low as possible while still ensuring that the read channel has sufficient mutual information to be successfully

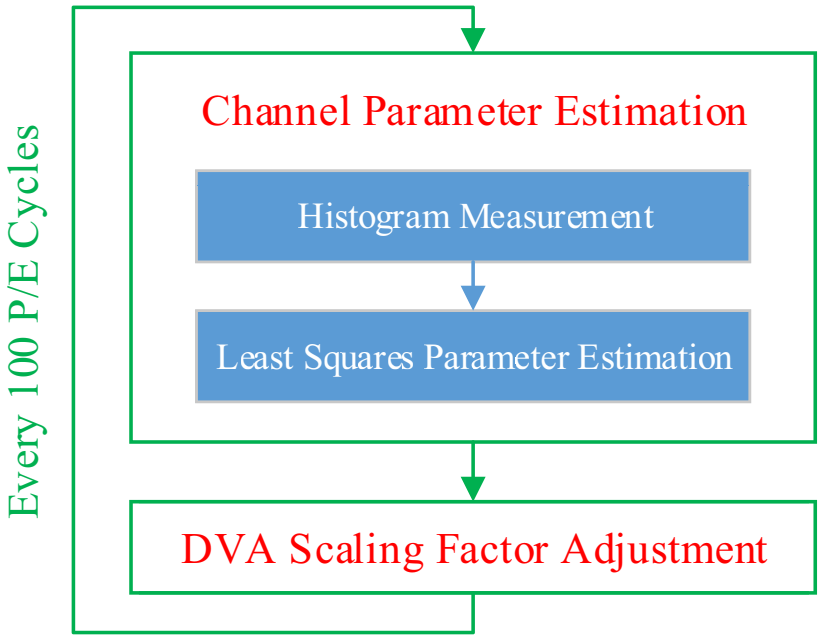


Figure 2.1: Estimation-based DVA framework.

decoded.

This chapter systematically explains and demonstrates the concept of DVA, and addresses several issues to support a practical implementation in the following ways:

1. Providing a comprehensive summary of Flash memory read channel degradation models;

2. Introducing the histogram-based Flash channel estimation as a method to estimate channel information in a practical system;

3. Summarizing the DVA algorithm, and analyzing DVA's performance with perfect channel information;

4. Analyzing DVA's performance when both estimation and scale-factor adjustment algorithms are simplified by using a simple Gaussian model to approximate the underlying (and more complex) ground truth channel;

5. Designing and analyzing DVA specialized to match an even-odd structure for writing to cells;

6. Exploring the improvement obtained by DVA over DTA;

7. Analyzing DVA's performance when both write and read voltage values are restricted to a finite set of available voltages;

8. Analyzing DVA's complexity.

DVA is first explored under the assumption of perfect knowledge of the channel state as in the paper [29], except with a more complex read channel that includes cell-to-cell interference and programming errors. This chapter introduces DVA in an idealized setting and then removes ideal assumptions about channel knowledge and threshold resolution to conclude with a practical scheme that has performance similar to that of the idealized setting.

Then the performance of DVA is explored when it must gain its channel information through estimation. All dynamic schemes such as DTA and DVA require some form of information about the read channel, and different methods can be employed to acquire the information. For DTA, knowledge of the voltage distribution is needed. In [27], repetitive read operations are needed at relatively precise voltages to enable a bisection algorithm to place the read thresholds. In [28], threshold measurements of a certain number of cells are required.

As shown in Fig. 2.1, our approach to acquire read channel information is to measure a limited-resolution histogram and interpret this histogram using certain assumptions about the channel model. In [30], the authors model the Flash read channel as a multi-modal Gaussian distribution with means and variances as parameters, and demonstrate that least squares algorithms estimate the means and variances well with histograms having as few as twelve bins. These estimated parameters are used to set read thresholds according to [31].

This channel estimation is first performed with a perfect channel model. Next, this chapter explores the practical scenario in which the channel model does not perfectly represent the true channel. This mismatch is both a reflection of the imperfect characterization information available about a specific Flash device and the fact that, even with perfect knowledge

of the channel model, simpler models might be preferable because they reduce the complexity of estimation.

Next, we explore how constraining the resolution of write and read thresholds affects performance. We also compare the performance of DVA to DTA and analyze complexity of the DVA framework. Possible solutions to reduce the cost of implementing DVA are proposed. Throughout the chapter, Multi-level Cell (MLC) Flash (with four levels) is assumed for all the models and simulations.

The remainder of this chapter is organized as follows: Sec. 2.2 presents the complete channel model. Sec. 2.3 introduces DVA using the complete channel model but in the idealized setting of perfect channel state information. Sec. 2.4 formulates the channel parameter estimation problem and presents the least squares algorithm and binning strategy used in this chapter. Sec. 2.5 examines the practical scenario in which channel estimation and DVA are based on a model that is not perfect but is simpler than the actual channel. Sec. 2.6 compares the performance of DTA and DVA. Sec. 2.7 adds practical constraints on the resolution of the read and write threshold voltages. Sec. 2.8 analyzes the complexity cost of DVA framework. Sec. 2.9 concludes the discussion. Sec. 2.10 provides the channel parameters used in this chapter. Sec. 2.11 presents the acknowledgment.

2.2 Modeling Channel Parameters & Degradation

Because the interfaces are proprietary, we are not able to measure data from actual flash devices. Instead, we use the models introduced in this section to generate the noise that reflects the behavior of the Flash memory read channel. These models are called *ground truth models*, which means that they are used for all simulations. The term "ground truth model" distinguishes these models from less precise simple Gaussian model (which is called the *channel model assumption*) that are used by the channel estimation and DVA algorithms. Note that all algorithms, regardless of the models they incorporate in their calculations, are simulated on the ground truth models.

The ground truth models are not matched to a particular Flash device, but based on the academic publications we cite as we present the models. We believe these models reflect the major channel degradations and provide a reasonable degradation trajectory over the lifetime of Flash memory. We use these qualitatively correct channel models to show that DVA can counter the major types of degradation in Flash memory channels. The models in this section can be replaced with device-specific models to apply DVA to a particular Flash memory system. In fact, we will show in Sec. 2.5 that DVA does not need a precise channel model to provide a significant improvement in lifetime.

2.2.1 Channel Model with Additive Components

We formulate a Flash memory read channel model with five additive noise components as follows:

$$y = x + n_{pe} + n_p + n_w + n_{c2c} + n_r, \tag{2.1}$$

where voltage x is the intended threshold voltage written to a cell, and y is the measured threshold voltage. Noise n_p denotes the programming noise, n_w denotes the wear-out noise, n_r denotes the retention noise, n_{c2c} denotes the cell-to-cell interference, and n_{pe} denotes the programming error.

Fig. 2.2 shows an example of voltage distribution probability distribution functions (PDFs) which demonstrates the additive effect of each noise component. The arrows in the figure represent delta functions.

Programming Error n_{pe} [4, 32]

Programming errors occur when a bit of the lower page is mis-read in preparation for writing a bit of the upper page to an MLC cell. Essentially, as the bit of the upper page is written, the error of the first bit is amplified. The programming error is modeled with a probability mass function (PMF). For an intended level x, channel parameter $P_{x,y} = P(Y = y|X = x)$

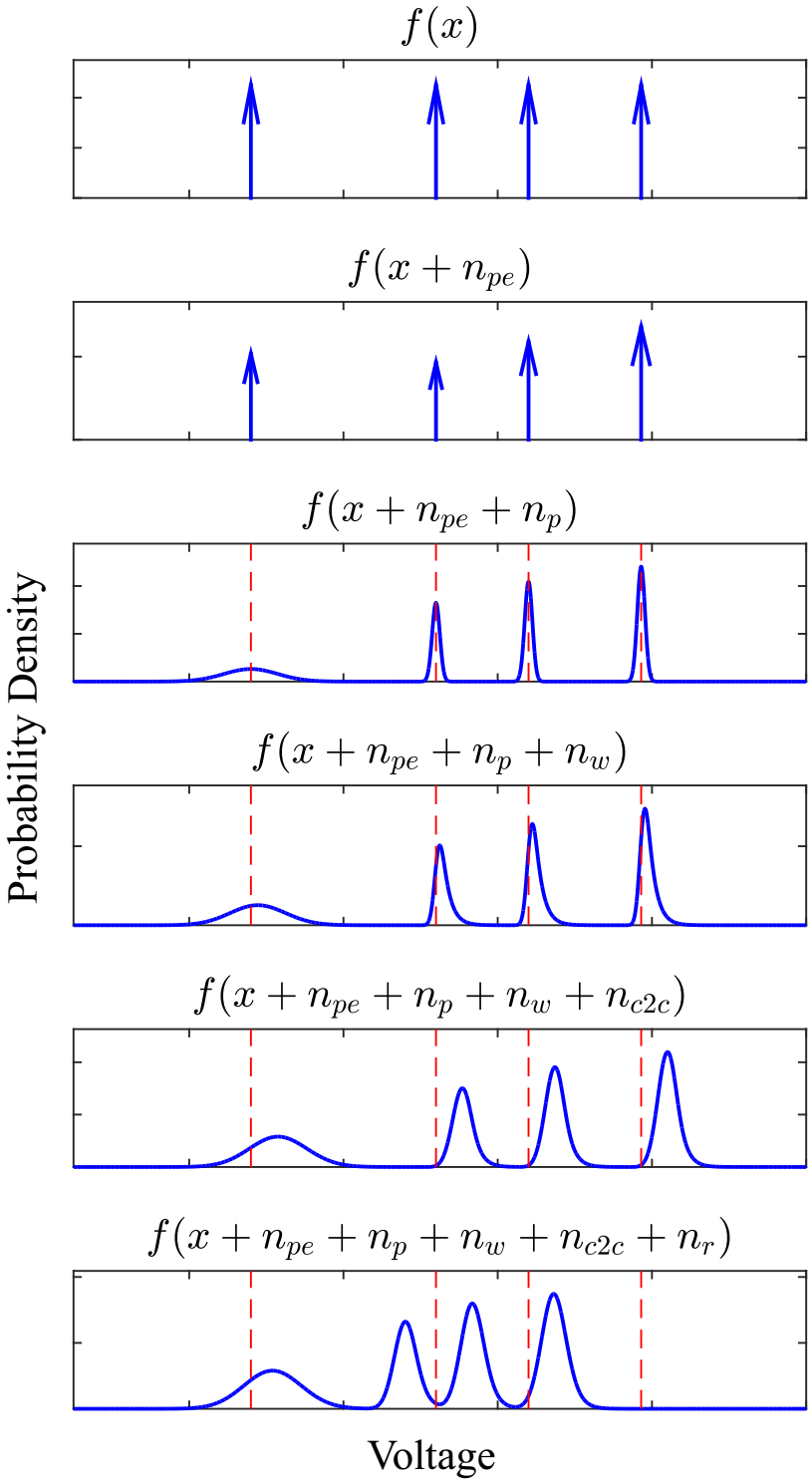


Figure 2.2: Flash read channel PDFs illustrating how the probability density of voltage thresholds is affected by various noise components.

is the conditional probability of actually writing y. For MLC Flash, we have

$$P_{x,0} + P_{x,1} + P_{x,2} + P_{x,3} = 1, \tag{2.2}$$

for each $x \in \{0, 1, 2, 3\}$. These channel parameters are strongly related to the number of P/E cycles.

This distortion changes the distribution of stored data. For example, if there is no other noise component and the original data is uniformly distributed (i.e. each level is equally likely), programming errors move some writes to higher levels so that the levels are no longer equally likely. Results in [32] indicate that the impact of programming errors becomes significant only after a large number of P/E cycles.

Programming Noise n_p [29, 2, 3, 4, 33, 34]

The programming noise is modeled as a Gaussian noise for each level, and the noise variance of programmed states is smaller than that of the erased state because the feedback control loop associated with programming reduces threshold variation. The PDF of the programming noise is represented as

$$f_{n_p}(n_p|x = l) = \begin{cases} \mathcal{N}(0, \sigma_e^2) & \text{if } l = 0 \\ \mathcal{N}(0, \sigma_p^2) & \text{if } l > 0 \end{cases}, \tag{2.3}$$

where $\sigma_e > \sigma_p$. Index l represents the level of the intended threshold voltage level. For MLC Flash, $l \in \{0, 1, 2, 3\}$ where $l = 0$ indicates the erased state. Standard deviations σ_e and σ_p are the channel parameters for this component, and remain constant throughout the lifetime of the device.

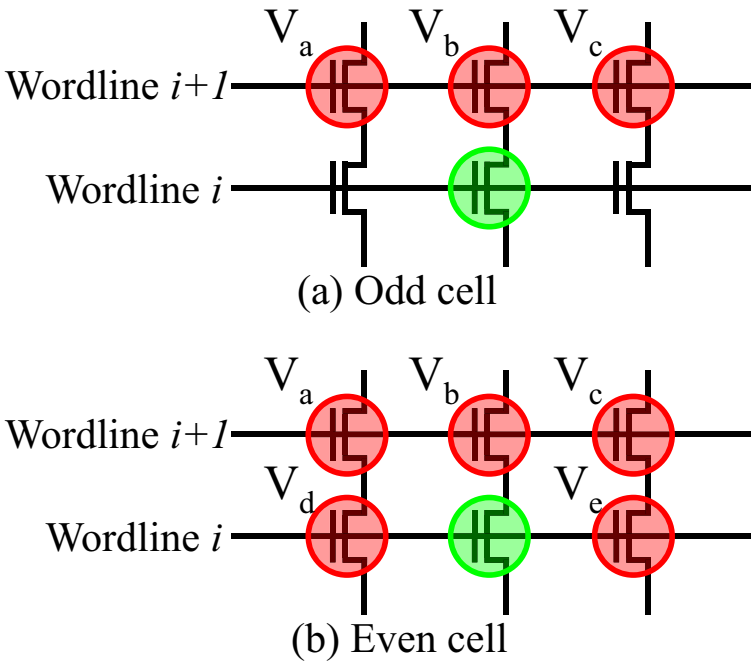


Figure 2.3: Spacial relationship between the interfering cells (red) and interfered cells (green). Flash memory is written wordline by wordline, i.e., wordline $i+1$ is written after wordline i. For the interference shown above, within a wordline even cells are written before odd cells.

Wear-out Noise n_w [29, 2, 3, 4, 35, 36, 37, 38, 39, 40]

The wear-out noise is modeled as a positive-side exponential[1] noise for each level, and the slope of the distribution is characterized by the channel parameter λ. The PDF is

$$f_{n_w}(n_w) = \begin{cases} \frac{1}{\lambda} e^{-\frac{n_w}{\lambda}} & \text{if } n_w \geq 0, \\ 0 & \text{if } n_w < 0. \end{cases} \tag{2.4}$$

Parameter λ increases as the device experiences more P/E cycles (or larger V_{acc}).

Cell-to-cell Interference n_{c2c} [4, 41, 42]

The cell-to-cell interference experienced by a cell is a weighted sum of the voltage increases in neighboring cells that occur *after* the cell of interest has been written [43]. Fig. 2.3

[1]For various device implementations wear-out noise can also be a negative-side exponential or a double-sided exponential (Laplace) distribution. The DVA and channel estimation techniques we present can be applied in all these cases.

shows the spacial relationships between the interfering cells and the cell of interest for Flash memories employing the common even-odd structure for writing and for reading. The green circle indicates the cell of interest, and the red circles indicate the interfering cells. Fig. 2.3(b) shows that even cells, which are written first, suffer interference from the odd cells in that wordline, which are written after the even cells. As is shown in Fig. 2.3(a), the odd cells do not suffer any interference from the even cells in the same wordline because the odd cells are written after the even cells. All the cells in wordline i suffer interference from the neighboring cells in wordline $i+1$, which is written after wordline i. The threshold voltage disturbance (increase) of odd cells $V_{c2c,odd}$ and even cells $V_{c2c,even}$ caused by this interference can be modeled as

$$V_{n_{c2c,odd}} = \gamma_a V_a + \gamma_b V_b + \gamma_c V_c \,, \tag{2.5}$$

$$V_{n_{c2c,even}} = V_{n_{c2c,odd}} + \gamma_d V_d + \gamma_e V_e \,. \tag{2.6}$$

As shown in Fig. 2.3, V_a, V_b, and V_c are voltage increases from the cells in the next wordline. V_b is from the cell directly above the cell of interest. V_a and V_c are voltage increases in cells diagonally adjacent on the next wordline that are located either upper left (V_a) or upper right (V_c). The voltage increases V_d and V_e are from adjacent cells on the same wordline that are to the left (V_d) or right (V_e). When these cells are programmed (subsequent to the programming of the cell of interest), they interfere with the cell of interest according to the coupling factors (γ's) between the interfering cells and the cell of interest. The magnitude of this noise component is thus related to two factors: the values of the voltage increases as the adjacent cells are written to their intended thresholds and the coupling factors γ.

Retention Noise n_r [29, 2, 3, 4, 44, 45, 46, 36, 41]

The retention noise is modeled as a Gaussian random variable with PDF

$$f_{n_r}(n_r) = \frac{1}{\sigma_r\sqrt{2\pi}} e^{-\frac{(n_r-\mu_r)^2}{2\sigma_r^2}}. \tag{2.7}$$

Both μ_r and σ_r^2 (characterized in detail below in Sec. 2.2.2) are dynamic channel parameters determined by the number of P/E cycles (or V_{acc} which will be introduced next), retention time and intended threshold voltage.

2.2.2 Channel Parameter Degradation Model

This subsection presents a model describing how the parameters describing the additive noise terms degrade as a function of P/E operations and retention time. From the discussion in Sec. 2.2.1, this degradation occurs for all the channel parameters except those describing the programming noise n_p.

Degradation model due to P/E operations [29, 2, 3, 4]

The degree of damage to a Flash cell's oxide layer caused by P/E operations is directly related to the volume of charge traveling through the oxide layer [47]. While the channel degradation caused by P/E operations is often modeled as a function of the number of P/E cycles, this model essentially assumes that approximately the same volume of charge travels through the oxide layer during each P/E cycle. The novel approach of DVA is to realize that the intended threshold voltages themselves can be varied over time so that less charge travels through the oxide layer during early P/E cycles when the channel is favorable. Because it is the charge stored in the floating gate that changes the threshold voltage, the difference between the intended threshold voltage and the erased state voltage is a good indicator of the amount of charged transferred. Thus, we define the voltage-based metric V_{acc}/V_{max} to

characterize P/E cycling where V_{acc} denotes the accumulated voltage over P/E cycles

$$V_{acc} = \sum_{j=1}^{N} (V_p^{(j)} - V_e), \tag{2.8}$$

and V_{max} is the maximum voltage difference between programmed and erased states of a Flash cell. In (2.8), N is the number of P/E cycles, $V_p^{(j)}$ is the intended threshold in jth P/E operation, and V_e is the intended threshold voltage of the erased state. Only the programming process is considered in this model, because program and erase are symmetric operations from the perspective of the amount of charge passing through the floating gate.

Degradation model for the wear-out noise parameter [39, 40]

P/E cycling operations cause the formation of oxide traps and interface traps in the cells [36, 35, 38, 37]. In [45] a power law describes how interface trap density depends on P/E cycle count as follows: $A_w \cdot (\text{P/E cycle count})^{k_i}$, where k_i and A_w are constant degradation parameters determined by the underlying physical properties of the Flash device. Replacing the P/E cycle count with V_{acc}/V_{max} yields our expression for interface trap density: $A_w \cdot ((V_{acc}/V_{max})^{k_i}$. Based on this expression and [39, 40], the wear-out channel parameter λ in (2.4) has a degradation model that can be formulated as

$$\lambda = C_w + A_w \cdot \left(\frac{V_{acc}}{V_{max}}\right)^{k_i}, \tag{2.9}$$

where C_w is another constant degradation parameters in addition to k_i and A_w .

Degradation model for the retention noise parameter [36, 45, 41]

Retention noise models channel degradation in the form a gradual decrease of threshold voltage. Both trap recovery and electron detrapping contributes to this degradation [44, 45, 46]. In addition to the interface trap density model, [45] suggests that oxide trap density as well can be modeled by a power of the P/E cycle count. With the V_{acc}-based characterization

of P/E cycling, total trap density can be represented as $A_r \cdot (V_{acc}/V_{max})^{k_i} + B_r \cdot (V_{acc}/V_{max})^{k_o}$. From [36, 45, 41], we define the degradation model for the retention noise channel parameters μ_r and σ_r^2 of (2.4) as [29],

$$\mu_r = -(V - V_0) \ln\left(1 + \frac{t}{t_0}\right) \left[A_r \left(\frac{V_{acc}}{V_{max}}\right)^{k_i} + B_r \left(\frac{V_{acc}}{V_{max}}\right)^{k_o}\right], \tag{2.10}$$

$$\sigma_r^2 = 0.1(V - V_0) \ln\left(1 + \frac{t}{t_0}\right) \left[A_r \left(\frac{V_{acc}}{V_{max}}\right)^{k_i} + B_r \left(\frac{V_{acc}}{V_{max}}\right)^{k_o}\right]^2, \tag{2.11}$$

where voltage V is the intended threshold voltage, and V_0 is the erased state threshold voltage. Degradation parameters k_i, k_o, A_r and B_r are constants determined by the physical properties of individual Flash memory product. Parameter t is the retention time and t_0 is its normalization factor. In this chapter, we study Flash read channel characteristics at a fixed retention target of one year ($t = 8760$ hours). The techniques presented in this chapter can, of course, be applied for any target retention time.

Note that often in practical systems the calculations in the equations (2.10) and (2.11) may not be easy to implement. The formulae can be simplified to

$$\mu_r = \gamma_{\mu_r}(V - V_0), \tag{2.12}$$

$$\sigma_r = \gamma_{\sigma_r}\sqrt{V - V_0}. \tag{2.13}$$

Cell-to-cell Interference Parameter Model [42]

In [42], the γ parameters in (2.5) are modeled as random variables with truncated Gaussian distributions. The authors of [42] mention that the means of interfering cells' γ's in the next wordline to be programmed are also random. We simplify the model by assuming the means of all the γ's are constant. In our research, the parameter model provided in [42] is utilized, and the cell-to-cell interference strength factor is set to $s = 0.2$ which yields the full set of cell-to-cell interference parameters given in Sec. 2.10. This setting generates a pair of even-cell and odd-cell read channels with sufficient difference and reasonable typicality.

Programming Error Parameter Degradation Model [32]

We employ the programming error parameter degradation model proposed in [32]. $P_{X,Y}$ has an exponential relationship with P/E cycle counts.

$$P_{X,Y} = \exp(c_1 x + c_0) \tag{2.14}$$

The parameters c_1 and c_0 are different for each $P_{X,Y}$, and variable x indicates the *normalized* P/E cycle count where the manufacture specified lifetime is used as the normalization factor. In this chapter, the normalization factor is set to 3000 P/E cycles. The resulting $P_{X,Y}$ expressions for our model are given in the Sec. 2.10.

2.2.3 Models Used in This Chapter

The results presented in this chapter are software simulation results based on two channel models that we have constructed based on the literature. Model 1 is the channel model used in [2], which consists of programming noise, wear-out noise and retention noise. Model 2 is the complete channel model introduced in Sec. 2.2.1.

Model 1

The channel model in [2] is an exponentially modified Gaussian distribution for each level. Cell-to-cell interference and programming error are not considered in this model. As a result, the channel characteristics of even and odd cells are the same. This channel model is still very similar to the multi-modal Gaussian model.

Model 2

The channel model proposed in Sec. 2.2.1 differentiates the even and odd cell read channels, and considers the effect of programming error and cell-to-cell interference.

2.3 Dynamic Voltage Allocation with Ideal Channel Information

Dynamic Voltage Allocation (DVA) [29] is an algorithm that uses a single scaling factor α to attenuate the standard threshold voltages used to write to the cells. The DVA algorithm computes the appropriate scaling factor such that the read channel achieves at least a certain minimal value of mutual information required to support the rate at which information is stored (the rate indicated by the channel code). The range between the smallest and largest intended voltages is scaled so that it expands as the channel becomes more degraded until it reaches the range that would be employed in a system without DVA. In some Flash devices it may be possible to increase the range beyond the standard range without DVA, but that is not a focus of this chapter.

For Flash memory described by Model 1, the mutual information is calculated as the difference of marginal and conditional differential entropies as follows [48]:

$$I(X;Y) = h(Y) - h(Y|X) \ . \tag{2.15}$$

Random variable Y describes the measured threshold voltage, and X represents the intended threshold voltage. The probability distribution of Y is the channel distribution $f_Y(y)$, so

$$h(Y) = -\int_{-\infty}^{+\infty} f_Y(y) \log \left(f_Y(y) \right) \mathrm{d}y \ . \tag{2.16}$$

The relevant conditional differential entropy is calculated as

$$h(Y|X) = -\int_{-\infty}^{+\infty} \sum_{i=1}^{4} f_{X,Y}(x_i, y) \log \Big(f_{Y|X}(y|x_i) \Big) \mathrm{d}y \ \ , \tag{2.17}$$

where the joint distribution is $f_{X,Y}(x,y) = f_{Y|X}(y|x)P_X(x)$, $P_X(x)$ is the probability mass function of X, and $f_{Y|X}(y|x)$ is the conditional PDF of Y when the cell is written to a single level indicated by x. Let $\{v_i\}$ represent the set of default intended threshold voltages, the

intended threshold voltage after scaling is $x_i = \alpha v_i$, where α is the scaling factor. As a result, adjusting the scaling factor will change both $h(Y)$ and $h(Y|X)$.

Because the cell-to-cell interference of a certain cell is a function of the threshold voltages of its surrounding cells, the Flash channel described by model 2 is a channel with memory. From [41], equation (2.15) is a lower bound on the actual mutual information because it treats interference as independent noise. Although equation (2.15) underestimates the mutual information, it is still a valid optimization objective function for DVA when cell-to-cell interference is treated as noise by the controller. If cell-to-cell interference is cancelled by signal processing in the controller, then the portion of cell-to-cell interference that is cancelled should be removed from the modeled noise before computing the mutual information.

As discussed in Sec. 2.2, the channel probability distribution function is determined by the distributions of the five noise components. In general, the distribution can be calculated by convolving the five distributions. In [49], a model very similar to the one in Sec. 2.2 is presented and the analytical channel distribution function is calculated. The expression has a relatively high complexity. In this chapter, numerical convolution is used to calculate an approximation of $f_{Y|X}(y|x)$. Note that for generating noise used in Monte Carlo simulation it is not necessary to compute this convolution. Rather, the individual noise terms, each of which is relatively simple in distribution, can be generated and added to the original signal.

Our implementation uses a bisection algorithm to find the scaling factor α. The scaling factor has a range of [0,1], where a scaling factor of 1 corresponds to the maximum allowed threshold voltage for each level (the levels that would be used by a system not implementing DVA). There are two modes for the DVA algorithm. In the first mode, DVA makes adjustments based on the current channel distribution, and aligns the mutual information to a preset threshold that builds in a fixed margin to account for degradation before the scale factor is adjusted again. In the second mode, DVA makes adjustments based on the prediction of channel conditions in a future time point corresponding to the next DVA scale factor update. In this mode, if the channel model and channel parameter degradation models are

correct, the mutual information at that future time point will be the target value. We use the first mode in this chapter, because in practice the channel degradation models (especially the parameters in the models) may not accurately reflect the degradation process for device from different manufactures or for the same device under different working conditions (e.g. temperature).

Assuming ideal channel information, which is the exact model and parameters of the channel, a simulation can be conducted where the DVA algorithm adjusts the scale-factor every certain number of P/E cycles. In this chapter, DVA functions every 100 P/E cycles for simplicity. The number of P/E cycles between DVA operations depends on how quickly the channel is varying and the appropriate interval itself might change over the lifetime of the device. For example, near the end of life when the channel is degrading rapidly a smaller interval might be appropriate.

In each iteration of the simulation, the DVA algorithm scales the intended threshold voltages to increase the channel mutual information to a predefined threshold. Because perfect knowledge of the channel state is assumed, the performance indicates the theoretical limit of the algorithm under these ideal conditions. For this ideal simulation we use Model 2, the full channel model presented in Sec. 2.2.

DVA is most effective in extending lifetime when the major channel degradation is caused by the accumulated effect of charge traveling through the oxide layer of the Flash memory cells. Retention loss is the major degradation in this case. In this chapter, the retention time is set to be a year to represent the worst possible channel for which a device is rated. If the actual retention time exceeds the fixed value, the channel could still provide enough mutual information depending on the distance between the actual channel mutual information and the mutual information limit for the channel code. DVA's performance when the retention time is 0 is not investigated, as we believe the lifetime extension achieved by DVA will be limited in this case.

In this chapter, all simulations assume MLC Flash (four levels) with each level used

equally likely, and the target mutual information is 1.945 bits per cell for both even and odd channels. Because raw BER is determined by both the channel condition and the placement of read threshold voltages, there is no simple way to translate between mutual information and BER. However, simulation results show that a target of 1.945 bits achieves a raw BER of 10^{-2} in Fig. 2.12 using DTA to allocate read thresholds. We note that the raw BER of 10^{-2} is shown in [17] to be an operating point for Code 2 in [17] with 6 reads using a version of mutual-information-based DTA. In any case, the mutual information target can be adapted easily to whatever mutual information is needed to support a specific channel code (LDPC, BCH, or other). The target for the DVA algorithm is set to be 1.965 bits to provide an additional 0.02 bits of margin above the code-based target to allow for channel degradation before the next DVA update. The retention time is set to be one year. The default lifetime measured in P/E cycles for the programming error model is set to be 3000 P/E cycles.

The default intended threshold voltages are described by the vector $T = [2.8, 5.2, 6.4, 7.86]$ [29]. The actual threshold voltages resulting from DVA updates have the form αT, and Fig. 2.6 shows how the scale factor α varies over time. When using fixed voltage allocation, the scaling factor α is fixed to be 1, so the default intended threshold voltages are used for each level through the entire lifetime of the device. When using DVA the scaling factor α starts at a value smaller than 1 and increases as the channel degrades until it reaches the value of 1 which corresponds to the maximum voltage levels supported by the device. The full set of channel parameters is given in Sec. 2.10.

For Flash memories using the even-odd write structure presented in Sec. 2.2, the even-cell and odd-cell read channel characteristics are different as even cells are written before odd cells. The even cells experience more severe cell-to-cell interference and provide less mutual information under the same intended threshold voltage allocation. One approach to optimize this type of Flash memory is to have a single DVA optimization process controlled by the mutual information of the even cells, which face the more severe channel conditions. This

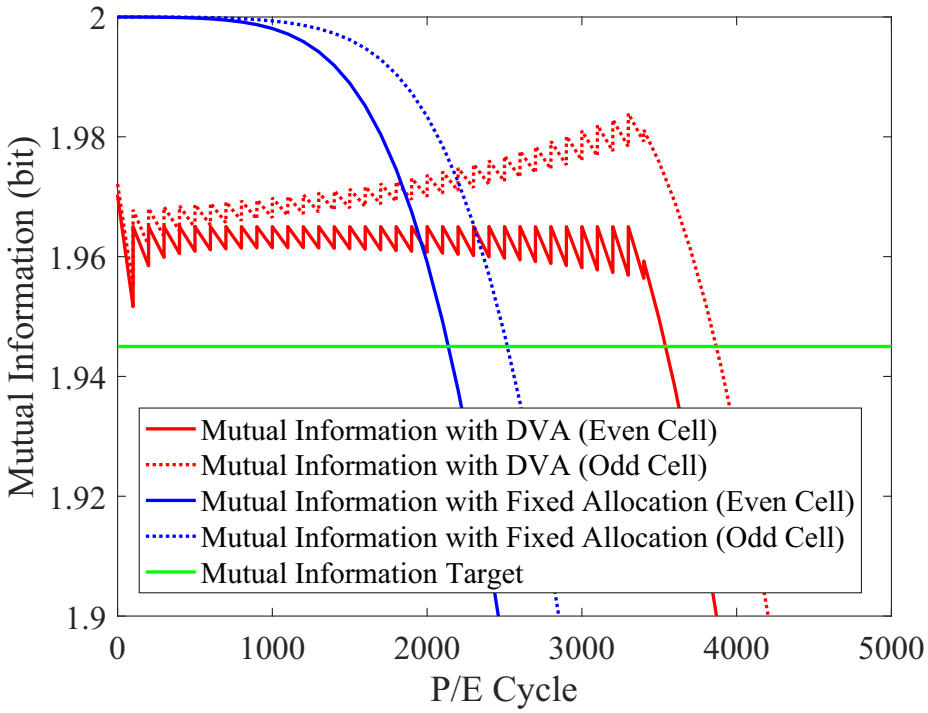


Figure 2.4: DVA performance with ideal channel information versus fixed allocation's performance. (Ground truth channel is Model 2. DVA targets even channel. Channel parameters are listed in the Sec. 2.10.)

approach guarantees the worst-case performance. Fig. 2.4 shows that with this approach the DVA algorithm can improve the lifetime of even cells by 65.7% from 2136 P/E cycles to 3540 P/E cycles and odd cells by 53.6% from 2517 P/E cycles to 3867 P/E cycles for the channel defined by the parameters in Sec. 2.10. Fig. 2.6 shows the scale factor α used for all cells (the black curve). Note the rapid reduction in the mutual information trajectory shown in Fig. 2.4 around the end of lifetime. This rapid reduction occurs after the scaling factor has become one and cannot be further increased as shown in Fig. 2.6.

One problem with the previous approach is that there is a significant performance difference between the even and odd cells. The intended threshold voltages provide too much mutual information margin for the odd cells. In fact, Fig. 2.4 shows the odd-cell mutual information margin increasing over P/E cycles.

This situation is improved by using two separate DVA optimization processes in the

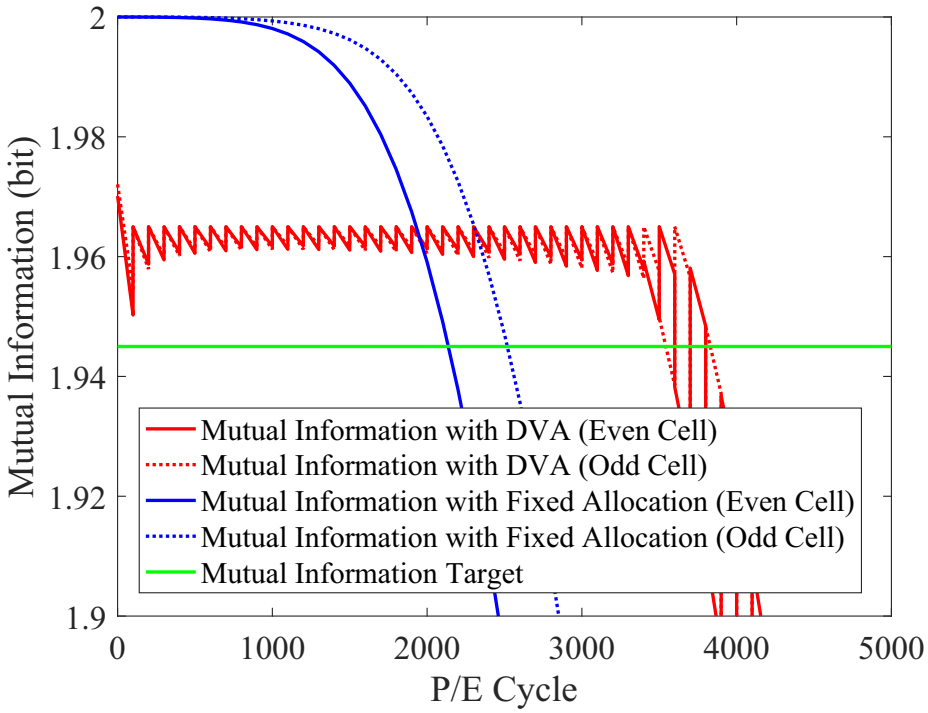


Figure 2.5: DVA performance with ideal channel information and the performance of the fixed allocation. (The ground truth channel is Model 2. The writing order of pages is switched every 100 P/E cycles. Joint DVA is use to adjust both scaling factors for even and odd cells. Channel parameters are listed in Sec. 2.10.)

system, one for the even-cell channel and one for the odd-cell channel. This solution will remove the excessive mutual information margin of the odd-cell channel, but the performance difference measured by lifetime will not be narrowed. One fundamental cause of the problem is that even cells experience more accumulated damage than odd cells.

To address this disparity, we alternate the writing order between even cells and odd cells every 100 P/E cycles. For example, even cells are written before odd cells from 0 to 99 P/E cycles, and odd cells are written before even cells from 100 P/E cycles to 199 P/E cycles. In this way, the two channels can experience similar accumulated damage by switching which channel receives the more severe cell-to-cell interference periodically. We apply DVA to this alternating-write-first approach with distinct scaling factors for the even-cell channel and the odd-cell channel, and use a joint DVA algorithm to adjust the two factors. The mutual information of the two channels will reach the predefined target together after each iteration.

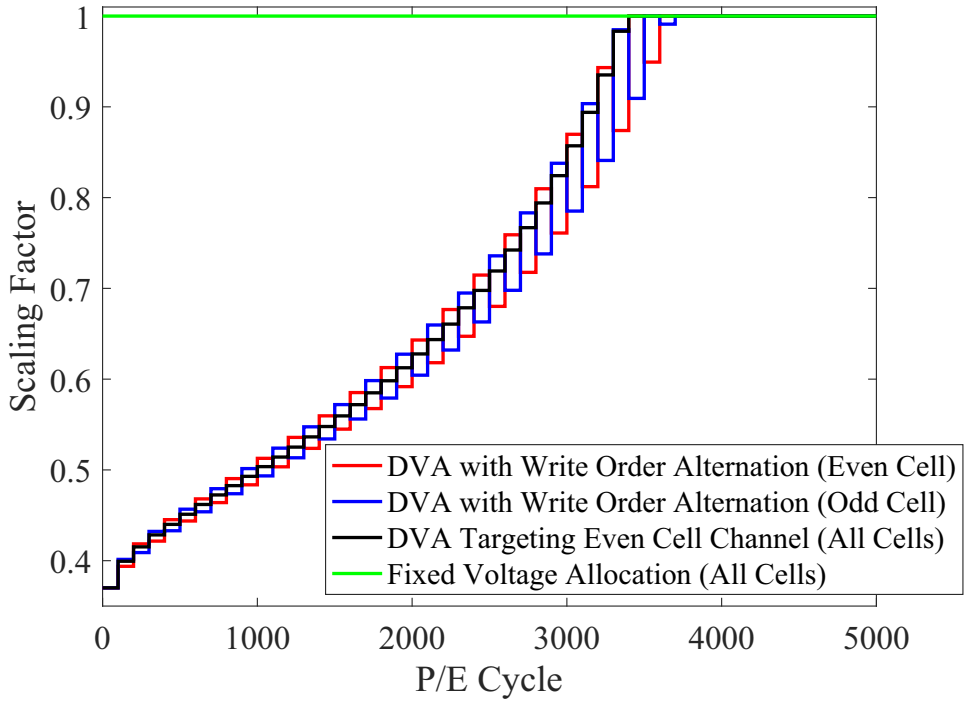


Figure 2.6: Scaling factors generated by DVA with ideal channel information. (Result corresponds to Fig. 2.4 and 2.5. Ground truth model is Model 2. Channel parameters are listed in Sec. 2.10. The starting value of α is 0.37.)

This approach equalizes the accumulated voltage V_{acc} of even-cell and odd-cell channel, and narrows the difference of the two channels' characteristics. Fig. 2.5 shows that with this approach the DVA algorithm can improve the lifetime of even cells by 68.5% from 2136 P/E cycles to 3600 P/E cycles and odd cells by 40.7% from 2517 P/E cycles to 3542 P/E cycles for the channel defined by the parameters in Sec. 2.10. Fig. 2.6 shows the scale factor α used for even and odd cells as the red and blue curves respectively.

If the overall lifetime is defined as the number of P/E cycles until the mutual information of either channel crosses below the 1.945 bits target, joint DVA with the alternating-write-first approach extends the lifetime by 65.8% from 2136 P/E cycles to 3542 P/E cycles. It may seem surprising that the overall lifetime is improved by 65.8% when the lifetime of odd cells only improves by 40.7%, but the odd cells were lasting longer than the even cells to begin with so that the overall lifetime improvement is primarily driven by the improvement

in the even cell lifetime.

Recall that the single DVA targeting even-cell channel extends the lifetime by 65.7% from 2136 P/E cycles to 3540 P/E cycles, so that the joint DVA did not improve lifetime significantly more than the simpler single DVA process for this channel model. When the strength factor of cell-to-cell interference s increases, the average distortion of the interference for even-cell and odd-cell channels will rise, and the performance gain brought by the joint DVA approach will be more significant. For the remainder of this chapter, the joint DVA with the alternating-write-first approach or its approximations will be used to implement DVA when the channel model in the simulation is Model 2, which includes cell-to-cell interference.

2.4 Channel Parameter Estimation

In order to dynamically allocate intended thresholds in a real system, the channel characteristics must be determined dynamically as they evolve. Channel estimation can be achieved by estimating the channel parameters in a channel model assumption. In [30], the authors demonstrate that accurate channel estimation can be achieved with limited computational complexity using multi-modal Gaussian channel model assumption and least squares algorithms.

Using both the channel model and the parameter degradation model, the channel characteristics after a specific number of P/E cycles can be predicted for a specified retention time. However, the exact channel model and parameter degradation model need to be known in advance to enable this approach. When considering on-the-fly scenarios, we may also want to use actual measurements from the working device to determine the channel parameters without making detailed assumptions about the parameter degradation model or perhaps even the channel model itself.

In practical scenarios, empirical histograms of the threshold voltage can be measured with multiple read operations. Combined with the knowledge of the read reference voltages used in the measurements, the empirical distributions provide enough information about

the ground truth channel (voltage) distribution. For a given channel model, this measured histogram can be used to estimate the model parameters using a least squares algorithm. This approach does not require the channel parameter degradation model.

In both [2] and [30] , the channel model assumptions are constructed to be as close to the ground truth model as possible. Later in this chapter, the idea of using the simple multi-modal Gaussian model assumption, different from the ground truth model, is explored. For intended threshold voltage $x_i, i \in \{1, \cdots, 4\}$, the conditional channel model assumption is

$$f_{Y|X}^{(\text{asm})}(y|x_i) = \frac{1}{\sigma_i\sqrt{2\pi}} e^{-\frac{(y-x_i-\mu_i)^2}{2\sigma_i^2}}, \tag{2.18}$$

where μ_i is the bias of the mean and σ_i is the standard deviation. Each intended threshold voltage level is treated as equally likely.

2.4.1 Channel Parameter Estimation Problem Formation

The channel parameter estimation is formulated as an optimization problem where a cost function is to be minimized by estimating the channel parameters. In [2], the channel model assumption is the same as the ground truth (both are Model 1), so the channel parameters to be estimated $\boldsymbol{\mathcal{P}} = [\lambda, \sigma_p, \sigma_e, \gamma_{\sigma_r}, \gamma_{\mu_r}]$ are the ones defined in the ground truth model. In Sec. 2.5 and later sections, the channel model assumption is a multi-modal Gaussian distribution, so the parameters to be estimated are the four biases and standard deviations of the four levels $\boldsymbol{\mathcal{P}} = [\mu_1, \mu_2, \mu_3, \mu_4, \sigma_1, \sigma_2, \sigma_3, \sigma_4]$.

Regardless of the choice of the ground truth model and channel model assumption. the cost function is formulated as follows. Define $[q_0, q_1, \ldots, q_M]$ as the boundaries of the M bins where $q_0 = -\infty$, and $q_M = \infty$. The expected number of cells in each bin can be computed as

$$\hat{N}_{bin,i} = \sum_{k=1}^{L} N_k P(q_i < y < q_{i+1}|x_k)\,, \tag{2.19}$$

where $P(q_i < y < q_{i+1}|x_k) = \int_{q_i}^{q_{i+1}} f_{Y|X}^{(\text{asm})}(y|x_k)\,\mathrm{d}y$ denotes the probability of a measured thresh-

old falling in the ith bin when the intended threshold is x_k according to the channel model assumption. $L = 4$ is the number of intended threshold levels, and N_k is the number of cells in each level for the stored data. The cost function is defined as the normalized square Euclidean distance between the expected histogram induced by the estimated parameters and the reference histogram

$$C_M = \sum_{i=0}^{M-1} \left(\frac{N_{bin,i} - \hat{N}_{bin,i}}{N} \right)^2 , \tag{2.20}$$

where N is the total number of cells measured, and $N_{bin,i}$ is the ith bin's cell count in the reference histogram. The gradient of the cost function is defined as

$$\nabla \boldsymbol{C}_M(\boldsymbol{\mathcal{P}}) = 2 \cdot (\boldsymbol{J}_{\boldsymbol{G}_M}(\boldsymbol{\mathcal{P}}))^T \cdot \boldsymbol{G}_M(\boldsymbol{\mathcal{P}}) , \tag{2.21}$$

where $\boldsymbol{J}_{\boldsymbol{G}_M}(\boldsymbol{\mathcal{P}})$ is the Jacobian matrix of the normalized difference vector $\boldsymbol{G}_M$ between the estimated histogram and the measured histogram.

2.4.2 Least Squares Algorithms

Least squares algorithms have been widely used to fit a parameterized model to a data set. Three algorithms are examined in the following discussion.

Gradient Descent (GD)

GD minimizes the cost function by refining initial estimation of the parameters based on a linear approximation. In each iteration, the estimation is renewed by a step vector following the gradient of the cost function.

Algorithm 1 Gradient Descent Algorithm

Initialize step size β and $\boldsymbol{\mathcal{P}} = \boldsymbol{\mathcal{P}}^{(0)}$
while $\|\boldsymbol{\mathcal{P}}^{(k+1)} - \boldsymbol{\mathcal{P}}^{(k)}\| > \eta$ and $k < MaxIteration$ **do**
    Compute $\boldsymbol{J}_{\boldsymbol{G}_M}(\boldsymbol{\mathcal{P}}^{(k)})$ and $\boldsymbol{G}_M(\boldsymbol{\mathcal{P}}^{(k)})$
    Compute $\nabla \boldsymbol{C}_M(\boldsymbol{\mathcal{P}}^{(k)}) = 2 \cdot (\boldsymbol{J}_{\boldsymbol{G}_M}(\boldsymbol{\mathcal{P}}^{(k)})^T \cdot \boldsymbol{G}_M(\boldsymbol{\mathcal{P}}^{(k)})$
    $\boldsymbol{\mathcal{P}}^{(k+1)} = \boldsymbol{\mathcal{P}}^{(k)} - \beta \cdot \nabla \boldsymbol{C}_M(\boldsymbol{\mathcal{P}}^{(k)})$
    $k = k + 1$
end while

Gauss-Newton (GN)

A quadratic model is employed to provide more accurate approximations of the cost function. The iterative relation can be represented as

$$\boldsymbol{\mathcal{P}}^{(k+1)} = \boldsymbol{\mathcal{P}}^{(k)} - (\boldsymbol{J}_{\boldsymbol{G}_M}^T \boldsymbol{J}_{\boldsymbol{G}_M})^{-1} \boldsymbol{J}_{\boldsymbol{G}_M}^T \boldsymbol{G} = \boldsymbol{J}_{\boldsymbol{G}_M}^+ \boldsymbol{G}\,, \tag{2.22}$$

where $\boldsymbol{J}_{\boldsymbol{G}_M}^+$ is the pseudo-inverse of $\boldsymbol{J}_{\boldsymbol{G}_M}$. Gauss-Newton algorithm can then be formulated as follows:

Algorithm 2 Gauss-Newton Algorithm

Initialize $\boldsymbol{\mathcal{P}} = \boldsymbol{\mathcal{P}}^{(0)}$
while $\|\boldsymbol{\mathcal{P}}^{(k+1)} - \boldsymbol{\mathcal{P}}^{(k)}\| > \eta$ and $k < MaxIteration$ **do**
    Compute $\boldsymbol{J}_{\boldsymbol{G}_M}(\boldsymbol{\mathcal{P}}^{(k)})$ and $\boldsymbol{G}_M(\boldsymbol{\mathcal{P}}^{(k)})$
    $\boldsymbol{\mathcal{P}}^{(k+1)} = \boldsymbol{\mathcal{P}}^{(k)} - (\boldsymbol{J}_{\boldsymbol{G}_M}(\boldsymbol{\mathcal{P}}^{(k)}))^+ \cdot \boldsymbol{G}_M(\boldsymbol{\mathcal{P}}^{(k)})$
    $k = k + 1$
end while

Levenberg-Marquardt (LM) [50]

By combining GD and GN, LM possesses the advantages of both algorithms. The update vector $\boldsymbol{\delta}_{\boldsymbol{\mathcal{P}}}$ is calculated by solving $(\boldsymbol{J}_{\boldsymbol{G}_M}^T \boldsymbol{J}_{\boldsymbol{G}_M} + \beta \cdot \mathrm{diag}((\boldsymbol{J}_{\boldsymbol{G}_M})^T \boldsymbol{J}_{\boldsymbol{G}_M}))\boldsymbol{\delta}_{\boldsymbol{\mathcal{P}}} = \boldsymbol{J}_{\boldsymbol{G}_M}^T \boldsymbol{G}$ where β acts as a weight to combine the two algorithms.

In [2], we studied the performance of the three algorithms with the ground truth model and the channel model assumption both to be Model 1. In this case, the channel parameters need to be estimated are $\mathcal{P} = [\lambda, \sigma_p, \sigma_e, \gamma_{\sigma_r}, \gamma_{\mu_r}]$ as is defined in Sec. 2.2. Fig. 2.7 compares

Algorithm 3 Levenberg-Marquardt Algorithm

Initialize $\beta, v, \boldsymbol{\mathcal{P}} = \boldsymbol{\mathcal{P}}^{(0)}$ and $UpdateFlag = 1$
while $\|\boldsymbol{\mathcal{P}}^{(k+1)} - \boldsymbol{\mathcal{P}}^{(k)}\| > \eta$ and $k < MaxIteration$ **do**
    if $UpdateFlag = 1$ **then**
        Compute $\boldsymbol{J}_{\boldsymbol{G}_M}(\boldsymbol{\mathcal{P}}^{(k)})$ and $\boldsymbol{G}_M(\boldsymbol{\mathcal{P}}^{(k)})$
    end if
    Solve $((\boldsymbol{J}_{\boldsymbol{G}_M})^T \boldsymbol{J}_{\boldsymbol{G}_M} + \beta \cdot diag((\boldsymbol{J}_{\boldsymbol{G}_M})^T \boldsymbol{J}_{\boldsymbol{G}_M}))\boldsymbol{\delta}_{\boldsymbol{\mathcal{P}}} = (\boldsymbol{J}_{\boldsymbol{G}_M})^T \boldsymbol{G}_M$
    Compute $\boldsymbol{J}_{\boldsymbol{G}_M}(\boldsymbol{\mathcal{P}}^{(k)})$ and $\boldsymbol{G}_M(\boldsymbol{\mathcal{P}}^{(k)})$
    $\boldsymbol{\mathcal{P}}_{temporary} = \boldsymbol{\mathcal{P}} - \boldsymbol{\delta}_{\boldsymbol{\mathcal{P}}}$
    if $\sum(err(\boldsymbol{\mathcal{P}}))^2 > \sum(err(\boldsymbol{\mathcal{P}}_{temporary}))^2$ **then**
        $UpdateFlag = 1$
        $\beta = \beta \cdot v$
        $\boldsymbol{\mathcal{P}} = \boldsymbol{\mathcal{P}}_{temporary}$
    else
        $UpdateFlag = 0$
        $\beta = \beta / v$
    end if
    $k = k + 1$
end while

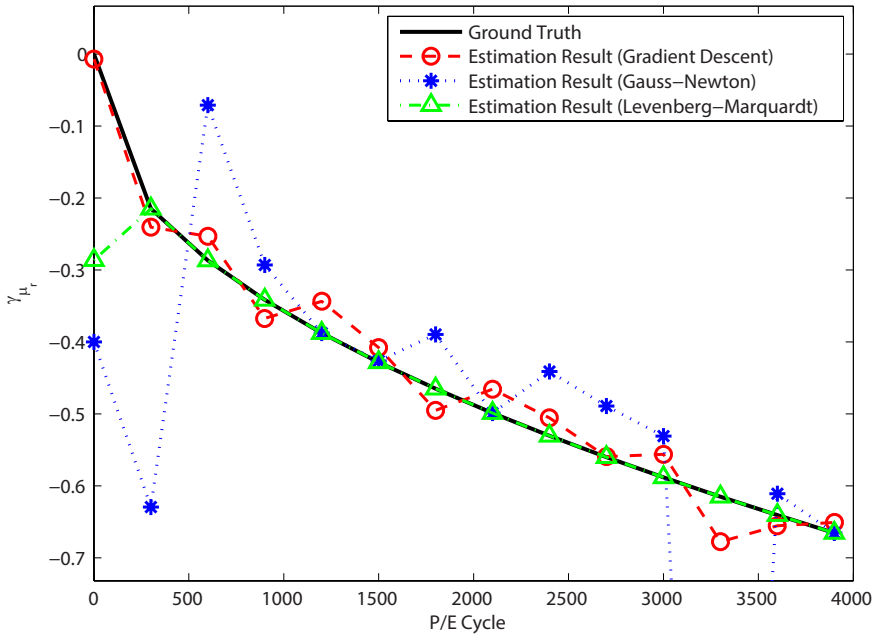


Figure 2.7: Estimation result versus ground truth for γ_{μ_r} using 10-bin equal-probability histogram.

Table 2.1: Converge counts of least square algorithms (over 14 cases).

No. of Reads	GD	GN	LM
6	0	1	12
9	0	3	13
12	0	4	11

the estimating results of γ_{μ_r} with the ground truth, where the estimation algorithms employ 10-bin equal-probability histograms (will be defined in Sec. 2.4.3) as input. The simulations are conducted over 14 different P/E cycle conditions. The LM algorithm performs significantly better than GD and GN in terms of both the estimation accuracy and the ability to adapt to different channel conditions.

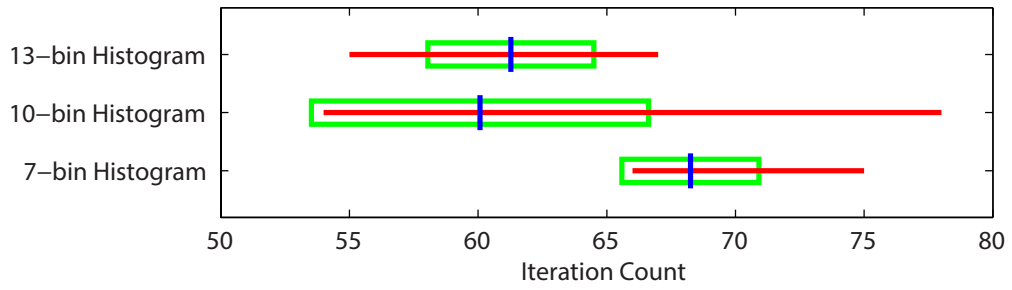
This behavior is further demonstrated in Table 2.1, which shows the convergence counts of the three algorithms over the 14 channels. Every estimated parameter needs to be within $\pm 1\%$ of the ground truth parameter to qualify as a converged result. GD fails in all simulation cases while reaching the maximum allowed number of iterations in every case. GN provides good results in certain cases with very few iterations. LM provides high estimation accuracy over different channel conditions, with some failures when the channel conditions are very good. Note that in Table 2.1, the 10-bin (9-read) histogram provides the best performance. This also contribute to the discussion about the binning strategy in the next subsection.

2.4.3 Binning Strategy

In this subsection, different binning strategies are analyzed. We will still focus on the scenario where both the ground truth model and channel modal assumption are Model 1. As the rest of the chapter will demonstrate, the results here are also suitable for a multi-model Gaussian channel assumption when the ground truth are either Model 1 or 2.

Number of Bins

A fairly accurate channel estimation can, of course, be derived from a complete voltage scan which uses the smallest possible bin width by reading at every available voltage level using



Note: Horizontal line segments represents the range of iteration counts; vertical line segment represents the means of iteration counts; rectangles represents the standard deviation of iteration counts.

Figure 2.8: Iteration count statistics using Levenberg-Marquardt algorithm.

the so-called debug mode. However, the large number of reads required by this process stalls normal operations. Such a large number of reads is likely not necessary. From the soft decoding literature [31, 17, 30], a relatively small number of read voltages is sufficient to give good performance in terms of both decoding and channel estimation.

Furthermore, too many bins in the histogram will cause high computational cost in each iteration of the least squares algorithms described in Section 2.4.2, and also require more storage space. Thus a relatively small number of bins can reduce both complexity and latency. The choice for the number of bins also depends on the channel estimation algorithm employed. Basic algorithms usually require more detailed channel measurements than advanced algorithms.

Fig. 2.8 depicts key statistical metrics about the number of iterations when employing LM with histograms that differ in resolution over the 14 cases in Sec. 2.4.2. 10-bin (9-read) histograms reduce the number of iterations needed with respect to the results using 7-bin (6-read) histograms. 13-bin (12-read) histograms do not provide significant reduction in iteration counts. We conjecture that the 10-bin histogram provides good performance as it strikes the right balance. If the number of bins is too small, then too many channels can match the measured histogram so that the optimization cannot get a clear direction. If the number of bins is too large, the estimation problem is over-constrained, which slows convergence.

From the discussion above and Table 2.1, we conclude that a 10-bin histogram can provide enough information for accurate estimations of the channel parameters in our model. Thus, in exploring the performance of the three bin-placement paradigms, we focus on the 10-bin case.

Bin-Placement Paradigm

We considered three bin-placement paradigms: equal-width, equal-probability, and maximum mutual information (MMI). Equal-width histograms have bins covering intervals of equal length except for the semi-infinite bins at the leftmost and rightmost boundaries. Equal-probability histograms allocate bins having the same probability (i.e. the same number of occurrences in each bin). MMI word-line voltage placement proposed in [31, 17] optimizes decoding performance by maximizing the mutual information between the distribution of levels and the histogram bin identified when the cell is read.

As presented in Section 2.4.1, channel parameters are estimated by minimizing the squared Euclidean distance between the measured histogram acting as the *reference* and the histogram induced by the channel model assumption and estimated channel parameters. To achieve good estimation accuracy, the measured histogram should be as close to the original channel distribution as possible. To compare bin-placement paradigms, the squared Euclidean distance between the channel distribution $f_Y(y)$ and the histogram induced by $f(y)$ is used as the metric to evaluate the amount of discretization error of a bin-placement paradigm. This metric D_{E^2} is defined as follows:

$$D_{E^2} = \sum_{i=0}^{M-1} \int_{q_i}^{q_{i+1}} \left(f_Y(y) - \frac{H_i}{q_{i+1} - q_i} \right)^2 \mathrm{d}y \,, \tag{2.23}$$

where $f_Y(y)$ is the ground truth channel distribution, M is the number of bins, and q_i, q_{i+1} represent the left and right boundary of the the ith interval. H_i is the probability of ith bin induced by $f_Y(y)$, $H_i = \int_{q_i}^{q_{i+1}} f_Y(y)\mathrm{d}y$, and $\frac{H_i}{q_{i+1}-q_i}$ denotes the probability density of the ith

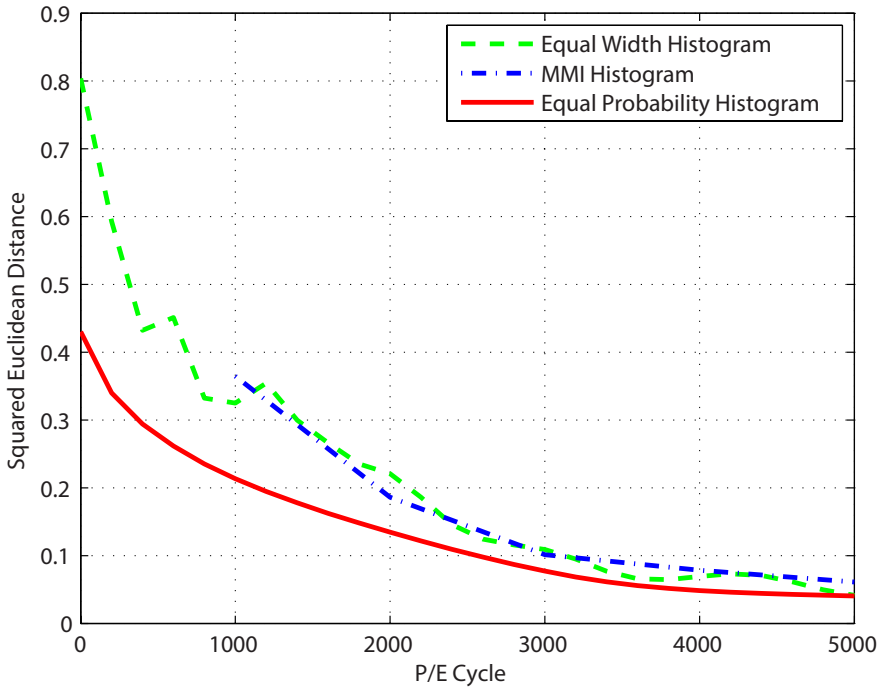


Figure 2.9: Squared Euclidean distance between the channel distributions and corresponding histograms (10 bins).

bin.

Fig. 2.9 shows D_{E^2} with 10-bin histograms for the three bin-placement paradigms. The equal-probability bin placement provides a lower D_{E^2}, and hence a better approximation to the original channel, than the other two strategies over a large span of P/E cycling conditions. The performance difference is especially significant when the device condition is new. This behavior is also seen with 7 bins.

As a complement to D_{E^2}, effective resolution is a second metric with which to compare bin placement paradigms. Effective resolution represents the bin count of an ideal non-redundant histogram that conveys the same amount of distribution shape information as the histogram under consideration. Because a read at the shared boundary of two zero-height bins does not provide additional information about the distribution, two adjacent zero-height bins can be combined as one. Effective resolution is essentially the bin count after combining adjacent zero-count bins.

Although histogram bin probabilities derived from integration of the channel model over

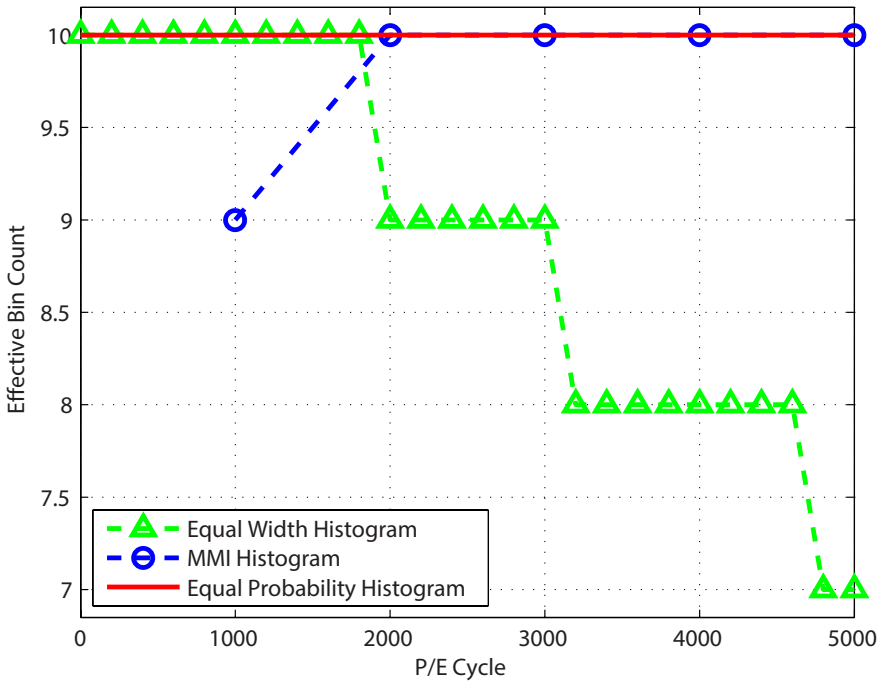


Figure 2.10: Effective resolution of different histograms (10 bins).

the width of bins are nonzero, real measurements from a finite number of cells (e.g. a block) will often produce zero-count bins if the bin probability is efficiently small. Fig. 2.10 shows the effective resolution as a function of the number of P/E cycles for the three bin-placement paradigms. Adjacent bins with induced probability less than 10^{-4} are combined. The equal-probability bin-placement paradigm has full resolution throughout the entire P/E cycling process, while the other paradigms lose resolution in some P/E cycling conditions. This suggests that the equal-probability bin-placement paradigm has a good tracking capability over the whole Flash lifetime.

To summarize, when the ground truth model and channel model assumption are both Model 1, 10-bin equal-probability histogram provides the best performance for channel parameter estimation. We will extend this result to the rest of the chapter where the channel model assumption (multi-modal Gaussian distribution) is different from the ground truth channel.

Because DVA is an iterative process, at every iteration, bin placements are updated based

on the channel estimation. The placements will only be optimal for the current channel condition even if the estimation is perfect. In order to provide more accurate placements for the next iteration, the placements are scaled by the ratio between the newly calculated scaling factor and the previous scaling factor used before the DVA process in this iteration. This approach provides a linear prediction of bin placements for next iteration's channel condition, and facilitates the channel estimation process.

2.5 Dynamic Voltage Allocation with Model/Channel Mismatch

When the channel model agrees exactly with the actual channel, channel estimation can precisely characterize the channel if the estimated channel parameters are accurate. However, such perfect agreement of the channel model with the actual channel is hard to achieve in practice because of the many factors involved in shaping the Flash read channel characteristics. Even if the exact model can be known, the complexity of the model may preclude its use in channel estimation and DVA because of the associated complexity.

The lack of perfect agreement limits the precision with which the channel can be characterized even with the best-case parameter estimation. However, this problem can be controlled by selecting a reasonable *channel model assumption* that focuses on the most significant channel characteristics. A good choice can reduce the computational complexity of channel estimation and DVA while also capturing the essential behavior of the channel.

In [30], the authors investigate a scenario where the channel model assumption matches the actual channel and the channel model is a relatively simple multi-modal Gaussian distribution. In [2], we demonstrated the estimation performance where the channel model matched the actual channel exactly, but the more complex Model 1 of Sec. 2.2.3 was used. In both cases, high estimation accuracy is achieved. In this section, DVA (and implicitly estimation) performance using the simple multi-modal Gaussian model as the assumption and the complex Models 1 and 2 of Sec. 2.2.3 for the actual channel is presented. Thus the channel parameters that need to be estimated for multi-modal Gaussian distribution are the

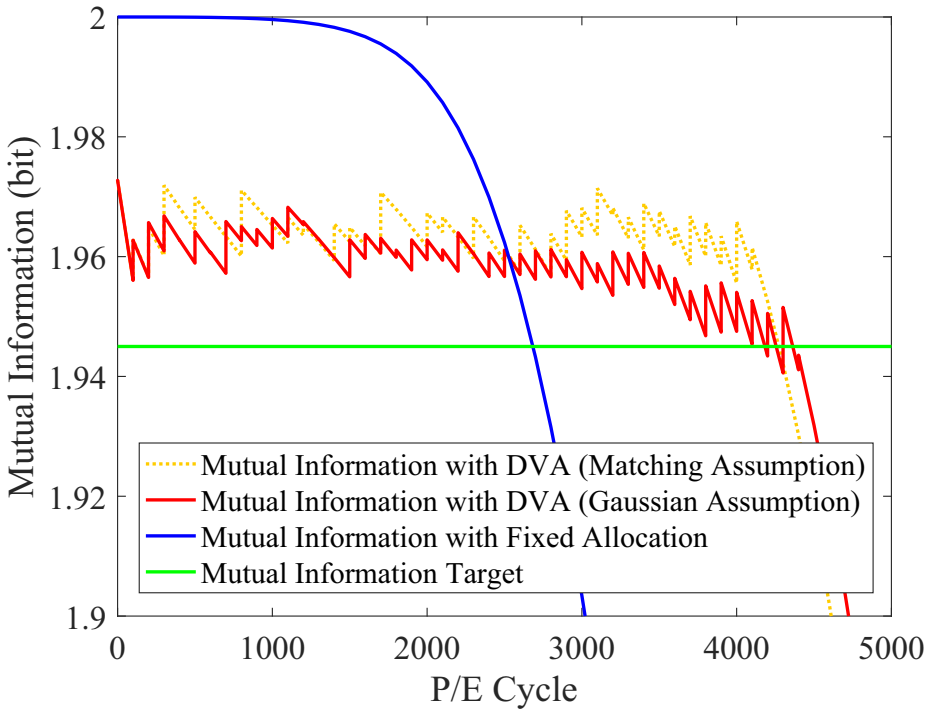


Figure 2.11: DVA's performance with multi-modal Gaussian model. (Ground truth model is Model 1. Channel parameters are listed in Sec. 2.10.)

variances and means (or biases of means to be precise) of the distribution regardless of the numerous parameters used to create the actual channel used in the simulation.

2.5.1 Model 1

Because this ground truth channel model is still very similar to the multi-modal Gaussian model, DVA can effectively extend Flash memory's lifetime although the channel model assumption is different than the ground truth. Fig. 2.11 shows the Monte Carlo simulation performance of DVA using a multi-modal Gaussian when Model 1 is the ground truth model. Here the lifetime is extended by 55.9% from 2683 P/E cycles to 4182 P/E cycles.

Note that unlike in Fig. 2.5, the *actual* mutual informations shown in Fig. 2.11 are not reset to exactly 1.965 bits after every DVA update. For the "Matching Assumption" case where the DVA channel model is the actual channel model, the variation in the mutual information after reset stems entirely from noise in the channel parameter estimation. For

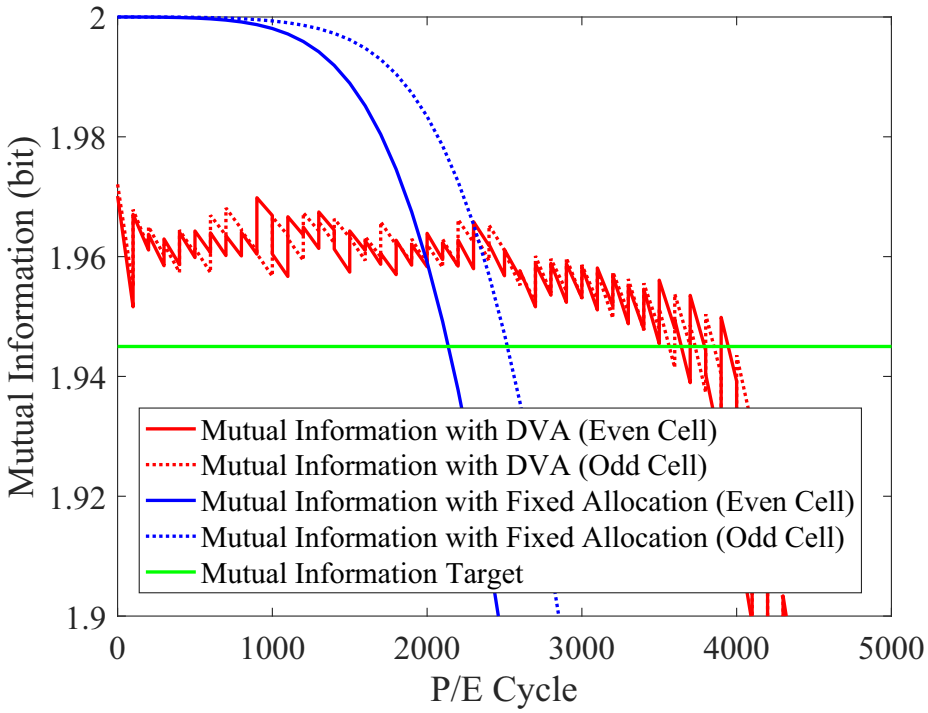


Figure 2.12: DVA performance with multi-modal Gaussian model. (The Ground truth model is Model 2. Channel parameters are listed in Sec. 2.10. The lifetime of even cells is extended by 70.5% from 2136 P/E cycles to 3642 P/E cycles. The lifetime of odd cells is extended by 41.6% from 2517 P/E cycles to 3564 P/E cycles. The overall lifetime is extended by 66.9% from 2136 P/E cycles to 3564 P/E cycles.)

the "Gaussian Assumption" case, there is a large variation, which stems from parameter estimation error *and* from the fact that the DVA algorithm chooses the scale factor that sets the mutual information of the multi-modal Gaussian channel to 1.965 bits, but the mutual information of the actual channel turns out to be different. In particular, the mutual information of the actual channel is noticeably less than that of the multi-modal Gaussian towards the end of the device lifetime as the Gaussian assumption becomes less accurate. Still, the overall DVA approach is successful because the 0.02 bits of additional margin was sufficient to overcome the mismatch error.

2.5.2 Model 2

For Model 2, the multi-modal Gaussian channel model assumption can only be considered as a rough approximation, but DVA can still provide a significant performance increase using this simple model. Fig. 2.12 shows the Monte Carlo simulation performance of DVA using the multi-modal channel model assumption while generating noise using Model 2. Because the only channel information available is the measured histograms from even and odd cells, DVA has to rely on channel estimation. The channel estimation only provides the estimated means and variances of the multi-modal Gaussian model rather than the parameters in the actual channel model.

The estimations of even-cell and odd-cell channels provide approximated channel characteristics for the odd and even channels respectively, when the write order is switched. Based on this observation, the scaling factors of the two channels can be updated based on the channel estimation of the other channel using DVA. This implementation is an approximation of the joint DVA approach in Sec. 2.3. The simulation result shows that DVA can extend the overall lifetime by 66.9% from 2136 P/E cycles to 3564 P/E cycles for this channel.

Here, it is perhaps surprising that the overall lifetime is improved more by DVA employing the simple Gaussian model than by DVA using the more accurate model and prefect channel knowledge in Fig. 2.5. The Gaussian model under-estimates the severity of the channel which causes it to use a smaller value of α that turns out to still provide sufficient mutual information. This extends lifetime a bit more by reducing the amount of charge written. Of course we could lower the amount of margin used in the simulation of Fig. 2.5 and also gain this benefit.

2.6 Comparison of Dynamic Voltage Allocation and Dynamic

Threshold Assignment

Both dynamic voltage allocation and dynamic threshold assignment (DTA) [27, 28] track the degradation of Flash memory channel during its lifetime. DTA allocates appropriate *read* threshold voltages under different channel conditions to reduce the asymmetric errors caused by distribution shifting and widening. DVA allocates appropriate *write* threshold voltages under different channel conditions to adjust the mutual information to a sufficient (but not extravagant) level. DVA depends on the preset target mutual information which relates to the capability of error correction codes employed. Considering Flash memory as a M-ary baseband Pulse Amplitude Modulation (MPAM) communication system (4PAM in this chapter), DTA provides hard decision thresholds closer to the optimal ones than fixed read thresholds, and DVA adjusts the average symbol power of the constellation to match channel conditions.

Assume the simplest fixed-read-voltage allocation schemes where hard decision boundaries are put in the midpoints between adjacent write threshold voltages, and DTA implementation from [28], simulations are conducted to compare the performance of DTA alone and DTA combined with DVA. Fig. 2.13 compares the raw BER (without error correction code) of reading with a fixed voltage allocation, DTA and DVA. The dash line in the figure represents the raw BER requirement for the LDPC code in [17] to function properly with soft information from 6 reads (Fig. 11 in [17]). To accommodate the rapid channel degradation at the beginning of the lifetime and meet the 10^{-2} raw BER target, the scaling factor for both even and odd cells are lower bounded by 0.45.

Because the retention time is fixed to be one year in our simulations and there is no operation on the cells during this period, the stored voltage values in the cells suffer relatively significant shifts. As a result, fixed read voltage allocation provides very bad BER performance. With DTA, the read thresholds are able to track the channel degradation, thus produces BER performance consistent with the channel capacity using fixed write voltage

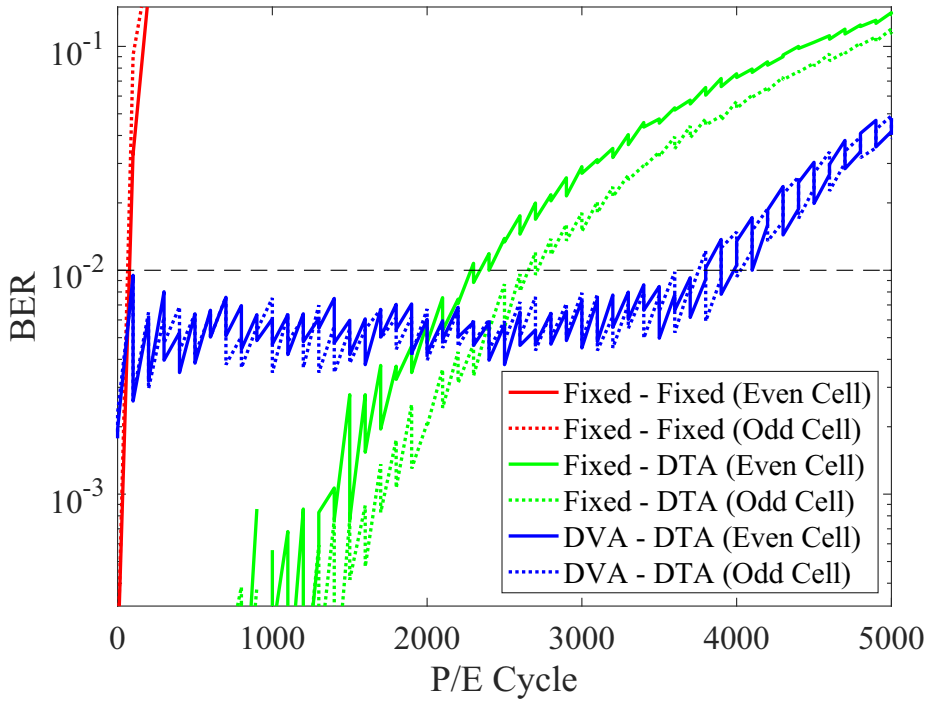


Figure 2.13: DTA and DVA's performance. (Even and odd cells switch positions when using DVA. The legend follows the format *write voltage allocation algorithm - read voltage allocation algorithm*. Comparing the result from DVA - DTA with Fixed - DTA, The lifetime of even cells is extended by 66.5% from 2282 P/E cycles to 3800 P/E cycles. The lifetime of odd cells is extended by 41.4% from 2654 P/E cycles to 3754 P/E cycles. The overall lifetime is extended by 64.5% from 2282 P/E cycles to 3754 P/E cycles.)

allocation. DVA allocates write thresholds to maintain a reasonable channel capacity, so a non-zero low level raw BER appears from the beginning of the device's lifetime, and it is expected to be easily corrected by error correction codes. However, this is in contrast to DTA without DVA, where almost no errors occur in the first 1000 P/E cycles.

To achieve good raw BER performance, DVA requires that DTA-like algorithms are used in conjunction to provide decision boundaries that adapt to the changing write levels. Even without DVA, DTA-like algorithms are a practical necessity in order to track the movements of the channel distribution caused by channel degradations such as retention loss. Fig. 2.13 shows that DVA and DTA combined provide a significant performance improvement over DTA alone. The results indicate that DVA with DTA can extend the overall lifetime

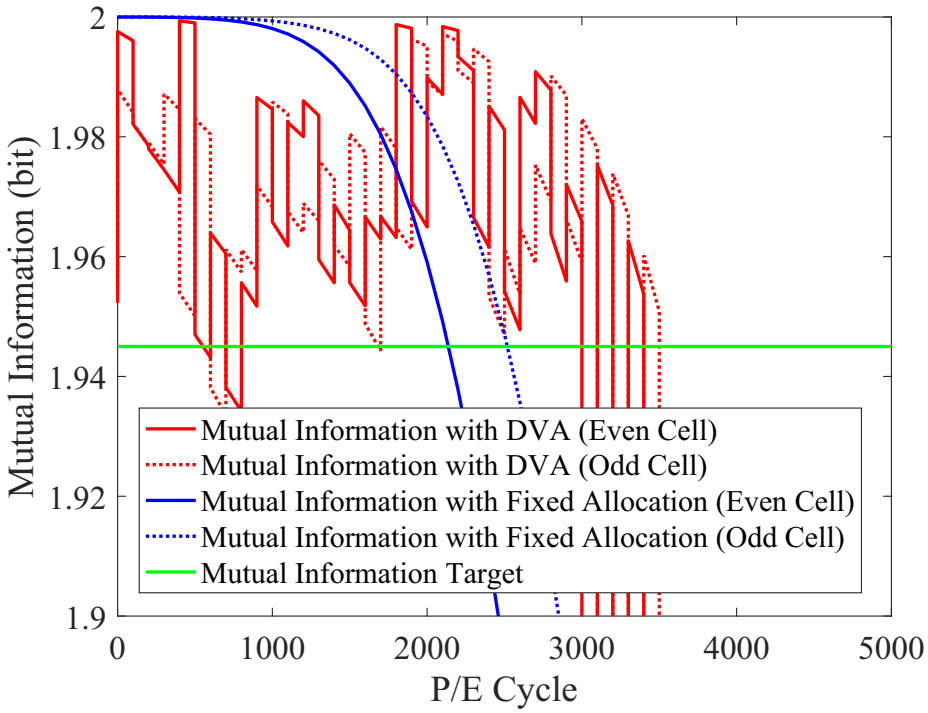


Figure 2.14: DVA performance with quantized placements. (Quantization provides 64 possible locations. Ground truth model is Model 2. Channel parameters are listed in Sec. 2.10.)

by 64.5% from 2282 P/E cycles to 3754 P/E cycles for this ground truth channel model compared with a fixed write-voltage allocation with DTA.

2.7 Dynamic Voltage Allocation when Voltage Placements are Quantized

In this section, we explore the performance of the DVA framework when voltage placements are limited to certain locations. In practice, hardware implementations introduce the constraint that both the read reference voltages and the intended write threshold voltage levels can only be set to certain values. This will add an additional constraint to both the intended threshold voltages and the read reference voltages, which affects DVA performance.

Monte Carlo simulation of DVA under several commonly used voltage quantization scenarios (64, 128 and 256-location quantization for the entire available voltage range) are

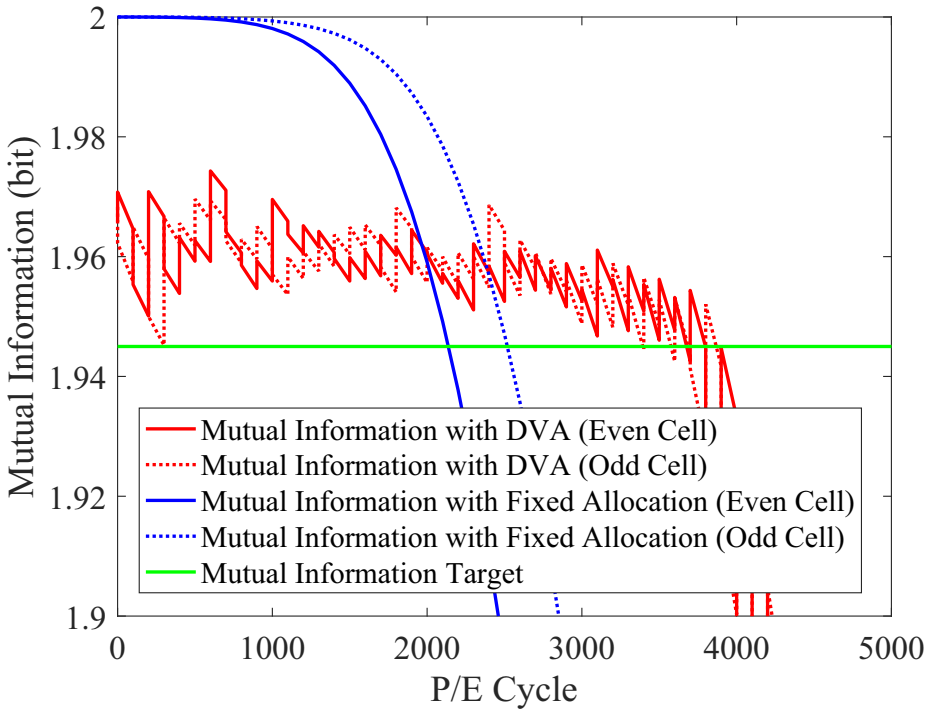


Figure 2.15: DVA's performance with quantized placements. (Quantization provides 128 possible locations. Ground truth model is Model 2. Channel parameters are listed in Sec. 2.10. The lifetime of even cells is extended by 72.1% from 2136 P/E cycles to 3676 P/E cycles. The lifetime of odd cells is extended by 34.8% from 2517 P/E cycles to 3393 P/E cycles. The overall lifetime is extended by 58.8% from 2136 P/E cycles to 3393 P/E cycles.)

conducted. We assume adjacent potential placement locations are separated by a constant distance, which corresponds to equal interval sampling over a certain voltage range. The quantization range is set to be from -1 Volt to 8 Volts. Figs. 2.14 and 2.15 show the mutual information versus P/E cycle curve when limiting the number of possible locations 64 and 128. Quantization with 256 possible locations results a performance very similar to Fig. 2.12. Quantization with 128 possible locations strikes a nice balance between DVA performance and the number of placement locations. When using 128-level quantization, the overall lifetime is extended by 58.8% from 2136 P/E cycles to 3393 P/E cycles.

Notice that the DVA framework often produces mutual information values that are too high (hence leading to a faster increase in V_{acc} and faster degradation) when the number of possible placement locations is constrained to 64. We also observed that under this condition,

the quantized read threshold voltage placements also suffered. Even though only nine reads are performed, with only 64 possible values, two or more would be in the same place. As a result, the effective resolution of the measured histogram is reduced, and the channel parameter estimation algorithm becomes less accurate and degrades DVA performance.

2.8 Complexity Analysis

The DVA framework consists of two processes: channel parameter estimation and scaling factor adjustment. This section considers complexity for the case discussed in Sec. 2.5 where we use a multi-modal Gaussian distribution as the channel model for both channel parameter estimation and scaling factor adjustment. For the channel parameter estimation process, there are two steps: read threshold voltage assignment and iterative Levenberg-Marquardt parameter estimation. The computational complexity of DVA can be analyzed according to these three primary components: read threshold voltage assignment, iterative Levenberg-Marquardt parameter estimation, and scaling factor adjustment.

Read threshold voltage assignment is used to provide boundaries for the bins in the histogram satisfying the equal-probability binning strategy discussed in Sec. 2.4.3. In the DVA framework, this assignment is calculated using a Gaussian channel model and the estimated channel parameters (means and variances) from the parameter estimation process 100 P/E cycles ago. Integration of a Gaussian probability distribution function is needed, but the operation is in essence the evaluation of the Q-function (tail probability of the standard normal distribution), and can be implemented with a look-up table. The exact position of each read is calculated using bisection method. Table 2.2 summarizes the number of additions (ADD), divisions (DIV) and table lookups (LUT) in each iteration, and the number of iterations required to calculate each read threshold voltage.

The computational complexity of using the Levenberg-Marquardt algorithm based on the Gaussian channel model assumption to estimate the Flash read channel is carefully analyzed in [30]. The results are summarized here in Table 2.3. Note that the $\lceil \frac{N^3}{3} \rceil$ term in the table

Table 2.2: Computational complexity of read threshold voltage allocation.

No. of ops. per iter.	ADD	DIV	LUT	No. of iterations
Boundary 1 (leftmost)	9	4	4	$\log_2 \frac{V_{r,max}-V_{r,min}}{\epsilon}$
Boundary i, $2 \leq i \leq N-1$	9	4	4	$\log_2 \frac{V_{r,max}-V_{r,i-1}}{\epsilon}$

Note: "ops." is the abbreviation for "operations". "iter." is the abbreviation for "iteration". The number of bins in the histogram is N. The ith boundary is $V_{r,i}$. The maximum and minimum possible read threshold voltage is $V_{r,max}$ and $V_{r,min}$ respectively. The tolerance (error) of the bisection algorithm is ϵ. ADD, DIV, LUT represent addition, division and table lookup operation respectively.

Table 2.3: Computational complexity of Levenberg-Marquardt algorithm.

	ADD	MUL	DIV	LUT
No. of ops. per iter.	$93N + \lceil \frac{N^3}{3} \rceil$	$89N + \lceil \frac{N^3}{3} \rceil$	$4N + \lceil \frac{N^3}{3} \rceil$	$8N$

Note: "ops." is the abbreviation for "operations". "iter." is the abbreviation for "iteration". The number of bins in the histogram is N. The tolerance (error) of the bisection algorithm is ϵ. ADD, MUL, DIV, LUT represent addition, multiplication, division and table lookup operation respectively.

comes from the inversion operation of an $N \times N$ matrix using the Gauss-Jordan method, where N is the number of bins in the measured histogram. Because the Levenberg-Marquardt algorithm has multiple stopping criteria, the exact number of iterations needed is not a fixed value for different channel conditions. In our simulations, we observe that the maximum number of iterations is less than 100.

The scaling factor adjustment process uses a bisection algorithm to search for the smallest scaling factor that achieves the mutual information target given the estimated means and variances. Table 2.4(a) summarizes the number of extra additions (ExADD), multiplications (ExMUL) other than the ones in the mutual information calculation, the number of mutual information calculations (MI) in each iteration, and the number of iterations required to calculate the scaling factor. The most computation-heavy operation in this process is the calculation of the mutual information (MI) of a multi-modal Gaussian distribution, which involves integration operations. Assuming the Gauss-Hermite quadrature [51] method is used to approximate the integration, the mutual information calculation needs to be represented

Table 2.4: Computational complexity of scaling factor adjustment.

(a) Number of operations per iteration

ExADD	ExMUL	MI	No. of iterations
5	4	1	$\log_2 \frac{\alpha_{max}-\alpha_{min}}{\epsilon}$

(b) Number of operations per MI

ADD	MUL	DIV	LUT
$44n-1$	$88n+1$	$42n$	$8n$

Note: The maximum and minimum possible scaling factor are α_{max} and α_{min} respectively. The tolerance (error) of the bisection algorithm is ϵ. ExADD, ExMUL, MI represent extra addition, extra division in additional to the calculation of mutual information, and mutual information calculation respectively. The number of sample points used in MI calculation is n. ADD, MUL, DIV, LUT represent addition, multiplication, division and table lookup operation respectively.

in the form of

$$\int_{-\infty}^{\infty} e^{-x^2} f(x) \mathrm{d}x \ , \tag{2.24}$$

where $f(x)$ is a function. Then the following approximation can be used:

$$\int_{-\infty}^{\infty} e^{-x^2} f(x) \mathrm{d}x \ \approx \sum_{i=1}^{n} w_i f(x_i) \ , \tag{2.25}$$

where x_i's are the roots of the Hermite polynomial $H_n(x)$, w_i's are the weights calculated by

$$\frac{2^{n-1} n! \sqrt{\pi}}{n^2 [H_{n-1}(x_i)]^2} \ , \tag{2.26}$$

and n is the number of sample points. The accuracy of the approximation increases with n. Following (2.15-2.17), the new mutual information representation is

$$I(X,Y) = -\frac{1}{4\sqrt{\pi}} \sum_{i=1}^{4} \int_{-\infty}^{\infty} e^{-z_i^2} \log(g(z_i)) \mathrm{d}z_i \ , \tag{2.27}$$

where

$$g(z_i) = \frac{1}{4} + \sum_{\substack{j=1 \\ j \neq i}}^{4} \frac{\sigma_i}{4\sigma_j} e^{z_i^2 - \frac{(\sqrt{2}\sigma_i z_i + x_i - x_j)^2}{2\sigma_j^2}} \quad . \tag{2.28}$$

Assuming all the weights can be pre-calculated, and $\log(\cdot), e^{(\cdot)}$ can be calculated with lookup tables, the number of operations needed for each MI calculation is shown is Table 2.4(b).

2.9 Conclusion

This chapter introduces a framework to extend Flash memory lifetime by dynamic allocation of intended threshold voltage levels based on the current mutual information of the Flash read channel. Analysis and simulation results demonstrate that this framework can provide significant lifetime extension in practical settings. Simple channel models can be used to estimate a complex Flash channel, and optimize the voltage placements. Good performance can be achieved even when the placement of voltages is constrained to 128 possible values. The frame can be modified to reduce computational complexity while providing comparable performance to a fully implemented system.

The results in this chapter can be improved by extending the ground truth channel models to include additional mechanisms for channel degradation in Flash memory and by using data measured from actual Flash devices. Another direction to further develop the DVA framework is to explore using raw BER as the optimization target for the DVA framework.

2.10 Channel Parameters Used in This Chapter

Table 2.5: Channel parameters used in this chapter.

Programming Noise [29]			
σ_e	0.35	σ_p	0.05
Wear-out Noise [29]			
A_w	1.8×10^{-4}	C_w	1.26×10^{-3}
k_i	0.62	——	——
Retention Noise [29]			
A_r	7.0×10^{-4}	B_r	4.76×10^{-3}
k_i	0.62	k_o	0.3
t_0	1(hour)	t	8760
Cell-to-cell Interference [42]			
$E\{\gamma_x\}$	$0.1s$	$E\{\gamma_y\}$	$0.08s$
$E\{\gamma_{xy}\}$	$0.006s$	s	0.2
w_k	$0.2E\{\gamma_k\}$	σ_k	$0.3E\{\gamma_k\}$
Programming Error [32]			
$P_{0,2}$	$\exp(0.87x\text{-}11.89)$	$P_{0,3}$	$\exp(1.41x\text{-}19.82)$
$P_{1,2}$	$\exp(1.63x\text{-}19.22)$	$P_{1,3}$	$\exp(0.73x\text{-}11.67)$
$P_{2,3}$	$\exp(1.50x\text{-}17.69)$	——	——

Default intended threshold voltages [29]: $[2.8, 5.2, 6.4, 7.86]$. $V_{max} = 16$. [29]

2.11 Acknowledgment

The majority of this chapter has been published in [2, 3, 4]. This research is conducted in collaboration with Nathan Wong, Dr. Tsung-Yi Chen, and Prof. Richard D. Wesel. The author would like to thank Dr. Tsung-Yi Chen for proposing the idea of DVA in [29]. The author also would like to thank Nathan Wong for his help in relevant simulations.

CHAPTER 3

Coding with Shared Incremental Redundancy

3.1 Introduction

Feedback communication using incremental redundancy (IR), such as IR-hybrid automatic repeat request (HARQ) or Type-II HARQ [52, 53], is widely used in deployed systems to increase throughput and provide high reliability [54, 55, 56]. While the use of IR with feedback has long been known to improve the error exponent of communication systems [57, 58], Polyanskiy et al. [59] showed that systems with feedback can achieve near-capacity throughput with significantly shorter average blocklengths than systems without feedback. Recent papers [60, 61] provide examples of variable-length (VL) error correction codes that can be used with a few rounds of feedback to exceed the random-coding lower bounds of [59] and approach capacity with average codeword lengths smaller than 500 bits. Separately, Zeineddine and Mansour [62] introduced inter-frame coding, which uses numerous VL codes in parallel and without feedback to address varying channel-state conditions in broadcast wireless communication.

This chapter introduces coding with shared IR in a point-to-point communication system [5, 6, 7], which builds on the inter-frame coding of [62]. The system described in this chapter does not benefit from feedback directly in the form of requesting IR from a transmitter. The IR of multiple codewords are compressed by the inter-frame code into a *common pool of redundancy* that is transmitted through the channel. At the receiver, the received shared

IR delivers varying numbers of IR increments locally to each VL decoder according to their need without the participation of the transmitter. In other words, the "feedback" process as is defined in a feedback communication system only happens inside the receiver. The system provides a method to design a code with very long block length, which can approach capacity essentially by using numerous short-blocklength encoders and decoders. In our systems, the inter-frame decoder employs a peeling process, which has been applied in a similar way for multiple access channels [63, 64, 65, 66].

Thus, the proposed system can approach capacity without feedback but with a complexity comparable to decoding VL codes with feedback. Much of the calculation at the encoder and decoder can be distributed to a large number of parallel VL encoders and decoders for the individual VL codes, and the VL codes have short average blocklengths. Only a relatively small number of bit-by-bit exclusive-ors (XOR, $\oplus$) required to implement the inter-frame code are performed outside of the parallel encoders at the transmitter. Similarly, the peeling decoder that provides incremental redundancy to VL decoders at the receiver has a complexity that is low relative to the complexity of the parallel VL decoders themselves.

Flash-based storage systems is a point-to-point communication system, and usually require codes with very long blocklength. Traditional code designs for Flash such as Bose-Chaudhuri-Hocquenghem (BCH) codes [11, 12, 13] and low density parity check (LDPC) codes [14, 15, 16, 17] provide very good error correction capabilities, but suffer from high complexity at long blocklength. Coding with shared IR provides another long blocklength code design that has high throughput, good error correction performance, and a parallel system structure that reduces the complexity in practical system implementations.

This chapter presents our design concept, provides design examples and performance analysis in the following ways:

1. Summarizing of the system structure of coding with shared IR;

2. Analyzing the convergence property of the generalized peeling decoder (GPD) which retrieves IR for individual VL codewords from the shared IR;

3. Introducing a general inter-frame coding design method based on differential evolution;

4. Introducing a decoder-analysis-inspired inter-frame coding design method that has a low design complexity;

5. Presenting system designs using convolutional VL code as the base code, and analyzing their performance;

6. Presenting system designs using non-binary low density parity check (NB-LDPC) with binary IR as the base VL code, and analyzing their performance.

In this chapter, we design and analyze the proposed system on the 2dB binary-input additive white Gaussian noise channel (BI-AWGN) as an example. The design methods presented in this chapter are not restricted to specific channel conditions and can be directly applied to other channels.

The chapter is organized as follows: Sec. 3.2 presents the overall architecture based on the inter-frame coding technique of [62]. Sec. 3.3 provides a framework with which to analyze the convergence of a generalized peeling decoder (GPD). Sec. 3.4 describes our convolutional VL code and the bit-by-bit VL design for the NB-LDPC code, and presents a differential evolution and quasi-regular design methods for the inter-frame bipartite graph. Sec. 3.5 presents the performance analysis of system design examples with the convolutional VL code and with the NB-LDPC VL code. Sec. 3.6 concludes the chapter. Sec. 3.7 presents the acknowledgment.

3.2 Architecture

Inter-frame coding [62] allows a set of parallel VL codewords to match their rates to the to the observed channel distortion without the use of feedback. All frames share a common pool of redundancy comprised of linear combinations of increments, each from a different frame, that enable rate-matching for each frame though a peeling decoder.

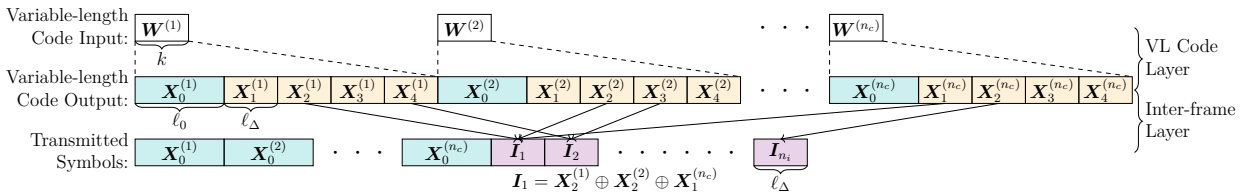


Figure 3.1: Inter-frame encoder structure. $\boldsymbol{W}_i$ is a k-bit message, $\boldsymbol{X}_0^{(i)}$ is a length-ℓ_0 vector, and $\{\boldsymbol{X}_1^{(i)}, \ldots, \boldsymbol{X}_4^{(i)}\}$ and $\boldsymbol{I}_j$ are length-ℓ_Δ vectors.

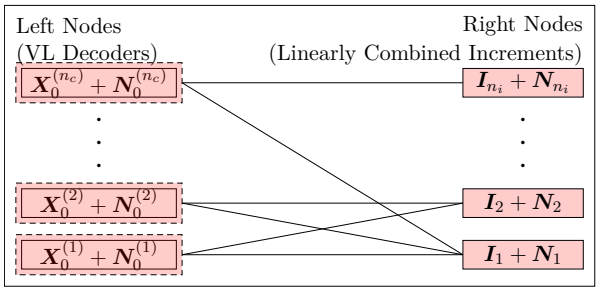
Two layers comprise the inter-frame coding system: the VL code layer and the inter-frame layer. Fig. 3.1 shows how input messages are transformed into transmitted symbols. The VL code layer has n_c VL encoders in parallel. Each VL encoder receives as input a message $\boldsymbol{W}_i$ of length k symbols and produces as output an initial transmission $\boldsymbol{X}_0^{(i)}$ having length ℓ_0 symbols, and a series of $m-1$ increments[1] $\{\boldsymbol{X}_1^{(i)}, \ldots, \boldsymbol{X}_{m-1}^{(i)}\}$ each having length ℓ_Δ symbols. In Fig. 3.1, there are $m = 5$ total outputs per VL encoder, an initial transmission and four increments.

The inter-frame layer produces the common pool of incremental redundancy for transmission by linearly combining (bit-by-bit XOR) increments $\boldsymbol{X}_j^{(i)}$ where $1 \leq j \leq m-1$ and no two increments with the same value of i, i.e., from the same VL encoder, are involved in the same linear combination. Every increment $\boldsymbol{X}_j^{(i)}$ for $1 \leq i \leq n_c$ and $1 \leq j \leq m-1$ is involved in exactly one linear combination.

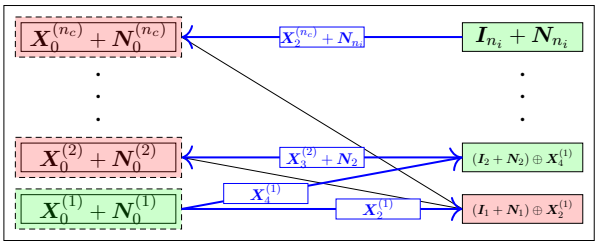
The inter-frame coding architecture generates a single long-blocklength super-codeword that encodes all n_c messages $\boldsymbol{W}_i$. This super-codeword includes the initial transmissions $\boldsymbol{X}_0^{(i)}$ for each VL encoder *and* the linear combinations $\boldsymbol{I}_j$ that comprise the common incremental redundancy. The n_i transmitted increments $\boldsymbol{I}_1, \cdots, \boldsymbol{I}_{n_i}$ form the common pool of incremental redundancy.

Following the definition of a bipartite graph in [67], if we consider each VL codeword as a left node, and each combined increment as a right node, a bipartite graph can describe the inter-frame layer. Because both the initial transmission $\boldsymbol{X}_0^{(i)}$'s and the linear combination of

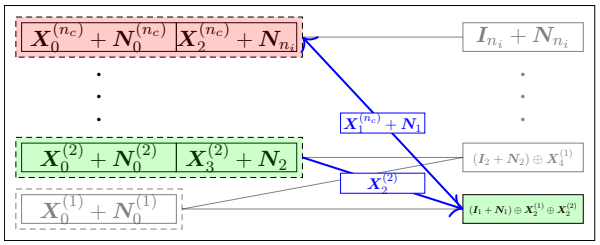
[1]In [62] the VL encoders can have a varying number of increments, even approaching an infinite number according to the analytically constructed degree distributions. In this chapter we assume that a practical system will have a fixed number $(m-1)$ of increments produced by each VL encoder.



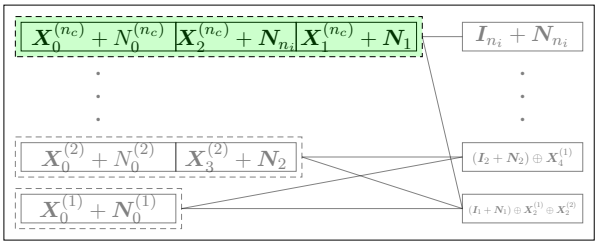
(a) Initialization: Each left node receives an initial transmission $\boldsymbol{X}_0^{(i)}$ distorted by channel noise $\boldsymbol{N}_0^{(i)}$. Each right node receives a distorted linear combination of increments $\boldsymbol{I}_i + \boldsymbol{N}_i$.



(b) Iteration 1: One the left side, the first (bottom) node decodes successfully with only $\boldsymbol{X}_0^{(1)} + \boldsymbol{N}_0^{(1)}$, and provides its increments $\boldsymbol{X}_2^{(1)}, \boldsymbol{X}_4^{(1)}$ to incident right nodes. The other left nodes failed to decode. On the right side, the first (bottom) node removes $\boldsymbol{X}_2^{(1)}$ from $\boldsymbol{I}_1 + \boldsymbol{N}_1$, and becomes a degree-two node $(\boldsymbol{I}_1 + \boldsymbol{N}_1) \oplus \boldsymbol{X}_2^{(1)}$. The second right node removes $\boldsymbol{X}_4^{(1)}$, and provides an increment with noise $\boldsymbol{X}_3^{(2)} + \boldsymbol{N}_2$ to the second left node. The n_i^{th} (top) right node, as a degree-1 right node, directly provides $\boldsymbol{X}_2^{(n_c)} + \boldsymbol{N}_{n_i}$ to the n_c^{th} (top) left node. In each iteration, all nodes that provide increments to other nodes are removed along with their incident edges.



(c) Iteration 2: On the left side, the second node extends its cumulative codeword with $\boldsymbol{X}_3^{(2)} + \boldsymbol{N}_2$ received in the first iteration and decodes successfully. It provides $\boldsymbol{X}_2^{(2)}$ to the first right node. The n_c^{th} left node extends its codeword with $\boldsymbol{X}_2^{(n_c)} + \boldsymbol{N}_{n_i}$ and still fails to decode. On the right side, the first node removes $\boldsymbol{X}_2^{(2)}$ from its linear combination and provides $\boldsymbol{X}_1^{(n_c)} + \boldsymbol{N}_1$ to the n_c^{th} left node.



(d) Iteration 3: the n_c^{th} left node extends its cumulative codeword with $\boldsymbol{X}_1^{(n_c)} + \boldsymbol{N}_1$, and decodes successfully.

Figure 3.2: Decoding process of an inter-frame coding system with the inter-frame layer represented as a bipartite graph.

increments $\boldsymbol{I}_i$'s are transmitted through the channel, the inter-frame layer is also functionally similar to a low-density generator matrix (LDGM) code [68].

The decoder has a corresponding two-layer structure including a VL decoding layer and an inter-frame decoding layer, which is essentially a peeling decoder for the LDGM code. Fig. 3.2 shows a bipartite graph representation of the decoding process corresponding to the encoder in Fig. 3.1. The left nodes in the bipartite graph describe the VL code layer, where each node corresponds to a VL decoder. The right nodes describe the inter-frame layer, where each node corresponds to a linear combination $\boldsymbol{I}_j$.

In Fig. 3.2, red identifies a codeword or increment for which decoding has not yet succeeded and green identifies a successful decoding. The left nodes are initially populated with the noisy initial transmissions $\boldsymbol{X}_0^{(i)} + \boldsymbol{N}_0^{(i)}$ as the VL decoder inputs. The right nodes are initialized with the received linear combinations $\boldsymbol{I}_i + \boldsymbol{N}_i$. In each iteration, the left nodes try decoding with their available inputs, and some will succeed. The VL decoders that succeed generate their remaining increments $\boldsymbol{X}_j^{(i)}$ and provide them to the neighboring right nodes. Each right node cancels the provided increments from $\boldsymbol{I}_i + \boldsymbol{N}_i$ through XOR operations. When a right node is able to cancel all but one of its increments, it provides the last remaining increment $\boldsymbol{X}_j^{(i)}$ with noise to the i^{th} VL decoder, providing a longer cumulative received codeword for that VL decoder to process.

3.3 Generalized Peeling Decoder Analysis

Inter-frame decoding is an iterative peeling process. After left nodes have been decoded or right nodes have provided an increment, the nodes and their incident edges are removed. A message in the inter-frame graph is a *vector* of ℓ_Δ reliability values, i.e., an increment having length ℓ_Δ symbols, whereas messages in standard LDPC/LDGM graphs are *scalar* values.

Following the definition in [67], define the left edge degree distribution polynomial as $\lambda(x) = \sum_{i=1}^{d_L} \lambda_i x^{i-1}$. Coefficient λ_i is the fraction of left-degree-i edges among all edges, and d_L is the maximum left edge degree. Similarly, define the right edge degree distribution as $\rho(x) = \sum_{i=1}^{d_R} \rho_i x^{i-1}$, where d_R is the maximum right degree, and ρ_i is the fraction of right degree i edges. Note that $\rho(x)$ can be expressed as an inner product of a coefficient vector $\boldsymbol{\rho}^{(\text{coeff})}$ and vector of indeterminant powers $[1, x, x^2, \ldots]$. For compact notation, we will describe $\rho(x)$ by providing $\boldsymbol{\rho}^{(\text{coeff})}$.

Success hinges on whether the inter-frame layer can provide each VL decoder with all of the increments it needs. This section provides an analytical framework to determine when the inter-frame layer will be successful in this mission by analyzing the evolution of the number of "available increments" as represented by the variable r_1. While our equation for

r_1 matches results in [67, 69, 62], our derivation follows directly from probability arguments rather than differential equations or a tree-based approach.

The overall inter-frame coding architecture can, at best, allow a VL decoder to perform as if it receives increments as a result of acknowledgment/negative acknowledgment (ACK/NACK) feedback. That feedback performance can be characterized by a probability mass function (PMF) $\boldsymbol{\delta} = [\delta_0, \delta_1, \cdots, \delta_m]$. Here, δ_0 is the probability of correctly decoding the VL codeword with only the initial transmission $X_0^{(i)}$. The probabilities $\delta_j, j = 1, \cdots, m-1$ correspond to the VL decoder succeeding *for the first time* after adding $X_j^{(i)}$ to the cumulative received codeword. The probability δ_m corresponds to failure to recover the message even after receiving all m transmissions, i.e., all $m-1$ increments.

3.3.1 Generalized Peeling Decoder (GPD)

In contrast to the standard peeling decoder in which any received increment removes the incident left node and all its edges, whether the GPD removes a left node and its edges depends on the current state s of the left node. Each left node in the bipartite graph has initial state $s_0 \in \{0, \cdots, m\}$ with probability δ_{s_0}, where s_0 represents the number of increments required by the VL decoder to successfully decode for the first time. The PMF $\boldsymbol{\delta}$ is induced by the specific channel, i.e., a specific signal to noise ratio for an AWGN channel, and represents the stochastic behavior of a VL decoder on that channel. Note that when the initial left node state is $s_0 = m$ and there are only $m-1$ increments possible, the VL decoder is doomed to fail.

The standard peeling decoder on the binary erasure channel with erasure probability p is a special case of the GPD with $m = 2$ and $\boldsymbol{\delta} = [1-p, p, 0]$.

Define the original bipartite graph of the inter-frame layer (e.g. Fig. 3.2a) as graph B with n_c left nodes, n_i right nodes, ratio $\beta = n_i/n_c$ of right nodes to left nodes, and E edges. Let G_0 be the graph that results from removing left nodes with initial state $s_0 = 0$ and their incident edges. G_0 is the graph in Fig. 3.2b that results from removing the left node

$\boldsymbol{X}_0^{(1)} + \boldsymbol{N}_0^{(1)}$ and its two edges $\boldsymbol{X}_2^{(1)}$ and $\boldsymbol{X}_4^{(1)}$.

Define Q_t be the randomly chosen t^{th} right-degree-one edge (available increment) used to lower the state of a left node. Looking at Fig. 3.2b, Q_1, which is the first edge removed, could be either of the edges $\boldsymbol{X}_3^{(2)} + \boldsymbol{N}_2$ or $\boldsymbol{X}_2^{(n_c)} + \boldsymbol{N}_{n_i}$. Q_t is called interchangeably an edge or an increment, because it is both.

If the state of a left node decreases to zero, i.e., the corresponding VL decoder succeeds, it is removed along with its remaining incident edges in G_{t-1} to produce G_t. For example, if Q_1 is $\boldsymbol{X}_2^{(n_c)} + \boldsymbol{N}_{n_i}$ in Fig. 3.2b, then G_1 is the result of removing the right node $\boldsymbol{I}_{n_i} + \boldsymbol{N}_{n_i}$ and its single incident edge $\boldsymbol{X}_2^{(n_c)} + \boldsymbol{N}_{n_i}$. The left node incident to the removed edge $\boldsymbol{X}_2^{(n_c)} + \boldsymbol{N}_{n_i}$ would have been removed along with all of its remaining incident edges if the VL decoder had succeeded with the additional increment $\boldsymbol{X}_2^{(n_c)} + \boldsymbol{N}_{n_i}$, but that did not happen in this example.

3.3.2 Computation of $r_1(t)$ a.k.a. $r_1(x)$

The peeling process can continue only as long as the next Q_t can be found. The fraction $r_1(t)$ of edges in G_t that are incident to a degree-one right node measures the availability of edges to serve as the next Q_t. This fraction is computed using E, the number of edges in the original graph B, as the denominator. Thus $r_1(t)$ can be understood to be the probability that an edge selected at random from B is incident to a degree-one right node in G_t.

Define $x(t)$ or simply x as the probability that a randomly selected edge in the original graph B is not in the set $Q_1, \cdots, Q_t$. The re-use of the symbol x, which is also the indeterminate of the degree distribution polynomials is intentional, as will become clear. We will interchangeably refer to the probability $r_1(t)$ as $r_1(x)$. These representations are equivalent because x is a bijective function of t and the probability x completely characterizes the state of the random graph G_t as it affects the value of r_1. For this reason, we will sometimes use x and t interchangeably in the following discussion. In order to derive $r_1(x)$, the following intermediary probabilities are needed:

- $p_l(x)$: This is probability that a randomly selected edge in B satisfies the event that it is incident to a *left* node with state $s > 0$ with $Q_1, \cdots, Q_t$ provided as increments to the *other* edges connecting to that left node.

- $p_r(x)$: This is probability that a randomly selected edge in B satisfies the event that it is incident to a *right* node with none of its *other* edges remaining after $Q_1, \cdots, Q_t$ have been provided as increments to the left nodes in G_0.

Computing $p_l(x)$

To calculate $p_l(x)$, we first need to calculate $p_l(x|i, s_0)$, where i is the initial left degree of the randomly selected edge from B, and s_0 is the initial state of its incident left node. If $s_0 > i - 1$, $p_l(x|i, s_0) = 1$ because the VL decoder will never receive enough increments to decode. If $s_0 = i - 1$, $p_l(x|i, s_0) = 1 - (1 - x)^{s_0}$. If $s_0 < i - 1$,

$$p_l(x|i, s_0) = \sum_{j=0}^{s_0-1} \binom{i-1}{j} (1-x)^j x^{i-1-j} \ . \tag{3.1}$$

Combine the three scenarios,

$$p_l(x|i, s_0) = \sum_{j=0}^{\min(s_0,i)-1} \binom{i-1}{j} (1-x)^j x^{i-1-j} \ , \tag{3.2}$$

for $s_0 > 0$ and $p_l(x|i, s_0) = 0$ when $s_0 = 0$.

Summing over all possible combinations for an edge in B of initial state s_0 (with probability δ_{s_0}) and initial left degree i (with probability λ_i),

$$p_l(x) = \sum_{s_0=1}^{m} \delta_{s_0} \sum_{i=1}^{d_L} \lambda_i \sum_{j=0}^{\min(s_0,i)-1} \binom{i-1}{j} (1-x)^j x^{i-1-j}. \tag{3.3}$$

For a given (i, j) pair in (3.3), the term

$$\binom{i-1}{j}(1-x)^j x^{i-1-j}$$

is shared by all the $s_0 > j$. Defining $\gamma_j = \sum_{s_0=j+1}^{m} \delta_{s_0}$,

$$p_l(x) = \sum_{i=1}^{d_L} \lambda_i \sum_{j=0}^{\min(m,i)-1} \gamma_j \binom{i-1}{j}(1-x)^j x^{i-1-j} \ . \tag{3.4}$$

When the left degree is always d_L where $d_L < m$ (typically $d_L = m-1$), $\sum_{i=1}^{d_L} \lambda_i = \lambda_L = 1$ and $\min(m, i) = d_L$. So

$$p_l(x) = \sum_{j=0}^{d_L-1} \gamma_j \binom{i-1}{j}(1-x)^j x^{i-1-j} \ . \tag{3.5}$$

Computing $p_r(x)$

To calculate $p_r(x)$, we first consider $p_r(x|i)$, where i is the initial right degree of the randomly selected edge. For a specified edge, define the "right neighboring edges" of an edge as the *other* edges connected to its incident right node. An edge can be right-degree-one only when all of its right neighboring edges in the original graph B have been removed because they are incident to a left node with $s = 0$. For each such right neighboring edge, the probability that the left node has $s = 0$ is $1 - p_l(x)$. Thus the probability that all $i-1$ right neighboring edges have left nodes corresponding to a VL decoder that has already successfully decoded is $p_r(x|i) = (1 - p_l(x))^{i-1}$. Summing over all possible initial right degrees, we have

$$p_r(x) = \sum_{i=1}^{d_R} \rho_i (1 - p_l(x))^{i-1} = \rho((1 - p_l(x)) \ . \tag{3.6}$$

Computing $r_1^*(x)$ and $r_1(x)$

Define $r_1^*(x)$ as the probability that a randomly selected edge in B satisfies both the event that defines $p_l(x)$ and the event that defines $p_r(x)$. For a sufficiently large graph, the events defining $p_l(x)$ and $p_r(x)$ are independent. Using the independence of these two events, $r_1^*(x) = p_l(x)p_r(x) = p_l(x)\rho(1 - p_l(x))$.

There exist edges that satisfy the definition of $r_1^*(x)$ but are not available to be used as an increment because they have *already* been used as an increment. To compute the probability $r_1(x)$ that an edge is available as an increment, the probability $r_1^*(x)$ needs to be reduced by the probability that an edge that satisfies the definition of $r_1^*(x)$ is already in the set $\mathcal{Q}_t = \{Q_1, \cdots, Q_t\}$. Note that the edges in $\mathcal{Q}_t$ automatically satisfy the event that defines $p_r(x)$. However, some edges in $\mathcal{Q}_t$ will not satisfy the event that defines $p_l(x)$. The event that an edge is in $\mathcal{Q}_t$, which has probability $1 - x$, is independent of whether the edge satisfies the event that defines $p_l(x)$. Thus, the probability that an edge satisfies the definition of $r_1^*(x)$ and is also in $\mathcal{Q}_t$ is $p_l(x)(1 - x)$.

Thus the final probability $r_1(x)$ that a randomly selected edge in B has not been removed after t iterations and is available as an increment is

$$\begin{aligned} r_1(x) &= r_1^*(x) - p_l(x)(1 - x) && (3.7) \\ &= p_l(x)\left[\rho(1 - p_l(x)) - (1 - x)\right] . && (3.8) \end{aligned}$$

Fig. 3.3 shows an example $r_1(x)$ computed according to the analysis above. Also shown (as red circles) are $\Big(x, r_1(x)\Big)$ points found at each density evolution iteration. Each density evolution iteration (or equivalently GPD iteration) supplies all available increments to the left nodes and then removes all the left nodes that achieve $s = 0$ and their incident edges. At the beginning of the decoding process, $x = 1$ because there is no edge in the set $\mathcal{Q}_t = \{Q_1, \cdots, Q_t\}$. As more edges are removed from the graph, x decreases monotonically towards 0. As x monotonically decreases, $r_1(x)$ initially decreases, but may also increase.

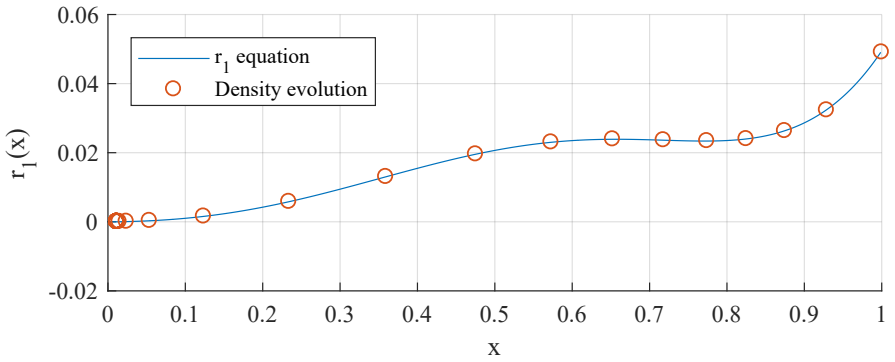


Figure 3.3: Example $r_1(x)$ from density evolution and equation (3.8) for $\lambda(x) = x^3$ and the irregular $\rho_{PEG}(x)$ in Section 3.5.2. The VL code characterization PMF $\boldsymbol{\delta} = [0.30909, 0.464, 0.1939, 0.02934, 0.00318, 0.00049]$.

3.3.3 Computing Probability of Left Node Failure ϵ_{FF}

In this subsection we explore the probability ϵ_{FF} that a left node is not recovered in the feedback-free (FF) system implemented by a GPD operating on the inter-frame code.

When $r_1(x) = 0$, no right-degree-one edges are left, and the peeling process stops. Define $x^{(\epsilon)}$ as the first point, i.e., largest x value less than 1, where $r_1(x) = 0$. We can use $x^{(\epsilon)}$ to predict the probability of recovery failure ϵ_{FF}. For a standard peeling decoder on the erasure channel, if $x^{(\epsilon)} = 0$, then all left nodes are successfully recovered. In contrast, for the GPD any left node with $s_0 = m$ cannot be recovered if the maximum left degree is $d_L = m - 1$. Thus, for the GPD even if $x^{(\epsilon)} = 0$, $\epsilon_{FF} > 0$. Indeed, when $x^{(\epsilon)} = 0$, $\epsilon_{FF} = \delta_m$. This assumes that every left node that receives its s_0 increments is recovered correctly. In Sec. 3.5.2, we will consider undetected errors which lead to a small increase in ϵ_{FF}.

Using $x^{(\epsilon)}$, the calculation of ϵ_{FF} is similar to that of $p_l(x)$. The difference is that for ϵ_{FF} the probability of failure describes the final state of the left *nodes*, so *all* edges incident to a left node are considered instead of all but one.

Consider a left node with initial state s_0 and degree i. If $s_0 > i$, the node can never be recovered. Thus $\epsilon_{FF}(i, s_0) = 1$. If $s_0 \leq i$, the node will fail to decode only if fewer than s_0 edges are removed when$x = x^{(\epsilon)}$, the end of the peeling process. Thus, $\epsilon_{FF}(i, s_0) =$

$\sum_{j=0}^{s_0-1} \binom{i}{j} (1-x^{(\epsilon)})^j x^{(\epsilon)^{i-j}}$. Combining these two cases, we have

$$\epsilon_{FF}(i, s_0) = \sum_{j=0}^{\min(s_0-1,i)} \binom{i}{j} (1-x^{(\epsilon)})^j x^{(\epsilon)^{i-j}} \ . \tag{3.9}$$

Consider all the initial left degrees and initial states s_0, the probability of failure is

$$\epsilon_{FF} = \sum_{s_0=1}^{m} \delta_{s_0} \sum_{i=1}^{d_L} \Lambda_i \sum_{j=0}^{\min(s_0-1,i)} \binom{i}{j} (1-x^{(\epsilon)})^j x^{(\epsilon)^{i-j}} \tag{3.10}$$

where $\Lambda_i = \frac{\lambda_i}{\sum_{j=1}^{d_L} \lambda_i / i}$ is the left *node* degree probability.

3.3.4 Throughput of Feedback and Feedback-Free Systems

The throughput of the feedback-free system is closely linked to the throughput of the associated feedback system that uses a VL code with ACK/NACK feedback. The throughput of the baseline feedback system is

$$R_t^{(FB)} = \frac{k(1-\epsilon_{FB})}{\ell_0 + \beta_{FB}\ell_\Delta}\,, \tag{3.11}$$

where ℓ_0 is the length of the initial transmission, ℓ_Δ is the length of the increments, $\beta_{FB} = \sum_{i=1}^{m-1} i\delta_i + (m-1)\delta_m$ is the average number of increments the feedback system requests from the receiver, k is the number of message symbols, and ϵ_{FB} is the failure probability of the feedback system.

The feedback-free throughput $R_t^{(FF)}$ is computed as

$$R_t^{(FF)} = \frac{k(1-\epsilon_{FF})}{\ell_0 + \beta_{FF}\ell_\Delta}\ , \tag{3.12}$$

where $\beta_{FF} = n_i/n_c$ is the average number of right nodes per left node, and ϵ_{FF} is the

feedback-free failure probability.

3.3.5 Throughput Loss in the Feedback-Free System

The feedback and feedback-free systems share the same VL code design, so both throughput calculations have the same k, ℓ_0, and ℓ_Δ. The feedback-free system is designed to have similar failure performance as the feedback system, so $\epsilon_{FB} \approx \epsilon_{FF}$ and both of these values are small. The main difference between the throughput of the two systems is determined by the difference between β_{FF} and β_{FB}.

This difference between β_{FF} and β_{FB} can be understood by studying the ways in which right nodes fail to provide increments in the feedback-free system. Three mechanisms in the decoding process prevent a linear combination of increments (right node) from providing a useful increment to a VL decoder (left node):

1. The degree of the right node of interest (RNOI) never decreases below two.
2. The degree of the RNOI decreases from two-or-more to zero in a single GPD iteration.
3. The degree of the RNOI achieves the value of one during a GPD iteration, but other increments supplied in the same GPD iteration achieve $s = 0$ for left node that this RNOI could assist.

The first mechanism prevents the RNOI from providing an increment to any incident left node. Both the second and the third mechanisms are scenarios where the RNOI is unable to help because it is superfluous. Define probabilities η_1, η_2 and η_3 as the probabilities that a randomly selected right node satisfies each of the mechanisms listed above respectively. The probability that a right node does not provide a useful increment to a left node is $\eta = \eta_1 + \eta_2 + \eta_3$.

Recall that $\beta_{FF} = n_i/n_c$ is the average number of right nodes (combined increments) per left node (VL decoder). Consider that β_{FF} is the number of right nodes divided by the number of left nodes in the original graph which has E edges. The number of right nodes

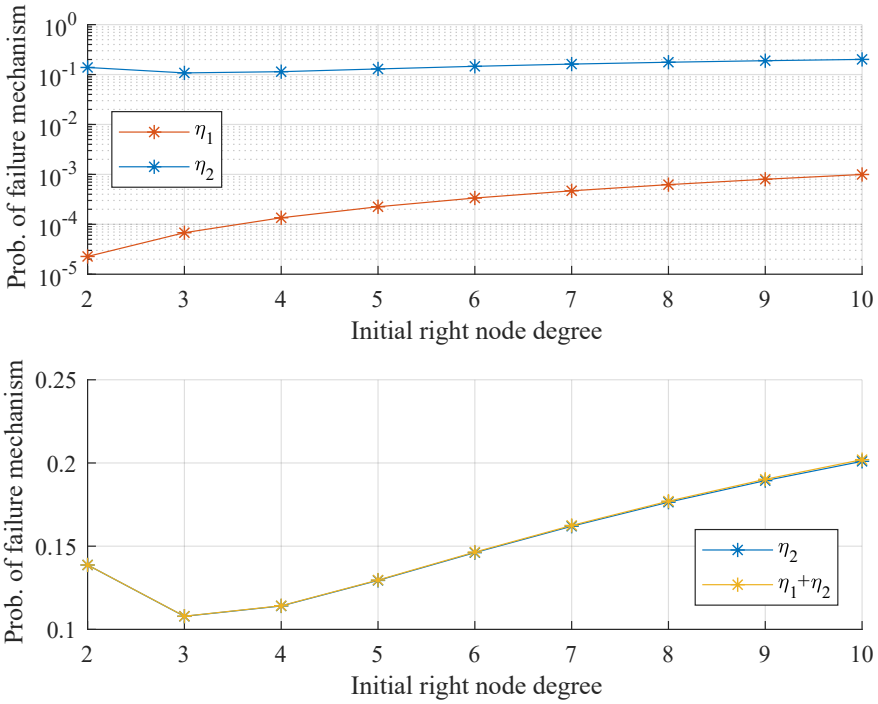


Figure 3.4: Probability of failure mechanisms of right nodes for the irregular $\rho_{PEG}(x)$ in Section 3.5.2.

is E/a_R, where a_R is the average right node degree. The degree of each left node is exactly $d_L = m - 1$ to guarantee that all VL decoders have the possibility to receive the maximum number of increments. Thus the number of left nodes is E/d_L. As a result, $\beta_{FF} = d_L/a_R$.

For any left and right degree distributions, the inter-frame code must provide the left nodes with at least the number of increments they need in a feedback system. Thus $\beta_{FF} \geq \beta_{FB}$ or $d_L/a_R \geq \beta_{FB}$ which implies an upper bound on the average right degree: $a_R \leq d_L/\beta_{FB}$. Considering that the number of *useful* right nodes is $E(1-\eta)/a_R$, a_R can be approximated using d_L, β_{FB} and the probability η as follows:

$$a_R \approx \frac{d_L(1-\eta)}{\beta_{FB}}. \tag{3.13}$$

Fig. 3.4 shows the fraction of right nodes of each possible degree that suffer from the first two failure mechanisms for the irregular $\rho_{PEG}(x)$ in Section 3.5.2 according to density

evolution analysis. The plot shows that η_2 is two or more orders of magnitude smaller than η_1 for this degree distribution. This same behavior was seen for a different irregular $\rho_{PEG}(x)$, designed for a different VL code, in Sec. 3.5.1. [6] The probability of the third mechanism is difficult to quantify, but should be extremely small for codes with good throughput.

As shown in Fig. 3.4, both high and low degree right degrees are undesirable because their degrees are more likely to decrease directly from two or more to zero in a single iteration. This observation leads to a simplified design method in Sec. 3.4.2, where the right degree distribution is quasi-regular.

3.4 Design Methods

The design of the inter-frame system involves three steps. The first step is to design a VL code. The second step is to design a degree distributions $\rho(x)$ and $\lambda(x)$ for the inter-frame code. The third step is to use the degree distribution to construct the bipartite graph for the inter-frame code.

3.4.1 VL Code Design

The VL code design process contains two parts. The first part is to specify a VL code and its feedback mechanism. The second part is to determine the parameters of the VL code: the maximum number of increments $m-1$, the initial transmission length ℓ_0, and the increment size ℓ_Δ.

For the inter-frame system to have low complexity and high throughput, the ideal VL code should have short average block and a throughput approaching capacity when analyzed in a feedback system. We will introduce two types of VL codes in the following discussion. The first is a 1024-state tail-biting convolutional code (TBCC) using ACK/NACK feedback from reliability output Viterbi algorithm (ROVA) [60]. The second is a $(N = 32, K = 24)$ rate 0.75 NB-LDPC code on Galois field $GF(256)$ described in [70] with binary IR.

Since at the receiver the shared IR connects to the VL decoders directly through a bipartite graph, VL codes with ACK/NACK-based feedback mechanism can be directly applied in the inter-frame system. If the feedback mechanism of the VL code of choice requires more informative receiver-to-transmitter feedback, adaption is required. The discussion below provides an example where a full noiseless feedback scheme for NB-LDPC is converted to an ACK/NACK scheme.

Regardless of the type of VL code used, the design process for m, ℓ_0 and ℓ_Δ remains the same. The emphasis of this chapter is on an architecture that allows short-blocklength VL codewords decoded in parallel without feedback (with a GPD connecting the parallel codewords) to approach the performance of any individual VL code implemented with ACK/NACK feedback. As we show below, the proposed system closely approaches the individual VL code's performance with feedback. As even better short-blocklength VL codes with feedback are identified, the performance of the proposed architecture will improve correspondingly.

A practical VL code has an initial rate (its highest rate) and a series of lower rates obtained by transmitting successive increments of additional redundancy. The lowest rate must guarantee a VL decoding failure probability below the desired target probability of decoding failure in a feedback system. In this chapter, the target probability of failure for the feedback-free system is $\epsilon_{FF} = 10^{-3}$. As a result, the target failure probability at the lowest rate when designing the VL codes should be upper-bounded by 10^{-3}. This lowest rate translates to a maximum allowed blocklength.

Given the maximum allowed blocklength, there is a trade-off between throughput and complexity. This trade-off involves the maximum number of increments and the transmission lengths (for the initial transmission and the increments). Increasing the maximum number of increments increases complexity. However, it also improves throughput by decreasing the individual transmission lengths, which reduces the number of unneeded symbols of incremental redundancy that are transmitted.

In the context of the feedback-free system using a GPD, a large maximum number of increments induces high left and right node degrees in the bipartite graph. In this chapter, we limit the maximum number of increments to 4 ($m = 5$) in part to control complexity. As we will see, this small value of m still yields good performance, i.e., an $R_t^{(FF)}$ close to $R_t^{(FB)}$.

With the maximum number of increments fixed, the increment sizes and the length of the initial transmission ℓ_0 need to be optimized. In our system design, all the increments have the same length ℓ_Δ, so there are only two parameters, ℓ_0 and ℓ_Δ, that need to be determined. This optimization is conducted in two steps. First, VL codewords are simulated on the target channel (2dB BI-AWGN in this chapter) with ACK/NACK feedback and bit-by-bit IR. The decoding status at each instantaneous blocklength for each codeword is recorded. Second, the decoding status data is used to calculate the empirical probability of decoding for the first time at each possible transmission length. Using this characterization, ℓ_0 and ℓ_Δ are optimized to maximize the throughput $R_t^{(FB)}$ of the feedback system.

Convolutional VL Code

Convolutional codes are widely used in feedback communication. We choose the $k = 64$ 1024-state TBCC using ACK/NACK feedback from ROVA in [60] as the first VL code design for the inter-frame system. This design uses a rate 1/3 mother code to encode 64 bits of information into a 192-bit codeword. The IR bits are generated by pseudo-random puncturing.

Table 3.1 shows a comparison between the best variable-size increment (VI) and constant-size increment (CI) system designs for the 1024-state TBCC of [60] with $k = 64$ information bits and $m = 5$. From Table 3.1, the best CI design achieves 99% of the $R_t^{(FB)}$ of the VI designs, indicating that the CI requirement does not significantly affect performance. To accommodate for the failure mechanisms discussed in Sec. 3.3.5, a more stringent target $\epsilon_{FB} = 5 \times 10^{-4}$ is needed. As a result, the increment size is increased by one to 16 for the CI system used in our simulations in Sec. 3.5.1, identified as "Actual" in Table 3.1.

Table 3.1: Performance comparison between the $k = 64$ 1024-state TBCC VL code's VI and CI designs. (2 dB BI-AWGN, $m = 5$, target $\epsilon_{FB} = 10^{-3}$ for the first two rows, target $\epsilon_{FB} = 5 \times 10^{-4}$ for the third row)

		Transmission Lengths $\ell_0, \ldots, \ell_4$ (bits)	Rate $R_t^{(FB)}$	Failure Rate $\delta(5)$
VI	ES	107, 9, 10, 13, 29	0.528756	9.4×10^{-4}
CI	Best	107, 15, 15, 15, 15	0.522791	9.3×10^{-4}
	Actual	108, 16, 16, 16, 16	0.520843	5×10^{-4}

Note: ES presents exhaustive search.

This CI design still achieves 98.5% of the $R_t^{(FB)}$ of the VI designs, and 81.1% of capacity on the 2dB BI-AWGN channel, with an average blocklength of 122 bits. The corresponding $\boldsymbol{\delta}$ = [0.33304, 0.44860, 0.18225, 0.03159, 0.00402, 0.00050]. The increment sizes of 15 or 16 considered here to combat dispersion are significantly smaller than those in [62] (on the order of 550 symbols) used to combat fading.

NB-LDPC VL Code

NB-LDPC VL codes have been demonstrated to achieve high throughput with short average blocklength [70, 61]. Our second VL code design is based on the $(N = 32, K = 24)$ rate 0.75 NB-LDPC code on Galois field $GF(256)$ in [70]. The base code encodes 24 8-bit symbols into a 32-byte codeword. As shown in Table 3.3, this VL code with the actual CI lengths (third row) used in simulations in Sec. 3.5.2 achieves $R_t^{(FB)} = 0.570534$ bits per transmission using $m = 5$ incremental transmissions, which is 88.9% of capacity on the 2dB BI-AWGN channel with an average blocklength of 336 bits.

Below we describe how binary feedback mechanism is designed for the NB-LDPC VL code and how ℓ_0 and ℓ_Δ are selected.

1) Design the feedback scheme for the NB-LDPC VL code:

This chapter uses the approach of [70, 61] to provide bit-by-bit incremental redundancy to the NB-LDPC code. This VL scheme differs from the traditional IR scheme where a

low-rate mother code is designed and incremental bits are *punctured* from the low-rate code to form higher rate codes. The NB-LDPC VL code design starts with a *high-rate* code, and *generates* incremental bits from the symbols of the base codeword. These incremental bits are not symbols in any non-binary code; they simply provide additional reliability about previously transmitted symbols.

On the BI-AWGN channel, each non-binary code symbol is converted to a binary representation $[g_1, g_2, \cdots, g_p]$ of length p, where $p = 8$ in our design, and transmitted through the channel using binary phase shift keying (BPSK). We use the primitive polynomial $F(x) = x^8 + x^4 + x^3 + x^2 + 1$ for the conversion. The decoder receives each symbol as a noise-distorted sequence $[y_1, y_2, \cdots, y_p]$. To characterize the reliability of each symbol, the decoder first calculates the log-likelihood ratio (LLR) of each received bit y_i,

$$LLR_{y_i} = \log \frac{P(g_i = 1|y_i)}{P(g_i = 0|y_i)}, \quad i = 1, 2, \cdots, p \ , \tag{3.14}$$

where $P(g_i = 1|y_i)$ is the posterior probability that $g_i = 1$ given that the observed symbol at the receiver is y_i. Since each symbol has 2^p possible values, its reliability is described by a length-$(2^p - 1)$ vector $\boldsymbol{LLR}^{(\text{sym})}$ of LLRs. Each entry in the vector is the LLR between a candidate non-zero Galois value and 0. For example, the first entry of vector is the LLR of "1",

$$LLR_1^{(\text{sym})} = \log \frac{P(g_1, \cdots, g_p = 00 \cdots 1|y_1, \cdots, y_p)}{P(g_1, \cdots, g_p = 00 \cdots 0|y_1, \cdots, y_p)} \, . \tag{3.15}$$

Consider the XOR of certain bits of a symbol. For example, $g_1 \oplus g_5 \oplus g_p$. The log-likelihood ratio of this combination of a received symbol can be calculated as

$$LLR_{g_1 \oplus g_5 \oplus g_p} = \log \frac{P(g_1 \oplus g_5 \oplus g_p = 1|y_1, y_5, y_p)}{P(g_1 \oplus g_5 \oplus g_p = 0|y_1, y_5, y_p)} \ . \tag{3.16}$$

For the Galois field $GF(2^p)$, there are $2^p - 1$ possible combinations for each symbol, including the singletons $g_1, g_2, \cdots, g_p$. Considering an l-bit combination, there are 2^{l-1} l-bit patterns

for which the XOR produces a zero and 2^{l-1} l-bit patterns for which the XOR produces a one. For example, when $g_1 \oplus g_5 \oplus g_p = 1$, $[g_1, g_5, g_p]$ can be any pattern in the set $\{001, 010, 100, 111\}$.

Now we consider the LLR associated with a combination, e.g. $g_1 \oplus g_5 \oplus g_p$, based on the observation of the original transmitted symbol at the receiver. Let $\boldsymbol{y}$ represent $[y_1, y_5, y_p]$, and $\boldsymbol{g}$ represent $[g_1, g_5, g_p]$,

$$LLR_{g_1 \oplus g_5 \oplus g_p} = \log \frac{P(\boldsymbol{g} \in \{001, 010, 100, 111\}|\boldsymbol{y})}{P(\boldsymbol{g} \in \{000, 011, 101, 110\}|\boldsymbol{y})} \tag{3.17}$$

$$= \log \left(\frac{P(001|\boldsymbol{y}) + P(010|\boldsymbol{y}) + P(100|\boldsymbol{y}) + P(111|\boldsymbol{y})}{P(011|\boldsymbol{y}) + P(110|\boldsymbol{y}) + P(101|\boldsymbol{y}) + P(000|\boldsymbol{y})} \right). \tag{3.18}$$

For each component probability

$$P(g_1, g_5, g_p|y_1, y_5, y_p) = P(g_1|y_1)P(g_5|y_5)P(g_p|y_p).$$

Initially, the NB-LDPC decoder will try to decode without any IR. If the decoding fails, then IR is provided using XOR combinations as described above.

1a) Binary IR with full noiseless feedback: Consider the case where the transmitter has the luxury of full noiseless feedback and provides IR one bit at a time. If decoding fails, the receiver computes the LLRs of all the $(2^p - 1) \times N$ possible combinations, i.e., all combinations for each received symbol. Then the receiver identifies the least reliable combination (as well as the corresponding symbol) and requests the transmission of that XOR combination for that symbol. The transmitter then sends the XOR of the designated bits as a single bit $g*$ of IR.

The receiver uses the received IR bit $y*$ to augment the LLR vector of the identified symbol. Let $\boldsymbol{x}$ represent a possible p-bit pattern for the symbol associated with the XOR combination. Let $\mathcal{X}_0$ be the set of such bit patterns for which the designated combination of bits produces a zero and $\mathcal{X}_1$ be the set of such bit patterns for which the designated combination of bits produces a one. For $\boldsymbol{x} \in \mathcal{X}_0$, the LLR is not affected by the IR bit $g*$

because the value of the XOR is the same for the numerator and denominator of the LLR. For any $\boldsymbol{x} \in \mathcal{X}_1$, the LLR of $\boldsymbol{x}$ is augmented using the LLR of the IR bit $g*$ as follows:

$$LLR^{(\text{sym})}_{\boldsymbol{x},\text{new}} = LLR^{(\text{sym})}_{\boldsymbol{x},\text{old}} + LLR_{g*} \quad . \tag{3.19}$$

The NB-LDPC decoder tries to decode again with the updated symbol LLR vector $\boldsymbol{LLR}^{(\text{sym})}_{\text{new}}$. If the decoding fails, e.g., according to a syndrome failure, cyclic redundancy check code (CRC) failure, or a genie, the receiver needs additional IR. To identify the new IR bit, the receiver updates the LLRs of combinations based on the updated LLR vector. To update the LLR of a combination, the posterior probability of a specific binary pattern of a combination is calculated from the posterior probabilities of candidates of the symbol. For the example IR combination $g* = g_1 \oplus g_5 \oplus g_p$, let $\boldsymbol{y}$ represent the original observations and the IR bit: $\boldsymbol{y} = [y_1, y_2, \cdots, y_p, y*]$,

$$P(g_1 g_5 g_p = 101 | y_1 y_5 y_p y*) = \sum_{\boldsymbol{x}: g_1 g_5 g_p = 101} P(\boldsymbol{x}|\boldsymbol{y}) \tag{3.20}$$

The probability $P(\boldsymbol{x}|\boldsymbol{y})$ above can be derived directly from the LLR vector of the symbol.

After augmenting the LLRs of the combinations of the previously identified symbol. The receiver requests the newly identified least reliable combination. This process repeats until the decoding succeeds.

1b) Binary IR for ℓ_Δ increments for inter-frame coding: The process discussed above requires full feedback from the decoder identifying the specific bit of IR to be sent. We can modify the process to remove the specific bit identification, and effectively construct a VL NB-LDPC code with ACK/NACK feedback. The intuition is for the transmitter and the receiver to use an ordered list of the symbols and combinations that need additional redundancy based on the expectation of bit reliabilities.

To design the sequence of IR transmissions, the order of symbols and the bit combinations for each symbol need to be determined as follows:

Table 3.2: The first 5 combinations for the first symbol of the rate 0.75 NB-LDPC code.

Priority	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8
1	●	●	●	●	●	●	●	●
2		●		●		●	●	
3	●	●	●		●			
4		●	●	●			●	
5		●			●	●	●	

Note: Dots represent the corresponding bit is in the combination.

Table 3.3: Performance comparison between the $K = 24$ NB-LDPC VL code with ACK/NACK feedback's VI and CI designs. (2dB BI-AWGN, $m = 5$, target $\epsilon_{FB} = 10^{-3}$ for the first two rows, target $\epsilon_{FB} = 5 \times 10^{-4}$ for the third row)

		Transmission Lengths $\ell_0, \ldots, \ell_4$ (bits)	Rate $R_t^{(FB)}$	Failure Rate $\delta(5)$
VI	ES	296, 24, 22, 28, 65	0.578042	9.9×10^{-4}
CI	Best	297, 34, 34, 34, 34	0.572789	10^{-3}
	Actual	302, 36, 36, 36, 36	0.570534	4.9×10^{-4}

Note: ES presents exhaustive search.

1. Since the BI-AWGN channel is memoryless and there is no preference for symbol ordering *a priori*, one bit of IR is sent for each symbol using the natural order of the symbols. This same ordering is used for subsequent bits of IR.

2. The combination of bits for each IR bit for each symbol is determined through simulation by finding the least reliable combination given the received symbols and previously received IR bits.

Table 3.2 shows the first 5 combinations of the first symbol generated for the rate 0.75 NB-LDPC code. An interesting observation is that the combination of all bits is always the least reliable for all the symbols. Combinations of 4 to 5 bits usually follow.

2) Design ℓ_0 and ℓ_Δ of the NB-LDPC VL code

Table 3.3 compares the performance of the designed constant-increment (CI) VL code with a design using variable-increment (VI) sizes, which can be considered a performance upper bound. The transmission lengths in the second row optimize the throughput of the

VL code while achieving $\epsilon_{FB} = 10^{-3}$. A failure occurs only when the VL decoder does not produce the correct codeword, as determined by a genie.

The actual system will have a higher failure rate because of the failure mechanisms discussed in Section 3.3.5 as well as undetected errors and the resulting error propagation that will occur in a real system where decoding failures are detected by CRC. We designed the VL code in the third row to achieve a more stringent requirement of $\epsilon_{FB} = 5 \times 10^{-4}$ in the genie-aided scenario so that we can achieve $\epsilon_{FF} = 10^{-3}$ in the final implementation. For the third row of Table 3.3, $\boldsymbol{\delta}$ = [0.30909, 0.464, 0.1939, 0.02934, 0.00318, 0.00049] on the 2dB BI-AWGN channel.

3.4.2 Design the Inter-Frame Degree Distribution

We studied two design approaches for the inter-frame degree distribution: using the differential evolution algorithm to design the degree distribution directly, and using the quasi-regular degree distribution formation with optimization by enumeration.

Differential Evolution Algorithm

Differential evolution is an iterative approach to designing degree distributions for linear block codes described by low-density incidence matrices (either LDPC or LDGM)[71, 72]. In our designs, the left node degree d_L, the maximum right node degree d_R, and the target probability of failure are predefined as the constraints of the optimization. The average number of increments per left node β is gradually decreased, so that throughput of the design is increased, until the designed degree distribution's probability of failure exceeds the target value.

There are d_L+d_R variables from the right and left degree distribution, d_L+d_R-3 of which need to be optimized. Define the parameter vector to be $\boldsymbol{p} = \{\lambda_2, \ldots, \lambda_{d_L-1}, \rho_2, \ldots, \rho_{d_R}\}$. Each iteration has a generation number G. Consider vector $\boldsymbol{p}$ as a point in the feasibility space with dimension $d_L + d_R - 3$.

For $G = 0$, an initial population of N_P points is generated using the hit-and-run sampler [73]. Each point in the generation has an associated $x^{(\epsilon)}$ that can be found from $r_1(x)$. The corresponding probability of failure ϵ_{FF} can be predicted using (3.10), or through density evolution. The following steps are applied on the generation G population to form generation $G + 1$:

Step 1: Mutation. Intermediate points $\boldsymbol{v}_{i,G+1}$ for $i \in \{0, \cdots, N_P - 1\}$ are produced using

$$\boldsymbol{v}_{i,G+1} = \boldsymbol{p}_{\text{best},G} + 0.5 \cdot (\boldsymbol{p}_{r_1,G} - \boldsymbol{p}_{r_2,G} + \boldsymbol{p}_{r_3,G} - \boldsymbol{p}_{r_4,G}), \tag{3.21}$$

where the point $\boldsymbol{p}_{\text{best},G}$ has the lowest ϵ_{FF} in generation G, and r_1, r_2, r_3 and r_4 are randomly chosen from $\{0, \cdots, N_P - 1\}$ without repetition. If $\boldsymbol{v}_{i,G+1}$ is out of the feasibility space, this step is repeated until a feasible point is produced.

Step 2: Recombination. Candidate generation $G + 1$ points $\boldsymbol{u}_{i,G+1}$ are generated from $\boldsymbol{v}_{i,G+1}$ using the following procedure for each element $j = 0, 1, \cdots, d_L + d_R - 4$:

$$u_{j,i,G+1} = \begin{cases} v_{j,i,G+1} & \text{if } U_j \leq P_{\text{CR}} \text{ or } j = K(i), \\ p_{j,i,G} & \text{otherwise.} \end{cases} \tag{3.22}$$

where U_j for each j is drawn uniformly from the unit interval $[0, 1]$. $K(i)$ for each i is randomly chosen from $\{0, \cdots, N_P - 1\}$, and P_{CR} is the crossover probability. In this chapter $P_{\text{CR}} = 1$ so that $u_{j,i,G+1} = v_{j,i,G+1}$ for all j.

Step 3: Selection. In this step, the actual points of generation $G+1$ are chosen. For each index i, either $\boldsymbol{u}_{i,G+1}$ or $\boldsymbol{p}_{i,G}$ is selected, depending on which point has the lower probability of failure ϵ_{FF}. During these iterations it may be that neither $\boldsymbol{u}_{i,G+1}$ nor $\boldsymbol{p}_{i,G}$ has a probability of failure ϵ_{FF} below the target. However, if both points' probabilities of failure are below the target, the point requiring the lower number of iterations is selected.

After G reaches a specified value, we consider the point with the lowest probability of failure. If this probability is below the target failure rate ϵ, this point is saved as a possible

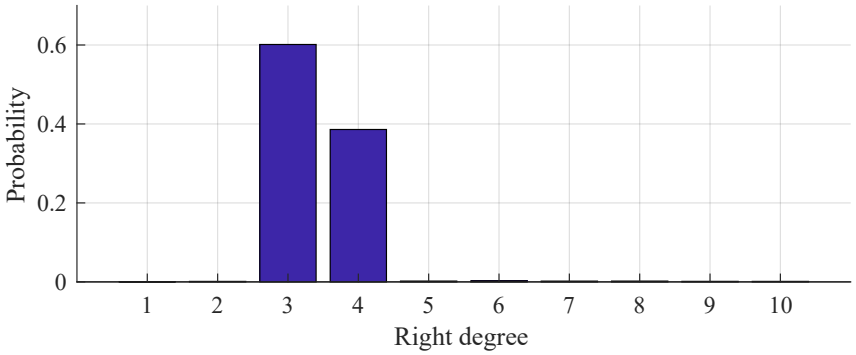


Figure 3.5: Example right degree distribution $\rho_{DE}(x)$ from differential evolution as described in Sec. 3.5.2.

final solution, and the process is restarted with a slightly decreased β. If this probability is higher than the target failure rate ϵ, then the process is terminated and the previous solution is considered the final answer. In this chapter, the maximum left node degree is $d_L = 4$. The maximum right node degree is $d_R = 10$. We choose the population size $N_P = 500$ and the maximum number of generations to be 50.

In our experiments the degree distributions generated with differential evolution had two characteristics:

- The left degree distributions have $\lambda_{d_L} \approx 1$. Only degree-d_L left nodes are allowed in practice.
- Two adjacent degrees of right nodes have all but an insignificant amount of the probability.

Fig. 3.5 illustrates the second characteristic, depicting the right degree distribution obtained by one of our differential evolution designs. The first characteristic can be easily understood as each left node requires the capability to receive the maximum number of increments from the right nodes. The second characteristic closely relates to the discussion in Section 3.3.5, where both high and low degree right nodes suffer from an increased probability that they cannot be used by a left node to decode. From these observations, we proposes the quasi-regular degree distribution formation.

Quasi-regular Design

The quasi-regular distribution formation is defined as follows:

- All the left nodes' degrees are fixed at $d_L = m - 1$,
- The right degree distribution is quasi-regular, with nonzero probability restricted to two or three adjacent integers determined by the range of β to be explored.

Any optimization algorithm can be employed to determine the optimum values in the degree distribution. Because there are only two or three parameters to be optimized, we use exhaustive search to find the results.

The quasi-regular approach is significantly less complex than differential evolution while producing the same or slightly better performance in our experiments. While differential evolution with an unlimited number of iterations and more degrees of freedom should always produce performance at least as good as a more constrained exhaustive search, our simulations were limited to 50 iterations of differential evolution, and all the results were observed to approximate the quasi-regular distribution.

3.4.3 Bipartite Graph Design

We use the progressive edge growth (PEG) algorithm of Hu et al. [74] to construct the inter-frame code bipartite graph from the degree distribution to maximize girth. Girth is important for the GPD; if a set of left nodes has each left node connecting to the same set of right nodes, then the peeling process can reach a bottleneck. In other words, the neighborhood of any right node should grow quickly with path length to enable the GDP to avoid bottlenecks, i.e., maintain a large value of $r_1(x)$ for as long as possible.

The bipartite graph generated by the PEG algorithm is slightly different from the asymptotic graph described by the degree distributions. The difference is more pronounced when the number of left nodes is limited. Results in Sec. 3.5 show that the difference does not significantly degrade the performance of the designs.

Because left degree distributions designed in this chapter are regular, the role of left and right nodes as described in [74] are swapped in our PEG implementation.

3.5 Results

In this section, we present the details of the feedback-free systems we designed and the corresponding simulation results. We focus on the two factors that determine the performance of the system: the throughput $R_t^{(FF)}$ and the probability of failure of the codewords ϵ_{FF}. The design and simulations are conducted on a 2dB BI-AWGN channel which has a capacity of 0.642149 bits per transmission. With the channel condition fixed, we seek an $R_t^{(FF)}$ of the designed feedback-free system that approaches the $R_t^{(FB)}$ of the reference VL code with feedback, while also guaranteeing the probability of failure $\epsilon_{FF} \leq 10^{-3}$.

3.5.1 Convolutional VL Code Based System

$R_t^{(FF)}$ for the inter-frame design is upper-bounded by $R_t^{(FB)}$ of the VL codes that form the foundation of the system. As is shown in Table 3.1, the convolutional VL code with noiseless ACK/NACK feedback and variable-increment sizes, achieves a throughput of $R_t^{(FB)} = 0.528756$ bits per transmission. We will compare our feedback-free designs against this VL code with feedback. From Table 3.1, the convolutional VL code we use to construct the feedback-free system has initial transmission length $\ell_0 = 108$ bits and increment size $\ell_\Delta = 16$ bits.

Design using regular bipartite graph

As a reference, we begin with a regular inter-frame bipartite graph design, without using differential evolution or quasi-regular degree distributions. The intended bipartite graph has only degree-4 left nodes and degree-3 right nodes. We use PEG to generate the actual bipartite graph with 100000 left nodes. The resulting graph has a left degree distribution

Table 3.4: Performance of the convolutional VL code based feedback-free systems.

	$R_t^{(FF)}$	ϵ_{FF}	% Capacity	% $R_t^{(FB)}$
Regular	0.494503	6.69×10^{-4}	77.0%	93.5%
Irregular	0.510182	8.98×10^{-4}	79.4%	96.5%

Note: "% Capacity" is the percentage $R_t^{(FF)}$ is of the channel capacity. "% $R_t^{(FB)}$" is the percentage $R_t^{(FF)}$ is of $R_t^{(FB)}$ of the best feedback system (the first row in Table 3.1).

$\lambda_{PEG}(x) = x^3$ and a right degree distribution $\boldsymbol{\rho}_{PEG}^{(\text{coeff})} = \{0, 0.0001, 0.9999\}$. The ratio $\beta_{FF} \approx 1.33334$ The rate of the overall feedback-free system $R_t^{(FF)} = 0.494503$ bits per transmission. The simulated failure probability is $\epsilon_{FF} = 6.691 \times 10^{-4}$.

Design using differential evolution

Using differential evolution, our highest-rate irregular bipartite graph' left degree distribution is $\lambda_{DE}(x) = x^3$, and right degree distribution is $\boldsymbol{\rho}_{DE}^{(\text{coeff})} = \{2.73614 \times 10^{-3}, 2.36956 \times 10^{-2}, 0.445366, 0.213356,\ 0.188564, 2.68009 \times 10^{-2}, 4.89986 \times 10^{-2}, 1.21422 \times 10^{-3}, 1.77664 \times 10^{-2}, 3.15029 \times 10^{-2}\}$.

The actual bipartite graph with 100000 left nodes has $\lambda_{PEG}(x) = x^3$ and $\boldsymbol{\rho}_{PEG}^{(\text{coeff})} = \{2.6575\times10^{-3}, 2.4735\times10^{-2}$,$0.4420125, 0.21653, 0.1872125, 2.8155\times10^{-2}, 4.84225\times10^{-2}, 1.46\times 10^{-3}, 1.9215 \times 10^{-2}, 2.96 \times 10^{-2}\}$. The corresponding $\beta_{FF} = 1.08330$. The rate of the feedback-free system $R_t^{(FF)} = 0.510182$ bits per transmission. The failure probability $\epsilon_{FF} = 8.979 \times 10^{-4}$.

Table 3.4 compares the results with the best feedback system under the same constraints. The irregular feedback-free system achieves more than 95% of the best $R_t^{(FB)}$ for $m = 5$. We also calculated the exact density evolution for both the regular and irregular code. Fig. 3.6 compares the density evolution prediction of the probability of failure at each iteration with the simulation result. The regular code's result is very closely approximated by the density evolution. The simulation result of the irregular code is also close to density evolution, and the prediction of the asymptotic probability of failure matches with the simulation result in

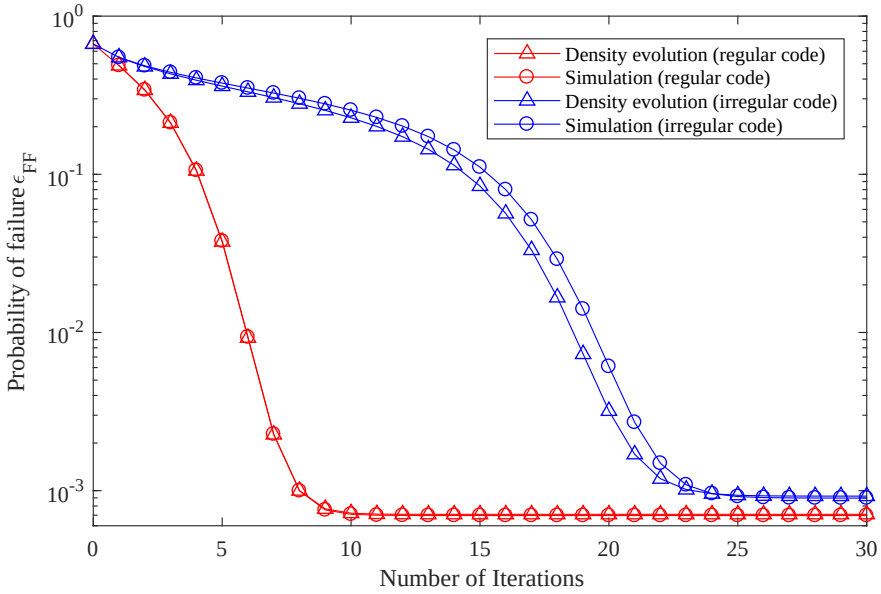


Figure 3.6: Probability of failure versus the number of iterations of the convolutional VL Code based feedback-free system.

both cases.

Design using quasi-regular degree distributions

The simulation results of the regular and irregular systems above shows that the performance of feedback-free systems with large enough bipartite graphs closely follows the asymptotic prediction from density evolution. In this section, we will use density evolution and the analysis in Sec. 3.3 to explore the highest possible throughput using quasi-regular degree distribution, assuming infinite large bipartite graphs. We will show later in the chapter that limiting the size of the bipartite graph will not significantly degrade the throughput and failure probability performance of the feedback-free system.

We first used a quasi-regular right degree distribution of $\rho_{QR}(x) = \alpha x^2 + (1-\alpha)x^3$ (degree 3 and 4) to explore β_{FF} values in between these two points, where α is a design parameter. For each row, we fix the maximum number of iterations, and search for the smallest value of α that achieves a probability of failure that is less than $\epsilon_{FF} = 1 \times 10^{-3}$ according to density evolution. As shown in Table 3.5 and Figs. 3.7 and 3.8 these distributions did quite well.

Table 3.5: Density evolution performance characteristics of quasi-regular $\rho_{QR}(x) = \alpha x^2 + (1-\alpha)x^3$ for the convolutional VL code based feedback-free system. $\lambda(x) = x^3$ in all cases.

α	a_R	β_{FF}	$R_t^{(FF)}$	No. iter.	% $R_t^{(FB)}$	ϵ_{FF}
1	3	1.33333	0.494495	15	93.5%	7.09×10^{-4}
0.531	3.39847	1.17700	0.504210	20	95.4%	7.82×10^{-4}
0.244	3.69914	1.08133	0.510342	30	96.5%	8.35×10^{-4}
0.168	3.78788	1.05600	0.511991	40	96.8%	8.50×10^{-4}
0.139	3.82287	1.04633	0.512623	50	96.9%	8.56×10^{-4}
0.108	3.86100	1.03600	0.513299	100	97.1%	8.63×10^{-4}

Note: "No. iter." is the number of iterations for the GPD to achieve the listed ϵ_{FF}. "% $R_t^{(FB)}$" is the percentage $R_t^{(FF)}$ is of $R_t^{(FB)}$ of the best feedback system (the first row in Table 3.1).

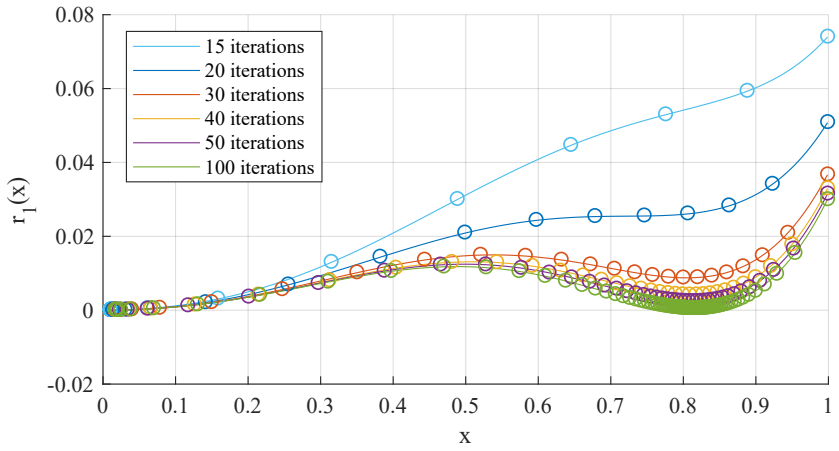


Figure 3.7: $r_1(x)$ versus x for Table 3.5. The curves are generated using equation (3.8). Circles indicate iteration points determined through density evolution.

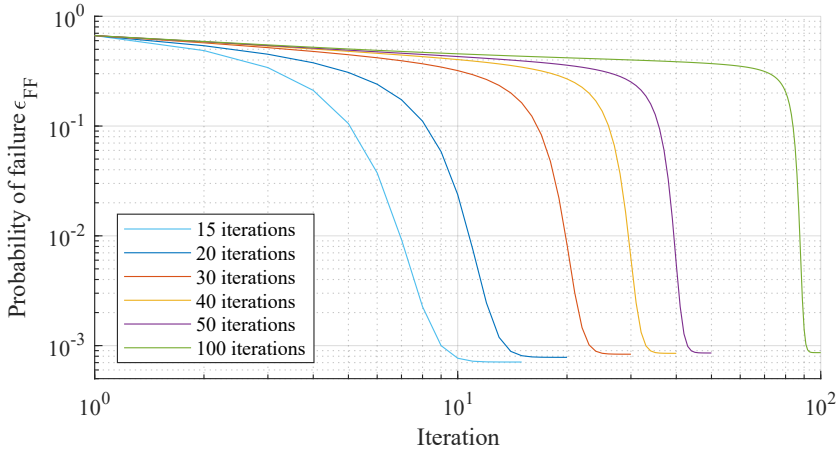


Figure 3.8: Probability of failure versus iterations for $\rho_{QR}(x)$ distributions in Table 3.5. The curves are generated using density evolution.

Table 3.6: Density evolution performance characteristics of the highest $R_t^{(FF)}$ (lowest-β_{FF}) quasi-regular right degree distributions for the convolutional VL code based feedback-free system. $\lambda(x) = x^3$ in all cases.

$\rho_{QR}(x)$	a_R	β_{FF}	$R_t^{(FF)}$	iter.	% $R_t^{(FB)}$	ϵ_{FF}
$0.0998x^2 + 0.9002x^3$	3.87122	1.03327	0.513478	1648	97.1%	8.65×10^{-4}
$0.409x^2 + 0.052x^3 + 0.539x^4$	3.88903	1.02853	0.513765	3748	97.2%	9.16×10^{-4}

Note: "iter." is the number of iterations for the GPD to achieve the listed ϵ_{FF}. "% $R_t^{(FB)}$" is the percentage $R_t^{(FF)}$ is of $R_t^{(FB)}$ of the best feedback system (the first row in Table 3.1).

For example, the $\alpha = 0.244$ distribution outperformed both the irregular distribution from differential evolution. If more iterations are possible, over 97% of $R_t^{(FB)}$ can be achieved by $\alpha = 0.108$ with 100 iterations.

We explored even more iterations and three consecutive degrees, as shown in Table 3.6, but even with thousands of iterations, negligible improvement over the 97.1% of $R_t^{(FB)}$ achieved by $\alpha = 0.108$ was seen. In Table 3.5 and the second row of Table 3.6, the precision of degree distribution coefficients is 0.001. And in the first row of Table 3.6, the precision is 0.0001.

To summarize, our feedback-free system based on convolution VL code can achieve 97.2% of the $R_t^{(FB)}$ of the reference convolution VL code with feedback, and 80% of the 2dB BI-AWGN channel capacity, while guaranteeing a failure probability below 10^{-3} with 3748 GPD iterations. More practical designs with reasonable numbers of GPD iterations require a small sacrifice of the throughput.

3.5.2 NB-LDPC VL Code based System

As is shown in Table 3.3, our best NB-LDPC VL code with variable increment sizes and noiseless ACK/NACK feedback, achieves a throughput of $R_t^{(FB)} = 0.578042$ bits per transmission. This is the upper bound of the throughput achievable with the NB-LDPC VL code based feedback-free system. From Table 3.3, the NB-LDPC VL code in the following designs has the initial transmission length $\ell_0 = 302$ bits and the increment size $\ell_\Delta = 36$ bits. We will

explore limiting the number of left nodes in the inter-frame bipartite graph in the following designs.

Performance with a genie

In this subsection, we compare the simulated performance of the inter-frame degree distribution designs using differential evolution and quasi-regular formation, assuming the decoding process only succeeds when the correct codeword is produced, as determined by a genie. In the next subsection we will replace the genie with a cyclic redundancy check (CRC) code for error detection to study the loss due to imperfect error detection.

1) Design using differential evolution:

The design for the inter-frame bipartite graph utilizes differential evolution, which yielded $\lambda(x) = x^3$ and $\boldsymbol{\rho}_{DE}^{(\text{coeff})} = [3.22862 \times 10^{-4}, 1.19006 \times 10^{-3}, 0.601297, 0.385985, 2.03269 \times 10^{-3}, 2.76451 \times 10^{-3}, 1.91570 \times 10^{-3}, 1.97568 \times 10^{-3}, 1.28470 \times 10^{-3}, 1.23207 \times 10^{-3}]$. Fig. 3.5 depicts the right degree distribution. Based on density evolution, this pair of degree distributions achieves $\epsilon_{FF} = 6.58 \times 10^{-4}$. The throughput of this design is $R_t^{(FF)} = 0.555961$ bits per transmission.

We used PEG to generate an actual inter-frame bipartite graph with **1000** left nodes that approximates the differential evolution degree distribution. The actual irregular degree distributions of the bipartite graph are $\lambda(x) = x^3$ and $\boldsymbol{\rho}_{PEG}^{(\text{coeff})} = [0.00175, 0.0085, 0.57525, 0.392, 0.0125, 0.0015, 0.00175, 0.002, 0.00225, 0.0025]$. The corresponding $\beta_{FF} = 1.198$. Simulation with the genie shows that the failure rate is $\epsilon_{FF} = 6.25 \times 10^{-4}$. The throughput of this design is $R_t^{(FF)} = 0.555968$ bits per transmission, achieving 96.2% of the $R_t^{(FB)}$ of the NB-LDPC VL code with ACK/NACK feedback. Limiting the number of left nodes to 1000 does not significantly impact the performance in this case.

2) Designs using quasi-regular degree distributions:

For this design, we fix $\lambda(x) = x^3$ and $\rho_{QR}(x) = \alpha x^2 + (1 - \alpha)x^3$, where α is the design parameter. Table 3.7 shows several designs with various α values. For each row, we fix the

Table 3.7: Density evolution performance characterization of quasi-regular $\rho_{QR}(x) = \alpha x^2 + (1-\alpha)x^3$ for the NB-LDPC VL code based feedback-free system. $\lambda(x) = x^3$ in all cases.

α	a_R	β_{FF}	$R_t^{(FF)}$	No. Iter.	% $R_t^{(FB)}$	ϵ_{FF}
0.597	3.33611	1.19900	0.555892	20	96.2%	6.55×10^{-4}
0.341	3.59174	1.11367	0.560870	30	97.0%	6.82×10^{-4}
0.273	3.66636	1.09100	0.562206	40	97.3%	6.90×10^{-4}
0.246	3.69686	1.08200	0.562739	50	97.4%	6.93×10^{-4}
0.217	3.73018	1.07233	0.563312	100	97.5%	6.97×10^{-4}

Note: "No. iter." is the number of iterations for the GPD to achieve the listed ϵ_{FF}. "% $R_t^{(FB)}$" is the percentage $R_t^{(FF)}$ is of $R_t^{(FB)}$ of the best feedback system (the first row in Table 3.3).

Table 3.8: Right degree distributions of the PEG generated inter-frame bipartite graphs in Fig. 3.9.

α	No. Left Nodes	ρ_2	ρ_3	ρ_4	ρ_5
0.597	1000	0.0035	0.59025	0.4	0.00625
0.597	2000	0.00275	0.591375	0.4015	0.004375
0.341	10000	0.0016	0.342225	0.6463	0.009875
0.341	20000	0.00055	0.3411	0.65535	0.003

Note: $\rho_i = 0$ if not listed.

maximum number of iterations, and identify the smallest value of α (with a precision of 0.001) that achieves a probability of failure which is less than $\epsilon_{FF} = 7 \times 10^{-4}$ according to density evolution.

We used PEG to generate actual bipartite graphs for the first two designs in Table 3.7 with different numbers of left nodes, resulting in the right degree distributions shown in Table 3.8. They approximate the ideal degree indicated by α.

Fig. 3.9 shows probability of failure ϵ_{FF} of the PEG-generated codes in Table 3.8 according to simulation. Also shown is ϵ_{FF} for the ideal quasi-regular distributions defined by the associated α. Real codes with a finite number of left nodes have a higher failure probability ϵ_{FF} than predicted by density evolution. Similarly, codes with a limited number of left nodes require more iterations to converge than predicted by density evolution. For example, the design for $\alpha = 0.597$ with 1000 left nodes requires 25 iterations to achieve $\epsilon_{FF} = 7.32 \times 10^{-4}$, while density evolution (or $r_1(x)$ analysis) predicts the failure probability to be 6.55×10^{-4}

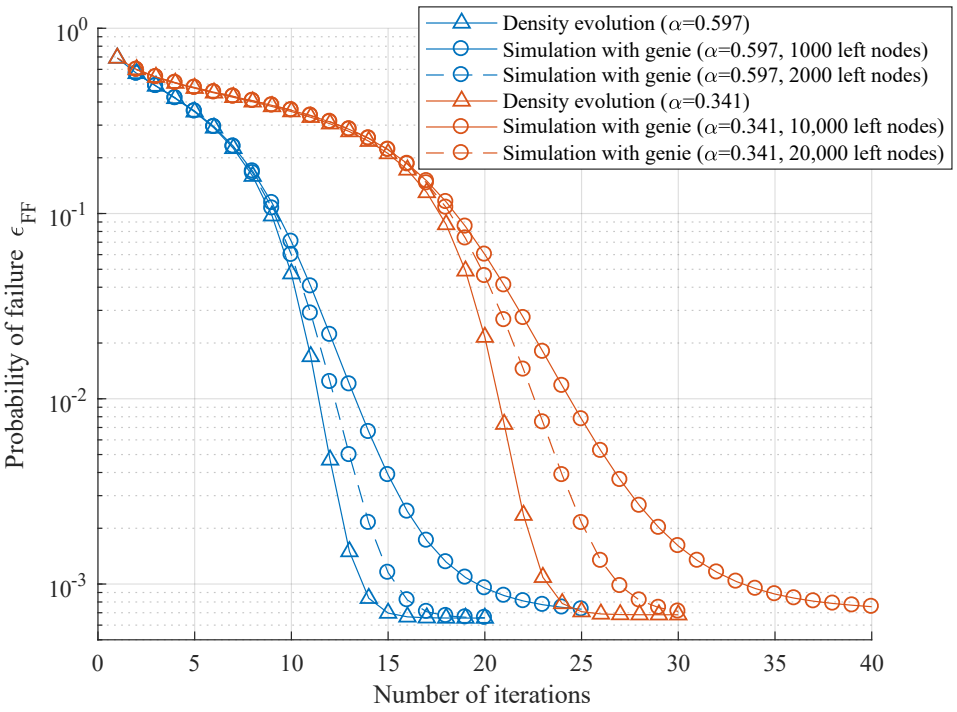


Figure 3.9: Probability of failure versus the number of iterations for the designs in Table 3.7 from density evolution and genie-aided simulations.

at 20 iterations. Codes with a slightly higher percentage of $R_t^{(FB)}$ can require many more left nodes to achieve the predicted performance. In Fig. 3.9, the $\alpha = 0.597$ design achieves 96% of $R_t^{(FB)}$ and requires 2000 left nodes to achieve asymptotic ϵ_{FF}. The $\alpha = 0.391$ design achieves 97% of $R_t^{(FB)}$ but requires 20000 left nodes to achieve asymptotic ϵ_{FF}.

Performance with CRC

We also designed and simulated systems using cyclic redundancy check (CRC) to detect decoding success instead of a genie. With low probability, the CRC decoder can fail to detect an incorrect codeword causing an undetected frame error [75]. In our system, the undetected errors cause the VL decoders to incorrectly declare decoding success, and pass incorrect increments to the right nodes causing error propagation.

The design of the VL codes already provides the margin needed to mitigate this issue by targeting a design failure rate at $5 \times 10^{-4} < 10^{-3}$ (the third row in Table 3.3). Additionally,

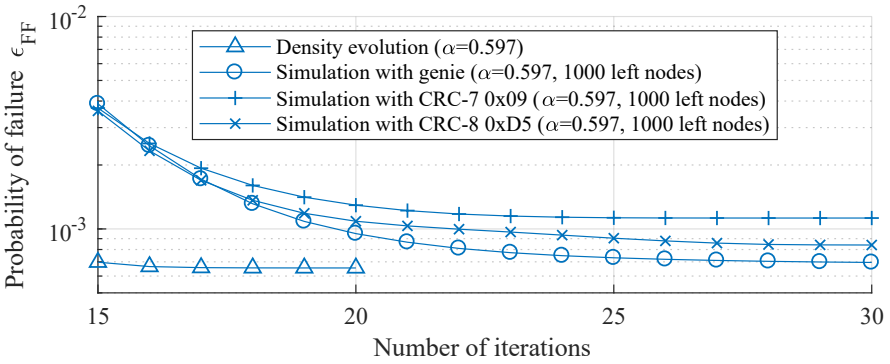


Figure 3.10: Probability of failure versus the number of iterations for the $\alpha = 0.597$ 1000-left-node design in Table 3.8 from density evolution, simulations with a genie-aided decoder and decoders using CRCs.

Table 3.9: Probability of failure of 1000-left-node $\alpha = 0.597$ code from Table 3.8 simulated with CRCs.

CRC	No. Iter.	Detected Failure	Undetected Error	Real ϵ_{FF}
7-bit 0x09	25	7.40×10^{-4}	3.89×10^{-4}	1.13×10^{-3}
8-bit 0xD5	25	7.15×10^{-4}	1.89×10^{-4}	9.04×10^{-4}

Note: "No. iter." is the number of iterations for the GPD to achieve the listed ϵ_{FF}. Real ϵ_{FF} is the sum of the CRC detected failure rate and the undetected error rate.

we uses a failure probability target of 7×10^{-4} for the quasi-regular degree distribution designs, leaving margin for undetected errors.

We examined both a 7-bit and a 8-bit CRC for the $\alpha = 0.597$ 20-iteration design with 1000 left nodes. The 7-bit CRC has a polynomial representation of 0x09 ($x^7 + x^3 + 1$), as defined in standards G.707, G.832 of the International Telecommunication Union (ITU-T) [76, 77]. The polynomial representation of the 8-bit CRC is 0xD5 ($x^8+x^7+x^6+x^4+x^2+1$). This CRC is used in the standard Digital Video Broadcasting - Satellite - Second Generation (DVB-S2) [78].

Fig. 3.10 shows the probability of failure curves for the simulation using a genie and the simulations using CRCs. Table 3.9 shows the failure probability of the simulations with the two CRCs at 25 GPD iterations. The ϵ_{FF} is the failure probability including both the undetected errors and detected failures. We conclude that the undetected errors which inject wrong increments into the GPD do not stall the decoding process, and only increase the real

failure probability ϵ_{FF} modestly. The undetected error rate of the 7-bit CRC is roughly two times of that of the 8-bit CRC. As a result, the 8-bit CRC is more desirable for the inter-frame system in this paper.

For our scenario, the overhead of the CRC causes approximately a 4% throughput reduction from the system with a genie, which achieved 96.02% of $R_t^{(FB)}$. Specifically, with 25 GPD iterations, the real throughput of the inter-frame code with the 7-bit CRC is 0.535371, which is 92.6% of the feedback throughput $R_t^{(FB)}$ in the first row of Table 3.3. The real throughput of the inter-frame code with the 8-bit CRC is 0.532598, achieving 92.1% of the $R_t^{(FB)}$.

To summarize, using the inter-frame coding architecture, a practical NB-LDPC VL code based feedback-free error correction design can achieve 92.1% of the $R_t^{(FB)}$ of the reference NB-LDPC VL code with feedback, and 82.9% of 2dB BI-AWGN channel capacity, while guaranteeing a failure probability below 10^{-3} with 25 GPD iterations.

3.6 Conclusion

A full design and simulation demonstrated that the throughput and probability of failure (frame error rate) performance of a VL code with ACK/NACK feedback on a point-to-point channel can be closely approached without feedback by using multiple such VL codes in parallel with common incremental redundancy shared through inter-frame coding. The results suggests a constructive proof (for the special case of ACK/NACK feedback) of the well-known information theoretic result that feedback does not increase capacity of point-to-point channels. The inter-frame decoding complexity is small so that the overall system's complexity is comparable to decoding the component VL codes with ACK/NACK feedback, and allows the opportunity for massive parallel processing.

The designs can be improved by exploring the applicability of additional VL encoder architectures, and analyzing the performance on more channel conditions. Further research can be conducted to identify the limits of the feedback-free system, such as a lower bound

on the number of left nodes required and a upper bound of the throughput.

3.7 Acknowledgment

The majority of this chapter has been published or has been submitted for publication in [5, 6, 7]. This research was conducted in collaboration with Sudarsan V.S. Ranganathan and Prof. Richard D. Wesel. The author would like to thank Sudarsan V.S. Ranganathan for providing the PEG design software. The author would also like to thank Fabian Steiner at Technical University of Munich (TUM) for valuable suggestions and comments.

References

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

[2] H. Wang, T. Chen, and R. D. Wesel, "Histogram-based Flash channel estimation," in *Proc. 2015 IEEE Int. Conf. Commun. (ICC)*, 2015, pp. 283–288.

[3] H. Wang, N. Wong, and R. D. Wesel, "Dynamic voltage allocation with quantized voltage levels and simplified channel modeling," in *2015 49th Asilomar Conference on Signals, Systems and Computers*, Nov 2015, pp. 834–838.

[4] H. Wang, N. Wong, T. Chen, and R. D. Wesel, "Using dynamic allocation of write voltage to extend flash memory lifetime," *IEEE Trans. Commun.*, vol. 64, no. 11, pp. 4474–4486, Nov 2016.

[5] H. Wang, S. V. S. Ranganathan, and R. D. Wesel, "Approaching capacity using incremental redundancy without feedback," in *Proc. 2017 IEEE International Symposium on Information Theory (ISIT)*, 2017, pp. 161–165.

[6] H. Wang and R. D. Wesel, "Channel code analysis and design using multiple variable-length codes in parallel without feedback," in *Proc. 2018 IEEE Global Communications Conference (GLOBECOM)*, 2018, to be published.

[7] ——, "Coding with shared incremental redundancy: Design methods and a non-binary ldpc example," *IEEE Trans. Commun.*, 2018, submitted for publication.

[8] J. Choi and K. S. Seol, "3D approaches for non-volatile memory," in *Proc. 2011 Symp. VLSI Technology (VLSIT)*, 2011, pp. 178–179.

[9] K. Park, D. Byeon, and D. Kim, "A world's first product of three-dimensional vertical NAND Flash memory and beyond," presented at 2014 14th Annu. Non-Volatile Memory Technology Symp. (NVMTS), Jeju Island, ROK, Oct. 27-29, 2014.

[10] J. Im, W. Jeong, D. Kim, S. Nam, D. Shim, M. Choi, H. Yoon, D. Kim, Y. Kim, H. Park, D. Kwak, S. Park, S. Yoon, W. Hahn, J. Ryu, S. Shim, K. Kang, S. Choi, J. Ihm, Y. Min, I. Kim, D. Lee, J. Cho, O. Kwon, J. Lee, M. Kim, S. Joo, J. Jang, S. Hwang, D. Byeon, H. Yang, K. Park, K. Kyung, and J. Choi, "A 128Gb 3b/cell V-NAND Flash memory with 1Gb/s I/O rate," in *Proc. 2015 IEEE Int. Solid-State Circuits Conf. (ISSCC) Tech. Dig.*, 2015, pp. 110–112.

[11] B. Chen, X. Zhang, and Z. Wang, "Error correction for multi-level NAND Flash memory using Reed-Solomon codes," in *Proc. 2008 IEEE Workshop on Signal Process. Syst. (SiPS)*, 2008, pp. 94–99.

[12] S. Cho, D. Kim, J. Choi, and J. Ha, "Block-wise concatenated BCH codes for NAND Flash memories," *IEEE Trans. Commun.*, vol. 62, no. 4, pp. 1164–1177, Apr. 2014.

[13] D. Kim and J. Ha, "Quasi-primitive block-wise concatenated BCH codes for NAND Flash memories," in *Proc. 2014 IEEE Inform. Theory Workshop (ITW)*, 2014, pp. 611–615.

[14] Y. Maeda and H. Kaneko, "Error control coding for multilevel cell Flash memories using nonbinary low-density parity-check codes," in *Proc. 2009 24th IEEE Int. Symp. on Defect and Fault Tolerance in VLSI Systems (DFT)*, 2009, pp. 367–375.

[15] K. Zhao, W. Zhao, H. Sun, T. Zhang, X. Zhang, and N. Zheng, "LDPC-in-SSD: Making advanced error correction codes work effectively in solid state drives," in *Proc. 11th USENIX Conf. on File and Storage Technologies*, 2013, pp. 243–256.

[16] F. Zhang, H. D. Pfister, and A. Jiang, "LDPC codes for rank modulation in Flash memories," in *Proc. 2010 IEEE Int. Symp. on Inform. Theory (ISIT)*, 2010, pp. 859–863.

[17] J. Wang, K. Vakilinia, T. Chen, T. Courtade, G. Dong, T. Zhang, H. Shankar, and R. Wesel, "Enhanced precision through multiple reads for LDPC decoding in Flash memories," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 5, pp. 880–891, May 2014.

[18] J. Jeong, S. S. Hahn, S. Lee, and J. Kim, "Improving NAND endurance by dynamic program and erase scaling," presented at the 5th USENIX Conf. on Hot Topics in Storage and File Syst., San Jose, CA, Jun 27-28, 2013.

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

[20] A. Jiang, "On the generalization of error-correcting WOM codes," in *Proc. 2007 IEEE Int. Symp. on Inform. Theory (ISIT)*, 2007, pp. 1391–1395.

[21] E. Yaakobi, S. Kayser, P. H. Siegel, A. Vardy, and J. K. Wolf, "Codes for write-once memories," *IEEE Trans. Inf. Theory*, vol. 58, no. 9, pp. 5985–5999, Sep. 2012.

[22] R. Gabrys and L. Dolecek, "Constructions of nonbinary WOM codes for multilevel Flash memories," *IEEE Trans. Inf. Theory*, vol. 61, no. 4, pp. 1905–1919, Apr. 2015.

[23] A. Jiang, M. Schwartz, and J. Bruck, "Error-correcting codes for rank modulation," in *Proc. 2008 IEEE Int. Symp. Inform. Theory (ISIT)*, 2008, pp. 1736–1740.

[24] A. Mazumdar, A. Barg, and G. Zemor, "Constructions of rank modulation codes," *IEEE Trans. Inf. Theory*, vol. 59, no. 2, pp. 1018–1029, Feb. 2013.

[25] M. Qin, A. Jiang, and P. H. Siegel, "Parallel programming of rank modulation," in *Proc. 2013 IEEE Int. Symp. Inform. Theory (ISIT)*, 2013, pp. 719–723.

[26] B. Peleato, R. Agarwal, J. M. Cioffi, M. Qin, and P. H. Siegel, "Adaptive read thresholds for NAND Flash," *IEEE Trans. Commun.*, vol. 63, no. 9, pp. 3069–3081, Sep. 2015.

[27] H. Zhou, A. Jiang, and J. Bruck, "Error-correcting schemes with dynamic thresholds in nonvolatile memories," in *Proc. 2011 IEEE Int. Symp. Inform. Theory (ISIT)*, 2011, pp. 2143–2147.

[28] F. Sala, R. Gabrys, and L. Dolecek, "Dynamic threshold schemes for multi-level non-volatile memories," *IEEE Trans. Commun.*, vol. 61, no. 7, pp. 2624–2634, Jul. 2013.

[29] T. Chen, A. R. Williamson, and R. D. Wesel, "Increasing flash memory lifetime by dynamic voltage allocation for constant mutual information," in *Proc. 2014 Inform. Theory and Applicat. Workshop (ITA)*, Feb. 2014, pp. 1–5.

[30] D. Lee and W. Sung, "Estimation of NAND Flash memory threshold voltage distribution for optimum soft-decision error correction," *IEEE Trans. Signal Process.*, vol. 61, no. 2, pp. 440–449, Jan. 2013.

[31] J. Wang, T. Courtade, H. Shankar, and R. Wesel, "Soft information for LDPC decoding in Flash: mutual-information optimized quantization," presented at 2011 IEEE Global Telecommun. Conf. (GLOBECOM), Houston, TX, Dec. 5-9, 2011.

[32] T. Parnell, N. Papandreou, T. Mittelholzer, and H. Pozidis, "Modelling of the threshold voltage distributions of sub-20nm NAND Flash memory," in *Proc. 2014 IEEE Global Commun. Conf. (GLOBECOM)*, 2014, pp. 2351–2356.

[33] K. Takeuchi, T. Tanaka, and H. Nakamura, "A double-level-Vth select gate array architecture for multilevel NAND Flash memories," *IEEE J. Solid-State Circuits*, vol. 31, no. 4, pp. 602–609, Apr. 1996.

[34] C. Compagnoni, A. Spinelli, R. Gusmeroli, A. Lacaita, S. Beltrami, A. Ghetti, and A. Visconti, "First evidence for injection statistics accuracy limitations in NAND Flash

constant-current Fowler-Nordheim programming," in *Proc. 2007 IEEE Int. Electron Devices Meeting (IEDM)*, 2007, pp. 165–168.

[35] P. Olivo, B. Ricco, and E. Sangiorgi, "High-field-induced voltage-dependent oxide charge," *Appl. Physics Lett.*, vol. 48, no. 17, pp. 1135–1137, Apr. 1986.

[36] N. Mielke, H. Belgal, I. Kalastirsky, P. Kalavade, A. Kurtz, Q. Meng, N. Righos, and J. Wu, "Flash EEPROM threshold instabilities due to charge trapping during program/erase cycling," *IEEE Trans. Device Mater. Rel.*, vol. 4, no. 3, pp. 335–344, Sep. 2004.

[37] P. Cappelletti, R. Bez, D. Cantarelli, and L. Fratin, "Failure mechanisms of Flash cell in program/erase cycling," in *Proc. 1994 IEEE Int. Electron Devices Meeting (IEDM)*, 1994, pp. 291–294.

[38] S. Yamada, Y. Hiura, T. Yamane, K. Amemiya, Y. Ohshima, and K. Yoshikawa, "Degradation mechanism of Flash EEPROM programming after program/erase cycles," in *Proc. 1993 IEEE Int. Electron Devices Meeting (IEDM)*, 1993, pp. 23–26.

[39] K. Fukuda, Y. Shimizu, K. Amemiya, M. Kamoshida, and C. Hu, "Random telegraph noise in Flash memories - model and technology scaling," in *Proc. 2007 IEEE Int. Electron Devices Meeting (IEDM)*, 2007, pp. 169–172.

[40] C. Compagnoni, M. Ghidotti, A. Lacaita, A. Spinelli, and A. Visconti, "Random telegraph noise effect on the programmed threshold-voltage distribution of Flash memories," *IEEE Electron Device Lett.*, vol. 30, no. 9, pp. 984–986, Sep. 2009.

[41] G. Dong, Y. Pan, N. Xie, C. Varanasi, and T. Zhang, "Estimating information-theoretical NAND Flash memory storage capacity and its implication to memory system design space exploration," *IEEE Trans. VLSI Syst.*, vol. 20, no. 9, pp. 1705–1714, Sep. 2012.

[42] G. Dong, S. Li, and T. Zhang, "Using data postcompensation and predistortion to tolerate cell-to-cell interference in MLC NAND Flash memory," *IEEE Trans. Circuits Syst. I*, vol. 57, no. 10, pp. 2718–2728, Oct. 2010.

[43] J. Lee, S. Hur, and J. Choi, "Effects of floating-gate interference on NAND Flash memory cell operation," *IEEE Electron Device Lett.*, vol. 23, no. 5, pp. 264–266, May 2002.

[44] N. Mielke, H. Belgal, A. Fazio, Q. Meng, and N. Righos, "Recovery effects in the distributed cycling of Flash memories," in *Proc. 2006 42nd Annu. IEEE Int. Reliability Physics Symp.*, 2006, pp. 29–35.

[45] H. Yang, H. Kim, S. Park, J. Kim, S. Lee, J. Choi, D. Hwang, C. Kim, M. Park, K. Lee, Y. Park, J. K. Shin, and J. Kong, "Reliability issues and models of sub-90nm NAND Flash memory cells," in *Proc. 2006 8th Int. Conf. Solid-State and Integrated Circuit Technology*, 2006, pp. 760–762.

[46] J. Lee, J. Choi, D. Park, and K. Kim, "Data retention characteristics of sub-100 nm NAND Flash memory cells," *IEEE Electron Device Lett.*, vol. 24, no. 12, pp. 748–750, Dec. 2003.

[47] M. Gill and S. Lai, "Flash reliability issues," in *Nonvolatile semiconductor memory technology: a comprehensive guide to understanding and to using NVSM devices*, W. D. Brown and J. Brewer, Eds. New York, NY: IEEE Press, 1998, ch. 4, sec. 6, pp. 255–281.

[48] T. M. Cover and J. A. Thomas, "Relationship between entropy entropy and mutual information," in *Elements of Information Theory*, 2nd ed. Hoboken, NJ: Wiley-Interscience, 2006, ch. 2, sec. 4, pp. 20–22.

[49] Q. Xu, P. Gong, T. M. Chen, J. Michael, and S. Li, "Modelling and characterization of NAND Flash memory channels," *Measurement*, vol. 70, pp. 225–231, Jun. 2015.

[50] 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. pp. 431–441, 1963.

[51] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, 10th ed. Washington, D.C.: U.S. Dept. of Commerce, 1972, p. 890.

[52] C. Lott, O. Milenkovic, and E. Soljanin, "Hybrid ARQ: Theory, state of the art and future directions," in *Prof. IEEE Inf. Theory Workshop (ITW)*, Bergen, Norway, 2007.

[53] D. J. Costello, J. Hagenauer, H. Imai, and S. B. Wicker, "Applications of error control coding," *IEEE Trans. Inform. Theory*, vol. 44, no. 6, pp. 2531–2560, Oct. 1998.

[54] A. M. Cipriano, P. Gagneur, G. Vivier, and S. Sezginer, "Overview of ARQ and HARQ in beyond 3G systems," in *Proc. 2010 IEEE 21st Int. Symp. on Personal, Indoor and Mobile Radio Communications Workshops*, Instanbul, Turkey, 2010.

[55] A. Masaracchia, R. Bruno, A. Passarella, and S. Mangione, "Analysis of MAC-level throughput in LTE systems with link rate adaptation and HARQ protocols," in *Proc. 2015 IEEE 16th Int. Symp. on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, Boston, MA, USA, 2015.

[56] E. Cabrera, G. Fang, and R. Vesilo, "Adaptive hybrid ARQ (A-HARQ) for ultra-reliable communication in 5G," in *Proc. IEEE Vehicular Technology Conf. (VTC Spring)*, Sydney, NSW, Australia, 2017.

[57] G. Forney, "Exponential error bounds for erasure list and decision feedback schemes," *IEEE Trans. Inf. Theory*, vol. 14, no. 2, pp. 206–220, Mar. 1968.

[58] M. V. Burnashev, "Data transmission over a discrete channel with feedback. random transmission time," *Problems Inf. Transmission*, vol. 12, no. 4, pp. 250–265, 1976.

[59] Y. Polyanskiy, H. V. Poor, and S. Verdu, "Feedback in the non-asymptotic regime," *IEEE Trans. Inf. Theory*, vol. 57, no. 8, pp. 4903–4925, Aug. 2011.

[60] A. R. Williamson, T. Y. Chen, and R. D. Wesel, "Variable-length convolutional coding for short blocklengths with decision feedback," *IEEE Trans. Commun.*, vol. 63, no. 7, pp. 2389–2403, Jul. 2015.

[61] K. Vakilinia, S. V. S. Ranganathan, D. Divsalar, and R. D. Wesel, "Optimizing transmission lengths for limited feedback with nonbinary LDPC examples," *IEEE Trans. Commun.*, vol. 64, no. 6, pp. 2245–2257, Jun. 2016.

[62] H. Zeineddine and M. M. Mansour, "Inter-frame coding for broadcast communication," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 2, pp. 437–452, Feb. 2016.

[63] E. Casini, R. D. Gaudenzi, and O. D. R. Herrero, "Contention resolution diversity slotted aloha (crdsa): An enhanced random access schemefor satellite access packet networks," *IEEE Trans. Wireless Commun.*, vol. 6, no. 4, pp. 1408–1419, April 2007.

[64] G. Liva, "Graph-based analysis and optimization of contention resolution diversity slotted aloha," *IEEE Trans. Commun.*, vol. 59, no. 2, pp. 477–487, February 2011.

[65] E. Paolini, G. Liva, and M. Chiani, "Coded slotted aloha: A graph-based method for uncoordinated multiple access," *IEEE Trans. Inf. Theory*, vol. 61, no. 12, pp. 6815–6832, Dec 2015.

[66] E. Sandgren, A. G. i Amat, and F. Brannstrom, "On frame asynchronous coded slotted aloha: Asymptotic, finite length, and delay analysis," *IEEE Trans. Commun.*, vol. 65, no. 2, pp. 691–704, Feb 2017.

[67] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Efficient erasure correcting codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 569–584, Feb. 2001.

[68] J.-F. Cheng and R. J. Mceliece, "Some high-rate near capacity codecs for the Gaussian channel," in *Proc. of 34th Allerton Conf. on Commun., Control and Comput.*, 1996, pp. 494–503.

[69] M. G. Luby, M. Mitzenmacher, and M. A. Shokrollahi, "Analysis of random processes via and-or tree evaluation," in *Proc. the Ninth Ann. ACM-SIAM Symp. on Discrete Algorithms*, Philadelphia, PA, USA, 1998, pp. 364–373.

[70] K. Vakilinia, T. Y. Chen, S. V. S. Ranganathan, A. R. Williamson, D. Divsalar, and R. D. Wesel, "Short-blocklength non-binary ldpc codes with feedback-dependent incremental transmissions," in *Proc. 2014 IEEE International Symposium on Information Theory (ISIT)*, 2014, pp. 426–430.

[71] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 619–637, Feb 2001.

[72] A. Shokrollahi and R. M. Storn, "Design of efficient erasure codes with differential evolution," in *Differential Evolution: A Practical Approach to Global Optimization*. Berlin, Germany: Springer, 2005, ch. 7, pp. 413–426.

[73] R. L. Smith, "Efficient monte carlo procedures for generating points uniformly distributed over bounded regions," *Oper. Res.*, vol. 32, no. 6, pp. 1296–1308, Nov. 1984.

[74] X.-Y. Hu, E. Eleftheriou, and D. M. Arnold, "Regular and irregular progressive edge-growth tanner graphs," *IEEE Trans. Inf. Theory*, vol. 51, no. 1, pp. 386–398, Jan. 2005.

[75] P. Koopman and T. Chakravarty, "Cyclic redundancy code (CRC) polynomial selection for embedded networks," in *Proc. Int. Conf. on Dependable Systems and Networks*, 2004, pp. 145–154.

[76] "Network node interface for the synchronous digital hierarchy (SDH)," ITU-T Standard G.707, 2007.

[77] "Transport of SDH elements on PDH networks - frame and multiplexing structures," ITU-T Standard G.832, 1998.

[78] "Digital video broadcasting (DVB); second generation framing structure, channel coding and modulation systems for broadcasting, interactive services, news gathering and other broadband satellite applications (DVB-S2)," ETSI Standard EN 302 307, 2014.