KARADENİZ TECHNICAL UNIVERSITY
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

ELECTRICAL AND ELECTRONICS ENGINEERING GRADUATE PROGRAM

LOW COMPLEXITY EARLY STOPPING STRUCTURE FOR BELIEF PROPAGATION DECODER

Ph.D. THESIS

Cemaleddin ŞİMŞEK, M.Sc.E.

JULY 2017
TRABZON

KARADENİZ TECHNICAL UNIVERSITY
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

ELECTRICAL AND ELECTRONICS ENGINEERING GRADUATE PROGRAM

LOW COMPLEXITY EARLY STOPPING STRUCTURE FOR BELIEF PROPAGATION DECODER

M. Sc. E. Cemaleddin ŞİMŞEK

This thesis is accepted to give the degree of
DOCTOR OF PHILOSOPHY
By
The Graduate School of Natural and Applied Sciences at
Karadeniz Technical University

The Date of Submission : 30 / 05 /2017
The Date of Examination : 17 / 07 /2007

Thesis Supervisor : Assoc. Prof. Dr. Kadir TÜRK

Trabzon 2017

KARADENİZ TECHNICAL UNIVERSITY
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

ELECTRICAL AND ELECTRONICS ENGINEERING GRADUATE PROGRAM
Cemaleddin ŞİMŞEK M.Sc.E.

LOW COMPLEXITY EARLY STOPPING STRUCTURE FOR BELIEF PROPAGATION DECODER

Has been accepted as a thesis of

DOCTOR OF PHILOSOPHY

after the Examination by the Jury Assigned by the Administrative Board of the Graduate School of Natural and Applied Sciences with the Decision Number 1707 dated 20 / 06 / 2017

Approved By

Chairman	:	Prof. Dr. İbrahim DEVELİ	....
Member	:	Assoc. Prof. Dr. Birol SOYSAL	....
Member	:	Assoc. Prof. Dr. Kadir TÜRK	....
Member	:	Assoc. Prof. Dr. Salim KAHVECİ	....
Member	:	Assoc. Prof. Dr. Mustafa ULUTAŞ	M. Ulutaş

Prof. Dr. Sadettin KORKMAZ
Director of Graduate School

FOREWORD

In today's world even the smallest contribution to sum of mankind's knowledge can have enormous impact on many things. However, statistically only one out of every five hundred Ph.D. thesis could find an useful application. Our motive is to make a contribution to sum of our knowledge which can actually solve or ease a problem. We believe, we have achieved our goal with this thesis regardless of its scale. Hopefully this study will pave the way for better ones and contribute to others.

I would like to offer my dearest gratitude and respect to my supervisor Assoc. Prof. Kadir TÜRK who endorsed and supported me with his vast knowledge, wisdom and patience throughout the journey.

I also want to thank my colleagues and friends Resch. Asst. Cenk ALBAYRAK, Asst. Prof. Emin TUĞCU and Asst. Prof. Ayhan YAZGAN who actually made contributions both to me and this thesis.

The last but not the least I thank to my wife, children and my all family, especially Ahmet UZUNDEDE for their spiritual guidance.

Cemaleddin ŞİMŞEK
Trabzon 2017

THESIS STATEMENT

I declare that, this Ph.D. thesis, I have submitted with the title "Low Complexity Early Stopping Structure For Belief Propagation Decoder" has been completed under the guidance of my Ph.D. supervisor Assoc. Prof. Dr. Kadir Türk. All the data used in this thesis are obtained by simulation and experimental works done as parts of this work in our research labs. All referred information used in the thesis has been indicated in the text and cited in reference list. I have obeyed all research and ethical rules during my research and I accept all responsibility if proven otherwise. 17/07/2017

Cemaleddin ŞİMŞEK

TABLE OF CONTENTS

Doktora Tezi

ÖZET

KANI YAYILIMI KOD ÇÖZÜCÜ İÇİN DÜŞÜK KARMAŞIKLIKLI ERKEN DURDURMA YAPISI

Cemaleddin ŞİMŞEK

Karadeniz Teknik Üniversitesi
Fen Bilimleri Enstitüsü
Elektrik-Elektronik Mühendisliği Anabilim Dalı
Danışman: Doç. Dr. Kadir TÜRK
2017, 100 Sayfa

Kanı yayılımı kod çözücü birçok hata düzeltme kod ailesinde kullanılan yinelemeli bir kod çözücüdür. Erken durdurma yöntemi olmayan yinelemeli bir kod çözücü, kod çözme işlemi için sabit sayıda yineleme yapar. Ancak kod çözme işlemi, yinelemeler bu sabit sayıya ulaşmadan önce tamamlanmış olabilir. Bu durumda yinelemelere devam eden kod çözücü gereksiz işlem yapmış olur. Bundan dolayı yeri geldiğinde yinelemeleri durdurmak işlem yükünü düşük tutabilmek için elzemdir.

Bu amaçla bu tez çalışmasıda kanı yayılımı kod çözücü için düşük karmaşıklıklı bir erken durdurma yapısı önerilmiştir. Literatürdeki diğer yineleme erken durdurma yöntemlerinin aksine, önerilen yöntem logaritmik olasılık oranları (LLR) mesajlarının sadece küçük bir miktarını kullanır ve bu mesajların sadece işaret bitlerini gözlemler.

Önerilen yineleme erken durdurma yapısı hem kutup hem de Luby dönüşümü (LT) kodlara uygulanmıştır ve kanı yayılımı kod çözücü kullanan tüm hata düzeltme kodlarına da kolayca uygulanabilir. Perfomans parametreleri, hem benzetim çalışmaları hem de donanım tanımlama dili (VHDL) uygulamaları ile kıyaslanmıştır. Sonuçlar, önerilen yöntemin literatürdeki diğer yöntemlere nazaran işlem yükü ve donanımsal ihtiyaçları azaltmasının yanında, veri hacmini de arttırdığını göstermektedir.

Anahtar Kelimeler: Hata düzeltme kodu, Kapasite başarımlı kodlar, Kutup kod, Luby dönüşüm kod, Yinelemeli kod çözücü, Kanı yayılımı kod çözücü, Erken durdurma kriteri, Erken sonlandırma metodu, Donanm tanımlama dilinde donanım tasarımı, Donanım en iyilemesi

Ph.D. Thesis

SUMMARY

LOW COMPLEXITY EARLY STOPPING STRUCTURE FOR BELIEF PROPAGATION DECODER

Cemaleddin ŞİMŞEK

Karadeniz Technical University
The Graduate School of Natural and Applied Sciences
Department of Electrical and Electronics Engineering Graduate Program
Supervisor: Assoc. Prof. Dr. Kadir TÜRK
2017, 100 Pages

Belief propagation (BP) decoder is a well known iterative decoder used for decoding many error correction code families. BP decoder without early stopping structure uses a fixed iteration number to end iterative decoding process. But decoder may be converged before iteration number reaches this fixed limit. In these case, decoder performs a redundant process. Therefore, stopping the iterations is essential to keep computational burden as low as possible when decoding is successful.

With this perspective, a low complexity early stopping structure for belief propagation decoders is proposed with this thesis. In contrast to previous early stopping methods in literature, proposed early stopping structure only uses small amount of log-likelihood ratio (LLR) messages and tracks only sign alterations of them.

Proposed structure is applied to both polar and Luby transform (LT) codes and can be easily applied to error correction codes use BP as decoder. Performance parameters are compared with simulation works and hardware description language (VHDL) implementations. Results illustrate that proposed approach significantly reduces the computational complexity and required hardware resources, also throughput values are increased compared to previous counterparts in literature.

Key Words: Error correction code, Capacity achieving codes, Polar code, Luby transform code, Iterative decoder, Belief propagation decoder, Early stopping criteria, Early termination method, VHDL hardware design, Hardware optimization

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ASIC	:	Application specific integrated circuit
BEC	:	Binary erasure channel
BER	:	Bit error rate
BI-AWGNC	:	Binary input additive white Gaussian noise channel
BLER	:	Block error rate
BP	:	Belief propagation
BSC	:	Binary symmetric channel
B-DMC	:	Binary input discrete memoryless channel
C	:	Capacity
CN($\oplus$)	:	Check node
CRA	:	Carry ripple adder
CRC	:	Cyclic redundancy check
CSR	:	Check-sum satisfaction ratio
C2S	:	Complement to signed
d	:	Degree
dB	:	Decibel
DE	:	Density evolution
DC-LRM	:	Determination condition of LRM
$E[\,.\,]$	:	Expectation value
E_b/N_0	:	Energy per Bit to the Spectral Noise Density
ECC	:	Error correction code
ESC	:	Early stopping criterion
Eqn.	:	Equation
exp	:	Exponential function
FER	:	Frame error rate
FPGA	:	Field programmable gate array
GA	:	Gaussian approximation
Gbps	:	Giga bit per second
GHz	:	Gigahertz
GND	:	Ground
G-Matrix	:	Generator matrix
H	:	Entropy
H	:	Parity check matrix
HDL	:	Hardware description language
I	:	Symmetric capacity

ISE	:	Integrated software environment
IBUF	:	Input buffer
K	:	Information bits length
LDPC	:	Low density parity check code
LLR	:	Logarithmic likelihood ratio
L_{LLR}	:	LLR propagating leftwards
LoL	:	Level of logic
LR(λ)	:	Likelihood ratio
LRM	:	Least reliable messages
LT	:	Luby transform
LUT	:	Look up table
M	:	Amount of consecutive iterations for WIB
mag	:	Magnitude
mC2S	:	Modified complement to signed
minLLR	:	Minimum LLR
ML	:	Maximum likelihood
ms	:	Millisecond
MS	:	Minimum and summation
mS2C	:	Modified signed to complement
MUX	:	Multiplexer
N	:	Code length
N_B	:	Amount of LRM
ns	:	Nanosecond
nm	:	Nanometer
n_{WIB}	:	Amount of WIB
$\mathcal{O}$	:	Asymptote notation
OBUF	:	Output buffer
P	:	Probability
PDF	:	Probability density function
PE	:	Processing element
PoB	:	Proportion of Bhattacharyya values
R	:	Code rate
R_{LLR}	:	LLR propagating rightwards
$randn$	:	Noise with random distribution
RCM	:	Randomly chosen messages
SC	:	Successive cancellation
SCH	:	Successive cancellation hybrid

SCL	:	Successive cancellation list
SCS	:	Successive cancellation stack
$sign$	:	Signum function
SMS	:	Scaled minimum and summation
SNR	:	Signal to noise ratio
S2C	:	Signed to complement
T_{adder}	:	Delay of adder circuit
$tanh$	:	Hyperbolic tangent function
TPM	:	Transition probability matrix
VHDL	:	Very high speed integrated circuit hardware description language
VN($\boxed{\equiv}$)	:	Variable node
W	:	Transition probability of B-DMC
WIB	:	Worst of information bits
XOR	:	Exclusive OR
Z	:	Bhattacharyya parameter
$\Omega(.)$	:	Degree distribution
Γ_{LC}	:	Amount of consecutive iterations
σ^2	:	Variance of Gaussian noise
β	:	Predetermined constant

1. INTRODUCTION

Ever since Shannon's [1] study pointed out the mathematical approach to communication systems, researchers are observing the limits of communication. Starting from 1948 theory of error free communication has become the elusive goal for information theorists. Many milestones have been passed especially by Shannon's coworkers, students and their students as well. First attempt was made by R. Hamming in 1950 with the name of Hamming Code [2] which is a member of linear error correction codes. Studies are followed by I.S. Reed, D. E. Muller and G. Solomon between 1954 and 1960 named Reed-Muller and Reed-Solomon [3, 4] codes. Another major breakthrough is achieved by R. G. Gallager who proposed low density parity check codes (LDPC) [5] in 1962 which was unused for nearly three decades due to technical limitations. C. Berrou, A. Glavieux and P. Thitimajshima introduced turbo codes in 1993 [6]. Another important error correction code family, fountain codes, was introduced with the name of Luby transform codes by M. Luby in 2002 [7] which evolved as raptor code later by A. Shokrollahi in 2006 [8]. Finally polar code which is the first theoretically proven capacity achieving error correction code (ECC)[9] was introduced in 2008 by E. Arikan who was a student of R.G. Gallager. Obviously this summary does not cover the whole story but we may say that these are the major milestones for channel coding in information theory field (see Fig. 1.1).

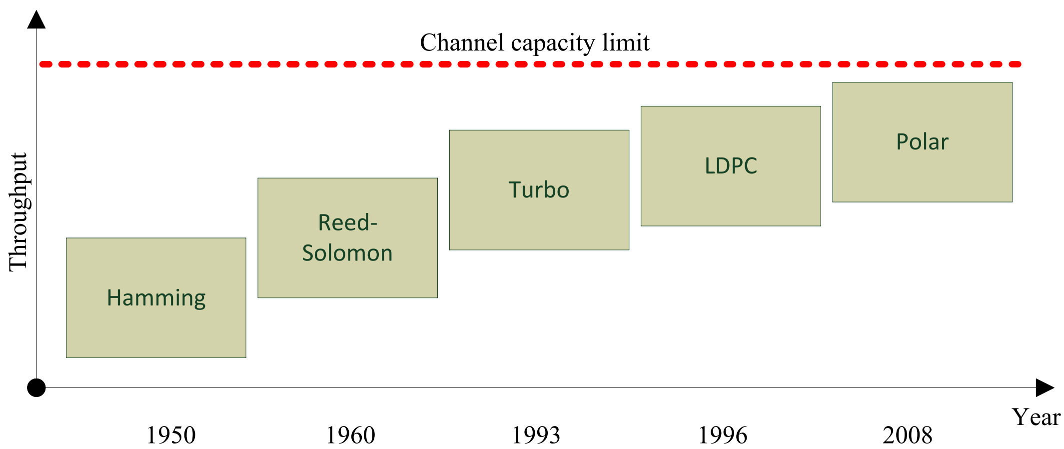


Figure 1.1. Throughput of error correction codes vs years [10]

With the increasing demand on communication speed and multimedia application requirements as the Moore's law [11] predicted increasing number of components per integrated circuit helps to overcome some of the technical limitations so far. Fortunately, users do not have to wait another three decades to see polar code on work. As a matter of fact polar code is one of the most powerful candidates to be used in fifth generation (5G) communication systems [12]. This situation brings some concerns out to surface such as applicability and feasibility of polar code for a real time communication system which is one of the main focuses of this thesis. Theoretically achieving the capacity is not the only requirement for real world practical applications. Technical limitations become the next important problem as it was with LDPC. According to application type, three important parameters taken into consideration to design a communication system [13]. These are; bandwidth, power and cost efficiencies. As an important component of a communication system, ECC's encoding and decoding complexities and practical applicability are directly linked to all three of these parameters. While capacity achieving property of polar code mostly deals with bandwidth efficiency researchers are focused on other topics. Designing power and cost efficient encoders and decoders for polar code have been widely investigated and still requiring further investigation. This thesis focused on reducing the complexity of polar code and Luby transform (LT) decoder.

Thesis is outlined as follows: fundamentals of error correction codes, encoding and decoding strategies of polar and LT codes followed with contributions to literature and finally we draw some conclusion.

2. PAST-WORKS

2.1. Fundamentals of Error Correction Codes

First approach to the communication theory as a statistical and probabilistic problem was made by C. Shannon [1]. In basics, Shannon's theory defines how to organize the information in order to withstand the disruptive effect of communication channel with a specific power, bandwidth and time. With the study named "A Mathematical Theory of Communication" Shannon developed information entropy as a measure for the uncertainty in a message while essentially inventing the field of information theory.

In a communication system with M different messages $(m_1, m_2, ...)$ with probabilities of occurrence $(p_1, p_2, ...)$ information amount carried with message m_k which has p_k probability can be expressed as in Eqn. (2.1).

$$I_k = log_2(\frac{1}{p_k}) \tag{2.1}$$

Unit of information is defined as bits and as can be seen with equation:

- Information amount increases when uncertainty of message increases,
- If the message is known by receiver $(p_k = 1)$ message does not carry any information,
- If there are $M = 2^N$ equal probable messages, information carried with each message equals to N bits.

Total information carried with independent messages is equal to summation of all messages information. If there are M messages with length L, total information for message m_k with probability p_k is defined as in Eqn. (2.2).

$$I_{k(total)} = p_k * L * log_2(\frac{1}{p_k}) \tag{2.2}$$

Total information amount can be expressed as in Eqn. (2.3).

$$\begin{aligned} I_{(total)} &= I_{1(total)} + I_{2(total)} + ... + I_{M(total)} \\ &= p_1 * L * log_2(\frac{1}{p_1}) + p_2 * L * log_2(\frac{1}{p_2}) + ... + p_M * L * log_2(\frac{1}{p_M}) \end{aligned} \tag{2.3}$$

This leads us to definition of Entropy which is average information carried per message Eqn. (2.4) with unit $bit/message$.

$$Entropy(H) = \frac{I_{(total)}}{L} = \sum_{k=1}^{M} p_k * log_2(\frac{1}{p_k}) \tag{2.4}$$

- As we can see Entropy is minimum $(H = 0)$ when messages are known $(pk = 1 \ or \ pk = 0)$.
- Entropy becomes maximum when all messages are equal probable. $(H = log_2 M)$

For instance; if we consider two messages with probabilities $(p \ and \ 1 - p)$ entropy is equal to Eqn. (2.5).

$$Entropy(H) = \sum_{k=1}^{M} p_k * log_2(\frac{1}{p_k}) = p * log_2(\frac{1}{p}) + (1-p) * log_2(\frac{1}{1-p}) \tag{2.5}$$

If we calculate the value of H according to p we get Fig. 2.1.

2.2. Communication Channels

A communication system basically consist of three elements; transmitter, receiver and channel (see Fig. 2.2). Here we only deal with discrete memoryless communication channels.

2.2.1. Discrete Communication Channels

- If a channel has input X and output Y which are discrete random variables, channel is called discrete channel.
- If current output is independent from previous inputs, channel is called memoryless channel.
- This channel is described with input and output alphabet and their conditional transition probabilities as in Eqn. (2.6). If output is y_j when input is x_i, conditional transition probability is represented as $p(y_j/x_i)$ and all probable values form the transition probabilities or channel matrix Eqn. (2.6).

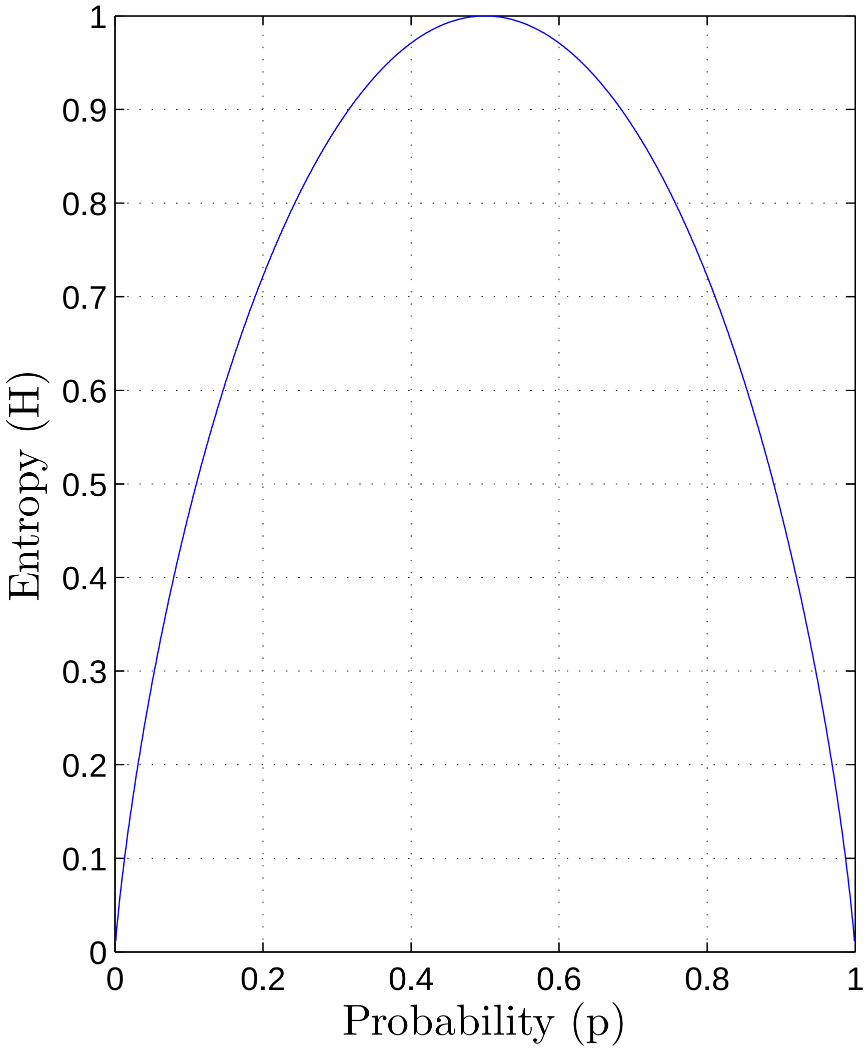


Figure 2.1. Entropy vs probability

$$\mathbf{P} = P(y_j/x_i) = \begin{bmatrix} P(y_1/x_1) & P(y_2/x_1) & \dots & P(y_M/x_1) \\ P(y_1/x_2) & P(y_2/x_2) & \dots & P(y_M/x_2) \\ \vdots & \vdots & & \vdots \\ P(y_1/x_N) & P(y_2/x_N) & \dots & P(y_M/x_N) \end{bmatrix} \tag{2.6}$$

For a specific input (an entire row) in Eqn. (2.6) summation of all values in that particular row equals to “1” i.e., $P(y_1/x_1) + P(y_2/x_1) + ...P(y_M/x_1) = 1 \quad \Rightarrow \quad \sum_{j=1}^{M} P(y_j/x_i) = 1$. For observing the possibility of an output we need to deal with joint probability of all possible inputs and that output. Joint probability of x_i and y_j is calculated as $P(x_i, y_j) = P(y_j/x_i) * P(x_i)$. To be able to calculate probability of an output y_j all of the conditional probabilities need to be added. With help of $\sum_{i=1}^{N} P(x_i, y_j) = \sum_{i=1}^{N} P(y_j/x_i) * P(x_i) = P(y_i)$ one can calculate the

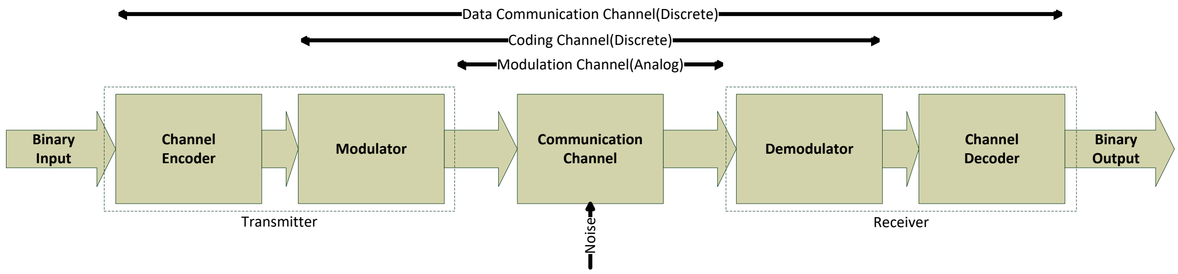


Figure 2.2. Communication channel

error probability as in Eqn. (2.7).

$$P_e = \sum_{\substack{j=1 \\ j \neq i}}^{M} P(y_j) = \sum_{\substack{j=1 \\ j \neq i}}^{M} \sum_{i=1}^{N} P(y_j/x_i) P(x_i) \tag{2.7}$$

Probability of correct reception will be $P_c = 1 - P_e$.

2.2.2. Binary Communication Channels

A channel is called binary channel if there are only two symbols for transmission (see Fig. 2.3). Probability transition matrix given in Eqn. (2.8).

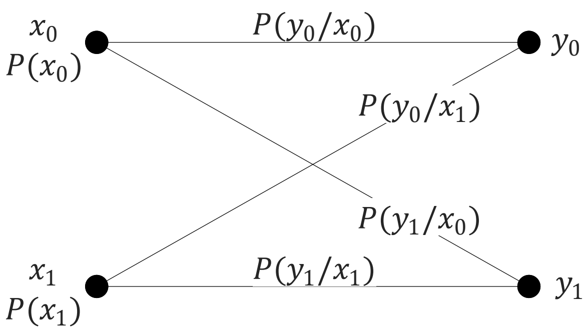


Figure 2.3. Binary communication channel

$$\begin{bmatrix} P(y_0) \\ P(y_1) \end{bmatrix} = \begin{bmatrix} P(x_0) & P(x_1) \end{bmatrix} \times \begin{bmatrix} P(y_0/x_0) & P(y_1/x_0) \\ P(y_0/x_1) & P(y_1/x_1) \end{bmatrix} \tag{2.8}$$

2.2.2.1. Binary Symmetric Channel

Channel is called symmetric if probabilities in Eqn. (2.8) are equal as $P(y_0/x_0) = P(y_1/x_1) = p$ and probability transition matrix can be written as Eqn. (2.9).

$$\begin{bmatrix} P(y_0) \\ P(y_1) \end{bmatrix} = \begin{bmatrix} P(x_0) & P(x_1) \end{bmatrix} \times \begin{bmatrix} p & 1-p \\ 1-p & p \end{bmatrix} \tag{2.9}$$

2.2.2.2. Binary Erasure Channel

Channel is called binary erasure channel if there are two input and three output as in Fig. 2.4. Third output means symbol is erased or lost.

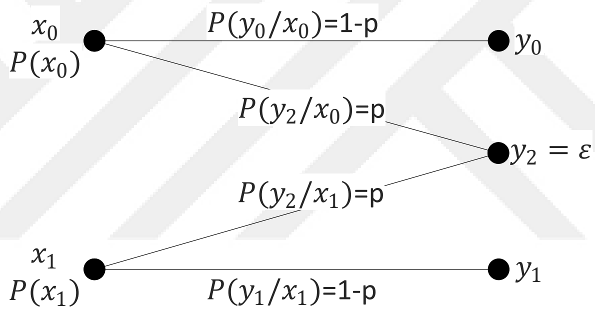


Figure 2.4. Binary erasure channel

Probability transition matrix becomes as in Eqn. (2.10).

$$\begin{bmatrix} P(y_0) \\ P(y_2) \\ P(y_1) \end{bmatrix} = \begin{bmatrix} P(x_0) & P(x_1) \end{bmatrix} \times \begin{bmatrix} 1-p & p & 0 \\ 0 & p & 1-p \end{bmatrix} \tag{2.10}$$

2.2.3. Conditional and Joint Entropy

To be able to calculate capacity of a discrete memoryless channel one need to understand conditional and joint entropy concepts. Conditional entropy Eqn. (2.11) is also called equivocation represented by $H(X/Y)$ which gives the uncertainty of X when Y is received or $H(Y/X)$ which gives the uncertainty of Y when X is transmitted. Simply it represents information loss trough noisy channel.

$$H(X/Y) = \sum_{i=1}^{N}\sum_{j=1}^{M} P(x_i, y_j) log_2 \frac{1}{P(x_i/y_j)} \tag{2.11}$$

Joint entropy is defined in Eqn. (2.12) and relation between conditional entropy is defined in Eqn. (2.13).

$$H(X,Y) = \sum_{i=1}^{N}\sum_{j=1}^{M} P(x_i, y_j) log_2 \frac{1}{P(x_i, y_j)} \tag{2.12}$$

$$H(X,Y) = H(X/Y) + H(Y) = H(Y/X) + H(X) \tag{2.13}$$

2.2.4. Mutual Information

Transferred information amount when x_i is transmitted and y_i is received defined as mutual information Eqn. (2.14) with unit $bits$. Average mutual information is given in Eqn. (2.15) with unit $bits/symbol$.

$$I(x_i, y_i) = log_2 \frac{P(x_i/y_i)}{P(x_i)} \tag{2.14}$$

$$I(X;Y) = \sum_{i=1}^{N}\sum_{j=1}^{M} P(x_i, y_i) log_2 \frac{P(x_i/y_i)}{P(x_i)} \tag{2.15}$$

- Mutual information is symmetric $I(X;Y) = I(Y;X)$.
- Mutual information can be expressed related to entropies $I(X;Y) = H(X) - H(X/Y) = H(Y) - H(Y/X)$ and it has always positive value $I(X;Y) \geq 0$.
- Mutual information can be expressed related to joint entropy $I(X;Y) = H(X) + H(Y) - H(X,Y)$.

2.3. A Brief Explanation of Belief Propagation Algorithm

In 1982 Judea Pearl offered a message passing algorithm called belief propagation (BP) [14] which is designed to perform inference on graphical models (Bayesian Networks). First introduction of Pearl's BP algorithm to information theory field is made in [15] with Turbo and LDPC codes. This approach made decoding section of Turbo and LDPC codes much less complex and more feasible, especially with sum-product algorithm [16].

Working principle of BP is to send messages along a factor (Tanner) graph which is a bipartite graph representing the factorization of a function. In information theory field this function is generally probability distribution function and propagated messages are log-likelihood ratio (LLR) values received from channel.

A factor graph includes two types of nodes called check (function) and variable (bit) nodes. Tanner graph of ECCs are produced from encoding or parity check matrices. As an example for Tanner graph a (N,K) LDPC parity check matrix and its graph is given below. Ones inside the matrix H represent the connections between check and variable nodes (see Fig. 2.5).

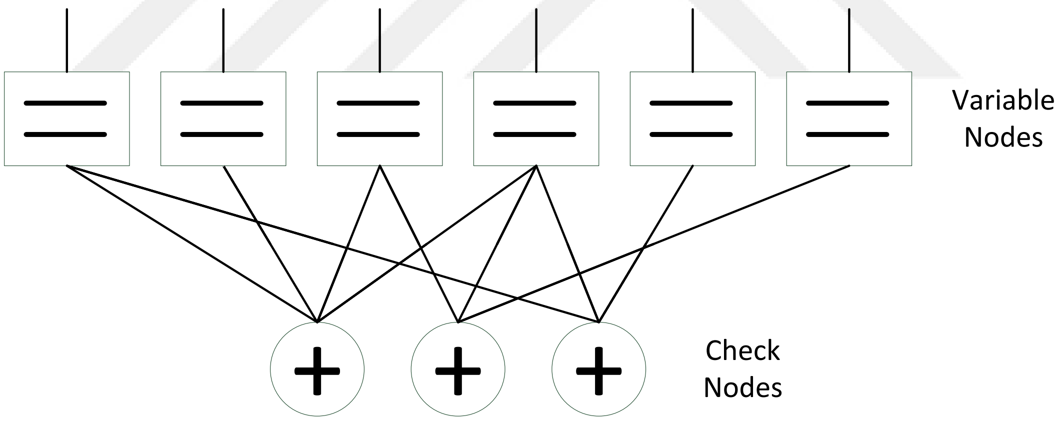


Figure 2.5. $(6,3)$ Tanner graph

For LDPC codes as in this particular example, the check nodes denote rows of the parity-check matrix H Eqn. (2.16). The variable nodes represent the columns of the matrix H Eqn. (2.16).

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \tag{2.16}$$

Update equations for LLR values propagates back and forth between variable and check nodes are given in Eqn. (2.17) along with Fig. 2.6.

$$\begin{aligned} m^{(v)} &= m_0 + \sum_{k=1}^{d_v-1} m_k^{(c)} \\ m^{(c)} &= \prod_k sign(m_k^{(v)}) \cdot \phi\left(\sum_{k=1}^{d_c-1} \phi\left(mag(m_k^{(v)})\right)\right) \end{aligned} \tag{2.17}$$

Here $\phi(x) = -logtanh(x/2)$ and m_0 is LLR from channel.

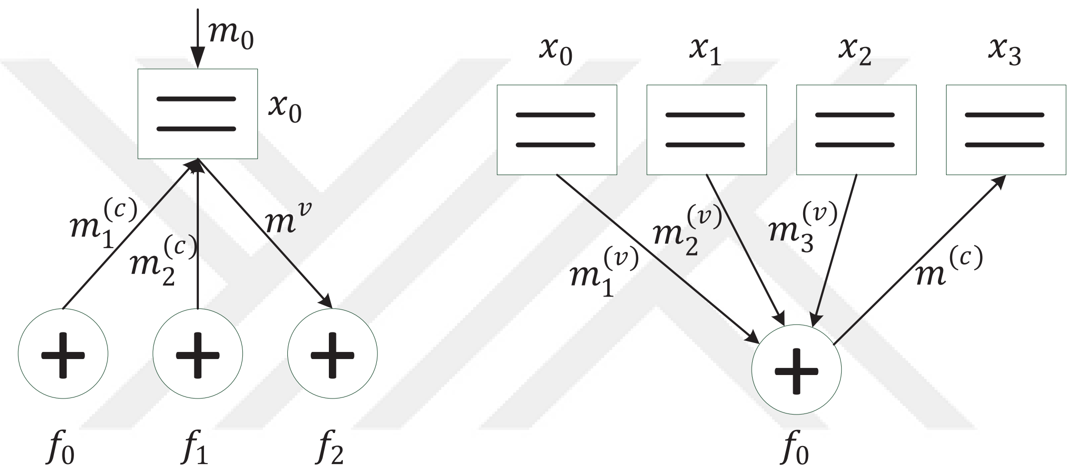


Figure 2.6. Check and variable node update scheme.

For an iterative process such as decoding LDPC codes with BP algorithm there needs to be a limit for iterations or a condition to end it. This limit means a fixed number of iterations which means wasting resources most of the time. The condition for terminating iterations can be observed relatively easy for ECCs which has a parity check structure. If parity check condition is satisfied as in Eqn. (2.18), iteration process can be terminated. However, there is no parity check structure for some ECCs such as Polar and LT codes, so some other and efficient methods needed which we tried to focus on in this thesis.

$$\begin{aligned} x &= u \otimes G \\ H \otimes G^T &= 0 \\ H \otimes x &= 0 \end{aligned} \tag{2.18}$$

2.4. Polar Code

Polar code [9] is a linear block error correction code (ECC). It is also the first deterministic construction of capacity-achieving (symmetric capacity $I(W)$) codes [17] for binary-input discrete memoryless channel (B-DMC)(denoted as W) with low encoding and decoding complexities. If code length is considered as N, both encoding and decoding complexities are $\mathscr{O}(Nlog_2N)$.

An ECC simply adds some redundancy to data in order to protect it from disruptive effects of communication channel. For polar code, addition of this redundancy based on polarization phenomena [9]. Arikan's proposition proves that code sequence constructed by channel polarization achieves the symmetric capacity $I(W)$. With polarization effect, combination of N independent copies of W $\{W_N^{(i)} : 1 \leq i \leq N\}$ construct a code sequence that $I(W_N^{(i)})$ is near "1" become closer to symmetric capacity while $I(W_N^{(i)})$ near "0" become closer to zero capacity $1 - I(W)$. Channels with near symmetric capacity used for transmission while rest is filled with known data. These channels are called information and frozen bits, respectively. We might consider the copies of W as virtual channels or bit channels as they are only used to prove the polarization effect.

2.4.1. Polarization Phenomena

Consider a B-DMC $W : X \rightarrow Y$ which input alphabet $X = \{1,0\}$ and output alphabet Y with transition probabilities $W(y|x)$, $x \in X$ $y \in Y$ as in Fig. 2.7.

Figure 2.7. B-DMC with input X and output Y.

For a symmetric channel the capacity is $0 \leq C(W) \triangleq I(X;Y) \leq 1$ where X is uniform over $\{0,1\}$. As one can see $C(W) = 0$ is an useless channel where $C(W) = 1$ is the perfect channel. In order to achieve such channels, channel combination should be made as in Fig. 2.8 Combination of two W creates two bit-channels with conditional transition probabilities as in

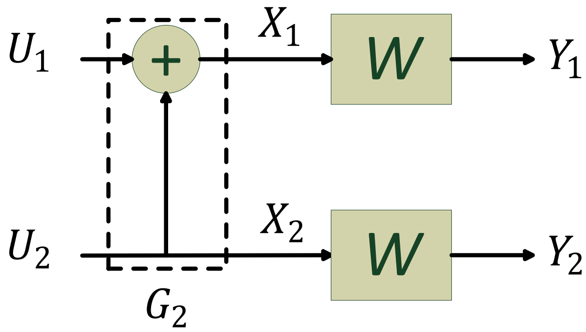


Figure 2.8. Combined two copies of W.

Eqn. (2.19). Capacity of two W becomes as in Eqn. (2.20).

$$\begin{aligned} W_N^{(i)}(y_1^N, u_1^{i-1}|u_i) &= \sum_{u_{i+1}^N \in X^{N-i}} \frac{1}{2^{N-1}} W_N(y_1^N|u_1^N) \\ W_1 &: U_1 \rightarrow (Y_1, Y_2) \\ W_2 &: U_2 \rightarrow (Y_1, Y_2, U_1) \end{aligned} \tag{2.19}$$

$$\begin{aligned} C(W_1) &= I(U_1; Y_1, Y_2) \\ C(W_2) &= I(U_2; Y_1, Y_2, U_1) \end{aligned} \tag{2.20}$$

Total capacity of bit channels are preserved but distributed unevenly as in Eqn. (2.21). Later we denote W_1 and W_2 as W^- and W^+, respectively.

$$\begin{aligned} C(W_1) + C(W_2) &= 2C(W) \\ C(W_1) \leq C(W) &\leq C(W_2) \end{aligned} \tag{2.21}$$

To increase the size of construction and polarization effect one simply needs to use the Fig. 2.8 as base point and duplicate it until desired size is reached as in Fig. 2.9.

Generator matrix is based on kernel matrix G_2 in Fig. 2.8 which is given by Eqn. (2.22).

$$G_2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \tag{2.22}$$

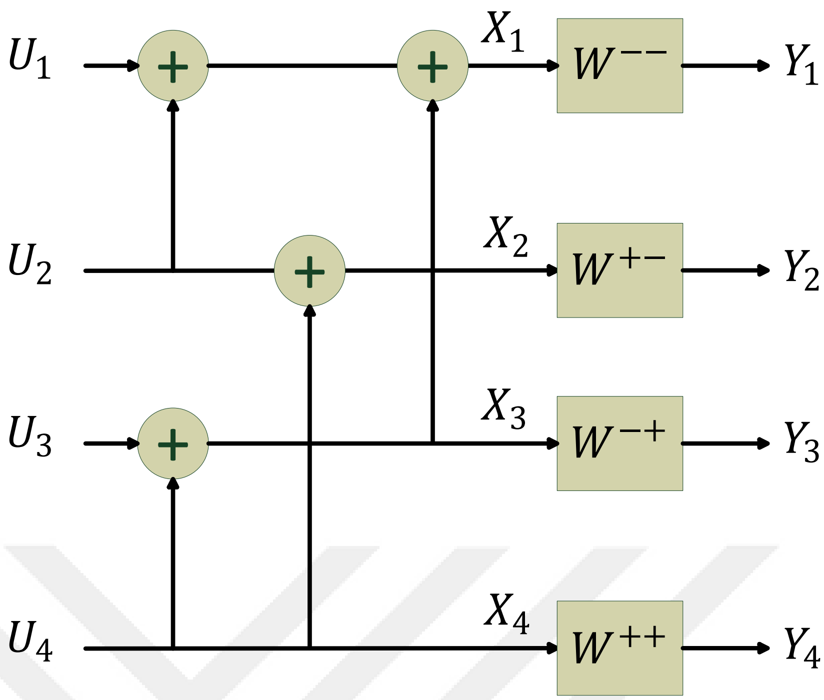


Figure 2.9. Combined four copies of W.

To increase the size one needs to take the Kronecker product $\left(G^{\otimes n}\right)$ of G_2 in order to produce the generator matrix for size $N * N \mid N = 2^n$. Kronecker product defined in Eqn. (2.23) as placing "0" matrix where a zero "0" value in base matrix and placing matrix itself where a value one "1" seen in base matrix (e.g. Eqn. (2.24)).

$$G_N = \begin{bmatrix} G_{N/2} & 0_{N/2} \\ G_{N/2} & G_{N/2} \end{bmatrix} \tag{2.23}$$

$$G_4 = G^{\otimes 2} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad G_8 = G^{\otimes 3} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{2.24}$$

The easiest way to test polarization effect is choosing W as BEC since there is no bit filliping possibility as seen in Eqn. (2.4). This brings the advantage of recursive capacity calculation simplicity for bit channels. Recursive calculations are made with Eqn. (2.25) and resulting $C(W_n^{(i)})$ are illustrated in Fig. 2.10 for $n = 0...8$ considering $C(W) = 0.5$.

$$\begin{aligned} \varepsilon^- &\triangleq 2\varepsilon - \varepsilon^2 \\ \varepsilon^+ &\triangleq \varepsilon^2 \end{aligned} \tag{2.25}$$

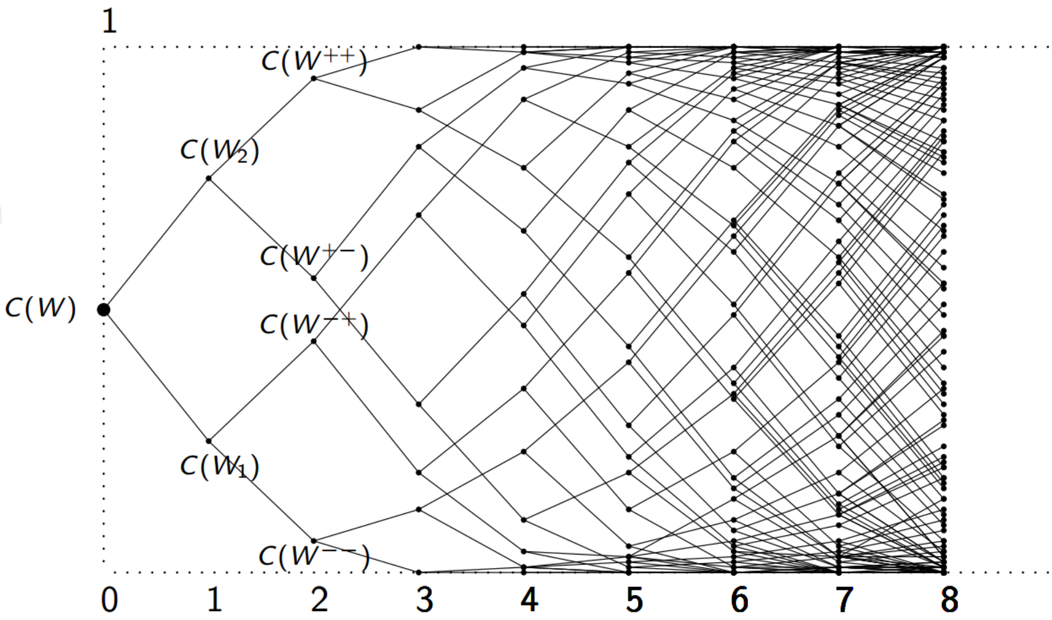
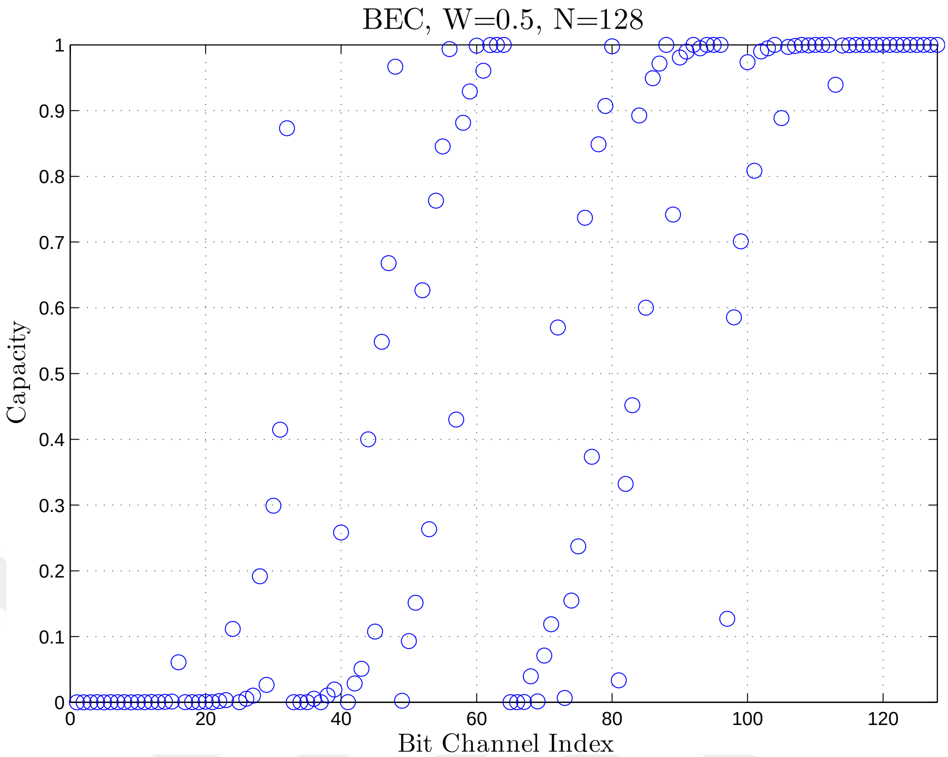


Figure 2.10. Bit channels capacity distribution vs $n = 0...8$.

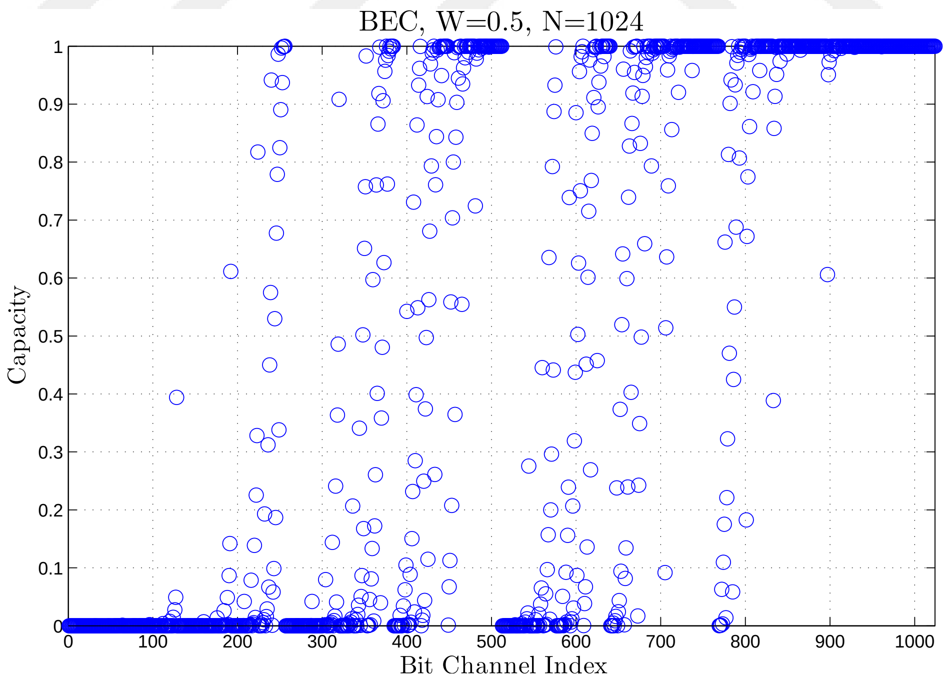
Fig. 2.11(a) and Fig. 2.11(b) is an illustration of polarized channel capacities for both $N = 128$ and $N = 1024$. As figures illustrated most of the bit channels capacities are polarized either "1" (perfect channel) or "0" (useless channel). This effect is more dominant when N goes to infinite [9].

Another important point is semi-polarized bit channels which their capacities remained unpolarized. As shown in Fig. 2.10, 2.11(a) and 2.11(b) there are some channels which are not polarized as bad or good. Some of these channels can be use to transfer information for instance if $N = 128$ and rate is chosen as $R = \frac{1}{2}$ there are 17 bit channels needed to be use as information bits with capacities between $(0.9 - 0.5)$. Most likely these channels are the ones will cause errors especially those capacities close to "0.5", since the error is upper bounded with error probabilities summation of information bits under successive cancellation (SC) decoder [9]. Semi-polarized channels are an important problem for short block polar codes.

This situation rises the questions that "which bit channels should be chosen and how?" to make polar code more efficient. The topic will be discussed in next Section 2.4.2 under Polar Code Construction.



(a)



(b)

Figure 2.11. Bit channels capacity distribution with N=128 and N=1024 for BEC with $C(W) = 0.5$.

2.4.2. Polar Code Construction

Constructing desired polar code basically means choosing right bit indexes for design signal to noise ratio (*SNR*) value. In generator matrix the columns which have more weights become the worst channels [18], so from application point of view determining the frozen and information bits seems rather easy. However, defining exact bit error rate (*BER*) or block error rate (*BLER*) is important for theoretical approach, error boundary calculations [19–21] and further improvements such as using other kernel matrix [22, 23].

Although, there exist some simplified and modified polar code construction methods, they all based on five methods which we are going to discuss in this section. Four of these methods are compared in [24] for binary input additive white Gaussian noise channel (BI-AWGNC). First one is proposed in original paper with polar code itself and called recursive Bhattacharyya bound [9] as mentioned in previous section. Instead of using BEC as initial channel, transition probability can be replaced for BI-AWGNC as proposed in [25]. Second method Monte-Carlo estimation also proposed by Arikan which is a simulation based method. Third method is proposed by Tal and Vardy which tries to estimate the transition probability matrix (TPM) for virtual channels in [26] to be able to calculate their capacities. Fourth method is proposed by Trifonov called Gaussian approximation in [25, 27] which only uses mean and standard deviation values of base Gaussian channel probability distribution function (PDF). Last method is similar with Gaussian approximation without simplifications called density evaluation [28, 29].

As concluded in [24] and confirmed by our simulation studies all methods can construct equally good polar code if proper design *SNR* is chosen.

2.4.2.1. Recursive Bhattacharyya Bound

As mentioned [24, 25] most codes are universal, meaning that their designs are independent from *SNR*. Polar code is different, its *BLER* and *BER* values are a function of *SNR* under SC decoder. However, this does not mean that a polar code design for a particular *SNR* will give the best results for that *SNR*. These topic is going to be demonstrated at the end of this section with performance comparisons of all methods as in [24].

Recursive construction of polar code is discussed in previous section as an example for polarization. Arikan proposed Bhattacharyya parameter $Z(W)$ as an upper bound of error probability for a particular W under maximum likelihood decoding strategy which is defined in Eqn. (2.26) and calculations are made in [9] only for BEC as in Eqn. (2.27). Here in Eqn. (2.27) $Z_N^{(i)}$ refers the Bhattacharyya parameter for $W_N^{(i)}$ where W is referred as $Z_1^{(1)}$

erasure probability. Recursive calculations of Bhattacharyya parameters for BI-AWGNC, replacing erasure probability with $exp(-RE_b/N_0)$ will be sufficient according to [30] and detailed algorithm is given in [24]. In [30] $Z(W_k^{(1)}) = e^{-SNR_k}$ is considered as Bhattacharyya parameter for kth use of BI-AWGNC and SNR_k is considered as its signal to noise ratio.

$$Z(W) = \sum_{y \in Y} \sqrt{p(y|0)p(y|1)} \tag{2.26}$$

$$\begin{aligned} Z_N^{(2j-1)} &= 2Z_{N/2}^{(j)} - (Z_{N/2}^{(j)})^2 \\ Z_N^{(2j)} &= (Z_{N/2}^{(j)})^2 \end{aligned} \tag{2.27}$$

2.4.2.2. Monte-Carlo Estimation

Monte-Carlo estimation for bit channels is a simulation based method. Originally, this method is proposed in [9] to calculate Bhattacharyya parameters. A modification is made in [24] to calculate BER of bit channels. Here we give a brief explanation of the modified method.

The modified method considers all-zero codeword transmission and all bits considered as frozen bits for each iterations under SC decoder. At the end of an iteration, bits are decided. At the end of all iterations, average BER of bit channels are calculated by averaging faulty detected bits. Each bit channel will have different BER according to polarization effect. As any iterative process this one also highly depended on iteration amount. If the iteration amount is not high enough, accuracy will be low to classify especially semi-polarized bit channels as frozen or information. Method can be summarized as follows:

- $y = -\sqrt{10^{(RE_b/N_0)/10}} + randn(N,1)$ is the input vector of SC decoder.
- $L = Pr(y_j|0)/Pr(y_j|1) = exp(-2y_j\sqrt{10^{RE_b/N_0/10}})$ $\forall j$ is likelihood ratio(LR) for each y of decoder input.
- LRs are updated according to SC decoder and output bits $\hat{u}(j)$ are decided and stored.
- Above steps are repeated for M amount of iterations.
- Results are normalized $\hat{u}(j)/M$ to find BER of jth bit channel.

As can be seen from above, complexity of Monte-Carlo method is much higher than recursive one. While recursive method has $\mathcal{O}(Nlog_2N)$ complexity, Monte-Carlo method has $\mathcal{O}(M \times Nlog_2N)$. Complexity increases with M to have better accuracy for bit channels BER.

2.4.2.3. Transition Probability Matrix Estimation

This method tries to estimate full transition probability matrix (TPM) of bit channels. BER of bit channels can be calculated once TPM is known [31]. If the output alphabet of size μ, TPM size become $2 \times 2\mu$. When structure size increase, bit channels output size will increase swiftly. Beacause of this limitation method tries to keep the output size at μ by quantizing it. This method has following steps:

- Quantization of BI-AWGNC for initial channel parameters with size μ.
- Convolutions of quantized bit channels according to generator matrix which will increase the size over μ.
- Reduction of output size to μ by quantizer.
- Determination of bit channels BER by choosing the minimum compared with recursive bound construction method.

Full TPM estimation has also high complexity compared to previous methods. Detailed explanation of this method with full pseudo code can be found in [21, 24, 26].

2.4.2.4. Gaussian Approximation

Gaussian approximation is a well known method as it has been used for various ECC such as LDPC [32]. It is a simplification of density evolution method [31]. Under the assumption of channel having symmetric Gaussian distribution, instead of input probability densities only mean and variance of Gaussian function is tracked according to coding structure [25, 27, 33]. If σ^2 denotes the Gaussian noise variance of original channel W, mean and variance of log-likelihood ratio (LLR) messages from channel will be $\frac{2}{\sigma^2}$ and $\frac{4}{\sigma^2}$ respectively $\mathcal{N}(\frac{2}{\sigma^2}, \frac{4}{\sigma^2})$.

Gaussian approximation for polar code has the same principle with [31, 32]. As elaborately explained in [33] summation of two LLRs having same Gaussian distribution with $\mathcal{N}(a, 2a)$, result is going to have Gaussian distribution with $\mathcal{N}(2a, 4a)$. Similarly, subtracted LLRs having same distribution with above ones is going to have a Gaussian distribution with $\mathcal{N}(0, 4a)$. This approach makes things easier under the assumption that all previous bits are decoded correctly with SC decoder because mean and variance of Gaussian variable are consistent. Therefore, only calculating the mean value of Gaussian variable will be sufficient to find the BERs of bit channels. Gaussian approximation for polar code construction has following pin points:

- Recursive update of LLRs under SC decoder for check and variable nodes respectively:
$L_N^{(1)}(L_1^N) = L_{N/2}^{(1)}(L_1^{N/2}) \boxplus L_{N/2}^{(1)}(L_{N/2+1}^N)$ and
$L_N^{(2)}(L_1^N, \hat{u}_1) = L_{N/2}^{(1)}(L_{N/2+1}^N) + (-1)^{\hat{u}_1} L_{N/2}^{(1)}(L_1^{N/2})$.

- Above equations indicate the results of them is a consistent normal density variable under the assumption of previous bits are correctly decoded.

- Consistent normal density variable has mean value which is the half of its variance. Therefore, only calculating the mean value is enough.

- Mean values of recursive LLRs can be calculated with an approximation [32] as below:
$E[L_N^{(i)}] = \phi^{-1}(1-(1-\phi(E[L_{N/2}^{((i+1)/2)}]))^2)$ if i is odd,
$E[L_N^{(i)}] = 2E[L_{N/2}^{(i/2)}]$ if i is even,
where;
$$\phi(x) = \begin{cases} exp(-0.4527x^{(0.86)} + 0.0218), & 0 < \quad x < 10, \\ \sqrt{\frac{\pi}{x}} exp(-\frac{x}{4})(1-\frac{10}{7x}), & \quad x \geq 10. \end{cases}$$
Here $E[.]$ indicates the mean value of probability density function.

- Rest is calculating error probabilities to find BERs of bit channels according to below equation.
$P(C_i) = \frac{1}{2} erfc\left(0.5\sqrt{E[L_N^{(i)}]}\right)$

This method has $\mathscr{O}(N)$ complexity [33].

2.4.2.5. Density Evolution

As mentioned in [28] each step of SC decoder can be handled as a BP decoding. Gaussian approximation follows same principle with density evolution method. The difference between these two methods is while Gaussian approximation calculates bit channels PDFs only tracking their mean values trough LLR update process, density evolution calculates the real PDFs without simplifications. Here is the main structure of density evolution method:

- Recursive update of LLRs under SC decoder for check and variable nodes respectively:
$L_N^{(2i-1)}(y_1^N, \hat{u}_1^{2i-2}) =$
$2\tanh^{-1}\left(\tanh(L_{N/2}^{(i)}(y_1^{N/2}, \hat{u}_{1,e}^{2i-2} \oplus \hat{u}_{1,o}^{2i-2})/2) \times \tanh(L_{N/2}^{(i)}(y_{N/2+1}^N, \hat{u}_{1,e}^{2i-2})/2)\right)$
and
$L_N^{(2i)}(y_1^N, \hat{u}_1^{2i-1}) = L_{N/2}^{(i)}(y_{N/2+1}^N, \hat{u}_{1,e}^{2i-2}) + (-1)^{\hat{u}_{2i-1}} L_{N/2}^{(i)}(y_1^{N/2}, \hat{u}_{1,e}^{2i-2} \oplus \hat{u}_{1,o}^{2i-2})$.

- If the original channel (W) has LLR PDF $a_1^{(1)} = a_w$ update of PDFs become as below: $a_{2N}^{(2i)} = a_N^i \star a_N^i$ and $a_{2N}^{(2i-1)} = a_N^i \boxed{*} a_N^i$.
  Here $\star$ and $\boxed{*}$ are the convolution operations in a variable node domain and a check node domain, respectively.

- Bit channels error probabilities are calculated as:
  $P(\mathscr{A}_i) = \int_{-\infty}^{0} 2^{-\mathbb{I}(x=0)} a_N^{(i)}(x)dx$ [28, 31].

- Rest is choosing bit channels according to lowest $P(\mathscr{A}_i)$.

This method need to perform $\mathscr{O}(N)$ convolutions to calculate $P(\mathscr{A}_i)$.

2.4.3. Polar Code Encoding

In this section we present polar encoding by its generally used notations. As mentioned at beginning of Section 2.4.1 a B-DMC which is defined as $W : X \to Y$ with input alphabet X and output alphabet Y. N times use of channel W is denoted as W^N and their transitions denoted as $W^N : X^N \to Y^N$ with $W^N(y_1^N|x_1^N) = \prod_{i=1}^{N} W(y_i|x_i)$.

Generator matrix produced by kernel matrix G_2 used to encode input sequence U given the output sequence X and represented as $x_1^N = u_1^N G_N$. Information set and frozen set can be represented individually with Eqn. (2.28). In this equation $\mathscr{A}$ represents information set and $\mathscr{A}^c$ represents frozen set. $u_{\mathscr{A}^c} \in X^{N-K}$ represents frozen bits.

$$x_1^N = u_{\mathscr{A}} G_N(\mathscr{A}) \oplus u_{\mathscr{A}^c} G_N(\mathscr{A}^c) \tag{2.28}$$

A polar code with code length N and information bits amount with K has rate $R = K/N$ and represented as $(N, K, \mathscr{A}, u_{\mathscr{A}^c})$. An example is given in Eqn. (2.29) which is exactly same given in [9].

$$x_1^4 = u_4^1 G_4 = (u_2, u_4) \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix} + (1,0) \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} \tag{2.29}$$

So the encoded sequence will become $x_1^4 = (1,1,0,1)$ if the information bits are $(u_2, u_4) = (1,1)$.

If we look at the Bhattacharyya parameters for a polar code constructed for BEC example for encoding can be seen more clearly. For $N = 8$ and $W_1^{(1)} = 0.5$ Bhattacharyya

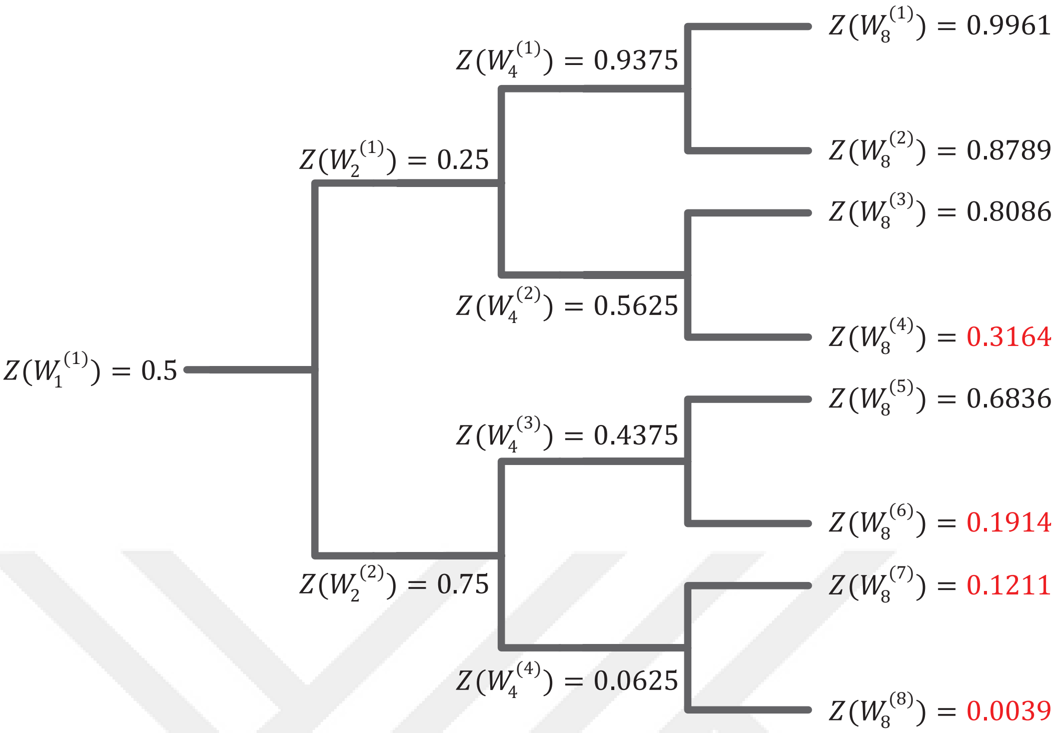


Figure 2.12. Recursive Bhattacharyya parameters for BEC with $W_1^{(1)} = 0.5$ for $N = 8$.

parameters will be as in Fig. 2.12. Red indicated values are the lowest Bhattacharyya values to be used as information bit channels.

The resulting code sequence can be calculated as in Eqn. (2.30). Here, inside the vector, "0" values are frozen bits and (u_4, u_6, u_7, u_8) are information bits.

$$x_1^8 = u_8^1 G_8 = (0,0,0,u_4,0,u_6,u_7,u_8) \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{2.30}$$

2.4.4. Polar Code Decoding

Mainly there are two types of decoder proposed for polar code. First one is SC decoder proposed in [9] which can be considered as a special form of belief propagation (BP) decoder which is the second one [34]. Both decoder has their advantages for instance SC decoder has serial structure and requires lower complexity, on the other hand BP decoder has parallel structure and higher complexity due to its iterative nature [34]. SC decoder's serial structure manifest itself as higher decoding latency resulting lower throughput compared to BP. As we can say so far this is the main disadvantage of polar code with SC decoder along with other ECCs which has serial decoding strategies.

2.4.4.1.Successive Cancellation Decoder

SC is the first decoder proposed by Arikan for decoding polar code [9]. As evident by its name bits are decoded sequentially starting from first frozen bit which is already known. Arikan referred this process as using N decision elements (DE) ordered from 1 to N by observing (y_1^N, u_{A^c}) and estimating $(\hat{u}_1^N)$. If $i \in \mathscr{A}^c$ meaning that (u_i) is a frozen bit then ith bit decoded as "0" and this information passes trough to all succeeding DEs. If $i \in \mathscr{A}$ meaning that (u_i) is an information bit and likelihood ratio (LR) (or in log domain LLR) of $(\hat{u}_i)$ is calculated by Eqn. (2.31) and decision is made by Eqn. (2.32).

$$L_N^{(i)}(y_1^N, \hat{u}_1^{i-1}) \triangleq \frac{W_N^{(i)}(y_1^N, \hat{u}_1^{i-1}|0)}{W_N^{(i)}(y_1^N, \hat{u}_1^{i-1}|1)} \qquad \lambda_N^{(i)}(y_1^N, \hat{u}_1^{i-1}) \triangleq log\frac{W_N^{(i)}(y_1^N, \hat{u}_1^{i-1}|0)}{W_N^{(i)}(y_1^N, \hat{u}_1^{i-1}|1)} \tag{2.31}$$

$$\hat{u}_i = \begin{cases} 0, & if \quad L_N^{(i)}(y_1^N, \hat{u}_1^{i-1}) \geq 1 \\ 1, & otherwise \end{cases} \tag{2.32}$$

LR and LLR values can be calculated using recursive formulas with Eqn. (2.33) and Eqn. (2.34) as in [9, 35].

$$\begin{aligned} L_N^{(2i-2)}(y_1^N, \hat{u}_1^{2i-2}) &= \frac{L_{N/2}^{(i)}\left(y_1^{N/2}, \hat{u}_{1,o}^{2i-2} \oplus \hat{u}_{1,e}^{2i-2}\right) L_{N/2}^{(i)}\left(y_{N/2+1}^{N}, \hat{u}_{1,e}^{2i-2}\right) + 1}{L_{N/2}^{(i)}\left(y_1^{N/2}, \hat{u}_{1,o}^{2i-2} \oplus \hat{u}_{1,e}^{2i-2}\right) + L_{N/2}^{(i)}\left(y_{N/2+1}^{N}, \hat{u}_{1,e}^{2i-2}\right)} \\ L_N^{(2i)}(y_1^N, \hat{u}_1^{2i-1}) &= \left[L_{N/2}^{(i)}(y_1^{N/2}, \hat{u}_{1,o}^{2i-2} \oplus \hat{u}_{1,e}^{2i-2})\right]^{1-2\hat{u}_{2i-1}} L_{N/2}^{(i)}(y_{N/2+1}^{N}, \hat{u}_{1,e}^{2i-2}) \end{aligned} \tag{2.33}$$

$$\begin{aligned} \lambda_N^{(2i-2)}(y_1^N, \hat{u}_1^{2i-2}) &= 2tanh^{-1}\Bigg(\quad tanh\Big(\lambda_{N/2}^{(i)}(y_1^{N/2}, \hat{u}_{1,o}^{2i-2} \oplus \hat{u}_{1,e}^{2i-2})\Big)/2 \\ &\qquad\qquad tanh\Big(\lambda_{N/2}^{(i)}(y_{N/2+1}^{N}, \hat{u}_{1,e}^{2i-2})\Big)/2 \quad \Bigg) \\ \lambda_N^{(2i)}(y_1^N, \hat{u}_1^{2i-1}) &= \lambda_{N/2}^{(i)}(y_{N/2+1}^{N}, \quad \hat{u}_{1,e}^{2i-2}) + (-1)^{\hat{u}_{2i-2}}\left(\lambda\left(y_1^{N/2}, \hat{u}_{1,o}^{2i-2} \oplus \hat{u}_{1,e}^{2i-2}\right)\right). \end{aligned} \tag{2.34}$$

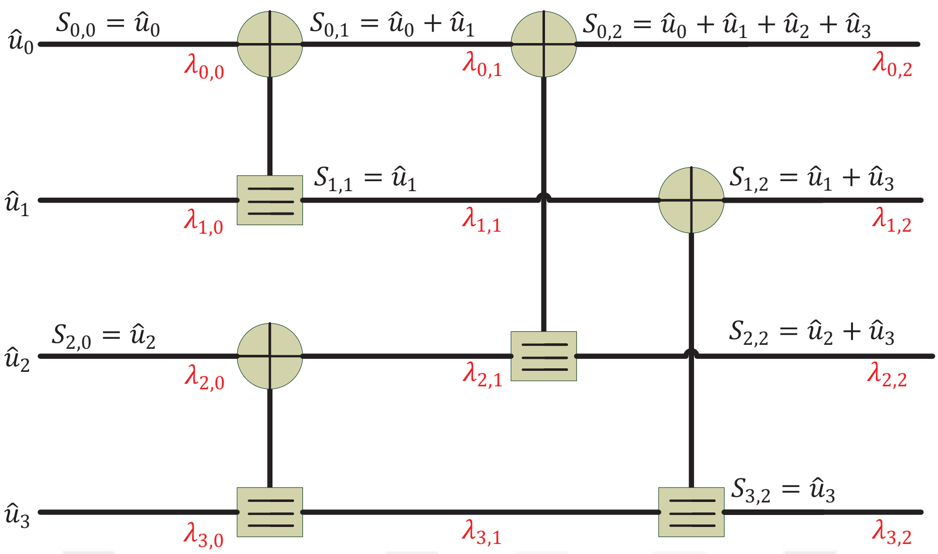


Figure 2.13. Tanner graph for $N = 4$ polar code and LLR evaluations.

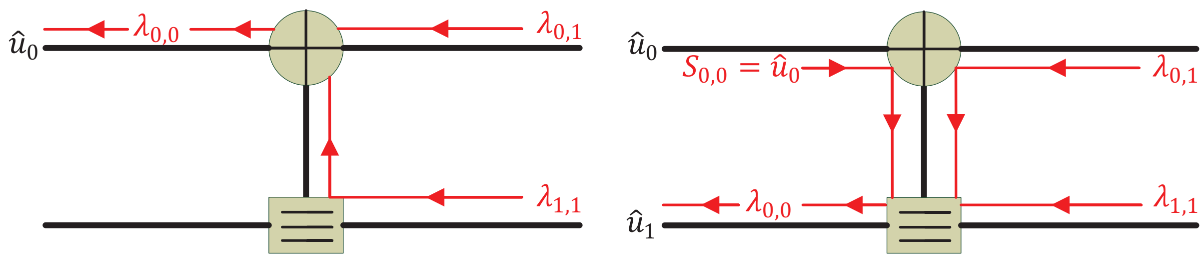
LLR domain is more practical for simplification of calculations especially with min-sum (MS) approach [36, 37] as given in Eqn. (2.35) and example with Fig. 2.13 and Fig. 2.14. If we consider LLR value upper and lower formulas in Eqn. (2.34) as f and g respectively and their input LLRs as a, b and s (as previously decided bit), MS formulation will become as in Eqn. (2.35). Here, $tanh$ and $tanh^{-1}$ functions are replaced by a multiplication and a compare process.

$$\begin{aligned} f(a,b) &= sign(ab) \times min(|a|,|b|) \\ g(a,b,s) &= b \oplus (-1)^s a \end{aligned} \tag{2.35}$$

In Fig. 2.13 Tanner graph for $N = 4$ polar decoder is given with LLR notations. In Fig. 2.14 successive decoding of $\hat{u}_0$ and $\hat{u}_1$ is explained. Here in both figures ($\oplus$) and ($\boxed{\equiv}$) symbolize check node (CN) and variable node (VN), respectively [36]. In Fig. 2.14(a) $\lambda_{0,0}$ is calculated with f in Eqn. (2.35) and hard decision is made for $\hat{u}_0$ by Eqn. (2.32). The information come from hard decision of $\hat{u}_0$ passed to next function g as s, finally decision is made for $\hat{u}_1$ by Eqn. (2.32).

Although, there are many enhanced SC decoding strategies in literature the most important ones are SC list (SCL) [38], stack (SCS) [39] and hybrid (SCH) [40], which is a combination of SCL and SCS decoders. Another important study is cyclic redundancy check (CRC)-aided SC decoder which outperforms ML bound, LDPC and Turbo codes proposed in [41] where both SCL and SCS are aided with CRC.

On the other hand improved performance comes with complexity cost. SCL can reach ML bound however list size (L) brings huge complexity to decoding process as $\mathcal{O}(L \times NlogN)$.



(a) Calculation order of $\hat{u}_0$ (b) Calculation order of $\hat{u}_1$

Figure 2.14. 2 bits polar SC decoder example

The best result with SCL decoder presented in [38] is when list size $L = 32$ which requires 32 times more calculations compared to traditional SC decoder. SCH decoder reduces the complexity compared to SCL but it has a large hardware space complexity. A simplified CRC-aided SCL decoder is proposed in [42] which selectively reduces the list size without any major performance degradation.

2.4.4.2. Belief Propagation Decoder

This thesis focused on BP strategy cause even most recent studies of both decoder shows that the best of SC [43] can not reach the throughput performance of BP [44, 45]. However, this cost is paid by $energy/bit$ increment. In [44] a BP decoding strategy is proposed with 4.68 Gb/s throughput while in [43] it is still 3.54 Gb/s with SC decoder. On the other hand, $energy/bit$ efficiency of SC decoder in [43] 5 times better than [44, 46].

BP is a well known decoder which has been widely used for decoding LDPC [5, 47], LT or raptor [8] codes. First use of BP for polar code was in [18, 48] with its conventional form. In [49] one of the first hardware implementation is published for polar BP decoder with min-sum (MS) approximation. Without scaled min-sum (SMS) approach proposed in [50] ,which is also optimized for logic implementations, BER performance of BP decoder is limited and even worse than SC.

BP decoder for polar code handle the Tanner graph by dividing it to $m = log_2 N$ stages (see Fig. 2.15) where each stage has $N/2$ processing elements (PE) (see Fig. 2.16) as the DEs in SC decoder. However, PEs in BP decoder produce outputs simultaneously and pass the information to successive stage which gives the benefit of parallel structure to BP.

Calculations made by PE given with formulas in LLR domain (from now on R and L will refer LLRs instead of λ) given with Eqn. (2.36). A detailed explanation of iterative BP decoding process is given with Algorithm 1.

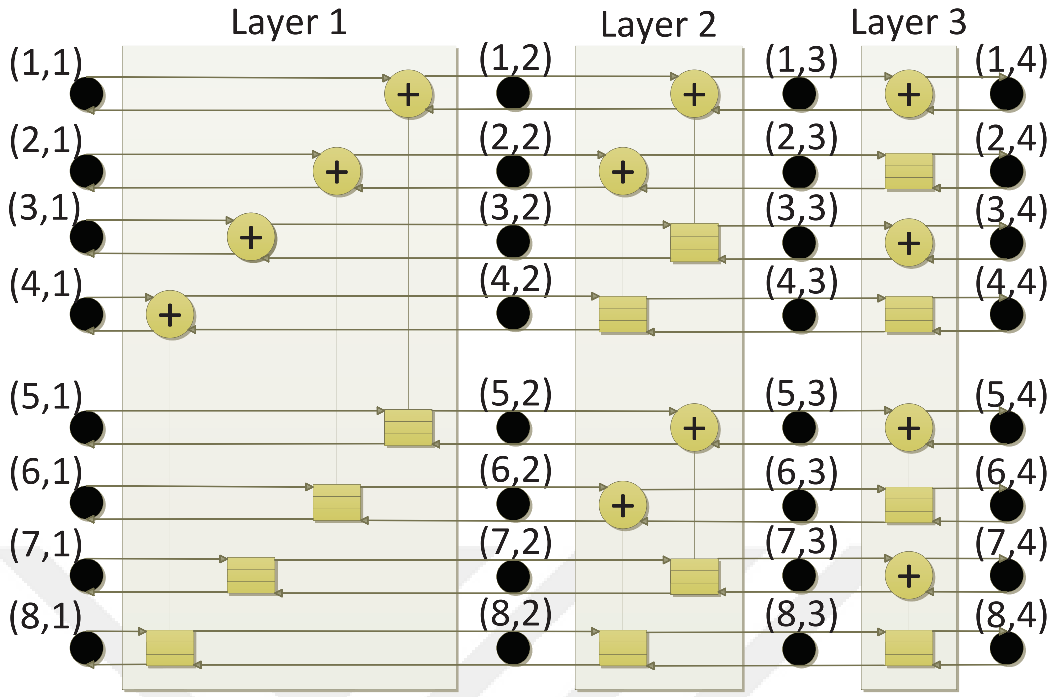


Figure 2.15. Tanner graph for $N = 8$ polar code with $m = 3$ layers.

$$\begin{aligned}
L_{i,j}^{t} &= s \cdot sign\left(L_{i,j+1}^{t-1}\right) sign\left(L_{i+n/2^j,j+1}^{t-1} + R_{i+n/2^j,j}^{t}\right) \\
&\quad min\left(\left|L_{i,j+1}^{t-1}\right|, \left|L_{i+n/2^j,j+1}^{t-1} + R_{i+n/2^j,j}^{t}\right|\right) \\
L_{i+n/2^j,j}^{t} &= L_{i+n/2^j,j+1}^{t-1} + s \cdot sign\left(L_{i,j+1}^{t-1}\right) sign\left(R_{i,j}^{t}\right) min\left(\left|L_{i,j+1}^{t-1}\right|, \left|R_{i,j}^{t}\right|\right) \\
R_{i,j+1}^{t} &= s \cdot sign\left(R_{i,j}^{t}\right) sign\left(L_{i+n/2^j,j+1}^{t-1} + R_{i+n/2^j,j}^{t}\right) \\
&\quad min\left(\left|R_{i,j}^{t}\right|, \left|L_{i+n/2^j,j+1}^{t-1} + R_{i+n/2^j,j}^{t}\right|\right) \\
R_{i+n/2^j,j+1}^{t} &= R_{i+n/2^j,j}^{t} + s \cdot sign\left(L_{i,j+1}^{t-1}\right) sign\left(R_{i,j}^{t}\right) min\left(\left|L_{i,j+1}^{t-1}\right|, \left|R_{i,j}^{t}\right|\right)
\end{aligned} \tag{2.36}$$

$$\hat{u}_i = sign(LLR_{i,1}^{max\_iter}) \triangleq sign(R_{i,1}^{max\_iter} + L_{i,1}^{max\_iter}) \tag{2.37}$$

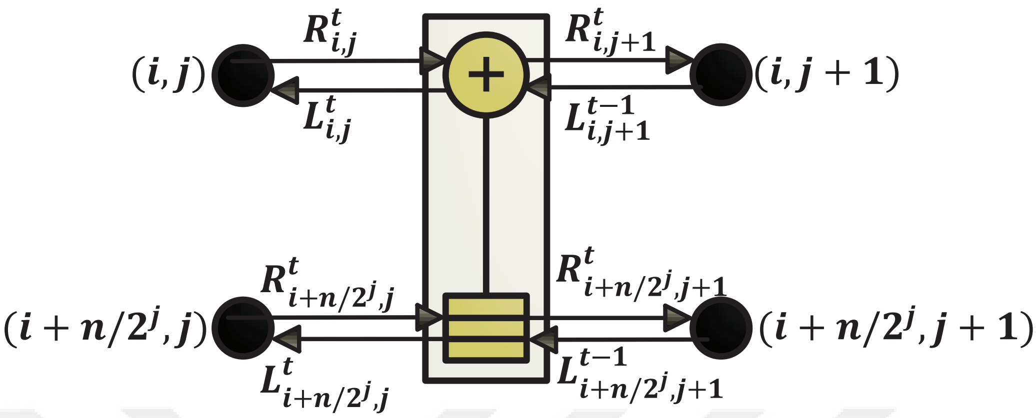


Figure 2.16. Processing element for BP polar decoder.

Algorithm 1 Iterative SMS BP (n,k) Polar Code Decoder:

```
procedure INITIALIZATION(LLR(r_i), Frozen)
    while t < max_iter do                     ▷ Fill R^t_{i,j} and L^t_{i,j} with initial values.
        if (j == 1)&(i ∈ Frozen) then
            R^t_{i,1} = ∞                      ▷ Frozen bits filled with high LLR values.
        else if (j == m+1) then
            L^t_{i,m+1} = LLR(r_i)             ▷ Channel output LLRs loaded.
        else
            R^0_{i,j} = L^0_{i,j} = 0
procedure ITERATION(Initials, max_iter, s)
    while t < max_iter do
        for i = 1 to i = N/2 do                ▷ N/2 PEs.
            for j = 1 to j = m+1 do            ▷ m Layers.
                Update LLR values according to Eqn. (2.36).
        t = t+1                                ▷ Next iteration.
    for i=1...N do
        û_i = sign(R^t_{i,1} + L^t_{i,1})      ▷ Bits are detected Eqn. (2.37).
```

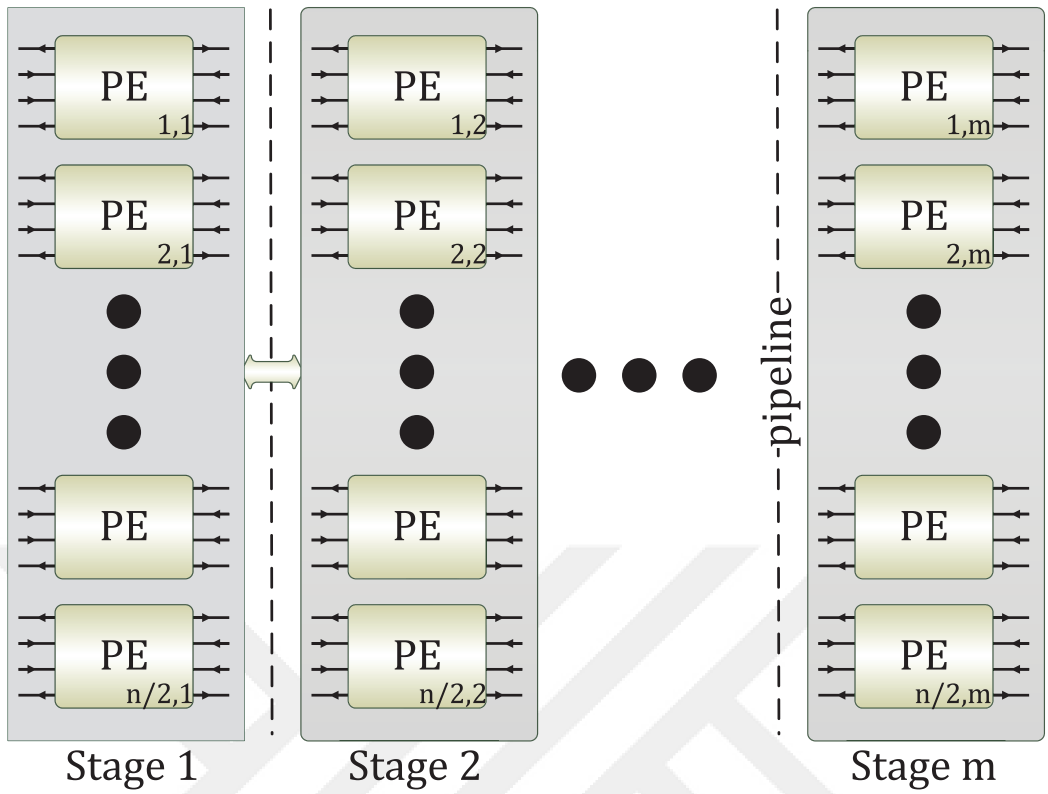


Figure 2.17. $N/2 \times m$ PEs pipelined BP decoder structure.

2.4.4.3. Hardware Structure of BP Decoder

LLR messages from channel inputs are propagated and updated iteratively as Tanner graph in Fig. 2.15. But the propagation is a parallel operation carried by PEs as in Fig. 2.17 by pipelined structure [50]. This iterative pipelined structure is the point where BP has the advantage over SC decoder, because with SC decoder bits need to be decoded sequentially which increases the delay and reduces the throughput.

PE is optimized for logic system implementation. For instance normally scaling operation is multiplying LLR values by a constant (s), but in [50] this operation is optimized as in Eqn. (2.38) and Fig. 2.21(a) for logic systems. Other elements in PE (Fig. 2.18) are optimized in [50] as well.

Type-I and Type-II block schemes are explained with Fig. 2.19 and Fig. 2.20. Blocks in PE (see Fig. 2.18) given with Fig. 2.22(a), 2.22(b), 2.21(a), 2.21(b) has a design for lower critical path delay and fewer logic elements. For instance 1 bit summation operation in 2's complement conversion units is distributed over summation operation to reduce critical path delay caused by extra addition operations.

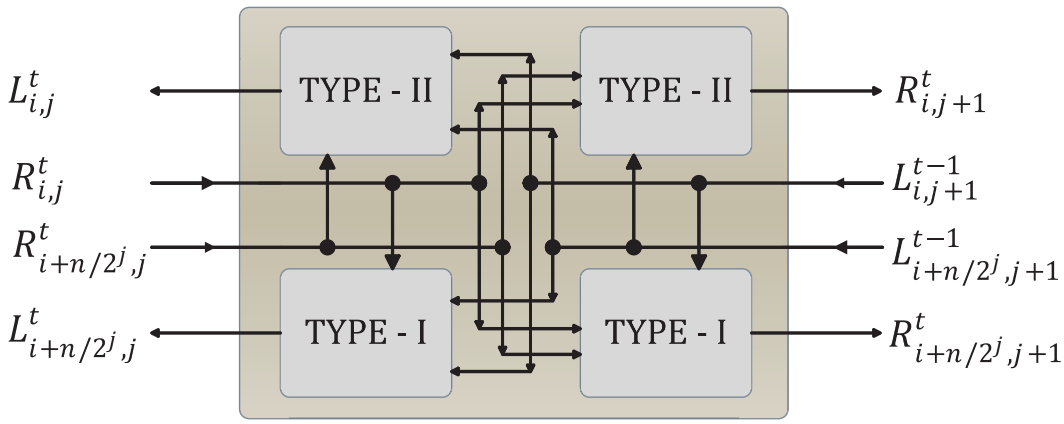


Figure 2.18. Inner structure of PE.

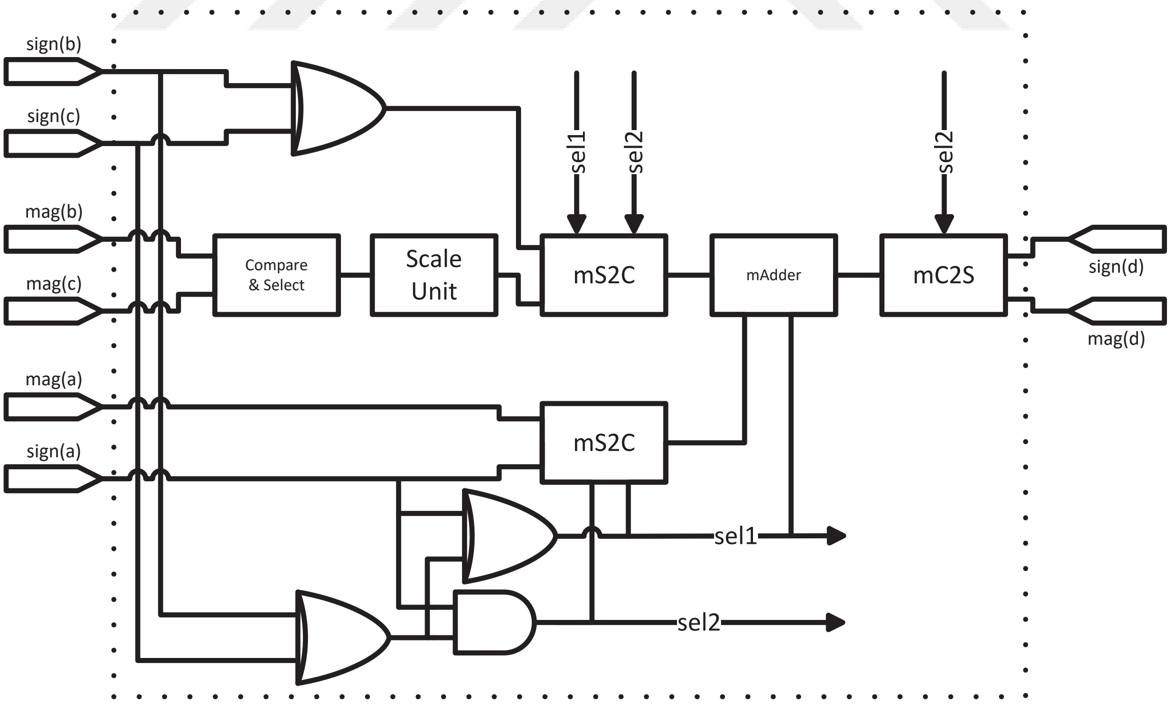


Figure 2.19. Inner structure of Type-I.

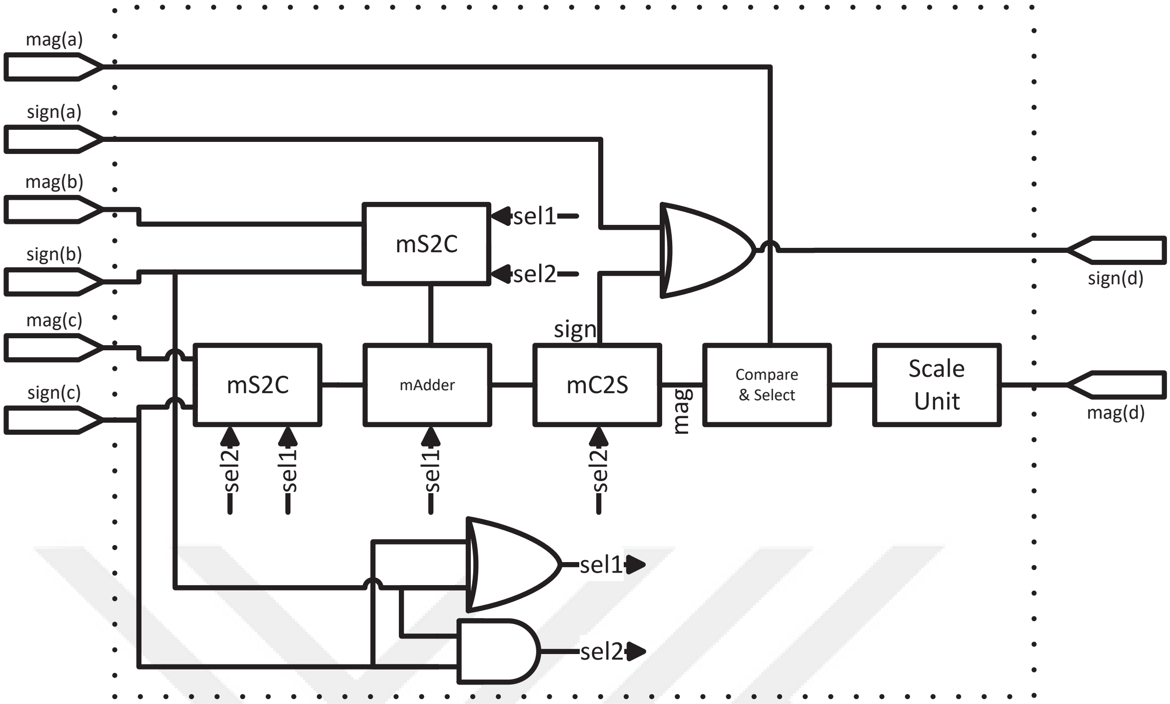


Figure 2.20. Inner structure of Type-II.

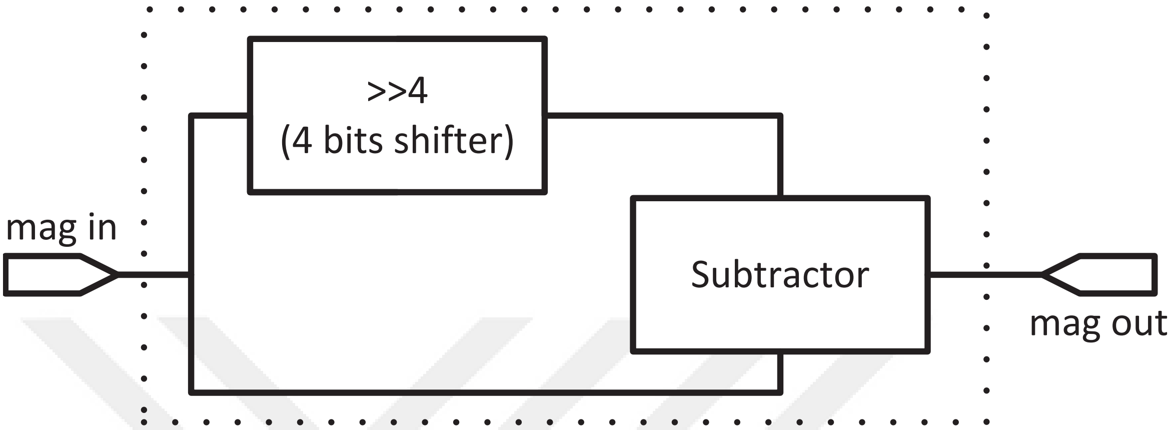
As one can notice choosing $s = 0.9375$ is a delicate one, as seen in Fig. 2.21(a) and Eqn. (2.38) operation can be done by 4 times right shifting a binary sequence lets say this sequence a, is equal to division of it 16 meaning that $a/16$ and subtracting the result from sequence itself $(0.9375 \times a = a - a/16)$. This is the most efficient and hardware friendly way for scaling factor used in SMS-BP polar code decoder. All of these hardware friendly structures proposed in [50] also the ones we used in our studies.

$$LLR \times 0.9375 = LLR - \frac{LLR}{16} \tag{2.38}$$

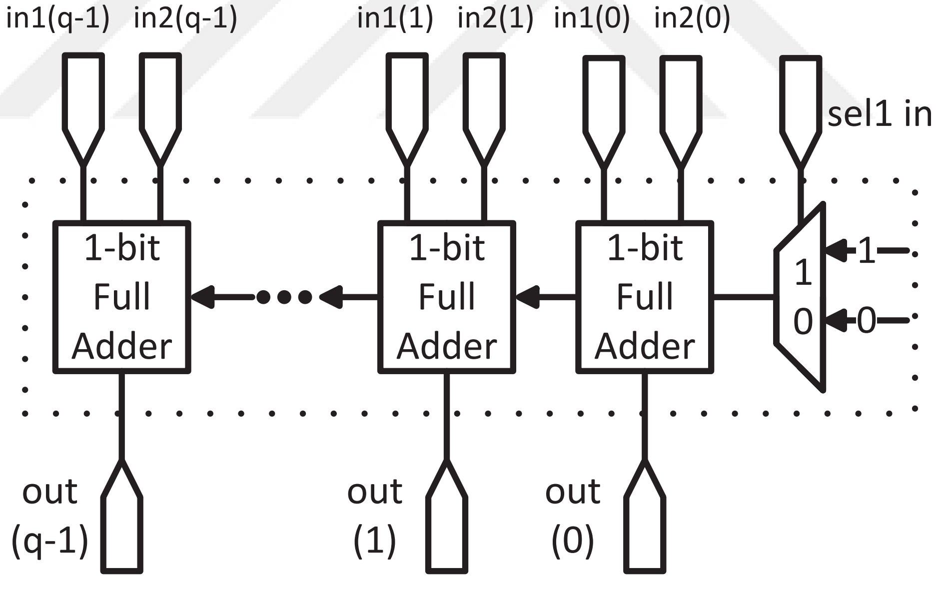
2.4.4.4. Early Stopping Criteria for BP Polar Code Decoders

An iterative BP decoder without early stopping method uses fixed iteration number to end iterative decoding process. But decoding may be successful before iteration number reaches this fixed limit. In these cases, decoder performs a redundant process. Therefore, stopping the iterations is essential to keep computational burden as low as possible when decoding is successful. In literature, there are two different early stopping criterion methods for BP polar code decoders both proposed in [46].

First method is called G-Matrix which uses the generator matrix of polar codes at each iteration to provide early stopping detection. During iterations, bits are detected according to

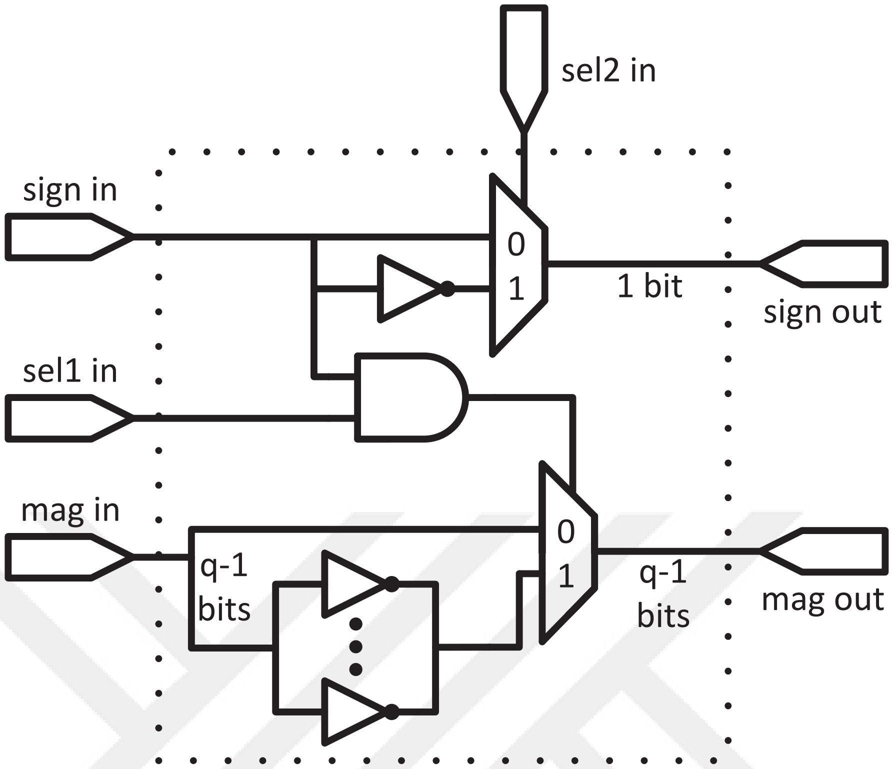


(a) Inner structure of scale unit $s = 0.9375$.

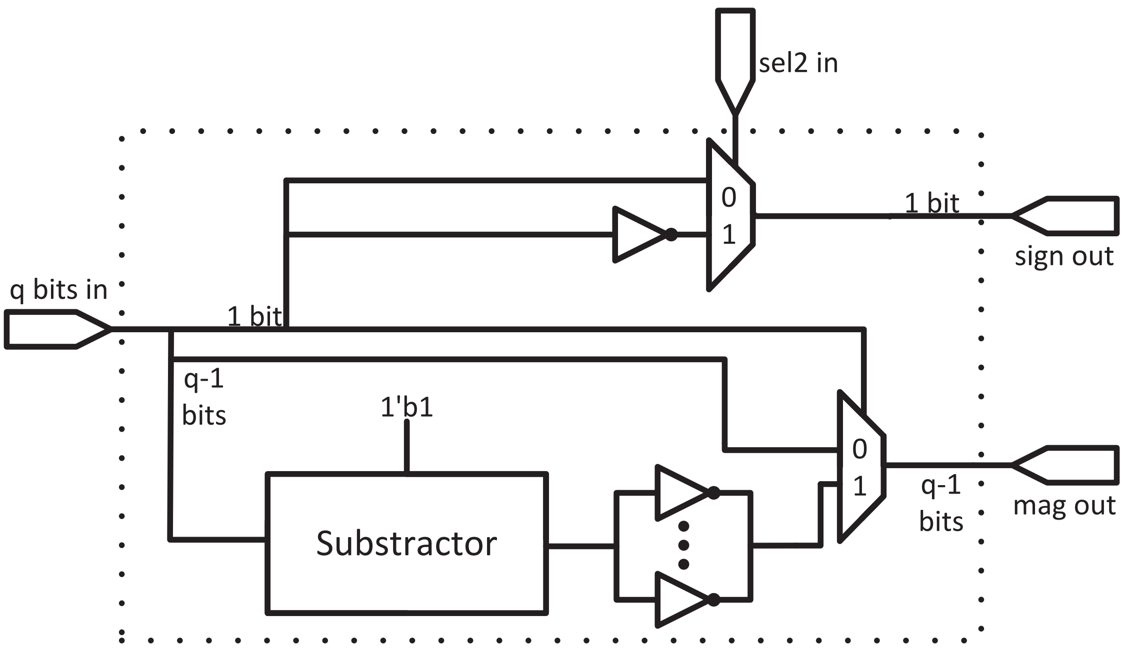


(b) Inner structure of mAdder module.

Figure 2.21. Logic implementations of elements used in mS2C and mC2S modules.



(a) Inner structure of mS2C module.



(b) Inner structure of mC2S module.

Figure 2.22. Logic implementations of elements used in PEs.

LLR values of input and output nodes at factor graph of decoder. Then, detected output bits are re-encoded by using the same generator matrix and compared with the input bits. If the input and output bits are equal to each other which makes the comparison result zero, method assumes that decoding is successful.

The second method is called minLLR. This method uses the magnitudes of LLR values at last nodes. MinLLR searches for minimum LLR magnitude to compare it with a predetermined β value. If this minimum value is bigger than β value, the method assumes that decoding is successful. However, the method has a performance loss at high SNR region. Because of that, this method is supported with a SNR detection method in order to switch the β to a higher predetermined constant for higher SNR region. This modified method is called adaptive minLLR.

2.4.4.5. G-Matrix Early Stopping Criterion

Usually block ECCs uses a parity check matrix denoted as H to detect a successful decoding such as LDPC [51]. Multiplication input of encoder (x) and transpose of parity check matrix (H^T) always produce zero result (see Eqn. (2.39)) [52].

$$xH^T = 0 \tag{2.39}$$

Same principle is used in G-Matrix method however for polar code there is no parity check matrix. Instead of parity check, generator matrix which is denoted as (G) is used. As in Eqn. (2.40) when the estimated decoder output bits $(\hat{u})$ are re-encoded with generator matrix (G) the result should be equal to input bits of decoder $(\hat{x})$ if decoding is successful.

$$\hat{x} = \hat{u}G \tag{2.40}$$

Detailed explanation of G-Matrix method is given with Algorithm 2 and block diagram in Fig. 2.23.

2.4.4.6. MinLLR Early Stopping Criterion

This ESC is the second early stopping strategy proposed in [46]. As given in Section 2.4.4.2 BP Decoder decision of estimated bits $(\hat{u}_i)$ is made by Eqn. (2.37) which hard decision is only made by sign values of $LLR^t_{i,1}$. However magnitude of $LLR^t_{i,1}$ is also an important metric to approximate reliability of bit estimation. One can see from

Algorithm 2 Iterative SMS BP (n,k) Polar Code Decoder with G-Matrix ESC:

procedure INITIALIZATION$(LLR(r_i), Frozen)$
  while $t < max\_iter$ **do** ▷ Fill $R^t_{i,j}$ and $L^t_{i,j}$ with initial values.
    if $(j == 1) \& (i \in Frozen)$ **then**
      $R^t_{i,1} = \infty$ ▷ Frozen bits filled with high LLR values.
    else if $(j == m+1)$ **then**
      $L^t_{i,m+1} = LLR(r_i)$ ▷ Channel output LLRs loaded.
    else
      $R^0_{i,j} = L^0_{i,j} = 0$
procedure ITERATION$(Initials, max_iter, s)$
  while $t < max\_iter$ **do**
    for $i = 1$ to $i = N/2$ **do** ▷ $N/2$ PEs.
      for $j = 1$ to $j = m+1$ **do** ▷ m Layers.
        Update LLR values according to Eqn. (2.36).
    if $(L^t_{i,m+1} + R^t_{i,m+1} \geq 0)$ **then**
      $\hat{x}_i = 0$
    else ▷ Update $\hat{x}$ vector.
      $\hat{x}_i = 1$
    if $(L^t_{i,1} + R^t_{i,1} \geq 0)$ **then**
      $\hat{u}_i = 0$
    else ▷ Update $\hat{u}$ vector.
      $\hat{u}_i = 1$
    if $\hat{u}G = \hat{x}$ **then**
      $Output = \hat{u}$ ▷ Decoding assumed successful and output is $\hat{u}$.
    else
      $t = t + 1$ ▷ Next iteration.
  Output: $\hat{u} = (\hat{u}_1, \hat{u}_2 ... \hat{u}_N)$

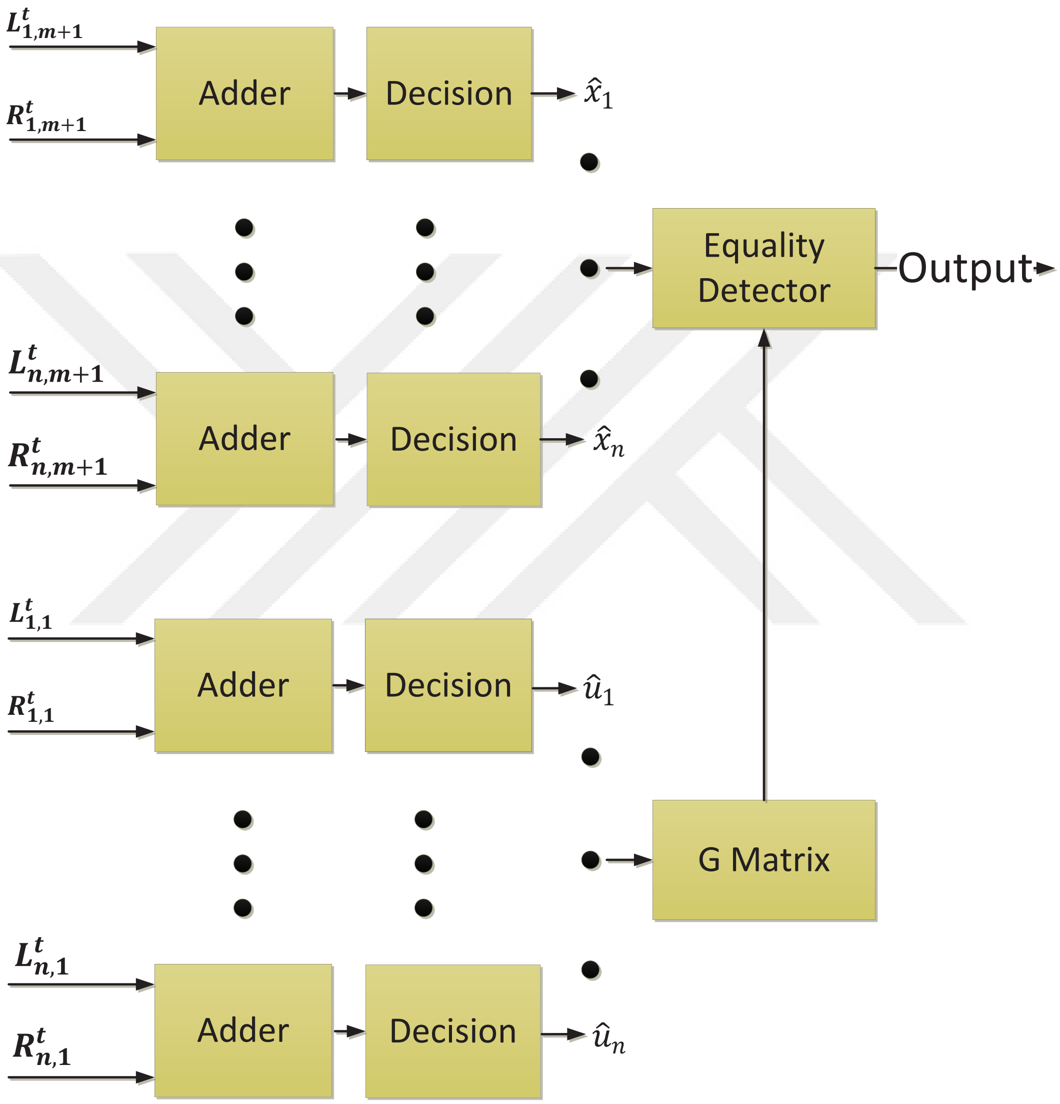


Figure 2.23. Block scheme of G-Matrix ESC.

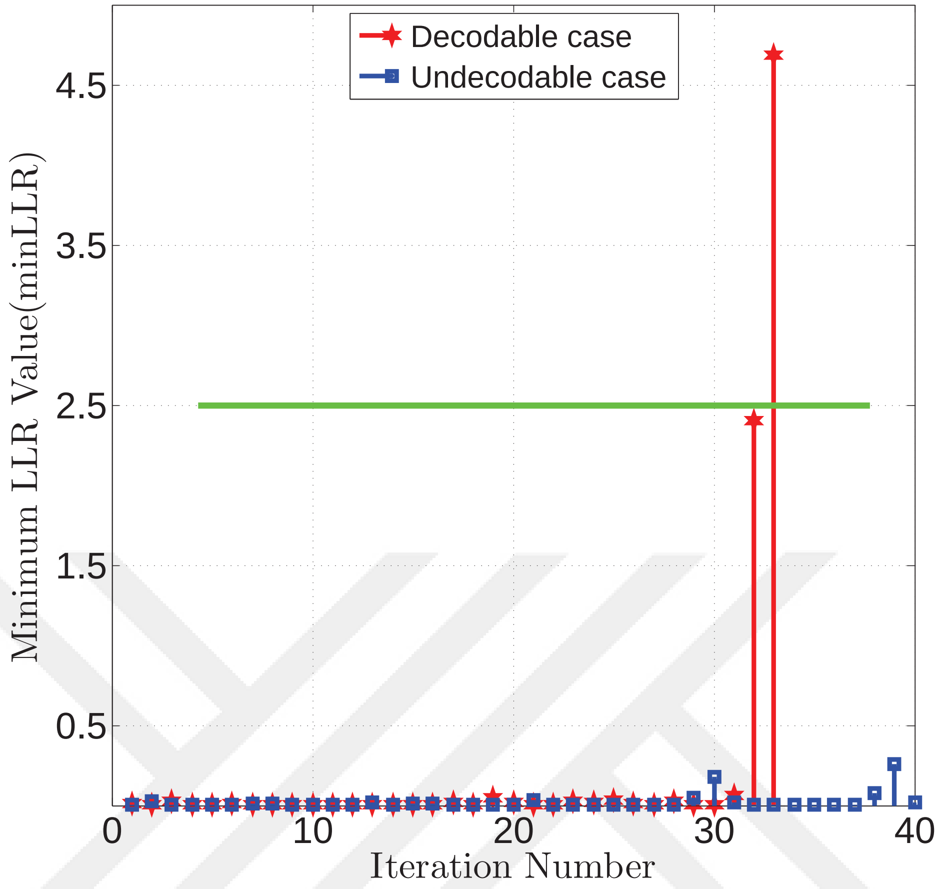


Figure 2.24. Evolution of β for one decodable and one undecodable case ($SNR = 2.5dB$, $N = 1024$ and $R = 0.5$).

Eqn. (2.41) without paying any attention to value of $sign(LLR^t_{i,1})$, magnitude $|LLR^t_{i,1}|$ tells whether probabilities are departed or not.

$$LLR^t_{i,1} = log(Pr(\hat{u}_i = 0)/Pr(\hat{u}_i = 1)) \tag{2.41}$$

If one of the probabilities becomes dominant the bit can be decided. This is made with an empirical approach in [46]. The minimum of $|LLR^t_{i,1}|$ value is selected among LLR magnitudes of decoder output and a β value is determined by observing it as in Fig. 2.24. As one can see from the figure *minLLR* value for decodable case starts to increase which means probabilities departs from each other, on the other side *minLLR* value for undecodable case remains under β limit till the end of iteration process. This situation suggests that even with the smallest LLR, bits can be estimated cause ratio of probabilities becomes at least $e^{\beta=2.5} = 12$. With the help of this empirical approach β is determined as 2.5 in [46]. Detailed explanation of minLLR method is given with Algorithm 3 and block scheme in Fig. 2.25.

Algorithm 3 Iterative SMS BP (n,k) Polar Code Decoder with minLLR ESC:

procedure INITIALIZATION($LLR(r_i), Frozen$)
 while $t < max\_iter$ **do**   ▷ Fill $R^t_{i,j}$ and $L^t_{i,j}$ with initial values.
  if $(j == 1)\&(i \in Frozen)$ **then**
   $R^t_{i,1} = \infty$ ▷ Frozen bits filled with high LLR values.
  else if $(j == m+1)$ **then**
   $L^t_{i,m+1} = LLR(r_i)$ ▷ Channel output LLRs loaded.
  else
   $R^0_{i,j} = L^0_{i,j} = 0$
procedure ITERATION($Initials, max\_iter, s$)
 while $t < max\_iter$ **do**
  for $i = 1$ to $i = N/2$ **do**   ▷ $N/2$ PEs.
   for $j = 1$ to $j = m+1$ **do**   ▷ m Layers.
    Update LLR values according to Eqn. (2.36).
  $min\{|R^t_{i,1} + L^t_{i,1}|\}$ ▷ Find minimum $|LLR|$ value among output LLRs.
  if $minLLR \geq \beta$ **then**
   $Output = \hat{u}$ ▷ Decoding assumed successful and output is $\hat{u}$.
  else
   $t = t+1$ ▷ Next iteration.
 Output: $\hat{u} = (\hat{u}_1, \hat{u}_2 ... \hat{u}_N)$

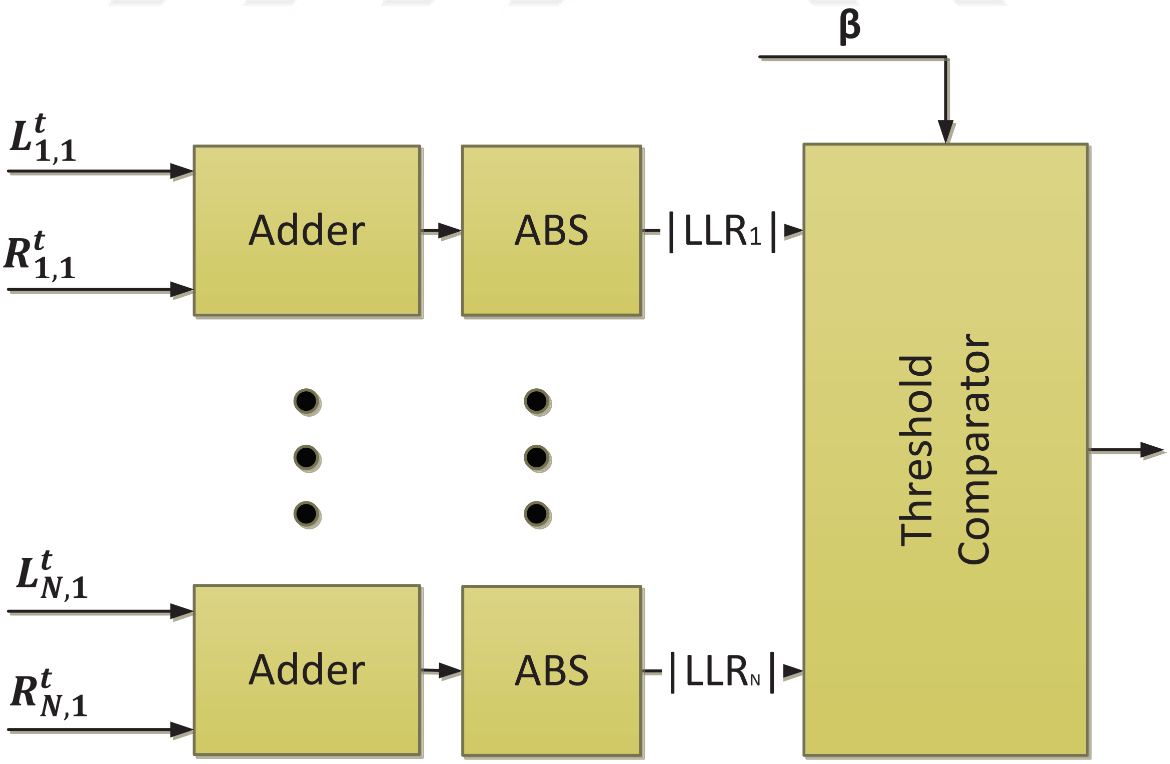


Figure 2.25. Block scheme of minLLR ESC.

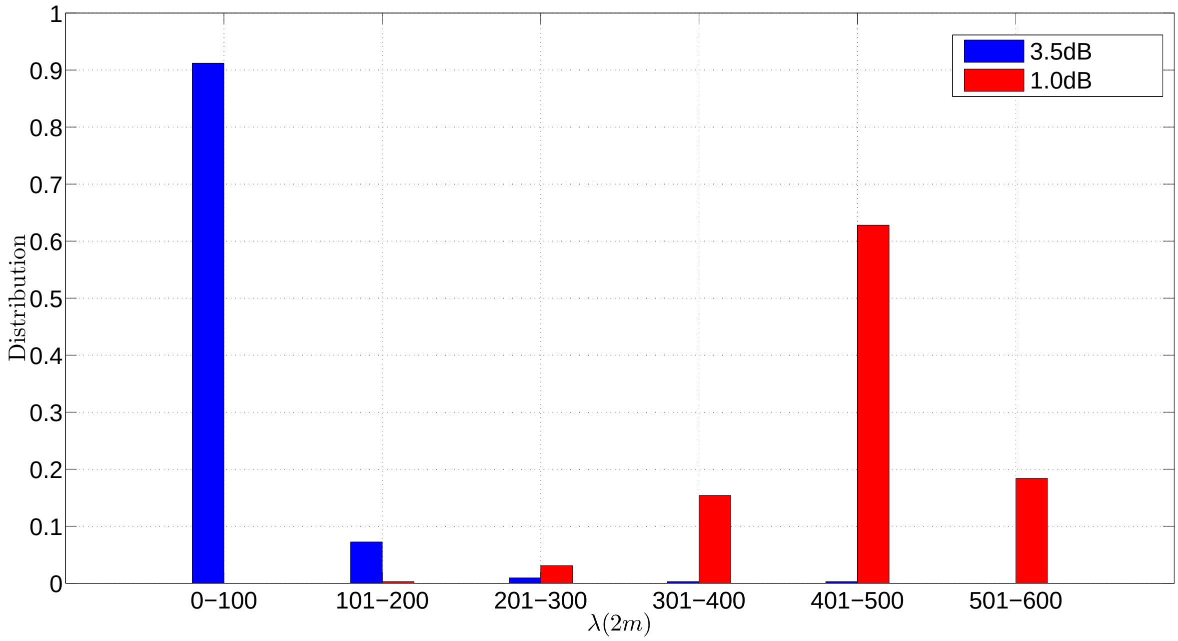


Figure 2.26. Emprical determination of μ value by observing $\lambda(2m)$ ($N = 1024$ and $R = 0.5$).

2.4.4.7. Adaptive minLLR Early Stopping Criterion

Adaptive minLLR method is needed because β limit is not valid for high SNR region especially higher than $3.0dB$. Higher β can not be use for lower SNR values, because it will increase average iteration number unnecessarily. For these reasons, β needs to be able to altered dynamically which requires to add a channel condition estimation method to the algorithm. In [46] a channel condition estimator is proposed based on Hamming distance or amount of different bits (λ) between $\hat{x}G$ and $\hat{u}$. As mentioned in G-Matrix ESC section, when $\hat{x}G = \hat{u}$ meaning that $\lambda = 0$ suggests $\hat{u}$ is a valid output. This means after a predetermined number of iterations $(2m)$ λ can be used as SNR estimator whether if it is under threshold value (μ) or not. This determination is also made empirically with a test that results are illustrated in Fig. 2.26. As it can be seen from the figure if μ value is between $100 \sim 200$ we may say the received package has high SNR. λ is measured at $2m^{th}$ iteration because LLRs propagation inside the decoder requires that. Block scheme of adaptive minLLR method is same with minLLR, so only its algorithm is given with Algorithm 4.

Algorithm 4 Iterative SMS BP (n,k) Polar Code Decoder with adaptive minLLR ESC:

1: **procedure** INITIALIZATION($LLR(r_i), Frozen$)
2:     **while** $t < max\_iter$ **do** ▷ Fill $R^t_{i,j}$ and $L^t_{i,j}$ with initial values.
3:         **if** $(j == 1)\&(i \in Frozen)$ **then**
4:             $R^t_{i,1} = \infty$ ▷ Frozen bits filled with high LLR values.
5:         **else if** $(j == m+1)$ **then**
6:             $L^t_{i,m+1} = LLR(r_i)$ ▷ Channel output LLRs loaded.
7:         **else**
8:             $R^0_{i,j} = L^0_{i,j} = 0$
9: **procedure** ITERATION($Initials, max\_iter, s$)
10:     **while** $t < max\_iter$ **do**
11:         **for** $i = 1$ to $i = N/2$ **do** ▷ $N/2$ PEs.
12:             **for** $j = 1$ to $j = m+1$ **do** ▷ m Layers.
13:                 Update LLR values according to Eqn. (2.36).
14:         **if** $t = 2m$ **then**
15:             Update $\hat{u}$ and $\hat{x}$.
16:             Calculate Hamming distance $\lambda(2m)$ between $\hat{u}G$ and $\hat{x}$.
17:         **if** $(t \geq 2m)\&(\lambda(2m) < \mu)$ **then**
18:             High SNR $\beta = 9.5$.
19:             $min\{|R^t_{i,1} + L^t_{i,1}|\}$ ▷ Find minimum $|LLR|$ among output LLRs.
20:             **if** $minLLR \geq \beta$ **then**
21:                 $Output = \hat{u}$ ▷ Decoding assumed successful and output is $\hat{u}$.
22:             **else**
23:                 Jump to line 31.
24:         **else if** $(t \geq 2m)\&(\lambda(2m) > \mu)$ **then**
25:             Low SNR $\beta = 2.5$.
26:             $min\{|R^t_{i,1} + L^t_{i,1}|\}$ ▷ Find minimum $|LLR|$ among output LLRs.
27:             **if** $minLLR \geq \beta$ **then**
28:                 $Output = \hat{u}$ ▷ Decoding assumed successful and output is $\hat{u}$.
29:             **else**
30:                 Jump to line 31.
31:         $t = t + 1$ ▷ Next iteration.
32:     **Output:** $\hat{u} = (\hat{u}_1, \hat{u}_2 ... \hat{u}_N)$

2.5. Brief Introduction for LT Codes and Early Stopping Criteria

Luby transform (LT) and Raptor codes are members of rateless codes family which are originally designed for the BEC. Due to their capacity-approaching and unique rateless properties, there has been a particular interest in using these codes over noisy channels [53, 54]. Message-passing algorithms such as belief propagation (BP) are used for decoding of rateless codes. This iterative decoder uses a pre-set fixed iteration number in order to stop decoding. However, BP mostly converges to original data at an early stage of decoding. Since the decoding continues until pre-set fixed iteration number, decoder performs redundant processes which cause high computational complexity, decoding latency and energy dissipation. To avoid the aforementioned negations, decoder should be supported by an early termination mechanism to detect convergence and stop decoding.

In literature, there are some ESCs based on check-sum satisfaction ratio (CSR) for rateless codes [55]-[56]. CSR is a common success criterion for BP decoding algorithm to observe whether message estimation satisfies constraints imposed by check nodes. Iterative BP decoding algorithm is performed through log-likelihood ratio (LLR) message-passing between nodes. At the end of each iteration CSR decides output bits, re-encodes them and compare with input bits to determine successful convergence. If difference amount of this comparison is less than a pre-determined user threshold, decoding process is terminated.

2.5.1. LT Encoding

Encoding process of LT code requires a predetermined degree distribution which is one of the main parameters with direct effect on error rate performance. A degree distribution can be represented as in Eqn. (2.42), where $d \in \{d_1, d_2, ..., d_n\}$ represents the degree and $P \in \{P_{d_1}, P_{d_2}, ..., P_{d_n}\}(\sum_{i=1}^{n} P_{d_i} = 1)$ represents distribution for particular degree. Encoding for determined $\Omega(x)$ proceeds as follows:

- Determination of degree d according to P from $\Omega(x)$.
- Uniform selection of bit with amount of degree d and XOR with each other.

$$\Omega(x) = P_{d_1} x^{d_1} \oplus P_{d_2} x^{d_2} \oplus ... \oplus P_{d_n} x^{d_n} \tag{2.42}$$

2.5.2. BP Decoder for LT Code

The graphical representation (Tanner graph) of LT codes (see Fig. 2.27) contains two types of nodes, check-node (CN) and variable-node (VN). BP decoding algorithm is performed through LLR message-passing between these CNs and VNs iteratively. After running LT decoder for a pre-set fixed iteration amount, decision process is done and decoding is completed [53, 54]. The updating equations of CN and VN in LT BP decoder are given in Eqn. (2.43) and Eqn. (2.44), respectively. In these equations, m_c stands for LLR values of the codewords come from channel and is directly sent to corresponding CN c, $m_{c\rightarrow v}$ and $m_{v\rightarrow c}$ represent the outgoing LLR messages from the CN c to VN v and vice versa. $tanh(\cdot)$ and $tanh^{-1}(\cdot)$ represent hyperbolic tangent and inverse hyperbolic tangent operations, respectively. Also, iteration index is denoted by superscript l.

$$\begin{aligned} m_{c\rightarrow v}^{(l)} =& sign\left(m_c \prod_{v'\neq v} m_{v'\rightarrow c}^{(l)}\right) \\ & \times\ 2tanh^{-1}\left[tanh\left(\frac{|m_c|}{2}\right)\prod_{v'\neq v} tanh\left(\frac{|m_{v'\rightarrow c}^{(l)}|}{2}\right)\right] \end{aligned} \tag{2.43}$$

$$m_{v\rightarrow c}^{(l+1)} = \sum_{c'\neq c} m_{c'\rightarrow v}^{(l)} \tag{2.44}$$

Hard-decision process of BP decoder is given as follows,

$$m_v = m_{c\rightarrow v}^{(l)} + m_{v\rightarrow c}^{(l+1)}, \qquad \hat{m}_v = \begin{cases} 1, & m_v \geq 0 \\ 0, & m_v < 0 \end{cases} \tag{2.45}$$

where, $\hat{m}_v$ represents hard decided value for corresponding VN v. Decoding process of LT codes can be given as in Algorithm 5.

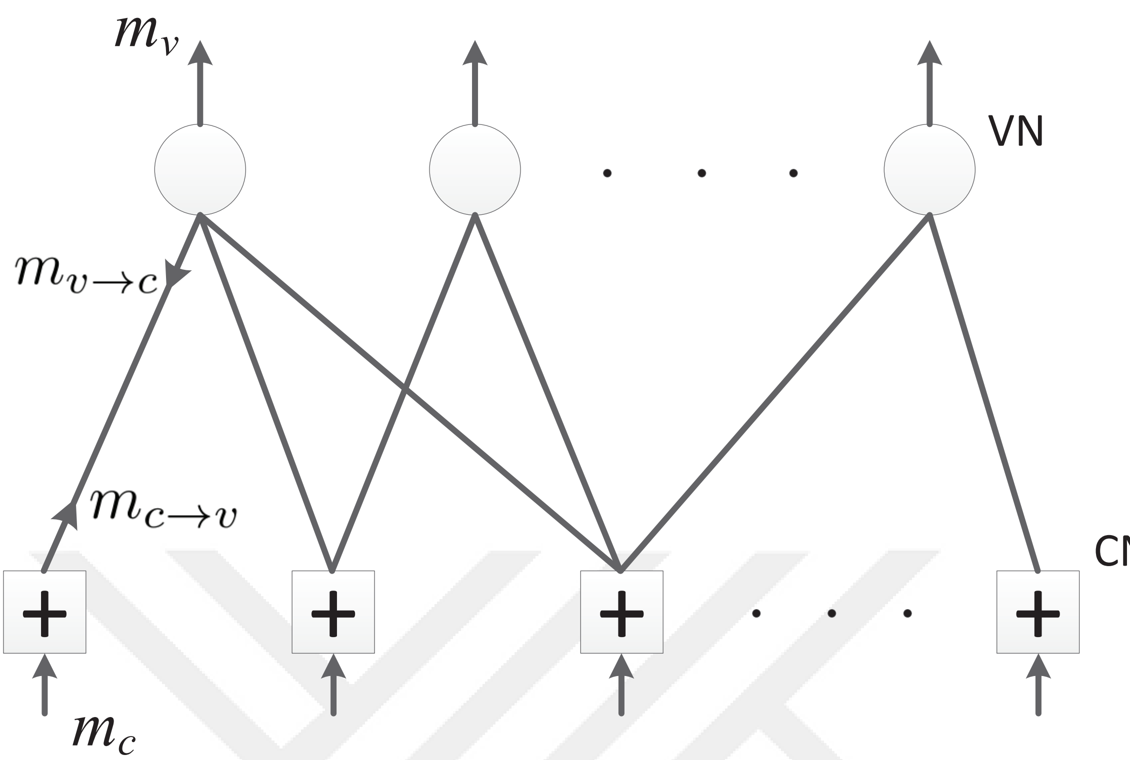


Figure 2.27. Tanner graph representation of LT code decoding.

Algorithm 5 LT BP Decoder:

Initialization
    Calculate m_c;
    Set $m^{(0)}_{c\to v}$ and $m^{(0)}_{v\to c}$ messages to zero, $l = 0$;
end Initialization
while $l < max\_iter$ **do**
    CN update(); ▷ Eqn. (2.43) is performed.
    VN update(); ▷ Eqn. (2.44) is performed.
    $l = l+1$; ▷ Next iteration.
end while
Decision(); ▷ Eqn. (2.45) is performed.

2.5.3. CSR Early Stopping Criterion

A common criterion [55–58] for early termination of rateless decoding is observing if the estimated messages $\hat{m}_v$ satisfy the constraints imposed by CNs [55, 56]. The criterion controls whether Eqn. (2.46) is equal to zero for all CNs,

$$\hat{m}_c \oplus \left(\bigoplus_v \hat{m}_v \right) \tag{2.46}$$

where $\hat{m}_c$ stands for hard decision of m_c messages, $\oplus$ represents modulo-2 addition and $\bigoplus$ denotes the summation operator for modulo-2 addition. In Eqn. (2.46) parenthetical expression represents re-encoding process and rest of it represents compare process. After performing the equation, CSR test is calculated by $\mu_{CSR} = s^{(l)}/N_{CN}$, where $s^{(l)}$ is number of satisfied CNs at decoding iteration l and N_{CN} is total number of CNs. The test is satisfied when inequality $\mu_{CSR} \geq \Gamma_{CSR}$ is correct, where Γ_{CSR} is a user-defined threshold. This method is known as CSR ESC. LT BP decoder with CSR is presented in Algorithm 6.

Algorithm 6 LT BP Decoder with CSR Method:

```
Initialization
    Calculate m_c;
    Set m^(0)_{c→v} and m^(0)_{v→c} messages to zero, l = 0;
end Initialization
while (l < max_iter) and (Γ_LC is not satisfied) do
    CN update();                          ▷ Eqn. (2.43) is performed.
    VN update();                          ▷ Eqn. (2.44) is performed.
    Decision();                           ▷ Eqn. (2.45) is performed.
    Calculate CSR and ΔCSR;
    l = l + 1;                            ▷ Next iteration.
end while
```

In the algorithm, the difference between CSR values of two consecutive iterations denoted as ΔCSR. If ΔCSR has a value of “0” for Γ_{LC} amount of consecutive iterations, decoding is terminated [57]. Γ_{LC} is a user-defined integer value and means last control test.

3. CONTRIBUTIONS AND FINDINGS

3.1. A Simplified Early Stopping Criterion for Belief-Propagation Polar Code Decoders

Channel polarization is the fundamental of polar code [9]. After a bit sequence is encoded with polar code, error probabilities of some bits in the sequence increase while error probabilities of others decrease. Depending on channel condition and desired code rate, bits polarized to lower error probabilities are used for transmitting information while others are filled with fixed data. These bits are called information (non-frozen) bits and frozen bits, respectively. However, as we mentioned in previous chapter some of the bit channels remain unpolarized called semi-polarized bits. Our hypothesis based on an idea that observation of a cluster of information bits that are polarized to the highest error probabilities in information bits, which we may call them as semi-polarized information bits, may be enough to detect successful decoding in order to stop the iterations. The reason we anticipate this approach is that, the cluster mentioned includes the bits which converge later then the other bits. We call this cluster as the worst of information bits (WIB). While performing BP decoding algorithm, if WIB are detected as successfully decoded, rest of the bits can be assumed as decoded with a high probability, too. Therefore, we only need to check the WIB to stop iterations which decreases the computational complexity of early stopping section. We call this method as WIB early stopping criterion.

In addition to using only a cluster of bits instead of using all bits, we use a completely different method to detect successful decoding with WIB. We only check sign alterations of log-likelihood ratio (LLR) values of bits in the cluster. Furthermore, we engage the proposed early stopping criterion at an approximate iteration number by observing the minimum decodable case for particular *SNR* value. This also provides additional reduction in complexity. Simulation results show that proposed method achieves significant complexity reduction without any performance loss.

As mentioned above, we only observe a cluster of information bits which we call WIB to detect successful decoding. It can be expected that information bits transmitting through higher error probability channels require more decoding iterations compared to other bits. By using this idea, we determine the WIB according to construction method used for polar codes. As stated in [24] all polar code construction methods could construct equally good polar codes if proper design *SNR* chosen. We performed three methods, recursive Bhattacharyya bound, Gaussian approximation (GA) and density evolution (DE), studied in [24] for polar codes construction and WIB determination. We come to the same conclusion as in [24]. Then we chose Bhattacharyya bound for polar codes construction which has the lowest complexity. Bhattacharyya parameter Z(W) for polar codes introduced in [9] as an upper bound on the probability of maximum likelihood (ML) decision error, where W is transition probability of binary-input discrete memoryless channel. We calculate Bhattacharyya parameter for binary-input additive white Gaussian noise channel (BI-AWGNC) as in [24]. Here, we define a new parameter called proportion of average Bhattacharyya values (PoB) using recursive Bhattacharyya bound method to provide a better understanding for our criterion as given in Eqn. (3.1).

$$PoB = \frac{\frac{1}{n_{WIB}} \sum_{l \in WIB} Z\left(\sigma_l^2\right)}{\frac{1}{k-n_{WIB}} \sum_{l \notin WIB} Z\left(\sigma_l^2\right)} \tag{3.1}$$

Here n_{WIB} is the amount of information bits observing for early stopping algorithm, l is the bit index indicates only information bits, k is the amount of whole information bits and σ^2 is the variance of Gaussian noise. PoB values are calculated for various n_{WIB}, SNR values and code rates over BI-AWGNC. Results are presented in Tables between (3.1 to 3.9).

These values tell us that average error probability of WIB are much higher than average error probability of rest information bits with a drastic increase especially for $N/8$ which is equal to 128 for (1024, 512) polar code. So, it is easy to see that WIB require more iteration for successful decoding. As seen in Tables (3.1 to 3.9) when n_{WIB} increases, PoB increases drastically which means higher n_{WIB} value increases the successful decoding detection probability. To keep computational complexity as low as possible, we choose n_{WIB} value high enough for successful detection of decoding without any performance loss according to physical channel condition. SNR values in Tables (3.1 to 3.9) are chosen according to code rates where polar code performs optimum.

Table 3.1. PoB for BI-AWGNC with various SNR and n_{WIB} $(N = 512, R = 0.33)$

	n_{WIB}			
SNR(dB)	***8***	***16***	***32***	***64***
-1	$3.7 * 10^1$	$9.9 * 10^1$	$7.1 * 10^2$	$1.8 * 10^5$
0	$8.0 * 10^1$	$2.7 * 10^1$	$4.6 * 10^3$	$2.6 * 10^6$
1	$1.2 * 10^2$	$4.8 * 10^2$	$1.4 * 10^4$	$1.1 * 10^8$
2	$1.7 * 10^2$	$1.1 * 10^3$	$4.3 * 10^4$	$4.2 * 10^{11}$

Table 3.2. PoB for BI-AWGNC with various SNR and n_{WIB} $(N = 512, R = 0.5)$

	n_{WIB}			
SNR(dB)	***8***	***16***	***32***	***64***
0	$3.0*10^1$	$4.5*10^1$	$1.2*10^2$	$1.9*10^3$
1	$4.8*10^1$	$9.2*10^1$	$3.6*10^2$	$2.1*10^4$
2	$7.2*10^1$	$1.5*10^2$	$9.5*10^2$	$8.4*10^5$
3	$7.5*10^1$	$1.9*10^2$	$5.2*10^3$	$1.0*10^8$

Table 3.3. PoB for BI-AWGNC with various SNR and n_{WIB} $(N = 512, R = 0.66)$

	n_{WIB}			
SNR(dB)	***8***	***16***	***32***	***64***
3	$6.6*10^1$	$1.1*10^2$	$7.2*10^2$	$4.4*10^4$
4	$1.3*10^2$	$3.4*10^2$	$3.3*10^3$	$4.3*10^5$
5	$2.1*10^2$	$7.7*10^2$	$1.1*10^4$	$1.8*10^6$
6	$2.4*10^2$	$1.1*10^3$	$1.4*10^4$	$6.6*10^6$

Table 3.4. PoB for BI-AWGNC with various SNR and n_{WIB} $(N = 1024, R = 0.33)$

	n_{WIB}			
SNR(dB)	***16***	***32***	***64***	***128***
-1	$5.5*10^1$	$1.8*10^2$	$2.7*10^3$	$5.0*10^7$
0	$1.0*10^2$	$3.1*10^2$	$1.2*10^4$	$2.2*10^{10}$
1	$2.6*10^2$	$1.3*10^3$	$2.6*10^5$	$2.6*10^{13}$
2	$8.7*10^2$	$8.4*10^4$	$2.7*10^7$	$5.0*10^{17}$

As a result of this search process n_{WIB} value is chosen $N/8$ for short block polar code e.g. $N = (512, 1024, 2048)$. As one can anticipate n_{WIB} can be lower than $N/8$ if block length is increased.

We should remind that these values are calculated with different design SNR values. If a fixed design SNR value is chosen, the best one is 0 dB for Bhattacharyya bound as stated in [24], results are as in Table 3.10. It is evident from Table 3.10 that when N increased needed amount of n_{WIB} should decrease. This result has also been confirmed by our simulation studies.

Table 3.5. PoB for BI-AWGNC with various SNR and n_{WIB} $(N = 1024, R = 0.5)$

	n_{WIB}			
SNR(dB)	***16***	***32***	***64***	***128***
0	$4.3*10^1$	$7.4*10^1$	$3.1*10^2$	$2.6*10^4$
1	$7.9*10^1$	$1.8*10^2$	$1.5*10^3$	$1.0*10^6$
2	$2.1*10^2$	$6.6*10^2$	$1.5*10^4$	$7.2*10^7$
3	$5.1*10^2$	$2.2*10^3$	$1.5*10^5$	$1.4*10^9$

Table 3.6. PoB for BI-AWGNC with various SNR and n_{WIB} $(N = 1024, R = 0.66)$

	n_{WIB}			
SNR(dB)	***16***	***32***	***64***	***128***
3	$1.2*10^2$	$3.1*10^2$	$2.6*10^3$	$6.3*10^5$
4	$2.1*10^2$	$6.4*10^2$	$9.8*10^3$	$5.2*10^7$
5	$2.3*10^2$	$7.6*10^2$	$6.4*10^4$	$2.1*10^{11}$
6	$3.8*10^2$	$2.3*10^3$	$1.2*10^6$	$2.8*10^{16}$

Table 3.7. PoB for BI-AWGNC with various SNR and n_{WIB} $(N = 2048, R = 0.33)$

	n_{WIB}			
SNR(dB)	***32***	***64***	***128***	***256***
-1	$1.2*10^2$	$5.9*10^2$	$7.3*10^4$	$3.0*10^{10}$
0	$3.6*10^2$	$5.8*10^3$	$8.1*10^6$	$3.7*10^{13}$
1	$1.3*10^3$	$8.2*10^4$	$2.1*10^8$	$8.3*10^{16}$
2	$2.9*10^3$	$4.1*10^5$	$4.0*10^9$	$1.6*10^{24}$

After observing the right amount of frozen bits to consider, early stopping detection is designed to have less complexity also. In Eqn. (3.3) we only observe sign alterations of last nodes LLR values calculated with Eqn. (3.2) for last M iterations which is a bit wise logical operation.

$$\widehat{u}_i^t = sign\left(R_{i,1}^t + L_{i,1}^t\right) \tag{3.2}$$

Table 3.8. PoB for BI-AWGNC with various SNR and n_{WIB} $(N = 2048, R = 0.5)$

	n_{WIB}			
SNR(dB)	***32***	***64***	***128***	***256***
0	$6.9 * 10^1$	$1.5 * 10^2$	$1.4 * 10^3$	$9.1 * 10^5$
1	$1.3 * 10^2$	$5.0 * 10^2$	$1.6 * 10^4$	$1.1 * 10^8$
2	$2.7 * 10^2$	$1.9 * 10^3$	$1.8 * 10^5$	$1.1 * 10^{11}$
3	$3.8 * 10^2$	$3.5 * 10^3$	$1.8 * 10^6$	$3.5 * 10^{15}$

Table 3.9. PoB for BI-AWGNC with various SNR and n_{WIB} $(N = 2048, R = 0.66)$

	n_{WIB}			
SNR(dB)	***32***	***64***	***128***	***256***
3	$2.1 * 10^2$	$7.1 * 10^2$	$2.0 * 10^4$	$3.7 * 10^8$
4	$5.3 * 10^2$	$4.3 * 10^3$	$1.3 * 10^6$	$2.1 * 10^{11}$
5	$6.6 * 10^3$	$2.2 * 10^5$	$4.2 * 10^8$	$9.6 * 10^{13}$
6	$1.1 * 10^4$	$4.4 * 10^5$	$1.2 * 10^9$	$1.7 * 10^{15}$

$$\sum_{l \in WIB} \sum_{v=t-M+1}^{t} \widehat{u}_l^{v} \oplus \widehat{u}_l^{v-1} \tag{3.3}$$

If the calculation of Eqn. (3.3) is equal to “0” WIB method assumes decoding is successful and stops the iterations. Block diagram of this process is shown in Fig. 3.1. Inside the figure, thin lines represent bit wise *xor* of sign bits while vertical block is an adder for single bits. And, the last adder block length is M which can be “7” at maximum (see Table 3.11). Detailed explanation for single iteration of SMS BP decoding process with WIB early stopping criterion method is presented in Algorithm 7. Here, as mentioned above, n_{WIB} indicates the number of WIB bits and M is the amount of last iterations that sign of WIB remain invariant.

Table 3.10. PoB for BI-AWGNC with $SNR = 0$ dB

Code Length \ Rate	n_{WIB}					
	8	***16***	***32***	***64***	***128***	***256***
$N = 512 \setminus R = 0.33$	$8.0 * 10^1$	$2.7 * 10^1$	$4.6 * 10^3$	$2.6 * 10^6$	–	–
$N = 512 \setminus R = 0.5$	$3.0 * 10^1$	$4.5 * 10^1$	$1.2 * 10^2$	$1.9 * 10^3$	–	–
$N = 512 \setminus R = 0.66$	$1.3 * 10^1$	$1.5 * 10^1$	$2.2 * 10^1$	$7.3 * 10^1$	–	–
$N = 1024 \setminus R = 0.33$	–	$1.0 * 10^2$	$3.1 * 10^2$	$1.2 * 10^4$	$2.2 * 10^{10}$	–
$N = 1024 \setminus R = 0.5$	–	$4.3 * 10^1$	$7.4 * 10^1$	$3.1 * 10^2$	$2.6 * 10^4$	–
$N = 1024 \setminus R = 0.66$	–	$1.5 * 10^1$	$1.8 * 10^1$	$3.0 * 10^1$	$1.6 * 10^2$	–
$N = 2048 \setminus R = 0.33$	–	–	$3.6 * 10^2$	$5.8 * 10^3$	$8.1 * 10^6$	$3.7 * 10^{13}$
$N = 2048 \setminus R = 0.5$	–	–	$6.9 * 10^1$	$1.5 * 10^2$	$1.4 * 10^3$	$9.1 * 10^5$
$N = 2048 \setminus R = 0.66$	–	–	$1.7 * 10^1$	$2.2 * 10^1$	$4.5 * 10^1$	$4.4 * 10^2$

Algorithm 7 Iterative SMS BP (n,k) Polar Code Decoder with Proposed WIB ESC:

procedure INITIALIZATION($LLR(r_i), Frozen$)
  while $t < max\_iter$ **do** ▷ Fill $R^t_{i,j}$ and $L^t_{i,j}$ with initial values.
    if $(j == 1) \& (i \in Frozen)$ **then**
      $R^t_{i,1} = \infty$ ▷ Frozen bits filled with high LLR values.
    else if $(j == m+1)$ **then**
      $L^t_{i,m+1} = LLR(r_i)$ ▷ Channel output LLRs loaded.
    else
      $R^0_{i,j} = L^0_{i,j} = 0$
procedure ITERATION($Initials, max\_iter, s$)
  while $t < max\_iter$ **do**
    for $i = 1$ to $i = N/2$ **do** ▷ $N/2$ PEs.
      for $j = 1$ to $j = m+1$ **do** ▷ m Layers.
        Update LLR values according to Eqn. (2.36).
    if $t > 2m - M$ **then** ▷ Engage WIB method.
      if Eqn. (3.3)$\neq 0$ **then**
        $t = t + 1$ ▷ Next iteration.
      else
        Decoding assumed successful and output is $\hat{u}$.
    else
      $t = t + 1$ ▷ Next iteration.
  Output: $\hat{u} = (\hat{u}_1, \hat{u}_2 \ldots \hat{u}_N)$

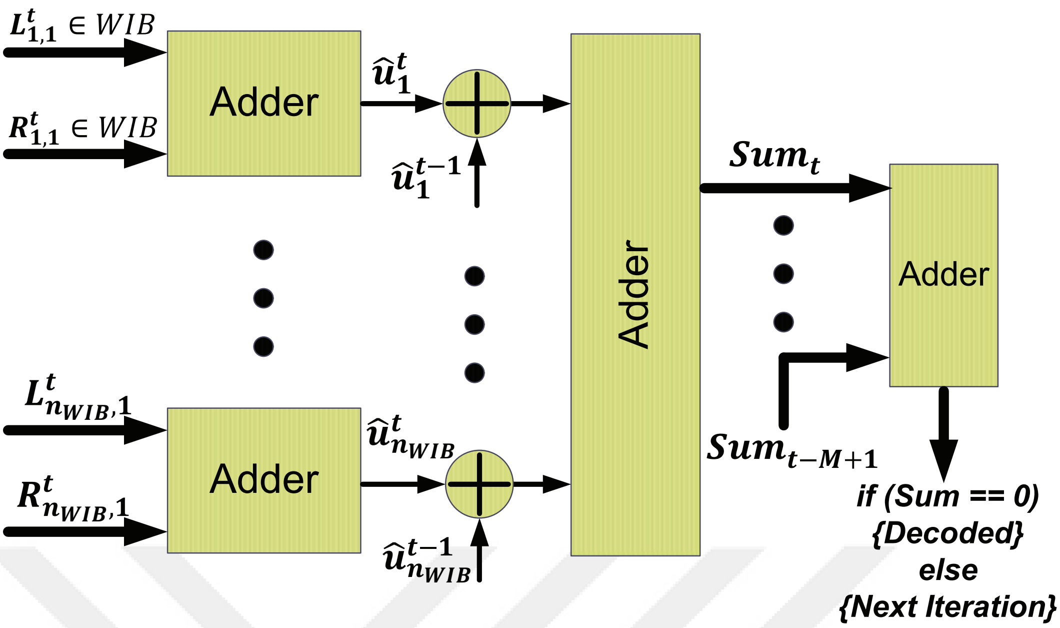


Figure 3.1. Block diagram of proposed WIB early stopping criterion method.

3.1.1. Simulation Results and Complexity Analysis of WIB ESC

For a proper comparison, simulations are performed with similar methodology as in [46]. In the simulation works, we consider BI-AWGNC with various code lengths $(512, 1024, 2048)$ and code rates $(0.33, 0.5, 0.66)$. At the receiver side, we employ SMS BP polar code decoder with scale parameter $s = 0.9375$ and average frame error rate (FER) over 10000 trials.

Fig. 3.2 illustrates the FER-SNR results of SMS BP decoding algorithm for 40 fixed iteration number and proposed WIB early stopping method for $(1024, 512)$ polar code. As it can be seen in Fig. 3.2 when M equals to 7 and n_{WIB} is 128, FER performances are exactly the same as decoding with fixed 40 iterations number. It means that there is no performance loss because of using early stopping method. But for lower M values there are performance degradations. We observe similar situation for various n_{WIB} values. It is easy to see that higher M and n_{WIB} values decrease the possibility of wrong decoding detection. But, it should be pointed that if M is increased by one, it will directly increase average iteration amount at least by one and also making Eqn. (3.3) equal to zero especially at lower SNR region will become harder. Also, higher n_{WIB} values increase the computational complexity of proposed early stopping algorithm. This means unnecessary high M and n_{WIB} values should be avoided to maintain the benefits of WIB method. With this perspective, simulations are performed for various N, coding rate R, M and n_{WIB} values to determine the most suitable pairs. Obtained results are illustrated in Table 3.11 and Table 3.12.

Additionally, other code lengths and rates are illustrated with Fig. 3.3 to Fig. 3.11 for BER-SNR performances. These results clearly shows that there are minor differences between chosen M and n_{WIB} values with smaller ones. However, from FER-SNR point of view this difference is more dominant, so Table 3.12 and Table 3.11 are determined according to FER-SNR results which WIB method does not yield even a single bit difference compared to 40 fixed iteration number. Also as the code length increased amount of n_{WIB} can be decreased according to both Table 3.1 to Table 3.10 and Fig. 3.3 to Fig 3.11.

Table 3.11. M values for various SNR, code lengths and code rates for $n_{WIB} = N/8$

N	512/1024/2048		
R	0.33	0.5	0.66
SNR(dB)	$\boldsymbol{M}$	$\boldsymbol{M}$	$\boldsymbol{M}$
−0.5	2/2/2	–	–
0.0	3/3/3	–	–
0.5	3/3/3	–	–
1.0	5/5/5	–	–
1.5	5/5/5	2/2/2	–
2.0	–	5/3/3	–
2.5	–	5/5/4	–
3.0	–	5/5/4	2/2/2
3.5	–	7/7/7	5/5/3
4.0	–	–	5/5/5
4.5	–	–	5/7/6
5.0	–	–	6/7/6
5.5	–	–	7/7/7

Table 3.12. Average Iteration Amounts of Stopping Criteria for (1024, 512) Polar Code and Iteration Reductions According to 40 Fixed Iterations SMS BP Decoder

Early Stopping Criterion	**G-Matrix / min LLR**		**The Proposed Adaptive WIB / Fixed WIB**			
SNR(dB)	*Average Iteration*	*Iteration Reduction (%)*	*Average Iteration*	*Iteration Reduction (%)*	M	n_{WIB}
1.5	39.6 / 39.9	1.0 / 0.2	39.7 / 39.9	0.8 / 0.2	2 / 5	128 / 128
2.0	36.5 / 38.4	8.7 / 4.0	38.0 / 39.1	5.0 / 2.3	3 / 5	128 / 128
2.5	30.8 / 35.7	23.0 / 10.7	35.3 / 35.3	11.8 / 11.8	5 / 5	128 / 128
3.0	26.1 / 33.9	26.1 / 15.2	28.4 / 28.4	29.0 / 29.0	5 / 5	128 / 128
3.5	23.0 / 30.7	42.5 / 23.2	28.4 / 26.7	29.0 / 33.3	7 / 5	128 / 128

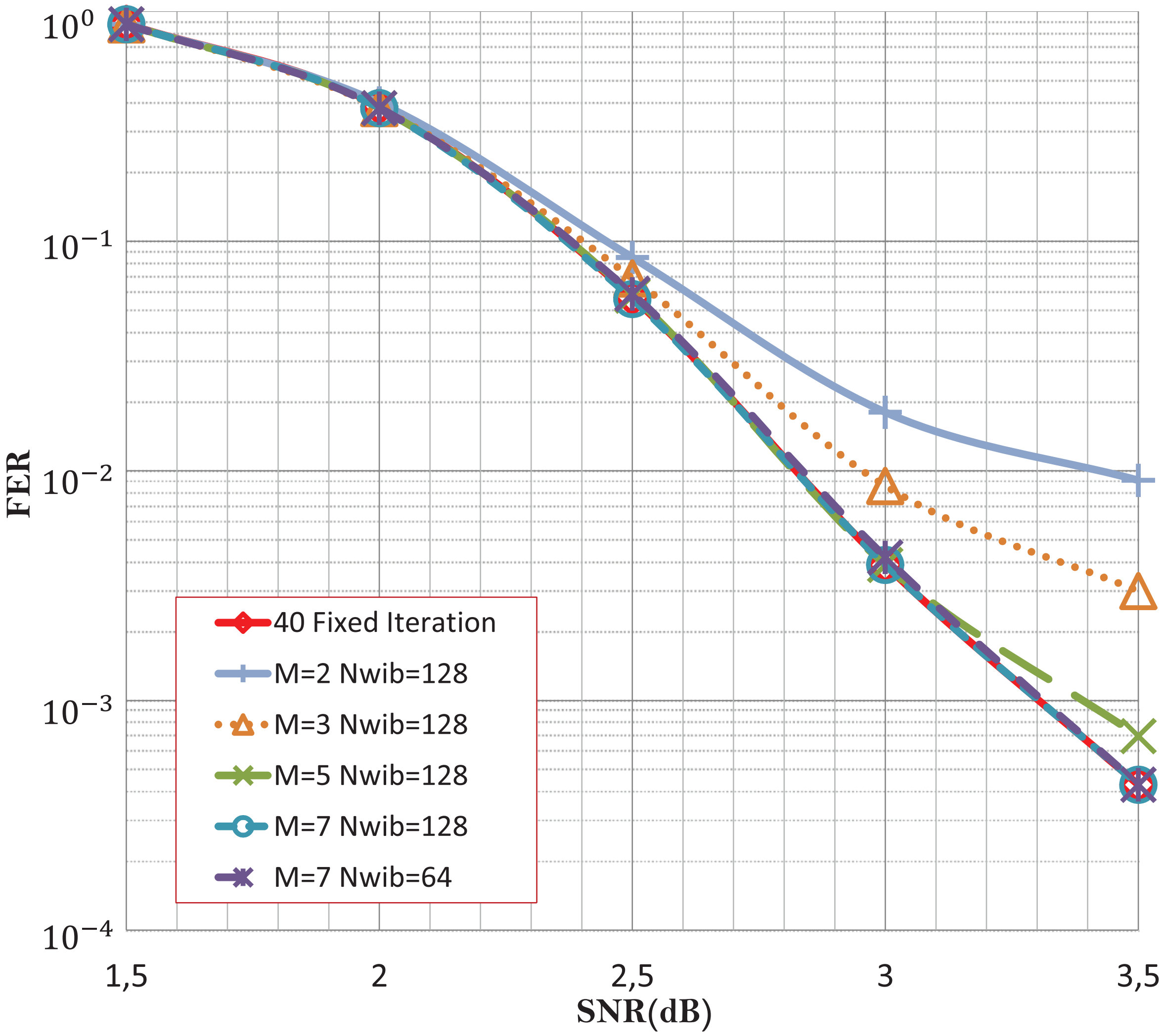


Figure 3.2. FER-SNR results of SMS BP $(1024, 512)$ polar code decoder with proposed WIB early stopping criterion method.

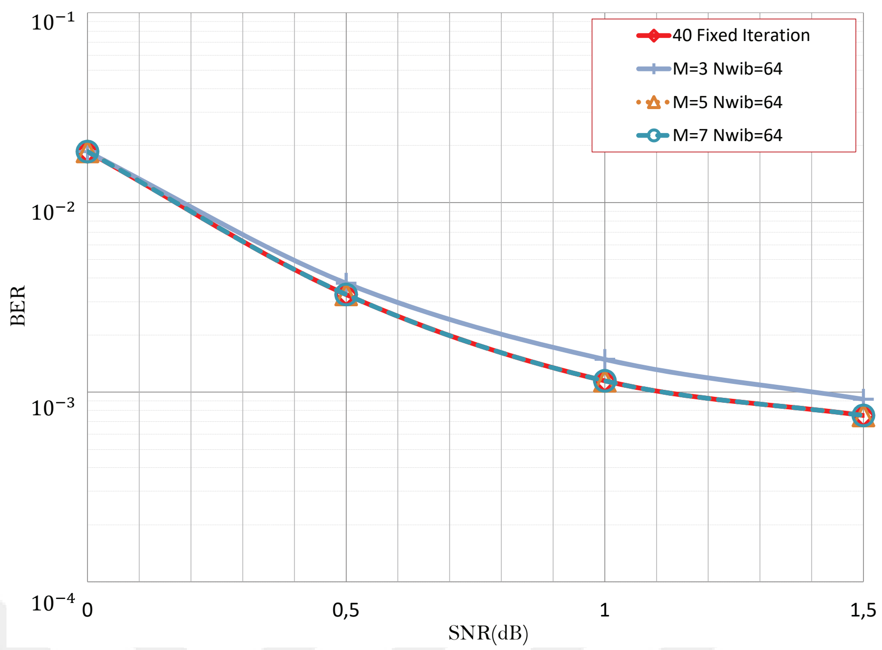


Figure 3.3. BER-SNR results of SMS BP $(512, 169)$ polar code decoder with proposed WIB early stopping criterion method.

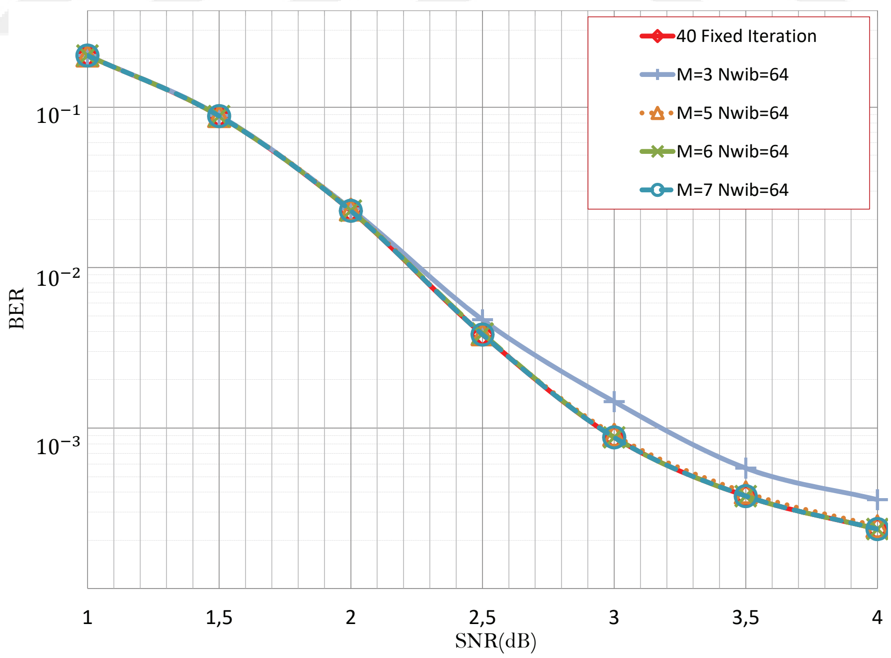


Figure 3.4. BER-SNR results of SMS BP $(512, 256)$ polar code decoder with proposed WIB early stopping criterion method.

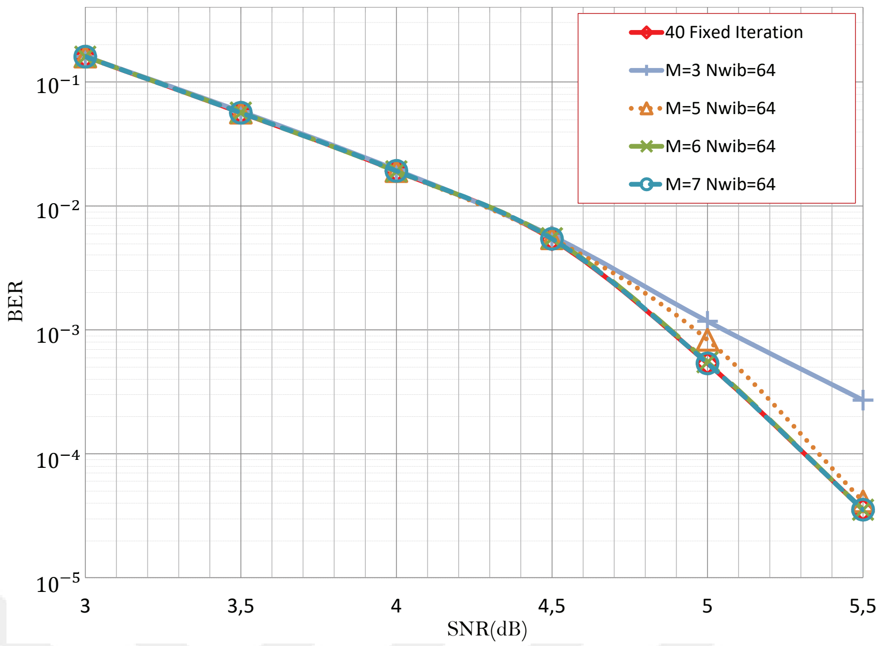


Figure 3.5. BER-SNR results of SMS BP $(512,338)$ polar code decoder with proposed WIB early stopping criterion method.

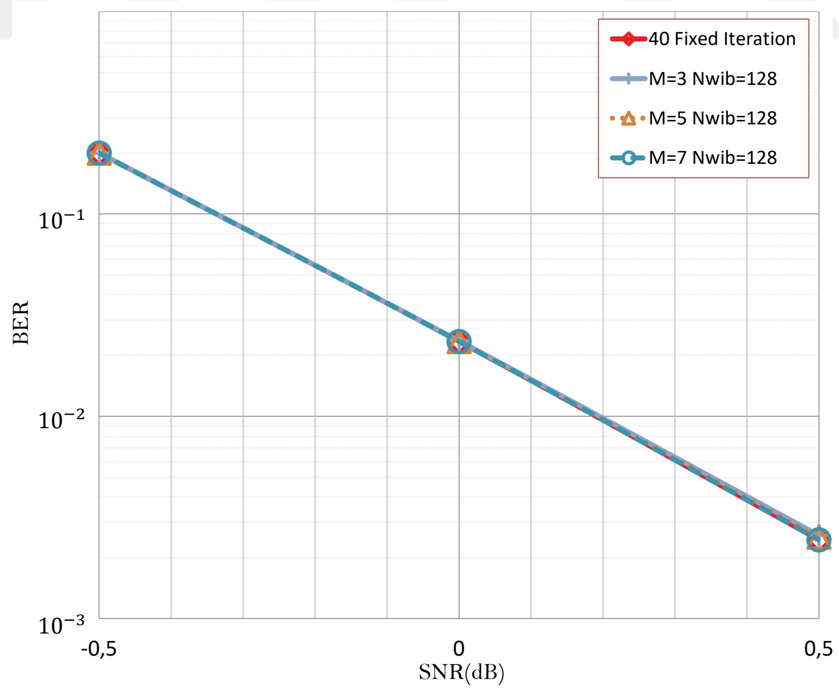


Figure 3.6. BER-SNR results of SMS BP $(1024,338)$ polar code decoder with proposed WIB early stopping criterion method.

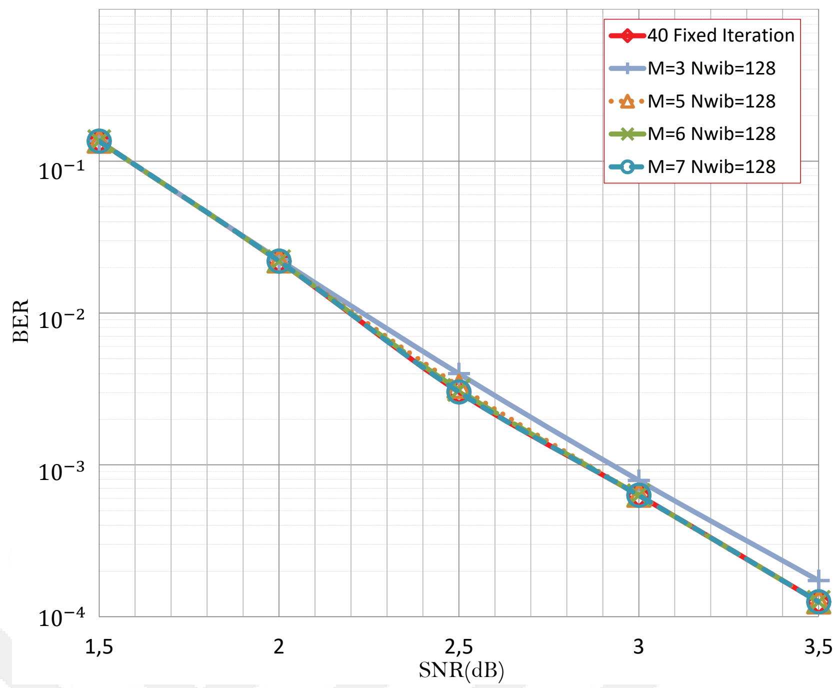


Figure 3.7. BER-SNR results of SMS BP $(1024, 512)$ polar code decoder with proposed WIB early stopping criterion method.

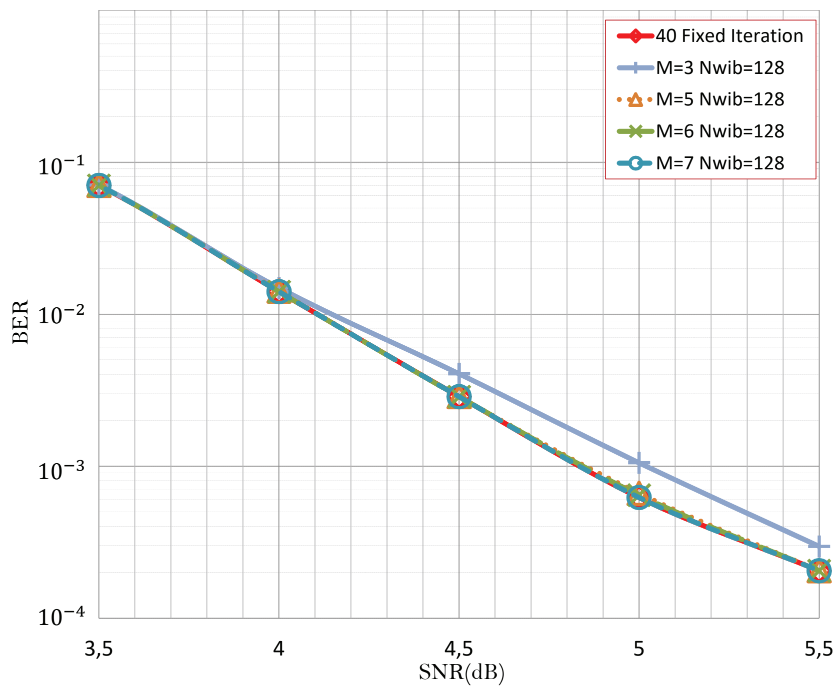


Figure 3.8. BER-SNR results of SMS BP $(1024, 676)$ polar code decoder with proposed WIB early stopping criterion method.

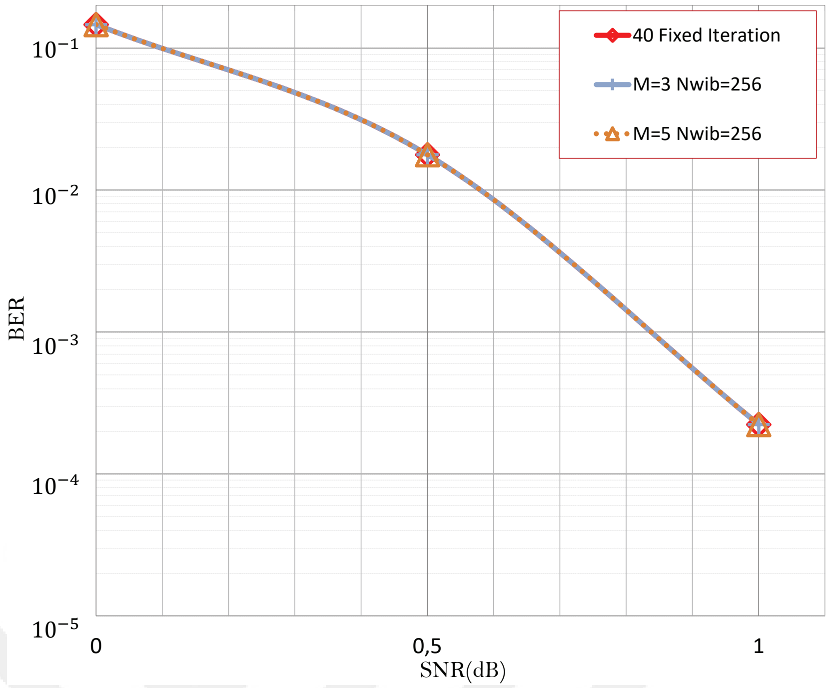


Figure 3.9. BER-SNR results of SMS BP $(2048,676)$ polar code decoder with proposed WIB early stopping criterion method.

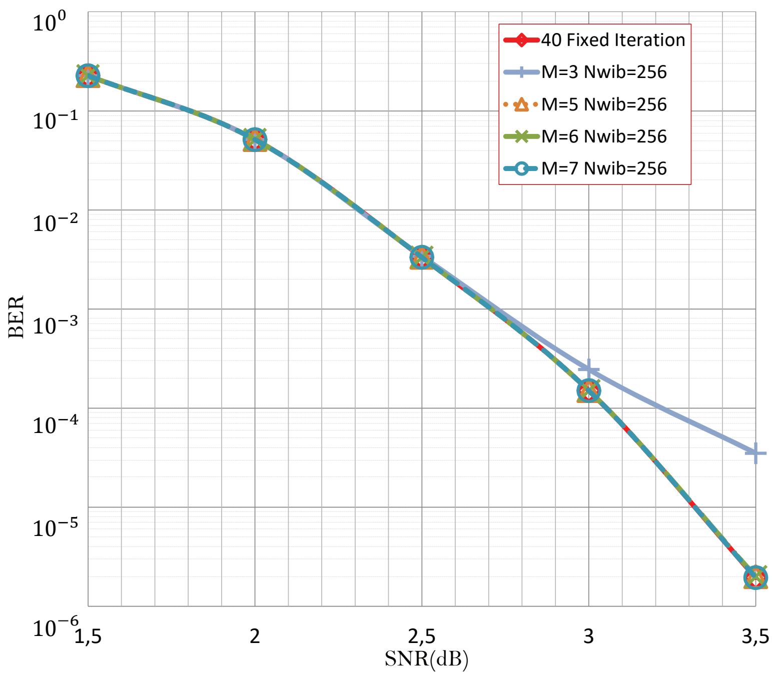


Figure 3.10. BER-SNR results of SMS BP $(2048,1024)$ polar code decoder with proposed WIB early stopping criterion method.

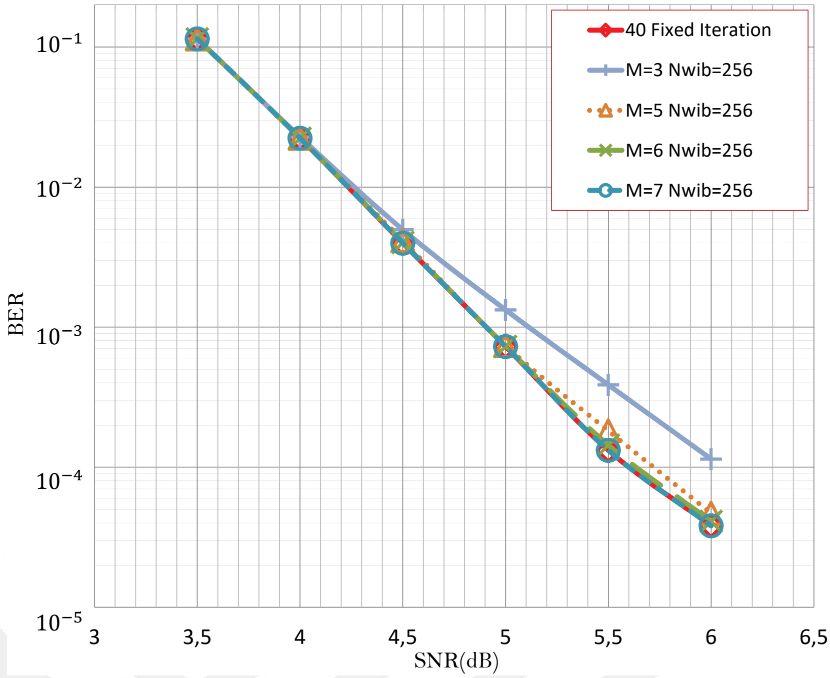


Figure 3.11. BER-SNR results of SMS BP $(2048, 1352)$ polar code decoder with proposed WIB early stopping criterion method.

The other important issue is the complexity of ESC which directly effect the throughput performance of entire decoder. As can be seen from Fig. 2.23, 2.25 and 3.1 WIB is the most efficient ESC from hardware point of view. Complexities of each method summarized in Table 3.13. Although, in WIB column the number of addition operations seems to be $2N$, half

Table 3.13. Complexities of Early Stopping Criteria for Single Iteration

	G-matrix	***minLLR***	***WIB***
Add	$2N$	N	$M+2N/8$
Compare	$3N$	$2N$	–
XOR	$Nlog_2N$	–	$N/8$

of them can be done by logical *OR* operations which provides more reduction for hardware and increases the speed of structure.

3.1.2. Modified WIB ESC

In this section we modified the WIB ESC for more complexity reduction. As suggested in [46] last layers LLR values are obtained by summing left and right LLR values to determine

$\hat{u}_i$ as with Eqn. (3.2). However, for WIB cluster this summation also seems unnecessary to detect early stopping. In Fig. 3.12 block scheme only left LLR values are observed and bit wise logic *OR* operations are performed resulted exactly the same with Fig. 3.2. As a result complexity table become as in Table 3.14. Some of these studies are published with [59] excluding simplified WIB.

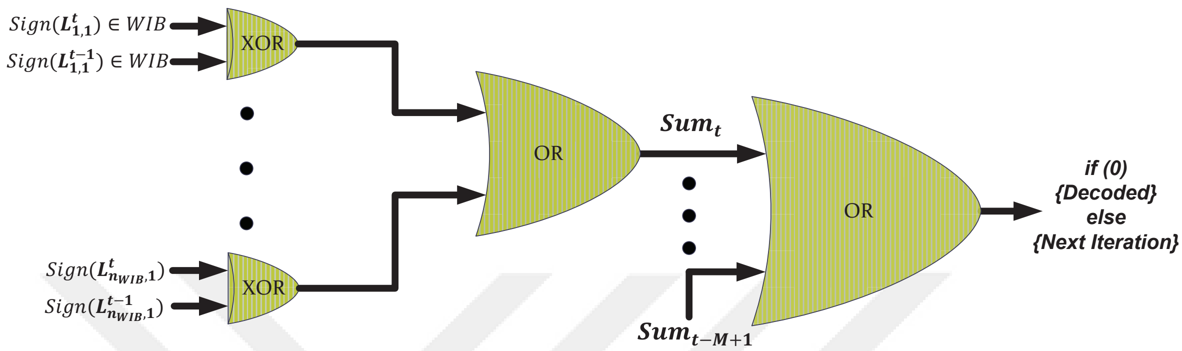


Figure 3.12. Block scheme of simplified WIB ESC.

Table 3.14. Complexities of Early Stopping Criteria for Single Iteration with simplified WIB ESC

	G-matrix	*minLLR*	*Simp. WIB*
Add	$2N$	N	–
Compare	$3N$	$2N$	–
XOR	$NlogN$	–	$N/8$
OR	–	–	$M+N/8$

3.2. WIB Aided minLLR Early Stopping Criterion for Belief-Propagation Based Polar Code Decoders

This method based on both WIB and minLLR ESCs [60]. According to previous study in [59] WIB covers the information bits with the highest error probabilities therefore it makes sense to look up minimum LLR value inside the WIB cluster instead of entire block. To test validity of the idea we provide Table 3.15 which gives us the probabilities of minimum LLR values being outside WIB cluster and the average difference from actual minLLR value when minimum LLR is outside the WIB.

Table 3.15. Probabilities of Minimum LLR Being Outside of WIB and Average LLR Value Difference when Minimum LLR is Outside WIB

	Probability of minimum LLR being outside of WIB						**Average LLR value difference**					
WIB	$N/16$			$N/8$			$N/16$			$N/8$		
RATE ***SNR(dB)***	1/3	1/2	2/3	1/3	1/2	2/3	1/3	1/2	2/3	1/3	1/2	2/3
–0.5	0.20	–	–	0	–	–	1.88	–	–	0	–	–
0.0	0.66	–	–	0	–	–	2.00	–	–	0	–	–
0.5	0.86	–	–	0	–	–	2.11	–	–	0	–	–
1.0	0.87	–	–	0	–	–	4.59	–	–	0	–	–
1.5	0.97	0.14	–	0	0	–	4.71	1.60	–	0	0	–
2.0	–	0.62	–	–	0	–	–	1.74	–	–	0	–
2.5	–	0.89	–	–	0	–	–	3.33	–	–	0	–
3.0	–	0.93	0.09	–	0	0.02	–	3.74	1.35	–	0	0.43
3.5	–	0.96	0.19	–	0	0.06	–	4.19	1.39	–	0	0.52
4.0	–	0.98	0.62	–	0	0.26	–	4.53	1.52	–	0	0.59
4.5	–	–	0.86	–	–	0.39	–	–	3.17	–	–	0.65
5.0	–	–	0.49	–	–	0.14	–	–	3.43	–	–	1.14
5.5	–	–	0.80	–	–	0.39	–	–	3.73	–	–	1.20
6.0	–	–	0.94	–	–	0.53	–	–	3.97	–	–	1.24

3.2.1. Simulation Results of WIB Aided minLLR ESC

Table 3.15 along with Figures 3.14 to 3.22 tell us if $n_{WIB} = N/8$ there is no need to look for minLLR value outside WIB cluster. This allows us to simplify minLLR early stopping criterion by reducing computational need.

As illustrated with Fig. 3.13, method has the same structure with minLLR method in [46]. Only difference is amount and index of bits used to detect successful decoding.

Additionally, as seen in Figures 3.14 to 3.22 there are some bended results which indicates the *SNR* point that β value switched to higher constant. When β value changed there is a slight improvement for $n_{WIB} = N/16$ but it does not suffice for correct detection of early stopping.

As a result one can conclude that $N/8$ of worst protected information bits among all package is enough to observe and trigger early stopping for BP polar code decoder even for short blocks such as $N = 512$. This amount may even be chosen lower for longer codes.

Another issue with these results (Figures 3.14 to 3.22) when $n_{WIB} = N/16$ chosen there is a breaking point in those BER-SNR results which caused by empirically determined β value. Also the SNR point that β value switched is a user defined parameter which deforms the waterfall shape of BER-SNR graphs.

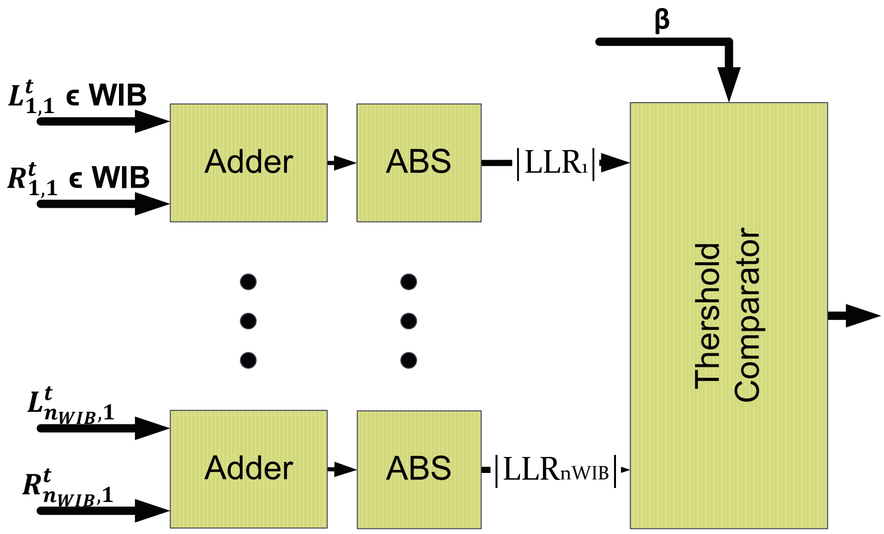


Figure 3.13. Block scheme of simplified minLLR ESC.

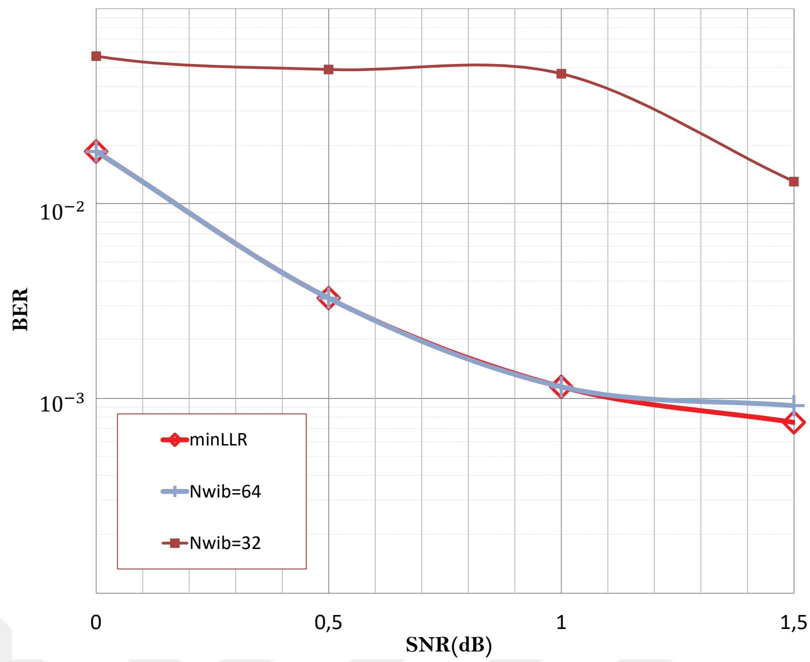


Figure 3.14. BER-SNR results of SMS BP $(512, 169)$ polar code decoder with WIB aided minLLR early stopping criterion method.

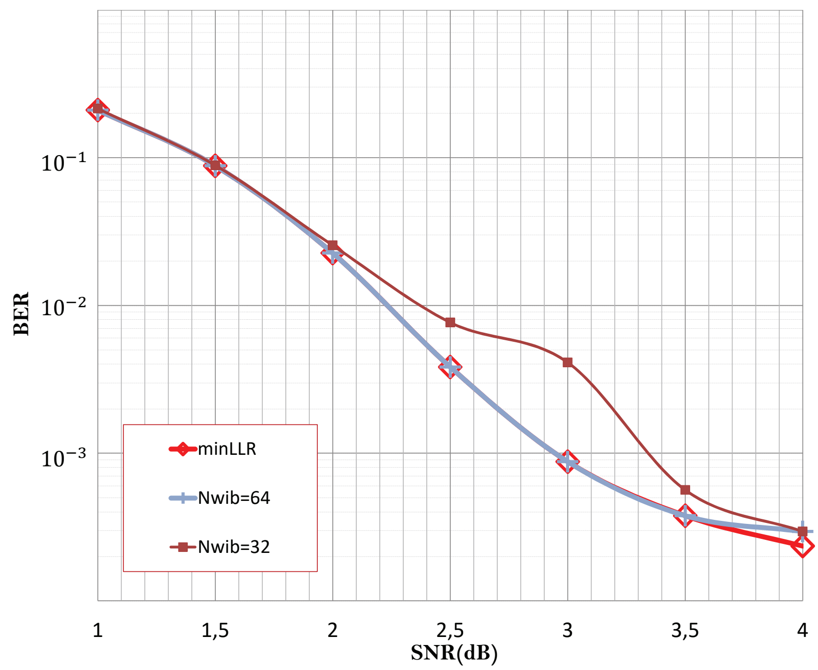


Figure 3.15. BER-SNR results of SMS BP $(512, 256)$ polar code decoder with WIB aided minLLR early stopping criterion method.

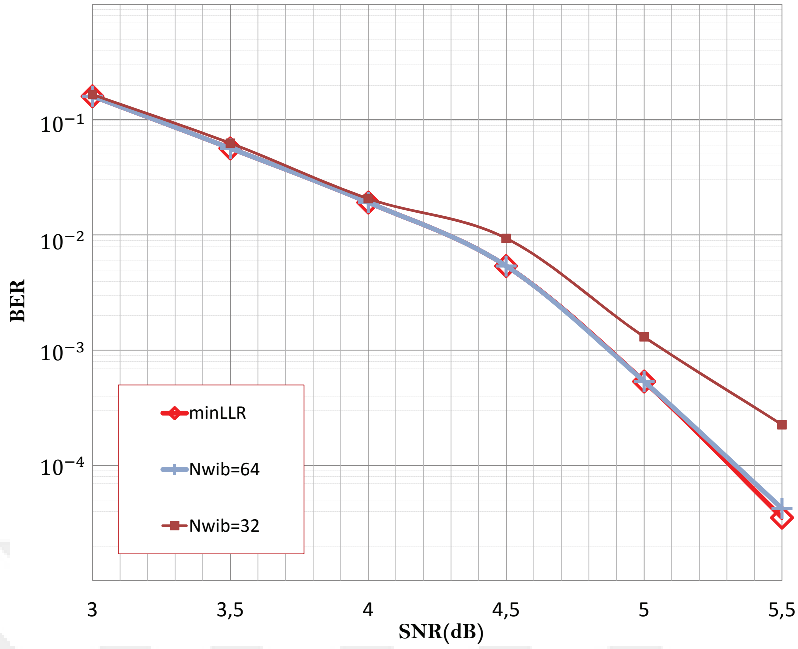


Figure 3.16. BER-SNR results of SMS BP $(512, 338)$ polar code decoder with WIB aided minLLR early stopping criterion method.

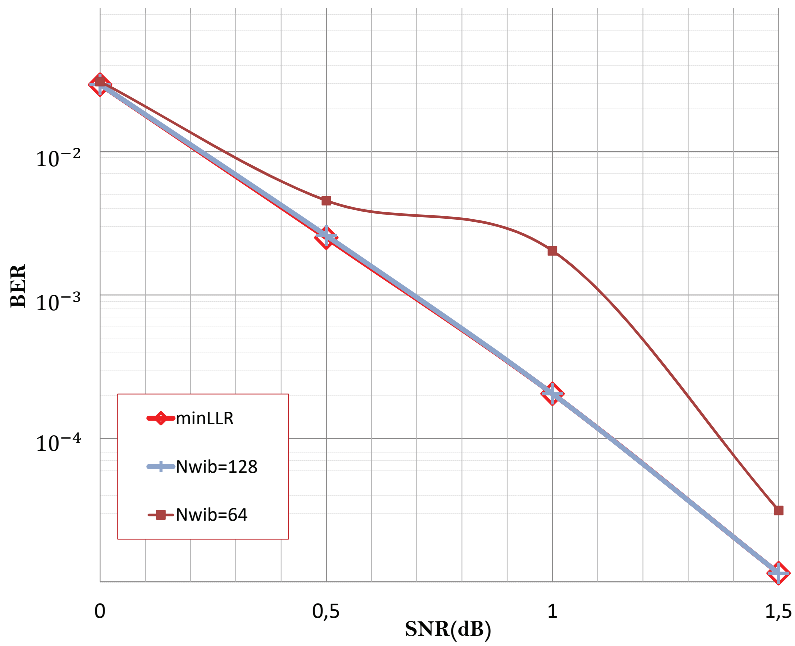


Figure 3.17. BER-SNR results of SMS BP $(1024, 338)$ polar code decoder with WIB aided minLLR early stopping criterion method.

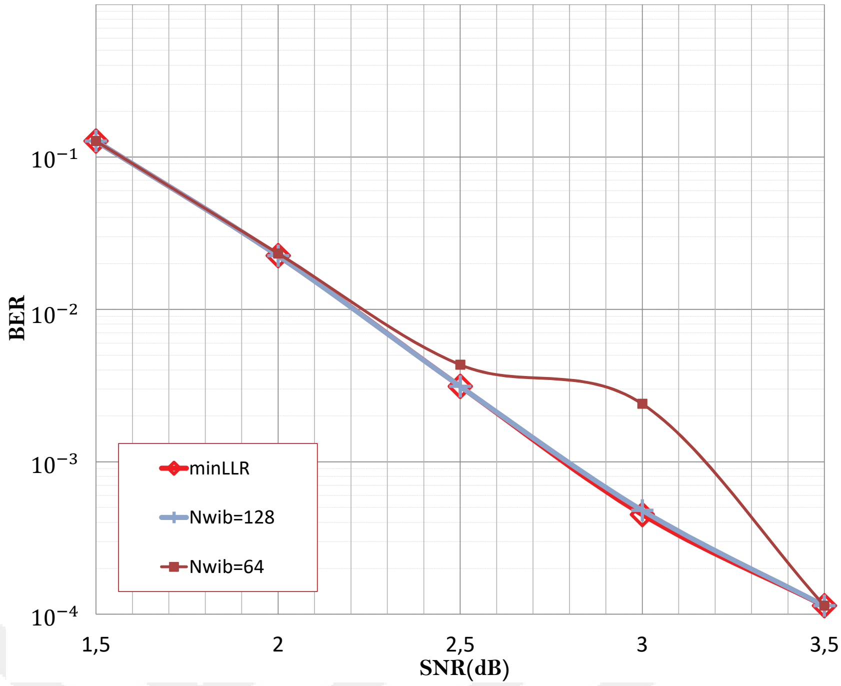


Figure 3.18. BER-SNR results of SMS BP $(1024,512)$ polar code decoder with WIB aided minLLR early stopping criterion method.

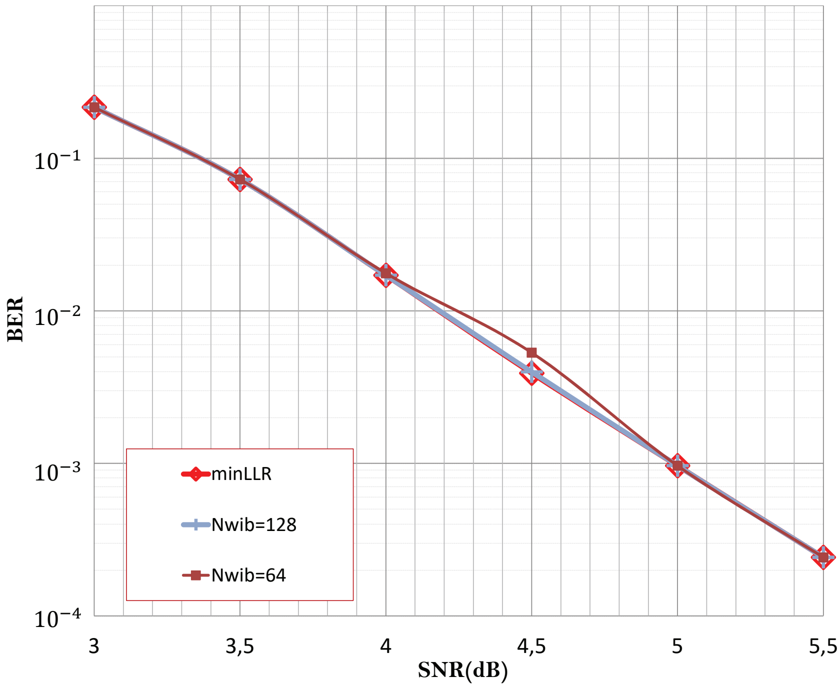


Figure 3.19. BER-SNR results of SMS BP $(1024,676)$ polar code decoder with WIB aided minLLR early stopping criterion method.

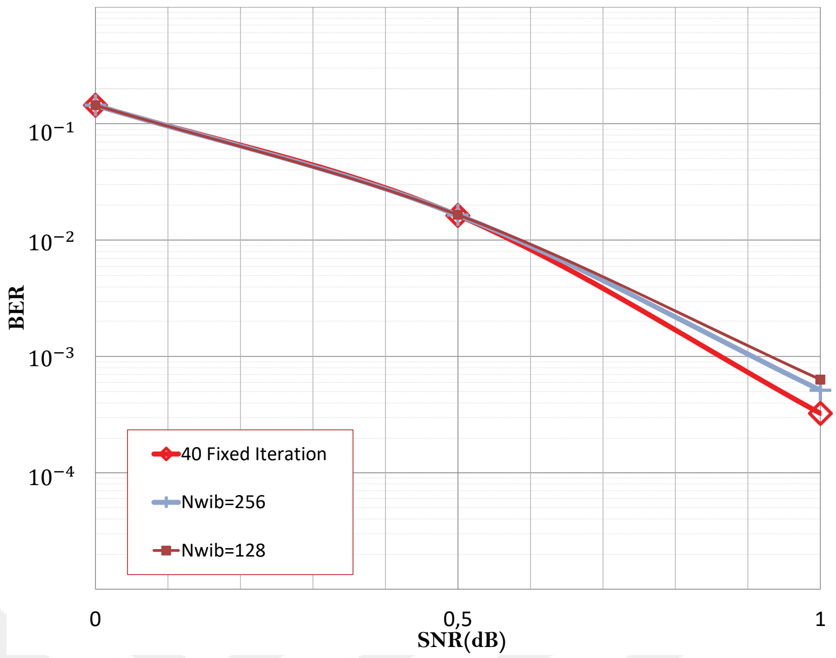


Figure 3.20. BER-SNR results of SMS BP $(2048, 676)$ polar code decoder with WIB aided minLLR early stopping criterion method.

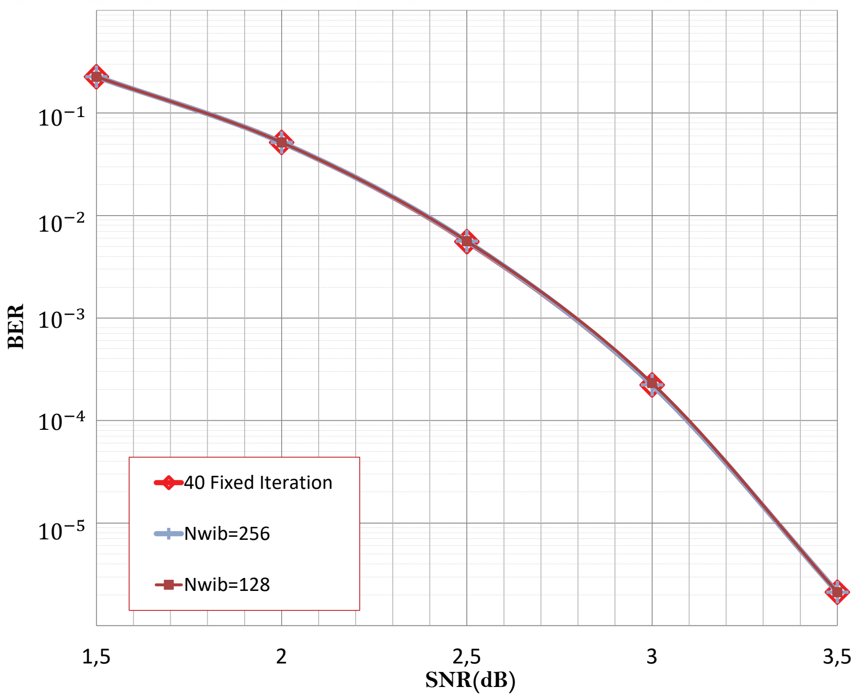


Figure 3.21. BER-SNR results of SMS BP $(2048, 1024)$ polar code decoder with WIB aided minLLR early stopping criterion method.

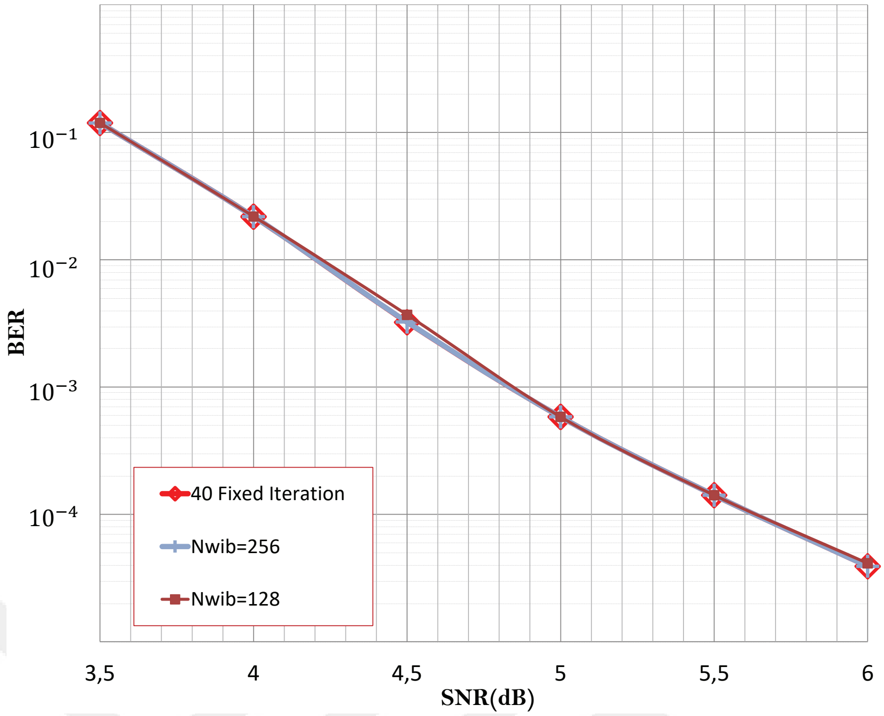


Figure 3.22. BER-SNR results of SMS BP $(2048,1352)$ polar code decoder with WIB aided minLLR early stopping criterion method.

3.3. Similar Early Stopping Approach for Luby Transform Codes

In this study we propose an ESC for LT BP decoder that has similar approach with WIB ESC. Our method observes only sign alterations of a small cluster in passing LLR messages between BP nodes. The method is basically based on the idea that messages have lower absolute LLR values are less reliable [54] and they converge later than messages have higher absolute LLR values. So, observing only the least reliable messages (LRM) which are a small cluster in entire LLR values can be enough to determine successful convergence such as WIB for polar code. We denote this method as LRM ESC. LRM doesn't engage until decoder reaches an empirically pre-determined iteration number varying according to signal to noise ratio (SNR). When iteration number of decoding reaches this pre-determined value, selection of LRM in entire LLR values is performed. During the rest of the decoding only signs of LRM are observed. If the signs of messages in the cluster don't change for a specific iteration number, LRM method assumes that decoder successfully converged. Note that, sign parts of the LLR values are utilized for hard-decision procedure. Therefore, early termination can be done by only observing sign changes of LLR values in BP decoder.

As mentioned in Section 2.5.3, CSR is a common criterion [55–58] for early termination of rateless decoding is observing if the estimated messages $\hat{m}_v$ satisfy the constraints imposed by CNs [55, 56] which means decision, re-encoding and comparing such as G-Matrix for polar code. In contrast to CSR, proposed LRM method doesn't require to perform decision, re-encoding and comparison processes at the end of each iteration. Furthermore, observing only signs of a small cluster in messages instead of all LLR values passing between nodes provides considerable complexity reduction in ESC section. Simulation results and complexity analyzes show that proposed LRM method significantly reduces the computational complexity of early termination section in decoder without any performance loss and also decreases the average iteration amounts compared to CSR.

Observing only worst protected information bits for polar code is generally means observing the lowest LLR values which we may call them as least reliable information bits. For polar code these bits can be sorted according to Bhattacharyya parameters, in a way they most likely will have the lowest LLR values among all information bits. However, ECCs without any parity check mechanism and decodable with BP such as LT or Raptor codes do not have this kind of systematic encoding scheme. In order to use such a simplified early stopping criterion, LLR values need to be sorted after certain amount of iterations.

With this study we tried to apply same principle to other BP decodable ECCs particularly LT code. Instead of using pre-determined bit indexes, we have sorted the LLR values after certain amount of iterations and chosen the lowest ones to observe for early stopping.

3.3.1. Proposed LRM Early Termination Method

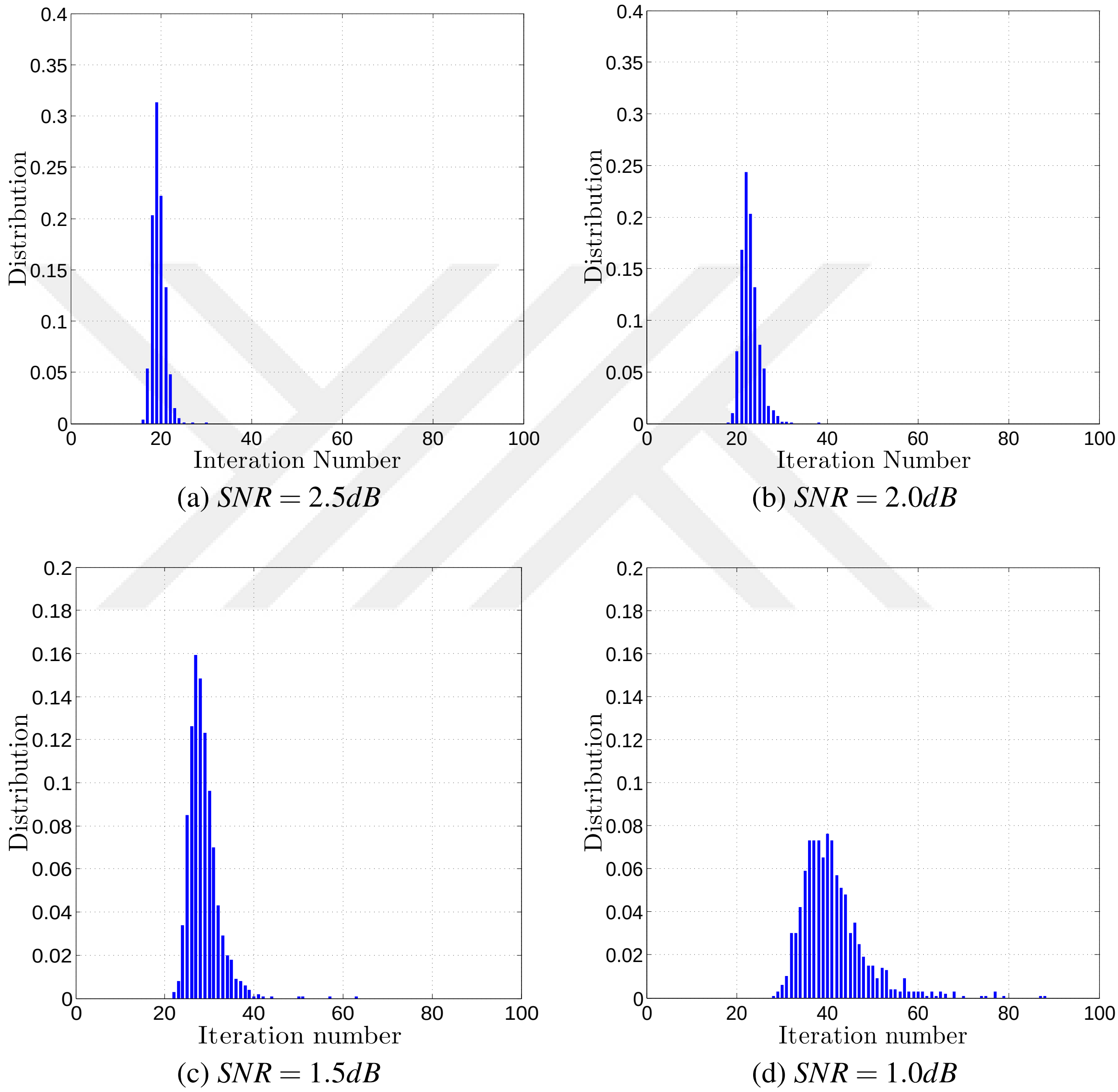
LRM method is based on observing sign alterations of a small cluster in $m_{v\to c}$ messages during the decoding process. As we represent in Algorithm 5, BP decoding algorithm is ended with **Decision()** procedure. In the decision part, after m_v messages are calculated hard-decision is performed according to Eqn. (2.45). Since the sign parts of the LLR values are utilized for hard-decision, observing sign alterations of m_v during successive iterations can be used to determine whether estimated data bits change. If the estimated data bits stop changing for a number of consecutive iterations (Γ_{LC}) it can be assumed that decoder successfully converged. To be able to get lowest average iteration amounts, Γ_{LC} value should be as low as possible. Additionally, this criterion is suitable for LT decoding since LT codes suffer from error floor.

Instead of m_v messages, our proposed method observes sign alterations of $m_{v\to c}$ messages that specify m_v. Therefore, our method doesn't require performing **Decision()** at each decoding iteration. On the other hand, proposed LRM method is basically based on the fact that $m_{v\to c}$ messages with lower absolute LLR values are less reliable among entire $m_{v\to c}$ messages [54] and they converge later than messages that have higher absolute LLR values. Therefore, we observe only LRM which are a small cluster of LLR values to determine successful convergence. This simplification also reduces the computational complexity of ESC section significantly. Determination of LRM which means finding the smallest absolute LLR values in all $m_{v\to c}$ messages, can be easily done by using a selection algorithm. We use quick-select algorithm which has low computational complexity [61].

Another point to take into consideration is that LRM should be determined after running decoder for a few iterations. This is because LT BP decoder typically needs a few iterations to propagate initial channel LLR values. We call these threshold for iteration numbers as determination condition of LRM (DC-LRM). It is easy to see that larger DC-LRM value increases probability of choosing accurate LRM because better propagation occurs when iteration number increases. On the other hand, DC-LRM shouldn't be larger than minimum iteration number that decoder converged. We determine DC-LRM values according to Fig. 3.23(a) - 3.23(d) for various SNR. The figure generated by simulation illustrates iteration number distributions of converged decoding processes. Simulation parameters will be given in next section. DC-LRM values are chosen as 45, 28, 22, 18 and 15 for 0.5, 1.0, 1.5, 2.0 and 2.5dB, respectively. DC-LRM values for different systems can be determined by simulations and previously loaded to a look-up table. LT BP decoding process with proposed LRM method is presented in Algorithm 8.

Algorithm 8 LT BP Decoder with LRM Method:

```
Initialization
    Calculate m_c;
    Set m_{c→v}^{(0)} and m_{v→c}^{(0)} messages to zero, l = 0;
end Initialization
while (l < max_iter) and (Γ_LC is not satisfied) do
    CN update();                                   ▷ Eqn. (2.43) is performed.
    VN update();                                   ▷ Eqn. (2.44) is performed.
    if (l == DC − LRM) then
        Quickselect();                             ▷ LRM are determined.
    end if
    if (l > DC − LRM) then
        Calculate amount of sign changes in LRM;
    end if
    l = l + 1;                                     ▷ Next iteration.
end while
Decision();                                        ▷ Eqn. (2.45) is performed.
```



(a) $SNR = 2.5dB$ (b) $SNR = 2.0dB$

(c) $SNR = 1.5dB$ (d) $SNR = 1.0dB$

Figure 3.23. Average iteration number vs distribution for various SNR values.

3.3.2. Complexity Analysis

In this section, we analyze the computational complexities of BP decoding algorithm, CSR ESC and proposed LRM ESC. VN update equation of BP for LT decoding consists of addition operations. Number of addition at each VN update can be calculated by $d_v(d_v - 2)$, where d_v represents VN degree. Thus, total addition for a single iteration can be calculated by $K \sum_{d_v=2}^{d_{v_{max}}} d_v(d_v - 2)\lambda_{d_v}$, where K is uncoded packet length, λ_{d_v} is the fraction of VNs of degree d_v and $d_{v_{max}}$ is maximum VN degree. According to this, we count up computational complexities of BP algorithm and considered ESCs separately and illustrate the results in Table 3.16. We assume abs, $sign$ and XOR operations have same complexities to simplify the comparison.

Table 3.16. Complexities of BP algorithm and ESCs for single iteration

Complexity Analysis of Early Stopping Criteria			
Operation	**BP Algorithm**	**Early Stopping Criteria**	
		CSR	**LRM**
$Multipication$	$3C_2$	$-$	$-$
$Addition$	$3C_3$	$N + C_4$	N_B
$tanh$	C_2	$-$	$-$
$tanh^{-1}$	C_1	$-$	$-$
$abs, sign, XOR$	$C_1 + C_2$	C_1	N_B
$Compare$	$-$	K	$N_B + 2N_{m_{v-c}}/l_{avg}$

In the table, N_B symbolizes number of LRM determined by $N_B = B * N_{m_{v\to c}}$, where $N_{m_{v\to c}}$ is number of all $m_{v\to c}$ messages and calculated by $N_{m_{v\to c}} = N\Omega'(1)$, where N is coded packet length and $\Omega'(1)$ is average degree of degree distribution chosen for LT code [62]. As we mentioned above, LRM method performs quick-select algorithm only one time for whole decoding process to determine least reliable messages. The quick-select uses less than $2N_{m_{v\to c}}$ compare operations to find the smallest N_B items of an array with length $N_{m_{v\to c}}$ [61]. We add the average effect of quick-select to computational complexities for each iteration by $2N_{m_{v\to c}}/l_{avg}$ comparisons in the table. Here l_{avg} is average iteration number. C_1, C_2, C_3 and C_4 are given with Eqn. (3.4), (3.5), (3.6) and (3.7), respectively.

$$C_1 = N \sum_{d_c=1}^{d_{c_{max}}} d_c \rho_{d_c} \tag{3.4}$$

$$C_2 = N \sum_{d_c=1}^{d_{c_{max}}} d_c^2 \rho_{d_c} \tag{3.5}$$

$$C_3 = K \sum_{d_v=2}^{d_{v_{max}}} d_v(d_v-2)\lambda_{d_v} \tag{3.6}$$

$$C_4 = K(1-\lambda_1) \tag{3.7}$$

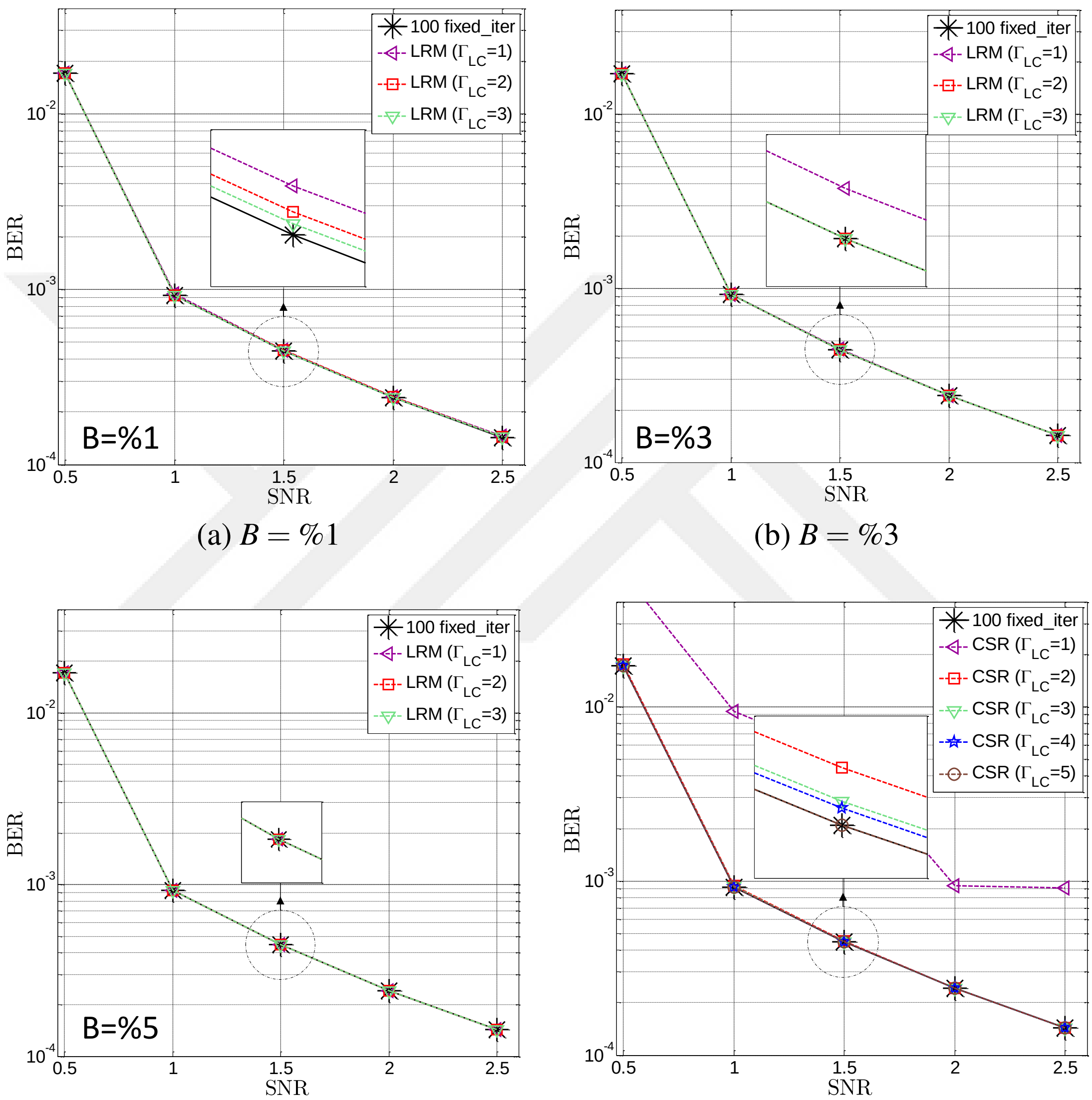
Here d_c is CN degree, ρ_{d_c} is the fraction of CNs of degree d_c and $d_{c_{max}}$ is maximum CN degree. It should be also emphasized that all operations required for CSR method are performed in every decoding iteration until decoding is terminated, while the operations for LRM method start after decoder runs DC-LRM iterations. This effect isn't shown in the table.

3.3.3. Numerical Results for LT BP Decoder with LRM ESC

We evaluate the BER performances of LT BP decoding algorithm with and without ESCs over BI-AWGNC by simulation works. Also, computational complexities of ESCs and average iteration amounts of BP algorithm with LRM and CSR ESCs are compared. For all simulation works and complexity analyzes, we consider the following degree distribution $\Omega(x)$ as in Eqn. (3.8) [62], code rate of $1/2$, data packet length of 4000 and fixed iteration number of 100.

$$\begin{aligned}\Omega(x) =& 0.008x + 0.494x^2 + 0.166x^3 + 0.073x^4 + 0.083x^5 + 0.056x^8 + 0.037x^9 \\ &+ 0.056x^{19} + 0.025x^{65} + 0.003x^{66}\end{aligned} \tag{3.8}$$

Fig. 3.24(a)-3.24(d) illustrates BER curves of LT BP decoder with CSR and proposed LRM ESCs. Results are given for various N_B and Γ_{LC} values. B is used to calculate N_B value from $N_{m_{v\to c}}$ as mentioned in previous section. We also provide BER curve for LT BP with 100 fixed iteration number without ESC as a benchmark. This benchmark shows the best BER values that decoder can reach. Differences between benchmark and other BER values indicate that ESCs stop decoding before decoder converges. An ESC shouldn't cause BER performance degradation. As it can be seen in the figure, LRM method with a few various parameters and CSR only with $\Gamma_{LC} = 5$ don't cause BER performance degradation. In addition to this, since higher Γ_{LC} cause larger average iteration amount we choose $\Gamma_{LC} = 1$ and $B = \%5$ for proposed LRM method and compare it to CSR with $\Gamma_{LC} = 5$.



(a) $B = \%1$

(b) $B = \%3$

(c) $B = \%5$

(d) *CSR*

Figure 3.24. BER curves of LT BP decoder with and without ESCs.

Table 3.17 compares average iteration amounts of LT BP decoder with selected LRM and CSR methods. Second column in Table 3.17 called "Decoder Convergence" is considered as benchmark. LRM ESC has smaller average iteration amounts than CSR but it has slightly higher than benchmark values.

Average computation times of ESCs for decoding a code block are compared in Table 3.18 with considered simulation parameters (CSR with $\Gamma_{LC} = 5$ and LRM with $\Gamma_{LC} = 1, B = \%5$). Results show that required computation time of LRM method is significantly lower than CSR. Note that timing results demonstrate only ESC section. Furthermore, decoder with proposed LRM method has small average iteration amounts compared to decoder with CSR as shown in Table 3.17. This provides additional reduction in computation time of whole decoding process.

Table 3.17. Average Iteration Performances of ESCs and Decoder Converged

Average Iteration Performances			
$SNR(dB)$	**Decoder Converged**	**Early Stopping Criteria**	
		CSR	**LRM**
0.5	90.74	91.65	91.53
1.0	41.25	45.19	43.73
1.5	28.65	32.45	30.15
2.0	22.84	26.70	24.04
2.5	19.42	23.33	20.37

Table 3.18. Average Computation Times of ESCs for Decoding Single Block

Average Computation Time Performances			
$SNR(dB)$	**Computation Times of ESCs (ms)**		**Computation Time Reduction(%)**
	CSR	**LRM**	
0.5	86.18	6.91	91.98
1.0	38.99	3.07	92.13
1.5	26.93	2.01	92.54
2.0	21.80	1.72	92.11
2.5	19.28	1.58	91.80

3.3.4. SNR Independent LRM ESC

As mentioned in previous section a few iterations needed to sort LRM. This method requires precise SNR knowledge. In order to make the method independent from SNR, we propose a different approach.

At the beginning of LT BP decoding SNR Independent LRM ESC randomly choose messages which is a small cluster of $m_{v \to c}$ messages. We call this message packet randomly chosen messages (RCM) whose amount is the same with LRM. First, proposed ESC observes sign alterations of RCM to determine whether RCM are stable. When signs of RCM become stable the method determines LRM from $m_{v \to c}$ messages whose absolute values are the smallest. LRM ESC observes sign alterations of LRM at the rest of iterations to determine whether LRM are stable. If the signs of LRM don't change for a specific iteration number (Γ_{LC}), the method assumes that decoder successfully converged. To be able to get lowest average iteration amounts, Γ_{LC} value should be as low as possible. LT BP decoding process with proposed LRM method is presented in Algorithm 9.

Algorithm 9 LT BP decoder with SNR Independent LRM method:

Initialization
    Calculate m_c;
    Set $m^{(0)}_{c \to v}$ and $m^{(0)}_{v \to c}$ messages to zero;
    $l = 0$, $flag =$ RCM, $\Gamma_{LC} = 0$;
    Determine RCM; ▷ Randomly choose the messages
end Initialization
while $(l < max\_iter)$ **and** $(\Gamma_{LC}\ is\ not\ satisfied)$ **do**
    CN update();
    VN update();
    if $(flag =$ RCM$)$ **and** (RCM *are stable*) **then**
        Quickselect(); ▷ LRM are determined.
        $flag =$ LRM;
    end if
    if $(flag =$ LRM$)$ **and** (LRM *are stable*) **then**
        $\Gamma_{LC}{+}{+}$;
    else $\Gamma_{LC} = 0$;
    end if
    $l{+}{+}$; ▷ Next iteration.
end while
Decision(); ▷ Eqn. (2.45) is performed.

3.3.5. Numerical Results for LT BP Decoder with SNR Independent LRM ESC

We evaluate the BER performances of LT BP decoding algorithm with and without ESCs over BI-AWGNC by simulation works using same parameters with previous LRM ESC. Also, computational complexities of ESCs and average iteration amounts of BP algorithm with SNR independent LRM and CSR ESCs are compared.

Fig. 3.25 (a) and (d) illustrate BER curves of LT BP decoder with proposed SNR independent LRM and CSR ESCs, respectively. We also provide BER curve for LT BP with 100 fixed iteration number without ESC as a benchmark. This benchmark shows the best BER values that decoder can reach. Differences between benchmark and other BER values indicate that ESCs stop decoding before decoder converges which causes performance degradation. Main purpose of an ESC is to reduce total decoding complexity without causing any BER performace degradations. As it can be seen in the figure, LRM method for $\Gamma_{LC} \geq 1$ and CSR method for $\Gamma_{LC} \geq 5$ don't cause BER performance degradations. Since higher Γ_{LC} leads to larger average iteration amounts we choose $\Gamma_{LC} = 1$ and $B = \%0.6$ ($N_B \approx 280$) for proposed LRM method and compare it to CSR with $\Gamma_{LC} = 5$. Fig. 3.25 (b) illustrates the results when sign alterations of m_v messages are observed, where Fig. 3.25 (c) illustares the results when sign alterations of only RCM (without LRM) are observed. According to these results, if only RCM or m_v messages are used to determine successful convergence, required average iteration amounts are larger than proposed LRM method at the point without any BER performance degradation.

Table 3.19 compares SNR independent LRM and CSR ESCs including LT BP decoder from various aspects with selected simulation parameters (CSR with $\Gamma_{LC} = 5$ and SNR independent LRM with $\Gamma_{LC} = 1, B = \%0.6$). Second column in the table named "Decoder Convergence" is considered as benchmark for iteration amounts. SNR independent LRM ESC has smaller average iteration amounts than CSR but it has slightly higher than benchmark values. Table 3.19 also illustrates average computation times of total decoding process and only ESC sections for decoding a code block. Results show that SNR independent LRM significantly reduces the required computation time for ESC section (up to %92.44 @ 2.5dB) as well as entire decoding process (up to %13.07 @ 2.5dB) compared to CSR method.

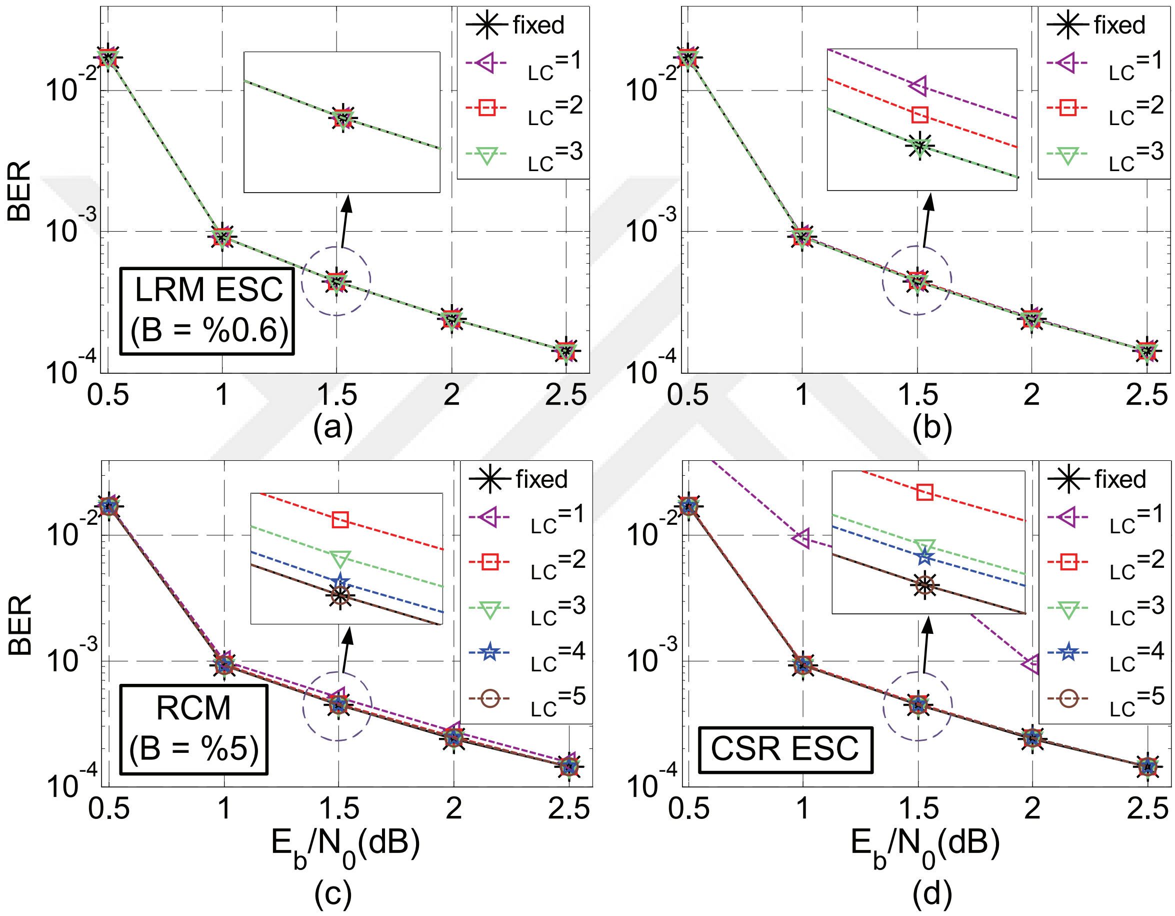


Figure 3.25. BER curves of LT BP decoder with and without CSR and SNR independent LRM ESC.

Table 3.19. Average iteration amounts of LT BP decoder with ETMs, LT BP decoder successfully converged and average computation times of LT BP decoder with and without EMTs for decoding a code block

E_b/N_0 (dB)	**Decoder**	**Iteration Amount**			**Computation Time (ESC Only)**			**Computation Time (BP + ESC)**		
	Converged	**CSR**	**LRM**	**Reduction (%)**	**CSR**	**LRM**	**Reduction (%)**	**CSR**	**LRM**	**Reduction (%)**
0.5	90.74	91.65	90.88	0.84	63.21	1.67	97.36	8781.87	8647.08	1.53
1.0	41.25	45.19	42.76	5.38	25.62	1.19	95.36	4324.54	4068.95	5.91
1.5	28.65	32.45	29.92	7.80	17.93	1.08	93.98	3104.90	2847.37	8.29
2.0	22.84	26.70	24.05	9.93	14.69	1.02	93.06	2554.66	2288.90	10.40
2.5	19.42	23.33	20.39	12.60	12.96	0.98	92.44	2232.34	1940.68	13.07

3.4. VHDL Implementation and Throughput Analysis of Early Stopping Criteria for Polar and LT Code Decoder

For any algorithm or ECC applicability is an important issue. As mentioned in Chapter 1, LDPC was invented in 1965 and could not be used until recent years because of resource limitations of hardwares. This perspective drives researchers to investigate the applicability of methods as in [43, 45, 46, 50, 63–73] for both SC and BP decoders of polar code.

This section of study presents the hardware implementations and synthesis reports of all polar code early stopping criteria which are written in VHDL language and synthesized with XILINX ISE.

3.4.1. Throughput Analysis of Simplified WIB ESC Compared with G-Matix ESC

The best way to analysis for throughput of hardware structure is to implement the structure with application specific integrated circuit (ASIC) design. However, in this study we did not have required infrastructure for an ASIC design. Another way to calculate the throughput levels is to convert the designs to a base design as in [43]. In this section we converted simplified WIB method design to the design base published in [46]. Table 3.20 illustrates the results of converted designs critical path delays and throughput levels.

Module PE (see Fig. 2.18) is the critical element to determine maximum clock frequency so early stopping section does not have a major effect on this. However, average latency (clock cycles) for decoding a code block as well as the gate count and average power consumption per bit is effected by how sooner ESC stops iterations and how simpler its hardware. As presented following sections, simplified WIB ESC has a really small gate count and mid-level average iteration numbers. All of these parameters and their effects are summarized in Table 3.20.

Table 3.20. Converted Design Summaries and Results for SNR=3.5dB

Converted Design Summaries			
Design	***G-Matrix***	***minLLR***	***Simplified WIB***
Total Gate Count	1961584	2018993	1920590
Average Number of Iterations	23	30.7	26.7
Average Latency(cycles)	56	73	63
Energy per bit(pJ/bit)	214	287	236
Average Throughput(Gbps)	4.51	3.51	4.11

As one can see from Table 3.20 WIB ESC has mid-level throughput and energy per bit values which is much closer to G-Matrix ESC and also has the lowest gate counts.

In Table 3.20 gate count results for G-Matrix and minLLR ESCs are taken from [46] where G-Matrix ESC has lower gate count then minLLR. However, our designs in following sections show that G-Matrix ESC has the highest gate counts which needs to be indicated as a contradiction with referred study. We also find energy per bits values slightly different then [46].

3.4.2. VHDL Implementation and Synthesis Reports of ESCs for Polar Code

Here with Table 3.21 to 3.35 we give important parameters such as path delays, resource usage and gate counts for ESCs design with VHDL and synthesized in XILINX ISE. Tables clearly indicates that simplified WIB ESC is way more efficient than others from hardware complexity point of view. Also delay parameters are the lowest among all ESCs.

Table 3.21. Design Summary of G-Matrix ESC

==

Design Summary

==

Primitive and Black Box Usage:

# BELS	:	30780
# GND	:	1
# LUT2	:	12316
# LUT3	:	1362
# LUT4	:	369
# LUT5	:	1220
# LUT6	:	3224
# MUXCY	:	10240
# XORCY	:	2048
# IOBUFFERS	:	24577
# IBUF	:	24576
# OBUF	:	1

==

Table 3.22. Macro Statistics of HDL Synthesis Report for G-Matrix ESC

HDL Synthesis Report

Macro Statistics

# Adders/Subtractors	:	2048
6-bit adder	:	2048
# Multiplexers	:	2048
1-bit 2-to-1 multiplexer	:	2048
# Xors	:	5832

Table 3.23. Timing Analysis of G-Matrix ESC

Timing constraint: Default path analysis

Data Path:

Cell:in ⟶ out	fanout	Gate Delay(ns)	Net Delay(ns)
IBUF:I⟶O	1	0.000	0.405
LUT2:I0⟶O	1	0.043	0.000
MUXCY:S⟶O	1	0.238	0.000
MUXCY:CI⟶O	1	0.014	0.000
MUXCY:CI⟶O	1	0.014	0.000
MUXCY:CI⟶O	1	0.014	0.000
MUXCY:CI⟶O	0	0.014	0.000
XORCY:CI⟶O	9	0.262	0.395
LUT3:I2⟶O	92	0.043	0.744
LUT5:I0⟶O	30	0.043	0.624
LUT5:I2⟶O	9	0.043	0.658
LUT6:I0⟶O	11	0.043	0.669
LUT6:I0⟶O	6	0.043	0.641
LUT6:I0⟶O	5	0.043	0.636
LUT6:I0⟶O	4	0.043	0.539
LUT6:I2⟶O	1	0.043	0.603
LUT5:I0⟶O	1	0.043	0.603
LUT6:I1⟶O	1	0.043	0.603
LUT6:I1⟶O	1	0.043	0.495
LUT6:I3⟶O	1	0.043	0.339
OBUF:I⟶O		0.000	

	Total Delay(ns)	Logic Delay(ns)	Route Delay(ns)
TOTAL	9.069	1.113	7.956

Table 3.24. Macro Statistics of HDL Synthesis Report for minLLR ESC

HDL Synthesis Report		
Macro Statistics		
# Adders/Subtractors	:	1024
6-bit adder	:	1024
# Comparators	:	1024
6-bit comparator greater	:	1024

Table 3.25. Design Summary of minLLR ESC

Design Summary		
Primitive and Black Box Usage:		
# BELS	:	5942
# LUT2	:	2049
# LUT3	:	511
# LUT4	:	6
# LUT5	:	5
# LUT6	:	3371
# IOBUFFERS	:	24577
# IBUF	:	10241
# OBUF	:	1

Table 3.26. Timing Analysis of minLLR ESC

Timing constraint: Default path analysis

Data Path:

Cell:in ⟶ out	fanout	Gate Delay(ns)	Net Delay(ns)
IBUF:I⟶O	2	0.000	0.618
LUT6:I0⟶O	3	0.043	0.362
LUT3:I2⟶O	1	0.043	0.522
LUT6:I2⟶O	1	0.043	0.603
LUT6:I1⟶O	1	0.043	0.613
LUT6:I0⟶O	1	0.043	0.350
LUT6:I5⟶O	1	0.043	0.350
LUT6:I5⟶O	1	0.043	0.522
LUT4:I0⟶O	1	0.043	0.495
LUT6:I3⟶O	1	0.043	0.339
OBUF:I⟶O		0.000	

	Total Delay(ns)	Logic Delay(ns)	Route Delay(ns)
TOTAL	5.163	0.387	4.776

Table 3.27. Macro Statistics of HDL Synthesis Report for Simplified minLLR ESC

HDL Synthesis Report

Macro Statistics

# Adders/Subtractors	:	128
6-bit adder	:	128
# Comparators	:	128
6-bit comparator greater	:	128

Table 3.28. Design Summary of Simplified minLLR ESC

Design Summary

Primitive and Black Box Usage:

# BELS		:	747
#	LUT2	:	257
#	LUT3	:	63
#	LUT4	:	12
#	LUT5	:	2
#	LUT6	:	413
# IOBUFFERS		:	1281
#	IBUF	:	1280
#	OBUF	:	1

Table 3.29. Timing Analysis of Simplified minLLR ESC

Timing constraint: Default path analysis

Data Path:

Cell:in ⟶ out	fanout	Gate Delay(ns)	Net Delay(ns)
IBUF:I⟶O	2	0.000	0.618
LUT6:I0⟶O	3	0.043	0.362
LUT3:I2⟶O	1	0.043	0.522
LUT6:I2⟶O	1	0.043	0.495
LUT4:I1⟶O	1	0.043	0.405
LUT6:I4⟶O	1	0.043	0.405
LUT6:I4⟶O	1	0.043	0.405
LUT6:I4⟶O	1	0.043	0.339
OBUF:I⟶O		0.000	

	Total Delay(ns)	Logic Delay(ns)	Route Delay(ns)
TOTAL	3.853	0.301	3.552

Table 3.30. Macro Statistics of HDL Synthesis Report for WIB ESC

HDL Synthesis Report		
Macro Statistics		
# Adders/Subtractors	:	128
6-bit adder	:	128
# Xors	:	256
1-bit xor2	:	256
# Multiplexers	:	128
1-bit 2-to-1 multiplexer	:	128

Table 3.31. Design Summary of WIB ESC

Design Summary		
Primitive and Black Box Usage:		
# BELS	:	1711
# GND	:	1
# LUT2	:	768
# LUT3	:	43
# LUT5	:	42
# LUT6	:	44
# MUXCY	:	684
# VCC	:	1
# XORCY	:	128
# IOBUFFERS	:	1671
# IBUF	:	1670
# OBUF	:	1

Table 3.32. Timing Analysis of WIB ESC

Timing constraint: Default path analysis			
Data Path:			
Cell:in $\longrightarrow$ out	fanout	Gate Delay(ns)	Net Delay(ns)
IBUF:I$\longrightarrow$O	2	0.000	0.405
LUT2:I0$\longrightarrow$O	1	0.043	0.000
MUXCY:S$\longrightarrow$O	2	0.476	0.000
MUXCY:CI$\longrightarrow$O	47	0.539	0.339
XORCY:CI$\longrightarrow$O	1	0.262	0.350
LUT3:I2$\longrightarrow$O	1	0.262	0.405
LUT5:I4$\longrightarrow$O	1	0.043	0.603
LUT6:I0$\longrightarrow$O	1	0.043	0.000
OBUF:I$\longrightarrow$O		0.000	
	Total Delay(ns)	Logic Delay(ns)	Route Delay(ns)
TOTAL	3.771	1.668	2.103

Table 3.33. Macro Statistics of HDL Synthesis Report for Simplified WIB ESC

HDL Synthesis Report		
Macro Statistics		
# Xors	:	128
1-bit xor2	:	128

Table 3.34. Design Summary of Simplified WIB ESC

Design Summary		
Primitive and Black Box Usage:		
# BELS	:	90
# GND	:	1
# LUT4	:	1
# LUT6	:	43
# MUXCY	:	44
# VCC	:	1
# IOBUFFERS	:	263
# IBUF	:	262
# OBUF	:	1

Table 3.35. Timing Analysis of Simplified WIB ESC

Timing constraint: Default path analysis			
Data Path:			
Cell:in ⟶ out	fanout	Gate Delay(ns)	Net Delay(ns)
IBUF:I⟶O	1	0.000	0.613
LUT6:I0⟶O	1	0.043	0.000
MUXCY:S⟶O	1	0.238	0.000
MUXCY:CI⟶O	42	0.704	0.339
OBUF:I⟶O		0.000	
	Total Delay(ns)	Logic Delay(ns)	Route Delay(ns)
TOTAL	1.937	0.985	0.952

3.4.3. VHDL Implementation and Synthesis Reports of ESCs for LT Code

Here with Table 3.36 to 3.41 we give important parameters such as path delays, resource usage and gate counts for LT BP decoder ESCs design with VHDL and synthesized in XILINX ISE. Tables clearly indicate that SNR independent LRM ESC is much more efficient than CSR ESC for both hardware complexity and latency parameters. It should be noted that quick-select algorithm is not included to this design science its effect will be very low concidering only performed once.

Table 3.36. Macro Statistics of HDL Synthesis Report for SNR Independent LRM ESC

HDL Synthesis Report		
Macro Statistics		
# Xors	:	300
1-bit xor2	:	300

Table 3.37. Timing Analysis of SNR Independent LRM ESC

Timing constraint: Default path analysis

Data Path:

Cell:in ⟶ out	fanout	Gate Delay(ns)	Net Delay(ns)
IBUF:I⟶O	2	0.000	0.618
LUT6:I0⟶O	1	0.043	0.603
LUT5:I0⟶O	1	0.043	0.603
LUT6:I1⟶O	1	0.043	0.613
LUT6:I0⟶O	1	0.043	0.603
LUT5:I0⟶O	1	0.043	0.339
OBUF:I⟶O		0.000	

	Total Delay(ns)	Logic Delay(ns)	Route Delay(ns)
TOTAL	3.595	0.215	3.380

Table 3.38. Design Summary of SNR Independent LRM ESC

Design Summary

Primitive and Black Box Usage:

# BELS			154
#	LUT3	:	1
#	LUT4	:	17
#	LUT5	:	12
#	LUT6	:	124
# IOBUFFERS			601
#	IBUF	:	600
#	OBUF	:	1

Table 3.39. Macro Statistics of HDL Synthesis Report for CSR ESC

==		
HDL Synthesis Report		
==		
Macro Statistics		
# Adders/Subtractors		3996
8-bit adder	:	3996
# Multiplexers		3996
1-bit 2-to-1 multiplexer	:	3996
# Xors		8021
1-bit xor17	:	2
1-bit xor18	:	13
1-bit xor19	:	193
1-bit xor2	:	6016
1-bit xor3	:	648
1-bit xor4	:	339
1-bit xor5	:	313
1-bit xor61	:	3
1-bit xor62	:	12
1-bit xor63	:	22
1-bit xor64	:	45
1-bit xor65	:	33
1-bit xor66	:	2
1-bit xor7	:	5
1-bit xor8	:	234
1-bit xor9	:	141
==		

Table 3.40. Timing Analysis of CSR ESC

Timing constraint: Default path analysis

Data Path:

Cell:in ⟶ out	fanout	Gate Delay(ns)	Net Delay(ns)
IBUF:I⟶O	1	0.000	0.405
LUT2:I0⟶O	1	0.043	0.000
MUXCY:S⟶O	1	0.238	0.000
MUXCY:CI⟶O	1	0.014	0.000
MUXCY:CI⟶O	1	0.014	0.000
MUXCY:CI⟶O	1	0.014	0.000
MUXCY:CI⟶O	1	0.014	0.000
MUXCY:CI⟶O	1	0.014	0.000
MUXCY:CI⟶O	0	0.014	0.000
XORCY:CI⟶O	1	0.262	0.350
LUT3:I2⟶O	10	0.043	0.663
LUT6:I0⟶O	1	0.043	0.613
LUT6:I0⟶O	1	0.043	0.613
LUT6:I0⟶O	1	0.043	0.350
LUT6:I5⟶O	1	0.043	0.350
LUT6:I5⟶O	1	0.043	0.613
LUT6:I0⟶O	1	0.043	0.495
LUT5:I2⟶O	1	0.043	0.495
LUT6:I3⟶O	1	0.043	0.339
OBUF:I⟶O		0.000	

	Total Delay(ns)	Logic Delay(ns)	Route Delay(ns)
TOTAL	6.299	1.011	5.288

Table 3.41. Design Summary of CSR ESC

===

Design Summary

===

Primitive and Black Box Usage:

# BELS			74650
#	GND	:	1
#	LUT2	:	31968
#	LUT3	:	3949
#	LUT4	:	279
#	LUT5	:	1368
#	LUT6	:	5117
#	MUXCY	:	27972
#	XORCY	:	3996
# IOBUFFERS			63937
#	IBUF	:	63936
#	OBUF	:	1

===

3.5. Hardware Optimization for Belief Propagation Polar Code Decoder with Early Stopping Criteria Using High-Speed Parallel-Prefix Ling Adder

Maximum clock frequency for a hardware implementation is determined by the block which has the highest critical path delay. This parameter is directly related to throughput performance. For BP polar decoder this unit is PE according to design made in [46, 50]. In this part of our study we propose an idea about how to increase polar BP decoder speed by decreasing the critical path delay of PE used in [46, 50] with the help of modified WIB ESC. As we remember from Section 3.1.2 modification rules out adder array at classical WIB ESC [59] without any performance degradation. While modified WIB ESC can accommodate the speed increment, G-Matrix and minLLR ESCs can not. Additionally, as stated in [46] when code length (N) is increased critical path delays of G-Matrix and minLLR ESCs increase proportionally which requires more pipelining to keep critical path delays inside a limit. However, modified WIB offers only three levels of logics (LoL) at any condition which provides flexibility for higher speeds.

Proposed and previous methods are also compared with FPGA implementations for logic gate delays. Although, this implementation does not have same parameters for an application specific chip design, but it provides a valuable insight about timing ratios which allows comparison.

According to design made in [50] PE is optimized to have approximately $4T_{adder}$ delay. As we mention in Section 2.4.4 there are two types of blocks as Eqn. (2.36) has two different types of calculations (TypeI Eqn. (3.9), TypeII Eqn. (3.10)). Hardware structures of both types include same components with different order [50].

$$d = a + s * sign(b)sign(c)min(|b|,|c|) \tag{3.9}$$

$$d = s * sign(a)sign(b+c)min(|a|,|b+c|) \tag{3.10}$$

3.5.1. Optimizing PE for Polar BP Decoder

Inside Fig. 2.19 there should be approximately $5T_{adder}$ delay from comparator unit, scale unit, addition unit, 2's complement and inverse conversion units (S2C, C2S). Modified form of 2's complement conversion unit (mS2C) simply carries out 1bit addition operation to mAdder circuit as carry input according to sign signals which decrease the critical path delay of PE to $4T_{adder}$ (see Fig. 2.21) [50]. However delay of carry ripple adder (CRA) used in [50] is not a good choice compared to Ling adder in [74, 75]. In CRA there are $3LoL$ per bit resulting $24LoL$ per $8bits$ adder where Ling adder has only $6LoL$ per $8bits$ adder.

As stated in ([43] Fig. 9) $6bits$ depth is enough to represent LLR values. According to this, $8bits$ Ling adder [74] can be used with only $6LoL$ per adder (see Fig. 3.26) which will decrease the critical path delay of PE further. When low delay PE is used, other parts of decoder (such as ESC) should have critical path delays as low as PE. Otherwise, other modules become the bottleneck for maximum frequency or they require a couple stages of pipelining which will increase required clock cycles for decoding process.

3.5.2. FPGA Implementation and Delay Results For Modified PE with Modified WIB ESC

We implement both ESCs (Modified WIB and G-Matrix) and PEs for device Xilinx VIRTEX7 7v2000tflg1925-2 using Xilinix ISE. Logic gate delay results are collected from design report summary. All results are provided for $(1024, 512)$ polar code.

As mentioned in previous section maximum clock frequency is determined by block which has the highest delay. As it can be seen from Table 3.42 if modified PE with G-Matrix ESC is used, G-Matrix block will be bottleneck for the maximum clock frequency. However, if modified PE with modified WIB is used the maximum clock frequency will be determined by modified PE block which will increase the maximum clock frequency resulting higher throughput for decoder.

With the help of Table 3.42 and results provided in [46, 59] throughput vs SNR results are calculated with Eqn. (3.11), (3.12) and illustrated in Fig. 3.27. It can be seen that proposed

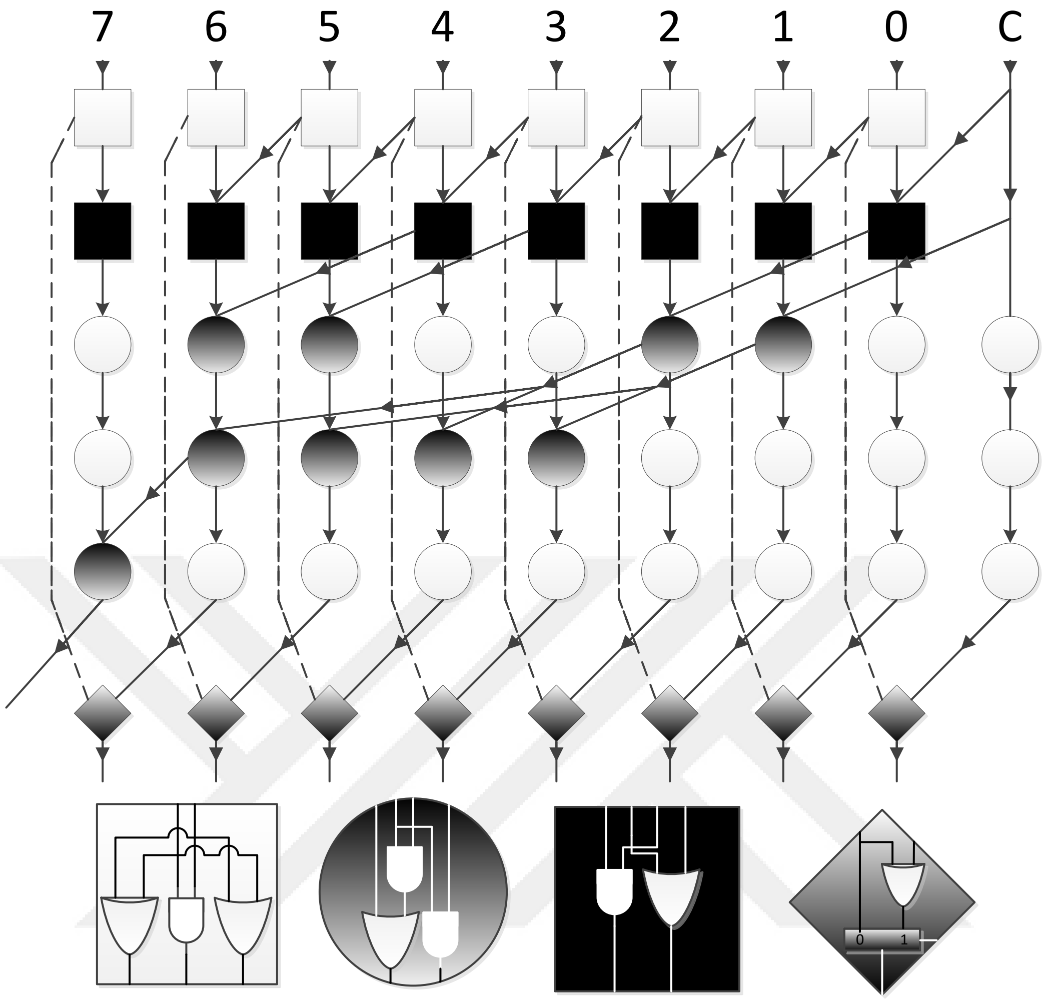


Figure 3.26. Hardware of Ling Adder [76].

design has the highest throughput. In Eqn. (3.11) and (3.12), k represents information bit amount, v is required average iteration, m is layer amount and FER_{SNR} frame error rate according to signal to noise ratio in [59].

$$Average\ Throughput = \frac{Clock\ Freq. * k * (1 - FER_{SNR})}{Average\ Clock\ Cycles} \tag{3.11}$$

$$Average\ Clock\ Cycles = 2 * v + m \tag{3.12}$$

Although this design does not have same parameters with [46], a close approximation can be made according to [74] with Ling adder to compare designs. We give this approximation in next section.

Table 3.42. Logic Gate Delays Produced with XILINX ISE

XILINX ISE Design Report Summary	
Block Name	***Logic Gate Delay(ns)***
G-Matrix(Ling)	1.140
G-Matrix(CRA)	1.532
Modified WIB	0.701
PE with CRA	2.594
PE with Ling	0.959

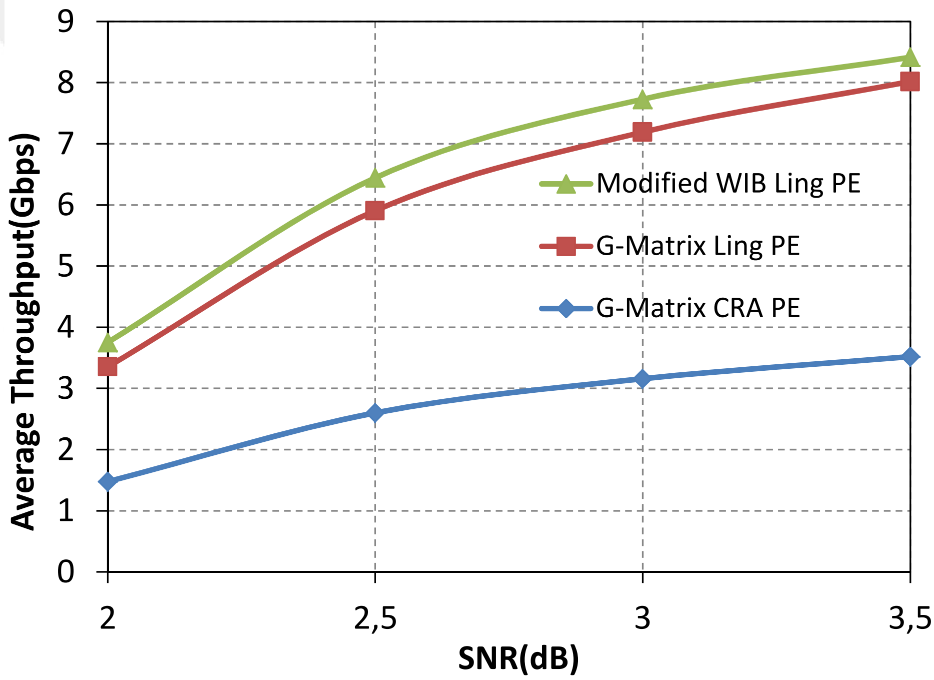


Figure 3.27. Throughput values according to logic gate delays in Table 3.42 and FER in [59].

3.5.3. Approximation for PE Delay Parameter

As stated in ([74] Fig. 8) $8bits$ single adder delay is between $T_{adder} = 0.21 - 0.067ns$ according to different voltage values. As stated in [50] PE has approximately $4T_{adder}$ delay where delays are calculated for $45nm$ $1.1V$ design parameters. With the help of these informations it is safe to say that delay of modified PE should be approximately

$4T_{adder} = 0.84ns$ for $45nm$ with the lowest voltage value. Outcome of this approximation is given in Table 3.43 for PE Ling with modified WIB method. Throughput and average clock cycle values for proposed method is also calculated by Eqns. (3.11) and (3.12).

As a result both FPGA implementation result and approximation made by [74] support that, delay of PE used in SMS-BP polar code decoder can be reduced. Therefore speed of decoder and throughput values can be increased with help of modified WIB ESC. Additionally modified WIB ESC does not require pipelining even with increased code length. This also provides a stable design for various parameters.

Table 3.43. Approximation for Decoder with Fixed WIB using Ling Adder PE According to Delay Results Given in [74]

Design Approximation @ $3.5dB$	**Decoder with Fixed WIB and Ling Adder**
Critical Delay(ns)	0.84
Maximum Clock Frequency(GHz)	1.19
Average Number of Iterations	26.7
Average Clock Cycles	64.4
Average Throughpu(Gbps)	9.45

4. CONCLUSIONS AND FUTURE WORKS

A low complexity early stopping structure for belief propagation decoders is proposed with this thesis. In contrast to previous early stopping methods in literature, proposed early stopping structure only uses small amount of LLR messages and tracks only sign alterations of them.

Proposed structure is applied to both polar and LT codes and can be easily applied to error correction codes use BP as decoder. Performance parameters are compared with simulation works and VHDL implementations. Results illustrate that proposed approach significantly reduces the computational complexity and required hardware resources, also throughput values are increased compared to previous counterparts in literature. Additionally, we proposed a modification for hardware structure of polar belief propagation decoder to increase throughput further with help of proposed early stopping criterion.

The methods we proposed here for polar code have the lowest complexities among others but WIB and simplified WIB methods does not produce the lowest average iteration amounts. However, as evident from the last section, this disadvantage is fading when PE's speed and code length is increased. Even throughput values of proposed method is becoming better from G-matrix ESC. On the other hand, SNR independent LRM ESC for LT BP decoder has the best performance from every aspect compared with literature.

For future work, we are planing to implement the proposed method with ASIC design to observe actual performance parameters (energy per bits, area, total gate count, speed and delay) and for better comparison with literature. Also it is our aim to use polar and LT code with ESCs for visible light communication systems which we are currently working on.

5. REFERENCES

1. Shannon, C., A Mathematical Theory of Communication, The Bell System Technical Journal, 27 (1948) 379–423.

2. Hamming, R. W., Error Detecting and Error Correcting Codes, The Bell System Technical Journal, 29 (1950) 147–160.

3 Reed, I. S., A class of multiple-error-correcting codes and the decoding scheme, Transactions of the IRE Professional Group on Information Theory, 4 (1954) 38–49.

4. Muller, D. E., Application of Boolean algebra to switching circuit design and to error detection, IRE Transactions on Electronic Computers, 3 (1954) 6–12.

5. Gallager, R.G., Low-Density Parity-Check Codes, Ph.D. thesis, Massachusetts Institute of Technology, Massachusetts, 1960.

6. Berrou, C., Glavieux, A., and Thitimajshima, P., Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes 1, IEEE International Conference on Communications (ICC'93), May 1993, Switzerland, In Proceedings of ICC'93: 1064–1070.

7. Luby, M., LT Codes, In Proceedings of the IEEE Symposium on the Foundations of Computer Science, Vancouver, BC, 2002, In proceedings: 271–280.

8. Shokrollahi, A., Raptor Codes, Transactions on Information Theory, 52, 6 (2006) 2551–2567.

9. Arikan, E., Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels, IEEE Transactions on Information Theory, 55, 7 (2009) 3051–3073.

10. Yuan, B., Algorithm and VLSI Architecture for Polar Codes Decoder, Ph.D. thesis, University of Minnesota, Minnesota, 2015.

11. Moore, G. E., Cramming more components onto integrated circuits, Electronics, 38, 8 (1965).

12. HUAWEI. 5G: Full Spectrum Access, New Architecture, New Air Interface. http://www.huawei.com/minisite/5g/en/technological-innovation.html, 1 May 2015.

13. Sklar, B., Designing Digital Communication Systems, Communication Magazine, e-book, 1993.

14. Pearl, J., Reverend Bayes on inference engines: A distributed hierarchical approach, National Conference on Artificial Intelligence, August 1982, USA, In Proceedings.

15. McEliece, R. J., MacKay, D. J. C., and Cheng, J.F., Turbo decoding as an instance of Pearl's "belief propagation" algorithm, IEEE Journal on selected areas in communications, 16,2 (1998) 140–152.

16. Kschischang, F. R., Frey, B. J., and Loeliger, H. A., Factor graphs and the sum-product algorithm, IEEE Transactions on information theory, 47,2 (2001) 498–519.

17. Pfister, H. D., A brief introduction to polar codes, Lecture Notes, Duke University, Durham North Carolina, 2014.

18. Arikan, E., A performance comparison of polar codes and Reed-Muller codes, IEEE Communications Letters, 12,6 (2008) 447–449.

19. Korada, S. B., Sasoglu, E. and Urbanke, R., Polar codes: Characterization of exponent, bounds, and constructions, IEEE Transactions on Information Theory, 56,12 (2010) 6253–6264.

20. Şaşoğlu, E., Telatar, E., and Arikan, E., Polarization for arbitrary discrete memoryless channels, IEEE Information Theory Workshop (ITW 2009), October 2009, Italy, In proceedings: 144–148.

21. Pedarsani, P., Hassani, S. H., Tal, I., and Telatar, E., On the construction of polar codes, IEEE International Symposium on Information Theory (ISIT 2011), August 2011, Russia, In proceedings: 11–15.

22. Serbetci, B. and Pusane, A. E., Practical polar code construction using generalized generator matrices, IET Communications, 8,4 (2014) 419–426.

23. Sha, J., Liu, J. and Wang, Z., Improved BP decoder for polar codes based on a modified kernel matrix, IET Electronics Letters, 52,24 (2016) 1982–1984.

24. Vangala, H., Viterbo, E. and Hong, Y., A comparative study of polar code constructions for the AWGN channel, arXiv preprint arXiv:1501.02473, 2015.

25. Li, H. and Yuan, J., A practical construction method for polar codes in AWGN channels, IEEE TENCON Spring Conference, April 2013, Australia, In proceedings: 223–226.

26. Tal, I. and Vardy, A., How to construct polar codes, IEEE Transactions on Information Theory, 59,10 (2013) 6562–6582.

27. Trifonov, P., Efficient design and decoding of polar codes, IEEE Transactions on Communications, 60,11 (2012) 3221–3227.

28. Mori, R. and Tanaka, T., Performance of polar codes with the construction using density evolution, IEEE Communications Letters, 13,7 (2009) 519–521.

29. Mori, R. and Tanaka, T., Performance and construction of polar codes on symmetric binary-input memoryless channels, IEEE International Symposium on Information Theory (ISIT 2009), July 2009, Korea, In proceedings: 1496–1500.

30. Zhang, Y., Liu, A., Pan, K., Gong, C. and Yang, S., A practical construction method for polar codes, IEEE Communications Letters, 18,11 (2014) 1871–1874.

31. Richardson, T. and Urbanke, R., Modern coding theory, Cambridge University Press, Cambridge, 2008.

32. Chung, S.Y., Richardson, T.J. and Urbanke, R., Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation, IEEE Transactions on Information Theory, 47,2 (2001) 657–670.

33. Wu, D., Li, Y. and Sun, Y., Construction and block error rate analysis of polar codes over AWGN channel based on Gaussian approximation, IEEE Communications Letters, 18,7 (2014) 1099–1102.

34. Niu, K., Chen, K., Lin, J. and Zhang, QT., Polar codes: Primary concepts and practical decoding algorithms, IEEE Communications Magazine, 52,7 (2014) 192–203.

35. Leroux, C., Tal, I., Vardy, A. and Gross, W.J., Hardware architectures for successive cancellation decoding of polar codes, IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2011), May 2011, Czech Republic, In proceedings: 1665–1668.

36. Berhault, G., Leroux, C., Jego, C. and Dallet, D., Partial sums computation in polar codes decoding, IEEE International Symposium on Circuits and Systems (ISCAS 2015), May 2015, Portugal, In proceedings: 826–829.

37. Leroux, C., Raymond, A.J., Sarkis, G., Tal, I., Vardy, A. and Gross, W.J., Hardware implementation of successive-cancellation decoders for polar codes, Journal of Signal Processing Systems, 69,3 (2012) 305–315.

38. Tal, I. and Vardy, A., List decoding of polar codes, IEEE Transactions on Information Theory, 61,5 (2015) 2213–2226.

39. Niu, K. and Chen, K., Stack decoding of polar codes, IET Electronics Letters, 48,12 (2012) 695–697.

40. Chen, K., Niu, K. and Lin. J., Improved successive cancellation decoding of polar codes, IEEE Transactions on Communications, 61,8 (2013) 3100–3107.

41. Niu, K. and Chen, K., CRC-aided decoding of polar codes, IEEE Communications Letters, 16,10 (2012) 1668–1671.

42. Li, B., Shen, H., and Tse, D., An adaptive successive cancellation list decoder for polar codes with cyclic redundancy check, IEEE Communications Letters, 16,12 (2012) 2044–2047.

43. Dizdar, O. and Arikan E., 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, 63,3 (2016) 436–447.

44. Park, Y.S., Tao, Y., Sun, S. and Zhang, Z., A 4.68 Gb/s belief propagation polar decoder with bit-splitting register file, IEEE Symposium on VLSI Circuits Digest of Technical Papers, June 2014, USA, In proceedings: 1–2.

45. Xu, J., Che, T. and Choi, G., XJ-BP: Express journey belief propagation decoding for polar codes, IEEE Global Communications Conference (GLOBECOM 2015), December 2015, USA, In proceedings: 1–6.

46. Yuan, B. and Parhi, K.K., Early stopping criteria for energy-efficient low-latency belief-propagation polar code decoders, IEEE Transactions on Signal Processing, 62,24 (2014) 6496–6506.

47. Gallager, R., Low-density parity-check codes, IRE Transactions on information theory, 8,1 (1962) 21–28.

48. Arikan, E., Polar codes: A pipelined implementation, IEEE 4th International Symposium on Broadband Communication (ISBC 2010), July 2010, Malaysia, In proceedings: 11–14.

49. Pamuk, A., An FPGA implementation architecture for decoding of polar codes, IEEE 8th International Symposium on Wireless Communication Systems (ISWCS 2011), November 2011, Germany, In proceedings: 437–441.

50. Yuan, B. and Parhi, K.K., Architecture optimizations for BP polar decoders, IEEE International Conference on Acoustics, Speech and Signal Processing, May 2013, Canada, In proceedings: 2654–2658.

51. Blahut, R.E., Theory and practice of error control codes, Addison-Wesley Reading (Ma), Michigan, 1983.

52. Goela, N., Korada, S.B. and Gastpar, M., On LP decoding of polar codes, IEEE Information Theory Workshop (ITW 2010), January 2010, Egypt, In proceedings: 1–5.

53. Sivasubramanian, B. and Leib, H., Fixed-rate Raptor codes over Rician fading channels, IEEE transactions on vehicular technology, 57,6 (2008) 3905–3911.

54. Türk, K. and Fan, P., Adaptive demodulation using rateless codes based on maximum a posteriori probability, IEEE Communications Letters, 16,8 (2012) 1284–1287.

55. AbdulHussein, A., Oka, A. and Lampe, L., Decoding with early termination for rateless (luby transform) codes, IEEE Wireless Communications and Networking Conference (WCNC 2008), April 2008, USA, In proceedings: 249–254.

56. AbdulHussein, A., Oka, A. and Lampe, L., Decoding with early termination for Raptor codes, IEEE Communications Letters, 6,12 (2008) 444-446.

57. Orozco, V.L. and Yousefi, S., Trapping sets of fountain codes, IEEE Communications Letters, 14,8 (2010) 755–757.

58. Chen, Y-M., Lee, H-C., Ueng, Y-L. and Yeh, C-Y., Flooding-assisted informed dynamic scheduling for rateless codes, IEEE Wireless Communications and Networking Conference (WCNC 2012), April 2012, France, In proceedings: 173–177.

59. Şimşek, C., and Türk, K., Simplified early stopping criterion for belief-propagation polar code decoders, IEEE Communications Letters, 20,8 (2016) 1515–1518.

60. Şimşek, C., and Türk, K., Simplified minLLR early stopping criterion for belief-propagation based polar code decoders, International Conference on Advanced Technology and Sciences (ICAT'16), September 2016, Konya, In proceedings: 570–574.

61. Sedgewick, K.W.R., Algorithms, 4th edition, Princeton University, USA:Addison-Wesley, 2011.

62. Etesami, O. and Shokrollahi, A., Raptor codes on binary memoryless symmetric channels, IEEE Transactions on Information Theory, 52,5 (2006) 2033–2051.

63. Leroux, C., Raymond, A.J., Sarkis, G. and Gross, W.J., A semi-parallel successive-cancellation decoder for polar codes, IEEE Transactions on Signal Processing, 61,2 (2013) 289–299.

64. Sarkis, G., Giard, P., Vardy, A., Thibeault, C. and Gross, W.J., Fast polar decoders: Algorithm and implementation, IEEE Journal on Selected Areas in Communications, 32,5 (2014) 946–957.

65. Zhang, C. and Parhi, K.K., Low-latency sequential and overlapped architectures for successive cancellation polar decoder, IEEE Transactions on Signal Processing, 61,10 (2013) 2429–2441.

66. Yuan, B. and Parhi, K.K., Low-latency successive-cancellation polar decoder architectures using 2-bit decoding, IEEE Transactions on Circuits and Systems I: Regular Papers, 61,4 (2014) 1241–1254.

67. Stimming, A.B., Raymond, A.J., Gross, W.J. and Burg, A., Hardware architecture for list successive cancellation decoding of polar codes, IEEE Transactions on Circuits and Systems II: Express Briefs, 61,8 (2014) 609–613.

68. Raymond, A.J. and Gross, W.J., A scalable successive-cancellation decoder for polar codes, IEEE Transactions on Signal Processing, 62,20 (2014) 5339–5347.

69. Stimming, A.B., Parizi, M.B. and Burg, A., LLR-based successive cancellation list decoding of polar codes, IEEE Transactions on Signal Processing, 63,19 (2015) 5165–5179.

70. Zhang, C. and Parhi, K.K., Latency analysis and architecture design of simplified SC polar decoders, IEEE Transactions on Circuits and Systems II: Express Briefs, 61,2 (2014) 115–119.

71. Fan, Y. and Tsui, C., An efficient partial-sum network architecture for semi-parallel polar codes decoder implementation, IEEE Transactions on Signal Processing, 62,12 (2014) 3165–3179.

72. Lin, J. and Yan, Z., An efficient list decoder architecture for polar codes, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 23,11 (2015) 2508–2518.

73. Yoo, H. and Park, I.C., Partially parallel encoder architecture for long polar codes, IEEE Transactions on Circuits and Systems II: Express Briefs, 62,3 (2015) 306–310.

74. Suganya, R. and Meganathan, D., High performance VLSI adders, IEEE 3rd International Conference on Signal Processing, Communication and Networking (ICSCN 2015), March 2015, India, In proceedings: 1–7.

75. Ling, H., High-speed binary adder, IBM Journal of Research and Development, 25,3 (1981) 156–166.

76. Dimitrakopoulos, G. and Nikolos, D., High-speed parallel-prefix VLSI ling adders, IEEE Transactions on Computers, 54,2 (2005) 225–231.

CURRICULUM VITAE

Cemaleddin ŞİMŞEK was born in Adana in 05.12.1983. He graduated form Adana Ayşe Atıl Teachers Training High School in 2001, he also joined Karadeniz Technical University Electrical and Electronics Engineering Department same year. He received his B.Sc. in 2005 and right after graduation he started working for the same department as research assistant. He received his M.Sc. in 2008 from same department as well. He finished his Ph.D. classes between 2009-2010 in Högskola i Hamstad as an ERASMUS student in Sweden. After returning to Turkey, he participated several TEKNOGEM projects as R&D engineer. In 2014, he is transferred from Karadeniz Technical University to Gümüşhane University. Currently, he is working for Gümüşhane University as a research assistant. He is married with two children.

PUBLICATIONS:

- Simplified Early Stopping Criterion for Belief-Propagation Polar Code Decoders, Cemaleddin ŞİMŞEK, Kadir TÜRK, IEEE Communications Letters, 20(8): 1515–1518, 2016.
- Low-Complexity Early Termination Method for Rateless Soft Decoder, Cenk ALBAYRAK, Cemaleddin ŞİMŞEK, Kadir TÜRK, IEEE Communications Letters, DOI: 10.1109/LCOMM.2017.2740207, IEEE Early Access Articles.
- Simplified minLLR Early Stopping Criterion for Belief-Propagation Based Polar Code Decoders, Cemaleddin ŞİMŞEK, Kadir TÜRK, ICAT 2017.
- Sign Alterations of LLR Values Based Early Termination Method for LT BP Decoder, Cenk ALBAYRAK, Cemaleddin ŞİMŞEK, Kadir TÜRK, SIU2017.
- Hardware Friendly Early Stopping Structure for Polar Code, Cemaleddin ŞİMŞEK, Cenk ALBAYRAK, Kadir TÜRK SIU2017.
- Hardware Optimization for Belief Propagation Polar Code Decoder with Early Stopping Criteria Using High-Speed Parallel-Prefix Ling Adder, Cemaleddin ŞİMŞEK, Kadir TÜRK, TSP2017.