Towards Achieving Ultra Reliable Low Latency Communications Using Guessing Random Additive Noise Decoding

by

Marwan Jalaleddine

Department of Electrical & Computer Engineering
McGill University
Montréal, Québec, Canada

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Masters in Engineering

July, 15, 2021

"A design cannot be disconnected from the values and assumptions in which it was created, from the ideologies behind it."

-Ruben Pater

Acknowledgements

First and foremost, I would like to thank Professor Warren J. Gross for introducing me to the field of communications. Professor Gross provided me with the support and guidance that made this work possible. Additionally, I am thankful to Thibaud Tonnellier and Syed Mohsin Abbas for their constructive feedback throughout the years. Thibaud helped me delve deeper and gain a better understanding of the field of communications. Syed Mohsin Abbas helped me realize the hardware constraints of implementing decoders in hardware and suggested the idea behind AGRAND. Without Thibaud's and Mohsin's insights in communications and hardware development, this work would not have been successful. I would also like to thank the Electrical Engineering Department and Foundation Pierre Arbour for their financial assistance throughout the years of my Master's degree.

Finally, I am forever grateful for my family's continuous support. Without them, all of this would not have been possible.

Abstract

Ultra-reliable and low latency communications (URLLCs) is one of the key pillars of the 5G communications standard which is used to enable applications ranging from the smart grid to robot control. In the upcoming communication standards, more stringent requirements are being established on the end-to-end latency and reliability of data.

In an effort to build upon the current advancements in URLLC and guessing random additive noise decoding (GRAND), we develop the partitioned GRAND (PGRAND) which uses the quantized reliability information from the channel to generate the most likely test error patterns. We assess the performance of PGRAND on 5G NR CA-polar code, random linear code, and cyclic redundancy check codes. PGRAND provides superior performance to that of ordered reliability bit GRAND at high signal-to-noise ratios (SNRs) by achieving a $0.2dB$ gain at a frame error rate (FER) of 10^{-4} and a 50% reduction in the average queries per frame performance at $\frac{E_b}{N_0} \geq 5.5dB$. Additionally, PGRAND approaches the FER performance of soft maximum likelihood GRAND at high SNRs with less scheduling complexity. This makes PGRAND a desirable candidate as a near maximum likelihood code agnostic decoder for any short, high rate code.

Alternatively, we also develop guessing random additive noise assisted decoding (AGRAND) that can be used alongside any conventional decoder to improve the decoder latency. If AGRAND succeeds to find a version of the codeword that belongs to the codebook, the decoder terminates early, saving latency and power. This decoding scheme can reduce latency by up to 84% at $\frac{E_b}{N_0} = 5.5dB$ when used with successive cancellation list decoding on CA-polar code. As such, AGRAND enables maximum likelihood low latency decoding of CA-polar codes.

Résumé

Les communicantions ultra-fiables et à très faible latence (URLLCs) constituent l'un des piliers essentiels du standard de communication 5G, adopté pour activer des applications allant du réseau électrique intelligent à la commande des robots. Dans les prochaines normes de communication, des exigences plus strictes sont établies en matière de latence et de fiabilité des données de bout en bout.

Dans le but de tirer parti des progrès actuels de l'URLLC et du décodage à bruit additif aléatoire (GRAND), nous développons le GRAND partitionné (PGRAND) qui utilise les informations de fiabilité quantifiées du canal pour générer les modèles d'erreur de test les plus probables. Nous évaluons la performance de PGRAND sur le code CA-polaire 5G NR, le code linéaire aléatoire et les codes de contrôle de redondance cyclique. Le PGRAND offre une performance supérieure à celle du GRAND à des rapports signal sur bruit (SNR) élevés en réalisant un gain de $0.2dB$ à un taux d'erreur sur les trames (FER) de 10^{-4} et une réduction de 50% de la performance des requêtes moyennes par trame à $\frac{E_b}{N_0} \geq 5.5dB$. En outre, PGRAND s'approche de la performance FER du GRAND à vraisemblance maximale douce à des SNR élevés avec une complexité d'ordonnancement moindre. Cela fait de PGRAND un candidat souhaitable pour décodeur agnostique de code à vraisemblance quasi maximale pour tout code court à haut débit.

Alternativement, nous développons aussi le décodage assisté par bruit additif aléatoire devinable (AGRAND) qui peut être utilisé à côté de n'importe quel décodeur conventionnel pour améliorer la latence du décodeur. Si AGRAND réussit à trouver une version du mot de code qui appartient au livre-code, le décodeur se termine prématurément, ce qui permet d'économiser de la latence

et de l’énergie. Ce schéma de décodage peut réduire la latence jusqu’à 84% à $\frac{E_b}{N_0} = 5.5dB$ lorsqu’il est utilisé avec le décodage d’annulation successive utilisant une liste sur code CA-polaire. Ainsi, AGRAND permet un décodage à faible latence et à vraisemblance maximale des codes CA-polaires.

CONTRIBUTIONS

All the simulations and algorithms for successive cancellation decoder and Berlekamp Massey algorithm presented in this thesis were developed using the AFF3CT toolkit [1]. The 5G MATLAB toolkit [2] was used to simulate the SCL decoder used in chapter 4. Additionally, the simulations for GRAND, SRGRAND, SGRAND and ORBGRAND were done using Thibaud Tonnellier's simulation toolkit and verified with the author's C++ scripts. The findings and analyses in Chapters 3 and 4 feature simulations of two decoders, PGRAND and Ad-GRAND, which represent the author's original work. Additionally, the idea for using GRAND as an adjunct decoder was first provided by Syed Mohsin Abbas; however, all the analysis and the simulations in this report represent the author's original work.

Contents

List of Figures

List of Tables

Abbreviations & Symbols

Abbreviations

AGRAND	Guessing Random Additive Noise Assisted Decoding
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
BM	Belekamp-Massey
BPSK	Binary Phase Shift Key
CA	CRC Assisted
CRC	Cyclic Redundancy Check
EA	Euclidean Algorithm
FER	Frame Error Rate
GBM	GRAND Assisted BM Decoding
GRAND	Guessing Random Additive Noise Assisted Decoding
GSC	GRAND Assisted SC Decoding
GSCL	GRAND Assisted SCL Decoding
HDD	Hard Decision Decoding
LPF	Latency per Frame
ML	Maximum Likelihood
ORBGRAND	Ordered Reliability Bit Guessing Random Additive Noise Decoding
PC	Polar Code
PGRAND	Partitioned Guessing Random Additive Noise Decoding
PGZ	Peterson Gorenstein Zierler
QPF	Queries per Frame
RLC	Random Linear Code
SC	Successive Cancellation

SCL	Successive Cancellation List
SGRAND	Soft Maximum Likelihood Decoding using Guessing Random Additive Noise
SNRs	Signal to Noise Ratio
TEP	Test Error Pattern
URLLC	Ultra Reliable Low Latency Communication

Symbols

AB	Abandonment Threshold
α	Log Likelihood Ratio
β	Partial Sum
BER_{in}	Bit-Flip Probability of the channel energy per transmitted information bit
$\mathbb{C}$	Codebook
c_s	Partition Count
d_{min}	Minimum Hamming Distance
$\vec{e}$	Bit Level Test Error Patterns
e_s	Number of Errors in the Partition
e_{symb}	Energy of the Transmitted Symbol
$\frac{E_b}{N_0}$	Energy per Bit to Noise Power Spectral Density Ratio
$\frac{E_s}{N_0}$	Energy per Symbol to Noise Power Spectral Density Ratio
f	Frequency of Pattern
$\mathbf{H}$	Parity Check Matrix
k	Length of Message
l	List Size
l_{SC}	Latency of SC decoding
l_{SCL}	Latency of SCL decoding
l_{smin}	Minimum partition length used

l_{si}	Length of partition i
m	Number of Bits per Symbol
MERR	Mask Error Rate
$\mathcal{N}$	Normal Distribution
n	Length of Codeword
$\vec{e_G}$	Gaussian Noise Signal
n_b	Number of Bit-Flips
p	Parallelization Degree
r	Rate Capability of the Code
s_{lvl}	Depth of Binary Tree
$\vec{s}_{synd}$	Syndrome
$\mathbf{S_{err}}$	Candidate Error Patterns
S_i	Partition Divisions
σ	Standard Deviation
t	Minimum Error Correcting Capability
θ	Hard Moduation Function
$\vec{u}$	Message
$\hat{\vec{u}}$	Estimated Message
$\vec{v}$	Codeword
$\vec{x}$	Modulated Message
$\hat{v}$	Estimated Codeword
w_c	Weight of the Codeword
w_s	Partition Weight
$\vec{y}$	Transmitted Channel Signal
$\vec{z}$	Hard Modulated Recieved Signal

Chapter 1

Introduction

Ultra-reliable and low latency communications (URLLCs) is the main enabler for many mission and safety-critical applications ranging from industrial automation to smart transportation and remote healthcare [5]. Associated with URLLCs are stringent constraints regarding reliability and latency. For instance, in 5G systems, the end-to-end latency for industrial automation ranges from 1 to 100ms, and the frame error rate (FER) should not exceed 10^{-4} [5]. As we head towards 6G communications, a strengthened version of URLLC is suggested to support new scenarios like distributed AI and mobile robotics [6].

To that end, researchers have been trying to achieve ultra-reliable low latency decoding by using near-capacity achieving codes for encoding. For instance, the polar codes that are used in the 5G network achieve channel capacity with successive cancellation decoding with infinite codeword size [7]. Additionally, near-capacity achieving turbo codes [8] and LDPC codes [9] are widely used in current communication standards [10] [3]. Even though the majority of the random linear codes (RLC) have been proven to asymptotically achieve channel capacity [11], RLC are seldom used for channel coding in the state-of-the-art communication standards [3]. The lack of a predefined structure for the RLC code makes all the efforts to design an efficient, low-latency maximum likelihood (ML) RLC decoder difficult, as it would entail designing a code agnostic decoder.

Several decoders have been designed to work with error correcting codes that lack a predefined structure. For example, the ordered statistics decoder (OSD) is a prominent approach which uses the most reliable bits to estimate the most likely codeword [12]. OSD can achieve ML decoding at the cost of an increased computational complexity especially when it comes to reordering the generator matrix in the gaussian elimination stage [12]. Several architectures have been proposed to reduce the latency of OSD by using early stopping conditions or by reducing the patterns considered [13]. Since the complexity of reordering the generator matrix with OSD is estimated to be $\mathcal{O}(n^3)$, with n being the codeword length, the design for an efficient low latency OSD decoders has been proven unwieldy.

Guessing Random Additive Noise Decoding (GRAND) has been recently developed as an ML decoder capable of achieving channel capacity with random codebooks [14]. Rather than trying to deduce the codeword from the structure of the channel code, GRAND leverages the reliability information obtained from the channel output to guess the codeword specific additive noise [14]. To limit the large number of queries performed by GRAND, an abandonment threshold is incorporated in GRANDAB [14]. A highly parallelized hardware architecture is proposed for GRANDAB in [4].

In the case where soft signal information can be used, Soft GRAND (SGRAND) has been developed as a GRAND variant that uses the real valued channel inputs to achieve ML decoding [15]. SGRAND has been shown to outperform all other GRAND decoders in frame error rate (FER) and average number of queries per frame (AQPF) [15]. It achieves ML decoding at the cost of an increased scheduling complexity in the decoding process.

More recently, ordered reliability bits guessing random additive noise decoding (ORBGRAND) has emerged as a universal decoder capable of matching the FER performance of successive cancellation list decoding (SCL) [16] applied on polar codes. ORBGRAND uses logistic weights as a metric to rank test error patterns in increasing likelihood of occurrence. The use of the logistic weight ordering decreases the scheduling complexity of ORBGRAND compared to SGRAND and allows ORBGRAND to outperform GRANDAB in FER performance [17]. A high throughput hardware implementation for ORBGRAND is described in [18].

1.1 Objectives

The main objective of this work is to tackle the shortcomings of ORBGRAND and SGRAND. We aim to develop an algorithm that has better error-correcting performance than ORBGRAND, while maintaining lower scheduling complexity than SGRAND.

Additionally, we aim to explore the use GRAND alongside conventional decoders to reduce latency and power consumption.

1.2 Summary of Contributions

In this thesis, we develop two guessing random additive noise decoding methods: the partitioned guessing random additive noise decoding (PGRAND) and the guessing random additive noise assisted decoding (AGRAND).

1.2.1 PGRAND

PGRAND is a standalone code agnostic channel decoder that can achieve 0.2 dB gain to ORBGRAND at a target FER of 10^{-5} using 50% less AQPF at $\frac{E_b}{N_0} = 5.5\ dB$. Partitioned GRAND is also highly parallelizable and achieves a lower scheduling complexity compared to SGRAND. The proposed PGRAND algorithm divides the received codewords into several partition and generates test error patterns for each partition. To determine the best flip error patterns, a highly parametrized pattern generator is introduced alongside an abandonment criterion.

1.2.2 AGRAND

Particularly useful for latency sensitive applications, GRAND can be used alongside the conventional decoder to reduce average latency. This construction relies on placing a GRAND stage with a low bit-flip limit before the conventional decoder stage. If AGRAND succeeds, the message is directly sent to the receiver. If AGRAND fails, the codeword is then relayed to the conventional decoder to be decoded. AGRAND thrives when used with high latency maximum-likelihood decoders on capacity achieving codes with a large Hamming distance. For instance, by using AGRAND alongside a successive cancellation decoder, we can achieve a 84% reduction in average latency at $\frac{E_b}{N_0} = 5.5dB$.

1.3 Thesis Organization

The remainder of this thesis is structured as follows: Chapter 2 provides an overview of the preliminaries of channel transmission and error correcting code. The construction of a digital communications system is discussed in detail with a focus on transmitting a linear error correcting code along an additive white Gaussian noise (AWGN) channel. Several error correcting codes are discussed alongside their conventional decoders. Additionally, a literature review of the present code agnostic decoders is conducted.

This is then followed by Chapter 3 where we introduce the PGRAND algorithm and discuss the effect of varying the partition length, partition count and the abandonment threshold on the performance of PGRAND. A detailed comparison is included concerning the performance of PGRAND and other decoders on the cyclic redundancy check aided (CA) polar code (PC) used in 5G New Radio communication (5G NR CA-PC) [3]. Additionally, Chapter 3 discusses the performance of PGRAND on three different codes with the same code rate and codeword length.

Following the discussion of a standalone code agnostic decoder, we introduce a GRAND scheme used to reduce the latency of conventional decoders. A cost benefit analysis is conducted in Chapter 4 where the effect of minimum Hamming distance, latency of conventional decoder and presence of cyclic redundancy check (CRC) bits is analyzed with AGRAND. Finally, Chapter 5 concludes this thesis with a summary of the presented research and an outline of directions for future research extensions.

Chapter 2

Background

This chapter provides a comprehensive review of the preliminaries of channel communications. Specifically, we introduce the central dogma behind any modern digital communication system. We focus on the most common error-correcting codes and provide a literature review on the state-of-the-art code agnostic channel decoders used with error-correcting codes.

In the following sections, we use the following notations. Matrices are denoted by a bold upper-case letter $\mathbf{H}$ while vectors are denoted by lower-case letters with an arrow superscript $\vec{v}$. The i^{th} element of a vector is represented by a lowercase subscript $_i$. Scalars are represented by a lower-case letter. The transpose operator is represented by $^\top$. The number of k-combinations from a given set of n elements is noted by $\binom{n}{k}$. $\mathbb{F}_q$ represents a finite field with q elements and $\oplus$ represents the modulo-2 addition operator.

2.1 Digital Communication Systems

Consider a discrete transmission channel susceptible to random additive white Gaussian noise. A message $\vec{u}$ of size k is encoded to a codeword $\vec{v}$ of size n at the sender end using the generator matrix $\mathbf{G}$. This codeword is then modulated to a physical signal $\vec{x}$ of size n which is transmitted through the noisy channel to the receiver end. Assume that the Gaussian noise $\vec{e_G}$ of size n is added to the codeword as shown in (2.1) and the received codeword on the receiver end is $\vec{y}$ of size n.

$$\vec{y} = \vec{x} + \vec{e_G} \tag{2.1}$$

At the receiver end, $\vec{y}$ is then dispatched to a channel decoder. The channel decoder tries to find the most likely guess of the transmitted message $\vec{\hat{u}}$. As such, fast encoders and decoders are needed to guarantee the success of this code transmission scheme. This section highlights the key aspects of channel encoding, decoding and transmission which are used to realize a modern digital communications system.

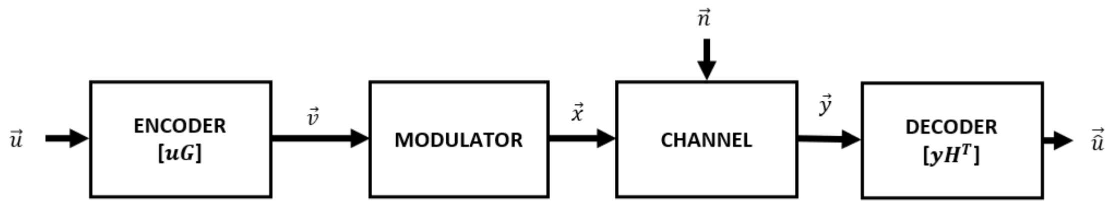

Figure 2.1: Transmission of a message through an additive noise channel

2.1.1 Linear error-correcting Code

Ever since Shannon's theory of communications [19], our understanding of digital communications has drastically changed. Shannon proved that adding a certain level of redundancy in the message allows us to communicate the message practically noise-free to the receiver end [19]. We can add redundancy to $\vec{u}$ by appending extra bits to the message before transmitting it along the channel. These extra bits can be used by the channel decoder to detect and correct errors.

Channel encoding is a linear mapping from $\mathbb{F}_k \longrightarrow \mathbb{F}_n$ in which $n - k$ parity check bits are appended to the message. The parity check bits are used to retrieve the original message from a noisy version of the message. Figure 2.2 shows the partitioning of a codeword into information bits and parity check bits. The choice of the parity check bits determines the error detecting and correcting capability of the code. Hence, we need to find a suitable way to define the parity check bits which, in practice, boils down to finding a code with elements as far as possible from each other.

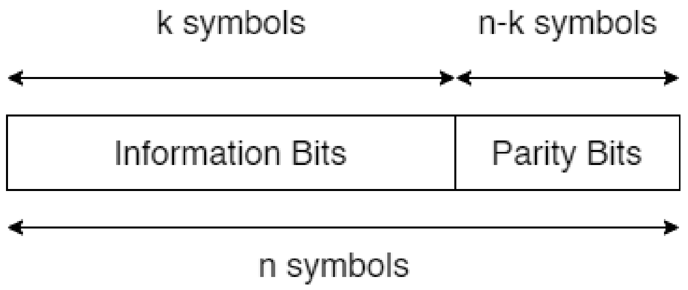

Figure 2.2: The partitioning of a codeword into information and parity check bits

This leads us to define the minimum Hamming distance of the error-correcting code (d_{min}) as the number of bits that differ between the the two closest elements of the codebook ($\mathbb{C}$). The minimum error detection capability of a linear block code is specified in terms of the minimum distance and equals $d_{min} - 1$ [20]. Additionally, the minimum error correction capability (t) equals $\frac{d_{min}-1}{2}$ [20]. Hence, a larger minimum distance increases the error detecting and correcting capability of the code.

This thesis focuses on linear block codes, which constitute a subset of error-correcting codes. Linearity makes it easier to analyze the performance of various error-correcting codes since it allows us to represent all operations in terms of matrix multiplications and matrix additions. A linear block code is characterized by its codeword length, message length, and minimum Hamming distance [n,k,d_{min}]. A code is also categorized based on its code rate defined in (2.2). The code rate (r) represents the proportion of the codeword that carries useful information.

$$r = \frac{k}{n} \tag{2.2}$$

2.1.2 BPSK Modulation

This work focuses on binary transmission along a channel. In other words, we assume that we only have two possible symbols, “ 1” and “0”.

Before transmitting a symbol through the channel, we need to map the binary symbols into physical signals. We primarily use BPSK (Binary Phase Shift Key) modulation in this thesis to map the binary symbols into a 0° or 180° phase shift in the modulator sine wave. In Figure 2.3, the “1” and “0” symbols are modulated into sine waves with the phase angles of 0° and 180° respectively.

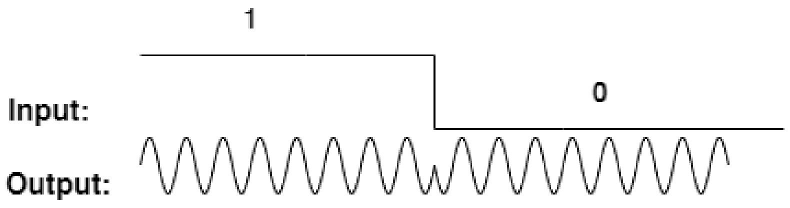

Figure 2.3: BPSK Modulation

Let e_{symb} denote the energy of a transmitted symbol. BPSK modulation employs the mapping in (2.3) [20].

$$\begin{cases} \vec{x}_i = \sqrt{e_{symb}} & if\ \vec{v}_i = 0 \\ \vec{x}_i = -\sqrt{e_{symb}} & if\ \vec{v}_i = 1 \end{cases} \tag{2.3}$$

In most cases, the modulated values are normalized according to $\sqrt{e_{symb}}$ giving us the mapping scheme in (2.4).

$$\begin{cases} \vec{x}_i = 1 & if\ \vec{v}_i = 0 \\ \vec{x}_i = -1 & if\ \vec{v}_i = 1 \end{cases} \tag{2.4}$$

After transmitting a codeword along the channel, a hard decision demodulator transforms the physical continuous signals to discrete digital signals. We define hard decision demodulation as a transformation θ that maps $\vec{y}$ to $\vec{z}$. This transformation is done by thresholding along zero as shown in (2.5).

$$\begin{cases} \vec{z}_i = 0 \quad if\ \vec{y}_i > 0 \\ \vec{z}_i = 1 \quad if\ \vec{y}_i < 0 \end{cases} \tag{2.5}$$

Alternatively, a soft modulator quantizes the physical signals that are passing through the channel into digital signals with a certain level of precision. The real-valued channel values are stored and later used with the soft decoder.

2.1.3 Additive White Gaussian Noise

In an AWGN channel, the noise follows a normal distribution $X \sim \mathcal{N}(0,\, \sigma^2)$ with mean 0 and variance σ^2. The standard deviation (σ) is defined in (2.6) using the code rate and energy per bit to noise power spectral density ratio ($\frac{E_b}{N_0}$).

$$\sigma = \sqrt{\frac{0.5}{r \times \frac{E_b}{N_0}}} \tag{2.6}$$

Additionally, the standard deviation of the noise signal can be defined in terms of the energy per symbol to noise power spectral density ratio ($\frac{E_s}{N_0}$). The relation between $\frac{E_b}{N_0}$ and $\frac{E_s}{N_0}$ is stated in (2.7) where m represents the number of bits per symbol.

$$\left(\frac{E_b}{N_0}\right)_{db} = 10 \log_{10} m + \left(\frac{E_s}{N_0}\right)_{db} - 10 \log_{10} r \tag{2.7}$$

To observe the distribution of physical signals along the channel, we overlap the distribution of the noise over the modulated signals. We obtain the normal distributions $X_1 \sim \mathcal{N}(-1,\, \sigma^2)$ and $X_2 \sim \mathcal{N}(+1,\, \sigma^2)$. For example, the Gaussian distributions centered around -1 and 1 with a standard deviation of 0.8 are presented in Figure 2.4.

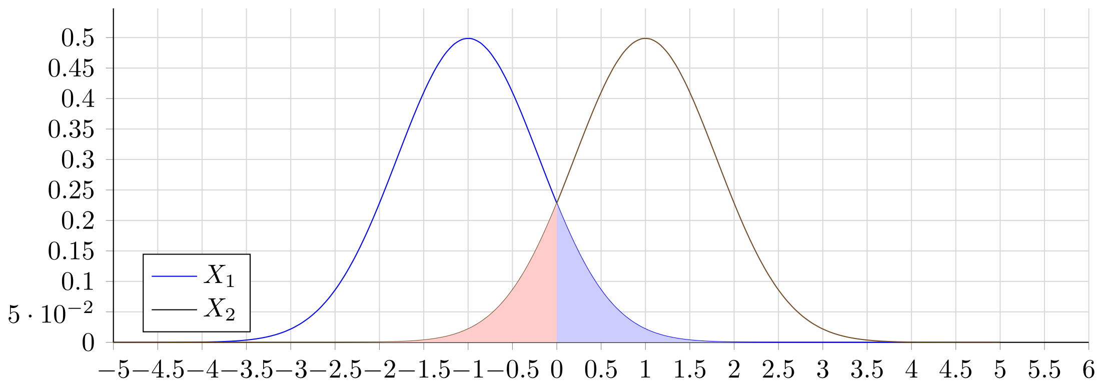

Figure 2.4: The probability density functions for BPSK symbols in an AWGN channel

The size of the tail of the normal distribution determines the bit error probability with hard decision demodulation in an AWGN BPSK channel. For instance, in Figure 2.4, the red shaded area determines the probability of having a transmitted “0” recognized as “1”. Similarly, the shaded blue area represents the probability of having a transmitted ‘1’ recognized as a ‘0’. A larger $\frac{E_b}{N_0}$ reduces the standard deviation of the normal distribution which, in turn, reduces the wrongfully demodulated values at the receiver end.

The probability of erroneous hard demodulation is referred to as the stationary bit flip probability of the channel energy per transmitted information bit (BER_{in}) and is calculated in (2.8).

$$BER_{in} = Q_{func}\left(\sqrt{2 \times r \times \frac{E_b}{N_0}}\right) \tag{2.8}$$

2.2 Error-correcting Codes

In this section, we discuss three commonly used error-correcting codes in great detail. These codes have significant historical, practical and theoretical uses.

2.2.1 Random Linear Code

The performance of random codes is one of the earliest topics explored in information theory in the classic works of Shannon [19], Elias [11], Fano [21] and Gallager [22]. Random linear codes (RLC) were one of the earliest codes to be proven to achieve channel capacity [11]. Ever since then, significant efforts have been conducted to create capacity-achieving random-like codes that can reach the Gilbert-Varshamov asymptotic bound on the minimum distance [23]. Eventhough most RLC codes have been proven to asymptotically obtain the highest achievable code rate with a given minimum distance and large codeword size [11], RLC codes are seldom used for channel coding in the state-of-the-art communication standards [24][10][3]. The lack of a predefined structure for the RLC code renders all efforts to design an efficient, capacity-achieving low-latency RLC decoder difficult, as it would entail designing a code agnostic decoder.

To construct random linear binary codes, k binary values $\in \{0, 1\}$ are randomly placed in the n columns of the generator matrix. We follow this by verifying that the rank of the matrix is equal to k. This check is essential to make sure all the information bits are represented in the encoded code word.

2.2.2 Polar Code

Ever since Shannon proved the existence of capacity achieving codes [19], significant efforts have been made to construct capacity achieving structured codes. Arikan was the first to discover that by using the concept of channel polarization, we can construct a code which achieves symmetric channel capacity when used with successive cancellation decoding [7].

By designing polar codes, Arikan designed an error-correcting code that divides channels into reliable and unreliable channels [7]. Hence, by using the reliable bits to encode message bits and by using unreliable bits to encode redundant bits, symbols can be sent error free across a noisy channel. The redundant bits are usually frozen to a well defined value known by both the encoder and decoder. The reliability of the channels can then be determined by evaluating the Bhattacharyya parameters using quantization methods [25]. A detailed explanation on the construction of polar codes is presented in [25] and a theoretical proof of channel polarization is presented in [7].

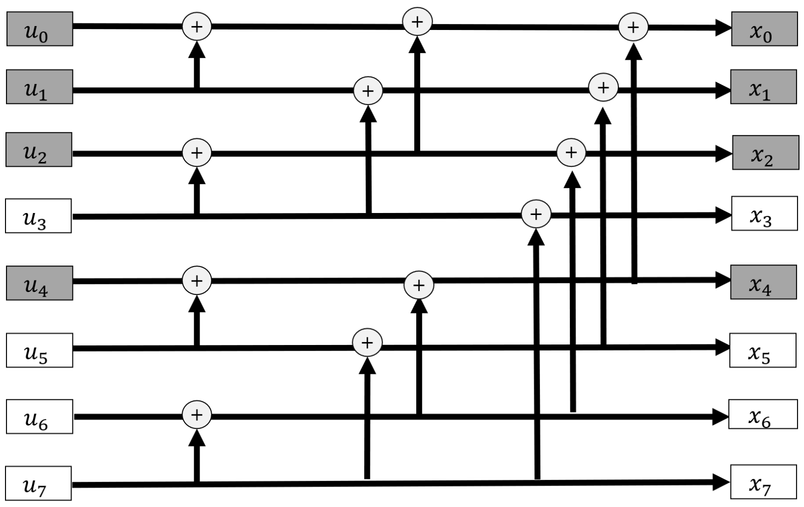

Figure 2.5: Polar encoding architecture where the grey inputs u_0, u_1, u_2 and u_4 represent the frozen bits and the white inputs carry the information bits.

Figure 2.5 shows the polar encoding scheme used to encode polar code [8,4]. Bits u_0, u_1, u_2 and u_3 are frozen since they are less reliable and contain redundant information from other bits. x_0 contains information from all the message bits $x_0 = u_0 \oplus u_1 \oplus u_2 \oplus u_3 \oplus u_4 \oplus u_5 \oplus u_6 \oplus u_7$, but u_0 can only be estimated through x_0. Hence, accurate knowledge of u_0 is harder to achieve knowing x_0. Bit u_7 is the most reliable bit as it can be estimated using all the channel bits. Additionally, x_7 solely contains the information from the u_7 bit. Hence, we can directly estimate u_7 by using the information presented in x_7.

Table 2.1: A subset of the frozen bits used with 5G NR polar code [3]

W	Q	W	Q	W	Q	W	Q	W	Q	W	Q	W	Q
0	0	20	256	40	35	60	516	80	23	100	140	120	53
1	1	21	34	41	258	61	49	81	134	101	30	121	193
2	2	22	24	42	26	62	74	82	384	102	146	122	152
3	4	23	36	43	513	63	272	83	76	103	71	123	77
4	8	24	7	44	80	64	160	84	137	104	262	124	164
5	16	25	129	45	37	65	520	85	82	105	265	125	768
6	32	26	66	46	25	66	288	86	56	106	161	126	268
7	3	27	512	47	22	67	528	87	27	107	576	127	274
8	5	28	11	48	136	68	192	88	97	108	45	128	518
9	64	29	40	49	260	69	544	89	39	109	100	129	54
10	9	30	68	50	264	70	70	90	259	110	640	130	83
11	6	31	130	51	38	71	44	91	84	111	51	131	57
12	17	32	19	52	514	72	131	92	138	112	148	132	521
13	10	33	13	53	96	73	81	93	145	113	46	133	112
14	18	34	48	54	67	74	50	94	261	114	75	134	135
15	128	35	14	55	41	75	73	95	29	115	266	135	78
16	12	36	72	56	144	76	15	96	43	116	273	136	289
17	33	37	257	57	28	77	320	97	98	117	517	137	194
18	65	38	21	58	69	78	133	98	515	118	104	138	85
19	20	39	132	59	42	79	52	99	88	119	162	139	276

The polar codes considered in this thesis are of dimensions [128,105] and [128,116]. The frozen bits are chosen based on a subset of the ordering present in [3] for Polar Code of length 1023. For example, for polar code (PC) [128,116], the

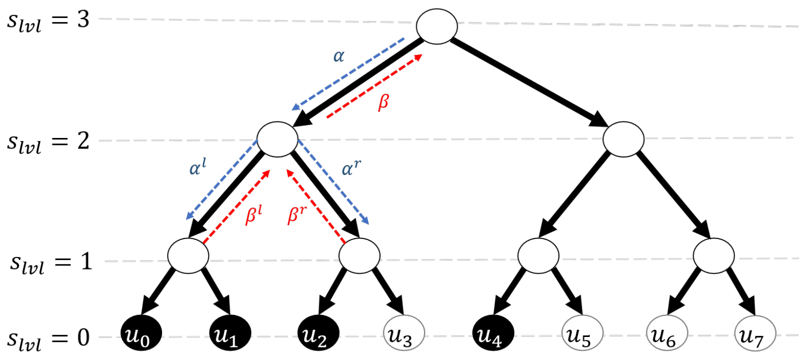

Figure 2.6: Successive Cancellation Decoding of Polar Code

following bits are frozen: $0, 1, 2, 4, 8, 16, 32, 3, 5, 64, 9, 6$. We present in Table 2.1 a subset of the bit ordering used with 5G NR PC. W sorts the bits in ascending order of reliability, and Q represents the bit index before polar encoding.

Recovering the message from the channel output is one of the most computationally intensive tasks. The methods commonly used in practice are successive cancellation (SC) decoding [7] and successive cancellation list (SCL) decoding [16]. In this thesis, PC [128,105] is typically used with SC decoding and PC [128,116] is used with SCL decoding. With SCL decoding, 11 CRC bits are first appended to the message which results in CA-PC [128,105+11]. This CRC aided encoding generates a code of effective $k = 105$ bits and $n - k = 23$ bits.

Successive Cancellation Decoder

The wide interest in using polar codes for channel encoding originates from the fact that long polar codes are capacity achieving with SC decoding [7]. SC decoding can be represented by a binary tree where the search in the left branch is prioritized. An example of this binary tree is presented in Figure 2.6 whereby $S = \log_2(n)$ defines the depth of the tree.

SC decoding uses the log likelihood ratio ($\vec{\alpha}_i$) of the received bits as defined in (2.9) to decode the noisy codeword.

$$\vec{\alpha}_i = \ln(\frac{P(\vec{y}_i|\vec{x}_i = 0)}{P(\vec{y}_i|\vec{x}_i = 1)}) \tag{2.9}$$

Each node receives from its parent a vector of log likelihood ratios whereby the log likelihood ratio associated with the left leaf node ($\vec{\alpha}_i^l$) and the right leaf node ($\vec{\alpha}_i^l$) can be calculated using (2.10) [26].

$$\begin{aligned} \vec{\alpha}_i^l &= sgn(\vec{\alpha}_i)sgn(\vec{\alpha}_{i+2^{s_{lvl}-1}})min(|\vec{\alpha}_i|, |\vec{\alpha}_{i+2^{s_{lvl}-1}}|) \\ \vec{\alpha}_i^r &= \vec{\alpha}_{i+2^{s_{lvl}-1}} + (1 - 2\vec{\beta}_i^l)\vec{\alpha}_i \end{aligned} \tag{2.10}$$

Based on these realizations and the partial sums of the left ($\vec{\beta}_i^l$) and right ($\vec{\beta}_i^r$) nodes, the partial sum of the parent node ($\vec{\beta}_i$) is updated using (2.11) [26].

$$\begin{cases} \vec{\beta}_i = \vec{\beta}_i^l \oplus \vec{\beta}_i^r & \text{if } i < 2^{s_{lvl}-1} \\ \vec{\beta}_i = \vec{\beta}_i^r & \text{otherwise} \end{cases} \tag{2.11}$$

At the leaf node level, a hard decision is made at each of the leaf nodes to determine whether the bit is 1 or 0 based on 2.12 [26].

$$\begin{cases} \vec{\beta}_i = 0 & \text{if } \vec{\alpha}_i > 0 \\ \vec{\beta}_i = 1 & \text{otherwise} \end{cases} \tag{2.12}$$

An optimized hardware implementation for SC decoding has been developed in [27] with a latency l_{sc} specified in (2.13) where p represents the parallelization factor.

$$l_{sc} = 2n + \frac{n}{p}\log_2(\frac{n}{4p}) \tag{2.13}$$

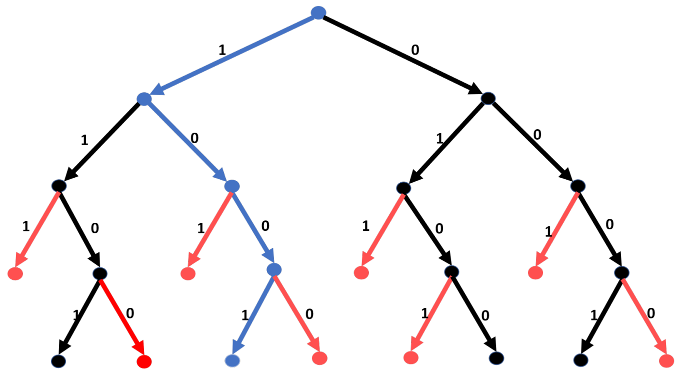

Figure 2.7: Successive Cancellation List Decoding Example

Successive Cancellation List Decoder

To improve the mediocre FER performance of SC decoding on short block codes, successive cancellation list decoding was used as a novel way to decode polar code [28]. One way to look at SCL decoding can be by representing the SCL decoder in a similar fashion to SC decoding, as a binary tree. The binary tree is traversed from the base node to the leaf nodes taking into account the possibility of having a ‘1’ or a ‘0’ in that position [28]. SCL uses a metric to choose the L most probable paths to be explored at a given time. This is done at each level until the entire codeword is decoded [28]. It was shown that by using SCL with a large list size, ML decoding becomes possible [16].

Figure 2.7 shows the progression of an SCL decoder. The red paths are discarded as they are less likely to occur. Only the black and blue paths are pursued. The blue path representing codeword ”1001” is chosen by the SCL decoder as the most probable codeword out of all the considered paths.

Once the tree is completely traversed, the decoder chooses the most likely codeword. To simplify this procedure, CRC bits are appended to the message bits. If the check sequence computation is valid, the route with this check sequence is selected. By using CRC bits, we are significantly improving the decoding performance of the SCL decoder which allows us to reach ML decoding with large list sizes [16].

Several highly parallelizable hardware architectures were created to implement SCL decoding [29] [30] [31]. The latency l_{scl} of the architecture designed by Balatsoukas and Stimming [29] is shown in (2.14).

$$l_{scl} = (2 + r)n + \frac{n}{p} \log \frac{n}{4p} \tag{2.14}$$

2.2.3 Cyclic Redundancy Check Code

Cyclic redundancy check (CRC) codes are very popular cyclic error-correcting codes which have been historically used for error detection. CRC codes are typically used alongside an additional error-correcting code seeing as CRC codes lack a conventional predefined error-correcting decoder.

CRC codes are defined by a generator polynomial which usually determines the error detection capability of the code. The goal of a good designer is to choose a suitable CRC polynomial that maximizes the error detection capability of the code of a given codeword length. The optimal polynomials for certain codeword lengths can be found in [32].

We used two CRC polynomials of different lengths in this report. We append a CRC polynomial of length 11 to the message bits in polar code [128,116] [3]. Hence, the effective number of information bits is reduced to 105 bits. Additionally, we used the CRC polynomial of length 24 to generate CRC code [128,104] [3]. The generator polynomials considered in this work are presented in (2.15).

Table 2.2: CRC bit calculation example for message ”11011000” with polynomial $x^3 + x + 1$

1	1	0	1	1	0	0	0		0	0	0
1	0	1	1								
0	1	1	0	1	0	0	0		0	0	0
	1	0	1	1							
0	0	1	1	0	0	0	0		0	0	0
		1	0	1	1						
0	0	0	1	1	1	0	0		0	0	0
			1	0	1	1					
0	0	0	0	1	0	1	0		0	0	0
				1	0	1	1				
0	0	0	0	0	0	0	1		0	0	0
							1		0	1	1
0	0	0	0	0	0	0	0		0	1	1

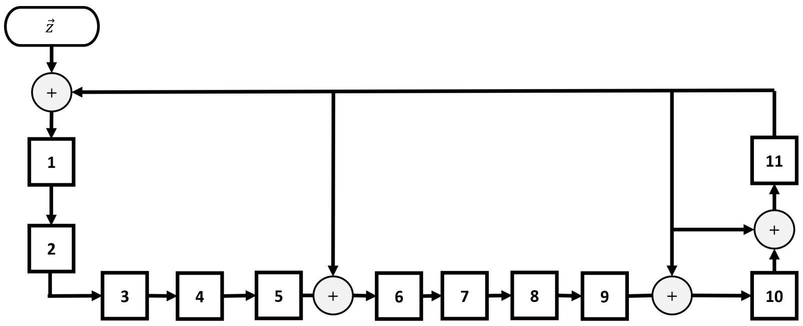

Figure 2.8: CRC Generation Scheme using Linear Feedback Shift Registers (LFSR)s for $g_{CRC_{11}}(x)$ where the boxes represent shift registers and $\oplus$ represents the XOR operation.

Encoding & Decoding CRC Code

To encode a message, it is first appended by zeros and then divided by the polynomial vector. In Table 2.2, we present the division of the message “11011000” by the CRC polynomial $x^3 + x + 1$ to produce the check sequence “011”.

Using the linear feedback shift register architecture described in Figure 2.8, we are able to encode and decode CRC bits with the following polynomial: $x^{11} + x^{10} + x^9 + x^5 + 1$. The message bits are inputted at the data port of the circuit and a structure consisting of shift registers and XOR gates is used to encode the message. The remaining values in the shift registers, after passing the entire message vector, represent the check sequence.

$$
\begin{aligned}
g_{CRC_{11}}(x) = &\quad x^{11} + x^{10} + x^{9} + x^{5} + 1 \\
g_{CRC_{24}}(x) = &\quad x^{24} + x^{23} + x^{21} + x^{20} + x^{17} + x^{15} + x^{13} + x^{12} + x^{8} + x^{4} + x^{2} + x + 1
\end{aligned}
\tag{2.15}
$$

In SCL decoding, CRC bits are used to prune the list of the most probable binary tree paths and reduce them to only one path.

2.2.4 Bose–Chaudhuri–Hocquenghem Code

Bose–Chaudhuri–Hocquenghem codes are algebraic error-correcting cyclic codes developed independently by Bose, Chaudhuri [33] and Hocquenghem [34]. BCH codes are arguably one of the most powerful error-correcting codes with applications ranging from ultra reliable memories [35][36][37][38] to maritime [39] and terrestrial [20] communication systems.

BCH codes are encoded in a similar fashion to CRC codes by using linear feedback shift registers [40]. The process of decoding binary BCH codes starts by computing the syndrome polynomial of the codeword. This is followed by computing the error-locator polynomial through evaluating the key equation using the euclidean algorithm (EA) [41], Peterson–Gorenstein–Zierler (PGZ) [42] or the Berlekamp–Massey algorithm (BM) [43] [40]. After computing the key equation, Chien search [44] is used to find the roots of error locator polynomials. This process is illustrated in Figure 2.9.

Conventional serial BCH decoders use a total of $2n + t + 1$ clock cycles to decode each codeword. Syndrome calculation and Chien search use n clock cycles

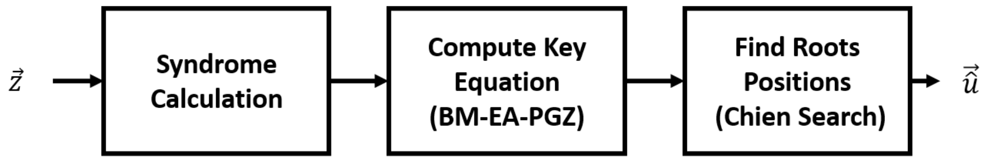
$\vec{z}$
Syndrome Calculation
Compute Key Equation (BM-EA-PGZ)
Find Roots Positions (Chien Search)
$\vec{\hat{u}}$

Figure 2.9: Stages of BCH decoding

each. Additionally, solving the key equation using BM takes $t + 1$ clock cycles [45]. Parallel architectures are commonly used to reduce the latency of BCH decoding by implementing a parallel syndrome check [46], implementing a parallel architecture for Chien search [47], and deriving new methods to solve the key equation [46] [48]. These parallel architectures are suitable for short block codes and are rarely used with large error-correcting codes due to their large area footprint.

2.3 Code Agnostic Decoding

2.3.1 Ordered Statistics Decoding

Several decoders have been designed to work with error-correcting code that lacks a predefined structure. The ordered statistics decoder (OSD) is a prominent approach that uses the most reliable bits to estimate the most likely codeword [12]. OSD can achieve ML decoding at the cost of an increased computational complexity especially when it comes to reordering the generator matrix in the Gaussian elimination stage [13]. Several architectures have been proposed to reduce the latency of OSD by employing early stopping conditions [49] [50] or by reducing the considered patterns [51]. Since the complexity of reordering the generator matrix with OSD is estimated to be $\mathcal{O}(n^3)$, the design for an efficient low latency OSD decoder has been proven to be unwieldy.

Table 2.3: Ordered Statistics Decoding Permutation Example

i	1	2	3	4	5	6	7
$\vec{\alpha}_i$	0.96	-0.70	-1.4	1.0	1.1	1.2	0.29
$\vec{p}_1$	3	6	5	4	1	2	7
Sorted $\vec{\alpha}_i$	-1.4	1.2	1.1	1.0	0.96	-0.70	0.29

Algorithm

The first step after obtaining the codeword from the channel is reordering the codeword based on decreasing reliability. This permutation is referred to as $\vec{p}_1$ and it sorts the received log likelihood ratios α_i in descending order. This can be seen in Table 2.3 where the transmitted codeword is sorted based on decreasing reliability and the permutation $\vec{p}_1$ is generated.

Then, we permute the columns of the generator matrix $\mathbf{G}$ using $\vec{p}_1$ to obtain $\mathbf{G}'$. However, since OSD relies on the k most reliable linearly independent bits and tries to reproduce the remaining bits based on them, we need to make sure the first k columns of $\mathbf{G}'$ are linearly independent. We used the Gaussian elimination function to create a systematic generator matrix $\mathbf{G}''$. This process produces a new permutation $\vec{p}_2$.

$$\mathbf{G}" = \vec{p}_2(\vec{p}_1(\mathbf{G}))$$

Finally, a hard decision decoder is used to map $\vec{y}$ into $\vec{z}$. We select the k most reliable bits for the order-l OSD algorithm. The pseudo-code for an order-l OSD is presented in Algorithm 1.

We initialize the algorithm by adding the most likely noise sequence to the most reliable k bits. Through this process, we generate our candidate vector ($\vec{z}_c$). $\vec{z}_c$ is then multiplied by the modified $\mathbf{G}$" matrix to generate the codeword $\vec{j}$. The Hamming distance between $\vec{j}$ and $\vec{z}_c$ is then calculated and the codeword with the least hamming distance from $\vec{z}_c$ is outputted as the result. The number of

Algorithm 1: OSD-l Algorithm

```
   Input  : $\vec{z}$, $\mathbf{G}$”, $l_{max}$
   Output: $\vec{z_m}$ or $ABANDON$
1  $n_b \leftarrow 0$ ; // Current number of bit-flips
2  $d_m \leftarrow \infty$ ; // Minimum value of the Hamming distance
3  $\vec{z_m} \leftarrow \emptyset$ ; // Current maximum likelihood codeword
4  $\vec{z_c} \leftarrow \emptyset$ ; // Candidate permuted message
5  while $n_b \leq l_{max}$ do
6      $\vec{e} \leftarrow$ most likely noise pattern ;
7      $\vec{z_c} = (\vec{p_2}(\vec{p_1}(\vec{z})) \oplus \vec{e})[1:k]$ ;
8      $\vec{j} = \vec{z_c} \times \mathbf{G}$” ;
9      $d_c = findHammingdistance(\vec{z}, \vec{j})$ ;
10     if $d_c = 0$ then
11         return $\vec{p_1}^{-1}(\vec{p_2}^{-1}(\vec{z_c}))$ ;
12     else if $d_c < d_m$ then
13         $\vec{z_m} \leftarrow \vec{p_1}^{-1}(\vec{p_2}^{-1}(\vec{z_c}))$ ;
14 end
15 return $\vec{z_m}$ ;
```

bit-flips (n_b) is varied to reach a user-defined maximum (l_{max}). It is important to note that a Hamming distance of 0 suggests that the generated codeword is an exact match with the complete codeword received over the channel.

2.3.2 Guessing Random Additive Noise Decoding

Guessing Random Additive Noise Decoding (GRAND) has been recently developed as an ML decoder capable of achieving channel capacity with random codebooks [14]. GRAND tries to guess the additive noise $\vec{e}$ impacting the codeword by leveraging the bit reliability information from the channel. To obtain the most likely codeword guess ($\hat{\vec{v}}$), the possible noise patterns are sorted in decreasing likelihoods of occurrence and are then subtracted from the received codewords. GRAND uses $\vec{\alpha}_i$ to sort the bit locations in terms of reliability. The number of bit-flips is also varied starting from 0 bit-flips to a user-defined maximum.

To determine if $\vec{\hat{v}}$ belongs in the codebook ($\mathbb{C}$), GRAND uses the transpose of the parity check matrix ($\mathbf{H}$) to compute the syndrome ($s_{\vec{synd}}$) of the codeword as shown in (2.16).

$$\vec{s}_{synd} = \vec{\hat{v}} \times \mathbf{H}^T \tag{2.16}$$

$\mathbf{H}$ is the orthogonal complement of $\mathbf{G}$ and therefore any codeword belonging to $\mathbf{G}$ is orthogonal to $\mathbf{H}$. If $\vec{\hat{v}}$ belongs to $\mathbb{C}$, then $\vec{\hat{v}}$ is orthogonal to $\mathbf{H}$ and $\vec{s}_{synd} = \vec{0}$.

Algorithm 2: GRANDAB Algorithm

```
   Input  : $\vec{y}$, $\mathbf{H}$, $lmax$
   Output: $\vec{\hat{v}}$ or $ABANDON$
1  while $n_b \leq l_{max}$ do
2      $\vec{e} \leftarrow$ most likely noise pattern ;
3      $n_b$++ ;
4      $\vec{\hat{v}} = \vec{y} \oplus \vec{e}$ ;
5      if $\vec{\hat{v}} \in \mathbb{C}$ then
6          return $\vec{\hat{v}}$;
7      end
8  end
```

In addition, it might be advantageous to limit the large number of queries performed by GRAND. To that end, an abandonment threshold is incorporated with GRANDAB [14]. The pseudo-code for the GRANDAB algorithm is presented in Algorithm 2.

Section 2.3.3, section 2.3.4, section 2.3.5 and chapter 3 describe alternative ways to generate the noise patterns.

2.3.3 Symbol Reliability Guessing Random Additive Noise Decoding

In the case where signal reliability information can be used, SRGRAND was developed to improve the performance of GRAND. SRGRAND uses the soft channel information to divide the received bits into reliable and non-reliable bit partitions [52]. To that end, SRGRAND introduces a new threshold τ in (2.17) which is a function of the standard deviation of the code-length, the channel noise and the mask error rate (MERR)[53]. $F_{\mathcal{N}}^{-1}$ represents the inverse of the normal distribution function. We define MERR as the probability that a bit is marked as reliable when it is in fact unreliable [53]. Since the MERR defines the resolution of this threshold, a lower MERR improves the FER performance of the code, especially at high SNRs.

$$\tau = \sigma F_{\mathcal{N}}^{-1}((1 - MERR)^{\frac{1}{n}}) - 1 \tag{2.17}$$

The received signals with absolute values between 0 and τ are labeled as unreliable bits. All the values with absolute values greater than τ are labeled as reliable bits. GRAND then flips only the bits labeled as unreliable which reduces the number of TEPs and improves the FER performance.

2.3.4 Soft Maximum Likelihood Decoding using GRAND

SGRAND is another soft decoding GRAND algorithm that uses the real-valued channel values to achieve ML decoding [15]. SGRAND uses the channel output realizations $p(\vec{y}_j|\theta(\vec{y}_j))$ to select the next maximum likelihood error pattern. This can be seen in (2.19) where the error vector ($\vec{e}$) is assigned the most likely error sequence from the candidate error patterns ($\mathbf{S_{err}}$). This step maximizes the probability of correct hard modulation.

$$p(\vec{y}|\theta(\vec{y})-\vec{v}) = \prod_{j=1}^{n} p(\vec{y}_j|\theta(\vec{y}_j))^{1-\vec{e}_{\vec{i}j}} \times (1-p(\vec{y}_j|\theta(\vec{y}_j)))^{\vec{e}_{\vec{i}j}} \tag{2.18}$$

Algorithm 3: SGRAND Algorithm [15]

```
   Input  : $\vec{y}$, $p(\vec{y}_i|\theta(\vec{y}_i))^{1-\vec{e}_{\vec{i}j}}$, $l_{max}$
   Output: $\hat{\vec{v}}$
 1 $n_b \leftarrow 0$ ;
 2 $\vec{i} \leftarrow$ GetSortedIndices($\vec{y}$) ; // Ordered Error Indices
 3 while $n_b \leq l_{max}$ do
 4     $\vec{e} \leftarrow \arg\max_{\vec{v}\in\mathbf{S_{eff}}} p(\vec{y}|\theta(\vec{y})-\vec{v})$ ;
 5     $\mathbf{S_{err}}=\mathbf{S_{err}} \setminus \vec{e}_{\vec{i}j}$;
 6     $n_b$++ ;
 7     $\hat{\vec{v}} = \theta(\vec{y}) \ominus \vec{e}$ ;
 8     if $\hat{\vec{v}} \in \mathbb{C}$ then
 9         return $\hat{\vec{v}}$;
10     else
11         if $\vec{e} = \vec{0}$ then
12             $j \leftarrow 0$;
13         else
14             $j \leftarrow max\{j : \vec{e}_{\vec{i}j} \neq 0\}$;
15         end
16         if $j < n$ then
17             $\vec{e}_{\vec{i}j+1} \leftarrow 1$;
18             $\mathbf{S_{err}} = \mathbf{S_{err}} \bigcup\{\vec{e}\}$;
19             if $j > 0$ then
20                 $\vec{e}_{\vec{i}j} \leftarrow 0$;
21                 $\mathbf{S_{err}} = \mathbf{S_{err}} \bigcup\{\vec{e}\}$;
22     end
23 end
```

$$\vec{e} \leftarrow \arg\max_{\vec{v}\in\mathbf{S_{err}}} p(\vec{y}|\theta(\vec{y})-\vec{v}) \tag{2.19}$$

To calculate $p(\vec{y}|\theta(\vec{y})-\vec{e})$ of each of the candidate vectors, the $p(\vec{y}_j|\theta(\vec{y}_j))$ of

Table 2.4: An example of the execution of SGRAND

n_b	$\vec{e_i}$	$p(\vec{y}\|\theta(\vec{y})-\vec{e})$	j	$\mathbf{S_{err}}$
1	(0,0,0)	0.06	0	{(1,0,0)}
2	(1,0,0)	0.14	1	{(1,1,0),(0,1,0)}
3	(1,1,0)	0.21	2	{(0,1,0),(1,0,1),(1,1,1)}
4	(1,1,1)	0.21	3	{(0,1,0),(1,0,1)}
5	(1,0,1)	0.14	4	{(0,1,0)}
6	(0,1,0)	0.09	5	{(0,1,1),(0,0,1)}
7	(0,1,1)	0.09	6	{(0,0,1)}
8	(0,0,1)	0.06	7	{}

the individual bit locations are used as shown in (2.18). During each iteration, SGRAND chooses the error vector with the largest $p(\vec{y}|\theta(\vec{y})-\vec{e})$ to be used with GRAND decoding. Additionally, during the same iteration, SGRAND adds two error vectors to $\mathbf{S_{err}}$ which results in a net addition of one vector per iteration. The complete pseudo-code for SGRAND is listed in Algorithm 3 [15].

An example of the the error candidates generated by SGRAND for n=3, $p(y_1|\theta(\vec{y}_1)) = 0.3$, $p(\vec{y}_2|\theta(\vec{y}_2)) = 0.4$ and $p(\vec{y}_3|\theta(\vec{y}_3)) = 0.5$ is presented in Table 2.4.

SGRAND has been shown to outperform all other GRAND decoders in terms of frame error rate (FER) and average number of queries per frame (QPF) [15]. It achieves ML decoding at the cost of an increased scheduling complexity in the decoding process.

2.3.5 Ordered Reliability Bits Guessing Random Additive Noise Decoding

More recently, the Ordered Reliability Bits GRAND (ORBGRAND) has emerged as an algorithm capable of matching the FER performance of SCL decoding applied on polar codes [17]. ORBGRAND uses logistic weights (w_L) as a metric

Table 2.5: Order of the Logistic Weight TEPs generated for use in ORBGRAND. Rows indicate patterns, which are ordered top to bottom in decreasing likelihood.

w_L	$\vec{e}_{\vec{i}1}$	$\vec{e}_{\vec{i}2}$	$\vec{e}_{\vec{i}3}$	$\vec{e}_{\vec{i}4}$
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	1	1	0	0
	0	0	1	0
4	1	0	1	0
	0	0	0	1

to rank test error patterns (TEPs) in increasing likelihood of occurrence as shown in (2.20) [17]. $\vec{e}$ represents the permuted error vector, sorted in increasing bit reliability, $\vec{i}$ represents the vector of ordered error indices. The logistic weight of the error vector is calculated as the sum of of ordered error indices with a nonzero value.

$$w_L(\vec{z}) = \Sigma_{j=1}^{n} \vec{i}_j \times \ 1_{(\vec{e}_{\vec{i}_j}=1)} \tag{2.20}$$

The error sequences are generated in increasing logistic weights. For example, Table 2.5 shows the generation of the error sequences for the first 4 logistic weights. Unlike the generation of Hamming weights, which only takes into account the number of bit-flips, the generation of logistic weights also takes into consideration the reliability of the bits being flipped.

The use of the logistic weight ordering decreases the scheduling complexity of ORBGRAND compared to SGRAND and allows ORBGRAND to outperform GRANDAB in FER performance [17].

Chapter 3

Partitioned Guessing Random Additive Noise Decoding

Based on the aforementioned GRAND methods, we have devised a new approach to generate the noise patterns. PGRAND effectively divides the channel output into several reliability partitions to generate partition noise patterns. In this chapter, we discuss the algorithm behind PGRAND, and we evaluate the FER and QPF performance of PGRAND on various types of code.

3.1 Partitioned GRAND algorithm

The algorithm for PGRAND is detailed in Algorithm 4. The algorithm starts by sorting the received bits of $\vec{y}$ based on increasing absolute value of the log likelihood ratio (LLR) vector. Additionally, a hard demodulated value of the received signal ($\theta(\vec{y})$) is generated. This is then followed by dividing the sorted codeword into several partitions and assigning weights ($\vec{w_s}$) to each partition using (3.1). The effects of varying the individual partition lengths ($\vec{l_{s_i}}$) and partition count (c_s) are thoroughly discussed in Section 3.3.1. Test error patterns (TEPs) are then generated at partition level ($\mathbf{P_{TEPs}}$) and at bit level ($\mathbf{B_{TEPs}}$). $\mathbf{P_{TEPs}}$ are uniquely generated based on (3.2) in increasing codeword weight (w_c), and the corresponding $\mathbf{B_{TEPs}}$ are derived by using the lexicographic order. $\mathbf{B_{TEPs}}$ are then tested for codebook membership. A maximum codeword weight (t_w) can

Algorithm 4: PGRAND Algorithm

```
Input  : $\mathbf{H^T}$, $\vec{y}$, $n$, $\vec{l_s}$, $t_w$
Output: $\vec{\hat{v}}$ or $ABANDON$
1  $\vec{ind} \leftarrow SortLLR(\vec{y})$ ;
2  $\theta(\vec{y}) \leftarrow HardDemodulate(\vec{y})$ ;
3  for $w_c = 0 : t_w$ do
4      $\mathbf{P_{TEPs}} \leftarrow GenerateP_{TEPs}(\vec{l_s}, w_c, \vec{ind})$ ;
5      $\mathbf{B_{TEPs}} \leftarrow GenerateB_{TEPs}(\mathbf{P_{TEPs}}, \vec{ind})$ ;
6      for $all\ \vec{e}\ in$ $\mathbf{B_{TEPs}}$ do
7          $\vec{\hat{v}} = \theta(\vec{y}) \oplus \vec{e}$ ;
8          if $\vec{\hat{v}} \cdot \mathbf{H^T} = \vec{0}$ then
9              return $\vec{\hat{v}}$;
10         end
11     end
12 end
```

be set by the user to limit the maximum number of queries done by PGRAND.

3.2 Partition Pattern Generation

3.2.1 Pattern Generation for PGRAND

Our proposed algorithm generates patterns based on increasing codeword weights. At first, the codewords are divided into c_s partitions each of length $\vec{l_{si}}$. Each partition is assigned a partition weight based on (3.1) where $l_{s,min}$ is the minimum partition length used. The partition weights are generated starting from $\vec{w}_{s1} = 1$. Equation 3.1 also takes into account the case where the partitions have different partition lengths ($\vec{l_{si}}$).

$$\vec{w}_{s\ i+1} = \vec{w}_{si} + \left\lfloor \frac{\vec{l_{si}}}{l_{s,min}} \right\rfloor \tag{3.1}$$

After generating all the partition weights, we start generating our $\mathbf{P_{TEPs}}$.

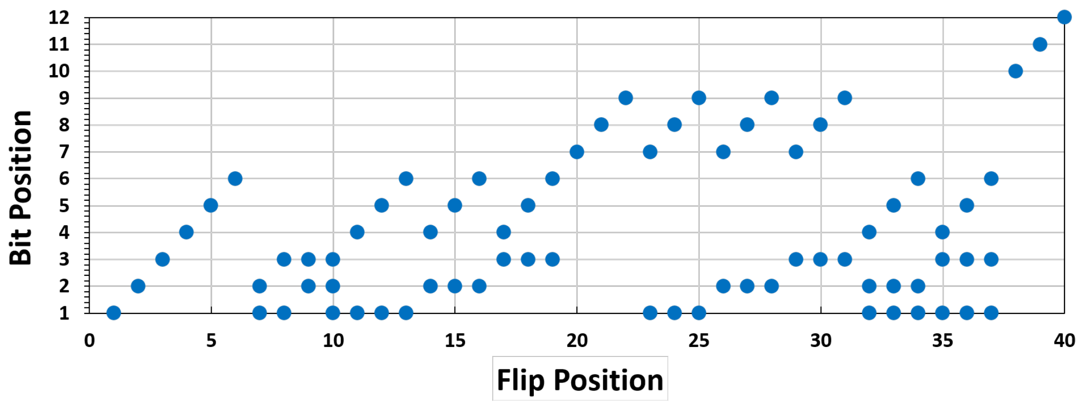

Figure 3.1: Patterns generated by PGRAND for a codeword of length n = 12, $\vec{l_s}$ = [3,3,3,3] and t_w = 4. Columns indicate $\mathbf{B_{TEPs}}$, which are ordered left to right in decreasing likelihood. Dots indicate flipped bit locations.

The codeword weight and the number of errors in the partition ($\vec{e_{si}}$) are used to sort the partition test error patterns in decreasing likelihood of occurrence. This criterion is presented in (3.2).

$$w_c = \sum_{i=1}^{i=c_s} \vec{w_{si}} \times \vec{e_{si}} \tag{3.2}$$

For example, using this pattern generation algorithm, a codeword of size n = 12 bits divided into 4 equal partitions with $t_w = 4$ produces the $\mathbf{P_{TEPs}}$ shown in Table 3.1. The $\mathbf{B_{TEPs}}$ that are generated based on the $\mathbf{P_{TEPs}}$ are presented in Fig. 3.1. For clarity, using $\vec{e_s} = [2,0,0,0]$, the generated $\mathbf{B_{TEPs}}$ are numbered from 7 to 9 in Fig. 3.1.

The test error pattern generation technique used by PGRAND is based on an analysis of the most likely partition errors generated in an AWGN channel. During numerical simulation on an AWGN channel, we collect most frequent error patterns for each partition and compute their cost using (3.3) and store them in ascending order of their cost. Pattern cost is directly proportional to the complexity of the bit-flips defined as $\prod_{i=1}^{i=c_s} \binom{\vec{l_{si}}}{\vec{e_{si}}}$. Pattern cost is also inversely

Table 3.1: Partition TEPs generated by PGRAND with $c_s = 4$ partitions of equal lengths and $t_w = 4$. Rows indicate patterns, which are ordered top to bottom in decreasing likelihood.

w_c	$\vec{e_{s1}}$	$\vec{e_{s2}}$	$\vec{e_{s3}}$	$\vec{e_{s4}}$
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
	2	0	0	0
3	3	0	0	0
	1	1	0	0
	0	0	1	0
4	1	0	1	0
	2	1	0	0
	0	0	0	1

proportional to the frequency (f) of this error pattern to occur in this channel. Hence, pattern cost penalizes the pattern generation algorithm for using more bit-flips and flipping more reliable partitions first.

$$\text{Pattern Cost} = \frac{\prod_{i=1}^{i=c_s} \binom{\vec{l_{si}}}{\vec{e_{si}}}}{f} \tag{3.3}$$

Increasing the number of errors in a partition contributes to the decay of the frame error rate performance of the decoder. This is mainly due to the fact that flipping too many bits results in generations of false positive codewords which degrade the FER performance. Additionally, flipping more reliable partitions first would produce test error patterns with a low likelihood of occurrence.

Table 3.2: Partition division overview for BCH [63,45,7]

	$\vec{l}_{s1}$	$\vec{l}_{s2}$	$\vec{l}_{s3}$	$\vec{l}_{s4}$	$\vec{l}_{s5}$	$\vec{l}_{s6}$	$\vec{l}_{s7}$	$\vec{l}_{s8}$	$\vec{l}_{s9}$
S_1	32	31							
S_2	21	21	21						
S_3	16	16	16	15					
S_4	8	8	16	16	15				
S_5	4	4	8	16	16	15			
S_6	4	4	4	4	4	4	8	16	15
S_7	5	5	5	5	5	5	10	10	13

3.3 Partition Length and Partition Count

3.3.1 Partition Length and Partition Count

We evaluate the performance of PGRAND with varying partition lengths and number of sections. We start by using PGRAND on BCH code $[63, 45, 7]$ with $[n, k, d_{min}]$ where d_{min} is the minimum Hamming distance of the code. For $c_s < 4$ partitions, the codeword is split into c_s partitions of equal size. For c_s larger than 4 partitions, the codeword is first divided into 4 large partitions and then the 4 sections are further divided starting from the least reliable partition and taking into account $l_{s,min}$. The sample partition divisions (S_i) considered in this section are listed in Table 3.2. Additionally, the FER performance of the sample divisions is presented in Fig. 3.2.

For example, in S_4, the codeword is first divided into equal 4 partitions of (16,16,16,15) bits each and then the least reliable 16 bits section is further divided into 2 sections of 8 bits each.

We can see from Fig. 3.2 that having large partitions as in S_1, S_2, S_3 and S_4 leads to a worse FER performance than the sample partition divisions with shorter partitions in S_5, S_6 and S_7. Therefore, it is imperative to reduce the

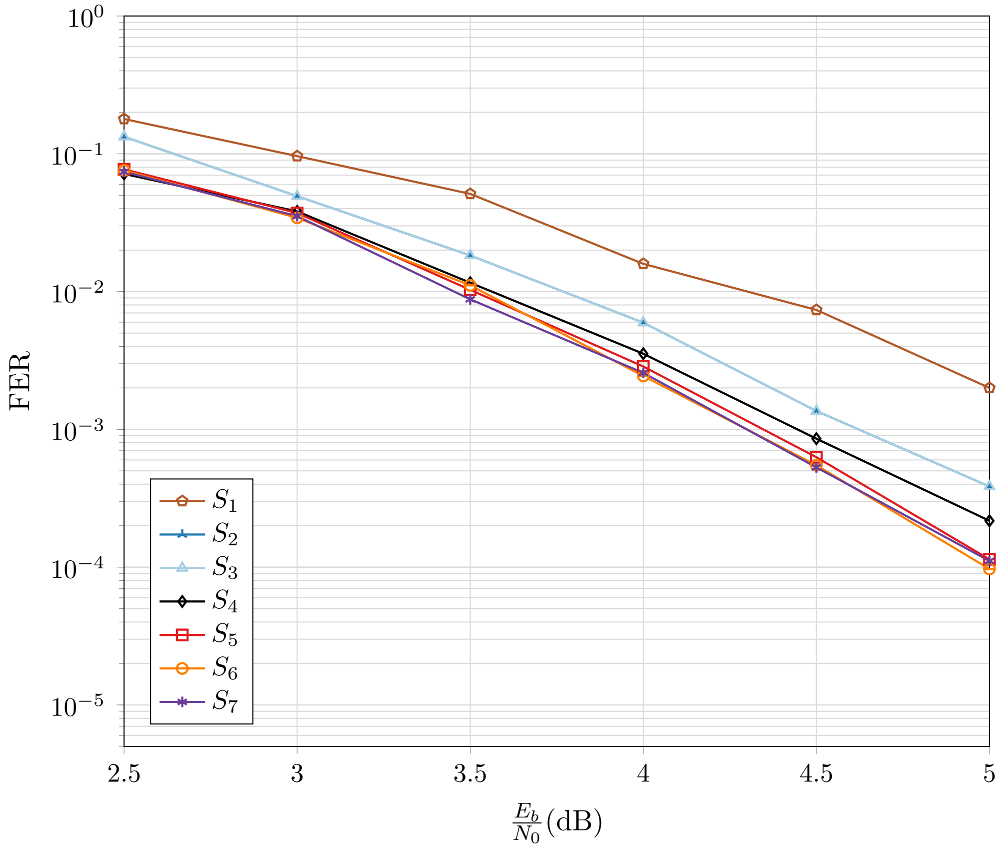

Figure 3.2: FER performance of PGRAND on BCH [63,45,7] with different sample partition divisions

length of less reliable partition to improve the FER performance of PGRAND. Smaller partitions boost the performance of PGRAND since PGRAND is more capable of distinguishing different partitions based on their reliability. Additionally, having many small partitions in S_6 and S_7 increases the complexity of producing partition weights without any additional FER performance boost compared to the FER performance of having a lower partition count in S_5.

3.4 PGRAND with Abandonment

In this section, we analyze the effect of limiting the number of bit-flips by using PGRAND with abandonment (PGRANDAB). Using PGRANDAB, a limit (AB) is imposed on the Hamming weight of generated test error patterns. To explore the effect of limiting the Hamming weight of the TEPs, we simulate PGRANDAB, GRANDAB and PGRAND on CA-Polar code [128,105]. The FER and AQPF results of our simulation are presented in Fig. 3.3. The bit-flip limits used with PGRANDAB are $[3 \rightarrow 8]$ and the bit-flip limit used with GRANDAB is 3 since it is shown in [53] that for $AB > 3$ there is no improvement in FER performance with GRANDAB for 5G NR CA-polar code [128,105].

We can see that increasing the bit-flip limit beyond 8 bit-flips does not improve the FER performance beyond that of PGRAND. Additionally, it can be seen that the AQPF increases with the increase of AB from AB=3 to AB=5 and then starts to decrease following that to reach the minimum AQPF at AB=8. Hence, we recommend the use of a high AB to reduce the AQPF performance and improve the FER performance of PGRANDAB.

In this section, we analyze the effect of limiting the number of bit-flips in PGRAND. PGRAND with abandonment (PGRANDAB) only considers partition TEPs with a number of bit-flips less or equal to the designated limit (AB). To that end, we simulate PGRANDAB, GRANDAB and PGRAND on CA-Polar code [128,105]. The FER and QPF results of our simulation are presented in Figure 3.3 and Figure 3.4 respectively. The bit-flip limits used with PGRANDAB are [3,8] and the bit-flip limit used with GRANDAB is 3 since it is shown in [53] that for $AB > 3$ there is no improvement in FER performance with GRANDAB.

Fig. 3.3 shows that increasing the bit-flip limit beyond 8 bit-flips does not im-

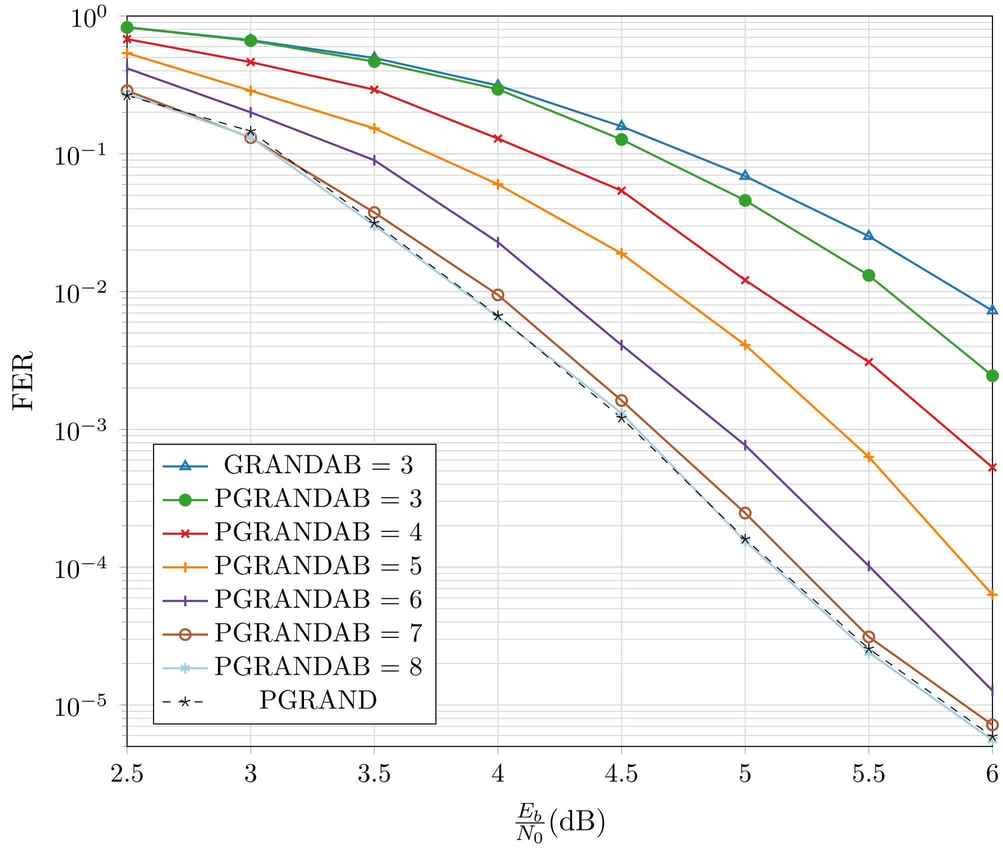

Figure 3.3: The FER Performance using GRANDAB, PGRANDAB and PGRAND on Polar Code [128,105]

prove the FER performance beyond that of PGRAND. This leads us to anticipate that there is an upper limit for AB with PGRAND which is significantly larger than that with GRANDAB. Additionally, we can also see that PGRANDAB outperforms GRANDAB even when using AB=3. This shows that PGRANDAB produces better TEPs to that of GRANDAB.

On the other hand, there does not seem to be a direct correlation between the increase in AB and the QPF of PGRANDAB. It can be seen from Fig. 3.4 that the QPF increases with the increase of AB from AB=3 to AB=5 and then starts to decrease following that to reach the minimum QPF at AB=8. Hence, we recommend the use of a high AB to reduce the QPF performance and improve the FER performance of PGRANDAB.

After simulating BCH Code [127,106,7], BCH Code [63,45,7], CA-Polar Code [128,105,4], CRC code [128,120,3] and Golay Code [24,12,8], we obtained AB ={8,7,8,4,6}.

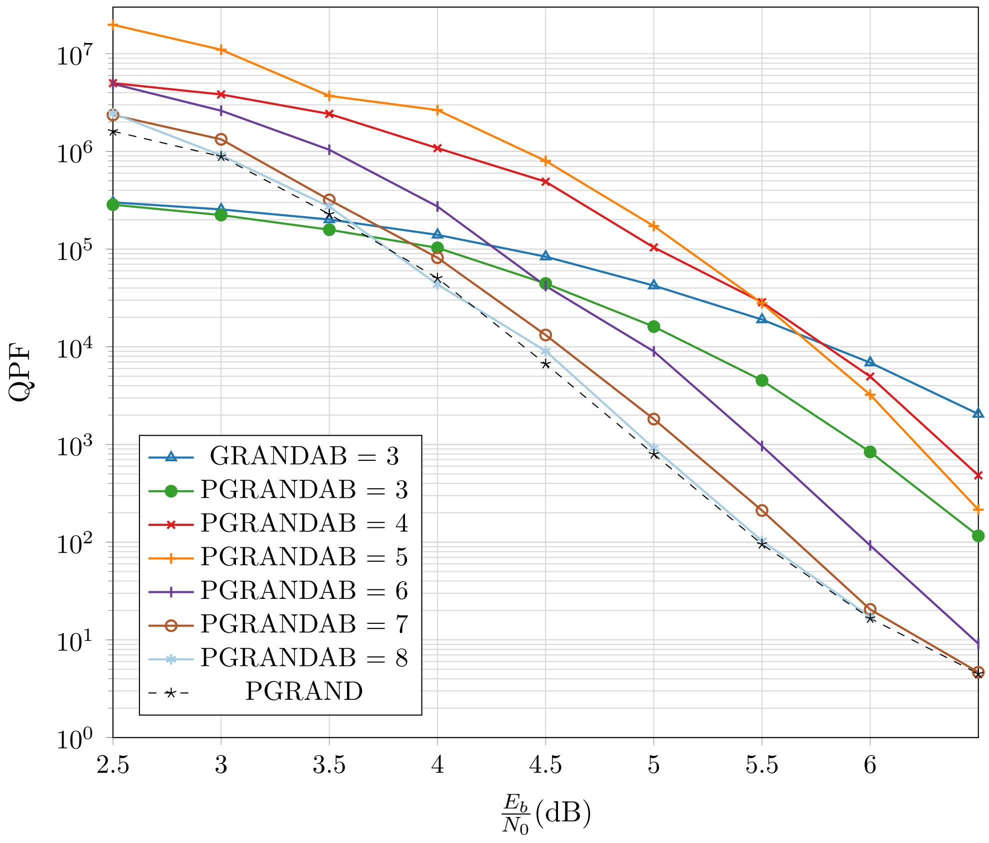

Figure 3.4: The QPF performance of using GRANDAB with AB=3, PGRANDAB with bit-flip limits [3,8] and PGRAND on CA-Polar Code [128,105,7]

3.5 Experimental Results & Analysis

In this section, we discuss the performance of PGRAND on three codes of great interest: PC, CRC and RLC. The simulations are done in an AWGN channel using $\frac{E_b}{N_0}$ as a measure for SNR. The SCL decoder applied on 5G NR Polar code

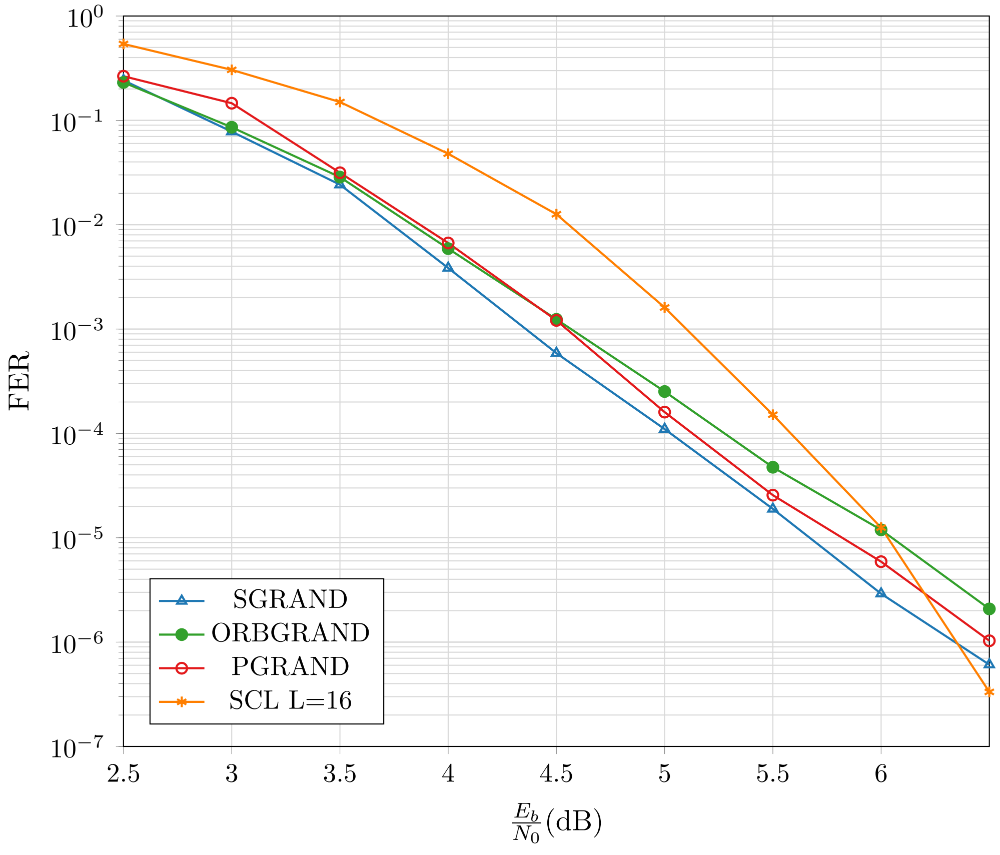

Figure 3.5: Frame Error Rate vs $\frac{E_b}{N_0}$ in an AWGN BPSK channel using SGRAND, PGRAND and SCL on CA-Polar Code [128,105]
with a maximum number of queries= 10^8

uses a list size of 16 and is based on the SCL decoder that is presented in the AFF3CT toolbox [1].

3.5.1 Performance of PGRAND with 5G NR CA-PC

Fig. 3.5 presents the FER performance and AQPF of SGRAND, ORBGRAND, PGRAND and SCL on 5G NR CA-Polar Code [128,105]. At target FER of 10^{-6}, we obtain 0.2 dB gain from PGRAND compared to ORBGRAND and a 0.1 dB loss in performance with PGRAND compared to SGRAND. We can also deduce that PGRAND is capable of producing near ML TEPs. Even though

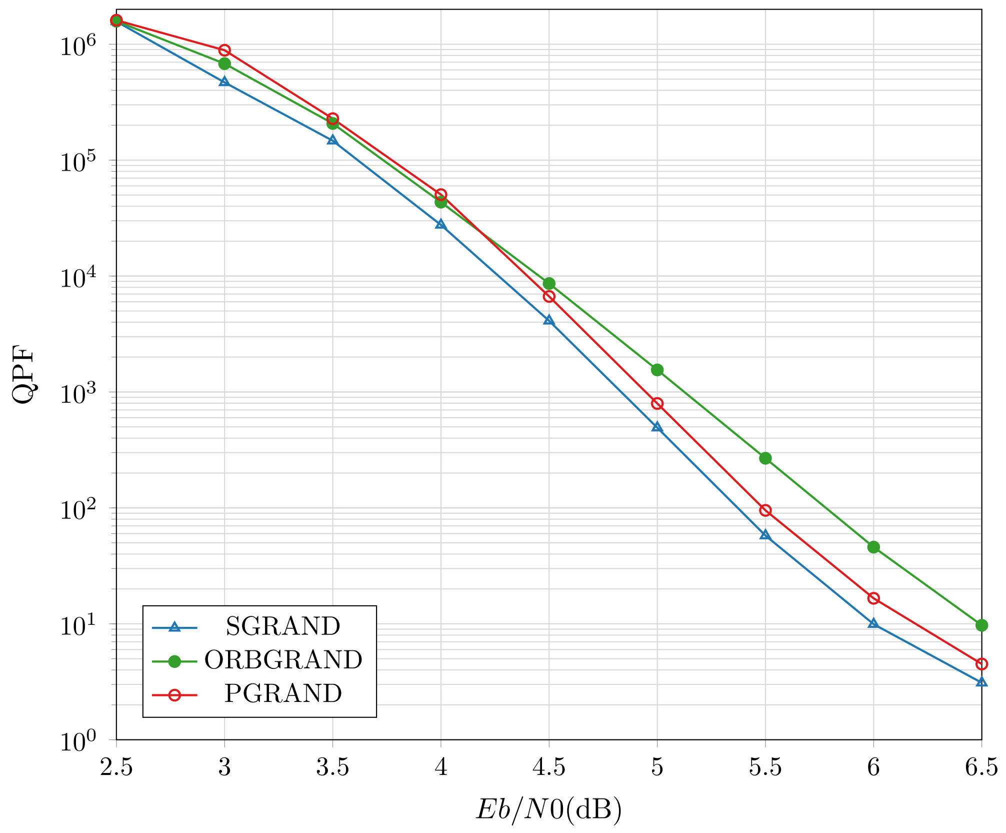

Figure 3.6: QPF vs $\frac{E_b}{N_0}$ in an AWGN BPSK channel using SGRAND, PGRAND, ASCL and SCL on Polar Code [128,105]

SGRAND has a better FER performance than PGRAND and ORBGRAND, every TEP generated by SGRAND depends on the previously generated TEP [15]. This dependency hinders the development of a low latency highly parallelized hardware architecture for SGRAND. We can also see that SCL decoding fails to compete with ORBGRAND, SGRAND and PGRAND at target FERs $> 10^{-5}$. However, with better channel conditions, SCL can outperform the performance of ORBGRAND and PGRAND.

By analysing the AQPF performance of SGRAND, PGRAND and ORBGRAND from Fig. 3.5, we can see that SGRAND uses lower AQPF than ORBGRAND and PGRAND. The average queries per frame metric does not take into consideration the considerable scheduling complexity and the number of comparisons needed with SGRAND. Additionally, we can observe that PGRAND produces less AQPF compared to ORBGRAND for $\frac{E_b}{N_0} > 4.5\ dB$. At $\frac{E_b}{N_0} = 6.5\ dB$, using PGRAND results in a 50% reduction in AQPF compared to using ORBGRAND. Hence, using PGRAND, we can develop a highly parallelized decoder that can exceed the FER and AQPF performance of ORBGRAND at high $\frac{E_b}{N_0}$ and introduce a lower scheduling overhead to that seen with SGRAND. Since there is no dependency in the generated TEPs, PGRAND lends itself to a highly parallelized architecture that dispatches several $\mathbf{P_{TEPs}}$ at the same time to produce $\mathbf{B_{TEPs}}$ with many error patterns. SCL is not included in our AQPF analysis since it is a fixed latency decoder [16].

3.5.2 Performance of PGRAND with Different Codebooks

Finally, we present in Figure 3.7 the FER performance of PGRAND on CA-Polar [128,105], RLC [128,105] and CRC code [128,104]. It is seen that the performance of PGRAND on RLC code is almost indistinguishable from the performance of PGRAND on CA-Polar code and CRC code. This shows that using structured codes provides no performance boost to PGRAND which makes PGRAND a desirable candidate for a code agnostic decoder for any high rate code.

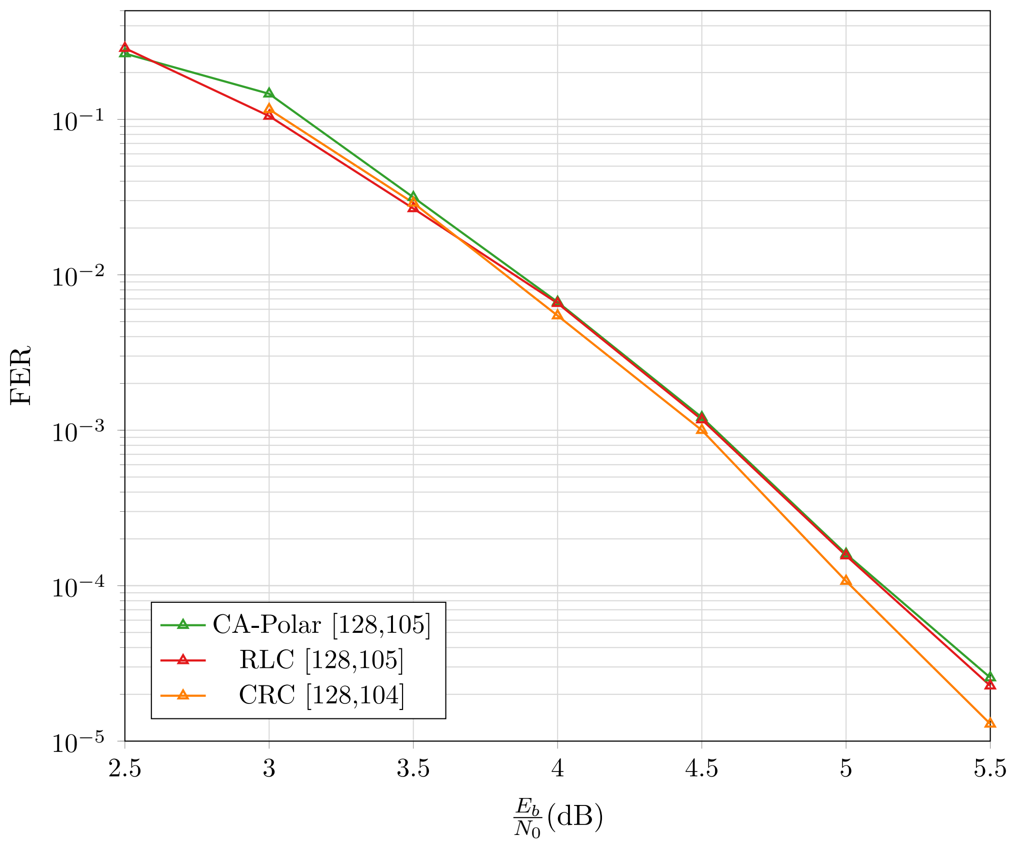

Figure 3.7: Frame Error Rate vs $\frac{E_b}{N_0}$ in an AWGN BPSK channel using PGRAND on CA-Polar [128,105], RLC [128,104] and CRC code [128,105]

Chapter 4

GRAND Assisted Decoding

GRAND can be a desirable decoder for use with codes with no defined structure. However, when we encounter a capacity-achieving error-correcting code, we can also use GRAND assisted decoding (AGRAND) to reduce the latency of the conventional decoder. This construction relies on adding a GRAND stage before the conventional decoder. If GRAND succeeds to decode the codeword, the message bypasses the conventional decoder stage and is directly outputted to the receiver end. In case GRAND fails, the codeword is sent to the conventional decoder with a latency of $l_{conventional}$. This scheme can be seen in Figure 4.1 and is explained in detail in section 4.1.

4.1 Construction of the Decoder Scheme

Unlike current GRAND constructions that exhaustively test all the TEPs in an effort to decode the codeword, AGRAND tests as many error sequences as possible within the least time duration. In an effort to reduce the latency of AGRAND, we limit the possible number of bit-flips (n_b) with the GRAND algorithm to a maximum of 2. This gives us a worst-case latency of $\lfloor n/2 \rfloor + 3$ clock cycles for the AGRAND block. The detailed latency division is presented in Table 4.1 [4].

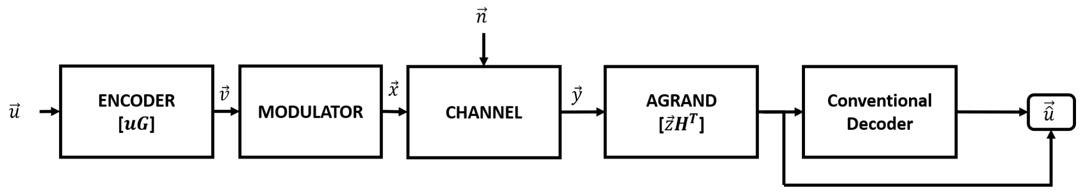

Figure 4.1: The use of AGRAND in a digital communications system

4.1.1 AGRAND Algorithm

The pseudo-code for AGRAND is presented in Algorithm 5. AGRAND starts by generating the noise patterns associated with n_b bit flips (line 3). The generated binary noise vector is XOR-ed to the hard modulated received signal (line 4) and then checked for codebook membership by calculating the syndrome of the codeword (line 5). If the syndrome calculation outputs the zero vector, the decoder terminates and outputs the message (line 6). Else, we need to test for other error patterns. A similar construction to the GRAND decoder discussed in [4] can be considered. In case $n_b = 2$, a dial structure needs to be implemented to compute all the 2 bit-flip TEPs in $\lfloor n/2 \rfloor$ clock cycles. This dial structure is identical to the dial structure discussed in [4].

Algorithm 5: AGRAND Algorithm

```
   Input  : $\vec{y}$, $\mathbf{H}^T$, $n_{b\ max}$
   Output: $\hat{\vec{v}}$ or $\hat{\vec{y}}$
1  $n_b \leftarrow 0$ ;
2  while $n_b \leq n_b\ max$ do
3      $\vec{e} \leftarrow$ GenerateNextNoisePattern $(n_b)$ ;
4      $\hat{\vec{v}} = \theta(\vec{y}) \oplus \vec{e}$ ;
5      if $\hat{\vec{v}} \cdot \mathbf{H}^\mathbf{T} = \vec{0}$ then
6          return $\hat{\vec{v}}$;
7      end
8  end
9  Conventional Decoder $\leftarrow \vec{y}$;
```

If AGRAND fails to decode the codeword, the received signal is relayed to

Table 4.1: The latency of the different stages of AGRAND presented in[4]

	Syndrome Calculation	1 Bit-Flip ($n_b = 1$)	2 Bit-Flips ($n_b = 2$)	Syndrome Check
Latency (clock cycles)	1	1	$\lfloor n/2 \rfloor$	1

the conventional decoder and new delays are introduced in the process. Even though the worst-case latency of the decoder scheme increases to reach $\lfloor n/2 \rfloor + 3 + l_{conventional}$ clock cycles, the average latency per frame (LPF) of the decoder is reduced significantly. Simulations were conducted to generate the experimental fraction of the frames AGRAND is capable of decoding (p_{ex}) with $n_b = \{0, 1, 2\}$.

4.1.2 Percentage of Codewords Decoded by AGRAND

By using (2.8) and by referring to the binomial distribution, we approximated the theoretical probability (p_{th}) of obtaining 0, 1 and 2 errors. This approximation assumes that all the 0,1 and 2 bit errors are decoded successfully. To that end, we use the stationary bit-flip probability computed over a AWGN channel. The value of the stationary bit-flip probability of the channel energy per transmitted information bit, and the probability of $n_b = \{0, 1, 2\}$ at an $\frac{E_b}{N_0} = 5.5dB$ are calculated in (4.1).

Table 4.2: The probability of messages to be decoded by the AGRAND scheme used with BM and SC decoding at an $\frac{E_b}{N_0} = 5.5$

	BM BCH [127,106]		SC PC [128,105]		SC CA-PC [128,105+11]	
n_b	p_{th}	p_{ex}	p_{th}	p_{ex}	p_{th}	p_{ex}
$n_b \leq 0$	0.386	0.385	0.362	0.358	0.486	0.481
$n_b \leq 1$	0.755	0.764	0.731	0.733	0.838	0.830
$n_b \leq 2$	0.93	0.927	0.918	0.920	0.964	0.962

$$\begin{aligned}
BER_{in} &= Q_{func}\left(\sqrt{2 \times R \times \tfrac{E_b}{N_0}}\right) \\
&= Q_{func}\left(\sqrt{2 \times 0.85 \times 3.55}\right) \\
&= 0.00747 \\
p_{th}(n_b = 0) &= (1 - BER_{in})^n \\
&= (0.99253)^{127} \\
&= 0.386 \\
p_{th}(n_b = 1) &= \tbinom{N}{1} BER_{in} \times (1 - BER_{in})^{n-1} \\
&= \tbinom{127}{1} \times 0.00747 \times (0.99253)^{126} \\
&= 0.369 \\
p_{th}(n_b = 2) &= \tbinom{n}{2} BER_{in}^2 \times (1 - BER_{in})^{n-2} \\
&= \tbinom{127}{2} \times 0.00747^2 \times (0.99253)^{125} \\
&= 0.175
\end{aligned} \tag{4.1}$$

Table 4.2 shows the p_{th} and p_{ex} for BCH [127,106] code with GRAND assisted BM (GBM), PC [128,105] with GRAND assisted successive cancellation (GSC) decoding and CA-PC [128,105+11] with GSC. We can see that the probability of obtaining a 1, 2 and 3 bit errors in the channel is consistent with the fraction of frames AGRAND can decode with $n_b = \{0, 1, 2\}$. For example, with $n_b \leq 2$, there is a difference of 0.3% between p_{th} and p_{ex} with GBM. This shows that AGRAND is fully capable of decoding most of the error sequences for $n_b \leq 2$ with BCH code [127,106,7].

We obtain higher values for p_{th} while decoding CA-PC with GSC compared to the values of p_{th} obtained using SC with PC and BM on BCH since the PC used on CA-PC has a higher code rate. A higher code rate leads to an increase in the BER_{in} which, in turn, leads to an increase in p_{th}. This increase in percentages is also reflected in p_{ex} of CA-PC with GSC.

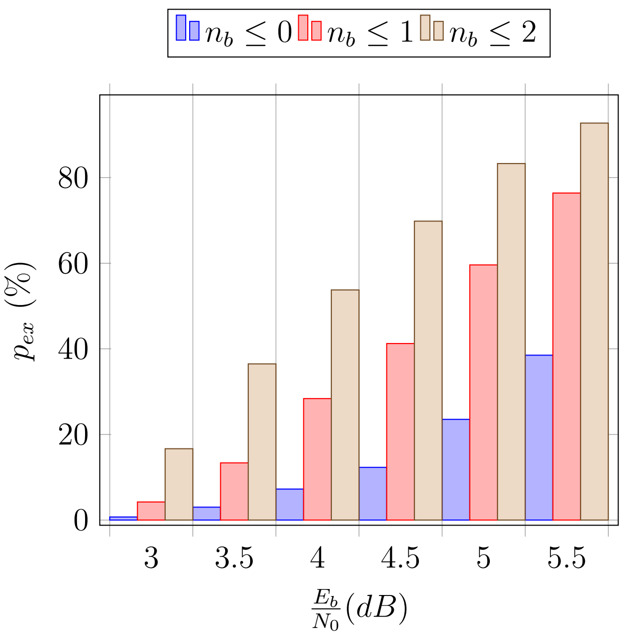

Figure 4.2: The experimental fraction of messages that can be decoded by AGRAND with varying $\frac{E_b}{N_0}$ and n_b

Alternatively, we present in Figure 4.2 the percentage of the total frames decoded by GBM with BCH [127,106] while varying the $\frac{E_b}{N_0}$ and n_b. We can observe that p_{ex} increases with the increase of $\frac{E_b}{N_0}$ with $n_b \leq 2$ from 16.7% at $\frac{E_b}{N_0} = 3.0$ to reach 93% at $\frac{E_b}{N_0} = 5.5dB$. Additionally, we can also observe an increase in p_{ex} from 38.5% with $n_b \leq 0$ to 92.7% with $n_b \leq 2$ at an $\frac{E_b}{N_0} = 5.5dB$. Hence, we can see that the new TEPs used with $n_b \leq 2$ are contributing towards the decoding the codeword in AGRAND decoding.

4.1.3 Area and Power Considerations

Since AGRAND decoding is typically used as a stage preceding conventional decoding, the conventional decoder can be clock-gated to reduce power consumption. This reduction in power consumption comes at a cost of increasing the area utilization on the device. This high area utilization originates from using a highly

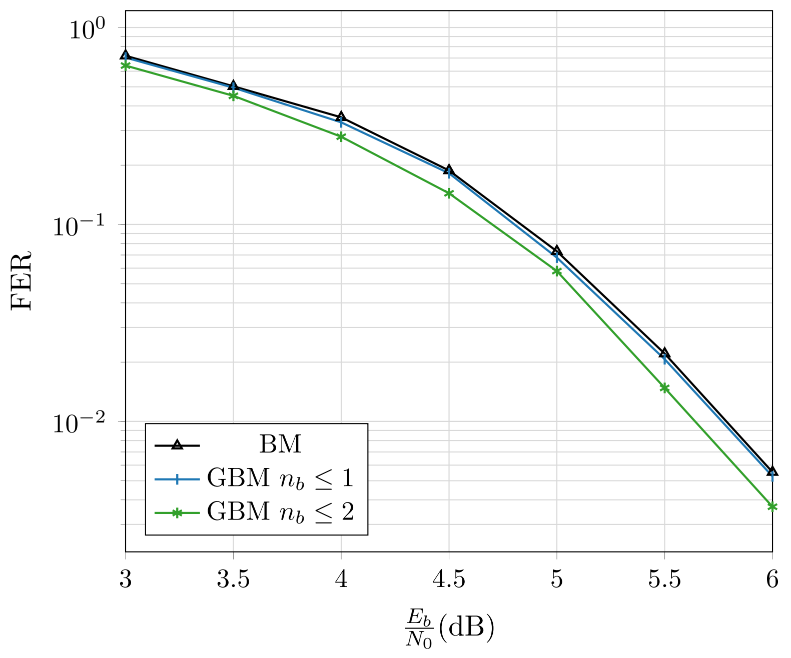

Figure 4.3: BCH FER Performance

parallelized architecture for GRAND. [4].

4.2 Experimental Results & Analysis

4.2.1 Performance of AGRAND with BCH Decoding

In this section, we will investigate the use of AGRAND with a low latency architecture for BCH decoding. We use a fixed latency parallel BCH decoder as a baseline for our analysis.

We can see from Figure 4.3 that the addition of the GRAND decoder to the decoding scheme improves the FER performance of BM. The improvement in FER performance is seen with $n_b \leq 2$ and is more pronounced at high SNRs. For example, at an $FER = 10^{-2}$ and a $n_b \leq 2$, using GBM provides a $0.2dB$ gain in performance as opposed to using the standalone BM decoder.

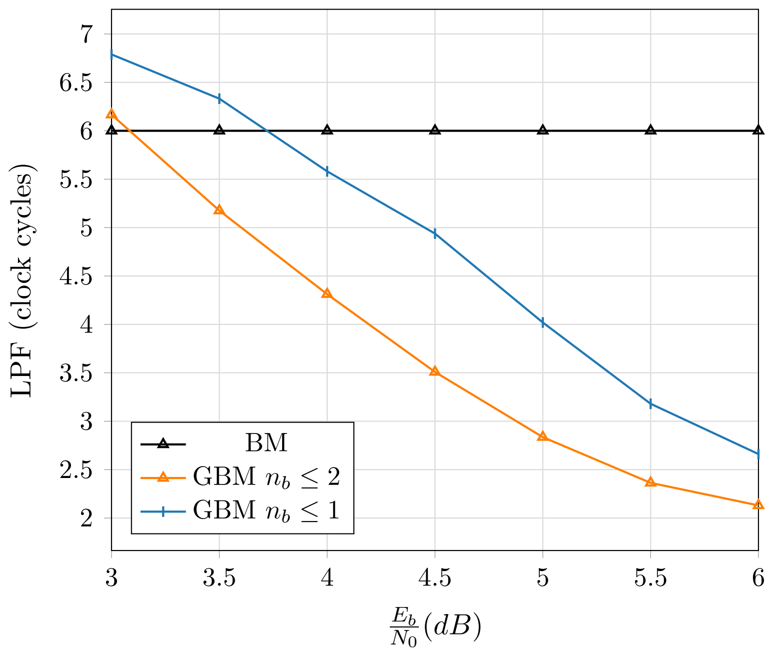

Figure 4.4: BCH Latency Performance

Using AGRAND with the BM decoder, however, increases the latency of the decoder for low SNRs. The latency of the conventional BCH decoder is calculated to be $t + 3$ by using a highly parallelized syndrome check, a highly parallelized Chien search architecture [47] and a low latency BM architecture [45].

We can observe that the use of AGRAND with $n_b \leq 1$ alongside the BCH decoding scheme increases the LPF by 13% at $\frac{E_b}{N_0} = 3.0dB$. This increase in LPF is not seen at higher SNRs as we can see a reduction of LPF by 48% at an $\frac{E_b}{N_0} = 5.5dB$ with $n_b \leq 1$. Alternatively, we can also observe a higher reduction in LPF by using $n_b \leq 2$ at high SNRs whereby the average latency is reduced by 60% at an $\frac{E_b}{N_0} = 5.5dB$.

The additional area utilized by AGRAND might offset all the benefits of the small reduction in latency. Hence, it is desirable to use the new architectures for parallelized BCH decoding to reduce power, latency and area at the same time. For instance, the BCH decoder used on BCH [79,64,6] in [46] reduces the latency by 52%–63% and achieves more than 70% power reduction compared to the fully parallelized conventional BCH decoder. The latency and power reduction occur with bit error rates less than $10^{-2} - 10^{-4}$.

4.2.2 Performance of AGRAND with SC Decoding

Recently, Arikan discovered the first family of capacity-achieving codes [7]. Arikan proved that polar codes with infinite code lengths asymptotically achieve channel capacity with successive cancellation decoding [7]. Since SC is capacity-achieving, and the current SC decoders have a high LPF, it logically follows to use AGRAND with SC decoding. We simulate 5G NR polar code [128,105] and 5G NR CA-Polar [128,105+11] with GSC. Since the codes are of different code rates, the $\frac{E_s}{N_0}$ is used in this section as a measure of SNR. $\frac{E_s}{N_0}$ normalizes the analysis with respect to the code rate.

Figure 4.5 shows the FER performance of GSC. We start by analysing the performance of PC with SC and GSC. GSC with $n_b \leq 0$ performs as well as SC; however, the same cannot be said about GSC with $n_b \leq 1$ and $n_b \leq 2$. For instance, at an FER of 10^{-3}, we achieve a $0.25dB$ loss in performance by using GSC with $n_b \leq 1$ as opposed to using GSC with $n_b \leq 0$ or SC. This degradation in performance is more pronounced with $n_b \leq 2$, seeing as we achieve a loss of $1.5dB$ at an FER of 5×10^{-2} compared to SC.

Since the minimum Hamming distance of PC [128,105] is 4, most of the degradation of performance originates from a bit being marked as erroneous when in fact it is not erroneous. For example, assuming a codeword has 3 bit errors, the GRAND decoder might create a codeword with 4 bit errors $\in$ codebook $\mathbb{C}$ with $n_b \leq 1$. This type of error is more pronounced with $n_b \leq 2$ whereby we risk converting codewords with 2 bit errors into codewords with 4 bit errors $\in \mathbb{C}$.

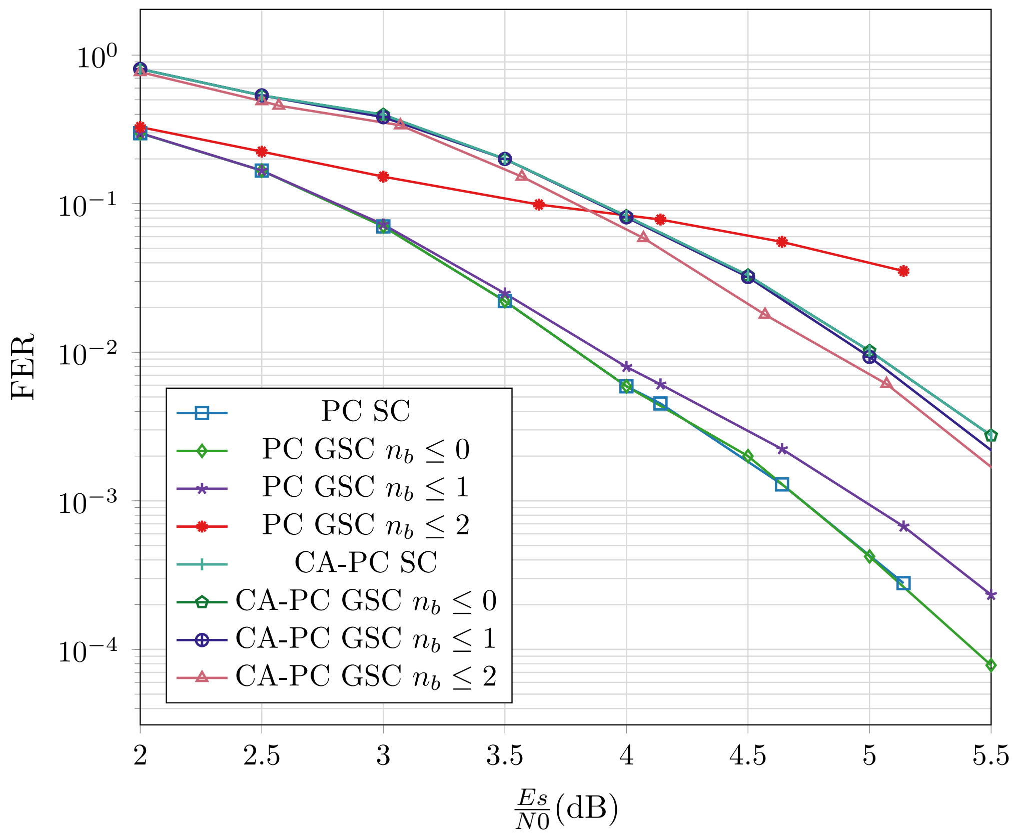

Figure 4.5: FER Performance of Polar Code [105,128] and CA-Polar Code [105+11,128] with GRAND assisted SC Decoding

CA-PC [128,105+11] does not suffer from the same degradation in performance as PC with the increase of n_b since the extra CRC bits provide an additional way to verify the validity of the codeword. This comes at the cost of a reduction in FER performance for SC decoding since we are effectively reducing the number of parity bits.

Hence, AGRAND should not be used for codes with a low minimum Hamming distance since the FER decoding performance might degrade quickly. In case we opt to use AGRAND for a code with low Hamming distance, we need to append extra CRC bits to prevent the decoder from changing the transmitted codeword into another codeword belonging to the same codebook.

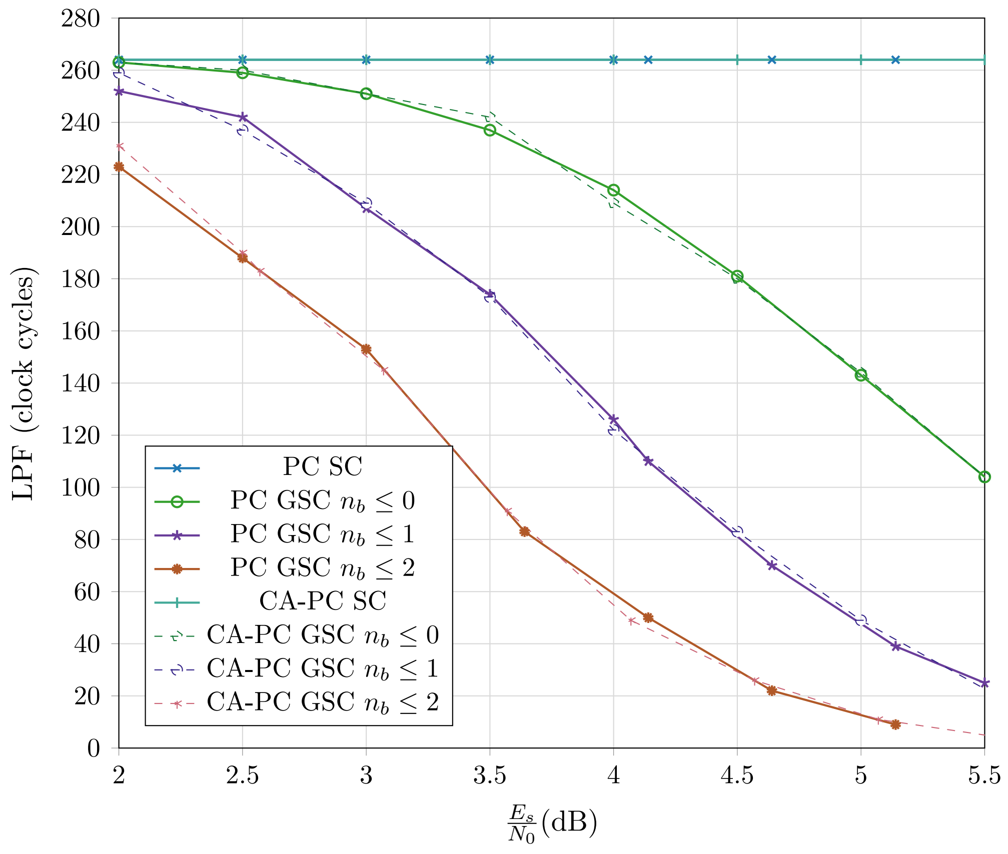

Figure 4.6: SC Code Latency Performance

The latency of SC decoding and the latency of GRAND assisted SC decoding are presented in figure 4.6. The latency of the conventional SC decoder is approximated to be 264 clock cycles by using (4.2) with $p = 16$ and $n = 128$ [27]. Usually, the second term in (4.2) is ignored since it is very small compared to the first term. Hence, our analysis remains valid for any degree of parallelism.

$$l_{list} = (2)n + \frac{n}{p} \log \frac{n}{4p} \tag{4.2}$$

We can see that by using GSC with $n_b \leq 2$, the latency of SC decoding could be decreased by 97% to reach 9 clock cycles at an $\frac{E_s}{N_0} = 5.14dB$. This decrease in LPF can be also seen with $n_b \leq 0$ and $n_b \leq 1$ where we achieve a 51% and 84%

reduction in LPF respectively at an $\frac{E_s}{N_0} = 5.14dB$. Additionally, at lower SNRs, we can observe a decrease in the LPF by 5%, 20% and 41% with $n_b \leq 0$, $n_b \leq 1$ and $n_b \leq 2$ respectively, at an $\frac{E_s}{N_0} = 3.0dB$.

4.2.3 Performance of AGRAND with SCL Decoding

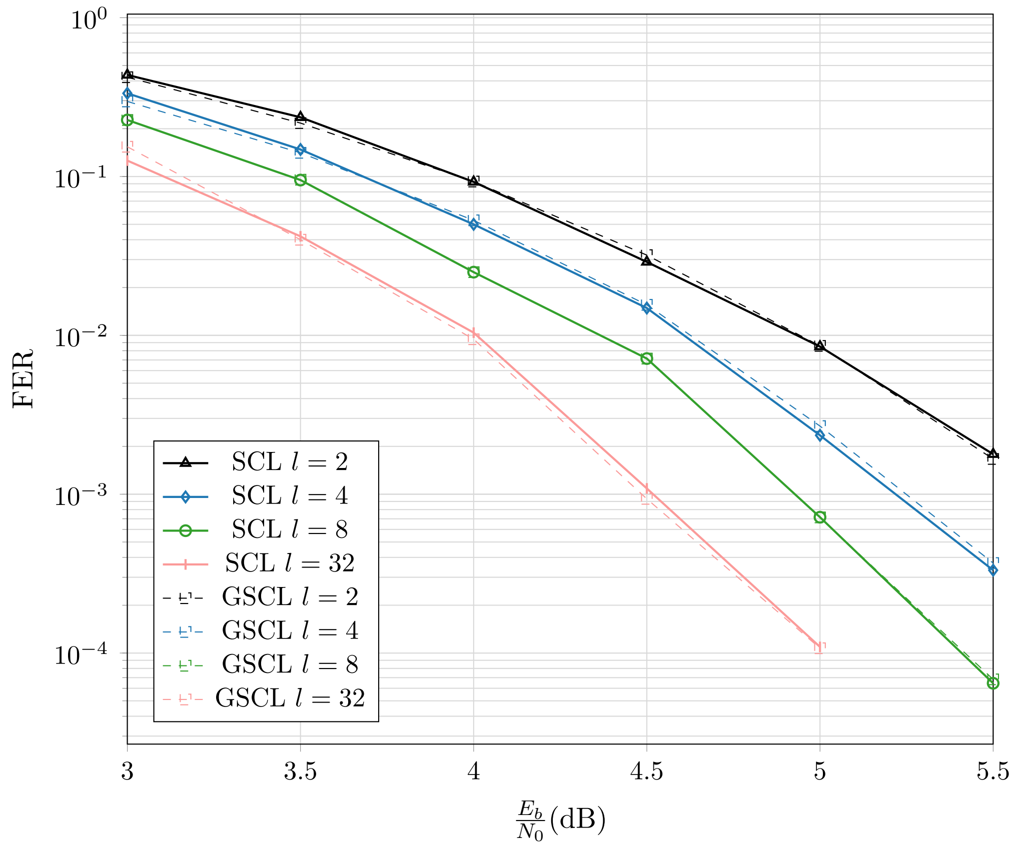

Figure 4.7: SCL FER Performance on 5G NR Polar code with AGRAND

AGRAND is useful when the conventional decoder achieves a desirable FER decoding performance but lacks a manageable latency. In the previous section, we showed that using GSC with CA-PC does not lead to the degradation of FER performance. In this section, we discuss the use of another decoder specifically designed to be used with CA-PC. One of the major decoders that is commonly used with short CRC aided polar code is the SCL decoder since it achieves ML performance with with a large list size.

We started our analysis by verifying that the AGRAND stage does not affect the FER performance of the decoding scheme. To that end, we simulated SCL decoders with list size l = {2,4,8,32} on 5G NR CA-PC. Additionally, we simulated the SCL decoders with added AGRAND decoding for each of the aforementioned list sizes. The results in Figure 4.7 show that the FER performance is not affected with the addition of the AGRAND stage regardless of list size. The same cannot be said about the latency of the decoding scheme.

The latency of SCL decoding is computed to be 369 clock cycles per frame by using (4.3) from [29], with $n = 128$, $r = 0.82$ and $p = 16$.

$$l_{SCL} = (2 + r)n + \frac{n}{p} \log \frac{n}{4p} \tag{4.3}$$

The latency of SCL decoding and the latency of GSCL decoding are presented in figure 4.8. We can see that, irrespective of list size, the latency of SCL decoding could be decreased by 83.7% to reach 60 clock cycles at an $\frac{E_b}{N_0} = 5.5dB$ by using GSCL. Additionally, at lower SNRs, we can see a decrease of 9.3% in the LPF at an $\frac{E_b}{N_0} = 5.5dB$. This clearly shows that AGRAND could be used with SCL decoding of short CA-PC to reduce the LPF without degrading the FER performance.

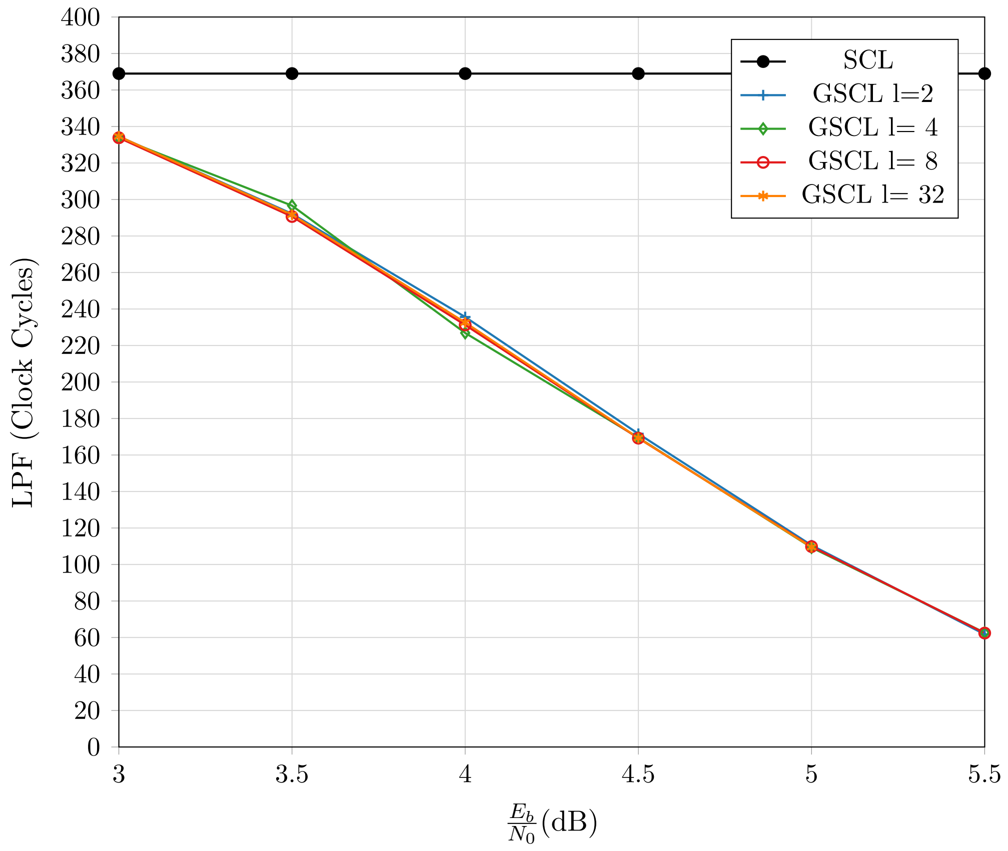

Figure 4.8: SCL Code LPF Performance

Chapter 5

Conclusion

In this thesis, we lay the groundwork for several interesting variations of GRAND decoding. Here, we summarize these contributions and discuss some avenues for future research.

5.1 Thesis Summary

In Chapter 3, we introduce PGRAND, a new GRAND algorithm capable of decoding any high rate code and achieving near ML decoding capability. PGRAND uses the quantized channel reliability information to determine the most likely test error patterns. This is done by dividing the codeword into several partitions and assigning a weight to each partition based on its reliability. It is shown that, irrespective of the type of error-correcting code used, PGRAND performs equally well with different codebooks. PGRAND also achieves a $0.2dB$ gain compared to ORBGRAND at an FER of 10^{-4} as well as a 50% reduction in the QPF performance at an $\frac{E_b}{N_0} = 5.5dB$. SGRAND slightly outperforms PGRAND at high $\frac{E_b}{N_0}$ at the cost of higher scheduling complexity and lower parallelizability with SGRAND. The superior performance of PGRAND over ORBGRAND and the lower scheduling complexity over SGRAND opens up new possibilities for the use of PGRAND in ultra reliable low latency communication for short high rate error-correcting code.

Chapter 4 continues to discuss the use of GRAND in order to assist other structured code decoders in reducing the overall latency of the decoding scheme. We begin by discussing the use of AGRAND with a highly parallelized conventional BM decoder on BCH code [127,106]. Although the scheme could reduce the LPF without impacting the FER performance, the area utilized by the AGRAND decoder could be put to better use with the faster BCH decoder architectures as in [46]. We move on to discuss a decoding scheme for GRAND assisted SC decoding of CA-polar [105+11,128] and polar [105,128]. It is noted that the FER performance of polar code [105,128] is reduced with the use of AGRAND, while that of CA-polar [105+11,128] remains the same. The main cause for the degradation of FER performance is the low minimum Hamming distance of the code. Additional CRC bits provide another way to verify the validity of the message. Hence, the added CRC bits prevent the degradation in FER performance with a large N_b value, but they still increase the code rate.

Hence, AGRAND is best used with a high latency decoder on a CA-code or on a code with a large minimum Hamming distance. Finally, we discuss the usage of AGRAND with SCL decoding on polar code with list size=$\{2,4,8,32\}$. It is shown that AGRAND is able to reduce the average latency per frame for the SCL decoder by 83.7% at an $\frac{E_b}{N_0} = 5.5$ with no degradation in the FER performance. These findings open up new avenues to use AGRAND as a latency and power reduction scheme to be used with ML decoders.

5.2 Future Work

There are many avenues for future work. This thesis highlights the benefits of using PGRAND and AGRAND; however, a highly parallelized architecture for both of these decoders is not yet discussed.

5.2.1 PGRAND with New Channels

PGRAND currently supports additive noise Gaussian channels; however, suitable adjustments can be made to extend its use for Gilbert-Elliot channels, binary erasure channels (BEC), Rayleigh fading channels and optical channels. The pattern generation could be modified to account for the error bursts in Gilbert-Elliot channels, erasure of bits in a binary erasure channel or signal fading with Rayleigh fading channels.

5.2.2 Hardware Implementation of PGRAND

In this thesis, we proposed PGRAND as a highly useful algorithm for decoding short high rate codes. To verify our simulation results and obtain the actual area, power and latency usage, we need to implement a highly parallelized architecture for PGRAND. Since PGRAND allows for a high number of bit-flips to occur during the decoding process, the development of a highly parallelized hardware architecture remains a daunting task. A bit-true model has been created which reduces the complexity of bit-flipping by using short segments and limiting the number of bit-flips in each segment. In this bit-true model, we use the same hardware to implement a 2 bit-flip and a 6 bit-flip in an 8 bit segment. The remainder of the work should be focused on reducing the area utilized by this decoder.

5.2.3 Hardware Implementation of AGRAND

A similar hardware architecture to the hardware implementation of GRAND [4] and ORBGRAND [18] should be developed for AGRAND. Implementing AGRAND on hardware allows us to fully determine the costs and benefits of using AGRAND alongside other decoders. It is important to note that the area usage, latency reduction and power dissipation of this module is of significant interest in our analysis.

Bibliography

[1] A. Cassagne, O. Hartmann, M. Léonardon, K. He, C. Leroux, R. Tajan, O. Aumage, D. Barthou, T. Tonnellier, V. Pignoly, B. Le Gal, and C. Jégo, "AFF3CT: a fast forward error correction toolbox!" *SoftwareX*, vol. 10, p. 100345, Jul. 2019.

[2] "5G New Radio Polar Coding." [Online]. Available: https://www.mathworks.com/help/5g/gs/polar-coding.html

[3] 3GPP, "NR Multiplexing and channel coding," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 3GPP.38.212, Nov. 2021.

[4] S. M. Abbas, T. Tonnellier, F. Ercan, and W. J. Gross, "High-throughput VLSI architecture for GRAND," in *2020 IEEE Workshop on Signal Processing Systems (SiPS)*, 2020, pp. 1–6.

[5] D. Feng, L. Lai, J. Luo, Y. Zhong, C. Zheng, and K. Ying, "Ultra-reliable and low-latency communications: applications, opportunities and challenges," *Science China Information Sciences*, vol. 64, no. 2, p. 120301, Jan. 2021.

[6] X. You, C. X. Wang, J. Huang, X. Gao, Z. Zhang, M. Wang, Y. Huang, C. Zhang, Y. Jiang, J. J. Wang, M. Zhu, B. Sheng, D. Wang, Z. Pan, P. P. Zhu, Y. Yang, Z. Liu, P. Zhang, X. Tao, S. Li, Z. Z. Chen, X. Ma, I. Chih-Lin, S. Han, K. Li, C. Pan, Z. Zheng, L. Hanzo, X. S. Shen, Y. J. Guo, Z. Ding, H. Haas, W. Tong, P. P. Zhu, G. Yang, J. J. Wang, E. G. Larsson, H. Q. Ngo, W. Hong, H. Wang, D. Hou, J. Chen, Z. Z. Chen, Z. Hao, G. Y. Li, R. Tafazolli, Y. Gao, H. V. Poor, G. P. Fettweis, Y. C. Liang, Y. Xiaohu, W. Chengxiang, H. Jie, G. Xiqi, W. Michael, H. Yongming, Z. Chuan, J. Yanxiang,

Z. Min, W. Dongming, P. Zhiwen, Z. Pengcheng, Y. Yang, L. I. U. Zening, Z. Ping, T. Xiaofeng, L. Shaoqian, M. Xinying, H. Shuangfeng, L. Ke, P. Chengkang, H. Lajos, S. Xuemin, G. Y. Jay, D. Zhiguo, T. Wen, Z. Peiying, Y. Ganghua, W. Jun, L. E. G, N. Hien, H. Wei, W. Haiming, H. Debin, C. Jixin, C. Zhe, H. Zhang-cheng, L. Geoffrey, T. Rahim, G. Yue, P. Vincent, F. Gerhard, L. Ying-chang, Z. Zhang, M. Wang, Y. Huang, C. Zhang, Z. Pan, P. P. Zhu, Y. Yang, Z. Liu, and P. Zhang, "Towards 6G wireless communication networks: Vision, enabling technologies, and new paradigm shifts," *SCIENCE CHINA Information Sciences*, vol. 020300, no. 2016, pp. 1–76, 2020.

[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, no. 7, pp. 3051–3073, Jul. 2009.

[8] 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*, 1993, pp. 1064–1070 vol.2.

[9] R. G. Gallager, "Low-Density Parity-Check Codes," *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, 1962.

[10] 3GPP, "Multiplexing and channel coding (FDD)," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 3GPP.25.212, Jul. 2021.

[11] P. Elias, "Coding for two noisy channels," *Information Theory (C. Cherry, ed.),*, pp. 61–76, 1956.

[12] M. P. Fossorier and S. Lin, "Soft decision decoding of linear block codes based on ordered statistics," *IEEE International Symposium on Information Theory - Proceedings*, vol. 41, no. 5, p. 395, 1994.

[13] C. Yue, M. Shirvanimoghaddam, B. Vucetic, and Y. Li, "A revisit to ordered statistic decoding: distance distribution and decoding rules," *arXiv:2004.04913*, Apr. 2020.

[14] K. R. Duffy, J. Li, and M. Médard, "Capacity-Achieving Guessing Random Additive Noise Decoding," *IEEE Transactions on Information Theory*, vol. 65, no. 7, pp. 4023–4040, Jul. 2019.

[15] A. Solomon, K. R. Duffy, and M. Médard, "Soft Maximum Likelihood Decoding using GRAND," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6.

[16] I. Tal and A. Vardy, "List Decoding of Polar Codes," *IEEE Transactions on Information Theory*, vol. 61, no. 5, pp. 2213–2226, 2015.

[17] K. R. Duffy, "Ordered Reliability Bits Guessing Random Additive Noise Decoding," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* Toronto, ON, Canada: IEEE, Jun. 2021, pp. 8268–8272.

[18] S. M. Abbas, T. Tonnellier, F. Ercan, M. Jalaleddine, and W. J. Gross, "High-Throughput VLSI Architecture for Soft-Decision Decoding with ORBGRAND," in *ICASSP*, 2021, pp. 8288–8292.

[19] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, Jul. 1948.

[20] R. H. Morelos-Zaragoza, *The art of error correcting coding*, 2nd ed. Chichester ; Hoboken, NJ: John Wiley, 2006.

[21] G. A. Barnard and R. M. Fano, *Transmission of Information: A Statistical Theory of Communications.* Cambridge, Mass.: MIT Press, 1962.

[22] R. G. Gallager, "The Random Coding Bound Is Tight for the Average Code," *IEEE Transactions on Information Theory*, vol. 19, no. 2, p. 244, 1973.

[23] Y. Fan, S. Ling, H. Liu, J. Shen, and C. Xing, "Cumulative Distance Enumerators of Random Codes and their Thresholds," *arXiv:1212.5679*, Dec. 2012.

[24] 3GPP, "Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and channel coding," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 3GPP.36.212, Aug. 2021.

[25] I. Tal and A. Vardy, "How to Construct Polar Codes," *IEEE Transactions on Information Theory*, vol. 59, no. 10, pp. 6562–6582, Oct. 2013. [Online]. Available: http://ieeexplore.ieee.org/document/6557004/

[26] C. Condo, F. Ercan, and W. J. Gross, "Improved successive cancellation flip decoding of polar codes based on error distribution," in *2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, Apr. 2018, pp. 19–24.

[27] O. Dizdar and E. Arikan, "A High-Throughput Energy-Efficient Implementation of Successive Cancellation Decoder for Polar Codes Using Combinational Logic," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 3, pp. 436–447, Mar. 2016. [Online]. Available: http://ieeexplore.ieee.org/document/7419248/

[28] K. Niu and K. Chen, "CRC-Aided Decoding of Polar Codes," *IEEE Communications Letters*, vol. 16, no. 10, pp. 1668–1671, Oct. 2012.

[29] A. Balatsoukas-Stimming, A. J. Raymond, W. J. Gross, and A. Burg, "Hardware architecture for list successive cancellation decoding of polar codes," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 61, no. 8, pp. 609–613, Aug. 2014.

[30] X. Liang, J. Yang, C. Zhang, W. Song, and X. You, "Hardware Efficient and Low-Latency CA-SCL Decoder Based on Distributed Sorting," in *2016 IEEE Global Communications Conference (GLOBECOM)*, 2016, pp. 1–6.

[31] S. A. Hashemi, C. Condo, and W. J. Gross, "Fast and Flexible Successive-Cancellation List Decoders for Polar Codes," in *IEEE Transactions on Signal Processing*, 2017, pp. 5756–5769.

[32] P. Koopman, "Best CRC Polynomials," 2018. [Online]. Available: https://users.ece.cmu.edu/ koopman/crc/

[33] R. Bose and D. Ray-Chaudhuri, "On a class of error correcting binary group codes," *Information and Control*, vol. 3, no. 1, pp. 68–79, Mar. 1960.

[34] A. Hocquenghem, "Codes correcteurs d'erreurs," *Chiffers*, vol. 2, pp. 147–156, 1959.

[35] M. Mao, Y. Cao, S. Yu, and C. Chakrabarti, "Optimizing Latency, Energy, and Reliability of 1T1R ReRAM Through Cross-Layer Techniques," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 6, no. 3, pp. 352–363, 2016.

[36] S. Gregori, A. Cabrini, O. Khouri, and G. Torelli, "On-chip error correcting techniques for new-generation flash memories," *Proceedings of the IEEE*, vol. 91, no. 4, pp. 602–616, 2003.

[37] S. H. Kang, A. M. Technology, Q. T. Incorporated, and S. Diego, "Embedded STT-MRAM for Energy-efficient and Cost-effective Mobile Systems," *2014 Symposium on VLSI Technology (VLSI-Technology): Digest of Technical Papers*, vol. 115, pp. 2013–2014, 2014.

[38] S. B. Wicker and V. K. Bhargava, *Reed-Solomon Codes and Their Applications.* Wiley-IEEE Press, 2010.

[39] I. T. Union, "Forward error correction for high bit-rate DWDM submarine systems," *SERIES G.975.1*, vol. 975, pp. 1–58, 2005.

[40] J. L. Massey, "Shift-Register Synthesis and BCH Decoding," *IEEE Transactions on Information Theory*, vol. 15, no. 1, pp. 122–127, 1969.

[41] Y. Sugiyama, M. Kasahara, S. Hirasawa, and T. Namekawa, "A method for solving key equation for decoding goppa codes," *Information and Control*, vol. 27, no. 1, pp. 87–99, 1975.

[42] D. Gorenstein, W. W. Peterson, and N. Zierler, "Two-error correcting Bose-Chaudhuri codes are quasi-perfect," *Information and Control*, vol. 3, no. 3, pp. 291–294, 1960.

[43] E. R. Berlekamp, "Non-binary BCH decoding," North Carolina State University. Dept. of Statistics, Tech. Rep., 1966.

[44] R. Chien, "Cyclic decoding procedures for Bose- Chaudhuri-Hocquenghem codes," *IEEE Transactions on Information Theory*, vol. 10, no. 4, pp. 357–363, Oct. 1964.

[45] H.-C. C. Chang and C. B. Shung, "New serial architecture for the Berlekamp-Massey algorithm," *IEEE Transactions on Communications*, vol. 47, no. 4, pp. 481–483, 1999.

[46] S. Choi, H. K. Ahn, B. K. Song, J. P. Kim, S. H. Kang, and S. Jung, "A Decoder for Short BCH Codes With High Decoding Efficiency and Low Power for Emerging Memories," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 2, pp. 387–397, Feb. 2019.

[47] J. Cho and W. Sung, "Strength-Reduced Parallel Chien Search Architecture for Strong BCH Codes," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 55, no. 5, pp. 427–431, 2008.

[48] M. Yin, M. Xie, and B. Yi, "Optimized algorithms for binary BCH codes," in *Proceedings - IEEE International Symposium on Circuits and Systems*, 2013, pp. 1552–1555.

[49] Y. Wu and C. N. Hadjicostis, "Soft-decision decoding using ordered recodings on the most reliable basis," *IEEE Transactions on Information Theory*, vol. 53, no. 2, pp. 829–836, 2007.

[50] ——, "Soft-decision decoding of linear block codes using preprocessing and diversification," *IEEE Transactions on Information Theory*, vol. 53, no. 1, pp. 378–393, 2007.

[51] C. Yue, M. Shirvanimoghaddam, Y. Li, and B. Vucetic, "Segmentation-discarding ordered-statistic decoding for linear block codes," in *2019 IEEE*

Global Communications Conference, GLOBECOM 2019 - Proceedings, 2019, pp. 1–6.

[52] K. R. Duffy and M. Médard, “Guessing random additive noise decoding with soft detection symbol reliability information - SGRAND,” in *IEEE International Symposium on Information Theory - Proceedings*, 2019, pp. 480–484.

[53] K. R. Duffy, A. Solomon, K. M. Konwar, and M. Médard, “5G NR CA-polar maximum likelihood decoding by GRAND,” in *2020 54th Annual Conference on Information Sciences and Systems (CISS)*, Mar. 2020, pp. 1–5.