JOINT DECODING OF CONTENT-REPLICATED CODES ON AWGNC FOR FLASH MEMORY

A Thesis

by

HUAN CHANG

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Anxiao Jiang
Committee Members,	Tie Liu
	Eun Jung Kim
Head of Department,	Dilma Da Silva

December 2015

Major Subject: Computer Engineering

ABSTRACT

Continuously increasing capacity and scales have made flash memory an affordable product for both consumer electronic and system storage. However, as flash density increases, flash memory becomes more subject to noise which leads to the degrading of data reliability and endurance. Error-correcting codes and memory scrubbing are two approaches to handle data reliability issues. Specifically, in flash memory, the "out-of-place" updating mechanism leads to the existence of content-replicated error-correcting codes (ECC) that contain the same data. The thesis proposes a joint-decoding scheme using those content-replicated ECCs to further enhance the reliability of data in flash memory. Three different categories of content-replicated ECCs are explored and analyzed. The density evolution analysis results show that using content-replicated ECCs with the corresponding joint-decoding algorithms can effectively improve the error-correcting performance. Additionally, it is shown that increasing the diversity of content-replicated ECCs with some limits may further extend the error-correcting ability.

ACKNOWLEDGEMENTS

Foremost, I would like to express my sincere gratitude to my advisor Prof. Anxiao Jiang for his continuous support of my Master study and research, for his patience, enthusiasm, and immense knowledge. His guidance inspires me during the process of research and writing of this thesis. I could not imagine having a better advisor and mentor than Professor Jiang for my Master study.

I also wish to thank my committee members who were more than generous with their expertise and precious time. Thank you Dr. Tie Liu and Dr. Eun Jung Kim for agreeing to serve as my committee, for the many hours of reflecting, reading, encouraging, and most of all patience throughout the entire process.

I would like to acknowledge and thank Computer Science and Engineering Department for allowing me to conduct my research and providing requested assistance. Special thanks go to the staffs and human resources members for their continued support.

Finally I would like to thank the members of Information Innovation Lab. Their enthusiasm and willingness to provide feedback made the completion of this research an enjoyable experience.

DEDICATION

I dedicate my thesis work to my family and my friends. A special feeling of gratitude to my husband, Fei Jia whose words of encouragement and push for tenacity ring in my ears everyday. My parents Dongmei and Hongyan are there as always to support me.

I also dedicate this thesis to my many friends who have supported me throughout the process. I will always appreciate what they have done during the process, especially the lab members, for the many hours of proofreading and helping me develop my technology skills.

Finally, I give special thanks to my dear roommate Zhuoying Wang and friend Zi Yin for being there for me throughout the entire master program. Both of you have been my best cheerleaders.

NOMENCLATURE

ECC	Error-correcting Code
P/E cycle	Program-erase Cycle
LDPC	Low-density Parity-check Code
AWGNC	Additive White Gaussian Noise Channel
BER	Bit Error Rate
BP	Brief Propagation
LLR	Logarithmic Likelihood Ratio

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

During the past decade, the capacity of flash memory has been impressively improved, which makes flash memory an economic product for a wide range of applications, from consumer electronics to modern storage systems. To customers, the reliability of data stored in flash memory is critical. However, along with the continuously increasing density and capacity, flash memory cells become more fragile to noise, which leads to the declining reliability of data. Even though flash memory is a relatively mature product, it still has four main limitations: block erasure, memory *"wear-out"*, *read disturb* and X-ray effects [2] [5] [6]. Among those four limitations, memory *"wear-out"*, *read disturb* are directly related to the reliability of data.

In flash memory, the "wear-out" issue stands for that flash memory can only tolerate for a finite number of program and erase cycles (P/E cycles) before failing. For example, the commercial single-level cells (SLC) NAND is guaranteed to withstand around 10^5 P/E cycles before the rate of decoding failings reaches a certain number, which may cause the deterioration of the storage integrity [7]. However, repeated wears, which are caused by reading and writing data from/to flash memory, and other external factors (e.g. temperature) degrade the accuracy of flash cells' contents. Even though MLC NAND increases the capacity of flash memory by storing two or more bits in the same physical location, its maximum endurance (measured in the number of P/E cycles) is significantly lower than SLC NAND. Specifically, the P/E cycles of MLC has dropped from 10 K for 5x-nm (represents 50- to 59-nm) to current 3K for 2x-nm (represents 20- to 29-nm) [8].

The content of a flash cell, which is represented as a flash cell level, is determined by its voltage and the predefined voltage thresholds which divides different cell levels.

For example, SLC NAND has two states: 0 and 1. Given the voltage threshold V_t of a flash cell, then the bit value is 1 if its current voltage is higher than V_t, otherwise the bit value is 0. The voltage threshold V_t is set according to a referenced voltage distributions of that SLC cell. Reading erros occur when 1 is read as 0 or reversely 0 is read as 1. If voltage of that cell shifts a lot from its predefined voltage distribution, the probability of reading errors will rise.

In one P/E cycle of flash memory, errors may be introduced in different stages. To start a P/E cycle, the flash block is firstly erased and then programmed in unit of pages. In the programming process, errors may occur when the flash block hasn't been reset to the initial voltage before being programmed or a programmed cell is disturbed by the programming process of neighboring flash cells. After being programmed, the flash block cannot be reprogrammed until be erased again. Between two erasures, flash cells may be accessed and read multiple times. In a read operation to a flash cell, a read *reference voltage* is applied to its transistor. At the same time, the transistors of other unread cells connected to the same *bitline* are powered with a *pass-through* voltage, which is a read reference voltage higher than any stored threshold voltage. As a result, it is guaranteed that only the target cell is read. Though those unread cells connected to the same *bitline* are not being read, the high *pass-through* voltage induces electric tunneling that can shift their threshold voltages to higher values, thereby disturbing the contents of those unread cells. Retention errors is another kind of errors that may be introduced before the next erasure. The charges in the floating gate of a flash cell is not constant but gradually lost through leaking current. As the voltage declines over time, the cell content changes.

Considering the limitations of flash memory mentioned above, improving the data reliability of flash memory is challenging, which is related to both the producing process and the reading/writing mechanisms. In the process of reading and writing

data, the encoding and the decoding mechanisms are very important for keeping the data stored in flash memory to be reliable. So, in this research, we mainly focus on exploring and analysis the encoding and decoding algorithms.

1.1 Problem Statement

For keeping the data stored in flash memory to be accurate and reliable, error-correction codes (ECC) are commonly used. The basic idea behind ECC is information redundancy: transfer more information than required to increase the distances between different valid codes and finally support detection and correction of errors. Along with the constant growth in the capacity and density of flash memory, the raw bit error rate increases, including read disturb errors, retention errors and other kinds of bit errors. Furthermore, the increasing bit error rate will lead to a declining rate of successful decoding processes. To deal with the higher raw bit error rate, multiple approaches have been published on using strong ECC, such as BCH and LDPC, to enhance the error-correcting performance. However, the error-correcting ability of single ECC is limited by Shannon Capacity. How to keep the reliability of data in flash memory on channels with a high bit error rate becomes a critical problem.

Beside ECC, *memory scrubbing* is another error-correcting technique. In the *memory scrubbing* scheme, a background process periodically checks the date integrity of memory and corrects errors or inconsistent data by using other copies of data. However, in flash memory, blocks are required to be erased before updating any cell in that block, which leads to high time complexity for "in-place" updating processes. Thus, a "out-of-place" updating mechanism is used, in which an updated codeword is always stored in another physical address and the original memory cells are marked as invalid. On one hand, the "out-of-place" updating mechanism opti-

mizes the time complexity of writing processes. On the other side, it wastes memory space, as the original memory cells are marked as "invalid" after the updating process, which contains a expired but useful ECC. Besides "out-of-place" updating mechanism, there are other factors which may also lead this kind of "invalid" cells which contain content-replicated data, e.g. garbage collection.

With the application of ECC, raw bit errors usually can be corrected since the raw bit error rate in flash memory is relatively low within thousands P/E cycles. However, there do exist a small number of situations in which error-correcting algorithm fails. The more P/E cycles has been used, the higher bit error rate as well as the larger probability of a decoding failure. The error-correcting performance at those rare cases is critical for a storage system. As discussed above, the existence of content-replicated ECCs stored in "invalid" flash memory cells provides a promising way to keep the reliability of data under those rare cases. Specifically, content-replicated codes can be decoded together to increasing the possibility of successful decoding.

Practically, for each storage unit, the control system of flash memory can keep an list of addresses of other "invalid" cells which carry the same information. After writing data into a new memory location, the original memory cell becomes an "invalid" content-replicated cell. The control system can add the location of this "invalid" cell into the address list of the new memory location. Once the decoding process of the valid flash cell fails, the control system can retrieve the address of its content-replicated codes from that list. Then, those content-replicated codes and the valid code can be decoded together. Since the decoding process fails rarely, the overal time complexity of decoding process will not increase a lot.

The main goal of this thesis is designing a joint decoding scheme which uses the existence of content-replicated ECCs (ECCs contains the same information) introduced by the unique updating mechanisms of flash memory. Due to its excellent

performance, low-density parity-check(LDPC) is applied as ECC in this work. Current researches on LDPC for flash memory focus on optimizing the error-correcting performance of a single LDPC code and memory response delay. There's a lack of study on joint decoding algorithms for multiple content-replicated codes. In [3], Qing presents a joint decoding algorithm for content-replicated codes on Binary Erasure Channel (BEC). In this thesis, we will discuss and explore the joint decoding algorithms for content-replicated LDPC codes on AWGN channels.

1.2 Thesis Structure

The rest of the thesis is organized as follows. Section 2 presents an brief overview of current approaches on data reliability design of flash memory, including error-correcting codes and *memory scrubbing*.

Based on the previous investigation, Section 3 illustrates a general definition of content-replicated LPDC codes. Three different categories of the content-replicated LDPC codes are listed in separate subsections. Each subsection also includes a construction method for combined LDPC codes and a joint-decoding algorithm.

Section 4 demonstrates an error-correcting performance analysis for the joint-decoding algorithms. Particularly, this Section firstly compares two density evolution mechanisms: Gaussian Approximation and Ergodicity theory based approximation and analyzes their adaptability to the performance analysis of the combined LDPC codes. Then, density evolution analysis is conducted exclusively for each category of the content-replicated LDPC codes. Finally, an experimental result of density evolution analysis is presented, which indicates that the combined content-replicated codes with a joint decoder can tolerant a higher error rate compared to single LDPC codes. This work has been published in 53rd Annual Allerton Conference on Communication, Control, and Computing [4].

On the practical side, this thesis uses the unique updating mechanisms in flash memory and provides a promising error-correcting scheme. On the theoretical side, this thesis shows that utilizing the content-replicated data stored in flash memory and the joint decoding algorithms can effectively improve the error-correcting performance. Meanwhile, it is indicated that increasing the diversity of content-replicated ECCs may increase the reliability of the combined code even if there exist constraints in the joint decoding algorithms.

2. LITERATURE REVIEW

Together with its extraordinary capacity, flash memory also experiences reliability issues introduced by *read disturb*, data retention and other failure factors. From the view of the flash memory controller, Yu *et al.* [1] classified flash memory errors in a P/E cycle into four different categories: erase errors, programming errors, retention errors and read disturb error. In Figure 2.1, Yu illustrates and compares the error rates of different kinds of errors. The result shows that the error rate of retention errors is significantly higher than other types of errors. The error rate of retention errors is dependent on the retention test time. The longer before a retention test, the more likely to lose more electrons which leads to higher probability of retention errors.

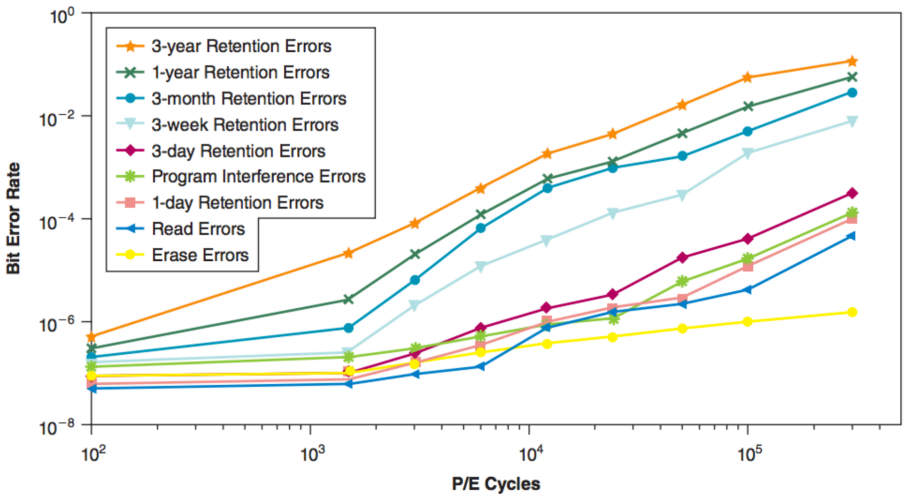


Figure 2.1: Error rates of different types of errors in Flash Memory [1].

2.1 Background on Data Reliability Design in Flash Memory

Currently, there are mainly two approaches for solving data reliability issues in flash memory: ECC and memory scrubbing. Both of them are briefly introduced and analyzed in the following sections.

2.1.1 Error-correcting Codes in Flash Memory

ECC is used to improve the reliability of flash memory by reducing raw bit error rate (BER) [6]. Figure 2.2 demonstrates the raw BER of a MLC flash memory with and without using ECC. For a SLC NAND flash memory, without using ECC, the bit error rate is around 10^{-9} after 10^5 P/E cycles. After applying a 4-bit ECC, in which up to 4 errors can be corrected for every 512 bytes, the bit error rate is reduced to 10^{-20}.

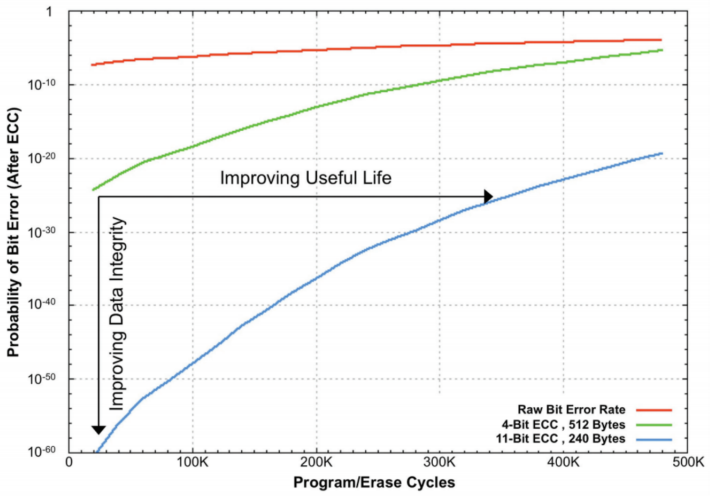


Figure 2.2: ECC improvement of raw BER as a function of P/E cycles [2]

The basic idea behind ECC is transferring more information than required to increase the distance between valid codewords. Classified by the encoding principles,

there are two basic types of ECCs: block codes and convolution codes [9]. Hamming codes and BCH codes are currently widely used in commercial flash memory. For example, Micron NAND Flash memory devices use cyclic and Hamming codes. Hamming codes are defined as $(2^n - 1, 2^n - n - 1)$, where n represents the number of over head bits, $2^n - 1$ represents the code block size and $(2^n - n - 1)$ represents the number of information bits. Common used Hamming codes are (7, 4), (15,11), and (31, 26). They have the same Hamming distance and are able to correct and detect single bit error. Wei *et al.* [10] show a high-throughput and low-power BCH (4148,4096) scheme for MLC NAND flash memories.

However, along with the continuously increasing capacity of NAND flash memory and the usage of MLC technique, the data reliability and endurance problems become more challenging. It leads to a need of strong ECC with higher error-correcting performance to handle the increasing BER.

2.1.2 Memory Scrubbing

Memory scrubbing is a technique that periodically checks the date integrity of memory and corrects errors or inconsistent data by using other copies of data. Gonzalez *et al.* [10] present a flash memory data correction method based on data scrubbing. By systematically scrubbing the data stored in different memory cells, they are able to reduce corruption of stored data. In DRAM memory, scrubbing mechanism is applied together with single-error correct double-error detect (SECDED) to address uncorrectable errors in a single ECC. To prevent the occurrence of second error, the memory periodically checks data correctness. If a decoding process fails, system will trigger the event that scrubs data from another location to correct error and finally write the recovered data back to its original address.

2.2 Low-density Parity-check Codes (LDPC)

Low-density parity-check (LDPC) codes are linear block codes using a very sparse parity check matrix. LDPC were originally introduced by Robert G. Glallager in 1962 [11]. The powerful capabilities of LDPC codes has led it to its usage in several standards [12]. In both historical and recent approaches, LDPC has demonstrated its outstanding error correction performance.

Symbol $\mathbf{m}_{K\times 1}$ represents the original message with length M. Let N to be the length of a (N, ω_c, ω_r) LDPC code $\mathbf{c}_{N\times 1}$ which is encoded from message $\mathbf{m}_{K\times 1}$. K is the length of redundant parts such that $M = N - K$. Following the definitions in [11], symbol $\mathbf{H}_{M\times N}$ represents the parity check matrix of the LDPC code and $\mathbf{G}_{N\times K}$ represents the corresponding generating matrix, such that $\mathbf{H} \times \mathbf{G} = \mathbf{0}$. The column weight of the LDPC code is ω_c, which means each column contains a fix number, ω_c, of 1's. Similarly, the row weight is ω_r representing the number of 1s in each row. ω_c and ω_r is significantly smaller than N.

$$\mathbf{c}_{N\times 1} = \mathbf{G}_{N\times K} \times \mathbf{m}_{K\times 1} \tag{2.1}$$

An LDPC code is specified and defined by a parity check matrix H. Since the introduction of Gallager LDPC Codes in 1962 [11], multiple approaches have been published on the construction of LDPC codes, mainly divided into two areas: regular codes and irregular codes. In regular codes, column weight and row weight are constant. While, in irregular codes, $\lambda(x)$ defines the probability distribution of column weights and respectively $\rho(x)$ defines the probability distribution of row weights. In Gallager Codes, a parity matrix H with column weight ω_c and row weight ω_r is constructed by firstly generating a $\frac{K}{\omega_c} \times N$ sub-matrix H_1. In H_1, there is only one 1 in

each column and 1's in columns $[(i-1)\omega_r + 1, i\omega_r]$ for i^{th} row.

$$H_1 = \begin{bmatrix} \mathbf{1}^{1\times\omega_r} & \mathbf{0}^{1\times\omega_r} & \dots & \mathbf{0}^{1\times\omega_r} \\ \mathbf{0}^{1\times\omega_r} & \mathbf{1}^{1\times\omega_r} & \dots & \mathbf{0}^{1\times\omega_r} \\ \dots & \dots & \ddots & \dots \\ \mathbf{0}^{1\times\omega_r} & \mathbf{0}^{1\times\omega_r} & \dots & \mathbf{1}^{1\times\omega_r} \end{bmatrix}_{\frac{K}{\omega_c}\times N} \tag{2.2}$$

Afterwards, the other $\omega_c - 1$ sub-matrices are generated by randomly permuting H_1. The final parity check matrix H is constructed as:

$$H_{N\times K} = \begin{bmatrix} H_1 \\ H_2 \\ \dots \\ H_{\omega_c} \end{bmatrix} \tag{2.3}$$

The encoding process is conducted by firstly using Gaussian Elimination to convert H into the following form:

$$\widetilde{\mathbf{H}} = \begin{bmatrix} \mathbf{I}_{(N-K)\times(N-K)} & \mathbf{A_2}_{(N-K)\times K} \end{bmatrix} \tag{2.4}$$

Having calculated $\widetilde{\mathbf{H}}$, the generating matrix G is defined as

$$\mathbf{G}_{N\times K} = \begin{bmatrix} \mathbf{I}_{K\times K} \\ \mathbf{A_2}_{M\times K} \end{bmatrix}_{N\times K} \tag{2.5}$$

After Gaussian Elimination, the real rate of the (N, ω_c, ω_r)-regular LDPC code R is always higher than designed rate: $R = \frac{K}{N} > \frac{N-M}{M}$. Associated with the parity check matrix H, tanner graph is applied to demonstrate the iterating decoding process by

R. Michael Tanner [13]. Tanner graph is a bipartite graph, which contains two kinds of nodes: variable nodes and check nodes.

The decoding process of a LDPC code uses iterative decoding algorithm, which is mainly classified into hard decision decoding and soft decision decoding.

Hard decision decoding: In hard decision schema, a check node detects bit errors by checking whether the parity-check equation of the incoming messages is satisfied or not. Then, the result message is sent back to variable nodes. The decoding process will terminate if all parity check equations become satisfied or reach the maximum number of iteration. Bit flipping algorithm is an example of hard decision based decoding algorithms. The details of bit flipping algorithm is illustrated below.

1. Initialization. Variable nodes are initialized with the original received LDPC code.

2. All variable nodes send a bit message to their connected check nodes. In the initial iteration, the message from a variable node v_i to a check node c_j is the same as its initial bit message. In other iterations, the message is calculated as the majority of initial bit value and all incoming message from v_i's connected check nodes excluding c_j.

3. Every check nodes send back a message to its connected variable nodes. The message from a check node c_i to a variable node v_j is calculated as the binary sum of all received incoming message at c_i from its connected variable nodes excluding v_j.

4. Repeat step 2-3 until satisfying all parity check equations or reaching a certain number of iterations.

Soft decision decoding: Soft decision decoding improves performance of LDPC by utilizing sum-product decoding algorithms [14]. In soft decision, messages between variable nodes and check nodes are obtained as the a-priori probability of received bits. For example, in sum product message passing algorithm, likelihood ratio is introduced as message. Figure 2.3 illustrates the message updating mechanism at a variable node and a check node.

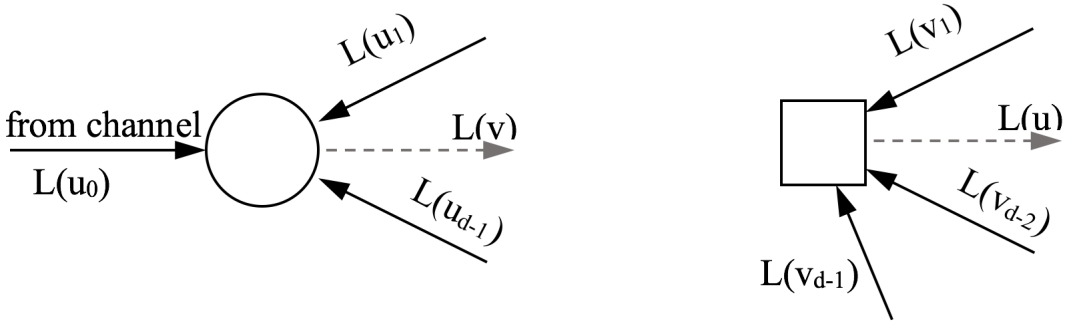


Figure 2.3: Diagram of sum-product message passing schema in variable nodes and check nodes.

The basic BP algorithm consists of the following main steps:

1. Initialization. The message at a variable node is initialized as the LLR of original bit value, which is denoted as u_0. Given the original code is $\hat{y} = y_1, y_2, \ldots, y_n$, the LLR message sequence u_0 is calculated by Equation 2.6

$$u_{0,i} = log\frac{Pr[c_i = 0|y_i]}{Pr[c_i = 1|y_i]} \tag{2.6}$$

2. Updating variable nodes. The LLR message from a variable node to a check node is updated using Equation 2.7. d_v is the degree of a variable node, which

is the same as ω_c.

$$v = u_0 + \sum_{i=1}^{d_v - 1} u_i \tag{2.7}$$

3. Updating check nodes. The LLR message from a check node to a variable node is updated by Equation 2.8. d_c is the degree of a check node, which is the same as ω_r.

$$u = 2tanh^{-1} \prod_{i=1}^{d_c - 1} tanh \frac{v_i}{2} \tag{2.8}$$

4. At one variable node v_i, calculate its current bit value x_i such that $x_i = 1$ if the probability of bit x_i to be 1 is greater than 0.5 and $x_i = 0$ for other cases. Respectively, calculated the bit value for $i = 1, 2, \ldots, n$. Finally, having generating $\hat{x} = x_1, x_2, \ldots, x_n$, terminate the iterative decoding process if

$$H \times \hat{x} = \mathbf{0}$$

or the iterator reaches a certain number. Otherwise, repeat step 2 and step 3.

Having demonstrated construction and decoding algorithms of LDPC codes, the following part discusses density evolution, which is an approximation technique used to analyze the error correction ability of LDPC codes. Under density evolution, it assumes that the length of LDPC codes is infinite and parity check matrix H is cycle-free, which means no cycles in its tanner graph. For general message passing decoding algorithms, there are three density evolution methods: quantization, Gaussian Approximation and population dynamics. In [15], Chung modified the original sum-decoding algorithm by using quantized input and output message in Equation 2.7 and Equation 2.8. Thus, 2.7 becomes $\mathcal{Q}(v) = \sum_{i=0}^{d_v - 1} \mathcal{Q}(u_i)$ and 2.8 becomes $\mathcal{Q}(u) = 2tanh^{-1} \prod_{i=1}^{d_c - 1} tanh \frac{\mathcal{Q}(v_i)}{2}$. $\mathcal{Q}(w)$ is the quantization operator. Then, the distribution evolution can be estimated by $p_v = p_{u_0} \bigotimes^{d_v - 1} p_u$, which can be done

efficiently using FFT. Similarly, the quantized message $\mathcal{Q}(u)$ can be calculated in time complexity $O(n^2)$.

Chung [16] proposed Gaussian Approximation based density evolution for both regular and irregular LDPC codes under Additive White Gaussian Noise(AWGN) Channels. In that approach, there's one important assumption called "symmetry condition" which is expressed as $f(x) = f(-x)e^x$. For AWGN channels, this assumption can be simplified to $\sigma^2 = 2m$, in which $sigma$ and m is the standard deviation and mean of the Gaussian distribution. The accuracy of Gaussian Approximation greatly relies to this assumption. The third density evolution algorithm is based on the usage of a large population of "samples". Fu [17] presents an ergodicity theory based density evolution. With the "cycle-free" assumption, the input messages at variable nodes are independent. Additionally, the updating mechanisms of variable nodes and check nodes preserve ergodicity. So, the iteration decoding process can be approximated by using a large number of samples.

The performance of LDPC relies on error location, soft decision accuracy and number of iterations. Typically, flash memory uses hard decision for a single cell voltage. Hard-decision sensing means memory only uses one quantization level between two adjacent storage states, e.g. 0 and 1. Respectively, soft-decision sensing is reading mechanism which uses more than one quantization levels between two adjacent states. Soft decision based decoding of LDPC can be obtained by reading with soft-decision sensing or transfer voltage into likelihood ratios (LLR). However, soft-decision sensing could increase response delay in flash memory. To solve this issue, Zhao [18] presents three approaches in optimizing reading delay in flash memory and presents the performance of hard decision LDPC and soft decision LDPC. The result of this work shows LDPC's strength in error correction and its better performance than BCH codes.

3. JOINT-DECODING DESIGN FOR CONTENT-REPLICATED CODES*

3.1 Problem Statement

This section discusses the joint-decoding scheme for two content-replicated LDPC codes, which can be easily extended to multiple content-replicated codes. Given an original message $\mathbf{m} = \{m_0, m_1, \ldots, m_{K-1}\}$ with length of K bits. Suppose there are two independent encoders: $Encoder_1$ and $Encoder_2$ with the same rate $R = \frac{K}{N}$. Let $\mathbf{x}_1 = \{x_{1,0}, x_{1,1}, \ldots, x_{1,N-1}\}$ and $\mathbf{x}_2 = \{x_{2,0}, x_{2,1}, \ldots, x_{2,N-1}\}$ to be the respective codewords generated by $Encoder_1$ and $Encoder_2$. Having $\mathcal{X}$ to be the alphabet sets of flash cell levels such that $x_{1,i} \in \mathcal{X}$ and $x_{2,i} \in \mathcal{X}$. Those ECCs like $\mathbf{x_0^{(1)}}$ and $\mathbf{x_0^{(2)}}$ are named as content-replicated ECCs which carry the same information. Let $\mathcal{P}$ and $\mathcal{Q}$ to be two independent Additive White Gaussian Noise (AWGN) channels. Figure 3.1 shows the model of joint decoding scheme for ECCs.

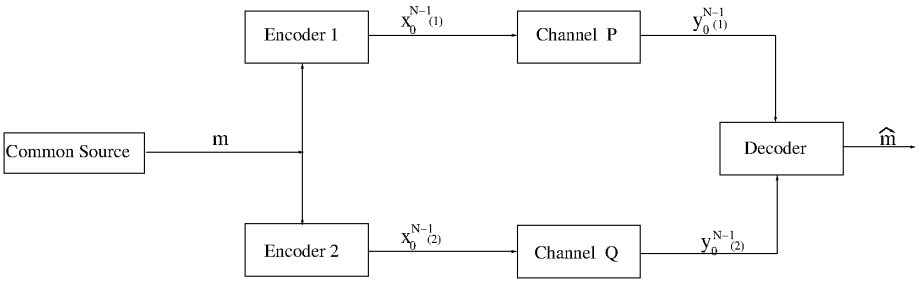


Figure 3.1: An illustration of the joint-decoding scheme for the content-replicated ECCs [3].

*Reprinted with permission from "Joint Decoding of Content-Replication Codes for Flash Memories " by Q. Li, H. Chang, A. Jiang and E. F. Haratsch, 53rd Annual Allerton Conference on Communication, Control, and Computing, Sept 29-Oct 2, 2015, Allerton Park and Retreat Center, Monticello, IL, USA, Copyright (2015) by IEEE.

The joint-decoding mechanism is applied under the cases that the decoding process of a single ECC fails. Here's a brief description of the joint-decoding problem for two content-replicated ECCs.

Definition 3.1.1. *Given two content-replicated ECCs* $\mathbf{x_1}$ *and* $\mathbf{x_2}$ *with a certain rate* R. $\mathbf{y_1}$ *and* $\mathbf{y_2}$ *are the received noisy codes through two AGWNC* $\mathcal{P}$ *and* $\mathcal{Q}$. *The target is designing a joint decoding function to reduce the probability of decoding failures.*

Specifically, for two content-replicated LDPC codes on AWGN channels $\mathcal{P} \sim \mathcal{N}(\mu, \sigma)$, $\mathcal{Q} \sim \mathcal{N}(\mu, \sigma)$, the target becomes designing a joint-sum-product decoding algorithm P_{JSP} to maximize the threshold σ^*, which denotes the upper bound of variance of AWGN channels such that ECC can be decoded correctly.

$$\sigma^* = sup\{\sigma : P_{JSP}(\sigma)_{N\to\inf} \to 0\}$$

According to [11], an LDPC code is defined by a parity-check matrix H. Suppose that $Encoder_1$ and $Encoder_2$ are two (N, ω_c, ω_r)-regular LDPC encoders with generating matrices G_1 and G_2. H_1 and H_2 are the corresponding parity check matrices.

In this section, we will discuss three different categories of content-replicated LPDC codes which are classified by the relationship between $\mathbf{x_1}$ and $\mathbf{x_2}$. Firstly, we considered the identical cases in which two content-replicated LDPC codes shares the same information bits and the same parity-check constraints.

Definition 3.1.2. *Identical content-replicated codes are ECCs which are encoded using the same encoder and the same original message. Having two LDPC encoders with* $H_1 = H_2$ *and* $G_1 = G_2$, *then* $\mathbf{x_1}$ *and* $\mathbf{x_2}$ *are identical content-replicated LDPC codes if* $\mathbf{x_1} = G_1 \times \mathbf{m}$ *and* $\mathbf{x_2} = G_2 \times \mathbf{m}$.

Identical content-replicated LDPC codes can reduce the noise variance by averag-

ing each bit of two LDPC, however, the parity-check equations stay the same as the single LDPC code. Introducing different LPDC codes can increase the diversity of parity-check equations, which may lead to higher probability of successful decoding. Thus, we further explore the joint-decoding method for content-replicated LDPC codes with different parity-check constraints.

Definition 3.1.3. *Content-replicated LDPC codes with different parity-check constraints are LDPC codes which are encoded using the same information but different encoders. Particularly, two LPDC codes* $\mathbf{x_1}$ *and* $\mathbf{x_2}$ *are different replicated only if* $\mathbf{x_1} = G_1 \times \mathbf{m}$, $\mathbf{x_2} = G_2 \times \mathbf{m}$ *and* $H_1 \neq H_2$.

Finally, we explore the joint-decoding algorithm for content-replicated codes with an intermediate parity-check matrix. The intermediate matrix defines a one-to-one mapping between information bits of two LDPC codes. By using this intermediate matrix, both of the two LDPC codes can be decoded individually. Besides, they can be decoded together and communicate through the intermediate matrix.

Definition 3.1.4. *Content-replicated LPDC codes with an intermediate parity-check matrix are content-replicated in the way that* $\mathbf{x_1} = G_1 \times \mathbf{m_1}$, $\mathbf{x_2} = G_2 \times \mathbf{m_2}$ *and* $H_1 \neq H_2$. $\mathbf{m_1}$ *and* $\mathbf{m_2}$ *are the information bits and parity-check bits of a code* $\mathbf{x_3}$ *which are encoded using original message* $\mathbf{m}$ *and* $Encoder_3$. $Encoder_3$ *is a LDPC encoder with rate* $1/2$, *generating matrix* G_3 *and parity-check matrix* H_3.

3.2 Identical Content-replicated LDPC Codes

As previous definition, two identical content-replicated LDPC codes $\mathbf{x_1}$ and $\mathbf{x_2}$ maintains that $\mathbf{x_1} = G_1 \times \mathbf{m}$, $\mathbf{x_2} = G_2 \times \mathbf{m}$ and $G_1 = G_2$. $\mathbf{x_1}$ and $\mathbf{x_2}$ are transferred through two AWGN channels $\mathcal{P}$ and $\mathcal{Q}$ such that $\mathcal{P}, \mathcal{Q} \sim \mathcal{N}(\mu, \sigma^2)$. The received noisy codes $\mathbf{y_1}$ and $\mathbf{y_2}$ can be decoded using the sum-product algorithm with $H_1 = H_2$

if the variance of AWGN channel noise σ stands within a certain range: $\sigma < \sigma^*_{SP}$. σ^*_{SP} represents the threshold of channel noise variance such that $\mathbf{y_i}$ can be decoded correctly with the parity-check matrix H_i for $i = 1, 2$. If $\sigma > \sigma^*_{SP}$, neither $\mathbf{y_1}$ nor $\mathbf{y_2}$ can be decoded errorless. Then, a combined codeword $\mathbf{y} = \{y_0, y_1, \ldots, y_{N-1}\}$ is obtained as follows:

$$y_i = \omega_1 * y_{1,i} + \omega_2 * y_{2,i} \tag{3.1}$$

in which $i = 0, 1, \ldots, N-1$ and $\omega_1 + \omega_2 = 1$. Suppose AWGN channel $\mathcal{P}$ is as same significant as channel $\mathcal{Q}$, it lead to that $\omega_1 = \omega_2 = \frac{1}{2}$.

Theorem 3.2.1. Given an LDPC code $\mathbf{x}_N$ and the corresponding parity-check matrix H. Let $\mathbf{y}_\mathcal{P}$ and $\mathbf{y}_\mathcal{Q}$ denote the received noisy codes transferred through two AWGN channels $\mathcal{P}$ and $\mathcal{Q}$. Then, for the combined LDPC code $\mathbf{y} = \omega_\mathcal{P}\mathbf{y}_\mathcal{P} + \omega_\mathcal{Q}\mathbf{y}_\mathcal{Q}$, the parity matrix is H.

Proof. From the statement, it is known that $H \times \mathbf{y}_\mathcal{P} = \mathbf{0}$ and $H \times \mathbf{y}_\mathcal{Q} = \mathbf{0}$, if there's no errors. Thus, $H \times \mathbf{y} = \omega_\mathcal{P} H \times \mathbf{y}_\mathcal{P} + \omega_\mathcal{Q} H \times \mathbf{y}_\mathcal{Q} = \mathbf{0}$. So, H is a parity-check matrix for LDPC code $\mathbf{y}$. □

Since $\mathbf{x_1}$ and $\mathbf{x_2}$ are encoded using the same generating matrix $G_1 = G_2$ and the same original message $\mathbf{m}$. Based on Theorem 3.2.1, the parity-check matrix for $\mathbf{y}$ is constructed as

$$H = H_1 = H_2. \tag{3.2}$$

The joint-sum-product decoding algorithm for identical content-replicated LDPC codes is presented as below.

1. Given two received identical content-replicated LDPC codes $\mathbf{y_1}$ and $\mathbf{y_2}$ and respective parity-check matrices H_1 and H_2. Construct a combined code and its parity-check matrix by Equation 3.1 and Equation 3.2.

2. Using logarithmic likelihood ratio(LLR) replacing probability, the initial message $\mu_{\mathbf{0}}$ at variable node V_i is calculated as $\mu_{0,i} = ln(\frac{Pr[V_i=0|y_i]}{Pr[V_i=1|y_i]})$ for $i = 0, 1, \ldots, N-1$.

3. Let $\mathbb{C}_i$ to be the set of check nodes which are connected to variable node V_i. Then, the message that V_i sends to C_j, which denotes a check node connected to V_i, at round l becomes:

$$m_{ij}^{l} = \mu_{0,i} + \sum_{j' \in \mathbb{C}_i \&\& j' \neq j} m_{j'i}^{l-1} \tag{3.3}$$

in which $l > 0$ and $m_{ij}^{0} = \mu_{0,i}$.

4. Let $\mathbb{V}_j$ to be the set of variable nodes which are connected to check node C_j. Then, the message that C_j sends to V_i at round l is obtained by:

$$m_{ji}^{l} = 2tanh^{-1} \prod_{i' \in \mathbb{V}_j \&\& i' \neq i} \frac{tanh(m_{i'j}^{l})}{2} \tag{3.4}$$

5. The joint sum-product decoding algorithm is executed for a maximum number of iterations or until the LLRs at variable nodes are closed to $\pm\infty$, whichever comes first.

3.3 Content-replicated LDPC Codes with Different Parity-check Constraints

With the insight of joint sum-product decoding algorithm for identical content-replicated LDPC codes, this section will explore joint decoding algorithm for content-replicated codes which are encoded using different encoders. As defined in Section 3.1, two LDPC codes are different content-replicated if $\mathbf{x}_1 = G_1 \times \mathbf{m}$, $\mathbf{x}_2 = G_2 \times \mathbf{m}$ and $G_1 \neq G_2$, $H_1 \neq H_2$.

Several approaches exists in constructing practical LDPC codes. To construct a LDPC code, a parse parity-check matrix is first constructed. Having a parity-check matrix H, the corresponding generating matrix G is generated by Gaussian Elimination or lower triangle modification [19]. Gaussian Elimination based encoding algorithm leads to a generating matrix G defined as Equation 2.5. Therefore, a code is obtained as $\mathbf{c} = G \times \mathbf{m} = \{\mathbf{m} \times \mathbf{I}, \mathbf{m} \times \mathbf{A_2}\}^T = \{\mathbf{m}, \mathbf{p}\}$. In [19], Richardson presents an efficient encoding algorithm, in which a LDPC code c is generated as $\mathbf{c} = \{\mathbf{m}, \mathbf{p_1}, \mathbf{p_2}\}$. $\mathbf{p_1}$ and $\mathbf{p_2}$ are parity-check bits. $\mathbf{p_1}$ has length g, $\mathbf{p_2}$ has length (k-g) and g is the gap. From those encoding algorithms of LDPC codes, it is obvious that a LDPC code can be divided into two parts: information bits and parity-check bits. Information bits are one-to-one mapped to the original message $\mathbf{m}$, and parity-check bits are generated as linear combinations of parts of information bits.

For LDPC codes $\mathbf{y_1}$ and $\mathbf{y_2}$, let $\mathcal{I}_1, \mathcal{I}_2$ denotes the set of information bit index and $\mathcal{P}_1, \mathcal{P}_2$ denotes the set of parity-check bit index. $\mathcal{I}_1, \mathcal{I}_2 \subseteq \{0, 1, \ldots, N-1\}$ and $\mathcal{P}_1, \mathcal{P}_2 \subset \{0, 1, \ldots, N-1\}$. Meanwhile, let $g(\cdot) : \mathcal{I}_1 \to \mathcal{I}_2$ to be a one-to-one mapping such that $y_{1,i} = y_{2,g(i)}$.

To construct a combined code, firstly, received codes $\mathbf{y_1}$ and $\mathbf{y_2}$ are divided into information bits and parity bits such that $\mathbf{y_1} = \{\mathbf{y}_{\mathbf{1},\mathcal{I}_1}, \mathbf{y}_{\mathbf{1},\mathcal{P}_1}\}$, $\mathbf{y_2} = \{\mathbf{y}_{\mathbf{2},\mathcal{I}_2}, \mathbf{y}_{\mathbf{2},\mathcal{P}_2}\}$. $\mathbf{y}_{\mathbf{1},\mathcal{I}_1} = \{y_{1,i} | i \in \mathcal{I}_1\}$ and $\mathbf{y}_{\mathbf{1},\mathcal{P}_1} = \{y_{1,i} | i \in \mathcal{P}_1\}$. Similar notations are applied to $\mathbf{y}_{\mathbf{2},\mathcal{I}_2}$ and $\mathbf{y}_{\mathbf{2},\mathcal{P}_2}$. Since the information bits of $\mathbf{y_1}$ and $\mathbf{y_1}$ carry the same original message $\mathbf{m}$, thus a combined code $\mathbf{y}$ is constructed as:

$$\mathbf{y} = \{\mathbf{y}_{\mathcal{I}}, \mathbf{y}_{1,\mathcal{P}_1}, \mathbf{y}_{2,\mathcal{P}_2}\} \tag{3.5}$$

Information bits $\mathbf{y}_{\mathcal{I}}$ are obtained as

$$\mathbf{y}_{\mathcal{I}} = \{\mathbf{y}_i = \frac{\mathbf{y}_{1,i} + \mathbf{y}_{2,g(i)}}{2} | i \in \mathcal{I} = \mathcal{I}_1\} \tag{3.6}$$

Then, a constructed *combined* codeword is $y_0^{2N-K-1} = (y_{\mathcal{I}_1}, (y_0^{N-1})(1)_{\mathcal{P}_1}, (y_0^{N-1}(2)_{\mathcal{P}_2})$. Having generated the combined code $\mathbf{y}$, the next step is constructing its parity-check matrix.

Theorem 3.3.1. Given two LDPC encoders $Encoder_1$ and $Encoder_2$ with parity-check matrices $\mathbf{H_1}$ and $\mathbf{H_2}$. $\mathbf{c}_1$ and $\mathbf{c}_2$ with length of N are encoded using the same original message and $Encoder_1$ and $Encoder_2$. For a combined code $\mathbf{c} = \{\mathbf{c}_{\mathcal{I}}, \mathbf{c}_{1,\mathcal{P}_1}, \mathbf{c}_{2,\mathcal{P}_2}\}$, $\mathbf{c}_{\mathcal{I}}$ is defined in Equation 3.6, its parity-check matrix H is constructed as follows.

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_{1,\mathcal{I}_1} & \mathbf{H}_{1,\mathcal{P}_1} & \mathbf{0} \\ \mathbf{H}_{2,\mathcal{I}_2} & \mathbf{0} & \mathbf{H}_{2,\mathcal{P}_2} \end{bmatrix} \tag{3.7}$$

Proof. Suppose that $\mathbf{H}_1 = [\mathbf{H}_{1,0}, \mathbf{H}_{1,1}, \ldots, \mathbf{H}_{1,N-1}]$, where $\mathbf{H}_{1,i}^{M\times 1}$ is a sub-matrix representing the i^{th} column of $\mathbf{H}_1$. Divide $\mathbf{H}$ into two sub-matrix according to information bits and parity-check bits as $\mathbf{H}_{1,\mathcal{I}_1} = [\mathbf{H}_{1,i} | i \in \mathcal{I}_1]$ and $\mathbf{H}_{1,\mathcal{P}_1} = [\mathbf{H}_{1,i} | i \in \mathcal{P}_1]$. Similarly, $\mathbf{H_2}$ is divided into $\mathbf{H}_{2,\mathcal{I}_2} = [\mathbf{H}_{2,g(i)} | i \in \mathcal{I}_1]$ and $\mathbf{H}_{2,\mathcal{P}_2} = [\mathbf{H}_{2,i} | i \in \mathcal{P}_2]$. $\mathcal{I}_j$ and $\mathcal{I}_j$ are information bits and parity check bits of code $\mathbf{c}_j$ for j = 1,2.

From the statement, it has that $\mathbf{H}_1 \times \mathbf{c}_1 = \mathbf{0}$ and $\mathbf{H}_2 \times \mathbf{c}_2 = \mathbf{0}$. By dividing each code into information bits and parity check bits, these equations lead to that $[\mathbf{H}_{1,\mathcal{I}_1}, \mathbf{H}_{1,\mathcal{P}_1}] \times [\mathbf{c}_{1,\mathcal{I}_1}, \mathbf{c}_{1,\mathcal{P}_1}]^T = \mathbf{0}$ and $[\mathbf{H}_{2,\mathcal{I}_2}, \mathbf{H}_{2,\mathcal{P}_2}] \times [\mathbf{c}_{2,\mathcal{I}_2}, \mathbf{c}_{2,\mathcal{P}_2}]^T = \mathbf{0}$. Afterward,

applying them to combined code $\mathbf{c}$ results in

$$\begin{aligned}\mathbf{H} \times \mathbf{c}^T &= \begin{bmatrix} \mathbf{H}_{1,\mathcal{I}_1} & \mathbf{H}_{1,\mathcal{P}_1} & \mathbf{0} \\ \mathbf{H}_{2,\mathcal{I}_2} & \mathbf{0} & \mathbf{H}_{2,\mathcal{P}_2} \end{bmatrix} \times \begin{bmatrix} \mathbf{c}_{\mathcal{I}}{}^T \\ \mathbf{c}_{1,\mathcal{P}_1}{}^T \\ \mathbf{c}_{2,\mathcal{P}_2}{}^T \end{bmatrix}^T \\ &= \begin{bmatrix} \mathbf{H}_{1,\mathcal{I}_1} \times \mathbf{c}_{\mathcal{I}}{}^T & \mathbf{H}_{1,\mathcal{P}_1} \times \mathbf{c}_{1,\mathcal{P}_1}{}^T & \mathbf{0} \\ \mathbf{H}_{2,\mathcal{I}_2} \times \mathbf{c}_{\mathcal{I}}{}^T & \mathbf{0} & \mathbf{H}_{2,\mathcal{P}_2} \times \mathbf{c}_{2,\mathcal{P}_2}{}^T \end{bmatrix}\end{aligned} \tag{3.8}$$

Since information bits $\mathbf{c}_{\mathcal{I}}$ is constructed as $\mathbf{c}_{\mathcal{I}} = \{\mathbf{c}_i = \frac{\mathbf{c}_{1,i}+\mathbf{c}_{2,g(i)}}{2} | i \in \mathcal{I} = \mathcal{I}_1\}$. Additionally, the information bits is supposed to be the same as original massage s, which means $\mathbf{c}_i = \mathbf{c}_{1,i} = \mathbf{c}_{2,g(i)}, i \in \mathcal{I}$. Therefore, combining Equation 3.13, it proofs that $\mathbf{H} \times \mathbf{c}^T = \mathbf{0}$. □

According to Theorem 3.3.1, the parity-check matrix $\mathbf{H}$ for $\mathbf{y}$ is constructed of the form shown in Equation 3.7. An example is illustrated in Figure 3.2. The joint sum-product decoding algorithm is similar as joint decoding algorithm for identical content-replicated codes. First, construct a combined code $\mathbf{y}$ by Equation 3.5 and its parity-check matrix $\mathbf{H}$ by Equation 3.7. Then, apply the sum-product iterative decoding process for the combined code $\mathbf{y}$ until the iteration reaches a certain amount or the LLRs at variable nodes becomes infinite.

1. Construct a combined code and its parity-check matrix referring to Equation 3.5 and Equation 3.7. Let $\lambda(V_i)$ to be the edge distribution at variable node V_i and $\rho(C_j)$ to be the edge distribution at parity-check node C_j. Then, $\lambda(V_i) = 2\omega_c$, for $i \in \mathcal{I}_1$, and $\lambda(V_i) = \omega_c$, for $i \in \mathcal{P}_1 \bigcup \mathcal{P}_2$.

2. Calculate the initial LLR message $\mu_{\mathbf{0}}$ at variable node V_i for $i \in [0, N+K-1]$.

3. Let $\mathbb{C}_i$ to be the set of check nodes which are connected to variable node V_i in

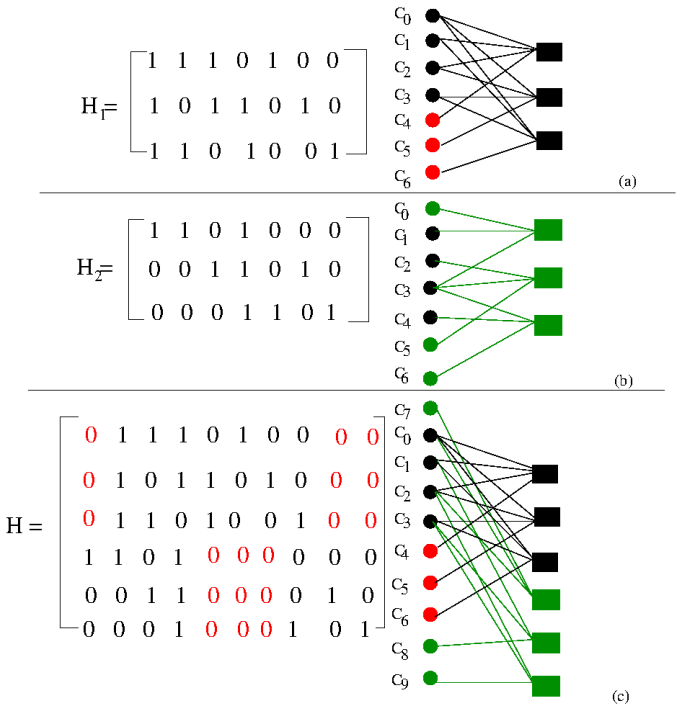


Figure 3.2: Illustration of the parity-check matrix H for combined LDPC code $\mathbf{c}$ of two content-replicated LDPC codes $\mathbf{c_1}$ and $\mathbf{c_2}$ with different parity-check constraints [3] [4].

parity-check matrix H. Then, the message that V_i sends to C_j, which denotes a check node connected to V_i, at round l becomes:

$$m_{ij}^{l} = \mu_{0,i} + \sum_{j' \in \mathbb{C}_i \&\& j' \neq j} m_{j'i}^{l-1} \tag{3.9}$$

in which $l > 0$ and $m_{ij}^{0} = \mu_{0,i}$.

4. Let $\mathbb{V}_j$ to be the set of variable nodes which are connected to check node C_j. Then, the message that C_j sends to V_i at round l is obtained by:

$$m_{ji}^{l} = 2tanh^{-1} \prod_{i' \in \mathbb{V}_j \&\& i' \neq i} \frac{tanh(m_{i'j}^{l})}{2} \tag{3.10}$$

5. The joint sum-product decoding algorithm is executed for a maximum number of iterations or until the LLRs at variable nodes are closed to $\pm\infty$, whichever comes first.

3.4 Content-replicated LDPC Codes with An Intermediate Parity-check Matrix

This section will discuss the joint decoding mechanism for the content-replicated LDPC codes with an intermediate parity-check matrix between information bits, named as related content-replicated LDPC codes. Two content-replicated LDPC codes are related in the case that there is a intermediate parity-check matrix between the information bits of two codes. More specifically, given two codes $\mathbf{y}_1 = G_1 \times \mathbf{m}_1$ and $\mathbf{y}_2 = G_2 \times \mathbf{m}_2$, $\mathbf{y}_{i,\mathcal{I}_1}$ and $\mathbf{y}_{i,\mathcal{P}_1}$, are information bits and parity-check bits of codes $\mathbf{y}_i$ as notated in Section 3.3, i = 1,2. $\mathbf{y}_1$ and $\mathbf{y}_2$ are related content-replicated if they satisfy that $\mathbf{y}_3 = G_3 \times \mathbf{m}$ and $\mathbf{m}_1 = \mathbf{y}_{3,\mathcal{I}_3}$, $\mathbf{m}_2 = \mathbf{y}_{3,\mathcal{P}_3}$. $\mathbf{m}_3$ is an intermediate LDPC code between $\mathbf{m}_1$ and $\mathbf{m}_2$ with rate as $\frac{1}{2}$. $\mathcal{I}_3$ and $\mathcal{P}_3$ denotes the index set of information bits and parity-check bits of $\mathbf{y}_3$. An example is presented in Figure 3.3

A combined code $\mathbf{y}$ is constructed by jointing $\mathbf{y}_1$ and $\mathbf{y}_2$ linearly in the following way:

$$\mathbf{y} = \{\mathbf{y}_{1,\mathcal{I}_1}, \mathbf{y}_{1,\mathcal{P}_1}, \mathbf{y}_{2,\mathcal{I}_2}, \mathbf{y}_{2,\mathcal{P}_2}\} \tag{3.11}$$

Theorem 3.4.1. Given two LDPC code $\mathbf{c}_1^N$ and $\mathbf{c}_2^N$ with parity-check matrix $\mathbf{H_1}$ and $\mathbf{H_2}$. $\mathbf{c}_1^N$ and $\mathbf{c}_2^N$ are related content-replicated and has the same rate $R = \frac{K}{N}$. Let $\mathbf{c}_3^{2K}$ to be the intermediated LDPC codes with rate 1/2. $\mathbf{c}_1^N$ is encoded using the information bits of $\mathbf{c}_3$ and $\mathbf{c}_2^N$ is encoded using the parity-check bits of $\mathbf{c}_3^{2K}$. For a combined code $\mathbf{c}$ generated using Equation 3.11, the parity-check matrix H is

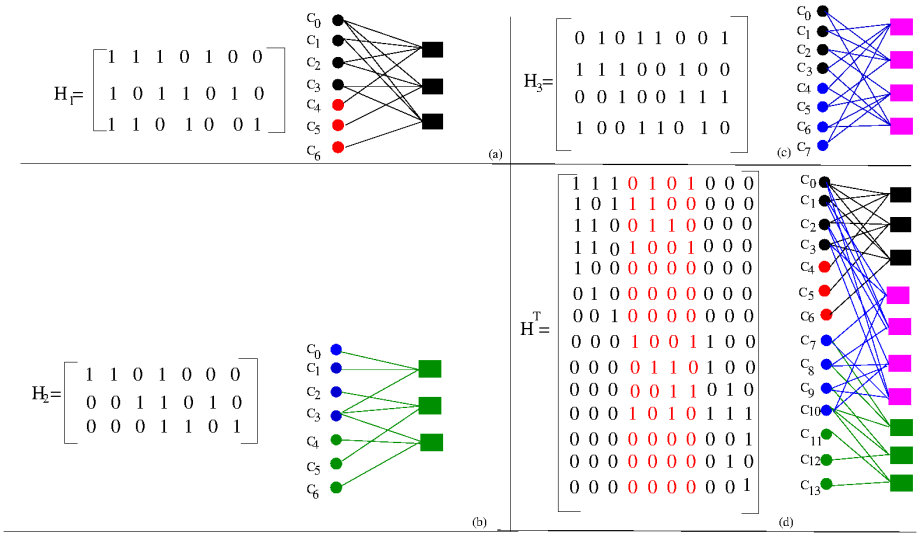


Figure 3.3: Illustration of two related content-replicated LDPC codes along with their tanner graphs and parity-check matrix [3] [4].

constructed as following.

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_{1,\mathcal{I}_1} & \mathbf{0} & \mathbf{H}_{1,\mathcal{P}_1} & \mathbf{0} \\ \mathbf{0} & \mathbf{H}_{2,\mathcal{I}_2} & \mathbf{0} & \mathbf{H}_{2,\mathcal{P}_2} \\ \mathbf{H}_{3,g(\mathcal{I}_3)} & \mathbf{H}_{3,f(\mathcal{P}_3)} & \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{3.12}$$

Proof. Let $H_{i,j}$ to be the j^{th} column of parity-check matrix $\mathbf{H}_i$. Similar as 3.3.1, $\mathbf{H}_{i,\mathcal{I}_i} = \{H_{i,j} | j \in \mathcal{I}_3\}$ and $\mathbf{H}_{i,\mathcal{P}_i} = \{H_{i,j} | j \in \mathcal{P}_3\}$ for i = 1, 2 and 3. From the statement, we have that $H_3 \times \mathbf{c}_3 = [H_{3,\mathcal{I}_3}, H_{3,\mathcal{P}_3}] \times [\mathbf{c}_{3,\mathcal{I}_3}, \mathbf{c}_{3,\mathcal{P}_3}]^T = \mathbf{0}$. Likely, $\mathbf{c}_1$ and $\mathbf{c}_2$ maintain that $H_3 \times \mathbf{c}_1 = [H_{1,\mathcal{I}_1}, H_{1,\mathcal{P}_1}] \times [\mathbf{c}_{1,\mathcal{I}_1}, \mathbf{c}_{1,\mathcal{P}_1}]^T = \mathbf{0}$ and $H_2 \times \mathbf{c}_2 = [H_{2,\mathcal{I}_2}, H_{2,\mathcal{P}_2}] \times [\mathbf{c}_{2,\mathcal{I}_2}, \mathbf{c}_{2,\mathcal{P}_2}]^T = \mathbf{0}$. Meanwhile, let $g(\cdot) : \mathcal{I}_3 \to \mathcal{I}_1$ to be a one-to-one mapping such that $c_{3,i} = c_{1,g(i)}$ and let $f(\cdot) : \mathcal{P}_3 \to \mathcal{I}_2$ to be a one-to-one mapping

such that $y_{3,i} = y_{2,f(i)}$. Therefore, we have:

$$\mathbf{H} \times \mathbf{c}^T = \begin{bmatrix} \mathbf{H}_{1,\mathcal{I}_1} & \mathbf{0} & \mathbf{H}_{1,\mathcal{P}_1} & \mathbf{0} \\ \mathbf{0} & \mathbf{H}_{2,\mathcal{I}_2} & \mathbf{0} & \mathbf{H}_{2,\mathcal{P}_2} \\ \mathbf{H}_{3,g(\mathcal{I}_3)} & \mathbf{H}_{3,f(\mathcal{P}_3)} & \mathbf{0} & \mathbf{0} \end{bmatrix} \times \begin{bmatrix} \mathbf{c}_{1,\mathcal{I}_1}{}^T \\ \mathbf{c}_{2,\mathcal{I}_2}{}^T \\ \mathbf{c}_{1,\mathcal{P}_1}{}^T \\ \mathbf{c}_{2,\mathcal{P}_2}{}^T \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{H}_{1,\mathcal{I}_1} \times \mathbf{c}_{\mathcal{I}_1}{}^T & \mathbf{0} & \mathbf{H}_{1,\mathcal{P}_1} \times \mathbf{c}_{1,\mathcal{P}_1}{}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{H}_{2,\mathcal{I}_2} \times \mathbf{c}_{\mathcal{I}_2}{}^T & \mathbf{0} & \mathbf{H}_{2,\mathcal{P}_2} \times \mathbf{c}_{2,\mathcal{P}_2}{}^T \\ \mathbf{H}_{3,g(\mathcal{I}_3)} \times \mathbf{c}_{1,\mathcal{I}_1}{}^T & \mathbf{H}_{3,f(\mathcal{P}_3)} \times \mathbf{c}_{2,\mathcal{I}_1}{}^T & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{H}_{1,\mathcal{I}_1} \times \mathbf{c}_{1,\mathcal{I}_1}{}^T & \mathbf{0} & \mathbf{H}_{1,\mathcal{P}_1} \times \mathbf{c}_{1,\mathcal{P}_1}{}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{H}_{2,\mathcal{I}_2} \times \mathbf{c}_{2,\mathcal{I}_2}{}^T & \mathbf{0} & \mathbf{H}_{2,\mathcal{P}_2} \times \mathbf{c}_{2,\mathcal{P}_2}{}^T \\ \mathbf{H}_{3,g(\mathcal{I}_3)} \times \mathbf{c}_{3,g(\mathcal{I}_3)}{}^T & \mathbf{H}_{3,f(\mathcal{P}_3)} \times \mathbf{c}_{3,f(\mathcal{P}_3)}{}^T & \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{3.13}$$

Finally, it proofs that $\mathbf{H} \times \mathbf{c}^T = \mathbf{0}$.

□

According to Theorem 3.4.1, the parity-check matrix $\mathbf{H}$ for $\mathbf{y}$ is constructed of the form shown in Equation 3.11 and Equation 3.12. The joint sum-product decoding algorithm is similar as joint decoding algorithm for identical content-replicated codes. First, construct a combined code $\mathbf{y}$ by Equation 3.11 and its parity-check matrix $\mathbf{H}$ by Equation 3.12. Then, apply the sum-product iterative decoding process for the combined code $\mathbf{y}$ until the iteration reaches a certain amount or the LLRs at variable nodes becomes infinite.

This section defines the joint decoding problems for content-replicated LDPC codes and lists three kinds of content-replicated codes. Further, for each category of content-replicated codes, the joint decoding algorithm for AWGN channels are explored. Next section will analyze the error correction performance of identical

content-replicated LDPC codes, different content-replicated LDPC codes and related content-replicated LDPC codes.

4. DENSITY EVOLUTION ANALYSIS*

Previous section presents the joint sum-product decoding algorithm for content-replicated codes. This section will further analyze error correction performance of the constructed joint sum-product decoding algorithms on AWGN channels. For LDPC codes, one measure of error correction performance is the expected fraction of incorrect messages passed at lth iteration. In [20], Richardson shows that, assuming the tanner graph doesn't contain any cycle with length equal or smaller than $2l$, $lim_{n\to\infty} P_e^n(l) = P_e^\infty(l)$, where n is the length of LPDC code. Thus, $P_e^\infty(l)$ could be estimated by density distribution evolution. Let σ^2 to be the variance of AWGN. There exist an threshold of AWGN channel noise variance σ^* with the properties: if $\sigma < \sigma^*$ then $\lim_{l\to\infty} P_e^\infty(l) = 0$, else if $\sigma < \sigma^*$ then there exists an constant $\upsilon(\sigma) > 0$ such that $\lim_{l\to\infty} P_e^\infty(l) > \upsilon(\sigma)$ for all $l > 0$.

Chung [16] present Gaussian Approximation method to estimate the threshold $\sigma^*$of LDPC codes using sum-product decoding on memory-less binary-input continuous-output AWGN channel. As mentioned previously, Gaussian Approximation based density evolution requires two assumptions: symmetry assumption and "cycle-free" assumption. "Cycle-free" assumption means that there is no cycles with length $2l$ or less in tanner graph which eliminates the existence of infinitely long loops in iterative decoding process. Symmetry assumption can be expressed as $f(x) = f(-x)e^x$. For a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$, symmetry assumption is reduced to $\sigma^2 = 2\mu$. The accuracy of Gaussian Approximation is largely improved by applying symmetry assumption. Another approach on density evolution is proposed

*Reprinted with permission from "Joint Decoding of Content-Replication Codes for Flash Memories " by Q. Li, H. Chang, A. Jiang and E. F. Haratsch, 53rd Annual Allerton Conference on Communication, Control, and Computing, Sept 29-Oct 2, 2015, Allerton Park and Retreat Center, Monticello, IL, USA, Copyright (2015) by IEEE.

by Fu [17]: represent the density distribution by a large population of "samples" and take advantage of the ergodic properties of iterative decoding process. To start the density evolution analysis of content-replicated codes with joint decoding algorithms, we firstly discussed the density distribution properties of combined LDPC codes and then compare the performance of Gaussian Approximation and Ergodicity theory based density evolution algorithm, typically for the joint-decoding problem.

As same as the notations in previous section, $\mathbf{y}_1$ and $\mathbf{y}_2$ are two noisy $(N, \omega_c, \omega_r)-$ *regular* LDPC codes. Let $\mathbb{P} \sim \mathcal{N}(\mu_{\mathbb{P}}, \sigma^2_{\mathbb{P}})$ and $\mathbb{Q} \sim \mathcal{N}(\mu_{\mathbb{Q}}, \sigma^2_{\mathbb{Q}})$ are memory-less binary-input continuous-output AWGN channels. This thesis concentrates on the case that $\mathbb{P}$ and $\mathbb{Q}$ have the same properties such that $\mu = \mu_{\mathbb{P}} = \mu_{\mathbb{Q}}$ and $\sigma = \sigma_{\mathbb{P}} = \sigma_{\mathbb{Q}}$. Other cases will be explored in future studies.

4.1 Performance of Identical Content-replicated LDPC Codes

According to Equation 3.1 in Section 3.2, given two identical content-replicated LDPC codes $\mathbf{y}_1$ and $\mathbf{y}_2$, a combined code $\mathbf{y}$ is obtained as $\mathbf{y} = \frac{\mathbf{y}_1 + \mathbf{y}_2}{2}$. For simplify the density evolution analysis process, it is assumed that the original code $\mathbf{x}$ is all zeros. Given that $\mathbf{y}_1$ and $\mathbf{y}_2$ are noisy codes transferred through AWGN channels $\mathbb{P}$ and $\mathbb{Q}$, thus we have that the probability distribution function of $\mathbf{y}_1$ and $\mathbf{y}_2$ is $\mathcal{N}(\mu, \sigma^2)$. Gaussian distribution has the following properties:

- If X and Y are two independent Gaussian distribution with means μ_1, μ_2 and standard deviation σ_1^2, σ_2^2, then their sum $X + Y$ will also be Gaussian distribution with mean $\mu_1 + \mu_2$ and variance $\sigma_1^2 + \sigma_2^2$.

- If X belongs to a Gaussian distribution with mean μ and standard deviation σ^2, then the variable $Y = aX + b$ is also a Gaussian distribution with mean $a\mu + b$ and standard deviation $(|a|\sigma)^2$, for $\forall a, \forall b \in \mathbb{R}$.

These properties of Gaussian distribution leads to that, if X_1 and X_2 are two independent Gaussian distributions, thus variable $X = \frac{X_1+X_2}{2}$ is also a Gaussian distribution with mean $\frac{\mu_1+\mu_2}{2}$ and variance $\frac{\sigma_1^2+\sigma_2^2}{4}$. Therefore, the probability distribution of noise in the combined code $\mathbf{y}$ should be a Gaussian distribution with mean μ and variance $\frac{\sigma^2}{2}$.

In Gaussian Approximation, one of the important assumptions is "symmetry assumption": $f(x) = f(-x)e^x$, which can be reduced to $2\mu = \sigma^2$ for Gaussian distributions. While, for joint-decoding process of identical LDPC codes, it is approved that $\mathbf{y} \sim \mathcal{N}(\mu, \sigma^2/2)$. Suppose that $\mathbf{y}_1$ and $\mathbf{y}_1$ satisfy "symmetry assumption", then it has that $2\mu_1 = \sigma_1^2$ and $2\mu_2 = \sigma_2^2$. With $\sigma_1 = \sigma_2$ and $\mu_1 = \mu_2$, it lead to that $2\mu = 2\mu_1 = \sigma_1^2 = \sigma^2$. Finally, it shows that the combine code $\mathbf{y}$ cannot hold the "symmetry assumption" as $\mathbf{y} \sim \mathcal{N}(\mu, \sigma^2/2)$: $2\mu_{\mathbf{y}} = 2\mu \neq \sigma_{\mathbf{y}}^2 = \sigma^2/2 = \mu$. As a result, the one parameter Gaussian Approximation model is not applicable to the density evolution of identical content-replicated LDPC codes with joint sum-product decoding algorithm.

Also, there is another approach on density evolution proposed by Fu. As mentioned previously, this ergodicity theory based density evolution algorithm takes advantages of a well- known property of ergodicity: any statistical parameter of the random process, including the density function itself, can be arbitrarily closely approximated by averaging over a sufficient number of samples [17]. There are two assumptions for this density evolution methods: cycle-free assumption and infinite size assumption, that the code is of infinite length. Under those two assumptions, it was approved that the updating process preserves ergodicity if the input message is i.i.d. So, for the identical content-replicated LDPC codes, firstly, this section will proof that the combined code $\mathbf{y}$ is i.i.d. Having that $\mathbf{y}_1$ and $\mathbf{y_2}$ are codes with additive white noise which is independent and random noise, it is shown that, for $\mathbf{y}$, the noisy

in each bit is also independent distributed with the same probability distribution.

Based on above consideration, the ergodicity theory based density evolution algorithm is used to estimate the threshold σ^*_{JSP} in the following sections. The ergodicity theory based density evolution algorithm for identical content-replicated LDPC codes using joint sum-product decoding algorithm is stated as below:

1. Choose a large number N and generate N LLR samples of combined code as $\mathbf{u}_0$. In particularly, each sample is generated according to Gaussian distribution $\mathcal{N}(\mu, \sigma^2/2)$.

2. Update messages from variable nodes to parity-check nodes. For iteration 0, let the message from variable node V_i to all its connected parity-check nodes be initialed as $v_i^{(0)} = u_{0,i}$. At other iterations l, take the N samples of messages send from parity-check nodes $u^{(l-1)}$. Generate $d_v - 1$ incoming message sequences for each variable node by randomly interleaving $u_i^{(l-1)}$ for $i = 1, 2, \ldots, d_v - 1$. Then, update the N samples of messages send from variable nodes to parity-check nodes as $v_i^{(l)} = u_{0,i} + \sum_{k=1}^{d_v-1} u_{k,i}$, where $\mathbf{u}_k = \{u_{k,0}, u_{k,1}, \ldots, u_{k,N-1}\}$ is the kth incoming message sample sequence. Figure 4.1 illustrates an example of the randomly interleaving based updating mechanism.

3. Update messages from parity-check nodes to variable nodes. For iteration l, take the N samples of messages send from variable nodes $v^{(l)}$. Similarly as Step 2, generate $d_c - 1$ incoming message sequences for each variable node by randomly interleaving $v_i^{(l)}$ for $i = 1, 2, \ldots, d_c - 1$. Then, update the N samples of messages send from parity-check nodes to variable nodes as $u_i^{(l)} = 2tanh^{-1} \prod_{k=1}^{d_c-1} \frac{tanh(v_{k,i})}{2}$, where $\mathbf{v}_k = \{v_{k,0}, v_{k,1}, \ldots, v_{k,N-1}\}$ is the kth incoming message sample sequence.

4. Repeat Step 2 and Step 3 until the iteration number reaches a certain amount or the LLRs at variable nodes converge to $\pm\infty$. For the message at ith variable node, $LLR_i = \infty$ represents that $Pr(y_i = -1) = 1$ and $LLR_i = -\infty$ represents that $Pr(y_i = 1) = 1$.

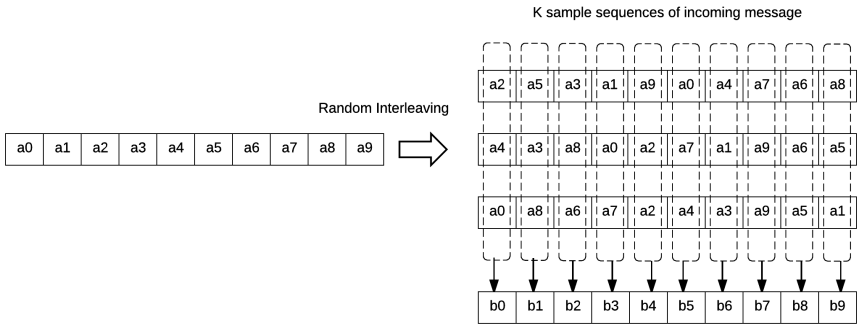


Figure 4.1: An example of updating mechanism based on random interleaving incoming sequence. Given an sample sequence $\mathbf{a}$ with length 10. For $j = 1, \ldots, K$, each times randomly interleaves $\mathbf{a}$ to get an incoming message sequence $\mathbf{a}_j$. Finally, update $\mathbf{b}$ based on the K incoming message sequences. Let $f(\cdot)$ to be the updating function: $\mathbf{b}_{,i} = f(\mathbf{a}_{1,i}, \ldots, \mathbf{a}_{K,i})$.

Using the ergodicity theory based density evolution, the threshold $\sigma^*_{JSP,Identical}$ for identical content-replicated LDPC codes with joint sum-product decoding algorithm is calculated. Different combinations of ω_c and ω_r is considered: (3,4), (3,5), (3,6), (4,6) and (4,8). Besides, in the following density evolution analysis, the number of samples is defined as 10^5, which is large enough to achieve good approximating accuracy in [17]. The maximum number of iterations is set to be 1000. The results are shown in Table 4.1 compared with σ^*_{SP} which represents the threshold of regular LDPC with sum-product decoding algorithm.

(ω_c,ω_r)	(3,4)	(3,5)	(3,6)	(4,6)	(4,8)
σ^*_{SP}	1.261	1.004	0.880	1.002	0.838
$\sigma^*_{JSP,Identical}$	1.555	1.264	1.116	1.242	1.044

Table 4.1: Density evolution analysis results of identical content-replicated LDPC codes with joint sum-product decoding algorithm $\sigma^*_{SP,Identical}$ and regular LDPC code with sum-product decoding algorithm σ^*_{SP}. Both results are obtained by using the ergodicity theory based density evolution algorithm.

4.2 Joint Decoder for Different Content-replicated Codes

This section will extend the density evolution analysis to different content-replicated codes defined in Section 3.3. The two content-replicated LDPC codes $\mathbf{y_1}$ and $\mathbf{y_2}$ are different in that way: $\mathbf{G}_1 \neq \mathbf{G}_2$, $\mathbf{H}_1 \neq \mathbf{H}_2$. Using the same notations as before, let $\mathcal{I}_1, \mathcal{I}_2$ denote the index set of information bits and $\mathcal{P}_1, \mathcal{P}_2$ denote the index set of parity-check bits. $g(\cdot)$ is a one-to-one mapping: $\mathcal{I}_1 \rightarrow \mathcal{I}_2$. According to Equation 3.5, a combined code is obtained as $\mathbf{y} = \{\mathbf{y}_{\mathcal{I}}, \mathbf{y}_{1,\mathcal{P}_1}, \mathbf{y}_{2,\mathcal{P}_2}\}$. The information bits of $\mathbf{y}$ is generated as $\mathbf{y}_{\mathcal{I}} = \{\mathbf{y}_i = \frac{\mathbf{y}_{1,i}+\mathbf{y}_{2,g(i)}}{2} | i \in \mathcal{I} = \mathcal{I}_1\}$. Meanwhile, the corresponding parity-check matrix H for code $\mathbf{y}$ is constructed as Equation 3.7 as:

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_{1,\mathcal{I}_1} & \mathbf{H}_{1,\mathcal{P}_1} & \mathbf{0} \\ \mathbf{H}_{2,\mathcal{I}_2} & \mathbf{0} & \mathbf{H}_{2,\mathcal{P}_2} \end{bmatrix} \tag{4.1}$$

Theorem 3.3.1 in Section 3.3 proofs that H is the parity-check matrix of combined LDPC code $\mathbf{y}$. So, this section will conduct density evolution for parity-check matrix H and code $\mathbf{y}$ with joint sum-product algorithm.

For the combined code constructed based on two identical content-replicated

codes, each bit is generated using the same method which leads to the same probability distribution of noise. However, for different content-replicated codes, the generating mechanism varies between information bits and parity-check bits. As a result, information bits and parity-check bits will have different probability distributions. Given two AWGN channels $\mathbb{P}$ and $\mathbb{Q}$ similar defined as previous section as $\mathbb{P}, \mathbb{Q} \sim \mathcal{N}(\mu, \sigma^2)$. Based on the combination method of different content-replicated codes, it easy to see that noise in information bits belongs to Gaussian distribution $\mathcal{N}(\mu, \sigma^2/2)$ and noise in parity-check bits is a Gaussian distributed as $\mathcal{N}(\mu, \sigma^2)$. In this case, information bits and parity-check bits should be considered separately.

4.2.1 Edge Distribution

In the tanner graph of a LDPC code, an edges represents the connection between a variable node and a parity-check node. Each variable node represents one bit of the LDPC code. Each parity-check node represents one parity-check equations on partial of information bits. An edge is called information edge if one of its end is connected to an information bit. Otherwise, the edge is a parity edge such that one of its end is connected to an parity-check bits. For an information edge, the distribution function of the degree of its connected variable nodes is

$$\lambda^{Info}(x) = \sum_{i=1}^{\omega_{Info}} \lambda_i^{Info} x^{i-1} \tag{4.2}$$

where λ_i^{Info} is the fraction of edges connecting to an variable node with degree i and $\sum_{i=1}^{\omega_{Info}} = 1$. ω_{Info} is the upper bound of the degree of a variable node which stands for an information bits. Similarly, for parity-check edges, that distribution function

of the degree of its connected variable nodes is

$$\lambda^{Parity}(x) = \sum_{i=1}^{\omega_{Parity}} \lambda_i^{Parity} x^{i-1} \tag{4.3}$$

where λ_i^{Parity} is the fraction of edges connecting to an variable node with degree i and $\sum_{i=1}^{\omega_{Parity}} = 1$. ω_{Parity} is the upper bound of the degree of a variable node which stands for a parity-check bit.

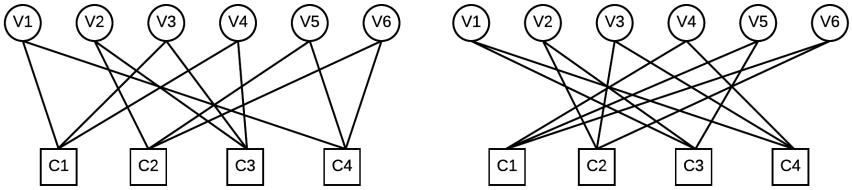
Theorem 4.2.1. Given two $(N, \omega_c, \omega_r) - regular$ LDPC code with parity-check matrix H_1 andH_2, which are not necessarily the same, a combined LDPC code is constructed according to Equation 3.5. The edge degree distribution of the variable nodes in the tanner graph of the combined LDPC code is:

$$\lambda^{Info}(x) = x^{2\omega_c - 1}, \lambda^{Parity}(x) = x^{\omega_c - 1} \tag{4.4}$$

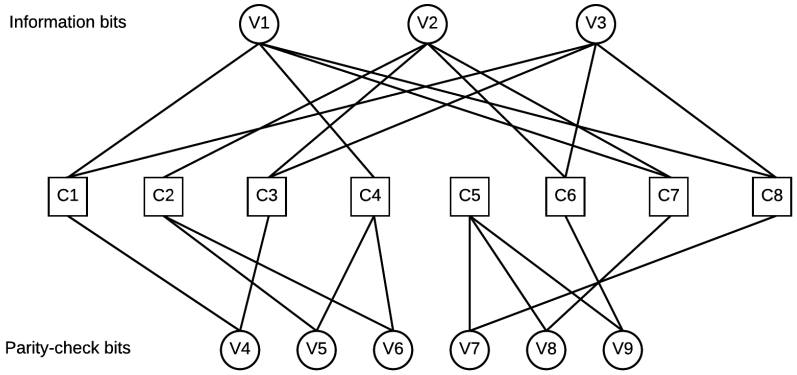
Proof. Based on the construction method of the parity-check matrix H of the combined code shown in Equation 3.7, in the corresponding tanner graph, information bits are connected to both check nodes in H_1 andH_2. Thus, the degree of information bits are doubled. Meanwhile, for the parity-check bits, which are only connected to either H_1 or H_2, the degree remains the same as ω_c. □

Figure 4.2a shows an example with two (6, 2, 4)-regular LDPC codes with parity-checcck matrices H_1 and H_2.

$$H_1 = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}, H_2 = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \tag{4.5}$$



(a) The tanner graphes of H_1(left) and H_2(right) according to Equation 4.5.



(b) The tanner graph of combined LDPC codes which are generated based on two regular LDPC codes shown in Figure 4.2a.

Figure 4.2: An example of identical content-replicated codes.

Then, the parity matrix for combined code is defined by the tanner graph shown in Figure 4.2b. In this example, according to Theorem 4.2.1, the degree distribution of variable nodes are $\lambda^{Info}(x) = x^{2\omega_c - 1} = x^3, \lambda^{Parity}(x) = x^{\omega_c - 1} = x$.

For a check node with degree ω_r, let $\rho_{j,k}$ to be the possibility that j edges are information edges and k edges are parity-check edges, and $j + k = \omega_r$. Thus, $\rho_{j,k} = \binom{j+k}{j}(\frac{\omega_r - \omega_c}{\omega_r})^j(\frac{\omega_c}{\omega_r})^k$. The degree distribution function of a check node becomes:

$$\rho(x, y) = \sum_{j=0, j+k=\omega_r}^{\omega_r} \rho_{j,k} x^j y^k$$

Then, the degree distribution function for an information related check nodes which

is a check node which is connected to at least one information edge is

$$\rho^{Info}(x,y) = \sum_{j=1,j+k=\omega_r}^{\omega_r} \frac{\rho_{j,k}}{1-\rho_{0,\omega_r}} x^{j-1} y^k$$

Similarly, the degree distribution function of a parity related check node which is a check node which is connected to at least one parity-check edges is

$$\rho^{Parity}(x,y) = \sum_{k=1,j+k=\omega_r}^{\omega_r} \frac{\rho_{j,k}}{1-\rho_{\omega_r,0}} x^{j} y^{k-1}$$

For the example shown in Figure 4.2b, $\rho(x,y) = \frac{1}{8}y^3 + \frac{1}{4}xy^2 + \frac{5}{8}x^2y$. For information related check node, the degree distribution function is $\rho(x,y) = \frac{2}{7}y^2 + \frac{5}{7}xy$. Meanwhile, for parity related check node, the degree distribution function is $\rho(x,y) = \frac{1}{8}y^2 + \frac{1}{4}xy + \frac{5}{8}x^2$.

4.2.2 Density Evolution Analysis

Previous discussion has shown that the probability distribution function varies between information bits and parity-check bits. Therefore, in the updating mechanism at check nodes and variable nodes, the incoming message send through an information edge and a parity-check edge should be treated separately. The message send from a check node to a variable node is estimated by considering all possible edge distribution cases.

To a check node which is connected to at least one information edges, in detail, there are ω_r possible edge distribution cases: there are j information edges, and $k = \omega_r - j$ parity edges with $1 \leq j \leq \omega_r$. For each case, let $u_i^{(l)}(j,k)$ denotes the message under the case: the check node is connected to j information edges and k parity edges. Having $v_i^{(l)}$,$v_p^{(l)}$ to be the received message from an information/parity

edge, then $u_i^{(l)}(j,k)$ is obtained as

$$u_i^{(l)}(j,k) = 2tanh^{-1}\Big(\prod_{s=1}^{j-1} tanh\frac{\nu_i^{(l)}(s)}{2}\prod_{s=1}^{k} tanh\frac{\nu_p^{(l)}(s)}{2}\Big) \tag{4.6}$$

By weighted summing up $u_i^{(l)}(j,k)$, for $j = 1,\ldots,\omega_r$, the message from a check node to an information related variable node could be approximated as:

$$u_i^{(l)} = \sum_{j=1}^{\omega_r} \rho_{j,k}^{Info} u_i^{(l)}(j,k) \tag{4.7}$$

where $\rho_{j,k}^{Info}$ is notated the same as previous section.

Similarly, the message send from a check node to a parity related variable node will be estimated by summing up generated message for all possible edge distribution cases. Let $u_p^{(l)}(j,k)$ represent the message calculated under the case that the check node has j parity edges and k information edges. $u_p^{(l)}(j,k)$ is defined by

$$u_p^{(l)}(j,k) = 2tanh^{-1}\Big(\prod_{s=1}^{k} tanh\frac{\nu_i^{(l)}(s)}{2}\prod_{s=1}^{j-1} tanh\frac{\nu_p^{(l)}(s)}{2}\Big) \tag{4.8}$$

Finally, the averaging message send from a check node to an parity related variable node is updated as:

$$u_p^{(l)} = \sum_{j=1}^{\omega_r} \rho_{j,k}^{Parity} u_p^{(l)}(j,k) \tag{4.9}$$

where $\rho_{j,k}^{Parity}$ is notated the same as previous section.

The density evolution analysis for different content-replicated codes is conducted based on the average probability distribution. Let $v_i^{(l)}$ to be the average message send from an information bit to a check node and $v_p^{(l)}$ to be the average message send from a parity-check bit to a check node. Meanwhile, let $u_i^{(l)}$ and $u_p^{(l)}$ to be the average message send from a check node to an information bit and a parity-check bit

respectively. $u_i^{(0)}$ and $u_p^{(0)}$ denotes the LLRs of received noisy LDPC code $\mathbf{y}$. Then, the updating mechanism is defined as follows.

Theorem 4.2.2. Given a combined LDPC code which is constructed based on different content-replicated LDPC codes as shown in Equation 3.5 and Equation 3.7. At the $l-round$ of sum-product decoding process, the message from a variable node to a check node is given by:

$$\begin{aligned} v_i^{(l)} &= u_i^{(0)} + \sum_{s=1}^{2\omega_c-1} u_i^{(l-1)}(s), \\ v_p^{(l)} &= u_p^{(0)} + \sum_{s=1}^{\omega_c-1} u_p^{(l-1)}(s), \end{aligned}$$

where $u_i^{(0)}$ is the LLRs of information bits of received code y_0^{2N-K-1}, and $u_p^{(0)}$ is the LLRs of parity bits. The updating mechanism of the message from a check node to a variable node is defined as:

$$\begin{aligned} u_i^{(l)} &= \sum_{j=1}^{\omega_r} \rho_{j,k}^{(i)} v_i^{(l)}(j,k), \\ u_p^{(l)} &= \sum_{j=1}^{\omega_r} \rho_{j,k}^{(p)} v_p^{(l)}(j,k), \end{aligned}$$

where $j+k=d_c$, $\rho_{j,k}^{(i)}$, $\rho_{j,k}^{(p)}$ represent edge degree distribution functions and are the same as previous sections.

Finally, this section presents a density evolution algorithm using the updating mechanism defined in Theorem 4.3.1. Similarly as the discussion on the density evolution method for identical content-replicated codes, the density evolution analysis will be conducted based on ergodicity theory using a large number of samples. There are two reasons for choosing ergodicity theory based density evolution algorithm rather than Gaussian Approximation: 1) the information bits and parity-check

bits of the combined code has different probability distribution functions, 2) message through information edges and parity edges is updated using different mechanisms. The method presented in [16] obtains approximate threshold for AWGN channel with sum-product decoding by making use of the so-called *symmetry condition* which requires a density function $f(x)$ to satisfy $f(x) = f(-x)e^x$. Because of these factors, the *symmetry condition* clearly does not hold for our joint decoder here, therefore we turn to the method presented in [17] to obtain the approximated threshold, which is verified by intensive numerical calculations as shown in Figure 4.3. The density evolution algorithm is listed below.

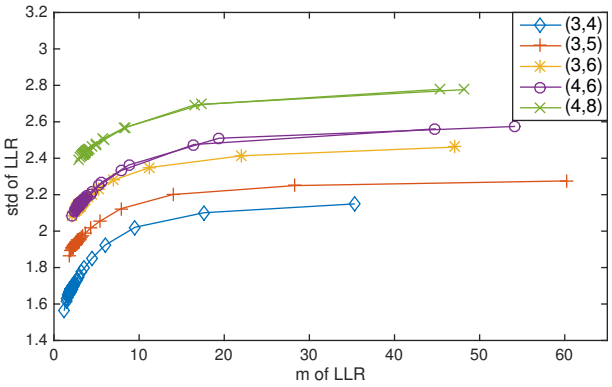


Figure 4.3: The evolution of μ and σ^2 in the joint decoding process of two content-replicated LDPC codes with different parity-check constraints [4].

1. Generate one initial LLR sequences $u_i^{(0)}$ with N samples as information bits according to $\mathcal{N}(0, \sigma^2/2)$, and respectively generate a N-sample sequence $u_p^{(0)}$ as parity bits according to $\mathcal{N}(0, \sigma^2)$.

2. Update message from variable nodes to check nodes. For information bits, at iteration 0, $\mathbf{v}_i^{(l)} = u_i^{(0)}$. At other iterations, generate a incoming message

sequence by randomly interleaving $u_i^{(l-1)}$, which is the LLR samples of parity nodes sent to information bits at $(l-1)$-round. Repeatedly generating $\omega_c - 1$ incoming message sequences. Then, calculate N samples of $\mathbf{v}_i^{(l)}$ by Theorem 4.2.2. Similarly, update N samples of $\mathbf{v}_p^{(l)}$.

3. Update $u_i^{(l)}$: message from a check node to an information bit. Firstly, calculate $u_i^{(l)}(j,k)$: generate $(\omega_r - 1)$ incoming message sequences, in which $(j-1)$ by interleaving $\mathbf{v}_i^{(l)}$, and the rest by interleaving $\mathbf{v}_p^{(l)}$; calculate N samples of $u_i^{(l)}(j,k)$ using these incoming sequences. Then, repeatedly calculated N samples of $u_i^{(l)}(j,k)$ for $j = 1, \ldots, \omega_r$. Finally $u_i^{(l)}$ is updated by Theorem 4.2.2.

4. Update $u_p^{(l)}$: message from a check node to a parity bit. Firstly, calculate $u_p^{(l)}(j,k)$: generate $(\omega_r - 1)$ incoming message sequences, in which $(j-1)$ by interleaving $\mathbf{v}_p^{(l)}$, and the rest by interleaving $\mathbf{v}_i^{(l)}$; calculate N samples of $u_p^{(l)}(j,k)$ using these incoming sequences. Then, repeatedly calculated N samples of $u_p^{(l)}(j,k)$ for $j = 1, \ldots, \omega_r$. Finally $u_p^{(l)}$ is updated by Theorem 4.2.2.

5. Repeat Step 2-3 until the iteration number reaches a certain amount or the LLRs at variable nodes converge to $\pm\infty$. For the message at ith variable node, $LLR_i = \infty$ represents that $Pr(y_i = -1) = 1$ and $LLR_i = -\infty$ represents that $Pr(y_i = 1) = 1$.

Using the ergodicity theory based density evolution, the threshold $\sigma^*_{JSP,Diff}$ for identical content-replicated LDPC codes with joint sum-product decoding algorithm is calculated. As same as previous section, different combinations of ω_c and ω_r is considered: (3,4), (3,5), (3,6), (4,6) and (4,8). The results are shown in Table 4.2 compared with σ^*_{SP} which represents the threshold of regular LDPC with sum-product decoding algorithm.

(ω_c,ω_r)	(3,4)	(3,5)	(3,6)	(4,6)	(4,8)
σ^*_{SP}	1.261	1.004	0.880	1.002	0.838
$\sigma^*_{JSP,Diff}$	1.690	1.379	1.190	1.300	0.065

Table 4.2: Density evolution analysis results of different content-replicated LDPC codes with joint sum-product decoding algorithm.

4.3 Joint Decoder for Related Content-replicated Codes

In this subsection, we present the density evolution analysis for related content-replicated codes using joint sum-product decoding algorithm. According to Section 3.4, given two $(N, \omega_c, \omega_r) - regular$ LDPC codes $\mathbf{y}_1$ and $\mathbf{y}_2$, $\mathbf{y}_1$ and $\mathbf{y}_2$ are related if there is an intermediate generating matrix between their information bits with rate 1/2. Let $\mathcal{I}_1$, $\mathcal{I}_2$ and $\mathcal{P}_1$, $\mathcal{P}_2$ be the same notations as before. Then, a combined LDPC code is obtained as $\mathbf{y} = \{\mathbf{y}_{1,\mathcal{I}_1}, \mathbf{y}_{1,\mathcal{P}_1}, \mathbf{y}_{2,\mathcal{I}_2}, \mathbf{y}_{2,\mathcal{P}_2}\}$. Meanwhile, the corresponding parity-check matrix H is constructed as shown in Equation 3.12.

For the density evolution, a $(2K, \omega'_c, \omega'_r) - regular$ LDPC codes with rate as 1/2 is used as the intermediate LDPC code between $\mathbf{y}_{1,\mathcal{I}_1}$ and $\mathbf{y}_{2,\mathcal{I}_1}$. Due to the existence of the intermediate LDPC codes, the updating mechanism is different for information bits and parity-check bits. Let $u_i^{(l)}$ and $u_p^{(l)}$ denote the message send from a information bit and a parity-check bit to a check node respectively. Similarly, let $v_i^{(l)}$ and $v_p^{(l)}$ denote the message send from a check node to a information bit and a parity-check bit. To give the updating algorithms for the combined code, firstly we explored the updating mechanism of the intermediate LDPC code.

Theorem 4.3.1. For the intermediate LDPC code, let x^l to be the average message

send from a variable node to a check node and y^l to be the average message send from a check node to a variable node. Thus, x^l and y^l are updated by

$$\begin{aligned} x^{(l)} &= u_i^{(0)} + \sum_{k=1}^{\omega_c - 1} u_i^{(l-1)}(k) + \sum_{k=1}^{\omega_c' - 1} y^{(l-1)}(k) \\ y^{(l)} &= 2tanh^{-1} \prod_{k=1}^{\omega_c' - 1} \frac{tanh(x^{(l-1)}(k))}{2} \end{aligned} \tag{4.10}$$

Proof. Both information bits and parity-check bits of the intermediate LDPC code are updated in the same way, since the information bits and parity-check bits share the same initial probability distribution function, which is independent white noise, also, H_1 and H_2 have the same rate and ω_c, ω_r. According to Equation 3.12, the parity check matrix of the combined code is constructed as

$$H = \begin{bmatrix} H_{1,\mathcal{I}_1} & \mathbf{0} & H_{1,\mathcal{P}_1} & \mathbf{0} \\ \mathbf{0} & H_{2,\mathcal{I}_2} & \mathbf{0} & H_{2,\mathcal{P}_2} \\ H_{3,\mathcal{I}_3} & H_{3,\mathcal{P}_3} & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

In the tanner graph of H, for variable nodes, there are two kinds of incoming message: one is from the edges which are involved in H_1 or H_2 and the other one is from the edges involved in H_3. Therefore, the message send from a variable node is updated by combining the incoming message from check nodes in H_1/H_2 and H_3. The incoming message send from the check nodes in H_1/H_2 is represented by $u_i^{(l)}$. On the other side, the check nodes of the intermediate code are only connected to the variable nodes of the intermediate code. So, the message from a check node to a variable node in the intermediate LDPC code is updated by combining all incoming message from its connected variable nodes. □

Based on the updating algorithms of the intermediate LDPC code, the message

between variable nodes and check nodes of H are updated using the following mechanism.

Theorem 4.3.2. For the joint sum-product decoding algorithm defined in Section 3.4, the message send from a variable node, which represents an information bit of the combined code, to a check node is updated as:

$$v_i^{(l)} = u_i^{(0)} + \sum_{k=1}^{\omega_c - 1} u_i^{(l-1)}(k) + \sum_{k=1}^{\omega_c' - 1} y^{(l-1)}(k) \tag{4.11}$$

Meanwhile, the message send from a variable node, which represents a parity-check bit of the combined code, to a check node is obtained as

$$v_p^{(l)} = u_p^{(0)} + \sum_{k=1}^{\omega_r - 1} u_p^{(l-1)}(k) \tag{4.12}$$

Similarly as the updating algorithms for the check nodes of different content-replicated codes, the message send from a check node to a variable node(information bit or parity-check bit) is updated as:

$$\begin{aligned} u_i^{(l)} &= \sum_{j=1}^{\omega_r} \rho_{j,k}^{(i)} v_i^{(l)}(j,k), \\ u_p^{(l)} &= \sum_{j=1}^{\omega_r} \rho_{j,k}^{(p)} v_p^{(l)}(j,k), \end{aligned}$$

where $j + k = \omega_r$ and $\rho_{j,k}^{(i)}$, $\rho_{j,k}^{(p)}$ denotes the degree distribution probability of check nodes with j information edges and k parity edges.

According to the updating mechanism at variable nodes and check nodes shown in Theorem 4.3.2, We present the density evolution algorithm for different content-replicated codes using joint sum-product decoding algorithm below.

1. Choose a large number N and generate N sample of initial LLR message at information related variable nodes as $u_i^{(0)}$ according to $\mathcal{N}(\mu, \sigma^2)$. Similarly, generate N samples of $u_p^{(0)}$ as the initial message at parity-check related variable nodes with the same probability distribution function as $u_i^{(0)}$.

2. Update message from information related variable nodes to check nodes $v_i^{(l)}$. For iteration 0, copy $u_i^{(0)}$ to $v_i^{(0)}$ and $u_p^{(0)}$ to $v_p^{(0)}$. For other iterations, take N samples of $u_i^{(l-1)}$ from previous iteration. For $j = 1, \ldots, \omega_c - 1$, each times generate a sample sequence of incoming message $u_i^{(l-1)}(k)$ by randomly interleaving $u_i^{(l-1)}$. Similarly, obtain $\omega_c' - 1$ sample sequences by randomly interleaving $y^{(l-1)}$. Then, update N samples of $v_i^{(l)}$ by the formula 4.11 of Theorem 4.3.2.

3. Update message from parity-check related nodes to check nodes $v_p^{(l)}$. For iteration 0, copy $u_p^{(0)}$ to $v_p^{(0)}$. For other iterations, take N samples of $u_p^{(l-1)}$ from previous iteration. For $j = 1, \ldots, \omega_c - 1$, each times generate a sample sequence of incoming message $u_p^{(l-1)}(k)$ by randomly interleaving $u_i^{(l-1)}$. Then, update N samples of $v_p^{(l)}$ by the formula 4.12 of Theorem 4.3.2.

4. Update message from check nodes to variable nodes $u_i^{(l)}$ and $u_p^{(l)}$. Take N samples of $v_i^{(l)}$ and $v_p^{(l)}$. For $j = 1, \ldots, \omega_r - 1$, each times generate a sample sequence of incoming message $v_i^{(l)}(k)$ by randomly interleaving $u_i^{(l-1)}$. Correspondingly, generate $(\omega_r - 1)$ sample sequences of incoming message for parity check bits by interleaving $v_p^{(l)}$.Then, update N samples of $u_i^{(l)}$ and $u_p^{(l)}$ by the formula 4.13 of Theorem 4.3.2.

5. Update message from check nodes to variable nodes of intermediate LDPC code $y^{(l)}$. Take N samples of $x^{(l)}$. For $j = 1, \ldots, \omega_r' - 1$, each times generate a sample sequence of incoming message $x^{(l)}(k)$ by randomly interleaving $x^{(l)}$.

Then, update N samples of $y^{(l)}$ by the formula 4.10.

6. Repeat Step 2-3 until the iteration number reaches a certain amount or the LLRs at variable nodes converge to $\pm\infty$. For the message at ith variable node, $LLR_i = \infty$ represents that $Pr(y_i = -1) = 1$ and $LLR_i = -\infty$ represents that $Pr(y_i = 1) = 1$.

Using the ergodicity theory based density evolution, the threshold for related content-replicated LDPC codes with joint sum-product decoding algorithm is calculated. Different intermediate LDPC codes are considered: $(N, \omega'_c, \omega'_r) - regular$ LDPC code with degree $(2, 4)$, $(3, 6)$ and $(4, 8)$. Let $\sigma^*_{JSP,\omega'_c,\omega'_r}$ be to threshold of different content-replicated codes with the intermediate LDPC code as $(N, \omega'_c, \omega'_r) - regular$ such that $\sigma^*_{JSP,\omega'_c,\omega'_r} = sup\{\sigma : u_i^{(l)} \to 0 \; as \; l \to \infty\}$. The results are shown in Table 4.3 compared with σ^*_{SP} which represents the threshold of regular LDPC with sum-product decoding algorithm.

(d_v,d_c)	(3,4)	(3,5)	(3,6)	(4,6)	(4,8)
σ^*_{SP}	1.261	1.004	0.880	1.002	0.838
$\sigma^*_{JSP,2,4}$	1.655	1.462	1.358	1.382	1.300
$\sigma^*_{JSP,3,6}$	1.500	1.267	1.161	1.207	1.091
$\sigma^*_{JSP,4,8}$	1.450	1.201	1.085	1.145	0.007

Table 4.3: Threshold of joint sum-product decoding algorithm for related content-replicated codes on AWGNC.

5. CONCLUSION

In this thesis, we study the joint-decoding problem of content-replicated codes in flash memory, especially the case that two LDPC codes carrying the same message. Also, we discuss three different kinds of content-replicated LDPC codes, which are classified according to the relationship between two noisy LDPC codes. For each category of content-replicated codes, I present the joint sum-product decoding algorithm along with the error-correcting performance analysis. The findings of this thesis include two parts:

- Using content-replicated codes with the corresponding joint sum-product decoding algorithm can effectively increase the error-correcting ability of LDPC codes, and further improve the data reliability of flash memory. All three kinds of content-replicated LDPC codes: identical, different and related content-replicated LDPC codes, show better performance than single LDPC on AWGN channels.

- Increasing the diversity of content-replicated codes may lead to the improvement of error-correcting performance under some constraints, according to the comparison of error-correcting performance of identical content-replicated LDPC codes and different/related content-replicated LDPC codes.

We propose the following future work: 1) In this study, we assume that two AWGN channels have the same properties. It is interesting to explore the joint decoding mechanism on channels with the same type but with different parameters or even different channels. 2) This thesis focuses on the joint decoding design for regular LDPC codes. Compared with regular LDPC codes, irregular LDPC codes have show

better error-correcting performance. Thus, exploring the joint decoder performance of content-replication codes consisting of irregular LDPC codes is another future work.

REFERENCES

[1] Y. Cai, G. Yalcin, O. Mutlu, E. F. Haratsch, A. Cristal, O. Unsal, and K. Mai. Error analysis and retention-aware error management for nand flash memory. *Intel Technol J*, 17(1):140, 2013.

[2] *TN-29-17: NAND Flash Design and Use Considerations Introduction*, 2010.

[3] Qing Li, Anxiao Jiang, and Erich F. Haratsch. Joint Decoder of Content-Replication Codes for NAND Flash Memories. In *Proc. Non-volatile Memory Workshop*, San Diego, CA, March 2015.

[4] Q. Li, H. Chang, A. Jiang, and E. F. Haratsch. Joint Decoding of Content-Replication Codes for Flash Memories. In *Proc. 53rd Annual Allerton Conference on Communication, Control and Computing (Allerton)*, Monticello, IL, October 2015.

[5] Richard Blish. Dose minimization during x-ray inspection of surface-mounted flash ics. Technical report, Spansion, 2008.

[6] J. Thatcher, T. Coughlin, J. Handy, and N. Ekker. Nand flash solid state storage for the enterprise: An in-depth look at reliability. Technical report, Solid State Storage Initiative (SNIA), 2009.

[7] Flash memory. `https://en.wikipedia.org/wiki/Flash_memory`.

[8] A. Maislos. A new era in embedded flash memory. *Flash memory summit*, 2011.

[9] *TN-29-63: Error Correction Code (ECC) in SLC NAND*, 2011.

[10] W. Liu, J. Rho, and W. Sung. Low-power high-throughput bch error correction vlsi design for multi-level cell nand flash memories. In *In Signal Processing Systems Design and Implementation, 2006. SIPS'06. IEEE Workshop on*, pages 303–308. IEEE, 2006.

[11] R. G. Gallager. Low-density parity-check codes. *IEEE Transaction on Information Theory*, 8(1):21–28, February 1962.

[12] T. K. Moon. *Error Correction Coding: Mathematical Methods and Algorithms*. John Wiley & Sons, Hoboken, NJ, 2005.

[13] R. M. Tanner. A recursive approach to low complexity codes. *IEEE Transactions on Information Theory*, pages 533–547, 1981.

[14] D. J. C. Mackay and R. M. Nea. Near shannon limit performance of low density parity check codes. *IEE Electronics Letters*, 32(8):1645–1626, August 1996.

[15] S.Y. Chung, G. D. Forney, T. J. Richardson, and R. Urbanke. On the design of low-density parity-check codes on the design of low-density parity-check codes within 0.0045 db of the shannon limit. *IEEE COMMUNICATIONS LETTERS*, 5(2):58–60, February 2001.

[16] Thomas J. Richardson Sae-Yong Chung and Rudiger L. Urbanke. Analysis of sum-product decoding of low-density parity-check codes using a gaussian approximation. *IEEE Transcations on Information Theory*, 47(2):657–670, Febuary 2001.

[17] Minyue Fu. On gaussian approximation for density evolution of low-density parity-check codes. In *Int. Conf. Communications*, Istanbul, May 2006.

[18] Kai Zhao, Wenzhe Zhao, Hongbin Sun, Tong Zhang, Xiaodong Zhang, and Nanning Zheng. Ldpc-in-ssd: Making advanced error correction codes work effectively in solid state drives. In *USENIX Conference on File and Storage Technologies (FAST)*, pages 243–256, San Jose, CA, Feburay 2013.

[19] T. J. Richardson and R. L. Urbanke. Efficient encoding of low-density parity-check codes. *Information Theory*, 47(2):638–656, 2001.

[20] Thomas J. Richardson and Rudiger L. Urbanke. The capacity of Low-Density Parity-Check Codes Under Message-Passing Decoding. *IEEE Transaction on Information Theory*, 47(2):599–618, February 2001.