國立交通大學

電子工程學系　電子研究所碩士班

碩士論文

適用於快閃記憶體之二位元軟輸入(9153,8256)
低密度奇偶校驗碼解碼器之設計與實作

Design and Implementation of a (9153,8256) LDPC Decoder with 2-bit Soft Input for NAND Flash Memory

學生：何堅柱

指導教授：張錫嘉 博士

中華民國　九十九年八月

適用於快閃記憶體之二位元軟輸入(9153,8256)

低密度奇偶校驗碼解碼器之設計與實作

Design and Implementation of a (9153,8256) LDPC Decoder

with 2-bit Soft Input for NAND Flash Memory

研究生：何堅柱	Student : Kin-Chu Ho
指導教授：張錫嘉　博士	Advisor : Dr. Hsie-Chia Chang

國立交通大學

電子工程學系 電子研究所 碩士班

碩士論文

A Thesis
Submitted to Department of Electronics Engineering & Institute Electronics
College of Electrical and Computer Engineering
National Chiao Tung University
In Partial Fulfillment of the Requirements
for the Degree of
Master of Science
in
Electronics Engineering
August 2010
Hsinchu, Taiwan, Republic of China

中華民國　九十九年八月

適用於快閃記憶體之二位元軟輸入(9153,8256)

低密度奇偶校驗碼解碼器之設計與實作

學生：何堅柱　　　　　　　指導教授：張錫嘉 博士

國立交通大學

電子工程學系　電子研究所碩士班

摘要

BCH碼因為硬體架構非常簡單，目前是應用在快閃記憶體系統上錯誤更正碼的主流。面對先進製程的發展與記憶體儲存容量的大幅提升所造成可靠度的降低，以代數解碼演算法為主的BCH碼只能不斷增加校驗碼的數量來提升解碼效能，如此一來也間接地減少資料所能儲存的空間。據此，本論文提出適用於快閃記憶體系統的低密度奇偶校驗碼（Low Density Parity Check, 簡稱LDPC Codes）及其解碼器架構，以二位元軟輸入之LDPC Codes提供在相同編碼率下比BCH碼更好的錯誤更正能力。

由於下世代快閃記憶體的儲存頁碼大小為1024Bytes，我們使用permutation matrix演算法建出編碼率為0.9 的 (9153,8256) LDPC Codes，並利用variable-node-centric sequential scheduling (簡稱VSS)來降低檢查節點運算元之電路複雜度。相較於傳統二階層 MinSum 硬體架構，本論文除有效地節省節點運算元的96%組合電路，藉由VSS，降低校驗節點運算元的76.6%暫存器。使用UMC 90nm製程，所提出的解碼器在工作頻率100MHz與10次解碼次數的情況下，最高吞吐量可達到每秒2.78Gbits。

Design and Implementation of a (9153,8256) LDPC Decoder with 2-bit Soft Input for NAND Flash Memory

Student : Kin-Chu Ho　　Advisor : Hsie-Chia Chang

Department of Electronics Engineering

Institute of Electronics

National Chiao Tung University

Abstract

This thesis proposes a LDPC decoder architecture for NAND flash memory system.BCH code is famous for NAND flash memory system because of its simple hardware architecture. However, advanced technology scale down and more bits of data stored per NAND Flash cell will cause the degradation of reliability. More parity bits are required to improve the correcting capability of BCH code. But this greatly degrades the storage capacity and is infeasible to commercial products. Soft input is required to improve the correcting capability of error correcting code. However, BCH code has only little improvement when soft input is provided. This thesis proposes a 2-bits soft input LDPC decoder, which can outperform BCH code under same code rate.

The (9153, 8256) LDPC code is constructed by permutation matrix algorithm with code rate 0.9. The variable-node-centric sequential scheduling (VSS) architecture is adopted and CNU is modified to reduce hardware complexity. Compared to the conventional Min-Sum two-stage pipelined architecture, the proposed architecture can reduce approximately 96% combination circuits of VNU and 76.8% registers. Using 90nm CMOS technology, the maximum throughput can achieve 2.78 Gbps under operating frequency of 100 Mhz with 10 iterations.

誌謝

不知不覺兩年的碩士生活就要結束了，要感謝很多人對我的照顧與幫忙。首先最要感謝是我的父母，很感激他們對我的支持。大學加碩士這六年，我都沒有辦法長時間陪伴在他們身邊，只有寒暑假才可以短暫回家探望他們。但他們還是沒有抱怨，支持我去做我想做的事。

我也要感謝我的指導教授張錫嘉老師，除了在學術研究上的指導外，也很關心我的生活狀況，很感謝他對我的包容。還有就是 LDPC GROUP 的陳志龍學長和嚴紹維學長，除了細心指導研究以外，還常常帶我去體驗新竹美食，對我非常照顧。

最後要感謝 OCEAN 與 OASIS 的每一位伙伴。一起在研究上共同奮鬥，一起聊天吃飯，慢慢培養了大家的感情。天下無不散之筵席，不少伙伴也在今年要離開 OCEAN 這個大家庭。雖然有點不捨，但也衷心祝福大家前程似錦。

其實心裡還有很多人想要感謝，但篇幅有限。最後讓我再次感謝每一位，謝謝你們的照顧與幫忙，謝謝您們!!

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Error correcting code is important to NAND flash memory system since error is unavoidable [1]. BCH code [2] [3] is famous for NAND flash memory system because of its simple hardware architecture and hard input requirement. As advanced technology scaled down and more bits of data stored per NAND flash cell, more errors are introduced. Under the limitation of number of parity bits, the correcting capability of BCH code is not enough to meet the requirement of next generatation NAND flash emory system. Soft input is required to improve the correcting capability of error correcting code. However, BCH code has only little improvement when soft input is provided [4] [5]. LDPC code [6] is a good candidate for its powerful correcting capability and simple decoding algorithm. 2-bit soft LDPC code can outperform BCH code with same code rate.

Low density parity check (LDPC) code is a famous error correcting code with near Shannon limit performance [7]. The parity check matrix H can be described by a Tanner graph [8]. The rows and columns of H are mapped to check nodes and variable nodes respectively. In standard belief propagation (BP) algorithm, a LDPC decoder exchanges messages between check nodes and variable nodes iteratively in fully parallel.

High code rate is a necessary condition for error correcting code applied on NAND flash memory system. A high code rate LDPC code introduces large row degree which causes implementation difficulty. The proposed LDPC code has a row degree of 81. The solution to this problem is variable-node-centric sequential scheduling (VSS) [9] [10]. VSS divides variable nodes into groups, and decodes in a scheduling order (partial parallel).

This greatly reduces the routing complexity and storage memory. A (9153, 8256) LDPC code is constructed by permutaion matrix algorithm with code rate is 0.9. The proposed LDPC code decoder has a better performance than BCH code with the same code rate when 2-bit soft input is provided. The maximum throughput can achieve 2.78 Gbps under operating frequency of 100Mhz with 10 iterations, using 90nm CMOS technology.

1.2 Thesis organization

The rest of this thesis is organized as follows. Chapter II gives the introduction of NAND flash memory. In Chapter III, we introduce the decoding algorithm, performance-related code paramemters and code construction. In Chapter IV, decoder architecture is presented. The simulation result is given in Chapter V and conclusion in Chapter VI.

Chapter 2

NAND Flash Memory

2.1 Introduction of NAND Flash Memory

This section introduces the flash memory system and basic operations : Programming, Erasing and Reading.

2.1.1 Flash Memory System

Flash memory is widely used for data storage in portable devices. Since flash memory is non-volatile, no power is needed to maintain the information stored. In addition, flash memory offers fast read access times comparing to hard disk. In this thesis, we take a NAND flash memory as the target flash memory.

There are three basic operations in NAND flash memory called programming, erasing and reading. NAND flash memory can be programmed and erased block by block. Each block contains number of pages. NAND flash memory can be read page by page. More details of these three operations will be presented in next section.

Fig. 2.1 shows the flash memory system. Data are transmitted in pages where a page size is equal to 4K or 8K bytes. One single page consists of data area and spare area. The data area stores the user data, and the spare area stores the system-control signal and parity bits of error correcting code (ECC). Pages are encoded before programming, and decoded after reading from flash memory.

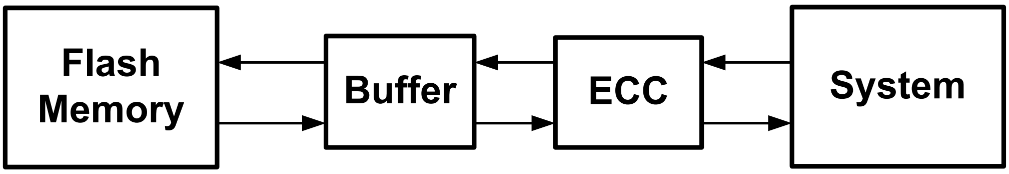


Figure 2.1: The Block Diagram of Flash Memory System.

2.1.2 NAND Flash Cell Programming

Fig. 2.2 shows a NAND flash Cell Programming. In a NAND flash Cell, there is a Floating Gate between the Gate and Substrate. When data is written into NAND flash Cell, 0V is applied to the Source and Drain. A high voltage (V_G) is applied to the Gate. Electrons in Substrate are attracted to the Floaging Gate. Different (V_G) can be applied to control the amount of electrons injected in Floating Gate. The amount of electrons injected in Floating Gate determines the threshold voltage of a NAND flash Cell.

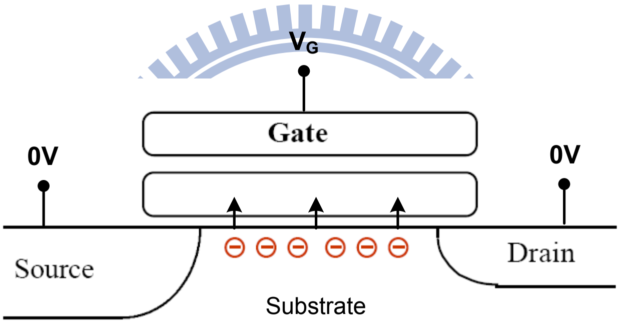


Figure 2.2: NAND Flash Cell Programming [1].

A Single Level Cell (SLC) means that only 1 bit data is stored per cell. Therefore, the threshold voltage region of a SLC is divided into two levels. Fig. 2.3 shows the threshold voltage distribution of SLC. For example, the threshold voltage is controlled to 2.5V if data 1 is stored, or 5.5V if data 0 is stored. There is variation of threshold voltage due to noise disturb and will be introduced in the next subsection.

2.1.3 NAND Flash Cell Erasing

Electrons in Floating Gate must be erased before reprogramming. When NAND flash Cell is earsed, 0V is applied to the Source, Drain and Gate. And high voltage (V_S) is

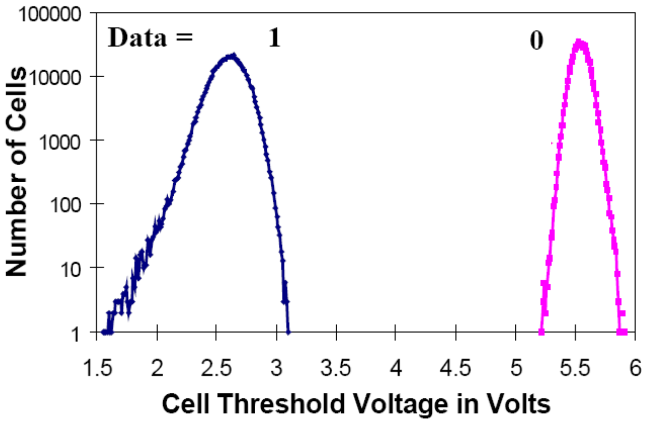


Figure 2.3: Threshold voltage distribution of a Signle Level Cell of NAND Flash Memory [1].

applied to the Substrate. Electrons in Floating Gate are attracted to the Substrate and no more electrons are left in Floating Gate.

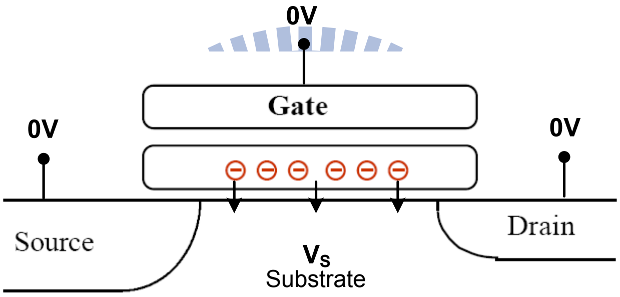


Figure 2.4: NAND Flash Cell Erasing [1].

2.1.4 NAND Flash Cell Reading

Fig. 2.5 shows NAND flash Cell Reading. To read a NAND flash cell, the selected wordlines are grounded and high voltage (V_D) is applied to the unselected wordlines. A bias is applied to the bitlines. Current will flow through the transistor if there is no charge stored in the cell.

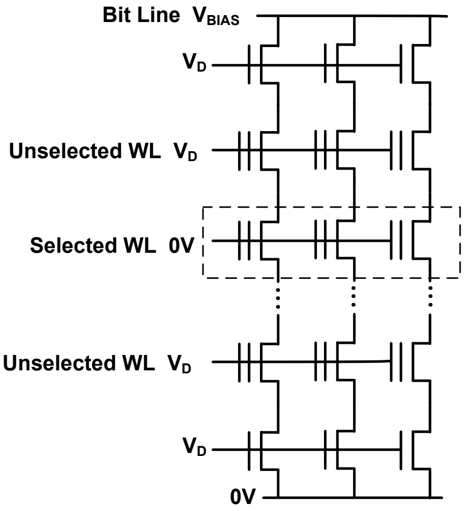


Figure 2.5: NAND Flash Cell Reading [1].

2.2 Reliability of NAND Flash Memory

Electron leakage, program and read disturb cause the variation of threshold voltage of NAND flash cell. Errors may be introducted if the threshold voltage shifts to other level. More details about noise disturb will be introduced in this subsection.

2.2.1 Electron Leakage

The number of electrons stored in Floating Gate decreases over time because electrons may leak from the NAND flash Cell. This problem can be solved by erasing and reprogramming periodly. But NAND flash Cell may be damaged when number of Program / Erase cycles increases. Leakage will be more serious if NAND flash Cell is damaged. Errors become unavoidable if NAND flash Cell is desired for a long time use.

2.2.2 Program Disturb

Fig. 2.6 shows the program disturb of a NAND flash Cell. Unselected cells on the same wordline or on adjacent wordlines of programmed cell, may suffer from voltage stress resulting in unwanted programming. Therefore, the threshold voltage of those unselected cells increases and may shift to other level.

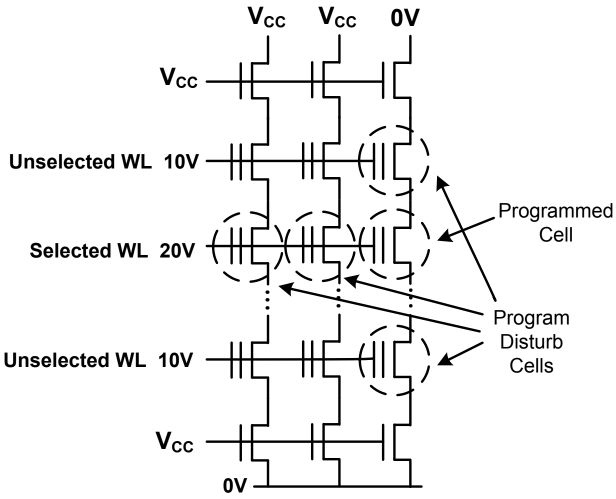


Figure 2.6: Program Disturb.

2.2.3 Read Disturb

Unselected cells adjacent to cells being read may suffer from voltage stress resulting in unwanted programming. As in program disturb case, the threshold voltage of those unselected cells increases and may shift to other level.

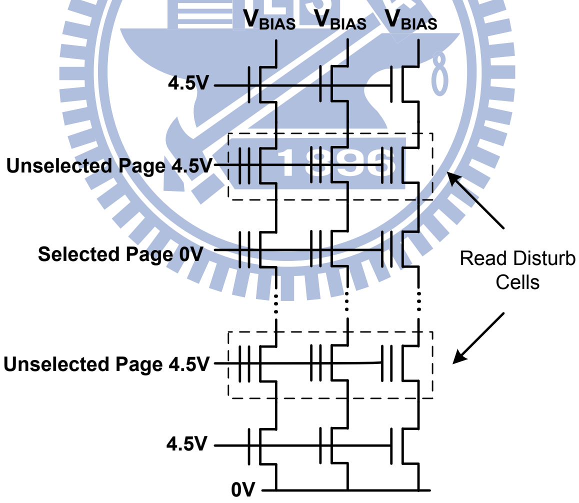


Figure 2.7: Read Disturb.

In Fig. 2.3 , threshold voltage below 4V represents data 1 is stored, and threshold voltage above 4V represents data 0 is stored. There is a tolerance range for the variation of threshold voltage. Data is still correct if the threshold voltage does not shift to other level.

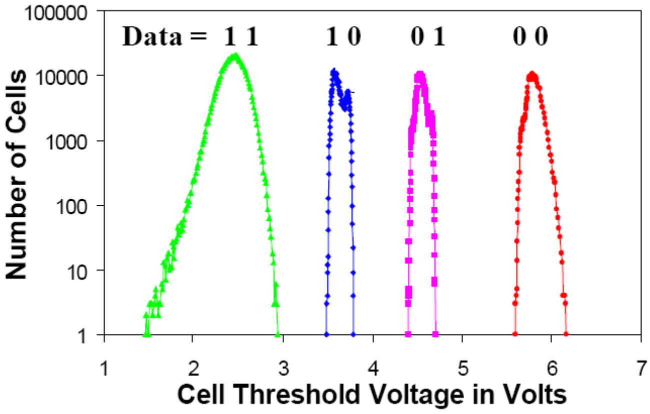


Figure 2.8: Threshold voltage distribution of a 2bits/cell NAND flash cell.

Fig. 2.8 shows a 2bits/cell NAND flash cell. The storage capacity is doubled comparing to the 1bit/cell NAND flash cell. Threshold voltage region is divided into 4 levels and region for each level is narrower. Therefore, the probability of threshold voltage shifting to other level is increased and led to degradation of reliability.

Nowadays, NAND flash memory system only provides hard input to error correcting code. For example, in Fig. 2.8, only three voltages (3.2V, 4V and 5.1V) are applied to check in which level the threshold voltage is. NAND flash memory system does not provide any information that how likely this bit to be ’0’ or ’1’. Information received by error correcting code is exactly ’0’ or ’1’. We call this hard input.

BCH code is feasible for its simple hardware architecture and only hard input requirement. However, advanced technology scale down and more bits of data stored per NAND flash cell will cause the degradation of reliability. More parity bits are required to improve the correcting capability of BCH code. The increase of spare area (area for parity bits storage) greatly degrades the data storage capacity and is infeasible to commerical product. To overcome this problem, NAND flash memory system will provide more information (soft input) in the next generation standard and much powerful error correcting code can be adopted.

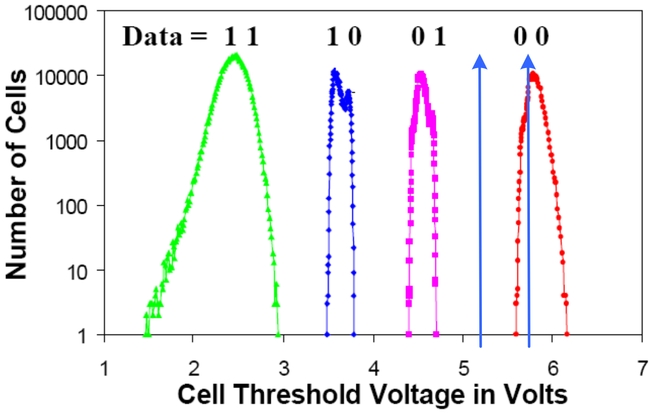


Figure 2.9: Threshold voltage distribution of a 2bits/cell NAND flash cell.

In Fig. 2.9, if data 01 is stored and threshold voltage shifts to 5.5V, hard input only provides that the second bit is a '0'. More information can be provided if one more voltage (5.8V) is applied to Gate. We can know that the threshold voltage is less than 5.8V, and the second bit has a high probability of being '1'. This provides more information for each data bit to error correcting code and we call this soft input.

BCH code has only little improvement when soft input is provided [4] [5]. LDPC code is probability-based and soft information can be well-used. Therefore, LDPC code is a good candidate for the next generation NAND flash memory system. Providing soft input will inrease reading latency in flash memory system. This is a trade-off between correcting capability and system latency. This thesis shows that only 2-bits soft input LDPC code can outperform BCH code under same code rate. Therefore, degradation to system latency is minimized.

Chapter 3

Low Density Parity Check Code

LDPC code was first discovered by Gallager [6] in the early 1960s. But it does not attract great attention until 1900s. The main reason is the high routing complexity making implementaion very difficult. Decoding algorithm of LDPC code is iterative message-passing decoding. Messages are passed between Check Node Unit (CNU) and Variable Node Unit (VNU) during decoding process. This iterative message-passing algorithm provides superior correcting ability and makes LDPC code widely adopted in communication application.

In this section, decoding algorithm will be introduced and performance-related code paramemters will be discussed. Finally, a code construction algorithm will be introduced.

3.1 Decoding Algorithm

3.1.1 Standard Belief Propagation (BP) Algorithm

The log-likelihood ratio (LLR) of intrinsic information of n^{th} variable node is denoted by P_n. The message from n^{th} variable node to m^{th} check node is denoted by z_{mn}. The message from m^{th} check node to n^{th} variable node is denoted by ϵ_{mn}. The a posteriori LLR of n^{th} bit is denoted by z_n. The current number of iteration and maximum number of iteration is represented by i and I_{Max} respectively. The standard BP is carried out as followed.

1.Initialzation:

Set $i = 1$. For each m, n, set $z^0_{mn} = P_n$

2.Iterative Decoding:

(a)check node to variable node update step, for $1 \leq m \leq M$ and each $n \in N(m)$, process

$$\epsilon^{i}_{mn} = 2\tanh^{-1}(\prod_{n' \in N(m)\backslash n}^{d} \tanh(\frac{z^{i-1}_{mn'}}{2})) \tag{3.1}$$

(b)variable node to check node update step, for $1 \leq n \leq N$ and each $m \in M(n)$, process

$$z^{i}_{mn} = P_n + \sum_{m' \in M(n)\backslash m} \epsilon^{i}_{m'n} \tag{3.2}$$

$$z^{i}_{n} = P_n + \sum_{m' \in M(n)} \epsilon^{i}_{m'n} \tag{3.3}$$

3.Hard Decision:

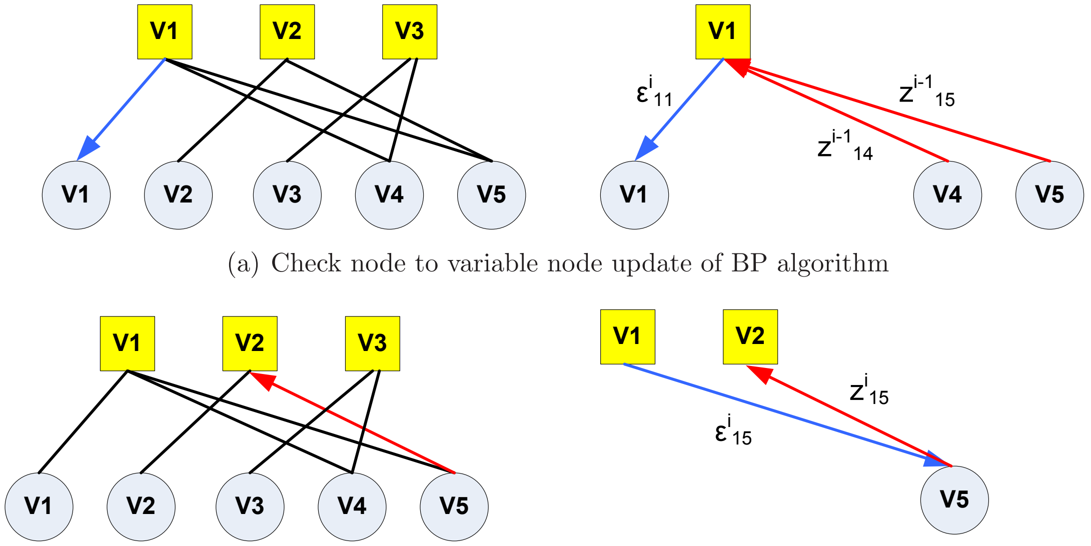
Let X_n be the n^{th} bit of decoded codeword. If $z^{(i)}_n \geq 0, X_n = 0$, else if $z^{(i)}_n < 0, X_n = 1$. If $H(x^{(i)})^t = 0$ or I_{MAX} is reached, the decoder stops and outputs the codeword. Otherwise, it sets $i = i + 1$ and goes on iterative decoding.

The iterative decoding processes for one iteration of standard BP is illustrated below. The messages are updated in parallel way between check nodes and variable nodes. The process is shown in Fig. 3.1.

3.1.2 Variable-node-centric Sequential Scheduling (VSS) Algorithm

High code rate LDPC code introduces high row degree. This makes implementation difficult due to the large number of inputs to sorter. The hardware cost and critical path of Check Node Unit (CNU) is greatly incresed. Shuffle decoding algorithm [9] [11] with variable-node-centric sequential scheduling architecture(VSS) [10] processes check node update procedure in G cycles, reducing the number of inputs to sorter.

In VSS approach, the initialization, and hard decision remain the same as the standard



(a) Check node to variable node update of BP algorithm

(b) Varibale node to check node update of BP algorithm

Figure 3.1: Illustratin of standard BP.

BP algorithm. The only difference between two algorithms is the updating procedure. Assume the N bits of a codeword are divided into G groups, so each group contains $N/G = N_G$ bits. The messages are only exchanged between variable nodes from one group and check nodes which are connected to that group. In addition, each group of messages is updated in order. Furthermore, one iteration takes N cycles. For $G = 1$, the VSS scheduling becomes standard BP.

The normalized min-sum (NMS) algorithm which compensates the approximation error in check node update step can also be applied to VSS approach with normalized factor $\beta = 0.5$. The updating procedure of NMS algorithm with VSS approach is carried out as follows.

1.Initialzation:

For each m, n, set $z^{0}_{mn} = P_n$

2.Iterative Decoding:

(a)check node to variable node update step, for $1 \leq g \leq G-1$, $1 \leq m \leq M$ and each

$n \in N(m)$, process

$$\begin{aligned}\epsilon^i_{mn} = &\prod_{n' \in N(m)\backslash n, n' \le g \cdot N_G - 1} sign(z^i_{mn'}) \times \prod_{n' \in N(m)\backslash n, n' \ge g \cdot N_G} sign(z^{i-1}_{mn'}) \times \\ &\min\left\{ \min_{n' \in N(m)\backslash n, n' \le g \cdot N_G - 1} \left\{|z^i_{mn'}|\right\}, \min_{n' \in N(m)\backslash n, n' \ge g \cdot N_G} \left\{|z^{i-1}_{mn'}|\right\} \right\} \times \beta\end{aligned} \tag{3.4}$$

(b)variable node to check node update step, for $g \cdot N_G \le n \le (g+1) \cdot N_G - 1$ and each $m \in M(n)$, process

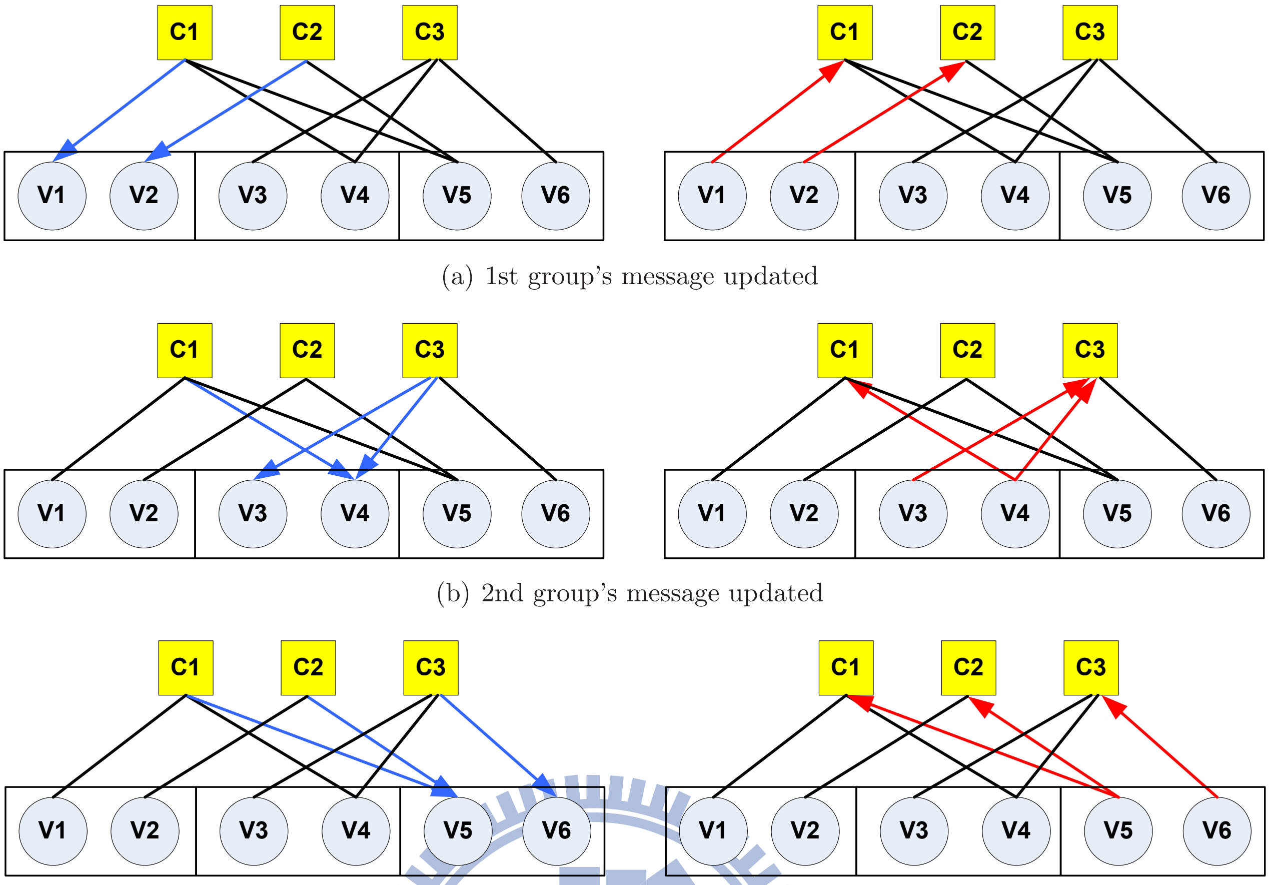
$$z^i_{mn} = P_n + \sum_{m' \in M(n)\backslash m} \epsilon^i_{m'n} \tag{3.5}$$

$$z^i_n = P_n + \sum_{m' \in M(n)} \epsilon^i_{m'n} \tag{3.6}$$

3.Hard Decision:

Let X_n be the n^{th} bit of decoded codeword. If $z^{(i)}_n \ge 0, X_n = 0$, else if $z^{(i)}_n < 0, X_n = 1$. If $H(x^{(i)})^t = 0$ or I_{MAX} is reached, the decoder stops and outputs the codeword. Otherwise, it sets $i = i + 1$ and goes on iterative decoding.

The decoding process for one iteration of VSS is illustrated in Fig. 3.2 with $G = 3$ as example. The arrows with blue color represent check node to variable node messages to be updated. The arrows with red color represent variable node to check node messages to be updated. On the other hand, black lines represent that messages are not updated in that cycle.



(a) 1st group's message updated

(b) 2nd group's message updated

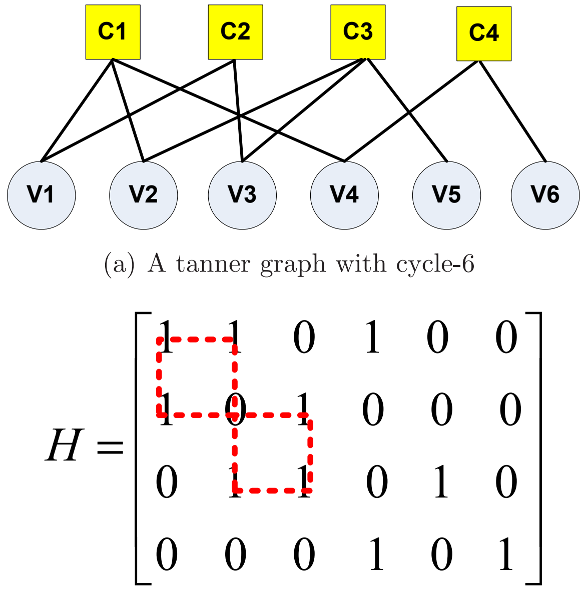
(c) 3rd group's message updated

Figure 3.2: Illusion of VSS.

3.2 Performance-Related Parameters

3.2.1 Cycles in Tanner Graph

A LDPC code with cycle-4 introduces smaller trapping set [12]. It will cause performance degradation in water fall region. For LDPC code, we call this performance degradation in water fall region, the error floor [13]. Therefore, constructing LDPC code with cycle-4 should be avoided and cycle should be as large as possible. Fig. 3.3 illustrates a Tanner Graph with cycle-6 cycles and its corresponding parity check matrix.



(a) A tanner graph with cycle-6

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

(b) Parity check matrix H corresponds to (a)

Figure 3.3: An example of a tanner graph with cycle-6.

3.2.2 Column Degree

A LDPC code with higher column degree has better performance in water fall region. It means that it can suppress the error floor in lower bit error rate region. Fig. 3.4 shows the performance of LDPC codes with different column degree. S represents scaling factor in this thesis.

In Fig. 3.4, (672, 588) is a LDPC code from IEEE 15.3c Standard, with column degree 3. It has poor performance at waterfall region due to its low column degree. LDPC code with column degree 8 and 12 has better performance at waterfall region.

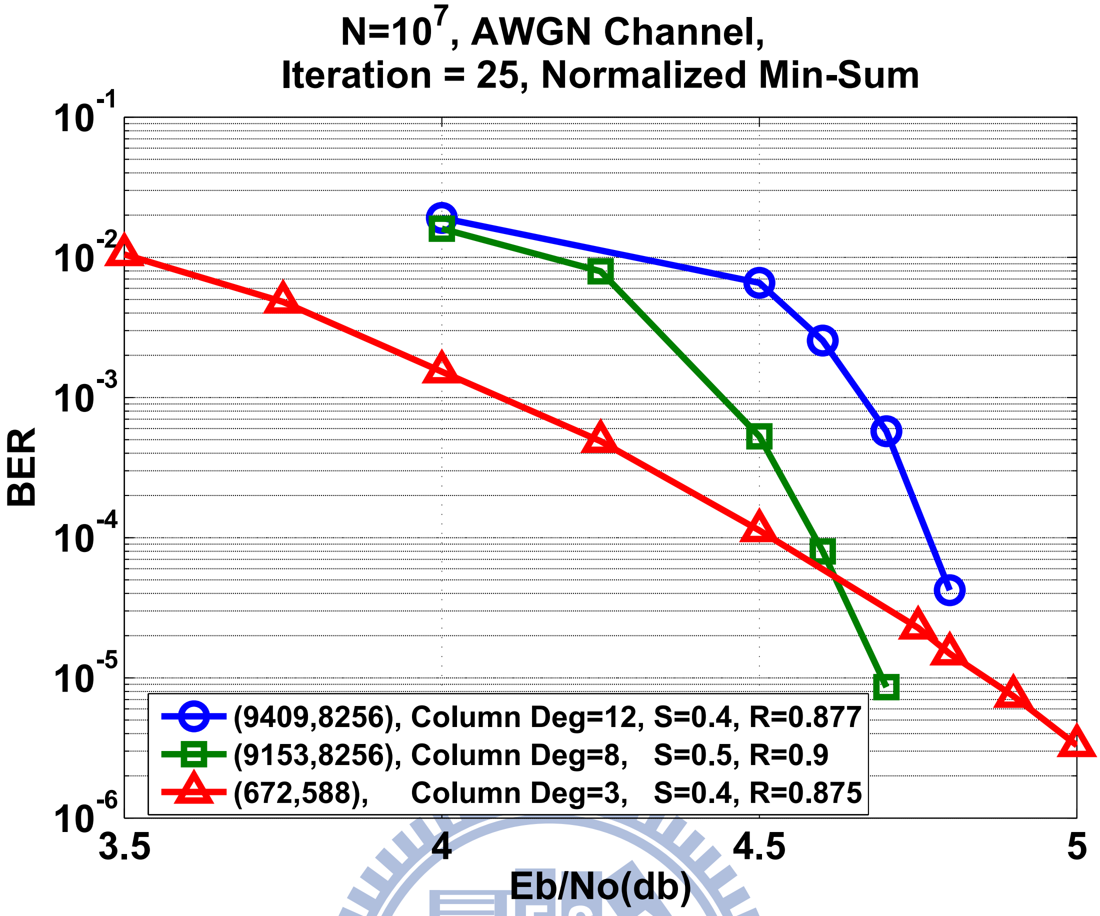


Figure 3.4: Performance of LDPC code with different column degree.

Fig 3.5 shows that LDPC code with higher column degree has better performance at waterfall region. Both (2071,1746) and (2033,1714) LDPC codes are constructed by permutation matrix algorithm [14] and will be introduced in next subsection. LDPC codes constructed by permutation matrix algorithm has no cycle-4. They are QC code [15] and their columne degree is 4. For (2048,1723) (IEEE 802.3an Standard [16]) LDPC, error floor will not appear until BER down to 10^{-10}. Thus, high column degree LDPC code is desired for NAND flash memory system.

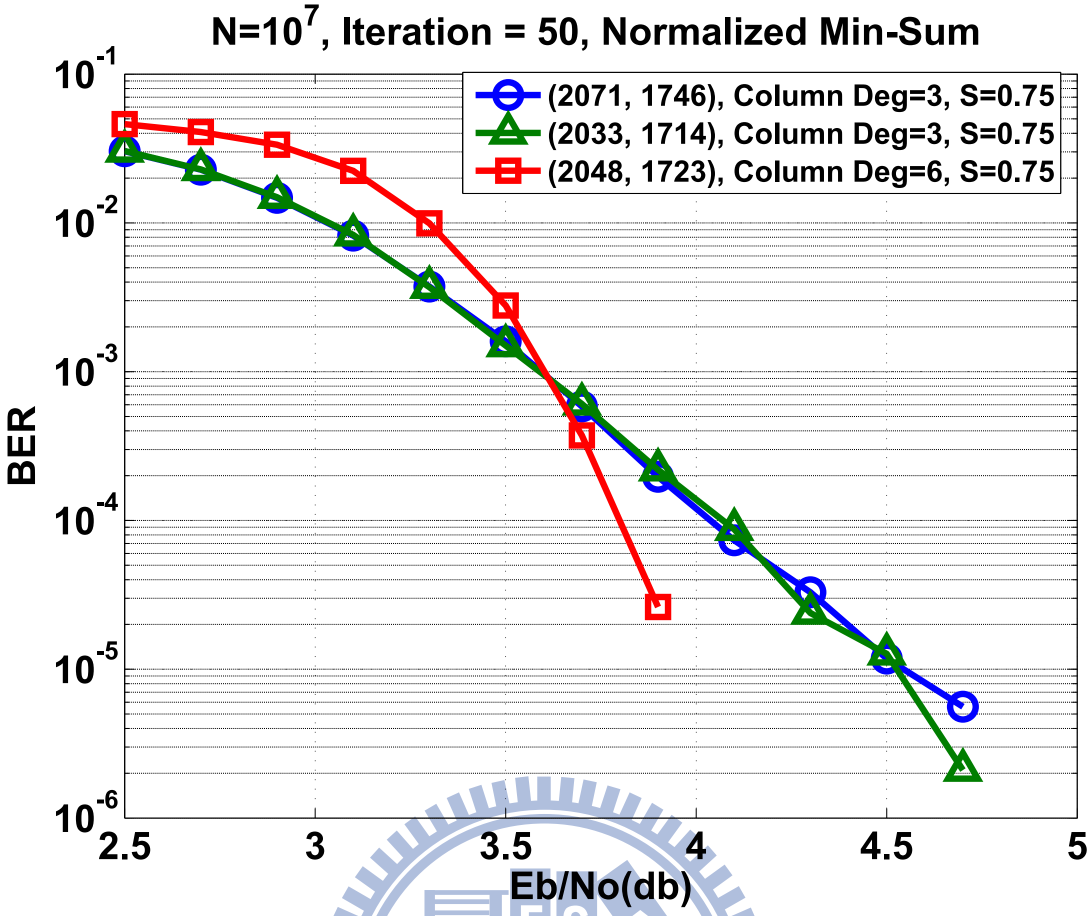


Figure 3.5: Performance of LDPC code with different column degree.

In Fig 3.6, improvement of performance in waterfall region from higher column degree is not clear. Since codeword length is very long, the improvement is expected to appear in deeper Bit Error Rate region. Software computation is not fast enough to investigate the error floor. FPGA simulation will be done in the future. Error correcting code applied on NAND flash memory system requires high code rate and no performance degradation down to bit error rate near 10^{-12}. Therefore, a higher column degree LDPC code with no cycle-4 is preferred. The proposed LDPC code in this thesis is (9153, 8256), with column degree 8 and no cycle-4.

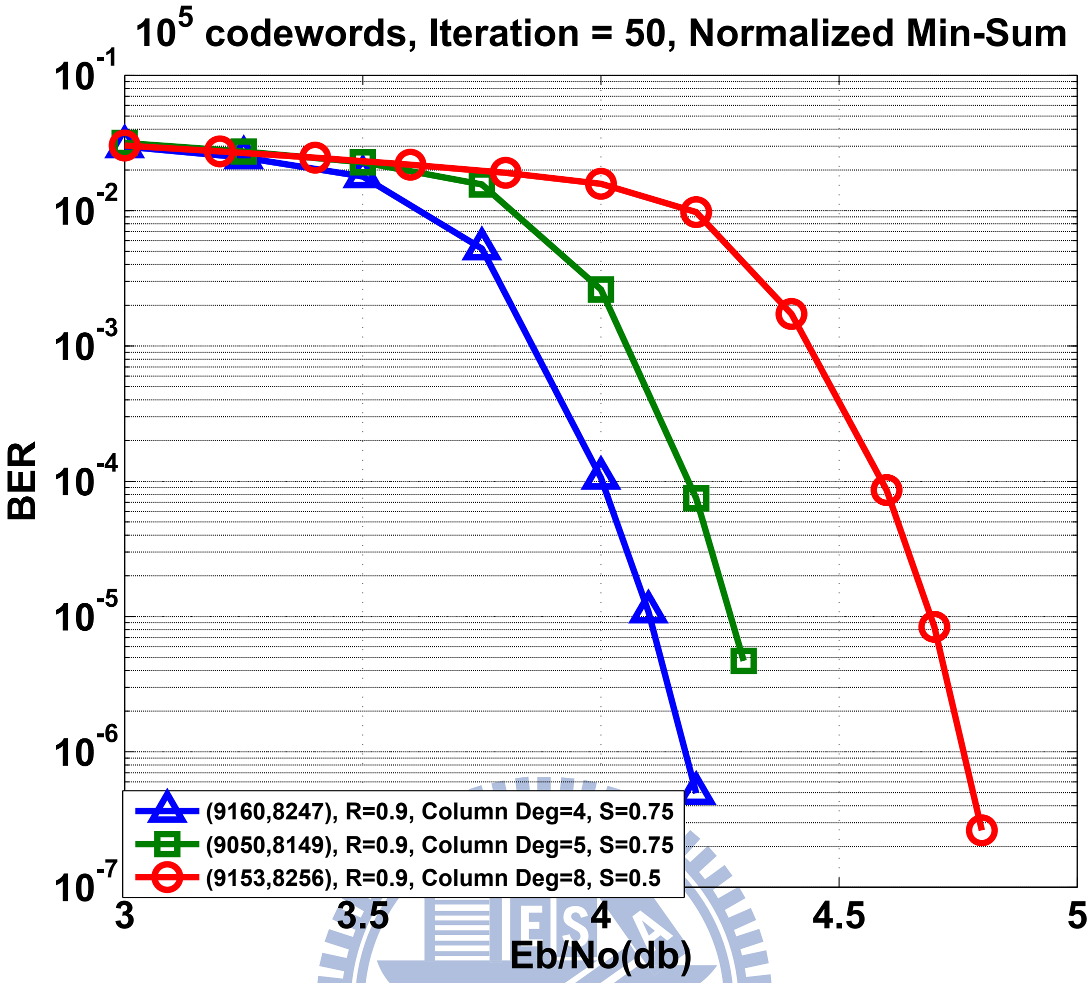


Figure 3.6: Performance of LDPC code with different column degree.

3.3 Code Construction

3.3.1 Permutation Matrix Algorithm

Permutation matrix [14] algorithm is a code construction of QC LDPC code. The parity check matrix H of QC code is composed of many sub-matrixes. Each sub-matrix will be an Identity matrix or cyclic shift of an Identity matrix. An example of QC code is demonstrated in Fig 3.7. The number inside a sub-matrix represents the amount of cyclic shift.

Cycle-4 causes performance degradation and this code construction can avoid any cycle-4. Algorithm of code construction is described in [14]. In this thesis, we provide another view of this algorithm. There are 3 parameters to be decided: row degree (d_c),

$$\mathbf{H} = \left(\begin{array}{cccc:cccc:cccc} 1&0&0&0&0&0&0&1&0&0&1&0\\ 0&1&0&0&1&0&0&0&0&0&0&1\\ 0&0&1&0&0&1&0&0&1&0&0&0\\ 0&0&0&1&0&0&1&0&0&1&0&0\\ \hdashline 1&0&0&0&0&1&0&0&0&0&0&1\\ 0&1&0&0&0&0&1&0&1&0&0&0\\ 0&0&1&0&0&0&0&1&0&1&0&0\\ 0&0&0&1&1&0&0&0&0&0&1&0 \end{array}\right) \qquad \mathbf{H} = \begin{bmatrix} 0&1&2\\ 0&3&1 \end{bmatrix}$$

Figure 3.7: An example of QC LDPC code, $d_c = 3$, $d_v = 2$ and $p = 4$.

variable node degree (d_v) and size of sub-matrix (p). Row degree determines the number of sub-matrix in one sub-matrix row. And variable node degree determines the number of sub-matrix in one sub-matrix column.

Another view of permutation matrix algorithm:

Let's $S_{i,j}$ represents the amount of cyclic shift in sub-matrix of i^{th} sub-matrix row and j^{th} sub-matrix column. d_c represents row degree. d_v represents variable node degree. And p represents size of sub-matrix and must be a prime number.

1.Initialization :

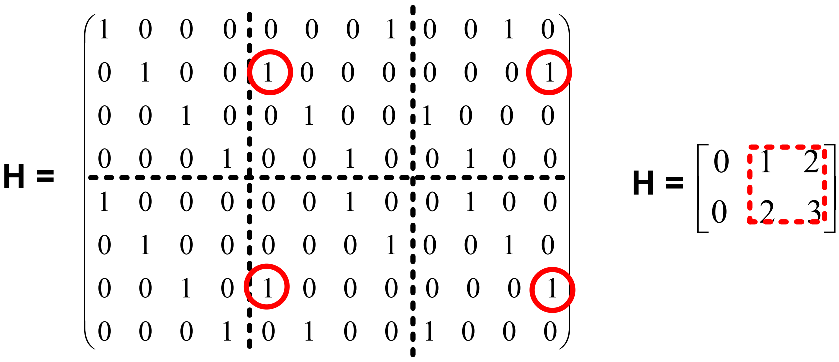
$S_{0,j} = j, 0 \leq j \leq d_c - 1$

2.Completion of the remaining $S_{i,j}$:

$S_{i,j} = (j + (j+1) \cdot i) \;\; mod \;\; p, 0 \leq j \leq d_c - 1, 0 \leq i \leq d_v - 1$

Fig. 3.8 demonstrates the condition that cycle-4 occurs. For any 4 numbers in a square (the red dash box), if the difference between the cyclic shift amount in one sub-matrix column, is equal to the difference between the cyclic shift amount in other sub-matrix column, cycle-4 is formed. For example in Fig. 3.8, the difference between 1 and 2 is equal to the difference between 2 and 3.

The following proof proves that the code construction will not produce any cycle-4. The matrix in Fig. 3.9 is a parity check matrix H. A small square represents a sub-matrix.



$$\mathbf{H} = \left(\begin{array}{cccc:cccc:cccc} 1&0&0&0&0&0&0&1&0&0&1&0\\ 0&1&0&0&\textcircled{1}&0&0&0&0&0&0&\textcircled{1}\\ 0&0&1&0&0&1&0&0&1&0&0&0\\ 0&0&0&1&0&0&1&0&0&1&0&0\\ \hdashline 1&0&0&0&0&0&1&0&0&1&0&0\\ 0&1&0&0&0&0&0&1&0&0&1&0\\ 0&0&1&0&\textcircled{1}&0&0&0&0&0&0&\textcircled{1}\\ 0&0&0&1&0&1&0&0&1&0&0&0 \end{array}\right) \qquad \mathbf{H} = \begin{bmatrix} 0 & 1 & 2 \\ 0 & 2 & 3 \end{bmatrix}$$

Figure 3.8: Demonstration of cycle-4.

The number in small square represents the cyclic shift amount of that sub-matrix. There are n sub-matrix in one sub-matrix row and m sub-matrix in one sub-matrix column. p is the size of a sub-matrix.

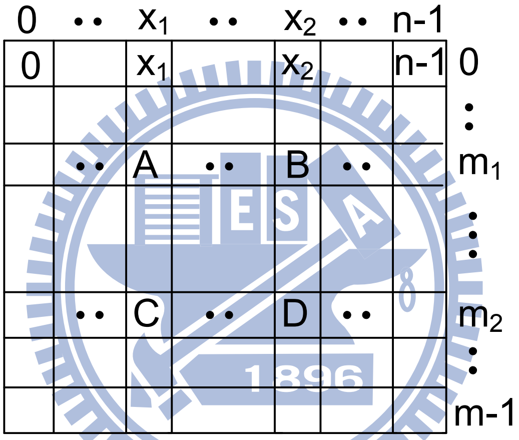


Figure 3.9: Pariyt check matrix H.

$$\begin{aligned} C &= [A + (x_1 + 1)m'] \ \ mod \ \ p \\ D &= [B + (x_2 + 1)m'] \ \ mod \ \ p \end{aligned} \tag{3.7}$$

$$\begin{aligned} C - A &= [(x_1 + 1)m'] \ \ mod \ \ p \\ D - B &= [(x_2 + 1)m'] \ \ mod \ \ p \end{aligned} \tag{3.8}$$

$where\ m' = m_2 - m_1;\ 0 \leq x_1 < x_2 \leq n-1;\ 0 \leq m_1 < m_2 \leq m-1;\ n, m \leq p$

Since p is a prime number and, $x_1 < x_2 \leq n - 1$ and $n \leq p$, $(C - A)$ will never be equal to $(D - B)$. Therefore, no cycle-4 is formed.

3.3.2 Code Performance

The proposed LDPC code in this thesis is (9153, 8256) with code rate 0.9. Column degree is 8. The size of a sub-matrix is 113 and decoding algorihtm is Normalized Min-Sum. S represents scaling factor and number of iteration is 40 .Fig. 3.10 shows its performance.

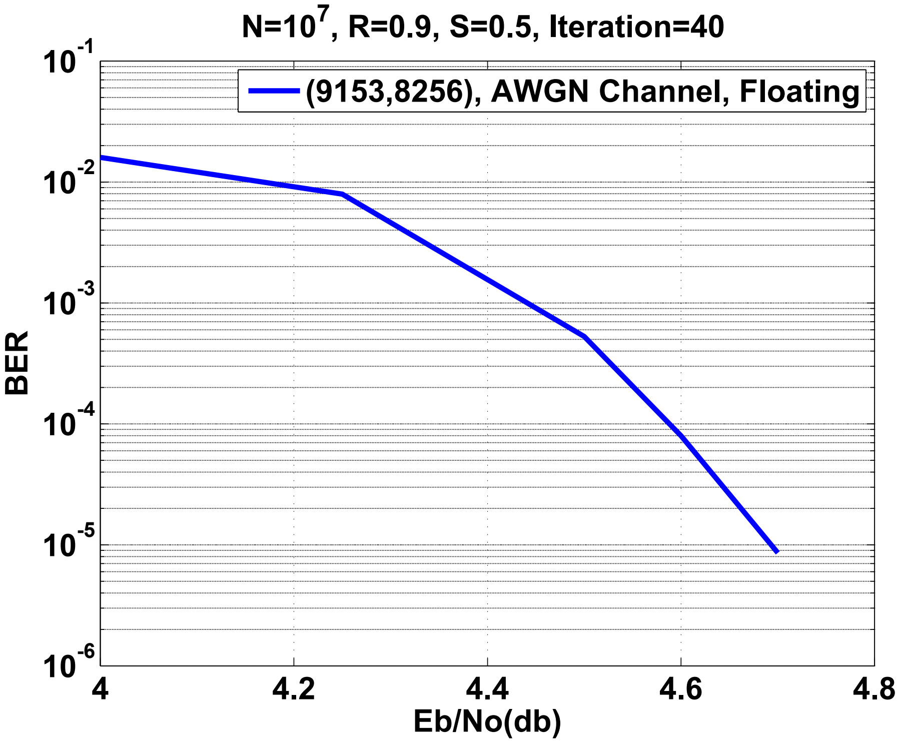


Figure 3.10: Performance of (9153, 8256) LDPC code.

Chapter 4

LDPC Decoder Architecture

4.1 Single Pipelined Architecture for VSS Algorithm

Details of variable-node-centric sequential scheduling algorithm(VSS) [10] is introduced in previous section. Hardware architecture will be fully explained in this section.

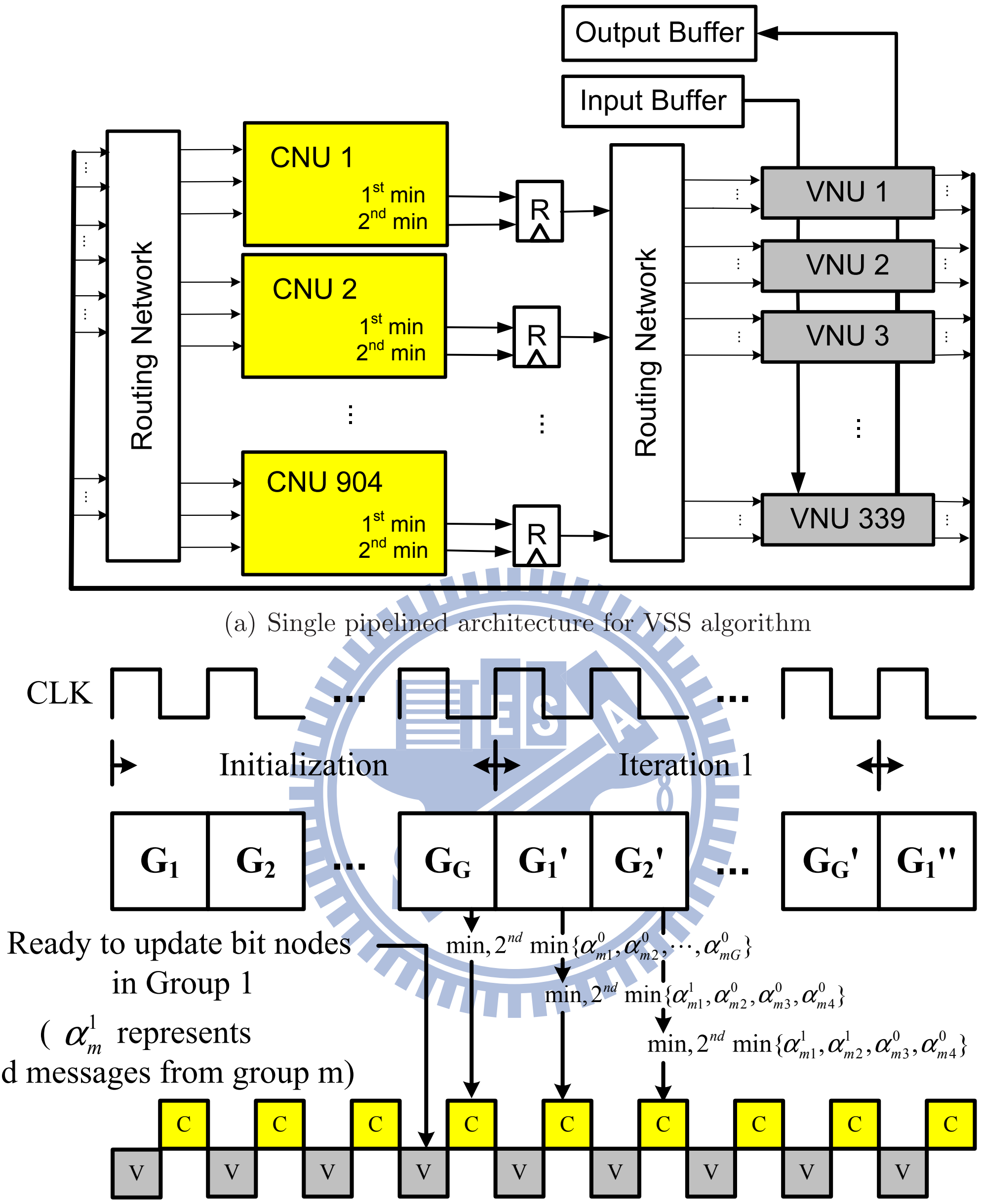
The entire decoder depicted in Fig. 4.1(a) is composed of fully-parallel CNUs and partial-parallel VNUs. Variable nodes are divided into 27 groups ($G = 27$). There are 904 Check Node Units (CNU) and 339 Variable Node Units (VNU). Let $\alpha^i_{g,m}$ denotes the sorted messages (1^{st} min, 2^{nd} min and indices) from variable nodes in the g^{th} group to m^{th} check node at i^{th} iteration, which is:

$$\alpha^i_{g,m} = \min_{n' \in N(m)\backslash n, g\cdot N_G \leq n' \leq (g+1)\cdot N_G - 1} \left\{ \left| z^i_{mn'} \right| \right\} \tag{4.1}$$

Then the magnitude part of check node to variable node message in equation 3.4 could be computed by the following equation:

$$\left| \epsilon^i_{mn} \right| = \min \left\{ \left\{ \alpha^i_{j,m} \right\}_{j<g}, \alpha^i_{g,m}, \left\{ \alpha^{i-1}_{k,m} \right\}_{k>g} \right\} \tag{4.2}$$

Fig. 4.1(b) demonstrates the timing diagram of proposed decoder. G initialization cycles are required to calculate $\alpha^0_{g,m}$ for $0 \leq g \leq G-1$. Since only one subgroup of the message z^i_{mn} is updated in each cycle of one iteration, the main operation of CNU could be simplified to calculate $\alpha^i_{g,m}$ (local sorting) in each cycle and then perform global sorting like equation 4.2. In single pipelined architecture, only messages $\alpha^i_{g,m}$ and ϵ^i_{mn} are stored, while the variable node to check node message z^i_{mn} is on-the-fly calculated. The CNU could be updated immediately after VNU's operations in VSS approach and no variable to check node message need to be stored.


Output Buffer
Input Buffer
Routing Network
CNU 1
1st min
2nd min
CNU 2
1st min
2nd min
CNU 904
1st min
2nd min
R
R
R
Routing Network
VNU 1
VNU 2
VNU 3
VNU 339
CLK
Initialization
Iteration 1
G1
G2
GG
G1'
G2'
GG'
G1''
Ready to update bit nodes in Group 1
(α^1_m represents sorted messages from group m)
$\min, 2^{nd} \min\{\alpha^0_{m1}, \alpha^0_{m2}, \cdots, \alpha^0_{mG}\}$
$\min, 2^{nd} \min\{\alpha^1_{m1}, \alpha^0_{m2}, \alpha^0_{m3}, \alpha^0_{m4}\}$
$\min, 2^{nd} \min\{\alpha^1_{m1}, \alpha^1_{m2}, \alpha^0_{m3}, \alpha^0_{m4}\}$
C
V

(a) Single pipelined architecture for VSS algorithm

(b) Timing Schedule

Figure 4.1: Architecture and scheduling for VSS algorithm.

4.2 Check Node Unit (CNU)

This section presents detail CNU architecture based on VSS scheduling. The CNU architecture is further optimized to reduce storage requirement and the number of sorters. Different CNU architectures will affect the convergence speed and performance which will be discussed in the next chapter. The messages sent from VNU are converted from two's complement format to sign-magnitude format for efficient computation of CNU. Therefore, the operation of check node to variable node update could be divided into magnitude part and sign part. For our proposed LDPC codes with row degree 81, the VSS approach with $G = 27$, the number of messages need to be computed in each CNU group is 3.

4.2.1 Accumulative Sorter

Fig. 4.2 illustrates the magnitude part of CNU, which is an accumulative sorter composed of a local sorter and a global sorter. The local sorter is used to find the local 1^{st} min and 2^{nd} min values in each subgroups, and global 1^{st} min and 2^{nd} min values of a row will be found by a global sorter. $G - 1$ registers are required to store local 1^{st} min from different group. And local 2^{nd} min is the same. The global sorter has $27 \times 2 = 54$ inputs in total. Number of registers will be increased if G becomes larger. This increases the number of inputs to global sorter and the critical path.

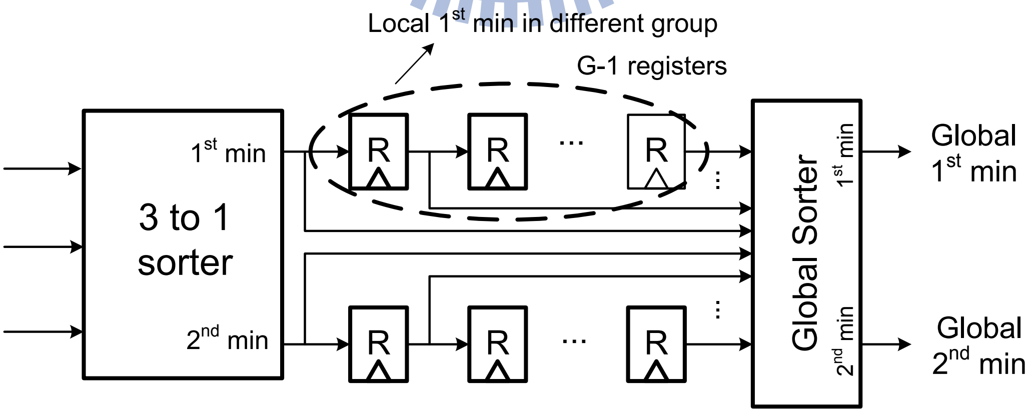


Figure 4.2: Conventional accumulative sorter.

A demonstration is provided in Fig 4.3. We assume row degree = 9 and number of group $(G) = 3$. R_{1stmin} and R_{2ndmin} represent the local 1^{st} min and 2^{nd} min of each group respectively. The value in registers is reset to infinity before initialization. Since G = 3, there are three variable nodes in each group and they provide new values to the sorter every cycle. Local 1^{st} min and 2^{nd} min will be obtained and stored in the registers. The values in each register is shifted to the right. The global sorter chooses the global 1^{st} min and 2^{nd} min from these 7 values (3 new inputs, local 1^{st} min and 2^{nd} min from 2 local groups). The red number represents the global 1^{st} min in that cycle.

	Initialization									1^{st} Iteration		
	Group 1			Group 2			Group 3			Group 1		
$R_{1st\ min}$	∞	∞		0.1	∞		0.4	0.1		0.7	0.4	
$R_{2nd\ min}$	∞	∞		0.2	∞		0.5	0.2		0.8	0.5	
Inputs	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	0.5	0.6	0.7

Figure 4.3: Demonstration of conventional accumulative sorter.

4.2.2 Accumulative Sorter without 2^{nd} minimum value

To reduce storage memory, local 2^{nd} min values and global 2^{nd} min values are not stored. The local 1^{st} min value is the minimum value from $G - 1$ groups. And global 2^{nd} min value is taken from local 1^{st} min value directly. This may cause some performance loss.

When local 1^{st} min value is smaller than global 1^{st} min value, global 1^{st} min value is replaced by local 1^{st} min value. Then value stored in local 1^{st} min register should be set to a maximum value. Local sorter starts to find the new local 1^{st} min value.

Conventionally, when the current updating group is the same as the group that global 1^{st} min value comes, global 2^{nd} min value should be sent to the variable nodes. Since global 2^{nd} min value is not stored, global 1^{st} min value is updated by local 1^{st} min value and sent to bit nodes as global 2^{nd} min value. Thus, both global 1^{st} min and 2^{nd} min value are equal to local 1^{st} min value at this cycle.

There are some methods for compensation on global 2^{nd} min such as multipling or adding a scalar to original global 1^{st} min. But these methods only provide limited improvement. Since local 1^{st} min value from $G-1$ groups contains updated information, taking local 1^{st} min value as global 2^{nd} min value can provide better improvement.

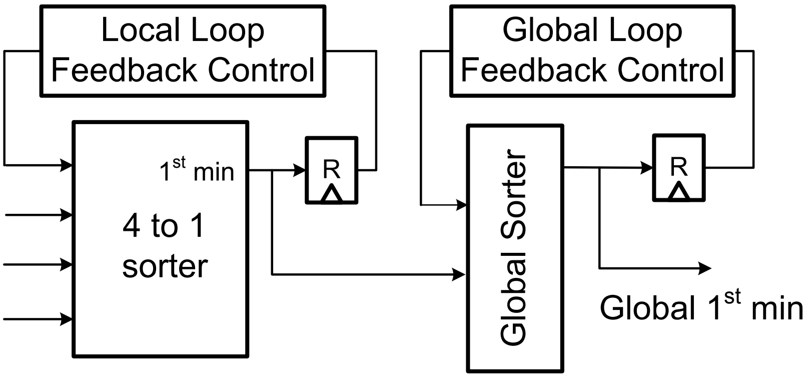
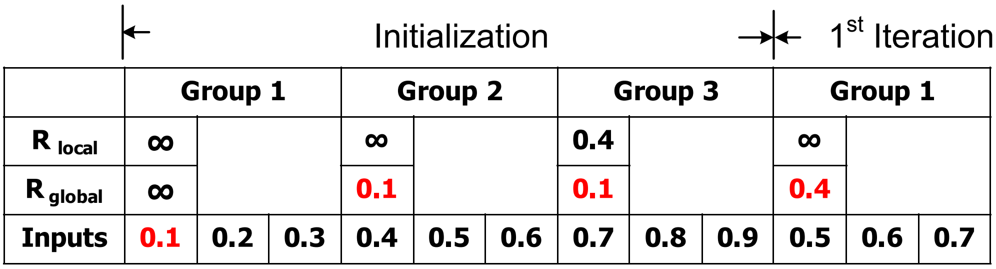


Figure 4.4: Accumulative sorter w/o 2nd min.

A demonstration is provided in Fig 4.4. We assume row degree = 9 and number of group $(G) = 3$. R_{local} represents the local 1^{st} min and R_{global} represents the global 1^{st} min. Number of registers is indepentent of G. Number of inputs to local sorter is equal to $N/G+1$ and number of inputs to local sorter is equal to 2. The new global 1^{st} min comes from the three new inputs, local 1^{st} min and previous gloabl 1^{st} min. The red number represents the global 1^{st} min in that cycle. After the initialization, the global 1^{st} min stored in register comes from gorup 1. At 4^{th} cycle (group 1 update of 1^{st} iteration), there are new valus from group 1 and the global 1^{st} min in register should be cleared. Threrfore, global 1^{st} min is replaced by local 1^{st} min.


Initialization
1st Iteration

	Group 1			Group 2			Group 3			Group 1		
R_{local}	∞			∞			0.4			∞		
R_{global}	∞			0.1			0.1			0.4		
Inputs	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	0.5	0.6	0.7

Figure 4.5: Demonstration of accumulative sorter w/o 2nd min.

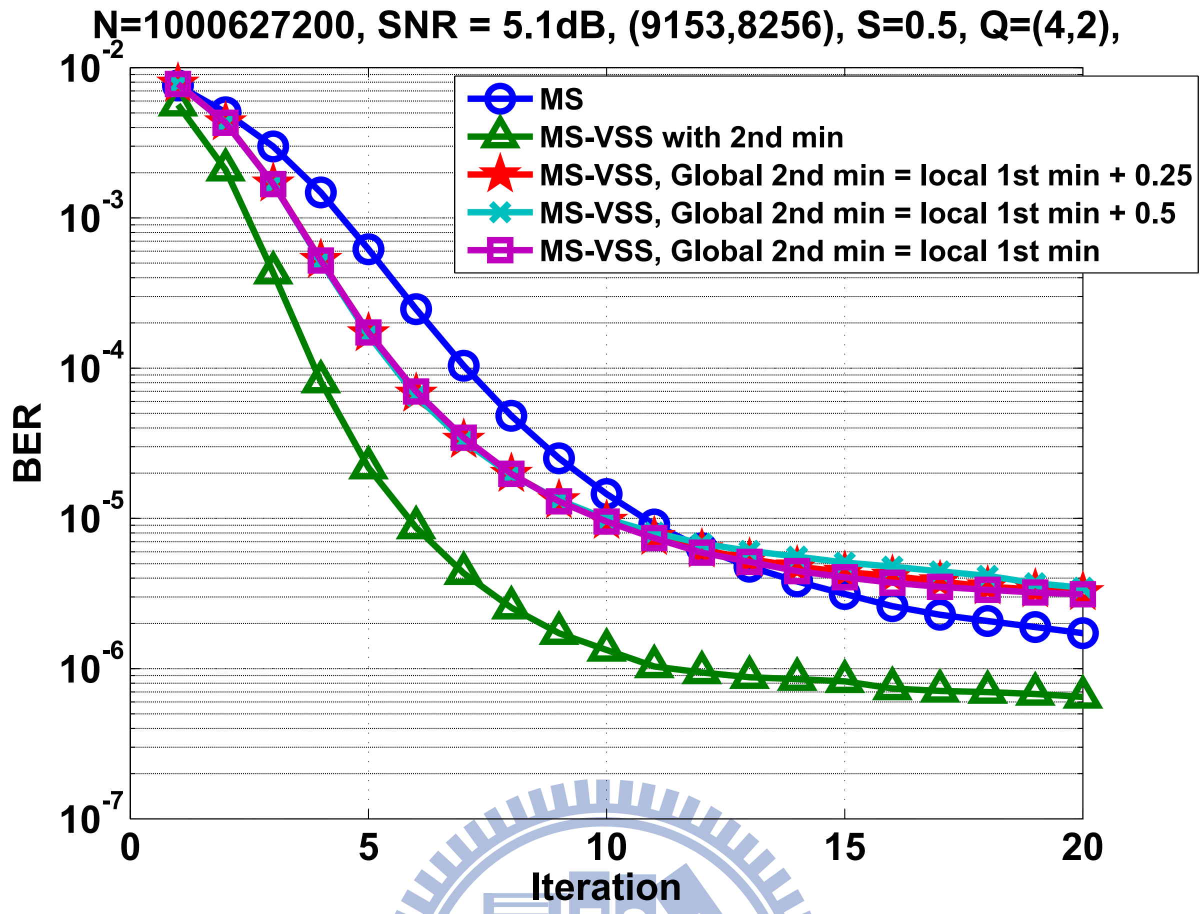


Figure 4.6: Performance of (9153, 8256) LDPC code with different global 2^{nd} min compensation, MS - MinSum, MS-VSS - MinSum with variable-node-centric sequential scheduling.

Figure 4.6 shows the performance of (9153, 8256) LDPC code with different global 2^{nd} min compensation. MinSum with VSS algorithm has a faster convergence speed than MinSum algorithm. If global 2^{nd} min is not stored, there is some performance degradation and the convergence speed decreases. But reduced storage memory version is preferred for the FPGA simulation. Compensation on global 2^{nd} min (local 1^{st} min) does not provide any improvement. Thus, no compensation on global 2^{nd} min is preferred. BER decreases slowly after 10th iteration due to the absence of original global 2^{nd} min value. Therefore. number of iteration is decided to be 10. Throughput can be further increased if early termination is applied. In addition, 2^{nd} min can be preserved if better performance is desired.

4.3 Varible Node Unit (VNU)

Fig. 4.7 shows the architecture of a VNU. SM to TC represents sign-magnitude to two's-complement conversion, and TC to SM represent two's-complement to sign-magnitude conversion. Registers are corresponding to different channel values in the different groups. Since $G = 27$, there are 27 2-bits registers to store channel values in one VNU. The bit width of messages passing between CNU and VNU is 4. The variable node degree is 8. Thus, number of inputs of adder is 9. 2 bits channel value is mapped to 4 bits value by non-linear quantization. More details of non-linear quantization will be discussed in next chapter.

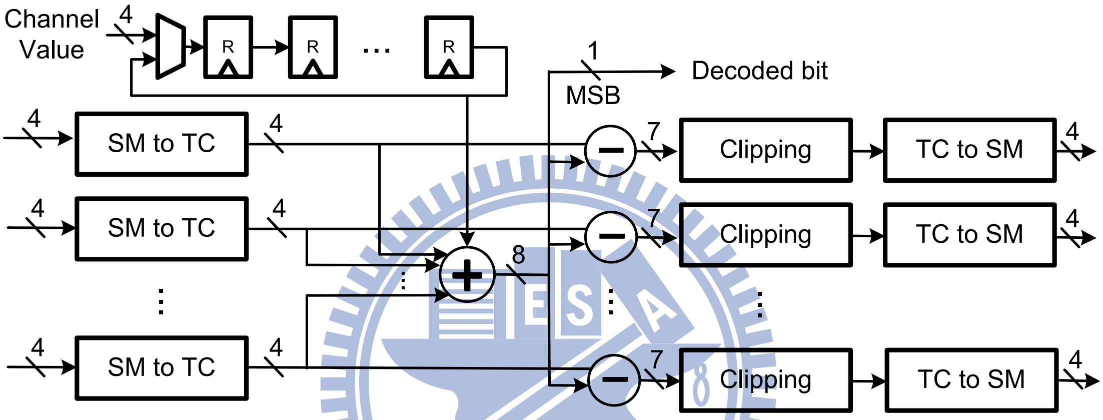


Figure 4.7: Variable node unit architecture.

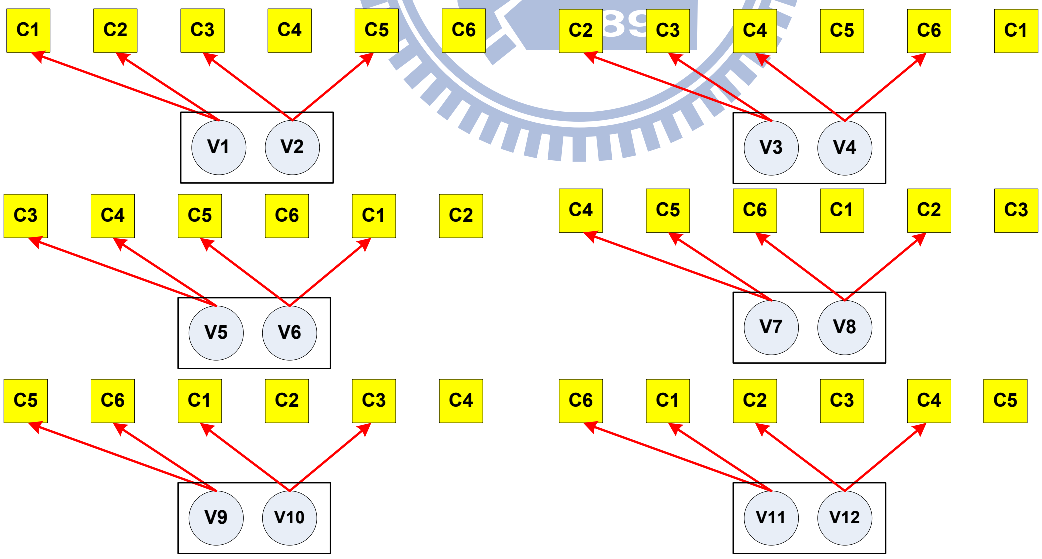
4.4 Shifting Network

High compexity of routing network between Check Node Units (CNU) and Varible Node Units (VNU), is the main difficulty for hardware implementation of LDPC code. Shifting Network [17] [18] [19] [20] has been proposed to reduce the routing complexity. There are two routing networks between CNU and VNU. One is the direction from CNUs to VNUs, while another one is the direction form VNUs to CNUs.

The shifting network of LDPC code, which is constructed by permutation matrix algorithm, can be simplified. The wire connection from CNUs to VNUs is fixed and no shifting network is needed. But messages of each CNU are shifted between CNUs. The idea is explained in Fig 4.8.

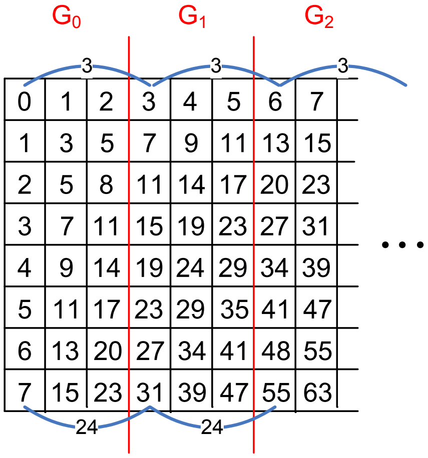
$$H = \left[\begin{array}{cc|cc|cc|cc|cc|cc}
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0
\end{array}\right]$$

(a) Parity Check Matrix of a LDPC code, with variables divided into 6 groups



(b) Messages shifted between CNUs

Figure 4.8: Illusion of messages shifted between CNUs.



0	1	2	3	4	5	6	7
1	3	5	7	9	11	13	15
2	5	8	11	14	17	20	23
3	7	11	15	19	23	27	31
4	9	14	19	24	29	34	39
5	11	17	23	29	35	41	47
6	13	20	27	34	41	48	55
7	15	23	31	39	47	55	63

Figure 4.9: Parity Check Matrix of (9153,8256) LDPC code.

Fig. 4.9 shows the cyclic shift amount of some sub-matrices in parity check matrix of (9153,8256) LDPC code. Since $G = 27$, 3 sub-martrices are processed in each decoding cycle. The difference between cyclic shift amount of each group is a constant. Thus, messages are shifted between CNUs after each decoding cycle and routing network can be eliminated.

4.5 Comparison with Conventional Architectures

For accumulative sorter in Fig. 4.2, larger subgroup number G will result in fewer inputs of local sorter but more inputs of global sorter. And the number of storage memory for 1^{st} min, 2^{nd} min, and index values will increase. In addition, the critical path will be shorter when G is larger becasue sorter is smaller. In traditional two-stage pipelined architecture, both check node to variable node message and variable node to check node message are kept in registers or memory. Assume the bit-width w of messages is 4 and variable node degree is d_v, then the required memory size (or registers) is as follows:

Conventional Min-Sum two-stage pipelined architecture:

$$\begin{aligned} & Reg_{VNU} + Reg_{CNU} \\ & = N \cdot d_v \cdot w + m \cdot \left(1^{st}min + 2^{nd}min + Index + Sign\right) \\ & = 9153 \cdot 8 \cdot 4 \ \ bits + 904 \cdot (3 + 3 + 7 + 81) \\ & = 377872 \ \ bits \end{aligned} \tag{4.3}$$

VSS architecture with conventional accumulative sorter(Fig. 4.2):

$$\begin{aligned} & Reg_{VNU} + Reg_{CNU} \\ & = 0 + m \cdot (local \ \ 1^{st}min + local \ \ 2^{nd}min + \\ & \ global \ \ 1^{st}min + global \ \ 2^{nd}min + Index + Sign) \\ & = 904 \cdot (3 \cdot 26 + 3 \cdot 26 + 3 + 3 + 7 + 81) \\ & = 226000 \ \ bits \end{aligned} \tag{4.4}$$

Proposed VSS architecture with No 2^{nd} min accumulative sorter(Fig. 4.4):

$$\begin{aligned}
&Reg_{VNU} + Reg_{CNU} \\
&= 0 + m \cdot \big(local\ \ 1^{st}min + global\ \ 1^{st}min + Index + Sign\big) \\
&= 904 \cdot (3 + 3 + 5 \cdot 2 + 81) \\
&= 87688\ \ bits
\end{aligned} \tag{4.5}$$

Compared to the conventional Min-Sum two-stage pipelined architecture, proposed architecture reduces 76.8% registers. Compared to the VSS architecture with conventional accumulative sorter, proposed architecture reduces 61.2% registers with some performance loss. Since $G = 27$, the reduction of combinational circuit of VNU is approximately 96%.

Chapter 5

Simulation and Implementation Result

5.1 Quantization

Belief Propagation (BP) is a probability-based message passing algorithm. When soft input is available, LDPC code can provide powerful correcting ability. LDPC code with 2-bit soft input can outperform BCH code under same code rate. Additive White Gaussian Noise (AWGN) channel with Binary Phase Shift Keying Modulation (BPSK) are used for demonstration and simulation. We assume that data '0' is mapped to '1' and data '1' is mapped to '-1'. 2-bit quantization represents 4 levels. A bit with channel value near 0 has a high probability to be an error bit. Therefore, a non-linear quantization is preferred. We make a threshold f to divide channel value into 4 levels.

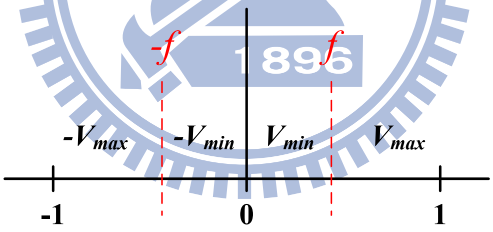


Figure 5.1: 2 bits (4 levels) non-linear quantization.

Fig. 5.2 shows the performance of LDPC code with different parameters $f,\ V_{min}\ and\ V_{max}$. The bit width of Input LLR after non-linear quantization and messages passing between CNUs and VNUs in decoder are 4 bits. Decoding algorithm is Normalized Min-Sum algorithm with scaling factor = 0.5.

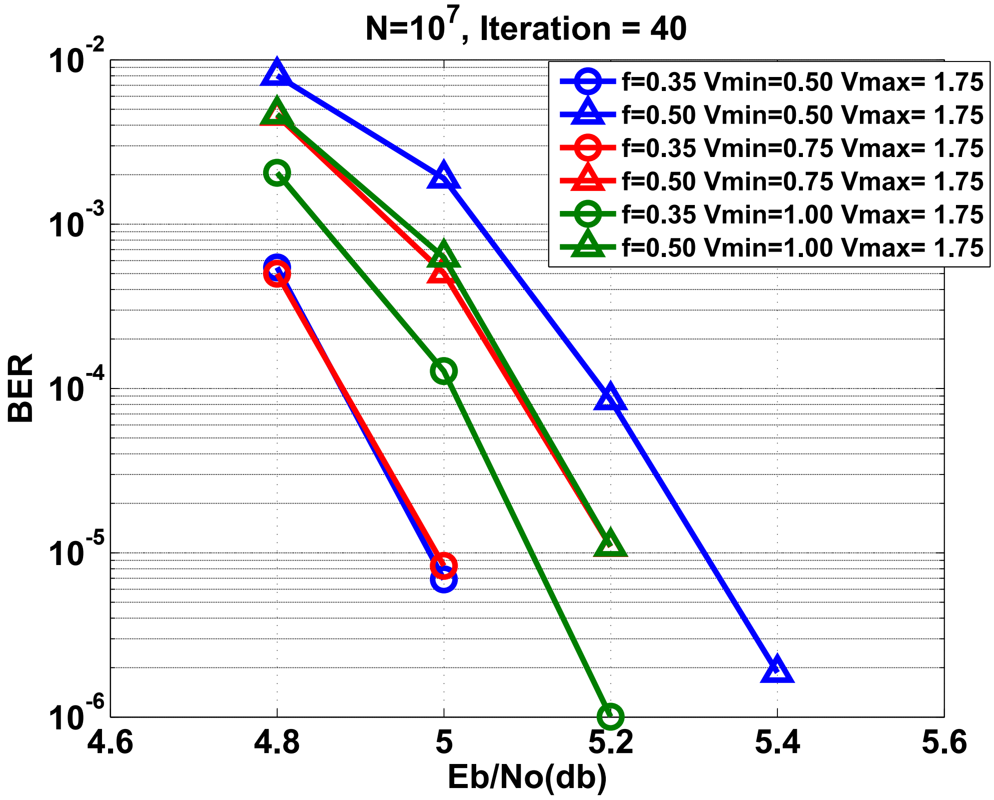


Figure 5.2: Performance of (9153, 8256) (Column deg = 8) LDPC code with different parameters.

Parameter $f = 0.35$, $V_{min} = 0.5$ and $V_{max} = 1.75$ provides the best performance.

In Fig. 5.3, the performance loss between floating input and 2 bits non-linear input quantization is 0.3dB. 2 bits non-linear input quantization can provides better performance than 4-bit linear input quantization. As more-bits input information requires more READ on NAND flash cell, latency of reading data will increase. Therefore, 2 bits non-linear input quantization is chosen.

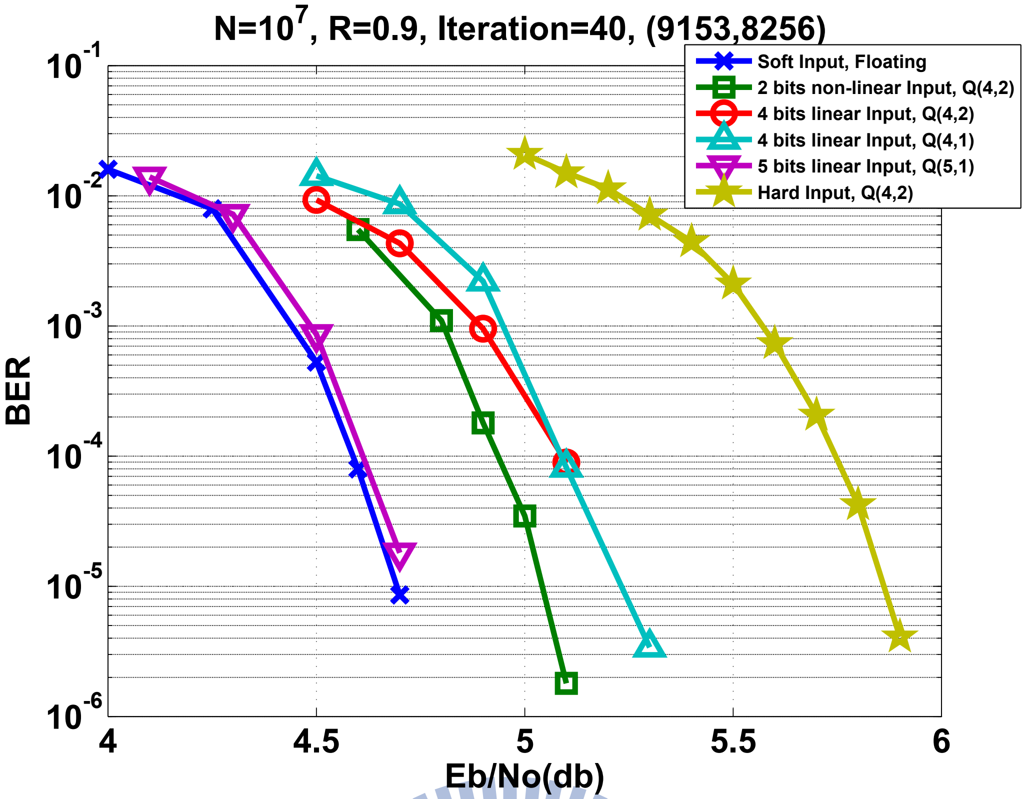


Figure 5.3: Performance of LDPC code with different input quantization.

5.2 Performance

In Fig. 5.4, there is 0.7dB coding gain of 2-bit non-linear soft input LDPC code over BCH code at BER=10^{-4}. 2-bit non-linear soft input LDPC code has a great potential to replace BCH code for NAND flash memory system. The simulation parameters of LDPC code are 4-bit quantization (2-bit integer and 2-bit decimal fraction), with scaling factor 0.5. The bit width of messages passing between CNU and VNU is 4.

Without storing global 2^{nd} min value introdueces 0.1dB performance loss. But Variable-node-centric Sequential Scheduling (VSS) architecture with no 2^{nd} min value reduces 76.6% registers and approximately 96% combinational circuit of VNU.

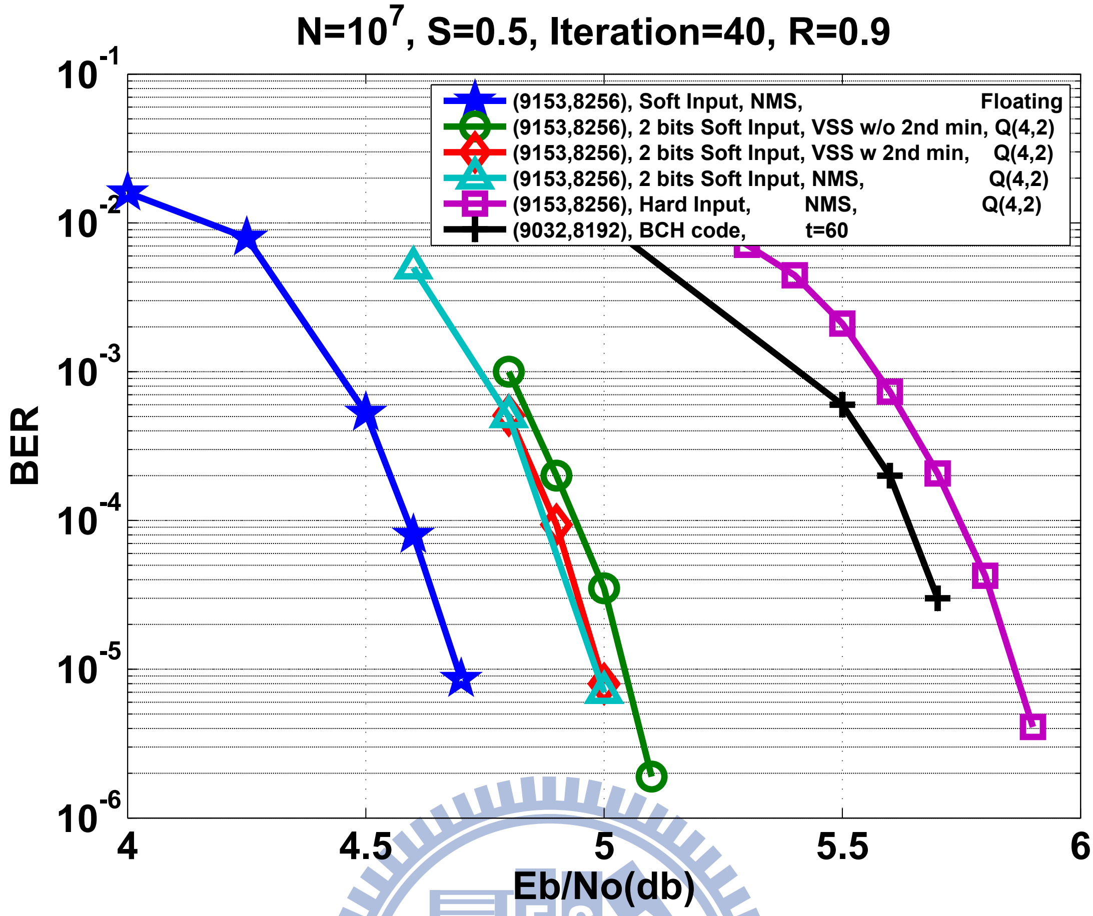


Figure 5.4: Performance comparison, Iteration = 40.

5.3 Throughput

Gate count and critical path of CNU and VNU after synthesize is listed in Table. 5.1. The critical path of CNU + VNU is 5ns. We assume that the critical path of control circuit is 2ns. Therefore the clock cycle is 7ns. The LDPC decoder can operate at a frequency of 125MHz.

Table 5.1: Synthesis result of CNU and VNU with technology UMC90.

	CNU(sign bit register is not included)	VNU
Gate count	225	620
Critical path (ns)	2	3

Number of iteration is 10 and clock frequency in Place and Route is 100Mhz.

$$\begin{aligned} Throughput &= \frac{Information\ \ length}{Cycles\ \ per\ \ iteration \cdot (Number\ \ of\ \ iteration + 1) \cdot Cycle\ \ length} \\ &= \frac{8256}{27 \cdot (10+1) \cdot 10ns} \\ &\approx 2.78Gbps \end{aligned}$$

5.4 Implementation Results

Table 5.2: Summary of implementation result (Place and Route).

	Proposed LDPC Decoder
Technology	UMC 90nm 1P9M
Code Spec	(9153,8256)
Code Rate	0.9
Row Degree	81
Column Degree	8
Algorithm	Variable-node-centric Sequential Scheduling
Area	4.82 mm^2 (No IO Pad)
Gate Count	1100k
Iteration	10
Input Quantization	2 bits
Clock Frequency	100MHz
Maximum Throughput	2.78 Gbits/s
Power	437 mW

Table 5.2 shows the postlayout result. Gate Count after synthesis is 1100k and Core area is $4.82mm^2$ without IO pad. Using 90nm CMOS technology, the maximum throughput can achieve 2.78 Gbps under operating frequency of 100Mhz with 10 iterations. Power consumption is 437mW.

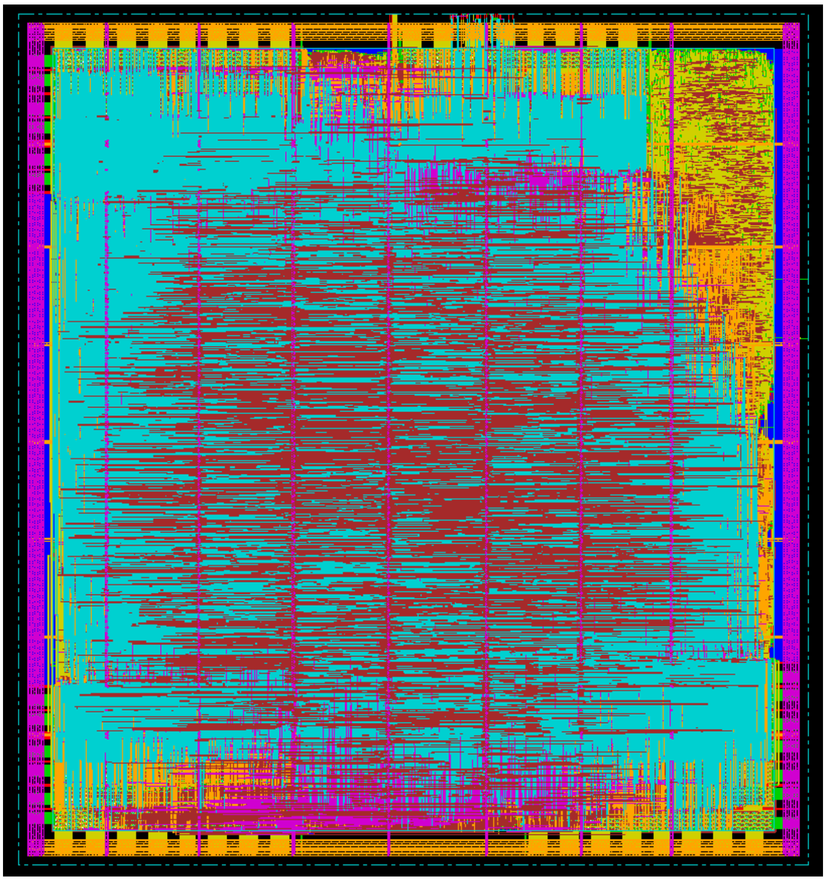


Figure 5.5: Layout of Place and Route.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

This thesis proposes a (9153, 8256) LDPC code with code rate 0.9 for NAND flash memory system. (9153,8256) LDPC code is constructed by permutation matrix algorithm, with column degree 8. Simulations show that LDPC code with 2-bit soft input can outperform BCH code under same code rate. Therefore, LDPC code is a good candidate to replace BCH code in the next generation standard.

High code rate LDPC code introduces high row degree. This makes implementation difficult due to the large number of inputs to sorter, and the routing complexity also increases. Variable-node-centric sequential scheduling (VSS) is a good solution to this problem. Variable nodes are divided into G groups. Check node update procedures are processed in G cycles, reducing the number of inputs to sorter. CNU is further modified to reduce the hardware cost. Compared to the conventional Min-Sum two-stage pipelined architecture, it saves approximately 96% combination circuits of VNU and reduces 76.8% registers. The maximum throughput can achieve 2.78 Gbps under operating frequency of 100Mhz with 10 iterations, using 90nm CMOS technology.

6.2 Future Work

Flash memory system requires Bit Error Rate (BER) down to 10^{-12}. And this thesis proposes a high column degree LDPC code in order to suppress error floor. Simulation of BER down to 10^{-12} consumes years on computer. Therefore, we will do simulation on FPGA to investigate the performance of LDPC code down to 10^{-12} in the future.

There is no standard flash memory channel for any simulation. Therefore, a standard flash memory channel is desired if we want to compare performances of different error correcting code on flash memory. It is a new challenge and more details about flash memory will be studied.

References

[1] D. M. Greg Atwood, Al Fazio and B. Reaves, “Intel StrataFlashTM Memory Technology Overview,” *Intel Technology Journal*, pp. 1–8, 4th Quarter 1997.

[2] R.C.Bose and D.K.Ray-Chaudhuri, “On a class of error-correcting binary group codes,” *Inform. and Contr*, vol. 3, pp. 68–79, March 1960.

[3] A. Hocquenghem, “Codes correcterus d’erreurs,” *Chiffres*, vol. 2, pp. 117–156, September 1959.

[4] W. J. ReidIII, L. L. Joiner, and J. J. Komo, “Soft Decision Decoding of BCH Codes Using Error Magnitudes,” *IEEE Int. Symp. on Info. Theory*, p. 303, June 1997.

[5] Y. M. Lin, C. L. Chen, H. C. Chang, , and C. Y. Lee, “A 26.9K 314.5Mbps Soft (32400, 32208) BCH Decoder Chip for DVB-S2 System,” in *IEEE Asian Solid-State Circuits Conference*, Nov. 2009, pp. 373–376.

[6] R.G.Gallager, “Low-Density Parity-Check Codes,” in *MA: MIT Press*, 1963.

[7] D. MacKay and R. Neal, “Near Shannon limit performance of low density parity check codes,” *Electron. Lett*, vol. 33, no. 6, pp. 457–458, March 1997.

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

[9] J. Zhang and M. Fossorier, “Shuffled iterative decoding,” *IEEE Transactions on Communications*, vol. 53, no. 2, pp. 209–213, Feb. 2005.

[10] C.-L. Chen, K.-S. Lin, H.-C. Chang, W.-C. Fang, and C.-Y. Lee, “A 11.5-Gbps LDPC Decoder Based on CP-PEG Code Construction,” in *ESSCIRC*, 2009, pp. 412–415.

[11] J. Sha, Z. Wang, M. Gao, and L. Lio, “Multi-Gb/s LDPC Code Design and Implementation,” *IEEE Transactions on VLSI Systems*, vol. 17, no. 2, pp. 262–268, Feb. 2009.

[12] T. Richardson, “Error floors of LDPC codes,” in *Proc. 41st Annu. Allerton Conf. Communications, Control and Computing*, Oct 2003, pp. 1426–1435.

[13] T. Tian, C. Jones, J. Villasenor, and R. D. Wesel, “Construction of irregular LDPC codes with low error floors,” in *Proceedings IEEE International Conference on Communications*, vol. 5, 2003, pp. 3125–3129.

[14] H. Song, V. Kumar, and B.V.K., “Low-density parity check codes for partial response channels,” *IEEE Signal Processing Magazine*, pp. 56–66, Jan. 2004.

[15] M. Fossorier, “Quasicyclic low-density parity-check codes from circulant permutation matrices,” *IEEE Trans. Inf. Theory*, vol. 50, no. 8, pp. 1785–1793, Aug 2004.

[16] *IEEE Std. 802.3an*, Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications Std., 2006.

[17] D.Oh and K.Parhi, “Area Efficient Controller Design of Barrel Shifters for Reconfigurable LDPC Decoders,” *IEEE Internatinal Symposium on Circuits and Systems*, pp. 240–243, May 2008.

[18] C.-H. Liu, C.-C. Lin, H.-C. Chang, and Y. C.-Y. Lee, “Multi-Mode Message Passing Switch Networks Applied for QC-LDPC Decoder,” *IEEE Internatinal Symposium on Circuits and Systems*, vol. 18, no. 1, pp. 85–94, Jan 2010.

[19] D.Oh and K.Parhi, “Low-Complexity Switch Network for Reconfigurable LDPC Decoders,” *IEEE Transactions on Very Large Scale Integration Systems*, pp. 752–755, May 2008.

[20] J. Lin, Z. Wang, L. Li, J. Sha, and M. Gao, “Efficient Shuffle Network Architecture and Application for WiMAX LDPC Decoders,” *IEEE Transcations on Circuits and Systems*, vol. 56, no. 3, pp. 215–219, March 2009.