Data Reliability and Error Correction for NAND Flash Memory System

Quan Xu

School of Mathematics, Computer Science & Engineering

City University London

This dissertation is submitted for the degree of

Doctor of Philosophy

June 2016

Declaration

This thesis is presented in fulfilment of the requirement for the degree of Doctor of Philosophy. The research reported in this doctoral thesis has been carried out at the City University London. I declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

Quan Xu

June 2016

Acknowledgements

I take this opportunity to express my sincere gratitude to my advisor, Professor Tom Chen, for his continuing guidance, support and encouragement. His enthusiasm and expertise in the area of signal processing, error correction coding and probability theory help me better understand the subtle facets of this work. His rigorous attitude toward science stimulates me throughout the writing of thesis.

I also thank my officemates, Mr Pu Gong, Mr Tareq Alalyani and Dr Jorge Blasco. Their advice and ideas help me walk forward and are very important to my research. Working with these colleagues has not only been a good learning experience, but also a great pleasure. In addition, I acknowledge iStorage Limited and Mr John Michael for his great support to my research project in flash channel modelling.

I would like to thank the coauthors of my publications for their collaborations. Special thanks to Dr Yupeng Hu at the Hunan University, and Dr Shancang Li at the Edinburgh Napier University for their help in applying LDPC coding to the flash storage error correction. I also acknowledge Qiang Liao, Jian Yu in Initio Corporation for their valuable advice on the implementation of ECC decoder and the data pattern processing schemes.

I am sincerely grateful to my parents and my fiancée for always being supportive and encouraging in my pursuit of academic excellence. Without them, I would not ever have completed this dissertation.

Abstract

NAND flash memory has been widely used for data storage due to its high density, high throughput, and low power. However, as the flash memory scales to smaller process technologies and stores more bits per cell, its reliability is decreasing. The error correction coding can be used to significantly improve the data reliability; nevertheless, the advanced ECCs such as low-density parity-check (LDPC) codes generally demand soft-decisions while NAND flash memory channel provides hard-decisions only. Extracting the soft information requires the accurate characterization of flash memory channel and the effective design of coding schemes.

To this end, we have presented a novel LDPC-TCM coding scheme for the Multi-level Cell (MLC) flash memories. The a posteriori TCM decoding algorithm is used in the scheme to generate soft information, which is fed to the LDPC decoder for further correction of data bits. It has been demonstrated that the proposed scheme can achieve higher error correction performance than the traditional hard-decisions based flash coding algorithms, and is feasible in the design practice. Further with the LDPC-TCM, we believe it is important to characterize the flash memory channel and investigate a method to calculate the soft decision for each bit, with the available channel outputs. We studied the various noises and interferences occurring in the memory channel and mathematically formulated the probability density function of the overall noise distribution. Based on the results we derived the final distribution for the cell threshold voltages, which can be used to instruct the calculation of soft decisions. The discoveries on the theoretical level have been demonstrated to be consistent with the real channel behaviours. The channel characterization and model provided in this dissertation can enable more design of soft-decisions based ECCs for future NAND flash memories.

The data pattern processing algorithm deals with the write patterns and targets to lower the proportion of patterns that would introduce data errors. On the other hand, the voltages applied to the memory cells charges the MOSFET capacitances frequently on programming these data patterns, leading to the power problem. The high energy consumption and current spikes also cause reliability issue to the data stored in the flash memory. This dissertation proposes a write pattern formatting algorithm (WPFA) attempting to solve the two problems together. We have designed and implemented the algorithm and evaluated its performance through both the software simulations and hardware synthesis.

Table of contents

List of figures **ix**

List of tables **xii**

List of Abbreviations **xiii**

List of Notations **xiv**

1 Introduction **1**

1.1 Motivation 3

1.2 Goals 8

1.3 Thesis Contribution 9

1.4 Thesis Outline 12

2 Background **14**

2.1 Basics of NAND Flash Memory 14

2.1.1 NAND Device Organization 15

2.1.2 NAND Cell Architecture and Basic Operations 16

2.1.3 Planar and 3D NAND Flash Memory 21

2.2 Data Reliability Issues 23

2.2.1 Reliability at Physical Level 23

2.2.2 Noises in NAND Flash Memory 24

2.2.3 Error Correction Codes for Flash Memories 30

2.3 NAND-based Memory System Overview 32

2.4 Summary and Challenges 34

3 Write Pattern Formatting Algorithm and Implementation 35

3.1 Problem Description 37

3.2 Related Work 40

3.3 Write Pattern Formatting Algorithm 44

3.4 Performance Evaluation 47

3.4.1 Maximum Length of Column Stripe Patterns 47

3.4.2 Average Program Current 48

3.4.3 Proportion of the Highest V_{th} State 50

3.4.4 Overhead of NAND Cell Area 54

3.5 Hardware Design and Implementation Complexity 56

3.5.1 Hardware Implementation of WPFA Encoder and Decoder . . . 58

3.5.2 Analysis of Implementation Complexity 60

3.6 Summary and Challenges 61

4 Concatenated LDPC-TCM Error Correction Coding 64

4.1 Problem Description 65

4.2 Background 67

4.2.1 Basics of Trellis Coded Modulation 68

4.2.2 Low Density Parity Check Codes 74

4.2.3 Related Work 77

4.3 Concatenated LDPC-TCM Coding System 79

4.3.1 System Description 79

4.3.2 Soft Decisions Calculation 82

4.3.3 Theoretical Analysis of Coded Modulation in Flash Memory . . 84

4.4 Performance Evaluation 86

4.5 Summary and Challenges 93

5 Modelling and Characterisation of NAND Flash Memory Channels 97

5.1 Problem Description 98

5.2 Background 100
5.2.1 Equivalent Channel Model Incorporated with Major Threshold Voltage Distortions 100
5.2.2 Characteristic Functions of Random Variables 107
5.2.3 Related Work 110
5.3 Characterisation of NAND Flash Memory Channel 111
5.3.1 Overall Distribution of the Noises and Distortions 112
5.3.2 Cell Threshold Voltage Distributions 124
5.4 Calculation of Soft Decisions 127
5.5 Summary and Challenges 129

6 Conclusions and Future Work 131
6.1 Thesis Conclusion 131
6.2 Directions for Future Work 133

References 136

List of Related Publications 142

List of figures

1.1 Overview of the thesis organization. 12

2.1 NAND flash memory array structure 16
2.2 Flash memory cell vs. MOSFET. 17
2.3 Control-gate voltage in program operation. 19
2.4 Read reference voltages for 4-level MLC. 20
2.5 A Typical structure of VG flash memory. 22
2.6 Capacitors involved in the NAND flash memory channel. 25
2.7 Cell to cell coupling trend. 26
2.8 Functional representation of a NAND-based memory system. 33

3.1 Functions implemented in the flash translation layer. 36
3.2 Example of multipage programming for 2 bits/cell NAND flash memory. 38
3.3 Operation of pre-charging under different programming data patterns . . 40
3.4 (a) Algorithm of the asymmetric coding. (b) Schematic circuit diagram of the asymmetric coding encoder. 41
3.5 Example of SPEA. 43
3.6 WPFA: lower page data. 44
3.7 Analysis of two specific data patterns 45
3.8 WPFA: upper page data. 47
3.9 Reduced program current over the system without power saving scheme 50
3.10 Data patterns before and after WPFA processing. 51

3.11 Simulation results of probabilities of "1"s in Tanakamaru's and the proposed schemes. 53
3.12 V_{th} distributions of the worst case, randomized interleaving, and the proposed algorithm. 54
3.13 The number of flag bits required for WPFA and Tanakamaru's algorithm. 55
3.14 The reduced NAND cell area overhead to Tanakamaru's design. 56
3.15 Framework of coding mechanism in SSDs. 57
3.16 Circuit schematic of the joint lower/upper pages WPFA encoder 59
3.17 Circuit schematic of the joint lower/upper pages WPFA decoder 59

4.1 A general trellis coded modulation 69
4.2 Pragmatic TCM uses a standard rate 1/2 convolutional code. 70
4.3 Tanner graph representation for an LDPC code 75
4.4 The approximate flash memory cell threshold voltage distribution model. 80
4.5 Block diagram of TCM LDPC coding system. 81
4.6 State transitions in pragmatic TCM. 83
4.7 Asymptotic coding gain of TCM for MLC flash memories and M-PAM modulation. 87
4.8 Performance comparisons for the raw BERs of 4 levels, 8 levels MLC and TCM without quantization. 89
4.9 Performance comparisons of the raw BER of 4 levels and the TCM with quantization 90
4.10 Performance comparisons of the raw BER of 4 levels, the BCH coding and the TCM for NAND flash memories. 91
4.11 Performance comparisons of the raw BER of 4 levels, the BCH coding and the proposed LDPC-TCM coding. 92
4.12 Performance comparisons of the raw BER of 4 levels, the BCH coding and the proposed LDPC-TCM coding. 94
4.13 Performance comparisons of the BCH coding and the proposed LDPC-TCM coding with quantized flash channel outputs. 95

5.1 Approximate NAND flash memory device model to incorporate major threshold voltage distortion sources. 102
5.2 Simulation of threshold-voltage distribution after 1K P/E Cycling and 1 month retention: memory cell suffers CCI from vertical and diagonal neighbouring cells 104
5.3 Simulation of memory threshold-voltage distribution after 1K P/E Cycles and 1 month retention: memory cell suffers CCI (random coupling ratio) from neighbouring cell in vertical direction only 105
5.4 Simulation of memory threshold-voltage distribution after 1K P/E Cycles and 1 month retention: memory cell suffers CCI (constant coupling ratio) from neighbouring cell in vertical direction only 106
5.5 Contours used for calculating the noise probability density function . . 115
5.6 The effects of RTN, CCI, and retention noise on cell threshold voltage distribution after 1K P/E cycling and 1 year retention (from top to bottom: L_0 to L_3). 123
5.7 Simulated curves to show the effects of RTN, CCI, and retention noise on memory cell threshold voltage distribution after 1K P/E cycling and 1 year retention. 124
5.8 Cell threshold voltage distributions 4-level MLC flash memory 127

List of tables

1.1 NAND Characteristics 6

3.1 Comparisons of Resource Utilization in Tanakamaru's and Proposed Designs 60

4.1 Simulations Setup 88

Abbreviations

ACG	Asymptotic Coding Gain
ALUT	Adaptive Look-up Table
ALM	Adaptive Logic Module
BER	Bit Error Rate
BM	Branch Metric
CCI	Cell-to-cell Interference
ECC	Error Correction Code
FTL	Flash Translation Layer
FPGA	Filed Programmable Gate Arrays
LLR	Log-likelihood Ratio
HDD	Hard Disk Drive
ISPP	Incremental Stair Pulse Programming
LDPC	Low Density Parity Check
MSED	Minimum Squared Euclidean Distance
MSB	Most Significant Bit
MLC	Multi-level Cell
ONFI	Open NAND Flash Interface
RTN	Random Telegraph Noise
SM	State Metric
SSD	Solid State Drive
SILC	Stress-induced Leakage Currents
TCM	Trellis Coded Modulation

Notations

V_p	ISPP Verify Voltage
ΔV_{pp}	Program Step Voltage
V_e	Threshold Voltage of Erased Cells
$V_i^{(k)}$	Threshold Voltage of the kth Programmed Level After Ideal Programming
$\gamma^{(l)}$	Coupling Ratio
C_{bl}	Bit Line Capacitance
ΔV_{RTN}	Threshold Voltage Shift Caused by RTN
ΔV_{CCI}	Threshold Voltage Shift Caused by CCI
ΔV_{ret}	Threshold Voltage Shift Caused by Retention Noise
I_{pre}	Average Current Consumption in the Pre-charge Phase
n_{bl}	Number of Bit-lines Charged in Parallel
N_{TH}	Threshold Value for the Stripe Pattern Elimination Algorithm
d_{min}	Minimum Euclidean Distance
$(X_l \rightarrow X_{l+1})$	State Transition from X_l to X_{l+1}
E_s	Average Power
$\lambda(X_k, X_{k+1})$	Branch Metric
V_{max}	Threshold Voltage for the Highest Programming Level
$\widetilde{p}(x)$	Overall Distribution of Total Noises Induced Threshold Voltage Shifts
$\widetilde{p}^{(k)}(x)$	Probability Density Functions of Noise-induced Threshold Voltage Shifts
$L(b_i)$	LLR of the ith Bit Stored in One Memory Cell

Chapter 1

Introduction

Hard disk drives (HDDs) have been around for a half century as the primary type of external storage drive in the computing industry. An HDD uses magnetism to store data on a rotating platter. A read/write head floats above the spinning platter reading and writing data. A faster spin rate for the platter generally means faster HDD performance. Typical laptop drives today spin at either 5400 RPM (revolutions per minute) or 7200 RPM, though some server-based platters spin at up to 15,000 RPM. The major advantage of an HDD is that it is capable of storing lots of data cheaply. Currently, 1 terabyte (TB) of storage is not unusual for a laptop hard drive, and the density continues to increase. However, HDDs are becoming less attractive today because of its long read/write access latency, high energy consumption and fragility. Due to the physical limitations of the mechanical devices, the time used to locate the data on the spinning disk can be as much as a few milliseconds, resulting in a relatively long access latency (e.g., compared to solid state memories). The mechanical elements also induce high power consumption and risk of failure. Having several moving parts means the hard drive is prone to damage and read or write errors if it is moved too vigorously or too frequently without sufficient protection.

The disadvantages of HDDs are not a critical limitation for computer systems because there are alternative technologies for external storage. For instance, solid state drives (SSDs) offer a range of sizes and capacities to meet a range of computing needs. SDDs are attractive for overcoming the above limitations of HDDs. SSDs are data

storage devices using integrated circuit assemblies as memory to store data persistently. It is generally built over non-volatile semiconductor memories, such as NAND flash memories. Compared to HDDs, SSDs are more durable as they feature a non-mechanical design of NAND flash mounted on circuit boards. They are commonly shock resistant up to 1500g/0.5ms. SSDs have a second advantage with respect to data access. SSDs can have 100 times greater performance, almost instantaneous data access, quicker boot ups, faster file transfers, and an overall snappier computing experience than HDDs. Regarding power consumption, SSDs use significantly less power at peak load than hard drives, less than 2W versus 6W for an HDD. Their energy efficiency can lead to longer battery life in notebooks, less power strain on system, and a cooler computing environment.

Regarding the storage media of SSDs, the NAND flash memory represents a large proportion and has gradually become dominant due to increasing capacity and decreasing per-bit cost. These factors help to relieve the concerns of many people that SSDs are substantially more costly than traditional drives per byte. Such advantages come from several fundamental revolutions in the semiconductor technologies. Firstly, the flash density has been driven year over year through aggressive silicon feature size scaling (shrinking the size of a transistor). Today the NAND flash technology has reached 1Xnm, where "1Xnm" denotes that the design rule of the NAND flash memory is in the range of 10 to 20 nm, from the 250 nm generation technology one decade ago. Moreover, the trend for NAND flash memory is anticipated to continue in the near future, and the storage density will continuously be increased.

The second technology revolution is the use of compact architectures in designing the NAND flash memories, in which an array of memory cells share a common word-line where all memory cells or a part of them are programmed or read simultaneously. On the other hand, up to 32 or 64 memory cells in series are connected to share a bit-line and an on-chip page buffer that hold the data being programmed or fetched. Using this architecture, the NAND flash memories achieve highly efficient storage, and the silicon areas can be greatly saved.

Thirdly, multi-level cell (MLC) NAND flash, which uses multiple levels per cell to allow more bits to be stored within the same number of transistors, is being used in designs to increase the flash storage capacity. For MLC NAND flash, the storage efficiency increases linearly with the number of bits programmed to each cell. Compared with single level cell (SLC) NAND flash, the 3-bit per cell flash memory, also known as the triple-level cell (TLC), and 4-bit per cell NAND flash can increase the storage capacity by nearly 3 and 4 times, respectively. Aside from the low cost and high capacity, the NAND flash memory is also attractive for SSDs because it adopts some excellent technologies to improve the data programming speed, for example, the Double Data Rate (DDR) NAND, and multiple logical unit numbers (LUN) operation.

It has been predicted that the worldwide data traffic will increase to 26.2 million TBs in 2015, which means data centres should upgrade their capacity to deliver up to 15 times their current capacity. Considering the benefits provided by NAND flash based SSDs, and the increasingly data traffic, it is expected that the SSDs will eventually replace HDDs in most of the applications in the near future.

1.1 Motivation

Despite its attractive features mentioned above, the challenges for using NAND flash memory are considerable as well. One of the shortcomings that has attracted the attention of researchers for years is the data reliability problem [19]. In the simplest terms, the data stored in NAND flash is represented by electrical charges that are stored in each cell, which approaches a certain programming level of the MLC NAND. When reading data from the memory, the voltage in the memory cell are sensed and compared to the programming levels to decide the original bits. Due to the noise and interference occurring in the NAND flash memory, the bits read from each cell are not usually consistent with the original information programmed but are affected by an output error probability.

By studying the new technologies used in flash memories, researchers first discovered that the memory scaling degrades the reliability as it introduces larger bit-line capac-

itance causing more interference during programming, more specifically, cell-to-cell interference. Cell-to-cell interference rises quickly as the bit-line capacitance increases, and has been well recognized as the major noise source in NAND flash memory. Secondly, the MLC technology increases more errors than SLC and reduces the reliability as well. Apparently, adding more bits to each cell makes it more difficult to distinguish between levels, lowering both reliability and performance. The reading circuits have to detect the threshold voltage of the memory cell and determine the information residing in it. Indeed, determining whether a container is either full or empty (SLC) is much simpler than determining whether it is one quarter full, one half full, three-quarters full, or entirely full (MLC).

Another side effect of storing more bits per cell is an increase in the rate at which the NAND cells semiconductor degrades. The level of a NAND cell is determined by the number of electrons present on the floating gate. The oxide layers that trap electrons on the floating gate wear out with every program and erase operation. This means the reliability reduces and more errors occur as the increase of program-erase (P/E) cycles. When the memory cells wear out, electrons become trapped, which affects the overall electrical properties of the cell and consequently subsequent program and erase operations. With the oxide weakened, charges sometimes leak from the floating gate, making the detection of programming level even difficult in the reading circuit. Although this is not a huge problem with SLC NAND because there are only two levels to distinguish between, it can be a huge problem for a TLC NAND because there are eight levels to distinguish and very little room for differentiation -– just a few electrons can make the difference between one state or another. The raw bit error rate (RBER) can be measured as an useful reliability metric for storage systems [45]. It has been experimentally shown that the RBER for 70xnm MLC can be as high as 10^{-2} under the condition that the P/E cycling number is greater than 200,000. The condition could be even worse in the enterprise applications where flash is generally written at a much higher rate than in the client space.

All the NAND flash data errors caused by the physical factors mentioned above can be grouped into two classes: program disturb and data retention errors. The first class errors happen in the programming process while the second one in the data retention process. These errors have caused serious data reliability problems for storage systems based on NAND flash. If not dealing with them properly, there will be a large amount of incorrect data discovered after retrieving the information from the storage systems.

One way to reduce the errors is the use of write pattern processing technology. In view of the overall situation, the data pattern are random before being programmed to the memory array. The key idea of write pattern processing is to change the randomness of the data and intentionally distribute the data to some levels of MLC flash that have higher reliability. The technology mainly aims to reduce the errors occurring in the data retention process where the four levels of MLC flash have different reliabilities. Some researchers have been trying to apply this idea to the NAND flash memory controller to explore its capability in improving the reliability. The experimental results have shown that NAND reliability can be improved, and data retention errors can be reduced with data pattern processing. However, the scheme also brings a cost to the flash controller implementation because it consumes considerable hardware resources and increases complexity. Since the flash controller has a heavy load already in current practice because of several required components, such as the bad blocks management, it is very important to design a write pattern processing algorithm that is implementation-oriented and consumes as little hardware resources as possible. On the other hand, the programming of data sequences will result in increase of current, and in this regard, it has been shown that the energy consumption differs significantly depending on the types of the sequences. As the written pattern processing will modify the data sequence, it is also desirable for the algorithm to reduce the power consumption of flash programming. In designing the pattern processing algorithm, this energy problem should also be considered as well as the hardware complexity.

Another efficient way to mitigate both program disturb and data retention errors is the use of error correction codes (ECC). NAND relies heavily on robust ECC to compensate

Table 1.1 NAND Characteristics

	SLC	MLC	TLC
Bits per Cell	one	two	three
Cost per Capability	highest	medium	lowest
Performance	best	medium	worst
Reliability	best	medium	worst
ECC Complexity	highest	medium	lowest

for its inherent weaknesses and has become an indispensable part of NAND flash memory systems. In the era of SLC, the simple Hamming code is used frequently although it can correct a limited number of errors only. For MLCs equipped in most NAND applications today, the Reed-Solomon (RS) codes and Bose-Chaudhuri-Hocquenghem (BCH) codes are being used to achieve better performance. All these ECCs are based on hard decisions making their error correction capability limited in general. Due to continuous advances in technology scaling and the fact that more bits will be added to each memory cell, data reliability will get even worse in the future solid state memories. Therefore, more advanced error correction codes based on soft decisions, such as turbo codes or low density parity check (LDPC) codes will be the popular ECC algorithms used for future NAND flash systems. The advanced error correction schemes can usually provide higher error correction ability but result in higher software or hardware design complexity. Table 1.1 illustrates the ECC requirements of NAND flash in different densities.

Among several soft-decision ECC algorithms, the LDPC codes are very promising because of their capacity-approaching ability in the additive white Gaussian noise (AWGN) channel. Additionally, the codes can be implemented with parallel decoding schemes to achieve high throughput. All these excellent features have made the LDPC-NAND a hot topic in the area of flash coding, and researchers are interested in removing any obstacles that may prohibit the application of LDPC coding. Generally speaking, there are three main challenges to adopting LDPC codes in SSDs, which includes designing LDPC codes of good performance, exploring approaches to address the LDPC decoder input initialization problem, and minimizing read latency induced by LDPC decoding [77]. Firstly, the construction of LDPC codes is quite tricky since different

structures for the same size LDPC may lead to very different performance. The good candidates for error correction of NAND flash memory should achieve a low error floor and fast convergence speed. Meanwhile, for additional improvement in error-correcting performance, it would be a plus for the codes to contain the redundant parity checks. Those prerequisites make the construction of LDPC codes for NAND flash channel more complicated than the similar case for digital communication channels.

Secondly, the NAND flash memories provide hard decisions only due to the intrinsic characteristics of the sensing circuits that decide the original bits during reading process. While the LDPC decoder requires to be initialized with soft inputs, more efficient ways have to be developed to get the required initialization values [30]. This has actually become a critical issue recently for the LDPC-NAND error correction applications. On the one hand, the flash channel is not easy to be precisely modelled. Simple Gaussian models are usually adopted for flash coding research but the channel noise distribution is actually far different from the Gaussian function. On the other hand, non-uniform quantization can usually achieve more fine-grained soft-decisions than traditional uniform quantization, however, it is difficult to find a good quantization scheme in the unknown flash channel.

The third challenge is the read latency caused by the LDPC decoder circuitry [76]. To obtain the soft decisions for LDPC codes, multiple-precision sensing should be used but it results in significant data access latency overhead. The read latency dramatically limits the throughput of storage systems and becomes critical today as the data speed of storage interfaces is getting faster and faster. For example, the Universal Serial Bus (USB) 3.0, a commonly used interface for storage systems today, is over ten times faster than the USB 2.0 used in the past.

This work is motivated by the problems discussed above. We wish to investigate NAND flash channel, the flash error correction coding, and the flash memory controller from the physical perspective. We propose and investigate efficient solutions to those problems. The next section describes the goals of this thesis in detail.

1.2 Goals

All throughout this thesis, the main intention is to improve the data reliability of NAND flash memories. To this end, based on the discussions above, we shall focus the research work on two topics: the data pattern processing and error correction. The problems related to the topics will be investigated from the intrinsics of NAND flash memories, and we aim to propose solutions based on the behaviours of the memory channel. Hardware complexity will be considered as well since both the processing unit and ECC module are implemented in flash controller in practice. The research goals set up for the doctoral projects are listed below.

1. The first goal is to explore the physical connections between data patterns and the flash read bit errors. The error probabilities based on different data patterns shall be quantitatively analysed through either software or hardware simulation. Based on discovered results, it is expected to propose a hardware-oriented write pattern processing algorithm to improve the data reliability of MLC flash memories. Ideally, the proposed algorithm shall exhibit better performance and lower complexity over the existing work in this direction. It is also preferable that the proposed algorithm is designed without facing too many implementation issues in the current practice.

2. In addition to the write pattern processing, this thesis also targets error correction in flash memories. With respect to ECCs, the second research goal is set in the dissertation to discover ways of extracting the soft information from the hard decisions output of MLC flash memory. The performance of the classic multiple reads method shall be evaluated by simulating a number of different reads over MLC flash memories, and the disadvantages can be pointed out quantitatively. Based on the experimental results, and the application background of LDPC coding, it is expected to propose an error correction scheme that issues a novel soft extraction algorithm and employs LDPC codes. The presented scheme shall exhibit better error correction performance than the existing flash LDPC coding

mechanisms. In the meantime, it is also expected that the soft extraction methods shall be easy to implement and not induce much overhead in the sensing circuits.

3. This thesis aims to find an efficient way to quantize the sensing outputs for the memory reads. Both the uniform and non-uniform sensing quantization schemes will be investigated and simulated using the flash memory access model. The read latency and complexity for different quantizations are compared, and the critical factor influencing the performance shall be figured out. In the end, it is expected to discover an optimized sensing quantization scheme that results in lower read latency and can be applied in flash memory with the proposed ECC scheme to achieve good performance.

4. The final goal is to characterize the flash memory channel and build a channel model that fits the real situation as much as possible. Various existing channel models are based on different kinds of approximation, while this thesis is trying to propose a model without using too much approximation. All channel sources of noise, distortion and interference shall be taken into account when calculating the channel distribution, and it is expected to present accurate probability density functions for the channel noise and final threshold voltage of each programming level. Based on the results, future research may find out better solutions for extracting soft information. Therefore it is of major importance for the future research in flash coding.

1.3 Thesis Contribution

Several approaches have been used in this work to meet the research goals. The first way is to investigate the storage systems and digital communication systems and compare both from the intrinsic view. Keeping the differences in mind, some techniques used in communication systems can be applied to flash memory and achieve remarkable performance, provided only that the techniques be modified properly to fit in the application. To mathematically characterise the channel distribution, another important approach

using the characteristic function of random variables is used in this thesis. Based on the approaches and experimental results, this dissertation creates a channel model for NAND flash memory and develops efficient fault tolerant solutions to improve the reliability of flash-based data storage. The main contributions of this thesis are summarized as follows.

1. This dissertation studied the practical low-complexity schemes that deal with data pattern processing. A write pattern formatting algorithm (WPFA) is presented to eliminate column stripe patterns and reduce the proportion of flash cells on high threshold-voltage levels. The WPFA improves data reliability and relieves the power pressure of flash memory effectively. With the proposed scheme, SSD systems consume low energy and avoid suffering huge current spike. Compared to the current design practice that employs asymmetric coding and stripe elimination algorithm (SPEA), the proposed algorithm consumes less overhead of NAND cell area and exhibits better performance on power consumption. Associated hardware design issues are considered with the aim of reducing complexity and utilizing less logic resources. The advantages of the proposed scheme have been evaluated by various metrics using both computer simulations and FPGA implementation. Due to all these benefits, the proposed algorithm will be an attractive solution for practical low-complexity data pattern processing schemes of SSDs.

2. As for the flash controller, the ECC module comes after the data pattern processing scheme. Following the WPFA algorithm, this dissertation designed a novel fault tolerant solution for flash memory that concatenates trellis coded modulation (TCM) with an outer LDPC code. The concatenated TCM-LDPC fulfilled the error correction capability of LDPC coding to achieve reliable storage in multi-level flash. Moreover, the error correction performance is further improved since TCM can decrease the raw error rate of MLC and hence relieve the burden of outer LDPC code. Compared to the flash coding system that provides hard-decisions and employs Bose-Chaudhuri-Hocquenghem (BCH) codes only, results show remarkable BER performance improvements from the system equipped with

LDPC codes and pragmatic TCM. This dissertation has also derived mathematical formulations to quantitatively analyse the asymmetric coding gain achieved in flash channels.

3. Another contribution that comes with the TCM-LDPC is the way it used to get the soft values. Thanks to the Bahl, Cocke, Jelinek, and Raviv (BCJR) algorithm, the TCM decoder can extract soft information from the hard-decision outputs of flash read circuits. Such approach will not influence the sensing circuits, and thus introduces less read latency than the classic multiple reads method. Given the soft initialization values, the decoding of LDPC codes is feasible; as a result, the BCJR and TCM will be a useful alternative in practice to extract flash memory soft information.

4. Based on the study of BCJR for flash LDPC-TCM, this dissertation continues to explore the ways of extracting soft information by investigating the flash memory channel. The threshold voltage distribution after ideal programming in flash memory cells is usually distorted by a combination of the random telegraph noise (RTN), cell-to-cell Interference (CCI), and the retention process. This research aims to characterize these various sources of distortion occurring in multi-level cell (MLC) flash memories. The first contribution of this work is a mathematical description of the overall distribution for the total flash channel distortion. The final threshold voltage distribution for each symbol stored in MLC flash is also formulated by using characteristic functions for the various sources of noise. With the final cell threshold-voltage distribution, the soft information for each bit in the cell can be calculated and applied for decoding soft decision-based error correction codes. The results presented here are novel in a theoretical way as the first time the flash channel has been characterized by incorporating the distributions of all sources of interference and noise. Using the 4-level MLC flash as a case study, the derived threshold voltage distributions and the results of the theoretical analysis have been validated through Monte Carlo simulations of the flash channel.

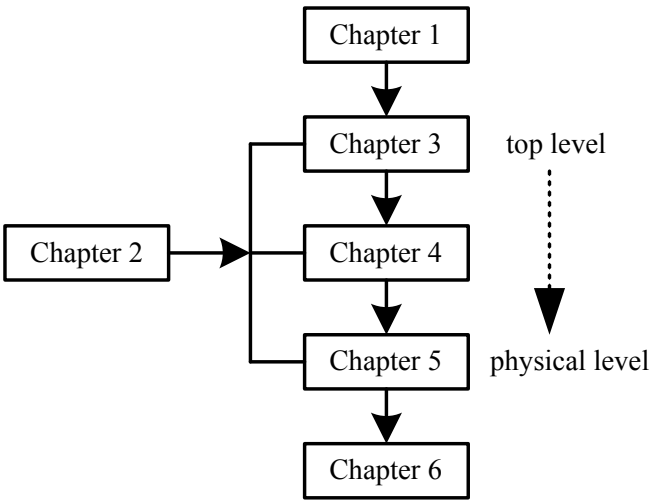


Fig. 1.1 Overview of the thesis organization.

1.4 Thesis Outline

The overall structure of this dissertation is shown in Figure 1.1. Chapter 2 provides a comprehensive background for the research topics investigated in the following chapters, and elaborates on the nature of the reliability issues based on a literature review. Chapter 3, 4 and 5 study the research problem from different angles and present the corresponding solutions or theoretical discoveries for each topic. The basic approach is to explore the problem from the top level to the physical level of memory systems, pushing the research deeper into the essential part of NAND flash. Chapter 6 briefly concludes the dissertation.

The rest of the dissertation is organized as follows. Chapter 2 introduces the basics of NAND flash memories. The reliability problem is brought out from the structure and application background of flash memory and explained clearly based on the open literature.

In chapter 3, we classify the errors in NAND flash memory into data retention errors and program disturb errors, and focus on the data retention problem by assuming that the program disturb has been alleviated in the sub-40 nm flash memory design. Furthermore, we examine retention issue from the top level of memory system, i.e., the data pattern processing module which is designed before the ECC and programming circuit, and discover that different data pattern results in different number of errors in the

retention process. Based on the measurement results and other scholars' discoveries in this direction, we present a novel write pattern formatting algorithm (WPFA) for MLC flash memory to deal with data pattern processing. Comprehensive simulations and implementation circuitry are also presented in this chapter. Additionally, the performance and hardware complexity of WPFA are evaluated and compared with the existing work.

Chapter 4 shifts the study of the reliability problem to the lower level of flash memory, the error correction coding, which deals with all kinds of errors. The hard-decision based codes are popular in the current design practice. We investigate the way of using the soft-decision based flash coding schemes that are capable to provide better error correction performance. In chapter 4, we propose a combined TCM-LDPC coding scheme for the error correction of MLC flash memories. The theoretical analysis, computer simulations and performance comparisons are presented in this chapter as well. For the further works of TCM-LDPC, we also discuss the approaches for soft-decision extraction and sensing quantization.

Chapter 5 further investigate the application of advanced ECC from the physical level and model the distribution distortion of flash memory channel. The noise and interference are characterized and a mathematical description of all the sources of distortion is presented in this chapter. Using the formularized expressions of noise distributions, we further derive the final threshold-voltage distribution for each level of MLC. The results are used to instruct the use of calculating soft decisions and applying soft-decision based error correction codes.

Chapter 6 provides conclusions of the key results and insights presented in this dissertation. Suggestions on how to use quantization, LDPC codes and other techniques for future NAND flash memories based on the proposed flash channel model are also discussed in this chapter.

Chapter 2

Background

The aim of this chapter is to provide descriptions of the state-of-the-art NAND flash technology. The NAND architecture and memory organization are reviewed first. Next, an overview of NAND data reliability issues is presented, devoting particular attention to the flash channel noise sources and error correction because they are necessary prerequisites for the following chapters.

2.1 Basics of NAND Flash Memory

As a type of non-volatile storage, the NAND flash memory is able to keep the data even when the power supply is switched off in the device. Due to this characteristic the NAND flash memories have been used in storage products including compact flash cards, solid state disk drives, flash memory dongles, and the image storage in digital videos and cameras. The first NAND flash memory was invented in 1989 by Toshiba, which surpassed previous types of non-volatile memory in terms of both capacity and cost. During 1990s, the technology advances in the portable devices that use a battery as power supply, including PDAs, mobile devices and digital storage, dramatically expanded the market share and the popularity of the NAND flash memories.

2.1.1 NAND Device Organization

Depending on the manufacturer, there is some variation in the implementation of NAND flash devices, resulting in two interface standards: Toggle NAND and Open NAND Flash Interface (ONFI) NAND. Samsung and Toshiba make Toggle NAND, whereas virtually everyone else makes ONFI NAND, including Intel, Micron, etc. These two standards support different bus operations but have similar device organization. Generally speaking, a NAND flash device is composed of one or more targets which is accessed through an enable pad equipped in the memory chip. Die is the unit of a NAND flash memory target, and one target is usually composed of one or more dies. The ONFI specification also uses the name of logical unit (LUN) to represent the flash die, implicating that it is the minimum unit that can execute commands and transfer data independently. Each flash die is partitioned into multiple planes and a plane is further divided into blocks. For the NAND flash device, block is an important which is the unit for erase and consists of a number of pages for programming.

From the perspective of circuitry, the NAND flash block is a memory array that consists of N_b bit lines (BLs) and N_w word lines (WLs), as shown in Figure 2.1. The memory cell string is the minimum array element which has 16 to 64 flash cells in typical organization. Each block of the NAND flash memory contains a number of pages, for example, the Micron 2Gb asynchronous SLC device is organized as 2048 blocks, with 64 pages per block. As for the erase operation, all the memory cells residing in the same block shall be executed simultaneously, however, the read and program operations are executed in the unit of page all the cells residing in the target page are written and sensed at the same time. Each page has a data area and a spare area. The data area is used to store the user data, and the spare area, also called the out-of-band (OOB) area, is used to store the non-information data, such as the redundant bits for error correction code or the flag bits for flash management algorithms. The bit-lines are shared by all memory blocks in the devices, using an on-chip page buffers to hold the data for program and read. The recent NAND Flash memories adopt either an interleaving bit-line architecture or all bit-line (ABL) architecture. For the interleaving architecture, even and odd bit lines are

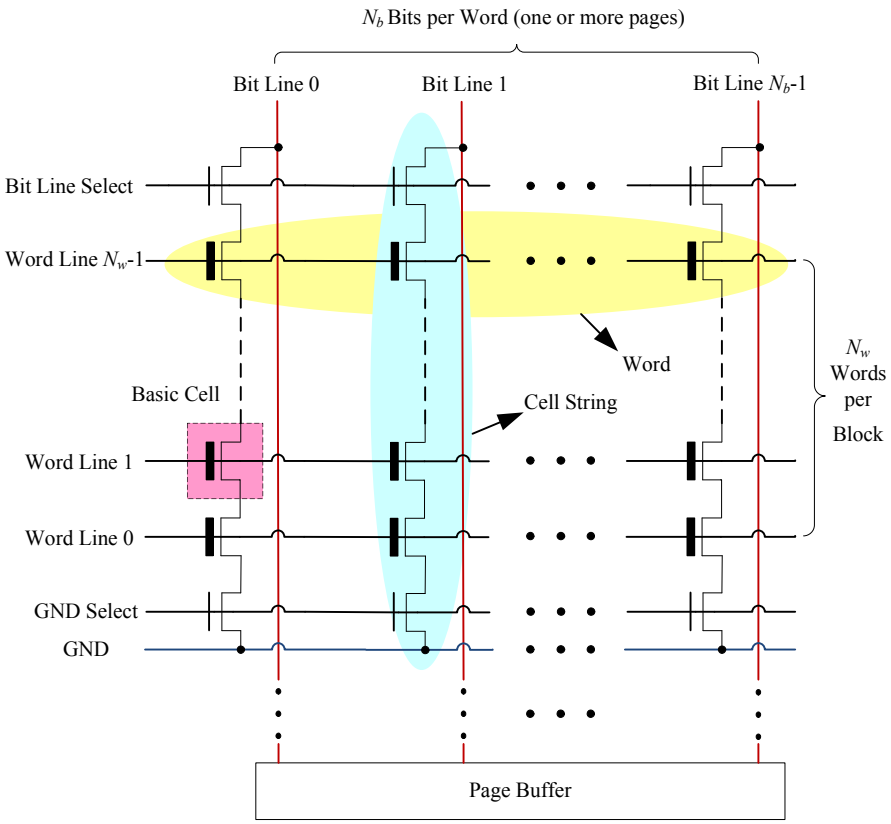


Fig. 2.1 NAND flash memory array structure

not accessed at the same time but interleaved along the word line. This feature allow each pair of even and odd bit lines to share the sense amplifiers and buffers, resulting in less silicon cost of peripheral circuits. The situation is different for the ABL architecture, where all the bit lines are accessed at the same time. The ABL NAND flash aims to reduce the errors caused by cell-to-cell interference (as elaborated later in Section 2.2.2), with a tradeoff to the peripheral-circuit silicon cost. Moreover, a current-sensing scheme is required in the ABL structure, which is thought to be more complicated than the voltage-sensing scheme used in the interleaving architecture.

2.1.2 NAND Cell Architecture and Basic Operations

NAND flash storage devices save data in a matrix of memory cells. Basically, the memory cell has a built-in floating-gate MOSFET, which is also known as FGMOS. The transistor in the flash cell is different from the standard four-terminal MOSFET with the special ability to keep the electrical charges for a long period (up to 10 years) based

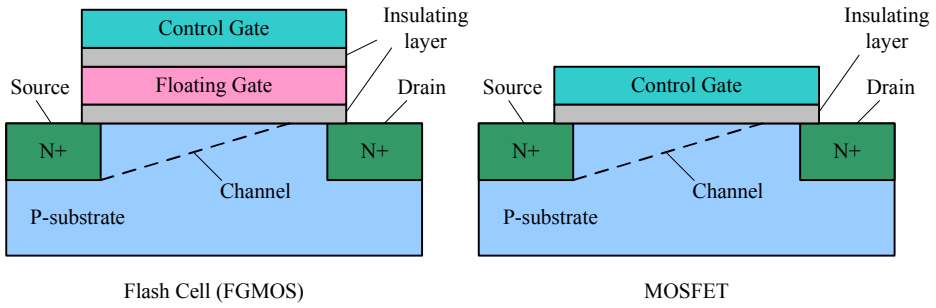


Fig. 2.2 Flash memory cell vs. MOSFET.

on the technology today, even in the absence of a power input. Physically, the FGMOS is originated from a standard MOS transistor but manufactured in a different way. It has two gates: control gate and the floating gate, which are isolated electrically from the body using an insulator like SiO_2. The control gate is placed above the floating gate, which decide the threshold voltage of the flash cell by modifying the amount of charges in the floating gate. There is no resistive connections between the control inputs and the floating gate due to the fact that it is fully surrounded by the isolators.

Each NAND flash memory cell is configured to store the prepared information bits by injecting a certain amount of charge into the floating gate. Usually Fowler-Nordheim tunneling and hot-carrier injection mechanisms are used to modify the amount of charge stored in the Floating Gate. The voltage level achieved by the amount of charge is commonly called the threshold voltage of the NAND flash memory cell. The number of electrons on the floating gate affects the threshold voltage, and the effect is measured to determine the bits saved in the cell. Single-level cell (SLC) NAND flash has the ability to store only one bit value per cell which therefore requires threshold voltage level for each cell. The bit value is interpreted as either a “0” or a “1”. However, from the perspective of developers, one of the major motivatives for NAND flash memory development was to achieve maximum memory capacity and reduce the cost per bit. Only after solving these two issues will the NAND flash devices be able to compete with magnetic storage devices in the current market. Apparently, the SLC cannot fulfil the requirements, therefore, the multi-level cell (MLC) flash was invented aiming to store more than a single bit of information. As suggested by its name, more than one bit

values can be represented in an MLC flash cell, and the threshold voltage is used in the same way as SLC to manipulate the multiple levels for the memory cell.

The basic operations on NAND flash include erasing, programming and reading, all of which involve changes of cell threshold voltages. Unlike traditional hard disks, a flash page has a write-once constraint because it can not be updated (or overwritten) directly unless the page's residing block is erased first. An erase operation involves removing the charges before programming a memory cell, which sets the memory cell threshold voltage to the lowest level. Comparing to the writing operation, the erasing is completed in a way called tunnel release. Due to the inconsistency for the erase operations, the threshold voltage of erased cells, denoted by V_e, tends to reach a wide normal distribution, which can be approximately modelled as

$$V_e = V_i^{(0)} \sim p_e(x) = \frac{1}{\sigma_e\sqrt{2\pi}} e^{-(x-\mu_e)^2/2\sigma_e^2} \tag{2.1}$$

where μ_e and σ_e are the mean and standard deviation of the erased state. Here we also use $V_i^{(0)}$ to denote the threshold voltage of erased cells for the consistence of the following discussions on NAND flash programming since the erased state is assumed as the programming level 0.

For programming, a scheme called incremental stair pulse programming (ISPP), or staircase algorithm, is utilized to reach a tight threshold voltage bound for the representation of each symbol (or level) of the MLC. The ISPP scheme recursively programs the memory cells on the same word-line through a program-and-verify method, as shown in Figure 2.3. Let V_p denote the verify voltage of the target programmed level, and ΔV_{pp} the program step voltage. Ideally, the ISPP results in an uniform distribution over the interval $[V_p, V_p + \Delta V_{pp}]$ which has width of ΔV_{pp}. Suppose V_p and $V_p + \Delta V_{pp}$ for the k-th programmed level are denoted as $V_l^{(k)}$ and $V_r^{(k)}$, respectively. Considering the K-level MLC NAND flash for discussion, the threshold voltage of the kth programmed level after ideal programming, denoted by $V_i^{(k)}, (1 \leq k \leq K-1)$, can be modelled as

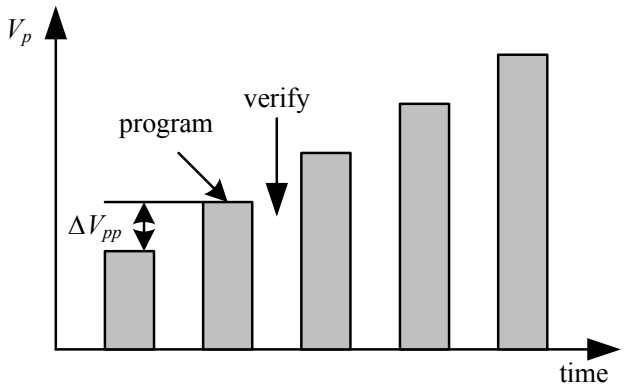


Fig. 2.3 Control-gate voltage in program operation.

$$V_i^{(k)} \sim p_p^{(k)}(x) = \begin{cases} \frac{1}{\Delta V_{pp}}, & V_l^{(k)} \leq x \leq V_r^{(k)} \\ 0, & otherwise \end{cases} \tag{2.2}$$

To summarize, the initial cell threshold voltages of K-level MLC flash memory, $V_i^{(k)}$ for $0 \leq k \leq K-1$) have the following distributions

$$V_i^{(k)} \sim p_i^{(k)}(x) = \begin{cases} p_e(x), & k = 0 \\ p_p^{(k)}(x), & 1 \leq k \leq K-1 \end{cases} \tag{2.3}$$

Note that the distributions in (2.3) are the ideal results for the programming of NAND flash memories which, however, can be significantly distorted in practice by a combination of noise and interference detailed in Section 2.2.2. Equipped with the ISPP scheme, the K-level MLC NAND flash memory stores data by moving the threshold voltage of each cell from the erase state to one of K predefined states that corresponds to K non-overlapping windows. Ideally, the threshold voltages for cells written with the same n-bit information shall fall into the same window, but the final threshold voltage distribution could be distinct. The non-overlap space between the adjacent windows is commonly called the distribution margin. For a K-level MLC NAND flash memory, we use $K-1$ predefined read reference voltages to discriminate the K possible states and make decisions on the final threshold voltage of each cell. These $K-1$ references are specified to the respective non-overlapping areas (i.e., distribution margins) of the

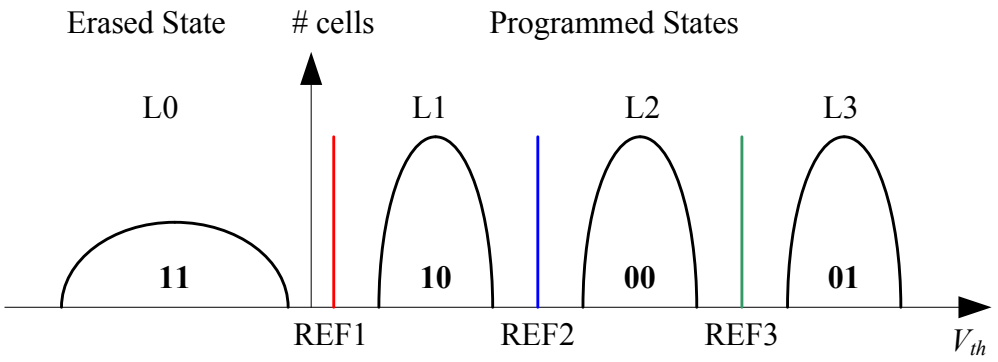


Fig. 2.4 Read reference voltages for 4-level MLC.

threshold voltage windows. Each threshold voltage window is determined by an upper and a lower bound read reference voltage.

During a read operation, the cell's threshold voltage is iteratively compared to predefined read reference voltages until the upper and lower bound read reference voltages are identified, thereby determining the stored bit value. The read reference voltages and distribution margin for 4-level MLC are illustrated in Figure 2.4. In this case we assume that the threshold voltages are not incurred with the flash channel noise, thus the distributions resemble the ideal programming modelled in (2.3). Figure 2.4 shows that the cells in a given programmed state (e.g., L1 which indicates a logical bit value of 10) have threshold voltages that fall into a distinct threshold voltage window (e.g., threshold voltage values between REF1 and REF2).

In past flash generations, the read reference voltage values were fixed at uniform intervals at design time. However, when the threshold voltage distributions are distorted by multiple sources of noise and interference, the distributions can shift causing distribution tails to enter the previously non-overlapping distribution margin regions, crossing the read reference voltage levels. As a result, a cell that stored one logical value can be misread as storing a different logical value, an event called a read error. This is even more serious for MLC flash memories which have narrower distribution margin than the SLC memories. One mechanism to combat such errors is called read retry implemented in some flash memories below 30 nm [82]. Read retry allows the

read reference voltages to be dynamically adjusted to track changes in distributions. The basic idea is to retry the read with the adjusted reference levels such that read errors are decreased or even eliminated. Another way is to use the non-uniform reference voltages in the ECC decoder [16]. The non-uniform strategy senses the governing overlap region with higher precision while leaving the other regions with lower sensing precision or even no sensing. It reduces the sensing latency as well as read errors. The sensing quantization is one of the important topics in the design of flash ECC schemes.

2.1.3 Planar and 3D NAND Flash Memory

Traditional planar NAND flash memory is classified as SLC or MLC flash memory. Although MLC flash memory has worse read/write performance, reliability, and endurance than SLC flash memory, it has gained the best market share because of its advantages in cost and memory density. As predicted by the industry, flash memory is hard to scale down below the 20 nm technology because too few electrons can be kept in the floating gate of each cell, and each cell can not have a good margin to verify the voltage level. In contrast, 3D NAND flash memory (also referred to as 3D flash memory) presents a new trend in chip designs. Instead of simply shrinking the fabrication process and flash memory cell size, a 3D flash memory chip applies single etching to punch through multiple layers so as to create some shared hardware structure and reduce the bit cost.

Researchers and manufacturers are attempting to develop 3D flash memory because 3D flash memory is considered to be a promising way to continue to scale down the die size and scale up the chip capacity. In the past few years, different types of manufacturing technologies for 3D flash memory have been proposed. Bit-Cost Scalable (BiCS) flash memory was first designed to share the process of bit line structures to reduce the design cost. After BiCS, various types of structures were also proposed. Well known examples are Pipe-shaped Bit-Cost Scalable (P-BiCS), Terabit Cell Array Transistor (TCAT), Vertical-Stacked-Array-Transistor (VSAT), Dual Control Gate with Surrounding Floating Gate (DC-SF), and Vertical Gate (VG). Among them, the VG structure was suggested as a good candidate design for 3D flash memory in recent studies

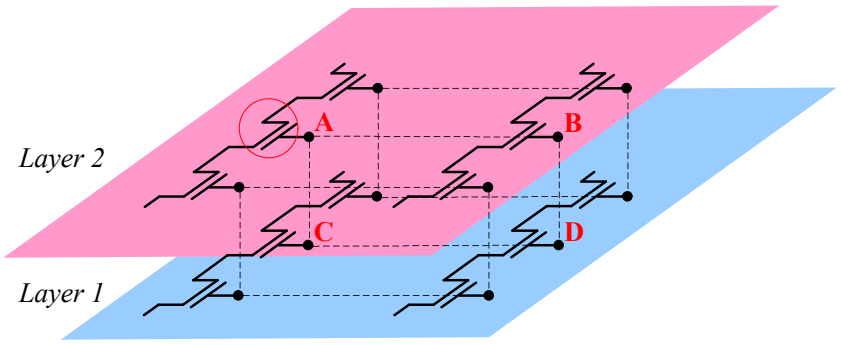


Fig. 2.5 A Typical structure of VG flash memory.

because of its scalability made possible by its small minimal cell size and well-controlled interface. Figure 2.5 shows the structure of a two-layer VG flash memory. All of the control gates of the cells from different layers but on the same vertical plane share the same word line. For example, the control gates of cells A, B, C, D are connected to the same word line because they are on the same vertical plane shown as a dotted rectangle in Figure 2.5. When the number of layers increases, the structure can be easily extended by connecting the control gates of the cells on the same vertical plane to the same word line. These layers reside in the same flash memory chip thus increase the capacity of a single chip significantly while keeping the similar die size.

In the past twenty years, the planar NAND technology has been developing to approach its scaling limits, which poses grim challenges for the non-volatile memory industry. In this circumstance the 3D NAND flash memory attempts to impress the industry by keeping flash memory solutions consistent with Moore's Law, the trajectory for the continued performance improvement and cost cutbacks, promoting more widespread application of NAND flash storage. Many big storage companies including Intel, Samsung, Micron, and Toshiba are developing their own product lines individually for the 3D NAND based storage solutions and expect to market those products in the impending future.

2.2 Data Reliability Issues

The continuing demand for NAND flash storage on better performance and higher memory capability has forced the manufacturers to probe to the limitation of the present technology and to develop new solutions, from both the physical and structural views [42].

For this technological evolution, the storage reliability is one of the critical obstacles because the customers wish to be assured not only when a new product is initiated but also demonstrated for the whole life cycle. Particularly, the manufacturers must guarantee the minimum number of program and erase operations and the ability to keep the saved data unaltered for a long enough period.

In this section, the prominent factors affecting the reliability of the traditional FGMOS based NAND flash memories will be addressed. We discuss the issues at a general level first and then delve into the detailed mathematics for each type of noise. The topics of NAND flash write endurance and wear levelling are also introduced in this section.

2.2.1 Reliability at Physical Level

The notion of reliability for NAND flash memory refers mainly to the errors and disturbances caused before reading from the flash channel. We have discussed the read errors caused by the narrowing down of distribution margin, which is not considered here as one of the channel errors. Generally speaking, the errors occurring during the flash programming and data storage can be classified into two major types: the program disturbance errors and the data retention errors. Essentially these two reliability issues are caused by flash program/erase (P/E) cycling effects and bit-line capacitance effects.

The frequent program/erase cycling causes the charges trapped in the oxide and interfaces, which damages the tunnel oxide of the floating gate transistors, resulting in the the cell threshold voltages variation and shifts. Therefore, the noise margin of the flash memory chips is gradually downgraded as the P/E cycles increase. To the threshold voltage of a certain flash cell, the first kind of distortion developed over the P/E cycling is the electrons capture and emission incidents occurring in the charge trapping fields.

These incidents eventually develop the electronic burst noise, bringing threshold voltage fluctuation directly to flash memory. Meanwhile, the recuperation of interface trapping and detrapping also causes slow reduction to the cell threshold voltage, leading to the data retention issue.

There are two types of capacitances discovered in the flash memory channel: one capacitance resides in the floating gate transistor and the other is the inter bit-line capacitance, as shown in Figure 2.6. When a voltage is added to the control gate, the floating gate starts to accumulate electrons and drives the cell threshold voltage to a certain level with the help of the first kind of capacitance. This includes the capacitance between control gate and floating gate (C_{ONO}), the capacitance between floating gate and source terminal (C_S), the capacitance between floating gate and drain terminal (C_D), and the capacitance between floating gate and the P-substrate body (C_B), as shown in Figure 2.6(a). Without these capacitances it is impossible to program the NAND flash memories.

The inter bit-line capacitances cause notable threshold voltage shift to the programmed cells, thereby further degrading reliability. These programmed cells are called victim cells because their threshold voltages will be raised as neighbouring cells are programmed. The neighbouring cells are called interfering cells. In Figure 2.6, cell 1 is the victim cell supposing it was programmed ahead of the five neighbouring cells. The two neighbouring cells on the same word line are assumed to cause equal effects because they have the same coupling capacitance to the victim cell, which applies as well to the two cells in the diagonal positions.

2.2.2 Noises in NAND Flash Memory

As discussed earlier, the threshold voltage distribution after ideal programming in NAND flash memory cells is usually distorted by a combination of noise and interference, which include the random telegraph noise (RTN), cell-to-cell interference (CCI), and the retention process. In this section, a mathematical analysis of the noise and interference is

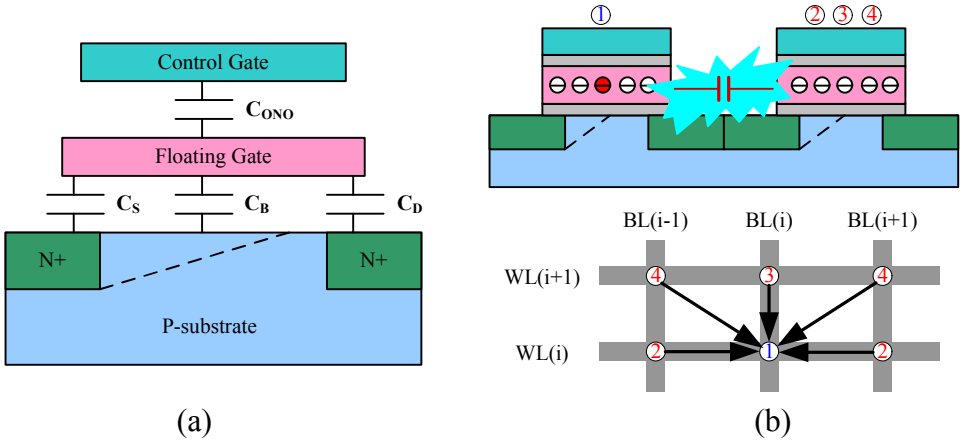


Fig. 2.6 Capacitors involved in the NAND flash memory channel.

presented, based on the open literature. We also present simulation results for threshold voltage distortion caused by the multiple sources of channel noise.

Random telegraph Noise

The random telegraph noise (RTN) phenomenon, also named in old fashion as burst noise or popcorn noise, was firstly observed in electronic devices including bipolar transistors and junction field effect transistors during the 1950s. Evidence for RTN in MOSFET devices has been reported since the mid-1980s [11]. The basic concept of RTN is based on a mechanism where the electrons capturing and emission by interfacial traps changes with time constants τ_c and τ_e, leading to the fluctuations of FGMOS drain current and threshold voltage. The effects caused by RTN depend on the flash memory P/E cycling number since the electrons capture and emission are developed over the cycling process [42]. Denote the threshold voltage shifts caused by RTN by ΔV_{RTN}. The probability density function of ΔV_{RTN} can be modeled as a symmetric exponential function

$$\Delta V_{RTN} \sim p_r(x) = \frac{1}{2\lambda_r} e^{-|x|/\lambda_r} \tag{2.4}$$

where the parameter λ_r scales with the P/E cycling number N in an approximate power-law fashion, i.e., λ_r is approximately proportional to N^{α}.

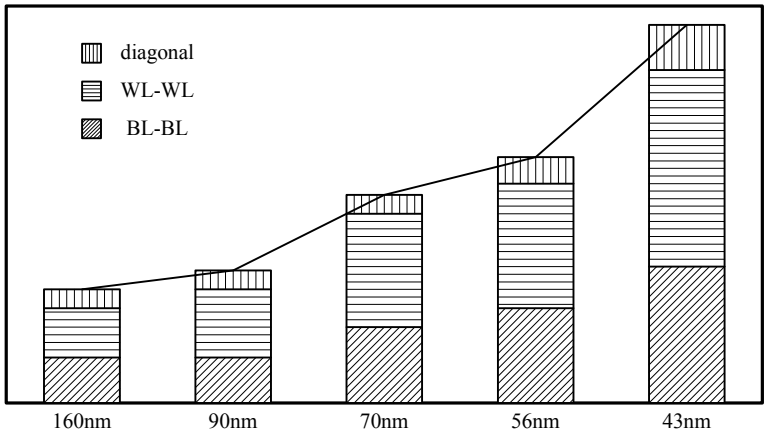


Fig. 2.7 Cell to cell coupling trend.

Cell-to-Cell Interference

In Section 2.2.1, we mentioned that the inter bit-line capacitances cause interference in the flash memory channel and distortion of cell threshold voltages. This interference is commonly known as the cell-to-cell interference (CCI) and has been recognized as the major noise source in NAND flash devices. As the NAND flash technology keeps scaling down, the neighbouring cells get closer and the inter bit-line capacitances between the neighbouring flash cells become more and more notable, resulting in progressively severe cell-to-cell interference and more serious distortion of threshold voltages. Conversely speaking, the cell-to-cell interference is also a crucial barrier that holds back the future NAND flash technology improvement. Figure 2.7 illustrates the trend of inter bit-line capacitances versus technology scaling. Recall that the capacitances include three parts: When the feature size of NAND flash memories reaches 4Xnm, where "4Xnm" means that the design rule of the NAND flash device is restricted to the range of 40 to 50 nm, the cell-to-cell coupling increases dramatically compared to old generations. Therefore, it becomes very hard to achieve a tight distribution when using the staircase programming algorithms in these feature-sized devices, making the real threshold voltage distributions far different from the ideal programming.

The strength of cell-to-cell interference has been mathematically modelled by Lee et al. [33] in their work discussing the effects of floating gate interference. Threshold

voltage shift of a victim cell caused by cell-to-cell interference, denoted ΔV_{CCI}, can be estimated as

$$\Delta V_{CCI} = \sum_{l} \delta V^{(l)} \gamma^{(l)} \tag{2.5}$$

where $\delta V^{(l)}$ represents the threshold voltage shift of one interfering cell which is programmed after the victim cell, and $\gamma^{(l)}$ is the coupling ratio subject to the parasitic capacitance between the interfering cell and the victim cell. It is defined as

$$\gamma^{(l)} = \frac{C^{(l)}}{C_{total}} \tag{2.6}$$

where $C^{(l)}$ is the parasitic capacitance between the interfering cell and the victim cell and C_{total} is the total capacitance of the victim cell. Understandably, a memory cell would be interfered only by the adjacent cells which are flashed after the write operations to the victim cell. Due to the fact that the parasitic coupling capacitance decreases rapidly with the distance to the victim cell increases, we only take the cell-to-cell interference from the immediately neighbouring cells into account in the design practice.

The significance of CCI to a victim cell could be different depending on the NAND flash architecture. For the interleaving architecture, cells on the even and odd bit lines share one page buffer so they are programmed alternatively at different time. Since the even cells are programmed ahead of the odd cells in the same word line, they are faced with more cell-to-cell interference. With respect to one even cell, it sees interference by five neighbouring cells: two adjacent odd cells, two diagonal cells on the next word line, and one cell on the next word line and sharing the same bit line; all of which are programmed after the even cell. On the other hand, the odd cell suffers interference from three neighbouring cells only: two diagonal cells on the next word line, and one cell on the next word line and sharing the same bit line.

For the all bit-line architecture, all the bit lines in the same word line share one page buffer and are programmed at the same time. Therefore, all cells suffer the cell-to-cell interference from their neighbours on the next word line only, i.e., one victim cell is

interfered by three neighbouring cells. As a result, the all bit-line architecture experience generally less cell-to-cell interference than the interleaving bit-line architecture. Moreover, the all-bit-line structure is able to maintain a high-speed current sensing which improves the speed of the memory read and verification dramatically. Although the ABL structure consumes more silicon area than the interleaving architecture, this disadvantage is getting less noticeable due to the intensive technology scaling. Therefore, the all bit-line architecture is more attractive in the modern NAND flash memory design. In this thesis, we would consider mainly the ABL architecture, but we note that the ideas presented in this work are also applicative to the interleaving structure.

Due to the process variability, the coupling ratios $\gamma^{(l)}$ are assumed to follow a bounded Gaussian distribution [13] as

$$p_\gamma(x) = \begin{cases} \frac{c_\gamma}{\sigma_\gamma\sqrt{2\pi}} \cdot e^{-\frac{(x-\mu_\gamma)^2}{2\sigma_\gamma^2}}, & if\,|x-\mu_\gamma| \le \omega_\gamma \\ 0, & elsewhere \end{cases} \tag{2.7}$$

where μ_γ and σ_γ are the mean and standard deviation, ω_γ represents the bounded distribution region, and c_γ is chosen to ensure that the bounded Gaussian distribution integrates to one. Denote the coupling ratios in the three directions shown in Figure 2.6 as γ^x, γ^y, and γ^{xy} respectively. We set $\mu_{\gamma^x} : \mu_{\gamma^y} : \mu_{\gamma^{xy}} = 0.1 : 0.08 : 0.006$ following the literature, for instance [13, 15, 16, 51, 59]. Note that the coupling ratio μ_{γ^x} would not be considered in the ABL architecture as the coupling effects from the cells in the same word line have been eliminated.

Data Retention Error

The data retention errors are caused by the interface trap recovery and electron detrapping happening in the program/erase operations and the lifetime of NAND flash memory. For the processes of programming and erasing, a trap generation can be observed in the tunnel oxide between the transistor body and the floating gate, which essentially is caused by the strong electric field placed to the oxide. The electric field produces some holes as an effect to speed up the trap generation, of which the rate therefore depends

strongly on this applied filed. The total charges stocked into the oxide, on the other hand, is a function of the square root of the P/E cycles. Average time consumed among the further cycles and processing temperature also affect this phenomenon in the device. The phenomenon discussed is called trapping phenomenon. It causes the the threshold voltage shifts as well, similar to the other sources of noise aforementioned. Because of the trap generation the electrons might be captured and more and more negative charges are gathered in the oxide, which therefore change the cell threshold voltage.

In the case of thermal heating, some charges entrapped in the oxide can overcome the forces and leave away from the oxide itself when an activation power is high enough. As a result, a negative reduction of the cell threshold voltage happens. The total charge lost due to the detrapping is functional to the time consumed for the further program cycles and the number of P/E cycles.

Let ΔV_{ret} denote the threshold voltage shifts resulting from the data retention errors. It has been mathematically demonstrated that the interface trap recovery and electron detrapping processes roughly follow Poisson statistics [43], thus it is reasonable for us to model the ΔV_{ret} caused by interface trap recovery and detrapping as a normal distribution $\mathcal{N}(\mu_d, \sigma_d^2)$:

$$\Delta V_{ret} \sim \mathcal{N}(\mu_d, \sigma_d^2), \;\; i.e., \;\; \sim p_{ret}(x) = \frac{1}{\sqrt{2\pi}\sigma_d} e^{-\frac{(x-\mu_d)^2}{2\sigma_d^2}} \tag{2.8}$$

where both μ_d and σ_d^2 scale to N in an approximate power-law way, and scale to the retention time t in a logarithmic fashion. Additionally, the significance of μ_d and σ_d^2 is also proportional to the magnitude of the initial threshold voltage $V_i^{(k)}$. In other words, the higher the initial cell threshold voltage is, the more frequently the interface trap recovery and electron detrapping occur and the faster the threshold voltage diminishes [32].

2.2.3 Error Correction Codes for Flash Memories

Error correction coding is part of information theory that handles the wireless communication systems. In this theory, the information is considered as a physical quantity that can be saved, detected, measured, and moved from one place to another. An essential idea for the information theory is to quantitatively measure the information with the amount of uncertainty determined. The uncertainty, on the other hand, can be described mathematically using the probability theory.

The information is transmitted over a physical means named channel in a common way. Some examples of channels include twisted-pair wire, telephone and internet cables, optical fibres and so on. These channels are always used to move the information from place to place; however, the information can also be transmitted to and received from the same end at different times. For example, we can write the information to a computer disk and read the data back at any time in the future. The carrier specified for such application can be considered as channel as well. Hard disk drives, solid state disk drives, CD-ROMs, DVDs and flash memories are examples of these channels which are most relevant to this dissertation.

Before the transmission, the information will be transferred to signals, which can be corrupted when flowing through the channels. Such corruption may shows in different forms: signals can be distorted by multiple sources of noise; signals can be delayed; or signals can be weakened or added by any events in the channel, which results in interfered signals and corrupted information in the receiver. In the transmission channel representing memories, the written information can be damaged as well, due to the different reasons for errors when the memory is used in the systems for the final application, and it is likely that the read message is no longer the same as the original information.

The reliability that a memory can offer is measured by the error probability. This probability may not be satisfactory to the user, thus it follows that a NAND flash memory needs some help to achieve a reliability suitable for computing applications. Error correction codes add redundant bits called parity bits to the information data bits so

that, on reception, it is possible to detect the errors and to recover the message that was most likely transmitted. Through error correction codes, the error probability offered by the memory can be reduced to a desired level. The error probability of the ECC decoder can be written as

$$p = \frac{Number\ of\ bit\ errors}{Total\ number\ of\ bits} \tag{2.9}$$

Two main issues arise when an ECC is used inside a NAND flash memory system. First of all, the ECC should not limit the bandwidth, being the bottleneck of the entire SSD drive. This implies a hardware implementation that needs to handle multiple devices (channels) in parallel. At the same time, ECC must avoid erroneous corrections when the code reaches its limitation of error correction capability.

Hamming made his achievements in the area of error correction coding while investigating the communication on long telephone lines with the corruptions from lightning and crosstalk. The ECC invented by Hamming has the ability to recover the data with one error bit only, but the implementation of this code is effortless, making it widely used in some systems with small probability of errors or on which the correction of one bit error is thought to be enough.

The BCH and Reed-Solomon codes were developed in the late 1950s, which were considered to be more powerful than the Hamming codes. The first ones were elaborated by Bose and Chaudhuri [5] and independently by Hocquengheim; the second ones were described by Reed and Solomon a few years later [53]. These codes had been quickly applied in satellite communication systems since they were invented, and they are still popular today in applications such as compact discs.

Afterwards, the convolutional codes, brought in firstly by by Elias in 1955, have been able to attact the attention of people and attempt to substitute the BCH and Reed-Solomon codes in space missions. The study of optimum convolutional codes and the efficient decoding algorithms continued until 1993 when turbo codes were introduced for the first time [3]. They have received great success in the sector of telecommunications.

LDPC (low density parity checks) codes have a singular history. First discovered in 1962 by Gallager [20], their applications are being studied only recently [38]. There is another class of capacity-approaching codes, Turbo codes, which were discovered in 1993, became the coding scheme of choice in the late 1990s [41]. The Turbo codes are used more frequently for applications such as the Deep Space Network and satellite communications. Nevertheless, in the last few years, the advances of LDPC codes have seen them surpass turbo codes in terms of performance in the higher code rate range, leaving turbo codes better suited for the lower code rates only.

As the NAND technology scales down, ECC is becoming a critical design issue. In the following chapters, we will investigate high-performance ECC schemes for the practical design of NAND flash memory systems.

2.3 NAND-based Memory System Overview

In terms of NAND flash applications, the flash disk drives, solid state drives and secure digital cards are unquestionably the best known examples worth being mentioned. A number of NAND flash based designs are now available commercially, with different data bus interface and form factors, according to the specific market needs. For instance the large size memory cards, such as compact flash (CF) cards are used in digital cameras that require big memory capacity while the very small-sized removable media are popular in mobile phones, such as micro secure digital (SD) card. There are several protocols available to implement the bus interface for flash cards, and they are classified into parallel or serial, synchronous or asynchronous. Moreover, most of the NAND flash cards have the feature of hot insertion and hot extraction, which require the ability to handle the unexpected loss of power supply while guaranteeing the integrity of the saved data. There are also multiple interface standards for the SSDs, such as the universal serial bus (USB), serial ATA (SATA) and parallel ATA (PATA), PCI express (PCIe), and so on.

Although there are different interfaces for communicating with the host, a typical NAND flash based memory system is generally composed of two parts: a memory

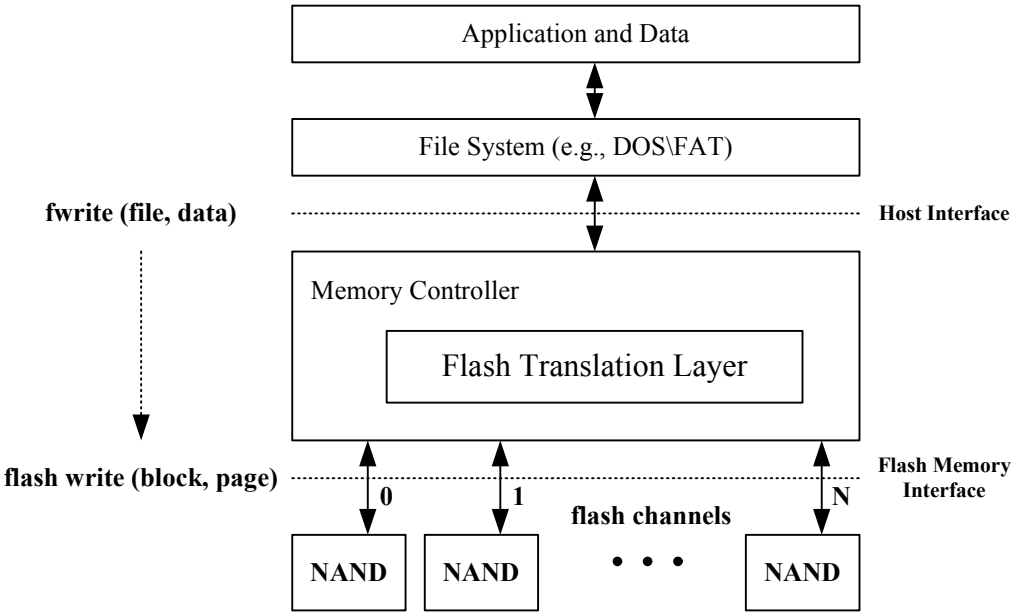


Fig. 2.8 Functional representation of a NAND-based memory system.

controller and NAND flash memories. The memory controller is the core part of the NAND-based system. Firstly, it provides the most suitable interface and protocol to both the host and the flash memories. Secondly, it efficiently handles data, and maximizes transfer speed, data integrity and information retention. Finally, the memory controller deals with the data in advance according to the NAND flash characteristics to achieve highly reliable storage. The Flash Translation Layer (FTL) is usually implemented in the form of firmware designed with the controller, and each algorithm is performed in the respective sub-layer. Detailed descriptions of these functions are presented in Chapter 3. On the edge of memories, an 8-bit data bus, generally called channel, is used to connect different memory chips to the controller. It is worth noting that placing multiple flash memories in a system is an efficient way to increase both storage capability and data throughput.

Figure 2.8 shows a functional representation of a memory card or SSD. Two types of components can be identified: the memory controller and the NAND flash memory components. Actual implementation may vary, but the FTL functions are always present.

2.4 Summary and Challenges

This chapter presents the general background on NAND flash memory as preliminaries for the next chapters. The three operations are detailed after discussing the NAND architecture and organization. Programming the MLC NAND flash moves the cell threshold voltage to certain levels that represent multiple bits, during which the cell threshold voltage is supposed to follow a uniform distribution. However, such situation is considered as ideal since the cell threshold voltage would be severely distorted by a combination of noise in the programming and retention, resulting in data unreliability. Mathematical descriptions of the channel noise and distortion are presented.

Using error correction codes is an important way to resolve the data reliability issue in NAND flash memory, however, the uniqueness of the flash memory channel compared to other communication channels prohibits the application of advanced ECCs. One challenge to be noted is the fact that the flash channel outputs are hard-decision only. Another challenge is there is no flash channel model available in the open literature that has incorporated all of the channels noise sources and mathematically characterized the final threshold voltage distributions. In the rest of this dissertation, we study these research problems and propose efficient solutions for each of them in an effort to further improve the NAND flash data reliability.

Chapter 3

Write Pattern Formatting Algorithm and Implementation

As discussed in Section 2.2.1, the data retention and program disturbance errors caused by various sources of noise are two critical factors affecting the NAND flash reliability at the flash memory physical level. In general, the reliability of both data retention and program disturbance degrades with the flash device scaling. The program disturbance problem has been alleviated in current design practice by employing dummy cells in recent NAND flash memories [67]. As a result, the data retention error rate has become the more prominent type of error.

Error correction codes are usually used as a system-level technique to improve reliability within almost all NAND flash memory systems. Before going into the ECC module, the data patterns should be well processed with the aim of improving reliability. Figure 3.1 shows the flash translation layer of SSD where the flash memory management functions, ECC, and pre-coding modules (that perform pattern processing before error control coding) are implemented [26]. Traditionally, simple algebraic codes such as Bose-Chaudhuri-Hocquenghem (BCH) codes [10, 24] are used for ECC, and pre-coding consists of randomized interleaving. In recent years however, many researchers have sought to use more advanced ECC, e.g., low density parity check (LDPC) codes, for flash memories to tolerate higher memory raw bit error rate (BER) introduced by device

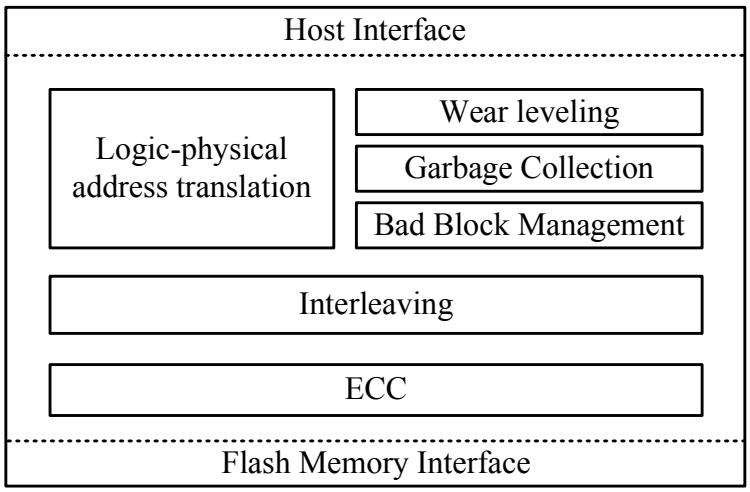


Fig. 3.1 Functions implemented in the flash translation layer.

scaling [16, 29]. Reportedly LDPC coding has already been implemented in the latest SSD products [2]. Thus, highly reliable pre-coding schemes that can be combined with stronger error correction codes are essential to the design of future NAND-based memory systems.

Beside the reliability issue, the power crisis that comes with technology scaling (specifically, the larger bit-line capacitance) is a major research issue. Bit-line capacitance in NAND flash memory is always large, a few picoFarads, since bit-lines are laid out with minimum half-pitch spacing and generally more than 5 mm long [18]. If the input data contains too many long column stripe patterns (referred to here as CSPs for convenience), these bit-line capacitances will be charged and discharged frequently during the programming, making a huge current flow over SSD which causes higher power consumption and possibly even malfunctions [64]. Detailed study on the impacts from CSPs can be found in Section 3.1. Therefore, pre-coding schemes that can reduce the power consumption will be preferable in future SSD systems.

For the hardware design of SSD systems, each individual module in FTL is generally implemented with field programmable gate arrays (FPGAs) [27, 34, 49]. To realize a cooperative interaction among all the modules, utilize the FPGA logic resources efficiently, and lower the processing latency, the complexity issue has to be carefully considered during the hardware implementation. Asymmetric coding and stripe pattern

elimination algorithm (SPEA) proposed by Tanakamaru et al. [65] have been shown to be excellent pre-ECC schemes to improve data retention reliability and reduce power consumption. Unfortunately, the complexity of the SPEA module increases significantly as the code length becomes larger. In addition, even though the CSPs of the input stream are successfully eliminated, some additional undesirable CSPs are also introduced during the data patterns processing.

Our goal in this chapter is to gain a strong understanding of the write pattern for NAND flash storage, and develop an efficient solution for the pre-coding module with low complexity implementation which will aid system designers to create a flash memory controller that ensures a high data reliability. We first present a write pattern formatting algorithm (WPFA) that achieves asymmetric processing and stripe elimination simultaneously, which allows the data stream to be modified only one time before being fed to the ECC module. Here it should be pointed out that the benefit of this solution comes with a cost of some minor loss of performance compared to asymmetric coding. WPFA is an improvement over the existing SPEA approach that avoids the extra CSPs introduced in SPEA while at the same time reduces power consumption. The hardware circuitry for WPFA is shown to use less gates and registers than Tanakamaru's design, with fully combinational logic. Both system complexity and latency are reduced substantially. Simulation and implementation results show a considerable reduction of both NAND cell overhead and FPGA resource utilization. The trade-offs between complexity and performance are analyzed quantitatively.

3.1 Problem Description

For the MLC flash memory, the cell threshold voltage is programmed to different levels by means of the ISPP scheme. Since one storage cell generally stores more than one bit belonging to different pages, it can be programmed by either the full-sequence programming, where all pages are programmed at the same time, or the multipage programming, where all pages are sequentially programmed at different times. Although full-sequence programming can achieve a higher programming throughput, it incurs more

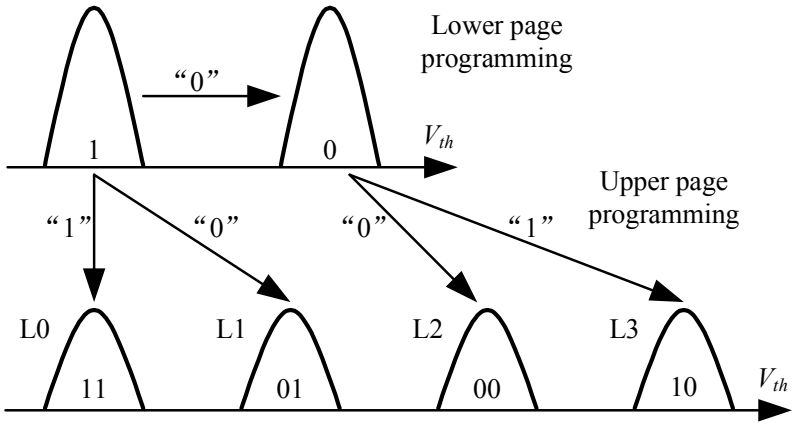


Fig. 3.2 Example of multipage programming for 2 bits/cell NAND flash memory.

severe cell-to-cell interference, which is an obstacle to technology scaling. Therefore, multipage programming is more attractive for NAND flash implementation today. In this chapter, we consider the prevalent 2 bits/cell MLC and multipage programming. The so-called lower page is programmed first, and the cell threshold voltages are divided into two programming states. The cell threshold voltages are then divided into four states (L0 to L3) by programming the upper page as shown in Figure 3.2.

Charge loss/gain occurring on the floating gate over time will lead to bit flipping. As a flash memory cell is programmed and erased repeatedly, the tunnel oxide and other insulating layers of the transistor become weak, which causes stress-induced leakage currents (SILC) and other charge losses [4, 25]. Additionally, de-trapping of trapped charges (surrogate for programmed charge) leads to changes in the original threshold voltage [43]. Both SILC and de-trapping will affect data retention and cause errors in reading data from flash memories; however, their impact varies. Fazio [17] demonstrated that charge losses for flash memory cells are not random but asymmetric, i.e., SILC leakages are mainly observed in cells of state L3 whereas de-trapping is the overwhelming phenomenon in cells of states L2 and L1. Furthermore, the measured results illustrated the asymmetric characteristic more clearly by stating that for data retention, only "0 $\rightarrow$ 1" errors occur in the lower pages and "1 $\rightarrow$ 0" errors are the dominant error in the upper pages [64]. It was also observed that most errors actually correspond to the V_{th} decrease (higher electric field across the tunnel oxide) of "10 $\rightarrow$ 00"

and "00 $\rightarrow$ 01". To reduce the data retention error rate, we must program most of the cells to the states "11" and "01" and fewer cells to "10" as possible, or in other words, distribute more 1's to lower pages and more 0's to upper pages. Different programming data patterns exert very different effects on the data retention reliability of NAND flash memory systems.

In the physical circuits of NAND flash memory, power consumption is increasing as the parasitic capacitance of the bit-lines (BLs) is charged or discharged alternatively in the programming. Denote by T_{pre} the time elapsed in waiting for the operation of charge transient and C_{bl} the total bit-line capacitances. During the pre-charge phase of programming, the average current consumption is

$$I_{pre} = C_{bl}\frac{\Delta V}{T_{pre}}n_{bl} \tag{3.1}$$

where n_{bl} and ΔV represent the number of bit-lines charged in parallel and the change of bit-line voltage applied in a program operation, respectively [42]. Figure 3.3 depicts the pre-charging condition under different programming data patterns. Column stripe patterns maximize the pre-charging current because bit-lines are alternatively biased to V_{cc} and V_{ss} such that all of the inter bit-line capacitances are charged and a large current flows. In contrast, when every bit-line is biased to V_{cc} (with all 1's data pattern) or V_{ss} (with all 0's data pattern), the inter bit-line capacitance diminishes and the pre-charging current decreases. The power crisis becomes much worse when the feature size of flash memory gets to be lower than 20 nm, then the bit-line capacitance sharply increases to more than ten picoFarads due to the increase of inter bit-line capacitance [55]. Figure 2.7 also shows the trend of the bit-line capacitance increases. In the design of power supply system of SSD, the worst case of the power consumption must be considered because the worst case large current may cause a catastrophic voltage drop and lead to malfunctions.

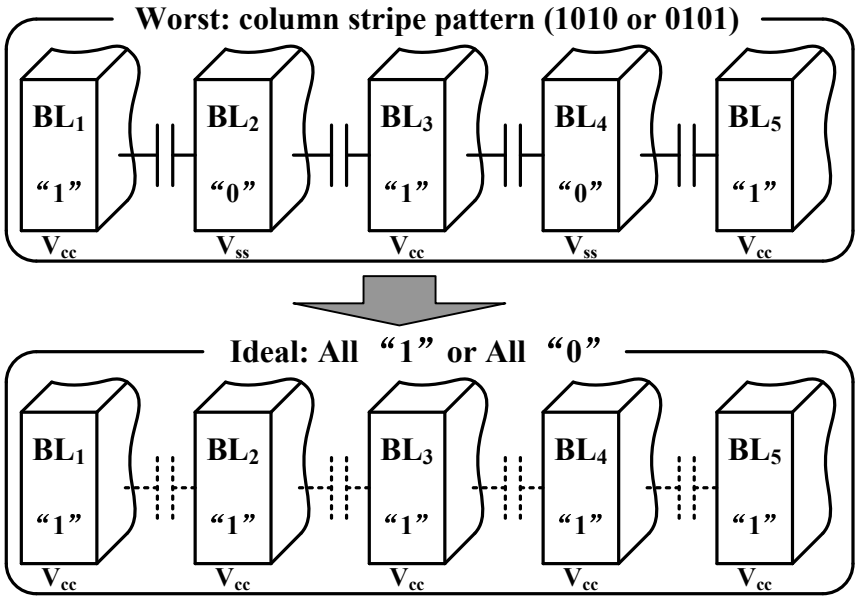
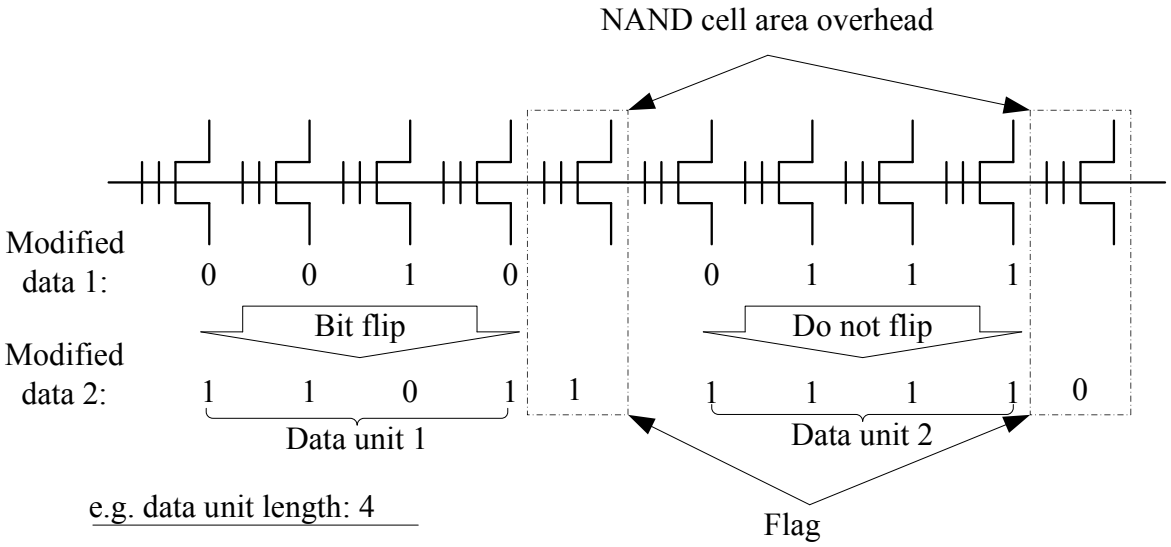


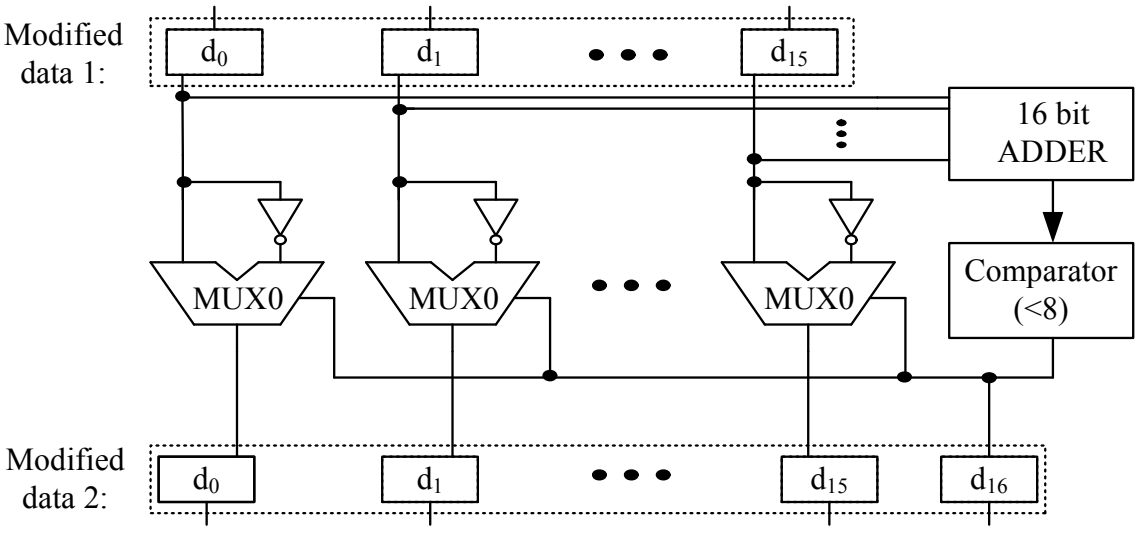
Fig. 3.3 Operation of pre-charging under different programming data patterns

3.2 Related Work

Considering data retention failures and high current pressure, bitstreams should be properly processed before being written to the NAND flash memory array. Asymmetric coding has been proposed by Tanakamaru et al. [65] to deal with the data patterns for improving the retention reliability. The algorithm calculates the number of 1's in the data unit and compares the number to a threshold value, and the result is used to determine whether the bits within the unit are flipped or not. If "0"s data is major, all data are inverted. Consequently, the distribution of 1's becomes asymmetric and therefore the number of cells at high electric field states such as "00" and "10" decreases. Figure 3.4(a) illustrates the concept of asymmetric decoding, where the code length is set to be 5 bits. Figure 3.4(b) shows the circuit schematic of the asymmetric coding encoder. The number of "1"s in the data unit is calculated as the sum of the input data with the 16 bit adder circuit. The comparator judges if the sum is larger than 8. If the comparator detects that the result of the summation is larger than 8, the comparator outputs "0" and the input data is output through MUX with "0" flag. If the comparator detects that the result of the adder is equal or smaller than 8, the comparator inverts the input data and outputs the inverted data with the "1" flag.



(a)



(b)

Fig. 3.4 (a) Algorithm of the asymmetric coding. (b) Schematic circuit diagram of the asymmetric coding encoder.

The "modified data 1" shown in Figure 3.4 is the processing result of the strip pattern elimination algorithm, which was proposed by Tanakamaru as well but used for saving power consumption in the NAND based memory systems. SPEA modifies the original data from the host to eliminate the worst case column stripe patterns. More specifically, SPEA calculates the difference between the numbers of 1's in even and odd columns of each data segment. If the difference is higher than a threshold value, bits of the segment are rearranged, and the odd or even column data which has the larger number of 1's is flipped. As a result, CSPs are eliminated and the current is lowered.

Even though Tanakamaru's asymmetric coding and SPEA improve the SSD performance considerably, the disadvantages are also clear after a careful study. Firstly, the implementation of these two approaches are still highly complex compared with randomized interleaving, especially when the code length of SPEA increases. In addition, the numbers of flag bits used in these two algorithms is relatively high. Usually, in the 20 nm-class NAND flash memory, the number of spare bits is almost 10% of that of data bits. Since a large portion of these spare bits are used for ECC, it is quite difficult to apply Tanakamaru's schemes when the data unit size is small. The implementation complexity will be discussed in detail in the following sections of this chapter. Thirdly, some extra CSPs are introduced during the SPEA processing which could cause power problems as well. Consider the example case shown in Figure 3.5 where the threshold value, denoted as N_{TH}, is set to 4, and SPEA is applied since the calculated difference is higher than the threshold value. After rearranging and flipping, lengthy CSPs can be observed in both even and odd bits of the modified data. This problem is serious since there are actually a notable number of unintentional CSPs induced after SPEA processing although we have not yet found a way to quantify the number. The example in Figure 3.5 represents a class rather than an individual of data patterns. For the original data of Figure 3.5, if we change "11001100$\cdots$" to "00110011$\cdots$", or if we change the size of "1010" to "101010", "10101010" etc., or if we insert "1010" to other positions, the resulting data patterns are problematic as well.

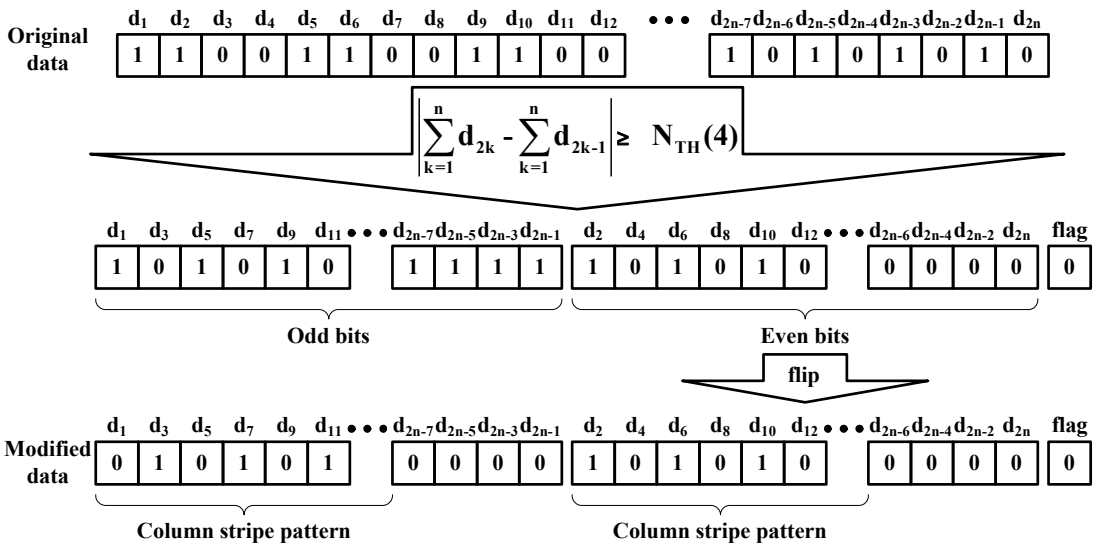


Fig. 3.5 Example of SPEA.

The proposed WPFA approach basically relies on bit-flipping operations over program data patterns. The bit-flipping aims to intentionally distribute the 1's and 0's within data patterns to make them become asymmetric and eliminate the column stripe patterns within. The drawback to SPEA described above has been avoided and no extra CSPs are introduced in the proposed design. In terms of the storage utilization, the proposed algorithm consumes less overhead of NAND cell area and exhibits better performance on power consumption compared to Tanakamaru's design. Associated hardware design issues are considered with the aim of reducing complexity and utilizing less logic resources. The pre-coding scheme based on the proposed algorithm would consume limited hardware resources in the implementation of FPGA-based FTL because of the low complexity design considerations. In the following sections, we will elaborate the WPFA scheme in detail and evaluate its performance through both computer simulations and FPGA implementation. We will also compare the proposed algorithm with asymmetric coding and SPEA approaches from different angles and analyse the hardware complexity from the practical point of view.

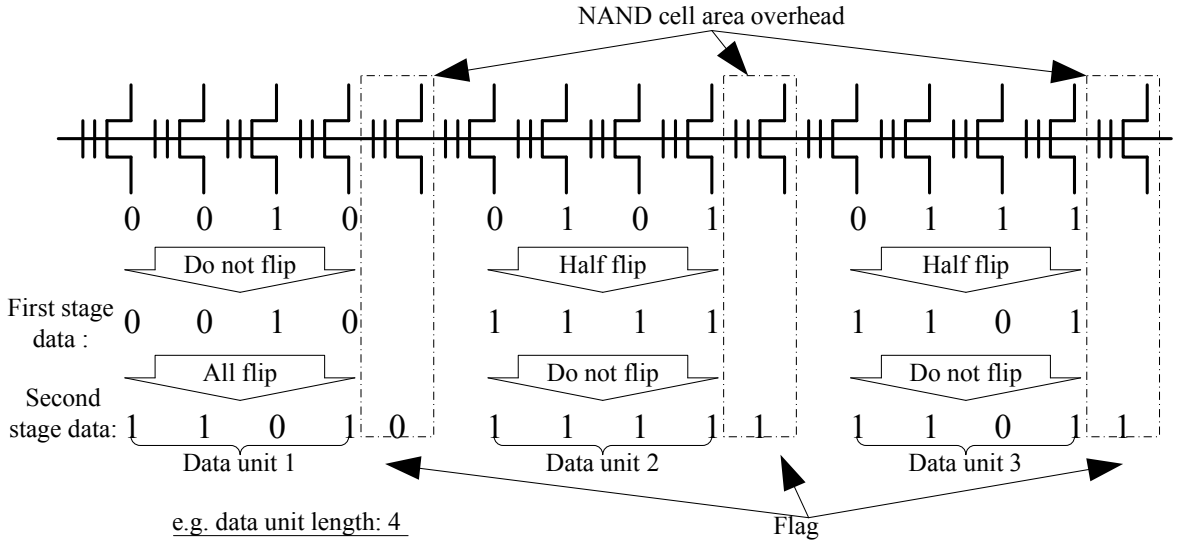


Fig. 3.6 WPFA: lower page data.

3.3 Write Pattern Formatting Algorithm

Although both are essentially write pattern processing, the asymmetric coding and SPEA take the data patterns separately and demand two pieces of hardware in the flash memory controller. As pointed out in the last section, in contrast our work proposes to use a single scheme for data pattern processing before ECC in SSD systems. Figure 3.6 illustrates the idea of the proposed WPFA approach with lower page input data. The presented solution discloses two primary stages, modifying the program data to eliminate the column stripe patterns and increase the number of "1"s, respectively. Note that the length of each data processing unit has been restricted to 2^n where n equals to $3, 4, 5, \ldots$. A data sequence which has been already specified as the upper page or lower page by the controller, can be divided into several data units for processing. In terms of a sequence that does not meet the length requirement, the last data unit will be ignored and passed to the ECC module directly. Initially, all bits of the data unit are added together and the result is stored in an n-bit sum register. For example, in Figure 3.6, the length of the user data unit is 4 bits and the width of the sum register is 2 bits.

At the first stage, all of the possible data patterns are organized into two separate groups differentiated by the most significant bit (MSB) of the sum register. WPFA

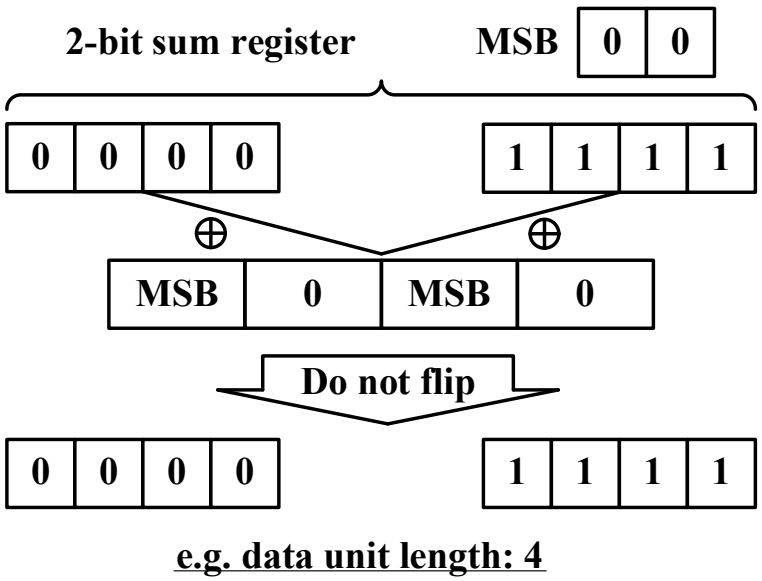


Fig. 3.7 Analysis of two specific data patterns

eliminates the CSPs in the following way. If the MSB equals 0, data is not modified and the data unit is passed directly to the next processing stage, such as “data unit 1” in Figure 3.6. If the MSB equals 1, the column stripe sequence is added to the input data unit, and the modulo-2 result is taken as “first stage data”, in other words, half of the input data will be flipped. In the example shown in Figure 3.6, both the “data unit 2” and “data unit 3” belong to this group so their data bits are half-flipped. As a result, column stripe pattern (data unit 2) has been eliminated.

To demonstrate the impossibility of extra CSPs, consider two specific data patterns: all-zero and all-one patterns as shown in Figure 3.7. Firstly, only these two kinds of data patterns are possible to bring extra CSPs if half-flipping operations are performed. Secondly, half-flipping will not be performed under the circumstances of these two kinds of data patterns since the MSB is 0 for both of the sum registers. Note that in the case of the all-one data pattern, the MSB of the register is also 0 because of the overflow. According to these two facts, it can be observed that no extra CSPs will be introduced in the proposed algorithm. Due to its smaller code length, the maximum length of CSPs after the WPFA processing is shorter than SPEA processing, which is proved in the next section.

As for data retention, the worst case data pattern occurs when all of the memory cells are programmed to the highest V_{th} level, where the large program voltage stress and higher electric field across the tunneling oxide degrades the memory cell reliability. To reduce these effects, randomized interleaving may be used, in which the probability of "10" and "00" is about 25% each of the total data on the condition that the output data is completely random. In this proposed scheme, data retention reliability has been further improved by increasing "1"- and "0"-data of lower and upper pages, respectively. This corresponds to the second stage of the WPFA scheme.

At the second stage, the MSB continues to be used for determining whether the "first stage data" is flipped or not. If the MSB equals 0, the 0's are considered to be major in the input data pattern so all bits of "first stage data" are flipped and the result is taken as "second stage data", such as "data unit 1" in Figure 3.6. On the other hand, if the MSB equals 1, data of "first stage data" will not be modified, for example, "data unit 2" and "data unit 3". Consequently, the number of 1's in lower pages data increases. This part of algorithm is similar to asymmetric coding; however, in the corresponding circuit, as discussed in Section 3.5.1, the comparator and multiplexer have been replaced with only XOR gates, thus resulting in simpler circuitry. At the end of the algorithm, "second stage data" together with the flag bit that comes from the MSB are used to form the output.

Regarding the WPFA for upper pages data, the idea is the same except to decrease the number of 1's. Figure 3.8 illustrates the idea of algorithm for the upper page data. We also divide the data patterns processing to two separate stages here. The first stage processing is exactly the same as that for the lower page data while the second stage is just the opposite way to that used in the lower page data processing. If the MSB equals 1, the 1's are considered to be major in the input data so all bits of "first stage data" are flipped and the result is taken as "second stage data", such as "data unit 2" and "data unit 3" in Figure 3.8. In this scenario we assume that the original data is random enough so the proportion of 1's will not be changed by the first stage processing. On the other hand, if the MSB equals 0, data of "first stage data" will not be modified, which

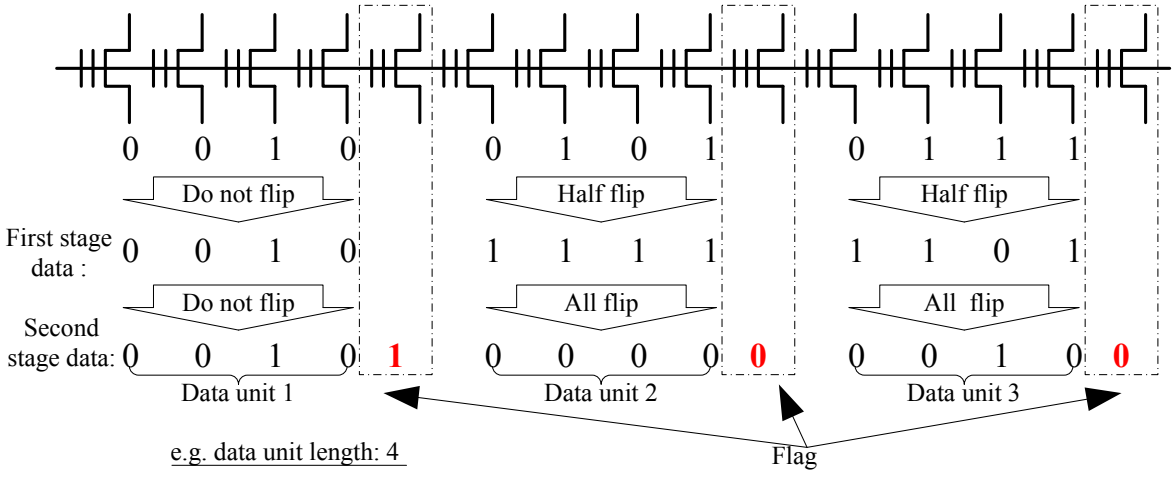


Fig. 3.8 WPFA: upper page data.

is exampled with “data unit 1” in the figure. Consequently, the number of 0’s in upper pages increases. Note that the data is actually not modified and passed directly to the following ECC module if the MSB equals 0. The flag bit is obtained by inverting the MSB of each data unit for the purpose of more “0”s in the resulted data.

3.4 Performance Evaluation

Based upon the algorithm described above, this section presents simulation results to demonstrate its advantages compared to randomized interleaving and Tanakamaru’s asymmetric and SPEA approaches. The trade-offs over asymmetric coding are also analyzed. In these simulations, the data unit length of the proposed algorithm is fixed to be the same as asymmetric coding for fair comparison.

3.4.1 Maximum Length of Column Stripe Patterns

A figure of merit for energy savings is taken to be the maximum length of column stripe patterns after processing, from which the current spike results. The current spike gets higher as the maximum length of CSP increases. If the NAND flash system is not designed to tolerate the current spike, a long pattern of CSP will probably cause damage

to NAND flash memory system. Let M and N denote the code lengths of WPFA and SPEA, respectively. We note the case that the flag can also become part of the CSP. Considering this, the maximum length of CSPs for the proposed WPFA will be M. It can be simply proved as follows. Firstly, the data width of the output for WPFA encoder, including flag, is $M+1$, so the maximum length $\leq M+1$. Secondly, CSP is impossible to be seen at the output of WPFA, otherwise it will be eliminated during the processing. It therefore implies that the maximum length of CSPs for WPFA is M. Figure 3.6 has also shown this merit under the situation of 4-bit data unit. For SPEA, on the other hand, the maximum length of CSP is $N/2 - N_{TH}$ based on Figure 3.5. Since $N >> M$ [65], the maximum length of CSPs has been substantially reduced which suggests the memory system is better protected from potential damage by current spikes.

3.4.2 Average Program Current

Apart from the maximum length of stripe patterns, the average program current is another metric for evaluating performance of the proposed algorithm on energy saving. The bit-line capacitance of a NAND flash memory is composed of the inter bit-line capacitance C_{bl-bl} and the others denoted C_{others}. In case that the program data of the memory cell connected to the n_{th} bit-line BL_n is 1, BL_n will be pre-charged to V_{cc} and the memory cell is forced to program inhibited state [61]. Meanwhile, the program data of memory cells connected to BL_{n-1} and BL_{n+1} determine whether the inter bit-line capacitance is charged or not. If both of adjacent bit-lines, BL_{n-1} and BL_{n+1} are pre-charged to V_{cc}, BL_n will only charge C_{others} because the effect of C_{bl-bl} has been eliminated. If both of adjacent bit-lines are not pre-charged, BL_n will charge C_{others} and two C_{bl-bl} because BL_{n-1} and BL_{n+1} are biased to V_{ss} (column stripe pattern). The last case is that one of the adjacent bit-lines is pre-charged to V_{cc}, in which BL_n charges C_{others} and one C_{bl-bl}.

Considering these three situations and assuming that charging the bit-lines is the dominant component of the program current for NAND flash memories, we can then calculate the average current per page-programming according to (3.1). To this end, we built a simulator based on 2 bits/cell MLC flash memory having the page length of

8 KB and 256 pages per block. The memory array is modelled using the MATLAB with some memory properties attached to the elements in the rows and columns of the MATLAB array. The simulator uses the physical parameters of NAND flash memory presented by Fukuda et al. [18], where C_{bl-bl} and C_{others} occupy 78% and 22% of the total bit-line capacitance, respectively. We assume 1 μs charging time and consider three memory systems with the random input data stream. SPEA and asymmetric coding are employed in the first system whilst the proposed algorithm is employed in the second one. The third system is used as a reference since it does not use power saving scheme and employs only asymmetric coding. Both the asymmetric coding and the proposed algorithm utilize the same length of data unit.

The average page-programming current is calculated when 16 blocks (4096 pages) data are written to the memory systems. The reduced program current over the reference system as the bit-line capacitance increases is shown in Figure 3.9. In this experiment, the data unit length is set to 8 and the N_{TH} of SPEA is set to 6. Interested readers are referred to reference [65] for more knowledge about the calculated bit-line capacitance under different feature sizes. The bit-line capacitance is approximately 1 pF for the NAND flash technology of 40 nm. In this simulation we compared the reduced program current rather than the reduction rate since the latter is not a good metric to illustrate the difference. The reduction rates are the same for every case of bit-line capacitance, but different as code length changes. For example, under the SPEA code length of 513 and WPFA 8, the reduction rate is 9.6% and 18% for $1pF \sim 10pF$ bit-line capacitances, respectively. Please note the total program current is not shown in this figure as it is different for each bit-line capacitance and code length.

As observed from the figure, in terms of energy consumption, the system employed with the proposed algorithm outperforms the system employed with SPEA whose codeword is higher than 257 bits. An exception is that the SPEA 129 reduces more current than WPFA 8. This is because the SPEA encoder will conduct more operations on the input data patterns when the data unit length is short, thus its performance improves. However, the short SPEA code length is generally not adopted in the Tanakamaru's

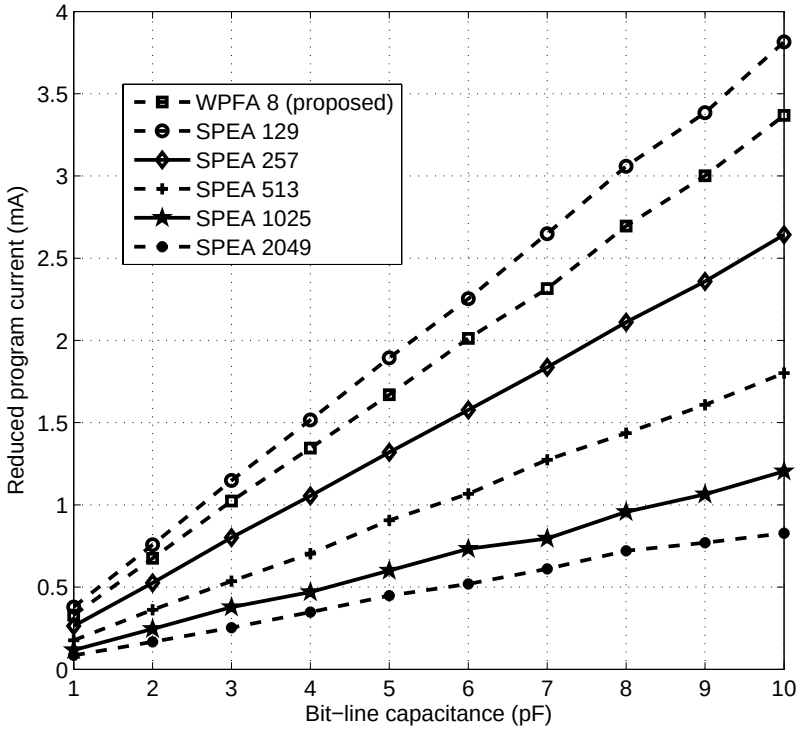


Fig. 3.9 Reduced program current over the system without power saving scheme

design since it will induce more cell area overhead. Besides, the maximum length of CSPs should be considered as well because it may cause a catastrophic voltage drop and leads to the malfunction of SSD, even though the average energy consumption may not be very high. Due to the fact that the large codeword is generally used in SPEA, the advantages of the proposed algorithm are noteworthy.

3.4.3 Proportion of the Highest V_{th} State

To theoretically analyze the proportion of NAND cells on the highest V_{th} state, it is necessary to derive the amount of 1's in lower pages (or 0's in upper pages) programming data. Assuming the input data is random enough and its length is sufficiently larger than the data processing unit, the proportion of 1's in lower pages is calculated in the following way.

All possible input patterns are divided into two groups: A and B, by the MSB of sum register, as shown in Figure 3.10. For group A with 0-MSB, the second stage of

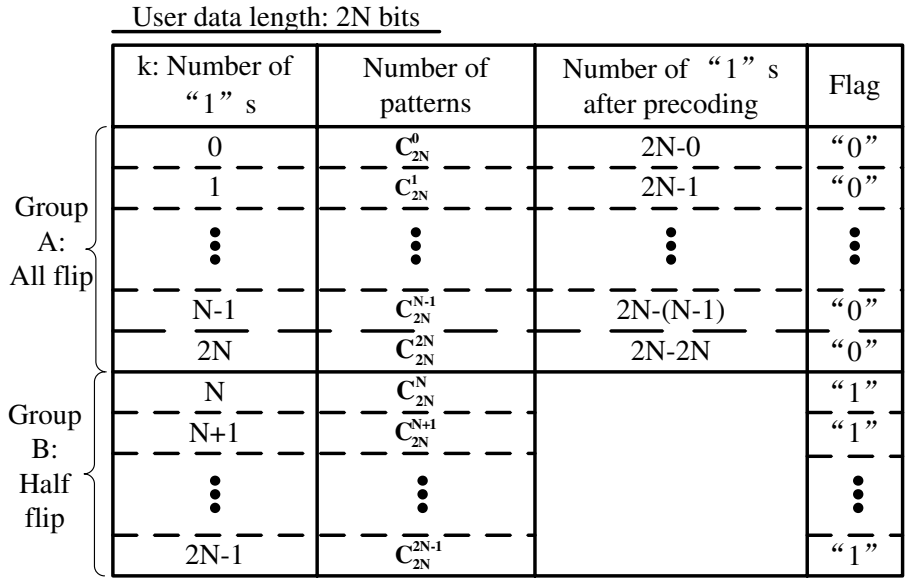


Fig. 3.10 Data patterns before and after WPFA processing.

algorithm is performed and all bits are flipped after WPFA processing. Based on the data presented in Figure 3.10, the number of 1's (with flag bits) for group A is given by

$$\mathrm{N}_1^A = \sum_{k=0}^{N-1} C_{2N}^k (2N-k) \tag{3.2}$$

For group B with 1-MSB, the first stage of the algorithm is performed and only half of the bits are flipped. It is not possible to exactly determine the number of 1's for each individual data pattern. Apparently there are a number of candidates (C_{2N}^N for the pattern with N "1"s) corresponding to each type of pattern. If the pattern is unknown it is also impossible to determine the number of 1's after the half-flip, which is the reason it is not shown in Figure 3.10. Nonetheless, the total number of 1's after processing can still be calculated as follows. Since the 1's data takes the majority in data patterns of group B, both half-flip and all-flip will result in the loss of number of 1's in the data pattern. Furthermore, it is easy to ascertain the loss due to half-flip is 1/2 of the loss due to all-flip because of the symmetric characteristic. If we think of the number of 1's related to flags, the total number of 1's in group B after WPFA processing is expressed as

$$\mathrm{N}_1^B = \sum_{k=N}^{2N-1} C_{2N}^k (2N) - \frac{1}{2} \sum_{k=N}^{2N-1} C_{2N}^k (k) + \sum_{k=N}^{2N-1} C_{2N}^k \tag{3.3}$$

Since the total number of data patterns is $2^{2N}(2N+1)$, the probability of 1's (P_1) is calculated by dividing the number of 1's in all data patterns with flag bits by $2^{2N}(2N+1)$, which is

$$\begin{aligned} \mathrm{P}_1 &= \frac{\mathrm{N}_1^A + \mathrm{N}_1^B}{2^{2N}(2N+1)} \\ &= \frac{\sum_{k=N}^{2N-1} C_{2N}^k (N+1) + \sum_{k=0}^{N-1} C_{2N}^k (2N-k)}{2^{2N}(2N+1)} \end{aligned} \tag{3.4}$$

Note that for the flag bits, we could choose either 1 or 0 for the specific group; however, setting the flag bits of group B to be 1 can increase the probability of 1's of the output bit stream, and the condition is adverse for the upper pages. Another special case is the processing of patterns consisting of all 1's data. If this pattern is fed into the encoder, the output data becomes all 0's due to the all flip operation, which seems to be contrary to the intention to increase the number of 1's. Nevertheless, we note that the number of all 1's pattern is only one which is very small compared to the all 2^M data patterns (suppose the data unit size is M). Besides, the simulations hereafter will supply evidence that the probability of 1's has been improved when considering all the data patterns. Sacrificing the all 1's pattern is mainly for avoiding the problem that Tanakamaru faces, since all 1's and all 0's are the only two patterns which may induce unintentional CSPs after the half-flip operations of the first stage.

In computer simulations, the same length of information bits (sufficiently larger than the length of data unit) are randomly generated and fed into an asymmetric encoder and WPFA encoder. At the output, we measured the probability of 1's in lower pages data for each scheme, as illustrated in Figure 3.11. For the proposed algorithm, the theoretical result analyzed previously is also shown in the figure. When the length of input data is large enough, the simulated probability of 1's is fairly close to its theoretical counterpart. Because of the half-flip operations on data patterns having 1's as the majority, performance loss has been observed in the proposed design comparing to asymmetric coding. At the unit length of 2^4, this loss is about 5% in the target of

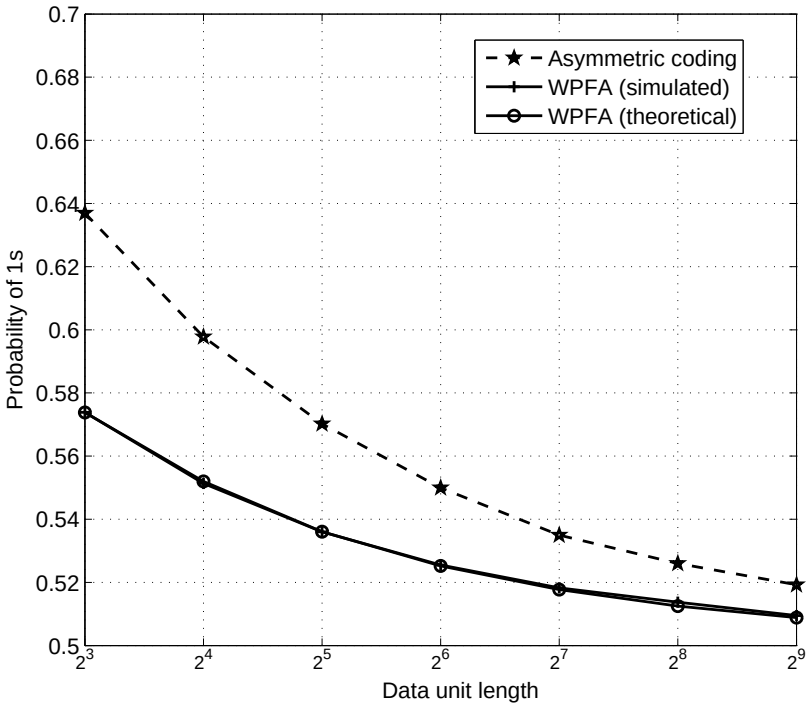


Fig. 3.11 Simulation results of probabilities of "1"s in Tanakamaru's and the proposed schemes.

1's probability. However, the performance gap between these two schemes gets smaller while the data unit length increases.

In the design of NAND flash based memory systems, if we set the data unit length to be 2^4, the WPFA approach will modify the data programmed to NAND so that at least 55% of the lower and upper pages are 1's and 0's, respectively. As a result, "10" and "00" occupy 20% and 24% of the total data, as shown in Figure 3.12. The probability of the highest V_{th} state, "10" has been reduced by 80% and 20% compared with the worst case and randomized coding, respectively. Consequently, the data retention error decreases and reliability is improved. Compared to asymmetric coding, the proposed design does not show better results in distributing the flash programming states. As shown in the figure, the "10" occupies 16% of the total data if using the asymmetric coding. This is considered to be the trade-off for our design to gain the benefits of less cell area overhead, lower hardware complexity, and lower power consumption.

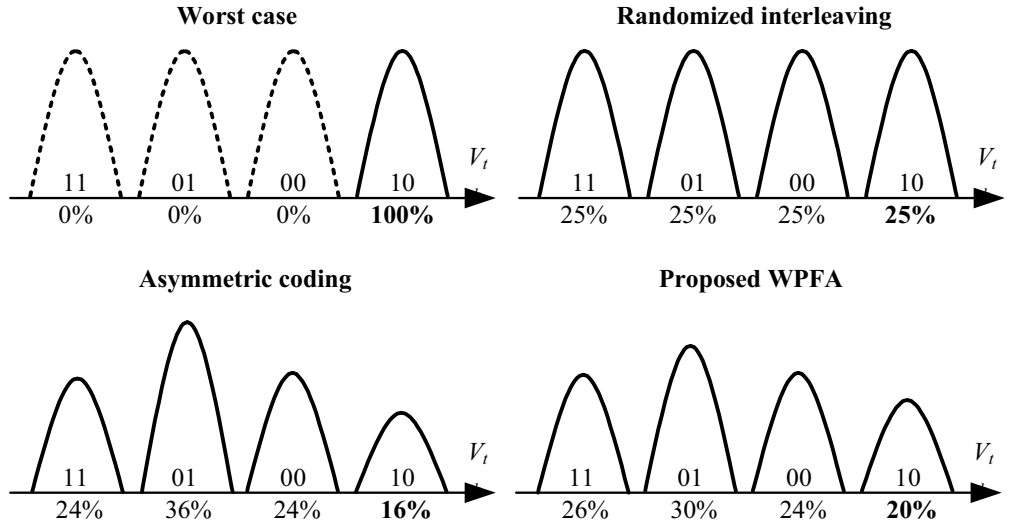


Fig. 3.12 V_{th} distributions of the worst case, randomized interleaving, and the proposed algorithm.

3.4.4 Overhead of NAND Cell Area

Due to the fact that extra flag bits have to be used for SPEA while the proposed scheme shares only one flag bit for both realizations, Tanakamaru's design consumes more NAND cell area. Figure 3.13 depicts the amount of flag bits required for different length of programming data input. In the figure, we simulated the system employed with WPFA of 128-bit data unit size, and three systems having the same 128-bit asymmetric coding but different data unit sizes of SPEA. It is seen that for the same number of processing data, the flag bits of WPFA are less than that of Tanakamaru's, which consumes even more cell area when the data unit size of SPEA gets shorter.

In general, the NAND flash memory is manufactured to offer a spare area upon production so that the flash management algorithms can use these space to store the redundant bits (with respect to the information bits). For example, in 20 nm class NAND flash memory, the number of spare bits is about 10% of that of data bits, which is 3 Kbits if the bit-line size is 32 Kbits. Usually, a large portion of these spare bits are required by the ECC thus we need to make sure the redundant bits used by other schemes are minimal. Therefore, the overhead of cell area is one of the important factors when

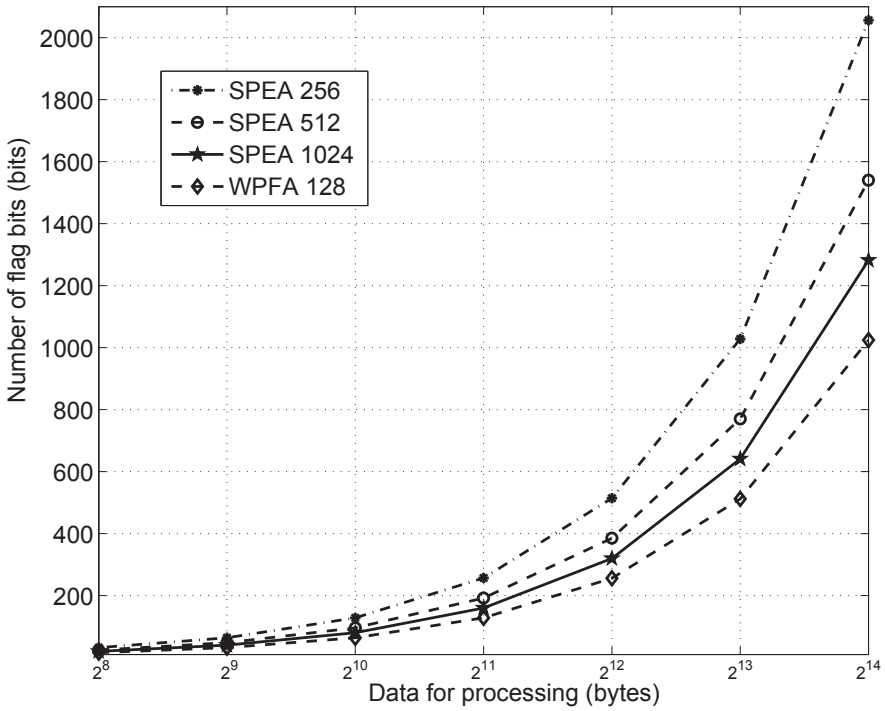


Fig. 3.13 The number of flag bits required for WPFA and Tanakamaru's algorithm.

designing the algorithms in the flash translation layer. The memory space for storing flag bits used in the WPFA comes from the spare area of NAND flash memory as well, so it is manipulated to prove that the current NAND flash memories have enough space to implement the proposed algorithm. Take MICRON's MT29E128G08 for example. This memory chip has the page size of 8640 bytes ($8192+448$ bytes), among which 448 bytes are spare bits. If we consider 128-bit data unit for WPFA, the number of flag bits for one-page data is 64 bytes. On the other hand, to accommodate the (525*8, 512*8, 8) BCH code which can correct up to 8 bit errors per 512 bytes data, a page only needs additional $(8192/512) \times 13 = 208$ bytes for the parities. As a result, the remaining spare space ($448 - 64 = 384$ bytes) is large enough for ECC and other memory algorithms, which is also possible for implementing other high-rate LDPC codes.

To numerically demonstrate the reduction for cell area overhead, we set the data unit length of WPFA to be same as asymmetric coding and calculate the flag bits used for each scheme respectively. Figure 3.14 depicts the reduced overhead of the system employed with the proposed scheme compared to three systems employed with same

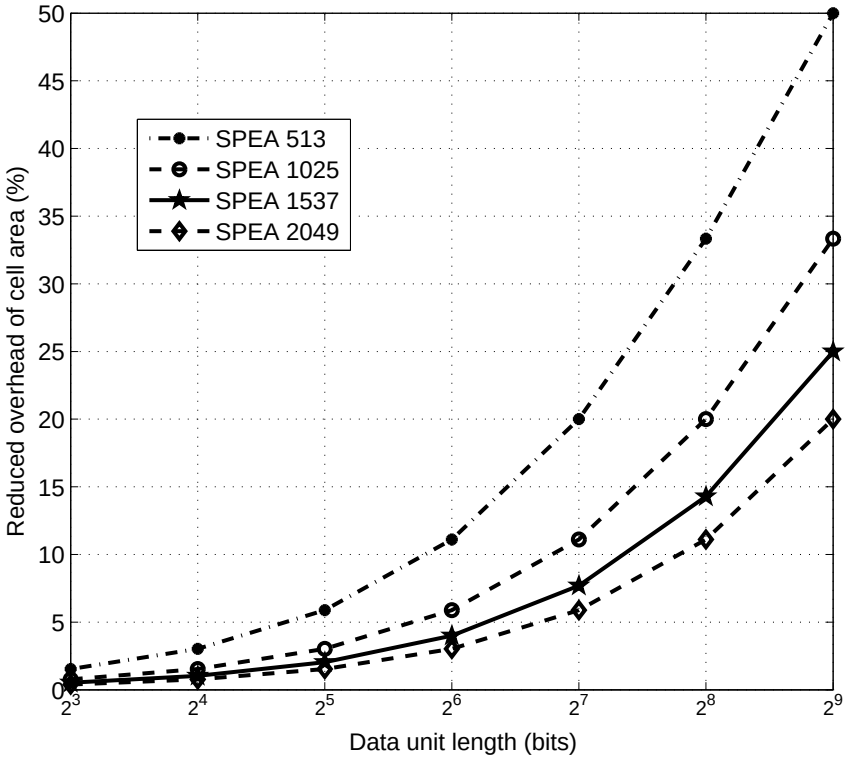


Fig. 3.14 The reduced NAND cell area overhead to Tanakamaru's design.

asymmetric coding and different length of SPEA codeword. As shown, the reduced overhead increases with data unit length and Tanakamaru's design consumes more cell due to the flag bits used in SPEA. Under the assumptions of 256-bit data unit and 1025-bit SPEA codeword, the extra cell area of Tanakamaru's design has been reduced as much as 20%. Hence, we can use a smaller length of data unit for WPFA to overcome the asymmetric performance loss discussed above.

3.5 Hardware Design and Implementation Complexity

In this section, we consider the logic circuits for the proposed algorithm and propose several ways to reduce the hardware complexity. Generally, WPFA will be implemented together with the ECC module as part of the flash controller in FPGA. The overall framework of FPGA-based flash controller is shown in Figure 3.15.

In terms of the proposed design, the proposed scheme is supposed to be applied before the ECC module in the data flow of programming. This is for two main reasons.

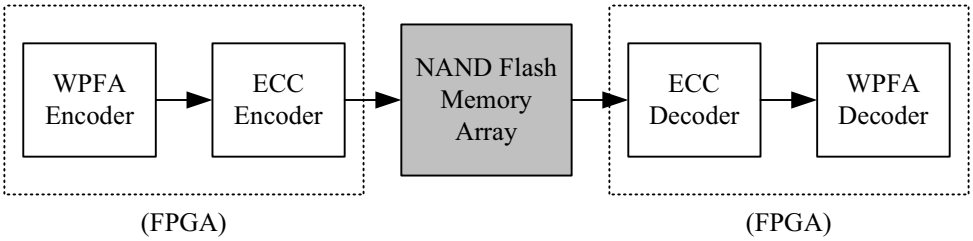


Fig. 3.15 Framework of coding mechanism in SSDs.

On the one hand, when the proposed scheme is applied first, data bits processing will not disorder the ECC encoding. In general, systematic error correction codes are adopted in SSD systems [79] thus the benefits and characteristics of WPFA can be well preserved during ECC encoding. On the other hand, it is more reasonable to apply ECC decoding first since errors occurring in flag bits of the proposed algorithm will result in higher bit error rate than other bits. For the design depicted in Figure 3.15, however, the ECC decoder will correct these errors in advance to make sure the exact flag bits are used in the subsequent WPFA decoding.

Generally, all of the memory management algorithms are implemented in the flash translation layer that is between the host system and memory array, as shown in Figure 2.8. The proposed WPFA will also be implemented in FTL, as a part of flash controller, which will precisely deliver user data and flag bits to the predetermined memory space. Except for the hardware used to realize the WPFA encoder and decoder, the extra management mainly results from the serial/parallel converter before WPFA and the data bits rearrangement after WPFA. Because the circuits for these two modules can be implemented with little resource of FPGA, the extra overhead is quite small. Meanwhile the serial/parallel converter is also required for the hardware implementation of asymmetric coding and SPEA, with which our design is compared in the experiments. On that basis we do not take the converter modules into account in this dissertation when discussing the utilization of hardware resources. The following sections detail the implementation of WPFA encoder and decoder and the potential issues in the practical design.

3.5.1 Hardware Implementation of WPFA Encoder and Decoder

The hardware for the proposed algorithm described above basically rely on simple logic gates commonly used in digital systems. In addition to the logic gates, an adder is used in the WPFA encoder to obtain the MSB, a critical signal for all the operations. Figure 3.16 illustrates the circuit structure for the WPFA encoder with 16-bit data unit. Before write pattern processing, the parallel 16-bit data and upper/lower-page select signal (U/L) are generated from the information bit stream by the serial/parallel converter. The number of 1's in the data unit is then calculated with 16-bit adder circuit and the MSB of the 4-bit sum register is used as a control signal for the subsequent computations. In the proposed circuit schematic, traditional comparators and multiplexers have been replaced with simple XOR gates to perform bit-flipping operations. The flag bit is created through a series of logic operations over MSB and U/L signals. More specifically, the MSB is XORed with the U/L signal and the result is inverted to represent the flag bit. For Tanakamaru's design, upper page and lower page asymmetric encoders are implemented with a separate circuit unit; nonetheless, these two WPFA encoders are integrated within a single circuit, which saves the FPGA resource utilization.

The upper page and lower page data processing are also integrated together in a single circuit in the implementation of WPFA decoder. Note that U/L signal will be correspondingly produced when the NAND controller fetches data from the memory array. Hence, in the decoding side, the circuit of decoder is easy to implement by performing exclusive OR operations over the data read, the flags and the U/L signal. Figure 3.17 illustrates the circuit structure for the WPFA decoder with 16-bit data unit. The last bit of the 16-bit data segment is treated as the flag and is inverted to represent the MSB that instructs the following logic operations. No adder/summer is used here, thus the hardware design for the decoder is even simpler. After a series of operations reversing the encoding, the original 15 bits data is recovered, i.e., d_0 to d_{15} shown in Figure 3.17.

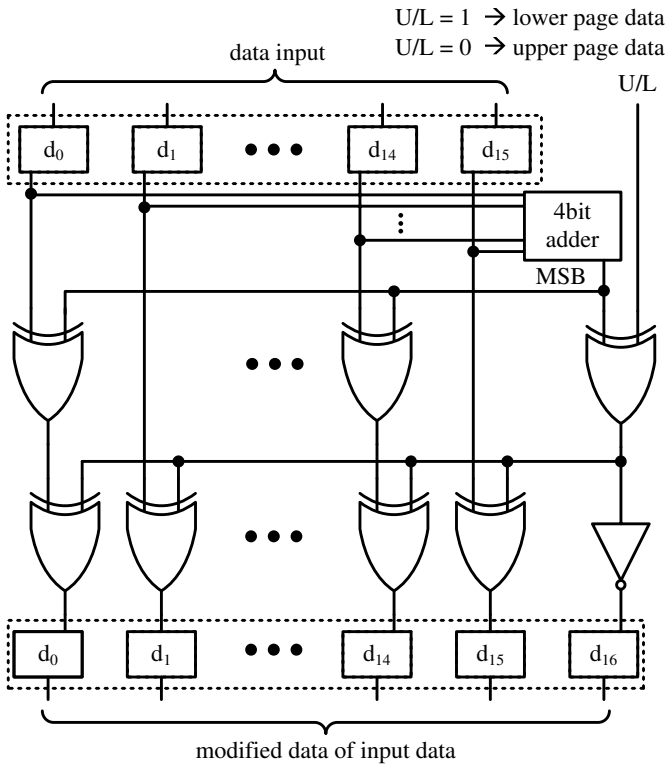


Fig. 3.16 Circuit schematic of the joint lower/upper pages WPFA encoder

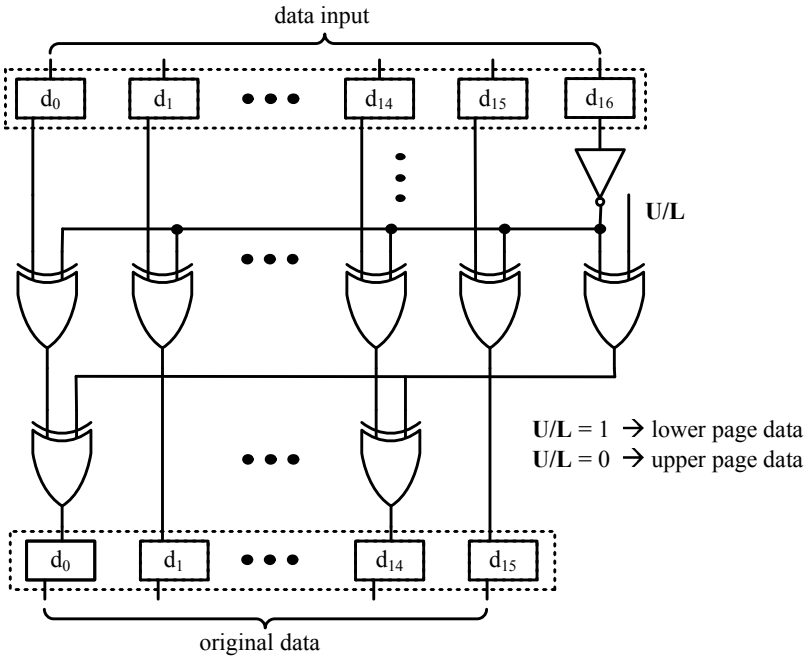


Fig. 3.17 Circuit schematic of the joint lower/upper pages WPFA decoder

Table 3.1 Comparisons of Resource Utilization in Tanakamaru's and Proposed Designs

Encoding Units	**ALUTs**	**Registers**	**Packed ALMs**
Asymmetric (len = 16)	81	34	43
Asymmetric (len = 128)	694	258	352
Asymmetric (len = 256)	1433	514	730
SPEA (len = 129, Nth = 6)	357	150	183
SPEA (len = 257, Nth = 6)	717	281	364
SPEA (len = 513, Nth = 6)	1432	540	727
SPEA (len = 1025, Nth = 6)	2828	1055	1433
WPFA (len = 16)	38	17	19
WPFA (len = 128)	352	129	178
WPFA (len = 256)	690	257	351

3.5.2 Analysis of Implementation Complexity

Due to the fully combinational circuits, the latency, circuit area, and logic resources related to the WPFA module are negligibly small. To quantify the hardware complexity of the discussed computation units, we used Verilog to model the proposed circuits. The encoding module is synthesized with Synplify Pro and Altera EP2S180 field-programmable gate array (FPGA) according to area optimization. The adaptive look-up tables (ALUTs) and logic registers utilized for encoding units are listed in Table 3.1. Results for Tanakamaru's schemes are also presented. In this experiment, the same serial/parallel conversion circuits are assumed for either design therefore we need only to compare the complexity of the computation units. The short data unit length, such as WPFA 16, is only used here only for comparisons. The synthesized results of this case are not very helpful for practical implementation since it introduces too much NAND cell overhead although it requires little FPGA resources.

Apparently, WPFA requires much less logic resources than that of asymmetric and SPEA encoders, especially when the code length of SPEA increases. Even compared to the design employed with asymmetric coding only, the proposed design still has lower complexity. For example, if we set the same length for the data processing unit, the utilization of all these three resources for WPFA is approximately half of that for asymmetric coding. The estimated adaptive logic modules (ALM) used for

WPFA encoder is about 8% of the ALMs consumed by Tanakamaru's design if the code length and N_{TH} of SPEA are set to 129 and 6, respectively. The resource utilization of Tanakamaru's design should be double if considering both lower and upper pages whereas that of the proposed design stays the same. Moreover, due to the fact that the adder is not required, the hardware for the decoder of WPFA is simpler than the encoder. Therefore, the hardware complexity is reduced even more if taking both the encoder and decoder into account.

In the practical design, there are a few issues that are worthy of note for the system designers. The first one is the parallelism. For the WPFA encoder and decoder, the parallelism is just the data width per processing unit. Choosing the long data unit will reduce the processing time. The second issue is the data processing throughput. As WPFA is implemented with fully combinational logics, the throughput of this block is actually equal to the transfer rate of serial input data. The throughput has to be well designed to match that of ECC to avoid data transferring errors, although this design rule may be changed subject to the specific timing. The third issue we would like to mention is the response time. On the one hand, the data processing of WPFA will influence the response time when writing and reading data from the NAND flash memory since the WPFA encoder and decoder are serial with the ECC encoder and decoder, respectively. On the other hand, due to the characteristics of the combinational logic, the processing time is very short in the proposed circuits for WPFA, which could achieve ultra-low latency. If serial/parallel converter is also implemented using combinational circuits, the operation of WPFA will not influence the response time of other parts of the controller. All these issues are subject to the specific requirements in the implementation of the proposed algorithm and flash memory controller.

3.6 Summary and Challenges

In this section, we present a write pattern formatting algorithm of low complexity to improve the data retention reliability and power consumption of NAND flash based memory systems. The proposed algorithm improves the existing SPEA approach to

completely eliminate column stripe patterns. With the proposed scheme, flash memory systems consume low energy and avoid suffering huge current spike. Furthermore, it reduces the overhead of NAND cell area while maintaining comparable performance. Simulation results show that the overhead for the proposed algorithm is about 80% of that for the existed design when using the same parameters of 256-bit data unit and 1025-bit SPEA codeword. Meanwhile, we studied the FPGA implementation issues for the proposed algorithm and analysed the hardware complexity. Hardware synthesized results over Altera EP2S180 demonstrate that the implementation complexity of the proposed scheme is much less than that of asymmetric coding and SPEA approach. For these reasons, the proposed algorithm is an attractive solution for practical low complexity pre-coding schemes of SSDs.

Although the benefits are noteworthy, a few challenges and issues must be resolved in the application of the proposed algorithm. Firstly, it is an important question to choose a proper length for the data processing unit. The larger data unit, the less storage overhead because fewer flag bits are required. And the performance in current reduction becomes worse since larger data unit leads to rougher processing in eliminating the column stripe pattern. However, there is a trade-off between these two aspects, and we believe the data unit size should be determined based on the specific situation. If the cell area overhead is more important, then a larger data unit can be used. If the design focus is more on energy saving, then a smaller size should be chosen.

Secondly, we understand that distributing the programming levels unevenly will improve the data retention reliability but are unable to quantitatively relate the proportion of each level to the retention reliability. Generally, the retention errors are not quantifiable but obtained statistically after thousands of hours of observations. In Tanakamaru's paper, the authors only theoretically demonstrated that increasing the probability of 1's in lower page data can reduce retention errors, but measured the retention errors experimentally. In our work, due to the limitations, we just analyse the probability of 1's, with which we prove the proposed algorithm can improve data retention in theory.

However, it will be practically useful if we can precisely know how many data retention induced bit errors can be reduced by applying the proposed scheme.

Thirdly, the impact on cell-to-cell interference resulting from the data patterns processing is unknown. Many flash vendors have implemented a randomizer on the NAND flash chip to make the portion of each of the levels close to 25%. The motivation for this is to reduce the cell-to-cell interference, in both word line direction and bit line direction. Apparently the proposed WPFA will make the data patterns unbalanced. Our hypothesis is that the cell-to-cell interference will be reduced after WPFA processing. The WPFA will lower the proportion of cells on high V_{th} states, so it is more likely for an interfering cell to have less threshold voltage shift, which further causes less threshold voltage shift on the victim cell. However, there is no evidence available to support this hypothesis until now. As cell-to-cell interference has now been recognized as one of the major noise sources in NAND flash memories, it will be very beneficial to dig deeper into this problem to get a clearer understanding of this issue.

Chapter 4

Concatenated LDPC-TCM Error Correction Coding

In this chapter, we investigate efficient and high-performance error correction mechanisms to improve the reliability of NAND flash memories. Based on the memory channel that is modelled with pulse amplitude modulation (PAM) plus Gaussian noise, we propose a concatenated TCM and LDPC coding scheme to better tolerate the errors produced during flash memory programming and retention. We demonstrate that with the coded modulation, storage reliability can be increased with the same signal-to-noise ratios (SNR). Furthermore, by performing the maximum *a posteriori* probability (MAP) decoding, we obtain soft information from TCM demodulator that can be utilized for LDPC decoding. Compared to flash memory with traditional hard decision ECC, significant performance improvement of the concatenated system is observed.

The rest of this chapter is organized as follows. The first section describes the importance of using error correction codes and obtaining the soft decisions from the flash channel outputs. Section 4.2 summarises the preliminaries and related work for understanding the topics in the forthcoming sections. In Section 4.3, the proposed LDPC-TCM coding scheme is described and elaborated in detail. Furthermore, we analyse the theoretically achievable asymmetric coding gain of the concatenated system and derive a mathematical equation for calculating this performance parameter of TCM

in flash memory system. Section 4.4 provides simulation results demonstrating the benefits of the proposed mechanism. Conclusions and challenges are drawn in Section 4.5.

4.1 Problem Description

It was discussed in previous chapters that technology scaling makes flash memory cells susceptible to increasingly severe noise and distortion, which would largely degrade the NAND flash memory storage reliability and performance. As a result, increasingly powerful and sophisticated error correction and signal processing capabilities become indispensable for future NAND-based memory systems. In particular, researchers have recognized that the conventional BCH codes, which are being used in all the commercial NAND flash memories today, become inadequate to handle continued technology scaling. BCH codes are simple algebraic codes but very popular in current design practise because the NAND flash systems only provide hard information to its decoders. As a member of cyclic codes, BCH codes are more attractive than its other counterparts since it can be implemented by using high-speed shift register encoders and decoders. However, this type of code is not perfect but its error-correcting capability is limited. The error-correcting capability of a BCH code is commonly represented by t, the maximum number of errors the code can correct. If a codeword is corrupted with more than t errors and moves to the right sphere of another codeword, the error correction becomes extremely difficult or even impossible. If more than t errors happen in the transmission, this weakness of BCH codes would result in the decoding failures and the codewords keep uncorrected. Nevertheless, it is also a good indication for the system designers to understand that more than t errors have occurred in the target message.

As the storage density of MLC flash increases however, there is a growing need for more advanced ECC techniques. Due to their superior error correction capability and recent success in commercial hard disk drives, LDPC codes have attracted much attention [23, 48, 60, 80, 81] and are seriously considered to be the choice of ECC for

future memory systems development. Due to their capability to approach the Shannon limit in the additive white Gaussian noise (AWGN) channel, the LDPC codes become famous and popular in the digital communication applications. Decoding the LDPC codes involves a iterative process called message propagation [9] using the probability information attached to each bit in the codeword as the inputs. For this reason the error correction performance of LDPC codes relies to a great extent on the accuracy of the probability information that is generated by the NAND flash read and sensing mechanism. The n-bit-per-cell NAND flash is realized by moving the threshold voltages of memory cells to 2^n non-overlapping storage levels, as discussed previously. Flash memory read operations aim to sense and digitally quantise the threshold voltage of each memory cell. If memory sensing uses only one quantization level between two adjacent storage states, it is called $hard-decision$ memory sensing; if more than one quantization levels are used between two adjacent storage states, it is $soft-decision$ memory sensing. Although LDPC code decoding with hard-decision memory sensing can achieve a coding gain over conventional BCH codes, soft-decision memory sensing can significantly improve the strength of LDPC code decoding error correction. The disadvantage for the hard decision is that it takes into account only a fixed set of possible values while ignoring some extra information which also indicates the reliability of each input data point. In contrary, the soft decisions consider the extra information as well thus form better estimates of the original data. Hence, we believe the soft information extraction issue is a major roadblock for us to continue increasing the capacity of flash memory error correction and decreasing the price of NAND based flash memory systems subject to a high reliability.

Widely used in modern digital communication systems, TCM considers multi-dimensional multi-level signal modulation and convolutional error correction code construction together in order to reduce the bit error rate with very low complexity. Given the nature of multi-level storage per cell, MLC flash may also benefit from using the TCM design principle. A TCM-only approach for MLC NOR flash memories was proposed by Fei et al. [62]. Although TCM itself cannot ensure a very low page error

rate for NAND flash memories because of the large page size, it may efficiently reduce the bit error rate and hence enable the use of a weaker error correction code to achieve the target page error rate. One of the attractive ECC schemes is to use LDPC coding under the condition that TCM is implemented in flash memories. To this end, the TCM module equipped in the flash memory read circuits will be able to provide soft decisions.

This thesis studies the flash memory and digital communication channel, and compares the differences in both the transmitter and receiver. Motivated by the excellent performances of TCM and LDPC in digital communication applications, we propose to apply TCM and ECC together in SSD to deal with the data storage errors caused by technology scaling. Considering the different characteristics of the NAND flash memory channel, we demodulate the TCM symbols with the BCJR algorithm to generate soft information for the following ECC decoder. We analyse calculation of such soft decisions mathematically and come up with two formulas for this purpose. In our design therefore, LDPC coding and decoding can be used as a more advanced ECC compared to traditional designs equipped with BCH or Hamming codes. Similar to the analysis in [69], we also investigate the coding gain achieved in our design and conclude that trellis coded modulation can offer a coding gain in the flash memory channel as well. Both TCM and LDPC contribute to correcting the data errors, thereby improving the reliability for SSDs significantly. This proposed scheme is simulated to demonstrate its performance in different aspects.

4.2 Background

This section presents the basics of TCM and LDPC codes, and demonstrates the significant benefits to be gained by using coded modulation and soft-decision LDPC coding. Furthermore, by studying the decoding schemes of TCM, we show that the maximum a posteriori algorithm can generate soft decisions with the channel outputs. Next we review the related research work in applying digital communication techniques to the error correction of flash memories, especially the tentative uses of coded modulation in flash channels. This preliminary work motivates us to apply the TCM technique in

memory systems for the extraction of soft information and use the state-of-the-art LDPC codes for the error correction of NAND flash memory.

4.2.1 Basics of Trellis Coded Modulation

TCM is an attractive solution to the band-limited channels encountered in telecommunications. This technique evolved in the late 1970's when Ungerboeck [68] addressed the issue of bandwidth expansion by combining coding and modulation. According to him, "redundancy" is provided by using an expanded signal set and the coding is done directly on the signal sequences. TCM is highly efficient in terms of high coding gain and the variability on data rate by coding and error correction, and especially efficient for multilevel modulation.

In simplest terms, TCM is a combination of coding and modulation, where the word trellis stands for the use of trellis (also called convolutional) codes [50, 56]. It uses ideas from modulation and coding as well as dynamic programming, lattice structures and matrix mathematics. Communication theory says that it is best to design codes in long sequences of messages. The allowed sequences should be very different from each other. The receiver can then make a decision between sequences on the basis of their statistics rather than symbol-by-symbol. When decoding in this way, the probability of error is an inverse function of the sequence length. Considering the sequences that are transmitted in an AWGN channel, the probability of error between sequences is given by

$$p_e \approx e^{-d_{min}^2/2\sigma^2} \tag{4.1}$$

where d_{min} is the Hamming Distance between sequences and σ^2 is the noise power. We measure the performance of TCM by asymptotic coding gain (ACG), which is the gain obtained over some baseline performance at high signal to noise ratio (SNR) in a Gaussian environment. Figure 4.1 shows the functions of a general TCM, which consists of a trellis code and a constellation mapper. TCM combines the functions of a convolutional encoder of rate $R = \frac{k}{k+1}$ and a M-ary signal mapper that maps $M = 2^k$ input points into a large constellation of $M = 2^{k+1}$ constellation points.

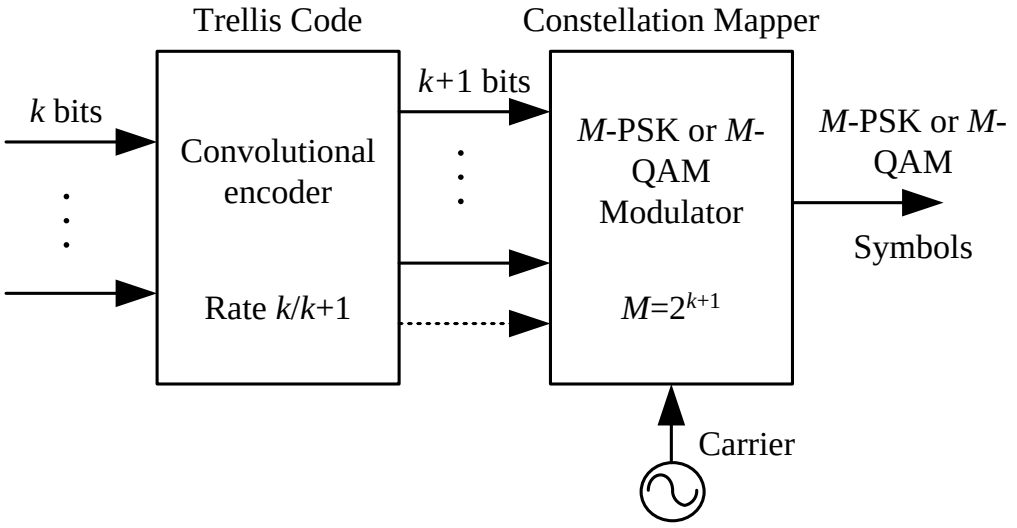


Fig. 4.1 A general trellis coded modulation

Note that the coding adds just one extra bit to the symbol bit size. The symbol size increases from k bits to $k+1$ bits. If coding increases the bit rate by one extra bit, then we need to double the constellation size to accommodate this bit. The input signals to the TCM encoder are referred to as the uncoded sequences while the sequences output are referred to as the coded signals.

A straight line distance between any two constellation points is called Euclidean distance. The coding gain is defined over the Euclidean distance and the average symbol energy, which is given by

$$\gamma = \frac{d^2_{free/coded}}{d^2_{min/uncoded}} \frac{E_{s/coded}}{E_{s/uncoded}} \tag{4.2}$$

where $d_{free/coded}$ is the coded sequence Euclidean distance and $d_{min/uncoded}$ is the minimum distance between the sequences in the uncoded constellation. To determine $d^2_{free/coded}$, we normally calculate the total minimum squared Euclidean distance (SED) by following the minimum distance path in the trellis diagram. Mapping is an important consideration in designing the TCM, which directly influences the coding gain achieved. In terms of 8-PSK coded signals under the natural mapping, the coding gain is 3.6 dB based on (4.2). Gray mapping is not helpful here to improve the coding gain. Some

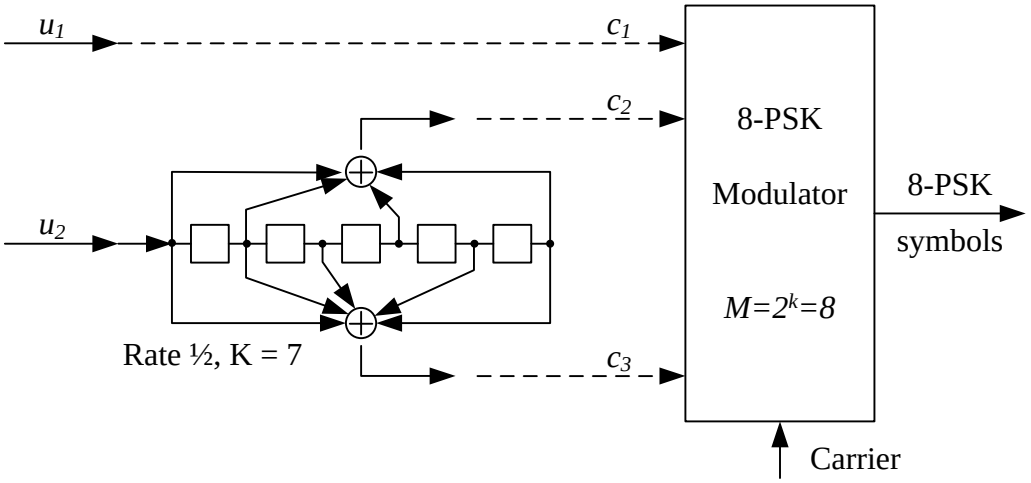


Fig. 4.2 Pragmatic TCM uses a standard rate 1/2 convolutional code.

researchers attempted to improve the coding gain by adding a code of rate 2/3 in front of an 8-PSK modulator, which however was still demonstrated as not beneficial.

Ungerboeck had an idea to improve upon this code by mapping the signals in a special way called set partitioning. The basic idea is to map k information bits to 2^{k+1} constellation points such that we can limit the transitions to occur only along the largest SED. As the subsets are partitioned, the signals get further apart increasing the Euclidean distance between the signals in that set. Each version of TCM as created by Ungerboeck requires a different rate code. To avoid this problem, Viterbi [69] suggested the use of the 64 state standard rate $1/2$ convolutional code which had been a de facto standard in communications for quite a while. The use of this code and not the custom codes of the original concept is called the pragmatic approach to TCM. The pragmatic approach uses the common 1/2 rate convolutional encoder (which can be punctured to provide all other rates) and can create the QPSK, 8PSK and 16QAM signal TCMs in the communication applications. Figure 4.2 is an example of pragmatic TCM for generating QPSK transmission symbols. The idea is similar to the NAND flash memory TCM that is implemented in the design proposed in this chapter.

The TCM we have been talking about so far is called two-dimensional schemes. Trellis-coded modulation schemes using four-, eight-, or 16-dimensional constellations have a number of potential advantages over the usual two-dimensional schemes: a

smaller constituent two-dimensional constellation, easier tolerance to phase ambiguities, and a better trade-off between complexity and coding gain. A method of increasing the dimensionality of TCM ($L > 1$) was proposed by Wei [75]. The term L denotes L dimensions of 2D signals. With $L = 1$ we transmit one symbol only and two symbols with $L = 2$, so the number of symbols transmitted is equal to L. The main concept of multi-dimensionality is to increase the number of symbols created in one processing period. The transmitted symbols are generated together and this co-generation creates dependence and allows better performance. There are a few advantages for the multi-dimensional trellis-coded modulation schemes. Firstly, the information rates could be flexible and even fractional in transmission. Take the 8-PSK TCM signal as an example, instead of the effective code rate being $2/3$, we can increase the rate to $5/6$, $8/9$ or $11/12$ by transmitting more than one symbol. Secondly, it is possible to achieve better bit efficiency. The bit efficiency is referred to as the number of input information bits divided by the number of symbols transmitted in one processing period. Thirdly, the multi-D schemes introduce no additional hardware complexity. We can use standard rate $1/2$ codes (punctured) with standard decoding. Multi-D TCM typically uses the pragmatic version of TCM.

BCJR Algorithm

The BCJR algorithm is named after its inventors, Bahl, Cocke, Jelinek, and Raviv. It can be set up to provide the most likely symbol at each time, equivalent to minimizing bit error probability. In the BCJR decoding, the trellis is processed in forward and backward direction, and the LLR for each bit is output at every trellis stage, which is the soft decisions we require. The following discussion presents the basic principles of BCJR algorithm using the example of additive white Gaussian noise (AWGN) channel [21, 39, 52] with variance $\sigma^2 = N_0/2$ common in digital communications.

Let $X_k \in [0, N-1], k \in Z$ denote a possible state of the convolutional encoder at time k. At the receiver side, the probability of a trellis transition from state X_k to X_{k+1} and

the received symbol $\mathbf{y}_k$ is given by

$$p(X_{k+1}, \mathbf{y}_k | X_k) = p(\mathbf{y}_k | X_k, X_{k+1}) Pr(X_{k+1} | X_k). \tag{4.3}$$

Here $p(\mathbf{y}_k | X_k, X_{k+1})$ is the likelihood function of the received symbol $\mathbf{y}_k$ given the transition (X_k, X_{k+1}) and $Pr(X_{k+1}|X_k)$ is the transition's *a priori* probability. For convolutional codes, there are c code symbols along a trellis branch and thus $\mathbf{y}_k = (y_{0,k} \cdots y_{c-1,k})$. For TCM codes, there are subsets along the branches. These subsets consist of two-dimensional signals and $\mathbf{y}_k$ is a two-dimensional signal.

On the previous assumption of AWGN channel, the likelihood function becomes

$$p(\mathbf{y}_k | X_k, X_{k+1}) = \frac{1}{\sqrt{\pi N_0}} exp\left(-\frac{|\mathbf{y}_k - \mathbf{c}_k|^2}{N_0} \right). \tag{4.4}$$

One can take the logarithm of Equation (4.4) and scale with $-N_0$ to yield the branch metric (BM)

$$\begin{aligned} \lambda(X_k, X_{k+1}) &= -N_0 \log p(X_{k+1}, \mathbf{y}_k | X_k) \\ &= |\mathbf{y}_k - \mathbf{c}_k|^2 - N_0 \log Pr(X_{k+1}|X_k) - N_0 \log \frac{1}{\sqrt{\pi N_0}} \end{aligned} \tag{4.5}$$

The first term in Equation (4.5) corresponds to the squared Euclidean distance between the received symbol $\mathbf{y}_k$ and the expected symbol $\mathbf{c}_k$ along the branch (X_k, X_{k+1}). The second term is the weighted *a priori* probability of the branch. The constant $-N_0 \log \frac{1}{\sqrt{\pi N_0}}$ can be neglected in the calculations since it contributes equally to all BMs.

In the viewpoint of BCJR, data symbols are encoded blockwise, and the trellis has M stages if each block contains M data symbols. The log-likelihood ratio (LLR), which is a measure of bit u_k being 0 or 1, is defined as

$$\mathrm{LLR}(u_k) \equiv \log \frac{\Pr(u_k = 1 | \mathbf{y}_k)}{\Pr(u_k = 0 | \mathbf{y}_k)} \tag{4.6}$$

It encapsulates both hard and soft information. The sign of Equation (4.6) corresponds to the hard decision, and the magnitude is the reliability estimate. For convenience, we

use a notation defined by Matthias [40], which is

$$\begin{aligned} \max^*(x,z) &\equiv \log(e^x + e^z) \\ &= \max(x,z) + \log(1 + e^{-|x-z|}) \end{aligned} \tag{4.7}$$

The BCJR algorithm supposes the encoding process is a Markov chain and the channel is memoryless; thus, numerator and denominator in Equation (4.6) can be divided into three terms each: forward and backward state metrics (SMs), and the BM for the respective state transitions. Forward SMs α are recursively calculated for $k = 0, \cdots, M-2$ according to

$$\alpha(X_{k+1}) = \max^*_{(X_k, X_{k+1})} \{\alpha(X_k) + \lambda(X_k, X_{k+1})\}. \tag{4.8}$$

where $\alpha(\mathbf{X}_0) = [0, -\infty, -\infty, \cdots]$. That is, at time 0, the metric for state 0 is initialized to 0, the remaining $N-1$ SMs to $-\infty$. Equivalently, the backward SMs for $k = M-1, \cdots, 1$ are

$$\beta(X_k) = \max^*_{(X_k, X_{k+1})} \{\beta(X_{k+1}) + \lambda(X_k, X_{k+1})\}. \tag{4.9}$$

and $\beta(X_M) = [0, -\infty, -\infty, \cdots]$. Together with the BM $\lambda(X_k, X_{k+1})$ from Equation (4.5), we get the LLR as the difference between soft estimates for branches $u_k = 1$ and $u_k = 0$, respectively:

$$\begin{aligned} \mathrm{LLR}(u_k) = &\max^*_{(X_k, X_{k+1})_{u_k=1}} \{\alpha(X_k) + \lambda(X_k, X_{k+1}) + \beta(X_{k+1})\} \\ &- \max^*_{(X_k, X_{k+1})_{u_k=1}} \{\alpha(X_k) + \lambda(X_k, X_{k+1}) + \beta(X_{k+1})\}. \end{aligned} \tag{4.10}$$

Note that if $\max^*$ is replaced with max in Equations (4.8-4.10), i.e., the additional offset in Equation (4.7) is omitted, log-MAP turns into the suboptimal max-log-MAP algorithm.

4.2.2 Low Density Parity Check Codes

Low-density parity check codes were developed by Gallager [20] in the early 1960's. An LDPC code is defined as the null space of a parity check matrix **H** with the following structural properties: (1) each row consists of ρ "ones"; (2) each column consists of η "ones"; (3) the number of "ones" in common between any two columns, denoted ζ, is no greater than 1; (4) both ρ and η are small compared to the length of the code and the number of rows in **H**. Since ρ and η are small, **H** has a small density of "ones" and hence is a sparse matrix.

The class of LDPC codes have held the attention of coding theorists in the past decade not only because of their near-capacity performance on data transmission and storage channels, but also because their decoders can be implemented with manageable complexity. In addition to introducing LDPC codes, Gallager also provided a decoding algorithm that is typically near optimal. Since then, other researchers have independently discovered several related algorithms, albeit sometimes for different applications. The class of decoding algorithms are collectively termed "message passing" algorithms since their operation can be explained by the passing of messages in graph-based model of LDPC codes. Generally, message passing decoders use soft reliabilities about the received bits; conversely, a quantisation of the received information or hard decisions can degrade the performance of an LDPC code.

Let C be a binary (N,K) LDPC code specified by a parity-check matrix H with M rows and N columns. A very useful way of representing LDPC codes is the Tanner graph, which is a bipartite graph separated into two partitions. These partitions are called variable nodes (VNs) and parity-check nodes (PNs). The messages passed along the edges in the "message passing" algorithms are probabilities, or beliefs. Figure 4.3 shows the Tanner graph for the (N,K) LDPC code discussed in this section.

Suppose that x_m and y_m, respectively, denote the m-th sample $(1 \leq m \leq N)$ in the transmitted vector, **x**, and in the received vector, **y**, in a Binary Phase-Shift Keying (BPSK) transmission over an AWGN channel with zero mean and a single-sided noise power spectral density of N_0. Let $P_{ml}^{(i)}$ be the probability message from the VN v_m to

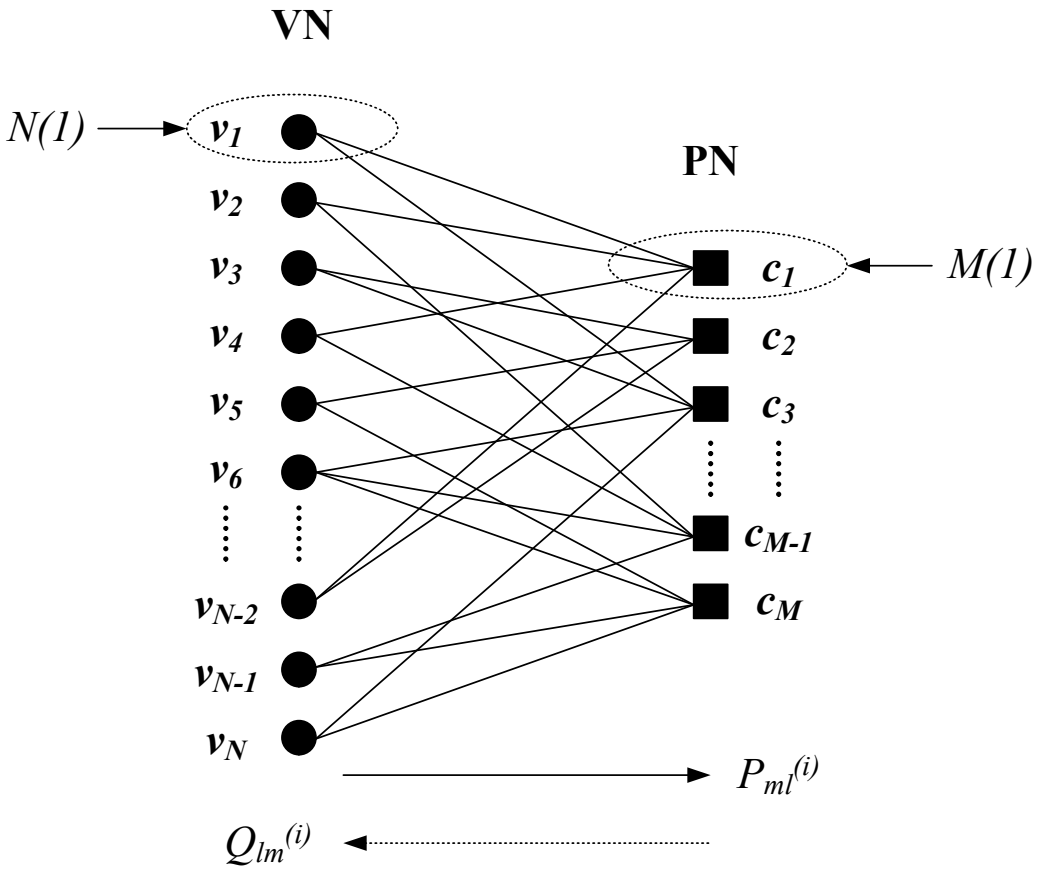


Fig. 4.3 Tanner graph representation for an LDPC code

the PN c_l and $Q_{lm}^{(i)}$ be the probability message from c_l to v_m in the i-th iteration. Also, let $N(m)$ be the set of PNs connected to v_m and $M(l)$ be the set of VNs connected to c_l. The steps for message passing algorithm in the probability domain can be described as follows [54]:

I Set the iteration counter to zero $(i = 0)$.

II For all VNs, i.e., for $1 \leq m \leq n$, $l \in N(m)$, initialize $P_{ml}^{(0)}$ to $P_{CH}^{(m)}$, the *a posteriori* probability (channel probability), computed as:

$$P_{CH}^{(m)} = Pr(x_m = 1|y_m) = \frac{exp(L_{CH}^{(m)})}{exp(L_{CH}^{(m)}) + 1} \tag{4.11}$$

where $L_{CH}^{(m)}$ is the log-likelihood ratio (LLR) of y_m and it is computed as:

$$L_{CH}^{(m)} = log\left(\frac{Pr(x_m = 1|y_m)}{x_m = 0|y_m}\right) = \frac{4y_m}{N_0} \tag{4.12}$$

III Update all the PNs, i.e., for $1 \leq l \leq n-k$, $m \in M(l)$ compute:

$$Q_{lm}^{(i)} = 0.5 - \left(0.5 \prod_{m' \in M(l)\backslash m} \left(1 - 2P_{m'l}^{(i-1)}\right)\right) \tag{4.13}$$

where $m' \in M(l)\backslash m$ denotes the set of VNs connected to c_l excluding v_m.

IV Update all the VNs, i.e., for $1 \leq m \leq N$, $l \in N(m)$ compute:

$$P_{ml}^{(i)} = \frac{P_{CH}^{(m)} \prod_{l' \in N(m)\backslash l} Q_{l'm}^{(i)}}{\left(P_{CH}^{(m)} \prod_{l' \in N(m)\backslash l} Q_{l'm}^{(i)}\right) + \left((1 - P_{CH}^{(m)}) \prod_{l' \in N(m)\backslash l} (1 - Q_{l'm}^{(i)})\right)} \tag{4.14}$$

V For all VNs, i.e., $1 \leq m \leq N$, $l \in N(m)$ compute P_{ext}^{m} as:

$$P_{ext}^{m} = \frac{P_{CH}^{(m)} \prod_{l' \in N(m)} Q_{l'm}^{(i)}}{\left(P_{CH}^{(m)} \prod_{l' \in N(m)} Q_{l'm}^{(i)}\right) + \left((1 - P_{CH}^{(m)}) \prod_{l' \in N(m)} (1 - Q_{l'm}^{(i)})\right)} \tag{4.15}$$

Make the hard-decision to obtain the estimated vector, $\hat{\mathbf{x}} = (\hat{x}_1, ..., \hat{x}_N)$, where $\hat{x}_m = 1$ if $P_{ext}^{m} > 0.5$, and $\hat{x}_m = 0$, otherwise.

VI Terminate decoding if $\hat{\mathbf{x}}\mathbf{H}^{\mathbf{T}} = 0$ or if i has reached the maximum number of iterations. Otherwise, set $i = i + 1$ and return to step III.

Because of the high hardware complexity of operations of VNs and PNs in the probability domain, the message passing algorithm is usually implemented in the log-domain where channel probabilities are considered as LLRs.

4.2.3 Related Work

Unlike wireless communication channels, NAND flash memory channels provide hard decisions, and the cell threshold voltages must be sensed and quantised as the channel outputs. To realize the maximum benefits of LDPC coding, the decoder requires the soft decisions that contain as much of the information to decide the original data. For this reason, many researchers are trying to propose solutions for the extraction of soft information in flash memory [16, 28, 31, 70–74, 76, 78, 83].

The goal of extracting soft decisions is to obtain the LLR values for the stored bits based on the quantised channel outputs. One of the most common approaches is called "read retry" in which cell threshold voltage levels on floating-gates are sensed multiple times with different reference voltages. The "read retry" provides an equivalent discrete channel with a larger output alphabet than binary, and the mutual information between the input and output of the channel depends on the choices of the reference voltages. On the basis of this principle, Wang et al. [70, 71, 73] show that LDPC codes over the equivalent channel provide significant performance improvements when the reference voltages are chosen to maximize the mutual information. In spite of the performance improvements, this technique usually results in a long latency and reduces throughput since the same user data must be read multiple times.

Another approach to achieve the goal is to characterize the flash memory channel using probability theory to obtain the distributions for the MLC storage levels. Dong et al. [16, 76] and Lee et al. [31] propose the channel model with threshold voltage distribution for each level, respectively. With the probability density functions and the sensed channel outputs, we can calculate the soft values for stored bits and apply

LDPC decoding. The techniques are excellent but the common problem is that only part of the noise and interference sources are considered in the channel modelling due to the complicated mathematical computation. This implies that the models proposed by these researchers can be applied in limited occasions where the corresponding noise and interference are dominant.

In addition to application of LDPC codes, several studies have also attempted to apply TCM to flash systems in recent years. Lou and Sundberg were among the first to use coded modulation in multilevel memories, but they did not consider outer ECC and sensing quantization [37]. Sun et al. successfully demonstrated that TCM can help to improve the performance, coding redundancy, silicon cost, and operation latency of flash coding systems [62, 63]. Nonetheless, they focused on short Hamming and convolutional codes rather than the advanced error correction codes.

Concatenation of TCM and BCH coding was proposed, considering both the coded modulation and outer codes design [35]. The key idea is that TCM can be good at leveraging the multi-level storage characteristic to reduce the memory bit error rate and hence relieve the burden of outer BCH code, at no cost of extra redundant memory cells. Meanwhile the concatenated coding solution can be applied to single-page programming MLC NAND flash memories in a straightforward manner. However, the Viterbi algorithm was chosen to perform TCM decoding, which still results in hard decisions. In addition, 5-level MLC flash memory considered in the work is not readily available in the current market.

In this thesis, we propose to concatenate the LDPC code with TCM for fault toleration in multi-level cell flash memory systems. We demonstrate that by using TCM properly, its advantage will not only benefit the use of hard decision ECCs but also the use of advanced soft decision codes. To this end, many aspects related to the concatenated solution must be designed carefully, which are discussed in detail in following sections. Compared to flash systems without error correction, with TCM only and BCH-TCM concatenated scheme, the proposed solution demonstrates excellent performance in both coding gain and error correction. The approach for extracting soft decisions from

channel outputs in the proposed design can be used with other advanced error correction codes as well. Given the significant potential for system performance improvement as shown by means of simulations, we believe that such LDPC-TCM concatenation can be a viable alternative to the existing design practice for future MLC NAND flash memories.

4.3 Concatenated LDPC-TCM Coding System

This section elaborates on the idea of the concatenated LDPC-TCM coding approach. We first describe the modules involved in the system and then propose a way to calculate the soft decisions with the memory channel outputs. A mathematical equation is presented for the calculation, and the results can be used to inform the practical design. As SSDs are different from wireless communication systems, the coding gain derived by Viterbi is not applicable to the memory channel. We study this problem and analyse the coding gain in the following section for our design.

4.3.1 System Description

A number of noise and interference were discussed in chapter 2, however we will not consider all these noise sources in this chapter for simplification of the channel model and focus on our design. The probability density function for the variations of threshold voltage in MLC flash memory cell is modelled here by a Gaussian distribution. The parameters such as means and variances in the Gaussian function are obtained through the best-fit to the practical distribution incorporated with those sources of noise and interference [31]. In this work, we assume an i.i.d. (independent and identically distributed) Gaussian threshold voltage for each level of the memory cell, therefore an m-level flash cell is equivalent to an m-PAM communication system with additive white Gaussian noise. This model has been previously used by Wang [70–74] and Lou [37] in their research as well. As an example, the threshold voltage distribution of 2 bits per cell flash memory are depicted in Figure 4.4 which shows four distributions representing

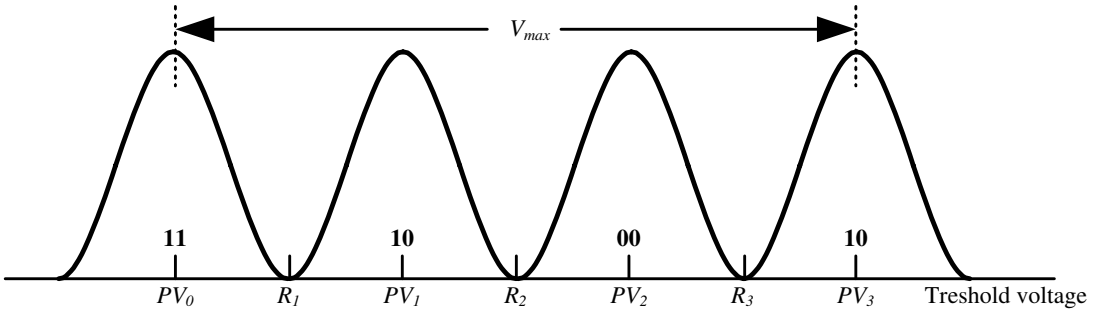


Fig. 4.4 The approximate flash memory cell threshold voltage distribution model.

the memory levels with mean values of PV_i for $i \in \{0,1,2,3\}$ and the same standard deviation of σ.

The block diagram of the proposed coding scheme is illustrated in Figure 4.5. In this figure, we denote the original data stream by $\boldsymbol{b}$ and the output data by $\hat{\boldsymbol{b}}$. The first step processing is the LDPC encoding, which adds some redundancy to the information bit stream based on the specific LDPC code structure. Different constructions of the LDPC codes show very distinct performance. On one hand, the performance of an LDPC code with iterative decoding is measured by the error floor. Performance curves of iterative coding schemes such as LDPC codes are well-known as "waterfall" curves. Sometimes, one observes the bottom of the waterfall, often referred to as the error floor. Therefore, it is important to know how much low the error rate it can achieve while designing an LDPC code. On the other hand, the performance of an LDPC code is determined by a number of structural properties collectively:

1. Minimum distance, i.e., the smallest Hamming distance between any two code-words of the LDPC code;
2. Girth of its Tanner graph;
3. Cycle distribution of its Tanner graph;
4. Graph connectivity;
5. Trapping set configurations and distribution of its Tanner graph;
6. Degree distributions of variable and check nodes of its Tanner graph;

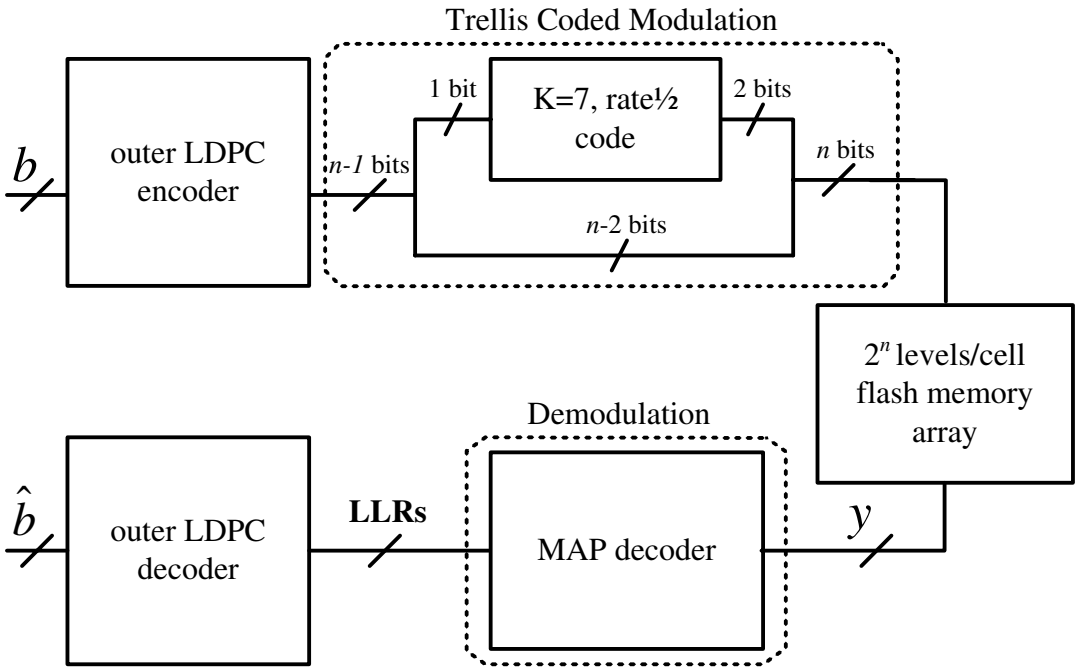


Fig. 4.5 Block diagram of TCM LDPC coding system.

7. Row redundancy of the parity-check matrix.

Before designing the NAND flash memory system, developers should define the target error rate as a systematic parameter for each module included in the system. For the ECC-only error correction scheme, the performance of an LDPC code with good performance is very important to target the defined error rate, which however, is not so strict in our design since the TCM module would correct a number of errors as well. After the encoding process, the resulting bit sequences are passed to the TCM module after serial/parallel conversion. In the TCM module, data bits are encoded again through the convolutional encoder and mapped to the MLC flash memory levels in the way of pragmatic TCM mapping. For ease of practical implementation, we adopt the industrial standard pragmatic approach to TCM. In the design here, $n-1$ encoded bits are to be stored in one memory cell, among which $n-2$ bits are stored directly while 1 bit is fed to a constraint-length 7 (64 states), rate $1/2$ convolutional code. The output n bits are stored in 2^n levels; as a result, the number of storage levels per cell must increase from 2^{n-1} to 2^n.

After the TCM encoding and mapping, the data symbols are programmed to the MLC flash memory levels through the ISPP scheme. This process is different from the

communication channel, where the data symbols are transmitted through the antenna or other media. On the receiver side, in other words, on reading the memory array, the threshold voltage within each MLC cell is sensed and quantized during the reading process, and the quantized value $\boldsymbol{y}$ goes to TCM demodulation and LDPC decoding. The TCM demodulation module employs a MAP decoder to deal with the channel outputs $\boldsymbol{y}$ and generate the LLR values for the data bits. The LLRs are the soft information of the data bits thus they can be used by the outer LDPC decoder to recover the original data stream. The precision of recovered data relies both on the iterative decoding algorithm and the number of iterations. In performance evaluations, we give an example of parameter settings for LDPC encoding, decoding, and code structure. The following section will discuss the way of calculating the soft information by MAP decoder from the theoretical perspective. The comparisons between $\boldsymbol{b}$ and $\hat{\boldsymbol{b}}$ are measured to evaluate the performance of the proposed error correction scheme.

4.3.2 Soft Decisions Calculation

Let X_l denote the trellis state of TCM in Figure 4.5 at time l; the coded 2 bits can be decided by the state transition from X_l to X_{l+1}. For the mapping of pragmatic TCM, 2 coded bits choose a cell voltage level within a subconstellation according to the Gray code, and $n-2$ uncoded bits choose the subconstellation lexicographically, thus there are in total 2^{n-2} parallel transitions for given two coded bits, as shown in Figure 4.6. Consider the pragmatic TCM on 8-level MLC for example, “111", “110", “100", “101", “011", “010", “000" and “001" are mapped to the voltage levels $PV0$, $PV1$, $PV2$, $PV3$, $PV4$, $PV5$, $PV6$ and $PV7$, respectively.

Due to the fact that LDPC decoder only accepts soft information, a demodulation module that can provide soft decisions is required, although the Viterbi algorithm is generally used to perform TCM decoding for the maximum likelihood sequence estimation (MLSE) which results in hard decisions. The BCJR algorithm [1] is employed in the MAP decoder, which generates soft information of each bit, in terms of log-likelihood ratios (LLRs), to the following LDPC decoder.

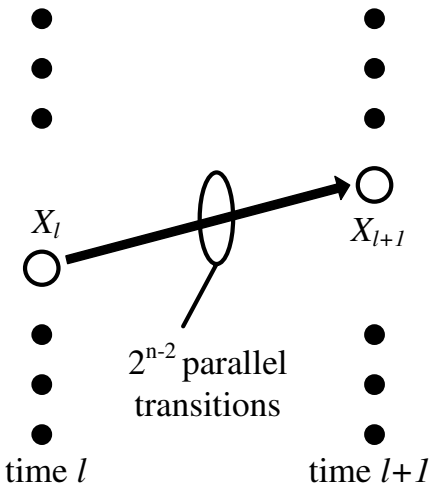


Fig. 4.6 State transitions in pragmatic TCM.

Suppose the demodulator receives $\boldsymbol{y} = (y_0, y_1, \ldots, y_{L-1})$ from each page of the memory block, where y_l is the quantized voltage sensed from one memory cell, and let c_l denote the expected n-bit symbol along with the transition from time l to time $l+1$. For each bit $c_{l,k}, k = 1, 2, ...n$, the LLR is defined as

$$\text{LLR}(c_{l,k}) = \ln\left(\frac{\Pr(c_{l,k}=1|\boldsymbol{y})}{\Pr(c_{l,k}=0|\boldsymbol{y})}\right) \tag{4.16}$$

As a demodulated noisy value y_l is received from the flash channel with AWGN noise with variance σ^2, the likelihood function becomes

$$p(y_l|X_l, X_{l+1}) = \frac{1}{\sigma\sqrt{2\pi}} exp\left(-\frac{|y_l - c_l|^2}{2\sigma^2}\right) \tag{4.17}$$

The BCJR algorithm finds the *a posteriori* probabilities using the forward and backward recursions, respectively, as:

$$\alpha(X_{l+1}) = \sum_{X_l} \alpha(X_l)\gamma(X_l \to X_{l+1}) \tag{4.18}$$

$$\beta(X_{l+1}) = \sum_{X_{l+1}} \beta(X_{l+1})\gamma(X_l \to X_{l+1}) \tag{4.19}$$

where the values for $\alpha(X_0)$ and $\beta(X_L)$ should be determined according to the initial conditions, and $\gamma(X_l \to X_{l+1})$ is the branch metric that is given by

$$\gamma(X_l \to X_{l+1}) = \begin{cases} \Pr(X_{l+1}|X_l)p(y_l|X_l, X_{l+1}), & \text{valid transition;} \\ 0, & \text{invalid transition.} \end{cases} \tag{4.20}$$

The first term in the above equation corresponds to the *a priori* probability of the transition, $(X_l \to X_{l+1})$, which is known at the TCM encoder. The *a posteriori* probability of c_l is given by

$$\Pr(c_l|\boldsymbol{y}) = \sum_{\Lambda_l(c_l)} \alpha(X_{l+1})\beta(X_l)\gamma(X_l \to X_{l+1}) \tag{4.21}$$

where $\Lambda_l(c_l)$ denotes the set of the state transitions, $(X_i \to X_{i+1})$, for given c_l. With the calculated probabilities of all the symbols in the trellis, we can further obtain the *a posteriori* probability of each bit and the corresponding LLRs.

4.3.3 Theoretical Analysis of Coded Modulation in Flash Memory

It is well known that the error performance of TCM systems in terms of SNR is measured by the free Euclidean distances, where SNR is defined as the ratio between the average signal power, E_s, and the average noise power, $2\sigma^2$. Assuming the mean value of the threshold voltage at erase level (PV_0) is 0, the programming voltage at the highest level becomes V_{max} (see Figure 4.4), and the peak power E_p equals V_{max}^2. In the discussions below, both the TCM and non-TCM systems use the same peak power for fair comparisons.

Let M represent the number of levels in a MLC flash memory. The minimum squared Euclidean distances (MSED) which equals the programming voltage difference between two adjacent levels and the mean value of each level can be expressed as

$$d_{min}^2 = \frac{V_{max}^2}{(M-1)^2} \tag{4.22}$$

and

$$PV_i = \frac{V_{max}}{M-1} i, i = 0,1,2...,M-1. \tag{4.23}$$

The average power E_s is calculated as

$$E_s = \frac{1}{M} \sum_{i=0}^{M-1} (PV_i)^2. \tag{4.24}$$

Substituting (4.22) and (4.23) into the equation above, we obtain

$$E_s = \frac{V_{max}^2(2M-1)}{6(M-1)} = \frac{(M-1)(2M-1)}{6} d_{min}^2 \tag{4.25}$$

With the flash channel model presented in Section 4.3.1, the SNR is given by

$$\text{SNR}(dB) = 10\log_{10} \frac{E_s}{2\sigma^2} \tag{4.26}$$

As for the TCM system, the free squared Euclidean distance d_{free}^2 [69] is given by

$$d_{free}^2 = 2(2d_{min})^2 + (d_f - 4)(d_{min})^2 = d_{min}^2(d_f + 4) \tag{4.27}$$

where d_f is a factor that depends on the convolutional codes used, and is equal to 10 for the constraint-length 7 (64 states) convolutional code of pragmatic TCM [69]. Substituting (4.22) into the above equation, d_{free}^2 can be rewritten as

$$d_{free}^2 = \frac{14V_{max}^2}{(M-1)^2} \tag{4.28}$$

The asymptotic coding gains (ACG) [68] (achieved at high SNR) of the proposed system is computed as

$$\text{ACG} = 10\log_{10} \left(\frac{d_{free}^2 E_{s/non-tcm}}{d_{min}^2 E_{s/tcm}} \right) \tag{4.29}$$

Substituting (4.22), (4.25) and (4.28) into the above equation and noting that the values of M for non-TCM and TCM system are 2^{n-1} and 2^n respectively, we obtain

$$\text{ACG} = 10\log_{10}\left(\frac{14(2^{n-1}-1)}{2^{n+1}-1}\right) \tag{4.30}$$

Increasing the number of flash memory cell levels reduces the Euclidean distances between the multiple levels; nevertheless, the trellis coded modulation could offer a coding gain that overcomes such disadvantage and further improve performance over the non-TCM system. In the next section, we will observe this error performance improvement versus the derived SNR.

4.4 Performance Evaluation

For purpose of comparison, we present the asymptotic coding gain achieved for M-PAM modulation [69], which is generally used in wireless communication with a similar model as shown in Figure 4.4.

$$\begin{aligned}\text{ACG} &= 10\log_{10}\left(4\times\frac{M^2-4}{M^2-1}\times\frac{7}{8}\right)\\ &= 10\log_{10}\left(\frac{7(2^{2n-1}-2)}{2^{2n}-1}\right)\end{aligned} \tag{4.31}$$

where n becomes the number of bits per transmitted symbol corresponding to the bits per cell in MLC flash memory. Figure 4.7 shows that the value of ACG increases with n for both M-PAM modulation and MLC flash memories, however, the M-PAM modulation achieves higher gain than flash memory at certain value of n. The reason is because the average energies have been normalized to unity in the transmitter of digital communication systems, while the peak energies rather than average energies are normalized in flash memories. In the market today, 4-level and 8-level MLC are the prevalent flash storage media, so these two types were chosen to be used in the following simulations. As shown in Figure 4.7, for TCM system on 8-level MLC, the asymptotic coding gain can be achieved as much as 4.5 dB in SNR.

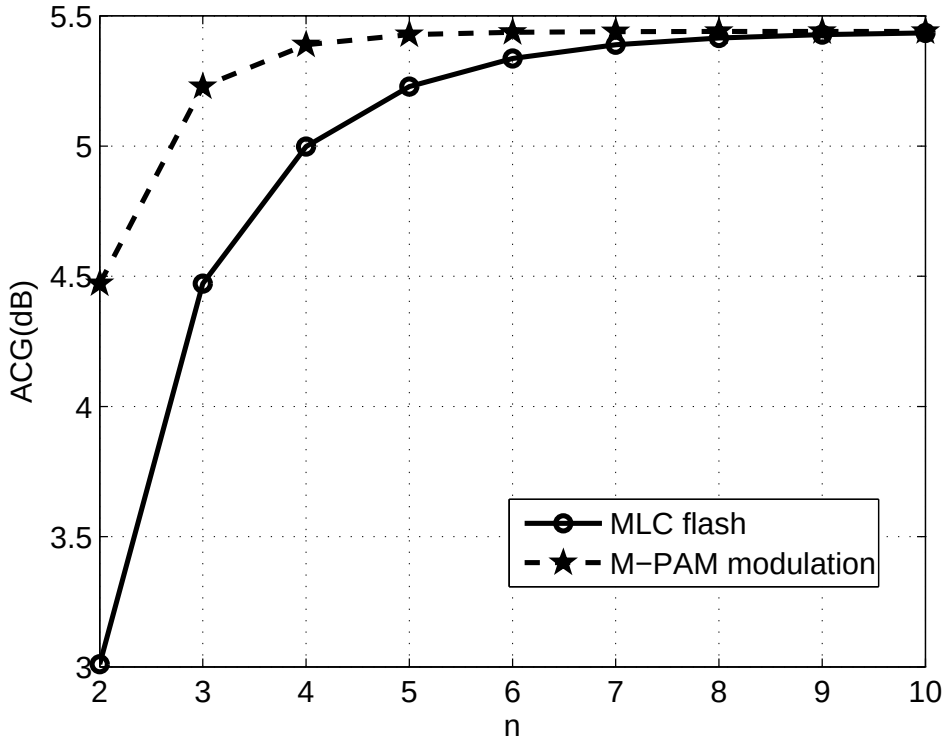


Fig. 4.7 Asymptotic coding gain of TCM for MLC flash memories and M-PAM modulation.

To evaluate performance improvements due to the LDPC-TCM concatenated coding provided in Figure 4.5, we run a series of simulation experiments over different kinds of flash coding systems. The simulations are programmed in the MATLAB environment and the bit and word lines of flash memory (see Figure 2.1) are emulated with the rows and columns of an MATLAB array. Table 4.1 lists the parameters for NAND flash memory applicable to all simulations. Two kinds of NAND flash are modelled in the simulations: 4 levels (2 bits/cell) and 8 levels (3 bits/cell) MLCs. For each kind we suppose the same number of bit lines that is equal to the size of each page. In addition, both of them use the same programming voltage for erase and highest level. Note that the average energy for the individual flash memory cell is not the same to both memories. The following equation gives the way of calculating the average energy.

$$E_s = \frac{V_{max}^2 * (2 * M - 1)}{6 * (M - 1)} \tag{4.32}$$

Table 4.1 Simulations Setup

Page size	4608 bytes (4096 + 320 bytes)
Block size	384 pages (1536K + 120K bytes)
Highest level threshold voltage	2.1 Volts
Erase level threshold voltage	0 Volts
SNR	15 : 0.25 : 20

where V_{max} is the threshold voltage for the highest programming level and M is the number of levels in MLC. Using E_s and the predefined SNR values, we can get the corresponding noise power for every single simulation.

Various error correction schemes are considered in the experiments. BCH coding and standard Gray mapping are applied in the 4 level memory system while LDPC coding and the pragmatic TCM mapping are used in the 8 level memory system. An important setting that we should point out is that the same size of information bits that are randomly generated and stored into these two memory systems. Therefore, the bit error rates (BERs) for the coding schemes in each simulation discussed afterwards are assumed for fair comparisons.

For the error correction codes, we first design a rate-0.927 (17664, 16384) structured LDPC code following the algorithm proposed by He et al. [22] whose parity-check matrix is specified by a triangular plus dual-diagonal form to lower the error floor and encoding complexity. Such class of LDPC code is irregular and has shown clearly low error floor and high throughput encoding, which therefore meet the first requirement for NAND flash LDPC codes discussed in Section 4.3.1. Additionally, the bipartite graph of the LDPC code used has been constructed to be free of cycle-4 with the bit-filling algorithm [6, 7, 14, 84, 85]. As a result, the code performance is further improved by increasing the rate and the minimum girth and the designed LDPC code is now more suitable for the coding of flash memories. Min-sum decoding algorithm [8, 12] is used to carry out LDPC code decoding. For the purpose of comparisons, we also consider a $[n,k,t]$ = [16383, 15200, 85] binary BCH code with the same rate-0.927.

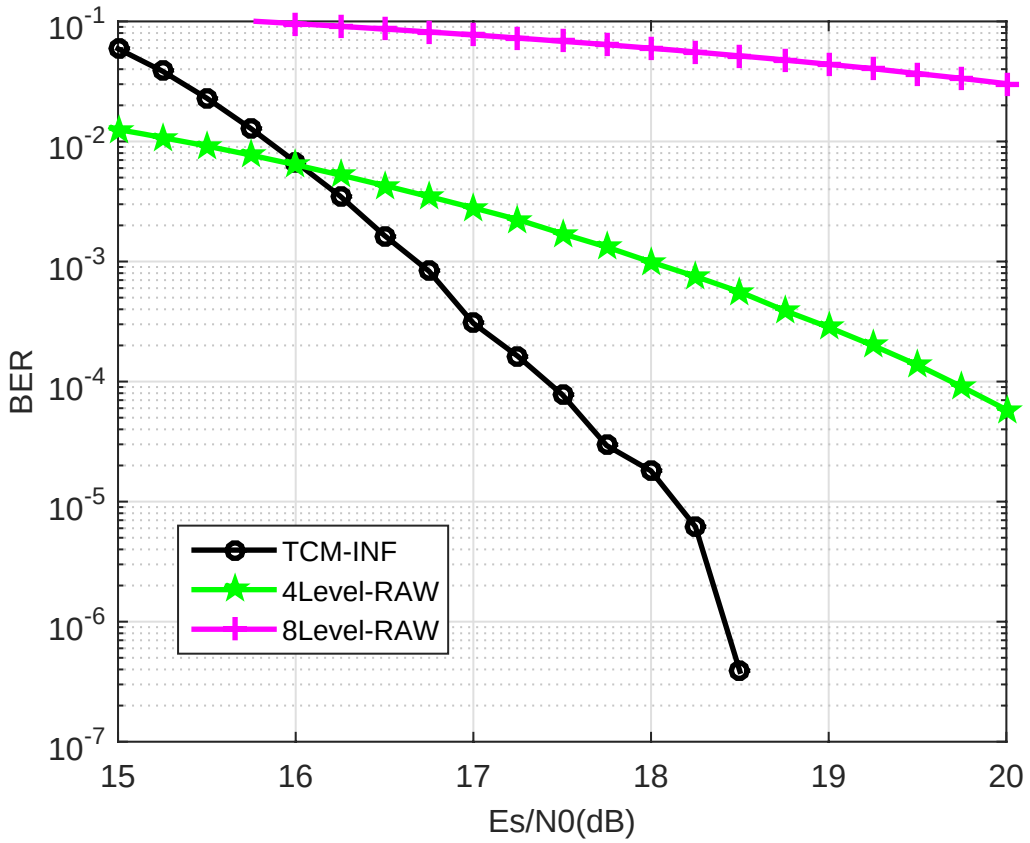


Fig. 4.8 Performance comparisons for the raw BERs of 4 levels, 8 levels MLC and TCM without quantization.

Figure 4.8 shows the raw bit errors in the flash systems without ECCs. The curve labelled as "TCM-INF" represents the TCM simulation without quantizing the flash channel outputs. If neither error correction codes nor TCM is used in NAND flash memory, 3 bits/cell MLC exhibits the worst performance and will generate the most bit errors on reading the data from the array. However, the 3 bits/cell flash can store more data per cell and therefore is able to achieve higher data density than 2 bit/cell and TCM memory systems. Compared to the flash system using 4-level MLC, the coding gain (CG) of TCM flash system using 8-level MLC is about 3.4 dB at the bit error probability of 10^{-6}. For the flash system at very low bit error probability (say less than 10^{-9}), the coding gain will reach the asymptotic coding gain of 4.5 dB that was derived earlier. Since we store the same size of data bits into the 4 levels system and TCM system, we conclude that the error performance of flash system is firstly improved by the use of TCM scheme.

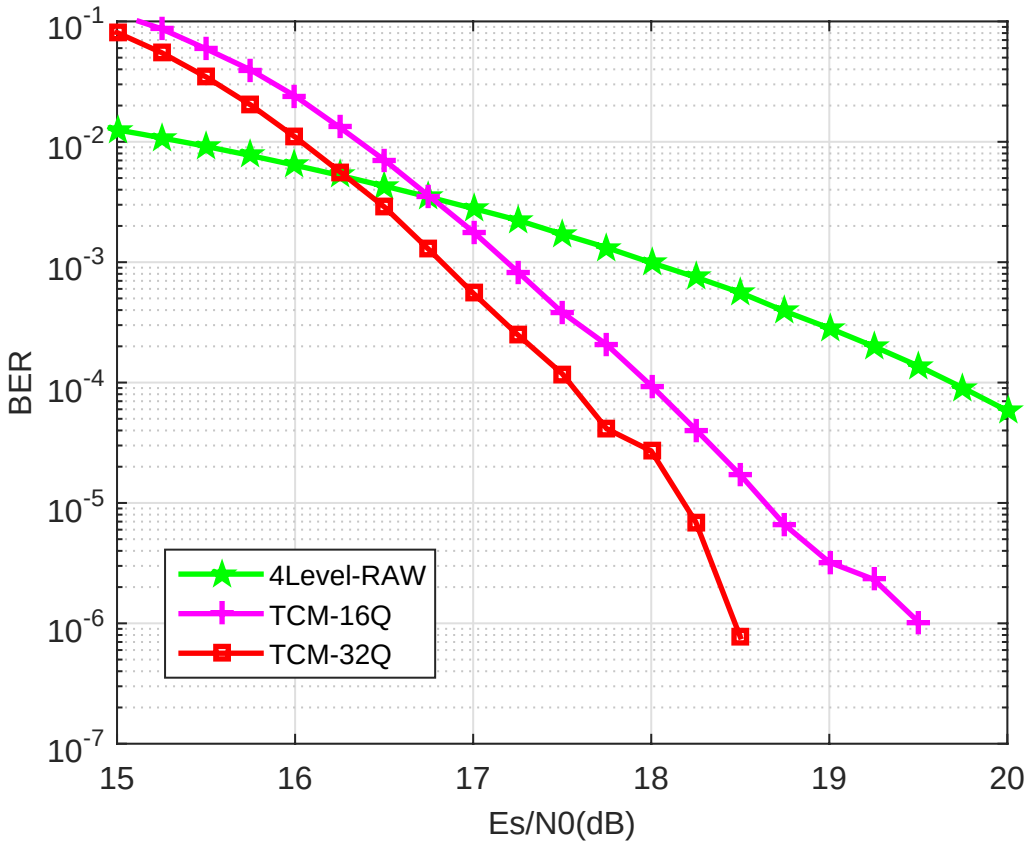


Fig. 4.9 Performance comparisons of the raw BER of 4 levels and the TCM with quantization

Figure 4.9 illustrates the BER of flash systems without ECC but the TCM scheme used in one of the system is decoded with the quantized channel outputs. In practice, the stored information (associated with cell threshold voltages) will be sensed and quantized during the reading so the flash channel can provide the quantized values only. The situation in the last simulation is ideal but not feasible in the practical application. We assume two types of uniform sensing quantization schemes in our simulations: 16 levels and 32 levels, labelled as TCM-16Q and TCM-32Q, respectively. Performance loss has been observed as the channel outputs are quantized more coarsely from 32 to 16 levels. However, both of them still demonstrate substantial performance improvements compared to the raw BER of 4-level flash system. The simulation results prove that the TCM scheme is also beneficial in the practical application of NAND flash memories.

Figure 4.10 depicts the error correction performance comparisons for flash system with BCH coding and systems without ECCs. Though the use of TCM reduces some

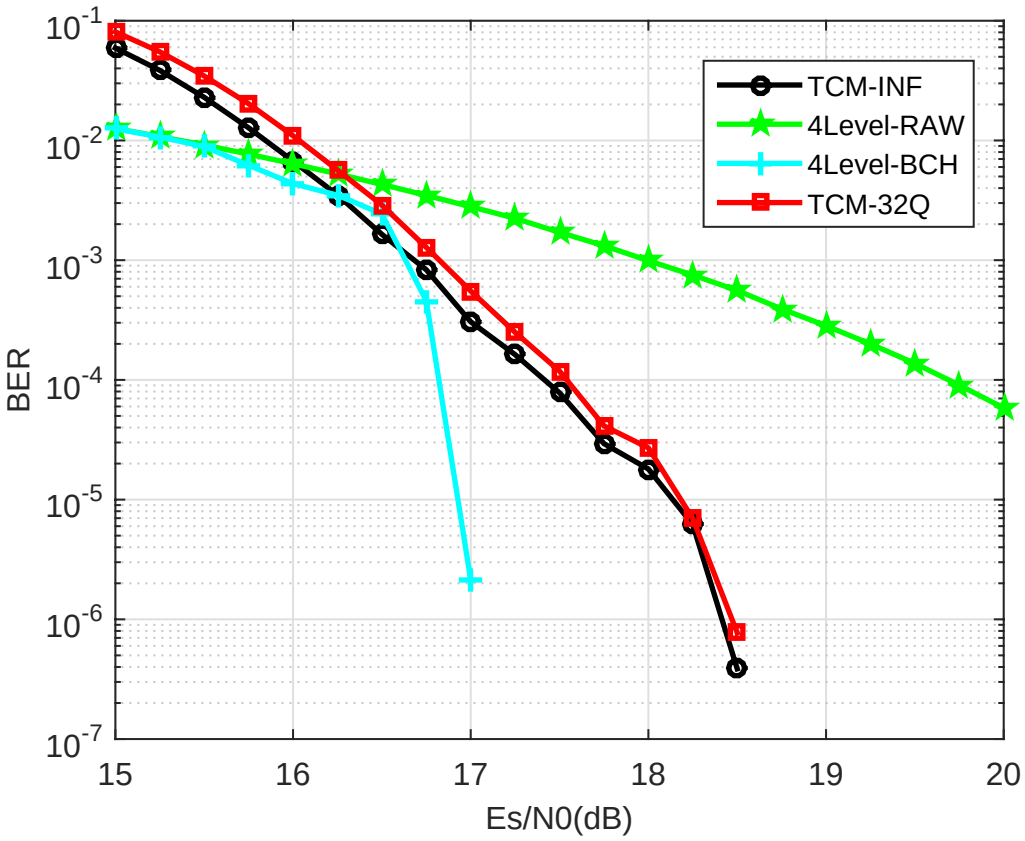


Fig. 4.10 Performance comparisons of the raw BER of 4 levels, the BCH coding and the TCM for NAND flash memories.

errors in flash storage, the application of ECCs, such as rate-0.927 BCH codes in this simulation has shown much stronger error correction performance. At the bit error probability of 10^{-5}, the flash system with BCH coding achieves more than 1 dB coding gain than the TCM flash coding system. Based on that, we believe it is enough to use the TCM only for the flash memory and the BER can be improved even more through the use of ECC codes, which directly motives us to come up with the idea proposed in this chapter, the concatenated LDPC-TCM coding scheme. In this figure we also observe that the 32-level quantization of TCM results in some performance loss to the "TCM-INF" scheme.

Figure 4.11 shows the BER performance of the flash systems with different outer ECCs. BCH coding is adopted in the flash system using 4-level MLC memory. In the experiments for LDPC-TCM coding, we simulated the scheme with channel outputs without quantization and with 32-level quantization. As shown, the proposed

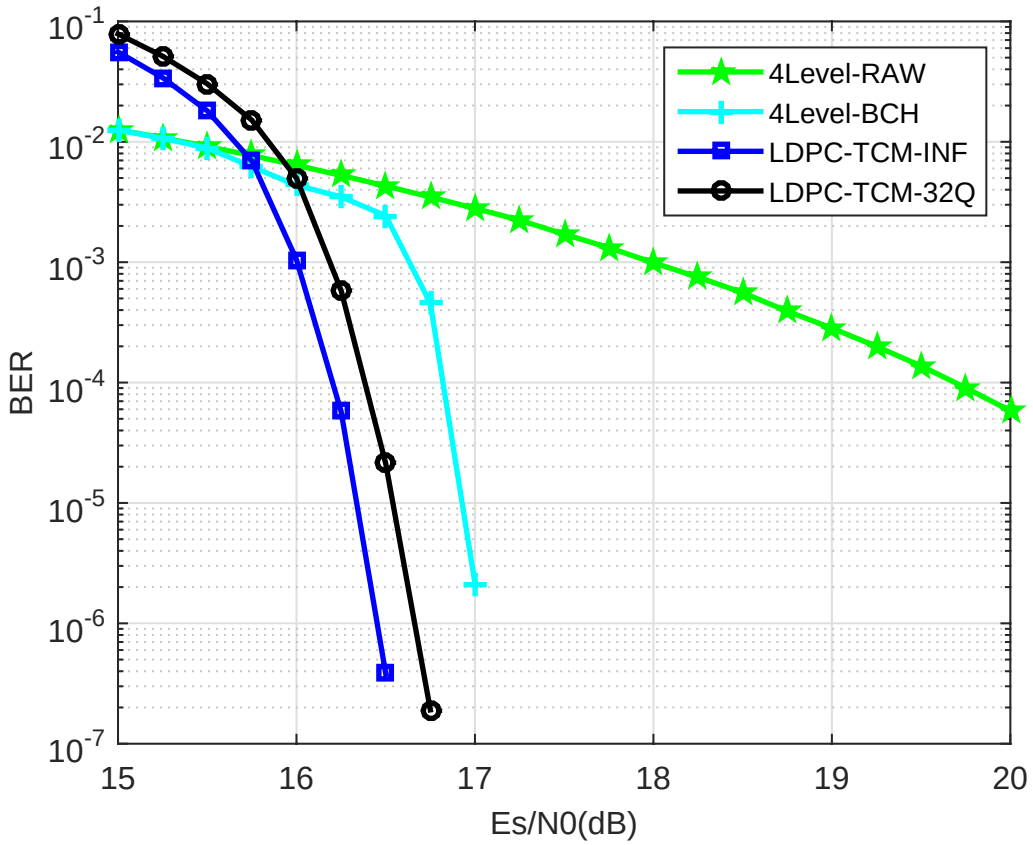


Fig. 4.11 Performance comparisons of the raw BER of 4 levels, the BCH coding and the proposed LDPC-TCM coding.

LDPC-TCM coding provides a remarkable performance contribution (about 0.75 dB improvement over BCH coding at the bit error probability of 10^{-6}) to the error correction of flash systems. Comparing to the simulation results shown in Figure 4.8, 4.9 and 4.10, we can say that the proposed scheme has drawn the advantages from both the TCM and ECCs codes. Since the BCH coding is getting more and more insufficient in most flash memory application, the LDPC-TCM would be an excellent alternative method for the implementation of flash error correction coding. Furthermore, the theoretical analysis and the detailed descriptions of the scheme has already provided guidance to designers in terms of a clear process. The uniform quantization also offers a way of dealing with the channel outputs, although in a similar case as before, some performance loss is introduced due to the quantisation.

The outer ECC of the proposed design can be replaced with BCH coding if the TCM decoder simply outputs the hard reliabilities. In this case, the soft decisions in

Equation (4.16) are generated from the TCM decoder and used to decide the value of bit $c_{l,k}$ in the following way.

$$c_{l,k} = \begin{cases} 1, & \text{LLR}(c_{l,k}) > 0 \\ 0, & \text{LLR}(c_{l,k}) \leq 0 \end{cases} \tag{4.33}$$

The above hard decisions of $c_{l,k}$ are then used for the decoding of outer BCH codes. Apparently the information embedded in the soft decisions will be lost due to such simple judgement, leading to the high BER in reading the data in flash memory. This can be seen in the simulation results shown in Figure 4.12. In this figure, the proposed LDPC-TCM is compared with such BCH-TCM coding system, both of which are simulated with the same memory parameters. Under the condition that no quantization is used in the schemes considered, LDPC-TCM presents the best performance. At the SNR of 16.5 dB, the BER of LDPC coding is close to 10^{-7} while that of BCH coding is 10^{-5} only. The results also demonstrate that the LDPC-TCM scheme makes good use of the soft decisions generated by the BCJR decoder thus is superior to the hard decision based coding schemes.

Figure 4.13 illustrates the BER performance for each scheme while using the quantized channel outputs. The TCM decoder deals with the quantized data and generates soft decisions for LDPC decoder and hard decisions for BCH decoder, respectively. The curves show that the LDPC code outperforms the BCH code under the condition of same TCM parameters and quantization. Additionally, compared to the results shown in Figure 4.11, it has been observed that BCH coding also benefits from the use of TCM if the quantization is not too coarse.

4.5 Summary and Challenges

An error correction scheme of concatenated LDPC-TCM coding for MLC flash memory is proposed in this chapter. The scheme deals with the storage data through two processes: TCM modulation (with convolutional encoding) and LDPC encoding. It also re-organises

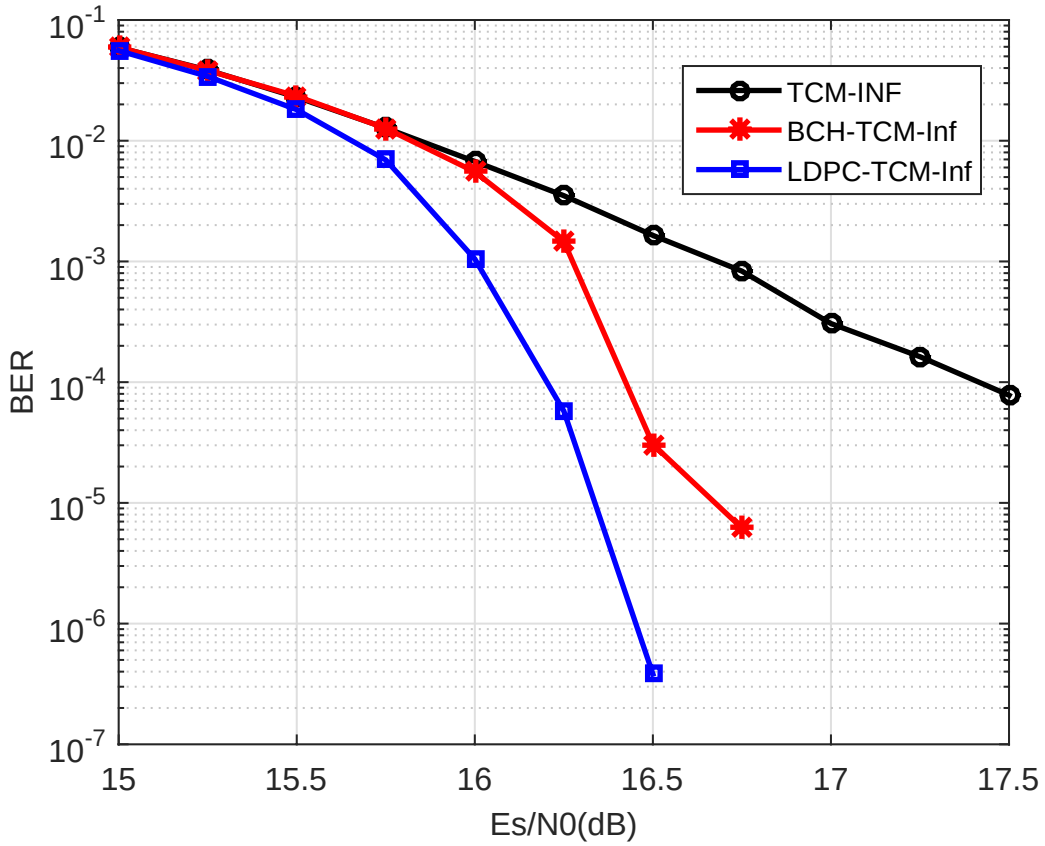


Fig. 4.12 Performance comparisons of the raw BER of 4 levels, the BCH coding and the proposed LDPC-TCM coding.

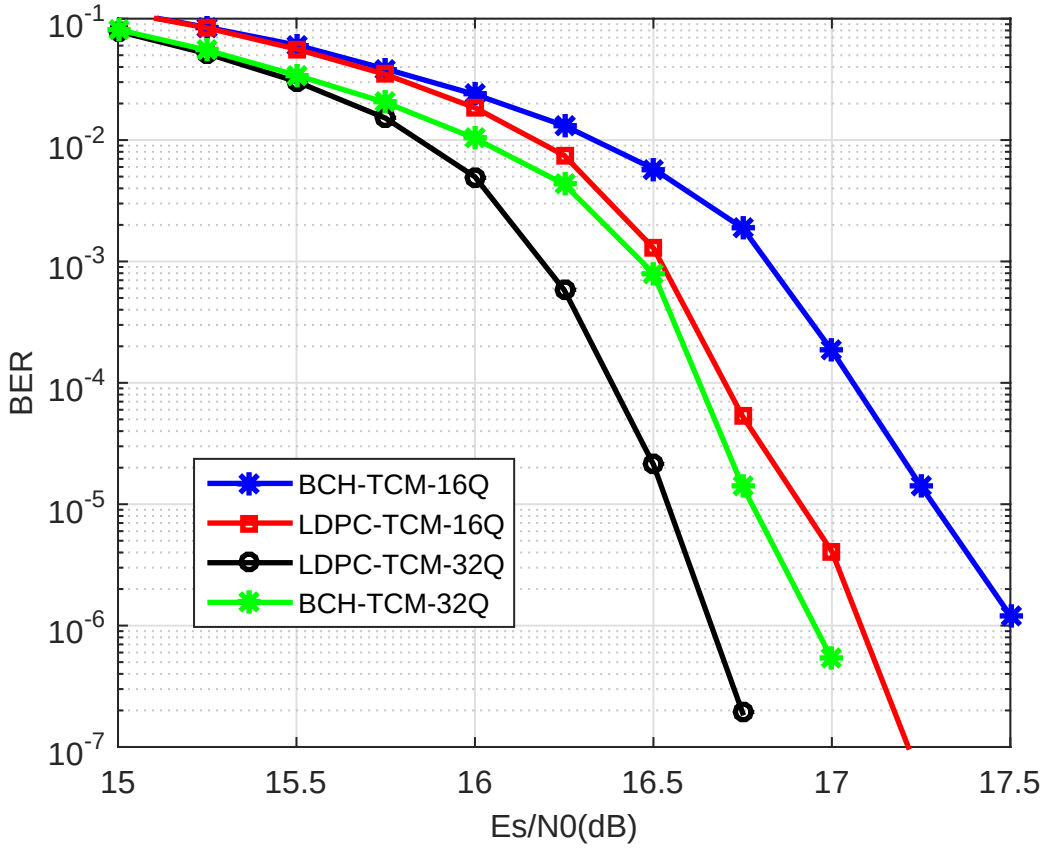


Fig. 4.13 Performance comparisons of the BCH coding and the proposed LDPC-TCM coding with quantized flash channel outputs.

the data bits for the purpose of MLC flash programming through the pragmatic TCM mapping. The proposed design offers the advantages of both the coded modulation and error correction codes.

The idea of the concatenated system has been explained in detail, including the data flow, encoding and decoding, generating soft decisions and deciding the original storage data. Aside from the system descriptions, we also presented a way of calculating soft decisions and analysed the achievable coding gain in theory. Since most of the methods used in this chapter are similar to the principles in wireless communication, we also investigated the ACG of our scheme in a way similar to coded modulation in digital communication. The results were specified for the flash memory application and compared to the M-PAM modulation in digital communication system.

Computer simulations were carefully conducted to demonstrate the benefits of the proposed designs. Compared to various flash coding systems, for example, the one that provides hard-decisions and employs BCH codes only, or the one uses hard decisions but employ both BCH and TCM, results show remarkable BER performance improvements from the system equipped with (17664, 16384) LDPC codes and industrial pragmatic TCM.

From the simulations, it can also be seen how we gradually discover the disadvantages of each scheme and come up with the proposed idea. BCJR algorithm is performed and associated with TCM, which has been demonstrated to be an alternative for converting the hard-decisions into soft-decisions. Using the same way of obtaining soft decisions, we could modify our design to accept other advanced soft information based ECC, such as Turbo codes. To further improve our work, better quantization schemes for memory sensing should be considered while designing the error correction system.

Chapter 5

Modelling and Characterisation of NAND Flash Memory Channels

Earlier we established that the threshold voltage distribution after ideal programming in NAND flash memory cells is usually distorted by a combination of the random telegraph noise (RTN), cell-to-cell interference (CCI), and the retention process. To decide the original bits more accurately using advanced error correction coding under this scenario, a precise flash channel model and soft decisions should be utilized on the basis of the measured threshold voltages. This chapter aims to characterise the various sources of distortion occurring in multi-level cell (MLC) flash memories. The first issue to examine is how these noise and interference sources interact together to develop the channel noise. The probability distribution function for each type of noise is investigated for the study of this issue. The noise distributions lead to insight into the overall probability distribution of channel noise and its influence on the cell threshold voltage. The ultimate aim of this chapter is to propose a set of mathematical formulas for the distributions of overall noise and cell threshold voltages for the analysis and discussion of channel behaviours. The last issue to examine in this chapter is the error correction for NAND flash memory, which is the main topic in this thesis.

The remainder of this chapter is organized as follows. Section 5.1 describes the problem under study and highlights the specific obstacles faced by researchers. Section

5.2 explains the basic operations and noise sources of NAND flash memory. Simulations of the channel behaviour are illustrated as reference for the remainder of the chapter. The basic mathematics and related work are also introduced in this section. In Section 5.3, the influences of various sources of distortion are explained. The main focus is on the probability distributions of the sources of noise and interference; their corresponding characteristic functions are used to investigate the overall channel noise distribution. As a case study, the final probability distribution functions for each symbol in 4-level MLC are found. Simulation results are also presented in this section. In Section 5.4, we study the issue of calculating soft information under the proposed statistical flash model. Conclusions are drawn in Section 5.5.

5.1 Problem Description

The importance of advanced ECCs for NAND flash memories has been discussed in the previous chapters. There are several recent works that employ the excellent LDPC codes or turbo product codes (TPCs) for the error correction of NAND-based memory systems [16, 29, 79]. Also several works on the efficient hardware implementation of LDPC decoding algorithms have been published recently [8, 57, 66].

The LDPC codes, or TPC codes, are soft decision in nature, i.e., their decoding mechanism demands the log-likelihood-ratio (LLR) information of each bit and their error-correction performance depends heavily on the quality and accuracy of soft information. To that end, NAND flash memory must carry out fine-grained soft-decision memory-cell sensing. For example, the threshold voltage of each cell in 2-bits/cell memory is quantized into 4 bits, corresponding to 15 sensing levels. Compared with traditional hard-decision memory sensing, the multiple senses cause longer on-chip memory sensing latency and longer flash-to-controller data transfer latency. In addition, fine-grained memory sensing also results in a larger on-chip page buffer since sensed data should be stored temporarily in an on-chip page buffer, which introduces more silicon cost. Thus, for the application of soft-decision ECC codes, it is critical to reduce the sensing levels while still achieving sufficient error-correction performance.

An intuitive idea is that an accurate memory-cell threshold-voltage statistical distribution model would be beneficial to provide well-developed soft information and thus minimize the fine-grained sensing precision, or in other words the number of memory sensing levels. In chapter 4, we simply used PAM with Gaussian noise to model flash cell threshold voltage since the coded modulation in the proposed concatenated LDPC-TCM scheme can compensate for inaccuracy of the channel model. Meanwhile the simplified channel model is more suitable for the study and investigation of the proposed design. However, it would be important to have a more accurate channel statistical model and precise threshold voltage distributions while using the soft-decision ECCs only for the error correction of NAND-based systems.

Based on experimental measurements, three types of noises have been widely recognised as the dominant sources of memory-cell threshold-voltage distribution distortion in the NAND flash memory channel. These distortion sources include random telegraph noise, cell-to-cell interference and data retention error. The statistical distributions for these noise sources have been characterised by previous researchers through experiments or theoretical analysis, which were described in Chapter 2. The simulation results in the following section clearly show the NAND flash channel behaviours and the changes of cell threshold voltage and its distribution. However, the curves offer only some visual information about the noise and its distribution, and we still do not have a statistical model that can be characterised by mathematical equations. In the absence of a statistical model, we will not be able to calculate the soft outputs mathematically. Therefore, it is critical to derive a reasonably accurate threshold-voltage probability distribution function for memory cells in the presence of these three kinds of distortion sources. However, to the best of our knowledge as elaborated in Section 5.2.3, the modelling and characterisation of flash memory channels have not been addressed in the open literature.

This thesis attempts to fill this missing link. Based upon NAND flash memory erase-and-programming scheme and the statistical characteristics of the noise and interference, mathematical formulas are derived to approximately model the threshold-voltage distri-

bution of memory cells. Furthermore, a formula is derived for the calculation of LLRs based on the mathematical model in the presence of multiple memory sensing. Using the hypothetical 2-bits/cell NAND flash memory model, we carry out computer simulations to evaluate the effectiveness of the developed techniques. The results presented in this chapter are important not only for flash coding study but also for the design of on-chip flash memory error correction.

5.2 Background

This section presents the preliminary knowledge needed for the analysis and mathematical derivations that will follow in subsequent sections. Firstly, we conclude the programming process that has been previously discussed and trace the shifts of cell threshold voltage in the memory channel. The behaviours of the channel elements are illustrated through computer simulations. Secondly, the basics of characteristic functions are introduced briefly to help readers to understand the mathematical derivations presented in this work. Finally, we review the existing work related to this problem and analyse their contributions and disadvantages.

5.2.1 Equivalent Channel Model Incorporated with Major Threshold Voltage Distortions

Chapter 2 described the NAND flash erase and programming from a statistical perspective, and introduced the multiple sources of noise and distortion occurring in the flash memory channel. Considering all the processes together, we could have an overview on all the shifts of the threshold voltages that represent the information stored in the NAND flash memory cells.

To start with, we will erase the block where the cell resides in before writing any data to a memory cell. The threshold voltages after this process follow a Gaussian distribution, $p_e(x)$. After being erased, the memory cell will be programmed to one of the four threshold voltage levels via the ISPP scheme, depending on the specific data to

be programmed. The threshold voltage distributions for the four levels after such ideal programming are different from one another but all following Gaussian distributions, which are represented as $p_i^{(k)}(x), 0 \leq k \leq K-1$.

In the flash channel, the cell threshold voltage will experience a series of sources of distortion and noise, leading to large departures from the results after ideal programming. The noise and distortion include random telegraph noise, cell-to-cell interference and retention noise. Researchers have characterized these sources from a large number of experiments with the NAND flash memory channel. They have come up with mathematical (statistical) formulas for each type of noise.

NAND flash has the ability to store the data without power supply. After a period of time, the information stored in the memory cell will be read out by sensing the cell threshold voltage. The final distributions for the K memory states, $p^{(k)}(x), 0 \leq k \leq K-1$, have been distorted severely after experiencing the noise and interference in the memory channel, which thus are far away from the threshold voltage distributions in the programming and actually beyond recognition. Figure 5.1 illustrates the approximate NAND flash memory device model that incorporates noise and distortion sources discussed above.

Based on this model, we can simulate the memory cell threshold voltage distribution and hence estimate the raw storage reliability. Using the notations in the figure, the final threshold voltage is obtained as

$$\begin{aligned} V^k &= V_i^{(k)} + \Delta V_{RTN} + \Delta V_{CCI} - \Delta V_{ret} \\ &= V_i^{(k)} + \Delta V_{RTN} + \sum_l \Delta V_{t(l)} \gamma_{(l)} - \Delta V_{ret} \end{aligned} \tag{5.1}$$

and the final threshold voltage distribution is obtained by a convolution of the initial threshold voltage distribution and the channel noise distributions, i.e.,

$$p^{(k)}(x) = p_i^{(k)}(x) \otimes p_r(x) \otimes p_t(x) \otimes p_{ret}(x), \quad 0 \leq k \leq K-1. \tag{5.2}$$

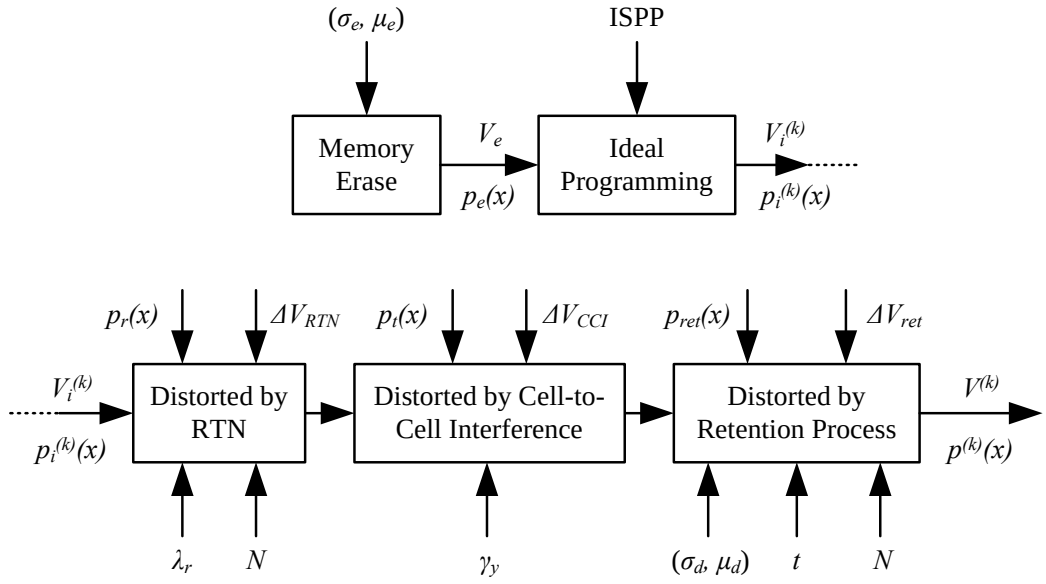


Fig. 5.1 Approximate NAND flash memory device model to incorporate major threshold voltage distortion sources.

The above approximate mathematical channel model for simulating NAND flash memory cell threshold voltage is further demonstrated in the following example. Let us consider 2 bits/cell MLC flash memory with all bit-line architecture. The normalized σ_e and μ_e of the erased state are set as 0.35 and 1.4, respectively. For the three programmed states, we set the normalized program step voltage ΔV_{pp} as 0.2, and the normalized verify voltages V_p as 2.6, 3.2, and 3.93, respectively. For the RTN distribution function $p_r(x)$, we set the parameter $\lambda_r = K_\lambda \cdot N^{0.5}$, where K_λ equals to 0.00025. For the function $\mathcal{N}(\mu_d, \sigma_d^2)$ to capture trap recovery and electron detrapping during retention, we assume that μ_d scales with $N^{0.5}$ and σ_d^2 scales with $N^{0.6}$, and both scale with $\ln(1+t/t_0)$, where t denotes the memory retention time and t_0 is an initial time set as 1 hour. In addition, both μ_d and σ_d^2 also depend on the initial threshold voltage. Hence we set that both approximately scale $K_s(x-x_0)$, where x is the initial threshold voltage, and x_0 and K_s are constants. Therefore, we have

$$\begin{cases} \mu_d = K_s(x-x_0)K_d N^{0.5} \ln(1+\frac{t}{t_0}) \\ \sigma_d^2 = K_s(x-x_0)K_m N^{0.6} \ln(1+\frac{t}{t_0}) \end{cases} \tag{5.3}$$

where we set $K_s = 0.38$, $x_0 = 1.4$, $K_d = 4 \times 10^{-4}$ and $K_m = 4 \times 10^{-6}$ by fitting the measurement data presented by Mielke et al [43, 44].

Based on the discussions in Section 2.2.2, the all bit-line architecture only suffers these three kinds of cell-to-cell interference from three neighbouring cells: two cells in the diagonal directions and one cell sharing the same bit line as the victim cell, which is considered here to be in the vertical direction. For all simulations in this chapter, we set the ratio between the means of γ_y and γ_{xy} as $0.08/0.006$, and $\omega_\gamma = 0.1\mu_\gamma$, $\sigma_\gamma = 0.4\mu_\gamma$ the for the two parameters in Equation (2.7). Accordingly, we carry out Monte Carlo simulations to obtain the cell threshold voltage distribution at different stages under 1K P/E cycling and with 1-month retention limit, as shown in Figure 5.2. The results clearly show the dynamic characteristics of NAND flash memory.

Since the γ_{xy} is one order of magnitude less than γ_y in the scenario of all bit-line architecture [36, 58, 59], we further simulate a case in which the γ_y is considered only. Figure 5.3 shows the simulation results for the coupling ratio under this situation. Comparing the third distribution in this figure to that of Figure 5.2, the difference is negligible and the influence from γ_{xy} cannot even be observed in the simulation.

In the study of cell-to-cell interference, the coupling ratio can be further approximated as a constant value which would not make much difference compared to real strength for such distortion [16]. The reason for this approximation is that the coupling ratio γ follows a bounded Gaussian distribution with very small value of variance and very narrow bounded distribution region. In this context most of the random values for γ_y are very close to the mean value. The simulation result for this case is depicted in Figure 5.4. As shown, the distribution after CCI in this figure is quite close to that of cases shown in Figure 5.2 and Figure 5.3.

The simulation results above demonstrate the changes of threshold voltages and the influences resulting from the noise and interference. Readers might be able to do a curve-fitting to find a mathematical description of the final threshold voltage. Take the approach proposed by Lee and Sung [31] for instance, the authors model the threshold voltage as Gaussian mixture and attempt to find the best-fit parameters by minimizing

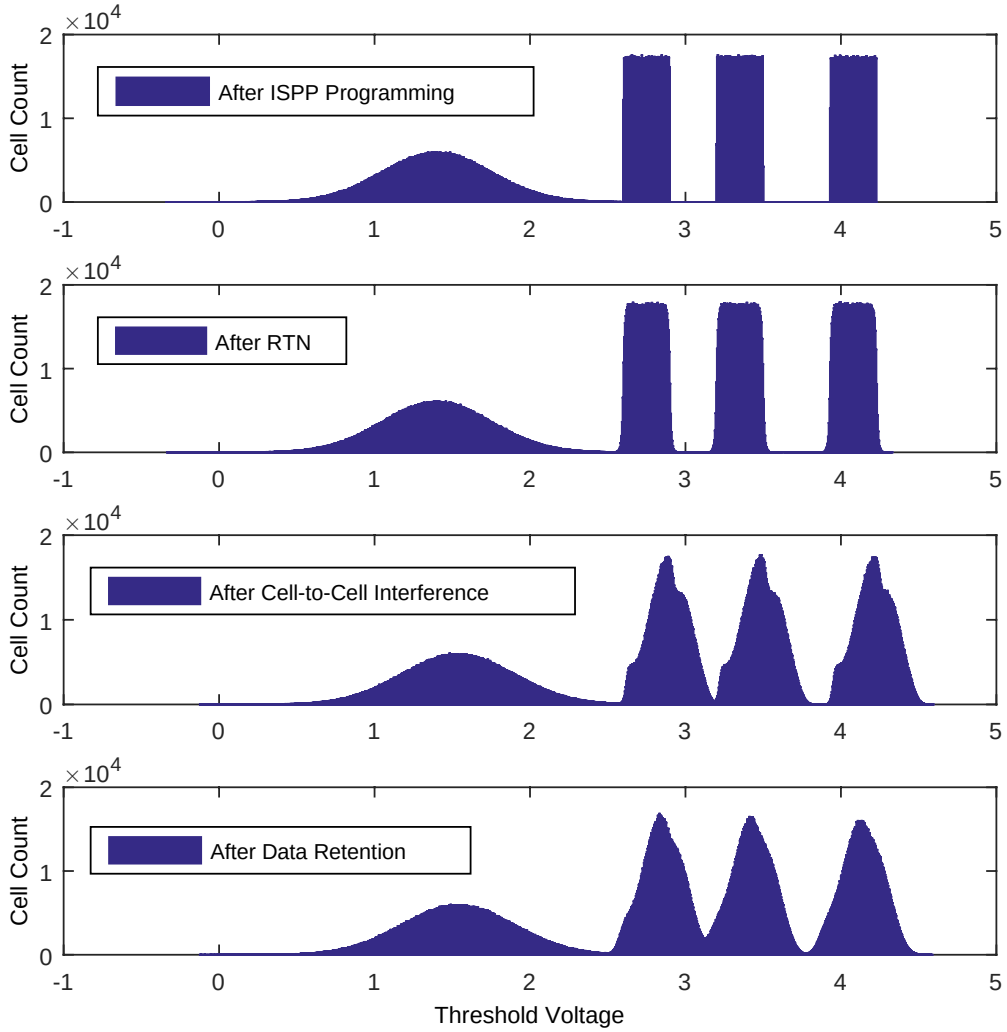


Fig. 5.2 Simulation of threshold-voltage distribution after 1K P/E Cycling and 1 month retention: memory cell suffers CCI from vertical and diagonal neighbouring cells

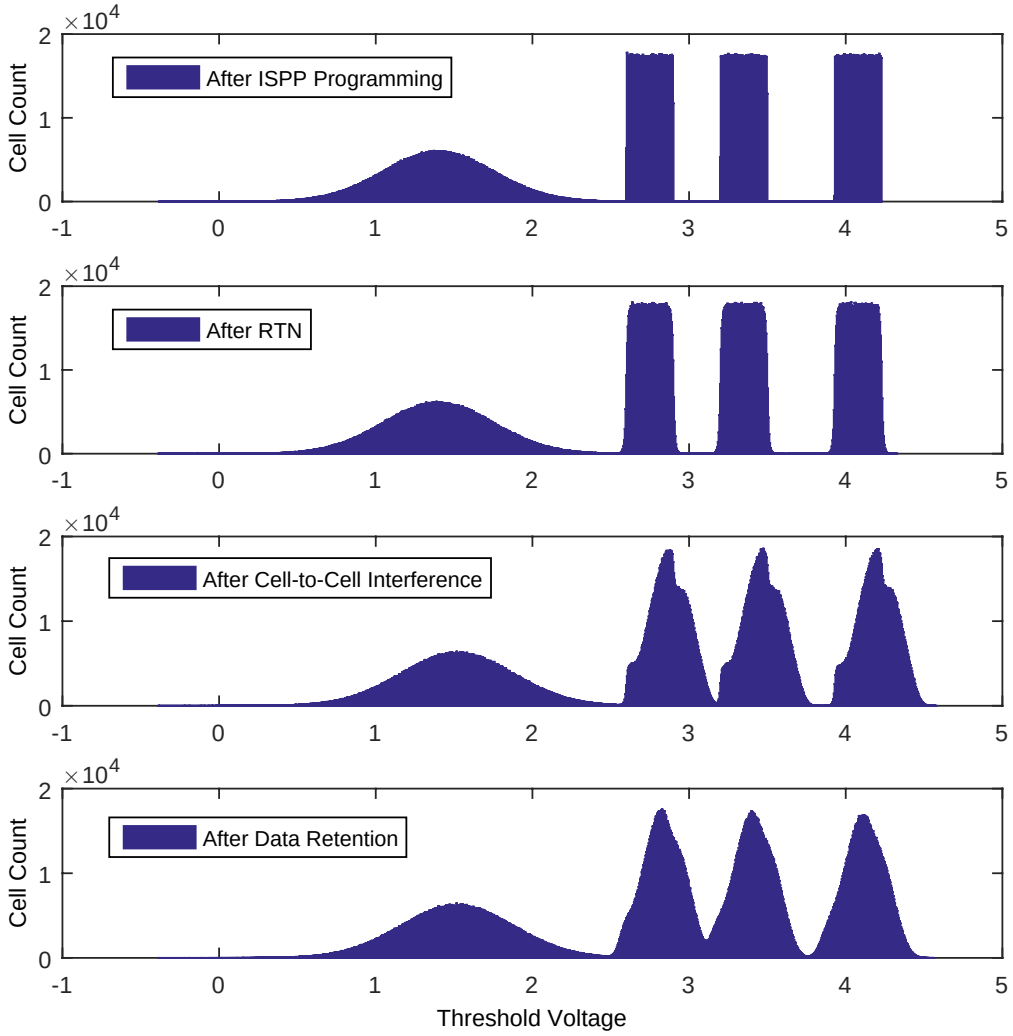


Fig. 5.3 Simulation of memory threshold-voltage distribution after 1K P/E Cycles and 1 month retention: memory cell suffers CCI (random coupling ratio) from neighbouring cell in vertical direction only

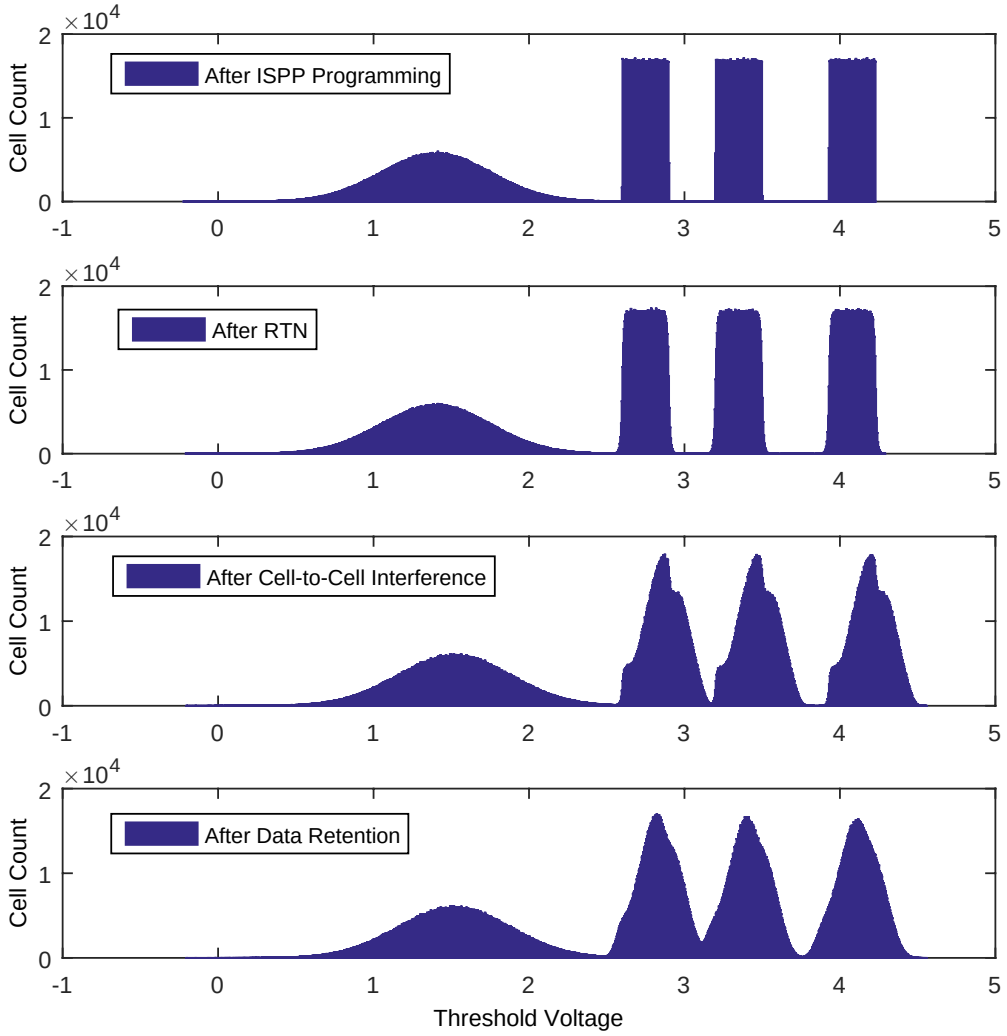


Fig. 5.4 Simulation of memory threshold-voltage distribution after 1K P/E Cycles and 1 month retention: memory cell suffers CCI (constant coupling ratio) from neighbouring cell in vertical direction only

the squared Euclidean distance between the measured threshold voltage values and those obtained from the Gaussian mixture model. Although the curve-fitting approaches can produce the flash channel models for specific applications and researches, such as the Gaussian mixture model, the distributions obtained would not be as precise as the curves shown in the figures above. As a sum of the multiple sources of physical noise and interference, the final threshold voltages are very hard to be classified as following any type of the known distributions. In addition, we need a well-developed channel model and more exact final threshold voltage distributions for the decoding of advanced ECCs. Based on this motivation, we characterise and model the NAND flash channel mathematically and attempt to propose a set of formulas to express the curves shown in the figure. Details on the results and derivations can be found in Section 5.3.

5.2.2 Characteristic Functions of Random Variables

This section briefly reviews the topic of characteristic functions in probability theory. By using characteristic functions, we have the benefit of changing the convolutions of noise distributions to multiplications, thus reducing the computational complexity greatly.

Basics of Characteristic Functions

Definition. If X is a random variable, then its *characteristic function* $c : \mathbb{R} \to \mathbb{C}$ is defined by

$$c(\xi) := \mathbb{E}(e^{i\xi x}),\ \xi \in \mathbb{R}\ arbitrary \tag{5.4}$$

If X has a probability density function, $f(x)$, then this becomes

$$c(\xi) = \int_{\mathbb{R}} e^{i\xi x} f(x) dx \tag{5.5}$$

Inversion Formula. The way of recovering $f(x)$ from $c(\xi)$ is given by

$$f(x) = \frac{1}{2\pi} \int_{\mathbb{R}} e^{-i\xi x} c(\xi) d\xi \tag{5.6}$$

Characteristic Functions of Normal Distribution

If X is a standard normal distribution, $X \sim \mathcal{N}(0,1)$, i.e., $f(x) = (1/\sqrt{2\pi})e^{-x^2/2}$, then

$$\begin{aligned} c(\xi) &= \int_{\mathbb{R}} e^{i\xi x} f(x)dx \\ &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} e^{i\xi x} e^{-x^2/2} dx = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} e^{-\frac{1}{2}[x^2 - 2ix\xi + (i\xi)^2]} e^{-\xi^2/2} dx \\ &= \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}\xi^2} \int_{-\infty}^{+\infty} e^{-\frac{1}{2}(x-i\xi)^2} dx \overset{\substack{u=x-i\xi \\ du=dx}}{=\!=\!=} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}\xi^2} \int_{-\infty}^{+\infty} e^{-\frac{1}{2}u^2} du \\ &= e^{-\frac{1}{2}\xi^2} \end{aligned} \tag{5.7}$$

Furthermore, if $X \sim \mathcal{N}(\mu, \sigma^2)$, i.e., $f(x) = (1/\sqrt{2\pi}\sigma)e^{-(x-\mu)^2/2\sigma^2}$, then

$$\begin{aligned} c(\xi) &= \int_{\mathbb{R}} e^{i\xi x} f(x)dx \\ &= \frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{+\infty} e^{i\xi x} e^{-(x-\mu)^2/2\sigma^2} dx = \frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{+\infty} e^{-\frac{1}{2\sigma^2}x^2 + (i\xi + \frac{\mu}{\sigma^2})x - \frac{\mu}{2\sigma^2}} dx \\ &= \frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{+\infty} e^{-\frac{1}{2\sigma^2}[x^2 - 2x(\sigma^2 i\xi + \mu) + (\sigma^2 i\xi + \mu)^2]} e^{-\frac{1}{2}\sigma^2\xi^2} \cdot e^{\mu i\xi} dx \\ &= \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}\sigma^2\xi^2} \cdot e^{\mu i\xi} \int_{-\infty}^{+\infty} e^{-\frac{1}{2}u^2} du = e^{-\frac{1}{2}\sigma^2\xi^2} \cdot e^{\mu i\xi} \end{aligned} \tag{5.8}$$

Characteristic Functions of Laplace Distribution

If X has a Laplace distribution, i.e., $f(x) = (1/2)e^{-|x|}$, then

$$\begin{aligned} c(\xi) &= \int_{\mathbb{R}} e^{i\xi x} f(x)dx = \frac{1}{2} \int_{-\infty}^{+\infty} e^{i\xi x} e^{-|x|} dx \\ &= \frac{1}{2} \left(\int_{-\infty}^{0} e^{i\xi x} e^{x} dx + \int_{0}^{+\infty} e^{i\xi x} e^{-x} dx \right) \\ &= \frac{1}{2} \left(\int_{-\infty}^{0} e^{(i\xi+1)x} dx + \int_{0}^{+\infty} e^{(i\xi-1)x} dx \right) \\ &= \frac{1}{2(i\xi+1)} \int_{-\infty}^{0} e^{u} du + \frac{1}{2(i\xi-1)} \int_{0}^{-\infty} e^{u} du \\ &= \frac{1}{1+\xi^2} \end{aligned} \tag{5.9}$$

More generally, if $f(x) = (\frac{1}{2\lambda})e^{-|x|/\lambda}$ with $\lambda > 0$, we get the following result by using properties (5.16) and (5.17):

$$c(\xi) = \frac{1}{1+\lambda^2\xi^2} \tag{5.10}$$

Characteristic Functions of Uniform Distribution

If X has a uniform distribution, i.e.,

$$f(x) = \begin{cases} 1, & 0 \le x \le 1 \\ 0, & otherwise \end{cases} \tag{5.11}$$

then we have

$$c(\xi) = \int_{\mathbb{R}} e^{i\xi x} f(x)dx = \int_0^1 e^{i\xi x}dx = \frac{1}{i\xi}\int_0^{i\xi} e^u du = \frac{1}{i\xi}(e^{i\xi} - 1) \tag{5.12}$$

Furthermore, if

$$f(x) = \begin{cases} a, & l \le x \le r \\ 0, & otherwise \end{cases} \tag{5.13}$$

then

$$c(\xi) = \int_{\mathbb{R}} e^{i\xi x} f(x)dx = a\int_l^r e^{i\xi x}dx = a \cdot \frac{1}{i\xi}\int_{i\xi l}^{i\xi r} e^u du = a \cdot \frac{1}{i\xi}(e^{i\xi r} - e^{i\xi l}) \tag{5.14}$$

Some Properties of Characteristic Functions

Suppose that Y and Z are two continuous random variables, and $Y \sim g_Y(x)$, $Z \sim f_Z(x)$ with $x \in (-\infty, +\infty)$. If $g_Y(x) = f_Z(x-\mu)$, then

$$\begin{aligned} c_Y(\xi) = \int_{\mathbb{R}} e^{i\xi x} f_Z(x-\mu)dx &= \int_{-\infty}^{+\infty} e^{i\xi u} e^{i\xi\mu} f_Z(u)du \\ &= e^{i\xi\mu} c_Z(\xi) \end{aligned} \tag{5.15}$$

If $g_Y(x) = a \cdot f_Z(x)$, then

$$\begin{aligned} c_Y(\xi) = \int_{\mathbb{R}} e^{i\xi x} a f_Z(x)dx &= a\int_{-\infty}^{+\infty} e^{i\xi x} f_Z(x)dx \\ &= a \cdot c_Z(\xi) \end{aligned} \tag{5.16}$$

If $g_Y(x) = f_Z(bx)$, then

$$\begin{aligned} c_Y(\xi) &= \int_{\mathbb{R}} e^{i\xi x} f_Z(bx)dx \\ &= \begin{cases} \frac{1}{b}\int_{-\infty}^{+\infty} e^{\frac{i\xi}{b}u} f_Z(u)du = \frac{1}{b}\cdot c_Z(\frac{\xi}{b}), & b>0 \\ \frac{1}{b}\int_{+\infty}^{-\infty} e^{\frac{i\xi}{b}u} f_Z(u)du = -\frac{1}{b}\cdot c_Z(\frac{\xi}{b}), & b<0 \end{cases} \\ &= \frac{1}{|b|}\cdot c_Z(\frac{\xi}{b}) \end{aligned} \tag{5.17}$$

5.2.3 Related Work

Due to the importance of advanced ECCs for NAND flash memories, many researchers have been trying to investigate flash coding using either LDPC codes or TPC codes. Generally, Gaussian-based channel models are widely used in the study of flash coding. For instance, pulse amplitude modulation (PAM) with Gaussian noise of same variances is used by Wang et al. to model flash cell threshold voltage levels [71]. Sun et al. approximated the flash channel with a similar model except with different variances for each level [62]. A similar model was used by Zhou et al. to explore the advantages of dynamic thresholds in flash reading [86]. All of these Gaussian-based channel models are relatively simple allowing researchers to focus more on the coding aspects. However, the fact that the real distribution is far different from the approximate model naturally raises doubts about the correctness of reliability results.

Other researchers have sought to characterise the final voltage distribution more accurately based on the noise sources. Lee and Sung [31] developed parameter estimation algorithms to find the means and variances of the threshold voltage. Their approach is to find the best-fit parameters by minimising the squared Euclidean distance between the measured threshold voltage values and those obtained from the flash model, which is more exact since the parameters are estimated separately for each symbol. However, the final distributions are still approximated as Gaussian mixtures. Dong et al. [15] treated the cell-to-cell interference as the dominant distortion and mathematically derived the final voltage distribution. They derive mathematical formulas to approximately model the threshold-voltage distribution of memory cells in the presence of cell-to-cell

interference, based on which the calculation of LLRs is also formulated. However, the results presented in their work were still approximate due to the neglect of the other noise sources. Another idea is to observe the channel from the statistical view and ignore the exact probability distribution function for distortion. The channel model based on this idea has been proposed recently by Moon et al. and successfully used to instruct the error correction coding [46, 47]. However, this statistical approach is not capable of providing clear mathematical descriptions of the channel noises and the final threshold voltage distribution.

This work derives exact probability density functions for the combined channel noise sources and the final cell threshold voltage which are then used for calculating the soft channel information. The approach here uses the characteristic function of each noise distribution to determine the cumulative influence caused by all sources of noise. For the case of 4-level MLC flash memory, the noise distribution is calculated and compared to simulation results. The final distributions of each symbol are determined in explicit formulas providing a way to calculate the soft information. Simulation results over the channel with specified P/E cycling number and retention time demonstrate that the theoretical deductions are consistent with the practical channel outputs.

5.3 Characterisation of NAND Flash Memory Channel

The goal of this work is to characterise the flash channel exactly in terms of the probability density functions of noise sources and cell threshold voltages, which were simulated in Section 5.2.1. For this purpose we investigate the channel behaviours theoretically in this section to understand their intrinsic characteristics. Generally, the effects caused by RTN and CCI are equal to each memory programming state while the distortion in data retention are proportional to the initial threshold voltages; thus, we present four equations corresponding to the four programming states of the 2 bits/cell MLC NAND flash memory. Based on these formulas, we continue to derive the final threshold voltage distribution of each level of the memory.

The results shown in this section are important to both theory and practise. Firstly, the Monte Carlo simulations of the channel model demonstrate a match of the theoretical results to the real channel behaviours. Secondly, the distributions of final threshold voltages can be used straightforwardly for advance ECCs decoding by doing some on-chip lookup table computations only. It is easy to implement the presented work in the NAND flash memory systems.

Based on Equations (5.1) and (5.2), there are two ways to characterise the final threshold voltage distribution, $V^{(k)}$. The first approach is to calculate the convolutions orderly from left to right in Equation (5.2). However, the distributions of initial threshold voltage and the noise sources are fairly complicated, and it is computationally intensive to get the results. Alternatively, we can derive the distribution for the distortion first and convolve the results with the initial threshold voltage distributions. Due to the associative law of convolution and independence of the sources of noise, this change in the calculation order will not influence the final results. Moreover, as seen in the following section, calculation is greatly simplified by the use of characteristic functions for the distortion.

Another thing to be pointed out is the coupling ratio. The analysis and simulation in Section 5.2.1 have shown that the change of distribution is negligibly small after ignoring the diagonal cell-to-cell interference and making the γ_y a constant. As a result, we also use these two approximations originally presented by Dong et al. [16] in our work to simplify the mathematical derivations.

5.3.1 Overall Distribution of the Noises and Distortions

Let ΔV denote the threshold voltage shift induced by the three kinds of noise sources, and $\widetilde{p}(x)$ the probability density function of ΔV. In the equivalent model shown in Figure 5.1, ΔV is the sum of the three shifts:

$$\begin{aligned} \Delta V &= \Delta V_{RTN} + \Delta V_{CCI} - \Delta V_{ret} \\ &= \Delta V_{RTN} + \gamma_y \cdot \Delta V_t - \Delta V_{ret} \end{aligned} \tag{5.18}$$

Let $\Delta V_{(n)},\ (0 \leq n \leq K-1)$ represent the threshold voltage shift when the interfering cell in the vertical direction is being programmed to the nth state, and let $\widetilde{p}_{(n)}(x)$ represent the probability density function of $\Delta V_{(n)}$. We have

$$\Delta V_{(n)} = \Delta V_{RTN} + \gamma_y (V_i^{(n)} - V_i^{(0)}) - \Delta V_{ret} \tag{5.19}$$

Assuming that the interfering cells have the same probability to be programmed to each symbol, the overall distribution of total noise induced threshold voltage shifts can be approximated as

$$\Delta V \sim \widetilde{p}(x) = \frac{1}{K} \sum_{n=0}^{K-1} \widetilde{p}_{(n)}(x) \tag{5.20}$$

In the following, we first derive $\widetilde{p}_{(n)}(x)$ and then use it to find $\widetilde{p}(x)$ for the overall distribution according to (5.20). Considering all K possible levels for n, (5.19) can be rewritten as

$$\Delta V_{(n)} = \begin{cases} \Delta V_{RTN} - \Delta V_{ret}, & n = 0 \\ \Delta V_{RTN} + \gamma_y V_i^{(n)} - \gamma_y V_i^{(0)} - \Delta V_{ret}, & 1 \leq n \leq K-1 \end{cases} \tag{5.21}$$

In order to obtain $\widetilde{p}_{(n)}(x)$, we need to calculate the convolution of probability density functions of all components in the equation above, but multiple convolutions is computationally difficult. As a simpler alternative, we use characteristic functions instead of convolutions. The characteristic functions of the major components in (5.21) are the following:

$$\Delta V_{RTN} \rightarrow c_{RTN}(\xi) = \frac{1}{1 + \lambda_r^2 \xi^2} \tag{5.22}$$

$$-\Delta V_{ret} \rightarrow c_{ret}(\xi) = e^{-\frac{1}{2}\sigma_d^2 \xi^2} \cdot e^{-i\xi\mu_d} \tag{5.23}$$

$$-\gamma_y V_i^{(0)} \rightarrow c_e(\xi) = e^{-\frac{1}{2}\sigma_e^2 \gamma_y^2 \xi^2} \cdot e^{-i\xi\mu_e\gamma_y} \tag{5.24}$$

$$\gamma_y V_i^{(n)} \to c_n(\xi) = \frac{1}{i\xi\gamma_y \Delta V_{pp}} \left(e^{i\xi\gamma_y V_r^{(n)}} - e^{i\xi\gamma_y V_l^{(n)}} \right), \quad 1 \le n \le K-1 \tag{5.25}$$

In terms of $1 \le n \le K-1$, we split the right side of (5.21) into two parts: $\Delta V_{(n)} = V_1 + V_2$ with $V_1 = \Delta V_{RTN} + \gamma_y V_i^{(n)}$ and $V_2 = -\gamma_y V_i^{(0)} - \Delta V_{ret}$.

To begin with, let us consider V_1. Let $f_1(x)$ denote the probability density function of V_1. The characteristic function of $f_1(x)$, denoted as $c_1(\xi)$, can be calculated as

$$\begin{aligned} c_1(\xi) &= c_{RTN}(\xi) \cdot c_n(\xi) \\ &= \frac{1}{i\gamma_y \Delta V_{pp}} \cdot \frac{1}{(1+\lambda_r^2\xi^2)\xi} \cdot \left[e^{i\gamma_y V_r^{(n)}\xi} - e^{i\gamma_y V_l^{(n)}\xi} \right] \end{aligned} \tag{5.26}$$

Therefore,

$$\begin{aligned} f_1(x) &= \frac{1}{2\pi} \int_{-\infty}^{+\infty} e^{-i\xi x} c_1(\xi) d\xi \\ &= \frac{1}{2\pi i \gamma_y \Delta V_{pp}} \int_{-\infty}^{+\infty} \frac{1}{(1+\lambda_r^2\xi^2)\xi} \cdot \left[e^{i\xi(\gamma_y V_r^{(n)} - x)} - e^{i\xi(\gamma_y V_l^{(n)} - x)} \right] d\xi \end{aligned} \tag{5.27}$$

Define

$$q(t) = \frac{1}{\pi i} \int_{-\infty}^{+\infty} \frac{1}{(1+\lambda_r^2\xi^2)\xi} \cdot e^{i\xi t} d\xi \tag{5.28}$$

and

$$Q(z) = \frac{1}{\lambda_r^2((1/\lambda_r^2)+z^2)z} \cdot e^{izt} \tag{5.29}$$

which have three simple poles in the real axis: $z_0 = 0$, $z_1 = i/\lambda_r$ and $z_2 = -i/\lambda_r$. Intuitively, we would believe the integral in (5.28) can be obtained based on the residue theorem.

Assume $t \ge 0$, and let us consider the contour shown in Figure 5.5 (a), then we have the following equation according to the Cauchy integral theorem:

$$\begin{aligned} \int_{C_{R+}} Q(z)dz + \int_{C_{r+}} Q(z)dz + \int_{-R}^{-r} Q(z)dz + \int_{+r}^{+R} Q(z)dz &= 2\pi i Res(Q(z), z_1) \\ &= -\pi i e^{-\frac{1}{\lambda_r}t} \end{aligned} \tag{5.30}$$

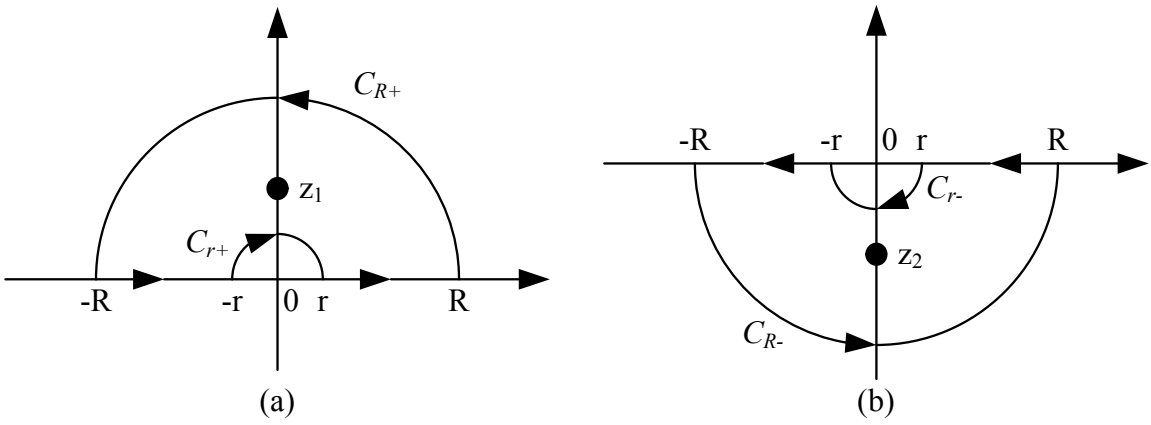


Fig. 5.5 Contours used for calculating the noise probability density function

If $R \to +\infty$ and $r \to 0$, then $\int_{-R}^{-r} Q(z)dz + \int_{+r}^{+R} Q(z)dz$ will be equal to $\pi i \cdot q(t)$. Considering the semicircle that has radius R, we have

$$\begin{aligned}\left|\int_{C_{R+}} Q(z)dz\right| &= \left|\int_{C_{R+}} \frac{1}{(1+\lambda_r^2 z^2)z} e^{izt} dz\right| \\ &= \left|\int_0^{\pi} \frac{e^{itR(cos\theta+isin\theta)}}{(1+\lambda_r^2 R^2 e^{i2\theta})Re^{i\theta}} Rie^{i\theta} d\theta\right| \\ &\leq \int_0^{\pi} \left|\frac{e^{-tR\sin\theta}}{1+\lambda_r^2 R^2 e^{i2\theta}}\right| d\theta \\ &\leq \int_0^{\pi} \left|\frac{1}{e^{tR\sin\theta}\cdot(\lambda_r^2 R-1)}\right| d\theta\end{aligned} \tag{5.31}$$

Since the integrand above vanishes to 0 when $R \to +\infty$, we have the following equation based on Jordan's lemma:

$$\int_{C_{R+}} Q(z)dz = 0, R \to +\infty \tag{5.32}$$

while

$$\begin{aligned}\lim_{r\to 0}\int_{C_{r+}} Q(z)dz &= -Res[Q(z), z_0]\pi i \\ &= -\pi i\end{aligned} \tag{5.33}$$

Therefore,

$$q(t) = \frac{1}{\pi i}\int_{-\infty}^{+\infty} \frac{1}{(1+\lambda_r^2\xi^2)\xi}\cdot e^{i\xi t} d\xi = 1 - e^{-\frac{1}{\lambda_r}t}, t \geq 0 \tag{5.34}$$

Now assume the other case $t < 0$, and let us consider the contour shown in Figure 5.5 (b), then we have the following equation based on the residue theory:

$$\begin{aligned}\int_{C_{R-}} Q(z)dz + \int_{C_{r-}} Q(z)dz + \int_{-r}^{-R} Q(z)dz + \int_{+R}^{+r} Q(z)dz &= 2\pi i Res(Q(z), z_2) \\ &= -\pi i e^{\frac{1}{\lambda_r}t}\end{aligned} \tag{5.35}$$

If $R \to +\infty$ and $r \to 0$, then $\int_{-r}^{-R} Q(z)dz + \int_{+R}^{+r} Q(z)dz$ becomes $-\pi i \cdot q(t)$. Considering the semicircle that has radius R, we have

$$\begin{aligned}\left|\int_{C_{R-}} Q(z)dz\right| &= \left|\int_{C_{R-}} \frac{1}{(1+\lambda_r^2 z^2)z} e^{izt} dz\right| \\ &= \left|\int_{\pi}^{2\pi} \frac{e^{itR(cos\theta + isin\theta)}}{(1+\lambda_r^2 R^2 e^{i2\theta})Re^{i\theta}} Rie^{i\theta} d\theta\right| \\ &\leq \int_{\pi}^{2\pi} \left|\frac{e^{-tR\sin\theta}}{1+\lambda_r^2 R^2 e^{i2\theta}}\right| d\theta \\ &\leq \int_{\pi}^{2\pi} \left|\frac{1}{e^{tR\sin\theta}\cdot(\lambda_r^2 R - 1)}\right| d\theta\end{aligned} \tag{5.36}$$

Since the integrand above vanishes to 0 when $R \to +\infty$, we have the following equation based on Jordan's lemma:

$$\int_{C_{R-}} Q(z)dz = 0, R \to +\infty \tag{5.37}$$

while

$$\begin{aligned}\lim_{r\to 0} \int_{C_{r-}} Q(z)dz &= -Res[Q(z), z_0]\pi i \\ &= -\pi i\end{aligned} \tag{5.38}$$

Therefore,

$$q(t) = \int_{-\infty}^{+\infty} \frac{1}{(1+\lambda_r^2 \xi^2)\xi} \cdot e^{i\xi t} d\xi = -1 + e^{\frac{1}{\lambda_r}t}, t < 0 \tag{5.39}$$

Formulas (5.34) and (5.39) can be written together as

$$q(t) = \int_{-\infty}^{+\infty} \frac{1}{(1+\lambda_r^2\xi^2)\xi} \cdot e^{i\xi t} d\xi = \begin{cases} 1 - e^{-\frac{1}{\lambda_r}t} & t \geq 0 \\ -1 + e^{\frac{1}{\lambda_r}t} & t < 0 \end{cases} \tag{5.40}$$

Based on (5.27), the function $f_1(x)$ can be written as

$$f_1(x) = \frac{1}{2\gamma_y \Delta V_{pp}} \left[q(t_1(x)) - q(t_2(x)) \right] \tag{5.41}$$

where

$$\begin{bmatrix} t_1(x) \\ t_2(x) \end{bmatrix} = \begin{bmatrix} \gamma_y V_r^{(n)} - x \\ \gamma_y V_l^{(n)} - x \end{bmatrix} \tag{5.42}$$

Next we consider the probability density function of V_2. The characteristic function of V_2, represented as $c_2(\xi)$, is given by

$$\begin{aligned} c_2(\xi) &= c_e(\xi) \cdot c_{ret}(\xi) \\ &= e^{-\frac{1}{2}(\sigma_e^2\gamma_y^2 + \sigma_d^2)\xi^2} \cdot e^{i(-\mu_d - \mu_e\gamma_y)\xi} \end{aligned} \tag{5.43}$$

The probability density function of V_2, denoted as $f_2(x)$, can be further obtained by

$$\begin{aligned} f_2(x) &= \frac{1}{2\pi} \int_{-\infty}^{+\infty} e^{-i\xi x} c_2(\xi) d\xi \\ &= \frac{1}{\sqrt{2\pi(\sigma_e^2\gamma_y^2 + \sigma_d^2)}} e^{\frac{-(x+\mu_d+\mu_e\gamma_y)^2}{2(\sigma_e^2\gamma_y^2+\sigma_d^2)}} \end{aligned} \tag{5.44}$$

With the results above, $\widetilde{p}_{(n)}(x)$ $(1 \leq n \leq K-1)$ can be determined by the convolution of $f_1(x)$ and $f_2(x)$:

$$\begin{aligned} \widetilde{p}_{(n)}(x) &= f_1(x) \otimes f_2(x) \\ &= \int_{-\infty}^{+\infty} f_1(\tau) f_2(x-\tau) d\tau \\ &= \frac{1}{2\gamma_y \Delta V_{pp}} \left(\int_{-\infty}^{+\infty} q(t_1(\tau)) f_2(x-\tau) d\tau - \int_{-\infty}^{+\infty} q(t_2(\tau)) f_2(x-\tau) d\tau \right) \end{aligned} \tag{5.45}$$

Let $B = -\mu_d - \mu_e\gamma_y$ and $C = \sqrt{\sigma_e^2\gamma_y^2 + \sigma_d^2}$. Before calculating the integrals above, let us consider the two simple forms below:

$$\begin{aligned}\frac{1}{\sqrt{2\pi}C}\int_{-\infty}^{\Lambda} e^{l\tau}\cdot e^{-\frac{(x-\tau-B)^2}{2C^2}}d\tau &= e^{l(x-B)}\cdot e^{\frac{l^2C^2}{2}}\cdot\frac{1}{\sqrt{2\pi}C}\int_{-\infty}^{\Lambda} e^{-\frac{[\tau-(x+lC^2-B)]^2}{2C^2}}d\tau \\ &= \frac{1}{2}e^{l(x-B)}\cdot e^{\frac{l^2C^2}{2}}\left[1+erf\left(\frac{-x+B-lC^2+\Lambda}{\sqrt{2}C}\right)\right]\end{aligned} \tag{5.46}$$

and

$$\begin{aligned}\frac{1}{\sqrt{2\pi}C}\int_{\Lambda}^{+\infty} e^{l\tau}\cdot e^{-\frac{(x-\tau-B)^2}{2C^2}}d\tau &= e^{l(x-B)}\cdot e^{\frac{l^2C^2}{2}}\cdot\frac{1}{\sqrt{2\pi}C}\int_{\Lambda}^{+\infty} e^{-\frac{[\tau-(x+lC^2-B)]^2}{2C^2}}d\tau \\ &= \frac{1}{2}e^{l(x-B)}\cdot e^{\frac{l^2C^2}{2}}\left[1-erf\left(\frac{-x+B-lC^2+\Lambda}{\sqrt{2}C}\right)\right]\end{aligned} \tag{5.47}$$

where

$$erf(\eta) = \frac{2}{\sqrt{\pi}}\int_0^{\eta} e^{-\tau^2}d\tau \tag{5.48}$$

with $erf(+\infty) = 1$ and $erf(-x) = -erf(x)$.

Considering the convolution $q(t_1(x)) \otimes f_2(x)$, we have

$$
\begin{aligned}
&\int_{-\infty}^{+\infty} q(t_1(\tau)) f_2(x-\tau) d\tau \\
&= \frac{1}{\sqrt{2\pi}C} \int_{-\infty}^{\gamma_y V_r^{(n)}} e^{-\frac{(x-\tau-B)^2}{2C^2}} d\tau - \frac{1}{\sqrt{2\pi}C} e^{-\frac{1}{\lambda_r}\gamma_y V_r^{(n)}} \int_{-\infty}^{\gamma_y V_r^{(n)}} e^{\frac{1}{\lambda_r}\tau} e^{-\frac{(x-\tau-B)^2}{2C^2}} d\tau \\
&- \frac{1}{\sqrt{2\pi}C} \int_{\gamma_y V_r^{(n)}}^{+\infty} e^{-\frac{(x-\tau-B)^2}{2C^2}} d\tau + \frac{1}{\sqrt{2\pi}C} e^{\frac{1}{\lambda_r}\gamma_y V_r^{(n)}} \int_{\gamma_y V_r^{(n)}}^{+\infty} e^{-\frac{1}{\lambda_r}\tau} e^{-\frac{(x-\tau-B)^2}{2C^2}} d\tau \\
&= \frac{1}{2}\left[1 + erf\left(\frac{-x+B+\gamma_y V_r^{(n)}}{\sqrt{2}C}\right)\right] - \frac{1}{2}\left[1 - erf\left(\frac{-x+B+\gamma_y V_r^{(n)}}{\sqrt{2}C}\right)\right] \\
&- \frac{1}{2} e^{-\frac{1}{\lambda_r}\left(-x+B+\gamma_y V_r^{(n)} - \frac{C^2}{2\lambda_r}\right)} \left[1 + erf\left(\frac{-x+B+\gamma_y V_r^{(n)} - \frac{C^2}{\lambda_r}}{\sqrt{2}C}\right)\right] \\
&+ \frac{1}{2} e^{\frac{1}{\lambda_r}\left(-x+B+\gamma_y V_r^{(n)} + \frac{C^2}{2\lambda_r}\right)} \left[1 - erf\left(\frac{-x+B+\gamma_y V_r^{(n)} + \frac{C^2}{\lambda_r}}{\sqrt{2}C}\right)\right] \\
&= erf\left(\frac{-x+B+\gamma_y V_r^{(n)}}{\sqrt{2}C}\right) - \frac{1}{2} e^{-\frac{1}{\lambda_r}\left(-x+B+\gamma_y V_r^{(n)} - \frac{C^2}{2\lambda_r}\right)} \left[1 + erf\left(\frac{-x+B+\gamma_y V_r^{(n)} - \frac{C^2}{\lambda_r}}{\sqrt{2}C}\right)\right] \\
&+ \frac{1}{2} e^{\frac{1}{\lambda_r}\left(-x+B+\gamma_y V_r^{(n)} + \frac{C^2}{2\lambda_r}\right)} \left[1 - erf\left(\frac{-x+B+\gamma_y V_r^{(n)} + \frac{C^2}{\lambda_r}}{\sqrt{2}C}\right)\right]
\end{aligned}
\tag{5.49}
$$

In a similar way, we obtain the convolution $q(t_2(x)) \otimes f_2(x)$ as

$$
\begin{aligned}
&\int_{-\infty}^{+\infty} q(t_2(\tau)) f_2(x-\tau) d\tau \\
&= erf\left(\frac{-x+B+\gamma_y V_l^{(n)}}{\sqrt{2}C}\right) - \frac{1}{2} e^{-\frac{1}{\lambda_r}\left(-x+B+\gamma_y V_l^{(n)} - \frac{C^2}{2\lambda_r}\right)} \left[1 + erf\left(\frac{-x+B+\gamma_y V_l^{(n)} - \frac{C^2}{\lambda_r}}{\sqrt{2}C}\right)\right] \\
&+ \frac{1}{2} e^{\frac{1}{\lambda_r}\left(-x+B+\gamma_y V_l^{(n)} + \frac{C^2}{2\lambda_r}\right)} \left[1 - erf\left(\frac{-x+B+\gamma_y V_l^{(n)} + \frac{C^2}{\lambda_r}}{\sqrt{2}C}\right)\right]
\end{aligned}
\tag{5.50}
$$

Furthermore, we can calculate $\widetilde{p}_{(n)}(x)$ $(1 \leq n \leq K-1)$ based on (5.45) to be:

$$\begin{aligned}
\widetilde{p}_{(n)}(x) &= \frac{1}{2\gamma_y \Delta V_{pp}} \left[q(t_1(x)) \otimes f_2(x) - q(t_2(x)) \otimes f_2(x) \right] \\
&= \frac{1}{2\gamma_y \Delta V_{pp}} \cdot \left[erf\left(\frac{-x+B+\gamma_y V_r^{(n)}}{\sqrt{2}C} \right) - erf\left(\frac{-x+B+\gamma_y V_l^{(n)}}{\sqrt{2}C} \right) \right] + \frac{e^{\frac{C^2}{2\lambda_r^2}}}{4\gamma_y \Delta V_{pp}} \cdot \\
&\left[e^{\frac{1}{\lambda_r}\left(-x+B+\gamma_y V_r^{(n)}\right)} \left(1 - erf\left(\frac{-x+B+\gamma_y V_r^{(n)} + \frac{C^2}{\lambda_r}}{\sqrt{2}C} \right) \right) \right. \\
&+ e^{-\frac{1}{\lambda_r}\left(-x+B+\gamma_y V_l^{(n)}\right)} \left(1 + erf\left(\frac{-x+B+\gamma_y V_l^{(n)} - \frac{C^2}{\lambda_r}}{\sqrt{2}C} \right) \right) \\
&- e^{-\frac{1}{\lambda_r}\left(-x+B+\gamma_y V_r^{(n)}\right)} \left(1 + erf\left(\frac{-x+B+\gamma_y V_r^{(n)} - \frac{C^2}{\lambda_r}}{\sqrt{2}C} \right) \right) \\
&\left. - e^{\frac{1}{\lambda_r}\left(-x+B+\gamma_y V_l^{(n)}\right)} \left(1 - erf\left(\frac{-x+B+\gamma_y V_l^{(n)} + \frac{C^2}{\lambda_r}}{\sqrt{2}C} \right) \right) \right]
\end{aligned} \tag{5.51}$$

Let us define

$$\zeta(x, \omega, \upsilon) = erf\left(\frac{-x+\omega}{\sqrt{2}\upsilon} \right) \tag{5.52}$$

The results in (5.51) can be simplified as

$$\begin{aligned}
\widetilde{p}_{(n)}(x) =& \frac{1}{2\gamma_y \Delta V_{pp}} \left[q(t_1(x)) \otimes f_2(x) - q(t_2(x)) \otimes f_2(x) \right] \\
=& \frac{1}{2\gamma_y \Delta V_{pp}} \cdot \left[\zeta(x, B+\gamma_y V_r^{(n)}, C) - \zeta(x, B+\gamma_y V_l^{(n)}, C) \right] + \\
& \frac{e^{\frac{C^2}{2\lambda_r^2}}}{4\gamma_y \Delta V_{pp}} \cdot \left[e^{\frac{1}{\lambda_r}\left(-x+B+\gamma_y V_r^{(n)}\right)} \left(1 - \zeta(x, B+\gamma_y V_r^{(n)} + \frac{C^2}{\lambda_r}, C) \right) \right. \\
& + e^{-\frac{1}{\lambda_r}\left(-x+B+\gamma_y V_l^{(n)}\right)} \left(1 + \zeta(x, B+\gamma_y V_l^{(n)} - \frac{C^2}{\lambda_r}, C) \right) \\
& - e^{-\frac{1}{\lambda_r}\left(-x+B+\gamma_y V_r^{(n)}\right)} \left(1 + \zeta(x, B+\gamma_y V_r^{(n)} - \frac{C^2}{\lambda_r}, C) \right) \\
& \left. - e^{\frac{1}{\lambda_r}\left(-x+B+\gamma_y V_l^{(n)}\right)} \left(1 - \zeta(x, B+\gamma_y V_l^{(n)} + \frac{C^2}{\lambda_r}, C) \right) \right], (1 \leq n \leq K-1)
\end{aligned} \tag{5.53}$$

For $n = 0$, the probability density function $\widetilde{p}_{(n)}(x)$ can be calculated by convolving $p_r(x)$ and $p_{ret}(x)$, expressed below:

$$\begin{aligned}\widetilde{p}_{(0)}(x) &= p_r(x) \otimes p_{ret}(x) = \int_{-\infty}^{+\infty} p_r(\tau) p_{ret}(x-\tau) d\tau \\ &= \frac{1}{2\lambda_r} \cdot \frac{1}{\sqrt{2\pi}\sigma_d} \int_{-\infty}^{+\infty} e^{-\frac{|\tau|}{\lambda_r}} \cdot e^{-\frac{(x-\tau-\mu_d)^2}{2\sigma_d^2}} d\tau \\ &= \frac{1}{2\lambda_r} \cdot \left[\frac{1}{\sqrt{2\pi}\sigma_d} \int_{-\infty}^{0} e^{\frac{1}{\lambda_r}\tau} \cdot e^{-\frac{(x-\tau-\mu_d)^2}{2\sigma_d^2}} d\tau + \frac{1}{\sqrt{2\pi}\sigma_d} \int_{0}^{+\infty} e^{-\frac{1}{\lambda_r}\tau} \cdot e^{-\frac{(x-\tau-\mu_d)^2}{2\sigma_d^2}} d\tau \right]\end{aligned} \tag{5.54}$$

Using the results shown in (5.46) and (5.47), the equation above can be revised as

$$\begin{aligned}\widetilde{p}_{(0)}(x) = &\frac{1}{4\lambda_r} \cdot e^{\frac{\sigma_d^2}{2\lambda_r^2}} \left[e^{\frac{1}{\lambda_r}(x-\mu_d)} \left(1 + erf\left(\frac{-x+\mu_d - \frac{1}{\lambda_r}\sigma_d^2}{\sqrt{2}\sigma_d} \right) \right) \right. \\ &\left. + e^{-\frac{1}{\lambda_r}(x-\mu_d)} \left(1 - erf\left(\frac{-x+\mu_d + \frac{1}{\lambda_r}\sigma_d^2}{\sqrt{2}\sigma_d} \right) \right) \right]\end{aligned} \tag{5.55}$$

Finally, the probability density function for noise sources in the flash channel, i.e., $\widetilde{p}(x)$ is given by

$$\begin{aligned}&\widetilde{p}(x) = \frac{1}{K} \times \sum_{n=0}^{K-1} \widetilde{p}_{(n)}(x) \\ &= \frac{1}{2\gamma_y \Delta V_{pp} K} \cdot \sum_{n=1}^{K-1} \left[erf\left(\frac{-x+B+\gamma_y V_r^{(n)}}{\sqrt{2}C} \right) - erf\left(\frac{-x+B+\gamma_y V_l^{(n)}}{\sqrt{2}C} \right) \right] + \frac{e^{\frac{C^2}{2\lambda_r^2}}}{4\gamma_y \Delta V_{pp} K} \\ &\cdot \sum_{n=1}^{K-1} \left[e^{\frac{1}{\lambda_r}\left(-x+B+\gamma_y V_r^{(n)}\right)} \left(1 - erf\left(\frac{-x+B+\gamma_y V_r^{(n)} + \frac{C^2}{\lambda_r}}{\sqrt{2}C} \right) \right) + e^{-\frac{1}{\lambda_r}\left(-x+B+\gamma_y V_l^{(n)}\right)} \cdot \right. \\ &\left(1 + erf\left(\frac{-x+B+\gamma_y V_l^{(n)} - \frac{C^2}{\lambda_r}}{\sqrt{2}C} \right) \right) - e^{-\frac{1}{\lambda_r}\left(-x+B+\gamma_y V_r^{(n)}\right)} \left(1 + erf\left(\frac{-x+B+\gamma_y V_r^{(n)} - \frac{C^2}{\lambda_r}}{\sqrt{2}C} \right) \right) \\ &\left. - e^{\frac{1}{\lambda_r}\left(-x+B+\gamma_y V_l^{(n)}\right)} \left(1 - erf\left(\frac{-x+B+\gamma_y V_l^{(n)} + \frac{C^2}{\lambda_r}}{\sqrt{2}C} \right) \right) \right] + \frac{1}{4\lambda_r K} \cdot \\ &e^{\frac{\sigma_d^2}{2\lambda_r^2}} \left[e^{\frac{1}{\lambda_r}(x-\mu_d)} \left(1 + erf\left(\frac{-x+\mu_d - \frac{1}{\lambda_r}\sigma_d^2}{\sqrt{2}\sigma_d} \right) \right) + e^{-\frac{1}{\lambda_r}(x-\mu_d)} \left(1 - erf\left(\frac{-x+\mu_d + \frac{1}{\lambda_r}\sigma_d^2}{\sqrt{2}\sigma_d} \right) \right) \right]\end{aligned} \tag{5.56}$$

which is simplified as well by using the definition of $\zeta(x,\omega,\upsilon)$ in (5.52), giving the result:

$$\begin{aligned}\widetilde{p}_{(0)}(x) = \frac{1}{4\lambda_r}\cdot e^{\frac{\sigma_d^2}{2\lambda_r^2}}\cdot\Bigg[e^{\frac{1}{\lambda_r}(x+\mu_d)}\left(1+\zeta(x,-\mu_d-\frac{1}{\lambda_r}\sigma_d^2,\sigma_d)\right)+\\ e^{-\frac{1}{\lambda_r}(x+\mu_d)}\left(1-\zeta(x,-\mu_d+\frac{1}{\lambda_r}\sigma_d^2,\sigma_d)\right)\Bigg]\end{aligned} \tag{5.57}$$

According to (5.20), the overall distribution of total noise-induced threshold voltage shifts, i.e., $\widetilde{p}(x)$ is given in (5.58).

$$\begin{aligned}\widetilde{p}(x) =&\frac{1}{K}\times\sum_{n=0}^{K-1}\widetilde{p}_{(n)}(x)\\ =&\frac{1}{2\gamma_y\Delta V_{pp}K}\cdot\sum_{n=1}^{K-1}\left[\zeta(x,B+\gamma_y V_r^{(n)},C)-\zeta(x,B+\gamma_y V_l^{(n)},C)\right]+\\ &\frac{e^{\frac{C^2}{2\lambda_r^2}}}{4\gamma_y\Delta V_{pp}K}\cdot\sum_{n=1}^{K-1}\Bigg[e^{\frac{1}{\lambda_r}\left(-x+B+\gamma_y V_r^{(n)}\right)}\left(1-\zeta(x,B+\gamma_y V_r^{(n)}+\frac{C^2}{\lambda_r},C)\right)\\ &+e^{-\frac{1}{\lambda_r}\left(-x+B+\gamma_y V_l^{(n)}\right)}\left(1+\zeta(x,B+\gamma_y V_l^{(n)}-\frac{C^2}{\lambda_r},C)\right)\\ &-e^{-\frac{1}{\lambda_r}\left(-x+B+\gamma_y V_r^{(n)}\right)}\left(1+\zeta(x,B+\gamma_y V_r^{(n)}-\frac{C^2}{\lambda_r},C)\right)\\ &-e^{\frac{1}{\lambda_r}\left(-x+B+\gamma_y V_l^{(n)}\right)}\left(1-\zeta(x,B+\gamma_y V_l^{(n)}+\frac{C^2}{\lambda_r},C)\right)\Bigg]\\ &+\frac{1}{4\lambda_r K}\cdot e^{\frac{\sigma_d^2}{2\lambda_r^2}}\Bigg[e^{\frac{1}{\lambda_r}(x+\mu_d)}\left(1+\zeta(x,-\mu_d-\frac{1}{\lambda_r}\sigma_d^2,\sigma_d)\right)\\ &+e^{-\frac{1}{\lambda_r}(x+\mu_d)}\left(1-\zeta(x,-\mu_d+\frac{1}{\lambda_r}\sigma_d^2,\sigma_d)\right)\Bigg]\end{aligned} \tag{5.58}$$

Since channel parameters (specifically B, C, σ_d, and μ_d) vary for different levels in the flash channel, the overall noise density functions with respect to all K levels (denoted as $L_0 \rightarrow L_{K-1}$) are not exactly the same. Denote the probability density functions of noise-induced threshold voltage shifts for $L_0 \rightarrow L_{K-1}$ as $\widetilde{p}^{(k)}(x), (0 \leq k \leq K-1)$. Based on 2 bits/cell NAND flash memory and the parameters presented in the literature [15], we simulated the distributions of noise-induced threshold voltage shifts and compared the results with the theoretical counterparts expressed in (5.58).

Simulations: Using the simulation model presented in Section 5.2.1, we evaluate the noise sources and threshold voltage distributions derived above. The NAND flash memory used in these simulations has the page size of 4K bytes and the block size of 256 pages. The coupling ratio is set and fixed as 0.08. Normalized parameters for erase state are 1.4 volts and 0.35 for μ_e and σ_e, respectively. All blocks in the NAND flash memory are erased before programming. The step voltage used for ISPP programming is set as 0.2. The three bit combinations: 10, 00 and 01 are respectively programmed to the three threshold voltage levels: 2.6 volts, 3.2 volts and 3.93 volts, where the 11 corresponds to the erase state that will not be changed during the programming process.

Setting the P/E cycling number N to 1000, and the retention time to 1 year, we get the probability density function based on (5.58) for channel noise sources occurring in all four levels, respectively, as illustrated in Figure 5.6.

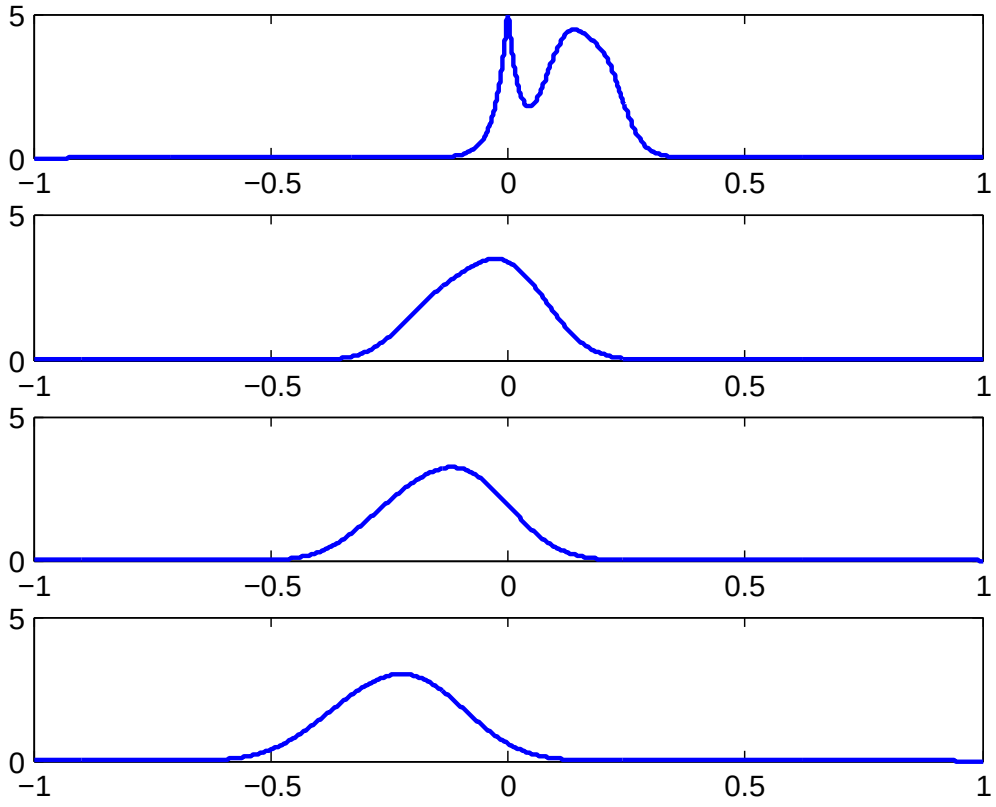


Fig. 5.6 The effects of RTN, CCI, and retention noise on cell threshold voltage distribution after 1K P/E cycling and 1 year retention (from top to bottom: L_0 to L_3).

Accordingly, we carry out Monte Carlo simulations to obtain the cell threshold voltage distribution at different levels under 1000 P/E cycling and 1 year retention

limit, as shown in Figure 5.7, which gives nearly the same results as in Figure 5.6. A close resemblance supports the correctness of the above theoretical derivations on the noise-induced threshold voltage shifts.

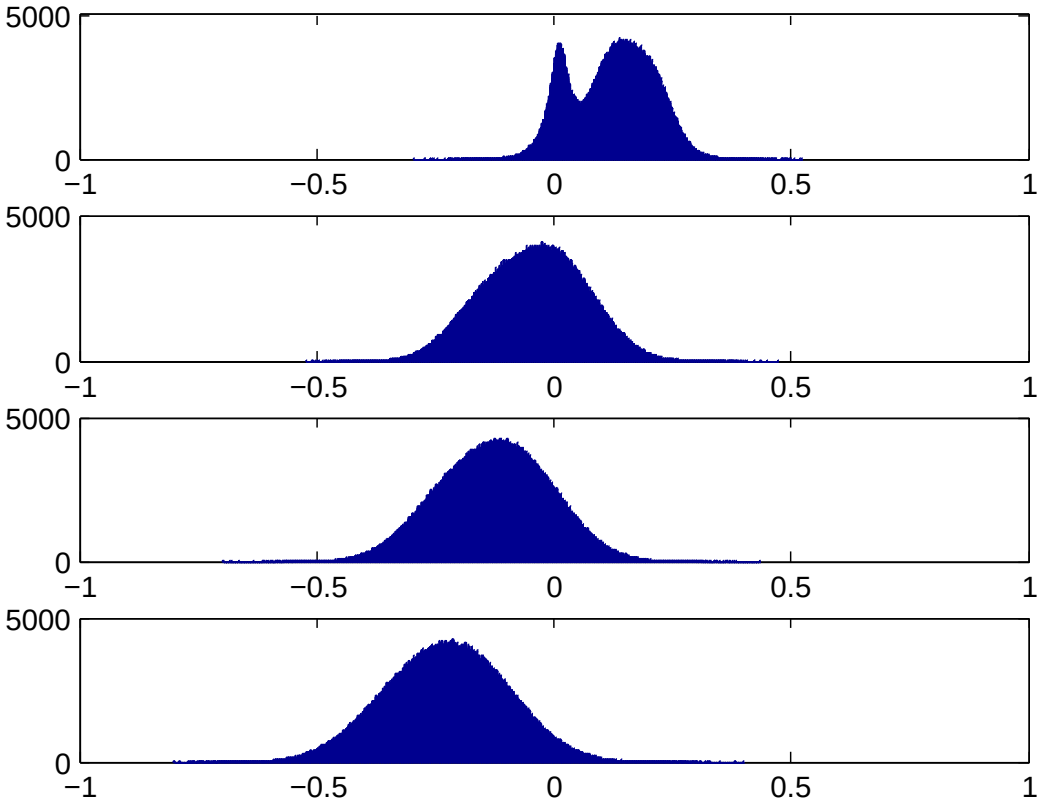


Fig. 5.7 Simulated curves to show the effects of RTN, CCI, and retention noise on memory cell threshold voltage distribution after 1K P/E cycling and 1 year retention.

5.3.2 Cell Threshold Voltage Distributions

In this section, the threshold voltage distribution of each level is determined, which can be used to calculate the log-likelihood ratio (LLR) of each bit stored in the memory, and further used for flash coding [13].

As shown in Figure 5.6, the curves for $L_1 \rightarrow L_3$ are bell-shaped, which is due to the fact that retention noise becomes dominant as the P/E cycling number and the retention time increase. Therefore, we intuitively expect a Gaussian distribution will be a good

fit for these probability density functions. The formulas below give the curve-fitting parameters of the proposed Gaussian distribution $(1 \leq k \leq K-1)$:

$$\mu_{(k)} = \int_{-\infty}^{+\infty} x \cdot \widetilde{p}^{(k)}(x)dx. \tag{5.59}$$

$$\sigma_{(k)}^2 = \int_{-\infty}^{+\infty} (x-\mu_{(k)})^2 \cdot \widetilde{p}^{(k)}(x)dx. \tag{5.60}$$

Therefore, the probability density function of the curve-fitted Gaussian distributions for $L_1 \rightarrow L_{K-1}$ are given by

$$g^{(k)}(x) = \frac{1}{\sqrt{2\pi}\sigma_{(k)}} e^{\frac{(x-\mu_{(k)})^2}{2\sigma_{(k)}^2}}, 1 \leq k \leq K-1. \tag{5.61}$$

Recall the initial distributions of $L_1 \rightarrow L_{K-1}$ after ideal programming in (2.3), we can obtain the final probability density functions, denoted as $p^{(k)}(x)$, by computing the convolution of $g^{(k)}(x)$ and $p_i^{(k)}(x)$. The results are given below:

$$\begin{aligned} p^{(k)}(x) &= g^{(k)}(x) \otimes p_i^{(k)}(x) = g^{(k)}(x) \otimes p_p^{(k)}(x) = \int_{-\infty}^{+\infty} p_p^{(k)}(\tau) g^{(k)}(x-\tau)d\tau \\ &= \int_{V_l^{(k)}}^{V_r^{(k)}} \frac{1}{\Delta V_{pp}} \cdot \frac{1}{\sqrt{2\pi}\sigma_{(k)}} e^{-\frac{\left(x-\tau-\mu_{(k)}\right)^2}{2\sigma_{(k)}^2}} d\tau \\ &= \frac{1}{2\Delta V_{pp}} \left[\zeta(x, V_r^{(k)} + \mu_{(k)}, \sigma_{(k)}) - \zeta(x, V_l^{(k)} + \mu_{(k)}, \sigma_{(k)}) \right], 1 \leq k \leq K-1 \end{aligned} \tag{5.62}$$

Since the retention noise has minor influence on the threshold voltage of L_0, the probability density function of L_0 shown in Figure 5.6 is not bell-shaped. Thus it would not be correct to use a Gaussian distribution for curve fitting in this case. However, the final distribution of L_0 can be easily determined with the results in (5.58). To this end, the two parameters: μ_d and σ_d in (5.58), should be revised as follows:

$$\begin{cases} \mu_d' = \mu_d - \mu_e \\ \sigma_d' = \sqrt{\sigma_e^2 + \sigma_d^2} \end{cases} \tag{5.63}$$

Accordingly, B and C, should be revised as well as follows.

$$\begin{cases} B' = -\mu_d' - \mu_e\gamma_y = -\mu_d - \mu_e\gamma_y + \mu_e \\ C' = \sqrt{\sigma_e^2\gamma_y^2 + \sigma_d'^2} = \sqrt{\sigma_e^2\gamma_y^2 + \sigma_d^2 + \sigma_e^2} \end{cases} \tag{5.64}$$

Consequently, the final threshold voltage distribution of L_0, i.e. $p^{(0)}(x)$, can be calculated in (5.65).

$$\begin{aligned} p^{(0)}(x) =& \widetilde{p}^{(0)}(x) \otimes p_e(x) \\ =& \frac{1}{2\gamma_y \Delta V_{pp} K} \cdot \sum_{n=1}^{K-1} \left[\zeta(x, B' + \gamma_y V_r^{(n)}, C') - \zeta(x, B' + \gamma_y V_l^{(n)}, C') \right] + \\ & \frac{e^{\frac{C'^2}{2\lambda_r^2}}}{4\gamma_y \Delta V_{pp} K} \cdot \sum_{n=1}^{K-1} \left[e^{\frac{1}{\lambda_r}\left(-x+B'+\gamma_y V_r^{(n)}\right)} \left(1 - \zeta(x, B' + \gamma_y V_r^{(n)} + \frac{C'^2}{\lambda_r}, C') \right) \right. \\ & + e^{-\frac{1}{\lambda_r}\left(-x+B'+\gamma_y V_l^{(n)}\right)} \left(1 + \zeta(x, B' + \gamma_y V_l^{(n)} - \frac{C'^2}{\lambda_r}, C') \right) \\ & - e^{-\frac{1}{\lambda_r}\left(-x+B'+\gamma_y V_r^{(n)}\right)} \left(1 + \zeta(x, B' + \gamma_y V_r^{(n)} - \frac{C'^2}{\lambda_r}, C') \right) \\ & \left. - e^{\frac{1}{\lambda_r}\left(-x+B'+\gamma_y V_l^{(n)}\right)} \left(1 - \zeta(x, B' + \gamma_y V_l^{(n)} + \frac{C'^2}{\lambda_r}, C') \right) \right] \\ & + \frac{1}{4\lambda_r K} \cdot e^{\frac{\sigma_d'^2}{2\lambda_r^2}} \left[e^{\frac{1}{\lambda_r}(x+\mu_d')} \left(1 + \zeta(x, -\mu_d' - \frac{1}{\lambda_r}\sigma_d'^2, \sigma_d') \right) \right. \\ & \left. + e^{-\frac{1}{\lambda_r}(x+\mu_d')} \left(1 - \zeta(x, -\mu_d' + \frac{1}{\lambda_r}\sigma_d'^2, \sigma_d') \right) \right] \end{aligned} \tag{5.65}$$

The aforementioned 4-level MLC flash and the related channel parameters have been used as well in this section to show the probability density functions derived above. The P/E cycling number is set to 1000 and retention time to 1 year. Figure 5.8 illustrates the final threshold voltage distributions of all four levels according to (5.62) and (5.64).

As shown, the channel modeled in this work is quite similar to the models used previously in the literature [15, 31, 62, 71]. Nonetheless, as only reasonable Gaussian approximation is used and all the channel noise have been considered in the derivations above, the model proposed here is a more accurate representation of the real flash channel. Moreover, the exact formula of cell threshold voltage distribution, i.e., the

results given in (5.64), makes the soft decisions calculated based on our work more accurate for the flash decoding.

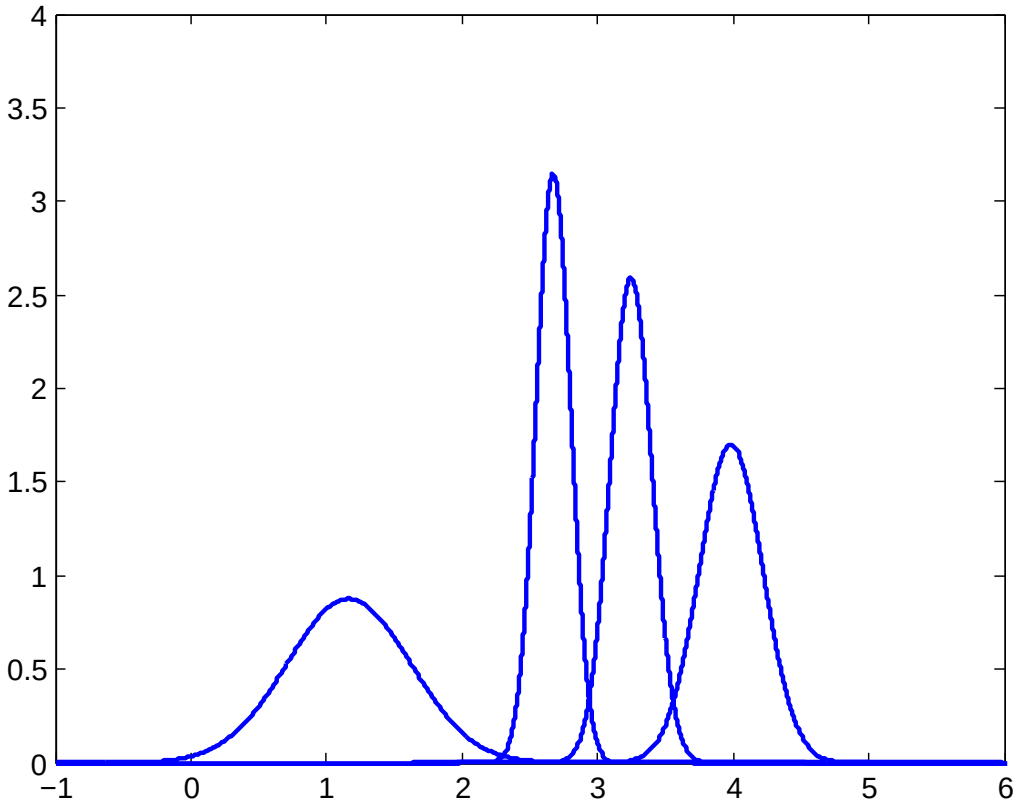


Fig. 5.8 Cell threshold voltage distributions 4-level MLC flash memory

5.4 Calculation of Soft Decisions

Soft decisions, or in other words, the log-likelihood-ratios (LLRs) information are critical in decoding the advanced flash error correction codes. The aim of this work is to characterise the flash memory channel and propose a way of generating precise soft information for the error correction of NAND based systems. In this section, we propose a mathematical formulation for calculating LLRs based upon the threshold-voltage distributions presented above, and the results are used to inform the practical design and implementation of decoding schemes.

Assume each bit is programmed to a flash cell with equal probability, i.e., all the storage levels in one memory cell have equal *a priori* probability, and let V_{th} represent

the sensed threshold voltage of one memory cell. We can calculate the LLR of the ith bit stored in one cell as

$$L(b_i) = \log \frac{p(b_i = 1|V_{th})}{p(b_i = 0|V_{th})} = \log \frac{p(V_{th}|b_i = 1)}{p(V_{th}|b_i = 0)} \tag{5.66}$$

For K levels MLC flash memory, there are $N_b = \log_2(K)$ bits stored in each cell, and the probability density function of the threshold voltage of kth storage level, $p^{(k)}(x), 0 \leq k \leq K-1$, is known from (5.62) and (5.65). Let $\mathbb{S}_i$ denote the set of levels whose ith bit is 1. Hence, given the threshold voltage V_{th} of a cell, the LLR of each bit would be calculated as

$$L(b_i) = \log \frac{\sum_{k \in \mathbb{S}_i} p^{(k)}(V_{th})}{\sum_k p^{(k)}(V_{th}) - \sum_{k \in \mathbb{S}_i} p^{(k)}(V_{th})} \tag{5.67}$$

With respect to practical NAND flash memories, the cell threshold voltages cannot be obtained precisely while being sensed by the comparison with a series of reference voltages. Assume that the threshold voltage V_{th} falls into the range $(R_l, R_r]$ (where R_l and R_r are two adjacent reference voltages), we can estimate the corresponding LLR of the ith bit as

$$L(b_i) = \log \frac{\int_{R_l}^{R_r} \sum_{k \in \mathbb{S}_i} p^{(k)} V_{th}}{\int_{R_l}^{R_r} \sum_k p^{(k)} V_{th} - \int_{R_l}^{R_r} \sum_{k \in \mathbb{S}_i} p^{(k)} V_{th}} \tag{5.68}$$

We assume the parameters σ_e, μ_e, σ_d, μ_d in (5.62) and (5.65) are known and treat the coupling ratio γ_y as a known constant. As the number of reference voltages in memory sensing is fixed, all possible LLRs can be calculated before the decoding starts. Therefore, only a lookup table is required at run time to obtain the LLRs, which reduces the system complexity and improves the speed and throughput. Let K_s denote the number of reference voltages being used in memory sensing, the LLR lookup table only contains $N_b(K_s + 1)$ entries for N_b-bit/cell NAND flash memory [16].

5.5 Summary and Challenges

In this chapter, we are concerned with the practical use of soft-decision ECCs in NAND flash based memory systems. The decoders of these ECCs require soft-decision LLR information, which demands the availability of a precise cell threshold-voltage distribution model and fine-grained soft-decision memory sensing. Since fine-grained memory-cell sensing tends to incur significant implementation overhead and access latency penalty, it is critical to minimize the quantisation levels while keeping the same error correction performance.

Additionally, since the RTN, CCI and retention noise predominantly contribute to memory-cell threshold-voltage distribution distortion, they must be carefully incorporated into the memory-cell threshold-voltage distribution model. Only by taking all these three types of distortion sources together is it possible to show a precise threshold voltage distribution in the channel outputs. In this chapter, we have derived the probability density function for the channel noise sources on the basis of MLC NAND flash memories, and formulated the final threshold voltage distribution for each cell level. This was possible by using characteristic functions for the various sources of noise.

The results presented here are novel in a theoretical way as the first time the flash channel has been characterised by incorporating the distributions of all sources of interference and noise. The channel model is shown with the threshold voltage distributions corresponding to all MLC levels. Based on the results we present a way to calculate LLRs, which can be used for on-chip soft-decisions extraction in the implementation of NAND flash memories. Using the 4-level MLC flash as a case study, we plot the final threshold voltage distributions and compare the curves to Monte Carlo simulations of the flash channel. The results of the theoretical analysis have been validated by various experiments.

Modelling a communication channel inevitably involves the convolutional computations of noise distributions. Although the noise distributions can be characterised in statistical terms, the computational complexities are still a major obstacle to the modelling of a precise channel. The methods of characteristic functions in our work

might be a promising technique. Meanwhile, the same approach can be extended if more noise forms are investigated in the flash memory channel.

In the study of channel modelling, we have tried to address the application of advanced ECCs using the results in this chapter, however, implementing the error correction coding also involves the programming and sensing circuits design, which are other topics associated with flash coding. Throughout this work these two modules have been relatively simplified so that we can emphasize the channel modelling and distributions of the cell threshold voltages. The application of LDPC or turbo codes using channel model presented here is very important and it would surely be pursued for future work.

Chapter 6

Conclusions and Future Work

6.1 Thesis Conclusion

The main goal of this dissertation is to develop a variety of design techniques that can tolerate different distortion and noise sources in NAND flash memory. We studied the reliability issue from the top level to the physical level of memory systems, and investigated two major research topics: write patterns processing and error correction codes for flash memories. By analysing the characteristics of NAND flash storage, some existing error correction theories in digital communication can be modified and efficiently used for the flash coding applications. Based on such connection, this dissertation also proposes and evaluates the error correction schemes to improve the reliability of flash based data storage.

Three different phases of research have been carried out in this dissertation.

In the first phase of this dissertation, we studied the pre-coding mechanism for NAND flash memory, which is basically the high level processing in the storage system. To unevenly distribute the data bits to the multiple levels of flash memory cells, we proposed a write pattern formatting algorithm for the processing of the data programming patterns. By incorporating a power saving stage, the proposed WPFA further reduces the energy consumption and eliminates long column strip patterns to protect the memory systems from current spikes. WPFA is not an error correction algorithm but is an important mechanism to support ECCs to improve the reliability of NAND flash memory systems.

The algorithm is optimized to combine the upper with lower page data processing together to achieve a small overhead of NAND cell area and simplified hardware implementation. The circuitry for the realization of the WPFA is described in this dissertation as well with an analysis of the resource utilization in the design practise.

Assuming that a good pre-coding scheme is employed, the second phase of research explored an efficient advanced error correction mechanism that can embrace the read errors while retrieving data from flash memories. This is motivated by the fact that NAND flash memory will rely heavily on the error correction codes that ensure overall system storage integrity and the ECCs have been used in nearly all the NAND based memory systems at present. A concatenated LDPC-TCM coding scheme was proposed in this dissertation to tolerate system-level faults in NAND flash memories. TCM is a technique used widely in telecommunications but is well suited here for memory system application due to the similarity between MLC flash and PAM modulation. TCM improves the error correction performance with low complexity by considering multi-level signal modulation and convolutional error correction code construction. Meanwhile, the TCM decoder provides soft bit decisions at the output stage which allows the use of soft decisions based error correction codes, which normally achieve better performance than the hard decisions based ECCs, such as BCH and Hamming codes. While the present ECC strategy is mostly BCH coding due to the fact that flash memory itself is limited to providing only hard decisions, the approach proposed in this dissertation gives developers a good way to realise advanced error correction coding. Since the proposed scheme originates from the technique in digital communications, we also investigated and analysed the coding gain achieved in MLC flash memories. Using pragmatic TCM and the (17664, 16384) irregular LDPC codes as the example, we carried out extensive simulations to demonstrate the performance of the proposed error correction scheme.

In the last phase of research, we directed attention to the physical level of the NAND flash memory and attempted to discover some intrinsic factors that influence the performance of error correction coding or write patterns processing employed in

higher level flash systems. An NAND flash memory device model was proposed which can account for the influence of erase/program operations, random telegraph noise, cell-to-cell interference, P/E cycling and the data retention effect. Using probability theory, we characterised the memory channel, investigated its behaviours, and derived the probability density functions for channel noise sources and threshold voltages. The characteristic function for each distribution is used in the channel modelling to simplify the mathematical derivations. Although this channel model is not absolutely accurate, we believe this approximate model can serve as a good vehicle to research the flash memory by explicitly capturing several major memory cell storage noise sources. In the meantime, the mathematical formulas derived for the final threshold voltage distributions provides a way to calculate the soft decisions from the theoretical perspective. The results can be revised to meet the requirements of practical designs by considering the sensing quantisations, which has been discussed in this dissertation as well. The Monte Carlo simulations of the hypothetical 2 bits/cell NAND flash memory has demonstrated the correctness of the probability density functions and threshold voltage distributions obtained in our work.

6.2 Directions for Future Work

There are still several challenges to be addressed on the topic of flash memory data reliability. Firstly, this dissertation focused on the one-dimensional implementation of the LDPC-TCM scheme. However, it is well known that multi-dimensional modulations in telecommunications can achieve better bit efficiency without increasing the hardware complexity. Considering the multiple flash cells to be a similar carrier as the multiple symbols, we believe it should be possible to concatenate multidimensional TCM with the LDPC coding for the flash error correction in a similar way as presented in this dissertation. For example, by combining 4 memory cells together to generate the TCM signal, we can realise the 4-D TCM. A better understanding can be developed to both qualitatively and quantitatively justify the effectiveness of this technique.

In addition, it is worth exploring another direction as the future work on the TCM-LDPC coding, which is to apply the scheme to the well developed channel model proposed in Chapter 5. In this dissertation we investigated the TCM-LDPC scheme with a channel model similar to M-PAM modulation, in which way we can readily migrate the theories in wireless communications to the flash memory research. If using the model proposed in Chapter 5, we can use for TCM decoding the threshold voltage distributions with all major distortion sources considered. An important obstacle for this work would be the decoding for TCM, which currently works very well on the PAM cases but was uncertain for different modulations. Thus, the steps for the research in this direction is to discover a way to decode the TCM on flash channel first, followed by a straightforward application of LDPC coding.

Current NAND flash products rely on BCH codes as ECC solution, but more and more errors discovered in future flash memories will be corrected by the soft-decision ECC. Although two ways of extracting the soft information have been proposed and the decoding of soft-decision ECC is not an obstacle, NAND flash products suffer a lot from soft-decision sensing. Not only the sensing latency but also data transfer latency is increased significantly. Uniform quantisation is simple but requires too many reading levels, thus introduces high overhead to the flash sensing circuits. As a result, it is important to have an optimized quantisation scheme to lower the complexity of flash sensing circuits. One possible solution is to decide the optimised sensing levels by maximising the mutual information. Applying the entropy encoding to soft-decision sensing results would produce a non-uniform quantisation with fewer levels and less latency. Therefore, the optimised quantisation, aiming to reduce the sensing levels and improve soft-decision error correction, can be considered as a possible future work.

In Chapter 3, we considered the FPGA implementation for the hardware realisation of the proposed algorithm. The resource utilisation on registers and logic modules will be consistent to the practical WPFA implementation in the flash controller. However, more and more NAND-based products use an application-specific integrated circuit (ASIC) to control the flash memory arrays and manage the flash translation layer algorithms.

The ASIC implementation is similar to FPGA using logic gates as the major hardware resources but has lower flexibility and longer development cycle. However, the low cost, low power and high speed are three advantages making the ASIC very attractive in the products design. In this regard, the ASIC implementation of the proposed WPFA would bring the benefits to the industry and promote the application of the algorithm. As a future work direction, the differences between FPGA and ASIC, the clock, high-speed serial, and power consumption can be studied comprehensively.

Finally, the soft-decision error correction is still an important future issue for the flash coding. Indeed, the ECCs are the best way at least currently to improve the reliability of flash memory systems. Regarding this, the channel model and threshold voltage distributions proposed in this dissertation would help a lot in proposing efficient coding schemes and applying the soft-decision ECCs such as LDPC and turbo codes. At the end of Chapter 5.3, the way to calculate soft information based on our distribution model was presented, which is considered as a processing module following the flash memory sensing circuits. As a future work, this module can be incorporated with the erase/programming and the quantisation in the error correction mechanism to improve the data reliability of the NAND flash based memory systems.

References

[1] Bahl, L., Cocke, J., Jelinek, F., and Raviv, J. (1974). Optimal decoding of linear codes for minimizing symbol error rate. *IEEE Trans. on Info. Theory*, 20(2):284–287.

[2] Barndt, R. D., Hu, X., and Weathers, A. D. (2012). LDPC Decoding For Solid State Storage Devices.

[3] Berrou, C., Glavieux, A., and Thitimajshima, P. (1993). Near shannon limit error-correcting coding and decoding: Turbo-codes. 1. In *1993 IEEE Int. Conference on Communications, ICC '93 Geneva*, volume 2, pages 1064–1070 vol.2.

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

[5] Bose, R. and Ray-Chaudhuri, D. (1960). On a class of error correcting binary group codes. *Information and Control*, 3(1):68 – 79.

[6] Campello, J. and Modha, D. (2001). Extended bit-filling and LDPC code design. In *IEEE GLOBECOM, 2001*, volume 2, pages 985–989.

[7] Campello, J., Modha, D., and Rajagopalan, S. (2001). Designing LDPC codes using bit-filling. In *IEEE ICC, 2001*, volume 1, pages 55–59.

[8] Chandrasetty, V. and Aziz, S. (2010). FPGA Implementation of High Performance LDPC Decoder Using Modified 2-Bit Min-Sum Algorithm. In *Computer Research and Development, 2010 Second International Conference on*, pages 881–885.

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

[10] Choi, H., Liu, W., and Sung, W. (2010). VLSI Implementation of BCH Error Correction for Multilevel Cell NAND Flash Memory. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 18(5):843–847.

[11] Compagnoni, C. M., Gusmeroli, R., Spinelli, A. S., Lacaita, A. L., Bonanomi, M., and Visconti, A. (2008). Statistical Model for Random Telegraph Noise in Flash Memories. *IEEE Trans. on Electron Devices*, 55(1):388–395.

[12] Darabiha, A., Carusone, A., and Kschischang, F. (2006). A bit-serial approximate min-sum LDPC decoder and FPGA implementation. In *IEEE International Symposium on Circuits and Systems, 2006*, page 4.

[13] Dong, G., Li, S., and Zhang, T. (2010). Using Data Postcompensation and Predistortion to Tolerate Cell-to-Cell Interference in MLC nand Flash Memory. *IEEE Trans. on Circuits and Systems I: Regular Papers*, 57(10):2718–2728.

[14] Dong, G., Li, Y., Xie, N., Zhang, T., and Liu, H. (2009). Candidate Bit Based Bit-flipping Decoding Algorithm for LDPC Codes. In *Proceedings of the 2009 IEEE International Conference on Symposium on Information Theory*, volume 3, pages 2166–2168. IEEE Press.

[15] Dong, G., Pan, Y., Xie, N., Varanasi, C., and Zhang, T. (2012). Estimating Information-Theoretical nand Flash Memory Storage Capacity and its Implication to Memory System Design Space Exploration. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 20(9):1705–1714.

[16] Dong, G., Xie, N., and Zhang, T. (2011). On the Use of Soft-Decision Error-Correction Codes in NAND Flash Memory. *IEEE Trans. on Circuits and Systems I: Regular Papers*, 58(2):429–439.

[17] Fazio, A. (2009). Future directions of non-volatile memory in compute applications. In *2009 IEEE Int. Electron Devices Meeting (IEDM)*, pages 1–4.

[18] Fukuda, K. and et al (2012). A 151-mm^2 64-Gb 2 Bit/Cell NAND Flash Memory in 24-nm CMOS Technology. *IEEE Journal of Solid-State Circuits*, 47(1):75–84.

[19] G, G. (2012). Charge-related phenomena and reliability of non-volatile memories. *Microelectronics Reliability*, 52(10):1876 – 1882.

[20] Gallager, R. (1962). Low-density parity check codes. *IRE Trans. Information Theory*, pages 21–28.

[21] Gallager, R. G. (2008). Principles of Digital Communication. *Cambridge University Press*.

[22] He, Z., Fortier, P., and Roy, S. (2006). A class of irregular LDPC codes with low error floor and low encoding complexity. *IEEE Comms Letters*, 10(5):372–374.

[23] Hu, X. (2012). LDPC codes for flash channel. In *in Proc. of Flash Memory Summit*.

[24] Hu, Y.-P., Xiao, N., and Liu, X.-F. (2013). An elastic error correction code technique for NAND flash-based consumer electronic devices. *IEEE Transactions on Consumer Electronics*, 59(1):1–8.

[25] Ielmini, D., Spinelli, A., Lacaita, A., and Visconti, A. (2002). Statistical profiling of SILC spot in flash memories. *IEEE Trans. on Electron Devices*, 49(10):1723–1728.

[26] Jung, H., Jung, S., and Song, Y. H. (2011). Architecture Exploration of Flash Memory Storage Controller Through a Cycle Accurate Profiling. *IEEE Transactions on Consumer Electronics*, 57(4):1756–1764.

[27] Kim, C., Rhee, S., Kim, J., and Jee, Y. (2010). Product Reed-Solomon Codes for Implementing NAND Flash Controller on FPGA Chip. In *Computer Engineering and Applications (ICCEA), 2010 Second International Conference on*, volume 1, pages 281–285.

[28] Kim, D., Choi, J., and Ha, J. (2012a). On the soft information extraction from hard-decision outputs in MLC NAND flash memory. In *2012 IEEE Global Communications Conference (GLOBECOM)*, pages 3208–3213.

[29] Kim, J., Lee, D., and Sung, W. (2012b). Performance of Rate 0.96 (68254, 65536) EG-LDPC Code for NAND Flash Memory Error Correction. In *IEEE International Conference on Communications (ICC)*, pages 7029–7033.

[30] Kim, J. and Sung, W. (2014). Rate-0.96 LDPC Decoding VLSI for Soft-Decision Error Correction of NAND Flash Memory. *IEEE Trans on Very Large Scale Integration (VLSI) Systems*, 22(5):1004–1015.

[31] Lee, D. and Sung, W. (2013). Estimation of NAND Flash Memory Threshold Voltage Distribution for Optimum Soft-Decision Error Correction. *IEEE Trans. on Signal Processing*, 61(2):440–449.

[32] Lee, J.-D., Choi, J.-H., Park, D., and Kim, K. (2004). Effects of interface trap generation and annihilation on the data retention characteristics of flash memory cells. *IEEE Trans. on Device and Materials Reliability*, 4(1):110–117.

[33] Lee, J.-D., Hur, S.-H., and Choi, J.-D. (2002). Effects of floating-gate interference on NAND flash memory cell operation. *IEEE Electron Device Letters*, 23(5):264–266.

[34] Lee, S., Park, J., Fleming, K., Arvind, and Kim, J. (2011). Improving Performance and Lifetime of Solid-State Drives Using Hardware-Accelerated Compression. *IEEE Transactions on Consumer Electronics*, 57(4):1732–1739.

[35] Li, S. and Zhang, T. (2010). Improving multi-level NAND flash memory storage reliability using concatenated BCH-TCM coding. *IEEE Trans. on VLSI Systems*, 18(10):1412–1420.

[36] Liu, H., Groothuis, S., Mouli, C., Li, J., Parat, K., and Krishnamohan, T. (2009). 3D Simulation Study of Cell-Cell Interference in Advanced NAND Flash Memory. In *IEEE Workshop on Microelectronics and Electron Devices, 2009*, pages 1–3.

[37] Lou, H.-L. and Sundberg, C.-E. (1998). Coded modulation to increase storage capacity of multilevel memories. In *IEEE Globecom*, pages 3379–3384.

[38] MacKay, D. and Neal, R. (1997). Near shannon limit performance of low density parity check codes. *Electronics Letters*, 33(6):457–458.

[39] Madhow, U. (2008). Fundamentals of Digital Communication. *Cambridge University Press*.

[40] Matthias, K. (2007). *Trellis Decoding: From Algorithm to Flexible Architectures*. PhD thesis, Lund University.

[41] McEliece, R., MacKay, D., and Cheng, J.-F. (1998). Turbo decoding as an instance of Pearl's "belief propagation" algorithm. *IEEE Journal on Selected Areas in Communications*, 16(2):140–152.

[42] Micheloni, R., Crippa, L., and Marelli, A. (2010). Inside NAND Flash Memories. *Springer Press*.

[43] Mielke, N., Belgal, H., Kalastirsky, I., Kalavade, P., Kurtz, A., Meng, Q., Righos, N., and Wu, J. (2004). Flash EEPROM threshold instabilities due to charge trapping during program/erase cycling. *IEEE Trans. on Device and Materials Reliability*, 4(3):335–344.

[44] Mielke, N., Belgal, H. P., Fazio, A., Meng, Q., and Righos, N. (2006). Recovery Effects in the Distributed Cycling of Flash Memories. In *44th Annual IEEE International Reliability Physics Symposium Proceedings, 2006*, pages 29–35.

[45] Mielke, N., Marquart, T., Wu, N., Kessenich, J., Belgal, H., Schares, E., Trivedi, F., Goodness, E., and Nevill, L. (2008). Bit error rate in NAND Flash memories. In *Reliability Physics Symposium, 2008. IRPS 2008. IEEE International*, pages 9–19.

[46] Moon, J., No, J., Lee, S., Kim, S., Choi, S., and Song, Y. (2013). Statistical Characterization of Noise and Interference in NAND Flash Memory. *IEEE Trans. on Circuits and Systems I: Regular Papers*, 60(8):2153–2164.

[47] Moon, J., No, J., Lee, S., Kim, S., and Yang, J. (2011). Statistical Analysis of Flash Memory Read Data. In *IEEE GLOBECOM*, pages 1–6.

[48] Motwani, R. and Ong, C. (2012). Robust decoder architecture for multi-level flash memory storage channels. In *in International Conference on Computing, Networking and Communications (ICNC)*, pages 492–496.

[49] Park, Y. and soo Kim, J. (2011). zFTL: Power-efficient Data Compression Support for NAND Flash-based Consumer Electronics Devices. *IEEE Transactions on Consumer Electronics*, 57(3):1148–1156.

[50] Peterson, W. W. and Weldon, E. J. (1972). Error-Correcting Codes, 2nd Edition. *The MIT Press*.

[51] Prall, K. (2007). Scaling Non-Volatile Memory Below 30nm. In *IEEE Non-Volatile Semiconductor Memory Workshop 2007*, pages 5–10.

[52] Proakis, J. and Salehi, M. (2007). Digital Communications, 5th Edition. *McGraw-Hill Education*.

[53] Reed, I. S. and Solomon, G. (1960). Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304.

[54] Ryan, W. E. (2003). An Introduction to LDPC codes. *CRC Handbook for Coding and Signal Processing for Recording Systems*.

[55] Sakurai, T. and Tamaru, K. (1983). Simple formulas for two- and three-dimensional capacitances. *IEEE Transactions on Electron Devices*, 30(2):183–185.

[56] Schlegel, C. and Perez, L. C. (1997). Trellis Coding. *IEEE Press*.

[57] Sha, J., Gao, M., Zhang, Z., Li, L., and Wang, Z. (2006). An FPGA Implementation of Array LDPC Decoder. In *IEEE Asia Pacific Conference on Circuits and Systems, 2006*, pages 1675–1678.

[58] Shibata, N., Maejima, H., Isobe, K., Iwasa, K., Nakagawa, M., Fujiu, M., Shimizu, T., Honma, M., Hoshi, S., Kawaai, T., and Kanebako, K. (2007). A 70nm 16Gb 16-level-cell NAND Flash Memory. In *2007 IEEE Symposium on VLSI Circuits*, pages 190–191.

[59] Shibata, N., Maejima, H., Isobe, K., Iwasa, K., Nakagawa, M., Fujiu, M., Shimizu, T., Honma, M., Hoshi, S., Kawaai, T., and Kanebako, K. (2008). A 70 nm 16 Gb 16-Level-Cell NAND flash Memory. *IEEE Journal of Solid-State Circuits*, 43(4):929–937.

[60] Shuhei, T., Yuki, Y., and Ken, T. (2012). Over-10x-Extended-Lifetime 76%-Reduced-Error Solid-State Drives (SSDs) With Error-Prediction LDPC Architecture and Error-Recovery Scheme. In *in IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 424–425.

[61] Suh, K.-D., Suh, B.-H., Lim, Y.-H., and et al (1995). A 3.3 V 32 Mb NAND Flash Memory with Incremental Step Pulse Programming Scheme. *IEEE Journal of Solid-State Circuits*, 30(11):1149–1156.

[62] Sun, F., Devarajan, S., Rose, K., and Zhang, T. (2006). Multilevel flash memory on-chip error correction based on trellis coded modulation. In *2006 IEEE Int. Symp. on Circuits and Systems (ISCAS 2006)*.

[63] Sun, F., Devarajan, S., Rose, K., and Zhang, T. (2007). Design of on-chip error correction systems for multilevel NOR and NAND flash memories. *IET Circuits, Devices and Systems*, 1(3):241–249.

[64] Tanakamaru, S., Hung, C., Esumi, A., Ito, M., Li, K., and Takeuchi, K. (2011). 95%-lower-BER 43%-lower-power Intelligent Solid-State Drive (SSD) with Asymmetric Coding and Stripe Pattern Elimination Algorithm. In *IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 204–206.

[65] Tanakamaru, S., Hung, C., and Takeuchi, K. (2012). Highly Reliable and Low Power SSD Using Asymmetric Coding and Stripe Bitline-Pattern Elimination Programming. *IEEE Journal of Solid-State Circuits*, 47(1):85–96.

[66] Tehrani, S. S., Mannor, S., and Gross, W. J. (2008). Fully Parallel Stochastic LDPC Decoders. *IEEE Trans on Signal Processing*, 56(11):5692–5703.

[67] Trinh, C. and et al (2009). A 5.6mb/s 64gb 4b/cell nand flash memory in 43nm cmos. In *IEEE International Conference on Solid-State Circuits*, pages 246–247.

[68] Ungerboeck, G. (1987). Trellis-coded modulation with redundant signal sets Part I: Introduction. *IEEE Com. Mag.*, 25(2):5–11.

[69] Viterbi, A., Wolf, J., Zehavi, E., and Padovani, R. (1989). A pragmatic approach to trellis-coded modulation. *IEEE Com. Mag.*, 27(7):11–19.

[70] Wang, J. (2012). *Absorbing Set Analysis of LDPC Codes and Read-Channel Quantization in Flash Memory*. PhD thesis, University of California Los Angeles.

[71] Wang, J., Courtade, T., Shankar, H., and Wesel, R. (2011). Soft information for LDPC decoding in flash: mutual-information optimized quantization. In *IEEE Globecom*, pages 1–6.

[72] Wang, J., Dong, G., Courtade, T. A., Shankar, H., Zhang, T., and Wesel, R. D. (2012a). LDPC Decoding with Limited-Precision Soft Information in Flash Memories. *CoRR*, abs/1210.0149.

[73] Wang, J., Dong, G., Zhang, T., and Wesel, R. D. (2012b). Mutual-Information Optimized Quantization for LDPC Decoding of Accurately Modeled Flash Data. *CoRR*, abs/1202.1325.

[74] Wang, J., Vakilinia, K., Chen, T.-Y., Courtade, T. A., Dong, G., Zhang, T., Shankar, H., and Wesel, R. D. (2014). Enhanced Precision Through Multiple Reads for LDPC Decoding in Flash Memories. *IEEE Journal on Selected Areas in Communications*, 32(5):880–891.

[75] Wei, L.-F. (1987). Trellis-coded modulation with multidimensional constellations. *IEEE Trans. on Information Theory*, 33(4):483–501.

[76] Wenzhe, Z., Guiqiang, D., Hongbin, S., Nanning, Z., and Tong, Z. (2012). Reducing latency overhead caused by using LDPC codes in NAND flash memory. *EURASIP Journal on Advances in Signal Processing*, 2012(1).

[77] Wu, G., Huang, P., and He, X. (2014). Reducing SSD access latency via NAND flash program and erase suspension. *Journal of Systems Architecture*, 60(4):345 – 356.

[78] Yaakobi, E., Grupp, L., Siegel, P., Swanson, S., and Wolf, J. (2012). Characterization and error-correcting codes for TLC flash memories. In *2012 International Conference on Computing, Networking and Communications (ICNC)*, pages 486–491.

[79] Yang, C., Emre, Y., and Chakrabarti, C. (2012). Product code schemes for error correction in mlc nand flash memories. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 20(12):2302–2314.

[80] Yang, J. (2012). Novel ECC architecture enhances storage system reliability. In *in Proc. of Flash Memory Summit*.

[81] Yeo, E. (2012). An LDPC-enabled flash controller in 40nm CMOS. In *in Proc. of Flash Memory Summit*.

[82] Yu, C., Haratsch, E. F., Mutlu, O., and Mai, K. (2013). Threshold Voltage Distribution in MLC NAND Flash Memory: Characterization, Analysis, and Modeling. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 1285–1290.

[83] Zhao, K., Zhao, W., Sun, H., Zhang, X., Zheng, N., and Zhang, T. (2013). LDPC-in-SSD: Making Advanced Error Correction Codes Work Effectively in Solid State Drives. In *11th USENIX Conference on File and Storage Technologies (FAST 13)*, pages 243–256, San Jose, CA.

[84] Zhong, H. and Zhang, T. (2003). Design of VLSI implementation-oriented LDPC codes. In *58th IEEE Vehicular Technology Conference, 2003*, volume 1, pages 670–673.

[85] Zhong, H. and Zhang, T. (2005). Block-LDPC: a practical LDPC coding system design approach. *IEEE Trans. on Circuits and Systems I: Regular Papers*, 52(4):766–775.

[86] Zhou, H., Jiang, A., and Bruck, J. (2011). Error-correcting schemes with dynamic thresholds in non-volatile memories. In *International Symposium on Information Theory*, pages 2143–2147.

List of Related Publications

[1] Quan Xu, Thomas M. Chen, Yupeng Hu and Pu Gong, "Write Pattern Format Algorithm for Reliable NAND-Based SSDs", *IEEE Trans. on Circuits and Systems II*, vol. 61, no. 7, pp. 516 - 520, July 2014.

[2] Quan Xu, Pu Gong, Thomas M. Chen, John Michael and Shancang Li, "Modelling and characterization of NAND flash memory channels", *Elsevier Journal of Measurement*, vol. 70, no. 0, pp. 225 - 231, Apr 2015.

[3] Quan Xu, Pu Gong and Thomas M. Chen, "Concatenated LDPC-TCM coding for reliable storage in multi-level flash memories", in *9th IEEE Int. Symposium on Communication Systems, Networks Digital Signal Processing (CSNDSP)*, pp. 166 - 170, July 2014.

[4] Yu Gan and Quan Xu, "An Improved SoS Method for Generating Multiple Uncorrelated Rayleigh Fading Waveforms", *IEEE Communications Letters*, vol. 14, no. 17, pp. 641 - 643, 10 pages, July 2010.

[5] Quan Xu and Jian-bing Lee, "An improved algorithm for large-capacity flash memory systems", *IEEE Youth Conference on Information Computing and Telecommunications (YC-ICT)*, pp. 307 - 310, Nov 2010.