

HIGH THROUGHPUT DECODING METHODS AND ARCHITECTURES FOR POLAR CODES WITH HIGH ENERGY-EFFICIENCY AND LOW LATENCY

A DISSERTATION SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

By
Onur Dizdar
November 2017

High Throughput Decoding Methods and Architectures for Polar
Codes with High Energy-Efficiency and Low Latency
By Onur Dizdar
November 2017

We certify that we have read this dissertation and that in our opinion it is fully
adequate, in scope and in quality, as a dissertation for the degree of Doctor of
Philosophy.

Erdal Arıkan(Advisor)

Orhan Arıkan

Ali Ziya Alkar

Tolga Mete Duman

Barış Bayram

Approved for the Graduate School of Engineering and Science:

Ezhan Karaşan
Director of the Graduate School

ABSTRACT

HIGH THROUGHPUT DECODING METHODS AND ARCHITECTURES FOR POLAR CODES WITH HIGH ENERGY-EFFICIENCY AND LOW LATENCY

Onur Dizdar

Ph.D. in Electrical and Electronics Engineering

Advisor: Erdal Arıkan

November 2017

Polar coding is a low-complexity channel coding method that can provably achieve Shannon's channel capacity for any binary-input discrete memoryless channels (B-DMC). Apart from the theoretical interest in the subject, polar codes have attracted attention for their potential applications.

We propose high throughput and energy-efficient decoders for polar codes using combinational logic targeting, but not limited to, next generation communication services such as optical communications, Massive Machine-Type Communications (mMTC) and Terahertz communications. First, we propose a fully combinational logic architecture for Successive-Cancellation (SC) decoding, which is the basic decoding method for polar codes. The advantages of this architecture are high throughput, high energy-efficiency and flexibility. The proposed combinational SC decoder operates at very low clock frequencies compared to synchronous (sequential logic) decoders, but takes advantage of the high degree of parallelism inherent in such architectures to provide a higher throughput and higher energy-efficiency compared to synchronous implementations. We provide ASIC and FPGA implementation results to present the characteristics of the proposed architecture and show that the decoder achieves approximately 2.5 Gb/s throughput with a power consumption of 190 mW with 90 nm 1.3 V technology and block length of 1024. We also provide analytical estimates for complexity and combinational delay of such decoders. We explain the use of pipelining with combinational decoders and introduce pipelined combinational SC decoders. At longer block lengths, we propose a hybrid-logic SC decoder that combines the advantageous aspects of the combinational and synchronous decoders.

In order to improve the throughput further, we use weighted majority-logic decoding for polar codes. Unlike SC decoding, majority-logic decoding fails to achieve channel capacity, but offers better throughput due its parallelizable schedule. We give a novel recursive description for weighted majority-logic decoding for

bit-reversed polar codes and use the proposed definition for implementations without determining the check-sums individually as done in conventional majority-logic decoding. We demonstrate by analytical estimates that the complexity and latency of the proposed architecture are $\mathcal{O}(N^{\log_2 3})$ and $\mathcal{O}(\log_2^2 N)$, respectively. Then, we validate the calculated estimates by a fully combinational logic implementation on ASIC. For a block length of 256, the implemented decoders achieve 17 Gb/s throughput with 90 nm 1.3 V technology. In order to compensate the error performance penalty of the majority-logic decoding, we propose novel hybrid decoders that combine SC and weighted majority-logic decoding algorithms. We demonstrate that very high latency gains can be obtained by such decoders with small error performance degradation with respect to SC decoding.

Keywords: High throughput, energy efficiency, error correcting codes, polar codes, successive cancellation decoder, majority logic decoder, VLSI.

ÖZET

KUTUPSAL KODLAR İÇİN YÜKSEK ENERJİ VERİMLİLİĞİNE VE DÜŞÜK GECİKMEYE SAHİP YÜKSEK VERİ HIZLI KOD ÇÖZME METOD VE MİMARİLERİ

Onur Dizdar

Elektrik ve Elektronik Mühendisliği, Doktora

Tez Danışmanı: Erdal Arıkan

Kasım 2017

Kutupsal kodlama, Shannon kanal kapasitesine ikili-girdi simetrik ayrık hafızasız kanallarda (B-DMC) ulaşabildiği analitik olarak kanıtlanmış düşük karmaşıklığa sahip bir kodlama metodudur. Konuya olan yoğun teorik ilginin yanı sıra, kutupsal kodlar olası uygulama alanları açısından da dikkat çekmiştir.

Tezde, bunlarla limitli olmamakla beraber, optik haberleşme, Masif Makina-Tipi Haberleşme (mMTC) ve Terahertz haberleşme gibi gelecek nesil haberleşme servisleri için birleşimsel mantık kullanılarak polar kodlar için yüksek hızlı ve enerji-verimli kod çözücüler önerilmektedir. İlk olarak, polar kodlar için temel kod çözme metodu olan Ardışık Giderme (SC) kod çözmesi için tamamen birleşimsel mantıktan oluşan bir mimari önerilmiştir. Bu mimarinin avantajları yüksek kod çözme hızı, enerji verimliliği ve esnekliktir. Önerilen birleşimsel kod çözücü, senkron (sıralı mantık) kod çözücülere göre daha düşük saat frekanslarında çalışmakta, fakat yüksek paralelliliği sayesinde yüksek kod çözme hızı ve enerji verimliliği sağlayabilmektedir. Önerilen mimarinin özelliklerini sunmak için ASIC ve FPGA gerçekleştirme sonuçları verilmiş ve kod çözücünün 90 nm 1.3 V teknoloji ve 1024 blok uzunluğu için 190 mW güç tüketimi ile yaklaşık 2.5 Gb/s kod çözme hızı sağladığı gösterilmiştir. Ayrıca bu kod çözücülerin karmaşıklık ve gecikme kestirimleri analitik olarak verilmiştir. Yüksek kod uzunlukları için, birleşimsel kod çözücünün avantajlı özelliklerini senkron kod çözücülerin düşük karmaşıklığa sahip yapısıyla birleştiren bir hibrit-mantıksal kod çözücü önerilmiştir. Bu kod çözücü tarafından elde edilen veri hızı kazancı analizi verilmiştir.

Kod çözme hızını daha fazla arttırmak için, kutupsal kodlar için ağırlıklandırılmış çoğunluk-mantığı kod çözmesine dayanan düşük gecikmeli bir kod çözücü mimarisi önerilmiştir. SC kod çözmenin aksine, çoğunluk-mantığı kanal kapasitesine erişemez, fakat paralelleştirmeye uygunluğu sayesinde daha

iyi veri hızı sağlar. Kutupsal kodlara yönelik ağırlıklandırılmış çoğunluk-mantığı kod çözmesi için yenilikçi bir özyinelemeli tanımlama verilmiş ve bu tanımlama, geleneksel çoğunluk-mantığı kod çözümünde olduğu gibi, kontrol-toplamları ayrı ayrı belirlenmeden gerçekleştirme yapmak için kullanılmıştır. Analitik kestirimler ile önerilen mimarinin karmaşıklık ve gecikmelerinin sırasıyla $\mathcal{O}(N^{\log_2 3})$ ve $\mathcal{O}(\log_2^2 N)$ olduğu gösterilmiştir. Daha sonra, bu hesaplanan kestirimler ASIC üzerinde tamamen birleşimsel mantık gerçekleştirme ile teyit edilmiştir. Gerçeklenen kod çözücüler 90 nm 1.3 V teknoloji ve 256 blok uzunluğu ile 17 Gb/s veri hızı sağlamıştır.

Çoğunluk-mantığı kod çözümünün hata performansı kaybını gidermek için, SC ve ağırlıklandırılmış çoğunluk-mantığı algoritmalarını birleştiren yenilikçi bir hibrid kod çözücü önerilmiştir. Bu kod çözücülerin, SC kod çözücüye göre az hata performansı kaybı ile oldukça yüksek gecikme kazançları sağladığı gösterilmiştir.

Anahtar sözcükler: Yüksek veri hızı, enerji verimliliği, hata düzelten kodlar, kutupsal kodlar, Ardışık Giderme kod çözücü, çoğunluk-mantığı kod çözücü, VLSI.

Acknowledgement

First and foremost, I would like to thank my supervisor Prof. Erdal Arıkan. His dedication, patience and support motivated me towards my PhD degree. His knowledge provided an invaluable guidance throughout my studies. I am truly grateful and honored to have had the chance of working with him.

I would like to express my sincere gratitude to my thesis monitoring committee members Prof. Orhan Arıkan and Prof. Ali Ziya Alkar for their valuable and constructive suggestions during the course of this work. I would also like to extend my thanks to Prof. Tolga Mete Duman and Assoc. Prof. Barış Bayram for their willingness to serve as examiners for my thesis defense. I wish to acknowledge the help provided by Prof. Abdullah Atalar and Prof. Sinan Gezici in a number of ways.

I would like to thank my wonderful wife Seçil for her patience, support and encouragement. She always believed in me and was always there for me in my times of need. Her support made it possible for me to complete this thesis.

This thesis would not have been possible without my family. I owe my deepest gratitude to them for all the patience, love and support during my studies. It is my privilege to have them in my life.

I am indebted to many of my colleagues in ASELSAN. I would like to thank Ertuğrul Kolagasioğlu for his support, attitude and teachings. Special thanks to Özlem Özbay, Dr. Defne Küçükyavuz and Dr. Füzüzan Atay Onat for their encouragement to begin my PhD studies. I thank deeply my colleague Güven Yenihayat, whom I started my career with and shared much throughout the journey. I am particularly grateful to Çağrı Göken, Dr. Oğuzhan Atak, Soner Yeşil and Mustafa Kesal for the invaluable technical discussions. I offer my gratitude to Dr. Mehmet Önder, Dr. Tolga Numanoğlu, Barış Karadeniz, Alptekin Yılmaz and Oğuz Özün for the encouragement to pursue my studies. My special thanks are extended to the administration of ASELSAN for the support on my PhD studies.

Particular thanks go to my labmates Bilkent University. I would like to thank Dr. Sinan Kahraman, Altuğ Süral and Tufail Ahmad for their help during the

course of my thesis. I am also thankful to administrative assistant of my department, Mürüvet Parlakay, for taking care of all administrative issues. I would also like to extend my thanks to Bilkent University for giving me the opportunity study here.



Contents

1	Introduction	1
1.1	ECC and Decoder Performances	3
1.2	Background and Motivation for the Thesis	7
1.2.1	State-of-the-Art in ECC and Motivation	9
1.3	Contributions of the Thesis	12
1.3.1	Combinational SC Decoder	13
1.3.2	Weighted Majority-Logic Decoding of Polar Codes	14
1.4	Outline of the Thesis	16
2	Background on Polar Coding	18
2.1	Notations and Preliminaries	18
2.2	Polar Codes	20
2.2.1	Code Construction	23
2.2.2	Encoding	24
2.2.3	Successive-Cancellation Decoding	26
2.3	Summary of the Chapter	32
3	Decoding Algorithms and Decoder Implementations for Polar Codes	34
3.1	Decoding Algorithms for Polar Codes	34
3.1.1	Successive-Cancellation List Decoding	35
3.1.2	Belief Propagation Decoding	38
3.1.3	Majority-Logic Decoding	39
3.2	State-of-the-Art Polar Decoders	45
3.3	Summary of the Chapter	51

4	Combinational SC Decoder	53
4.1	Architecture Description	54
4.1.1	Base Decoder for $N = 4$	54
4.1.2	Combinational SC Decoder	55
4.1.3	Pipelined Combinational SC Decoder	59
4.1.4	Hybrid-Logic SC Decoder	61
4.2	Complexity and Delay Analyses	64
4.2.1	Complexity	64
4.2.2	Combinational Delay	65
4.3	Implementation Results	68
4.3.1	ASIC Synthesis Results	69
4.3.2	FPGA Implementation Results	75
4.4	Throughput Analysis for Hybrid-Logic Decoders	78
4.5	Summary of the Chapter	79
5	Weighted Majority-Logic Decoding of Polar Codes	82
5.1	Architecture Description	83
5.1.1	Recursive Definition for Weighted Majority-Logic Decoder	83
5.1.2	Hybrid Decoder	92
5.2	Complexity and Latency Analyses	93
5.2.1	Weighted Majority-Logic Decoder	93
5.2.2	Hybrid Decoder	97
5.3	Implementation Results	98
5.4	Error Performance	102
5.4.1	Weighted Majority-Logic Decoder	102
5.4.2	Hybrid Decoder	110
5.5	Summary of the Chapter	114
6	Conclusions and Future Works	115
6.1	Conclusions	115
6.2	Suggestions for Future Work	119
6.2.1	Combinational SC Decoder	119
6.2.2	Weighted Majority-Logic Decoding for Polar Codes	120

List of Figures

1.1	Communication scheme with ECC	1
1.2	Net coding gain obtained by (1024, 512) polar code with SC decoding	4
1.3	Latency, pipelining and throughput	6
2.1	Communication scheme with polar codes	18
2.2	Channel combining process ($N = 2$)	21
2.3	Polar encoding graph for $N = 8$	27
2.4	Encoding circuit of $\mathcal{C}$ with component codes $\mathcal{C}_1$ and $\mathcal{C}_2$ ($N = 8$ and $N' = 4$)	29
2.5	SC algorithm decoding steps for $\hat{u}_0, \hat{u}_1, \hat{u}_2$ and $\hat{u}_3$. The red nodes and LLRs carried on the red lines are used for decoding the specified bit.	30
3.1	SCL performance	37
3.2	Processing element for BP decoding	38
3.3	Factor graph for BP decoding of polar codes	39
4.1	SC decoding trellis for $N = 4$	56
4.2	Combinational decoder for $N = 4$	56
4.3	Recursive architecture of polar decoders for block length N	57
4.4	RTL schematic for combinational decoder ($N = 8$)	58
4.5	Recursive architecture of pipelined polar decoders for block length N	60
4.6	Decoding trellis for hybrid-logic decoder ($N = 8$ and $N' = 4$)	66
4.7	FER performance with different numbers of quantization bits ($N = 1024, R = 1/2$)	77

4.8	FER performance of combinational decoders for different block lengths and rates	80
5.1	Circuit diagram for weighted majority-logic decoder for $N = 8$ using decoders for $N = 4$	85
5.2	Visualizations of $f_4^1(\ell)$, $f_4^2(\ell)$ and $f_4^4(\ell)$. The connected ℓ_i are input to the f function together.	88
5.3	Weighted majority-logic decoder for $N = 8$ using decoders for $N = 4$	89
5.4	Weighted majority-logic decoder for N using decoders for $N/2$. .	91
5.5	Decoding trellis for hybrid decoder ($N = 8$ and $N' = 4$)	92
5.6	FER performance with different numbers of quantization bits ($N = 64$, $K = 57$)	99
5.7	FER performance of weighted majority-logic and SC decoders ($N = 64$)	104
5.8	BER performance of weighted majority-logic and SC decoders ($N = 64$)	104
5.9	FER performance of weighted majority-logic and SC decoders ($N = 64$)	105
5.10	BER performance of weighted majority-logic and SC decoders ($N = 64$)	105
5.11	FER performance of weighted majority-logic and SC decoders ($N = 256$)	106
5.12	BER performance of weighted majority-logic and SC decoders ($N = 256$)	106
5.13	FER performance of weighted majority-logic and SC decoders ($N = 256$)	107
5.14	BER performance of weighted majority-logic and SC decoders ($N = 256$)	107
5.15	FER performance of weighted majority-logic and SC decoders ($N = 256$)	108
5.16	BER performance of weighted majority-logic and SC decoders ($N = 256$)	108
5.17	FER performance of weighted majority-logic and SC decoders ($N = 1024$)	109

5.18 BER performance of weighted majority-logic and SC decoders ($N = 1024$)	109
5.19 FER performance of hybrid decoders ($N = 8192$, $K = 6554$) . . .	111
5.20 BER performance of hybrid decoders ($N = 8192$, $K = 6554$) . . .	111
5.21 FER performance of hybrid decoders ($N = 8192$, $K = 4096$) . . .	112
5.22 BER performance of hybrid decoders ($N = 8192$, $K = 4096$) . . .	112
5.23 FER performance of hybrid-256 decoders for $N = 8192$ and $N =$ 16384	113
5.24 BER performance of hybrid-256 decoders for $N = 8192$ and $N =$ 16384	113

List of Tables

1.1	ECC Performance Metrics	3
1.2	Services and Primary Requirements	8
1.3	Examples for State-of-the-Art Turbo Decoders	10
1.4	Examples for State-of-the-Art LDPC Decoders	11
1.5	ASIC Implementation Results for Combinational SC Decoder . . .	14
1.6	ASIC Implementation Results for Combinational Weighted Major- ity Logic Decoder	15
1.7	Approximate Latency Gains	16
3.1	State-of-the-Art SC Polar Decoders on ASIC	47
3.2	State-of-the-Art BP Polar Decoders on ASIC	49
3.3	State-of-the-Art SCL Polar Decoders on ASIC	50
4.1	Schedule for Single Stage Pipelined Combinational Decoder	61
4.2	Combinational Delays of Components in $\text{DECODE}(\ell, \mathbf{a})$	66
4.3	ASIC Implementation Results	70
4.4	Power Consumption	70
4.5	Comparison with Existing Polar Decoders	72
4.6	Comparison with State-of-the-Art LDPC Decoders	75
4.7	Combinational SC Decoder FPGA Implementation Results	76
4.8	Pipelined Combinational SC Decoder FPGA Implementation Results	77
4.9	Approximate Throughput Increase for Semi-Parallel SC Decoder .	80
5.1	Number of Calculations for Block Lengths 2^2 - 2^{10}	95
5.2	Latencies of Hybrid Decoders	97
5.3	Approximate Latency Gains	98

5.4	ASIC Implementation Results	100
6.1	Comparison of State-of-the-Art ECC Decoding Schemes	116



List of Abbreviations

10GBASE-T	10 gigabit ethernet
3GPP	3rd generation partnership project
5G	5th generation mobile networks
ASIC	application specific integrated circuit
AWGN	additive white gaussian noise
B-DMC	binary-input discrete memoryless channel
BEC	binary erasure channel
BER	bit error rate
BLER	block error rate
BP	belief-propagation
CRC	cyclic redundancy check
DL	downlink
ECC	error correction coding
eMBB	enhanced mobile broad band
FER	frame error rate
FF	flip-flop

FPGA field-programmable gate array

GCC generalized concatenated codes

HD hard decision

HSPA high speed packet access

LDPC low-density parity-check

LLR log-likelihood ratio

LTE long-term evolution

LTE-A long-term evolution advanced

LUT look-up table

mMTC massive machine-type communications

NR new radio

PE processing element

RAM random access memory

SC successive cancellation

SCAN soft cancellation

SCL successive-cancellation list

SD soft decision

SNR signal to noise ratio

SSC simplified successive-cancellation

TP throughput

UAV unmanned air vehicle

UL uplink

URLLC ultra-reliable and low-latency communications

WiFi wireless fidelity

WiMAX worldwide interoperability for microwave access

WPAN wireless personal area network

XOR exclusive-or

Chapter 1

Introduction

In his seminal paper [1], C. E. Shannon introduced the concept of channel capacity as the ultimate limit at which reliable communication is possible over a noisy communications channel. The rate of information in a transmitted block is adjusted by the amount of redundancy introduced to the block. The method of introducing redundancy so as to achieve reliable communications is called Error Correction Coding (ECC).



Figure 1.1: Communication scheme with ECC

Fig. 1.1 shows a communication system employing an ECC scheme. Suppose we want to transmit a sequence of K information bits, $u_0, \dots, u_{K-1}$. The *encoder* block in the system maps the information bit sequence to a sequence of N bits $x_0, \dots, x_{N-1}$, for $N \geq K$. The sequence $x_0, \dots, x_{N-1}$ is called a *codeword*. The codeword is transmitted through a channel and a noisy version of the codeword, $y_0, \dots, y_{N-1}$, is received. A *decoder* tries to recover the information bits from the received codeword. Shannon's theorem states that by proper design of the

encoder and the decoder, the information bits can be recovered at the receiver with a vanishing error probability in the limit of large N if $R = \frac{K}{N} < C$, where R is called the *coding rate* and

$$C = \max_{p(x)} I(X; Y) \quad (1.1)$$

is the channel capacity. Here, $I(X; Y)$ is the mutual information between the channel input and output and maximization is over the all probability distributions $p(x)$ on the channel input.

Design of practical ECC methods has been a challenge ever since Shannon's paper. Until 1990's no general method was found that could achieve channel capacity. In 1993, a breakthrough in channel coding was achieved with the introduction of Turbo codes by Berrou, Glavieux, and Thitimajshima [2]. Around the same time, low-density-parity-check (LDPC) codes, originally proposed by Gallager in 1963 in his thesis [3], were rediscovered by MacKay [4] and Spielman [5]. Experiments showed that both schemes could achieve capacity with practical iterative decoding algorithms. Turbo and LDPC have been employed in many modern communication standards, such as, HSPA, WiMAX, 10GBASE-T, WiFi, LTE and LTE-A, and constitute the state-of-the-art in existing communication systems.

Although Turbo and LDPC codes achieve channel capacity for practical purposes, they have defied exact mathematical analysis due to the iterative (loopy) nature of their decoding algorithms. In fact, no code was known until the introduction of polar codes that could *provably* achieve channel capacity with low-complexity encoding and decoding algorithms. Polar codes were introduced by Arıkan [6] in 2009, along with an analytical proof showing that they achieve channel capacity in B-DMC with SC decoding. The well-defined structure and low complexity encoding and decoding algorithms made polar codes appealing for both academic research and industrial applications. Recently, polar codes have been selected as the ECC scheme for uplink (UL) and downlink (DL) control channels in the "New Radio" (NR) communications standard developed by the 3rd Generation Partnership Project (3GPP) consortium for the 5th generation of mobile communications (5G) [7].

1.1 ECC and Decoder Performances

Evaluation of an ECC and decoding scheme for any specific application is a process that requires consideration of several parameters. These parameters are listed in Table 1.1.

Table 1.1: ECC Performance Metrics

Metric	Typical Units	Explanation
Error performance	Net coding gain, BER/FER vs. SNR	Error correction capability
Throughput	Mb/s	Number of encoded / decoded bits per second
Latency	s, clock cycles, decoding steps	Duration of encoding / decoding one codeword
Power	mW	Power dissipation by encoder / decoder circuit
Area	mm ²	Area spanned by the encoder / decoder circuit
Energy-per-bit	nJ/bit	Energy required to decode one bit
Hardware efficiency	Mb/s/mm ²	Throughput per unit area
Flexibility	-	Capability of an encoder / decoder implementation to support multiple code rates and block lengths

The error performance of an ECC scheme is measured by measuring the probability of error at the decoder output. Bit error rate (BER), which is the rate of the number of erroneous bits to the number of all information bits at the decoder output, or frame error rate (FER) (also called block error rate, BLER), which is the rate of the number of decoded codewords with at least one erroneous bit to the number of all decoded codewords at the decoder output, characteristics of an ECC with a specific decoder can be used to report the error performance. We consider the error performance in Additive White Gaussian Noise (AWGN) channels in this thesis. For an AWGN channel, the error performance can be measured by plotting BER or FER against the signal-to-noise ratio (SNR) or

E_b/N_0 . The relation between SNR and E_b/N_0 is given by

$$E_b/N_0(\text{dB}) = \text{SNR}(\text{dB}) - 10 \log_{10}(\eta),$$

where η is the spectral efficiency in (b/s/Hz).

Another metric for the error performance of any ECC and decoding scheme is the *net coding gain*. The net coding gain is the difference between the E_b/N_0 values required to obtain a specific BER with and without a specific ECC and decoder scheme. As an example, Figure 1.2 shows the net coding gain obtained by a (1024, 512) polar code with SC decoding at $\text{BER}=10^{-5}$.

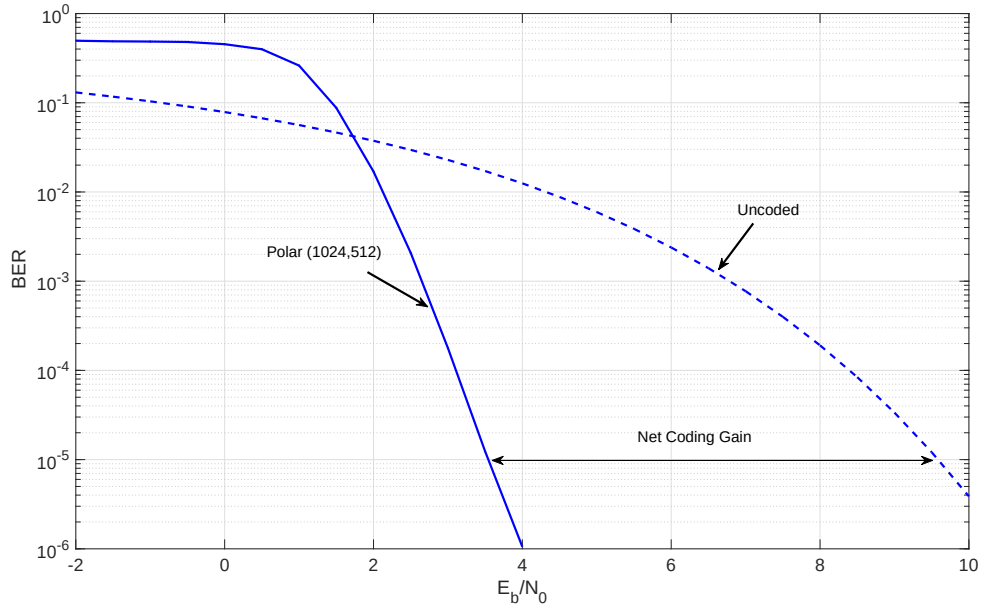


Figure 1.2: Net coding gain obtained by (1024, 512) polar code with SC decoding

Implementation procedure may change the error performance of a decoding algorithm. The number of quantization bits used to represent the real values, algorithmic alterations and analytical simplifications to simplify the decoder architecture are several causes of such changes.

The encoding and decoding complexities of an ECC determine its feasibility for industrial applications. In this thesis, we mainly focus on the decoder characteristics. The conventional method of reporting the complexity in terms of the

number of algorithmic operations is mainly oriented towards software implementations. The *algorithmic complexity* reported this way generally does not directly reflect the *hardware complexity* of a decoder implementation [8]. The hardware complexity of a decoder is not only related to the number of required calculations but also the number of memory elements, data transfers, interconnect network, etc. in the circuit.

Hardware complexity effects the throughput, hardware and power consumptions of any decoder implementation. In order to analyze the hardware complexity and perform fair comparisons between different decoder implementations, two meaningful metrics have been proposed in [8]; those are

$$\begin{aligned} \text{Energy Efficiency}[\text{bit/nJ}] &= \frac{\text{Throughput}[\text{Mb/s}]}{\text{Power}[\text{mW}]}, \\ \text{Area Efficiency}[\text{Mb/s/mm}^2] &= \frac{\text{Throughput}[\text{Mb/s}]}{\text{Area}[\text{mm}^2]}. \end{aligned} \quad (1.2)$$

It has been shown in [8] that the metrics in (1.2) return meaningful comparison results between different decoder implementations. In this thesis, we use the inverse of energy-efficiency metric and call it “energy-per-bit”, and use the area-efficiency metric synonymously with “hardware efficiency”.

Latency is a characteristic that depends on both the definition and implementation of a decoding algorithm, similar to the hardware complexity. Latency measures the decoding cycles, clock cycles or time required for any decoder algorithm or implementation to complete its process. Throughput and latency are most generally inversely proportional metrics in decoder implementations; an example for exceptions is completely pipelined decoder architectures. The latency of a decoder measures the duration that takes a decoder to complete one decoding process. Throughput measures the “speed” of a decoder using the number of decoded bits in a second. The explained relationship in implementations is illustrated in Fig. 1.3.

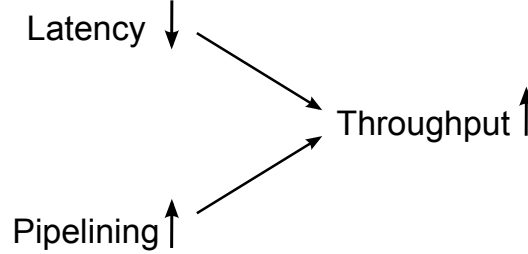


Figure 1.3: Latency, pipelining and throughput

Generally, decoder architectures with low latencies are sought for applications with high throughput requirements. There are also applications with low-latency decoding as a primary requirement. An example is the Ultra-Reliable Low-Latency Communications (URLLC) service of the new generation mobile communications standard, intended for applications, such as, real-time industrial/robotic control applications [9].

Flexibility represents the ability of a decoder implementation for a given ECC to decode codes with different block lengths and/or code rates. The flexibility of a decoder affects all implementation metrics mentioned above and it should be taken into account in comparisons between different decoder implementations [8], [10]. A decoder optimized for a fixed code (block length and code rate) can outperform a flexible decoder in terms of complexity and throughput; however, in many applications flexible ECC implementations are desired. Thus, flexibility of an ECC implementation is an indispensable measure of performance in modern communication systems.

There are also factors related to the hardware platform that determine the performance of any decoder implementation. For ASIC, the implementation performance is heavily related with the preferred VLSI technology. The achievable clock frequency and throughput improves with improving CMOS technology due to the reduced critical path delays. The area spanned by the circuits decreases due to the reduced dimensions. The dynamic power consumption is also improved as the supply voltage can be reduced without a penalty in the achievable

frequency with respect to older technologies [11]. Similar arguments are applicable to FPGA. However, due to the pre-determined routing paths in the chips and the varying difficulties of place-and-route processes in different architectures and chip sizes, the improvements may not be identical to those in ASIC depending on the implementation characteristics.

1.2 Background and Motivation for the Thesis

We explain the requirements for decoder implementations targeting various existing and emerging communications services. Then, we summarize the state-of-the-art in ECC and decoder implementation schemes and give the motivations for the studies in this thesis.

Table 1.2 lists a number of telecommunication services and their primary requirements. The first three scenarios given in the table are data services for mobile communications standards. The primary decoder requirements for the data scenarios of LTE and LTE-A are specified to be peak throughputs of 300 Mb/s and 1 Gb/s for DL, respectively. In the NR standard, the throughput requirement for the data scenario (Enhanced Mobile Broad Band (eMBB) data) is determined to be 20 Gb/s in DL [12]. Energy-efficient decoding has become more crucial in this scenario due to the increased throughput requirement. For example, a rough calculation reveals an energy-per-bit requirement of 50 pJ/b or less [13].

In the NR standard, several other scenarios are aimed to be supported. URLLC and Massive Machine-Type Communications (mMTC) are two such scenarios that are listed in Table 1.2. URLLC targets real-time control applications. The key requirements are low latency in encoding/decoding processes and good error performance with an achievable BER requirement below 10^{-5} [9]. The aim in mMTC scenario is to provide continuous and ubiquitous coverage with massive number of devices connected. In common mMTC scenarios, the connected devices are assumed to be battery-powered that are expected to run for at least 10 years [12]. Throughput and latency requirements are more relaxed for the mMTC

Table 1.2: Services and Primary Requirements

Service	Primary Requirements
LTE Data (DL/UL)	Peak throughput=300/75 Mb/s High coding gain Flexibility
LTE-A Data (DL/UL)	Peak throughput=1/0.5 Gb/s High coding gain Flexibility
NR eMBB Data (DL/UL)	Peak throughput=20/10 Gb/s High coding gain High energy-efficiency in decoder High hardware-efficiency in decoder Flexibility
NR URLLC (DL/UL)	Low decoder latency $\text{BER} \leq 10^{-5}$ Flexibility
NR mMTC DL	High energy-efficiency in decoder High hardware-efficiency in decoder Flexibility
NR mMTC UL	High coding gain Low complexity in encoder Flexibility
Optical Communications	Peak throughput ≥ 100 Gb/s $\text{BER} \leq 10^{-15}$ High coding gain High energy-efficiency in decoder High hardware-efficiency in decoder
Data Kiosk/ Terahertz Communications	Peak throughput ≥ 1 Tb/s High energy-efficiency in decoder

scenario compared to the eMBB data and URLLC scenarios. Depending on the service being UL or DL, the important requirements are good error performance, low encoding/decoding hardware complexities and high energy efficiency [14].

Next generation optical systems aim to surpass the throughput limit of 100 Gb/s. The ECC schemes that are going to be used in such systems will be named as “The 3rd Generation Forward Error Correction (FEC)”. The primary requirements for the 3rd Generation FEC are a net coding gain greater than 10 dB at a BER level of 10^{-15} at the decoder output, a redundancy percentage (overhead) up to 20% and a throughput value exceeding 100 Gb/s. The desired coding gain is shown to be achievable by soft-decision (SD) decoding algorithms [15]. As the required BER is smaller than 10^{-15} , ECC with no or very low error floors are sought for. Energy efficiency is a key requirement to support such high throughput values and expected to be ≤ 10 pJ/b [16].

The peak throughput requirements for the next generation communication systems are predicted to be on the order of Tb/s [17] - [21]. According to [18], the areas of wireless communications demanding such high throughput values are wireless back-haul links and data access provided via unmanned air vehicles (UAVs) and satellites. Data kiosk services are pointed out in [20] as an application which requires Tb/s throughput on short links. A data kiosk is a machine that transfers large amounts of data (*e.g.*, a movie) to a user device (*e.g.*, a mobile phone) in a very short time period and over short distances (≤ 1 m). Net coding gain is not a crucial requirement since the transmission distance is very small. Another service with Tb/s throughput requirement over short distances is the communications between chips and boards in a computer or data centers [20]. Such applications are also in the study field of IEEE 802.15 WPAN THz Interest Group.

1.2.1 State-of-the-Art in ECC and Motivation

Turbo codes and Turbo decoding architectures have been studied for a long time in the scope of practical applications. The characteristics of the codes

with rate matching methods are well-known and decoder implementations have matured. They have been employed in several existing communication standards, including DVB-RCS, HSPA, WiMAX, LTE and LTE-A. In order to meet high data rate requirements of new generation standards, parallel architectures for Turbo decoders have been proposed and studied extensively [22]. Table 1.3 gives ASIC implementation results for several state-of-the-art parallel Turbo decoders.

Table 1.3: Examples for State-of-the-Art Turbo Decoders

	[22]	[23]	[24]
Technology	45 nm/0.81 V	65 nm/1.2 V	65 nm/1.08 V
Parallelism	64	16	6144
Iterations	5.5	11	39
Block Lengths	All LTE Block Lengths	All LTE Block Lengths	6144
Code Rates	All LTE Code Rates	All LTE Code Rates	-
Freq. [MHz]	600	410	100
Area [mm ²]	2.43	2.49	109
Power [mW]	870	1894*	9618
TP [Gb/s]	1.67	1.01	15.8
Hard. Eff. [Gb/s/mm ²]	0.68	0.41	0.145
Engy.-per-bit [pJ/b]	521*	1870	608

* Calculated from the presented results

The main drawback of Turbo codes is the lack of flexible decoder implementations that can support the increasing throughput requirements with reasonable power consumption levels. The reasons for the problems of Turbo decoders are addressed as diminishing throughput returns with increasing number of parallel SISO decoders in [23] and memory conflict problem due to concurrent memory reading/writing in parallel Turbo decoding architectures in [22].

LDPC codes can be considered as the strongest candidates for the emerging communications standards with their error and decoder performances. They have been employed in several existing standards; DVB, WiMAX, 10GBASE-T and

WiFi being among the most notable ones. The most commonly used decoding method for LDPC codes is the Belief Propagation (BP) decoding algorithm. Compared to the state-of-the-art Turbo decoders, state-of-the-art BP LDPC decoders provide higher throughput and energy-efficiency with competitive error performance [10], [13]. Table 1.4 gives several state-of-the-art LDPC decoders. One can observe from the Tables 1.3 and 1.4 that LDPC decoders can achieve higher throughput with better hardware and energy efficiencies than those of Turbo decoders.

Table 1.4: Examples for State-of-the-Art LDPC Decoders

	[25]	[26]	[27]
Technology	28 nm/1.1 V	65 nm/1.1 V	65 nm/-
Algorithm	Min-Sum	1's Complement	Min-Sum
Architecture	Semi-parallel Layered	Pipelined Layered	Layered
Iterations	3.75	7	10
Block Lengths / Standard	672 / IEEE 802.11ad	672 / IEEE 802.11ad	2304 / -
Code Rates	1/2, 5/8, 3/4, 13/16	1/2, 5/8, 3/4, 13/16	1/2 - 1
Freq. [MHz]	260	400	1100
Area [mm ²]	0.63	0.575	1.96
Power [mW]	180*	273**	908
TP [Gb/s]	12	9.25	1.28
Hard. Eff. [Gb/s/mm ²]	19	16.08	0.65
Engy.-per-bit [pJ/b]	30*	29.4	709

* Power consumption is for rate-1/2 code at a BER of 10^{-6} to 10^{-7}

** Power consumption is for rate-1/2 code at SNR 2.5 dB

Several issues have been addressed for LDPC codes and decoders. One important issue is about the characteristics of the LDPC decoders: it is still not clear whether LDPC decoders can preserve their good characteristics in more flexible implementations [13]. Another issue is about the error floor problem of LDPC codes. For services with low FER/BER requirements, such as optical communications, LDPC codes performing with low error floor and their decoders with

good characteristics are sought for [28], [15].

Polar codes may overcome the problems of Turbo and LDPC decoders with low-complexity and efficient decoders, and error performance characteristics without any error floors. However, the state-of-the-art polar decoders have not yet been shown to achieve implementation performances that can compete with the state-of-the-art LDPC decoders with flexible implementations, as will be demonstrated in Chapter 3. In this thesis, we aim to design high-throughput, low-latency and energy-efficient polar decoders. The decoders we propose are especially suitable for, but not limited to, services such as mMTC, optical communications and Terahertz communications. It was shown in [16] that polar codes outperform the 2nd Generation FEC in optical communications with SC decoding. Therefore, polar codes can be considered as candidates for 3rd Generation FEC even with low-complexity SC decoding algorithm. They are also good candidates for wireless communication applications that require energy-efficient decoding, such as, mMTC. Furthermore, we aim to reduce the decoding latency further to improve the throughput of polar decoders for very high throughput services, such as Terahertz communications. The proposed decoders are also suitable for any communications service with high throughput and energy-efficiency requirements. We investigate the characteristics of the decoders in an effort to demonstrate that polar codes are promising ECC candidates for the emerging application areas along with LDPC codes.

1.3 Contributions of the Thesis

The contributions of the thesis are given in 2 parts. In the first part (Chapter 4), we propose a novel SC decoder architecture that achieves the highest throughput and energy-efficiency among the state-of-the-art SC polar decoders while preserving the inherent flexibility of polar codes with SC decoding. In the second part (Chapter 5), we investigate the majority-logic decoding algorithm for polar codes in an effort to reduce the decoding latency.

1.3.1 Combinational SC Decoder

We propose a novel SC decoder composed of only combinational circuitry, which is possible thanks to the feed-forward (non-iterative) and recursive structure of the SC algorithm. We name the proposed decoder as *combinational SC decoder*. Combinational SC decoders operate at lower clock frequencies compared to ordinary synchronous (sequential logic) decoders. However, in a combinational SC decoder, an entire codeword is decoded in one clock cycle. This allows combinational SC decoders to operate with less dynamic power consumption while maintaining a high throughput. Furthermore, the combinational SC decoders retain the inherent flexibility of polar coding to operate at any desired code rate for a given block length.

We give analytical estimates for the hardware consumption and combinational delay of the proposed decoder in terms of the parameters of basic circuit elements. The hardware consumption is calculated by finding the number of comparator and adder/subtractor blocks in the circuit and shown to be

$$N \left(\frac{3}{2} \log N - 1 \right).$$

We show that the combinational delay, D_N , can be written as

$$D_N = N \left(\frac{3\delta_m}{2} + \delta_c + \delta_x + \frac{\delta_a}{2} \right) - [\delta_c + 2\delta_m + (\log N + 1) \delta_x] + T_N,$$

where δ_m , δ_c , δ_x , δ_a and T_N are the delays of a multiplexer, a comparator, a 2-input XOR gate and the overall interconnect network, respectively.

Post-synthesis ASIC implementation results for the combinational SC decoder are given in Table 1.5 for 90 nm 1.3 V technology. We also apply technology conversion to the results to show that the proposed decoders can achieve more than 8 Gb/s throughput with an energy requirement on the order of pJ/b in 28 nm technology. Table 1.5 summarizes the implementation results of combinational SC decoder for block length 1024.

We compare the ASIC implementation results of combinational SC decoders with those of the state-of-the-art polar and LDPC decoders. The results show that

Table 1.5: ASIC Implementation Results for Combinational SC Decoder

(N, K)	Tech.	Freq. [MHz]	TP [Gb/s]	Power [mW]	Engy./bit [pJ/b]	Hard. Eff. [Gb/s/mm ²]
(1024, Any)	90 nm, 1.3 V	2.5	2.56	190.7	74.5	0.8
	28 nm, 1.0 V [†]	-	8.22	38.0	4.6	26.4

[†] Technology conversion by analytical formulas in [29] and [30]

the combinational SC decoders achieve highest throughput and energy-efficiency among the SC decoder architectures proposed so far. The results also show that combinational SC decoders have comparable performance with BP polar and LDPC decoders in terms of throughput, error performance and energy-efficiency with a high flexibility. The promising results imply that combinational SC decoders are good candidates as polar decoder architectures for high throughput applications.

We investigate pipelining with combinational SC decoders and provide FPGA implementation results for both combinational and pipelined combinational decoders. The results show that the a one stage pipelined combinational SC decoder can achieve a throughput of 1.24 Gb/s for block length 1024 on FPGA. We also propose the combinational SC decoder as an “accelerator” module as part of a novel hybrid decoder that combines a synchronous SC decoder with a combinational decoder to take advantage of the best characteristics of the two types of decoders. Such decoders, named *hybrid-logic decoders*, extend the range of applicability of the purely combinational design to very large block lengths. We give analytical estimates for the throughput gain obtained by such decoders in terms of the decoder latencies.

1.3.2 Weighted Majority-Logic Decoding of Polar Codes

We investigate weighted majority-logic algorithm of [31] to decode polar codes. First, we introduce a novel recursive definition for the weighted majority-logic

algorithm for the bit-reversed polar codes (we summarize the conventional definition of majority-logic decoding in Section 3.1.3) for implementation purposes. We present analytical estimates for the complexity and latency of weighted majority-logic algorithm with the introduced definition. We show that the algorithmic complexity of the decoder is

$$C_N = 2(N^{\log 3} - N),$$

and the latency is

$$L_N = \frac{\log^2 N + 3 \log N}{2}$$

for block length N . The drawback of such decoders is shown to be the error performance loss with respect to SC decoding, which is dependent on the block length, code rate and optimization SNR values of the polar codes.

Based on the introduced recursive definition, we implement the weighted majority-logic decoders using only combinational circuitry on ASIC. We name the proposed decoder as *combinational weighted majority-logic decoder*. Table 1.6 shows the weighted majority-logic decoder implementation results for block length 256.

Table 1.6: ASIC Implementation Results for Combinational Weighted Majority Logic Decoder

(N, K)	Tech.	Freq. [MHz]	TP [Gb/s]	Power [mW]	Engy./bit [pJ/b]	Hard. Eff. [Gb/s/mm ²]
(256, Any)	90 nm, 1.3 V	68.0	17.4	1960	112.6	5.7
	28 nm, 1.0 V [†]	-	55.9	360.8	6.4	190.7

[†] Technology conversion by analytical formulas in [29] and [30]

We develop a decoder that employs a weighted majority-logic decoder as an “accelerator” module in a decoder structure employing both SC and weighted majority-logic decoders. We name the proposed decoder as *hybrid decoder*. The hybrid decoder aims to introduce a trade-off between the decoder latency and error performance in decoding of polar codes. We derive an analytical formula

for the latency of hybrid decoders as

$$L_N = \frac{N}{N'} \left(2 + \frac{\log N'(\log N' + 3)}{2} \right) - 2,$$

where N' is the component code block length for which weighted majority-logic decoding is employed in the hybrid decoder. Table 1.7 shows the approximate latency gain values obtained by hybrid decoding with respect to SC decoding for different N' values. We show by simulations that the error performance loss can be reduced significantly by hybrid decoders with properly designed polar codes for large block lengths.

Table 1.7: Approximate Latency Gains

	N'			
	1 (SC)	64	128	256
Latency Gain	1	4.4	6.9	11.1

1.4 Outline of the Thesis

We give background information on polar codes and SC decoding in Chapter 2. In Chapter 3, we summarize SC List (SCL) (Section 3.1.1), BP (Section 3.1.2) and majority-logic (Section 3.1.3) decoding algorithms. We also summarize the state-of-the-art polar decoder implementations and point out the throughput bottleneck problem of SC decoders (Section 3.2).

In Chapter 4, we introduce the proposed architectures for SC decoding of polar codes. We start with the description of combinational SC decoder in Section 4.1. We introduce pipelined combinational SC decoders and hybrid-logic decoders in Sections 4.1.3 and 4.1.4, respectively. We present formulas for the complexity and combinational delay of the combinational SC decoders in Section 4.2. Detailed implementation results for ASIC and FPGA are presented in Section 4.3. We also compare the implementation results of the combinational SC decoders with state-of-the-art polar and LDPC decoders in Sections 4.3.1.3 and 4.3.1.4, respectively. An analytical analysis for the throughput improvement by hybrid-logic decoders with respect to the synchronous decoders is given in Section 4.4.

Chapter 5 starts with the recursive definition for the weighted majority-logic algorithm for bit-reversed polar codes (Section 5.1.1). We introduce the hybrid decoder in Section 5.1.2. The complexity and latency analyses for the proposed decoders are given in Section 5.2. We present the implementation results of weighted majority-logic decoding in Section 5.3 and analyze the error performances of the weighted majority-logic and hybrid decoders in Section 5.4.

The thesis is concluded with Chapter 6, where we compare examples of the state-of-the-art decoder implementations for Turbo, LDPC and polar codes and the proposed decoders. We also give suggestions on new research directories related with the topics of the thesis.

Chapter 2

Background on Polar Coding

In this chapter, we introduce the notation and give background information on the basics related to the polar codes.

2.1 Notations and Preliminaries

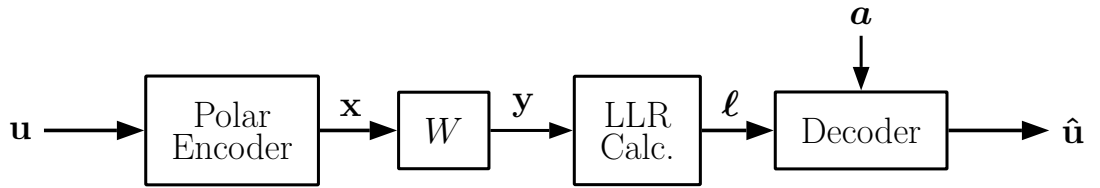


Figure 2.1: Communication scheme with polar codes

In this thesis, we consider the system given in Fig. 2.1, in which a polar code is used for channel coding. The block length of a polar code is represented by $N = 2^m$, where m is an integer and $m > 0$. The signals denoted by boldface lowercase letters in the system are vectors. The uncoded bit vector $\mathbf{u} \in \mathbb{F}_2^N$, consisting of both information and redundant bits, is input to the polar encoder for channel coding. The output codeword, $\mathbf{x} \in \mathbb{F}_2^N$, is transmitted through the channel. The channel W in the system is an arbitrary memoryless channel with input alphabet

$\mathcal{X} = \{0, 1\}$, output alphabet $\mathcal{Y}$ and transition probabilities $\{W(y|x) : x \in \mathcal{X}, y \in \mathcal{Y}\}$. In each use of the system, a codeword is transmitted and a channel output vector $\mathbf{y} \in \mathcal{Y}^N$ is received. The receiver first calculates the log-likelihood ratio (LLR) vector $\ell = (\ell_1, \dots, \ell_N)$ with

$$\ell_i = \ln \left(\frac{W(y_i|x_i = 0)}{W(y_i|x_i = 1)} \right), \quad (2.1)$$

for each element of the channel output vector and feeds it into a decoder for polar codes. The decoder is also given the *frozen-bit indicator vector* $\mathbf{a}$, which is a 0-1 vector of length N with

$$a_i = \begin{cases} 0, & \text{if } i \in \mathcal{A}^c \\ 1, & \text{if } i \in \mathcal{A}. \end{cases}$$

Throughout this thesis, all matrix and vector operations are over vector spaces over the binary field $\mathbb{F}_2$. Addition over $\mathbb{F}_2$ is represented by the $\oplus$ operator. The logarithms are in base-2 unless stated otherwise. For any set $\mathcal{S} \subseteq \{0, 1, \dots, N-1\}$, $\mathcal{S}^c$ denotes its complement. For any vector $\mathbf{u} = (u_0, u_1, \dots, u_{N-1})$ of length N and set $\mathcal{S} \subseteq \{0, 1, \dots, N-1\}$, $\mathbf{u}_{\mathcal{S}} \stackrel{\text{def}}{=} [u_i : i \in \mathcal{S}]$. We define the *sign* function $s : \mathbb{R} \rightarrow \{0, 1\}$ as

$$s(\alpha) = \begin{cases} 0, & \text{if } \alpha \geq 0 \\ 1, & \text{otherwise.} \end{cases} \quad (2.2)$$

We introduce two channel parameters for any B-DMC W : the symmetric capacity

$$I(W) = \sum_{y \in \mathcal{Y}} \sum_{x \in \{0,1\}} \frac{1}{2} W(y|x) \log \frac{W(y|x)}{\frac{1}{2}W(y|0) + \frac{1}{2}W(y|1)} \quad (2.3)$$

and the Bhattacharyya parameter

$$Z(W) = \sum_{y \in \mathcal{Y}} \sqrt{W(y|0)W(y|1)} \quad (2.4)$$

which measure rate and reliability of the channel, respectively. Both parameters take values in $[0, 1]$ and are inversely proportional.

2.2 Polar Codes

Polar codes were proposed in [6] as a low-complexity channel coding method that can provably achieve Shannon's channel capacity for any B-DMC W . The codes create N synthetic channels from N independent uses of such channel, which turn out to be less or more noisy than the original channel.

Channel polarization consists of a *channel combining* and a *channel splitting* process. For the explanations of the mentioned concepts, we follow the notation in [6] and use $\mathbf{c}_1^N$ to denote the vector of length N with elements c_i , for $1 \leq i \leq N$. The channel combining process combines N independent copies of W by a transformation operation and produces a vector channel

$$W_N : \mathcal{X}^N \rightarrow \mathcal{Y}^N,$$

for which the transition probability can be written as

$$W_N(\mathbf{y}_1^N | \mathbf{u}_1^N) = W^N(\mathbf{y}_1^N | \mathbf{u}_1^N G_N), \quad \mathbf{y}_1^N \in \mathcal{Y}^N, \mathbf{u}_1^N \in \mathcal{X}^N. \quad (2.5)$$

The matrix G_N is the transformation matrix applied to the bit vector to be transmitted over W . The channel splitting process splits the combined vector channel W_N back into a set of N binary-input synthetic channels

$$W_N^{(i)} : \mathcal{Y}^N \times \mathcal{X}^{i-1}, 0 \leq N-1$$

where

$$W_N^{(i)}(\mathbf{y}_1^N, \mathbf{u}_1^{i-1} | u_i) = \sum_{\mathbf{u}_{i+1}^N \in \mathcal{X}^{N-i}} \frac{1}{2^{N-i}} W^N(\mathbf{y}_1^N | \mathbf{u}_1^N). \quad (2.6)$$

Channel combining is established by the polar encoder at the transmitter and channel splitting by a genie aided SC decoder at the receiver.

We demonstrate the polarization effect with an example. Consider the channel combining process depicted in Fig. 2.2 for $N = 2$. Assume $\mathbf{u}_1^2$ is uniform on $\mathcal{X}^2$. The operation in Fig. 2.2 creates the channel vector $W_2 : \mathcal{X}^2 \rightarrow \mathcal{Y}^2$, for which the transition probabilities are given as

$$W_2(y_1, y_2 | u_1, u_2) = W(y_1 | u_1 \oplus u_2) W(y_2 | u_2)$$

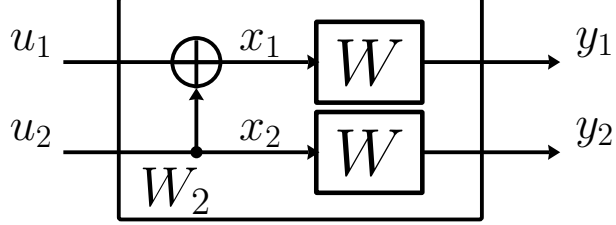


Figure 2.2: Channel combining process ($N = 2$)

We can also write the transformation in Fig. 2.2 in the vector-matrix multiplication form as

$$\begin{bmatrix} u_1 & u_2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (2.7)$$

so that

$$W_2(y_1, y_2 | u_1, u_2) = W^2(\mathbf{y}_1^2 | \mathbf{u}_1^2 G_2)$$

In order to complete the channel polarization process, we move to the channel splitting phase. Without any prior information on the values of u_1 and u_2 and assuming equal likely transmitted bits, the transition probability for the first synthetic channel $W_2^{(1)}$ can be written as

$$\begin{aligned} W_2^{(1)}(\mathbf{y}_1^2 | u_1) &= \sum_{u_2 \in \mathcal{X}} \frac{1}{2} W(y_1 | u_1 \oplus u_2) W(y_2 | u_2) \\ &= \frac{1}{2} W(y_1 | u_1) W(y_2 | 0) + \frac{1}{2} W(y_1 | u_1 \oplus 1) W(y_2 | 1). \end{aligned} \quad (2.8)$$

The estimate for u_1 , $\hat{u}_1$, can be given by observing the values of $W_2^{(1)}(\mathbf{y}_1^2 | 0)$ and $W_2^{(1)}(\mathbf{y}_1^2 | 1)$.

Assume the correct value of u_1 is provided for the second synthetic channel $W_2^{(2)}$ by the genie-aided decoder. With the perfect knowledge on u_1 , we can write the transition probability for $W_2^{(2)}$ as

$$W_2^{(2)}(\mathbf{y}_1^2, u_1 | u_2) = \frac{1}{2} W(y_1 | u_1 \oplus u_2) W(y_2 | u_2). \quad (2.9)$$

It is proved in [6] that the relations between the capacities of the original and

synthetic channels are expressed as

$$\begin{aligned} I(W_2^{(1)}) &\leq I(W) \leq I(W_2^{(2)}), \\ I(W_2^{(1)}) + I(W_2^{(2)}) &= 2I(W). \end{aligned} \quad (2.10)$$

The expressions (2.10) show that the total capacity is preserved when channel polarization occurs and one synthetic channel yields a higher capacity than the original channel while the other yields a lower value. A similar relation is derived in terms of the Bhattacharyya parameters of the channels as

$$\begin{aligned} Z(W_2^{(1)}) &\geq Z(W) \geq Z(W_2^{(2)}), \\ Z(W_2^{(1)}) + Z(W_2^{(2)}) &\leq 2Z(W). \end{aligned} \quad (2.11)$$

with the inequality in the second expression satisfied iff W is binary erasure channel (BEC).

If one wants to transmit a single bit of information using the above polarization scheme, the information is loaded on u_2 and transmitted through the more reliable synthetic channel $W_2^{(2)}$. The other bit, u_1 , is chosen as a *frozen* bit and assigned a value which is also known by the decoder. It is used in the decoder to recover the information. The channel transformation scheme described above can be generalized recursively by the formulas [6]

$$\begin{aligned} W_{2N}^{(2i-1)}(\mathbf{y}_1^{2N}, u_1^{2i-2} | u_{2i-1}) &= \sum_{u_{2i}} \frac{1}{2} W_N^{(i)}(y_1^N, u_{1,o}^{2i-2} \oplus u_{1,e}^{2i-2} | u_{2i-1} \oplus u_{2i}) \\ &\quad \cdot W_N^{(i)}(y_{N+1}^{2N}, u_{1,e}^{2i-2} | u_{2i}) \\ W_{2N}^{(2i)}(\mathbf{y}_1^{2N}, u_1^{2i-1} | u_{2i}) &= \frac{1}{2} W_N^{(i)}(y_1^N, u_{1,o}^{2i-2} \oplus u_{1,e}^{2i-2} | u_{2i-1} \oplus u_{2i}) \\ &\quad \cdot W_N^{(i)}(y_{N+1}^{2N}, u_{1,e}^{2i-2} | u_{2i}), \end{aligned}$$

for $1 < i < N$, so that we obtain the $2N$ synthetic channels in $\log N + 1$ recursions. Then, the transformations of $I(W_N^{(i)})$ and $Z(W_N^{(i)})$ are written as

$$\begin{aligned} I(W_N^{(2i-1)}) &\leq I(W_{N/2}^{(i)}) \leq I(W_N^{(2i)}), \\ I(W_N^{(2i-1)}) + I(W_N^{(2i)}) &= 2I(W_{N/2}^{(i)}). \end{aligned} \quad (2.12)$$

and

$$\begin{aligned} Z(W_N^{(2i-1)}) &\geq Z(W_{N/2}^{(i)}) \geq Z(W_N^{(2i)}), \\ Z(W_N^{(2i-1)}) + Z(W_N^{(2i)}) &\leq 2Z(W_{N/2}^{(i)}). \end{aligned} \quad (2.13)$$

It is proved in [6] that for any B-DMC W , the synthetic channels $W_N^{(i)}$ polarize. For any fixed $\delta \in \{0, 1\}$, the fraction of synthetic channels for which $I(W_N^{(i)}) \in (1 - \delta, 1]$ goes to $I(W)$ and the fraction for which $I(W_N^{(i)}) \in [0, \delta)$ goes to $1 - I(W)$ as N goes to infinity. In other words, almost all synthetic channels become either completely noiseless or noisy and the number of noiseless channels scales as $NI(W)$ as N goes to infinity. Polar coding rule suggests transmitting data on the noiseless synthetic channels and *freezing* the inputs of the noisy synthetic channels to values that are known and used by the decoder. Based on this polarization phenomenon, data transmission with rate $R < I(W)$ can be achieved with a block error probability

$$P_e(N, R) = \mathcal{O}\left(2^{-N^\beta}\right),$$

for any $\beta < 1/2$ [32].

2.2.1 Code Construction

For any (N, K) polar code, the encoder input vector $\mathbf{u} \in \mathbb{F}_2^N$ is separated into a data part $\mathbf{u}_{\mathcal{A}}$ of K elements and a frozen part $\mathbf{u}_{\mathcal{A}^c}$ of $N - K$ elements. It is proved in [6] that the block error probability for any B-DMC W under SC decoding is upper bounded as

$$P_e(N, K, \mathcal{A}, \mathbf{u}_{\mathcal{A}^c}) \leq \sum_{i \in \mathcal{A}} Z(W_N^{(i)}).$$

Thus, the elements of the sets $\mathcal{A}$ and $\mathcal{A}^c$ can be determined from the Bhattacharyya parameters of each synthetic channel for a given original channel. More specifically, the K bit locations with lowest Bhattacharyya parameters are assigned to $\mathcal{A}$ as information bit locations. The rest are assigned to $\mathcal{A}^c$ as frozen bit locations.

For the case of W being BEC, the Bhattacharyya parameters can be calculated analytically using the recursive formulas given in [6], such that

$$\begin{aligned} Z(W_N^{(2i-1)}) &= 2Z(W_{N/2}^{(i)}) - Z(W_{N/2}^{(i)})^2 \\ Z(W_N^{(2i)}) &= Z(W_{N/2}^{(i)})^2 \end{aligned}$$

For general W , a Monte Carlo approach was proposed in [6], which is a simulation based method to determine the reliabilities of the synthetic channels with complexity $\mathcal{O}(MN \log N)$, where M is the number of Monte Carlo runs. Due to the Monte Carlo method having a high complexity order, several other methods have been proposed to construct polar codes, such as density evolution ([33], [34], [35]) and Gaussian approximation ([36], [37], [38]). We adopt the Monte Carlo approach to determine the bit locations in the thesis. We also fix the frozen part $\mathbf{u}_{\mathcal{A}^c}$ to zero for implementation purposes.

2.2.2 Encoding

We present different methods to describe the polar encoding operation for generic N that are relevant for our studies. The first method is the generalization of the expression in (2.7). For generic $N = 2^m$, the encoding operation of polar codes can be written in vector-matrix multiplication form as

$$\mathbf{x} = \mathbf{u} \mathbf{G}_N, \quad (2.14)$$

where

$$\mathbf{G}_N = \mathbf{B}_N \mathbf{F}^{\otimes m} \quad (2.15)$$

and

$$\mathbf{F} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \quad (2.16)$$

and $\mathbf{F}^{\otimes m}$ is m th Kronecker power of the kernel matrix $\mathbf{F}$. The matrix $\mathbf{B}_N$ is the bit-reversal matrix for a vector of length N . Denote the binary representation of an integer $k \in \{0, \dots, N-1\}$ by $(i_0, \dots, i_{m-1})$. Vectors $\mathbf{a}$ and $\mathbf{b}$ of length- N

have the relation $a_{(i_0, \dots, i_{m-1})} = b_{(i_{m-1}, \dots, i_0)}$ if $\mathbf{a} = \mathbf{bB}_N$. It should be noted here that polar codes can be defined without the bit-reversal operation without changing any code properties other than the locations of information and redundant bits.

We demonstrate the process with an example for block length 8. The Kronecker power 3 of the kernel matrix $\mathbf{F}$ is given in (2.16).

$$\mathbf{F}^{\otimes 3} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad (2.17)$$

Then, the encoding operation with bit-reversal for $N = 8$ becomes

$$\begin{bmatrix} u_0 & u_1 & u_2 & u_3 & u_4 & u_5 & u_6 & u_7 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix}^T \quad (2.18)$$

The vector-matrix multiplication given above can be represented by the encoding graph given in Fig. 2.3. From the graph, one can observe that the polar encoding operation can be performed with an algorithmic complexity of $\mathcal{O}(N \log N)$ [6].

Next, we present the recursive definition for polar encoding. Algorithm 1 gives the recursive definition of polar encoding for block length N . The vectors $\mathbf{u}_{\mathcal{O}_N}$ and $\mathbf{u}_{\mathcal{E}_N}$ in Algorithm 1 represent the vectors of odd and even indexed uncoded

bits, respectively. Algorithm 1 states that one can obtain a polar encoder function for block length N using two polar encoder functions for block length $N/2$.

Finally, we present the concatenated code form for polar encoding. Polar codes are a class of generalized concatenated codes (GCC). More precisely, a polar code $\mathcal{C}$ of length- N is constructed from two length- $N/2$ codes $\mathcal{C}_1$ and $\mathcal{C}_2$, using the well-known Plotkin $|\mathbf{u}| \mathbf{u} + \mathbf{v}|$ code combining technique [39]. The constituent codes $\mathcal{C}_1$ and $\mathcal{C}_2$ are polar codes in their own right and each can be further decomposed into two polar codes of length $N/4$, and so on, until the block length is reduced to one. The GCC structure is illustrated in Fig. 2.4, which shows that a polar code $\mathcal{C}$ of length $N = 8$ can be seen as the concatenation of two polar codes $\mathcal{C}_1$ and $\mathcal{C}_2$ of length $N' = N/2 = 4$, each.

The dashed boxes in Fig. 2.4 represent the component codes $\mathcal{C}_1$ and $\mathcal{C}_2$. The input bits of the component codes are $\hat{\mathbf{u}}^{(1)} = (\hat{u}_0^{(1)}, \dots, \hat{u}_3^{(1)}) = (\hat{u}_0, \dots, \hat{u}_3)$ and $\hat{\mathbf{u}}^{(2)} = (\hat{u}_0^{(2)}, \dots, \hat{u}_3^{(2)}) = (\hat{u}_4, \dots, \hat{u}_7)$ for $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively. For a polar code of block length 8 and $R = 1/2$, the frozen bits are $\hat{u}_0$, $\hat{u}_1$, $\hat{u}_2$, and $\hat{u}_4$. This makes 3 input bits of $\mathcal{C}_1$ and 1 input bit of $\mathcal{C}_2$ frozen bits; thus, $\mathcal{C}_1$ is a $R = 3/4$ code with $\hat{u}_0^{(1)}$, $\hat{u}_1^{(1)}$, $\hat{u}_2^{(1)}$ and $\mathcal{C}_2$ is a $R = 1/4$ code with $\hat{u}_0^{(2)}$ frozen.

Encoding of $\mathcal{C}$ is done by first encoding $\hat{\mathbf{u}}^{(1)}$ and $\hat{\mathbf{u}}^{(2)}$ separately using encoders for block length 4 and obtain coded outputs $\hat{\mathbf{x}}^{(1)}$ and $\hat{\mathbf{x}}^{(2)}$. Then, each pair of coded bits $(\hat{\mathbf{x}}_i^{(1)}, \hat{\mathbf{x}}_i^{(2)})$, $0 \leq i \leq 3$, is encoded again using encoders for block length 2 to obtain the coded bits of $\mathcal{C}$.

2.2.3 Successive-Cancellation Decoding

The decoding algorithm considered in [6] for polar codes is SC, which is a low-complexity algorithm. An SC decoder takes the channel output LLRs and the frozen-bit locations as inputs and calculates the bit estimate vector $\hat{\mathbf{u}} \in \mathbb{F}_2^N$ for the data vector $\mathbf{u}$. In SC decoding algorithm bits are decoded sequentially, one at a time (in natural index order if bit-reversion is applied), with each bit decision depending on prior bit decisions. A high level definition for SC is given

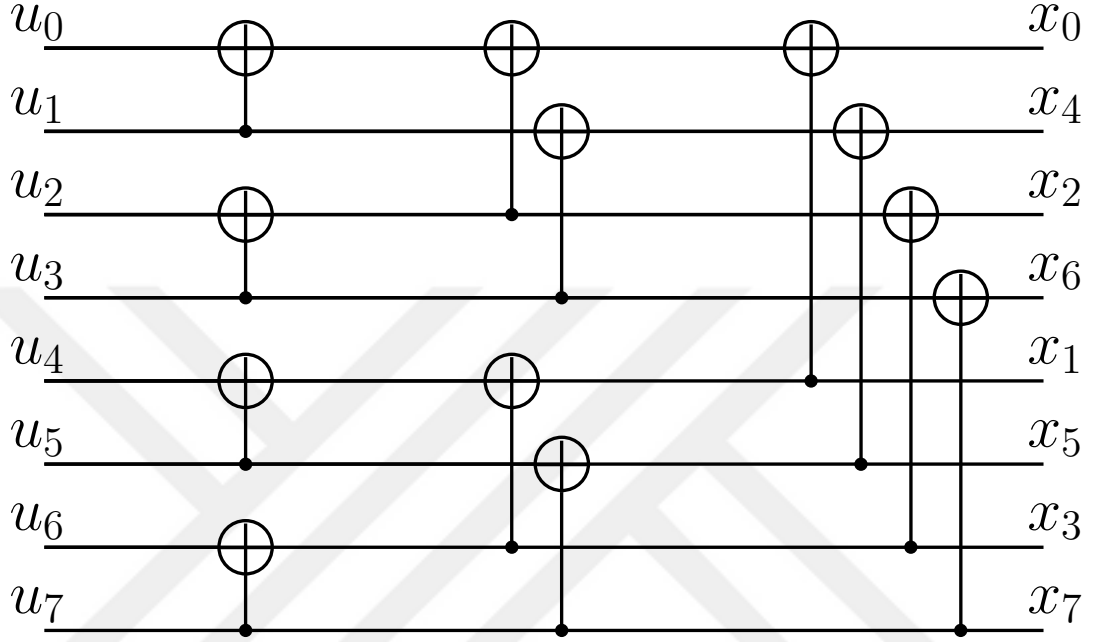


Figure 2.3: Polar encoding graph for $N = 8$

Algorithm 1: $\mathbf{x} = \text{ENCODE}(\mathbf{u})$

```

 $N = \text{length}(\mathbf{u})$ 
if  $N == 2$  then
     $x_0 \leftarrow u_0 \oplus u_1$ 
     $x_1 \leftarrow u_1$ 
    return  $\mathbf{x} \leftarrow (x_0, x_1)$ 
else
     $\mathbf{u}' \leftarrow \mathbf{u}_{\mathcal{E}_N} \oplus \mathbf{u}_{\mathcal{O}_N}$ 
     $\mathbf{x}' \leftarrow \text{ENCODE}(\mathbf{u}')$ 
     $\mathbf{u}'' \leftarrow \mathbf{u}_{\mathcal{O}_N}$ 
     $\mathbf{x}'' \leftarrow \text{ENCODE}(\mathbf{u}'')$ 
    return  $\mathbf{x} \leftarrow (\mathbf{x}', \mathbf{x}'')$ 
end

```

in Algorithm 2. The metric, $\ln \left(\frac{W_N^{(i)}(\mathbf{y}, \mathbf{u}_0^{i-1} | u_i=0)}{W_N^{(i)}(\mathbf{y}, \mathbf{u}_0^{i-1} | u_i=1)} \right)$, in Algorithm 2 is the decision LLR for $\hat{u}_i$.

The decision LLRs for each bit are calculated through $\log N$ decoding stages starting with the channel observation LLRs ℓ_i . At each new decoding stage, the LLRs from previous decoding stages are updated using one of the functions

$$f(\ell_1, \ell_2) = 2 \tanh^{-1} (\tanh(\ell_1/2) \tanh(\ell_2/2)) \quad (2.19)$$

and

$$g(\ell_1, \ell_2, v) = \ell_1(-1)^v + \ell_2. \quad (2.20)$$

The function f in (2.19) requires only two LLRs from the previous decoding stage as inputs, whereas the function g in (2.20) requires an additional input $v \in \{0, 1\}$. This third input is calculated by addition of specific combinations of previously estimated bits and named as a *partial-sum*. A total of N calculations are required at each decoding stage, which are completed at different cycles of the algorithm schedule. As explained in [6], the decoding process can be completed in $2N - 2$ cycles in a fully parallel implementation, yielding a decoding latency of $\mathcal{O}(N)$.

We demonstrate the SC decoding process with an example. Consider a polar code with block length 8. Fig. 2.5 illustrates the decoding steps for the first 4 bits of such code. The decoding graph in Fig. 2.5 consists of 3 decoding stages. The channel observation LLRs, ℓ_i , are provided to the graph from the right-hand side and the decoder outputs the bit decisions $\hat{u}_i$ from the left-hand side, for $0 \leq i \leq 7$. The nodes in the graph show the required functions to calculate the *intermediate* LLR values at each decoding stage. In Fig. 2.5, the nodes and lines that are active in calculations for each bit are highlighted by red. The highlighted nodes at the same decoding stages can be conducted in parallel. The calculations at consecutive stages are processed sequentially in different decoding cycles.

The decoding starts with the calculations for $\hat{u}_0$, which are depicted in Fig. 2.5a. Decoding of $\hat{u}_0$ is completed using only f functions at each decoding stage in 3 decoding cycles. Note that the number of parallel calculations decrease with each advance in decoding stages. The decoding of $\hat{u}_1$ starts after

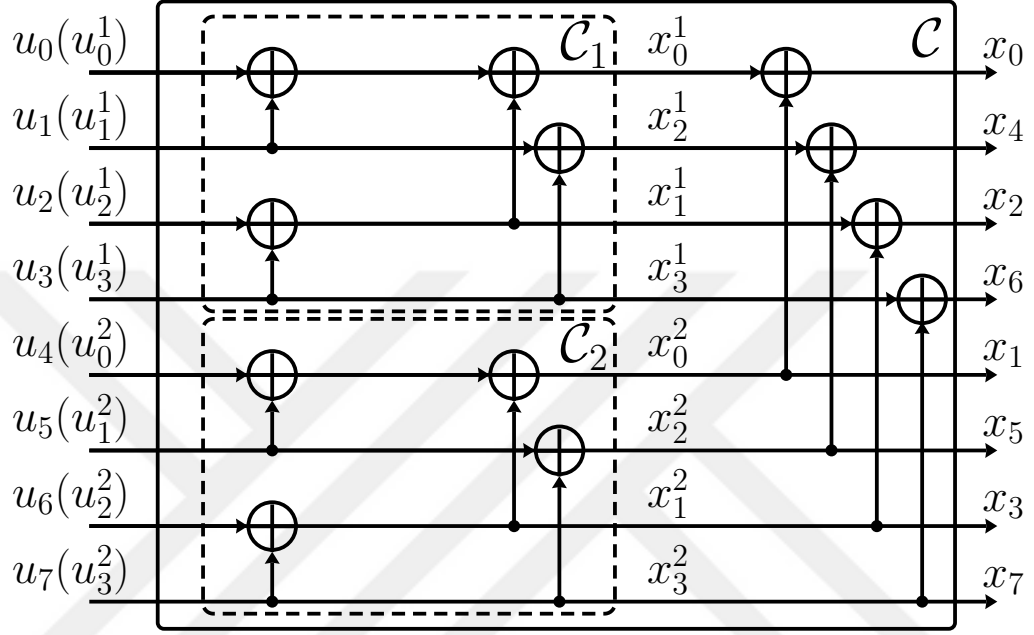


Figure 2.4: Encoding circuit of $\mathcal{C}$ with component codes $\mathcal{C}_1$ and $\mathcal{C}_2$ ($N = 8$ and $N' = 4$)

Algorithm 2: $\hat{\mathbf{u}} = \text{SC}(\mathbf{y}, \mathcal{A}, \mathbf{u}_{\mathcal{A}^c})$

```

 $N = \text{length}(\mathbf{y})$ 
for  $i = 0$  to  $N - 1$  do
    if  $i \notin \mathcal{A}$  then
         $\hat{u}_i \leftarrow u_i$ 
    else
        if  $\ln \left( \frac{W_N^{(i)}(\mathbf{y}, \mathbf{u}_0^{i-1} | u_i=0)}{W_N^{(i)}(\mathbf{y}, \mathbf{u}_0^{i-1} | u_i=1)} \right) \geq 0$  then
             $\hat{u}_i \leftarrow 0$ 
        else
             $\hat{u}_i \leftarrow 1$ 
        end
    end
end
return  $\hat{\mathbf{u}}$ 

```

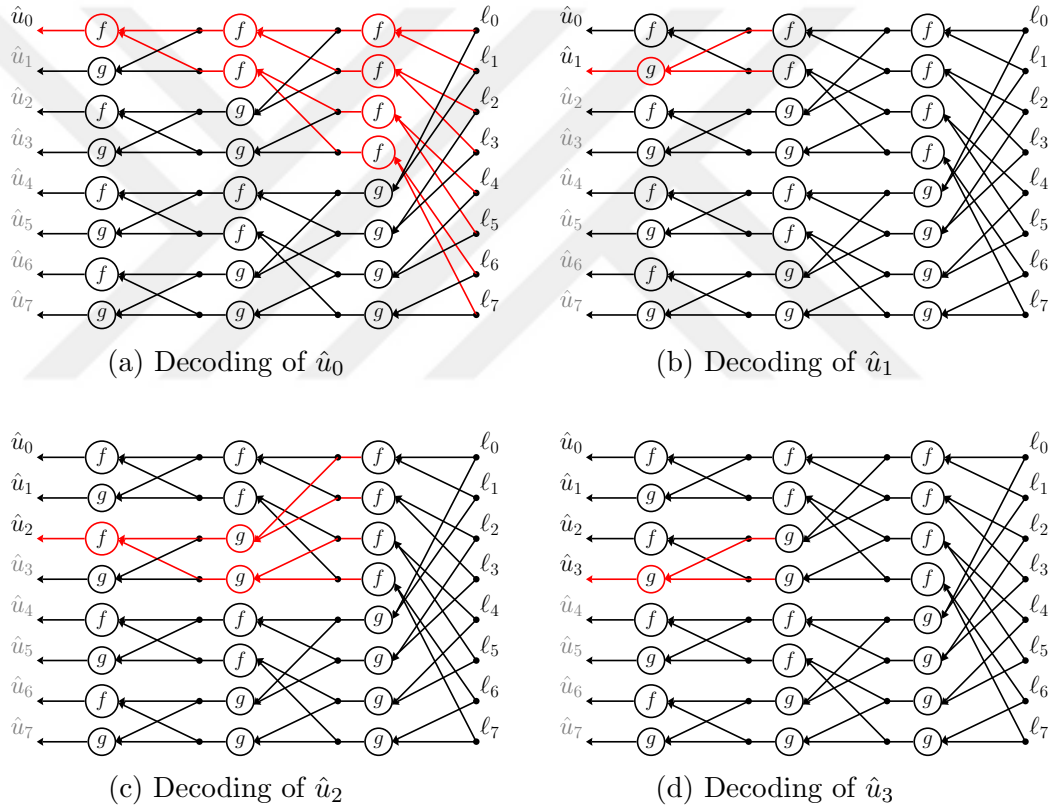


Figure 2.5: SC algorithm decoding steps for $\hat{u}_0$, $\hat{u}_1$, $\hat{u}_2$ and $\hat{u}_3$. The red nodes and LLRs carried on the red lines are used for decoding the specified bit.

the value of $\hat{u}_0$ is decided. One can see from Fig. 2.5b that the decision LLR of $\hat{u}_1$ is calculated by the g function node which uses the same LLRs with the f function node that calculates the decision LLR of $\hat{u}_0$. Recall that g function requires a third binary input called a partial-sum, which in this case is the value of $\hat{u}_0$.

In order to decode $\hat{u}_2$ and $\hat{u}_3$, the decoder moves one stage back and activates two g function nodes using the values $\hat{u}_0 \oplus \hat{u}_1$ and $\hat{u}_1$ as partial-sums. An additional f function is required to decide for $\hat{u}_2$. The value for $\hat{u}_3$ is calculated in a similar manner to that of $\hat{u}_1$; by means of a g function and $\hat{u}_2$ for partial-sum. The SC decoding process is completed after all bits are decoded.

The SC decoder schedule is explained in more detail in [6]. In this thesis, we consider the recursive description of the SC algorithm, where a decoding instance of block length N is broken into two decoding instances of lengths $N/2$ each. Algorithm 3 gives such description with the functions $f_{N/2}$ and $g_{N/2}$ defined as

$$\begin{aligned} f_{N/2}(\boldsymbol{\ell}) &= (f(\ell_0, \ell_1), \dots, f(\ell_{N-2}, \ell_{N-1})) \\ g_{N/2}(\boldsymbol{\ell}, \mathbf{v}) &= (g(\ell_0, \ell_1, v_0), \dots, g(\ell_{N-2}, \ell_{N-1}, v_{N/2-1})) . \end{aligned}$$

In actual implementations discussed in this thesis, the function f is approximated using the *min-sum* formula

$$f(\ell_1, \ell_2) \approx (1 - 2s(\ell_1)) \cdot (1 - 2s(\ell_2)) \cdot \min \{|\ell_1|, |\ell_2|\} . \quad (2.21)$$

and g is realized in the exact form

$$g(\ell_1, \ell_2, v) = \ell_2 + (1 - 2v) \cdot \ell_1 . \quad (2.22)$$

There are a total of $N \log N$ calculations in SC algorithm. Thus, the algorithmic complexity order of SC decoding is $\mathcal{O}(N \log N)$.

2.3 Summary of the Chapter

In this chapter, we summarized the basics of polar coding. We explained the polarization concept and polar encoding process. Then, we gave the code construction methods and the details of SC decoding algorithm.

In the next chapter, we briefly give background information on the decoding algorithms for polar codes other than SC algorithm and compare their state-of-the-art implementations, which aid us to validate the motivations for the studies in the thesis.

Algorithm 3: $\hat{\mathbf{u}} = \text{DECODE}(\ell, \mathbf{a})$

```
 $N = \text{length}(\ell)$ 
if  $N == 2$  then
     $\hat{u}_0 \leftarrow s(f(\ell_0, \ell_1)) \cdot a_0$ 
     $\hat{u}_1 \leftarrow s(g(\ell_0, \ell_1, \hat{u}_0)) \cdot a_1$ 
    return  $\hat{\mathbf{u}} \leftarrow (\hat{u}_0, \hat{u}_1)$ 
else
     $\ell' \leftarrow f_{N/2}(\ell)$ 
     $\mathbf{a}' \leftarrow (a_0, \dots, a_{N/2-1})$ 
     $\hat{\mathbf{u}}' \leftarrow \text{DECODE}(\ell', \mathbf{a}')$ 
     $\mathbf{v} \leftarrow \text{ENCODE}(\hat{\mathbf{u}}')$ 
     $\ell'' \leftarrow g_{N/2}(\ell, \mathbf{v})$ 
     $\mathbf{a}'' \leftarrow (a_{N/2}, \dots, a_{N-1})$ 
     $\hat{\mathbf{u}}'' \leftarrow \text{DECODE}(\ell'', \mathbf{a}'')$ 
    return  $\hat{\mathbf{u}} \leftarrow (\hat{\mathbf{u}}', \hat{\mathbf{u}}'')$ 
end
```

Chapter 3

Decoding Algorithms and Decoder Implementations for Polar Codes

In this chapter, we summarize SCL and BP decoding algorithms for polar codes and present the state-of-the art decoder implementations for SC, SCL and BP algorithms. We also explain the conventional majority-logic decoding algorithm.

3.1 Decoding Algorithms for Polar Codes

SC algorithm is used in [6] as a low-complexity decoding algorithm for polar codes. Since then, several architectures and their implementation results for SC decoders have been reported [40]-[45]. The drawbacks of the SC algorithm have been identified as its error performance in AWGN channels and the throughput bottleneck (which will be explained in more detail later in this chapter). In an effort to overcome the performance and throughput problems, SCL [46] and BP [47] algorithms have been proposed, respectively. We note that sphere [48], SC flip [49], SC stack [50] and soft cancellation (SCAN) [51] algorithms were also

proposed to decode polar codes. These algorithms are not covered in the thesis as implementation studies are mainly focused on SCL and BP. We also explain majority-logic decoding, since the algorithm will be investigated and implemented in scope of polar codes in Chapter 5.

3.1.1 Successive-Cancellation List Decoding

While being simple, SC decoding algorithm is suboptimal. In [46], SCL decoding was proposed for decoding polar codes, following similar ideas developed earlier by [52] for RM codes. SCL decoders improve the error performance with respect to SC decoders with a penalty in complexity.

A high level description of SCL algorithm is given in Algorithm 4. SCL decoders are based on SC algorithm. However, unlike SC decoders, SCL decoders keep L alternative decoded bit sequences during the decoding process in order to enhance the error performance. Ordinary SC decoding is a special case of SCL decoding with list size $L = 1$.

As observed in Algorithm 4, SCL decoders avoid making direct decisions for each $\hat{u}_i, i \in A$. Instead, a SCL decoder splits into two alternative decision paths at such stages for the bit values of 0 and 1. The aim of the procedure is to reduce the probability of eliminating the correct bit sequence path during decoding process. In order to avoid the exponential growth of the number of alternative paths with the number of decoded bits, SCL decoders choose the L most likely paths among the alternative paths as soon as the number of the alternatives reach $2L$. The path elimination process is performed over the decision probabilities of each path k , $W_N^{(i)}(\mathbf{y}, \mathbf{u}_0^{i-1}[k]|u)$, for $u \in \{0, 1\}$. The decoder completes the decoding process with a list of L most likely paths $\hat{\mathbf{u}}[k], k \in \{1, \dots, L\}$ and outputs the most likely path in the list.

The error performances of SC and SCL decoders with different list sizes are given in Fig. 3.1. It is seen from the figure that SCL decoder achieves an improvement in error performance with respect to SC decoder. A more significant

Algorithm 4: $\hat{\mathbf{u}} = \text{SCL}(\mathbf{y}, \mathcal{A}, \mathbf{u}_{\mathcal{A}^c}, L)$

```

 $N = \text{length}(\mathbf{y})$ 
 $\gamma = 1$  // current list size
for  $i = 0$  to  $N - 1$  do
    if  $i \notin \mathcal{A}$  then
        for  $k = 1$  to  $\gamma$  do
             $\hat{u}_i[k] \leftarrow u_i$ 
        end
    else
        if  $\gamma < L$  then
            for  $k = 1$  to  $\gamma$  do
                 $\hat{u}_0^{i-1}[\{k, k + \gamma\}] \leftarrow \hat{u}_1^{i-1}[k]$ 
                 $\hat{u}_i[k] \leftarrow 0$ 
                 $\hat{u}_i[k + \gamma] \leftarrow 1$ 
            end
             $\gamma \leftarrow 2\gamma$ 
        else
            // sort the 2L paths according to the decision
            // probabilities in descending order
             $\Gamma \leftarrow \text{SORT}\left(\left(\mathbf{u}_0^{i-1}[k], u\right), W_N^{(i)}\left(\mathbf{y}, \mathbf{u}_0^{i-1}[k]|u\right)\right), \forall k \in \{1, \dots, L\},$ 
             $\forall u \in \{0, 1\}$ 
            for  $k = 1$  to  $L$  do
                // the first L paths in  $\Gamma$  survive
                 $\hat{u}_0^i[k] \leftarrow \Gamma_k$ 
            end
        end
    end
end
end
 $k' \leftarrow \text{argmax}_{k \in \{1, \dots, \gamma\}} W_N^{(N-1)}\left(\mathbf{y}, \mathbf{u}_0^{N-1}[k]|u_N[k]\right)$ 
return  $\hat{\mathbf{u}}[k']$ 

```

gain is obtained when polar codes are concatenated with cyclic redundancy check (CRC) codes, as proposed in [46]. It was reported in [46] that in most of the cases a SCL decoder fails, the correct bit sequence is found among the L most likely paths at the decoder output. Employing CRC helps to choose the correct path in such cases, the effect of which is observed in Fig. 3.1.

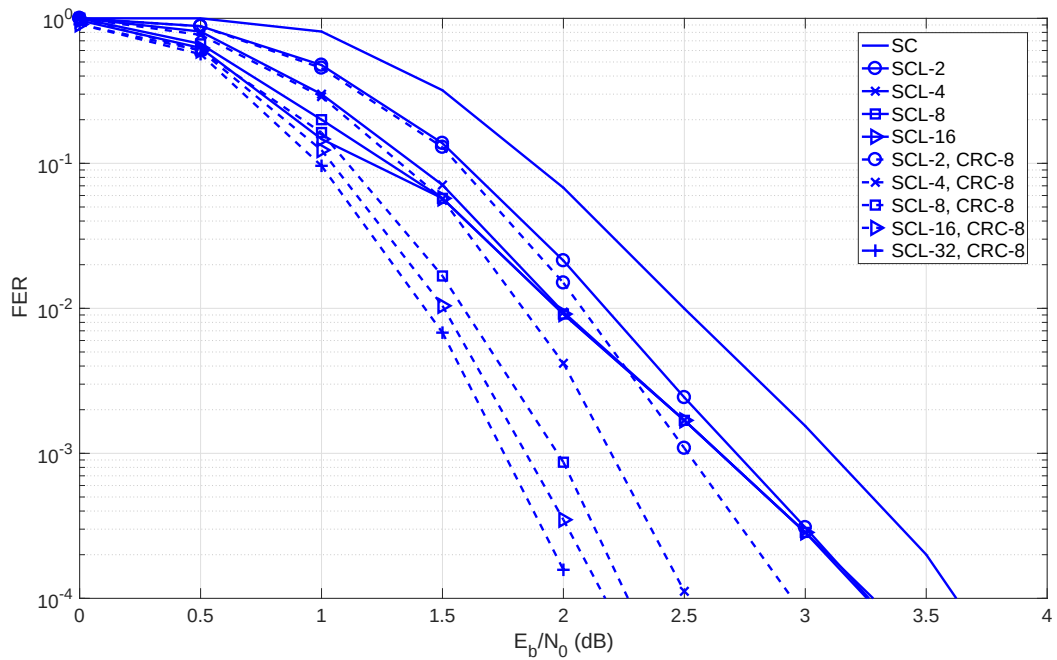


Figure 3.1: SCL performance

SCL decoders show markedly better error performance compared to SC decoders at the expense of complexity. It was shown in [52] and [46] that the conventional SCL algorithm has the overall algorithmic complexity $\mathcal{O}(LN \log N)$. It will be demonstrated in Section 3 that SCL decoders are not suitable for applications with very high throughput or very low power consumption requirements due to high hardware complexity.

3.1.2 Belief Propagation Decoding

BP decoding for polar codes, first mentioned in [47], was proposed to improve the decoder throughput by the inherent parallelism of the message-passing algorithm. Different from SC decoding, where bits are decoded in serial fashion, BP decoding can output all bit decisions in parallel. This property of BP algorithm improves the decoder throughput in a fully-parallel decoder with an increased algorithmic complexity.

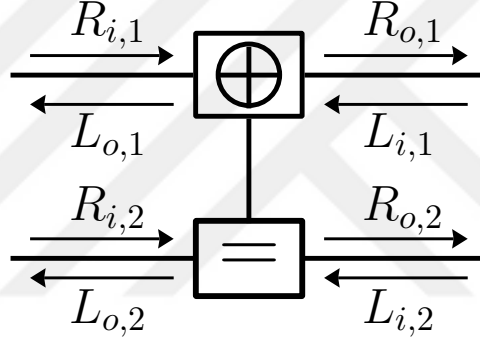


Figure 3.2: Processing element for BP decoding

The basic processing element and the factor graph for BP decoding of polar codes are given in Figures 3.2 and 3.3, respectively. In BP polar decoding, soft messages are passed between the processing elements in the factor graph in an iterative fashion. The particular soft messages with min-sum approximation are defined as

$$\begin{aligned} L_{o,1} &= f(L_{i,1}, L_{i,2} + R_{i,2}), & L_{o,2} &= f(R_{i,1}, L_{i,1}) + L_{i,2} \\ R_{o,1} &= f(R_{i,1}, L_{i,2} + R_{i,2}), & R_{o,2} &= f(R_{i,1}, L_{i,1}) + R_{i,2}. \end{aligned} \quad (3.1)$$

The soft message calculations are of similar complexity to those in SC decoding, as seen in (3.1). A decoder iteration is defined as one activation of all nodes in the factor graph. The algorithmic complexity of BP polar decoding $\mathcal{O}(IN \log N)$, where I is the number of decoding iterations.

The decoding schedule determines the activation sequence of the nodes in a single iteration. The error performance of the BP decoding depends on the

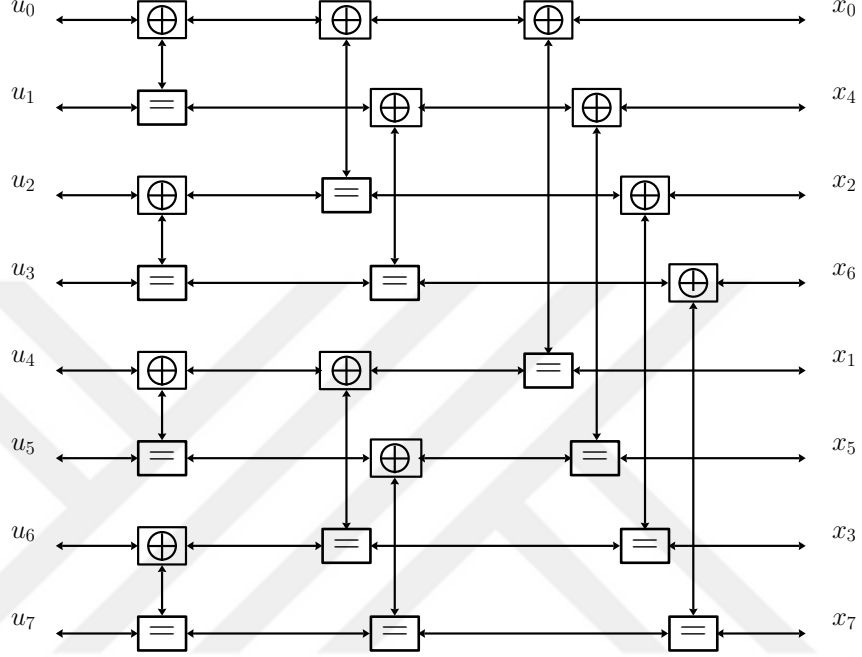


Figure 3.3: Factor graph for BP decoding of polar codes

number of decoding iterations and the scheduling [53]. For polar codes, the error performance of BP is similar to that of SC decoding, which is due to the short-length loops of the polar code factor graph [53]. However, it has not been proved that polar codes achieve channel capacity with BP decoding.

We investigate the implementation performances of BP polar decoders in Chapter 3, along with those of SC and SCL decoders.

3.1.3 Majority-Logic Decoding

Majority-logic algorithm is based on Reed's decoding algorithm [54], which was proposed for an ECC introduced by Muller in [55] (the mentioned ECC is named as Reed–Muller (RM) codes). Majority-logic is a low-latency decoding method owing to its ability to decode multiple bits in parallel. An HD algorithm in its

original definition, weighted and SD versions of majority-logic have been proposed in [31] and [56], respectively.

Majority-logic uses a number of *check-sums* for each information bit for the decoding process. A check-sum is a sum over multiple codeword bits, the result of which is the value of the information bit in consideration. We start explaining the concept of check-sums and the majority-logic algorithm with an example over RM codes. For this purpose, we briefly introduce some concepts on RM codes in this section.

The generator matrix of an RM code of block length $N = 2^m$ can be formed from 2^m -tuples in $\mathbb{F}_2$ of the form

$$\mathbf{v}_l = (\underbrace{0 \dots 0}_{2^{l-1}}, \underbrace{1 \dots 1}_{2^{l-1}}, \dots, \underbrace{1 \dots 1}_{2^{l-1}})$$

for $1 \leq l \leq m$ and their element-wise multiplications in $\mathbb{F}_2$ such that

$$\mathbf{a} \cdot \mathbf{b} = (a_0 b_0, a_1 b_1, \dots, a_{n-1} b_{n-1}).$$

The vectors that are products of any k number of 2^m -tuples $\mathbf{v}_l$, for $1 \leq k \leq m$, are shown as

$$\mathbf{v}_{i_1} \mathbf{v}_{i_2} \dots \mathbf{v}_{i_k},$$

where $1 \leq i_1 \leq i_2 \leq \dots \leq i_k \leq m$. Such vectors are said to be *degree- k* vectors.

For any integers r and m , $0 \leq r \leq m$, there exists an $\text{RM}(r, m)$ code with code length $N = 2^m$ and information block length $K = 1 + \binom{m}{1} + \binom{m}{2} + \dots + \binom{m}{r}$. As an example, consider the $\text{RM}(1, 3)$ code. Such code has a block length $N = 2^3 = 8$ and $K = 1 + \binom{3}{1} = 4$. We can express the encoding operation for such code in

vector-matrix multiplication form as

$$\mathbf{u} \begin{bmatrix} \mathbf{v}_0 \\ \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \\ \mathbf{v}_1\mathbf{v}_2 \\ \mathbf{v}_1\mathbf{v}_3 \\ \mathbf{v}_2\mathbf{v}_3 \\ \mathbf{v}_1\mathbf{v}_2\mathbf{v}_3 \end{bmatrix} = \mathbf{x},$$

so that

$$\begin{bmatrix} u_{(0)} & u_{(1)} & u_{(2)} & u_{(3)} & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix}^T \quad (3.2)$$

We use the indexing method $u_{(i_1, \dots, i_k)}$, $1 \leq i_1 \leq i_2 \leq \dots \leq i_k \leq m$, to represent the information bit multiplying $\mathbf{v}_{i_1} \mathbf{v}_{i_2} \dots \mathbf{v}_{i_k}$. From the expression in (3.2), one can notice that certain information bits can directly be recovered by summing specific combinations of the codeword bits x_i . Consider the information bit $u_{(1)}$. As observed from the 2nd row of the generator matrix, which is v_1 , the information bit $u_{(1)}$ is carried in four codeword bits, x_1, x_3, x_5 and x_7 , along with other information bits. In order to obtain the value of $u_{(1)}$ from the mentioned codeword bits, we can form four separate sums over x_1, x_3, x_5 and x_7 and disjoint sets of other codeword bits. For block length 8, such sums are easily determined from the generator matrix, as stated in [54]. The sums are given in (3.3).

$$u_{(1)} = x_0 \oplus x_1 = x_2 \oplus x_3 = x_4 \oplus x_5 = x_6 \oplus x_7. \quad (3.3)$$

We obtain four independent reconstructions of $u_{(1)}$ from the sums in (3.3). The sums for $u_{(2)}$ and $u_{(3)}$ can also be written in a similar fashion.

$$\begin{aligned} u_{(2)} &= x_0 \oplus x_2 = x_1 \oplus x_3 = x_4 \oplus x_6 = x_5 \oplus x_7, \\ u_{(3)} &= x_0 \oplus x_4 = x_1 \oplus x_5 = x_2 \oplus x_6 = x_3 \oplus x_7. \end{aligned} \quad (3.4)$$

Assume that the codeword is transmitted through a binary-input binary-output channel and the received codeword is $\mathbf{y}$. We write the sums for $u_{(1)}$ using the received codeword as shown in (3.5).

$$\begin{aligned} \gamma_1 &= y_0 \oplus y_1, \\ \gamma_2 &= y_2 \oplus y_3, \\ \gamma_3 &= y_4 \oplus y_5, \\ \gamma_4 &= y_6 \oplus y_7. \end{aligned} \quad (3.5)$$

The sums are named *check-sums*. If there are no errors in $\mathbf{y}$, then all check-sums return the same value which is equal to the value of $u_{(1)}$. If there is 1 y_i with error, then three of the four check-sums return the same value and it is assigned to the estimate for $u_{(1)}$. We can formulate the decision-making process as

$$\hat{u}_{(1)} = \begin{cases} 1, & \text{if } \sum_{i=1}^4 (2\gamma_i - 1) > 0 \\ 0, & \text{otherwise,} \end{cases} \quad (3.6)$$

The rule in (3.6) is the *majority-logic decision* rule. The same rule is applied to obtain $\hat{u}_{(2)}$ and $\hat{u}_{(3)}$ using the sums in (3.4).

After the bits $u_{(1)}$, $u_{(2)}$ and $u_{(3)}$ are decoded, we say that stage-0 of the decoding process is complete. In order to continue the decoding process with the remaining bits, the effects of the decoded bits are removed from the received codeword. The modified codeword is denoted by $\mathbf{y}^{(1)}$ as it will be used in decoding stage-1.

$$\mathbf{y}^{(1)} = \mathbf{y} - \sum_{i=1}^m \hat{u}_i \mathbf{v}_i.$$

The decoding process continues with the estimation of $u_{(0)}$. Since the modified received codeword $\mathbf{y}^{(1)}$ does not carry any other information bit, the rule to

decode $u_{(0)}$ can directly be written as

$$\hat{u}_{(0)} = \begin{cases} 1, & \text{if } \sum_{l=1}^8 (2y_l^{(1)} - 1) > 0 \\ 0, & \text{otherwise,} \end{cases} \quad (3.7)$$

The example above gives an insight about the basics of majority-logic decoding algorithm. A more general description is essential for the applicability of the algorithm to RM codes with different block lengths and code rates. We take [57, p.110] as reference to explain the generalized description of majority-logic decoding for RM codes. Consider the majority-logic decoding of an $\text{RM}(r, m)$ code. There are $r + 1$ stages in the decoding process of such code. Suppose that we are at the decoding stage- k , $0 \leq k \leq r$. The bits to be decoded at stage- k are $u_{(i_1 i_2 \dots i_{r-k})}$, $1 \leq i_1 < \dots < i_{r-k} \leq m$. We use the modified received vector $\mathbf{y}^{(k)}$ in the check-sums, which is obtained as

$$\mathbf{y}^{(k)} = \mathbf{y}^{(k-1)} - \sum_{1 \leq i_1 < \dots < i_{r-k+1} \leq m} \hat{u}_{(i_1 i_2 \dots i_{r-k+1})} v_{i_1 i_2 \dots i_{r-k+1}}.$$

Note that $\mathbf{y}^{(0)} = \mathbf{y}$.

Let us define the index set $\mathcal{S}$ for any information bit $u_{(i_1 i_2 \dots i_{r-k})}$, such that

$$\mathcal{S} = \{a_{i_1-1}2^{i_1-1} + \dots + a_{i_{r-k}-1}2^{i_{r-k}-1} : a_{i_l-1} \in \{0, 1\}, 1 \leq l \leq r-k\}.$$

The set $\mathcal{S}$ contains 2^{r-k} non-negative integers which are less than 2^m . Let the set $\mathcal{E}$ be defined as

$$\begin{aligned} \mathcal{E} &= \{0, 1, \dots, m-1\} \setminus \{i_1-1, i_2-1, \dots, i_{r-k}-1\} \\ &= \{j_1, j_2, \dots, j_{m-r+k}\}, \end{aligned} \quad (3.8)$$

with $0 \leq j_1 < j_2 < \dots < j_{m-r+k} \leq m-1$. Using the elements of $\mathcal{E}$, we form a second index set $\mathcal{S}^c$, such that

$$\mathcal{S}^c = \{b_{j_1}2^{j_1} + \dots + b_{j_{m-r+k}}2^{j_{m-r+k}} : b_{j_l} \in \{0, 1\}, 1 \leq l \leq m-r+k\}.$$

The set $\mathcal{S}^c$ contains 2^{m-r+k} non-negative integers which are less than 2^m . We use the integers in the sets $\mathcal{S}$ and $\mathcal{S}^c$ to obtain the indexes of bits in $\mathbf{y}^{(k)}$ to be used

in the check-sums for $u_{(i_1 i_2 \dots i_{r-k})}$. For each integer $q_i \in \mathcal{S}^c$, $1 \leq i \leq 2^{m-r+k}$, we form the set $\mathcal{C}_i$, such that

$$\mathcal{C}_i = \{q_i + s : s \in \mathcal{S}\}.$$

Each set $\mathcal{C}_i$ contains 2^{r-k} integers. The particular integers are used as bit indexes in a check-sum for the considered information bit. We write such check-sum $\gamma_i^{(k)}$ by

$$\gamma_i^{(k)} = \sum_{j \in \mathcal{C}_i} y_j^{(k)}, \quad 1 \leq i \leq 2^{m-r+k}.$$

As a result, we obtain 2^{m-r+k} check-sums for an information bit at decoding stage- k , each check-sum being over sets of 2^{r-k} bits. This procedure is repeated for each information bit. We give an example to demonstrate the procedure. Consider the decoding $u_{(1)}$ again. We have the sets

$$\begin{aligned} \mathcal{S} &= \{a_0 2^0 : a_0 \in \{0, 1\}\}, \\ &= \{0, 1\}, \\ \mathcal{E} &= \{0, 1, 2\} \setminus \{0\}, \\ &= \{1, 2\}, \\ \mathcal{S}^c &= \{b_1 2^1 + b_2 2^2 : b_1, b_2 \in \{0, 1\}\}, \\ &= \{0, 2, 4, 6\}, \end{aligned} \tag{3.9}$$

for $u_{(1)}$. Using the provided sets, we form the check-sum index sets $\mathcal{C}_i$ as

$$\begin{aligned} \mathcal{C}_1 &= \{0 + s : s \in \mathcal{S}\} = \{0, 1\}, \\ \mathcal{C}_2 &= \{2 + s : s \in \mathcal{S}\} = \{2, 3\}, \\ \mathcal{C}_3 &= \{4 + s : s \in \mathcal{S}\} = \{4, 5\}, \\ \mathcal{C}_4 &= \{6 + s : s \in \mathcal{S}\} = \{6, 7\}. \end{aligned} \tag{3.10}$$

The check-sums formed using the index sets in (3.10) are the same as the ones given in (3.5).

The algorithm defined above is a hard decision (HD) decoding algorithm that operates with two-level-quantized channel output observations and calculations.

Weighted [31] and SD [56] versions of majority-logic decoding that operate with real-valued channel observations and calculations have been proposed to enhance the error performance. The weighted majority-logic algorithm uses the received bit reliabilities to assign weights to each check-sum before using them to make bit decisions. In AWGN, the decision making procedure of weighted-majority logic decoding for an information bit u_i is given in (3.11).

$$\hat{u}_i = \begin{cases} 1, & \text{if } \sum_{j=1}^L (2\gamma_j - 1) |y_j|_{\min} > 0 \\ 0, & \text{otherwise} \end{cases} \quad (3.11)$$

where L is the number of check-sums for $\hat{u}_i$ and $|y_j|_{\min}$ is the minimum of the absolute values of the received codeword symbols used in the check-sum γ_j . Note that the use of absolute values of received codeword symbols in the check-sums correspond to the use of LLRs in AWGN channel. This implies that each check-sum is weighted by the reliability of the least reliable received codeword symbol in the set of symbols it is defined over.

The SD majority-logic algorithm directly calculates soft values for check-sums using *a posteriori* probabilities. The algorithm estimates the value of any information bit $\hat{u}_i$ by

$$\hat{u}_i = \begin{cases} 0, & \text{if } \sum_{j=1}^L \prod_{k \in C_j} \tanh(\ell_k) \geq 0 \\ 1, & \text{otherwise.} \end{cases}$$

In Chapter 5 of this thesis, we give a recursive definition for the weighted majority-logic algorithm described above. This definition allows us to implement and investigate the characteristics of flexible weighted majority-logic decoder architectures that can support any code rate for any given block length. We investigate the algorithmic complexity of the presented recursive definition in the mentioned chapter.

3.2 State-of-the-Art Polar Decoders

As mentioned in the previous sections, one of the drawbacks of SC decoding is its limited throughput. In fully-parallel SC decoder implementations, many of

the SC decoding steps can be carried out in parallel and the latency of the SC decoder can be reduced to roughly $2N$, as pointed out in [6] and [58]. This means that the throughput of any synchronous SC decoder is limited to $\frac{f_c}{2}$ in terms of the clock frequency f_c [59]. The throughput is reduced further in semi-parallel architectures, such as [40] and [41], which increase the decoding latency further in exchange for reduced hardware complexity. The throughput bottleneck in SC decoding is inherent in the logic of SC decoding and stems from the fact that the decoder makes its final decisions one at a time in a sequential manner. Some algorithmic and hardware implementation methods have been proposed to overcome the problem.

Implementation methods such as precomputations, pipelined, and unrolled designs, have been proposed to improve the throughput of SC decoders. These methods trade hardware complexity for gains in throughput. For example, it has been shown that the decoding latency may be reduced to N by doubling the number of adders in a SC decoder circuit [60]. A similar approach has been used in the first ASIC implementation of a SC decoder to reduce the latency at the decision-level LLR calculations by $N/2$ clock cycles and provide a throughput of 49 Mb/s with 150 MHz clock frequency for a rate-1/2 code [40]. In contrast, pipelined and unrolled designs do not affect the latency of the decoder; the increase in throughput is obtained by decoding multiple codewords simultaneously without hardware resource sharing. Pipelining in the context of polar decoders is used in various forms and in a limited manner in [58], [59], [61], [60], and [62]. A recent study on pipelined SC decoders [63] exhibits a fast-simplified SC (SSC, which will be detailed later in this section) decoder achieving 25.6 Gb/s throughput with a highly-pipelined architecture using 65 nm technology. In this section, we consider decoders without high-levels of pipelining in order to have a better understanding on the advantages and disadvantages of different decoding algorithms and decoder architectures.

An algorithmic approach to break the throughput bottleneck is to exploit the fact that polar codes are a class of GCC. In order to improve the throughput of a polar code, one may introduce specific measures to speed up the decoding of the constituent polar codes encountered in the course of such recursive decomposition.

For example, when a constituent code $\mathcal{C}_i$ of rate 0 or 1 is encountered, the decoding becomes a trivial operation and can be completed in one clock cycle. Similarly, decoding is trivial when the constituent code is a repetition code or a single parity-check code. Such techniques have been applied earlier in the context of RM codes by [64] and [65]. They have been also used in speeding up SC decoders for polar codes by [66]. Indeed, results of decoder implementations using such technique, named simplified SC (SSC), show increased throughput values [67], [44], [45]. On the other hand, decoders utilizing such shortcuts require reconfiguration when the code is changed, which may alter their implementation characteristics and makes their use difficult in systems using adaptive coding methods.

Table 3.1: State-of-the-Art SC Polar Decoders on ASIC

	[40]	[41]	[42]	[43]	[44]	[45]
Block Length	1024	1024	1024	1024	1024	1024
Code Rate	1/2	Any	Any	1/2	1/2	1/2
Arch.	SC / Semi-Parallel	SC / Semi-Parallel	SC / Semi-Parallel	SC / Tree-Based	SSC / Semi-Parallel	SSC / Tree-Based
Quant. Bits	5	5	5	5	(6,5,1)	(4,5,0)
PEs	64	64	64	1023	-	1023
Tech.[nm]	180	65	65	45	65	45
Voltage [V]	1.3	1.2	-	-	1.0	-
Area [mm ²]	1.71	0.68	0.30	-	0.69	0.28
Freq. [MHz]	150	1010	500	750	600	1040
Power [mW]	67.0	-	-	-	215	-
TP [Mb/s]	49 ^{††}	497	246	500	1860 ^{††}	2010
Engy.-per-bit [pJ/b]	1370	-	-	-	115	-
Hard. Eff. [Gb/s/mm ²]	0.03 [†]	0.7 [†]	0.8 [†]	-	2.7	7.2 [†]

[†] Not presented in the paper, calculated from the presented results

^{††} Information bit throughput

Table 3.1 summarizes the implementation performances of state-of-the-art SC polar decoders. The semi-parallel and tree-based architectures follow the SC

decoding scheduling explained in [6] with a given number of processing elements (PEs) and a control logic. The mentioned PEs are circuit blocks capable of calculating both f and g functions. The semi-parallel architecture is based on the idea of limiting the number of maximum parallel calculations in a single decoding clock cycle by the number of PEs employed in the decoder. The PEs are controlled by a control logic and used in accordance with the SC algorithm scheduling. Thus, the decoder latency is changed in a reversely proportional manner with the number of employed PEs. The hardware complexity also depends on the number of PEs. In the tree-based architectures, $N - 1$ PEs are employed to conduct calculations at different decoding stages. This specific number of PEs is enough to perform the maximum number of parallel calculations at each decoding stage by reserved PEs for that stage, so that the decoder latency is not increased.

Drawbacks of the state-of-the-art synchronous decoders are as follows: at decoding stages with number of parallel calculations less than the number of PEs, hardware-utilization is reduced in semi-parallel and tree-based architectures. A reduction in decoder latency is not possible by the number of PEs employed in the decoder at such stages, as explained at the beginning of this section. Furthermore, the intermediate LLRs calculated during the decoding process need to be stored for further calculations. The storage requirement in synchronous architectures increase the decoding time and power consumption due to the read/write operations at each clock cycle.

Another algorithmic method to overcome the throughput bottleneck is BP decoding, starting with [47]. In BP decoding, the decoder has the capability of making multiple bit decisions in parallel. Indeed, BP polar decoder throughputs of 2 Gb/s (with clock frequency 500 MHz) and 4.6 Gb/s (with clock frequency 300 MHz) are reported in [68] and [69], respectively. Implementation performances for state-of-the-art BP decoders for polar codes are given in Table 3.2. Generally speaking, the throughput advantage of BP decoding is observed at high SNR values, where correct decoding can be achieved after a small number of iterations. This advantage of BP decoders over SC decoders diminishes with decreasing SNR, as the throughputs of SC decoders are independent of the SNR values at the inputs of the decoders.

Table 3.2: State-of-the-Art BP Polar Decoders on ASIC

	[68]	[69]*		[70]
Block Length	1024	1024		1024
Code Rate	1/2	Any		1/2
Average Iterations	-	6.57		6.34
Quantization Bits	7	5		5
Technology[nm]	45	65		65
Area [mm ²]	-	1.476		1.60
Voltage [V]	-	1.0	0.475	-
Freq. [MHz]	500	300	50	334
Power [mW]	-	477.5	18.6	-
TP [Mb/s]	2000	4676	779.3	10700
Engy.-per-bit [pJ/b]	-	102.1	23.8	-
Hard. Eff. [Gb/s/mm ²]	-	3.1	0.5	6.68

* Results are given for (1024, 512) code at 4dB SNR with 6.57 iterations

One can make rough conclusions comparing the results in Tables 3.1 and 3.2 even though the implementation technologies differ. BP decoders achieve higher throughput than SC decoders for low number of decoder iterations in general. The area consumption of BP decoders are higher than those of SC decoders. The hardware efficiencies are greater than 0.5 Gb/s/mm² for both SC and BP decoders (except [40], which is implemented in 180 nm technology) and vary significantly among the same type of decoders. Decoder flexibility is not considered in any of the reported BP implementations, whereas, the SC decoders in [41] and [42] can decode codes with any code rate.

Table 3.3 gives the implementation results for state-of-the-art SCL decoders with varying list sizes. For a significant improvement in error performance, the list size of SCL decoders should be increased. In consequence, the hardware complexity of the SCL decoders increase and the achievable throughput values decrease. Such changes can be observed from the presented results in Table 3.3. The operating frequencies of SCL decoders are observed to decrease with respect to SC decoders, which is a factor that reduces the achievable throughput values for these decoders. The areas spanned by the SCL decoders are clearly larger than those of SC decoders, as expected. Indeed, it is shown in [77] that the

Table 3.3: State-of-the-Art SCL Polar Decoders on ASIC

	[71]	[72]	[73]	[74]	[75]	[76]
Block Length	1024	1024	1024	1024	1024	1024
Code Rate	1/2	1/2	1/2	1/2	1/2	1/2
L	8	4	16	4	8	4
Quantization Bits	(5,6)	5	(6,8)	6	6	$3+i^{**}$
Technology[nm]	90	90	90	65	90	65
Area [mm ²]	7.22	1.89	7.46	1.18 [*]	3.58	2.14
Voltage [V]	-	-	-	-	-	-
Freq. [MHz]	289	409	641	360 [*]	637	400
Power [mW]	-	-	-	-	-	718
TP [Mb/s]	374 ^{††}	547 ^{††}	220	675	246	401
Engy.-per-bit [pJ/b]	-	-	-	-	-	1790 [†]
Hard. Eff. [Gb/s/mm ²]	0.05	0.29	0.03 [†]	0.57	0.07	0.19

^{*} Results scaled to 90 nm ^{**} Quantization bits at decoding stage- i

[†] Not presented in the paper, calculated from the presented results

^{††} Information bit throughput

throughput and hardware efficiency metrics of state-of-the-art SCL decoders fall short of the SC decoder metrics.

The power consumption characteristics of SC and SCL decoders are compared over [44] and [76], as only [40], [44] and [76] report decoder power consumptions among the SC and SCL implementations. The power consumed by the SCL decoder in [76] is 718 mW with a throughput of 401 Mb/s with $L = 4$, which is higher than the consumption of 215 mW with a throughput of 1.86 Gb/s for the SC decoder in [44]. In general, the power consumption characteristics of SCL decoders are expected to be higher than those of SC decoders owing to the increased hardware resources and storage elements in SCL decoders. At this point, one can conclude that SCL decoders are not suitable for applications with very high throughput and/or low power consumption requirements.

An important observation from the reported results is that power consumption characteristics have not been studied except for a few decoders. Power

consumption is an important metric and should be investigated, especially in high throughput applications, for which it may exceed practical levels. In this thesis, we also focus on power consumption besides the other characteristics of the decoders we propose. Furthermore, the proposed decoders are flexible, which is another characteristic that is not considered in any but a few of the implementations.

3.3 Summary of the Chapter

In this chapter, we explained several decoding algorithms for polar codes other than SC decoding, namely SCL, BP and majority-logic algorithms. SCL and BP algorithms have been studied in the scope of hardware implementations, and majority-logic is investigated for polar codes in this thesis. We presented the implementation methods and results for state-of-the-art SC, SCL and BP decoders. The reported results showed that by means of algorithmic or implementation methods, the main focuses of the state-of-the-art SC decoder implementations are maximizing the throughput and/or minimizing the hardware complexity using tree-based or semi-parallel architectures. Comparing the decoders, it was observed that SC decoder implementations achieve less throughput than BP decoder implementations, suffering from the bottleneck problem of the SC algorithm. On the other hand, the performances of BP decoders are dependent on the number of decoding iterations, which is effected by the SNR and desired error correction performance. SCL decoders, while achieving better error performance, were shown to perform worse than SC and BP decoders in terms of throughput and hardware efficiency. We claimed that their power consumption characteristics are expected to be higher than those of SC decoders, and compared one decoder of each type to verify the claim. An important observation over the reported results is that power consumption and flexibility has not been considered in the state-of-the-art polar decoder implementations, except a few studies.

The combinational SC decoder architecture proposed in Chapter 4 of this thesis takes a different approach than the state-of-the-art SC decoder architectures.

Combinational SC decoders benefit from the non-iterative and recursive structure of the SC algorithm to implement a decoder consisting of only combinational circuitry. Such decoders decode an entire codeword in one clock cycle with a period larger than those of ordinary synchronous decoders. This allows combinational decoders to operate with less power while maintaining a high throughput, as we demonstrate in the corresponding chapter.



Chapter 4

Combinational SC Decoder

In this chapter, we propose 3 different architectures for implementing the SC decoding algorithm for polar codes. We describe the architectures and give analytical estimates for the complexity, latency and throughput. We provide ASIC and FPGA implementation results to show the performance of the proposed architectures.

The first architecture we propose is a flexible SC decoder that is fully composed of combinational circuitry, namely the combinational SC decoder. The combinational SC decoders are proposed in order to break the throughput bottleneck problem of the SC algorithm discussed in Chapter 3, with low power consumption.

Pipelining can be applied to combinational decoders at any recursion depth to adjust their throughput, hardware usage, and power consumption characteristics. We investigate the performance of pipelined combinational decoders, which is the second decoder we propose in this chapter.

We do not use any of the multi-bit decision shortcuts, which were mentioned in Section 3.2, in the architectures we propose. Thus, the combinational SC decoders retain the inherent flexibility of polar coding to operate at any desired code rate between zero and one for a given block length. Retaining such flexibility

is important since one of the main motivations behind the combinational decoder is to use it as an “accelerator” module as part of a hybrid decoder that combines a synchronous SC decoder with a combinational decoder to take advantage of the best characteristics of the two types of decoders. Hybrid-logic decoder is the final architecture we propose in this section. We give the details of the architecture as well as an analytical discussion of their throughput to quantify the advantages of the hybrid decoder.

4.1 Architecture Description

The pseudocode in Algorithm 3 shows that the logic of SC decoding contains no loops, hence it can be implemented using only combinational logic. The potential benefits of a combinational implementation are high throughput and low power consumption, which we show are feasible goals. In this section, we first describe a combinational SC decoder for length $N = 4$ to explain the basic idea. Then, we describe the three architectures that we propose.

4.1.1 Base Decoder for $N = 4$

In a combinational SC decoder, the decoder outputs are expressed directly in terms of decoder inputs without any registers or memory elements in between the input and output stages. Below we give the combinational logic expressions for a decoder of size $N = 4$, for which the signal flow graph (trellis) is depicted in Fig. 4.1.

At Stage 0 we have the LLR relations

$$\begin{aligned}\ell'_0 &= f(\ell_0, \ell_1), & \ell'_1 &= f(\ell_2, \ell_3), \\ \ell''_0 &= g(\ell_0, \ell_1, \hat{u}_0 \oplus \hat{u}_1), & \ell''_1 &= g(\ell_2, \ell_3, \hat{u}_1).\end{aligned}$$

At Stage 1, the decisions are extracted as follows.

$$\begin{aligned}\hat{u}_0 &= s[f(f(\ell_0, \ell_1), f(\ell_2, \ell_3))] \cdot a_0, \\ \hat{u}_1 &= s[g(f(\ell_0, \ell_1), f(\ell_2, \ell_3), \hat{u}_0)] \cdot a_1, \\ \hat{u}_2 &= s[f(g(\ell_0, \ell_1, \hat{u}_0 \oplus \hat{u}_1), g(\ell_2, \ell_3, \hat{u}_1))] \cdot a_2, \\ \hat{u}_3 &= s[g(g(\ell_0, \ell_1, \hat{u}_0 \oplus \hat{u}_1), g(\ell_2, \ell_3, \hat{u}_1), \hat{u}_2)] \cdot a_3,\end{aligned}$$

where the decisions $\hat{u}_0$ and $\hat{u}_2$ may be simplified as

$$\begin{aligned}\hat{u}_0 &= [s(\ell_0) \oplus s(\ell_1) \oplus s(\ell_2) \oplus s(\ell_3)] \cdot a_0, \\ \hat{u}_2 &= [s(g(\ell_0, \ell_1, \hat{u}_0 \oplus \hat{u}_1)) \oplus s(g(\ell_2, \ell_3, \hat{u}_1))] \cdot a_2.\end{aligned}$$

Fig. 4.2 shows a combinational logic implementation of the above decoder using only comparators and adders. We use sign-magnitude representation, as in [42], to avoid excessive number of conversions between different representations. Channel observation LLRs and calculations throughout the decoder are represented by Q bits. The function g of (2.22) is implemented using the precomputation method suggested in [60] to reduce latency. In order to reduce latency and complexity further, we implement the decision logic for odd-indexed bits as

$$\hat{u}_{2i+1} = \begin{cases} 0 & , \quad \text{if } a_{2i+1} = 0 \\ s(\lambda_2) & , \quad \text{if } a_{2i+1} = 1 \text{ and } |\lambda_2| \geq |\lambda_1| \\ s(\lambda_1) \oplus \hat{u}_{2i}, & \text{otherwise.} \end{cases} \quad (4.1)$$

Thanks to the recursive structure of the SC decoder, the above combinational decoder of size $N = 4$ will serve as a basic building block for the larger decoders that we will discuss.

4.1.2 Combinational SC Decoder

A combinational decoder architecture for any block length N using the recursive description in Algorithm 3 (Section 2.2.3) is shown in Fig. 4.3. This architecture uses two combinational decoders of size $N/2$, with glue logic consisting of one $f_{N/2}$ block, one $g_{N/2}$ block, and one size- $N/2$ encoder block.

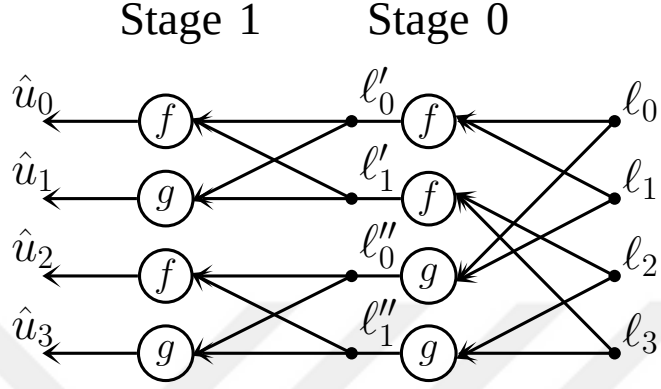


Figure 4.1: SC decoding trellis for $N = 4$

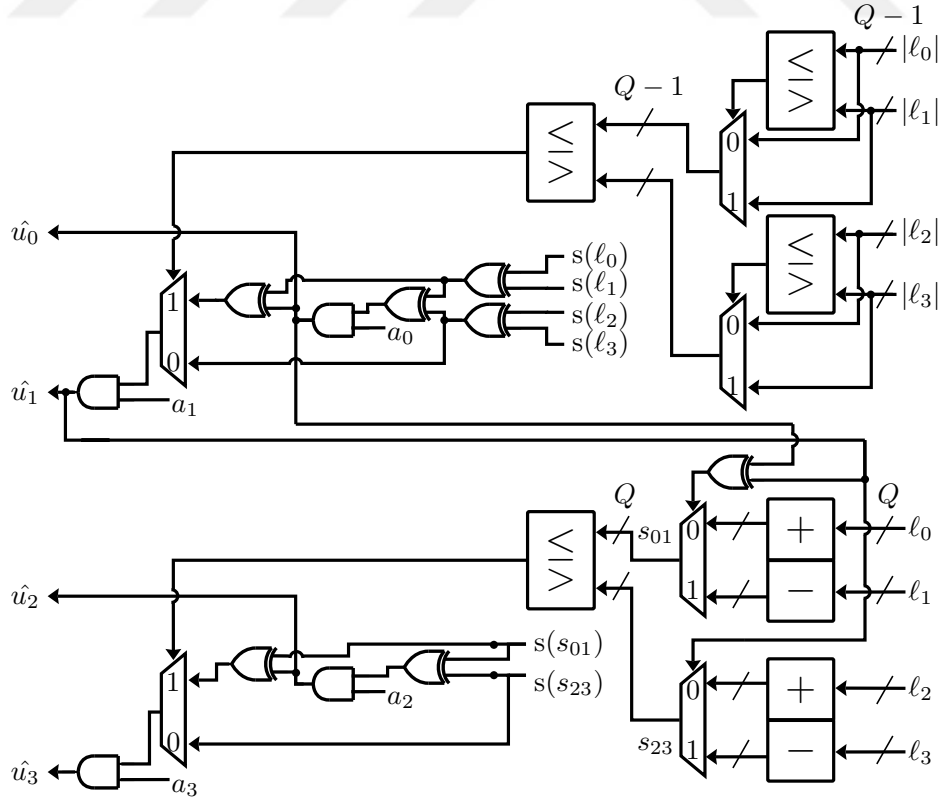


Figure 4.2: Combinational decoder for $N = 4$

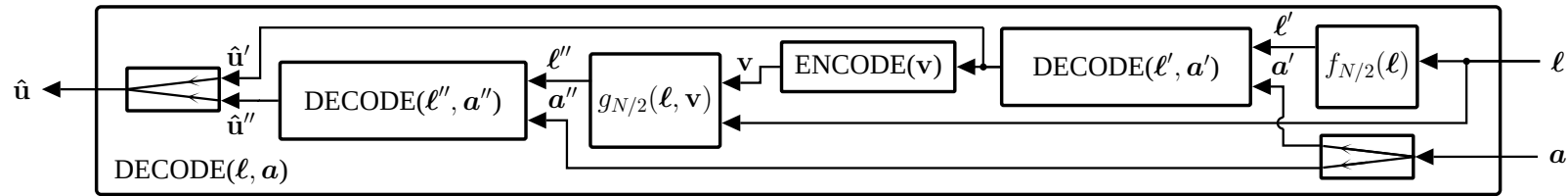


Figure 4.3: Recursive architecture of polar decoders for block length N

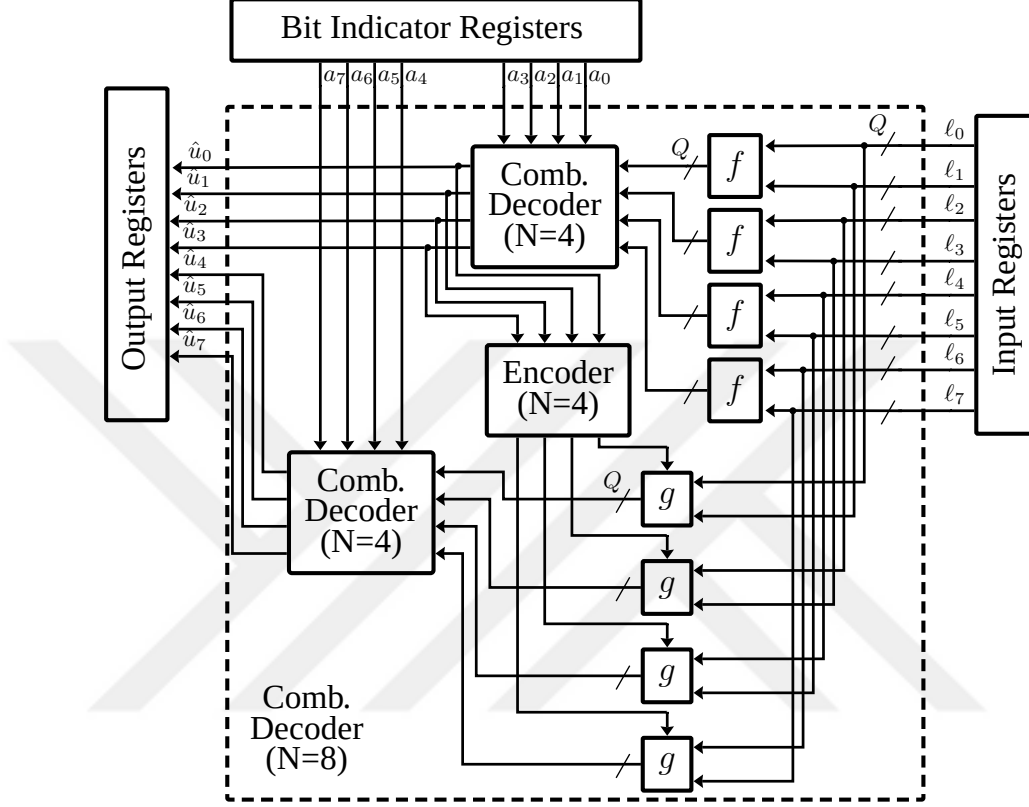


Figure 4.4: RTL schematic for combinational decoder ($N = 8$)

The RTL schematic for a combinational decoder of this type is shown in Fig. 4.4 for $N = 8$. The decoder submodules of size-4 are the same as in Fig. 4.2. The size-4 encoder is implemented using combinational circuit consisting of exclusive-or (XOR) gates. The logic blocks in a combinational decoder are directly connected without any synchronous logic elements in-between, which helps the decoder to save time and power by avoiding memory read/write operations. Avoiding the use of memory also reduces hardware complexity. In each clock period, a new channel observation LLR vector is read from the input registers and a decision vector is written to the output registers. The clock period is equal to the overall combinational delay of the circuit, which determines the throughput of the decoder. The decoder differentiates between frozen bits and data bits by AND gates and the frozen bit indicators a_i , as shown in Fig. 4.2. The frozen-bit indicator vector can be changed at the start of each decoding operation, making it possible

to change the code configuration in real time. Advantages and disadvantages of combinational decoders will be discussed in more detail in Section 4.3.

4.1.3 Pipelined Combinational SC Decoder

Unlike synchronous circuits, the combinational architecture explained above has no need for any internal storage elements. In this subsection, we introduce pipelining in order to increase the throughput at the expense of some extra hardware utilization.

It is seen in Fig. 4.3 that the outputs of the first decoder block ($\text{DECODE}(\ell', \mathbf{a}')$) are used by the encoder to calculate partial-sums. Therefore, this decoder needs to preserve its outputs after they settle to their final values. However, this particular decoder can start the decoding operation for another codeword if these partial-sums are stored with the corresponding channel observation LLRs for the second decoder ($\text{DECODE}(\ell'', \mathbf{a}'')$). Therefore, adding register blocks to certain locations in the decoder enable a pipelined decoding process.

In synchronous design with pipelining, shared resources at certain stages of decoding have to be duplicated in order to prevent conflicts on calculations when multiple codewords are processed in the decoder. The number of duplications and their stages depend on the number of codewords to be processed in parallel. Since pipelined decoders are derived from combinational decoders, they do not use resource sharing; therefore, resource duplications are not needed. Instead, pipelined combinational decoders aim to reuse the existing resources. This resource reuse is achieved by using storage elements to save the outputs of smaller combinational decoder components and re-employ them in decoding of another codeword.

A single stage pipelined combinational decoder is shown in Fig. 4.5. The channel observation LLR vectors ℓ_1 and ℓ_2 in this architecture correspond to different codewords. The partial-sum vector $\mathbf{v}_1$ is calculated from the first half of

the decoded vector for ℓ_1 . Output vectors $\hat{\mathbf{u}}'_2$ and $\hat{\mathbf{u}}''_1$ are the first and second halves of decoded vectors for ℓ_2 and ℓ_1 , respectively. The schedule for this pipelined combinational decoder is given in Table 4.1.

Table 4.1: Schedule for Single Stage Pipelined Combinational Decoder

Clock Cycle	1	2	3	4	5	6	7	8
Input of DECODE($\ell, \mathbf{a}$)	ℓ_1	ℓ_2	ℓ_3	ℓ_4	ℓ_5	ℓ_6		
Output of DECODE($\ell', \mathbf{a}'$)		$\hat{\mathbf{u}}'_1$	$\hat{\mathbf{u}}'_2$	$\hat{\mathbf{u}}'_3$	$\hat{\mathbf{u}}'_4$	$\hat{\mathbf{u}}'_5$	$\hat{\mathbf{u}}'_6$	
Output of DECODE($\ell'', \mathbf{a}''$)			$\hat{\mathbf{u}}''_1$	$\hat{\mathbf{u}}''_2$	$\hat{\mathbf{u}}''_3$	$\hat{\mathbf{u}}''_4$	$\hat{\mathbf{u}}''_5$	$\hat{\mathbf{u}}''_6$
Output of DECODE($\ell, \mathbf{a}$)			$\hat{\mathbf{u}}_1$	$\hat{\mathbf{u}}_2$	$\hat{\mathbf{u}}_3$	$\hat{\mathbf{u}}_4$	$\hat{\mathbf{u}}_5$	$\hat{\mathbf{u}}_6$

As seen from Table 4.1, pipelined combinational decoders, like combinational decoders, decode one codeword per clock cycle. However, the maximum path delay of a pipelined combinational decoder for block length N is approximately equal to the delay of a combinational decoder for block length $N/2$. Therefore, the single stage pipelined combinational decoder in Fig. 4.5 provides approximately twice the throughput of a combinational decoder for the same block length. On the other hand, power consumption and hardware usage increase due to the added storage elements and increased operating frequency. Pipelining stages can be increased by making the two combinational decoders for block length $N/2$ in Fig. 4.5 also pipelined in a similar way to increase the throughput further. Comparisons between combinational decoders and pipelined combinational decoders are given in more detail in Section 4.3.

4.1.4 Hybrid-Logic SC Decoder

In this part, we give an architecture that combines synchronous decoders with combinational decoders to carry out the decoding operations for component codes. In sequential SC decoding of polar codes, the decoder slows down every time it approaches the decision level (where decisions are made sequentially and number

of parallel calculations decrease). In a hybrid-logic SC decoder, the combinational decoder is used near the decision level to speed up the SC decoder by taking advantage of the GCC structure of polar code explained in Section 2.2. Fig. 4.6 shows the decoding trellis for the given example.

Two separate decoding sessions for block length 4 are required to decode component codes $\mathcal{C}_1$ and $\mathcal{C}_2$. We denote the input LLRs for component codes as $\boldsymbol{\lambda}^{(1)}$ and $\boldsymbol{\lambda}^{(2)}$, as shown in Fig. 4.6. These inputs are calculated by the operations at stage 0. The frozen bit indicator vector of $\mathcal{C}$ is $\mathbf{a} = (0, 0, 0, 1, 0, 1, 1, 1)$ and the frozen bit vectors of component codes are $\mathbf{a}^{(1)} = (0, 0, 0, 1)$ and $\mathbf{a}^{(2)} = (0, 1, 1, 1)$. It is seen that $\boldsymbol{\lambda}^{(2)}$ depends on the decoded outputs of $\mathcal{C}_1$, since g functions are used to calculate $\boldsymbol{\lambda}^{(2)}$ from input LLRs. This implies that the component codes cannot be decoded in parallel.

The dashed boxes in Fig. 4.6 show the operations performed by a combinational decoder for $N' = 4$. The operations outside the boxes are performed by a synchronous decoder. The sequence of decoding operations in this hybrid-logic decoder is as follows: a synchronous decoder takes channel observations LLRs and use them to calculate intermediate LLRs that require no partial-sums at stage 0. When the synchronous decoder completes its calculations at stage 0, the resulting intermediate LLRs are passed to a combinational decoder for block length 4. The combinational decoder outputs $\hat{u}_0, \dots, \hat{u}_3$ (uncoded bits of the first component code) while the synchronous decoder waits for a period equal to the maximum path delay of combinational decoder. The decoded bits are passed to the synchronous decoder to be used in partial-sums ($\hat{u}_0 \oplus \hat{u}_1 \oplus \hat{u}_2 \oplus \hat{u}_3$, $\hat{u}_1 \oplus \hat{u}_3$, $\hat{u}_2 \oplus \hat{u}_3$, and $\hat{u}_3$). The synchronous decoder calculates the intermediate LLRs using these partial-sums with channel observation LLRs and passes the calculated LLRs to the combinational decoder, where they are used for decoding of $\hat{u}_4, \dots, \hat{u}_7$ (uncoded bits of the second component code). Since the combinational decoder architecture proposed in this work can adapt to operate on any code set using the frozen bit indicator vector input, a single combinational decoder is sufficient for decoding all bits. During the decoding of a codeword, each decoder (combinational and synchronous) is activated 2 times.

Algorithm 5 shows the algorithm for hybrid-logic polar decoding for general N and N' . For the i^{th} activation of combinational and synchronous decoders, $1 \leq i \leq N/N'$, the LLR vector that is passed from synchronous to combinational decoder, the frozen bit indicator vector for the i^{th} component code, and the output bit vector are denoted by $\boldsymbol{\lambda}^{(i)} = (\lambda_0^{(i)}, \dots, \lambda_{N'-1}^{(i)})$, $\boldsymbol{a}^{(i)} = (a_{(i-1)N'}, \dots, a_{iN'-1})$, and $\hat{\mathbf{u}}^{(i)} = (\hat{u}_{(i-1)N'}, \dots, \hat{u}_{iN'-1})$, respectively. The function `DECODE_SYNC` represents the synchronous decoder that calculates the intermediate LLR values at stage $(\log \frac{N}{N'} - 1)$, using the channel observations and partial-sums at each repetition.

During the time period in which combinational decoder operates, the synchronous decoder waits for $\lceil D_{N'} \cdot f_c \rceil$ clock cycles, where f_c is the operating frequency of synchronous decoder and $D_{N'}$ is the delay of a combinational decoder for block length N' . We can calculate the approximate latency gain obtained by a hybrid-logic decoder with respect to the corresponding synchronous decoder as follows: let $L_S(N)$ denote the latency of a synchronous decoder for block length N . The latency reduction obtained using a combinational decoder for a component code of length- N' in a single repetition is $L_r(N') = L_S(N') - \lceil D_{N'} \cdot f_c \rceil$. In this formulation, it is assumed that no numerical representation conversions are needed when LLRs are passed from synchronous to combinational decoder. Furthermore, we assume that maximum path delays of combinational and synchronous decoders do not change significantly when they are implemented together. Then, the latency gain factor can be approximated as

$$g(N, N') \approx \frac{L_S(N)}{L_S(N) - (N/N') L_r(N')}. \quad (4.2)$$

The approximation is due to the additional latency from partial-sum updates at the end of each repetition using the N' decoded bits. Efficient methods for updating partial sums can be found in [41] and [78]. This latency gain multiplies the throughput of synchronous decoder, so that:

$$TP_{HL}(N, N') = g(N, N') TP_S(N),$$

where $TP_S(N, N')$ and $TP_{HL}(N)$ are the throughputs of synchronous and hybrid-logic decoders, respectively. An example of the analytical calculations for throughputs of hybrid-logic decoders is given in Section 4.3.

4.2 Complexity and Delay Analyses

In this section, we analyze the complexity and delay of combinational SC decoders. We benefit from the recursive structure of polar decoders (Algorithm 3) in deriving estimates of complexity and delay.

4.2.1 Complexity

Combinational decoder complexity can be expressed in terms of the total number of comparators, adders and subtractors in the design, as they are the basic building blocks of the architecture with similar complexities.

Proposition 3.1: The total number of comparators, adders and subtractors in the combinational SC decoder is equal to $N \left(\frac{3}{2} \log N - 1 \right)$.

Proof: First, we estimate the number of comparators. Comparators are used in two different places in the combinational decoder as explained in Section 4.1.1: in implementing the function f in (2.21), and as part of decision logic for odd-indexed bits. Let c_N denote the number of comparators used for implementing the function f for a decoder of block length N . From Algorithm 3, we see that the initial value of c_N may be taken as $c_4 = 2$. From Fig. 4.2, we observe that there is the recursive relationship

$$c_N = 2c_{N/2} + \frac{N}{2} = 2 \left(2c_{N/4} + \frac{N}{4} \right) + \frac{N}{2} = \dots$$

This recursion has the following (exact) solution

$$c_N = \frac{N}{2} \log \frac{N}{2}$$

as can be verified easily.

Let s_N denote the number of comparators used for the decision logic in a combinational decoder of block length N . We observe that $s_4 = 2$ and more

generally $s_N = 2s_{N/2}$; hence,

$$s_N = \frac{N}{2}.$$

Next, we estimate the number of adders and subtractors. The function g of (2.22) is implemented using an adder and a subtractor, as explained in Section 4.1.1. We define r_N as the total number of adders and subtractors in a combinational decoder for block length N . Observing that $r_N = 2c_N$, we obtain

$$r_N = N \log \frac{N}{2}.$$

Thus, the total number of basic logic blocks with similar complexities is given by

$$c_N + s_N + r_N = N \left(\frac{3}{2} \log N - 1 \right), \quad (4.3)$$

which completes the proof. The expression (4.3) shows that the complexity of the combinational decoder is roughly $N \log N$.

4.2.2 Combinational Delay

We approximately calculate the delay of combinational decoders using Fig. 4.3.

Proposition 3.2: The delay of a combinational SC decoder for block length $N > 4$ is approximately given by

$$D_N = N \left(\frac{3\delta_m}{2} + \delta_c + \delta_x + \frac{\delta_a}{2} \right) - [\delta_c + 2\delta_m + (\log N + 1) \delta_x] + T_N, \quad (4.4)$$

where δ_c is the delay of a comparator, δ_m is the delay of a multiplexer, δ_x is the delay of a 2-input XOR gate and T_N is the overall interconnect delay.

Proof: The combinational logic delays, excluding interconnect delays, of each component forming $\text{DECODE}(\ell, \mathbf{a})$ block is listed in Table 4.2.

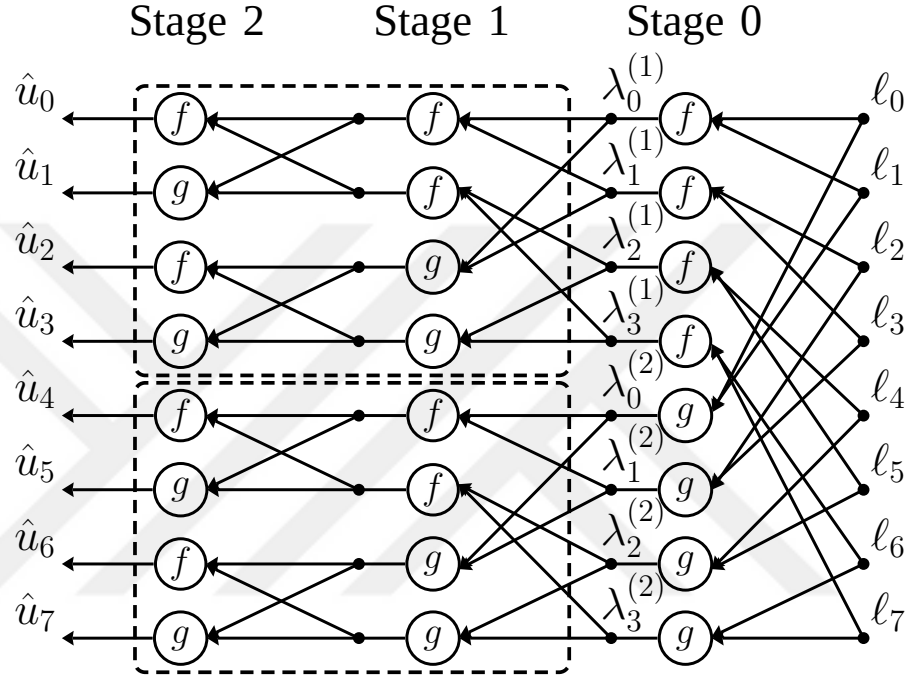


Figure 4.6: Decoding trellis for hybrid-logic decoder ($N = 8$ and $N' = 4$)

Table 4.2: Combinational Delays of Components in $\text{DECODE}(\ell, \mathbf{a})$

Block	Delay
$f_{N/2}(\ell)$	$\delta_c + \delta_m$
$\text{DECODE}(\ell', \mathbf{a}')$	$D'_{N/2}$
$\text{ENCODE}(\mathbf{v})$	$E_{N/2}$
$g_{N/2}(\ell, \mathbf{v})$	δ_m
$\text{DECODE}(\ell'', \mathbf{a}'')$	$D''_{N/2}$

Algorithm 5: HL_DECODE($\ell, \mathbf{a}, N'$)

```

for  $i = 1$  to  $N/N'$  do
  if  $i == 1$  then
     $\lambda^{(i)} \leftarrow \text{DECODE\_SYNCH}(\ell, i, N')$ 
  else
     $\lambda^{(i)} \leftarrow \text{DECODE\_SYNCH}(\ell, i, N', \hat{\mathbf{u}}^{(i-1)})$ 
  end
   $\hat{\mathbf{u}}^{(i)} \leftarrow \text{DECODE}(\lambda^{(i)}, \mathbf{a}^{(i)})$ 
end
return  $\hat{\mathbf{u}}$ 

```

The parallel comparator block $f_{N/2}(\ell)$ in Fig. 4.3 has a combinational delay of $\delta_c + \delta_m$, where δ_c is the delay of a comparator and δ_m is the delay of a multiplexer. The delay of the parallel adder and subtractor block $g_{N/2}(\ell, \mathbf{v})$ appears as δ_m due to the precomputation method, as explained in Section 4.1.1. The maximum path delay of the encoder can be approximated as $E_{N/2} \approx \lceil \log \frac{N}{2} \rceil \delta_x$, where δ_x denotes the propagational delay of a 2-input XOR gate.

We model $D'_{N/2} \approx D''_{N/2}$, although it is seen from Fig. 4.3 that $\text{DECODE}(\ell', \mathbf{a}')$ has a larger load capacitance than $\text{DECODE}(\ell'', \mathbf{a}'')$ due to the $\text{ENCODE}(\mathbf{v})$ block it drives. However, this assumption is reasonable since the circuits that are driving the encoder block at the output of $\text{DECODE}(\ell', \mathbf{a}')$ are bit-decision blocks and they compose a small portion of the overall decoder block. Therefore, we can express D_N as

$$D_N = 2D'_{N/2} + \delta_c + 2\delta_m + E_{N/2}. \quad (4.5)$$

We use the combinational decoder for $N = 4$ as the base decoder to obtain combinational decoders for larger block lengths in Section 4.1.1. Therefore, we can write D_N in terms of D'_4 and substitute the expression for D'_4 to obtain the final expression for combinational delay. Using the recursive structure of combinational decoders, we can write

$$D_N = \frac{N}{4} D'_4 + \left(\frac{N}{4} - 1 \right) (\delta_c + 2\delta_m) + \left(\frac{3N}{4} - \log N - 1 \right) \delta_x + T_N. \quad (4.6)$$

Next, we obtain an expression for D'_4 using Fig. 4.2. Assuming $\delta_c \geq 3\delta_x + \delta_a$, we

can write

$$D'_4 = 3\delta_c + 4\delta_m + \delta_x + 2\delta_a, \quad (4.7)$$

where δ_a represents the delay of an AND gate. Finally, substituting (4.7) in (4.6), we get

$$D_N = N \left(\frac{3\delta_m}{2} + \delta_c + \delta_x + \frac{\delta_a}{2} \right) - [\delta_c + 2\delta_m + (\log N + 1) \delta_x] + T_N, \quad (4.8)$$

for $N > 4$. The interconnect delay of the overall design, T_N , cannot be formulated since the routing process is not deterministic.

We had mentioned in Section 4.1.1 that the delay reduction obtained by pre-computation in adders increases linearly with N . This can be seen by observing the expressions (4.6) and (4.7). Reminding that we model the delay of an adder with precomputation by δ_m , the first and second terms of (4.6) contain the delays of adder block stages, both of which are multiplied by a factor of roughly $N/4$. This implies that the overall delay gain obtained by precomputation is approximately equal to the difference between the delay of an adder and a multiplexer, multiplied by $N/2$.

The expression (4.8) shows the relation between basic logic element delays and maximum path delay of combinational decoders. As N grows, the second term in (4.6) becomes negligible with respect to the first term, making the maximum path delay linearly proportional to $\left(\frac{3\delta_m}{2} + \delta_c + \delta_x + \frac{\delta_a}{2}\right)$ with the additive interconnect delay term T_N . Combinational architecture involves heavy routing and the interconnect delay is expected to be a non-negligible component in maximum path delay. The analytical results obtained here will be compared with implementation results in the next section.

4.3 Implementation Results

In this section, implementation results of combinational and pipelined combinational decoders are presented. Throughput and hardware complexity are studied

both in ASIC and FPGA, and a detailed discussion of the power consumption characteristics is given for the ASIC design.

We compare the combinational decoders with state-of-the-art polar and LDPC decoders in ASIC. The metrics we use in the comparisons are throughput, power, area, energy-per-bit and hardware efficiency. The number of look-up tables (LUTs) and flip-flops (FFs) in the design are studied in addition to throughput in FPGA implementations. Formulas for achievable throughputs in hybrid-logic decoders are also given in this section.

4.3.1 ASIC Synthesis Results

4.3.1.1 Post-Synthesis Results

Table 4.3 gives the post-synthesis results of combinational decoders using Cadence Encounter RTL Compiler for block lengths 2^6 - 2^{10} with Faraday's UMC 90 nm 1.3 V FSD0K-A library. Combinational decoders of such sizes can be used as standalone decoders, *e.g.*, wireless transmission of voice and data; or as parts of a hybrid-logic decoder of much larger size, as discussed in Section 4.1.4. We use $Q = 5$ bits for quantization in the implementation. As shown in Fig. 4.7, the performance loss with 5-bit quantization is negligible at $N = 1024$ (this is true also at lower block lengths, although not shown here).

The results given in Table 4.3 verify the analytical analyses for complexity and delay. It is expected from (4.3) that the ratio of decoder complexities for block lengths N and $N/2$ should be approximately 2. This can be verified by observing the number of cells and area of decoders in Table 4.3. As studied in Section 4.2.2, (4.6) implies that the maximum path delay is approximately doubled due to the basic logic elements, and there is also a non-deterministic additive delay due to the interconnects, which is also expected to at least double when block length is doubled. The maximum delay results in Table 4.3 show that this analytical derivation also holds for the given block lengths.

Table 4.3: ASIC Implementation Results

N	2^6	2^7	2^8	2^9	2^{10}
Technology	90 nm, 1.3 V				
Area [mm ²]	0.153	0.338	0.759	1.514	3.213
Number of Cells	24.3K	57.2K	127.5K	260.8K	554.3K
Dec. Power [mW]	99.8	138.8	158.7	181.4	190.7
Frequency [MHz]	45.5	22.2	11.0	5.2	2.5
Throughput [Gb/s]	2.92	2.83	2.81	2.69	2.56
Engy.-per-bit [pJ/b]	34.1	49.0	56.4	67.4	74.5
Hard. Eff. [Gb/s/mm ²]	19.1	8.3	3.7	1.8	0.8
Converted to 28 nm, 1.0 V					
Area [mm ²]	0.015	0.033	0.073	0.147	0.311
Dec. Power [mW]	18.4	25.5	29.2	33.4	35.1
Throughput [Gb/s]	9.39	9.10	9.03	8.65	8.23
Engy.-per-bit [pJ/b]	1.9	2.8	3.2	3.8	4.2
Hard. Eff. [Gb/s/mm ²]	633.8	278.0	122.9	59.0	26.4

It is seen from Table 4.3 that the removal of registers and random access memory (RAM) blocks from the design keeps the hardware usage at moderate levels despite the high number of basic logic blocks in the architecture. Moreover, the delays due to register read and write operations and clock setup/hold times are discarded, which accumulate to significant amounts as N increases.

4.3.1.2 Power Analysis

A detailed report for power characteristics of combinational decoders is given in Table 4.4.

Table 4.4: Power Consumption

N	2^6	2^7	2^8	2^9	2^{10}
Stat. [nW]	701.8	1198.7	2772.8	6131.2	14846.7
Dyn. [mW]	99.8	138.8	158.7	181.3	190.5

Table 4.4 shows the power consumption in combinational decoders in two parts: static and dynamic power [79, p.142-p.158]. Static power is due to the leakage currents in transistors when there is no voltage change in the circuit. Therefore,

it is proportional to the number of transistors and capacitance in the circuit. By observing the number of cells given in Table 4.3, we can verify the static power consumption doubling in Table 4.4 when N is doubled. On the other hand, dynamic power consumption is related with the total charging and discharging capacitance in the circuit and defined as

$$P_{\text{dynamic}} = \alpha C V_{DD}^2 f_c, \quad (4.9)$$

where α represents the average percentage of the circuit that switches with the switching voltage, C is the total load capacitance, V_{DD} is the drain voltage, and f_c is the operating frequency of the circuit ([79]). The behavior of dynamic power consumption given in Table 4.4 can be explained as follows: The total load capacitance of the circuit is approximately doubled when N is doubled, since load capacitance is proportional to the number of cells in the decoder. On the other hand, operating frequency of the circuit is approximately reduced to half when N is doubled, as discussed above. Activity factor represents the switching percentage of load capacitance, thus, it is not affected from changes in N . Therefore, the multiplication of these parameters produce approximately the same result for dynamic power consumption in decoders for different block lengths.

The decoding period of a combinational decoder is almost equally shared by the two combinational decoders for half code length. During the first half of this period, the bit estimate voltage levels at the output of the first decoder may vary until they are stabilized. These variations cause the input LLR values of the second decoder to change as they depend on the partial-sums that are calculated from the outputs of the first decoder. Therefore, the second decoder may consume undesired power during the first half of decoding period. In order to prevent this, the partial-sums are fed to the $g_{N/2}$ block through 2-input AND gates, the second input of which is given as low during the first half of delay period and high during the second half. This method can be recursively applied inside the decoders for half code lengths in order to reduce the power consumption further.

We have observed that small variations in timing constraints may lead to significant changes in power consumption. More precise figures about power

consumption will be provided in the future when an implementation of this design becomes available.

4.3.1.3 Comparison With Existing Polar Decoders

In order to have a better understanding of decoder performance, we compare the combinational decoder for $N = 1024$ with existing polar decoders in Table 4.5. We use standard conversion formulas in [29] and [30] to convert all designs to 65 nm, 1.0 V for a fair (subject to limitations in any such study) comparison.

Table 4.5: Comparison with Existing Polar Decoders

	Comb.	[40]	[41]	[69]**	
Decoder Type	SC	SC	SC	BP	
Block Length	1024	1024	1024	1024	
Code Rate	Any	1/2	Any	Any	
Technology[nm]	90	180	65	65	
Voltage [V]	1.3	1.3	1.2	1.0	0.475
Area [mm ²]	3.213	1.71	0.68	1.476	
Freq. [MHz]	2.5	150	1010	300	50
Power [mW]	190.7	67	-	477.5	18.6
TP [Mb/s]	2560	49 [†]	497	4676	779.3
Engy.-per-bit [pJ/b]	74.5	1370	-	102.1	23.8
Hard. Eff. [Gb/s/mm ²]	0.8	0.03*	0.7*	3.1	0.5
Converted to 65 nm, 1.0 V					
Area [mm ²]	1.676	0.223	0.68	1.476	
Power [mW]	81.5	14.3	-	477.5	18.6
TP [Mb/s]	3544	136	497	4676	779.3
Engy.-per-bit [pJ/b]	23.0	105.2	-	102.1	23.8
Hard. Eff. [Gb/s/mm ²]	2.1	0.6	0.7	3.1	0.5

* Not presented in the paper, calculated from the presented results

** Results are given for (1024, 512) code at 4dB SNR with 6.57 iterations

[†] Information bit throughput for (1024, 512) code

As seen from the technology-converted results in Table 4.5, combinational decoder provides the highest throughput among the state-of-the-art SC decoders. Combinational decoders are composed of simple basic logic blocks with no storage elements or control circuits. This helps to reduce the maximum path delay of

the decoder by removing delays from read/write operations, setup/hold times, complex processing elements and their management. Another factor that reduces the delay is assigning a separate logic element to each decoding operation, which allows simplifications such as the use of comparators instead of adders for odd-indexes bit decisions. Furthermore, the precomputation method reduces the delay of an addition/subtraction operation to that of a multiplexer. These elements create an advantage to the combinational decoders in terms of throughput with respect to even fully-parallel SC decoders; and therefore, [40] and [41], which are semi-parallel decoders with slightly higher latencies than fully-parallel decoders. The reduced operating frequency gives the combinational decoders a low power consumption when combined with simple basic logic blocks, and the lack of read, write, and control operations.

The use of separate logic blocks for each computation in decoding algorithm and precomputation method increase the hardware consumption of combinational decoders. This can be observed by the areas spanned by the three SC decoders. This is an expected result due to the trade-off between throughput, area, and power in digital circuits. However, the high throughput of combinational decoders make them hardware efficient architectures, as seen in Table 4.5.

Implementation results for BP decoder in [69] are given for operating characteristics at 4 dB SNR, so that the decoder requires 6.57 iterations per codeword for low error rates. The number of required iterations for BP decoders increase at lower SNR values. Therefore, throughput of the BP decoder in [69] is expected to decrease while its power consumption increases with respect to the results in Table 4.5. On the other hand, SC decoders operate with the same performance metrics at all SNR values since the total number of calculations in conventional SC decoding algorithm is constant ($N \log N$) and independent from the number of errors in the received codeword.

The performance metrics for the decoder in [69] are given for low-power-low-throughput and high-power-high-throughput modes. The power reduction in this decoder is obtained by reducing the operating frequency and supply voltage for the same architecture, which also leads to the reduction in throughput. Table 4.5

shows that the throughput of the combinational decoder is only lower than the throughput of [69] when it is operated at high-power mode. In this mode, [69] provides a throughput which is approximately 1.3 times larger than the throughput of combinational decoder, while consuming 5.8 times more power. The advantage of combinational decoders in power consumption can be seen from the energy-per-bit characteristics of decoders in Table 4.5. The combinational decoder consumes the lowest energy per decoded bit among the decoders in comparison.

4.3.1.4 Comparison With LDPC Decoders

A comparison of combinational SC polar decoders with state-of-the-art LDPC decoders is given in Table 4.6. In addition to the decoder characteristics considered so far, the table also presents the approximate SNR values that the ECC and decoder schemes require to achieve a BER of 10^{-4} for a fair comparison. It is seen from Table 4.6 that the throughputs of LDPC decoders for 5 and 10 iterations without early termination are higher than those of combinational decoders. The throughput is expected to increase for higher and decrease for lower SNR values, as explained above. Power consumption and area of the LDPC decoders are seen to be higher than those of the combinational decoder. The energy-per-bit metric of the combinational SC polar decoder is the lowest among the considered decoders.

An advantage of combinational architecture is that it provides a flexible architecture in terms of throughput, power consumption, and area by its pipelined version. One can increase the throughput of a combinational decoder by adding any number of pipelining stages. This increases the operating frequency and number of registers in the circuit, both of which increase the dynamic power consumption in the decoder core and storage parts of the circuit. The changes in throughput and power consumption with the added registers can be estimated using the characteristics of the combinational decoder. Therefore, combinational architectures present an easy way to control the trade-off between throughput, area, and power. FPGA implementation results for pipelined combinational decoders are given in the next section.

Table 4.6: Comparison with State-of-the-Art LDPC Decoders

	Comb.*		[80]	[81]	[26]
Code/Decoder Type	Polar/SC		LDPC/BP	LDPC/BP	LDPC/BP
Standard	-		IEEE 802.15.3c	IEEE 802.11ad	IEEE 802.11ad
Block Length	512	1024	672	672	672
Code Rate	Any		1/2, 5/8, 3/4, 7/8	1/2	1/2, 5/8, 3/4, 13/16
Technology [nm]	65		65	65	65
Voltage [V]	1.0		1.0	1.15	1.1
E_b/N_0 for BER= 10^{-4} w/ R=1/2 (dB)	3.5	3.1	5.1 (16QAM)	3.0	3.25
Area [mm ²]	0.79	1.676	1.56	1.60	0.575
Power [mW]	77.5	81.5	361 [†]	782.9 ^{††}	273 ^{†††}
TP [Gb/s]	3.72	3.54	5.79 [†]	9.0 ^{††}	9.25
Engy.-per-bit [pJ/b]	20.8	23.0	62.4	89.5 [*]	29.4
Hard. Eff. [Gb/s/mm ²]	4.70	2.11	3.7	5.63 [*]	16.08

* Technology converted to 65 nm, 1.0 V

[†] Results are given for (672, 588) code and 5 iterations without early termination

^{††} Results are given for (672, 336) code and 10 iterations without early termination

^{†††} Power consumption is for rate-1/2 code at SNR 2.5 dB with 7 iterations

4.3.2 FPGA Implementation Results

Combinational architecture involves heavy routing due to the large number of connected logic blocks. This increases hardware resource usage and maximum path delay in FPGA implementations, since routing is done through pre-fabricated routing resources as opposed to ASIC. In this section, we present FPGA implementations for the proposed decoders and study the effects of this phenomenon.

Tables 4.7 and 4.8 shows the place-and-route results of combinational and pipelined combinational decoders on Xilinx Virtex-6-XC6VLX550T (40 nm)

FPGA core. The implementation strategy is adjusted to increase the speed of the designs. We use RAM blocks to store the input LLRs, frozen bit indicators, and output bits in the decoders. FFs in combinational decoders are used for small logic circuits and fetching the RAM outputs, whereas in pipelined decoder they are also used to store the input LLRs and partial-sums for the second decoding function (Fig. 4.3). It is seen that the throughputs of combinational decoders in FPGA drop significantly with respect to their ASIC implementations. This is due to the high routing delays in FPGA implementations of combinational decoders, which increase up to 90% of the overall delay.

Table 4.7: Combinational SC Decoder FPGA Implementation Results

	LUT	FF	RAM (bits)	TP [Gb/s]
2^4	1479	169	112	1.05
2^5	1918	206	224	0.88
2^6	5126	392	448	0.85
2^7	14517	783	896	0.82
2^8	35152	1561	1792	0.75
2^9	77154	3090	3584	0.73
2^{10}	193456	6151	7168	0.60

Pipelined combinational decoders are able to obtain throughputs on the order of Gb/s with an increase in the number FFs used. Pipelining stages can be increased further to increase the throughput with a penalty of increasing FF usage. The results in the tables show that we can double the throughput of combinational decoder for every N by one stage of pipelining as expected.

The error rate performance of combinational decoders is given in Fig. 4.8 for different block lengths and rates. The investigated code rates are commonly used in various wireless communication standards (*e.g.*, WiMAX, IEEE 802.11n). It is seen from Fig. 4.8 that the decoders can achieve very low error rates without any error floors.

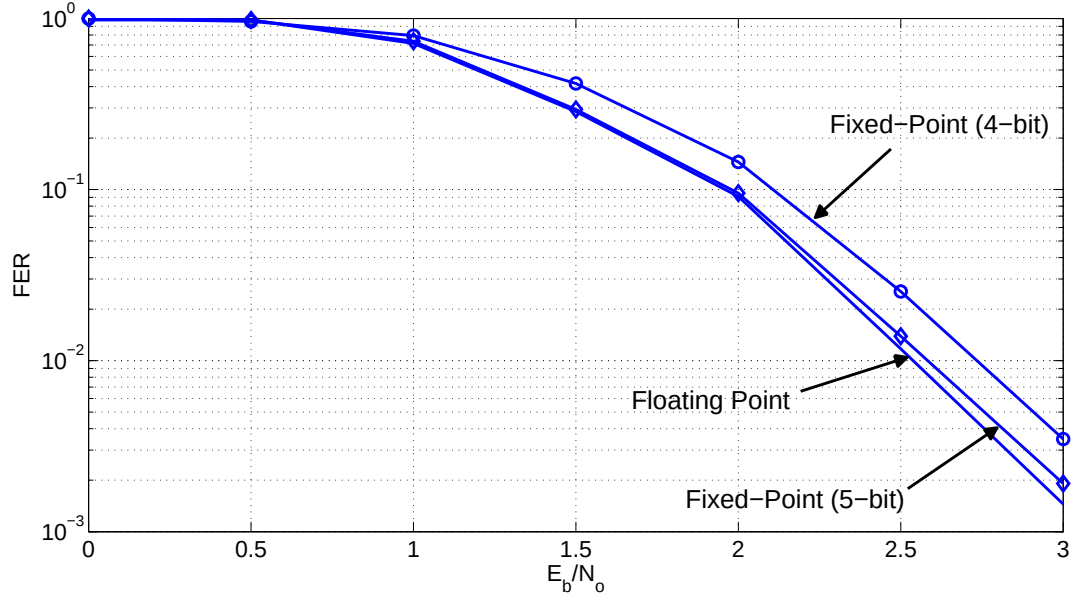


Figure 4.7: FER performance with different numbers of quantization bits ($N = 1024$, $R = 1/2$)

Table 4.8: Pipelined Combinational SC Decoder FPGA Implementation Results

	LUT	FF	RAM (bits)	TP [Gb/s]	TP Gain
2^4	777	424	208	2.34	2.23
2^5	2266	568	416	1.92	2.18
2^6	5724	1166	832	1.80	2.11
2^7	13882	2211	1664	1.62	1.97
2^8	31678	5144	3328	1.58	2.10
2^9	77948	9367	6656	1.49	2.04
2^{10}	190127	22928	13312	1.24	2.06

4.4 Throughput Analysis for Hybrid-Logic Decoders

As explained in Section 4.1.4, a combinational decoder can be combined with a synchronous decoder to increase its throughput by a factor $g(N, N')$ as in (4.2). In this section, we present analytical calculations for the throughput of a hybrid-logic decoder. We consider the semi-parallel architecture in [42] as the synchronous decoder part and use the implementation results presented before for the calculations.

A semi-parallel SC decoder employs P processing elements, each of which are capable of performing the operations (2.21) and (2.22) and perform one of them in one clock cycle. The architecture is called semi-parallel since P can be chosen smaller than the numbers of possible parallel calculations in early stages of decoding. The latency of a semi-parallel architecture is given by

$$L_{\text{SP}}(N, P) = 2N + \frac{N}{P} \log \frac{N}{4P}. \quad (4.10)$$

The minimum latency that can be obtained with the semi-parallel architecture by increasing hardware usage is $2N - 2$, the latency of a conventional SC algorithm, when $P = N/2$. Throughput of a semi-parallel architecture is its maximum operating frequency divided by its latency. Therefore, using $N/2$ processing elements does not provide a significant multiplicative gain for the throughput of the decoder.

We can approximately calculate the approximate throughput of a hybrid-logic decoder with semi-parallel architecture using the implementation results given in [42]. Implementations in [42] are done using Stratix IV FPGA, which has a similar technology with Virtex-6 FPGA used in this work. Table 4.9 gives these calculations and comparisons with the performances of semi-parallel decoder.

Table 4.9 shows that throughput of a hybrid-logic decoder is significantly better than the throughput of a semi-parallel decoder. It is also seen that the multiplicative gain increases as the size of the combinational decoder increases. This

increase is dependent on P , as P determines the decoding stage after which the number of parallel calculations become smaller than the hardware resources and causes the throughput bottleneck. It should be noted that the gain will be smaller for decoders that spend less clock cycles in final stages of decoding trellis, such as [82] and [43]. The same method can be used in ASIC to obtain a high increase in throughput.

Hybrid-logic decoders are especially useful for decoding large codewords, for which the hardware usage is high for combinational architecture and latency is high for synchronous decoders.

4.5 Summary of the Chapter

In order to solve the throughput bottleneck problem of SC decoding, we proposed a novel combinational architecture for SC polar decoders with high throughput and low power consumption. The proposed combinational SC decoders operate at much lower clock frequencies compared to state-of-the-art synchronous SC decoders and decode a codeword in one clock cycle. Due to the low operating frequency and lack of storage elements, the combinational decoder consumes less dynamic power, which reduces the overall power consumption.

Post-synthesis results show that the proposed combinational architectures are capable of providing a throughput of approximately 2.5 Gb/s with a power consumption of 190 mW using 90 nm 1.3 V technology, while preserving the inherent flexibility of polar codes and SC decoders. These figures are independent of the SNR level at the decoder input. We gave analytical formulas for the complexity and delay of the proposed combinational decoders in terms of the basic circuit component parameters and verified the implementation results.

We compared the implementation results of combinational SC decoders with those of the state-of-the-art polar and LDPC decoders. We showed that combinational SC decoders achieve the highest throughput and energy-efficiency among

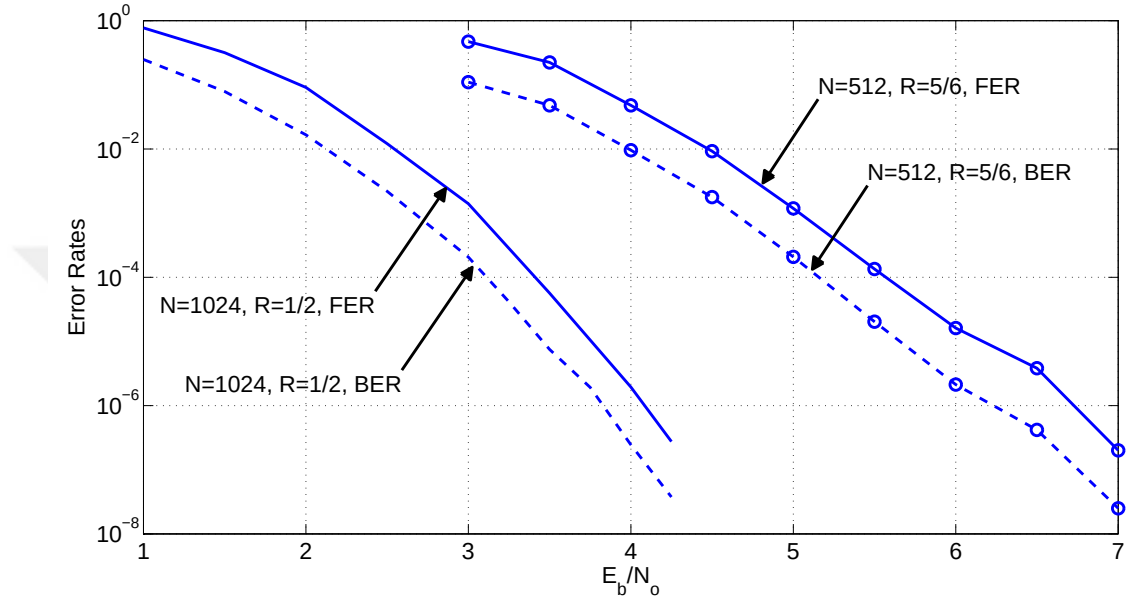


Figure 4.8: FER performance of combinational decoders for different block lengths and rates

Table 4.9: Approximate Throughput Increase for Semi-Parallel SC Decoder

N	P	f [Mhz]	TP _{SP} [Mb/s]	N'	g	TP _{HLSP} [Mb/s]
2^{10}	64	173	85	2^4	5.90	501
2^{10}	64	173	85	2^5	6.50	552
2^{10}	64	173	85	2^6	7.22	613
2^{11}	64	171	83	2^4	5.70	473
2^{11}	64	171	83	2^5	6.23	517
2^{11}	64	171	83	2^6	7.27	603

the SC polar decoders and have comparable throughput and error performance with BP polar decoders. Comparisons with LDPC decoders showed that polar codes can compete with state-of-the-art LDPC codes with combinational logic SC decoders. Thus, one can conclude that combinational SC decoders offer a fast, energy-efficient, and flexible alternative for implementing polar codes.

We also proposed two decoder architectures based on the combinational architecture. We showed that one can add pipelining stages at any desired recursion depth to the combinational architecture in order to increase its throughput at the expense of increased power consumption and hardware complexity. We also proposed a hybrid-logic SC decoder that combines the combinational SC decoder with a synchronous SC decoder so as to extend the range of applicability of the purely combinational design to larger block lengths. We performed analysis to show that hybrid-logic decoders can increase the throughputs of synchronous polar decoders by multiplicative factors.

Chapter 5

Weighted Majority-Logic Decoding of Polar Codes

As explained in Section 3, the sequential nature of SC algorithm is the main factor limiting its throughput. Although the combinational SC decoders have improved throughput with respect to the synchronous SC decoders, they are still subject to the limitations of the sequential decoding schedule of SC. In this chapter, we use weighted majority-logic algorithm described in [31] to decode polar codes for very high-throughput applications. We propose a novel recursive definition for the considered algorithm to be used in implementations of decoders for bit-reversed polar codes, instead of the conventional way of implementing check-sums for each bit separately (note that we do not propose a novelty in the algorithm itself). We analyze the complexity and latency of the proposed architecture analytically. With the proposed recursive definition, we implement the weighted majority-logic algorithm with a fully combinational circuit and give ASIC implementation results. In addition, we propose a novel hybrid decoder that employs weighted majority-logic and SC algorithms to mitigate the error performance loss of the pure majority-logic decoding. We show that such decoder has a considerably low-latency and small error performance loss with respect to SC decoding, thus being suitable for very high throughput applications.

5.1 Architecture Description

We give the definitions for weighted majority-logic and hybrid decoders for polar codes.

5.1.1 Recursive Definition for Weighted Majority-Logic Decoder

Implementing a majority-logic decoder involves determining the check-sums for each bit, which is the major part of the decoding process [57, p.109]. An example of this implementation procedure is given in [83] for two-step HD majority-logic algorithm. We take a different approach and develop a recursive definition for the weighted majority-logic algorithm for polar codes with bit-reversal operation. Using the developed recursive definition, we propose a decoder architecture that implements the check-sums inherently and removes the necessity to determine the check-sums for each information bit.

We start explaining the definition with an example. Consider the RM or polar code with block length 4, for which the generator matrix is given in (5.1). The expressions for SC decoding of such code are given in (4.1.1) and repeated here. The function s is the bit decision sign function defined in (2.2).

$$G_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad (5.1)$$

$$\begin{aligned}
\hat{u}_0 &= s[f(f(\ell_0, \ell_1), f(\ell_2, \ell_3))] \cdot a_0, \\
\hat{u}_1 &= s[g(f(\ell_0, \ell_1), f(\ell_2, \ell_3), \hat{u}_0)] \cdot a_1, \\
\hat{u}_2 &= s[f(g(\ell_0, \ell_1, \hat{u}_0 \oplus \hat{u}_1), g(\ell_2, \ell_3, \hat{u}_1))] \cdot a_2, \\
\hat{u}_3 &= s[g(g(\ell_0, \ell_1, \hat{u}_0 \oplus \hat{u}_1), g(\ell_2, \ell_3, \hat{u}_1), \hat{u}_2)] \cdot a_3.
\end{aligned} \tag{5.2}$$

The uncoded bits u_1 and u_2 are multiplied with the 2nd and 3rd rows of G_4 , both of which are degree-1 vectors. In SC decoding for polar codes, u_2 is decoded after the decision for u_1 is obtained. Thus, an asymmetry is generated between the information bit locations that are multiplied with generator matrix rows of the same degree. Owing to this asymmetry, the latency of SC algorithm is obtained as $2N - 2$ for block length N . As explained in Section 3.1.3, majority-logic algorithm exploits the symmetry in such bits by decoding them in parallel; thus reducing the decoding latency.

We give the weighted majority-logic decoding expressions for block length 4 in (5.3). The function f in (2.19) is used to obtain the weighted check-sums. The majority-logic decision rule of (3.11) is implemented by the function g in (2.20). The effects of the decoded bits are removed from the intermediate calculated LLR values instead of the channel LLRs by the g functions during the decoding process.

$$\begin{aligned}
\hat{u}_0 &= s[f(f(\ell_0, \ell_1), f(\ell_2, \ell_3))] \cdot a_0, \\
\hat{u}_1 &= s[g(f(\ell_0, \ell_1), f(\ell_2, \ell_3), \hat{u}_0)] \cdot a_1, \\
\hat{u}_2 &= s[g(f(\ell_0, \ell_2), f(\ell_1, \ell_3), \hat{u}_0)] \cdot a_2, \\
\hat{u}_3 &= s[g(g(\ell_0, \ell_1, \hat{u}_0 \oplus \hat{u}_1), g(\ell_2, \ell_3, \hat{u}_1), \hat{u}_2)] \cdot a_3.
\end{aligned} \tag{5.3}$$

Observing the expressions for $\hat{u}_1$ and $\hat{u}_2$ in (5.3), one can note the same sequence of functions with different combinations of channel LLRs as inputs. It is seen that the expression for $\hat{u}_2$ does not require $\hat{u}_1$ at any stage of calculations. Therefore, both bits can be decoded in parallel.

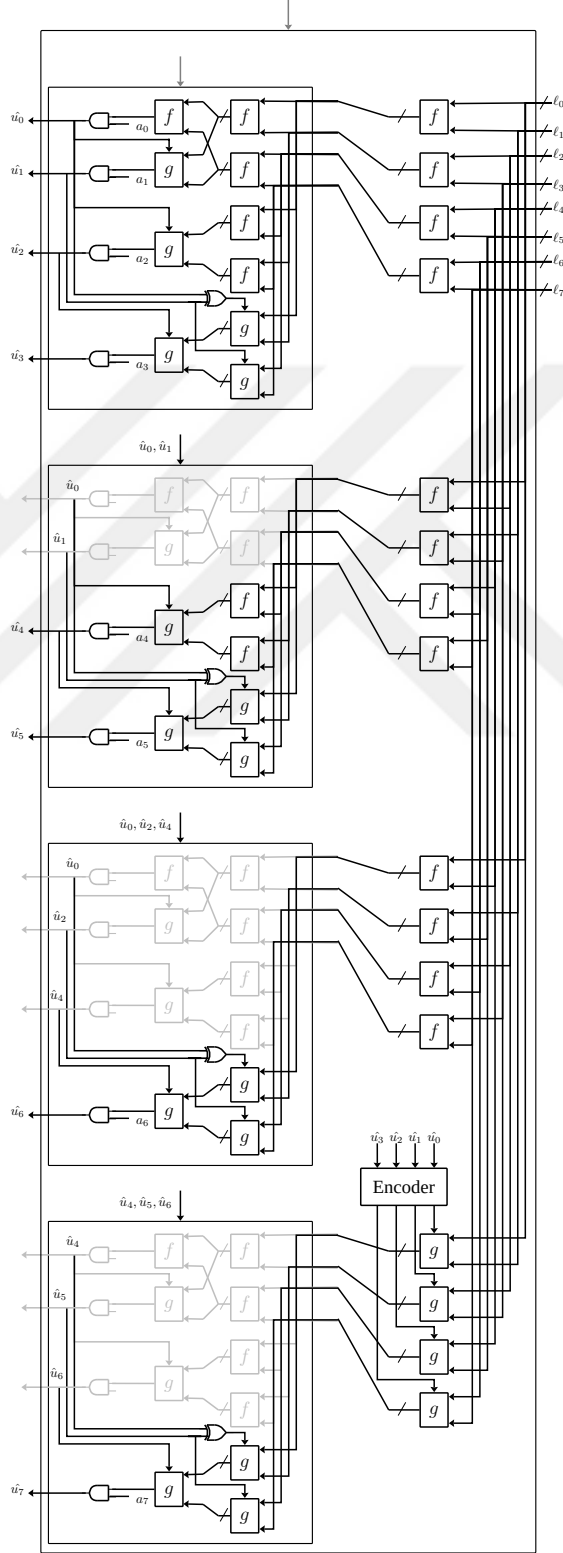


Figure 5.1: Circuit diagram for weighted majority-logic decoder for $N = 8$ using decoders for $N = 4$

Next, we consider the bit decision expression for block length 8 given in (5.4).

$$\begin{aligned}
\hat{u}_0 &= s[f\{f[f(\ell_0, \ell_1), f(\ell_2, \ell_3)], f[f(\ell_4, \ell_5), f(\ell_6, \ell_7)]\}] \cdot a_0, \\
\hat{u}_1 &= s[g\{f[f(\ell_0, \ell_1), f(\ell_2, \ell_3)], f[f(\ell_4, \ell_5), f(\ell_6, \ell_7)], \hat{u}_0\}] \cdot a_1, \\
\hat{u}_2 &= s[g\{f[f(\ell_0, \ell_1), f(\ell_4, \ell_5)], f[f(\ell_2, \ell_3), f(\ell_6, \ell_7)], \hat{u}_0\}] \cdot a_2, \\
\hat{u}_3 &= s[g\{g[f(\ell_0, \ell_1), f(\ell_2, \ell_3), \hat{u}_0 \oplus \hat{u}_1], g[f(\ell_4, \ell_5), f(\ell_6, \ell_7), \hat{u}_1], \hat{u}_2\}] \cdot a_3, \\
\hat{u}_4 &= s[g\{g[f(\ell_0, \ell_2), f(\ell_4, \ell_6)], f[f(\ell_1, \ell_3), f(\ell_5, \ell_7)], \hat{u}_0\}] \cdot a_4, \\
\hat{u}_5 &= s[g\{g[f(\ell_0, \ell_2), f(\ell_1, \ell_3), \hat{u}_0 \oplus \hat{u}_1], g[f(\ell_4, \ell_6), f(\ell_5, \ell_7), \hat{u}_1], \hat{u}_4\}] \cdot a_5, \\
\hat{u}_6 &= s[g\{g[f(\ell_0, \ell_4), f(\ell_1, \ell_5), \hat{u}_0 \oplus \hat{u}_2], g[f(\ell_2, \ell_6), f(\ell_3, \ell_7), \hat{u}_2], \hat{u}_4\}] \cdot a_6, \\
\hat{u}_7 &= s[g\{g[g(\ell_0, \ell_1, \hat{u}_0 \oplus \hat{u}_1 \oplus \hat{u}_2 \oplus \hat{u}_3), g(\ell_2, \ell_3, \hat{u}_2 \oplus \hat{u}_3), \hat{u}_4 \oplus \hat{u}_5], \\
&\quad g[g(\ell_4, \ell_5, \hat{u}_1 \oplus \hat{u}_3), g(\ell_6, \ell_7, \hat{u}_3), \hat{u}_5], \hat{u}_6\}] \cdot a_7.
\end{aligned} \tag{5.4}$$

The difference between the expressions (5.4) and the majority-logic decoding example given in Chapter 3.1.3 is the decoding scheduling. In (5.4), the effects of the decoded bits are not directly removed from the received codeword as opposed to the case in conventional majority-logic algorithm. Instead, their effects are removed from intermediate LLR calculations, as expressed above. For example, the expression for $\hat{u}_3$ in (5.4) requires the outputs of $f(\ell_{2k}, \ell_{2k+1})$, $k \in \{0, 1, 2, 3\}$, which are also calculated for the decisions $\hat{u}_0$, $\hat{u}_1$ and $\hat{u}_2$. Following the schedule of conventional majority-logic algorithm, one removes the effects of the mentioned decoded bits from the received LLRs ℓ_i and recalculates $f(\ell_{2k}, \ell_{2k+1})$. In the proposed recursive description, the architecture reuses the intermediate calculations inherently by removing the effects of previously decoded bits by the function g and specific combinations of such bits. Such operation is analogous to the check-sum reuse explained in [56]. Also, the use of specific combinations of the previously decoded bits is analogous to the use of partial-sums in SC algorithm.

The circuit diagram of a weighted majority-logic decoder for $N = 8$ obtained from decoders for $N = 4$ is given in Fig. 5.1. We use the expressions in (5.3) and (5.4) to obtain the circuitry. The uppermost weighted majority-logic decoder for $N = 4$ in the figure is a complete decoder that outputs 4 bit estimates. The rest of the weighted majority-logic decoders for $N = 4$ contain grayed-out

paths which represent the idle circuits in those decoders during the decoding process. The other calculations in such decoders are performed normally and the required partial-sums are calculated from bits estimates obtained from specific other decoders for $N = 4$, as shown in the figure.

The parallel calculations in majority-logic decoding can be observed from the circuit diagram Fig. 5.1. For example, the bits $\hat{u}_1$, $\hat{u}_2$ and $\hat{u}_4$ are calculated in parallel once $\hat{u}_0$ is obtained. Similarly, $\hat{u}_3$, $\hat{u}_5$ and $\hat{u}_6$ can be calculated at the same time using the previously estimated bits $\hat{u}_0$, $\hat{u}_1$, $\hat{u}_2$ and $\hat{u}_4$.

In order to give a general recursive description for weighted majority-logic decoding, we define the functions f_N^L and g_N^L , for $L = 2^t$ and $t \in \{0, 1, \dots, \log N - 1\}$, such that

$$\begin{aligned} f_{N/2}^L(\ell) &= \left(f \left(\ell_{0+L \lfloor \frac{0}{L} \rfloor}, \ell_{0+L(\lfloor \frac{0}{L} \rfloor + 1)} \right), \right. \\ &\quad \left. f \left(\ell_{1+L \lfloor \frac{1}{L} \rfloor}, \ell_{1+L(\lfloor \frac{1}{L} \rfloor + 1)} \right), \dots, \right. \\ &\quad \left. f \left(\ell_{N/2-1+L \lfloor \frac{N/2-1}{L} \rfloor}, \ell_{N/2-1+L(\lfloor \frac{N/2-1}{L} \rfloor + 1)} \right) \right), \\ g_{N/2}^L(\ell, \mathbf{v}) &= \left(g \left(\ell_{0+L \lfloor \frac{0}{L} \rfloor}, \ell_{0+L(\lfloor \frac{0}{L} \rfloor + 1)}, v_0 \right), \right. \\ &\quad \left. g \left(\ell_{1+L \lfloor \frac{1}{L} \rfloor}, \ell_{1+L(\lfloor \frac{1}{L} \rfloor + 1)}, v_1 \right), \dots, \right. \\ &\quad \left. g \left(\ell_{N/2-1+L \lfloor \frac{N/2-1}{L} \rfloor}, \ell_{N/2-1+L(\lfloor \frac{N/2-1}{L} \rfloor + 1)}, v_{N/2-1} \right) \right). \end{aligned} \quad (5.5)$$

As an example, the expressions for $f_4^1(\ell)$, $f_4^2(\ell)$ and $f_4^4(\ell)$ are given in (5.6). We also give the visual descriptions of the functions $f_4^1(\ell)$, $f_4^2(\ell)$ and $f_4^4(\ell)$ in Fig. 5.2.

$$\begin{aligned} f_4(\ell)^1 &= (f(\ell_0, \ell_1), f(\ell_2, \ell_3), f(\ell_4, \ell_5), f(\ell_6, \ell_7)), \\ f_4(\ell)^2 &= (f(\ell_0, \ell_2), f(\ell_1, \ell_3), f(\ell_4, \ell_6), f(\ell_5, \ell_7)), \\ f_4(\ell)^4 &= (f(\ell_0, \ell_4), f(\ell_1, \ell_5), f(\ell_2, \ell_6), f(\ell_3, \ell_7)). \end{aligned} \quad (5.6)$$

We define the binary vector representation $\hat{u}_{0:M-1}^K$, for $K > 0$, as

$$\hat{u}_{0:M-1}^K = (u_0, \dots, u_{K-1}, u_{2K}, \dots, u_{3K-1}, \dots, u_{M-2K}, \dots, u_{M-K-1}). \quad (5.7)$$

$$f_4^1(\ell) = (\overbrace{\ell_0 \ell_1}^f \overbrace{\ell_2 \ell_3}^f \overbrace{\ell_4 \ell_5}^f \overbrace{\ell_6 \ell_7}^f)$$

(a) Visualization of $f_4^1(\ell)$

$$f_4^2(\ell) = (\overbrace{\ell_0 \ell_1 \ell_2 \ell_3}^{f \ f} \overbrace{\ell_4 \ell_5 \ell_6 \ell_7}^{f \ f})$$

(b) Visualization of $f_4^2(\ell)$

$$f_4^4(\ell) = (\overbrace{\ell_0 \ell_1 \ell_2 \ell_3 \ell_4 \ell_5 \ell_6 \ell_7}^{f \ f \ f \ f})$$

(c) Visualization of $f_4^4(\ell)$

Figure 5.2: Visualizations of $f_4^1(\ell)$, $f_4^2(\ell)$ and $f_4^4(\ell)$. The connected ℓ_i are input to the f function together.

In order to demonstrate the uses of the definitions in (5.5) and (5.7), we give the block diagram for the circuitry depicted by Fig. 5.1 in Fig. 5.3. The inputs of the first 3 decoders for $N = 4$ are obtained by the function blocks for which the expressions are given in (5.6). As also demonstrated in Fig. 5.1, the 2nd, 3rd and 4th decoders for $N = 4$ do not perform the calculations for estimating their first 2 and 3 bits, respectively. The input vectors $\hat{u}_{0:3}^2 = (u_0, u_1)$, $\hat{u}_{0:5}^1 = (u_0, u_2, u_4)$, $\hat{u}_{4:6} = (u_4, u_5, u_6)$ replace the bits that are not estimated in those decoders to be used in partial-sums.

We generalize the recursive formulation of the weighted majority-logic decoding in Algorithm 6. The input $\hat{\mathbf{u}}'$ in Algorithm 6 is a binary vector of length $M = \sum_{j=0}^{k-1} N/2^{j+1}$, for $k \in \{0, 1, \dots, \log N\}$, that contains certain previously estimated bits. A decoder of the proposed definition treats the bits in such input

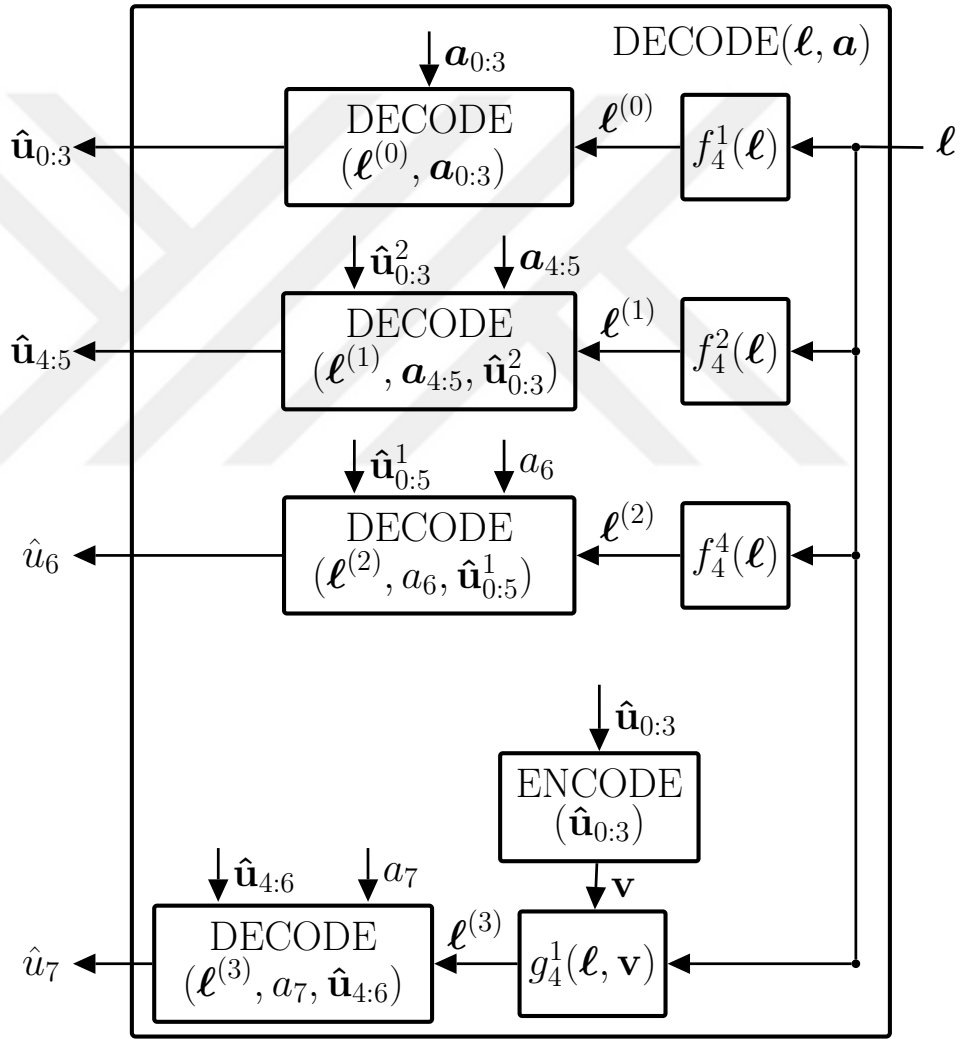


Figure 5.3: Weighted majority-logic decoder for $N = 8$ using decoders for $N = 4$

vector as if they were the first M decoded bits in that decoder and starts the decoding process from the $(M + 1)$ st bit. The output of the decoder is a bit vector of length $N - M$, obtained by using the bits in $\hat{\mathbf{u}}'$ in partial-sums when necessary. Fig. 5.4 shows the block diagram of Algorithm 6.

Algorithm 6: $\hat{\mathbf{u}} = \text{DECODE}(\ell, \mathbf{a}, \hat{\mathbf{u}}')$

```

 $N = \text{length}(\ell)$ 
 $M = \text{length}(\hat{\mathbf{u}}')$ 
 $(\gamma_0, \dots, \gamma_{M-1}) \leftarrow \hat{\mathbf{u}}'$ 
if  $N == 4$  then
     $(\gamma_M, \dots, \gamma_3) \leftarrow$  corresponding expressions in (5.3)
else
    if  $M == 0$  then
         $l \leftarrow 0$ 
    else
         $l \leftarrow \left\{ k \in \{0, 1, \dots, \log N\} \mid \sum_{j=0}^{k-1} N/2^{j+1} = M \right\}$ 
    end
    for  $i = l$  to  $\log N - 1$  do
         $k_{\text{lower}} \leftarrow \sum_{j=0}^{i-1} N/2^{j+1}$ 
         $k_{\text{upper}} \leftarrow \sum_{j=0}^i N/2^{j+1}$ 
         $\ell^{(i)} \leftarrow f_{N/2}^{2^i}(\ell)$ 
         $\mathbf{a}^{(i)} \leftarrow (a_{k_{\text{lower}}}, \dots, a_{k_{\text{upper}}-1})$ 
         $(\gamma_{k_{\text{lower}}}, \dots, \gamma_{k_{\text{upper}}-1}) \leftarrow \text{DECODE}(\ell^{(i)}, \mathbf{a}^{(i)}, \gamma_{0:k_{\text{lower}}-1}^{N/2^{(i+1)}})$ 
    end
     $\mathbf{v} \leftarrow \text{ENCODE}(\gamma_0, \dots, \gamma_{N/2-1})$ 
     $\ell^{(\log N)} \leftarrow g_{N/2}^1(\ell, \mathbf{v})$ 
     $\gamma_{N-1} \leftarrow \text{DECODE}(\ell^{(\log N)}, a_{N-1}, \gamma_{N/2:N-2})$ 
end
return  $\hat{\mathbf{u}} \leftarrow (\gamma_M, \dots, \gamma_{N-1})$ 

```

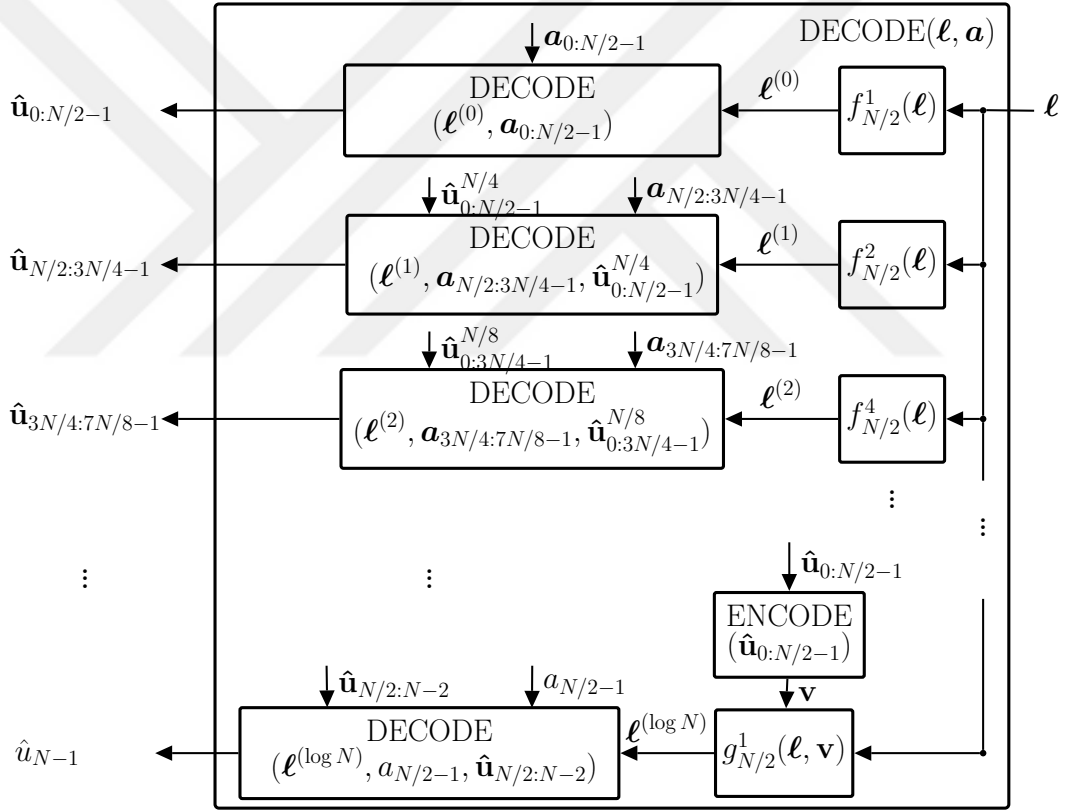


Figure 5.4: Weighted majority-logic decoder for N using decoders for $N/2$

As majority-logic decoding outputs more than one bits in parallel, polarization is not fully exploited in the decoding process. Therefore, an error performance degradation is expected to occur at the decoder output. We analyze the error performance of weighted majority-logic decoders in Section 5.4.

5.1.2 Hybrid Decoder

We propose a hybrid decoding method using SC and weighted majority-logic algorithms. The purpose of such decoder is to reduce the decoding latency while keeping the error performance loss at low levels.

The proposed decoder uses the GCC structure of polar codes, as explained in Section 2.2. The hybrid decoder follows similar principles to those of hybrid-logic decoders in Section 4.1.4. Decoding operations of component codes in a polar code are carried out by weighted majority-logic decoders to speed up the SC decoding process. Figure 5.5 shows the decoding trellis for the proposed architecture for block length 8 and component code block length 4.

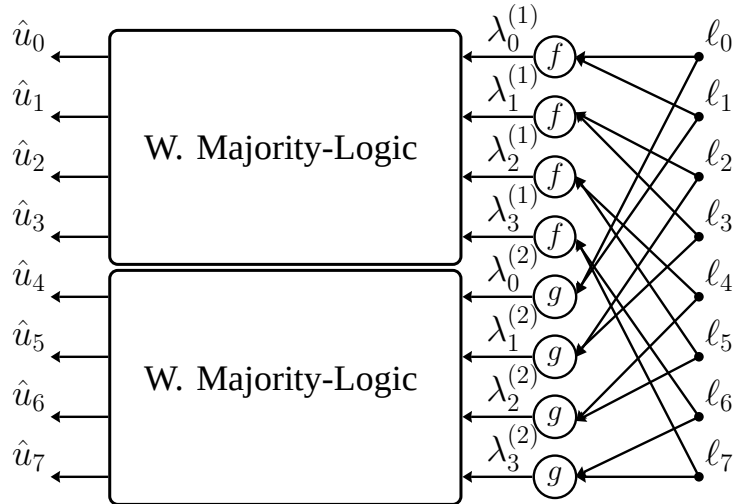


Figure 5.5: Decoding trellis for hybrid decoder ($N = 8$ and $N' = 4$)

We estimate the latency of the hybrid decoders analytically and investigate the error performance in the next section and in Section 5.4, respectively.

5.2 Complexity and Latency Analyses

In this section, we analyze the complexity and latency of the proposed weighted majority-logic and hybrid decoders. We benefit from the structure given in Algorithm 6 in the provided analyses.

5.2.1 Weighted Majority-Logic Decoder

5.2.1.1 Complexity

We perform the complexity analysis by calculating the the total number of f and g functions in the decoding process using the definition in Algorithm 6. Let C_N denote the total number of f and g functions performed in a decoder for block length N and $C_{N/2}^{(i)}$ denote the total number of f and g functions carried out in the decoding function for block length $N/2$ in turn $i \in (0, 1, \dots, \log N - 1)$ in Algorithm 6. According to the definition in Algorithm 6, the decoding function for block length $N/2$ called in turn i outputs $N/2^{i+1}$ bits, so that the calculations for the first $N/2 - N/2^{i+1}$ bits are not performed. Also, the decoding function called in turn $\log N$ outputs 1 bit.

Proposition 4.1: The total number of calculations for decoding each bit of a codeword with block length N is given by

$$C_N = 2(N^{\log 3} - N) \quad (5.8)$$

Proof: We begin the proof by calculating the number of f and g functions in decoders obtained using the definition in Algorithm 6. From Algorithm 6, we can express C_N in terms of $C_{N/2}^{(i)}$ as

$$C_N = \left(\frac{N}{2} + C_{N/2}^{(\log N - 1)} \right) + \sum_{i=0}^{\log N - 1} \left(\frac{N}{2} + C_{N/2}^{(i)} \right). \quad (5.9)$$

The terms $\frac{N}{2}$ in (5.9) are due to the functions $f_{N/2}^{2^i}$, for $0 \leq i \leq \log N - 1$, and $g_{N/2}^1$ in Algorithm 6. The term $C_{N/2}^{(\log N - 1)}$ in the first component of (5.9) represents the

required number of calculations to estimate the final bit in turn $\log N$, which is also equal to the number of operations in turn $\log N - 1$, thus the representation. We give the total number of calculations for block lengths 2^2 - 2^{10} in each turn i found using the expression (5.9) in Table 5.1.

One can notice from Table 5.1 that the number of calculations for block length N can be written in terms of the number of calculations for block length $N/2$ as

$$C_N = N + 3C_{N/2}. \quad (5.10)$$

As seen from (5.10), the complexity of the proposed architecture is approximately multiplied by 3 when the block length is doubled. Expanding the recursive expression (5.10) and using $C_4 = 10$, it is straightforward to show that

$$C_N = 2(3^{\log N} - N).$$

We obtain the expression given in (5.8) by

$$2(3^{\log N} - N) = 2(3^{\frac{\log_3 N}{\log_3 2}} - N) = 2(N^{\log 3} - N), \quad (5.11)$$

which completes the proof.

The calculated algorithmic complexity order $\mathcal{O}(N^{\log 3})$ is in correspondence with the complexity order calculated in [56] for hard and soft majority-logic decoding algorithms benefiting from the reuse of calculated check-sum values between decoding stages. Such reuse is inherent in the architecture description we present. Note that the complexity of conventional majority-logic decoding is $\mathcal{O}(N^2)$ (for the case where each bit is decoded) without employing the reuse of calculated check-sums.

5.2.1.2 Latency

The proposed architecture is described in a sequential manner in Algorithm 6. This description can be misleading when considering the parallelism of the architecture. An example is the series of functions $f_{N/2}^{2^i}(\ell)$, for $0 \leq i \leq \log N - 1$, called in each turn before the decoding functions for block length $N/2$. In a

Table 5.1: Number of Calculations for Block Lengths 2^2 - 2^{10}

i	N								
	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}
0	4	4+ 10	8+ 38	16+ 130	32+ 422	64+ 1330	128+ 4118	256+ 12610	512+ 38342
1	3	4+ 6	8+ 24	16+ 84	32+ 276	64+ 876	128+ 2774	256+ 8364	512+ 25476
2	3	4+ 3	8+ 14	16+ 52	32+ 176	64+ 568	128+ 1784	256+ 5512	512+ 16856
3		4+ 3	8+ 7	16+ 30	32+ 108	64+ 360	128+ 1152	256+ 3600	512+ 11088
4			8+ 7	16+ 15	32+ 62	64+ 220	128+ 728	256+ 2320	512+ 7232
5				16+ 15	32+ 31	64+ 126	128+ 444	256+ 1464	512+ 4656
6					32+ 31	64+ 63	128+ 254	256+ 892	512+ 2936
7						64+ 63	128+ 127	256+ 510	512+ 1788
8							128+ 127	256+ 255	512+ 1022
9								256+ 255	512+ 511
10									512+ 511
Total	10	38	130	422	1330	4118	12610	38342	116050

hardware implementation, these functions can be implemented in parallel as they do not require any previous bit estimates. Similarly, specific functions in the sequential calls of decoding functions in Algorithm 6 can be processed in parallel as explained in Section 5.1 over Fig. 5.1. In fact, the decoders called in each turn in Algorithm 6 complete their operations at the same time, except the final decoder that outputs $\hat{u}_{N-1}$. The decoding operation for $\hat{u}_{N-1}$ is completed in $\log N$ stages of addition/subtraction after the bits $(\hat{u}_0, \dots, \hat{u}_{N-2})$ are decoded.

Proposition 4.2: The latency of the proposed weighted majority-logic architecture for block length N is given by

$$L_N = \frac{\log^2 N + 3 \log N}{2} \quad (5.12)$$

Proof: We can write the latency expression for the proposed decoder using the above explanations. Using the recursive description, L_N can be written in terms of $L_{N/2}$ as

$$L_N = L_{N/2} + 1 + \log N \quad (5.13)$$

The additive 1 term in (5.13) represents the additional delay from parallel $f_{N/2}^L$ functions at the input of each decoder for block length $N/2$. The term $\log N$ is the additional delay required to calculate $\hat{u}_{N-1}$, as explained above. We expand the recursion in (5.13) and use $L_4 = 5$ to obtain

$$\begin{aligned} L_N &= (\log N)^2 - \log N + 3 - \sum_{i=1}^{\log N - 3} i \\ &= (\log N)^2 - \log N + 3 - \frac{(\log N - 3)(\log N - 2)}{2} \\ &= \frac{\log N(\log N + 3)}{2}, \end{aligned} \quad (5.14)$$

which completes the proof.

The throughput of the decoder is directly proportional with N/L_N so that

$$\text{Throughput} \propto \frac{2N}{\log^2 N + 3 \log N} \quad (5.15)$$

A important implication of (5.15) is that the decoder throughput increases with increasing with block length.

5.2.2 Hybrid Decoder

We calculate the latency of hybrid decoding assuming conventional SC decoding of latency $2N - 2$ and using (5.12).

Proposition 4.3: The latency of the hybrid decoder for block length N and component code block length N' is given by

$$L_N = \frac{N}{N'} \left(2 + \frac{\log N'(\log N' + 3)}{2} \right) - 2 \quad (5.16)$$

Proof: The proof is straightforward. The latency of a SC decoder excluding the decoding latencies of component codes of block length N' is calculated as

$$2N - 2 - \frac{N}{N'}(2N' - 2) = 2\frac{N}{N'} - 2 \quad (5.17)$$

We add the latencies of $\frac{N}{N'}$ weighted majority-logic decoders to the expression (5.17) to obtain

$$\begin{aligned} L_N &= 2\frac{N}{N'} - 2 + \frac{N}{N'} \frac{\log N'(\log N' + 3)}{2} \\ &= \frac{N}{N'} \left(2 + \frac{\log N'(\log N' + 3)}{2} \right) - 2, \end{aligned} \quad (5.18)$$

which completes the proof.

To exemplify, we give the latencies of hybrid decoders for several N and N' values in Table 5.2.

Table 5.2: Latencies of Hybrid Decoders

N	N'			
	1 (SC)	64	128	256
2048	4094	926	590	366
4096	8190	1854	1182	734
8192	16382	3710	2366	1470
16384	32766	7422	4734	2942

It is seen from Table 5.2 that significant reductions in decoding latency can be achieved with hybrid decoders. The ratio of the latencies of SC and hybrid decoders can be expressed as

$$\begin{aligned}
\text{Latency Gain} &= \frac{\text{SC Latency}}{\text{Hybrid Latency}} \\
&= \frac{2N - 2}{\frac{N}{N'} \left(2 + \frac{\log N'(\log N' + 3)}{2} \right) - 2} \\
&\approx \frac{4N'}{4 + \log N'(\log N' + 3)}, \tag{5.19}
\end{aligned}$$

for large N . The approximate latency gain expression (5.19) shows that the obtained reduction in latency depends only on N' as N increases. We give the approximate latency gains for various N' values in Table 5.3. The results in Table 5.3 show that even with a small component code block length (64), the decoding latency can be reduced by approximately 4.5 times. The latency is reduced more than 10 times when the component code block length is 256.

Table 5.3: Approximate Latency Gains

	N'			
	1 (SC)	64	128	256
Latency Gain	1	4.4	6.9	11.1

5.3 Implementation Results

The proposed weighted majority-logic decoder architecture is non-iterative and recursively defined. Similar to SC algorithm, a fully combinational implementation of the architecture is possible with or without pipelining. On the other hand, synchronous implementations with different levels of parallelism are also possible. In this section, we implement the proposed weighted majority-logic architecture with a fully combinational circuit using the structure in Algorithm 6. The implemented combinational weighted majority-logic decoders are fully flexible in the code rate so that for a fixed block length, any number of information bits can be decoded.

Table 5.4 shows the ASIC implementation results of the weighted majority-logic decoders for block lengths 64, 128 and 256. We use the same library as in

Section 4 for the implementations. The implementation results for the combinational SC decoder are also given in the same table for comparison since both decoders are fully-combinational architectures. We use $Q = 5$ bits for quantization, for which the performance loss is negligible as shown in Fig. 5.6 (the performance loss is similar with block lengths other than 64, although not shown here). We note that the ASIC synthesis could not be performed with the computation memory at hand for larger block lengths.

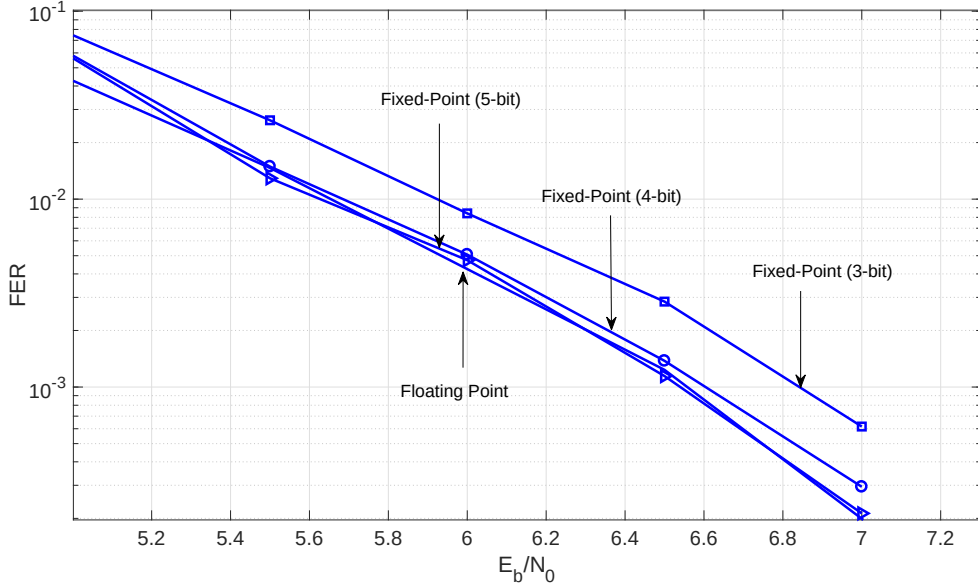


Figure 5.6: FER performance with different numbers of quantization bits ($N = 64$, $K = 57$)

The calculated latencies given in the table for the combinational SC and weighted majority-logic architectures do not represent the number of clock cycles in a decoding process, since the proposed architectures are fully combinational circuits. In the combinational case, calculated latencies serve as an analytical measure for the combinational delays of the corresponding decoders using similar basic logic blocks. The latency of a combinational SC decoder for block length N is given as $N - 1$ due to the use of adder/subtractor blocks with multiplexers. We use the expression (5.12) for the latency of combinational weighted majority-logic decoders.

The area and delay results in Table 5.4 verify our analyses for complexity and

Table 5.4: ASIC Implementation Results

N	2 ⁶	2 ⁷	2 ⁸	2 ⁶	2 ⁷	2 ⁸
Decoder	Combinational Weighted Majority-Logic			Combinational SC		
Area [mm ²]	0.32	1.08	3.03	0.153	0.338	0.759
Dec. Power [mW]	122	462	1960	99.8	138.8	158.7
Delay [ns]	8.0	10.8	14.7	22	45	91
Calculated Latency	27	35	44	63	127	255
Frequency [MHz]	125.0	92.6	68.0	45.5	22.2	11.0
Throughput [Gb/s]	8.0	11.8	17.4	2.92	2.83	2.81
Engy.-per-bit [pJ/b]	15.2	39.7	112.6	34.1	49.0	56.4
Hard. Eff. [Gb/s/mm ²]	25.0	10.9	5.7	19.1	8.4	3.7
Converted to 28 nm, 1.0 V						
Area [mm ²]	0.03	0.10	0.29	0.015	0.033	0.073
Dec. Power [mW]	22.4	85.0	360.8	18.4	25.5	29.2
Throughput [Gb/s]	25.7	37.9	55.9	9.39	9.10	9.03
Engy.-per-bit [pJ/b]	0.8	2.2	6.4	1.9	2.8	3.2
Hard. Eff. [Gb/s/mm ²]	830.2	362.8	190.7	633.8	278.0	122.9

latency given in the previous section. One can observe the increase in throughput with the increasing block length, which was stated in the latency analysis. The particular property of the proposed weighted majority-logic architecture is an advantage with respect to any SC decoder, for which the throughput tends to saturate with increasing block length even in a fully-unrolled architecture that is the combinational SC decoder. The results show that the low-latency architecture enables reaching throughput values at higher orders than SC decoder.

The algorithmic complexity $\mathcal{O}(N^{\log 3})$ of the proposed majority-logic architecture is clearly higher than $\mathcal{O}(N \log N)$, that is the complexity of conventional SC algorithm. Implementation results verify that the combinational weighted majority-logic decoder spans higher area than the combinational SC decoder. However, the hardware efficiencies of combinational weighted majority-logic decoders are higher than those of combinational SC decoders. The high-parallelism of the combinational weighted majority-logic decoder enables higher operating frequencies than those of combinational SC decoder. The increased hardware consumption and operating frequency lead to higher power consumption in combinational weighted majority-logic decoders with respect to the combinational SC decoders. The energy required per decoded bit is lower in weighted majority-logic decoder for block lengths 2^6 and 2^7 , but it is higher for block length 2^8 with respect to the SC decoder. The latter case is expected to be preserved for higher block lengths as the increase in hardware consumption has a higher order ($N^{\log 3}$) than the decrease order in operating frequency ($\frac{1}{\log^2 N}$) for the combinational weighted majority-logic decoder. However, the obtained values show that the combinational weighted majority-logic decoders are energy-efficient architectures.

From the results given in Table 5.4, one can predict that throughput values exceeding 100 Gb/s can be achieved with the combinational weighted majority-logic decoder for larger block lengths. Pipelining can also be applied to the proposed architecture. A fully-pipelined combinational weighted majority-logic decoder will contain a register after each basic element (comparators, adders and subtractors) to store the outputs. Such a decoder is expected to achieve throughput values on the order of Tb/s. These topics are to be studied in the

future.

There are no reported ASIC implementations of weighted majority-logic decoders for RM codes to the best of our knowledge. Therefore, a comparison with state-of-the-art majority-logic decoders is not possible.

5.4 Error Performance

In this section, we investigate the error performances of the weighted majority-logic and hybrid decoders for polar codes. For the construction of polar codes in the simulations, we use the Monte-Carlo method proposed in [6].

Polar code construction rule depends on the channel error probability characteristics. In AWGN channel, the signal SNR value, or the noise variance for normalized signal power, is the channel metric used for polar code construction. The error performances of codes optimized for different SNR values vary significantly, as demonstrated in [84]. In the simulation results, we present the performances of the polar codes that provide the best FER and BER performances in the observed E_b/N_0 region. The coded bit SNR values that codes are optimized for are specified in the graphs. We note that different optimization SNR values may result with the same codes. We report the lowest optimization SNR value in such cases. We also note that other optimization SNR values may provide similar or better performance at certain E_b/N_0 values than the ones specified in the graphs. For specific block lengths and code rates, polar codes become equivalent to RM codes at specific optimization SNR values. We state such situations in the graphs.

5.4.1 Weighted Majority-Logic Decoder

We investigate the error performance of the weighted majority-logic decoder by decoders for $N = 64$, $N = 256$ and $N = 1024$. The FER and BER performances

for $N = 64$ are given in Figures 5.7 - 5.10 for different code rates with the weighted majority-logic and SC decoders. In Figures 5.7 and 5.8, we consider the code rates specified by the RM coding rule. In Figures 5.9 and 5.10, arbitrary code rates are examined.

One can observe that weighted majority-logic causes a degradation in error performance with respect to SC algorithm. For the code rates specified by the RM coding rule, the performance loss increases as the code rate decreases for the same optimization SNR values. With different optimization SNR values that yield RM codes, the performance loss is decreased. For the arbitrary code rates, we observe that the performance loss is approximately 1 dB.

Next, we observe the FER and BER performance for $N = 256$. As in the case for $N = 64$, we first investigate the performance of code rates specified by RM coding rule in Figures 5.11 and 5.14. The error performances for arbitrary code rates are given in Figures 5.15 and 5.16. The results for $N = 256$ are similar to those for $N = 64$, so that, the performance loss is reduced with codes optimized at different SNR values than the ones optimized for SC decoding and the performance gap increases with decreasing code rate. Furthermore, the error performance gap between weighted majority-logic and SC decoding is observed to widen for $N = 256$ with respect for $N = 64$. For example, the performance gap of the code (64,50) is approximately 1 dB (Figures 5.9 and 5.10), whereas the gap is approximately 1 dB for the code (256,200) (Figures 5.15 and 5.16). In order to investigate the phenomenon, we investigate the error performances for $N = 1024$, which are given in Figures 5.17 and 5.18.

The error gap is observed to increase from Figures 5.17 and 5.18 for $N = 1024$. For the code (1024,800), which has the code rate considered in the example above, the gap is approximately 2.5 dB. The performance degradation with weighted majority-logic decoding is more severe for the case of (1024,512). One can conclude from the presented results that weighted majority-logic decoding is suitable for decoding codes with small block lengths and high coding rates.

In order to benefit from the low-latency decoding characteristic of weighted

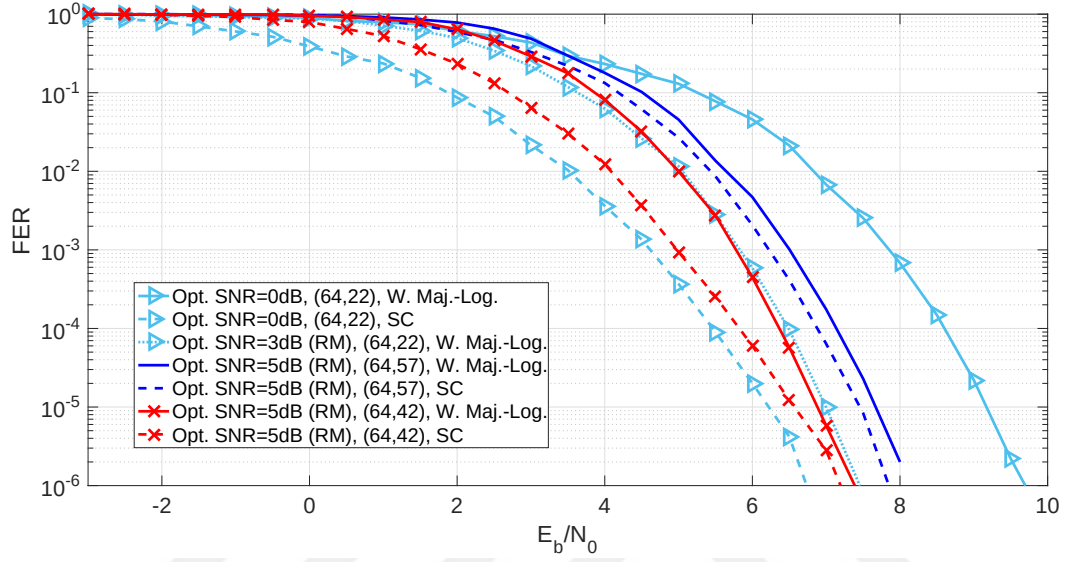


Figure 5.7: FER performance of weighted majority-logic and SC decoders ($N = 64$)

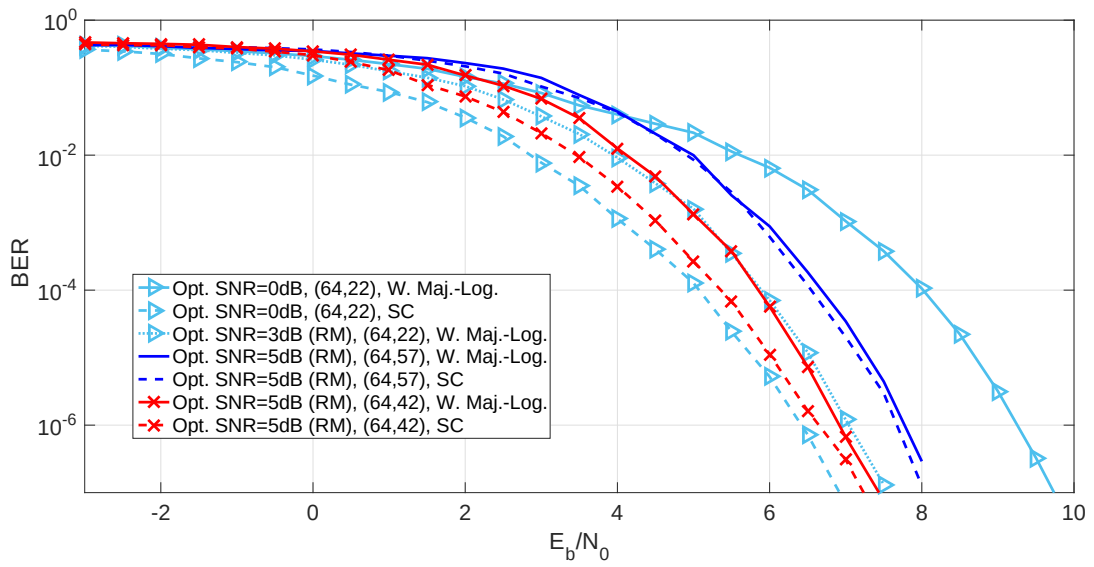


Figure 5.8: BER performance of weighted majority-logic and SC decoders ($N = 64$)

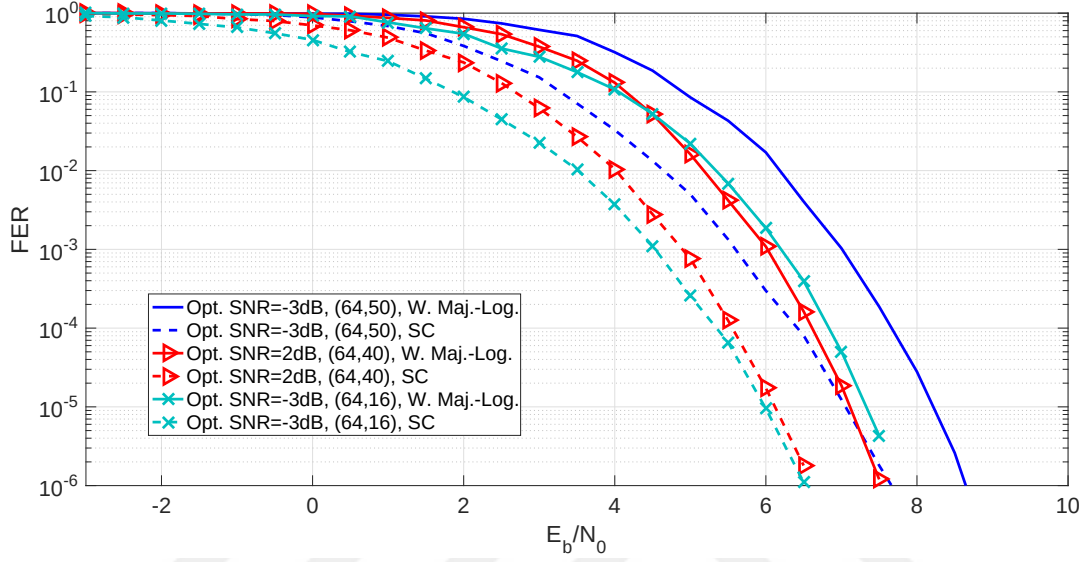


Figure 5.9: FER performance of weighted majority-logic and SC decoders ($N = 64$)

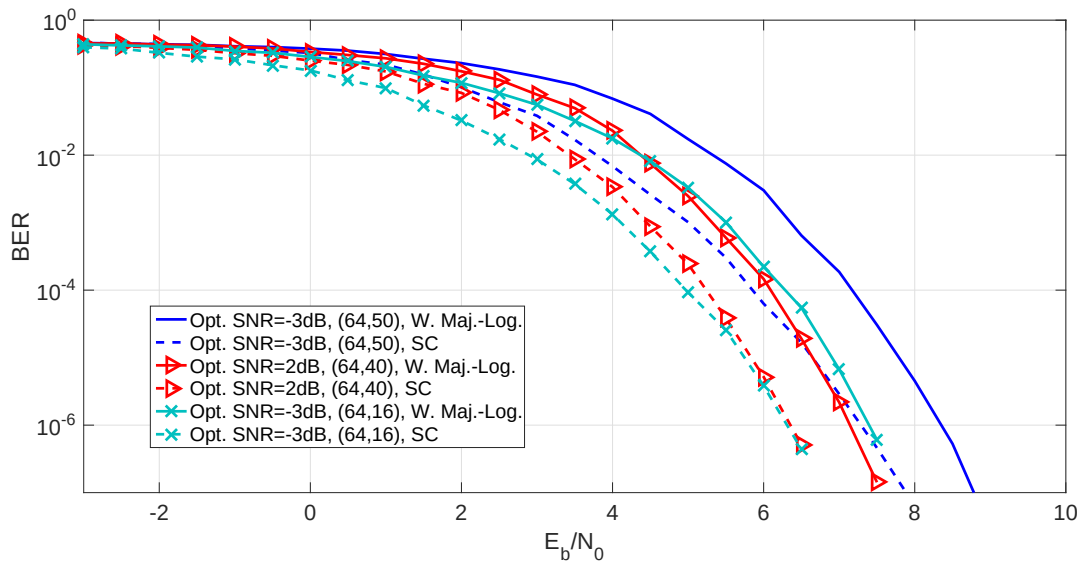


Figure 5.10: BER performance of weighted majority-logic and SC decoders ($N = 64$)

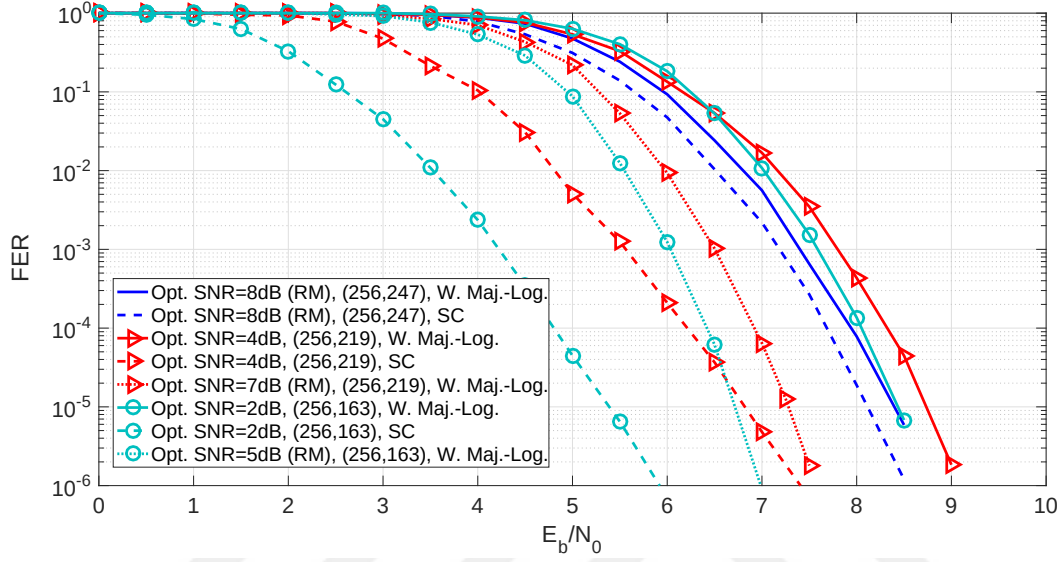


Figure 5.11: FER performance of weighted majority-logic and SC decoders ($N = 256$)

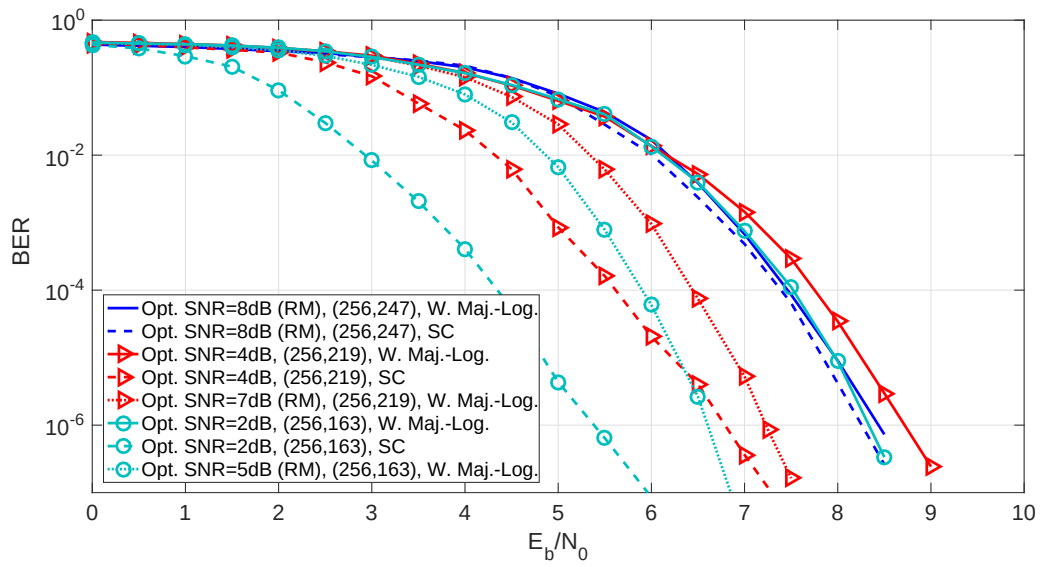


Figure 5.12: BER performance of weighted majority-logic and SC decoders ($N = 256$)

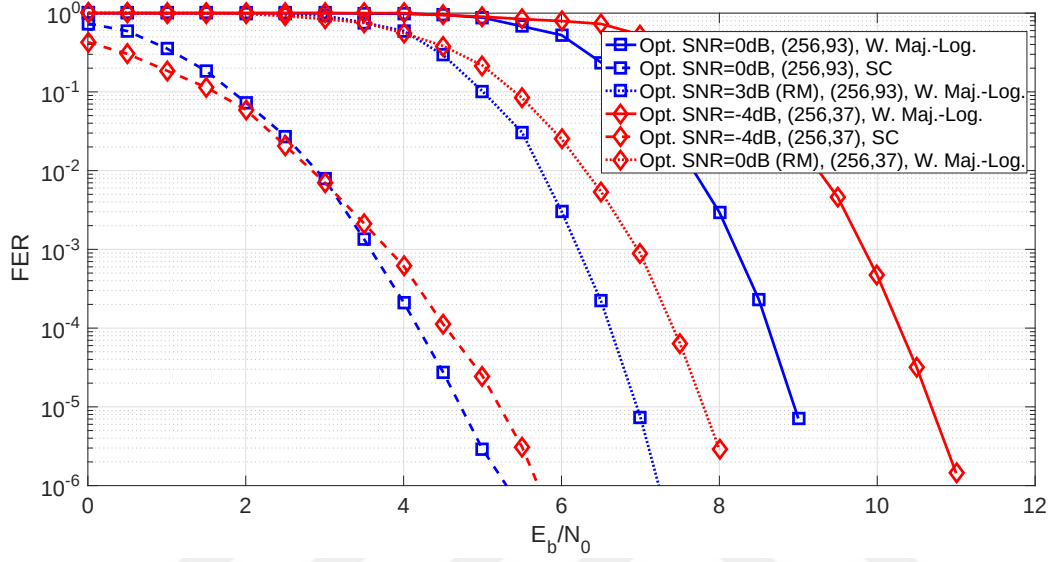


Figure 5.13: FER performance of weighted majority-logic and SC decoders ($N = 256$)

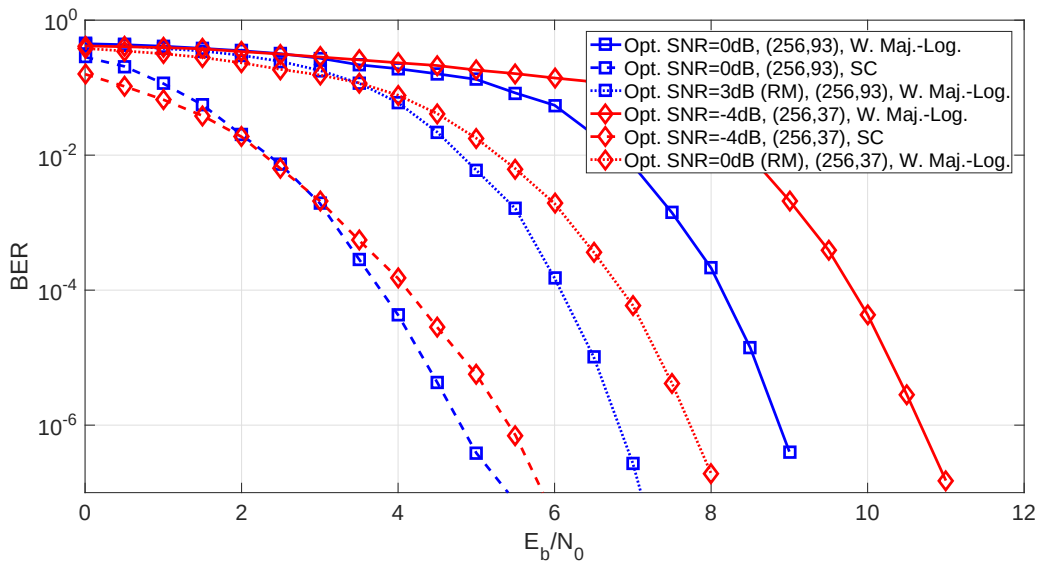


Figure 5.14: BER performance of weighted majority-logic and SC decoders ($N = 256$)

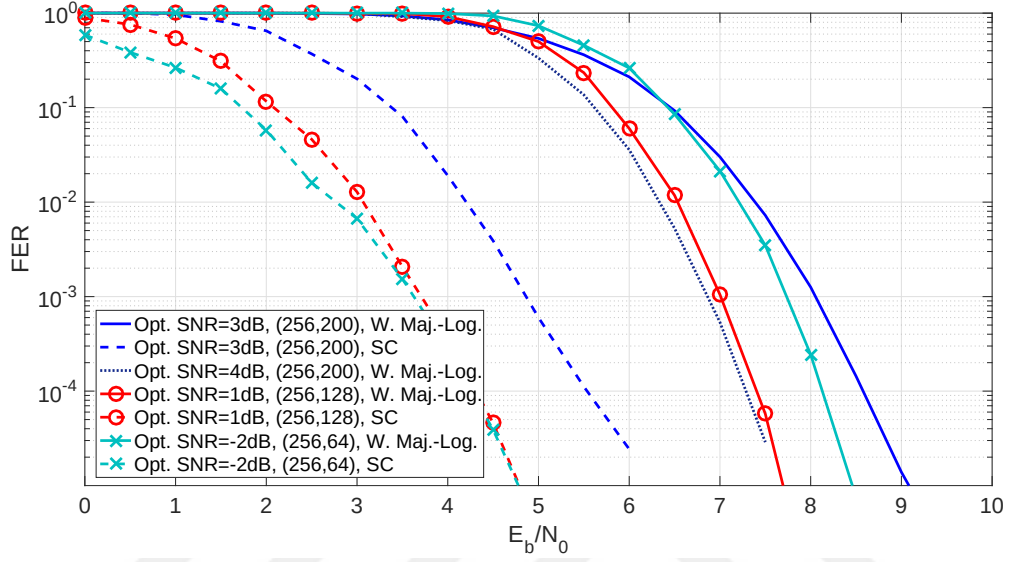


Figure 5.15: FER performance of weighted majority-logic and SC decoders ($N = 256$)

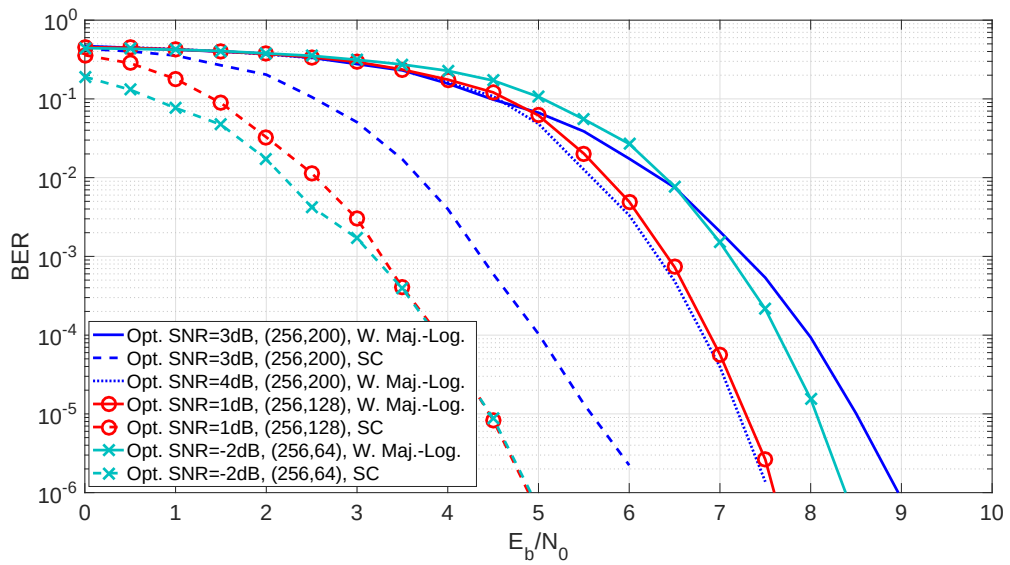


Figure 5.16: BER performance of weighted majority-logic and SC decoders ($N = 256$)

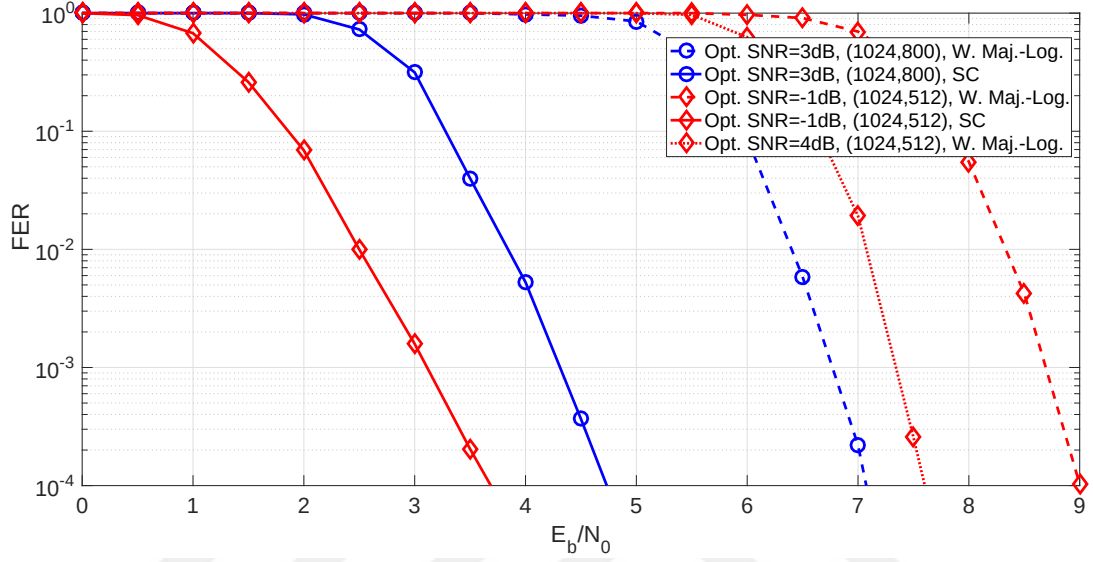


Figure 5.17: FER performance of weighted majority-logic and SC decoders ($N = 1024$)

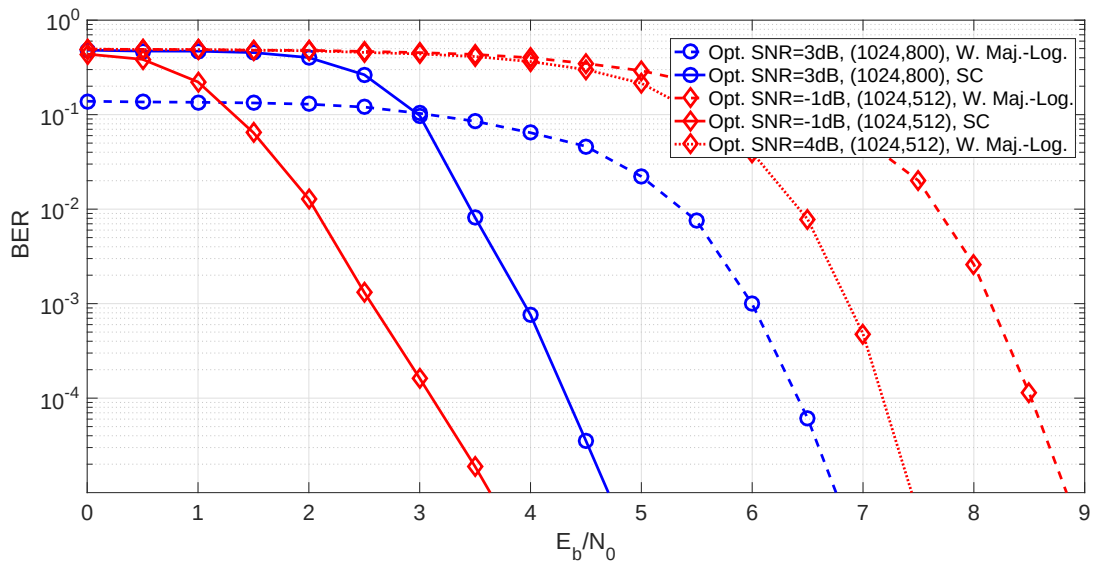


Figure 5.18: BER performance of weighted majority-logic and SC decoders ($N = 1024$)

majority-logic algorithm at higher block lengths and lower coding rates, we proposed a hybrid decoding scheme in Section 5.2.2. In the next subsection, we investigate the error performance characteristics of hybrid decoding for polar codes.

5.4.2 Hybrid Decoder

We use the polar codes of (8192, 4096) and (8192, 6554) to investigate the error performances of hybrid decoders. Figures 5.20 - 5.22 show the FER and BER performances of the considered codes. The codes are optimized for SNR values of $0dB$ and $3dB$, respectively.

The figures show the performance gain obtained by the hybrid architecture with respect to the weighted majority-logic decoding. For performance comparison, the RM codes (8192, 7099) and (8192, 4096) are used. The error performance is observed to improve considerably with hybrid decoding with respect to weighted majority-logic decoding, even for the code rate $1/2$. The error performances can further be improved by choosing the frozen bit locations according to the decoder architecture. We use the Monte-Carlo method to determine the frozen bit locations under the hybrid decoding with different N' values. The performances of the codes optimized for hybrid decoding are also given in the same figures. It can be observed that for the considered N' values, the performance degradation becomes at most 1.1 dB for the considered code rates. It is also observed that the performance loss becomes independent of the code rate. The results show that with proper choice of frozen bit indexes, the error performance of hybrid decoders improve significantly.

Finally, we investigate the characteristics of a hybrid- N' decoder with increasing N . Figures 5.20 - 5.22 show the FER and BER performances of hybrid-256 decoder for $N = 8192$ and $N = 16384$. From the figures, one can observe that the performance gap closes as we increase the block length. Thus, we can conclude that the hybrid decoders can achieve a significant latency gain with tolerable error performance loss for large block lengths.

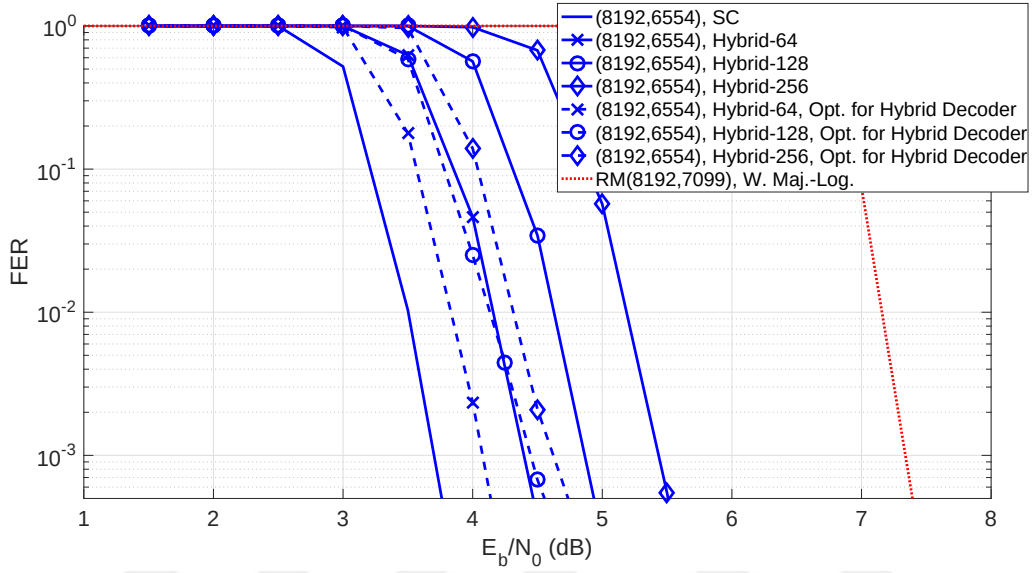


Figure 5.19: FER performance of hybrid decoders ($N = 8192$, $K = 6554$)

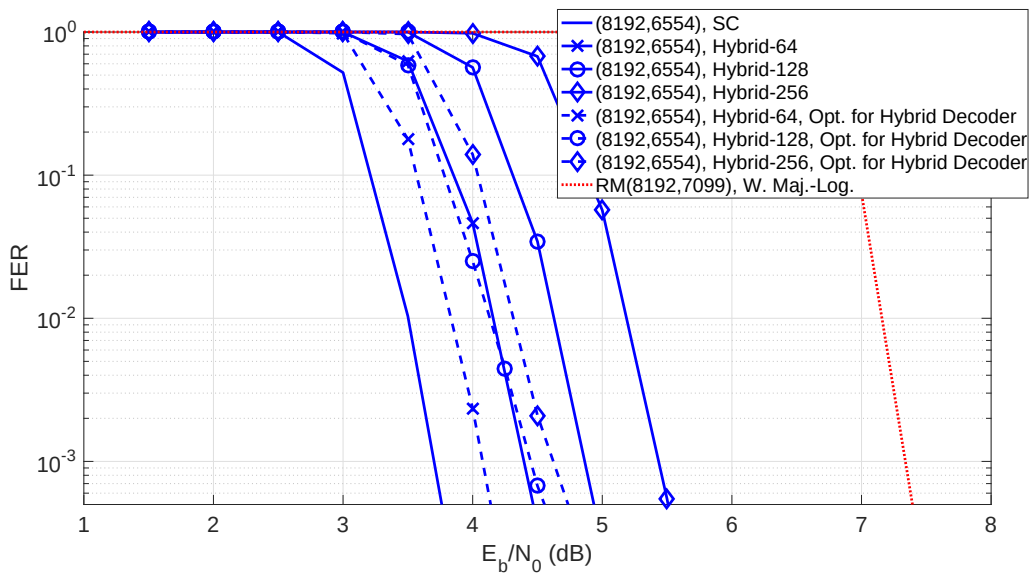


Figure 5.20: BER performance of hybrid decoders ($N = 8192$, $K = 6554$)

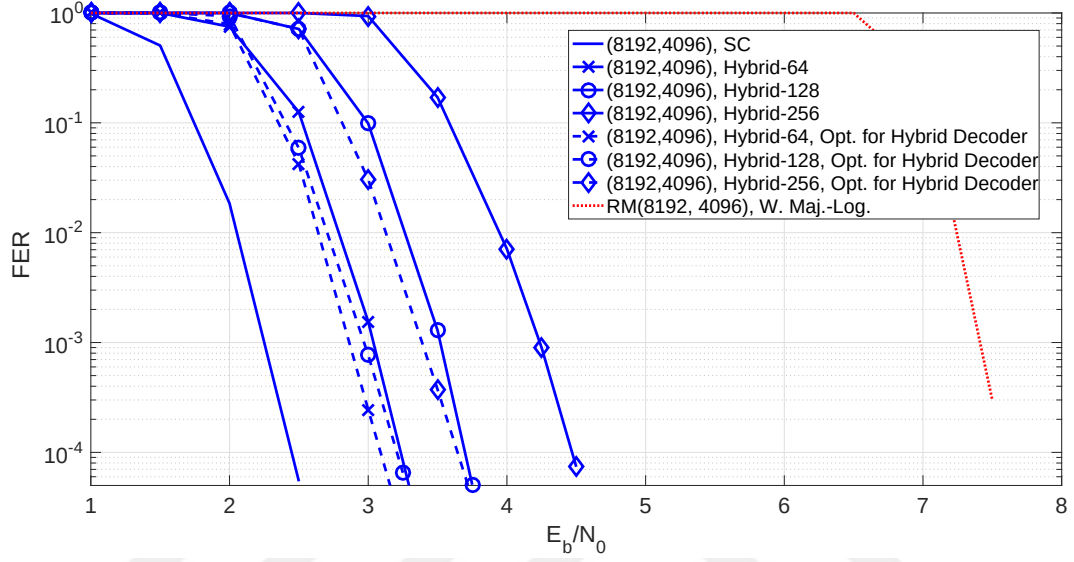


Figure 5.21: FER performance of hybrid decoders ($N = 8192$, $K = 4096$)

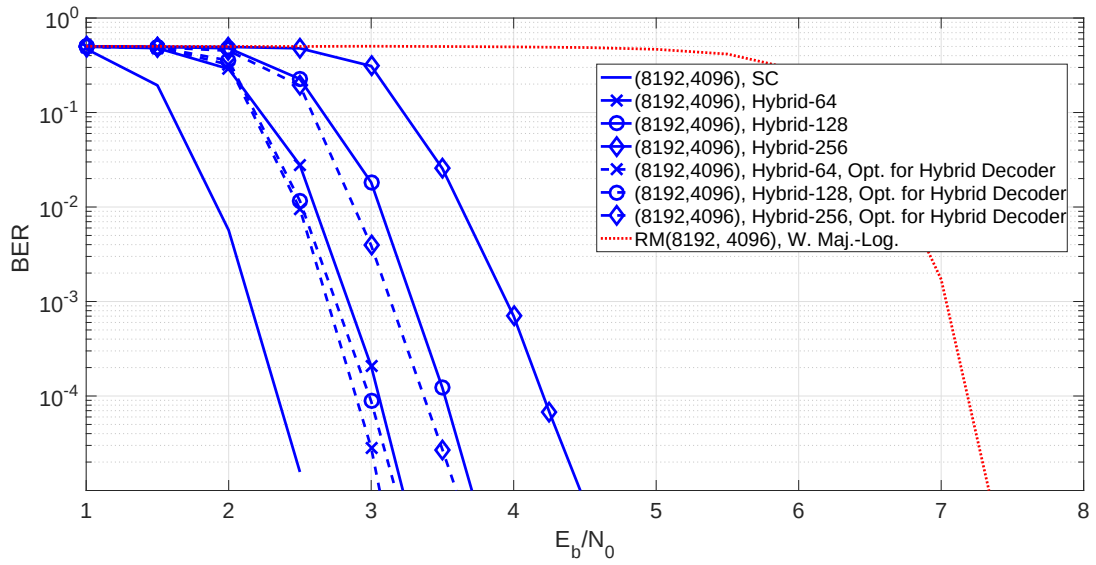


Figure 5.22: BER performance of hybrid decoders ($N = 8192$, $K = 4096$)

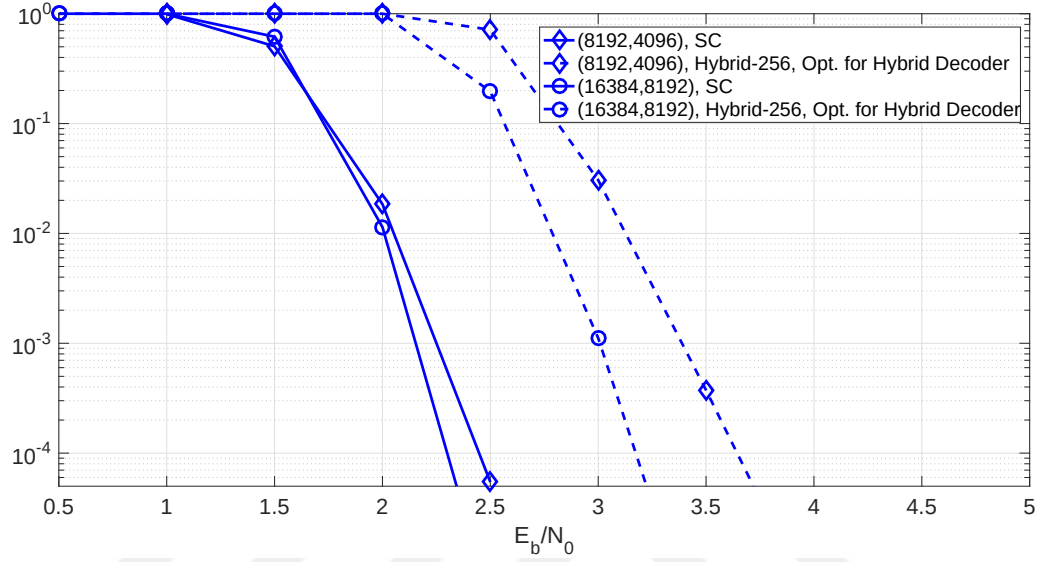


Figure 5.23: FER performance of hybrid-256 decoders for $N = 8192$ and $N = 16384$

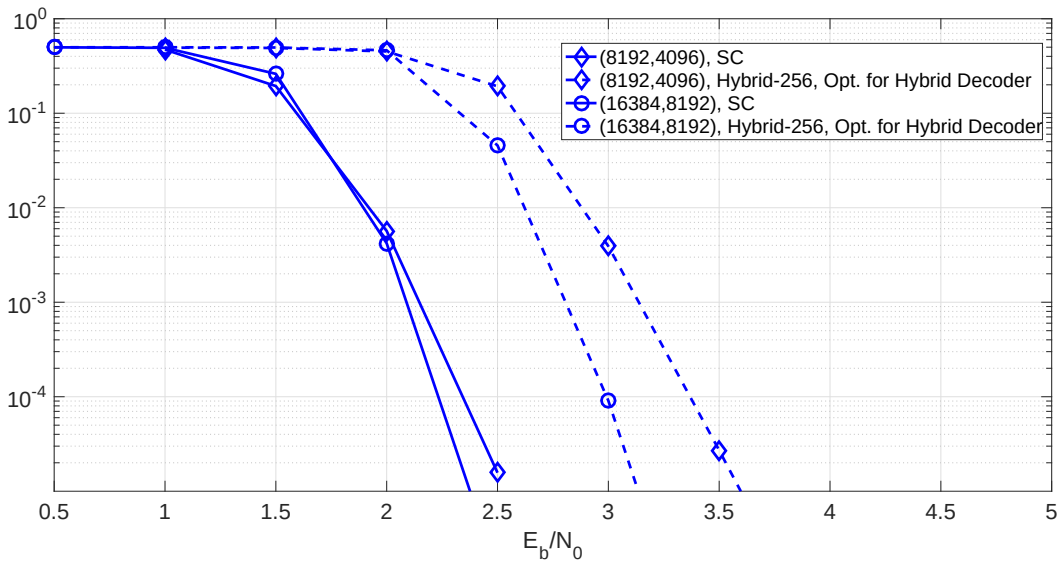


Figure 5.24: BER performance of hybrid-256 decoders for $N = 8192$ and $N = 16384$

5.5 Summary of the Chapter

In this chapter, we investigated weighted majority-logic decoder of [31] for polar codes. First, we presented a novel recursive description for the weighted majority-logic decoding for polar codes. It was analytically shown that the latency of the proposed decoder is $\mathcal{O}(\log^2 N)$ and the algorithmic complexity is $\mathcal{O}(N^{\log 3})$.

We implemented the proposed decoder as a fully combinational decoder using the recursive description in a flexible manner. Post-synthesis ASIC results showed that the proposed weighted majority-logic decoders can achieve a throughput of 17.4 Gb/s for a 90 nm 1.3 V technology. The achieved throughput was shown to become approximately 55 Gb/s with the technology normalized to 28 nm 1.0 V by analytical formulas. The implementation results also showed that the proposed decoders are energy-efficient decoders. The error performance of the proposed decoders are investigated for short and medium block lengths. We showed that the performance loss of weighted majority-logic decoding with respect to SC decoding depends on the code rate, block length and optimization SNR values that the codes are designed for.

In order to reduce the error performance loss, we proposed hybrid decoders that employ weighted majority-logic algorithm for decoding component codes in a SC decoder. Such decoders benefit from the low-latency decoding of weighted majority-logic with an improved error performance. We performed an analytical analysis for the latency reduction obtained by such decoders with respect to SC algorithm and simulations for their error performances. We demonstrated that a latency gain of approximately 7 times can be achieved by hybrid-128 and 11 times by hybrid-256 decoders with respect to SC decoding with error performance degradations less than or approximately equal to 1dB, with the performance loss decreasing with increasing N .

Chapter 6

Conclusions and Future Works

In this chapter, we first give a summary and present a final comparison between different state-of-the-art decoders for Turbo, LDPC and polar codes and the proposed decoders in the thesis. Then, we suggest some future works on the subjects covered.

6.1 Conclusions

In this thesis, we designed and implemented high-throughput and energy-efficient decoder architectures for polar codes, mainly targeting communications services such as optical communications, mMTC and Terahertz communications. First, we proposed a flexible combinational architecture for SC polar decoders in Chapter 4. The proposed combinational SC decoder operates at much lower clock frequencies compared to typical synchronous SC decoders and decodes a codeword in one long clock cycle. Due to the low operating frequency, the combinational decoder consumes less dynamic power, which reduces the overall power consumption. We also proposed pipelined (Section 4.1.3) and hybrid-logic (Section 4.1.4) decoders based on combinational SC decoders. We provided the analytical estimates for the combinational delay and hardware consumption of combinational

Table 6.1: Comparison of State-of-the-Art ECC Decoding Schemes

	[22]	[26]	Comb. SC [85]	Comb. W. Maj.-Log.	[44]	[76]	[69]**	
Code	Turbo	LDPC	Polar	Polar	Polar	Polar	Polar	
Algorithm	BCJR	BP	SC	Maj.-Log.	SSC	SCL ($L=4$)	BP	
Design	Fabricated	Post- layout	Post- synthesis	Post- synthesis	Post- synthesis	Post- synthesis	Fabricated	
Block Length	All LTE Block Lengths	672	1024	256	1024	1024	1024	
Code Rate	All LTE Code Rates	1/2, 5/8, 3/4, 13/16	Any	Any	1/2	1/2	Any	
Area [mm ²]	5.070	0.575	1.676	1.580	0.69	2.14	1.476	
Voltage [V]	0.81	1.1	1.0	1.0	1.0	-	1.0	0.475
Power [mW]	1256.7	273 [†]	81.5	837.6	215	718	477.5	18.6
TP [Gb/s]	1.16	9.25	3.54	24.09	1.86 ^{††}	0.40	4.68	0.78
Engy.-per-bit [pJ/b]	1083.4	29.4	23.0	34.8	115	1790 [*]	102.1	23.8
Hard. Eff. [Gb/s/mm ²]	0.23	16.08	2.11	15.24	2.7	0.19	3.1	0.5

* Not presented in the paper, calculated from the presented results

** Results are given for (1024, 512) code at 4dB SNR with 6.57 iterations

[†] Power consumption is for rate-1/2 code at SNR 2.5 dB with 7 iterations

^{††} Information bit throughput

SC decoders in terms of basic circuit component parameters in Section 4.2 and throughput gain obtained by the hybrid-logic decoders in Section 4.1.4.

Second, we investigated weighted majority-logic algorithm of [31] to decode bit-reversed polar codes in Chapter 5. For this purpose, we gave a novel recursive definition for the weighted majority-logic algorithm in Section 5.1.1 in order to define and implement the decoder for polar codes. In Section 5.2, we showed by analytical estimates that the decoder latency is $\mathcal{O}(\log^2 N)$ and algorithmic complexity is $\mathcal{O}(N^{\log 3})$ for block length N . We implemented the algorithm with fully combinational circuitry using the proposed recursive definition (Section 5.3). We also proposed a hybrid decoder in Section 5.1.2 that employs weighted majority-logic decoding to decode component codes of a polar code in a SC decoder. We provided an analytical latency analysis and error performance simulations to show that high latency gains can be obtained with a small degradation in error performance by the hybrid decoders with respect to SC decoding (Section 5.2).

In Table 6.1, we give implementation results of the decoders proposed in this thesis and examples for state-of-the-art decoders for Turbo, LDPC and polar codes. The chosen examples reflect the general characteristics of state-of-the-art implementations in the corresponding schemes. The implementation results are converted to 65 nm technology for a fair comparison. Note that the provided ASIC implementation results are post-synthesis or post-layout results, except the works in [22] and [69] which are measurement results from fabricated chips.

The error performances of the considered decoder implementations were mentioned in the corresponding chapters. For polar codes, it was stated in Section 3.1 that the error performances of SC and BP decoders are close and fall short of the performances of SCL decoders, especially when CRC-appended polar codes are used. For the considered LDPC decoder of block length 672, the error performance is similar to that of a SC decoder for block length 1024, as given in Section 4.3.1.4. It was shown in Section 5.4 that the error performance of weighted majority-logic decoding is poor compared to SC decoding, with the performance gap depending on the block length, code rate and optimization SNR values. The best error performance is achieved by the Turbo decoder among the decoders

considered in the table [77]. Note that the error performance characteristics summarized above are valid for FER values higher than 10^{-5} .

Keeping the explained error performances in mind, major observations from the results in the table can be listed as:

- Turbo decoders and SCL polar decoders consume much higher energy per decoded bit, have poorer hardware-efficiency and achieve less throughput than state-of-the-art LDPC and all other types of polar decoders as a penalty for better error performance.
- Combinational SC decoders achieve higher throughput than Turbo, SCL and SC polar decoders with higher energy-efficiency and flexibility.
- Combinational SC decoders achieve comparable throughput with better energy-efficiency and higher flexibility compared to BP polar decoders. They achieve lower throughput with higher flexibility with respect to BP LDPC decoders. It should be noted that the performances of BP decoders are dependent on the number of decoder iterations and input SNR values.
- Combinational weighted majority-logic polar decoders can achieve highest throughput among the presented decoders with high energy-efficiency and flexibility at the expense of error performance. Such decoders are suitable for codes with short block lengths and high code rates.

With the results obtained throughout the thesis and observations given above, we can list some conclusions as:

- Combinational SC decoders offer a fast, energy-efficient, and flexible alternative for implementing polar decoders. With combinational SC decoders, polar codes can compete with LDPC codes in terms of decoder hardware performance.
- Pipelined combinational SC decoders offer an easy trade-off between throughput and hardware consumption.

- Hybrid-logic decoders offer an energy-efficient method to improve the throughput characteristics of synchronous SC decoders for very long block lengths.
- Combinational weighted majority-logic decoders can be used to decode polar codes with a significantly smaller latency than that of SC algorithm for short block lengths and high code rates.
- Hybrid decoders achieve a considerable amount of latency gain with respect to SC decoders with acceptable levels of error performance degradation by properly designed codes for long block lengths and any code rate. Such decoders are good candidates for applications with very high throughput requirements.

6.2 Suggestions for Future Work

The subjects covered in this thesis are open for future studies. We mention some study topics in this section.

6.2.1 Combinational SC Decoder

The presented ASIC implementation results for the combinational SC decoder are post-synthesis results obtained by Cadence Encounter RTL Compiler software with 90 nm CMOS library. The results were also converted to 65 nm and 28 nm to estimate the limits of the proposed architecture with newer CMOS technologies. It would be interesting to measure the characteristics of the decoder on an actual ASIC chip implemented with newer VLSI technologies and compare them with the provided results in this thesis.

The FinFET technology has attracted attention as an alternative to CMOS technology in recent years. It has been shown to achieve superior characteristics than those of CMOS circuits in several aspects. However, power density

and cooling problems have been reported in FinFET circuits that require careful thermal management [86]. Combinational SC decoders may be suitable for implementation with FinFET technology to avoid such problems since their operating frequencies are much smaller than those of synchronous decoders. Investigating the performance of combinational SC decoders with FinFET technology is also an interesting topic to be studied.

6.2.2 Weighted Majority-Logic Decoding for Polar Codes

The proposed recursive definition makes it easy to design majority-logic decoders for longer block lengths. However, such decoders could not be synthesized with the resources at hand. The architecture characteristics for longer block lengths should be investigated in the next step.

The error performance of the weighted majority-logic decoders for polar codes are poor compared to the SC decoding algorithm. This phenomenon arises from the fact that majority-logic decoders do not fully exploit the channel polarization effect. Methods to improve the error performance of majority-logic algorithm for polar codes could further be investigated. In the thesis, we propose hybrid decoders to achieve better error performance with reduced latency. The hybrid decoders were shown to achieve less performance degradation with proper code design that reflects the effects of majority-logic decoding at component codes. Theoretical design and analysis of these codes optimized for hybrid decoding can be interesting and reserved for future study. Furthermore, the error performances of such codes at very low BER/FER region should also be investigated.

Bibliography

- [1] C. E. Shannon, “A mathematical theory of communication,” *Bell Syst. Tech. J.*, vol. 27, pp. 379–423, 623–656, 1948.
- [2] C. Berrou, A. Glavieux, and P. Thitimajshima, “Near Shannon limit error-correcting coding and decoding: turbo-codes,” in *IEEE Int. Conf. Commun. (ICC)*, vol. 2, pp. 1064–1070 vol.2, May 1993.
- [3] R. G. Gallager, *Low Density Parity-Check Codes*. PhD thesis, MIT Press, Cambridge, MA, 1963.
- [4] D. J. C. MacKay and R. M. Neal, “Near shannon limit performance of low density parity check codes,” *Electron. Lett.*, vol. 33, pp. 457–458, Mar 1997.
- [5] D. A. Spielman, “Linear-time encodable and decodable error-correcting codes,” *IEEE Transactions on Information Theory*, vol. 42, pp. 1723–1731, Nov 1996.
- [6] E. Arıkan, “Channel polarization: a method for constructing capacity-achieving codes for symmetric binary-input memoryless channels,” *IEEE Trans. Inform. Theory*, vol. 55, pp. 3051–3073, July 2009.
- [7] Chairman’s notes, 3GPP TSG RAN WG1 #87 meeting.
- [8] F. Kienle, N. Wehn, and H. Meyr, “On complexity, energy- and implementation-efficiency of channel decoders,” *IEEE Transactions on Communications*, vol. 59, pp. 3301–3310, December 2011.
- [9] 3GPP TR 45.820 V13.1.0 (2015-11), “Cellular system support for ultra-low complexity and low throughput internet of things (CIoT),”

- [10] S. Scholl, S. Weithoffer, and N. Wehn, “Advanced iterative channel coding schemes: When shannon meets moore,” in *2016 9th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, pp. 406–411, Sept 2016.
- [11] S. Borkar, “Design challenges of technology scaling,” *IEEE Micro*, vol. 19, pp. 23–29, Jul 1999.
- [12] 3GPP TR 38.913 V14.2.0 (2017-03), “Study on scenarios and requirements for next generation on new radio access technologies,”
- [13] R1-167272, “Implementation aspects of eMBB coding schemes,” Nokia, Alcatel-Lucent Shanghai Bell, Verizon Wireless, Xilinx.
- [14] R1-167276, “Evaluation criteria for URLLC and mMTC coding schemes,” Nokia, Alcatel-Lucent Shanghai Bell.
- [15] G. Tzimpragos, C. Kachris, I. B. Djordjevic, M. Cvijetic, D. Soudris, and I. Tomkos, “A survey on FEC codes for 100 G and beyond optical networks,” *IEEE Commun. Surveys Tutorials*, vol. 18, pp. 209–221, Firstquarter 2016.
- [16] T. Ahmad, *Polar codes for optical communications*. PhD thesis, Bilkent Univ., Ankara, 2016.
- [17] A. E. W. I. P. Kaminow, T. Li, *Optical Fiber Communications VIB - Systems and Networks*. Academic Press, 2013.
- [18] F. Khan, “Multi-comm-core architecture for terabit-per-second wireless,” *IEEE Commun. Magazine*, vol. 54, pp. 124–129, April 2016.
- [19] A. Li, X. Chen, G. Gao, and W. Shieh, “Transmission of 1 Tb/s unique-word DFT-spread OFDM superchannel over 8000 km EDFA-only SSMF link,” *J. Lightwave Tech.*, vol. 30, pp. 3931–3937, Dec 2012.
- [20] G. Fettweis, F. Guderian, and S. Krone, “Entering the path towards terabit/s wireless links,” in *2011 Design, Automation Test in Europe*, pp. 1–6, March 2011.

- [21] J. Yeon and H. Lee, "High-performance iterative BCH decoder architecture for 100 Gb/s optical communications," in *2013 IEEE Intern. Symp. Circuits and Syst. (ISCAS2013)*, pp. 1344–1347, May 2013.
- [22] G. Wang, H. Shen, Y. Sun, J. R. Cavallaro, A. Vosoughi, and Y. Guo, "Parallel interleaver design for a high throughput HSPA+/LTE multi-standard turbo decoder," *IEEE Trans. Circuits and Syst. I, Reg. Papers*, vol. 61, pp. 1376–1389, May 2014.
- [23] C. Roth, S. Belfanti, C. Benkeser, and Q. Huang, "Efficient parallel turbo-decoding for high-throughput wireless systems," *IEEE Trans. Circuits and Syst. I, Reg. Papers*, vol. 61, pp. 1824–1835, June 2014.
- [24] A. Li, L. Xiang, T. Chen, R. G. Maunder, B. M. Al-Hashimi, and L. Hanzo, "VLSI implementation of fully parallel lte turbo decoders," *IEEE Access*, vol. 4, pp. 323–346, 2016.
- [25] M. Weiner, M. Blagojevic, S. Skotnikov, A. Burg, P. Flatresse, and B. Nikolic, "A scalable 1.5-to-6Gb/s 6.2-to-38.1mW LDPC decoder for 60GHz wireless networks in 28nm UTBB FDSOI," in *2014 IEEE Intern. Solid-State Circuits Conf. Digest of Technical Papers (ISSCC)*, pp. 464–465, Feb 2014.
- [26] S. Ajaz and H. Lee, "Multi-Gb/s multi-mode LDPC decoder architecture for IEEE 802.11ad standard," in *2014 IEEE Asia Pacific Conf. Circuits and Syst. (APCCAS)*, pp. 153–156, Nov 2014.
- [27] K. Zhang, X. Huang, and Z. Wang, "A high-throughput LDPC decoder architecture with rate compatibility," *IEEE Trans. Circuits and Syst. I, Reg. Papers*, vol. 58, pp. 839–847, April 2011.
- [28] Z. Zhang, V. Anantharam, M. J. Wainwright, and B. Nikolic, "An efficient 10gbase-t ethernet ldpc decoder design with low error floors.," *J. Solid-State Circuits*, vol. 45, no. 4, pp. 843–855, 2010.
- [29] C.-C. Wong and H.-C. Chang, "Reconfigurable turbo decoder with parallel architecture for 3GPP LTE system," *IEEE Trans. Circuits and Syst. II, Express Briefs*, vol. 57, pp. 566–570, July 2010.

- [30] A. Blanksby and C. Howland, "A 690-mW 1-Gb/s 1024-b, rate-1/2 low-density parity-check code decoder," *IEEE J. Solid-State Circuits*, vol. 37, pp. 404–412, Mar. 2002.
- [31] V. D. Kolesnik, "Probabilistic decoding of majority codes," *Probl. Peredachi Inform.*, pp. 7:3–12, July 1971.
- [32] E. Arıkan and E. Telatar, "On the rate of channel polarization," in *2009 IEEE International Symposium on Information Theory*, pp. 1493–1495, June 2009.
- [33] R. Mori and T. Tanaka, "Performance of polar codes with the construction using density evolution," *IEEE Communications Letters*, vol. 13, pp. 519–521, July 2009.
- [34] I. Tal and A. Vardy, "How to construct polar codes," *IEEE Transactions on Information Theory*, vol. 59, pp. 6562–6582, Oct 2013.
- [35] R. Pedarsani, S. H. Hassani, I. Tal, and E. Telatar, "On the construction of polar codes," in *2011 IEEE International Symposium on Information Theory Proceedings*, pp. 11–15, July 2011.
- [36] P. Trifonov, "Efficient design and decoding of polar codes," *IEEE Transactions on Communications*, vol. 60, pp. 3221–3227, November 2012.
- [37] H. Li and J. Yuan, "A practical construction method for polar codes in awgn channels," in *IEEE 2013 Tencon - Spring*, pp. 223–226, April 2013.
- [38] D. Wu, Y. Li, and Y. Sun, "Construction and block error rate analysis of polar codes over awgn channel based on gaussian approximation," *IEEE Communications Letters*, vol. 18, pp. 1099–1102, July 2014.
- [39] M. Plotkin, "Binary codes with specified minimum distance," *IRE Trans. Inform. Theory*, vol. 6, pp. 445–450, September 1960.
- [40] A. Mishra, A. Raymond, L. Amaru, G. Sarkis, C. Leroux, P. Meinerzhagen, A. Burg, and W. Gross, "A successive cancellation decoder ASIC for a 1024-bit polar code in 180nm CMOS," in *IEEE Asian Solid State Circuits Conf. (A-SSCC)*, pp. 205–208, Nov. 2012.

- [41] Y. Fan and C.-Y. Tsui, "An efficient partial-sum network architecture for semi-parallel polar codes decoder implementation," *IEEE Trans. Signal Process.*, vol. 62, pp. 3165–3179, June 2014.
- [42] C. Leroux, A. Raymond, G. Sarkis, and W. Gross, "A semi-parallel successive-cancellation decoder for polar codes," *IEEE Trans. Signal Process.*, vol. 61, pp. 289–299, Jan. 2013.
- [43] B. Yuan and K. Parhi, "Low-latency successive-cancellation polar decoder architectures using 2-bit decoding," *IEEE Trans. Circuits Syst. I, Regular Papers*, vol. 61, pp. 1241–1254, Apr. 2014.
- [44] P. Giard, A. Balatsoukas-Stimming, G. Sarkis, C. Thibault, and W. J. Gross, "Fast low-complexity decoders for low-rate polar codes," *Journal of Signal Processing Systems*, Aug 2016.
- [45] T. Che, J. Xu, and G. Choi, "Tc: Throughput centric successive cancellation decoder hardware implementation for polar codes," in *2016 IEEE Int. Conf. Acoustics, Speech and Signal Process. (ICASSP)*, pp. 991–995, March 2016.
- [46] I. Tal and A. Vardy, "List decoding of polar codes," in *Proc. IEEE Int. Symp. Inform. Theory (ISIT)*, pp. 1–5, July 2011.
- [47] E. Arıkan, "A performance comparison of polar codes and Reed-Muller codes," *IEEE Commun. Lett.*, vol. 12, pp. 447–449, June 2008.
- [48] S. Kahraman and M. E. Çelebi, "Code based efficient maximum-likelihood decoding of short polar codes," in *2012 IEEE International Symposium on Information Theory Proceedings*, pp. 1967–1971, July 2012.
- [49] O. Afisiadis, A. Balatsoukas-Stimming, and A. Burg, "A low-complexity improved successive cancellation decoder for polar codes," in *2014 48th Asilomar Conference on Signals, Systems and Computers*, pp. 2116–2120, Nov 2014.
- [50] K. Niu and K. Chen, "Stack decoding of polar codes," *Electronics Letters*, vol. 48, pp. 695–697, June 2012.

- [51] U. U. Fayyaz and J. R. Barry, “Low-complexity soft-output decoding of polar codes,” *IEEE Journal on Selected Areas in Communications*, vol. 32, pp. 958–966, May 2014.
- [52] I. Dumer and K. Shabunov, “Soft-decision decoding of Reed-Muller codes: recursive lists,” *IEEE Trans. Inform. Theory*, vol. 52, pp. 1260–1266, Mar. 2006.
- [53] K. Niu, K. Chen, J. Lin, and Q. T. Zhang, “Polar codes: Primary concepts and practical decoding algorithms,” *IEEE Communications Magazine*, vol. 52, pp. 192–203, July 2014.
- [54] I. Reed, “A class of multiple-error-correcting codes and the decoding scheme,” *Trans. of the IRE Prof. Group Inform. Theory*, vol. 4, p. 38–49, Sep. 1954.
- [55] D. E. Muller, “Applications of boolean algebra to switching circuits design and to error detection,” *Trans. IRE Prof. Group Electronic Computers*, vol. 3, pp. 6–12, Sep. 1954.
- [56] I. Dumer and R. Krichevskiy, “Soft-decision majority decoding of Reed-Muller codes,” *IEEE Trans. Inform. Theory*, vol. 46, pp. 258–264, Jan. 2000.
- [57] S. Lin and D. J. Costello, *Error Control Coding, Second Edition*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2004.
- [58] E. Arıkan, “Polar codes: A pipelined implementation,” in *Proc. Int. Symp. Broadband Commun. (ISBC2010)*, pp. Melaka, Malaysia, 2010.
- [59] C. Leroux, I. Tal, A. Vardy, and W. J. Gross, “Hardware architectures for successive cancellation decoding of polar codes,” in *2011 IEEE Intern. Conf. Acoustics, Speech and Signal Process. (ICASSP)*, pp. 1665–1668, May 2011.
- [60] C. Zhang and K. Parhi, “Low-latency sequential and overlapped architectures for successive cancellation polar decoder,” *IEEE Trans. Signal Process.*, vol. 61, pp. 2429–2441, May 2013.

- [61] A. Pamuk, “An FPGA implementation architecture for decoding of polar codes,” in *Proc. 8th Int. Symp. Wireless Commun. (ISWCS)*, pp. 437–441, 2011.
- [62] C. Zhang and K. Parhi, “Interleaved successive cancellation polar decoders,” in *Proc. IEEE Int. Symp. Circuits and Syst. (ISCAS)*, pp. 401–404, June 2014.
- [63] P. Giard, G. Sarkis, C. Thibeault, and W. J. Gross, “Multi-mode unrolled architectures for polar decoders,” *IEEE Trans. Circuits and Syst. I, Regular Papers*, vol. 63, pp. 1443–1453, Sept 2016.
- [64] G. Schnabl and M. Bossert, “Soft-decision decoding of Reed-Muller codes as generalized multiple concatenated codes,” *IEEE Trans. Inform. Theory*, vol. 41, pp. 304–308, Jan. 1995.
- [65] I. Dumer and K. Shabunov, “Recursive decoding of Reed-Muller codes,” in *Proc. IEEE Int. Symp. Inform. Theory (ISIT)*, pp. 63–, 2000.
- [66] A. Alamdar-Yazdi and F. Kschischang, “A simplified successive-cancellation decoder for polar codes,” *IEEE Commun. Lett.*, vol. 15, pp. 1378–1380, Dec. 2011.
- [67] G. Sarkis, P. Giard, A. Vardy, C. Thibeault, and W. Gross, “Fast polar decoders: algorithm and implementation,” *IEEE J. Sel. Areas Commun.*, vol. 32, pp. 946–957, May 2014.
- [68] B. Yuan and K. Parhi, “Architectures for polar BP decoders using folding,” in *IEEE Int. Symp. Circuits Syst. (ISCAS)*, pp. 205–208, June 2014.
- [69] Y. S. Park, Y. Tao, S. Sun, and Z. Zhang, “A 4.68gb/s belief propagation polar decoder with bit-splitting register file,” in *Symp. VLSI Circuits Dig. of Tech. Papers*, pp. 1–2, June 2014.
- [70] S. M. Abbas, Y. Fan, J. Chen, and C. Y. Tsui, “High-throughput and energy-efficient belief propagation polar code decoder,” *IEEE Trans. VLSI Syst.*, vol. 25, pp. 1098–1111, March 2017.

- [71] J. Lin, C. Xiong, and Z. Yan, "A high throughput list decoder architecture for polar codes," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, pp. 2378–2391, June 2016.
- [72] C. Xiong, J. Lin, and Z. Yan, "A multimode area-efficient scl polar decoder," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, pp. 3499–3512, Dec 2016.
- [73] Y. Fan, J. Chen, C. Xia, C. y. Tsui, J. Jin, H. Shen, and B. Li, "Low-latency list decoding of polar codes with double thresholding," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1042–1046, April 2015.
- [74] B. Yuan and K. K. Parhi, "LLR-based successive-cancellation list decoder for polar codes with multibit decision," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 64, pp. 21–25, Jan 2017.
- [75] 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.
- [76] B. Yuan and K. Parhi, "Low-latency successive-cancellation list decoders for polar codes with multibit decision," *IEEE Trans. VLSI Syst.*, vol. 23, pp. 2268–2280, Oct. 2015.
- [77] A. Balatsoukas-Stimming, P. Giard, and A. Burg, "Comparison of polar decoders with existing low-density parity-check and turbo decoders," *CoRR*, vol. abs/1702.04707, 2017.
- [78] A. Raymond and W. Gross, "A scalable successive-cancellation decoder for polar codes," *IEEE Trans. Signal Process.*, vol. 62, pp. 5339–5347, Oct. 2014.
- [79] N. Weste and D. Harris, *Integrated Circuit Design*. Pearson, 2011.
- [80] S.-W. Yen, S.-Y. Hung, C.-L. Chen, Chang, Hsie-Chia, S.-J. Jou, and C.-Y. Lee, "A 5.79-Gb/s energy-efficient multirate LDPC codec chip for IEEE 802.15.3c applications," *IEEE J. Solid-State Circuits*, vol. 47, pp. 2246–2257, Sep. 2012.

- [81] Y. S. Park, *Energy-Efficient Decoders of Near-Capacity Channel Codes*. PhD thesis, Univ. of Michigan, Ann Arbor, 2014.
- [82] A. Pamuk and E. Arıkan, “A two phase successive cancellation decoder architecture for polar codes,” in *Proc. IEEE Int. Symp. Inform. Theory (ISIT)*, pp. 957–961, July 2013.
- [83] P. H. J. Bertram and M. Huber, “An improved majority-logic decoder offering massively parallel decoding for real-time control in embedded systems,” *IEEE Trans. Commun.*, vol. 61, p. 4808–4815, Dec. 2013.
- [84] H. Vangala, E. Viterbo, and Y. Hong, “A comparative study of polar code constructions for the AWGN channel,” *CoRR*, vol. abs/1501.02473, 2015.
- [85] O. Dizdar and E. Arıkan, “A high-throughput energy-efficient implementation of successive cancellation decoder for polar codes using combinational logic,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, pp. 436–447, March 2016.
- [86] T. Cui, Q. Xie, Y. Wang, S. Nazarian, and M. Pedram, “7nm FinFET standard cell layout characterization and power density prediction in near- and super-threshold voltage regimes,” in *International Green Computing Conference*, pp. 1–7, Nov 2014.