University of Southampton

Faculty of Physical Sciences and Engineering

School of Electronics and Computer Science

An Investigation on Belief Propagation Decoding Algorithms for Polar Codes

by

Zeynep Burcin Kaykac- 28971701

September 7, 2017

Supervisor: Prof Robert G. Maunder

Second Examiner: Dr Rong Zhang

A dissertation submitted in partial fulfilment of the degree of

MSc Wireless Communication

Contents

List of Figures

List of Tables

List of Acronyms

BP	Belief Propagation
BEC	Binary Erasure Channel
B-DMC	Binary-Discrete Memoryless Channels
BLER	Block Error Rate
E_s/N_o	Energy Per Symbol to Noise Power Spectral Density Ratio
eMBB	enhanced Mobile Broad Band
FEC	Forward Error Correction
SC	Successive Cancellation
SCAN	Soft Cancellation Decoding
SCL	Successive Cancellation List
SMS- BP	Scaled-Min Sum Approximation Belief Propagation
SNR	Signal to Noise Ratio
STG	Stage
LDPC	Low-Density Parity-Check
LLR	Logarithmic Likelihood Ratio
MS- BP	Min Sum Approximation Belief Propagation
PE	Processing Element
QPSK	Quadrature Phase Shift Keying
RCSC	Reduced Complexity Soft Cancellation
XJ- BP	Express Journey Belief Propagation

Nomenclature

K ..... Information bits length

M ..... Code block length

N ..... Mother code block length $2^{ceil(\log_2 M)}$

R ..... Code rate $R = K/M$

W ..... Weight sequence

Q ..... Ordered bits positions sequence

P ..... Puncture set

F ..... Frozen bit set

U ..... Frozen bit position sequence

a ..... Information Block

u ..... Kernal Information Block

x ..... Kernal Encoded Block

b ..... Encoded Block

$\hat{a}$ ..... Recovered Information Block

$\hat{u}$ ..... Recovered Kernal Information Block

$\tilde{x}$ ..... Soft Kernal Encoded Block

$\tilde{b}$ ..... Soft Encoded Block

Abstract

The recently discovered Polar Codes (PC), draw lots of attention as a family of error correction codes in noisy channels because they can achieve the channel capacity. A number of decoding methods have been proposed for polar codes since their introduction, among these, successive cancellation list (SCL) decoding and belief propagation (BP) decoding, which will be called as Original BP in this paper, are the two most popular methods. The novel contribution of this thesis is that proposed different belief propagation (BP) decoding schedules for Polar Codes. In result section, we provide BLER (Block Error Rate) performances as numerical evidence with using MATLAB software to show that performance of each decoding schedules as well as comparing the results with performances of successive cancellation (SC) and SCL decoding methods. In addition, we have also reviewed the proposed new schedules in terms of complexity. According to that results, we observed proposed schedule 1 and 2 increase required steps to complete each iteration but doing less computing, and they present same BLER performance with Original BP. However, we could not observe considerable reducing in terms of the decoding time although they required less computing for each step. Furthermore, we observed proposed schedule 3 which is called as proposed SCAN (soft-cancellation) with the same schedule of SC, has better performance than Original BP and we succeed the reduced the complexity and required space in the memory according to original SCAN by not computing frozen bits LLRs when we updating right-to-left propagation messages.

Acknowledgement

The author would like to express her sincere gratitude to her supervisor Prof. Robert G. Maunder , for the continuous support of her study, for his patience, motivation, and immense knowledge. His guidance helped in all the time of research and writing of this thesis. Also thanks to my family who endured this long process with me, always offering support and love.

1 Introduction

This section presents why the title of the thesis is selected and what is the contribution of the thesis in Subsection 1.1 and 1.2, respectively . Moreover, Subsection 1.3, presents previous attempts in the literature for BP. The gap in the literature and our approach will be presented in Subsection 1.4 and as final, Subsection 1.5 introduces thesis organization.

1.1 Importance of Polar Codes

In a globalized world, effective communication is almost a bare necessity. We need communication systems which are more reliable, faster and serve more users at the same time because of the increasing communication demand, rising using of social media, continuously uplink and downlink and due to many more reasons. However, because of the hostile mobile channel characterize it is not easy to maintain the required communication integrity. We know that for 5th (5G) and future generations we will need significant data rate, coverage, and service gain to provide effective wireless communication. Forward Error Correction (FEC) coding methods such as Turbo codes and low-density parity-check (LDPC) codes is needed because of the transmission errors in noisy channels. They can provide very high data rates and integrity for qualified communication by making error correction. At this point, recently introduced Polar Codes (PC) by Arıkan [1] attract attention because they may provide superior performance than Turbo Code and LDPC [5,6] owing to their capacity-achieving performance, excellent error correction performance and fine rate flexibility [3]. Moreover, it can provide lower encoding and decoding complexity than others. Partly due to the many reasons mentioned above, polar codes have been adopted as a candidate of Forward Error Correction (FEC) coding method in control channels of enhanced Mobile Broad Band (eMBB) scenario which is a possible choice of 5th generation (5G) mobile communication [3,4].

1.2 Importance of BP Decoding

After the discovery of Polar Codes, a number of decoding methods have been proposed, among these, successive cancellation (SC)[1] decoding and belief propagation (BP)[16] provide the basis for other decoding methods [14]. However, SC suffers from long latency to achieve a high throughput, although it requires less computation [14,20,23]. Hence, several methods have been proposed such as list successive cancellation decoding (SCL) to improve the error-correcting performance for short and moderate code lengths [14]. On the other hand, BP decoders have some advantage due to its parallel processing nature[14,18,20,23]. Therefore, compared with SC , polar BP decoders are more attractive for low-latency applications. Another advantage of BP decoding is that it provides soft outputs that are necessary for joint detection and decoding[15].

1.3 Previous Attempts for Improving BP Decoding

BP decoding is an iterative decoding algorithm where information changes iteratively between nodes according to a particular schedule, which defines which node will be activated in which step [18]. Due to iterative nature of BP decoding, the required latency and energy dissipation increases the requirement computation complexity and required memory space[14,18,20,23]. Many studies have been done in the literature to get over these disadvantages of BP. In this subsection, we look briefly at the some of the important studies which are done for the last 4 years, starting with 2013.

In 2013, to reduce requirement computation complexity in [18] scaled-min sum approximation(SMS)-BP decoding is proposed and it is claimed that SMS-BP achieves error performance as original BP with low complexity. In [16], Fayyaz and Barry are introduced soft-cancellation decoding which is called SCAN decoding which achieves performance gain with lower complexity compared to BP, and also in [17], they propose a memory-efficient version of SCAN decoder.

In 2014, to improve the performance of original BP , in [19] check nodes are added to each node of BP. Moreover, in [20] a simplified BP decoder is proposed, according to this study to achieve the same performance as BP with the significantly low complexity they give the priority to the frozen nodes during the procedure of BP decoding.

In 2015, express-journey (XJ)-BP decoding method which simplifies SMS-BP decoding with almost same error performance is proposed in [21]. In [22], efficient early termination schemes for BP decoding are proposed which called as a low-complexity LLR- magnitude-aided (LMA) scheme also for the better sensitivity and lower complexity they proposed cyclic redundancy check (CRC)- aided (CA) scheme.

In 2016, a stage-combined BP decoder is proposed in [23], which is considerably reduced the message memory requirement of BP and also which is increased the speed of BP decoding. In 2017, reduced complexity soft cancellation (RCSC) method is proposed which is achieved comparable or even better performance than SCAN with reduced memory, in [24]. An improved BP decoding by adopting parity-check matrices is proposed in [25] which have remarkable performance gain over original BP with relatively low decoding as well as even can compete with CRC-SCL decoder.

1.4 Summary of Thesis Contributions

As you can see in the above subsection although many studies carried out in literature there is still a gap for BP decoding, especially the area of examining different BP decoding schedules. This papers aim that to provide a contribution about that point. We have proposed three different belief propagation (BP) decoding schedules for Polar Codes, also comparing these schedules with SC, SCL and Original BP decoding in terms of BLER performances. Moreover, we also have examined their complexity. According to the results we have observed, Original BP has slightly better performance than SC for moderate and longer code lengths, although required more memory space and more computation complexity.

Moreover, we observed proposed schedule 1 and 2 increase required steps to complete each iteration but doing less computing, and they present same BLER performance with Original BP. However, we could not observed considerable reducing in the decoding time although they required less computing for each step. Furthermore, we observed proposed schedule 3 which is called as proposed SCAN with same schedule of SC, has better performance than Original BP and we succeed the reduced the complexity and required space in the memory according to original SCAN by not computing frozen bits LLRs when we updating right-to-left propagation messages.

1.5 Thesis Organization

The rest of this paper is organized as follows. In section 2, which is called background section, we briefly review the polar encoding and decoding processes and we describe common decoding algorithms which are SC, SCL, and BP. In section 3, we present all planning details about the project such as top level objectives, tasks, and risks, also by showing time flow of the workload on the Gantt chart. In next section, which is called implementation section, explanations about proposed BP schedules are presented. In section 5, we analyze the obtained simulation BLER results considering coding rate, block length and list size as well as considering different decoding methods. In section 6, is a critical evaluation section, which reflects on the work that has been carried out, also contains ideas on what can be done for further studies. And finally, section 7 is the conclusion section.

2 Background

In this section, which is divided into two main parts as 2.1 and 2.2, we briefly review the Polar Codes also giving some examples. In subsection 2.1, will be mentioned about polar encoding and decoding processes, and in subsection 2.2, common decoding algorithms which are SC, SCL, and BP will be described briefly.

2.1 Polar Codes

Polar codes take information bits with length K and add $(M - K)$ redundancy bits to generate code block with length M, based on the following channel polarization principles(see Appendix A): polar codes divide channel into sub-channels with some of them have good quality while some of them have poor quality, then sending required information bits at "good" positions those are strongly guaranteed the reliability of transmission and sending fixed "0" at "bad" positions which are highly unreliable [8], hence, these positions cannot be used for sending required information bits. In [1], those "0" bits are called "frozen" bits and their positions are known by both the encoder and the decoder. Polar encoding is applied before modulation in transmitter, and polar decoding is applied after demodulation in receiver, as it can be seen in the Fig.1.

Code rate of polar code can be described as $R = K/M$ and N is mother code block length which can be compute with $2^{ceil(\log_2 M)}$. Fig.2 shows that both polar encoder and decoder include three main components which will be detailed in the next sections.

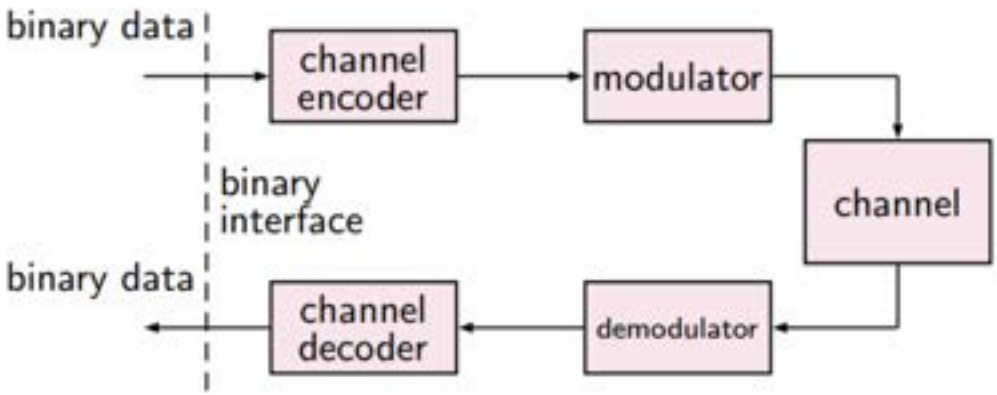

Figure 1: Block Diagram of Communication Channel with Channel Encoding [7]

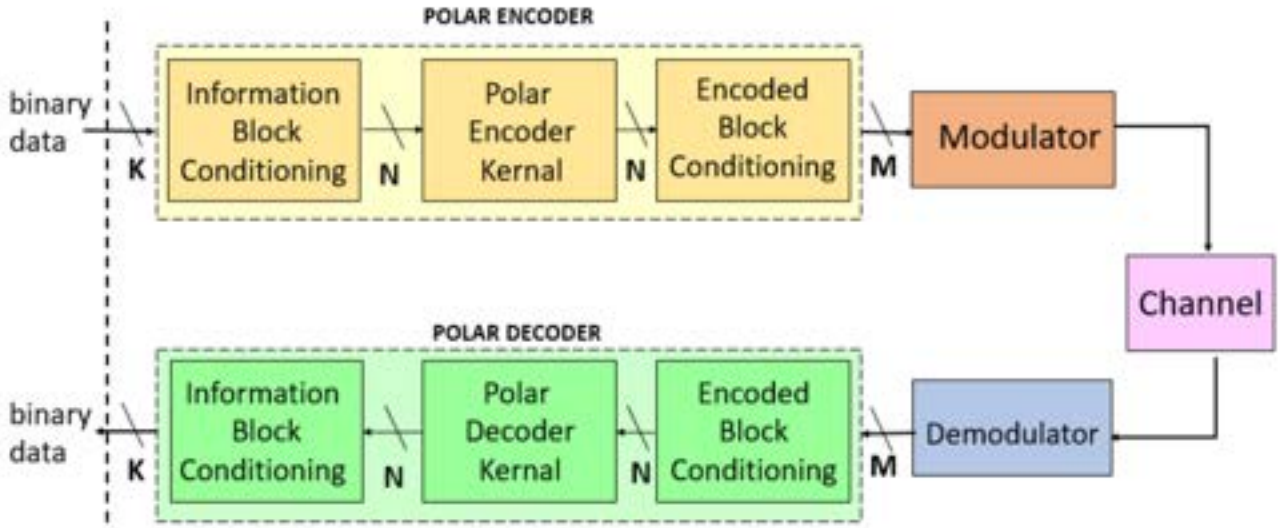

Figure 2: Polar Encoder and Decoder with Their Main Components

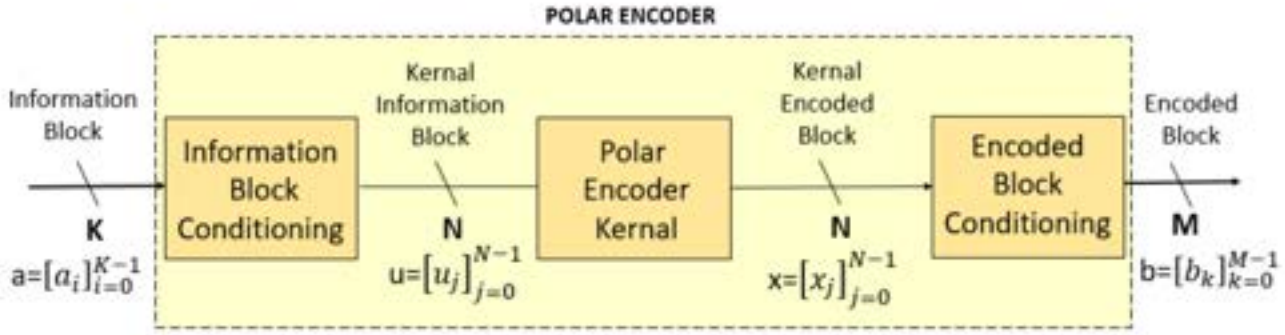

Figure 3: Block Diagram of Polar Encoding

2.1.1 Polar Encoder

As we mention before, polar encoder includes three main components which are *information block conditioning*, *polar encoder kernal* and *encoded block conditioning*, respectively. All process can be seen from the Fig.3.

Information block conditioning [4] takes a row vector such that $a = [a_i]_{i=0}^{K-1}$ as input where $a_i \in 0, 1$ and K is the block size of information bits. Information block conditioning generates N-K redundant bits with using different methods such as frozen bits, cyclical redundancy check (CRC) bits or parity check (PC)- frozen bits and obtain a row vector $u = [u_j]_{j=0}^{N-1}$ which is called as *kernal information block* with block size N, which should be a power of 2 and greater than K. If Information block conditioning use frozen bits method, it may always adopt fixed "0" bits to generate redundant bits and also it have to interlace them into positions using various methods such as rate matching method. These positions of redundant bits, which is also called as frozen bits, should know by the polar decoder.

After this process *polar encoder kernal* takes kernal information block as an input and generate $x = [x_j]_{j=0}^{N-1}$ which called *kernal encoded block*. The relationship between them can be given with following equation where $F^{\otimes n}$ is described as $(n = \log_2 N)^{th}$ Kronecker power of kernal matrix.

$$x = uF^{\otimes n} \tag{1}$$

Kronecker power of kernal matrix can be obtain recursively, for examle to obtain $F^{\otimes n}$ each 1 should be replace with $F^{\otimes(n-1)}$ while each 0 replace with 2 x 2 zero matrix as it can be seen in the below;

$$F^{\otimes n} = \begin{bmatrix} F^{\otimes(n-1)} & 0 \\ F^{\otimes(n-1)} & F^{\otimes(n-1)} \end{bmatrix}$$

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

$$F^{\otimes 2} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$F^{\otimes 3} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Polar kernal operation can be represented by the recursive graph representation of the generator matrix $F^{\otimes n}$ as it can be seen from the Fig.4, Fig.5 and Fig.6. The input of this representation is kernal information bits and it is represented by u row matrix and output is the kernal encoded bits which is represented by x. Here, each modulo-2 addition $\oplus$ can compute with $eXlusive-OR$ (XOR)operation and this gives totally $(N/2)\log_2 N$ XORs.

After polar encoder kernal,*encoded block conditioning* takes $x = [x_j]_{j=0}^{N-1}$ as input and generate $b = [b_k]_{k=0}^{M-1}$ row vector with block size of M where $b_k \in 0, 1$. Here M should be greater than K but it can be greater or smaller than N. Encoded block conditioning can use different methods to obtain encoded block b such as repetition, shortening or puncturing methods. Although, repetition repeats some of the bits in the kernal encoded block, shortening and puncturing removes some bits. Encoded block b can be provided to a modulator after this process to get the some advantages of modulation in the hostile communication environment before its transitions in the channel.

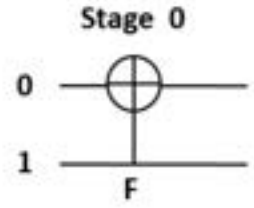

Figure 4: Graph Representation of the Generator Matrix F

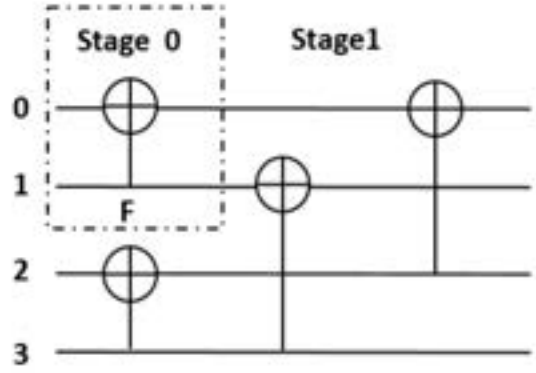

Figure 5: Graph Representation of the Generator Matrix $F^{\otimes 2}$

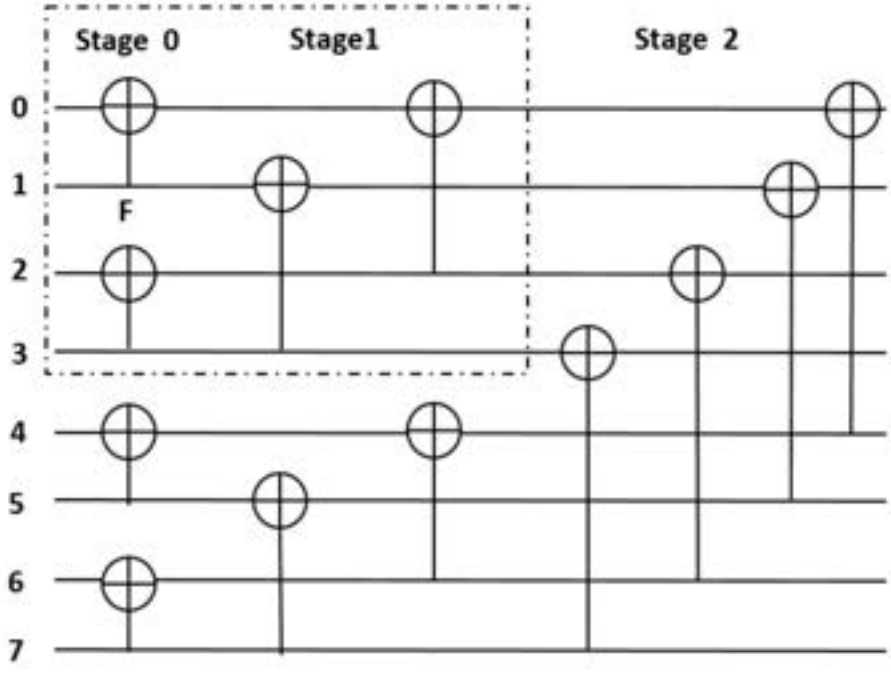

Figure 6: Graph Representation of the Generator Matrix $F^{\otimes 3}$

2.1.1.1 Example 1

Example 1: Polar Encoding with Using Rate Matching Method

Rate Matching Method is the one of the encoder methods. In [9] Quasi-uniform Puncturing (QUP) based rate matching methods have been discussed to construct polar code. For this method reliability of each synthesized sub-channel are computed using Density Evolution based on Gaussian approximation (DE/GA)[9], then high reliable sub-channels chosen for transmit the information bits while unreliable ones set to to zero which are called as frozen bits. Moreover, both encoder and decoder have to compute this uniqueness set before encoding and decoding. However, because of the complexity of this method in [10] the positions of the frozen bits are computing in a simple way for any coding rate considering mother code length N as nested combination of two polar codes of length N/2. Hence, it constructs a ordered sequence of bits positions (index sequence) for N using subset of ordered sequences of N/2 polar codes. As a result, rate matching method can be performed at low complexity and with fine granularity. Consequently, we will follow the structure of [10] as polar encoding scheme to obtain encoded word for this example. The following notations are used throughout this example and also throughout this paper;

K; Information bits length

M; Code block length

N; Mother code block length $2^{ceil(\log_2 M)}$

R; Code rate $R = K/M$

W; Weight sequence

Q; Ordered bits positions sequence

P; Puncture set

F; Frozen bit set

U; Frozen bit position sequence

We will consider an example for encoding $(M = 8, K = 4, R = 1/2)$ and information bits a=[1011] and we will get encoded bits b=[10100101] with length M=8.

Step 1: Compute weight sequence $W_0^{N_{max}-1}$ where $N_{max} = N = 2^{ceil(\log_2 M)}$.

Here, $N = 8$ and it is equal the M. Hence, $W_0^{N_{max}-1} = W_0^7$. We can compute weight sequence with following equations where $i = B_{n-1}B_{n-2}..B_0$ and $B_j \in 0, 1$ and $j = [0, 1, ..n-1]$ [10];

$$W_i = \sum_{j=0}^{n-1} B_j * 2^{j*1/4} \tag{2}$$

Here $n = \log_2 N$ and $n = 3$. Hence;

$$W_i = \sum_{j=0}^{n-1} B_j * 2^{j*1/4} = B_0 * 2^{0*1/4} + B_1 * 2^{1*1/4} + B_2 * 2^{2*1/4} \tag{3}$$

$(i = 000) \iff W_0 = 0 * 2^{0*1/4} + 0 * 2^{1*1/4} + 0 * 2^{2*1/4} = 0$

$(i = 001) \iff W_1 = 1 * 2^{0*1/4} + 0 * 2^{1*1/4} + 0 * 2^{2*1/4} = 1$

$(i = 010) \iff W_2 = 0 * 2^{0*1/4} + 1 * 2^{1*1/4} + 0 * 2^{2*1/4} = 1.1892$

$(i = 011) \iff W_3 = 1 * 2^{0*1/4} + 1 * 2^{1*1/4} + 0 * 2^{2*1/4} = 2.1892$

$(i = 100) \iff W_4 = 0 * 2^{0*1/4} + 0 * 2^{1*1/4} + 1 * 2^{2*1/4} = 1.4141$

$(i = 101) \iff W_5 = 1 * 2^{0*1/4} + 0 * 2^{1*1/4} + 1 * 2^{2*1/4} = 2.4142$

$(i = 110) \iff W_6 = 0 * 2^{0*1/4} + 1 * 2^{1*1/4} + 1 * 2^{2*1/4} = 2.6034$

$(i = 111) \iff W_7 = 1 * 2^{0*1/4} + 1 * 2^{1*1/4} + 1 * 2^{2*1/4} = 3.6034$

Step 2: Compute ordered bits positions sequence $Q_0^{N_{max}-1}$. To obtain ordered bits positions sequence $Q_0^{N_{max}-1}$ we should sort $W_0^{N_{max}-1}$ such that

$$W_{Q0} \leqslant W_{Q1} \leqslant W_{Q2} \leqslant ... \leqslant W_{QN_{max}-1}$$

and we will save the corresponding index sequence as $Q_0^{N_{max}-1}$.

$$W_0 \leqslant W_1 \leqslant W_2 \leqslant W_4 \leqslant W_3 \leqslant W_5 \leqslant W_6 \leqslant W_7$$

$$Q_0^{N_{max}-1} = Q_0^7 = \begin{bmatrix} 0 & 1 & 2 & 4 & 3 & 5 & 6 & 7 \end{bmatrix}$$

Step 3: Generate a $(N-M)-$entry puncture position set as row vector

Here N=M=8, so there is no need to do puncturing hence P_0^{N-M-1} will be an empty matrix

$$P_0^{N-M-1} = \begin{bmatrix} \emptyset \end{bmatrix}$$

Step 4: Generate a $(M-K)-$entry frozen set as a row vector F_0^{M-K-1}

To obtain frozen set we will select the first $(M-K) = 4$ entries from the Q_0^7 whose values are smaller than N and do not exist in P_0^{N-M-1}, here P is the empty matrix so all values not exist in P.

$$F_0^{M-K-1} = F_0^3 = \begin{bmatrix} 0 & 1 & 2 & 4 \end{bmatrix}$$

Step 5: Generate a $(N-K)-$entry frozen positions as row vector U_0^{N-K-1}

To obtain U_0^{N-K-1} we will use following equations;

$$U_0^{N-K-1} = P_0^{N-M-1} \cup F_0^{M-K-1} \tag{4}$$

Here;

$$P_0^{N-M-1} = \begin{bmatrix} \emptyset \end{bmatrix}$$
$$F_0^{M-K-1} = F_0^3 = \begin{bmatrix} 0 & 1 & 2 & 4 \end{bmatrix}$$

Hence;

$$U_0^{N-K-1} = U_0^3 = \begin{bmatrix} 0 & 1 & 2 & 4 \end{bmatrix}$$

Step 6: Initialize the input binary vector x_0^7 by the setting the zero onto the positions defined by U_0^{N-K-1} (frozen bits) and putting $K = 4$ information block $a = [1011]$ onto the remaining entries. Here,

we used red colour to represent frozen bits and black color to represent information bits.

$$x_0^7 = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Step 7: Polar Encoder step to obtain encoded block $b_{k=0}^{M-1}$ is basically modulo-2 computing

$$b_{k=0}^{M-1} = x_0^{N-1} * F^{\otimes n} \tag{5}$$

here;

$$F^{\otimes 3} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

As a result, encoded block $b_{k=0}^{M-1}$

$$b_0^7 = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Step 8: Shortening

Here we did not do puncturing so there is no need to do shortening, because length of N and M equals each other . However, if there is a need to do shortening, we should generate $b_i = b_{k=0}^{M-1}$ when $k \notin P_0^{N-M-1}$ to obtain encoded block.

2.1.1.2 Example 2

Example 2: Using Graph Representation Method to Obtain Encoded Block

Graph representation method also can be used to explain encoding process more clearly. We mentioned before how polar kernal operation can be represented by the recursive graph representation of the generator matrix $F^{\otimes n}$. Similarly we can compute encoded bits using graph methods. Here $F^{\otimes n} = F^{\otimes 3}$ and as input frozen bit pattern can be taken, as it can be seen from the Fig.7 each step can be computed with modulo−2 addition $\oplus$ such that each of them an $eXlusive - OR$ (XOR) operation and this gives totally $(N/2)\log_2 N = 12$ XORs operation. At the end of the operations we got the encoded bits sequence.

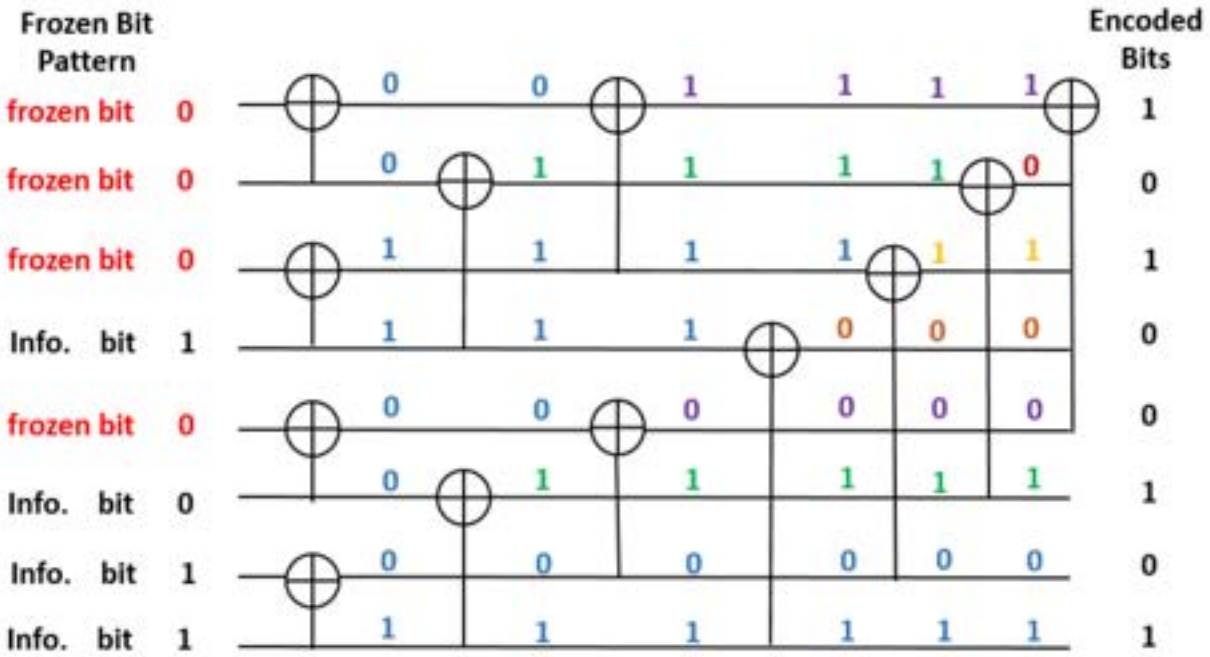

Figure 7: Using Graph Representation Method to Convert the $K = 4$ Information Bits a=[1011] into the $M = 8$ Encoded Bits b=[10100101].

2.1.2 Polar Decoder

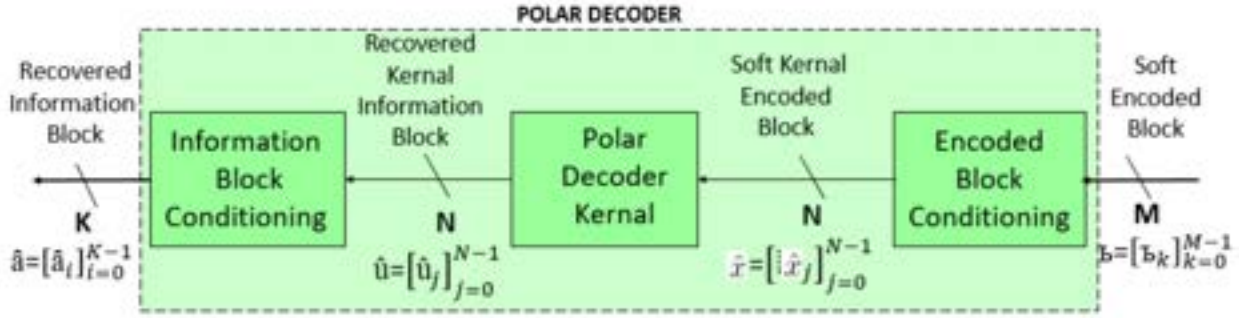

Figure 8: Block Diagram of Polar Decoder

As we mentioned before communication channel has a hostile environment which causes distortions on the transmitted signal such as adding noise. As a result of this, generally demodulator do not have the absolute confidence about the value of the recovered information bits [4]. To express the confidence about encoded blocks, demodulator can generate soft encoded block [4] such that each soft bit represents with form of Logarithmic Likelihood Ratio (LLR). For example, assume soft encoded block is $\hat{b} = [\hat{b_k}]_0^{M-1}$ and we can express LLR value like that;

$$\hat{b_k} = LN\left[\frac{Pr(b_k = 0)}{Pr(b_k = 1)}\right] \tag{6}$$

where $Pr(b_k = 0)$ is the probability of the real value of $\hat{b_k}$ is zero and $Pr(b_k = 1)$ is the probability of the real value of $\hat{b_k}$ is one, and their sum must be one.

$$Pr(b_k = 0) + Pr(b_k = 1) = 1 \tag{7}$$

Here the sign of the LLR express the value of the bits [4]. For example if $\hat{b_k} > 0$ demodulator decide the value of $\hat{b_k} = 0$ while $\hat{b_k} < 0$ demodulator decide the value of $\hat{b_k} = 1$ with greater confidence. Also, magnitude of LLR express the confidence level of demodulator [4], hence recovered bits with higher magnitudes will be more reliable.

As we mention before, polar decoder includes three main components which are *encoded block conditioning*, *polar decoder kernal* and *information block conditioning*, respectively. They have showed in Fig.8.

Encoded block conditioning [4] takes a row vector of soft encoded block $\hat{b} = [\hat{b_k}]_0^{M-1}$ as input, where M is the block size and it can be any value bigger than K, to obtain a row vector $\hat{x} = [\hat{x_j}]_0^{N-1}$, which is called as *soft kernal encoded block* with block size N which must be power of 2. In order to convert the M encoded LLRs into the N kernal encoded LLRs infinite- valued LLRs and 0- valued LLRs can be interlaced with the soft encoded block to occupy the positions which were removed by shortening and puncturing, respectively. Moreover, if interleaving was employed into the encoded block conditioning component of the polar encoder, deinterleaving can employed by encoded block conditioning component of the polar decoder [4].

After this process *polar decoder kernal* [4] takes soft kernal encoded block $\hat{x}$ as an input and generate $\hat{u} = [\hat{u_j}]_{j=0}^{N-1}$ which called *recovered kernal information block*. Here, polar decoder kernal can use different algorithms for decoding, most common ones listed in below.

- Successive Cancellation (SC)
- Successive Cancellation List (SCL)
- Belief Propagation (BP)

The last component of polar decoder is *information block conditioning* [4] which takes recovered kernal information block as an input and generates recovered information block which is $\hat{a} = [\hat{a_i}]_{i=0}^{K-1}$ with K block size. To obtain $\hat{a}$ from the $\hat{u}$ all redundant bits can be removed.

In next three sections we will explain decoding algorithms which have mentioned before.

2.2 Polar Decoding Algorithms

2.2.1 Successive Cancellation (SC)

In [1] is provided that Polar codes achieve the capacity of symmetric binary - input discrete memoryless channels (B-DMCs) with successive cancellation (SC) decoding with complexity $O(N \log N)$ when the code length N and this paper won the 2010 "Information Theory Society Best Paper Award".

To operate SC decoding [4], similar graph structure with basic computations units can be used which have explained before in the polar encoding chapter. However, this time computations will not be performed only from left to right, also right to left operations will be computed. The starting point of computations units will be from right to left side. Here, we should highlighted that there is data dependencies.

In [4] basic computation units of SC decoding have been separated into three group. Order of this three basic computation units depend on two important input. First one is the available LLRs which are provided from right- hand connections and similarly second one available bits which are provided from left-hand connections. In Fig.9, Fig.10 and Fig.11, these three types of basic computation units are showed.

The first step is the calculate left- hand edge LLRs value when both right- hand edge LLRs values available [4]. For this calculation we will use Eq.8 and Eq.9. Here, this two equations are approximated of each other and Fig.9 represent their function graph. Here, $\tilde{x_a}$, $\tilde{x_b}$ and $\tilde{x_c}$ are all LLRs value and $sign(.)$ returns -1 if the LLRs value negative, while return $+1$ if it is positive.

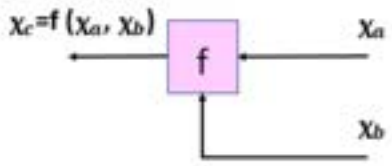

Figure 9: Basic Computation Unit: f Function Graph Representation [4]

$$\tilde{x_c} = f(\tilde{x_a}, \tilde{x_b}) \tag{8}$$

$$\begin{aligned} &= 2tanh^{-1}(tanh(\tilde{x_a}/2)tanh(\tilde{x_b}/2)) \\ &\approx sign(\tilde{x_a})sign(\tilde{x_b})min(|\tilde{x_a}|, |\tilde{x_b}|) \end{aligned} \tag{9}$$

This f function should be computed until get a LLR value at left- hand edge, then according to sign of LLR which we will called it second step, a bit $\tilde{U_a}$ will be available. If $\tilde{x_c} < 0$ the bit value will be 1 while $\tilde{x_c} > 0$ the bit value will be 0. Here , the important thing is that if connection corresponding to a redundant bit such as a frozen bit within the kernal information block, it should be take as it is know, regardless of the sign of obtained LLR. In that point we can said again the positions of frozen bits should know both encoder and decoder. Hence, with provided bit $\tilde{U_a}$ and LLRs values $\tilde{x_a}$ and $\tilde{x_b}$, we can compute another LLR value for different left- hand edge connection with using Eq.10 and Eq.11, also their graph representation have been showed in Fig.10.

$$\tilde{x_d} = g(\tilde{x_a}, \tilde{x_b}, \tilde{U_a}) \tag{10}$$

$$= -1^{\tilde{U_a}}\tilde{x_a} + \tilde{x_b} \tag{11}$$

After second step is computed until get a LLR value at left- hand edge , a new bit $\tilde{U_b}$ will be available according to sign of LLR, which we will called it third step, and this enables the partial sum computations of bits $\tilde{U_c}$ and $\tilde{U_d}$ as it can be shown from the Fig.11. We used following equations for calculate these

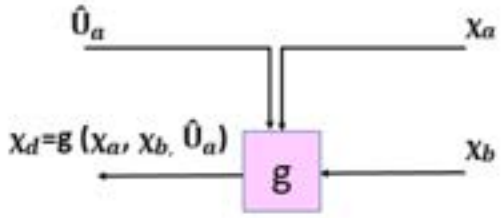

Figure 10: Basic Computation Unit: g Function Graph Representation [4]

bits;

$$\tilde{U}_c = XOR(\tilde{U}_a, \tilde{U}_b) \tag{12}$$

$$\tilde{U}_d = \tilde{U}_b \tag{13}$$

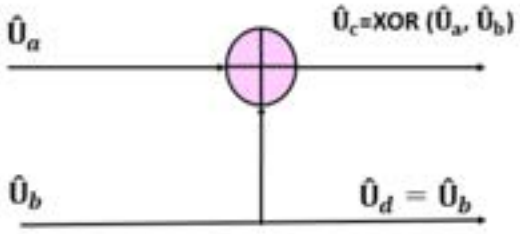

Figure 11: Basic Computation Unit: Partial sum Calculation Graph Representation [4]

Now we will give an example to explain more clearly how the polar decoder making decoding with using SC decoding method.

2.2.1.1 Example 3

Example 3: Using Graph Representation Method to Obtain Decoded Block with Using SC Decoding

Step 0: We should highlighted that frozen bits sequence are know by decoder. It means that decoder knows that exactly positions of frozen bits before starting decoding process. And we also assumed that

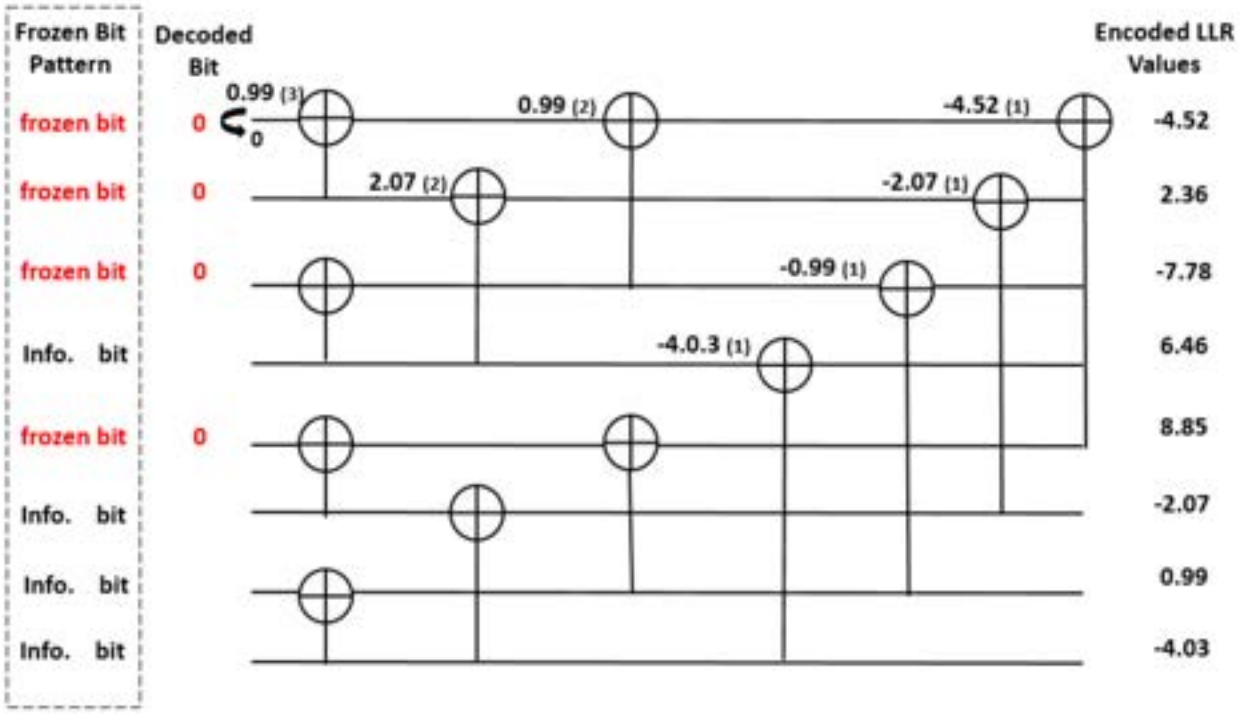

Figure 12: Step 1,2 and 3 Graph Representation of Polar Decoding for SC Decoder

we will know channel output LLRs before starting the decoding. Our information block is $a = [1011]$ and we will use (M=8, K=4) polar code for decoding because of the its simplicity.

Step 1,2 and 3: We will compute left-hand edge LLRs values using right- hand edge LLR values with the aid of f function which is given in Eg.10.

Firstly, we will compute first step LLRs which is represented in Fig.13 as (1), then using this new LLRs we will compute second step (2) LLRs values and finally using these LLRs of (2) we will compute LLR of (3). After that, we should decide encoded value considering sign of (3), in our example, it is positive so the value of encoded bit should be 0. However, we should emphasis that even the LLR value of (3) is negative we should encoded it as positive because we already know that this position belong the frozen bit.

Using the following equation, we can obtain first step LLRs (1), and similarly LLRs of step (2) and step

(3)

$$\tilde{x_c} \approx sign(\tilde{x_a})sign(\tilde{x_b})min(|\tilde{x_a}|, |\tilde{x_b}|)$$

$$-4.52_{(1)} \approx sign(-4.52)sign(8.85)min(|-4.52|, |8.85|)$$

$$-2.07_{(1)} \approx sign(2.36)sign(-2.07)min(|2.36|, |-2.07|)$$

$$-0.99_{(1)} \approx sign(-7.78)sign(0.99)min(|-7.78|, |0.99|)$$

$$-4.03_{(1)} \approx sign(6.46)sign(-4.03)min(|6.46|, |-4.03|)$$

Step 4: Now after step 3, a bit $\tilde{U_a}$ will be available as $\tilde{U_a} = 0$. Hence, we can compute another LLR value for different left- hand edge connection with using g function which is given in Eq.10. As a result, from the following calculations we can get new LLR value as 3.06, and we know that it is positive and we can obtain new encoded bit as 0, but as we said before, here is the frozen bit positions and we already expected that we will get zero.

$$\tilde{x_d} = g(\tilde{x_a}, \tilde{x_b}, \tilde{U_a}) \tag{14}$$

$$\begin{aligned} \tilde{x_d} &= -1^0 0.99 + 2.07 \\ \tilde{x_d} &= 3.06 \end{aligned} \tag{15}$$

Step 5: After step 4 we obtain two bit which are represented with $\tilde{U_a}$ and $\tilde{U_b}$ in the left-side edge and this enables the partial sum computations of bits using Eg.12 and 13. As a result, we have obtained new bits as $\tilde{U_c} = 0$ and $\tilde{U_d} = 0$.

Step 6: Now after step 5, we have a bit $\tilde{U_a}$ again, so we can compute another LLR value for different left- hand edge connection with using g function again as it can be seen from Fig.13 and Fig.14.

After that by repeating the same steps we can complete all graph and we can obtain decoded word as it can be seen from the Fig.15. After step (11) you can see that actually how decoded process can be done as N/2 code length which are represented different colours. Moreover, if we remove all frozen bits from the decoded bits we can obtain our decoded word $\tilde{a} = [1011]$.

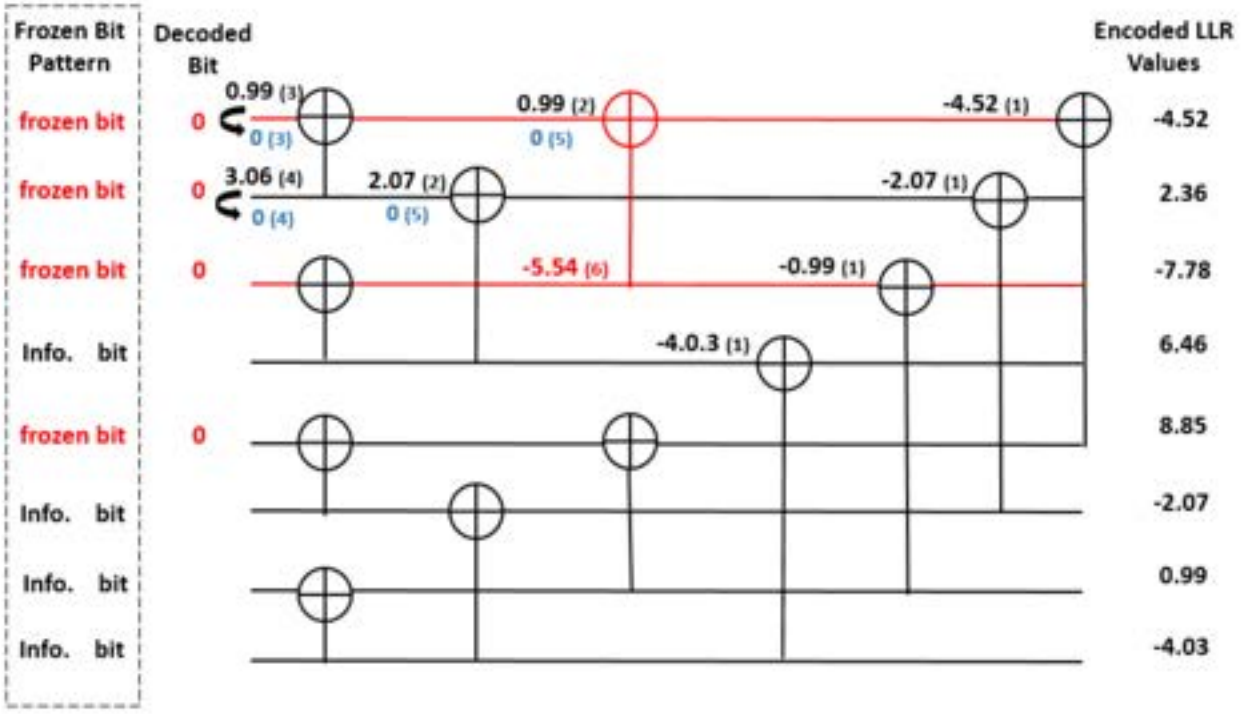

Figure 13: Step 6 Graph Representation of Polar Decoding for SC Decoder

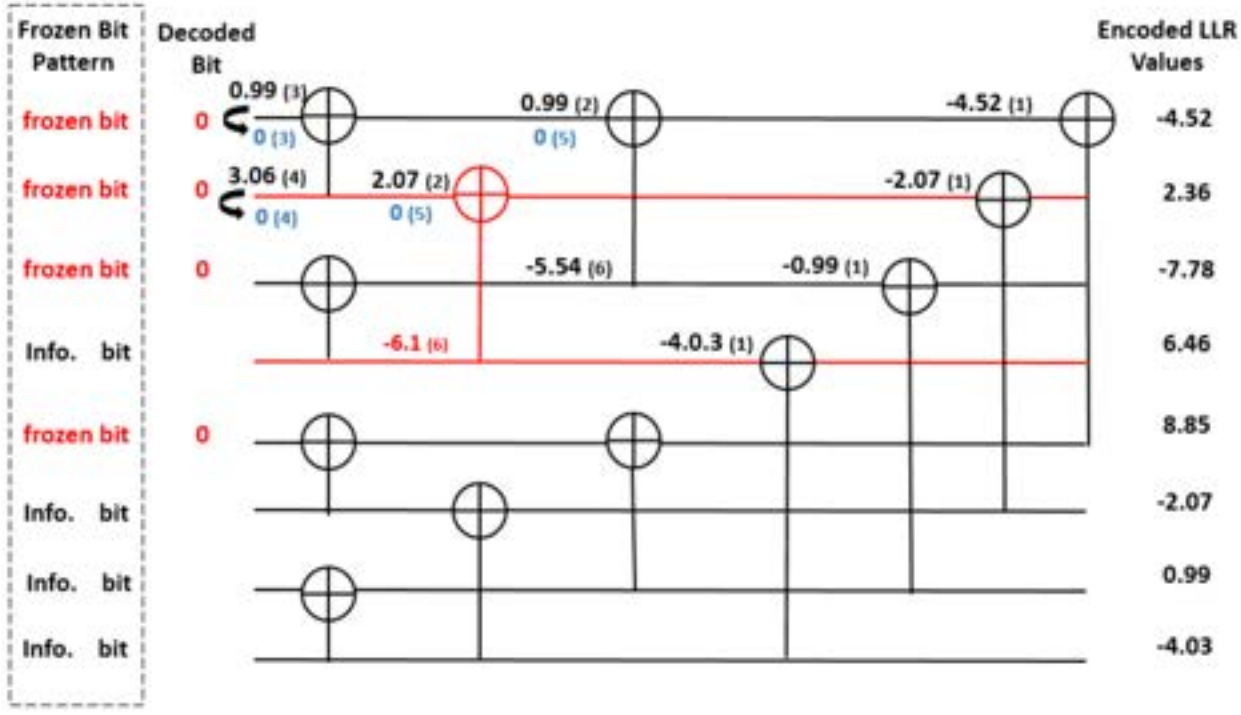

Figure 14: Step 6 Graph Representation of Polar Decoding for SC decoder (Other)

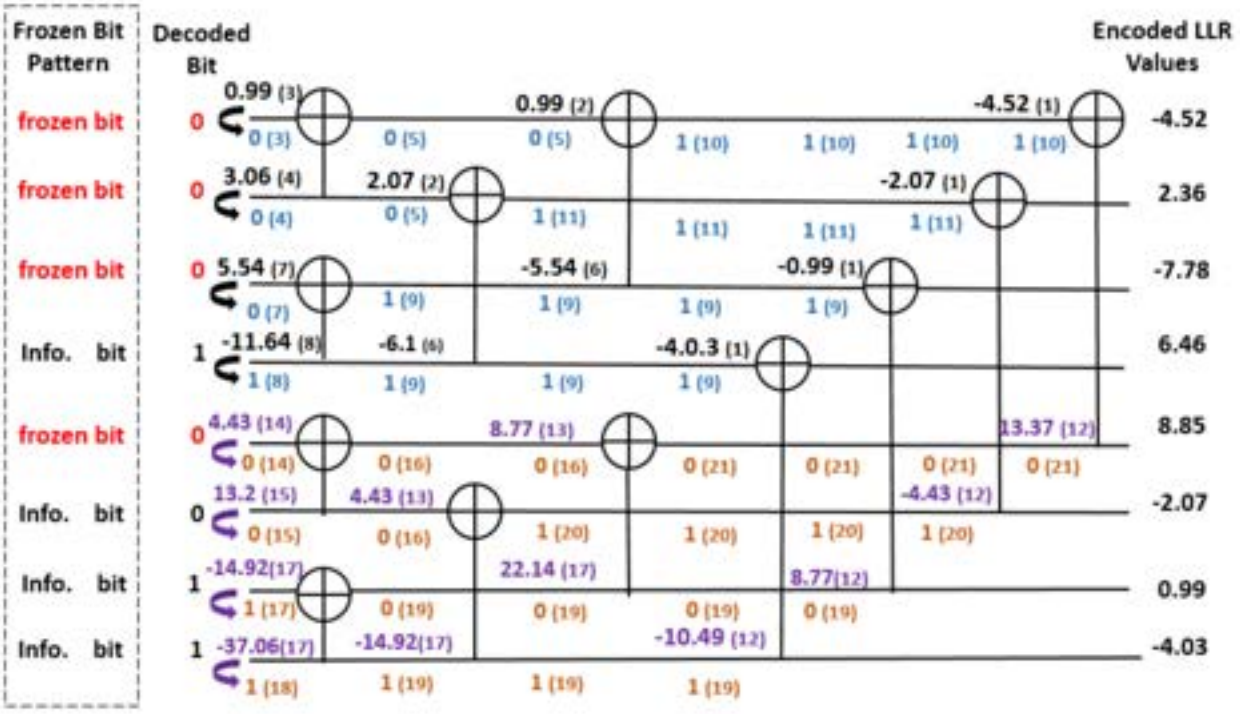

Figure 15: Graph Representation of Polar Decoding for SC Decoder

2.2.2 Successive Cancellation List (SCL)

Successive Cancellation List (SCL) decoding method was introduced by Tal and Vardy [11] to improve performance of SC decoding because empirical studies indicate that for short and moderate block lengths, SC decoding does not perform as well as turbo codes or low density parity check (LDPC) codes. As we mentioned before (SC) decoding complexity is $O(N \log N)$ when the code length N [1], however the complexity of SCL is $O(LN \log N)$ [11] where the L denotes the *list size*. L can be any number but it should be integer, in general it is taken as power of 2, and as mentioned in [11] larger values of L occupy more memory space and increases running time although causes lower error rates.

We have mentioned that in the SC decoding section each recovered information bits depends on two main things [4], the first one is the sign of the corresponding LLR and the other one dependency with previous recovered bits, hence, SC causes cascading errors if an error happens for a particular bits. To prevent this SCL decoding process considers a list of alternative values for the preceding information

bits.

SCL process start with L include only a single kernal information block with length zero bits [4]. If decoding process reaches a frozen bit, a bit value of zero is added at the end of the list, while the process reaches a information bits two replicas of the list of candidate bits are generated which are called as candidate paths [12]. Here, the first replica adds bit value as zero at the end of the list while the second one adds bit value as one. The number of decoding paths is doubled until the list size reaches decided$L$ size [12] after that, in the next step, 2L paths are generated and the best L paths are selected based on their path metrics (PMs). Moreover, to decide which path will be pruned following equations can be used[4], here to select the best path, lowest metric should be chosen where $\tilde{x}_{l,j}$ represent corresponding LLR, $\phi_{l,j}$ is the metric and $\tilde{u}_{l,j}$ bit value which will be added to the list with position $j \in [0, N-1]$;

$$\phi_{l,j}(\tilde{u}_{l,j}) = \phi_{l,j-1} + In(1 + e^{-(1-2\tilde{u}_{l,j})\tilde{x}_{l,j}}) \tag{16}$$

$$\approx \begin{cases} \phi_{l,j-1} & \text{if } \tilde{u}_{l,j} = \dfrac{1}{2}(1 - sign(\tilde{x}_{l,j})) \\ \phi_{l,j-1} + |\tilde{x}_{l,j}| & \text{otherwise} \end{cases} \tag{17}$$

Also we can represent PM calculation as a function in Eq.18 which uses Eq.16 and Eq.17 where PM_{i-1} last path PM value, L_i is the candidate LLR value and u_i is the candidate bit value ;

$$PM_i = phi(PM_{i-1}, L_i, u_i) \tag{18}$$

2.2.2.1 Example 4

Example 4: Using Graph Representation Method to Obtain Decoded Block with Using SCL Decoding when L=2

Step 0: We should highlighted that frozen bits sequence are know by decoder. It means that decoder knows that exactly positions of frozen bits before starting decoding process. Also it is important that initial path metric (PM_0) should set up zero. And we also assumed that we will know channel output LLRs before starting the decoding. Our information block is $a = [1011]$ and we will use (M=8, K=4) polar code for decoding because of the its simplicity.

Step 1 We will compute left-hand edge LLRs values using right- hand edge LLR values with the aid of f function which is given in Eg.10, as in the SC decoding. After a bit $\tilde{U}_a$ is available we can compute PM_1 using Eg.19 as following;

$$PM_i = phi(PM_{i-1}, L_i, u_i)$$

$$PM_1 = phi(PM_0, 0.5748, 0)$$

$$\phi_{l,j}(\tilde{u}_{l,j}) = \phi_{l,j-1} + In(1 + e^{-(1-2\hat{u}_{l,j})\tilde{x}_{l,j}})$$

$$PM_1 = PM_0 + In(1 + exp(-(1 - 2 * 0) * 0.5748)) = 0.4465$$

We know that when we reached the frozen bit there is no need to replicate the paths, so as you can see from the Fig.17 we can continue with one path.

Step 2: After a bit $\tilde{U}_a$ is available as $\tilde{U}_a = 0$ we can compute another LLR value for different left- hand edge connection with using g function which is given in Eq.12, as in the SC decoding. Following this we can get new LLR value as 2.4221, and we know that it is positive and we can obtain new encoded bit as 0, because here is the frozen bit positions and we already expected that we will get zero. Moreover

we should continue to calculate PM_2, using PM_1 and new $\tilde{U}_a$ as following;

$$PM_2 = phi(PM_1, 2.4221, 0) = 0.5315$$

And still no need to replicate paths because we reached frozen bits positions.

Step 3: Until reaching an information bit position we should continue to do the same steps of SC decoding, only difference we also calculate PMs values. When we reached the information bit position we should consider two situation, for first situation we will except the value of information bit as 0, while in the second situation we will except the value of information bit as 1. Hence, as it showed in the Fig.17 we should replicate paths and also we should start to consider both values on the graph representation which is given in Fig.16 with (8.1) and (8.2) with blue and orange colors. Now, we have two path and when we reach 2L paths we should prune it to get best path. Most importantly, because of the duplication we should calculate two different PM value which are represented with $PM_{4.1}$ and $PM_{4.2}$ in Fig.18.

$$PM_{4.1} = phi(0.5397, -10.97, 0) = 11.5180$$

$$PM_{4.2} = phi(0.5397, -10.97, 1) = 0.5397$$

Step 4: After partial sum computation of both candidate using g function as in SC decoding ,we will got two different LLR values as 4.34 (12.1) and 13.38 (12.2) which are belong to bit value 0 and 1, respectively, as it can be seen in the Fig.18. Then we will repeat same steps with SC decoding until reaching a new position.

Step 5: Now we have reached the frozen bit positions and we will compute two PM which are computed for different LLRs. Here, our final decision for the value of bit is 0 because as we mentioned before this position belong the frozen bit, but we have different PMs because of the duplicated paths. Fig.18 and 19 represent this process.

Step 6: Using new bit value, we will continue to compute g function for both candidates and when we

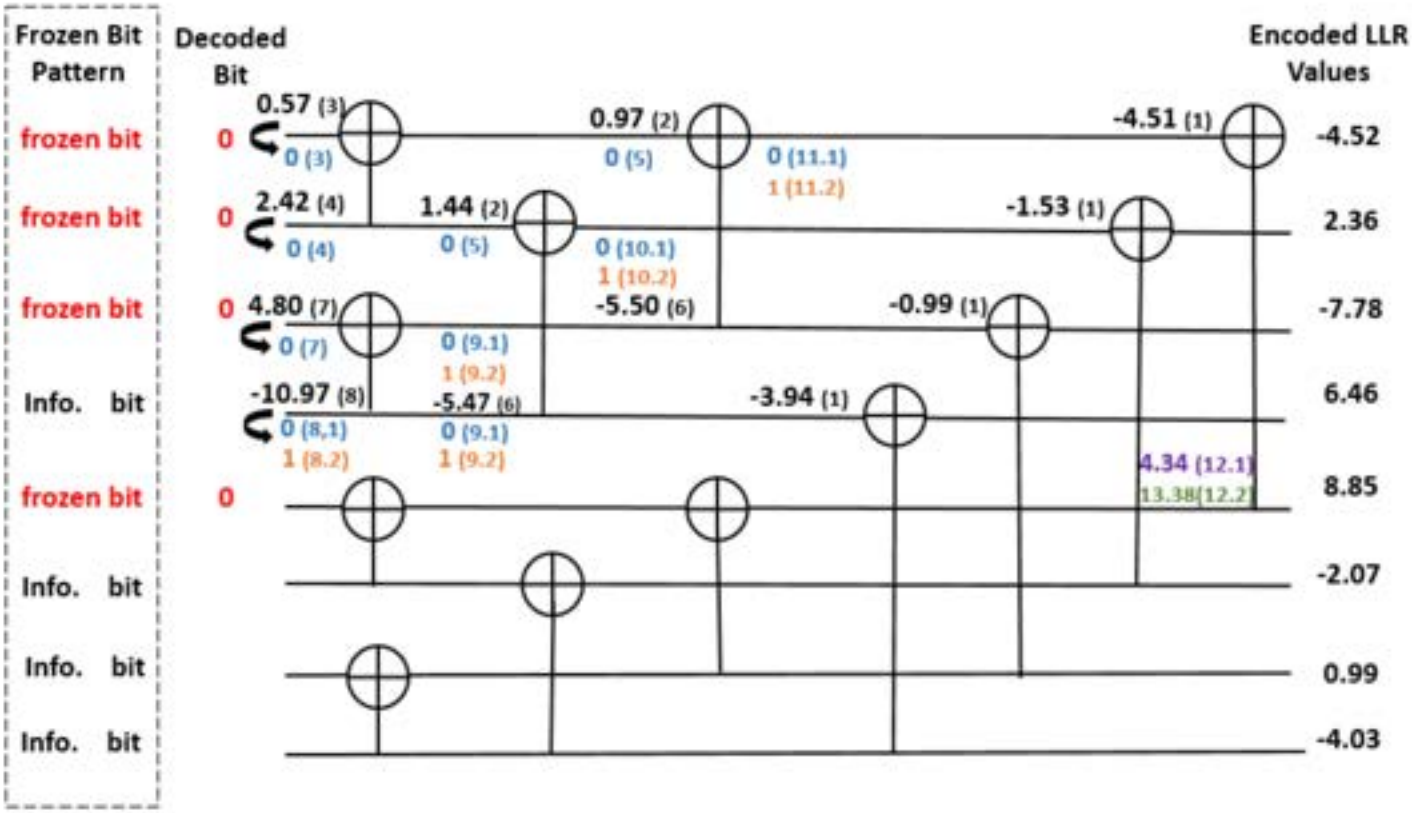

Figure 16: Graph Representation of SCL Decoding for $L = 2$

reached position of an information bit, we should duplicate our paths considering two different bit value for each LLR as shown in the Fig.20 and as it can be seen from the Fig.20 we reached L=4 paths, so we should prune it because L boundary is given as 2. Here, to choose best path we will choose the smallest PMs.

$$PM_{6.1} = phi(12.3446, -4.0095, 0) = 16.3721$$

$$PM_{6.2} = phi(12.3446, -4.0095, 1) = 12.3626$$

$$PM_{6.3} = phi(0.5516, 13.2190, 0) = 0.5516$$

$$PM_{6.4} = phi(0.5516, 13.2190, 1) = 13.7706$$

We should prune $PM_{6.1}$ and $PM_{6.4}$ because they have highest PM values.

Step 7: After that repeating same steps we can reach another information bit positions with new LLRs

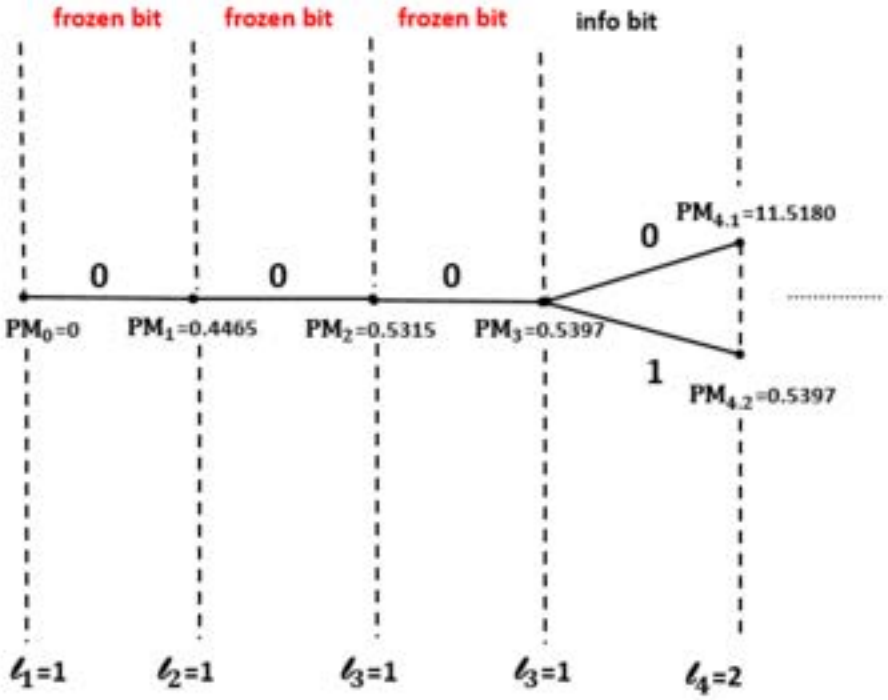

Figure 17: Paths Representation of SCL Decoding for $L = 2$

candidates, hence PMs calculations can done as following;

$$PM_{7.1} = phi(12.3626, -2.1406, 0) = 14.6144$$

$$PM_{7.2} = phi(12.3626, -2.1406, 1) = 12.4738$$

$$PM_{7.3} = phi(0.5516, -14.9393, 0) = 15.4909$$

$$PM_{7.4} = phi(0.5516, -14.9393, 1) = 0.5516$$

We should prune $PM_{7.1}$ and $PM_{7.3}$ because they have highest PMs. This process is represented in Fig.21.

Step 8: After that similarly step 7, with new LLRs candidates, we will compute last PMs values for last position of information bit as following;

$$PM_{8.1} = phi(12.3626, 13.2627, 0) = 12.4738$$

$$PM_{8.2} = phi(12.3626, 13.2627, 1) = 25.7365$$

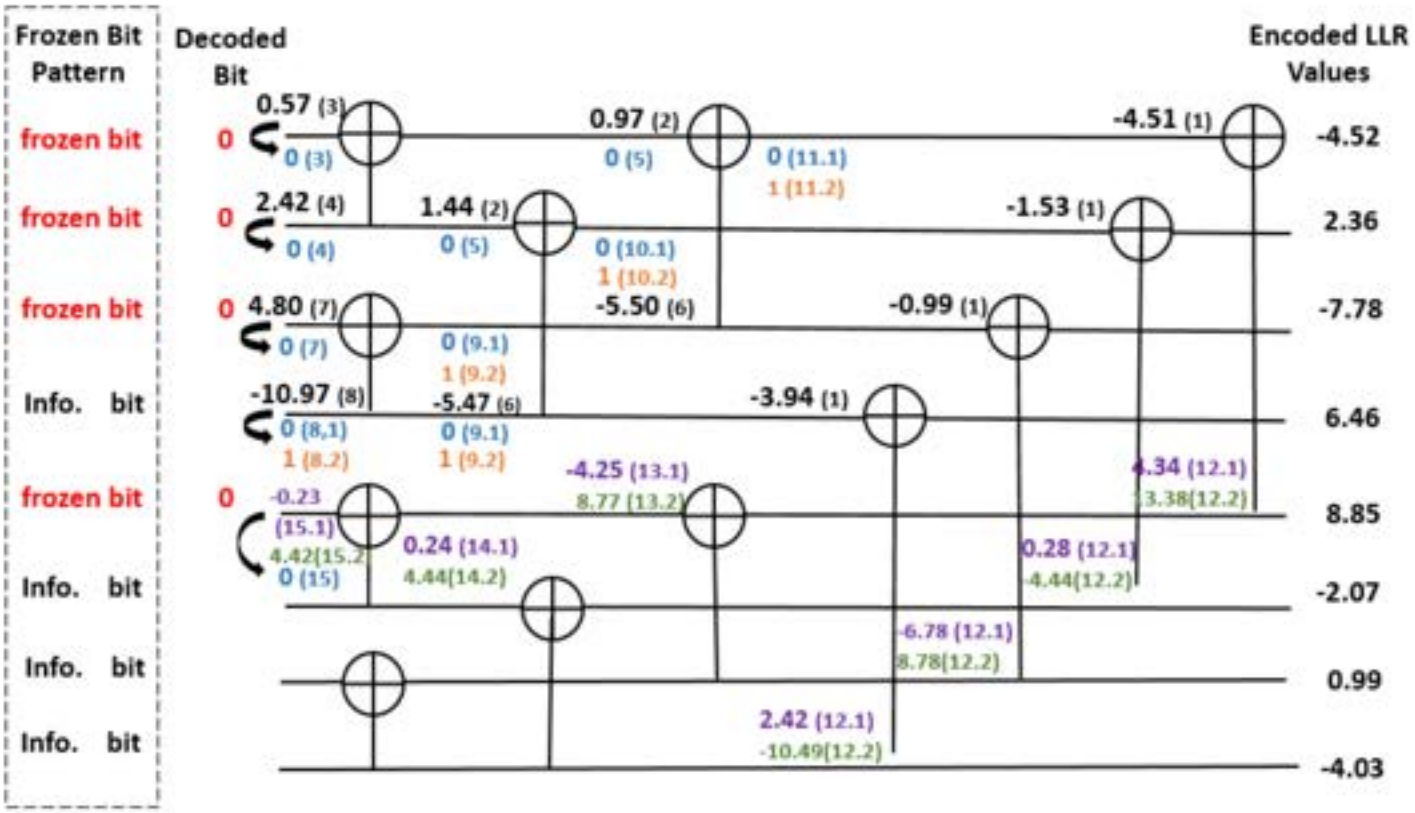

Figure 18: Graph Representation of SCL Decoding for $L = 2$

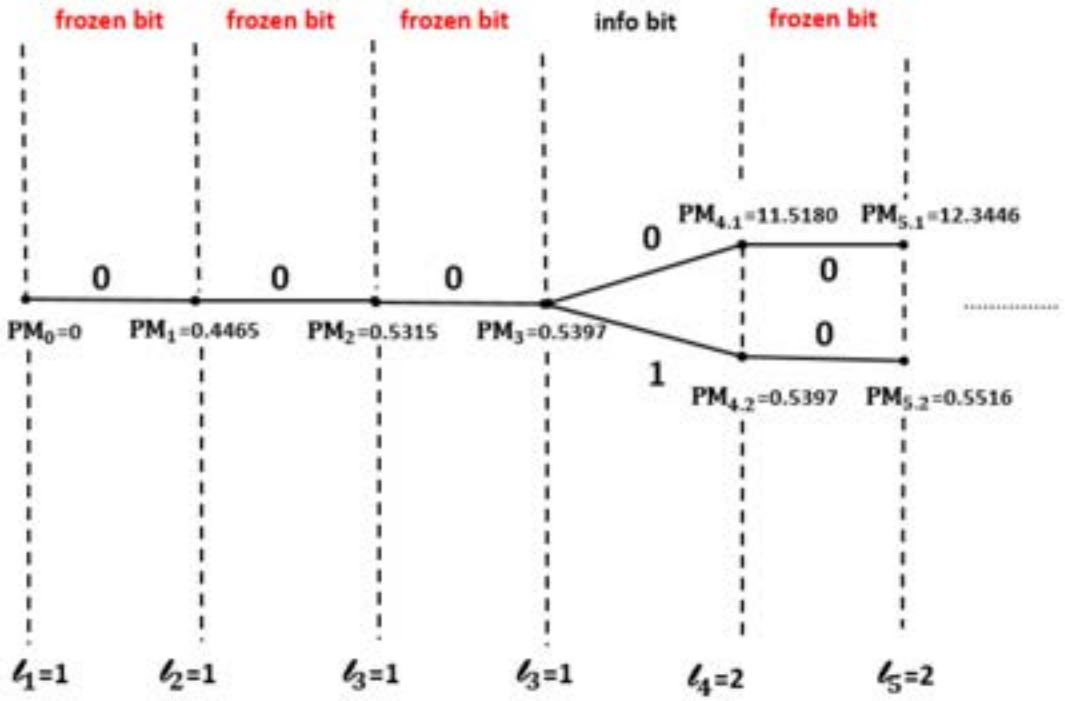

Figure 19: Paths Representation of SCL Decoding for $L = 2$

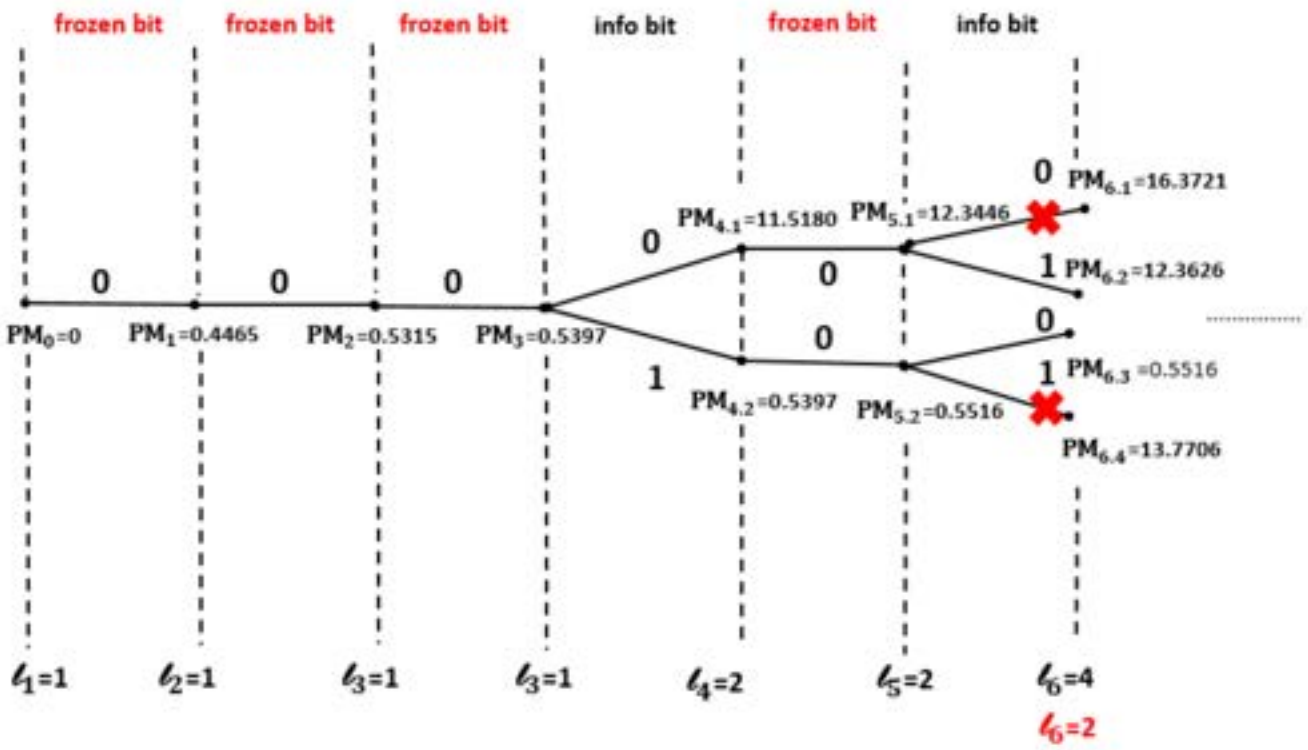

Figure 20: Paths Representation of SCL Decoding for $L = 2$

$$PM_{8,3} = phi(0.5516, -37.1070, 0) = 37.6586$$

$$PM_{8,4} = phi(0.5516, -37.1070, 1) = 0.5516$$

We should prune $PM_{8,2}$ and $PM_{8,3}$ because they have highest PMs. This process is represented in Fig.22.

As final, we have two paths as candidates, to choose the best path we will choose the smallest PM and for last decision, we will follow that path for decoding, as shown in Fig.23. According to this, we have obtained [00010011] and when we remove the frozen bits, we can get decoded word as $\tilde{a} = [1011]$.

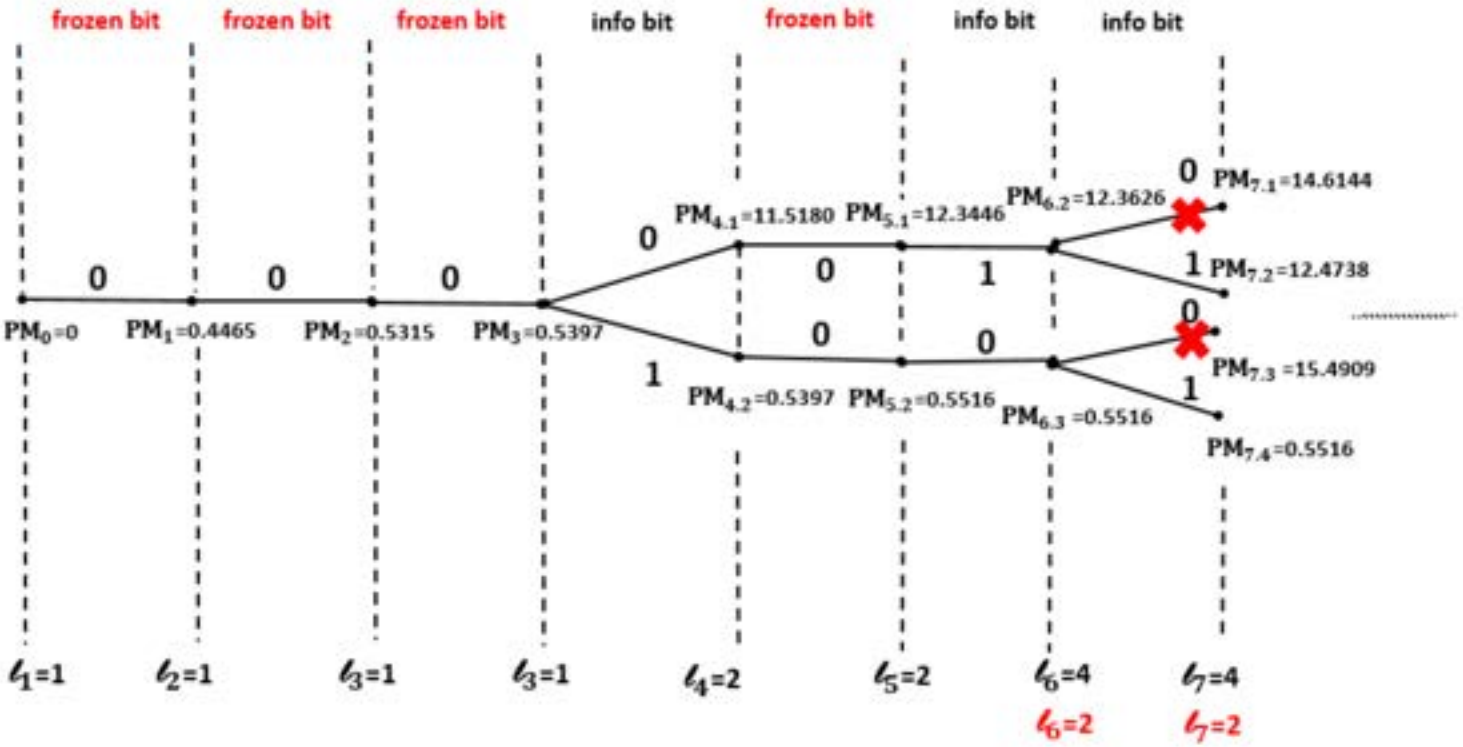

Figure 21: Paths Representation of SCL Decoding for $L = 2$

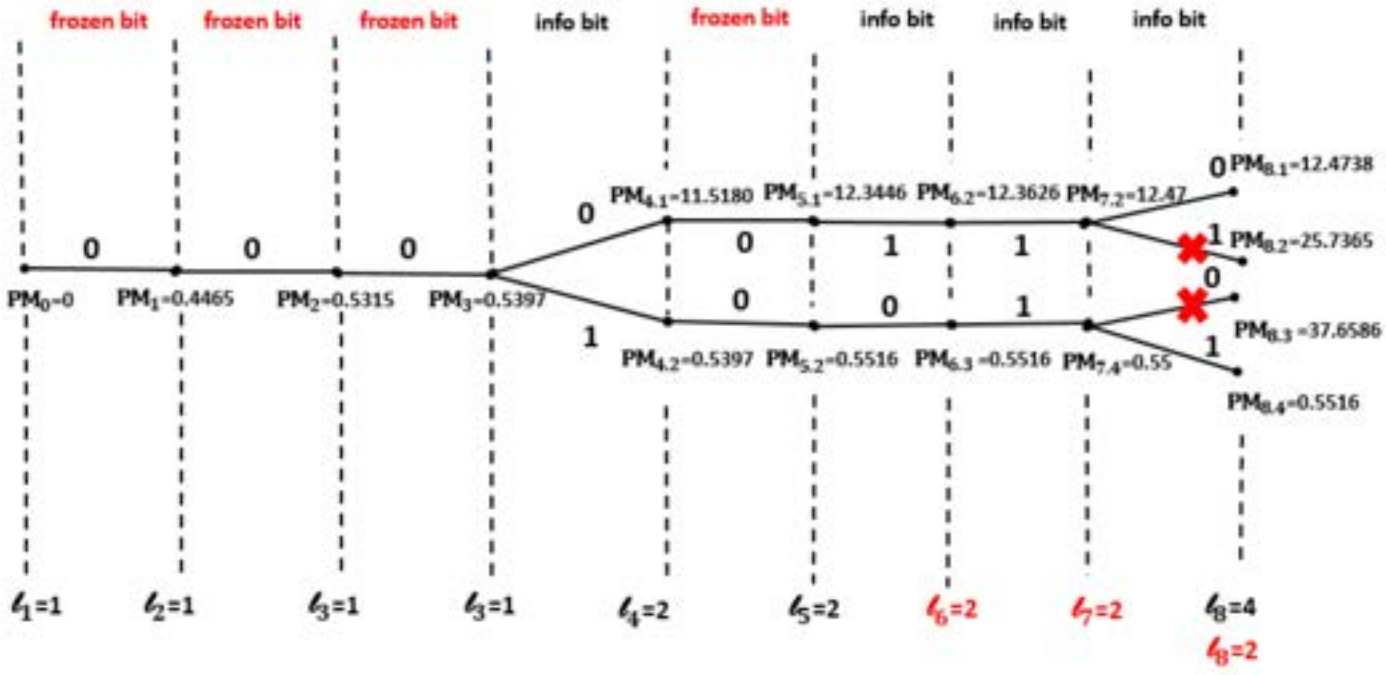

Figure 22: Paths Representation of SCL Decoding for $L = 2$

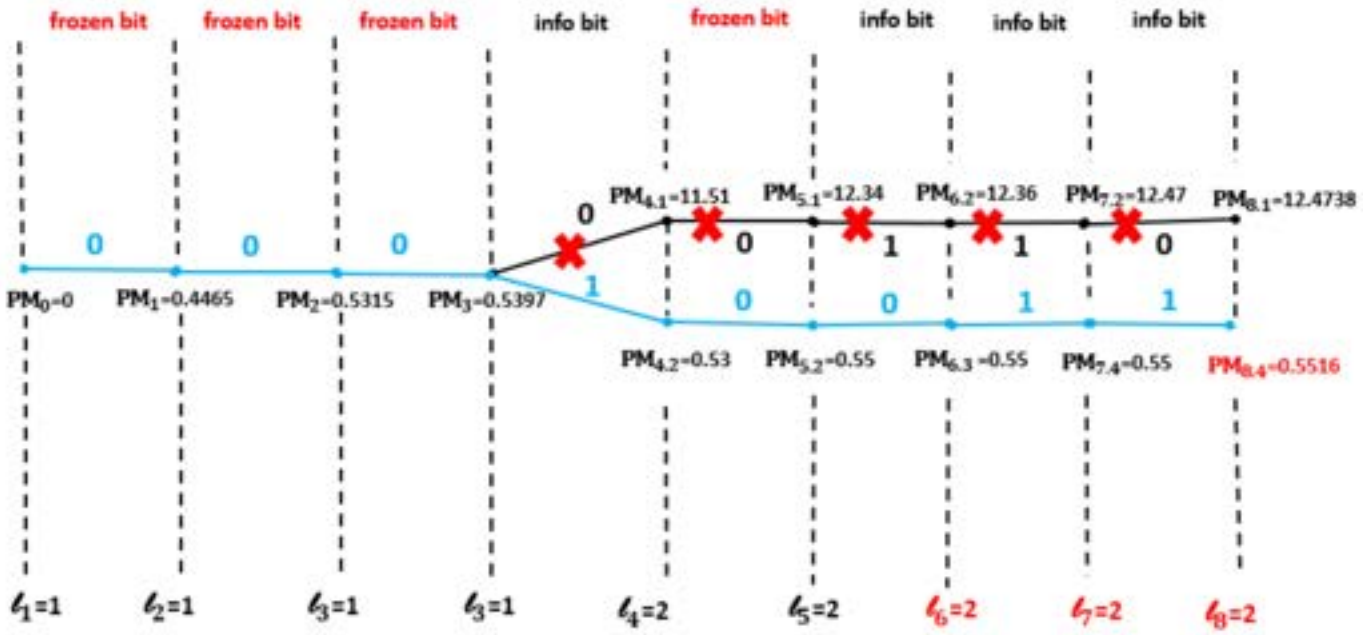

Figure 23: Final Path Representation of SCL Decoding for $L = 2$

2.2.3 Belief Propagation (BP)

2.2.3.1 Original-BP Algorithm

The belief propagation(BP) decoding algorithm for polar codes is based on the factor graph representation of the code [20,23]. For moderate length polar codes, it achieves the error correction performance similar to the successive cancellation algorithm at the cost of high storage and computation requirements, due to its parallel nature [23]. In the BP decoder, each node (i,j) is associated two types of likelihood message which are called R and L messages [18,23]. We should highlighted that, according the graph representation of encoding scheme starting point of L and R matrix can be changed. In this paper, right-to-left messages are represented by $L_{(i,j)}$ and left-to-right messages are represented by $R_{(i,j)}$. $R_{(i,j)}$ are initiated according to Eq.19, where information bits set is A and frozen bits set is A_c [18,23],

and $m = log_2(M)$ represents number of stages in factor graph for (K, M) polar code.

$$R_{(i,1)} = \begin{cases} 0 & i \in A \\ \infty & i \in A_c \end{cases} \tag{19}$$

In order to initialize right-to-left messages, we set up $L_{(i,m_{stages}+1)}$ with channel output LLRs (logarithmic likelihood ratio). And other nodes set up zero for initial. A factor graph includes totally $(m+1)N$ nodes, and each stage consists $N/2$ processing element(PEs) for (K, M) polar code. This processing elements is used for updating the propagation messages iteratively for both $L_{(i,j)}$ and $R_{(i,j)}$ [18,23]. The computations for updating the propagation messages can efficiently be carried out by using a factor graph as proposed in Fig.24 with using Eq.20.

$$\begin{aligned} L_{(i,j)} &= f(L_{(i,j+1)}, (L_{(i+2^{(k-1)},j+1)} + R_{(i+2^{(k-1)},j)})) \\ L_{(i+2^{(k-1)},j)} &= L_{(i+2^{(k-1)},j+1)} + f(L_{(i,j+1)} + R_{(i,j)}) \\ R_{(i,j+1)} &= f(R_{(i,j)}, (L_{(i+2^{(k-1)},j+1)} + R_{(i+2^{(k-1)},j)})) \\ R_{(i+2^{(k-1)},j+1)} &= R_{(i+2^{(k-1)},j)} + f(L_{(i,j+1)} + R_{(i,j)}) \end{aligned} \tag{20}$$

Here $k = m_{stages}$ and $f(a, b)$ is given in Eq.21

$$f(a,b) = 2 * tanh^{-1} * (tanh(\frac{a}{2}) * tanh(\frac{b}{2})) \tag{21}$$

Fig.24 shows an example factor graph for BP for the case of $(K = 4, M = 8)$ polar code is given in Fig.24. Here, factor graph has $m = log_2(8) = 3$ stages with totally $(3+1) * 8 = 32$ nodes, and each stage will consist of $8/2 = 4$ processing elements(PEs), also an example PE is given in Fig.25.

Belief propagation algorithm needs a maximum iteration number ($max_{iteration}$) to obtain stable LLRs. In general, $max_{iteration} = 30$ can be enough for all block lengths. Based on equations in (20), after the decoder reaches maximum iteration number $max_{iteration}$, node (i,1) will output the decoded bits based

on hard decision of messages according to Eq.22.

$$\tilde{u_j} = \begin{cases} 0 & \text{if LLR of } L_{(i,1)}^{max-iteration} > 0 \\ 1 & \text{if } else \end{cases} \tag{22}$$

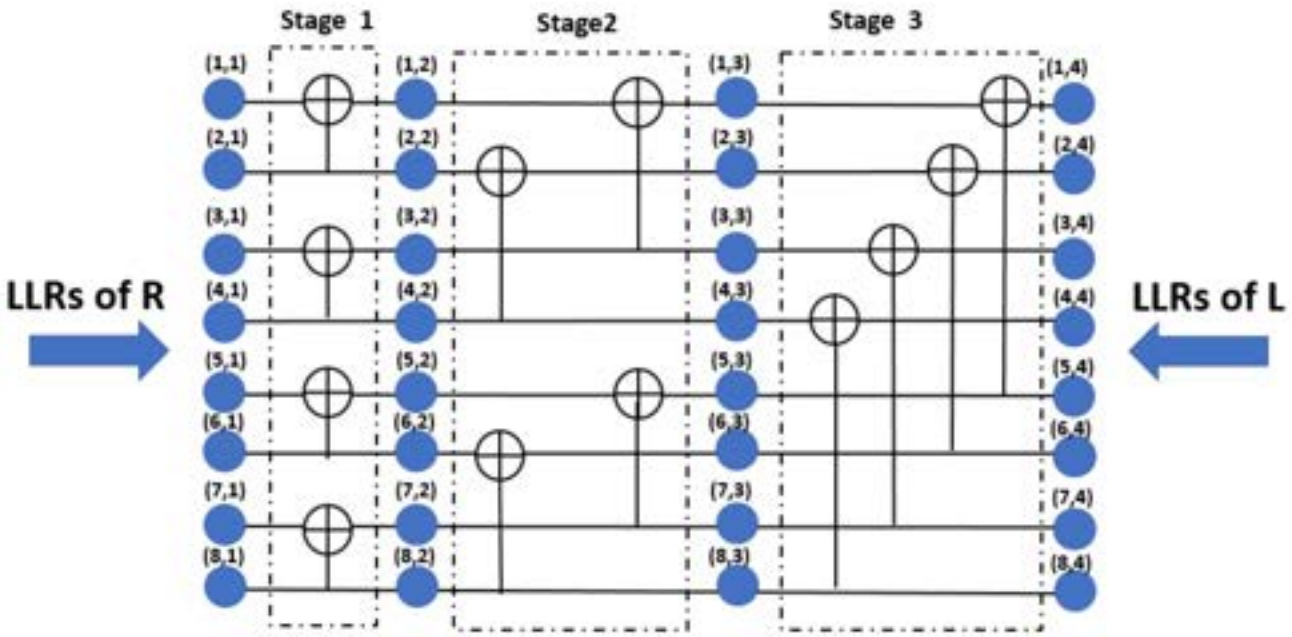

Figure 24: Factor Graph of (8, 4) Polar Code

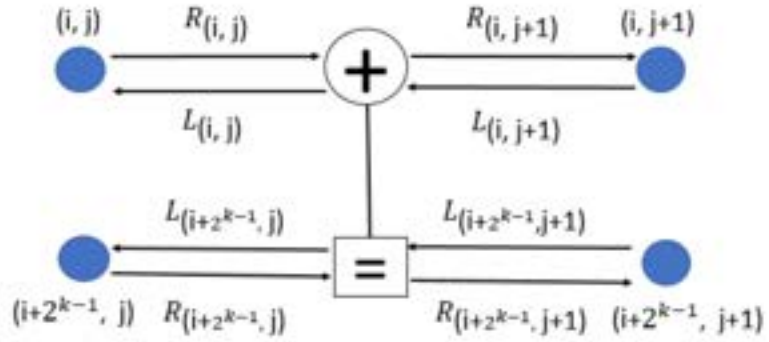

Figure 25: Diagram of Processing Element (PE) for BP

2.2.3.2 Min- Sum(MS)-BP Algorithm

Min-sum approximation method (MS-BP)[18] uses for decreasing complexity of original BP decoding. According to this approximation processing elements are updating based on the Eq.23. However, error-

correcting performance is degraded due to the approximation, as it showed in results (Section 5.3.2).

$$f(a,b) = 2 * tanh^{-1} * (tanh(\frac{a}{2}) * tanh(\frac{b}{2}))$$

can be approximated as

$$f(a,b) = sign(a) * sign(b) * min(|a|,|b|) \tag{23}$$

$sign(.)$ returns -1 if the LLRs value negative, while returns +1 if it is positive, as it is mentioned before. According to Eq.23 PE equations can be written as below

$$\begin{aligned}
L_{(i,j)} &= sign(L_{(i,j+1)}) * sign(L_{(i+2^{(k-1)},j+1)} + R_{(i+2^{(k-1)},j)}) * min(|L_{(i,j+1)}|, |L_{(i+2^{(k-1)},j+1)}|) \\
L_{(i+2^{(k-1)},j)} &= L_{(i+2^{(k-1)},j+1)} + sign(L_{(i,j+1)}) * sign(R_{(i,j)}) * min(|L_{(i,j+1)}|, |R_{(i,j)}|) \\
R_{(i,j+1)} &= sign(R_{(i,j)}) * sign(L_{(i+2^{(k-1)},j+1)}) * min(|R_{(i,j)}|, |(L_{(i+2^{(k-1)},j+1)}) + R_{(i+2^{(k-1)},j)}|) \\
R_{(i+2^{(k-1)},j+1)} &= R_{(i+2^{(k-1)},j)} + sign(L_{(i,j+1)}) * sign(R_{(i,j)}) * min(|L_{(i,j+1)}|, |R_{(i,j)}|)
\end{aligned} \tag{24}$$

2.2.3.3 Scaled- Min- Sum(SMS)-BP Algorithm

MS- BP algorithm has some disadvantages due to the approximation. In order to avoid performance loss in [18] scaled-min -sum (SMS)- BP algorithm is proposed. They introduced a "s" scaling parameter in order to offset the approximation error. The proposed SMS algorithm with $s = 0.9375$ can obtain extra 0.5 dB decoding gain over MS-BP. In that case, the SMS algorithm can achieve a similar error-correcting performance with the original BP and SC algorithm, as it can be seen in the result section.

$$\begin{aligned}
L_{(i,j)} &= s * sign(L_{(i,j+1)}) * sign(L_{(i+2^{(k-1)},j+1)} + R_{(i+2^{(k-1)},j)}) * min(|L_{(i,j+1)}|, |L_{(i+2^{(k-1)},j+1)}|) \\
L_{(i+2^{(k-1)},j)} &= L_{(i+2^{(k-1)},j+1)} + s * sign(L_{(i,j+1)}) * sign(R_{(i,j)}) * min(|L_{(i,j+1)}|, |R_{(i,j)}|) \\
R_{(i,j+1)} &= s * sign(R_{(i,j)}) * sign(L_{(i+2^{(k-1)},j+1)} + +R_{(i+2^{(k-1)},j)}) * min(|R_{(i,j)}|, |L_{(i+2^{(k-1)},j+1)} + +R_{(i+2^{(k-1)},j)}|) \\
R_{(i+2^{(k-1)},j+1)} &= R_{(i+2^{(k-1)},j)} + s * sign(L_{(i,j+1)}) * sign(R_{(i,j)}) * min(|L_{(i,j+1)}|, |R_{(i,j)}|)
\end{aligned} \tag{25}$$

2.2.3.4 Example 5

Example 5: Using MS-BP Decoding Method to Obtain Decoded Block

Step 0: We should highlighted that frozen bits sequence are know by decoder. It means that decoder knows that exactly positions of frozen bits before starting decoding process. And we also assume that we will know channel output LLRs before starting the BP decoding. Decoded information block is $a = [1011]$ and we will use $(K = 4, M = 8)$ polar code for decoding. According to these;

$$\text{Frozen Bits Sequence} = \begin{pmatrix} Frozen & Frozen & Frozen & Info. & Frozen & Info. & Info. & Info. \end{pmatrix}$$

$$\text{Encoded LLRs= Channel Outputs } = \begin{pmatrix} -4.52 & 2.36 & -7.78 & 6.46 & 8.85 & -2.07 & 0.99 & -4.03 \end{pmatrix}$$

Step 1: Factor graph of BP is created for the case of $(K = 4, M = 8)$ polar code. Here, $N = M$ and factor graph has $m = log_2(8) = 3$ stages with totally $(3 + 1) * 8 = 32$ nodes, and each stage will consist of $8/2 = 4$ processing elements(PEs).

Step 2: $R_{(i,1)}$ are initiated according to Eq.19 and $L_{(i,m+1)} = L_{(i,4)}$ are initiated with channel output LLRs . According to these, we obtain following matrices. And other nodes set up zero. For the case of $(K = 4, M = 8)$ polar code $max_{iteration} = 2$ will be enough for correct decoding.

$$R_{(i,j)} = \begin{bmatrix} \infty & 0 & 0 & 0 \\ \infty & 0 & 0 & 0 \\ \infty & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \infty & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$L_{(i,j)} = \begin{bmatrix} 0 & 0 & 0 & -4.52 \\ 0 & 0 & 0 & 2.36 \\ 0 & 0 & 0 & -7.78 \\ 0 & 0 & 0 & 6.46 \\ 0 & 0 & 0 & 8.85 \\ 0 & 0 & 0 & -2.07 \\ 0 & 0 & 0 & 0.99 \\ 0 & 0 & 0 & -4.03 \end{bmatrix}$$

Step 3: Firstly, we will update the right-to-left messages for $L_{(i,j)}$. Based on the first equation in Eq.24 we can start to calculate $L_{(1,3)}$.

$$L_{(i,j)} = sign(L_{(i,j+1)}) * sign(L_{(i+2^{(k-1)},j+1)} + R_{(i+2^{(k-1)},j)}) * min(|L_{(i,j+1)}|, |L_{(i+2^{(k-1)},j+1)}|)$$

$$L_{(1,3)} = sign(L_{(1,4)}) * sign(L_{(5,4)} + R_{(5,3)}) * min(|L_{(1,4)}|, |L_{(5,2)} + R_{(5,3)}|)$$

$$L_{(1,3)} = sign(-4.52) * sign(8.85 + 0) * min(|-4.52|, |8.85 + 0|)$$

$$L_{(1,3)} = -4.52$$

And for $L_{(5,3)}$ we should use second equation in Eq.24;

$$L_{(i+2^{(k-1)},j)} = L_{(i+2^{(k-1)},j+1)} + sign(L_{(i,j+1)}) * sign(R_{(i,j)}) * min(|L_{(i,j+1)}|, |R_{(i,j)}|)$$

$$L_{(5,3)} = L_{(5,4)} + sign(L_{(1,4)}) * sign(R_{(1,3)}) * min(|L_{(1,4)}|, |R_{(1,3)}|)$$

$$L_{(5,3)} = 8.85 + sign(-4.52) * sign(0) * min(|0|, |-4.52|)$$

$$L_{(5,3)} = 8.85 + 0$$

$$L_{(5,3)} = 8.85$$

And new matrix will be as followed;

$$L_{(i,j)} = \begin{bmatrix} 0 & 0 & -4.5200 & -4.5200 \\ 0 & 0 & 0 & 2.3600 \\ 0 & 0 & 0 & -7.7800 \\ 0 & 0 & 0 & 6.4600 \\ 0 & 0 & 8.8500 & 8.8500 \\ 0 & 0 & 0 & -2.0700 \\ 0 & 0 & 0 & 0.9900 \\ 0 & 0 & 0 & -4.0300 \end{bmatrix}$$

Step 4: After that by repeating the similar steps we can complete all graph. We must be careful about $R_{(i,1)}$ values when we come to the stage 1 which showed in Fig.24 because of the initial values of R matrix. Hence, final graph will be as follows and we will call it as first iteration for L matrix;

$$L^1_{(i,j)} = \begin{bmatrix} 0.9900 & 0.9900 & -4.5200 & -4.5200 \\ 3.0600 & 2.0700 & -2.0700 & 2.3600 \\ 0.9900 & -0.9900 & -0.9900 & -7.7800 \\ -5.0200 & -4.0300 & -4.0300 & 6.4600 \\ 0.9900 & 0.9900 & 8.8500 & 8.8500 \\ 3.0600 & 2.0700 & -2.0700 & -2.0700 \\ -0.9900 & 0.9900 & 0.9900 & 0.9900 \\ -4.0300 & -4.0300 & -4.0300 & -4.0300 \end{bmatrix}$$

Step 5: Now we can start to compute $R_{(i,j)}$ matrix, using last two equation in Eg.24.

$$R_{(i,j+1)} = sign(R_{(i,j)}) * sign(L_{(i+2^{(k-1)},j+1)} + +R_{(i+2^{(k-1)},j)}) * min(|R_{(i,j)}|, |L_{(i+2^{(k-1)},j+1)} + +R_{(i+2^{(k-1)},j)}|)$$

$$R_{(1,2)} = sign(R_{(1,1)}) * sign(L_{(2,2)} + +R_{(2,1)}) * min(|R_{(1,1)}|, |L_{2,2)} + +R_{(2,1)}|)$$

$$R_{(1,2)} = sign(Inf) * sign(2.0700 + Inf) * min(|Inf|, |2.0700 + Inf|)$$

$$R_{(1,2)} = Inf = \infty$$

And for $R_{(2,2)}$ we should use the last equation in Eq.20;

$$R_{(i+2^{(k-1)},j+1)} = R_{(i+2^{(k-1)},j)} + sign(L_{(i,j+1)}) * sign(R_{(i,j)}) * min(|L_{(i,j+1)}|, |R_{(i,j)}|)$$

$$R_{(2,2)} = R_{(2,1)} + sign(L_{(1,2)}) * sign(R_{(1,1)}) * min(|L_{(1,2)}|, |R_{(1,1)}|)$$

$$R_{(2,2)} = Inf + sign(0.9900) * sign(0) * min(|0.9900|, |0|)$$

$$R_{(2,2)} = Inf = \infty$$

And new matrix will be as you can see below;

$$R_{(i,j)} = \begin{bmatrix} Inf & Inf & 0 & 0 \\ Inf & Inf & 0 & 0 \\ Inf & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ Inf & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Step 6: After that by repeating the similar steps we can complete all graph for $R_{(i,j)}$. Hence, final graph will be as follows and we will call it as first iteration for R matrix;;

$$R^1_{(i,j)} = \begin{bmatrix} Inf & Inf & -5.0200 & -5.0200 \\ Inf & Inf & -5.0200 & 3.0600 \\ Inf & -4.0300 & -8.5500 & -3.0600 \\ 0 & -0.9900 & -3.0600 & 3.0600 \\ Inf & 2.0700 & 0.9900 & 5.5100 \\ 0 & 0.9900 & -0.9900 & -3.3500 \\ 0 & 0 & 2.0700 & 9.8500 \\ 0 & 0 & -0.9900 & -4.0500 \end{bmatrix}$$

Step 7: Now we can said that one iteration completed. For the second iteration, we will use $R^1_{(i,j)}$ to calculate $L^2_{(i,j)}$ which is the second iteration of L matrix. If we follow similar step with we done before, new calculate $L^2_{(i,j)}$ will be as you can see below;

$$L^2_{(i,j)} = \begin{bmatrix} 4.5200 & 4.5200 & -4.5200 & -4.5200 \\ 6.8800 & 2.3600 & -2.3600 & 2.3600 \\ 7.3800 & -7.5800 & -3.0600 & -7.7800 \\ -14.9600 & -7.3800 & -5.0200 & 6.4600 \\ 4.4300 & 8.7700 & 13.3700 & 8.8500 \\ 13.2000 & 4.4300 & -4.4300 & -2.0700 \\ -8.0800 & 10.8400 & 8.7700 & 0.9900 \\ -8.0800 & -8.0800 & -7.0900 & -4.0300 \end{bmatrix}$$

Step 8: Similarly for $R^2_{(i,j)}$ we follow similar step with we done before, and new matrix will be as

followed and also we reached $max_{iteration} = 2$;

$$R^2_{(i,j)} = \begin{bmatrix} Inf & Inf & -10.4400 & -10.4400 \\ Inf & Inf & -12.6000 & 9.1600 \\ Inf & -7.3800 & -11.9000 & -5.4200 \\ 0 & -7.5800 & -9.9400 & 8.4600 \\ Inf & 4.4300 & 4.4300 & 8.9500 \\ 0 & 8.7700 & -7.0900 & -9.4500 \\ 0 & 0 & 4.4300 & 12.2100 \\ 0 & 0 & -4.4300 & -10.8900 \end{bmatrix}$$

Step 9: After the decoder reaches maximum iteration number $max_{iter} = 2$, node (i, 1) will output the decoded bits based on hard decision of messages according to Eq.22. Hence decoded bits will be ;

$$\tilde{u}_j = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{pmatrix}$$

if we remove the frozen bits ,we can get the decoded bit sequence which is equal the information bit sequence;

$$\tilde{a} = \begin{pmatrix} 1 & 0 & 1 & 1 \end{pmatrix}$$

Also we want to highlighted that although second and first iteration can give the same decoded word, LLRs values of $L_{(i,1)}$ in the second iteration has higher magnitude then first one as you can see below,

and this increases the reliability of decoding.

$$L_{(i,4)} = \begin{array}{c} \begin{matrix} \text{First Iteration} & \text{Second Iteration} & \text{Hard Decision} \end{matrix} \\ \begin{bmatrix} 0.9900 & 4.5200 & 0 \\ 3.0600 & 6.8800 & 0 \\ 0.9900 & 7.3800 & 0 \\ -5.0200 & -14.9600 & 1 \\ 0.9900 & 4.4300 & 0 \\ 3.0600 & 13.2000 & 0 \\ -0.9900 & -8.0800 & 1 \\ -4.0300 & -8.0800 & 1 \end{bmatrix} \end{array}$$

2.2.3.5 Soft Cancellation (SCAN) Algorithm

The soft-cancellation (SCAN) decoder is introduced by Fayyaz and Barry in [16], which achieves performance gain with lower complexity compared to original BP and converges much faster than the original BP algorithms [24]. The SCAN decoding can be seen as a mixture of the SC and Original BP decoding [25],because the operation scheduling is similar to the SC one while propagation messages are updating iteratively in both ways as soft information for both $L_{(i,j)}$ and $R_{(i,j)}$[25]. Initializing of $L_{(i,j)}$ and $R_{(i,j)}$ are same as given in Original BP, and for updating propagation messages Eq.20 can be used which is given in original BP section. Moreover, same factor graph in Fig.24 can be used. In [25], it is claimed that only few iteration with the SCAN decoding algorithm can give better performance than 50 iteration of the Original BP algorithm.

2.2.3.6 Example 6

Example 6: Using SCAN Decoding Method to Obtain Decoded Block Step 0: We should highlighted that frozen bits sequence are know by decoder. It means that decoder knows that exactly positions of frozen bits before starting decoding process. And we also assume that we will know channel output LLRs before starting the SCAN decoding. Decoded information block is $a = [1011]$ and we will use $(K = 4, M = 8)$ polar code for decoding. Moreover, same factor graph in Fig.24 will be used. According to these;

$$\text{Frozen Bits Sequence} = \begin{pmatrix} Frozen & Frozen & Frozen & Info. & Frozen & Info. & Info. & Info. \end{pmatrix}$$

$$\text{Encoded LLRs= Channel Outputs } = \begin{pmatrix} -4.52 & 2.36 & -7.78 & 6.46 & 8.85 & -2.07 & 0.99 & -4.03 \end{pmatrix}$$

Step 1: $R_{(i,1)}$ are initiated according to Eq.19 and $L_{(i,m+1)} = L_{(i,4)}$ are initiated with channel output LLRs . According to these, we obtain following matrices. And other nodes set up zero. For the case of $(K = 4, M = 8)$ polar code $max_{iteration} = 1$ will be enough for correct decoding for SCAN decoding.

$$R_{(i,j)} = \begin{bmatrix} \infty & 0 & 0 & 0 \\ \infty & 0 & 0 & 0 \\ \infty & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \infty & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$L_{(i,j)} = \begin{bmatrix} 0 & 0 & 0 & -4.52 \\ 0 & 0 & 0 & 2.36 \\ 0 & 0 & 0 & -7.78 \\ 0 & 0 & 0 & 6.46 \\ 0 & 0 & 0 & 8.85 \\ 0 & 0 & 0 & -2.07 \\ 0 & 0 & 0 & 0.99 \\ 0 & 0 & 0 & -4.03 \end{bmatrix}$$

Step 2: To update propagation messages, we will follow schedule which is presented in table Table 1.

Every column should be followed from top to bottom and then should be pass other column.

Table 1: Scheduling for SCAN to Update LLRs Values in One Iteration for $(K = 4, M = 8)$ and *STG=stage*

1	2	3	4	5	6	7	8	9	10	11	12	13
STG3	STG2	STG1	STG2	STG1	STG2	STG3	STG2	STG1	STG2	STG1	STG2	STG1
$L_{(1,3)}$	$L_{(1,2)}$	$R_{(1,2)}$	$L_{(3,2)}$	$R_{(3,2)}$	$R_{(1,3)}$	$L_{(5,3)}$	$L_{(5,2)}$	$R_{(5,2)}$	$L_{(7,2)}$	$R_{(7,2)}$	$R_{(5,3)}$	$L_{(1,1)}$
$L_{(2,3)}$	$L_{(2,2)}$	$R_{(2,2)}$	$L_{(4,2)}$	$R_{(4,2)}$	$R_{(2,3)}$	$L_{(6,3)}$	$L_{(6,2)}$	$R_{(6,2)}$	$L_{(8,2)}$	$R_{(8,2)}$	$R_{(6,3)}$	$L_{(2,1)}$
$L_{(3,3)}$					$R_{(3,3)}$	$L_{(7,3)}$					$R_{(7,3)}$	$L_{(3,1)}$
$L_{(4,3)}$					$R_{(4,3)}$	$L_{(8,3)}$					$R_{(8,3)}$	$L_{(4,1)}$
												$L_{(5,1)}$
												$L_{(6,1)}$
												$L_{(7,1)}$
												$L_{(8,1)}$

Step 3: According to Table 1 if we follow columns 1 and 2 for updating the right-to-left messages for $L_{(i,j)}$ based on the first equation in Eq.24, we obtained followed matrix.

$$L_{(i,j)} = \begin{bmatrix} 0 & 0.9900 & -4.5200 & -4.5200 \\ 0 & 2.0700 & -2.0700 & 2.3600 \\ 0 & 0 & -0.9900 & -7.7800 \\ 0 & 0 & -4.0300 & 6.4600 \\ 0 & 0 & 0 & 8.8500 \\ 0 & 0 & 0 & -2.0700 \\ 0 & 0 & 0 & 0.9900 \\ 0 & 0 & 0 & -4.0300 \end{bmatrix}$$

Step 4: According to Table 1 if we follow column 3 for updating the left-to-right messages for $R_{(i,j)}$ based on the third equation in Eq.24, we obtained followed matrix.

$$R_{(i,j)} = \begin{bmatrix} Inf & Inf & 0 & 0 \\ Inf & Inf & 0 & 0 \\ Inf & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ Inf & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Step 5: According to Table 1 if we follow column 4 for updating the right-to-left messages for $L_{(i,j)}$ based on the second equation in Eq.24, we obtained followed matrix.

$$L_{(i,j)} = \begin{bmatrix} 0 & 0.9900 & -4.5200 & -4.5200 \\ 0 & 2.0700 & -2.0700 & 2.3600 \\ 0 & -5.5100 & -0.9900 & -7.7800 \\ 0 & -6.1000 & -4.0300 & 6.4600 \\ 0 & 0 & 0 & 8.8500 \\ 0 & 0 & 0 & -2.0700 \\ 0 & 0 & 0 & 0.9900 \\ 0 & 0 & 0 & -4.0300 \end{bmatrix}$$

Step 6: According to Table 1 if we follow columns 5 and 6 for updating the left-to-right messages for $R_{(i,j)}$ based on equation in Eq.24, we obtained followed matrix.

$$R_{(i,j)} = \begin{bmatrix} Inf & Inf & -7.0900 & 0 \\ Inf & Inf & -9.5400 & 0 \\ Inf & -6.1000 & -10.6200 & 0 \\ 0 & -5.5100 & -7.5800 & 0 \\ Inf & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Step 7: Now, we finished the first half of the both $L_{(i,j)}$ and $R_{(i,j)}$ matrix. To calculate second half, according to Table 1 we will do step 7 and 8 for updating the right-to-left messages for $L_{(i,j)}$ based on

the second equation in Eq.24, we obtained followed matrix.

$$L_{(i,j)} = \begin{bmatrix} 0 & 0.9900 & -4.5200 & -4.5200 \\ 0 & 2.0700 & -2.0700 & 2.3600 \\ 0 & -5.5100 & -0.9900 & -7.7800 \\ 0 & -6.1000 & -4.0300 & 6.4600 \\ 0 & 8.7700 & 13.3700 & 8.8500 \\ 0 & 4.4300 & -4.4300 & -2.0700 \\ 0 & 0 & 8.7700 & 0.9900 \\ 0 & 0 & -10.4900 & -4.0300 \end{bmatrix}$$

Step 8: After that by repeating the steps which are given in Table 1 we can complete all graphs.

$$L^1_{(i,j)} = \begin{bmatrix} 0.9900 & 0.9900 & -4.5200 & -4.5200 \\ 3.0600 & 2.0700 & -2.0700 & 2.3600 \\ 5.5100 & -5.5100 & -0.9900 & -7.7800 \\ -11.6100 & -6.1000 & -4.0300 & 6.4600 \\ 4.4300 & 8.7700 & 13.3700 & 8.8500 \\ 13.2000 & 4.4300 & -4.4300 & -2.0700 \\ -13.2000 & 13.2000 & 8.7700 & 0.9900 \\ -14.9200 & -14.9200 & -10.4900 & -4.0300 \end{bmatrix}$$

$$R^1_{(i,j)} = \begin{bmatrix} Inf & Inf & -7.0900 & 0 \\ Inf & Inf & -9.5400 & 0 \\ Inf & -6.1000 & -10.6200 & 0 \\ 0 & -5.5100 & -7.5800 & 0 \\ Inf & 4.4300 & 4.4300 & 0 \\ 0 & 8.7700 & -8.7700 & 0 \\ 0 & 0 & 4.4300 & 0 \\ 0 & 0 & -4.4300 & 0 \end{bmatrix}$$

Step 9: After the decoder reaches maximum iteration number $max_{iter} = 1$, node (i, 1) will output the decoded bits based on hard decision of messages according to Eq.22. Hence decoded bits will be ;

$$\tilde{u_j} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{pmatrix}$$

if we remove the frozen bits, we can get the decoded bit sequence which is equal the information bit sequence;

$$\tilde{a} = \begin{pmatrix} 1 & 0 & 1 & 1 \end{pmatrix}$$

Also we want to highlighted that although SCAN has only first iteration, magnitudes of LLRs values

considerably higher than the first iteration of BP decoding, as it shown in the below

$$L_{(i,4)} = \begin{array}{c} \begin{array}{cccc} \text{First Iteration BP} & \text{Second Iteration BP} & \text{First Iteration SCAN} & \text{Hard Decision} \end{array} \\ \left[\begin{array}{cccc} 0.9900 & 4.5200 & 0.9900 & 0 \\ 3.0600 & 6.8800 & 3.0600 & 0 \\ 0.9900 & 7.3800 & 5.5100 & 0 \\ -5.0200 & -14.9600 & -11.6100 & 1 \\ 0.9900 & 4.4300 & 4.4300 & 0 \\ 3.0600 & 13.2000 & 13.2000 & 0 \\ -0.9900 & -8.0800 & -13.2000 & 1 \\ -4.0300 & -8.0800 & -14.9200 & 1 \end{array}\right] \end{array}$$

3 Planning

Time is used effectively and efficiently thanks to planning, so that the project can be completed successfully and on time. Hence, in this section planning details of this project will be presented. We divided planning steps into five main part. Section 3.1 introduces top-level project objectives. Section 3.2 presents project tasks. Success criteria and available resources are explained in Section 3.3. The problems that may be encountered and possible solutions are given in section 3.4 and finally Gantt Chart to overview of workflow of the project is presented in Section 3.5.

3.1 Top- level Project Objectives

This project aims to achieve the following top- level objectives. First of all, this thesis is intended to provide detailed information on the subject of Polar Codes (PC) encoding and decoding process for anyone interested in learning about it. Especially for PC decoding, this project aims that providing information in the form of a background knowledge for each common decoding technique such as SC, SCL and BP, with examples and simulation results, also considering different combinations such as different coding rates and list sizes. Another and most important top- level objective of this project is that implement BP decoder with different schedules to find optimized BP schedule and comparing this schedules with provided BLER curves of SC and SCL decoding methods.

3.2 Project Tasks

The above-mentioned top- level objectives can be divided into the following sub-tasks. First of all, search background information for Polar Codes (PC) to understand encoder and decoder process. Secondly, do an example using hand calculation for Polar Encoding. After that, use graph representation method

for Polar Encoding example. After creating the detailed background knowledge for encoding, give background information for common three decoding methods which are SC, SCL and BP. After each decoding method explanation, do an example using hand calculation to make them more understandable. After creating a basic idea about the operation of common decoding methods, give detailed explanation of each proposed belief propagation (BP) decoding schedules for Polar Codes. Then, simulate to get BLER curve of SC with different combinations such as using different coding rate and block lengths. Similarly, simulate to get BLER curve of SCL with different combinations such as using different coding rate, block lengths and list sizes. Simulate proposed schedules to find best BP schedule. After obtain all results for decoding methods compare them to show pros and cons of proposed BP decoding schedules. And finally, complete 15.000-word report.

3.3 Success Criteria and Available Resources

In this sub-section, success criteria which are should be met for this project to be successful, will be present, as well as resources which are provided by the university of Southampton to met needs of the project.

In order for this project to be successful the following criteria should be met; We should obtain same results using both hand calculation and MATLAB software for Polar Encoding example. Similarly, we should obtain same results using both hand calculation and MATLAB software for each decoding method example which are belong to SC, SCL and BP. For proposed BP schedules we should achieve BP decoding outputs as LLR values. Moreover, we should manage to create self consisted LLRs as output of BP. And as final, we should create 15,000-word report before final deadline which is 7th of September in 2017.

The following resources have been provided by the University of Southampton for the success of the

project; R2017a MATLAB software program which will be used to obtain BLER curves. Also the University of Southampton Supercomputing facility which is called "Lyceum" can be used to obtain simulation results in order to prevent time wasting because of the complexity of algorithms. Above all, WebChat and DelphiS which are interfaces to allows you to discover electronic and print items in the library's collection are the resources which are provided by the University of Southampton. Finally, template codes of SC and SCL encoding and decoding are provided by Prof. Rob Maunder.

3.4 Risks and Solutions

The following risk were outlined together with their impact, probability and solution. Firstly, we can said that giving an example using hand calculation to show all steps of encoding and decoding for Polar Codes may constitute an obstacle for our project because sometimes calculations can be complex and difficult to follow decoding order for humans, especially for long block length. As a solution of this, we will choose very basic example with small block lengths, also using approximations for some equations. Another risk is that failing to produce BP schedules (some of them or all of them) will create significantly negative effect on the project and we can said that it has very high probability, to reduce effect or to remove completely we should research past dissertation papers on the same topic and actively work with supervisor in order to debug all arising problems. Also complexity can be very high to run the simulations in individual computers for decoding algorithms, this risk have high probability and high negative effect on project, but we can reduce of this effect using the University of Southampton Supercomputing facility which is called "Lyceum". Another high probability risk is that failing to obtain self consisted LLRs values as BP outputs, to solve this problem we should check our algorithms using small block length as an example also we should decide enough max iteration number for each decoding schedule. Above all, being ill can have a negative impact on the project, however, taking care of yourself and following the plan will reduce this effect. Another risk is that computer can be damaged although this situation

has low probability, but it has very negative impact on the project, as a solution, we recommend that copies of studies, should always store also in safe area. Moreover, failing to meet deadlines is another risk which also have significantly negative effect on project, to combat this we should carefully manage time with the aid of Gantt charts.

3.5 Gantt Chart

The final deadline for this master thesis is the 7th of September, which means that to complete all project successfully we only need almost 13 weeks. We have used a Gantt Chart to overview of how the workflow of the project was carried out. In order to use this limited time effectively and efficiently Gantt Chart should be design carefully, the Gantt Chart is given in Fig.26.

According to Fig.26, to search resources for background information of PC, 1 week is appointed. After this process, we have appointed 1 week for encoding and 1 week for decoding process in order to create the detailed background knowledge for project. For SC decoding method and SCL decoding method 2 and 2.5 weeks are appointed, respectively. Because SC decoding is the basis for all decoding techniques, we will first study this method and we can not pass it on to other techniques without understanding it completely. The reason why we are going to share 2.5 weeks for SCL is that the SCL technique is more complicated than the SC technique. After finishing these two basic decoding techniques, we were able to separate 2 weeks for the BP technique, which is very important for us to complete the project. The reason of that, BP decoding which will be called as Original BP, will be the basis for our proposed methods. Hence, we will spent time on it as much as SC decoding after finished SCL decoding method.

We also separated the last days of the duration for SC decoding to get BLER simulation results of SC at the same time, which will take 1 week. We will use these results as a reference for other decoding techniques, because we first do SC simulations. Similarly, we also separated the last days of the duration

for SCL decoding to get BLER simulation results of SCL at the same time, which will take 1.5 weeks. The reason why we are going to share 1.5 weeks for simulations of SCL is that as we mentioned before, the SCL technique is more complicated than the SC technique and according to the list size, to obtain results can take much more time. As already mentioned, template codes of SC and SCL encoding and decoding are provided by Prof. Rob Maunder. Hence, there is no need to appoint a time for writing code for them, although there is a need for appointing a time for original BP algorithm which will take 1 week. Also, we are going to share 1.5 weeks for simulations of original BP because we should be sure that original BP results are correct to make comparison with proposed schedules.

After completed understanding BP decoding method process, we should start to write proposed BP schedule algorithms. Hence, we will start to work on the algorithms of proposed methods from the middle of in August, and we plan to spend about 3 weeks for it. During the almost same time, we will also obtain BLER results for proposed algorithms. We should highlighted that last month is crucial to complete project successfully. Any problem that may arise during this month, may have a very negative impact on the completed successfully of the project. We can not specify it in the Gantt chart because of the time limit, if we want to define additional time to solve the problems that may arise. It is therefore recommended that the Gantt chart be followed closely. In addition to all these, we will write our thesis as report throughout the whole process, which is about 11 weeks apart.

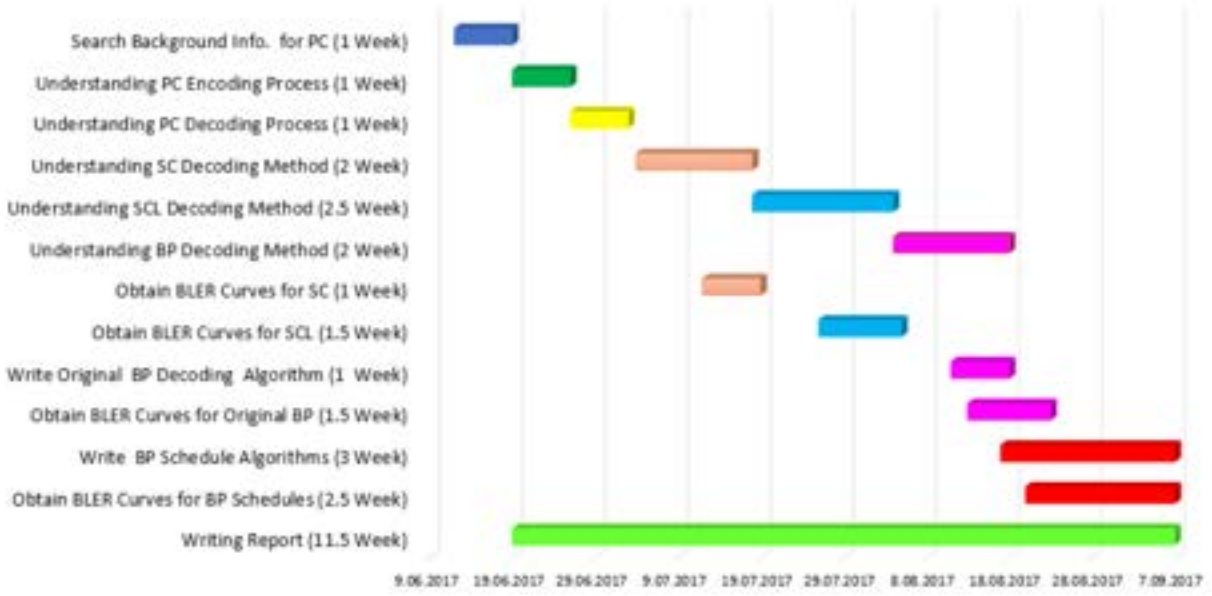

Figure 26: A Gantt Chart to Show Workflow of the Project

4 Implementation

In this section, implementation details of this project will be presented. In the first subsection, we will implement original BP decoding method to use as reference BLER curve for proposed schedules. In next subsections, we will present proposed schedules 1, 2 and 3, respectively. All algorithms are generated with using MATLAB software, and also are used QPSK (Quadrature Phase Shift Keying) modulation. The reasons of choosing QPSK modulation are that it can be easily applied and also in practice it is one of the common modulation techniques. Thanks to the MATLAB program, the desired modulation technique can be easily applied only by modifying the modulation and demodulation part. For this project, all simulation results obtained using University of Southampton supercomputing facility "Lyceum".

4.1 Original BP Decoding Method

As it mentioned before in Section 2.2.3., to implement Original BP decoding method we will follow equations which are given in Eq.20 as well as similar factor graph in Fig.24 with same PE in Fig.25.

4.1.1 Generate Random Input bits

The first step to establishing the polar encoder is to generate random bits which are 0 or 1. The random bits are generated by using some functions available in MATLAB, which continuously generate random bits in the form of an array, and array size depends only K information bits length. We will generate bits as frame. The reason of generating bits as frame is that it makes easy and fast to compute BLER. This generated bits will be input of the polar encoder.

Listing 1: MATLAB code for Generating Random Input bits

```
% Generate a random frame of bits
    a = round(rand(1,K));
```

4.1.2 Apply Polar Encoder and Generate LLRs

As it mentioned before in Section 2.1.1., Polar Encoder takes information bits with length K and turn into encoded block with length M by adding frozen bits. The steps to be followed at this stage are quite similar to the Example 1. The inputs of the polar encoder are that generated bits a and encoded block length M.

Firstly, generate $(N-M)-$entry puncture position set as a row vector which is called P. Then, generate $(M-K)-$entry frozen set as a row vector F^{M-K-1}. To obtain frozen set, we will select the first $(M-K)$ entries from the Q whose values are smaller than N and do not exist in P^{N-M-1}. After that, generate

$(N-K)-$entry frozen positions as a row vector U^{N-K-1}. Initialize the input binary vector by the setting the zero onto the positions defined by U^{N-K-1} (frozen bits) and putting K information block onto the remaining entries. To obtain encoded block $bk = 0^{M-1}$, we use Eq.5 which is given in Example 1. And finally with shortening, we should generate $b = b_{k=0}^{M-1}$ when $k \notin P^{N-M-1}$ to obtain encoded block. We should highlighted that this encoding steps are also same for other decoding techniques.

After completed polar encoder, QPSK modulation will take encoded block as input. Moreover, to simulate a real transmission we will add channel noise onto QPSK signal, then we will do QPSK demodulation. We should highlighted that we will generate LLRs values of the channel output from the demodulated signal. These LLRs values will be input of the polar decoder.

4.1.3 Apply Polar Decoder

We will take LLRs values of the channel output and information bits length K as input of the polar decoder. As it mentioned before, decoder always knows exact positions of the frozen bits before decoding process. Hence, firstly we should start with same steps in encoder to teach frozen bit sequence to the decoder.

After this, we should decide the $max_{iteration}$ number to get stable LLRs. After some experiment, we have decided $max_{iteration} = 30$ will be enough for all block lengths. Although small code lengths reaches target BLER with small number of iterations, we need to decide on a fixed number of max iteration to make a fair comparison between BLER performances.

Then we should initialize left to right messages $L_{(i,j)}$ and right to left messages $R_{(i,j)}$ as it showed in Example 5. According to Fig.24, we will compute processing element (PE) which is given in Fig.25 for each stages. In that point, similarly Fig.25, Fig.27 represents functions names and aims which are called to compute nodes. The stages will start for $L_{(i,j)}$ from $m_{stages} = log2(M) = k$ and it will end in stage

1, and vice versa for $R_{(i,j)}$.

$$\begin{aligned} CN\text{-}a\text{-}out &= L_{(i,j)} & VN\text{-}a\text{-}out &= L_{(i+2^{k-1},j)} \\ CN\text{-}c\text{-}out &= R_{(i,j+1)} & VN\text{-}v\text{-}out &= R_{(i+2^{k-1},j+1)} \end{aligned} \tag{26}$$

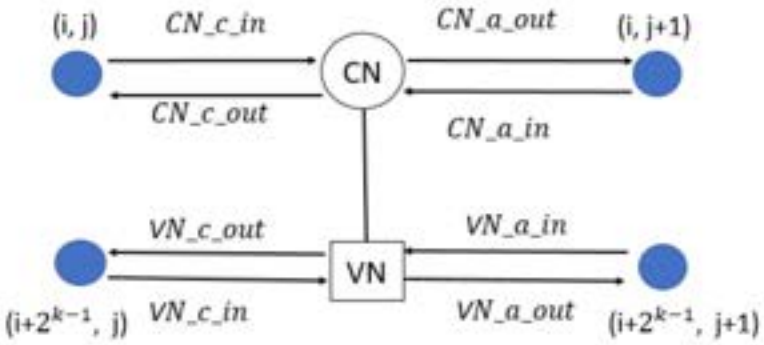

Figure 27: Functions of Processing Element for BP

After reach max iteration number, according to Eg.22 we will decide value of each bit. We called this decision step as hard decision. Finally, we can remove frozen bit sequence from the decoded block and we can obtain decoded information block.

We must emphasize that, as a result of the first decoding operation, our BLER performance was very poor, we notice that the reason of this, after the recommendation of our supervisor, the factor graphs for encoding and decoding were not the same. If we clarify more clearly, you should follow the encoding graph in reverse for decoding.

To make it more understandable, followed schedule for original BP when (K=4,M=8) is given in Table 2 as an example. Every column should be followed from top to bottom and then should be pass other column.

Table 2: Scheduling for Original BP to Update LLRs Values in One Iteration for $(K = 4, M = 8)$

1	2	3	4	5	6
Stage3	Stage2	Stage1	Stage1	Stage2	Stage3
$L_{(1,3)}$	$L_{(1,2)}$	$L_{(1,1)}$	$R_{(1,2)}$	$R_{(1,3)}$	$R_{(1,4)}$
$L_{(2,3)}$	$L_{(2,2)}$	$L_{(2,1)}$	$R_{(2,2)}$	$R_{(2,3)}$	$R_{(2,4)}$
$L_{(3,3)}$	$L_{(3,2)}$	$L_{(3,1)}$	$R_{(3,2)}$	$R_{(3,3)}$	$R_{(3,4)}$
$L_{(4,3)}$	$L_{(4,2)}$	$L_{(4,1)}$	$R_{(4,2)}$	$R_{(4,3)}$	$R_{(4,4)}$
$L_{(5,3)}$	$L_{(5,2)}$	$L_{(5,1)}$	$R_{(5,2)}$	$R_{(5,3)}$	$R_{(5,4)}$
$L_{(6,3)}$	$L_{(6,2)}$	$L_{(6,1)}$	$R_{(6,2)}$	$R_{6,3)}$	$R_{(6,4)}$
$L_{(7,3)}$	$L_{(7,2)}$	$L_{(7,1)}$	$R_{(7,2)}$	$R_{(7,3)}$	$R_{(7,4)}$
$L_{(8,3)}$	$L_{(8,2)}$	$L_{(8,1)}$	$R_{(8,2)}$	$R_{(8,3)}$	$R_{(8,4)}$

4.1.4 Preparing "Job" File for Supercomputer

Although, small block lengths such as $K = 4, 8$ or 16 can give results on individual computers after few hours, moderate and longer block lengths take much more times, like days. In order to take more smoother results in less time we should use supercomputer. University of Southampton supercomputing facility "Lyceum" consists of 16 powerful computers and each one can be run 24 hours. As an example, to obtain BLER of $(K = 512, M = 1024)$, from 0 to 5 E_s/N_o, job divided into 6 powerful computers and each one run 24 hours. After simulation completed, results are combined to draw curve.

4.2 Proposed BP Decoding Schedule 1

The steps to be followed are exactly the same except for the polar decoder part in the original BP decoding steps, so in this section only the proposed polar decoder will be presented in detail. Moreover, similar stages will not be repeated in original BP decoder such as determine exact positions of the frozen bits before decoding process, decide $max_{iteration}$ number to get stable LLRs , initialize $L_{(i,j)}$ and $R_{(i,j)}$, and as final hard decision step. If we clarify more clearly, proposed BP decoding schedule can be easily applied only by modifying the computing part of LLRs of L and R Matrices.

We should emphasize that in original BP decoding schedule, after the completion of each node in one stage, the other stage is passed, as it can be seen in the Table 2. According to our proposed schedule, we first calculate the odd nodes in one stage, which are represent by $(i = odd, j)$, then the other stage is passed, after all the odd nodes ends in each stage, we calculate the only even nodes for each stage.

We can summarize the proposed schedule as follows

1) Compute odd nodes in each stage for L matrix

2) Compute odd nodes in each stage for R matrix

3) Compute even nodes in each stage for L matrix

4) Compute even nodes in each stage for R matrix

5) Repeat above stages until reach $max_{iteration}$ number

To make it more understandable, followed schedule when (K=4,M=8) is given in Table 3 as an example. Every column should be followed from top to bottom and then should be pass other column.

Table 3: Proposed BP 1 Scheduling to Update LLRs Values in One Iteration for $(K = 4, M = 8)$

1	2	3	4	5	6	7	8	9	10	11	12
Stage3	Stage2	Stage1	Stage1	Stage2	Stage3	Stage3	Stage2	Stage1	Stage1	Stage2	Stage3
$L_{(1,3)}$	$L_{(1,2)}$	$L_{(1,1)}$	$R_{(1,2)}$	$R_{(1,3)}$	$R_{(1,4)}$	$L_{(2,3)}$	$L_{(2,2)}$	$L_{(2,1)}$	$R_{(2,2)}$	$R_{(2,3)}$	$R_{(2,4)}$
$L_{(3,3)}$	$L_{(3,2)}$	$L_{(3,1)}$	$R_{(3,2)}$	$R_{(3,3)}$	$R_{(3,4)}$	$L_{(4,3)}$	$L_{(4,2)}$	$L_{(4,1)}$	$R_{(4,2)}$	$R_{(4,3)}$	$R_{(4,4)}$
$L_{(5,3)}$	$L_{(5,2)}$	$L_{(5,1)}$	$R_{(5,2)}$	$R_{(5,3)}$	$R_{(5,4)}$	$L_{(6,3)}$	$L_{(6,2)}$	$L_{(6,1)}$	$R_{(6,2)}$	$R_{(6,3)}$	$R_{(6,4)}$
$L_{(7,3)}$	$L_{(7,2)}$	$L_{(7,1)}$	$R_{(7,2)}$	$R_{(7,3)}$	$R_{(7,4)}$	$L_{(8,3)}$	$L_{(8,2)}$	$L_{(8,1)}$	$R_{(8,2)}$	$R_{(8,3)}$	$R_{(8,4)}$

4.3 Proposed BP Decoding Schedule 2

Similar to the proposed BP decoding schedule 1, similar stages will not be repeated for proposed schedule 2, we only explain the computing part of LLRs of L and R matrices. For this schedule we will followed vice versa of the schedule 1. By expressing it more clearly we will compute even nodes first, and then we will compute odd nodes. We can summarize the proposed schedule as follows

1) Compute even nodes in each stage for L matrix

2) Compute even nodes in each stage for R matrix

3) Compute odd nodes in each stage for L matrix

4) Compute odd nodes in each stage for R matrix

5) Repeat above stages until reach $max_{iteration}$ number

And in Table 4 followed schedule is given in as an example, when (K=4,M=8). Every column should be followed from top to bottom and then should be pass other column.

Table 4: Proposed BP 2 Scheduling to Update LLRs Values in One Iteration for $(K = 4, M = 8)$

1	2	3	4	5	6	7	8	9	10	11	12
Stage3	Stage2	Stage1	Stage1	Stage2	Stage3	Stage3	Stage2	Stage1	Stage1	Stage2	Stage3
$L_{(2,3)}$	$L_{(2,2)}$	$L_{(2,1)}$	$R_{(2,2)}$	$R_{(2,3)}$	$R_{(2,4)}$	$L_{(1,3)}$	$L_{(1,2)}$	$L_{(1,1)}$	$R_{(1,2)}$	$R_{(1,3)}$	$R_{(1,4)}$
$L_{(4,3)}$	$L_{(4,2)}$	$L_{(4,1)}$	$R_{(4,2)}$	$R_{(4,3)}$	$R_{(4,4)}$	$L_{(3,3)}$	$L_{(3,2)}$	$L_{(3,1)}$	$R_{(3,2)}$	$R_{(3,3)}$	$R_{(3,4)}$
$L_{(6,3)}$	$L_{(6,2)}$	$L_{(6,1)}$	$R_{(6,2)}$	$R_{(6,3)}$	$R_{(6,4)}$	$L_{(5,3)}$	$L_{(5,2)}$	$L_{(5,1)}$	$R_{(5,2)}$	$R_{(5,3)}$	$R_{(5,4)}$
$L_{(8,3)}$	$L_{(8,2)}$	$L_{(8,1)}$	$R_{(8,2)}$	$R_{(8,3)}$	$R_{(8,4)}$	$L_{(7,3)}$	$L_{(7,2)}$	$L_{(7,1)}$	$R_{(7,2)}$	$R_{(7,3)}$	$R_{(7,4)}$

4.4 Proposed BP Decoding Schedule 3

As proposed BP decoding schedule 3, we will apply SCAN decoding, only difference than SCAN decoding which mentioned in Subsection 2.2.3.5, is that we will not compute LLRs for frozen bits, and this will reduce the complexity and required space in memory. As an example, in Fig.28, red nodes are not computed because they belong the frozen bits, and this influence complexity. Depends on the frozen bit pattern an frozen bit positions, if information bits length increases, the number of nodes which are not computed will increase. The numbers of reduced nodes are presented in Table 5 for different coding rates and lengths. Moreover,in Fig.29, comparison of node numbers to be calculated for SCAN and proposed SCAN is presented for longer block lengths as graph. Above all, we should highlighted that we applied SC schedule for proposed SCAN with updating propagation messages iteratively in both ways as soft information for both $L_{(i,j)}$ and $R_{(i,j)}$.

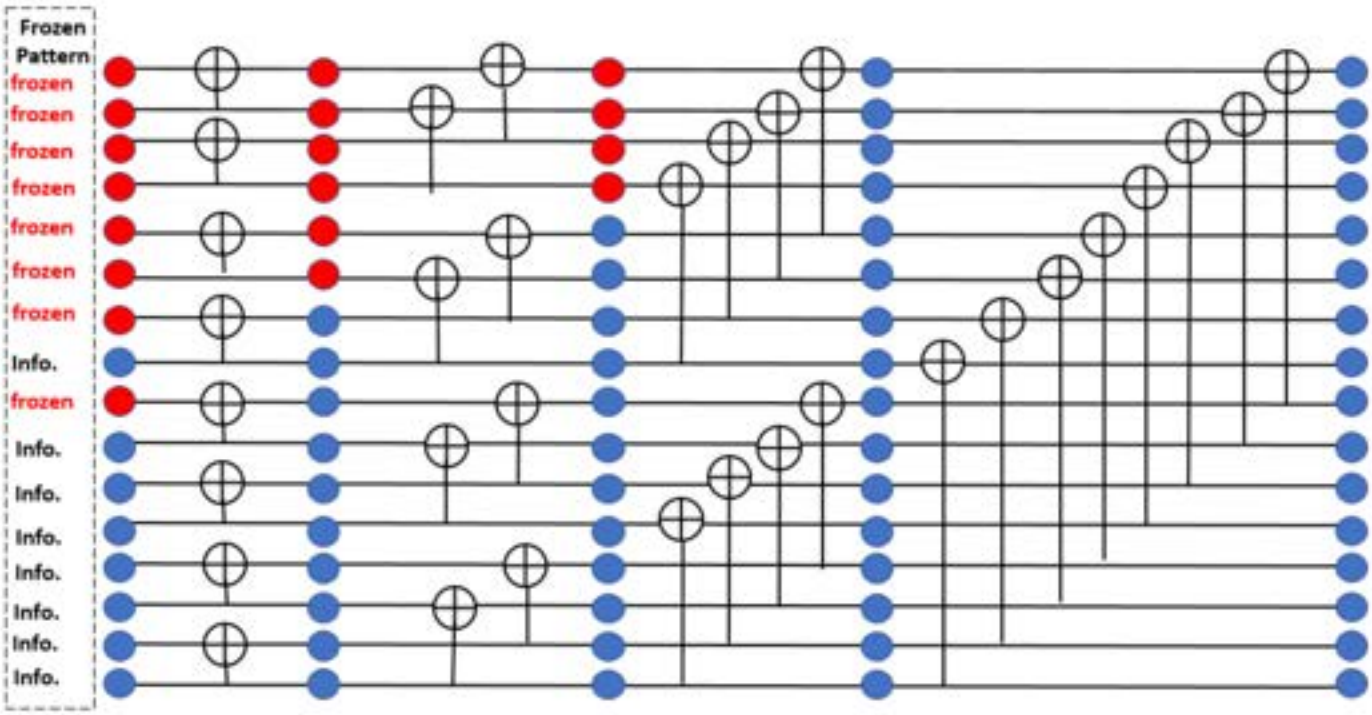

Figure 28: Proposed SCAN Algorithm Factor Graph When (K=8, M=16) for Compute LLRs of $L_{(i,j)}$

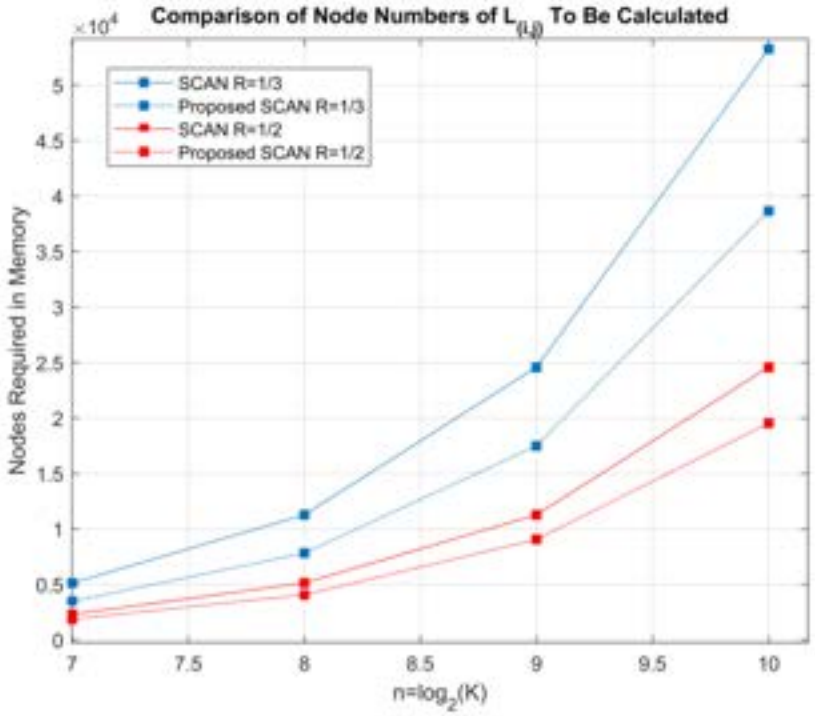

Figure 29: Comparison of Required Node Numbers in Memory of $L_{(i,j)}$ To Be Calculated for SCAN and Proposed SCAN

Table 5: Comparison of Node Numbers of $L_{(i,j)}$ To Be Calculated for SCAN and Proposed SCAN

Code Lengths	SCAN Node Num.	Proposed SCAN Node Num.	Reduced Node Num.
(K=8,M=16,R=1/2)	80	64	16
(K=16,M=32,R=1/2)	192	152	40
(K=32,M=64,R=1/2)	448	364	84
(K=64,M=128,R=1/2)	1024	828	196
(K=128,M=216,R=1/2)	2304	1852	452
(K=216,M=512,R=1/2)	5120	4064	1056
(K=512,M=1024,R=1/2)	11264	9052	2212
(K=1024,M=2048,R=1/2)	24576	19528	5048
(K=128,M=384,R=1/3)	5120	3464	1656
(K=216,M=648,R=1/3)	11264	7812	3452
(K=512,M=1536,R=1/3)	24576	17480	7096
(K=1024,M=3072,R=1/3)	53248	38660	14588

4.5 Complexity Analysis of Decoding Methods

In this subsection, we will evaluate complexities of all decoding methods which mentioned in this paper.

We know that SC decoding has the smallest complexity, where N is the block length with $O(NlogN)$. And as it can be seen in the result section SCL has better performance than SC with higher complexity which is $O(LNlogN)$. Although required more memory space and more computation complexity, Original BP has slightly better performance than SC for moderate and longer code lengths. And we can said that proposed schedule 1 and 2 increase required steps to complete each iteration but doing

less computing, and they present same BLER performance with Original BP. However, we could not observed considerable reducing in the decoding time although they required less computing for each step. Furthermore, we observed proposed schedule 3 which called as proposed SCAN with same schedule of SC, has better performance than Original BP and we succeed the reduced the complexity and required space in the memory according to original SCAN by not computing frozen bits LLRs when we updating right-to-left propagation messages. However, in terms of performances there will be no difference as it can be seen in the the result section.

5 Results

This section, presents BLER vs. E_s/N_0 performances obtained from the MATLAB software by basing on implementation section,and using Lyceum supercomputer. In Subsection 5.1, SC decoding performance presents considering different coding rates and block lengths. Similarly, in Subsection 5.2, SCL decoding performance presents also considering different list sizes. In Subsection 5.3, proposed schedules performances present with comparing Original BP decoding. We should highlighted that all decoding methods will use SC decoding performance as reference to investigate improvement on BLER. In order to draw BLER curves target frame errors are taken as 10000 with simulating 10000 frames.

5.1 BLER Performances of SC Decoding Method

5.1.1 BLER Performances Considering Coding Rate for SC

As it can be seen the Fig.30, when the information block length K is fixed and encoded block length M is increased, performance of block error rate increases. It can be easily seen that, at the point of 10^{-2}, there is 2 dB improvement between coding rates $R = 1/3$ and $R = 1/2$. The reason of this, when we increase the encoded block length, we increase the number of frozen bits, hence channel divided smaller pieces and this makes positions to send informations bits more reliable. However, for longer and moderate block lengths this advantage can be turn into disadvantages because of the complexity and requiring of more memory space. Moreover, they take more time and occupy more bandwidth.

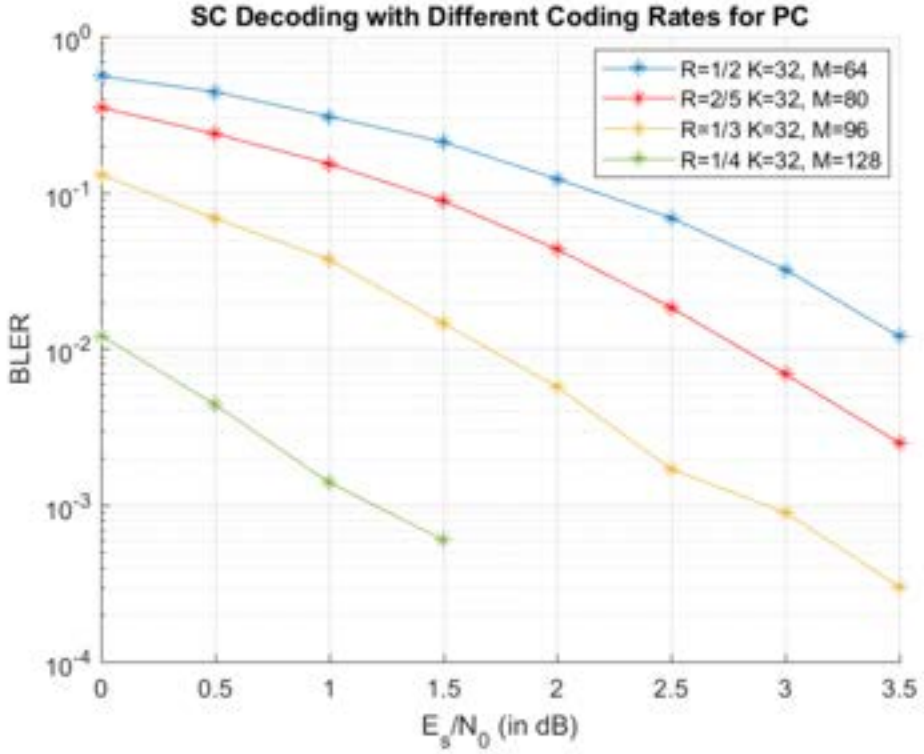

Figure 30: SC Decoding for Different Coding Rates When K=32

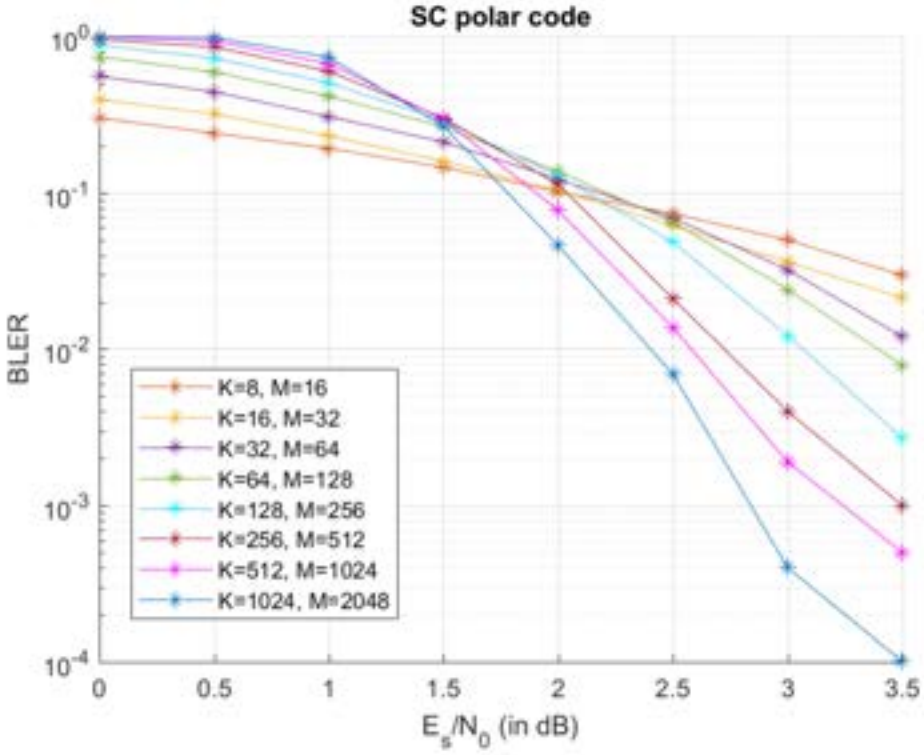

Figure 31: SC Decoding for Different Block Length When R=1/2

5.1.2 BLER Performances Considering Block Length for SC

Fig.31 represents BLER performance under SC decoding for different block lengths when the code rate fixed and $R = 1/2$. It can be easily seen that although longer block lengths giving worse performance under 1.5 dB, they have better performances after 1.5 dB. Also smaller code lengths such as $K = 8$ cannot reached the even the point of 10^{-2} for 3.5 dB, while longer code lengths such as $K = 1024$ reaches easily the point of 10^{-4}.

5.2 BLER Performances of SCL Decoding Method

5.2.1 BLER Performances Considering Coding Rate for SCL

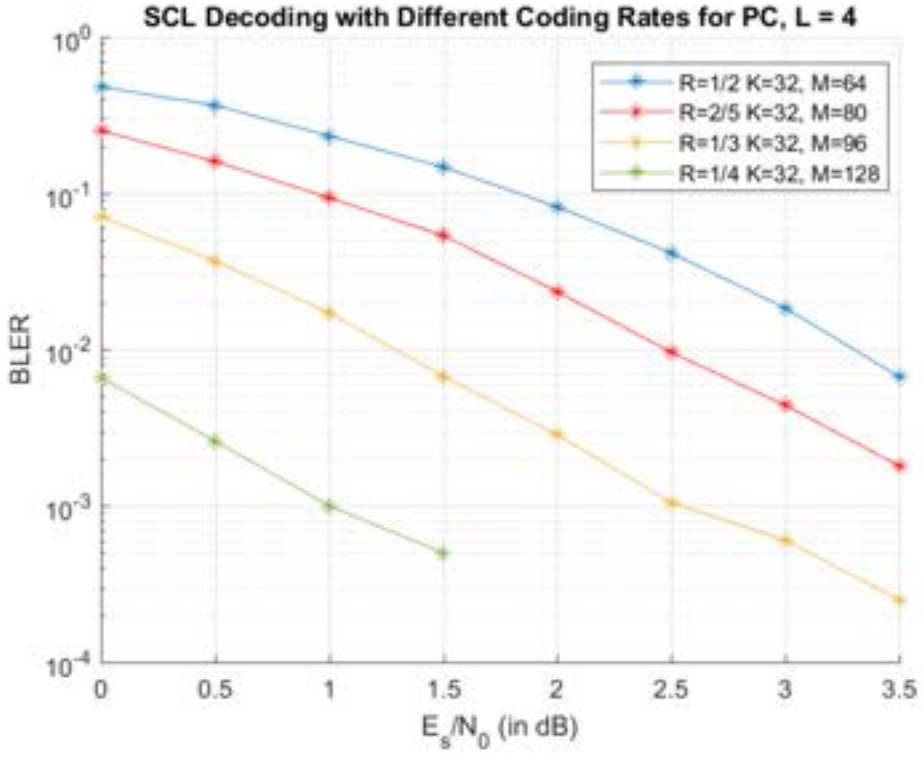

Figure 32: SCL Decoding for Different Coding Rates When K=32 and L=4

Similarly SC decoding, when the information block length K and list size L are fixed and encoded block

length M is increased for SCL decoding, performance of block error rate increases. It can be easily seen that in Fig.32, at the point of 10^{-2}, there is same 2 dB improvement between coding rate $R = 1/3$ and $R = 1/2$ with same reasons as SC decoding.

5.2.2 BLER Performances Considering Block Length for SCL

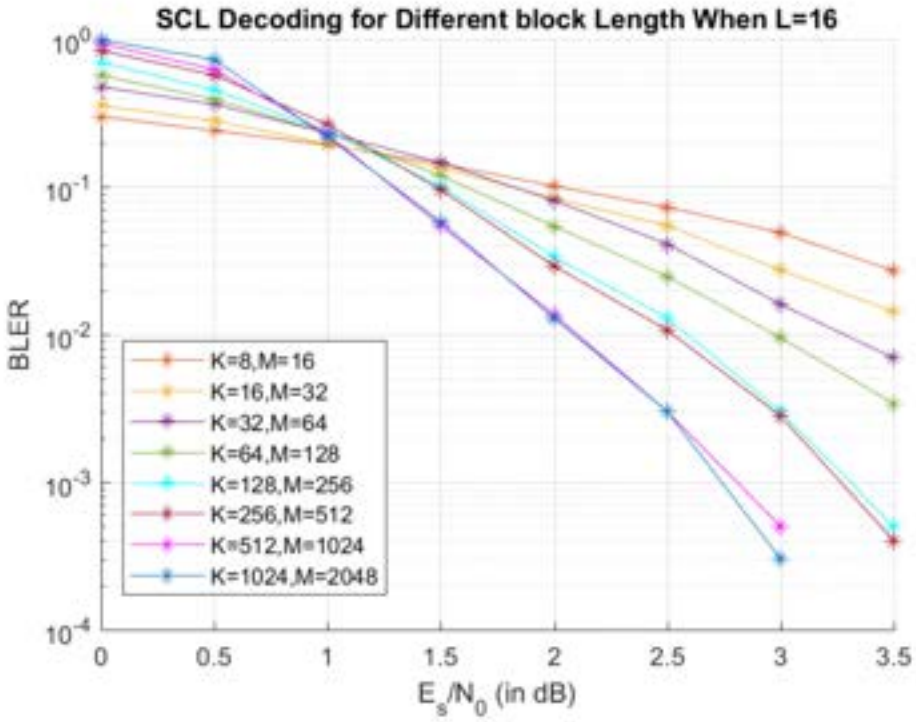

Figure 33: SCL Decoding for Different Block Length When R=1/2 and L=16

Similarly to the SC decoding, Fig.33 shows that BLER performance of SCL decoding for different block lengths when the code rate $R = 1/2$ and $L = 16$ are fixed although longer block lengths giving worse performance under 1 dB, they have better performances after 1 dB. Hence, we can say that there is 0.5 dB improvement between SC and SCL in terms of block lengths.

5.2.3 BLER Performances Considering List Size for SCL

Fig.34 presents SCL decoding for different list size when polar code ($K = 512, M = 1024$). In this figure $L = 1$ represents SC decoding. The increase in performance as you grow list size can easily be seen. At the point of 10^{-2}, there is almost 1.5 dB improvement between list size $L = 32$ and SC decoding. Similarly, for Fig.35 for (1024, 2048) shows that increasing list size increases BLER performance. Also we should emphasis that there is big improvement between $L = 1$ and $L = 2$, although after one point there is no big improvement between list size such as $L = 8$ and $L = 32$.

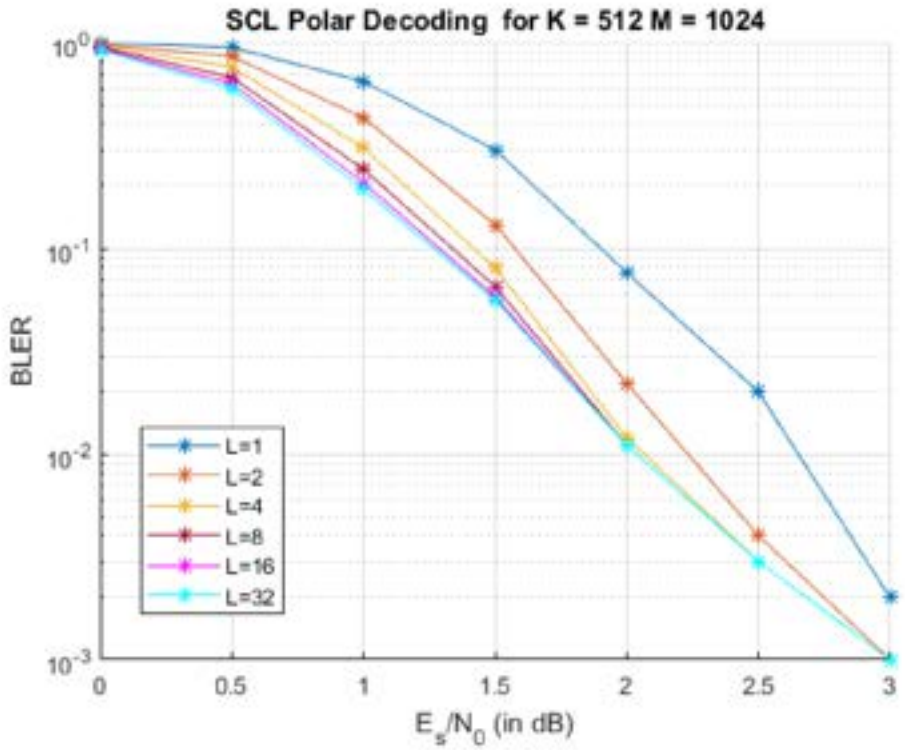

Figure 34: SCL Decoding with Different List Size for (512, 1024) Polar Code

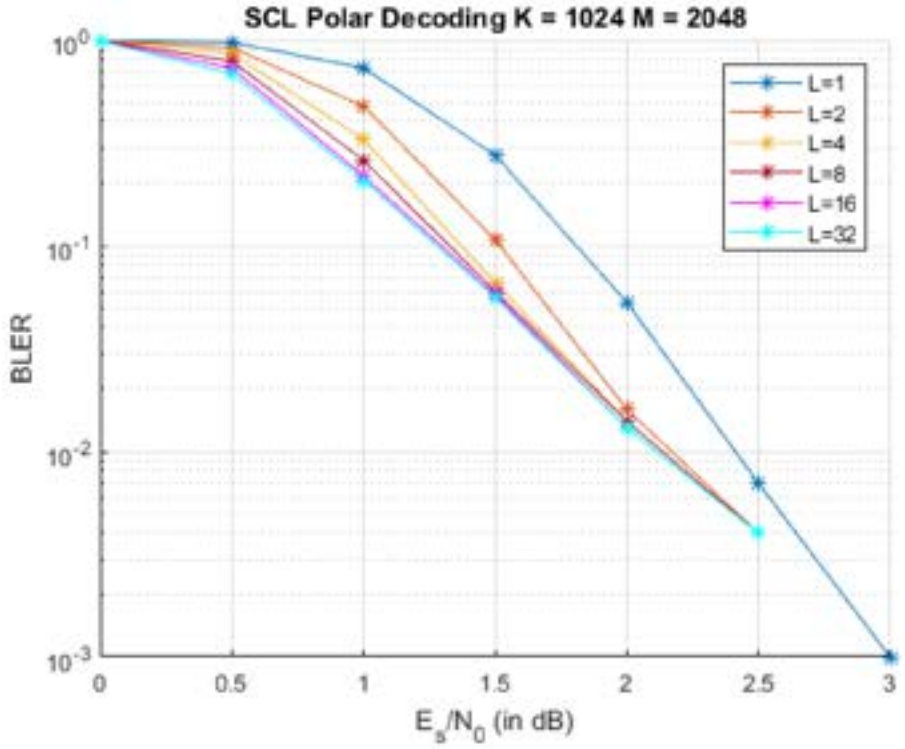

Figure 35: SCL Decoding with Different List Size for (1024, 2048) Polar Code

5.3 BLER Performances of Original BP and Proposed BP Schedules

5.3.1 BLER Performances Considering Max Iteration Number

As it mentioned before, for BP decoder we need $max_{iteration}$ number to get stable LLRs. As it can be seen the Fig.36 for Original BP, $max_{iteration} = 30$ will be enough for all block lengths. Although small code lengths reaches target BLER with small number of iterations,(it can be seen for (K=32,M=64) even 7 iteration will be enough) we need to decide on a fixed number of max iteration to make a fair comparison between BLER performances. Hence, we decided $max_{iteration} = 30$.

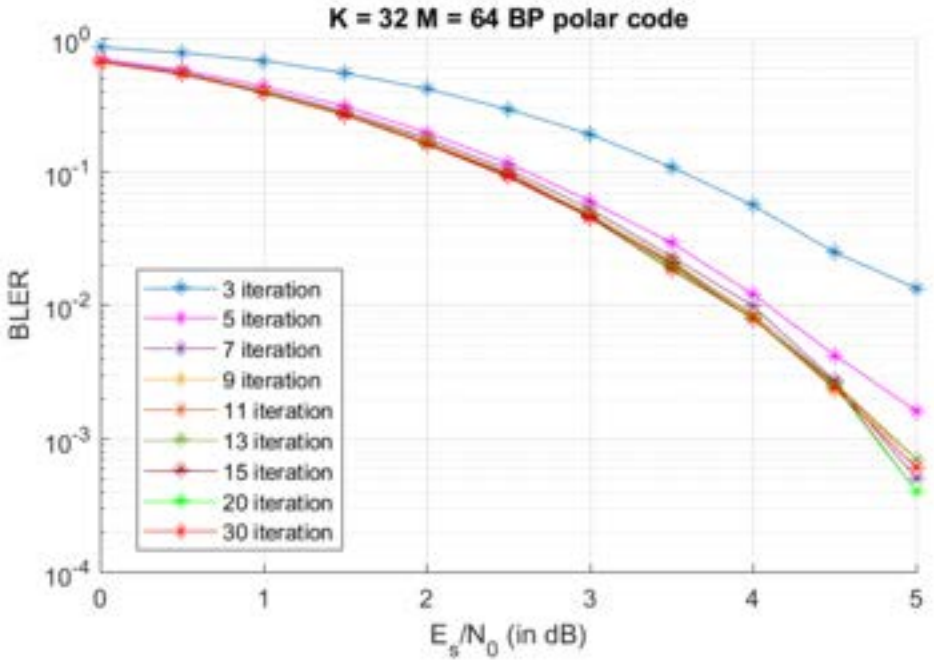

Figure 36: Iteration effect on BLER Performance for Original BP Decoding for (K=32, M=64)

5.3.2 Comparison of BLER Performance of MS/ SMS/ Original-BP Algorithms

From the Fig.37 we can said that for small block lengths there is no difference in terms of performance among MS-BP, SMS-BP and Original BP, although MS-BP and SMS-BP have less computational complexity. However, when we increase the block lengths like in Fig.38 and Fig.39 , we can observe that MS-BP cannot achieve same performance with SMS-BP and Original BP. Hence, we can recommend that for all block lengths SMS-BP can be used instead of Original BP with less complexity but almost same performance. As we mentioned before, we used scaling parameter "$s = 0.9375$" which proposed in [18]

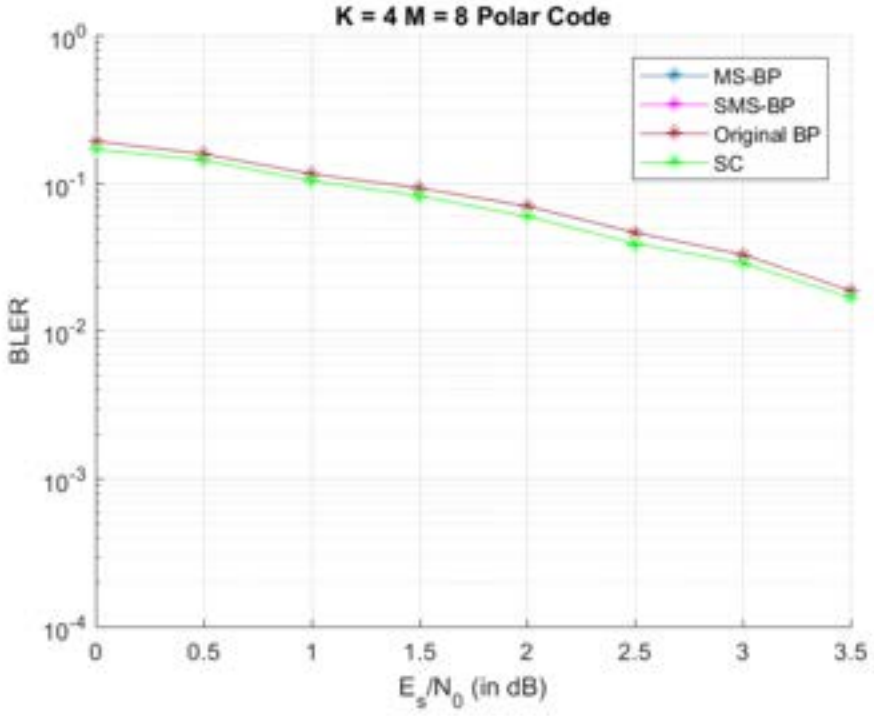

Figure 37: Comparison of BLER Performance of MS-BP, SMS-BP and Original-BP Algorithms, When (K=4,M =8) with $max_{iteration} = 30$ and scaling parameter "$s = 0.9375$" for SMS-BP

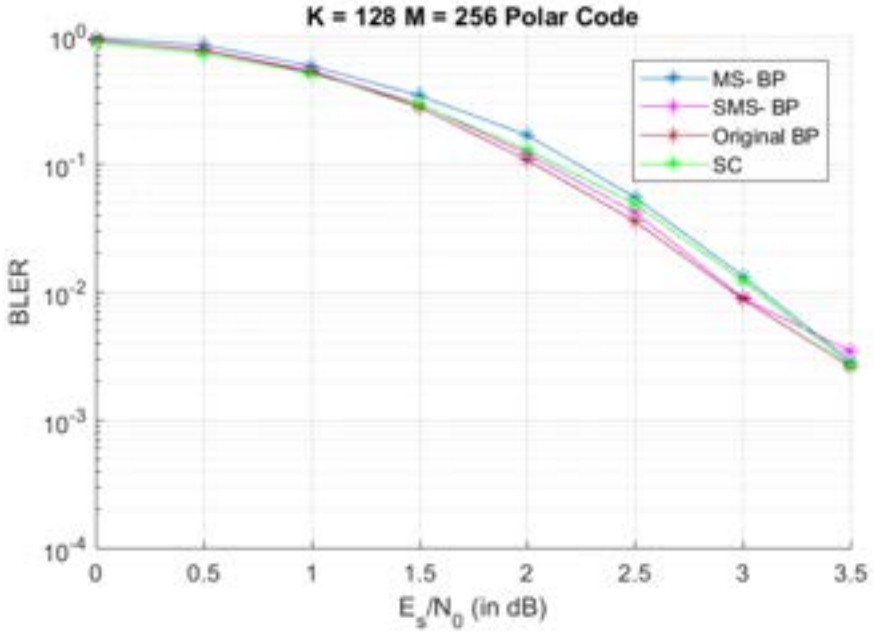

Figure 38: Comparison of BLER Performance of MS-BP, SMS-BP and Original-BP Algorithms, When (K=128,M =256) with $max_{iteration} = 30$ and scaling parameter "$s = 0.9375$" for SMS-BP

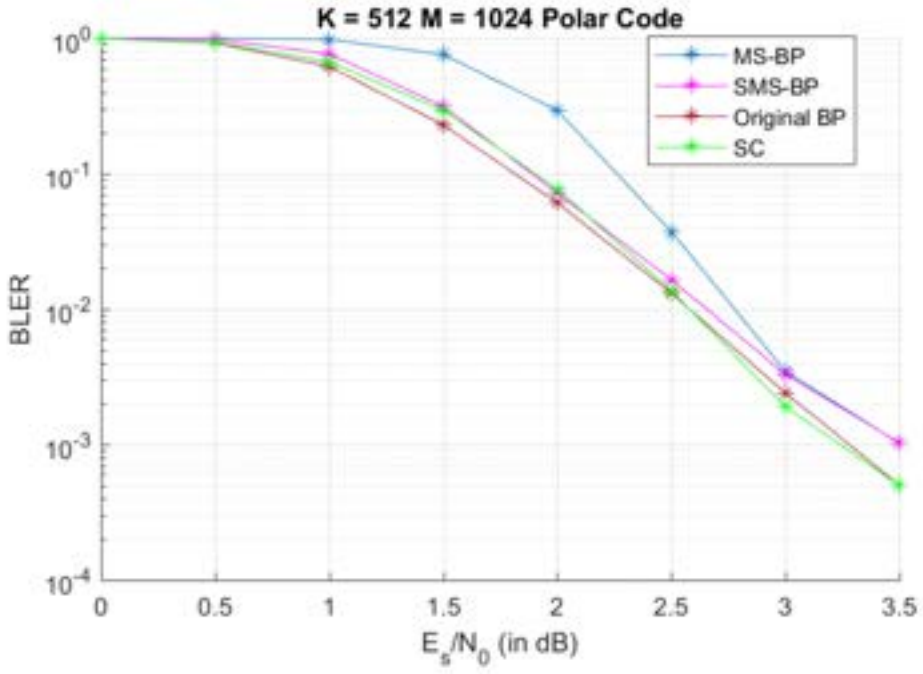

Figure 39: Comparison of BLER Performance of MS-BP, SMS-BP and Original-BP Algorithms, When (K=512,M =1024) with $max_{iteration} = 30$, and scaling parameter "$s = 0.9375$" for SMS-BP

5.3.3 Comparison of BLER Performances of SC and Original BP

In Fig.40, we have compared Original BP with SC decoding performances, we can see that BP can have slightly better performance than SC for moderate and longer block lengths, while it has worse performance for small block lengths.

5.3.4 Comparison of BLER Performances of Proposed SCAN and Original BP

From the Fig.41 we can observed that even 5 iteration for proposed SCAN has better performance than Original BP which obtained with 30 iteration. Furthermore, we can see that there is no huge difference between 30 and 5 iteration of proposed SCAN. Hence, we can obtain better performance with less complexity according to BP by using SCAN decoding.

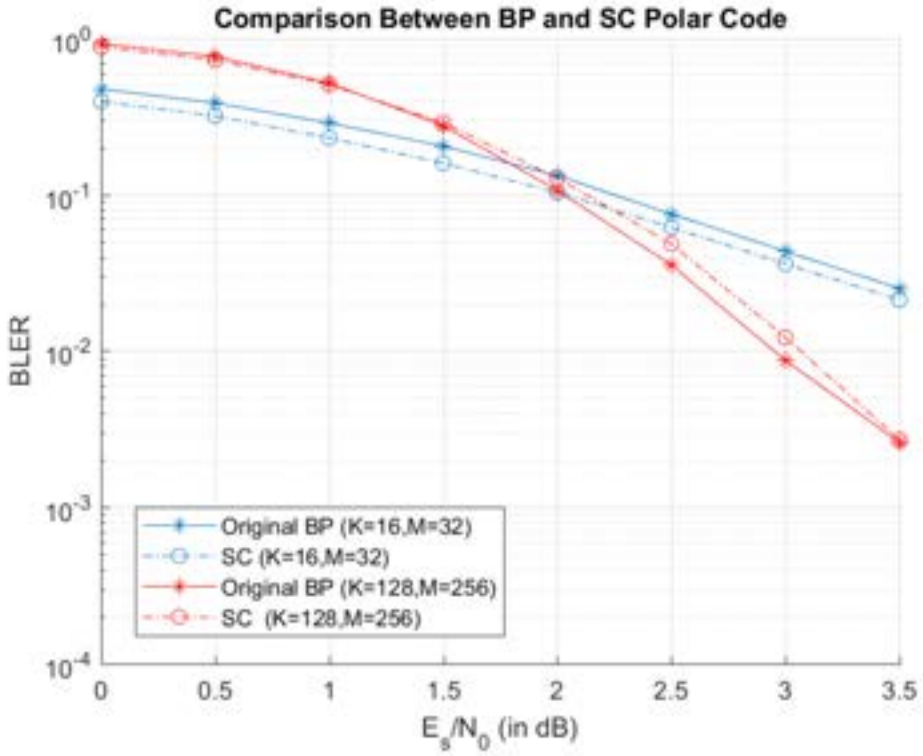

Figure 40: BLER Performance of SC and Original BP for Different Block Size

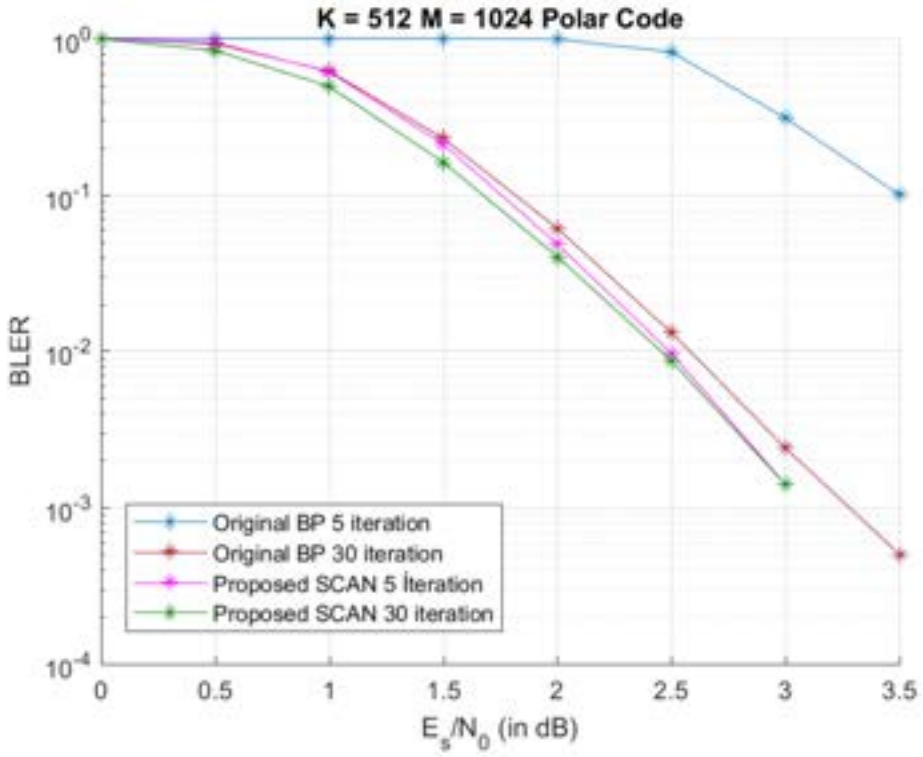

Figure 41: BLER Performance of Proposed SCAN and Original BP for $(K = 512, M = 1024)$

5.3.5 BLER Performances of Proposed Schedules

Fig.42 shows, proposed schedule 1 and 2 have the same performances with Original BP, although SC decoding has better performance than all of them for small code lengths ($K = 16, M = 32$), while SCAN has better performance than BP. When we look at the moderate code lengths with ($K = 128, M = 256$) in Fig.43, SCAN has the better performance like for longer code lengths which is given in Fig.44 with ($K = 1024, M = 2048$).

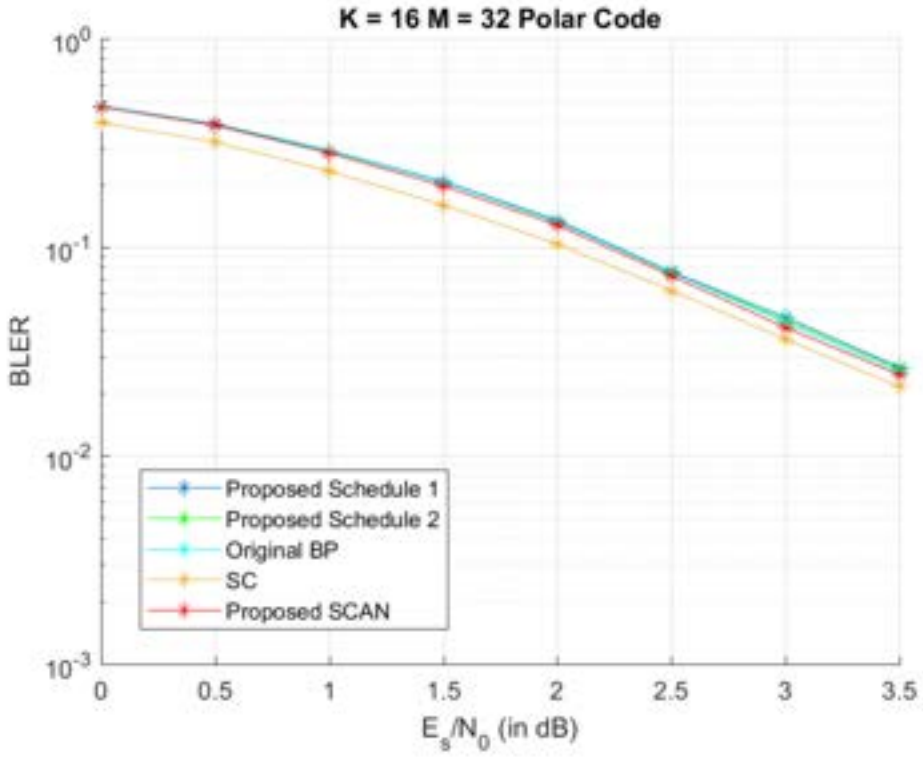

Figure 42: BLER Performances of Proposed Schedules for ($K = 16, M = 32$), when $max_{iteration} = 30$

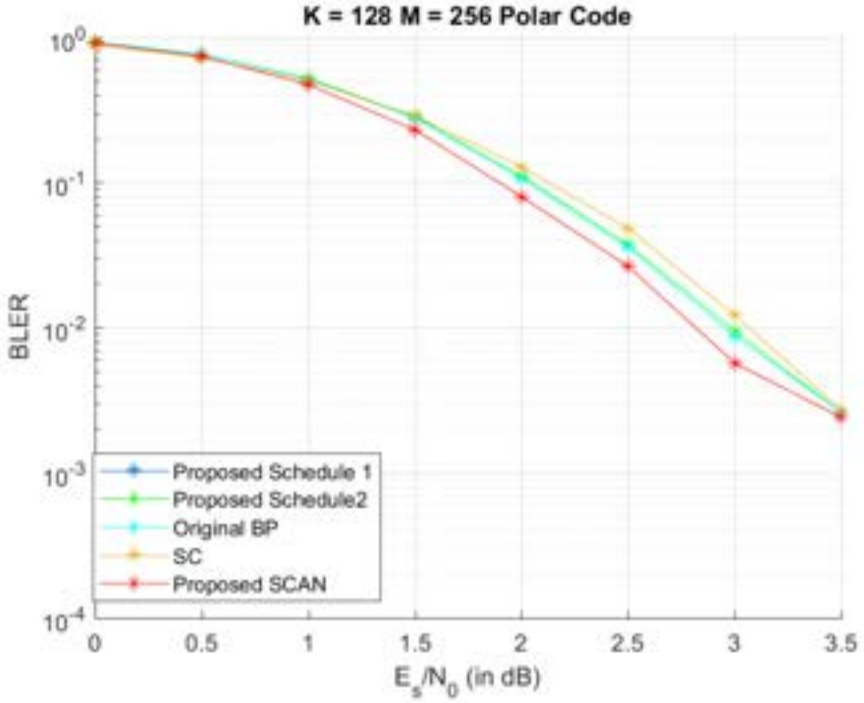

Figure 43: BLER Performances of Proposed Schedules for $(K = 128, M = 256)$, when $max_{iteration} = 30$

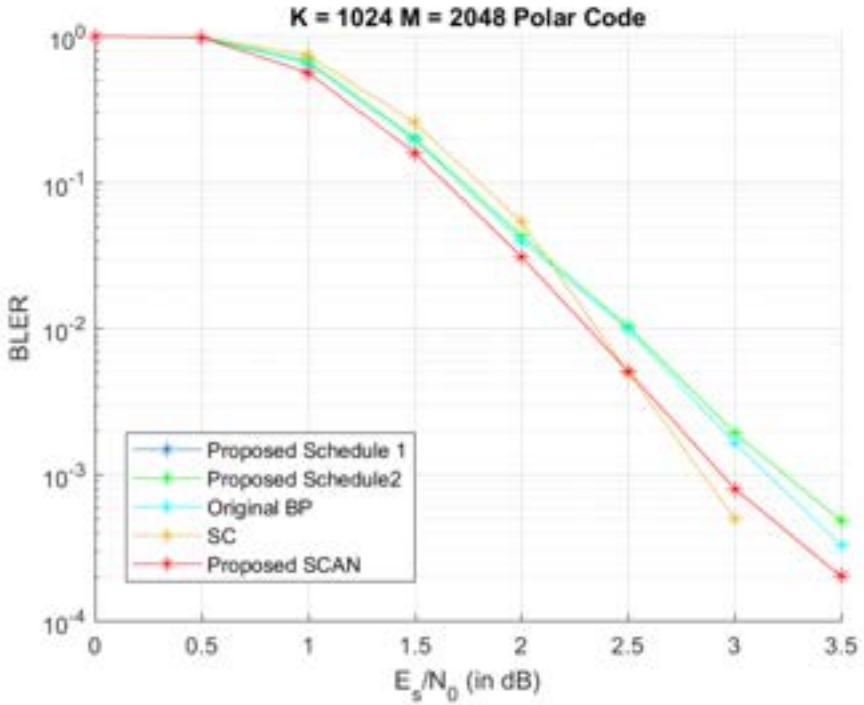

Figure 44: BLER Performances of Proposed Schedules for $(K = 1024, M = 2048)$, when $max_{iteration} = 30$

6 Reflection

This section comments on how well the planning section has been followed and reflects on what has been learned from completing the project as well as present some ideas for the future development of the project.

The project successfully meets top- level objectives in section 3.1 and project tasks in section 3.2 together with all of the success criteria listed in section 3.3. Although the planning section was followed up well, there was a waste of time to complete some tasks because of some obstacles such as finding detailed resources, problems in algorithms and simulations which take days to run on individual computer. If we specify problems in more detail, firstly, writing an example for SCL took much more time than our expectation because we could not reach a source which has the detailed explanation of SCL decoding for each step, also because of the complexity of SCL, preparing path and graph representation figures for each step took much more time than our expectation. The second problem we encountered was the MATLAB code of BP algorithm, because in the beginning we used different factor graphs for encoding and decoding. Hence, we obtained wrong results before debug the problem. Another problem we are facing is that the simulations took days on the personal computer since we start using the super computer late and we did not send enough frames and target frame errors to simulate them so we could not get smooth curves. So we had to resume all simulations again but this time using super computer. As a result of this, we spent more time on simulations than what we had on Gantt Chart.

During the this process we learned to use time efficiently and detailed knowledge about PC. Consequently, some aspects of the developed project could have gone better, if the Gantt Chart was prepared much more carefully considering all risks but we should consider that time was very limited for this project. For any future project, checking process time can be allocated for revising the work between tasks on Gantt Chart.

7 Conclusion

In order to provide effective mobile communication, FEC codes are needed in noisy channels. Recently introduced Polar Codes attract attention owing to their capacity achieving performance, excellent error correction performance and fine rate flexibility and they have been adopted as a candidate of forward error correction codes for 5th generation mobile communication due to many reasons mentioned above. In Section 2.1, background knowledge for polar encoding and decoding process have been presented. In Section 2.2, common decoding algorithms which are SC, SCL, BP have been introduced with detailed hand calculation examples to make them more understandable for readers.

This paper provides a contribution to investigate different BP decoding schedules impact on BLER performances. These schedules are introduced in Section 4 with their complexity analysis and results have been presented in Section 5. According to these results, Original BP has slightly better performance than SC for moderate and longer code lengths, although required more memory space and more computation complexity. And we can said that proposed schedule 1 and 2 increase required steps to complete each iteration but doing less computing, and they present same BLER performance with Original BP. However, we could not observed considerable reducing in the decoding time although they required less computing for each step. Furthermore, we observed proposed schedule 3 which called as proposed SCAN with same schedule of SC, has better performance than Original BP and we succeed the reduced the complexity and required space in the memory according to original SCAN by not computing frozen bits LLRs when we updating right-to-left propagation messages.

Consequently, this project can be used as a source for anyone who undergraduate or postgraduate level and interested in learning about Polar Codes and who want to learn impacts of different BP schedules on BLER performance. For the future development of the project, early stopping criteria can be studying for proposed schedules to decrease complexity and increase speed.

References

[1] Arikan, E. Channel polarization:" A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels". IEEE Trans. Inf. Theory 2009, 55, 3051–3073.

[2] Raymond, A. J. and Gross,W. J." A Scalable Successive-Cancellation Decoder for Polar Codes",IEEE TRANSACTIONS ON SIGNAL PROCESSING, Vol. 62, No. 20, October 15, 2014,5339-5347

[3] Hwan, J., Hyo, S., Woong, J. and Sik, Y., "Low Complexity List Decoding for Polar Codes with Multiple CRC Codes",Entropy,VOL. 19, NO. 183,April 24, 2017

[4] Maunder, R. G.,"Polar codes", CTO, AccelerComm, 2017

[5] ZTE, ZTE Microelectronics,"Performance Comparison of eMBB's Channel coding candidates for Short Block Lengths", 3GPP TSG RAN WG1 Meeting 87, R1-1611107,Reno, USA 14th - 18th November 2016

[6] Huawei, HiSilicon,"Performance evaluation of channel codes for small block sizes", 3GPP TSG RAN WG1 Meeting 87, R1-1611256,Reno, USA 14th - 18th November 2016

[7] http://www2.egr.uh.edu/ zhan2/ECE6332/slidesarikan.pdf

[8] Yuan,B. and Keshab K.,P.," Low-Latency Successive-Cancellation Polar Decoder Architectures Using 2-Bit Decoding",IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS, VOL. 61, NO. 4, APRIL 2014,pp.1241-1254

[9] Huawei, HiSilicon,"Polar Codes: Encoding and Decoding" ,R1-164039,Gothenburg,Sweden, 22nd-26th August,2016

[10] Huawei, HiSilicon,"Polar Code Design and Rate Matching" ,R1-167209,Gothenburg,Sweden, 22nd-26th August,2016

[11] Tal, I.; Vardy, A." List decoding of polar codes". IEEE Trans. Inf. Theory 2015, 61, 2213–2226.

[12] Balatsoukas-S., A., Parizi, M.," LLR-Based Successive Cancellation List Decoding of Polar Codes",IEEE TRANSACTIONS ON SIGNAL PROCESSING, VOL. 63, NO. 19, OCTOBER 1, 2015, 5165-5179

[13] Fayyaz, U. and Barry, J.,Low-Complexity Soft-Output Decoding of Polar Codes,IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, VOL. 32, NO. 5, MAY 2014

[14] S. M. Abbas, Y. Z. Fan, J. Chen, and C. Y. Tsui, "High-Throughput and Energy-Efficient Belief Propagation Polar Code Decoder," IEEE Trans. Very Large Scale Integr. Syst., vol. 25, no. 3, pp. 1098–1111, 2017.

[15] J.Lin,J.Sha,L. Li, Z. Wang and others, " A High-Throughput Belief Propagation Decoder Architecture for Polar Codes," IEEE Trans. 978-1-4799-5341-7/16/ .pp. 153–156, 2016.

[16] U. U. Fayyaz and J. R. Barry, "Polar codes for partial response channels," IEEE Int. Conf. Commun., pp. 4337–4341, 2013.

[17] U. U. Fayyaz and J. R. Barry, "A low-complexity soft-output decoder for polar codes," GLOBECOM - IEEE Glob. Telecommun. Conf., vol. 2, pp. 2692–2697, 2013.

[18] "Bo Yuan and Keshab K . Parhi Department of Electrical and Computer Engineering , University of Minnesota Twin Cities," Architecture Optimizations for BP Polar Decoders" , no. 1, pp. 2654–2658, 2013.

[19] Y. Zhang, A. Liu, X. Pan, Z. Ye, and C. Gong, "A modified belief propagation polar decoder," IEEE Commun. Lett., vol. 18, no. 7, pp. 1091–1094, 2014.

[20] Y. Zhang, Q. Zhang, X. Pan, Z. Ye, and C. Gong, "A simplified belief propagation decoder for polar codes," 2014 IEEE Int. Wirel. Symp. IWS 2014, pp. 2–5, 2014.

[21] J. Xu, T. Che, and G. Choi, "XJ-BP: Express journey belief propagation decoding for polar codes," 2015 IEEE Glob. Commun. Conf. GLOBECOM 2015, 2015.

[22] Y. Ren, C. Zhang, X. Liu, and X. You, "Efficient early termination schemes for belief-propagation decoding of polar codes," Proc. - 2015 IEEE 11th Int. Conf. ASIC, ASICON 2015, 2016.

[23] J. Sha, J. Liu, J. Lin, and Z. Wang, "A Stage-Combined Belief Propagation Decoder for Polar Codes," J. Signal Process. Syst., pp. 1–8, 2016.

[24] J. Lin, Z. Yan, and Z. Wang, "Efficient Soft Cancelation Decoder Architectures for Polar Codes," IEEE Trans. Very Large Scale Integr. Syst., vol. 25, no. 1, pp. 87–99, 2017.

[25] G. Berhault, C. Leroux, C. Jego, and D. Dallet, "Hardware implementation of a soft cancellation decoder for polar codes," Conf. Des. Archit. Signal Image Process. DASIP, vol. 2015–December, 2015.

Appendix A

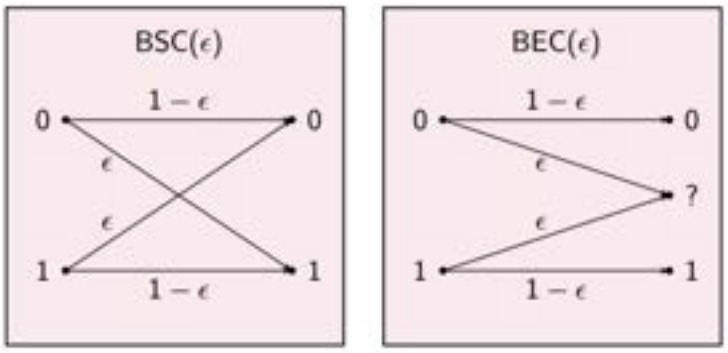

Figure 45: Channels Have Input-Output Symmetry and Discrete Memoryless [7]

In [1] the idea of polar code is described as construct a code sequence based on capacities of N channels which are independent copies of a given B-DMC where W_N^i polarized channels as it can be seen from Fig.45;

$$W_N^i = 1 \leqslant i \leqslant N) \tag{27}$$

The important thing is that only send data at rate 1 through those channels which are $I(W_N^i)$ near 1 and send data at rate 0 through the remaining. And in the Fig.46 redistributing of channel to generate polarized channels are showed.

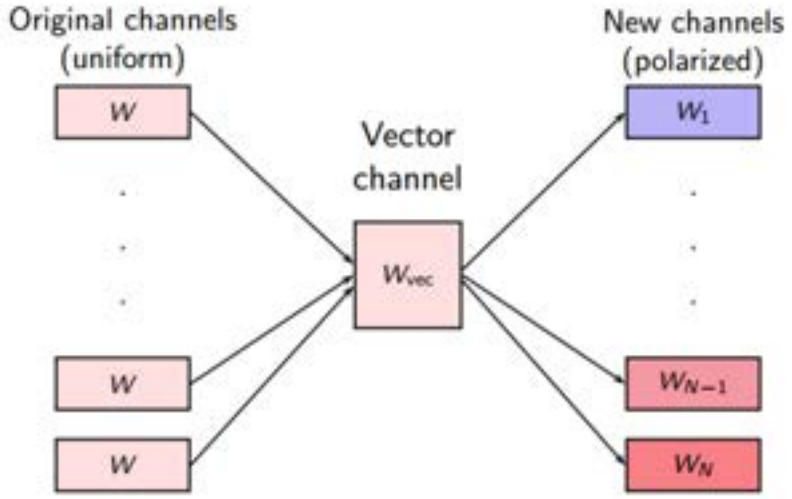

Figure 46: A Scheme of Redistribute of N Channels to Generate Polarized Channels W_N^i [7]