Towards Practical Length-Compatible Polar Codes

by Adam Cavatassi

Department of Electrical and Computer Engineering
McGill University, Montréal, Québec

April 2019

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

“I fear that I am ordinary just like everyone.”

- Billy Corgan

Contents

ABSTRACT

In 2008, a new class of block error correction codes, known as polar codes, were proven by Erdal Arıkan to be able to achieve the Shannon limit. Through inventive new decoding algorithms and fast code constructions, polar codes have become an attractive high-performance error correction code for practical use. These innovations have resulted in adoption of polar codes in the upcoming 3GPP 5^{th} generation standard for New Radio. Still, polar codes are hindered by certain inflexible characteristics. Arıkan's original polar code definition limits block lengths to powers of two, due to a recursive Kronecker product of the 2×2 polarizing kernel. This constraint presents a considerable obstacle, as many realistic scenarios call for all code lengths to be readily available. Rate-matching techniques, known as puncturing and shortening, allow for flexible polar code lengths, albeit with inefficient decoding complexity. Multi-kernel polar codes produce native code lengths that are powers of two and/or three with the addition of a 3×3 ternary kernel, although they necessitate specialized decoders and code design. This thesis will explore and propose techniques that are intended for maximizing the flexibility and efficiency of polar codes, as well as analyze any trade-offs affecting error correction performance. An in-depth study is presented that compares state-of-the-art length-flexible polar codes with the 3GPP standardized polar codes. This inquiry finds that the 5G standard offers a highly simplified polar code construction with minimal loss to error correction performance. Further, multi-kernel polar codes were found to have a negative correlation between error correction performance and the quantity of ternary Kronecker constituents. This thesis also proposes a new fast successive cancellation decoder that is compliant with multi-kernel polar codes. The ternary kernel is further investigated by testing its rate-matching and systematic properties. Finally, this thesis proposes a new scheme called asymmetric polar codes. We present details on generator matrix definition, information set design, and decoding schedules, as well as perform comparisons with competing schemes using simulations and a comprehensive analysis. Asymmetric polar codes offer flexible block lengths with decoding complexity lower than equivalent length-compatible polar codes under successive cancellation. The enclosed findings indicate that asymmetric polar codes afford comparable error correction performance to the competing schemes, while dividing the number of successive cancellation decoding operations by up to a factor of two. The thesis is then concluded by recommending appropriate extensions of this work for future research.

ABRÉGÉ

En 2008, une nouvelle classe de codes correcteurs d'erreurs en bloc, connus sous le nom de codes polaires, a été prouvée par Erdal Arıkan capable d'atteindre la limite de Shannon. Grâce à des algorithmes innovents et des constructions de code rapides, les codes polaires sont devenus des codes correcteurs d'erreurs à haute-performance attrayants pour une utilisation pratique. Ces innovations ont résulté dans l'adoption des codes polaires dans la prochaine norme 3GPP de téléphonie, dite de 5^{e} génération. Pourtant, les codes polaires sont entravés par la rigidité de certaines de leur propriétés. La définition du code polaire original d'Arıkan limite la longueur des blocs à seulement des puissances de deux, en raison de l'utilisation du produit de Kronecker et du noyau polarisant de taille 2×2. Cette contrainte constitue un obstacle considérable, car de nombreux scénarios pratiques exigent la disponibilité facile de toutes longueurs de code. Des techniques permettant d'atteindre des longueurs de code polaires flexibles existent notamment sous le nom de poinçonnage et de raccourcissement. Cependant elles possèdent une complexité de décodage inefficace. Les codes polaires multi-noyaux produisent des longueurs de code natives qui sont des puissances de deux et/ou trois avec l'ajout d'un noyau ternaire de 3×3, bien qu'ils nécessitent des décodeurs et la conception de code spécialisés. Cette thèse explore et propose des techniques destinées à maximiser la flexibilité et l'efficacité des codes polaires, ainsi que d'analyser tout compromis affectant la performance de la correction d'erreur. Une étude approfondie est présentée qui compare ces codes polaires flexibles en longueur avec les codes polaires stardardisées par le consortium 3GPP. Cette étude démontre que la norme 5G offre une construction de codes polaires très simplifiée tout en garrantissant une très bonne performance de correction d'erreurs. D'autre part, il a été démontré que les codes polaires multi-noyaux ont une corrélation négative entre la performance de la correction d'erreur et le nombre de noyaux ternaires utilisés. Cette thèse propose également un nouveau décodeur rapide à annulations successives pour des codes polaires multi-noyaux. Le noyau ternaire fait l'objet d'une étude plus approfondie en évaluant ses propriétés systematiques et de flexibilité. Enfin, cette thèse propose un nouveau schéma de codage appelé codes polaires asymétriques. Nous présentons la définition de la matrice génératrice, la conception de l'ensemble des bits gelés, et le décodage. Nous effectuons des comparaisons avec des schémas concurrents. Il apparait alors que les codes polaires asymétriques offrent des performances en terme de correction d'erreurs comparables à celles des schémas concurrents, tout en divisant le nombre d'opérations de décodage via l' algorithme à annulation successives par un facteur de deux. La thèse se termine ensuite par la recommandation d'extensions appropriées de cette étude pour des recherches futures.

ACKNOWLEDGEMENTS

I would like to give thanks to my supervisor Dr. Warren Gross for his valuable guidance and support. His direction and wisdom was paramount during the development of our new coding scheme.

I need to further thank my peers for all of their valued advice and insight, alphabetically: Arash Ardakani for imparting his vast knowledge in hardware design and neural network optimization; Harsh Aurora for inducting me into the ISIP lab with utmost kindness, his limitless help with coding, research, and teaching, his accompaniment on late night work, and his valuable friendship; Carlo Condo for introducing me to polar codes and his collaboration on my first research paper; Furkan Ercan for helping develop my polar codes knowledge; Ali Hashemi for always answering all my questions about polar codes and life, his critical review of my work, and his essential guidance; Jerry Ji for his endless knowledge of all things Linux, information theory, and hardware design, and his aid in confirming my ideas; Loren Lugosh for his sense of humour and his machine learning expertise, and finally Thibaud Tonnellier, my esteemed coauthor, for his valuable mentorship, companionship, and collaboration in developing my research and C++ proficiency. I would like to extend special mentions to the following peers, mentors, and collaborators: Jeremy Cooperstock, Benoit Champagne, Nghia Doan, Elie Mambou, Brett Meyer, and Joelle Pineau. Through all of your advice, in-depth discussions, and collaborations, I have grown both as an engineer and an academic.

Finally, I will thank all my family, friends, and loved ones for their support, visits, and jars of tomatoes while completing my graduate degree in Montréal.

CONTRIBUTIONS OF AUTHOR

All simulations and algorithms presented in this thesis were developed using the author's own C++ simulation suite, *PocoSim*. The findings in Chapter 3 feature simulations and analyses that are the author's original work. The algorithms and evaluations presented in Chapter 4 are the author's own research contribution, which has been detailed in a research paper accepted to IEEE Wireless Communications and Networking Conference (WCNC) 2019, titled "Fast Decoding of Multi-Kernel Polar Codes". Chapter 5 outlines a novel coding scheme developed by the author, which is assessed by an in-depth simulation campaign and complexity analysis. The new code design is documented in a research paper accepted to IEEE International Conference on Communications (ICC) 2019, titled "Asymmetric Construction of Low-Latency and Length-Flexible Polar Codes". All software, data, plots, figures, and tables are original work.

List of Figures

List of Tables

Abbreviations

5G	**5**th **g**eneration
APC	**a**symmetric **p**olar **c**ode
AWGN	**a**dditive **w**hite **G**aussian **n**oise
BEC	**b**inary **e**rasure **c**hannel
BER	**b**it **e**rror **r**ate
BPSK	**b**inary **p**hase-**s**hift **k**eying
BR	**b**it-**r**eversed
CA-SCL	CRC-**a**ided **s**uccessive **c**ancellation **l**ist
CRC	**c**yclic **r**edundancy **c**heck
ECC	**e**rror-**c**orrection **c**ode
eMBB	**e**nhanced **m**obile **b**road**b**and
FER	**f**rame **e**rror **r**ate
FEC	**f**orward **e**rror **c**orrection
FSSC	**f**ast **s**implified **s**uccessive **c**ancellation
GA	**G**aussian **a**pproximation
LLR	**l**og **l**ikelihood **r**atio
LDPC	**l**ow **d**ensity **p**arity **c**heck
MK	**m**ulti-**k**ernel
ML	**m**aximum **l**ikelihood
NR	**N**ew **R**adio
QUP	**q**uasi-**u**niform **p**uncturing
R0	**r**ate-**0**
R1	**r**ate-**1**
REP	**rep**etition

SC	**s**uccessive **c**ancellation
SCL	**s**uccessive **c**ancellation **l**ist
SPC	**s**ingle **p**arity **c**heck
SNR	**s**ignal to **n**oise **r**atio
UPO	**u**niversal **p**artial **o**rder
WL	**W**ang-**L**iu

Chapter 1

Introduction

The research field of error correction codes is born out of the necessity to mitigate the inherent unreliability of data transmission over physical media. Information theory, or the study of quantification and communication of information, was pioneered by Claude Shannon in 1948 with his landmark paper, “A Mathematical Theory of Communication” [1]. Information theory serves as the backbone for designing error correction codes, which act to minimize the probability of data transmission error over noisy channels by encoding messages into robust codewords. For many years, industrial digital communication standards turned to turbo codes [2] and low-density parity check (LDPC) codes [3] for reliable information transfer. These codes are highly effective error correction codes and come close to the theoretical limit of reliable communication known as the channel capacity.

In 2008, Erdal Arıkan proved that a new coding scheme, known as a polar codes, is able to asymptotically achieve the channel capacity of a binary memoryless channel with infinite code length [4]. When considering polar codes for practical use, there are several inherent deficiencies that have hindered their acceptance as a valuable coding scheme. The proposed decoder for polar codes, successive cancellation, is sequential in nature, which induces a low throughput. Further, polar codes offer mediocre error correction performance at short to medium code lengths compared with turbo codes or LDPC codes. These issues were quickly overcome with the advent of fast decoders [5] [6], which significantly reduce decoding complexity, and list decoders [7] [8] [9], which considerably improve the error correction performance of polar codes. In 2016, polar codes became practical to the point where they were included in the latest 3GPP 5th generation standard for New Radio [10] [11].

Polar codes lack the native flexibility that is desired for practical application. Namely, Arıkan's polar code definition can only achieve code lengths that are powers of two. Rate-matching techniques, known as puncturing [12] and shortening [13], have been applied to polar codes to grant a flexible block length. However, these schemes are inefficient to decode. Multi-kernel polar codes [14] improve natural block length flexibility, although they introduce convoluted code designs and require specialized decoders [15]. As such, there is currently a demand for efficient and flexible polar coding techniques.

1.1 Objective

The objective of this thesis is to maximize the practicality of polar codes by augmenting code design, decoding efficiency, and improving overall flexibility.

1.2 Summary of Contributions

Outlined below are brief summaries of the research contributions of this graduate thesis.

Analysis of Length-Compatible Polar Codes

With the advent of the 3GPP 5th generation New Radio specification, there now exists an industry standard for length-flexible polar codes. This section outlines the alternate state-of-the-art flexible polar coding schemes, namely puncturing, shortening, and multi-kernel construction, and evaluates their efficacy with respect to the newly designed 3GPP standard. Simulations and an in-depth analysis are presented.

Analysis of Multi-Kernel Code Construction and Systematic Encoding

Multi-kernel polar codes provide a new polarizing linear code construction that improves the native flexibility of polar codes, albeit at the expense of increased decoding and code design complexity. Moreover, many of the innovative algorithms proposed for conventional polar codes over the last 10 years have not been demonstrated to be useful for multi-kernel polar codes. This investigation establishes that complex kernel order design can be simplified and that multi-kernel polar codes cannot be systematically encoded in the same way that Arıkan's polar codes can be.

Fast Decoding of Multi-Kernel Polar Codes

Arıkan's polar coding scheme was explained to posses embedded subcodes that permit simplified decoding, allowing a significant reduction in decoding time complexity. The proposed decoding algorithm in this work modifies the original fast decoder to be compliant with multi-kernel polar codes. Although generalized rules are disclosed for decoding any arbitrary multi-kernel code, efficient simplifications are devised, which are valid under imposed design constraints that reflect typical multi-kernel behaviour. Results indicate that the proposed fast decoder has a minimum 72% latency reduction compared with successive cancellation decoding.

Rate-Matching of Multi-Kernel Polar Codes

Although conventional polar codes are limited to block lengths that are powers of two, rate-matching techniques allow for any length to be attained. The presented research demonstrates that rate-matching is possible for multi-kernel polar codes. Simulations of suggested useful scenarios are exhibited and analyzed. Rate-matching of multi-kernel polar codes can grant decoding complexity reduction while maintaining error correction performance. Alternatively, the featured rate-matching scheme allows for an improvement in error correction performance at the expense of increased decoding latency.

Asymmetric Polar Codes

Motivated by the inefficiency of punctured or shortened polar codes and the overall complexity of multi-kernel polar codes, a novel length-flexible coding scheme is presented. Asymmetric polar codes provide a well-defined code construction that supports any arbitrary codeword length. Asymmetric polar codes display comparable error correction performance to the state-of-the-art length-flexible polar coding schemes, while maintaining reduced decoding complexity over punctured and shortened codes. Code design instructions are introduced, and a comprehensive analysis is performed. Natural extensions are explored; in particular, the effectiveness of rate-matching and systematic encoding of asymmetric polar codes are investigated.

1.3 Thesis Organization

The proceedings of this thesis are as follows: Chapter 2 details the relevant preliminary knowledge required to carry out the presented research, including digital communication systems, error correction codes, and polar codes. Chapter 3 contains all length-flexible polar coding techniques simulated in this work, as well as an in-depth discussion on these techniques as they compare to the 3GPP polar code standard. Chapter 4 outlines all research conducted on maximizing the practicality of multi-kernel polar codes. Chapter 5 presents asymmetric polar codes with details on their implementation and a thorough analysis. Chapter 6 concludes this thesis with a summary of the presented research and an outline of directions for future research extensions.

Chapter 2

Preliminaries

This section will review the background material of the presented research. In order to properly motivate the enclosed analysis, we must cover the basics of forward error correction and define the necessary simulation parameters.

2.1 Forward Error Correction

2.1.1 Model of Communication Systems

A digital communication system can be modeled such that a source consisting of a string of binary numbers is transmitted over a noisy channel and collected by a receiver, as outlined in Fig. 2.1. A fundamental issue associated with communication systems is that the received message may be corrupted and not match the original. For example, if a 0 is sent as the message, there is a chance that channel noise can change this value to a 1 and cause the received value to be incorrect. The possibility of this error compromises the reliability of this model. As such, it is often required for a communication protocol to make the message more robust using forward error correction (FEC), or channel coding.

If the communication protocol is modified to encode some redundancy to the message, it may be possible to mitigate the possibility of message corruption to some degree. For

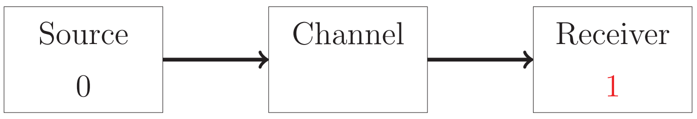


Figure 2.1: A basic communication model.

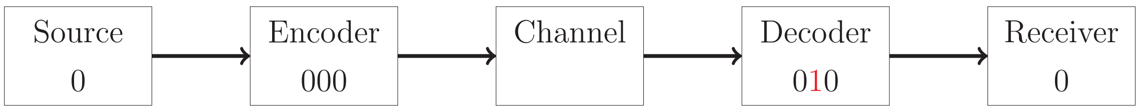


Figure 2.2: A modified communication model using forward error correction.

instance, we can add some verbosity to the message by simply repeating it multiple times. On the receiving side, the message can then be decoded by taking a majority vote between the values of the received bits. This particular error-correction code (ECC) is known as a repetition (REP) code. Although this is a simple channel coding scheme, there exist alternative ECCs that are more robust and more efficient. Fig. 2.2 depicts the modified communication chain that includes a channel encoder and decoder.

2.1.2 Linear Codes

The scope of this work will be limited to binary alphabet ECCs. Linear block codes are a subclass of ECCs that accept a binary message vector a of length K and encode it into a codeword vector x of length $N \geq K$ with a code rate $R = \frac{K}{N}$. These codes are described as linear because any linear combination of valid codewords is also a codeword. A message can be encoded into a codeword by:

$$x = a\boldsymbol{G}, \tag{2.1}$$

where $\boldsymbol{G}$ is a $K \times N$ generator matrix. Each linear block code has a codebook of size 2^K that contains all unique codewords associated with all possible messages. Some ECCs have a structure that causes the encoded message to be explicitly contained within the codeword. These codes are said to be systematic. As an example, the systematic repetition code used in Fig. 2.2 encoded message $a = [0]$ into codeword $x = [000]$ using generator matrix $G = \left[\begin{smallmatrix} 1 & 1 & 1 \end{smallmatrix}\right]$.

Once the source has been encoded and transmitted over the noisy channel, the received noisy vector y must be decoded in order to recover the original message. Although the transmitted codeword is always binary, y may be defined by a different alphabet, depending on the type of channel. This discrepancy will be discussed in detail in the next section. Depending on the type of data that is received, there are various decoding methods that can be employed. Generally, decoding can be described in a few ways: hard-input-hard-output (HIHO), soft-input-hard-output (SIHO), or soft-input-soft-output (SISO). Hard data is represented by bits and soft data is represented by real value probabilities.

An example of a simple HIHO decoding scheme is to convert the received channel vector y directly to a binary vector, compute the codeword estimate $\hat{x}$ by comparing the received binary data to all possible codewords, and select the candidate with the minimum Hamming distance. This particular method is considered a hard-input maximum likelihood (ML) decoding.

In a SIHO decoding method, y is used to compute the probability of a 0 or 1 for each bit in $\hat{x}$. The optimal SIHO decoder is a soft-input maximum likelihood (ML) decoder. This scheme minimizes the Euclidian distance of the computed probabilities and every possible codeword. This decoding method suffers from what is known as the curse of dimensionality; its practicality is limited since 2^K candidates must be evaluated. There exist many alternative soft decoding algorithms for different ECCs that have low decoding complexity but compromise the accuracy of evaluating the transmitted codeword. The entire research field of FEC is highly focused towards optimizing error correcting performance and decoding complexity. This dichotomy will be a central theme of this thesis.

2.1.3 Channel Models

2.1.3.1 Modulation

A modulation scheme is always used to convert digital data to a physical waveform. For the scope of this thesis, it is sufficient to consider the modulator and demodulator as part of the channel. Most digital modulation schemes use a pair of sinusoidal carrier waves that are of the same frequency but phase-shifted by 90°. One carrier is said to be *in phase* while the other is said to be *in quadrature*, which is depicted in Fig. 2.5. A common modulation scheme for measuring the viability of error correction algorithms is binary phase-shift keying (BPSK). This will be the only considered modulator. Phase-shift keying encodes $\mathcal{M}$ bits into one of $2^{\mathcal{M}}$ possible symbols using two sinusoidal carrier waves by mapping symbols to phase shifts according to the constellation of the modulator. BPSK and quadrature phase-shift keying (QPSK) constellations are depicted in Fig. 2.3. In the case of BPSK, there are only two possible symbols, ”0” or ”1”, and so each symbol is represented by either a 0° or 180° phase shift of a single carrier, as demonstrated by Fig. 2.4. In a simulation setting, these phase shifts can be represented as real values 1 and -1. Formally, a source vector $x \in \{0, 1\}$ of length N can be modulated using BPSK

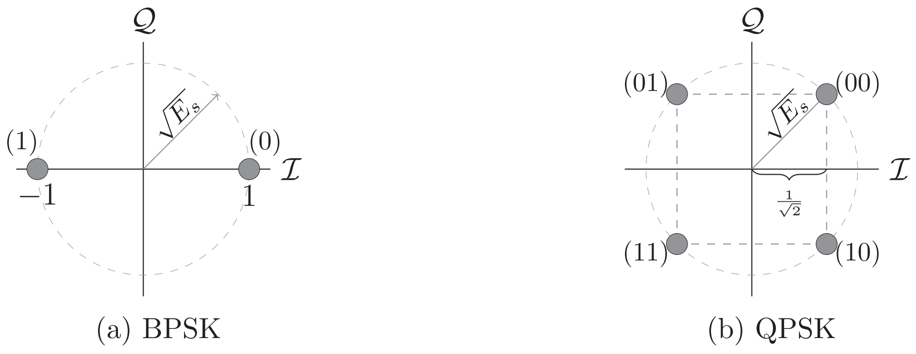


Figure 2.3: Symbol constellations for phase-shift keying modulation schemes.

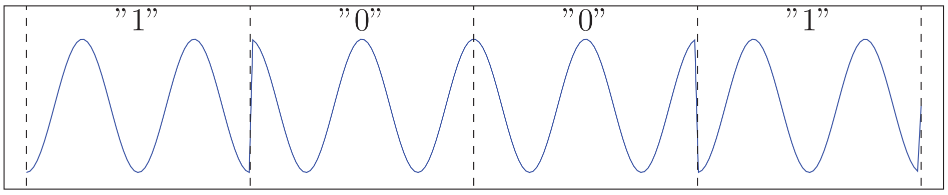


Figure 2.4: A BPSK modulation waveform.

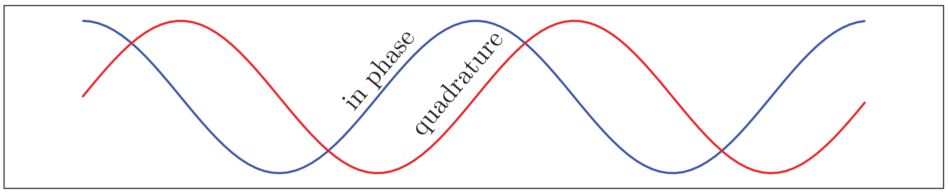


Figure 2.5: Quadrature carriers.

into vector $c \in \{\pm 1\}$ as such:

$$c_i = 1 - 2x_i \quad , \forall\ i \in [0, N) \tag{2.2}$$

It should be noted that BPSK does not require the quadrature carrier. Similarly, QPSK has four possible symbols: "00", "01", "10", or "11". These can be mapped to 180° phase shifts on two carriers, or be represented as complex numbers. A source vector $x \in \{0, 1\}$ of length N can be modulated using QPSK into vector $c \in \{\pm\frac{1}{\sqrt{2}} \pm j\frac{1}{\sqrt{2}}\}$ with:

$$c_i = \frac{1}{\sqrt{2}}(1 - 2x_i) + j\frac{1}{\sqrt{2}}(1 - 2x_{i+1}) \quad , \forall\ 2i \in [0, N) \tag{2.3}$$

2.1.3.2 Types of Channels

The communication channel can be modeled many different ways depending on the level of analysis. For simulation of wireless or optical transmission systems, there exist complex models that accurately reflect the physical behaviour of these channels. For this work, we will only examine simple memoryless models that allow for sufficient comparison of various error correction techniques. This class of channels is known as binary-input discrete memoryless channels (B-DMC). Binary inputs indicate that the channel inputs have alphabet $\mathcal{X} = \{0, 1\}$. The alphabet of the channel output $\mathcal{Y}$ can differ depending on the channel. All channels W have a transition from the input to the output, *ie.* $W : \mathcal{X} \to \mathcal{Y}$. This transition is described by a probability of the output value based on the input value, denoted as $W(y|x), x \in \mathcal{X}, y \in Y$.

Claude Shannon demonstrated that all channels have a capacity $I(W)$ that dictates that maximum allowable code rate for reliable data transmission [1]. Channel capacity represents the maximum amount of information that the channel can carry, measured in shannons, or bits. Shannon's second theorem states that for any $e > 0$, there is a code with a sufficiently large block length N with a rate $R < I(W)$ for which the probability of error is smaller than e. This theorem then provides a performance measurement for communication systems. An ECC that achieves the channel capacity is considered optimal. From this point, the motivation of channel coding research is to design error correction codes that become as close to this theoretical limit as possible while minimizing decoding complexity.

A simple B-DMC is a binary erasure channel (BEC). Under this model, each transmitted bit x has a probability ϵ of having its corresponding received bit $y \in \{0, 1, ?\}$ value become unknown, or erased, at the receiver. Conversely, there is a $1 - \epsilon$ probability of receiving the correct value. Fig. 2.6a depicts the behaviour of a BEC. A BEC has a channel capacity of $I(W) = 1 - \epsilon$.

The disruptive noise in many physical communication systems can be described by a Gaussian random process. As such, the additive white Gaussian noise (AWGN) channel serves as an appropriate model for ECC simulation. AWGN channels can be simulated by adding a Gaussian random variable with a mean of 0 and a variance of σ^2 to the modulated codeword, as depicted in Fig. 2.6b. The variance σ^2 is dependent on the code rate R, modulation rate $\mathcal{M}$, and signal-to-noise ratio (SNR) of the channel, as detailed in the next section. The received data will then be a real value such that $y \in \mathbb{R}$. The transition probability of an AWGN channel when employing BPSK is described by:

$$W(y|x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-x)^2}{2\sigma^2}} \tag{2.4}$$

where x is in modulated form. This distribution of this transition is detailed in Fig. 2.7. AWGN in conjunction with BPSK will be the primary use case for simulations throughout this work.

Under this transmission model, the transition probabilities serve as a measure of the likelihood (P) of a transmitted bit having the value 0 or 1:

$$\begin{aligned} P_0(y) = W(y|0) &= \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-(-1))^2}{2\sigma^2}} \\ P_1(y) = W(y|1) &= \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-1)^2}{2\sigma^2}} \end{aligned} \tag{2.5}$$

For convenience, the same information can be conveyed with simply the likelihood ratio (LR) between the two possibilities:

$$LR(y) = \frac{W(y|0)}{W(y|1)} = \frac{\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-(-1))^2}{2\sigma^2}}}{\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-1)^2}{2\sigma^2}}} \tag{2.6}$$

In a practical setting, LR calculations have a tendency to overflow or underflow. Taking the logarithm of the likelihood ratio (LLR), denoted by λ, helps to mitigate this issue and often simplifies decoding operations:

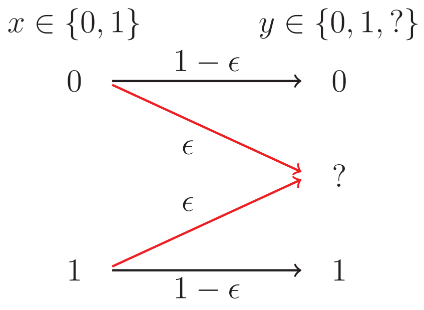
$$\lambda = LLR(y) = \ln \frac{W(y|0)}{W(y|1)} = \ln \frac{\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-(-1))^2}{2\sigma^2}}}{\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-1)^2}{2\sigma^2}}} = -\frac{2y}{\sigma^2} \tag{2.7}$$

Now with a single value, we can express the probabilities of a received corrupted bit. $\hat{x}$ is more likely to be 0 when λ is positive, and more likely to be 1 when λ is negative. A hard decision can be made using an LLR by:

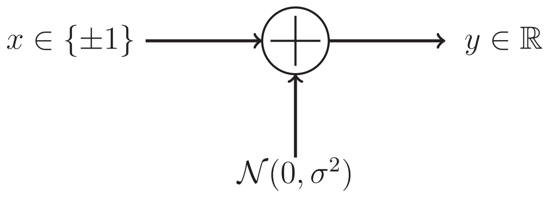
$$\hat{x} = HD(LLR(y)) = \begin{cases} 0 & LLR(y) \geq 0 \\ 1 & \text{otherwise} \end{cases} \tag{2.8}$$

The magnitude of the LLR indicates the reliability, or confidence, of the hard decision. The value of $\hat{x}$ is only known to be completely certain if $LLR(y) = \pm\infty$. Conversely, an LLR of 0 indicates that x is equally likely to be ”0” or ”1”. In this case, the bit is considered erased.

From this point onwards, a refers to the source vector of length K, x refers to the codeword of length N, and y refers to the corrupted codeword of length N.



(a) BEC



(b) AWGN channel using BPSK modulation

Figure 2.6: Models of B-DMCs.

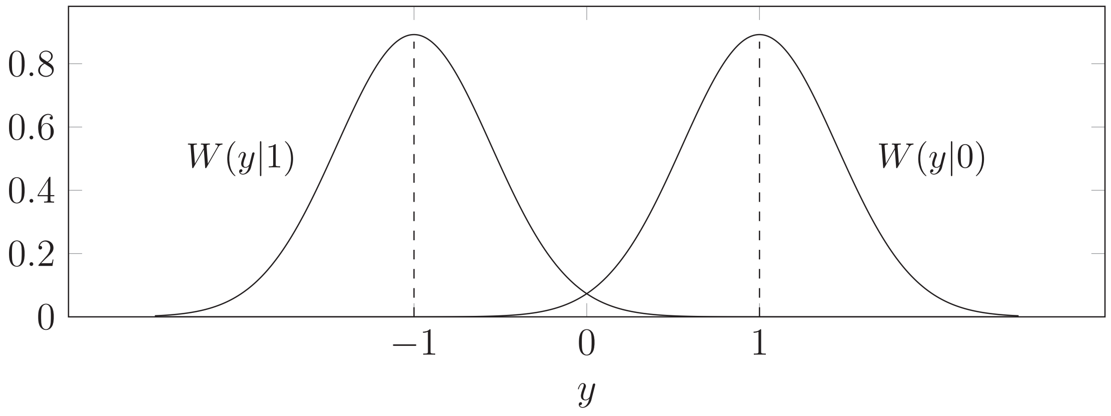


Figure 2.7: Distributions of transition probabilities of AWGN channel using BPSK.

2.1.4 ECC Simulation

This section will outline the parameters necessary for sufficient testing of ECCs. Typically a Monte-Carlo simulation can be used to estimate the error correction performance of an ECC. Such a simulation involves transmitting a number of encoded messages, or frames, over W and obtaining estimates by a decoding algorithm. The bit error rate (BER) can be measured by counting the number of incorrectly decoded bits over the course of the simulation and then dividing that count by the total number transmitted. The frame error rate (FER) can be measured similarly, except by counting the number of codewords instead. Depending on the application of the code design, either of BER or FER may be a more meaningful metric. The simulations presented in this work will only present FER results since we are primarily interested in the frequency with which an entire message is correctly decoded.

To simulate an AWGN channel, the variance must be computed according to the code parameters. The SNR is written as the ratio $\frac{E_s}{N_0}$, where N_0 is the noise power spectrum density and E_s is the normalized energy per modulated symbol. The variance is expressed as $\sigma^2 = \frac{N_0}{2}$. Rearranging the equation, the channel variance can be written as:

$$\sigma^2 = \left(2\frac{E_s}{N_0}\right)^{-1} = \left(2R\mathcal{M}\frac{E_b}{N_0}\right)^{-1}, \tag{2.9}$$

where E_b is the more common energy per bit notation. If SNR is given in decibels (dB), then it must be linearized first before computing σ^2, *ie.* $\frac{E_b}{N_0} = 10^{\frac{SNR}{10}}$.

2.1.5 Cyclic Redundancy Check

Occasionally it is useful to encode a message twice with two different ECCs in series. An outer code is first used to encode the source message a into a', and then a' is encoded into x with an inner code. This process is known as serial code concatenation. This enhancement can be done to have the outer code relay information to the inner code during decoding to improve error correction performance [16].

A common outer code is known as the cyclic redundancy check (CRC). A CRC of length C is a linear cyclic code that performs polynomial division on a binary message of length K with a characteristic polynomial of length $C + 1$ to obtain a unique remainder. The remainder is appended to the original message to obtain a codeword of length $K+C$. After the message is transmitted and decoded, division is applied to the estimated codeword. If the division results in a remainder of all zeros, then no errors are detected. Conversely, if the remainder has at least one 1, it is known that the received codeword has an error.

There are several polynomials for the most common CRC lengths. Table 2.1 lists all the CRC lengths to be referenced in this work, along with the selected polynomials. A divisor is obtained from the polynomial by first converting the hexadecimal representation into a binary string, and then appending a 1 to the leftmost bit. To perform polynomial division with the message, a series of XOR and bit shift operations are carried out, according to the example in Fig. 2.8. This example examines a message [10101101] and a CRC of length 3 with polynomial 0x5.

2.2 Polar Codes

Polar codes are a relatively new class of linear block ECCs introduced originally by Norbert Stotle in 2002 [17] and then later verified by Erdal Arıkan in 2008 [4]. Polar

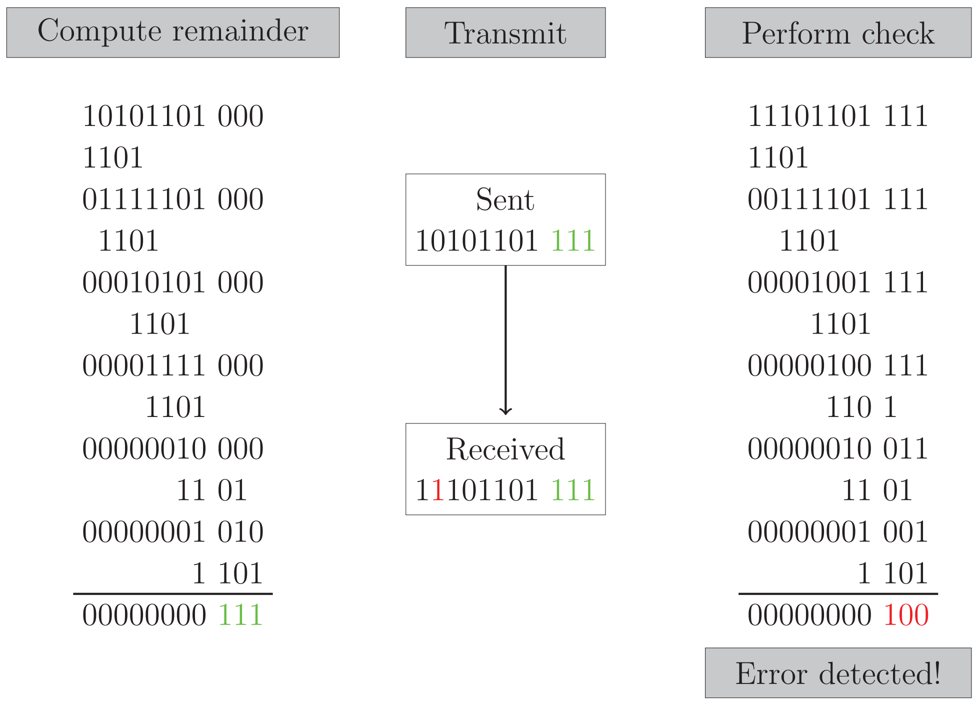
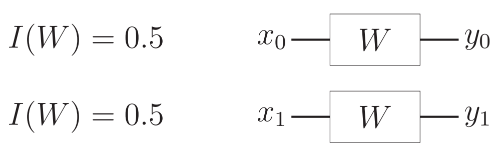


Figure 2.8: An example of the usage of a CRC.

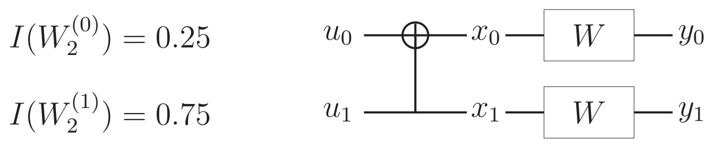
C	Hexadecimal	Divisor
6	0x21	1100001
8	0xD5	111010101
11	0x621	111000100001
16	0x1021	10001000000100001
24	0xB2B117	1101100101011000100010111
32	0x04C11DB7	100000100110000010001110110110111

Table 2.1: A list of CRC polynomials referenced in this work.

codes have become of strong interest because of their proven ability to achieve the channel capacity of any B-DMC at infinite block lengths. Although polar codes are shown to have good error correction performance at very long block lengths using Arıkan's proposed decoder, they are prone to inherent shortcomings such as high decoding latency and mediocre error correction performance at short to medium block lengths. The field of polar code research in the last 10 years has worked to mitigate these issues to the point where polar codes have become a practical coding scheme. Polar codes have also been included in the new 3GPP 5th generation wireless communication standard for New Radio. Since polar codes have demonstrated their capacity for industrial utility, much of the research presented in this thesis is dedicated to making polar codes more practical.



(a) Two copies of W with equal reliability



(b) Transformed channels W_2 with polarized reliability

Figure 2.9: Channel polarization process.

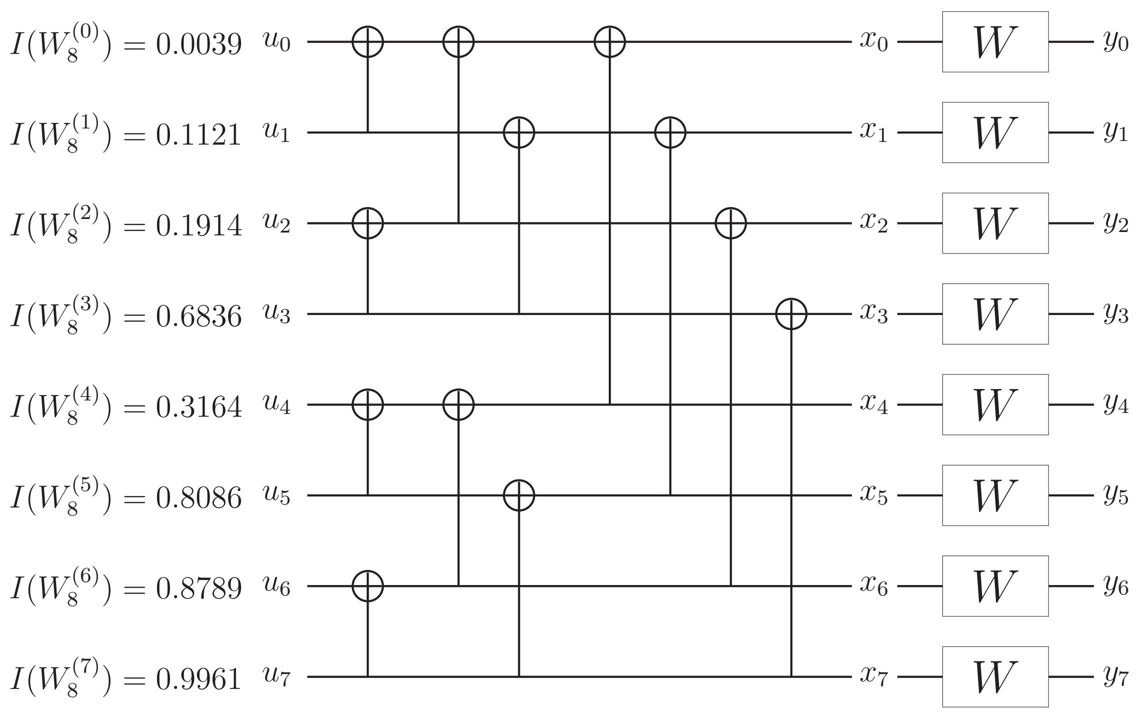


Figure 2.10: Recursive polarization resulting in subchannels W_8.

2.2.1 Channel Polarization

The mechanism from which polar codes are named is channel polarization. The principle of channel polarization causes two copies of a channel W to be transformed into two synthetic subchannels W_2, such that one becomes stochastically upgraded ($W_2^{(1)}$) and the other becomes stochastically degraded ($W_2^{(0)}$). These subchannels are then said to be *polarized* in their reliability. The transformation used to polarize two binary input channels is an XOR operation. The transformation is mathematically represented by a polarizing kernel, further referred to as the *Arıkan* kernel:

$$\boldsymbol{T_2} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \tag{2.10}$$

The reliability of a channel W can be measured by its capacity $I(W)$. Arıkan showed that the reliability of the synthetic subchannels can be accurately tracked for a BEC according to:

$$\begin{aligned} I(W_2^{(0)}) &= I(W)^2 \\ I(W_2^{(1)}) &= 2I(W) - I(W)^2 \end{aligned} \tag{2.11}$$

This polarizing transform is depicted in Fig. 2.9 for a BEC W with $I(W) = 0.5$. The polarization process can be applied recursively to obtain larger sets of polarized synthetic channels. Fig. 2.10 depicts synthetic channels W_8 obtained by applying the transformation two more times. From this example, it should be observed that longer polar codes offer a higher degree of polarization.

2.2.2 Encoding

The generator matrix $\boldsymbol{G}$ for a polar code of length N is obtained by computing the n^{th} Kronecker product, denoted $\otimes$, of $\boldsymbol{T_2}$, where $n = \log_2 N$:

$$\boldsymbol{G} = \boldsymbol{T_2}^{\otimes n} \tag{2.12}$$

It should be noted that $\boldsymbol{G}$ is an $N \times N$ matrix, which does not match the definition of a linear block code in Section 2.1.2. To encode a message a of length K into a polar code of length N, denoted $\mathcal{PC}(N, K)$, a must first be expanded into a vector u of length N. This transformation is done simply by placing the data in a in the indices of u contained in the information set $\mathcal{I}$, which pertain to the K most reliable synthetic channels corresponding to $\boldsymbol{G}$. The remaining $N - K$ indices of u are contained in the frozen set $\mathcal{F}$ and are set to a known value, which is typically 0. This format can be mathematically described with an expanding matrix $\boldsymbol{E}$. If we borrow the notation used in [18], u can be obtained by:

$$u = a\boldsymbol{E}, \tag{2.13}$$

where

$$\boldsymbol{E} = (E_{i,j})_{i=0,j=0}^{K-1,N-1}, \qquad \text{and } E_{i.j} = \begin{cases} 1 & j \in \mathcal{I} \\ 0 & \text{otherwise} \end{cases}. \tag{2.14}$$

There are different reliability metrics and algorithms that can be used to determine $\mathcal{I}$ and $\mathcal{F}$, and these will be outlined in later sections. Once u is determined, the polar encoding process can be computed with the matrix multiplication:

$$x = u\boldsymbol{G}. \tag{2.15}$$

A more convenient representation of the polar encoder is to use a low-complexity factor graph of XOR operations. Fig. 2.11 depicts polar encoders along with the corresponding generator matrices for polar codes with a rate of $R = \frac{1}{2}$.

All instances of outer CRC code concatenation for $\mathcal{PC}(N, K)$ in this thesis will refer to K as the sum of the message length and the CRC length. In other words, only the rate of the inner polar code will be considered when discussing ECC comparisons. However, the computation of the noise variance σ^2 will take into account the rate of the outer code for accurate simulation. For instance, a message of length 6 with an outer CRC of length 3 encoded into an inner polar code of length 16 will have rate $R = \frac{9}{16}$, but $\sigma^2 = \left(2\frac{6}{16}\mathcal{M}\frac{E_b}{N_0}\right)^{-1}$.

This polar encoder is not systematic. A simple systematic polar encoder was proposed in [18] that was shown to be valid under certain constraints. The systematic polar encoding process is carried out by first encoding in the usual manner, then re-freezing all frozen bits (set all bits whose indices are in $\mathcal{F}$ to 0), and then performing hard decoding as shown in Fig. 2.12. It should be noted that the original paper describes the hard decoding stage as an additional stage of encoding. However, polar codes posses a unique structure that causes their encoder and hard decoder to be the same operation, *ie.* $\boldsymbol{G} = \boldsymbol{G^{-1}}$. Although this distinction is somewhat inconsequential, later polar code definitions in this thesis will make use of it.

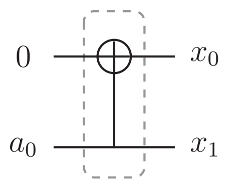
In order for a polar code to be able to be encoded systematically with this method, we need to ensure that the message a appears in the codeword. The systematic encoder can be written as a series of matrix multiplications:

$$a(\boldsymbol{E} \cdot \boldsymbol{G} \cdot \boldsymbol{E^T})(\boldsymbol{E} \cdot \boldsymbol{G^{-1}} \cdot \boldsymbol{E^T}) \tag{2.16}$$

In [18], it was shown that if this formula results in the identity matrix, then the systematic encoder is valid:

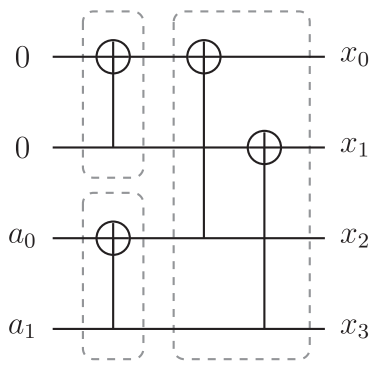
$$(\boldsymbol{E} \cdot \boldsymbol{G} \cdot \boldsymbol{E^T})(\boldsymbol{E} \cdot \boldsymbol{G^{-1}} \cdot \boldsymbol{E^T}) = I \tag{2.17}$$

$$G = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$$



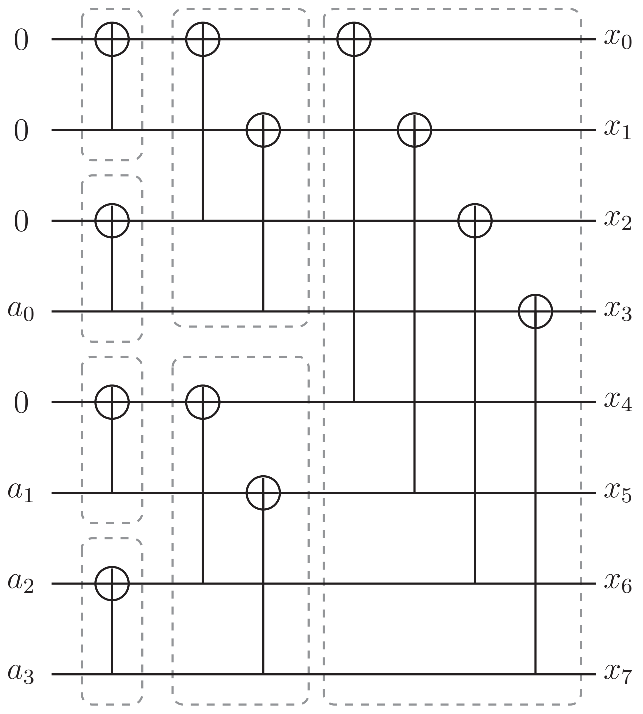
(a) $\mathcal{PC}(2, 1)$ with $\mathcal{I} = \{1\}$

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$



(b) $\mathcal{PC}(4, 2)$ with $\mathcal{I} = \{3, 4\}$

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$



(c) $\mathcal{PC}(8, 4)$ with $\mathcal{I} = \{3, 5, 6, 7\}$

Figure 2.11: Factor graphs of polar encoders with $R = \frac{1}{2}$.

The validity of this encoder is then dependent on $\mathcal{I}$, which dictates $\boldsymbol{E}$. It was proven that $\mathcal{I}$ needs to satisfy a property called *dominant contiguity*, which has a special meaning with regards to polar codes. Formally, an information set $\mathcal{I} \subseteq \{0, 1, \ldots, N-1\}$ that is dominant contiguous satisfies:

$$(h, j \in \mathcal{I} \;\text{ and }\; h \succeq i \succeq j) \implies i \in \mathcal{I}, \qquad \forall\, i \in [0, N), \tag{2.18}$$

where the operator $\succeq$ indicates what is referred to as the *binary domination* relation in [18]. Polar code index $i \in [0, N)$ with binary representation $i_b = (i_0, i_1, \ldots, i_n)$ is said to

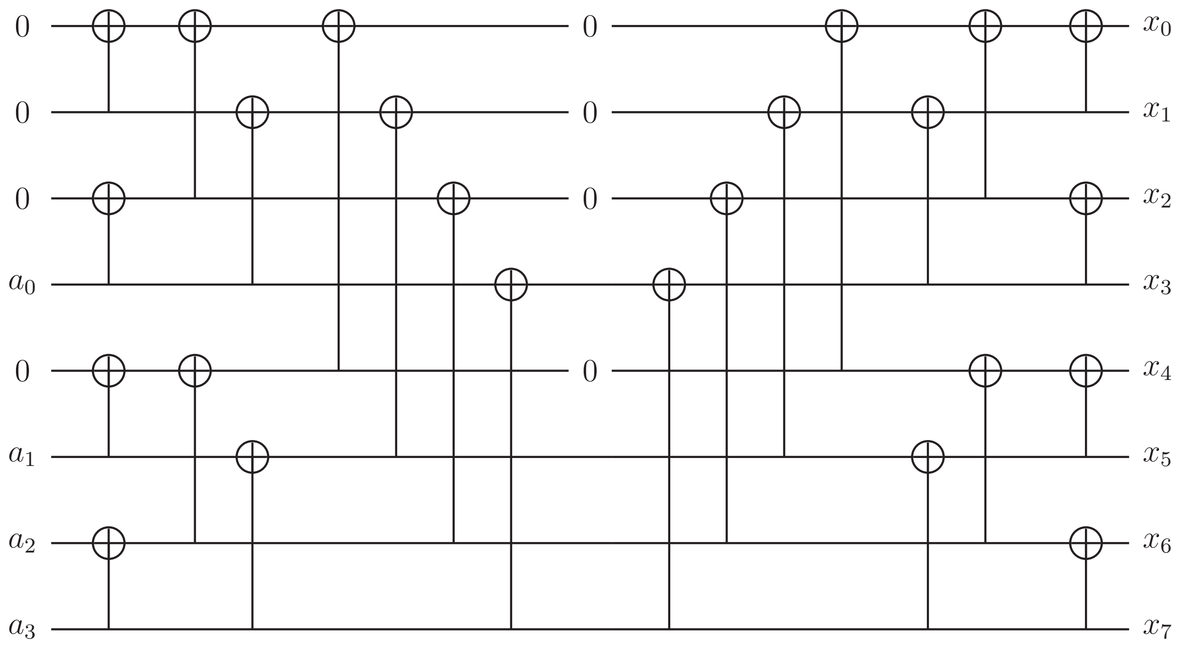


Figure 2.12: Factor graph for a systematic polar code with $R = \frac{1}{2}$ and $\mathcal{I} = \{3, 5, 6, 7\}$.

be binary dominant to index $j \in [0, N)$ with binary representation $j_b = (j_0, j_1, \ldots, j_n)$, or $i \succeq j$, if and only if:

$$i_t \geq j_t, \qquad \forall\, t \in [0, n). \tag{2.19}$$

The binary representation for polar code indices often serves as a useful tool for various polar code algorithms and proofs. However, the effectiveness of this representation is merely a fortunate coincidence that results from the fact that polar codes have a recursive structure that manifests itself in powers of 2. As such, the definition of binary domination has limited use outside of this polar code definition. The binary domination relation does not fully provide a deep understanding of *why* the information set $\mathcal{I}$ must adhere to a particular pattern.

Now we will redefine the dominance relation between polar code indices such that it can be used for alternate polar code definitions that will be introduced in later sections. We can say that a polar code index $i \in [0, N)$ is *column dominant* to index $j \in [0, N)$, or $i \rhd j$, if and only if:

$$\boldsymbol{G}_{k,i} \leq \boldsymbol{G}_{k,j} \;\text{ and }\; \boldsymbol{G}_{i,i}, \boldsymbol{G}_{j,j} = 1, \qquad \forall\, k \in [0, N). \tag{2.20}$$

In other words, i dominates j when the support of column j of $\boldsymbol{G}$ contains the support of column i and the diagonal entry of each column is one. In the setting of Arıkan's polar code definition, column dominance has the exact same meaning as binary dominance, although column dominance has a definition that is applicable to other types of polar codes. Further, column dominance between indices is a more descriptive relationship than binary dominance. Column dominance implies that a polar code index has a coded value that is dependent on another if it is dominated by that index. For example, in a polar code with $N = 8$, index 7 is dominant to all other indices since the coded value of all other indices are dependent on u_7. Conversely, index 3 is dominant to indices 0, 1, and 2, but is dominated only by index 7.

The systematic encoder can then rely on a new definition of dominant contiguity that satisfies:

$$(h, j \in \mathcal{I} \text{ and } h \triangleright i \triangleright j) \implies i \in \mathcal{I}, \qquad \forall\, i \in [0, N), \tag{2.21}$$

Intuitively, dominant contiguity ensures that there are no information bits that are dependent on dominant frozen bits. If this constraint is violated, there is a risk that some frozen bit indices will change the value of dominated information bits during the re-freezing step. It was proven in [18] that perfectly designed information sets are always dominant contiguous in theory. However, the authors conceded that practical code design algorithms with limited computation precision may possibly produce an information set that violates dominant contiguity. Although their empirical evidence suggests that such an event is rare, it can be easily remedied by imposing code construction constraints for cases of systematic encoding. The lower triangular structure of a polar code generator matrix allows for useful information sets that are dominant contiguous. Namely, indices that are dominant to a large number of other indices are more likely to be highly reliable. This observation allows information sets that provide good error correction performance while remaining able to be systematically encoded.

2.2.3 Information Set Design

This section will detail some common methods for designing useful information sets. Because the reliability of polar code indices are said to be channel dependent [4], there exist accurate construction methods that design information sets for a target SNR. However, there has been recent interest in effective fast construction schemes that offer a complexity trade-off by neglecting channel conditions.

2.2.3.1 Bhattacharyya Parameter Expansion

One of the earlier polar code construction algorithms proposed by Arıkan [4] was to determine the Bhattacharyya parameter of each index corresponding to a synthetic subchannel. The Bhattacharyya parameter $Z(W)$ of a channel W is an indicator of its probability of error and is defined as:

$$Z(W) \triangleq \sum_{y \in \mathcal{Y}} \sqrt{W(y|0)W(y|1)}. \tag{2.22}$$

The idea behind this algorithm is that for a set of polarized subchannels W_N, the Bhattacharyya parameter of all synthetic channels $Z(W_N)$ can be determined given the Bhattacharyya parameter of the uncoded channel $Z(W)$ according to the following equations:

$$\begin{aligned} Z(W_N^{(2j-1)}) &= 2Z(W_{\frac{N}{2}}^{(j)}) - Z(W_{\frac{N}{2}}^{(j)})^2, \\ Z(W_N^{(2j)}) &= Z(W_{\frac{N}{2}}^{(j)})^2, \end{aligned} \tag{2.23}$$

where the uncoded channel has Bhattacharyya parameter $Z(W) = \epsilon$, which is obtained from the target SNR. After computing N unique Bhattacharyya parameters, the indices are sorted according to $Z(W_N)$ in ascending order and select the first K indices to be inserted into $\mathcal{I}$. The remaining indices are inserted into $\mathcal{F}$. This method has only been proven to be exact for BECs. While this method is not designed for AWGN construction, it can still be used at the expense of accuracy. It was shown in [19] that the Bhattacharyya construction method is inferior to other construction algorithms when considering the AWGN channel.

2.2.3.2 Gaussian Approximation

A more commonly used construction scheme is known as Gaussian approximation (GA). Proposed by Peter Trifonov [20], this method is designed for the AWGN channel and offers a lower complexity algorithm when compared with density evolution. Under the assumption that uncoded channel W is Gaussian, the LLR of that channel has a Gaussian distribution of $\lambda(y) \sim \mathcal{N}(\frac{2}{\sigma^2}, \frac{4}{\sigma^2})$ when an all-zero codeword is transmitted. Although the intermediate stages of a factor graph have bimodal distributions during soft decoding algorithms, they are nonetheless assumed to be Gaussian for simplicity. The reliability of the synthetic subchannels W_N is then proportional to the mean of their distributions

λ_N, denoted in this work by z_N. The reliabilities of each index can be computed using:

$$\begin{aligned} z_N^{(2j-1)} &= \phi^{-1}(1-(1-\phi(z_{\frac{N}{2}}^{(j)}))^2), \\ z_N^{(2j)} &= 2z_{\frac{N}{2}}^{(j)}, \end{aligned} \tag{2.24}$$

where

$$z_1^{(0)} = \frac{2}{\sigma^2} = 4R\frac{E_b}{N_0}, \tag{2.25}$$

where $\frac{E_b}{N_0}$ is the linearized SNR. Note that the computation of σ^2 neglects the modulation rate, since GA does not take into account any modulation schemes when computing LLRs. The function $\phi(x)$ is defined precisely as:

$$\phi(x) = \begin{cases} 1 - \frac{1}{\sqrt{4\pi x}} \int_{-\infty}^{\infty} \tanh \frac{u}{2} e^{\frac{-(u-x)^2}{4x}} dx, & x > 0 \\ 1, & x = 0 \end{cases}. \tag{2.26}$$

Alternatively, there exist simple approximations from the open source FEC simulation tool, *aff3ct* [21]:

$$\phi(x) = \begin{cases} e^{0.0564x^2 - 0.485x} & x < 0.8678 \\ e^{\alpha x^\gamma + \beta} & \text{otherwise} \end{cases}, \qquad \text{and} \tag{2.27}$$

$$\phi^{-1}(x) = \begin{cases} 4.3049(1-\sqrt{1+0.9567\log x}) & x > 0.6846 \\ (a \log x + b)^c & \text{otherwise} \end{cases}, \tag{2.28}$$

where $\alpha = -0.4527$, $\beta = 0.0218$, $\gamma = 0.86$, $a = \frac{1}{\alpha}$, $b = \frac{-\beta}{\alpha}$, and $c = \frac{1}{\gamma}$. Similarly to the Bhattacharyya method, sort the indices according to z_N in descending order, then insert the first K indices into $\mathcal{I}$ and the remaining indices into $\mathcal{F}$. All codes in this thesis are constructed with GA using the aff3ct simplifications unless otherwise noted.

The GA algorithm is useful beyond sorting indices by reliability. Since the output of GA are the means of the evolved channel LLRs, that information can be used to analytically compute a theoretical FER under SC decoding for an AWGN channel [22]:

$$FER_{\text{th}} = 1 - \prod_{i \in \mathcal{I}} \left(1 - \frac{1}{2} erfc\left(\frac{1}{2}\sqrt{z_N^i}\right)\right). \tag{2.29}$$

As can be seen in Fig, 2.13 this formula computes FER quite accurately in relation to the simulated FER.

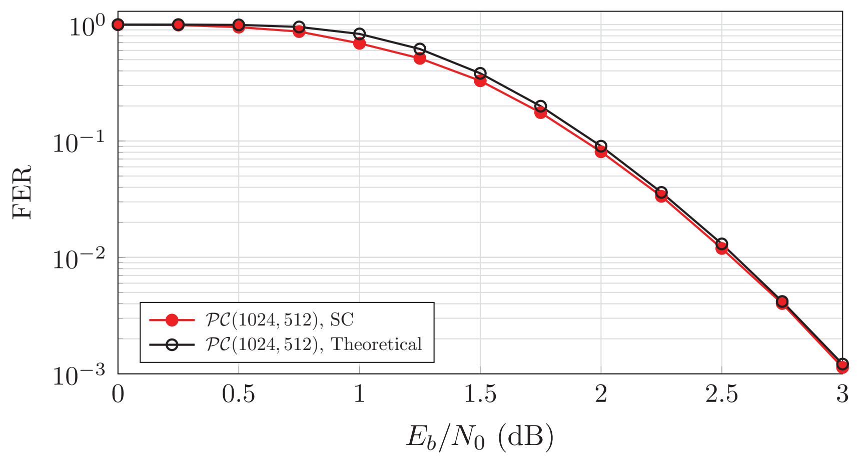


Figure 2.13: A comparison of simulated and theoretical FER measurement under SC decoding in an AWGN environment.

2.2.3.3 Nested Sequences

The recursive property of polar codes alludes to the possibility that reliability orders have a certain universality. It has been demonstrated that ranked reliability sets $\mathcal{R} \in [0, N)$ of polar codes have a nested structure [23], [24]. Namely, the sorted reliability order $\mathcal{R}_{N'}$ for a polar code with length N' can be obtained from any reliability order designed for a polar code with length $N'' > N'$. This property is a byproduct of the fractal structure of polar codes and their inherent universal partial order (UPO). Although the subchannels of a polar code have reliabilities that are channel dependent, it is known that some indices will undeniably be more reliable that other, regardless of the channel conditions. For example, no information set construction algorithm will ever displace the last index as the most reliable.

Given a carefully constructed reliability order $\mathcal{R}''_N \subseteq [0, N'')$ for a polar code of length N'', then any polar code of length $N' < N''$ can be constructed by removing all indices from $\mathcal{R}''_N$ that are less than N' such that $\mathcal{R}_{N'} \subseteq [0, N')$. For example, given a reliability order for a polar code with $N = 16$:

$$\mathcal{R}_{16} = \{15, 14, 13, 11, 7, 12, 10, 9, 6, 5, 3, 8, 4, 2, 1, 0\}, \tag{2.30}$$

the reliability order for a polar code with $N = 8$ can be obtained by:

$$\begin{aligned} \mathcal{R}_8 &= \mathcal{R}_{16} \setminus [8, 16) \\ \mathcal{R}_8 &= \{15, 14, 13, 11, 7, 12, 10, 9, 6, 5, 3, 8, 4, 2, 1, 0\} \\ \mathcal{R}_8 &= \{7, 6, 5, 3, 4, 2, 1, 0\} \end{aligned} \tag{2.31}$$

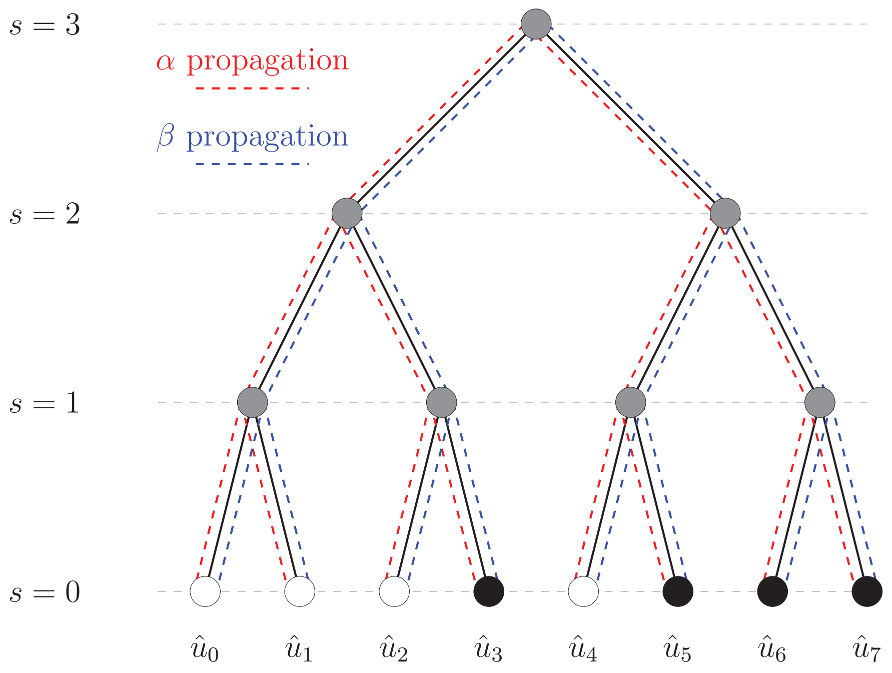


Figure 2.14: SC tree for $\mathcal{PC}(8, 4)$ and $\mathcal{I} = \{3, 5, 6, 7\}$.

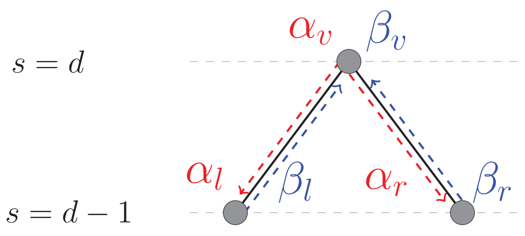


Figure 2.15: SC decoding rules are dictated by left-priority tree traversal.

$\mathcal{I}$ can then be determined by selecting the first K elements of $\mathcal{R}_8$, and $\mathcal{F} = \mathcal{R}_8 \setminus \mathcal{I}$.

2.3 Polar Code Decoding Algorithms

This section will outline the key state-of-the-art soft decoding algorithms for polar codes. All decoding equations are in the LLR domain.

2.3.1 Successive Cancellation

The primary decoder that is used to decode polar codes is known as successive cancellation (SC), which was proposed by Arıkan in [4]. The SC decoder can be visualized as binary tree traversal with left-branch-first priority. The encoder factor graph can be restructured into a tree with a depth of $n+1$. The tree then has N leaf nodes, which represent the estimated message $\hat{u}$. The tree also has n stages of intermediate nodes with 2^{n-s} nodes corresponding to the polarizing transforms emphasized in Fig. 2.11c, where $s \in [0, n]$ is the stage number counting from the bottom of the tree. Each node v contains $N_v = 2^s$ each of LLRs α_v and bit partial sums β_v. Fig. 2.14 visualizes the SC decoder where gray nodes are intermediate, and black and white nodes represent information and frozen bit leaf nodes, respectively. To begin decoding, the received channel values y are assigned to the top of the tree where $s = n$. At each node, the left and right children LLRs, α_l and α_r are computed using functions f and g, respectively:

$$\begin{aligned} \alpha_l^i &= f(\alpha_v^i, \alpha_v^{i+\frac{N_v}{2}}) = \alpha_v^i \boxplus \alpha_v^{i+\frac{N_v}{2}}, \\ \alpha_r^i &= g(\alpha_v^i, \alpha_v^{i+\frac{N_v}{2}}, \beta_l^i) = (-1)^{\beta_l^i} \cdot \alpha_v^i + \alpha_v^{i+\frac{N_v}{2}}, \end{aligned} \qquad \forall\, i \in [0, \frac{N_v}{2}), \tag{2.32}$$

where $a \boxplus b \approx \operatorname{sign}(a) \operatorname{sign}(b) \min(|a|, |b|)$. At each leaf node $i \in [0, N)$, a hard decision is made to estimate the value of $\hat{u}_v$ if the node corresponds to an information bit. Otherwise, $\hat{u}_v$ is set to 0 since it is frozen:

$$\hat{u}_i = \begin{cases} HD(\alpha_i), & \text{if } i \in \mathcal{I} \\ 0, & \text{otherwise.} \end{cases} \tag{2.33}$$

After returning from a right branch, the partial sums in the parent node β_v are updated with:

$$\begin{aligned} \beta_v^i &= \beta_l^i \oplus \beta_r^i, \\ \beta_v^{i+\frac{N_v}{2}} &= \beta_r^i, \end{aligned} \qquad \forall\, i \in [0, \frac{N_v}{2}), \tag{2.34}$$

before traversing the next branch. Ostensibly, an SC decoder can be reduced to a schedule of f and g operations, where the total number of operations is given by $N \log_2 N$. The SC schedule is deterministic given the length of the polar code. The schedule of the decoder in Fig. 2.14 is depicted in Fig. 2.16. The error correction performance of SC improves with the length of the polar code, although SC is considered to have mediocre performance overall. The poor performance is a result of the fact that SC only considers the possibility of a single codeword path candidate, and LLR information in frozen bit positions is discarded. The sequential nature of the SC decoder also poses latency issues. Both of these issues will be rectified with modifications of the SC algorithm in the next sections. A comparison of decoding performance of several polar codes is presented in Fig. 2.17. $\mathcal{I}$ is optimized for each $\frac{E_b}{N_0}$ point using GA.

The memory requirements of the SC decoder have a space complexity of $\mathcal{O}(N \log_2 N)$

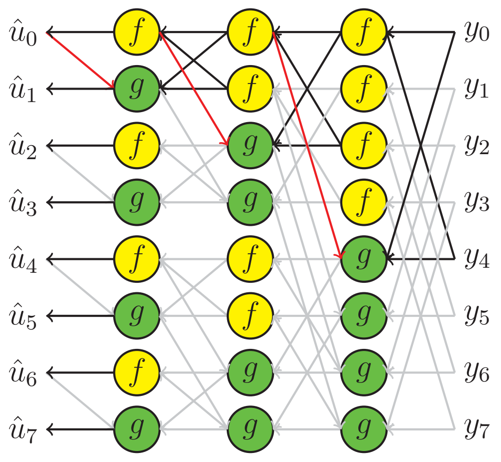


Figure 2.16: SC decoder expressed as a schedule of f and g operations for $N = 8$.

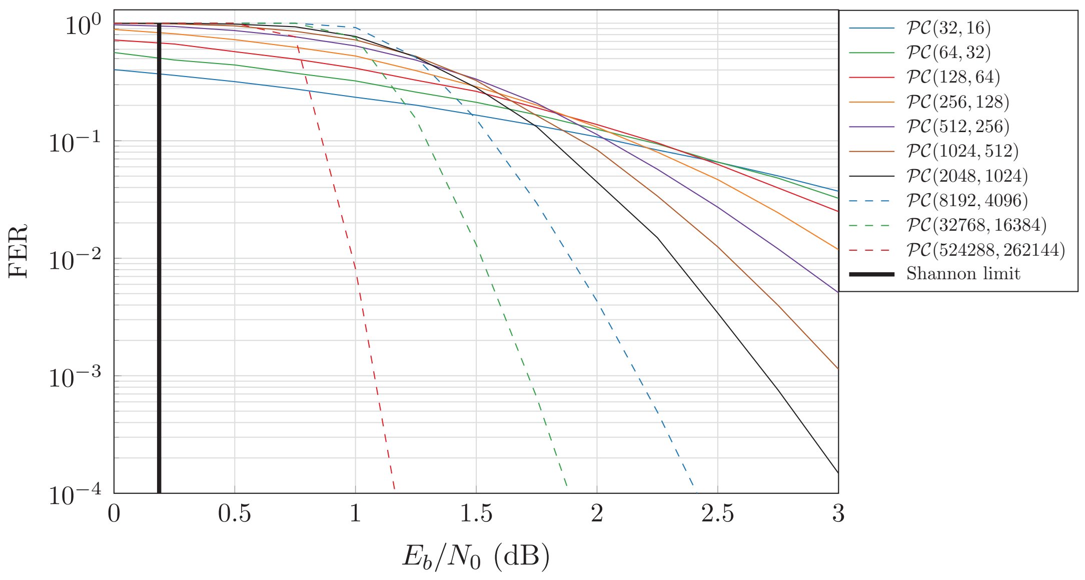


Figure 2.17: Theoretical SC performance on polar codes with $R = \frac{1}{2}$, $N = \{32, \ldots, 500k\}$.

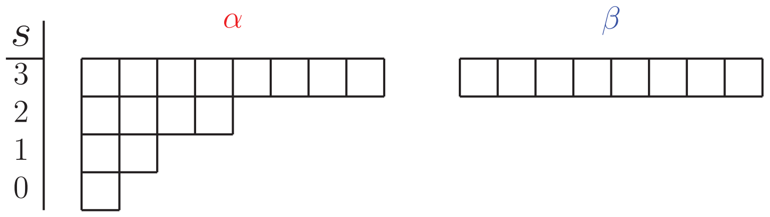


Figure 2.18: SC α and β memory requirements for a polar code with $N = 8$.

with a naive implementation. However, the sequential nature of the decoder allows α and β data to be discarded once it is no longer needed. As such, a memory-efficient SC decoder allows for linear space complexity [25], as depicted in Fig. 2.18.

2.3.2 Successive Cancellation List

To improve upon the lackluster error correction performance of SC, the Successive Cancellation List (SCL) decoder was proposed [8]. Under SCL decoding, L decoding path candidates are considered simultaneously. This is accomplished by duplicating decoding paths at each information bit leaf node i by considering the possibility that $\hat{u}$ is equal to "0" or "1". The number of considered paths is then doubled at each information bit. In order to maintain the list size at L, a pruning criteria must be used to determine which paths to eliminate. In the LLR domain [9], each path $l \in [0, L)$ is associated with a path metric γ_l^i that is updated at each leaf node i using:

$$\gamma_l^i = \begin{cases} \gamma_l^{i-1}, & \text{if } \hat{u}_i = \frac{1}{2}(1 - \operatorname{sign}(\alpha_i^0)) \\ \gamma_l^{i-1} + |\alpha_i^0|, & \text{otherwise.} \end{cases} \tag{2.35}$$

After computing each leaf node, if the number of path candidates is larger than the maximum list size L, then only the L paths with the lowest γ_l^i are allowed to continue decoding. The rest of the paths are discarded. Once all leaf nodes are computed, the path with the lowest path metric is returned as the estimated message $\hat{u}$. Fig. 2.19 presents the performance of the SCL decoder with various maximum list sizes. The SCL decoder is equivalent to SC when $L = 1$. Observe that increasing the list size does not necessarily equate to improved performance. It is clear that a large enough list size causes the polar code to converge to the ML bound.

In [8], it was shown that the SCL decoder can be substantially enhanced by employing an outer CRC concatenation, as described in Section 2.1.5. Tal. et al observed that in instances of decoding error under SCL, the correct path was often among the L best candidates. To improve upon the path selection criteria at the end of decoding, the augmented CRC-aided SCL decoder (CA-SCL) returns the message with the lowest path metric that also has a valid CRC check. This modification was shown to significantly lower the error floor of polar codes to the point where this decoding scheme can outperform state-of-the-art LDPC codes. Although SCL and CA-SCL both have improved error correction performance over SC, they both suffer from the sequential nature of the SC

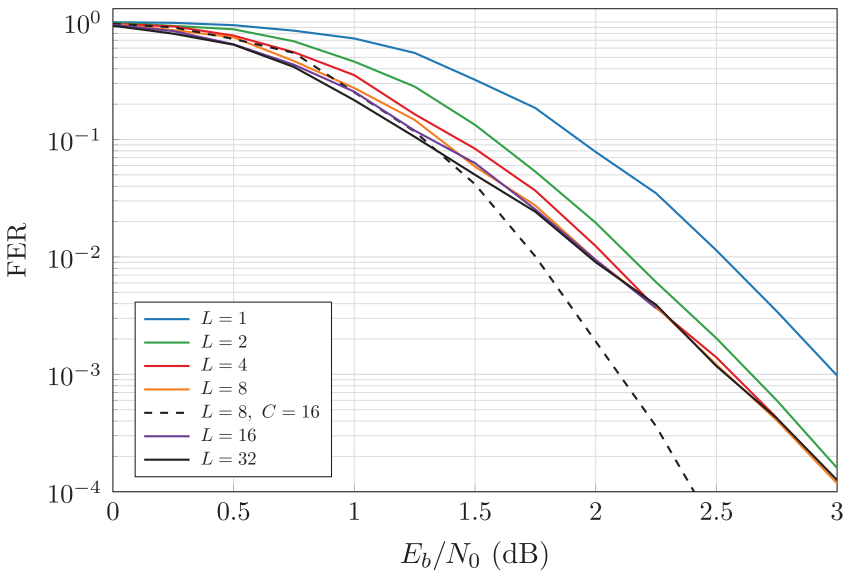


Figure 2.19: SCL and CA-SCL performance on $\mathcal{PC}(1024, 512)$ with $L = \{1, \ldots, 32\}$ and $C = 16$.

tree. In fact, SCL and CA-SCL have a substantial increase in latency over SC with a time complexity of $\mathcal{O}(LN \log_2 N)$. Moreover, the requirement of selecting the best paths imposes the necessity of a sorting algorithm for each instance of list pruning, which further increases the decoding complexity. Fig. 2.19 shows the substantial improvement in decoding performance CA-SCL offers for a list size of 8 using a CRC with $C = 16$.

2.3.3 Fast Simplified Successive Cancellation

The serial nature of the SC decoder is a serious impediment to the practicality of polar codes. In [5], it was shown that an SC decoding tree can be pruned by identifying particular frozen bit patterns. By allowing special intermediate nodes that reflect simple subcodes with fast decoders, the decoding schedule can be altered to reduce latency. When the decoder arrives at a specialized node, a specialized decoder can be enacted with high efficiency that eliminates the need to traverse that branch of the polar code further. This simplified decoder was extended in [6] to recognize additional embedded codes that allowed for further latency reduction. The pruned decoder is known as fast simplified successive cancellation (FSSC). This section will outline the four basic fast nodes that are commonly used.

Rate-0 Node

A Rate-0 (R0) node v with indicies $i \in v$ is one that is parent only to frozen leaf nodes such that $i \in \mathcal{F}$. Further, any node that is parent to a set of intermediate R0 nodes is also a R0 node. Such nodes need not be further traversed since all estimated bits and partial sums are known to be "0":

$$\begin{matrix} \beta_v^i = 0, \\ \hat{u}_v^i = 0, \end{matrix} \qquad \forall\ i \in [0, N_v), \tag{2.36}$$

Rate-1 Node

Similarly, a Rate-1 (R1) node v is one that is parent only to information leaf nodes such that $i \in \mathcal{I}$, as are any nodes that are parent to only R1 nodes. R1 nodes can be decoded by first computing β_v with a hard decision:

$$\beta_v^i = HD(\alpha_v^i), \qquad \forall\ i \in [0, N_v), \tag{2.37}$$

and then updating $\hat{u}_v$ with hard decoding:

$$\hat{u}_v = \beta_v \boldsymbol{G}_{N_v}^{-1}, \tag{2.38}$$

where $\boldsymbol{G}_{N_v}$ is the generator matrix of a polar code of length N_v.

Repetition Node

Any polar code with a rate $R = \frac{1}{N}$, only the highest order bit u_{N-1} will contain information. This frozen set pattern renders the polar code into a repetition (REP) code. Any node v identified as a REP node is decoded simply by summing all LLRs and taking a hard decision on the result. The result is stored in all indices of β_v:

$$\beta_v^i = HD(\sum_j \alpha_v^j), \qquad \forall\ i, j \in [0, N_v), \tag{2.39}$$

The partial sum can then be hard decoded:

$$\hat{u}_v^i = \begin{cases} 0, & i < N_v - 1 \\ HD(\sum_j \alpha_v^j), & i = N_v - 1 \end{cases} \qquad \forall\ i, j \in [0, N_v), \tag{2.40}$$

Single Parity Check Node

For a polar code of rate $R = \frac{N-1}{N}$, only the lowest order bit u_0 will be frozen. This polar code configuration can be interpreted as a single-parity check (SPC) code. For example, if $N = 4$, then $u = (0, u_0, u_1, u_2)$ and $x = (u_0 \oplus u_1 \oplus u_2, u_0 \oplus u_2, u_0 \oplus u_1, u_2)$. An SC node v with $\mathcal{F}_v$ that reflects this pattern can be optimally decoded with low complexity. First, the partial sums β_v are estimated with a hard decision according to eq. 2.37. The parity of β_v is then computed using:

$$\text{parity} = \bigoplus_i HD\beta_v^i, \qquad \forall\; i \in [0, N_v). \tag{2.41}$$

If the parity constraint is not fulfilled, then the least reliable bit at index j is flipped as in

$$\beta_v^j := \beta_v^j \oplus \text{parity}, \tag{2.42}$$

where

$$j = \operatorname*{argmin}_i |\alpha_v^i|, \qquad \forall\; i \in [0, N_v), \tag{2.43}$$

Finally, $\hat{u}_v$ is estimated using eq. 2.38.

Combining these four specialized decoders with SC allows for substantial latency reduction. It is also possible to derive fast decoding rules for these nodes in list decoders [26]. There are exact formulations for R0, R1, and REP nodes for list decoding that allow for the exact same error correction performance as the original SCL decoder. However, SPC nodes require a special approximation in order to remain practical [27], though the FER loss is negligible. Other flavours of SC exist throughout polar code literature, but this thesis will not cover these topics.

Chapter 3

Analysis of Length-Compatible Polar Codes

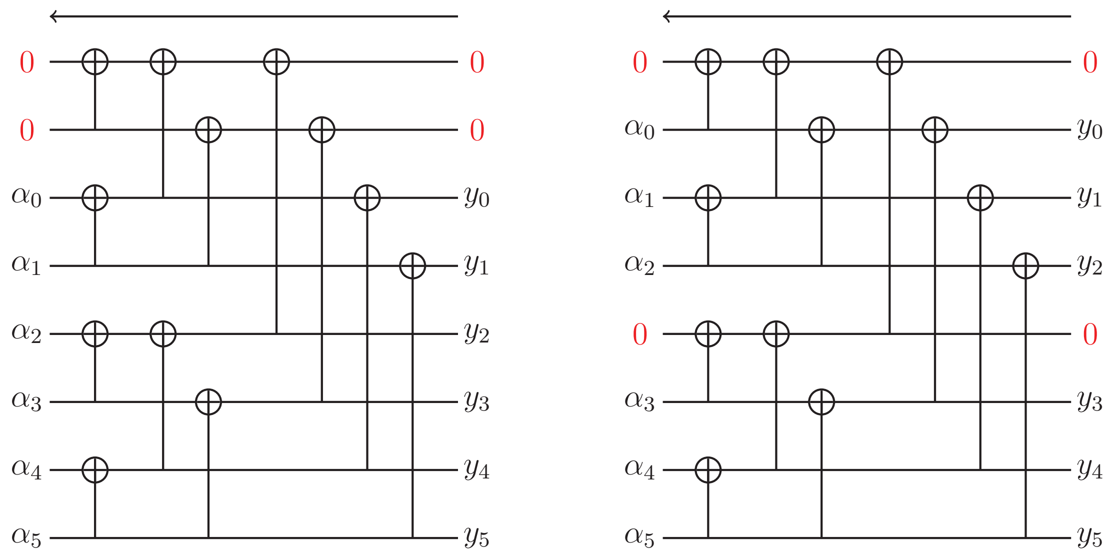
The central theme of this thesis is non-conventional polar coding techniques. A major drawback associated with polar codes is their lack of flexibility, since they can only attain native block lengths that are powers of 2. This chapter will introduce the most studied methods used to attain length-compatible polar codes and present an analysis that compares the feasibility of these schemes. This chapter contains sections on puncturing and shortening techniques, mixed-kernel construction, and the latest 3GPP 5G standard.

3.1 Punctured and Shortened Polar Codes

Puncturing and shortening techniques, sometimes known as rate-matching, support polar code construction of any length. These algorithms can be useful for practical scenarios where channel bandwidth or communication standards require a certain amount of flexibility.

3.1.1 Puncturing

To build a punctured polar code $\mathcal{PC}(E, K)$ of arbitrary length E with rate $R = \frac{K}{E}$, a larger *mother* polar code of length $N = 2^{\lceil log_2 E \rceil}$ is required to rectify the length difference. The mother code is used for the encoding and decoding steps. To puncture, $P = N - E$ indices must be carefully selected to form the puncturing set $\mathcal{P} \subset [0, N)$. All indices in



(a) *First* pattern, $\mathcal{P} = \{0, 1\}$ (b) bit-reversed pattern, $\mathcal{P} = \{0, 4\}$

Figure 3.1: Puncturing patterns for a polar code with $E = 6$.

$\mathcal{P}$ are said to be *punctured*. To form a valid puncturing set, all contained indices must be *incapable*. An incapable index is an index that results in an LLR of 0 at its leaf node under SC decoding when the corresponding channel value is also 0. In other words, a channel value indicating an erasure must propagate the erasure to the associated leaf node, as depicted in Fig. 3.1. There are two simple patterns that can be used to build valid puncture sets. The first pattern involves simply adding the first P indices to $\mathcal{P}$ as in Fig. 3.1a, which was proposed in [12]. The second pattern, visualized in Fig. 3.1b, similarly punctures the first P indices in bit-reversed (BR) order [28].

Once the puncturing set is determined, those indices are added to the frozen set, and removed from $\mathcal{R}$, *ie.* $\mathcal{P} \subset \mathcal{F}$, $\mathcal{R} = [0, N) \setminus \mathcal{P}$. This step will be further referred to as the pre-freezing step. $\mathcal{I}$ is determined by selecting the first K indices remaining in $\mathcal{R}$. Encoding is performed with the mother code, and the punctured indices are not transmitted with the rest of the codeword. $\mathcal{P}$ is known at the decoder, and an LLR of 0 is injected into the decoder at punctured channel indices:

$$y_i = 0, \quad \forall\, i \in \mathcal{P}. \tag{3.1}$$

Decoding is then performed on the mother code in the usual way.

Determining $\mathcal{R}$ can take into account the punctured indices, since puncturing can affect the reliability of non-punctured indices, though it is not required. In order to consider the effect of punctured bits when using GA construction, simply set punctured indices to 0 rather than $z_1^{(0)}$ before carrying out the algorithm. Other construction algorithms such as Bhattacharyya parameter expansion can be modified in a similar manner. This puncturing scheme is called quasi-uniform puncturing (QUP) [12]. QUP has only been formally defined for using the *First* puncturing pattern, though the method could easily be adapted for the BR formation. All future uses of QUP in this work refer to a code that punctures the first P indices. QUP produces an information set that is optimized for the puncturing set, though it requires that the information set be recomputed for every possible code length.

Alternatively, reliability set design can neglect the information provided by the puncturing set to varying degrees of success [28]. Using this method, the reliability order is determined in the usual way for the mother code. While this method has no bearing on optimality of the information set, it only requires computing reliability once for the mother code so that it can be used for any desired punctured polar code length. This low-complexity method will later be referred to as either *First* or BR, depending on the puncturing pattern used.

Although puncturing can be used to attain any block length, it was noted in [11] that puncturing techniques generally have poor performance at high rates. This observation is a based on the fact that high rates imply that $\mathcal{P}$ is more likely to contain useful indices in it. The systematic encoder defined in Section 2.2.2 is compatible with all outlined puncturing schemes since both the *First* and BR puncturing patterns allow $\mathcal{I}$ to remain dominant contiguous.

3.1.2 Shortening

Shortening schemes operate on a different principle than puncturing schemes, but their implementations are very similar. Building a shortened polar code $\mathcal{PC}(E, K)$ of arbitrary length E requires a mother code of length $N = 2^{\lceil log_2 E \rceil}$. A valid shortening set $\mathcal{S} \subset [0, N)$ is comprised of $S = N - E$ *shortened* indices that must be *overcapable*. An overcapable index is one whose codeword value is "0" when its uncoded value is also "0", as outlined in Fig. 3.2. As with puncturing, there are two useful shortening patterns that can be employed. One shortening pattern simply adds the last S indices to $\mathcal{S}$ [13], as depicted in Fig. 3.2a. The second shortening pattern shortens the last S indices in BR order [28], as in Fig. 3.2b. These patterns will be referred to as the *Last* and BR shortening schemes from this point on.

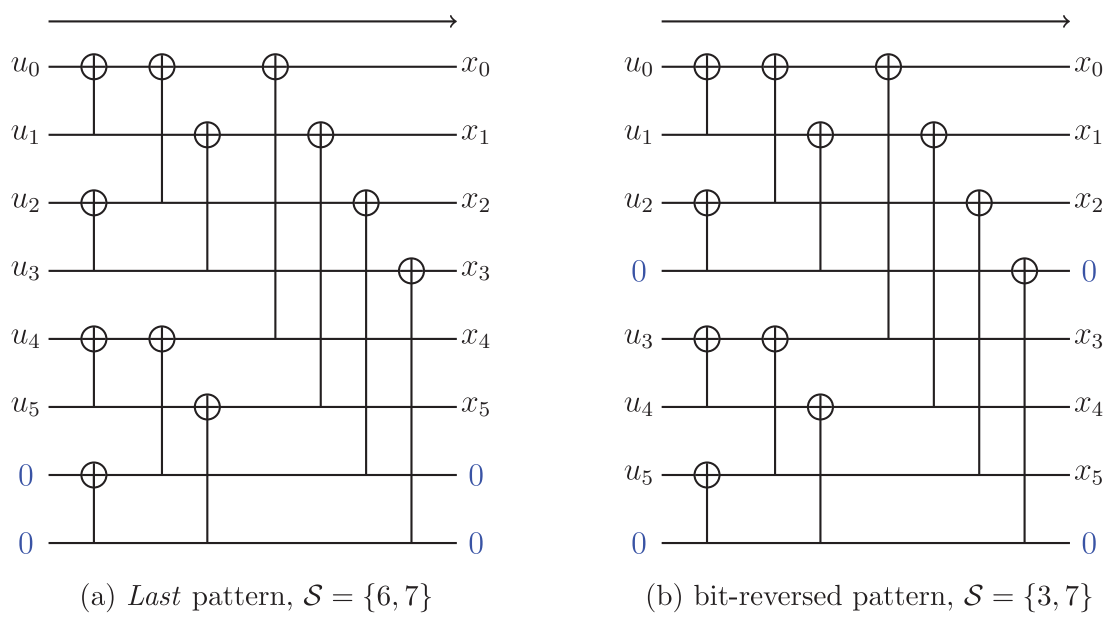


Figure 3.2: Shortening patterns for a polar code with $E = 6$.

After the shortening set is decided, all shortened indices are pre-frozen in the same way as with puncturing. The mother code is encoded, and the shortened indices are not transmitted. $\mathcal{S}$ is known at the decoder, and all channel values corresponding to shortened indices are set to infinity since those codeword bits are known to be zero:

$$y_i = \infty, \quad \forall\ i \in \mathcal{S}. \tag{3.2}$$

The mother code can then be decoded. Just as with puncturing, computing $\mathcal{R}$ for a shortened polar code can take into account the effect that shortening has on the reliability of transmitted bits. To perform GA that takes shortening into account, set all indices in $\mathcal{S}$ to ∞ before computing the reliabilities, as proposed in [13]. This method will be referred to as the Wang-Liu (WL) method, which exclusively uses the *Last* shortening pattern. It is also possible to ignore the effect of shortening on index reliability, as suggested in [28], and simply employ *Last* or BR formations with the natural reliability order of the mother code. Again, this method has no guarantee of optimality, but allows for a single reliability construction for all code lengths. All later references to *Last* or BR in this work refer to this low complexity method.

Shortening is observed to have inadequate performance at low rates [11]. In low coding rate scenarios, only a few information bits are required. In the case of shortening, the most reliable bits are shortened and not available for the data transmission, which impairs the quality of the polar code. Regarding systematic shortened polar codes, it was noted that both the *Last* and BR shortening pattern are compatible with the encoding technique described in Section 2.2.2 [18].

3.2 Multi-Kernel Polar Codes

Introduced in [14], multi-kernel (MK) polar codes, or sometimes mixed-kernel polar codes, are an alternative construction that offers increased native length flexibility over Arıkan's definition. Traditional polar codes, now referred to as Arıkan polar codes, are limited to block lengths that are powers of 2. This limitation is the result of the recursive Kronecker expansion of the 2×2 polarizing matrix $\boldsymbol{T_2}$, further referred to as the Arıkan kernel. Gabry et al proposed the possibility of obtaining polar codes using kernels with alternate dimensions. Specifically, the 3×3 polarizing matrix was introduced:

$$\boldsymbol{T_3} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}. \tag{3.3}$$

This new matrix, referred to as the *ternary* kernel, was shown to be optimal for polarization [29], although it has a polarization exponent that is less than that of $\boldsymbol{T_2}$. $\boldsymbol{T_3}$ can be used as a Kronecker product constituent in conjunction with the Arıkan kernel to produce any polar code with length $N = 2^n 3^m$ where $n, m \in \mathbb{N}^0$. The number of terms in the Kronecker product is equal to $M = n + m$. Table 3.1 outlines the increased block length flexibility that MK polar codes offer by varying m and n for $M \in [1, 10]$ to list all possible lengths. Values in bold indicate block lengths where $32 \leq N \leq 1024$, which are possible use cases of 5G polar codes. Further, the table points out that with a maximum of 10 stages, Arıkan polar codes can only attain 10 lengths, where as MK polar codes using $\boldsymbol{T_2}$ and $\boldsymbol{T_3}$ can attain 65.

3.2.1 Multi-Kernel Encoding

With the addition of improved block length flexibility, there must be further consideration for the order of the kernels in the Kronecker product that defines $\boldsymbol{G}$. For example, a polar code of length $N = 6$ could be encoded with either $\boldsymbol{T_2} \otimes \boldsymbol{T_3}$ or $\boldsymbol{T_3} \otimes \boldsymbol{T_2}$, which are two unique generator matrices, depicted in Fig. 3.3. For clarity, we can refer to a kernel vector k that stores the sizes of the kernels in order pertaining to the generator matrix,

$M=1$	$M=2$	$M=3$	$M=4$	$M=5$	$M=6$	$M=7$	$M=8$	$M=9$	$M=10$
2	4	8	16	**32**	**64**	**128**	**256**	**512**	**1024**
3	6	12	24	**48**	**96**	**192**	**384**	**768**	1536
	9	18	**36**	**72**	**144**	**288**	**576**	1152	2304
		27	**54**	**108**	**216**	**432**	**864**	1728	3456
			81	**162**	**324**	**648**	1296	2592	5184
				243	**486**	**972**	1944	3888	7776
					729	1458	2916	5832	11664
						2187	4374	8748	17496
							6561	13122	26244
								19682	39366
									59048

Table 3.1: List of block lengths attainable by MK polar codes using $\boldsymbol{T_2}$, $\boldsymbol{T_3}$.

ie. $k_i \in \{2,3\}\ \forall\ i \in [0,M)$. The generator matrix can then be defined as:

$$\boldsymbol{G} = \bigotimes_{i=0}^{M-1} \boldsymbol{T_{k_i}}. \tag{3.4}$$

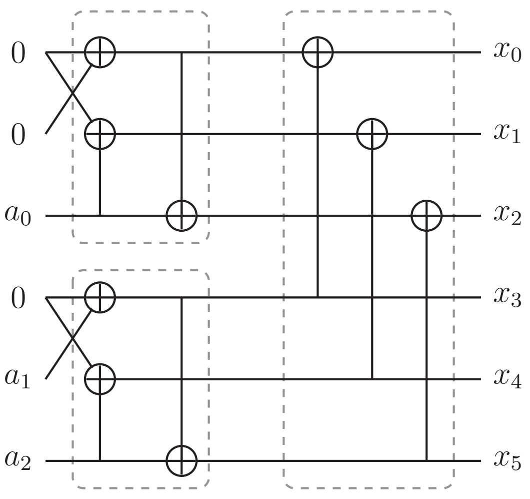
Observe that the kernel order in a MK factor graph is reversed from that of the the Kronecker product. Further, additional kernels of size higher than 3 have been proposed [30] [31] and proven [29], although the Arıkan and ternary kernels are the most common [15] and least complex to use. As such, only MK polar codes derived from these two kernels will be inspected.

3.2.2 Multi-Kernel Decoding

Decoding of MK polar codes can be accomplished using SC [14]. Just as with Arıkan polar codes, the encoding factor graph is restructured into a tree with M stages, which visualizes the SC algorithm. MK SC involves a tree search with left-to-right branch priority, where the leaf nodes in the tree represent the estimated sourceword $\hat{u}$. The top of the tree acts as the decoder input, which is the received noisy channel vector y in the form of LLRs.

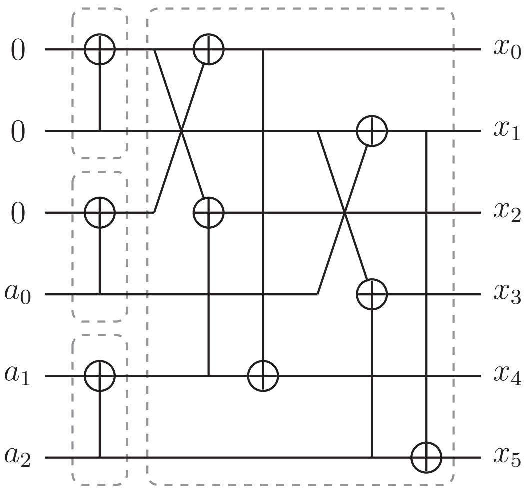
Beginning from the top of the tree, the branches are traversed by applying LLR transformations and storing the results in the proceeding stage. Each stage $s \in (0, M]$ pertains to either the Arıkan or ternary kernel and contains $\frac{N}{p}$ nodes, which represent groups of polar transforms emphasized in Fig. 3.3, where $p = \prod_{i=0}^{s-1} k_{M-1-i}$. At the bottom stage

$$G = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$



(a) $\boldsymbol{G} = \boldsymbol{T_2} \otimes \boldsymbol{T_3}$

$$G = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

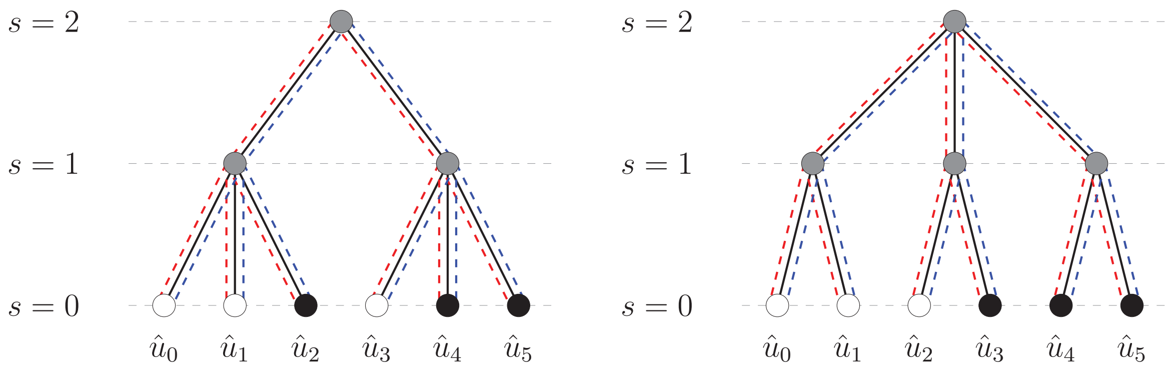


(b) $\boldsymbol{G} = \boldsymbol{T_3} \otimes \boldsymbol{T_2}$

Figure 3.3: Factor graphs for two unique multi-kernel polar codes with $N = 6$.

of the tree (*ie.* $s = 0$), $p = 1$. Each node v in stage s stores both p LLRs and bit partial sums and invokes $\frac{p}{2}$ or $\frac{p}{3}$ transformations upon entering, depending on the kernel of stage $s-1$. If the stage s pertains to the Arıkan kernel, the functions f or g, found in eq. 2.32, are applied to the left and right branches, respectively. In the case that stage s corresponds to the ternary kernel, the left, center, and right children LLRs, α_l, α_c, and α_r, are computed using functions λ_0, λ_1, and λ_2, respectively:

$$\begin{aligned} \alpha_l^i &= \lambda_0(\alpha_v^i, \alpha_v^{i+\frac{N_v}{3}}, \alpha_v^{i+\frac{2N_v}{3}}) = \alpha_v^i \boxplus \alpha_v^{i+\frac{N_v}{3}} \boxplus \alpha_v^{i+\frac{2N_v}{3}}, \\ \alpha_c^i &= \lambda_1(\alpha_v^i, \alpha_v^{i+\frac{N_v}{3}}, \alpha_v^{i+\frac{2N_v}{3}}, \beta_l^i) = (-1)^{\beta_l^i} \cdot \alpha_v^i + \alpha_v^{i+\frac{N_v}{3}} \boxplus \alpha_v^{i+\frac{2N_v}{3}}, \qquad \forall\, i \in [0, \frac{N_v}{3}), \\ \alpha_r^i &= \lambda_2(\alpha_v^{i+\frac{N_v}{3}}, \alpha_v^{i+\frac{2N_v}{3}}, \beta_l^i, \beta_c^i) = (-1)^{\beta_l^i} \cdot \alpha_v^{i+\frac{N_v}{3}} + (-1)^{\beta_l^i \oplus \beta_c^i} \cdot \alpha_v^{i+\frac{2N_v}{3}}, \end{aligned} \tag{3.5}$$


$s = 2$
$s = 1$
$s = 0$
$\hat{u}_0$ $\hat{u}_1$ $\hat{u}_2$ $\hat{u}_3$ $\hat{u}_4$ $\hat{u}_5$

(a) $\boldsymbol{G} = \boldsymbol{T_2} \otimes \boldsymbol{T_3}$, $\mathcal{I} = \{3, 4, 5\}$ (b) $\boldsymbol{G} = \boldsymbol{T_3} \otimes \boldsymbol{T_2}$, $\mathcal{I} = \{2, 4, 5\}$

Figure 3.4: SC tree for $\mathcal{PC}(6, 3)$.

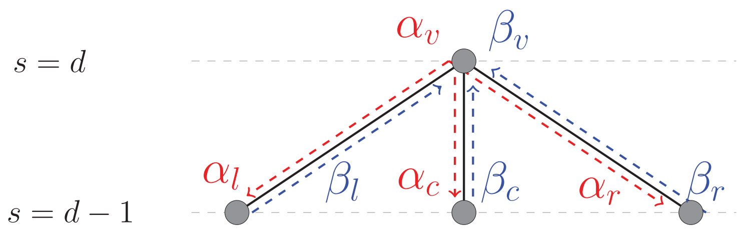


Figure 3.5: Mk SC decoding rules are dictated by left-to-right priority tree traversal.

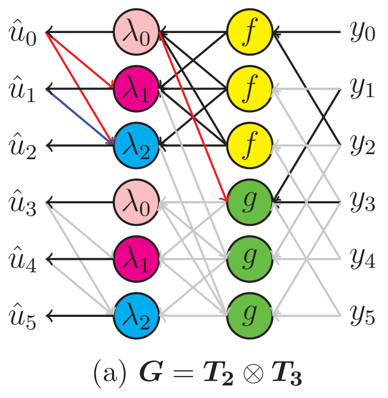
Hard decisions are applied to leaf nodes in the usual way. After returning from a right branch, p partial sum updates are executed at the previous node before moving down another branch. For Arıkan stages, eq. 2.34 is applied. For a ternary stage, instead apply:

$$\begin{aligned} \beta_v^i &= \beta_l^i \oplus \beta_c^i, \\ \beta_v^{i+\frac{N_v}{3}} &= \beta_l^i \oplus \beta_r^i, \qquad\qquad \forall\ i \in [0, \frac{N_v}{3}), \\ \beta_v^{i+\frac{2N_v}{3}} &= \beta_l^i \oplus \beta_c^i \oplus \beta_r^i, \end{aligned} \tag{3.6}$$

An SC decoder can be expressed as schedule of f, g, λ_0, λ_1, and λ_2 operations, where the total number of operations described by MN. Fig. 3.4 depicts SC trees for both kernel configurations for $N = 6$, and Fig. 3.6 depicts their corresponding schedules. A SCL decoding can also be performed on MK codes by employing the same method outlined in Section 2.3.2.

3.2.3 Multi-Kernel Information Set Design

The following section assumes that the order of the kernels and generator matrix are known to the designer. Kernel order optimization will be discussed in Section 4.1, although this chapter can assume that the kernel order is designed for highest overall code



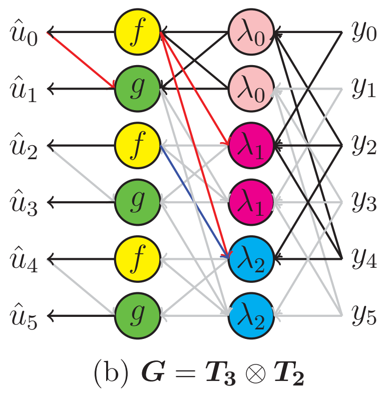


Figure 3.6: SC decoder expressed as a schedule of LLR operations for $N = 6$.

reliability.

3.2.3.1 Bhattacharyya Parameter Expansion

MK polar codes can have their reliability ordering determined with Bhattacharyya parameter expansion just as with Arıkan polar codes. The overall algorithm is the same, but there lies a change in computing ternary stages. Different formulas must be used for computing the Bhattacharyya parameter for ternary stages. These equations can be easily derived by noticing that eq. 3.5 can be expressed in terms of eq. 2.32:

$$\begin{aligned} \lambda_0(a,b,c) &= a \boxplus b \boxplus c = f(a, f(b,c)), \\ \lambda_1(a,b,c,d) &= (-1)^d \cdot a + b \boxplus c = g(a, f(b,c), d), \\ \lambda_2(a,b,c,d) &= (-1)^c \cdot a + (-1)^{c\oplus d} \cdot b = g(a, (-1)^{c\oplus d} \cdot b, c). \end{aligned} \tag{3.7}$$

Specifically, the SC LLR equations for a ternary stage can be rewritten as nested combinations of f and g operations. This pattern can be exploited to rewrite the reliability construction algorithms for ternary polar codes. As such, eq. 2.23 can be modified accordingly:

$$\begin{aligned} Z(W_N^{(3j-2)}) &= 3Z(W_{\frac{N}{3}}^{(j)}) - 3Z(W_{\frac{N}{3}}^{(j)})^2 + Z(W_{\frac{N}{3}}^{(j)})^3, \\ Z(W_N^{(3j-1)}) &= 2Z(W_{\frac{N}{3}}^{(j)})^2 - Z(W_{\frac{N}{3}}^{(j)})^3, \\ Z(W_N^{(3j)}) &= Z(W_{\frac{N}{3}}^{(j)})^2, \end{aligned} \tag{3.8}$$

$\mathcal{R}$ can then be sorted according to $Z(W_N)$, and the first K bits can be selected for $\mathcal{I}$.

3.2.3.2 Gaussian Approximation

The GA algorithm outlined in Section 2.2.3.2 can be adapted in a similar manner to the Bhattacharyya modification in the previous section. For stages of the polar code corresponding to an Arıkan kernel, use eq. 2.24. For computing GA for a ternary stage, use the following modified formulas:

$$\begin{aligned} z_N^{(3j-2)} &= \phi^{-1}\left(1-\left(1-\phi(\phi^{-1}(1-(1-\phi(z_{\frac{N}{3}}^{(j)}))^2))\right)(1-\phi(z_{\frac{N}{3}}^{(j)}))\right), \\ z_N^{(3j-1)} &= \phi^{-1}(1-(1-\phi(z_{\frac{N}{3}}^{(j)}))^2)+z_{\frac{N}{3}}^{(j)}, \\ z_N^{(3j)} &= 2z_{\frac{N}{3}}^{(j)}, \end{aligned} \tag{3.9}$$

Then, select $\mathcal{I}$ in the usual way.

MK polar codes can also be constructed for minimized codeword Hamming distance [30]. This is a valid construction, although this scheme is known to only be effective for short code lengths, *ie.* $N < 200$. At this time, there exist no proposals for the possibility of nested sequences for MK polar codes.

3.3 3GPP 5th Generation New Radio

This section will outline the length-compatible polar coding schemes that have been devised for industrial use in the control channel for the upcoming 3GPP 5th generation (5G) standard for New Radio (NR). The schemes referenced are detailed in the 3GPP specification [10], which is summarized in [11].

3.3.1 5G Polar Codes System Overview

5G NR features several different use cases. This thesis is primarily focused on the error correction performance and efficiency of length-compatible polar code techniques. As such, this work will limit the scope of the 5G parameters to only include cases that are relevant to the other state-of-the-art length compatible schemes. This work will not consider the performance of uplink scenarios (labeled PUCCH/PUSCH in the literature) since those use cases employ embedded parity checks within the codeword. The parity

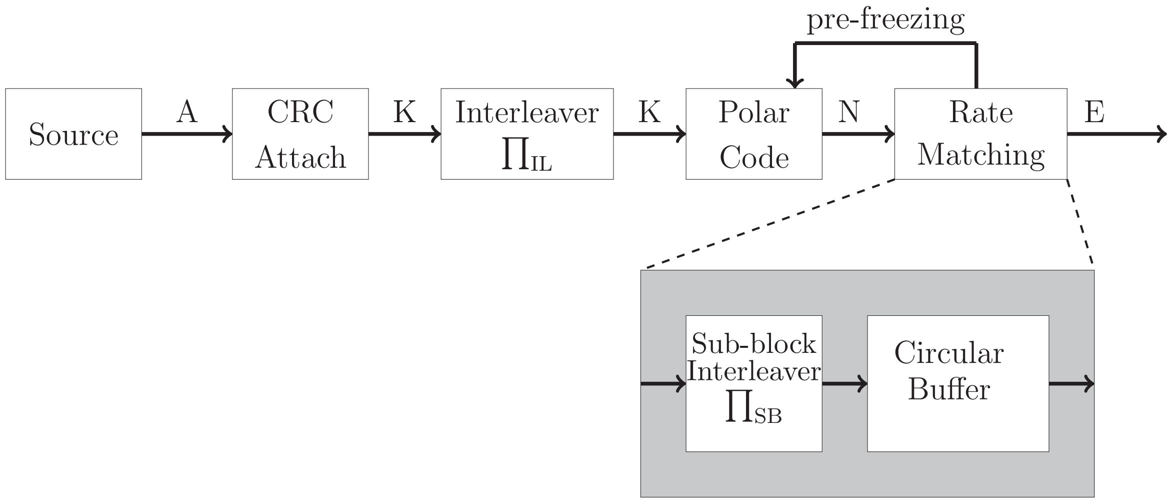

pre-freezing
Source
A
CRC Attach
K
Interleaver Π_{IL}
K
Polar Code
N
Rate Matching
E
Sub-block Interleaver Π_{SB}
Circular Buffer

Figure 3.7: The 5G polar coding scheme for the downlink case.

check bits will not yield a fair comparison of the 5G standard against puncturing, shortening, or multi-kernel schemes due to the added system complexity and potential difference in error correction performance. Since only the downlink scenarios (PDCCH/PBCH) will be studied, only the block components depicted in Fig. 3.7 will be considered.

3.3.2 5G Polar Code Encoding

The encoding chain begins with the source, which is a binary message a of length A. Next, a CRC is computed on a and appended to form vector c such that the message is now of length $K = A+C$, where $C = 24$ in the downlink case. c is then placed through an input bit interleaver Π_{IL} to obtain vector c'. The interleaver is implemented as a lookup table, where the table is defined in Table 5.3.1.1-1 in the specification [10].

Before c' can be encoded into a polar code of $\mathcal{PC}(E, K)$, several parameters must be specified. First, the size of the mother code N must be determined using Algorithm 1. From this point, the rate-matching scheme needs to be selected in order to perform the pre-freezing step. If $E \leq N$ and $R \leq \frac{7}{16}$, then puncturing is employed. Alternatively, if $E \leq N$ and $R > \frac{7}{16}$, then shortening is used instead. This corroborates the suggested use cases in Sections 3.1.1 and 3.1.2: puncturing and shortening are effective for low rates and high rates, respectively. However, there is the possibility that $E > N$, in which case a repetition rate-matching scheme is used. For shorthand, the puncturing set, shortening set, and repetition set, will all be represented interchangeably with a rate-matching set $\mathcal{RM} \subset \mathcal{F}$. The rate-matching set is not determined with the typical *First*, *Last*, or

BR patterns, but instead a customized pattern dictated by the sub-block interleaver Π_{SB} defined in eq. 3.10.

```
Algorithm 1: Determine the size of the mother code.
  Input  : $E, K$
  Output: $N$
1  $n_{\min} = 5$;
2  if uplink then
3      $n_{\max} = 10$;
4  else
5      $n_{\max} = 9$;
6  if $E \le \frac{9}{8}(2^{\lfloor \log_2 E \rfloor})$ AND $\frac{K}{E} < \frac{9}{16}$ then
7      $n_1 = \lfloor \log_2 E \rfloor$;
8  else
9      $n_1 = \lceil \log_2 E \rceil$;
10 $n_2 = \lceil \log_2 8K \rceil$;
11 $n = \max(\min(n_1, n_2, n_{\max}), n_{\min})$;
12 $N = 2^n$;
13 return $N$
```

$$\Pi_{\text{SB}}(j) = i, \quad \forall\, j \in [0, N), \tag{3.10}$$

where

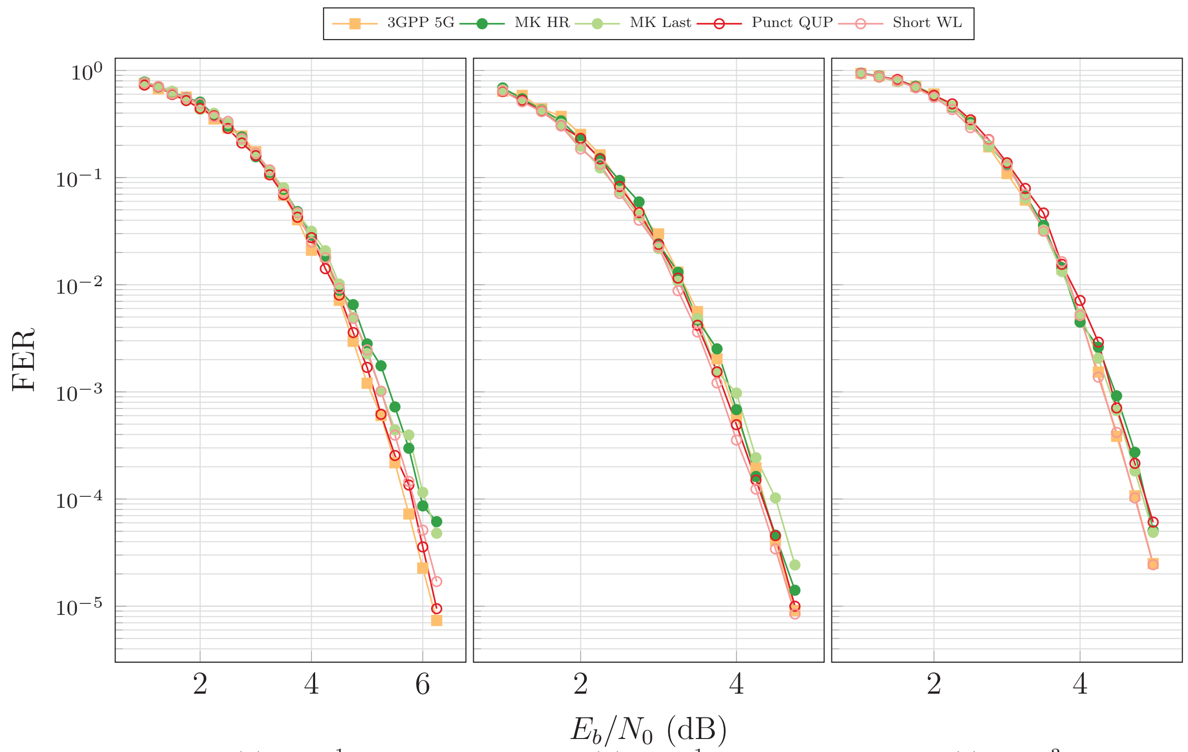
$$i = B \cdot P\left\lfloor \frac{j}{B} \right\rfloor + q, \tag{3.11}$$

and $B = \frac{N}{32}$, $q = j \bmod B$. Evidently, the indices to be pre-frozen are the first or last $N - E$ indices after interleaving such that $\Pi_{\text{SB}}(k) \subset \mathcal{F}, \mathcal{RM} \;\; \forall\, k \in [\omega, N - E + \omega)$ where $\omega = 0$ when puncturing or repeating, and $\omega = E$ otherwise.

In the case of puncturing, there is an *extra freezing* step to ensure that reliable bits are used for data transmission. The indices that are frozen in this step are all that satisfy $[0, T) \subset \mathcal{F}$ where:

$$T = \begin{cases} \left\lfloor \frac{3}{4}N - \frac{E}{2} \right\rfloor & \text{if } E \ge \frac{3}{4}N, \\ \left\lfloor \frac{9}{16}N - \frac{E}{4} \right\rfloor & \text{otherwise.} \end{cases} \tag{3.12}$$

Finally, the first K non-frozen indices may be selected from the nested sequence presented in Table 5.3.1.2-1 of [10] to insert into $\mathcal{I}$. If $N < 1024$, which is the case during downlink, then a reliability sequence for the appropriate block length is extracted using the technique described in Section 2.2.3.3. As usual, the remaining indices are added to



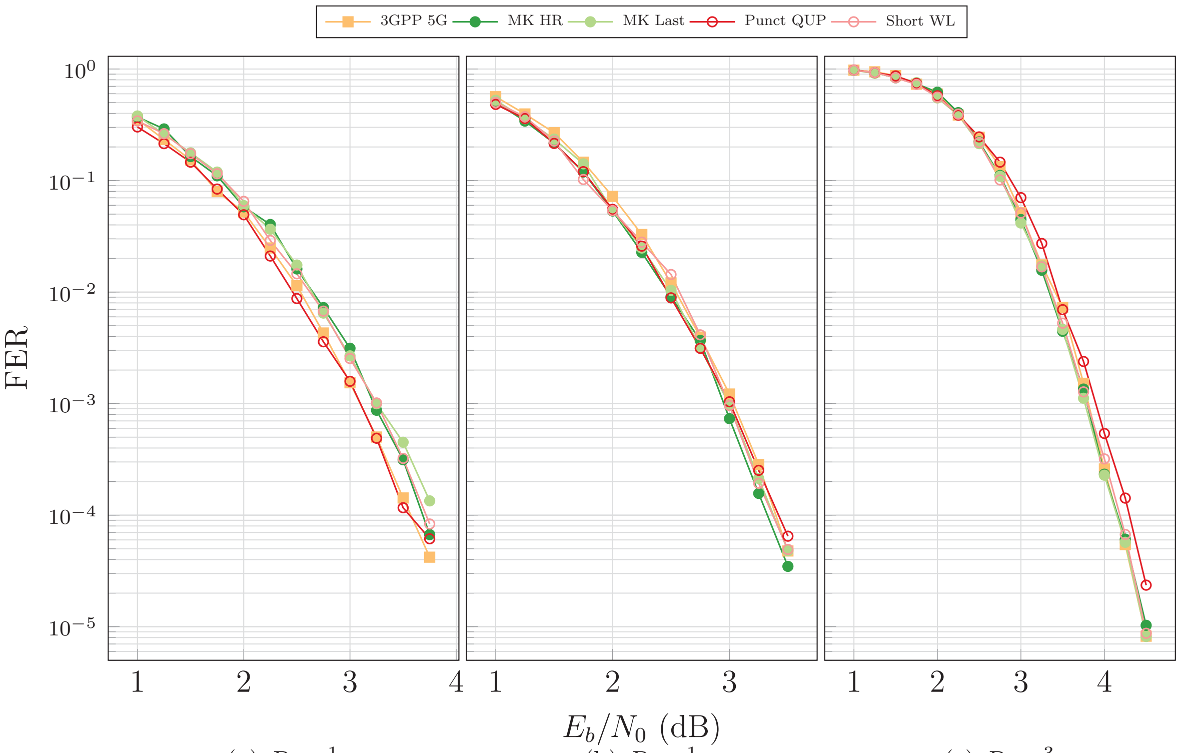
(a) $R = \frac{1}{4}$ (b) $R = \frac{1}{2}$ (c) $R = \frac{3}{4}$

Figure 3.8: CA-SCL, L=8, C=24, $E = 192$, $R = \{\frac{1}{4}, \frac{1}{2}, \frac{3}{4}\}$.

$\mathcal{F}$. Vector c' is then expanded into u and the mother polar code can then be encoded into x. The codeword is next sent to the circular buffer.

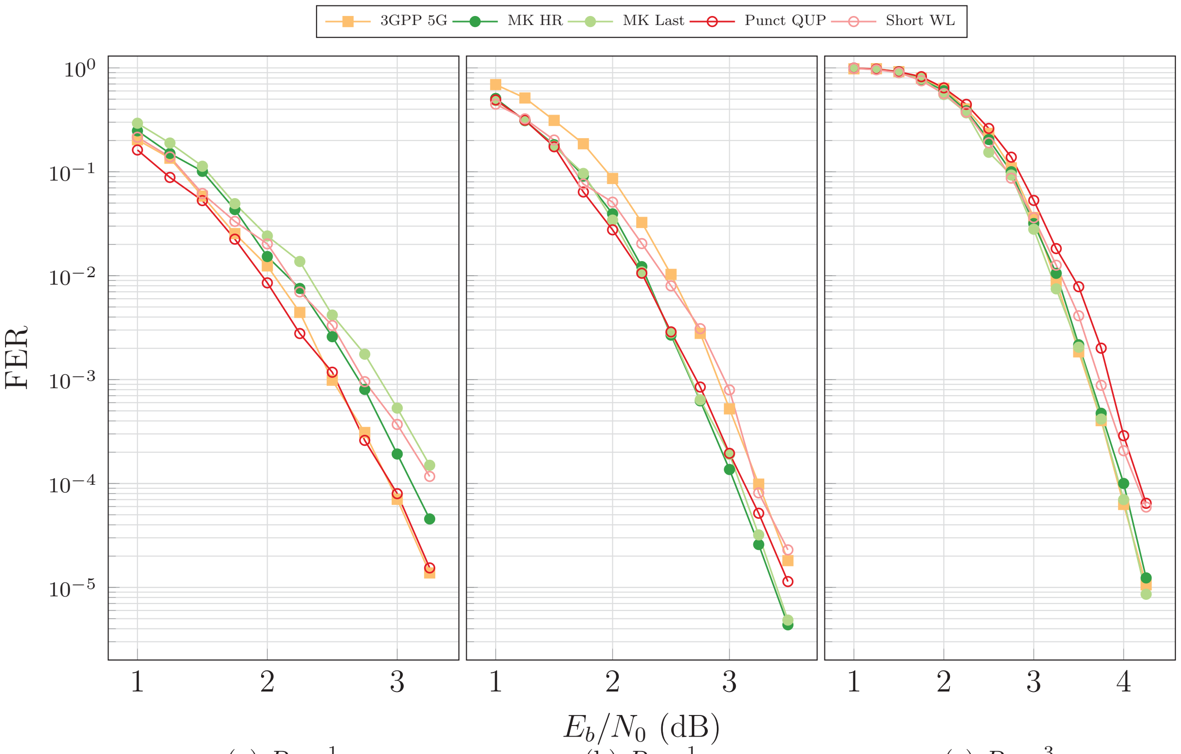
The circular buffer is the device that chooses which indices are transmitted across the channel. If puncturing or shortening is used, all indices in $\mathcal{RM}$ are not transmitted. However, if repetition is the elected rate-matching scheme, then all indices in $\mathcal{RM}$ are transmitted twice. On the receiver side, LLRs corresponding to punctured and shortened indices are set to 0 and ∞, respectively. Repeated indices sum their multiple LLRs together before decoding. Decoding is then performed using the mother code. It is preferable to employ a decoder that takes advantage of the appended CRC, such as CA-SCL.

For the remainder of this work, all simulations of 5G polar codes will neglect the input bit interleaver Π_{IL}, which has a maximum input length of $K = 164$. This omission will allow for an unconstrained message length K, and a fair comparison of length-compatible polar code schemes.



(a) $R = \frac{1}{4}$ (b) $R = \frac{1}{2}$ (c) $R = \frac{3}{4}$

Figure 3.9: CA-SCL, L=8, C=24, $E = 384$, $R = \{\frac{1}{4}, \frac{1}{2}, \frac{3}{4}\}$.



(a) $R = \frac{1}{4}$ (b) $R = \frac{1}{2}$ (c) $R = \frac{3}{4}$

Figure 3.10: CA-SCL, L=8, C=24, $E = 576$, $R = \{\frac{1}{4}, \frac{1}{2}, \frac{3}{4}\}$.

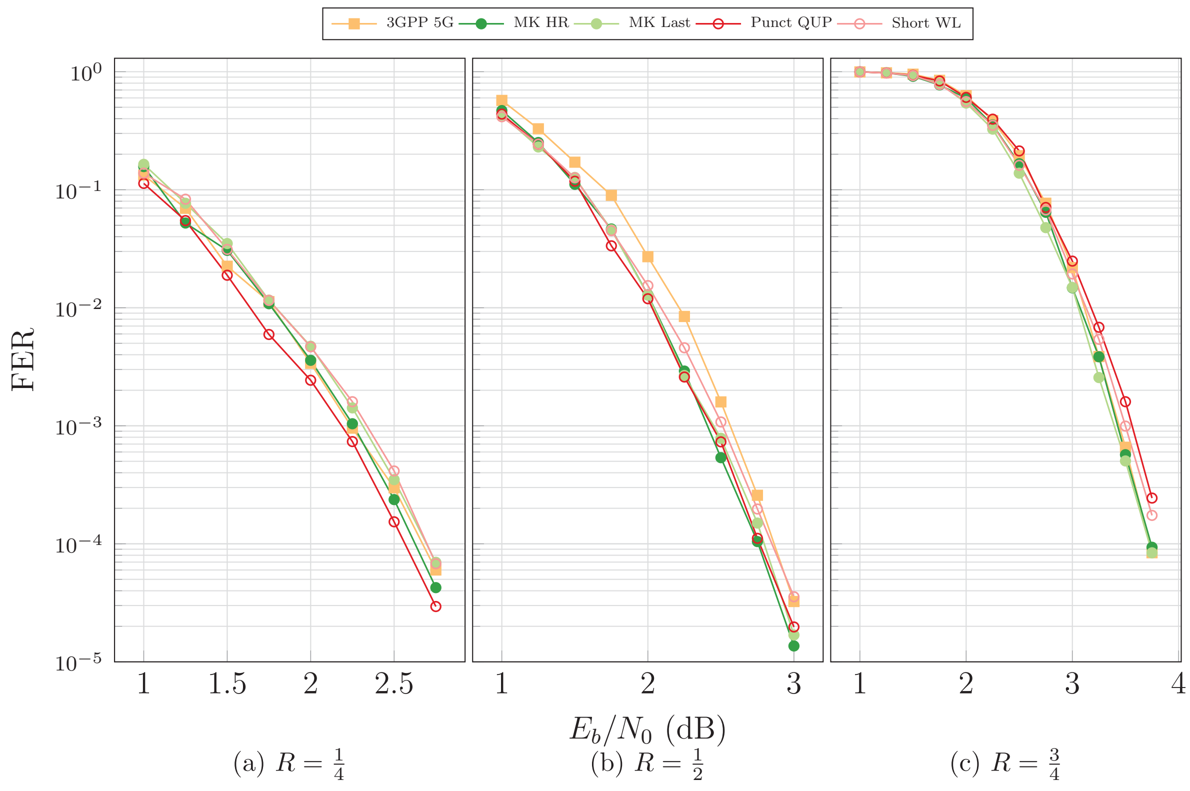


Figure 3.11: CA-SCL, L=8, C=24, $E = 768$, $R = \{\frac{1}{4}, \frac{1}{2}, \frac{3}{4}\}$.

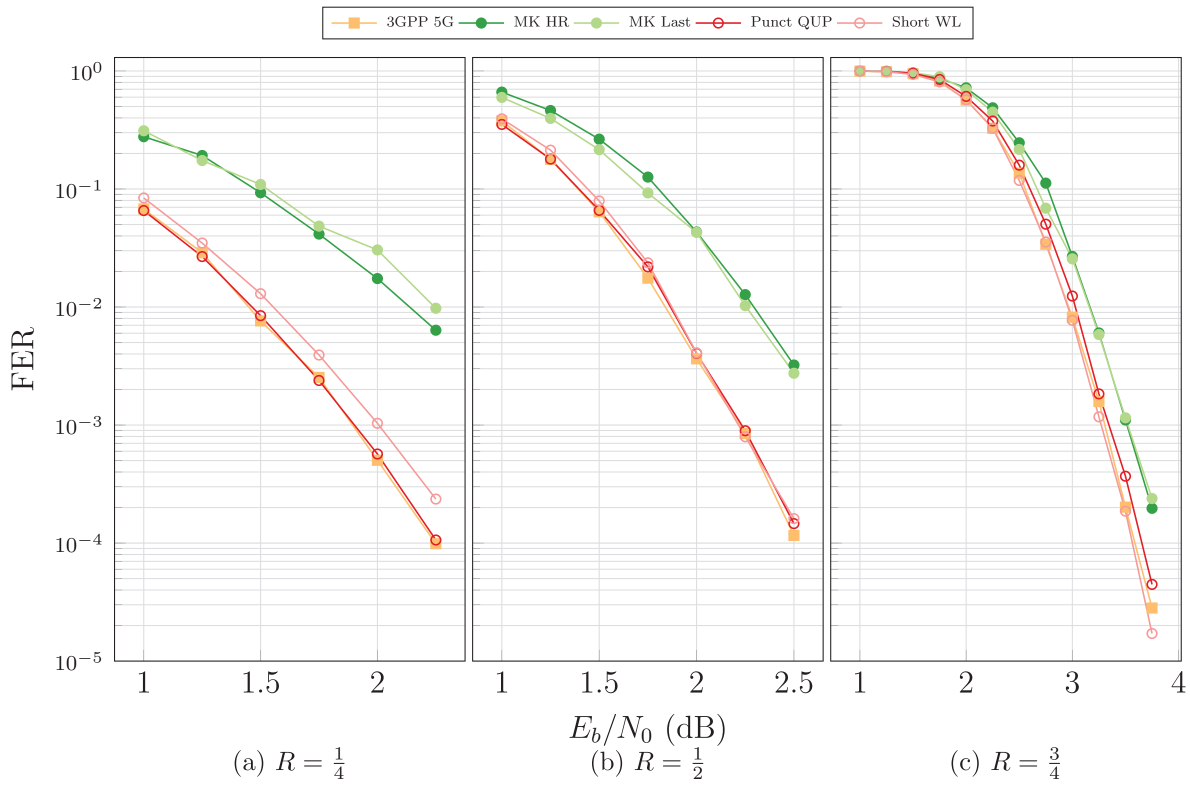


Figure 3.12: CA-SCL, L=8, C=24, $E = 972$, $R = \{\frac{1}{4}, \frac{1}{2}, \frac{3}{4}\}$.

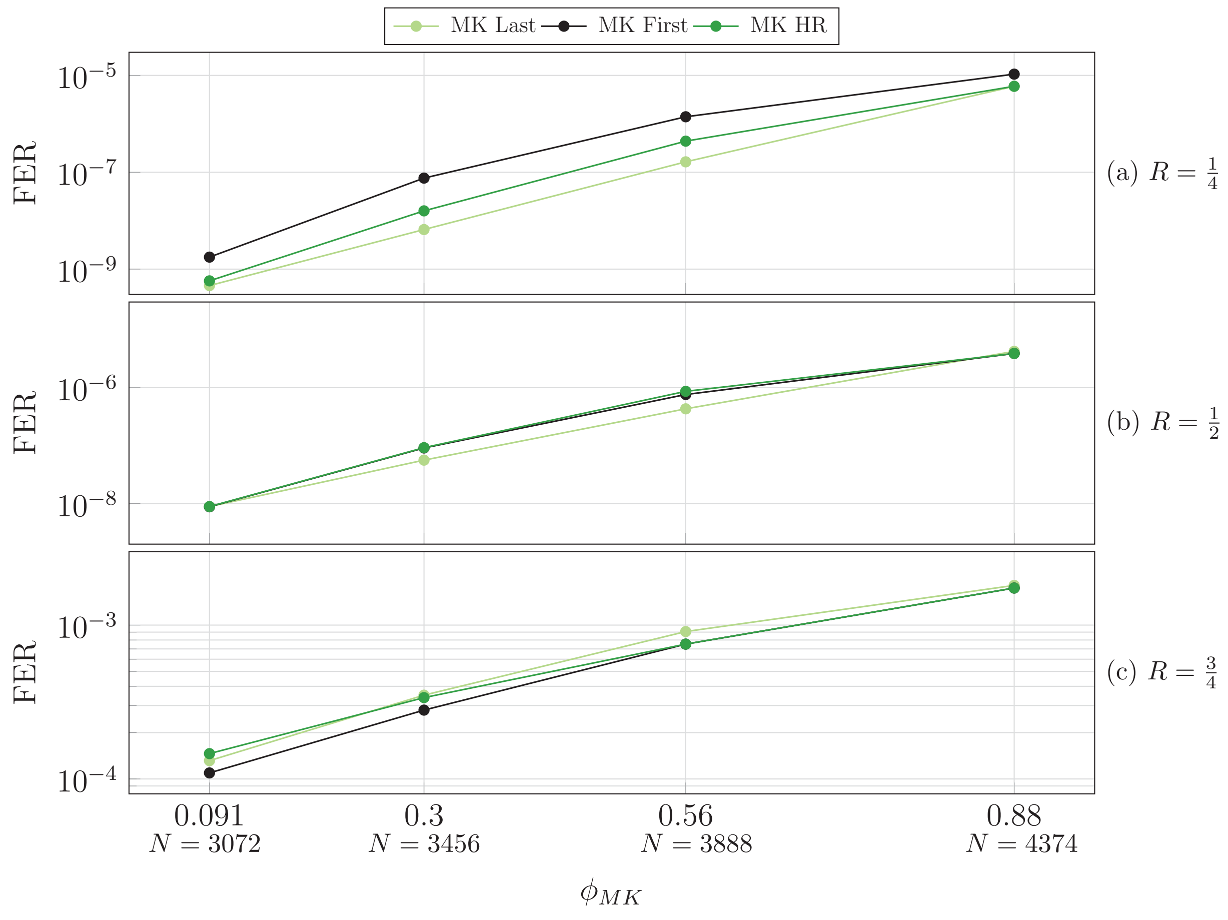


Figure 3.13: The effect of the ternary kernel on error correction performance for $N \approx 3723 \pm 18\%$.

3.4 Simulations

This section presents simulation data that test the error correction performance of the length-compatible polar coding schemes detailed in this chapter. Although puncturing and shortening schemes can attain any arbitrary length, the scope of tested cases must accommodate the limitations of 5G polar codes and MK polar codes. As such, the simulation campaign encompassed a range of block lengths and rates that exhibit the strengths and weaknesses of the studied coding schemes. The examined coding schemes, are the 3GPP 5G standard, puncturing using QUP, shortening using WL, and MK polar codes. "MK HR" refers to codes constructed such that their kernel orders were chosen for highest overall reliability, as suggested in [14], and "MK Last" indicates codes constructed with ternary kernels as the last constituents in the Kronecker product. The decoder used is CA-SCL with a list size of 8 and a CRC size of 24. The code lengths tested are $N = \{192, 394, 576, 768, 972\}$ with rates $R = \{\frac{1}{4}, \frac{1}{2}, \frac{3}{4}\}$. All information sets are designed for each plot point using GA.

3.5 Analysis

For the purposes of discussion, we will introduce two new metrics with which to measure degradation against Arıkan polar codes. These metrics can serve as an indicator of the magnitude of compromise that the considered scheme needs to make in order to achieve non-standard code lengths. For puncturing and shortening, it will be useful to consider the proportion of the mother code length N that is discarded to attain the desired length E. This will be called the *rate-matching ratio*, $\phi_{RM} = \frac{N-E}{N}$. For MK polar codes, the fraction of ternary stages, $\phi_{MK} = \frac{m}{M}$, will be referred to as the *ternary ratio* and may serve as an effective measurement of deterioration.

In most simulated conditions all considered schemes have comparable error correction performance. We will then investigate the circumstances where some schemes are superior. From these simulations, there are several notable observations. The 5G polar codes often have among the best error correction performance of all the presented schemes in most cases. 5G polar codes have near identical performance to either punctured or shortened polar codes, which is to be expected since typically the underlying schemes are very similar. The literature on rate-matching techniques acknowledges that shortening tends to outperform puncturing in high rates, while the opposite is true for low rates. From these simulations, these characteristics can indeed be confirmed to be true. For $R = \frac{1}{2}$, puncturing and shortening perform similarly. An interesting result can be found in the simulations for $N = 576$: this code length suggests that puncturing and shortening will suffer in comparison to MK codes since a significant portion of the code is discarded, *ie.* $\phi_{RM} \approx 0.44$. However, the 5G scheme outperforms both QUP and WL for high rates, and outperforms MK codes in low rates. In fact, the 5G polar codes only ever produce inferior results for some medium rates. QUP and WL do deteriorate when $\phi_{RM} \to 0.5$, which occurs when the difference in N and E are largest. This issue lends sufficient justification to the design choices for the complex rate-matching system used for 5G polar codes. The 5G scheme ensures that ϕ_{RM} remains as low as possible to preserve the error correction capabilities of the polar code. When considering that the 5G algorithm uses a nested sequence to determine the information and frozen sets, the simulations indicate little compromise in FER. This impressive result demonstrates a well designed low-complexity error correction scheme for industrial use.

Regarding MK polar codes, two different kernel ordering strategies were selected to indicate the impact of this code design step. From the simulations, it could be argued that optimizing the kernel order is certainly not required. The "MK Last" codes produced consistent results when compared with "MK HR". In most tested cases, MK polar codes had commensurate performance in comparison with the other coding schemes. However, there exist MK constructions that cause an unavoidable performance loss. For $N = 972$, both considered MK kernel order strategies resulted in a FER that lags the puncturing and shortening schemes by at least one order of magnitude. This deficiency can be attributed to multiple factors. Firstly, puncturing and shortening methods are more likely to perform well when $\phi_{RM} \to 0$. Secondly, MK polar codes do not perform well when $\phi_{MK} \to 1$ since it is known that $\boldsymbol{T_3}$ has a reduced rate of polarization compared with $\boldsymbol{T_2}$. This shortcoming suggests that reducing the number of ternary stages in a MK polar code will improve performance. This motion is confirmed by the experiment depicted in Fig. 3.13. These plots were generated using the theoretical FER formula in 2.29. This calculation measures the FER of MK polar codes with an approximate length of $N = 3723$ while varying ϕ_{MK} and maintaining the block length as much as possible. The experiment reveals that increasing the number of ternary stages reduces the error correction performance by multiple orders of magnitude in some extremes. Though the code lengths are not constant, the FER drops as N increases. Polar codes generally improve in error correction performance as block length increases, which suggests that the trend presented in this experiment is quite strong. The evidence points out that "deeper" polar codes, or polar codes with a smaller ϕ_{MK}, will have better error correction performance. These simulations also suggest that using *Last* is optimal for low rates, while using *First* is best for high rates. However, the difference in FER is negligible.

Regarding decoding complexity, MK polar codes with a higher ϕ_{MK} for a given N will have reduced decoding time since there are fewer stages to the decoder. On the other hand, MK polar codes always have reduced decoding time compared with punctured or shortened polar codes since they must always decode using their larger mother code.

Chapter 4

Towards Practical Multi-Kernel Polar Codes

Thus far, MK polar codes have been shown to have similar error correction performance to previously proposed puncturing and shortening schemes while offering reduced decoding complexity [14]. They have also been shown to be feasible for a hardware implementation [15]. However, MK polar codes are still lacking many of the useful features and innovations of Arıkan polar codes, such as simplified code construction, fast decoding, rate-matching, and systematic encoding. This chapter will investigate the viability of these techniques as they apply to MK polar codes in order to maximize their capabilities.

4.1 Kernel Order Optimization

To design a MK polar code, one must first decide the order of the kernels to perform encoding, decoding, and frozen set design. For a given block length N, there exist $\kappa = \frac{M!}{n!m!}$ different possible kernel orders π that make up the set $\mathcal{K}$. In [14], it was suggested that k can be theoretically optimized for error correction performance by searching all permutations for the order that produces the most overall reliable information set. If $\mathcal{R}$ is to be constructed using Bhattacharyya parameters, the kernel order can be chosen to minimize the probability of error of the information set:

$$k = \underset{\pi}{\operatorname{argmin}} \sum_{i} Z(W_N^{(i)}), \quad \forall\, i \in \mathcal{I}_\pi,\ \pi \in \mathcal{K}. \tag{4.1}$$

Alternatively, the kernel order can be selected to maximize reliability of the information set using GA:

$$k = \underset{\pi}{\operatorname{argmax}} \sum_{i} z_N^{(i)}, \quad \forall\; i \in \mathcal{I}_\pi,\; \pi \in \mathcal{K}. \tag{4.2}$$

While these methods are effective, it is not desirable to have this additional optimization over κ possible configurations when comparing the practicality of MK polar codes with rate-matched Arıkan polar codes. Fig. 4.1 demonstrates that for long MK polar codes, the kernel optimization step is largely inconsequential, in particular for high rates. The figure depicts MK polar codes with $N = 786$ and $N = 2304$ sweeping rates $R = \{\frac{1}{4}, \frac{1}{2}, \frac{3}{4}\}$ under SC decoding. The codes were constructed using GA for each point in the plot to ensure that the frozen sets are optimal through the simulation. Three different kernel ordering strategies are investigated. The *First* and *Last* labels indicate that kernels are ordered such that the ternary kernels serve as either the first or last components of the Kronecker product, respectively. ”Highest Reliability” uses the kernel ordering method outlined in the previous paragraph. We can conclude that placing the ternary kernel in the Kronecker product at either the first or last positions produces comparable error correction results against an optimized kernel order for long polar codes. Further, it can observed that low rate codes have better performance using the *Last* method, while the opposite configuration performs best for medium to high rate codes.

4.2 Fast Decoding of Multi-Kernel Polar Codes

4.2.1 Ternary-Compatible Fast Nodes

This section will outline the techniques required to adapt the FSSC decoder described in Section 2.3.3 to MK polar codes that use $\boldsymbol{T_3}$. Just as with FSSC in the purely Arıkan case, the objective behind FSSC for MK codes is to prune the decoding tree in order to alter the schedule and thus reduce latency. This work will revisit the four fast nodes previously discussed and extend them to be compatible with the ternary kernel. While Arıkan polar codes need only consider frozen set patterns when scheduling FSSC, MK polar codes must also consider the added dimension of kernel order. As such, each type of node will be generalized for any kernel ordering. To keep MK polar codes practical, simplified fast node implementations are proposed that are valid under certain code constraints.

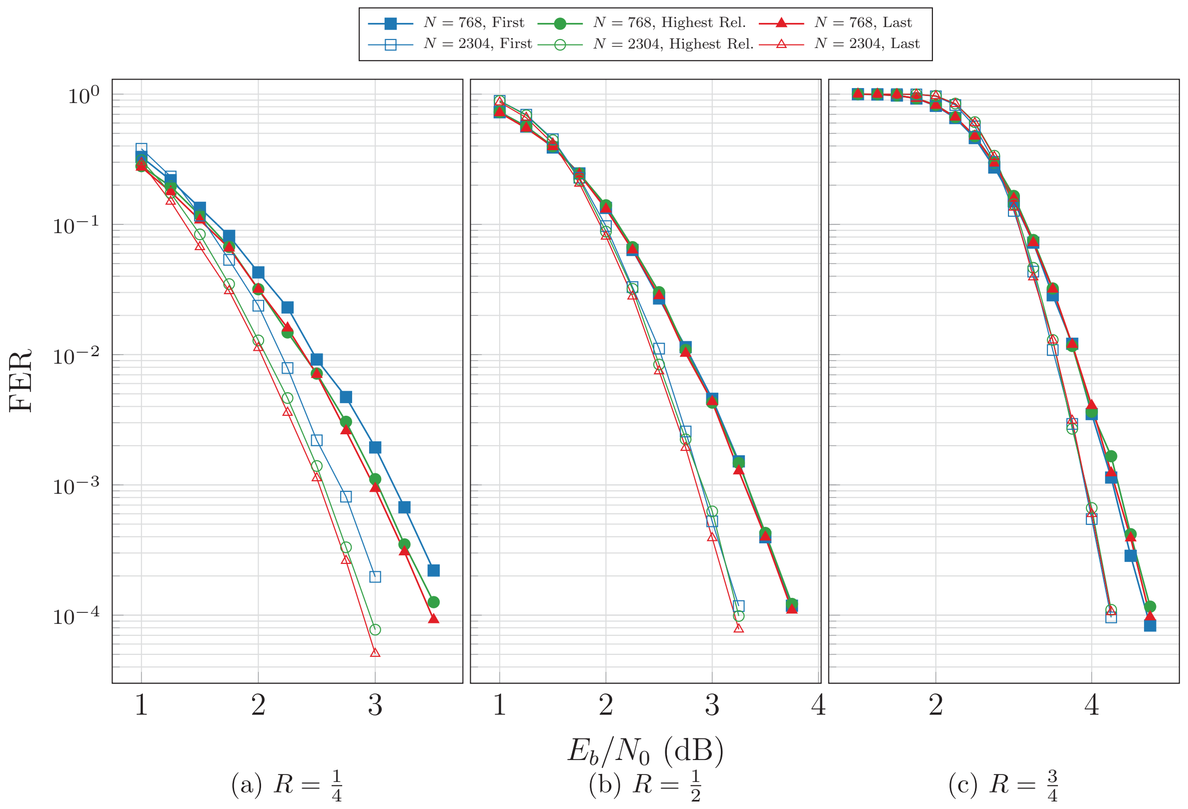


Figure 4.1: FER curves for MK polar codes with $N = 768, 2304$ sweeping rates $R = \{\frac{1}{4}, \frac{1}{2}, \frac{3}{4}\}$ comparing kernel ordering strategies.

Rate-0 Node

Recall that R0 nodes are parent only to frozen indices where leaf nodes and partial sums are known to be 0 [5]. This relationship does not change when considering the ternary kernel since encoding $(0, 0, 0)$ with $\boldsymbol{T_3}$ results in $(0, 0, 0)$. R0 decoding is thus unchanged for $\boldsymbol{T_3}$.

Rate-1 Node

Just as before, R1 nodes are those that are parent only to information indices. A R1 node v can be decoded by evaluating the p partial sums β_v at that stage with a hard decision on α_v in tv and then updating the values of the leaf nodes by hard decoding the partial sums [5]. Specifically, the partial sum vector is obtained by $\beta_v = HD(\alpha_v)$. The estimated sourceword values $\hat{u}_v \subset \hat{u}$ with indices in $\mathcal{I}_v \subset \mathcal{I}$ are hard decoded using $\hat{u}_v = \beta_v \boldsymbol{G}_p^{-1}$ where $\boldsymbol{G}_p$ is the generator matrix obtained by performing the Kronecker product of kernels pertaining to all stages below that of node v. Alternatively, $\hat{u}_v$ may be estimated by propagating the partial sums β_v to the leaf nodes with inverse partial sum equations for each stage where

$$\begin{aligned} \beta_l^i &= \beta_v^i \oplus \beta_v^{i+\frac{N_v}{2}}, \\ \beta_r^i &= \beta_v^{i+\frac{N_v}{2}}, \end{aligned} \qquad \forall\, i \in [0, \frac{N_v}{2}), \tag{4.3}$$

are used for Arıkan stages and

$$\begin{aligned} \beta_l^i &= \beta_v^i \oplus \beta_v^{i+\frac{N_v}{3}} \oplus \beta_v^{i+\frac{2N_v}{3}}, \\ \beta_c^i &= \beta_v^{i+\frac{N_v}{3}} \oplus \beta_v^{i+\frac{2N_v}{3}}, \\ \beta_r^i &= \beta_v^i \oplus \beta_v^{i+\frac{2N_v}{3}}, \end{aligned} \qquad \forall\, i \in [0, \frac{N_v}{3}), \tag{4.4}$$

are to be used for ternary stages.

The following is a proof that verifies this R1 decoding method for stages corresponding to $\boldsymbol{T_3}$. Each R1 node has the property that

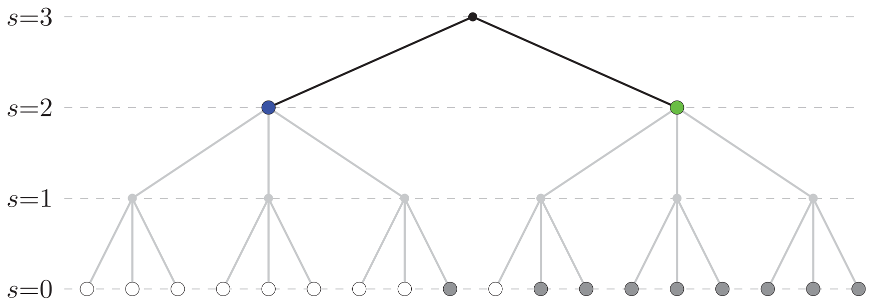
$$\beta_{v_l} = h(\alpha_{v_l}), \beta_{v_c} = h(\alpha_{v_c}), \beta_{v_r} = h(\alpha_{v_r}). \tag{4.5}$$

where α_{v_l}, α_{v_c}, and α_{v_r} are the LLRs in the three branches below node v, as depicted in Fig. 4.2. For shorthand, let $e = 3i$, $o = 3i+1$, and $u = 3i+2$ for all fixed i. Recall that the $\boxplus$ operator has the property

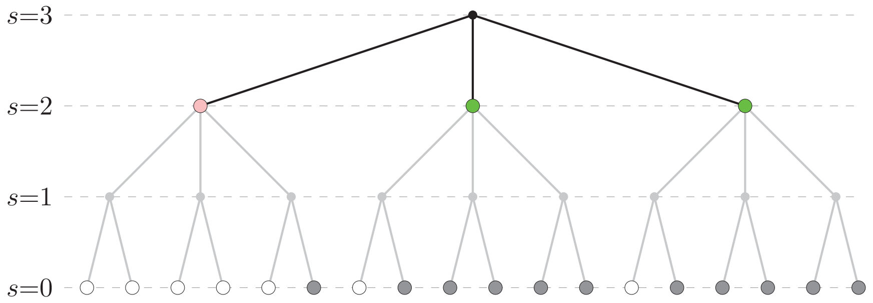
$$HD(a \boxplus b) = HD(a) \oplus HD(b) \text{ if } ab \neq 0. \tag{4.6}$$

And so assuming that $\alpha_v[e] \neq 0$, $\alpha_v[o] \neq 0$, and $\alpha_v[u] \neq 0$ indicates that

$$\begin{aligned} h(\alpha_{v_l}[i]) &= h(\alpha_v[e]) \oplus h(\alpha_v[o]) \oplus h(\alpha_v[u]);\ \text{thus} \\ h(\alpha_{v_c}[i]) &\overset{(a)}{=} h(\alpha_v[o] \boxplus \alpha_v[u] + (1 - 2h(\alpha_{v_l}[i]))\alpha_v[e]) \\ &= h(\alpha_v[o] \boxplus \alpha_v[u] + (1 - 2(h(\alpha_v[e]) \oplus h(\alpha_v[o]) \oplus h(\alpha_v[u])))\alpha_v[e]) \\ &= h(\alpha_v[o] \boxplus \alpha_v[u]) = h(\alpha_v[o]) \oplus h(\alpha_v[u]) \\ h(\alpha_{v_r}[i]) &\overset{(a)}{=} h((1-2h(\alpha_{v_l}[i]))\alpha_v[o] + (1-2h(\alpha_{v_l}[i] \oplus \alpha_{v_c}[i]))\alpha_v[u]) \\ &= h((1 - 2(h(\alpha_v[e]) \oplus h(\alpha_v[o]) \oplus h(\alpha_v[u])))\alpha_v[o] \\ &\qquad + (1 - 2(h(\alpha_v[e]) \oplus h(\alpha_v[o]) \oplus h(\alpha_v[u]) \oplus h(\alpha_v[o]) \oplus h(\alpha_v[u])))\alpha_v[u]) \\ &= h(\alpha_v[e]) \oplus h(\alpha_v[u]) \end{aligned}$$



(a) $\mathcal{PC}(18, 9)$ with $\boldsymbol{G} = \boldsymbol{T_2} \otimes \boldsymbol{T_3} \otimes \boldsymbol{T_3}$



(b) $\mathcal{PC}(18, 11)$ with $\boldsymbol{G} = \boldsymbol{T_3} \otimes \boldsymbol{T_3} \otimes \boldsymbol{T_2}$

Figure 4.2: SC decoding trees (light grey) that are pruned to their FSSC counterparts (black). White and grey leaf nodes represent frozen and information bits, respectively. Blue nodes are REP3A, pink nodes are REP3B, and green nodes are SPC.

where (a) uses Eq. 4.5. Further,

$$\begin{aligned}
\beta_v[u] &= \beta_{v_l}[i] \oplus \beta_{v_c}[i] \oplus \beta_{v_r}[i] \\
&\overset{(b)}{=} h(\alpha_{v_l}[i]) \oplus h(\alpha_{v_c}[i]) \oplus h(\alpha_{v_r}[i]) \\
&= h(\alpha_v[e]) \oplus h(\alpha_v[o]) \oplus h(\alpha_v[u]) \oplus h(\alpha_v[o]) \oplus h(\alpha_v[u]) \oplus h(\alpha_v[e]) \oplus h(\alpha_v[u]) \\
&= h(\alpha_v[u]);\ \text{thus} \\
\beta_v[o] &= \beta_{v_l}[i] \oplus \beta_{v_r}[i] \overset{(b)}{=} h(\alpha_{v_l}[i]) \oplus h(\alpha_{v_r}[i]) \\
&= h(\alpha_v[o]) \\
\beta_v[e] &= \beta_{v_l}[i] \oplus \beta_{v_c}[i] \overset{(b)}{=} h(\alpha_{v_l}[i]) \oplus h(\alpha_{v_c}[i]) \\
&= h(\alpha_v[e])
\end{aligned}$$

where (b) also makes use of Eq. 4.5. Hence, the proof is completed and confirms that decoding of a R1 node can be accomplished using the same method for $\boldsymbol{T_3}$ as for $\boldsymbol{T_2}$. The original proof in [5] grants that this decoder holds for a R1 node of any depth.

Single-Parity Check Node

In MK polar codes with rate $R = \frac{N-1}{N}$, u_0 will always be frozen regardless of the order of kernels, and so MK polar codes also have an embedded SPC property. Observe that if $N = 3$, then $\boldsymbol{u} = (0, u_0, u_1)$ and $\boldsymbol{x} = (u_0, u_1, u_0 \oplus u_1)$. As such, ternary SPC nodes can be identified and decoded in exactly the same way as with Arıkan polar codes. The same is true for an SPC node of any depth with any combination of $\boldsymbol{T_2}$ and $\boldsymbol{T_3}$.

Repetition Node

The ternary kernel also mirrors repetition codes when rate $R = \frac{1}{N}$ since only u_{N-1} will contain information. However, the decoding procedures are more involved than with Arıkan polar codes. When considering only an Arıkan kernel, a REP node results in x where all indices are equal to u_{N-1}. Conversely, x does not repeat u_{N-1} in all indices in a MK REP node, although a pattern is still present. Specifically, if $N = 3$ and $u = (0, 0, a_0)$, then $x = (0, a_0, a_0)$. In this example, repetition decoding can still be carried out, supposing that the first index is excluded in the sum in Eq. 2.39. Generally, a REP node at stage S has a REP pattern P_v, which is determined for any combination of Arıkan or ternary kernels by performing a Kronecker product of repetition patterns $P_2 = (1, 1)$ or $P_3 = (0, 1, 1)$:

$$P_v = \bigotimes_i P_{k_i}, \quad \forall\, i \in [0, S). \tag{4.7}$$

Eq. 2.39 can then be modified to accommodate this addendum:

$$\beta_v^i = HD\left(\sum_j \alpha_v^j \cdot P_v^j\right) \quad \forall\, i, j \in v. \tag{4.8}$$

Table 4.1 outlines several examples of REP patterns with varying kernel orders. A new notation for repetition nodes may be appropriate for the purposes of decoder scheduling depending on the order of kernels. A REP node that is made up of only Arıkan kernels, previously the only type of REP node, can now be labeled as a *REP2* node. REP nodes comprised of only ternary kernels can be labeled as *REP3A* nodes. Mixed kernel repetition nodes can be labeled *REP3B* or *REP3C* depending on the order of $\boldsymbol{T_2}$ or $\boldsymbol{T_3}$.

N_v	k_v	P_v	Type
3	(3)	(0,1,1)	REP3A
6	(2,3)	(0,1,1,0,1,1)	REP3C
6	(3,2)	(0,0,1,1,1,1)	REP3B
8	(2,2,2)	(1,1,1,1,1,1,1,1)	REP2
9	(3,3)	(0,0,0,0,1,1,0,1,1)	REP3A
12	(2,2,3)	(0,1,1,0,1,1,0,1,1,0,1,1)	REP3C
12	(3,2,2)	(0,0,0,0,1,1,1,1,1,1,1,1)	REP3B
18	(2,3,3)	(0,0,0,0,1,1,0,1,1,0,0,0,0,1,1,0,1,1)	REP3C

Table 4.1: Examples of P_v patterns.

Although a MK polar code can be built using any arbitrary order of kernels, it is often the case that the non-Arıkan kernels are either the first or last constituents in the Kronecker expression used to compute $\boldsymbol{G}$. Further, Section 4.1 demonstrates that assuming the order of kernels without optimization presents comparable error correction results. Moreover, using LDPC WiMAX code lengths [8] as a guideline suggests that MK polar codes are able to achieve many desired code lengths with a limited number of ternary stages. Acknowledging this behaviour, it is unnecessary to implement generalized ternary repetition nodes as the majority of cases can be efficiently decoded under a few constraints. As such, the scheduling and implementation of MK REP nodes can be simplified by eliminating the computation of P_v. We propose to limit REP3A nodes to have a maximum size of 27 so that there are only 3 possible P_v, which can be simply stored instead of computed. Additionally, we limit REP3B and REP3C nodes to contain only a single ternary stage so that computation of β_v can be carried out efficiently:

$$\beta^i_{v_{\text{REP3B}}} = HD\left(\sum_j \alpha^j_v\right) \quad \forall\, i,j \in (\frac{N_v}{3}, N_v],$$
$$\beta^i_{v_{\text{REP3C}}} = HD\left(\sum_j \alpha^j_v\right) \quad \forall\, i,j \not\equiv \mod 3.$$

The summation in Eq. 2.39 is modified for REP3B nodes by skipping the first third of indices, while for REP3C node it is modified by skipping every third index. Of course, Eq. 2.40 still applies to ternary repetition nodes. Under the requirement of only a single ternary stage, these nodes do not need to be limited in size. We will utilize these simplifications in our numerical analysis.

N	R	# SC	# FSSC	# R0	# R1	# SPC	# REP2	# REP3	% Reduction
	0.25	158/189	37/27	7/2	1/0	4/4	0/4	1/0	76.6/85.7
96	0.5	158/189	43/45	8/5	1/5	6/3	0/3	0/0	72.8/76.2
	0.75	158/189	37/42	3/3	5/6	4/4	0/3	0/1	76.6/77.8
	0.25	654/849	101/118	15/11	4/6	16/13	0/11	4/2	84.5/86.1
432	0.5	654/849	110/136	14/9	4/7	21/19	0/15	7/0	83.2/83.4
	0.75	654/849	106/109	13/9	9/9	17/14	0/8	2/0	83.8/87.2
	0.25	1278/1533	196/186	34/17	5/8	24/19	0/19	3/0	84.6/87.9
768	0.5	1278/1533	223/222	31/15	9/14	31/24	0/22	4/0	82.5/85.5
	0.75	1278/1533	172/192	19/12	10/19	25/19	0/15	4/0	86.5/87.5
	0.25	3582/4602	409/453	62/31	8/16	71/54	0/52	5/0	88.6/90.1
2304	0.5	3582/4602	487/516	63/23	17/17	86/78	0/56	8/0	86.4/88.8
	0.75	3582/4602	395/441	45/24	27/39	60/50	0/36	9/0	88.9/90.4
	0.25	9840	1056	159	66	165	0	33	89.3
6561	0.5	9840	1186	142	111	176	0	46	87.9
	0.75	9840	1021	111	102	165	0	31	89.6

Table 4.2: Latency reduction of MK polar codes with ternary kernels as last/first Kronecker constituents.

4.2.2 Decoding Complexity Evaluation

This section outlines the effectiveness of the proposed MK-compatible FSSC decoder with numerical examples. With a sufficiently large processor, it can be assumed that each node in a decoding tree constitutes a single operation. As such, all measurements of decoding complexity refer to the number of decoding nodes. All polar codes analyzed were constructed using GA with a target $\frac{Eb}{N_0}$ of 3 dB. Fig. 4.3 compares the number of computations in FSSC decoders for various length-compatible polar codes over a range of codeword lengths and rates. *Punct QUP* [12] and *Short WL* [13] indicate puncturing and shortening patterns that allow for large numbers of frozen sets to be grouped together. Generally, puncturing and shortening methods have comparable decoding complexity to both kernel orderings of MK polar codes. However, MK codes with a high proportion of ternary stages, such as lengths $N = (216, 324, 648)$, have the fewest decoding operations overall when built with the *Last* kernel ordering. This is a result of the fast decoders for ternary nodes, which decode a larger number of bits simultaneously than would Arıkan nodes. This is depicted in Fig. 4.2a where a node in stage $s = 2$ decodes 9 bits at once, where as node in the same stage of Fig. 4.2b decodes only 6 bits at once. Therefore, it may be desirable to construct MK polar codes using the *Last* kernel ordering from a decoding complexity standpoint.

Regarding latency reduction of MK decoding, Table 4.2 outlines various comparisons between SC and FSSC along with the node makeup of FSSC decoding schedules. Just as

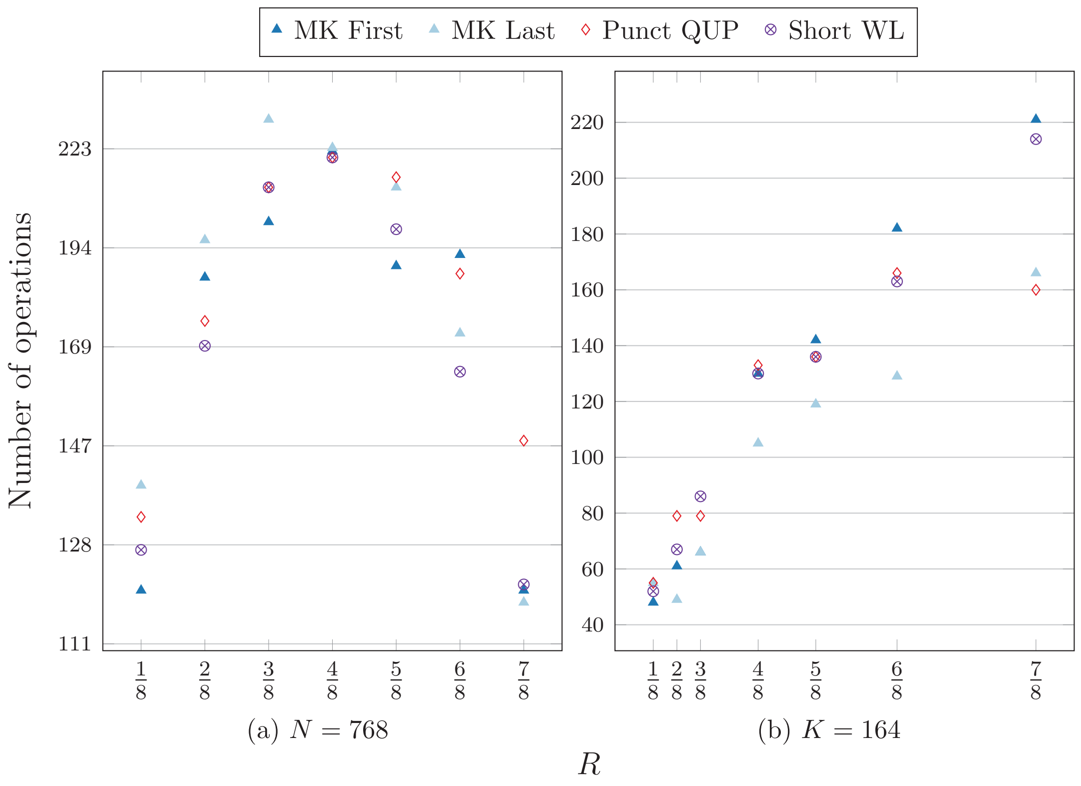


Figure 4.3: FSSC complexity for (a) $N = 768$ and (b) $K = 164$ sweeping rates $R \approx \{\frac{1}{8}, \dots, \frac{7}{8}\}$.

with Arıkan polar codes, there is a greater latency reduction for extreme rates. This gain is due to the higher proportion of R1 and SPC nodes for high rates and R0 and REP nodes for low rates. Observe that longer MK polar codes have greater latency reduction than short polar codes. It should also be noted that MK polar codes with a higher number of ternary stages have increased latency with the *First* kernel permutation compared with *Last*. This is a result of the fact that MK polar codes with a *Last* kernel order have all ternary stages at the bottom of the SC tree, where fast nodes are typically identified, indicating that the newly designed ternary fast nodes in this work are highly effective. Further, the number of ternary repetition nodes is proportionately low, so it is sufficient to consider only the most common cases for a simplified implementation rather than a complex generalized algorithm.

4.3 Punctured Multi-Kernel Polar Codes

This section will evaluate rate-matching techniques for MK polar codes. We will demonstrate that it is possible to puncture MK polar codes, and discuss possible use cases

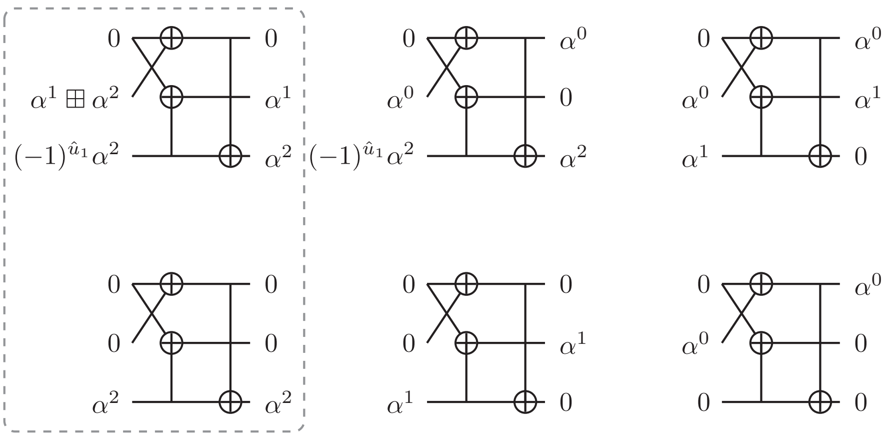


Figure 4.4: The *First* puncturing pattern is valid for $\boldsymbol{T_3}$.

complete with simulations. We will further explain that shortening MK codes is not straightforward and thus not viable.

To determine whether MK polar codes can be punctured, we must establish a viable puncturing pattern for the ternary kernel. Recall that any polar code index can be punctured so long as its leaf node LLR is 0 when the corresponding channel value is also 0. To test for this, we can examine all possible puncturing patterns on $\boldsymbol{T_3}$ to see which configurations adhere to this requirement. The test involves applying a 0 at a given index at the channel side of a $\boldsymbol{T_3}$ decoder and observing whether the corresponding leaf node also receives a soft 0. For the ternary kernel, there are 3 possible one-index and two-index permutations each. The case where all 3 indices are injected with 0 is not examined since there is no data to decode. The experiment is detailed in Fig. 4.4. Note that there are only two valid puncturing configurations, both of which follow the *First* puncturing pattern described in Section 3.1.1. Since it is known that this pattern is also valid for T_2, we can conclude that this pattern is valid for MK polar codes with any combination of both kernels. This will then be the considered puncturing pattern for MK polar codes.

Puncturing MK polar codes improves their flexibility to the point where any arbitrary length can be achieved. Additionally, MK polar codes offer a choice of block length for the mother code, since there exists an array of selections, unlike rate-matching of Arıkan polar codes. For example, if $E = 700$, the code design may utilize either of, but is not limited to, $N = 729$ or $N = 768$. The decision of the mother code may then prioritize either error correction performance or decoding complexity. In this case, a mother code block length of $N = 729$ requires only 4374 decoding operations under SC, while $N = 768$ requires 6912. However, a length of $N = 768$ is expected to have superior error correction performance since it was shown in Section 3.5 that error correction performance of MK

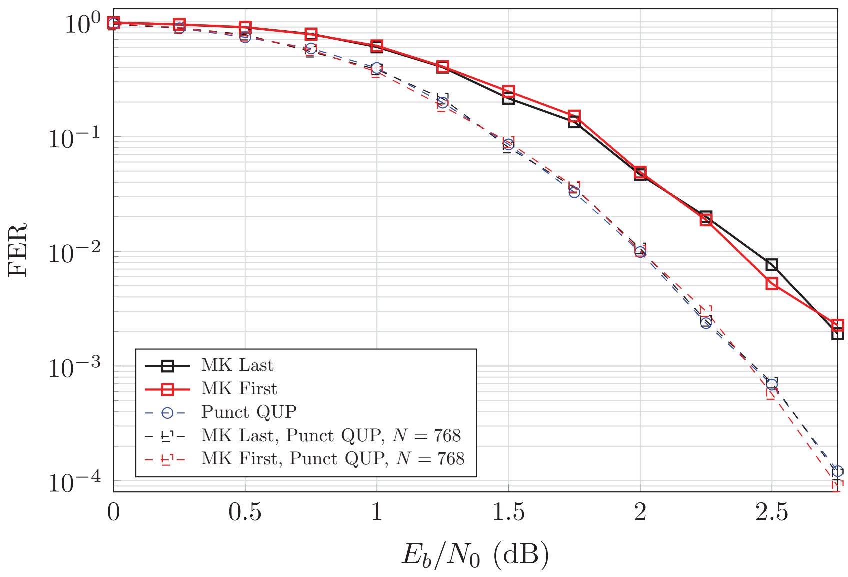


Figure 4.5: SCL performance on $\mathcal{PC}(729, 365)$ with $L = 8$, $C = 16$.

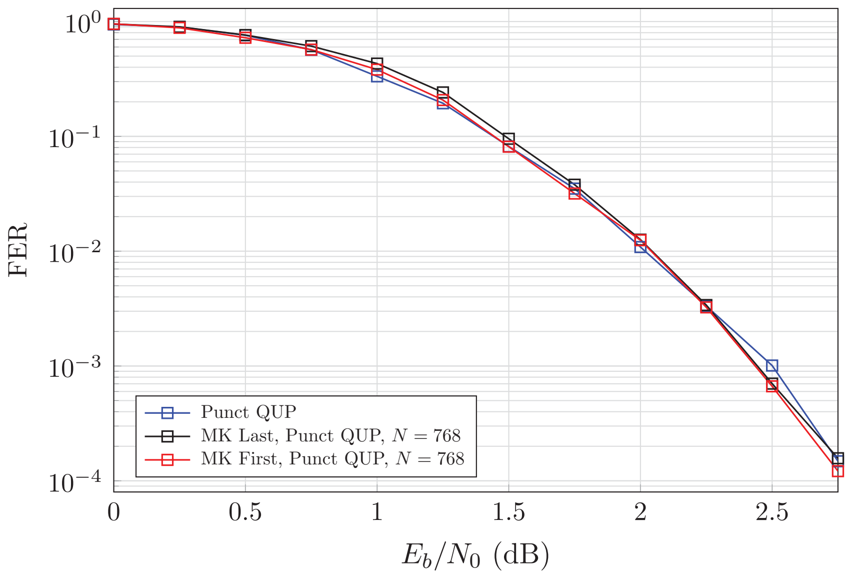


Figure 4.6: SCL performance on $\mathcal{PC}(700, 350)$ with $L = 8$, $C = 16$.

polar codes degrades with the increase of ϕ_{MK}. To weigh the new code design parameters, we will now consider use cases for punctured MK polar codes.

Fig. 4.5 outlines a scenario where E is a length that is attainable by a MK polar code, but the error correction performance can be improved upon by selecting a slightly larger mother code with fewer ternary stages. Observe that by setting $N = 768$ and $E = 729$, and puncturing 39 bits, an order of magnitude is gained in FER. As outlined in the previous paragraph, this improvement comes at the expense of increased decoding complexity.

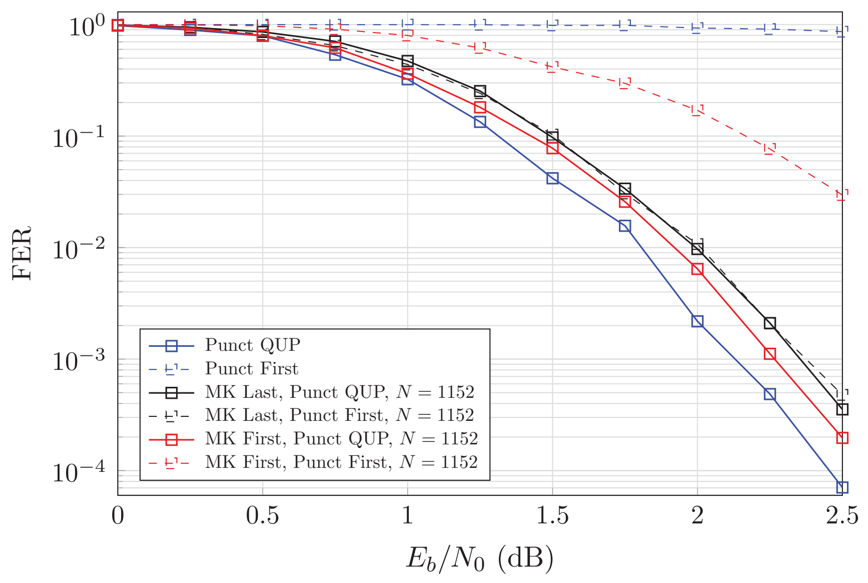


Figure 4.7: SCL performance on $\mathcal{PC}(1056, 528)$ with $L = 8$, $C = 16$.

Fig. 4.6 illustrates another scenario when E is not naturally attainable by a MK polar code, but ϕ_{RM} is lower than when considering puncturing an Arıkan polar code. This practice offers comparable error correction performance to a punctured Arıkan code, but the decoding complexity is significantly reduced. In this specific case, the Arıkan polar code requires that $N = 1024$ where SC requires 10240 decoding operations, while the MK polar code can select $N = 768$ where only 6912 decoding operations are needed. Fig. 4.7 details another example of this application to a greater extreme. If $E = 1056$, then a punctured Arıkan code uses $N = 2048$ and $\phi_{RM} = 0.48$, while a punctured MK code can use $N = 1152$ with $\phi_{RM} = 0.08$. Note the punctured MK codes afford a 53.9% SC decoding complexity reduction over the punctured Arıkan codes for an approximate 0.2 dB loss in FER for this configuration. Further, this simulation reveals that the low-complexity *First* puncturing scheme, which neglects accounting for punctured bits when computing $\mathcal{R}$, is highly unreliable.

On shortening of the ternary kernel

With a brief inspection, it becomes clear that $\boldsymbol{T_3}$ cannot be easily shortened. A valid shortening pattern must ensure that any index with an uncoded "0" results in a coded "0". This requirement is tested in Fig. 4.8 where it is made clear that this prerequisite is never satisfied. It can then be concluded that MK polar codes cannot be shortened in any practical manner.

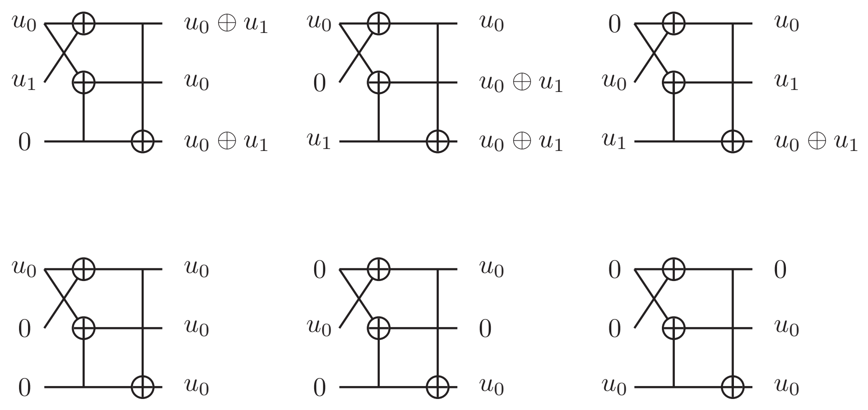


Figure 4.8: $\boldsymbol{T_3}$ does not support any valid shortening pattern.

4.4 Ternary Kernel Denies Dominant Contiguity

In [6], it was shown that systematic polar codes can be decoded with reduced latency when employing FSSC compared with non-systematic polar codes while maintaining the same FER. This improvement is a result of the fact that the source bits appear in the indices of $\mathcal{I}$ in the codeword, and so FSSC decoding can be completed by obtaining the partial sums at the top of the decoding tree rather than the bottom. Since FSSC often eliminates the majority of SC operations, there are fewer partial sum computations required to traverse to the top of the tree than the bottom. Thus, the geometry of the pruned decoding tree yields a reduced number of operations when employing systematic polar codes. As such, a simple systematic encoder can be desirable for further latency reduction.

Section 2.2.2 outlines a low complexity method for systematic encoding of Arıkan polar codes that is valid under the constraint of a dominant contiguous information set. Although not all polar codes with any arbitrary $\mathcal{I}$ can be systematically encoded this way, most useful $\mathcal{I}$ are dominant contiguous in practice.

We will now investigate the possibility of low complexity systematic encoding of MK polar codes. Recall that Eq. 2.17 is only valid when the information set is dominant contiguous. To test for dominant contiguity of $\mathcal{I}$, we must first establish any dominance relations of indices in the ternary kernel. For this test, the most appropriate dominance relation is the column dominance definition from Eq. 2.20, since binary dominance does not translate to the ternary kernel. The only dominance relation present in $\boldsymbol{T_3}$ among indices $\{0, 1, 2\}$ is $0 \triangleright 2$. Indeed, we can observe that systematic encoding is possible when $\mathcal{I} = \{0\}$:

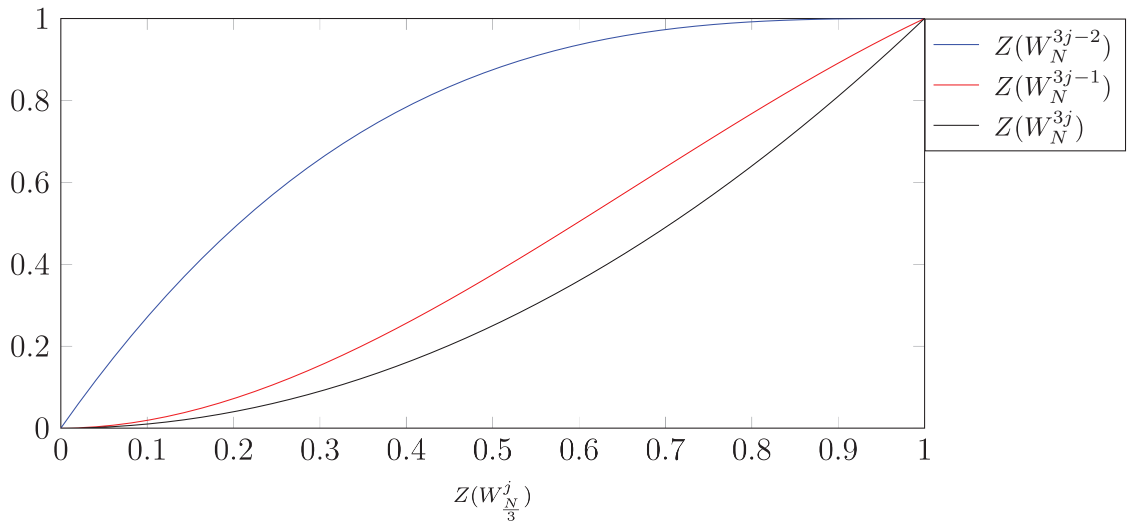


Figure 4.9: Bhattacharyya parameter evolution for a single ternary stage.

$$\boldsymbol{E} \cdot \boldsymbol{G} \cdot \boldsymbol{E^T} \cdot \boldsymbol{E} \cdot \boldsymbol{G}^{-1} \cdot \boldsymbol{E^T} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \end{bmatrix} = I. \quad (4.9)$$

As this example is a highly impractical, it does not make a strong case for the systematic encodability of $\boldsymbol{T_3}$. Further, we can demonstrate that this scenario is not likely to occur using Bhattacharyya parameters. Observe that index 0 will not be the first inserted in $\mathcal{I}$ since it is stochastically the least reliable in $\boldsymbol{T_3}$:

$$Z(W_N^{(3j-2)}) \geq Z(W_N^{(3j-1)}) \geq Z(W_N^{(3j)}). \quad (4.10)$$

This inequality is visualized in Fig. 4.9, where it is evident that index 0 is stochastically degraded while indices 1 and 2 are stochastically upgraded. This result indicates that the order in which indices of $\boldsymbol{T_3}$ are inserted into $\mathcal{I}$ is 2, followed by 1, followed by 0. Maintaining this order, $\mathcal{I}$ will never be dominant contiguous. We can further rule out the simplified systematic encoding of MK polar codes with a final counterexample. If we examine a plausible information set of $\mathcal{I} = \{1, 2\}$, we see that

$$\boldsymbol{E} \cdot \boldsymbol{G} \cdot \boldsymbol{E^T} \cdot \boldsymbol{E} \cdot \boldsymbol{G}^{-1} \cdot \boldsymbol{E^T} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\cdot \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \neq I, \quad (4.11)$$

and that the systematic encoding scheme is thus invalid. Thus MK polar codes do not exhibit the systematic property that Arıkan polar codes display. This finding suggests that the key to a valid systematic polar encoder is a generator matrix that is lower triangular and has a main diagonal of all "1"s. This requirement of the main diagonal reveals that each coded bit must be dependent at least on the uncoded bit of the same index.

Chapter 5

Asymmetric Polar Codes

This chapter proposes a new length-compatible polar coding scheme that is called *asymmetric polar codes* (APC). This new class of polar codes addresses key issues associated with the current state-of-the-art length-compatible polar codes discussed in Chapter 3.

Puncturing and shortening of Arıkan polar codes allow for any arbitrary block length to be achieved, though there are several issues with these methods. Because puncturing and shortening perform encoding and decoding on the mother code, their decoding complexity is not directly related to the size of the mother code, which renders them computationally inefficient. Also, provisions are sometimes made for the impact of puncturing and shortening on index reliability, which often necessitates a multiple-step code design. Even though low-complexity rate-matching code design has been successfully proposed [28], standardized polar coding in 5G indicates that the robustness of an industrial application evokes multi-step code design nonetheless. Multi-kernel polar codes certainly improve upon the native length flexibility of Arıkan polar codes, yet they introduce more complexity to an otherwise simple scheme. Moreover, it is worth noting that LDPC codes were favoured over polar codes for the data channel in 5G due to their length flexibility and decoding complexity that is directly dependent on their block length [32].

Asymmetric polar codes are a new coding scheme in which polar codes of unequal lengths are linked together using polarizing transformations. By linking together multiple polar codes in this way, any desired block length can be attained. The new method offers a straightforward approach to polar coding that is length-flexible and exhibits length-dependent decoding complexity. APCs are similar to Arıkan polar codes in the sense that they both have a recursive structure and contain smaller polar codes in their generator matrix. Many previously designed polar decoders and construction methods for Arıkan codes are fully compatible with APCs. Notably, APCs offer reduced time and space complexity with comparable error correction performance when measured against puncturing and shortening schemes.

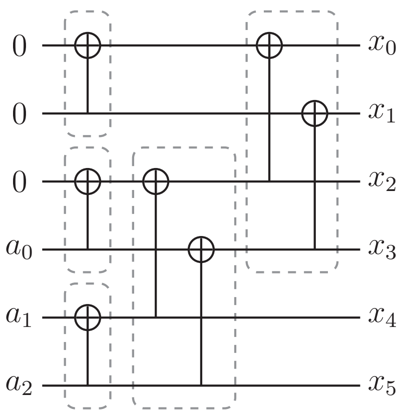
5.1 Generator Matrix Construction

An asymmetric polar code, denoted by $\mathcal{PC}(N, K)$, is a linear block code of length N and rate $R = \frac{K}{N}$. The codeword is obtained with $x = u\boldsymbol{G}$, as usual. An APC of length N is constructed from a minimum φ partial polar codes, which are determined by the decomposition of N into a sum of powers of 2, whose summands comprise the vector $\mathcal{A} = \{N_0, N_1, \ldots, N_{\varphi-1}\}$. The minimum required partial code lengths can be obtained using binary decomposition. For each 1 in bin(N), the corresponding binary index value is inserted into $\mathcal{A}$. For example, a code length of $N = 14$ is represented in binary as “1110”, and therefore $\mathcal{A} = \{8, 4, 2\}$ and $\varphi = 3$. $\mathcal{A}$ can be ordered so that APCs be constructed in either an *ascending* ($asc = 1$) or *descending* ($asc = 0$) permutation. The ascending order indicates that the size of the partial codes increases with bit indices, as in Fig. 5.1a. Alternatively, the descending order follows that the partial code sizes decrease with bit indices, which requires that $\mathcal{A}$ be reversed, as in Fig. 5.1b.

The linking process is executed recursively whereby each partial code matrix $\boldsymbol{G_{N_l}}$ is linked with the next partial code $\boldsymbol{G_{N_{l+1}}}$ according to the sequence $\mathcal{A}$. Generator matrix assembly for APCs requires $\varphi - 1$ iterations of this process. The polarizing stage used for each iteration l is the same XOR operation that is used for the last stage of an Arıkan code of length $2N_{l+1}$. This stage will include $J_l = \min(\sum_{k=0}^{l+1} N_k - N_{l+1}, N_{l+1})$ sum junctions, detailed in Fig. 5.3, which join indices j and $j + N_{l+1}$ for $j \in [0, J_l)$. The generator matrix is equal to the final iteration of the linking matrix, such that $\boldsymbol{G} = \boldsymbol{L_{\varphi-2}}$, where $\boldsymbol{L_l}$ is defined in eq. 5.1:

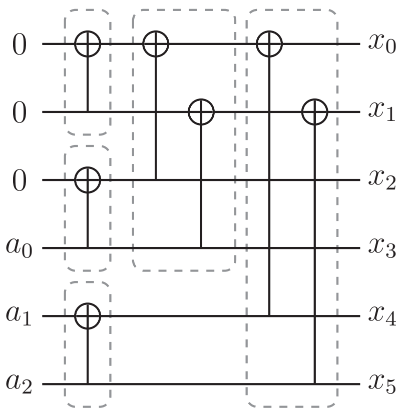
$$\boldsymbol{L_l} = \begin{cases} G_{N_l} & l = 0 \\ \left[\begin{array}{c|c} G_{N_l} & \mathbf{0} \\ \hline G_{N_l} \otimes \boldsymbol{\hat{1}^T_{\frac{\sum_{j<l} N_j}{N_l}}} & L_{l-1} \end{array}\right] & l > 0,\ asc = 1 \\ \left[\begin{array}{c|c} G_{N_l} & \mathbf{0} \\ \hline G_{N_l}[\sum_{j<l} N_j;] & L_{l-1} \end{array}\right] & l > 0,\ asc = 0, \end{cases} \tag{5.1}$$

$$G = \left[\begin{array}{cc|cccc} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ \hline 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right]$$



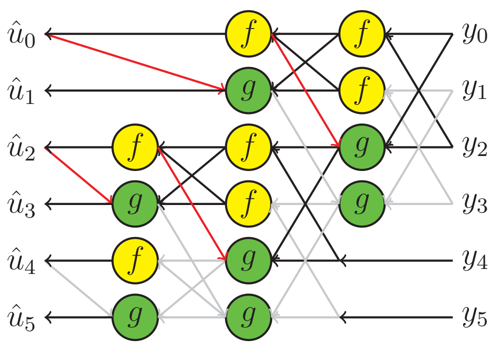
(a) *Ascending*, $\mathcal{A} = \{4, 2\}$, $\varphi = 2$

$$G = \left[\begin{array}{cccc|cc} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \end{array}\right]$$

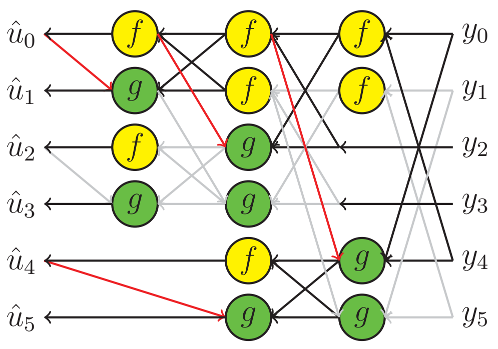


(b) *Descending*, $\mathcal{A} = \{2, 4\}$, $\varphi = 2$

Figure 5.1: APC encoders of length $N = 6$.



(a) *Ascending*, $\mathcal{A} = \{4, 2\}$, $\varphi = 2$



(b) *Descending*, $\mathcal{A} = \{2, 4\}$, $\varphi = 2$

Figure 5.2: SC decoding schedule for APC of length $N = 6$.

where $\hat{\mathbf{1}}_i$ is the all-one vector of length i, and $\boldsymbol{G}[i;]$ is generator matrix $\boldsymbol{G}$ with only its first i rows. It should be noted that $\mathcal{A}$ can contain a code length of 1, which equates to a single uncoded bit, *ie.* $\boldsymbol{G_1} = \begin{bmatrix}1\end{bmatrix}$. Observe that the linking matrix closely resembles the Arıkan kernel when $l > 0$. Further, this representation makes it possible to obtain Arıkan polar codes when $\mathcal{A}$ contains only two equal lengths. APCs can be constructed

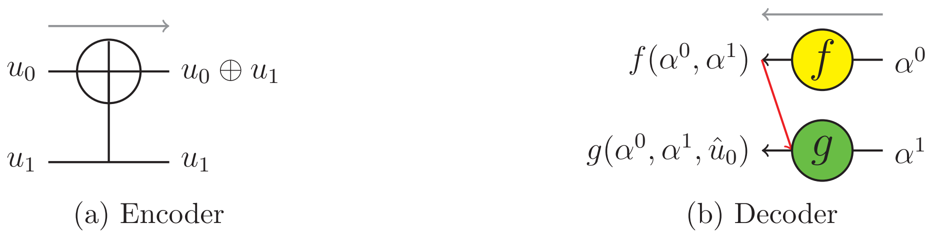


Figure 5.3: The polarizing sum junction.

using any $\mathcal{A}$, so long as its elements are powers of two. For example, an APC with $N = 6$ could be constructed with either of, but not limited to, $\mathcal{A} = \{2, 2, 2\}$ or $\mathcal{A} = \{4, 2\}$. Only the ascending and descending constructions using the minimum number of partial polar codes will be considered in the analysis.

5.1.1 Example: $N_A = 6$

Note in Fig. 5.1a the encoder of an ascending APC with $N = 6$ that has constituent polar codes of length 2 and 4. Combining these two partial codes is done with the additional polarizing stage. The additional stage is the same as the last stage of an Arıkan polar code of length twice that of the upper code in the factor graph. In particular, the upper code in this example has length 2 and thus requires the last stage of an Arıkan polar code of length 4. The SC decoding schedule must be modified to match the new factor graph, as depicted in Fig. 5.2a.

An APC generator matrix $\boldsymbol{G}$ must contain those of the constituent codes so as to be consistent with the original polar code definition. Thus, $\boldsymbol{G}$ is a block matrix comprised of the partial code matrices and the additional polarizing transforms discussed above. In this case, the final ascending generator matrix in Fig. 5.1a has block components $\boldsymbol{G_2}$ (red) and $\boldsymbol{G_4}$ (blue) that arranged according to the definition in eq. 5.1. A similar example for the corresponding descending APC is visualized in presented in Fig. 5.1b.

5.2 Asymmetric Information Set Design

Most code construction algorithms used to build reliability sets for Arıkan codes can be adapted to work for APCs with minor adjustments. GA serves as an effective method for designing reliability order. One must consider the asymmetry of the factor graphs of APCs in order to carry out the algorithm accurately. GA entails tracking the reliabilities

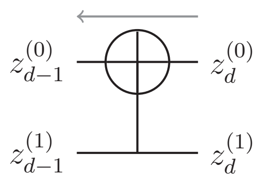


Figure 5.4: Gaussian approximation computation across an independent sum junction.

$$G = \left[\begin{array}{c|cc|cccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ \hline 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right]$$

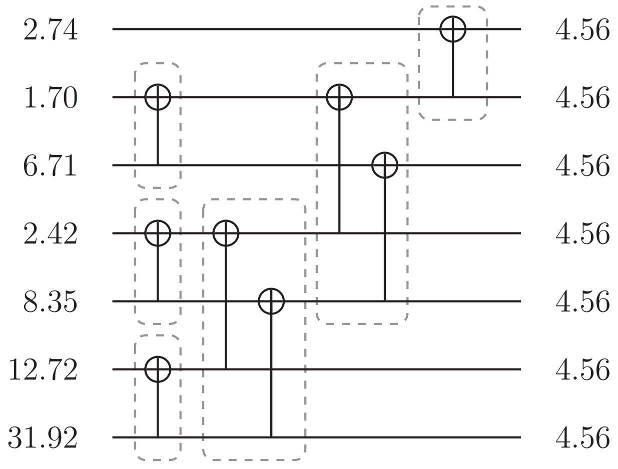


Figure 5.5: Gaussian approximation reliability ordering of an ascending APC with $N = 7$ and $\mathcal{A} = \{4, 2, 1\}$.

of each index at each stage of the factor graph, where the reliabilities are represented by the mean of the LLRs of the synthetic channels. To compute GA for APCs, begin by assigning each coded bit in the factor graph the LLR mean of the uncoded channel $z_1^{(0)}$, as described in Section 2.2.3.2. For Arıkan polar codes, it is sufficient to assume each input to a sum junction has been transformed an equal number of times and by equal amounts. APCs require special attention to the sum junctions, and so eq. 2.24 can be modified by applying the GA transformations independently at each sum junction. The reliabilities are transformed at each summation junction between stage d and $d-1$, as seen in Fig. 5.4, according to

$$\begin{aligned} z_{d-1}^{(0)} &= \phi^{-1}(1 - (1 - \phi(z_d^{(0)}))(1 - \phi(z_d^{(1)}))), \\ z_{d-1}^{(1)} &= z_d^{(0)} + z_d^{(1)}. \end{aligned} \tag{5.2}$$

Computing the entire graph results in vector z_N. As usual, this vector can be used to rank the indices in terms of reliability to form $\mathcal{R}$.

A construction example for $\mathcal{PC}(7, 4)$ with $\mathcal{A} = \{4, 2, 1\}$ is now presented for a target $\frac{E_b}{N_0} = 3\text{dB}$. The propagation results in $z_N^{(0)} = \{2.74, 1.70, 6.71, 2.42, 8.35, 12.72, 31.92\}$, as is detailed in Fig. 5.5. In this case, $\mathcal{R} = \{6, 5, 4, 2, 0, 3, 1\}$, $\mathcal{I} = \{6, 5, 4, 2\}$, and

$\mathcal{F} = \{0, 3, 1\}$. The Bhattacharyya parameter algorithm from Section 2.2.3.1 can be modified in the same way.

5.3 Asymmetric Polar Code Decoding

Just as when decoding Arıkan codes, the operations f and g in eq. 2.32 are used for decoding APCs since the same polarizing transform is used. As such, all standard polar decoders can be used with only a change in schedule according to the new factor graph. The SC decoding tree has φ partial code trees that are children of asymmetric nodes and are decoded in the same manner as described in Section 2.3.1. The $\varphi - 1$ asymmetric nodes are decoded with a schedule that corresponds to the additional polarizing stages. Specifically, each sum junction corresponds to an f and g function with equivalent indexing, as indicated by Fig. 5.3.

The $\varphi - 1$ asymmetric nodes each have α and β vectors of length $2J_l$ pertaining to the l^{th} iteration of the generator matrix linking process. The top asymmetric node accepts $2J_l$ channel values as its input. The input to intermediate asymmetric nodes is a combination of J_{l+1} LLRs from the previous asymmetric node and $N_l - J_{l+1}$ unused channel values when $asc = 1$, or simply J_{l+1} LLRs from the previous asymmetric node when $asc = 0$. Traversing the left and right branches of an asymmetric node calls for J_l f and g operations, respectively, replacing the $\frac{N_v}{2}$ offset in eq. 2.32 with N_{l+1}. In the descending case, the last $N_{l+1} - J_l$ values in α are unprocessed and directly stored in the next stage. Examples of APC decoding schedules corresponding to the encoding examples in Section 5.1.1 are depicted in Figs. 5.2a and 5.2b.

The number of computations required by an asymmetric SC decoder is described precisely as

$$\sum_{l=0}^{\varphi-1} N_l \log_2 N_l + \sum_{l=1}^{\varphi-1} 2N_l - (1 - asc) \sum_{l=0}^{\varphi-1} \big(2N_l - \sum_{k=0}^{l} N_k\big), \tag{5.3}$$

and at most as

$$\sum_{l=0}^{\varphi-1} N_l \log_2 N_l + \sum_{l=1}^{\varphi-1} 2N_l. \tag{5.4}$$

We will now prove that APCs always have fewer decoding operations than equivalent punctured or shortened Arıkan polar codes, and thus have lower decoding complexity overall. We begin by examining the worst case scenario: when employing the descending permutation and the block length is one less than a power of 2. This case yields the

Scheme	$N = 576$	$N = 768$	$N = 1536$	$N = 2304$	$N = 3072$
APC	5120	7168	15872	25088	34816
PS	10240	10240	22528	49152	49152
MK	4608	6912	15360	23040	33792

Table 5.1: The number of SC decoding operations for the tested block lengths.

highest possible φ. It is sufficient to demonstrate that if

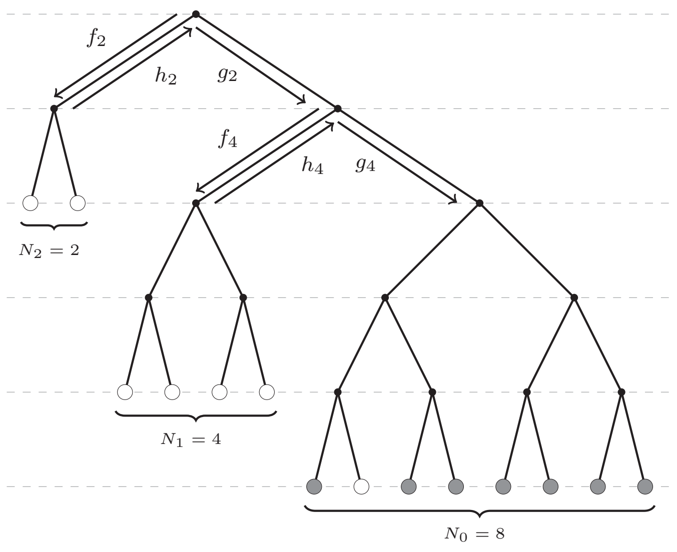
$$\sum_{l=0}^{\varphi-1} N_l \log_2 N_l + \sum_{l=1}^{\varphi-1} 2N_l < 2^{\lceil \log_2 N \rceil} \log_2 2^{\lceil \log_2 N \rceil}, \tag{5.5}$$

is true in the worst case where $N = 2^n - 1, n \in \mathbb{N}^+$, then APCs always have reduced decoding complexity compared with rate-matched Arıkan codes. As such, $\varphi = \lceil \log_2 N \rceil = \log_2 (N+1)$ and $\mathcal{A}$ contains all powers of two less than $N_M = 2^{\lceil \log_2 N \rceil}$:

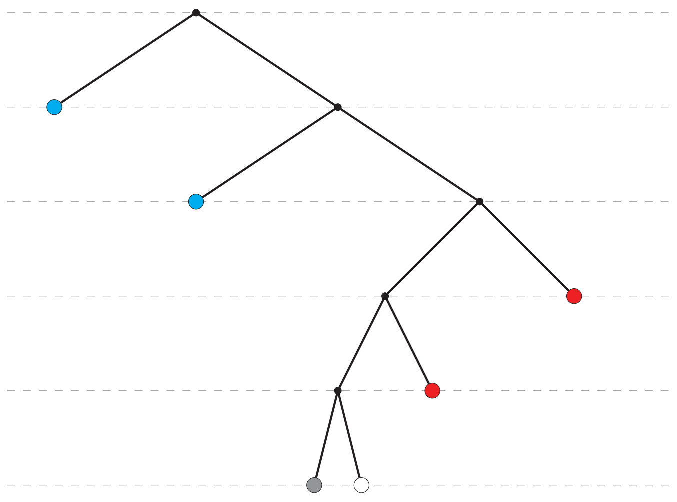
$$\begin{aligned} \sum_{l=0}^{\varphi-1} 2^l \log_2 2^l + \sum_{l=1}^{\varphi-1} 2 \cdot 2^l &< 2^{\varphi} \log_2 2^{\varphi}, \\ \sum_{l=0}^{\varphi-1} l \cdot 2^l + \sum_{l=1}^{\varphi-1} 2 \cdot 2^l &< \varphi \cdot 2^{\varphi}, \\ \varphi \cdot 2^{\varphi} - 2(2^{\varphi} - 1) + 2(2^{\varphi} - 2) &< \varphi \cdot 2^{\varphi}, \\ \varphi \cdot 2^{\varphi} - 2 &< \varphi \cdot 2^{\varphi}. \end{aligned} \tag{5.6}$$

Thus, APCs always have lower decoding complexity than PS codes. It can also be shown that APCs require fewer decoding operations when $asc = 1$ than when $asc = 0$ by replacing the second summation term with $\sum_{l=1}^{\varphi-1} 2 \cdot 2^{l-1}$, which results in a maximum number of decoding operations of $\varphi \cdot 2^{\varphi} - 2^{\varphi} < \varphi \cdot 2^{\varphi}$. In the case where $\varphi \leq 2$, the ascending and descending APCs have the same number of decoding operations.

FSSC decoding schedules can be obtained for APCs since all symmetric nodes remain powers of 2 and thus fast nodes can be decoded the same way as with Arıkan polar codes. The SC tree for an ascending APC where $N = 14$, along with its FSSC counterpart, can be seen in Fig. 5.6. Regarding error correction performance, the theoretical FER under SC decoding for APCs can be analytically computed using GA bit reliabilities, as was observed for Arıkan polar codes in Section 2.2.3.2.



(a) SC



(b) FSSC

Figure 5.6: SC tree for an ascending APC with $N = 14$ and $\mathcal{A} = \{8, 4, 2\}$. White, blue, and red leaves are frozen bits, R0 nodes, and R1 nodes, respectively. Subscripts indicate vector size.

5.4 Error Correction Performance

The error correction capabilities of APCs has been evaluated through a series of simulations using the AWGN channel and BPSK modulation. FER curves have been obtained for $N \in \{576, 768, 1536, 2304, 3072\}$ and $R \in \{\frac{1}{4}, \frac{1}{2}, \frac{3}{4}\}$. CA-SCL is the decoding algorithm used with list size $L = 8$ and CRC size $C = 16$ using polynomial $0x1021$. All frozen sets were reconstructed for each $\frac{E_b}{N_0}$ value in the plots using GA. We compared APCs with the two standard PS schemes as well as with MK polar codes. The kernel order of the MK codes was optimized for highest overall reliability by an exhaustive search, as proposed in [14].

Under CA-SCL decoding, asymmetric codes have comparable performance to leading length-compatible schemes when using a range of code rates, as depicted in Figs. 5.7 to 5.11. In all cases considered, APC performance is in the approximate range of $[-0.05\text{dB}, +0.05\text{dB}]$ compared to the best performing state-of-the-art codes for FER $= 10^{-4}$. Just as with Arıkan polar codes, APC error correction performance generally improves with length. It should be noted that APCs excel when they contain fewer and larger partial polar codes.

For all code lengths, APCs built with the ascending partial code permutation have superior error correction performance to APCs utilizing the descending permutation at low rates, while the opposite is true for high rates. This is due to the fact that the reliability of partial codes is related not only to their location in the factor graph, but also their size. Moreover, APCs have reduced performance when indices in $\mathcal{I}$ are found in smaller partial codes. The small partial codes in the descending permutation are where the few highly reliable indices are located at low rates, and so these bits exhibit reduced polarization effects due to the smaller partial code size. Conversely, the ascending permutation allows most information bits to be located in the largest partial codes at low rates, which present the highest reliability. It should also be noted that ascending and descending APCs have very similar performance to punctured and shortened polar codes, respectively. MK codes generally do not outperform PS or APC schemes in the tested scenarios. However, they are more likely to have worse performance when they have more than one ternary stage, as seen in Figs. 5.7ac to 5.11ac.

5.5 Complexity Evaluation

We will analyze the complexity of APCs by comparing them with PS and MK schemes in terms of SC and FSSC decoding complexity, space complexity, and code design. The SC time complexity was measured for all considered techniques by examining the number of LLR operations required for decoding a single codeword. The number of operations required for each scheme is given in Sections 2.3.1, 3.2.2, and 5.3. Table 5.1 outlines the number of decoding operations required for each tested scenario. Observe that both APC and MK codes have comparable decoding complexity that is directly related to their block length. Although MK codes can have reduced time complexity over equivalent punctured

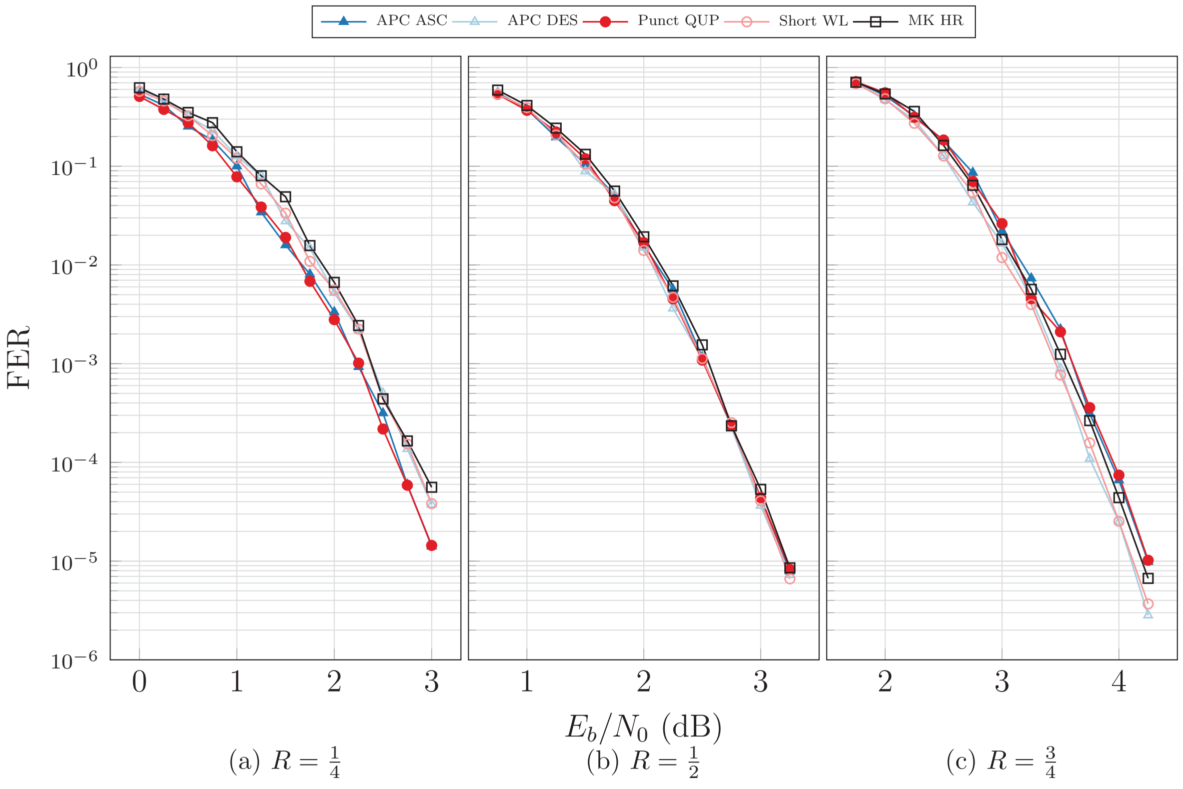


Figure 5.7: CA-SCL, L=8, C=16, $N = 576$, $R = \{\frac{1}{4}, \frac{1}{2}, \frac{3}{4}\}$.

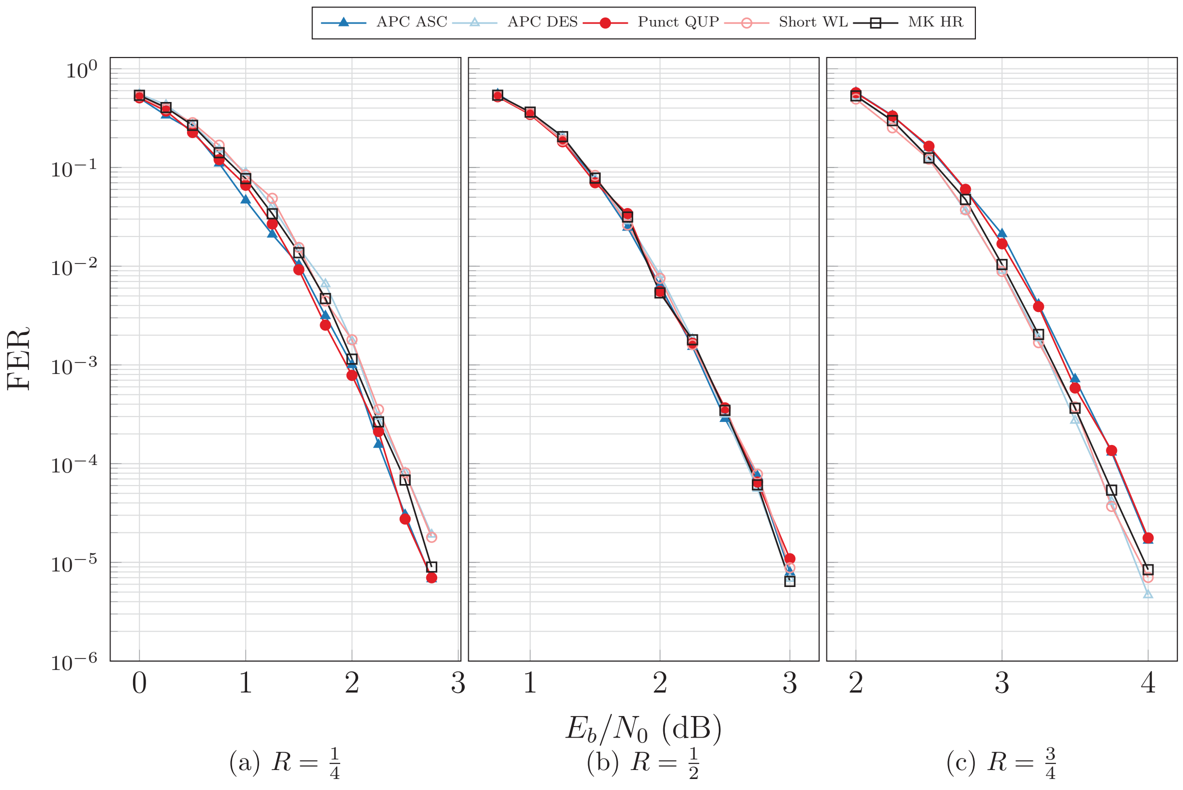
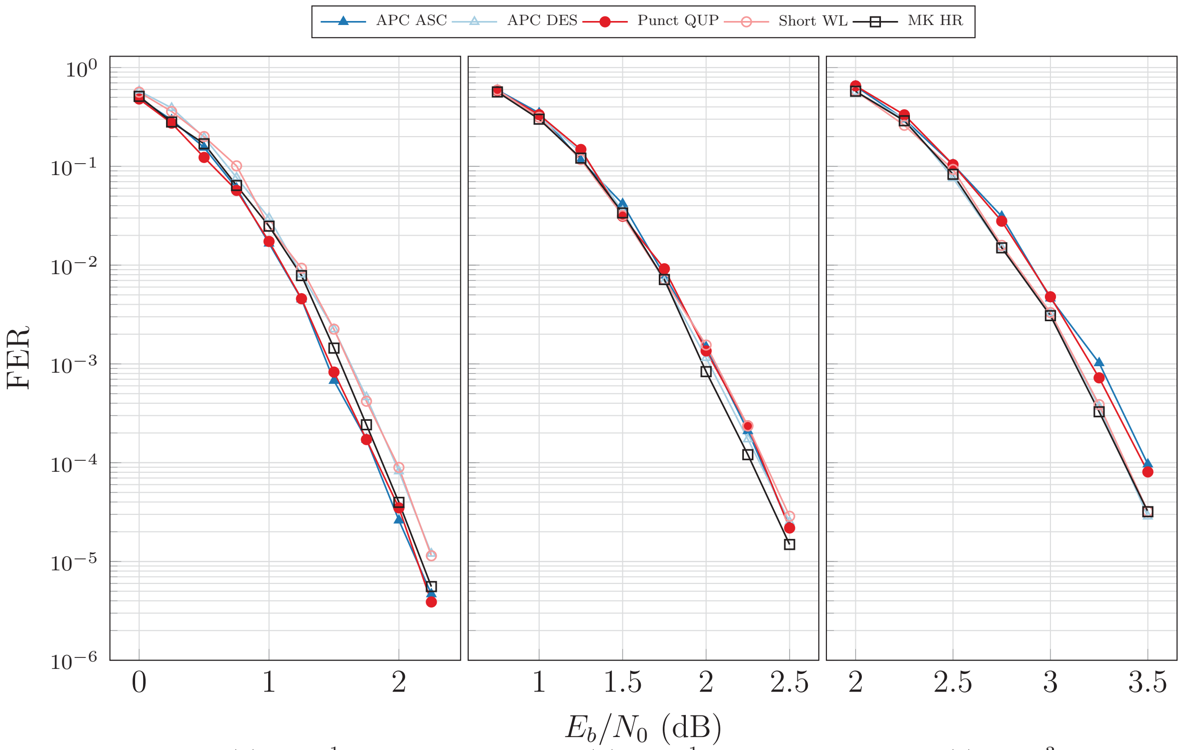
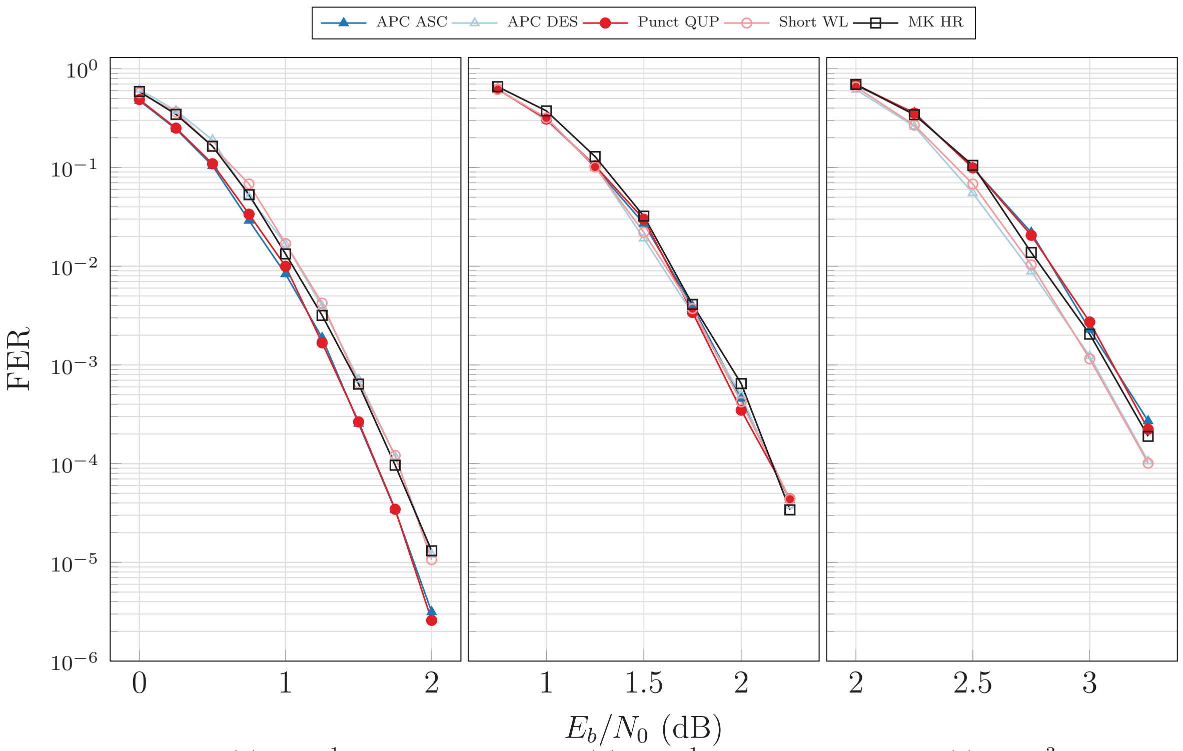


Figure 5.8: CA-SCL, L=8, C=16, $N = 768$, $R = \{\frac{1}{4}, \frac{1}{2}, \frac{3}{4}\}$.



(a) $R = \frac{1}{4}$ (b) $R = \frac{1}{2}$ (c) $R = \frac{3}{4}$

Figure 5.9: CA-SCL, L=8, C=16, $N = 1536$, $R = \{\frac{1}{4}, \frac{1}{2}, \frac{3}{4}\}$.



(a) $R = \frac{1}{4}$ (b) $R = \frac{1}{2}$ (c) $R = \frac{3}{4}$

Figure 5.10: CA-SCL, L=8, C=16, $N = 2304$, $R = \{\frac{1}{4}, \frac{1}{2}, \frac{3}{4}\}$.

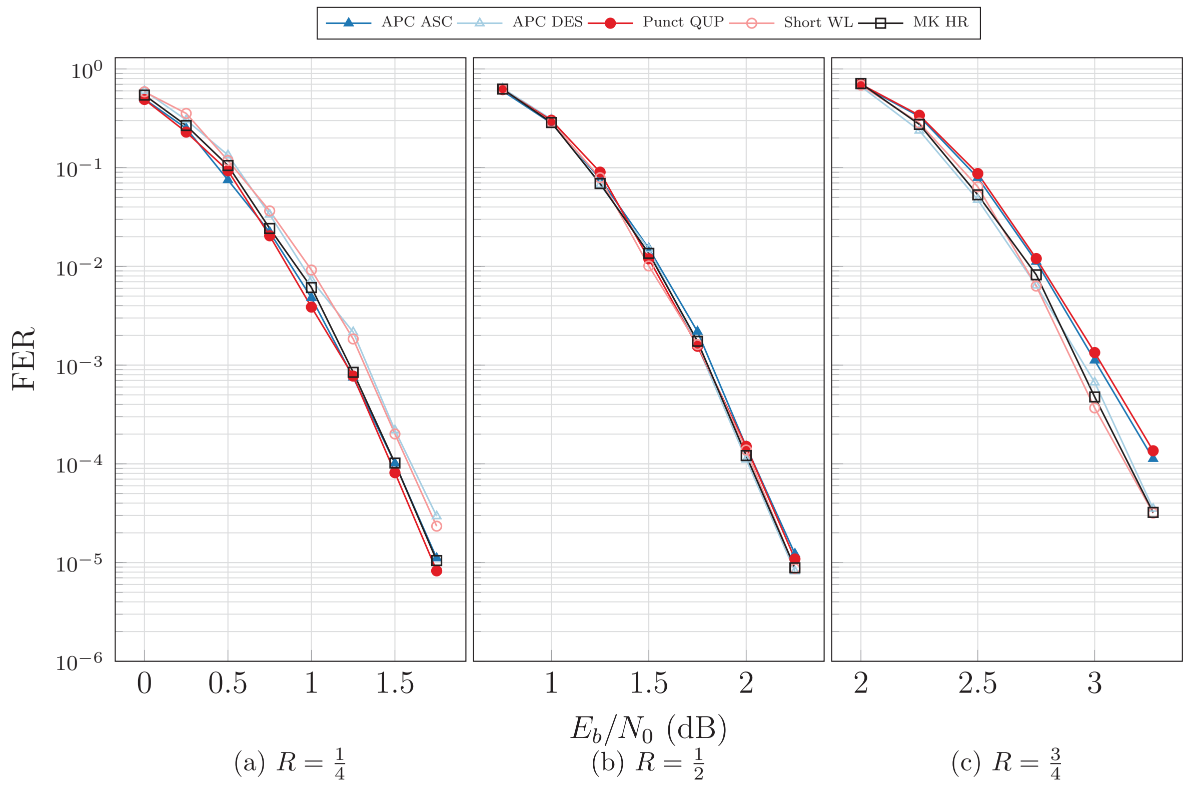


Figure 5.11: CA-SCL, L=8, C=16, $N = 3072$, $R = \{\frac{1}{4}, \frac{1}{2}, \frac{3}{4}\}$.

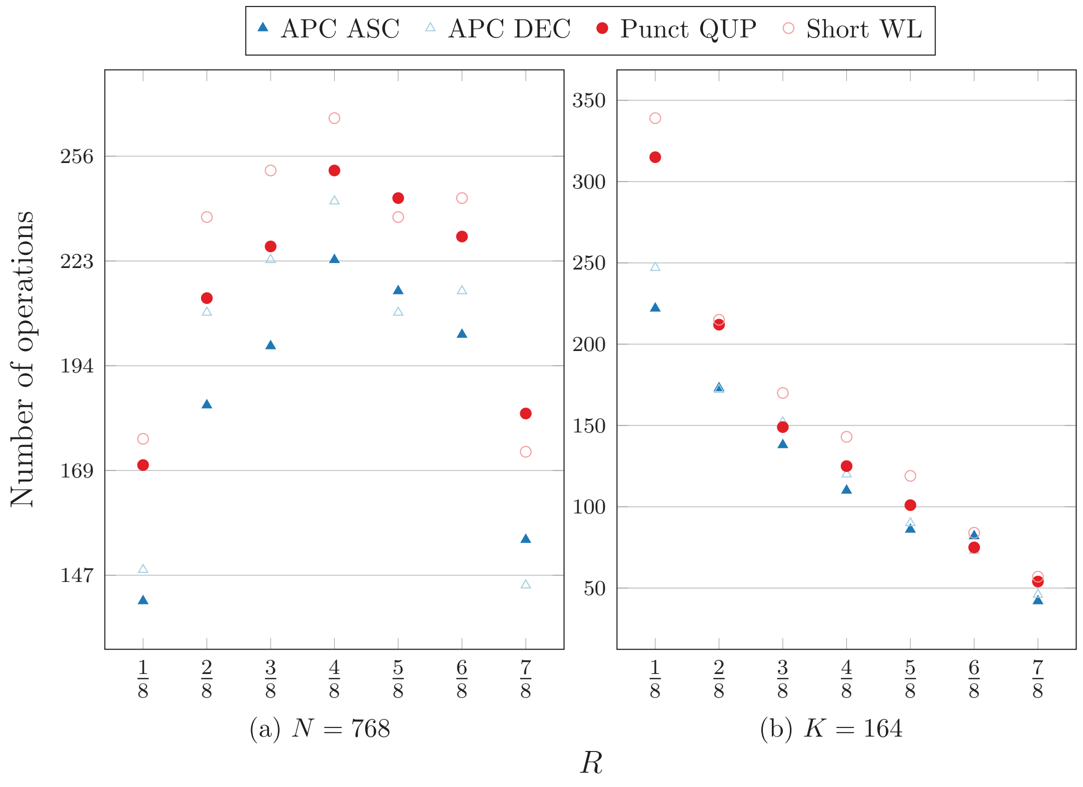


Figure 5.12: FSSC complexity for (a) $N = 768$ and (b) $K = 164$ sweeping rates $R = \{\frac{1}{8}, \ldots, \frac{7}{8}\}$.

or shortened codes and APCs, MK codes require concessions for ternary decoding using eq. 3.5, and kernel order when scheduling the decoder. The logic required for practical MK implementations is more complex and demands specialized hardware and increased memory over Arıkan codes [15]. Further, MK codes are limited in their length flexibility without considering the use of higher order kernels or rate-matching, which introduces further complication. As such, MK complexity is difficult to compare with that of APCs or PS codes.

Comparing FSSC decoding complexity requires a more refined metric. Since we are interested in a highly granular comparison and MK polar codes cannot natively achieve any block length, they will be excluded from this comparison. It was shown in [25] that a processing element size of 64 permits an acceptable balance between decoder throughput and hardware utilization for Arıkan polar codes with block length $N = 1024$. We will use this baseline for our comparisons, seeing that our experiments operate on a similar order of magnitude. Given that specialized nodes R1, R0, SPC, and REP are available with a maximum node size of 64, each specialized node is considered a single operation. Additionally, all computations of α^l and α^r of children nodes are counted as $\lceil \frac{N_v}{64} \rceil$ operation(s), where N_v is the child node size. It is worth noting that punctured or shortened decoders are likely to have a higher number of R0 and REP nodes due to their higher proportion of frozen bits. However, Fig. 5.12 demonstrates that APCs have between $[2.7\%, 27\%]$ time complexity reduction under Fast-SSC decoding when compared against PS schemes over a range of block lengths and rates. The degree to which APC complexity is reduced is dependent on ϕ_{RM}. When N is slightly larger than a power of 2, APCs exhibit significant complexity reduction.

Regarding space complexity, ascending APCs have decreased memory requirements than puncturing or shortening schemes. Using the efficient SC implementations proposed in [25], ascending APCs only require α and β memory for the largest partial code, which is at most half that of equivalent puncturing or shortening codes. For example, when $N = 2304$, PS decoders require the same α and β storage capacity as an Arıkan code of $N = 4096$, while APCs can be decoded with just the memory capacity of a $N = 2048$ Arıkan code and extended channel memory. Conclusively, PS approaches are not an efficient use of time and space resources since they must decode their mother code in order to receive an effectively smaller codeword. Further, MK codes necessitate ternary β memory, which results in increased storage requirements over purely Arıkan decoders [15].

Regarding information set construction, MK schemes must optimize the kernel order to maximize performance, and PS schemes must build a reliability set after the PS sets are determined, or vice versa. Thus, APCs demand only a single-step frozen set construction, while the competing schemes call for further considerations.

5.6 Extensions of Asymmetric Polar Codes

In this section, we will investigate a couple of useful extensions that will enhance the flexibility of asymmetric polar codes.

5.6.1 Punctured and Shortened Asymmetric Polar Codes

The proposition of a rate-matched asymmetric polar code seems somewhat paradoxical. If APCs are already length-flexible, what is the purpose of further altering the block length? Some APCs are constructed from several partial codes, and as such can suffer in error correction performance from the lack of symmetrically polarized subchannels. Overall, asymmetric polar codes are quite effective ECCs, but they may suffer from their design at extreme rates. In some circumstances, it may be beneficial to construct a slightly larger code that is less asymmetric and then puncture or shorten it. This premise may allow for an improvement in error correction performance while still maintaining reduced decoding complexity compared with a standard rate-matched Arıkan polar code.

We must first investigate whether it is possible for APCs to be punctured or shortened. Since APCs are comprised of Arıkan polar codes, we know that all partial codes can be punctured or shortened. Further, the sum junctions discussed in Section 5.1 should also be able to be punctured or shortened since they are the same units found inside Arıkan polar codes. Then, it should follow logically that some puncturing or shortening patterns from Section 3.1 should be compatible with APCs. Observe from Fig. 5.15 that APCs reliably honour the requirements of incapable and overcapable indices. This figure highlights the use of puncturing (red) and shortening (blue) of an APC with $E = 12$ and $N = 14$. Note that the *First* and *Last* patterns are employed. Note that the notion of bit-reversal does not apply to APCs, and so the BR puncturing and shortening patterns are not valid.

Fig. 5.13 depicts a simulation of a high rate polar code with a block length that is borrowed from the WiMAX standard for LDPC codes [33]. Because this is a high-rate

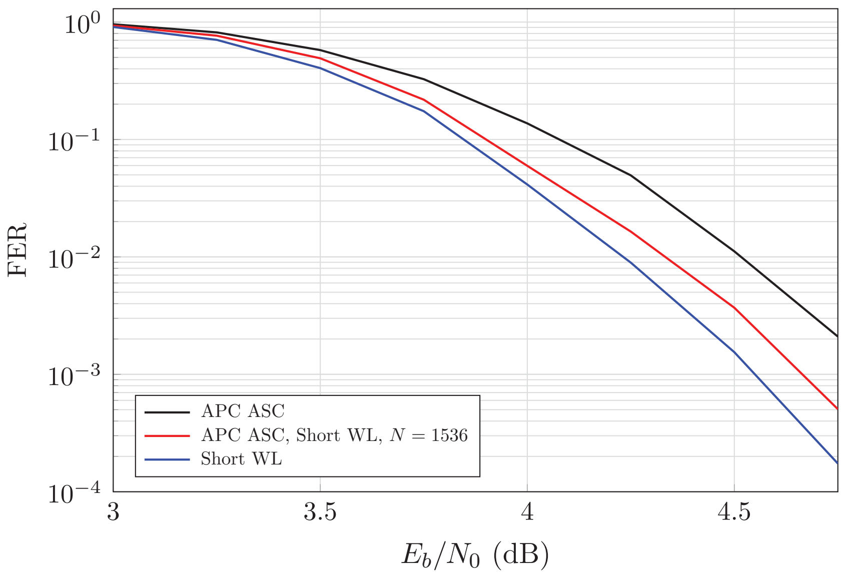


Figure 5.13: CA-SCL, $L = 8$, $C = 16$, $\mathcal{PC}(1440, 1296)$.

application, puncturing need not be considered. Three different codes were applied to construct $\mathcal{PC}(1440, 1296)$: A standard APC, a shortened APC with $N = 1536$, and a shortened Arıkan polar code. The results outline the continuum of performances that are possible with rate-matched asymmetric polar codes. This particular scenario induces a direct trade off between decoding complexity and error correction performance. On the one hand, decoding complexity can be minimized at the expense of one order of magnitude of FER by simply using a true APC. Alternatively, the shortened Arıkan code can be used to boost error correction performance with a sharp increase in decoding latency. Finally, this dichotomy can is mitigated when utilizing the shortened APC. As such, the size of the mother code, if any, can be selected to suit the needs of the application.

However, not all rate-matched APCs are useful. Through empirical evidence gathered during this research, many simulation cases resemble Fig. 5.14. Many instances of punctured or shortened APCs end up not improving error correction performance at all while also escalating decoding complexity. In these scenarios, the true APC is the most effective coding solution. This issue appears most prominent at medium block lengths and medium rates. This perhaps indicates that puncturing and shortening of APCs is only required for extreme rates.

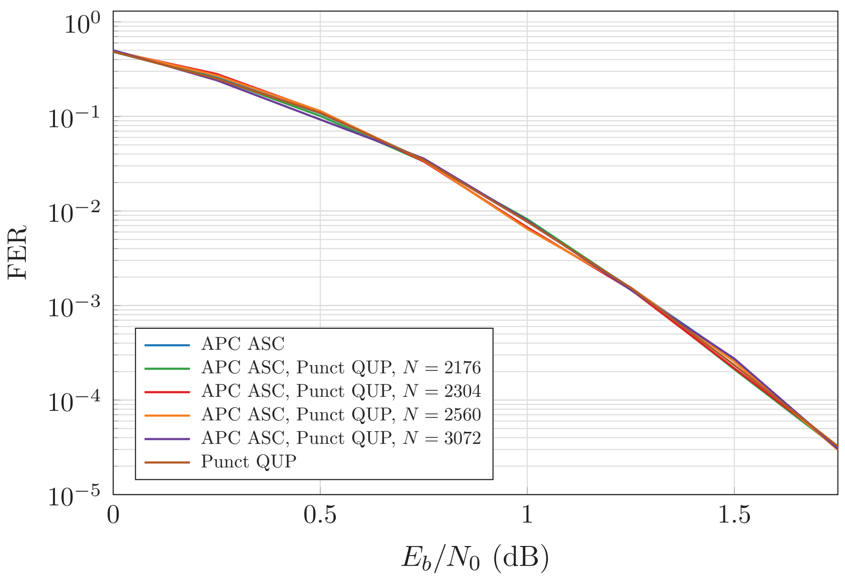


Figure 5.14: CA-SCL, $L = 8$, $C = 16$, $\mathcal{PC}(2112, 528)$.

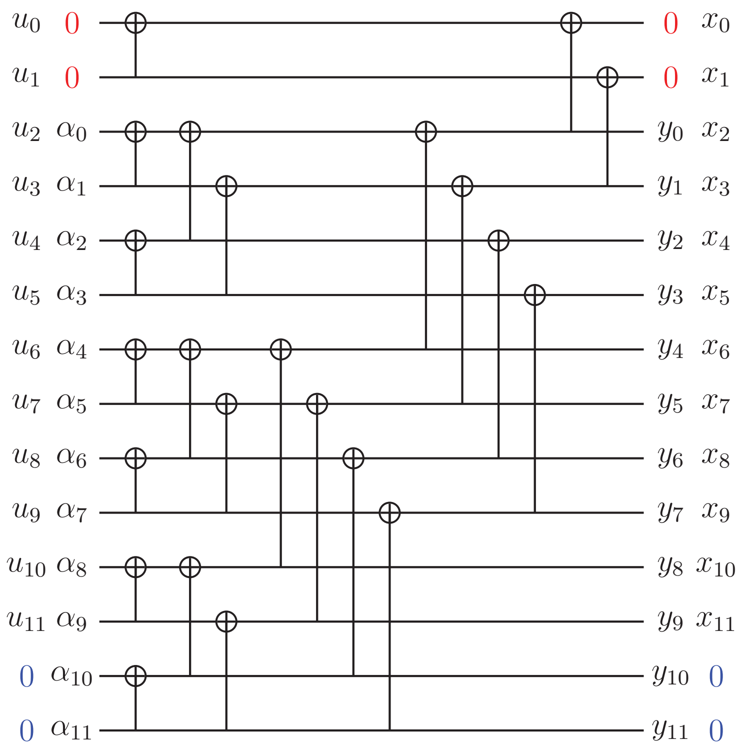


Figure 5.15: Factor graph for an ascending APC with $N = 14$ and $\mathcal{A} = \{8, 4, 2\}$. *First* puncturing pattern is highlighted in red, while *Last* shortening pattern is highlighted in blue.

5.6.2 Systematic Asymmetric Polar Codes

We will now examine whether APCs are able to be systematically encoded with the low-complexity method described in Section 2.2.2. Recall that the ability of a polar code to be systematic encoded is dependent on the dominant contiguity of information set $\mathcal{I}$, as implied by eq. 2.17. If this constraint is not met, then the re-freezing step of the systematic encoder risks altering codeword values.

Through the investigation conducted in Section 4.4 for testing $\boldsymbol{T_3}$ for dominant contiguous codes, it was demonstrated that failing to maintain a diagonal of all "1"s in the generator matrix proved detrimental to the systematic encoding of MK polar codes. Since APCs are assembled from Arıkan polar codes, their generator matrices, defined in eq. 5.1, always satisfy this property. Just as with MK polar codes, binary dominance is not an appropriate dominance relation for APCs. Column dominance, defined in eq. 2.20, will then serve as the dominance relation metric for APCs.

We can demonstrate that APCs can certainly achieve systematic encoding with an example. For an ascending APC with $N = 7$, we have the following column dominance relations:

$$\begin{aligned}
&6 \rhd 5, 4, 3, 2, 1, 0 \\
&5 \rhd 3, 1, 0 \\
&4 \rhd 3, 2, 1, 0 \\
&3 \rhd 1, 0 \\
&2 \rhd 1, 0 \\
&1 \rhd 0 \\
&0
\end{aligned} \tag{5.7}$$

Note that the highest order index is the most dominant, while the lowest order index is the least dominant, just as with Arıkan polar codes. As long as $\mathcal{I}$ honors this dominance order, this APC is able to be systematically encoded. For example, if we set $K = 4$ and use $\mathcal{R}$ from Fig. 5.5 to set $\mathcal{I} = \{2, 4, 5, 6\}$, which is dominant contiguous, we can confirm that this code can be systematically encoded:

$$\begin{aligned}
&\boldsymbol{E} \cdot \boldsymbol{G} \cdot \boldsymbol{E^T} \cdot \boldsymbol{E} \cdot \boldsymbol{G}^{-1} \cdot \boldsymbol{E^T} = \\
&\begin{bmatrix} 0&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0 \\ 0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&1 \end{bmatrix} \cdot \begin{bmatrix} 1&0&0&0&0&0&0 \\ 1&1&0&0&0&0&0 \\ 1&1&1&0&0&0&0 \\ 1&1&0&1&0&0&0 \\ 1&1&1&1&1&0&0 \\ 1&1&0&1&0&1&0 \\ 1&1&1&1&1&1&1 \end{bmatrix} \cdot \begin{bmatrix} 0&0&0&0 \\ 0&0&0&0 \\ 1&0&0&0 \\ 0&0&0&0 \\ 0&1&0&0 \\ 0&0&1&0 \\ 0&0&0&1 \end{bmatrix} \\
&\cdot \begin{bmatrix} 0&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0 \\ 0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&1 \end{bmatrix} \cdot \begin{bmatrix} 1&0&0&0&0&0&0 \\ 1&1&0&0&0&0&0 \\ 0&1&1&0&0&0&0 \\ 0&1&0&1&0&0&0 \\ 0&1&1&1&1&0&0 \\ 0&0&0&1&0&1&0 \\ 0&0&0&1&1&1&1 \end{bmatrix} \cdot \begin{bmatrix} 0&0&0&0 \\ 0&0&0&0 \\ 1&0&0&0 \\ 0&0&0&0 \\ 0&1&0&0 \\ 0&0&1&0 \\ 0&0&0&1 \end{bmatrix} \\
&= \begin{bmatrix} 1&0&0&0 \\ 0&1&0&0 \\ 0&0&1&0 \\ 0&0&0&1 \end{bmatrix} = I.
\end{aligned} \tag{5.8}$$

Alternatively, if we set $K = 5$ and use the same reliability vector, we obtain $\mathcal{I} = \{0, 2, 4, 5, 6\}$, which is not dominant contiguous. We can confirm that this violation results in an invalid systematic encoder:

$$\boldsymbol{E}\cdot\boldsymbol{G}\cdot\boldsymbol{E^T}\cdot\boldsymbol{E}\cdot\boldsymbol{G}^{-1}\cdot\boldsymbol{E^T} =$$

$$\begin{bmatrix}1&0&0&0&0&0&0\\0&0&1&0&0&0&0\\0&0&0&0&1&0&0\\0&0&0&0&0&1&0\\0&0&0&0&0&0&1\end{bmatrix}\cdot\begin{bmatrix}1&0&0&0&0&0&0\\1&1&0&0&0&0&0\\1&1&1&0&0&0&0\\1&1&0&1&0&0&0\\1&1&1&1&1&0&0\\1&1&0&1&0&1&0\\1&1&1&1&1&1&1\end{bmatrix}\cdot\begin{bmatrix}1&0&0&0&0\\0&0&0&0&0\\0&1&0&0&0\\0&0&0&0&0\\0&0&1&0&0\\0&0&0&1&0\\0&0&0&0&1\end{bmatrix}$$

$$\cdot\begin{bmatrix}1&0&0&0&0&0&0\\0&0&1&0&0&0&0\\0&0&0&0&1&0&0\\0&0&0&0&0&1&0\\0&0&0&0&0&0&1\end{bmatrix}\cdot\begin{bmatrix}1&0&0&0&0&0&0\\1&1&0&0&0&0&0\\0&1&1&0&0&0&0\\0&1&0&1&0&0&0\\0&1&1&1&1&0&0\\0&0&0&1&0&1&0\\0&0&0&1&1&1&1\end{bmatrix}\cdot\begin{bmatrix}1&0&0&0&0\\0&0&0&0&0\\0&1&0&0&0\\0&0&0&0&0\\0&0&1&0&0\\0&0&0&1&0\\0&0&0&0&1\end{bmatrix} \tag{5.9}$$

$$=\begin{bmatrix}1&0&0&0&0\\1&1&0&0&0\\1&0&1&0&0\\1&0&0&1&0\\1&0&0&0&1\end{bmatrix}\neq I.$$

The conclusion from this example is that APCs can indeed be systematically encoded, but that care must be taken when designing the information set. Since APCs have a tendency to exhibit unusual reliability orders, it is possible for an index to be dominated by another that is stochastically less reliable. Such is the case when $N = 7$: index 0 is dominated by all other indices, but it is more reliable than indices 1 and 3. Empirically, this eccentricity tends to happen in APCs with small block lengths and a large number of partial codes. In practical scenarios with long block lengths, APCs are largely dominant contiguous. In [18], it was suggested that if $\mathcal{I}$ is not dominant contiguous for Arıkan polar codes, that the error can simply be corrected by removing any violating indices and replacing them appropriately. Indeed, this fix can easily be applied to APCs as well.

Chapter 6

Conclusion

This thesis presented several analyses and improvements of existing length-compatible polar coding algorithms, as well as a novel polar coding scheme that was demonstrated to be an effective channel coding solution. This final chapter will summarize the presented research and propose avenues for future work.

Summary

After presenting the state-of-the-art for length-compatible polar codes in Sections 3.1 to 3.3, Section 3.4 presented an original simulation campaign that visualizes the error correction performance of all outlined coding techniques. Section 3.5 analyzed the simulation results to conclude that the 3GPP rate matching system presents a similar FER compared with the remaining length-compatible codes, but does so with a simplified code construction. This finding suggested that the 3GPP standard shows little indication of performance compromise. Further, multi-kernel polar codes were shown to exhibit a negative correlation between their proportion of ternary kernels and error correction performance.

Chapter 4 presented several optimizations for multi-kernel polar codes. Section 4.1 demonstrated that when constructing a multi-kernel polar code, the kernel order optimization step is largely unnecessary. Error correction performance was shown not to suffer when placing all ternary kernels at either the first or last positions of the Kronecker product, especially for long block lengths. This outcome signified that optimizing the kernel order for highest overall reliability is not essential. Section 4.2 presented a FSSC decoder that is compatible with MK polar codes using the ternary kernel. The decoder was shown to have a minimum complexity reduction of 72% over SC for all tested cases. This MK-compatible FSSC decoder was documented in a research paper titled “Fast Decoding of Multi-Kernel Polar codes”, which was accepted to IEEE Wireless Communications and Networking Conference (WCNC) 2019 in February 2019. Puncturing of MK polar codes was confirmed to be possible in Section 4.3, where it was also verified that MK shortening is not feasible. Finally, it was shown that low complexity systematic encoding of MK polar codes using $\boldsymbol{T_3}$ is not achievable.

Asymmetric polar codes are a novel length-flexible polar coding scheme that were introduced in Chapter 5. Two different generator matrix constructions, ascending and descending, were proposed, along with the corresponding information set design algorithms. SC and FSSC scheduling were demonstrated to be possible for asymmetric polar codes using the same decoding functions as Arıkan polar codes. Asymmetric polar codes were demonstrated to have excellent error correction performance; they exhibit virtually the same FER as the other considered polar coding schemes while maintaining a reduced decoding complexity compared with punctured and shortened codes. Moreover, asymmetric polar codes were proven to have a decoding complexity that is directly dependent on their block length, much like LDPC codes. The preceding research on asymmetric polar codes was presented in a research paper titled “Asymmetric Construction of Low-Latency and Length-Flexible Polar Codes”, which was accepted to IEEE International Conference on Communications (ICC) 2019 in February 2019. Furthermore, rate-matching techniques were revealed to be compatible with asymmetric polar codes due to their lower triangular generator matrix. For this same reason, asymmetric polar codes were illustrated to be able to be systematically encoded under the constraint of a dominant contiguous information set.

Future Work

Although the work featured in this thesis is sound, appropriate extensions exist for any piece of good research. This section outlines some possible continuations of the presented work.

Lower Triangular Ternary Kernel

Although most multi-kernel polar code literature deals with the ternary kernel $\boldsymbol{T_3}$ presented in Chapter 3, the research in this thesis has demonstrated that this kernel suffers from several shortcomings. Namely, this ternary kernel cannot be systematically encoded or shortened, which are both byproducts of the lack of a lower triangular generator matrix. To overcome this issue, one might consider constructing a multi-kernel polar code using an alternate ternary kernel with the same polarization exponent that is lower triangular; perhaps the example presented in [31] would be sufficient.

Nested Reliability Sequences for Asymmetric Polar Codes

The onset of nested sequences and universal partial order for Arıkan polar codes presented a revolution in fast and simple code construction. This feature is currently not available for asymmetric polar codes. As such, discovering a way to construct nested reliability sequences for asymmetric polar codes would surely cement their potential for practical application.

Asymmetric Polar Code Hardware Implementation

Asymmetric polar codes are an excellent candidate for a hardware implementation of an SC or SCL decoder since their latency reduction properties are most likely to be beneficial in a dedicated architecture. In fact, the SC schedules presented in this thesis for asymmetric polar codes suggest that only minor modifications might be needed on existing polar code hardware designs to generate a valid asymmetric polar code decoder.

Partially Asymmetric Polar Codes

For some desired code lengths, asymmetric codes have a large set of partial polar codes, which can affect error correction performance in extreme cases. If the code length is limited to even numbers, it could be possible to construct partially asymmetric polar codes where only a limited number of the stages in the factor graph are asymmetric. This adjustment then allows the polar code overall to remain symmetric, which may improve the error correction performance over asymmetric polar codes for lengths with extreme asymmetry. For example, if we consider $N = 10$, an asymmetric polar code would typically construct this code with $\mathcal{A} = \{8, 2\}$. Alternatively, we could first build an asymmetric code with $N = 5$ and $\mathcal{A} = \{4, 1\}$ and then compute the Kronecker product of the resulting generator matrix and the Arıkan kernel, resulting in a partially asymmetric polar code with $N = 10$.

The end of this chapter thus marks the end of this graduate thesis.

Bibliography

[1] C. E. Shannon, “A mathematical theory of communication,” *The Bell System Technical Journal*, vol. 27, pp. 379–423, July 1948.

[2] C. Berrou, A. Glavieux, and P. Thitimajshima, “Near Shannon limit error-correcting coding and decoding: Turbo-codes,” in *Proceedings of ICC ’93 - IEEE International Conference on Communications*, vol. 2, pp. 1064–1070 vol.2, May 1993.

[3] R. Gallager, “Low-density parity-check codes,” *IRE Transactions on Information Theory*, vol. 8, pp. 21–28, Jan. 1962.

[4] E. Arikan, “Channel Polarization: A Method for Constructing Capacity-Achieving Codes for Symmetric Binary-Input Memoryless Channels,” *IEEE Trans. on Inform. Theory*, vol. 55, pp. 3051–3073, Jul. 2009.

[5] A. Alamdar-Yazdi and F. R. Kschischang, “A Simplified Successive-Cancellation Decoder for Polar Codes,” *IEEE Communications Letters*, vol. 15, pp. 1378–1380, Dec. 2011.

[6] G. Sarkis, P. Giard, A. Vardy, C. Thibeault, and W. J. Gross, “Fast Polar Decoders: Algorithm and Implementation,” *IEEE J. on Select. Areas in Commun.*, vol. 32, pp. 946–957, May 2014.

[7] K. Chen, K. Niu, and J. R. Lin, “List successive cancellation decoding of polar codes,” *Electronics Letters*, vol. 48, pp. 500–501, Apr. 2012.

[8] I. Tal and A. Vardy, “List Decoding of Polar Codes,” *IEEE Trans. on Inform. Theory*, vol. 61, pp. 2213–2226, May 2015.

[9] A. Balatsoukas-Stimming, M. B. Parizi, and A. Burg, “LLR-Based Successive Cancellation List Decoding of Polar Codes,” *IEEE Transactions on Signal Processing*, vol. 63, pp. 5165–5179, Oct. 2015.

[10] 3GPP, "NR; Multiplexing and Channel Coding," Tech. Rep. TS 38.212, June 2018. Release 15.

[11] V. Bioglio, C. Condo, and I. Land, "Design of Polar Codes in 5G New Radio," *CoRR*, 2018.

[12] K. Niu, K. Chen, and J.-R. Lin, "Beyond Turbo Codes: Rate-Compatible Punctured Polar Codes," in *2013 IEEE Int. Conf. Commun. (ICC)*, 2013.

[13] R. Wang and R. Liu, "A Novel Puncturing Scheme for Polar Codes," *IEEE Commun. Lett.*, vol. 18, pp. 2081–2084, Dec. 2014.

[14] F. Gabry, V. Bioglio, I. Land, and J.-C. Belfiore, "Multi-Kernel Construction of Polar Codes," *IEEE Int. Conf. Commun. (ICC)*, 2017.

[15] G. Coppolino, C. Condo, G. Masera, and W. J. Gross, "A Multi-Kernel Multi-Code Polar Decoder Architecture," *IEEE Trans. on Circuits and Syst. I: Regular Papers*, pp. 1–10, 2018.

[16] G. D. Forney, "Concatenated codes.," *Cambridge, MA: MIT Press*, 1965.

[17] N. Stolte, *Rekursive Codes mit der Plotkin-Konstruktion und ihre Decodierung*. PhD thesis, Technische Universität Darmstadt, Jan. 2002.

[18] G. Sarkis, I. Tal, P. Giard, A. Vardy, C. Thibeault, and W. J. Gross, "Flexible and Low-Complexity Encoding and Decoding of Systematic Polar Codes," *IEEE Trans. on Commun.*, 2016.

[19] H. Vangala, E. Viterbo, and Y. Hong, "A Comparative Study of Polar Code Constructions for the AWGN Channel," *CoRR*, 2015.

[20] P. Trifonov, "Efficient Design and Decoding of Polar Codes," *IEEE Trans. on Commun.*, vol. 60, pp. 3221–3227, Nov. 2012.

[21] A. Cassagne, O. Hartmann, M. Leonardon, T. Tonnellier, G. Delbergue, C. Leroux, R. Tajan, B. L. Gal, C. Jego, O. Aumage, and D. Barthou†, "Fast Simulation and Prototyping with AFF3CT," in *Int. Workshop on Signal Proces. Syst.*, IEEE, Oct. 2017.

[22] D. Wu, Y. Li, and Y. Sun, "Construction and Block Error Rate Analysis of Polar Codes Over AWGN Channel Based on Gaussian Approximation," *IEEE Commun. Letters*, vol. 18, pp. 1099–1102, Jul. 2014.

[23] M. Mondelli, S. H. Hassani, and R. Urbanke, "Construction of Polar Codes with Sub-linear Complexity," *IEEE International Symposium on Information Theory (ISIT)*, pp. 1853–1857, June 2017.

[24] G. He, J.-C. Belfiore, X. Liu, Y. Ge, R. Zhang, I. Land, Y. Chen, R. Li, J. Wang, G. Yang, and W. Tong, "Beta-expansion: A Theoretical Framework for Fast and Recursive Construction of Polar Codes," *IEEE Global Communications Conference*, Dec. 2017.

[25] C. Leroux, A. J. Raymond, G. Sarkis, and W. J. Gross, "A Semi-Parallel Successive-Cancellation Decoder for Polar Codes," *IEEE Trans. on Signal Processing*, vol. 61, pp. 289–299, Jan. 2013.

[26] G. Sarkis, P. Giard, A. Vardy, C. Thibeault, and W. J. Gross, "Fast List Decoders for Polar Codes," *IEEE J. on Select. Areas in Commun.*, vol. 34, pp. 318–328, Feb. 2016.

[27] S. A. Hashemi, C. Condo, and W. J. Gross, "Fast Simplified Successive-Cancellation List Decoding of Polar Codes," in *2017 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pp. 1–6, March 2017.

[28] V. Bioglio, F. Gabry, and I. Land, "Low-Complexity Puncturing and Shortening of Polar Codes," in *2017 IEEE Wireless Commun. and Networking Conf. Workshops (WCNCW)*, IEEE, March 2017.

[29] M. Benammar, V. Bioglio, F. Gabry, and I. Land, "Multi-Kernel Polar Codes: Proof of Polarization and Error Exponents," *IEEE Info. Theory Workshop*, 2017.

[30] V. Bioglio, F. Gabry, I. Land, and J.-C. Belfiore, "Minimum-Distance Based Construction of Multi-Kernel Polar Codes," *CoRR*, 2017.

[31] H.-P. Lin, S. Lin, K. A. S, and Abdel-Ghaffar, "Linear and Nonlinear Binary Kernels of Polar Codes of Small Dimensions With Maximum Exponents," *IEEE Trans. on Inform. Theory*, vol. 61, pp. 5253–5270, Oct. 2015.

[32] T. Richardson and S. Kudekar, "Design of Low-Density Parity Check Codes for 5G New Radio," *IEEE Communications Magazine*, vol. 56, pp. 28–34, March 2018.

[33] K.-W. Shin and H.-J. Kim, "A Multi-mode LDPC Decoder for IEEE 802.16e Mobile WiMAX," *JSTS:J. of Semiconductor Technol. and Sci.*, vol. 12, pp. 24–33, March 2012.