國立交通大學

電子工程學系　電子研究所碩士班

碩 士 論 文

具結構性且低錯誤地板的
CP-PEG 低密度同位元檢查碼之設計

Design of Structured CP-PEG LDPC Codes with Low Error Floor

學生：林義凱

指導教授：張錫嘉 博士

中華民國 九十六年十一月

具結構性且低錯誤地板的CP-PEG 低密度同位元檢查碼之設計

Design of Structured CP-PEG LDPC Codes with Low Error Floor

研究生：林義凱 Student：Yi-Kai Lin

指導教授：張錫嘉 博士 Advisor：Dr. Hsie-Chia Chang

國立交通大學

電子工程學系 電子研究所 碩士班

碩士論文

A Thesis
Submitted to Department of Electronics Engineering & Institute Electronics
College of Electrical and Computer Engineering
National Chiao Tung University
In Partial Fulfillment of the Requirements
for the Degree of Master of Science
in
Electronics Engineering
November 2007
Hsinchu, Taiwan, Republic of China

中華民國九十六年十一月

具結構性且低錯誤地板的
CP-PEG 低密度同位元檢查碼之設計

學生：林義凱　　　　　　指導教授：張錫嘉 博士

國立交通大學

電子工程學系 電子研究所碩士班

摘　要

PEG 演算法已被證明是一種簡單且有效率的方法，可用以設計出好的低密度同位元檢查碼。然而，以 PEG 演算法建構出來的 Tanner 圖並無結構性，所對應的同位檢查矩陣裡 1 的位置完全沒有規則性。在這篇論文裡，我們提出一個以 PEG 演算法為基礎的通用型方法，可用來建構有結構性的 Tanner 圖。如同 PEG 演算法，我們提出的 CP-PEG 演算法可彈性地選擇參數來建構規則和不規則的 Tanner 圖。此種硬體導向的低密度同位元檢查碼可減少超大型積體電路實現的複雜度。為了編碼複雜度及錯誤地板的考量，我們所提出的演算法的變型也會被討論到。就位元錯誤率和封包錯誤率而論，模擬結果顯示我們的低密度同位元檢查碼勝過其他以 PEG 為基礎的低密度同位元檢查碼，而且也優於 IEEE 802.16e 標準裡所採用的低密度同位元檢查碼。

Design of Structured CP-PEG LDPC Codes with Low Error Floor

Student：Yi-Kai Lin Advisor：Dr. Hsie-Chia Chang

Department of Electronics Engineering

Institute of Electronics

National Chiao Tung University

Abstract

Progressive edge-growth (PEG) algorithm was proven to be a simple and effective approach to design good LDPC codes. However, the Tanner graph constructed by PEG algorithm is non-structured, leading to the positions of 1's of the corresponding parity check matrix being fully random. In this thesis, a general method based on PEG algorithm is proposed to construct structured Tanner graphs. These hardware-oriented LDPC codes can not only reduce the VLSI implementation complexity but also provide comparable performance. Similar to PEG method, our CP-PEG approach can construct both regular and irregular Tanner graphs with flexible parameters. For considering the encoding complexity and error floor, modifications of proposed algorithm are discussed. Simulation results show that our CP-PEG approach, in terms of bit error rate (BER) or packet error rate (PER), outperforms other PEG-based and IEEE 802.16e LDPC codes.

誌謝

時光匆匆，轉眼間已到了畢業的時節。在這短短兩年多的碩士生涯裡，承蒙了許多人的照顧與指教，讓我能順順利利的完成碩士論文裡的研究。首先我要感謝 TWT 實驗室的溫瓌岸教授。謝謝您曾給過的指導與鼓勵，讓我學習到了作研究的態度，也培養出一些作研究的能力。另外，TWT 的所有學長姐及同學們，感謝你們給我的討論與關懷，還有我們一同擁有的歡笑時光，這都將成為我碩士生涯中美好的回憶。

再者我要感謝我的指導教授張錫嘉老師。謝謝您在研究上所給予的指導與意見，讓我能順利的完成整個研究工作。也謝謝 Ocean group 裡的所有成員，因為你們的幫助與指教，使我能在研究這條路上走的更加的穩健。特別是建青學長、彥欽學姐和胖達學長，感謝你們在我迷惘時所給予的關懷及協助。此外，我也要感謝阿龍學長在模擬上提供的協助，讓我能及時的看到結果，並做進一步的改進。

最後我謹以此論文獻給我摯愛的父母及家人。謝謝你們的支持與體諒，讓我能無後顧之憂的完成此研究工作。謝謝你們!

Contents

1 Introduction **1**

1.1 Motivation 1

1.2 Thesis Organization 2

2 Low-Density Parity-Check Codes **3**

2.1 LDPC Codes 3

2.2 Encoding of LDPC Codes 4

2.2.1 Conventional Method 5

2.2.2 Dual Diagonal Form 5

2.2.3 Lower Triangular Form 6

2.2.4 Approximate Lower Triangular Form 6

2.3 Decoding of LDPC Codes 8

2.3.1 Message Passing on Graph 8

2.3.2 Sum-Product Algorithm 10

2.3.3 Min-Sum Algorithm 17

2.3.4 Layered Belief Propagation Algorithm 18

3 Tanner Graph Construction **21**

3.1 Performance-Related Code Parameters 21

3.1.1 Degree Distribution Pairs 21

3.1.2 Cycles in Tanner Graph 24

3.1.3 Stopping Sets in Tanner Graph 25

3.2 PEG Tanner Graphs 26

3.3 PEG-QC LDPC Codes 29

4 Design of PEG-Based Structured LDPC Codes 35

4.1 Proposed Structured LDPC Codes 36

4.2 ALT Form for Encoding Consideration 40

4.3 EMD Criterion for Lowering Error Floor 42

4.4 Simulation Results and Comparison 43

5 Conclusion 52

E S A
1896

List of Figures

2.1 Example of the parity check matrix of an LDPC code 4

2.2 Tanner graph corresponds to H given in Fig.2.1 4

2.3 Example of the 6×6 matrix with dual-diagonal form 5

2.4 A matrix with lower triangular form 6

2.5 The parity check matrix in approximate lower triangular form 7

2.6 Message passing on a node with d edges 9

2.7 Message passing on a check node with d edges 11

2.8 Message passing on a bit node with d edges 12

2.9 Graph representation for decoding a (10, 5) LDPC code 15

2.10 Plot of the $\Psi(x)$ function 17

2.11 Rows of $H_{layered}$ are grouped into three layers 19

3.1 A (3, 6)-regular Tanner graph 22

3.2 Parity check matrix H corresponds to the graph given in Fig.3.1 22

3.3 A Tanner graph with length-6 cycles 25

3.4 Parity check matrix H corresponds to the graph given in Fig.3.3 25

3.5 Illustration of a S_3 stopping set 26

3.6 (a) Current graph setting by using PEG-QC algorithm, (b) and its corresponding parity check matrix 32

3.7 Tree spreading from variable node v_8 33

3.8 The Tanner graph constructed by PEG-QC algorithm, and its corresponding parity check matrix. There are 4-cycles in this graph. 34

4.1 (a) The Tanner graph constructed by CP-PEG algorithm, (b) and its corresponding parity check matrix 39

4.2 The parity check matrix in approximate lower triangular form 40
4.3 A simple example of ACE criterion 42
4.4 A simple example of EMD criterion 43
4.5 Performance comparison of the regular codes with rate 8/9 constructed by the proposed and PEG-QC algorithms 45
4.6 Performance comparison of the irregular codes with rate 7/8 constructed by the proposed and PEG-QC algorithms 46
4.7 Performance comparison of the irregular codes with rate 1/2 constructed by the proposed and PEG-QC algorithms 46
4.8 Performance comparison of the irregular codes with rate 3/4 constructed by the proposed and PEG-QC algorithms 47
4.9 Iterative decoding with various maximum number of iteration for (2560, 1280) ALT-CP-PEG LDPC code with rate 1/2 47
4.10 Iterative decoding with various maximum number of iteration for (2560, 1920) ALT-CP-PEG LDPC code with rate 3/4 48
4.11 Iterative decoding with various maximum number of iteration for (2560, 2240) ALT-CP-PEG LDPC code with rate 7/8 48
4.12 Performance of (2560, 1280) ALT-CP-PEG LDPC code with rate 1/2 by using layered decoding algorithm 49
4.13 Performance of (2560, 1920) ALT-CP-PEG LDPC code with rate 3/4 by using layered decoding algorithm 49
4.14 Performance of (2560, 1280) ALT-CP-PEG LDPC code with rate 1/2 by using min-sum and modified min-sum algorithms 50
4.15 Performance of (2560, 1920) ALT-CP-PEG LDPC code with rate 3/4 by using min-sum and modified min-sum algorithms 50
4.16 Performance of (2560, 2240) ALT-CP-PEG LDPC code with rate 7/8 by using min-sum and modified min-sum algorithms 51
4.17 Performance comparison of the irregular codes with rate 1/2 constructed by the proposed algorithm and of IEEE 802.16e 51

List of Tables

3.1 Sets of code parameters which will be used hereafter 31

3.2 Occurrence probabilities of codes with various girth constructed by PEG-QC algorithm 31

4.1 Occurrence probabilities of codes with various girth constructed by the proposed CP-PEG and PEG-QC algorithms 38

Chapter 1

Introduction

1.1 Motivation

Low-density parity-check (LDPC) code, a linear block code defined by a very sparse parity check matrix, was first invented by Gallager in 1960's [1]. Owing to the difficulty of VLSI implementation, it has been ignored for about thirty years excepts for the papers by Zyablov and Pinsker [2], Margulis [3], and Tanner [4]. The rediscovery of LDPC code was done by Spielman et al. [5] and MacKay et al. [6,7]. LDPC codes with long block length show good capacity-approached capability under iterative decoding algorithm [8], so they attract much research interests in recent years. In practical applications, construction of good LDPC codes at short and intermediate block length is of great importance.

Among the existing methods, one of the most successful approaches to construct finite-length LDPC codes is so-called progressive edge-growth (PEG) algorithm proposed by Hu et al. [9, 10]. The code parameters specified in the PEG method are highly flexible and can be chosen for the practical applications. However, the positions of 1's of the resulting parity check matrix constructed by PEG algorithm are fully random. This makes it incur higher complexity for VLSI design.

A structured LDPC code decreases both the encoder and decoder complexity and is suitable for the hardware implementation. The recently proposed communication standards, IEEE 802.16e [11] and IEEE 802.11n [12], all adopt structured LDPC codes as error-correcting codes. Z. Li et al. [13] added a circulant constraint into original PEG algorithm to construct a class of structured LDPC codes, named PEG-QC LDPC codes. It had shown performance comparable to several existing methods at high code rate. However, through simulation, we find that PEG-QC algorithm is much easier to construct a Tanner graph with small girth which degrades the error-correcting performance.

In this thesis, we propose a general method based on PEG algorithm to construct structured Tanner graphs. Simulation results show that the proposed algorithm can suppress the probability to generate a graph with short cycles. Moreover, we present that code performance of our LDPC codes outperforms that of codes based on PEG-QC algorithm. For the consideration of encoding complexity and error floor, the modifications of the proposed algorithm are also discussed.

1.2 Thesis Organization

The remainder of this thesis is organized as follows. Chapter 2 describes the basic concepts of the LDPC codes including characteristics, encoding, and decoding algorithms. Chapter 3 reviews the Progressive edge-growth (PEG) and PEG-QC algorithms and introduces some factors that affect the performance of LDPC codes. The proposed algorithm with its modifications and simulation results will be given in Chapter 4. Finally, we conclude this thesis in Chapter 5.

1896

Chapter 2

Low-Density Parity-Check Codes

A binary low-density parity-check (LDPC) code is a linear block code specified by a sparse parity check matrix with fewer 1's relates to the entries 0's. Non-binary LDPC codes over $GF(q)$ are discussed in [14]. LDPC codes mentioned in this thesis are all binary codes, they will be called LDPC codes for short hereafter. In this chapter, an introduction to LDPC codes will be given, including the code characteristics, encoding, and decoding algorithms.

2.1 LDPC Codes

A parity check matrix H which has N columns and M rows defines a (N, K) LDPC code with codeword length N and contains K information bits. Assuming the matrix is of full rank, the number of information bits is $K = N - M$, and the code rate R equals to $1 - M/N$. The parity check matrix with dimension $M \times N$ can correspond to a bipartite graph with N variable nodes and M check nodes. This was first suggested by Tanner [4], so this bipartite graph is also called Tanner graph. On the one side of the graph is the set of variable (bit) nodes corresponds to the N columns of the matrix H, on the other side of the graph is the set of check nodes which corresponds to the M rows of H. If we label the variable and check nodes from 1 to N and 1 to M, respectively, an edge $e(c_i, v_j)$ which connects the variable node v_j with check node c_i corresponds to the 1 in the entry (i, j) of H. Fig. 2.1 shows the parity check matrix H of an LDPC code and Fig. 2.2 is the corresponding Tanner graph relates to H specified in Fig.2.1.

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

Figure 2.1: Example of the parity check matrix of an LDPC code

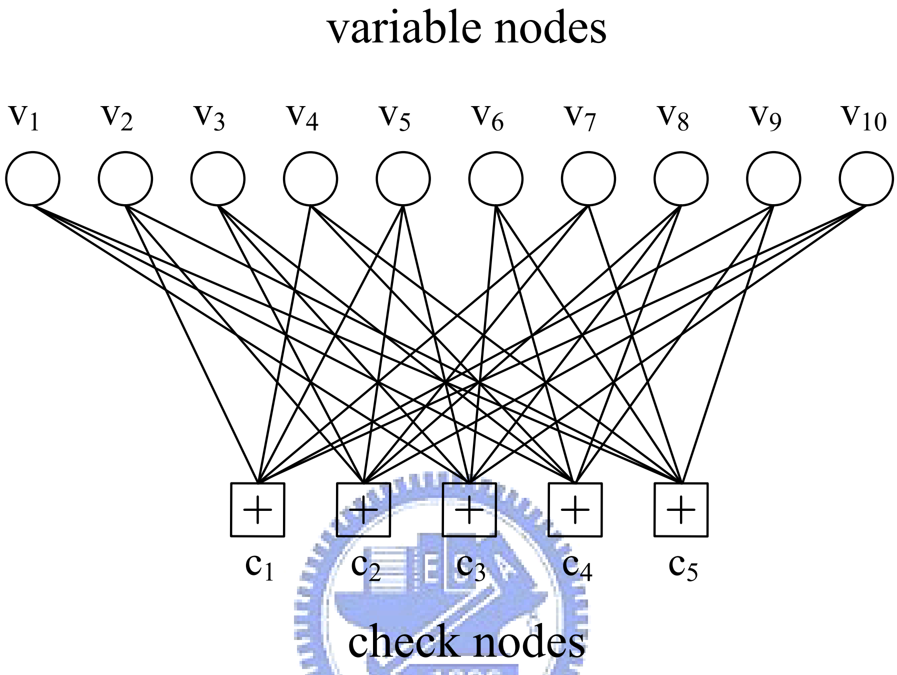


Figure 2.2: Tanner graph corresponds to H given in Fig.2.1

2.2 Encoding of LDPC Codes

Since an LDPC code is a linear block code, we can encode the message u through its generator matrix G. The parity check matrix of an LDPC code is sparse, however, the generator matrix of it often contains many 1's. The above encoding complexity is proportional to N^2, where N is the block length. Previous works suggested us to force the parity check matrix into some special forms and directly encode the message through H. Here we introduce conventional method and several well-known ones.

2.2.1 Conventional Method

Assuming the matrix H is of full rank, for systematic encoding, we can use Gauss-Jordan elimination to put H into systematic form $H_{sys} = [P|I_M]$, where P is an $M \times K$ matrix and I_M is the $M \times M$ identity matrix. The systematic generator matrix is then $G_{sys} = \left[I_K|P^T\right]$ and the encoding can be accomplished as codeword $X = u \cdot G$.

2.2.2 Dual Diagonal Form

For a linear block code, $G \cdot H^T = 0$. A legal codeword X of a linear block code satisfies

$$\begin{aligned} XH^T &= uGH^T \\ &= 0. \end{aligned} \tag{2.1}$$

If the parity check matrix can be divided into two parts $H = \left[H^d|H^p\right]$, where H^d is an $M \times K$ matrix and H^p is an $M \times M$ square matrix with dual-diagonal form. Fig. 2.3 shows a 6×6 matrix with the dual-diagonal form. Corresponding to the parity check matrix H, we partitions the codeword vector X into two parts $X = \left[X^d|X^p\right]$, where X^d is the information part of the codeword vector and X^p is the parity part of it. From $X \cdot H^T = 0$, we get

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

Figure 2.3: Example of the 6×6 matrix with dual-diagonal form

$$H^d(X^d)^T = H^p(X^p)^T. \tag{2.2}$$

For an given $H = \left[H^d|H^p\right]$ and a deterministic information vector X^d, we can derive the parity part of the codeword vector by a projection vector defined as

$$\begin{aligned} v &= H^d(X^d)^T \\ &= H^p(X^p)^T. \end{aligned} \tag{2.3}$$

We note that $(H^p)^{-1} = U^p$, where U^p is a upper triangular matrix, and thus

$$\begin{aligned} X^p &= (H^p)^{-1} v \\ &= U^p v. \end{aligned} \tag{2.4}$$

We can derive the X^p by a back-substitution procedure [15].

2.2.3 Lower Triangular Form

If the parity check matrix satisfies $H = \left[H^d | H^p\right]$, where H^d is an $M \times K$ matrix and H^p is an $M \times M$ square matrix with lower triangular form as shown in Fig. 2.4. We can get

$$H^p = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ h^p_{2,1} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ h^p_{M,1} & h^p_{M,2} & \cdots & 1 \end{bmatrix}$$

Figure 2.4: A matrix with lower triangular form

the parity part of the codeword vector through the following equation.

$$x_i^p = \left(\sum_{j=1}^{i-1} h^p_{i,j} x^p_j + \sum_{j=1}^{N-M} h^d_{i,j} x^d_j\right) \pmod 2 \tag{2.5}$$

2.2.4 Approximate Lower Triangular Form

T. Richardson et al. [16] brought a parity check matrix into approximate lower triangular form indicated in Fig. 2.5 by performing row and column permutations only. Since this transformation involved with permutations only, the matrix is still sparse. More precisely, assume that the parity check matrix H is full rank and we transform the matrix in the form

$$H = \begin{pmatrix} A & B & T \\ C & D & E \end{pmatrix} \tag{2.6}$$

where 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, finally, E is $g \times (M-g)$. Further, these matrices are all sparse and T is

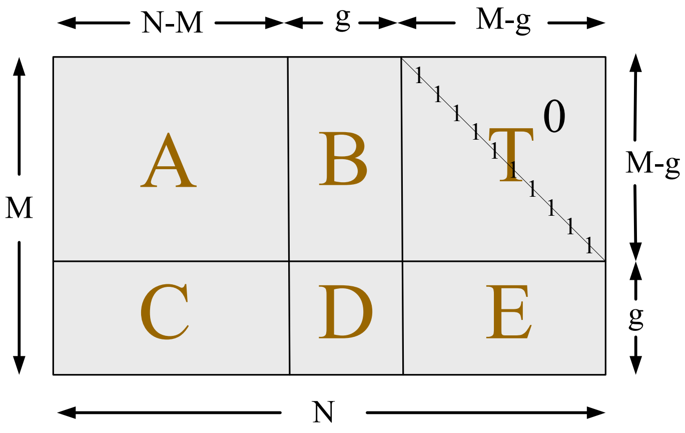


Figure 2.5: The parity check matrix in approximate lower triangular form

lower triangular matrix with ones along the diagonal. Multiplying

$$\begin{pmatrix} I & 0 \\ -ET^{-1} & I \end{pmatrix} \tag{2.7}$$

on the left side of equation (2.6) and we get

$$\begin{pmatrix} A & B & T \\ -ET^{-1}A + C & -ET^{-1}B + D & 0 \end{pmatrix}. \tag{2.8}$$

Let the codeword $X = (u, p_1, p_2)$ where u denotes the systematic part, p_1 and p_2 combined denote the parity part, p_1 has length g, and p_2 has length $(M - g)$. The check equation $HX^T = 0^T$ can split naturally into two equations, namely

$$Au^T + Bp_1^T + Tp_2^T = 0 \tag{2.9}$$

and

$$(-ET^{-1}A + C)u^T + (-ET^{-1}B + D)p_1^T = 0. \tag{2.10}$$

Define $\Phi \equiv -ET^{-1}B + D$ and assume that Φ is nonsigular. Then from equation (2.10) we derive that

$$p_1^T = -\Phi^{-1}(-ET^{-1}A + C)u^T. \tag{2.11}$$

Hence, once the $g \times (N - M)$ matrix $-\Phi^{-1}(-ET^{-1}A + C)$ has been precomputed, the determination of p_1 can be accomplished in complexity $O(g \times (N - M))$ simply by performing a multiplication with this (generically dense) matrix. In a similar manner, we can calculate p_2 from equation (2.9) and derive that

$$p_2^T = -T^{-1}(Au^T + Bp_1^T). \tag{2.12}$$

Following the steps listed in table I and II of [16], we can derive p_1 and p_2 step by step, where the complexity of p_1 and p_2 are $O(N+g^2)$ and $O(N)$, respectively. In fact, $\Phi \equiv -ET^{-1}B+D$ may be a singular matrix. If the resulting Φ is seen to be singular after clearing the matrix E, we can simply perform further column permutations to remove this singularity. This is always possible when H is of full rank, as assumed.

2.3 Decoding of LDPC Codes

In this section, the message passing algorithm which is used to perform probabilistic decoding will be introduced. Then we apply it to decode an LDPC code and derive the well-known iterative decoding algorithm. To reduce the complexity of decoding algorithm, min-sum approach will be discussed further. For the consideration of decoding convergence rate, we study the layered belief propagation decoding algorithm to speed up the convergence rate.

2.3.1 Message Passing on Graph

The soft iterative decoding algorithm relies on message passing or so-called belief propagation (BP) algorithm [17, 18]. Considering the following conditional probability

$$P(x=a|C) \tag{2.13}$$

which denotes the **a posteriori probability** based on the knowledge of constraint C. According to Bayes' theorem, we can rewrite (2.13) as

$$P(x=a|C)=\frac{P(C|x=a)P(x=a)}{P(C)}. \tag{2.14}$$

The term $P(x=a)$ is the priori probability which refers to the probability that variable x chooses the value a. The priori probability is also called **intrinsic probability**, denoted by $P_{int}(x=a)$. The term $P(C|x=a)$ is proportional to the extrinsic probability which is used to provide a new information for x according to the constraint C. The extrinsic probability is defined by

$$P_{ext}(x=a)=\rho_e P(C|x=a), \tag{2.15}$$

where the normalization factor

$$\rho_e = \frac{1}{\sum_{a' \in \mathbf{A}} P(C|x=a')} \tag{2.16}$$

is necessary so that $\sum_{a' \in \mathbf{A}} P_{ext}(x=a^{'}) = 1$, assuming a take values from the alphabet set $\mathbf{A}$. Then, the a posteriori probability in (2.14) can be expressed as

$$P_{post}(x=a) = P(x=a|C) = \rho_p P_{ext}(x=a) P_{int}(x=a), \tag{2.17}$$

where $\rho_p = (\rho_e P(C))^{-1}$. If $\mathbf{A} = GF(2)$, $\mathbf{A}$ contains only two values 0 and 1, the log-likelihood ratio for (2.17) will become

$$L_{post}(x) = ln\frac{P_{post}(x=0)}{P_{post}(x=1)} = ln\frac{P_{ext}(x=0)}{P_{ext}(x=1)} + ln\frac{P_{int}(x=0)}{P_{int}(x=1)} = L_{ext}(x) + L_{int}(x). \tag{2.18}$$

Fig. 2.6 illustrates a graph consisting of one node with d edges. There are d variables, x_1, x_2,…, and x_d, which correspond to the constraint C. A set $\mathbf{S_C}$ is a subspace of the d-dimensional vector space $\mathbf{A^d}$, and for any d-tuple $x = (x_1, x_2, \cdots, x_d) \in \mathbf{S_C}$ will satisfy the constraint C. Assuming that each edge has the intrinsic probability $P_{int}(x_j)$ associated with the variable x_j for $j = 1 \sim d$, the a posteriori probability of each variable x_i with respect to C can be derived through the combination of the intrinsic probability $P_{int}(x_i)$ and the extrinsic probability $P_{ext}(x_i)$. Therefore, we need to evaluate $P_{ext}(x_i)$ based on the constraint C and the other intrinsic probabilities $P_{int}(x_i)$, $j \neq i$. The extrinsic probability

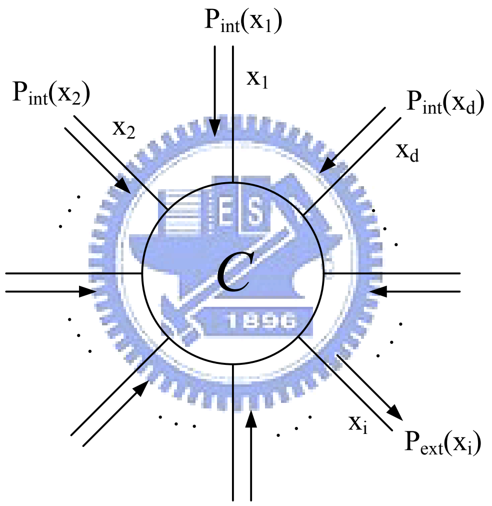


Figure 2.6: Message passing on a node with d edges

is

$$
\begin{aligned}
P_{ext}(x_i) &= \rho_c P(C|x_i) \\
&= \rho_c \sum_{x_j, \forall j \neq i, \mathbf{x} \in \mathbf{S_C}} P(x_1, \cdots, x_{i-1}, x_{i+1}, \cdots, x_d) \\
&= \rho_c \sum_{x_j, \forall j \neq i, \mathbf{x} \in \mathbf{S_C}} \prod_{j=1, j \neq i}^{d} P_{int}(x_j),
\end{aligned} \tag{2.19}
$$

where we assume the symbol variables x_1, x_2,..., and x_d are independent, and ρ_c is a normalization factor.

2.3.2 Sum-Product Algorithm

Since for a codeword X of an LDPC code, it follows that $HX^T = 0^T$ which can be regarded as a set of constraints. Take the parity check matrix in (2.20) for example, a codeword $X = (x_0, x_1, \cdots, x_9)$ of this LDPC code satisfies the constraints in (2.21).

$$
H = \begin{bmatrix}
1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1
\end{bmatrix} \tag{2.20}
$$

$$
\begin{cases}
c_0 : x_0 + x_1 + x_4 + x_6 = 0 \\
c_1 : x_2 + x_6 + x_7 + x_8 = 0 \\
c_2 : x_0 + x_3 + x_5 + x_8 = 0 \\
c_3 : x_1 + x_2 + x_3 + x_9 = 0 \\
c_4 : x_4 + x_5 + x_7 + x_9 = 0
\end{cases} \tag{2.21}
$$

The decoding of an LDPC code is based on sum-product algorithm or message passing algorithm, which exchanges the check-to-bit messages and bit-to-check messages iteratively. After finishing the current iteration, the a posteriori probabilities of the bit nodes will be updated. This above process can be conveniently viewed on a bipartite graph. We first consider the message passing for the check nodes. A check node receives bit-to-check messages from bit nodes connecting to itself and updates the check-to-bit messages. After updating the check-to-bit messages, it transmits those messages back to the bit nodes involved with it. Fig. 2.7 shows a check node containing d edges with each edge corresponding to a variable in $GF(2)$. The constraint set for the node is

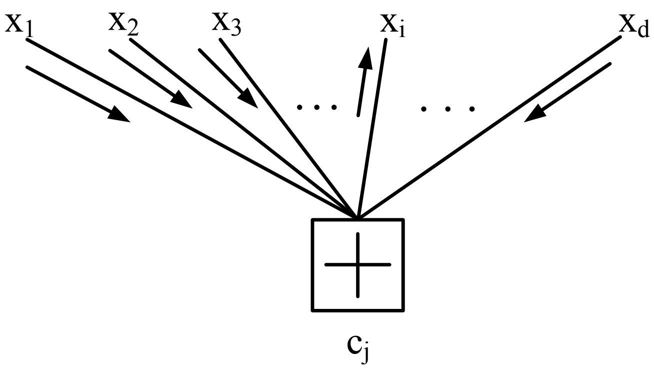


Figure 2.7: Message passing on a check node with d edges

$$\mathbf{S}_{c_j} = \{(x_1, x_2, \cdots, x_d) | x_1 + x_2 + \cdots + x_d = 0\}. \tag{2.22}$$

So the output message on the edge x_i should be

$$\mu_{c_j \to x_i}(x_i) = P_{ext}(x_i) = P(x_1 + \cdots + x_{i-1} + x_{i+1} + \cdots + x_d = x_i). \tag{2.23}$$

Before deriving (2.23), we simplify the question to the two variables condition:

$$\begin{aligned} P(x_1 + x_2 = 0) &= P_{int}(x_1 = 0)P_{int}(x_2 = 0) + P_{int}(x_1 = 1)P_{int}(x_2 = 1) \\ &= (1 - p_1)(1 - p_2) + p_1 p_2, \end{aligned} \tag{2.24}$$

where $p_i = P_{int}(x_i = 1)$. Moreover, the above equation can be expressed as

$$2P(x_1 + x_2 = 0) - 1 = (1 - 2p_1)(1 - 2p_2). \tag{2.25}$$

If we assume

$$\begin{aligned} 2P(x_1 + x_2 + \cdots + x_j = 0) - 1 &= 2\Pi_j - 1 \\ &= (1 - 2p_1)(1 - 2p_2) \cdots (1 - 2p_j) \\ &= \prod_{l=1}^{j} (1 - 2p_l), \end{aligned} \tag{2.26}$$

the following probability will become

$$\begin{aligned} \Pi_{j+1} &= P(x_1 + x_2 + \cdots + x_{j+1} = 0) \\ &= P(x_1 + x_2 + \cdots + x_j = 0)(1 - p_{j+1}) + P(x_1 + x_2 + \cdots + x_j = 1)p_{j+1} \\ &= \Pi_j(1 - p_{j+1}) + (1 - \Pi_j)p_{j+1}. \end{aligned} \tag{2.27}$$

As a result, we can obtain

$$\begin{aligned} 2\Pi_{j+1} - 1 &= (2\Pi_j - 1)(1 - 2p_{j+1}) \\ &= \prod_{l=1}^{j+1}(1 - 2p_l) \end{aligned} \tag{2.28}$$

from (2.26). By induction, we derive that

$$\begin{aligned} \Pi_k &= P(x_1 + x_2 + \cdots + x_d = 0) \\ &= \frac{1}{2}\left[1 + \prod_{i=1}^{d}(1 - 2p_i)\right] \end{aligned} \tag{2.29}$$

for any $d \geq 1$. Then the probability in (2.23) can be written as

$$\mu_{c_j \to x_i}(x_i = 0) = \frac{1}{2}\left[1 + \prod_{l=1, l \neq i}^{d} (1 - 2\mu_{x_i \to c_j}(x_l = 1))\right] \tag{2.30}$$

$$\mu_{c_j \to x_i}(x_i = 1) = \frac{1}{2}\left[1 - \prod_{l=1, l \neq i}^{d} (1 - 2\mu_{x_i \to c_j}(x_l = 1))\right], \tag{2.31}$$

where $p_l = \mu_{x_i \to c_j}(x_l = 1)$ is the message from x_l. For the message passing at the bit node as shown in Fig. 2.8, the node x_i will receive messages from check nodes connecting to itself and from communication channel (the received symbol r_i). Since the constraint set

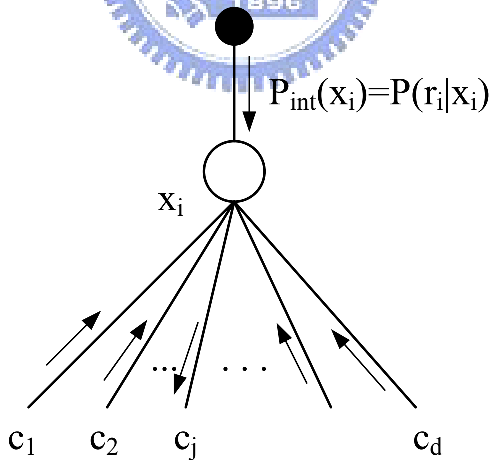


Figure 2.8: Message passing on a bit node with d edges

for x_i is

$$\mathbf{S}_{x_i} = \{x_i = a \mid a \in GF(2)\}, \tag{2.32}$$

the output message from x_i to c_j will be

$$\mu_{x_i \to c_j}(x_i = 0) = \rho_b \cdot P_{int}(x_i = 0) \prod_{l=1, l \neq j}^{d} \mu_{c_l \to x_i}(x_i = 0) \tag{2.33}$$

$$\mu_{x_i \to c_j}(x_i = 1) = \rho_b \cdot P_{int}(x_i = 1) \prod_{l=1, l \neq j}^{d} \mu_{c_l \to x_i}(x_i = 1), \tag{2.34}$$

where

$$\rho_b = \sum_{x_i} P_{int}(x_i) \prod_{l=1, l \neq j}^{d} \mu_{c_l \to x_i}(x_i). \tag{2.35}$$

The intrinsic probability $P_{int}(x_i) = P(r_i \mid x_i)$, and the r_i is received symbol comes from communication channel. For the simplicity, we can transform the messages from probability domain to logarithmic domain by using log-likelihood ratio. The ratio is defined to be

$$L(x) = \ln \frac{P(x = 0)}{P(x = 1)}, \tag{2.36}$$

and

$$P(x = 1) = \frac{1}{e^{L(x)} + 1}. \tag{2.37}$$

Alternatively, we can write

$$1 - 2P(x = 1) = \frac{e^{L(x)} - 1}{e^{L(x)} + 1} = \tanh(\frac{L(x)}{2}), \tag{2.38}$$

where the hyperbolic tangent is defined as

$$\tanh(\frac{x}{2}) = \frac{e^x - 1}{e^x + 1}. \tag{2.39}$$

According to the definition of (2.36), the messages from check node c_j to bit node x_i can be

$$\begin{aligned} L_{c_j \to x_i}(x_i) &= \ln \frac{1 + \prod_{l=1, l \neq i}^{d} (1 - 2\mu_{x_l \to c_j}(x_l = 1))}{1 - \prod_{l=1, l \neq i}^{d} (1 - 2\mu_{x_l \to c_j}(x_l = 1))} \\ &= \ln \frac{1 + \prod_{l=1, l \neq i}^{d} \tanh(\frac{L_{x_l \to c_j}(x_l)}{2})}{1 - \prod_{l=1, l \neq i}^{d} \tanh(\frac{L_{x_l \to c_j}(x_l)}{2})} \\ &= 2 \tanh^{-1} \left(\prod_{l=1, l \neq i}^{d} \tanh(\frac{L_{x_l \to c_j}(x_l)}{2}) \right). \end{aligned} \tag{2.40}$$

We further define a function $\Psi(x)$ as follows for $x > 0$,

$$\Psi(x) = \Psi^{-1}(x) = \ln \frac{1 + e^{-x}}{1 - e^{-x}} = -\ln(\tanh(\frac{x}{2})) \tag{2.41}$$

and note that the inverse hyperbolic tangent is

$$\tanh^{-1}(y) = \frac{1}{2}\ln\frac{1+y}{1-y}. \tag{2.42}$$

We decompose the term in the parentheses of (2.40) and use the property that the sign of $A_l = \tanh(\frac{L_{x_l\to c_j}(x_l)}{2})$ is consistent with $L_{x_l\to c_j}(x_l)$. The result is as (2.43) shows.

$$\begin{aligned}
\prod_{l=1,l\neq i}^{d}\tanh(\frac{L_{x_l\to c_j}(x_l)}{2}) &= \prod_{l=1,l\neq i}^{d} A_l \\
&= \left(\prod_{l=1,l\neq i}^{d}\operatorname{sgn}(A_l)\right)\exp\left(\sum_{l=1,l\neq i}^{d}\ln|A_l|\right) \\
&= \left(\prod_{l=1,l\neq i}^{d}\operatorname{sgn}(L_{x_l\to c_j}(x_l))\right)\exp\left(\sum_{l=1,l\neq i}^{d}\ln(\tanh(\frac{\left|L_{x_l\to c_j}(x_l)\right|}{2}))\right)
\end{aligned} \tag{2.43}$$

Moreover, it is true that for any integer s

$$(-1)^s\Psi^{-1}(x) = \ln\frac{1+(-1)^s e^{-x}}{1-(-1)^s e^{-x}}. \tag{2.44}$$

If we let

$$x = -\left(\sum_{l=1,l\neq i}^{d}\ln\left(\tanh(\frac{\left|L_{x_l\to c_j}(x_l)\right|}{2})\right)\right) \tag{2.45}$$

in (2.44), (2.40)can be then rewritten as

$$\begin{aligned}
L_{c_j\to x_i}(x_i) &= \left(\prod_{l=1,l\neq i}^{d}\operatorname{sgn}(L_{x_l\to c_j}(x_l))\right)\Psi^{-1}\left(-\sum_{l=1,l\neq i}^{d}\ln\left(\tanh(\frac{\left|L_{x_l\to c_j}(x_l)\right|}{2})\right)\right) \\
&= \left(\prod_{l=1,l\neq i}^{d}\operatorname{sgn}(L_{x_l\to c_j}(x_l))\right)\Psi^{-1}\left(\sum_{l=1,l\neq i}^{d}\Psi(\left|L_{x_l\to c_j}(x_l)\right|)\right),
\end{aligned} \tag{2.46}$$

where the function $\Psi(x)$ is previously defined in (2.41). As compared with (2.40), the multiplications has been converted to the additions in (2.46). The message from bit node x_i to check node c_j can also be expressed as

$$\begin{aligned}
L_{x_i\to c_j}(x_i) &= \ln\frac{\rho_b\cdot P_{int}(x_i=0)\prod_{l=1,l\neq j}^{d}\mu_{c_j\to x_i}(x_i=0)}{\rho_b\cdot P_{int}(x_i=1)\prod_{l=1,l\neq j}^{d}\mu_{c_j\to x_i}(x_i=1)} \\
&= L_{int}(x_i) + \sum_{l=1,l\neq j}^{d} L_{c_l\to x_i}(x_i).
\end{aligned} \tag{2.47}$$

In the AWGN channel with variance σ^2 and zero mean, we derive

$$P\left(r_i \mid x_i = +1\right) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(r_i-1)^2}{2\sigma^2}} \tag{2.48}$$

and

$$P\left(r_i \mid x_i = -1\right) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(r_i+1)^2}{2\sigma^2}}. \tag{2.49}$$

Therefore, the value $L_{int}(x_i)$ which is also termed channel value can be obtained by

$$\begin{aligned} L_{int}(x_i) &= \ln \frac{P\left(r_i \mid x_i = +1\right)}{P\left(r_i \mid x_i = -1\right)} \\ &= \ln e^{-\frac{1}{2\sigma^2}\left[(r_i-1)^2-(r_i+1)^2\right]} \\ &= \frac{2}{\sigma^2} r_i, \end{aligned} \tag{2.50}$$

assuming 1 is mapped to (-1), and 0 to (+1). Fig. 2.9 shows a bipartite graph as example for decoding a (10, 5) LDPC code, and the number of check nodes is five. We summarize the overall iterative decoding flow as follows:

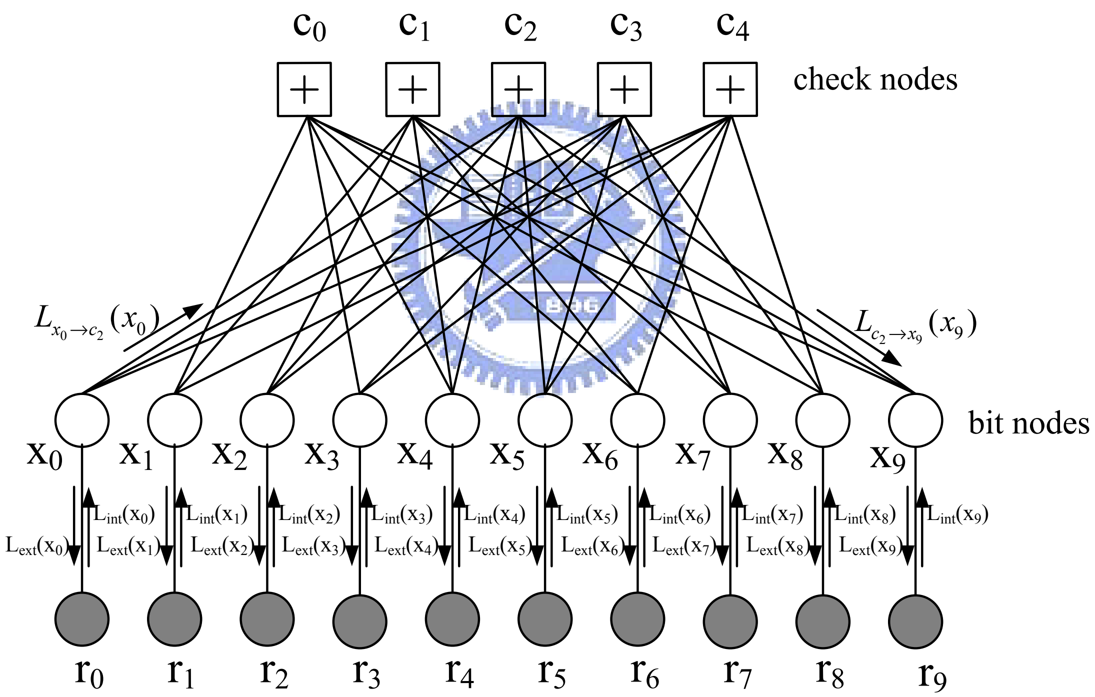


Figure 2.9: Graph representation for decoding a (10, 5) LDPC code

Step 1. (Initialization): Set the current iteration number $n_{ite} = 1$. For each edge existing in the corresponding Tanner graph, the message sent from bit node x_i to check node c_j is initialized to $L_{int}(x_i)$,

Step 2. (Horizontal step): For each check node, we calculate the messages conveyed to the bit nodes with

$$L_{c_j \to x_i}(x_i) = \left(\prod_{l \in N(j) \backslash \{i\}} \operatorname{sgn}(L_{x_l \to c_j}(x_l)) \right) \Psi^{-1} \left(\sum_{l \in N(j) \backslash \{i\}} \Psi(\left| L_{x_l \to c_j}(x_l) \right|) \right). \quad (2.51)$$

The set $N(j)$ comprises all the indexes of the bit nodes that involve the check node c_j. (2.51) can be accomplished through a check-node processing element (CNPE).

Step 3. (Vertical step): For each bit node x_i, the message from x_i to check node c_j which is connecting to this bit node can be updated by

$$L_{x_i \to c_j}(x_i) = L_{int}(x_i) + \sum_{l \in M(i) \backslash \{j\}} L_{c_l \to x_i}(x_i), \quad (2.52)$$

where the set $M(i)$ denotes the indexes of all the check nodes connecting to x_i. Moreover, the a posterior information for codeword symbol x_i is obtained by

$$\begin{aligned} L_{post}(x_i) &= L_{int}(x_i) + L_{ext}(x_i) \\ &= L_{int}(x_i) + \sum_{l \in M(i)} L_{c_l \to x_i}(x_i). \quad (2.53) \end{aligned}$$

A bit-node processing element (BNPE) can carry out the functions depicted in (2.52) and (2.53).

Step 4. (Hard decision and syndrome check): We estimate the codeword $\tilde{X} = (\tilde{x}_0, \tilde{x}_1, \cdots, \tilde{x}_{N-1})$ by

$$\tilde{x}_i = \begin{cases} 1, & \text{if } L_{post}(x_i) < 0 \\ 0, & \text{if } L_{post}(x_i) \geq 0 \end{cases}. \quad (2.54)$$

If the parity check

$$H \cdot \tilde{X}^T = 0^T \quad (2.55)$$

is satisfied or the current iteration number n_{ite} reaches the predefined maximum iteration number N_{ite}, the decoding process halts, and the estimated codeword $\tilde{X} = (\tilde{x}_0, \tilde{x}_1, \cdots, \tilde{x}_{N-1})$ is outputted. Otherwise, the decoder repeats the step $2 \sim 4$ for the next decoding iteration, and n_{ite} is increased by one.

2.3.3 Min-Sum Algorithm

For the sum-product algorithm in logarithmic domain, the horizontal step is the most computationally complex part because of the nonlinear function $\Psi(x) = -\ln(\tanh(\frac{x}{2}))$. Noting that the function $\Psi(x)$ is equal to its own inverse $\Psi^{-1}(x)$ over the range $x > 0$. This function is depicted in Fig. 2.10. In (2.51), it is the large values that dominate the summation. In fact, if there is a single large value $\Psi(|L_{x_l \to c_j}(x_l)|)$, the summation will be a large positive number, and the other terms do not matter. Since the function $\Psi(x)$ is monotonically decreasing for $x > 0$, a large value of $\Psi(|L_{x_l \to c_j}(x_l)|)$ corresponds to a small value of $|L_{x_l \to c_j}(x_l)|$. Hence, as an approximation it is possible to replace the term $\Psi^{-1}\left(\sum_{l \in N(j)\setminus\{i\}} \Psi(|L_{x_l \to c_j}(x_l)|)\right)$ by the minimum value of $|L_{x_l \to c_j}(x_l)|$ over all l in the relevant range $(l \in N(j) \setminus \{i\})$, using the fact that $\Psi^{-1}(\Psi(x)) = x$. In other words, the expression (2.51) can be approximated by

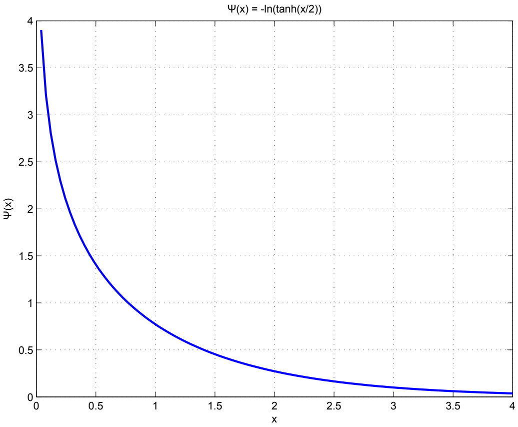


Figure 2.10: Plot of the $\Psi(x)$ function

$$L_{c_j \to x_i}(x_i) \approx \left(\prod_{l \in N(j)\setminus\{i\}} \operatorname{sgn}(L_{x_l \to c_j}(x_l)) \right) \min_{l \in N(j)\setminus\{i\}} (|L_{x_l \to c_j}(x_l)|). \tag{2.56}$$

The decoding procedure based on (2.56) and (2.52) is referred to **min-sum algorithm**. It is more practical for implementation due to its simplicity, however, it has a degradation in performance. Comparing the magnitude parts of (2.56) and (2.51), it is true that the former is no less than the latter. To decrease the difference between them, (2.56) can be modified as

$$L_{c_j \to x_i}(x_i) \approx \beta \cdot \left(\prod_{l \in N(j)\setminus\{i\}} \operatorname{sgn}(L_{x_l \to c_j}(x_l)) \right) \min_{l \in N(j)\setminus\{i\}} (|L_{x_l \to c_j}(x_l)|), \tag{2.57}$$

where β is the normalization factor with $0 < \beta \leq 1$ and usually to be $0.6 \sim 0.8$. The decoding procedure based on (2.57) and (2.52) is called **modify min-sum algorithm**. A further improvement using dynamic normalization technique is reported in [19].

2.3.4 Layered Belief Propagation Algorithm

An iteration for the decoding algorithm mentioned in Section 2.3.2 can mainly be decomposed into two phases, horizontal and vertical phases. At the first phase, the LDPC decoder operates the horizontal step for each row. Then, the decoder executes the vertical step for each column during the second phase. The horizontal operation for each row at the first phase can be carried out simultaneously, so as the vertical operations for columns at the second phase. Assume there is a parity check matrix with N columns and M rows, it corresponds to a Tanner graph with N bit nodes and M check nodes. According to different decoding scheduling, the implementation of LDPC decoders can be partitioned into two categories, fully parallel decoders and partially parallel decoders. A fully parallel decoder directly maps the corresponding Tanner graph into hardware and all the processing units are hard-wired according to the connectivity of the graph [20]. Thus it can achieve very high decoding throughput but suffer a large hardware cost. The partially parallel architecture groups several nodes of the graph into a subset and maps these nodes to a single processing unit by using time-division multiplexing [21]. It trades the decoding throughput for the reduction of hardware complexity. The parity check matrix of an LDPC code can be viewed as a collection of horizontal layers. Each layer represents a component code and is a subset of rows. The intersection of all these component codes forms the full LDPC code. A soft-input soft-output (SISO) decoder can be applied to each layer in sequence. As a layer starts to decode, it uses the latest messages as inputs which updated by the recently processed layers. In addition, a SISO decoder spends a sub-iteration to process a component code. After processing all the layers one time, we call it a iteration. Assume that the rows of the parity check matrix are grouped into non-overlapping subsets (layers) where each subset has the following property: the column weight in each layer is at most one as shown in Fig. 2.11. The layered approach to the log-BP algorithm is described as follows, and assuming that the AWGN channel and BPSK mapping are used.

$$H_{layered} = \left[\begin{array}{cccccccccccccccc}
1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0\\
\hline
0&0&0&0&1&0&0&0&0&0&0&1&0&1&0&0\\
0&0&0&0&0&1&0&0&0&0&1&0&0&0&1&0\\
0&0&0&0&0&0&1&0&0&1&0&0&0&0&0&1\\
0&0&0&0&0&0&0&1&1&0&0&0&1&0&0&0\\
\hline
0&1&0&0&0&0&0&1&0&0&0&0&0&0&1&0\\
0&0&1&0&1&0&0&0&0&0&0&0&0&0&0&1\\
1&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0\\
0&0&0&1&0&1&0&0&0&0&0&0&1&0&0&0
\end{array}\right]
\begin{array}{l}
\\ \text{Layer 1} \\ \\ \\ \\ \text{Layer 2} \\ \\ \\ \\ \text{Layer 3} \\ \\ \\
\end{array}$$

Figure 2.11: Rows of $H_{layered}$ are grouped into three layers

Step 1. (Initialization): Set the current iteration number $n_{ite} = 1$. For each edge existing in the corresponding Tanner graph, the message sent from check node c_j to bit node x_i is initialized to 0. Moreover, for $i = 0 \sim (N-1)$, the a posteriori information $L_{post}(x_i)$ is initialized to $L_{int}(x_i)$.

Step 2. (Messages updating): We apply step 2 to layers of the parity check matrix in a layer-by-layer manner. For each layer, the decoder executes the following three sub-step.

2.1 : For each bit node x_i participates in the current horizontal layer, message $L_{x_i \to c_j}(x_i)$ that corresponds to a particular check equation c_j is computed according to

$$L_{x_i \to c_j}(x_i) = L_{post}(x_i) - \underbrace{L_{c_j \to x_i}(x_i)}_{old}. \tag{2.58}$$

2.2 : For each check node c_j, message $L_{c_j \to x_i}(x_i)$ corresponding to variable node x_i which involved with the particular parity check equation c_j is computed according to

$$L_{c_j \to x_i}(x_i) = \left(\prod_{l \in N(j) \backslash \{i\}} \operatorname{sgn}(L_{x_l \to c_j}(x_l)) \right) \Psi^{-1} \left(\sum_{l \in N(j) \backslash \{i\}} \Psi(\left| L_{x_l \to c_j}(x_l) \right|) \right). \tag{2.59}$$

This is the same as (2.51) in conventional log-BP algorithm.

2.3 : The a posteriori information $L_{post}(x_i)$ for the current horizontal layer is updated according to

$$L_{post}(x_i) = L_{x_i \to c_j}(x_i) + \underbrace{L_{c_j \to x_i}(x_i)}_{new}, \tag{2.60}$$

where the term $L_{c_j \to x_i}$ is recently updated in step 2.2.

Step 3. (Hard decision and syndrome check): The codeword

$\tilde{X} = (\tilde{x}_0, \tilde{x}_1, \cdots, \tilde{x}_{N-1})$ is estimated by

$$\tilde{x}_i = \begin{cases} 1, & \text{if } L_{post}(x_i) < 0 \\ 0, & \text{if } L_{post}(x_i) \geq 0 \end{cases} \tag{2.61}$$

If the parity check

$$H \cdot \tilde{X}^T = 0^T \tag{2.62}$$

is satisfied or the current iteration number n_{ite} reaches the predefined maximum iteration number N_{ite}, the decoding process halts, and the estimated codeword $\tilde{X} = (\tilde{x}_0, \tilde{x}_1, \cdots, \tilde{x}_{N-1})$ is outputted. Otherwise, the decoder repeats the step 2 and 3 for the next decoding iteration, and n_{ite} is increased by one.

This layered decoding algorithm is a variation of the conventional BP algorithm, and could **speed up** the decoding convergence rate **around two times**. This is due to the optimized scheduling of reliability messages [22, 23]. In addition, the layered approach is suitable for the LDPC decoder with partially parallel architecture, and can lead to memory reduction in VLSI implementation.

Chapter 3

Tanner Graph Construction

Since an LDPC code can be specified by a parity check matrix or equivalently a Tanner graph, it is great important to construct a Tanner graph with good properties. In this chapter, we review two existing methods for constructing Tanner graphs, called progressive edge-growth (PEG) and PEG-QC algorithms, and introduce some factors that will affect the performance of LDPC codes.

3.1 Performance-Related Code Parameters

Before discussing construction of Tanner graphs, several factors that affect error-correcting performance of LDPC codes will be introduced. First is the degree distribution pair for a Tanner graph, we describe it in 3.1.1. Furthermore, the Tanner graph of a practical LDPC code usually contains cycles. Cycles in a Tanner graph will degrade the code performance, this will be depicted in 3.1.2. An LDPC code whose Tanner graph contains small stopping sets will suffer higher error floor. 3.1.3 discusses this phenomenon and gives two parameters to the set of variable nodes.

3.1.1 Degree Distribution Pairs

An LDPC code can be well represented by a Tanner graph, in which one set of nodes forms variable node set and the other becomes check node set. A Tanner graph is called a (d_v, d_c)-regular one if every variable node participates in d_v check nodes and every check node involves d_c variable nodes; otherwise, it is called irregular. For a (d_v, d_c)-regular

Tanner graph, the corresponding parity check matrix will contain d_v 1's for each column and d_c 1's for each row. The following figures show a (3, 6)-regular Tanner graph and its corresponding parity check matrix. For a given codeword length N and a given degree distribution pair (λ, ρ) [24], it forms an ensemble of codes by choosing edge, i.e., the connections between variable and check nodes, randomly. More precisely, we enumerate the edges emanating from the variable nodes in some arbitrary order and proceed in the same way with the edges emanating from the check nodes. Assume that the total number of edges is E. Then a code (a particular instance of this ensemble) can be specified by a permutation on E edges. By definition, all instances in this ensemble are equiprobable. However, in practice, the edges are not chosen entirely randomly since certain potentially ill events in the graph construction can be easily avoided. We say that a polynomial $\gamma(x)$

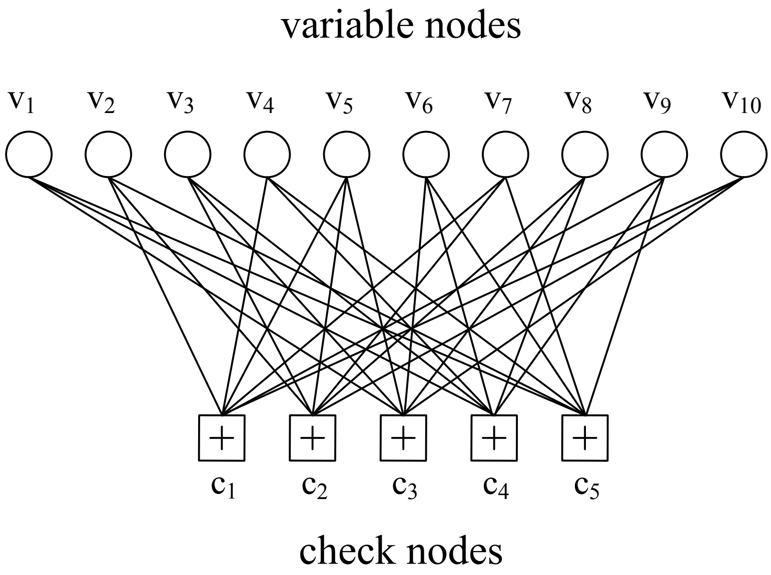


Figure 3.1: A (3, 6)-regular Tanner graph

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

Figure 3.2: Parity check matrix H corresponds to the graph given in Fig.3.1

of the form

$$\gamma(x) = \sum_{i \geq 2} \gamma_i x^{i-1} \tag{3.1}$$

is a *degree distribution* if $\gamma(x)$ has nonnegative coefficients and $\gamma(1) = 1$. Note that we associate the coefficient γ_i to x^{i-1} rather than x^i. We will see that this notation leads to very elegant and compact descriptions of the main results. Given a degree distribution pair (λ, ρ) associate to it a sequence of code ensembles $C^N(\lambda, \rho)$, where N is the length of the code,

$$\lambda(x) = \sum_{i=2}^{d_v} \lambda_i x^{i-1}, \tag{3.2}$$

and

$$\rho(x) = \sum_{i=2}^{d_c} \rho_i x^{i-1}. \tag{3.3}$$

$\lambda(x)$ and $\rho(x)$ specifies the variable and check node degree distribution, respectively. More precisely, λ_i (ρ_i) represents the fraction of edges emanating from variable (check) node of degree i. For example, for the (3, 6)-regular code we have $\lambda(x) = x^2$ and $\rho(x) = x^5$. The maximum *variable degree* and *check degree* is denoted by d_v and d_c, respectively. Assume that the code has N variable nodes. The number of variable nodes of degree i is then

$$N \frac{\lambda_i / i}{\sum_{j \geq 2} \lambda_j / j} = N \frac{\lambda_i / i}{\int_0^1 \lambda(x) dx} \tag{3.4}$$

and so the total number of edges emanating from all variable nodes E is equal to

$$E = N \sum_{i \geq 2} \frac{\lambda_i / i}{\int_0^1 \lambda(x) dx} i = N \frac{1}{\int_0^1 \lambda(x) dx}. \tag{3.5}$$

In the same manner, assuming that the code has M check nodes, E can also be expresses as

$$E = M \frac{1}{\int_0^1 \rho(x) dx}. \tag{3.6}$$

Equating these two expressions, (3.5) and (3.6), for E, we derive that

$$M = N \frac{\int_0^1 \rho(x) dx}{\int_0^1 \lambda(x) dx}. \tag{3.7}$$

Generically, assuming that all these check equations are linearly independent, we see that the *code rate* is equal to

$$r(\lambda, \rho) = \frac{N - M}{N} = 1 - \frac{\int_0^1 \rho(x) dx}{\int_0^1 \lambda(x) dx}. \tag{3.8}$$

It was proven that degree distribution pairs affect performance of LDPC codes. Degree distribution pairs of infinite long LDPC codes can be optimized by using *density evolution* which determines the performance threshold of these codes.

For parity check matrices with the same dimension $M \times N$, an irregular LDPC code with proper degree distribution pair usually outperforms a regular one. However, a finite length and irregular LDPC code with optimized degree distribution pair usually contains a large portion of degree-2 variable nodes, which degrades the code performance in *high-SNR* region. To partially overcome this problem, we can suppress the number of degree-2 variable nodes not more than the number of check nodes and assign the degree-2 variable nodes to parity-check bits of the codeword. In [25], it also provides two approaches. One is to convert part (or all) of these variable nodes to nodes of the next higher degree, say degree-3. Another is to increase every degree in the degree distribution of variable nodes by 1, i.e., degree-2 is converted to degree-3, degree-3 is converted to degree-4, and so on. This degree +1 adjustment is confirmed to be very effective for high-rate code. Moreover, a variable node with higher degree can provide better protection and show faster decoding convegence rate for the corresponding code bit. A strictly concentrated check node degree distribution

$$\rho(x) = \rho_i x^{i-1} + \rho_{i+1} x^i \tag{3.9}$$

maximizes the convergence speed of the code [26].

3.1.2 Cycles in Tanner Graph

If a graph satisfies that 1) doesn't contain self-loops, 2) is at most one edge between a pair of vertices, and 3) all edges of it are nondirected, it is called a *simple* graph [10]. For a simple graph, a closed path with l edges starting from a vertex v_j and ending at v_j is called a length-l cycle. Girth g refers to the length of the shortest cycle in a graph. If there is a Tanner graph without cycles, belief propagation algorithm can provide optimum decoding. However, for the practical LDPC codes, graphs often contain cycles. It is proven that cycles make iterative decoding algorithm become sub-optimal, which degrades error-correcting performance of LDPC codes. The following figures illustrate a Tanner graph with length-6 cycles and its corresponding parity check matrix.

For Tanner graphs with length-4 cycles, they show serious degradation in code per-

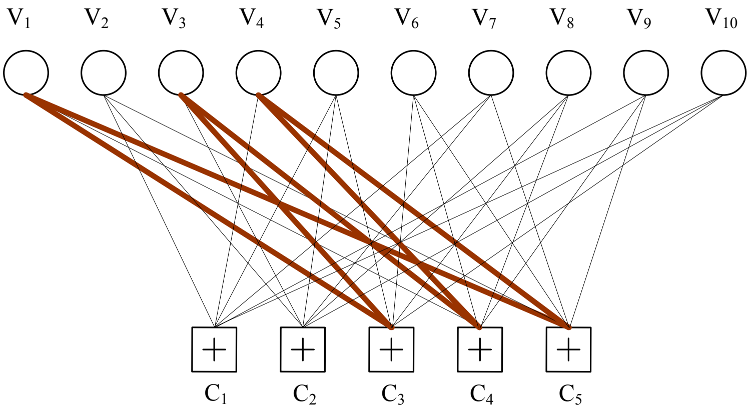


Figure 3.3: A Tanner graph with length-6 cycles

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

Figure 3.4: Parity check matrix H corresponds to the graph given in Fig.3.3

formance. In practice, we avoid constructing graphs with length-4 cycles and make girth of graphs as larger as possible. Local girth of variable node v_j is defined as the length of the shortest cycle passing through the node v_j. In [27], it confirmed not only the girth but also the average of local girth, belong to the Tanner graphs, affecting performance of LDPC codes. When designing Tanner graphs, we can take both girth and average of local girth into account and make lower-degree variable nodes have larger local girth to ensure better error-correcting performance of the resulting LDPC codes.

3.1.3 Stopping Sets in Tanner Graph

Stopping sets in Tanner graphs are defined as follows [28].

Definition 1: (S_d Stopping set) A variable node set is called an S_d set if it has d elements and all its neighbors are connected to it at least twice.

Fig. 3.5 gives a Tanner graph containing an S_3 stopping set and this S_3 set is composed of variable nodes $\{v_0, v_1, v_2\}$. In binary erasure channel (BEC), a set of variable nodes is called a erasure set if the corresponding code-bits of all these variable nodes are erasure. If there is a stopping set S_d contained in the erasure set, all these variable nodes in S_d can

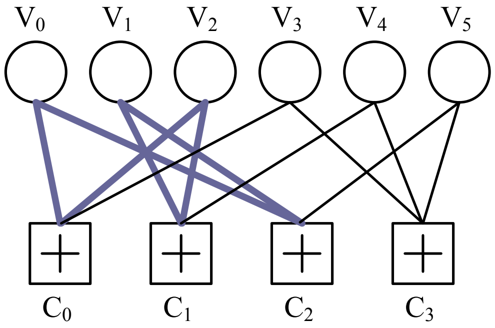


Figure 3.5: Illustration of a S_3 stopping set

not be determined through an iterative decoder [29]. It is also proven that every stopping set contains cycles if there are no degree-1 variable nodes in the Tanner graphs. This can be seen in Fig. 3.5. Moreover, it is proven that preventing small stopping sets can avoid small minimum distance which will incur performance loss in *error-floor* region. In order to avoid small stopping sets, we have to make the subsets of variable nodes have as many extrinsic check nodes as possible. An extrinsic check node of a variable node set is a check node that singly connects to this set. There are two quantities to evaluate the number of extrinsic check nodes, namely *extrinsic message degree* (EMD) and *approximate cycle EMD* (ACE).

Definition 2: The EMD of a variable node set is the number of extrinsic check nodes of this variable node set.

A variable node set with larger EMD has the better degree of connectivity to the rest of the graph. An approximate approach to calculate EMD for cycles is using approximate cycle EMD (ACE) and given as follows.

Definition 3: The ACE of a length $2d$ cycle is $\sum_i (d_i - 2)$, where d_i is the degree of the ith variable in this cycle.

3.2 PEG Tanner Graphs

Progressive edge-growth (PEG) method has been shown to be an effective and simple one to construct Tanner graphs in an edge-by-edge manner [9,10]. It maximizes the local girth of the proceeding variable node when adding an edge into the current graph. Code parameters specified in PEG method are highly flexible, so they can be chosen for the practical applications.

Before describing the PEG algorithm, we introduce some necessary definitions and notations on graph. The parity check matrix H with dimension $M \times N$ of an LDPC code can be represented by a (V, E) Tanner graph with N variable nodes and M check nodes. On the one side of the graph is the set of variable (bit) nodes $V_v = \{v_0, v_1, \cdots, v_{N-1}\}$ corresponds to the N columns of the matrix; on the other side of the graph is the set of check nodes $V_c = \{c_0, c_1, \cdots, c_{M-1}\}$ corresponds to the M rows of H. An edge denoted by $e(c_i, v_j)$ connects the variable node v_j with check node c_i, which corresponds to the 1 in the entry (i, j) of H. The nodes in V_c and V_v form the set $V = V_c \cup V_v$, and the collection of edges in the graph forms the set $E \subseteq V_c \times V_v$. Denote the variable degree sequence by

$$D_v = \{d_{v_0}, d_{v_1}, \cdots, d_{v_{N-1}}\} \tag{3.10}$$

in which d_{v_j} is the degree of variable node v_j, $0 \leq j \leq N-1$, in nondecreasing order, i.e., $d_{v_0} \leq d_{v_1} \leq \cdots \leq d_{v_{N-1}}$, and the parity-check degree sequence by

$$D_c = \{d_{c_0}, d_{c_1}, \cdots, d_{c_{M-1}}\} \tag{3.11}$$

in which d_{c_i} is the degree of parity-check node c_i, $0 \leq i \leq M-1$, and $d_{c_0} \leq d_{c_1} \leq \cdots \leq d_{c_{M-1}}$. Let's also partition the set of edges E as $E = E_{v_0} \cup E_{v_1} \cup \cdots \cup E_{v_{N-1}}$ in terms of V_v, where E_{v_j} contains all edges incident on variable node v_j. Moreover, denote the k-th edge incident on v_j by $E^k_{v_j}$, $0 \leq k \leq d_{v_j} - 1$. A subgraph of a graph $G = (V, E)$ is a graph whose node and edge set are subsets of those of G. For a given variable node v_j, define its neighborhood within depth l, $N^l_{v_j}$, as the set consisting of all check nodes reached by a subgraph (or a tree) spreading from variable node v_j within depth l. Its complementary set, $\overline{N}^l_{v_j}$, is defined as $V_c \setminus N^l_{v_j}$, i.e. V_c excludes $N^l_{v_j}$. The subgraph rooted from v_j is generated by means of unfolding the Tanner graph in a breadth-first way; we start from v_j, and traverse all edges incident on v_j; let these edges be $(v_j, c_{i_0}), (v_j, c_{i_1}), \cdots, (v_j, c_{i_{d_{v_j}-1}})$. Then we traverse all other edges incident on nodes $c_{i_0}, c_{i_1}, \cdots, c_{i_{d_{v_j}-1}}$, excluding $(v_j, c_{i_0}), (v_j, c_{i_1}), \cdots, (v_j, c_{i_{d_{v_j}-1}})$. This process continues until the desired depth is reached.

Here, we describe how to construct a Tanner graph using PEG algorithm as shown in Algorithm 1. First, we specify the graph parameters, i.e., the number of variable nodes N, the number of check nodes M, and the variable degree sequence D_v. Then, the following pseudo-code in Algorithm 1 can be used to construct the Tanner graph edge-by-edge.

Algorithm 1 PEG algorithm

```
for $j = 0$ to $N - 1$ do
  for $k = 0$ to $d_{v_j} - 1$ do
    if $k = 0$ then
      $E_{v_j}^0 \leftarrow e(c_i, v_j)$, where $E_{v_j}^0$ is the first edge incident to $v_j$ and $c_i$ is a check node such that it has the lowest check-node degree under the current graph setting $E_{v_0} \cup E_{v_1} \cup \cdots \cup E_{v_{j-1}}$.
    else
      expand a subgraph from variable node $v_j$ up to depth $l$ under the current graph setting such that the cardinality of $N_{v_j}^l$ stops increasing but is less than $M$, or $\overline{N}_{v_j}^l \neq \emptyset$ but $\overline{N}_{v_j}^{l+1} = \emptyset$, then $E_{v_j}^k \leftarrow e(c_i, v_j)$, where $E_{v_j}^k$ is the $k$-th edge incident to $v_j$ and $c_i$ is a check node picked from the set $\overline{N}_{v_j}^l$ having the lowest check-node degree.
    end if
  end for
end for
```

Whenever a subgraph from variable node v_j is expanded before adding an edge to the current graph, two situations can occur: 1) the cardinality of $N_{v_j}^l$ stops increasing but is smaller than M; (2) $\overline{N}_{v_j}^l \neq \emptyset$ but $\overline{N}_{v_j}^{l+1} = \emptyset$. In the first case, not all check nodes are reachable from v_j, so the PEG algorithm chooses the one that is not reachable (in $\overline{N}_{v_j}^l$), thus not creating any additional cycle. This often occurs in the initial phase of graph construction. In the second case, all check nodes are reachable from v_j, and the algorithm chooses the one that is at the largest distance from v_j, at depth $l + 1$, so that the cycle created by establishing an edge is of the largest possible length $2(l + 2)$.

However, during going through the procedure in Algorithm 1, we may still face a situation in which multiple choices exist because multiple check node in $\overline{N}_{v_j}^l$ might have the same lowest degree, particularly in the initial phase of PEG construction. There are two main approaches to solve this problem. One is to randomly select one of these check nodes. The other is to always select one according to its position in the order of

$c_0, c_1, \cdots, c_{M-1}$. For example, we can first sort the check node in $\overline{N}_{v_j}^{l}$ that have the same lowest degree according to their subscripts, and then always pick the first one. There are other versions of PEG algorithm, nongreedy and look-ahead-enhanced versions, have also been discussed in [10], but none of them provides any nonnegligible performance improvement over the standard PEG.

3.3 PEG-QC LDPC Codes

The parity check matrices consists of submatrices are attractive due to their hardware-friendly properties. They reduce the hardware complexity and show performance comparable to random LDPC codes at the short to medium length. The codes constructed by PEG method are proven to be good enough, however, the positions of 1's of the corresponding H are random. Z. Li et al. [13] proposed one kind of structured LDPC codes based on PEG construction. They added a circulant constraint into original PEG algorithm. The parity check matrix generated by this PEG-QC method can be put in the form below

$$H = \begin{bmatrix} H_{0,0} & H_{0,1} & \cdots & H_{0,n-1} \\ H_{1,0} & H_{1,1} & \cdots & H_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ H_{m-1,0} & H_{m-1,1} & \cdots & H_{m-1,n-1} \end{bmatrix}, \tag{3.12}$$

where $H_{i,j}$ is a $p \times p$ circulant or all-zero matrix, and m and n are two positive integer with $m < n$. The circulant matrix is a square matrix, where each row vector is rotated one element to the right relative to the preceding row vector. Equation (3.13) illustrates a circulant matrix with dimension 4×4.

$$H_{i,j} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix} \tag{3.13}$$

The null space of H gives an LDPC code over $GF(2)$ of length $N = np$. The rank of H is at most mp. Hence the code rate is at least $1 - \frac{m}{n}$. Assume that the Tanner graph has to be designed has $N = np$ variable nodes and $M = mp$ check nodes.

To generate a structured LDPC code with circulant size $p \times p$ by this PEG-QC algorithm, first it needs to divide all variable nodes and check nodes into small subgroups with each group has p nodes. Then it adds edges to the Tanner graph group by group instead of node by node compared to the original PEG algorithm. For each variable node group, it only has to expand trees for the first variable node in the group to find its optimized edges. The edges of other variable nodes in the same group can be determined by the circulant constraint automatically. Here we give the pseudo-code of PEG-QC algorithm as follows. It had been shown that the codes constructed by this PEG-QC method exhibited performance comparable to other kinds of QC-LDPC codes.

Algorithm 2 PEG-QC algorithm

```
for $j = 0$ to $n - 1$ do
    for $k = 0$ to $d_{v_{jp}} - 1$ do
        if $k = 0$ then
            $E^0_{v_{jp}} \leftarrow e(c_i, v_{jp})$, where $E^0_{v_{jp}}$ is the first edge incident to $v_{jp}$ and $c_i$ is a check node such that it has the lowest check-node degree under the current sub-graph setting (randomly pick one if there are more than one such check nodes).
        else
            Expand a tree from variable node $v_{jp}$ up to depth $l$ under the current graph setting such that the cardinality of $N^l_{v_{jp}}$ stops increasing but is less than $M$, or $\overline{N}^l_{v_{jp}} \neq \emptyset$ but $\overline{N}^{l+1}_{v_{jp}} = \emptyset$, then $E^k_{v_{jp}} \leftarrow e(c_i, v_{jp})$, where $E^k_{v_{jp}}$ is the $k$-th edge incident to $v_{jp}$ and $c_i$ is a check node picked from the set $\overline{N}^l_{v_{jp}}$ having the lowest check-node degree (randomly pick one if there are more than one such check nodes).
        end if
        for $r = 1$ to $p - 1$ do
            $E^k_{v_{jp+r}} \leftarrow e(c_{\lfloor i/p \rfloor \cdot p + mod(i+r,p)}, v_{jp+r})$, where $E^k_{v_{jp+r}}$ is the $k$-th edge incident to $v_{jp+r}$, and $c_{\lfloor i/p \rfloor \cdot p + mod(i+r,p)}$ is the check node which cyclic shifts $r$ positions of $c_i$ in the circulant $H_{\lfloor i/p \rfloor, j}$.
        end for
    end for
end for
```

Table 3.1: Sets of code parameters which will be used hereafter

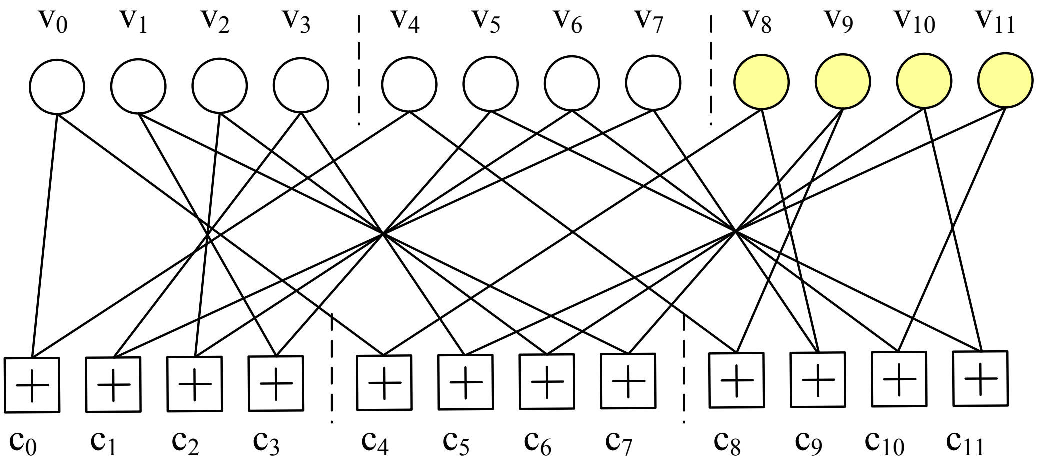
Index	Type	Rate	Dimension of H	Size of $H_{i,j}$
I	irregular	1/2	1280×2560	32×32
II	irregular	3/4	640×2560	32×32
III	irregular	7/8	320×2560	32×32
IV	(3, 27)-regular	8/9	512×4608	128×128

Table 3.2: Occurrence probabilities of codes with various girth constructed by PEG-QC algorithm

Index	Algorithm	Prob. of girth=4	Prob. of girth=6
I	PEG-QC	36%	64%
II	PEG-QC	55%	45%
III	PEG-QC	73%	27%
IV	PEG-QC	100%	0%

However, through simulation, we find that **PEG-QC algorithm is much easier to construct a Tanner graph with short cycles which degrade the error-correcting performance of an LDPC code. Moreover, it is sometimes impossible to construct a graph without 4-cycles**. Table 3.1 gives four sets of code parameters whose degree distribution are optimized from density evolution [24]. For irregular ones, the maximum variable-node degree d_v are 12 for both rate 1/2 and 3/4 and 9 for rate 7/8. We construct code ensembles, with parameters specified in Table 3.1, by above-mentioned PEG-QC algorithm. Each code ensemble contains one hundred LDPC codes. Table 3.2 shows the occurrence probabilities of codes with various girth constructed by PEG-QC algorithm. We observe that Tanner graphs constructed by PEG-QC algorithm are much easier to contain short cycles. This is due to the improper **circulant** constraint.

Here, we take a small graph for example. Assume there is a Tanner graph with 3 variable node subgroups and 3 check node subgroups, with each group has 4 nodes. We



(a) Graph setting before adding 3rd edge to v_8

$$H_{current} = \begin{array}{cccc|cccc|cccc|c}
v_{11} & v_{10} & v_9 & v_8 & v_7 & v_6 & v_5 & v_4 & v_3 & v_2 & v_1 & v_0 & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & c_0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & c_1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & c_2 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & c_3 \\
\hline
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & c_4 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & c_5 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & c_6 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & c_7 \\
\hline
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & c_8 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & c_9 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & c_{10} \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & c_{11}
\end{array}$$

(b) The corresponding parity check matrix of Fig. 3.6(a)

Figure 3.6: (a) Current graph setting by using PEG-QC algorithm, (b) and its corresponding parity check matrix

specify the variable degree sequence as

$$\begin{aligned} D_v &= \{\{d_{v_0}, d_{v_1}, d_{v_2}, d_{v_3}\}, \{d_{v_4}, d_{v_5}, d_{v_6}, d_{v_7}\}, \{d_{v_8}, d_{v_9}, d_{v_{10}}, d_{v_{11}}\}\} \\ &= \{\{2,2,2,2\}, \{2,2,2,2\}, \{3,3,3,3\}\} \end{aligned} \tag{3.14}$$

in a non-decreasing order. Before adding the 3rd edge of variable node v_8 into the graph, by using PEG-QC algorithm, the graph is set as shown in Fig. 3.6(a). In this case, we always choose the check node with smallest index when there are more than one candidates. We continue to add the 3rd edge of variable node v_8 to this graph through operating tree spreading from v_8 as shown in Fig. 3.7. According to the PEG-QC algorithm, $\overline{N}_{v_8}^{4} \neq \emptyset$ but $\overline{N}_{v_8}^{4+1} = \emptyset$, we will add an edge $e(c_6, v_8)$ to this graph. The resulting graph and parity check matrix are showed in Fig. 3.8, in which the dash lines are generated from the circulant constraint. It is obvious that **the new edge accompanies length-4 cycles**.

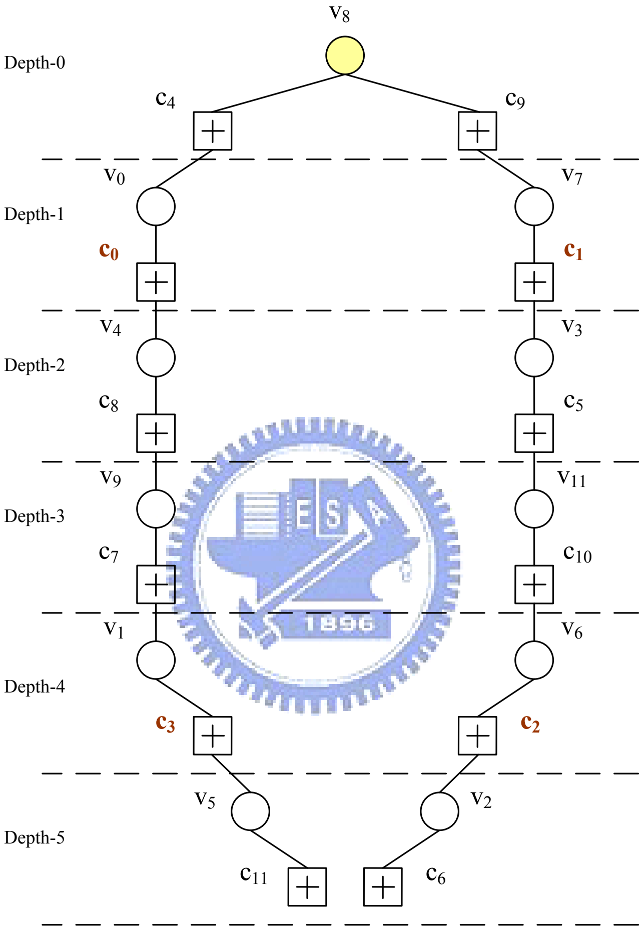
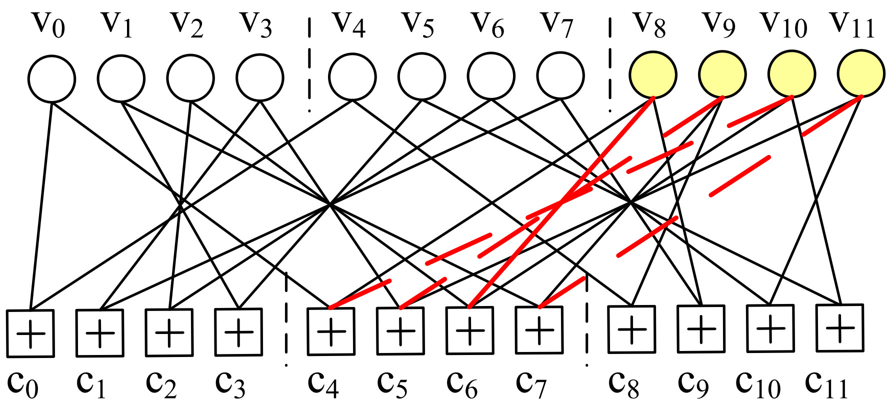
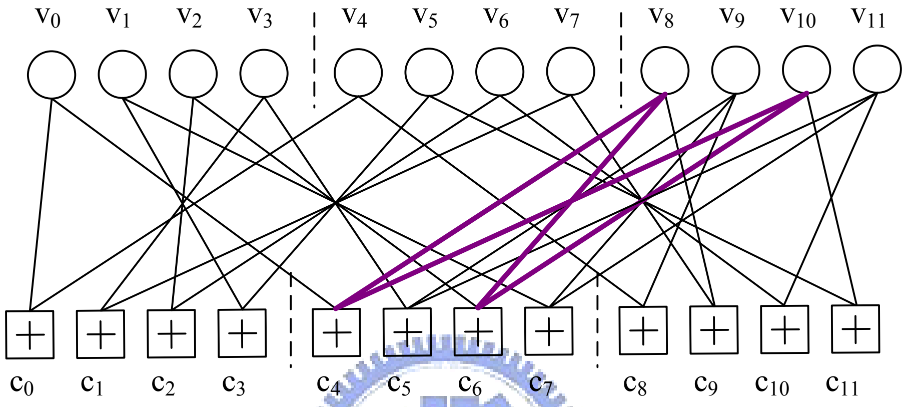


Figure 3.7: Tree spreading from variable node v_8


v0 v1 v2 v3 v4 v5 v6 v7 v8 v9 v10 v11
c0 c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11

(a) Tanner graph constructed by PEG-QC algorithm


v0 v1 v2 v3 v4 v5 v6 v7 v8 v9 v10 v11
c0 c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11

(b) The 4-cycle contained in the above figure

$$
H_{PEG-QC} = \left[\begin{array}{cccc|cccc|cccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
\hline
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
\hline
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0
\end{array}\right]
$$

(c) The resulting parity check matrix H

Figure 3.8: The Tanner graph constructed by PEG-QC algorithm, and its corresponding parity check matrix. There are 4-cycles in this graph.

Chapter 4

Design of PEG-Based Structured LDPC Codes

As shown in Section 3.3, the circulant constraint of PEG-QC algorithm makes it much easier to construct Tanner graphs with short cycles. In other words, because there are more than one 1 in a row of a circulant submatrix, the corresponding Tanner graph is easier to contain short cycles. In this chapter, we add a more strict constraint, compared with PEG-QC algorithm, to original PEG algorithm and propose a general method to construct structured Tanner graphs. For the consideration of encoding complexity and error floor, the modifications of the proposed algorithm are also discussed. Simulation results show that the proposed algorithm can suppress the probability to generate a graph with short cycles. Moreover, it confirms that our codes, in terms of bit error rate (BER) or packet error rate (PER), outperform PEG-QC LDPC codes and are better than the codes in IEEE 802.16e.

The remainder of this chapter is organized as follows. Section 4.1 gives several definitions and notations will be used and describes the proposed CP-PEG algorithm. In Section 4.2 and 4.3, we depict the modifications of CP-PEG algorithm. Finally, the code performance will be presented in Section 4.4.

4.1 Proposed Structured LDPC Codes

Here, we add a more strict constraint, called **circulant permutation (CP)** constraint, to original PEG algorithm. The parity check matrix H generated by the proposed **CP-PEG** algorithm is also structured as shown in equation (3.12). We write it again in equation (4.1). However, the submatrix $H_{i,j}$ of the proposed parity check matrices is either a $p \times p$ **circulant permutation** or all-zero matrix. A circulant permutation matrix is a special case of the circulant matrix, it contains exactly one 1 for each row vector, i.e. it is a right cyclic version of a $p \times p$ identity matrix $I_{p\times p}$. Thus, we can denote each submatrix $H_{i,j}$ of the proposed parity check matrices more concise as it in IEEE 802.11n.

$$H = \begin{bmatrix} H_{0,0} & H_{0,1} & \cdots & H_{0,n-1} \\ H_{1,0} & H_{1,1} & \cdots & H_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ H_{m-1,0} & H_{m-1,1} & \cdots & H_{m-1,n-1} \end{bmatrix} \tag{4.1}$$

Using the same notations and definitions as in Section 3.2, we further give several definitions and notations. Assume there are N variable nodes and M check nodes in the Tanner graph. We partition the N variable nodes and M check nodes into subgroups where each group has p nodes. After partition, it forms n variable node subgroups and m check node subgroups. We also partition edges E, in term of variable node subgroup vsg, as $E = E_{vsg_0} \cup E_{vsg_1} \cup \cdots \cup E_{vsg_{n-1}}$, with E_{vsg_j} containing all edges emanate from jth variable node subgroup vsg_j. Check node subgroup csg_i is said to be connected to variable node subgroup vsg_j if there is an edge $e(c_{ip+r}, v_{jp+r}) \in E_{vsg_j}$, where $c_{ip+r} \in csg_i$, $v_{jp+r} \in vsg_j$, $0 \leq i \leq (m-1)$, $0 \leq j \leq (n-1)$, and $0 \leq r \leq (p-1)$. Moreover, if check node subgroup csg_i is connected to variable node subgroup vsg_j under current graph setting, check nodes contained in csg_i form the set N_{vsg_j}. In the proposed algorithm, we won't connect the proceeding variable node to the check nodes in N_{vsg_j}. The complementary set of N_{vsg_j}, $\overline{N}_{vsg_j}$, is defined as the check-node set $V_c \setminus N_{vsg_j}$, i.e. V_c excludes N_{vsg_j}. $\overline{N}^{l}_{v_{jp+r}}$ is the complement of $N^{l}_{v_{jp+r}}$ which is set of all check nodes reached by a tree spreading from variable node v_{jp+r} within depth l. A complementary set $\overline{R}^{l}_{v_{jp+r}}$ is defined as $\overline{R}^{l}_{v_{jp+r}} = \overline{N}_{vsg_j} \cap \overline{N}^{l}_{v_{jp+r}}$, where $v_{jp+r} \in vsg_j$. Finally, a variable-subgroup degree sequence is defined as $D_{vsg} = \{d_{vsg_0}, d_{vsg_1}, \cdots, d_{vsg_{n-1}}\}$, where $d_{vsg_0} \leq d_{vsg_1} \leq \cdots \leq d_{vsg_{n-1}}$. In a variable-subgroup degree sequence, d_{vsg_j} denotes the degree of v_{jp+r}, where $v_{jp+r} \in vsg_j$ and

$0 \leq r \leq (p-1)$.

Then, we describe how to design a Tanner graph using the proposed CP-PEG algorithm. First, we specify the number of variable node subgroups n and the number of check node subgroups m, with each subgroup has p nodes. For the specified variable-subgroup degree sequence D_{vsg}, construction of the Tanner graph is described by the following Algorithm 3 which adds edges to Tanner graph group by group instead of node by node.

Algorithm 3 CP-PEG Algorithm

```
for $j = 0$ to $n-1$ do
    for $k = 0$ to $d_{vsg_j} - 1$ do
        if $k = 0$ then
            $E^0_{v_{jp}} \leftarrow e(c_i, v_{jp})$, where $c_i$ is a check node with lowest check-node degree under the current sub-graph setting $E_{vsg_0} \cup E_{vsg_1} \cup \cdots \cup E_{vsg_{j-1}}$ (randomly pick one if such check nodes are more than one).
        else
            Expand a tree from variable node $v_{jp}$ up to depth $l$ under the current graph setting such that the cardinality of $N^l_{v_{jp}}$ stops increasing but is less than $M$, or $\overline{R}^l_{v_{jp}} \neq \emptyset$ but $\overline{R}^{l+1}_{v_{jp}} = \emptyset$, then $E^k_{v_{jp}} \leftarrow e(c_i, v_{jp})$, where $c_i$ is a check node picked from $\overline{R}^l_{v_{jp}}$ having the lowest check-node degree (randomly pick one if such check nodes are more than one).
        end if
        for $r = 1$ to $p-1$ do
            $E^k_{v_{jp+r}} \leftarrow e(c_{\lfloor i/p \rfloor \cdot p + mod(i+r,p)}, v_{jp+r})$, where $E^k_{v_{jp+r}}$ is the $k$-th edge incident to $v_{jp+r}$, and $c_{\lfloor i/p \rfloor \cdot p + mod(i+r,p)}$ is the check node which cyclic shifts $r$ positions of $c_i$ in $H_{\lfloor i/p \rfloor, j}$.
        end for
    end for
end for
```

In Algorithm 3, $E^k_{v_{jp}}$ is the k-th edge incident on v_{jp} and we call this depicted **else-statement** as **tree spreading procedure** for convenience. Due to the circulant permutation form of $H_{i,j}$, the proposed LDPC codes are suitable for layered decoding mentioned

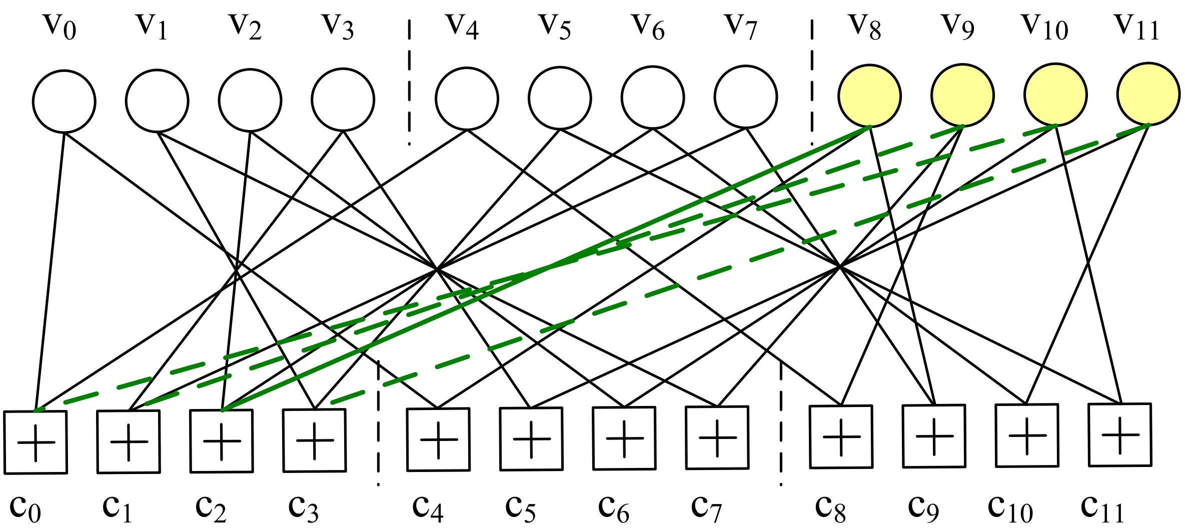
Table 4.1: Occurrence probabilities of codes with various girth constructed by the proposed CP-PEG and PEG-QC algorithms

Index	Algorithm	Prob. of girth=4	Prob. of girth=6
I	PEG-QC	36%	64%
	CP-PEG	0%	100%
II	PEG-QC	55%	45%
	CP-PEG	0%	100%
III	PEG-QC	73%	27%
	CP-PEG	16%	84%
IV	PEG-QC	100%	0%
	CP-PEG	0%	100%

in Section 2.3.4.

We construct a small graph, as previously done in Section 3.3, by the proposed CP-PEG algorithm. It contains 3 variable node subgroups and 3 check node subgroups with $D_{vsg} = \{d_{vsg_0}, d_{vsg_1}, d_{vsg_2}\} = \{2, 2, 3\}$. Before adding the 3rd edge of variable node v_8 into graph, the graph is the same as shown in Fig. 3.6(a), in which we always choose the check node with smallest index when there are more than one candidate. For the 3rd edge of variable node v_8, by using CP-PEG algorithm, we expand a tree from v_8 as shown in Fig. 3.7. Instead of selecting the check node c_6, we pick up the node c_2. It is because of $\overline{R}_{v_8}^{4} \neq \emptyset$ but $\overline{R}_{v_8}^{4+1} = \emptyset$. The resulting Tanner graph and parity check matrix are given in Fig. 4.1. **It doesn't contain 4-cycles anymore**.

We further use the sets of code parameters specified in Table 3.1 to construct 100 codes for each ensemble by the proposed CP-PEG algorithm. The resulting occurrence probabilities of codes with various girth are presented in Table 4.1. For the purpose of comparison, the results had been shown in Table 3.2 are also listed in Table 4.1. It is obvious that Tanner graphs constructed by the proposed CP-PEG algorithm have lower probabilities to contain short cycles than those by PEG-QC algorithm.



(a) Tanner graph constructed by CP-PEG algorithm

$$H_{CP-PEG} = \left[\begin{array}{cccc|cccc|cccc}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
\hline
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
\hline
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0
\end{array}\right]$$

(b) The resulting parity check matrix H

Figure 4.1: (a) The Tanner graph constructed by CP-PEG algorithm, (b) and its corresponding parity check matrix

4.2 ALT Form for Encoding Consideration

Assume that the parity check matrix H is in *approximate lower triangular* (ALT) form as previously mentioned in Section 2.2.4. Let $X = (u, p_1, p_2)$ is a codeword of H where u denotes the systematic part, p_1 and p_2 combined denote the parity part, p_1 has length $g = \gamma \cdot p$, and p_2 has length $(m - \gamma) \cdot p$. Given a message u, the systematic encoding can be achieved by

$$p_1^T = -\Phi^{-1}(-ET^{-1}A + C)u^T \tag{4.2}$$

and

$$p_2^T = -T^{-1}(Au^T + Bp_1^T), \tag{4.3}$$

where $\Phi = -ET^{-1}B + D$ is assumed to be nonsingular.

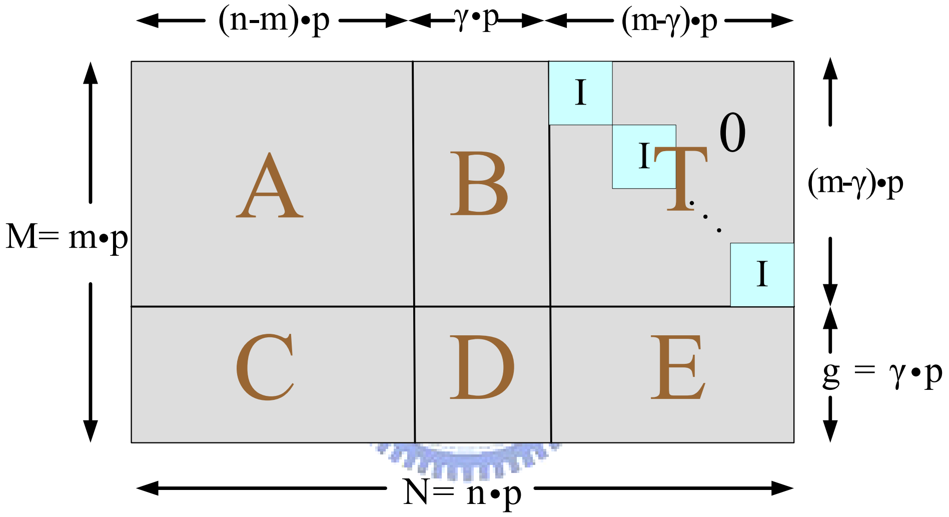


Figure 4.2: The parity check matrix in approximate lower triangular form

For the encoding consideration, we can restrict the structured parity check matrix generated by the proposed CP-PEG algorithm in ALT form [30] as shown in Fig. 4.2 and assign the variable nodes with higher degree to be the systematic part to get better protection from noises. This can be accomplished by the following ALT-CP-PEG algorithm.

In the encoding procedure for codes with ALT form, the overall complexity to determine p_1 and p_2 are $O(N + g^2)$ and $O(N)$, respectively. To reduce the complexity, we choose g (or γ) as small as possible under degree distribution setting D_{vsg}. Then, after constructing a parity check matrix H, we perform **block-column permutation** to make

Algorithm 4 ALT-CP-PEG Algorithm

```
for $j = 0$ to $n - 1$ do
  for $k = 0$ to $d_{vsg_j} - 1$ do
    if $k = 0$ then
      if $(0 \leq j \leq m - \gamma - 1)$ then
        $E^0_{v_{jp}} \leftarrow e(c_{(m-\gamma-j)\cdot p-1}, v_{jp})$, where $E^0_{v_{jp}}$ is the first edge incident to $v_{jp}$. This edge corresponds to the 1 in the diagonal line of submatrix $T$ of matrix $H$.
      else
        $E^0_{v_{jp}} \leftarrow e(c_i, v_{jp})$, where $c_i$ is a check node with lowest check-node degree under the current sub-graph setting $E_{vsg_0} \cup E_{vsg_1} \cup \cdots \cup E_{vsg_{j-1}}$ (randomly pick one if such check nodes are more than one).
      end if
    else
      if $(0 \leq j \leq m - \gamma - 1)$ then
        Setup that $csg_0, csg_1, \cdots, csg_{m-\gamma-2-j}$ are connected to $vsg_j$. This can ensure that submatrix $T$ is in lower triangular form.
      end if
      Expand a tree from variable node $v_{jp}$ up to depth $l$ under the current graph setting such that the cardinality of $N^l_{v_{jp}}$ stops increasing but is less than $M$, or $\overline{R}^l_{v_{jp}} \neq \emptyset$ but $\overline{R}^{l+1}_{v_{jp}} = \emptyset$, then $E^k_{v_{jp}} \leftarrow e(c_i, v_{jp})$, where $c_i$ is a check node picked from $\overline{R}^l_{v_{jp}}$ having the lowest check-node degree (randomly pick one if such check nodes are more than one).
    end if
    for $r = 1$ to $p - 1$ do
      $E^k_{v_{jp+r}} \leftarrow e(c_{\lfloor i/p \rfloor \cdot p + mod(i+r,p)}, v_{jp+r})$, where $E^k_{v_{jp+r}}$ is the $k$-th edge incident to $v_{jp+r}$, and $c_{\lfloor i/p \rfloor \cdot p + mod(i+r,p)}$ is the check node which cyclic shifts $r$ positions of $c_i$ in $H_{\lfloor i/p \rfloor, j}$.
    end for
  end for
end for
```

Φ nonsingular and keep the resulting H' structured. This is usually possible when H is not rank deficient. By the way, when operating the block-column permutation, Φ is made to be an identity matrix I if possible. This can further reduce the complexity because of $\Phi^{-1} = I^{-1} = I$.

4.3 EMD Criterion for Lowering Error Floor

Comparing with regular codes, irregular codes with optimized degree distribution often suffer higher error floor. In the *tree spreading procedure* of the proposed CP-PEG and ALT-CP-PEG algorithms, among all check-node candidates with the same check-node degree, we choose one at random. For irregular PEG Tanner graph, similar to above situation, [31] suggests us to choose the check node that maximizes the minimum ACE for the new cycles (ACE criterion). This causes the performance improvement in error floor region. For example, in Fig. 4.3, it suggests us to choose the check node with $ACE = 3$.

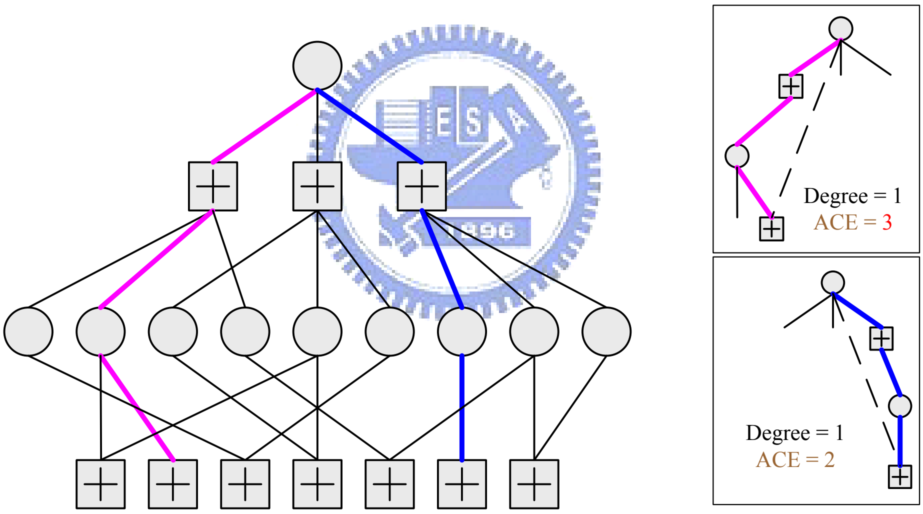


Figure 4.3: A simple example of ACE criterion

However, it may be still more than one candidate after adopting this criterion. In [32], it further gives an EMD criterion, to choose the check node that induces a subgraph with highest EMD, which improves the performance in error floor region further. Take Fig. 4.4 for example, the check node with $EMD = 3$ is more proper than the other to be connected. We adopt the EMD criterion into our code construction procedure to choose a proper check node from $\overline{R}^{l}_{v_{jp}}$, and confirm that it also lowers the error floor of our codes.

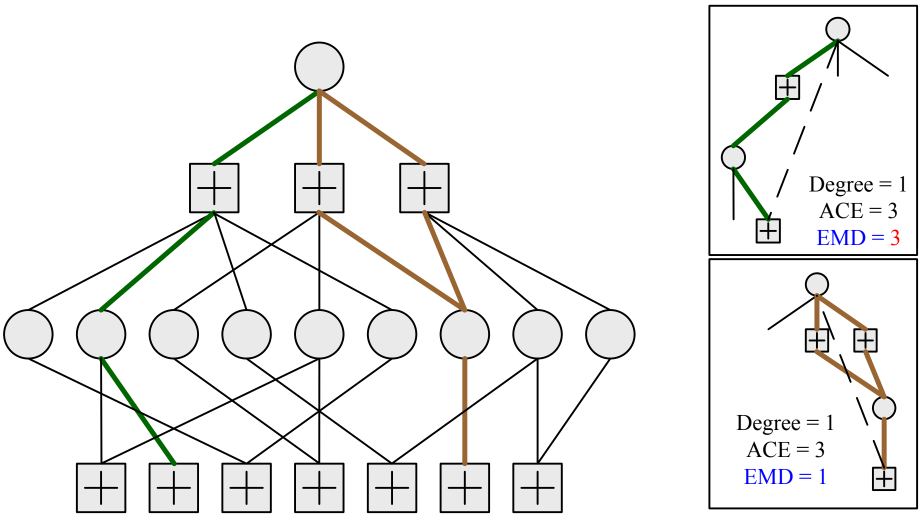


Figure 4.4: A simple example of EMD criterion

4.4 Simulation Results and Comparison

In this section, we study the performance of codes constructed by the proposed algorithm and give some comparisons. Randomly generated data with binary phase-shift keying (BPSK) mapping are simulated through a zero-mean additive white Gaussian noise (AWGN) channel. First, we construct codes by the proposed algorithm, with parameters specified in Table 3.1, and compare the code performance with those constructed by previously mentioned PEG-QC algorithm. Except for the (3,27)-regular LDPC code, all the other codes constructed by the proposed algorithm are in approximate lower triangular form. Moreover, we choose the codes with girth 6 for simulation except for the (3,27)-regular LDPC code based on PEG-QC algorithm. This is because that it doesn't report codes with girth larger than 4 even then it has constructed more than 10000 codes.

The performance comparison of $(4608, 4096)$ and $(2560, 2240)$ LDPC codes are given in Fig. 4.5 and Fig. 4.6, respectively, with maximum 80 iterations. As these two figures show, our codes have performance not worse than that of the codes constructed by PEG-QC algorithm. In Fig. 4.7, the 2560-bit rate 1/2 code constructed by the proposed ALT-CP-PEG algorithm shows much better performance. Lower bit error rate (BER) and packet error rate (PER) are perceived as compared to PEG-QC LDPC codes. Fig. 4.8 consists of three codes, all of them are of length 2560 and rate 3/4. As the figure shows, our codes outperform PEG-QC LDPC codes, especially a PER improvement of one order in magnitude in high-SNR region. Moreover, we compare the ALT-CP-PEG LDPC code with EMD criterion to that without EMD criterion. The former has lower error floor, which is lower than BER of 10^{-7}, and it shows better performance than the latter in high-SNR region. This confirms that the EMD criterion is also suitable for our algorithms.

To further compare the performance of above-mentioned ALT-CP-PEG LDPC codes, the following figures are presented. Fig. 4.9 $\sim$ 4.11 focus on the convergence rate of iterative decoding, and codes with various maximum numbers of iteration are simulated. Comparing performance under 80 iterations to those under 10 iterations, we can see that the high-code-rate code (rate = 7/8) converges faster than the code with lower code-rate (rate = 1/2). Moreover, because our codes contain at most one 1 in each column of submatrix $H_{i,j}$, we can use layered decoding algorithm referred in Section 2.3.4 to speed up the convergence rate. Fig. 4.12 and Fig. 4.13 show that they can achieve performance similar to a traditional log-BP decoder, but only half of the iterations are required.

To trade off the complexity with error-correcting performance, we can use the min-sum algorithm introduced in Section 2.3.3 instead of log-BP to decode LDPC codes. Fig. 4.14 shows that min-sum algorithm introduces about 0.7 dB SNR loss at BER = 10^{-5} when both decoders perform 15 decoding iterations. However, we can compensate the loss by modified min-sum algorithm. The result is also shown in Fig. 4.14, where β is the normalization factor as referred in equation (2.57). Fig. 4.15 shows the same trend but less performance loss compared to Fig. 4.14. In Fig. 4.16, we can see that the performance by using modified min-sum algorithm outperforms that by using log-BP under BER = 10^{-6}. This may result from cycles in the corresponding Tanner graph of this LDPC code. Log-BP algorithm is not an optimum solution for an cyclic graph, so the modified min-sum algorithm may perform better.

At the end of this section, we compare performance of codes constructed based on the proposed method to that of the irregular LDPC code adopted in IEEE 802.16e. Here, these two codes have the same degree distribution pair and the same submatrices size. Both codes have length as 2304 and rate as 1/2, and their parity check matrices are both in ALT form. Girth of the code constructed by our algorithm is 8, however, that of the code in IEEE 802.16e is 6. As shown in Fig. 4.17, we can see that our code slightly outperforms the IEEE 802.16e LDPC code. Moreover, because of the ALT form, our code remains encoding complexity similar to that of IEEE 802.16e.

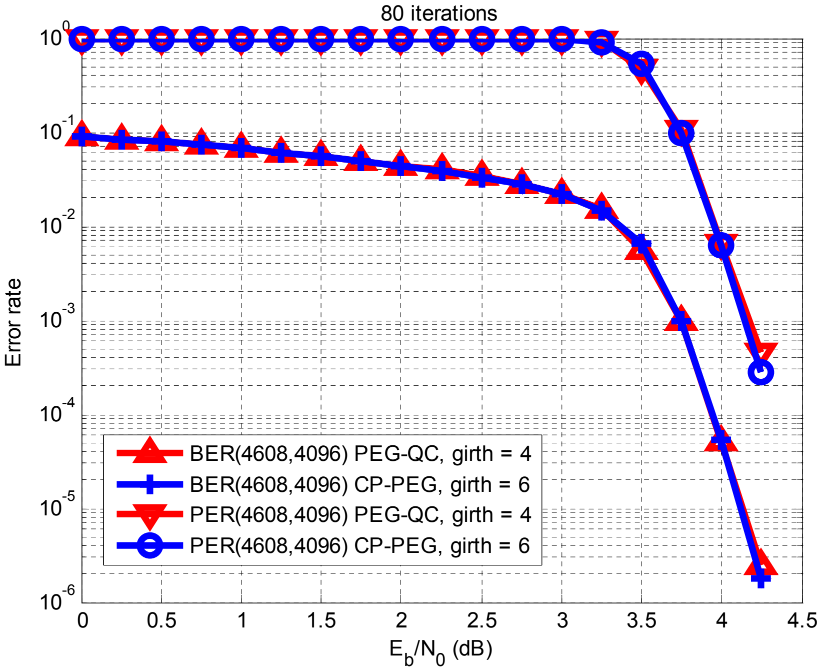


Figure 4.5: Performance comparison of the regular codes with rate 8/9 constructed by the proposed and PEG-QC algorithms

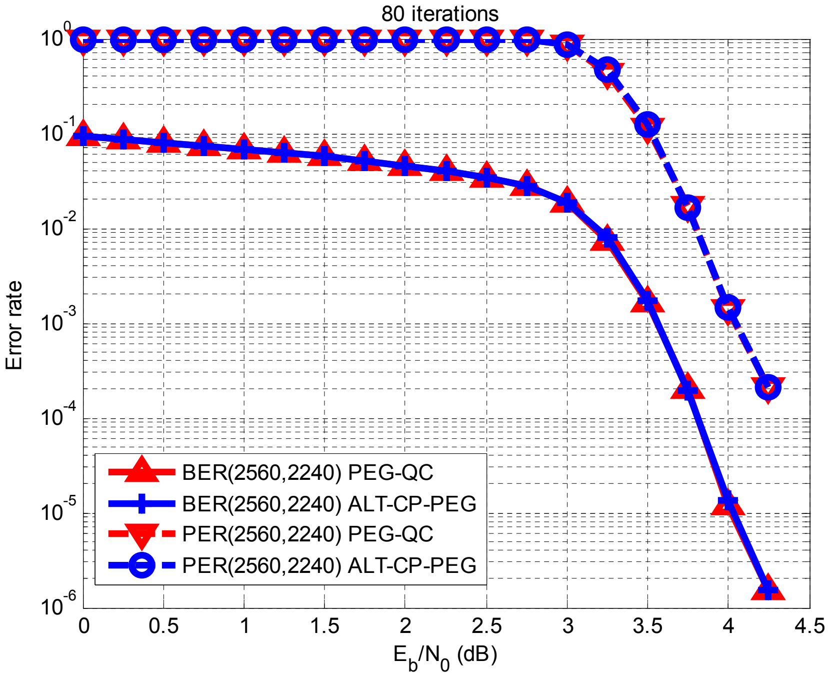


Figure 4.6: Performance comparison of the irregular codes with rate 7/8 constructed by the proposed and PEG-QC algorithms

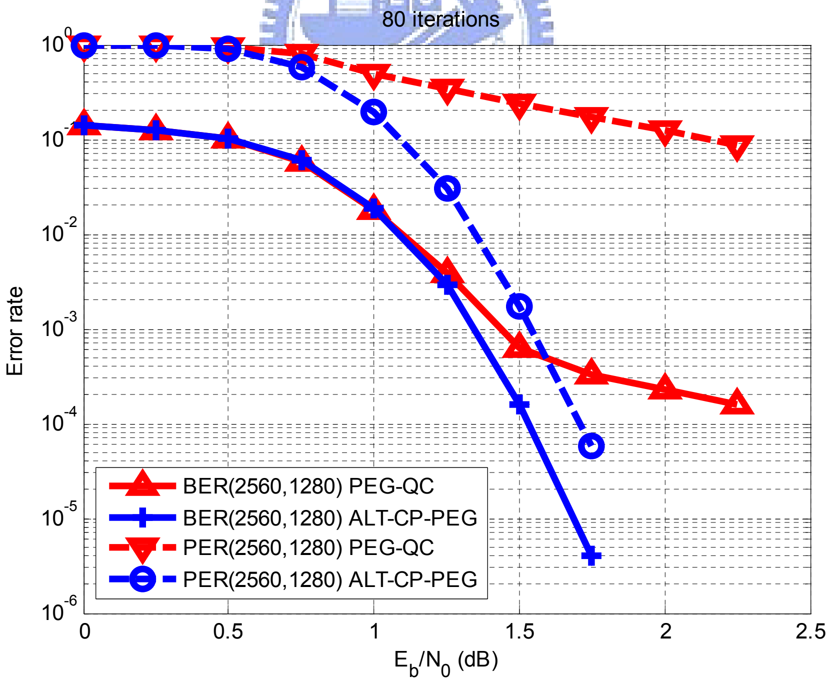


Figure 4.7: Performance comparison of the irregular codes with rate 1/2 constructed by the proposed and PEG-QC algorithms

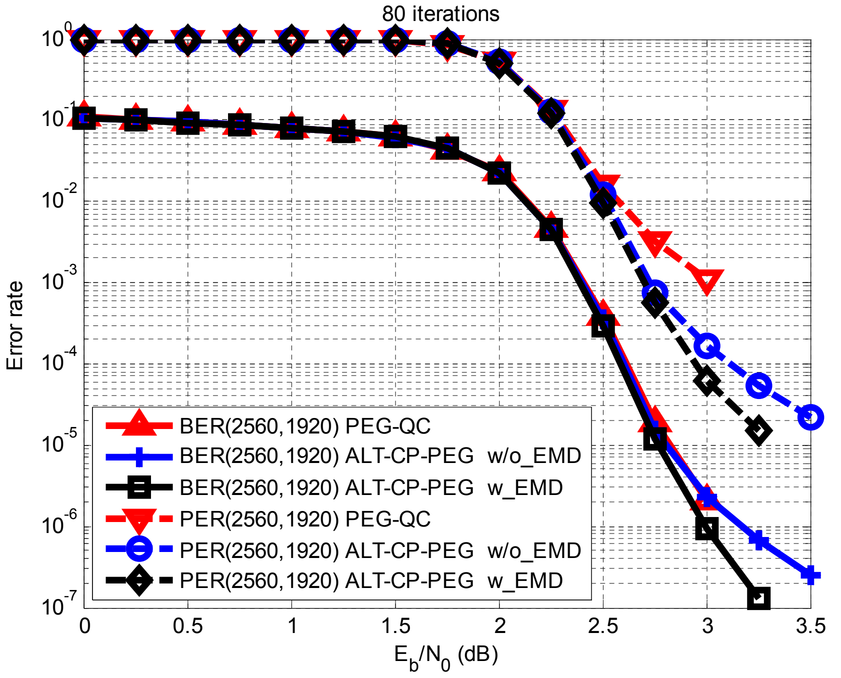


Figure 4.8: Performance comparison of the irregular codes with rate 3/4 constructed by the proposed and PEG-QC algorithms

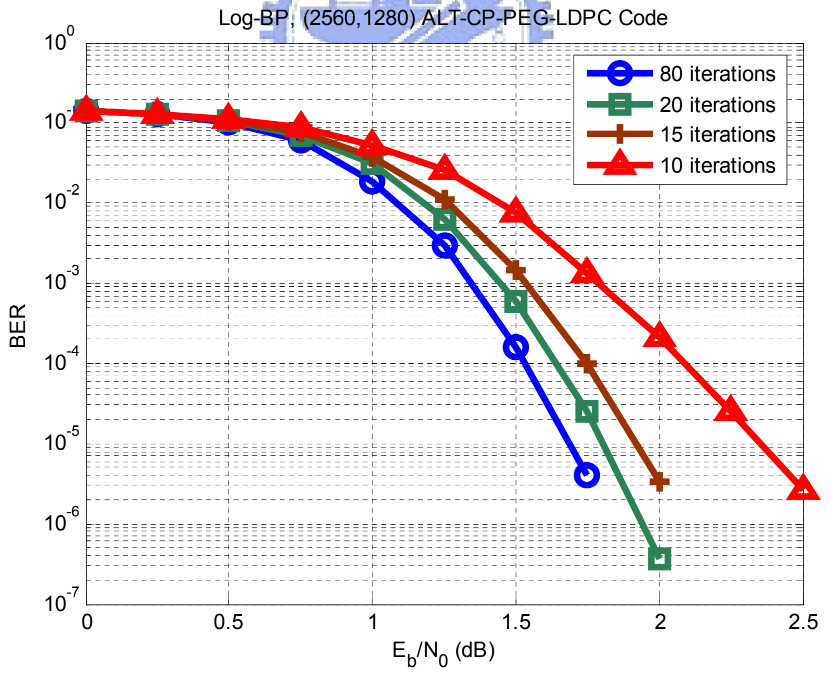


Figure 4.9: Iterative decoding with various maximum number of iteration for (2560, 1280) ALT-CP-PEG LDPC code with rate 1/2

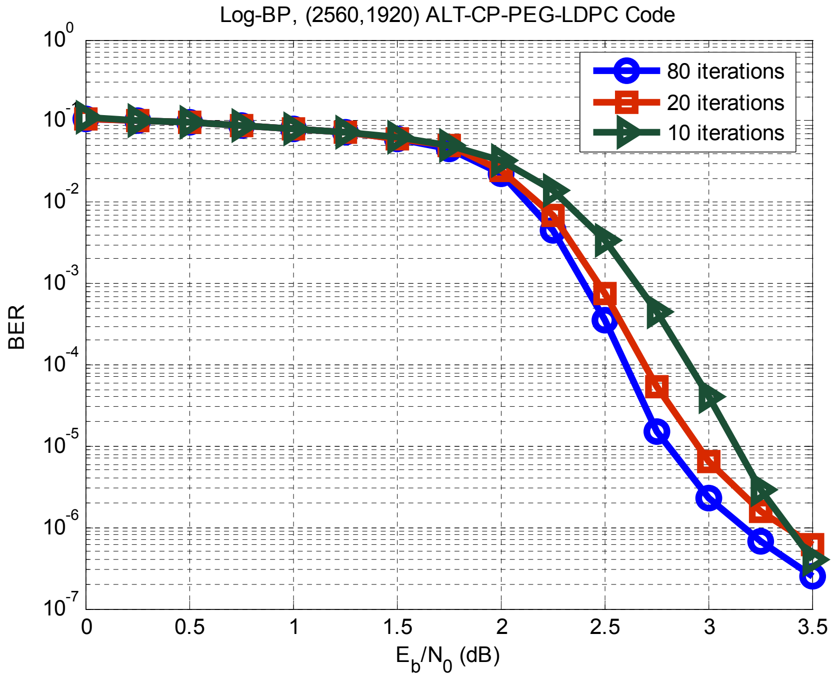


Figure 4.10: Iterative decoding with various maximum number of iteration for (2560, 1920) ALT-CP-PEG LDPC code with rate 3/4

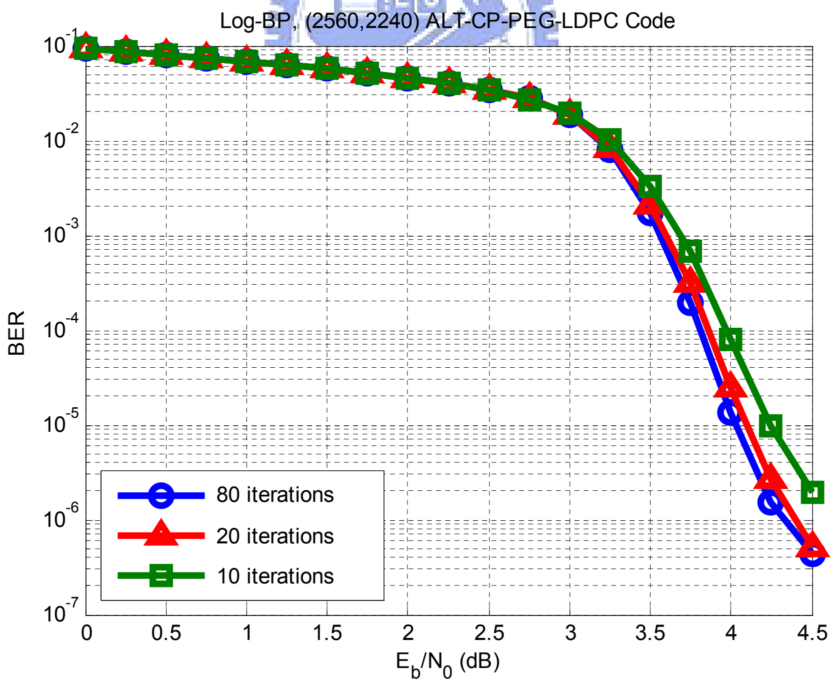


Figure 4.11: Iterative decoding with various maximum number of iteration for (2560, 2240) ALT-CP-PEG LDPC code with rate 7/8

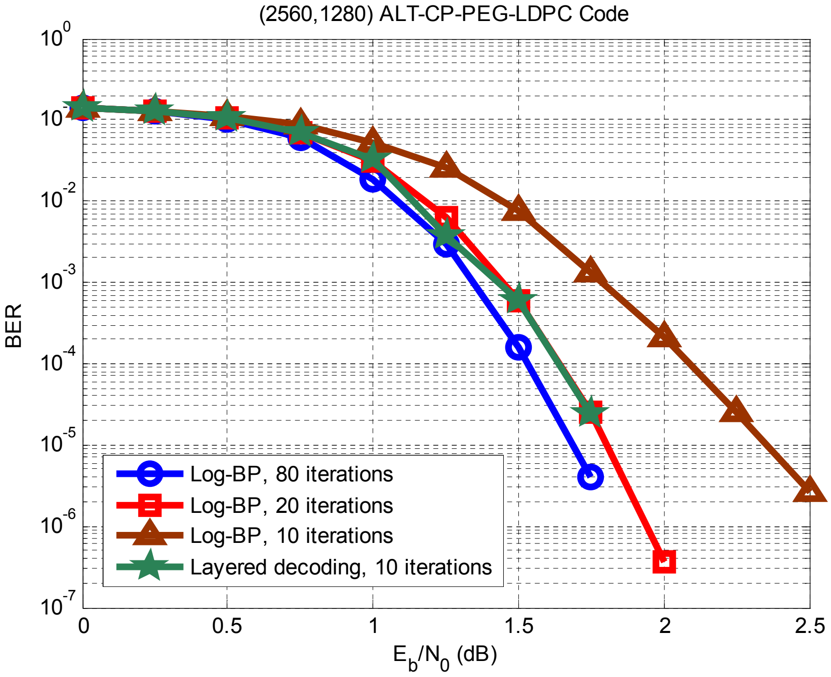


Figure 4.12: Performance of (2560, 1280) ALT-CP-PEG LDPC code with rate 1/2 by using layered decoding algorithm

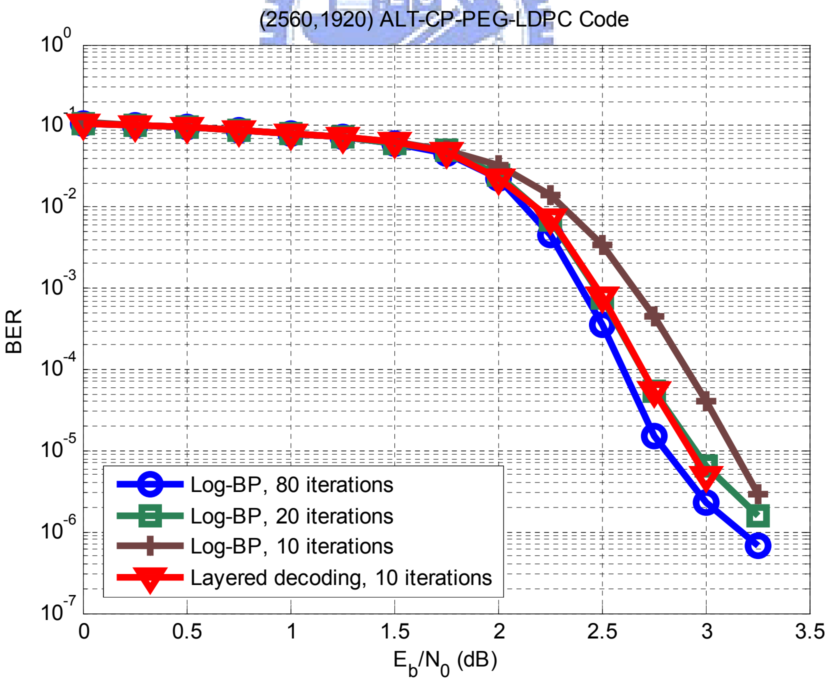


Figure 4.13: Performance of (2560, 1920) ALT-CP-PEG LDPC code with rate 3/4 by using layered decoding algorithm

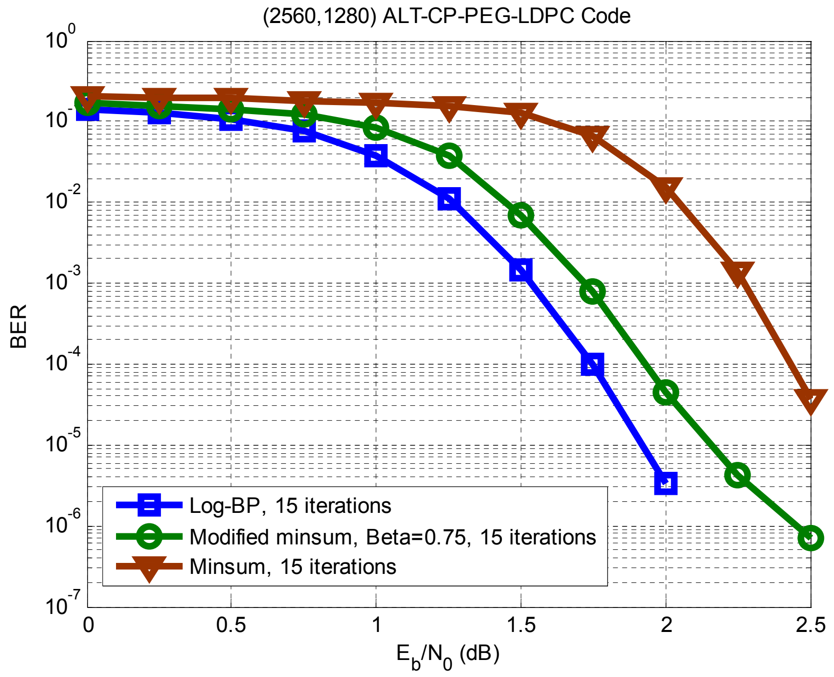


Figure 4.14: Performance of (2560, 1280) ALT-CP-PEG LDPC code with rate 1/2 by using min-sum and modified min-sum algorithms

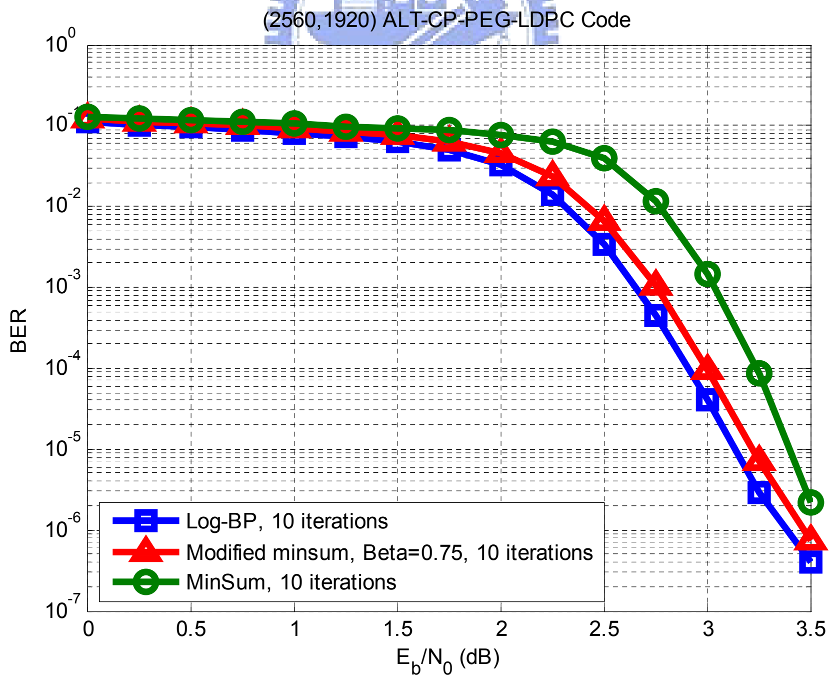


Figure 4.15: Performance of (2560, 1920) ALT-CP-PEG LDPC code with rate 3/4 by using min-sum and modified min-sum algorithms

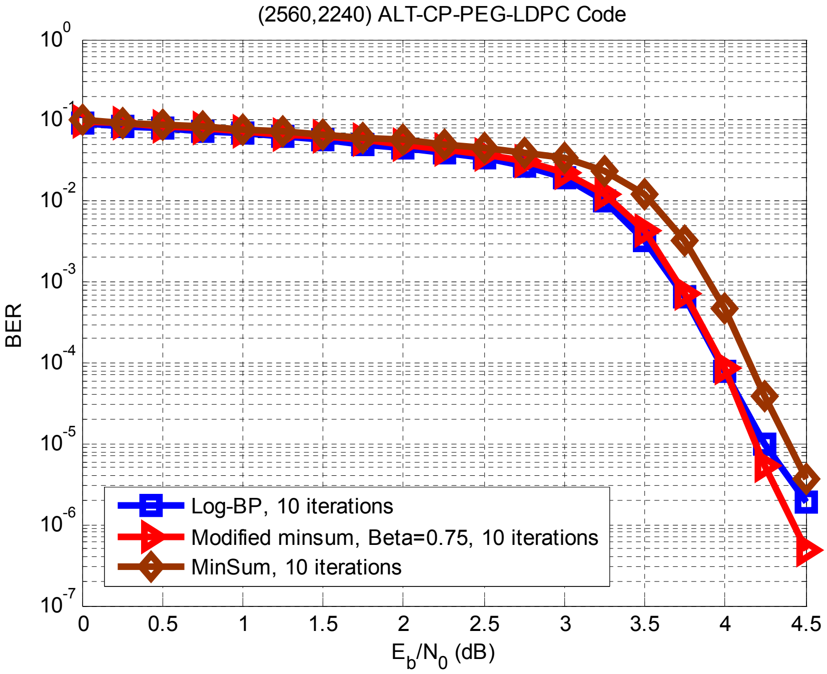


Figure 4.16: Performance of (2560, 2240) ALT-CP-PEG LDPC code with rate 7/8 by using min-sum and modified min-sum algorithms

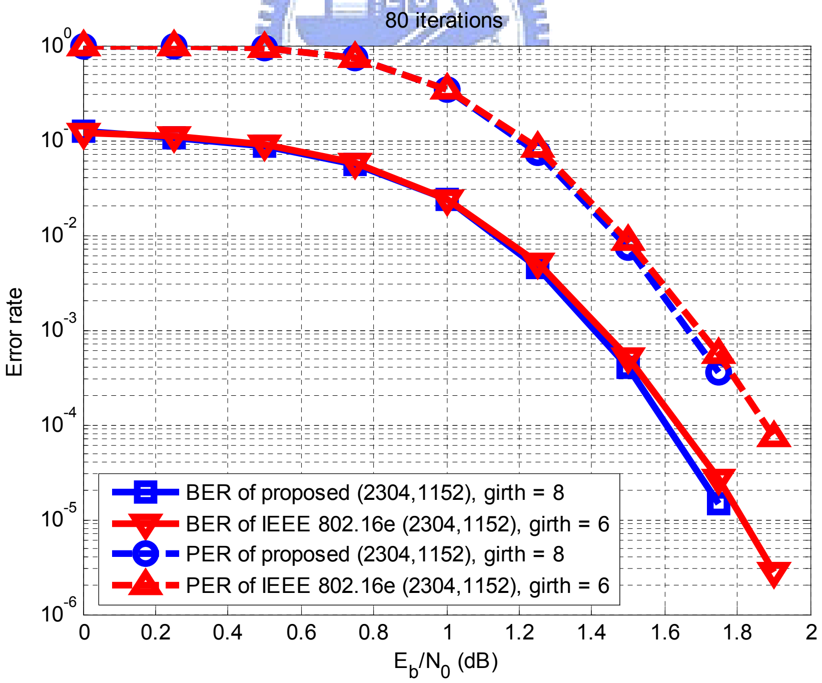


Figure 4.17: Performance comparison of the irregular codes with rate 1/2 constructed by the proposed algorithm and of IEEE 802.16e

Chapter 5

Conclusion

In this thesis, we propose a general method, called CP-PEG algorithm, to construct hardware-oriented LDPC codes for reducing VLSI design complexity. As compared to other algebraic methods, our algorithm is practical due to code parameters such as rate, block length, and degree distribution more flexible. In order to reduce encoding complexity, the parity check matrices in ALT form are also presented. Moreover, we can combine EMD criterion into CP-PEG or ALT-CP-PEG algorithm to enhance the error floor performance. With our approach, the resulting LDPC codes don't suffer error floor even though at BER = 10^{-7}.

Simulation results confirm that proposed algorithms are much better than traditional PEG-QC algorithm in terms of BER or PER. Comparing with the LDPC code adopted in IEEE 802.16e standard, our code performance outperforms that of IEEE 802.16e LDPC codes. Finally, for the convergence rate consideration, our structured codes are much suitable for layered decoding algorithm which can provide around two times faster decoding convergence. Because of the abovementioned advantages, the proposed CP-PEG algorithm can be a good candidate for designing practical LDPC codes.

Bibliography

[1] R. G. Gallager, *Low-Density Parity-Check Codes.* Cambridge, MA: MIT Press, 1963.

[2] V. Zyablov and M. Pinsker, "Estimation of the error-correction complexity of Gallager low-density codes," *Probl. Pered. Inform.*, vol. 11, pp. 23–26, Jan. 1975.

[3] G. A. Margulis, "Explicit construction of graphs without short cycles and low density codes," *Combinatorica*, vol. 2, no. 1, pp. 71–78, 1982.

[4] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inf. Theory*, vol. IT-27, no. 5, pp. 533–547, Sep. 1981.

[5] M. Sipser and D. A. Spielman, "Expander codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 1710–1722, Nov. 1996.

[6] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," *Electron. Lett.*, vol. 32, pp. 1645–1646, Aug. 1996.

[7] D. J. C. MacKay, "Good error correcting codes based on very sparse matrices," *IEEE Trans. Inf. Theory*, vol. 45, no. 2, pp. 399–431, Mar. 1999.

[8] S.-Y. Chung, J. G. D. Forney, T. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit," *IEEE Commun. Lett.*, vol. 5, no. 2, pp. 58–60, Feb. 2001.

[9] X.-Y. Hu, E. Eleftheriou, and D.-M. Arnold, "Progressive edge-growth Tanner graphs," in *Proc. IEEE Global Telecommunications Conf. (GLOBECOM)*, San Antonio, TX, Nov. 2001, pp. 995–1001.

[10] ——, “Regular and irregular progressive edge-growth Tanner graphs,” *IEEE Trans. Inf. Theory*, vol. 51, no. 1, pp. 386–398, Jan. 2005.

[11] *Air interface for fixed and mobile broadband wireless access systems*, IEEE Std. P802.16e/D12 Draft, Oct. 2005.

[12] *IEEE 802.11n Wireless LANsWWiSE Proposal: High Throughput extension to the 802.11 Standard*, IEEE Std. 11-04-0886-00-000n.

[13] Z. Li and B. V. K. V. Kumar, “A class of good quasi-cyclic low-density parity check codes based on progressive edge growth graph,” in *Proc. 38th Asilomar Conf. Signals, Syst. Comput.*, 2004, pp. 1990–1994.

[14] M. C. Davey and D. J. C. MacKay, “Low density parity check codes over $GF(q)$,” *IEEE Commun. Lett.*, vol. 2, pp. 165–167, June 1998.

[15] L. Ping, W. K. Leung, and N. Phamdo, “Low density parity check codes with semi-random parity check matrix,” *Electron. Lett.*, vol. 35, no. 1, pp. 38–39, Jan. 1999.

[16] T. Richardson and R. Urbanke, “Efficient encoding of low-density parity-check codes,” *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 638–656, Feb. 2001.

[17] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, “Factor graphs and the sum-product algorithm,” *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.

[18] J. F. Fan, *Constrained Coding and Soft Iterative Decoding.* Kluwer Academic Publishers, 2001.

[19] Y. C. Liao, C. C. Lin, C. W. Liu, and H. C. Chang, “A dynamic normalization technique for decoding LDPC codes,” in *IEEE Workshop Signal Processing Syst.*, Nov. 2005, pp. 768–772.

[20] A. J. Blanksby and C. J. Howland, “A 690-mW 1-Gb/s 1024-b, rate-1/2 low-density parity-check code decoder,” *IEEE Journal of Solid-State Circuits*, vol. 37, no. 3, pp. 404–412, Mar. 2002.

[21] E. Yeo, P. Pakzad, B. Nikolic, and V. Anantharam, "VLSI architectures for iterative decoders in magnetic recording channels," *IEEE Trans. on Magnetics*, vol. 37, no. 2, pp. 748–755, Mar. 2001.

[22] M. M. Mansour and N. R. Shanbhag, "High-throughput LDPC decoders," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 11, no. 6, pp. 976–996, Dec. 2003.

[23] D. E. Hocevar, "A reduced complexity decoder architecture via layered decoding of LDPC codes," in *Proc. IEEE Workshop on Signal Processing Systems (SiPS'04)*, Oct. 2004, pp. 107–112.

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

[25] J. Xu, L. Chen, I. Djurdjevic, S. Lin, and K. Abdel-Ghaffar, "Construction of regular and irregular LDPC codes: Geometry decomposition and masking," *IEEE Trans. Inf. Theory*, vol. 53, no. 1, pp. 121–134, Jan. 2007.

[26] S.-Y. Chung, T. J. Richardson, and R. Urbanke, "Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 657–670, Feb. 2001.

[27] Y. Mao and A. H. Banihashemi, "A heuristic search for good low-density parity-check codes at short block lengths," in *Proc. IEEE Int. Conf. Commun.*, Helsinki, Finland, June 2001.

[28] T. Tian, C. Jones, J. D. Villasenor, and R. D. Wesel, "Construction of irregular LDPC codes with low error floors," in *Proc. IEEE Int. Conf. Communications*, May 2003, pp. 3125–3129.

[29] C. Di, D. Proietti, E. Telatar, T. J. Richardson, and R. Urbanke, "Finite-length analysis of low-density parity-check codes on the binary erasure channel," *IEEE Trans. Inf. Theory*, vol. 48, no. 6, pp. 1570–1579, June 2002.

[30] H. Zhong and T. Zhang, "Block-LDPC: A practical LDPC coding system design approach," *IEEE Trans. Circuits Syst. I*, vol. 52, no. 4, pp. 766–775, Apr. 2005.

[31] H. Xiao and A. H. Banihashemi, “Improved progressive-edge-growth (PEG) construction of irregular LDPC codes,” *IEEE Commun. Lett.*, vol. 8, no. 12, pp. 715–717, Dec. 2004.

[32] S. H. Kim, J. S. Kim, D. S. Kim, and H. Y. Song, “LDPC code construction with low error floor based on the IPEG algorithm,” *IEEE Commun. Lett.*, vol. 11, no. 7, pp. 607–609, July 2007.

作者簡歷

姓名：林義凱

出生地：台灣省高雄市

出生日期：1983年10月5日

學歷：
1989.9 ~ 1995.6　高雄縣立林園國民小學
1995.9 ~ 1998.6　高雄縣立林園國民中學
1998.9 ~ 2001.6　高雄市立小港高級中學
2001.9 ~ 2005.6　國立中央大學　電機工程學系　學士
2005.9 ~ 2007.11　國立交通大學　電子研究所　系統組　碩士