COLLEGE MATHSTIC
DOCTORAL BRETAGNE
BRETAGNE OCEANE

Université Bretagne Sud

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ BRETAGNE SUD

ÉCOLE DOCTORALE N° 644
Mathématiques et Sciences et Technologies de l'Information et de la Communication en Bretagne Océane
Spécialité : *Télécommunications*

Par
Joseph JABOUR

Optimisation Algorithmique des Décodeurs Non-Binaires

Algorithmic Optimization of Non-Binary Decoders

Thèse présentée et soutenue à Lorient, le Dec 14, 2023.
Unité de recherche : «Lab-STICC UMR6285»
Thèse N° : 686

Rapporteurs avant soutenance :

Francisco Miguel GARCÍA HERRERO	Professeur à Universidad Complutense de Madrid.
Christophe JEGO	Professeur à ENSEIRB-MATMECA - IMS.

Composition du Jury :

Président :	Charly POUILLAT	Professeur à INP-ENSEEIHT.
Examinateurs :	Francisco Miguel GARCÍA HERRERO	Professeur à Universidad Complutense de Madrid.
	Christophe JEGO	Professeur à ENSEIRB-MATMECA - IMS.
	Charly POUILLAT	Professeur à INP-ENSEEIHT.
Dir. de thèse :	Emmanuel BOUTILLON	Professeur à l'Université Bretagne Sud.
Co-dir. de thèse :	Ali AL-GHOUWAYEL	Maître de conférences à l'IMT Atlantique.
	Hussein HIJAZI	Maître de conférences à Lebanese International University.
	Cédric MARCHAND	Docteur-Ingénieur de recherche à l'Université Bretagne Sud

To my loving parents and family, your unwavering belief, endless support, and unconditional love are the true fuel of this journey. This work is dedicated to you with heartfelt gratitude.

ACKNOWLEDGMENT

This thesis is part of a collaborative framework between the Université Bretagne Sud (UBS) and the Lebanese International University (LIU) under the direction of Prof. Emmanuel Boutillon and the supervision of Dr. Ali Al-Ghouwayel, Dr. Cédric Marchand, and Dr. Hussein Hijazi.

Also, this thesis is a part of the Quasi-Cyclic Short Packet (QCSP) [1] where the association of a spreading code called Cyclic-Code Shift Keying (CCSK) along with non-binary codes is used to generate self-synchronized and self-identified waveforms for IoT networks.

Foremost, I express profound indebtedness to Prof. Boutillon for his invaluable encouragement during the direction of this thesis. His unwavering support is a cornerstone of this research success. I am genuinely thankful for the intellectual growth and exploration opportunities he has facilitated.

Also, I extend my heartfelt appreciation to the thesis supervisors, Dr. Al-Ghouwayel and Dr. Hijazi, for their support in both academic and personal dimensions. The nomination for this Ph.D. program by them is a recognition I deeply honor.

I am equally grateful to Dr. Marchand for his invaluable assistance and engaging discussions. He was not only a thesis supervisor, but also a great friend.

Special thanks are due to Prof. Charbel Abdelnour, Dr. Valentin Savin, and all members of the QCSP project for their indirect yet fruitful contributions to this thesis.

To the esteemed committee members, Profs. Catherine Douillard, Francisco Miguel García Herrero, Christophe Jego, and Charly Pouillat, I express deep appreciation and gratitude for your insightful comments and suggestions, which have significantly enhanced the quality of this work.

In addition, my thanks extend to Université de Bretagne Sud for providing an academically conducive environment and essential resources vital for successfully completing this research. I also acknowledge the limitless help of the administrative staff, including Mrs. Noluenn Chavin, Mrs. Béatrice Guern, Mrs. Virginie Guillet, Mrs. Clarie Jurysta, and Mrs. Florence Palin.

Last but not least, I thank all my friends in Lorient for making this journey as sweet

as a sugar cube. I cherish every moment, joke, and discussion we have together.

Table of Contents

Introduction **16**

1 Fundamentals: From Finite Fields to Spread Spectrum. **21**
1.1 Introduction to Galois Fields 21
1.1.1 Abelian Groups 21
1.1.2 Galois Fields 22
1.2 Asymptotic and Finite Block-Length Achievable Code Rates 24
1.2.1 Asymptotic Achievable Rate 24
1.2.2 Finite-Block Achievable Rate 26
1.3 Cyclic Code Shift Keying Modulation 27

2 Non-Binary Low-Density Parity Check Codes **28**
2.1 Introduction to NB-LDPC Codes 30
2.1.1 Structure of NB-LDPC Decoder 31
2.1.2 Overview on NB-LDPC Code Construction 32
2.2 NB-LDPC Iterative Decoding Algorithms 33
2.2.1 Belief Propagation Algorithm 33
2.2.2 Belief Propagation in Logarithmic Domain 35
2.2.3 Min-Sum Algorithm 37
2.2.4 Min-Max Algorithm 37
2.3 Extended Min Sum Algorithm and Its Implementation Approaches 37
2.3.1 Extended Min-Sum Algorithm 38
2.3.2 Forward-Backward CN Processing 40
2.3.3 Syndrome-Based CN Processing 43
2.3.4 Presorted EMS Algorithm 47
2.3.5 Hybrid Check Node Processing 49
2.4 Trellis Extended Min-Sum Algorithm and Its Variants 49
2.4.1 Trellis Extended Min-Sum Algorithm 50
2.4.2 One Minimum Only Trellis-EMS 52

2.4.3 Two-Extra Column Trellis EMS 53
2.5 The Best, The Requested, and The Default Algorithm 55
2.5.1 Introduction to the BRD Algorithm 55
2.5.2 Statistical and Requested Symbols Analysis 59
2.5.3 Trellis BRD Decoder 61
2.5.4 Syndrome-based BRD Algorithm 63
2.5.5 Forward-Backward BRD Decoder 68
2.5.6 Implementation of FB-BRD Check Node 75
2.6 Conclusion 89

3 Non-Binary Polar Codes and Decoders 92
3.1 Introduction to Non-binary Polar Codes 94
3.1.1 Polar Transformation and Channel Polarization 94
3.1.2 Polar Coding 95
3.1.3 Overview on Non-Binary Polar Codes 96
3.2 Successive Cancellation Decoding for Non-Binary Polar Codes 98
3.3 Construction Methodology of NB-PCs 102
3.3.1 Selection of the Generator Matrix Coefficients Coefficients 102
3.3.2 Selection of Frozen Symbol Channels 104
3.4 Efficient Implementation of the SC-based Polar Decoders 105
3.4.1 Simplified Successive Cancellation Decoder 105
3.4.2 Min-Sum Successive Cancellation Decoder 106
3.4.3 Simplified Min-Sum SC Decoder 108
3.4.4 Extended Min-Sum Successive Cancellation Decoder 108
3.5 Performance and Complexity Comparison of NB-LDPC and NB-PC Decoder 109
3.5.1 Performance Comparison 110
3.5.2 Complexity Comparison 110
3.6 Asymmetrical Extended Min-Sum SC Decoders 113
3.6.1 Introduction to Asymmetrical Extended Min-Sum CN 113
3.6.2 Determination of the Asymmetrical Sizes 114
3.6.3 Computation Reduction Using L-bubble Approach 115
3.6.4 Complexity Analysis and Simulation Results 117
3.7 Polarization-Aware SC Decoder 123
3.7.1 Definition of Nodes Clusters 123

3.7.2 Statistical Computation Using EMS CNs 123
3.7.3 Bubbles Pruning Process 125
3.7.4 Proposed Design of SC-PA Decoders 128
3.7.5 Complexity Analysis and Simulation Results 131
3.8 Conclusion 136

Conclusion and Perspectives **140**

Bibliography **144**

A Supplementary Material for Non-Binary Polar Codes **154**
A.1 Reliability Sequences 154
A.2 Kernels Coefficients for Non-Binary Polar Codes with BPSK Modulation . 165

List of Figures

2.1 A Graphical Representation of **H** with a Tanner Graph 32
2.2 Messages Exchanged Between a VN and a CN Edge. 35
2.3 Example of Message Truncation in EMS algorithm 38
2.4 Forward-Backward Architecture for $d_c = 5$ 41
2.5 Potential Elements of T_Σ Computed using Bubble Sorters. 43
2.6 Syndrome-based EMS Decoder [45] 44
2.7 Presorting Algorithm [47] 48
2.8 Extended Forward Check Node for $d_c = 6$. 49
2.9 Trellis-EMS Example on GF(4) 53
2.10 Radix-2 One Minimum Finder 54
2.11 The Best, the Requested, and the Default Decoder 56
2.12 Toy Example on the BRD Decoder over GF(8). 59
2.13 Probability that the symbol $x_j \in \Gamma(M_{C2V_j})$ 61
2.14 TEMS-BRD Simulation Results for Code Rates $r = 1/2$, $r = 5/6$, $r = 9/10$. 63
2.15 SYN-BRD Architecture 64
2.16 Simulation of the SYN-BRD Decoder 66
2.17 Structure of the Presorted SYN-BRD Decoder 67
2.18 Paths Elimination Block 67
2.19 Simulation Results of Presorted SYN-BRD Decoder 69
2.20 Forward-Backward BRD Decoder 69
2.21 Simulation Results for FB-BRD Decoder over GF(64) 73
2.22 Simulation Results for FB-BRD Decoder over GF(256) 73
2.23 Structure of the Presorted FB-BRD for $N = 60, K = 20$ 74
2.24 Structure of the Presorted FB-BRD for $N = 144, K = 120$ 75
2.25 Simulation Performance of Presorted FB-BRD for $r = 1/3$ and $r = 5/6$ . . 76
2.26 Schematic Representation of Comparators. 77
2.27 Full Sixteen-to-Sixteen Bitonic Sorter 78
2.28 Top-Level View of a Full Sixteen-to-Sixteen Sorter 79
2.29 Parallel S-Bubble Sorter for ECN with Input Size $n_m = 17$. 81

2.30 Optimized 32-to-16 Sorter for S-bubble. 81
2.31 Optimized 16-to-16 Sorter for S-bubble. 82
2.32 FB-BRD Architecture for $d_c = 12$. 82
2.33 Sixteen-to-Sixteen Sorter for ECNs F_1 and B_1. 83
2.34 Twenty Four-to-Sixteen S-bubble Sorter Architecture 84
2.35 Architecture of Best Finder Sorter 85
2.36 Internal Structure of Request Finder Block. 86
2.37 Internal Structure of LLR Finder Block. 86
2.38 Internal Structure of Column Finder Block. 87

3.1 Polar Transformation 95
3.2 Non-binary Kernel Transformation 97
3.3 Non-binary Polar Transformation for $N = 4$. 98
3.4 NB-PC Kernel Encoding and Decoding Processes 99
3.5 Decoding Process of the t^{th} kernel at layer l 100
3.6 Factor Graph of SC Decoder for $N = 8$ 101
3.7 Non-Binary Polar Decoder with $n = 3$ Polarization Steps. 103
3.8 Simplified Polar Decoder for $N = 8$ 106
3.9 FER Performance at $N \approx 64$ and $r \approx 1/3$. 110
3.10 FER Performance at $N \approx 64$ and $r \approx 1/2$. 111
3.11 FER Performance at $N \approx 128$ and $r \approx 2/3$. 111
3.12 FER Performance at $N \approx 256$ and $r \approx 2/3$. 112
3.13 Contribution Rate Matrix C for Layer $l = 1$ CNs. 116
3.14 Schematic Structure of an AEMS CN. 117
3.15 Simulation Results over $GF(64)$ for $N = 64$, $r \approx 1/3$ ($r_e \approx 1/32$) and $r \approx 2/3$ ($r_e \approx 1/16$) respectively. 119
3.16 Simulation Results over $GF(64)$ for $N = 256$, $r \approx 1/3$ ($r_e \approx 1/32$) and $r \approx 2/3$ ($r_e \approx 1/16$) respectively. 120
3.17 Simulation Results over $GF(64)$ for $N = 1024$, $r \approx 1/3$ ($r_e \approx 1/32$) and $r \approx 2/3$ ($r_e \approx 1/16$) respectively. 121
3.18 Performance Simulation of SC-AEMS Decoder with BPSK Modulation . . 122
3.19 Contribution Rate $\mathcal{C}_1^{(2)}$ (in %) at Layer $l = 2$ and node cluster $s = 1$ for $N = 64$. 126
3.20 Pruned Computed Region $\mathcal{R}_1^{(2)}$ (in %) at Layer $l = 2$ and node cluster $s = 1$ for $N = 64$ over GF(64). 127

3.21 SC Decoder for $N = 8$ with Node Clusters Illustration 128
3.22 EMS-based SC Decoder with $K = 42$ and $N = 64$ over Different n_m sizes. . 129
3.23 Performance of SC-PA Decoder over Different Threshold Values $\mathcal{P}_t$ for $N = 64$ and $K = 21$. 130
3.24 Simulation Results for $N = 64$ over GF(64). 133
3.25 Simulation Results for $N = 256$ over GF(64). 133
3.26 Simulation Results for $N = 1024$ over GF(64). 134
3.27 Simulation Results of SC-PA for $N = 128$ and $K = 42$ over BPSK Modulation. 136

List of Tables

1.1 Addition ($\oplus$) and Multiplication ($\circledast$) Operations over GF(2). 22
1.2 Representation of Field Elements on GF(8). 23
1.3 Multiplication ($\circledast$) Operations over GF(8) using Power Representation. . . 24
1.4 Addition ($\oplus$) Operations over GF(8) using Polynomial Representation. . . 25

2.1 Syndrome Set for $d_1 = 3$ 46
2.2 Syndrome Set for $d_2 = 2$ 46
2.3 Size of Exchanged Messages per Edge on GF(64) 72
2.4 Synthesis Results for $d_c = 12$ on GF(64) 89

3.1 Arithmetic Operations Performed per CN 118
3.2 Design Parameters for SC-PA Decoder and Required Arithmetic Operations over Different Codes. 132

A.1 Reliability Sequences for $N = 64$ over CCSK Modulation 154
A.2 Reliability Sequences for $N = 256$ over CCSK Modulation 155
A.3 Reliability Sequences for $N = 1024$ over CCSK Modulation 157
A.4 Coefficients for $N = 128$ over BPSK Modulation 165

List of Notations

N	Code length
K	Information length
M	Redundancy length
q	Field order
$\mathbf{H}$	Parity check matrix
$h_{i,j}$	Parity check matrix element at index i, j
X	Codeword vector
Y	Received vector
d_c	Check node degree of connectivity
d_v	Variable node degree of connectivity
$M^{\oplus}$	GF vector of a message M
M^{+}	LLR vector of a message M
C_i	i^{th} check node
V_j	j^{th} variable node
I_j	Intrinsic vector of variable node j
$M_{V_j 2 C_i}$	Message sent from variable j to check i
$M^{P}_{V_j 2 C_i}$	Permuted version of $M_{V_j 2 C_i}$
$M^{P}_{C_i 2 V_j}$	Message sent from check i to variable j
$M_{C_i 2 V_j}$	Inversely permuted version of $M^{P}_{C_i 2 V_j}$
APP_j	A Posteriori Probability vector of variable V_j
$\hat{x}_j$	Estimated symbol of variable node V_j
n_m	Total size of truncated messages
n_{vc}	Total size of variable to check messages
n_{cv}	Total size of check to variable messages
$M^{B}_{C_i 2 V_j}$	Message with best candidates from check C_i sent to variable V_j
$M^{R\oplus}_{j}$	Requested symbols from variable V_j to check C_i
$M^{R+}_{C_i 2 V_j}$	LLR vector of requested symbols from variable V_j to check C_i
n_B	Size of the best candidates
n_R	Size of the requested candidates

$\Omega(M_{V_j 2C_i})$	Compressed variable V_j to check C_i message
$\Omega^{-1}(\Omega(M^P_{V_j 2C_i}))$	Decompressed permuted version of $\Omega(M_{V_j 2C_i})$
$\Gamma(M^P_{C_i 2V_j})$	Compressed check C_i to variable V_j message
$\Gamma^{-1}(\Gamma(M^P_{C_i 2V_j}))$	Decompressed inversely permuted check C_i to variable V_j message
n_{IN}	Size of internal messages between two elementary check nodes
U	Input message of polar encoder
G_N	Polar generator matrix for $N = 2^n$
$\mathcal{A}_D$	Set of data channels
$P_i^{(l)}$	Probability Density function for node i and layer l
$L_i^{(l)}$	Log-likelihood ratio vector for node i and layer l
$M_i^{(l)}$	Truncated Log-likelihood ratio vector for node i and layer l
n_L	Size of most reliable input of an asymmetrical check node
n_H	Size of least reliable input of an asymmetrical check node
$\hat{u}_i^l$	Estimated symbol at node i and layer l
$S_s^{(l)}$	Set of kernel indices that correspond to cluster s
$\mathcal{B}^{(l)_t}$	Bubble pattern matrix of check node at kernel t and layer l
$\mathcal{C}_s^{(l)}$	Contribution Rate matrix of cluster s at layer l
$\mathcal{R}_s^{(l)}$	Bubble indicator matrix of cluster s at layer l
$T_{\Sigma_s}^{'(l)}$	Bubble region assigned for cluster s at layer l
$n_s^{(l)}$	Output size for cluster s at layer l
$\mathcal{P}_t$	Inclusion threshold

GLOSSARY

AWGN	Additive White Gaussian Noise
BP	Belief Propagation
BPSK	Binary Phase Shift Keying
BRD	Best, Requested, and Default
CCSK	Cyclic-Code Shift Keying
CN	Check Node
CO	Compare-Only comparator
CS	Compare-Swap comparator
DBV	Discard Binary Vector
ECN	Elementary Check Node
EF	Extended Forward
EMS	Extended Min-Sum
GF	Galois Field
FB	Forward-Backward
LLR	Log-Likelihood Ratio
Log-BP	Logarithmic Belief Propagation
NB-LDPC	Non-Binary Low-Density Parity Check
NB-PC	Non-Binary Polar Code
OMO-TEMS	One Minimum Only Trellis Extended Min-Sum
PCM	Parity Check Matrix
PDF	Probability Density Function
RE	Redundancy Elimination
SC	Successive Cancellation
SC-AEMS	Asymmetrical Extended Min-Sum Successive Cancellation
SC-EMS	Extended Min-Sum Successive Cancellation
SC-MS	Successive Cancellation Min-Sum
SC-PA	Polarization-Aware Successive Cancellation
SYN	Syndrome-based
TEC-TEMS	Two-Extra Column Trellis Extended Min-Sum

TEMS	Trellis Extended Min-Sum
VN	Variable Node

INTRODUCTION

Amid the ongoing industrial revolution, communication services play a major role in the development of society. These services are extended to involve newly emerged services such as the Internet of Things (IoT), human-machine interaction, machine-to-machine communication, multimedia services, autonomous vehicles, holographic communications, deep space communication, etc. The proliferation of these services will lead to a significant increase in the number of connected devices which is expected to reach around 30 billion devices in 2030, according to Statista [2]. Therefore, due to spectral scarcity, reliable and spectral-efficient communications are the main challenges for the research community.

Back in the mid-twentieth century, Claude Shannon in his famous paper on information theory [3] showed that reliable communication is possible using error control coding. Prior to Shannon's seminal work in 1948, data repetition was the only way to reduce the error probability of transmission over a noisy channel. Hence, it was inefficient, as it required significantly more bandwidth to transmit the same amount of data.

Since then, plenty of error-correcting schemes have been proposed. Error control coding is a coding procedure followed to control the occurrences of errors which helps in error detection and correction. In any error-correcting scheme (code), a message (information) block of K symbols is transformed into a codeword of N symbols by adding $M = N - K$ redundant (parity) symbols. Those parity symbols serve in error detection and correction to check the consistency of the delivered message and to recover the corrupted data.

Binary codes such as the Low-Density Parity Check (LDPC) [4], [5], Turbo (TC) [6], and Polar Codes (PC) [7] are standardized for use in different communication standards such as the Digital Video Broadcasting (DVB) [8], and the Third Generation Partnership Project (3GPP)[9]. The binary LDPC, turbo, and polar codes can approach the channel capacity limits under certain scenarios. Nevertheless, with the emergence of novel communication standards and applications such as the Ultra-Reliable Low-Latency Communication (URLLC) [10], and the Internet of Things (IoT), the communication standards are more focused on low power, low latency, and high throughput codes with very low-error floors. Furthermore, small packet communication became more popular in wireless systems and broadband cellular networks, such as the Fifth Generation (5G) technology

standard.

Moreover, the coexistence of several radio applications constrains the spectrum resources. The amount of data transmitted in a variety of applications continues to expand, while the number of users grows at an exponential rate. In this context, operators and manufacturers strive to maximize the spectral efficiency of the systems. The use of high-order modulations is the ultimate solution to the paucity of frequency spectrum resources and the necessity for high data rate transmissions.

Consequently, neither of the present standardized codes is capable of satisfying the recent requirements simultaneously. Hence, the study of new codes is essential for the research community. The Non-Binary (NB) codes are one of the novel codes that gained much attraction in recent years.

Non-binary codes are extensions of the binary codes with symbols and coefficients represented on a field of $GF(q = 2^p)$ with $p > 1$. Therefore, each symbol or coefficient is a binary vector of $p-$bits. The motivation to study non-binary codes is due to two main reasons. Firstly, the performance of the binary codes degrades at short frames due to the high correlation experienced in the iterative decoding process [11], [12]. In contrast, the correlation is reduced in the non-binary codes since a higher number of symbols represent the data. Therefore, the non-binary decoding outperforms the binary decoding as provided in [13]. Secondly, the binary codes suffer from binary marginalization when modulated on $q-$ary modulation [14]. Therefore, the non-binary codes maintain a better performance than binary codes when associated with high-order modulations [15]–[17], especially at low coding rates. The performance gain between binary codes and non-binary codes can range between 1 dB to 2 dB according to the assessment provided in [15] over different fields and modulation orders.

On the other hand, the decoding process of the NB codes is much more complicated than their binary counterparts. Thus, the complexity of the non-binary decoders is the bottleneck of a widespread standardization of the non-binary codes. Nevertheless, non-binary LDPC codes, have been adopted in the experimental specification for channel coding in the Consultative Committee for Space Data Systems (CCSDS) [18]. Additionally, they have been standardized recently for the Chinese Satellite Navigation System, BeiDou [19].

In addition, the non-binary codes combined with the CCSK modulation [20] provide several advantages compared to state-of-the-art waveform as it offers self-synchronization and self-identification capabilities, and can operate at ultra-low Signal-to-Noise Ratios

(SNR).

In the context of the QCSP project, the objective of this PhD is to study and optimize the decoding algorithms of non-binary codes such as the LDPC, TC, and PC to allow the achievement of high-throughput decoders with a hardware-friendly complexity.

During this PhD, the non-binary turbo codes were studied and the state-of-the-art were reproduced. However, due to the insignificant optimization, the study on non-binary turbo codes is not discussed in this report.

Thesis Outline

This thesis comprises four key chapters. The initial chapter, Chapter 1, delves into fundamental subjects, encompassing finite fields and the maximum achievable rate, along with a brief introduction to a spread spectrum technique known as cyclic-code shift keying. Subsequently, Chapter 2 is devoted to a comprehensive investigation of non-binary low-density parity-check codes. Proceeding to Chapter 3, a dedicated exploration of non-binary polar codes is presented. To round out the thesis, a concluding chapter offers a summary of the primary content discussed and outlines potential avenues for future research.

Chapter 2 initiates with an exposition on NB-LDPC codes and their decoders in Section 2.1. Subsequently, Section 2.2 delves into the discussion of iterative decoding algorithms, encompassing both optimal and sub-optimal solutions. Section 2.3 then introduces the extended min-sum algorithm and elaborates on its implementation techniques. Furthermore, Section 2.4 presents the trellis extended min-sum algorithm, along with a discussion of state-of-the-art optimization techniques. The principal contribution to the NB-LDPC decoder within the scope of this doctoral work is expounded in Section 2.5. This section outlines the proposed algorithm, known as the BRD (Best, Requested, and Default) algorithm, as introduced in Section 2.5.1. Section 2.5.2 provides a thorough analysis supported by statistical justification. The implementation of the BRD algorithm, juxtaposed with state-of-the-art alternatives, is comprehensively detailed from Section 2.5.3 to Section 2.5.5. The synthesis results of the implemented check node are documented in Section 2.5.6, specifically for the forward-backward BRD check node. Finally, Section 2.6 encapsulates the chapter's key insights and findings.

Additionally, Chapter 3 commences with an introductory exploration of non-binary polar codes in Section 3.1. This section encompasses the concept of channel polarization, polar coding, and the extension of polar codes into non-binary field orders. Subsequently,

Section 3.2 introduces the successive cancellation decoding for non-binary polar codes. Section 3.3 elaborates on the construction of polar codes and the kernel coefficients. In Section 3.4, state-of-the-art approaches for optimizing successive cancellation decoding are delineated. Section 3.6 introduces the first optimization contribution for non-binary successive cancellation decoding, referred to as the asymmetrical extended min-sum. The exploration of optimization is extended in Section 3.7, where a highly customized decoder is designed, with dynamic processing tailored to specific groups of nodes, known as polarization-aware decoding.

Contributions of the Thesis

This thesis encompasses novel contributions aimed at reducing the decoding complexity using algorithmic optimization on non-binary LDPC and Polar decoders.

The optimization strategy put forth to mitigate the decoding complexity associated with non-binary LDPC (NB-LDPC) decoders is referred to as the Best, Requested, and Default (BRD) algorithm. This algorithm represents a novel development and has been granted a patent. Fundamentally, the BRD algorithm reconfigures the decoding processes by facilitating bidirectional information exchange through a push-and-pull mechanism, as opposed to exclusively utilizing a unidirectional push method. This technique leads to a substantial reduction in communication load when compared to alternative approaches. Moreover, it achieves a noteworthy reduction in overall memory allocation without introducing any supplementary computational complexity, while also offering a modest reduction in computational demands.

In the realm of polar decoders, two optimization approaches have been proposed. The initial approach involves the integration of asymmetrical processing alongside a reduction in the internal processing of the decoding units. The second, more intricate approach employs statistical information to craft a highly personalized polar decoder, wherein diverse nodes are allocated distinct processing resources and message sizes.

List of Publications

J. Jabour, A. C. Al-Ghouwayel, and E. Boutillon, "Asymmetrical Extended Min-Sum for Successive Cancellation Decoding of Non-Binary Polar Codes," 2023 12th International Symposium on Topics in Coding (ISTC), Brest, France, 2023, pp. 1-5, doi: 10.1109/ISTC57237.2023.10273502.

J. Jabour, C. Marchand, and E. Boutillon, "The Best, the Requested, and the Default Elementary Check Node for EMS NB-LDPC Decoder," 2023 IEEE Wireless Communications and Networking Conference (WCNC), Glasgow, United Kingdom, 2023, pp. 1-6, doi: 10.1109/WCNC55385.2023.10118720.

J. Jabour, C. Marchand, and E. Boutillon, "The Best, The Requested, and The Default Non-Binary LDPC Decoding Algorithm," 2021 11th International Symposium on Topics in Coding (ISTC), Montreal, QC, Canada, 2021, pp. 1-5, doi: 10.1109/ISTC49272.2021.9594148.

Patents:

E. Boutillon, J. Jabour, and C. Marchand, "A method for decoding a codeword encoded using a non-binary code, corresponding device, and computer program", WO2023025960A, 2023.

Chapter 1

Fundamentals: From Finite Fields to Spread Spectrum.

In this chapter, the fundamental concepts that are mentioned (or used) in this thesis are described such as the Galois Fields presented in section 1.1, the maximum achievable rates presented in 1.2, and the cyclic code shift keying modulation presented in 1.3.

1.1 Introduction to Galois Fields

Finite Fields are widely used in digital communication to improve the robustness, performance, and efficiency of the different blocks in the digital communication chain. In channel coding, the finite fields are used to enhance the decoding performance of the error-correction codes. Therefore, in this section, a brief explanation of the finite fields (also Galois fields) is outlined.

In general, a field is defined as a collection of elements that are subject to addition and multiplication operations along with some characteristics that control these operations.

1.1.1 Abelian Groups

Definition 1.1.1 *An Abelian group G is a set of elements $G = \{g_0, g_1, g_2, \dots\}$ and an operation $\oplus$ that satisfies the following axioms:*

- *Closure: For any g_i and $g_j \in G$ with $i \neq j$, the element $g_i \oplus g_j \in G$.*
- *Associative Law: For any $g_i, g_j, g_k \in G$, if $(g_i \oplus g_j) \oplus g_k = g_i \oplus (g_j \oplus g_k)$.*
- *Identity: There is an identity element $0 \in G$ for which $0 \oplus g_i = g_i \oplus 0 = g_i$.*
- *There is an inverse element $-g_i$ for each element such that $g_i \oplus (-g_i) = 0$.*

In addition, a group is called an abelian group if the commutative condition is satisfied such that $g_i \oplus g_j = g_j \oplus g_i$

1.1.2 Galois Fields

Definition 1.1.2 *A Galois (or finite) field is a set $\mathbb{F}$ of at least two elements, with two operations $\oplus$ and $\circledast$, satisfying the following axioms:*

- *The set $\mathbb{F}$ forms an abelian group (whose identity is called 0) under the operation $\oplus$.*
- *The set $\mathbb{F}^{\circledast} = F - \{0\} = g_i \in \mathbb{F}, g_i \neq 0$ forms an abelian group under the operation $\circledast$.*
- *For all $g_i, g_j, g_k \in \mathbb{F}, (g_i \oplus g_j) \circledast g_k = (g_i \circledast g_k) \oplus (g_j \circledast g_k)$.*

The operation $\oplus$ corresponds to field addition, and $\circledast$ corresponds to the field multiplication.

The set $\mathbb{F} = \{0, 1\}$ with modulo-2 addition ($\oplus$) and multiplication ($\circledast$) is an example of a binary Galois Field denoted as GF(2).

$\oplus$	0	1
0	0	1
1	1	0

$\circledast$	0	1
0	0	0
1	0	1

Table 1.1 – Addition ($\oplus$) and Multiplication ($\circledast$) Operations over GF(2).

A field of order $q = m^p$ is denoted as GF(q) if m is prime and p is a positive number.

Polynomials over Finite Fields

The polynomial $g(A)$ of degree p over a finite field of GF(q) is expressed as

$$g(A) = \sum_{i=0}^{m} g_i \cdot \alpha^i, \tag{1.1}$$

where $g_i \in$ GF(q) and $g_m \neq 0$.

The addition and multiplication operations follow the ordinary polynomial rules except that the operations of the coefficients are performed over modulo-q. If a polynomial cannot be written as a product of two lower-degree polynomials, then it is called an irreducible polynomial. In addition, if $a^m = 1$, the polynomial is called a monic polynomial.

Assume an irreducible and monic polynomial $f(A)$ of degree p. Consider all the polynomials with degrees less than p over GF(q) with ordinary polynomial addition and multiplication operations performed over modulo-q. The set of the resultant elements of the

addition and multiplication operations are the elements of the Galois field GF(q). The modulo-q multiplication entails multiplying two polynomials, dividing the product by the irreducible polynomial, and finding the remainder.

As an example, assume the irreducible polynomial $\alpha^3 + \alpha + 1$ for GF($8 = 2^3$), the elements of GF(8) are $\{0, 1, \alpha, \alpha{+}1, \alpha^2, \alpha^2{+}1, \alpha^2{+}\alpha, \alpha^2{+}\alpha{+}1\}$. Assume the multiplication operation $(\alpha + 1) \circledast 1$, it can be computed as $\mod(\frac{\alpha+1}{\alpha^3+\alpha+1})$, i.e., $\alpha^3 = \alpha + 1$. Similarly, all elements in GF(8) can be written as powers of α.

Therefore, the elements of GF(8) can be written in three ways, the polynomial form with a degree less than p, the power form with a degree less than $q - 2$, or the vector form which only considers the coefficient of the polynomial g_i. The three representations are depicted in Table 1.2 for GF(8) with the irreducible polynomial $f(A) = \alpha^3 + \alpha + 1$.

Vector	Polynomial	Power
000	0	$\alpha^{-\infty}$
001	1	α^0
010	α	α^1
100	α^2	α^2
011	$\alpha + 1$	α^3
110	$\alpha^2 + \alpha$	α^4
111	$\alpha^2 + \alpha + 1$	α^5
101	$\alpha^2 + 1$	α^6

Table 1.2 – Representation of Field Elements on GF(8).

The multiple representation of the field elements helps in simplifying the addition ($\oplus$) and multiplication ($\circledast$) operations. The power representation simplifies the multiplication operation $\circledast$ of two elements $\alpha^i \oplus \alpha^j$ such that the result is equal to an element with a power that is the sum of the operands power modulo $q-1$, i.e., $\alpha^i \circledast \alpha^j = \alpha^{(i+j) \mod (q-1)}$. As a proof example, assume $\alpha^6 \circledast \alpha = \alpha^7 = \frac{(\alpha^3+1)\alpha}{\alpha^3+\alpha+1} = 1$, i.e., $\alpha^7 = 1$. This can be easily obtained when using $\alpha^{7 \mod 7} = \alpha^0 = 1$.

On the other hand, the vector representation is used for the field addition operations which can be seen as the $p-$bit XOR of the two (vector-represented) elements. As an example, assume the addition of $(\alpha^2 + 1) \oplus (\alpha + 1)$. The polynomial element $(\alpha^2 + 1)$ is represented as $(101)_2$ and the polynomial $(\alpha + 1)$ is represented as $(011)_2$ in the vector representation. Thus, the sum in polynomial can be performed as $(101)_2$ XOR $(011)_2 = (110)_2$ which is the vector representation of the polynomial $\alpha^2{+}\alpha$, and the XOR represents the Exclusive OR logical gate.

The field multiplication and addition tables on GF(8) are as depicted in Table 1.3 and 1.4 respectively.

$\circledast$	0	1	α	α^2	α^3	α^4	α^5	α^6
0	0	0	0	0	0	0	0	0
1	0	1	α	α^2	α^3	α^4	α^5	α^6
α	0	α	α^2	α^3	α^4	α^5	α^6	1
α^2	0	α^2	α^3	α^4	α^5	α^6	1	α
α^3	0	α^3	α^4	α^5	α^6	1	α	α^2
α^4	0	α^4	α^5	α^6	1	α	α^2	α^3
α^5	0	α^5	α^6	1	α	α^2	α^3	α^4
α^6	0	α^6	1	α	α^2	α^3	α^4	α^5

Table 1.3 – Multiplication ($\circledast$) Operations over GF(8) using Power Representation.

1.2 Asymptotic and Finite Block-Length Achievable Code Rates

The theoretical bound of the maximal achievable rate plays a major role in assessing the performance of the error-correction codes. There are two regimes to deduce the upper bound; The asymptotic and the finite block lengths regimes. The asymptotic regime is discussed in section 1.2.1 and the finite block length regime is discussed in section 1.2.2 in brief.

1.2.1 Asymptotic Achievable Rate

The asymptotic maximal achievable rate also called the channel capacity, is the maximum possible coding rate at which information can be reliably transmitted for an asymptotic block length.

Channel capacity is defined mathematically as the maximum mutual information between the channel's input and output. In the context of channel coding, mutual information is defined as the amount of shared information between two random variables. Hence, the asymptotic maximal achievable rate can be expressed as follows

$$C \triangleq I(X;Y) = H(X) - H(X|Y), \tag{1.2}$$

$\oplus$	0	1	α	α^2	$\alpha+1$	$\alpha^2+\alpha$	$\alpha^2+\alpha+1$	α^2+1
0	0	1	α	α^2	$\alpha+1$	$\alpha^2+\alpha$	$\alpha^2+\alpha+1$	α^2+1
1	1	0	$\alpha+1$	α^2+1	α	$\alpha^2+\alpha+1$	$\alpha^2+\alpha$	α^2
α	α	$\alpha+1$	0	$\alpha^2+\alpha$	1	α^2	α^2+1	$\alpha^2+\alpha+1$
α^2	α^2	α^2+1	$\alpha^2+\alpha$	0	$\alpha^2+\alpha+1$	α	$\alpha+1$	1
$\alpha+1$	$\alpha+1$	α	1	$\alpha^2+\alpha+1$	0	α^2+1	α^2	$\alpha^2+\alpha$
$\alpha^2+\alpha$	$\alpha^2+\alpha$	$\alpha^2+\alpha+1$	α^2	α	α^2+1	0	1	$\alpha+1$
$\alpha^2+\alpha+1$	$\alpha^2+\alpha+1$	$\alpha^2+\alpha$	α^2+1	$\alpha+1$	α^2	1	0	α
α^2+1	α^2+1	α^2	$\alpha^2+\alpha+1$	1	$\alpha^2+\alpha$	$\alpha+1$	α	0

Table 1.4 – Addition ($\oplus$) Operations over GF(8) using Polynomial Representation.

where $I(X;Y)$ is the mutual information between channel input X and channel output Y, $H(X)$ is the information entropy of X, and $H(X|Y)$ is the conditional entropy of X given Y.

1.2.2 Finite-Block Achievable Rate

As known, the asymptotic regime is a theoretical bound that assumes an infinite block length. But in practice, the block lengths are finite and can be short lengths, medium lengths, or high lengths. Therefore, a more realistic regime was proposed by Y. Polyaniskiy in [21] that estimates the maximal achievable rates for finite-block lengths.

In the non-asymptotic regime, the maximal achievable rate is given by

$$R = C - \sqrt{\frac{V}{N}} Q^{-1}(P_e), \tag{1.3}$$

where C corresponds to the channel capacity expressed in (1.2), V is the channel dispersion, P_e is the desired error probability, and $Q^{-1}()$ is the inverse of the complementary Gaussian cumulative distribution function.

The channel dispersion V is expressed as

$$V = H_2(X|Y) - H^2(X|Y), \tag{1.4}$$

where $H_2(X|Y) \triangleq \mathbb{E}_y \left(-\sum P(x) \log^2(P(x)) \right)$.

Using (1.3), the minimum achievable error rate can be expressed as

$$P_e = Q\left(\frac{C - R}{\sqrt{V/N}} \right). \tag{1.5}$$

Thus, the deduced probability of error P_e is the minimum achievable error probability for a given code rate R, and code length N. It is used to assess the performance of the error-correcting codes by estimating the performance gap between the studied error-correction code and the minimum achievable error probability.

1.3 Cyclic Code Shift Keying Modulation

The Cyclic-Code Shift Keying (CCSK) modulation [22], [23] is a spread-spectrum modulation technique. It has been adopted in the Quasi Cyclic Small Packet (QCSP) project [24] for developing a new coded modulation scheme for IoT networks. In short, the association of the CCSK and the non-binary decoders has provided self-synchronization, self-identification, and operation at ultra-low signal-to-noise ratios.

In CCSK modulation, a message of size $p-$bits denoted as $\alpha \in \mathrm{GF}(q = 2^p)$, is modulated over a Pseudo-random Noise (PN) sequence of size q. Therefore, the message is spread by a spreading factor $S_F = \frac{p}{q}$.

Assume a fundamental PN sequence (corresponds to $\alpha = 0$) denoted as ψ_0 such that $\psi_0 = (\psi_0(0), \psi_0(1), \ldots, \psi_0(q-1))$ with $\psi_0(k) \in \{0, 1\}\ \forall\ k = 0, \ldots, q-1$. A PN sequence ψ_α is deduced for any symbol α from ψ_0 by circularly shifting the latter as follows:

$$\psi_\alpha(k) = \psi_0((k + \alpha) \mod q)\ \forall\ k = 0, \ldots, q-1. \tag{1.6}$$

Let $\psi = (\psi_{c_0}, \psi_{c_1}, \ldots, \psi_{c_{N-1}})$ denote the CCSK modulated frame of the codeword $c = (c_0, \ldots, c_{N-1})$ where c_i is a symbol of $\mathrm{GF}(q)$ (of size p-bits). The length of the frame is thus Nq bits (or chips). The chips of ψ are modulated over a Binary Phase Shift Keying (BPSK) and transmitted over a Binary Input Additive White Gaussian Noise (BI-AWGN) channel with noise variance σ^2. The i^{th} transmitted CCSK symbol ψ_{c_i} is received as a vector $y_i = (y_i(k))_{k=0,1,\ldots,q-1}$ with $y_i(k) = (2\psi_{c_i}(k) - 1) + w_i(k)$ where $w_i(k)$ represents a sample of the additive noise.

The LLR vector L_i computed from the channel observation is given as $L^i = (L_i(0), \ldots, L_i(q-1))$ with

$$L_i(\alpha) = \sum_{k=0}^{q-1} \frac{2y_i(k)}{\sigma^2}(\psi_{\hat{c}_i}(k) - \psi_\alpha(k)),\ \forall \alpha \in \mathrm{GF}(q) \tag{1.7}$$

where $\hat{c}_i$ represents the maximum likelihood decision on y_i defined as

$$\hat{c}_i = \underset{\alpha \in\ \mathrm{GF}(q)}{\mathrm{argmax}} \{\sum_{k=0}^{q-1} \frac{2y_i(k)}{\sigma^2}\psi_\alpha(k)\}. \tag{1.8}$$

Note that, by construction, the LLR values are all positive and $L(\hat{c}_i) = 0$. In addition, the generated LLR vectors L_i are inputted to the non-binary decoder as the intrinsic beliefs.

Chapter 2

Non-Binary Low-Density Parity Check Codes

Contents

2.1 Introduction to NB-LDPC Codes . . . 30
2.1.1 Structure of NB-LDPC Decoder . . . 31
2.1.2 Overview on NB-LDPC Code Construction . . . 32
2.2 NB-LDPC Iterative Decoding Algorithms . . . 33
2.2.1 Belief Propagation Algorithm . . . 33
2.2.2 Belief Propagation in Logarithmic Domain . . . 35
2.2.3 Min-Sum Algorithm . . . 37
2.2.4 Min-Max Algorithm . . . 37
2.3 Extended Min Sum Algorithm and Its Implementation Approaches . . . 37
2.3.1 Extended Min-Sum Algorithm . . . 38
2.3.2 Forward-Backward CN Processing . . . 40
2.3.3 Syndrome-Based CN Processing . . . 43
2.3.4 Presorted EMS Algorithm . . . 47
2.3.5 Hybrid Check Node Processing . . . 49
2.4 Trellis Extended Min-Sum Algorithm and Its Variants . . . 49
2.4.1 Trellis Extended Min-Sum Algorithm . . . 50
2.4.2 One Minimum Only Trellis-EMS . . . 52
2.4.3 Two-Extra Column Trellis EMS . . . 53
2.5 The Best, The Requested, and The Default Algorithm . . . 55
2.5.1 Introduction to the BRD Algorithm . . . 55
2.5.2 Statistical and Requested Symbols Analysis . . . 59
2.5.3 Trellis BRD Decoder . . . 61
2.5.4 Syndrome-based BRD Algorithm . . . 63

2.5.5 Forward-Backward BRD Decoder 68
2.5.6 Implementation of FB-BRD Check Node 75
2.6 Conclusion 89

2.1 Introduction to NB-LDPC Codes

NB-LDPC Codes were invented by R. Gallager in 1960 [4] and rediscovered by D. Mackay in 1996 [5], [25]. An NB-LDPC code [26] is a linear block code defined by a sparse Parity Check Matrix (PCM) $\mathbf{H}$ of dimension $M \times N$ over a Galois field GF($q = 2^p$) with $p > 1$. Therefore, the ratio of the non-zeros elements and the zero elements of the PCM tends to zero as the dimension of the PCM tends to infinity. An LDPC code allows the encoder to transform an information message of K symbols into a codeword of N symbols (using a generator matrix) by adding redundant information of $M = N - K$ symbols. The number of parity check constraints, M, represents the number of rows in $\mathbf{H}$, and the length of the codewords, N, represents the number of columns in $\mathbf{H}$. At the encoding stage, M redundant symbols are added to the K information symbols to obtain a codeword of size N. Moreover, the M parity constraints of $\mathbf{H}$ are used for verifying the validity of the received message. A codeword X is valid if and only if $X \cdot \mathbf{H}^T = 0$ where $\mathbf{H}^T$ is the transpose of the matrix $\mathbf{H}$.

Consider a PCM $\mathbf{H}$ with $N = 9, K = 3$ as follows

$$\mathbf{H} = \begin{bmatrix} h_{0,0} & 0 & 0 & h_{0,3} & 0 & 0 & h_{0,6} & 0 & 0 \\ 0 & h_{1,1} & 0 & 0 & h_{1,4} & 0 & 0 & h_{1,7} & 0 \\ 0 & 0 & h_{2,2} & 0 & 0 & h_{2,5} & 0 & 0 & h_{2,8} \\ h_{3,0} & 0 & 0 & 0 & h_{3,4} & 0 & 0 & 0 & h_{3,8} \\ 0 & h_{4,1} & 0 & 0 & 0 & h_{4,5} & h_{4,6} & 0 & 0 \\ 0 & 0 & h_{5,2} & h_{5,3} & 0 & 0 & 0 & h_{5,7} & 0 \end{bmatrix}$$

A codeword $X = [x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8]$ should satisfy six parity check equations as follows

$$\begin{aligned} x_0 \circledast h_{0,0} \oplus x_3 \circledast h_{0,3} \oplus x_6 \circledast h_{0,6} &= 0 \\ x_1 \circledast h_{1,1} \oplus x_4 \circledast h_{1,4} \oplus x_7 \circledast h_{1,7} &= 0 \\ x_2 \circledast h_{2,2} \oplus x_5 \circledast h_{2,5} \oplus x_8 \circledast h_{2,8} &= 0 \\ x_0 \circledast h_{3,0} \oplus x_4 \circledast h_{3,4} \oplus x_8 \circledast h_{3,8} &= 0 \\ x_1 \circledast h_{4,1} \oplus x_5 \circledast h_{4,5} \oplus x_6 \circledast h_{4,6} &= 0 \\ x_2 \circledast h_{5,2} \oplus x_3 \circledast h_{5,3} \oplus x_7 \circledast h_{5,7} &= 0 \end{aligned} \tag{2.1}$$

where any element of X and $\mathbf{H}$ is an element over a Galois field of order q, denoted as GF(q). The operators $\oplus$ and $\circledast$ depict the GF addition and multiplication over GF(q)

respectively.

2.1.1 Structure of NB-LDPC Decoder

An NB-LDPC decoder consists of VNs (VNs) and CNs (CNs). The interconnections of those nodes are determined by the PCM **H**. The interconnections of the variable and CNs can be visualized as a graph called a bipartite (or Tanner) graph [27] as illustrated in Fig. 2.1.

The code structure is well described using the Tanner graph such that each row in **H** corresponds to a CN. Similarly, each column in **H** corresponds to a VN. Moreover, the interconnections between the check and VNs are also determined by **H**. A CN C_i is related to a VN V_j if and only if the element of **H** at row i and column j is a non-zero element, ($h_{i,j} \neq 0$). The number of non-zero elements in each row is denoted as d_c and is known as the CN degree of connectivity, whereas, the number of non-zero elements in each column is denoted as d_v and is known as the VN degree of connectivity.

An LDPC code is said to be a regular or irregular code based on the degree of connectivity of the CNs d_c and the VN d_v. A regular LDPC code has constant degrees of connectivity d_v for all VNs and d_c for all CNs. In addition, the degree of connectivity d_c is expressed as $d_c = \frac{N}{M} \cdot d_v$. Assuming the PCM **H** is a full rank matrix, the coding rate r can be expressed in terms of d_c and d_v as

$$r = \frac{K}{N} = 1 - \frac{d_v}{d_c} \tag{2.2}$$

The regular NB-LDPC is more hardware-friendly than the irregular due to the static structure of the nodes (same d_c and d_v).

Since the PCM **H** of an NB-LDPC consists of symbols over GF($q = 2^p$) with $p > 1$, a permutation node is added to perform the multiplication of the symbols by the non-zero element of $h_{i,j}$ between each edge (a connection between a VN V_j and a CN C_i with $h_{i,j} > 0$). As illustrated in Fig.2.1, the Tanner graph shows that it consists of $N = 9$ VNs and $M = 6$ CNs with the permutation block in the middle of the connections. In addition, the illustrated graph and hence, the PCM is considered as a regular code since the number of CNs connected to each VN is constant, $d_v = 2$ and the number of VNs connected to each CN is constant, $d_c = \frac{N}{M} \cdot d_c$, $d_c = \frac{9}{6} \cdot 2 = 3$.

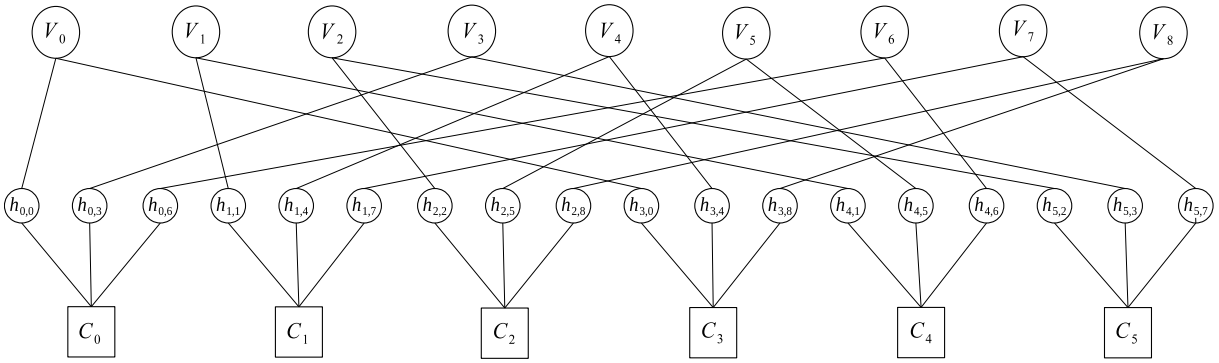


Figure 2.1 – A Graphical Representation of **H** with a Tanner Graph

2.1.2 Overview on NB-LDPC Code Construction

The construction of good NB-LDPC codes can be achieved through plenty of methods in the literature [28]–[33]. The design aims to find a PCM H that allows the decoder to achieve near-capacity performance with efficient encoding and decoding performance-complexity trade-off.

The assessment of the asymptotic performance of the NB-LDPC has been estimated using different approaches. A. Bennatan, et al. studied the error-correcting performance of NB-LDPC codes by applying the Maximum Likelihood decoding and the Belief Propagation algorithm as presented in [29], the author extended his work by including the development of extrinsic information transfer charts for arbitrary channels as presented in [30]. Furthermore, G. Li et al. studied the performance of the NB-LDPC codes using Gaussian approximation in [31].

The study presented in [33] by X. Hu proved that when the field order, q, increases, the decoding performance increases with ultra-sparse NB-LDPC codes. In addition, codes with the minimum VN degree of connectivity of $d_v = 2$ have been proposed and studied in [28] by C. Pouillat.

The construction of a good NB-LDPC PCM can be performed in two stages. The first stage consists of finding the positions of the non-zero elements of **H** that minimize the error floor. Then, the coefficients of the non-zero elements for a given row of **H** are selected based on the decoding performance at the waterfall region and the error floor region.

Finding the positions of the non-zero elements is related to the girth size of the obtained matrix **H**. The girth is the length of the shortest cycle (a path that starts and ends at the same node and visits no node more than once) in the Tanner graph. The higher the

girth, the better the decoding performance. This is because a code with a higher girth decreases the trapping sets in the message-passing algorithm. Thus, the impact of passing a bad belief is minimized. The work presented in [32] by L. Zeng et al. proposes using a Quasi-Cyclic PCM constructed from smaller photographs.

The coefficient selection of the non-zero elements can be obtained in an arbitrary approach (random selection) or concisely. The proposed approach in [28] relies on choosing almost optimal non-zero coefficients in the PCM for codes with a variable degree of $d_v = 2$. Once the non-zero coefficients are selected to optimize one PCM row, they are concisely allocated across the PCM to provide good cycle characteristics.

A large enough database of good performance Quasi-Cyclic NB-LDPC codes has been designed and allowed for public use in [34]. The performance of all codes provided throughout this chapter is based on the codes available in the aforementioned database.

2.2 NB-LDPC Iterative Decoding Algorithms

The decoding process of the NB-LDPC decoder is based on iterative message-passing decoding such that the beliefs are propagated from the VNs toward the corresponding CNs and vice versa.

2.2.1 Belief Propagation Algorithm

The Belief Propagation (BP) decoding algorithm is based on message passing between the VNs and the connected CNs at each iteration. The decoding algorithm should converge before reaching the maximum number of iterations ($iter_{max}$); otherwise, the decoding process is considered a failure. The BP algorithm is the decoding algorithm proposed in [26] to decode an NB-LDPC codeword.

Assume a transmitted codeword $X = [x_0, x_1, x_2, \cdots, x_{N-1}]$ such that $x_i \in GF(q)$, and a received codeword $Y = [y_0, y_1, \cdots, y_{N-1}]$. A decoding algorithm should converge from y towards a valid codeword $\hat{X} = [\hat{x_0}, \hat{x_1}, \cdots, \hat{x}_{N-1}]$, such that a successful decoding process is achieved when $\hat{X} = X$.

Each VN V_j has intrinsic information I_j computed from the channel observation where I_j is a vector of size q of a posteriori probabilities defined as

$$I_j = [\mathcal{P}(V_j = \alpha_0|y_j), \mathcal{P}(V_j = \alpha_1|y_j), \cdots, \mathcal{P}(V_j = \alpha_{q-1}|y_j)], \tag{2.3}$$

where $\mathcal{P}(X|Y)$ is the conditional probability of X given Y.

The decoding algorithm can be divided into the following steps:

1. Initialization: Each VN V_j transmits the intrinsic information I_j to its connected CNs via the message $M_{V_j2C_i}$.
2. Permutation: The variable-to-check message $M_{V_j2C_i}$ is multiplied by the element $h_{i,j}$ of the PCM $\mathbf{H}$ and a permuted version of the message $M_{V_j2C_i}$, denoted as $M^P_{V_j2C_i}$ is obtained as

$$M^P_{V_j2C_i}[\alpha] = M_{V_j2C_i}[\alpha \circledast h_{i,j}] \ \forall \ \alpha \in GF(q). \tag{2.4}$$

3. Check Node Update: The CN update yields the message $M^P_{C_i2V_j}$ computed as the combination of all symbols that satisfy the parity check constraint

$$M^P_{C_i2V_j}[\alpha] = \sum_{\substack{\sum_{\substack{k\neq j \\ h_{i,k}\neq 0}} \beta_k = \alpha}} \prod_{\substack{k\neq j \\ h_{i,k}\neq 0}} M^P_{V_j2C_i}[\beta_k] \ \forall \ \alpha, \ \beta_k \in GF(q). \tag{2.5}$$

4. Inverse Permutation: The message $M^P_{C_i2V_j}$ is computed from the permuted variable-to-check message $M^P_{V_j2C_i}$ and hence, an inverse permutation is performed by dividing the GF symbols of the message $M^P_{C_i2V_j}$ by the PCM element $h_{i,j}$. Therefore, the check-to-variable message is obtained as

$$M_{C_i2V_j}[\alpha] = M^P_{C_i2V_j}[\alpha \circledast h^{-1}_{i,j}] \ \forall \ \alpha \in GF(q) \tag{2.6}$$

5. Variable Node Update: A VN V_j receives d_v messages and updates them as follows

$$M_{V_j2C_i}[\alpha] = \eta_{v_jc_i} \times I_j[\alpha] \times \prod_{\substack{k\neq i \\ h_{k,j}\neq 0}} M_{C_k2V_j}[\alpha] \ \forall \alpha \in GF(q). \tag{2.7}$$

   where $\eta_{v_jc_i}$ is a normalization factor such that $\sum_{\alpha \in GF(q)} M_{V_j2C_i}[\alpha] = 1$.
6. Codeword Estimation: At each iteration, the a posteriori probability (APP) vector is computed as in (2.8), and the decision is taken as the most probable symbol in the a posteriori probability vector as in (2.9)

$$APP_j[\alpha] = \eta_{v_j} \times I_j[\alpha] \times \prod_{h_{i,j}\neq 0} M_{C_i2V_j}[\alpha]. \tag{2.8}$$

where η_{v_j} is a normalization factor such that $\sum_{\alpha \in GF(q)} APP_j[\alpha] = 1$.

$$\hat{x}_j = \arg\max_{\alpha \in GF(q)} \{APP_j[\alpha]\}, j = 0, 1, \cdots, N-1. \tag{2.9}$$

The messages exchanged between a VN and a CN (processed as aforementioned) are illustrated in Fig. 2.2 for ease of realization. The BP algorithm suffers from high complex-

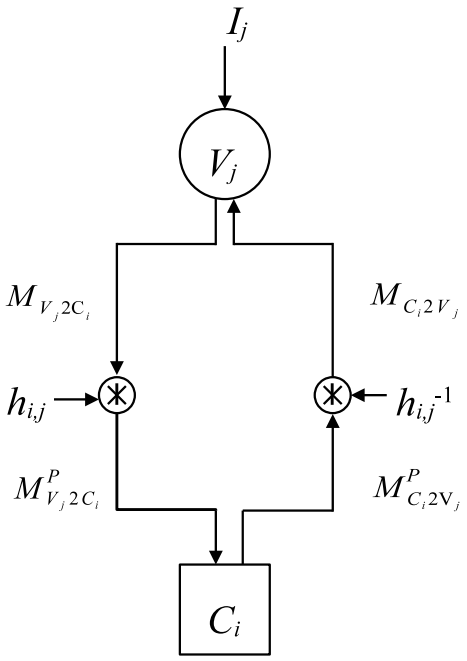


Figure 2.2 – Messages Exchanged Between a VN and a CN Edge.

ity. Therefore, different algorithms have been proposed to reduce the complexity of the BP algorithm such as the Logarithmic Belief Propagation.

2.2.2 Belief Propagation in Logarithmic Domain

The Logarithmic Belief Propagation (Log-BP), proposed by H. Wymeersch in [35], is a simplified BP algorithm that performs the computations of the BP algorithm in the logarithmic domain, such that a probability belief is represented by a Log-Likelihood Ratio (LLR).

An LLR value for a symbol α is defined as

$$LLR(\alpha) = \ln \frac{\mathcal{P}(v_j = \alpha | y_j)}{\mathcal{P}(v_j = \alpha_0 | y_j)}, \ \alpha, \ \alpha_0 \in GF(q), \tag{2.10}$$

where α_0 is a reference symbol. Replacing the probability metric with the LLR metric allows for reducing the hardware complexity of the decoder by converting all multiplication operations to addition. Moreover, it allows for a better representation of small numbers in fixed-point precision.

In the Log-BP decoding algorithm, the decoding processes are similar to that in the Belief Propagation with some modifications to the computations in (2.3), (2.5),(2.7), and (2.8). The intrinsic information I_j is redefined as

$$I_j[\alpha] = \ln\left(\frac{\mathcal{P}(v_j = \bar{\alpha}|y_j)}{\mathcal{P}(v_j = \alpha)|y_j}\right), \tag{2.11}$$

where $\bar{\alpha} = \arg\max_{\alpha \in GF(q)} \mathcal{P}(\alpha|y_j)$, i.e., $\bar{\alpha}$ is the hard decision. The advantage of using the aforementioned LLR ratio [36] is that all LLR values are positive and the one with a value of zero corresponds to the highest reliable symbol, i.e., $I_j(\bar{\alpha}) = 0$.

The following equation replaces the VN update in (2.7).

$$M_{V_j 2C_i}[\alpha] = I_j[\alpha] + \sum_{\substack{k \neq i \\ h_{k,j} \neq 0}} M_{C_k 2V_j}[\alpha] \ \forall \alpha \in \mathrm{GF}(q). \tag{2.12}$$

The CN C_i updates the message $M^P_{C_i 2V_j}$ sent to the VN V_j as follows

$$M^P_{C_i 2V_j}[\alpha] = \ln\left(\sum_{\substack{\sum_{\substack{k \neq j \\ h_{i,k} \neq 0}} \alpha_k = \alpha}} \exp\Big(\sum_{\substack{k \neq j \\ h_{i,k} \neq 0}} M^P_{V_k 2C_i}[\alpha_k]\Big)\right) \tag{2.13}$$

The message $M^P_{C_i 2V_j}$ is inversely permuted to obtain the message $M_{C_i 2V_j}$. The a posteriori information APP_j of the VN V_j is updated as

$$APP_j[\alpha] = I_j[\alpha] + \sum_{h_{i,j} \neq 0} M_{C_i 2V_j}[\alpha] \ \forall \ \alpha \in GF(q) \tag{2.14}$$

The main drawback of the log-BP algorithm is that the complexity of the decoder increases to an unimplementable level at high field levels due to the exponential and logarithmic functions. Therefore, the min-sum algorithm was proposed to approximate some metrics such that the complexity is further reduced.

2.2.3 Min-Sum Algorithm

The Min-Sum algorithm, proposed by M. Fossorier in [37], is an approximated decoding algorithm that approximates the CN update computed as (2.13) such that the exponential and logarithmic functions are eliminated. This allows for reducing the look-up tables dedicated to the exponential and logarithmic functions.

The min-sum approximates the CN outputs, the permuted check-to-variable messages, as follows

$$M^P_{C_i 2V_j}[\alpha] \approx \min_{\substack{\sum_{\substack{k\neq j \\ h_{i,k}\neq 0}} \alpha_k = \alpha}} \{ \sum_{\substack{k\neq j \\ h_{i,k}\neq 0}} M^P_{V_k 2C_i}[\alpha_k] \} \tag{2.15}$$

Even though the min-sum algorithm eliminates the logarithmic and exponential functions by using mathematical approximations, the CN processing still suffers from complexity of order $\mathcal{O}(d_c q^2)$.

2.2.4 Min-Max Algorithm

The min-max algorithm, proposed by V. Savin in [38], is another approximating decoding algorithm that replaces the addition of the min-sum algorithm for further complexity reduction. The CN processing in (2.15) is modified such that

$$M^P_{C_i 2V_j}[\alpha] \approx \min_{\substack{\sum_{\substack{k\neq j \\ h_{i,k}\neq 0}} \alpha_k = \alpha}} \{ \max_{\substack{k\neq j \\ h_{i,k}\neq 0}} M^P_{V_k 2C_i}[\alpha_k] \} \tag{2.16}$$

2.3 Extended Min Sum Algorithm and Its Implementation Approaches

The EMS algorithm aims to reduce the complexity of the MS algorithm which is of order $\mathcal{O}(d_c q^2)$ such that a NB-LDPC decoder is implementable at high field orders such as $q \geq 32$. This is achieved by reducing the size of the exchanged messages, which, in turn, reduces the overall complexity of the CNs.

In this section, the general algorithm of the Extended Min-Sum (EMS) is explained in section 2.3.1. The algorithm can be implemented using different approaches such as the Forward-Backward approach discussed in section 2.3.2 or using the Syndrome-Based

approach discussed in section 2.3.3. In addition, the presorting approach combined with the EMS CN is discussed in section 2.3.4.

2.3.1 Extended Min-Sum Algorithm

The Extended Min-Sum (EMS) is a truncated version of the min-sum algorithm such that the messages exchanged are truncated from q-ary into an n_m-ary vector [39], [40] by extracting in a sorted ordered the n_m most reliable candidates from the propagated messages. In Log-BP, an exchanged message A is a vector of LLRs where each $LLR(x)$ corresponds to its index x. Hence, $A = [LLR(\alpha_0), LLR(\alpha_1), \cdots, LLR(\alpha_{q-1})]$. In the EMS algorithm, the message A is truncated to the message B such that the vector B includes the most reliable n_m GF symbols and their corresponding LLR values sorted in the descending order of their reliability (ascending order of the LLR values).

For clearer insight, assume a message A including the reliability of $q = 8$ symbols as shown in Fig. 2.3. Also, assume that $n_m = 4$ such that the vector B contains the most reliable elements of vector A. Each element in A is of one tuple which corresponds to the reliability of the index symbol. On the contrary, each element of B is of two tuples, the GF symbol, and its corresponding reliability estimate. In the sequel, the vector that includes the first tuple of all elements (GF symbols) is represented as $B^{\oplus}$. Similarly, the LLR vector of B is represented as B^{+}. Hence the vector $B = (B^{\oplus}, B^{+})$. In the provided example, the 4 most reliable candidates of A are at indices $\alpha^2, \alpha^5, 0$, and α^1 and their corresponding LLR values are $0, 2, 3,$ and 5 respectively. Therefore, both the indices (symbols) and their LLR values are copied in the same order to B.

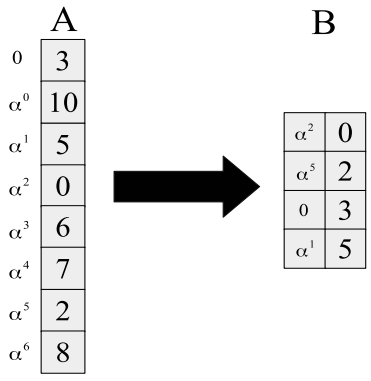


Figure 2.3 – Example of Message Truncation in EMS algorithm

The EMS algorithm can be formalized as follows,

1. Initialization: Each VN V_j sends the n_m most reliable candidates in the intrinsic information to its connected CNs.

2. Variable Node Update: Each VN receives d_v messages $M_{C_i 2V_j}$ and a default value κ_i from the connected CNs C_i where the default value κ_i is used to compensate the degradation resulted due to the truncation, and computed as follows

$$\kappa_i = M^{+}_{C_i 2V_j}[n_m - 1] + O \tag{2.17}$$

where O is an offset value determined empirically to reduce the error rate. The reliability of the variable-to-check message $M^{\oplus}_{V_j 2C_i}(m)_{m=0,1,\cdots,n_m-1}$ is updated as

$$M^{+}_{V_j 2C_i}(m) = I_j[M^{\oplus}_{V_j 2C_i}(m)] + \sum_{\substack{k \neq i \\ h_{k,j} \neq 0}} M_k(m) \tag{2.18}$$

such that

$$M_k(m) = \begin{cases} M^{+}_{C_k 2V_j}[M^{\oplus}_{V_j 2C_i}(m)] & \text{if } M^{\oplus}_{V_j 2C_i}(m) \in M^{\oplus}_{C_k 2V_j} \\ \kappa_i & \text{otherwise} \end{cases} \tag{2.19}$$

3. Normalization: Each variable-to-check message $M_{V_j 2C_i}$ is normalized as follows,

$$M^{+}_{V_j 2C_i}(m) = M^{+}_{V_j 2C_i}(m) - \min(M^{+}_{V_j 2C_i}(m)). \tag{2.20}$$

The normalization process (in both MS and EMS) is performed after every iteration to avoid a significant increase in the LLR value which leads to arithmetic overflow.

4. Permutation: Each symbol in $M_{V_j 2C_i}$ is multiplied by the non-zero $h_{i,j}$ to obtain the permuted vector $M^{P\oplus}_{V_j 2C_i}$

$$M^{P\oplus}_{V_j 2C_i}(m) = M^{\oplus}_{V_j 2C_i}(m) \circledast h_{i,j} \text{ for } m = 0, 1, \cdots, n_m - 1 \tag{2.21}$$

5. Check Node Update: The CN updates the permuted check-to-variable message

$M^{P}_{C_i 2V_j}$ as

$$M^{P+}_{C_i 2V_j}(m) \approx \min_{\substack{\sum_{k\neq j} M^{P\oplus}_{V_k 2C_i}(\omega)=\alpha \\ h_{i,k}\neq 0}} \{ \sum_{\substack{k\neq j \\ h_{i,k}\neq 0}} M^{P+}_{V_k 2C_i}(\omega)\} \text{ for } 0 \leq \omega \leq n_m - 1$$
$$M^{P\oplus}_{C_i 2V_j}(m) = \alpha. \tag{2.22}$$

6. Inverse Permutation: The symbols of $M^{P}_{C_i 2V_j}$ are divided by $h_{i,j}$ to obtain the message $M_{C_i 2V_j}$.
$$M^{\oplus}_{C_i 2V_j}(m) = M^{P\oplus}_{C_i 2V_j}(m) \circledast h^{-1}_{i,j} \tag{2.23}$$

7. Codeword Estimation: Each VN V_j updates the a posteriori information vector APP_j as:

$$APP_j[\alpha] = I_j[\alpha] + \sum_{h_{i,j}\neq 0} M_i(\alpha) \ \forall \ \alpha \in GF(q) \tag{2.24}$$

where

$$M_i[\alpha] = \begin{cases} M^{+}_{C_i 2V_j}[\alpha], & \text{if } \alpha \in M^{\oplus}_{C_i 2V_j} \\ \kappa_i & \text{Otherwise} \end{cases}$$

The EMS algorithm reduces the complexity of the NB-LDPC decoder down to $\mathcal{O}(d_c n_m^2)$ such that the decoder can be implemented on high-field orders such as $q \geq 32$.

2.3.2 Forward-Backward CN Processing

The Forward-Backward (FB) approach is an approach proposed in [41] to efficiently implement the EMS algorithm. It reduces the complexity of the EMS-based CN unit by exploiting the (commutative and associative) properties of the addition operation in GF(q). In the FB approach, the CN unit is decomposed into three layers namely, the forward, the backward, and the merge where each layer consists of $d_c - 2$ Elementary Check Node (ECNs).

An example is shown in Fig. 2.4 for a degree of $d_c = 5$. In each layer, there are $d_c - 2 = 3$ ECNs. As shown, the decomposition of the CN into multiple ECNs reduces the redundant computations by ensuring that the maximum information is reused (not recomputed) at the merge layer.

An ECN unit is a processing unit that has two input vectors A and B, and an output

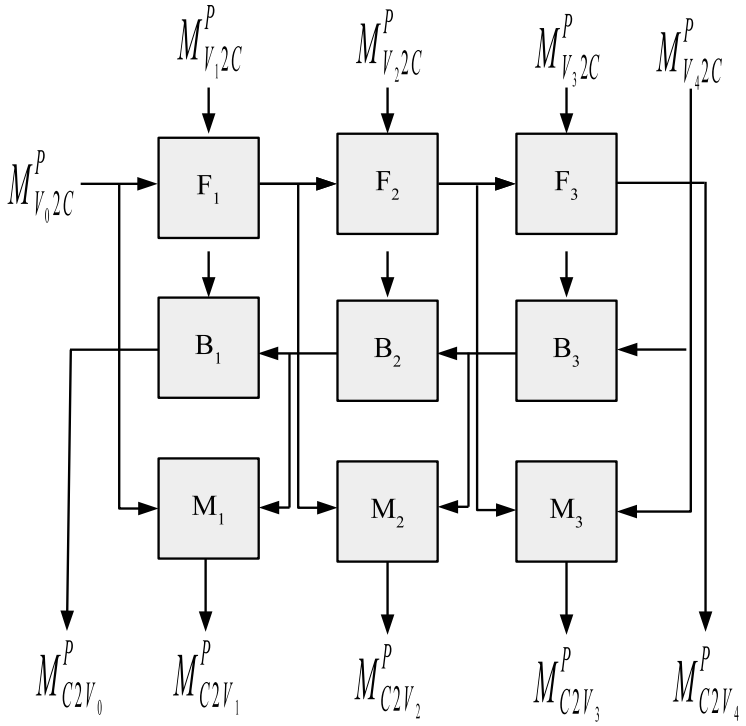


Figure 2.4 – Forward-Backward Architecture for $d_c = 5$

vector C each of size n_m. Each element in the vectors is of two tuples, a GF and an LLR value. The notation $A^{\oplus}$ represents the GF vector, whereas A^{+} represents the LLR vector of the vector A.

An ECN performs three operations sequentially,

1. **Addition**: An intermediate matrix T_{Σ} is computed as follows

$$\begin{aligned} T^{+}_{\Sigma}[a][b] &= A^{+}[a] + B^{+}[b], \\ T^{\oplus}_{\Sigma}[a][b] &= A^{\oplus}[a] \oplus B^{\oplus}[b] \\ &: a, b = 0, \cdots, n_m - 1. \end{aligned} \tag{2.25}$$

   where $T^{\oplus}_{\Sigma}$ and T^{+}_{Σ} correspond to the matrices of the GF symbols and the LLR values respectively.

2. **Sorting**: After obtaining the matrix T_{Σ}, the matrix is sorted in ascending order of T^{+}_{Σ} and therefore, the vector C is obtain by selecting the most reliable n_{op} candidates of T_{Σ}.

3. **Redundancy Elimination**: if two elements in the vector C, $C^{\oplus}[a]$ and $C^{\oplus}[b]$ cor-

respond to the same GF element with $a < b$, then the element $C^+[b]$ is suppressed by assigning the highest LLR value (lowest reliability). To ensure that n_m valid candidates are obtained after the redundancy elimination, n_{op} should always be greater than n_m.

The forward-backward approach tries to implement the EMS efficiently. However, the computation of the matrix T_Σ is still intensive since it performs n_m^2 GF additions and $(n_m - 1)^2$ LLR additions. The number of LLR addition operations is less than the GF addition operations because the first row and column of T_Σ corresponds to the (normalized) LLR input vectors since $T_\Sigma^+[a][0] = A^+[a] + B^+[0]$ and $T_\Sigma^+[0][b] = A^+[0] + B^+[b]$ respectively with $A^+[0] = 0$ and $B^+[0] = 0$.

Simplified ECN Computations

To reduce the computations performed in T_Σ required to generate the n_m, the authors in [42] proposed the bubble check processing for the ECNs. The bubble check allows for reducing the GF and LLR addition operations down to $n_m\sqrt{n_m}$ and $n_m(\sqrt{n_m} - 2) + 1$ operations while maintaining the same performance as the conventional ECN processing of T_Σ.

Furthermore, the L-bubble approach presented in [43] considered a fraction of the computations performed in T_Σ to generate the n_m output elements. It considers only the first two rows and columns of T_Σ to generate the output vector. Therefore, four regions R_0, R_1, R_2, and R_3 are computed as follows

$$\begin{aligned} R_0 &= A[0] \boxplus B[b], b = 0, \cdots, n_m - 1 \\ R_1 &= A[a] \boxplus B[0], a = 1, \cdots, n_m - 1 \\ R_2 &= A[1] \boxplus B[b], b = 1, \cdots, n_m - 1 \\ R_3 &= A[a] \boxplus B[1], a = 2, \cdots, n_m - 1 \\ &: a, b = 0, \cdots, n_m - 1. \end{aligned} \tag{2.26}$$

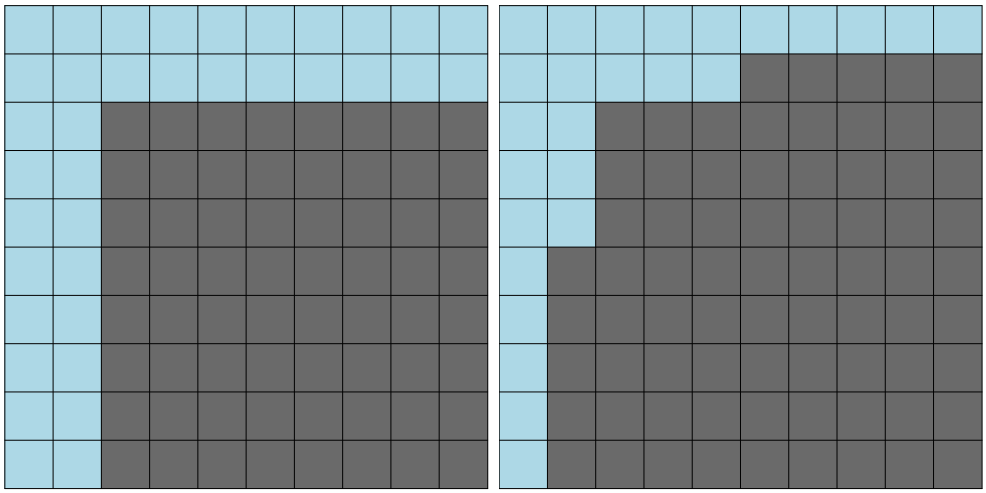
Consequently, the n_m most reliable (distinct) candidates are obtained using the regions mentioned above only. As a result, the number of GF and LLR additions are reduced to $4n_m - 4$ and $2n_m - 3$ respectively.

Besides, the authors in [44] proposed an optimized version of the L-bubble, called the S-bubble sorter. The authors proved that computing only the first row and column, along with the first half of the second row and column, yields a similar outcome to

computing the first two rows and columns. Therefore, no degradation is experienced in the decoding performance. The computed regions R_0, R_1, R_2, and R_3 can be re-expressed as the following

$$\begin{aligned} R_0 &= A[0] \boxplus B[b], b = 0, \cdots, n_m - 1 \\ R_1 &= A[a] \boxplus B[0], a = 1, \cdots, n_m - 1 \\ R_2 &= A[1] \boxplus B[b], b = 1, \cdots, n_m/2 - 1 \\ R_3 &= A[a] \boxplus B[1], a = 2, \cdots, n_m/2 - 1 \\ &: a, b = 0, \cdots, n_m - 1. \end{aligned} \tag{2.27}$$

For illustrative purposes, the schematic of the computed regions of T_Σ with $n_m = 10$ using the L-bubble and the S-bubble approaches are depicted in Fig. 2.5(a) and Fig. 2.5(b) respectively.



(a) L-Bubble Sorter (b) S-Bubble Sorter

Figure 2.5 – Potential Elements of T_Σ Computed using Bubble Sorters.

2.3.3 Syndrome-Based CN Processing

Syndrome-based (SYN) CN processing is an alternative approach to forward-backward CN processing proposed in [45] that offers higher parallelism. In this approach, the CN relies on deviation paths to compute the syndromes.

A syndrome-based CN has a basic structure as illustrated in Fig. 2.6. As shown, once the processing of the sorting unit is performed, the d_c output messages $M^P_{C2V_j}$ are generated in parallel. Therefore, a higher level of parallelism can be achieved.

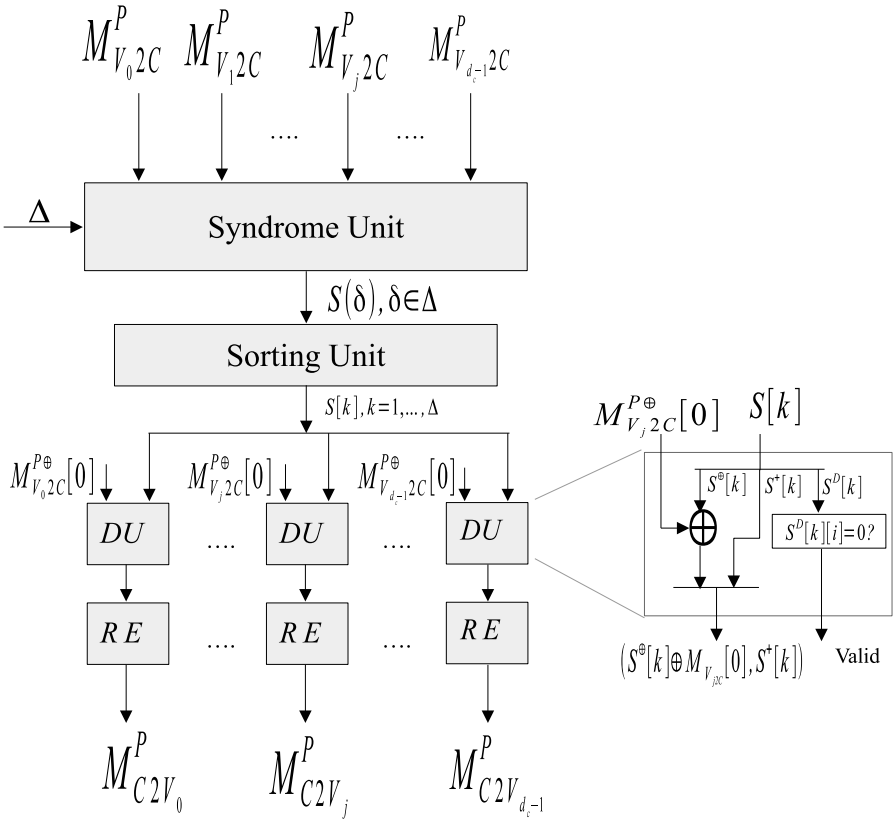


Figure 2.6 – Syndrome-based EMS Decoder [45]

The syndrome-based CN relies on the deviation paths for obtaining the syndromes. A deviation path denoted as δ is a d_c-tuple $\delta = \{\delta(0), \delta(1), \cdots, \delta(d_c - 1)\}$ and $\delta(j) \in 0, 1, \cdots, n_{vc} - 1,\ j = 0, \cdots, d_c - 1$ where n_{vc} is the size of the CN input messages $M^P_{V_j 2C}$. A syndrome $S(\delta)$ is the syndrome obtained by adding the elements of the deviation path δ and defined as $(S^+(\delta), S^{\oplus}(\delta), S^D(\delta))$.

The syndrome $S(\delta)$ is expressed as

$$\begin{aligned} S^+(\delta) &= \sum_{j=0}^{d_c-1} M^{P+}_{V_j 2C}[\delta(j)] \\ S^{\oplus}(\delta) &= \bigoplus_{j=0}^{d_c-1} M^{P\oplus}_{V_j 2C}[\delta(j)] \\ S^D(\delta)(j) &= \begin{cases} 0 & \text{if } \delta(j) = 0 \\ 1 & \text{otherwise} \end{cases} \end{aligned} \tag{2.28}$$

where $S^+(\delta)$ is the LLR value, $S^{\oplus}(\delta)$ is the GF symbol, and $S^D(\delta)$ is the Discard Binary Vector (DBV) of the syndrome $S(\delta)$ of size d_c bits. The DBV is used at the decorrelation stage of the syndrome decoder.

Assume Δ_A is the set of all possible deviation paths, i.e., $\Delta_A = \{0, 1, \cdots, n_{vc} - 1\}^{d_c}$. Then, the set Δ is a subset of Δ_A that reduces the total computed syndromes without majorly affecting the decoding performance. The output message $M^P_{C2V_j}$ corresponding to the VN V_j is computed from the syndromes $S(\delta),\ \delta \in \Delta$ with $S^D(\delta)[j] = 0$ by the decorrelation unit (DU) and the redundancy elimination block (RE) as follows

$$\begin{aligned} M^{P\oplus}_{C2V_j}[k] &= S^{\oplus}(\delta) \oplus M^{P\oplus}_{V_j 2C}[0]; \\ M^{P+}_{C2V_j}[k] &= \min_{\delta \in \Delta,\ S^D(\delta)[j]=0} S^+(\delta); \end{aligned} \qquad k = 0, \cdots, n_{cv} - 1. \tag{2.29}$$

where n_{cv} denotes the size of the check-to-variable messages.

The size of the deviation paths Δ is based on the deviation parameters d_1 and d_2 [45]. The deviation parameters specify the maximum size of the deviation paths such that only d_k elements from each VN can contribute to obtaining a syndrome with a maximum of k deviations from the zeroth element of the VN messages. As an example, assume that $d_1 = 3$ and $d_2 = 2$. Since $d_1 = 3$, only the first three elements in the variable-to-check messages can contribute to one-deviation paths. For clarity, the full deviation paths for $d_1 = 3$ are as depicted in Table 2.1 for a CN degree $d_c = 3$.

Similarly, the full syndromes set for $d_2 = 2$ is depicted in Table 2.2.

δ	$S^{\oplus}(\delta)$	$S^{+}(\delta)$	$S^{D}(\delta)$
100	$M^{P\oplus}_{V_0 2C_i}[1] \oplus M^{P\oplus}_{V_1 2C_i}[0] \oplus M^{P\oplus}_{V_2 2C_i}[0]$	$M^{P+}_{V_0 2C_i}[1] + M^{P+}_{V_1 2C_i}[0] + M^{P+}_{V_2 2C_i}[0]$	100
010	$M^{P\oplus}_{V_0 2C_i}[0] \oplus M^{P\oplus}_{V_1 2C_i}[1] \oplus M^{P\oplus}_{V_2 2C_i}[0]$	$M^{P+}_{V_0 2C_i}[0] + M^{P+}_{V_1 2C_i}[1] + M^{P+}_{V_2 2C_i}[0]$	010
001	$M^{P\oplus}_{V_0 2C_i}[0] \oplus M^{P\oplus}_{V_1 2C_i}[0] \oplus M^{P\oplus}_{V_2 2C_i}[1]$	$M^{P+}_{V_0 2C_i}[0] + M^{P+}_{V_1 2C_i}[0] + M^{P+}_{V_2 2C_i}[1]$	001
200	$M^{P\oplus}_{V_0 2C_i}[2] \oplus M^{P\oplus}_{V_1 2C_i}[0] \oplus M^{P\oplus}_{V_2 2C_i}[0]$	$M^{P+}_{V_0 2C_i}[2] + M^{P+}_{V_1 2C_i}[0] + M^{P+}_{V_2 2C_i}[0]$	100
020	$M^{P\oplus}_{V_0 2C_i}[0] \oplus M^{P\oplus}_{V_1 2C_i}[2] \oplus M^{P\oplus}_{V_2 2C_i}[0]$	$M^{P+}_{V_0 2C_i}[0] + M^{P+}_{V_1 2C_i}[2] + M^{P+}_{V_2 2C_i}[0]$	010
002	$M^{P\oplus}_{V_0 2C_i}[0] \oplus M^{P\oplus}_{V_1 2C_i}[0] \oplus M^{P\oplus}_{V_2 2C_i}[2]$	$M^{P+}_{V_0 2C_i}[0] + M^{P+}_{V_1 2C_i}[0] + M^{P+}_{V_2 2C_i}[2]$	001
300	$M^{P\oplus}_{V_0 2C_i}[3] \oplus M^{P\oplus}_{V_1 2C_i}[0] \oplus M^{P\oplus}_{V_2 2C_i}[0]$	$M^{P+}_{V_0 2C_i}[3] + M^{P+}_{V_1 2C_i}[0] + M^{P+}_{V_2 2C_i}[0]$	100
030	$M^{P\oplus}_{V_0 2C_i}[0] \oplus M^{P\oplus}_{V_1 2C_i}[3] \oplus M^{P\oplus}_{V_2 2C_i}[0]$	$M^{P+}_{V_0 2C_i}[0] + M^{P+}_{V_1 2C_i}[3] + M^{P+}_{V_2 2C_i}[0]$	010
003	$M^{P\oplus}_{V_0 2C_i}[0] \oplus M^{P\oplus}_{V_1 2C_i}[0] \oplus M^{P\oplus}_{V_2 2C_i}[3]$	$M^{P+}_{V_0 2C_i}[0] + M^{P+}_{V_1 2C_i}[0] + M^{P+}_{V_2 2C_i}[3]$	001

Table 2.1 – Syndrome Set for $d_1 = 3$

δ	$S^{\oplus}(\delta)$	$S^{+}(\delta)$	$S^{D}(\delta)$
110	$M^{P\oplus}_{V_0 2C_i}[1] \oplus M^{P\oplus}_{V_1 2C_i}[1] \oplus M^{P\oplus}_{V_2 2C_i}[0]$	$M^{P+}_{V_0 2C_i}[1] + M^{P+}_{V_1 2C_i}[1] + M^{P+}_{V_2 2C_i}[0]$	110
011	$M^{P\oplus}_{V_0 2C_i}[0] \oplus M^{P\oplus}_{V_1 2C_i}[1] \oplus M^{P\oplus}_{V_2 2C_i}[1]$	$M^{P+}_{V_0 2C_i}[0] + M^{P+}_{V_1 2C_i}[1] + M^{P+}_{V_2 2C_i}[1]$	011
101	$M^{P\oplus}_{V_0 2C_i}[1] \oplus M^{P\oplus}_{V_1 2C_i}[0] \oplus M^{P\oplus}_{V_2 2C_i}[1]$	$M^{P+}_{V_0 2C_i}[1] + M^{P+}_{V_1 2C_i}[0] + M^{P+}_{V_2 2C_i}[1]$	101
120	$M^{P\oplus}_{V_0 2C_i}[1] \oplus M^{P\oplus}_{V_1 2C_i}[2] \oplus M^{P\oplus}_{V_2 2C_i}[0]$	$M^{P+}_{V_0 2C_i}[1] + M^{P+}_{V_1 2C_i}[2] + M^{P+}_{V_2 2C_i}[0]$	110
012	$M^{P\oplus}_{V_0 2C_i}[0] \oplus M^{P\oplus}_{V_1 2C_i}[1] \oplus M^{P\oplus}_{V_2 2C_i}[2]$	$M^{P+}_{V_0 2C_i}[0] + M^{P+}_{V_1 2C_i}[1] + M^{P+}_{V_2 2C_i}[2]$	011
102	$M^{P\oplus}_{V_0 2C_i}[1] \oplus M^{P\oplus}_{V_1 2C_i}[0] \oplus M^{P\oplus}_{V_2 2C_i}[2]$	$M^{P+}_{V_0 2C_i}[1] + M^{P+}_{V_1 2C_i}[0] + M^{P+}_{V_2 2C_i}[2]$	101
210	$M^{P\oplus}_{V_0 2C_i}[2] \oplus M^{P\oplus}_{V_1 2C_i}[1] \oplus M^{P\oplus}_{V_2 2C_i}[0]$	$M^{P+}_{V_0 2C_i}[2] + M^{P+}_{V_1 2C_i}[1] + M^{P+}_{V_2 2C_i}[0]$	110
021	$M^{P\oplus}_{V_0 2C_i}[0] \oplus M^{P\oplus}_{V_1 2C_i}[2] \oplus M^{P\oplus}_{V_2 2C_i}[1]$	$M^{P+}_{V_0 2C_i}[0] + M^{P+}_{V_1 2C_i}[2] + M^{P+}_{V_2 2C_i}[1]$	011
201	$M^{P\oplus}_{V_0 2C_i}[2] \oplus M^{P\oplus}_{V_1 2C_i}[0] \oplus M^{P\oplus}_{V_2 2C_i}[1]$	$M^{P+}_{V_0 2C_i}[2] + M^{P+}_{V_1 2C_i}[0] + M^{P+}_{V_2 2C_i}[1]$	101
220	$M^{P\oplus}_{V_0 2C_i}[2] \oplus M^{P\oplus}_{V_1 2C_i}[2] \oplus M^{P\oplus}_{V_2 2C_i}[0]$	$M^{P+}_{V_0 2C_i}[2] + M^{P+}_{V_1 2C_i}[2] + M^{P+}_{V_2 2C_i}[0]$	110
022	$M^{P\oplus}_{V_0 2C_i}[0] \oplus M^{P\oplus}_{V_1 2C_i}[2] \oplus M^{P\oplus}_{V_2 2C_i}[2]$	$M^{P+}_{V_0 2C_i}[0] + M^{P+}_{V_1 2C_i}[2] + M^{P+}_{V_2 2C_i}[2]$	011
202	$M^{P\oplus}_{V_0 2C_i}[2] \oplus M^{P\oplus}_{V_1 2C_i}[0] \oplus M^{P\oplus}_{V_2 2C_i}[2]$	$M^{P+}_{V_0 2C_i}[2] + M^{P+}_{V_1 2C_i}[0] + M^{P+}_{V_2 2C_i}[2]$	101

Table 2.2 – Syndrome Set for $d_2 = 2$

This yields a total of deviation paths (similarly, syndromes), denoted as n_{SYN} and deduced as

$$n_{SYN} = 1 + d_c \times d_1 + \binom{d_c}{d_2} \times (d_2)^2 \tag{2.30}$$

The total syndromes size is the size of the two-deviations syndromes, one-deviation syndromes, and the zero-deviation (only one syndrome) $S(\delta) = (0, M^{P\oplus}_{V_0 2C_i}[0] \oplus M^{P\oplus}_{V_1 2C_i}[0] \oplus M^{P\oplus}_{V_2 2C_i}[0], 000)$. The value of d_1 and d_2 is mostly set as $n_{vc}-1$ and 2 respectively as provided in [45].

2.3.4 Presorted EMS Algorithm

The presorting algorithm [46], [47] proposed by C. Marchand and H. Harb is an algorithm used to polarize the statistics of the CN input messages such that the internal processing of the CNs is simplified, hence, the complexity is reduced.

The presorting algorithm relies on presorting the variable-to-check messages $M^P_{V_j 2C}, j = 0, \cdots, d_c - 1$ before the CN processing. The statistical polarization of the inputs is achieved by sorting the messages based on the LLR difference between the first and second element of $M^P_{V_j 2C_i}$, $M^{P+}_{V_j 2C_i}[0]$ and $M^{P+}_{V_j 2C_i}[1]$ respectively. The presorting helps in reducing the size of the messages $M^P V_j 2C_i$ by eliminating the elements with high LLR values that are unlikely to contribute to an output, and hence, reducing the number of computed bubbles/syndromes.

A concrete example of the presorting algorithm is illustrated in Fig. 2.7. Firstly, assume a CN with four ($d_c = 4$) input messages $M^P_{V_j 2C}$ (denoted also as U_j) for $j = 0, \cdots, d_c - 1$. The second element of the messages $M^P_{V_j 2C}$, i.e., $M^P_{V_j 2C}[1]$ is processed by the presorting block such that the output of the presorter, π is the index of the variable-to-check messages U_j ordered in an ascending of the LLR value inputted. In the example, the entry $U_0[1]$ has the least LLR value, then, the entry $U_2[1]$, after that, the entry $U_3[1]$ and lastly, the entry $U_1[1]$. Therefore, the output of the presorter is $\pi = [0, 2, 3, 1]$. The output of the presorter is an input to the switch which permutes the message U_j to $U^{'}_j$ based on the indices vector π. For illustration purposes, assume that j entries are discarded at each message index j such that the size of the message $U^{'}_0$ is the highest and the size of the message $U^{'}_{d_c-1}$ is the least. The reason behind this is that the difference between the hard decision of the message, i.e., $U^{'}_0[0]$ and the second reliable element $U^{'}_0[1]$ is too low, therefore, both elements should be considered as potential elements since the VN is not reliable. Contrary, since the gap between the LLR value of the first and the second element of

U_3' is significant, the remaining candidates are unlikely to contribute to obtaining a valid output since the VN is confident about its hard decision. Furthermore, the permuted messages are processed by the CN using one of the CN processing algorithms and then, the output of the CN V_j' is inversely permuted by π^{-1} to V_j and sent to the corresponding VNs as shown.

The presorting technique has been integrated with both the forward-backward algorithm in [46] and the syndrome-based algorithm in [47] for further complexity reduction of the NB-LDPC decoder.

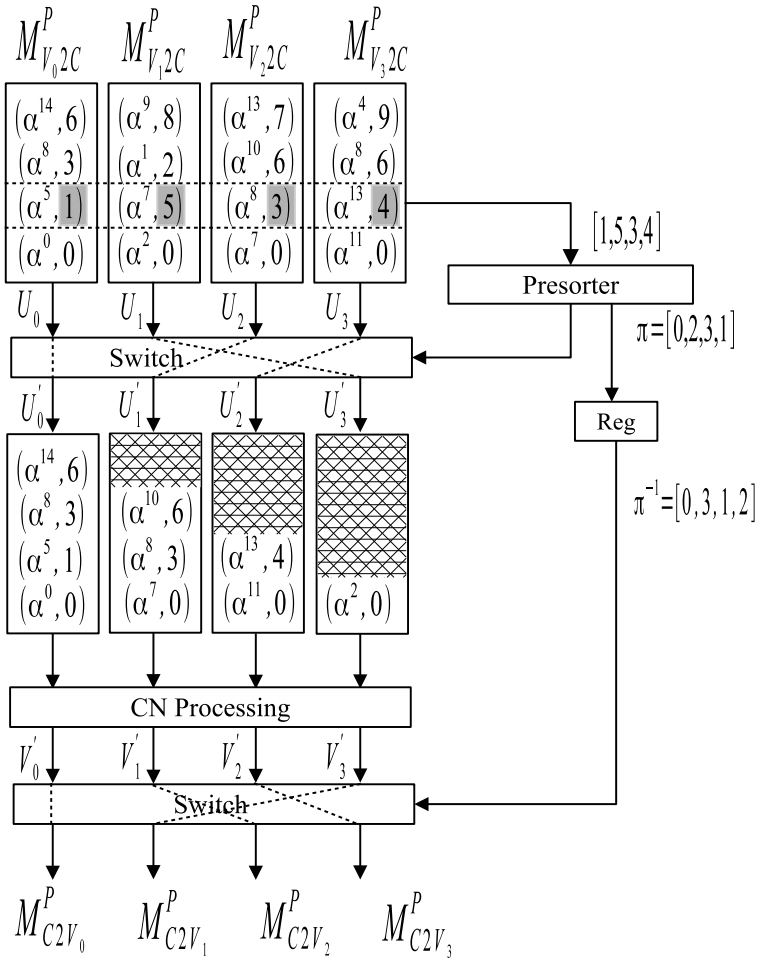


Figure 2.7 – Presorting Algorithm [47]

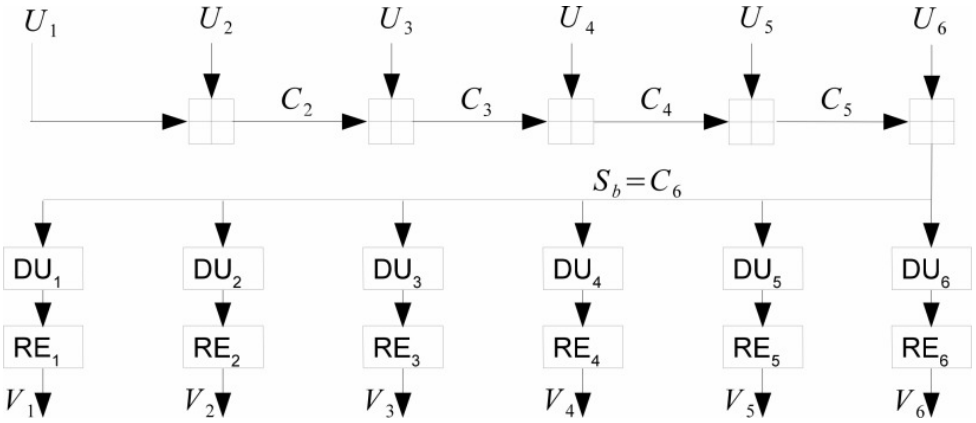


Figure 2.8 – Extended Forward Check Node for $d_c = 6$.

2.3.5 Hybrid Check Node Processing

The CN proposed in [48] is the first hybrid CN that relies on the forward-backward and the syndrome-based approaches to enhance the efficiency of the CN. This hybrid processing is named the Extended Forward (EF) approach. In the EF approach, the forward-layer with $d_c - 1$ ECNs is processed and passed in parallel to $d_c - 1$ decorrelator units and the redundancy elimination blocks as shown in Fig. 2.8.

Later in [49], C. Marchand et al. extended the hybrid processing by including the presorting and the backward layer to generate the outputs efficiently. Therefore, the proposed hybrid relies on three processes, the extended forward, the syndrome-based, and the forward-backward approaches. Hence, three parameters ρ_{SN}, ρ_{EF} and ρ_{FB} are defined that correspond to the number of inputs processed by the syndrome-based nodes, extended forward block, and the forward-backward layer respectively such that $\rho_{SN} + \rho_{EF} + \rho_{FB} = d_c$. According to the authors, the hybrid architecture designed for $d_c = 12$ over GF(64) and GF(256) increased the power and hardware efficiency by a factor of 6.

2.4 Trellis Extended Min-Sum Algorithm and Its Variants

The Trellis Extended Min-Sum (TEMS) algorithm, initially proposed in [50] by E. Li et al., is an alternative way to efficiently process the CNs with a high degree of connectivity d_c. It is re-proposed in [51] with a low latency approach as a solution to reduce the high latency of the EMS algorithm at high code rates, i.e., at high d_c. In this section, the

algorithm description of the Trellis extended min-sum is presented in section 2.4.1. In addition, two simplification approaches called the One Minimum Only and Two-Extra Columns are presented in section 2.4.2 and section 2.4.3 respectively.

2.4.1 Trellis Extended Min-Sum Algorithm

The Trellis extended min-sum presented in [50]–[52] is based on transforming the input messages into another domain called the delta domain which allows the extraction of the reliable LLR values for each symbol concurrently and building the configuration sets using the trellis representation of the input messages. In addition, the TEMS, contrary to the EMS, does not require the truncation and the sorting of the variable-to-check messages.

At the beginning of the CN processing, each variable-to-check message $M^{P}_{V_j 2C}$ is transferred to another domain called the delta message domain. The delta domain transformation processes the message $M^{P}_{V_j 2C}$ for obtaining the delta messages $M^{\Delta}_{V_j 2C}$ for all $j = 0, \cdots, d_c - 1$ by adding the symbol of the most reliable element (having an LLR value 0), $\bar{\alpha}_j$, of the message $M^{P}_{V_j 2C}$ to all other elements as follows

$$M^{\Delta}_{V_j 2C}[\beta = \bar{\alpha}_j \oplus \alpha] = M^{P}_{V_j 2C}[\alpha] \ \forall \ \alpha, \ \bar{\alpha}_j \ \in \ GF(q) \tag{2.31}$$

Since the delta message $M^{\Delta}_{V_j 2C}$ includes the LLR values of q GF elements, the index of each entry in the vector represents the corresponding GF symbol. The concatenated delta messages can be viewed as a $d_c \times q$ matrix denoted as $\mathbf{M}^{\Delta}$ where each row corresponds to d_c LLR values (from each VN) of a GF symbol. Moreover, the element at index 0 ($\beta = 0$) of the delta messages $M^{\Delta}_{V_j 2C}$ is always the most reliable GF element of the VNs V_j. For each row in the remaining $q - 1$ entries, only n_r (lowest) LLR values (out of the d_c available LLR values) are considered for building the configuration sets.

Let $\mathcal{A}$ denote the subset of the considered trellis nodes for building the configuration sets such that a trellis node in $\mathcal{A}$ are represented as tuples (η, j) where η is the row (symbol) index and j is the column (VN) index. Moreover, assume two subsets of $\mathcal{A}$, $\mathcal{A}_j$ and $\mathcal{A}^{\eta}$ that include the possible row elements (symbols) of the j-th column and the possible column elements (VNs) of the η-th row (symbol η) respectively such that

$$\mathcal{A} = \bigcup_{j=0}^{d_c-1} \mathcal{A}_j = \bigcup_{\eta=0}^{q-1} \mathcal{A}^{\eta} \tag{2.32}$$

The configuration set that includes up to n_c deviations using n_r out of d_c LLR values

for each symbol is denoted as $Tconf_\eta(n_r, n_c)$ and defined as

$$Tconf_\eta(n_r, n_c) = \{\eta = [\eta_0, \eta_1, \cdots, \eta_{d_c-1}]^T : \eta = \bigoplus_{j=0}^{d_c-1} \eta_j, (\eta_j, j) \in \mathcal{A}, |\{\eta_j : \eta_j \geq 1\}| \leq n_c\}. \quad (2.33)$$

To compute the check-to-variable (delta) reliability messages for the q GF symbols, an extra column denoted as ΔW of size q is defined as

$$\Delta W[\eta] = \min_{\eta \in \mathcal{T}_\eta(n_r, n_c)} \sum_{j=0}^{d_c-1} M^{\Delta}_{V_j 2C}[\eta_j] \ \forall \ \eta \in \ GF(q) \quad (2.34)$$

Let the most reliable configuration used to obtain $\Delta W[\eta]$ be denoted as $\eta_R = [\eta_{R0}, \cdots, \eta_{R_{d_c-1}}]$. The elements $M^{\Delta}_{C2V_j}[\eta]$ for $j = 0, \cdots, d_c - 1$ are updated as follows

$$M^{\Delta}_{C2V_j}[\eta] = \begin{cases} \Delta W[\eta] & \text{if } \eta_{R_j} = 0 \\ \Upsilon[\eta] & \text{otherwise} \end{cases} \quad (2.35)$$

Some of the entries of $M^{\Delta}_{C2V_j}[\eta]$aren't updated. Those entries are updated with the value $\Upsilon[\eta]$. The value of $\Upsilon[\eta]$ is obtained based on the number of deviations in η_R, $D = |\{\eta_{R_j} : \eta_{R_j} \geq 1\}|$.

$$\Upsilon[\eta] = \begin{cases} \mathbf{M}^{\Sigma}[\eta][\mathcal{A}^\eta(0)] \\ \mathbf{M}^{\Sigma}[\eta][\mathcal{A}^\eta(1)] \quad \text{if D==1} \end{cases} \quad (2.36)$$

where $\mathcal{A}^\eta(0)$ and $\mathcal{A}^\eta(1)$ denote the columns of the first and the second minimum LLR values of the symbol η in the matrix $\mathbf{M}^{\Delta}$.

The last step required to obtain the normal domain check-to-variable message $M^{P}_{C2V_j}$ is the delta inverse transformation, performed as follows

$$M^{P}_{C2V_j}[\beta = \eta \oplus \iota \oplus \bar{\alpha}_j] = M^{\Delta}_{C2V_j}[\eta] \ \forall \ \eta \in \ GF(q), j = 0, \cdots, d_c - 1 \quad (2.37)$$

where $\iota = \sum_{j=0}^{d_c-1} \bar{\alpha}_j$.

The symbol β is the normal domain symbol corresponding to the symbol η in the delta domain. Since the symbol η is obtained using the delta domain symbols $\eta_{R_j j} = 0, \cdots, d_c - 1$, each symbol η_{R_j} is permuted at the delta transformation by adding to it the most reliable symbol $\bar{\alpha}_j$ as in (2.31). Therefore, the symbol β could be obtained from

the delta domain symbol η by subtracting the symbol of $\bar{\alpha}_j$ from each symbol η_{R_j}, which is equivalent to subtracting the symbol ι from η as expressed in (2.37).

A concrete example on GF(4) with a CN degree $d_c = 5$ in Fig. 2.9 illustrates the T-EMS algorithm. The first step is to transform the variable-to-check messages from the normal to the delta domain. In the example, the most reliable element of the VNs V_j for $j = 0, \cdots, d_c - 1$ are $\bar{\alpha} = 1, \alpha, (1+\alpha), 1, 0$ respectively. In addition, the symbol ι is computed as $\bigoplus_{j=0}^{d_c-1} \bar{\alpha}_j = 1 \oplus \alpha \oplus (1 \oplus \alpha) \oplus 1 \oplus 0 = 1$. The most reliable element of V_j is added to all elements to obtain the new message $M^{\Delta}_{V_j 2C}$ based on (2.31) as depicted in the figure. The d_c columns are aggregated to form the matrix M^{Δ}. The nodes of the set $\mathcal{A}$ are depicted in shaded-blue boxes of M^{Δ}. The configuration sets are built based on (2.33). In the example, the vector ΔW is computed based on (2.34). The element $\Delta W[\eta = 0]$ is always 0 (corresponding to the path with no deviations). For $\Delta W[\eta = 1]$, it is shown in the figure (blue line) that the configuration that yields the lowest LLR (highest reliability) is a one-deviation configuration since $\eta_R^{(\eta=1)} = 0 + 1 + 0 + 0 + 0$ with an LLR value equal to 1. Similarly, the lowest LLR value for $\eta = \alpha$ is for the two-deviation configuration $\eta_R^{(\eta=\alpha)} = 0 + 1 + 0 + 0 + (1 + \alpha)$ with an LLR of 3. After updating the vector ΔW, the vector $M^{\Delta}_{C2V_j}$ is updated based on (2.35). In the example, the row $M^{\Delta}_{C2V_j}[\beta = 0]$ is all set to 0 since they are obtained from the 0-order configurations. For $M^{\Delta}_{C2V_j}[\beta = 1]$, it is updated by $\Delta W[1]$ unless the symbol at index j of the configuration $\eta_R^{\eta=1}$ is different from 0. Therefore, the element $M^{\Delta}_{C2V_2}[\beta = 1]$ is updated with the second minimum since the configuration $\eta_R^{\eta=1}$ is a one-deviation configuration. For the row $M^{\Delta}_{C2V_j}[\beta = \alpha]$, since the configuration $\eta_K^{\eta=\alpha}$ contains two deviation at indices 2 and 5 respectively. The reliability value at those locations in the row $M^{\Delta}_{C2V_j}[\beta = \alpha]$ is set with the first minimum LLR value in $M^{\Delta}_{V_j 2C}$. After obtaining the messages $M^{\Delta}_{C2V_j}$, the messages are inversely transformed back to the normal domain as depicted in (2.37). As an example, the message $M^{\Delta}_{C2V_1}$ is transformed to M_{C2V_0} by subtracting ι and keeping $\bar{\alpha}_0$ such that each symbol β is shifted to $\beta \oplus \iota \oplus \bar{\alpha}_0$. Since $\iota \oplus \bar{\alpha}_0 = 0$, then the vector $M^{P}_{C2V_0}$ is a simple copy of $M^{\Delta}_{C2V_0}$. Moreover, the message $M^{P}_{C2V_1}$ is generated by permuting the indices $\beta \in GF(4)$ of $M^{\Delta}_{C2V_1}$ by $\iota \oplus \bar{\alpha}_1 = 1 + \alpha$ such that the element $M^{\Delta}_{C2V_1}[0]$ correspond to the LLR of the element $M^{P}_{C2V_1}[1 + \alpha]$ and so on.

2.4.2 One Minimum Only Trellis-EMS

In the Trellis-EMS, the $n_r = 2$ minimum LLR values (most reliable) are extracted for each symbol α $\forall \alpha \in GF(q)$, thus, two minima are extracted per symbol. The second

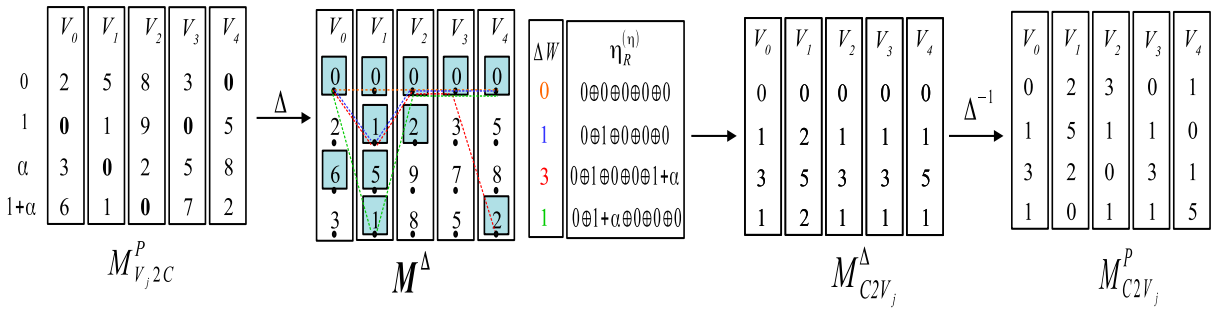


Figure 2.9 – Trellis-EMS Example on GF(4)

minimum value is used for updating the one-deviation configurations as in (2.36).

In One-Minimum Only TEMS (OMO-TEMS) proposed in [53] by J. Lacruz et al. , the second minimum is never extracted but approximated to reduce the complexity of the CN processing. This allows for reducing the $q-1$ two-minima block by a radix-2 one minimum finder.

Assume the minimum LLR value for each symbol α is denoted as $min_1(\alpha)$, and the second (estimated) minimum LLR value for the symbol α is then denoted as $min_2^*(\alpha)$. In OMO-TEMS the value of $min_2^*(\alpha)$ is computed as

$$\begin{aligned} min_2^*(\alpha) &= \frac{min_2^{''}(\alpha) + min_2^{'''}(\alpha)}{2} \\ min_2^{''}(\alpha) &= min_1(\alpha) \times \gamma_p \end{aligned} \tag{2.38}$$

The value of $min_2^{'''}(\alpha)$ is obtained by a radix-2 one minimum as shown in Fig. 2.10. The CS block in the radix-2 one minimum finder corresponds to compare-and-swap where the output of the block is the ascending values of the inputs. The parameter γ_p is estimated by simulation and computed as the mean of the ratio between $min_2(a)$ and $min_1(a)$.

2.4.3 Two-Extra Column Trellis EMS

The conventional Trellis EMS suffers from high complexity with parameters $n_r = 2$ and $n_c = 3$ due to the large configuration set. Therefore, the Two-Extra Column (TEC) Trellis, initially proposed by H. Thi et al. for Trellis Min-Max [54], is adopted in [55] for the Trellis EMS. The TEC approach aims to reduce the number of configurations in the configuration sets (2.33). This is achieved by the appendage of an additional column denoted as ΔW_2 beside the column ΔW.

In TEC-TEMS, the two columns ΔW and ΔW_2 are used for updating the delta check-

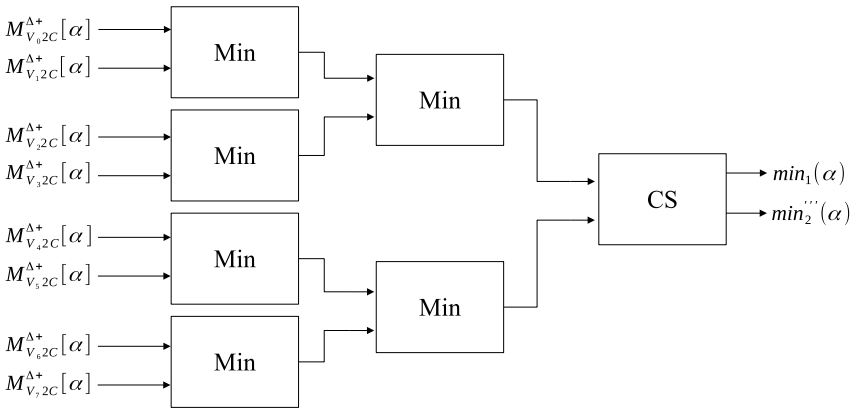


Figure 2.10 – Radix-2 One Minimum Finder

to-variable $M^{\Delta}_{C2V_j}$. The column ΔW contains the reliability of the most reliable configuration η_R for the symbol, $\eta \; \forall \; \eta \; \in \; GF(q)$ as defined in (2.34). In addition, the column ΔW_2 includes the reliability of the second most reliable configuration for the symbol $\eta, \; \forall \; \eta \; \in \; GF(q)$, and computed as

$$\Delta W_2[\eta] = \min_{\eta \in \mathcal{T}_\eta(n_r,n_c), \eta \neq \eta_R} \sum_{j=0}^{d_c-1} M^{\Delta}_{V_j2C}[\eta_j] \; \forall \; \eta \in \; GF(q) \tag{2.39}$$

In the original TEMS, the unfilled entries of $M^{\Delta}_{C2V_j}, j = 0, \cdots, d_c - 1$ are updated with the first or the second minimum of their corresponding symbols in $M^{\Delta}_{V_j2C}$ depending on the number of deviations in the configuration as in (2.35) and (2.36). Contrary, in TEC-TEMS, the unfilled entries (deviated) are filled by the vector ΔW_2 hence, the check-to-variable update equation in (2.35) is modified such that

$$M^{\Delta}_{C2V_j}[\eta] = \begin{cases} \Delta W[\eta] & \text{if } \eta_{R_j} = 0 \\ \Delta W_2[\eta] & \text{otherwise} \end{cases} \tag{2.40}$$

The aforementioned appendage allows for reducing the size of the configuration sets from $n_r = 2$ and $n_c = 3$ as in the conventional TEMS down to $n_r = 1$ and $n_c = 2$. Hence, reducing the computational complexity of the configuration sets of (2.33). However, even though the trellis extended-min sum algorithm reduces the complexity at high coding rates, it still suffers from high complexity that increases as the field order q increases.

2.5 The Best, The Requested, and The Default Algorithm

This section presents a novel algorithm called the Best, the Requested, and the Default (BRD) algorithm [56] that has been developed during this PhD. The algorithm has been published as a general NB-LDPC algorithm in [56] and a patent has been filed [57]. In addition, the implementation of the BRD algorithm using the forward-backward approach has been published in [58].

The section includes an introduction to the BRD algorithm in section 2.5.1, then, the integration of the BRD algorithm with different CN processing algorithms such as the EMS and TEMS algorithms and their different implementations.

2.5.1 Introduction to the BRD Algorithm

The BRD algorithm is a generic decoding algorithm for the NB-LDPC decoders compatible with any CN processing algorithms such as the EMS or TEMS-based decoders.

In the BRD algorithm, the VN requests the reliability of specific symbols from to the CN. This leads to a check-to-variable message that consists of three subsets, the best candidates having the highest reliability, the requested candidates, and the default candidates that are the least reliable among the two subsets.

After the CN processes the information sent from the connected VNs, the CN sends to each connected VN the LLR values of the requested symbols. The requested symbols allow for reducing the size of the exchanged messages, i.e., the communication load at the variable and the CN edges. Hence, allows for reducing the decoding complexity by reducing the complexity of the sorting processes, arithmetic operations, and memory allocations.

The BRD algorithm includes four processes in between each node edge as shown in Fig. 2.11, used for compressing and decompressing the messages exchanged, and for processing the requested symbols discussed in the following sections. Note that the CN index i is omitted in the sequel for simplicity and readability of the notations.

Compression and Decompression of M_{V2C} Message

The compression block Ω generates the message $\Omega(M_{V2C})$ from M_{V2C} by selecting the n_{vc} smallest LLRs and their associated GF value. Since the smallest LLR is always equal to 0, $\Omega(M_{V2C})$ is composed of $(n^+, n^{\oplus}) = (n_{vc}-1, n_{vc})$ LLR and GF elements respectively.

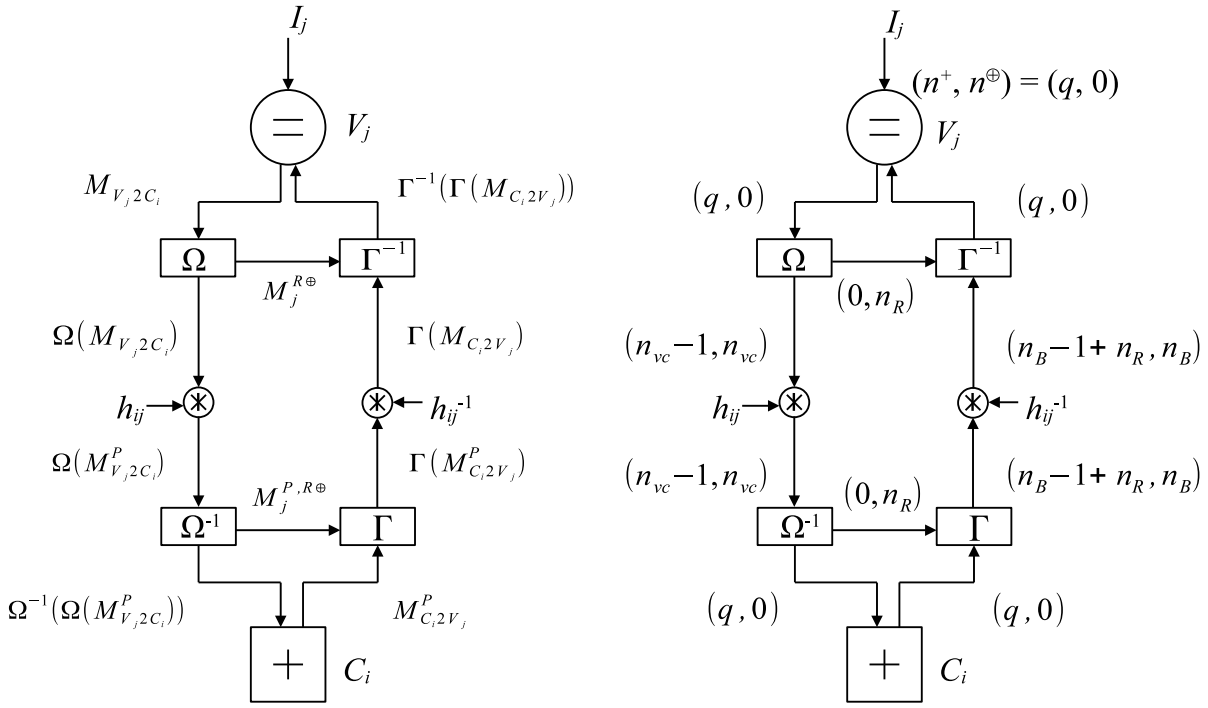


Figure 2.11 – The Best, the Requested, and the Default Decoder

The compression block Ω also sends to the decompression block Γ^{-1} the message $M^{R,\oplus}$ that is composed of the n_R requested GF symbols, i.e. the GF values of the first n_R couples of $\Omega(M_{V_j2C})$ as a result the constraint $n_R \leq n_{vc}$ should be always fulfilled. The edge multiplicative factor h is applied to each GF element of $\Omega(M_{V_j2C})$ to generate the permuted message $\Omega(M^P_{V_j2C})$.

The decompression module Ω^{-1} decompresses the message $\Omega(M^P_{V_j2C})$ back to a message of size q (depending on the used CN processing algorithm) by setting the n_{vc} GF values of $\Omega(M_{V_j2C})$ with their corresponding LLRs and by setting an infinite value for the remaining GF values (such that they never contribute to an output). Moreover, the decompression block Ω^{-1} also sends to the compression block Γ the message $M^{P,R,\oplus}$ containing the n_R permuted requested symbols extracted from the message $\Omega(M^P_{V_j2C})$.

Compression and Decompression of M_{C2V_j} Message

Once all the variable-to-check messages are received, the CN processes the $M^P_{V_j2C}$ messages using any processing algorithm such as EMS or TEMS. The generated $M^P_{C2V_j}$ message is then truncated by the Γ compression block in two steps. Firstly, the n_B most reliable candidates are selected to generate the subset $M^{P,B}_{C2V_j}$ of size $(n^+, n^\oplus) = (n_B - 1, n_B)$. Secondly, the LLR of the n_R requested symbols of $M^{P,R\oplus}$ are extracted from $M^P_{C2V_j}$ and concatenated with $M^{P,B}_{C2V_j}$ to generate the $\Gamma^(M^P_{C2V_j})$ message of a total size $(n^+, n^\oplus) = (n_B + n_R - 1, n_B)$. The n_B GF symbols are inversely permuted by h^{-1} to generate $\Gamma(M_{C2V_j})$ and sent to Γ^{-1}.

The Γ^{-1} decompression block reconstructs the $q-$ary (full-set) message $\Gamma^{-1}(\Gamma(M_{C2V_j}))$ (depending on the CN processing algorithm). The reconstruction consists of three processes. Firstly, the LLRs of the best candidates are placed into their corresponding GF positions. Then, the LLR of the requested symbols is set into their corresponding GF positions with a saturation process that prevents the reliability of a requested symbol from being greater than the default reliability S_R. Finally, the remaining positions are filled with the default LLR value S_D. In summary, for a given GF value $a \in GF(q)$,

$$\Gamma^{-1}(\Gamma(M^+_{C2V_j}))[a] = \begin{cases} M^{B+}_{C2V_j}[a], & a \in M^{B\oplus}_{C2V_j} \\ \min\left(M^{R+}_{C2V_j}[a], S_R\right), & a \in M^{R\oplus}_j \\ S_D, & Otherwise \end{cases} \tag{2.41}$$

The saturation values of S_R and S_D significantly impact the decoder performance. By

empirical analysis, S_R and S_D are determined as $S_R = S + O_R$ and $S_D = S + O_D$, with S given as a linear function of the maximum LLR values of $M^{B+}_{C2V_j}$ and $M^{R+}_{C2V_j}$, i.e.

$$S = \gamma_B \cdot \max\{M^{B+}_{C2V_j}\} + \gamma_R \cdot \max\{M^{R+}_{C2V_j}\} \tag{2.42}$$

where the values of $\gamma_B, \gamma_R, O_R, O_D$ can be found by empirically estimating the error rate.

Toy Example on the BRD Algorithm over GF(8).

Assume an edge between a VN V_j and a CN C_i as shown in Fig.2.12. The VN V_j sends a message $M_{V_j2C_i} = [7, 1, 12, 4, 18, 9, 0, 8]$ with each LLR element corresponding to its associated GF symbol (the index of the vector). The GF symbols are represented in the figure using distinct colors for better illustration.

The compression block Ω extracts the $n_{vc} = 3$ most reliable elements within the vector $M_{V_j2C_i}$ to generate a compressed message $\Omega(M_{V_j2C_i})$. As a result, two vectors are associated to generate $\Omega(M_{V_j2C_i})$, the GF vector $\Omega(M_{V_j2C_i})^{\oplus} = [\alpha^5, \alpha^0, \alpha^2]$ and the corresponding LLR vector $\Omega(M_{V_j2C_i})^{+} = [0, 1, 4]$. In the example, the coefficient $h_{i,j}$ is assumed to be 1 for simplicity.

Once the n_{vc} elements are propagated to the CN, the decompression block Ω^{-1} decompresses the message $\Omega(M_{V_j2C_i})$ back to a $q-$ary vector. The decompression procedure is dependent on the CN processing such that the decompression is necessary for algorithms such as the min-sum or the trellis extended min-sum CN processing. However, when using the extended min-sum algorithm, the decompression can be easily omitted. In the example, a $q-$ary processing is assumed, and therefore, the message $\Omega(M_{V_j2C_i})$ is decompressed to obtain the message $\Omega^{-1}(\Omega(M_{V_j2C_i}))$. The decompressed message is used by the CN to update the $d_c - 1$ connected VNs.

At the outgoing edge from C_i to V_j, the generated message $M^{P}_{C_i2V_j}$ is assumed to be $M^{P}_{C_i2V_j} = [0, 9, 4, 7, 11, 10, 8, 3]$. The Γ compression block compresses $M^{P}_{C_i2V_j}$ by extracting the $n_B = 2$ most reliable elements along with the LLR values of the $n_R = 2$ requested symbols. The requested symbols are a subset of the $\Omega(M_{v_j2C_i})$ message and correspond to the n_R most reliable elements of the n_{vc} elements. As shown in the example illustrated in Fig. 2.12, the requested message is of size $n_R = 2$, and therefore, the GF values of the two most reliable elements (α^5, α^0) are passed to the compression and decompression blocks, Γ and Γ^{-1} respectively. The compressed message $\Gamma(M_{C_i2V_j})$ consists of n_B GF symbols and $n_B - 1$ (excluding the propagation of LLR 0) LLR elements, along with n_R

LLR elements. After that, the message is propagated to the VN and decompressed based on (2.42) to assign the default value (depicted as D in the vector) for the $q-(n_B+n_R)$ remaining elements.

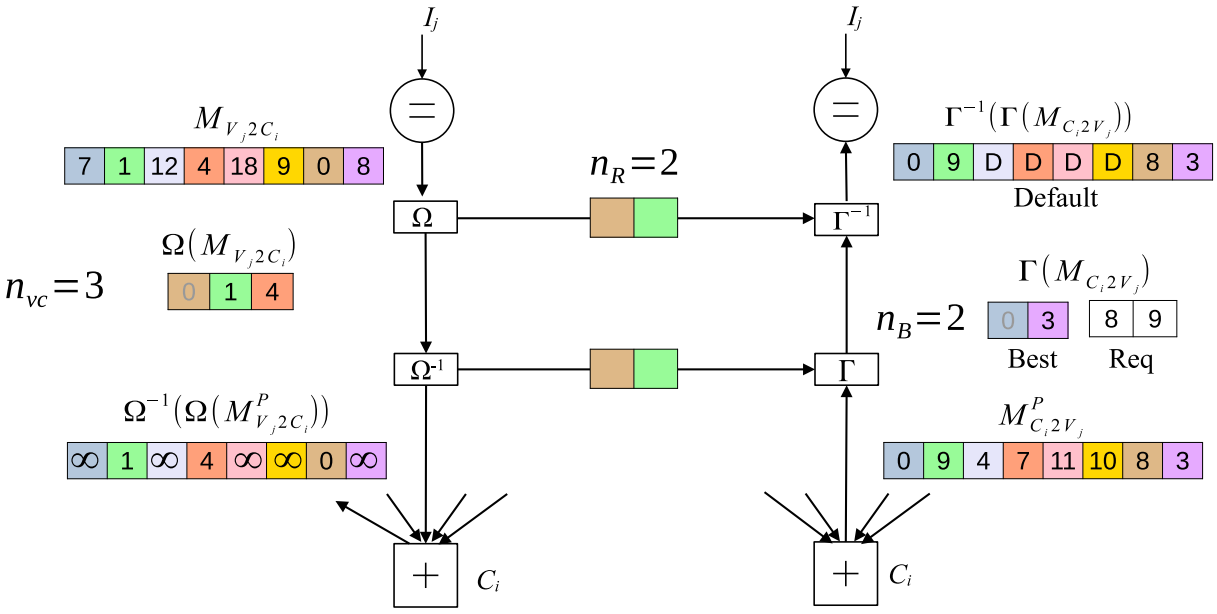


Figure 2.12 – Toy Example on the BRD Decoder over GF(8).

2.5.2 Statistical and Requested Symbols Analysis

In the proposed BRD algorithm, the variable-to-check messages M_{V_j2C} are considered for obtaining the requested symbols. But the VN V_j has three possible messages that can be requested, the intrinsic message I_j obtained from the channel observation, the a posteriori information APP_j message, and the variable-to-check message M_{V_j2C} sent to a CN C_i. The variable-to-check messages M_{V_j2C} are considered for obtaining the requested symbols due to the best performance-complexity trade-off.

It has been observed that requesting the n_R most reliable symbols of the intrinsic message I_j degrades the performance of the BRD algorithm by about 0.1 dB compared to the BRD algorithm with M_{V_j2C} symbols being requested. This is justified since the intrinsic message is generated by observing the channel and is never updated during the decoding process. Hence, if the transmitted symbol is not included in the n_R requested symbols from the intrinsic message, it will never be requested throughout the decoding process.

On the other hand, requesting the n_R most reliable symbols of a posteriori APP_j message yields a similar performance to that of the requested variable-to-check message $M_{V_j2C_i}$. Nevertheless, the complexity of requesting the APP_j message is higher when compared to the M_{V_j2C} due to the additional sorting of the APP message (at each iteration and for each VN) required to extract the n_R reliable symbols. Additionally, the size of the message sent by the VN to the CN is also increased from n_{vc} GF values and $n_{vc}-1$ LLRs to $n_{vc}+n_R$ GFs and $n_{vc}-1$ LLRs since the n_R symbols of the APP_j are distinct from the symbols of the M_{V_j2C}. Therefore, they should be sent to the CN in addition to the message M_{V_j2C}.

Therefore, it is believed that the most suitable message to be requested in terms of complexity and performance is the variable-to-check message $M_{V_j2C_i}$.

Moreover, a statistical analysis has been carried out to determine the reasoning behind the performance preservation of the BRD algorithm. The statistical analysis indicates that the main criterion that affects the performance is not the size of the message but rather the probability that the encoded symbol x_j belongs effectively to the exchanged message M_{C2V_j}, i.e, $P(x_j \in M_{C2V_j})$. It is noticed that the requested candidates increase the probability significantly, thus leading to good decoding performance even with a small message size.

A Monte-Carlo estimation of $P(x_j \in \Gamma(M_{C2V_j}))$ is presented in Fig. 2.13 as a function of the message size for several decoding algorithms. The Monte-Carlo simulations are performed for a code rate $r = 5/6$ ($d_v = 2, d_c = 12$) over GF(64) NB-LDPC code of size $N = 144$ symbols (864 bits). The SNR (per bit) is set at 3 dB (beginning of the waterfall region) with a maximum of 30 decoding iterations. The probability $P(x_j \in \Gamma(M_{C2V_j}))$ is estimated in the whole decoding process as a function of the M_{C2V_j} message length. For the EMS algorithm, the message length is given by n_m. The algorithm parameters are $n_{op} = n_m + 5$ and the offset value is equal to 0.3. For the BRD algorithm, the message length is characterized by $n_{cv} = n_B + n_R$. The CN processing is based on the TEC-TEMS algorithm with the following parameter values: $\gamma_R = 1/8, \gamma_B = 2,\ O_D = 0.4,\ O_R = 0.2$ for the BRD decoder. Additionally, the TEC-TEMS is also simulated where only $n_m = 20$ candidates are analyzed from the full-set vector $q = 64$.

It is noticeable in Fig. 2.13 that the impact of including the requested symbols greatly enhances the probability that the transmitted symbol x_j belongs to the propagated message $\Gamma(M_{C2V_j})$, and hence becomes a possible candidate to be processed at both VN and CN. The probability $P(x_j \in \Gamma(M_{C2V_j}))$ is around 85% in EMS with $n_m = 20$. The prob-

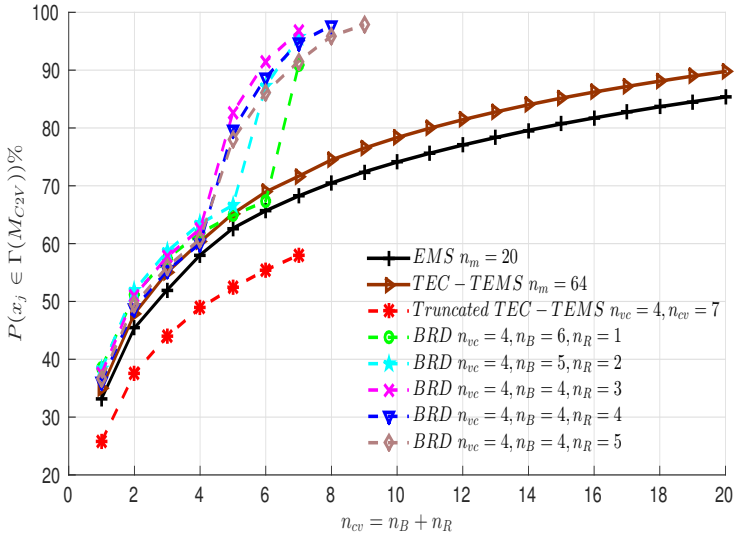


Figure 2.13 – Probability that the symbol $x_j \in \Gamma(M_{C2V_j})$

ability $P(x_j \in \Gamma(M_{C2V_j}))$ in TEC-TEMS has a similar percentage (87%) as the EMS decoder. When truncating the size of considered candidates to $n_{vc} = 4, n_B = 7, n_R = 0$, the probability drops down to 59%. Including one requested symbol ($n_B = 6, n_R = 1$) enhances the overall probability from 59% to 89% with the same message size ($n_{cv} = 7$). The proposed decoder with $n_B = 4, n_R = 3$ achieves a probability of 96%. Further increase in the size of the requested symbols leads to a minor enhancement in the inclusion probability. Therefore, the parameters $n_{vc} = 4$, $n_B = 4$, $n_R = 3$ are assumed to be a good configuration for the BRD algorithm at rate $r = 5/6$.

The BRD algorithm is an NB-LDPC algorithm that may be used with any CN processing algorithm, including the EMS, TEMS, and SYN. However, some processes must be altered so that the requested symbols can be well generated. Hence, the application of the BRD to CN processing algorithms is covered in the following sections.

2.5.3 Trellis BRD Decoder

The Trellis BRD Decoder is a BRD decoder that uses the TEC-TEMS algorithm proposed in [55] for the CN processing. As discussed in section 2.4.3, the TEMS is well-efficient for high code rates since the configuration sets are computed independent of the CN degree of connectivity (dependent on n_r instead of d_c) this leads to a reduction in the complexity of the configuration builder block in the CN unit (2.34). However, the main

drawback of the Trellis EMS is the size of the exchanged messages. The TEMS algorithm doesn't truncate the exchanged messages, it exchanges an LLR vector of size q. This leads to high complexity and routing for high-order fields ($q > 64$).

The BRD decoder is tested firstly with the Trellis EMS algorithm since the TEMS algorithm is well compatible with the BRD algorithm by construction. This is because the output of the CN $M^{P}_{C_i 2 V_j}$ is a vector of q LLR values where the index of the vector corresponds to the GF element of that LLR value. Therefore, the Γ compression block can easily extract the LLR value of the requested symbols to generate the message $M^{R+}_{C_i 2 V_j}$. In addition, the message $M^{B,P}_{C_i 2 V_j}$ can be generated by extracting the n_B most reliable elements of the message $M^{P}_{C_i 2 V_j}$.

At a given decoding iteration, the variable-to-check message sent by a VN V_j to the CN C_i $M_{V_j 2 C_i}$ is truncated by the Ω compression block from q down to n_{vc} where $n_{vc} \ll q$. Moreover, the Ω block sends the symbols of the most reliable n_R candidates of the truncated message $\Omega(M_{V_j 2 C_i})$ as the requested symbols to the CN where $n_R \leq n_{vc}$ (See Fig. 2.11.b). The message $\Omega(M_{V_j 2 C_i})$ is permuted by $h_{i,j}$ to obtain the message $\Omega(M^{P}_{V_j 2 C_i})$.

Before the CN receives the input message, the Ω^{-1} decompression block expands the message $\Omega(M^{P}_{V_j 2 C_i})$ to a q-elements vector $\Omega^{-1}(\Omega(M^{P}_{V_j 2 C_i}))$ by assigning the maximum LLR value to the $(q - n_{vc})$ GF elements, not in $\Omega(M^{P}_{V_j 2 C_i})$ such that they won't contribute in the configuration set building process. In addition, similar to the Ω block, the Ω^{-1} block sends to the Γ block the message $M^{P,R\oplus}_{j}$ which contains the n_R GF symbols of the most reliable candidates of the message $\Omega(M^{P}_{V_j 2 C_i})$.

The CN C_i processes the messages $\Omega^{-1}(\Omega(M^{P}_{V_j 2 C_i})) \forall j = 0, \cdots, d_c - 1$ as discussed in section 2.4.3. The CN outputs the message $M^{P}_{C_i 2 V_j}$ of size q. The message is sent to the Γ compression block for selecting the best and the requested candidates. The Γ compression selects the most reliable n_B candidates of the message $M^{P}_{C_i 2 V_j}$ and includes them in the message $M^{P,B}_{C_i 2 V_j}$. Additionally, the LLR values of the requested message $M^{P,R\oplus}_{j}$ is included in the message $M^{P,R+}_{j}$. The message $\Gamma(M^{P}_{C_i 2 V_j})$ is then obtained by concatenating the message $M^{P,R+}_{j}$ after $M^{P,B}_{C_i 2 V_j}$. After the inverse permutation, the inversely permuted message $\Gamma(M_{C_i 2 V_j})$ is processed by the block Γ^{-1} similar to the process discussed in section 2.5.1.

The simulation results illustrated in Fig. 2.14 are estimated for NB-LDPC codes of size (in symbols) $(8, 16)$, $(120, 144)$, and $(189, 210)$ on GF(64) [34]. The simulated decoder is based on layered scheduling and a maximum of 10 decoding iterations. In addition, the noise variance σ is expressed for a Binary Phase Shift Keying (BPSK) over an Additive White Gaussian Noise (AWGN) channel in terms of SNR per bit E_b/N_0 such that $\sigma =$

$\sqrt{2.r10^{\frac{-E_b/N_0}{10}}}$. As shown, TEMS-BRD has a Frame Error Rate (FER) performance similar to that of the EMS ($n_m = 20,\ n_{op} = 25$). The performance is simulated down to a FER of 10^{-6} with no significant performance degradation. The parameters used for simulating the BRD decoder for $r \geq 5/6$ are $n_{vc} = 4,\ n_B = 4,\ n_R = 3,\ \gamma_B = 2,\ \gamma_R = 0.125,\ O_D = 0.4,\ O_R = 0.2$ and a compensation factor (TEC-TEMS) $= 0.8$. For the code of size $N = 8, K = 16$ with $d_c = 4$ and $r = 1/2$, the BRD decoder showed similar performance to TEMS and EMS with a lower number of exchanged messages $n_{vc} = 8$, $n_B = 6$ and $n_R = 5$. The parameters used for simulation are $\gamma_B = 2,\ \gamma_R = 0.125,\ O_D = 0.4,\ O_R = 0.3$ and a compensation factor (TEC-TEMS) $= 0.8$.

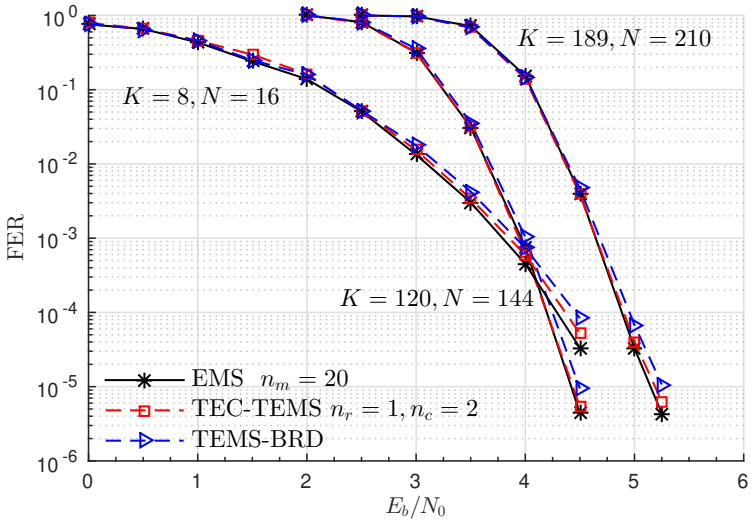


Figure 2.14 – TEMS-BRD Simulation Results for Code Rates $r = 1/2$, $r = 5/6$, $r = 9/10$.

On the other hand, the implementation of the BRD decoder with the TEMS algorithm is expected to have a higher complexity due to the sorting processes required at the variable and CNs. However, it could be a good solution for the routing congestion in fully parallel implementation. A better integration alternative is the EMS-based decoders (such as the FB decoder and the SYN decoder) where the sorting processes are already assigned within the nodes.

2.5.4 Syndrome-based BRD Algorithm

The Syndrome-based algorithm (SYN) is a CN processing algorithm [45] that uses predefined deviation paths for obtaining the syndromes. In the SYN-BRD algorithm, the

syndrome algorithm is integrated with the BRD algorithm such that the CN processing is a hybrid of both algorithms. The basic architecture of the SYN-BRD algorithm is illustrated in Fig. 2.15. The compressed variable-to-check messages $\Omega(M^P_{V_j 2C_i})$ are received by the CN and the syndromes are computed as in (2.28). After the syndromes are computed, they are sorted and inputted in parallel to d_c paths. In each path, a decorrelation unit DU decorrelates the syndromes similar to the decorrelation illustrated in Fig. 2.6. The decorrelated configurations are sent in parallel to the request-finder and the redundancy elimination blocks. The n_B most reliable configurations are inputted into the redundancy elimination unit (RE) for eliminating redundant symbols. In parallel, the request-finder block generates the LLR values for the requested symbols in $M_j^{P,R\oplus}$ sent by the VN (see Fig.2.11). Lastly, the messages $M^{P,B}_{C_i 2V_j}$ and $M_j^{P,R+}$ are concatenated to form the message $\Gamma(M^P_{C_i 2V_j})$.

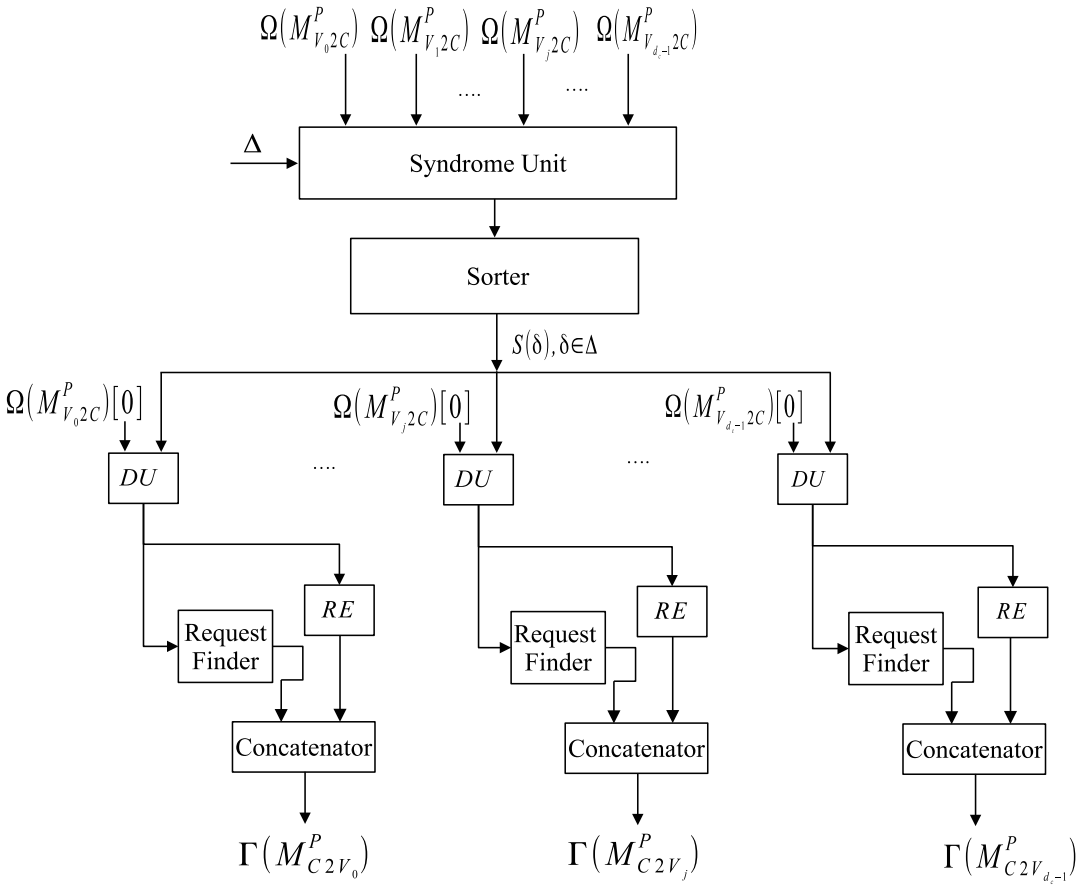


Figure 2.15 – SYN-BRD Architecture

The SYN-BRD decoder is simulated over the code $N = 144, K = 120$ and $N =$

$60, K = 20$ on GF(64) with a CN degree $d_c = 12$ and $d_c = 3$ respectively as shown in Fig. 2.16. The maximum number of iterations $iter_{max}$ is set to 10 over all simulations. The BRD parameters are set as follows, $\gamma_R = 1/8$, $\gamma_B = 2$, and $O_D = 0.3$. As for the SYN decoder, the deviation parameters are set as follows, $d_1 = n_{vc} - 1$, $d_2 = 2$ (for $N = 144$) and $d_2 = 3$ (for $N = 60$).

The advantage of using the SYN-BRD decoder is that it allows for reducing the communication load between the variable and the CNs. In addition, the size of the computed syndromes is further reduced, which allows for reducing the complexity of the main sorter.

For the code $N = 144$, the size of the variable-to-check message, of the best candidates and the requested candidates are $n_{vc} = 5$, $n_B = 4$, and $n_R = 3$ respectively. This yields a total of 313 computed syndromes (according to (2.30)), whereas in SYN decoder, for $n_m = 16$, the total syndromes computed is 456.

For the code $N = 60$, the size of the variable-to-check message, of the best candidates and the requested candidates are $n_{vc} = 14$, $n_B = 7$, and $n_R = 8$ respectively. This yields a total of 67 computed syndromes, whereas in the SYN decoder, for $n_m = 20$, the total syndromes computed is 70.

The simulation results for $N = 144$ and $N = 60$ are shown in Fig. 2.16. The FER of SYN-BRD is plotted against the FER of the EMS with $n_m = 20$ candidates and no performance degradation is noticed down to a FER of 10^{-5}.

Presorted SYN-BRD Algorithm

The presorted SYN-BRD decoder is an integration of the three algorithms, the presorting, the syndrome, and the BRD algorithms. The presorting algorithm is used to reduce the number of syndromes computed by the syndrome unit. In this decoder, the internal processing of the syndrome CN is similar to that previously described but with reduced deviation paths Δ_R.

The basic structure of the presorted SYN-BRD is illustrated in Fig. 2.17. The inputs to the CN $\Omega(M^P_{V_j 2C_i})$ are presorted to obtain a presorted version $\Omega'(M^P_{V_j 2C_i})$. The CN considers reduced deviation paths Δ_R obtained offline using a path elimination block.

In the presorted syndrome CN [47], the reduced deviation path Δ_R is generated by counting the occurrence of each deviation path $\delta \in \Delta$ that yields a syndrome $S(\delta)$ included in any of the d_c check-to-variable messages. The occurrence estimation is accumulated over several decoded codewords for accuracy.

In this proposed approach, the reduced deviation path Δ_R is generated by eliminating

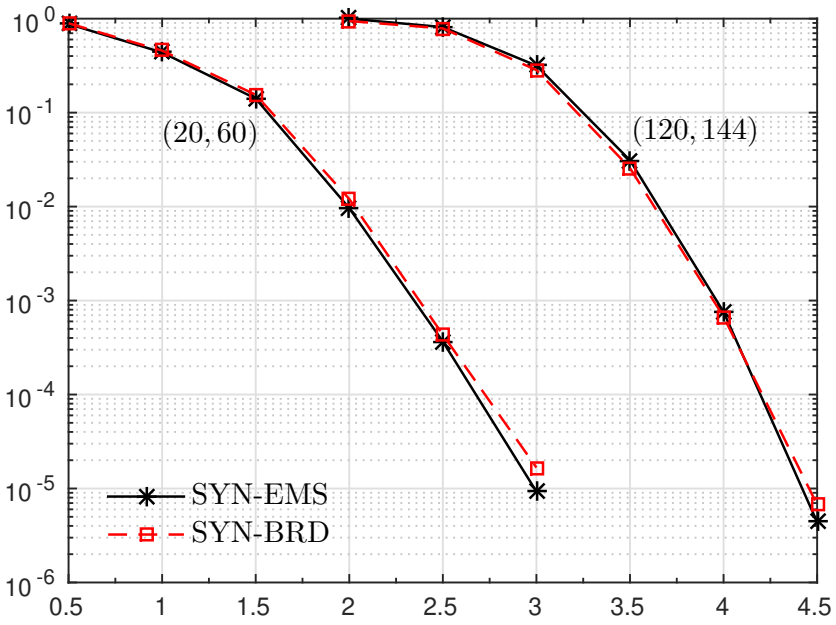


Figure 2.16 – Simulation of the SYN-BRD Decoder

the deviations δ using a cost function that gives a cost for each deviation path δ. To do so, assume the path elimination block depicted in Fig.2.18. The full deviation paths set Δ is inputted to the path elimination block, the block assigns for each deviation path a cost, denoted as $\xi(\delta)$. The cost $\xi(\delta)$ should reflect the reliability of the deviation path taking into consideration two factors, the first is the value of $\delta(i)$, the index of the symbol in the message $\Omega^{'}(M^{P}_{V_j 2C})$. Secondly, the value of j specifies the index of the VN. As $\delta(i)$ increases, the LLR value of the element increases since the input messages are sorted. In addition, with presorting, the symbol obtained from the message is less likely to contribute to an output element due to the higher LLR value as the value of j increases.

Therefore, the cost function $\xi(\delta)$ is found empirically as a linear function that depends on the element index and the message index, and computed as follows,

$$\xi(\delta) = \sum_{j=0}^{d_c-1} \delta(j) + (3 \cdot j),\ \delta(j) \neq 0,\ \forall\ \delta \in \Delta. \tag{2.43}$$

Thus, using the cost function $\xi(\delta)$ helps in computing the reduced Δ_R on the spot without any statistical estimation or Monte Carlo simulation.

Any deviation path δ with a cost $\xi(\delta)$ greater than ξ_{max} is eliminated and discarded

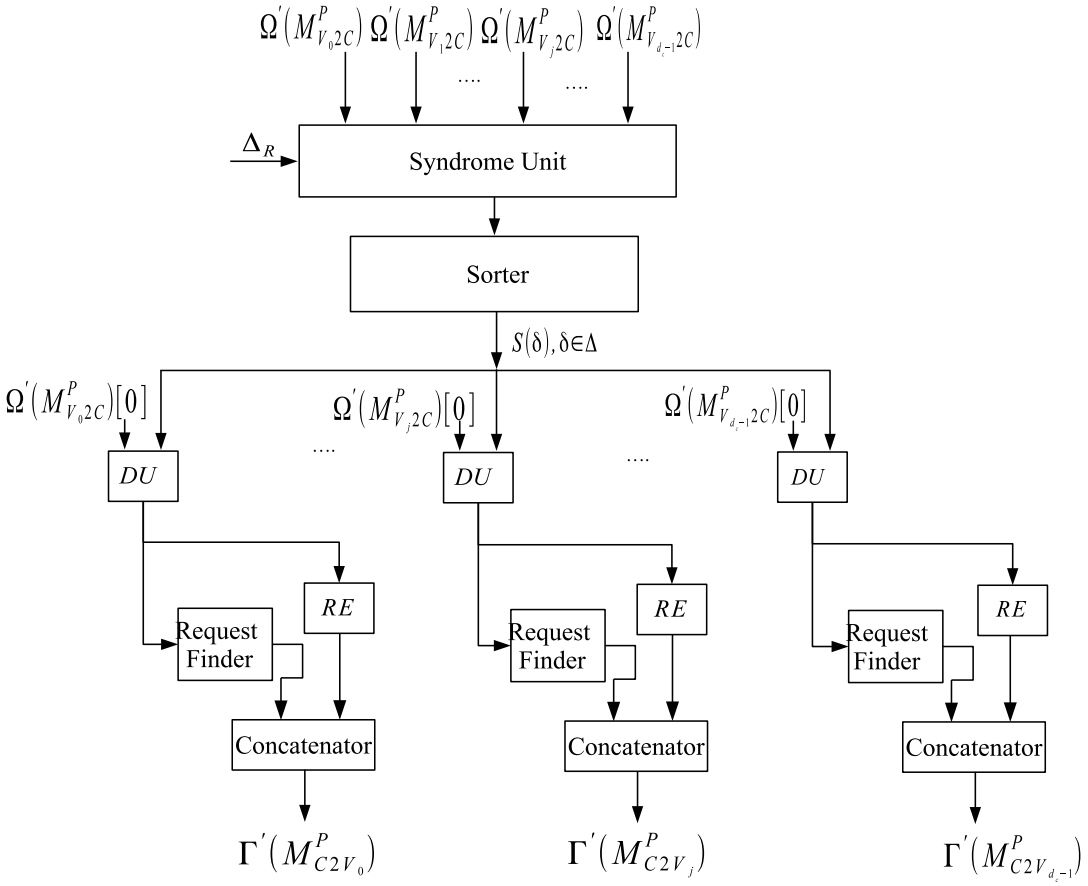


Figure 2.17 – Structure of the Presorted SYN-BRD Decoder

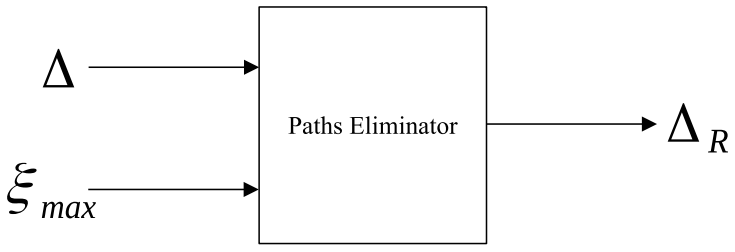


Figure 2.18 – Paths Elimination Block

such that all paths in the new set Δ_R have a cost less than ξ_{max}.

The simulation results presented in Fig. 2.19 illustrate the FER performance of the codes $N = 144$ and $N = 60$. The simulation is launched with BRD parameters similar to that in section 2.5.4 where $\gamma_R = 1/8$, $\gamma_B = 2$. As for the SYN parameters, they are also similar to that of the SYN-BRD such that the deviation parameters are set as follows, $d_1 = n_{vc} - 1$, $d_2 = 2$ (for $N = 144$) and $d_2 = 3$ (for $N = 60$). The maximum cost could be determined empirically, and chosen as the value that minimizes the FER. The maximum cost is set to $\xi_{max} = 17$ and $\xi_{max} = 52$ for the codes $N = 60$ and $N = 144$ respectively.

For the code $N = 144$, the size of the variable-to-check messages, of the best candidates and the requested candidates are $n_{vc} = 5$, $n_B = 4$, and $n_R = 3$ respectively. This yields a total of 220 computed syndromes (according to (2.30)), whereas in SYN decoder with $n_m = 16$, the total syndromes computed is 456 and in SYN-BRD, 313 syndromes. This results in a reduction of about 52% of the syndromes set compared to the conventional SYN and about 30% compared to SYN-BRD.

For the code $N = 60$, the size of the variable-to-check message, of the best candidates and the requested candidates are $n_{vc} = 14$, $n_B = 7$, and $n_R = 8$ respectively. This yields a total of 55 computed syndromes, whereas in the SYN decoder with $n_m = 20$, the total syndromes computed is 70.

As shown in Fig. 2.19, the performance of the presorted SYN-BRD is quite similar to that of the EMS with $n_m = 20$. The reduction of the deviation paths had no impact on decoding performance.

2.5.5 Forward-Backward BRD Decoder

In the FB algorithm, the CN is decomposed into multiple ECNs and layers as described in section 2.3.2. For the BRD algorithm to be compatible with the FB algorithm, some of the ECNs (the outer ECNs) are alerted with an additional process for obtaining the LLR values of the requested symbols. Therefore, the FB-BRD decoder has two types of ECNs, the conventional (inner) ECN and the BRD (outer) ECN. The location of the BRD-ECNs is known and static for any degree d_c. The right-most ECN in the forward layer, the left-most ECN in the backward layer, and the $d_c - 2$ ECNs of the merge layer are all BRD-ECNs. The FB-BRD decoder for $d_c = 5$ is depicted in Fig. 2.20(a), the white ECNs are the conventional ECNs, whereas the gray-shaded ECNs are the BRD-ECNs.

An edge between a VN and a CN in a FB-BRD decoder excludes the decompression process of Ω^{-1} (illustrated in Fig. 2.11) due to unnecessity since the forward-backward

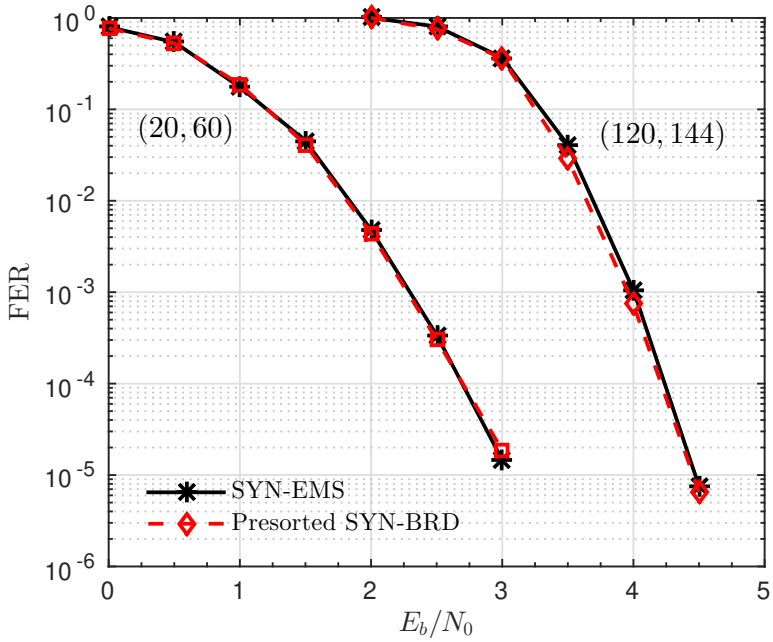
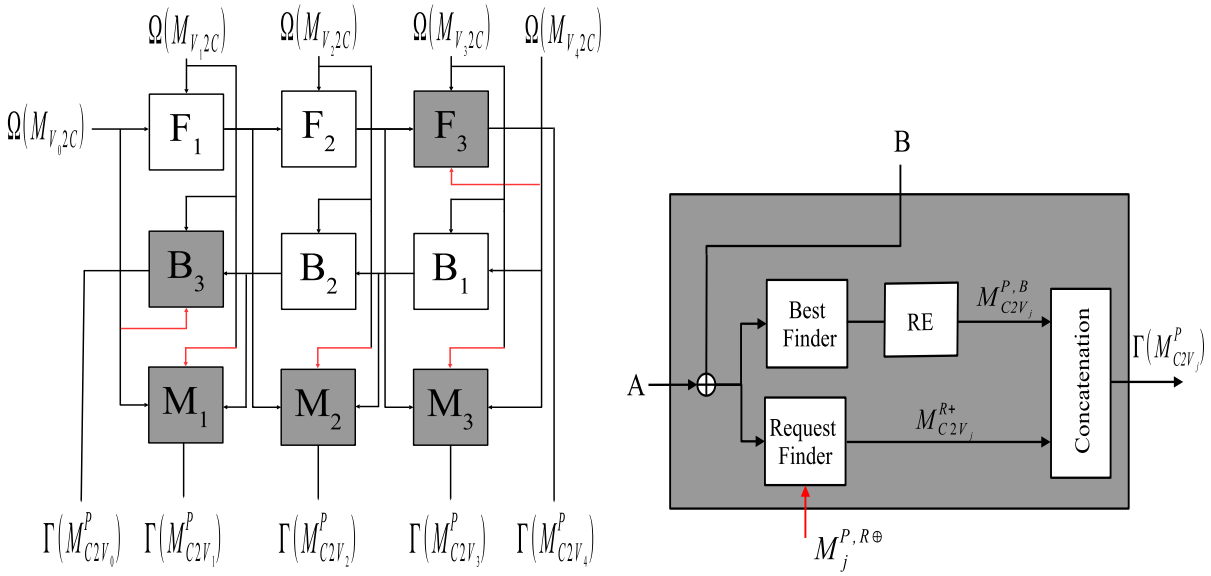


Figure 2.19 – Simulation Results of Presorted SYN-BRD Decoder



(a) Forward-Backward BRD Architecture for $d_c = 5$

(b) BRD-ECN Internal Structure

Figure 2.20 – Forward-Backward BRD Decoder

EMS (by construction) uses truncated messages. Additionally, since the CN is decomposed, the Γ compression process is considered one of the internal BRD-ECN processes.

A BRD-ECN consists of two internal blocks as illustrated in Fig. 2.20(b) named as the best finder and the request finder blocks. The best finder block is responsible for generating the n_B best candidates and the request finder block is responsible for generating the reliable LLR value of the n_R requested symbols. Therefore, a BRD-ECN has (similar to conventional ECN) two input messages A and B of size n_{m_A} and n_{m_B} respectively. In addition, the BRD-ECN has an input vector $M_j^{P,R\oplus}$ (red connection in Fig. 2.20(a)) of size n_R that contains the requested GF symbols where j is the index of the VN. The output of an BRD-ECN (see Fig. 2.20(b)) is the message $\Gamma(M_{C_i2V_j}^P)$ that consists of two subset messages $\Gamma(M_{C2V_j}^P) = M_{C2V_j}^{P,B}|M_{C2V_j}^{R+}$.

The message $M_{C_i2V_j}^{P,B}$ (a vector of n_B GF and LLR couples represented as $M_{C2V_j}^{B\oplus}$ and $M_{C2V_j}^{B+}$ respectively) is obtained by the best finder block that uses the S-bubble sorter with inputs A and B of size n_B (out of n_{m_A} and n_{m_B} respectively) candidates. The output of the bubble sorter is inputted to a redundancy elimination block to eliminate any redundant symbols. Thereafter, the message $M_{C_i2V_j}^{P,B}$ is well generated.

The request finder block generated the LLR vector $M_{C2V_j}^{R+}$ of the requested candidates in $M_j^{P,R\oplus}$. The algorithm 1 explains the main process of the request finder block. For a given requested symbol $M_j^{P,R\oplus}[k]$, the full region of the matrix T_Σ is explored. The matrix T_Σ is decomposed into n_{m_B} columns, in each column, the n_{m_A} elements are processed to check if the requested symbol corresponds to one of the elements if found, the LLR value of the symbol is updated in a temporary vector L of size $n_R \times n_{m_B}$. Since the input messages A and B consist of unique GF symbols, the requested symbol $M_j^{P,R\oplus}[k]$ might have a maximum of n_{m_B} LLR values in T_Σ because each column in T_Σ can have a maximum of one symbol that corresponds to the requested symbol $M_j^{P,R\oplus}[k]$ (if found). Therefore, having in a given row (or column) two elements that correspond to the same GF value is impossible.

Finally, the concatenation of $M_{C_i2V_j}^{R+}$ after $M_{C_i2V_j}^{B}$ results in obtaining the message $\Gamma(M_{C_i2V_j}^P)$ that is propagated to the decompression block Γ^{-1} before the VN update processing.

Integrating the BRD algorithm with the FB algorithm reduces the communication load of the decoder and hence, reduces the complexity of the sorting process at the CN and VN level. The algorithm reduces the communication load significantly at high code rates.

Algorithm 1: Process of Request Finder

Input: $T_{\Sigma},\ M_j^{P,R\oplus}$
Initialization: $L[n_R][n_{m_A}]$
for $k = 0$ **to** $n_R - 1$ **do**
  for $a = 0$ **to** $n_{m_A} - 1$ **do**
    for $b = 0$ **to** $n_{m_B} - 1$ **do**
      if $T_{\Sigma}^{\oplus}[a][b] = M_j^{P,R\oplus}[k]$ **then**
        $L[k][a] = T_{\Sigma}^{+}[a][b]$
        break;
      end
    end
  end
  $M_{C2V_j}^{P,R+}[k] = \min\{L[k]\}$
end
Output: $M_{C2V_j}^{P,R+}$

For very low code rates ($r = 1/3$), the size of the exchanged messages is slightly reduced since the ECN chain in all layers (forward, backward, and merge) depends on the inputs of the corresponding variable-to-check messages only. Hence, to obtain the LLR of the requested symbols, sufficient candidates should be considered. The size of the messages exchanged at low code rates are as follows, the variable to check message $M_{V_j2C_i}$ is $n_{vc} = 13$ couples of GF and LLR values, the check-to-variable $\Gamma(M_{C_i2V_j})$ is $n_B = 7$ best candidates (couples of GF and LLR values) along with $n_R = 8$ requested candidates (LLR values only).

For the code rate $r = 1/2$, the size of the exchanged messages is well-reduced. The variable-to-check message $M_{V_j2C_i}$ is reduced from $n_m = 20$ down to $n_{vc} = 8$ and the check-to-variable message is also reduced from $n_m = 20$ down to $n_B = 6$ and $n_R = 5$ candidates. Moreover, the size of the internal messages (between two ECNs) is also reduced from $n_m = 20$ to $n_{IN} = 10$.

The size of the exchanged messages is drastically reduced for high code rates ($r \geq 5/6$). The size of the variable-to-check message M_{V_j2C} is reduced from $n_m = 20$ (couples of GF and LLR values) for the conventional EMS down to $n_{vc} = 5$ (couples of GF and LLR values). Moreover, the size of the check-to-variable message is also reduced from $n_m = 20$ (couples of GF and LLR values) down to $n_B = 4$ (couples of GF and LLR values) and $n_R = 3$ (LLR values only). The size of the internal messages (between the ECN chain) is reduced from $n_m = 20$ down to $n_{IN} = 15$.

Table 2.3 – Size of Exchanged Messages per Edge on GF(64)

Scheme	Code Rate	Inputs		Outputs	
		$n_{vc}^{\oplus}$	n_{vc}^{+}	$n_{cv}^{\oplus}$	n_{cv}^{+}
FB-EMS[41]	any	20	19	20	19
FB-BRD	$r \geq 5/6$	4	3	4	6
	$r = 1/2$	8	7	6	10
	$r = 1/3$	13	12	7	14

Table 2.3 summarizes the size of the exchanged messages for both the FB-EMS and FB-BRD algorithms. The notations $n_{cv}^{\oplus}$ and n_{cv}^{+} denote the number of LLR elements and GF elements respectively, in the message $\Gamma(M_{C2V_j}^{P})$, and hence, could be computed as $n_{cv}^{\oplus} = n_B$ and $n_{cv}^{+} = n_B + n_R - 1$ (the first element in $M_{C2V_j}^{P,B+}$ is always zero, therefore, not propagated).

The FER performance proposed decoder is simulated over an AWGN channel with a BPSK modulation scheme for different code rates and lengths as illustrated in Fig. 2.21. The values of the saturation parameters of (2.42) are common for all codes and set as $\gamma_R = 1/8$, $\gamma_B = 2$, $O_R = 0.2$ and $O_D = 0.4$.

In addition, the importance of the FB-BRD decoder is seen at high field orders. To show that, the FB-BRD is simulated over GF(256) with $K = 60$ and $N = 75$ symbols (each symbol of 8 bits) to estimate the communication load required at both the CN as well as the VN. As shown in Fig. 2.22, the FB-BRD decoder requires only $n_{vc} = 18$, $n_B = 9$, $n_R = 9$ and $n_{IN} = 20$ to achieve similar performance as EMS with $n_m = 60$. The FB-EMS decoder with a similar communication load to that of the FB-BRD, i.e., $n_m = 20$, suffers from a performance degradation of around 0.2 dB.

Presorted FB-BRD Algorithm

The presorting algorithm discussed in section 2.3.4 helps in reducing the complexity of the FB-BRD algorithm by reducing the number of bubbles processed in each ECN and hence, the number of internal messages between the ECNs. The aim of using the presorting algorithm is to reduce the size of the computed candidates in T_{Σ} without affecting the decoding performance.

The internal processing of the ECNs with the presorting is identical to the conventional processing of the FB-BRD presented in section 2.5.5 with messages of different sizes. Thus, instead of having a fixed message size for all variable-to-check messages denoted as n_{vc},

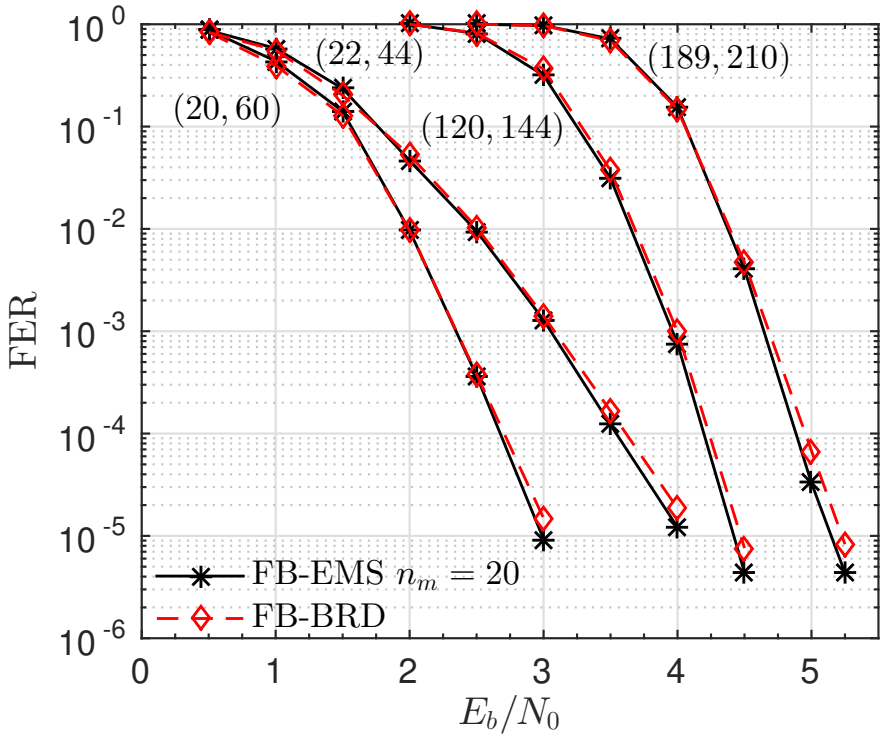


Figure 2.21 – Simulation Results for FB-BRD Decoder over GF(64)

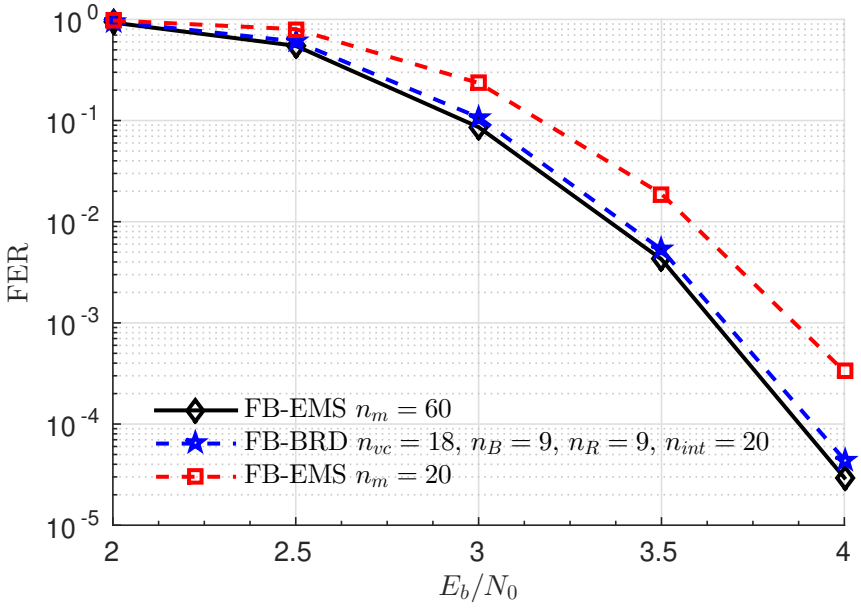


Figure 2.22 – Simulation Results for FB-BRD Decoder over GF(256)

each variable-to-check message $\Omega(M_{V_j 2C})$ has a size of $n_{v_j c}$. Moreover, instead of having a fixed message size for the internal messages (output messages of the non-output ECNs) denoted as n_{IN}, there are $2 \times (d_c - 3)$ message sizes denoted as $n_{IN_{L,j}}$ where L corresponds to the ECN layer (forward or backward only), and j corresponds to the index of the VN V_j. As for the check-to-variable messages $\Gamma(M^P_{C2V_j})$, all messages have a similar size denoted as n_{cv} that consist of n_B best candidates and n_R requested candidates.

The presorted FB-BRD algorithm is customized for the codes $N = 144, K = 120$ and $N = 60, K = 20$ with $d_c = 12$ and $d_c = 3$ respectively, and $d_v = 2$ based on the statistical analysis of each ECN.

For the code $N = 60, K = 20$, in each layer (forward, backward, and merge), only one ECN is required. Since $d_c = 3$, there are three variable-to-check messages $\Gamma(M^P_{V_j 2C})$ where $j = 0, \cdots, 2$. The presorting is applied to the messages before the CN processes them. The messages are presorted in the ascending order of the LLR values of the second element in each message. The first presorted message has size $n_{v_1 c} = 14$, the second presorted message has a size $n_{v_2 c} = 10$, and the third presorted message has a size $n_{v_3 c} = 6$. The size of the output messages is similar to the conventional FB-BRD, i.e., $n_B = 7$ and $n_R = 8$.

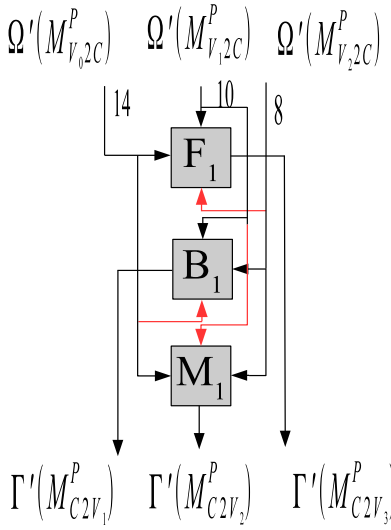


Figure 2.23 – Structure of the Presorted FB-BRD for $N = 60, K = 20$

For the code $N = 144, K = 120$, the twelve variable-to-check messages have a fixed size $n_{vc} = 4$ as shown in Fig. 2.24. The size of the internal messages varies depending on the input messages, as an example, the ECN F_1 outputs a message of size $n_{IN_{1,1}} = 8$

whereas the ECN F_{10} outputs a message of size $n_{IN_{1,10}} = 15$. As shown, the size of the internal messages in the backward layer could be further reduced compared to the forward layer. Moreover, in the merge layer, a partial part of the backward internal messages are processed. As an example, for the ECN M_1, the output message size of the ECN B_2 is $n_{IN_{2,2}} = 13$, but only the first 10 elements are considered in the ECN M_2 as illustrated in Fig. 2.24.

The size of the output messages $\Gamma(M^P_{C2V_j})$ is similar to the conventional FB-BRD, where $n_B = 5$, and $n_R = 3$.

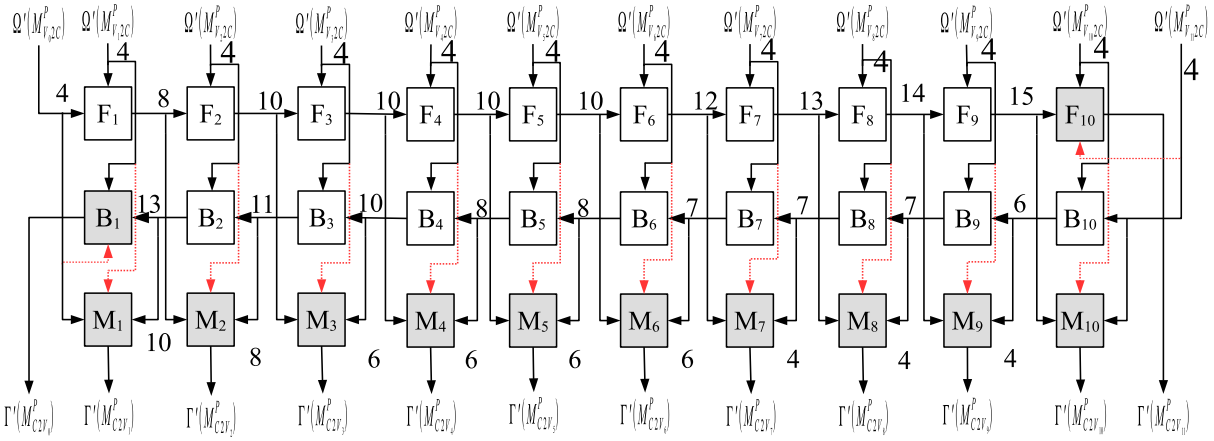


Figure 2.24 – Structure of the Presorted FB-BRD for $N = 144, K = 120$

The simulation results of the aforementioned structures of the codes $N = 60, K = 20$ and $N = 144, K = 120$ are illustrated in Fig. 2.25. The maximum number of iterations are all fixed to $iter_{max} = 10$. The simulation shows the FER down to 10^{-5}, as illustrated in the figure, the performance loss is around 0.06 dB for the code $N = 144, K = 120$, and no performance loss is noticed for the code $N = 60$ when both compared to the EMS algorithm with $n_m = 20$.

2.5.6 Implementation of FB-BRD Check Node

The implementation of the CNs using the FB-EMS and FB-BRD decoders (without presorting) discussed in sections 2.3.2 and section 2.5.5 respectively have been designed and implemented on FPGA for $d_c = 12$ on GF(64). The architecture of the FB-EMS decoder is straightforward to implement due to the same processing behavior of all ECNs. On the contrary, the architecture of the FB-BRD is more dynamic due to the different ECN processes.

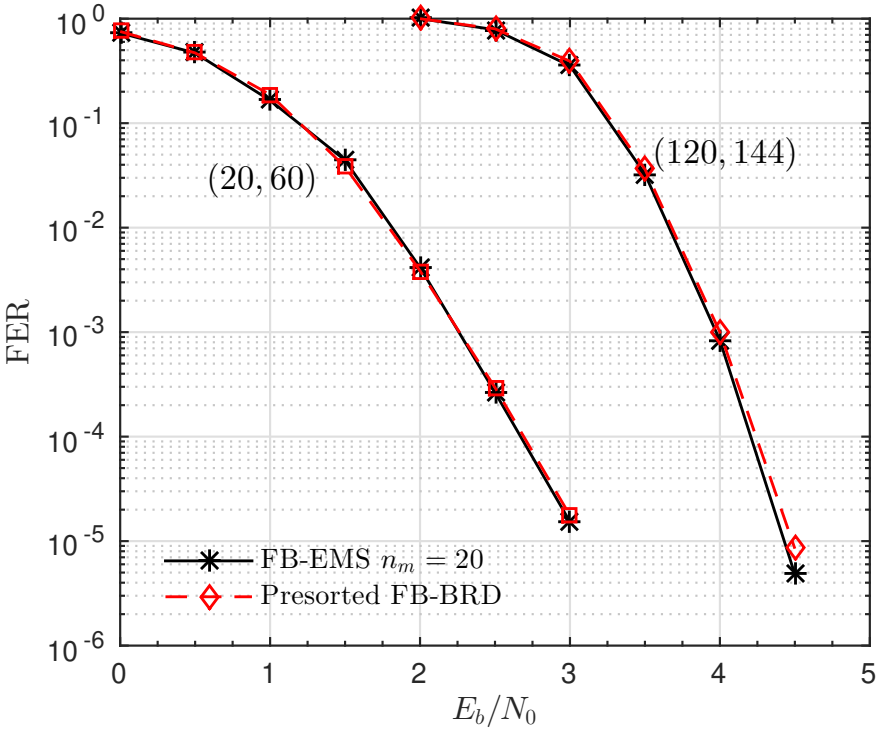
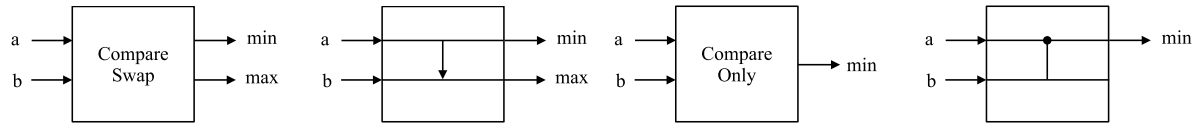


Figure 2.25 – Simulation Performance of Presorted FB-BRD for $r = 1/3$ and $r = 5/6$

The L-bubble and the S-bubble [43], [44] are serially implemented. However, they can be implemented in parallel using the bitonic-sorter [59]. In [60], [61], H. Harb et al. proposed and implemented a full parallel and pipelined NB-LDPC decoder for $N = 144$ symbols with $d_c = 12$ using parallel sorters and hybrid architecture mentioned in section 2.3.5.

A sorter consists of multiple comparators that are either compare-swap (CS) or compare-only (CO) comparators. The compare-swap comparator is represented by a vertical arrow line with the edges connected to the corresponding inputs and the arrow pointing to the maximum as shown in Fig. 2.26(a). The compare-only comparator is represented by a vertical line with the edges connected to the corresponding inputs and a rounded circle representing the minimum as shown in Fig. 2.26(b)



(a) Compare and Swap Comparator (b) Compare Only Comparator

Figure 2.26 – Schematic Representation of Comparators.

Figure 2.27 illustrates a generic architecture of a sixteen-to-sixteen bitonic sorter [59]. The sorter has an unordered input vector U of size 16 and outputs a vector V of size 16 too. The input vectors consist of two-tuple elements that correspond to the GF and LLR values. The comparison is performed on the corresponding LLR values and the GF elements are only propagated to the next block. The sorter consists of $\log_2(16) = 4$ layers. Layer L_0 consists of 8 compare swap blocks, each block obtains the min and the max of the corresponding two elements of the input vector A. For example, the first block yields the outputs $\min(U[0], U[1])$ and $\max(U[0], U[1])$, the second block yields the outputs $\max(U[2], U[3])$ and $\min(U[2], U[3])$, the third block yields the outputs $\min(U[4], U[5])$ and $\max(U[4], U[5])$, and so on. Moreover, the second layer, L_1 has 4 major blocks, each major block yields an ordered output of the four corresponding inputs. For example, the first major block of layer L_1 yields an output of 4 elements that correspond to the sorted sequence (ascending order) of the first four input elements, i.e., $U[0], U[1], U[2]$ and $U[3]$. Similarly, the second major block of L_1 generates the ordered sequence (descending ing order) of $U[4], U[5], U[6]$ and $U[7]$. Furthermore, layer L_2 consists of two major blocks. Each major block sorts the eight elements inputted from the previous layer. Hence, the first major block of L_2 generates an ordered sequence (ascending order) of

the input elements $U[0], U[1], U[2], U[3], U[4], U[5], U[6]$ and $U[7]$. Similarly, the second major block of L_2 generates the ordered sequence (descending order) of the input elements $U[8], U[9], U[10], U[11], U[12], U[13], U[14]$ and $U[15]$. At the last layer, L_3, the two ordered set of elements generated by the two major blocks of layer L_2 is processed to generate the ordered sequence of the input elements $\{U[i]\}_{i=0,\cdots,15}$.

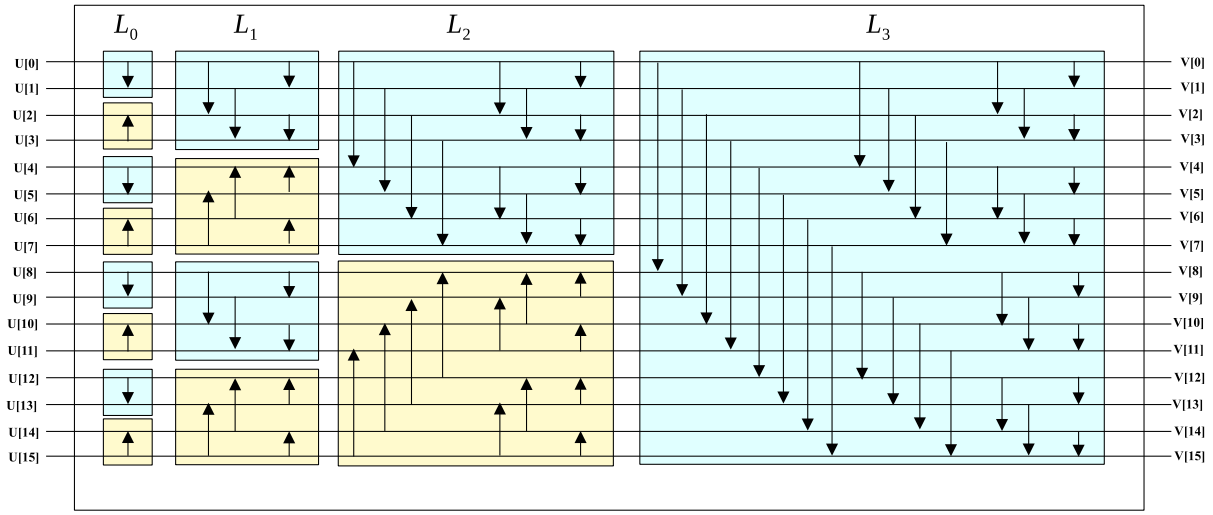


Figure 2.27 – Full Sixteen-to-Sixteen Bitonic Sorter

The sixteen-to-sixteen sorter can be represented as a group of sorters as shown in Fig. 2.28. Each layer L_i, consists of 2^{n-i-1} sorters each of size 2^{i+1} with n representing the size of the original sorter, i.e., $n = 16$. As shown, the 2^{n-i-1} inputs of any block consist of two ordered sequences the first 2^{n-i-2} inputs are ordered in the ascending order (generated from the corresponding cyan-colored block of the previous layer), and the second 2^{n-i-2} inputs are ordered in the descending order (generated from the corresponding cyan-colored block of the previous layer). Thus, if the vector U is assumed to consist of two ordered subsets A and B each of size 8, the first three layers ($L_0, L_1, and L_2$) of the full 16-to-16 sorter can be eliminated due to unnecessity. Thus, the top view of the sorter allows for optimizing the sorting process (by eliminating unnecessary comparisons) based on the ordered fashion of the inputs inputted to the ECNs.

The FB-EMS decoder has been synthesized for a message size of $n_m = 17$ candidates (not $n_m = 20$ as in the simulation results) to optimize the design of the parallel sorter. Therefore, each ECN has two inputs A and B, each of size 17 that are processed using a parallel implementation of the S-bubble architecture to obtain an output C of size $n_m = 17$ too.

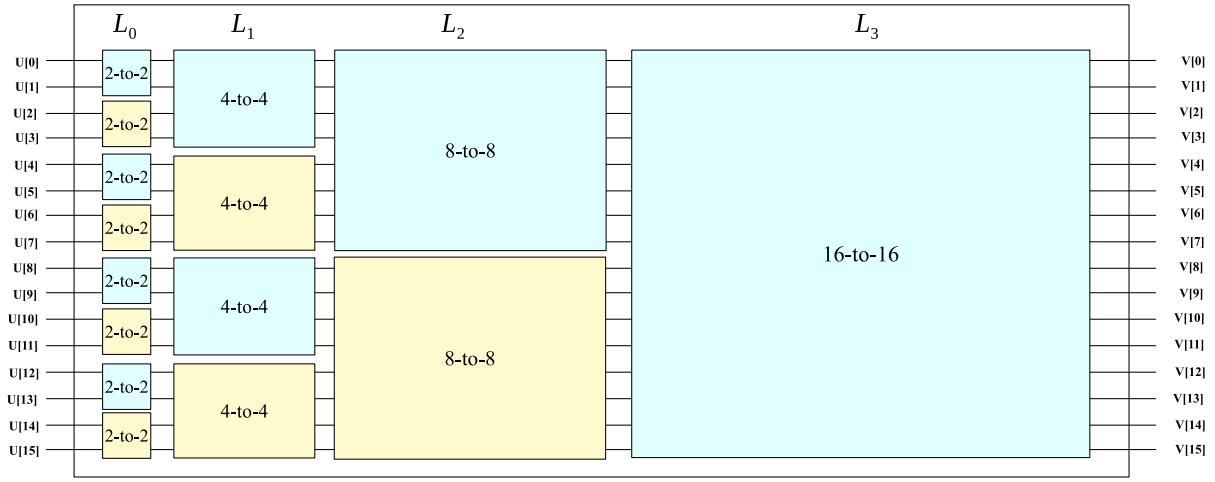


Figure 2.28 – Top-Level View of a Full Sixteen-to-Sixteen Sorter

To efficiently implement the parallel S-bubble sorter, the pattern exploration of the potential candidates computed in T_Σ is required to eliminate unnecessary comparisons. As previously mentioned, the S-bubble sorter computes only the first row and column, and the half-second row and column of T_Σ. Those can be expressed as 4 distinct regions R_0, R_1, R_2, and R_3 as

$$\begin{aligned} R_1 &= \{A[0] \boxplus B[j]\}_{j=0,\cdots,16}. \\ R_2 &= \{A[1] \boxplus B[j]\}_{j=1,\cdots,16}. \\ R_3 &= \{B[0] \boxplus A[j]\}_{j=1,\cdots,8}. \\ R_4 &= \{B[1] \boxplus A[j]\}_{j=2,\cdots,9}. \end{aligned} \tag{2.44}$$

As expressed in (2.44), the regions R_3 and R_4 are altered to include one and two additional elements compared to the conventional S-bubble proposed in [44] for implementation reasons.

Since A (similarly B) is a sorted vector, any element $A[i] < A[j]$ for $j > i$. Therefore, an element $R_1[i]$ (similarly $R_3[i]$) is always lower than the element $R_2[i]$ (similarly $R_4[i]$). This helps in reducing a bunch of comparisons that have known and fixed results. The sorter has a total of $17 + 16 + 8 + 8 = 49$ candidates from the regions R_0, R_1, R_2, and R_3 respectively, and should output 17 unique candidates. Hence, the sorter is a 48-to-16 sorter.

The proposed S-bubble architecture is illustrated in Fig. 2.29. As shown, the first element of R_1, $R_1[0]$, is directly outputted as the first sorted element (not included in the sorting process) since it is the global minimum (addition of $A[0]$ and $B[0]$). The 48-to-16

sorter is implemented in two stages, the first stage consists of two sorters, a 32-to-16 sorter, and a 16-to-16 sorter. The 32-to-16 sorter generates the 16 most reliable elements of R_1 and R_3. Since R_1 and R_3 are sorted vectors the 32-to-16 sorter can be simplified as illustrated in Fig. 2.30. Similarly, since R_2 and R_4 include sorted elements, the 16-to-16 sorter illustrated in Fig. 2.27 can be simplified as illustrated in Fig. 2.31. The second stage consists of a 32-to-16 sorter that generates the 16 minimum elements of the two vectors previously generated. Similar to the 32-to-16 sorter presented in Fig. 2.30, the two 16-element outputs of the 32-to-16 and the 16-to-16 sorters are inputted to a 32-to-16 sorter in the opposite order (one in ascending order and another in descending order). Thus, the 16 most reliable elements are generated in addition to the global minimum $R_1[0]$, and therefore, a total of 17 elements are generated using the parallel S-bubble sorter. The compare-swap comparators in the sorters are modified such that the redundancy elimination process is included in a parallel fashion. This can be done by checking the equality of the two inputs at each comparator, if the inputs are equal, the output of the maximum is set to saturation (maximum LLR). The output size of the last sorter can be increased to generate $n_m + 2$ candidates to ensure that n_m unique candidates are generated. In this implementation, the main focus is on the complexity of the FB-BRD, therefore, assuming a best-case scenario for the FB-EMS CN won't transform the analysis outcomes.

For ease of clarification, the FB-BRD architecture is illustrated in Fig. 2.32 for $d_c = 12$. Therefore, each layer consists of 10 ECNs. The ECNs F_{10} and B_{10} are BRD-ECN in addition to all ECNs of the merge layer.

The implementation of the FB-BRD requires three types of ECNs. The first ECN type is the ECNs at F_1 and B_1 where both have an input size of $n_m = 4$ and generate an output of size $n_{IN} = 15$. Therefore, the ECNs are not based on the S-bubble but a regular 16-to-16 sorter. The ECN F_1 (similarly B_1) with two inputs A and B of size $n_m = 4$ computes four regions of T_Σ denoted R_0, R_1, R_2 and R_3 and expressed as

$$\begin{aligned} R_0 &= \{A[0] \boxplus B[j]\}_{j=0,\cdots,3}. \\ R_1 &= \{A[1] \boxplus B[j]\}_{j=0,\cdots,3}. \\ R_2 &= \{A[2] \boxplus B[j]\}_{j=0,\cdots,3}. \\ R_3 &= \{A[3] \boxplus B[j]\}_{j=0,\cdots,3}. \end{aligned} \tag{2.45}$$

The sorter used at ECNs F_1 and B_1 is as depicted in Fig.2.33. The blue double-

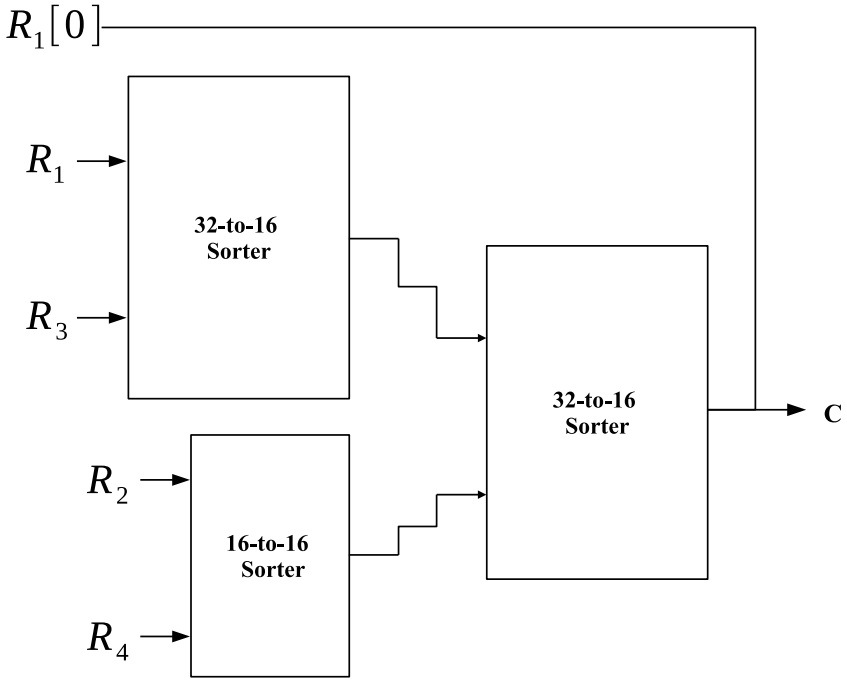


Figure 2.29 – Parallel S-Bubble Sorter for ECN with Input Size $n_m = 17$.

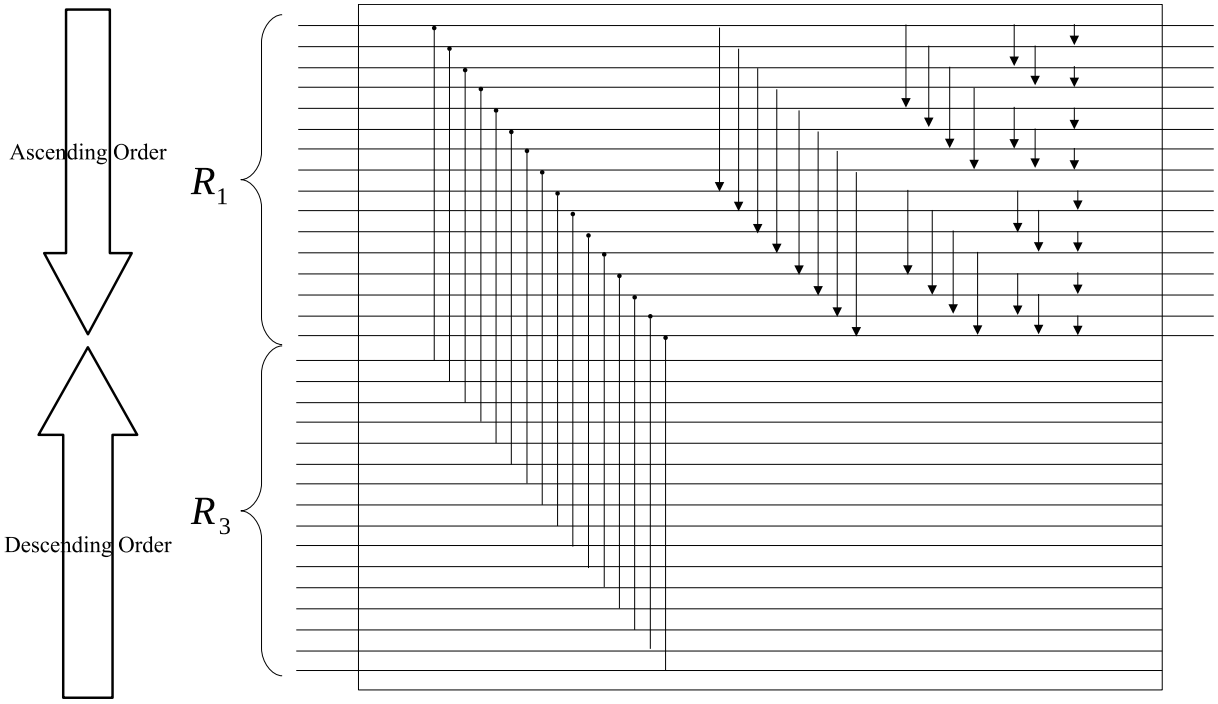


Figure 2.30 – Optimized 32-to-16 Sorter for S-bubble.

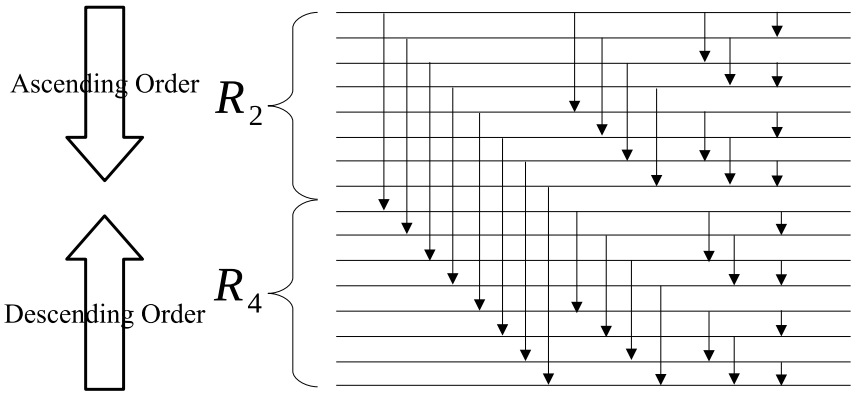


Figure 2.31 – Optimized 16-to-16 Sorter for S-bubble.

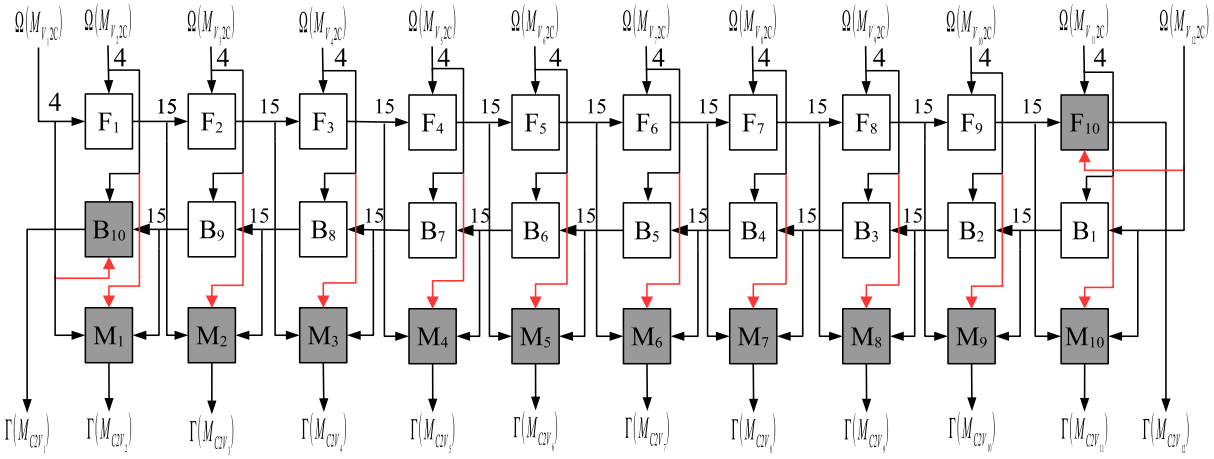


Figure 2.32 – FB-BRD Architecture for $d_c = 12$.

arrow lines represent a switch rather than a comparator and are replaced because $R_0[0]$ is the global minimum and therefore, no comparison is required. In addition, the two grey comparators are eliminated since the comparisons $R_3[2] > R_2[1]$ and $R_3[3] > R_2[0]$ are always satisfied and needless to be checked by dedicated comparators.

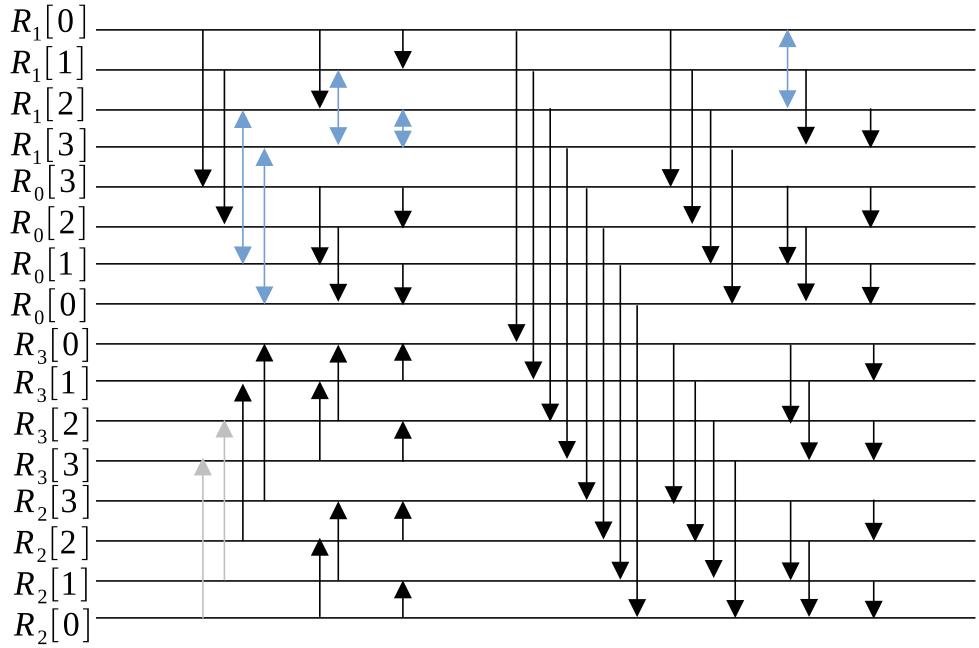


Figure 2.33 – Sixteen-to-Sixteen Sorter for ECNs F_1 and B_1.

The second ECN type includes the non-edge ECNs of the forward and the backward layers ($F_2, \cdots, F_9$ and $B_2, \cdots, B_{10}$). Assume an ECN of the aforementioned ECNs with inputs A and B of size n_{m_A} and n_{m_B} and an output of size n_{IN}. The input sizes are asymmetrical, i.e., $n_{m_A} \neq n_{m_B}$. Let A be the input vector with fewer elements ($n_{m_A} = 4$) and B be the input vector of higher size ($n_{m_B} = 15$). The ECN computes the $n_{IN} = 15$ most reliable elements from the following four regions R_0, R_1, R_2, and R_3 expressed in (2.46) that sums up to a total of 24 elements. Thus, a 24-to-17 sorter is required to generate the output vector C. The 24-to-17 sorter demonstrated in Fig. 2.34 consists of three stages, the first stage includes two sorters, a 16-to-8 sorter (to generate the most reliable elements of the first half of R_2 and R_3), and a 4-to-4 (to generate an ordered sequence of R_1 and two elements of R_2) sorter. The second stage includes an 8-to-8 sorter (to generate the ordered sequence of R_1 and the last six elements of R_2), and the last stage includes a 16-to-16 sorter that sorts out the eight minimum elements of the first half of R_2 and R_3, and the eight minimum values of R_1 and the second half of R_2. All sorters

include the last layer of their corresponding full sorter since the inputs of any n-sorter consist of two ordered vectors of size $n/2$. In addition, some comparators are eliminated based on the relation between the compared elements.

$$\begin{aligned} R_0 &= A[0] \boxplus B[0] \\ R_1 &= \{A[j] \boxplus B[0]\}_{j=1,\cdots,3}. \\ R_2 &= \{A[0] \boxplus B[j]\}_{j=1,\cdots,14}. \\ R_3 &= \{A[1] \boxplus B[j]\}_{j=1,\cdots,7}. \end{aligned} \tag{2.46}$$

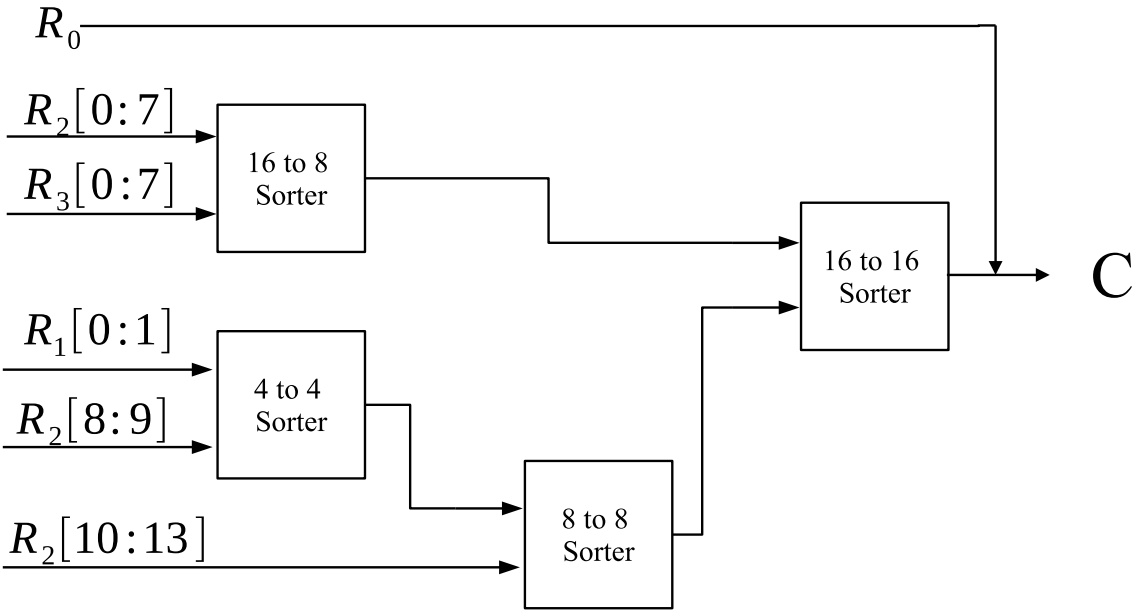


Figure 2.34 – Twenty Four-to-Sixteen S-bubble Sorter Architecture

The third type of ECN is the BRD-ECN. The BRD-ECN includes different processing compared to the conventional ECN based on the input size, the size of the best candidates, and the size of the requested candidates. As illustrated in Fig. 2.20(b), the BRD-ECN consists of two blocks: the best finder block, and the request finder block.

The best finder generates the n_B candidates using only the n_B most reliable elements of the input vectors A and B regardless of the input sizes n_{m_A} and n_{m_B}. The computed

candidates are expressed by three regions R_0, R_1, and R_2 as follows.

$$\begin{aligned} R_0 &= \{A[0] \boxplus B[j]\}_{j=0,\cdots,3}. \\ R_1 &= \{A[j] \boxplus B[0]\}_{j=1,\cdots,3}. \\ R_2 &= A[1] \boxplus B[1]. \end{aligned} \tag{2.47}$$

The sorter used to obtain $n_B = 4$ most reliable elements of the regions in (2.47) is a simple sorter as illustrated in Fig.2.35. $R_0[0]$ is the global minimum and hence, propagated directly to the output. The second minimum is either $R_0[1]$ or $R_1[0]$, or denoted as min of CS1. The third minimum is either max of $CS1$ or min of $CO1$ which correspond to either $R_0[2]$ or $R_1[1]$. If $R_0[2]$ is not min of $CO1$, then $R_0[3]$ is never a potential candidate for the fourth minimum. Therefore, a multiplexer is added to select between $R_0[3]$ and $R_1[2]$ based on if $R_0[2] > R_1[1]$. Furthermore, the comparator only $CO2$ selects the minimum among the selected element of the multiplexer and $R_2[0]$. Then, the min of $CO2$ and the max of $CS2$ are compared to generate a minimum that corresponds to the fourth reliable element. Hence, the n_B most reliable candidates are efficiently generated.

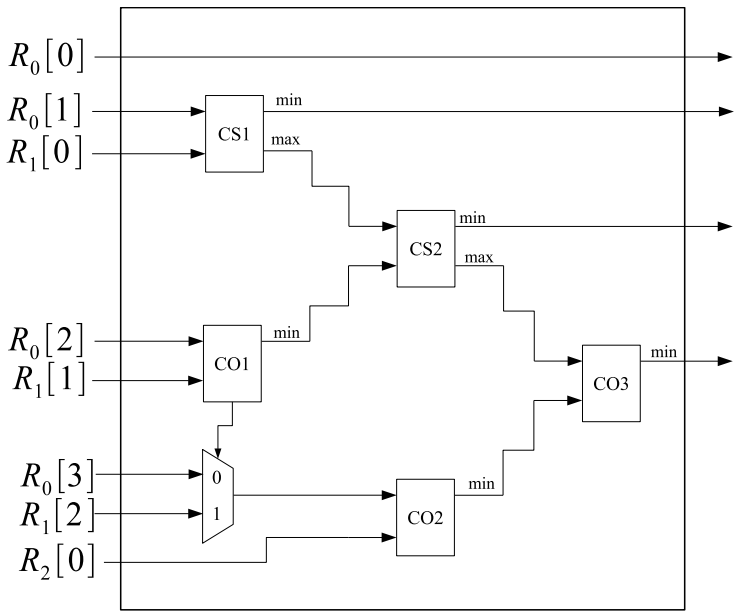


Figure 2.35 – Architecture of Best Finder Sorter

Furthermore, the request finder block generates the n_R LLR values of the requested message $M_j^{P,R\oplus}$ in parallel using n_R LLR Finder blocks as illustrated in Fig. 2.36.

Each LLR Finder block generates an LLR value for the requested symbol $M_j^{P,R\oplus}[k]$ sent

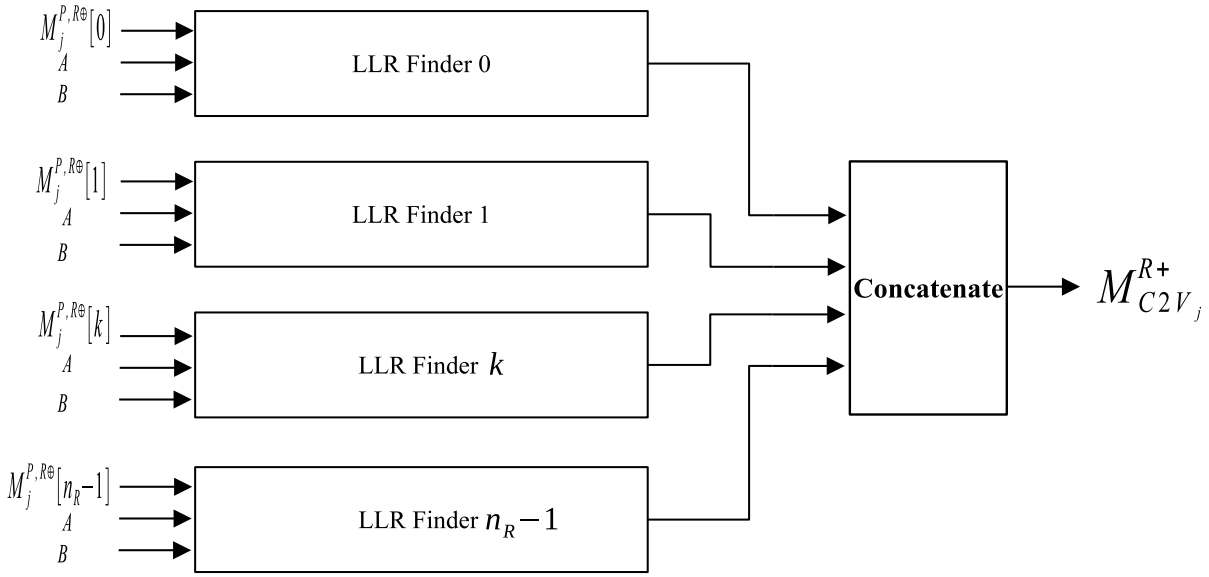


Figure 2.36 – Internal Structure of Request Finder Block.

by the VN V_j. The LLR Finder block considers the two inputs of the BRD-ECN, A and B of size n_{m_A} and n_{m_B} respectively to generate the n_R LLR values of the requested symbols. Internally, the LLR Finder block consists of n_{m_A} Column Finder blocks as depicted in Fig.2.37.

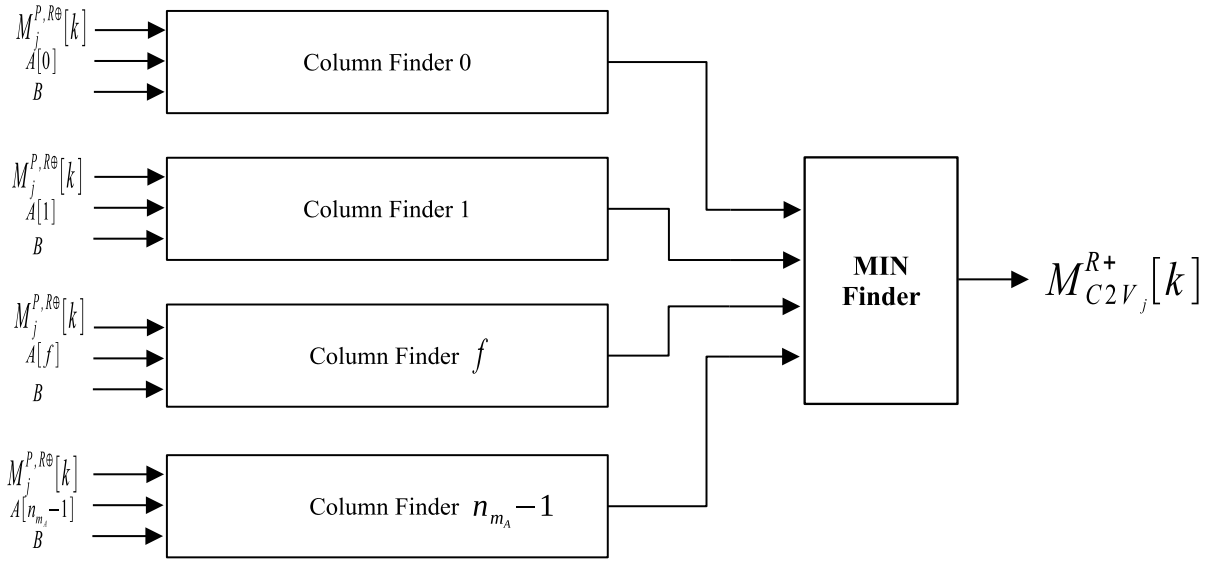


Figure 2.37 – Internal Structure of LLR Finder Block.

Each Column Finder block generates an LLR value for the requested symbol $M_j^{P,R\oplus}[k]$.

Assume a block Column Finder f, the block generates an LLR value for the requested symbol $M_j^{P,R\oplus}[k]$ using only one column which is obtained as $A[f] \boxplus B$. Therefore, the block should select one LLR value from the column's n_{m_B} elements. As a result, the block checks if $M_j^{P,R\oplus}[k] = A^{\oplus}[f] \oplus B^{\oplus}[b]$ for $b = 0, \cdots, n_{m_B}$. If $M_j^{P,R\oplus}[k] = A^{\oplus}[f] \oplus B^{\oplus}[b]$ is satisfied, the output LLR is $A^{+}[f] + B^{+}[b]$. If neither of the column elements represents the requested symbols the output LLR corresponds to the maximum column LLR, i.e., $A^{+}[f] + B^{+}[n_{m_B} - 1]$. The output of the n_{m_A} column blocks is inputted to a minimum finder to obtain the most reliable LLR value among all generated values. This processing can be performed using the proposed architecture depicted in Fig. 2.38.

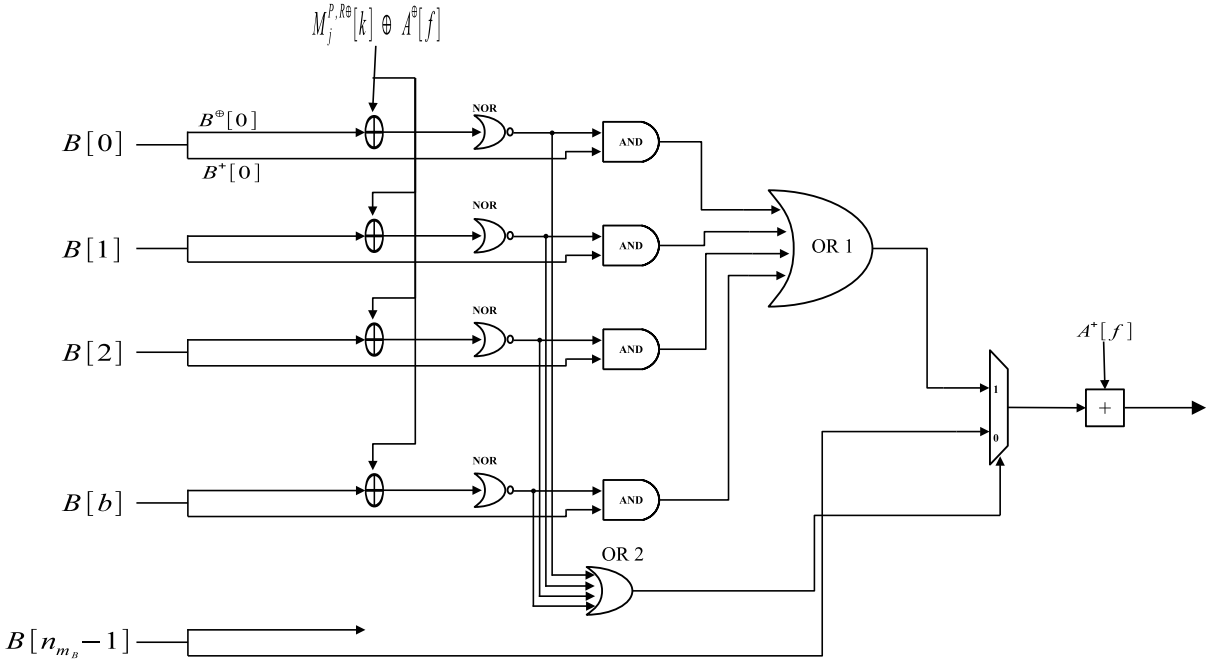


Figure 2.38 – Internal Structure of Column Finder Block.

In the proposed architecture of the Column Finder block, the addition processes are optimized such that only one LLR adder is required instead of n_{m_B} adders. As discussed, the block should check if $M_j^{P,R\oplus}[k] = A^{\oplus}[f] \oplus B^{\oplus}[b]$ for $b = 0, \cdots, n_{m_B} - 1$ to generate an output element. The equality check is a p-bit XOR that outputs a $p-$bit vector. If the vector is all 0, then the two inputs are equal. Otherwise, unequal inputs. Instead of adding $A^{\oplus}[f] \oplus B^{\oplus}[b]$ (which requires n_{m_B} GF adders) to check the equality, the requested symbol is added to the element $A_j^{\oplus}$, i.e., $M_j^{P,R\oplus}[k] \oplus A^{\oplus}[f]$ and the equalities are reformulated as $M_j^{P,R\oplus}[k] \oplus A^{\oplus}[f] = B^{\oplus}[b]$ for $b = 0, \cdots, n_{m_B} - 1$. Hence, only one GF addition is required to check for equalities. In addition, it is known that if $M_j^{P,R\oplus}[k] \oplus A^{\oplus}[f] = B^{\oplus}[b]$, the

output LLR should be $A^{+}[f]+B^{+}[b]$. Thus, a NOR gate is added after the equality checker such that the output of the NOR gate is 1 if and only if the equality check is satisfied. Otherwise, the NOR gate output is 0. The output of the NOR gate is inputted along with the LLR value $B[b]$ to an AND gate. The AND gate allows the propagation of the LLR value $B[b]$ if and only if the output of the NOR gate is 1 ($M_j^{P,R\oplus}[k] \oplus A^{\oplus}[f] = B^{\oplus}[b]$ is satisfied). The output of the $n_{m_B}-1$ AND gates are propagated to an OR gate (depicted as $OR1$ in the figure). As shown, the last element $B[n_{m_B}-1]$ is excluded from the equality check since if neither of the first $n_{m_B}-1$ equality checks are satisfied, the LLR value $B^{+}[n_{m_B}-1]$ is propagated to the output. Hence, a 2-to-1 multiplexer is used to select between the output of the OR gate and the LLR value $B^{+}[n_{m_B}-1]$. The selection is based on the result propagated from the OR gate ($OR1$) that considers the NOR gate output. If one of the equality checks $M_j^{P,R\oplus}[k] \oplus A^{\oplus}[f] = B^{\oplus}[b]$ for $b = 0, \cdots, n_{m_B}-2$ is satisfied, one NOR gate will output 1 and the remaining NOR gates will have an output 0 (since B is a vector of distinct GF symbols). Therefore, the output of the OR gate ($OR2$) is always 1 if and only if one equality is satisfied, therefore, the multiplexer considered the output of the OR gate $OR1$ (that corresponds to the LLR value of $B_{[}b]$). Thus, the output of the multiplexer is either one of the $n_{m_B}-1$ LLRs that satisfied the corresponding condition $M_j^{P,R\oplus}[k] \oplus A^{\oplus}[f] = B^{\oplus}[b]$ or the LLR of the last element in B, $B^{+}[n_{m_B}-1]$. In any case, the output of the multiplexer is propagated to an LLR adder that adds the LLR element $A^{+}[f]$ to the output of the multiplexer. Consequently, if a symbol $B^{\oplus}[b]$ equals to the resultant symbol $A^{\oplus}[f] \oplus M_j^{P,R\oplus}[k]$ the output of the column finder block is $A^{+}[j] + B^{+}[b]$. Otherwise, the output of the column finder block is $A^{+}[j] + B^{+}[n_{m_B}-1]$. This allows replacing the n_{m_B} LLR adders with only one LLR adder.

The hardware complexity of the CN is studied using the Quartus Prime synthesis tool where different parts of the CN (i.e. ECNs) are synthesized in a fully parallel implementation for a code rate $r = 5/6$ with $d_c = 12$ on Cyclone IV FPGA (summarized in Table 2.4). The synthesis results show that the CN of the FB-EMS requires 109860 logic elements and 89940 registers. Similarly, for the BRD-based FB architecture with parameters $n_{vc} = 4, n_B = 4, n_R = 3$, the CN requires 94782 logic elements and 37308 registers. This shows that the FB-BRD algorithm reduces the memory allocations by around 58% compared to the FB-EMS, and reduces the computational complexity by around 15%.

Moreover, the study includes the comparison of the synthesis results of the CN units only, further complexity reduction is expected when considering the VN unit since the size of the sorter at the VN units is reduced from $n_m = 17$ candidates down to $n_B = 4$

sorted out of $q = 64$ total candidates.

Table 2.4 – Synthesis Results for $d_c = 12$ on GF(64)

Scheme	Logic Elements	Registers
FB-EMS ($n_m = 16$) [41]	109860	89940
FB-BRD ($n_{vc} = 4, n_B = 4, n_R = 3$)	94782	37308

2.6 Conclusion

This chapter presented the Non-Binary Low-Density Parity Check (NB-LDPC) codes and decoders. The chapter begins with section 2.1 which is an introductory section about NB-LDPC codes. In this section, the NB-LDPC code structure and construction are explained.

The following section 2.2 presents the main iterative decoding algorithms for the NB-LDPC decoders. In this section, the optimal algorithms such as the belief propagation are presented in section 2.2.1, and the logarithmic belief propagation in section 2.2.2. In addition, the sub-optimal algorithms such as the min-sum and the min-max are presented in section 2.2.3 and section 2.2.4 respectively.

The min-sum efficiently reduces the number of required look-up tables in the logarithmic belief propagation but still suffers from intensive computations. The min-max simplifies the complexity more than the min-sum, but it suffers from a higher degradation in performance. Nevertheless, there are two main algorithms proposed for further simplification by extending the sub-optimal approximations. The first is called the extended min-sum and the second is the trellis extended min-sum.

The extended min-sum algorithm is presented in section 2.3. The first section 2.3.1 explains the general processing performed in an extended min-sum algorithm such as the truncated messages of size n_m and the performed sorting of the messages. Then some of the efficient approaches used to implement the extended min-sum CN such as the forward-backward approach in section 2.3.2. The forward-backward implementation decomposes the CN into three layers with each layer having $d_c - 2$ ECNs. The decomposition of the CNs allows for data reuse at the third layer, and hence, allows for a reduced complexity CN. The syndrome-based decoder is introduced in section 2.3.3 and relies on the concept of deviation paths that include a reduced set of computed syndromes based on the number of allowed deviations. Furthermore, the presorted CN is presented in section 2.3.4.

The presorting of the inputs of the VN reduces the number of computations in both the forward-backward and the syndrome-based approaches. This is because the presorting allows to elimination of a significant amount of useless elements from the reliable VNs, and hence, prunes the unused bubbles or deviation paths. Additionally, the three aforementioned approaches are used to design a hybrid architecture as presented in section 2.3.5.

The trellis extended min-sum algorithm is presented in section 2.4. The trellis extended min-sum is proposed as an alternative approach to process the CNs at high coding rates due to the high latency experienced in the EMS-based CNs at high d_c. The CN processing of the trellis extended min-sum is explained in section 2.4.1. Then, two sub-optimal approaches added to the traditional trellis EMS are presented. In section 2.4.2, the one-minimum-only approach approximates the second minimum extracted from each symbol of the delta matrix such that the two-minima finder is simplified into a radix-2 minimum finder. Further, the two-extra column presented in section 2.4.3 reduces the size of the configuration sets by considering the second most reliable configuration to update the reliability of the deviated elements.

Additionally, the Best, the Requested, and the Default (BRD) algorithm which is proposed during this PhD is tackled in section 2.5. In the BRD algorithm, the VN requests the reliability of particular symbols to be sent back by the CN. As a result, a check-to-variable message is produced by three subsets: the best candidates, which have the highest reliability, the requested candidates, and the default candidates which have the lowest reliability of the two groupings. This allows for reducing the size of the propagated messages between the variable and the CNs as well as reducing the sorting processes within the decoding process. The BRD algorithm is explained in section 2.5.1 including the messages compression and decompression. Then, the statistical justification behind the BRD concept is presented in section 2.5.2. Moreover, the BRD has been combined with multiple CN processing algorithms such as the trellis extended min-sum CN as presented in section 2.5.3, the syndrome-based CN in section 2.5.4 (with and without presorting), and the forward-backward CN in section 2.5.5 (with and without presorting). The simulation results of the aforementioned decoders confirm a negligible performance loss compared to the corresponding extended min-sum decoders. In addition, the implementation of a parallel forward-backward BRD and forward-backward extended min-sum is presented in section 2.5.6. The synthesis results show that the forward-backward BRD requires 40% of the memory allocation and 85% of the computation complexity utilized by the forward-

backward extended min-sum.

Chapter 3

Non-Binary Polar Codes and Decoders

Contents

3.1 Introduction to Non-binary Polar Codes 94
3.1.1 Polar Transformation and Channel Polarization 94
3.1.2 Polar Coding 95
3.1.3 Overview on Non-Binary Polar Codes 96
3.2 Successive Cancellation Decoding for Non-Binary Polar Codes 98
3.3 Construction Methodology of NB-PCs 102
3.3.1 Selection of the Generator Matrix Coefficients Coefficients 102
3.3.2 Selection of Frozen Symbol Channels 104
3.4 Efficient Implementation of the SC-based Polar Decoders 105
3.4.1 Simplified Successive Cancellation Decoder 105
3.4.2 Min-Sum Successive Cancellation Decoder 106
3.4.3 Simplified Min-Sum SC Decoder 108
3.4.4 Extended Min-Sum Successive Cancellation Decoder 108
3.5 Performance and Complexity Comparison of NB-LDPC and NB-PC Decoder 109
3.5.1 Performance Comparison 110
3.5.2 Complexity Comparison 110
3.6 Asymmetrical Extended Min-Sum SC Decoders 113
3.6.1 Introduction to Asymmetrical Extended Min-Sum CN 113
3.6.2 Determination of the Asymmetrical Sizes 114
3.6.3 Computation Reduction Using L-bubble Approach 115
3.6.4 Complexity Analysis and Simulation Results 117
3.7 Polarization-Aware SC Decoder 123
3.7.1 Definition of Nodes Clusters 123

3.7.2 Statistical Computation Using EMS CNs 123
3.7.3 Bubbles Pruning Process 125
3.7.4 Proposed Design of SC-PA Decoders 128
3.7.5 Complexity Analysis and Simulation Results 131
3.8 Conclusion 136

3.1 Introduction to Non-binary Polar Codes

Polar Codes are one of the latest forward-error correction codes proposed by E. Arikan in 2009 [7] that encodes a message of size K into a codeword of size N. Polar codes approach the channel capacity as N tends to infinity (at very long code lengths). At practical code lengths, the decoding performance of the polar codes degrades, therefore, it is combined with the list decoding [62] and cyclic redundancy check [63] for enhancing the decoding performance. The polar codes have been adopted in different standards such as the 5G for control channels. The polar codes are based on channel polarization where the physical channel is transformed into N virtual channels. The virtual channels are polarized into either noiseless channels or extremely noisy channels. The encoding process of the polar codes relies on utilizing the K most reliable (noiseless) channels for encoding the information block.

3.1.1 Polar Transformation and Channel Polarization

Channel polarization [7] is the phenomenon that allows the existence of polar codes. Assume a binary discrete memory-less channel W with an input X and an output Y. Let $I(W)$ indicate the symmetric capacity (the highest rate that can be achieved with reliable communication) of channel W and expressed as

$$I(W) = \sum_{y \in Y} \sum_{x \in X} \frac{1}{2} W(y|x) \log \frac{W(y|x)}{\frac{1}{2}(W(y|0) + W(y|1))}, \tag{3.1}$$

where $W(y|x)$ represents the conditional probability of y given x. The value of $I(W)$ tends to 1 when the channel is noiseless and tends to 0 when the channel is noisy.

The mutual information between symbols x_0 and x_1, and the received symbols y_0 and y_1 at two independent transmission instances of W can be formulated as

$$I(y_0, y_1; x_0) = I(W) = I(y_0, y_1; x_1), \tag{3.2}$$

However, by transforming u_0 and u_1 to generate x_0 and x_1 as

$$\begin{aligned} x_0 &= u_0 \oplus u_1 \\ x_1 &= u_1 \end{aligned}, \tag{3.3}$$

the mutual information between the information and the received symbols is altered such that

$$I(y_0, y_1; x_0) \leq I(W) \leq I(y_0, y_1; x_1). \tag{3.4}$$

The transformation in (3.3) (called also polar transformation) is visualized as in Fig. 3.1. It increases the probability of the correct estimation of u_1 while decreasing the probability of the correct estimate of u_0. Hence, polarizing the channel into two virtual channels. The justification and proof of the polarization phenomenon are deeply discussed in [7].

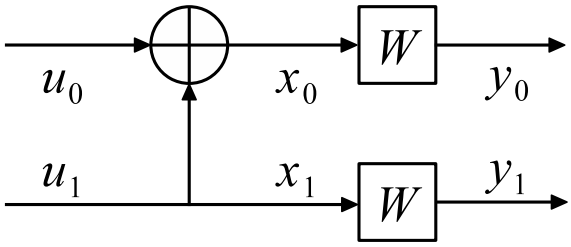


Figure 3.1 – Polar Transformation

3.1.2 Polar Coding

Polar codes utilize the polar transformation to encode an information block $\mathcal{M} = (m_0, \cdots, m_{K-1})$ of size K into a codeword $X = (x_0, \cdots, x_{N-1})$ of size $N = 2^n$. The information symbols are allocated at the K most reliable channels and the remaining $N-K$ channels are frozen (set into a fixed and known symbol, usually zero). The encoding process is based on a generator matrix G_N obtained as the n^{th} Kronecker power of the kernel G_2 defined in (3.5), i.e., $G_N = G_2^{\otimes n}$.

$$G_2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \tag{3.5}$$

The codeword X is generated by multiplying the input U with the generator matrix G_N such that $X = U \cdot G_N$. The encoder's input U is a vector of size N that consists of the K information symbols being allocated at the most reliable indices of U and the remaining $N - K$ indices are frozen (set to a known value, usually zero).

As an example, assume a polar code with $K = 6$ and $N = 8$. This corresponds to a polar code with $N - K = 2$ symbols being frozen (set to 0). Therefore, the frozen symbols are u_0 and u_1 and thus, $U = (0, 0, u_2, u_3, u_4, u_5, u_6, u_7)$. The generator matrix G_8 can be

obtained by performing the Kronecker product on the kernel G_2 such that

$$G_8 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}. \tag{3.6}$$

The generated codeword X is a vector of N symbols and is obtained as $X = U \cdot G_8$ such that

$$[0, 0, u_2, u_3, u_4, u_5, u_6, u_7] \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} u_2 + u_3 + u_4 + u_5 + u_6 + u_7 \\ u_3 + u_5 + u_7 \\ u_2 + u_3 + u_6 + u_7 \\ u_3 + u_7 \\ u_4 + u_5 + u_6 + u_7 \\ u_5 + u_7 \\ u_6 + u_7 \\ u_7 \end{bmatrix}^T. \tag{3.7}$$

3.1.3 Overview on Non-Binary Polar Codes

Non-Binary Polar Codes (NB-PCs) are extended from binary polar codes using different approaches [64]–[67]. In NB-PCs, the symbols and coefficients are all defined over a Galois field of $GF(q = 2^p)$. Therefore, each symbol and coefficient is a vector of $p-$bits.

R. Mori et al. in [64] constructed NB-PCs based on Reed-Solomon matrices. In [65], S. Byun et al. constructed NB-PCs using 4×4 kernels. Furthermore, the construction of the NB-PCs proposed by P. Yuan et al. in [66] is based on a 2×2 non-binary kernel. Additionally, in [67], E. Şaşoğlu et al. proposed a kernel transformation using a random permutation of the non-binary alphabet. Moreover, E. Şaşoğlu et al. also proposed the construction of the NB-PCs using linear permutations defined by the multiplication with a non-zero GF element [68].

However, constructing NB-PCs using higher-dimensional $(l \times l)$ non-binary kernels is more complex because of the increased field order (q) and kernel dimension. Therefore, in the sequel, the NB-PCs are constructed using the original kernel defined by E. Arikan [7] with coefficients defined over $GF(q)$. Thus, the adopted non-binary kernel G_2 is defined as

$$G_2 = \begin{bmatrix} 1 & 0 \\ 1 & \gamma \end{bmatrix}, \ \gamma \ \in \ GF(q) \notin \{0\}. \tag{3.8}$$

Consequently, the kernel transforms the symbols u_0 and u_1 into the symbols x_0 and x_1 as in (3.9). Hence, the structure of the non-binary kernel can be represented as shown in Fig. 3.2

$$\begin{aligned} x_0 &= u_0 \oplus u_1 \\ x_1 &= \gamma \circledast u_1 \end{aligned} \quad \forall \ u_0, u_1, x_0, x_1, \gamma \in GF(q), \tag{3.9}$$

where $\oplus$ and $\circledast$ represent the field addition and multiplication over $GF(q)$ respectively.

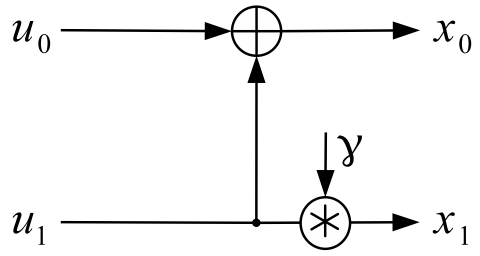


Figure 3.2 – Non-binary Kernel Transformation

Like the binary polar codes, a generator matrix G_N for a code length of $N = 2^n$ can be generated from the kernel G_2 with $\gamma \neq 0$ that varies for each kernel.

For illustration purposes, assume a non-binary generator matrix G_4 expressed as

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & \gamma_0 & 0 & 0 \\ 1 & 0 & \gamma_2 & 0 \\ 1 & \gamma_1 & \gamma_2 & \gamma_1\gamma_2 \end{bmatrix}. \tag{3.10}$$

Such a generator matrix allows transforming an input $U = (u_0, u_1, u_2, u_3)$ into an

output $X = (x_0, x_1, x_2, x_3)$ as follows

$$\begin{aligned} x_0 &= u_0 \oplus u_1 \oplus u_2 \oplus u_3 \\ x_1 &= \gamma_0 \circledast u_1 \oplus \gamma_1 \circledast u_3 \\ x_2 &= \gamma_2 \circledast u_2 \oplus \gamma_2 \circledast u_3 \\ x_3 &= \gamma_1 \circledast \gamma_2 \circledast u_3 \end{aligned} \quad \forall\ x_i, u_i, \gamma_j \in GF(q). \tag{3.11}$$

The aforementioned polar transformation (3.11) can be visualized using the factor graph as depicted in Fig. 3.3. The graph of the polar transformation consists of $n = 2$ layers with each layer having $N/2 = 2$ kernels (generator matrix for $N = 2$).

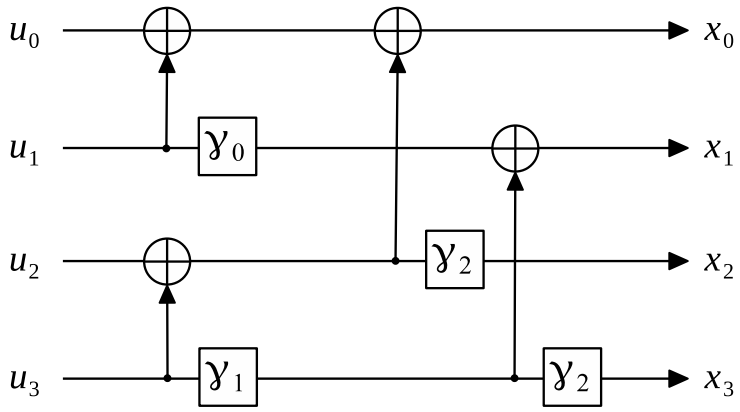
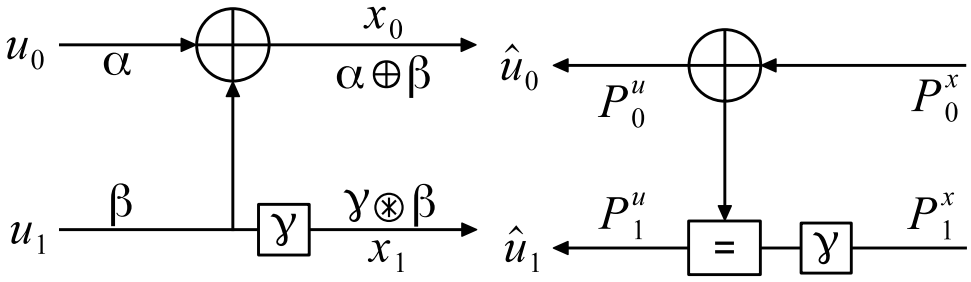


Figure 3.3 – Non-binary Polar Transformation for $N = 4$.

The polar encoding process is identical to the polar transformation with the inputs u_i being frozen (set to a known symbol, zero) for all $i \notin \mathcal{A}_D$ where $\mathcal{A}_D$ denotes the set of indices of the data symbol channels (unfrozen virtual channels).

3.2 Successive Cancellation Decoding for Non-Binary Polar Codes

The Successive Cancellation (SC) decoder generates an estimated message $\hat{U}$ of the information message U using the received message Y. For easier demonstration, assume the decoding of an NB-PC kernel. At the kernel encoding stage demonstrated in Fig. 3.4(a), the value of the inputs u_0 and u_1 is α and β respectively. From those values, the codeword symbols x_0 and x_1 are generated as $\alpha \oplus \beta$ and $\gamma \circledast \beta$ respectively. At the



(a) Encoding Process (b) Decoding Process

Figure 3.4 – NB-PC Kernel Encoding and Decoding Processes

decoding stage demonstrated in Fig. 3.4(b), the Probability Density Functions (PDFs) on x_0 and x_1, denoted as P_0^x and P_1^x respectively, are generated by channel observation and are inputted to the kernel decoder. The PDFs P_0^x and P_1^x are processed by the kernel decoder to obtain the a posteriori probability vector P_0^u and P_1^u used at the decision stage to generate the estimates $\hat{u}_0$ and $\hat{u}_1$ respectively.

The a posteriori probability of u_0, denoted as P_0^u, is generated using the PDFs P_0^x and P_1^x as follows

$$P_0^u(\alpha) = \sum_{\beta \in GF(q)} P_0^x(\alpha \oplus \beta) \cdot P_1^x(\gamma \circledast \beta) \ \forall \alpha \in GF(q). \tag{3.12}$$

Moreover, the estimated symbol $\hat{u}_0$ corresponds to the symbol having the maximum a posteriori probability in as follows the PDF P_0^u.

$$\hat{u}_0 = \underset{\alpha \in \ GF(q)}{\operatorname{argmax}} \ P_0^u(\alpha), \tag{3.13}$$

once the decision is made on $\hat{u}_0$, the probability density function P_1^u is generated as follows

$$P_1^u(\beta) = \mu \cdot P_0^x(\hat{u}_0 \oplus \beta) \cdot P_1^x(\gamma \circledast \beta) \ \forall \beta \in GF(q), \tag{3.14}$$

where μ is a normalization factor obtained such that $\sum_{\beta \in GF(q)} P_1^u(\beta) = 1$. Furthermore, the estimated symbol $\hat{u}_1$ is decided as the symbol having the maximum probability in P_1^u.

The processing of the decoder is more intricate as N increases since the estimation of a symbol $\hat{u}_i, 0 \leq i < N$, depends on all previous estimates $\hat{u}_j, 0 \leq j < i$. In general, an SC decoder has $n = \log_2(N)$ layers, and the PDFs generated at each layer l, $1 \leq l \leq n$, are denoted by $P_i^{(l)}$, with $P_i^{(l)} = (P_i^{(l)}(0), P_i^{(l)}(1), \cdots, P_i^{(l)}(q-1))$, $0 \leq i < N$. The processing schedule of a kernel at layer l and position t is illustrated in Fig. 3.5. The kernel has two

PDF inputs at indices θ_t^{l-1} and ϕ_t^{l-1} derived as follows

$$\begin{aligned} \theta_t^{l-1} &= 2t - (t \mod 2^{n-l}) \\ \phi_t^{l-1} &= 2^{n-l} + 2t - (t \mod 2^{n-l}). \end{aligned} \tag{3.15}$$

for $t = 0, \cdots, N/2 - 1$. The kernel node t and layer l are omitted from the indices θ_t^{l-1} and ϕ_t^{l-1} in the sequel for better readability of the notations. The two PDF input vectors of the kernel are denoted $P_\theta^{(l-1)}$ and $P_\phi^{(l-1)}$ which are processed by the kernel t to generate the output PDFs $P_\theta^{(l)}$ (using a single parity CN (CN) represented as $F()$) and $P_\phi^{(l)}$ (using a repetition node called VN (VN) represented as $G()$) as shown in Fig. 3.5.

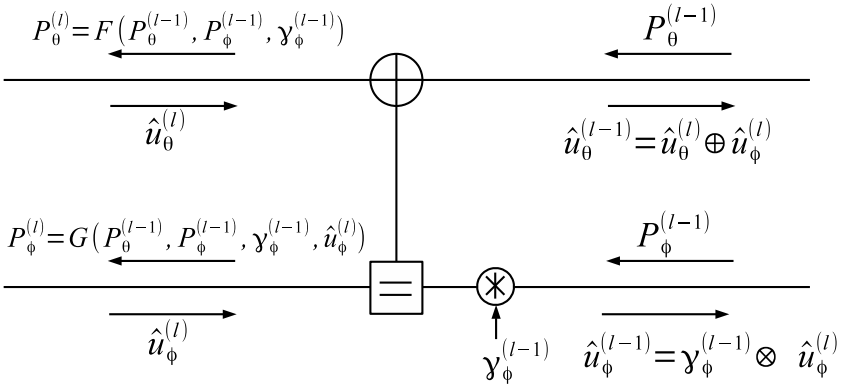


Figure 3.5 – Decoding Process of the t^{th} kernel at layer l

The processing of the CN is identical to the processing expressed in (3.12) and can be generalized for any kernel t with inputs $P_\theta^{(l-1)}$ and $P_\phi^{(l-1)}$, and an output PDF $P_\theta^{(l)}$ as follows

$$P_\theta^{(l)}(\alpha) = \sum_{\beta \in GF(q)} P_\theta^{(l-1)}(\alpha \oplus \beta) \cdot P_\phi^{(l-1)}(\gamma \circledast \beta) \ \forall \alpha \in GF(q). \tag{3.16}$$

Similarly, the processing of the VN in (3.14) can be generalized for any kernel t with inputs $P_\theta^{(l-1)}$ and $P_\phi^{(l-1)}$, and an output PDF $P_\phi^{(l)}$ such that

$$P_\phi^{(l)}(\beta) = \mu \cdot P_\theta^{(l-1)}(\hat{u}_\theta^{(l)} \oplus \beta) \cdot P_\phi^{(l-1)}(\gamma \circledast \beta) \ \forall \beta \in GF(q). \tag{3.17}$$

The estimated symbols are back-propagated to obtain the corresponding symbols at the previous layer by computing the following

$$\begin{aligned} \hat{u}_\theta^{(l-1)} &= \hat{u}_\theta^{(l)} \oplus \hat{u}_\phi^{(l)} \\ \hat{u}_\phi^{(l-1)} &= \gamma \circledast \hat{u}_\phi^{(l)}. \end{aligned} \tag{3.18}$$

The hard decision $\hat{u}_i^{(n)}$ at layer $l = n$ and for a node $i = 0, 1, \cdots, N-1$ is estimated as

$$\hat{u}_i^{(n)} = \begin{cases} 0, & \text{if } i \notin \mathcal{A}_D \\ \underset{\alpha \in GF(q)}{\operatorname{argmax}} P_i^{(n)}(\alpha), & \text{if } i \in \mathcal{A}_D. \end{cases} \tag{3.19}$$

Assume an SC decoder for $N = 8$ as shown in Fig. 3.6. The decoder consists of three layers, and in each layer, there are four kernel decoders. The digits in red represent the processing order of the check and VNs and are depicted for explanation purposes. At the beginning of the decoding process, the decoder processes the first four CNs of layer $l = 1$ (1), then, the first two CNs of layer $l = 2$ (2), and after the first CN of layer $l = 3$ is processed (3). Once the estimated symbol $\hat{u}_0^{(3)}$ is generated, it is propagated to the first VN of the third layer (4). At this stage, the estimated symbols $\hat{u}_0^{(3)}$ and $\hat{u}_1^{(3)}$ are obtained and hence, they are back-propagated to the first two VNs (5) of layer $l = 2$. Thereafter the second CN of layer $l = 3$ is updated and the estimated symbol $\hat{u}_2^{(3)}$ is generated (6). The estimated symbol $\hat{u}_2^{(3)}$ is propagated to the second VN of layer $l = 3$ and then the estimated symbol $\hat{u}_3^{(3)}$ is generated (7). The symbols $\hat{u}_0^{(3)}$, $\hat{u}_1^{(3)}$, $\hat{u}_2^{(3)}$, $\hat{u}_3^{(3)}$ are back-propagated to update the first four VNs of layer $l = 1$ (8) and the processing proceeds as the schedule illustrates in the figure.

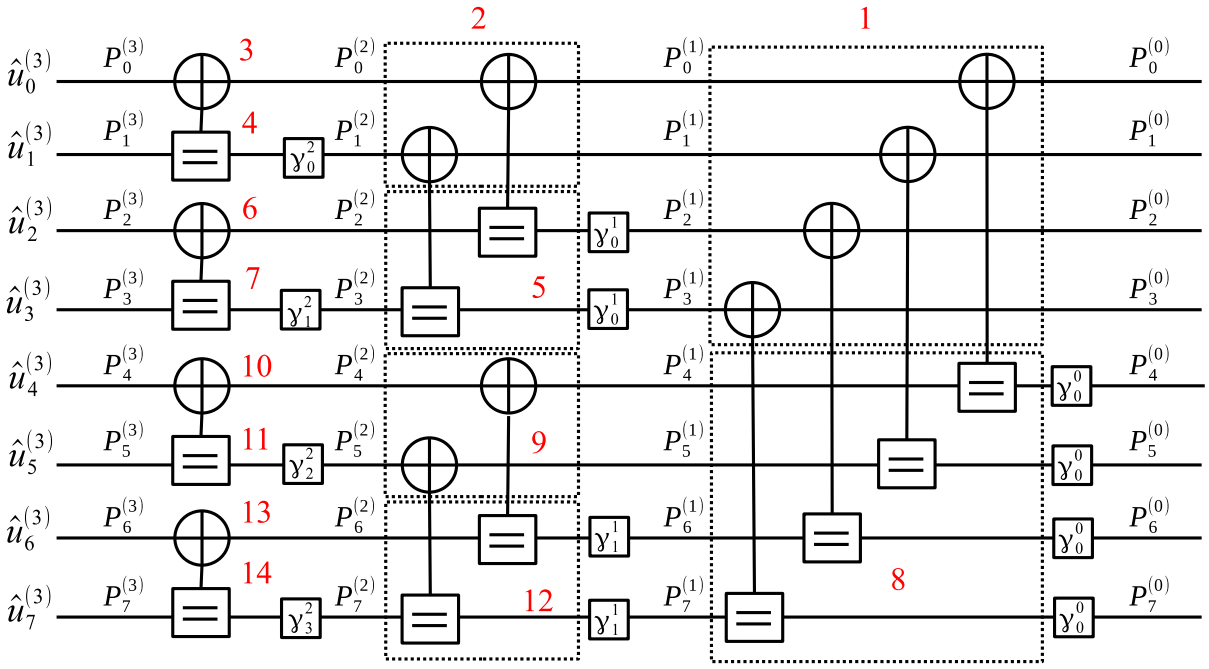


Figure 3.6 – Factor Graph of SC Decoder for $N = 8$

To improve the decoding performance, I. Tal et al. combined the successive cancellation decoding with list decoding [62]. In brief, the SC List decoding initiates L parallel decoding paths at each decision node (non-frozen nodes). A path metric is also defined to assess the reliability of each of the L paths. The SC list decoder suffers from high complexity, especially for high-field orders. This is due to the intensive processing at the selection stage of the surviving paths which requires extracting L paths out of L^2 paths at each of the $K-1$ decision nodes (except for the first decision node). The list decoding can be aided with the cyclic redundancy checks to further improve the decoding performance[63]. In addition, the belief propagation is also proposed in [69] to naturally benefit from parallelism, which can efficiently provide schedule control and high throughput.

3.3 Construction Methodology of NB-PCs

The construction of the NB-PCs depends on two major elements, the coefficients of the generator matrix G_N and the positions of the frozen symbols. In this section, the generation of good coefficients for the generator matrix is discussed in section 3.3.1, and the selection of the frozen positions is discussed in section 3.3.2 representing the comprehensive study presented by V. Savin in [70].

3.3.1 Selection of the Generator Matrix Coefficients Coefficients

The work presented in [67] shows that the random selection of the coefficients provides good decoding performance, but it might be a sub-optimal method. Therefore, the aim of selecting specific values for the coefficients is to accelerate the polarization effect of the virtual channels. The adopted polarizing parameter used for generating the coefficients is the error rate. Nevertheless, the polarizing parameter could be the Bhattacharyya parameter [7] or the mutual information.

The technical objective underlying the coefficient selection is to enhance the disparity between the polarizing parameters of the bad and good synthesized channels. The original channel is denoted by W, and after one polarization step, the channels $W(0)$ and $W(1)$ are synthesized as the bad and good channels respectively. This can be generalized for the synthesized channels at any $n = \log_2(N)$ as

$$W(i_1 \dots i_n) := \left(W(i_1 \dots i_{n-1})^{(i_n)}\right), \forall\ (i_1 \dots i_n) \in \{0,1\}^n$$

Figure 3.7 illustrates a non-binary polar decoder for $N = 8$. The decoder consists of $n = 3$ polarization steps and the synthesized channels in each polarization step l are depicted as $W(i_1 \cdots i_l)$.

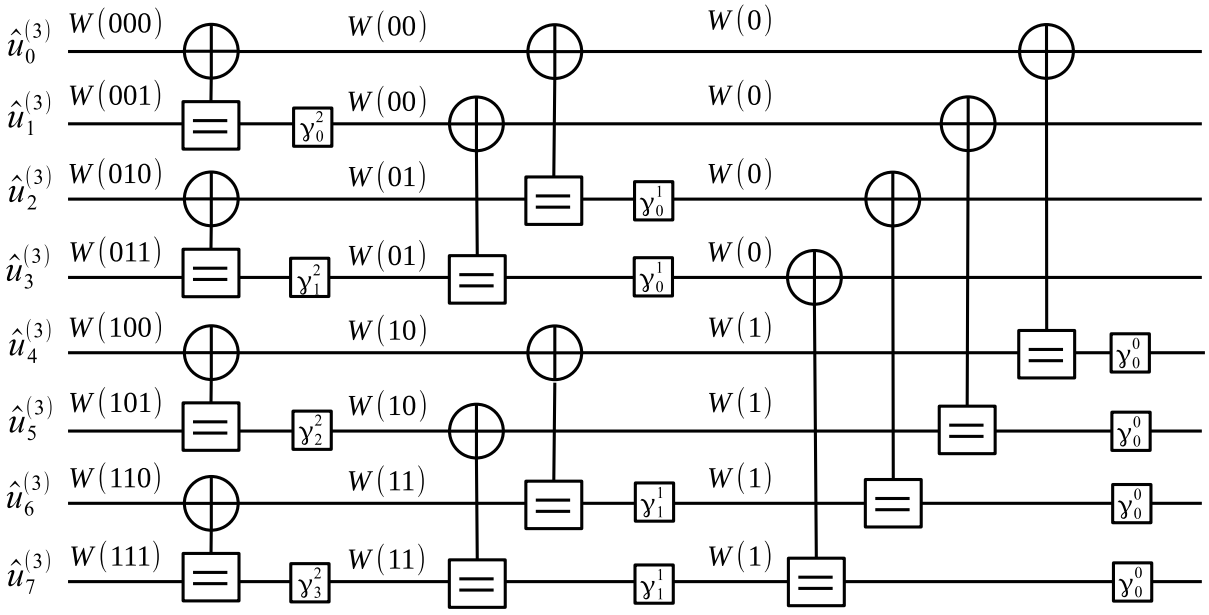


Figure 3.7 – Non-Binary Polar Decoder with $n = 3$ Polarization Steps.

At the first polarization step, $l = 1$, All kernels combine two copies of the real channel W. Therefore, only one coefficient is needed for the four kernels, denoted as $\gamma_0^{(0)}$ and deduced as

$$\gamma_0^{(0)} = \underset{\gamma \in\ GF(q)}{\operatorname{argmax}} |P^{(0)}(\gamma) - P^{(1)}(\gamma)|. \tag{3.20}$$

where $P^{(0)}(\gamma)$ and $P^{(1)}(\gamma)$ denote the polarizing parameter (error rate) of $W(0)$ and $W(1)$ virtual channels when the coefficient $\gamma_0^{(0)} = \gamma$.

The coefficients of the second polarization step are estimated recursively after obtaining the coefficient $\gamma_0^{(0)}$. As shown in the figure, the second polarization step consists of two types of kernels, one combining a bad channel and a good channel (depicted as $W(01)$) and the other combining two good channels (depicted as $W(11)$). Hence, two coefficients are needed and denoted as $\gamma_0^{(1)}$ and $\gamma_1^{(1)}$ respectively. Those coefficients can be deduced as follows

$$\begin{aligned} \gamma_0^{(1)} &= \underset{\gamma \in\ GF(q)}{\operatorname{argmax}} |P^{(00)}(\gamma) - P^{(01)}(\gamma)|. \\ \gamma_1^{(1)} &= \underset{\gamma \in\ GF(q)}{\operatorname{argmax}} |P^{(10)}(\gamma) - P^{(11)}(\gamma)|. \end{aligned} \tag{3.21}$$

where $P^{(i_1 i_2)}(\gamma)$ denotes the polarizing parameters of the channel $W(i_1 i_2)$ given that the corresponding coefficient is set to γ.

The selection of the coefficients proceeds in the same recursive fashion for any desired polarization steps n.

3.3.2 Selection of Frozen Symbol Channels

The frozen channels are the virtual channels that have the highest error rate among all virtual channels. An effective approach for quantitatively calculating the error probability of virtual channels is described below

1. Generate a random vector $U = (u_0, u_1, \cdots, u_{N-1}) : u_i \in GF(q)$.
2. Encode the vector U using the polar transformation to obtain the codeword $X = (x_0, x_1, \cdots, x_{N-1})$.
3. Transmit the codeword over the chosen channel.
4. Run a genie-aided polar decoder to generate the probability density functions of the virtual channels input U denoted as $P^{(n)}_{i=0,\cdots,N-1}$.
5. Compute the error probability ϵ_i for all virtual channels $i = 0, \cdots, N-1$ as

$$\epsilon_i = 1 - P_i^{(n)}(u_i)$$

6. Repeat the steps (1-5) over many transmissions to estimate the average error probability $\mathcal{E}_i$ of each virtual channel $i = 0, \cdots, N-1$ such that

$$\mathcal{E}_i = \mathbb{E}[\epsilon_i],$$

   where $\mathbb{E}$ denotes the expected value.
7. Sort the virtual channels in the ascending order of $\mathcal{E}_i$ and use the most reliable channels to transmit information symbols.

The genie-aided decoder is used to avoid biasing the node error rate when passing the erroneous decision of a node to the proceeding nodes. Hence, it allows for estimating the node error rate independent of previous errors. This is achieved by modifying the VN processing in (3.17) such that

$$P_\phi^{(l)}(\beta) = \mu \cdot P_\theta^{(l-1)}(u_\theta^{(l)} \oplus \beta) \cdot P_\phi^{(l-1)}(\gamma \circledast \beta) \; \forall \beta \in GF(q), \tag{3.22}$$

where $u_\theta^{(l)}$ is the transmitted encoded symbol.

Hence, the set $\mathcal{A}_D$ can be generated by considering the nodes indices with the lowest K error rate $\mathcal{E}_i$. The $N-K$ indices are frozen channels with a fixed input known to both the encoder and the decoder, usually equal to 0.

The channel model used in this work is the additive white Gaussian noise channel with a standard deviation $\sigma = \sqrt{10^{-\text{SNR}/10}}$ where SNR is the Signal-to-Noise Ratio.

The sorted channel indices in ascending order of their node error rate, also called the reliability sequence, for different code rates and lengths are provided in section A.1.

3.4 Efficient Implementation of the SC-based Polar Decoders

The complexity of the successive cancellation decoder is reduced by optimizing different metrics and by introducing minor mathematical approximations. In this section, state-of-the-art SC decoders are presented and discussed.

3.4.1 Simplified Successive Cancellation Decoder

In the work presented by A. Alamdar-Yazdi et al. in [71], the SC decoder is simplified by eliminating useless processing nodes. Assume a polar code with $K = 4$ and $N = 8$, and assume that the symbols u_0, u_1, u_2, and u_4 are frozen (set to zero). The frozen symbols are known to the encoder and the decoder and hence, all processing stages required to generate those symbols can be eliminated without affecting the decoding process or the performance.

The simplified decoder for $N = 8$ is demonstrated in Fig. 3.8. The red nodes are the nodes that can be eliminated with no effect on the following decoding processes.

The first decision to be taken is $\hat{u}_3$ at the second VN of layer $l = 3$ (stage 7). To generate a decision, the VN needs the output of the first two VNs of layer $l = 2$ (5). Moreover, the first two VNs of layer $l = 2$ (5) require the output of the first four CNs of layer $l = 1$ (1). Hence, to generate the decision $\hat{u}_3$, the first four CNs of layer 1, and the first two VNs of layer 2 should be processed. However, since the first three decision nodes are frozen, their corresponding nodes (3,4, and 6) can be omitted. Omitting the processing of stages 3,4, and 6 allows for omitting the processing of the first two CNs of layer $l = 2$ (2). Nevertheless, the VNs shaded in yellow correspond to the simplified VNs

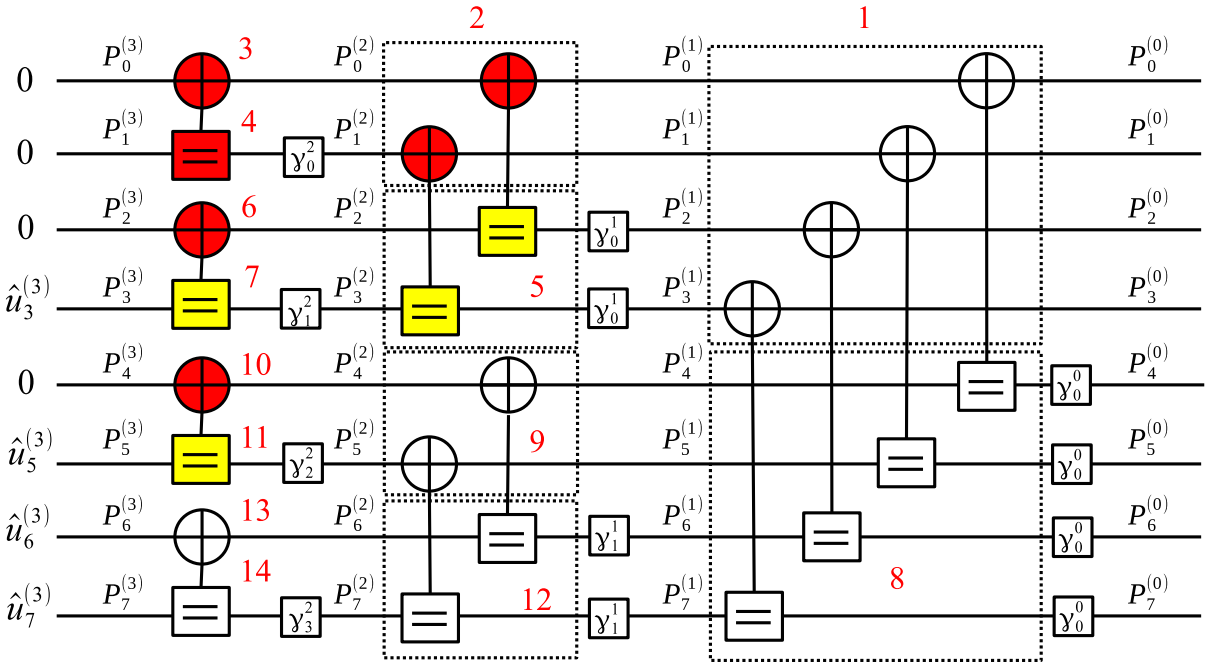


Figure 3.8 – Simplified Polar Decoder for $N = 8$

that have a symbol feedback equal to zero. In such a case, the VN processing in (3.17) can be simplified by omitting the field addition (since $u_\theta^{(l)} = 0$) and re-expressed as

$$P_\phi^{(l)}(\beta) = \mu \cdot P_\theta^{(l-1)}(\beta) \cdot P_\phi^{(l-1)}(\gamma \circledast \beta)\ \forall \beta \in GF(q). \tag{3.23}$$

The simplified successive cancellation decoder reduces the complexity of the global decoder without affecting the decoding performance and schedule.

3.4.2 Min-Sum Successive Cancellation Decoder

The SC decoder has a complexity that is mainly dominated by the CN processing (3.16). Therefore, a mathematical approximation has been introduced by F. Cochachin in [72] to reduce the complexity of the check and the VNs. The approximation used is the min-sum approximation algorithm, originally used for NB-LDPC decoders [37]. The algorithm relies on processing the beliefs from the channel observation in the LLR domain, and hence, allows for reducing the decoding complexity. Processing the reliability messages, the LLR values, in the logarithmic domain allows replacing all the multiplication operations with additions, and hence, no multipliers are required.

In the Min-Sum Successive Cancellation (SC-MS) decoder, the LLR vectors generated

at each layer l, $1 \leq l \leq n$, are denoted by $L_i^{(l)}$, with $L_i^{(l)} = (L_i^{(l)}(0), \cdots, L_i^{(l)}(q-1))$, $0 \leq i < N$. For instance, at layer 1, the input LLR vectors $L_i^{(0)}, 0 \leq i < N$, corresponds to the intrinsic channel observation computed as

$$L_i^{(0)}(\alpha) = \ln\left(\frac{\mathcal{P}(\bar{\alpha}|y_i)}{\mathcal{P}(\alpha|y_i)}\right) \forall \alpha \in \mathrm{GF}(q), \tag{3.24}$$

where $\bar{\alpha} = \arg\max_{\alpha \in \mathrm{GF}(q)} \mathcal{P}(\alpha|y_i)$, i.e., $\bar{\alpha}$ is the hard decision.

The decoding schedule of the SC-MS decoder is identical to that of the SC decoder. The major difference is the internal processing of the kernel nodes. The processing of the CN indicated in (3.16) is modified such that the CN has two LLR vectors $L_\theta^{(l-1)}$ and $L_\phi^{(l-1)}$, and generates an LLR vector $L_\theta^{(l)}$ computed as

$$L_\theta^{(l)}(\alpha) = \min_{\beta \in GF(q)} L_\theta^{(l-1)}(\alpha \oplus \beta) + L_\phi^{(l-1)}(\gamma \circledast \beta)) \ \forall \ \alpha \in GF(q). \tag{3.25}$$

Similarly, the VN update in (3.17) is modified such that the VN has two LLR vectors ($L_\theta^{(l-1)}$ and $L_\phi^{(l-1)}$), and generates an LLR vector $L_\phi^{(l)}$ computed as

$$L_\phi^{(l)}(\beta) = L_\theta^{(l-1)}(\hat{u}_\theta^{(l)} \oplus \beta) + L_\phi^{(l-1)}(\gamma \circledast \beta) \ \forall \ \beta \in GF(q). \tag{3.26}$$

Normalizing the LLR vectors generated in (3.26) should be constantly maintained to prevent arithmetic overflow due to data accumulation. The normalization is performed on an LLR vector $L_\phi^{(l)}$ by subtracting the minimum LLR value of the vector from all elements as follows

$$L_\phi^{(l)}(\beta) = L_\phi^{(l)}(\beta) - \min(L_\phi^{(l)}) \ \forall \ \beta \in GF(q). \tag{3.27}$$

Furthermore, the hard decision $\hat{u}_i^{(n)}$ at layer $l = n$ and for a node $i = 0, 1, \cdots, N-1$ is estimated as

$$\hat{u}_i^{(n)} = \begin{cases} 0, & \text{if } i \notin \mathcal{A}_D \\ \underset{\alpha \in \ GF(q)}{\operatorname{argmin}} L_i^{(n)}(\alpha), & \text{if } i \in \mathcal{A}_D. \end{cases} \tag{3.28}$$

Hence, in an SC-MS decoder, each CN requires q^2 addition operations and each VN requires q additions.

3.4.3 Simplified Min-Sum SC Decoder

Based on the min-sum algorithm, a simplified MS algorithm is presented in [73]. The authors, F. Cochachin et al, proposed simplifying the non-binary SC decoder based on two aspects. The first is optimizing the coefficients of the generator matrix, and the second is simplifying the internal processing of the check and VNs.

The authors proved that while using CCSK modulation (described in section 1.3), the coefficients γ of the generator matrix G_N can be all set to one, yielding a kernel that is similar to the kernel of the binary polar codes (3.5). This simplifies the encoding and decoding processes by eliminating all field multiplications.

In addition, the authors proposed a truncation scheme based on truncating one of the two inputs of the CNs from q down to n_0 with $0 < n_0 \ll q$ which allows reducing the complexity order of the CN from $\mathcal{O}(q^2)$ down to $\mathcal{O}(q \times n_0)$ (for a kernel of order 2).

Assume a CN with two LLR inputs $L_\theta^{(l-1)}$ and $L_\phi^{(l-1)}$. The second input $L_\phi^{(l-1)}$ is truncated from q elements down to n_0 elements. Let the vector $M_\phi^{(l-1)}$ represent a vector of n_0 sorted elements that correspond to the n_0 most reliable elements. Thus, each element of $M_\phi^{(l-1)}$ is a two-tuple element, one for the GF symbol and the other for the LLR value. For clarification, the GF vector of the message $M_\phi^{(l-1)}$ is denoted as $M_\phi^{(l-1)\oplus}$ and the LLR vector is denoted as $M_\phi^{(l-1)+}$.

The CN processing in (3.25) is simplified as follows

$$L_\theta^{(l)}(\alpha \oplus M_\phi^{(l-1)\oplus}(i)) = \min\left(L_\theta^{(l-1)}(\alpha) + M_\phi^{(l-1)+}(i)\right) \forall\ \alpha \in GF(q),\ i = 0, \cdots, n_0 - 1. \tag{3.29}$$

Moreover, the VN processing in the simplified SC-MS decoder is not modified and kept as it is expressed in (3.26).

3.4.4 Extended Min-Sum Successive Cancellation Decoder

The authors C. Peiyao et al. proposed the utilization of the EMS algorithm [74] to reduce the complexity of the successive cancellation decoder. The main contribution is to reduce the size of the propagated messages from q down to n_m where $n_m < q$. Thus, reducing the complexity of the CN from $\mathcal{O}(q^2)$ down to $\mathcal{O}(n_m\sqrt{n_m})$ as in [42].

Assume a CN with two inputs $M_\theta^{(l-1)}$ and $M_\phi^{(l-1)}$ that correspond to the n_m sorted elements of the vectors $L_\theta^{(l-1)}$ and $L_\phi^{(l-1)}$. Each element of $M_i^{(l-1)}$ is a two-tuple element, one for the GF symbol and the other for the LLR value. The GF vector of the message

$M_i^{(l-1)}$ is denoted as $M_i^{(l-1)\oplus}$ and the LLR vector is denoted as $M_i^{(l-1)+}$. The CN processes the inputs to generate the output message $M_\theta^{(l)}$ that consists of n_m elements.

An intermediate matrix T_Σ is computed as follows

$$\begin{matrix} T_\Sigma^{\oplus}(i,j) = M_\theta^{(l-1)\oplus}(i) \oplus \gamma \circledast M_\phi^{(l-1)\oplus}(j) \\ T_\Sigma^{+}(i,j) = M_\theta^{(l-1)+}(i) + M_\phi^{(l-1)+}(j), \end{matrix} : 0 \leq i,j < n_m, \tag{3.30}$$

where $T_\Sigma^{\oplus}$ and T_Σ^{+} denote the GF and the LLR matrices respectively. The concurrent GF and LLR addition operation is denoted by $\boxplus$ such that the equation (3.30) can be rewritten simply as

$$T_\Sigma = M_\theta^{(l-1)}(i) \boxplus M_\phi^{(l-1)}(j) \ : 0 \leq i,j < n_m. \tag{3.31}$$

On the other hand, the VN processing at kernel t is similar to that in (3.26) with two vectors $\bar{L}_\theta^{(l-1)}$ and $\bar{L}_\phi^{(l-1)}$ of size q that are obtained using the messages $M_\theta^{(l-1)}$ and $M_\phi^{(l-1)}$ as follows

$$\begin{matrix} \bar{L}_\theta^{(l-1)}(M_\theta^{(l-1)\oplus}(i)) = M_\theta^{(l-1)+}(i) \\ \bar{L}_\phi^{(l-1)}(M_\phi^{(l-1)\oplus}(i)) = M_\phi^{(l-1)+}(i) \end{matrix} : \ 0 \leq i < n_m. \tag{3.32}$$

The remaining $q - n_m$ elements of $\bar{L}_\theta^{(l-1)}$ and $\bar{L}_\phi^{(l-1)}$ are assigned a default value that corresponds to the maximum LLR of the truncated vectors $M_\theta^{(l-1)}$ and $M_\phi^{(l-1)}$ respectively with an added offset O to compensate the truncation as follows

$$\begin{matrix} \bar{L}_\theta^{(l-1)}(\alpha) = M_\theta^{(l-1)+}(n_m - 1) + O \\ \bar{L}_\phi^{(l-1)}(\beta) = M_\phi^{(l-1)+}(n_m - 1) + O \end{matrix} \forall \ \alpha \notin M_\theta^{(l-1)\oplus}, \ \beta \notin M_\phi^{(l-1)\oplus}. \tag{3.33}$$

3.5 Performance and Complexity Comparison of NB-LDPC and NB-PC Decoder

This section provides a glance at the performance and complexity comparison between the NB-LDPC and NB-PC decoders.

3.5.1 Performance Comparison

The FER performance of the NB-LDPC and NB-PC is simulated over an AWGN with CCSK modulation over GF(64) for different code lengths N and coding rates r.

In the figures Fig. 3.9, Fig. 3.10, Fig. 3.11, and Fig. 3.12, the FER performance is plotted for NB-LDPC and NB-PC at comparable code lengths N and coding rate r. The polar code is not punctured and therefore, the code length should be a power of 2. Also, since the CCSK modulation has a spreading factor of p/q, then the effective coding rate $r_e = rp/q$.

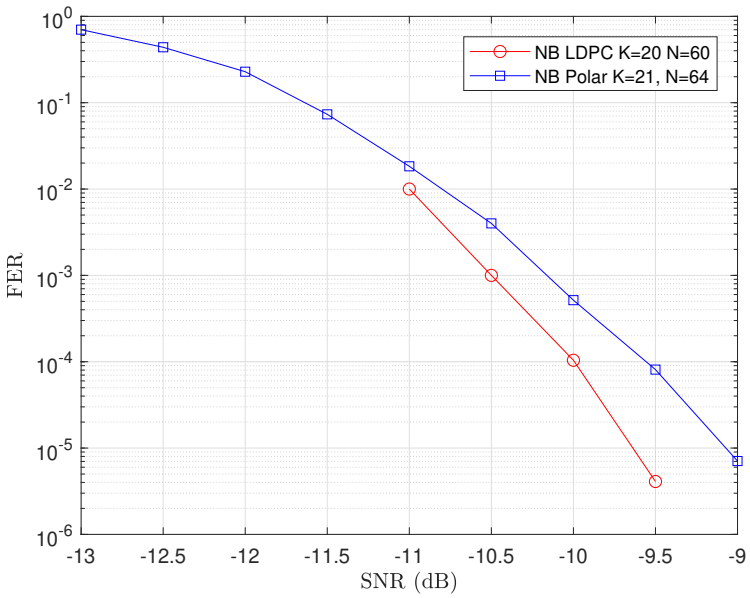


Figure 3.9 – FER Performance at $N \approx 64$ and $r \approx 1/3$.

In Fig. 3.9, the simulation results show an outperformance of the NB-LDPC with $K = 20, N - 60$ over NB-PC with $K = 21, N = 64$. The remaining simulation results indicate that the NB-PC outperformed the NB-LDPC by a maximum of 0.45 dB as in Fig.3.12 for an NB-LDPC code with $K = 160, N = 240$ and an NB-PC with $K = 171 and N = 256$.

3.5.2 Complexity Comparison

The complexity comparison between the NB-LDPC and NB-PC decoders can be explored by analyzing the CN processes performed within the decoders.

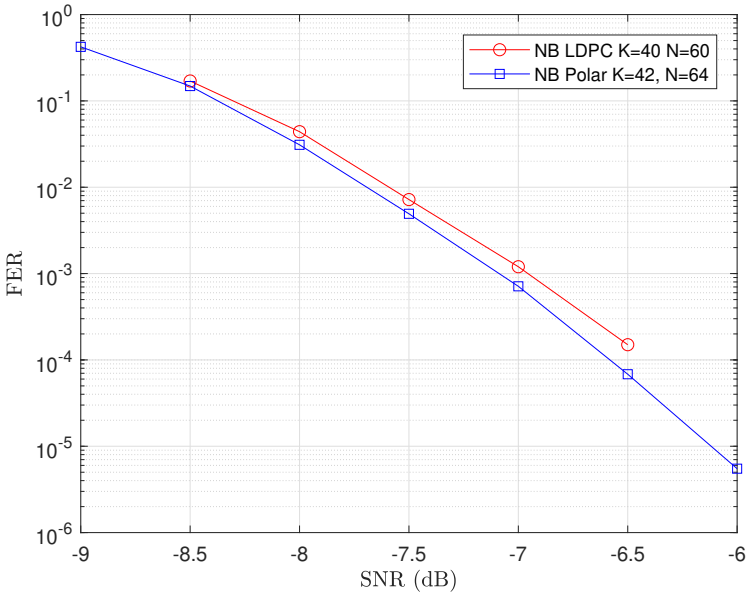


Figure 3.10 – FER Performance at $N \approx 64$ and $r \approx 1/2$.

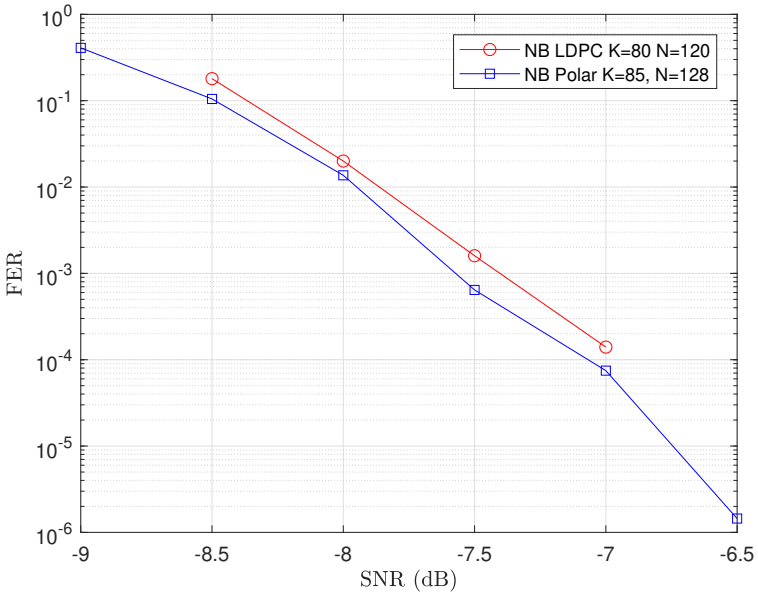


Figure 3.11 – FER Performance at $N \approx 128$ and $r \approx 2/3$.

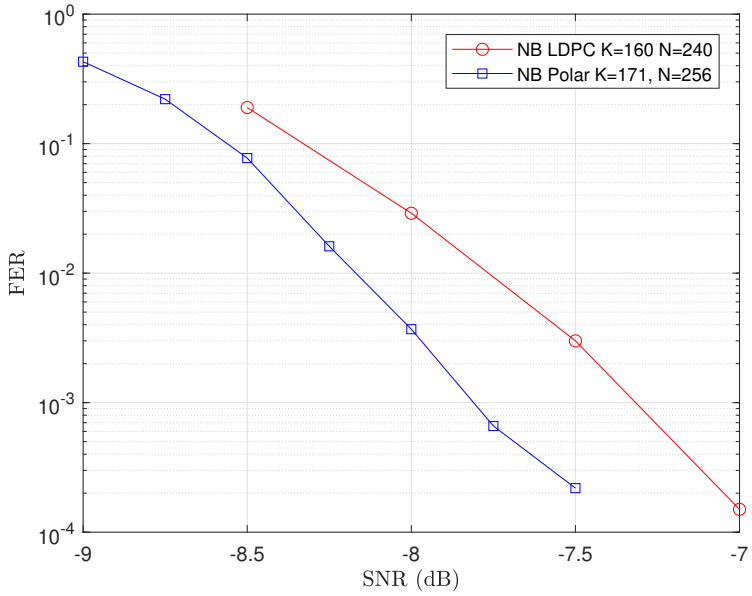


Figure 3.12 – FER Performance at $N \approx 256$ and $r \approx 2/3$.

Since the polar decoder, i.e. the SC decoder, is already discussed in section 3.2, it can be recognized that the processing of the check nodes in the SC decoder is similar to the elementary check nodes in the FB approach proposed to the NB-LDPC decoder as discussed in 2.3.2. Therefore, the number of ECNs in an NB-LDPC and NB-PC decoder can give an insight into the complexity of each decoder.

In the NB-LDPC decoder, the decoding processes iterate for $iter_{max}$ iterations. In each iteration, a variable node is updated $d_v = 2$ times, and a check node updates d_c connected variable nodes. Hence, an average of $\frac{d_v}{d_c}$ CN processing is required per VN. As a result, for a total of N VNs, the total CN processing required T_{CN} can be expressed as below

$$T_{CN} = \frac{2N}{d_c} \tag{3.34}$$

Using the FB approach, the CN processing is decomposed into $3(d_c - 2)$ ECNs. This leads to a total of T_{ECN} ECN processing per iteration which can be expressed

$$T_{ECN} = \frac{2N}{d_c} \times 3(d_c - 2) \tag{3.35}$$

For $iter_{max}$ iterations, the total ECNs performed in an LDPC decoder is

$$T_{LDPC} = iter_{max}\left(6N - \frac{2N}{d_c}\right). \tag{3.36}$$

On the other hand, an SC decoder for a given code length N has $N/2$ CNs per each of the n layers. The CNs of the last layer are used to generate a hard decision and therefore consist of only a GF addition. This results in a total CNs T_{PC} as expressed below

$$T_{PC} = (n-1)\frac{N}{2}. \tag{3.37}$$

This assessment excludes the simplification of the polar decoding that can achieved using the simplified approach noted in section 3.4.1. However, even without using the simplified SC decoding, it is obvious that the complexity of the polar decoder is much less than that of the LDPC decoder.

3.6 Asymmetrical Extended Min-Sum SC Decoders

This section includes the first contributed work to NB-PCs accomplished during this Ph.D. project [75]. The CN processing of the non-binary polar decoder is similar to the NB-LDPC CN for $d_c = 3$, and thus, the earned expertise can be efficiently exploited to simplify the non-binary polar decoder. The Asymmetrical Extended Min-Sum Successive Cancellation (SC-AEMS) decoder is an SC decoder with an optimized CN that considers two inputs of different sizes (asymmetrical inputs). The asymmetrical processing is inspired by the presorting algorithm [46], [47] used for NB-LDPC codes which presort the d_c CN inputs to reduce the complexity of the elementary CNs in the forward-backward approach (see section 2.3.4).

3.6.1 Introduction to Asymmetrical Extended Min-Sum CN

Assume a CN of kernel t at layer l with two sorted input messages $M_\theta^{(l-1)}$ and $M_\phi^{(l-1)}$ (most reliable candidates of $L_\theta^{(l-1)}$ and $L_\phi^{(l-1)}$ respectively) of size n_L. The CN processing can be reduced by eliminating the bubbles (elements) computed in T_Σ (3.31) that are rarely used to generate the output $M_\theta^{(l)}$ bubbles.

To do so, unlike the EMS-based CN, the asymmetrical CN processes the inputs in an asymmetrical manner with two asymmetrical sizes n_L and n_H with $n_L > n_H$. The

messages M_L and M_H of size n_L and n_H respectively are defined as the least and the highest (relatively) reliable input vector of the inputs $M_\theta^{(l-1)}$ and $M_\phi^{(l-1)}$. For clarity, relative reliability represents the reliability of one input with respect to the other.

The estimation of the relative reliability is accomplished by comparing the z^{th} LLR element of the two messages, i.e., $M_\theta^{(l-1)+}(z)$ and $M_\phi^{(l-1)+}(z)$. The LLR element $M_\theta^{(l-1)+}(z)$ reflects the reliability of the z^{th} most reliable symbol compared to the most reliable symbol at index zero, $M_\theta^{(l-1)+}(0)$. Hence, the higher the LLR element $M_\theta^{(l-1)+}(z)$, the higher will be the reliability of $M_\theta^{(l-1)+}(0)$. Therefore, lower candidates are needed from $M_\theta^{(l-1)}$ and vice versa. Comparing the $M_\theta^{(l-1)+}(z)$ and $M_\phi^{(l-1)+}(z)$ helps in assessing which of the two vectors can be further truncated down to n_H since the elements of the highest reliable input at indices $j > n_H$ are less likely to contribute to the output $M_\theta^{(l)}$.

The input with the highest reliability is allocated in M_H with a size of n_H elements (instead of the n_L inputted elements). Similarly, the input with the least reliability is allocated in M_L with a size of n_L elements. This can be expressed as follows

$$\begin{aligned}(M_L^{\oplus}, M_H^{\oplus}) &\leftarrow \begin{cases}(M_\theta^{(l-1)\oplus}, \gamma \circledast M_\phi^{(l-1)\oplus}) & \text{If } (M_\theta^{(l-1)+}(z) < M_\phi^{(l-1)+}(z)) \\ (\gamma \circledast M_\phi^{(l-1)\oplus}, M_\theta^{(l-1)\oplus}) & \text{Otherwise}\end{cases}, \\ (M_L^{+}, M_H^{+}) &\leftarrow \begin{cases}(M_\theta^{(l-1)+}, M_\phi^{(l-1)+}) & \text{If } (M_\theta^{(l-1)+}(z) < M_\phi^{(l-1)+}(z)) \\ (M_\phi^{(l-1)+}, M_\theta^{(l-1)+}) & \text{Otherwise}\end{cases}.\end{aligned} \tag{3.38}$$

The allocation of the messages M_H and M_L allows for redefining the computed region T_Σ in (3.31) to T'_Σ which is expressed as

$$\begin{aligned}T_\Sigma'^{\oplus}(i,j) &= M_H^{\oplus}(i) \oplus M_L^{\oplus}(j), \\ T_\Sigma'^{+}(i,j) &= M_H(i) + M_L(j)\end{aligned} : i = 0, \cdots, n_H - 1; j = 0, \cdots, n_L - 1. \tag{3.39}$$

3.6.2 Determination of the Asymmetrical Sizes

The sizes n_L and n_H can be found using empirical and statistical approaches using Monte-Carlo simulation. The size of the least reliable message M_L, n_L, and the most reliable message M_H, n_H can be determined using the statistical estimation of the contribution rate matrix at the CNs of layer $l = 1$. To do so, compute T'_Σ as in (3.39) with an initial size of the field order, i.e., $n_L = n_H = q$. Let the contribution rate matrix C of size $q \times q$ be the matrix that includes the probability of each element T'_Σ being selected

at any element of $M_\theta^{(l)}$ such that

$$C(i,j) = C(i,j) + \begin{cases} 1 & \text{If } T'_\Sigma(i,j) \in M_\theta^{(1)} \\ 0 & \text{Otherwise} \end{cases} \quad \forall\ t = 0, 1, \cdots, N/2-1; 0 \leq i,j < q. \quad (3.40)$$

The estimation of the contribution rate matrix C to an output element is aggregated over the CN of kernels $t = 0, \cdots, N/2-1$ at layer $l = 1$ (since they exhibit similar polarization levels, it helps in increasing the estimation accuracy) as expressed in (3.40).

For better accuracy, the estimation of the contribution rate matrix C is considered when the decoded codeword is valid and is accumulated over N_r successfully decoded frames. Therefore, the estimation of an element $C(i,j)$ is amplified by $N_r \times N/2$. Consequently, to get a normalized contribution rate, all elements of the matrix C should be divided by $N_r \times N/2$ as follows

$$C = \frac{2C}{(N_r \times N)}. \quad (3.41)$$

The contribution rate is illustrated in Fig. 3.13 for $K = 42$ symbols and $N = 256$ symbols over CCSK modulation and on $GF(64)$ at SNR of -17 dB with $z = 2$ and $N_r = 100$. It can be noticed that the size of the elements from the least reliable input contributes much more to the output than the elements of the most reliable input. Based on both the statistical study presented and the empirical results, the values $n_L = 20$ and $n_L = 8$ show a good performance-complexity trade-off.

3.6.3 Computation Reduction Using L-bubble Approach

The internal processing of T'_Σ computed in (3.39) can be reduced using the L-bubble approach, initially proposed for NB-LDPC (see section 2.3.2). The L-bubble only considers the elements computed in the first two rows and columns of T'_Σ. The computed elements can be categorized into four regions R_0, R_1, R_2, and R_3, where each region is expressed as follows

$$\begin{aligned} R_0 &= \{M_H(0) \boxplus M_L(j)\}_{j=0,\cdots,n_L-1}, \\ R_1 &= \{M_H(i) \boxplus M_L(0)\}_{i=0,\cdots,n_H-1}, \\ R_2 &= \{M_H(1) \boxplus M_L(j)\}_{j=1,\cdots,n_L-1}, \\ R_3 &= \{M_H(i) \boxplus M_L(1)\}_{i=2,\cdots,n_H-1}. \end{aligned} \quad (3.42)$$

The complete structure of the AEMS CN is presented in Fig. 3.14 and formalized as

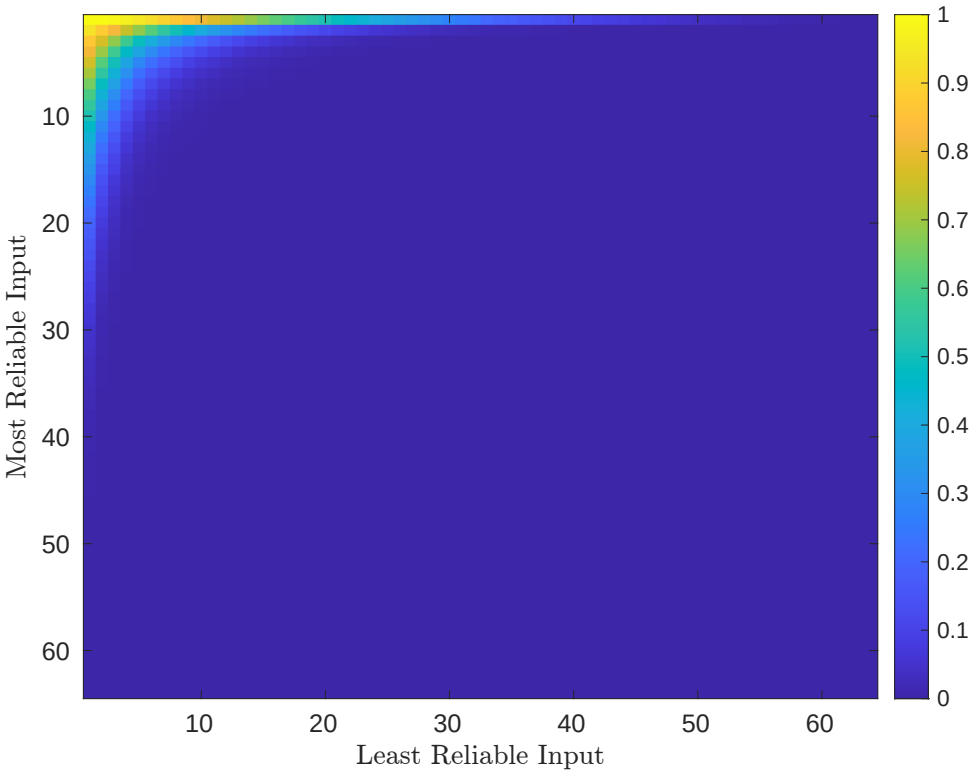


Figure 3.13 – Contribution Rate Matrix C for Layer $l = 1$ CNs.

depicted in Algorithm 2.

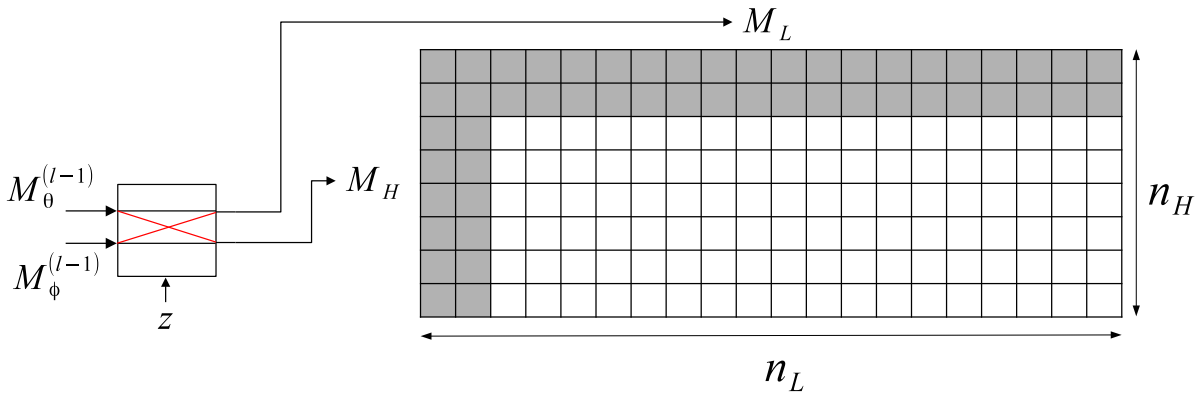


Figure 3.14 – Schematic Structure of an AEMS CN.

Algorithm 2: Asymmetrical EMS-based CN

Input: $M_\theta^{(l-1)}$, $M_\phi^{(l-1)}$, n_L,n_H, z.
Pre-processing Steps: Comparison of the z^{th} most reliable LLR value:
if $M_\theta^{(l-1)}(z) < M_\phi^{(l-1)}(z)$ **then**
  $(M_L, M_H) \leftarrow (M_\theta^{(l-1)}, M_\phi^{(l-1)})$
else
  $(M_L, M_H) \leftarrow (M_\phi^{(l-1)}, M_\theta^{(l-1)})$
end
Processing Steps:
Step 1: Generate T'_Σ as in (3.39).
Step 2: Extract n_L most reliable distinct bubbles in T_Σ to obtain $M_\theta^{(l)}$.
Output: M_θ^l

On the other hand, the VN processing at kernel t is similar to the VN processing for the SC-EMS as in (3.32) and (3.33) with $n_m = n_L$ elements.

3.6.4 Complexity Analysis and Simulation Results

The complexity of the SC-AEMS can be estimated by estimating the arithmetic operations per CN over the different decoders. In all algorithms that use both sorting and normalization, the real addition operations are less than the GF additions. This is because the first LLR element in both inputs is equal to zero. Therefore, the LLR values of the first row and the first column are identical to their corresponding non-zero input elements

(see R_0 and R_1 with $M_H(0) = 0$ and $M_L(0) = 0$ in (3.42)). The GF and LLR addition operations for the SC-AEMS are compared with the SC-MS and SC-EMS. In addition, the L-bubble EMS-based SC decoder (denoted as SC-LEMS) is also considered in the comparison, and the results are provided in Table 3.1.

Table 3.1 – Arithmetic Operations Performed per CN

Algorithm	**GF Additions**	**Real Additions**
SC-MS	q^2	q^2
SC-EMS	$n_m\sqrt{n_m}$	$n_m\sqrt{n_m} - (2n_m - 1)$
SC-LEMS	$4n_m - 4$	$2n_m - 3$
SC-AEMS	$2(n_H + n_L) - 4$	$n_H + n_L - 3$

Using Monte-Carlo simulation over $GF(64)$, an SC-AEMS decoder with $n_H = 8$, $n_L = 20$, and $z = 2$ is found to perform as well as the SC-LEMS with $n_m = 20$ and is negligibly degraded compared to the performance of the SC-MS decoder. Therefore, the CN processing can be realized by performing the arithmetic and logic (over $GF(64)$) operations as indicated in Table 3.1. The required GF and LLR additions for the MS decoder are 4096 operations for each. Furthermore, the required GF and LLR addition operations for the SC-EMS decoder with $n_m = 20$ are 90 and 51 respectively. For the SC-LEMS with $n_m = 20$, the required GF and LLR addition operations are 76 and 37 respectively. Lastly, in the proposed SC-AEMS decoder with $n_H = 20, n_L = 8$, the total GF additions are only 52 operations and 25 LLR additions.

Compared to the SC-EMS CN, the total computed candidates by an SC-AEMS CN is reduced by 42% for GF operations, and by 72% for LLR addition operations. Additionally, when compared to the SC-LEMS CN, the SC-AEMS CN achieves a reduction of 30% in terms of GF addition and 32% in terms of LLR addition operations.

On the other hand, the SC-AEMS decoder requires a comparator at each CN. This sums up to $N/2$ CNs in each layer over the $n-1$ layers (at the last layer, only one element from each input is required to get a hard decision). Therefore, $N/2 \times n - 1$ comparators and multiplexers are required as demonstrated in Fig. 3.14.

The proposed decoder is simulated over a BI-AWGN channel with CCSK modulation and a sequence of size $q = 64$ chips. The coefficient γ is set to 1 for all codes based on the results provided in [73], and the frozen positions are obtained by analyzing the error rate at each channel position $i = 0 \cdots N - 1$ using a genie-aided decoder [7].

The simulation results for codes $N = 64$, $N = 256$, and $N = 1024$ in Fig.3.15, Fig. 3.16, and Fig 3.17 respectively. The continuous black plot corresponds to the finite-block achievable rate computed as in (1.5). The performance of the SC-MS is plotted dashed-black, and the SC-LEMS performance with $n_m = 20$ and $n_m = 14$ are plotted in dashed and dotted blue plots respectively. Lastly, the SC-AEMS with $n_H = 8$ and $n_L = 20$ is plotted in dashed-cyan.

The effective code rate over a CCSK modulation is deduced as $r_e = r \times S_F$ where $r = K/N$ is the coding rate and S_F is the spreading factor. Since the simulations are performed over $GF(q = 64)$, the effective code rate is $r_e = r \times \frac{6}{64}$.

The performance of the SC-LEMS decoder with $n_m = 14$ is provided to compare the difference in decoding capability between the SC-LEMS and SC-AEMS at a similar arithmetic complexity ($n_m = (n_H + n_L)/2$). As shown in Fig.3.15 and Fig. 3.16, the performance of the SC-LEMS decoder with similar complexity to that of the SC-AEMS has an additional degradation in performance of around 0.15 dB. Thus, the asymmetrical EMS algorithm maintains the performance with fewer candidates.

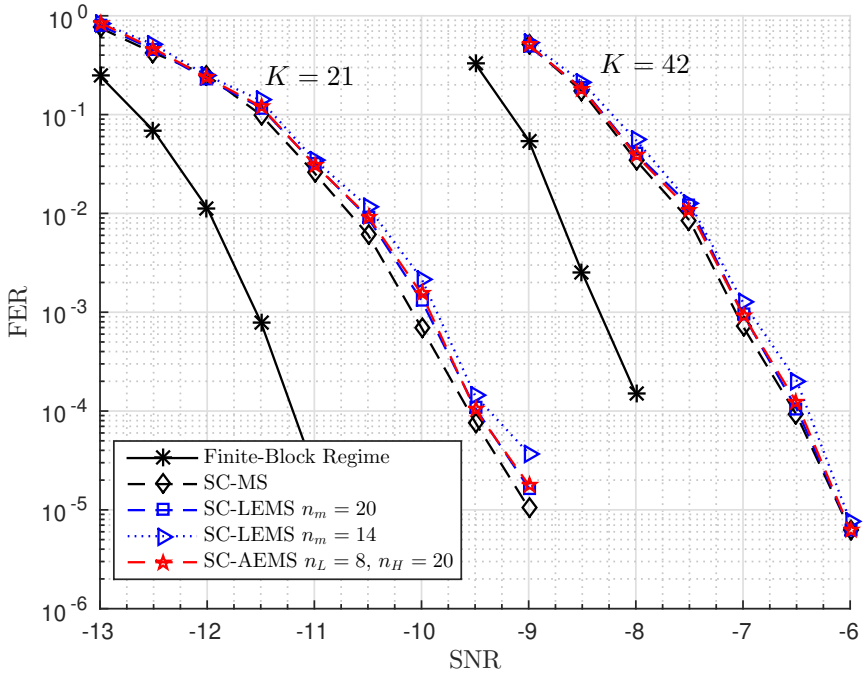


Figure 3.15 – Simulation Results over $GF(64)$ for $N = 64$, $r \approx 1/3$ ($r_e \approx 1/32$) and $r \approx 2/3$ ($r_e \approx 1/16$) respectively.

Nevertheless, the SC-AEMS decoder with $n_H = 8$ and $n_L = 20$ is simulated over

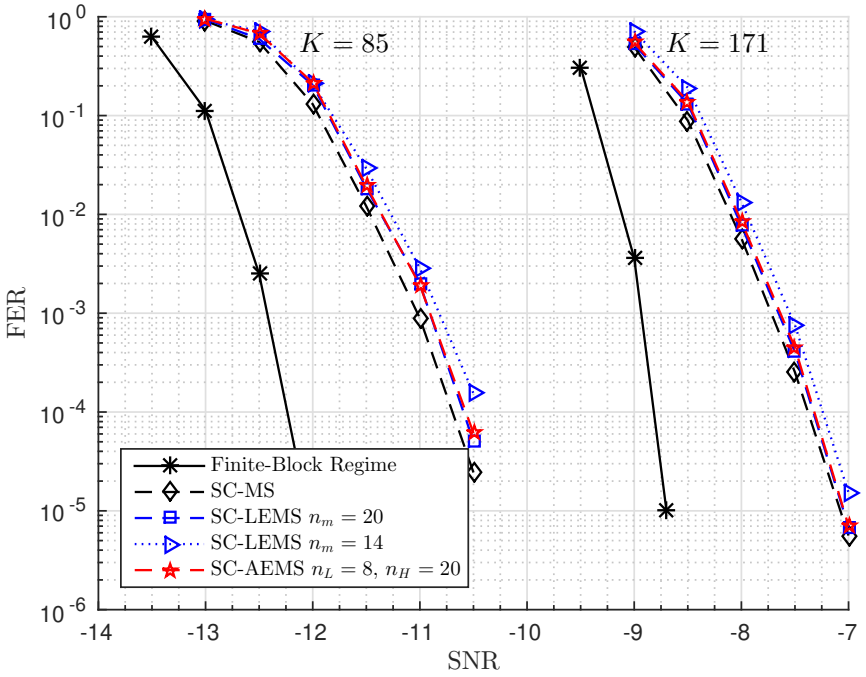


Figure 3.16 – Simulation Results over $GF(64)$ for $N = 256$, $r \approx 1/3$ ($r_e \approx 1/32$) and $r \approx 2/3$ ($r_e \approx 1/16$) respectively.

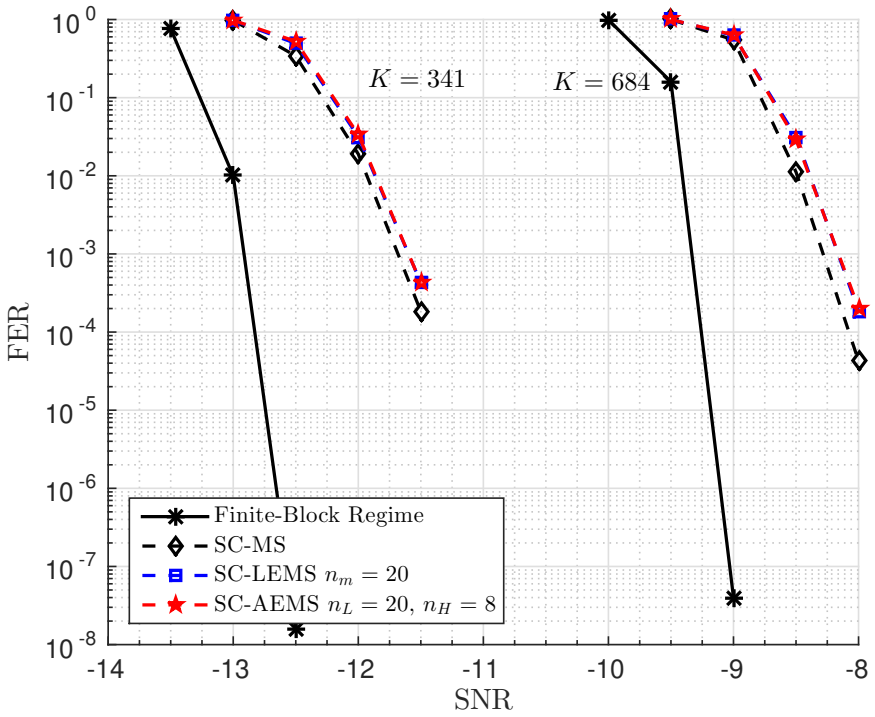


Figure 3.17 – Simulation Results over $GF(64)$ for $N = 1024$, $r \approx 1/3$ ($r_e \approx 1/32$) and $r \approx 2/3$ ($r_e \approx 1/16$) respectively.

BI-AWGN channel with Binary Phase Shift Keying (BPSK) modulation for $K = 42$ and $N = 128$ on GF(64). In Fig. 3.18, the performance of the SC-MS is plotted in black, SC-EMS with $n_m = 20$ in blue, SC-LEMS with $n_m = 20$ in red, and SC-AEMS with $n_L = 20$ and $n_H = 8$ in cyan.

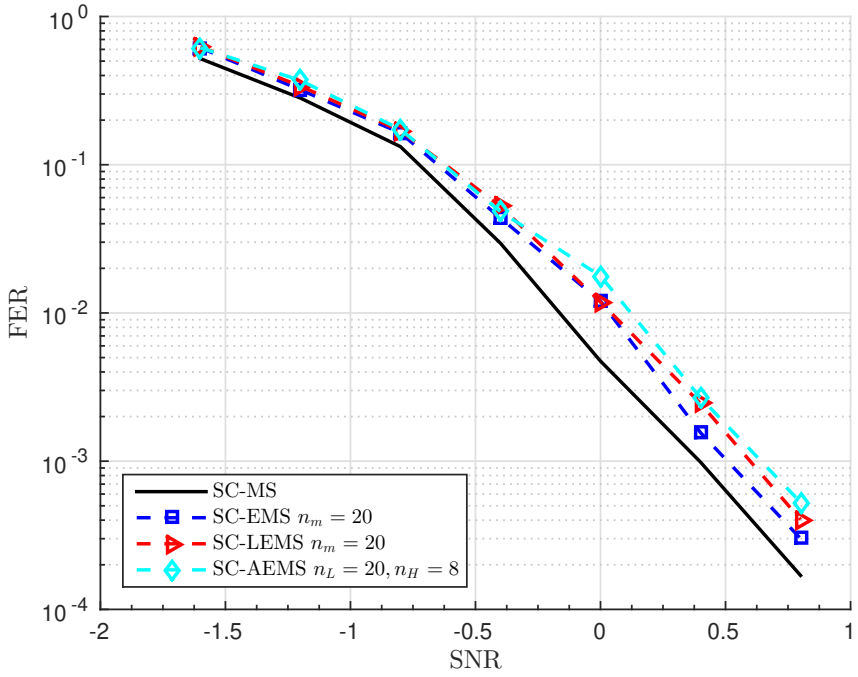


Figure 3.18 – Performance Simulation of SC-AEMS Decoder with BPSK Modulation

As shown, the SC-AEMS decoder has a degraded performance of around 0.3 dB to that of the SC-MS decoder. It can be noticed that the performance of the SC-EMS experiences a degradation of about 0.125 from that of the SC-MS performance. In addition, the performance of the SC-LEMS decoder is degraded by around 0.25 dB from the SC-MS decoder. Focusing on the performance of both the SC-EMS and SC-LEMS justifies the reason behind the degraded performance. Firstly, the message size $n_m = 20$ of the SC-EMS is insufficient for good convergences under BPSK modulation (similar behavior experienced in NB-LDPC CN with $d_c = 3$). Secondly, the L-bubble region is inappropriate for generating the output messages under BPSK modulation. Lastly, the asymmetrical sizes approach has no major impact on performance degradation.

Therefore, a more dynamic and sophisticated approach should be developed that considers the effect of polarization on the CN processing.

3.7 Polarization-Aware SC Decoder

The Polarization-Aware Successive Cancellation (SC-PA) decoding algorithm is the second contribution to the non-binary polar decoder during this PhD. The SC-PA is a rate-dependent simplified decoding approach of the EMS-decoding algorithm for NB-PC. It is based on investigating the statistical behavior of the computations performed within the CNs at different polarization levels.

At first, the notion of node cluster is defined in section 3.7.1. Then, the notion of potential regions assigned to the node cluster is derived in section 3.7.2. In addition, the proposed pruning process is explained in section 3.7.3. Further, the proposed design of the polarization-aware decoding is discussed in section 3.7.4. Lastly, the complexity and the simulation analysis are discussed in section 3.7.5.

3.7.1 Definition of Nodes Clusters

In a polar decoder of size N, each of the n layers comprises $N/2$ kernels, with each kernel consisting of a CN and a VN. In [70], nodes are clustered based on their polarization level to determine optimal polarizing coefficients γ for the polar code.

Following a similar pattern, the kernel cluster $S_s^{(l)}$ at layer l can be defined as a set of kernels with indices $t = s \times (2^{n-l})$ up to $t = (s+1) \times (2^{n-l}) - 1$, where s ranges from 0 to $2^{l-1} - 1$. Consequently, cluster $S_s^{(l)}$ encompasses 2^{n-l} kernels, with all CNs sharing similar polarization levels and similarly, all VNs having similar polarization levels (albeit different from those of the CNs). The reason for including VNs in the same cluster as CNs is their shared inputs. As a result, any optimization applied to the CNs will also affect the corresponding VNs.

Thus, the definition of these clusters serves as the foundation for dynamic computation and message size determination for the kernels within each cluster, based on statistical estimations made at the CNs.

3.7.2 Statistical Computation Using EMS CNs

Estimating the potential elements in T'_Σ (3.39) helps in reducing the overall complexity of the SC decoder. This can be achieved by the computation of the bubble pattern matrices $\mathcal{B}_t^{(l)}$. A bubble pattern matrix is defined as a matrix of size $n_m \times n_m$ which includes the probability of the element (bubble) of $T'_\Sigma(i,j)$ (with symmetrical inputs, i.e., $n_L = n_H =$

n_m) being selected at any output element of $M_\theta^{(l)}$ at layer l and CN at kernel index t.

To do so, for a decoded codeword, a bubble pattern matrix $\mathcal{B}_t^{(l)}$ can be computed as follows

$$\mathcal{B}_t^{(l)}(i,j) = \begin{cases} 1 & \text{If } T'_\Sigma(i,j) \in \ M_\theta^{(l)} \\ 0 & \text{Otherwise} \end{cases}, \quad \forall\ 0 \leq t < N/2;\ 1 \leq l < n. \tag{3.43}$$

Estimating the average behavior of the bubble pattern of a given CN gives useful insight into partitioning its bubbles (the elements of T'_Σ) into two sets: the set of useful bubbles (i.e. the ones that are frequently used) and the set of useless bubbles (i.e., the ones that are never, or only rarely used). This partition allows for reducing the complexity of the CN processing by only computing the set of useful bubbles. Experimentation shows that the CNs within the same cluster $S_s^{(l)}$ have similar sets of useful bubbles. It is thus logical to accumulate the statistical information from the CNs, first to increase the accuracy of estimation of the average behavior of bubbles, and to apply to each cluster an identical truncation pattern. Aggregating the nodes into clusters helps in reducing the total customized regions of T'_Σ from $(n-1) \times N/2$ down to $N/2 - 1$. To do so, the contribution rate matrices $\mathcal{C}_s^{(l)}$ are defined as the aggregated statistical information $\mathcal{B}_t^{(l)}$ of the CNs at $t = s \times (2^{n-l})$ up to $t = (s+1) \times 2^{n-l} - 1$. Hence, the contribution rate matrix $\mathcal{C}_s^{(l)}$ at layer l and cluster s can be expressed as

$$\mathcal{C}_s^{(l)} = \frac{1}{2^{n-l}} \sum_{t=s.2^{n-l}}^{(s+1).2^{n-l}-1} \mathcal{B}_t^{(l)}. \tag{3.44}$$

The contribution rate matrices $\mathcal{C}_s^{(l)}$ are taken into account if and only if the frame is successfully decoded to avoid biasing the statistical results by a faulty decision. The estimation of $\mathcal{C}_s^{(l)}$ can be accumulated for N_r well-decoded frames to generate accurate statistics as

$$\mathcal{C}_s^{(l)} = \frac{\mathcal{C}_s^{(l)}}{N_r}. \tag{3.45}$$

The last layer $l = n$ includes simple CNs that perform one field addition of the most reliable element of the two inputs to obtain the estimate $\hat{u}_i\ \forall\ i \mod 2 = 0$. Therefore, they are excluded from the CN simplification process.

3.7.3 Bubbles Pruning Process

The pruning process is performed offline to get the pruned form of T'_{Σ} denoted as $T'^{(l)}_{\Sigma_s}$ for all node clusters $s = 0, \ldots, 2^{l-1} - 1$ in layers $l = 1, \ldots, n-1$. In the pruning process, a threshold $\mathcal{P}_t$ is defined as the minimum contribution rate value for a bubble to be computed. Therefore, any bubble with a contribution rate $\mathcal{C}_s^{(l)}(i,j) < \mathcal{P}_t$ is omitted from the CN processing of cluster $S_s^{(l)}$.

The indicator matrices $\mathcal{R}_s^{(l)}$ can be found for all clusters $s = 0, \ldots, 2^{l-1} - 1$ over the $l = 1, \ldots, n-1$ layer as follows

$$\mathcal{R}_s^{(l)} = \begin{cases} 1 & \text{If } \mathcal{C}_s^{(l)}(i,j) > \mathcal{P}_t \\ 0 & \text{Otherwise} \end{cases}. \tag{3.46}$$

These $\mathcal{R}_s^{(l)}$ matrices indicate which bubbles of $T'^{(l)}_{\Sigma_s}$ should be exclusively computed to generate $M_\theta^{(l)}$, i.e.,

$$T'^{(l)}_{\Sigma_s}(i,j) = \begin{cases} M_H(i) \boxplus M_L(j) & \text{If } \mathcal{R}_s^{(l)}(i,j) = 1 \\ (0, +\infty) & \text{Otherwise} \end{cases} : 0 \leq i,j < n_{s'}^{(l-1)}, \tag{3.47}$$

where $n_{s'}^{(l-1)}$ is the size of the input messages (output size of the connected cluster in the previous layer), i.e., $s' = \lfloor s \rfloor$.

The tuple $(0, \infty)$ correspond to the GF and LLR values given to GF tuple $T'^{(l)\oplus}_{\Sigma_s}(i,j)$ and LLR tuple $T'^{(l)+}_{\Sigma_s}(i,j)$ of the element $T'^{(l)}_{\Sigma_s}(i,j)$ respectively.

Moreover, since the bubbles of the first row (having maximum contribution) are directly generated from the corresponding input $M_L^{(l-1)}$, the omitted bubbles consequently reduce the maximum size of the propagated messages such that the CNs within the node cluster $S_s^{(l)}$

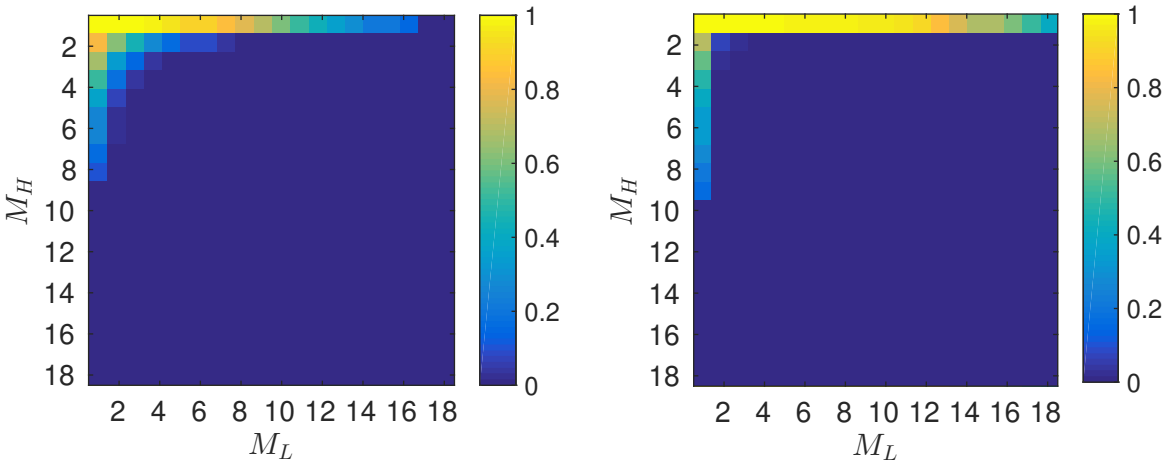
$$n_s^{(l)} = \sum_{j=0}^{n_{s'}^{(l-1)}} \mathcal{R}_s^{(l)}(0,j). \tag{3.48}$$

In addition, the size of inputs at layer $l = 1$ denoted as $n_0^{(0)}$ is equal to the chosen EMS size n_m, i.e., $n_0^{(0)} = n_m$.

As a result, both the check and VNs in cluster $S_s^{(l)}$ have an output size of $n_s^{(l)}$ elements but different input sizes. The CN in cluster $S_s^{(l)}$ generates an output of $n_s^{(l)}$ and has an inputs of size $n_{s'}^{(l-1)}$. Therefore, only $n_s^{(l)}$ among the $n_{s'}^{(l-1)}$ elements of the inputs are con-

sidered (sufficient) to generate $n_s^{(l)}$ reliable elements (see summation boundaries of (3.47)). The reason behind generating the $n_{s'}^{(l-1)}$ elements from the nodes in the previous layer is that the VNs belonging to cluster $S_s^{(l)}$ process the $n_{s'}^{(l-1)}$ elements of the corresponding inputs to generate $n_s^{(l)}$ reliable elements at the output. This is due to the processing nature of the VN processing which relies on the intersection between the inputs to generate reliable (meaningful) elements.

For further clarification, the contribution rate matrix $\mathcal{C}_1^{(2)}$ is depicted for $N = 64$ with $K = 11$ in Fig. 3.19(a) and $K = 42$ in Fig. 3.19(b) estimated at SNR of -13.5 dB and -7.5 dB respectively. As shown, the two matrices have different potential regions.

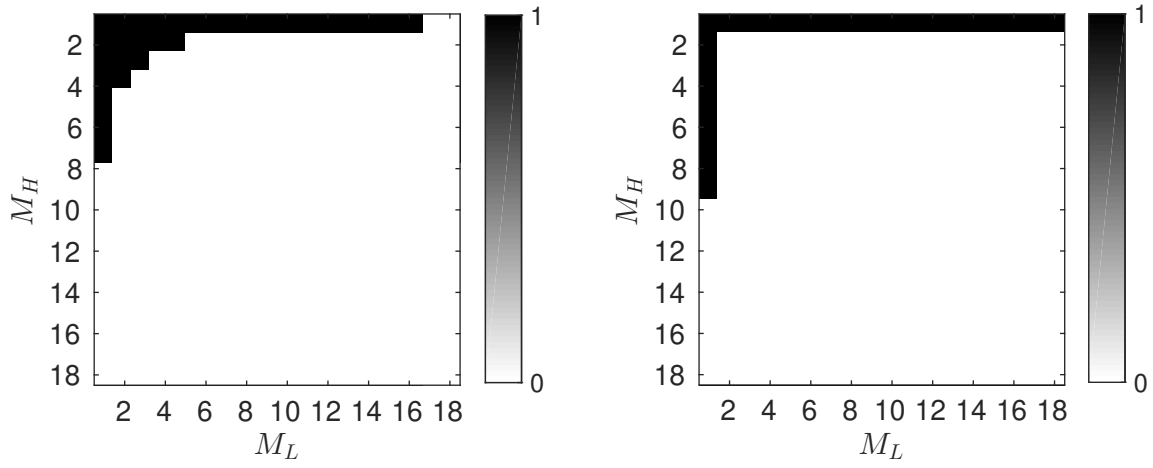


(a) $K = 11$ (b) $K = 42$

Figure 3.19 – Contribution Rate $\mathcal{C}_1^{(2)}$ (in %) at Layer $l = 2$ and node cluster $s = 1$ for $N = 64$.

When applying the pruning process as in (3.46) to the contribution rates in Fig. 3.19 with a threshold $\mathcal{P}_t = 0.12$, the region $\mathcal{R}_1^{(2)}$ is deduced for $K = 11$ as in Fig. 3.20(a) and for $K = 42$ as in Fig. 3.20(b). As shown, the two regions have different potential elements even though they correspond to the same node cluster and layer, but over different code rates. It can be noticed that the potential region of $K = 42$ is the first row and column, which means that no LLR additions are required to generate the output since $M_L^+(0)$ and $M_H^+(0)$ are equal to zero. Furthermore, the output size is deduced using (3.48 such that $n_0^{(2)} = 16$ for $K = 11$, and $n_0^{(2)} = 18$ for $K = 42$.

Additionally, algorithm 3 summarizes the aforementioned approach used for designing a polarization-aware decoder.



(a) $K = 11$ (b) $K = 42$

Figure 3.20 – Pruned Computed Region $\mathcal{R}_1^{(2)}$ (in %) at Layer $l = 2$ and node cluster $s = 1$ for $N = 64$ over GF(64).

Algorithm 3: Designing Procedure for Polarization-Aware Polar Decoder.

Input: $q, N, K, n_m, N_r, \mathcal{P}_t$
Initialization Step:
Set $\mathcal{C}_s^{(l)}$ to the $n_m \times n_m$ null matrix, $l = 1, 2, \ldots, n-1$, $s = 0, 1, \ldots, 2^l - 1$.
Pre-Processing Step: Determine SNR μ required to have a FER of 10^{-2} using EMS-based SC with n_m.
while $i < N_r$ **do**
  Step 1: Receive a codeword Y through an AWGN at SNR μ.
  Step 2: Decode Y with the EMS-based SC decoder with parameter n_m.
  Step 3: Trace useful bubbles for statistical analysis
  Compute (3.39) and (3.43).
  Step 4: Computation of Contribution Rate Matrix
  if *decoding success* **then**
    Compute (3.44)
end
Step 4: Normalization of $\mathcal{C}_s^{(l)}$ as in (3.45)
Step 5: Pruning Process: Deduce $\mathcal{R}_s^{(l)}$ and $n_s^{(l)}$ as in (3.46) and (3.48).
Output: $\mathcal{R}_s^{(l)}$ $\forall$ $l = \{1, \ldots, n-1\}, s = \{0, \ldots, 2^{l-1} - 1\}$.

The schematic graph for a polar decoder with $N = 8$ is depicted in Fig. 3.21.

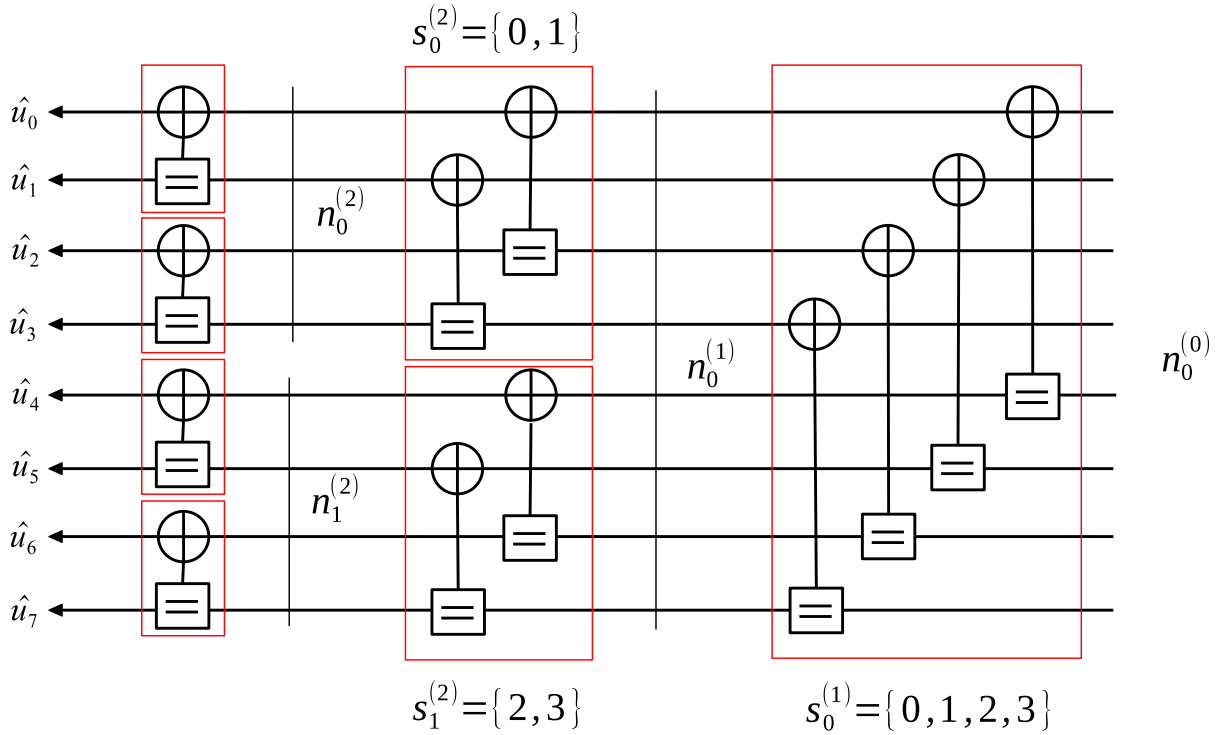


Figure 3.21 – SC Decoder for $N = 8$ with Node Clusters Illustration

3.7.4 Proposed Design of SC-PA Decoders

As it can be noticed, the design of the polarization-aware decoder depends on two parameters, the first is the maximum input size denoted as $n_0^{(0)}$, and the second is the inclusion threshold $\mathcal{P}_t$. The former can be found by estimating the decoding performance of the conventional EMS-based decoder at different sizes n_m. Once the good n_m is found in the EMS-based decoder, it can be adopted in the designing stage of the polarization-aware decoder, i.e., $n_0^{(0)} = n_m$.

As an example, let the desired code be $N = 64$ with $K = 42$ on GF(64). The first requirement is to find the decoding performance for different n_m using Monte-Carlo simulation (or mathematical approximation as in [70]). The decoding performance of the EMS-based decoder over different sizes n_m is depicted in Fig. 3.22. As shown, the parameter $n_m = 18$ is the minimum size that maintains a good performance at a FER $\approx 10^{-4}$. Hence, the parameter n_m is set to 18 for the statistical analysis held at a FER region of 10^{-2}, i.e., at SNR of -7.5 dB.

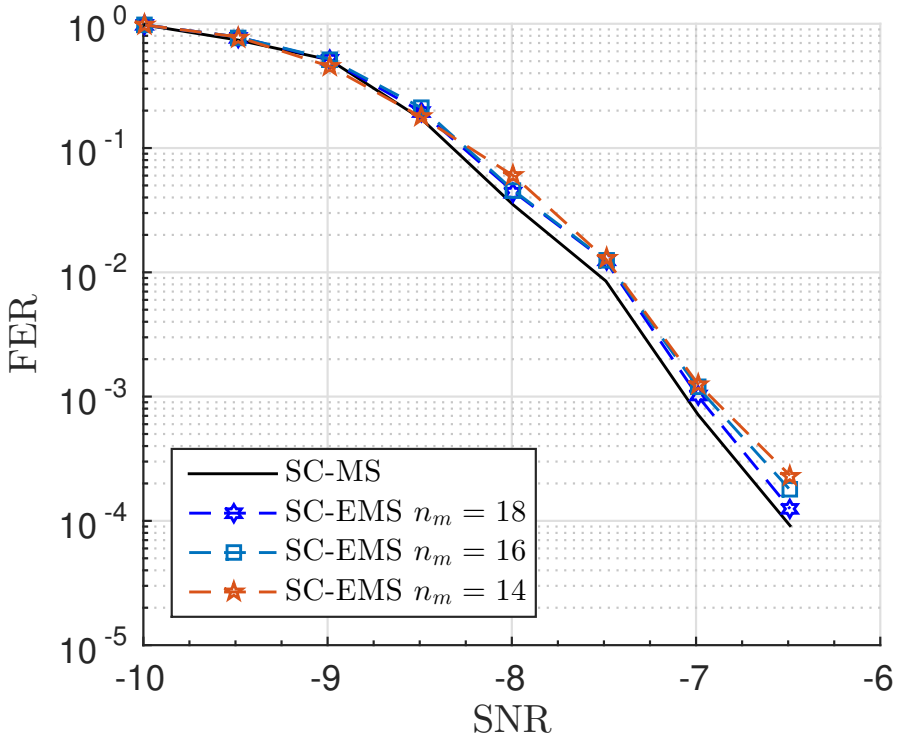


Figure 3.22 – EMS-based SC Decoder with $K = 42$ and $N = 64$ over Different n_m sizes.

Furthermore, once the contribution rate matrices $\mathcal{C}_s^{(l)}$ are obtained, the pruning process is launched offline. The threshold value $\mathcal{P}_t$ impacts both the complexity and the performance such that choosing a high threshold reduces the complexity and the decoding performance, and vice versa.

In Fig. 3.23, the FER performance of the proposed SC-PA over different threshold values is depicted. For $\mathcal{P}_t = 0.05$ up to $\mathcal{P}_t = 0.12$, the decoding performance is negligibly degraded compared to the performance of the SC Min-Sum decoder (SC-MS). At $\mathcal{P}_t \geq 0.15$, the degradation increases to 0.25 dB. Hence, the threshold $\mathcal{P}_t = 0.12$ is considered to have a good performance-complexity trade-off.

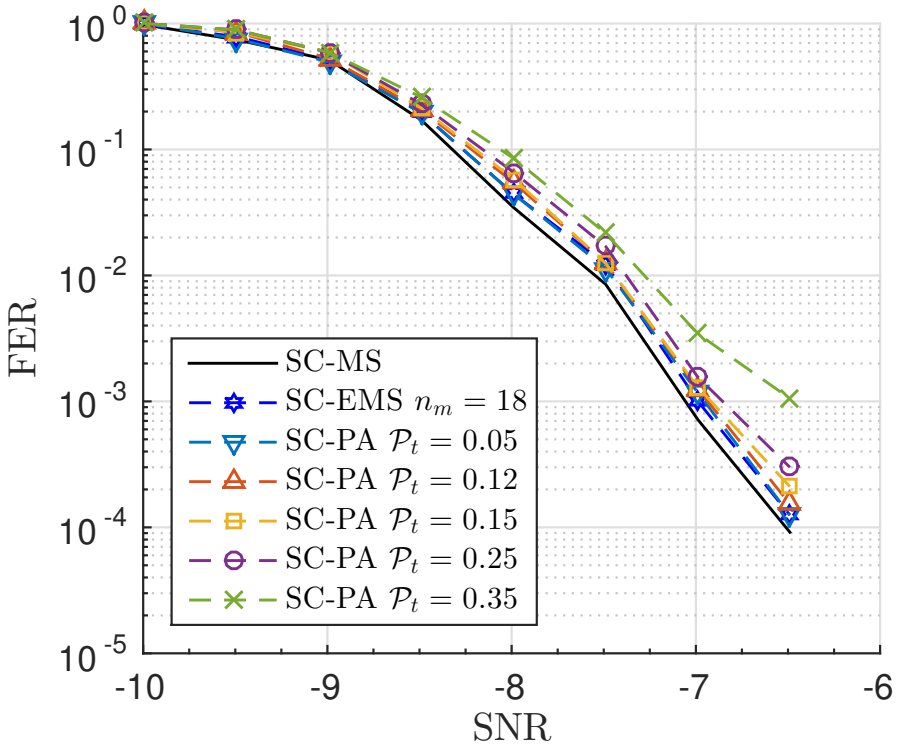


Figure 3.23 – Performance of SC-PA Decoder over Different Threshold Values $\mathcal{P}_t$ for $N = 64$ and $K = 21$.

Based on the described algorithm in Algo. 3, the design parameters $n_0^{(0)}$ and $\mathcal{P}_t$ have been found for different codes as shown in Table 3.2.

3.7.5 Complexity Analysis and Simulation Results

The global complexity reduction of the SC-PA decoder can be estimated by estimating the savings in the computation operations saved at the CNs compared to an EMS-based decoder with n_m elements at each input. In addition, the Asymmetrical Extended Min-Sum CN (AEMS) [75] with n_L and n_H elements is also considered in the comparison.

An EMS-based CN [42] with n_m candidates requires computing O_{GF} GF additions and O_{LLR} LLR additions that can be estimated as follows

$$\begin{aligned} O_{\text{GF}} &= n_m\sqrt{n_m}, \\ O_{\text{LLR}} &= n_m\sqrt{n_m} - 2n_m + 1. \end{aligned} \tag{3.49}$$

Furthermore, the AEMS-based CN has two inputs of sizes n_L and n_H and uses the L-bubble [43]. Therefore, the total computed GF candidates O_{GF} and LLR candidates O_{LLR} can be expressed as follows

$$\begin{aligned} O_{\text{GF}} &= 2(n_L + n_H) - 4, \\ O_{\text{LLR}} &= n_L + n_H - 3. \end{aligned} \tag{3.50}$$

In an SC decoder of size N, there are $N/2$ CNs in each layer $l = 1, \ldots, n-1$ that perform O_{GF} GF additions and O_{LLR} LLR additions. Therefore, the total GF and LLR addition operations over an SC decoder of size N are as follows:

$$\begin{aligned} T_{\text{GF}} &= O_{\text{GF}} \times (n-1) \times N/2 \\ T_{\text{LLR}} &= O_{\text{LLR}} \times (n-1) \times N/2 \end{aligned}, \tag{3.51}$$

where O_{GF} and O_{LLR} depends on the CN processing.

In [75], the adopted value of n_L and n_H in an AEMS-based CN is 20 and 8 respectively for any code length N. As for the EMS-based CN, the minimum value of n_m that maintains a good performance is $n_m = 18$ for $N \leq 512$ and $n_m = 22$ for $N = 1024$. On the other hand, the total operations performed by the SC-PA decoder cannot be formulated in an equation due to the dynamic computations in each node cluster.

Based on the aforementioned complexity analysis, the SC-PA decoder saves around half the computation resources compared to the asymmetrical EMS-based SC decoder. This excludes the savings in the complexity of the repetition (variable) nodes and sorting procedures at CNs since they are not taken into consideration in this assessment. Never-

Table 3.2 – Design Parameters for SC-PA Decoder and Required Arithmetic Operations over Different Codes.

K	μ	SC-PA		SC-AEMS		SC-EMS	
		T_{GF}	T_{LLR}	T_{GF}	T_{LLR}	T_{GF}	T_{LLR}
$N = 64, \mathcal{P}_t = 0.12, n_0^{(0)} = 18$				$n_L = 20, n_H = 8$		$n_m = 18$	
11	-13.5 dB	3776	916	8320	4000	12160	6560
21	-10.5 dB	4082	788				
42	-7.5 dB	4370	424				
58	-5 dB	4168	56				
$N = 256, \mathcal{P}_t = 0.08, n_0^{(0)} = 18$				$n_L = 20, n_H = 8$		$n_m = 18$	
42	-14 dB	22362	6048	46592	22400	68096	36736
85	-11.5 dB	24294	5744				
17	-8 dB	24742	3180				
213	-6.5 dB	23526	1878				
$N = 1024, \mathcal{P}_t = 0.09, n_0^{(0)} = 25$				$n_L = 20, n_H = 8$		$n_m = 22$	
171	-15 dB	151986	45756	239616	115200	474264	276480
341	-12 dB	180396	45420				
683	-8.5 dB	222488	33314				
922	-6 dB	223104	11916				

theless, the SC-PA decoder requires additional $N/2 \times (n-1)$ comparators and multiplexers for the reallocation of the inputs as expressed in (3.38).

The proposed decoder is simulated over a BI-AWGN channel with CCSK modulation for a bunch of code rates and lengths on GF(64). For better readability, the proposed work is depicted in dashed red and denoted as SC-PA in all provided simulation plots. In addition, the min-sum SC decoder is denoted as SC-MS and depicted in a plain black plot. The SC-EMS with $n_m = 18$ is depicted for $N \leq 256$ and $n_m = 22$ for $N = 1024$. The performance of the SC-AEMS is similar to that of the SC-EMS [75] and hence, is not provided for simpler illustration of figures. The offset value (3.33) is set to 0.5 for the SC-EMS, and 0.5 for the SC-AEMS, and 0.7 for the SC-PA.

The simulation results for $N = 64$ with $K = 11$, $K = 21$, $K = 42$, and $K = 58$ are depicted in Fig. 3.24 from left to right respectively using the parameters in Table 3.2.

Moreover, the simulation results in Fig. 3.25 correspond to $N = 256$ with $K = 43$, $K = 85$, $K = 171$, and $K = 213$ respectively from left to right using the parameters in Table 3.2.

Lastly, for $N = 1024$, the simulation results in Fig. 3.26 include the following infor-

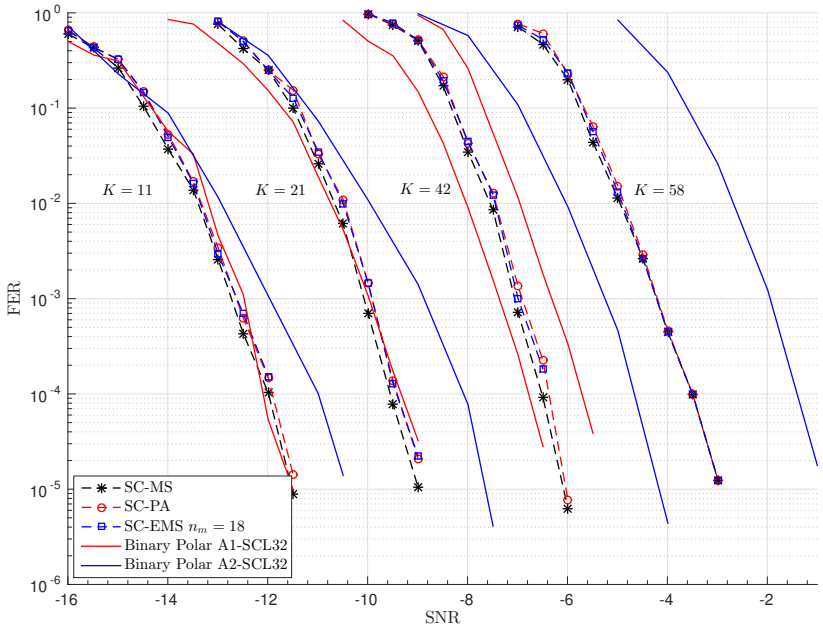


Figure 3.24 – Simulation Results for $N = 64$ over GF(64).

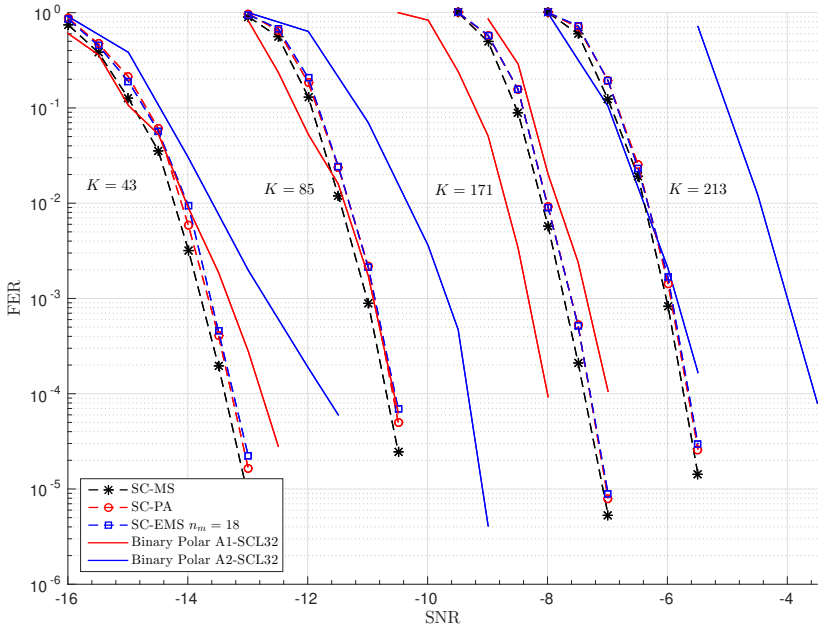


Figure 3.25 – Simulation Results for $N = 256$ over GF(64).

mation block lengths (from left to right in order) $K = 171$, 341, 683, and 922.

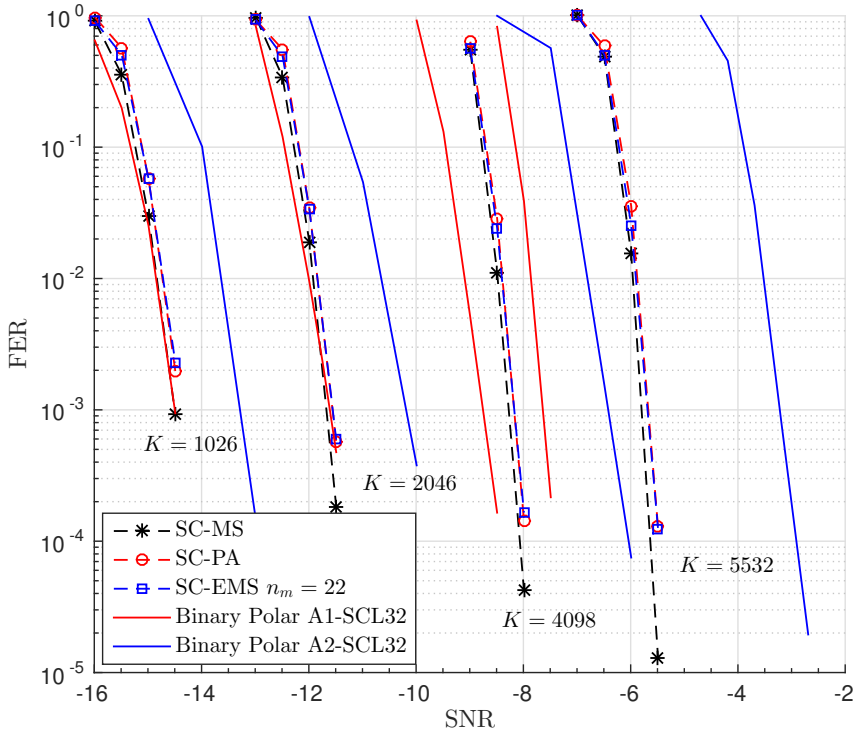


Figure 3.26 – Simulation Results for $N = 1024$ over GF(64).

The polarization-aware decoder has a similar performance as the SC-EMS with $n_m = 18$ and $n_m = 22$ for code lengths $N \leq 512$ and $N \geq 1024$ respectively. Compared to the min-sum SC decoder, the degradation is at a maximum of 0.2 dB at a FER of 10^{-4}. Moreover, the SCL decoder using the first approach has a similar performance at a low code rate but outperforms the NB SC decoder at high code rates. This is expected since at a very high code rate $r = 9/10$, the effective coding rate is $r_e = 27/320$ due to CCSK spreading. However, for the binary SCL32 it has a coding rate of 27/320, thus, the effect of the decoding error correction is much more effective than that of the CCSK spreading. As for the second approach, the NB SC always outperforms the SCL32 with the same code rate and repetition for compensating the CCSK spreading.

The FER performance of the binary SCL decoder is also compared with the non-binary SC decoder plotted using two approaches. The first approach (denoted as A1-SCL32) assumes a binary polar code with a similar payload (in bits) and the same code rate as the effective code rate of the NB-PC, i.e., $K_b = Kp$ and $r_b = r_e$. Therefore, the

simulated binary polar code has $K_b = K \times 6$ and $N_b = K_b/r_b$. The FER performance of the aforementioned binary decoder is illustrated via the continuous red plots. The second approach (denoted as A2-SCL32) assumes a binary polar code with the same payload as the NB-PC and the same coding rate (instead of the effective code rate), i.e., $K_b = Kp$ and $r_b = r$, along with a repetition of $1/S_F$ (shifting the plot to left by $10\log_{10}(S_F)$ dB) such that the effective coding rate is similar to that of the NB-PC. The FER performance of this approach is depicted via the continuous blue plots. The comparison with binary SCL is to show the decoding capabilities of each configuration and not to assess the performance of binary and non-binary polar decoders since different modulation schemes are used.

For better comparison, the SC-PA decoder has been simulated using BPSK modulation over a BI-AWGN channel. For BPSK-modulated symbols, the kernel coefficient γ should be greater than 1. Thus, the coefficients are generated based on the methodology described in [70] which depends on selecting the GF coefficient that maximizes the polarization difference between the CN (bad channel) and VN (good channel) of the same kernel (provided in section A.2).

The performance of the SC-PA with BPSK modulation is illustrated in Fig. 3.27 for $K = 42$ symbols and $N = 128$ on GF(64). The performance of the SC-MS decoder is plotted in black. Similarly, the performance of the SC-EMS decoder with $n_m = 20, 25, 32$ is represented in starred red, blue, and green plots respectively. In addition, the performance of the SC-PA decoder is depicted in squared plots over different n_m and $\mathcal{P}_t$. Furthermore, the performance of (SC and SC List (SCL) decoding [62] with a list size of 32) 5G binary polar code (using Aff3ct Simulator [76]) with $N_b = Np = 768$ bits and $K_b = Kp = 252$ bits.

As shown, the non-binary SC decoder outperforms the binary SC decoder by 0.75 dB at a FER of 10^{-2}. The SCL decoder slightly outperforms the non-binary SC decoder but experiences an error floor at FER of 10^{-3} leading to the outperformance of the non-binary SC.

Furthermore, the SC-EMS decoder with BPSK requires a higher size of inputs than the SC-EMS decoder with CCSK. For a performance degradation of less than 0.2 dB, the SC-EMS decoder requires messages of size $n_m = 25$ (contrary to CCSK+SC-EMS which requires $n_m = 18$). As for the SC-PA, the required size at the first layer is $n_0^{(0)} = 32$ with $\mathcal{P}_t = 0.1$. The SC-PA requires 32 candidates at the first layer only. The two clusters in the second layer have a size of $n_0^{(2)} = 26$ and $n_0^{(2)} = 21$. Therefore, only the CNs at the

first layer require additional sorting. Based on (3.49) and (3.51) the total GF additions and LLR additions for $N = 128$ is $T_{\text{GF}} = 48000$ and $T_{\text{LLR}} = 29184$. As for the SC-PA decoder, the total computed GF and LLR elements are $T_{\text{GF}} = 18872$ and $T_{\text{LLR}} = 7616$ respectively. This leads to a reduction of around 60% in terms of GF additions and 70% in terms of LLR additions.

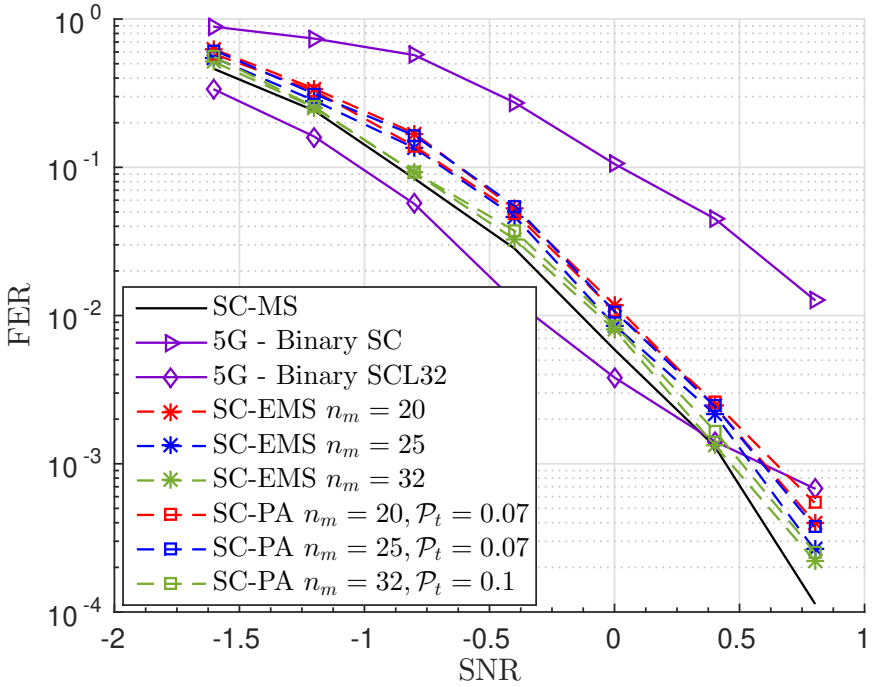


Figure 3.27 – Simulation Results of SC-PA for $N = 128$ and $K = 42$ over BPSK Modulation.

Lastly, it is worth mentioning that the obtained complexity results are based on the computed bubbles in all CNs at all layers. This introduces a bias in favor of the SC-PA decoder since the useless CNs that are considered can be omitted using the simplified successive cancellation mentioned in section 3.4.1. As a result, there might be a decrease in the mentioned complexity reduction ratio.

3.8 Conclusion

This chapter includes the study of the NB-PCs and decoders. The polar codes are introduced in section 3.1, the introduction explains the channel polarization and the

polar transformation in section 3.1.1, and the polar encoding in section 3.1.2. In addition, section 3.1.3 introduces the NB-PCs.

Moreover, section 3.2 explains the (non-binary) successive cancellation decoding. The successive cancellation can be combined with the list decoding to enhance the decoding performance. The list decoding for non-binary decoder suffers from much higher complexity than its binary counterpart.

Furthermore, section 3.4 presents the state-of-the-art simplifications applied to the successive cancellation decoder to reduce its complexity. The simplified successive cancellation decoder is presented in section 3.4.1. The simplified successive cancellation decoder simplifies the global decoder by eliminating some nodes having deterministic outputs due to the known frozen channels. However, the internal processing of the nodes is complex. Therefore, the min-sum algorithm has been applied to the nodes processing within the successive cancellation decoder as presented in section 3.4.2. The min-sum successive cancellation decoder reduces the complexity by withdrawing the multiplication operations and replacing them with additions. But still, the complexity is high and dominated in the CNs. The complexity of a min-sum CN is of order $\mathcal{O}(q^2)$, therefore, the decoder suffers from high complexity at high field orders $q > 32$. The min-sum CN is simplified using a partial truncation scheme as presented in section 3.4.3. In the partial min-sum processing, one of the two inputs is truncated from q down to n_0 yielding a complexity of order $\mathcal{O}(q \times n_0)$. In addition, the authors proved that under the CCSK modulation, the kernel coefficients γ can be unitary coefficients (set to 1) with no degradation in decoding performance. Nevertheless, in section 3.4.4, the extended min-sum (EMS) has been adopted as a CN processing to reduce the complexity of the CN to an order of $\mathcal{O}(n_m\sqrt{n_m})$ with n_m being the size of the propagated messages.

Besides, in section 3.5, a brief performance and complexity comparison between the NB-LDPC and NB-PC decoder is illustrated and mentioned. The simulation results shows an outperformance of NB-PC over NB-LDPC codes over CCSK modulation and AWGN channel. Also, the complexity comparison that is based on the total check node processing units also indicates that the NB-PC decoder is simpler than that of the NB-LDPC.

Section 3.6 discusses the first proposed work on successive cancellation decoding, called the Asymmetrical Extended Min-Sum Successive Cancellation (SC-AEMS) decoder. An AEMS CN has two asymmetrical inputs and a reduced bubbles computation. The assignment of the asymmetrical inputs is based on the assessment of the relative reliability of the input vectors. The input with the highest reliability is found to contribute less (to any

output) than the input with the least reliability. Therefore, the two asymmetrical sizes are assigned, one for the least reliable input denoted as n_L, and another for the most reliable input, denoted as n_H with $n_L > n_H$. The internal processing of the asymmetrical CN is based on the L-bubble approach presented in section 2.3.2. The proposed SC-AEMS decoder performs well with CCSK modulation. However, the SC-AEMS decoder experiences a significant degradation in performance (0.3 dB) when using BPSK modulation. Thus, a more dynamic approach is proposed, called polarization-aware polar decoding.

The Polarization-Aware Successive Cancellation (SC-PA) decoder tackled in section 3.7 is the second proposed work accomplished during this PhD on non-binary polar decoder. The SC-PA decoder considers the polarization level of the CNs to simplify the internal computation performed. To do so, the notion of node clusters is defined and justified in section 3.7.1. In short, a cluster is a group of kernels that have CNs with similar levels of polarization. Then, the statistical analysis is demonstrated in section 3.7.2. A bubble pattern matrix is defined to include the elements that are used at the output of the CN. In addition, the contribution rate matrix is also defined as the accumulated bubble pattern matrices of all CNs within the same cluster. Once the contribution rate matrices are obtained for all clusters, the pruning process is launched. The bubble pruning process is discussed in section 3.7.3, which prunes the computed region of the clusters based on a predefined threshold. If any bubble has a contribution rate below the defined threshold, it is pruned and omitted from being processed. Thus, a small fraction of bubbles are preserved since their contribution rate to an output is higher than the threshold. This allows for reducing the internal processing of all CNs with the clusters, and also, reducing the size of the input messages in the following layers. The configuration of the proposed SC-PA decoder is provided comprehensively in section 3.7.4. Lastly, the complexity analysis is addressed in section 3.7.5. The performance of the SC-PA decoder is simulated over a set of code lengths and rates over CCSK modulation. It achieves a similar performance to the SC-EMS decoder with a degradation of around 0.2 dB compared to the min-sum SC decoder. The 0.2 dB degradation in performance allowed for achieving a reduction in the total computed field additions and real additions by around 50% and 90% compared to the SC-EMS decoder. Furthermore, the SC-PA decoder has been designed and simulated over BPSK modulation with similar performance and computation reduction. However, the required size of the messages to have a degradation that is less than 0.2 dB is 25 candidates for the SC-EMS decoder and 32 candidates for the SC-PA. The SC-PA requires 32 candidates at the first layer only, at the following layers the sizes are even less than

the SC-EMS decoder.

To sum up, the complexity analysis focuses on the arithmetic operations performed at the CN level. However, the implementation of the proposed decoder and the state-of-the-art decoder should be done to achieve an accurate complexity comparison. Nevertheless, a deep study of the performance/complexity trade-off should be performed to compare the NB-PCs and their binary counterparts over non-binary modulations.

Conclusion and Perspectives

This thesis investigates and proposes new techniques to optimize the decoding algorithms of NB-LDPC codes and NB-PCs. Algorithmic optimization is essential because hardware optimization is insufficient alone to meet the current and future latency constraints of communication systems.

During this PhD, state-of-the-art decoders for NB-LDPC codes, NB-TCs, and NB-PCs were implemented. However, no significant contributions were made to the optimization of NB-TC decoders. As a result, the manuscript doesn't include the study of the non-binary turbo codes.

The BRD algorithm is the first algorithmic optimization made for NB-LDPC decoders. The BRD algorithm is designed to reduce the size of the messages propagated between the variable and check nodes and the number of sorting operations required within the decoding process. The BRD algorithm does this by generating reliability beliefs for the requested symbols asked by the variable nodes.

The BRD algorithm has been combined with various check node processing algorithms, such as the trellis extended min-sum, forward-backward, and syndrome-based check node algorithms. The simulation results of these decoders have shown that they achieve similar performance to the extended min-sum decoder over GF(64) and GF(256), at a lower complexity. Additionally, the forward-backward BRD check node (with a degree $d_c = 12$) algorithm has been implemented and synthesized using a Quartus Cyclone IV FPGA. The synthesis results showed that the forward-backward BRD algorithm reduces the memory allocation by 60% and the logical elements by 15% compared to the forward-backward extended min-sum check node algorithm.

The non-binary polar codes have been also studied and optimized during this PhD. A non-binary polar code of code length $N = 2^n$ consists of n layers and $N/2$ kernels in each layer. During this Ph.D., two optimization approaches were proposed to reduce the complexity of the non-binary decoder. The first is the asymmetrical extended min-sum check node, and the second is the polarization-aware decoding.

The AEMS check node is an optimized check node with asymmetrical processing of the input messages. The internal (asymmetrical) processing is simplified by using the

L-bubble sorter that was initially proposed for the NB-LDPC decoders. The simulation results are obtained for GF(64) codes at different lengths and rates with cyclic-code shift keying modulation using successive cancellation decoding. The performance degradation is around 0.15 dB compared to the successive cancellation min-sum decoding. However, the performance degradation of the proposed approach increases to 0.3 dB when using the binary phase shift keying. This is due to the L-bubble computed region that doesn't fit the potential region of the check nodes. Therefore, an alternative approach called polarization-aware decoding has been proposed.

A polarization-aware decoder is a highly customized decoder that considers the polarization level of the check nodes to generate the output size and potential region. To do so, a kernel cluster is defined to include all kernels with check nodes of similar polarization levels. Each cluster has a fixed output size and specific computed region of the check nodes. The estimation of the potential region is performed using a statistical analysis and a pruning process. In the pruning process, the rare elements that contribute to an output are excluded if they contribute less than the predefined threshold. The pruning process leads to a huge reduction of the total field and real additions performed within the polar decoder as well as the size of the propagating messages. The simulation results over cyclic-code shift keying on GF(64) are obtained for different code lengths and rates. A 0.2 dB performance degradation allows for reducing the number of field and real additions by around 50% and 90% compared to the successive cancellation extended min-sum decoder. The polarization-aware decoder has been simulated over BPSK modulation. The simulation results showed that the 0.2 dB degradation allows for reducing the total arithmetic field and real addition operations by around 60% in terms of GF additions and 70% in terms of LLR additions.

Moreover, the performance of NB-PC decoders is compared to that of binary polar decoders. The comparison includes the successive cancellation decoding for the non-binary and the successive cancellation list decoding for the binary decoder with a list size of 32. Since the encoded symbols are CCSK modulated in NB-PC and the binary PC is BPSK modulated, the comparison is held in two approaches. The first approach is to maintain a coding rate of the binary polar code similar to the effective coding rate, and the second approach is to assume the same coding rate for the NB-PC and the binary PC in addition to repetition codes that compensate for the rate difference of the BPSK and the CCSK modulations. At a low coding rate, the successive cancellation decoding of the NB-PC outperforms the successive cancellation list using the two approaches. However, at high

coding rates, the successive cancellation list decoding of the binary PC outperforms the successive cancellation of NB-PC using the first approach. This is expected since the decoding effect of the successive cancellation is more significant than the spreading effect of the cyclic-code shift keying modulation.

Perspective and Future Work

The work through this PhD research has not only provided answers but has also ignited the flame of curiosity, leading to a host of exciting questions for the future. Initially, the proposed decoders were put to the test using floating-point simulations. However, the real thrill comes from exploring the quantization effect to understand how these decoders (both NB-LDPC and NB-PC) behave in more realistic scenarios.

Moreover, while the initial estimation of complexity reduction is motivating, they remain somewhat abstract, focusing on arithmetic operations only. However, What is truly interesting is to implement the proposed decoder alongside the state-of-the-art decoders to yield tangible, comprehensive results that consider all aspects of decoding, from scheduling and sorting to memory allocation and variable node processing.

As the research delves deeper into the intricate interplay of complexity and performance, a compelling challenge unfolds. The assessment of the delicate balance between complexity and performance of the non-binary polar and binary polar decoders within non-binary modulation. Specifically, when considering polar codes at low coding rates, where the omission of specific nodes from both binary and non-binary codes streamlines hardware implementation.

Furthermore, concerning the polarization-aware decoding. The proposed optimization strategy has thus far been centered on check node statistics, offering a glimpse into efficiency. However, there's another uncharted region to explore – the variable node statistics. It is believed that further exploration of variable node behavior holds the potential for significant complexity reduction, promising a new and exciting chapter in this research journey.

In the grand finale, the non-binary turbo codes can steal the spotlight with their impressive decoding performance among all other non-binary codes to date. However, this thesis couldn't dive deeper into the depths of the non-binary turbo decoders due to time constraints and the intricate interdependencies among their decoding metrics. Nevertheless, the optimization of non-binary turbo decoders is a compelling avenue for

future research.

BIBLIOGRAPHY

[1] K. Saied, « Quasi-Cyclic Short Packet (QCSP) Transmission for IoT », Theses, Université Bretagne Sud, Mar. 2022. [Online]. Available: https://hal.science/tel-03628626.

[2] S. Lionel Sujay Vailshery. « Number of Internet of Things (IoT) connected devices worldwide from 2019 to 2023, with forecasts from 2022 to 2030 ». (2023), [Online]. Available: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/#main-content.

[3] C. E. Shannon, « A mathematical theory of communication », *The Bell System Technical Journal*, vol. 27, *3*, pp. 379–423, 1948. DOI: 10.1002/j.1538-7305.1948.tb01338.x.

[4] R. Gallager, *Low Density Parity-Check Codes*. Cambridge MA: MIT Press, 1963.

[5] R. Neal, « Near shannon limit performance of low density parity check codes », English, *Electronics Letters*, vol. 32, 1645–1646(1), 18 Aug. 1996, ISSN: 0013-5194. [Online]. Available: https://digital-library.theiet.org/content/journals/10.1049/el_19961141.

[6] C. Berrou, A. Glavieux, and P. Thitimajshima, « Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1 », *in Proceedings of ICC '93 - IEEE International Conference on Communications*, vol. 2, 1993, 1064–1070 vol.2. DOI: 10.1109/ICC.1993.397441.

[7] E. Arikan, « Channel Polarization: A Method for Constructing Capacity-Achieving Codes for Symmetric Binary-Input Memoryless Channels », *IEEE Transactions on Information Theory*, vol. 55, *7*, pp. 3051–3073, 2009. DOI: 10.1109/TIT.2009.2021379.

[8] « The Digital Video Broadcasting Project ». (), [Online]. Available: https://dvb.org.

[9] « The 3rd Generation Partnership Project ». (), [Online]. Available: https://www.3gpp.org/.

[10] G. Durisi, T. Koch, and P. Popovski, « Toward Massive, Ultrareliable, and Low-Latency Wireless Communication With Short Packets », *Proceedings of the IEEE*, vol. 104, *9*, 2016. DOI: 10.1109/JPROC.2016.2537298.

[11] J. Hokfelt, O. Edfors, and T. Maseng, « Turbo codes: correlated extrinsic information and its impact on iterative decoding performance », *in 1999 IEEE 49th Vehicular Technology Conference (Cat. No.99CH36363)*, vol. 3, 1999, 1871–1875 vol.3. DOI: 10.1109/VETEC.1999.778362.

[12] C. Berrou, M. Jezequel, C. Douillard, and S. Kerouedan, « The advantages of non-binary turbo codes », *in Proceedings 2001 IEEE Information Theory Workshop (Cat. No.01EX494)*, 2001, pp. 61–63. DOI: 10.1109/ITW.2001.955136.

[13] S. El Hassani, M.-H. Hamon, and P. Pénard, « A comparison study of binary and non-binary ldpc codes decoding », *in SoftCOM 2010, 18th International Conference on Software, Telecommunications and Computer Networks*, 2010, pp. 355–359.

[14] G. Caire, G. Taricco, and E. Biglieri, « Bit-interleaved coded modulation », *IEEE Transactions on Information Theory*, vol. 44, *3*, pp. 927–946, 1998. DOI: 10.1109/18.669123.

[15] D. Declercq, M. Colas, and G. Gelle, « Regular GF (2^q)-LDPC Modulations for higher order QAM-AWGN channels », 2004.

[16] A. Abdmouleh, E. Boutillon, L. Conde-Canencia, C. Abdel Nour, and C. Douillard, « A new approach to optimise non-binary ldpc codes for coded modulations », *in 2016 9th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, 2016, pp. 295–299. DOI: 10.1109/ISTC.2016.7593124.

[17] A. Abdmouleh, E. Boutillon, L. Conde-Canencia, C. A. Nour, and C. Douillard, « On signal space diversity for non binary coded modulation schemes », *in 2016 23rd International Conference on Telecommunications (ICT)*, 2016, pp. 1–5. DOI: 10.1109/ICT.2016.7500494.

[18] T. C. C. for Space Data Systems. « EXPERIMENTAL SPECIFICATION CCSDS 231.1-O-1 ». (2015), [Online]. Available: https://public.ccsds.org/Pubs/231x1o1s.pdf.

[19] C. S. N. Office. « BeiDou Navigation Satellite System Signal In Space Interface Control Document ». (2019), [Online]. Available: http://www.beidou.gov.cn/xt/gfxz/201912/P020230516558050038035.pdf.

[20] O. Abassi, L. Conde-Canencia, M. Mansour, and E. Boutillon, « Non-binary low-density parity-check coded cyclic code-shift keying », *in 2013 IEEE Wireless Communications and Networking Conference (WCNC)*, 2013, pp. 3890–3894. DOI: 10.1109/WCNC.2013.6555196.

[21] Y. Polyanskiy, H. V. Poor, and S. Verdu, « Channel coding rate in the finite block-length regime », *IEEE Transactions on Information Theory*, vol. 56, *5*, pp. 2307–2359, 2010. DOI: 10.1109/TIT.2010.2043769.

[22] G. Dillard, M. Reuter, J. Zeiddler, and B. Zeidler, « Cyclic Code-Shift Keying: A Low Probability of Intercept Communication Technique », *IEEE Transactions on Aerospace and Electronic Systems*, 2003.

[23] A.-C. Wong and V. Leung, « Code-phase-shift keying: a power and bandwidth efficient spread spectrum signaling technique for wireless local area network applications », *in CCECE '97. Canadian Conference on Electrical and Computer Engineering. Engineering Innovation: Voyage of Discovery. Conference Proceedings*, vol. 2, 1997, 478–481 vol.2. DOI: 10.1109/CCECE.1997.608262.

[24] K. Saied. « Quasi Cyclic Small Packet ». (2019), [Online]. Available: https://qcsp.univ-ubs.fr.

[25] D. MacKay, « Good error-correcting codes based on very sparse matrices », *in Proceedings of IEEE International Symposium on Information Theory*, 1997, pp. 113–. DOI: 10.1109/ISIT.1997.613028.

[26] M. Davey and D. Mackay, « Low Density Parity Check Codes over GF(q) », *IEEE Communications Letters*, vol. 2, pp. 165–167, 1998.

[27] R. Tanner, « A recursive approach to low complexity codes », *IEEE Transactions on Information Theory*, vol. 27, *5*, pp. 533–547, 1981. DOI: 10.1109/TIT.1981.1056404.

[28] C. Poulliat, M. Fossorier, and D. Declercq, « Design of regular (2,d/sub c/)-ldpc codes over gf(q) using their binary images », *IEEE Transactions on Communications*, vol. 56, *10*, pp. 1626–1635, 2008. DOI: 10.1109/TCOMM.2008.060527.

[29] A. Bennatan and D. Burshtein, « On the application of ldpc codes to arbitrary discrete-memoryless channels », *IEEE Transactions on Information Theory*, vol. 50, *3*, pp. 417–438, 2004. DOI: 10.1109/TIT.2004.824917.

[30] A. Bennatan and D. Burshtein, « Design and analysis of non-binary ldpc codes for arbitrary discrete-memoryless channels », *IEEE Transactions on Information Theory*, vol. 52, *2*, pp. 549–583, 2006. DOI: 10.1109/TIT.2005.862080.

[31] G. Li, I. Fair, and W. Krzymieri, « Analysis of Non-binary LDPC codes using Gaussian approximation », *in IEEE International Symposium on Information Theory, 2003. Proceedings.*, 2003, pp. 234–234. DOI: 10.1109/ISIT.2003.1228248.

[32] L. Zeng, L. Lan, Y. Y. Tai, S. Song, S. Lin, and K. Abdel-Ghaffar, « Constructions of Non-binary Quasi-Cyclic LDPC Codes: A Finite Field Approach », *IEEE Transactions on Communications*, vol. 56, *4*, pp. 545–554, 2008. DOI: 10.1109/TCOMM.2008.060024.

[33] X.-Y. Hu and E. Eleftheriou, « Binary representation of cycle tanner-graph gf(2/sup b/) codes », *in 2004 IEEE International Conference on Communications (IEEE Cat. No.04CH37577)*, vol. 1, 2004, 528–532 Vol.1. DOI: 10.1109/ICC.2004.1312545.

[34] C. Marchand, H. Harb, T. Gendron, B. Orvoine, and E. Boutillon. « Free NB-LDPC Code Database of the Lab-STICC Laboratory ». (2018), [Online]. Available: http://www-labsticc.univ-ubs.fr/nb_ldpc.

[35] H. Wymeersch, H. Steendam, and M. Moeneclaey, « Log-Domain Decoding of LDPC codes over GF(q) », *in 2004 IEEE International Conference on Communications (IEEE Cat. No.04CH37577)*, vol. 2, 2004, 772–776 Vol.2.

[36] V. Savin, « Min-Max Decoding for Non-binary LDPC Codes », *in 2008 IEEE International Symposium on Information Theory*, 2008, pp. 960–964. DOI: 10.1109/ISIT.2008.4595129.

[37] M. Fossorier, M. Mihaljevic, and H. Imai, « Reduced complexity iterative decoding of low-density parity check codes based on belief propagation », *IEEE Transactions on Communications*, vol. 47, *5*, pp. 673–680, 1999. DOI: 10.1109/26.768759.

[38] V. Savin, « Min-Max decoding for non binary LDPC codes », *in 2008 IEEE International Symposium on Information Theory*, 2008, pp. 960–964.

[39] D. Declercq and M. Fossorier, « Decoding Algorithms for Non-Binary LDPC Codes Over GF(q) », *IEEE Transactions on Communications*, vol. 55, *4*, pp. 633–643, 2007.

[40] A. Voicila, D. Declercq, F. Verdier, M. Fossorier, and P. Urard, « Low-complexity, low-memory ems algorithm for non-binary ldpc codes », *in 2007 IEEE International Conference on Communications*, 2007, pp. 671–676. DOI: 10.1109/ICC.2007.115.

[41] A. Voicila, D. Declercq, F. Verdier, M. Fossorier, and P. Urard, « Low-complexity Decoding for Non-Binary LDPC Codes in High Order Fields », *IEEE Transactions on Communications*, vol. 58, *5*, pp. 1365–1375, 2010.

[42] E. Boutillon and L. Conde-Canencia, « Bubble check: a simplified algorithm for elementary check node processing in Extended Min-Sum non-binary LDPC decoders », *IEE Electronics Letters*, vol. 46, *9*, pp. 633–631, 2010.

[43] E. Boutillon and L. Conde-Canencia, « Simplified check node processing in nonbinary LDPC decoders », *in 2010 6th International Symposium on Turbo Codes Iterative Information Processing*, 2010, pp. 201–205. DOI: 10.1109/ISTC.2010.5613839.

[44] O. Abassi, L. Conde-Canencia, A. Al Ghouwayel, and E. Boutillon, « A Novel Architecture for Elementary-Check-Node Processing in Nonbinary LDPC Decoders », *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 64, *2*, pp. 136–140, 2017. DOI: 10.1109/TCSII.2016.2551550.

[45] P. Schläfer, N. Wehn, M. Alles, T. Lehnigk-Emden, and E. Boutillon, « Syndrome-based Check Node Processing of High Order NB-LDPC Decoders », *in 2015 22nd International Conference on Telecommunications (ICT)*, 2015, pp. 156–162.

[46] H. Harb, C. Marchand, A. A. Ghouwayel, L. Conde-Canencia, and E. Boutillon, « Pre-Sorted Forward-Backward NB-LDPC Check Node Architecture », *in 2016 IEEE International Workshop on Signal Processing Systems (SiPS)*, 2016, pp. 142–147. DOI: 10.1109/SiPS.2016.33.

[47] C. Marchand and E. Boutillon, « NB-LDPC check node with pre-sorted input », *in 2016 9th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, 2016, pp. 196–200. DOI: 10.1109/ISTC.2016.7593104.

[48] C. Marchand, E. Boutillon, H. Harb, L. Conde-Canencia, and A. A. Ghouwayel, « Extended-forward architecture for simplified check node processing in nb-ldpc decoders », *in 2017 IEEE International Workshop on Signal Processing Systems (SiPS)*, 2017, pp. 1–6. DOI: 10.1109/SiPS.2017.8109992.

[49] C. Marchand, E. Boutillon, H. Harb, L. Conde-Canencia, and A. Al Ghouwayel, « Hybrid Check Node Architectures for NB-LDPC Decoders », *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, *2*, pp. 869–880, 2019.

[50] E. Li, K. Gunnam, and D. Declercq, « Trellis-Based Extended Min-Sum for Decoding Non-Binary LDPC Codes », *in 2011 8th International Symposium on Wireless Communication Systems*, 2011, pp. 46–50.

[51] E. Li, F. García-Herrero, D. Declercq, K. Gunnam, J. O. Lacruz, and J. Valls, « Low Latency T-EMS Decoder for Non-Binary LDPC codes », *in 2013 Asilomar Conference on Signals, Systems and Computers*, 2013, pp. 831–835.

[52] E. Li, D. Declercq, and K. Gunnam, « Trellis-Based Extended Min-Sum Algorithm for Non-Binary LDPC Codes and its Hardware Structure », *IEEE Transactions on Communications*, vol. 61, *7*, pp. 2600–2611, 2013.

[53] J. O. Lacruz, F. García-Herrero, J. Valls, and D. Declercq, « One Minimum Only Trellis Decoder for Non-Binary Low-Density Parity-Check Codes », *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, *1*, pp. 177–184, 2015.

[54] H. P. Thi and H. Lee, « Two-Extra-Column Trellis Min–Max Decoder Architecture for Nonbinary LDPC Codes », *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, *5*, pp. 1787–1791, 2017.

[55] J. Tian, S. Song, J. Lin, and Z. Wang, « Efficient T-EMS Based Decoding Algorithms for High-Order LDPC Codes », *IEEE Access*, vol. 7, pp. 50 980–50 992, 2019.

[56] J. Jabour, C. Marchand, and E. Boutillon, « The Best, The Requested, and The Default Non-Binary LDPC Decoding Algorithm », *in 2021 11th International Symposium on Topics in Coding (ISTC)*, 2021, pp. 1–5. DOI: 10.1109/ISTC49272.2021.9594148.

[57] E. Boutillon, J. Jabour, and C. Marchand, « A Method for Decoding a Codeword Encoded using a Non-Binary Code, Corresponding Device and Computer Program », pat. WO2023025960A1, Mar. 1, 2023.

[58] J. Jabour, C. Marchand, and E. Boutillon, « The Best, the Requested, and the Default Elementary Check Node for EMS NB-LDPC Decoder », *in 2023 IEEE Wireless Communications and Networking Conference (WCNC)*, 2023, pp. 1–6. DOI: 10.1109/WCNC55385.2023.10118720.

[59] K. E. Batcher, « Sorting networks and their applications », *in Proceedings of the April 30–May 2, 1968, Spring Joint Computer Conference*, ser. AFIPS '68 (Spring), Atlantic City, New Jersey: Association for Computing Machinery, 1968, pp. 307–314, ISBN: 9781450378970. DOI: 10.1145/1468075.1468121. [Online]. Available: https://doi.org/10.1145/1468075.1468121.

[60] H. Harb, C. Marchand, L. Conde-Canencia, E. Boutillon, and A. C. Al Ghouwayel, « Parallel cn-vn processing for nb-ldpc decoders », *in 2021 IEEE Workshop on Signal Processing Systems (SiPS)*, 2021, pp. 88–93. DOI: 10.1109/SiPS52927.2021.00024.

[61] H. Harb, A. Al Ghouwayel, L. Conde-Canencia, C. Marchand, and E. Boutillon, « Ultra-high-throughput ems nb-ldpc decoder with full-parallel node processing », *Journal of Signal Processing Systems*, vol. 94, Jul. 2022. DOI: 10.1007/s11265-022-01795-y.

[62] I. Tal and A. Vardy, « List Decoding of Polar Codes », *IEEE Transactions on Information Theory*, vol. 61, *5*, pp. 2213–2226, 2015. DOI: 10.1109/TIT.2015.2410251.

[63] K. Niu and K. Chen, « Crc-aided decoding of polar codes », *IEEE Communications Letters*, vol. 16, *10*, pp. 1668–1671, 2012. DOI: 10.1109/LCOMM.2012.090312.121501.

[64] R. Mori and T. Tanaka, « Non-binary Polar Codes using Reed-Solomon Codes and Algebraic Geometry Codes », *in 2010 IEEE Information Theory Workshop*, 2010, pp. 1–5. DOI: 10.1109/CIG.2010.5592755.

[65] S. C. Byun, G. Kim, W. J. Kim, and H.-Y. Song, « A Construction of Non-binary Polar codes with 4 by 4 kernels », *in 2019 Ninth International Workshop on Signal Design and its Applications in Communications (IWSDA)*, 2019, pp. 1–5. DOI: 10.1109/IWSDA46143.2019.8966125.

[66] P. Yuan and F. Steiner, « Construction and Decoding Algorithms for Polar Codes based on 2×2 Non-Binary Kernels », *in 2018 IEEE 10th International Symposium on Turbo Codes & Iterative Information Processing (ISTC)*, 2018, pp. 1–5. DOI: 10.1109/ISTC.2018.8625284.

[67] E. Şaşoğlu, E. Telatar, and E. Arikan, « Polarization for Arbitrary Discrete Memoryless Channels », *in 2009 IEEE Information Theory Workshop*, 2009, pp. 144–148. DOI: 10.1109/ITW.2009.5351487.

[68] M.-C. Chiu, « Non-binary Polar Codes with Channel Symbol Permutations », *in 2014 International Symposium on Information Theory and its Applications*, 2014, pp. 433–437.

[69] E. Arıkan, « Polar codes: A pipelined implementation », *in Proc. 4th ISBC*, vol. 2010, 2010, pp. 11–14.

[70] V. Savin, « Non-Binary Polar Codes for Spread-Spectrum Modulations », *in 2021 11th International Symposium on Topics in Coding (ISTC)*, 2021, pp. 1–5. DOI: 10.1109/ISTC49272.2021.9594166.

[71] A. Alamdar-Yazdi and F. R. Kschischang, « A Simplified Successive-Cancellation Decoder for Polar Codes », *IEEE Communications Letters*, vol. 15, *12*, pp. 1378–1380, 2011. DOI: 10.1109/LCOMM.2011.101811.111480.

[72] F. Cochachin, L. Luzzi, and F. Ghaffari, « Reduced Complexity of a Successive Cancellation Based Decoder for NB-Polar Codes », *in 2021 11th International Symposium on Topics in Coding (ISTC)*, 2021.

[73] F. Cochachin and G. Fakhreddine, « A Lightweight Encoder and Decoder for Non-Binary Polar Codes », *in 2023 22nd International Conference on Wireless Networks (ICWN)*, 2023.

[74] P. Chen, B. Bai, and X. Ma, « Non-Binary Polar Coding with Low Decoding Latency and Complexity », *Journal of Information and Intelligence*, 2022, ISSN: 2949-7159.

[75] J. Jabour, A. C. Al Ghouwayel, and E. Boutillon, « Asymmetrical Extended Min-Sum for Successive Cancellation Decoding of Non-Binary Polar Codes », *in 2023 13th International Symposium on Topics in Coding (ISTC)*, 2023, pp. 1–5.

[76] A. Cassagne, O. Hartmann, M. Léonardon, *et al.*, « AFF3CT: A Fast Forward Error Correction Toolbox! », *Elsevier SoftwareX*, 2019.

COLLEGE MATHSTIC
DOCTORAL BRETAGNE
BRETAGNE OCEANE

Université Bretagne Sud

Titre : Optimisation Algorithmique Des Codes Non-Binaires.

Mot clés : Codes non-binaires, Low-Density Parity-Check Codes, Polar Codes, Turbo Codes.

Résumé : Les codes non binaires sont des codes correcteurs d'erreurs très efficaces pour les petites tailles de message. Ils sont de plus naturellement adapté aux modulations codées et permettent ainsi de s'approcher de la capacité du canal. Toutefois, les décodeurs non binaires construit sur des ensembles de cardinalité élevée (supérieure ou égale à 64) sont encore marginalement utilisés dû à leur grande complexité de décodage. Cette thèse étudie et propose de nouveaux algorithmes de simplification du traitement des noeuds de parités pour deux types de codes correcteurs d'erreur Non-Binaire : les codes Low Density Parity Check non-binaire (NB-LDPC) et les codes polaire Non-Binaire (NB-PC).

Pour les codes NB-LDPC, une nouvelle technique de décodage appelée "algorithme Best-Request-Default" (BRD) a été proposée. Dans un contexte d'échange de messages tronquées, il devient intéressant de relaxer le principe d'information extrinsèque et de rendre interdépendants les messages échangés sur une connection entre un noeud de variable et un noeud de parité. Plus précisement, une partie du traitement du noeud de parité sera dédié pour fournir une réponse aux symboles "demandés" par le noeud de variable (terme "request" de l'algorithme). L'algorithme BRD permet de réduire très significativement la taille des messages échangés, et donc, la complexité du décodeur, et ce, sans dégradation significative de performances.

Le décodage de code NB-PC de grande cardinalité est un domaine encore très peu exploré et l'état de l'art consiste à l'adaptation de l'algorihtme Extended Min-Sum développé pour les codes NB-LDPC aux codes NB-PC. Nous proposons deux optimisations nouvelles : d'une part, une réduction de la complexité du traitement noeud de parité par un traitement asymétrique de ses entrées basés sur un critère de fiabilité relative (algorithme Asymmetrical Extended Min-Sum), d'autre part, une simplification ad-hoc de chaque contrainte de parité en fonction de la polarisation des messages lui arrivant (Polarisation Aware Polar Code Decoder). La combinaison de ces deux optimisations permettent de réduire la complexité d'un facteur 2 par rapport à l'état de l'art, de nouveau sans dégradation notable des performances de décodage.

Title: Algorithmic Optimization of Non-Binary Codes

Keywords: Non-Binary Codes, Low-Density Parity-Check Codes, Polar Codes, Turbo Codes.

Abstract: Non-binary codes are highly efficient error-correcting codes for short message sizes. They are also naturally adapted to coded modulations, and therefore, can be used to approach channel capacity. However, non-binary decoders built on sets of high cardinality (greater than or equal to 64) are still marginally used due to their high decoding complexity. This thesis studies and proposes new algorithms for simplifying the processing of the parity nodes for two types of non-binary error-correcting codes: Non-Binary

Low-Density Parity Check (NB-LDPC) and Non-Binary Polar Codes (NB-PC).

For NB-LDPC codes, a new decoding technique called the Best-Request-Default (BRD) algorithm has been proposed. In the context of truncated message exchange, it becomes interesting to relax the extrinsic information principle and make messages exchanged over a connection between a variable node and a parity node interdependent. More precisely, part of the parity node's processing will be dedicated to responding to the "requested" symbols by the variable node (the algorithm's "request" term). The BRD algorithm makes it possible to significantly reduce the size of the messages exchanged, and therefore the complexity of the decoder, without any significant degradation in performance.

The decoding of high-cardinality NB-PC codes is still a largely unexplored field, and the state-of-the-art consists of adapting the Extended Min-Sum algorithm developed for NB-LDPC codes to NB-PC codes. We propose two new optimizations: firstly, a reduction in the complexity of parity node processing through asymmetrical processing of its inputs based on a relative reliability criterion (Asymmetrical Extended Min-Sum algorithm), and secondly, an ad-hoc simplification of each parity constraint according to the polarization of the messages arriving at it (Polarization-Aware Polar Decoder). The combination of these two optimizations reduces complexity by a factor of 2 compared with the state-of-the-art, again without any noticeable degradation in decoding performance.

Appendix A

Supplementary Material for Non-Binary Polar Codes

This chapter includes the supplementary material required to simulate a non-binary polar code. Section A.1 includes the reliability sequence for different code lengths. In addition, the coefficients used for simulating the polar codes over BPSK modulation are provided in section A.2.

A.1 Reliability Sequences

The reliability sequences for $K = 11, 21, 42,$ and 58 at FER of $\approx 10^{-}3$ and for $N = 64$ on GF(64) are provided in Table A.1.

Table A.1 – Reliability Sequences for $N = 64$ over CCSK Modulation

$K = 11$	0 1 2 4 8 16 32 3 5 6 17 9 24 12 10 18 20 33 34 36 40 14 7 48 13 11 22 21 19 26 25 28 35 37 38 41 15 42 23 27 44 29 49 39 30 50 43 52 45 56 46 31 51 53 54 47 57 58 60 55 59 61 62 63
$K = 21$	0 1 2 4 8 16 32 3 5 6 9 10 12 17 18 20 24 33 7 11 34 13 14 19 36 21 22 25 40 35 26 15 37 48 28 23 38 41 27 42 44 29 49 39 30 50 52 56 43 45 46 31 51 53 54 58 57 60 47 55 59 61 62 63
$K = 42$	0 1 2 4 8 3 5 16 6 9 10 32 12 17 7 18 11 20 33 13 24 34 14 19 36 40 21 48 22 25 35 26 15 28 37 38 41 42 23 44 49 50 56 52 27 29 30 39 31 43 45 46 47 51 53 54 55 57 58 59 60 61 62 63
$K = 58$	0 1 2 4 8 16 32 3 5 6 9 10 12 17 33 18 24 20 34 36 40 48 7 11 14 13 19 21 35 22 25 37 26 28 42 41 38 50 49 44 56 52 15 23 27 29 30 31 39 43 45 46 47 51 53 54 55 57 58 59 60 61 62 63

In addition, the reliability sequences for $K = 43, 85, 171,$ and 213, at FER of $\approx 10^{-}3$

and for $N = 256$ on GF(64) are provided in Table A.2.

Table A.2 – Reliability Sequences for $N = 256$ over CCSK Modulation

$K = 43$	0 1 2 4 8 16 32 3 5 6 20 64 132 34 36 130 10 136 12 68 24 18 144 40 72 80 160 66 48 96 9 17 129 33 65 8 128 5 192 7 11 14 13 67 19 22 25 35 69 97 21 26 56 28 70 49 38 37 98 81 88 41 44 50 73 131 42 52 76 74 82 104 84 100 112 133 134 137 145 138 140 152 146 148 161 162 164 168 15 23 27 30 29 60 46 58 54 71 45 57 53 43 78 92 193 39 51 77 90 86 75 89 85 83 99 102 101 106 108 105 194 135 113 139 141 142 114 147 150 149 176 153 116 154 163 31 47 55 59 196 61 79 62 165 87 91 156 93 166 103 94 107 169 120 109 143 170 110 151 200 115 63 195 155 117 172 177 95 197 157 118 167 208 178 121 198 158 111 171 180 201 122 224 173 202 184 124 119 174 179 204 209 199 159 210 181 123 212 182 225 203 216 185 125 226 175 186 126 205 228 188 206 232 211 240 183 213 214 217 218 227 220 187 127 229 189 190 230 207 241 233 236 234 242 244 248 215 191 219 221 222 223 231 235 237 238 239 243 245 246 247 249 250 251 252 253 254 255
$K = 85$	0 1 2 4 8 16 32 3 5 6 12 10 18 34 66 24 20 64 48 9 40 36 17 128 65 33 68 72 80 96 129 130 132 136 7 144 11 13 14 22 19 21 35 28 26 25 49 56 37 67 50 38 42 41 44 52 69 70 74 73 76 81 82 84 88 97 131 98 133 134 137 15 100 138 27 29 30 23 39 45 43 46 51 53 54 160 71 57 75 58 77 78 83 140 85 60 145 86 104 31 146 47 89 135 55 99 90 148 192 139 59 101 112 79 161 92 102 141 61 152 162 87 105 142 147 62 106 164 91 149 193 108 113 168 150 93 103 194 114 153 163 94 176 143 63 196 154 116 107 165 156 120 200 166 109 208 151 169 110 224 115 195 170 95 172 177 155 117 197 178 118 198 180 157 121 184 158 201 167 122 202 124 204 111 209 210 225 212 171 216 226 228 240 173 232 119 174 199 179 182 181 188 185 186 159 206 203 123 125 126 127 175 183 187 189 190 191 205 207 211 213 214 215 217 218 219 220 221 222 223 227 229 230 231 233 234 235 236 237 238 239 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255

$K = 171$	0 1 2 4 8 16 32 3 5 6 9 10 12 17 18 20 64 33 24 34 7 11 13 14 36 19 21 128 22 65 40 25 66 35 26 15 37 48 68 28 38 129 23 72 41 130 67 42 27 80 132 49 44 69 96 29 39 50 70 136 30 52 73 144 56 74 131 43 160 76 192 81 45 133 82 46 134 84 97 51 71 88 98 137 31 100 138 53 104 140 54 145 112 75 146 57 58 148 60 161 152 77 162 78 164 193 176 83 168 196 47 200 194 208 86 89 135 85 224 99 92 90 101 139 105 55 102 141 59 61 62 63 79 87 91 93 94 95 103 106 107 108 109 110 111 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 142 143 147 149 150 151 153 154 155 156 157 158 159 163 165 166 167 169 170 171 172 173 174 175 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 195 197 198 199 201 202 203 204 205 206 207 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255
$K = 213$	0 1 2 4 8 3 5 16 6 9 10 32 12 17 7 64 18 11 20 128 33 24 13 34 14 36 19 65 40 66 21 48 68 22 129 72 25 35 130 80 26 132 15 96 28 136 37 144 38 160 41 192 67 42 44 49 23 50 69 52 70 56 74 73 29 27 97 76 131 81 133 82 84 30 88 137 98 161 134 100 148 43 138 104 112 145 146 140 39 162 164 193 31 45 46 47 51 53 54 55 57 58 59 60 61 62 63 71 75 77 78 79 83 85 86 87 89 90 91 92 93 94 95 99 101 102 103 105 106 107 108 109 110 111 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 135 139 141 142 143 147 149 150 151 152 153 154 155 156 157 158 159 163 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255

In addition, the reliability sequences for $K = 171, 341, 683$ and 922, at FER of $\approx 10^{-}3$ and for $N = 1024$ on GF(64) are provided in Table A.3.

Table A.3 – Reliability Sequences for $N = 1024$ over CCSK Modulation

$K = 171$	0 1 2 4 8 16 32 3 5 6 9 10 12 17 18 64 20 33 24 34 7 11 13 14 36 19 21 128 22 65 40 25 66 35 26 15 256 37 48 68 28 38 129 23 72 41 512 130 67 42 80 27 132 49 44 69 257 96 29 39 136 50 70 258 30 52 73 144 260 131 513 74 56 160 43 264 76 514 192 81 272 133 45 516 82 288 134 46 520 84 97 320 51 71 88 528 137 98 384 259 544 31 138 100 53 576 140 104 145 54 640 112 261 57 75 146 768 148 262 161 58 152 60 162 77 265 193 515 164 266 78 268 176 194 273 276 83 517 168 196 289 208 135 524 290 545 224 532 522 274 296 89 280 518 200 521 99 47 292 322 85 86 102 328 529 139 108 385 321 304 324 530 536 92 90 101 392 548 336 386 546 105 388 400 448 352 141 577 552 142 416 580 584 55 59 61 62 63 79 87 91 93 94 95 103 106 107 109 110 111 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 143 147 149 150 151 153 154 155 156 157 158 159 163 165 166 167 169 170 171 172 173 174 175 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 195 197 198 199 201 202 203 204 205 206 207 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 263 267 269 270 271 275 277 278 279 281 282 283 284 285 286 287 291 293 294 295 297 298 299 300 301 302 303 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 323 325 326 327 329 330 331 332 333 334 335 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 387 389 390 391 393 394 395 396 397 398 399 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511

cont.	519 523 525 526 527 531 533 534 535 537 538 539 540 541 542 543 547 549 550 551 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 578 579 581 582 583 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000 1001 1002 1003 1004 1005 1006 1007 1008 1009 1010 1011 1012 1013 1014 1015 1016 1017 1018 1019 1020 1021 1022 1023

$K = 341$	0 1 2 4 8 3 5 16 6 9 10 32 12 17 7 64 18 11 20 128 33 24 13 34 256 14 36 512 19 65 40 66 21 48 68 22 129 72 25 130 35 80 26 132 257 96 15 28 136 258 37 144 260 38 513 160 264 514 192 41 272 42 516 67 288 44 520 320 49 384 528 23 50 544 69 52 576 70 640 76 56 768 81 73 74 131 82 133 84 27 97 134 98 104 88 29 100 145 148 140 259 137 138 265 146 39 30 112 261 193 161 208 262 268 290 545 31 43 45 46 47 51 53 54 55 57 58 59 60 61 62 63 71 75 77 78 79 83 85 86 87 89 90 91 92 93 94 95 99 101 102 103 105 106 107 108 109 110 111 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 135 139 141 142 143 147 149 150 151 152 153 154 155 156 157 158 159 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 194 195 196 197 198 199 200 201 202 203 204 205 206 207 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 263 266 267 269 270 271 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 289 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511

cont.	515 517 518 519 521 522 523 524 525 526 527 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000 1001 1002 1003 1004 1005 1006 1007 1008 1009 1010 1011 1012 1013 1014 1015 1016 1017 1018 1019 1020 1021 1022 1023

$K = 683$	0 1 2 4 8 16 3 5 6 9 10 32 12 17 18 7 64 20 11 33 13 24 34 14 19 128 36 21 65 256 40 22 66 25 35 512 48 15 68 26 129 37 72 28 130 38 80 23 132 41 257 96 67 136 42 258 144 44 49 260 513 69 27 160 50 264 514 70 192 52 272 516 29 73 56 288 520 74 30 131 39 320 528 76 384 81 544 82 133 576 640 134 97 84 88 43 768 98 137 104 100 45 259 138 145 140 148 112 46 261 146 265 193 262 208 161 162 51 71 515 152 268 524 54 194 164 290 266 545 224 532 517 60 276 273 176 274 289 168 53 296 521 522 196 58 57 518 200 75 280 292 31 328 322 529 47 55 59 61 62 63 77 78 79 83 85 86 87 89 90 91 92 93 94 95 99 101 102 103 105 106 107 108 109 110 111 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 135 139 141 142 143 147 149 150 151 153 154 155 156 157 158 159 163 165 166 167 169 170 171 172 173 174 175 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 195 197 198 199 201 202 203 204 205 206 207 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 263 267 269 270 271 275 277 278 279 281 282 283 284 285 286 287 291 293 294 295 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 321 323 324 325 326 327 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511

cont.	519 523 525 526 527 530 531 533 534 535 536 537 538 539 540 541 542 543 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000 1001 1002 1003 1004 1005 1006 1007 1008 1009 1010 1011 1012 1013 1014 1015 1016 1017 1018 1019 1020 1021 1022 1023

$K = 922$	0 1 2 4 8 16 32 64 128 256 3 512 5 6 9 10 17 12 18 20 33 65 24 34 129 36 66 48 40 72 68 130 258 132 96 513 257 80 520 272 264 136 160 144 260 288 514 192 516 384 320 528 544 576 7 768 640 14 11 35 13 19 28 37 22 21 69 25 38 26 41 42 67 49 70 44 50 131 73 56 82 134 74 88 133 52 76 145 81 84 259 97 265 104 146 138 137 100 193 98 289 140 274 290 148 545 515 261 532 266 517 112 262 524 276 268 162 161 152 164 208 273 521 296 194 518 176 280 196 200 168 522 529 328 292 530 304 321 536 548 224 546 385 322 324 336 400 448 386 352 392 577 388 552 580 584 560 578 641 15 23 27 29 30 31 39 43 45 46 47 51 53 54 55 57 58 59 60 61 62 63 71 75 77 78 79 83 85 86 87 89 90 91 92 93 94 95 99 101 102 103 105 106 107 108 109 110 111 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 135 139 141 142 143 147 149 150 151 153 154 155 156 157 158 159 163 165 166 167 169 170 171 172 173 174 175 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 195 197 198 199 201 202 203 204 205 206 207 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 263 267 269 270 271 275 277 278 279 281 282 283 284 285 286 287 291 293 294 295 297 298 299 300 301 302 303 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 323 325 326 327 329 330 331 332 333 334 335 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 387 389 390 391 393 394 395 396 397 398 399 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511

cont.	519 523 525 526 527 531 533 534 535 537 538 539 540 541 542 543 547 549 550 551 553 554 555 556 557 558 559 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 579 581 582 583 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000 1001 1002 1003 1004 1005 1006 1007 1008 1009 1010 1011 1012 1013 1014 1015 1016 1017 1018 1019 1020 1021 1022 1023

A.2 Kernels Coefficients for Non-Binary Polar Codes with BPSK Modulation

The coefficient of the polar kernels used for simulating the polar codes with BPSK modulation at a code length of $N = 128$ on GF(64) is given in Table A.4 in the decimal form of the vector representation (with an irreducible polynomial of $\alpha^6 + \alpha + 1$) for the $N/2$ kernel coefficients at each layer l.

Table A.4 – Coefficients for $N = 128$ over BPSK Modulation

$l = 1$	23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23
$l = 2$	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15
$l = 3$	25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21
$l = 4$	34 34 34 34 34 34 34 34 11 11 11 11 11 11 11 11 55 55 55 55 55 55 55 55 29 29 29 29 29 29 29 29 9 9 9 9 9 9 9 9 22 22 22 22 22 22 22 22 43 43 43 43 43 43 43 43 28 28 28 28 28 28 28 28
$l = 5$	38 38 38 38 19 19 19 19 16 16 16 16 43 43 43 43 15 15 15 15 38 38 38 38 17 17 17 17 45 45 45 45 11 11 11 11 46 46 46 46 23 23 23 23 51 51 51 51 19 19 19 19 23 23 23 23 4 4 4 4 28 28 28 28
$l = 6$	35 35 19 19 52 52 43 43 35 35 1 1 40 40 1 1 7 7 42 42 52 52 16 16 29 29 27 27 33 33 60 60 16 16 17 17 53 53 43 43 52 52 2 2 34 34 25 25 45 45 38 38 21 21 27 27 34 34 14 14 63 63 63 63
$l = 7$	25 53 43 55 60 63 29 15 28 15 60 37 46 7 19 43 2 63 36 49 45 45 61 31 61 28 45 39 43 61 60 15 30 33 20 28 49 47 25 53 54 37 55 54 16 5 43 43 60 43 36 42 35 31 29 29 22 51 10 10 6 6 6 6