CODING AND SIGNAL PROCESSING FOR NAND FLASH MEMORY

ADNAN ASLAM CHAUDHRY (G1203374L)

School of Electrical and Electronic Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

2017

Acknowledgments

First and foremost, thanks to Almighty Allah, the most gracious and the most merciful, for His showers of blessings which enabled me to successfully accomplish this research work.

I extend my profound thanks and gratitude to my supervisor Prof. Yong Liang Guan for his exemplary guidance, consistent patience and strong hand of support without which this research study would not have seen the light of the day. Acknowledgments, being an essential part of a thesis, generally carry a note of appreciation by the researcher. Not always, but in many cases these words appear to be more exaggeration than truth. It should, therefore, not be taken just as formality when I say that it was his high scholarship and strong expertise in the technical and administrative matters that helped me to a great extent in the successful completion of this research study. Research is a very time consuming and painstaking activity. Occasions came during the course of the study when I felt tired, confused and frustrated due to a number of difficulties. But it was always his encouragement that kept me motivated towards my research work. I consider myself very fortunate to have him as my supervisor.

I would like to express my heartfelt thanks and gratitude to my co-supervisor Prof. Kui Cai from Singapore University of Technology and Design for her extremely useful discussions and insightful comments on my research work. Despite being busy and in a different university, she always spared her time to advise me on my research. I am greatly indebted to her for sharing her invaluable ideas and providing consistent

guidance throughout the course of my research work.

I would also like to sincerely thank Dr. Guojun Han from Guangdong University of Technology, China, for his helpful discussions and suggestions on my work. I have benefited immensely from his solid knowledge on the LDPC code design and belief-propagation decoding.

Last but not the least, my thanks are also due to my wife Dr. Raheela Paras, who stood firmly with me in facing all the difficulties and challenges over the years. I will be failing in my duties if I do not thank my father Dr. M. Aslam Chaudhry, my mother Mrs. Fakhrunisa Memon and my mother-in-law Farida Memon, who always prayed for me while living all alone and facing problems of aging with patience. I am also thankful to my elder brother Umair Aslam and his wife Kanwal Umair, who consistently encouraged me to concentrate towards my studies. I wish to thank all my family and friends for their prayers and well wishes through all these years. Special thanks to my friend Sunil Shardha and his wife Dr. Lata, who welcomed me in Singapore on my arrival and remained very helpful during my stay here.

Long live NTU, your precious memories will ever be shining in my feelings and thoughts !!!

Adnan Aslam Chaudhry

May 2017, Singapore

Contents

Acknowledgments	iii
Contents	ix
Abstract	ix
List of Abbreviations	xi
List of Symbols	xiii
List of Figures	xv
List of Tables	xxi
1 Introduction	1
1.1 Data Storage Reliability Issues in NAND Flash Memory	2
1.2 Motivation and Objectives	5
1.3 Major Contributions of the Thesis	8
1.3.1 Read and Write Voltage Signal Design	8
1.3.2 Retention-Failure Recovery with Channel Update	9
1.3.3 Cell-to-Cell Interference (CCI) Cancellation	10
1.3.4 Improved Belief-Propagation (BP) Scheduling	11
1.4 Organization of the Thesis	13
2 An Overview of NAND Flash Memory	15
2.1 NAND Flash Memory Organization	15
2.2 Basic NAND Flash Memory Operations	16
2.2.1 Program/Erase Operation	17
2.2.2 Read Operation	18
2.3 NAND Flash Memory Channel Model	19
2.3.1 Programming Noise	20
2.3.2 Cell-to-Cell Interference (CCI)	21
2.3.3 Random Telegraph Noise (RTN)	22
2.3.4 Data Retention Noise	23
2.3.5 Overall Cell Threshold Voltage Distribution	23
2.4 Chapter Summary	24

3	Read and Write Voltage Signal Optimization	27
3.1	Background	28
3.2	Write Voltage Signal Optimization	31
3.3	Quantization Design for the Read Threshold Voltage Levels	35
3.3.1	Entropy Quantization Scheme	37
3.3.2	Controlled Erasure Decoding using Entropy Quantization	43
3.4	Write Voltage Optimization for Faster Decoding Convergence	46
3.5	Early vs. End-of-Retention (EOR) Write Voltage Optimization	48
3.6	Chapter Summary	49
4	Decision-Directed Retention-Failure Recovery with Flash Channel Update	51
4.1	Background	52
4.1.1	Retention Effect on Cell's Voltage Distribution	53
4.1.2	Related Work	55
4.2	Retention-Failure Recovery via RABP Decoding	57
4.2.1	Retention-Error Correction	59
4.2.2	Overall Error-Rate Performance of RABP Decoding	62
4.2.3	Average Complexity and Storage Requirement for RABP Decoding	63
4.3	RABP-Assisted Channel Update (RA-CU)	64
4.3.1	Channel Update after RABP Decoding Failure	68
4.4	Performance Evaluation of RA-CU Algorithm	70
4.4.1	Average Estimation Error	70
4.4.2	Estimation Algorithm Complexity	71
4.4.3	Error-Rate Performance	73
4.5	Chapter Summary	74
5	Neighbor-A-Priori Detector for Cell-to-Cell Interference (CCI) Cancellation	77
5.1	Background	78
5.1.1	CCI Statistical Model	79
5.1.2	CCI Effect on Cell's Voltage Distribution	81
5.1.3	Related Work	82
5.2	Neighbor- <i>A-Priori</i> Detector	84
5.2.1	The Prior-art Post-Compensation Detector	85
5.2.2	The Proposed Even- <i>A-Priori</i> (Even-AP) Detector	86
5.2.3	The Proposed All- <i>A-Priori</i> (All-AP) Detector	88
5.2.4	The Proposed All- <i>A-Priori</i> CCR (All-AP-CCR) Detector	89
5.2.4.1	Estimation of Mean CCR Values	91
5.3	Performance of Neighbor- <i>A-Priori</i> Detection Schemes	93
5.3.1	Normalized Estimation Error	93
5.3.2	Symmetric Information Rate	94
5.3.3	Detector Error-Rate Performance	95

5.3.4	Complexity and Design Trade-off	96
5.4	A Joint Application of Techniques Proposed in Chapter 3, 4 and 5	98
5.5	Chapter Summary	101
6	Improved Fixed and Dynamic Scheduling for Belief-Propagation (BP) Decoding	103
6.1	Background	104
6.1.1	Overview of Belief-Propagation (BP) Decoding	105
6.1.2	Related Work	106
6.2	Column-Weight Scheduling (CWS)	108
6.2.1	Aggregated Message Count	108
6.2.2	Mutual Information Increment	111
6.3	Informed Fixed Scheduling (IFS)	112
6.3.1	Maximum Variable Node Counter	114
6.4	Edge-based Shuffled (e-Shuffled) Scheduling	116
6.4.1	Graph Edge Convergence in BP Decoding	117
6.4.2	Selection of Reliability Threshold	120
6.4.3	Avoiding Code Graph Cycles	122
6.5	Dynamic Silent-Variable-Node-Free Scheduling (D-SVNFS)	125
6.6	Simulation Results	128
6.6.1	Average Decoding Complexity	130
6.6.2	Error-Rate and Decoding Convergence	132
6.6.3	Semi-Serial Scheduling by employing CWS, IFS and e-Shuffled Schemes	134
6.7	Combining Voltage Optimization and Scheduling Schemes	137
6.8	Chapter Summary	139
7	Conclusions and Future Work	141
7.1	Conclusions	141
7.2	Future Work	145
	Author's Publications	149
	Bibliography	151
A	Derivation of hard-decision levels R_1 (3.9), R_2 (3.10) and R_3 (3.11)	161
B	Derivation of Check-node-type probability p (Type-2) (6.13) and p (Type-3) (6.14)	163

Abstract

NAND flash memory is a ubiquitous storage medium which has revolutionized the non-volatile memory industry by offering large storage capacity, high data throughput, fast read-response time and low power consumption. This has become possible mainly because of the advanced manufacturing processes, which have pushed the device scaling to its limits, and progressive read/write methods, such as the multi-level-cell (MLC) technology that enables multiple bits to be stored over a single memory cell. As the flash memory cells are integrated closer to each other, several channel impairment effects have emerged, which tend to degrade the reliability and endurance of flash memory system. This thesis is devoted to the design and analysis of new coding and signal processing solutions that can improve the reliability and endurance of the NAND flash memory.

This thesis first investigates the non-stationary and asymmetrical behavior of major channel impairment effects in a model-based NAND flash channel, and presents optimization methods to adjust the read and write voltage levels to adapt to this non-stationary flash channel. The proposed optimization scheme is able to determine the write voltage levels that minimize the channel raw error probability throughout the operational lifetime of NAND flash memory, for different program/erase (PE) cycles and data retention time. Next, to enhance soft LDPC decoding performance while minimizing complexity, a novel entropy based quantization scheme is introduced to design the read voltage levels. Then, by joint read and write voltage signal optimization, the decoded error-rate performance is

minimized and the decoding convergence speed is also improved.

In NAND flash memory, the data retention noise induces cell voltage down-drift due to charge leakage and causes catastrophic decoding failure due to outdated bit boundaries. To recover from the retention noise-induced decoding failure, this thesis presents a two-stage retention-aware belief-propagation (RABP) decoding scheme. The most prominent advantage of this approach compared to the prior art is that new channel estimation which requires time- and power-consuming memory sensing operations can be avoided. In RABP decoding, the probable victim cells are determined with the help of read-back voltage signal and the decoded bit decision. For such suspected victim cells, their log-likelihood-ratio (LLR) regions are modified in such a way as to absorb the effect of cell voltage downshift caused by retention noise, and then a second round of BP decoding is performed afresh, often with decoding failure recovery. Furthermore, leveraging on the RABP decoded bit-error pattern, this thesis further develops a decision-directed channel update algorithm to re-estimate the latest cell voltage distribution parameters.

Cell-to-cell interference (CCI) is yet another well-recognized source of errors in new generation NAND flash memories. This thesis exploits the data-dependent nature of CCI and proposes three post-processing schemes for its mitigation. The main idea is to remove (clean) the CCI component from the interfering cells before CCI cancellation from the victim cell. The proposed detection schemes are shown to outperform the existing CCI cancellation schemes.

As the LDPC code is becoming the mainstream error-correcting code for NAND flash-based data storage, the application of iterative BP decoding incurs long processing latency which threatens to overshadow the benefits of NAND flash technology. Thus, LDPC codes for flash memories are desired to have fast decoding convergence so as to minimize the memory-read latency. This thesis investigates and presents various novel fixed and dynamic BP schedules which not only expedite the convergence of BP decoder, but also improve the system error-rate performance.

List of Abbreviations

SLC	Single-Level per Cell.
MLC	Multi-Level per Cell.
SSD	Solid-State Drive.
HDD	Hard Disk Drive.
EEPROM	Electrically Erasable Programmable Read-Only Memory.
FN	Fowler-Nordheim.
ISPP	Incremental Step Pulse Programming.
FGT	Floating Gate Transistor.
RTN	Random Telegraph Noise.
CCI	Cell-to-Cell Interference.
CCR	Coupling-Capacitance-Ratio.
PE	Program / Erase cycles.
LDPC	Low-Density Parity-Check.
BCH	Bose-Chaudhri-Hocquenghem.
RS	Reed-Solomon.
BP	Belief-Propagation.
LLR	Log-Likelihood-Ratio.
PEG	Progressive Edge Growth.
ECC	Error Correcting Code.
BER	Bit-Error-Rate.
FER	Frame-Error-Rate.

MMI	Maximum Mutual-Information.
AWGN	Additive White Gaussian Noise.
MSB	Most Significant Bit.
LSB	Least Significant Bit.
EOL	End-Of-Life.
RABP	Retention-Aware Belief-Propagation.
RA-CU	RABP assisted Channel Update.
RA-CUS	RABP assisted Channel Update with decoding Success.
RA-CUF	RABP assisted Channel Update with decoding Failure.
GD	Gradient Descent.
MSE	Mean Squared Error.
LUT	Look-Up Table.
AP	A-Priori.
SIR	Symmetric Information Rate.
C2V	Check-to-Variable.
V2C	Variable-to-Check.
CWS	Column-Weight Scheduling.
IFS	Informed Fixed Scheduling.
e-Shuffled	Edge-based Shuffled.
D-SVNFS	Dynamic Silent-Variable-Node-Free Scheduling.
H-L	High-to-Low column-weight.
L-H	Low-to-High column-weight.

List of Symbols

R_i	i^{th} read threshold voltage level.
V_p	Write (verify) voltage level of cell-state s_p .
$p_{s_p}(v)$	Voltage distribution of cell-state s_p .
μ_{s_p}	Mean value of voltage distribution of cell-state s_p .
σ_{s_p}	Variance of voltage distribution of cell-state s_p .
$p_{e s_p}(v)$	Error probability of cell-state s_p .
P_e	Average channel error probability.
ΔV_{pp}	ISPP voltage step size.
σ_e	Standard deviation of erased state noise.
σ_p	Standard deviation of programmed state noise.
ΔV	Change in threshold voltage level after cell programming.
$\Delta \mu$	Change in mean value of voltage distribution.
$\Delta \sigma$	Change in standard deviation value of voltage distribution.
γ	Coupling-capacitance-ratio between adjacent cells.
∂	Partial derivative.
$\mathcal{MI}$	Mutual information.
$\mathcal{D}_i$	i^{th} Data voltage region.
$\mathcal{E}_i$	i^{th} Erasure voltage region.
N_t	Total number of cells per memory-page.
$N_{\mathcal{D}_{i-1}}^{\mathcal{D}_i}$	Modeled retention-error memory cells.
$N_{\mathcal{E}_{i-1}}^{\mathcal{D}_i}$	Modeled retention-erasure memory cells.

$\tilde{N}_{\mathcal{D}_{i-1}}^{\mathcal{D}_i}$	Measured retention-error memory cells.
$\tilde{N}_{\mathcal{E}_{i-1}}^{\mathcal{D}_i}$	Measured retention-erasure memory cells.
$\hat{N}_{\mathcal{D}_{i-1}}^{\mathcal{D}_i}$	Approximated retention-error memory cells.
$\hat{N}_{\mathcal{E}_{i-1}}^{\mathcal{D}_i}$	Approximated retention-erasure memory cells.
$Z_{\mathcal{D}_i}$	Observed memory cells in voltage data region $\mathcal{D}_i$.
$Z_{\mathcal{E}_i}$	Observed memory cells in voltage erasure region $\mathcal{E}_i$.
∇C	Gradient of function C .
$L_{\text{lsb} \text{msb}}$	Input LSB MSB log-likelihood-ratio information.
Λ	Output log-likelihood-ratio information.
$\hat{a}_{\text{lsb} \text{msb}}$	LSB MSB decoded code-word bit.
$\odot$	Index of the set elements.
$I_{\max}$	Maximum decoder iterations.
$\lambda(x)$	Variable node degree distribution.
$\rho(x)$	Check node degree distribution.
$\bar{d}_v$	Average variable node degree.
$\bar{d}_c$	Average check node degree.
$\mathcal{M}(v_n)$	Set of neighbors of variable node v_n in code graph.
$\mathcal{N}(c_m)$	Set of neighbors of check node c_m in code graph.
$\mathcal{N}_{t-\min}(c_m)$	Set of t minimum-reliability neighbors of check node c_m .
$m_{v_n \rightarrow c_m}$	Message update from variable node v_n to check node c_m .
$m_{c_m \rightarrow v_n}$	Message update from check node c_m to variable node v_n .
$r(c_m \rightarrow v_n)$	Message residual from check node c_m to variable node v_n .
s_{c_m}	Syndrome (parity-check) of check node c_m .
ζ_m	Check node counter of c_m .
η_n	Variable node counter of v_n .
$\mathcal{R}(c_m)$	Reliability measure of check node c_m .
$\mathcal{T}$	Pre-defined reliability threshold for e-Shuffled schedule.

List of Figures

2.1	Organization of NAND flash memory cells.	16
2.2	Illustration of program-and-verify voltage pulses while employing the Incremental Step Pulse Programming (ISPP) method.	18
2.3	Illustration of noisy NAND flash channel incorporating programming noise (n_u and n_p), CCI (I), RTN (n_w) and retention noise (n_r).	20
2.4	Illustration of voltage distribution functions in 2-bit per cell NAND flash memory.	24
3.1	Illustration of four write/verify voltage levels (V_{s11} , V_{s10} , V_{s00} and V_{s01}) and three read threshold voltage levels (R_1 , R_2 and R_3) in 2-bit per cell NAND flash memory.	29
3.2	Plot of channel raw error probability (P_e) as a function of write voltage levels V_{s10} and V_{s00} , setting PE=5K and $T = 0$	35
3.3	Comparison of channel raw error probability (P_e) between the proposed write voltage optimization scheme and the conventional channel invariant write voltage scheme: <i>solid curve</i> (V_{s10}^* , V_{s00}^* from Table 3.1), <i>dotted curve</i> ($V_{s10} = 2.6$, $V_{s00} = 3.2$).	36
3.4	Illustration of <i>uniform interval</i> D_u based quantization scheme in a 2-bit per cell flash memory with 6 (R_1 to R_6) equi-spaced read threshold voltage levels.	37
3.5	Illustration of <i>entropy quantization</i> scheme for 2-bit per cell flash memory: quantization levels (R_1 to R_6) are placed closer to the erasure (overlapping) regions.	38
3.6	Comparison of <i>mutual information</i> computed by employing 6-level Entropy (proposed) and 6-level MMI quantization schemes.	41
3.7	Frame-error-rate (FER) performance of LDPC 4K-code by employing the Entropy (proposed), MMI and uniform quantization schemes: <i>solid curves</i> are plotted for optimized write voltage levels and <i>dotted curves</i> are plotted for channel invariant write voltage levels.	41
3.8	Frame-error-rate (FER) performance of LDPC 8K-code by employing the Entropy (proposed), MMI and uniform quantization schemes: <i>solid curves</i> are plotted for optimized write voltage levels and <i>dotted curves</i> are plotted for channel invariant write voltage levels.	42
3.9	Plot of <i>cell storage capacity</i> by employing 3-, 6-, 9-, 12- and ideal <i>soft</i> decoding schemes.	42
3.10	Frame-error-rate (FER) for entropy quantization scheme by employing 3-, 6-, 9-, 12- and ideal <i>soft</i> decoding schemes.	43

3.11	Comparison of 6-level memory sensing between the Entropy (proposed) and MMI quantization schemes: Entropy quantization levels contribute to wider erasure regions.	44
3.12	Bit-error-rate/Frame-error-rate (BER/FER) performance of LDPC 4K-code and 8K-code plotted against Entropy parameter θ : values of θ corresponding to optimal error-rate performance are marked with vertical arrows and the MMI quantization levels corresponding to θ is marked with slanted arrow.	45
3.13	Comparison of symbol error probability (P_{error}) and symbol erasure probability (P_{erasure}) between the Entropy (proposed) and MMI quantization schemes: Entropy quantization produces lower P_{error} at the expense of higher P_{erasure}	47
3.14	Bit-error-rate (BER) performance of LDPC 4K-code plotted versus decoding iteration (I_{max}) for different PE cycles: <i>solid</i> and <i>dotted</i> curves represent channel optimized and channel invariant (fixed) write voltage schemes, respectively.	48
3.15	Comparison of <i>flash memory endurance</i> between optimizing write voltage levels for early retention $T_{\text{write}} = 0$ (<i>solid curves</i>) and end of retention (EOR) $T_{\text{write}} = 12$ -months (<i>dotted curves</i>), keeping the FER fixed at 10^{-3} and 10^{-5} : employing LDPC 4K-code with 6-level Entropy quantization.	49
4.1	Illustration of voltage distribution functions in 2-bit per cell NAND flash memory before (<i>top figure</i>) and after (<i>bottom figure</i>) data retention noise effect: voltage levels tend to drop gradually with data retention time.	54
4.2	The amount of change in mean $\Delta\mu$ (<i>top figure</i>) and standard deviation $\Delta\sigma$ (<i>bottom figure</i>) values with increasing data retention time at PE = 5K (<i>solid curves</i>) and PE = 10K (<i>dotted curves</i>).	54
4.3	Illustration of 6-level quantization for 2-bit per cell NAND flash memory: voltage range is split into 4 data regions ($\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3, \mathcal{D}_4$, correspond to state s_{11}, s_{10}, s_{00} and s_{01} , respectively) and 3 erasure regions ($\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3$).	60
4.4	Plot of retention-erasure probability $p(N_{\mathcal{E}_{i-1}}^{\mathcal{D}_i}, T)$ and retention-error probability $p(N_{\mathcal{D}_{i-1}}^{\mathcal{D}_i}, T)$ versus data retention time T at PE = 5K (<i>solid curves</i>) and PE = 10K (<i>dotted curves</i>).	60
4.5	Plot of average errors observed <i>before</i> and <i>after</i> the first round of BP decoding versus data retention time at PE = 5K.	62
4.6	Fraction of retention-error <i>corrected/uncorrected</i> cells versus data retention time at PE = 5K.	62
4.7	Frame-error-rate (FER) versus data retention time (without channel update) by employing the conventional BP (CBP) (<i>dotted curves</i>) and the proposed RABP decoding (<i>solid curves</i>) at PE = 3K, 4K and 5K.	63

4.8	Illustration of cost function (4.11) plotted against data retention time at PE = 5K (<i>top figure</i>) and PE = 10K (<i>bottom figure</i>): minimum points correspond to actual retention age set in the simulations.	67
4.9	Block diagram of the RABP assisted channel update algorithm invoked after decoding <i>success</i> (RA-CUS) and decoding <i>failure</i> (RA-CUF).	70
4.10	Average error of the estimated mean (ε_μ) plotted against data retention time at PE = 5K (<i>top figure</i>) and PE = 10K (<i>bottom figure</i>) by employing the RA-CUS (<i>solid curves</i>) and RA-CUF (<i>dotted curves</i>) algorithms.	71
4.11	Average error of the estimated standard deviation (ε_σ) plotted against data retention time at PE = 5K (<i>top figure</i>) and PE = 10K (<i>bottom figure</i>): employing RA-CUS (<i>solid curves</i>) and RA-CUF (<i>dotted curves</i>) algorithms.	72
4.12	Average number of GD iterations required to achieve convergence versus data retention time, plotted at PE = 5K and 10K.	73
4.13	Frame-error-rate (FER) performance versus data retention time by employing the proposed RA-CUS algorithm for distribution parameter update: comparison between non-updated parameters and updated parameters at $T_{update} = 70, 110$ and 600 hours for PE = 10K, 8K and 6K, respectively.	74
4.14	Frame-error-rate (FER) performance versus data retention time plotted at PE = 10K by employing the proposed RA-CUF algorithm for distribution parameter update at different values of T_{update} , compared with non-updated distribution parameters.	75
5.1	Illustration of cell-to-cell interference (CCI) in the <i>even-odd</i> bit-line architecture: <i>even</i> and <i>odd</i> bit-line cells are interfered by 5 and 3 neighboring cells, respectively.	82
5.2	Illustration of voltage distribution functions for 2-bit per cell NAND flash memory: before CCI (<i>top figure</i>) and after CCI (<i>middle and bottom figures</i>) effect.	83
5.3	Block diagram of the signal pre-processing and post-processing techniques for CCI mitigation in NAND flash memory.	84
5.4	Illustration of the proposed <i>Even-AP</i> , <i>All-AP</i> and <i>All-AP-CCR</i> detection schemes: (a) cleaned odd bit-line cells are used for the compensation of even bit-line cells (b) cleaned next word-line ($i + 1$) cells are used for the compensation of both odd and even bit-line cells (c) All-AP detection combined with coupling-capacitance-ratio (CCR) estimation algorithm.	88
5.5	Surface plot of cost function $\mathcal{F}(\mu_{\gamma_y}, \mu_{\gamma_{xy}})$: mean squared error (MSE) between the measured and modeled CCI voltage-shift expressions.	92
5.6	Plot of normalized estimation error (ε) for <i>Post-Comp</i> , <i>Even-AP</i> , <i>All-AP</i> and <i>All-AP-CCR</i> detection schemes over varying CCI factor (s), keeping the PE cycles fixed at 5K and retention time fixed at $T = 1$ hour.	94

5.7	Symmetric-information-rate ($\mathcal{S}$) (average mutual-information) for the proposed <i>Even-AP</i> , <i>All-AP</i> and <i>All-AP-CCR</i> detection schemes, compared with the <i>Dong's</i> , <i>Post-Comp</i> , <i>Pre-Compensation</i> and <i>Opt-AP</i> detection schemes over varying CCI factor (s), keeping the PE cycles fixed at 5K and retention time at $T = 1$ hour.	96
5.8	Illustration of uncoded bit-error-rate (BER) curves simulated under the proposed <i>Even-AP</i> , <i>All-AP</i> and <i>All-AP-CCR</i> detection schemes, compared with prior-art <i>Post-Comp</i> detection scheme over varying CCI factor (s), keeping the PE cycles fixed at 5K and retention time at $T = 1$ hour.	97
5.9	Illustration of LDPC 4K-code bit-error-rate (BER) performance by employing the prior-art Post-Comp detector, and the proposed Even-AP detector combined with the write voltage optimization, plotted against increasing PE cycles, keeping the CCI factor fixed at $s = 1.50$ and $s = 1.75$, and retention time at $T = 1$ hour.	100
5.10	Illustration of log-likelihood-ratio (LLR) re-mapping for second round of RABP decoding while employing the uniform based quantization levels.	101
5.11	Illustration of LDPC 4K-code bit-error-rate (BER) performance by employing the prior-art Post-Comp detector, and the proposed Even-AP detector combined with the write voltage optimization and RABP decoding, plotted against increasing retention time T , keeping the CCI factor fixed at $s = 1.00$ and PE cycles at 5K.	101
6.1	A simple tanner graph example showing the <i>aggregated message count</i> observed while employing the proposed high-to-low column-weight scheduling $\mathcal{A}_{H-L}$ (<i>top figure</i>), and compared with low-to-high column-weight scheduling $\mathcal{A}_{L-H}$ (<i>bottom figure</i>).	110
6.2	Number of aggregated messages for high-to-low $\mathcal{A}_{H-L}$, random $\mathcal{A}_R$ and low-to-high $\mathcal{A}_{L-H}$ column-weight based scheduling: employing capacity approaching LDPC codes with $d_{v,max} = 5$, $d_{v,max} = 10$, $d_{v,max} = 20$, $d_{v,max} = 30$ and $d_{v,max} = 50$, respectively.	111
6.3	Illustration of mutual information curves for the high-to-low ($\mathcal{MI}_{H-L}$) and low-to-high ($\mathcal{MI}_{L-H}$) column-weight based scheduling schemes for $d_{v,max} = 50$ LDPC code.	112
6.4	An example of how shuffled BP decoding utilizes the newly available message updates: variable node v_2 receives new message $v_{v_1 \rightarrow c_1}$ via check node c_1	113
6.5	A working example of variable node scheduling $\{v_1, \dots, v_7\}$ by using the proposed <i>informed fixed scheduling</i> (IFS) scheme: scheduled sequence is given by $v_5 \Rightarrow v_7 \Rightarrow v_6 \Rightarrow v_3 \Rightarrow v_1 \Rightarrow v_2 \Rightarrow v_4$	116
6.6	Illustration of message-passing along the unreliable edges of the code graph under the proposed <i>edge-based shuffled</i> (e-Shuffled) scheduling scheme.	120

6.7	Check-node-type probability p (Type-2) and p (Type-3), plotted against the variable node log-likelihood-ratio (Λ) for check node degree $\bar{d}_c = 10$, $\bar{d}_c = 20$ and $\bar{d}_c = 30$	122
6.8	Illustration of reliability threshold ($\mathcal{T}$) versus the average check node degree ($\bar{d}_c$): plotted using (6.15).	123
6.9	Code graph cycle in BP decoding: the conventional BP schedule receives correlated message update at variable node v_1 after 3 iterations via cycle $v_1 \leftrightarrow c_1 \leftrightarrow v_2 \leftrightarrow c_2 \leftrightarrow v_4 \leftrightarrow c_4 \leftrightarrow v_1$; the proposed edge-based BP schedule avoids the correlated message by skipping update operation along edge $\vec{e}_{2,2}$	124
6.10	BP decoding by using the proposed <i>dynamic silent-variable-node-free scheduling</i> (D-SVNFS): dynamic order of variable node participation (first round follows $v_1 \Rightarrow v_5 \Rightarrow v_6 \Rightarrow v_2$ and so on).	129
6.11	Average variable-to-check (V2C) decoding complexity ($\mathcal{V}$) of the proposed CWS, IFS, e-Shuffled and D-SVNFS schedules, compared (normalized) w.r.t. conventional shuffled schedule.	133
6.12	Average check-to-variable (C2V) decoding complexity ($\mathcal{C}$) of the proposed CWS, IFS, e-Shuffled and D-SVNFS schedules, compared (normalized) w.r.t. conventional shuffled schedule.	133
6.13	Frame-error-rate (FER) performance versus memory PE cycles of LDPC 4K-code by employing the proposed CWS and IFS schedules, and compared with conventional shuffled schedule: simulated at $I_{\max} = 5$ (<i>solid curves</i>) and $I_{\max} = 20$ (<i>dashed curves</i>).	135
6.14	Frame-error-rate (FER) performance versus memory PE cycles of LDPC 2K-code by employing the proposed e-Shuffled and D-SVNFS schedules, and compared with conventional shuffled schedule: simulated at $I_{\max} = 5$ (<i>solid curves</i>) and $I_{\max} = 20$ (<i>dashed curves</i>).	135
6.15	Frame-error-rate (FER) versus iteration ($I_{\max}$) of LDPC 4K-code by employing the proposed CWS and IFS schedules, and conventional shuffled schedule.	136
6.16	Frame-error-rate (FER) versus iteration ($I_{\max}$) of LDPC 2K-code by employing the proposed e-Shuffled and D-SVNFS schedules, and conventional shuffled schedule.	136
6.17	Frame-error-rate (FER) versus iteration ($I_{\max}$) of LDPC 4K-code by employing the proposed CWS and IFS, e-Shuffled and D-SVNFS schedules, and conventional shuffled schedule.	137
6.18	Average check-to-variable (C2V) / variable-to-check (V2C) complexity of LDPC 4K-code of the proposed IFS schedule employing group shuffled decoding with size p , normalized by conventional shuffled schedule.	138
6.19	Frame-error-rate (FER) performance versus memory PE cycles of LDPC 2K-code by employing the conventional shuffled schedule, and compared with the proposed e-Shuffled schedule when used in conjunction with write voltage optimization method: simulated at $I_{\max} = 20$	138

List of Tables

1.1	Summary of key features of different non-volatile memory technologies.	3
2.1	Important features of different NAND flash channel noise components.	25
3.1	Optimized write voltage levels over varying PE cycles.	35
3.2	Optimized read threshold voltage levels $\{R_1, R_2, R_3, R_4, R_5, R_6\}$ for 4K-code over varying PE cycles.	40
3.3	Width of erasure regions $\mathcal{E}_1$, $\mathcal{E}_2$ and $\mathcal{E}_3$ and the corresponding read threshold voltage levels over varying entropy parameter θ , keeping the PE cycles fixed at 21K.	45
4.1	The number of arithmetic operations performed by the Gradient Descent (GD) based parameter estimation algorithm (per iteration).	73
5.1	Complexity requirement of different CCI cancellation schemes (per N_t -cell memory-page).	98
6.1	Edge-state ($\varepsilon_{m,n}$) update criterion.	120
6.2	Complexity analysis of fixed and dynamic scheduling schemes (per iteration)	132

Chapter 1

Introduction

The usage of NAND flash memory has grown tremendously over the past several years, due to its decreasing per-bit cost and increasing data storage capacity. The application areas of NAND flash memory range from consumer electronics to more specialized enterprise data applications. In this light, one of the most significant developments in recent years in data storage industry is the advent of NAND flash-based solid-state drives (SSDs). As there are no moving parts in the SSD, it provides better performance and durability over a hard disk drive (HDD). Furthermore, due to no mechanical seek latency, SSD can offer access to the stored data with significantly higher read throughput, nearly $100\times$ faster than the HDD [1]. Besides the reliability and high data transfer rates, SSD has a better power-efficiency and occupies smaller physical size than the traditional HDD. Considering these important features of NAND flash based SSDs, the share of NAND flash memory in the whole storage market is likely to increase in time to come.

The large storage capacity and low per-bit cost features of NAND flash memory are attributed to the state-of-the-art manufacturing processes and key technological innovations. The NAND flash-based memory devices are fabricated with compact architecture in which a number of memory cells are serially connected with a common contact point to the bit-line terminal. This results in high storage efficiency

and great saving in silicon area, realizing storage density of $4F^2$ [2], where F is process feature size. Apart from compact fabrication, the continuous growth in storage density is driven by aggressive node scaling where storage cell sizes are shrunk to sub-10-nm generation technology within the past few years [3]. Besides node scaling, NAND flash storage efficiency also increases with multi-level-cell (MLC) technology in which m bits are stored over a single memory cell by configuring the storage cell to hold more than two voltage levels (called cell threshold voltages). Currently, 2-bit or 3-bit per cell NAND flash devices occupy a major segment of the commercial market [4–6], which linearly increases the storage capacity by almost $2\times$ or $3\times$, while 4-bit NAND flash configurations have also appeared in the open literature [7, 8].

Besides the NAND flash memory, there are some new emerging non-volatile memory technologies which are being investigated in the industry and may lead to as the potential alternatives to the existing memory technologies. Emerging non-volatile memory technologies include the magnetic random-access memory (MRAM), spin-transfer torque random-access memory (STT-RAM), phase-change memory (PCM), and resistive random-access memory (RRAM). These new memory technologies have read/write/power/endurance/capacity/cost characteristics which are different from those of NAND flash memory, as reported in Table 1.1 [9]. Despite the distinctive technical features offered by these emerging memory technologies, the price and scalability are the main obstacles to competing with dominant NAND flash memory.

1.1 Data Storage Reliability Issues in NAND Flash Memory

Despite the obvious advantages of NAND flash memory, there are many data reliability issues that needs to be addressed. With increasing NAND flash density, while

TABLE 1.1: Summary of key features of different non-volatile memory technologies.

Features	MRAM	STT-RAM	PCM	RRAM	NAND Flash
Endurance	High (10^{15})	High (10^{16})	Medium (10^8)	Low (10^6)	Low (10^5)
Read speed (ns)	High (3-20)	High (2-20)	Medium (10-100)	High (20-30)	Low (20,000)
Write/erase speed (ns)	High (3-20)	High (2-20)	Medium (75)	High (10)	Low (10,000)
Power	Medium	Medium	Medium	Low	High
Cell size (F^2)	Large (25)	Medium (6-20)	Small (6-12)	Small (6-12)	Very Small (4)
Cost	High	High	Medium	High	Very Low

employing the MLC configuration and pursuing aggressive technology node scaling, flash memory cells have become vulnerable to several undesirable circuit-level noise and interference effects. This subsequently leads to poor endurance performance and significant degradation of flash data reliability [10–12].

NAND flash memory cells are made of floating gate transistors (FGTs). The program/erase (PE) operation of a memory cell induces electron tunneling through the tunnel oxide of FGT [13]. The tunneling mechanism gradually generates electron trap sites (defects) into the tunnel oxide, causing irreparable damage to the tunnel oxide, which in turn affect the stability of cell's threshold voltage. After receiving a certain number of PE cycles, enough traps are accumulated in the tunnel oxide which induce trap-assisted tunneling. This could cause severe reliability problems, such as leakage of electrons and threshold voltage fluctuations. Thus, NAND flash memory devices can only sustain a limited number of PE cycles and suffer from low PE endurance. Due to further node scaling, future NAND flash memories will likely to have even lower PE endurance, and thus shorter operational lifetime.

As the node is scaled down, flash memory cells have become increasingly smaller. As a result, only a small number of electrons can be stored over a cell's FGT [14]. Thus, only a slight variation in the desired amount of electron tunneling during cell programming or even a small amount of charge loss during data retention

can cause dramatic change (shift) in the expected threshold voltage levels [11]. When this happens, the data retention capability of a flash memory cell tends to be significantly reduced. This problem exacerbates with the MLC technology, where intervals between adjacent voltage levels are increasingly shortened. This is because when multiple bits are programmed per cell, more threshold voltage regions (windows) need to be created for each cell, and the threshold voltage window representing each cell value becomes narrower. This translates to even fewer number of electrons to represent each threshold voltage window. Hence, making a correct decision on cell's logical value has become increasingly difficult, which leads to high bit detection errors.

Finally, as the gap between the adjacent FGTs is reduced, this has induced more parasitic coupling-capacitance between the adjacent memory cells. As a result, the programming of a memory cell affects the threshold voltages of neighboring memory cells through the coupling-capacitance induced cell-to-cell interference (CCI) [15]. The CCI has been well recognized as one of the leading source of errors in today's NAND flash memories [16, 17]. The effect of CCI on threshold voltage level is independent of memory PE cycles, thus a fresh NAND flash with small PE cycles is also susceptible to CCI errors. Furthermore, due to the compact fabrication of NAND flash, the memory cells are also subject to small process variations in feature geometry or oxide thickness [18]. The random process variations produce a measurable effect on the channel noise and CCI magnitude, and thus result in an increase in raw memory bit errors. The combined retention, CCI and process variation errors can get well beyond the error correcting capability of the underlying ECC decoder.

A detailed description of the NAND flash channel model, incorporating all major voltage signal impairments and the corresponding probability distribution functions, will be presented in Section 2.3.

1.2 Motivation and Objectives

In view of the above-mentioned reliability issues related to NAND flash memory, effective fault-tolerant solutions are clearly required to ensure the overall storage reliability. Significant measures need to be taken at the system-level to reduce fabrication/manufacturing process variability. In addition, signal-level measures should also be taken to mitigate the cell voltage noise and interference effects left behind by the fundamental limits of device physics and circuit design. This is to make sure that the key performance indicators of NAND flash memory, such as the endurance and retention limit, are maintained.

One of the prime objective of this research study is to extend the PE endurance of NAND flash memory by coding and signal processing means. As the recurrent PE cycles inflict irreparable damage to the tunnel oxide, a NAND flash device can only endure a limited number of PE cycles. The NAND flash aging process is typically managed by flash memory controller through the sophisticated wear-leveling algorithms [19–21], which attempt to uniformly distribute the program/erase operations across the entire flash memory. In wear-leveling, it is ensured that all memory blocks will deteriorate at about the same rate and hence reach their end-of-service lifetime roughly simultaneously. Despite the wear management, the gradual deterioration of NAND flash cells over increasing PE cycles escalates the channel noise magnitude. This subsequently makes the default programmed (write) voltage levels to become outdated and sub-optimal. To cater for the non-stationary flash channel noise, we aim to design the write voltage levels to adapt to these channel variations.

In the current design practice, flash memory controllers invariably employ error correction code (ECC) to tolerate the raw memory errors [22–24]. The ECC scheme protects the stored data against the channel induced errors by leveraging on the additional parity bits. Unfortunately, the conventional hard-decision decoding codes, like the Bose-Chaudhri-Hocquenghem (BCH) [25, 26] and Reed-Solomon (RS) codes [27], are no longer adequate for the application to high density NAND flash

technology. The larger PE endurance and longer retention requirements of NAND flash storage call for more powerful ECC schemes. As a consequence, soft-decision decoding low-density parity-check (LDPC) code have gained widespread acceptance in new-generation NAND flash memories [28–33]. However, the belief-propagation (BP) decoding of an LDPC code demands high precision memory sensing quantization for accurate computation of log-likelihood-ratio (LLR) information, and high-precision memory sensing consumes power and incurs sensing latency. In this thesis, we aim to design the quantized read threshold levels to approach the ideal soft LDPC decoding performance limits without excessive implementation cost, that outperform the existing quantization design schemes.

Despite the apparent advantages of LDPC code over the conventional BCH and RS codes in terms of superior error performance, the application of LDPC code also introduces new challenges which need to be addressed. Keeping aside the high-precision memory sensing latency, the iterative BP decoding process may take tens to hundreds of decoding iterations to achieve decoding convergence (stable minimized error rate). Such decoding latency further increases the total read response time in NAND flash memory. Prior-art works on LDPC implementation mostly focus on reducing the latency overhead caused by the fine-grained memory sensing precision [30, 31], whereas the underlying BP decoding latency is almost overlooked. To fully reap the benefits of LDPC code, an effective design solution is required which can reduce the latency associated with BP decoding. In this context, it is well known that a message-passing scheduling strategy can play an important role as it can affect the decoding convergence, complexity and error-rate performance of BP decoder. Realizing the importance of BP scheduling, we aim to design improved scheduling methods and investigate their performance and complexity on iterative decoding of high rate LDPC codes suitable for use in NAND flash memory.

For soft LDPC decoding, the quantized read threshold levels are designed

based upon the knowledge of flash channel voltage distribution parameters (mean and variance). These quantization levels are then used with the read voltage signal to compute the LLR information. However, in practice, the up-to-date voltage distribution parameters are often unknown to the flash memory controller. This is because the voltage distribution tends to drift continuously due to the unremitting charge leakage from cell's FGT. Thus, the initially known voltage distribution parameters tend to become outdated and sub-optimal with increasing data retention time. This may lead to LDPC decoding failure due to inaccurate LLR information, and thus demand flash channel estimation afresh. All prior-art works on retention-failure recovery [34–37] and flash channel estimation [38–41] require additional memory sensing operations. Note that the total read latency of NAND flash memory is composed of firmware processing, on-chip memory sensing, and memory-to-controller data transfer, while the most time consuming process is the memory sensing operation. Since the read response time is very critical in NAND flash memory based devices, it is of utmost importance to reduce the read latency overhead associated with the additional memory sensing operations required in the prior-art works [34–41]. Thus, we aim to develop new retention-failure recovery schemes and flash channel updating algorithms that can preclude the time and power consuming additional read operations.

Unlike the retention noise errors which grow with PE cycles and retention time, the CCI induced errors emerge irrespective of the flash wear-out dynamics and data retention age. Instead, they grow with increasing NAND flash density. This is because when memory cells are integrated closer to each other, the coupling-capacitance induced CCI distorts the threshold voltages of adjacent memory cells. Thus, it is essential to develop signal processing solutions that can minimize or mitigate the CCI effect. The CCI in flash channel is quite similar to the inter-symbol interference encountered in digital communication systems. Hence, techniques identical to channel detector/equalizer can be employed to address the

CCI in NAND flash memory. The existing CCI cancellation schemes, such as [42–45], typically attempt to subtract the estimates of CCI from the output voltage signal. As the estimated CCI values are calculated based upon the output voltage of interfering cells, their estimation accuracy can be further improved if the interfering cells are equalized (cleaned) before the actual victim cell. In this direction, we aim to develop post-processing channel detection schemes that can more accurately compute and subtract the CCI estimates from the output voltage signal.

1.3 Major Contributions of the Thesis

The main contributions of this thesis are summarized as follows.

1.3.1 Read and Write Voltage Signal Design

In this research work, we present an analytical approach to optimize the *write voltage levels* for NAND flash memory. In view of the time-varying flash memory channel whose cell threshold voltage distributions change over the number of PE cycles and data retention time, we propose to adapt the write voltage levels on-the-fly such that the channel raw error probability is minimized. The impact of write voltage optimization between the flash memory’s early and end-of-retention times is discussed. Optimization of write voltage levels to cater for the non-stationary flash channel extends the PE endurance capability and thus the operational lifetime of NAND flash memory. Furthermore, we present a voltage entropy based *non-uniform quantization scheme* to facilitate soft belief-propagation (BP) decoding of LDPC codes. Under the proposed quantization scheme, the dominating error regions (erasure) are enclosed within the designed quantization levels. The proposed read and write voltage signal optimization is shown to minimize the LDPC error-rate performance and outperforms the prior-art flash quantization schemes.

The main results related to this work have been published in the following conference and journal papers:

- **C. A. Aslam**, Y. L. Guan, and K. Cai, “Read and write voltage signal optimization for multi-level-cell (MLC) NAND flash memory,” *IEEE Trans. Commun.*, vol. 64, no. 4, pp. 1613-1623, April 2016.
- **C. A. Aslam**, Y. L. Guan, and K. Cai, “Dynamic write-level and read-level signal design for MLC NAND flash memory,” in *Proc. 9th Int. Symp. Commun. Syst., Netw., Digit. Signal Process. (CSNDSP)*, Manchester, UK, July 2014, pp. 336-341.

1.3.2 Retention-Failure Recovery with Channel Update

In this research study, we introduce a *retention-aware belief-propagation* (RABP) decoding scheme for LDPC codes to recover the retention noise induced errors in NAND flash memory. The RABP scheme is comprised of two rounds of BP decoding wherein the flash cell’s charge-loss effect is systematically compensated by performing retention-erasure correction or retention-error correction. Upon BP decoding failure, unlike the existing schemes which conduct read retries for data recovery, the probable retention-affected memory cells are first identified, leveraging only on the read-back threshold voltage and the decoded bit decisions. Then, for all such likely retention-affected memory cells, the corresponding log-likelihood-ratio (LLR) voltage regions are modified in such a way as to absorb the effect of charge leakage, and a second round of BP decoding is performed afresh.

Furthermore, we propose an *RABP assisted channel update* (RA-CU) algorithm which estimates the voltage distribution mean and standard deviation parameters without incurring new memory sensing operations. This is achieved by expressing the *measured* (actual) and *predicted* bit error/erasure counts in terms of a cost function, which defines the mean squared error (MSE) between the two quantities. This cost function is subsequently minimized to obtain the desired voltage distribution parameters. Through simulations, it is shown that the RABP decoder increases the retention time limit of NAND flash channel by up to 70% compared to single round

of BP decoding. The proposed RA-CU algorithm further improves the retention time limit compared to a system which employs non-updated voltage distribution parameters.

The main results related to this work have been published/submitted in the following conference and journal papers:

- **C. A. Aslam**, Y. L. Guan, and K. Cai, “Retention-aware belief-propagation decoding for NAND flash memory,” in *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 64, no. 6, pp. 725-729, June 2017.
- **C. A. Aslam**, Y. L. Guan, and K. Cai, “Decision-directed retention-failure recovery with channel update for MLC NAND flash memory,” **under review (minor revision)** in *IEEE Trans. Circuits Syst. I, Reg. Papers*.
- **C. A. Aslam**, K. Cai, and Y. L. Guan, “On the Recovery of Retention-Failure Data in MLC NAND Flash Memory,” in *Non-volatile Memory Workshop (NVMW)*, San Diego, USA, March 2017.

1.3.3 Cell-to-Cell Interference (CCI) Cancellation

We present three *post-processing detection schemes* for the cancellation of CCI in NAND flash memory. In this research study, we assume the *even-odd* bit-line architecture, which offers sharing of peripheral circuitry at the cost of increased CCI, to illustrate the full potential of the proposed detection schemes. In the even-odd bit-line architecture, the even bit-line memory cells are programmed first followed by the odd bit-line memory cells. Nevertheless, the proposed schemes can easily be modified for the *all* bit-line architecture, where all the memory cells are programmed at the same time. Exploiting the data-dependent and stationary nature of CCI, which can be represented in terms of the change in threshold voltage (voltage-shift) of neighboring (interfering) cells and the coupling capacitance ratio (CCR) between the victim and interfering cells, we estimate the CCI strength and subtract it from the read-back voltage signal of the victim cell. In particular, for more accurate estimation of the CCI strength, we precisely compute the individual factors that constitute the overall CCI component. Specifically, we first remove

the CCI from the interfering cells to get a cleaned (CCI-removed) version of the read-back voltage signal. This cleaned voltage signal will subsequently provide a more accurate estimate of the voltage-shift values of interfering cells. Next, we formulate a cost function which defines the MSE between the measured and expected CCI strengths. This cost function is then minimized by employing the gradient descent (GD) method to find the best-fit mean CCR values. Based on the estimated voltage-shift and mean CCR values, we compute a more reliable estimate of the overall CCI component.

The main results related to this work have been published in the following journal paper:

- **C. A. Aslam**, Y. L. Guan, and K. Cai, “Detector for MLC NAND flash memory using neighbor-a-priori information,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 9, pp. 2827-2836, Sep. 2016.

1.3.4 Improved Belief-Propagation (BP) Scheduling

In this research work, we propose some improved fixed and dynamic scheduling schemes for belief-propagation (BP) decoding of LDPC code. Fixed schedules can be determined off-line while dynamic schedules are determined on-line. Among them, fixed scheduling enjoys no run-time complexity (to determine the sequence of message passing) while dynamic scheduling yields better convergence and decoding performance. Our first scheme employs the high-to-low *column-weight scheduling* (CWS) policy to sequentially update the variable nodes in the LDPC code graph. By design, the CWS scheme works very well on irregular LDPC codes, but has inconsequential effect on regular LDPC codes which have constant column weight. To overcome this shortcoming, our second BP scheduling scheme, namely the *informed fixed scheduling* (IFS), aims to improve the performance of both regular and irregular LDPC codes. In IFS scheme, a fixed updating sequence of variable nodes is determined based on the information of latest message updates, ensuring

that the maximum number of such latest message updates is utilized within a single decoding iteration. In our third scheme, in contrast to CWS and IFS schemes where the entire code graph is updated per iteration, we propose an edge-based shuffled (*e-Shuffled*) scheduling which constrains the flow of message updates over a limited set of edges in the LDPC code graph. Since there are partial set of graph edges updated in each iteration, the total number of message updates, and consequently the overall decoding complexity, is significantly reduced. As a further consequence, the adverse effect of short code graph cycles is also diminished which leads to a better error-rate performance. Finally, we present a *dynamic silent-variable-node-free scheduling* (D-SVNFS) which regulates the propagation of message updates based on the check-to-variable message residuals. All the proposed scheduling schemes demonstrate significant improvement compared to the existing schemes in terms of decoding complexity, convergence speed and error-rate performance.

The main results related to this work have been published in the following conference and journal papers:

- **C. A. Aslam**, Y. L. Guan, and K. Cai, “Edge-based dynamic scheduling for belief-propagation (BP) decoding of LDPC and RS codes,” *IEEE Trans. Commun.*, vol. 65, no. 2, pp. 525-535, Feb. 2017.
- **C. A. Aslam**, Y. L. Guan, K. Cai, and G. Han, “Informed fixed scheduling for faster convergence of shuffled belief-propagation (BP) decoding,” *IEEE Commun. Lett.*, vol. 21, no. 1, pp. 32-35, Jan. 2017.
- **C. A. Aslam**, Y. L. Guan, K. Cai, and G. Han, “Low-complexity belief propagation (BP) decoding via dynamic silent-variable-node-free scheduling,” *IEEE Commun. Lett.*, vol. 21, no. 1, pp. 28-31, Jan. 2017.
- **C. A. Aslam**, Y. L. Guan, and K. Cai, “Improving the belief propagation convergence of irregular LDPC codes using column-weight based scheduling,” *IEEE Commun. Lett.*, vol. 19, no. 8, pp. 1283-1286, Aug. 2015.
- **C. A. Aslam**, K. Cai, and Y. L. Guan, “Edge-based scheduling for decoding non-binary LDPC codes for MLC flash memories,” in *Non-volatile Memory Workshop (NVMW)*, San Diego, USA, March 2016.
- **C. A. Aslam**, Y. L. Guan, and K. Cai, “Low-complexity quantization-aware belief-propagation (QA-BP) decoding for MLC NAND flash memory,” in *Proc. Int. Conf. Information, Communication and Signal Processing (ICICS)*, Singapore, Dec. 2015, pp. 1-5.

1.4 Organization of the Thesis

This thesis is organized as follows.

Chapter 1, the current chapter, introduces the reliability issues in NAND flash memory and presents the motivation, objectives, main research contributions, and organization of this thesis.

Chapter 2 presents an overview of NAND flash memory and describes its basic operations and channel model.

Chapter 3 focuses on the read and write voltage signal optimization for NAND flash memory under a non-stationary and asymmetric flash channel. The optimum write voltage levels are determined by minimizing the channel error probability. Besides, a novel entropy based quantization scheme is proposed to facilitate soft LDPC decoding. The error-rate and decoding convergence results under the proposed read/write voltage optimization scheme are analyzed.

Chapter 4 proposes a decision-directed retention-failure recovery scheme for NAND flash memory. Besides, a decoder-assisted flash channel estimation algorithm is proposed to update the threshold voltage distribution parameters. Most importantly, the failure recovery and channel estimation algorithms are performed without incurring new read operations. The average estimation error, estimation complexity and error-rate performance under the proposed algorithm are analyzed.

Chapter 5 presents three post-processing detection schemes for the cancellation of CCI in NAND flash memory. The proposed techniques aim to clean the interfering (neighboring) cells before the cancellation of CCI from the victim cell. The performance improvements and the design trade-offs under the proposed detection schemes are evaluated.

Chapter 6 proposes some improved fixed and dynamic scheduling schemes for BP decoding of LDPC codes and discusses their implications on the decoding convergence, complexity and error-rate performance.

Chapter 7 draws conclusions of the important results presented in this thesis and suggests possible future research directions.

Chapter 2

An Overview of NAND Flash Memory

In this chapter, we describe the organization of memory cells inside a NAND flash array and explain the basic operations performed by a flash memory controller. We also study the major flash channel noise components that affect the data storage reliability and present the statistical models for the overall NAND flash channel.

2.1 NAND Flash Memory Organization

NAND flash memory cells are organized into a two dimensional array of rows and columns, where the individual rows and columns are connected with word-line and bit-line terminals, respectively, as shown in Fig. 2.1. The memory array is divided into several memory blocks, where each block is further subdivided into multiple memory pages. One memory block usually has 32 to 64 word-lines which collectively form a cell-string, where each word-line contains a series of memory cells, ranging from 16K to 64K. A single word-line can either represent one or more memory pages, depending on the type of MLC technology, where the size of a memory page is generally in between 2K-byte to 8K-byte. For instance, the Micron's 32Gb 2-bit

FIGURE 2.1: Organization of NAND flash memory cells.

per cell MLC NAND flash memory contains 8192 blocks with 128 pages per block and 4314 bytes per memory page [46]. In NAND flash array, the bit-line cells share a common on-chip sense amplifier which detects the data content stored over a memory cell, and a page buffer which holds the data content during program and read operations. Today's NAND flash memory chips either employ an *even-odd* bit-line architecture [47], or an *all* bit-line architecture [48]. In an even-odd bit-line architecture, even and odd bit-lines are interlaced and the corresponding word-line cells are alternatively accessed. In an all bit-line architecture, all the word-line cells are simultaneously accessed. The even-odd bit-line architecture facilitates sharing of peripheral circuitry leading to less silicon cost, however, at the expense of more cell-to-cell interference (CCI). In contrast, the all bit-line architecture offers better CCI immunity at the expense of increased silicon cost.

2.2 Basic NAND Flash Memory Operations

A NAND flash device is a kind of non-volatile EEPROM (Electrically Erasable Programmable Read-Only Memory) that can support *program*, *erase* and *read* as

its basic operations. The read and program operations are performed at a page granularity while an erase operation is performed at a block granularity. A typical flash memory cell is made of a floating gate transistor (FGT) which is fabricated with two poly gates; floating gate and control gate. The control gate is connected with the word-line voltage terminal whereas the floating gate is electrically isolated and completely surrounded by an isolation layer. This enables the floating gate to trap electrons over longer period of time and to be used as a non-volatile memory.

2.2.1 Program/Erase Operation

A flash memory cell can either be represented with the *erased* state, where no electrons are stored over the floating gate, or with the *programmed* state, where electrons are tunneled through the oxide layer and stored over the floating gate. Initially, all memory cells are in the erased state. A program operation involves the insertion of a certain amount of charged particles (electrons) into the cell's floating gate whereas an erase operation involves the removal of the stored electrons. The program and erase operations are performed by using the Fowler-Nordheim (FN) tunneling [13] mechanism. The tunneling of electrons to and from the FGT configures the threshold voltage level (voltage needed to turn-on the transistor) of a memory cell, while the varying threshold voltage levels in-turn represent the data bits (cell value) being stored. Hence, data storage through FN tunneling for an m -bit per cell NAND flash memory involves the configuration of cell's threshold voltage level into one of 2^m non-overlapping voltage windows.

Most NAND flash memories employ an iterative Incremental Step Pulse Programming (ISPP) method [49] to ensure a strict control on the threshold voltage window. In iterative programming, a series of program-and-verify voltage pulses are applied across the control gate, with an incremental step voltage size of ΔV_{pp} , until the target write (verify) voltage V_p is reached, as shown in Fig. 2.2. During each iteration, the cell's threshold voltage V is increased by a step size of ΔV_{pp} and

FIGURE 2.2: Illustration of program-and-verify voltage pulses while employing the Incremental Step Pulse Programming (ISPP) method.

subsequently compared with V_p . If the configured voltage level V is below the target voltage V_p , the program-and-verify cycle is repeated, otherwise the configured cell is inhibited by applying a high voltage across the corresponding bit-line terminal so that the programmed cell will not receive further programming pulses.

2.2.2 Read Operation

A read operation involves sensing the threshold voltage of a memory cell in order to detect the corresponding stored cell value. To read a flash memory cell, the amount of stored electrical charge is measured through the sense amplifier by applying a set of discrete read threshold voltage levels across the selected word-line terminal, whereas all the unselected word-line cells are typically biased at some high voltage, known as the pass voltage, so that they can act like pass-transistors, irrespective of their threshold voltage levels. The selected word-line memory cells which are configured to distinct write voltage levels would require different amount of read threshold levels to turn-on the FGT. For instance, an erased flash cell has a smaller threshold voltage, while a programmed cell has relatively a higher threshold voltage.

In this context, when a memory cell is configured to threshold voltage level V , it would require read threshold voltage level greater than or equal to V in order to turn-on the corresponding FGT. Hence, considering an m -bit per cell flash memory, it requires at least $2^m - 1$ read operations to successfully distinguish between 2^m possible voltage levels, where each voltage represents a distinct cell value.

2.3 NAND Flash Memory Channel Model

In this thesis, without loss of generality, we focus on 2-bit per cell MLC NAND flash memory and describe the corresponding channel model. However, the work presented in this thesis can readily be extended for 3-bit or 4-bit per cell NAND flash technology. Due to various circuit-level noise components, the desired threshold voltage signal is severely affected and thus introduces serious data reliability issues. To this end, we model our NAND flash memory channel by incorporating the effects of programming noise, random telegraph noise (RTN), data retention noise and cell-to-cell interference (CCI) as major signal degradation components. Channel models similar to ours are extensively reported in the open literature [50–56]. Considering such noise impairments, the cell's threshold voltage signal V can be formulated with five additive noise components, given by

$$V = V_p + \{n_u + n_p + I + n_w + n_r\} \quad (2.1)$$

where $V_p \in \{V_{s_{11}}, V_{s_{10}}, V_{s_{00}}, V_{s_{01}}\}$ is the set of target verify voltage levels. The voltage level $V_{s_{11}}$ corresponds to the erased state, denoted by s_{11} , and the other three voltage levels $V_{s_{10}}, V_{s_{00}}$ and $V_{s_{01}}$ correspond to three programmed states, denoted by s_{10}, s_{00} and s_{01} , respectively. The n_u and n_p are the state-dependent programming noise components which tend to appear just after the ideal programming operation. The ideal programmed voltage level is further affected by the parasitic

FIGURE 2.3: Illustration of noisy NAND flash channel incorporating programming noise (n_u and n_p), CCI (I), RTN (n_w) and retention noise (n_r).

coupling-capacitance induced CCI I , the wear-out dependent RTN noise n_w and the storage time dependent data retention noise n_r . Due to the presence of such noise and interference effects, the NAND flash channel exhibits non-stationary behavior over varying program/erase (PE) cycles and data retention time (T). Fig. 2.3 illustrates this NAND flash channel model where the desired write voltage V_p is distorted by the noisy NAND flash channel and subsequently appeared as a noise-corrupted output voltage signal V after memory read operation.

2.3.1 Programming Noise

The state-dependent ISPP noise n_u only affects the programmed state memory cells. Under the iterative program-and-verify method, each programmed state is configured to the respective verify voltage level V_p through a series of program and verify voltage pulses which are applied across the cell's control gate. Using the ISPP programming approach, the threshold voltage signal corresponding to the programmed state memory cells tends to follow a uniform distribution model in the interval $[0, \Delta V_{pp}]$ [50–52, 56], given by

$$p_{n_u}(v) = \begin{cases} 0, & \text{for } V_p \in \{V_{s_{11}}\} \\ \frac{1}{\Delta V_{pp}}, \text{ if } 0 \leq v \leq \Delta V_{pp}, & \text{for } V_p \in \{V_{s_{10 \sim 01}}\} \end{cases} \quad (2.2)$$

The programming noise n_p affects both the erased and programmed state cells. It is caused by the manufacturer's process variations during the time of NAND flash fabrication. This noise component can be modeled by using the Gaussian distribution function [52–55], given by

$$p_{n_p}(v) = \begin{cases} \frac{1}{\sqrt{2\pi\sigma_e^2}} e^{-\frac{v^2}{2\sigma_e^2}}, & \text{for } V_p \in \{V_{s11}\} \\ \frac{1}{\sqrt{2\pi\sigma_p^2}} e^{-\frac{v^2}{2\sigma_p^2}}, & \text{for } V_p \in \{V_{s10 \sim 01}\} \end{cases} \quad (2.3)$$

where σ_e^2 and σ_p^2 are noise variance parameters, with $\sigma_e > \sigma_p$. Unfortunately, this initial threshold voltage distribution which is observed after ideal programming operation is subject to further distortion, mainly due to PE cycling effect, circuit-level coupling-capacitance and increasing data retention time.

2.3.2 Cell-to-Cell Interference (CCI)

The cell-to-cell interference (CCI) is induced due to the existence of parasitic coupling-capacitance between the neighboring memory cells. According to [50–52, 54, 56], the CCI component linearly adds to the threshold voltage signal, given by

$$I = \sum_k \Delta V_k \gamma_k \quad (2.4)$$

where ΔV_k is the change in the threshold voltage of interfering (neighboring) memory cells that is realized during their programming operation and γ_k is the coupling-capacitance ratio (CCR) between the victim and interfering cells. The CCI distribution function $p_I(v)$ is formulated in [28]. In current design practice, a victim cell can be interfered by three or five immediate neighboring cells, depending on the type of bit-line architecture being used, whereas the CCI from distant neighboring cells is assumed to be negligible. For instance, in an *even-odd* bit-line architecture, where the even bit-lines cells are programmed before the odd bit-lines

cells, there are five interfering cells for each even bit-line cell and three interfering cells for each odd bit-line cell. Alternatively, in the *all* bit-line architecture, where all word-line cells are programmed simultaneously, a victim cell is interfered by three neighboring cells located on the next word-line. Nevertheless, contrary to the non-stationary RTN and retention noise components, the CCI is a stationary and neighbor data-dependent component and thus can be estimated and subsequently neutralized either during cell programming [42] or during memory sensing operation [43, 57].

2.3.3 Random Telegraph Noise (RTN)

The recurring PE cycles in flash memory cause damage to the oxide layer of cell's FGT in the form of trap generation in the oxide and interface states [58]. These trap sites, in turn, can capture or release the charged particles which straightaway cause fluctuations in cell's threshold voltage signal. Such unintentional voltage signal fluctuations, however, increase the noise variance of NAND flash channel. This circuit-level threshold voltage distortion is known as the random telegraph noise (RTN). Note that the RTN is a non-stationary noise which varies with increasing PE cycles. Based upon empirical measurements, the RTN component is shown to widen the voltage distribution functions by introducing exponential tails around the target voltage V_p [59]. In this work, we model the RTN with zero-mean Gaussian distribution function, as

$$p_{n_w}(v) = \frac{1}{\sqrt{2\pi\sigma_w^2}} e^{-\frac{v^2}{2\sigma_w^2}} \quad (2.5)$$

where the noise standard deviation σ_w scales with memory PE cycles.

2.3.4 Data Retention Noise

The non-stationary data retention noise n_r is caused by the continuous charge leakage through cell's FGT over increasing data retention age (the length of time since memory cells were last programmed). The relentless leakage of electrons significantly attenuates (downward shifts) the cell's voltage level and eventually alters the state of the cell. The retention noise induced errors are among the leading source of errors in the new generation NAND flash memories. The retention noise distribution tends to scale with the initial voltage V_p , memory PE cycles and data retention time T . According to [50–56], the retention noise component can be approximated with Gaussian distribution function, given by

$$p_{n_r}(v) = \frac{1}{\sqrt{2\pi\sigma_r^2}} e^{-\frac{(v-\mu_r)^2}{2\sigma_r^2}} \quad (2.6)$$

where the state-dependent mean (μ_r) and standard deviation (σ_r) parameters are set according to [52], given by

$$\mu_r = (V_p - x_0) \cdot [A_t(PE)^{\alpha_i} + B_t(PE)^{\alpha_o}] \cdot \log(1 + T) \quad (2.7)$$

$$\sigma_r = 0.3 \mid \mu_r \mid \quad (2.8)$$

where x_0 , A_t , B_t , α_i and α_o are constant distribution parameters.

2.3.5 Overall Cell Threshold Voltage Distribution

The overall voltage distribution function for the erased state, denoted by $p_{s_{11}}(v)$, and three programmed states, denoted by $p_{s_{10}}(v)$, $p_{s_{00}}(v)$ and $p_{s_{01}}(v)$, respectively, can be computed as the convolution integral of all the additive noise components

FIGURE 2.4: Illustration of voltage distribution functions in 2-bit per cell NAND flash memory.

formulated in (2.1). Mathematically, it can be written as

$$p_{s_{11} \sim s_{01}}(v) = p_{n_u}(v) \otimes p_{n_p}(v) \otimes p_I(v) \otimes p_{n_w}(v) \otimes p_{n_r}(v) \quad (2.9)$$

where $\otimes$ is the convolution operation. This integral can be well-approximated by using a Gaussian mixture model with $\mu_{s_{11} \sim s_{01}}$ and $\sigma_{s_{11} \sim s_{01}}^2$ representing the distribution mean and variance parameters.

For all our simulations, we set the following flash memory parameters according to [52]: $V_{s_{11}} = 1.4$, $V_{s_{10}} = 2.6$, $V_{s_{00}} = 3.2$, $V_{s_{01}} = 3.93$, $\Delta V_{pp} = 0.20$, $\sigma_e = 0.35$, $\sigma_p = 0.05$, $\sigma_w = 0.00027(PE)^{0.62}$, $x_0 = 1.4$, $A_t = 0.000035$, $B_t = 0.000235$, $\alpha_i = 0.62$ and $\alpha_o = 0.30$. In Fig. 2.4, we illustrate an example of voltage distribution functions correspond to 2-bit per cell NAND flash memory.

2.4 Chapter Summary

This chapter presents an overview of NAND flash memory and describes the basic *program*, *erase* and *read* operations associated with NAND flash memory. Furthermore, a 2-bit per cell MLC NAND flash memory channel model is introduced which includes the programming noise, cell-to-cell interference (CCI), random telegraph noise (RTN) and retention noise as main source of voltage signal impairments. The presented channel model shows a non-stationary behavior over varying PE cycles

TABLE 2.1: Important features of different NAND flash channel noise components.

Important Features	Programming Noise (n_u, n_p)	CCI (I)	RTN (n_w)	Retention Noise (n_r)
Stationary	✓	✓	✗	✗
Own data-dependent	✓	✗	✗	✓
Neighbor data-dependent	✗	✓	✗	✗
Varies within a code-word	✓	✓	✗	✓
Varies from memory block-to-block	✗	✓	✓	✓

and data retention time. The statistical models for these noise sources are described and the overall voltage distribution expression is presented. The important features of different NAND flash channel noise components are summarized in Table 2.1.

Chapter 3

Read and Write Voltage Signal Optimization

The multi-level per cell (MLC) NAND flash channel exhibits non-stationary behavior over increasing program and erase (PE) cycles and data retention time. In this chapter, we present an optimization method for adjusting the read and write voltage levels to adapt to the non-stationary MLC NAND flash channel. Using the NAND flash channel model of Chapter 2, which incorporates the programming noise, CCI, RTN and data retention noise as major signal degradation components, the write voltage levels are optimized to minimize the channel raw error probability. Moreover, for selecting the quantization levels for the read threshold voltage to facilitate soft-decision low-density parity-check (LDPC) decoding, we introduce an entropy-based function by which the voltage erasure regions (error dominating regions) are controlled to produce the lowest LDPC encoded bit/frame error-rate performance. The proposed read and write voltage optimization schemes not only minimize the error probability throughout the operational lifetime of flash memory, but also improve the LDPC decoding convergence speed.

3.1 Background

In NAND flash memory, cells are configured to distinct write voltage levels to store the set of all possible data symbols. Considering an example of 2-bit per cell flash memory, a memory cell can be configured to four distinct voltage levels, denoted by V_{s11} , V_{s10} , V_{s00} and V_{s01} , for representing data symbols “11”, “10”, “00” and “01”, respectively, where V_{s11} is the verify voltage of the erased state and V_{s01} is the verify voltage of the programmed state configured with the highest write voltage level (see Fig. 3.1). It is observed that the voltage distribution functions corresponding to these different cell states are not uniform due to the presence of non-stationary and asymmetric channel noise. In this scenario, V_{s11} , V_{s10} , V_{s00} and V_{s01} should not be equally-spaced but should be determined based on some optimization criteria.

To read the data stored in a flash memory cell, the amount of stored electrical charge is measured by applying a set of read threshold levels in discrete steps. For a 2-bit per cell NAND flash memory, it requires at least three read voltage levels, denoted by R_1 , R_2 and R_3 , where R_1 can be set between V_{s11} and V_{s10} to distinguish between symbols “11” and “10”, R_2 can be set between V_{s10} and V_{s00} to distinguish between symbols “10” and “00” and R_3 can be set between V_{s00} and V_{s01} to distinguish between symbols “00” and “01”, as shown in Fig. 3.1. Since the write voltage levels are not equally-spaced, the read threshold levels should not be uniformly spaced but should be appropriately designed so that the error-rate is minimized.

For a state-of-the-art NAND flash memory, the 3-level memory sensing scheme may not be sufficient to achieve the satisfactory coded error-rate performance. This is because the modern flash memory chips experience severe voltage level distortions due to circuit-level noise and interference effects which significantly affect the data reliability of NAND flash memory [60–62]. To overcome the reliability issues, strong error correcting codes (ECCs), such as LDPC codes, are employed [28–33]. To reap the full benefit of LDPC code, it is desirable to have LDPC decoder’s input values

FIGURE 3.1: Illustration of four write/verify voltage levels (V_{s11} , V_{s10} , V_{s00} and V_{s01}) and three read threshold voltage levels (R_1 , R_2 and R_3) in 2-bit per cell NAND flash memory.

quantized as finely as possible, so that soft BP decoding can be performed. This requires high-precision memory sensing, and consequently more read latency, which may not be acceptable for the latency-sensitive NAND flash memory.

There are several prior-art works on the read and write voltage signal design. In [63], the write voltage levels are optimized based on a simple flash channel model by considering the random telegraph noise (RTN) as the only source of voltage signal degradation. Another relevant work is reported in [64] where the write voltage levels are optimized assuming a symmetric AWGN channel for NAND flash memory. The work presented in this chapter optimizes the write voltage levels using a more comprehensive asymmetric NAND flash channel model that includes the effects of programming noise, CCI as well as non-stationary RTN and data retention noise.

Instead of MLC technology, where discrete write voltage levels are used to store information, relative value (ordering) of voltage levels can instead be used to represent the stored data, as proposed in rank modulation scheme [65]. In rank modulation, non-overlapping voltage regions are used to represent the various data symbols. In [66], these continuous voltage regions are optimized. The rank modulation scheme can increase the cell storage capacity and overcome the programming overshoot errors. However, to implement this scheme, a large number of read operations are required to learn the voltage level ordering. Thus, we focus on MLC technology in this thesis and strive to optimize the corresponding discrete write voltage levels.

In the conventional MLC flash technology, to avoid the effect of voltage overshoot errors, the cell's threshold voltage is configured on desired write voltage levels in multiple rounds of programming. In [66–68], for both MLC and rank modulation schemes, the optimal programming step size has been investigated, keeping the number of programming rounds constant and assuming fixed write voltage levels. In this chapter, we address the problem of finding the optimized write voltage levels to ensure that the flash channel raw error probability is minimized.

With regards to the read voltage quantization for soft-decision BP decoding, multiple memory read operations are typically performed. In this direction, a simplistic approach would be to apply equally spaced read voltage levels (uniform memory sensing). However, uniform level quantization is not suitable for NAND flash channel and yields poor error-rate performance [28]. For this reason, a non-uniform quantization scheme is reported in [28], where the quantization levels are obtained at the intersecting region between two adjacent distribution functions by using a constant ratio method. Alternative to enhanced precision, hard-decision based dynamic quantization schemes are reported in [36, 69] where the read voltage levels are adjusted according to the non-stationary behavior of flash channel.

Another important work related to quantization design is presented in [70] in which the quantization levels are obtained by maximizing the mutual-information (MMI) between flash channel's input and output voltage signals. Assuming an infinite long code-word size, the MMI quantization can produce an optimal error-rate performance. However, this assumption does not hold in practice due to the finite length code-word constraint. To overcome this shortcoming, we propose a new read signal quantization scheme based on a novel voltage entropy function that is able to give better LDPC decoded error-rate performance than the MMI scheme.

3.2 Write Voltage Signal Optimization

For a non-stationary flash memory channel, which varies w.r.t. PE cycles and data retention time T , it is important to optimize and update the write voltage levels for the optimal error-rate performance. It should be noted that the CCI is a stationary random process, hence, it does not affect the optimization of write voltage levels. In this chapter, it is therefore assumed that the CCI is compensated during cell programming via signal pre-compensation method [42]. In current design practice, flash memory controller keeps track of the PE count history of individual memory blocks. The PE count history is typically recorded inside the memory controller to perform the flash wear-leveling [19–21]. However, the data retention time is generally difficult to predict at the time of cell programming, hence the retention time is typically set to zero ($T = 0$, representing the early (short) retention time) or to some large value (e.g. $T = 1$ year, representing the flash memory’s end-of-retention). Given these two flash channel parameters, the four distribution functions given by (2.9) can be evaluated, paving the way for finding the optimal write voltage levels. In other words, given the minimum and maximum write voltage levels V_{s11} and V_{s01} (based on the device physics), and the current PE count, we strive to optimize the intermediate write voltage levels (V_{s10} and V_{s00}). To answer this question, we formulate the probability of error expression (P_e) for the flash channel. This expression can be defined using the conditional error probability of individual symbols $p_{e|s_p}(v)$, given as

$$P_e = \frac{1}{4} \{p_{e|s11}(v) + p_{e|s10}(v) + p_{e|s00}(v) + p_{e|s01}(v)\} \quad (3.1)$$

where $1/4$ models the equi-probable input data symbols. To express $p_{e|s_p}(v)$, we need the decision boundaries R_1 , R_2 and R_3 between adjacent distribution functions

as shown in Fig. 3.1. Following this figure, we can define

$$\begin{aligned} p_{e|s_{11}}(v) &= p_{s_{11}}(v > R_1) \\ &= Q\left(\frac{R_1 - \mu_{s_{11}}}{\sigma_{s_{11}}}\right) \end{aligned} \quad (3.2)$$

$$\begin{aligned} p_{e|s_{10}}(v) &= p_{s_{10}}(v < R_1) + p_{s_{10}}(v > R_2) \\ &= 1 - Q\left(\frac{R_1 - \mu_{s_{10}}}{\sigma_{s_{10}}}\right) + Q\left(\frac{R_2 - \mu_{s_{10}}}{\sigma_{s_{10}}}\right) \end{aligned} \quad (3.3)$$

$$\begin{aligned} p_{e|s_{00}}(v) &= p_{s_{00}}(v < R_2) + p_{s_{00}}(v > R_3) \\ &= 1 - Q\left(\frac{R_2 - \mu_{s_{00}}}{\sigma_{s_{00}}}\right) + Q\left(\frac{R_3 - \mu_{s_{00}}}{\sigma_{s_{00}}}\right) \end{aligned} \quad (3.4)$$

$$\begin{aligned} p_{e|s_{01}}(v) &= p_{s_{01}}(v < R_3) \\ &= 1 - Q\left(\frac{R_3 - \mu_{s_{01}}}{\sigma_{s_{01}}}\right) \end{aligned} \quad (3.5)$$

where $Q(x)$ -function¹ represents the tail probability of standard normal distribution.

The voltage distribution mean and standard deviation parameters, $\mu_{s_{11} \sim s_{01}}$ and $\sigma_{s_{11} \sim s_{01}}$, can be approximated by

$$\begin{aligned} \mu_{s_{11}} &= V_{s_{11}} - \mu_{r_{s_{11}}} \\ \mu_{s_{10} \sim s_{01}} &= V_{s_{10} \sim s_{01}} + \frac{\Delta V_{pp}}{2} - \mu_{r_{s_{10} \sim s_{01}}} \\ \sigma_{s_{11}} &= \sqrt{\sigma_e^2 + \sigma_w^2 + \sigma_{r_{s_{11}}}^2} \\ \sigma_{s_{10} \sim s_{01}} &= \sqrt{\sigma_p^2 + \sigma_w^2 + \sigma_{r_{s_{10} \sim s_{01}}}^2} \end{aligned}$$

where ΔV_{pp} denotes the ISPP voltage step size, and σ_e^2 , σ_p^2 , σ_w^2 and $\sigma_{r_{s_{10} \sim s_{01}}}^2$ denote the noise distribution variance for programming, RTN and retention noise components, respectively, as explained in Chapter 2. The decision boundaries R_1 , R_2 and R_3 can also be considered as hard-decision levels on individual symbols. To get these decision boundaries, using the notion of maximum likelihood detection, we

¹ $Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty \exp\left(-\frac{u^2}{2}\right) du.$

equate the adjacent distribution functions and solve for the common intersecting point. Thus, for obtaining the expression for R_1 , R_2 and R_3 , we solve the following three equations

$$p_{s_{11}}(v = R_1) = p_{s_{10}}(v = R_1) \quad (3.6)$$

$$p_{s_{10}}(v = R_2) = p_{s_{00}}(v = R_2) \quad (3.7)$$

$$p_{s_{00}}(v = R_3) = p_{s_{01}}(v = R_3) \quad (3.8)$$

By simplifying the above expressions, we get R_1 , R_2 and R_3 , given by

$$R_1 = \frac{-\left(\mu_{s_{11}}\sigma_{s_{10}}^2 - \mu_{s_{10}}\sigma_{s_{11}}^2\right) - \sqrt{\left(\mu_{s_{11}}\sigma_{s_{10}}^2 - \mu_{s_{10}}\sigma_{s_{11}}^2\right)^2 - \left(\sigma_{s_{11}}^2 - \sigma_{s_{10}}^2\right)(c_1)}}{\left(\sigma_{s_{11}}^2 - \sigma_{s_{10}}^2\right)} \quad (3.9)$$

$$\text{where } c_1 = \mu_{s_{10}}^2\sigma_{s_{11}}^2 - \mu_{s_{11}}^2\sigma_{s_{10}}^2 - 2\sigma_{s_{11}}^2\sigma_{s_{10}}^2 \log\left(\frac{\sigma_{s_{11}}}{\sigma_{s_{10}}}\right)$$

$$R_2 = \frac{-\left(\mu_{s_{10}}\sigma_{s_{00}}^2 - \mu_{s_{00}}\sigma_{s_{10}}^2\right) - \sqrt{\left(\mu_{s_{10}}\sigma_{s_{00}}^2 - \mu_{s_{00}}\sigma_{s_{10}}^2\right)^2 - \left(\sigma_{s_{10}}^2 - \sigma_{s_{00}}^2\right)(c_2)}}{\left(\sigma_{s_{10}}^2 - \sigma_{s_{00}}^2\right)} \quad (3.10)$$

$$\text{where } c_2 = \mu_{s_{00}}^2\sigma_{s_{10}}^2 - \mu_{s_{10}}^2\sigma_{s_{00}}^2 - 2\sigma_{s_{10}}^2\sigma_{s_{00}}^2 \log\left(\frac{\sigma_{s_{10}}}{\sigma_{s_{00}}}\right)$$

$$R_3 = \frac{-\left(\mu_{s_{00}}\sigma_{s_{01}}^2 - \mu_{s_{01}}\sigma_{s_{00}}^2\right) - \sqrt{\left(\mu_{s_{00}}\sigma_{s_{01}}^2 - \mu_{s_{01}}\sigma_{s_{00}}^2\right)^2 - \left(\sigma_{s_{00}}^2 - \sigma_{s_{01}}^2\right)(c_3)}}{\left(\sigma_{s_{00}}^2 - \sigma_{s_{01}}^2\right)} \quad (3.11)$$

$$\text{where } c_3 = \mu_{s_{01}}^2\sigma_{s_{00}}^2 - \mu_{s_{00}}^2\sigma_{s_{01}}^2 - 2\sigma_{s_{00}}^2\sigma_{s_{01}}^2 \log\left(\frac{\sigma_{s_{00}}}{\sigma_{s_{01}}}\right)$$

The detailed steps for the derivation of (3.9), (3.10) and (3.11) are provided in Appendix A. Given the conditional symbol error probabilities (3.2)-(3.5), we can re-write (3.1) in terms of $V_{s_{10}}$, $V_{s_{00}}$, PE and T , and optimize the desired parameters $V_{s_{10}}$ and $V_{s_{00}}$. In this section, we perform the voltage optimization for $T = 0$. The end-of-retention optimization will be discussed in Section 3.5. Thus, instead of keeping the write voltage levels fixed for the entire operational lifetime of NAND flash memory, we propose to adapt them according to the current channel condition such that the error probability is minimized. From the practical implementation

perspective, the write voltage levels can be pre-computed as a function of PE cycles and stored into a look-up table to preclude the run-time optimization complexity. To formulate the optimization problem, we define the objective function by using (3.1), given as

$$(V_{s10}^*, V_{s00}^*) = \min_{(V_{s10}, V_{s00})} P_e(V_{s10}, V_{s00}, PE, T = 0) \quad (3.12)$$

This optimization function behaves as a convex function over V_{s10} and V_{s00} as shown in Fig. 3.2, and thus can be solved by using any convex optimization technique. Since the write voltage levels are optimized in off-line mode, the complexity of the chosen optimization algorithm does not affect the performance of NAND flash system. In this work, we apply the gradient descent (GD) method [71] to find the optimal V_{s10}^* and V_{s00}^* that yield the minimum P_e . The GD method minimizes the objective function P_e by iteratively solving the following equation

$$\begin{bmatrix} V_{s10}^{(k+1)} \\ V_{s00}^{(k+1)} \end{bmatrix} = \begin{bmatrix} V_{s10}^{(k)} \\ V_{s00}^{(k)} \end{bmatrix} - \alpha \begin{bmatrix} \frac{\partial P_e}{\partial V_{s10}} \\ \frac{\partial P_e}{\partial V_{s00}} \end{bmatrix} P_e(V_{s10}^{(k)}, V_{s00}^{(k)}, PE, T = 0) \quad (3.13)$$

where ∂ denotes the partial derivative, and k and α are the GD iteration count and step size parameters, respectively.

The optimized write voltage levels along with the respective minimized P_e values over different PE cycles are enumerated in Table. 3.1. To make comparison between the proposed write voltage optimization scheme and the conventional channel invariant (fixed) write voltage scheme, we plot the corresponding P_e functions in Fig. 3.3. In this figure, the solid curve represents the minimized P_e function obtained against the optimal write voltage levels, as given in Table 3.1, whereas the dotted curve shows the P_e values computed against channel invariant write voltage levels, keeping the intermediate voltage levels fixed at $V_{s10} = 2.6$ and $V_{s00} = 3.2$ [52]. As expected, the proposed channel optimized write voltage scheme yields lower

TABLE 3.1: Optimized write voltage levels over varying PE cycles.

PE	V_{s10}^*	V_{s00}^*	P_e
1K	2.77	3.35	7.15×10^{-4}
2K	2.75	3.34	0.0010
5K	2.69	3.31	0.0023
10K	2.61	3.27	0.0072
15K	2.55	3.24	0.0115

FIGURE 3.2: Plot of channel raw error probability (P_e) as a function of write voltage levels V_{s10} and V_{s00} , setting PE=5K and $T = 0$.

error probability, particularly at low-to-medium PE count. In addition to improved error-rate performance, the optimal write voltage levels also help to reduce the ECC decoding latency, as discussed in Section 3.4.

3.3 Quantization Design for the Read Threshold Voltage Levels

After write voltage optimization, we present a novel quantization scheme, using the voltage entropy function, to design the read threshold voltage levels. For 2-bit per cell NAND flash memory, it require at least three read threshold voltage levels

FIGURE 3.3: Comparison of channel raw error probability (P_e) between the proposed write voltage optimization scheme and the conventional channel invariant write voltage scheme: *solid curve* (V_{s10}^*, V_{s00}^* from Table 3.1), *dotted curve* ($V_{s10} = 2.6, V_{s00} = 3.2$).

to detect four possible stored data symbols. However, when LDPC code is used as ECC for flash channel, we may need to perform high precision memory sensing for more accurate computation of log-likelihood-ratios (LLR) information for soft LDPC decoding. To achieve this, one straightforward approach is to set the read threshold levels between the minimum and maximum write voltage levels, V_{s11} and V_{s01} , with equally-spaced separation D_u . Fig. 3.4 shows a pictorial representation of 6-level uniform quantization scheme, where each vertical dashed-line (R_1 to R_6) represent a particular quantization level. Since each cell value represents 2 bits of information, we compute the LLR values corresponding to the most significant bit and the least significant bit positions, denoted by L_{msb} and L_{lsb} , respectively. Given the read threshold voltage v , when $R_{n-1} \leq v \leq R_n$ for $n = 1, 2, 3, 4, 5, 6, 7$, where $R_0 = -\infty$ and $R_7 = +\infty$, we have

$$L_{msb}(n) = \log \frac{\int_{R_{n-1}}^{R_n} \{p_{s00}(v) + p_{s01}(v)\} dv}{\int_{R_{n-1}}^{R_n} \{p_{s10}(v) + p_{s11}(v)\} dv} \quad (3.14)$$

FIGURE 3.4: Illustration of *uniform interval* D_u based quantization scheme in a 2-bit per cell flash memory with 6 (R_1 to R_6) equi-spaced read threshold voltage levels.

$$L_{\text{lsb}}(n) = \log \frac{\int_{R_{n-1}}^{R_n} \{p_{s_{00}}(v) + p_{s_{10}}(v)\} dv}{\int_{R_{n-1}}^{R_n} \{p_{s_{01}}(v) + p_{s_{11}}(v)\} dv} \quad (3.15)$$

However, assuming a fixed memory sensing precision (e.g. 6-level, 9-level, 12-level, etc.), the uniform quantization levels tend to become non-optimal when compared to the non-uniform quantization levels [28]. The objective of a non-uniform quantization scheme is to set the memory sensing levels closer to the erasure (intersection) regions where adjacent distribution functions tend to overlap. This helps to distinguish between adjacent voltage levels. Since the symbol error probability is dominant in the erasure region, it therefore requires more accurate memory sensing.

3.3.1 Entropy Quantization Scheme

For designing a non-uniform quantization scheme, it is crucial to identify the optimum width of erasure region so that the resultant soft-information (LLR values) achieves the best decoder error performance. For this purpose, we first define the entropy of the cell's threshold voltage, denoted by $H(v)$, given as

$$H(v) = \sum_i \left[\frac{p_{s_i}(v)}{\sum_i p_{s_i}(v)} \log_2 \left(\frac{\sum_i p_{s_i}(v)}{p_{s_i}(v)} \right) \right] \quad (3.16)$$

FIGURE 3.5: Illustration of *entropy quantization* scheme for 2-bit per cell flash memory: quantization levels (R_1 to R_6) are placed closer to the erasure (overlapping) regions.

where $s_i \in \{s_{11}, s_{10}, s_{00}, s_{01}\}$. An example of voltage entropy function is illustrated in Fig. 3.5. Note that the voltage entropy is only dominant within the erasure region, denoted as *high entropy region*. This should be the targeted memory sensing region as it incurs high error probability. Therefore, we set the quantization levels closer to the erasure region, instead of setting them uniformly from $V_{s_{11}}$ to $V_{s_{01}}$. With 3 erasure regions in-place, we require at least 6 quantization levels in order to enclose them within the quantization boundary. To this end, we define an entropy parameter θ in the range $[0, 1]$ to select the width of each erasure region. To be precise, we set each quantization level such that

$$H(R_n) = \theta \quad (3.17)$$

for $n = 1, 2, 3, 4, 5, 6$. In other words, the memory sensing levels are designed where the voltage entropy is equal to θ . By varying the entropy parameter θ , we can obtain the desired width for erasure regions. We refer to this memory sensing scheme as the *entropy quantization* scheme.

Besides, another well-known non-uniform quantization scheme for MLC flash is reported in [70], where the quantization levels are obtained by maximizing the

mutual-information (MMI) between the input and output of flash channel. This scheme involves solving a multi-variable optimization problem to get the quantization levels. However, MMI quantization does not allow to alter the width of erasure regions. This constraint affects the resultant error-rate performance of LDPC decoder, which is further discussed in the following section. Since both proposed and MMI schemes employ an entropy like function to optimize the quantization levels, we mainly treat MMI quantization as the benchmark for comparison.

To study the performance of the proposed entropy quantization, we simulate the following two binary LDPC codes:

- 8K-code is (8000, 7200), rate-0.90 regular LDPC code with uniform column weight 5.
- 4K-code is (4544, 4096), rate-0.90 irregular LDPC code with following degree distribution:

$$\lambda(x) = 0.0682x + 0.1822x^2 + 0.1329x^3 + 0.6167x^4$$

$$\rho(x) = 0.22x^{38} + 0.78x^{39}$$

where $\lambda(x)$ and $\rho(x)$ are the variable node and check node degree distribution pairs, respectively, optimized through density evolution [72]. Both LDPC codes are constructed using the progressive-edge-growth (PEG) algorithm [73] which is well-known to construct finite-length LDPC codes with very good error performance by ensuring a large girth of the underlying code graph. These LDPC codes are decoded using the shuffled belief-propagation (BP) decoder [74]. The maximum number of BP iterations, denoted by $(I_{\max})$, is set to 25. We apply the proposed entropy based quantization scheme (6-level) and compare with the MMI (6-level) [70] and uniform (12-level) quantization schemes. The entropy parameter is set to $\theta = 0.35$ for 4K-code and $\theta = 0.40$ for 8K-code through exhaustive search method as these values lead to optimum error-rate performance. The entropy based optimized read threshold voltage levels for 4K-code over a range of PE cycles are given in

TABLE 3.2: Optimized read threshold voltage levels $\{R_1, R_2, R_3, R_4, R_5, R_6\}$ for 4K-code over varying PE cycles.

PE	R_1	R_2	R_3	R_4	R_5	R_6
18K	2.21	2.44	2.95	3.10	3.65	3.80
19K	2.20	2.44	2.94	3.10	3.65	3.80
20K	2.18	2.43	2.93	3.10	3.64	3.80
21K	2.18	2.43	2.93	3.09	3.64	3.81
22K	2.16	2.43	2.92	3.09	3.63	3.81
23K	2.15	2.42	2.91	3.09	3.62	3.81
24K	2.14	2.42	2.90	3.09	3.62	3.81
25K	2.13	2.41	2.89	3.08	3.61	3.81

Table 3.2. Note that the optimization of entropy parameter θ needs to be done only once in the offline mode.

We first notice that, contrary to MMI quantization, the entropy quantization scheme does not maximize the mutual information as shown in Fig. 3.6 for 4K-code. Next, we illustrate the frame-error-rate (FER) curves for LDPC 4K-code and 8K-code in Fig. 3.7 and Fig. 3.8, respectively. In these two figures, we show the error performance of MMI and uniform quantization schemes by employing both channel invariant (conventional) and channel optimized (proposed) write voltage levels. We can observe that the proposed entropy-based quantization outperforms the former schemes. Furthermore, as stated earlier, the uniform quantization scheme is not very effective as the 12-level quantization is, surprisingly, worse than the 6-level non-uniform quantization schemes.

It should be noted that, in this chapter, we only strive to optimize 6 read threshold voltage levels. This is because the cell storage capacity with 6-level quantization registers a big jump from 3-level quantization (hard decision), while further increase in the quantization levels registers diminishing returns towards the infinite-precision limit (ideal soft decoding), as shown in Fig. 3.9. Note that the cell storage capacity measures the symmetric information rate, given by

$$\max_{\{R_1, \dots, R_Q\}} \frac{1}{4} \sum_{s \in \{s_{11}, \dots, s_{01}\}} \left[\sum_{i=1}^{Q+1} \int_{R_{i-1}}^{R_i} p_s(v) \log \left(\frac{p_s(v)}{P(v)} \right) dv \right]$$

FIGURE 3.6: Comparison of *mutual information* computed by employing 6-level Entropy (proposed) and 6-level MMI quantization schemes.

FIGURE 3.7: Frame-error-rate (FER) performance of LDPC 4K-code by employing the Entropy (proposed), MMI and uniform quantization schemes: *solid curves* are plotted for optimized write voltage levels and *dotted curves* are plotted for channel invariant write voltage levels.

$$P(v) = \frac{1}{4} \sum_{s \in \{s_{11}, \dots, s_{01}\}} p_s(v)$$

where Q is the number of quantization levels, $R_0 = -\infty$ and $R_{Q+1} = +\infty$. In this figure, we have plotted the cell storage capacity curves against the memory PE cycles for 3-, 6-, 9-, 12-, and ideal soft quantization schemes. To further analyze the

FIGURE 3.8: Frame-error-rate (FER) performance of LDPC 8K-code by employing the Entropy (proposed), MMI and uniform quantization schemes: *solid curves* are plotted for optimized write voltage levels and *dotted curves* are plotted for channel invariant write voltage levels.

FIGURE 3.9: Plot of *cell storage capacity* by employing 3-, 6-, 9-, 12- and ideal *soft decoding* schemes.

effect of increasing quantization levels, in Fig. ??, we only the FER verses memory PE cycles for different quantization levels. As can be seen, the advantage beyond 6-level quantization is diminishing.

FIGURE 3.10: Frame-error-rate (FER) for entropy quantization scheme by employing 3-, 6-, 9-, 12- and ideal *soft* decoding schemes.

3.3.2 Controlled Erasure Decoding using Entropy Quantization

We now investigate the reason behind the improvement in error-rate performance achieved by using the proposed entropy based quantization scheme over the MMI quantization scheme [70]. We first draw attention to the fact that the entropy based quantization levels (given by the optimal value of θ) tend to produce wider erasure regions than the MMI quantization levels, as depicted in Fig. 3.11. In this figure, we observe that the threshold voltage range is split into 3 erasure regions $\mathcal{E}_1$, $\mathcal{E}_2$ and $\mathcal{E}_3$, and 4 data regions $\mathcal{D}_1$, $\mathcal{D}_2$, $\mathcal{D}_3$ and $\mathcal{D}_4$, respectively. Here, it must be realized that adjusting the width of erasure regions affects the error performance of the code, as shown in Fig. 3.12. This plot shows the error performance of LDPC 4K-code at PE = 21K and LDPC 8K-code at PE = 21.5K over a range of θ values. The width of erasure regions and the respective read threshold levels corresponding to different θ values are shown in Table. 3.3. Observing Fig. 3.12 and Table. 3.3 reveals that if the erasure region is too narrow, the decoder will not be able to

FIGURE 3.11: Comparison of 6-level memory sensing between the Entropy (proposed) and MMI quantization schemes: Entropy quantization levels contribute to wider erasure regions.

determine sufficient erasures; while if the erasure region is too wide, it will get too many erased symbols which exceed the code's error correcting capability; and the entropy parameter θ allows the erasure region width to be optimized. With MMI quantization, it is not possible to alter the width of erasure region as it is optimized (fixed) over a given channel (for a fixed channel noise), irrespective of the type of code used. Fig. 3.12 also marks the entropy parameter value corresponding to the MMI quantization levels, which clearly does not coincide with the minimum error-rate point. This is because the MMI quantization optimization attempts to achieve zero pre-decoded error probability by assuming infinite channel code length, but in practical systems this assumption is violated as only finite block-length codes can be adopted. In contrast, the proposed entropy based quantization is able to optimize the post-decoded error probability and thus may arrive at a different optimum θ for a different code. This observation is shown in Fig. 3.12 where 4K-code and 8K-codes have produced different optimum θ values.

In entropy based quantization, changing the value of θ influences the symbol error and erasure probabilities of the flash channel, denoted by P_{error} and P_{erasure} , respectively, and consequently alters the magnitude of input LLRs used in BP decoding. Here, P_{error} refers to the uncoded error probability computed in 4 data regions and P_{erasure} refers to the erasure probability computed in 3 erasure regions.

FIGURE 3.12: Bit-error-rate/Frame-error-rate (BER/FER) performance of LDPC 4K-code and 8K-code plotted against Entropy parameter θ : values of θ corresponding to optimal error-rate performance are marked with vertical arrows and the MMI quantization levels corresponding to θ is marked with slanted arrow.

TABLE 3.3: Width of erasure regions $\mathcal{E}_1$, $\mathcal{E}_2$ and $\mathcal{E}_3$ and the corresponding read threshold voltage levels over varying entropy parameter θ , keeping the PE cycles fixed at 21K.

θ	$\mathcal{E}_1$	$\mathcal{E}_2$	$\mathcal{E}_3$	R_1	R_2	R_3	R_4	R_5	R_6
0.20	0.336	0.221	0.220	2.152	2.488	2.904	3.125	3.614	3.384
0.25	0.303	0.202	0.202	2.163	2.466	2.914	3.116	3.624	3.826
0.30	0.277	0.185	0.184	2.172	2.449	2.922	3.107	3.632	3.816
0.35	0.253	0.171	0.170	2.180	3.433	2.929	3.100	3.639	3.809
0.40	0.233	0.158	0.156	2.188	2.421	2.936	3.094	3.646	3.802
0.45	0.214	0.146	0.144	2.195	2.409	2.942	3.088	3.652	3.796
0.50	0.197	0.135	0.133	2.202	2.399	2.947	3.082	3.657	3.790
0.55	0.181	0.124	0.122	2.208	2.389	2.953	3.077	3.663	3.785
0.60	0.166	0.114	0.112	2.215	2.381	2.958	3.072	3.668	3.780

These two quantities are mathematically written as

$$\begin{aligned}
P_{\text{error}} = & \int_{v \in \mathcal{D}_1} (p_{s_{10}}(v) + p_{s_{00}}(v) + p_{s_{01}}(v)) dv + \int_{v \in \mathcal{D}_2} (p_{s_{11}}(v) + p_{s_{00}}(v) + p_{s_{01}}(v)) dv \\
& + \int_{v \in \mathcal{D}_3} (p_{s_{11}}(v) + p_{s_{10}}(v) + p_{s_{01}}(v)) dv + \int_{v \in \mathcal{D}_4} (p_{s_{11}}(v) + p_{s_{10}}(v) + p_{s_{00}}(v)) dv
\end{aligned} \tag{3.18}$$

$$\begin{aligned}
P_{\text{erasure}} = & \int_{v \in \mathcal{E}_1} (p_{s_{11}}(v) + p_{s_{10}}(v) + p_{s_{00}}(v) + p_{s_{01}}(v)) dv \\
& + \int_{v \in \mathcal{E}_2} (p_{s_{11}}(v) + p_{s_{10}}(v) + p_{s_{00}}(v) + p_{s_{01}}(v)) dv \\
& + \int_{v \in \mathcal{E}_3} (p_{s_{11}}(v) + p_{s_{10}}(v) + p_{s_{00}}(v) + p_{s_{01}}(v)) dv
\end{aligned} \tag{3.19}$$

We evaluate and plot these two expressions for different PE cycles in Fig. 3.13. In this figure, compared to MMI quantization, the proposed entropy based quantization reduces the symbol error probability by having a higher erasure probability. It should be noted that the LLR values associated with erasure regions are the least reliable LLRs and therefore reset to zero (approximately erased). To be precise, the LSB LLRs $L_{\text{lsb}}(2)$ and $L_{\text{lsb}}(6)$ corresponding to $\mathcal{E}_1$ and $\mathcal{E}_3$, respectively, and the MSB LLR $L_{\text{msb}}(4)$ corresponding to $\mathcal{E}_2$ are erased. This is because the LSB values in region $n = 2$ ($\mathcal{E}_1$) and $n = 6$ ($\mathcal{E}_3$) are undefined. Similarly, the MSB value in region $n = 4$ ($\mathcal{E}_2$) is undefined. Thus, with entropy based quantization scheme, we perform a controlled erasure decoding by means of optimizing the parameter θ , in such a way that some LLR values are erased to ensure that the symbol error probability is reduced. This adjustment between the symbol error and symbol erasure probability helps the entropy scheme to perform better than the MMI quantization scheme.

3.4 Write Voltage Optimization for Faster Decoding Convergence

In the previous section, we mainly analyze the improvement in error-rate performance that comes as a result of employing the optimized write voltage signals, as shown in Fig. 3.3, 3.7 and 3.8. Note that the need for write voltage optimization is not critical in the initial state of flash memory as compared to its end-of-life (EOL). This is because the raw memory errors in the initial state typically does not exceed the error correction capability of ECC decoder. However, as the LDPC code

FIGURE 3.13: Comparison of symbol error probability (P_{error}) and symbol erasure probability (P_{erasure}) between the Entropy (proposed) and MMI quantization schemes: Entropy quantization produces lower P_{error} at the expense of higher P_{erasure} .

is becoming the mainstream ECC for flash memory, its long decoding latency starts to deteriorate the system performance. In this situation, the proposed optimization method to reduce the error-rate, and consequently to reduce the decoding latency, becomes important. In other words, we can improve the decoding convergence in the initial state of flash memory by optimizing the write voltage levels.

In Fig. 3.14, we plot the LDPC 4K-code BER curves versus maximum decoder iteration (I_{max}). It can be observed that the decoding latency is reduced by up to 16% in the initial state, even though the BER at these early states are low and typically have enough margins from ECC failures, and up to 22% in the EOL. Therefore, in comparison with conventional flash memory system where the write voltage levels are fixed, our approach is more effective in terms of error performance and decoding latency for both early and late stages of flash memory usage.

FIGURE 3.14: Bit-error-rate (BER) performance of LDPC 4K-code plotted versus decoding iteration ($I_{\max}$) for different PE cycles: *solid* and *dotted* curves represent channel optimized and channel invariant (fixed) write voltage schemes, respectively.

3.5 Early vs. End-of-Retention (EOR) Write Voltage Optimization

Previously, we performed the write voltage optimization for the early retention time of flash memory, by setting the retention time to zero. Now, we address the write voltage optimization for the flash memory's end-of-retention (EOR) time, and compare between the two voltage design schemes. In this direction, we set the retention time to some large value and then minimize the objective function (3.12). As a case study, we set the retention time to 12 months and optimize the write voltage levels over varying PE cycles. The optimization for EOR is recommended if the flash data is expected to be stored for a long period of time without reading/re-writing. We use $T_{\text{write}} = 0$ and $T_{\text{write}} = 12\text{-months}$ to distinguish between optimizing the write voltage schemes for early retention and end-of-retention times, respectively.

In Fig. 3.15, we plot the flash memory endurance (PE cycles) versus the data retention time (T) for the two voltage design schemes, $T_{\text{write}} = 0$ and $T_{\text{write}} =$

FIGURE 3.15: Comparison of *flash memory endurance* between optimizing write voltage levels for early retention $T_{\text{write}} = 0$ (*solid curves*) and end of retention (EOR) $T_{\text{write}} = 12\text{-months}$ (*dotted curves*), keeping the FER fixed at 10^{-3} and 10^{-5} : employing LDPC 4K-code with 6-level Entropy quantization.

12-months, keeping the FER fixed at 10^{-3} and 10^{-5} . We observe that if the data retention time is relatively short, under 4 months, the write voltage scheme optimized for early retention time outperforms the EOR optimization scheme by providing longer flash memory endurance. Conversely, as the data retention time increases, the write voltage scheme optimized for EOR time yields better endurance performance. In summary, the problem of write voltage optimization for MLC NAND flash is a design trade-off between the achievable endurance performance and the expected retention time of stored flash data.

3.6 Chapter Summary

This chapter investigates the optimization of read and write voltage signals for 2-bit per cell NAND flash memory. Based on a non-stationary NAND flash channel model, which includes programming noise, RTN, CCI and data retention noise, the write voltage levels are optimized to minimize the channel raw error probability. The trade-off between write voltage optimization based on early retention time versus

the end-of-retention time is discussed. Since the error-rate of early retention time has enough margins from ECC decoding failures, while the end-of-retention state is more susceptible to raw memory errors, optimizing the error-rate of end-of-retention is more important. The proposed write voltage optimization scheme is validated to have more superior error-rate performance than the conventional channel-invariant write voltage scheme.

Besides, a novel read voltage quantization scheme based on the voltage entropy function is proposed to find the right balance between the flash channel symbol-error and symbol-erasure probabilities through controlled erasure decoding. This quantization scheme achieves better BP decoding performance than the prior-art quantization schemes. By joint read and write voltage signal optimization, the overall error-rate performance and decoding convergence are improved, as shown in Fig. 3.7, 3.8 and 3.14.

Chapter 4

Decision-Directed Retention-Failure Recovery with Flash Channel Update

The data retention noise, caused by the leakage of stored electrons through the cell's floating gate, is well-recognized as a dominant source of errors in the state-of-the-art NAND flash memory chips [75–77]. To overcome the retention noise induced errors, we propose a two-stage *retention-aware belief-propagation* (RABP) decoding scheme. In this scheme, we employ the BP decoder to recover the stored code-word bits. In case of decoding failure, we modify the input LLR computation by re-mapping the LLRs of erasure and data regions, based on the measured read-back voltage signal and the decoded code-word bits, and initialize a second round of BP decoding by using the modified LLR values. The proposed RABP decoder improves the flash channel tolerance against the data retention errors and enables the BP algorithm to successfully decode the flash data for longer period of retention time.

Next, we propose an *RABP assisted channel update* (RA-CU) algorithm for estimating the mean and standard deviation parameters of threshold voltage distribution by using the symbol error/erasure patterns measured at the output of RABP

decoder. Specifically, we formulate a cost function which defines the mean squared error (MSE) between the *measured* and *predicted* symbol error/erasure counts. This cost function is subsequently minimized by employing the gradient descent method to determine the best-fit mean and standard deviation values. In case of RABP decoding failure, wherein the measured symbol error/erasure values are unavailable, we compute the approximated values of symbol error/erasure counts with the help of *detected* symbol counts, and then invoke the RA-CU algorithm. More importantly, we perform the proposed retention-failure recovery and flash channel update by only using the initial (default) read threshold levels, thus avoiding the time and power consuming additional read operations.

The overall scheme works as follows. The flash memory controller first attempts to read the cell values with initial read threshold levels, and then initiates the conventional BP decoding. If the retention noise is weak and the decoding outcome is successful, no further processing is required. Otherwise, the controller attempts to recover the retention-failure data by using the proposed RABP decoding and then proceeds to the RA-CU algorithm.

4.1 Background

In Chapter 2, we designed the quantization levels over varying PE cycles based on the knowledge of cell's voltage distribution functions $p_{s_{11} \sim s_{01}}(v)$. Since these voltage distribution functions tend to follow Gaussian distribution, they can be described with their mean and standard deviation parameters. The optimized quantization levels are therefore stored into a look-up table against varying mean and standard deviation values, instead of memory PE cycles, and retrieved on-the-fly based on the given channel noise characteristics. For instance, during memory read operation, flash controller selects the stored quantization levels against the initial (zero retention age) mean and standard deviation values. In practice, however, the initial

distribution parameters tend to become outdated with increasing data retention time and the corresponding quantization levels, that are needed to compute the LLR information, tend to become sub-optimal. Consequently, the likely inaccurate LLR values may lead to ECC decoding failure and thus demand renewal/update of flash channel parameters. The prior-art works on retention-failure recovery and flash channel estimation require new memory sensing operations which are time and power consuming [38–41], nullifying the benefits brought by the flash technology. Hence, we propose the retention-failure recovery and flash channel estimation algorithms which precludes the additional read operations.

4.1.1 Retention Effect on Cell's Voltage Distribution

The incessant charge leakage through the cell's FGT over increasing data retention time reduces the threshold voltage level and ultimately shifts the memory cell down, into an adjacent (lower) cell state, resulting into an incorrect detection of stored cell value, as shown in Fig. 4.1. Note that the retention noise not only shifts the voltage distribution functions towards the erased state s_{11} , but also increases the distribution variance. While the retention noise affects the programmed state voltage distribution functions, it has inconsequential effect on the erased state memory cells [36]. Thus, in this chapter, we only attempt to find the mean and standard deviation parameters for the programmed state memory cells.

In Fig. 4.2, we simulate our 2-bit per cell NAND flash channel model, and plot the amount of change in mean and standard deviation values, denoted by $\Delta\mu_{s_{10} \sim s_{01}}$ and $\Delta\sigma_{s_{10} \sim s_{01}}$, respectively, over a range of data retention time, keeping the PE cycles fixed at 5K and 10K. It should be noted that the voltage distribution parameters correspond to the erased state s_{11} changes very slightly. Thus, we assume the values of $\Delta\mu_{s_{11}}$ and $\Delta\sigma_{s_{11}}$ to be negligible. We observe that the highest programmed state s_{01} shows the largest amount of shift, where the mean and standard deviation values change up to 0.4 V and 0.055 V, respectively, by the retention time of 10^5

FIGURE 4.1: Illustration of voltage distribution functions in 2-bit per cell NAND flash memory before (*top figure*) and after (*bottom figure*) data retention noise effect: voltage levels tend to drop gradually with data retention time.

FIGURE 4.2: The amount of change in mean $\Delta\mu$ (*top figure*) and standard deviation $\Delta\sigma$ (*bottom figure*) values with increasing data retention time at PE = 5K (*solid curves*) and PE = 10K (*dotted curves*).

hours. This is because the retention shift is proportional to the amount of charged electrons stored over the cell's floating-gate, thus a higher programmed state shifts faster than a lower programmed state.

4.1.2 Related Work

To mitigate the retention noise induced errors, a *flash correct-and-refresh* (FCR) method, wherein the cell values are periodically read and re-programmed before a large number of retention errors are accumulated, is reported in [77]. However, the FCR method only works when the flash controller is continuously powered-on. Furthermore, the cell re-programming operation may itself induce more cell-to-cell interference (CCI) which is yet another leading source of errors in today's flash memory chips. When the number of retention errors are large and exceed the error correcting capability of ECC decoder, the FCR scheme fails. In this situation, some other retention-failure recovery techniques, like the *error-prediction error-recovery* (EPER) [34] and *flash-defibrillator* (FD) [35], where the charge-reduced cells are recharged by applying recovery pulses, and the *retention-optimized-reading* (ROR) [36] and *read-channel-optimization* (RCO) [37], where the failed stored data is recovered by adaptively applying read threshold levels, can be employed. However, all these methods require extra memory sensing operations which may increase the read latency and total power consumption.

With regards to flash channel estimation, which involves finding the voltage distribution mean and standard deviation parameters corresponding to each cell state, the authors in [38] and [39] present *off-line* and *on-line* voltage modeling schemes (Off-VM) and (On-VM), respectively, which characterize the flash channel using empirical measurements and develop statistical models for threshold voltage distribution. The *adaptive read threshold* (ART) [41] is another method for characterizing the voltage distribution while using a limited number of re-reads. Apart from that, a *detector parameter estimation* (DPE) algorithm, which determines the best-fit voltage distribution parameters by matching the actual and modeled distributions functions, is reported in [40]. When ECC decoder of a memory page fails with initial distribution parameters, these estimation algorithms can be invoked by employing extra memory sensing operations. This is followed by more memory

sensing operations for the re-trial of error correction. This whole process results in significant read latency due to excessive memory sensing operations and large amount of memory-to-controller data transfer.

To preclude the excessive re-reads, a *decision-directed estimation* (DDE) strategy is presented in [78], wherein the decoded bit-error patterns are used to formulate an optimization problem for the adjustment of voltage distribution parameters. However, for the DDE scheme to work, the availability of correctly decoded data is a pre-requisite. Furthermore, the DDE scheme only finds the distribution mean values, whereas the distribution standard deviation values are assumed to be scaled factors of distribution mean values, which may not be true. Finally, the DDE scheme is also susceptible to large estimation errors and thus may sometimes become unstable.

Researchers have also proposed some innovative code construction methods which facilitate the memory controller in the selection of read threshold levels, particularly when no prior information of voltage distribution is available. One such example is the *balanced modulation* which ensures that the code-word contains an equal number of zeros and ones, hence facilitating the adjustment of dynamic read thresholds [79, 80]. The *rank modulation* [65], which stores the data contents in the relative voltage order over a set of memory cells, can also tolerate the charge leakage. Since the charge leakage process results in a unidirectional drift of cell voltages, it is likely that the relative order of cell voltages is still preserved after the retention period. However, rank modulation faces some technological issues which need to be addressed before its practical implementation. For instance, rank modulation requires a large number of read operations to learn the voltage level ordering, which may significantly increase the read latency.

4.2 Retention-Failure Recovery via RABP Decoding

When BP decoding fails due to a large number of retention errors, new memory sensing operations may be performed in order to update the voltage distribution parameters and to subsequently employ a new set of read threshold levels [38–41]. To preclude excessive re-reads, we propose the *retention-aware belief-propagation* (RABP) decoding which consists of 2 rounds of BP decoding. The first round provides a hard-decision estimate of the code-word bits, denoted by $\{\hat{a}_{\text{msb}}, \hat{a}_{\text{lsb}}\}$, corresponding to the MSB and LSB positions. In case of decoding failure, we will update the LLR information, denoted by $\hat{L}_{\text{msb}}$ and $\hat{L}_{\text{lsb}}$, by detecting the charge leakage effect, and execute a new (second) round of BP decoding.

In Fig. 4.3, we illustrate an example of 6-level non-uniform quantization scheme where threshold voltage range is split into 4 data regions ($\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3, \mathcal{D}_4$) and 3 erasure regions ($\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3$). The quantities $N_{\mathcal{D}_{i-1}}^{\mathcal{D}_i}$ and $N_{\mathcal{E}_{i-1}}^{\mathcal{D}_i}$, which we will refer to as the *retention-error* and *retention-erasure* counts, for $i \in \{2, 3, 4\}$, represent the number of memory cells that were originally programmed to data region $\mathcal{D}_i$ but drifted into data region $\mathcal{D}_{i-1}$ and erasure region $\mathcal{E}_{i-1}$, respectively, due to downshifting of cell's voltage as a result of charge leakage. Notably, with increasing retention time, the voltage distribution functions corresponding to cell state s_{10} , s_{00} and s_{01} will tend to drift into the erasure region $\mathcal{E}_1$, $\mathcal{E}_2$ and $\mathcal{E}_3$, respectively. Meanwhile, the voltage distribution functions corresponding to cell state s_{11} , s_{10} and s_{00} will drift away from the erasure region $\mathcal{E}_1$, $\mathcal{E}_2$ and $\mathcal{E}_3$, respectively. Equivalently, as the retention time increases, the values of $N_{\mathcal{E}_1}^{\mathcal{D}_2}$, $N_{\mathcal{E}_2}^{\mathcal{D}_3}$ and $N_{\mathcal{E}_3}^{\mathcal{D}_4}$ will tend to increase, whereas the values of $N_{\mathcal{E}_1}^{\mathcal{D}_1}$, $N_{\mathcal{E}_2}^{\mathcal{D}_2}$ and $N_{\mathcal{E}_3}^{\mathcal{D}_3}$ will tend to decrease. This observation is illustrated in Fig. 4.4, where we plot the *retention-erasure probability* as a function

of retention time, given by

$$p\left(N_{\mathcal{E}_{i-1}}^{\mathcal{D}_i}, T\right) = \frac{N_{\mathcal{E}_{i-1}}^{\mathcal{D}_i}}{N_{\mathcal{E}_{i-1}}^{\mathcal{D}_{i-1}} + N_{\mathcal{E}_{i-1}}^{\mathcal{D}_i}} \quad (4.1)$$

for $i \in \{2, 3, 4\}$, and

$$N_{\mathcal{E}_k}^{\mathcal{D}_j} = \frac{N_t}{4} \int_{v \in \mathcal{E}_k} p_{s_p}(v) dv \quad (4.2)$$

where $p_{s_p} \in \{p_{s_{11}}, p_{s_{10}}, p_{s_{00}}, p_{s_{01}}\}$ for $j \in \{1, 2, 3, 4\}$ as given in (2.9), whereas the value of N_t represents the total number of cells per memory-page. It can be seen that the erasure regions $\mathcal{E}_1$, $\mathcal{E}_2$ and $\mathcal{E}_3$ are dominated by memory cells belonging to cell state s_{10} , s_{00} and s_{01} , respectively. In order to recover from this retention-erasure effect, we modify the LLR value for a voltage in region n to be set to LLR value for voltage in region $n + 1$. To be precise, we compute new LLR information for all the erasure-bound memory cells where $v \in \{\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3\}$ such that, when $R_{n-1} < v \leq R_n$, for $n = 2, 4, 6$, their new LLR values become

$$\hat{L}_{\text{msb}}(n) = L_{\text{msb}}(n + 1) \quad (4.3)$$

$$\hat{L}_{\text{lsb}}(n) = L_{\text{lsb}}(n + 1) \quad (4.4)$$

This process is referred to as the *retention-erasure correction*. Furthermore, after a long retention time, it is also possible that the voltage distribution functions belonging to cell state s_{10} , s_{00} and s_{01} may shift past the lower erasure region and into the lower data region $\mathcal{D}_1$, $\mathcal{D}_2$ and $\mathcal{D}_3$, respectively. To see this effect, we plot the *retention-error probability* versus the retention time in Fig. 4.4, given by

$$p\left(N_{\mathcal{D}_{i-1}}^{\mathcal{D}_i}, T\right) = \frac{N_{\mathcal{D}_{i-1}}^{\mathcal{D}_i}}{N_{\mathcal{D}_{i-1}}^{\mathcal{D}_{i-1}} + N_{\mathcal{D}_{i-1}}^{\mathcal{D}_i}} \quad (4.5)$$

for $i \in \{2, 3, 4\}$. We can notice that the retention-error count, $N_{\mathcal{D}_{i-1}}^{\mathcal{D}_i}$, tends to increase with retention time. Thus, we also need to identify these charge-depleted

retention-error memory cells in order to modify their LLR information for the second round of BP decoding. However, in this situation, contrary to the retention-erasure correction, it is not possible to change the LLR information merely on the basis of read-back voltage signal. To this end, we leverage on the decoded code-word bits for the detection of such charge-loss memory cells. Specifically, if the cells have read-back voltage in the range $\mathcal{D}_1$, $\mathcal{D}_2$ or $\mathcal{D}_3$, but they are decoded into the next higher programmed state $\mathcal{D}_2$, $\mathcal{D}_3$, or $\mathcal{D}_4$, respectively, we mark them as the likely affected memory cells. For instance, when a memory cell is observed in data region $\mathcal{D}_1$ and the corresponding pair of bits is decoded to $\{\hat{a}_{\text{msb}}, \hat{a}_{\text{lsb}}\} = \{1, 0\}$, it will be marked as a likely charge-loss memory cell. Thus, for all such likely charge-loss memory cells, we modify their LLRs such that, when $R_{n-1} < v \leq R_n$ for $n = 1, 3, 5$, the new LLR values become

$$\hat{L}_{\text{msb}}(n) = L_{\text{msb}}(n + 2) \quad (4.6)$$

$$\hat{L}_{\text{lsb}}(n) = L_{\text{lsb}}(n + 2) \quad (4.7)$$

This step is referred to as the *retention-error correction*. Using the modified LLR information $\hat{L}_{\text{msb}}$ and $\hat{L}_{\text{lsb}}$, we execute a fresh second round of BP decoding. The detailed steps of RABP decoding scheme are outlined in Algorithm 1.

4.2.1 Retention-Error Correction

Since the proposed RABP decoder performs retention-error correction by using the partially corrected code-word bits (obtained after first round of decoding), it is important to understand the underlying mechanism which facilitates the proposed retention-error correction strategy. Here, it is worth noting that the number of corrected retention errors is much higher than the number of uncorrected or wrongly-corrected retention errors (called residual errors), so that a second round

FIGURE 4.3: Illustration of 6-level quantization for 2-bit per cell NAND flash memory: voltage range is split into 4 data regions ($\mathcal{D}_1$, $\mathcal{D}_2$, $\mathcal{D}_3$, $\mathcal{D}_4$, correspond to state s_{11} , s_{10} , s_{00} and s_{01} , respectively) and 3 erasure regions ($\mathcal{E}_1$, $\mathcal{E}_2$, $\mathcal{E}_3$).

FIGURE 4.4: Plot of retention-erasure probability $p(N_{\mathcal{E}_{i-1}}^{\mathcal{D}_i}, T)$ and retention-error probability $p(N_{\mathcal{D}_{i-1}}^{\mathcal{D}_i}, T)$ versus data retention time T at PE = 5K (solid curves) and PE = 10K (dotted curves).

of decoding will be able to correct the residual errors with adjusted LLR values (i.e. adjusted to take into account the retention error effect). To show that, in Fig. 4.5, we plot the average number of errors observed before and after the first round of decoding (per code-word failure). It can be seen that the number of errors is noticeably decreased after the first round of decoding. This helps the proposed

Algorithm 1 : Retention-Aware Belief-Propagation (RABP) Decoding.

```

1: Memory sensing to obtain read-back voltage ( $v$ )
2: Compute  $L_{\text{msb}}$  and  $L_{\text{lsb}}$  using (3.14) and (3.15)
3: Perform first round of BP decoding to obtain  $\{\hat{c}_{\text{msb}}, \hat{c}_{\text{lsb}}\}$ 
4: if successful decoding then
5:   Proceed to 14
6: end if
7: if ( $v \in \mathcal{E}_1$ ) OR ( $v \in \mathcal{E}_2$ ) OR ( $v \in \mathcal{E}_3$ ) then
8:   Compute  $\hat{L}_{\text{msb}}$  and  $\hat{L}_{\text{lsb}}$  using (4.3) and (4.4),  $\Rightarrow$  retention-erasure correction
9: end if
10: if [ $(v \in \mathcal{D}_1) \&\& \{\hat{a}_{\text{msb}}, \hat{a}_{\text{lsb}}\} = \{1, 0\}$ ] OR
    [ $(v \in \mathcal{D}_2) \&\& \{\hat{a}_{\text{msb}}, \hat{a}_{\text{lsb}}\} = \{0, 0\}$ ] OR
    [ $(v \in \mathcal{D}_3) \&\& \{\hat{a}_{\text{msb}}, \hat{a}_{\text{lsb}}\} = \{0, 1\}$ ] then
11:   Compute  $\hat{L}_{\text{msb}}$  and  $\hat{L}_{\text{lsb}}$  using (4.6) and (4.7),  $\Rightarrow$  retention-error correction
12: end if
13: Perform second round of BP decoding
14: End

```

RABP decoding to perform the retention-error correction.

Nevertheless, the existence of residual errors may sometimes lead to incorrect detection of retention-error memory cells. To observe the performance of the proposed retention-error detection scheme, we plot the probability of retention-error corrected (wrongly-corrected) cells as a function of retention time in Fig. 4.6, given by

$$P_{\text{corrected}} = \frac{\text{retention-error corrected cells}}{\text{Total (corrected+uncorrected) cells}} \quad (4.8)$$

$$P_{\text{uncorrected}} = \frac{\text{retention-error uncorrected (wrongly-corrected) cells}}{\text{Total (corrected+uncorrected) cells}} \quad (4.9)$$

We perform Monte-Carlo simulations to compute these probabilities via step 10 in Algorithm 1 by counting the number of retention-error corrected and wrongly corrected memory cells. In Fig. 4.6, we can notice that the number of corrected retention errors is much higher, around 80% at $T = 1,000$ hours, than the number of uncorrected retention errors, which validates the effectiveness of the proposed decision-directed retention-error correction strategy. This performance is achieved because the reliable input LLR values corresponding to the unaffected memory cells help to correct the unreliable LLR values corresponding to retention-affected

FIGURE 4.5: Plot of average errors observed *before* and *after* the first round of BP decoding versus data retention time at PE = 5K.

FIGURE 4.6: Fraction of retention-error *corrected/uncorrected* cells versus data retention time at PE = 5K.

memory cells during the BP decoding, thus producing fewer bit errors in the decoded bit stream and subsequently leads to high accuracy of retention-error correction.

4.2.2 Overall Error-Rate Performance of RABP Decoding

To analyze the overall error-rate improvement under the proposed RABP decoding scheme, employing both the retention-erasure and retention-error correction, we simulate the previously used (8000, 7200), rate-0.90 regular LDPC code with uniform column weight 5. This LDPC code is decoded using the shuffled BP algorithm [74], where maximum iteration count is set to 25 per BP decoding round.

In Fig. 4.7, we plot the frame-error-rate (FER) performance curves against the data retention time without performing channel update, employing the RABP

FIGURE 4.7: Frame-error-rate (FER) versus data retention time (without channel update) by employing the conventional BP (CBP) (*dotted curves*) and the proposed RABP decoding (*solid curves*) at PE = 3K, 4K and 5K.

decoder and compared with conventional BP decoder, keeping the PE cycles fixed at 3K, 4K and 5K, respectively. We can observe that the flash channel tolerance against the retention noise induced errors is significantly improved. For instance, considering FER at 10^{-4} , flash memory can endure up to 1,000 hours at PE = 5K using the conventional BP decoding without performing flash channel estimation. In contrast, for the same FER performance, the endurance limit is extended by up to 1,600 hours (60% improvement) under the proposed RABP decoding. Similarly, we can notice that the RABP decoding improves the FER performance by up to 3 *orders-of-magnitude* at retention time of 1,500 hours.

4.2.3 Average Complexity and Storage Requirement for RABP Decoding

In terms of overall complexity/latency, the proposed RABP decoding scheme needs to re-run the error correction decoding a second time for the recovery of failed code-word, which is also required by all the prior-art retention failure recovery

schemes [38–41, 78]. The only added complexity is the re-mapping of LLR information which incurs additional binary comparison operations against the decoded code-word bits $\{\hat{c}_{\text{msb}}, \hat{c}_{\text{lsb}}\}$, which can be upper bounded by $\mathcal{Z}$, given by

$$\mathcal{Z} = 2N \int_{v \in \{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3\}} \{p_{s_{11}}(v) + p_{s_{10}}(v) + p_{s_{00}}(v) + p_{s_{01}}(v)\} dv \quad (4.10)$$

Despite some increase in decoding complexity, the proposed scheme avoids the additional memory sensing operations for the recovery of retention failure data, which are otherwise required in case of all the prior-art schemes [38–41, 78]. In terms of storage requirement, the proposed RABP decoding scheme needs an additional $2N$ binary storage units to store the decoded code-word bits that are required to perform the retention-error correction.

4.3 RABP-Assisted Channel Update (RA-CU)

The failure of conventional BP decoding due to retention noise effect clearly indicates the need for flash channel estimation as the initial voltage distribution parameters, and the corresponding read threshold voltage levels, tend to become outdated due to the down-shifting of cell's threshold voltage levels. The flash channel re-estimation process updates the voltage distribution parameters in order to select the optimal read threshold levels, and hence facilitates more accurate computation of LLR information for BP decoding during the subsequent read requests. This reduces the future possibility of BP decoding failure. To perform flash channel re-estimation without additional memory read operations, we present the *RABP assisted channel update* (RA-CU) algorithm in which we employ the symbol error/erasure corrected patterns, which are obtained by comparing the input and output of the RABP decoder, to update the voltage distribution parameters. Specifically, using the measured retention-erasure and retention-error counts, denoted by $\tilde{N}_{\mathcal{E}_{i-1}}^{\mathcal{D}_i}$ and $\tilde{N}_{\mathcal{D}_{i-1}}^{\mathcal{D}_i}$,

respectively, we find the best-fit voltage distribution mean and standard deviation parameters. The measured retention-erasure and retention-error counts are obtained by looking at the channel output voltage values and then comparing the raw decoded bits to the final decoded bit values after the second round of RABP decoding. In Section 4.3.1, we will further explore the possibility of flash channel estimation under decoding failure scenario, wherein the measured retention-erasure and retention-error counts will be unavailable. Since this estimation method will not rely upon the corrected cell values, this algorithm will be invoked only when the retention noise induced errors exceed the error-correcting capability of RABP decoder.

To find the updated mean and standard deviation parameters, we define a cost function which computes the mean squared error (MSE) between the measured error/erasure counts $(\tilde{N}_{\mathcal{E}_{i-1}}^{\mathcal{D}_i}, \tilde{N}_{\mathcal{D}_{i-1}}^{\mathcal{D}_i})$ and the predicted error/erasure counts $(N_{\mathcal{E}_{i-1}}^{\mathcal{D}_i}, N_{\mathcal{D}_{i-1}}^{\mathcal{D}_i})$, for $i = \{2, 3, 4\}$, given as

$$\begin{aligned} \mathcal{C}(\boldsymbol{\mu}, \boldsymbol{\sigma}) = & \frac{1}{N_t^2} [w_{\mathcal{D}_1}^{\mathcal{D}_2} (\tilde{N}_{\mathcal{D}_1}^{\mathcal{D}_2} - N_{\mathcal{D}_1}^{\mathcal{D}_2})^2 + w_{\mathcal{E}_1}^{\mathcal{D}_2} (\tilde{N}_{\mathcal{E}_1}^{\mathcal{D}_2} - N_{\mathcal{E}_1}^{\mathcal{D}_2})^2 \\ & + w_{\mathcal{D}_2}^{\mathcal{D}_3} (\tilde{N}_{\mathcal{D}_2}^{\mathcal{D}_3} - N_{\mathcal{D}_2}^{\mathcal{D}_3})^2 + w_{\mathcal{E}_2}^{\mathcal{D}_3} (\tilde{N}_{\mathcal{E}_2}^{\mathcal{D}_3} - N_{\mathcal{E}_2}^{\mathcal{D}_3})^2 \\ & + w_{\mathcal{D}_3}^{\mathcal{D}_4} (\tilde{N}_{\mathcal{D}_3}^{\mathcal{D}_4} - N_{\mathcal{D}_3}^{\mathcal{D}_4})^2 + w_{\mathcal{E}_3}^{\mathcal{D}_4} (\tilde{N}_{\mathcal{E}_3}^{\mathcal{D}_4} - N_{\mathcal{E}_3}^{\mathcal{D}_4})^2] \end{aligned} \quad (4.11)$$

where $\boldsymbol{\mu} = [\mu_{s_{10}}, \mu_{s_{00}}, \mu_{s_{01}}]$ and $\boldsymbol{\sigma} = [\sigma_{s_{10}}, \sigma_{s_{00}}, \sigma_{s_{01}}]$ are the unknown distribution parameters, and $w_{\mathcal{D}_{i-1}}^{\mathcal{D}_i}$ and $w_{\mathcal{E}_{i-1}}^{\mathcal{D}_i}$ are the weighting coefficients for the error and erasure counts in the range of $[0, 1]$, given by

$$w_{\mathcal{D}_{i-1}}^{\mathcal{D}_i} = \frac{\tilde{N}_{\mathcal{D}_{i-1}}^{\mathcal{D}_i}}{\max \{ \tilde{N}_{\mathcal{D}_1}^{\mathcal{D}_2}, \tilde{N}_{\mathcal{E}_1}^{\mathcal{D}_2}, \tilde{N}_{\mathcal{D}_2}^{\mathcal{D}_3}, \tilde{N}_{\mathcal{E}_2}^{\mathcal{D}_3}, \tilde{N}_{\mathcal{D}_3}^{\mathcal{D}_4}, \tilde{N}_{\mathcal{E}_3}^{\mathcal{D}_4} \}} \quad (4.12)$$

$$w_{\mathcal{E}_{i-1}}^{\mathcal{D}_i} = \frac{\tilde{N}_{\mathcal{E}_{i-1}}^{\mathcal{D}_i}}{\max \{ \tilde{N}_{\mathcal{D}_1}^{\mathcal{D}_2}, \tilde{N}_{\mathcal{E}_1}^{\mathcal{D}_2}, \tilde{N}_{\mathcal{D}_2}^{\mathcal{D}_3}, \tilde{N}_{\mathcal{E}_2}^{\mathcal{D}_3}, \tilde{N}_{\mathcal{D}_3}^{\mathcal{D}_4}, \tilde{N}_{\mathcal{E}_3}^{\mathcal{D}_4} \}} \quad (4.13)$$

The process of weighting emphasizes the contribution of large error/erasure counts, giving them more weight in the optimization process. The expression for

predicted error/erasure counts ($N_{\mathcal{E}_{i-1}}^{\mathcal{D}_i}$, $N_{\mathcal{D}_{i-1}}^{\mathcal{D}_i}$) are obtained by computing the area under the Gaussian tail probability, as shown in Fig. 4.3, employing the Gaussian distribution based Q-functions¹, as follows

$$N_{\mathcal{D}_1}^{\mathcal{D}_2}(\mu_{s_{10}}, \sigma_{s_{10}}) = \frac{N_t}{4} \left[1 - Q\left(\frac{R_1 - \mu_{s_{10}}}{\sigma_{s_{10}}}\right) \right] \quad (4.14)$$

$$N_{\mathcal{E}_1}^{\mathcal{D}_2}(\mu_{s_{10}}, \sigma_{s_{10}}) = \frac{N_t}{4} \left[Q\left(\frac{R_1 - \mu_{s_{10}}}{\sigma_{s_{10}}}\right) - Q\left(\frac{R_2 - \mu_{s_{10}}}{\sigma_{s_{10}}}\right) \right] \quad (4.15)$$

$$N_{\mathcal{D}_2}^{\mathcal{D}_3}(\mu_{s_{00}}, \sigma_{s_{00}}) = \frac{N_t}{4} \left[Q\left(\frac{R_2 - \mu_{s_{00}}}{\sigma_{s_{00}}}\right) - Q\left(\frac{R_3 - \mu_{s_{00}}}{\sigma_{s_{00}}}\right) \right] \quad (4.16)$$

$$N_{\mathcal{E}_2}^{\mathcal{D}_3}(\mu_{s_{00}}, \sigma_{s_{00}}) = \frac{N_t}{4} \left[Q\left(\frac{R_3 - \mu_{s_{00}}}{\sigma_{s_{00}}}\right) - Q\left(\frac{R_4 - \mu_{s_{00}}}{\sigma_{s_{00}}}\right) \right] \quad (4.17)$$

$$N_{\mathcal{D}_3}^{\mathcal{D}_4}(\mu_{s_{01}}, \sigma_{s_{01}}) = \frac{N_t}{4} \left[Q\left(\frac{R_4 - \mu_{s_{01}}}{\sigma_{s_{01}}}\right) - Q\left(\frac{R_5 - \mu_{s_{01}}}{\sigma_{s_{01}}}\right) \right] \quad (4.18)$$

$$N_{\mathcal{E}_3}^{\mathcal{D}_4}(\mu_{s_{01}}, \sigma_{s_{01}}) = \frac{N_t}{4} \left[Q\left(\frac{R_5 - \mu_{s_{01}}}{\sigma_{s_{01}}}\right) - Q\left(\frac{R_6 - \mu_{s_{01}}}{\sigma_{s_{01}}}\right) \right] \quad (4.19)$$

Note that the values of predicted error/erasure counts $N_{\mathcal{D}_{i-1}}^{\mathcal{D}_i}$ and $N_{\mathcal{E}_{i-1}}^{\mathcal{D}_i}$ serve as the estimators of measured error/erasure counts $\tilde{N}_{\mathcal{D}_{i-1}}^{\mathcal{D}_i}$ and $\tilde{N}_{\mathcal{E}_{i-1}}^{\mathcal{D}_i}$, respectively. If the cost function is minimized, the estimated values of error/erasure counts will be close enough to the actual ones, thus providing the best-fit distribution parameters. Therefore, we minimize the cost function (4.11), given by

$$(\boldsymbol{\mu}^*, \boldsymbol{\sigma}^*) = \arg \min \{\mathcal{C}(\boldsymbol{\mu}, \boldsymbol{\sigma})\} \quad (4.20)$$

We plot this cost function against increasing data retention time in Fig. 4.8, keeping the PE cycles fixed at 5K and 10K and setting the data retention age to 10^3 , 5×10^3 , 10^4 and 2×10^4 , respectively. We can verify in Fig. 4.8 that the minimum points of cost function (4.11) matches with the actual data retention age set in the simulations. These minimum points also correspond to the desired

¹ $Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty \exp\left(-\frac{u^2}{2}\right) du.$

FIGURE 4.8: Illustration of cost function (4.11) plotted against data retention time at PE = 5K (*top figure*) and PE = 10K (*bottom figure*): minimum points correspond to actual retention age set in the simulations.

voltage distribution mean and standard deviation parameters (μ^*, σ^*) . To solve this convex optimization problem, we make use of the gradient descent (GD) method [71] that iteratively minimizes the cost function by using the following expression

$$(\mu^{k+1}, \sigma^{k+1}) = (\mu^k, \sigma^k) - \alpha \nabla \mathcal{C}(\mu^k, \sigma^k) \quad (4.21)$$

where $\nabla \mathcal{C}$ is the gradient of cost function $\mathcal{C}$, α is the iteration step size and k is the iteration index. In (4.21), the starting mean and standard deviation parameters (μ^0, σ^0) can be initialized with the previously known (initial) mean and standard deviation values that were optimal for the case of $T = 0$. The GD algorithm stops once when the value of cost function evaluated at $(\mu^{k+1}, \sigma^{k+1})$ is smaller than β , that is $\mathcal{C}(\mu^{k+1}, \sigma^{k+1}) \leq \beta$, where β is a predefined threshold level, otherwise the maximum number of iteration, denoted by I_{GD} , will be raised and the process will be re-run.

4.3.1 Channel Update after RABP Decoding Failure

In case of unsuccessful decoding attempts made by both the conventional and RABP decoders, caused by accumulation of large number of retention noise induced errors, the measured error/erasure counts which are essential to the RA-CU algorithm cannot be obtained. To perform channel estimation in decoding failure scenario, all the prior-art schemes, including [38–41, 78], employ additional memory sensing operations. To preclude these additional reads, we strive to obtain the approximated values of retention-erasure and retention-error counts, which we will denote as $\hat{N}_{\mathcal{E}_{i-1}}^{\mathcal{D}_i}$ and $\hat{N}_{\mathcal{D}_{i-1}}^{\mathcal{D}_i}$, respectively. These approximated counts will then facilitate the estimation of best-fit mean and standard deviation parameters. We derive these error/erasure count approximations with the help of detected cell counts, denoted by $Z_{\mathcal{D}_i}$ for $i \in \{1, 2, 3, 4\}$ and $Z_{\mathcal{E}_j}$ for $j \in \{1, 2, 3\}$, as shown in Fig. 4.3. The detected cell counts can be obtained by counting the total number of memory cells observed within each voltage quantization region. Referring to the retention-erasure probability in Fig. 4.4, we notice that with increasing data retention time, the erasure regions $\mathcal{E}_1$, $\mathcal{E}_2$ and $\mathcal{E}_3$ tend to include majority of cells corresponding to cell state s_{10} , s_{00} and s_{01} , respectively. This observation leads to the approximated retention-erasure count values, given by

$$\hat{N}_{\mathcal{E}_3}^{\mathcal{D}_4} \approx Z_{\mathcal{E}_3} \quad (4.22)$$

$$\hat{N}_{\mathcal{E}_2}^{\mathcal{D}_3} \approx Z_{\mathcal{E}_2} \quad (4.23)$$

$$\hat{N}_{\mathcal{E}_1}^{\mathcal{D}_2} \approx Z_{\mathcal{E}_1} \quad (4.24)$$

Next, to find the approximated retention-error count values $\hat{N}_{\mathcal{D}_{i-1}}^{\mathcal{D}_i}$, we assume that each voltage distribution function spans over three adjacent voltage regions. In particular, the distribution function $p_{s_{10}}(v)$ spans over $\mathcal{D}_1$, $\mathcal{E}_1$ and $\mathcal{D}_2$, $p_{s_{00}}(v)$ spans over $\mathcal{D}_2$, $\mathcal{E}_2$ and $\mathcal{D}_3$, and $p_{s_{01}}(v)$ spans over $\mathcal{D}_3$, $\mathcal{E}_3$ and $\mathcal{D}_4$, as illustrated in

Fig. 4.3. This seems to be a reasonable assumption, specifically for longer retention time, which can be easily substantiated by verifying the following approximation

$$\frac{\hat{N}_{\mathcal{D}_i}^{\mathcal{D}_i} + \hat{N}_{\mathcal{E}_{i-1}}^{\mathcal{D}_i} + \hat{N}_{\mathcal{D}_{i-1}}^{\mathcal{D}_i}}{\frac{N_t}{4}} \approx 1$$

where $i \in \{2, 3, 4\}$. In view of the above relation, we can now find the approximated retention-error counts, as follows

$$\begin{aligned} \hat{N}_{\mathcal{D}_3}^{\mathcal{D}_4} + \hat{N}_{\mathcal{E}_3}^{\mathcal{D}_4} + \hat{N}_{\mathcal{D}_4}^{\mathcal{D}_4} &\approx \frac{N_t}{4} \\ \hat{N}_{\mathcal{D}_3}^{\mathcal{D}_4} &\approx \frac{N_t}{4} - \hat{N}_{\mathcal{E}_3}^{\mathcal{D}_4} - \hat{N}_{\mathcal{D}_4}^{\mathcal{D}_4} \\ &\approx \frac{N_t}{4} - Z_{\mathcal{E}_3} - Z_{\mathcal{D}_4} \end{aligned} \quad (4.25)$$

$$\begin{aligned} \hat{N}_{\mathcal{D}_2}^{\mathcal{D}_3} + \hat{N}_{\mathcal{E}_2}^{\mathcal{D}_3} + \hat{N}_{\mathcal{D}_3}^{\mathcal{D}_3} &\approx \frac{N_t}{4} \\ \hat{N}_{\mathcal{D}_2}^{\mathcal{D}_3} &\approx \frac{N_t}{4} - \hat{N}_{\mathcal{E}_2}^{\mathcal{D}_3} - \hat{N}_{\mathcal{D}_3}^{\mathcal{D}_3} \\ &\approx \frac{N_t}{4} - Z_{\mathcal{E}_2} - (Z_{\mathcal{D}_3} - \hat{N}_{\mathcal{D}_3}^{\mathcal{D}_4}) \\ &\approx \frac{N_t}{2} - Z_{\mathcal{E}_2} - Z_{\mathcal{D}_3} - Z_{\mathcal{E}_3} - Z_{\mathcal{D}_4} \end{aligned} \quad (4.26)$$

$$\begin{aligned} \hat{N}_{\mathcal{D}_1}^{\mathcal{D}_2} + \hat{N}_{\mathcal{E}_1}^{\mathcal{D}_2} + \hat{N}_{\mathcal{D}_2}^{\mathcal{D}_2} &\approx \frac{N_t}{4} \\ \hat{N}_{\mathcal{D}_1}^{\mathcal{D}_2} &\approx \frac{N_t}{4} - \hat{N}_{\mathcal{E}_1}^{\mathcal{D}_2} - \hat{N}_{\mathcal{D}_2}^{\mathcal{D}_2} \\ &\approx \frac{N_t}{4} - Z_{\mathcal{E}_1} - (Z_{\mathcal{D}_2} - \hat{N}_{\mathcal{D}_2}^{\mathcal{D}_3}) \\ &\approx \frac{3N_t}{4} - Z_{\mathcal{E}_1} - Z_{\mathcal{D}_2} - Z_{\mathcal{E}_2} \\ &\quad - Z_{\mathcal{D}_3} - Z_{\mathcal{E}_3} - Z_{\mathcal{D}_4} \end{aligned} \quad (4.27)$$

The approximated error/erasure count values, given by (4.22) to (4.27), will be used in place of the measured error/erasure counts in the cost function (4.11) to find the best-fit mean and standard deviation parameters. The two channel update methods that are invoked after RABP decoding success and failure will be referred to as the retention-aware channel update after success (*RA-CUS*) and

FIGURE 4.9: Block diagram of the RABP assisted channel update algorithm invoked after decoding *success* (RA-CUS) and decoding *failure* (RA-CUF).

retention-aware channel update after failure (*RA-CUF*) algorithms, respectively, as shown in Fig. 4.9.

4.4 Performance Evaluation of RA-CU Algorithm

4.4.1 Average Estimation Error

To analyze the estimation accuracy under the proposed RA-CU algorithm, we compute the average standard error of the estimated voltage distribution mean and standard deviation parameters, given by

$$\varepsilon_{\mu_{s_{10} \sim s_{01}}} = |\mu_{s_{10} \sim s_{01}} - \mu_{s_{10} \sim s_{01}}^*| \quad (4.28)$$

$$\varepsilon_{\sigma_{s_{10} \sim s_{01}}} = |\sigma_{s_{10} \sim s_{01}} - \sigma_{s_{10} \sim s_{01}}^*| \quad (4.29)$$

In (4.28) and (4.29), the values of $\mu_{s_{10} \sim s_{01}}$ and $\sigma_{s_{10} \sim s_{01}}$ represent the actual mean and standard deviation parameters, whereas $\mu_{s_{10} \sim s_{01}}^*$ and $\sigma_{s_{10} \sim s_{01}}^*$ are the estimated ones. Fig. 4.10 shows the standard error for the estimated mean values, $\varepsilon_{\mu_{s_{10} \sim s_{01}}}$, over a range of data retention time, keeping the PE cycles fixed at 5K and 10K. These curves are obtained by employing both the RA-CUS and RA-CUF algorithms. It can be observed the total range of estimation errors fall in between 0.02 V to

FIGURE 4.10: Average error of the estimated mean (ε_μ) plotted against data retention time at PE = 5K (*top figure*) and PE = 10K (*bottom figure*) by employing the RA-CUS (*solid curves*) and RA-CUF (*dotted curves*) algorithms.

0.14 V. However, the estimation accuracy improves with increasing retention time. In particular, as the retention time exceeds beyond 10^4 hours, the estimation errors reduce to 0.02 V, and remain almost constant thereafter. This improving estimation accuracy substantiates the effectiveness of the proposed RA-CU algorithm. As the longer retention time induces a high raw memory BER, it therefore requires more accurate flash channel estimation which can be achieved by employing the RA-CU algorithm. Similar performance results are observed for the average error of the estimated standard deviation values, $\varepsilon_{\sigma_{s_{10} \sim s_{01}}}$, as shown in Fig. 4.11. The total estimation errors are found to be in between 0.02 V to 0.08 V, whereas they tend to decline over increasing data retention time.

4.4.2 Estimation Algorithm Complexity

We measure the overall computational complexity of the proposed RA-CU algorithm in terms of arithmetic operations performed by the GD algorithm per iteration, as

FIGURE 4.11: Average error of the estimated standard deviation (ε_σ) plotted against data retention time at PE = 5K (*top figure*) and PE = 10K (*bottom figure*): employing RA-CUS (*solid curves*) and RA-CUF (*dotted curves*) algorithms.

shown in Table 6.1. The term “LUT” refers to the look-up table implementation for the calculation of exponential terms and Q-functions. The arithmetic operation complexity includes the computations associated with expected error/erasure counts ($N_{\mathcal{D}_{i-1}}^{\mathcal{D}_i}, N_{\mathcal{E}_{i-1}}^{\mathcal{D}_i}$), partial derivatives, gradient of cost function ($\nabla \mathcal{C}$), and voltage distribution mean and standard deviation updates (μ^{k+1}, σ^{k+1}). Note that the proposed RA-CU algorithm not only avoids the additional memory reads but also offers a few times lower computational complexity per iteration compared to that of DPE algorithm [40]. For 6-level quantization, the DPE algorithm requires 153 ADD/SUB, 913 MUL/DIV and 144 LUT operations per iteration, whereas the proposed RABP-CU scheme requires only 34 ADD/SUB, 165 MUL/DIV and 33 LUT operations per iteration. In Fig. 4.12, we plot the average number of iterations required by the GD algorithm to achieve convergence, where the maximum number of GD iteration I_{GD} is set to 200 and the value of β is set to 20. We can observe the similar trends as that of estimation errors, where the average iteration count

TABLE 4.1: The number of arithmetic operations performed by the Gradient Descent (GD) based parameter estimation algorithm (per iteration).

S.No.		ADD/SUB	MUL/DIV	LUT
(a)	$\{N_{\mathcal{D}_{i-1}}^{\mathcal{D}_i}, N_{\mathcal{E}_{i-1}}^{\mathcal{D}_i}\}$	17	17	11
(b)	Derivatives	10	124	22
(c)	$\nabla \mathcal{C}$	12	12	0
(d)	$(\mu^{k+1}, \sigma^{k+1})$	6	6	0
	Total	34	165	33

FIGURE 4.12: Average number of GD iterations required to achieve convergence versus data retention time, plotted at PE = 5K and 10K.

reduces with increasing data retention time.

4.4.3 Error-Rate Performance

To simulate the error-rate performance, we employ the previously used rate-0.90, (8000,7200) regular LDPC code. In Fig. 4.13, we plot the FER curves over a range of data retention time while employing the RA-CUS based updated distribution parameters and compare with that of the non-updated (initial) distribution parameters. We assume that the distribution parameters are updated within the error-correcting capability of RABP decoder, specifically at $T_{update} = 70, 110$ and 600 hours, while setting the PE count at PE = 10K, 8K and 6K, respectively. We can notice that updating the distribution parameters yields significant improvement in flash channel's retention time limit. For instance, comparing at FER 10^{-4} , the RA-CUS algorithm can tolerate the data retention time by up to 1,000, 3,000 and 25,000 hours, while keeping the PE cycles fixed at 10K, 8K and 6K, respectively.

FIGURE 4.13: Frame-error-rate (FER) performance versus data retention time by employing the proposed RA-CUS algorithm for distribution parameter update: comparison between non-updated parameters and updated parameters at $T_{update} = 70, 110$ and 600 hours for $PE = 10K, 8K$ and $6K$, respectively.

This translates to an improvement in retention time by approximately $14\times$, $27\times$ and $41\times$ (times), respectively, compared to non-updated flash channel model. To verify the error-rate performance under the proposed RA-CUF algorithm, in Fig. 4.14, we plot the FER curves against data retention time, assuming that the distribution parameters are updated beyond the error-correcting capability of RABP decoder, in particular, at $T_{update} = 100, 200, 500, 1,000, 2,000, 5,000$ and $10,000$ hours, while keeping the PE cycles fixed at $10K$. Comparing at FER 10^{-4} , the proposed RA-CUF algorithm improves the retention time limit by approximately $18\times$, $28\times$, $50\times$, $78\times$, $121\times$, $185\times$ and $285\times$ (times) by supplying the updated voltage distribution mean and standard deviation parameters.

4.5 Chapter Summary

This chapter presents an innovative retention-aware belief-propagation (RABP) decoding strategy for LDPC codes to mitigate the data loss associated with retention noise in NAND flash memory. The proposed RABP decoding exploits the knowledge

FIGURE 4.14: Frame-error-rate (FER) performance versus data retention time plotted at PE = 10K by employing the proposed RA-CUF algorithm for distribution parameter update at different values of T_{update} , compared with non-updated distribution parameters.

that the memory cell's threshold voltage tends to drop gradually with data retention time. Under this effect, the probable charge-loss memory cells are detected by jointly analyzing their read-back voltage signal and the corresponding decoded code-word bits. In case of decoding failure, a new set of log-likelihood-ratio (LLR) values that enables retention-erasure correction or retention-error correction is computed. Then a fresh round of BP decoding is performed. Simulating a 2-bit per cell NAND flash channel model, it is shown that the retention time endurance of the proposed RABP decoding scheme is increased by up to 70% and the error-rate performance is improved by up to 3 *orders-of-magnitude*, when compared to conventional BP decoding.

In conjunction with RABP decoder, a NAND flash channel update algorithm (RA-CU) is proposed to re-estimate the voltage distribution mean and standard deviation parameters, leveraging on the decoded error/erasure patterns. The proposed RA-CU algorithm finds the best-fit voltage distribution parameters by employing the gradient descent (GD) method. The estimated mean and standard deviation values are shown to improve the channel estimation accuracy over increasing data

retention time, with estimation errors below 0.02 V. Meanwhile, the complexity of the proposed estimation algorithm also reduces with increasing data retention time. By virtue of the proposed RA-CU algorithm, the retention time tolerance is increased by up to $41\times$ (times) at 6K PE cycles. More importantly, this performance gain is achieved without performing additional memory sensing operations which are time and power consuming.

Chapter 5

Neighbor-A-Priori Detector for Cell-to-Cell Interference (CCI) Cancellation

Cell-to-cell interference (CCI), arising from parasitic coupling-capacitance between adjacent cells, is another major source of errors in today's flash memory chips. In this chapter, we present three novel post-processing detection schemes that exploit the *a-priori* information of neighboring/interfering cells to mitigate the CCI effect in 2-bit per cell NAND flash memory. The proposed schemes are referred to as the Even-*A-Priori* (Even-AP), the All-*A-Priori* (All-AP), and the All-AP coupling-capacitance-ratio (All-AP-CCR) detectors. The main idea is to remove the CCI component from the interfering cells before CCI cancellation from the victim cell. In this work, we also estimate the mean CCR values along victim cell's vertical and diagonal directions to enable more accurate CCI cancellation. Performance analysis and simulation results show significant improvement in the cell storage capacity and error-rate performance with the proposed detection schemes.

5.1 Background

As the flash memory cells are scaled-down and integrated closer to each other, several reliability issues have emerged which compromise the overall performance of flash memory system [60, 61]. With cell shrinking, there are now fewer electrons that can be stored over the transistor's floating gate, resulting in narrower gap between adjacent voltage levels. Besides, for m -bit per cell flash technology, it requires 2^m voltage levels to distinctly represent the m -bit data symbol. Adjusting these voltage levels, in an already contracted voltage window, further reduces the noise margin and consequently degrades the system error performance. The other problem that stems from node-scaling is the increase in parasitic coupling-capacitance between the adjacent memory cells. This circuit-level impairment introduces the cell-to-cell interference (CCI) inside the flash cell array, which is well recognized as one of the major source of cell's threshold voltage distortion [16, 17]. It leads to data reliability problems and complicates the design of channel detector and encoder/decoder units inside the flash memory controller.

In the previous chapter, we addressed the data retention problem caused by cell's charge leakage, while we employed the signal pre-compensation method [42] to handle the CCI induced errors in NAND flash channel. However, the pre-compensation technique has certain limitations as it can only eliminate CCI from the programmed state cells but can not remove CCI from the erased state cells. In this chapter, we address the data reliability issues associated with CCI and present three post-processing channel detection schemes for its mitigation. The proposed detection schemes are used with the even-odd bit-line architecture to fully exploit their potential. The effectiveness of the proposed schemes and the corresponding design trade-offs are discussed in detail in the following sections.

5.1.1 CCI Statistical Model

The emergence of circuit-level parasitic coupling-capacitance between adjacent memory cells may affect the threshold voltage of a memory cell and lead to severe voltage signal distortions [16, 17]. In particular, the threshold voltage signal of a victim cell may be affected due to the programming of its neighboring memory cells through the coupling-capacitance induced CCI. According to [15], the victim cell's voltage-shift due to the CCI can be written as

$$I = \sum_k \Delta V_k \gamma_k \quad (5.1)$$

where I is the CCI component as given in (2.1), ΔV_k is the change in the threshold voltage of interfering cell k which is configured after the victim cell, and γ_k is the coupling-capacitive ratio (CCR) between the interfering and the victim cells. In flash memory array, only the immediate neighboring cells affect the victim cell through CCI, whereas the influence of distant neighboring cells is generally assumed to be negligible [42]. For the even-odd bit-line architecture, there are 3 and 5 interfering cells for the odd and even bit-line cells, respectively, as shown in Fig. 5.1. Here, we observe that the odd bit-line cells may influence the threshold voltage of even bit-line cells, whereas the programming of even bit-line cells has no effect on the threshold voltage of odd bit-line cells. This observation will be used on-wards for the design of the proposed neighbor-apriori detection schemes. Using Fig. 5.1, we can write the equivalent channel matrix (Γ) for the odd and even bit-line cells, given as

$$\Gamma^{\text{odd}} = \begin{bmatrix} \gamma_{xy} & \gamma_y & \gamma_{xy} \\ 0 & 1 & 0 \end{bmatrix}$$

and

$$\Gamma^{\text{even}} = \begin{bmatrix} \gamma_{xy} & \gamma_y & \gamma_{xy} \\ \gamma_x & 1 & \gamma_x \end{bmatrix}$$

where γ_x , γ_y and γ_{xy} are the CCRs along victim cell's horizontal, vertical and diagonal directions, respectively. We can now expand the CCI expression (5.1) into more comprehensive form for the odd and even bit-line cells, as follows

$$I_{(i,j)}^{\text{odd}} = \gamma_{xy} [V_{(i+1,j-1)}^{\text{new}} - V_{(i+1,j-1)}^{\text{old}}] + \gamma_y [V_{(i+1,j)}^{\text{new}} - V_{(i+1,j)}^{\text{old}}] + \gamma_{xy} [V_{(i+1,j+1)}^{\text{new}} - V_{(i+1,j+1)}^{\text{old}}] \quad (5.2)$$

$$I_{(i,j)}^{\text{even}} = \gamma_x [V_{(i,j-1)}^{\text{new}} - V_{(i,j-1)}^{\text{old}}] + \gamma_x [V_{(i,j+1)}^{\text{new}} - V_{(i,j+1)}^{\text{old}}] + \gamma_{xy} [V_{(i+1,j-1)}^{\text{new}} - V_{(i+1,j-1)}^{\text{old}}] + \gamma_y [V_{(i+1,j)}^{\text{new}} - V_{(i+1,j)}^{\text{old}}] + \gamma_{xy} [V_{(i+1,j+1)}^{\text{new}} - V_{(i+1,j+1)}^{\text{old}}] \quad (5.3)$$

where subscript (i, j) denotes the cell's position connecting with i^{th} word-line and j^{th} bit-line terminals, as shown in Fig. 5.1. The random variables V^{old} and V^{new} denote the cell's threshold voltage signals before and after the cell programming, respectively (i.e. representing voltage shift $\Delta V = [V^{\text{new}} - V^{\text{old}}]$). Note that the random variable V^{old} represents the threshold voltage of erased state memory cells as all the memory cells are initially in the erased state. Thus, it can be modeled by using Gaussian distribution function, given as

$$p_{V^{\text{old}}}(v) = \frac{1}{\sigma_e \sqrt{2\pi}} e^{-\frac{(v - V_{s11})^2}{2\sigma_e^2}} \quad (5.4)$$

Referring to (2.1), the cell's initial threshold voltage after programming operation can be defined by using the desired write voltage level V_p and the programming noise components n_u and n_p , i.e. $V^{\text{new}} = V_p + n_u + n_p$, with distribution functions

$$p_{n_u}(v) = \begin{cases} \frac{1}{\Delta V_{pp}}, & \text{if } 0 \leq v \leq \Delta V_{pp} \\ 0, & \text{otherwise} \end{cases} \quad (5.5)$$

and

$$p_{n_p}(v) = \frac{1}{\sigma_p \sqrt{2\pi}} e^{-\frac{v^2}{2\sigma_p^2}} \quad (5.6)$$

for $V_p \in \{V_{s10}, V_{s00}, V_{s01}\}$. According to [42], the CCI coupling-capacitive ratios can be modeled as bounded Gaussian random variables. The distribution function for horizontal CCR γ_x is given by

$$p_{\gamma_x}(v) = \begin{cases} \frac{c_\gamma}{\sigma_{\gamma_x} \sqrt{2\pi}} e^{-\frac{(v-\mu_{\gamma_x})^2}{2\sigma_{\gamma_x}^2}}, & \text{if } |v - \mu_{\gamma_x}| \leq 0.2\mu_{\gamma_x} \\ 0, & \text{otherwise} \end{cases} \quad (5.7)$$

where c_γ is the normalization factor, and μ_{γ_x} and σ_{γ_x} are constant mean and standard deviation parameters. Similarly, distribution function for vertical/diagonal CCR $\gamma_{y|xy}$ is given by

$$p_{\gamma_{y|xy}}(v) = \begin{cases} \frac{c_\gamma}{\sigma_{\gamma_{y|xy}} \sqrt{2\pi}} e^{-\frac{(v-\mu_{\gamma_{y|xy}}^*)^2}{2\sigma_{\gamma_{y|xy}}^2}}, & \text{if } |v - \mu_{\gamma_{y|xy}}^*| \leq 0.2\mu_{\gamma_{y|xy}}^* \\ 0, & \text{otherwise} \end{cases} \quad (5.8)$$

Note that, due to the word-line pitch variations, the mean values for γ_y and γ_{xy} , denoted by $\mu_{\gamma_y}^*$ and $\mu_{\gamma_{xy}}^*$, are also assumed to be bounded Gaussian random variables [42], given by

$$p_{\mu_{\gamma_{y|xy}}^*}(v) = \begin{cases} \frac{c_t}{0.2\mu_{\gamma_{y|xy}} \sqrt{2\pi}} e^{-\frac{(v-\mu_{\gamma_{y|xy}})^2}{2(0.2\mu_{\gamma_{y|xy}})^2}}, & \text{if } |v - \mu_{\gamma_{y|xy}}| \leq 0.2\mu_{\gamma_{y|xy}} \\ 0, & \text{otherwise} \end{cases} \quad (5.9)$$

In this work, the CCR distribution parameters are set according to [42], as follows:

$\sigma_{\gamma_x} = 0.3\mu_{\gamma_x}$, $\sigma_{\gamma_{y|xy}} = 0.3\mu_{\gamma_{y|xy}}^*$, $\mu_{\gamma_x} = 0.1s$, $\mu_{\gamma_y} = 0.08s$ and $\mu_{\gamma_{xy}} = 0.006s$, where factor s is used to vary the CCI strength.

5.1.2 CCI Effect on Cell's Voltage Distribution

To visualize the influence of CCI on cell's threshold voltage signal, in Fig. 5.2, we illustrate an example of how threshold voltage distribution functions are affected

FIGURE 5.1: Illustration of cell-to-cell interference (CCI) in the *even-odd* bit-line architecture: *even* and *odd* bit-line cells are interfered by 5 and 3 neighboring cells, respectively.

due to the effect of CCI. We can notice that the CCI shifts the voltage distribution functions upward into the next higher programmed state. Note that the amount of voltage shift is dissimilar for even and odd bit-line cells. This is due to the fact that there are 5 interfering cells for the even bit-line cells, whereas there are only 3 interfering cells for the odd bit-line cells, as shown in Fig. 5.1. Apart from voltage shift, the voltage distribution functions also tend to become wider due to CCI, thus narrowing the gap between the adjacent voltage levels and subsequently reducing the channel noise margins. Hence, an effective system-level solution is inevitably required to tolerate the CCI induced errors in NAND flash memory.

5.1.3 Related Work

To mitigate the CCI effect, there have been several works reported in the open literature. These prior-art works can be generally placed into two separate categories, namely the *pre-processing* and *post-processing* schemes, as shown in Fig. 5.3. The pre-processing schemes are applied before/during the cell programming, whereas the post-processing schemes are employed after the memory sensing operation. The contributions in [42] and [81] are two quite similar pre-processing techniques in which the CCI strength is estimated at the time of cell programming and

FIGURE 5.2: Illustration of voltage distribution functions for 2-bit per cell NAND flash memory: before CCI (*top figure*) and after CCI (*middle and bottom figures*) effect.

subsequently subtracted from the desired write/verify voltage levels. However, these schemes can not remove the CCI from the erased state memory cells. Besides, there are some *constrained coding* methods, such as reported in [82–85], wherein certain voltage levels of neighboring cells are forbidden such that a wider gap between adjacent voltage distributions is ensured. The constrained codes reduce the effect of CCI, however, they incur additional coding overhead. Furthermore, to minimize the CCI effect, a programming approach is reported in [47], where the memory cells are temporarily programmed into intermediate voltage levels. Recently, some dirty paper coding methods are reported in the open literature that attack the problem of CCI by using coding methods designed for stuck-at defect channels [86, 87]. By changing the flash memory channel with the CCI into the flash memory channel with defective cells with the help of one pre-read operation, the encoder obtains the side information of defects. Based on the available side information, the additive encoding method improves the probability of decoding failure.

An extensive effort has also been devoted towards the design of CCI post

FIGURE 5.3: Block diagram of the signal pre-processing and post-processing techniques for CCI mitigation in NAND flash memory.

processing (detection) schemes. In [81, 88, 89], the channel detector is designed by adaptively applying the read threshold voltage levels. Another channel detection scheme is presented by Dong's *et al.* in [28], where a closed-form expression for cell's threshold voltage distribution is obtained to compute the LLR information for soft LDPC decoding. However, all these post-processing schemes overlook to remove the CCI component from the read-back voltage signal, which subsequently result in sub-optimal error-rate performance. More effective post-processing schemes are presented in [42–44] and [45], wherein the detrimental CCI component is removed from the read-back voltage signal. Likewise, there are some trellis-based detection schemes, such as reported in [90] and [91], for the optimal detection and mitigation of CCI component. However, these schemes are more computationally extensive as they require the calculation of branch-metric values for the entire trellis structure in order to produce the soft-input information for the stored code-word. Therefore, the trellis-based detection schemes, though optimal in terms of error performance, may not be best suited for the time and power sensitive NAND flash memory.

5.2 Neighbor-*A-Priori* Detector

We present three post-processing CCI cancellation schemes based on the a-priori information of the interfering memory cells, while employing the even-odd bit-line architecture. The proposed work is inspired by the prior-art *Post-Comp* detector

[42] which computes the CCI estimate by measuring the threshold voltage of the interfering memory cells through uniform memory sensing, followed by the removal of the estimated CCI component from the read-back voltage of the victim cell. In the proposed work, we further improve on the Post-Comp detector by more accurately estimating the individual factors that constitute the overall CCI component. Based on that, we obtain a more reliable estimate of the CCI signal and subsequently remove it from the victim cell's threshold voltage signal.

5.2.1 The Prior-art Post-Compensation Detector

The Post-Comp detection scheme reported in [42] is a simple yet effective technique for the mitigation of CCI in NAND flash memory. It is a symbol-based detector in which the CCI cancellation is performed on cell-by-cell basis. The basic idea is to estimate the CCI strength, denoted by $\hat{I}_{(i,j)}$, by using the threshold voltage of neighboring memory cells and then remove this quantity from victim cell's read-back threshold voltage signal $V_{(i,j)}$, as follows

$$\vec{V}_{(i,j)} = V_{(i,j)} - \hat{I}_{(i,j)} \quad (5.10)$$

where $\vec{V}_{(i,j)}$ represents the cleaned (CCI removed) voltage signal. According to [42], the CCI strength for the odd and even bit-line cells can be estimated as

$$\begin{aligned} \hat{I}_{(i,j)}^{\text{odd}} = & \mu_{\gamma_{xy}} [V_{(i+1,j-1)} - V_{s11}] + \mu_{\gamma_y} [V_{(i+1,j)} - V_{s11}] \\ & + \mu_{\gamma_{xy}} [V_{(i+1,j+1)} - V_{s11}] \end{aligned} \quad (5.11)$$

$$\begin{aligned} \hat{I}_{(i,j)}^{\text{even}} = & \mu_{\gamma_x} [V_{(i,j-1)} - V_{s11}] + \mu_{\gamma_x} [V_{(i,j+1)} - V_{s11}] \\ & + \mu_{\gamma_{xy}} [V_{(i+1,j-1)} - V_{s11}] + \mu_{\gamma_y} [V_{(i+1,j)} - V_{s11}] \\ & + \mu_{\gamma_{xy}} [V_{(i+1,j+1)} - V_{s11}] \end{aligned} \quad (5.12)$$

By comparing the CCI estimates (5.11) and (5.12) with actual CCI expressions (5.2) and (5.3), respectively, we can observe that the random variables V^{new} (initial programmed voltage level) are replaced with V (read-back voltage). Similarly, the random variables V^{old} (initial erased-state voltage level) and CCR values $\gamma_{x|y|xy}$ are replaced with their respective mean values V_{s11} and $\mu_{x|y|xy}$, respectively.

We make two observations from the Post-Comp detector. First, the read-back voltage V is not an accurate estimate of the initial threshold voltage V^{new} . This is because the read-back voltage signal includes other noise factors as given in (2.1). Second, the mean values μ_{γ_y} and $\mu_{\gamma_{xy}}$ for CCRs along the vertical and diagonal directions are not constant terms, but represent random variables. In the proposed neighbor-a-priori detection schemes, we will address these two issues and replace the read-back voltage and the mean CCR values with more accurate estimates.

5.2.2 The Proposed Even-*A-Priori* (Even-AP) Detector

In the Even-*A-Priori* (Even-AP) detection scheme, since the even bit-line cells are influenced by the odd bit-line cells, the odd bit-line cells will be cleaned first by using (5.10) and their a-priori information will be used for the CCI compensation of even bit-line cells. Thus, in the Even-AP scheme, CCI compensation for the odd bit-line cells will remain the same as given in (5.11). Using the cleaned odd bit-line cells with voltage level $\vec{V}$, we compute the corresponding a-priori information, denoted by $V^{\mathcal{AP}}$, given by

$$V_{(i,j)}^{\mathcal{AP}} = \begin{cases} V_{s11}, & \text{if } \mathcal{J} = 1 \\ V_{s10} + \frac{\Delta V_{pp}}{2}, & \text{if } \mathcal{J} = 2 \\ V_{s00} + \frac{\Delta V_{pp}}{2}, & \text{if } \mathcal{J} = 3 \\ V_{s01} + \frac{\Delta V_{pp}}{2}, & \text{if } \mathcal{J} = 4 \end{cases} \quad (5.13)$$

where $\mathcal{J}$ is the index set, given by

$$\mathcal{J} = \odot \left\{ \min \left\{ \left| \vec{V}_{(i,j)} - V_{s11} \right|, \left| \vec{V}_{(i,j)} - \left(V_{s10} + \frac{\Delta V_{pp}}{2} \right) \right|, \right. \right. \\ \left. \left. \left| \vec{V}_{(i,j)} - \left(V_{s00} + \frac{\Delta V_{pp}}{2} \right) \right|, \left| \vec{V}_{(i,j)} - \left(V_{s01} + \frac{\Delta V_{pp}}{2} \right) \right| \right\} \right\} \quad (5.14)$$

where $\odot\{\cdot\}$ denote the index value of the set element. The index set $\mathcal{J}$ finds the minimum Euclidean distance between the CCI removed voltage signal $\vec{V}$ and the four possible initial voltage levels, given by $\{V_{s11}, V_{s10} + \frac{\Delta V_{pp}}{2}, V_{s00} + \frac{\Delta V_{pp}}{2}, V_{s01} + \frac{\Delta V_{pp}}{2}\}$, and returns the corresponding index set value. Using this index set value, the a-priori information determines the cell's expected initial threshold voltage level. Using the a-priori information (5.13), we estimate the CCI component for the even bit-line cells as

$$\begin{aligned} \bar{I}_{(i,j)}^{\text{even}} &= \mu_{\gamma_x} [V_{(i,j-1)}^{\mathcal{AP}} - V_{s11}] + \mu_{\gamma_x} [V_{(i,j+1)}^{\mathcal{AP}} - V_{s11}] \\ &\quad + \mu_{\gamma_{xy}} [V_{(i+1,j-1)} - V_{s11}] + \mu_{\gamma_y} [V_{(i+1,j)} - V_{s11}] \\ &\quad + \mu_{\gamma_{xy}} [V_{(i+1,j+1)} - V_{s11}] \end{aligned} \quad (5.15)$$

By comparing (5.12) with (5.15), we notice that the read-back voltage terms along the horizontal direction $V_{(i,j-1)}$ and $V_{(i,j+1)}$ are replaced with their a-priori information $V_{(i,j-1)}^{\mathcal{AP}}$ and $V_{(i,j+1)}^{\mathcal{AP}}$, respectively. In Fig. 5.4 (a), we illustrate the pictorial representation of the proposed Even-AP detection scheme. The detailed steps of the Even-AP scheme are outlined in Algorithm 2.

Algorithm 2 : Even-*A-Priori* (Even-AP) Detector

- 1: **for** every *odd* bit-line cell on word-line (i) **do**
 - 2: Perform CCI cancellation using (5.11)
 - 3: Compute a-priori-information ($V^{\mathcal{AP}}$) using (5.13) and (5.14)
 - 4: **end for**
 - 5: **for** every *even* bit-line cell on word-line (i) **do**
 - 6: Perform CCI cancellation using (5.15)
 - 7: **end for**
-

FIGURE 5.4: Illustration of the proposed *Even-AP*, *All-AP* and *All-AP-CCR* detection schemes: (a) cleaned odd bit-line cells are used for the compensation of even bit-line cells (b) cleaned next word-line ($i + 1$) cells are used for the compensation of both odd and even bit-line cells (c) All-AP detection combined with coupling-capacitance-ratio (CCR) estimation algorithm.

5.2.3 The Proposed All-*A-Priori* (All-AP) Detector

We further observe that there exist 3 more uncleaned aggressor cells, connected with the next word-line ($i + 1$), interfering with both even and odd bit-line cells. Therefore, we extend the same technique and estimate the a-priori information for the remaining 3 aggressor cells. This scheme will be referred to as the All-*A-Priori* (All-AP) detection scheme. Thus, in All-AP detector, the sequence of CCI cancellation will be such that the odd bit-line cells on word-line ($i + 1$) will be cleaned

first, followed by the even bit-line cells on word-line $(i + 1)$, followed by the odd bit-line cells on word-line i , followed by the even bit-line cells on word-line i . We can write the estimated CCI expression for the odd and even bit-line cells as

$$\begin{aligned} \tilde{I}_{(i,j)}^{\text{odd}} = & \mu_{\gamma_{xy}} [V_{(i+1,j-1)}^{\mathcal{AP}} - V_{s11}] + \mu_{\gamma_y} [V_{(i+1,j)}^{\mathcal{AP}} - V_{s11}] \\ & + \mu_{\gamma_{xy}} [V_{(i+1,j+1)}^{\mathcal{AP}} - V_{s11}] \end{aligned} \quad (5.16)$$

$$\begin{aligned} \tilde{I}_{(i,j)}^{\text{even}} = & \mu_{\gamma_x} [V_{(i,j-1)}^{\mathcal{AP}} - V_{s11}] + \mu_{\gamma_x} [V_{(i,j+1)}^{\mathcal{AP}} - V_{s11}] \\ & + \mu_{\gamma_{xy}} [V_{(i+1,j-1)}^{\mathcal{AP}} - V_{s11}] + \mu_{\gamma_y} [V_{(i+1,j)}^{\mathcal{AP}} - V_{s11}] \\ & + \mu_{\gamma_{xy}} [V_{(i+1,j+1)}^{\mathcal{AP}} - V_{s11}] \end{aligned} \quad (5.17)$$

In Fig. 5.4 (b), we show the diagram and the sequence of CCI cancellation steps for the All-AP detection scheme. The corresponding pseudo-code is given in Algorithm 3. Note that the All-AP detection scheme performs CCI cancellation for the next word-line memory cells, which may cause some increase in the computational complexity. In the subsequent section, we will present a detailed complexity analysis for different detection schemes. Nevertheless, we will show in the next section that the All-AP detector outperforms the Even-AP detection scheme in terms of error-rate performance.

5.2.4 The Proposed All-*A-Priori* CCR (All-AP-CCR) Detector

Building upon the premise of the All-AP detection scheme, where all the interfering cells are cleaned, we will further address the estimation of mean coupling-capacitance ratio (CCR) values and incorporate them into the CCI estimates. Specifically, we will find the optimal mean CCR values along victim cell's vertical (μ_{γ_y}) and diagonal ($\mu_{\gamma_{xy}}$) directions, as these two quantities are random variables with distribution

Algorithm 3 : All-*A-Priori* (All-AP) Detector

```

1: for every odd bit-line cell on word-line  $(i + 1)$  do
2:   Perform CCI cancellation using (5.11)
3:   Compute a-priori-information ( $V^{\mathcal{AP}}$ ) using (5.13) and (5.14)
4: end for
5: for every even bit-line cell on word-line  $(i + 1)$  do
6:   Perform CCI cancellation using (5.15)
7:   Compute a-priori-information ( $V^{\mathcal{AP}}$ ) using (5.13) and (5.14)
8: end for
9: for every odd bit-line cell on word-line  $(i)$  do
10:  Perform CCI cancellation using (5.16)
11:  Compute a-priori-information ( $V^{\mathcal{AP}}$ ) using (5.13) and (5.14)
12: end for
13: for every even bit-line cell on word-line  $(i)$  do
14:  Perform CCI cancellation using (5.17)
15: end for

```

function given in (5.9). This new scheme, which comprises of All-AP detection along with mean CCR estimation, will be referred to as the All-*A-Priori*-CCR (All-AP-CCR) detector. Its block diagram is shown in Fig. 5.4 (c). In this figure, the CCR estimation module is introduced before the CCI cancellation step of odd bit-line cells, which supplies the optimized mean CCR values in vertical ($\mu_{\gamma_y}^*$) and diagonal ($\mu_{\gamma_{xy}}^*$) directions. Since the CCI effect is stationary w.r.t. flash memory wear-out, the CCR estimation algorithm can be invoked only once in the offline mode and the estimated parameters can be stored into a look-up table to avoid any run-time computations. With the optimized mean CCR values, the CCI estimates, given by (5.16) and (5.17), can be modified as follows

$$\begin{aligned} \tilde{I}_{(i,j)}^{\text{odd}*} &= \mu_{\gamma_{xy}}^* [V_{(i+1,j-1)}^{\mathcal{AP}} - V_{s11}] + \mu_{\gamma_y}^* [V_{(i+1,j)}^{\mathcal{AP}} - V_{s11}] \\ &\quad + \mu_{\gamma_{xy}}^* [V_{(i+1,j+1)}^{\mathcal{AP}} - V_{s11}] \end{aligned} \quad (5.18)$$

$$\begin{aligned} \tilde{I}_{(i,j)}^{\text{even}*} &= \mu_{\gamma_x} [V_{(i,j-1)}^{\mathcal{AP}} - V_{s11}] + \mu_{\gamma_x} [V_{(i,j+1)}^{\mathcal{AP}} - V_{s11}] \\ &\quad + \mu_{\gamma_{xy}}^* [V_{(i+1,j-1)}^{\mathcal{AP}} - V_{s11}] + \mu_{\gamma_y}^* [V_{(i+1,j)}^{\mathcal{AP}} - V_{s11}] \\ &\quad + \mu_{\gamma_{xy}}^* [V_{(i+1,j+1)}^{\mathcal{AP}} - V_{s11}] \end{aligned} \quad (5.19)$$

5.2.4.1 Estimation of Mean CCR Values

To determine the optimized mean CCR values, we write a cost function $\mathcal{F}(\mu_{\gamma_y}, \mu_{\gamma_{xy}})$ which defines the mean squared error (MSE) between the measured and modeled voltage-shift expressions, denoted by $\Delta V_{(i,j)}^M$ and $\Delta V_{(i,j)}^E$, respectively, realized due to the CCI effect. This cost function can be mathematically written as

$$\mathcal{F}(\mu_{\gamma_y}, \mu_{\gamma_{xy}}) = \frac{1}{N_t} \sum_{j=1}^{N_t} (\Delta V_{(i,j)}^M - \Delta V_{(i,j)}^E)^2 \quad (5.20)$$

where N_t represents the number of memory cells per memory-page. The desired mean CCRs can then be obtained by minimizing this cost function, as

$$(\mu_{\gamma_y}^*, \mu_{\gamma_{xy}}^*) = \arg \min \{ \mathcal{F}(\mu_{\gamma_y}, \mu_{\gamma_{xy}}) \} \quad (5.21)$$

The amount of measured voltage-shift at cell's position (i, j) can be determined by finding the minimum Euclidean distance between the read-back voltage $V_{(i,j)}$ and cell's initial mean voltage, given by

$$\Delta V_{(i,j)}^M = \min \left\{ \left| V_{(i,j)} - V_{s11} \right|, \left| V_{(i,j)} - \left(V_{s10} + \frac{\Delta V_{pp}}{2} \right) \right|, \left| V_{(i,j)} - \left(V_{s00} + \frac{\Delta V_{pp}}{2} \right) \right|, \left| V_{(i,j)} - \left(V_{s01} + \frac{\Delta V_{pp}}{2} \right) \right| \right\} \quad (5.22)$$

and the expected voltage-shift at cell's position (i, j) can be expressed as

$$\begin{aligned} \Delta V_{(i,j)}^E(\mu_{\gamma_y}, \mu_{\gamma_{xy}}) &= \mu_{\gamma_{xy}} [V_{(i+1,j-1)}^{\mathcal{AP}} - V_{s11}] \\ &+ \mu_{\gamma_y} [V_{(i+1,j)}^{\mathcal{AP}} - V_{s11}] + \mu_{\gamma_{xy}} [V_{(i+1,j+1)}^{\mathcal{AP}} - V_{s11}] \end{aligned} \quad (5.23)$$

The surface plot of cost function $\mathcal{F}(\mu_{\gamma_y}, \mu_{\gamma_{xy}})$ is shown in Fig. 5.5. We notice that the cost function is convex in shape over varying μ_{γ_y} and $\mu_{\gamma_{xy}}$ values. Therefore, this optimization problem can be solved either by exhaustive search method or by employing any convex optimization technique. Since the mean CCR values are

FIGURE 5.5: Surface plot of cost function $\mathcal{F}(\mu_{\gamma_y}, \mu_{\gamma_{xy}})$: mean squared error (MSE) between the measured and modeled CCI voltage-shift expressions.

stationary and do not change w.r.t. flash wear-out dynamics, the CCR optimization needs to be done only once in the offline mode. Hence, the complexity of the chosen optimization algorithm is inconsequential. To this end, we employ the gradient-descent (GD) method to find the optimal $\mu_{\gamma_y}^*$ and $\mu_{\gamma_{xy}}^*$ values. The GD algorithm minimizes the objective function by iteratively solving the following equation

$$\begin{bmatrix} \mu_{\gamma_y}^{(k+1)} \\ \mu_{\gamma_{xy}}^{(k+1)} \end{bmatrix} = \begin{bmatrix} \mu_{\gamma_y}^{(k)} \\ \mu_{\gamma_{xy}}^{(k)} \end{bmatrix} - \alpha \begin{bmatrix} \frac{\partial \mathcal{F}}{\partial \mu_{\gamma_y}} \\ \frac{\partial \mathcal{F}}{\partial \mu_{\gamma_{xy}}} \end{bmatrix} \mathcal{F}(\mu_{\gamma_y}^{(k)}, \mu_{\gamma_{xy}}^{(k)}) \quad (5.24)$$

where ∂ denotes the partial derivative, and k and α represent the GD iteration count and step size parameters, respectively.

5.3 Performance of Neighbor-*A-Priori* Detection Schemes

In this section, we first measure the estimation accuracy of different detection schemes by computing the average estimation error. Then, we analyze the symmetric information rate (SIR) of NAND flash channel, which is also known as the cell storage capacity. This SIR is considered as a true-figure-of-merit, and thus exactly reflects the performance of a given channel detection scheme. Next, we present the error-rate performance. Finally, we discuss the complexity analysis of the proposed and prior-art detection schemes. All the following simulation results are based on 16-level uniform memory sensing quantization.

5.3.1 Normalized Estimation Error

To analyze the CCI estimation accuracy, we define the normalized estimation error for the Post-Comp ($\epsilon^{\text{Post-Comp}}$) [42], Even-AP ($\epsilon^{\text{Even-AP}}$), All-AP ($\epsilon^{\text{All-AP}}$) and All-AP-CCR ($\epsilon^{\text{All-AP-CCR}}$) detection schemes, as follows

$$\epsilon^{\text{Post-Comp}} = \frac{1}{N_t} \sum_{j=1}^{N_t} \frac{|I_{(i,j)} - \hat{I}_{(i,j)}|}{I_{(i,j)}} \times 100 \quad (5.25)$$

$$\epsilon^{\text{Even-AP}} = \frac{1}{N_t} \sum_{j=1}^{N_t} \frac{|I_{(i,j)} - \bar{I}_{(i,j)}|}{I_{(i,j)}} \times 100 \quad (5.26)$$

$$\epsilon^{\text{All-AP}} = \frac{1}{N_t} \sum_{j=1}^{N_t} \frac{|I_{(i,j)} - \tilde{I}_{(i,j)}|}{I_{(i,j)}} \times 100 \quad (5.27)$$

$$\epsilon^{\text{All-AP-CCR}} = \frac{1}{N_t} \sum_{j=1}^{N_t} \frac{|I_{(i,j)} - \tilde{I}_{(i,j)}^*|}{I_{(i,j)}} \times 100 \quad (5.28)$$

In Fig. 5.6, we plot the above error expressions over varying CCI factor (s), keeping the PE cycles fixed at 5K and retention time fixed at $T = 1$ hour. We can

FIGURE 5.6: Plot of normalized estimation error (ε) for *Post-Comp*, *Even-AP*, *All-AP* and *All-AP-CCR* detection schemes over varying CCI factor (s), keeping the PE cycles fixed at 5K and retention time fixed at $T = 1$ hour.

observe that the CCI estimates obtained by employing the All-AP and All-AP-CCR schemes tend to closely match with the actual CCI expression $I_{(i,j)}$, and thus produce lowest estimation errors compared to the Even-AP and Post-Comp [42] detection schemes. More importantly, the respective estimation errors grow very slowly with increasing CCI factor (s), which validates the effectiveness of the proposed detection schemes.

5.3.2 Symmetric Information Rate

Besides the estimation error, the other relevant performance indicator is the symmetric information rate (SIR) of the flash channel. The SIR indicates the quality of soft-information available for the channel decoder, and allows us to analyze the performance limit of a channel detector without getting involved into the tedious simulations for the coded error performance. We compute the overall SIR ($\mathcal{S}$) as the average mutual-information ($\mathcal{MI}$) of the even and odd bit-line cells, as

$$\mathcal{S} = \frac{1}{2} \{ \mathcal{MI}(V_p; \vec{V})^{\text{even}} + \mathcal{MI}(V_p; \vec{V})^{\text{odd}} \} \quad (5.29)$$

where $\mathcal{MI}(V_p; \vec{V}) = H(\vec{V}) - H(\vec{V}|V_p)$ is the mutual-information between the cell's input (V_p) and output ($\vec{V}$) voltage signals, and $H(\cdot)$ denotes the signal entropy function. In Fig. 5.7, we plot the SIR curves under the proposed Even-AP, All-AP and All-AP-CCR detection schemes against increasing CCI factor s , keeping the PE cycles fixed at 5K. To show the improvement that the proposed schemes provide over the prior-art schemes, we also plot the SIR curves for the Post-Comp [42], Pre-Compensation [42] and Dong's [28] detection schemes. It is pertinent to mention that some other prior-art detection schemes, among [81], [88] or [89], can also be used instead of the Dong's [28] detector as all of them have quite similar performance as that of the Dong's detector. Here, the Opt-AP curve shows an upper-bound on the SIR performance. This curve is plotted by assuming that the exact initial voltage levels for the interfering cells are known at the channel detector. In other words, the Opt-AP is an optimal detector which exactly knows the data symbols stored over the interfering cells and consequently makes use of the exact amount of voltage-shift values in (5.18) and (5.19), instead of using their a-priori information V^{AP} and erased-state voltage level V_{s11} . We compare the simulation curves at $\mathcal{S} = 1.8$ bits per cell which corresponds to code-rate of 0.9.

We can observe that the proposed detection schemes outperform all the prior-art schemes. Although the Pre-Compensation and Dong's detection schemes can be employed with non-uniform memory sensing quantization, however, their SIR performance degrades with increasing CCI. As an important observation, we notice that the SIR performance curves under the proposed All-AP and All-AP-CCR schemes nearly approach the performance of the optimal Opt-AP curve, leaving very narrow margin for further improvements.

5.3.3 Detector Error-Rate Performance

To investigate the detector error-rate performance under the proposed detection schemes, in Fig. 5.8, we plot the uncoded bit-error-rate (BER) curves over varying

FIGURE 5.7: Symmetric-information-rate ($\mathcal{S}$) (average mutual-information) for the proposed *Even-AP*, *All-AP* and *All-AP-CCR* detection schemes, compared with the *Dong's*, *Post-Comp*, *Pre-Compensation* and *Opt-AP* detection schemes over varying CCI factor (s), keeping the PE cycles fixed at 5K and retention time at $T = 1$ hour.

channel CCI factor (s), keeping the PE cycles fixed at 5K. We observe a superior error-rate performance under the proposed detection schemes. Among all, the All-AP-CCR detection scheme provides the best error-rate performance, which coincides with the previously reported estimation error and SIR performance results. The superiority of the proposed detection schemes can be further substantiated by simulating the LDPC coded error-rate performance.

5.3.4 Complexity and Design Trade-off

In previous sections, we mainly discussed the performance improvements obtained by using the proposed Even-AP, All-AP and All-AP-CCR detection schemes. Despite the obvious advantages in terms of better estimation accuracy and superior SIR performance, there are certain additional complexity overheads associated with the proposed detection schemes. First note that, in contrast to signal Pre-Compensation [42] and Dong's [28] detection scheme in which non-uniform memory sensing can be

FIGURE 5.8: Illustration of uncoded bit-error-rate (BER) curves simulated under the proposed *Even-AP*, *All-AP* and *All-AP-CCR* detection schemes, compared with prior-art *Post-Comp* detection scheme over varying CCI factor (s), keeping the PE cycles fixed at 5K and retention time at $T = 1$ hour.

employed, all the proposed and prior-art post-processing detection schemes require fine-grained uniform memory sensing precision. This may lead to an increase in memory sensing latency. Besides, the proposed detection schemes incur some more arithmetic and memory sensing operations.

In Table 5.1, we present a detailed complexity analysis for different channel detection schemes. Regarding the Dong's detector [28], it only estimates the CCI distorted voltage distribution functions in order to compute the LLR information for soft-input BP decoding, but ignores the CCI removal, and thus requires no additional computations. In contrast, the Post-Comp detector [42] requires $15N_t$ additions/subtractions and $8N_t$ multiplications for estimating $\hat{I}_{(i,j)}$ and subsequently removing the estimated CCI component from cell's read-back voltage V , where N_t is the number of memory cells per word-line. Comparatively, the proposed detection schemes require a few more operations to compute the a-priori-information $V^{\mathcal{AP}}$ prior to the estimation of CCI components $\bar{I}_{(i,j)}$ and $\tilde{I}_{(i,j)}$, as enumerated in Table 5.1. However, the complexity grows only linearly with N_t and both the proposed and Post-Comp detection schemes require $O(N_t)$ additions and multiplication per

TABLE 5.1: Complexity requirement of different CCI cancellation schemes (per N_t -cell memory-page).

Detection Scheme	ADD/ SUB	MUL	# of Cell Reads	Storage Elements	Detection Latency
Dong's [28]	0	0	N_t	N_t	0
Post-Comp [42]	$15N$	$8N_t$	$2N_t$	$2N_t$	ϕ
Even-AP	$15N_t + \frac{7}{2}N_t$	$8N_t$	$2N_t$	$2N_t$	2ϕ
All-AP	$25N_t + \frac{7}{2}N_t$	$9N_t$	$2N_t$	$2N_t$	4ϕ
All-AP-CCR	$25N_t + \frac{7}{2}N_t$	$9N_t$	$2N_t$	$2N_t$	4ϕ

Note: ϕ is used as a reference detection latency for comparison with the proposed detection schemes.

memory word-line. There is also an increase in the detection latency associated with the proposed detection schemes. This is because the distinct bit-line cells are processed sequentially, as shown in Fig. 5.4, hence requiring additional clock cycles. In terms of memory read operations (the most time consuming process) and hardware storage requirements, the proposed and Post-Comp detection schemes need the same amount of resources. Since the mean CCR values are estimated only once in the offline mode, the proposed All-AP and All-AP-CCR detection schemes have the same level of computational complexity.

In summary, the proposed detection schemes require more additions and multiplications, more clock cycles, but no more memory reads and storage, than the prior art in [42]. However, the additional computations and clock cycles are small and of linear order, just like the Post-Comp detector [42].

5.4 A Joint Application of Techniques Proposed in Chapter 3, 4 and 5

In this section, we adopt a comprehensive fault-tolerate approach by combining the techniques of Chapter 3, 4 and 5. Specifically, we jointly apply the write voltage optimization method of Chapter 3 and the Even-AP detector of Chapter 5. Note that the simulation results presented in Section 5.3 have been plotted for channel

invariant (fixed) write voltage scheme. However, as discussed in Chapter 3, the optimization of intermediate programmed write voltage levels, V_{s10} and V_{s00} , reduce the raw channel error probability and thus result in the improvement of NAND flash endurance performance. Therefore, we first perform the write voltage optimization and then employ the proposed Even-AP detection scheme. For comparison, we also illustrate the performance of prior-art Post-Comp detector [42], and the proposed Even-AP detector by employing the channel invariant (fixed) write voltage levels. Using LDPC 4K-code and 16-level uniform quantization, in Fig. 5.9, we plot the bit-error-rate (BER) curves against varying PE cycles, keeping the CCI factor fixed at $s = 1.50$ and $s = 1.75$, and retention time at $T = 1$ hour. It can be observed that the joint application of Even-AP detector with write voltage optimization method significantly outperforms the prior-art method which employs fixed write voltage levels.

Next, we verify the joint applicability of RABP decoder of Chapter 4 with the Even-AP detector of Chapter 5 and write voltage optimization method of Chapter 3. It should be noted that, contrary to the non-uniform quantization scheme of Fig. 4.3, the uniform based quantization scheme does not split the voltage range into unequal data and erasure regions. Instead, the individual quantization regions are divided into equally spaced voltage regions with uniform interval D_u . Thus, the retention-error correction and retention-erasure correction techniques of RABP decoder (see Algorithm 1) can not be directly applied, but are required to be modified in-accordance with the uniform quantization regions. Specifically, we re-map the LLR computation for second round of RABP decoding such that, when $R_{n-1} < v \leq R_n$ for $n = \{1, 2, 3, \dots, 16\}$, the new LLR values, $\hat{L}_{msb}$ and $\hat{L}_{lsb}$, will become

$$\hat{L}_{msb}(n) \Rightarrow L_{msb}(n+1) \quad (5.30)$$

$$\hat{L}_{lsb}(n) \Rightarrow L_{lsb}(n+1) \quad (5.31)$$

FIGURE 5.9: Illustration of LDPC 4K-code bit-error-rate (BER) performance by employing the prior-art Post-Comp detector, and the proposed Even-AP detector combined with the write voltage optimization, plotted against increasing PE cycles, keeping the CCI factor fixed at $s = 1.50$ and $s = 1.75$, and retention time at $T = 1$ hour.

This LLR re-mapping procedure is depicted in Fig. 5.10. Recall that the process of LLR modification tends to neutralize the effect of retention noise by re-mapping the LLR values into the next higher quantization region. To show the overall performance improvement, in Fig. 5.11, we plot the LDPC 4K-code BER curves versus data retention time by employing the the prior-art Post-Comp detector [42], and the proposed Even-AP detector (Chapter 5), Even-AP plus write voltage optimization method (Chapter 3+Chapter 5), and Even-AP plus write voltage optimization plus RABP decoder schemes (Chapter 3+Chapter 4+Chapter 5), keeping the CCI factor fixed at $s = 1.00$ and PE cycles at 5K. It can be seen that the combination of proposed signal processing techniques yields substantial improvement in the error-rate performance compared to the prior-art method or the system which separately employs the individual proposed techniques.

FIGURE 5.10: Illustration of log-likelihood-ratio (LLR) re-mapping for second round of RABP decoding while employing the uniform based quantization levels.

FIGURE 5.11: Illustration of LDPC 4K-code bit-error-rate (BER) performance by employing the prior-art Post-Comp detector, and the proposed Even-AP detector combined with the write voltage optimization and RABP decoding, plotted against increasing retention time T , keeping the CCI factor fixed at $s = 1.00$ and PE cycles at 5K.

5.5 Chapter Summary

In this chapter, three post-processing cell-to-cell interference (CCI) cancellation schemes, namely the Even-AP, the All-AP and the All-AP-CCR detectors, for 2-bit per cell NAND flash memory are presented. The proposed schemes are implemented over the even-odd bit-line architecture, but they can easily be extended for the all bit-line architecture. In the Even-AP detector, the even bit-line cells are cleaned using the *a-priori* information from the odd bit-line interfering cells. In the All-AP

detection scheme, the odd bit-line cells are cleaned first, based on the *a-priori* information from the next word-line cells, followed by the even bit-line cells. In the All-AP-CCR scheme, the All-AP detector is extended by estimating the mean CCR values along victim cell's vertical and diagonal directions. For the All-AP-CCR scheme, the estimation of mean CCRs can be performed in the offline mode, thus avoiding the need for run-time computations. All the proposed schemes yield substantial improvement in the cell storage capacity and error-rate performance. Finally, this chapter explores the possibility of integrating all the proposed coding and signal processing solutions to jointly mitigate the effect of major threshold voltage signal impairments.

Chapter 6

Improved Fixed and Dynamic Scheduling for Belief-Propagation (BP) Decoding

A belief-propagation (BP) scheduling strategy, which defines the order of message passing along the edges of the LDPC code graph, is considered to be an important aspect of BP algorithm as it affects the decoding complexity, speed of decoding convergence and error-rate performance of BP decoder. In this chapter, we present a novel *column-weight based scheduling* (CWS) scheme for shuffled BP decoding of irregular LDPC codes. In this scheme, the variable nodes are updated in descending order of their column weight. Using simulation results, it is shown that the proposed CWS can noticeably increase the convergence speed, without introducing additional complexity or error rate degradation.

Next, we introduce an *informed fixed scheduling* (IFS) for improving the decoding convergence of both regular and irregular LDPC codes. The IFS finds an appropriate order of variable nodes in accordance with the number of updated neighbors in the code graph, ensuring that the maximum number of latest message updates are utilized within a single iteration. This allows the utilization of most

reliable message updates in a timely manner, leading to faster error-rate convergence. This scheduled sequence is then serially decoded under the shuffled BP algorithm.

Furthermore, we present an *edge-based* dynamic scheduling for low-complexity BP decoding of high-rate LDPC codes. In contrast to CWS and IFS schemes, where all the graph edges are updated per decoding iteration, the proposed edge-based scheduling selectively updates the edges of the code graph based on the run-time reliability of variable and check nodes. Specifically, new message update is propagated exclusively along the unreliable edges of the code graph. This reduces the decoding complexity of BP algorithm as only a partial set of message updates is computed per decoding iteration. Besides, restricting the flow of message updates may also preclude the occurrence of short graph cycles, which will preserve the BP message independence assumption at certain variable and check nodes. Using numerical simulations, it is shown that the proposed edge-based schedule significantly reduces the BP decoding complexity compared to the prior-art BP schedules while simultaneously improving on the error-rate performance.

Finally, we propose a *dynamic silent-variable-node-free scheduling* (D-SVNFS) which regulates the message passing by using the check-to-variable message residuals. To determine the next message update, the D-SVNFS scheme computes the on-demand message residuals and selects the largest one associated with the last updated variable and check nodes. Additionally, the proposed D-SVNFS scheme attempts to propagate more message updates towards the erroneous variable nodes. This facilitates the correction of incorrect variable nodes with higher priority, which leads to faster decoding convergence speed.

6.1 Background

In the previous chapters, we only employ the conventional shuffled algorithm [74] for BP decoding LDPC codes. However, considering the large decoding latency

associated with the iterative BP algorithm, in this chapter, we devise some improved scheduling schemes which not only offer lower decoding complexity and faster rate of decoding convergence, but also provide superior error-rate performance when compared to the conventional shuffled schedule. It is pertinent to mention that the proposed scheduling schemes are applicable and equally effective for data storage channels as well as for AWGN and Rayleigh fading communication channels. However, in this thesis, we restrict our analysis and present simulation results only for the 2-bit per cell NAND flash channel model of Chapter 2.

6.1.1 Overview of Belief-Propagation (BP) Decoding

The belief-propagation (BP) decoding is an iterative message-passing algorithm which is being used to decode low-density parity-check (LDPC) codes. Under BP decoding algorithm, an LDPC code can approach the performance of Shannon's capacity limit. A binary (N, K) LDPC code can be defined by an $M \times N$ parity-check matrix $\mathbf{H}$, with $M = N - K$, where K and N represent the input and output code-word lengths, respectively. Equivalently, an LDPC code can be described by using a bi-partite code graph with N variable nodes, v_n for $1 \leq n \leq N$, and M check nodes, c_m for $1 \leq m \leq M$, where the matrix element $h_{m,n} = 1$ defines a bi-directional edge $\vec{e}_{m,n}$ between c_m and v_n . In BP decoding algorithm, new message updates are iteratively propagated from variable node v_n to check node c_m , denoted by $m_{v_n \rightarrow c_m}$, and from check node c_m to variable node v_n , denoted by $m_{c_m \rightarrow v_n}$, along the edges $\vec{e}_{m,n}$ in the code graph. These message updates can be computed as [92]

$$m_{v_n \rightarrow c_m}^{(i)} = L_{v_n} + \sum_{c_j \in \mathcal{M}(v_n) \setminus c_m} m_{c_j \rightarrow v_n}^{(i)} \quad (6.1)$$

$$m_{c_m \rightarrow v_n}^{(i)} = 2 \tanh^{-1} \left(\prod_{v_j \in \mathcal{N}(c_m) \setminus v_n} \tanh \left(\frac{m_{v_j \rightarrow c_m}^{(i-1)}}{2} \right) \right) \quad (6.2)$$

where i is the iteration count, L_{v_n} is the intrinsic channel information, also referred to as the log-likelihood-ratio (LLR), of variable node v_n , $\mathcal{M}(v_n) \setminus c_m = \{c_j : h_{jn} = 1, j \neq m\}$ is the set of neighbors of variable node v_n excluding check node c_m and $\mathcal{N}(c_m) \setminus v_n = \{v_j : h_{mj} = 1, j \neq n\}$ is the set of neighbors of check node c_m excluding variable node v_n . The output LLR $\Lambda_{v_n}^{(i)}$ corresponding to variable node v_n can be computed as

$$\Lambda_{v_n}^{(i)} = L_{v_n} + \sum_{c_j \in \mathcal{M}(v_n)} m_{c_j \rightarrow v_n}^{(i)} \quad (6.3)$$

After every decoding iteration, a hard-decision on code-word bit a_n , denoted by $\hat{a}_n^{(i)}$, is performed on the basis of sign of output LLR $\Lambda_{v_n}^{(i)}$. For a valid decoded code-word $\hat{\mathbf{a}}^{(i)} = (\hat{a}_1^{(i)}, \hat{a}_2^{(i)}, \dots, \hat{a}_N^{(i)})$, the M parity-check (syndrome) equations must be satisfied, given by

$$s_{c_m}^{(i)} = \sum_{n: v_n \in \mathcal{N}(c_m)} h_{m,n} \hat{a}_n^{(i)} = 0 \quad (6.4)$$

The BP decoding stops once these parity-check equations are satisfied or when the maximum iteration count, denoted by $I_{\max}$, is reached. For ease of notation, we will remove the iteration count superscript (i) in all the subsequent equations.

6.1.2 Related Work

There has been an extensive research on the design and analysis of BP decoding schedules. The *flooding* [92] is the conventional BP schedule in which all the variable-to-check (V2C), and subsequently the check-to-variable (C2V), message updates are simultaneously passed in each iteration. To improve the decoding convergence speed, and equivalently to reduce the average iteration count that is required to obtain the error-rate convergence, individual nodes can be updated in sequential order. The *shuffled* [74] and *layered* [93] are two commonly used sequential BP schedules, wherein the variable and check nodes are updated by

following a pre-determined (fixed) sequential order. Analytical studies reveal that the sequential schedules tend to converge two times faster as compared to the flooding schedule [94]. Yet, the demand for achieving faster decoding convergence motivates further research towards the design of new BP scheduling schemes.

To reduce the BP iteration cost, the *forced convergence* (FC) [95], *variable-node lazy* (VLS) [96] and *check-node lazy* (CLS) [97] schedules update a partial set of variable or check nodes per decoding iteration. Similarly, for reducing the BP iteration cost for the non-binary LDPC codes, the check-node and the variable-node reliability-based scheduling schemes are reported in [98] and [99], respectively. Apart from reliability-based scheduling, the *probabilistic* [100] and *graph-based* [101] scheduling schemes regulate message passing in-accordance with the length of the cycles in the code graph. All these existing BP schedules either reduce the decoding complexity or improve on the error-rate performance.

In addition to fixed order BP schedules, there are also *residual-based* dynamic schedules reported in the open literature [102–106], which regulate the message-passing by using the message residuals. These dynamic BP schedules achieve lower error rates because they enable the BP decoder to overcome many trapping sets encountered during the decoding process [107]. On the bleak side, these schedules incur significant additional complexity, striving to compute the message-residuals. In order to reduce the complexity cost associated with dynamic schedules, a reliability-based dynamic BP schedule is proposed in [108] which offers faster error-rate convergence at reduced complexity. Inspired by the dynamic schedules, a *maximum mutual-information* based BP schedule, which predicts the next message update based on the mutual information increase, is proposed in [109]. Again, this algorithm improves on the error-rate performance, however, the resultant hardware implementation complexity and the storage requirement may become prohibitively large as it has to record the complete sequence of message updates.

6.2 Column-Weight Scheduling (CWS)

We propose a variable node based sequential BP schedule, specifically for irregular LDPC codes, by using the column weight of variable nodes. Since an irregular LDPC code graph contains variable nodes with varying column weights, we propose to update them in a descending order such that high column-weight variable nodes are updated first, followed by low column-weight variable nodes. We refer to this scheduling as high-to-low column-weight based scheduling (CWS). For the proposed scheme, the decoder needs to sort the variable nodes by decreasing column weight order only once in the offline mode, thereby precluding the run-time scheduling computations for the selection of variable nodes. Thus, the proposed CWS scheme does not introduce any additional run-time computational complexity. The detailed steps of the proposed CWS scheme are outlined in Algorithm 4. The vector $\mathbf{s}$ is used to store the variable nodes in high-to-low column-weight order. Then the variable nodes are serially updated following the sequence of vector $\mathbf{s}$.

Algorithm 4 : Column-Weight based Scheduling (CWS)

- 1: Load vector $\mathbf{s} = \{v_1, v_2, \dots, v_N\}, \Rightarrow$ variable nodes sorted by high-to-low column-weight order
 - 2: **for** all $n = 1 : N$ **do**
 - 3: **for** every $c_m \in$ neighborhood of v_n **do**
 - 4: Generate and propagate $m_{c_m \rightarrow v_n}$
 - 5: **end for**
 - 6: **for** every $c_m \in$ neighborhood of v_n **do**
 - 7: Generate and propagate $m_{v_n \rightarrow c_m}$
 - 8: **end for**
 - 9: **end for**
-

6.2.1 Aggregated Message Count

To investigate the reason behind the improvement in convergence rate of the CWS based shuffled BP decoder, we first note that the number of code graph edges that

are updated per decoding iteration is fixed, irrespective of any particular sequence of variable node processing. Thus, in one decoding iteration, each edge is updated exactly once regardless of which sequence of variable node is followed. However, when the variable nodes are updated in descending order of their column weight, the BP decoder tends to generate a large *aggregated message count* per decoding iteration. Here, aggregated message refers to the total number of updated messages (soft information, $m_{v \rightarrow c}$) summed over all the variable nodes. This quantity measures the amount of updated information available for processing the next-in-queue variable node. Therefore, a higher number of aggregated messages provides more timely information updates for the subsequent variable nodes and thus helps to accelerate the convergence speed of CWS based shuffled BP decoder.

Example 6.1. In Fig. 6.1, a simple irregular LDPC code graph is shown which contains five variable nodes with three column-weight 2 $\{v_1, v_2, v_3\}$ and two column-weight 3 $\{v_4, v_5\}$ variable nodes. Under the proposed high-to-low CWS scheme, variable nodes will be processed in the following order $v_5 \Rightarrow v_4 \Rightarrow v_3 \Rightarrow v_2 \Rightarrow v_1$, whereas the conventional shuffled schedule will follow this order $v_1 \Rightarrow v_2 \Rightarrow v_3 \Rightarrow v_4 \Rightarrow v_5$. We observe that the total number of updated edges per iteration is equal to 12. However, in the high-to-low order, individual variable nodes v_4, v_3, v_2 and v_1 can take advantage of 3, 6, 8 and 10 new information updates, respectively, at the instance of their processing. In contrast, in the low-to-high order, the variable nodes v_2, v_3, v_4 and v_5 can receive 2, 4, 6 and 9 new information updates, respectively. Overall, the number of aggregated messages for the high-to-low ($\mathcal{A}_{H-L}$) and low-to-high ($\mathcal{A}_{L-H}$) ordering are 39 and 33, respectively. This difference between $\mathcal{A}_{H-L}$ and $\mathcal{A}_{L-H}$ influences the convergence speed of shuffled BP decoder.

To compute the number of aggregated messages for a general irregular LDPC code, let us denote the fraction of degree- i variable nodes with $\tilde{\lambda}_i$ ¹ (node perspective),

¹The node perspective degree $\tilde{\lambda}_i$ and the edge perspective degree λ_i are related by:

$$\tilde{\lambda}_i = \frac{\lambda_i/i}{\int_0^1 \lambda(x)dx}$$

FIGURE 6.1: A simple tanner graph example showing the *aggregated message count* observed while employing the proposed high-to-low column-weight scheduling $\mathcal{A}_{H-L}$ (top figure), and compared with low-to-high column-weight scheduling $\mathcal{A}_{L-H}$ (bottom figure).

the maximum variable node degree with $d_{v,max}$ and LDPC code-word length with N . Then, the number of aggregated messages for high-to-low and low-to-high column-weight ordering can be given by (6.5) and (6.6) as

$$\mathcal{A}_{H-L} = \sum_{i=1}^{d_{v,max}} \left[\sum_{k=1}^{N\tilde{\lambda}_i} \left(\sum_{j=i+1}^{d_{v,max}} j(N\tilde{\lambda}_j) + ik \right) \right] \quad (6.5)$$

$$\mathcal{A}_{L-H} = \sum_{i=1}^{d_{v,max}} \left[\sum_{k=1}^{N\tilde{\lambda}_i} \left(\sum_{j=1}^{i-1} j(N\tilde{\lambda}_j) + ik \right) \right] \quad (6.6)$$

For the case of random column-weight ordering, we can approximate the number of aggregated messages $\mathcal{A}_R$ as

$$\mathcal{A}_R \approx \bar{d}_v \frac{N}{2} (N + 1) \quad (6.7)$$

FIGURE 6.2: Number of aggregated messages for high-to-low $\mathcal{A}_{H-L}$, random $\mathcal{A}_R$ and low-to-high $\mathcal{A}_{L-H}$ column-weight based scheduling: employing capacity approaching LDPC codes with $d_{v,max} = 5$, $d_{v,max} = 10$, $d_{v,max} = 20$, $d_{v,max} = 30$ and $d_{v,max} = 50$, respectively.

where $\bar{d}_v$ is the average variable node degree, given by

$$\bar{d}_v = \sum_{i=1}^{d_{v,max}} i \tilde{\lambda}_i$$

To plot the number of aggregated messages, we make use of the capacity approaching rate-1/2 irregular LDPC codes with optimized degree distribution, as given in [110]. For $N = 2000$ and code ensembles with $d_{v,max} = 5$, $d_{v,max} = 10$, $d_{v,max} = 20$, $d_{v,max} = 30$ and $d_{v,max} = 50$ obtained from [110], we plot $\mathcal{A}_{H-L}$, $\mathcal{A}_R$ and $\mathcal{A}_{L-H}$ quantities in Fig. 6.2. In this figure, we notice that the value of $\mathcal{A}_{H-L}$ is larger than both $\mathcal{A}_{L-H}$ and $\mathcal{A}_R$, as expected.

6.2.2 Mutual Information Increment

In order to analyze the effect of higher aggregated message count that is realized by employing the proposed CWS scheme, in Fig. 6.3, we plot the average mutual information ($\mathcal{MI}$) computed at the output of variable node processor against

FIGURE 6.3: Illustration of mutual information curves for the high-to-low ($\mathcal{MI}_{H-L}$) and low-to-high ($\mathcal{MI}_{L-H}$) column-weight based scheduling schemes for $d_{v,max} = 50$ LDPC code.

the number of decoding iterations for the AWGN channel, keeping the channel SNR fixed at 0.5 dB, 1.0 dB and 2.0 dB. In this figure, we employ the proposed high-to-low ($\mathcal{MI}_{H-L}$) column-weight ordering and compare with the low-to-high ($\mathcal{MI}_{L-H}$) column-weight ordering by using $d_{v,max} = 50$ LDPC code [110]. We observe that, for moderate channel SNR values, the mutual information tends to converge faster for the proposed high-to-low CWS scheme. This increase in mutual information leads to faster convergence of shuffled BP decoder.

6.3 Informed Fixed Scheduling (IFS)

The CWS scheme is only applicable for irregular LDPC codes. To improve the convergence of both regular and irregular LDPC codes, we propose an informed fixed scheduling (IFS). In order to explain the proposed IFS methodology, let us first observe the difference between the shuffled and flooding BP schedules. Under the flooding schedule, the message propagation takes place simultaneously. In contrast, the shuffled schedule updates variable nodes serially. For better understanding, we

FIGURE 6.4: An example of how shuffled BP decoding utilizes the newly available message updates: variable node v_2 receives new message $v_{v_1 \rightarrow c_1}$ via check node c_1 .

split the C2V update expression (6.2) into two factors, given by

$$m_{c_m \rightarrow v_n}^{(i)} = 2 \tanh^{-1} \left(\prod_{\substack{v_j \in \mathcal{N}(c_m) \\ v_j < v_n}} \tanh \left(\frac{v_{v_j \rightarrow c_m}^{(i)}}{2} \right) \cdot \prod_{\substack{v_j \in \mathcal{N}(c_m) \\ v_j > v_n}} \tanh \left(\frac{v_{v_j \rightarrow c_m}^{(i-1)}}{2} \right) \right) \quad (6.8)$$

In this expression, the first term incorporates the V2C messages computed in the current iteration i , while the second one takes on the V2C messages computed in the previous iteration $i - 1$. The shuffled decoder makes it possible to access the latest message updates $v_{v_j \rightarrow c_m}^{(i)}$ for $v_j < v_n$ via check nodes c_m for $c_m \in \mathcal{M}(v_n)$ while updating variable node v_n . To see this effect, a simple code graph is shown in Fig. 6.4. Given the selected variable node v_2 , the newly available message $v_{v_1 \rightarrow c_1}$ is utilized via check node c_1 . This is how the shuffled BP decoder yields faster convergence as compared to the flooding schedule. Therefore, it makes sense to select a variable node such that its neighboring check nodes are connected with maximum number of updated variable nodes. In this way, a large number of variable nodes will receive the latest message updates within a single decoding iteration, consequently reducing the iteration count required for error-rate convergence. Thus, instead of arbitrary variable node selection, scheduling based on the availability of latest message updates is adapted in the proposed IFS scheme.

6.3.1 Maximum Variable Node Counter

The proposed IFS scheme is an extension of our CWS scheme, hence it follows a high-to-low column-weight order for the variable nodes with unequal column weights. In addition, a new objective of the IFS scheme is to determine the scheduling order for the equal column-weight variable nodes, ensuring that each selected node receives maximum number of latest message updates from previously updated variable nodes. To this end, the set of variable nodes $\{v_1, v_2, \dots, v_N\}$ are divided into different groups according to their column weights. Thereafter, following the high-to-low column-weight order, the individual groups are scheduled using the IFS scheme. To achieve this, we define two set of counters, namely the *check-counter* (ζ_m) for $1 \leq m \leq M$, and the *var-counter* (η_n) for $1 \leq n \leq N$. Given the iteration count i , the value of ζ_m represents the number of already updated variable nodes connected with check node c_m , whereas the quantity η_n represents the cumulative sum of ζ_m for $c_m \in \mathcal{M}(v_n)$, as given by

$$\eta_n = \sum_{c_m \in \mathcal{M}(v_n)} \zeta_m \quad (6.9)$$

In this perspective, the value of η_n reflects the total number of latest message updates accessible to variable node v_n . Thus, the IFS scheme tends to find the sequence of variable nodes by selecting variable node v_n such that the corresponding variable-counter η_n attains the maximum value. To begin with, all ζ_m and η_n are initialized to zero. This is to represent the starting point of a new decoding iteration. The IFS scheme proceeds to select the variable node associated with the largest η_n . In case of multiple variable nodes sharing the same maximum η_n , it randomly selects any one among them, say v_i , and records the scheduling order into a first-in-first-out (FIFO) structure $\mathbf{S}$. Then, the check-counter ζ_m for all $c_m \in \mathcal{M}(v_i)$ are incremented and the relevant var-counter are updated according to (6.9). This procedure is repeated until all the variable nodes are scheduled. The detailed steps of IFS

scheme are outlined in Algorithm 5. Note that the IFS scheme needs to be run only once in the off-line mode and the stored scheduling order is retrieved at the time of decoding. This precludes the need for additional run-time computations.

Example 6.2. *To show the working of IFS scheme, we use a simple code graph, as shown in Fig. 6.5. Since this is an irregular code graph, column-weight 3 variable nodes $\{v_5, v_6, v_7\}$ are scheduled first followed by column-weight 2 variable nodes $\{v_1, v_2, v_3, v_4\}$. In the first place, variable node v_5 is selected as η_5, η_6 and η_7 are all equal to 0. Then, the check-counter ζ_1, ζ_4 and ζ_5 are incremented since c_1, c_4 and c_5 are connected with v_5 , and the relevant var-counter values are updated, as shown in Fig. 6.5(b). Among the remaining two column-weight 3 variable nodes, v_7 is selected next on the account of larger η_7 , and the respective check-counter and var-counter values are updated, as shown in Fig. 6.5(c). Similarly, the waiting column-weight 3 variable node v_6 is selected afterwards, as shown in Fig. 6.5(d). Likewise, the column-weight 2 variable nodes are scheduled, as shown in Fig. 6.5(f). The final sequence obtained in this example is given by $v_5 \Rightarrow v_7 \Rightarrow v_6 \Rightarrow v_3 \Rightarrow v_1 \Rightarrow v_2 \Rightarrow v_4$. In contrast, the conventional shuffled BP schedule will process variable nodes according to $v_1 \Rightarrow v_2 \Rightarrow v_3 \Rightarrow v_4 \Rightarrow v_5 \Rightarrow v_6 \Rightarrow v_7$, whereas the CWS scheme will follow the sequence $v_7 \Rightarrow v_6 \Rightarrow v_5 \Rightarrow v_4 \Rightarrow v_3 \Rightarrow v_2 \Rightarrow v_1$.*

Algorithm 5 : Informed Fixed Scheduling (IFS)

- 1: Initialize $\eta_n = 0, \forall 1 \leq n \leq N \Rightarrow$ variable-counter
 - 2: Initialize $\zeta_m = 0, \forall 1 \leq m \leq M \Rightarrow$ check-counter
 - 3: **while** $\mathbf{S} \notin \{v_1, v_2, \dots, v_N\}$ **do**
 - 4: Find $i = \arg \max_{n=\{1,2,\dots,N\}} \{\eta_n\}$
 - 5: Select variable node v_i and store into vector $\mathbf{S}$
 - 6: Reset $\eta_i = 0$
 - 7: **for** every $c_m \in \mathcal{M}(v_i)$ **do**
 - 8: Increment check-counter: $\zeta_m = \zeta_m + 1$
 - 9: **for** every $v_n \in \mathcal{N}(c_m) \setminus v_i$ **do**
 - 10: Increment variable-counter: $\eta_n = \eta_n + 1$
 - 11: **end for**
 - 12: **end for**
 - 13: **end while**
-

FIGURE 6.5: A working example of variable node scheduling $\{v_1, \dots, v_7\}$ by using the proposed *informed fixed scheduling* (IFS) scheme: scheduled sequence is given by $v_5 \Rightarrow v_7 \Rightarrow v_6 \Rightarrow v_3 \Rightarrow v_1 \Rightarrow v_2 \Rightarrow v_4$.

6.4 Edge-based Shuffled (e-Shuffled) Scheduling

The main objective of the proposed edge-based shuffled (e-Shuffled) scheduling is to reduce the computational complexity and improve the decoding convergence rate of BP decoder by partially updating the edges of the code graph. In our previous two scheduling schemes, CWS and IFS, all the edges in the code graph are updated once per decoding iteration. This involves bi-directional message passing from C2V and V2C nodes. However, in BP decoding, it is often observed that some of the

graph edges converge (achieve large LLR values) within a few decoding iterations and do not require message updates thereafter. Thus, in the proposed e-Shuffled scheduling, we monitor the run-time convergence status of the graph edges and update only the non-converged (less reliable) ones in the next decoding iteration, while skipping those that have already converged.

In addition to complexity reduction, we also demonstrate the better error-rate performance of the proposed e-Shuffled scheduling. In BP decoding, the message passing is performed based on the fundamental assumption that all the incoming and outgoing message updates at any given node are statistically independent, which only holds when the underlying code graph is cycle-free. However, for short-to-moderate length LDPC codes, the underlying code graph does contain short cycles which violate this message independence assumption, leading to sub-optimal decoder performance. This behavior is even more prominent in high-rate LDPC codes, which are preferably used for NAND flash memory. Thus, curtailing the flow of message updates with the proposed e-Shuffled schedule may also preclude the occurrence of graph cycles, resulting in an improved error-rate performance.

6.4.1 Graph Edge Convergence in BP Decoding

To perform the edge-based BP scheduling, we need to describe the convergence (reliability) criterion for a graph edge. In this direction, we first define the reliability measure of a check node c_m , denoted by $\mathcal{R}_{c_m}$. This is given by the smallest LLR magnitude observed among the neighboring variable nodes of c_m , computed as

$$\mathcal{R}_{c_m} = \min_{v_n \in \mathcal{N}(c_m)} \{|\Lambda_{v_n}|\} \quad (6.10)$$

where $|\cdot|$ is the absolute value function. Note that the LLR magnitude $|\Lambda_{v_n}|$ indicates the confidence level on the hard-decision decoding of bit $\hat{a}_n$. Thus, larger values of $\mathcal{R}_{c_m}$ reflect a high level of confidence on the set of tentatively decoded code-word

bits associated with the neighboring variable nodes of c_m , $\{\hat{a}_n | v_n \in \mathcal{N}(c_m)\}$. Using the check node reliability measure $\mathcal{R}_{c_m}$ and the parity-check (syndrome) value s_{c_m} , we split the M number of check nodes into 3 distinct types, namely the *Type-1*, *Type-2* and *Type-3* check nodes, as follows

- *Type-1* (most unreliable): All the check nodes with unsatisfied parity-check, given by $s_{c_m} \neq 0$, will be classified into Type-1 category. These nodes are among the most unreliable set of check nodes because the non-zero parity-check condition certainly indicates an error among the set of decoded code-word bits $\{\hat{a}_n | v_n \in \mathcal{N}(c_m)\}$. Therefore, we will pass new message update towards all the variable nodes $v_n \in \mathcal{N}(c_m)$.
- *Type-2* (unreliable): All the check nodes corresponding to satisfied parity-check, $s_{c_m} = 0$, and having reliability measure smaller than a pre-defined threshold, that is $\mathcal{R}_{c_m} \leq \mathcal{T}$, will be categorized as Type-2 check nodes. Although, the parity-check equation is satisfied, the low reliability measure of Type-2 check node leads to a weaker level of confidence and hence sheds doubt on the tentatively decoded code-word bits. It suggests that the Type-2 check node may still have some incorrectly decoded neighboring variable nodes. To this end, we will identify the set of t *minimum-reliability neighbors* of check node c_m , denoted by $\mathcal{N}_{t-\min}(c_m)$, treating them as among the likely erroneous variable nodes. Subsequently, we will pass new message updates only towards these probable erroneous variable nodes $v_n \in \mathcal{N}_{t-\min}(c_m)$.
- *Type-3* (reliable): All the remaining check nodes, where $s_{c_m} = 0$ and $\mathcal{R}_{c_m} > \mathcal{T}$, will be classified as Type-3 check nodes. These are among the most reliable set of check nodes because all the neighboring variable nodes have achieved high reliability. Therefore, we will not update any of the neighboring variable nodes of Type-3 check node in the next iteration.

In this work, we select the value of t by using numerical simulations such that the decoded error-rate is minimized, whereas we choose the reliability threshold $\mathcal{T}$ based on the check-node-type probability, as explained in the next section. The minimum-reliability neighbor set $\mathcal{N}_{t\text{-min}}(c_m)$ is given by

$$\begin{aligned} \mathcal{N}_{t\text{-min}}(c_m) = \{ & \arg \min_{v_n \in \mathcal{N}(c_m)}^1 \{|\Lambda_{v_n}|\}, \arg \min_{v_n \in \mathcal{N}(c_m)}^2 \{|\Lambda_{v_n}|\}, \dots, \\ & \arg \min_{v_n \in \mathcal{N}(c_m)}^{t-1} \{|\Lambda_{v_n}|\}, \arg \min_{v_n \in \mathcal{N}(c_m)}^t \{|\Lambda_{v_n}|\} \} \end{aligned} \quad (6.11)$$

where function $\min^t\{.\}$ finds the t -th minimum element. Based on the check node categorization, we represent the convergence status of an edge by using a *binary edge-state vector*, $\mathbf{U}$, given by

$$\mathbf{U} = \{\varepsilon_{m,n} | 1 \leq m \leq M, v_n \in \mathcal{N}(c_m)\} \quad (6.12)$$

where $\varepsilon_{m,n} \in \{0, 1\}$. This edge-state vector will be used to identify whether an edge is to be updated at the following iteration. The value $\varepsilon_{m,n} = 0$ indicates that the edge $\vec{e}_{m,n}$ has converged and will not be updated, whereas, $\varepsilon_{m,n} = 1$ indicates that the corresponding edge has not converged and will be updated at the next iteration. The edge-state update criterion is formally described in Table 6.1.

In summary, at each decoding iteration, we will update the edge-state vector according to Table 6.1, and propagate new message updates only where $\varepsilon_{m,n} = 1$, as shown in Fig. 6.6. Using this edge-state vector, we regulate the message-passing under the proposed e-Shuffled schedules as described in Algorithm 6. The counter variables k and z in Algorithm 6 count the average number of V2C/C2V message updates and average number of edge-state vector ($\mathbf{U}$) updates performed per decoded code-word, respectively.

TABLE 6.1: Edge-state ($\varepsilon_{m,n}$) update criterion.

Check node Type	Edge-state update criterion
Type-1 ($s_{c_m} \neq 0$)	$\varepsilon_{m,n} = 1, \forall v_n \in \mathcal{N}(c_m)$
Type-2 ($s_{c_m} = 0, \mathcal{R}_{c_m} \leq \mathcal{T}$)	$\varepsilon_{m,n} = 1, \forall v_n \in \mathcal{N}_{t-\min}(c_m)$ $\varepsilon_{m,n} = 0, \forall v_n \notin \mathcal{N}_{t-\min}(c_m)$
Type-3 ($s_{c_m} = 0, \mathcal{R}_{c_m} > \mathcal{T}$)	$\varepsilon_{m,n} = 0, \forall v_n \in \mathcal{N}(c_m)$

FIGURE 6.6: Illustration of message-passing along the unreliable edges of the code graph under the proposed *edge-based shuffled* (e-Shuffled) scheduling scheme.

6.4.2 Selection of Reliability Threshold

To select the check node reliability threshold $\mathcal{T}$, we define the *check-node-type* probability. This is given by the probability of a check node being selected as either Type-2, or Type-3, denoted by $p(\text{Type-2})$ and $p(\text{Type-3})$, respectively, as a function of the neighboring variable node LLR (Λ) and the check node degree ($\bar{d}_c$). These probability expressions are derived in the Appendix B, given by

$$p(\text{Type-2}) = \left[1 - \left(\frac{e^\Lambda}{1 + e^\Lambda} \right)^{\bar{d}_c} \right] \left[\frac{1}{2} + \frac{1}{2} \prod_{i=1}^{\bar{d}_c} \left(1 - \frac{2e^\Lambda}{1 + e^\Lambda} \right) \right] \quad (6.13)$$

$$p(\text{Type-3}) = \left(\frac{e^\Lambda}{1 + e^\Lambda} \right)^{\bar{d}_c} \left[\frac{1}{2} + \frac{1}{2} \prod_{i=1}^{\bar{d}_c} \left(1 - \frac{2e^\Lambda}{1 + e^\Lambda} \right) \right] \quad (6.14)$$

Algorithm 6 : Edge-based shuffled (e-Shuffled) Schedule

```

1: Initialize all  $m_{v_n \rightarrow c_m} = L_{v_n}$ 
2: Reset  $k = 0, z = 0$ 
3: Set  $k_{\max} = M\bar{d}_c I_{\max}$ 
4: while  $k < k_{\max}$  do
5:   Update edge-state vector U according to Table 6.1
6:   for  $n = 1 : N$  do
7:     for every  $c_m \in \mathcal{M}(v_n)$  do
8:       if  $\varepsilon_{m,n} = 1$  then
9:         Generate and propagate  $m_{c_m \rightarrow v_n}$  (6.2)
10:      end if
11:    end for
12:    for every  $c_m \in \mathcal{M}(v_n)$  do
13:      if  $\varepsilon_{m,n} = 1$  then
14:        Generate and propagate  $m_{v_n \rightarrow c_m}$  (6.1)
15:         $k = k + 1$ 
16:      end if
17:    end for
18:  end for
19:   $z = z + 1$ 
20: end while

```

In Fig. 6.7, we plot these two expressions for $\bar{d}_c = 10$, $\bar{d}_c = 20$ and $\bar{d}_c = 30$. It can be observed that, with increasing LLR, the probability of Type-2 check node tends to decrease, whereas the probability of Type-3 check node tends to increase. To balance between the two, we will set the value of $\mathcal{T}$ such that both type of check nodes will have an identical probability. This leads to the intersection point of two probability curves, as shown in Fig. 6.7. Therefore, we will equate (6.13) and (6.14) and solve for $\mathcal{T}$, as follows

$$\begin{aligned}
p(\text{Type-2})_{|\Lambda=\mathcal{T}} &= p(\text{Type-3})_{|\Lambda=\mathcal{T}} \\
1 - \left(\frac{e^{\mathcal{T}}}{1 + e^{\mathcal{T}}} \right)^{\bar{d}_c} &= \left(\frac{e^{\mathcal{T}}}{1 + e^{\mathcal{T}}} \right)^{\bar{d}_c} \\
\mathcal{T} &= \ln \left(\frac{\left(\frac{1}{2}\right)^{1/\bar{d}_c}}{1 - \left(\frac{1}{2}\right)^{1/\bar{d}_c}} \right)
\end{aligned} \tag{6.15}$$

Using the above expression, we plot the reliability threshold $\mathcal{T}$ as a function of average check node degree $\bar{d}_c$ in Fig. 6.8. We can notice that the value of $\mathcal{T}$ is an

FIGURE 6.7: Check-node-type probability $p(\text{Type-2})$ and $p(\text{Type-3})$, plotted against the variable node log-likelihood-ratio (Λ) for check node degree $\bar{d}_c = 10$, $\bar{d}_c = 20$ and $\bar{d}_c = 30$.

increasing function of $\bar{d}_c$. Thus, given a fixed value of $\bar{d}_c$ for a particular LDPC code, we choose the reliability threshold $\mathcal{T}$ according to (6.15) such that the two type of check nodes will have an equal opportunity to participate in the next decoding iteration. In general, there will be a larger value of threshold $\mathcal{T}$ for high rate LDPC codes as compared to low rate LDPC codes. This is because the underlying code graphs for high rate codes tend to have a large check node degree $\bar{d}_c$.

6.4.3 Avoiding Code Graph Cycles

As discussed earlier, the BP algorithm operates on the basic assumption that the incoming message updates at every check and variable node are statistically independent. If the code graph is cycle-free, throughout the iteration process, all the incoming message updates remain independent of each other. However, in code graph with cycles, such as the case of LDPC code, dependencies are created

FIGURE 6.8: Illustration of reliability threshold ($\mathcal{T}$) versus the average check node degree ($\bar{d}_c$): plotted using (6.15).

particularly when the number of iterations is large. To be precise, for a graph cycle of length s , the incoming message update becomes correlated after $s/2$ iterations [100, 101]. In this situation, the conventional BP algorithm becomes sub-optimal and consequently yields inferior error-rate performance. In contrast, the proposed edge-based scheduling helps to maintain the message independence assumption by precluding (delaying) the occurrence of some graph cycles which leads to an improved error-rate performance.

Example 6.3. *This example describes the effect of code graph cycles encountered during BP decoding and explains how the proposed edge-based scheduling avoids the occurrence of such graph cycles. The code graph shown in Fig. 6.9 induces a cycle of length $s = 6$ at variable node v_1 , given by $v_1 \leftrightarrow c_1 \leftrightarrow v_2 \leftrightarrow c_2 \leftrightarrow v_4 \leftrightarrow c_4 \leftrightarrow v_1$. Here, the intrinsic information of v_1 is propagated towards check node c_1 and c_4 . After 3 decoding iterations, the same correlated information is passed back to v_1 through this graph cycle. Subsequently, all the incoming message updates at v_1 become statistically dependent. In contrast, as the message flow is constrained under the proposed edge-based scheduling, the BP decoder may skip the update operation along certain edges of the code graph. Consequently, some of the graph nodes may not receive the correlated information after $s/2$ iterations, as illustrated in Fig. 6.9. Here, it is assumed that the check node c_2 belongs to Type-2 and the neighboring edge $\vec{e}_{2,2}$ has achieved high reliability ($\varepsilon_{2,2} = 0$). In this situation, the graph edge $\vec{e}_{2,2}$*

FIGURE 6.9: Code graph cycle in BP decoding: the conventional BP schedule receives correlated message update at variable node v_1 after 3 iterations via cycle $v_1 \leftrightarrow c_1 \leftrightarrow v_2 \leftrightarrow c_2 \leftrightarrow v_4 \leftrightarrow c_4 \leftrightarrow v_1$; the proposed edge-based BP schedule avoids the correlated message by skipping update operation along edge $\vec{e}_{2,2}$.

will not be updated in the subsequent iteration. As a result, the intrinsic information of v_1 received at v_2 , via $v_1 \rightarrow c_1 \rightarrow v_2$, will not be propagated beyond v_2 . Meanwhile, the intrinsic information of v_1 received at c_2 , via $v_1 \rightarrow c_4 \rightarrow v_4 \rightarrow c_2$, will not be propagated beyond c_2 . Consequently, node v_1 will not receive the correlated information either from check node c_1 or from check node c_4 after 3 decoding iterations, thus protecting the message independence assumption at v_1 .

6.5 Dynamic Silent-Variable-Node-Free Scheduling (D-SVNFS)

Our previous scheduling schemes mainly focus on reducing the complexity of BP decoder by modifying the order of variable node processing, such as in CWS and IFS schemes, or by updating the partial set of edges in the code graph, as in e-Shuffled schedule. In this section, we propose an improved check-to-variable residual-based dynamic silent-variable-node-free scheduling (D-SVNFS) scheme which is inspired by the SVNFS scheme [105]. The residual-based dynamic schedules are known for their capability to overcome the trapping sets encountered during BP decoding [102]. In the C2V based dynamic schedules, BP decoder computes the message residuals which represent the absolute difference between the C2V message before and after an update, given by

$$r(c_m \rightarrow v_n) = ||m_{c_m \rightarrow v_n}^{\text{pre}} - m_{c_m \rightarrow v_n}|| \quad (6.16)$$

where $m_{c_m \rightarrow v_n}^{\text{pre}}$ is the pre-computed C2V message, which is calculated using (6.2). This message is not propagated along the edge $c_m \rightarrow v_n$, but only used for selecting the next updating edge. The message propagation takes place between check node c_m and variable node v_n that corresponds to the largest residual $r(c_m \rightarrow v_n)$. Among the existing dynamic schedules, SVNFS [105] provides the best error-rate performance by enforcing an equal variable node participation. Furthermore, SVNFS scheme also eliminates the artificial decoding loops which are commonly observed in other residual-based dynamic BP schedules.

Although SVNFS eliminates the silent (non participating) variable nodes, it does not specify the order of variable node participation. In fact, it only follows a fixed sequential order in each subsequent round of participation. However, it should be noted that a well-planned scheduling order can improve the decoding

convergence. In particular, it has been shown that the updated variable nodes carry more reliable information, and incorporating their message updates can expedite the decoding process [74, 94]. This is how the variable node based sequential schedules achieve faster convergence than the flooding schedule. With this understanding, we propose to allow the last updated variable node to participate in the decoding process in the D-SVNFS scheme. This criterion ensures that the decoding process incorporates reliable message updates stemming from already updated variable nodes in a timely manner. The proposed D-SVNFS scheme also preserves the feature of silent-variable-node-free decoding by allowing each variable node to contribute only once in every single round of participation. To make this happen, it uses a binary vector $\mathbf{u} = \{u_n, 1 \leq n \leq N\}$ to mark the participation status. Whenever a variable node is selected, say v_p , its corresponding term in vector $\mathbf{u}$ is set to 1 ($u_p = 1$). Given a single round of participation, if an already participated variable node is selected twice, the D-SVNFS decoder preempts its participation and finds a new variable node who has yet to participate. On the completion of each participation round, the vector $\mathbf{u}$ is reset to zero to clear the participation status. This approach not only incorporates the latest message updates into the decoding process, but also eliminates any possible occurrence of artificial decoding loops.

Furthermore, the proposed D-SVNFS scheme strives to correct the potentially erroneous variable nodes on a priority basis by allowing unsatisfied check nodes to forward new message updates. In this process, given a selected variable node v_p , only its unsatisfied (if any) neighbor check nodes are short-listed for the next C2V message propagation based on the largest C2V residual. This step expedites the convergence speed as the unreliable variable nodes (connected with unsatisfied check nodes) will receive new message updates more often.

To summarize the variable node selection method, if the last updated variable node, say v_p , is eligible for participation (i.e. $u_p = 0$), the next C2V message is

selected from the unsatisfied neighbor check nodes of v_p . Otherwise, the next eligible variable node is chosen for decoding participation. Targeting the updated variable nodes for message participation and unsatisfied check nodes for message propagation makes a significant impact on the error-rate convergence of BP decoder, as shown in the following section. The detailed steps of the proposed D-SVNFS scheme are outlined in Algorithm 7.

Example 6.4. *Fig. 6.10 shows the execution of D-SVNFS scheme over a simple code graph. Here, it is assumed that the check node equations corresponding to c_2 , c_3 and c_5 are unsatisfied ($s_{c_2} = 1$, $s_{c_3} = 1$, $s_{c_5} = 1$) due to the incorrect variable node v_5 . Decoding starts with the participation of variable node v_1 , as shown in Fig. 6.10(a). In this case, only residuals from the unsatisfied check node c_2 are computed on-demand and used for comparison. Based on the largest residual $r(c_2 \rightarrow v_5)$, C2V message $m_{c_2 \rightarrow v_5}$, and V2C messages $m_{v_5 \rightarrow c_3}$ and $m_{v_5 \rightarrow c_5}$ are propagated. Next, the recently updated variable node v_5 is selected. After computing the required (new) on-demand residuals from the unsatisfied check nodes c_2 , c_3 and c_5 , and finding the largest one among them, the C2V and V2C message updates are propagated accordingly, as shown in Fig. 6.10(b). Similarly, the participation of variable node v_6 , being the last updated node, follows next, as shown in Fig. 6.10(c). Following the participation of v_6 , the last updated variable node v_5 is found to be ineligible for participation ($u_5 = 1$). Therefore, the next available variable node, v_2 in this case ($u_2 = 0$), is given the opportunity to participate, as shown in Fig. 6.10(d). First round of participation ends when all the variable nodes contribute in the decoding process. In contrast, the SVNFS [105] schedule will follow a fixed sequence of participation, given by: $v_1 \Rightarrow v_2 \Rightarrow v_3 \Rightarrow v_4 \Rightarrow v_5 \Rightarrow v_6 \Rightarrow v_7$.*

Algorithm 7 : Dynamic Silent-Variable-Node-Free Scheduling (D-SVNFS)

```

1: Initialize all  $m_{v_n \rightarrow c_m} = L_{v_n}$ 
2:  $m_{c_m \rightarrow v_n} = 0$ 
3: while stopping rule is not satisfied do
4:   if all  $\{u_n = 1, 1 \leq n \leq N\}$  then
5:     Reset  $\{u_n = 0, 1 \leq n \leq N\}$ 
6:   end if
7:   Select  $v_p : u_p = 0, u_i = 1 \forall i < p$ 
8:   Set  $u_p = 1$ 
9:   if some  $s_{c_m} = 1, c_m \in \mathcal{M}(v_p)$  then
10:    for  $c_m \in \mathcal{M}(v_p), s_{c_m} = 1, v_n \in \mathcal{N}(c_m) \setminus v_p$  do
11:      Compute  $m_{c_m \rightarrow v_n}^{\text{pre}}, r(c_m \rightarrow v_n)$  with (6.2), (6.16)
12:    end for
13:    Find  $r(c_i \rightarrow v_j) = \max\{r(c_m \rightarrow v_n) \mid$ 
       $c_m \in \mathcal{M}(v_p), s_{c_m} = 1, v_n \in \mathcal{N}(c_m) \setminus v_p\}$ 
14:  else
15:    for  $c_m \in \mathcal{M}(v_p), v_n \in \mathcal{N}(c_m) \setminus v_p$  do
16:      Compute  $m_{c_m \rightarrow v_n}^{\text{pre}}, r(c_m \rightarrow v_n)$  with (6.2), (6.16)
17:    end for
18:    Find  $r(c_i \rightarrow v_j) = \max\{r(c_m \rightarrow v_n) \mid$ 
       $c_m \in \mathcal{M}(v_p), v_n \in \mathcal{N}(c_m) \setminus v_p\}$ 
19:  end if
20:  Generate and propagate  $m_{c_i \rightarrow v_j}$  with (6.2)
21:  Set  $r(c_i \rightarrow v_j) = 0$ 
22:  for  $c_a \in \mathcal{M}(v_j) \setminus c_i$  do
23:    Generate and propagate  $v_{v_j \rightarrow c_a}$  with (6.1)
24:  end for
25:  if  $u_j = 1$  then
26:    Proceed to 4
27:  else
28:    Set  $v_p = v_j, u_p = 1$ 
29:    Proceed to 9
30:  end if
31: end while

```

6.6 Simulation Results

In this section, we analyze the decoding complexity, error-rate performance and decoding convergence speed of BP algorithm of the proposed CWS, IFS, e-Shuffled and D-SVNFS schedules. We consider two high-rate binary LDPC codes. The first code is the (4544, 4090), rate-0.90 irregular LDPC code, as given in Chapter 3. The second code is the Gallager's (1998, 1776), rate-0.89 regular LDPC code obtained

FIGURE 6.10: BP decoding by using the proposed *dynamic silent-variable-node-free scheduling* (D-SVNFS): dynamic order of variable node participation (first round follows $v_1 \Rightarrow v_5 \Rightarrow v_6 \Rightarrow v_2$ and so on).

from Mackay's LDPC database which is available online at [111]. We refer to the first and second LDPC codes as 4K-code and 2K-code, respectively. We decode the 2K-code by using the e-Shuffled and D-SVNFS schemes, and the 4K-code by using the CWS and IFS schemes. We numerically optimize the value of t (number of minimum-reliability neighbors) for e-Shuffled schedule to be $t = 5$ for both LDPC codes. We set threshold level $\mathcal{T}$ according to the check node degree $\bar{d}_c$ by using (6.15). Throughout this section, we assume that the binary LDPC code-word bits, $a_n \in \{0, 1\}$ for $1 \leq n \leq N$, are stored over the 2-bit per cell NAND flash memory channel, and these bits are, therefore, subject to various channel noise effects including programming noise, CCI, RTN and data retention noise. These code-word

bits are then retrieved by using 6-level non-uniform memory sensing quantization. The corresponding read-back voltage signal v is converted into LLR information L_{v_n} by using (3.14) and (3.15) which is then supplied to the BP algorithm for the initialization of variable nodes.

6.6.1 Average Decoding Complexity

The BP decoding complexity is measured as the total number of computations that are performed up to the error-rate convergence. It includes the variable-to-check (V2C) and check-to-variable (C2V) complexity, denoted by $\mathcal{V}$ and $\mathcal{C}$, respectively, and the scheduling complexity. In Table 6.2, we report a detailed complexity analysis of the proposed scheduling schemes. Note that the CWS and IFS schemes do not require run-time scheduling computations as they follow a fixed order of message propagation. Thus, we can write the corresponding V2C and C2V complexity as

$$\mathcal{V}_{\text{CWS}} = \bar{d}_v k$$

$$\mathcal{C}_{\text{CWS}} = k$$

$$\mathcal{V}_{\text{IFS}} = \bar{d}_v k$$

$$\mathcal{C}_{\text{IFS}} = k$$

where k is the average number of V2C/C2V update operations computed per decoded code-word to attain the error-rate convergence. Here, the V2C complexity is written in terms of the equivalent addition operations as each V2C update operation requires $\bar{d}_v$ additions. This kind of representation facilitates in making a fair complexity comparison with other BP schedules. It should be noted that the C2V complexity dominates the overall BP decoding complexity as it involves multiplication and $\tanh(\cdot)$ operations.

In contrast, the dynamic e-Shuffled and D-SVNFS schemes do require additional run-time scheduling computations. The scheduling complexity for the

proposed e-Shuffled schedule, denoted here as ψ , comprises of real value comparisons that are required to identify the set of t minimum-reliability neighbors, $\mathcal{N}_{t-\min}(c_m)$, corresponding to all the satisfied check node equations $s_{c_m} = 0$. The value of ψ can be approximately computed by

$$\psi \approx M \left[\frac{1}{2} + \frac{1}{2} \prod_{i=1}^{\bar{d}_c} \left(1 - \frac{2e^{\mathcal{T}}}{1 + e^{\mathcal{T}}} \right) \right] [t(\bar{d}_c - 1) + 1] z \quad (6.17)$$

In order to define a complete complexity cost, it is important to incorporate the run-time scheduling complexity in the overall decoding complexity. To this end, we notice that a real value comparison can be realized in hardware by using a full-adder circuit [112], thus it has an equivalent complexity as that of an addition operation. In this perspective, we include the scheduling complexity ψ as part of the V2C update operation (which already represents addition operations), and write the overall V2C and C2V complexity for the proposed e-Shuffled schedule as

$$\mathcal{V}_{\text{e-Shuffled}} = \bar{d}_v k + \psi$$

$$\mathcal{C}_{\text{e-Shuffled}} = k$$

In a similar manner, we can define the equivalent V2C complexity cost for the proposed D-SVNFS scheme by incorporating the real value comparison operations in the V2C complexity cost, given by

$$\mathcal{V}_{\text{D-SVNFS}} = \bar{d}_v (\bar{d}_v - 1) k + [\bar{d}_v (\bar{d}_c - 1) - 1] k$$

$$\mathcal{C}_{\text{D-SVNFS}} = k$$

Apart from real comparisons, the D-SVNFS scheme also incurs additional operations associated with C2V message residual computations. We compare the decoding complexity of the proposed schedules with that of the conventional shuffled schedule. In order to demonstrate the complexity reduction that the proposed BP schedules offer over the existing schedule, we normalize the V2C and C2V complexity

TABLE 6.2: Complexity analysis of fixed and dynamic scheduling schemes (per iteration)

Schedule	V2C update	C2V update	Message Residual	Real-value Comparison
Shuffled (prior-art)	k	k	0	0
CWS	k	k	0	0
IFS	k	k	0	0
e-Shuffled	k	k	0	ψ
D-SVNFS	$(\bar{d}_v - 1) k$	k	$\approx (\bar{d}_v - 1) (\bar{d}_c - 1) k$	$\approx [\bar{d}_v (\bar{d}_c - 1) - 1] k$

 $\bar{d}_v \Rightarrow$ average variable node degree $\bar{d}_c \Rightarrow$ average check node degree

of the proposed schedules. To be precise, we simulate the LDPC codes under different schedules until the error rates converge, or up to a maximum count of 20 iterations ($I_{\max} = 20$) and plot the ratio $\frac{\mathcal{V}_{\text{proposed}}}{\mathcal{V}_{\text{shuffled}}}$ and $\frac{\mathcal{C}_{\text{proposed}}}{\mathcal{C}_{\text{shuffled}}}$, as shown in Fig. 6.11 and 6.12. This normalization method reveals an exact amount of complexity reduction achieved by employing the proposed schemes. It can be observed that the average V2C and C2V complexity is significantly reduced compared to the conventional shuffled schedule. For instance, for 4K-code at PE = 10K, the proposed CWS and IFS schedules offer V2C and C2V complexity reduction by up to 30%. Similarly, for 2K-code at PE = 10K, the proposed e-Shuffled schedule reduces the V2C and C2V complexity cost by up to 30% and 75%, respectively, compared to the conventional shuffled schedule. The proposed D-SVNFS also offers huge C2V complexity reduction, however, it induces more V2C and residual computation cost.

6.6.2 Error-Rate and Decoding Convergence

As the BP algorithm is a sub-optimal decoding algorithm for finite-length block codes due to code graph cycles, the message-passing scheduling technique not only influences the decoding complexity but also affects the error-rate performance of BP decoder. To demonstrate the improvement in the error-rate performance that the

FIGURE 6.11: Average variable-to-check (V2C) decoding complexity ($\mathcal{V}$) of the proposed CWS, IFS, e-Shuffled and D-SVNFS schedules, compared (normalized) w.r.t. conventional shuffled schedule.

FIGURE 6.12: Average check-to-variable (C2V) decoding complexity ($\mathcal{C}$) of the proposed CWS, IFS, e-Shuffled and D-SVNFS schedules, compared (normalized) w.r.t. conventional shuffled schedule.

proposed schedules can provide over the conventional shuffled schedule, we plot the frame-error-rate (FER) curves against the memory PE cycles. All the FER curves are plotted at $I_{\max} = 5$ and $I_{\max} = 20$. In Fig. 6.13, we show the FER performance of the proposed CWS and IFS schedules and compare with the conventional shuffled schedule. It can be observed that the proposed schedules outperform

the prior-art shuffled schedule for the initial decoding iterations. In Fig. 6.14, we compare the FER performance of the proposed e-Shuffled and D-SVNFS schedules with shuffled schedule. As can be seen, the proposed e-Shuffled and D-SVNFS schedules outperform the existing shuffled schedule, irrespective of the iteration count. For instance, the D-SVNFS scheme provides FER improvement by up to 2 *orders-of-magnitude* at $I_{\max} = 5$ and PE = 17K cycles.

Arising from the lower decoding complexity, the proposed schedules also provide faster decoding convergence, which determines the speed at which the decoder attains error-rate convergence. To show that, in Fig. 6.15 and 6.16, we plot the FER performance of different schedules by varying the maximum number of decoder iteration ($I_{\max}$), keeping the PE cycles fixed at PE = 18K and PE = 19K for 2K-code and 4K-code, respectively. Note that a single decoder iteration is comprised of E C2V update operations, where E denotes the total number of edges in the code graph (i.e. $E = M \times \bar{d}_c$ or $E = N \times \bar{d}_v$). It can be observed that the proposed schedules, specifically the dynamic e-Shuffled and D-SVNFS schemes, achieve faster decoding convergence and thus well-outperform the existing shuffled schedule. In Fig. 6.17, we show the convergence performance for 4K-code by employing all the proposed scheduling schemes. Among all, the D-SVNFS has the fastest convergence rate.

6.6.3 Semi-Serial Scheduling by employing CWS, IFS and e-Shuffled Schemes

In addition to fully sequential order variable node processing, the proposed CWS, IFS and e-Shuffled schedules can be invoked under semi-serial scheduling in order to trade between the decoding convergence speed and the measured decoding throughput. In other words, the decoder's hardware throughput can be increased at the expense of lower error-rate convergence speed, and vice versa, by implementing

FIGURE 6.13: Frame-error-rate (FER) performance versus memory PE cycles of LDPC 4K-code by employing the proposed CWS and IFS schedules, and compared with conventional shuffled schedule: simulated at $I_{\max} = 5$ (solid curves) and $I_{\max} = 20$ (dashed curves).

FIGURE 6.14: Frame-error-rate (FER) performance versus memory PE cycles of LDPC 2K-code by employing the proposed e-Shuffled and D-SVNFS schedules, and compared with conventional shuffled schedule: simulated at $I_{\max} = 5$ (solid curves) and $I_{\max} = 20$ (dashed curves).

the semi-serial scheduling. In this chapter, we only evaluate the performance of the proposed IFS scheme under the semi-serial group-shuffled BP decoding scheme, whereas the proposed CWS and e-Shuffled schemes can also be evaluated under the semi-serial group-shuffled decoding. The group-shuffled decoding scheme performs

FIGURE 6.15: Frame-error-rate (FER) versus iteration (I_{max}) of LDPC 4K-code by employing the proposed CWS and IFS schedules, and conventional shuffled schedule.

FIGURE 6.16: Frame-error-rate (FER) versus iteration (I_{max}) of LDPC 2K-code by employing the proposed e-Shuffled and D-SVNFS schedules, and conventional shuffled schedule.

parallel sub-matrix processing in which, instead of single variable node update, p variable nodes are simultaneously updated, where p is any arbitrary group size. To cater for group-shuffled decoding, the IFS algorithm should be modified according to the group size p , such that p maximum variable counters will be selected in Step 4 of Algorithm 5. This will be followed by incrementing all the relevant variable and

FIGURE 6.17: Frame-error-rate (FER) versus iteration (I_{max}) of LDPC 4K-code by employing the proposed CWS and IFS, e-Shuffled and D-SVNFS schedules, and conventional shuffled schedule.

check counters. This process will continue until all the variable nodes are scheduled.

In Fig. 6.18, we plot the normalized complexity for 4K-code for different values of p by employing the IFS scheme. Note that the curves in Fig. 6.18 are normalized w.r.t. the group-shuffled decoding by employing the conventional scheduling with same group size p . These results again show that the IFS scheme provides significant complexity reduction under the group-shuffled decoding, specifically for smaller values of p . It is pertinent to mention that the variable node grouping can also be done on the basis of code structure, instead of random grouping with arbitrary size p , and similar performance improvement can be realized.

6.7 Combining Voltage Optimization and Scheduling Schemes

We can further improve the error-rate performance by integrating the techniques presented in Chapter 3, 4, 5 with the proposed scheduling schemes presented in Chapter 6. For instance, in Fig. 6.19, we show the performance of e-Shuffled

FIGURE 6.18: Average check-to-variable (C2V) / variable-to-check (V2C) complexity of LDPC 4K-code of the proposed IFS schedule employing group shuffled decoding with size p , normalized by conventional shuffled schedule.

FIGURE 6.19: Frame-error-rate (FER) performance versus memory PE cycles of LDPC 2K-code by employing the conventional shuffled schedule, and compared with the proposed e-Shuffled schedule when used in conjunction with write voltage optimization method: simulated at $I_{\max} = 20$.

scheduling in conjunction with the write voltage optimization method of Chapter 3. We can observe that the error-rate performance is further improved by combining the two proposed schemes.

6.8 Chapter Summary

Noting the importance of decoding schedule on the complexity, convergence speed and error-rate performance of iterative BP decoder, a novel shuffled BP decoding schedule based on the column weight (CWS) of variable nodes is presented for irregular LDPC codes. Specifically, we propose to update the high column-weight variable nodes first, followed by the low column-weight variable nodes. This schedule requires sorting of variable nodes in descending order of their column-weights only once in the offline mode, eliminating the need for run-time computations. Using numerical simulations, a noticeable improvement in the convergence behavior of irregular LDPC codes is observed under the proposed CWS scheme.

An improved version of CWS scheme is introduced in IFS scheme which offers faster decoding convergence for both regular and irregular LDPC codes. In the IFS scheme, a fixed sequence of variable nodes is determined based on the maximum number of newly available message updates. Again, the variable nodes are scheduled only once in the off-line mode, precluding the additional run-time computations. This sequence of variable nodes is then serially/semi-serially decoded under the shuffled BP algorithms. Simulation results show complexity reduction of up to 30% by employing the proposed CWS and IFS schemes, compared to the conventional shuffled [74] schedule. Moreover, the error performance improves for the initial decoding iterations and remains comparable to the conventional shuffled BP decoding scheme for subsequent decoding iterations.

To further reduce the BP decoding complexity and simultaneously improve on the error-rate performance of the CWS and IFS schemes which update all the edges of the code graph per iteration, a novel e-Shuffled scheduling strategy is proposed which updates only a partial set of code graph edges. In the e-Shuffled schedule, the reliability of individual graph edges is monitored and new message updates are propagated only along the less reliable edges in the code graph. Simulation results show that the proposed e-Shuffled schedule outperforms the shuffled schedule

in terms of decoding complexity, convergence speed and error-rate performance altogether as the measure of effectiveness.

Finally, if the LDPC decoder can afford some run-time computation, a C2V residual-based D-SVNFS scheme is introduced in which the last updated variable and check nodes are allowed to participate in the decoding process, ensuring that more reliable updates are exchanged between the variable and check nodes. Meanwhile, to prioritize the correction of erroneous variable nodes, the unsatisfied check nodes are enforced to propagate the next message update towards the likely erroneous variable nodes. Simulation results show significant improvement in the error-rate performance of up to 2 *orders-of-magnitude*.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

To improve the data reliability of NAND flash memory, this thesis develops a variety of coding and signal processing solutions that can tolerate different flash channel noise and interference effects. Considering a 2-bit per cell NAND flash channel model that incorporates all the major threshold voltage impairments, an optimization method for selecting the best read and write voltage levels to adapt to the non-stationary NAND flash channel is first proposed. In this scheme, the write voltage levels are optimized in an off-line mode and adjusted on-the-fly such that the flash channel raw error probability is minimized. The proposed write voltage optimization not only prolongs the program/erase (PE) endurance, and thus the service lifetime of NAND flash memory, but also improves the LDPC decoding convergence speed. In addition, a novel read voltage quantization scheme based on a voltage entropy function is proposed to attain an appropriate balance between the flash channel symbol-error and symbol-erasure probabilities by controlled erasure decoding. This quantization scheme enables the LDPC decoder to achieve soft BP decoding performance improvement over the best-known prior-art MMI quantization scheme [70], without incurring any additional overhead. The main reason

behind the performance improvement is that the proposed scheme minimizes the post-decoding error rate while the MMI scheme only minimizes the raw channel error rate.

Next, the thesis investigates the data retention problem in NAND flash memory that is caused by the leakage of electrons through the memory cell's floating gate transistor, and introduces a retention-aware belief propagation (RABP) decoding strategy for LDPC codes to recover the retention noise-induced errors. Unlike the prior-art schemes which rely upon the time- and power-consuming read retries for data recovery [36, 37], the proposed RABP decoder innovatively modifies the LLR computation by re-mapping the LLR values of different quantization regions, leveraging on the decoded code-word bits and the read-back voltage signal, and initiates a second round of BP decoding. The proposed RABP decoder increases the flash channel tolerance against the retention noise-induced errors by up to 70% and improves the asymptotic error-rate performance by up to 3 *orders-of-magnitude*, when compared to a single round of BP decoding. To further improve the retention time endurance, an RABP assisted channel update (RA-CU) algorithm is proposed to re-estimate the cell voltage mean and standard deviation by using the output of RABP decoder, while avoiding new memory sensing operations. With RA-CU, the resultant average estimation errors for the mean and standard deviation fall below 0.02 V over increasing retention time. Correspondingly, the retention time tolerance of flash channel by up to $41\times$ (times) at 6K PE cycles, compared to a system which does not update the voltage distribution parameters and simply employs the initial values provided at source by the memory chip supplier.

To mitigate the cell-to-cell interference (CCI) noise arising from the parasitic coupling-capacitance between adjacent memory cells, this thesis presents three simple yet effective post-processing solutions. Exploiting the stationary and neighbor data-dependent characteristics of CCI, the proposed channel detection schemes estimate the overall CCI strength by using the cell voltage of interfering (neigh-

boring) memory cells. This estimated CCI is subsequently compensated from the read-back voltage signal of victim cell. In particular, the proposed detection schemes systematically remove the CCI such that the odd bit-line cells are cleaned first, providing some a-priori information for the even bit-line cells. This is followed by the cancellation of CCI in the even bit-line cells. This systematic CCI cancellation approach facilitates more accurate estimation of CCI component, which is corroborated via numerical simulations. It is shown that the proposed detection schemes yield higher symmetric information rate (cell storage capacity) and superior error-rate performance, compared to the prior-art Post-Comp detector [42] which performs CCI cancellation without cleaning the interfering cells, and Dong's detector [28] which estimates the CCI affected voltage distribution functions without removing the CCI component. Meanwhile, the complexity and design trade-off analysis reveals that only a small increase of linear order in total computations is introduced by the proposed detection schemes.

As the LDPC codes are increasingly adopted as the error correction codes for the NAND flash-based data storage systems, their long iterative belief-propagation (BP) decoding latency starts to deteriorate the system read response time. To manage this challenge, this thesis explores the design of some improved fixed and dynamic BP message-passing scheduling algorithms. The first scheduling algorithm performs shuffled BP decoding [74] by employing a pre-determined high-to-low sequential column-weight (CWS) order. The proposed CWS algorithm works very well for irregular LDPC codes by offering faster decoding convergence, but yields no improvements for regular LDPC codes as they have uniform column weights in the parity check matrix. To overcome this limitation, an improved informed fixed scheduling (IFS) algorithm is proposed which maximizes the utilization of newly available message updates within every single decoding iteration. The IFS algorithm is shown to accelerate the BP convergence speed for both regular and irregular LDPC codes when compared to the CWS algorithm, without affecting the

error-rate performance or introducing additional run-time scheduling complexity.

Unlike the CWS and IFS algorithms, where each graph edge is updated exactly once per iteration, a new edge reliability-based shuffled scheduling algorithm, e-Shuffled schedule in short, is proposed which constrains the flow of message updates by updating only a selected subset of code graph edges. This is the first partial edge-updating schedule ever proposed. Since only a partial set of edges are updated in each iteration, the total updated message count, and consequently the overall decoding complexity required for error-rate convergence, is significantly reduced. As an additional benefit, the adverse effect of short code graph cycles is also diminished, leading to a better asymptotic error-rate performance. Finally, a check-to-variable (C2V) residual-based dynamic silent-variable-node-free scheduling (D-SVNFS) algorithm is presented which propagates new message update along a graph edge that corresponds to the largest C2V residual and is simultaneously connected with the recently updated check and variable nodes. Moreover, the proposed D-SVNFS algorithm strives to propagate more message updates towards the likely incorrect variable nodes, forcing them to correct their hard-decision with higher priority. This is not done by the prior-art schemes.

Numerical results show that the proposed CWS and IFS fixed scheduling schemes reduce the BP decoding complexity by up to 30%, compared to the conventional shuffled schedule [74]. They also exhibit error-rate performance improvement in the initial decoding iterations, and comparable performance in the subsequent decoding iterations. The proposed e-Shuffled dynamic scheduling scheme offers complexity reduction of 60% to 70% and improves the asymptotic error-rate performance. Last but not least, the proposed D-SVNFS shows improvement in the error-rate performance by up to 2 *orders-of-magnitude*, at the price of some increase in the decoding complexity due to residual computations.

7.2 Future Work

We have introduced several new system-level fault tolerant solutions to improve the reliability of NAND flash-based data storage system, such as the read/write voltage optimization in Chapter 3, the retention-aware belief-propagation decoding for retention-failure recovery in Chapter 4, the neighbor-a-priori detection schemes for CCI cancellation in Chapter 5, and some improved belief-propagation scheduling schemes in Chapter 6. We finally describe a few more research directions that could be extended in the future.

1. The NAND flash channel model presented in Chapter 2 can be made more comprehensive by including the effect of other noise sources. For instance, the influence of program/read disturb noise, which induces a weak programming effect by injecting unwilling electrons through the cell's tunnel oxide in the floating gate [113, 114], can be encapsulated into the flash channel. Besides the noise induced soft errors, NAND flash memory also experiences permanent hard errors due to cell defects [115, 116]. Thus, modeling the transient soft errors and permanent hard errors together into the NAND flash channel can be an important research direction. Furthermore, it is highly desired to apply the proposed signal processing schemes over a real NAND flash device.
2. The Gaussian mixture model of the threshold voltage distribution does not capture the asymmetric tails observed in real NAND flash memories. It would be useful to evaluate the proposed techniques using something like the Normal-Laplace model studied in recent papers, e.g., [117], and to explore ways to modify the proposed techniques to suit these other channel models.
3. The thesis is limited to 2-bits per cell (MLC) flash memory. Flash memory with 3-bits per cell (TLC) is already available commercially and will likely become the dominant technology. Memories with 4-bits per cell (QLC) will no doubt

appear in the not too distant future. It would be interesting to determine the applicability of the proposed techniques to such newer technologies, and to see if similar conclusions about their effectiveness and practicality still apply.

4. The wear induced by program-erase cycling is known to be cell-level dependent. That is, repeated programming of certain levels, typically higher levels, induces more cell wear. It would be interesting to incorporate this into the channel model.
5. The proposed neighbor-a-priori detection schemes presented in Chapter 5, as well as the prior-art CCI cancellation schemes reported in [42–45, 90, 91], inevitably employ uniform memory sensing to obtain an accurate estimate of the CCI. However, NAND flash-based systems experience a large memory sensing and memory-to-controller data transfer latency by applying the fine-grained uniform memory sensing. Instead, non-uniform memory sensing schemes, such as the proposed entropy quantization or prior-art MMI [70] quantization, have been shown to reduce these latency costs. In this context, a hybrid (uniform plus non-uniform) quantization strategy can be devised by combining the signal pre-compensation [42] and the proposed post-processing detection schemes. Note that the pre-compensation method can only handle CCI for the programmed-state cells but is not effective for erased-state cells. This observation can be exploited by allowing the signal pre-compensation to partially mitigate the CCI, while employing post-processing to remove the CCI from the erased-state cells. This will facilitate the implementation of non-uniform quantization for the programmed-state cells, which will not require CCI cancellation due to pre-compensated voltage levels, and uniform quantization for the erased-state cells which will require CCI cancellation via the proposed detection schemes. This hybrid quantization scheme will help in reducing the total memory sensing and data transfer latency to the limits.

6. Studies of cell-level configurations in neighboring cells have shown that certain patterns of neighboring cell-levels are much more likely to cause errors in victim cells. It would be interesting to explore whether CCI cancellation techniques could exploit this to reduce complexity or improve performance.
7. In current design practice, it is assumed that only the adjacent neighbor cells induce the CCI effect. However, due to further node scaling, future flash memory chips may experience the CCI effect not only from the immediate neighbors but also from the distant neighbor cells. In that situation, the proposed detection schemes to successively clean the interfering cells first, and then to remove the CCI effect from the victim cell will become even more important. This can be an interesting topic for future investigation.
8. The proposed e-Shuffled scheduling strategy presented in Chapter 6 offers a significant reduction in BP decoding complexity. In e-Shuffled schedule, the selection of the number of minimum-reliability edges is based on numerical simulations. Possible future work includes an analytical determination of the set of minimum-reliability edges in the code graph to fully exploit the potential of the proposed e-Shuffled schedule. Furthermore, combining the proposed scheduling schemes with reduced check node complexity algorithms, such as the min-sum approximation [118] or iterative reliability-based one-step majority-logic decoding (OSMLGD) algorithm [119], is also an interesting topic for future research.

Author's Publications

Journal Papers

- [1] **C. A. Aslam**, Y. L. Guan, and K. Cai, "Improving the belief propagation convergence of irregular LDPC codes using column-weight based scheduling," *IEEE Commun. Lett.*, vol. 19, no. 8, pp. 1283-1286, Aug. 2015.
- [2] **C. A. Aslam**, Y. L. Guan, and K. Cai, "Read and write voltage signal optimization for multi-level-cell (MLC) NAND flash memory," *IEEE Trans. Commun.*, vol. 64, no. 4, pp. 1613-1623, April 2016.
- [3] **C. A. Aslam**, Y. L. Guan, and K. Cai, "Detector for MLC NAND flash memory using neighbor-a-priori information," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 9, pp. 2827-2836, Sep. 2016.
- [4] **C. A. Aslam**, Y. L. Guan, and K. Cai, "Edge-based dynamic scheduling for belief-propagation (BP) decoding of LDPC and RS codes," *IEEE Trans. Commun.*, vol. 65, no. 2, pp. 525-535, Feb. 2017.
- [5] **C. A. Aslam**, Y. L. Guan, and K. Cai, "Retention-aware belief-propagation decoding for NAND flash memory," in *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 64, no. 6, pp. 725-729, June 2017.
- [6] **C. A. Aslam**, Y. L. Guan, K. Cai, and G. Han, "Informed fixed scheduling for faster convergence of shuffled belief-propagation (BP) decoding," *IEEE Commun. Lett.*, vol. 21, no. 1, pp. 32-35, Jan. 2017.
- [7] **C. A. Aslam**, Y. L. Guan, K. Cai, and G. Han, "Low-complexity belief propagation (BP) decoding via dynamic silent-variable-node-free scheduling," *IEEE Commun. Lett.*, vol. 21, no. 1, pp. 28-31, Jan. 2017.
- [8] **C. A. Aslam**, K. Cai, and Y. L. Guan, "Mitigating stuck-cell failures in MLC NAND flash memory via inferred erasure decoding," accepted in *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*.
- [9] **C. A. Aslam**, Y. L. Guan, and K. Cai, "Decision-directed retention-failure recovery with channel update for MLC NAND flash memory," **under review (minor revision)** in *IEEE Trans. Circuits Syst. I, Reg. Papers*.

Conference Papers

- [1] **C. A. Aslam**, Y. L. Guan, and K. Cai, "Dynamic write-level and read-level signal design for MLC NAND flash memory," in *Proc. 9th Int. Symp. Commun. Syst., Netw., Digit. Signal Process. (CSNDSP)*, Manchester, UK, July 2014, pp. 336-341.
- [2] **C. A. Aslam**, Y. L. Guan, and K. Cai, "Non-binary LDPC code with multiple memory reads for multi-level-cell (MLC) flash," in *Proc. Annual Conf. Asia-Pacific Signal and Information Processing Association (APSIPA)*, Dec. 2014, pp. 1-9.
- [3] **C. A. Aslam**, Y. L. Guan, and K. Cai, "Low-complexity quantization-aware belief-propagation (QA-BP) decoding for MLC NAND flash memory," in *Proc. Int. Conf. Information, Communication and Signal Processing (ICICS)*, Singapore, Dec. 2015, pp. 1-5.
- [4] **C. A. Aslam**, K. Cai, and Y. L. Guan, "Edge-based scheduling for decoding non-binary LDPC codes for MLC flash memories," in *Non-volatile Memory Workshop (NVMW)*, San Diego, USA, March 2016.
- [5] **C. A. Aslam**, K. Cai, and Y. L. Guan, "On the Recovery of Retention-Failure Data in MLC NAND Flash Memory," in *Non-volatile Memory Workshop (NVMW)*, San Diego, USA, March 2017.

Bibliography

- [1] “Advantages of SSD over HDD,” available online at <http://www.computerhope.com/issues/ch001396.htm>.
- [2] “Technical note: NAND flash 101 introduction,” available online at https://www.micron.com/~media/documents/products/technical-note/nand-flash/tn2919_nand_101.pdf.
- [3] J. Choe, “Comparison of 20 nm and 10 nm-class 2D planar NAND and 3D V-NAND architecture,” 2015, available online at https://www.flashmemorysummit.com/English/Collaterals/Proceedings/2015/20150811_FM12-Choe.pdf.
- [4] “TLC NAND,” available online at <https://www.micron.com/products/nand-flash/tlc-nand>.
- [5] “Toshiba Announces Client SSD with Embedded TLC NAND Flash Memory,” available online at <https://www.samsung.com/global/business/semiconductor/minisite/SSD/M2M/html/ssd845dcevo/MlcNandFlash.html>.
- [6] “SSD 845DC EVO,” available online at <http://toshiba.semicon-storage.com/us/company/taec/news/2016/02/storage-20160216-1.html>.
- [7] N. Shibata *et al.*, “A 70 nm 16 Gb 16-Level-Cell NAND flash Memory,” *IEEE J. Solid-State Circuits*, vol. 43, no. 4, pp. 929–937, April 2008.
- [8] C. Trinh *et al.*, “A 5.6MB/s 64Gb 4b/cell NAND flash memory in 43nm CMOS,” in *Proc. IEEE ISSCC - Digest of Technical Papers*, Feb 2009, pp. 246–247, 247a.
- [9] J. S. Meena, S. M. Sze, U. Chand, and T.-Y. Tseng, “Overview of emerging nonvolatile memory technologies,” *Nanoscale Research Letters*, vol. 9, no. 1, p. 526, 2014.
- [10] N. Mielke *et al.*, “Bit error rate in NAND flash memories,” in *Proc. Int. Symp. Reliability Physics*. IEEE, 2008, pp. 9–19.
- [11] J. H. Yoon and G. A. Tressler, “Advanced flash technology status, scaling trends and implications to enterprise SSD technology enablement,” 2012, available online at http://www.flashmemorysummit.com/English/Collaterals/Proceedings/2012/20120821_TA12-Yoon-Tressler.pdf.

- [12] O. Mutlu, “Error analysis and management for MLC NAND flash memory,” 2014, available online at https://users.ece.cmu.edu/~omutlu/pub/flash-memory-errors_mutlu_fms14-talk.pdf.
- [13] C. M. Compagnoni *et al.*, “First evidence for injection statistics accuracy limitations in NAND flash constant-current Fowler-Nordheim programming,” in *IEEE IEDM*, Dec 2007, pp. 165–168.
- [14] E. Grochowski and R. E. Fontana Jr, “Future technology challenges for NAND flash and HDD products,” *Flash Memory Summit*, 2012.
- [15] J.-D. Lee, S.-H. Hur, and J.-D. Choi, “Effects of floating-gate interference on NAND flash memory cell operation,” *IEEE Electron Device Lett.*, vol. 23, no. 5, pp. 264–266, May 2002.
- [16] Y. Cai *et al.*, “Error patterns in MLC NAND flash memory: Measurement, characterization, and analysis,” in *Proc. Conf. Exhibition Design, Automation Test in Europe*, March 2012, pp. 521–526.
- [17] V. Taranalli, H. Uchikawa, and P. H. Siegel, “Error analysis and inter-cell interference mitigation in multi-level cell flash memories,” in *Proc. IEEE Int. Conf. Commun. (ICC)*, June 2015, pp. 271–276.
- [18] L. M. Grupp *et al.*, “Characterizing flash memory: anomalies, observations, and applications,” in *Proc. IEEE/ACM Annual Int. Symp. Microarchitecture (MICRO)*. IEEE, 2009, pp. 24–33.
- [19] “Technical note: Wear leveling in nand flash memory,” available online at <http://www.macronix.com/Lists/ApplicationNote/Attachments/1163/AN0289V1-Wear%20Leveling%20in%20NAND%20Flash%20Memory.pdf>.
- [20] “Technical note: Wear-leveling techniques in nand flash devices,” available online at https://www.micron.com/~media/documents/products/technical-note/nand-flash/tn2942_nand_wear_leveling.pdf.
- [21] Y.-H. Chang and L.-P. Chang, *Efficient Wear Leveling in NAND Flash Memory: Inside Solid State Drives (SSDs)*. Netherlands: Springer, 2013.
- [22] S. Li and T. Zhang, “Improving multi-level NAND flash memory storage reliability using concatenated BCH-TCM coding,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 10, pp. 1412–1420, Oct 2010.
- [23] C. Yang, Y. Emre, and C. Chakrabarti, “Product code schemes for error correction in MLC NAND flash memories,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 12, pp. 2302–2314, Dec 2012.
- [24] F. Sala, K. A. S. Immink, and L. Dolecek, “Error control schemes for modern flash memories: Solutions for flash deficiencies,” *IEEE Consum. Electron. Mag.*, vol. 4, no. 1, pp. 66–73, Jan 2015.

- [25] S. g. Cho and J. Ha, "Concatenated BCH codes for NAND flash memories," in *Proc. IEEE Int. Conf. Commun. (ICC)*, June 2012, pp. 2611–2616.
- [26] Y. Lee, H. Yoo, J. Jung, J. Jo, and I. C. Park, "A 2.74-pJ/bit, 17.7-Gb/s iterative concatenated-BCH decoder in 65-nm CMOS for NAND flash memory," *IEEE J. Solid-State Circuits*.
- [27] B. Chen, X. Zhang, and Z. Wang, "Error correction for multi-level NAND flash memory using Reed-Solomon codes," in *Proc. IEEE Workshop Signal Processing Systems*, Oct 2008, pp. 94–99.
- [28] G. Dong, N. Xie, and T. Zhang, "On the use of soft-decision error-correction codes in NAND flash memory," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 58, no. 2, pp. 429–439, Feb 2011.
- [29] J. Kim, Dong-hwan Lee, and W. Sung, "Performance of rate 0.96 (68254, 65536) EG-LDPC code for NAND flash memory error correction," in *Proc. IEEE Int. Conf. Commun.*, 2012, pp. 7029–7033.
- [30] W. Zhao, G. Dong, H. Sun, N. Zheng, and T. Zhang, "Reducing latency overhead caused by using LDPC codes in NAND flash memory," *EURASIP Journal on Advances in Signal Processing*, vol. 2012, no. 1, p. 203, 2012.
- [31] K. Zhao *et al.*, "LDPC-in-SSD: Making advanced error correction codes work effectively in solid state drives," in *Proc. 11th USENIX Conf. File and Storage Technol.*, ser. FAST'13, San Jose, CA, 2013, pp. 243–256.
- [32] C. A. Aslam, Y. L. Guan, and K. Cai, "Non-binary LDPC code with multiple memory reads for multi-level-cell (MLC) flash," in *Proc. Int. Conf. Asia-Pacific Signal and Information Processing Association (APSIPA)*, Dec 2014, pp. 1–9.
- [33] —, "Low-complexity quantization-aware belief-propagation (QA-BP) decoding for MLC NAND flash memory," in *Proc. Int. Conf. Information, Communication and Signal Processing (ICICS)*, Dec 2015, pp. 1–5.
- [34] S. Tanakamaru, Y. Yanagihara, and K. Takeuchi, "Error-prediction LDPC and error-recovery schemes for highly reliable solid-state drives (SSDs)," *IEEE J. Solid-State Circuits*, vol. 48, no. 11, pp. 2920–2933, Nov 2013.
- [35] J. Jeong, Y. Song, and J. Kim, "Flashdefibrillator: a data recovery technique for retention failures in NAND flash memory," in *Proc. IEEE Symp. Non-Volatile Memory System, Applications*, Aug 2015, pp. 1–6.
- [36] Y. Cai *et al.*, "Data retention in MLC NAND flash memory: Characterization, optimization, and recovery," in *Proc. IEEE 21st Int. Symp. High Performance Computer Architecture*, Feb 2015, pp. 551–563.
- [37] N. Papandreou *et al.*, "Enhancing the reliability of MLC NAND flash memory systems by read channel optimization," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 20, no. 4, pp. 62:1–62:24, Sep. 2015.

- [38] Y. Cai *et al.*, “Threshold voltage distribution in MLC NAND flash memory: Characterization, analysis, and modeling,” in *Proc. Conf. Design, Automation Test in Europe*, March 2013, pp. 1285–1290.
- [39] Y. Luo, S. Ghose, Y. Cai, E. Haratsch, and O. Mutlu, “Enabling accurate and practical online flash channel modeling for modern MLC NAND flash memory,” *IEEE J. Sel. Areas Commun.*, vol. PP, no. 99, pp. 1–1, 2016.
- [40] D.-hwan Lee and W. Sung, “Estimation of NAND flash memory threshold voltage distribution for optimum soft-decision error correction,” *IEEE Trans. Signal Process.*, vol. 61, no. 2, pp. 440–449, Jan 2013.
- [41] B. Peleato *et al.*, “Adaptive read thresholds for NAND flash,” *IEEE Trans. Commun.*, vol. 63, no. 9, pp. 3069–3081, Sept 2015.
- [42] G. Dong, S. Li, and T. Zhang, “Using data postcompensation and predistortion to tolerate cell-to-cell interference in MLC NAND flash memory,” *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 10, pp. 2718–2728, Oct 2010.
- [43] D. Park and J. Lee, “Floating-gate coupling canceller for multi-level cell NAND flash,” *IEEE Trans. Magn.*, vol. 47, no. 3, pp. 624–628, March 2011.
- [44] D.-h. Lee and W. Sung, “Least squares based coupling cancelation for MLC NAND flash memory with a small number of voltage sensing operations,” *Journal of Signal Processing Systems*, vol. 71, no. 3, pp. 189–200, 2013. [Online]. Available: <http://dx.doi.org/10.1007/s11265-012-0716-0>
- [45] D. h. Lee and W. Sung, “Soft-decision decoding with cell to cell interference removed signal in NAND flash memory,” in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, May 2013, pp. 2518–2522.
- [46] “32Gb, 64Gb, 128Gb: NAND Flash Features,” available online at <http://www.szyuda88.com/uploadfile/cfile/201341914845630.pdf>.
- [47] K. T. Park *et al.*, “A zeroing cell-to-cell interference page architecture with temporary LSB storing and parallel MSB program scheme for MLC NAND flash memories,” *IEEE J. Solid-State Circuits*, vol. 43, no. 4, pp. 919–928, April 2008.
- [48] R. A. Cernea *et al.*, “A 34 MB/s MLC write throughput 16 Gb NAND with all bit line architecture on 56 nm technology,” *IEEE J. of Solid-State Circuits*, vol. 44, no. 1, pp. 186–194, Jan 2009.
- [49] S. Kang-Deog *et al.*, “A 3.3 V 32 Mb NAND flash memory with incremental step pulse programming scheme,” *IEEE J. Solid-State Circuits*, vol. 30, no. 11, pp. 1149–1156, 1995.
- [50] L. P. R. Z. Xueqiang Wang, Guiqiang Dong, “Error correction codes and signal processing in flash memory,” in *Flash Memories*, P. I. Stievano, Ed. InTech, 2011.

- [51] D. Guiqiang *et al.*, “Estimating information-theoretical NAND flash memory storage capacity and its implication to memory system design space exploration,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 9, pp. 1705–1714, Sept 2012.
- [52] G. Dong, N. Xie, and T. Zhang, “Enabling NAND flash memory use soft-decision error correction codes at minimal read latency overhead,” *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 60, no. 9, pp. 2412–2421, Sept 2013.
- [53] T.-Y. Chen, A. Williamson, and R. Wesel, “Increasing flash memory lifetime by dynamic voltage allocation for constant mutual information,” in *Proc. Inf. Theory and Appl. Workshop*, San Diego, CA, Feb 2014, pp. 1–5.
- [54] G. Dong, Y. Pan, and T. Zhang, “Using lifetime-aware progressive programming to improve SLC NAND flash memory write endurance,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 6, pp. 1270–1280, June 2014.
- [55] H. Wang, T.-Y. Chen, and R. Wesel, “Histogram-based flash channel estimation,” in *Proc. IEEE Int. Conf. Commun.*, June 2015, pp. 283–288.
- [56] X. Quan *et al.*, “Modelling and characterization of NAND flash memory channels,” *Measurement*, vol. 70, pp. 225 – 231, 2015.
- [57] C. A. Aslam, Y. L. Guan, and K. Cai, “Detector for MLC NAND flash memory using neighbor-a-priori information,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 9, pp. 2827–2836, Sept 2016.
- [58] N. Mielke *et al.*, “Flash EEPROM threshold instabilities due to charge trapping during program/erase cycling,” *IEEE Trans. Device Mater. Rel.*, vol. 4, no. 3, pp. 335–344, Sept 2004.
- [59] C. M. Compagnoni, M. Ghidotti, A. L. Lacaita, A. S. Spinelli, and A. Visconti, “Random telegraph noise effect on the programmed threshold-voltage distribution of flash memories,” *IEEE Electron Device Lett.*, vol. 30, no. 9, pp. 984–986, 2009.
- [60] K. Kim, “Future memory technology: challenges and opportunities,” in *Proc. Int. Symp. VLSI Technol., Syst. Appl.*, San Jose, CA, 2008, pp. 5–9.
- [61] L. M. Grupp, J. D. Davis, and S. Swanson, “The bleak future of NAND flash memory,” in *Proc. 10th USENIX Conf. File and Storage Technol.*, ser. FAST’12, Berkeley, CA, USA, 2012, pp. 2–2.
- [62] Q. Li, A. Jiang, and E. Haratsch, “Noise modeling and capacity analysis for NAND flash memories,” in *IEEE Int. Symp. Inf. Theory*, Honolulu, HI, June 2014, pp. 2262–2266.
- [63] C. A. Aslam, Y. L. Guan, and K. Cai, “Dynamic write-level and read-level signal design for MLC NAND flash memory,” in *Proc. 9th Int. Symp. Commun. Syst., Netw., Digit. Signal Process.*, Manchester, UK, July 2014, pp. 336–341.

- [64] Y. Kim *et al.*, “Verify level control criteria for multi-level cell flash memories and their applications,” *EURASIP J. Adv. Sig. Proc.*, vol. 2012, no. 1, p. 196, 2012.
- [65] J. Anxiao *et al.*, “Rank modulation for flash memories,” *IEEE Trans. Inf. Theory*, vol. 55, no. 6, pp. 2659–2673, June 2009.
- [66] A. Jiang, H. Li, and J. Bruck, “On the capacity and programming of flash memories,” *IEEE Trans. Inf. Theory*, vol. 58, no. 3, pp. 1549–1564, March 2012.
- [67] A. Jiang and H. Li, “Optimized cell programming for flash memories,” in *Proc. IEEE PACRIM*, Victoria, B.C., Canada, Aug 2009, pp. 914–919.
- [68] M. Qin, E. Yaakobi, and P. Siegel, “Optimized cell programming for flash memories with quantizers,” *IEEE Trans. Inf. Theory*, vol. 60, no. 5, pp. 2780–2795, May 2014.
- [69] F. Sala, R. Gabrys, and L. Dolecek, “Dynamic threshold schemes for multi-level non-volatile memories,” *IEEE Trans. Commun.*, vol. 61, no. 7, pp. 2624–2634, July 2013.
- [70] W. Jiadong *et al.*, “Enhanced precision through multiple reads for LDPC decoding in flash memories,” *IEEE J. Sel. Areas Commun.*, vol. 32, no. 5, pp. 880–891, May 2014.
- [71] J. A. Snyman, *Practical mathematical optimization : An introduction to basic optimization theory and classical and new gradient-based algorithms*, ser. Applied optimization. New York: Springer, 2005.
- [72] T. Richardson, M. Shokrollahi, and R. Urbanke, “Design of capacity-approaching irregular low-density parity-check codes,” *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 619–637, Feb 2001.
- [73] X.-Y. Hu, E. Eleftheriou, and D.-M. Arnold, “Regular and irregular progressive edge-growth Tanner graphs,” *IEEE Trans. Inf. Theory*, vol. 51, no. 1, pp. 386–398, Jan 2005.
- [74] J. Zhang and M. P. C. Fossorier, “Shuffled iterative decoding,” *IEEE Trans. Commun.*, vol. 53, no. 2, pp. 209–213, Feb 2005.
- [75] Y. Cai *et al.*, “Error patterns in MLC NAND flash memory: Measurement, characterization, and analysis,” in *Proc. Conf. Exhibition Design, Automation Test in Europe*, March 2012, pp. 521–526.
- [76] —, “Error analysis and retention-aware error management for NAND flash memory,” *Intel Technology Journal*, 2013.
- [77] —, “Flash correct-and-refresh: Retention-aware error management for increased flash memory lifetime,” in *Proc. 30th Int. Conf. Computer Design*, Sept 2012, pp. 94–101.

- [78] Dong-hwan Lee and W. Sung, “Decision directed estimation of threshold voltage distribution in NAND flash memory,” *IEEE Trans. Signal Process.*, vol. 62, no. 4, pp. 919–927, Feb 2014.
- [79] F. Sala, R. Gabrys, and L. Dolecek, “Dynamic threshold schemes for multi-level non-volatile memories,” *IEEE Trans. Commun.*, vol. 61, no. 7, pp. 2624–2634, July 2013.
- [80] M. Qin *et al.*, “Balanced codes for data retention of multi-level flash memories with fast page read,” in *53rd Annual Allerton Conf. Communication, Control, and Computing (Allerton)*, Sept 2015, pp. 704–711.
- [81] K. Wang *et al.*, “Improving flash memory reliability with dynamic thresholds: Signal processing and coding schemes,” in *Proc. 7th Int. Conf. Communications and Networking in China*, 2012, pp. 161–166.
- [82] A. Berman and Y. Birk, “Constrained flash memory programming,” in *Proc. IEEE Int. Symp. Inf. Theory*, July 2011, pp. 2128–2132.
- [83] M. Qin, E. Yaakobi, and P. H. Siegel, “Constrained codes that mitigate inter-cell interference in read/write cycles for flash memories,” *IEEE J. Sel. Areas Commun.*, vol. 32, no. 5, pp. 836–846, May 2014.
- [84] S. Buzaglo, P. H. Siegel, and E. Yaakobi, “Coding schemes for inter-cell interference in flash memory,” in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, June 2015, pp. 1736–1740.
- [85] V. Taranalli, H. Uchikawa, and P. H. Siegel, “Error analysis and inter-cell interference mitigation in multi-level cell flash memories,” in *Proc. IEEE Int. Conf. Commun. (ICC)*, June 2015, pp. 271–276.
- [86] Y. Kim and B. V. K. V. Kumar, “Coding for memory with stuck-at defects,” in *Proc. IEEE Int. Conf. Commun. (ICC)*, June 2013, pp. 4347–4352.
- [87] —, “Writing on dirty flash memory,” in *Proc. 52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Sept 2014, pp. 513–520.
- [88] Y. Cai *et al.*, “Neighbor-cell assisted error correction for MLC NAND flash memories,” in *Proc. ACM Int. Conf. Measurement and modeling of computer systems*. ACM, 2014, pp. 491–504.
- [89] Y. C. *et al.*, “Program interference in MLC NAND flash memory: Characterization, modeling, and mitigation,” in *Proc. IEEE 31st Int. Conf. Computer Design*, Oct 2013, pp. 123–130.
- [90] D. Park and J. Lee, “Coupling canceller maximum-likelihood (CCML) detection for multi-level cell NAND flash memory,” *IEEE Trans. Consum. Electron.*, vol. 57, no. 1, pp. 160–163, February 2011.

- [91] M. Asadi *et al.*, “Optimal detector for multilevel NAND flash memory channels with intercell interference,” *IEEE J. Sel. Areas Commun.*, vol. 32, no. 5, pp. 825–835, May 2014.
- [92] R. Gallager, “Low-density parity-check codes,” *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, 1962.
- [93] D. Hocevar, “A reduced complexity decoder architecture via layered decoding of LDPC codes,” in *IEEE Workshop. Signal Processing Systems*, Oct 2004, pp. 107–112.
- [94] E. Sharon, S. Litsyn, and J. Goldberger, “Efficient serial message-passing schedules for LDPC decoding,” *IEEE Trans. Inf. Theory*, vol. 53, no. 11, pp. 4076–4091, Nov 2007.
- [95] E. Zimmermann and G. Fettweis, “Reduced complexity LDPC decoding using forced convergence,” in *Proc. 7th Int. Symp. Wireless Personal Multimedia Communications*, 2004, pp. 12–15.
- [96] D. Levin, E. Sharon, and S. Litsyn, “Lazy scheduling for LDPC decoding,” *IEEE Commun. Lett.*, vol. 11, no. 1, pp. 70–72, Jan 2007.
- [97] J. Zhang, Y. Lei, and Y. Jin, “Check-node lazy scheduling approach for layered belief propagation decoding algorithm,” *Electronics Letters*, vol. 50, no. 4, pp. 278–279, February 2014.
- [98] G. Han, Y. L. Guan, and X. Huang, “Check node reliability-based scheduling for BP decoding of non-binary LDPC codes,” *IEEE Trans. Commun.*, vol. 61, no. 3, pp. 877–885, March 2013.
- [99] X. Liu, J. Cai, and L. Wu, “Improved decoding algorithm of serial belief propagation with a stop updating criterion for LDPC codes and applications in patterned media storage,” *IEEE Trans. Mag.*, vol. 49, no. 2, pp. 829–836, Feb 2013.
- [100] Y. Mao and A. Banihashemi, “Decoding low-density parity-check codes with probabilistic scheduling,” *IEEE Commun. Lett.*, vol. 5, no. 10, pp. 414–416, Oct 2001.
- [101] H. Xiao and A. Banihashemi, “Graph-based message-passing schedules for decoding LDPC codes,” *IEEE Trans. Commun.*, vol. 52, no. 12, pp. 2098–2105, Dec 2004.
- [102] A. Vila Casado, M. Griot, and R. Wesel, “LDPC decoders with informed dynamic scheduling,” *IEEE Trans. Commun.*, vol. 58, no. 12, pp. 3470–3479, December 2010.
- [103] G. Han and X. Liu, “An efficient dynamic schedule for layered belief-propagation decoding of LDPC codes,” *IEEE Communications Letters*, vol. 13, no. 12, pp. 950–952, December 2009.

- [104] Y. Gong, X. Liu, W. Yecai, and G. Han, "Effective informed dynamic scheduling for belief propagation decoding of LDPC codes," *IEEE Trans. Commun.*, vol. 59, no. 10, pp. 2683–2691, October 2011.
- [105] H.-C. Lee *et al.*, "Two informed dynamic scheduling strategies for iterative LDPC decoders," *IEEE Trans. Commun.*, vol. 61, no. 3, pp. 886–896, March 2013.
- [106] X. Liu, Y. Zhang, and R. Cui, "Variable-node-based dynamic scheduling strategy for belief-propagation decoding of LDPC codes," *IEEE Commun. Lett.*, vol. 19, no. 2, pp. 147–150, Feb 2015.
- [107] T. Richardson, "Error floors of LDPC codes," in *Proc. Annual Allerton Conf. Communication Control and Computing*, vol. 41, no. 3. The University; 1998, 2003, pp. 1426–1435.
- [108] C. Healy and R. C. de Lamare, "Knowledge-aided informed dynamic scheduling for LDPC decoding," in *Proc. IEEE Int. Conf. Commun. Workshop (ICCW)*, June 2015, pp. 2212–2217.
- [109] H.-C. Lee and Y.-L. Ueng, "LDPC decoding scheduling for faster convergence and lower error floor," *IEEE Trans. Commun.*, vol. 62, no. 9, pp. 3104–3113, Sept 2014.
- [110] T. Richardson, M. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Commun.*, vol. 47, no. 2, pp. 619–637, Feb 2001.
- [111] "Encyclopedia of sparse graph codes," available online at <http://www.inference.phy.cam.ac.uk/mackay/codes/data.html>.
- [112] M. Damle, D. S. Limaye, and M. Sonwani, "Comparative analysis of array multiplier using different logic styles."
- [113] R. Shirota *et al.*, "A new programming scheme for the improvement of program disturb characteristics in scaled NAND flash memory," *IEEE Trans. Electron Devices*, vol. 59, no. 10, pp. 2767–2773, Oct 2012.
- [114] Y. Cai, Y. Luo, S. Ghose, and O. Mutlu, "Read disturb errors in MLC nand flash memory: Characterization, mitigation, and recovery," in *Proc. Annual IEEE/IFIP Int. Conf. on Dependable Systems and Networks*, June 2015, pp. 438–449.
- [115] Q. H. *et al.*, "A concatenation scheme of LDPC codes and source codes for flash memories," *EURASIP J. Adv. Sig. Proc.*, pp. 208–208, 2012.
- [116] M. Marrow, "Coding architecture for multi-level NAND flash memory with stuck cells," Jun. 2 2015, US Patent 9,047,213. [Online]. Available: <http://www.google.com/patents/US9047213>

- [117] T. Parnell, N. Papandreou, T. Mittelholzer, and H. Pozidis, “Modelling of the threshold voltage distributions of sub-20nm NAND flash memory,” in *Proc. IEEE Global Communications Conference*, Dec 2014, pp. 2351–2356.
- [118] M. P. C. Fossorier, M. Mihaljevic, and H. Imai, “Reduced complexity iterative decoding of low-density parity check codes based on belief propagation,” *IEEE Trans. Commun.*, vol. 47, no. 5, pp. 673–680, May 1999.
- [119] Q. Huang, J. Kang, L. Zhang, S. Lin, and K. Abdel-Ghaffar, “Two reliability-based iterative majority-logic decoding algorithms for LDPC codes,” *IEEE Trans. Commun.*, vol. 57, no. 12, pp. 3597–3606, December 2009.
- [120] W. Ryan and S. Lin, *Channel codes: classical and modern*. Cambridge University Press, 2009.

Appendix A

Derivation of hard-decision levels R_1 (3.9), R_2 (3.10) and R_3 (3.11)

In this appendix, we only solve the expression for hard-decision level R_1 . The expression for other two hard-decision levels R_2 and R_3 can be obtained by following the similar steps. Using the equality (3.6), we simplify it for R_1 , as follows

$$\begin{aligned} p_{s_{11}}(v = R_1) &= p_{s_{10}}(v = R_1) \\ \Rightarrow \frac{1}{\sqrt{2\pi\sigma_{s_{11}}^2}} \exp - \frac{(R_1 - \mu_{s_{11}})^2}{2\sigma_{s_{11}}^2} &= \frac{1}{\sqrt{2\pi\sigma_{s_{10}}^2}} \exp - \frac{(R_1 - \mu_{s_{10}})^2}{2\sigma_{s_{10}}^2} \end{aligned}$$

Taking natural logarithm on both sides, we get

$$\begin{aligned} \log \left(\frac{\sigma_{s_{11}}}{\sigma_{s_{10}}} \right) &= \frac{(R_1 - \mu_{s_{10}})^2}{2\sigma_{s_{10}}^2} - \frac{(R_1 - \mu_{s_{11}})^2}{2\sigma_{s_{11}}^2} \\ \Rightarrow (\sigma_{s_{11}}^2 - \sigma_{s_{10}}^2) R_1^2 + 2(\mu_{s_{11}}\sigma_{s_{10}}^2 - \mu_{s_{10}}\sigma_{s_{11}}^2) R_1 + \\ &\left[\mu_{s_{10}}^2\sigma_{s_{11}}^2 - \mu_{s_{11}}^2\sigma_{s_{10}}^2 - 2\sigma_{s_{11}}^2\sigma_{s_{10}}^2 \log \left(\frac{\sigma_{s_{11}}}{\sigma_{s_{10}}} \right) \right] = 0 \end{aligned}$$

This is a quadratic equation which can be solved by finding the roots of R_1 in the interval $[V_{s_{11}}, V_{s_{10}}]$, given by

$$R_1 = \frac{-b - \sqrt{b^2 - 4ac}}{2a} \quad (\text{A.1})$$

where

$$\begin{aligned} a &= (\sigma_{s_{11}}^2 - \sigma_{s_{10}}^2) \\ b &= 2 (\mu_{s_{11}} \sigma_{s_{10}}^2 - \mu_{s_{10}} \sigma_{s_{11}}^2) \\ c &= \left[\mu_{s_{10}}^2 \sigma_{s_{11}}^2 - \mu_{s_{11}}^2 \sigma_{s_{10}}^2 - 2 \sigma_{s_{11}}^2 \sigma_{s_{10}}^2 \log \left(\frac{\sigma_{s_{11}}}{\sigma_{s_{10}}} \right) \right] \end{aligned}$$

Replacing a , and b in (A.1), we get the final expression for R_1 (3.9), given by

$$R_1 = \frac{- (\mu_{s_{11}} \sigma_{s_{10}}^2 - \mu_{s_{10}} \sigma_{s_{11}}^2) - \sqrt{(\mu_{s_{11}} \sigma_{s_{10}}^2 - \mu_{s_{10}} \sigma_{s_{11}}^2)^2 - (\sigma_{s_{11}}^2 - \sigma_{s_{10}}^2) (c)}}{(\sigma_{s_{11}}^2 - \sigma_{s_{10}}^2)}$$

Similarly, R_2 can be obtained by replacing $\mu_{s_{11}}$, $\sigma_{s_{11}}$, $\mu_{s_{10}}$ and $\sigma_{s_{10}}$ with $\mu_{s_{10}}$, $\sigma_{s_{10}}$, $\mu_{s_{00}}$ and $\sigma_{s_{00}}$, respectively, and R_3 can be obtained by replacing $\mu_{s_{10}}$, $\sigma_{s_{10}}$, $\mu_{s_{00}}$ and $\sigma_{s_{00}}$ with $\mu_{s_{00}}$, $\sigma_{s_{00}}$, $\mu_{s_{01}}$ and $\sigma_{s_{01}}$, respectively.

Appendix B

Derivation of Check-node-type probability p (Type-2) (6.13) and p (Type-3) (6.14)

Consider a set of $\bar{d}_c$ independent binary random variables $\{a_1, a_2, \dots, a_{\bar{d}_c}\}$, with $p_{i_1} = \Pr(a_i = 1)$ and $p_{i_0} = \Pr(a_i = 0)$, that are connected with check node c_m . Given that, we can compute the probability that the set contains an even number of 1s (check node is satisfied), given by [120]

$$\left[\frac{1}{2} + \frac{1}{2} \prod_{i=1}^{\bar{d}_c} (1 - 2p_{i_1}) \right] \quad (\text{B.1})$$

Then, the probability that all $\bar{d}_c$ variables have probability p_{i_1} is given by $(p_{i_1})^{\bar{d}_c}$, and the probability that atleast one variable that has probability not equal to p_{i_1} is given by $1 - (p_{i_1})^{\bar{d}_c}$. Multiplying these probabilities with (B.1) and replacing p_{i_1} with LLR representation $p_{i_1} = \frac{e^\Lambda}{1+e^\Lambda}$, where $\Lambda = \log\left(\frac{p_{i_0}}{p_{i_1}}\right)$ and $p_{i_0} = 1 - p_{i_1}$, we get

the expression for $p(\text{Type-2})$ (6.13) and $p(\text{Type-3})$ (6.14), as

$$p(\text{Type-2}) = \left[1 - \left(\frac{e^\Lambda}{1 + e^\Lambda} \right)^{\bar{d}_c} \right] \left[\frac{1}{2} + \frac{1}{2} \prod_{i=1}^{\bar{d}_c} \left(1 - \frac{2e^\Lambda}{1 + e^\Lambda} \right) \right]$$

$$p(\text{Type-3}) = \left(\frac{e^\Lambda}{1 + e^\Lambda} \right)^{\bar{d}_c} \left[\frac{1}{2} + \frac{1}{2} \prod_{i=1}^{\bar{d}_c} \left(1 - \frac{2e^\Lambda}{1 + e^\Lambda} \right) \right]$$