Efficient design of LDPC code using circulant matrix and eIRA code

Seul-Ki Bae

The Graduate School

Yonsei University

Department of Electrical and Electronic Engineering

Efficient design of LDPC code using circulant matrix and eIRA code

by

Seul-Ki Bae

A Dissertation Submitted to the

Graduate School of Yonsei University

in partial fulfillment of the

requirements for the degree of

MASTER of SCIENCE

Supervised by

Professor Hong-Yeop Song, Ph.D.

Department of Electrical and Electronic Engineering
The Graduate School

YONSEI University

December 2004

This certifies that the dissertation of
Seul-Ki Bae is approved.

Thesis Supervisor: Hong-Yeop Song

Kwang-Soon Kim

Sae-Joon Kim

The Graduate School
Yonsei University
December 2004

감사의 글

무한한 학문의 길에서 가야할 길을 잃고, 무엇을 해야할지 고민하던 제게 큰 가르침을 주시며 본 논문의 작성을 위해 지도해 주신 송홍엽 교수님께 존경과 감사를 드립니다. 그리고 여러모로 부족한 논문을 하나하나 지적해 주시며 보다 나은 방향으로 가르침을 주신 김광순 교수님과 김세준 교수님께 감사를 드립니다.

2년 이라는 짧은 석사과정을 하는 동안 정들었던 연구실 선후배님에게도 감사를 드립니다. 언제나 꿋꿋하게 열심히 하는 모습을 보면서 존경했던 은유창 선배님, 그리고 항상 같이 밤을 새며 도움을 주며, 야참도 같이 먹었던 신민호 선배님, 연구실의 규율을 잘 통제해주셨던 홍윤표 선배님, 친형처럼 다정한 대선형, 언제나 어린 왕자 기훈형, 본 논문을 위해 큰 힘이 되어 주었던 준성형, 바른 생활 사나이 영준형, 연구실 지킴이 석용선배, 입학과 함께 동고동락을 함께 했던 동규형, 넉넉한 웃음으로 따스하게 대해주었던 석훈형, 천진난만한 어린 왕자 진석형, 자신의 일을 묵묵히 잘하는 기천이, 연구실 멋쟁이 성준이, 척척박사 비웅이 모두 제게 연구실 생활을 힘차고 생기있게 할 수 있는 원동력이었습니다. 앞으로 연구실 생활을 하게 될 신입생들도 선후배님들과 함께 재밌게 생활하길 바랍니다.

그리고 힘든 서울 생활을 정성껏 도와주셨던 이모부와 이모, 그리고 부산에서 음으로 양으로 저를 믿고 후원해준 할머니, 이모, 삼촌들께도 깊은 감사를 드립니다.

미운정 고운정으로 따뜻하게 보살펴줬던 우리 예리 누나에게 감사드리며, 마지막으로 못난 아들을 위해 모든 것을 뒤로 하시고 사랑으로 보살펴 주신 아버님, 어머님께 무한한 존경과 감사와 함께 이 논문을 바칩니다.

2004년 12월

배 슬 기 드림

Contents

List of Figures

List of Tables

ABSTRACT

Efficient Encoder Design of LDPC code using circulant matrix and eIRA code

Seul-Ki Bae
Department of Electrical and Electronic Eng.
The Graduate School
Yonsei University

For reliable data transmission or storage system, channel coding is necessary. LDPC codes are defined as parity-check matrices that consist almost entirely of zeros and small number of ones. Like turbo codes, LDPC codes can achieve near Shannon limit. But unlike very simple encoder of turbo codes, that of LDPC code is much complex.

In this dissertation, we concentrate on reducing the complexity for efficient encoder. We design structural LDPC code using circulant matrix and permutation matrix and eIRA code. It is possible to design low complex encoder by using shift register and differential encoder and interleaver than general LDPC encoder that use matrix multiplication and addition operation. The code designed by this structure shows better performance than random code. And the proposed method can considerably reduce a number of XOR gates. And the proposed code shows a property such that there is less degradation than randomly generated parity-check matrix when the codeword length be-

comes shorter. However this code shows error floor effect and performance degradation in high SNR region. We can improve its performance by design short cycle free codes.

Key words : LDPC codes, efficient encoder, circulant matrix, row permutation matrix, eIRA codes

Chapter 1

Introduction

1.1 History

In recent year, there has been an increasing demand for efficient and reliable digital data transmission and storage systems. In 1948, Shannon demonstrated in a landmark paper that, if proper encoding method is used, errors induced by a noisy channel can be reduced to any desired level without sacrificing the rate of information transmission as long as the information rate is less than capacity of the channel [1]. The use of coding for error control has become an important part in the design of modern communication and data storage systems.

There are two major types of code, one is convolution codes and the other is block codes. Convolutional codes operate on serial data and block codes operate on message blocks. Convolution encoder contains memory, and the encoder outputs at any given unit time depend not only on the inputs at that time but also on some number of previous inputs. The encoder for a block codes divides the information sequence into message blocks and transforms each message independently in to discrete symbols, called code-word. Turbo codes are one of many applications of convolution codes and LDPC codes

are one of those of block codes.

Low-density parity-check(LDPC) codes [2] were first proposed by Gallager in the 1960's. LDPC codes are defined as parity-check matrices that consist almost entirely of zeros and small number of ones. However, Gallager's LDPC code was forgotten for the next 30 years until the discovery of Turbo codes [3]. And LDPC codes were rediscovered by MacKay and Neal [4]. Like turbo codes, LDPC codes can achieve near Shannon limit error performance, and represent a very promising prospect for error control coding. The past few years have brought many developments in this area.

1.2 Motivation

Since the discovery of LDPC codes, a great deal of research effort has been expended in construction, decoding, performance analysis, structural study, and applications of these codes.

But an encoding of LDPC codes is more complex than turbo codes which can be encoded in linear time. In general, an encoding process of LDPC codes is done by matrix multiplication operation. However, though parity-check matrix has very small number of ones, corresponding generator matrix has much more number of 1's than parity-check matrix. So if we use only matrix multiplication operation to encode LDPC codes, there needs a great number of calculations. So various methods for constructing LDPC codes to reduce encoding complexity have been proposed and devised [5] [6].

The aim of this paper is to reduce the complexity of LDPC encoder. Thus we propose parity-check matrix to have structural design using circulant matrix, permutation matrix and eIRA codes. And we will show that the proposed code with moderate length

and high code rate is superior or equivalent in performance compared with randomly constructed LDPC codes.

1.3 Overview

In chapter 2, we introduce Gallager's LDPC codes and its construction method. In addition, encoding and decoding algorithm of LDPC codes are introduced. In chapter 3, we introduce some structural designs of codes which can reduce the encoding complexity.In chapter 4, we modify parity-check matrix using circulant matrix and permutation matrix. And its performance results are shown and analyzed. Finally, all those results of this paper are summarized and some discussions follow in chapter 5 .

Chapter 2

LDPC codes

In this chapter we introduce Gallager's LDPC codes and basic construction method. The relation of generator matrix and parity-check matrix is introduced. And encoding and decoding algorithm of LDPC codes are introduced.

2.1 Low-Density Parity-Check codes

Linear block codes use a generator matrix **G** to map message **s** to transmitted blocks **c**, also known as *codeword*. They have and equivalent description in terms of a related parity-check matrix **H**. All codewords satisfy $\mathbf{cH}^T = 0$.

Low-density parity-check(LDPC) codes are a class of linear error-correcting codes [2]. As their name suggests, LDPC codes are defined in terms of parity-check matrices **H** that consist almost entirely of zeros. Gallager defined (n, p, q) LDPC codes to have a block length n and a parity-check matrix with exactly p ones per column and q ones per row, where $p \geq 3$. In Gallager's construction in Figure 2.1, the lower two blocks are column permutations of the upper block.

There exist also irregular LDPC codes [7, 8]. The parity check matrix of an irregular

1	1	1	1	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	1	1	1	1	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	1	1	1
1	0	0	0	1	0	0	0	1	0	0	0	1	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	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	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	1	0	0	0	1	0	0	0	1	0	0	0
1	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0
0	1	0	0	0	0	1	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	0	1
0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0
0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1

Figure 2.1: Example of Gallager's parity-check matrix for a (20, 3, 4) LDPC codes.

LDPC code has checks with different weights. As a result, neither the degrees of the bits in the bipartite graph nor the degrees of the checks are constant. The irregular bipartite graph is deliberately determined by some rules. It has been reported that irregular LDPC codes can achieve better performance than regular LDPC codes.

Another special family of LDPC codes can be constructed based on finite geometries [9]. Theses codes are either cyclic or quasi-cyclic and therefore their encoding can be implemented with simple linear feedback shift registers.

2.2 Construction of LDPC codes

The weight of a vector is the number of non-zero components in that vector. The parity-check matrix **H** is constructed randomly, while constraining the distribution of row and column weights to be as uniform as possible. We also require that no two columns of **H** have overlap greater than 1 to reduce the probability of low weight codewords.

2.2.1 Gallager's construction

Gallager imposed a fixed column weight j and a fixed row weight k in his work. The parity-check matrix was divided horizontally into j equal size submatrices, each containing a single '1' in each column. Without loss of generality the first submatrix was constructed in some predetermined manner. the subsequent submatrices were random column permutations of the first. An example of a matrix constructed in this way is shown in Figure 2.1

2.2.2 MacKay's construction

MacKay [8] was interested in keeping to a minimum the number of short cycles in the bipartite graph representing the parity-check matrix. The presence of short cycles can create difficulties for the belief propagation decoder.

- Construction 1A

  This is the basic construction, in which we have a fixed weight per column t and construct the matrix at random keeping the weight per row as uniform as possible, and overlap between any two columns no greater than 1.

- Construction 2A

  As per 1A, except up to $m/2$ of the columns have weight 2. These weight 2 columns are constructed in the form of two indentity matrices of size $m/2 \times m/2$, one above the other.

- Construction 1B, 2B

  Some carefully chosen columns form a 1A or 2A matrix are deleted, so that the bipartite graph of the matrix has no short cycles of length less than some length l.

2.3 Generator and Parity-check matrix

2.3.1 Generator matrix

LDPC codes of block length n and rate k/n can be described by a generator matrix **G** of dimension $k \times n$ that describes the mapping from source words **m** to codewords **c** $:=$ **mG** (where the vectors m, c are column vectors). It is common to consider **G** in systematic form, **G** $= [\mathbf{I}_k|\mathbf{P}]$ so that the first k transmitted symbols are the source symbols. The notation $[A \mid B]$ indicates the concatenation of matrix A with matrix B and $\mathbf{I}_k$ represents the $k \times k$ identity matrix. The remaining $m = n - k$ symbols are parity-checks.

2.3.2 Parity-check matrix

LDPC codes are also described by a parity-check matrix **H** of dimension $m \times n$, where $m = n - k$. If the corresponding generator matrix **G** is written in systematic form as above, then H has the form [-**P** | $\mathbf{I}_m$]. Otherwise, in case **H** is not in systematic form we perform Gaussian elimination using row operations and reordering of columns to derive a parity-check matrix as above. Note that for codes over finite fields $GF(2^p)$

$\mathbf{P} = -\mathbf{P}$. Each row of the parity-check matrix describes a linear constraint satisfied by all codewords. $\mathbf{GH}^T = \mathbf{0}$ and hence the parity-check matrix can be used to detect errors in the received vector:

$$\mathbf{rH}^T = (\mathbf{c} + \mathbf{n})\mathbf{H}^T = \mathbf{mGH}^T + \mathbf{nH}^T = \mathbf{nH}^T := \mathbf{z}$$

Here we have introduced the syndrome vector **z**. If the syndrome vector **z** is null, we assume there have been no errors. Otherwise, the decoding problem is to find the most likely noise vector n that explains the observed syndrome given the assumed properties of the channel.

2.4 Encoding and Decoding

LDPC codes are defined by a sparse parity-check matrix. When parity-check matrix **H** = [-**P** | $\mathbf{I}_m$] is given, the corresponding generator matrix is **G** = [$\mathbf{I}_k$ | **P**]. But generator matrix is never sparse matrix, as shown in Figure 2.2. In Figure 2.2, the submatrix **P** of generator matrix **G** is a very dense. In general, encoding process are done by matrix multiplication and addition in block codes. So if we encode **m** through generator matrix **G** to get codeword, a large number of operations are needed. Whereas turbo codes can be encoded in linear time, a straightforward encoder implementation for an LDPC code has complexity quadratic in the block length. So many research to reduce the complexity of encoder has been achieved [5] [9] [10] [11] .

To decode LDPC codes, an iterative probabilistic decoding algorithm known as a Sum-Product or Belief Propagation Algorithm can be used. At each step the posterior probability of the value of each noise symbol is estimated, given the received signal and

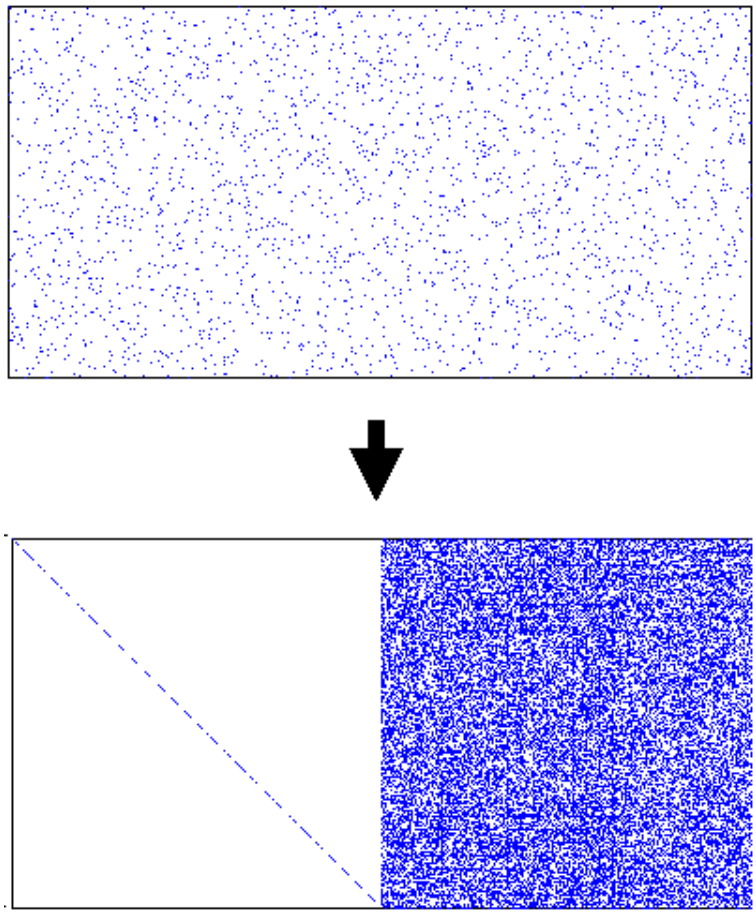

Figure 2.2: Parity-check matrix and corresponding Generator matrix

the channel properties. The process is best viewed as a message passing algorithm on the bipartite graph defined by **H** which have two sets of nodes. In Figure 2.3 there are nodes representing variable symbol and nodes are representing check symbols. Variable node and check node are connected if the corresponding matrix entry H_{ij} is 1's. At each step of the iteration, each variable node sends probability messages to check nodes. Also each check node sends probability messages to variable nodes. After this each step, we

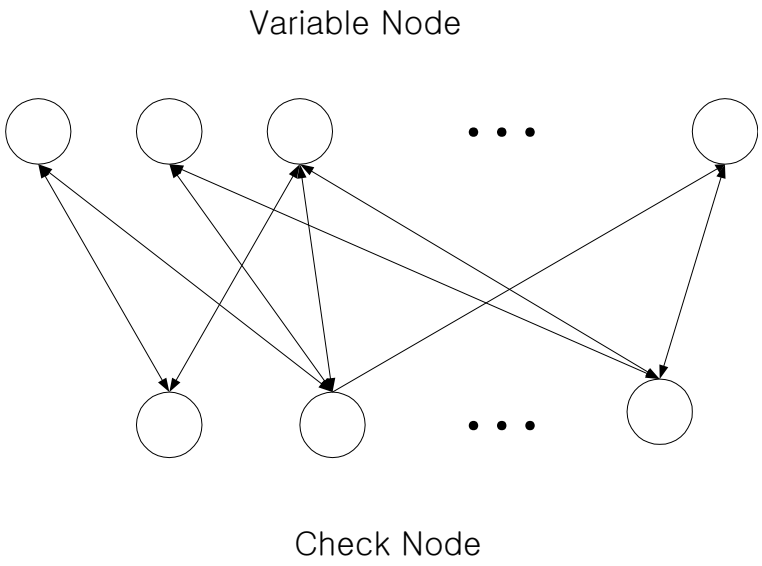

Figure 2.3: Bipartite graph representing a parity-check matrix

examine the messages and do a tentative decoding. Like this, decoding algorithm consists of iteratively updating these message until tentative decoding satisfies the observed syndrome vector or a preset maximum number of iterations is reached.

Chapter 3

Design of Structural Codes

In this chapter we will introduce some structural codes. These codes can be encoded efficiently. Cyclic code and quasi-cyclic code can be encoded using shift register which is simplest digital circuit. And irregular repeat-accumulate(IRA) code can be encoded using differential encoding.

3.1 Cyclic codes

A linear block code C is said to be cyclic code if every codeword $c = (c_0, c_1, \cdots, c_{n-2}, c_{n-1} \in C$, there is also a codeword $c' = (c_{n-1}, c_0, c_1, \cdots, c_{n-2}) \in C$ [12]. Cyclic codes possess full cyclic symmetry; that is, cyclically shifting a codeword any number of symbol positions, either to the right or to the left, results in another codeword. This symmetry structure makes it possible to implement the encoding and decoding of cyclic codes with simple shift registers and logic circuits.

Example 3.1 Binary Cyclic Codes of Length 7

Consider the case $g(x) = x^4 + x^3 + x^2 + 1$ and the corresponding parity polynomial is $h(x) = x^3 + x^2 + 1$. Then generator matrix G and parity-check matrix H is shown

below, respectively.

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

and,

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

They are derived from the generator polynomial $g(x)$ and parity-check polynomial $h(x)$. And the message polynomials consist of all binary polynomials of degree less than or equal to 2. We let message polynomial $m(x) = x^2 + 1$.

Shift registers are among the simplest of digital circults, consisting of a collection of flip-flops connected in series. They are thus operable at speeds quite close to the maximum speed possible for a single gate using a given device technology.

The product $m(x)g(x)$ can be computed as a weighted sum of cyclic shifts of $g(x)$. The corresponding Shift Register circuit is shown in Figure 3.1. The coefficients of $m(x)$

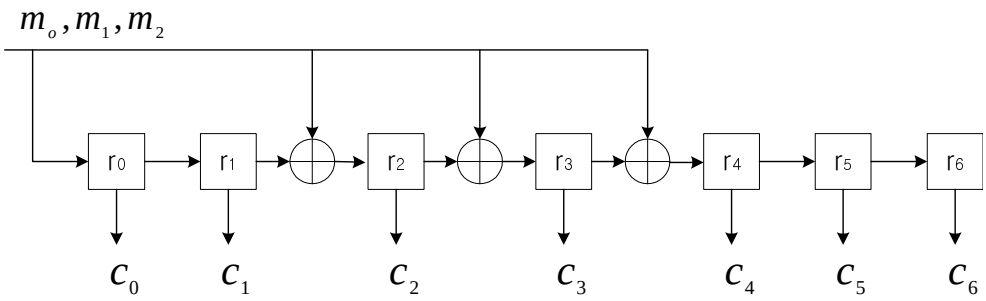

Figure 3.1: Encdoer of (7, 3) cyclic code using Shift Register circuit

Table 3.1: Shift-Register Cell contents during encoding of $m(x) = x^2 + 1$

SRcells	r_0	r_1	r_0	r_0	r_0	r_0	r_0
Initialstate	0	0	0	0	0	0	0
$m_2 = 1$	1	0	1	1	1	0	0
$m_1 = 0$	0	1	0	1	1	1	0
$m_0 = 1$	1	0	0	1	0	1	1
$codeword$	1	0	0	1	0	1	1

are fed into the shift register in descending order of index. Each time a new coefficient is placed on the input line, the shift register clock is pulsed, and the contents of the shift register cells shifted one cell to the right. When the final coefficient(m_0) has been fed into the shift register, the shift register cells contain the codeword **c**

Figure 3.1 shows the polynomials multiplier encoder for the (7, 3) binary code. In Table 3.1 the contents of the shift register cells int this encoder are traced during the encoding of the message block $m = (101)$. At the end of the encoding process the shift register cells contain the codeword **c** $= (1001011)$.

3.2 Circulant Matrix

An n-tuple **c** is said to be primitive if its cycle span is n. A primitive n-tuple and its $n-1$ cyclic shifts give n different n-tuples. For a primitive **c** we can form an $n \times n$ square matrix whose rows are **c** and its $n-1$ cyclic shifts $\mathbf{c}^{(1)}, \mathbf{c}^{(2)}, \ldots, \mathbf{c}_{(n-1)}$. A circulant

matrix of order n is a square matrix of the form

$$\begin{bmatrix} c_0 & c_1 & \cdots & c_{n-1} \\ c_{n-1} & c_0 & \cdots & c_{n-2} \\ \vdots & \vdots & & \vdots \\ c_1 & c_2 & \cdots & c_0 \end{bmatrix}$$

The first row is **c** and then every row in the matrix is the cyclic shift of the row above it and the first row is the cyclic shift of the last row. Such a square matrix is called a circulant matrix or simply circulant. We can construct LDPC codes using this circulant matrix [6, 13–15].

3.3 Quasi-Cyclic codes

There are other linear block codes that do not possess full cyclic symmetry but do have partial cyclic structure, namely, quasi-cyclic codes. A quasi-cyclic code is a linear code for which cyclically shifting a codeword a fixed number $n_0 \neq 1$ (or a multiple of n_0) of symbol positions either to the right or to the left results in another codeword. It is clear that for $n_0 = 1$, a quasi-cyclic code is a cyclic code. The integer n_0 is called the shifting constraint. It is clear that the dual of a quasi-cyclic code is also quasi-cyclic. We can also construct LDPC codes using quasi-cyclic codes [15] [16].

3.3.1 Circulant Decomposition and Quasi-Cyclic codes

Given a circulant matrix, it is possible to decompose it into an array of circulant matrix of the same size. Then this array gives a quasi-cyclic code. Consider an $n \times n$ circulant **G** with column (or row) weight δ. Since the column and row weights of a circulant matrix are the same, for simplicity, we say that **G** has weight δ. For $1 \leq t \leq \delta$, let

$w_1, w_2, \cdots, w_t$ be a set of positive integers such that $1 \leq w_1, w_2, \cdots, w_t \leq \delta$ and $w_1 + w_2 + \cdots + w_t = \delta$. Then it is possible to decompose **G** into t $n \times n$ circulant matrix with weights $w_1, w_2, \cdots, w_t$, respectively. Let $\mathbf{g}_1$ be the first column of **G**. Split $\mathbf{g}_1$ into t columns of the same length $n, g_1^{(1)}, g_1^{(2)}, \cdots, g_1^{(t)}$, such that the first w_1 1-components of $\mathbf{g}_1$ are put in $\mathbf{g}_1^{(1)}$, the next w_2 1-components of $\mathbf{g}_1$ are put in $\mathbf{g}_1^{(2)}$, $\cdots$, and the last w_t 1-components of $\mathbf{g}_1$ are put in $\mathbf{g}_1^{(t)}$. For each new column $\mathbf{g}_1^{(i)}$, form an $n \times n$ circulant matrix $\mathbf{G}_i$ by cyclically shifting $\mathbf{g}_1^{(i)}$ downward n times. This process results in t $n \times n$ circulant matrices, $\mathbf{G}_1, \mathbf{G}_2, \cdots, \mathbf{G}_t$, with weights $w_1, w_2, \cdots, w_t$, respectively. These circulants are called the descendants of **G**. We readily see that two corresponding columns in two descendants $\mathbf{G}_i$ and $\mathbf{G}_j$ do not have any 1-component in common and $\mathbf{G} = \mathbf{G}_1 + \mathbf{G}_2 + \cdots + \mathbf{G}_t$. The above decomposition, called column decomposition of **G**, results in an $n \times tn$ matrix $\mathbf{H} = [\mathbf{G}_1 \mathbf{G}_2 \cdots \mathbf{G}_t]$. If $w_1 = w_2 = \cdots = w_t = \gamma$, then **H** is a regular matrix with column and row weights γ and $\delta = t\gamma$, respectively. If $t = \delta$ and $w_1 = w_2 = \cdots = w_\delta = 1$, each descendant $\mathbf{G}_i$ of **G** is an $n \times n$ permutation matrix. The parameter t is called the column splitting factor. It is clear that we can decompose **G** by splitting the first row of **G** into multiple rows in the same manner as we split the first column of **G** and cyclically shifting each new row to the right n times.

Let $1 \leq c \leq max\{w_i : 1 \leq i \leq t\}$. For $1 \leq i \leq t$, let $w_{i,1}, w_{i,2}, \cdots, w_{i,c}$ be a set of nonnegative integers such that $0 \leq w_{i,1}, w_{i,2}, \cdots, w_{i,c} \leq wi$ and $w_{i,1} + w_{i,2} + \cdots + w_{i,c} = w_i$. Each descendant $\mathbf{G}_i$ in **H** can be decomposed into c second-generation descendant circulants with weights $w_{i,1}, w_{i,2}, \cdots, w_{i,c}$, respectively, by splitting its first row $\mathbf{q}_{i,1}$ into c rows of length n, denoted $\mathbf{q}_{i,1}^{(1)}, \mathbf{q}_{i,1}^{(2)}, \cdots, \mathbf{q}_{i,1}^{(c)}$, where $\mathbf{q}_{i,1}^{(1)}$ contains the first $w_{i,1}$ 1-components of $\mathbf{q}_{i,1}, \cdots$, and $\mathbf{q}_{i,1}^{(c)}$ contains the last $w_{i,c}$ 1-components of $\mathbf{q}_{i,1}$.

Cyclically shifting each new row $\mathbf{q}_{i,1}^{(k)}$, for $1 \leq k \leq c$, to the right n times, we obtain c circulants $\mathbf{G}_i^{(1)}, \mathbf{G}_i^{(2)}, \cdots, \mathbf{G}_i^{(c)}$, which are descendants of $\mathbf{G}_i$. Decomposition of $\mathbf{G}_i$ results in a $cn \times n$ matrix $\mathbf{Z}_i$ with $\mathbf{G}_i^{(1)}, \mathbf{G}_i^{(2)}, \cdots, \mathbf{G}_i^{(c)}$ as submatrices arranged in a column. We call $\mathbf{Z}_i$ the row-decomposition of $\mathbf{G}_i$ and c is called the row splitting factor. In row splitting, we allow $w_{i,k} = 0$. If $w_{i,k} = 0$, $G_i^{(k)}$ is an $n \times n$ zero matrix regarded as a circulant matrix. If each circulant $\mathbf{G}_i$ in $\mathbf{H}$ is replaced by its decomposition $\mathbf{Z}_i$, we obtain the following $cn \times tn$ matrix:

$$Z = [Z_1 Z_2 \cdots Z_t] = \begin{Bmatrix} G_1^{(1)} & G_2^{(1)} & \cdots & G_t^{(1)} \\ G_1^{(2)} & G_2^{(2)} & \cdots & G_t^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ G_1^{(c)} & G_2^{(c)} & \cdots & G_t^{(c)} \end{Bmatrix}$$

Matrix Z consists of a ct array of nn circulants which are descendants of G. If each G(k) i has weight 1, then Z is an array of permutation matrices.

If all the descendant circulants in Z have the same weight w, then the row and column weights of Z are tw and cw, respectively. In this case, the quasi-cyclic LDPC code generated by Z is a (cw, tw)-regular LDPC code with minimum distance at least cw+1.

Figure 3.2 shows a column decomposition of an 8 circulant matrix of weight 3 into two descendants with weight 2 and 1, respectively.

3.4 IRA codes and eIRA codes

3.4.1 IRA codes

Irregular Repeat-Accumulate(IRA) codes are a generalization of the Repeat-Accumulate(RA) codes [10] [17]. The IRA codes combine many of the favorable attributes of turbo codes and LDPC codes. Like turbo codes, they can be encoded in linear time. Like LDPC

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

Figure 3.2: A column decomposition of a circulant matrix of weight 3.

codes, they are amenable to an exact Richardson-Urbanke style analysis.

Figure 3.3 shows a Tanner graph of an IRA code with parameter (f_1, ..., v_{ra};a), where $f_i \geq 0$, $\sum_i f_i = 1$ and a is a positive integer. The Tanner graph graph is a bipartite graph with two kinds of nodes: variable nodes (open circles) and check nodes (filled circles). There are k variable nodes on the left, called information nodes; there are $r = (k \sum_i i f_i)/a$ check nodes; and there are r variable nodes on the right, called parity nodes. Each information node is connected to a number of check nodes: the fraction of information nodes connected to exactly i check nodes is f_i. Each check node is connected to exactly a information nodes. These connections can made in many ways, as indicated in Figure 3.3 by the "arbitrary permutation" of the ra edges joining information nodes and check nodes. The check nodes are connected to the parity nodes in the simple zigzag pattern shown in the figure. If the "arbitrary permutation" in Figure 3.3 is fixed, the Tanner graph represents a binary linear code with k information bits (u_1,..., u_k) and r parity bits (x_1,... , x_r), as follows. Each of the information bits is

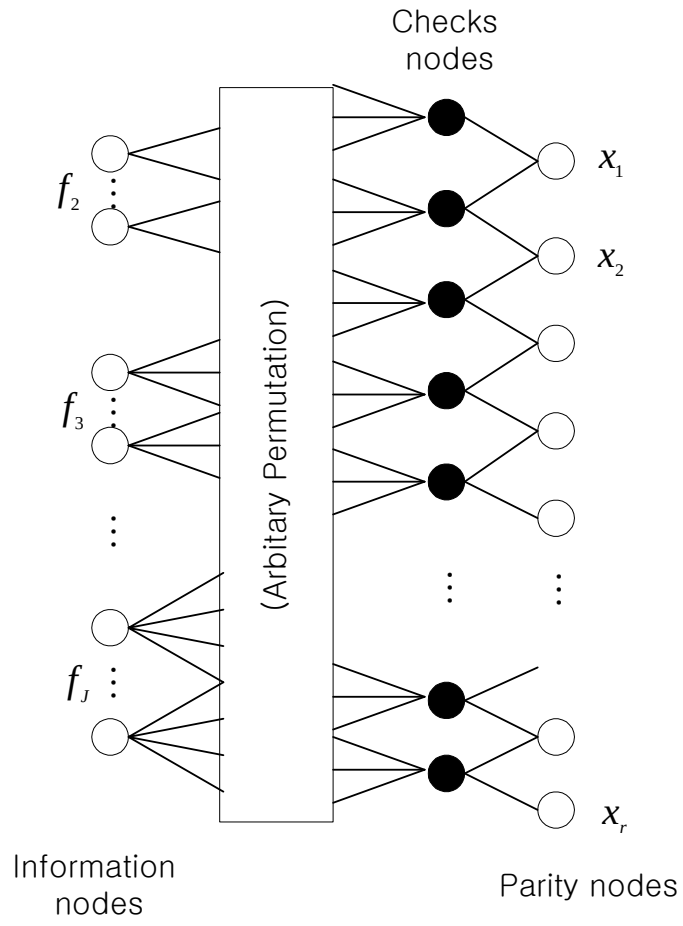

Figure 3.3: Tanner graph for IRA code with parameters ($f_1, \ldots, v_{ra}$;a)

associated with one of the information nodes; and each of the parity bits is associated with one of the parity nodes. The value of a parity bit is determined uniquely by the condition that the mod-2 sum of the values of the variable nodes connected to each of the check nodes is zero. To see this, let us conventionally set $x_0 = 0$. Then if the values of the bits on the ra edges coming out of the permutation box are $(v_1, \ldots, v_{ra})$, we have

the recursive formula

$$x_j = x_{j-1} + \sum_{i=1}^{a} v(j-1)a + i \tag{3.1}$$

for $j = 1, 2, \ldots, r$. This is in effect the encoding algorithm, and so if a is fixed and $n \to \infty$, the encoding complexity is $O(n)$.

There are two versions of the IRA code in Figure 3.3: the nonsystematic and the systematic versions. The nonsystematic version is an (r, k) code, in which the codeword corresponding to the information bit $(u_1, \cdots, u_k)$ is $(x_1, \cdots, x_r)$. The systematic version ia a $(k + r, k)$ code, in which the codeword is $(u_1, \cdots, u_k, ; x_1, \cdots, x_r)$

3.4.2 eIRA codes and Encoder Structure

For a codeword length n and information message length k,

A. Structure of Parity-check matrix

If n is length of codeword, and k is length of information message, then the size of parity-check matrix **H** is $(n - k) \times n$. [11] proposed parity-check matrix as

$$\mathbf{H} = [\mathbf{H}_1 \mid \mathbf{H}_2] \tag{3.2}$$

$\mathbf{H}_1$ is $(n-k) \times k$ matrix and can be constructed irregularly by density evolution according to optimal weight distribution [7]. $\mathbf{H}_2$ is $(n - k) \times (n - k)$ matrix and should be constructed by below [10] [11].

$$\mathbf{H}_2 = \begin{bmatrix} 1 & & & & & & & \\ 1 & 1 & & & & & & \\ & 1 & 1 & & & & & \\ & & 1 & & & & & \\ & & & \ddots & & & & \\ & & & & 1 & 1 & & \\ & & & & & 1 & 1 & \\ & & & & & & 1 & 1 \end{bmatrix}$$

Observe from the form of **H** that encoding may b performed directly from **H** by solving for the parity bits recursively:

1. the first check equation(first row) involves only the first parity bits
2. the second check equation involves only the first two parity bits
3. the third check equation involves only the next two parity bits and so on.

Thus, the form not only avoids degree-two cycles, but it also achieves the ultimate situation in [5] in which a full $m \times m$ submatrix possesses the diagonal form which permits a recursive solution of the parity bits.

B. Structure of Generator Matrix

When **H** matrix is given by eq. 3.2, then corresponding generator matrix is

$$\mathbf{G} = [\mathbf{I} \mid \mathbf{H}_1^T \mathbf{H}_2^{-T}] \tag{3.3}$$

Then $\mathbf{G}\mathbf{H}^T = \mathbf{0}$. $\mathbf{I}_k$ is $k \times k$ identity matrix and $\mathbf{H}_2^{-T}$ matrix is

$$\mathbf{H}_2^{-T} = \begin{bmatrix} 1 & 1 & 1 & \cdots & & 1 & 1 \\ & 1 & 1 & \cdots & & \vdots & 1 \\ & & 1 & \cdots & & \vdots & 1 \\ & & & & & & \\ & & & \ddots & \ddots & \vdots & \vdots \\ & & & & & & \\ & & & & & 1 & 1 \\ & & & & & & 1 \end{bmatrix}$$

$\mathbf{H}_2^{-T}$ matrix corresponds differential encoder transfer function of which is $\frac{1}{1 \oplus D}$. Consequently we can do differential encoding not matrix multiplication to get parity-check bit, so it can reduce the complexity of LDPC encoder. The structure of LDPC encoder corresponding Eq. 3.3 is shown in Figure 3.4. In Figure 3.4 the structure is composed of two part one of which is systematic encoder block that corresponds identity matrix and the other is the block to get parity-check bit that corresponds $H_1^T H_2^{-T}$. In this stucture it is much simpler than original LDPC encoder but there remains the necessity of matrix multiplication operation with H_1^T.

This class of efficiently encodable codes was independently discovered by Narayanaswami

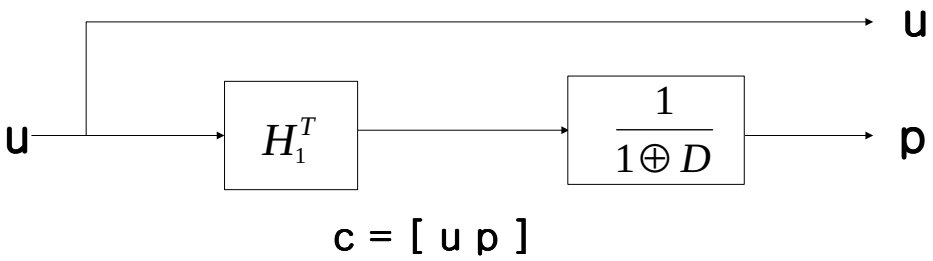

Figure 3.4: Block diagram of encdoer of eIRA code

and Narayanan [18]. These codes resemble the systematic version of the IRA codes [10], except for systematic IRA codes, the matrix $\mathbf{H}_1^T$ in Figure 3.4 which has dimension is replaced by a low-density generator matrix. For this reason, [11] call these codes extended IRA(eIRA) codes.

Chapter 4

Modified Encoder Structure

In this chapter we introduce some structure of encoder that encode codeword efficiently. We modified the structure of parity-check matrix $\mathbf{H}_1$. Instead using random matrix $\mathbf{H}_1$, we construct $\mathbf{H}_1$ as circulant matrix. And we use permutation matrix to grant random property to parity-check matrix. Circulant matrix can be constructed by Shift Register and permutation matrix constructed by equivalent interleaver. Proposed encoder is more efficient than random structure of encoder.

4.1 Modified Generator and Parity check Matrix

4.1.1 Modified structure of Parity check matrix

In this chapter, in order to make more efficient encoder, we propose that parity-check matrix $\mathbf{H}_1$ is constructed by using circulant matrix. This $\mathbf{H}_1$ matrix can be composed of $p \times q$ submatrix which are also circulant matrix.

$$\mathbf{H}_1 = \begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1q} \\ h_{21} & h_{22} & \cdots & h_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ h_{p1} & h_{p2} & \cdots & h_{pq} \end{bmatrix}$$

where $\mathbf{h}_{ij}$ matrix($1 \leq i \leq p$, $1 \leq j \leq q$) is all $l \times l$ size and $k = lq$ and $n - k = lp$. Generally, randomly constructed matrix has good performance. But since proposed matrix is circulant matrix, random property is gone away, so its performance is degraded somewhat. Therefore we need to give random property to $\mathbf{H}_1$ matrix. In order to do so, we use row permutation matrix **P** that is $m \times m$ matrix and each of rows and columns has only one 1's. So modified $\mathbf{H}_1^{'}$ matrix is

$$\mathbf{H}_1^{'} = \mathbf{P}\mathbf{H}_1 \tag{4.1}$$

$\mathbf{H}_1^{'}$ matrix is the same size of $\mathbf{H}_1$ matrix and each row of $\mathbf{H}_1$ matrix is rearranged. For encoding and decoding, we can use $\mathbf{H}_1^{'}$ matrix instead of $\mathbf{H}_1$ matrix, then the modified parity check matrix $\mathbf{H}^{'}$ is as shown below.

$$\mathbf{H}^{'} = [\mathbf{H}_1^{'} | \mathbf{H}_2] \tag{4.2}$$

4.1.2 Modified structure of Generator matrix

For a given parity-check matrix such as eq. 4.2, the corresponding generator matrix is given by

$$\mathbf{G}^{'} = [\mathbf{I} | \mathbf{H}_1^{'T} \mathbf{H}_2^{-T}] \tag{4.3}$$

and since $\mathbf{H}_1$ matrix is circulant matrix, so $\mathbf{H}_1^{'T}$ matrix is also circulant matrix. And $\mathbf{H}_1^{'T}$ is equal to $\mathbf{H}_1^T \mathbf{P}^T$, and $\mathbf{P}^T$ matrix performs that change the position of parity. So we can

substitute interleaver that perform same function as $\mathbf{P}^T$. For example, when 5×5 $\mathbf{P}^T$ matrix is given below

$$P^T = \begin{bmatrix} & & & 1 & \\ & 1 & & & \\ 1 & & & & \\ & & 1 & & \\ & & & & 1 \end{bmatrix},$$

then interleaver that perform same function as $\mathbf{P}^T$ matrix is same as Figure 4.1.

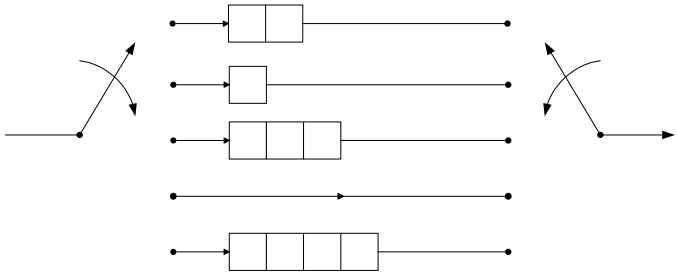

Figure 4.1: Example of inner structure of interleaver

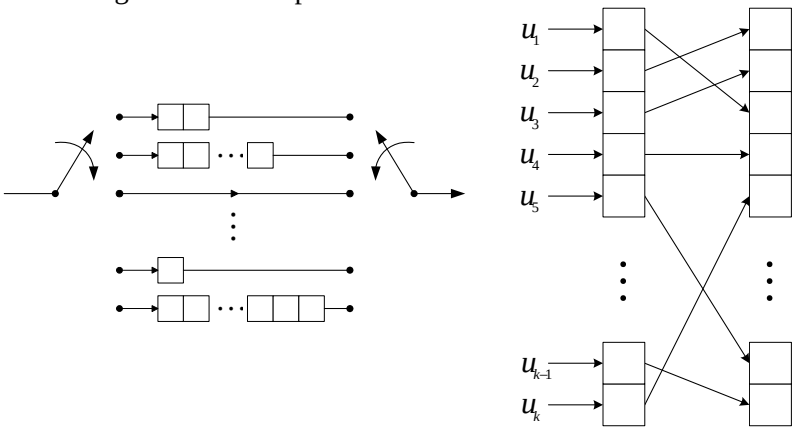

Figure 4.2: Two types of Interleaver structure

Now using modified $\mathbf{H}_1^{'}$ matrix, we can make encoder as shown in Figure 4.3. Figure 4.3 is added to interleaver compared with Figure 3.4. Then the bit encoded through $\mathbf{H}_1^{'T}$ is interleaved, which is same as $\mathbf{uH}_1^{'T}\mathbf{P}$.

An inner structure of encoder to get parity bit is shown in Figure 4.3. For an information message $\mathbf{u}$ whose length is m,

$$\mathbf{u} = [\mathbf{u}_1\mathbf{u}_2\mathbf{u}_3 \dots \mathbf{u}_m] \tag{4.4}$$

and each l messages as same length as register size are inputted to shift register. For example, assume that $\mathbf{H}_1 = [\mathbf{h}_{11}\mathbf{h}_{12}]$ and a size of $\mathbf{h}_{11}$and $\mathbf{h}_{12}$ is all 5, and each generator polynomial of circulant matrix is

$$\begin{aligned} g_1 &= 1 + x^3 \\ g_2 &= x + x^3, \end{aligned} \tag{4.5}$$

then structure of shift register is shown in Figure 4.4

Inputted messages are cyclic shifted l times by shift register. The connection of shift register is determined by generator polynomial of circulant matrix $\mathbf{h}_{11}$ and $\mathbf{h}_{12}, \cdots, \mathbf{h}_{pq}$. At each cyclic shift state, through this connection and XOR operation, a value is updated. Finally these bits are differentially encoded and transmitted to channel.

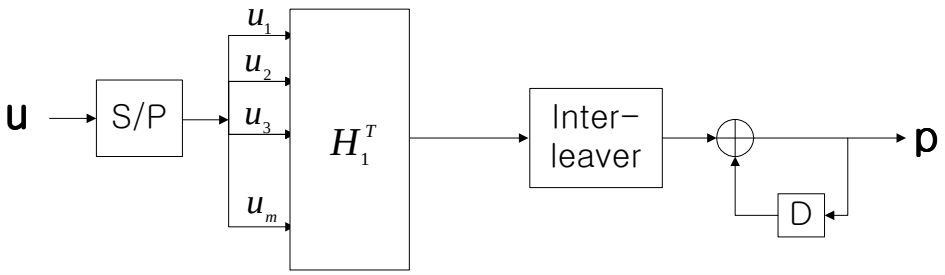

Figure 4.3: Encoder structure when using one circulant matrix

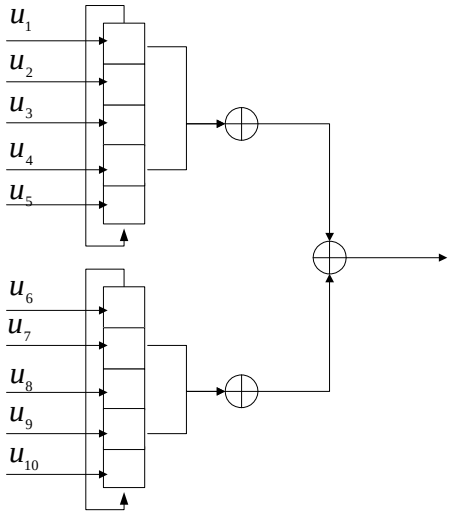

Figure 4.4: Example of Shift Register structure when $g_1 = 1 + x^3$ and $g_2 = x + x^3$

4.2 Encoding complexity

Assume the density of ones in $\mathbf{H}_1$ to be δ and encoding is performed by matrix multiplication and addition operation via $\mathbf{H} = [\mathbf{H}_1|\mathbf{H}_2]$ matrix shown in Figure 4.6. Since H_2 matrix has 1' in its main diagonal and below diagonal, the number of binary additions required to compute the $n - k$ parity bits is approximately

$$N_1 = \delta(k + 1))(n - k)$$

If we use the encoder of Figure 4.4 to encode, multiplication by the matrix H_1^T results in $\delta k(n - k)$ additions, and differential encoding results in $n - k$ additions. Thus total number of binary addition required to compute the $n - k$ parity bits is

$$N_2 = (\delta k + 1))(n - k).$$

N_2 is a small fraction larger than N_1.

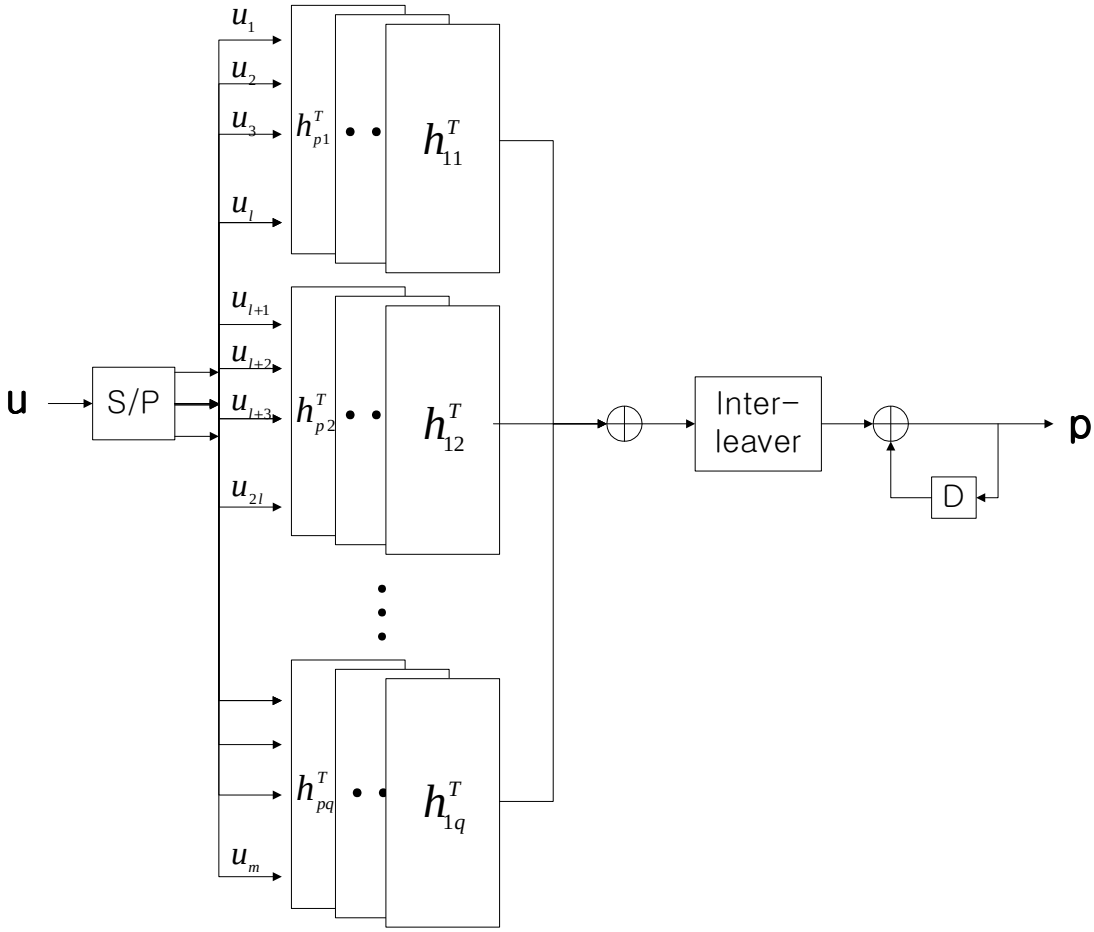

Figure 4.5: Encoder structure when using $p \times q$ circulant matrices

For given H matrix, consider encoding via a generator matrix $G = [I|P]$ which are obtained by Gauss-Jordan elimination. In general, the $k \times (n-k)$ matrix P has a density 0.5, and so the number of binary additions required to do the multiplication mP is approximately $0.5k(n-k)$. Thus, for example, for a density of $\delta = 0.01$, this complexity reduction factor is $\frac{N_2}{N_1} = \frac{0.5k}{\delta(k+1)} \approx \frac{0.5}{\delta} = 50$.

In table 4.1 and 4.2, we compare the complexity of proposed encoding method and

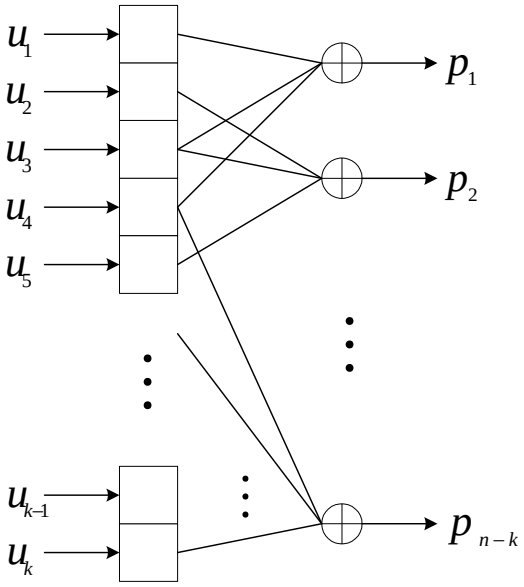

Figure 4.6: Encoding process using matrix multiplication and addition operation

others. Where δ is the density of $\mathbf{H}_1$ matrix and δ' is the density of submatrix of generator matrix **G**. And n is a length of encode bits and k is a length of message bits. In the first row, it shows matrix multiplication and addition operation and its complexity. In the second row, it shows that Yang's encoding method. Next third and fourth row show our proposed encoding method. Compared with second and third row, needed XOR gate numbers are $\delta k(n-k)$ and δkp, respectively. In general n is approximately larger than 1000. Since a code rate is 0.75, k is 250 if $n = 1000$. Thus $n - k = 750$ and $p = 1$ in our simulation. So needed XOR gate is $1/750$ times than before, so this is much smaller than Yang's encoding method.

Table 4.1: Encoding complexity comparison

Encoding Method	δ	Total Computation	Total XOR	Total Memory
Matrix multi.	$\delta' \approx 0.5$	$\delta' k(n-k)$	$\delta' k(n-k)$	$n-k$
Michael Yang	≈ 0.01	$(\delta k+1)(n-k)$	$\delta k(n-k)$	$n-k$
SR only	≈ 0.01	$(\delta k+1)(n-k)$	δkp	$n-k$
SR, Interleaver	≈ 0.01	$(\delta k+1)(n-k)$	δkp	$\leq 2(n-k)$

Table 4.2: Example of encoding complexity comparison

Encoding Method	Computation Comparison	XOR num. comparison (n=1024, k=768, p=1)
Matrix multi.	1	•
Michael Yang	1/50	1
SR only	1/50	1/256
SR, Interleaver	1/50	1/256

4.3 Simulations and Results

In this section, we present the performance results for the proposed codes. We shall show the bit error rate(BER) P_b. In additive white gaussian noise(AWGN) channel, we use sum-product algorithm to decode LDPC code. A codeword length is 512, 1024 and 2924, respectively and code rate is all 0.75. In addition, to compare other case we show a performace in case of n is 1023 and coderate is 0.66. A Parity-check matrix $\mathbf{H}$ used has the form same as below

$$\mathbf{H} = [\mathbf{H}_1 | \mathbf{H}_2] = [\mathbf{h}_{11} | \mathbf{h}_{12} | \mathbf{h}_{13} | \mathbf{H}_2]. \tag{4.6}$$

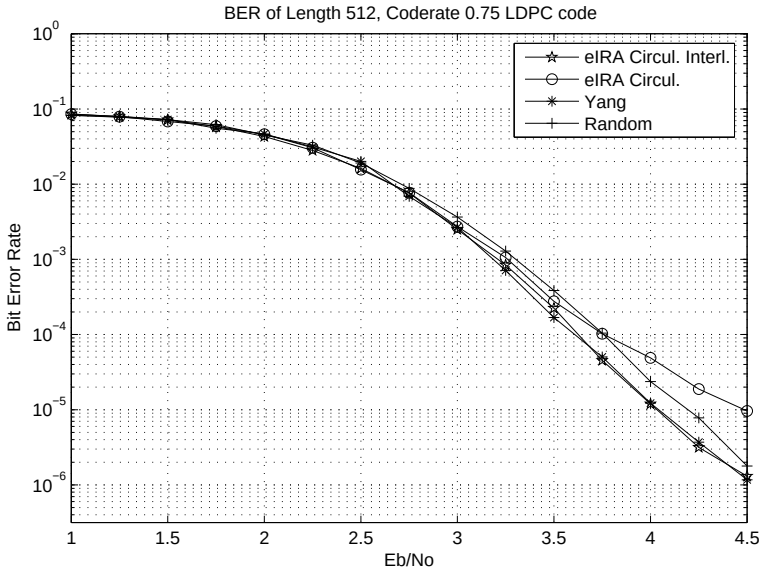

Figure 4.7: BER of LDPC code for length 512, code rate 0.75

We designed $\mathbf{H}_1$ as three concatenated circulant submatrix. For each submatrix, the weights of generator polynomials are different each other. In our simulation, we set weight of $\mathbf{h}_{11}$ to 3, $\mathbf{h}_{12}$ to 4, and $\mathbf{h}_{13}$ to 7. Under this condition and given fixed $\mathbf{H}_2$ matrix, we change $\mathbf{H}_1$ matrix and get the performance results. The Bit Error Rate(BER) performance graph is shown in Figure 4.7, 4.8, 4.9 and 4.11 in case a coderate is fixed as 0.75 and a various codeword length is given such as 512, 1024 and 2924. Figure 4.10 shows the BER performance as different codeword length. And Figure 4.12 and 4.13 show Frame Error Rate(FER) performace in case a codeword length is 512 and 1024, respectively.

In Figure 4.7 first line from the graph, i.e. '$*$' line, means the performance of the parity-check matrix $\mathbf{H}_1$ which is randomly constructed by using MacKay's algorithm

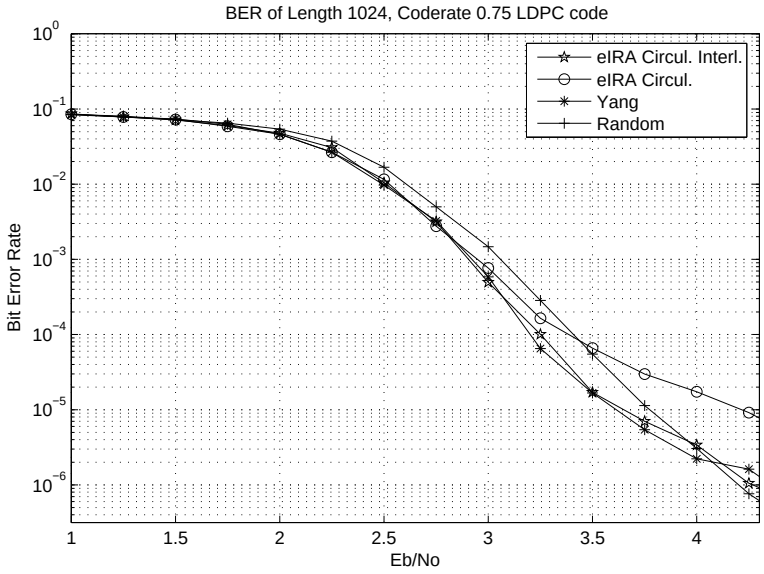

Figure 4.8: BER of LDPC code for length 1024, code rate 0.75

and H_2 using previous fixed form. Second, '$\circ$' line, $\mathbf{H}_1$ is constructed by using circulant matrix and $\mathbf{H}_2$ using previous fixed form. Third, '$+$' line, $\mathbf{H}_1$ and $\mathbf{H}_2$, i.e. $\mathbf{H}$ is randomly and irregularly constructed by using progressive edge growth algorithm. This method resembles Michael Yang's one. At last, '$\star$' line, $\mathbf{H}_1$ is constructed by using circulant matrix and row permutation matrix.

In this result, our proposed code shows better performance than the MacKay's random LDPC codes and similar performance as Yang's method. This code shows better performance up to $P_b = 10^{-6}$ and have nearly error floor effect up to 4.5dB Signal to Noise Ratio(SNR) region. And the parity-check matrix using circulant matrix and permutation matrix is much better than the matrix using only circulant matrix. Thus we can know that permutation matrix gives good performance as expected.

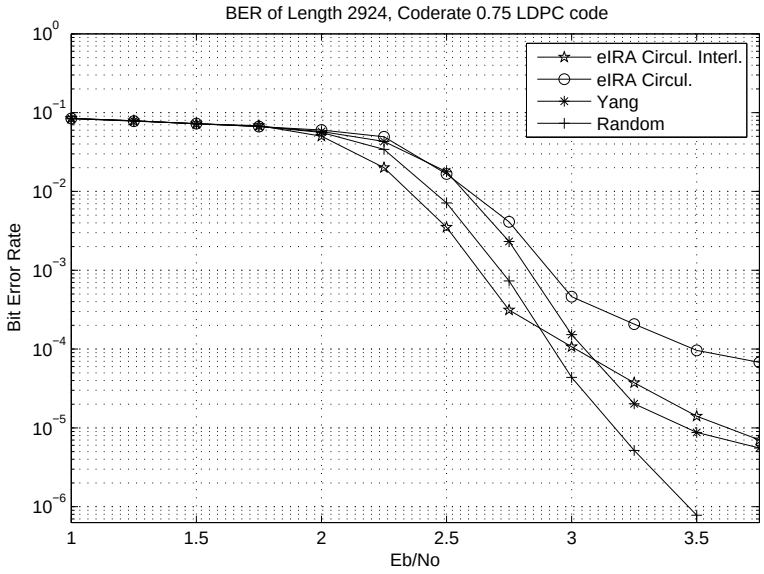

Figure 4.9: BER of LDPC code for length 2924, code rate 0.75

In Figure 4.8, we use the code of which length is 1024 and code rate 0.75. As before its performance show an analogous tendency to Figure 4.7. But in this case we can see error floor effect in case of **H** matrix constructed by circulant matrix, however randomly constructed matrices don't show this effect. Even though our proposed matrix shows error floor effect at about $P_b = 10^{-5}$, it has better performance than random code up to $P_b = 10^{-6}$. But without permutation, it approaches error floor earlier at $P_b = 10^{-4}$. Consequently, we confirm that row permutation gives good performance once again.

In Figure 4.9, like as Figure 4.7 and 4.8, our proposed matrix shows good performance but approaches error floor more earlier. At about $P_b = 10^{-3.5}$ error floor effect appear, this shows bad performance in high SNR region.

We compare the performance according to a codeword length in Figure 4.10. We

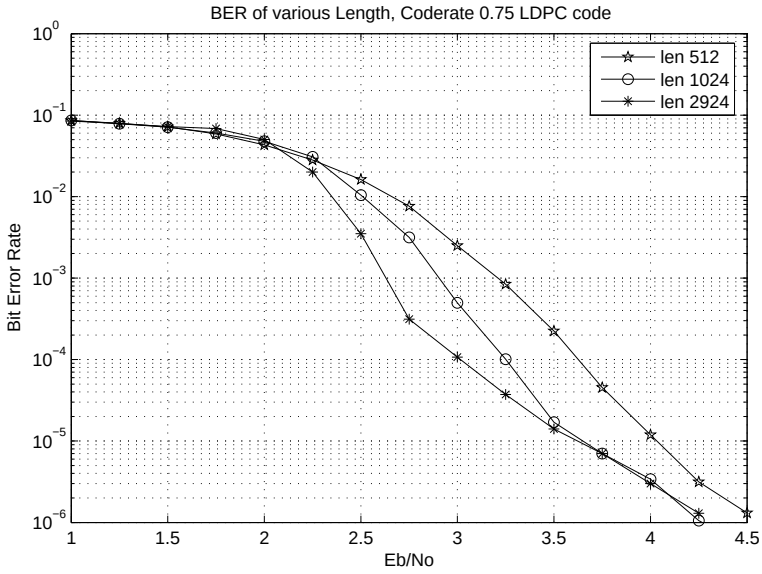

Figure 4.10: BER of LDPC code for various length, code rate 0.75

can see that the larger the codeword length is, The better the performance is. But our proposed code with length 2924 approaches error floor earlier than others, so it became no better than others any more as SNR comes higher.

From previous simulation results, we conclude that our designed code show better performance as the codeword length is shorter. Random code show better performance as its length is larger. Reversely saying, random code are worse as its length is shorter. But our code shows a little degradation even though codeword length is shorter. Thus there is some length our code is better than random code. But it remains conjecture how short length code shows better performance than random code.

Now, we present BER performance graph in case of code rate is 2/3 and length is 1023. Our codes are designed to high code rate, we expect that the performance is worse

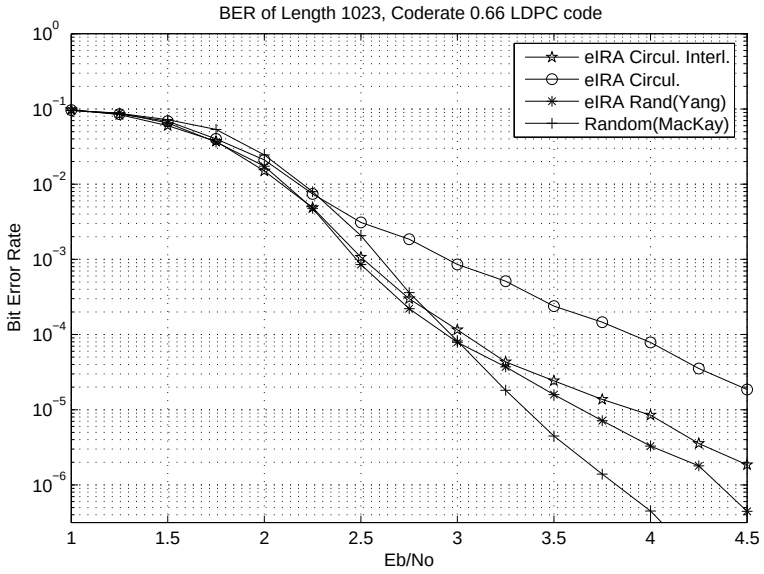

Figure 4.11: BER of LDPC code for length 1023, code rate 0.66

than random code. Actually our code show bad performance as shown in Figure 4.11.

Finally we show frame error rate(FER) performance. In packet system, a important criterion of transmission quality is FER. In a view of FER if there is only one bit error in frame, it is determined as error, so much power is required to reduce FER.

In Figure 4.12 and 4.13 show FER of our code whose length is 512 and 1024, respectively. Our proposed code is better than other codes. But error floor effect is still presented like as BER. In Figure 4.12, our first proposed matrix shows error floor at $P_b = 10^{-2}$. But our proposed code doesn't have error floor in this SNR region. But in Figure 4.13 our two proposed codes, which are constructed using circulant matrix and permutation matrix and using only circulant matrix, show error floor effect at $P_b = 10^{-3.7}$ and $P_b = 10^{-2.8}$, respectively. Thus using permutation matrix gives more

better performance to circulant matrix.

We think that error floor effect is due to no randomness and short cycle of circulant matrix. There are little randomness, so the performance is degraded. But when we applied row permutation thus granting random property, the degradation is somewhat overcome. When we do row permutation with $\mathbf{H}_1$, there may come to being 4-cycle in $\mathbf{H}$ even though $\mathbf{H}$ has no 4-cycle. So we can do row permutation carefully not to make 4-cycle. We can guess that the degraded performance by $\mathbf{H}_2$ can overcome by well constructed $\mathbf{H}_1$. And the performance will be much better if we remove the short cycle, so making large girth.

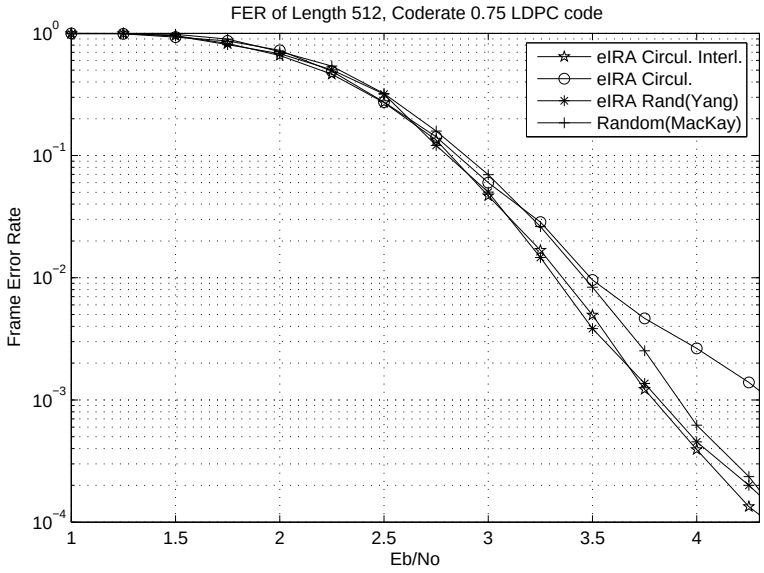

Figure 4.12: FER of LDPC code for length 512, code rate 0.75

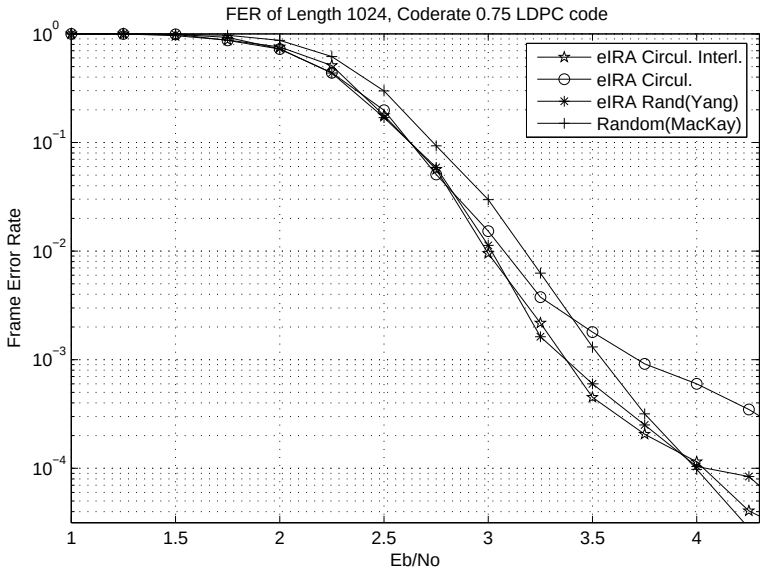

Figure 4.13: FER of LDPC code for length 1024, code rate 0.75

Chapter 5

Concluding Remark

In this dissertation, we concentrate on reducing the complexity of LDPC encoder since general LDPC encoder is much complex than turbo codes. We design structural LDPC code using circulant matrix and permutation matrix and eIRA code for efficient encoder. It is possible to design low complex encoder by using shift register and differential encoder and interleaver than general LDPC encoder that use matrix multiplication and addition operation. The code designed by this structure shows better performance up to 10^{-6} of BER in our simulation results. The proposed code shows a property such that there is less degradation than randomly generated parity-check matrix when the codeword length becomes shorter. However the proposed LDPC code shows error floor effect and a performance degradation in high SNR region. We need short cycle free and well designed matrix to overcome this error floor effects.

For the further research themes, the following problems will be desirable: (1) study on reducing short cycle methods which can improve a performance of LDPC code. (2) study on a permutation rule. Even though there is no 4-cycle, after row permutation there may be exist 4-cycle unless appropriate permutation rule is applied. (3) study on

optimal weight distribution of given codeword length and rate.

Bibliography

[1] C. E. Shannon, “A mathematical theory of communication,” *Bell Syst. Tech. J.*, pp. 379–423(Part 1); pp.623–656(Part 2), Jul. 1948.

[2] R. G. Gallager, “Low-density parity-check codes,” *IRE Trans. Inform. Theory*, vol. IT-B, pp. 21–28, Jan. 1962.

[3] C. Berrou, A. Glavieux, and P. Thitimajshima, “Near shannon limit error correcting coding and decoding:turbo codes,” *in Proc. IEEE Int. Conf. on Communications(ICC 93)*, pp. 1064–1070, May. 1993.

[4] D. J. C. MacKay and R. M. Neal, “Near shannon limit performance of low density parity check codes,” *Electronics Letters*, vol. 32, pp. 1645–1646, Aug. 1996.

[5] T. J. Richardson and R. Urbanke, “Efficient encoding of low-density parity-check codes,” *IEEE Trans. Inform. Theory*, vol. 47, pp. 638–656, Feb. 2001.

[6] H. Tang and S. Lin, “On algebraic construction of gallager and circulant low-density parity-check codes,” *IEEE Trans. Inform. Theory*, vol. 50, pp. 1269–1279, June. 2004.

[7] T. J. Richardson, , A. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 619–637, Feb. 2001.

[8] D. J. C. MacKay and S. T. Wilson, "Comparison of constructions of irregular gallager codes," *IEEE Trans. Commun.*, vol. 47, pp. 1449–1454, Oct. 1999.

[9] Y. Kou, S. Lin, and M. P. C. Fossorier, "Low-density parity-check codes based on finite geometries: A rediscovery and new results," *IEEE Trans. Inform. Theory*, vol. 47, pp. 2711–2736, Feb. 2001.

[10] H. Jin, A. Khandekar, and R. McEliece, "Irregular repeat-accumulate codes," *in Proc. 2nd. Int. Symp. Turbo Codes and Related Topics, Brest, France*, pp. 1–8, Sept. 2000.

[11] M. Yang, W. E. Ryan, and Y. Li, "Design of efficiently encodable moderate-length high-rate irregular ldpc codes," *IEEE Trans. Commun.*, vol. 52, pp. 564–571, April. 2004.

[12] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*. Prentice Hall Inc., 1995.

[13] P. J. Davis, *Circulant Matrices*, 2nd ed. Chelsea Publishing, 1994.

[14] B. Ammar, B. Honary, and S. Lin, "Construction of low-density parity-check codes based on balanced incomplete block designs," *IEEE Trans. Inform. Theory*, vol. 50, pp. 1257–1268, June. 2004.

[15] S. Lin, L. Chen, J. Xu, and I. Djurdjevic, “Near shannon limit quasi-cyclic low-density parity- check codes,” *IEEE Trans. Inform. Theory*, vol. 50, pp. 2030–2035, June. 2003.

[16] S. Lin and J. D. J. Costello, *Error Control Coding:Fundamentals and Applications*, 2nd ed. Prentice Hall, 2004.

[17] D. Divsalar, H. Jin, and R. McEliece, “Coding theorems for ’turbo-like’ codes,” *in Proc. 36th Allerton Conf. on Communication, Control, and Computing*, pp. 201–210, Sept. 1998.

[18] R. Narayanaswami, “Coded modulation with low-density parity-check codes,” M.S. thesis, Dept. Elect. Eng. Texas A&M Univ., Colleage Station, Tx, 2001.

국 문 요 약

순환 행렬과 eIRA 부호를 이용한 효율적인 LDPC 부호화기 설계

신뢰성 있는 데이터 통신이나 저장장치를 위해서는 채널 코딩은 필수적이다. LDPC 부호는 패리티 검사 행렬이 대부분 0인 선형 블록 부호로서 터보 부호처럼 Shannon의 채널 용량의 한계의 거의 근접하는 성능을 보인다. 그러나 터보 부호의 간단한 부호화기와를 달리 LDPC 부호화기는 매우 복잡하다.

본 논문에서는 효율적인 부호화기의 설계를 위하여 패리티 행렬을 순환 행렬과 행치환 행렬, 그리고 eIRA 부호를 이용하여 패리티 검사 행렬을 구조적으로 설계하였다. 순환 행렬은 Shift Register, eIRA 부호는 차등 부호화기, 치환 행렬은 인터리버로 각각 구현이 가능하므로 행렬 곱을 사용하는 일반적인 LDPC 부호화기보다 복잡도가 작은 부호화기의 설계가 가능하였다. 본 논문에서 제시하는 방법은 기존에 제시된 부호화기 보다 계산량은 같지만 XOR 회로의 수를 훨씬 많이 줄였다. 그리고 이 구조로 설계된 부호는 짧은 길이와 높은 부호율을 가질때 랜덤하게 생성된 패리티 검사 행렬보다 좋은 성능을 보인다. 그러나 순환 행렬을 사용함으로 랜덤성의 상실로 인해 높은 SNR에서는 오류 마루 현상이 생겨 성능 열화가 생겼다. 이런 오류 마루 현상을 해결하기 위해서는 짧은 주기가 없는 잘 설계된 행렬로 성능을 향상시킬 수 있다.

핵심되는 말: LDPC, 효율적인 부호화기, 순환 행렬, 행치환 행렬, eIRA 부호