Status of thesis

Title of thesis | Novel Code-Construction for (3, *k*) Regular Low Density Parity Check Codes

I, SILVIA ANGGRAENI

hereby allow my thesis to be placed at the Information Resource Center (IRC) of Universiti Teknologi PETRONAS (UTP) with the following conditions:

1. The thesis becomes the property of UTP.
2. The IRC of UTP may make copies of the thesis for academic purposes only.
3. This thesis is classified as

☐ Confidential

☑ Non-confidential

If this thesis is confidential, please state the reason:

_____

_____

The contents of the thesis will remain confidential for _____ years.

Remarks on disclosure:

_____

Endorsed by

Signature of Author	Signature of Supervisor
	Name of Supervisor
Permanent: Komplek Timah Kelapa Dua,	Assoc Prof. Dr. Varun Jeoti
Blok CC 53 Depok – Jawa Barat,	Electrical and Electronic Department
Indonesia, 16951	Universiti Teknologi PETRONAS
	Tronoh, Bandar Seri Iskandar
	Perak-Malaysia
Date: March , 2010	Date: March , 2010

UNIVERSITI TEKNOLOGI PETRONAS

Approval by Supervisor

The undersigned certify that they have read and recommend to the postgraduate studies programme for acceptance, a thesis entitled "**Novel Code-Construction for (3, *k*) Regular Low Density Parity Check Codes**" submitted by (**SILVIA ANGGRAENI)** for the fulfillment of the requirements for the DEGREE OF MASTER OF SCIENCE IN ELECTRICAL AND ELECTRONIC ENGINEERING.

March , 2010

Date:

Signature : _____

Main Supervisor : Assoc.Prof. Dr.Varun Jeoti J.

Date : March , 2010

UNIVERSITI TEKNOLOGI PETRONAS

Novel Code-Construction for (3, *k*) Regular Low Density Parity Check Codes

By

Silvia Anggraeni

A THESIS

SUBMITTED TO THE POSTGRADUATE STUDIES PROGRAMME

AS A REQUIREMENT FOR THE

DEGREE OF MASTER OF SCIENCE IN

ELECTRICAL AND ELECTRONIC ENGINEERING PROGRAMME

BANDAR SERI ISKANDAR

PERAK

December - 2009

DECLARATION

I hereby declare that the thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at UTP or other institutions.

Signature : _____

Name : Silvia Anggraeni

Date : March , 2010

ACKNOWLEDGEMENTS

All praises be to ALLAH SWT, the most gracious and most merciful, who gave me strength, courage, ideas, love, hope and ways to solve some obstacles in this research work that enabled me to finish my study. I pray to Him to include me among those who are rewarded for just seeking knowledge.

I would like to express my gratitude and thanks to all people who encouraged and helped me throughout this research work.

First of all, I would like to express my gratitude and appreciation to my supervisor Assoc. Prof. Dr. Varun Jeoti. His support, encouragement, valuable assistance, and unwavering patience guided me throughout my research. It would have been impossible to finish this work without his powerful guidance. His positive influence on my professional development will be carried forward into my future career.

I would like to thank my lovely husband, my dearest parents, my lovely daughter, my sister, and my brother for their endless support throughout this research work. They gave me courage and strength to never give up during the hard situation in my research work. They provided me sturdy motivation and strong emotional and moral support which I have needed throughout this research and my whole life.

Special thanks are extended to my colleagues at Electrical and Electronics Department and my Indonesian friends for their support, help, and encouragement. They have been very friendly and gracious which provided me a good study environment. I would also like to acknowledge the support of Post Graduate office and Electrical and Electronics Department of Universiti Teknologi PETRONAS, MALAYSIA.

ABSTRACT

Communication system links that do not have the ability to retransmit generally rely on forward error correction (FEC) techniques that make use of error correcting codes (ECC) to detect and correct errors caused by the noise in the channel. There are several ECC's in the literature that are used for the purpose. Among them, the low density parity check (LDPC) codes have become quite popular owing to the fact that they exhibit performance that is closest to the Shannon's limit.

This thesis proposes a novel code-construction method for constructing not only (3, k) regular but also irregular LDPC codes. The choice of designing (3, k) regular LDPC codes is made because it has low decoding complexity and has a Hamming distance, at least, 4. In this work, the proposed code-construction consists of information sub-matrix (H_{inf}) and an almost lower triangular parity sub-matrix (H_{par}). The core design of the proposed code-construction utilizes expanded deterministic base matrices in three stages. Deterministic base matrix of parity part starts with triple diagonal matrix while deterministic base matrix of information part utilizes matrix having all elements of ones. The proposed matrix H is designed to generate various code rates (R) by maintaining the number of rows in matrix H while only changing the number of columns in matrix H_{inf}.

All the codes designed and presented in this thesis are having no rank-deficiency, no pre-processing step of encoding, no singular nature in parity part (H_{par}), no girth of 4-cycles and low encoding complexity of the order of $O(N + g^2)$ where $g^2 \ll N$. The proposed (3, k) regular codes are shown to achieve code performance below 1.44 dB from Shannon limit at bit error rate (BER) of 10^{-6} when the code rate greater than $R = 0.875$. They have comparable BER and block error rate (BLER) performance with other techniques such as (3, k) regular quasi-cyclic (QC) and (3, k) regular random LDPC codes when code rates are at least $R = 0.7$. In addition, it is also shown that the proposed (3, 42) regular LDPC code performs as close as 0.97 dB from Shannon limit at BER 10^{-6} with encoding complexity $O(1.0225\ N)$, for $R = 0.928$ and $N = 14364$ – a result that no other published techniques can reach.

ABSTRAK

Sambungan sistem komunikasi yang tidak memancar semula umumnya bergantung kepada forward error correction (FEC), error correction code (ECC) digunakan untuk mengesan dan memperbaiki kesalahan disebabkan oleh kebisingan dalam saluran. Ada beberapa ECC dalam kajian terdahulu yang digunakan untuk tujuan tersebut. Antara kesemuanya, low density parity check code (LDPC) telah menjadi pilihan utama kerana ia mempamerkan prestasi yang paling hampir dengan batas Shannon. Tesis ini mencadangkan sebuah kaedah baru pembinaan kod tidak hanya (3, *k*) reguler, tetapi juga kod-kod LDPC tidak reguler. Kod (3, *k*) LDPC reguler ini dibuat kerana ia memiliki kerumitan menyahkod rendah dan mempunyai jarak Hamming, 4. Dalam kajian ini, pembinaan kod yang dicadangkan terdiri daripada maklumat sub-matriks (H_{inf}) dan persamaan segitiga sub-matriks (H_{par}). Rekabentuk utama pembinaan kod yang dicadangkan menggunakan asas matriks penentu yang dikembangkan dalam tiga tahap. Matriks *H* yang dicadangkan direka untuk menghasilkan kod pelbagai peringkat (*R*) dengan mengekalkan jumlah baris matriks *H* di samping hanya menukar jumlah lajur matriks H_{inf}. Semua kod yang direka dalam tesis ini tidak mempunyai kekurangan peringkat, tidak ada pra-langkah pengolahan pengekodan, tidak ada sifat tunggal pada persamaan (H_{par}), tidak ada girth 4-kitaran dan pengekodan kerumitan yang rendah iaitu $O(N + g^2)$ di mana $g^2 \ll N$. Kod-kod (3, *k*) regular yang dicadangkan mencapai prestasi kod di bawah 1.44 dB daripada batas Shannon pada bit error rate (BER) daripada 10^{-6} ketika kadar kod lebih besar daripada $R = 0.875$. Mereka memiliki BER setanding dan prestasi kadar kesalahan blok (BLER) dengan teknik lain seperti (3, *k*) quasi-siklik reguler (QC) dan (3, *k*) LDPC kod rawak reguler ketika kadar kod sekurang-kurangnya $R = 0.7$. Selain itu, kajian menunjukkan (3, 42) kod reguler LDPC yang dicadangkan mempamerkan nilai paling hampir iaitu 0.97 dB daripada batas Shannon BER 10^{-6} dengan kerumitan pengekodan $O(1{,}0225\ N)$, $R = 0.928$ dan $N = 14364$ -- keputusan yang tidak ada teknik kajian lain boleh mencapainya.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
BLER	Block Error Rate
BPSK	Binary Phase-Shift Keying
DVB-S	Digital Video Broadcasting – Satellite
DVB-S2	2nd Standard of Digital Video Broadcasting
ECC	Error Correcting Code
eIRA	extended Irregular Repeat Accumulate
FEC	Forward Error Correction
FER	Frame Error Rate
LDPC	Low Density Parity Check
MAP	Maximum a posteriori Probability
PAN	Personal Area Network
QC	Quasi Cyclic
RA	Repeat Accumulate
RCS	Right Cyclic Shift
SNR	Signal to Noise Ratio
SPA	Sum Product Algorithm
UWB	Ultra Wideband
WER	Word Error Rate

CHAPTER 1

INTRODUCTION

1.1 LDPC Codes

There are several techniques that are used to help achieve reliable communication in digital communication systems. One of these methods is applying error control coding in the system. In the case of communication links where retransmission of information is not conveniently implemented and there is no reverse channel to inform an error has occurred, it is necessary to include enough redundancy (also called parity-bits) in the information bits that can be used for error detection and also correction so that errors during transmission will not cause an unacceptable loss. This technique of recovering transmission errors is termed as forward error correction (FEC) scheme with error correcting code (ECC).

The capability of detecting and correcting the errors in FEC scheme is mainly determined by the type of ECC applied in the system and that influences its error performance measured by bit error rate (BER) and block error rate (BLER). Among several ECC's in the literature, the low density parity check (LDPC) codes have become quite popular owing to the fact that they exhibit performance that is closest to the Shannon's limit. These codes are derived from a pre-designed parity check matrix that consists of extremely low density of ones.

LDPC codes were invented by Gallager in 1960's [1, 2]. Gallager's work received no attention for a long time except for the papers by Zyablov and Pinsker, Margulis [7] and Tanner [18]. LDPC codes were re-discovered and brought back in 1996 independently by

MacKay [4, 9], Neal [4], Wiberg [5], M.Sipser and D.A. Spielman [6]. Irregular LDPC codes have been able to achieve a threshold of just 0.0045 dB from Shannon limit in additive white Gaussian noise (AWGN) channel at BER of 10^{-6} with code rate ½, block length of 10^7 bits [8]. Up until now, the outstanding performance of LDPC codes has not been obtained for any real implementation and it is mostly reported in simulations.

A useful parameter that describes the amount of redundancy in an LDPC code is called code rate denoted by R. It is defined as the ratio of the number of information bits (K) to the value of code length (N). Thus, the length of its parity bits or redundancy is measured by ($N - K$). If we are using low code rate ($R < 0.5$) of LDPC codes, it means that these LDPC code have more redundancy but carry less information per code bit. Meanwhile, if we are using high code rate ($R > 0.5$) of LDPC codes up to a maximum of $R = 1$, these LDPC codes convey more information bits than their redundancy or less parity part is applied in those codes. Such applications of high code rate communication ($R \geq 0.8$) are found in magnetic recording and optical communication with high speed.

1.2 Motivation

A parity-check matrix (H) describes any block code completely. It is used to check if the received codeword is a legitimate codeword or not. Most commonly, the design of LDPC codes is derived from a specific parity check matrix (H) that consists of low density of ones -- hence the name and also the benefit of having very low decoding complexity. Since the code lengths are quite large, the property of low density of ones proves very useful in decoder implementation. In 1960's, Gallager proposed LDPC codes by randomly constructing H with fixed number of j ones in each column, fixed number of k ones in each row and zeros elsewhere, called (j, k) regular LDPC codes with the choice of $j = 3$ and $k = 4$ [1]. In his regular LDPC codes, there are at least ($j - 1$) rows of matrix H which are linearly dependent [1]. This condition means that matrix H is rank-deficient and the codes have a slightly higher code rate than the matrix H indicates.

Many researchers have contributed to the design of regular LDPC codes by random and structured construction. Structured construction of regular LDPC codes such as algebraic [10], [11], [12], [13], [18], [23], [26], [31], [35] and combinatorial [14], [15], [49], [51] also have some dependent rows of matrix *H*. Quasi-cyclic (QC) LDPC codes as one example of algebraic codes have at least $(j - 1)$ dependent rows in matrix *H* as stated in [11] and [24]. Therefore, generally in regular LDPC codes, there are likely to be some dependent rows in matrix *H*.

A generator matrix (*G*) that encodes the information bits is derived from this *H*. Generally, obtaining *G* from *H* involves Gauss elimination before encoding that may require computation of the order of $O(N^3)$ where *N* is the code length [11, 16]. In case, *H* has some rows that are linearly dependent, it is not possible to manipulate *H* to get *G*. Therefore, it is important to start with a full-rank matrix *H* in the design of regular LDPC codes. Many existing technique, either erase the linearly dependent rows in *H* or replace it with another independent row. In addition, regular LDPC codes are also required to avoid girth 4-cycles that may exist in matrix *H*. These cycles imply endless looping at the time of decoding. Since the parity part of matrix *G* is generally dense, encoding complexity of LDPC codes becomes prohibitively complex ~ $O(N^2)$. Moreover, the value of *N* in LDPC codes is large, around hundreds to thousands of bits or more.

QC LDPC codes solve high encoding complexity in regular LDPC codes since they have sufficient structure to allow simple encoding and they can be encoded with simple shift registers based on their generator polynomial in matrix *G* [12, 13, 14]. This type of encoding is only useful for a class of QC and cyclic codes. If QC LDPC codes are encoded by parity check matrix using inversion method, we need other process before encoding. Hanghang Qi and Norbert Goertz [52] show that if regular QC LDPC codes are encoded by Richardson-Urbanke method as one example of encoding by parity check matrix using inversion method, they require a pre-processing step like triangulation and check-rank of the order of $O(N^3)$.

There are two designs of LDPC codes that have been implemented in some standards, their code performances are not near the best performance of LDPC codes reported in the literature. These designs, called extended irregular repeat-accumulate (eIRA) and irregular quasi-cyclic (QC) designs, utilize base matrices of dual-diagonal parity part [17] and dual-diagonal parity part with single weight-3 column stated in [45,46].

Parity check matrix using dual-diagonal parity part, also known as semi-random parity check matrix, yields non-singular matrix with lower or upper triangular structure that is suitable to encode by its matrix H with no other process before it. Dual-diagonal parity part in matrix H gives non-singular parity part, linear encoding complexity and no pre-processing step before encoding. At the same time, dual-diagonal parity part padded with one weight-3 column, also recognized as irregular QC LDPC codes, has the same properties as dual- diagonal parity part alone.

Unfortunately, these advantages of parity-check matrix with dual-diagonal parity part and/or dual-diagonal parity part with single weight-3 column are only applicable for irregular LDPC codes precluding regular LDPC codes. Irregular LDPC codes are type of LDPC codes that do not satisfy the constant number of ones in each row and column. Therefore, there is scope for research into code-construction method for regular LDPC codes that has all the stated advantages of dual-diagonal parity part.

1.3 Problem Statement

The first class of (3, k) regular LDPC codes were invented by Gallager in 1960's. (3, k) regular LDPC codes have low decoding complexity and the value of their minimum distance (d_{min}) is lower bounded by 4.

It has been observed by MacKay and Davey [22] that (3, k) regular LDPC codes of code length around thousand bits ($N \leq 4000$) are not suitable for high code rate application

($R > 0.875$) as BER performance is not good. So, design of (3, k) regular LDPC codes that perform satisfactorily for above condition is still an open problem.

In general, (j, k) regular LDPC codes suffer from following design limitations such as:

- Rank-deficiency of matrix H
- High encoding complexity and
- Pre-processing step of encoding

Therefore, design methods of (3, k) regular codes that can overcome the above limitations are still open research problem.

1.4 Objective

The objectives of this thesis are given below:

- To develop novel code-construction for (3, k) regular LDPC codes :
  The first objective of this thesis is to develop a novel code-construction method for designing (3, k) regular LDPC codes with good error performance in high code rate ($R \geq 0.875$). This is accomplished by constructing suitable parity check matrix, H.

  The following characteristics are included in the proposed H:
  - No rank-deficiency of matrix H,
  - No singular parity part,
  - No pre-processing step of encoding in the order of $O(N^3)$ if it is encoded by parity check matrix (H) using inversion method,
  - No girth of length 4,
  - Low encoding complexity.

- To obtain the coded information bits from proposed H :

  The coded information bits that consist of information bits and designed parity bits are achieved by encoding using parity check matrix. Encoding by parity check matrix is chosen to simplify encoding and decoding process by using only matrix H without converting it into G. Moreover, this method can be applied to any LDPC codes like random or structured codes.

- To test the performance of coding using simulation :

  Finally, the performance of coded information bits is investigated in the simulation system in terms of bit error rate performance, closeness to Shannon limit and complexity analysis.

1.5 Scope of Thesis

Scope of the thesis is outlined as follows:

- Code-Construction :

  The proposed code-construction is aimed at developing binary LDPC codes. It makes use of sparse matrix function in MATLAB® for constructing (3, k) regular and irregular LDPC codes for code length $N < 15000$ bits with the characteristics mentioned in the objective.

- Simulation System :

  No hardware implementation is proposed for the proposed code-construction in this thesis. Instead, detailed study is conducted to investigate the performance using simulation developed by binary phase shift keying (BPSK) modulation, AWGN channel and MATLAB® 7.4.

- Performance Parameters
  - The performance of the proposed (3, k) regular LDPC codes in this thesis is evaluated in terms of the bit-error rate (BER) and block-error rate

(BLER) as a function of signal-to-noise ratio (SNR). Since the proposed code-construction can be utilized to construct irregular LDPC codes, we give one example of the proposed irregular LDPC code compared with other irregular LDPC codes in term of bit-error rate (BER).

The performance comparison of the proposed (3, *k*) regular with other (3, *k*) regular QC and (3, *k*) regular random LDPC codes is made by the BER in low and high code rate, and also the BLER in high code rate. Mean while, the performance comparison of irregular LDPC code is the BER performance.

- LDPC codes manage to reach the Shannon limit in an error rate of 10^{-6} at the lowest SNR as compared to all other error control codes. The performance of the proposed (3, *k*) regular codes are also measured in terms of this limit and compared with other regular LDPC codes in the literature particularly for high code rate ($R \geq 0.875$). The proposed (3, *k*) regular LDPC codes are also compared with theoretical un-coded BPSK and theoretical upper bound of an [*N*, *K*] binary block code with soft-decision decoding and BPSK modulation.

- The proposed (3, *k*) regular LDPC code is said to avoid several computational steps of pre-processing that most other popular regular LDPC codes must carry out, the performance of the proposed (3, *k*) regular LDPC code is also measured in terms of the computational load.

1.6 Methodology of Research

This research is conducted through two stages:

- Code-Construction Development

  Derived from the analysis of literature review, a new design of code-construction method is developed to obtain matrix *H* with characteristics mentioned in the

objective. The proposed code-construction is designed for constructing not only (3, k) regular LDPC codes but also irregular LDPC codes.

The proposed matrix H is checked for girth of length 4 when its information part is combined with parity part and in case this condition exists, the columns of expanded base matrices of information part are randomly cyclically shifted till it is devoid of any girth of length 4.

- Simulation System Development

  Simulation model is developed for the purpose of evaluating the coded information bits using BPSK modulation and AWGN channel in conjunction with the encoder-decoder of MATLAB® 7.4. The proposed parity check matrix, H, is provided to the MATLAB® 7.4 encoder.

  The encoder of MATLAB® encodes the binary data using matrix inversion of H as explained by Richardson-Urbanke in their work [16]. Similarly, the decoder of MATLAB® makes use of sum-product algorithm to decode the received codeword.

  In order to validate the proposed code-construction, the comparison of code performance is conducted by comparing the BER and BLER performance of the proposed codes.

  In addition, the performance of the proposed (3, k) regular LDPC code is also compared in terms of how close it performs to the Shannon limit particularly for high code rate ($R \geq 0.875$) LDPC codes. This is compared with those of regular QC and cyclic LDPC codes. The proposed (3, k) regular LDPC code is also compared with theoretical un-coded BPSK and theoretical upper bound on the BER of an [N, K] binary block code with soft-decision decoding and BPSK modulation.

Since encoding is the strength of the proposed code-construction, complexity analysis is made based on comparison of pre-processing step of encoding followed by encoding complexity with (3, 5) regular QC LDPC codes. This section is ended by computational time of actual encoding step in the proposed codes that includes a description of encoding complexity in the proposed codes and also compares the time of encoding between the proposed (3, 6) regular with (3, 6) regular QC LDPC code. The complexity analysis for irregular LDPC codes has been excluded from the scope.

1.7 Thesis Organization

This thesis is organized as follows: Chapter 2 gives background theory and literature review of this work that gives some related concepts of LDPC codes, encoding of LDPC codes and some related works for this thesis. By understanding some related concepts of LDPC codes, one is brought to an appreciation of some concepts used in this thesis such as linear block code, the minimum distance of code (d_{min}), generator matrix (G) and parity check matrix (H), syndrome error and a description of LDPC codes which covers explanation of Tanner graph including a cycle and girth properties, concept of regular-irregular LDPC codes, quasi cyclic (QC) LDPC codes, repeat accumulate (RA) LDPC codes, decoding of LDPC codes and code performance utilized in this thesis. After introducing some background of this work, encoding issues of LDPC codes is reported that presents encoding by generator matrix and encoding by parity check matrix. Some relevant works in this thesis as the literature review are introduced to give an overview of this work that includes code-construction method of regular, irregular and implemented irregular LDPC codes.

Chapter 3 shows the steps of designing the proposed matrix H in three stages of code-construction. Calculation of code rate (R) in the proposed LDPC codes is given after the explanation of constructing the proposed matrix H. After introducing calculation of R,

encoding step utilized in the proposed LDPC codes is given, followed by a description of encoding complexity based on Richardson-Urbanke method.

Chapter 4 presents code performance, simulation model and error performance conducted in this thesis to validate the proposed LDPC.

Chapter 5 presents some results, analysis and discussion of the proposed codes in terms of BER, BLER, Shannon limit performance and complexity analysis. Code performance of (3, k) regular LDPC codes presents BER and BLER performance of the proposed codes and compares them with other regular LDPC codes such as QC and regular random at both high and low code rate. The limiting performance is also obtained and compared with the Shannon limit and those of a selected few codes at high code rate ($R \geq 0.875$). Moreover, the proposed (3, k) regular LDPC codes are also compared with theoretical upper bound of an [N, K] binary block code with soft-decision decoding and BPSK modulation. Code performance of irregular LDPC codes presents BER performance of the proposed irregular LDPC code compared with other irregular LDPC codes. Discussion of complexity analysis begins by exploring pre-processing step in the proposed (3, k) regular LDPC codes based on Richardson-Urbanke method. The complexity analysis for irregular LDPC codes has been excluded from the scope. The discussion of complexity analysis is continued for comparison of pre-processing step of encoding followed by encoding complexity with (3, 5) regular QC LDPC codes. This section is ended by computational time of actual encoding step in the proposed codes that includes comparison with regular QC LDPC and a description of encoding complexity in the proposed codes.

The conclusion of this work and recommendations for future work of this thesis are given in the last chapter.

Appendix of this thesis gives feasible values of parameter *base* and expansion factor L with given choice of deterministic right cyclic shift (RCS) that give matrix H with no rank-deficiency and girth of 4-cycles.

CHAPTER 2

BACKGROUND THEORY AND LITERATURE REVIEW

2.1 Introduction

LDPC codes nowadays are in vogue and are often used to have a reliable communication with low power consumption. In order to employ LDPC codes as one of Shannon limit approaching codes, one needs to utilize some patterns of redundancy and to fulfill some requirements of LDPC codes like the need of designing sparse code.

In this chapter, we present some background material related to LDPC codes and some crucial issues in the design of regular LDPC codes such as the rank-deficiency of matrix *H*, high encoding complexity and pre-processing before encoding. The focus of this chapter is to give a description about some background concepts relevant to this thesis and present the related literature.

Section 2.2 will discuss the background concepts used in this work, starting from basic concept of generator matrix and parity check matrix to the code performance utilized in this thesis. Encoding of LDPC codes is explained in section 2.3 that covers encoding by generator matrix and encoding by parity check matrix. Section 2.4 focuses on some relevant works of code-construction method for regular, irregular and implemented irregular LDPC codes. The last section, section 2.5, summarizes this chapter.

2.2 Related Concepts

This section explores some introductory concepts used in this thesis for better understanding of the work presented herein. Since LDPC codes are a special class of linear block codes, there are some notations that are common to both codes such as generator matrix (G), parity check matrix (H), minimum distance (d_{min}), encoding by a generator matrix (G) and syndrome error. In order to get more details about LDPC codes and linear block codes, the reader is referred to read reference [3], [20].

2.2.1 Linear Block Codes

Linear block codes, represented by (n, k) notation, are a subclass of block codes that having k information bits, (n - k) redundant or parity bits and n bits as code length. Transformation of k information bits into a longer block of n codeword bits is constructed with linear mapping transformation.

One example of linear block codes is Hamming codes, the first code in error correcting code (ECC) invented by Richard Hamming in 1950 [21]. In our discussion of linear block codes, we restrict ourselves to binary codes that consist of two elements (0 and 1). A binary block code is said to be a linear code if and only if modulo-2 sum of any two codeword is also a codeword (c) known as closure property.

Linear block codes have special properties as listed below:

1. All zero codeword (0 0 0 0 … 0) is always a codeword (c).
2. The distance between any two codeword is the same as the weight of the sum of both codeword.

$$d(\boldsymbol{c}_i, \boldsymbol{c}_j) = w(\boldsymbol{c}_i + \boldsymbol{c}_j) \tag{2.1}$$

   where distance is the number of different places in two codeword.

The Hamming weight known as weight of a codeword (c) is defined by the number of non-zero elements in codeword (c) denoted by $w(c)$.

3. The minimum distance (d_{min}) of linear block code is equal to the weight of the smallest weight (w_{min}) of any non-zero codeword excluding the-all zero codeword.

$$d_{min} = w_{min} \tag{2.2}$$

where w_{min} is the smallest weight or the number of non-zero elements in a codeword.

A desirable property of linear block codes is systematic structure of codeword as given in Figure 2.1 also known by systematic linear block codes [20]. Systematic (n, k) linear block codes are divided into information part and parity check part. Information part consists of k information bits and parity check part consists of (n - k) redundant bits.

In Figure 2.1, parity part or redundancy part bits are placed at the beginning of a codeword while information bits are placed at the end of a codeword, however this can be done the other way round. The choice of position does not modify the properties of a given code, although there will be some different forms in mathematical expressions related to the code itself.

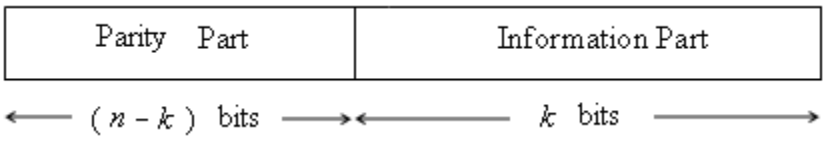

Figure 2.1: Systematic structure of a codeword.

2.2.2 The Minimum Distance (d_{min})

As already mentioned in section 2.2.1, the minimum distance ($d_{\min}$) of LDPC codes is also equal to the weight of the smallest weight ($w_{\min}$) of any non-zero codeword excluding the-all zero codeword.

The minimum distance (d_{min}) is an essential parameter that will determine the capability of error detection and correction in LDPC codes. The minimum distance (d_{min}) relates to other parameter such as the Hamming weight, the weight of a codeword and the Hamming distance, the distance between two codewords c_1 and c_2, denoted by $d(c_1, c_2)$.

Linear block codes with minimum distance (d_{min}) is able for correcting all error patterns of weight t or smaller weight than t. Parameter t is recognized by the random error correcting capability of a linear block code which is also utilized in LDPC codes [3].

$$t = \lfloor (d_{min} - 1)/2 \rfloor \tag{2.3}$$

where $\lfloor \rfloor$ means the largest integer number not greater than $(d_{min} - 1)/2$.

2.2.3 Generator Matrix (G) and Parity Check Matrix (H)

Generator matrix (G) is a linear independent basis matrix for any codeword in the vector space. A desirable property of a $(k \times n)$ matrix G with k linearly independent rows and n columns is a systematic structure having two part of k information bits as information part and $(n - k)$ redundant bits as parity check part. Therefore, a $(k \times n)$ matrix G for systematic structure is represented below:

$$G = [P \mid I_k] \tag{2.4}$$

$$G = \begin{pmatrix} p_{11} & p_{12} & \dots & p_{1,(n-k)} & 1 & 0 & \dots & 0 \\ p_{21} & p_{22} & \dots & p_{2,(n-k)} & 0 & 1 & \dots & 0 \\ . & . & & . & & . & & \\ . & . & & . & & . & & \\ . & . & & . & & . & & \\ p_{k1} & p_{k2} & \dots & p_{k,(n-k)} & 0 & 0 & \dots & 1 \end{pmatrix} \quad (2.5)$$

where P is a $k \times (n - k)$ parity part of matrix G with p_{ij} = (0 or 1) and I_k is an $(k \times k)$ identity matrix with ones on the main diagonal and zeros elsewhere.

Parity check matrix (H) is a form of matrix utilized at decoding stage to decode the received codeword and to check whether an error has occurred or not in the transmitted data stream. Relation between matrix G and matrix H is given by equation (2.6) [3]:

$$G\,H^T = 0. \quad (2.6)$$

Equation (2.6) means that each matrix G of size $(k \times n)$ will have an $(n - k) \times n$ matrix H, such that the rows of matrix G are orthogonal to the rows of matrix H. Matrix H^T is an $n \times (n - k)$ matrix and 0 is a $k \times (n - k)$ matrix with all-zeros elements. Therefore, matrix H for systematic structure will be:

$$H = [I_{n-k} \,|P^T]. \quad (2.7)$$

$$H = \begin{pmatrix} 1 & 0 & \dots & 0 & p_{11} & p_{12} & \dots & p_{1,k} \\ 0 & 1 & \dots & 0 & p_{21} & p_{22} & \dots & p_{2,k} \\ . & . & & . & . & & . & \\ . & . & & . & & . & & \\ . & . & & . & & . & & \\ 0 & 0 & \dots & 1 & p_{(n-k),1} & p_{(n-k),2} & \dots & p_{(n-k),k} \end{pmatrix} \quad (2.8)$$

where P^T is a $(n - k) \times k$ matrix as the transpose of matrix P and I_{n-k} is an $(n-k) \times (n-k)$ identity matrix with ones on the main diagonal and zeros elsewhere. Matrix H is used in decoding to check whether a codeword (c) is a valid codeword using equation (2.9). This is represented as the condition:

$$\boldsymbol{c}H^T = 0 \tag{2.9}$$

2.2.4 Syndrome Error

Syndrome is the result of parity check performed on received codeword (r). Equation of syndrome (S) is denoted by the following equation:

$$S = \boldsymbol{r}\, H^T \tag{2.10}$$

Whenever a non-zero error syndrome (S) is obtained, the decoder knows that at least one error has occurred. If the received codeword (r) is a valid codeword and there are no received errors then the value of syndrome (S) will be all zero. This condition is described in the equation as follows:

$$\boldsymbol{r}\, H^T = 0 \tag{2.11}$$

There is a possibility that received codeword (r) contains errors but the value of syndrome (S) is zero. This condition occurs when errors (e) convert received codeword (r) to another codeword that gives zero value of syndrome (S). This kind of error pattern is called as undetectable error pattern and causes an incorrect decoding.

Since received codeword $r = c + e$ and c a valid codeword with $\boldsymbol{c}\, H^T = 0$, it is possible to measure error syndrome (S) of r by [3]:

$$S = \boldsymbol{r}\, H^T = (\boldsymbol{c} + \boldsymbol{e})\, H^T = \boldsymbol{c}H^T + \boldsymbol{e}\, H^T = \boldsymbol{e}\, H^T \tag{2.12}$$

If there is no error ($e = 0$), $r = c$ would give $\boldsymbol{r}\ H^T = \boldsymbol{c}\ H^T = 0$. In fact, syndrome ($S$) is linear combination of error pattern (e) and depends on it not on transmitted codeword (c) [20].

2.2.5 Tanner Graph

The Tanner graph, a bipartite graph, is a graphical representation of matrix H. It is commonly represented by a diagram in which nodes or vertices are represented by points. Two nodes are connected by an edge shown as a line joining the end nodes.

There are two classes of nodes in Tanner graph, namely, variable nodes or bit nodes and check nodes. Nodes corresponding to the columns are recognized by variable nodes, while nodes corresponding to the rows are recognized by check nodes. Degree of variable node is equal to the number of j_n ones in the n^{th} column of matrix H while degree of check node is equal to the number of k_m ones in the m^{th} row of matrix H.

It is desirable to have high degree of check nodes. Variable nodes with small degree (one or two) do not get enough information to make good estimate, it is also desirable to have high degree of variable nodes.

Figure 2.2 shows a Tanner graph for an LDPC code and its matrix H with the size of 4×8. In this Tanner graph there exist four numbers of check nodes (c_i) and eight numbers of variable nodes (v_i).

Each variable node (v_i) in Figure 2.2 is connected to two check nodes that implies each variable node has degree of two while each check node (c_i) is connected to four variable nodes that means each check node has degree of four. Since each column and each row have constant number of ones in its matrix H, this LDPC code is recognized by (2, 4) regular LDPC code.

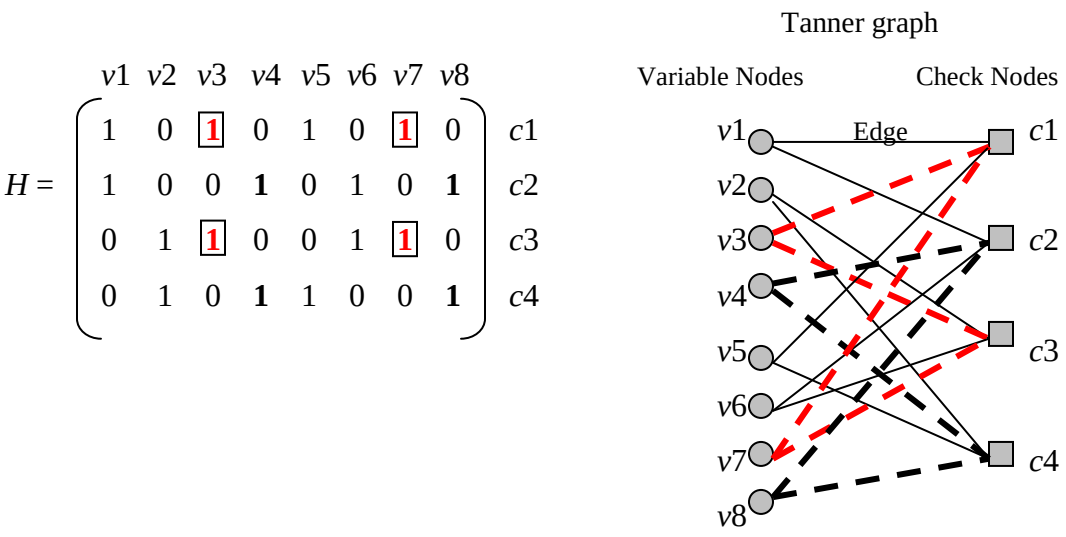

Figure 2.2: Tanner graph of (2, 4) regular LDPC code and its matrix *H*.

A cycle of length *n* in Tanner graph implies that there is a path comprised of *n* edges which returns back to its starting point. A cycle is called even if its length is even and similarly, a cycle is odd if its length is odd. Tanner graph in Figure 2.2 has two cycles of length four that is shown by dashed line, described by bold ones in its matrix *H* and comprised of nodes (*v*3, *c*1), (*v*7, *c*3) and (*v*4, *c*2), (*v*8, *c*4).

A girth of Tanner graph is the length of the shortest cycle in a graph. Obviously, simple bipartite graph or Tanner graph does not have odd cycles. Therefore, the shortest possible cycle exists in a Tanner graph is a cycle of length four or a length-4 cycle [25]. Actually, there is an impact of girth-4 cycle in decoding algorithm.

If there are too many girth-4 cycles in a graph, it will slow down the decoding convergence of iterative sum product algorithm (SPA) or it will never reach the decoding convergence. The reason why sum product algorithm (SPA) avoids girth-4 cycle is that SPA will be a correct algorithm if the graph is tree-like. Girth-4 cycle yields a low-weight codeword that influences the capability of detecting and correcting error and degrades the error performance by having an error floor.

If Tanner graph is cycle free, information bits are sent independently and iterative decoding is optimal. A graph without cycles is said to be acyclic and called a tree. Figure 2.3 describes acyclic graph [3]. In [20], it is proven that codes with cycle free Tanner graph have very poor minimum distance (d_{min}) and very low rates. However, girth does not seem the only parameter that has impact on the code performance.

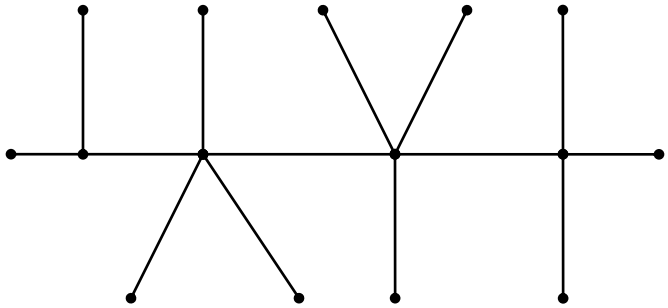

Figure 2.3: An acyclic graph [3].

2.2.6 LDPC Codes

Low density parity check (LDPC) codes are designed based on their parity check matrix *H* that has sparse density of ones. Sparse density means proportion of non-zero entries in matrix *H* is very small compared with its zero entries. The extremely low density of ones in matrix *H* lends to very low decoding complexity.

Since matrix *H* plays a major role in designing an LDPC code, a code-construction in LDPC codes begins with techniques to build its matrix *H*. Based on this *H*, we can classify types of LDPC codes which are regular and irregular LDPC codes. If matrix *H* has fixed number of *j* ones in each column and fixed number of *k* ones in each row, this matrix *H* is recognized as (*j*, *k*) regular LDPC codes. Matrix *H* that does not satisfy this condition is recognized as irregular LDPC codes.

The requirement for constructing an LDPC code is two-fold as follows [3], [13], [20]:

1. The density (r) of its parity check matrix (H) must be small or have low density of ones.

   Density of ones in parity check matrix (H) is defined as the ratio of the number of ones to the total number of entries in it. Density is denoted by $r = j/M = k/N$.

2. No two columns or two rows in parity check matrix (H) can have more than one non-zero entry in common.
   This property avoids girth with cycle of length four in parity check matrix (H) and hence has girth at least 6 [13].

Code rate (R) of an LDPC code is given by ratio of information bits (K) to code length (N). Let C be an LDPC code specified by matrix H of size M by N. The number of information bits, denoted by K, is $N – M$. The length of redundant bits or parity bits added in the code is ($N – K$) bits or M bits. Therefore the value of R for an LDPC code is defined by:

$$R = K/N \tag{2.13}$$

If we change the value of K in equation (2.13) with $N – M$ then:

$$R = K/N = (N – M) \div N = 1 – M/N \tag{2.14}$$

2.2.7 *Quasi Cyclic (QC) LDPC Codes*

A quasi cyclic is a linear block code for which cyclically shifting a codeword a fixed number shift of symbol positions either to the right or to the left results in another codeword. It is clear that for one shift of symbol positions, a quasi-cyclic code is a cyclic code [3]. The structure of a quasi cyclic (QC) LDPC code can be viewed from its parity check matrix (H) in circulant form.

Circulant matrix itself is defined as a square matrix in which each row is a cyclic shift (one place to the right) of the row above it, with the first row being a cyclic shift of the last row. Each column is a downward cyclic shift of the column on its left, with the first column is a downward cyclic shift of the last column. A circulant is completely characterized by its first row (or first column), which called the generator of the circulant.

An example of 7×7 circulant matrix $H_{j,k}$ with $w = 3$ is given below:

$$H_{j,k} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

The row and column weights of a circulant matrix are the same, say w, in other words the circulant has weight w. If a circulant matrix has $w = 1$, then the circulant is a permutation matrix, called by circulant permutation matrix [12].

A QC LDPC code is characterized by matrix H that consists of small square blocks which are zeros matrix and circulant permutation matrices. Matrix H of a QC LDPC code can be written as:

$$H = \begin{pmatrix} H_{1,1} & H_{1,2} & \dots & H_{1,k} \\ H_{2,1} & H_{2,2} & \dots & H_{2,k} \\ H_{3,1} & H_{3,2} & \dots & H_{3,k} \\ \dots & \dots & & \dots \\ H_{j,1} & H_{j,2} & \dots & H_{j,k} \end{pmatrix} \tag{2.15}$$

Let sub-matrix $H_{j,k} = H_x$ be a $p \times p$ circulant permutation matrix which shifts the identity matrix I to the right or to the left by x position for any integer x, $0 \leq x < p$ given by:

$$H_x = \begin{pmatrix} 0 & 1 & 0 & & 0 \\ 0 & 0 & 1 & \dots & 0 \\ . & . & . & & . \\ 0 & 0 & 0 & \dots & 1 \\ 1 & 0 & 0 & & 0 \end{pmatrix}$$

Figure 2.4: Example of circulant permutation matrix.

The QC LDPC code may be regular or irregular depending on the choice of x of H_x Figure 2.4. When matrix H has no blocks corresponding to zero matrices, matrix H represents a (j, k) regular LDPC code with column weight j and row weight k. The resulting binary matrix H is of size $j_p \times k_p$.

Cyclic structure of matrix H yields some dependant rows in matrix H [12]. In fact, there are at least (j −1) dependent rows in matrix H. Due to linear dependence among the rows of matrix H, the code rate (R) may be greater than ($1-j/k$) or denoted by $R \geq (1-j/k)$ [l1].

In order to construct QC LDPC codes, we need to follow the requirement for constructing an LDPC code described in section 2.2.4.

2.2.8 Repeat Accumulate (RA) LDPC Codes

Another type of codes that has weight-2 columns is called repeat accumulate (RA) codes. The advantage of systematic RA codes is that its encoding is linear.

An $(n - k) \times n$ matrix H of RA code has two parts H_1 and H_2 denoted by $H = [H_1, H_2]$ where H_1 is information part and H_2 is a $(n - k) \times (n - k)$ matrix of parity part in the form of Figure 2.5. Since parity part of RA codes is in the form of Figure 2.5, RA codes have a lower triangular form already built into matrix H during the code design. The built in lower triangular form is also called dual diagonal code construction.

RA codes also consist of regular and irregular repeat accumulate (RA). The distinction between regular and irregular RA codes is defined by the composition of ones in their information part or matrix H_1 while the construction of their parity part or matrix H_2 for both regular and irregular RA codes are the same.

$$H_2 = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 & 0 & 0 \\ 1 & 1 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & 1 & \dots & 0 & 0 & 0 \\ & . & & & & . & \\ & . & & & & . & \\ & . & & & & . & \\ 0 & 0 & 0 & \dots & 1 & 0 & 0 \\ 0 & 0 & 0 & \dots & 1 & 1 & 0 \\ 0 & 0 & 0 & \dots & 0 & 1 & 1 \end{pmatrix}$$

Figure 2.5: Example of dual diagonal parity part.

RA codes are called (q, a) regular RA codes if all the rows of matrix H_1 have the same weight of a, and all the columns of matrix H_1 have the same weight of q. An example of regular RA code is given below with the size of 6×10. Information part or matrix H_1 of this example has a regular composition of ones that is three ones in each column and two

ones in each row. Therefore, we call this matrix as (3, 2) regular RA code with code length of 10 bits. The value of R for this (3, 2) regular RA code is 4/10 = 2/5.

$$H = [H_1 \mid H_2]$$

$$H = \left(\begin{array}{cccc|cccccc} 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{array}\right)$$

Irregular RA codes have irregular columns or rows or both columns and rows have irregular weight distribution in their information part or matrix H_1 with the size of $(n - k) \times p$ with $p > k$. Since information part or matrix H_1 has the size of $(n - k) \times k$, this type of irregular RA code belongs to extended irregular RA (eIRA) codes.

An example of a 7×10 irregular RA code with its information part or matrix H_1 has an irregular composition of ones in each column and regular composition of ones in each row that is only one value in each row is described below. The value of R for this irregular RA code is 3/10 = 0.3 with code length of 10 bits.

$$H = [H_1|H_2]$$

$$H = \left(\begin{array}{ccc|cccccc} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{array}\right) \qquad (2.16)$$

If density of ones in RA codes is sparse and having no girth-4 cycle, then these RA codes belong to LDPC codes known as RA LDPC codes. Since RA codes consist of regular and irregular, RA LDPC codes are also divided into regular and irregular RA LDPC codes.

Regular RA LDPC codes belong to irregular LDPC codes in the sense of general description for (j, k) regular LDPC codes since matrix H_2 in regular RA LDPC codes has all columns of weight-2 except one column having weight-1. Even though we call these codes as regular, it denotes irregular code construction in a general classification of LDPC codes.

2.2.9 Sum Product Decoding Algorithm

Sum product algorithm (SPA) is an iterative decoding algorithm. It is also known as belief propagation algorithm. Sum product algorithm is a decision algorithm that accepts the probability of each received bit as input. The aim of sum product algorithm is to compute maximum a posteriori probability (MAP) for each codeword. The structure of SPA directly matches the Tanner graph since decoding messages are iteratively computed for all the variable nodes, check nodes and exchanged through the edges between the neighboring nodes.

The input bit probabilities are called a-priori probabilities for the received bits because they were known in advance before running LDPC decoder. The bit probabilities returned by decoder are called a-posteriori probabilities. A-posteriori probabilities can only be established after many events like symbol transmission and receptions have been completed. These probabilities are expressed as log-likelihood ratios. The benefit of logarithmic representation is that log-likelihood ratios need only be added when probabilities need to be multiplied. Therefore, it reduces the complexity.

Maximum a posteriori probability (MAP) of each codeword bit, $P_i = P\{c_i = 1| N\}$, is the probability that the-ith codeword bit is a 1 conditioned on the event N that all parity check constraints are satisfied. The extra information about bit received from parity checks is called extrinsic information for bit i.

Sum product algorithm iteratively computes an approximation of MAP value for each code bit. However, a posteriori probabilities returned by sum product decoder are only exact MAP probabilities if the Tanner graph does not contain too many short cycles.

The computed MAP values, at the end of decoding iteration, are used as input for the next iteration. Decoding iteration process continues until a certain stopping condition (or criteria) is met.

Computational complexity and decoding delay (decoding time) of sum product algorithm (SPA) increases when the number of decoding iteration increases. Unfortunately, long decoding delay is not desirable in high speed communication and data storage system.

There are several parameters that influence sum product algorithm (SPA) and relate to error performance [25]:

1. Girth value of its Tanner graph.
2. The minimum distance (d_{min}).
3. Column weight and row weight of its parity check matrix (H).

2.3 Encoding of LDPC Codes

Encoding process takes information of k bits and assigns redundancy of (N - k) bits as additional parity to be used to detect and correct the errors at the decoder that gives a codeword (c) its length N bits. Usually, the process of encoding is determined in the encoder.

Generally, encoding of LDPC codes utilizes either the generator matrix (G) or the parity check matrix (H). Encoding by matrix G is a straight forward encoding of LDPC codes utilized in linear block codes while encoding by matrix H is applied to make use the sparseness of matrix H in LDPC codes.

2.3.1 Encoding by Generator Matrix (G)

Encoding by matrix *G* is done by multiplying information bits (*i*) of *k* bits and matrix *G* of (*k* × *N*) matrix to produce a codeword (*c*) with length of *N* bits given in equation (2.17) until equation (2.19).

Suppose a (1 × *k*) matrix of information bits (*i*) has length of *k* bits and a (*k* × *N*) generator matrix (*G*). The relationship of encoding is stated below:

$$G = \begin{pmatrix} g_1 \\ g_2 \\ . \\ . \\ . \\ g_k \end{pmatrix} = \begin{pmatrix} g_{11} & g_{12} & g_{13} & \cdots & g_{1N} \\ g_{21} & g_{22} & g_{23} & \cdots & g_{2N} \\ & & . & & \\ & & . & & \\ & & . & & \\ g_{k1} & g_{k2} & g_{k3} & \cdots & g_{kN} \end{pmatrix} \tag{2.17}$$

$$c = i.G \tag{2.18}$$

$$c = i_1 g_1 + i_2 g_2 + \ldots + i_k g_k \tag{2.19}$$

Information bits (*i*) are added by certain pattern of redundancy to become a codeword (*c*) by taking the product of information bits (*i*) with every column in matrix *G*. Each element in *i* is multiplied by the corresponding element in column of matrix *G* then summed using modulo-2 addition.

Since code-construction in LDPC codes is started with its matrix *H*, encoding by matrix *G* has the purpose of converting matrix *H* into matrix *G* that can be achieved into two ways: achieving systematic matrix *G* and Mac-Kay method.

2.3.1.1 Achieving Systematic Structure of Matrix G

In section 2.2.2, finding a systematic structure of matrix *G* from its matrix *H* is easy if matrix *H* has a systematic structure. Therefore, using systematic structure of matrix *H,* encoding process will be easier to be done.

Unfortunately, systematic structure of matrix *H* has a disadvantage. Generally, systematic matrix $H = [I_{n-k} \mid P^T]$ has parity part (*P*) that is a dense matrix. This condition will yield a many girths with 4–cycles that will degrade the decoding performance [27]. As mentioned in section 2.2.3, girth-4 cycles in a graph will slow down the decoding convergence or it never reaches the decoding convergence.

In order to encode information bits, one should find systematic structure of matrix *G* in equation (2.5) by performing Gauss elimination on matrix *H*. Achieving structured matrix *G* by Gauss elimination can achieve complexity of the order $O(N^3)$ [16]. Moreover, since the parity part of matrix *G* is generally a dense matrix, it leads to encoding complexity of the order of $O(N^2)$ where *N* is code length of LDPC code [27].

Since the value of *N* is large in LDPC codes, around hundreds to thousands of bits or more, the encoder will become prohibitively complex with $O(N^2)$. That is why encoding complexity is one of the crucial issues in LDPC codes.

2.3.1.2 MacKay Method

There is another way of finding a matrix *G* without altering matrix *H* proposed by [9]. Suppose an $M \times N$ matrix *H* having *K* information bits in the form of:

$$H = [C_1 | C_2] \tag{2.20}$$

where C_1 is an ($M \times K$) matrix and C_2 is an ($M \times M$) matrix. The difference is that matrix C_2 should be a non-singular matrix or invertible matrix and matrix C_1 can be any random matrix. Matrix G is computed by solving $H\,G^T = 0$ and yields:

$$G^T = [I_K \mid (C_2^{-1}\, C_1)^T] \tag{2.21}$$

This method also has a drawback in achieving a non-singular of matrix C_2 which is not a trivial task. Encoding complexity of MacKay method is still of the order of $O(N^2)$ since generally matrix $(C_2^{-1}\, C_1)^T$ is a dense matrix.

2.3.2 Encoding by Parity Check Matrix (H)

Encoding by parity check matrix is utilized to take advantage of the sparseness of matrix H using matrix inversion method without constructing its matrix G. The benefit of this type of encoding method is that it can be used for random and structured LDPC codes. Encoding of LDPC codes via matrix H is done in accordance with $c\,H^T = 0$.

There are two approaches used in encoding by matrix H. The first one is encoding by semi random parity check matrix introduced by Li Ping [17] and the other is Richardson-Urbanke method. Encoding by semi random parity check matrix is given in section 2.3.2.1 while Richardson-Urbanke method is explained in section 2.3.2.2 that utilizes an approximate lower triangular form of parity part.

2.3.2.1 Encoding by Semi Random Parity Check Matrix

Semi random parity check matrix divides its matrix H into information part and parity part. This approach is introduced in [17] and is called semi random technique since the information part of matrix H is created randomly and the parity part has deterministic construction of dual diagonal matrix. The choice of dual diagonal matrix of parity part will be used in encoding process from equation (2.22) to (2.24).

Let matrix $H = [H_i \mid H_p]$ be a $(N-K) \times N$ matrix with K information. H_i is a $(N-K) \times K$ matrix of information part and H_p is a $(N-K) \times (N-K)$ matrix of parity part. Suppose a codeword (c) has a structure $c = [m \mid p]$ where m is the information bit with length of K bits and p is parity bit with length of $(N-K)$ bits. Applying equation $H\,c^T = 0$ yields:

$$[H_i \mid H_p] \,.\, [m \mid p]^T = 0 \tag{2.22}$$

$$H_i\, m^T \oplus H_p\, p^T = 0 \tag{2.23}$$

$$p^T = H_p^{-1} H_i\, m^T \tag{2.24}$$

where $\oplus$ represents operation of modulo-2 addition.

Note that matrix H_p is a square non-singular matrix with size $(N-K) \times (N-K)$ and matrix H_i is $(N-K) \times K$. After getting the value of p^T, we could find codeword (c) by inserting the value of p into $c = [m \mid p]$. Ensuring matrix H_p to be non-singular is an important task to be carried out since it is used to find its parity bit p by solving equation (2.24) and complete the encoding process. Since the construction of matrix H_p is dual diagonal, matrix H_p is always non-singular and in a full-rank condition. This gives encoding complexity of the order of $O(N)$.

The details of how linear encoding complexity results for dual diagonal parity part code construction are given below. Derived from equation (2.23), the equivalence of LDPC code using dual diagonal construction for matrix H_p is described below:

$$z = H_i\, m^T = H_p\, p^T \tag{2.25}$$

To find parity bit p in equation (2.24), one needs to solve $H_p\, p^T = z = H_i\, m^T$. The value of z is computed by determining $z = H_i\, m^T$ then the redundant bit is computed through back substitution with $p_0 = z_0$ as described below:

$$
\left.
\begin{aligned}
p_0 &= z_0 \\
p_1 &= z_1 + p_0 \\
p_2 &= z_2 + p_1 \\
p_3 &= z_3 + p_2 \\
&\cdot \quad\quad \cdot \\
&\cdot \quad\quad \cdot \\
&\cdot \quad\quad \cdot \\
p_i &= z_i + p_{i-1} \text{ with } i = 1, 2, \ldots, (N-K)-1
\end{aligned}
\right\} \qquad (2.26)
$$

Since all computation for encoding are on binary values, XOR and AND gates are used instead of adders and multipliers that are more expensive. According to equation (2.26), back substitution requires $(N-K)-1$ XOR's.

The total number of ones in matrix H_p is equal to $((N-K)\times j)-1$ where $j = 2$ is the highest degree in each column. Since the value of $(N-K)$ is equal to $N\times(1-R)$, the overall computational complexity amounts to $N\times(1-R)\times j-1 \approx O(N)$ described in [25].

2.3.2.2 Richardson – Urbanke Method

Encoding procedure of Richardson-Urbanke method is accomplished in two steps, a pre-processing step followed by actual encoding step [16]. Pre-processing step is an offline calculation and actual encoding of Richardson-Urbanke method divides its matrix H into information and parity part that are further divided into another six sub-matrices A, B, C, D, E and T for encoding process.

Pre-processing step of encoding in Richardson-Urbanke method is accomplished in two processes, triangulation and check-rank process. Triangulation is done to get an approximate triangular form of matrix H and check-rank process is intended to achieve non-singular condition for matrix $(ET^{-1}B + D)$. An example of matrix H with an approximate triangular form is given by Figure 2.6.

In the pre-processing step, an assumption is used that matrix H is non-singular with the size of $M \times N$ matrix and already in full-rank condition. Since it is assumed that matrix H is non-singular matrix, an approximate lower triangular form of matrix H is achieved by performing row and column permutation only.

Matrix H consists of information part (H_i) and parity part (H_p). Then, parity part of matrix H is divided into sub matrices that are going to be used in mathematical calculation of actual encoding step as given in equation (2.27) and indicated in Figure 2.6.

$$H = \left[\begin{array}{c|cc} A & B & T \\ C & D & E \end{array}\right] \tag{2.27}$$

According to equation (2.27), information part (H_i) of matrix H is divided into two sub matrices which are matrix A and C. Parity part (H_p) of matrix H is, however, divided into four sub matrices that are matrix B, D, E and matrix T. Matrix T has a lower triangular form with ones along diagonal.

All of these sub matrices are sparse and each matrix contains at most $O(N)$ elements. Based on Figure 2.6, the size of matrix A is $(M - g) \times (N - M)$, B is $(M - g) \times g$, T is $(M - g) \times (M - g)$, C is $g \times (N - M)$, D is $g \times g$ and E is $g \times (M - g)$.

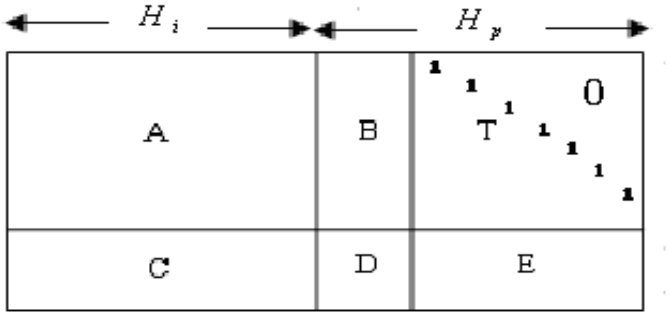

Figure 2.6: Parity check matrix (H).

Before applying encoding step, we need to perform check- rank process in pre-processing step of encoding by multiplying matrix H from the left in accordance with equation (2.28) and (2.29) that can be done by Gaussian elimination to effectively perform the pre-multiplication. This process includes clearing matrix E and checking whether matrix ($ET^{-1}B + D$) or matrix Φ is non-singular.

$$\begin{bmatrix} I & 0 \\ -ET^{-1} & I \end{bmatrix} \times H \tag{2.28}$$

$$\begin{bmatrix} I & 0 \\ -ET^{-1} & I \end{bmatrix} \times \begin{bmatrix} A & B & T \\ C & D & E \end{bmatrix} \tag{2.29}$$

A full-rank condition of matrix ($ET^{-1}B + D$) is achieved if all rows of matrix ($ET^{-1}B + D$) are independent to each other. If matrix Φ is singular then column permutation is performed so as to remove this singularity in matrix H. As such, removing singularity is not a trivial task and needs more time to go through all the process in matrix H.

Matrix H will be:

$$\begin{bmatrix} A & B & T \\ -ET^{-1}A + C & -ET^{-1}B + D & 0 \end{bmatrix} \tag{2.30}$$

Let $c = [m, p_1, p_2]$ be a codeword where m is the information bit and its parity part is divided into p_1 with length g and p_2 with length $M - g$. Actual encoding step starts with equation (2.31) as follows:

$$\begin{bmatrix} A & B & T \\ -ET^{-1}A + C & -ET^{-1}B + D & 0 \end{bmatrix} \times \begin{bmatrix} m, p_1, p_2 \end{bmatrix}^T = 0 \tag{2.31}$$

The result from equation (2.31) gives several equations to define the value of p_1^T and p_2^T as stated in equation (2.32) until equation (2.36) given below.

$$Am^T + Bp_1^T + Tp_2^T = 0 \tag{2.32}$$

$$(-ET^{-1}A + C) \times m^T + (-ET^{-1}B + D) \times p_1^T = 0 \tag{2.33}$$

$$\Phi = -ET^{-1}B + D \tag{2.34}$$

$$p_1^T = -\Phi^{-1}(-ET^{-1}A + C) \times m^T \tag{2.35}$$

$$p_2^T = -T^{-1}(Am^T + Bp_1^T) \tag{2.36}$$

Actual encoding step is aimed at solving the parity p_1^T and p_2^T in equation (2.35) and (2.36). In actual encoding step, it appears that the solution is obtained by pre-computing matrix $-\Phi^{-1}(-ET^{-1}A + C)$ of size $g \times (N-M)$ and then multiplying it with m^T of size $(N-M) \times 1$, giving complexity of the order of $O(g \times (N-M))$.

It is, however, possible to reduce complexity in actual encoding step by breaking the computation into smaller steps as given in Table 2.1 and Table 2.2. Table 2.1 and Table 2.2 accomplish computation of parity p_1^T and p_2^T in equation (2.35) and (2.36) in several smaller steps.

Table 2.1 starts to compute Am^T which has complexity of the order $O(N)$ since matrix A is sparse. Computation of $T^{-1}(Am^T)$ is of the order of $O(N)$ when computed using back-substitution since matrix T is lower triangular form and $Y = T^{-1}(Am^T)$ is equivalent to $TY = (Am^T)$.

All operations in Table 2.1 have complexity of the order of $O(N)$ since matrix A, C, E and matrix T are all sparse matrices except operation number 6 since there is multiplication of matrix $-\Phi^{-1}$ which is dense matrix of size $(g \times g)$. The complexity of determining p_1^T based on Table 2.1 is of the order of $O(N + g^2)$.

Computation of matrix $-\Phi^{-1}$ is not included in Table 2.1 and Table 2.2 since an assumption is used that matrix $-\Phi^{-1}$ is already solved by other processes and yields a dense matrix of size $g \times g$. Therefore, complexity in operation number 6 of Table 2.1 is of the order of $O(g^2)$.

All operations in Table 2.2 have complexity of order of $O(N)$ since matrix A, B and T are all sparse matrices. Operation number 4 in Table 2.2 also has complexity in order of $O(N)$ since it can be solved by back substitution with lower triangular form of matrix T and $Z = T^{-1}(Am^T + B\, p_1{}^T)$ is equivalent to $T\, Z = (Am^T + B\, p_1{}^T)$.

The overall encoding complexity for Richardson-Urbanke method is seen to be of the order of $O(N + g^2)$ as shown in Table 2.1 since computation of $p_1{}^T$ is very crucial for getting a codeword (c).

Table 2.1: Computation for $p_1{}^T$ derived from [16]

No	Operation	Note	Complexity
1	Am^T	Multiplication by sparse matrix	$O(N)$
2	$T^{-1}(Am^T)$	Solved by back substitution $Y = T^{-1}(Am^T) \Leftrightarrow T\, Y = (Am^T)$	$O(N)$
3	$-E(T^{-1}Am^T)$	Multiplication by sparse matrix	$O(N)$
4	Cm^T	Multiplication by sparse matrix	$O(N)$
5	$(-ET^{-1}Am^T) + Cm^T$	Addition matrices number 3 and 4	$O(N)$
6	$-\Phi^{-1}(-ET^{-1}Am^T + Cm^T)$	Multiplication by dense $-\Phi^{-1}$ matrix with size of $g \times g$	$O(g^2)$
			$O(N + g^2)$

Table 2.2: Computation for p_2^T based on [16]

No	Operation	Note	Complexity
1	Am^T	Multiplication by sparse matrix	$O(N)$
2	Bp_1^T	Multiplication by sparse matrix	$O(N)$
3	$(Am^T + Bp_1^T)$	Addition matrices number 1 and 2	$O(N)$
4	$-T^{-1}(Am^T + Bp_1^T)$	Solved by back substitution $Z = T^{-1}(Am^T + Bp_1^T) \Leftrightarrow TZ = (Am^T + Bp_1^T)$	$O(N)$
			$O(N)$

Summary of the proposed encoding procedure proposed by Richardson-Urbanke method is described by Table 2.3 [16].

Table 2.3: Summary of encoding by Richardson-Urbanke method based on [16]

Pre-Processing Step
Input: Non-singular parity check matrix (H)
1. Triangulation Perform row and column permutations to bring matrix H into approximate lower triangular form with a gap g as small as possible. $H = \left[\begin{array}{c\|cc} A & B & T \\ C & D & E \end{array}\right]$
2. Check-Rank Use Gaussian elimination to effectively perform pre-multiplication in order to check matrix ($ET^{-1}B + D$) or matrix Φ is non-singular. If matrix Φ is singular then performs column permutation in parity check matrix (H) to remove this singularity. $\begin{bmatrix} I & 0 \\ -ET^{-1} & I \end{bmatrix} \times \begin{bmatrix} A & B & T \\ C & D & E \end{bmatrix}$ $\begin{bmatrix} A & B & T \\ -ET^{-1}A+C & -ET^{-1}B+D & 0 \end{bmatrix}$
Output: An equivalent parity check matrix given in triangulation process such that matrix ($ET^{-1}B + D$) or matrix Φ is non-singular.
Actual Encoding Step
Input: Parity check matrix (H) given in triangulation process with matrix ($ET^{-1}B + D$) or matrix Φ is non-singular.
1. Determine p_1 as shown in Table 2.2
2. Determine p_2 as shown in Table 2.3
Output: Codeword $c = [m, p_1, p_2]$ with $\boldsymbol{c}H^T = 0$.

2.4 Relevant Works

LDPC codes were invented by Gallager in 1960's. These LDPC codes are classified into regular codes [1, 2]. In terms of how LDPC codes are constructed, they can be divided into two categories which are random codes and structured codes.

Random LDPC codes are constructed by computer search based on certain design rules or graph structures such as girth and degree distributions of nodes [14]. Long random LDPC codes with large value of code length (N), in general, perform closer to the Shannon limit than structured LDPC codes but the lack of structure due largely to randomness presents serious disadvantages in terms of storing, accessing large parity check matrix (H) and no simple encoding [11]. If LDPC codes are designed with some structure then some of these problems can be overcome [11].

Structured LDPC codes are constructed based on algebraic (geometric) and combinatorial methods [14]. LDPC codes derived from algebraic methods are either cyclic or quasi cyclic. Some LDPC codes from combinatorial methods are also classified as quasi cyclic codes. One class of structured LDPC codes which is the most relevant to this work is a class of quasi-cyclic (QC) LDPC codes.

QC LDPC codes can be used to construct two type LDPC codes which are regular and irregular LDPC codes. As stated above, regular LDPC codes are denoted by notation (j, k) regular LDPC code having fixed number j of ones in each column and also k ones in each row while LDPC codes that do not satisfy the stated requirements in regular LDPC codes are called by irregular codes. It implies that irregular codes have sparse density of ones in matrix H but does not have fixed number of ones in each column or in each row. In the next section, we will explore some relevant works in regular LDPC codes that is continued by relevant works in irregular LDPC codes in section 2.4.2.

2.4.1 Regular LDPC Codes

The first code-construction of regular LDPC codes was suggested by Gallager in 1960's [1, 2]. Gallager proposed constructing LDPC codes by randomly placing ones and zeros in an $M \times N$ matrix H with constraint that each column of matrix H had the same number of j ones and each row of matrix H had the same number of k ones. This type of low density parity check (LDPC) codes is called (j, k) regular LDPC codes.

The construction of regular LDPC codes should obey two requirements mentioned earlier with one additional requirement. The first requirement is that the parity check matrix H, containing only elements zero and one, should contain very small proportion of ones. The second requirement says that the number of ones, common between any two columns, should be no greater than one. The additional requirement of regular LDPC codes is that parity check matrix (H) has fixed number of ones in each column and each row numbering j and k.

The value of j determines the capability of error detection and correction. The larger the value of j in regular LDPC code, the better the performance of the code, the more complex the hardware realization.

An example of LDPC code was proposed by Gallager in [1, 2] with the size of (15×20) matrix H, $M = 15$, $N = 20$, $j = 3$ and $k = 4$ described in Figure 2.5. Therefore, LDPC code invented by Gallager is recognized by (3, 4) regular LDPC code. Density (r) of Gallager's code below is $r = j / M = k / N = 0.2$. In Gallager's regular LDPC codes, there are at least $(j - 1)$ rows of matrix H which are linearly dependent [1].

It is also possible to calculate code rate for regular LDPC codes by counting the number of ones in matrix H. The number of ones in each column of H is denoted by the value of j and the number of ones in each row of H is denoted by the value of k. The total number of ones in matrix H is equal to $M \times k = N \times j$, then the value of M / N is equal to j / k.
If we change the value of M / N in equation (2.14) with j / k then:

$$R = 1 - M / N = 1 - j / k \tag{2.37}$$

The value of R in Gallager's code with (3, 4) regular LDPC code is given below:

$$R = 1 - j / k = 1 - 3/4 = ¼.$$

$$H = \left(\begin{array}{cccccccccccccccccccc}
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
\hline
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
\hline
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1
\end{array}\right)$$

Figure 2.7: (3, 4) Regular LDPC code

Another example of regular LDPC codes is given in the previous of Figure 2.2 that describes a (2, 4) regular LDPC code and its Tanner graph of size (4×8) for $M = 4$, $N = 8$, $j = 2$ and $k = 4$. In Figure 2.2, the value of code length N is 8 bits while the value of j and k are 2 and 4 in (2, 4) regular LDPC code. Therefore, the value of R is given by:

$$R = 1 - j / k = 1 - 2/4 = ½.$$

Let c be a codeword in C having j ones in each column with $\boldsymbol{c}\, H^T = 0$. If the code C is a regular LDPC code, there are j parity check sums orthogonal on every code bit c. This means that smaller error or any error pattern with $\lfloor j/2 \rfloor$ can be corrected, where $\lfloor \rfloor$ means the largest integer number not greater than $j/2$ [3], [20].

Therefore, the minimum distance (d_{min}) of regular LDPC code is at least $j + 1$ that is [3], [13]:

$$d_{min} \geq j + 1 \tag{2.38}$$

Based on the results in [22], the minimum distance (d_{min}) of (j, k) regular with quasi cyclic (QC) LDPC codes satisfy this equation:

$$d_{min} \leq (j + 1)! \tag{2.39}$$

This means that the largest value of the minimum distance (d_{min}) of (j, k) regular with QC LDPC codes is at most $(j + 1)!$.

Regular LDPC codes with minimum distance (d_{min}) are able to correct all error patterns of weight t or smaller weight than t as given in equation (2.3).

Encoding complexity of regular Gallager LDPC codes is shown to be of the order of $O(N^2)$. An improvement of Gallager's original random construction is done by Mackay who rediscovered Gallager LDPC codes in 1999 [9]. The key idea is to construct parity check matrix with a well-defined invertible sub-matrix of parity part. Thus, one can generate the generator matrix in term of invertible matrix.

As mention in section 1.3, MacKay and Davey [22] stated that regular LDPC codes with column weight $j = 3$ give weak codes if they are applied in high code rate ($R > 0.875$) using code length around thousand bits ($N \leq 4000$). They suggested to use regular LDPC codes with column weight $j = 4$ to get a better performance than column weight $j = 3$ in high code rate ($R > 0.875$).

The other construction of regular LDPC codes using an algorithm is proposed by Xiao Yu Hu et al [28] as non-algebraic method that builds Tanner graph with large girth by progressively establishing edges or connections between variable and check nodes in an edge-by-edge manner, called progressive edge-growth (PEG) construction. Regular construction of PEG algorithm has a good performance and large girth but encoding complexity is of the order of $O(N^2)$. Yet, another algorithm for regular LDPC codes proposes additional pivoting and bit-reversed algorithm to achieve non-singular parity check matrix in regular Gallager LDPC code that requires process of swapping rows-columns and bit-reversed procedure [29]. The process of swapping rows-columns and bit reversed procedure prepares matrix H in a manner suitable to be used in actual encoding step. Work in [29] claims to have low encoding complexity but how low the complexity, does not state clearly.

The algebraic construction of regular QC LDPC codes perform quite well compared to regular random LDPC codes at short to moderate block lengths. But, for long block lengths, a randomly constructed regular LDPC code typically performs somewhat better [11]. In fact, well-designed structured LDPC codes perform equally well as their equivalent random LDPC codes do in terms of BER performance, BLER performance and error-floor [14].

Some works on structured regular LDPC codes from a class of QC codes are given in [10], [11], [12], [13], [14], [18], [23], [26], [31], [32] and [35]. Fossorier [10] investigates the construction of LDPC codes from circulant permutation matrices. He shows that LDPC codes with a girth of at most 12 are relatively easy to obtain but such codes can not have a girth larger than 12. This construction does not guarantee achieving a full-rank matrix H.

R. Michael Tanner et al [11] present a class of algebraically structured QC LDPC codes and their convolutional counterparts. These QC LDPC codes compare favorably with that of randomly constructed for short to moderate code length while the performance of

LDPC convolutional codes is superior to that of the QC codes on which they are based. These QC LDPC codes have at least $(j - 1)$ dependent rows in matrix H.

Zongwang Li et al [12] address the issue of efficient encoding of QC LDPC codes. They find the generator polynomial of QC-LDPC codes from their parity check matrices. Even though they show that the encoding complexity of a QC LDPC code is linearly proportional to the length of the code, there may still exist linear dependent rows of matrix H due to cyclic structure [12].

Yu Kou et al [13] show a geometric approach to the construction of LDPC codes based on lines and points. Furthermore, they can obtain either cyclic or quasi-cyclic structure and their encoding can be achieved in linear time with simple feedback shift register. These long extended codes achieve a performance a few tenths of a decibel away from Shannon limit. However, the rows of their matrix H are not necessarily linearly independent [13].

Bassem Ammar et al [14] present a method for constructing structured regular LDPC codes based on a special type of combinatorial design. Several classes of these codes are quasi-cyclic and their encoding can be implemented with simple feedback shift registers. R. Michael Tanner [18] generates (3, 5) regular QC LDPC codes with code length 155 and minimum distance 20. T. Zang and Parhi [23] report a construction of (3, k) regular LDPC codes that fit to partly parallel decoder. Gabofetswe Malema and Liebelt [26] propose an algorithm to construct (2, k) regular QC LDPC code codes over wide range of girths, rates and lengths. Xu Xia et al [31] construct (3, k) regular QC LDPC codes at high code rate derived from permutation theorem for Latin squares. Sunghwan Kim et al [32] analyze the cycles of Tanner (3, 5) QC LDPC codes and their girth values are derived. H. Tang et al [35] propose regular Gallager LDPC codes and circulant LDPC codes using algebraic construction. All of these constructions of regular QC LDPC codes in [14], [18], [23], [26], [31], [32], [35] may have some linear dependent rows of matrix H.

Another structured construction of regular LDPC is derived from combinatorial methods as given by [14], [15], [49], [51]. In [14], Bassem Ammar et al present a method for constructing structured regular LDPC codes based on a special type of combinatorial design. Sarah. J. Johnson and S. R. Weller [15, 49] propose (3, k) and (4, k) regular LDPC codes based on resolvable Steiner 2-designs based on combinatorial method. Masaya Fujisawa and S. Sakata [51] show a class of regular LDPC codes from a cyclic difference family which is also a kind of combinatorial design. Unfortunately, all combinatorial constructions of regular LDPC codes in [14], [15], [49], [51] do not guarantee linearly independence rows among matrix H.

Based on discussion above, random construction of regular Gallager codes and regular structured construction of LDPC codes like QC LDPC and combinatorial designs may have some dependent rows in its matrix H. This condition means that the matrix H is rank-deficient and the codes have a slightly higher code rate (R) than the matrix H indicates denoted by $R \geq (1 - j/k)$.

In order that the rows of matrix H are linearly independent, a full-rank condition of matrix H is needed to be achieved by either erasing these dependent rows or replacing with new rows which are independent to each other. Basically, erasing or replacing these dependent rows to achieve a full-rank of matrix H needs additional time. The characteristic of rank–deficiency of matrix H occurs not only in regular LDPC codes but also in some of irregular LDPC codes.

Despite rank-deficiency of matrix H, another issue in regular LDPC codes is high encoding complexity. In general, before encoding, regular LDPC codes convert matrix H into systematic structure of matrix G by Gauss elimination that has complexity of the order of $O(N^3)$ where N is the code length of LDPC codes [11, 16]. When N is large, Gauss elimination can be costly in terms of both memory and operations involved [17]. Moreover, the parity part of matrix G is generally a dense matrix, so encoding complexity of LDPC codes becomes prohibitively complex, $\sim O(N^2)$ since the value of N achieves hundreds to thousands of bits or more [11, 52].

Observation

As shown previous, the general fact that the regular LDPC codes employ a very low complexity decoding, the rank-deficiency of matrix *H*, high encoding complexity and Gauss elimination before encoding are serious issues that researchers are still grappling with.

QC LDPC codes solve high encoding complexity in regular LDPC codes since they have sufficient structure to allow simple encoding and they can be encoded with simple shift registers based on their generator polynomial in matrix *G* [12, 13, 14]. Unfortunately, this type of encoding is only useful for a class of QC and cyclic codes.

Moreover, if QC LDPC codes are encoded by parity check matrix using inversion method, we still need other process before encoding. Hanghang Qi and Norbert Goertz [52] show that if regular QC LDPC codes are encoded by Richardson-Urbanke method as one example of encoding by parity check matrix using inversion method, they require a pre-processing step of the order of $O(N^3)$.

In order to overcome the general issues in the design of regular LDPC codes, we propose a code-construction method for constructing not only (3, *k*) regular but also irregular LDPC codes that has no rank-deficiency in matrix *H*. The proposed code-construction follows the construction method of quasi-cyclic (QC) LDPC codes that use right cyclic shift.

Moreover, the proposed LDPC codes can be encoded by parity check matrix using Richarson-Urbanke as one of matrix inversion method. It implies that the proposed LDPC codes are able to avoid pre-processing step of the order of $O(N^3)$. The proposed LDPC codes also have low encoding complexity and non-singular parity part.

2.4.2 Irregular LDPC Codes

Irregular LDPC codes are LDPC codes that do not satisfy the stated requirements in regular LDPC codes. It implies that their matrix *H* has sparse density of ones but does not have fixed number of ones in each column and in each row. If matrix *H* has variable number of ones occurring only in each column or in each row, such codes are still identified as irregular LDPC codes.

Since irregular LDPC codes have variable number of ones, we use the term 'degree distribution' to measure the variability of ones in matrix *H*. Degree distribution of variable nodes describes degree distribution of ones in each column while degree distribution of check nodes represents degree distribution of ones in each row. Degree distribution is useful to measure the value of *R* in random construction of irregular LDPC codes, since arbitrary matrix *H* is having unclear boundary between information part and its redundant part.

In degree distribution, we need to find fraction of edges which are connected to degree-*i* variable nodes denoted by λ_i in each column and fraction of edges which are connected to degree-*i* check nodes denoted by ρ_i in each row. Therefore, the value of *R* in irregular LDPC codes is defined in equation (2.40) where *i* is the total number of nodes with the value of $\sum_i \lambda_i = 1$ and $\sum_i \rho_i = 1$.

$$R = 1 - ((\sum_i \lambda_i / i) / (\sum_i \rho_i / i)) \qquad (2.40)$$

An example of a 4×8 irregular LDPC code and its Tanner graph are explained in Figure 2.8 that has an irregular composition of ones in each column and regular composition of four ones in each row.

$\sum_i \lambda_i / i = 1/8.$

$\sum_i \rho_i / i = 1/4.$

$$H = \begin{array}{c} \begin{matrix} v1 & v2 & v3 & v4 & v5 & v6 & v7 & v8 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{matrix} c1 \\ c2 \\ c3 \\ c4 \end{matrix} \end{array}$$

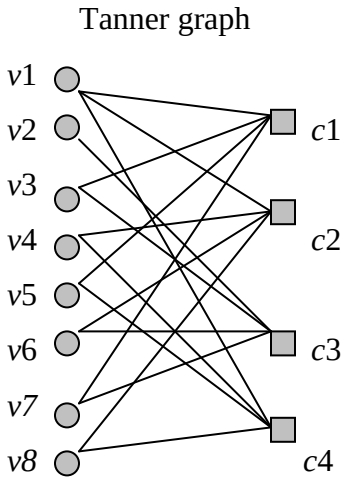

Figure 2.8: Irregular LDPC code and its Tanner graph

This irregular LDPC code has a code length of 8 bits with total number of variable nodes is eight and total number of check nodes is four. Therefore, the value of R for this irregular LDPC code is $R = 1 - 4/8 = ½$. Measuring the value of R for this irregular LDPC code can also be done by equation (2.14) above, so the value of R is $1 – 4/8 = ½$ that gives the same result as equation (2.40).

If the code C is an irregular code with minimum column weight (j_{min}), there are at least j_{min} parity check sums orthogonal on every code bit. Then, smaller errors or any error pattern with $\lfloor j_{min}/2 \rfloor$ can be corrected, where $\lfloor \rfloor$ means the largest integer number not greater than $j_{min}/2$ [3], [20].

Thus, the minimum distance (d_{min}) of irregular LDPC code is lower bounded by $j_{min}+1$ that is [3], [20]:

$$d_{min} \geq j_{min} + 1 \qquad (2.41)$$

Irregular LDPC codes with minimum distance (d_{min}) are able for correcting all error patterns of weight t or smaller weight than t as given in equation (2.3).

Irregular code-construction of LDPC codes were developed after the invention of (3, *k*) regular Gallager codes, firstly introduced in [36], [37] and further studied in [38], [39], [40] as randomly constructed LDPC codes. Luby et al showed that irregular random LDPC codes perform better than regular random ones [38, 39].

Another approach in random construction of irregular LDPC codes computed the threshold of noise level for a large class of binary input channels by density evolution like [8], [30]. Here, the threshold of noise level means the maximum noise level that has the zero probability as the code length tends to infinity [24]. In other word, density evolution is an expected behavior with cycle free graph to obtain the zero probability as the code length goes to infinity [8], [30].

Density evolution tracks the evolution of message distribution associated with probability of error as a function of iteration number and iterative decoding [25]. Density evolution observes convergence at some fixed SNR from one decoder iteration to the next iteration. A plot is made showing the evolution of density that will be used to determine a threshold. A threshold is defined when the probability of error converges to zero as the number of iterations tends to infinity and gives asymptotical performance of infinite code length in LDPC codes.

This approach has proven to achieve arbitrarily close to Shannon limit by long random irregular LDPC codes show by Chung et al that achieve outstanding performance using degrees of nodes varying from 2 to 8000. It showed that in computer simulation at a BER of 10^{-6}, code rate ½ and block length of 10^{7} bits with binary input can achieve a threshold of just 0.0045 dB away from Shannon limit in additive white Gaussian noise (AWGN) channel [8].

If the asymptotical performance of density evolution utilized in finite length of LDPC codes, a high error floor may result since Tanner graph for finite lengths can not be made cycle free. Therefore, it is not guaranteed that finite length LDPC codes with degree distribution suggested by density evolution will have good performance [24].

Generally, long random irregular LDPC codes outperform algebraically constructed irregular LDPC codes like cyclic or quasi cyclic. On the other hand, for medium-length LDPC codes (up to a few thousands bits long for code rate $R = ½$), the situation is quite different. For these lengths, irregular LDPC codes are generally not better than regular ones, and algebraic constructions can outperform random ones [10].

Works of irregular QC LDPC codes are shown in [11], [12], [24], [45], [46], [47], [48], [50]. R. Michael Tanner et al [11] and Zongwang Li et al [12] are able to construct irregular QC LDPC codes even though there is rank-deficiency of matrix *H*. Rank–deficiency of matrix *H* in irregular QC LDPC codes is also found in Sarah. J. Johnson and S. R. Weller [47] and Gianluigi Liva et al [50].

Other works on irregular QC LDPC codes are shown by Seho Myung et al [24], IEEE P802.16e™ [45], IEEE P802.11.n™/D1.02 [46] and Jeong Ki Kim et al [48]. These QC LDPC codes can be encoded using Richardson-Urbanke method and achieve linear encoding complexity by solving inverse matrix ($ET^{-1}B + D$) as an identity matrix with the value of $g^2 = 0$.

Some works on structured irregular LDPC codes that can be encoded by parity check matrix using inversion method are these of Li Ping et al [17], Michael Yang et al [41], R. Echard and S. Chang [42], Hughes Network System [43] by constructing parity part using dual diagonal construction.

Dual diagonal parity part is introduced by Li Ping et al [17] in 1999 known as semi random parity check matrix and classified into eIRA LDPC codes. Parity check matrix using dual diagonal parity part yields non-singular matrix and solves equation (2.37) without Gauss elimination of *H* and converting into *G*. Therefore, advantages of dual diagonal parity part in matrix *H* are non-singular parity part, no Gauss elimination in *H* if encoded by inversion method and linear encoding complexity denoted by $O(N)$.

Nowadays, LDPC codes have been implemented in the 2nd standard of digital video broadcasting (DVB-S2) for the satellite transmission of digital television in 2003 [44], adopted as one of optional ECC in IEEE 802.16e [45] and adopted as the up-coming IEEE 802.11n standard [46] even though these code performances are not categorized into outstanding performance in LDPC codes.

DVB-S2 standard applies dual diagonal construction of parity part in order to utilize all of its advantages that follow the construction in [17], [41], [42], [43]. Meanwhile, adopted LDPC codes for IEEE 802.16e standard [45] and the up-coming IEEE 802.11n standard [46] utilize base matrices of dual-diagonal parity part with single weight-3 column.

Base matrices of dual-diagonal parity part with single weight-3 column have the same advantages as dual-diagonal parity part and are categorized into irregular QC LDPC codes. An example of base 6 × 6 matrix of dual-diagonal parity part with single weight-3 column is given by Figure 2.9 [55].

$$H_p = \left(\begin{array}{c|ccccc} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{array}\right)$$

Figure 2.9: Matrix H_p of dual-diagonal parity with single weight-3 column.

Observation

In this section, we observe all relevant works of this thesis that have been presented above including regular and irregular LDPC codes. Derived from the discussion above, there are two categories of method of code-construction which are random and structured LDPC codes.

The most relevant to this work among structured LDPC codes is QC LDPC. Most commonly, the construction of QC LDPC codes whether regular or irregular LDPC codes have rank-deficiency of matrix *H*.

However, there is one example of irregular QC LDPC codes in the literature [45, 46, 55] that utilizes base matrices with dual-diagonal parity part and single weight-3 column. It gives rise to no rank-deficiency and has linear encoding complexity when encoded by their parity check matrix using Richardson-Urbanke. Linear encoding complexity of Richardson-Urbanke method in this irregular QC LDPC codes is achieved by solving a matrix equation where the inverse matrix ($ET^{-1}B + D$) is equal to an identity matrix while $g^2 = 0$. This type of irregular QC LDPC code has the same advantages as those with dual-diagonal parity part. As already mentioned above, these irregular QC LDPC codes have been adopted for IEEE 802.16e standard [45] and the up-coming IEEE 802.11n standard [46].

Unfortunately, these advantages of parity-check matrix with dual-diagonal parity part and/or dual-diagonal parity part with single weight-3 column are only applicable for irregular LDPC codes precluding regular LDPC codes. Irregular LDPC codes are type of LDPC codes that do not satisfy the constant number of ones in each row and column.

Therefore, there is scope for research into code-construction method for regular LDPC codes that has all the stated advantages of dual-diagonal parity part.

The summary of relevant works in regular and irregular LDPC codes is presented in Table 2.4.

Table 2.4: Summary of relevant works in regular and irregular LDPC codes

References		Proposed Technique Contribution	Results	Analysis / Comments
Ref	Year			
[1, 2] Gallager	1960's	The first code-construction method of LDPC codes with iterative decoding.	○ (3, 4) regular LDPC code with $N = 20$. ○ There are at least $(j - 1)$ linear dependent rows of H.	Encoding complexity is $O(N^2)$ by converting H to G.
[36] M. Luby et al	1998	Novel algorithm to construct irregular LDPC codes.	Encoding complexity is $O(\ln(1/\epsilon) \times N)$ with ϵ is any real number.	This construction does not guarantee to obtain a full-rank matrix H.
[38] M. Luby et al	1998	Novel algorithm to construct irregular random LDPC codes.	Compare LDPC codes with code rate $R = ½$ and show that irregular random LDPC perform better than regular random ones.	○ Encoding complexity is $O(N^2)$. ○ This construction does not guarantee a full-rank H.
[17] Li Ping et al	1999	The first dual-diagonal construction of parity part. This code is known as extended-irregular RA (e-IRA).	Dual-diagonal parity part with $R = \{1/3, ½, 2/3\}$, $K = 3000$ in AWGN channel.	○ Encoding complexity is $O(N)$ by inversion of sub matrix of H. ○ This construction guarantees a full-rank H.
[9] MacKay	1999	Matrix H with a well-defined invertible parity part. Thus, one can generate matrix G in term of invertible matrix.	(3, 6) regular LDPC codes with $R = ½$, $N = \{504, 1008\}$ and AWGN channel.	○ Encoding complexity is $O(N^2)$. ○ This construction does not guarantee a full-rank H.
[22] MacKay and Davey	2000	Evaluation of Gallager codes for short block length and high rate application.	○ (4, k) regular LDPC codes with $R > 0.875$, $N \leq 4000$ and AWGN channel. ○ (3, k) regular LDPC codes with $j = 3$, $R > 0.875$ and $N \leq 4000$ are weak codes. ○ The minimum distance (d_{min})	○ Encoding complexity is $O(N^2)$. ○ This construction does not guarantee a full-rank H.

			of (j, k) regular QC LDPC codes.	
[8] S.Y. Chung et al	2001	The best random irregular LDPC code using density evolution and varying degrees of nodes from 2 to 8000.	An irregular LDPC code achieves 0.0045 dB away from Shannon limit at a BER of 10^{-6} using with R = ½, $N = 10^7$ and AWGN channel.	○ Encoding complexity is $O(N^2)$. ○ This construction does not guarantee a full-rank H.
[17] Richardson and Urbanke	2001	An algorithm to encode the information bits for any type of LDPC codes using an approximate lower triangular form of H.	Example of (3, 6) regular LDPC code with R = ½, N = 12, g = 2 and encoding complexity $O(N + g^2)$.	This construction does not guarantee a full-rank H.
[30] Richardson et al	2001	Random regular and irregular LDPC codes using density evolution.	Random (3, 6) regular LDPC code achieves 0.06 dB away from Shannon limit at a BER of 10^{-6} with R = ½, $N = 10^6$ bits and AWGN channel.	○ Encoding complexity is $O(N^2)$. ○ This construction does not guarantee a full-rank H.
[13] Yu Kou et al	2001	The first construction of (j, k) regular QC LDPC codes.	Regular LDPC codes with j > 3, R > ½, N > 250, AWGN channel, SPA decoding and how close to Shannon limit at BER of 10^{-4} and 10^{-5}.	○ Encoding complexity is $O(N)$ using shift registers that is useful only for a class of QC and cyclic. ○ This construction does not guarantee a full-rank H.
[44] DVB-S2 Standard	2003	Outer codes for encoder/decoder of DVB-S2 using eIRA LDPC and dual-diagonal parity.	Irregular LDPC codes with variable R, code length N = {16200, 64800} and full-rank of H.	Encoding complexity is $O(N)$ by inversion of sub matrix of H.
[29] Su Chang Chae and Park	2004	Additional pivoting and bit-reversed algorithm to achieve non-singular matrix H.	(3, 6) regular LDPC codes with R = ½, N = {900, 1800}, AWGN channel and BPSK modulation.	This claims to have low encoding complexity but how low the complexity, does not state clearly.
[10] Fossorier	2004	Construction of QC LDPC codes from	○ (3, 13), (4, 9), (4, 18), (8, 18) and (16, 16) regular	○ Encoding complexity is $O(N)$ using shift registers

		circulant permutation matrices.	QC LDPC codes with R = {0.55, 0.77}, N = {1050, 4100} and AWGN channel. ○ A simple and sufficient condition to determine QC LDPC codes with girth ≤ 12.	that is useful only for a class of QC and cyclic. ○ This construction does not guarantee a full-rank H.
[11] R. Michael Tanner et al	2004	Design of QC LDPC codes and their convolutional counterparts. The regular convolutional LDPC codes outperform regular QC LDPC codes.	○ (3, 5), (5, 7) regular and irregular QC LDPC codes with $R < 0.5$, $N \leq 10000$ and AWGN channel. ○ These QC LDPC codes have at least $(j - 1)$ dependent rows in matrix H.	○ Encoding complexity is $O(N)$ using shift registers that is useful only for a class of QC and cyclic. ○ This construction does not guarantee a full-rank H.
[14] Bassem Ammar et al	2004	(4, k) and (5, k) regular LDPC codes based on combinatorial design.	(4, k) and (5, k) regular LDPC codes with $R > 0.7$, $N < 8500$, AWGN channel, SPA decoding and how close to Shannon limit at BER of 10^{-6}.	○ Several classes are QC codes and encoding complexity is $O(N)$ by shift register. While, other classes have complexity of the order of $O(N^2)$. ○ This construction does not guarantee a full-rank H.
[23] T. Zang and Parhi	2004	A partly parallel decoder using (3, k) regular LDPC codes.	(3, 6) regular LDPC codes with R = 0.5, N = {2304, 4608}, AWGN channel and BPSK modulation.	○ Encoding complexity is $O(N)$ using shift registers that is useful only for a class of QC and cyclic. ○ This construction does not guarantee a full-rank H.
[45] IEEE 802.16e™	2005	A standard for encoder/decoder of IEEE 802.16.e using irregular QC LDPC codes.	Irregular QC LDPC codes with base matrices of dual-diagonal parity part with single weight-3 column.	○ Encoding complexity is O(N) using Richardson - Urbanke method. ○ This construction guarantees a full-rank H.

[28] Xiao Yu Hu et al	2005	An algorithm to build Tanner graph with large girth by progressively establishing edges called progressive edge-growth (PEG) as non-algebraic method.	(3, 6) regular LDPC codes with R = ½, N = {504, 1008} and AWGN channel compared with MacKay's work [9].	Encoding complexity is $O(N^2)$ by converting H to G
[24] Seho Myung et al	2005	A code-construction for irregular LDPC codes with linear encoding complexity.	Irregular QC LDPC code with R = ½, N = 1000 and AWGN channel.	Encoding complexity is $O(N)$ using Richardson - Urbanke method.
[12] Zongwang Li et al	2006	Efficient encoding of QC LDPC codes using simple shift register.	(4, 32) regular QC LDPC and irregular QC LDPC code with R = 0.875, N = {8176, 10272}, AWGN channel, BPSK modulation, SPA decoding and how close to Shannon limit at BER of 10^{-6}.	○ Encoding complexity is $O(N)$ using shift registers that is useful only for a class of QC and cyclic. ○ This construction does not guarantee a full-rank H.
[26] Gabofetswe Malema and Liebelt	2007	An algorithm to construct (2, k) regular QC LDPC codes over wide range of girths.	(2, k) regular QC LDPC code with R < 0.875, N < 4500, AWGN channel and BPSK modulation.	○ Encoding complexity is $O(N)$ using shift registers that is useful only for a class of QC and cyclic. ○ This construction does not guarantee a full-rank H.
[52] Hanghang Qi and Norbert Goertz	2007	Investigation of encoding process when regular QC LDPC codes are encoded by Richardson-Urbanke method.	(2, k), (3, k), (4, k) and (5, k) regular QC LDPC codes with N < 3000.	○ Encoding complexity is $O(N + g^2)$. ○ Regular QC LDPC codes require a pre-processing step of the order of $O(N^3)$.

2.5 Summary

This chapter explores background concept of this work, some crucial issues in regular LDPC codes and some related works in LDPC codes. In general, issues in regular LDPC codes are rank-deficiency of matrix *H*, Gauss elimination before encoding and high encoding complexity.

QC LDPC codes achieve linear encoding complexity since they have sufficient structure to allow simple encoding and they can be encoded with simple shift registers based on their generator matrices. Unfortunately, QC LDPC codes may have some dependent rows in matrix *H*. If QC LDPC codes are encoded by parity check matrix using inversion method, they require a pre-processing step of the order of $O(N^3)$.

Meanwhile, implemented irregular LDPC codes in some standards [44], [45], [46] achieve non-singular parity part, linear encoding complexity and no pre-processing step of the order of $O(N^3)$ if encoded by inversion method even though their code performances are not the best in irregular LDPC codes.

Derived from explanation above, one characteristic of implemented LDPC codes is that they can be encoded by their parity check matrix using matrix inversion method with no pre-processing step.

CHAPTER 3

PROPOSED LDPC CODE-CONSTRUCTION

3.1 Introduction

In the previous chapter, we covered background study and literature review of this thesis that includes basic concepts of LDPC codes, encoding of LDPC codes and some relevant literature on (3, *k*) regular and irregular LDPC codes. It also included some critical issues in general design of regular LDPC codes like the rank-deficiency of matrix *H*, Gauss elimination before encoding and high encoding complexity.

In order to overcome these problems in the design of regular LDPC codes, section 3.2 proposes a novel code-construction method for constructing not only (3, *k*) regular LDPC codes but also irregular LDPC codes that has no rank-deficiency in matrix *H*, has low encoding complexity and also has no singular parity part.

The details of how to obtain the desired code rate, *R*, for (3, *k*) regular and irregular LDPC codes is described in section 3.3. Section 3.4 explores encoding scheme utilized in the proposed code-construction that covers both the encoding procedure and its encoding complexity.

3.2 Code-Construction

The proposed code-construction method follows the construction method of quasi-cyclic (QC) LDPC codes that use right cyclic shift. It consists of an information part and a non-

singular parity part. For achieving no rank-deficiency of matrix H, low encoding complexity and non-singular parity part, we propose a structured construction of matrix H, wherein the information part (H_{inf}) employs random right cyclic shift of identity matrix and the parity part (H_{par}) employs an approximate lower triangular form.

The proposed code-construction is divided into three stages. The first stage consists of building the parity part of the matrix H (using deterministic base matrix) in the form of approximate lower triangular structure, then expanding the parity part by one expansion factor. The second stage builds the information part of the matrix H (using deterministic base matrix), then expanding the information part by another expansion factor, and lastly combined together as one matrix H. In the third stage, each element in H is expanded by another expansion factor (L_1), wherein element 0 becomes the $L_1 \times L_1$ matrix of 0's and each element 1 becomes an identity matrix of $L_1 \times L_1$.

The proposed matrix H is represented by $H = [H_{inf} \mid H_{par}]$ as given in Figure 3.1.

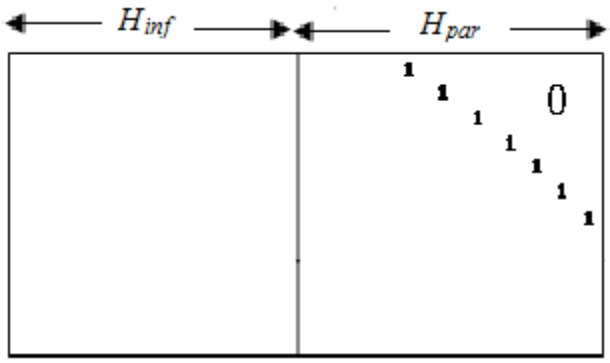

Figure 3.1: Proposed parity check matrix (H).

An advantage of the proposed matrix H is that its parity part (H_{par}) can be used for constructing not only (3, k) regular but also irregular LDPC codes. The distinction between the proposed (3, k) regular and irregular LDPC codes is mainly in their information part (H_{inf}) that is constructed in the second stage of code-construction. The

(3, k) regular code follows the condition of 3 ones in each column of the information part while the irregular code is not required to abide by this condition.

Suppose we would like to construct (3, k) regular LDPC codes, it means that there is fixed number of 3 ones in each column and fixed number of k ones in each row of matrix H. In order to satisfy (3, k) regular in the proposed matrix H, there must be constant number of 3 ones in each column of information part and there should be constant number of 3 ones in each column of parity part. For a general formulation, let j_i be the number of ones in each column of information part and let j_p be the number of ones in each column of the parity part. Of course, for (3, k) regular LDPC codes, $j = j_i = j_p = 3$.

Since the proposed matrix H has two parts, constant number of k ones in each row of matrix H is achieved by adding constant number of ones in each row of information part denoted by parameter k_i and also with constant number of ones in each row of parity part denoted by parameter k_p. In other words, the value of parameter k is equal to $k = k_i + k_p$. These four parameters (parameter j_i, j_p, k_i and k_p) retain their values in the first, the second and the third stage of code-construction.

There are eight parameters used in the proposed code-construction. They are parameter *base*, expansion factor L, expansion factor Z, expansion factor L_1, parameter j_i, parameter k_i, parameter j_p and parameter k_p. Parameters utilized in the first stage of code-construction are parameter *base*, expansion factor L, parameter j_p and parameter k_p. The second stage employs expansion factor Z, parameter j_i and parameter k_i while the third stage of code-construction utilizes only one parameter called expansion factor L_1.

There are two parameters ($j_p = 3$ and $k_p = 3$) determined from the specifications while expansion factor Z is influenced by expansion factor L, parameter *base* and j_i. This implies that only five parameters (*base*, L, L_1, j_i and k_i) influence the proposed code-construction. Parameters applied in the proposed code-construction are explained in Table 3.1 for (3, k) regular LDPC and irregular LDPC codes.

Table 3.1: Parameters utilized in the proposed code-construction

Type of LDPC Codes	**1st Stage of Code-Construction**				**2nd Stage of Code-Construction**			**3rd Stage of Code Construction**
	Parity Part				**Information Part**			
	j_p **(Given)**	k_p **(Given)**	*Base* **(Design variable)**	L **(Design variable)**	j_i **(Design variable)**	k_i **(Design variable)**	Z **(Derived)**	L_1 **(Design variable)**
(3, *k*) Regular	$j_p = 3$ Constant value	$k_p = 3$ Constant value	*Base* Variable Value	L Variable Value	$j_i = 3$ Constant value	k_i Variable Value	Influenced by L, *base* and $j_i = 3$ In Equation (3.3)	L_1 Variable Value
Irregular	$j_p = 3$ Constant value	$k_p = 3$ Constant value	*Base* Variable Value	L Variable Value	$j_i \neq 3$ Variable Value not equal to 3	k_i Variable Value	Influenced by L, j_i and *base*. In Equation (3.2)	L_1 Variable Value

The proposed code-construction is also described in the flowchart given in Figure 3.2.

The use of these parameters in the proposed code-construction will be presented in the following section. The description of the proposed matrix H in the first stage, the second stage and the third stage of code-construction for (3, k) regular and irregular LDPC codes are given in section 3.2.1, section 3.2.2 and section 3.2.3.

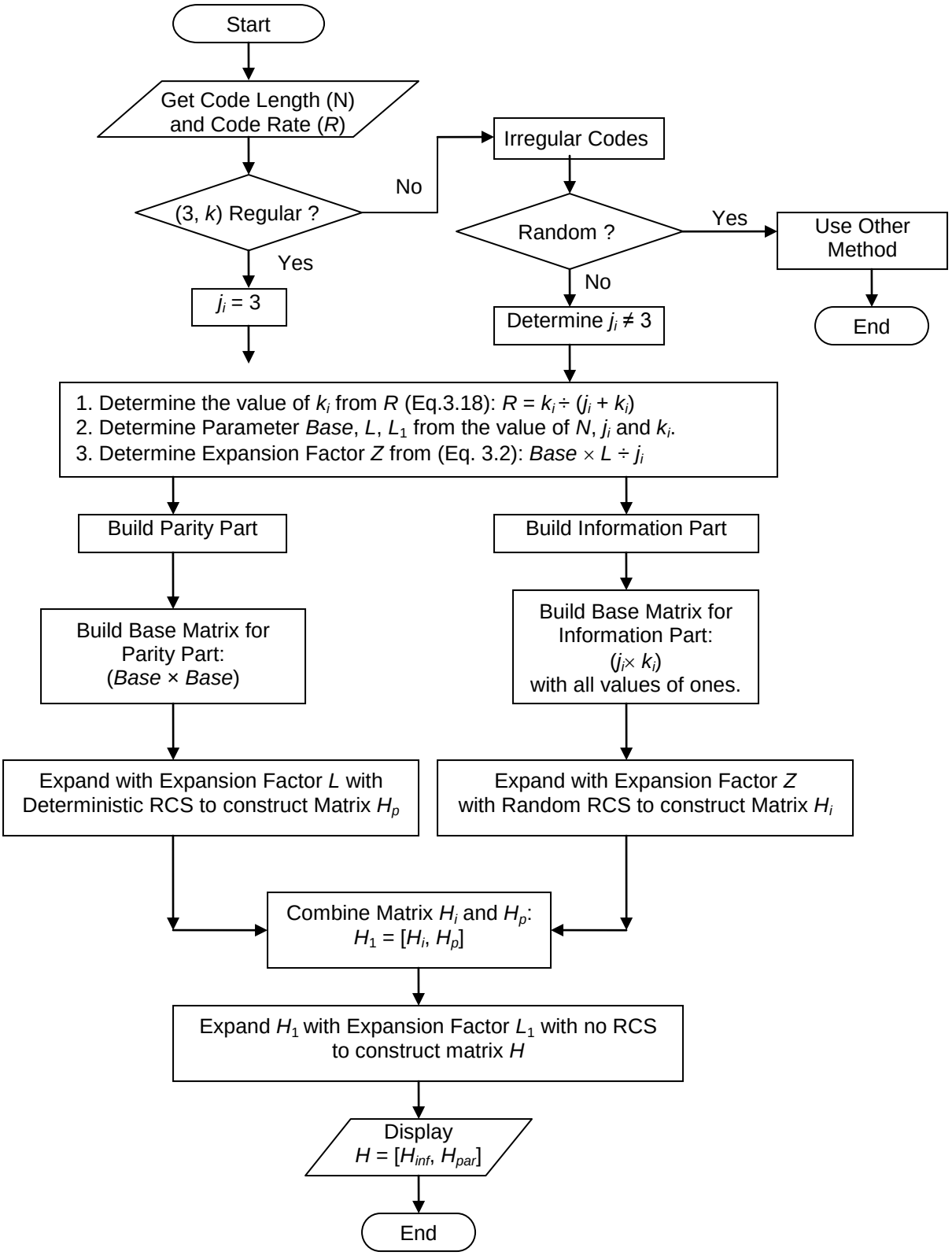

Figure 3.2: Flowchart of the proposed code-construction.

3.2.1 First Stage of Code-Construction – Parity Part

The basic design philosophy of the proposed parity part in the first stage of code-construction is to build a triple diagonal base matrix, to put some constraints to meet the condition of 3 ones in each column and each row in the form of (3, 3) regular LDPC code and to expand it with expansion factor L.

The main purpose of deterministic construction for parity part is to achieve non-singular parity matrix with an approximate triangular form which avoids pre-processing step of encoding as stated in section 2.3.2.2.

As already mentioned, there are four parameters utilized in the code-construction. Parameters utilized in constructing parity part in the first stage of code-construction are parameter *base*, expansion factor L and two deterministic parameters ($j_p = k_p = 3$).

Parameter *base* creates deterministic base matrix in parity part. This base matrix is then expanded by expansion factor L in the first stage of code-construction. Expanded base matrix of parity part is denoted by matrix H_p. The parity part of the proposed (3, k) regular and irregular LDPC codes is the same that constructs (3, 3) regular LDPC codes denoted by H_p.

Since the goal of deterministic base matrix of parity part is to build (j_p, k_p) regular LDPC code with $j_p = k_p = 3$, there is an additional parameter needed to construct base matrix in the form of triple diagonal matrix, given by parameter *base*. The parameter *base* acts as a variable with *base* > 3 so as to construct a deterministic base matrix of size (*base* × *base*) having (3, 3) regular form. Variable expansion factor $L \geq 3$ expands this base matrix into matrix H_p.

The design methodology given hereafter must adhere to the following requirements:

1. The base matrix, and therefore, the matrix H_p must have 3 ones in all the columns and rows to be (3, 3) regular base matrix.
2. The base matrix, and therefore, the matrix H_p must have none or as few as possible the girth of 4-cycles.
3. The base matrix, and therefore, the matrix H_p should be approximately lower triangular form.

Accordingly the following are the steps in designing the base matrix which is then suitably expanded to form the parity check matrix *H*.

- Build base matrix in the form of triple diagonal matrix with the size of (*base* × (*base*+*j*-1)). Each element of base matrix has coordinate position of (*x*, *y*) in the triple diagonal base matrix.

  An example is illustrated in Figure 3.3 with parameter *base* = 7, *j* = 3 and the size of (7×9). The column of base matrix in Figure 3.3 has coordinate of (1, 1) until (1, 9) while the row of base matrix in Figure 3.3 has coordinate of (1, 1) until (7, 1).

Base matrix of Parity Part =

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

Figure 3.3: Triple diagonal matrix with *base* = 7.

- Rearrange the location of all elements of ones in triple diagonal base matrix to construct (3, 3) regular base matrix with the new size of (*base* × *base*).

    - At position (*x*, *y*) of triple diagonal base matrix, each one with column $y \leq base$ has the new location of (x, y mod ($base$ + 1)).
    - At position (*x*, *y*) of triple diagonal base matrix, each one with column $y > base$ has the new location of (x, y mod $base$).
    - The new size of base matrix is (*base* × *base*) with an approximate lower triangular form and the size is no longer (*base* × (*base*+2)).

    Examples of the new location of ones in triple diagonal base matrix with parameter *base* = 7 is described below.

    The new location of one in base matrix at position (1, 3) is (1, 3 mod 8) = (1, 3).
    The new location of one in base matrix at position (2, 4) is (2, 4 mod 8) = (2, 4).
    The new location of one in base matrix at position (5, 7) is (5, 7 mod 8) = (5, 7).
    The new location of one in base matrix at position (6, 7) is (6, 7 mod 8) = (6, 7).
    The new location of one in base matrix at position (6, 8) is (6, 8 mod 7) = (6, 1).
    The new location of one in base matrix at position (7, 8) is (7, 8 mod 7) = (7, 1).
    The new location of one in base matrix at position (7, 9) is (7, 9 mod 7) = (7, 2).

    Figure 3.4 presents the new location of ones in triple diagonal matrix and the new size of base matrix with parameter *base* = 7. The new size of base matrix is (7 × 7) with an approximate lower triangular form.

    Total girth of length 4 in Figure 3.4 is six. Therefore, six girths of length 4 should be reduced as few as possible.

Base matrix of Parity Part =

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Figure 3.4: (3, 3) Regular base matrix with *base* = 7.

- Manipulate (3, 3) regular base matrix to reduce the existence of girth of 4-cycles as small as possible by shifting the second element of main diagonal until the last element of main diagonal to the left for 1 column.

  The ones marked with the straight line shown in Figure 3.5 is the second element of main diagonal until the last element of main diagonal which is moved 1 column to the left.

Base matrix of Parity Part =

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Figure 3.5: Manipulate main diagonal of base matrix with *base* = 7.

- Check the consistency of 3 ones in all the columns and rows.
  If there are more than 3 ones in one column, remove the additional one to the column that has ones below 3 to construct (3, 3) regular base matrix having 3 ones in all the columns and rows.

In Figure 3.5, there are 4 ones in the first column of base matrix and there are 2 ones in the last column of base matrix. Therefore, additional one in the first column of base matrix must be removed to the last column of base matrix to construct (3, 3) regular base matrix.

The bold one with square in Figure 3.6 is removed to the last column of base matrix to construct (3, 3) regular base matrix. The expected result of (3, 3) regular base matrix after one element has been removed is given by Figure 3.7.

Base matrix of Parity Part =

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ \boxed{\mathbf{1}} & 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Remove this one to the last column

Figure 3.6: Removing one element of base matrix with *base* = 7.

Base matrix of Parity Part =

$$\begin{pmatrix} \mathbf{1} & 1 & \mathbf{1} & 0 & 0 & 0 & 0 \\ \mathbf{1} & 0 & \mathbf{1} & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & \mathbf{1} & \mathbf{1} \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} \end{pmatrix}$$

Figure 3.7: Expected (3, 3) regular base matrix with *base* = 7.

After manipulating main diagonal of base matrix and checking 3 ones in all the columns and rows, the total girth of length 4 in Figure 3.7 is reduced to two. The bold ones in Figure 3.7 describe the total girth of 4-cycles in base matrix.

The two girths of 4-cycles are eliminated by the value of parameter *base* and expansion factor *L* applied in the construction of parity part. Feasible values of parameter *base* and expansion factor *L* with given choice of deterministic RCS that give matrix *H* with no rank-deficiency and girth of 4-cycles are given in the appendix.

An example of deterministic base matrix after applying some constrains with parameter *base* = 7 and the size of (7 × 7) is shown in Figure 3.8.

Base matrix of Parity Part =

$$\begin{pmatrix} 1 & 1 & \mathbf{1} & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & \mathbf{1} & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & \mathbf{1} \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Figure 3.8: The last form of base matrix of parity part with *base* = 7.

- Expand this base matrix with expansion factor of *L* to obtain parity part (H_p).
  Each zero in base matrix at position (*x*, *y*) is expanded to (*L* × *L*) zero matrix and each one in base matrix at position (*x*, *y*) is expanded to (*L* × *L*) identity matrix and cyclically shifted right according to some methods of deterministic right cyclic shift (RCS) as described in Table 3.2. The shift is to avoid the girth of 4-cycles in H_p.

  The ones marked with the straight line shown in Figure 3.8 should be maintained as (*L* × *L*) identity matrix and not be cyclically shifted, so as to achieve lower triangular

matrix form in parity part (H_p). The choice of deterministic RCS in Table 3.2 is determined by the value of parameter *base* and expansion factor L applied in the construction of parity part.

For example, if we take $L = 9$ and deterministic RCS given by $(x \times (y - 1)) \bmod L$ then each zero at position (x, y) in base matrix is expanded to (9×9) zero matrix.
And each one at position (x, y) in the base matrix is expanded to (9×9) identity matrix shifted according to $(x \times (y - 1)) \bmod L$. The ones marked with the straight line as shown in Figure 3.8 are retained as (9×9) identity matrix.

Table 3.2: Some methods of deterministic RCS

No	Deterministic RCS
1	$(x \times y) \bmod L$
2	$((x - 1) \times y) \bmod L$ This equation is based on [23].
3	$((x - 2) \times y) \bmod L$
4	$((x - 3) \times y) \bmod L$
5	$(x \times (y - 1)) \bmod L$
6	$(x \times (y - 2)) \bmod L$
7	$(x \times (y - 3)) \bmod L$

Following the method described above, the design yields, after expansion factor L, three ones in each column of parity part and three ones in each row of parity part (H_p). Thus, for (3, k) regular LDPC codes, we build (3, 3) parity part (H_p).

Assume that the size of the matrix of parity part (H_p) in the first stage of code-construction is $M_p \times N_p$ where M_p is the number of rows in matrix H_p and N_p is the number of columns in matrix H_p.

The value of M_p and N_p are given by:

$$M_p = N_p = Base \times L \tag{3.1}$$

3.2.2 Second Stage of Code-Construction – Information Part

The core design of the second stage of the proposed code-construction builds information part of the matrix H using deterministic base matrix with all elements of ones and expands it with expansion factor Z.

Parameters applied in constructing information part in the second stage of code-construction are expansion factor Z, parameter j_i and k_i. Parameter j_i and k_i construct a deterministic base matrix of information part with the size of ($j_i \times k_i$). This base matrix is then expanded by expansion factor Z in the second stage of code-construction. Expanded base matrix of information part is denoted by matrix H_i in the form of (j_i, k_i) regular LDPC code.

The primary design of base matrix of information part for (3, k) regular and irregular LDPC code is almost the same which builds a base matrix of ($j_i \times k_i$) with all elements of one expanded by expansion factor Z. The dissimilarity between (3, k) regular and irregular LDPC codes is given by the value of parameter j_i.

The value of parameter j_i for (3, k) regular LDPC code is $j_i = 3$ while the value of parameter j_i for irregular LDPC code is variable, not equal to 3 (parameter $j_i \neq 3$). In other

words, base matrix of irregular information part (H_i) has varying number of rows not equal to 3 ($j_i \neq 3$).

The design methodology of information part must adhere to the following requirements:

1. The base matrix of matrix H_i must have j_i ones in all the columns and k_i ones in all the rows to be (j_i, k_i) regular base matrix.
2. Matrix H_i is designed to produce none girth of 4-cycles in matrix $H_1 = [H_i \mid H_p]$ when combined with matrix H_p.

Construction of deterministic base matrix of information part for (3, k) regular and irregular LDPC code is given below:

- Build base matrix of information part (H_i) with the size of ($j_i \times k_i$) with all values of one. An example of base matrix of information part with $j_i = 3$ and $k_i = 4$ is shown in Figure 3.9.

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

Figure 3.9: Base matrix of information part with $j_i = 3$ and $k_i = 4$.

- Expand this base matrix with expansion factor Z to become information part (H_i).

  The value of expansion factor Z is determined by equation below:

$$Z = (Base \times L) \div j_i \qquad (3.2)$$

Each zero in base matrix at position (x, y) is expanded to ($Z \times Z$) zero matrix and each one in base matrix at position (x, y) is expanded to ($Z \times Z$) identity matrix and cyclically random shifted right.

The value of right cyclic shift (RCS) is created randomly to construct information part (H_i). The shift is to avoid the girth of 4-cycles in matrix $H_1 = [H_i \mid H_p]$ when combined with matrix H_p.

The resulting expanded base matrices of information part and parity part in the first and second stage of code-construction are combined to construct matrix H_1 in the form of $H_1 = [H_i \mid H_p]$ at the end of the second stage of code-construction.

The specific differences between base matrices of information part and parity part are given below:

- In the size of respective base matrix:
  The size of base matrix in information part is influenced by parameter j_i and k_i while the size of base matrix of parity part is determined by parameter *base*.

- In the respective base matrix:
  The elements of the base matrix in information part are all '1', while the elements of the base matrix in parity part are both '1' and '0' except that the '1's are typically along triple diagonal.

- In the way right cyclic shift (RCS) carried out:
  RCS in information part (H_i) with expansion factor Z is random while in parity part (H_p) it is deterministic in accordance with Table 3.2.

3.2.2.1 Information Part (H_i) for (3, k) Regular LDPC Codes.

The construction of information part in the second stage of code-construction for (3, k) regular LDPC code is the same as mentioned above. The proposed information part must follow two requirements of design methodology mentioned above.

The information part is built by firstly constructing a deterministic base matrix. As stated above, the distinction between (3, k) regular and irregular LDPC code is given by the value of parameter j_i. The value of parameter j_i for (3, k) regular LDPC code is $j_i = 3$. A deterministic base matrix of information part for (3, k) regular LDPC code is determined by parameter $j_i = 3$ and variable k_i. Therefore, the base matrix of (3, k) regular LDPC code has the size of $(3 \times k_i)$.

This base matrix is then expanded by expansion factor Z with random RCS to avoid girth of length 4 in matrix $H_1 = [H_i \mid H_p]$. The value of Z for (3, k) regular LDPC with $j_i = 3$:

$$Z = (Base \times L) \div 3 \tag{3.3}$$

Variables in equation (3.3) are expansion factor L and parameter *base* while parameter j_i remains constant that is $j_i = 3$. Therefore, expansion factor Z in (3, k) regular LDPC code is mainly affected by the value of expansion factor L and parameter *base* applied in parity part (H_p) of the first stage of code-construction.

Let the number of rows in matrix H_i be denoted by M_i and the number of columns in matrix H_i by N_i. The size of information part (H_i) is given by:

$$M_i = Z \times j_i = Z \times 3 \tag{3.4}$$

$$N_i = Z \times k_i \tag{3.5}$$

After the construction of information part (H_i) is complete, we combine parity part (H_p) in the first stage of code-construction with information part (H_i) to get (3, k) regular LDPC code in the second stage of code-construction denoted by matrix $H_1 = [H_i \mid H_p]$.

The example of matrix H_1 in the second stage of code-construction for (3, 5) regular LDPC code after information part (H_i) is combined with its parity part (H_p) having $j_i = 3$ and $k_i = 2$ is described in Figure 3.10.

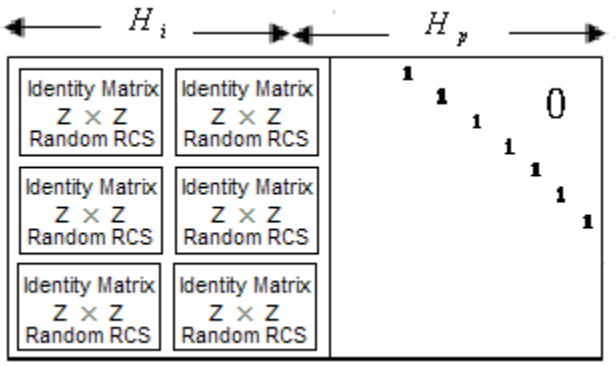

Figure 3.10: Matrix H_1 of (3, 5) regular LDPC code with $j_i = 3$ and $k_i = 2$.

Assume that the parity check matrix in the second stage of code-construction for (3, k) regular LDPC code is denoted by H_1. Matrix H_1 has $M_1 \times N_1$ where M_1 is the number of rows in matrix H_1 and N_1 is the number of columns in matrix H_1.

Since the number of rows in H_i denoted by M_i is the same with the number of rows in H_p denoted by M_p, the number of rows in matrix H_1 denoted by M_1 is also the same as M_i and M_p as described below:

$$M_1 = M_p = M_i = Base \times L = Z \times 3 \tag{3.6}$$

Based on equation (3.1), (3.2) and (3.5) with $j_i = 3$ the value of N_i and N_1 are:

$$N_i = (Z \times k_i) = (Base \times L \div j_i) \times k_i \tag{3.7}$$

$$N_1 = N_p + N_i = (Base \times L) + (Base \times L \times k_i \div j_i) \quad (3.8)$$

$$N_1 = (3 + k_i) \div 3 \times Base \times L \quad (3.9)$$

Table 3.3 gives a relation between variable k_i and code length (N_1) in the second stage of code-construction with any value of expansion factor L derived from equation (3.9).

Table 3.3: Code length (N_1) for (3, k) regular LDPC codes.

No	k_i	$(3 + k_i)$	$\boldsymbol{N_1}$ $(3 + k_i) \div 3 \times Base \times L$
1	1	4	$4/3 \times Base \times L$
2	2	5	$5/3 \times Base \times L$
3	3	6	$2 \times Base \times L$
4	4	7	$7/3 \times Base \times L$
5	5	8	$8/3 \times Base \times L$
6	6	9	$3 \times Base \times L$
7	7	10	$10/3 \times Base \times L$
8	8	11	$11/3 \times Base \times L$
9	9	12	$4 \times Base \times L$
10	10	13	$13/3 \times Base \times L$

3.2.2.2 Information Part (H_i) for Irregular LDPC Codes.

The construction of information part in the second stage of code-construction for irregular LDPC code is the same as mentioned above. The proposed information part must follow two requirements of design methodology mentioned above.

The information part is built by firstly constructing a deterministic base matrix. The value of parameter j_i for irregular LDPC code is not equal to 3 ($j_i \neq 3$). A deterministic base matrix of information part for irregular LDPC code is determined by variable j_i and k_i. Therefore, the base matrix of irregular LDPC code has the size of ($j_i \times k_i$). This base matrix is then expanded by expansion factor Z with random RCS to avoid girth of length 4 in matrix $H_1 = [H_i \mid H_p]$.

Since expansion factor Z for irregular LDPC code is determined by equation (3.2) above, expansion factor Z is not only affected by expansion factor L and parameter *base* applied in parity part (H_p) but also by variable j_i. Therefore, in order to satisfy equation (3.2), expansion factor L or parameter *base* should fulfill division by j_i.

Assume the size of information part (H_i) for irregular LDPC code is $M_i \times N_i$, where M_i is the number of rows in matrix H_i and N_i is the number of columns in matrix H_i. The value of N_i is the same as stated in equation (3.5) while the value of M_i with variable j_i for irregular LDPC codes is given below:

$$M_i = Z \times j_i \tag{3.10}$$

We combine parity part (H_p) and information part (H_i) to get irregular LDPC code in the second stage of code-construction denoted by matrix $H_1 = [H_i \mid H_p]$. An example of matrix H_1 for irregular LDPC code after information part (H_i) is combined with its parity part (H_p) having $j_i = 2$ and $k_i = 2$ is described in Figure 3.11.

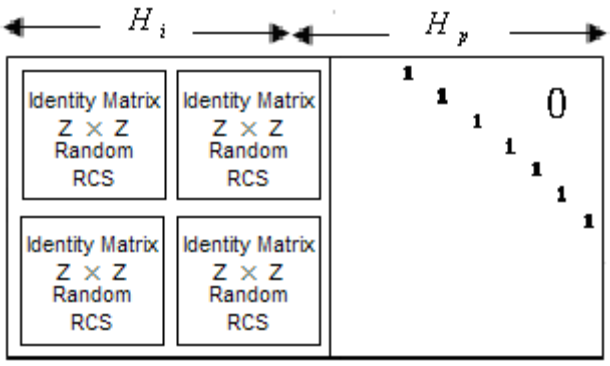

Figure 3.11: Matrix H_1 of irregular LDPC code with $j_i = 2$ and $k_i = 2$.

Suppose matrix H_1 in the second stage of code-construction has the size of $M_1 \times N_1$ where M_1 is the number of rows in matrix H_1 and N_1 is the number of columns in matrix H_1.

Since the number of rows in matrix H_i is the same with the number of rows in matrix H_p, the number of rows in matrix H_1 denoted by M_1 is also the same as M_i and M_p as described below:

$$M_1 = M_p = M_i = Base \times L = Z \times j_i \tag{3.11}$$

The value of code length (N_1) for irregular LDPC codes in the second stage of code-construction is defined below:

$$N_1 = (j_i + k_i) \div j_i \times Base \times L \tag{3.12}$$

Table 3.4 gives a relation between variable k_i, code length (N) and variable j_i based on equation (3.12) for irregular LDPC codes.

Table 3.4: Code length (N_1) for irregular LDPC codes

No	k_i	$\boldsymbol{N_1}$ $(j_i + k_i) \div j_i \times Base \times L$
1	1	$(j_i + 1) \div j_i \times Base \times L$
2	2	$(j_i + 2) \div j_i \times Base \times L$
3	3	$(j_i + 3) \div j_i \times Base \times L$
4	4	$(j_i + 4) \div j_i \times Base \times L$
5	5	$(j_i + 5) \div j_i \times Base \times L$
6	6	$(j_i + 6) \div j_i \times Base \times L$
7	7	$(j_i + 7) \div j_i \times Base \times L$
8	8	$(j_i + 8) \div j_i \times Base \times L$
9	9	$(j_i + 9) \div j_i \times Base \times L$
10	10	$(j_i + 10) \div j_i \times Base \times L$

3.2.3 Third Stage of Code-Construction

The purpose of third stage of code-construction achieves matrix $H = [H_{inf} \mid H_{par}]$ after expanding matrix $H_1 = [H_i \mid H_p]$ in the second stage of code-construction with identity matrix without any cyclic shift to build a longer code length than in the first and the second stage of code-construction. The construction of the third stage is the same for (3, k) regular LDPC and irregular LDPC codes. The third stage of code-construction is determined by one parameter that is expansion factor L_1.

Each zero in matrix $H_1 = [H_i \mid H_p]$ at position (x, y) is expanded to ($L_1 \times L_1$) zero matrix and each non-zero in matrix $H_1 = [H_i \mid H_p]$ at position (x, y) is expanded to ($L_1 \times L_1$) identity matrix with no right cyclic shift (RCS).

Let matrix $H = [H_{inf} \mid H_{par}]$ in the third stage of code-construction has the size of $M \times N$ where M is the number of rows in matrix H and N is the number of columns in matrix H. In relation to equation (3.11) and expansion factor L_1, the value of M is given by:

$$M = M_p = M_i = Base \times L \times L_1 = M_1 \times L_1 \tag{3.13}$$

According to equation (3.8) and expansion factor L_1, the value of N would be:

$$N = (N_p + N_i) \times L_1 = N_1 \times L_1 \tag{3.14}$$

The value of information part in the third stage of code-construction is denoted by $(N_i \times L_1) = (Z \times k_i \times L_1)$ while the value of parity part in the third stage of code-construction is denoted by $(N_p \times L_1) = (Base \times L \times L_1)$. Therefore, the value of code length N of LDPC codes would be:

$$N = N_1 \times L_1 = (N_p + N_i) \times L_1 = ((Base \times L) + (Z \times k_i)) \times L_1 \tag{3.15}$$

Based on equation (3.2), the value of $Base \times L$ is equal to $Z \times j_i$. Therefore, the value of N in equation (3.14) and (3.15) will be:

$$N = N_1 \times L_1 = ((Z \times j_i) + (Z \times k_i)) \times L_1 = (Z \times (j_i + k_i)) \times L_1 \tag{3.16}$$

If we replace $Z = (Base \times L) \div j_i$, the value of code length (N) for the third stage of code-construction is given by:

$$N = (Z \times (j_i + k_i)) \times L = Base \times L \times L_1 \times (j_i + k_i)) \div j_i \tag{3.17}$$

3.3 Code Rate

Derived from equation (2.13), code rate (R) of LDPC code is evaluated by measuring the number of information bits (K) over code length of LDPC codes (N). The number of information bits (K) in the proposed code-construction is the same as the length of information part in the third stage of code-construction denoted by ($N_i \times L_1$).

If we take the value of N in equation (3.16) and $K = N_i \times L_1 = Z \times k_i \times L_1$, the proposed value of code rate (R) is the same as:

$$R = K \div N = (Z \times k_i) \times L_1 \div ((Z \times (j_i + k_i)) \times L_1) = k_i \div (j_i + k_i) \qquad (3.18)$$

Variable code rate is developed by changing the number of columns in matrix H_{inf} and maintaining the number of rows in matrix H of the third stage of code-construction. Changing the number of columns in matrix H_{inf} is achieved by changing the value of parameter k_i in information part (H_{inf}).

3.3.1 (3, k) Regular LDPC Code

In order to measure the value of code rate (R) for (3, k) regular LDPC code in equation (3.18), we need to find the value of parameter k_i and j_i. Since we generate (3, k) regular LDPC code, the value of parameter j_i is always equal to $j_i = 3$. Therefore, parameter that influences equation (3.18) is parameter k_i.

If we put $j_i = 3$ in equation (3.18), the value of code rate (R) for (3, k) regular LDPC is:

$$R = k_i \div (3 + k_i) \qquad (3.19)$$

If we compare code rate (R) for (3, k) regular LDPC code in equation (3.19) with equation (2.15), the value of parameter j is equal to $j = j_i = j_p = 3$ where parameter j_i is the

value of ones in each column of information part and parameter j_p is the value of ones in each column of parity part.

The value of parameter k is determined by the value of ones in parity part denoted by k_p and information part denoted by k_i. The value of k with $k_p = 3$ would be:

$$k = k_p + k_i = 3 + k_i \tag{3.20}$$

Therefore, code rate (R) in equation (2.15) with $j = 3$ and the value of k derived from equation (3.20) are as follows:

$$R = 1 - (3 \div (3 + k_i)) \tag{3.21}$$

Table 3.5 gives a description of generating code rate (R) for (3, k) regular LDPC with $j = j_p = j_i = 3$, variable k_i from 1 until 10 and any value of parameter L and *base*. Based on equation (3.20) and (3.21), the value of k and R are:

Table 3.5: Variable code rate (R) for (3, k) regular LDPC codes

No	k_i	(3+ k_i)	(3, k) Regular LDPC	Code Rate (R) $k_i \div (3+ k_i)$
1	1	4	(3,4) Regular LDPC	¼
2	2	5	(3,5) Regular LDPC	2/5
3	3	6	(3,6) Regular LDPC	½
4	4	7	(3,7) Regular LDPC	4/7
5	5	8	(3,8) Regular LDPC	5/8
6	6	9	(3,9) Regular LDPC	2/3
7	7	10	(3,10) Regular LDPC	7/10
8	8	11	(3,11) Regular LDPC	8/11
9	9	12	(3,12) Regular LDPC	¾
10	10	13	(3,13) Regular LDPC	10/13

$k = k_p + k_i = 3 + k_i$.

$R = k_i \div (3 + k_i)$.

3.3.2 Irregular LDPC Code

Calculating code rate (R) of irregular LDPC code is the same as explained in section 3.3 and equation (3.18). Since the value of ones in each column of parity check matrix (H) (parameter j) is not the same for information and parity part, we couldn't measure the value of code rate (R) for irregular LDPC codes by equation (2.15).

Variable code rate (R) is obtained by changing the value of parameter k_i that yields variable length of information bits in matrix H_{inf}. Since the value of parameter j_i is variable, the value of parameter L or *base* should accomplish division by parameter j_i in order to satisfy equation (3.2). Examples of variable code rate (R) with variable j_i are described in Table 3.6 that lists variable code rate (R) for irregular LDPC codes with variable parameter k_i from 1 until 10, variable L and *base*.

Table 3.6: Variable code rate (R) for irregular LDPC codes.

No	k_i	Code Rate (R) $k_i \div (j_i + k_i)$
1	1	$1 \div (j_i + 1)$
2	2	$2 \div (j_i + 2)$
3	3	$3 \div (j_i + 3)$
4	4	$4 \div (j_i + 4)$
5	5	$5 \div (j_i + 5)$
6	6	$6 \div (j_i + 6)$
7	7	$7 \div (j_i + 7)$
8	8	$8 \div (j_i + 8)$
9	9	$9 \div (j_i + 9)$
10	10	$10 \div (j_i + 10)$

Based on the description above, the summary of how to construct the proposed code - construction is divided into six steps that are explained below:

1. Set initial parameters that are going to be used in the code design.
   These parameters are the value of code length (N) and code rate (R).

2. Specify type of LDPC code used: (3, k) regular or irregular LDPC code.
   This process influences the value of ones in each column of information part represented by parameter j_i.
   - If we choose to use (3, k) regular LDPC code, the value of parameter j_i will be $j_i = 3$.
   - If we select the value of parameter $j_i \neq 3$ (not equal to 3), this code will be classified into irregular LDPC code.

3. Set the other parameters utilized in the proposed codes which are parameter k_i, *base*, L, Z and L_1.
   - Define parameter k_i from its code rate for (3, k) regular or irregular LDPC code.
     - Parameter k_i of (3, k) regular LDPC code is derived from equation (3.19) that is $R = k_i / (3 + k_i)$.
     - Parameter k_i of Irregular LDPC code is derived from equation (3.18) that is $R = k_i / (j_i + k_i)$.

   - Define parameter *base*, L and L_1.
     - The values selected for parameter *base*, L and L_1 should fulfill equation (3.17) for the given value of N, j_i and k_i.
     - Parameter *base* and L must satisfy division by j_i in equation (3.2).
     - It is suggested to take parameter L_1 and L as small as possible and to enlarge parameter *base* since parameter L_1 and parameter L gives impact to encoding complexity and the value of g.

- Define parameter Z.

  According to equation (3.2) which is $Z = (Base \times L) \div j_i$.

4. Build parity part in the first stage of code-construction.
   - Build deterministic base matrix of parity part (H_p) with the size of ($base \times base$) having constraints as given in section 3.2.1.
   - Expand deterministic base matrix of parity part with expansion factor L and methods of deterministic right cyclic shift (RCS) based on Table 3.2.

5. Build information part in the second stage of code-construction.
   - Build deterministic base matrix of information part with the size of ($j_i \times k_i$) with all value of ones.
   - Expand deterministic base matrix of information part with expansion factor Z and random deterministic right cyclic shift (RCS).
   - Combine parity part (H_p) and information part (H_i) to get the proposed LDPC code in the second stage of code-construction denoted by matrix $H_1 = [H_i \mid H_p]$. Expansion factor Z with random RCS avoids girth of length 4 in matrix H_1.

6. Build the proposed code in the third stage of code-construction.
   Expand matrix $H_1 = [H_i \mid H_p]$ with expansion factor L_1 to construct matrix H in the third stage of code-construction.

   Expansion factor L_1 in the third stage of code-construction expands each one of $H_1 = [H_i \mid H_p]$ into ($L_1 \times L_1$) identity matrix with no RCS and expands each zero of $H_1 = [H_i \mid H_p]$ into ($L_1 \times L_1$) zero matrix to construct matrix $H = [H_{inf} \mid H_{par}]$.

Example of generating the proposed codes with initial parameter $N = 1008$ and $R = ½$:

1. (3, k) Regular LDPC Code
    - Parameter j_i: $j_i = 3$.
    - Parameter k_i: Derived from $R = k_i \div (3 + k_i) = ½$, it gives parameter $k_i = 3$.
    - Parameter *base*, L and L_1: Based on equation (3.17) to fulfill the given value of N, j_i and k_i.

      $N = (Base \times L \times L_1 \times (j_i + k_i)) \div j_i = (Base \times L \times L_1 \times 6) \div 3 = 1008$.

      Parameter *base* = 72 with deterministic RCS $(x \times (y - 1)) \bmod L$.

      Parameter $L = 7$.

      Parameter $L_1 = 1$.
    - Parameter $Z = (Base \times L) \div j_i = (72 \times 7) \div 3 = 168$.
    - The size of deterministic base matrix of information part (H_{inf}):

      $(3 \times k_i) = (3 \times 3)$ with all value of ones.
    - The size of base matrix of parity part (H_{par}): $(base \times base) = (72 \times 72)$ with some constraints given in section 3.2.1.

2. Irregular LDPC Code

   Parameter j_i: The value of parameter j_i in irregular LDPC codes can be applied to any integer value not equal to 3 denoted by $j_i \neq 3$.

    - If we take parameter $j_i = 4$.

      Derived from $R = k_i \div (j_i + k_i) = ½$, it gives parameter $k_i = 4$.

      The other parameter will be:

      Parameter *base* = 72 with deterministic RCS $(x \times (y - 1)) \bmod L$.

      Parameter $L = 7$ and parameter $L_1 = 1$.

      Parameter $Z = (Base \times L) \div j_i = (72 \times 7) \div 4 = 126$.

      The size of deterministic base matrix of information part (H_{inf}):

      $(j_i \times k_i) = (4 \times 4)$ with all value of ones.

      The size of base matrix of parity part (H_{par}): $(base \times base) = (72 \times 72)$ with some constraints given in section 3.2.1.

- If we take parameter $j_i = 6$.

  Based on $R = k_i \div (j_i + k_i) = ½$, it gives parameter $k_i = 6$.

  Parameter *base* = 72 with deterministic RCS $(x \times (y - 1)) \bmod L$.

  Parameter $L = 7$ and parameter $L_1 = 1$.

  Parameter $Z = (Base \times L) \div j_i = (72 \times 7) \div 6 = 84$.

  The size of deterministic base matrix of information part (H_{inf}):

  $(j_i \times k_i) = (6 \times 6)$ with all value of ones.

  The size of base matrix of parity part (H_{par}): $(base \times base) = (72 \times 72)$ with some constraints given in section 3.2.1.

- If we take parameter $j_i = 8$.

  Derived from $R = k_i \div (j_i + k_i) = ½$, it gives parameter $k_i = 8$.

  Parameter *base* = 72 with deterministic RCS $(x \times (y - 1)) \bmod L$.

  Parameter $L = 7$ and parameter $L_1 = 1$.

  Parameter $Z = (Base \times L) \div j_i = (72 \times 7) \div 8 = 63$.

  The size of deterministic base matrix of information part (H_{inf}):

  $(j_i \times k_i) = (8 \times 8)$ with all value of ones.

  The size of base matrix of parity part (H_{par}): $(base \times base) = (72 \times 72)$ with some constraints given in section 3.2.1.

- If we take parameter $j_i = 9$.

  Based on $R = k_i \div (j_i + k_i) = ½$, it gives parameter $k_i = 9$.

  Parameter *base* = 72 with deterministic RCS $(x \times (y - 1)) \bmod L$.

  Parameter $L = 7$ and parameter $L_1 = 1$.

  Parameter $Z = (Base \times L) \div j_i = (72 \times 7) \div 9 = 56$.

  The size of deterministic base matrix of information part (H_{inf}):

  $(j_i \times k_i) = (9 \times 9)$ with all value of ones.

  The size of base matrix of parity part (H_{par}): $(base \times base) = (72 \times 72)$ with some constraints given in section 3.2.1.

- If we take parameter $j_i = 12$.

  Derived from $R = k_i \div (j_i + k_i) = ½$, it gives parameter $k_i = 12$.

  Parameter *base* = 72 with deterministic RCS $(x \times (y - 1)) \bmod L$.

  Parameter $L = 7$ and parameter $L_1 = 1$.

  Parameter $Z = (Base \times L) \div j_i = (72 \times 7) \div 12 = 42$.

  The size of deterministic base matrix of information part (H_{inf}):

  $(j_i \times k_i) = (12 \times 12)$ with all value of ones.

  The size of base matrix of parity part (H_{par}): $(base \times base) = (72 \times 72)$ with some constraints given in section 3.2.1.

- If we take parameter $j_i = 18$.

  Based on $R = k_i \div (j_i + k_i) = ½$, it gives parameter $k_i = 18$.

  Parameter *base* = 72 with deterministic RCS $(x \times (y - 1)) \bmod L$.

  Parameter $L = 7$ and parameter $L_1 = 1$.

  Parameter $Z = (Base \times L) \div j_i = (72 \times 7) \div 18 = 28$.

  The size of deterministic base matrix of information part (H_{inf}):

  $(j_i \times k_i) = (18 \times 18)$ with all value of ones.

  The size of base matrix of parity part (H_{par}): $(base \times base) = (72 \times 72)$ with some constraints given in section 3.2.1.

- If we take parameter $j_i = 24$.

  Derived from $R = k_i \div (j_i + k_i) = ½$, it gives parameter $k_i = 24$.

  Parameter *base* = 72 with deterministic RCS $(x \times (y - 1)) \bmod L$.

  Parameter $L = 7$ and parameter $L_1 = 1$.

  Parameter $Z = (Base \times L) \div j_i = (72 \times 7) \div 24 = 21$.

  The size of deterministic base matrix of information part (H_{inf}):

  $(j_i \times k_i) = (24 \times 24)$ with all value of ones.

  The size of base matrix of parity part (H_{par}): $(base \times base) = (72 \times 72)$ with some constraints given in section 3.2.1.

3.4 Encoding

This section discusses encoding procedure and encoding complexity that are utilized in the proposed code-construction. The encoding procedure of the proposed codes is given in section 3.4.1 that is based on Richardson-Urbanke method [16]. Encoding method based on Richardson-Urbanke is selected since this method can be applied to any LDPC codes like random or structured codes.

Encoding complexity of the proposed code-construction is explained in section 3.4.2 that focuses to formulate the value of gap g in relation to code length N. The value of gap g is going to be used to measure encoding complexity of (3, k) regular and irregular LDPC codes as in [16].

3.4.1 Encoding Procedure

In order to utilize the sparseness of parity check matrix (H) and to simplify encoding and decoding process by using only one matrix, this work applies encoding by matrix H. Since the proposed code-construction focuses on avoiding pre-processing step of encoding envisaged in Richardson-Urbanke method, we use only actual encoding step of this method as described in section 2.3.2.2.

Derived from actual encoding step of Richardson-Urbanke method, we need to divide the proposed information part (H_{inf}) and parity part (H_{par}) into several sparse sub matrices and each matrix contains at most $O(N)$ elements as given in equation (2.27) and Figure 2.6.

Sub matrices of information part (H_{inf}) are matrix A and C while sub matrices of parity part (H_{par}) are matrix B, D, E and T, of which matrix T has a built-in lower triangular with ones along diagonal.

All of these sub matrices are going to be used in mathematical equations of encoding step that are given in equation (2.32) until equation (2.36). An example of how parity check matrix (*H*) is divided into *A*, *B*, *C*, *D*, *T* and *E* sub matrices is explained in Figure 3.12 with a (7×10) parity check matrix (*H*).

$$H = [\, H_{inf} \mid H_{par}]$$

$$H = \left[\begin{array}{c|cc} A & B & T \\ C & D & E \end{array}\right]$$

$$H = \left(\begin{array}{ccc|cc|ccccc} \multicolumn{3}{c|}{\mathbf{A}} & \multicolumn{2}{c|}{\mathbf{B}} & \multicolumn{5}{c}{\mathbf{T}} \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ \hline 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ \multicolumn{3}{c|}{\mathbf{C}} & \multicolumn{2}{c|}{\mathbf{D}} & \multicolumn{5}{c}{\mathbf{E}} \end{array}\right)$$

Figure 3.12: Sub matrices *A*, *B*, *C*, *D*, *T* and *E* in a (7×10) matrix *H*.

How the proposed base matrix of parity part in section 3.2.1 is divided into sub matrix *B*, *D*, *T* and *E* is given in Figure 3.13. Figure 3.13 takes Figure 3.8 with (7×7) base matrix as an example.

The parameter *g* in Figure 3.13 is a measure of gap in its lower triangular form. Based on Figure 3.13, the value of *g* for base matrix of parity part is equal to $2 \times L$ while the value of *g* for the third stage of code-construction is given by $g = 2 \times L \times L_1$.

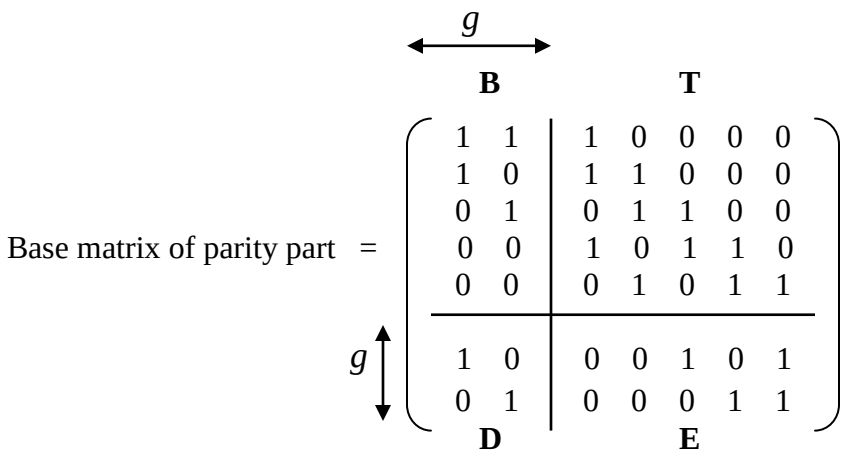

Figure 3.13: Sub matrices *B, D, T* and *E* in base matrix of parity part with *base* = 7.

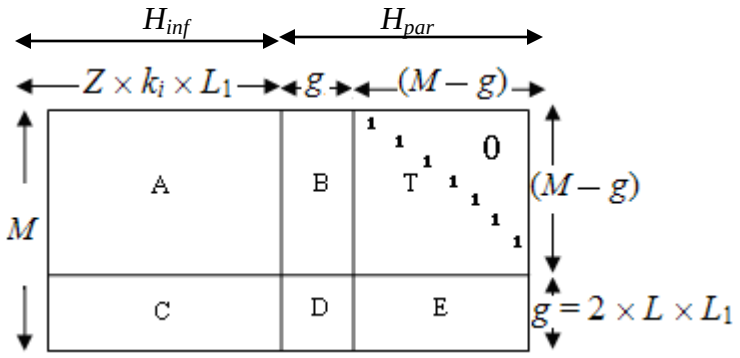

Figure 3.14: Matrix *H* in the third stage of code-construction

The detail of sub matrices at the third stage of code-construction is described by Figure 3.14 with $g = 2 \times L \times L_1$. Derived from Figure 3.14, the size of sub matrices at the third stage of code-construction is achieved by *A* is $(M - g) \times (Z \times k_i \times L_1)$, *C* is $g \times (Z \times k_i \times L_1)$, *B* is $(M - g) \times g$, *T* is $(M - g) \times (M - g)$, *D* is $g \times g$ and *E* is $g \times (M - g)$.

Let $c = [m, p_1, p_2]$ be a codeword in the third stage of code- construction where *m* is the information bit with the length of $k_i \times Z \times L_1$, the parity part, divided into parity p_1 with length *g* and parity p_2, with length $(M - g) = (M - (2 \times L \times L_1))$.

The actual encoding begins with equation (2.9) and uses equation (2.35) and (2.36) to solve the parity p_1^T and p_2^T. Finding a codeword (c) is accomplished by inserting those values of parity p_1 and p_2 into $c = [m, p_1, p_2]$. To check whether the value of codeword c is a correct codeword, it must satisfy equation (2.9).

Example:

An example of how the proposed (3, k) regular LDPC code is encoded is given below. The proposed (3, 6) regular LDPC code uses parameter $L_1 = 1$, $L = 8$, $base = 15$, $j_i = 3$, $k_i = 3$ and deterministic RCS $((x - 1) \times y) \bmod L$. The matrix H is divided into 6 sub-matrices as given in Figure 3.13 and 3.14. Encoding solves parity p_1^T and p_2^T based on equation (2.35) and (2.36).

$p_1^T = -\Phi^{-1}(-ET^{-1}A + C)$

$p_2^T = -T^{-1}(Am^T + B p_1^T)$

The size of matrix $H = 120 \times 240$

The value of p_1, p_2, m and codeword $c = [m, p_1, p_2]$ are given below:

- The size of parity p_1 is 1×16.

p_1 = [1 1 1 1 0 1 1 1 0 0 1 0 0 0 1 1].

- The size of parity p_2 is 1×104.

p_2 = [1 1 0 1 0 1 0 1 0 0 0 0 1 0 1 1 1 1 0 0 1 0 1 0 0 1 0 1 1 0 0 1 1 1 0 0 1 1 0 1 0 1 1 0 0
1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 1 1 1 1 0 1 0 0 0 0 0 1 0 0 1 1 0 1 1 1 1 1 1 0 0 0 1
1 1 0 1 0 0 0 1 0 1 0 1 1 0].

- The size of random information bit m is 1×120.

m = [0 0 1 0 1 1 0 1 1 1 0 1 1 0 0 0 1 1 1 0 1 1 1 1 1 0 0 0 0 0 1 1 0 0 0 0 1 1 0 1 1 0 0 1 0
0 1 1 1 1 1 1 0 1 0 0 1 1 0 0 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 0 0 1 1 0 0 0 0 1 1
0 1 1 1 0 1 0 1 0 1 1 1 1 0 0 0 0 1 1 0 1 0 0 1 0 0 0 1 0 1].

- The size of codeword $c = [m, p_1, p_2]$ is 1×240.

$c = [m, p_1, p_2]$

c = [0 0 1 0 1 1 0 1 1 1 0 1 1 0 0 0 1 1 1 0 1 1 1 1 1 0 0 0 0 0 1 1 0 0 0 0 1 1 0 1 1 0 0 1 0
0 1 1 1 1 1 1 0 1 0 0 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 1 1 0 0 0 0 1 1
0 1 1 1 0 1 0 1 0 1 1 1 1 0 0 0 0 1 1 0 1 0 0 1 0 0 0 1 0 1 1 1 1 1 0 1 1 1 0 0 1 0 0 0 1
1 1 1 0 1 0 1 0 1 0 0 0 0 1 0 1 1 1 1 0 0 1 0 1 0 0 1 0 1 1 0 0 1 1 1 0 0 1 1 0 1 0 1 1 0
0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 1 1 1 1 0 1 0 0 0 0 0 1 0 0 1 1 0 1 1 1 1 1 1 1 0 0 0
1 1 1 0 1 0 0 0 1 0 1 0 1 1 0].

In order to validate the codeword c, we use equation (2.9) that is $\boldsymbol{c}\, H^T = 0$. If c is a valid codeword, the result of $\boldsymbol{c}\, H^T$ will be zero matrix of size of 1×120.

3.4.2 Encoding Complexity

Encoding complexity gives a description on how many multiplications, additions and logical operations are needed in encoding process of the proposed code-construction. When encoding complexity is linear, it implies that the complexity grows linearly with the code length (N) and is said to be computationally most efficient.

Since the proposed code-construction uses actual encoding step from Richardson-Urbanke method, encoding complexity of the proposed code is almost linear given by $O(N + g^2)$. This complexity is derived from computation of p_1^T in Table 2.1.

In this section, we will introduce the notation of g which describes a gap of matrix H_{par} into its lower triangular form. Since the value of g relates to encoding complexity of the proposed LDPC code, the smaller the value of g the lower is its encoding complexity.

The value of g in the third stage of code-construction is given below based on Figure 3.14.

$$g = 2 \times L \times L_1 \tag{3.22}$$

Encoding complexity of the proposed (3, k) regular and irregular LDPC code with the value of g^2 is described as follow:

$$O(N + g^2) = O(N + 4 \times L^2 \times L_1^2) \quad (3.23)$$

Example:

An example of how to get encoding complexity of the proposed (3, k) regular LDPC code is described below. The proposed (3, 39) regular LDPC code uses parameter $L_1 = 1$, $L = 9$, $base = 102$, $j_i = 3$, $k_i = 36$ and deterministic RCS $(x \times (y - 1)) \bmod L$ base.

$g = 2 \times L \times L_1 = 18$.

$N = Base \times L \times L_1 \times (j_i + k_i)) \div j_i = 11934$.

Since $g = 18$ and $N = 11934$, the value of g can be seen to be $g << N$.

The value of g^2 is 324 bits which is equal to 0.0271 N.

Therefore, encoding complexity is $O(N + g^2) = O(1.0271\ N)$

3.5 Decoding

This section discusses decoding used in the proposed code-construction. Decoding algorithm of the proposed codes applies sum product algorithm (SPA) as the decoding standard in LDPC codes. Since the focus of this work is encoding, we do not give more detail about SPA as already explained in section 2.2.9.

3.6 Summary

The proposed code-construction for (3, k) regular, as well as, irregular LDPC codes, has been presented consisting of three stages of code-construction. The first stage of code-construction builds parity part with an approximate lower triangular form that utilizes two variables (parameter *base* and expansion factor L) and two deterministic parameters (parameter $j_p = 3$ and parameter $k_p = 3$). The basic design philosophy of the proposed

parity part in the first stage of code-construction is to build a triple diagonal base matrix, meeting the requirement of 3 ones in each column and each row while minimizing girth of 4-cycles, and to expand it with expansion factor L. The parity part of the proposed (3, k) regular and irregular LDPC codes is same, of (3 × 3) size.

The second stage of code-construction builds information part that employs three parameters (expansion factor Z, parameter j_i and parameter k_i). The core design of the second stage of the proposed code-construction builds information part of the matrix H using deterministic base matrix with all elements of ones and expands it with expansion factor Z. After the construction of information part is done, we combine parity part in the first stage of code-construction with information part to get the proposed LDPC code in the second stage of code-construction denoted by matrix $H_1 = [H_i \mid H_p]$.

The purpose of third stage of code-construction achieves matrix $H = [H_{inf} \mid H_{par}]$ after expanding matrix $H_1 = [H_i \mid H_p]$ in the second stage of code-construction with identity matrix without any cyclic shift to build a longer code length than in the first and the second stage of code-construction.

The construction of the third stage is the same for (3, k) regular LDPC and irregular LDPC codes. The third stage of code-construction is determined by expansion factor L_1. Therefore, three stages of code-construction make use of eight parameters namely, expansion factor Z, expansion factor L_1, parameter j_i, parameter k_i, parameter j_p and parameter k_p.

Encoding complexity of the proposed (3, k) regular and irregular LDPC code is almost linear by $O(N + g^2)$ where $g << N$. Notation of g describes a gap in matrix H_{par} in its lower triangular form. Since the value of g relates to encoding complexity of the proposed LDPC code, the smaller the value of g the lower is its encoding complexity. The value of g in the third stage of code-construction is given by $g = 2 \times L \times L_1$.

CHAPTER 4

SYSTEM SIMULATION

4.1 Introduction

The proposed code-construction has been described in chapter 3. It overcomes general issues in the design of regular LDPC codes like rank-deficiency of matrix *H*, pre-processing before encoding and high encoding complexity. The advantage of the proposed code-construction is that the parity part of the design can be used not only for $(3, k)$ regular but also irregular LDPC codes.

This chapter presents code performance, simulation model and error performance conducted in this thesis as instruments to validate the proposed construction. The performance results of the proposed LDPC codes using this system simulation will be obtained and discussed in the next chapter.

4.2 Code Performance

Code performance of the proposed LDPC codes designed using the proposed code-construction method is measured in terms of the following:

- *Bit error rate (BER).*

  BER is the probability that a decoded information bit is in error, while signal-to-noise ratio (SNR) describes the ratio of energy-per-information bit at the input to

one sided noise power spectral density at the receiver denoted by E_b/N_0 and usually expressed in decibels (dB). A reliable communication is achieved by getting the BER performance as low as possible under certain system constraints, such as power constraint, bandwidth constraint, or low encoding and decoding complexity constraint. This measure is used to evaluate the performance of both (3, k) regular LDPC codes and also irregular code.

Since there are no theoretical BER of LDPC codes, we use theoretical BER of un-coded binary phase-shift keying (BPSK) for lower bound and Shannon's limit as the upper bound for comparison sake. Hereunder, we give theoretical upper bound on the BER of [n, k] binary block code over AWGN channel. It is, however, recognized that the LDPC codes outperform these linear block codes for large number of information bits and low values of $d_{\min}$.

Theoretical un-coded BPSK on the BER (P_b) over AWGN channel is given below [54]:

$$P_b = \quad Q(\sqrt{(2 \times E_b/N_0)}) \tag{4.1}$$

In general, notation of block code is represented by [n, k]. In order to be consistent with notation in this thesis, the notation of [n, k] is replaced by [N, K]. Theoretical upper bound on the BER (P_b) of an [N, K] binary block code with soft-decision decoding and BPSK modulation over AWGN channel is described below [54]:

$$P_b \leq \ ½ \times (2^K - 1) \times Q(\sqrt{(2 \times E_b/N_0 \times R \times d_{min})}) \tag{4.2}$$

- *Block error rate (BLER) or frame error rate (FER) or word error rate (WER).*

  BLER is the probability that a decoded codeword is in error. A decoded codeword consists of a decoded information bit and a decoded parity bit. It is generally required that the BLER be as small as possible in order to get a reliable communication. This performance measure is used for (3, k) regular LDPC codes.

Since there are no theoretical BLER of LDPC codes, we use theoretical BLER of un-coded binary phase-shift keying (BPSK) that is the same as equation (4.1) and theoretical upper bound on the BLER of [n, k] binary block code over AWGN channel. In general, notation of block code is represented by [n, k]. In order to be consistent with notation in this thesis, the notation of [n, k] is replaced by [N, K].

Theoretical upper bound on the BLER (P_{block}) of an [N, K] binary block code with soft-decision decoding and BPSK modulation over AWGN channel is given below [54]:

$$P_{block} \leq (2^K - 1) \times Q(\sqrt{(2 \times E_b/N_0 \times R \times d_{min}}) \quad (4.3)$$

- *Comparison with Shannon limit.*

  As per famous Shannon's coding theorem, as long as the data rate in bits/s is less than the theoretical capacity of the communication system, a channel code can be so designed that the system will have arbitrarily small probability of error. As an extension of this theorem, Shannon limit is defined as the theoretical limit on minimum SNR required for a coded system at a given code rate (R) to achieve arbitrarily small probability of error only if the SNR exceeds this limit. Generally, this limit is used as a yardstick to evaluate the performance of codes.

  Shannon limit is also used to measure the maximum achievable coding gain for a coded system with a given code rate (R) over un-coded system with the same modulation signal set.

  For example to achieve a BER of 10^{-5}, an un-coded BPSK system requires an SNR of 9.65 dB and a coded BPSK system with code rate R = ½ has the value of Shannon limit 0.188 dB. Therefore, the maximum achievable (potential) coding gain for coded BPSK with a code rate R = ½ is 9.462 dB [3].

Table 4.1 gives the value of Shannon limit for various code rate (R) with binary input, AWGN channel and binary phase-shift keying (BPSK) modulation as given in [3]. The Shannon limit performance is also evaluated only for (3, k) regular LDPC codes.

Table 4.1: Shannon limit with AWGN channel, BPSK and binary input [3].

Code Rate (R)	E_b/N_0 (dB)	Code Rate (R)	E_b/N_0 (dB)	Code Rate (R)	E_b/N_0 (dB)	Code Rate (R)	E_b/N_0 (dB)
0.3	-0.616	0.49	0.144	0.68	1.143	0.846	2.503
0.31	-0.579	0.5	0.188	0.69	1.208	0.855	2.6
0.32	-0.544	0.51	0.233	0.70	1.275	0.857	2.62
0.33	-0.507	0.52	0.279	0.71	1.343	0.875	2.84
0.34	-0.469	0.53	0.326	0.72	1.412	0.888	3.05
0.35	-0.432	0.54	0.374	0.73	1.483	0.9	3.2
0.36	-0.394	0.55	0.424	0.74	1.554	0.909	3.34
0.37	-0.355	0.56	0.474	0.75	1.628	0.916	3.47
0.38	-0.314	0.57	0.526	0.76	1.708	0.923	3.59
0.39	-0.276	0.58	0.574	0.77	1.784	0.928	3.7
0.4	-0.236	0.59	0.628	0.78	1.867	0.933	3.8
0.41	-0.198	0.6	0.682	0.79	1.952	0.937	3.91
0.42	-0.156	0.61	0.734	0.8	2.045	0.941	3.98
0.43	-0.118	0.62	0.791	0.807	2.108	0.944	4.05
0.44	-0.074	0.63	0.844	0.817	2.204	0.947	4.1
0.45	-0.032	0.64	0.904	0.822	2.25	0.95	4.2
0.46	0.01	0.65	0.96	0.827	2.302	0.952	4.26
0.47	0.055	0.66	1.021	0.833	2.36	0.954	4.3
0.48	0.097	0.67	1.084	0.837	2.402	0.956	4.38

- *Complexity*

  Complexity is a measure of how many multiplications, additions and logical operations are needed in a given design. An encoding efficient LDPC encoder

should have as low computational load as possible. When the complexity is linear, it implies that the complexity grows linearly with the size of the data. In the design of encoder/decoder, designs with linear complexity are said to be, computationally, most efficient, as it implies that the computation grows linearly with the length of the code.

Since the proposed (3, *k*) regular LDPC codes have low density of ones in *H*, the sparseness of proposed *H* lends to low decoding complexity. Therefore, complexity analysis presented in chapter 5 mainly focuses in encoding portion. The complexity analysis for irregular LDPC codes has been excluded from the scope.

Complexity analysis starts by exploring pre-processing step in the proposed (3, *k*) regular LDPC codes based on Richardson-Urbanke method and is continued for comparison of pre-processing step of encoding followed by encoding complexity of other regular QC LDPC codes. This section ends with listing the computational time of actual encoding step in the proposed codes that also compares the time of encoding between the proposed (3, *k*) regular with other regular QC LDPC code and regular repeat accumulate (RA) LDPC.

4.3 Simulation Model

This section gives an overview of the simulation model which is used to measure the performance parameters of section 4.2. It assumes binary digits input, binary phase shift keying (BPSK) modulation and AWGN channel used in LDPC encoder-decoder of MATLAB® 7.4. Encoding method used in the encoder is based on matrix inversion of matrix *H* while decoding algorithm used in the decoder of our simulation is sum product algorithm (SPA). The encoder is supplied with the parity-check matrix separately built. Encoding by parity check matrix is based on Richardson-Urbanke method.

The block diagram of the simulation procedure of our code construction is described in Figure 4.1. Input to simulation is information bits represented by random binary bits. The length of random binary bits depends on the value of code rate (R) applied in the system.

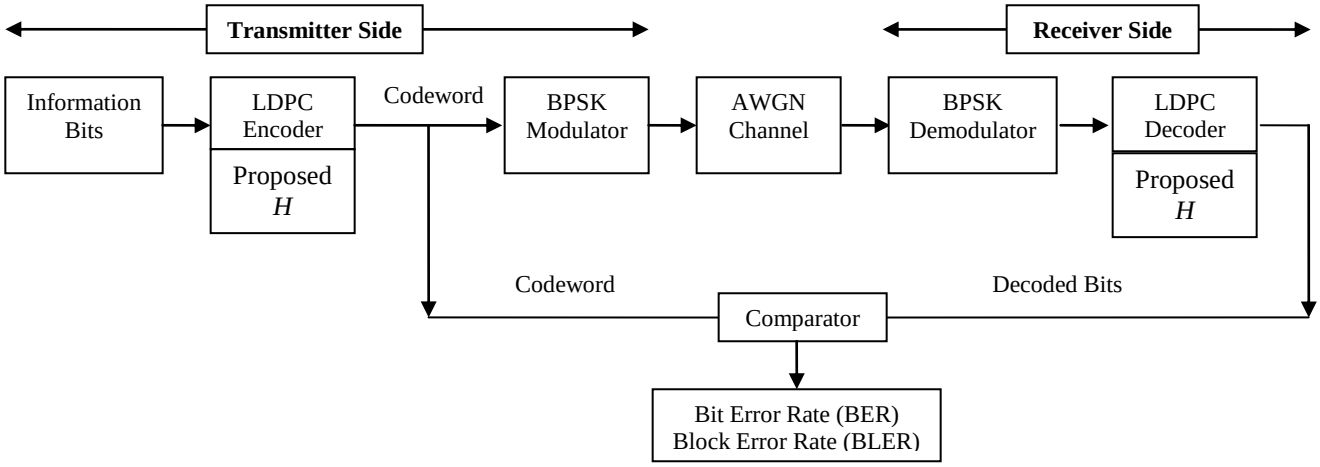

Figure 4.1: Simulation block diagram

Inserting certain pattern of redundancy into random binary bits is done in LDPC encoder that produces codeword. Addition of redundancy in terms of parity bits is used for controlling transmission errors by detecting and correcting error in the transmitted data stream without requesting retransmission of information bits.

After passing through LDPC encoder, the codeword is modulated by BPSK modulator and added with white Gaussian noise known as AWGN channel. After going through AWGN channel in the receiver side, the received codeword is demodulated by BPSK demodulator and decoded by LDPC decoder. LDPC decoder reconstructs information bits based on type of redundancy used in the encoder, and chooses the one closest to the noisy received codeword. Decoding algorithm utilized in LDPC decoder is sum product algorithm (SPA), also known as ‘belief propagation’.

4.4 Error Performance Conducted in the Thesis

In order to validate the proposed LDPC codes, the error performance is measured in term of bit error rate (BER), block error rate (BLER) and Shannon limit as follows:

- Bit error rate (BER) performance

  BER performance of this thesis is achieved by evaluating the performance of both (3, *k*) regular LDPC codes and also irregular code. The BER of proposed (3, *k*) regular LDPC codes is compared with published results of (3, *k*) regular QC LDPC and (3, *k*) regular random LDPC codes in low code rate ($R < 0.5$) and high code rate ($R > 0.7$). Mean while, the BER of proposed irregular LDPC code is also compared with published results of irregular LDPC codes.

  - Results of the proposed (3, *k*) regular LDPC codes

    This section investigate some behaviors related with the proposed (3, *k*) regular LDPC codes.

  - Comparison of proposed (3, *k*) regular LDPC codes with published (3, *k*) regular QC LDPC codes.
    - (3, 5) regular QC LDPC codes [11].

      These codes are chosen as they have low code rates ($R = 0.4125$ and $R = 0.4029$).
    - (3, 12) regular QC LDPC codes [48].

      These codes are chosen since they have code rate $R = 0.75$.

  - Comparison of proposed (3, *k*) regular LDPC codes with published (3, *k*) regular random LDPC codes.
    - (3, 5) regular random LDPC codes [11].

      These regular codes give low code rate $R = 0.4$
    - (3, 24) regular random LDPC codes [19].

      This regular random LDPC code produces high code rate $R = 0.875$.

- Comparison of proposed irregular LDPC codes with published irregular codes
  - Irregular LDPC codes using dual diagonal parity [53].
  - Irregular random LDPC codes using density evolution by Richardson et. al [53].
  - Irregular random LDPC codes by Mackay [53].
  - The performance result of the proposed irregular code is also compared with (64, 64) regular cyclic LDPC code [53] in order to show that the proposed design outperforms it even though the proposed irregular code has lower code rate and lower code length.

- Block error rate (BLER) performance

  BLER performance of this work is conducted by comparing the BLER of proposed (3, k) regular LDPC codes with published results of (3, k) regular QC LDPC and (3, k) regular random LDPC codes in high code rate ($R > 0.7$).

  BLER performance is determined by two following sections:

  - Comparison with published (3, 13) regular QC LDPC codes

    These codes give high code rate of BLER performance ($R = 0.769$) based on M. P. C. Fossorier [10].

  - Comparison with published (3, 13) regular random LDPC codes

    These codes produce high code rate of BLER performance ($R = 0.769$) based on M. P. C. Fossorier [10].

- Comparison with Shannon limits

  Shannon limit performance in this thesis is obtained in terms of BER performance that is evaluated in high code rate ($R \geq 0.875$). The choice of high code rate ($R \geq 0.875$) is taken since the proposed (3, k) regular LDPC codes are not able to outperform the other regular LDPC in low code rate performance.

Comparison of Shannon limit in term of BER performance of published QC and cyclic LDPC codes with the same value of code rate is also evaluated in this section. Published QC and cyclic LDPC codes are chosen since those codes are also classified into structured LDPC codes and the most related to the proposed code-construction.

Shannon limit of the proposed (3, k) regular LDPC codes in $R \geq 0.875$ are compared with published QC and cyclic LDPC codes at the same code rate.

- High code rate of BER performance of regular QC LDPC code (R = 0.875) derived from Shu Lin and Daniel J. Costello [1].

- High code rate of BER performance of regular cyclic LDPC codes (R = 0.875, R = 0.9, R = 0.916, R = 0.923 and R = 0.928) based on Yu Kou, Shu Lin and M. P. C. Fossorier [13].

The proposed design method has been used to obtain viable choices of design parameters such as *base* and expansion factor L that lead to viable matrix H with no rank deficiency of matrix H, no girth of 4-cycles and no singularity. This has been done for parameter *base* $\leq$ 114 and expansion factor $L \leq 35$.

The proposed (3, k) regular LDPC codes is limited by the size of H = 1026 × 14364 with code length N = 14364 bits that is inverted in the MATLAB® 7.4 encoder. Hence, the results obtained in this thesis are limited to code- length N < 15000 bits.

4.5 Summary

The code performance of the proposed (3, k) regular LDPC code is measured in terms of BER, BLER and Shannon limit. BER performance of this thesis is achieved by evaluating the performance of both (3, k) regular LDPC codes and also irregular code.

The BER performance for (3, k) regular LDPC codes explores the behavior of the proposed (3, k) regular LDPC codes and evaluates the proposed (3, k) regular LDPC compared with published results of (3, k) regular QC LDPC codes and (3, k) regular random LDPC codes in low code rate and high code rate. Mean while, the BER of proposed irregular LDPC code is also compared with published results of irregular LDPC codes.

BLER performance evaluates the proposed (3, k) regular LDPC codes with published results of (3, k) regular QC LDPC and (3, k) regular random LDPC codes in high code rate. Shannon limit performance in this thesis is measured in term of BER performance evaluated at high code rate ($R \geq 0.875$). This performance is compared with the published QC and cyclic LDPC codes for the same code rate.

Complexity analysis is made based on complexity analysis of encoding of the proposed (3, k) regular LDPC code excluded irregular code. The complexity is also compared complexity of encoding with that of other designs like QC LDPC code and regular repeat accumulate (RA) LDPC having linear encoding complexity.

The simulation model used is developed with binary digits input, BPSK modulation and AWGN channel used in LDPC encoder-decoder of MATLAB® 7.4. Encoding method used in the encoder is based on matrix inversion of matrix H while decoding algorithm used in the decoder of our simulation is sum product algorithm (SPA).

The proposed design method has been used to obtain viable choices of design parameters such as *base* and expansion factor L that lead to viable matrix H with no rank deficiency of matrix H, no girth of 4-cycles and no singularity. This has been done for parameter $base \leq 114$ and expansion factor $L \leq 35$.

The proposed (3, k) regular LDPC codes is limited by the size of $H = 1026 \times 14364$ with code length $N = 14364$ bits that is inverted in the MATLAB® 7.4 encoder. Hence, the results obtained in this thesis are limited to code- length $N < 15000$ bits.

CHAPTER 5

RESULTS, ANALYSIS AND DISCUSSION

5.1 Introduction

An overview of code performance and system simulation has been described in the last chapter whereas the construction of the proposed code itself is reported in chapter 3. This chapter presents some results, analysis and discussion of the proposed codes in terms of BER performance, BLER performance, Shannon limit performance and complexity analysis.

Code performance of (3, *k*) regular LDPC codes in section 5.2 presents BER and BLER performance of the proposed codes and compares them with other regular LDPC codes such as quasi cyclic (QC), and regular random at both high and low code rate. The limiting performance is also obtained and compared with the Shannon limit and those of a selected few codes at high code rate ($R \geq 0.875$). The proposed (3, *k*) regular LDPC codes are also compared with theoretical upper bound of an [N, K] binary block code with soft-decision decoding and BPSK modulation.

Code performance of irregular LDPC codes is given in section 5.3 that presents BER performance of the proposed irregular LDPC code compared with other irregular LDPC codes.

Discussion of complexity analysis is given in section 5.4 that begins by exploring pre-processing step in the proposed (3, *k*) regular LDPC codes based on Richardson-Urbanke

method. The complexity analysis for irregular LDPC codes has been excluded from the scope. The discussion of complexity analysis is continued for comparison of pre-processing step of encoding followed by encoding complexity with (3, 5) regular QC LDPC codes. This section is ended by computational time of actual encoding step in the proposed codes that includes a description of encoding complexity in the proposed codes and also compares the time of encoding between the proposed (3, 6) regular with (3, 6) regular QC LDPC code.

5.2 Code Performance of (3, *k*) Regular LDPC Codes

According to chapter 3, there are five variables used for constructing (3, *k*) regular LDPC codes, namely, j_i, k_i, *base*, L and L_1. In this work, parameter j_i is the key factor that defines whether parity check matrix (H) is categorized into regular or irregular LDPC codes. Since this thesis mainly evaluates the performance of (3, *k*) regular LDPC codes, the value of parameter j_i used in this chapter is $j_i = 3$.

Simulation model of the proposed code-construction is developed based on procedures in chapter 4 which evaluates the proposed (3, *k*) regular LDPC codes using LDPC encoder-decoder of MATLAB® 7.4. The block diagram of the simulation of the proposed code-construction is described in Figure 4.1.

This section is divided into three sections that present the evaluation and comparison of the proposed (3, *k*) regular LDPC codes in terms of bit error rate (BER), block error rate (BLER) and Shannon limit. Comparison with other published (3, *k*) regular LDPC codes is presented in each of those sections. All the codes presented in this section are having no rank-deficiency of matrix H, no pre-processing step of encoding, no girth of 4-cycles, low encoding complexity and non-singular parity part (H_{par}).

The value of girth in the proposed matrix H is at least girth 6-cycles since girth 4-cycles in the proposed codes is eliminated by the choice of parameter *base* and expansion factor L applied in the design of parity part.

Based on [1], [13], the minimum distance (d_{min}) of (j, k) regular LDPC code is at least $j + 1$. Therefore, the minimum distance (d_{min}) of the proposed (3, k) regular LDPC codes with $j = 3$ is at least 4 that gives the ability of correcting at least 1 error.

5.2.1 Bit Error Rate (BER) Performance

As stated in chapter 4, bit error rate (BER) is the probability that a decoded information bit is in error. In this section, code performance is evaluated in term of bit error rate (BER) versus signal to noise ratio (SNR) curves. BER performance in this section is obtained by simulating across 200 errors at each point of SNR. After getting BER value at each point of SNR, linear interpolation is applied.

BER performance is categorized into three sub-sections. The first sub-section describes BER results of the proposed (3, k) regular LDPC codes while the last two sub-sections report the BER comparison of the proposed (3, k) regular with other (3, k) regular quasi-cyclic (QC) and (3, k) regular random LDPC codes.

5.2.1.1 Results of the Proposed (3, k) Regular LDPC Codes

Derived from section 5.2.3 and 5.2.4, the choice of parameter *base* and expansion factor L with given method of deterministic right-cyclic shift (RCS) impact the pre-processing step of encoding. This section would investigate the impact of these parameters and the methods of deterministic RCS on the code performance. The number of iterations used in this section is set to 50 iterations of sum product decoding algorithm (SPA).

- *Code performance and its dependence on variable k_i.*
  The influence of variable k_i on the code performance in (3, k) regular LDPC is obtained by varying k_i while keeping all other variables constant. This is shown in

Figure 5.1 where variable k_i = {1, 2, 3}, while j_i = 3, *base* = 30, L = 6, L_1 = 1 and deterministic RCS is given by $(x \times (y - 1)) \bmod L$.

Derived from equation (3.24), the value of code rate (R) is influenced by parameter k_i. Therefore, varying k_i leads to varying code rate. Parameter k_i = {1, 2, 3} produces code rate R = {0.25, 0.4, 0.5}. According to equation (3.22), varying k_i = {1, 2, 3} while keeping other variables constant gives varying code length. Parameter k_i = {1, 2, 3} gives code length N = {240, 300, 360}. Figure 5.1 shows three curves based on three variables of parameter k_i = {1, 2, 3} while keeping other variables constant. It is shown that circle marked curve gives the best BER performance since this curve has the longest value of code length N.

- *Code performance and its dependence on variable L*.
  The influence of variable L on the code performance in (3, k) regular LDPC is obtained by varying expansion factor L while keeping other variables constant. This condition is reported in Figure 5.2 where expansion factor L = {6, 10, 30, 25}, while L_1 = 1, *base* = 18, k_i = 2, deterministic RCS = $(x \times (y - 1)) \bmod L$ with the same code rate R = 0.4.

  Based on equation (3.22), varying expansion factor L while keeping other variables constant yields varying code length N. Expansion factor L = {6, 10, 25, 30} yields code length N = {150, 300, 750, 900}.

  Figure 5.2 presents four curves derived from four variable expansion factor L = {6, 10, 25, 30} while keeping other variables constant. It can be seen that the longer the value of N, the better the performance. Therefore, curve having N = 900 gives the best performance of all in Figure 5.2.

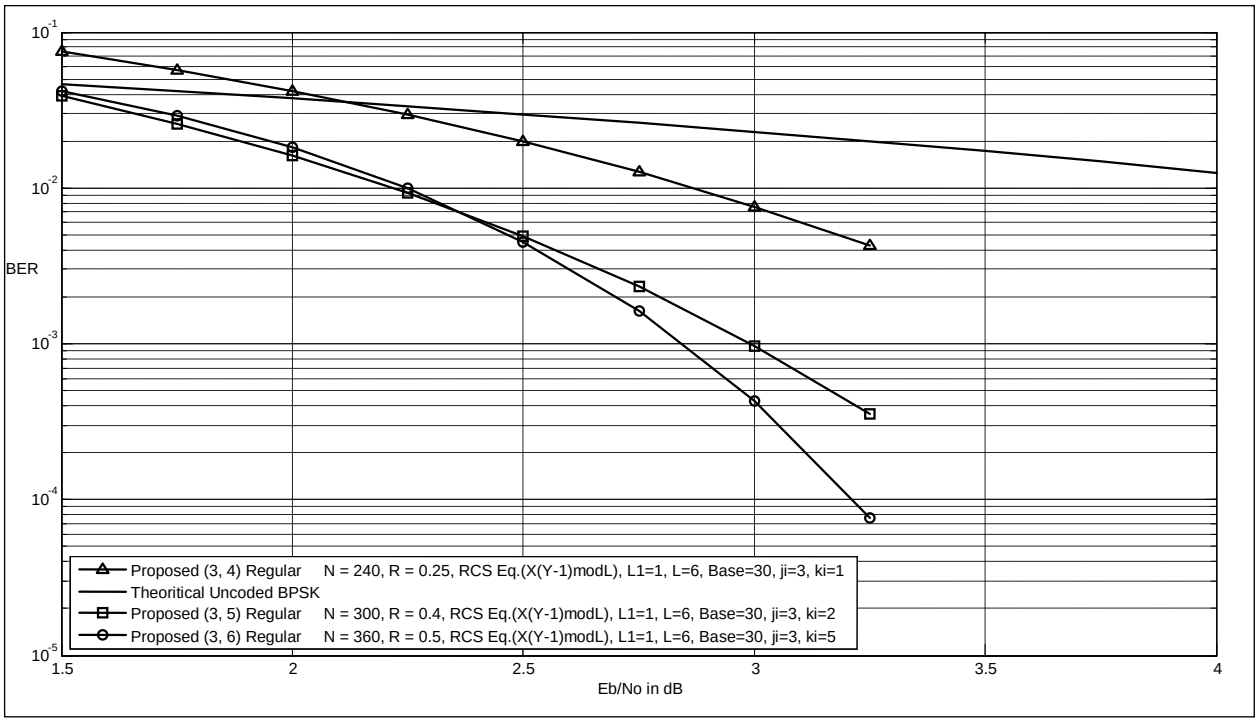

Figure 5.1: (3, k) Regular LDPC with k_i = {1, 2, 3}, *base* = 30, L = 6 and L_1 = 1.

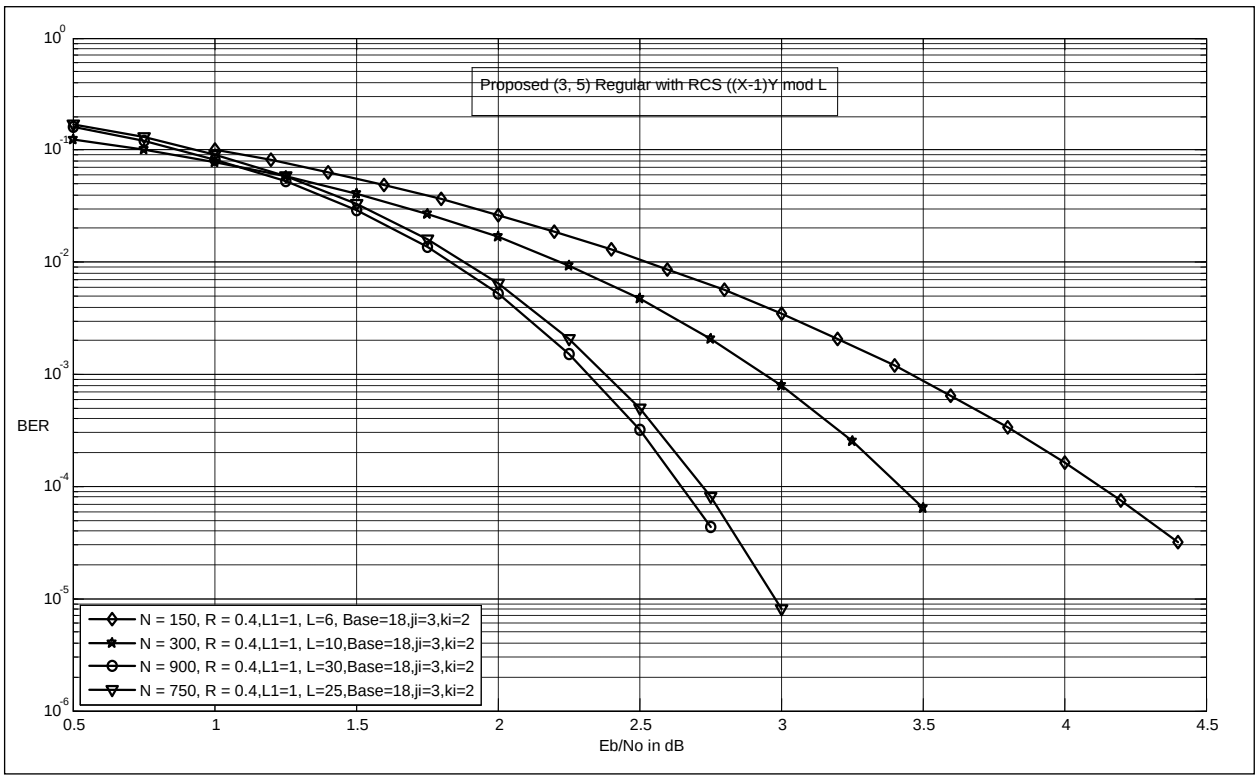

Figure 5.2: (3, 5) Regular LDPC with variable L = {6, 10, 25, 30}

Based on chapter 3, encoding complexity of the proposed code-construction is determined by the value of g. The value of g is influenced by expansion factor L and L_1 since g is $2 \times L \times L_1$. At the same time, increasing expansion factor L while keeping other variables constant yields encoding complexity that is also becoming higher and more complex. Encoding complexity is reduced by applying as small as possible expansion factor L and L_1 while utilizing as big as possible parameter *base*.

- *Code performance and its dependence on parameter base and L*.

  The impact of parameter *base* and L on the code performance of (3, k) regular LDPC codes is examined in Figure 5.3 by applying two different parameters of *base* and L while keeping other variables constant. Figure 5.3 shows three curves that maintain parameters $j_i = 3$ and $k_i = 27$, and produce the same value of code rate (R) but yield variable code length (N).

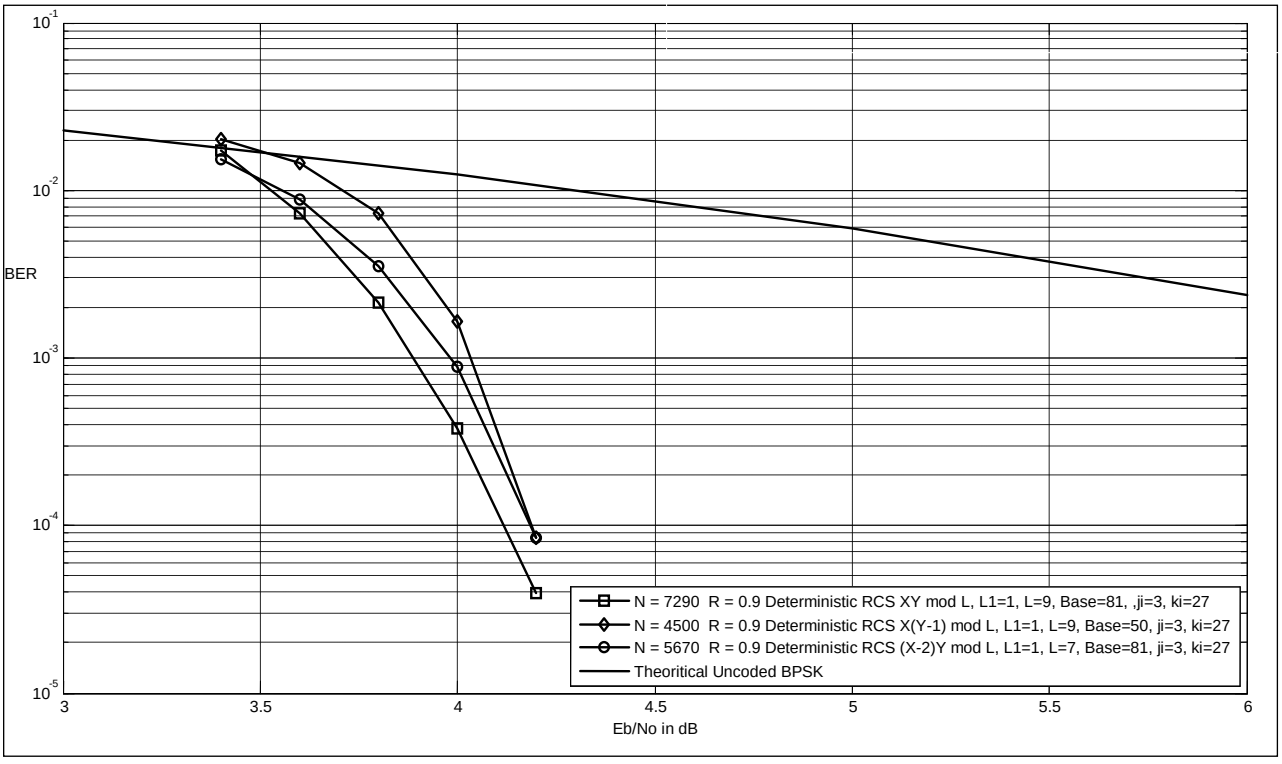

Figure 5.3: Proposed (3, 30) regular LDPC codes with variable *base* and L.

Parameter difference between curve marked with square legend and that with diamond in Figure 5.3 is in the value of parameter *base*. Similarly the difference between the square marked curve and circle marked curve is in the value of expansion factor *L*. It is shown in Figure 5.3 that for the same code rate $R = 0.9$ the longer the value of *N*, the better the performance for the proposed (3, 30) regular LDPC codes. Since there will be no difference in the BER performance, different RCS are used in Figure 5.3. This condition will be explained in the next section.

- *Code performance and its dependence on the method of RCS*.
  The influence of some methods of deterministic RCS in the BER performance is shown in Figure 5.4 and Figure 5.5. All variables in Figure 5.4 and Figure 5.5 are kept constant except the method of deterministic RCS.

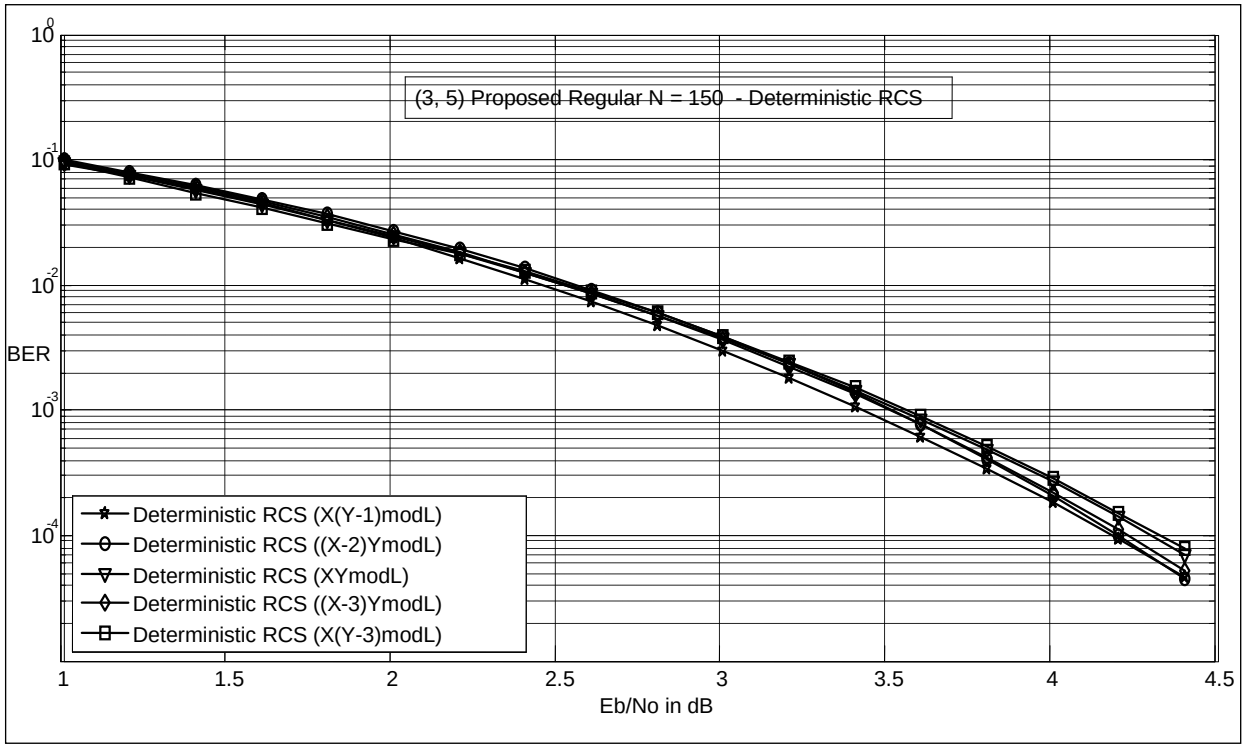

Figure 5.4: (3, 5) Regular LDPC with deterministic RCS and $N = 150$.

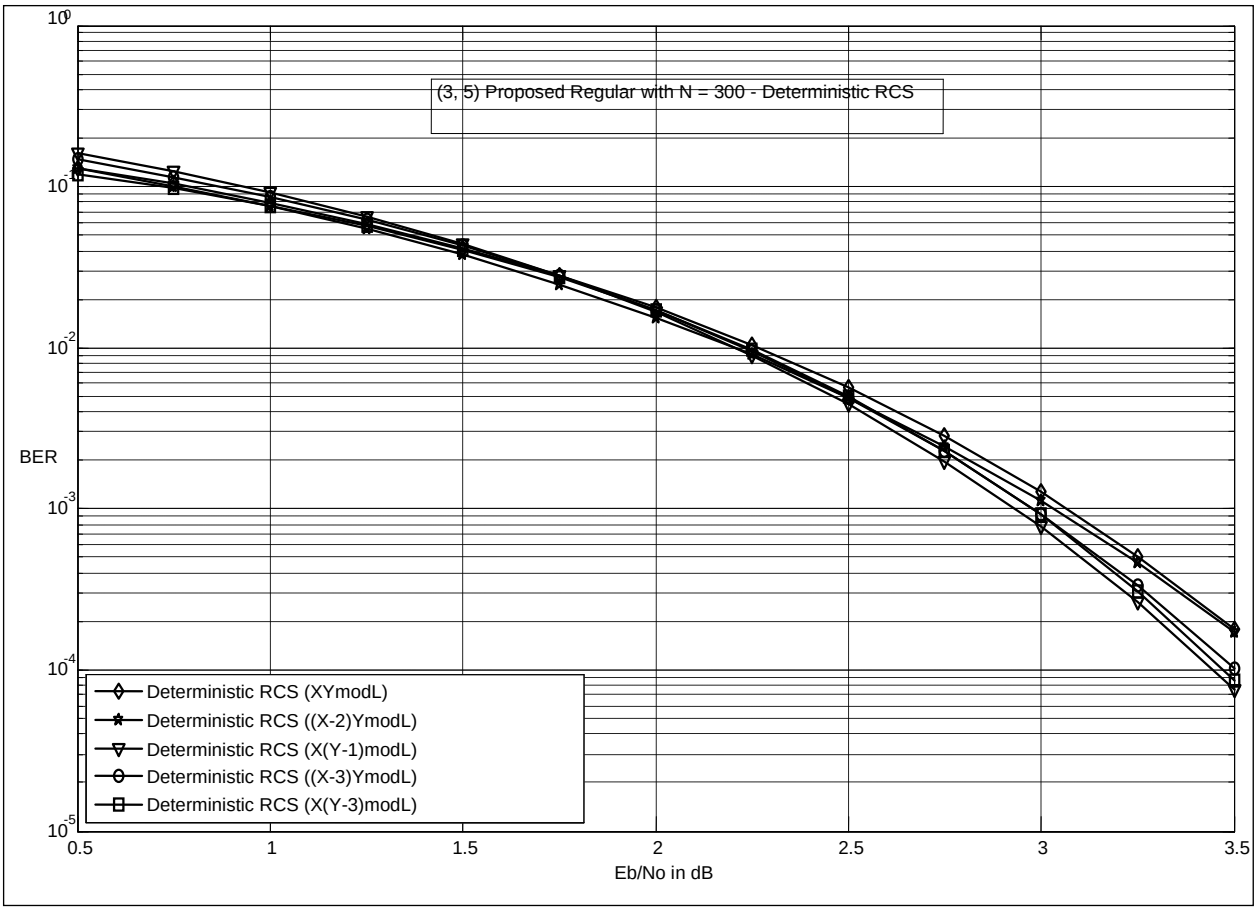

Figure 5.5: (3, 5) Regular LDPC with deterministic RCS and N = 300.

Figure 5.4 utilizes parameter L_1 = 1, L = 3, *base* = 30, j_i = 3, k_i = 2, R = 0.4 and it compares the five methods of deterministic RCS given by {$(x \times y)$ mod L, $((x-2) \times y)$ mod L, $((x-3) \times y)$ mod L, $(x \times (y-1))$ mod L, and $(x \times (y-3))$ mod L}. These five methods of deterministic RCS are chosen since all of them use *base* = 30 without column permutation and Gauss elimination as listed in Table 5.5. Since five curves in Figure 5.4 use the same value of code length N, the BER performance of five curves is having a comparable performance.

Parameters used in Figure 5.5 are $L_1 = 1$, $L = 6$, $base = 30$, $j_i = 3$, $k_i = 2$, $R = 0.4$ while five methods of deterministic RCS are $(x \times y) \bmod L$, $((x - 2) \times y) \bmod L$, $((x - 3) \times y) \bmod L$, $(x \times (y - 1)) \bmod L$ and $(x \times (y - 3)) \bmod L$. The same situation of comparable performance using five methods of deterministic RCS can be seen also in Figure 5.5 even though there is a slightly difference in the BER performance. In this Figure, all of the five curves utilize the same value of code length $N = 300$ bits.

According to Figure 5.1 to Figure 5.5, following are some conclusions that can be drawn about the behavior of the proposed (3, k) regular LDPC codes:

1. The longer the value of code length N, the better the BER performance.
2. Variable code rate (R) is achieved by applying variable value of k_i.
3. Changing one parameter while keeping the other variables constant generates variable value of code length (N).
4. Encoding complexity is reduced by applying as small as possible parameter L and L_1 while utilizing as large as possible parameter $base$.
5. The comparison of code performance will be fair if all curves have the same code length (N) and code rate (R) even though they are achieved by applying different parameter of j_i, k_i, $base$, L and L_1.
6. The choice of deterministic RCS yields a comparable BER performance for all methods of deterministic RCS. The comparison of deterministic RCS in terms of complexity will be discussed in the section 5.4.

5.2.1.2 Comparison with (3, k) Regular Quasi Cyclic (QC) LDPC Codes

This section presents the comparison of the proposed (3, k) regular with (3, k) quasi cyclic (QC) LDPC codes for low code rate ($R = 0.4125$ and $R = 0.4029$) and high code rate ($R = 0.75$).

- *(3, k) regular QC LDPC codes in low code rate (*R = 0.4125 *and* R = 0.4029*).*
  Low code rate regular QC LDPC codes are constructed according to [11] presented by (3, 5) regular QC LDPC codes in Figure 5.6 and Figure 5.7.

  Figure 5.6 compares BER performance of (3, 5) regular QC LDPC code having R = 0.4125 and N = 155 to the proposed (3, 5) regular LDPC having R = 0.4 and N = 150 with 50 iterations applied in sum product decoding algorithm (SPA). The proposed regular LDPC codes in Figure 5.6 utilize parameter L_1 = 1, L = 3, *base* = 30, j_i = 3, k_i = 2 with two different deterministic RCS $(x \times (y - 1)) \bmod L$ and $(x \times (y - 3)) \bmod L$.

  Figure 5.6 shows that BER performance of the proposed (3, 5) regular LDPC code is not better than that of (3, 5) regular QC LDPC code when the value of code rate $R < 0.5$.

  Moreover, the value of code rate and code length between the proposed (3, 5) regular and (3, 5) regular QC LDPC in Figure 5.6 are not the same. As mentioned in [11], the matrix H of (3, 5) regular QC LDPC has linear dependency among its rows that causes erasure of rows. Hence, there is a code rate gain ($R > 0.4$) that results in a better waterfall performance than the proposed regular LDPC codes in Figure 5.6

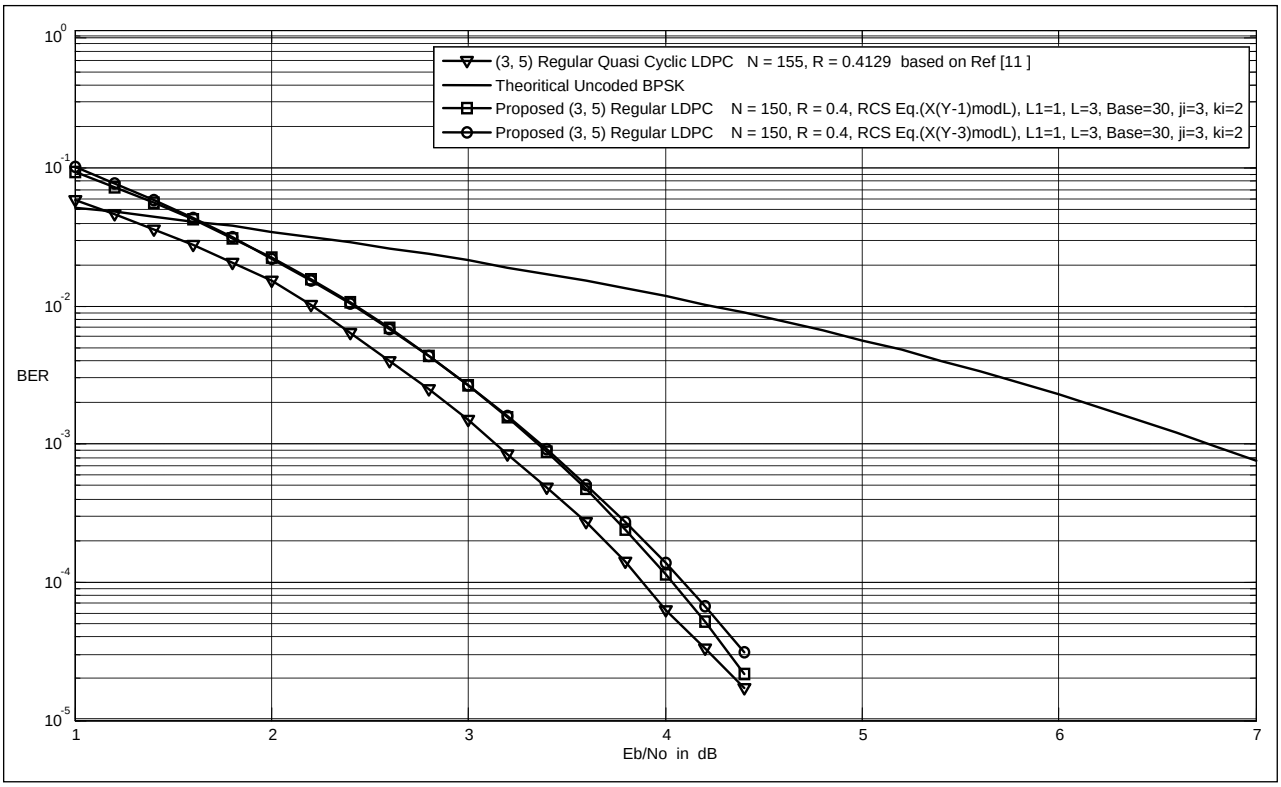

Figure 5.6: BER of (3, 5) regular LDPC with $R < 0.5$ and $N \leq 155$.

The comparison of BER performance for (3, 5) regular QC LDPC codes having R = 0.4029, N = 305 and the proposed (3, 5) regular LDPC having R = 0.4, N = 300 is shown Figure 5.7 with 50 iterations SPA. The proposed regular LDPC code in Figure 5.7 has parameter L_1 = 1, L = 6, *base* = 30, j_i = 3, k_i = 2 with deterministic RCS $(x \times (y - 1)) \bmod L$.

In this figure is also shown that (3, 5) regular QC LDPC has the best performance when compared with the proposed (3, 5) regular LDPC in code length N = 300. In the value of code rate $R < 0.5$, the proposed (3, 5) regular LDPC codes are not able to outperform (3, 5) QC LDPC codes even though code length of the proposed codes are increased into 300 bits.

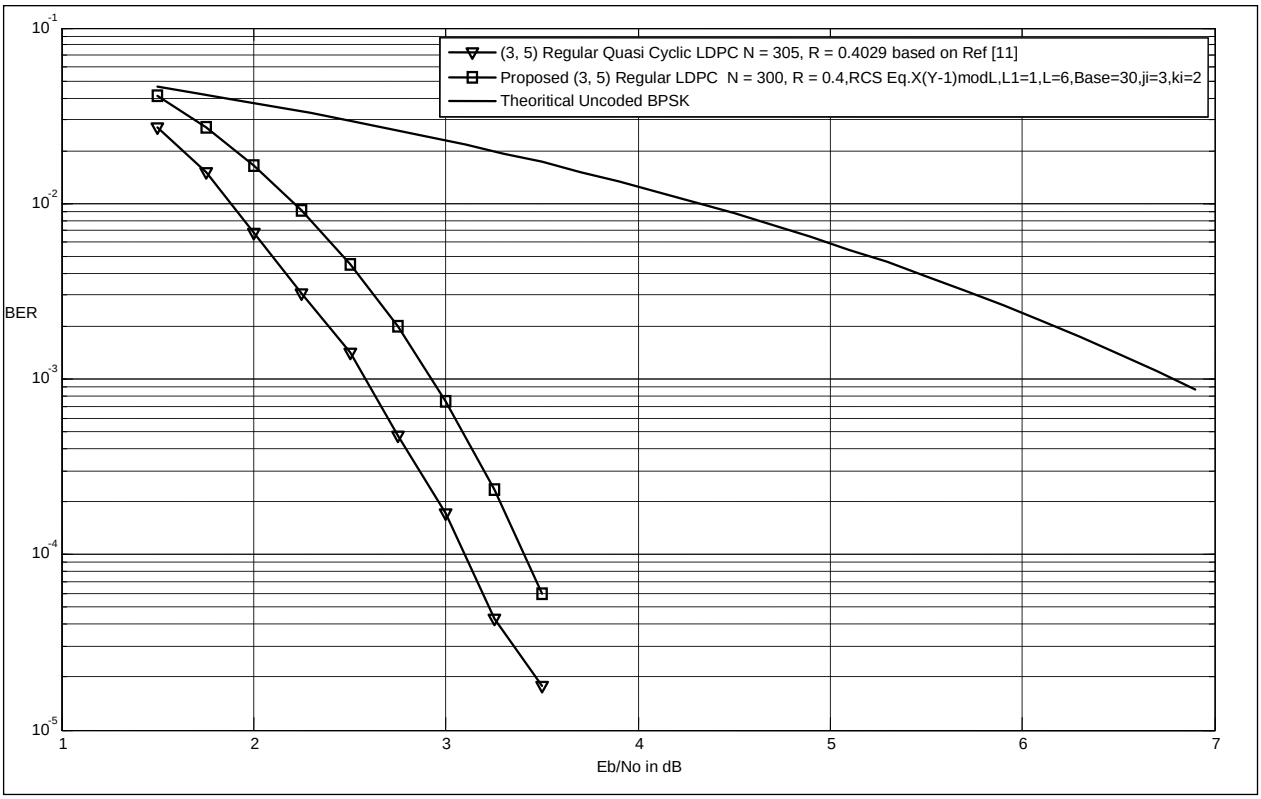

Figure 5.7: BER of (3, 5) regular LDPC with $R < 0.5$ and $N = 305$.

In Figure 5.7, the value of code rate and code length between the proposed (3, 5) regular and (3, 5) regular QC LDPC are also not the same. There is a code rate gain ($R > 0.4$) in Figure 5.7 that results in a better waterfall performance than the proposed regular LDPC codes since the linear dependency among rows in matrix H for (3, 5) regular QC LDPC codes.

- *(3, k) regular QC LDPC codes in high code rate (R = 0.75).*
  This regular QC LDPC code is built based on [48] represented by (3, 12) regular QC LDPC in Figure 5.8 with 30 iterations SPA. Comparison of LDPC codes with R = 0.75 and N = 2412 is by comparing BER performance of (3, 12) regular QC LDPC codes to the proposed (3, 12) regular LDPC code.

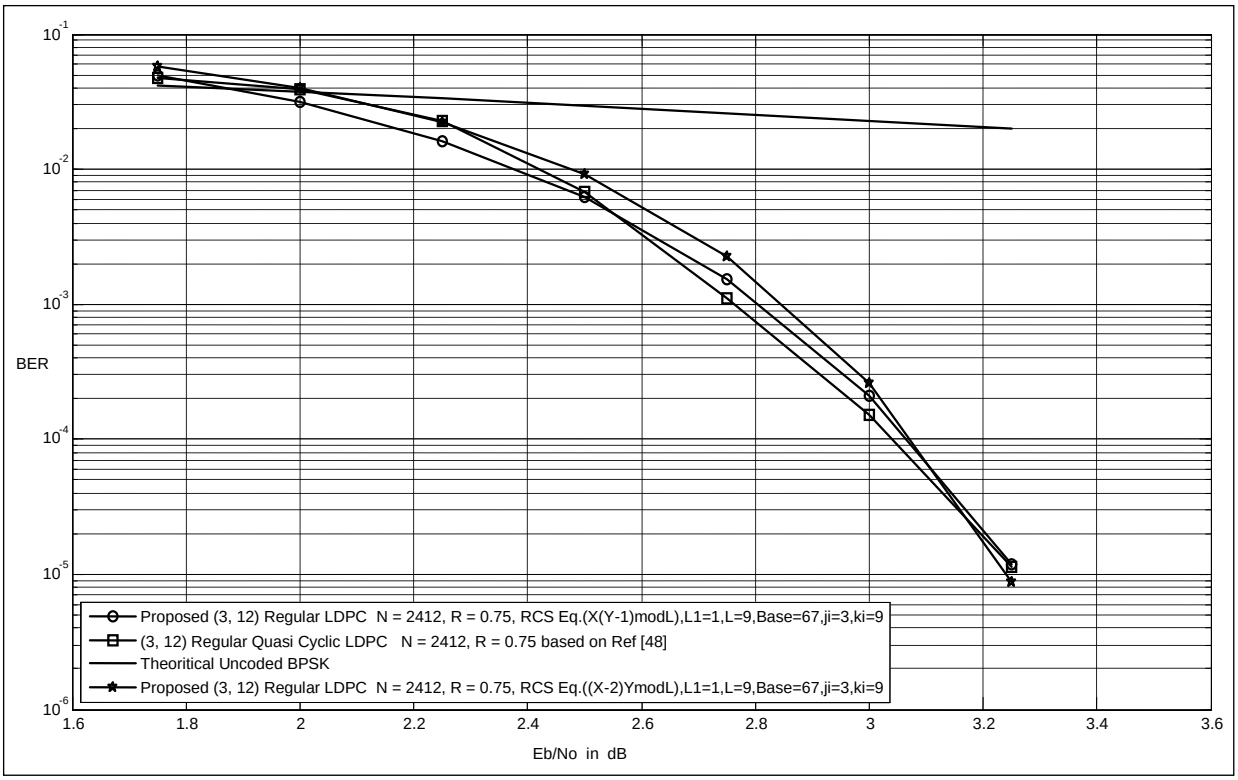

Figure 5.8: BER of (3, 12) regular LDPC with R = 0.75 and N = 2412.

The proposed regular LDPC codes in Figure 5.8 use parameter L_1 = 1, L = 9, *base* = 67, j_i = 3, k_i = 9 with two different deterministic RCS $(x \times (y - 1)) \bmod L$ and $((x - 2) \times y) \bmod L$. The proposed codes in Figure 5.8 have a comparable performance when the value of code rate R = 0.75 and both values of code rate also code length are the same.

Based on Figure 5.6 to Figure 5.8, following are some conclusions that can be drawn about the BER performance of the proposed (3, k) regular LDPC codes compared with (3, k) regular QC LDPC codes:

1. In terms of BER, the proposed code is not able to outperform regular QC LDPC codes when the value of code rate $R < 0.5$.

2. The linear dependency among rows in matrix H for (3, 5) regular QC LDPC codes, resulting in erasure of rows, gives a code rate gain that results in a BER performance better than the proposed code.
3. The proposed (3, 12) LDPC code has a comparable performance with regular QC when the value of code rate $R = 0.75$.

5.2.1.3 Comparison with (3, k) Regular Random LDPC Codes

In this section, comparison is done by comparing (3, k) regular random LDPC codes with the proposed (3, k) regular LDPC codes for low code rate ($R = 0.4$) and high code rate ($R = 0.875$).

- *(3, k) regular random LDPC codes in low code rate ($R = 0.4$).*
  Low code rate regular random LDPC codes are constructed according to [11] presented by (3, 5) regular random LDPC codes in Figure 5.9 and Figure 5.10. The number of iterations used in this section is set to 50 iterations of sum product decoding algorithm (SPA).

  The BER performance of (3, 5) regular random LDPC codes having $R = 0.4$ and $N = 155$ are compared with the proposed (3, 5) regular LDPC having $R = 0.4$ and $N = 150$ shown Figure 5.9. Meanwhile, Figure 5.10 gives a comparison of BER performance in code rate $R = 0.4$ and $N \leq 305$.

  The proposed regular LDPC codes in Figure 5.9 utilize parameter $L_1 = 1$, $L = 3$, $base = 30$, $j_i = 3$, $k_i = 2$ with two different deterministic RCS $(x \times (y - 1)) \bmod L$ and $(x \times (y - 3)) \bmod L$ while the proposed codes in Figure 5.10 use parameter $L_1 = 1$, $L = 6$, $base = 30$, $j_i = 3$, $k_i = 2$ with deterministic RCS $(x \times (y - 1)) \bmod L$ and $((x - 1) \times y) \bmod L$.

In Figure 5.9, the proposed (3, 5) regular LDPC code have a better error floor than regular random LDPC code. This may be due to the minimum distance (d_{min}) of random regular codes that can not grow linearly with the code length (N).

As can be seen from Figure 5.9 to Figure 5.10, the proposed (3, 5) regular LDPC codes are not able to outperform (3, 5) regular QC LDPC codes when the value of $R < 0.5$. Moreover, the value of code length between the proposed (3, 5) regular and (3, 5) regular QC LDPC in Figure 5.9 and Figure 5.10 are not the same.

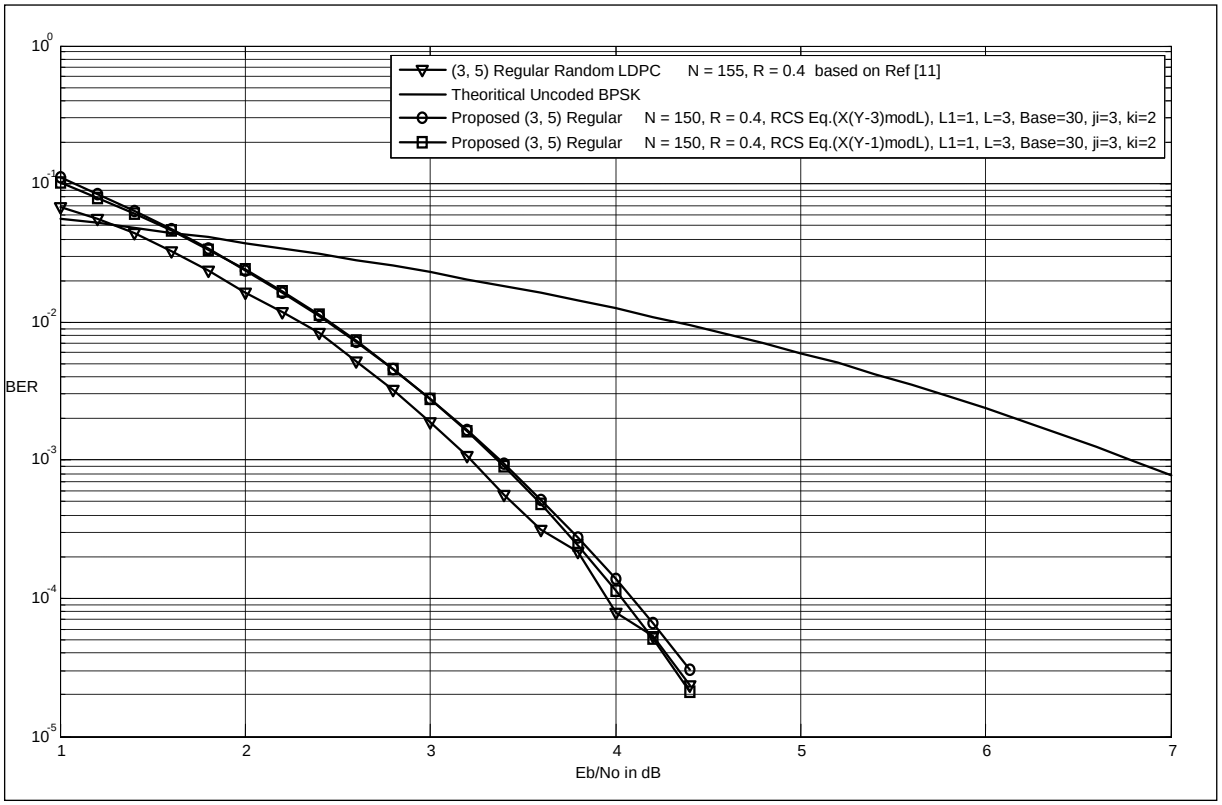

Figure 5.9: BER of (3, 5) regular LDPC with $R = 0.4$ and $N \leq 155$.

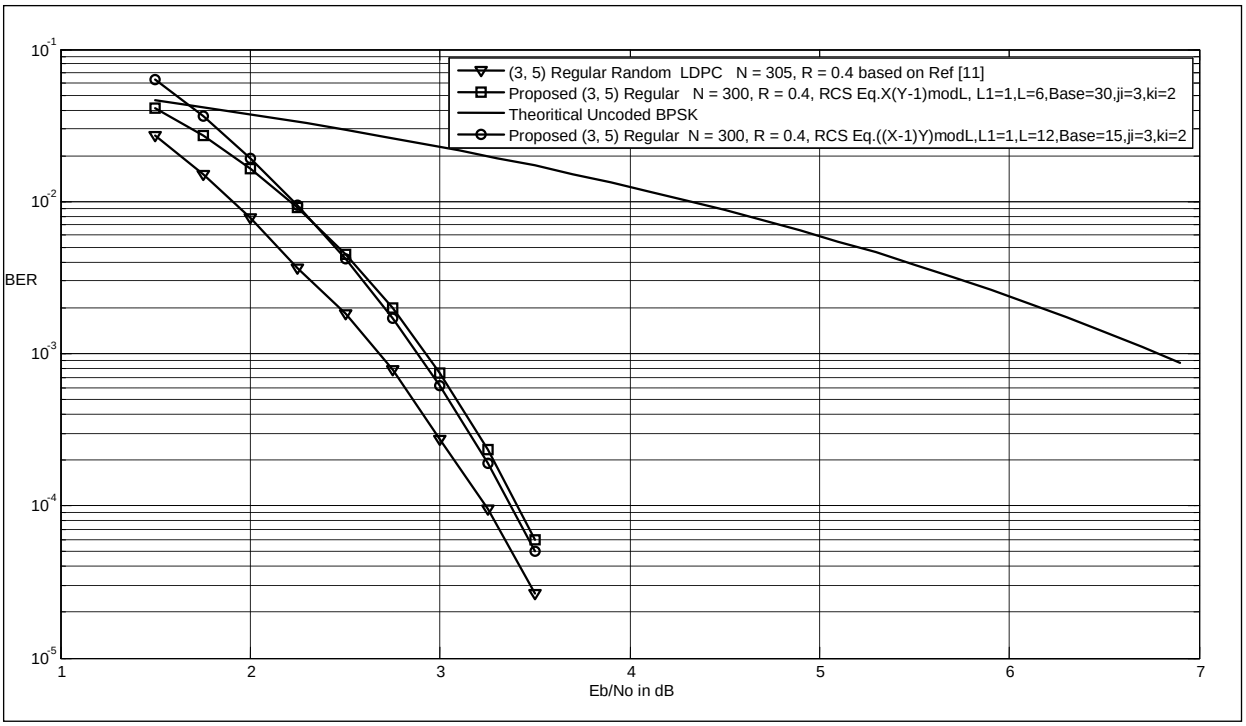

Figure 5.10: BER of (3, 5) regular LDPC with $R = 0.4$ and $N \leq 305$.

- *(3, k) regular random LDPC codes in high code rate (*$R = 0.875$*).*

  This regular random LDPC code is built based on [19] represented by (3, 24) regular random LDPC in Figure 5.11. The proposed regular LDPC codes in Figure 5.11 apply parameter $L_1 = 1$, $L = 7$, $base = 81$, $j_i = 3$, $k_i = 21$ with two different deterministic RCS $((x - 2) \times y) \bmod L$ and $(x \times (y - 3)) \bmod L$.

  In Figure 5.11, the proposed (3, 24) regular LDPC codes have a comparable performance when the value of $R = 0.875$ and code length $N = 2412$. We also see in Figure 5.11 that the BER curve of (3, 24) regular random LDPC code does not have a smooth waterfall shape. Instead, it flattens out for higher SNR. This behavior can be due to the fact that the minimum distance (d_{min}) of random regular code, perhaps, does not grow linearly with the code length. However, the proposed code does not suffer from such problem.

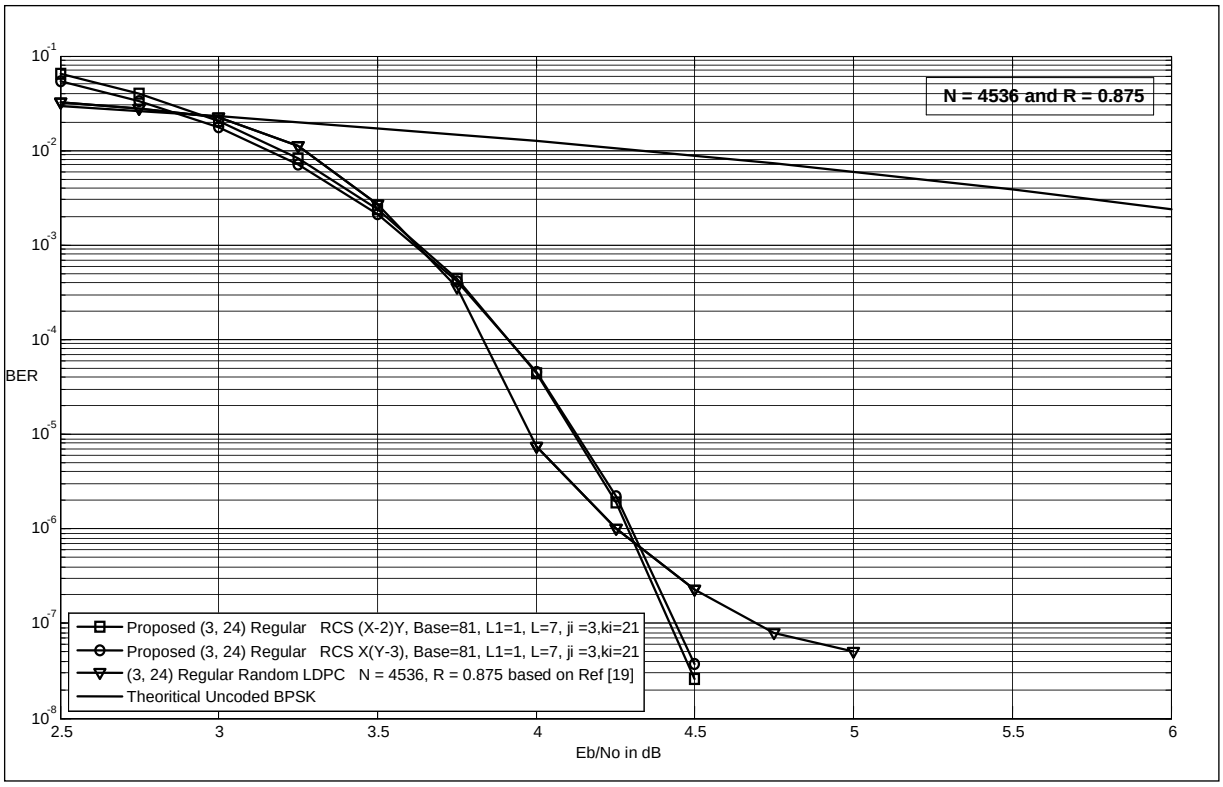

Figure 5.11: BER of (3, 24) regular LDPC with R = 0.875 and N = 4536.

Derived from Figure 5.9 to Figure 5.11, following are some conclusions that can be drawn about the BER performance of the proposed (3, k) regular LDPC codes when compared with (3, k) regular random LDPC codes:

1. The BER performance of the proposed code performance is not able to outperform regular QC LDPC codes when the value of code rate $R < 0.5$.
2. The minimum distance (d_{min}) of the proposed (3, k) regular codes grows linearly with the code length as seen in Figure 5.9 to Figure 5.11.
3. The proposed (3, 12) LDPC code has a comparable performance with regular random when the value of code rate R = 0.875 and N = 4536.

5.2.2 Block Error Rate (BLER) Performance

As mentioned in chapter 4, block error rate (BLER) is the probability that a decoded codeword is in error. A decoded codeword consists of a decoded information bit and a decoded parity bit. Code performance in this section is evaluated in term of block error rate (BLER) versus signal to noise ratio (SNR) curves.

Since there are not much works of BLER performance in (3, k) regular LDPC codes, this thesis takes only one comparison in BLER performance that is categorized into high code rate with $R = 0.769$.

BLER performance of this work is conducted by comparing the BLER of proposed (3, 13) regular LDPC codes with published (3, 13) regular QC LDPC and (3, 13) regular random LDPC codes in high code rate ($R = 0.769$) [10].

High rate BLER comparison for $R = 0.769$ is taken since it is shown in the BER performance that the proposed (3, k) regular are not able to outperform other (3, k) regular QC and random LDPC codes in code rate $R < 0.5$.

BLER performance in this section is obtained by simulating across 200 errors at each point of SNR. Linear interpolation is applied after getting BLER value at each point of SNR. The number of iterations used in this section is set into 200 iterations of sum product decoding algorithm (SPA).

BLER performance is classified into two sub-sections that gives comparison between the proposed (3, k) regular LDPC codes with other (3, k) regular quasi-cyclic (QC) and (3, k) regular random LDPC codes in code rate $R = 0.769$.

5.2.2.1 Comparison with (3, *k*) Regular Quasi Cyclic (QC) LDPC Codes

In order to compare with regular QC LDPC codes in code rate R = 0.769, this section takes (3, 13) regular QC LDPC code. (3, 13) regular QC LDPC code shown in Figure 5.12 is taken based on [10]. Figure 5.12 gives BLER performance of LDPC codes with R = 0.769 and N = 1053 between (3, 13) regular QC LDPC codes and the proposed (3, 13) regular LDPC code.

The proposed regular LDPC codes in Figure 5.12 employ parameter L_1 = 1, L = 3, *base* = 81, j_i = 3, k_i = 10 and girth at least 6-cycles with two different deterministic RCS $(x \times y) \bmod L$ and $(x \times (y - 3)) \bmod L$. In Figure 5.12, the proposed (3, 13) regular LDPC codes outperform (3, 13) regular QC LDPC codes in the same value of code rate R = 0.769 and code length N = 1053.

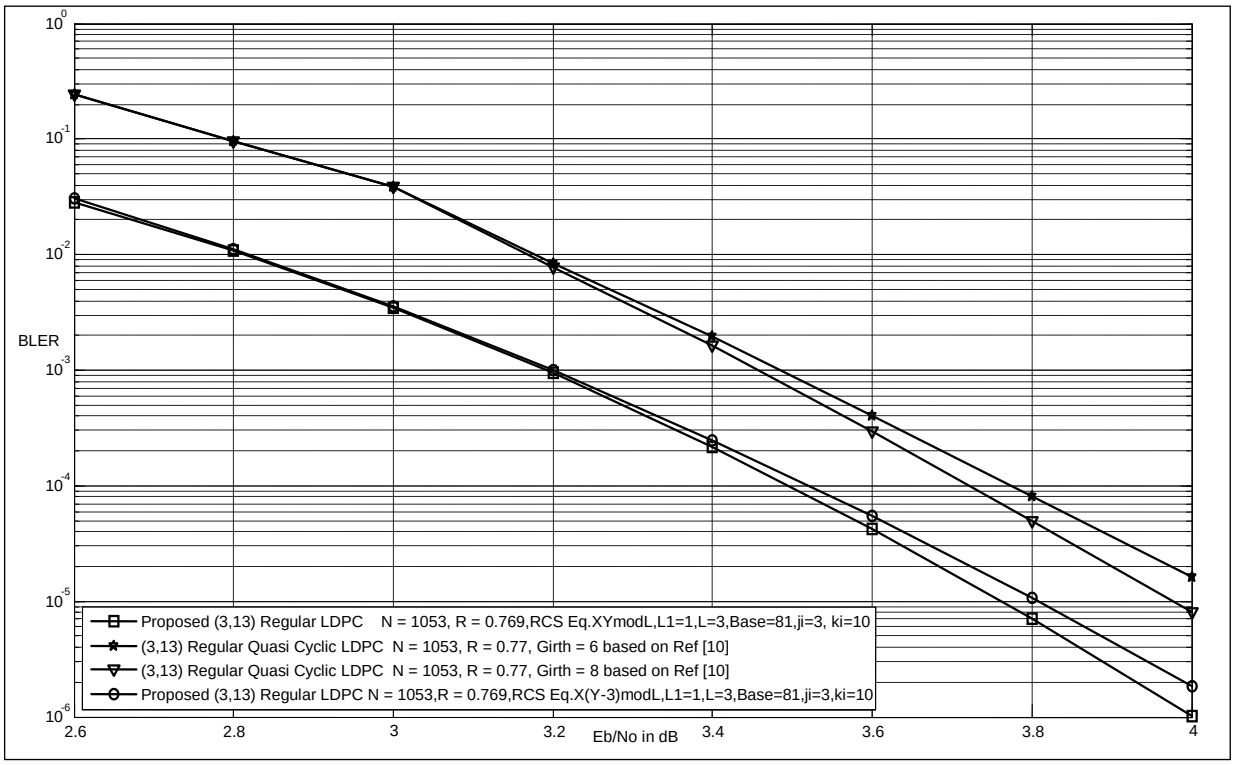

Figure 5.12: BLER Comparison with (3, 13) Regular QC LDPC R = 0.769 and N = 1053.

5.2.2.2 Comparison with (3, *k*) Regular Random LDPC Codes

This section takes an example of (3, 13) regular QC LDPC code for $R = 0.769$. Regular random LDPC for code rate $R = 0.769$ shown Figure 5.13 is built according to [10]. The proposed regular LDPC codes in Figure 5.13 employ parameter $L_1 = 1$, $L = 3$, $base = 81$, $j_i = 3$, $k_i = 10$ and girth at least 6 with two different deterministic RCS $(x \times y) \bmod L$ and $(x \times (y - 3)) \bmod L$.

Based on Figure 5.13, we note that the proposed (3, 13) regular LDPC codes give very good value of BLER compared with (3, 13) regular random LDPC codes with $R = 0.769$, $N = 1057$ and girth of 6-cycles even though the proposed code has smaller value of code length N.

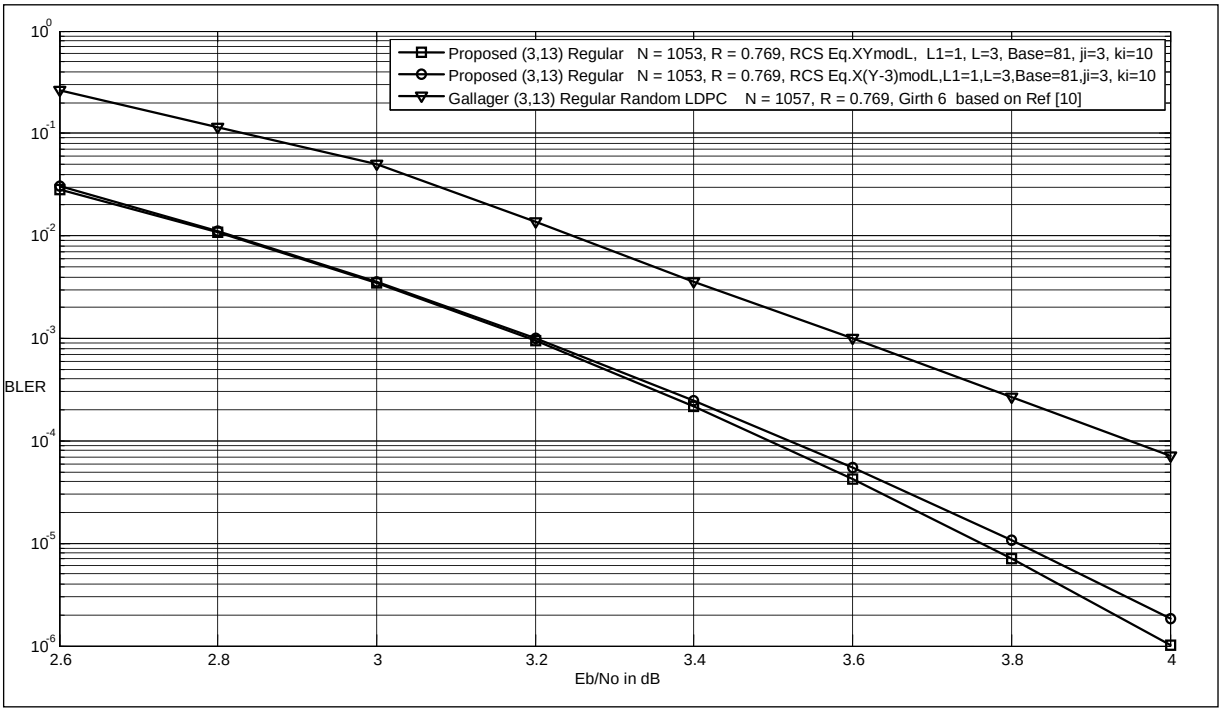

Figure 5.13: BLER Comparison with (3, 13) Regular Random $R = 0.769$ and $N = 1053$.

As can be seen from Figure 5.12 to Figure 5.13, the BLER performance of the proposed (3, 13) regular LDPC codes performance outperforms (3, 13) regular QC and (3, 13) regular random LDPC codes when code rate $R = 0.769$.

The followings are some conclusions that can be drawn about the BER and the BLER performance of the proposed (3, k) regular LDPC codes compared with other (3, k) regular QC and random LDPC codes:

The comparison of the BER and the BLER performance are made under the assumption that, for same code length (N) and code rate (R), the LDPC codes can be fairly compared irrespective of their design parameters.

1. The proposed (3, k) LDPC codes are not able to outperform other (3, k) regular LDPC codes when the value of code rate $R < 0.5$.
2. The proposed (3, k) LDPC codes have a comparable BER and BLER performance with other (3, k) regular QC and random LDPC codes in high code rate $R > 0.7$.
3. The minimum distance (d_{min}) of the proposed (3, k) regular codes grows linearly with the code length since they have no error floor.

5.2.3 Comparison with Shannon Limit

This section compares the proposed (3, k) regular LDPC codes with Shannon limit for binary input, BPSK modulation, AWGN channel. As mentioned in chapter 4, Shannon limit is defined as the theoretical limit on minimum SNR required for a coded system at a given code rate (R) to achieve arbitrarily small probability of error only if the SNR exceeds this limit. In this section, the proposed (3, k) regular LDPC codes are also compared with theoretical BER of un-coded BPSK and theoretical upper bound of an [N, K] binary block code with soft-decision decoding and BPSK modulation. As mentioned in the scope of this thesis, the code length (N) in this section is limited to $N < 15000$ bits.

The Shannon limit performance is only obtained for high code rate proposed LDPC codes with $R \geq 0.875$, since the comparison results of BER and BLER performance from the last section demonstrates that the proposed (3, k) regular have a comparable with other (3, k) regular LDPC codes for code rate $R > 0.7$.

The proposed (3, k) regular curves are obtained by simulating across 200 errors at each point of SNR. The number of iterations used in sum product decoding algorithm (SPA) is set into 50 iterations.

Figure 5.14 to Figure 5.18 show comparison of the proposed (3, k) regular codes with theoretical un-coded BPSK based on equation 4.1 and Shannon limit at BER 10^{-6} in code rate $R \geq 0.875$. The closest performance of the proposed regular codes in this thesis achieves 0.97 dB from Shannon limit at BER of 10^{-6} by (3, 42) regular LDPC code with R = 0.928, N = 14364 and encoding complexity $O(1.0225\ N)$ as can be seen in Figure 5.18. All the details of comparison obtained from Figure 5.14 to Figure 5.18 are consolidated and presented in Table 5.1.

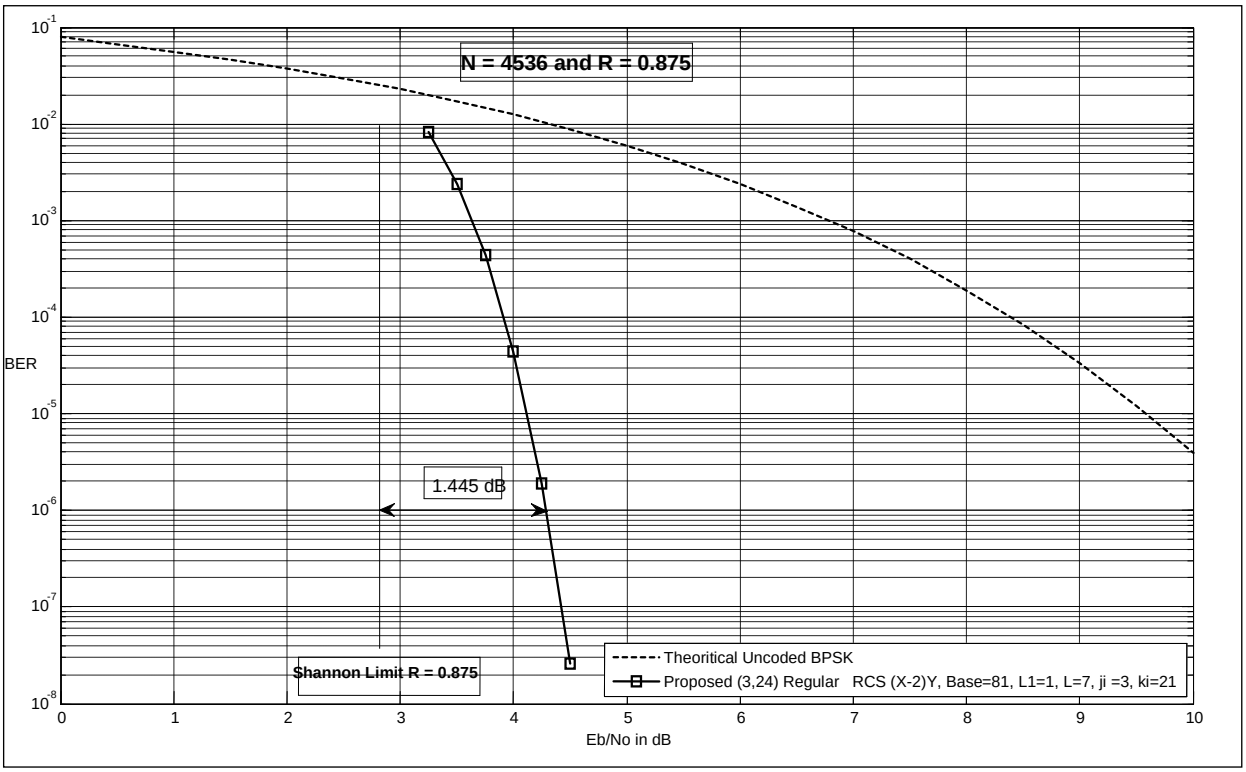

Figure 5.14: Proposed (3, 24) regular with R = 0.875 and N = 4536.

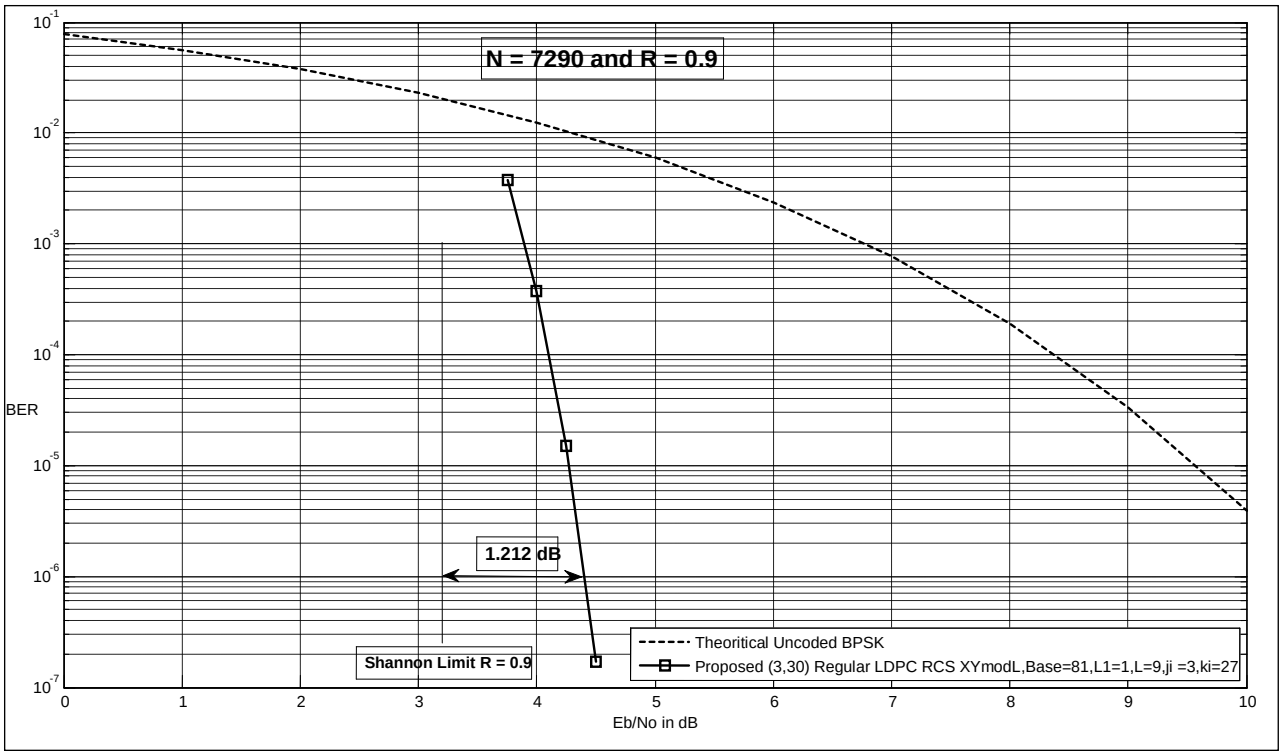

Figure 5.15: Proposed (3, 30) regular with R = 0.9 and N = 7290.

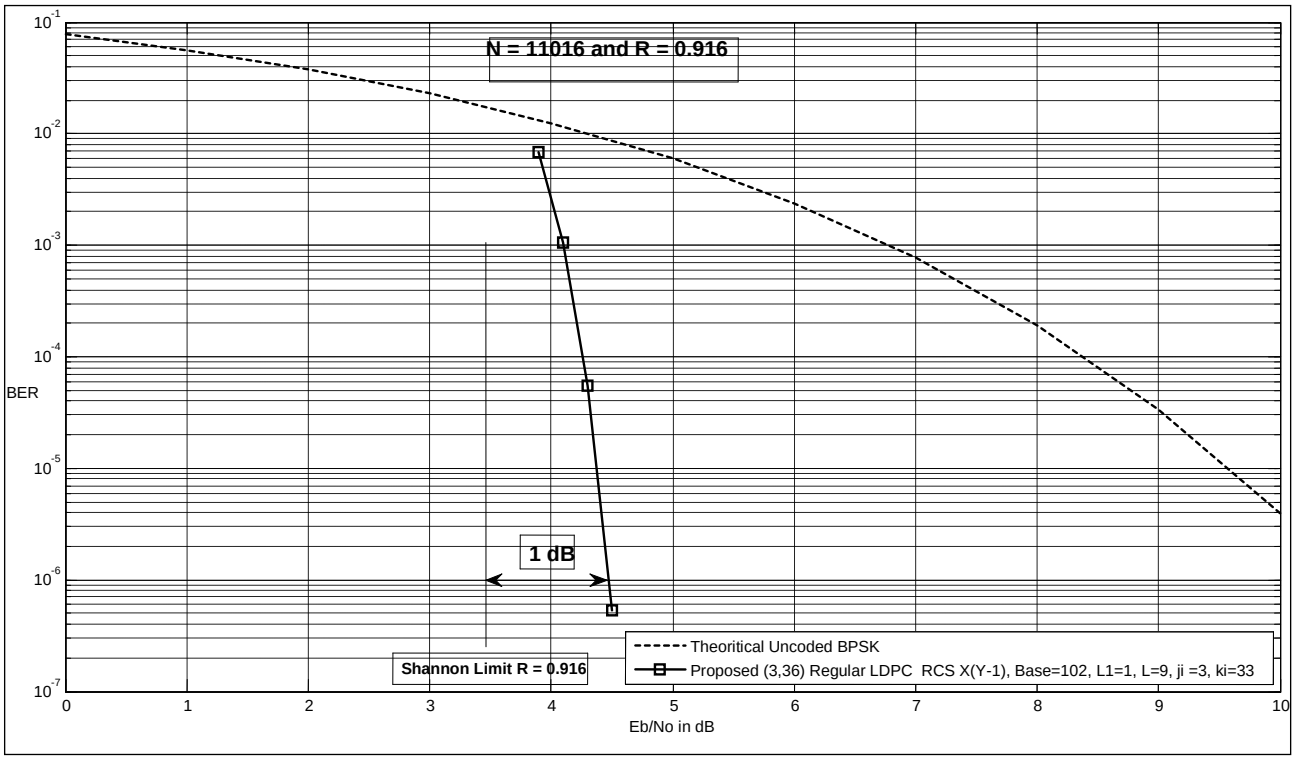

Figure 5.16: Proposed (3, 36) regular with R = 0.916 and N = 11016.

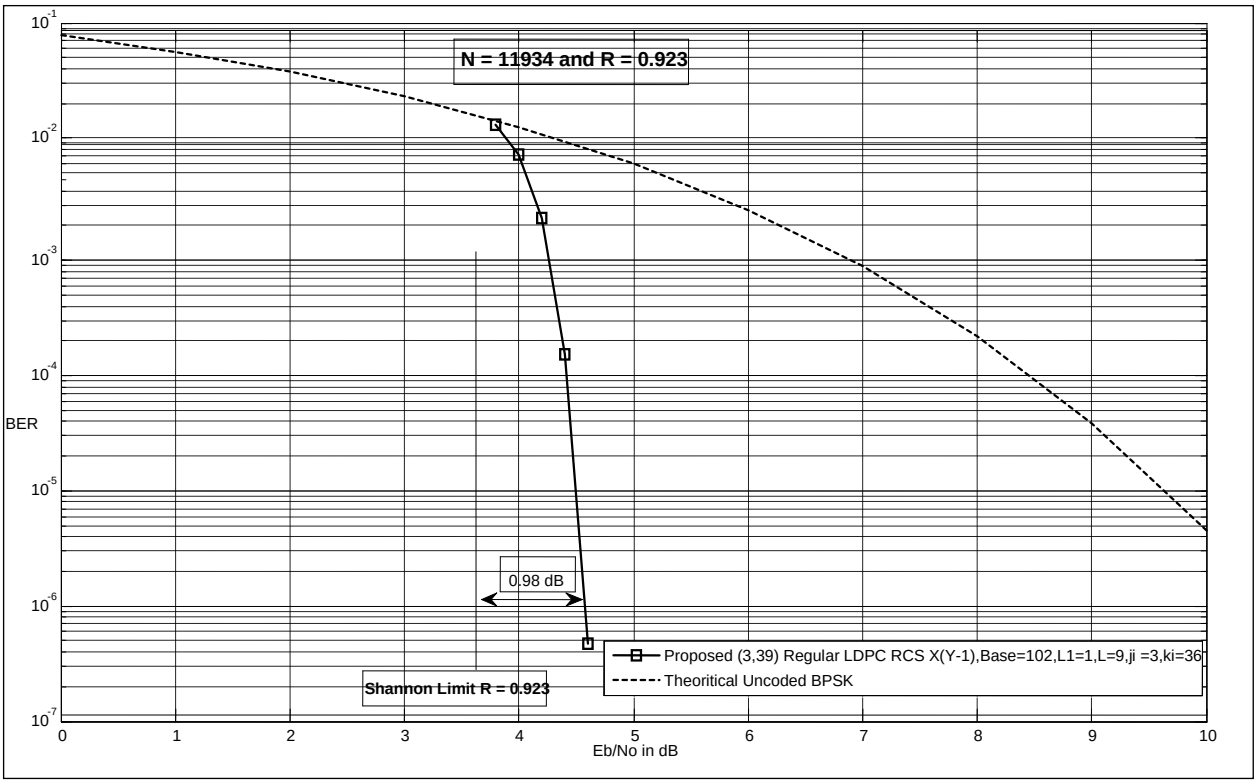

Figure 5.17: Proposed (3, 39) regular with R = 0.923 and N = 11934.

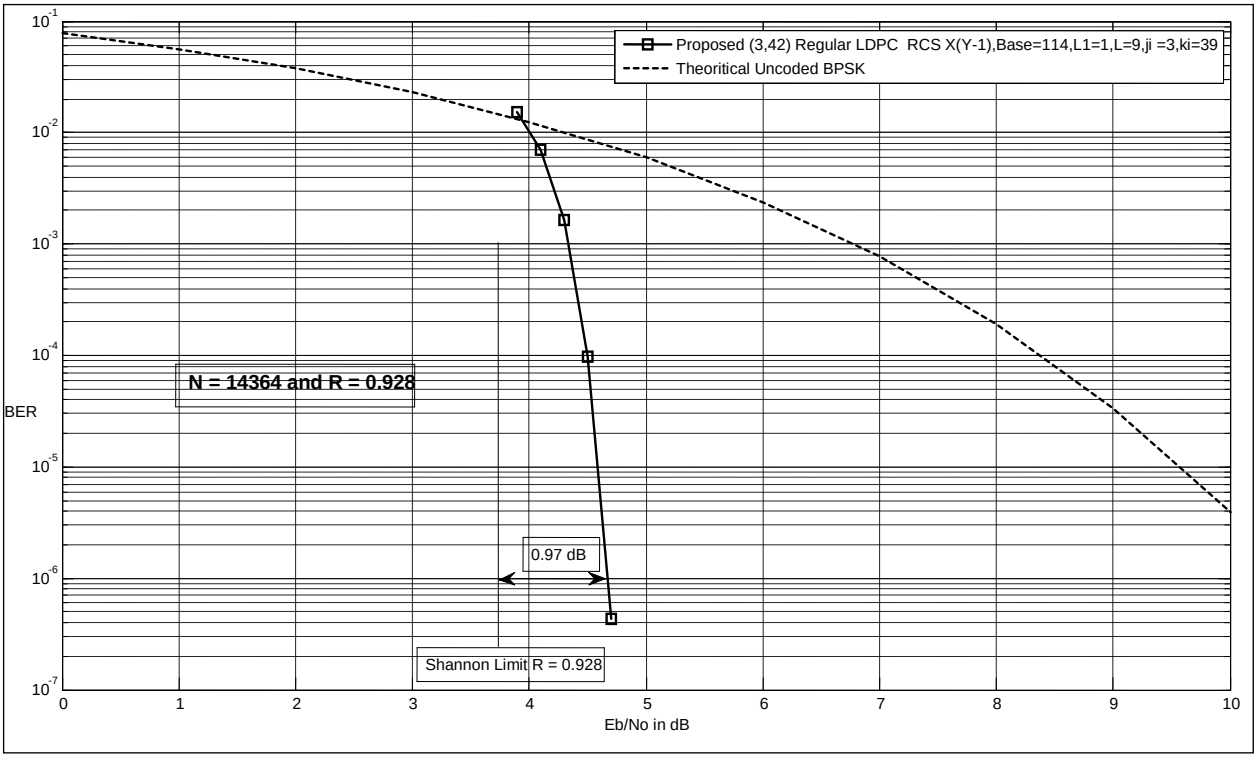

Figure 5.18: Proposed (3, 42) regular with R = 0.928 and N = 14364.

Table 5.1: Comparison of proposed (3, k) regular LDPC codes with Shannon limit.

Proposed (3, k) Regular LDPC Codes	Code Rate (R) and Code Length (N)	Parameter of Proposed (3, k) Regular LDPC Codes	Shannon Limit at BER 10^{-6} $R \geq 0.875$	Distance from Shannon limit
(3, 24) Regular LDPC code $g = 14$ $O(N + 0.0432\ N)$	$R = 0.875$ $N = 4536$	$L_1 = 1$, $L = 7$, $base = 81$, $j_i = 3$, $k_i = 21$ and Deterministic RCS $((x - 2) \times y) \bmod L$	2.84 dB	1.44 dB
(3, 24) Regular LDPC code $g = 18$ $O(N + 0.0346\ N)$	$R = 0.875$ $N = 9360$	$L_1 = 1$, $L = 9$, $base = 130$, $j_i = 3$, $k_i = 21$ and Deterministic RCS $((x - 2) \times y) \bmod L$	2.84 dB	1.12 dB
(3, 30) Regular LDPC code $g = 18$ $O(N + 0.044\ N)$	$R = 0.9$ $N = 7290$	$L_1 = 1$, $L = 9$, $base = 81$, $j_i = 3$, $k_i = 27$ and Deterministic RCS $x \times y \bmod L$	3.2 dB	1.212 dB
(3, 36) Regular LDPC code $g = 18$ $O(N + 0.0294\ N)$	$R = 0.916$ $N = 11016$	$L_1 = 1$, $L = 9$, $base = 102$, $j_i = 3$, $k_i = 33$ and Deterministic RCS $(x \times (y - 1)) \bmod L$	3.47 dB	1 dB
(3, 39) Regular LDPC code $g = 18$ $O(N + 0.0271\ N)$	$R = 0.923$ $N = 11934$	$L_1 = 1$, $L = 9$, $base = 102$, $j_i = 3$, $k_i = 36$ and Deterministic RCS $(x \times (y - 1)) \bmod L$	3.59 dB	0.98 dB
(3, 42) Regular LDPC code $g = 18$ $O(N + 0.02255\ N)$	$R = 0.928$ $N = 14364$	$L_1 = 1$, $L = 9$, $base = 114$, $j_i = 3$, $k_i = 39$ and Deterministic RCS $(x \times (y - 1)) \bmod L$	3.7 dB	0.97 dB

Column 1 of Table 5.1 states the value of parameter j and k in (j, k) regular LDPC codes. Since we are investigating (3, k) regular LDPC codes, the value of j is always j = 3. Column 1 of Table 5.1 also lists the value of g and encoding complexity of the proposed (3, k) regular LDPC codes.

Column 2 of Table 5.1 gives the value of code rate and code length in the proposed (3, k) regular LDPC codes while column 3 of Table 5.1 presents variable parameters utilized in the proposed codes such as parameter *base*, j_i, k_i, L, L_1 and method of deterministic RCS. Column 4 of Table 5.1 gives the theoretical value of Shannon limit for given code rate while column 5 of Table 5.1 lists the distance of the proposed (3, k) regular LDPC code from Shannon limit at BER 10^{-6} as obtained in the figures.

Table 5.1 shows that the longer the code length of the proposed (3, k) regular LDPC codes, the closer the performance is to Shannon limit. It shows that the closest performance of the proposed regular code to Shannon limit in Table 5.1 achieves 0.97 dB at BER of 10^{-6} by (3, 42) regular LDPC code with R = 0.928, N = 14364 and encoding complexity of the order of $O(1.0225\ N)$.

Comparison of BER theoretical upper bound of an [N, K] binary block code with soft-decision decoding with the proposed (3, k) regular LDPC codes is described in Table 5.2.

Table 5.2: Comparison of proposed (3, k) regular LDPC codes with theoretical upper bound of an [N, K] binary block code with soft-decision decoding

Proposed (3, k) Regular LDPC Codes with $d_{min} = 4$	Code Rate (R), Code Length (N) and Information bits (K)	E_b/N_0	BER of Proposed (3, k) Regular	BER Theoretical Upper bound of Block Code in Equation (4.1)
(3, 24) Regular LDPC code $g = 14$ $O(N + 0.0432\ N)$	$R = 0.875$ $N = 4536$ $K = 3969$	4.28 dB	10^{-6}	0.5
(3, 24) Regular LDPC code $g = 18$ $O(N + 0.0346\ N)$	$R = 0.875$ $N = 9360$ $K = 8190$	3.96 dB	10^{-6}	0.5
(3, 30) Regular LDPC code $g = 18$ $O(N + 0.044\ N)$	$R = 0.9$ $N = 7290$ $K = 6561$	4.412 dB	10^{-6}	0.5
(3, 36) Regular LDPC code $g = 18$ $O(N + 0.0294\ N)$	$R = 0.916$ $N = 11016$ $K = 10098$	4.47 dB	10^{-6}	0.5
(3, 39) Regular LDPC code $g = 18$ $O(N + 0.0271\ N)$	$R = 0.923$ $N = 11934$ $K = 11016$	4.57 dB	10^{-6}	0.5
(3, 42) Regular LDPC code $g = 18$ $O(N + 0.02255\ N)$	$R = 0.928$ $N = 14364$ $K = 13338$	4.67 dB	10^{-6}	0.5

Theoretical upper bound of an [N, K] binary block code with soft-decision decoding and BPSK modulation over AWGN channel is taken based on equation (4.2).

Column 1 of Table 5.2 lists the value of parameter j and k in (j, k) regular LDPC codes, the value of g, encoding complexity and the value of minimum distance (d_{min} = 4) of the proposed (3, k) regular LDPC codes while column 2 of Table 5.2 states the value of code rate (R), code length (N) and information bits (K) in the proposed (3, k) regular LDPC codes.

The value of E_b/N_0 (in dB) is given in column 3 of Table 5.2 that is used to obtain the BER value of theoretical upper bound of binary block code with soft-decision decoding and the proposed (3, k) regular LDPC codes. Column 4 of Table 5.2 gives the BER result of the proposed (3, k) regular LDPC codes when using the value E_b/N_0 in column 3.

Meanwhile, column 5 of Table 5.2 lists the theoretical upper bound of binary block code with soft-decision decoding and BPSK modulation using the same value of N, K and d_{min} with the proposed (3, k) regular LDPC codes.

Comparison of Shannon limit with published regular LDPC codes such as QC LDPC and cyclic LDPC codes is described in Table 5.3.

Table 5.3: Comparison of several regular LDPC codes with Shannon limit

Code Rate	Type of Regular LDPC Codes	(j, k) Regular LDPC	Code Length (N)	Distance from Shannon Limit	
				BER 10^{-6}	BER 10^{-4}
R = 0.875	Proposed Regular LDPC	(3, 24)	N = 4536 $O(1.0432\ N)$	1.44 dB	1. 074 dB
	Proposed Regular LDPC	(3, 24)	N = 9360 $O(1.0346\ N)$	1.12 dB	0. 945 dB
	Regular QC LDPC based on [1]	(4, 32)	N = 9360	0.95 dB	-
	Regular Cyclic LDPC based on [13]	(8, 64)	N = 32760	-	1.26 dB
R = 0.9	Proposed Regular LDPC	(3, 30)	N = 7290 $O(1.044\ N)$	1.212 dB	0.921 dB
	Regular Cyclic LDPC derived from [13]	(6, 64) or (7, 64)	N = 40950	-	0.92 dB
R = 0.916	Proposed Regular LDPC	(3, 36)	N = 11016 $O(1.0294\ N)$	1 dB	0.796 dB
	Regular Cyclic LDPC according to [13]	(5, 64) or (6, 64)	N = 49140	-	0.73 dB
R = 0.923	Proposed Regular LDPC	(3, 39)	N = 11934 $O(1.0271\ N)$	0.98 dB	0.824 dB
	Regular Cyclic LDPC derived from [13]	(4, 64) or (5, 64)	N = 53235	-	0.64 dB
R = 0.928	Proposed Regular LDPC	(3, 42)	N = 14364 $O(1.0225\ N)$	0.97 dB	0.798 dB
	Regular Cyclic LDPC according to [13]	(4, 64) or (5, 64)	N = 57330	-	0.57dB

Since there are not available comparison of (3, k) regular LDPC codes with Shannon limit in high code rate in the literature, we compare the proposed (3, k) regular LDPC codes with the existing comparison of regular LDPC codes with Shannon limit in high code rate.

Generally, the value of parameter j in the existing comparison of regular LDPC codes with Shannon limit for high code rate is above $j = 3$ which are $j = \{ 4, 5, 6, 7, 8\}$. Since the value of j is not $j = 3$, it makes that the value of parameter k of the existing comparison of regular LDPC codes with Shannon limit in high code rate is not the same with the proposed (3, k) regular LDPC codes.

Therefore, we only maintain the same value of code rate having different values of parameter j, k and code length (N). Only one value in Table 5.3 for code rate $R = 0.875$ that gives the same value of code rate and code length but different value of parameter j and k.

Column 1 of Table 5.3 states the value of code rate while column 2 of Table 5.3 gives the type of regular LDPC codes whether it is proposed regular, regular QC or regular cyclic LDPC code. Column 3 of Table 5.3 presents the value of parameter j and k in (j, k) regular LDPC code while column 4 of Table 5.3 reports the value of code length for regular LDPC codes. The last column of Table 5.3 describes the distance of regular LDPC codes from Shannon limit at BER $10^{-6}$and BER 10^{-4}.

Table 5.3 shows that the proposed (3, k) regular LDPC codes have a comparable performance with other QC LDPC and cyclic LDPC even though the proposed (3, k) regular LDPC codes have a smaller value of code length (N) and smaller value of parameter j and k.

From Table 5.1 to Table 5.3, following are some conclusions that can be drawn about the BER performance of the proposed (3, k) regular LDPC codes when compared with Shannon limit:

1. The longer the code length of the proposed (3, k) regular LDPC codes, the closer the BER performance to Shannon limit.
2. (3, 42) regular LDPC code with R = 0.928, N = 14364 and encoding complexity of the order of $O(1.0225\ N)$ achieves 0.97 dB at BER of 10^{-6}.
3. The BER value between the proposed (3, k) regular LDPC codes and theoretical upper bound of an [N, K] binary block code with soft-decision decoding is very far away even though both codes are using the same value of N, K and d_{min}.
4. Even though the proposed (3, k) regular LDPC codes have a smaller value of code length (N) and smaller value of parameter j and k, the proposed (3, k) regular codes have a comparable performance with other QC LDPC and cyclic LDPC.

5.3 Code Performance of Irregular LDPC Codes

Since the scope of this thesis is to develop a code construction method that can construct not only (3, k) regular LDPC codes but also irregular codes, this section gives one typical example of an irregular LDPC code so designed. Even though the proposed irregular LDPC code in this section is not meant to be the best performing code, it has characteristics of having no rank-deficiency of matrix H, no pre-processing step of encoding, no girth of 4-cycles, no singular nature of parity part (H_{par}) and low encoding complexity. The value of girth in the proposed irregular LDPC code is at least girth of 6-cycles.

Code performance in this section is evaluated in terms of BER curves. BER performance is obtained by simulating across 200 errors at each point of SNR. After getting BER value at each point of SNR, linear interpolation is applied.

Based on chapter 3, there are also five variables used for constructing irregular LDPC codes, namely, j_i, k_i, *base*, L and L_1. Since this section evaluates the performance of irregular LDPC codes, the value of parameter j_i used is not equal to $j_i = 3$. Simulation model of the proposed irregular LDPC code is developed based on procedures in chapter 4 using LDPC encoder-decoder of MATLAB® 7.4. The block diagram of the simulation is also described in Figure 4.1.

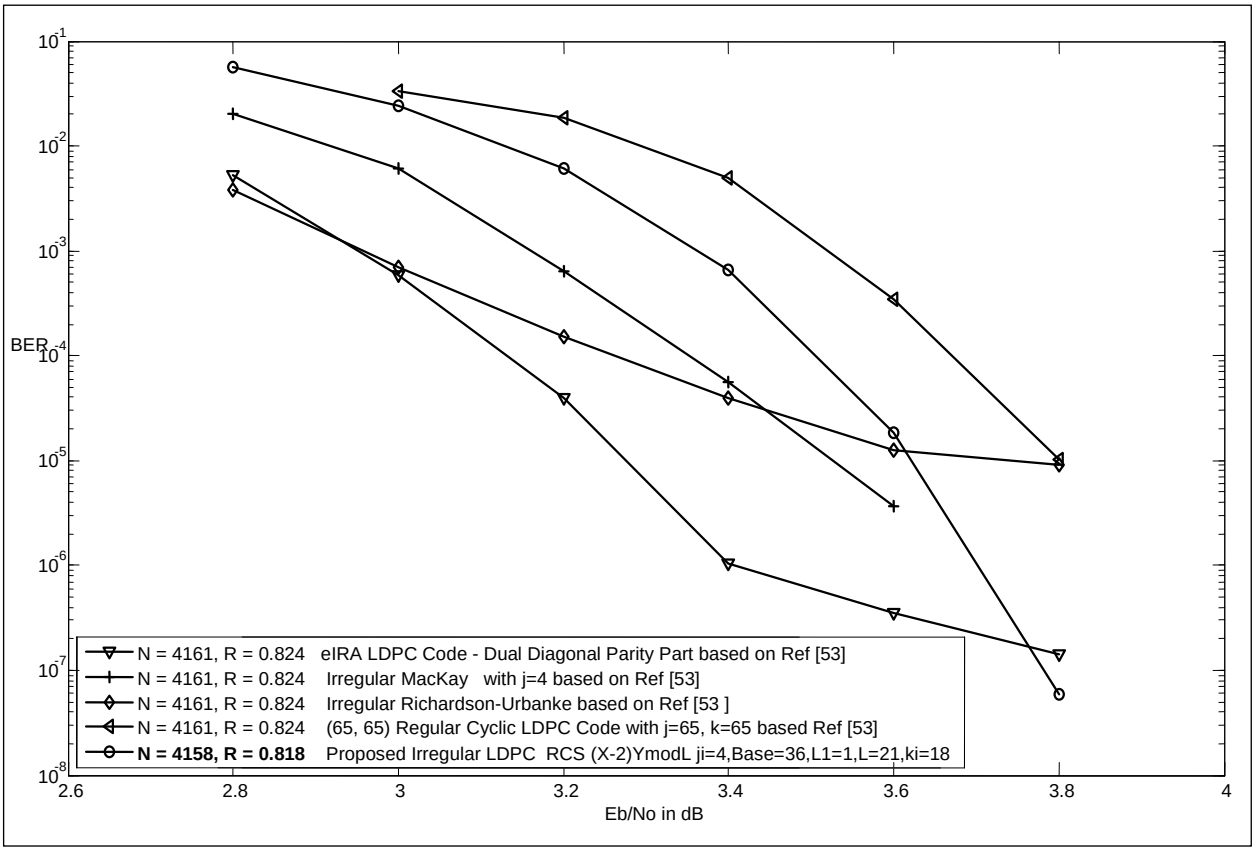

Figure 5.19: Comparison of irregular LDPC codes.

The proposed irregular LDPC code is compared with other published results, namely irregular LDPC codes using dual diagonal parity [53], irregular random LDPC codes using density evolution by Richardson et. al [53] and irregular random LDPC codes by Mackay [53]. This comparison is shown in Figure 5.19.

The performance result of the proposed irregular code is also compared with (64, 64) regular cyclic LDPC code [53] in order to show that the proposed design outperforms it even though the proposed irregular code has lower code rate and lower code length.

The proposed irregular LDPC code has parameter $L_1 = 1$, $L = 21$, $base = 36$, $j_i = 4$, $k_i = 18$ with deterministic RCS $((x - 2) \times y) \bmod L$ that give code length $N = 4158$ and code rate $R = 0.818$.

Based on [1], [13], the minimum distance (d_{min}) of irregular LDPC code is at least $j_{min} + 1$. Therefore, the minimum distance (d_{min}) of the proposed irregular LDPC codes with $j_{min} = 3$ is at least 4 that gives the ability of correcting at least 1 error.

Figure 5.19 has five BER curves. These are obtained for SNR over a range of 2.6 to 4 dB. The BER curve marked with circles is that of proposed irregular LDPC code while curve marked with triangle legend is eIRA LDPC code with dual diagonal parity part. Curve marked with diamond legend is irregular LDPC code using density evolution by Richardson et al while curve marked with plus legend is irregular LDPC code by Mackay.

It can be seen from Figure 5.19 that the proposed irregular LDPC code is not able to outperform other irregular LDPC codes. We also see in Figure 5.19 that the BER curve of eIRA LDPC code using dual diagonal parity part (triangle legend curve) and irregular random LDPC codes using density evolution by Richardson et al (diamond legend curve) do not have a smooth waterfall shape. Instead, they flatten out for higher SNR. This behavior can be due to the fact that the minimum distance (d_{min}) of these irregular codes, do not grow linearly with the code length. However, the proposed irregular LDPC code (circle legend curve) does not suffer from this problem.

5.4 Complexity Analysis

The proposed LDPC codes have been designed to have small proportion of ones in their matrix *H*. Since the low density of ones in the proposed *H* lends to low decoding complexity, complexity analysis in this section mainly focuses on encoding. The complexity analysis for irregular LDPC codes has been excluded from the scope.

Encoding in the proposed (3, *k*) regular LDPC codes employs Richardson-Urbanke method. Richardson-Urbanke method encodes by using inverse of parity check matrix. Encoding by parity check matrix simplifies encoding and decoding process by using only matrix *H* without converting it into *G*. An Advantage of encoding by inversion method is that it can be applied to any LDPC codes like random or structured codes. Moreover, the MATLAB® 7.4 utilizes the same method for encoding in its LDPC encoder built-in function, fec.ldpcenc(H).

As mentioned in chapter 2, there are two steps of encoding in Richardson-Urbanke method while assuming non-singular parity check matrix (*H*). Non-singular matrix *H* implies that matrix *H* is in a full-rank condition.

These two steps are pre-processing step of encoding and actual encoding. A pre-processing step prepares any parity check matrix to have an approximate triangular form and non-singular matrix ($ET^{-1}B + D$) before encoding while actual encoding computes parity p_1^T and p_2^T that depend on inverse of matrix ($ET^{-1}B + D$).

- Pre-processing step of encoding in Richardson-Urbanke method is divided into:

    1. *An approximate triangular form of parity check matrix (H).*
       This process is also called triangulation. The objective of triangulation in the pre-processing step is to get an approximate triangular form of matrix *H*. Therefore, if matrix *H* does not have an approximate triangular form, we need

to construct this form in matrix H by doing this step. This has the complexity of the order $O(N^3)$.

2. *Check-rank process*.

   The goal of check-rank process in the pre-processing step is to achieve non-singular condition for matrix $(ET^{-1}B + D)$ in equation (2.36). Check-rank process in pre-processing step of encoding is done by multiplying matrix H from the left based on equation (2.28) to (2.30) to clear matrix E – a process that can be achieved by Gaussian elimination. This process includes checking whether matrix $(ET^{-1}B + D)$ known as matrix Φ is singular or not and solves the problem of singularity by performing column permutation in matrix H. Based on [52], check-rank process has complexity of the order of $O(N^2 + g^3)$.

- Actual encoding step of Richardson-Urbanke method

  This step solves the calculation of parity p_1^T and p_2^T that depends on the inverse of matrix $(ET^{-1}B + D)$ denoted by matrix $-\Phi^{-1}$ in equation (2.34). Note that inverse of the matrix $(ET^{-1}B + D)$ can not be obtained if matrix H is rank-deficient.

  The inverse of matrix $(ET^{-1}B + D)$ is very crucial in computing parity p_1^T stated in equation (2.35). Since the value of p_2^T is also derived from parity p_1^T as given in equation (2.36), encoding complexity is determined by computation of p_1^T in Table 2.1. The complexity is derived to be of order $O(N + g^2)$.

  Complexity analysis in this section starts by exploring pre-processing step in the proposed (3, k) regular LDPC codes based on Richardson-Urbanke method. The discussion of complexity analysis is continued for comparison of pre-processing step of encoding followed by encoding complexity of (3, 5) regular QC LDPC codes. This section ends with listing the computational time of actual encoding step in the proposed codes that also compares the time of encoding between the proposed (3, k) regular with regular QC LDPC code and regular RA LDPC.

5.4.1 Pre-processing Step of the Proposed (3, k) Regular LDPC Codes

Since we focus to avoid pre-processing step of encoding, this section mainly discusses pre-processing step in the proposed (3, *k*) regular LDPC codes. Actual encoding of the proposed (3, *k*) regular LDPC codes gives complexity in the order of $O(N + g^2)$.

As mentioned above, there are two conditions in the pre-processing step of encoding in Richardson-Urbanke method:

1. An approximate triangular form of parity check matrix (*H*)
   In the proposed code construction, there is no triangulation process since the proposed codes have a built-in approximate lower triangular matrix of parity part (H_{par}). Therefore, the proposed code-construction is able to avoid complexity of the order of $O(N^3)$.

2. Check-rank process
   In the proposed code-construction, there is no process of clearing matrix *E*. It implies that the proposed code-construction is able to avoid complexity of the order of $O(N^2 + g^3)$.

   If there is girth of 4-cycles, the expected matrix *H* does not result. Therefore, one should be careful in choosing the value of parameter *base* and expansion factor *L* in the code design.

   Even though rank deficiency and girth of 4-cycle are not found in the proposed matrix, there is a possibility that matrix $(ET^{-1}B + D)$ is singular. Removing singularity as the worst case in the proposed code is solved by performing column permutation in sub- matrix *H* that is permuting one column of matrix H_{inf} and one column of matrix H_{par} until the determinant of matrix $(ET^{-1}B + D)$ is not zero anymore.

The sub matrices of matrix H_{inf} which are permuted by column permutation are matrix A and C while the sub matrices of matrix H_{par} which are permuted by column permutation are matrix B and matrix D.

Column permutation of matrix H_{inf} in the proposed code construction starts with the first column of matrix H_{inf} until the $(k_i \times Z \times L_1)$-th column of matrix H_{inf} while column permutation of matrix H_{par} starts with the first column of matrix H_{par} until g-th column of matrix H_{par}. The location of $(k_i \times Z \times L_1)$-th column of matrix H_{inf} and g-th column of matrix H_{par} can be viewed according to Figure 3.14 in section 3.4.1.

Actual encoding without any pre-processing step is achieved in two conditions:

- Matrix H has an approximate triangular form.
- Matrix $(ET^{-1}B + D)$ is non-singular.

If we fail to obtain the two conditions mentioned above, it means that we have to do a pre-processing step of encoding that is not trivial task and needs extra time in constructing matrix H.

5.4.2 Comparison with (3, k) Regular Quasi Cyclic (QC) LDPC Codes

This section compares pre-processing step of the proposed (3, 5) regular with (3, 5) quasi cyclic (QC) LDPC codes for low code rate (R = 0.4125 and R = 0.4029). It further compares encoding complexity of both codes.

We encode the BER curves in Figure 5.6 and Figure 5.7 using Richardson-Urbanke method and compare pre-processing step of the proposed (3, 5) regular with (3, 5) QC LDPC code. In both figures, we determine the amount of complexity needed in the pre-processing step of encoding for all curves.

- *(*3, 5*) regular QC LDPC codes in code rate R* = 0.4125.

  Comparison of pre-processing step of encoding of (3, 5) regular QC with R = 0.4125 and the proposed (3, 5) regular with R = 0.4 based on Figure 5.6 is presented in Table 5.3. Table 5.3 also compares encoding complexity of both codes.

  In order to be encoded by Richardson-Urbanke method, matrix H in Figure 5.6 needs to be transformed into an approximate triangular form. The process of transforming matrix H into an approximate triangular form yields the value of g as a gap of matrix H into its lower triangular form. As stated in actual encoding of Richardson-Urbanke method, encoding complexity is in the order of $O(N + g^2)$.

Table 5.4: Comparison with (3, 5) regular QC LDPC in Figure 5.6.

Type of LDPC Codes	Pre-Processing Step of Encoding	Encoding	
		g	Complexity $O(N + g^2)$
(3, 5) Regular QC N = 155 R = 0.4129	Based on [52]: 1. Triangulation: Complexity by $O(N^3)$ 2. Check rank: Gauss elimination to clear matrix E and column permutation by $O(N^2 + g^3)$. Total complexity is $O(N^3)$	4	$O(N + 0.103N)$
Proposed (3, 5) Regular N = 150, R = 0.4 *Base* = 30, L_1 = 1, L = 3 Deterministic RCS $(x \times (y - 1)) \bmod L$	No Pre-Processing Step	6	$O(N + 0.24N)$
Proposed (3, 5) Regular N = 150, R = 0.4 *Base* = 30, L_1 = 1, L = 3 Deterministic RCS $(x \times (y - 3)) \bmod L$	No Pre-Processing Step	6	$O(N + 0.24N)$

Based on [52], the value of g for regular QC in Figure 5.6 is $g = 4$. Since code length of (3, 5) regular QC is $N = 155$, encoding complexity is $O(N + 0.103N) = O(1.103N)$. The value of g in the proposed (3, 5) regular LDPC code is $g = 2 \times L \times L_1 = 6$. Encoding complexity of the proposed (3, 5) regular LDPC code is $O(N + 0.24N)$ since code length of proposed (3, 5) regular is $N = 150$. It is shown in Table 5.3 that the proposed code-construction is able to avoid triangulation and check-rank process of pre-processing step of encoding that has total complexity in the order of $O(N^3)$.

- *(*3, 5*) regular QC LDPC codes in code rate* $R = 0.4029$.
  Comparison of pre-processing step of encoding of (3, 5) regular QC with $R = 0.4029$ and the proposed (3, 5) regular with $R = 0.4$ based on Figure 5.6 is presented in Table 5.4.

Table 5.5: Comparison with (3, 5) regular QC LDPC in Figure 5.7.

Type of LDPC Codes	Pre-Processing Step of Encoding	Encoding	
		g	Complexity $O(N + g^2)$
(3, 5) Regular QC $N = 305$ $R = 0.4029$	Based on [52]: 1. Triangulation: Complexity by $O(N^3)$ 2. Check rank: Gauss elimination to clear matrix E and column permutation by $O(N^2 + g^3)$. Total complexity is $O(N^3)$	10	$O(N + 0.328N)$
Proposed (3, 5) Regular $N = 300$, $R = 0.4$ *Base* = 30 $L_1 = 1$, $L = 6$	No Pre-Processing Step	12	$O(N + 0.48N)$

Table 5.4 also compares encoding complexity of both codes. In order to be encoded by Richardson-Urbanke method, an approximate triangular form for three curves in Figure 5.7 is needed and the value of gap g needed to be searched for.

Derived from [52], the value of g for regular QC in Figure 5.8 is $g = 10$. Since code length of (3, 5) regular QC is $N = 305$, encoding complexity is $O(N + 0.328N) = O(1.328N)$. The value of g in the proposed (3, 5) regular LDPC code is $g = 2 \times L \times L_1 = 12$. Encoding complexity of the proposed (3, 5) regular LDPC code is $O(N + 0.48N)$ since code length of proposed (3, 5) regular is $N = 300$.

It is shown in Table 5.5 that the proposed code-construction is able to avoid triangulation and check-rank process of pre-processing step of encoding that has total complexity in the order of $O(N^3)$.

5.4.3 Comparison of Computational Time of Encoding

This section calculates computational time of actual encoding step based on Richardson-Urbanke method in the proposed code-construction. Computational time of actual encoding step is accomplished by utilizing time elapsed of a built-in function in MATLAB. This section also compares the time of encoding between the proposed (3, 6) regular with (3, 6) regular QC LDPC code that was given in Table 5.7.

Encoding step of the proposed code-construction solves parity p_1^T and p_2^T according to equation (2.35) and (2.36). Time of getting codeword is taken after combining information bits m^T, parity p_1^T and p_2^T written in milliseconds (ms).

Computational time of actual encoding in this section is taken 100 times and we take median value of time elapsed in 100 measurements. The comparison of computational time is applied in the same computer having the same specification. The specification of

our computer is using AMD Turion 64 X2 TL-60 processor having speed of 2.0 GHz and 3.18 GB of random access memory (RAM).

The results of computational time of encoding step are given in Table 5.6 until Table 5.9 in milliseconds. Table 5.6 gives computational time of the proposed (3, 8) regular and (3, 5) regular LDPC code while Table 5.7 presents computational time of the proposed irregular LDPC codes.

Table 5.6: Proposed (3, *k*) regular LDPC codes

Parameters	Parity p_1^T (Milliseconds)	Parity p_2^T (Milliseconds)	Codeword (Milliseconds)
(3, 8) Regular LDPC $L_1 = 1$, $L = 4$, $Base = 15$, $k_i = 5$, $j_i = 3$. Deterministic RCS : $((x - 1) \times y) \bmod L$ The size of matrix $H = 60 \times 160$	0.65	0.94	1.28
(3, 5) Regular LDPC $L_1 = 1$, $L = 8$, $Base = 15$, $k_i = 2$, $j_i = 3$. Deterministic RCS : $((x - 1) \times y) \bmod L$ The size of matrix $H = 120 \times 200$	3.72	5.62	7.49

Table 5.7: Proposed irregular LDPC codes

Parameters of Proposed Irregular LDPC Codes	Parity p_1^T (Milliseconds)	Parity p_2^T (Milliseconds)	Codeword (Milliseconds)
$L_1 = 1$, $L = 8$, $Base = 15$, $k_i = 3$, $j_i = 4$. Deterministic RCS : $((x - 1) \times y) \bmod L$ The size of matrix $H = 120 \times 240$	3.8	5.6	7.5
$L_1 = 1$, $L = 8$, $Base = 15$, $k_i = 2$, $j_i = 5$. Deterministic RCS : $((x - 1) \times y) \bmod L$ The size of matrix $H = 120 \times 200$	3.7	5.5	7.3

Table 5.8 compares time elapsed of encoding between the proposed (3, 6) regular with (3, 6) regular QC LDPC code. The proposed (3, 6) regular LDPC code uses parameter L_1 = 1, L= 7, *base* = 9, j_i = 3, k_i = 3 and deterministic RCS ($x \times y$) mod L. Meanwhile, (3, 6) regular QC LDPC code is derived from Gabofetswe Malema and Michael Liebelt [26]. In (3, 6) regular QC LDPC code, there are two linear dependent rows that are needed to be erased and to be replaced by the new ones. Both of matrices H are divided into 6 sub-matrices as given in Figure 3.14.

The last column of Table 5.8 gives time improvement of getting the codeword between the proposed (3, 6) regular LDPC code and (3, 6) regular QC LDPC code based on reference [26]. Since there is no difference in time elapsed in getting the codeword between both (3, 6) regular LDPC codes, the time improvement is 0 %. The time improvement of Table 5.8 is calculated based on equation below:

$$\frac{\left|\text{Time of getting proposed codeword - Time of getting QC codeword}\right|}{\text{Time of getting QC codeword}} \times 100\ \% \qquad (5.1)$$

Table 5.8: Comparison of computational time of encoding

The size of matrix H = 63 × 126		
Type of Code	Codeword (Milliseconds)	Time Improvement of Getting Codeword
Proposed (3, 6) Regular LDPC Codes L_1 = 1, L = 7, *Base* = 9, k_i = 3, j_i = 3 Deterministic RCS ($x \times y$) mod L	1.2	0 %
(3, 6) Regular QC LDPC Codes Based on Ref [26]	1.2	

Table 5.9 compares time elapsed of encoding between the proposed (3, 6) regular with regular repeat accumulate (RA) LDPC codes. The choice of RA LDPC codes is taken since these codes have linear encoding complexity. Table 5.9 also measures encoding complexity of both codes.

Table 5.9: Computational time of encoding with $L_1 = 1$, $L = 3$, $base = 18$ and $k_i = 3$.

<table>
<tr><td colspan="4">Common Parameter: $L_1 = 1$, $L = 3$, $Base = 18$ and $k_i = 3$.
Deterministic RCS of Proposed Regular and Irregular Codes: $((x - 1) \times y) \bmod L$
The size of matrix $H = 54 \times 108$</td></tr>
<tr><td colspan="3">Type of Code</td><td>Codeword (Milliseconds)</td><td>Encoding Complexity</td></tr>
<tr><td rowspan="2">Proposed (3, 6) Regular LDPC Codes
$j_i = 3$</td><td>Parity p_1^T (Milliseconds)</td><td>Parity p_2^T (Milliseconds)</td><td rowspan="2">1.16</td><td rowspan="2">$O(N) + 0.333\ N$</td></tr>
<tr><td>0.58</td><td>0.85</td></tr>
<tr><td rowspan="2">Proposed Irregular LDPC Codes
$j_i = 1$</td><td>Parity p_1^T (Milliseconds)</td><td>Parity p_2^T (Seconds)</td><td rowspan="2">1.25</td><td rowspan="2">$O(N) + 0.333\ N$</td></tr>
<tr><td>0.60</td><td>0.91</td></tr>
<tr><td rowspan="2">Proposed Irregular LDPC Codes
$j_i = 2$</td><td>Parity p_1^T (Milliseconds)</td><td>Parity p_2^T (Seconds)</td><td rowspan="2">1.18</td><td rowspan="2">$O(N) + 0.333\ N$</td></tr>
<tr><td>0.59</td><td>0.86</td></tr>
<tr><td rowspan="2">Regular RA LDPC Codes
$j_i = 3$</td><td colspan="2">Parity p^T (Milliseconds)</td><td rowspan="2">0.96</td><td rowspan="2">$O(N)$</td></tr>
<tr><td colspan="2">0.47</td></tr>
</table>

Derived from Table 5.9, regular RA LDPC codes with dual diagonal parity part are having the smallest value of time elapsed of encoding in milliseconds. The complexity of regular RA LDPC codes is linear. Even though encoding complexity of the proposed code-construction is not as linear as regular RA LDPC codes, computational time of encoding in the proposed code-construction is still comparable with regular RA LDPC codes.

Encoding complexity of the proposed code is determined by computation of p_1^{T} in Table 2.1 having complexity of the order of $O(N + g^2)$. The measurement of encoding complexity involves calculating the value of g in relation to code length of LDPC codes denoted by N.

In the proposed (3, k) regular and irregular LDPC codes, the value of g is influenced by parameter L and L_1 since g is $2 \times L \times L_1$. The value of encoding complexity of the proposed (3, k) regular and irregular LDPC code is written by equation below:

$$O(N + g^2) = O(N + 4 \times L^2 \times L_1^2) \tag{5.2}$$

Using, as small as possible, parameter L and L_1 while utilizing, as large as possible, the parameter *base* decreases the encoding complexity of the proposed codes.

The influence of some methods of deterministic RCS in computational time of encoding is presented in Table 5.10. Applying variable methods of deterministic RCS by maintaining the same parameters of $L_1 = 1$, $L = 6$, $Base = 39$, $j_i = 3$, $k_i = 2$ and the same value of encoding complexity gives a comparable time elapsed of encoding in seconds for getting the codeword, the parity $p_1{}^T$ and the parity $p_2{}^T$.

Since the time for getting codeword using either of 7 methods of deterministic RCS is the same and is in the range of 50 milliseconds, the most recommended deterministic RCS is determined by the value of parameter *base* and expansion factor L chosen that yield no pre-processing before encoding. The details of feasible values of parameter *base* and expansion factor L that gives matrix H with no pre-processing before encoding, no rank-deficiency and no girth of 4-cycles is given in the appendix.

Table 5.10: Computational time with variable deterministic RCS

(3, 5) Regular LDPC codes Common Parameter: $L_1 = 1$, $L = 6$, $Base = 39$, $j_i = 3$ and $k_i = 2$. The size of matrix $H = 234 \times 390$				
Deterministic RCS	Parity $p_1{}^T$ (Milliseconds)	Parity $p_2{}^T$ (Milliseconds)	Codeword (Milliseconds)	Encoding Complexity
$(x \times y) \bmod L$	26.7	39.2	52.4	$O(N) + 0.369\ N$
$((x-1) \times y) \bmod L$	27	39.3	52.2	$O(N) + 0.369\ N$
$((x-2) \times y) \bmod L$	27.8	40.7	53.6	$O(N) + 0.369\ N$
$((x-3) \times y) \bmod L$	26.1	38.3	51.1	$O(N) + 0.369\ N$
$(x \times (y-1)) \bmod L$	26.7	39.2	52.2	$O(N) + 0.369\ N$
$(x \times (y-2)) \bmod L$	27.3	39.9	52.9	$O(N) + 0.369\ N$
$(x \times (y-3)) \bmod L$	26.7	39.2	52.3	$O(N) + 0.369\ N$

5.5 Summary

In this chapter, results, analysis and discussion of the proposed (3, *k*) regular LDPC codes in terms of BER performance, BLER performance, Shannon limit and complexity analysis have been presented. The discussion in this chapter is initiated by code performance parameters that evaluate the proposed (3, *k*) regular LDPC codes in terms of BER, BLER and Shannon limit. In order to validate the proposed (3, *k*) regular LDPC codes, the error performance is compared with some published results of (3, *k*) regular QC LDPC, (3, *k*) regular random LDPC codes theoretical un-coded BPSK and theoretical upper bound of an [*N*, *K*] binary block code with soft-decision decoding and BPSK modulation.

All the parity-check matrices designed for the purpose and used in this section are having no rank-deficiency of matrix *H*, no pre-processing step of encoding, low encoding complexity and non-singular parity part (H_{par}).

Based on results presented in section 5.2, the proposed (3, *k*) regular codes have a comparable performance with (3, *k*) regular QC and (3, *k*) regular random when $R > 0.7$. Moreover, the BER value between the proposed (3, *k*) regular LDPC codes and theoretical upper bound of an [*N*, *K*] binary block code with soft-decision decoding is very far away even though both codes are using the same value of *N*, *K* and d_{min}.

The proposed (3, *k*) regular codes are proven to achieve code performance below 1.445 dB from Shannon limit at BER of 10^{-6} when the value of code rate is greater than $R = 0.875$. It is also shown that the proposed (3, 42) regular code achieves a performance of only 0.97 dB from Shannon limit at BER 10^{-6} with encoding complexity $O(1.0225\ N)$, for $R = 0.928$ and $N = 14364$.

This section is followed by one example of BER performance of the proposed irregular LDPC code. Even though the proposed irregular LDPC code is not able to outperform other irregular LDPC codes, the BER curve of the proposed irregular LDPC code does not flatten out for higher SNR.

After that, it is followed by investigation of complexity analysis that covers pre-processing step in the proposed (3, k) regular LDPC codes based on Richardson-Urbanke method. The complexity analysis for irregular LDPC codes has been excluded from the scope.

The discussion of complexity analysis is continued and pre-processing step of encoding is compared in terms of its complexity with (3, 5) regular QC LDPC codes. It is shown that the proposed code-construction is able to avoid pre-processing step of encoding that has otherwise total complexity of the order of $O(N^3)$. This section ends with listing down computational time of actual encoding step in the proposed codes and also compares the time of encoding between the proposed (3, 6) regular with (3, 6) regular QC LDPC code.

In the next chapter, we will conclude the entire work of this thesis and recommend future work that can be carried out.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Introduction

In the previous chapters, the proposed code-construction method for (3, k) regular LDPC codes has been described and the performance results thereof presented. These codes are shown to have no rank-deficiency of matrix H, no pre-processing before encoding, no singular nature of parity part (H_{par}) and fairly low encoding complexity. In this chapter, we conclude the entire work and suggest future work for further research in this area.

6.2 Conclusion

In this thesis, a novel code-construction method is proposed. It attempts to design its information and parity sub-matrices independently. The advantage of the proposed parity part design is that it can be used not only (3, k) regular but also irregular LDPC codes. Therefore, this code-construction is able to construct not only (3, k) regular but also irregular LDPC codes.

In any design of LDPC codes, there are issues like rank-deficiency of parity check matrix (H), high encoding complexity of the order of $O(N^2)$ and pre-processing steps that may require computation of the order of $O(N^3)$ where N is the code length.

Since high encoding complexity is one of the critical issues, we have explained some of the existing methods of encoding in LDPC codes such as encoding by generator matrix

and encoding by parity check matrix. It has been shown that encoding by parity check matrix is preferable than generator matrix since it simplifies encoding and decoding processes by using only one matrix *H*. Encoding by inversion method is also preferable than generator polynomial since it can be applied to any random or structured LDPC codes.

Accordingly, the proposed code-construction is so designed that it has certain desirable structure in its base matrices that are expanded in two stages to have no rank-deficiency of matrix *H*, no girth of 4-cycles, no pre-processing before encoding, no singular nature of parity part (H_{par}) and low encoding complexity. Moreover, the proposed code-construction is encoded by parity check matrix using Richardson-Urbanke method that uses inversion method.

The proposed code-construction for (3, *k*) regular and irregular LDPC codes is determined by five design parameters, namely parameter *base*, parameter j_i, parameter k_i, expansion factor *L* and expansion factor L_1.

In the proposed (3, *k*) regular LDPC codes, parameter *base* and expansion factor *L* are the key factors that are able to avoid rank-deficiency, girth of 4-cycles. If necessary, it carries out column permutation for avoiding singularity in the proposed matrix *H*. Therefore, one should be careful in choosing the value of parameter *base* and expansion factor *L* in the proposed code design.

It is shown that the proposed code-construction is able to avoid pre-processing step of encoding that has, otherwise, total complexity of the order of $O(N^3)$. The value of code rate (*R*) for the proposed (3, *k*) regular LDPC codes is given by $3 \div (3 + k_i)$ while the value of code rate (*R*) for the proposed irregular LDPC codes are given by $k_i \div (3 + k_i)$.

All the parity check matrices designed and presented in this thesis are having no rank-deficiency, no pre-processing step of encoding, no singular nature in parity part (H_{par}) and low encoding complexity.

Furthermore, looking forward to test and evaluate the performance of the proposed (3, k) regular LDPC codes, the simulation model is appropriately developed with binary digit input, BPSK modulation, AWGN channel and LDPC encoder-decoder of MATLAB® 7.4. The proposed parity check matrix, H, is provided to the MATLAB® 7.4 encoder which inverts it to encode the binary data. Similarly, the decoder of MATLAB® makes use of sum-product algorithm to decode the received codeword. The code performance of the proposed (3, k) regular LDPC code is measured in terms of bit-error rate (BER), block-error rate (BLER) and Shannon limit.

Based on the results and discussion in chapter 5, the proposed (3, k) regular codes are proven to achieve code performance below 1.44 dB from Shannon limit at BER of 10^{-6} when the value of code rate, R, is greater than 0.875. Moreover, the BER value between the proposed (3, k) regular LDPC codes and theoretical upper bound of an [N, K] binary block code with soft-decision decoding is very far away even though both codes are using the same value of N, K and d_{min}.

The proposed (3, k) regular LDPC codes have comparable BER and BLER performance with other published techniques such as (3, k) regular quasi-cyclic (QC) and (3, k) regular random LDPC codes when code rates are at least $R = 0.7$ or more.

It is shown that the proposed (3, 42) regular code gets as close as 0.97 dB from Shannon limit at BER 10^{-6} for $R = 0.928$ and $N = 14364$ with encoding complexity of only $O(N + 0.0225\ N)$. The code performance results suggest that the proposed (3, k) regular codes are suitable for high code rate $R \geq 0.875$.

Even though the proposed irregular LDPC code is not able to outperform other irregular LDPC codes, the BER curve of the proposed irregular LDPC code does not flatten out for higher SNR.

6.3 Contribution of Research Work

Some contributions of this research work are as listed below:

- A novel code-construction method, in that it avoids pre-processing steps that are otherwise required in the design of (3, k) regular LDPC codes.

- Since the parity part of this code-construction can be utilized also in irregular LDPC codes, the proposed code-construction is aimed for constructing not only (3, k) regular LDPC codes but also irregular LDPC codes.

- The proposed codes are categorized as QC LDPC codes and consist of an information sub-matrix (H_{inf}) and a non-singular parity sub-matrix (H_{par}) that comes in the form of an almost lower triangular.

- The core design of the proposed code-construction utilizes expanded deterministic base matrices in three stages of code-construction. Deterministic base matrix of parity part starts with triple diagonal matrix while deterministic base matrix of information part utilizes matrix having all elements of ones. Expansion factor L_1 in the third stage of code-construction expands combination of expanded base matrices of parity part (H_p) and information part (H_i) into H_{par} and H_{inf} to construct matrix $H = [H_{inf} \mid H_{par}]$.

- Various code rates (R) are generated by maintaining the number of rows in matrix H while only changing the value of parameter k_i that represents the number of ones in each row of information part.

- The proposed design method has been used to obtain viable choices of design parameters such as *base* and expansion factors that lead to viable matrix H with no girth 4-cycles and no singularity. This has been done for parameter $base \leq 114$ and expansion factor $L \leq 35$ that are sufficient to generate code lengths of up to

15000 – a limit that the built-in encoder of MATLAB (fec.ldpcenc(H)) is not able to go beyond.

- The encoding complexity of the proposed matrix H is upper-bounded by $O(N + g^2)$ where g is the gap between matrix H and its lower triangular form with $g^2 \ll N$. The value of g is equal to $2L \times L_1$ where L and L_1 are both expansion factors. For example, for $N = 11016$ and $g^2 = 324$ only.

- The proposed (3, k) regular LDPC codes have comparable BER and BLER performance with other techniques such as (3, k) regular quasi-cyclic (QC) and (3, k) regular random LDPC codes when code rates are at least $R = 0.7$ or more.

- The proposed (3, k) regular LDPC codes are shown to achieve code performance as close as 1.44 dB from Shannon limit of SNR at bit error rate (BER) of 10^{-6} when the value of code rate is greater than $R = 0.875$ and code length $N \sim 15000$ bits. The BER value between the proposed (3, k) regular LDPC codes and theoretical upper bound of an [N, K] binary block code with soft-decision decoding is very far away even though both codes are using the same value of N, K and d_{min}.

- It is shown in this thesis that the proposed (3, 42) regular LDPC code performs as close as 0.97 dB from Shannon limit of SNR at BER 10^{-6} with encoding complexity $O(1.0225\ N)$, for $R = 0.928$ and $N = 14364$.

- The BER curve of the proposed irregular LDPC code does not flatten out for higher SNR even though the proposed irregular LDPC code is not able to outperform other irregular LDPC codes.

6.4 Future Work

There are some results of this thesis that can be improved in future and some areas that can be undertaken as new research as described below:

- It is possible to extend the proposed design methodology to other regular LDPC codes like (2, *k*), (4, *k*), (5, *k*), (6, *k*) etc and also other regular LDPC codes.

- It is also possible to design codes of length higher than 15000 bits with appropriate changes in the computer and also the codes written for the design.

- There is an opportunity to know how far the proposed (3, *k*) regular LDPC code performance be from Shannon limit at BER 10^{-6} when the value of code rate exceeds $R = 0.928$.

- Alternative methods of deterministic RCS for parity part (H_{par}) and its influence on pre-processing stage of encoding have to be adequately analyzed.

- It would also be interesting to see how other decoding algorithm influence the code performance of the proposed (3, *k*) regular LDPC codes.

- For lower code rate ($R \leq 0.5$), other construction method of information part (H_{inf}) needs to be investigated adequately for the proposed (3, *k*) regular LDPC codes.

- Testing the code performance of the proposed (3, *k*) regular LDPC codes in other practical channel conditions such as fading channel has to be adequately analyzed.

- The hardware implementation of the proposed (3, *k*) regular LDPC codes and its cost analysis are also needed to be examined in detail.

Other interesting future works of this research work are summarized below:

- The code performance comparison of the proposed irregular LDPC codes in terms of the BER and the BLER performance with other irregular LDPC codes.

- The code performance of the proposed irregular LDPC codes from Shannon limit at BER 10^{-6} when the value of code rate greater than $R = 0.875$ in AWGN channel.

- Investigating the influence of another decoding algorithm in the code performance of the proposed irregular LDPC codes.

- Utilizing another construction of information part (H_{inf}) for a better code performance of the proposed irregular LDPC codes in lower code rate ($R \leq 0.5$).

- Hardware implementation of the proposed irregular LDPC codes has to be investigated in detail.

- The proposed code construction should also target typical application scenarios like digital video broadcast – satellite (DVB-S), ultra wideband (UWB) based wireless personal area network (PAN) and other wireless networks like local area network, metropolitan area network and regional area network etc.

REFERENCES

[1] R. G. Gallager, *Low-Density Parity-Check Codes*, MIT Press, 1963.

[2] R. G. Gallager, "Low-Density Parity-Check Codes", *IRE Transaction on Information Theory*, Vol. IT-8, pp. 21-28, January 1962.

[3] Shu Lin and Daniel J. Costello, *Error Control Coding*, 2d ed., Pearson Prentice Hall, Upper Saddle River, New Jersey, 2004.

[4] David J.C, MacKay and R. M. Neal, "Near Shannon Limit Performance of Low Density Parity Check Codes", *Electron. Lett., Journal on Advances in Signal Processing*, Vol. 32, pp. 1645-1646, August 1996.

[5] N. Wiberg, *Codes and Decoding on General Graphs*, Ph.D. Dissertation No. 440, Dept. Elect. Eng. Linkoping Univ. Linkoping, Sweden, 1996.

[6] M. Sipser and D. A. Spielman, "Expander Codes", *IEEE Trans. Inf. Theory*, Vol. 42, No. 6, pp. 1710-1722, November 1996.

[7] G. A. Margulis, "Explicit Construction of Graph without Short Cycles and Low Density Codes", *Combinatorica*, Vol. 2, No.1, pp.71-78, 1982.

[8] S.Y. Chung, G.D Forney, T.J. Richardson and R.L Urbanke, "On the design of Low- Density Parity Check codes within 0.0045 dB of the Shannon Limit", *IEEE Comm. Lett*. Vol. 5, pp. 58-60, February 2001.

[9] David J.C. MacKay, "Good Error-Correcting Codes based on very Sparse Matrices", *IEEE Transactions on Information Theory,* Vol. 45, No. 2, pp. 399-431 March 1999.

[10] M. P. C. Fossorier, "Quasi-Cyclic Low Density Parity Check Codes from Circulant Permutation Matrices", *IEEE Trans. Inform. Theory*, Vol. 50, No. 8, pp. 1788-1794, August 2004.

[11] R. Michael Tanner, Deepak Sridhara, Arvind Sridharan, Thomas E. Fuja and Daniel J. Costello, "LDPC Block and Convolutional Codes Based on Circulant Matrices", *IEEE Trans. Inform. Theory*, Vol. 50, No. 12, pp. 2966-2984, December 2004.

[12] Zongwang Li, Lei Chen, Lingqi Zeng, Shu Lin and Wai H. Fong, "Efficient Encoding of Quasi-Cyclic Low Density Parity Check Codes", *IEEE Transactions on Communications,* Vol. 54, No.1, pp. 71-81, January 2006.

[13] Yu Kou, Shu Lin and Marc P. C. Fossorier, "Low Density Parity Check Codes Based on Finite Geometries: A Rediscovery and New Results", *IEEE Transactions on Information Theory*, Vol. 47, No.7, pp. 2711-2736, November 2001.

[14] Bassem Ammar, Bahram Honary, Yu Kou, Jun Xu and Shu Lin, "Construction of Low Density Parity Check Codes Based on Incomplete Block Designs", *IEEE Transactions on Information Theory*, Vol. 50, No.6, pp. 1257-1268, June 2004.

[15] Sarah. J. Johnson and Steven R. Weller, "Resolvable 2-Design for Regular Low Density Parity Check Codes", *IEEE Transactions on Communications*, Vol. 51, No.9, pp. 1413-1419, September 2003.

[16] T. J. Richardson and R. L. Urbanke, "Efficient Encoding of Low Density Parity Check Codes", *IEEE Trans. Inform. Theory*, Vol. 47, pp. 638-656, February 2001.

[17] Li Ping, W.K. Leung and Nam Phamdo, "Low Density Parity Check Codes with Semi Random Parity Check Matrix", *Electronic Letters*, Vol. 35, No. 1, pp. 38-39, January 1999.

[18] R. M. Tanner, "A [155, 64, 20] Sparse Graph LDPC Code", presented at the Recent Results Session at IEEE Intl. Sympo. on Information Theory, (Sorrento, Italy), June 2000.

[19] J. Lu and J. M. F. Moura, "Structured LDPC codes for High-Density Recording: Large Girth and Low Error Floor", *IEEE Transactions on Magnetics*, Vol. 42, No.2, pp. 208-213, February 2006.

[20] Ivana Djurdjevic, *Algebraic and Combinatorial Construction of Low Density Parity Check Codes*. PhD Thesis, University of California Davis, 2003.

[21] R. W. Hamming, "Error Detecting and Error Correcting Codes", *Bell System Technical Journal*, Vol. 29, pp. 147-160, April, 1950.

[22] D. MacKay and M. Davey, "Evaluation of Gallager Codes for Short Block Length and High Rate Applications", in *Proc. IMA Workshop Codes Systems and Graphical Models; Volume 123 of IMA Volumes in Mathematics and Its Applications*, New York: Springer-Verlag, Vol.123, pp. 113-130, 2000.

[23] T. Zhang, K. Parhi, "Joint (3, *k*) - Regular LDPC Code and Decoder/Encoder Design", *IEEE Transaction on Signal Processing*, Vol. 52, No. 4, pp. 1065-1079, April 2004.

[24] Seho Myung, Kyeongcheol Yang and Jaeyeol Kim, "Quasi-cyclic LDPC Codes for Fast Encoding", *IEEE Transactions on Information Theory*, Vol. 51, No. 8, pp. 2894-2901, August 2005.

[25] Massoud Khajeh, *A New High Performance LDPC Code for DVB-S2*, MSc Thesis, Concordia University Montreal, Quebec, Canada, April 2006.

[26] Gabofetswe Malema and Michael Liebelt,"Quasi-cyclic LDPC Codes of Column-Weight Two using Search Algorithm", *EURASIP Journal on Advance in Signal Processing*, Volume 2007, Issue 4, Article No.: 1, December 2007.

[27] Joakim Grahl Knudsen, *Randomised Construction and Dynamic Decoding of LDPC Codes*. MSc thesis, University of Bergen, Department of Informatics, November 2005.

[28] Xiao Yu Hu, Evangelos Eleftheriou and Dieter M. Arnold, "Regular and Irregular Progressive Edge-Growth Tanner Graphs", *IEEE Transactions on Information Theory*, Vol. 51, No.1, pp. 386-398, January 2005.

[29] Su Chang Chae and Y. O. Park,"Low Complexity Encoding of Regular LDPC Codes", *Vehicular Technology Conference, 2003. VTC 2003-Fall. 2003 IEEE 58th* Vol. 3, pp. 1822 - 1826, October 2003.

http://doi.ieeecomputersociety.org/10.1109/MWSCAS.2009.5236155

[30] T. J. Richardson, M. A. Shokrollahi and R.Urbanke, "Design of Capacity Approaching Irregular Low Density Parity Check Codes", *IEEE Trans. Inform. Theory*, Vol. 47, No. 2, pp. 619-637, February 2001.

[31] Xu Xia, Moon Ho Lee, Choi Seung Je, Moon Myung Ryong,"Performance Analysis and Design of (3, ρ) - Regular Quasi-Cyclic LDPC codes", in *Proc. International Conference on Wireless Networks, Communication and Mobile Computing*, pp. 1187-1191, Maui, Hawaii, USA, July 2005.

[32] Sunghwan Kim, Jong Seon No, Habong Chung, Dong Joon Shin, "On the Girth of Tanner (3, 5) Quasi-Cyclic LDPC Codes", *IEEE Trans. Inform. Theory*, Vol. 52, No. 4, pp. 1739-1744, April 2006.

[33] Joachim Hagenauer, Frank Burkert and Helmut Nickl, "The Race to Shannon's Limit: Discipline High-Rate Codes". *Proc. Int. Symp. Turbo Codes and Related Topics*, Brest, France, pp. 239–242, September 1997.

[34] Ian A. Glover and Peter M. Grant, *Digital Communications*, Second Edition, Pearson Prentice Hall, 2004.

[35] H. Tang, J. Xu,Y. Kou, S. Lin and K. Abdel Ghaffar, "On Algebraic Construction of Gallager and Circulant Low Density Parity Check Codes", *IEEE Trans. Inform. Theory*, Vol. 50, pp. 1269-1279, June 2004.

[36] M. Luby, M. Mitzenmacher, A. Shokrollahi, D. Spielman and V. Stemann, "Practical Loss-Resilient Codes", in *Proc. 29th Annu. ACM Symp. Theory of Computing*, Dallas, TX, USA, pp. 150-159, May 1998.

[37] M. Luby, M. Mitzenmacher, A. Shokrollahi, D. Spielman, and V. Stemann, "Efficient Erasure Correcting Codes", *IEEE Trans. Inform. Theory*, Vol. 47, No.2, pp. 569-584, February 2001.

[38] M. Luby, M. Mitzenmacher , M. A. Shokrollahi, D. A. Spielman , "Analysis of Low Density Codes and Improved design using Irregular Graphs", in *Proc. 30th Annu. ACM. Symp. Theory of Computing* 1998, Dallas, TX, USA, pp. 249-258, May 1998.

[39] M. Luby, M. Mitzenmacher , M. A. Shokrollahi, D. A. Spielman, "Improved Low-Density Parity-Check Codes using Irregular Graphs and Belief Propagation", in *Proc. 1998 IEEE Int. Symp. Information Theory*, Cambridge, MA, pp. 117, August 1998.

[40] D. MacKay, S. Wilson and M. Davey, "Comparison of Construction of Irregular Gallager Codes", *IEEE Trans. Comm.*, Vol. 47. pp. 1449-1454, October 1999.

[41] Michael Yang, William E. Ryan and Yan Li,"Design of Efficiently Encodable Moderate-Length High Rate Irregular LDPC Codes", *IEEE Transactions on Communications*, Vol. 52, pp. 564-571, April 2004.

[42] R. Echard and S. Chang, "The π–rotation Low Density Parity Check Codes", in *Proc. IEEE GLOBECOM 2001*, San Antonio, TX, USA , Vol. 2, pp. 980-984, November 2001.

[43] Hughes Network System, DVB-S2 Coding Standard Proposal, Technical Report, DVB Standardization Committee, Geneva, Switzerland, January 2003.

http://ieeexplore.ieee.org/iel5/7693/30028/01374941.pdf?arnumber=1374941

[44] European Telecommunication Standards Institute (ETSI) Digital Video Broadcasting (DVB), Second Generation Framing Structure, Channel Coding and Modulation Systems for Broadcasting, Interactive Services, News Gathering and Other Broadband Satellite Applications (DVB-S2), ETSI Std. EN 302.307, June 2004. http://www.dvb.org/technology/standards/index.xml

[45] IEEE P802.16e™, "Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems," *IEEE 802.16 document*, February 2005.

http://ieeexplore.ieee.org/ISOL/allstandards.jsp

[46] IEEE P802.11.n™/D1.02, "Draft Amendment to Standard Information Technology Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Enhancements for Higher Throughput", *IEEE 802.11 document*, July 2006.

http://ieeexplore.ieee.org/ISOL/allstandards.jsp

[47] Sarah J. Johnson and Steven R. Weller,"A Family of Irregular LDPC Codes with Low Encoding Complexity", *IEEE Communications Letters*, Vol. 7, No. 2, pp. 79-81, February 2003.

[48] Jeong Ki Kim, S. P Balakannan, Moon Ho Lee and Chang Joo Kim, "Low Complexity Encoding of LDPC codes for High Rate and High Speed Communication", *1st International Conference of Distributed Framework and Applications (DFmA)*, Penang, Malaysia, pp. 189-193, October 2008.

[49] S. Johnson and S. R. Weller, "Regular Low Density Parity Check Codes from Combinatorial Design", *Proc. 2001 IEEE Information Theory Workshop*, Cairns, Australia, pp. 90-92, September 2001.

[50] Gianluigi Liva, William E. Ryan and Marco Chiani, "Quasi-Cyclic Generalized LDPC Codes with Low Error Floors", *IEEE Transactions on Communications*, Vol. 56, No.1, pp. 49-57, January 2008.

[51] Masaya Fujisawa and Shojiro Sakata, "A Class of Quasi-Cyclic Regular LDPC Codes from Cyclic Difference Families with Girth 8", *in Proc. International Symposium on Information Theory 2005 (ISIT 2005)*, Adelaide, Australia, pp. 2290-2294, September 2005.

[52] Hanghang Qi and Norbert Goertz, "Low Complexity Encoding of LDPC Codes: A New Algorithm and its Performance", *9th International Symposium on Communication Theory and Application*, July 2007.

http://publik.tuwien.ac.at/files/pubDat_166941.pdf.

[53] Michael Yang, William E. Ryan and Yan Li, "Design of Efficiently Encodable Moderate-Length High-Rate Irregular LDPC Codes", *IEEE Transactions on Communications*, Vol. 52, No. 4, pp. 564-571, April 2004.

[54] Proakis, J. G., *Digital Communications*, 4th ed., McGraw-Hill, 2001.

[55] A. Nimbalker, Y. Blankenship, B. Classon, "Turbo-like Decoding Algorithm for Structured LDPC codes", *IEEE International Symposium on Information Theory*, pp. 1708 – 1712, 9-14 July 2006.

PUBLICATIONS

1. Silvia Anggraeni and Varun Jeoti "Efficient Code Construction of (3, *k*) Regular LDPC Codes", IEEE 3rd International Symposium on Information Technology 2008 (ITSIM'08), Vol. 3, pp: 1-7, 26-28 August 2008. Kuala Lumpur, Malaysia. IEEE Catalog Number: CFP0833E-PRT, ISBN: 978-1-4244-2327-9.

2. Silvia Anggraeni and Varun Jeoti "Performance Analysis of Efficient (3, *k*) Regular LDPC Codes", Proceedings of Student Conference on Research and Development (SCOReD 2008), 26-27 Nov. 2008, Johor, Malaysia.
   IEEE Catalog Number: CFP08526-CDR, ISBN: 978-1-4244-2869-4

APPENDIX
FEASIBLE VALUES OF PARAMETER *BASE* AND EXPANSION *L*

Introduction

As mentioned in chapter 3 section 3.2.1, this is an appendix of feasible values of parameter *base* and expansion factor *L* with given choice of deterministic RCS that give matrix *H* with no rank-deficiency and girth of 4-cycles.

Appendix A: Parameter *Base*

This section explores feasible values of parameter *base* investigated by producing the proposed matrix *H* using a given value of parameter *base*, seven methods of deterministic RCS according to Table 3.2 and constant parameters of L, L_1, j_i, k_i that are taken randomly. The range value of parameter *base* is also chosen randomly in order to investigate all valid values of parameter *base* that gives matrix *H* without rank deficiency and girth of length 4.

Parameter *base* is applied as a basic design parameter of parity part in the proposed codes. The proposed parity part begins by a triple diagonal base matrix in the size of (*base* × *base*), puts some constraints to construct a deterministic form of (3, 3) regular LDPC code and expands it with expansion factor *L* to construct parity part (H_p) in the first stage of code- construction having characteristics of no rank-deficiency of matrix *H*, no pre-processing steps of encoding, low encoding complexity and non-singular parity part.

Each method of deterministic RCS in Table 1 is tested to produce the proposed matrix H by applying variable value of parameter *base* in the range value of *base* = 3 until *base* = 55 using constant parameters of $L_1 = 1$, $L = 6$, $j_i = 3$ and $k_i = 3$. The range of parameter *base* that is *base* = 3 until *base* = 55 is chosen since this range of parameter *base* gives the value of code length $N < 15000$ bits that is code length $N = 36$ until $N = 660$ bits. Then, the proposed matrix H is provided to the MATLAB® 7.4 encoder which inverts H to encode the binary data.

As stated in section 5.2.1, there is a possibility that matrix $(ET^{-1}B + D)$ is singular even though rank deficiency and girth of 4-cycle are not found in the proposed matrix H. Removing singularity is solved by permuting one column of matrix H_{inf} and one column of matrix H_{par} until the determinant of matrix $(ET^{-1}B + D)$ is not zero anymore.

Since we focus to avoid any task or additional computation before actual encoding, we prefer to use the proposed matrix H that has no column permutation at all to avoid singularity of matrix $(ET^{-1}B + D)$ in our simulation.

Derived from Table 1, the smallest total value of parameter *base* that produces encoding of matrix H without column permutation is given by deterministic RCS No.2 $((x - 1) \times y)$ mod L derived from [23].

Applying the value of parameter *base* in the process of code design is useful in eliminating rank-deficiency, girth of 4-cycles and column permutation for avoiding singularity in the proposed matrix H. Therefore, parameter *base* is one factor that is able to overcome those issues in the proposed matrix H.

Table 1: Feasible values of parameter *base* for designing H without rank-deficiency and girth of length 4 having constant parameter $L_1 = 1$, $L = 6$, $j_i = 3$ and $k_i = 3$.

No	Methods of Deterministic RCS in Parity Part	Range Values of Parameter *Base* *Base* = 3 until *Base* = 55	Comment	Total Valid Choices of Parameter *Base*
1	$(x \times y) \bmod L$	9, 10, 12, 13, 14, 15, 16, 17, 19, 20, 21, 24, 25, 26, 27, 28, 31, 33, 34, 35, 37, 38, 40, 41, 42, 44, 45, 47, 48, 49, 50, 52, 54, 55	Do column permutation of H to avoid singularity of matrix $(ET^{-1}B + D)$ denoted by Φ.	34
		8, 11, 18, 22, 23, 29, 30, 32, 36, 39, 43, 46, 51, 53	No column permutation of matrix H	14
2	$((x-1) \times y) \bmod L$ This equation is based on [23].	9, 12, 21, 24, 27,30, 33, 36, 42, 45, 48, 51, 54	Do column permutation of H to avoid singularity of matrix Φ.	13
		15, 18, 39	No column permutation of matrix H	3
3	$((x-2) \times y) \bmod L$	6, 7, 9, 10, 12, 13, 14, 15, 17, 19, 20, 21, 23, 24, 26, 27, 28, 29, 31, 33, 34, 35, 38, 40, 41, 42, 43, 44, 45, 47, 48, 49, 52, 54, 55	Do column permutation of H to avoid singularity of matrix $(ET^{-1}B + D)$.	35
		8, 11,16,18, 22, 25, 30, 32, 36, 37, 39, 46, 50,51,53	No column permutation of matrix H	15
4	$((x-3) \times y) \bmod L$	7, 9, 10, 12, 13, 14, 15, 16, 17, 19, 20, 21, 24, 25, 26, 27, 28, 31, 33, 34, 35, 37, 38, 40, 41, 42, 44, 45, 47, 48, 49, 50, 52, 54, 55	Do column permutation of H to avoid singularity of matrix $(ET^{-1}B + D)$.	35
		8, 11, 18, 22, 23, 29, 30, 32, 36, 39, 43, 46, 51, 53	No column permutation of matrix H	14
5	$(x \times (y-1)) \bmod L$	10, 12, 14, 16, 17, 19, 20, 21, 23, 24, 25, 26, 27, 28, 29, 31, 33, 34, 35, 36, 37, 38, 40, 41, 42, 44, 45, 47, 48, 49, 52, 54, 55	Do column permutation of H to avoid singularity of matrix $(ET^{-1}B + D)$.	34
		9, 11, 13, 15, 18, 22, 30, 32, 39, 43, 46, 50, 51, 53	No column permutation of matrix H	14
6	$(x \times (y-2)) \bmod L$	7, 9, 13, 14, 15, 17, 19, 20, 21, 26, 27, 31, 32, 33, 35, 37, 38, 41, 45, 47, 49, 55	Do column permutation of H to avoid singularity of matrix Φ.	22
		8, 11, 23, 25, 29, 39, 43, 44, 50, 51, 53	No column permutation of matrix H	11
7	$(x \times (y-3)) \bmod L$	9, 10, 12, 13, 14, 15, 16, 17, 19, 20, 21, 24, 25, 26, 27, 28, 31, 33, 34, 35, 37, 38, 40, 41, 42, 44, 45, 47, 48, 49, 50, 52, 54, 55	Do column permutation of H to avoid singularity of matrix $(ET^{-1}B + D)$.	34
		8, 11, 18, 22, 23, 29, 30, 32, 36, 39, 43, 46, 51, 53	No column permutation of matrix H	14

Appendix B: Expansion Factor *L*

This section investigates feasible value of expansion factor L investigated by producing the proposed matrix H using a given value of expansion factor L, seven methods of deterministic RCS according to Table 3.2 and constant parameters of *base*, L_1, j_i, k_i that are taken randomly. The range value of expansion factor L is also chosen randomly in order to investigate total valid values of expansion factor L that gives matrix H without rank deficiency and girth of length 4.

Expansion factor L is utilized to expand each non-zero of base matrix of parity part at position (x, y) by ($L \times L$) identity matrix with deterministic right cyclic shift (RCS). Meanwhile, each zero of base matrix of parity part is also expanded by ($L \times L$) zeros matrix.

Each method of deterministic RCS in Table 2 is tested to produce the proposed matrix H by applying variable value of expansion factor L in the range value of $L = 3$ until $L = 35$ using constant parameters of $L_1 = 1$, *Base* = 39, $j_i = 3$ and $k_i = 1$. The range of expansion factor L that is $L = 3$ until $L = 35$ is chosen since this range of expansion factor L gives the value of code length $N < 15000$ bits that is code length $N = 156$ until $N = 1820$ bits. Then, the proposed matrix H is provided to the MATLAB® 7.4 encoder which inverts H to encode the binary data.

All seven methods of deterministic RCS in Table 2 yield bigger value of expansion factor L that produces encoding of matrix H without column permutation value than Table 1. The reason is that the value of constant parameter *base* is taken by *base* = 39 that gives no column permutation of matrix H for all seven methods of deterministic RCS in Table 1.

Table 2: Feasible values of expansion factor L for designing H without rank-deficiency and girth of length 4 using constant parameter L_1=1, *base*=39, j_i=3, k_i = 1.

No	Methods of Deterministic RCS in Parity Part	Range Values of Expansion Factor L $L = 3$ until $L = 35$	Comment	Total Valid Choices of Expansion Factor L
1	$(x \times y) \bmod L$	7, 14, 21, 28, 35	Do column permutation of H to avoid singularity of matrix $(ET^{-1}B + D)$.	5
		3, 4, 6, 8, 9, 10, 11, 12, 13, 15, 16, 18, 19, 20, 22, 23, 24, 25, 26, 27, 29, 30, 31, 32, 33, 34	No column permutation of matrix H	26
2	$((x-1) \times y) \bmod L$ This equation is based on [23].	3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35	No column permutation of matrix H	32
3	$((x-2) \times y) \bmod L$	3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35	No column permutation of matrix H	33
4	$((x-3) \times y) \bmod L$	7, 14, 21, 28, 31, 35	Do column permutation of H to avoid singularity of matrix $(ET^{-1}B + D)$.	6
		3, 5, 6, 9, 10, 11, 12, 13, 15, 17, 18, 19, 20, 22, 23, 24, 25, 26, 27, 29, 30, 32, 33, 34	No column permutation of matrix H	24
5	$(x \times (y-1)) \bmod L$	3, 5, 6, 7, 9, 10, 12, 13, 14, 15, 17, 18, 19, 20, 21, 23, 24, 25, 26, 27, 28, 29, 30, 31, 33, 34, 35	No column permutation of matrix H	27
6	$(x \times (y-2)) \bmod L$	4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35	No column permutation of matrix H	32
7	$(x \times (y-3)) \bmod L$	7, 14, 21, 28, 35	Do column permutation of H to avoid singularity of matrix $(ET^{-1}B + D)$.	5
		3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 22, 24, 25, 26, 27, 30, 31, 32, 33, 34	No column permutation of matrix H	26

Based on Table 2, there are four methods of deterministic RCS that produce the proposed matrix H with no column permutation for expansion factor L in the range of $L = 3$ until $L = 35$. These four methods of deterministic RCS are $((x - 1) \times y) \bmod L$, $((x - 2) \times y) \bmod L$, $(x \times (y - 1)) \bmod L$ and $(x \times (y - 2)) \bmod L$.

Derived from Table 2, the highest total value of expansion factor L that produces encoding of matrix H without column permutation is given by deterministic RCS No.3 that is $((x - 2) \times y) \bmod L$.

According to Table 1 and Table 2, the crucial parameters in eliminating rank-deficiency, girth of 4-cycles and column permutation to avoid singularity in the proposed matrix H are parameter *base* and expansion factor L.

Summary

This appendix gives feasible values of parameter *base* and expansion factor L with given choice of deterministic RCS that give matrix H with no rank-deficiency and girth of 4-cycles. It is shown that expansion factor L gives larger feasible values of matrix H with no rank-deficiency and girth of 4-cycles than parameter *base* since the chosen constant parameter *base* is *base* = 39 that gives no column permutation of matrix H for all seven methods of deterministic RCS in Table 1.

The crucial parameters in eliminating rank-deficiency, girth of 4-cycles and column permutation to avoid singularity in the proposed matrix H are parameter *base* and expansion factor L