

國立交通大學

電子工程學系 電子研究所碩士班

碩 士 論 文

高速低密度同位元檢查碼之解碼器設計



High-Throughput

Low-Density Parity-Check Code Decoder Designs

學生：林凱立

指導教授：李鎮宜 教授

中華民國九十四年七月

高速低密度同位元檢查碼之解碼器設計

High-Throughput

Low-Density Parity-Check Code Decoder Designs

研究生：林凱立

Student : Kai-Li Lin

指導教授：李鎮宜

Advisor : Chen-Yi Lee

國立交通大學
電子工程學系 電子研究所 碩士班
碩士論文



Submitted to Institute of Electronics
College of Electrical Engineering and Computer Science
National Chiao Tung University
in Partial Fulfillment of the Requirements
for the Degree of
Master of Science
in

Electronics Engineering

July 2005

Hsinchu, Taiwan, Republic of China

中華民國九十四年七月

高速低密度同位元檢查碼之解碼器設計

學生：林凱立

指導教授：李鎮宜 教授

國立交通大學

電子工程學系 電子研究所碩士班

摘要

在本論文中，我們提出了兩個高傳輸速度之低密度同位元檢查碼解碼器的設計。第一個設計為應用於MB-OFDM UWB系統，區塊長度為 600 之解碼器。此架構採用了對於通道資訊的資料流重新排程以及管線化來減低繞線上的擁擠程度和最長之延遲路徑。經由 0.18 μm 製程實作晶片，我們所提出的此部份平行解碼器設計，於固定 8 次迴圈的解碼模式下，可提供之最高資料傳輸速度為每秒 480Mb。第二個是基於區塊長度為 1200 之解碼器設計。為了達到更高的晶片密度及降低繞線上所造成的時間延遲，我們所提出的架構採用了一個新的資料重新排序技術，將訊息記憶體和計算單元之間的資料匯流排簡單化。經由此方法，由於晶片密度的提高，我們可大幅的縮減晶片的大小。另外，此解碼器可同時處理兩筆不同之codeword來加快傳輸速度及資料路徑的工作效率。此設計經由 0.18 μm 製程實作後，於晶片面積為 21.23mm²，固定 8 次迴圈的解碼模式下，其最大資料傳輸速度可達到每秒 3.33Gb。另外，將此設計經由 0.13 μm 製程實作後，資料傳輸速度可提升到每秒 5.92Gb，晶片面積縮小為 10.24mm²，而晶片之密度可提高至 75.4%。

High-Throughput Low-Density Parity-Check Code Decoder Designs

Student : Kai-Lin Lin

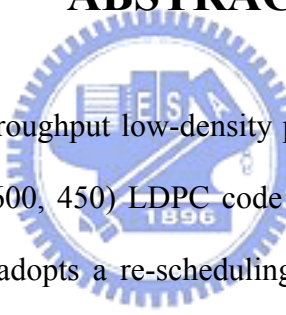
Advisor : Dr. Chen-Yi Lee

Department of Electronics Engineering

Institute of Electronics

National Chiao Tung University

ABSTRACT



In this thesis, two high-throughput low-density parity-check (LDPC) code decoders are presented. The first one is a (600, 450) LDPC code decoder applied for MB-OFDM UWB applications. The architecture adopts a re-scheduling data flow for channel values and the pipeline structure to reduce routing congestion and critical path delay. After fabricated in 0.18 μm 1P6M process, the proposed partially parallel decoder can support 480Mb/s data rate under 8 decoding iterations. Second decoder is implemented based on a (1200, 720) irregular parity check matrix. For achieving higher chip density and less interconnection delay, the proposed architecture features a new data reordering technique to simplify data bus between message memories and computational units; as a result, the chip size can be greatly reduced due to the increased chip density. Moreover, the LDPC decoder can also process two different codewords concurrently to increase throughput and datapath efficiency. After 0.18 μm 1P6M chip implementation, a 3.33Gb/s data rate with 8 decoding iterations is achieved in the 21.23mm² silicon area. The other experiment using 0.13 μm 1P8M technology can further reach a 5.92Gb/s data rate within 10.24mm² area while the chip density is 75.4%.

誌 謝

隨著鳳凰花開，轉眼間又到了畢業的季節。在這二年的碩士生涯中，首先我要向指導教授李鎮宜博士表達最誠摯的謝意。由於老師指導有方，讓我能在短時間內找到正確的研究方向；在遇到挫折時也能從經驗中學習，培養正確的研究精神。另外，我也要感謝 Si2 實驗室中的每一位成員。在這裡的每個人研究領域或有不同，但都願意彼此幫助，讓我不僅了解團隊工作的重要性，更令人倍感溫馨；尤其我要感謝林建青學長，在我研究過程中不厭其煩地提供不少建議。最後，我要謝謝在背後默默支持著我的家人和朋友，讓我順利完成了這份學業。在大家的鼓勵下，讓我過得更多采多姿，我一定不會忘記這段令人充滿回憶的生活。



Contents

CHAPTER 1 INTRODUCTION.....	1
1.1 MOTIVATION.....	1
1.2 THESIS ORGANIZATION.....	2
CHAPTER 2 LOW-DENSITY PARITY-CHECK CODES.....	4
2.1 LDPC CODES.....	4
2.2 MESSAGE PASSING ALGORITHM.....	7
2.2.1 <i>Principle of Message Passing Algorithm</i>	7
2.2.2 <i>Message Passing on Bit Nodes</i>	9
2.2.3 <i>Message Passing on Check Nodes</i>	10
2.3 LDPC CODE DECODING ALGORITHM.....	12
2.3.1 <i>Sum-Product Algorithm (SPA)</i>	12
2.3.2 <i>Log-Likelihood Ratio Sum-Product Algorithm (LLR-SPA)</i>	15
2.3.3 <i>Min-Sum Algorithm (MS)</i>	16
CHAPTER 3 HIGH-SPEED COMMUNICATION SYSTEMS WITH LDPC CODES .	18
3.1 INTRODUCTIONS TO HIGH-SPEED COMMUNICATION SYSTEMS.....	18
3.1.1 <i>Satellite Wireless Communication</i>	18
3.1.2 <i>60GHz Band Wireless Communication</i>	19
3.1.3 <i>Ultra-Wideband System</i>	20

3.2	ERROR-CORRECTING PERFORMANCE OF LDPC CODES IN UWB SYSTEM	22
3.2.1	<i>Performance Analysis of Code I</i>	23
3.2.2	<i>Performance Analysis of Code II</i>	27
3.2.3	<i>Performance Comparison with Convolutional Codes</i>	29
CHAPTER 4 ARCHITECTURES OF PROPOSED LDPC CODE DECODERS		31
4.1	INTRODUCTION TO THE CONVENTIONAL DESIGN	32
4.2	PROPOSED LDPC CODE I DECODER DESIGN.....	34
4.2.1	<i>Channel Value Interconnection</i>	36
4.2.2	<i>Check Node Unit</i>	38
4.2.3	<i>Bit Node Unit</i>	42
4.2.4	<i>Chip Implementation</i>	43
4.3	PROPOSED LDPC CODE II DECODER DESIGN	45
4.3.1	<i>Input Buffer</i>	46
4.3.2	<i>Check Node Unit and Bit Node Unit</i>	50
4.3.3	<i>Message Memory Unit</i>	52
4.3.4	<i>Timing Schedule</i>	56
4.3.5	<i>Chip Implementation</i>	57
4.4	SUMMARY AND COMPARISON	60
CHAPTER 5 CONCLUSION AND FUTURE WORK.....		61
5.1	CONCLUSION	61
5.2	FUTURE WORK	61
BIBLIOGRAPHY.....		62

List of Figures

FIG. 2.1 EXAMPLE OF REGULAR LDPC CODE PARITY CHECK MATRIX	5
FIG. 2.2 BIPARTITE GRAPH OF THE CODE SPECIFIED BY MATRIX IN FIG. 2.1	6
FIG. 2.3 MESSAGE PASSING ON A NODE	8
FIG. 2.4 MESSAGE PASSING ON A BIT NODE	10
FIG. 2.5 MESSAGE PASSING ON A CHECK NODE.....	11
FIG. 2.6 ITERATIVE DECODING FLOW CHART FOR LDPC CODES	13
FIG. 2.7 MESSAGE PASSING IN LDPC CODE DECODING.....	14
FIG. 3.1 FUNCTIONAL BLOCK DIAGRAM OF THE DVB-S.2 SYSTEM.....	19
FIG. 3.2 BLOCK DIAGRAM OF MB-OFDM UWB SYSTEM	21
FIG. 3.3 BLOCK DIAGRAM OF THE PROPOSED LDPC-COFDM UWB SYSTEM.....	22
FIG. 3.4 PERFORMANCE RESULTS OF THE (600, 450) LDPC CODE.....	24
FIG. 3.5 FIXED POINT SIMULATION OF THE (600, 450) LDPC CODE	26
FIG. 3.6(A) BER OF THE (1200, 720) LDPC CODE	27
FIG. 3.6(B) PER OF THE (1200, 720) LDPC CODE.....	28
FIG. 3.7(A) FIXED POINT SIMULATION OF BER FOR THE (1200, 720) LDPC CODE	28
FIG. 3.7(B) FIXED POINT SIMULATION OF PER FOR THE (1200, 720) LDPC CODE	29
FIG. 3.8 PERFORMANCE COMPARISON FOR DIFFERENT CODES	30
FIG. 4.1 BLOCK DIAGRAM OF CONVENTIONAL LDPC CODE DECODER.....	32
FIG. 4.2 ARCHITECTURE OF CONVENTIONAL CNU BASED ON: (A) LLR-SPA AND (B) MIN-SUM ALGORITHM	33

FIG. 4.3 ARCHITECTURE OF CONVENTIONAL BNU	34
FIG. 4.4 THE ARCHITECTURE OF LDPC CODE I DECODER.....	35
FIG. 4.5 THE PARTITION FOR PARITY CHECK MATRIX H OF CODE I.....	36
FIG. 4.6 DATA PATH OF PROPOSED PARTIAL-PARALLEL DECODER.....	36
FIG. 4.7 PROPOSED LDPC DECODING FLOW	37
FIG. 4.8 TIMING DIAGRAM OF THE PROPOSED LDPC CODE I DECODER	38
FIG. 4.9 BLOCK DIAGRAM OF CS14	39
FIG. 4.10(A) BLOCK DIAGRAM OF PROPOSED CMP-4	40
FIG. 4.10(B) BLOCK DIAGRAM OF PROPOSED CMP-14.....	40
FIG. 4.11 THE PROPOSED 14-INPUT CNU ARCHITECTURE	41
FIG. 4.12 THE PROPOSED BNU ARCHITECTURE	43
FIG. 4.13 DIE MICROGRAPH OF THE LDPC-COFDM UWB TRANSCEIVER CHIP	44
FIG. 4.14 THE PARTITION OF PARITY CHECK MATRIX H OF CODE II.....	45
FIG. 4.15 THE PROPOSED LDPC CODE II DECODER ARCHITECTURE.....	46
FIG. 4.16 THE CONVENTIONAL ARCHITECTURE OF INPUT BUFFER	47
FIG. 4.17(A) THE ARCHITECTURE OF RE BASED INPUT BUFFER.....	47
FIG. 4.17(B) THE TIMING DIAGRAM OF RE BASED INPUT BUFFER.....	48
FIG. 4.18 THE ARCHITECTURE AND TIMING DIAGRAM OF RS-BASED INPUT BUFFER	49
FIG. 4.19 THE COMPARISON OF THREE INPUT BUFFER DESIGNS	50
FIG. 4.20 CNU ARCHITECTURE OF PROPOSED LDPC CODE II DECODER	51
FIG. 4.21 BNU ARCHITECTURE OF PROPOSED LDPC CODE II DECODER	51
FIG. 4.22 THE ARCHITECTURE AND TIMING DIAGRAM OF MUX-BASED MMU.....	52
FIG. 4.23(A) THE ARCHITECTURE OF RE-4B BASED MMU.....	53
FIG. 4.23(B) THE TIMING DIAGRAM OF RE-4B BASED MMU.....	54
FIG. 4.24 THE ARCHITECTURE AND TIMING DIAGRAM OF RE-5B BASED MMU	55
FIG. 4.25 COMPARISON OF THREE MMU DESIGNS	56

FIG. 4.26 TIMING DIAGRAM OF PROPOSED LDPC CODE II DECODER 57

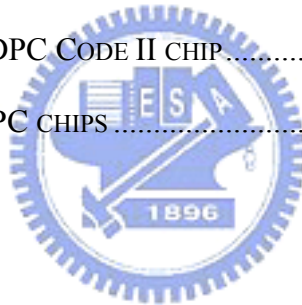
FIG. 4.27 DIE MICROGRAPH OF THE 0.18 μ m LDPC CODE II DECODER CHIP..... 58

FIG. 4.28 LAYOUT VIEW OF THE 0.13 μ m LDPC CODE II DECODER CHIP 59



List of Tables

TABLE 3.1 SPECIFICATION OF REFERENCED MB-OFDM UWB SYSTEM.....	22
TABLE 3.2 BIT WIDTH DISTRIBUTION FOR DIFFERENT QUANTIZATION SCHEMES.....	25
TABLE 4.1 SUMMARY OF THE TWO LDPC CODES	31
TABLE 4.2 COMPARISON OF DIFFERENT CNU ARCHITECTURES	42
TABLE 4.3 SUMMARY OF THE LDPC CODE I CHIP.....	44
TABLE 4.4 SUMMARY OF THE LDPC CODE II CHIP.....	59
TABLE 4.5 COMPARISON OF LDPC CHIPS.....	60



Chapter 1

Introduction

1.1 Motivation

Low-density parity-check (LDPC) code, a linear block code defined by a very sparse parity check matrix, was first introduced by Gallager [1], [2]. Due to the difficulty of circuit implementation, LDPC codes have been ignored for about forty years except for the study of codes defined on graphs by Tanner [3]. The rediscovery of LDPC codes were done by Spielman *et al.* [4] and MacKay *et al.* [5], [6]. It has engaged much research interest because the sparse property of parity check matrix makes the decoding algorithm simple and practical at good communication rates [5]. It was proven [7] that the LDPC code with large block length can beat turbo code [8], and achieve a capacity within 0.0045dB of the Shannon limit on AWGN channels. Besides their good error-correcting capability, LDPC codes have inherently fully parallelism and the simplicity of arithmetic computations. As a result, LDPC codes have been considered as next-generation forward error-control (FEC) technology for many high speed applications such as magnetic storage and telecommunications. However, the very large scale integrated circuits (VLSI) implementation of LDPC code decoders still remains a challenge in real applications.

The main challenge of LDPC code decoder falls in the complex interconnections due to the randomness of parity check matrix. To efficiently design the decoder, the realization of its iterative decoding process which is referred to the message passing algorithm [5] becomes the most critical issue. According to different decoding schedules, the implementation of LDPC code decoders can be partitioned into two categories, fully parallel decoders and partially

parallel decoders.

Fully parallel decoders directly map the corresponding bipartite graph [3] into hardware and all the processing units are hard-wired according to the connectivity of the graph. Thus they can achieve very high decoding speed but have a large hardware cost. The fully parallel implementation in [9] presents a 1024-bit, 1-Gb/s LDPC code decoder, which demands large area due to large amount of processing units and the complicated interconnections. The partially parallel architecture in [10] maps a certain number of processing unit into a single hardware block by using time-division multiplexing. It trades the decoding throughput for the reduction of hardware complexity. However, they also suffer from the same routing complications, and may be even worse due to multiplexers. Another implementation approach is presented in [11], which employs a turbo-like decoding algorithm with structured parity check matrices. The throughput is quite low due to the trellis-based decoding process.

In this thesis, two decoders with different block lengths are implemented based on the partially parallel architecture. To solve the problems mentioned previously, efficient methods are proposed and applied to the decoders to eliminate multiplexers for less signal routing. The implementation results show how the proposed methods improve the performance. The detail discussion and architecture will be given in the following chapters.

1.2 Thesis Organization

The remainder of this thesis is organized as follows. Chapter 2 describes the characteristics and decoding algorithms of LDPC codes. High-speed applications which adopted LDPC codes or potentially will adopt LDPC codes as the FEC kernel are introduced in Chapter 3. Simulation results and performance analysis will also be discussed here. In Chapter 4, the proposed LDPC code decoders, including functional units, data rescheduling and memory arrangement, are presented in detail. Besides, the chip implementation results

and comparisons with the state-of-the-arts will also be shown. Finally, conclusion and future work are made in Chapter 5.



Chapter 2

Low-Density Parity-Check Codes

Low-density parity-check (LDPC) codes are linear block codes that are specified by sparse parity check matrices containing mostly 0's and only a small number of 1's [1]. The code structures and decoding algorithms can be represented by bipartite graph [2]. Furthermore, it has been shown that the codes can achieve a capacity near Shannon limit with large block length. In this chapter, the code characteristics and decoding algorithms are presented.

2.1 LDPC Codes



The parity check matrix \mathbf{H} which has N columns and M rows defines a LDPC code with the block length of N bits and M parity checks. Assuming the matrix is of full rank, the number of information bits is $K = N - M$, and the code rate is $R = 1 - M/N$. It was shown by Gallager [2] that for large block lengths, the minimum distance of the code grows linearly with N . Thus block lengths of LDPC codes are often designed as large as possible. For a regular LDPC code, each column and row contains a fixed number of 1's in \mathbf{H} , leading to equal weights for both columns and rows. Otherwise, the code is termed irregular. It has been shown that irregular codes outperform regular codes due to wave effect [12]. An example of regular LDPC code parity check matrix is shown in Fig. 2.1.

Generation a set of valid codewords requires the generator matrix \mathbf{G} , which can be derived from \mathbf{H} . The relationship between \mathbf{G} and \mathbf{H} can be expressed as

$$\mathbf{G} \cdot \mathbf{H}^T = \mathbf{0}. \quad (2. 1)$$

Let $\mathbf{u} = (u_1, u_2, u_3, \dots, u_K)$ with $u_i \in \{0, 1\}$ be the information bits, a LDPC code C is defined as

$$C = \{\mathbf{x} \mid \mathbf{x} = \mathbf{u} \cdot \mathbf{G}\}. \quad (2.2)$$

Note that matrix G is not generally sparse; as a result, the complexity of encoding process is much higher due to the large and dense matrix multiplication. From equation (2.1) and (2.2), a valid codeword vector $\mathbf{x} = (x_1, x_2, x_3, \dots, x_N)$ should satisfy M parity check equations

$$\mathbf{x} \cdot \mathbf{h}_i^T = 0 \quad i = 1, 2, \dots, M, \quad (2.3)$$

where $\mathbf{h}_i = (h_{i,1}, h_{i,2}, \dots, h_{i,N})$ denotes the row space of \mathbf{H} .

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

Fig. 2.1 Example of regular LDPC code parity check matrix

LDPC codes can also be represented in bipartite graph. On one side the graph has N bit nodes which correspond to the N columns of \mathbf{H} and M check nodes which correspond to the M rows of \mathbf{H} on the other side. An edge which connects a bit node B_j and check node C_i corresponds to a 1 in the entry (i, j) of \mathbf{H} . Fig. 2.2 is the corresponding bipartite graph of the LDPC code specified by the parity check matrix in Fig. 2.1.

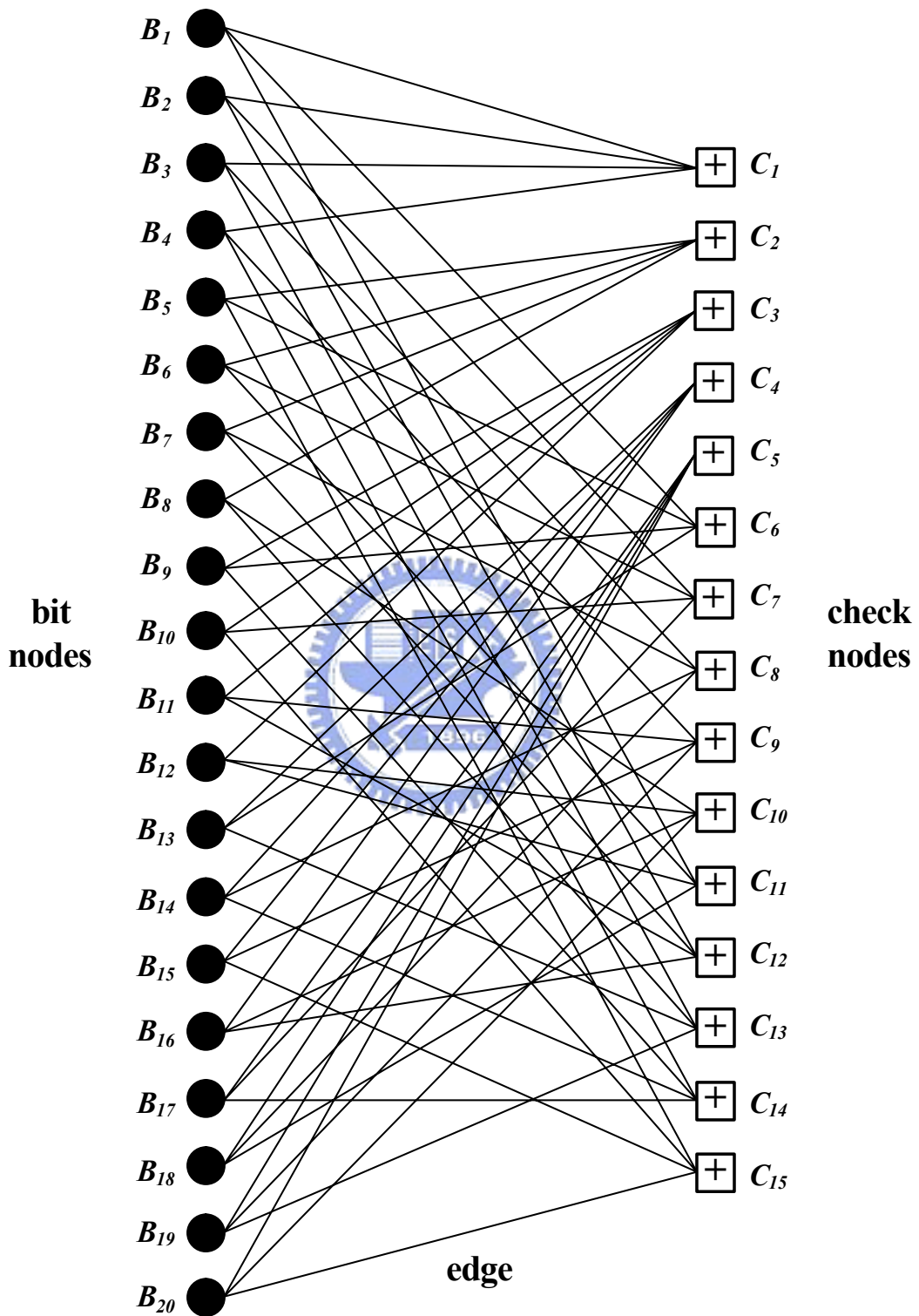


Fig. 2.2 Bipartite graph of the code specified by matrix in Fig. 2.1

2.2 Message Passing Algorithm

In this section, message passing algorithm which is used to perform probabilistic decoding is introduced. The **intrinsic probability** $P_E^{\text{int}}(x = a)$ represents the probability that the variable x chooses the value a . The **extrinsic probability** $P_E^{\text{ext}}(x = a)$ describes the new information for variable x which is obtained from the event E . Moreover, the ***a posteriori* probability** $P_E^{\text{post}}(x = a)$ represents the conditional probability that the variable x takes the value a based on the knowledge of event E .

2.2.1 Principle of Message Passing Algorithm

The key factor of the message passing algorithm is to iteratively pass and exchange probabilistic messages in a graph. Extrinsic and *a posteriori* probabilities can be evaluated based on given intrinsic probabilities and the construction of the graph.

Consider a node G with $K+1$ edges, which are associated with the variables e_0, e_1, \dots, e_K belonging to the alphabet sets A_0, A_1, \dots, A_K , respectively. The connection is shown as Fig. 2.3. For simplicity, only the case of binary variables is discussed in the following. That is, $A_i \in \mathbb{Z}_2$. Denote the intrinsic, extrinsic and *a posteriori* probability for e_i with respect to event G as $P_G^{\text{int}}(e_i = \xi_i)$, $P_G^{\text{ext}}(e_i = \xi_i)$ and $P_G^{\text{post}}(e_i = \xi_i)$, respectively. Assuming that the intrinsic probability for variable e_i is available, the *a posteriori* probability can be derived by Bayes' theorem as

$$\begin{aligned}
 P_G^{\text{post}}(e_i = \xi_i) &= P(e_i = \xi_i | G) \\
 &= \frac{P(G, e_i = \xi_i)}{P(G)} \\
 &= \frac{1}{P(G)} P(G | e_i = \xi_i) P_G^{\text{int}}(e_i = \xi_i).
 \end{aligned} \tag{2.4}$$

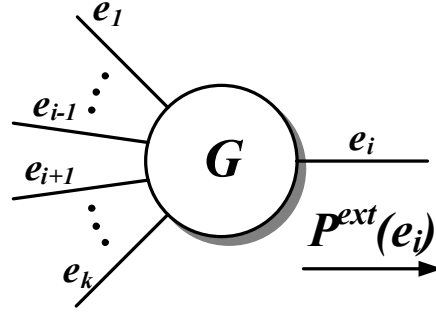


Fig. 2.3 Message passing on a node

Note that the extrinsic probability is in proportion to $P(G | e_i = \xi_i)$. That is

$$P_G^{ext}(e_i = \xi_i) = \alpha_i P(G | e_i = \xi_i), \quad (2.5)$$

where α_i is a scaling constant. A constraint set $S_G \in A_0 \times A_1 \times L \times A_K$ that the values of variables (e_0, e_1, \dots, e_K) have to satisfy is defined on node G . Therefore, event G is true only when

$$(\xi_0, \xi_1, \dots, \xi_K) \in S_G, \quad (2.6)$$

where $e_0 = \xi_0, e_1 = \xi_1, \dots, e_K = \xi_K$.

To evaluate the extrinsic and *a posteriori* probabilities of variables $\{e_i\}_{i=0}^K$, the probabilities of variable e_0 are considered without loss of generality. Note that the product of alphabets $A_1 \times A_2 \times \dots \times A_K$ forms a complete set of values for variables (e_1, e_2, \dots, e_K) .

Hence,

$$\sum_{(\xi_1, \dots, \xi_K) \in A_1 \times L \times A_K} P(\{e_i = \xi_i\}_{i=1}^K) = 1. \quad (2.7)$$

In this way, the probability of event G can be decomposed as

$$P(G) = \sum_{(\xi_1, \dots, \xi_K) \in A_1 \times L \times A_K} P(G, \{e_i = \xi_i\}_{i=1}^K). \quad (2.8)$$

The extrinsic probability $P_G^{ext}(e_0 = \xi_0)$ can thus be derived by

$$P_G^{ext}(e_0 = \xi_0) = \alpha_0 P(G | e_0 = \xi_0) = \alpha_0 \sum_{(\xi_1, \dots, \xi_K) \in A_1 \times L \times A_K} P(G, \{e_i = \xi_i\}_{i=1}^K | e_0 = \xi_0), \quad (2.9)$$

where α_0 is a scaling constant. With chain rule and the independence of the variables $\{e_i\}_{i=0}^K$, the following result is obtained.

$$\begin{aligned}
P(G, \{e_i = \xi_i\}_{i=1}^K | e_0 = \xi_0) &= P(G | \{e_i = \xi_i\}_{i=0}^K) \cdot P(\{e_i = \xi_i\}_{i=1}^K | e_0 = \xi_0) \\
&= P(G | \{e_i = \xi_i\}_{i=0}^K) \cdot \prod_{i=1}^K P(e_i = \xi_i).
\end{aligned} \tag{2.10}$$

Because event G is true only when equation (2.6) is satisfied, the first term in equation (2.10) can be written as

$$P(G | \{e_i = \xi_i\}_{i=0}^K) = \begin{cases} 1 & \text{if } (\xi_0, \xi_1, \dots, \xi_K) \in S_G \\ 0 & \text{otherwise} \end{cases}. \tag{2.11}$$

By putting together equation (2.9), (2.10) and (2.11), the expression of $P_G^{ext}(e_0 = \xi_0)$ can be rewritten as

$$P_G^{ext}(e_0 = \xi_0) = \alpha_0 \sum_{\substack{\xi_1, \dots, \xi_K \\ (\xi_0, \xi_1, \dots, \xi_K) \in S_G}} \prod_{i=1}^K P_G^{int}(e_i = \xi_i). \tag{2.12}$$

The *a posteriori* probability $P_G^{post}(e_0 = \xi_0)$ can be derived by combining equation (2.4) and (2.12).

$$\begin{aligned}
P_G^{post}(e_0 = \xi_0) &= \frac{1}{P(G)} \cdot \left(c_0 \sum_{\substack{\xi_1, \dots, \xi_K \\ (\xi_0, \xi_1, \dots, \xi_K) \in S_G}} \prod_{i=1}^K P_G^{int}(e_i = \xi_i) \right) \cdot P_G^{int}(e_0 = \xi_0) \\
&= \alpha'_0 \sum_{\substack{\xi_1, \dots, \xi_K \\ (\xi_0, \xi_1, \dots, \xi_K) \in S_G}} \prod_{i=0}^K P_G^{int}(e_i = \xi_i),
\end{aligned} \tag{2.13}$$

where $\alpha'_0 = \alpha_0 \cdot 1/P(G)$ is a normalization constant.

2.2.2 Message Passing on Bit Nodes

Representing one bit of the codeword, a bit node in a bipartite graph corresponds to a specified column in the parity check matrix \mathbf{H} which defines the code. Thus the constraint on a bit node specifies that the associated variables should be equal. The constraint set S_B on bit node B , which connects to $K+1$ check nodes, can be expressed as

$$S_B = \{(e_0, e_1, \dots, e_K) | e_0 = e_1 = \dots = e_K\}. \tag{2.14}$$

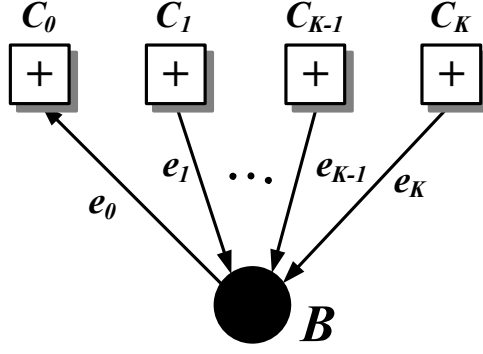


Fig. 2.4 Message passing on a bit node

The connection is also shown in Fig. 2.4.

For bit node B, the input message vector along edge e_i is defined as $\mu_{C_i \rightarrow B}(e_i)$, where $i = 1 \sim K$. Based on equation (2.12) and (2.14), the output message $\mu_{B \rightarrow C_0}(e_0 = \xi_0)$ along edge e_0 is

$$\begin{aligned}
 \mu_{B \rightarrow C_0}(e_0 = \xi_0) &= P_B^{ext}(e_0 = \xi_0) \\
 &= \alpha_0 \sum_{\substack{\xi_1, \dots, \xi_K \\ (\xi_0, \xi_1, \dots, \xi_K) \in \mathcal{S}_B}} \prod_{i=1}^K \mu_{C_i \rightarrow B}(e_i = \xi_i). \\
 &= \alpha_0 \prod_{i=1}^K \mu_{C_i \rightarrow B}(e_i = \xi_0),
 \end{aligned} \tag{2.15}$$

where α_0 is the normalization constant.

2.2.3 Message Passing on Check Nodes

In a bipartite graph, a check node, denoting a parity check equation of the code, corresponds to a specified row in the parity check matrix H. Thus the constraint on a check node specifies that the summation of the associated bits should be zero. The constraint set S_C on check node C, which connects to K+1 bit nodes, can be expressed as

$$S_C = \{(e_0, e_1, \dots, e_K) \mid e_0 + e_1 + \dots + e_K = 0\}, \tag{2.16}$$

where the operation “+” represents the modulo-2 summation. The connection is shown in Fig 2.5.

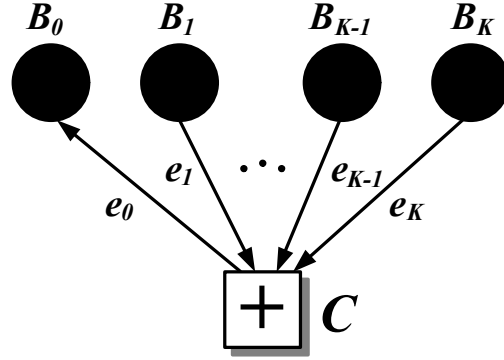


Fig. 2.5 Message passing on a check node

The input message vector along edge e_i is denoted by $\mu_{B_i \rightarrow C}(e_i)$ for $i = 1 \sim K$. With equation (2.12) and (2.15), the output message $\mu_{C \rightarrow B_0}(e_0 = \xi_0)$ along edge e_0 can be derived as

$$\begin{aligned}
 \mu_{C \rightarrow B_0}(e_0 = \xi_0) &= P_C^{ext}(e_0 = \xi_0) \\
 &= \alpha'_0 \sum_{\substack{\xi_1, \dots, \xi_K \\ (\xi_0, \xi_1, \dots, \xi_K) \in \mathcal{S}_C}} \prod_{i=1}^K \mu_{B_i \rightarrow C}(e_i = \xi_i) \\
 &= \alpha'_0 \sum_{\xi_1, \dots, \xi_K} [\xi_0 + \xi_1 + L + \xi_K = 0] \prod_{i=1}^K \mu_{B_i \rightarrow C}(e_i = \xi_i),
 \end{aligned} \tag{2.17}$$

where $[\xi_0 + \xi_1 + L + \xi_K = 0]$ is an *indicator function* that determines whether the parity check equation is satisfied. Because the indicator function consists of large number of possible configurations, the summation operation in equation (2.17) is very complicated. Thus we first consider the case of $K=2$ for simplicity. Therefore,

$$\begin{bmatrix} \mu_{C \rightarrow B_0}(e_0 = 0) \\ \mu_{C \rightarrow B_0}(e_0 = 1) \end{bmatrix} = \begin{bmatrix} \sum_{\xi_1, \xi_2} [0 + \xi_1 + \xi_2 = 0] \prod_{i=1}^2 \mu_{B_i \rightarrow C}(e_i = \xi_i) \\ \sum_{\xi_1, \xi_2} [1 + \xi_1 + \xi_2 = 0] \prod_{i=1}^2 \mu_{B_i \rightarrow C}(e_i = \xi_i) \end{bmatrix}. \tag{2.18}$$

When $e_0 = 0$, the indicator function is true if and only if the configuration is either $e_1 = e_2 = 0$ or $e_1 = e_2 = 1$. Hence equation (2.18) can be decomposed as

$$\begin{aligned}
\begin{bmatrix} \mu_{C \rightarrow B_0}(e_0 = 0) \\ \mu_{C \rightarrow B_0}(e_0 = 1) \end{bmatrix} &= \begin{bmatrix} \mu_{B_1 \rightarrow C}(e_1 = 0)\mu_{B_2 \rightarrow C}(e_2 = 0) + \mu_{B_1 \rightarrow C}(e_1 = 1)\mu_{B_2 \rightarrow C}(e_2 = 1) \\ \mu_{B_1 \rightarrow C}(e_1 = 0)\mu_{B_2 \rightarrow C}(e_2 = 1) + \mu_{B_1 \rightarrow C}(e_1 = 1)\mu_{B_2 \rightarrow C}(e_2 = 0) \end{bmatrix} \\
&= \begin{bmatrix} (1 - \mu_{B_1 \rightarrow C}(e_1 = 1))(1 - \mu_{B_2 \rightarrow C}(e_2 = 1)) + \mu_{B_1 \rightarrow C}(e_1 = 1)\mu_{B_2 \rightarrow C}(e_2 = 1) \\ (1 - \mu_{B_1 \rightarrow C}(e_1 = 1))\mu_{B_2 \rightarrow C}(e_2 = 1) + \mu_{B_1 \rightarrow C}(e_1 = 1)(1 - \mu_{B_2 \rightarrow C}(e_2 = 1)) \end{bmatrix},
\end{aligned} \tag{2.19}$$

where $\mu_{B_1 \rightarrow C}(e_1 = 0) = 1 - \mu_{B_1 \rightarrow C}(e_1 = 1)$ and $\mu_{B_2 \rightarrow C}(e_2 = 0) = 1 - \mu_{B_2 \rightarrow C}(e_2 = 1)$. Furthermore, the expression in equation (2.19) can be rewritten as

$$\begin{bmatrix} 2\mu_{C \rightarrow B_0}(e_0 = 0) - 1 \\ 2\mu_{C \rightarrow B_0}(e_0 = 1) - 1 \end{bmatrix} = \begin{bmatrix} (1 - 2\mu_{B_1 \rightarrow C}(e_1 = 1))(1 - 2\mu_{B_2 \rightarrow C}(e_2 = 1)) \\ -(1 - 2\mu_{B_1 \rightarrow C}(e_1 = 1))(1 - 2\mu_{B_2 \rightarrow C}(e_2 = 1)) \end{bmatrix}. \tag{2.20}$$

By induction [13], the results in equation (2.20) can be generalized for $K > 2$ and becomes

$$\begin{bmatrix} 2\mu_{C \rightarrow B_0}(e_0 = 0) - 1 \\ 2\mu_{C \rightarrow B_0}(e_0 = 1) - 1 \end{bmatrix} = \begin{bmatrix} \prod_{i=1}^K (1 - 2\mu_{B_i \rightarrow C}(e_i = 1)) \\ -\prod_{i=1}^K (1 - 2\mu_{B_i \rightarrow C}(e_i = 1)) \end{bmatrix}. \tag{2.21}$$

As a result, the output messages can be expressed in terms of the input messages:

$$\begin{bmatrix} \mu_{C \rightarrow B_0}(e_0 = 0) \\ \mu_{C \rightarrow B_0}(e_0 = 1) \end{bmatrix} = \begin{bmatrix} \frac{1}{2} (1 + \prod_{i=1}^K (1 - 2\mu_{B_i \rightarrow C}(e_i = 1))) \\ \frac{1}{2} (1 - \prod_{i=1}^K (1 - 2\mu_{B_i \rightarrow C}(e_i = 1))) \end{bmatrix}. \tag{2.22}$$

2.3 LDPC Code Decoding Algorithm

2.3.1 Sum-Product Algorithm (SPA)

For a $M \times N$ parity check matrix H and the corresponding graph, B_i for $i = 1, 2, \dots, N$ denote the bit nodes, C_j for $j = 1, 2, \dots, M$ are check nodes, and e_{ij} is the edge connecting B_i and C_j . Furthermore, $M(i)$ is the set of check nodes connected to bit node B_i , and $L(j)$ is the set of bit nodes that are associated with check node C_j . The codeword is also represented by $\mathbf{x} = [x_1, x_2, \dots, x_N]$. The intrinsic probabilities with respect to the LDPC code can thus be

written as

$$P_{LDPC}^{\text{int}}(x_i) = P(x_i = \xi_i), \quad (2.23)$$

where $\xi_i \in \{0, 1\}$ and $P(x_i = 0) = 1 - P(x_i = 1)$, assuming binary symmetric channel.

Fig 2.6 illustrates the iterative decoding flow of LDPC codes where each step will be described as follows [5].

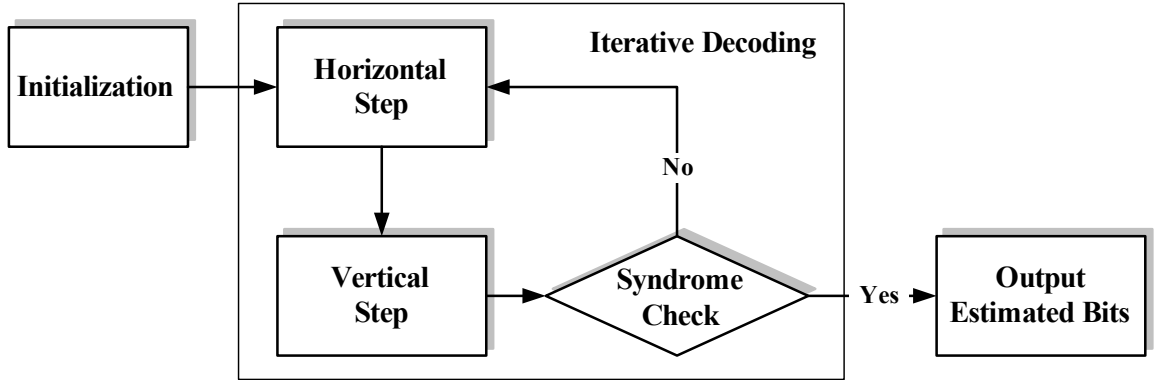


Fig. 2.6 Iterative decoding flow chart for LDPC codes

(1) Initialization: The messages from bit node B_i to check node C_j are initialized as

$$\begin{bmatrix} \mu_{B_i \rightarrow C_j}(e_{ij} = 0) \\ \mu_{B_i \rightarrow C_j}(e_{ij} = 1) \end{bmatrix} = \begin{bmatrix} P(x_i = 0) \\ P(x_i = 1) \end{bmatrix}. \quad (2.24)$$

(2) Horizontal step: As shown in Fig. 2.7(a), message updates associated with check nodes are completed in this step. As shown in equation (2.22), the update equations can be expressed as

$$\begin{bmatrix} \mu_{C_j \rightarrow B_i}(e_{ij} = 0) \\ \mu_{C_j \rightarrow B_i}(e_{ij} = 1) \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \left(1 + \prod_{B_i \in L(j) \setminus B_i} (1 - 2\mu_{B_i \rightarrow C_j}(e_{ij} = 1)) \right) \\ \frac{1}{2} \left(1 - \prod_{B_i \in L(j) \setminus B_i} (1 - 2\mu_{B_i \rightarrow C_j}(e_{ij} = 1)) \right) \end{bmatrix}, \quad (2.25)$$

where $L(j) \setminus B_i$ is the set of bit nodes that participate in check node C_j except B_i .

(3) Vertical step: In vertical step, the messages associated with bit nodes are updated as illustrated in Fig. 2.7(b). According to equation (2.15), the update equations can be

expressed as

$$\begin{bmatrix} \mu_{B_i \rightarrow C_j}(e_{ij} = 0) \\ \mu_{B_i \rightarrow C_j}(e_{ij} = 1) \end{bmatrix} = \begin{bmatrix} \alpha_{ij} P(x_i = 0) \prod_{C_{j'} \in M(i) \setminus C_j} \mu_{C_{j'} \rightarrow B_i}(e_{ij'} = 0) \\ \alpha_{ij} P(x_i = 1) \prod_{C_{j'} \in M(i) \setminus C_j} \mu_{C_{j'} \rightarrow B_i}(e_{ij'} = 1) \end{bmatrix}, \quad (2.26)$$

where $M(i) \setminus C_j$ is the set of check nodes that connect to bit node B_i except C_j and α_{ij} is chosen such that $\mu_{B_i \rightarrow C_j}(e_{ij} = 0) + \mu_{B_i \rightarrow C_j}(e_{ij} = 1) = 1$.

(4) Syndrome check: The *a posteriori* probabilities for each codeword bit can be computed as

$$\begin{bmatrix} P^{post}(x_i = 0) \\ P^{post}(x_i = 1) \end{bmatrix} = \begin{bmatrix} \alpha_i P(x_i = 0) \prod_{C_j \in M(i)} \mu_{C_j \rightarrow B_i}(e_{ij} = 0) \\ \alpha_i P(x_i = 1) \prod_{C_j \in M(i)} \mu_{C_j \rightarrow B_i}(e_{ij} = 1) \end{bmatrix}, \quad (2.27)$$

where normalization factor α_i is used to ensure $P^{post}(x_i = 0) + P^{post}(x_i = 1) = 1$. The estimated bit \hat{x}_i is set to 1 if $P^{post}(x_i = 1) > 0.5$, otherwise it is set to 0. Then the syndrome equation $\hat{\mathbf{x}}H^T = \mathbf{0}$ is verified whether the estimated sequence $\hat{\mathbf{x}} = [\hat{x}_1, \hat{x}_2, \dots, \hat{x}_N]$ is a valid codeword.

The decoding process halts when the syndrome check equation is satisfied; otherwise it goes into the next decoding iteration. A failure is declared if some maximum number of iterations occurs without finding a valid codeword.

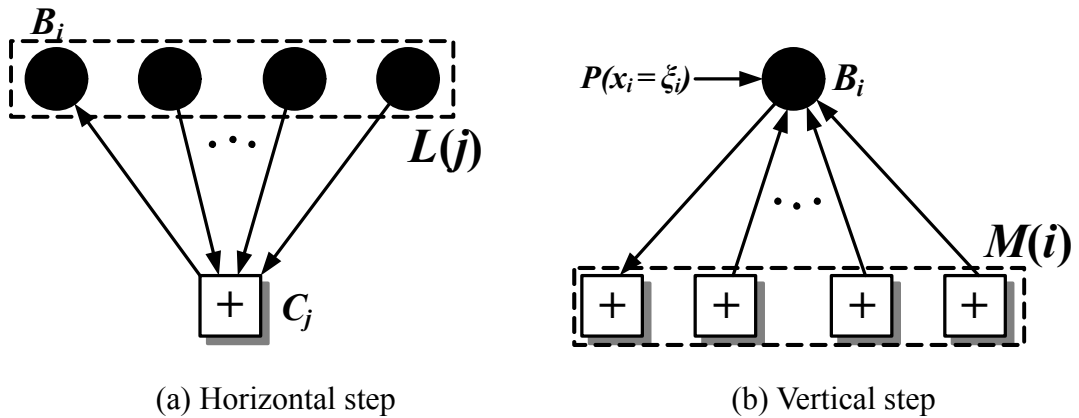


Fig. 2.7 Message passing in LDPC code decoding

2.3.2 Log-Likelihood Ratio Sum-Product Algorithm (LLR-SPA)

For a binary codeword, the decoding operations can be performed in terms of log-likelihood ratios [15]. The log-likelihood ratio (LLR) of a random variable U can be defined as

$$L(U) = \log \frac{P(U=0)}{P(U=1)}. \quad (2.28)$$

Therefore, the decoding flow can be modified as follows.

(1) Initialization: The messages sent from bit node B_i to check node C_j are initialized by

$$L_{B_i \rightarrow C_j}(e_{ij}) = \log \frac{P(x_i=0)}{P(x_i=1)}, \quad (2.29)$$

which is the so-called “channel value” or “channel information”.

(2) Horizontal step: Based on equation (2.25), the update operation in logarithmic domain can be rewritten as

$$\begin{aligned} L_{C_j \rightarrow B_i}(e_{ij}) &= \log \frac{\mu_{C_j \rightarrow B_i}(e_{ij}=0)}{\mu_{C_j \rightarrow B_i}(e_{ij}=1)} \\ &= \log \frac{1 + \prod_{B_t \in L(j) \setminus B_i} (1 - 2\mu_{B_t \rightarrow C_j}(e_{tj}=1))}{1 - \prod_{B_t \in L(j) \setminus B_i} (1 - 2\mu_{B_t \rightarrow C_j}(e_{tj}=1))}. \end{aligned} \quad (2.30)$$

Based on the hyperbolic tangent function and the arc-hyperbolic tangent function,

$$\tanh\left(\frac{u}{2}\right) = \frac{e^u - 1}{e^u + 1} \quad \text{and} \quad \tanh^{-1}(y) = \frac{1}{2} \log \frac{1+y}{1-y}, \quad (2.31)$$

the term $1 - 2\mu_{B_t \rightarrow C_j}(e_{tj}=1)$ in equation (2.31) can be expressed as

$$1 - 2\mu_{B_t \rightarrow C_j}(e_{tj}=1) = \tanh\left(\frac{1}{2} L_{B_t \rightarrow C_j}(e_{tj})\right). \quad (2.32)$$

Combining (2.30), (2.31) and (2.32), we can derive

$$\begin{aligned}
L_{C_j \rightarrow B_i}(e_{ij}) &= \log \frac{1 + \prod_{B_r \in L(j) \setminus B_i} \tanh\left(\frac{1}{2} L_{B_r \rightarrow C_j}(e_{ij})\right)}{1 - \prod_{B_r \in L(j) \setminus B_i} \tanh\left(\frac{1}{2} L_{B_r \rightarrow C_j}(e_{ij})\right)} \\
&= 2 \tanh^{-1} \left(\prod_{B_r \in L(j) \setminus B_i} \tanh\left(\frac{1}{2} L_{B_r \rightarrow C_j}(e_{ij})\right) \right) .
\end{aligned} \tag{2.33}$$

(3) Vertical step: Using LLR, the update equation can be rewritten as

$$\begin{aligned}
L_{B_i \rightarrow C_j}(e_{ij}) &= \log \frac{\mu_{B_i \rightarrow C_j}(e_{ij} = 0)}{\mu_{B_i \rightarrow C_j}(e_{ij} = 1)} \\
&= \log \frac{\alpha_{ij} P(x_i = 0) \prod_{C_r \in M(i) \setminus C_j} \mu_{C_r \rightarrow B_i}(e_{ij'})}{\alpha_{ij} P(x_i = 1) \prod_{C_r \in M(i) \setminus C_j} \mu_{C_r \rightarrow B_i}(e_{ij'})} \\
&= L(x_i) + \sum_{C_r \in M(i) \setminus C_j} L_{C_r \rightarrow B_i}(e_{ij'}) ,
\end{aligned} \tag{2.34}$$

where $L(x_i)$ is the intrinsic log-likelihood ratio of bit x_i .

(4) Syndrome check: The pseudo-*a posteriori* probabilities for each codeword bit can be computed as

$$\begin{aligned}
L^{post}(x_i) &= \log \frac{P^{post}(x_i = 0)}{P^{post}(x_i = 1)} \\
&= L(x_i) + \sum_{C_j \in M(i)} L_{C_j \rightarrow B_i}(e_{ij}) .
\end{aligned} \tag{2.35}$$

Hard decision are performed based on the sign of $L^{post}(x_i)$; therefore, bit \hat{x}_i is set to 1 if $L^{post}(x_i)$ is negative, otherwise it is set to 0.

Compared with the SPA, multiplications are replaced by additions and the normalization factors are eliminated in the LLR-SPA. Less complexity in implementation is achieved when LLR-SPA is employed.

2.3.3 Min-Sum Algorithm (MS)

In the LLR-SPA, the horizontal step is the most computationally complex part because of hyperbolic tangent functions. Hence it is difficult to implement in hardware based on

LLR-SPA. To further simplify the decoding process, the min-sum algorithm [16] is introduced.

We first consider a check node with 3 edges without loss of generality. Combining equation (2.20), (2.31) and (2.32), we can obtain

$$\begin{aligned}
L_{C \rightarrow B_0}(e_0) &= \log \frac{1 + \left(\frac{e^{L_{B_1 \rightarrow C}(e_1)} - 1}{e^{L_{B_1 \rightarrow C}(e_1)} + 1} \cdot \frac{e^{L_{B_2 \rightarrow C}(e_2)} - 1}{e^{L_{B_2 \rightarrow C}(e_2)} + 1} \right)}{1 - \left(\frac{e^{L_{B_1 \rightarrow C}(e_1)} - 1}{e^{L_{B_1 \rightarrow C}(e_1)} + 1} \cdot \frac{e^{L_{B_2 \rightarrow C}(e_2)} - 1}{e^{L_{B_2 \rightarrow C}(e_2)} + 1} \right)} \\
&= \log \frac{1 + e^{L_{B_1 \rightarrow C}(e_1)} e^{L_{B_2 \rightarrow C}(e_2)}}{e^{L_{B_1 \rightarrow C}(e_1)} + e^{L_{B_2 \rightarrow C}(e_2)}}.
\end{aligned} \tag{2.36}$$

Based on the approximation in [17], equation (2.36) becomes

$$\begin{aligned}
L_{C \rightarrow B_0}(e_0) &= \log \left(1 + e^{L_{B_1 \rightarrow C}(e_1) + L_{B_2 \rightarrow C}(e_2)} \right) - \log \left(e^{L_{B_1 \rightarrow C}(e_1)} + e^{L_{B_2 \rightarrow C}(e_2)} \right) \\
&= \max \left(0, L_{B_1 \rightarrow C}(e_1) + L_{B_2 \rightarrow C}(e_2) \right) + \log \left(1 + e^{-|L_{B_1 \rightarrow C}(e_1) + L_{B_2 \rightarrow C}(e_2)|} \right) \\
&\quad - \max \left(L_{B_1 \rightarrow C}(e_1), L_{B_2 \rightarrow C}(e_2) \right) - \log \left(1 + e^{-|L_{B_1 \rightarrow C}(e_1) - L_{B_2 \rightarrow C}(e_2)|} \right) \\
&= \text{sign} \left(L_{B_1 \rightarrow C}(e_1) \right) \text{sign} \left(L_{B_2 \rightarrow C}(e_2) \right) \min \left(\left| L_{B_1 \rightarrow C}(e_1) \right|, \left| L_{B_2 \rightarrow C}(e_2) \right| \right) + g(e_1, e_2) \\
&\approx \text{sign} \left(L_{B_1 \rightarrow C}(e_1) \right) \text{sign} \left(L_{B_2 \rightarrow C}(e_2) \right) \min \left(\left| L_{B_1 \rightarrow C}(e_1) \right|, \left| L_{B_2 \rightarrow C}(e_2) \right| \right),
\end{aligned} \tag{2.37}$$

where $g(e_1, e_2) = \log \left(1 + e^{-|L_{B_1 \rightarrow C}(e_1) + L_{B_2 \rightarrow C}(e_2)|} \right) - \log \left(1 + e^{-|L_{B_1 \rightarrow C}(e_1) - L_{B_2 \rightarrow C}(e_2)|} \right)$ is the correction factor.

By induction [15], the result in equation (2.37) can be generalized to obtain a sub-optimal expression of the horizontal step, which is

$$L_{C_j \rightarrow B_i}(e_{ij}) \approx \left(\prod_{B_r \in L(j) \setminus B_i} \text{sign} \left(L_{B_r \rightarrow C_j}(e_{rj}) \right) \right) \min_{B_r \in L(j) \setminus B_i} \left(\left| L_{B_r \rightarrow C_j}(e_{rj}) \right| \right). \tag{2.38}$$

This approximation results in a significant reduction of hardware complexity but little penalty of degraded performance [18].

In the min-sum algorithm, all steps of the decoding are the same with LLR-SPA except for the horizontal step. Thus the min-sum algorithm can be derived by just replacing equation (2.33) with (2.38) in LLR-SPA.

Chapter 3

High-Speed Communication Systems with LDPC Codes

In communication systems, channel coding is a key technique to minimize the interferences from the noisy channel. Due to the excellent error-correcting ability and the inherent parallelism, LDPC codes are suitable for high-speed applications. In this chapter, high-speed communication systems that adopted LDPC codes or potentially will apply LDPC codes as the channel coding technology are introduced. The simulation results of the error-correcting performance are also shown in the following.



3.1 Introductions to High-Speed Communication Systems

3.1.1 Satellite Wireless Communication

Digital video broadcasting (DVB) standards are established to deliver videos for the subscriber to provide various entertainments. Over past few years, different broadcasting modes have been designed for kinds of purposes, including the terrestrial, cable and satellite broadcasts. The original satellite digital video broadcasting (DVB-S) was developed in 1994 [19], whose forward error correction (FEC) technology is the concatenation of convolutional codes and Reed-Solomon codes. It is now used worldwide by most of the satellite operators for data and television broadcasting services. To improve the overall performance of the digital satellite transmission technology, the second generation of DVB-S (DVB-S.2) was developed [20]. As a successor to the current DVB-S standard, DVB-S.2 is expected to

provide not only existing but also new services, including TV, High Definition Television (HDTV), audio and other multimedia services.

Employing a powerful FEC system based on LDPC codes concatenated with BCH codes, DVB-S.2 allows quasi-error-free (QEF) operation at about 0.7dB to 1.0dB from the Shannon limit, depending on the transmission mode [20]. Moreover, a capacity gain in the order of 30 percent over DVB-S is achieved due to higher order modulation schemes. The functional block diagram of the DVB-S.2 system is illustrated in Fig. 3.1.

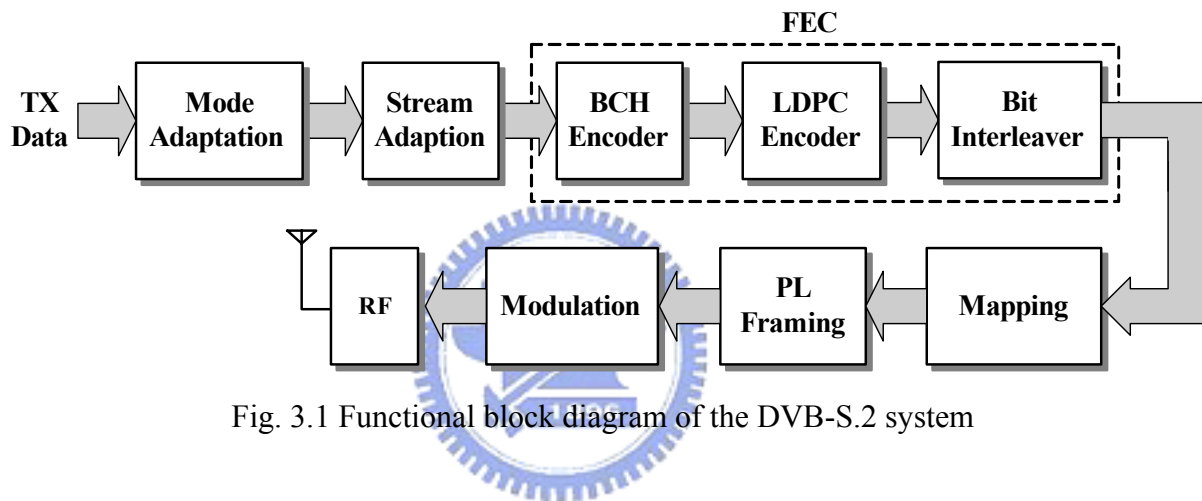


Fig. 3.1 Functional block diagram of the DVB-S.2 system

To transmit data via satellite, DVB-S.2 targets for a robust and reliable communication service. The corresponding packet error rate for DVB-S.2 at QEF over AWGN channel is 10^{-7} , which is very low as compared to other systems. Therefore LDPC codes with large block lengths, which are 64,800 and 16,200, are chosen to accomplish excellent error performance. And different coding rate of LDPC codes are specified to accommodate various transmission modes.

3.1.2 60GHz Band Wireless Communication

Recently, the Federal Communications Commission (FCC) released the RF band around 60GHz, leading to a new era in the millimeter wave based communications. It potentially can

provide a variety of applications including high-speed wireless personal area network (WPAN), automotive radar at nearby frequencies and multimedia communications. The corresponding standardization (IEEE 802.15.3c) is now under construction by IEEE 802.15 Working Group for WPANs. It is intended to offer higher data transmission, higher frequency re-usage and superior coexistence than the existing wireless systems. The working group also suggest IEEE 802.15.3c will be widely used for Gigabit Ethernet and replace the cables and other wired links.

One of the optional data rate suggested by IEEE 802.15.3c is greater than 2Gb/s in order to satisfy an evolutionary set of consumer multimedia industry in WPAN communications. Due to the required high data rate, LDPC codes are potential candidates for the FEC technique. With parallel implementation, the LDPC code decoders can easily achieve the demands for data rates over Gb/s.

3.1.3 Ultra-Wideband System



Ultra-wideband (UWB) is an emerging wireless physical (PHY)-layer technology that uses a very large bandwidth [21], [22]. It possesses unique advantages that are attractive to the communication applications: i) the potential for very high data throughput and large increase in user capacity; ii) the implementation of UWB potentially takes small size and processing power; and iii) ultra high precision ranging at centimeter level [22].

Due to the lack of available spectral bands, the applications of UWB devices prior to 2001 were mainly for military usage. In the spring of 2002, the FCC unleashed 3.1GHz to 10.6GHz RF band for increasing high-speed data transmission. Responding to this FCC ruling, industries, government agencies and academic institutions made many research efforts that adopted UWB technology in various areas. These include short-range high-speed wireless communication, localization system, high-resolution radar and imaging system. In this thesis,

we will focus on the UWB applications for wireless networks.

UWB addresses short-range connections among digital home electronics appliances that are applied for the wireless personal area network (WPAN). It is expected to provide high-speed data exchange among storage systems and real-time video/audio distribution for home entertainment devices. Due to small power consumption and high data rate, UWB technology will be exploited to replace existing wireless services.

In [23], the multi-band orthogonal frequency-division multiplexing (MB-OFDM) PHY-layer proposal indicates the coded OFDM based solution can provide up to 480Mb/s for 528MHz UWB system. The desired range in MB-OFDM is 10m for 110Mb/s and can be reduced for higher data rates [23]. To enhance the overall system performance, the convolutional codes and interleaving techniques are applied in the FEC mechanism, whose block diagram is shown in Fig. 3.2.

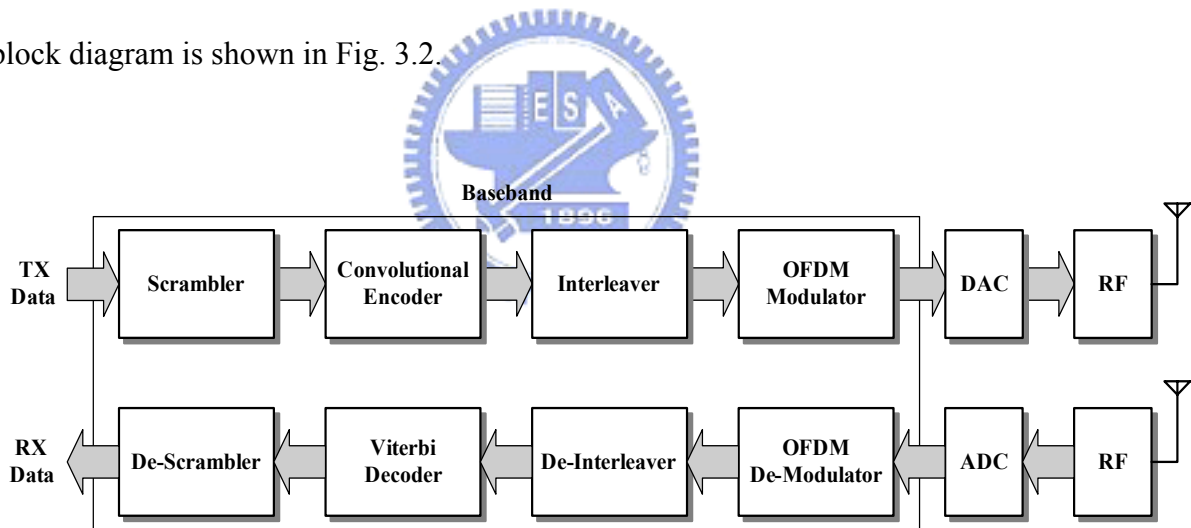


Fig. 3.2 Block diagram of MB-OFDM UWB system

For improving PHY-layer capacity, LDPC codes can increase the throughput to over 500Mb/s in future WLAN applications [24]. And the LDPC coded OFDM baseband system has been silicon proven to achieve 480 Mb/s data rate [25]. To provide better performance, the original convolutional codes and bit interleaving are replaced with LDPC codes in MB-OFDM UWB systems [25] as shown in Fig. 3.3. The overall system performance will be

described and discussed later.

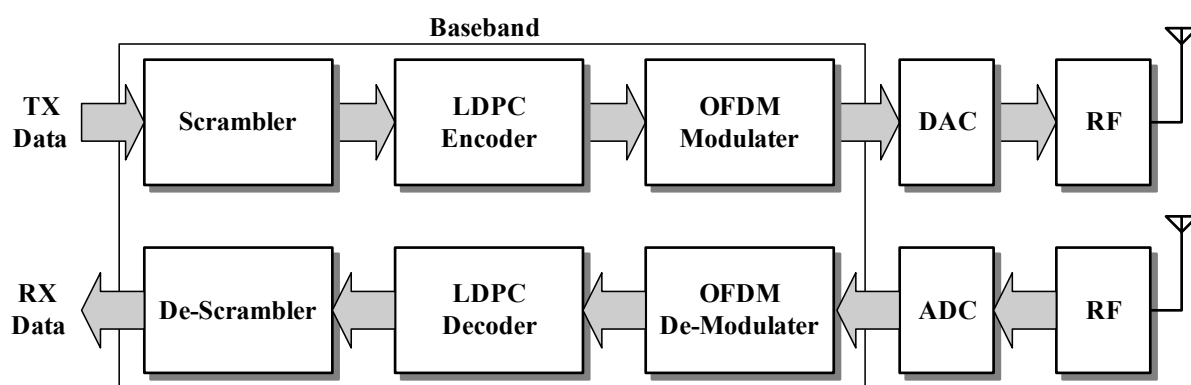


Fig. 3.3 Block diagram of the proposed LDPC-COFDM UWB system

3.2 Error-Correcting Performance of LDPC Codes in UWB System

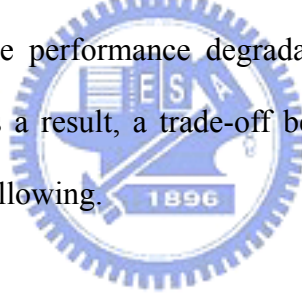
In the MB-OFDM UWB systems [25], the maximum 480Mb/s data rate with a bandwidth of 528MHz is specified. The time domain spreading scheme is used to change the data rate for different channel state information. In the following, the simulation results are based on the system illustrated in Fig 3.3, whose detail specification is given in Table 3.1. Two different irregular LDPC codes are constructed by the progressive edge-growth (PEG) algorithm [26] to enhance the system performances. One is (600, 450) LDPC code (Code I), and the other is (1200, 720) LDPC code (Code II).

Table 3.1 Specification of referenced MB-OFDM UWB system

Data rate (Mb/s)	120	240	480
Spreading gain	4	2	1
Constellation	QPSK		

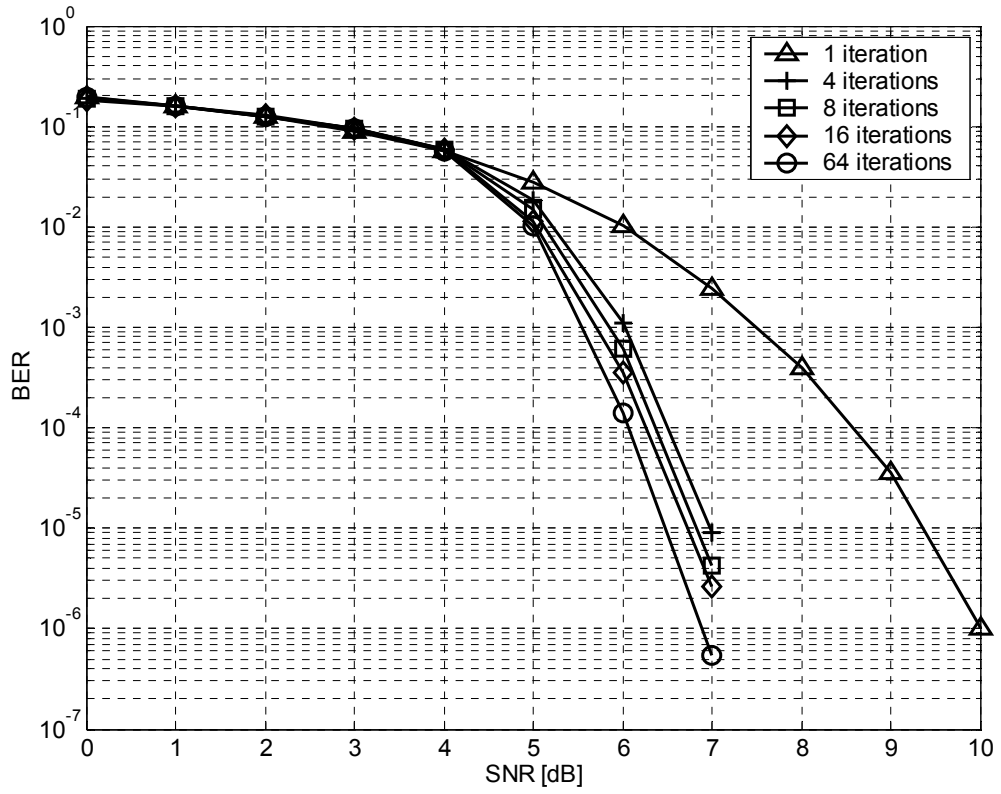
Data carrier	100
FFT size	128
Packet size (Bytes)	1024
Signal bandwidth (MHz)	528
Channel model	Additive White Gaussian Noise (AWGN)

As stated in Chapter 2, the pseudo- *a posteriori* probabilities of the codeword bits gradually converge to the real *a posteriori* probabilities as the number of decoding iterations grows. And the internal messages which are exchanged between check nodes and bit nodes are soft values. However, since infinite decoding iterations and infinite signal precision are impossible for practical implementation, the maximum iteration number and the quantization bits have to be decided. Some performance degradation would be introduced due to the implementation limitations. As a result, a trade-off between the performance and hardware cost will be concerned in the following.

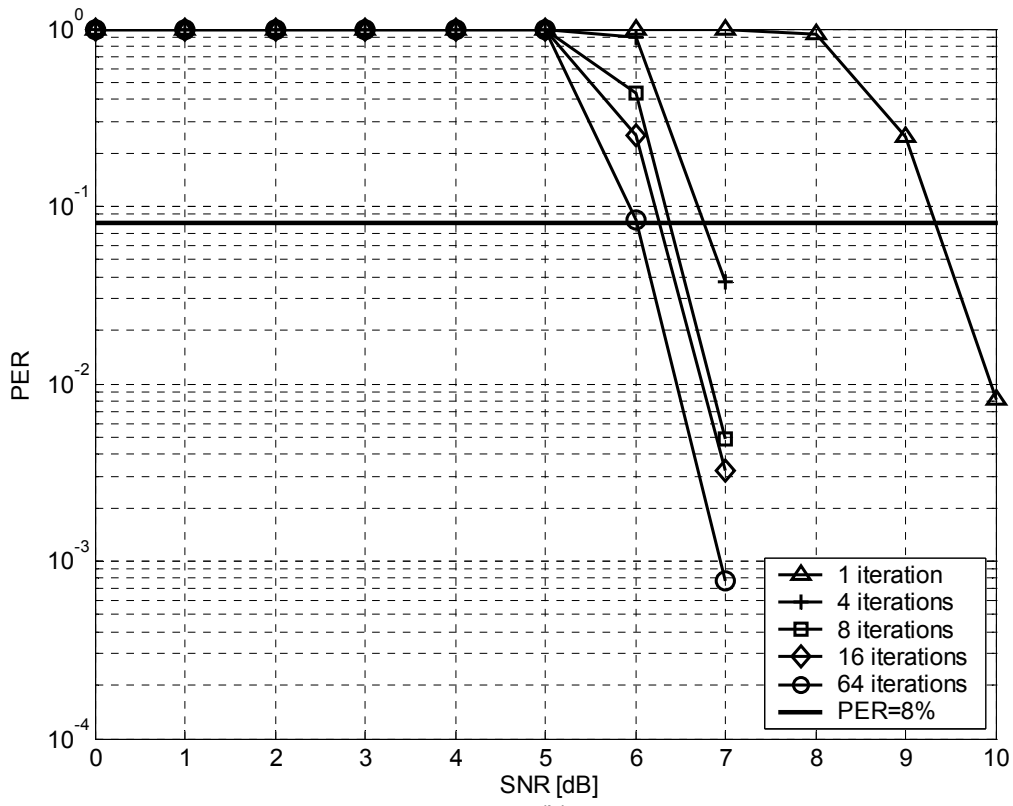


3.2.1 Performance Analysis of Code I

Code I is a (600, 450) rate-3/4 irregular LDPC code, whose column weights are fixed to 3 and row weights are ranging from 11 to 14. Based on the referenced MB-OFDM UWB system, its performances with different decoding iterations including the bit-error rate (BER) and packet-error rate (PER), which is demanded to be less than 8% [21], is shown in Fig 3.4.



(a)



(b)

Fig. 3.4 Performance results of the (600, 450) LDPC code

Note that the required signal to noise ratio (SNR) is reduced as the iteration number increases. In Fig. 3.4(b), 3dB SNR gain at PER = 8% is achieved as the number of decoding iterations moves from 1 to 8. However, the improvement tends to be insignificant after 8 iterations, which is only about 0.3dB. As a result, LDPC decoding for Code I with 8 iterations in referenced MB-OFDM UWB system is considerably a good trade-off for practical implementation.

Quantization has to be performed for two types of signal values. One is the channel values, and the other is the internal messages. Fig. 3.5 shows the fixed point simulation results of Code I, where the notation (p, q) represents that the bit width of channel values and internal messages are p and q bits, respectively. The number of bits used for the integer and the fractional part in each (p, q) quantization schemes are shown as Table 3.2.

Table 3.2 Bit width distribution for different quantization schemes

Quantization scheme	Channel value		Internal message	
	Integer part	Fractional part	Integer part	Fractional part
(4, 5)	1	3	1	4
(5, 6)	1	4	1	5

Many combinations of the quantization schemes and the bit width distributions have been tested through simulations. The performances of the quantization with more precision than (5, 6) scheme are almost the same as those with infinite precision. Consequently, the (5, 6) scheme together with the bit width distribution listed in Table 3.2 are used for the proposed LDPC Code I decoder.

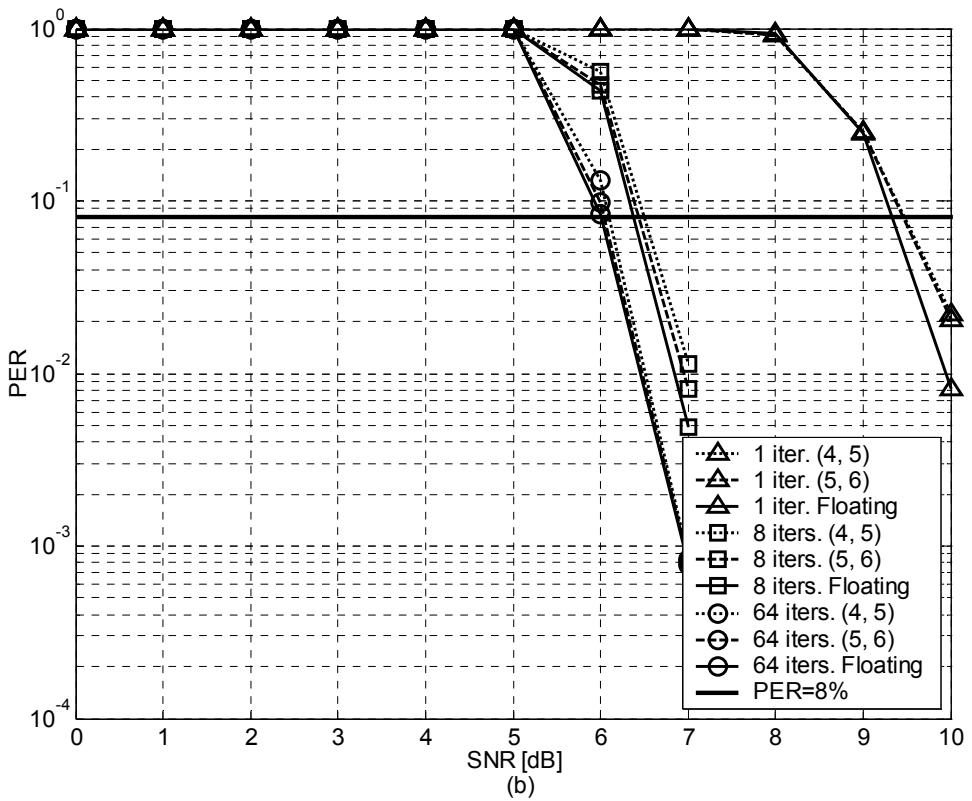
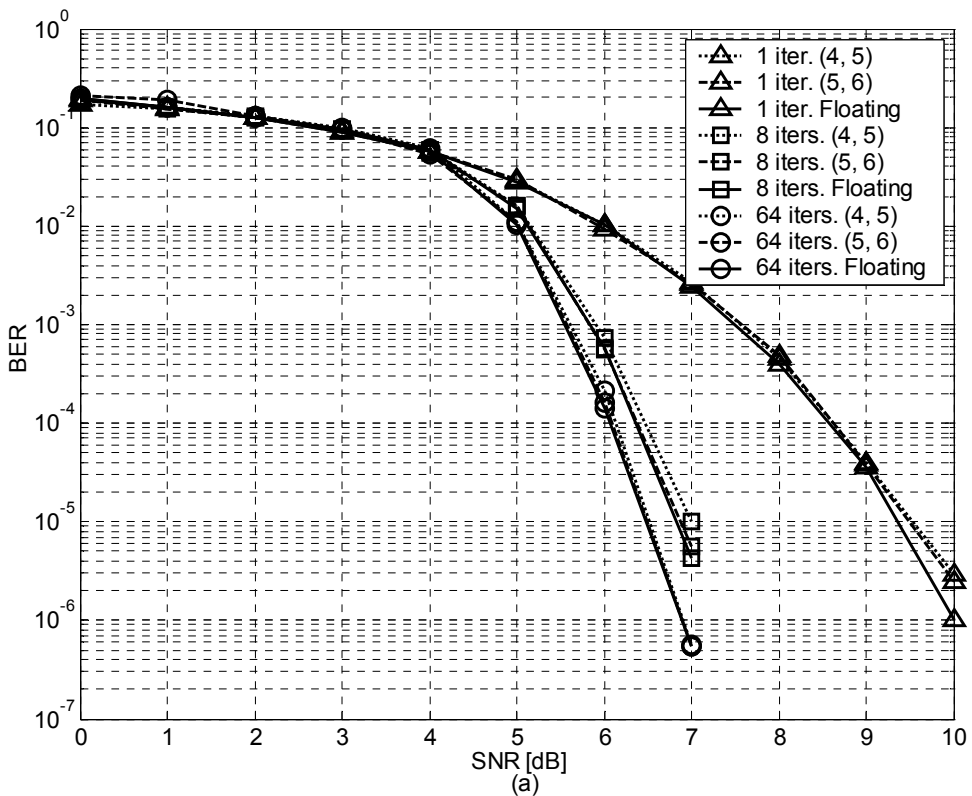


Fig. 3.5 Fixed point simulation of the (600, 450) LDPC code

3.2.2 Performance Analysis of Code II

Code II is a (1200, 720) rate-3/5 irregular LDPC code, whose column weights are also fixed to 3 and row weights range from 7 to 9. Its performances on the MB-OFDM UWB system including BER and PER under different decoding iterations are shown in Fig. 3.6.

In Fig. 3.6(b), The performance has 4.5 dB SNR gain under PER=8% is obtained as the number of decoding iterations grows from 1 to 8, but only 0.4 dB from 8 iterations to 64 iterations. Therefore, LDPC decoding for Code II with 8 iterations is considered as a good trade-off between implementation and error-correcting performance. The fixed point simulation results of Code II are shown in Fig. 3.7, and the bit width distributions are given in Table 3.2. According to the results, the (5, 6) quantization scheme is chosen as the implementation parameter for the proposed decoder for Code II.

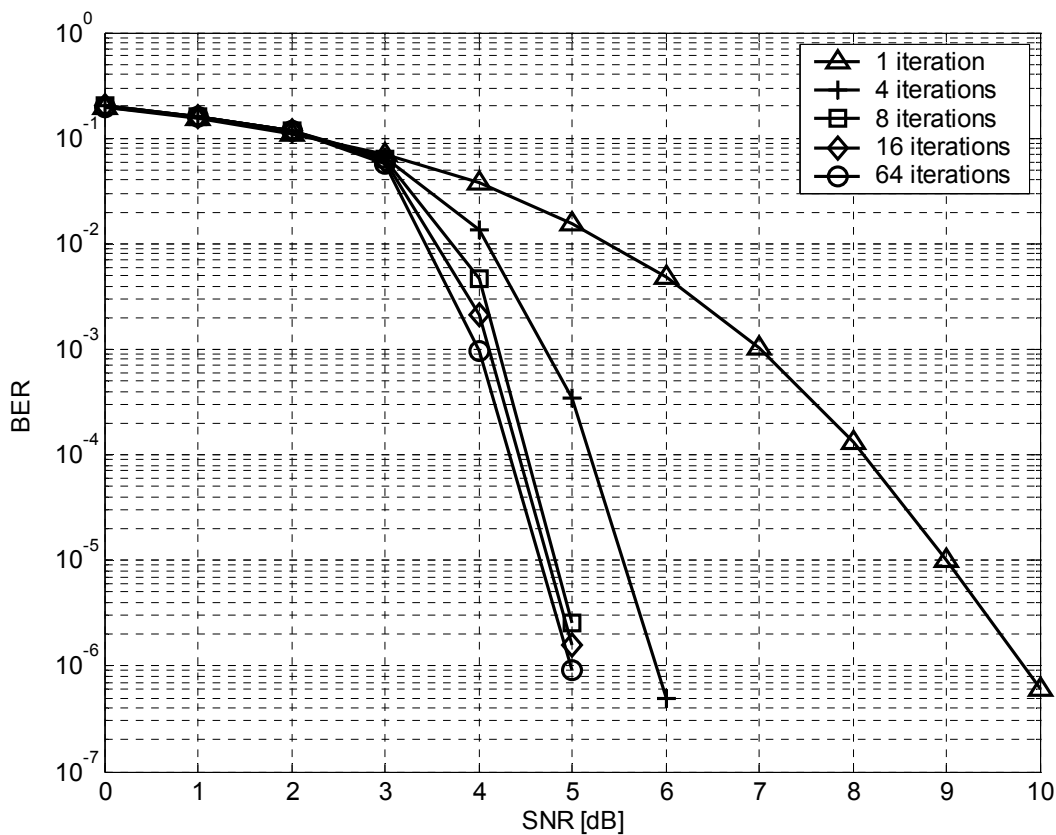


Fig. 3.6(a) BER of the (1200, 720) LDPC code

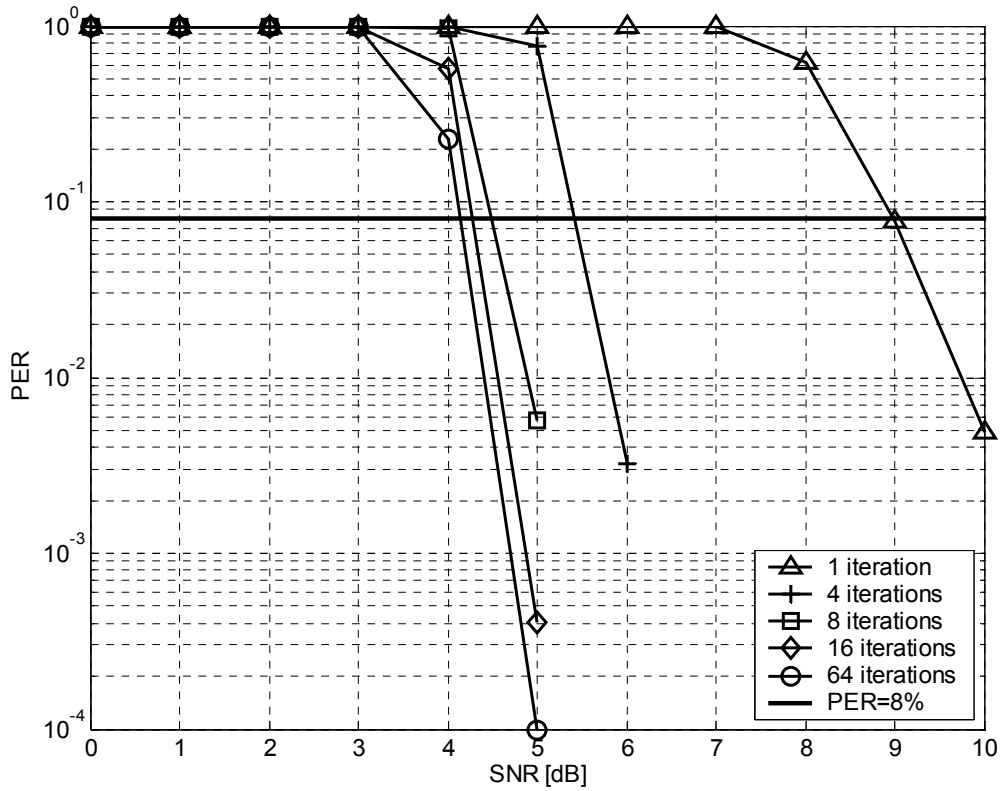


Fig. 3.6(b) PER of the (1200, 720) LDPC code

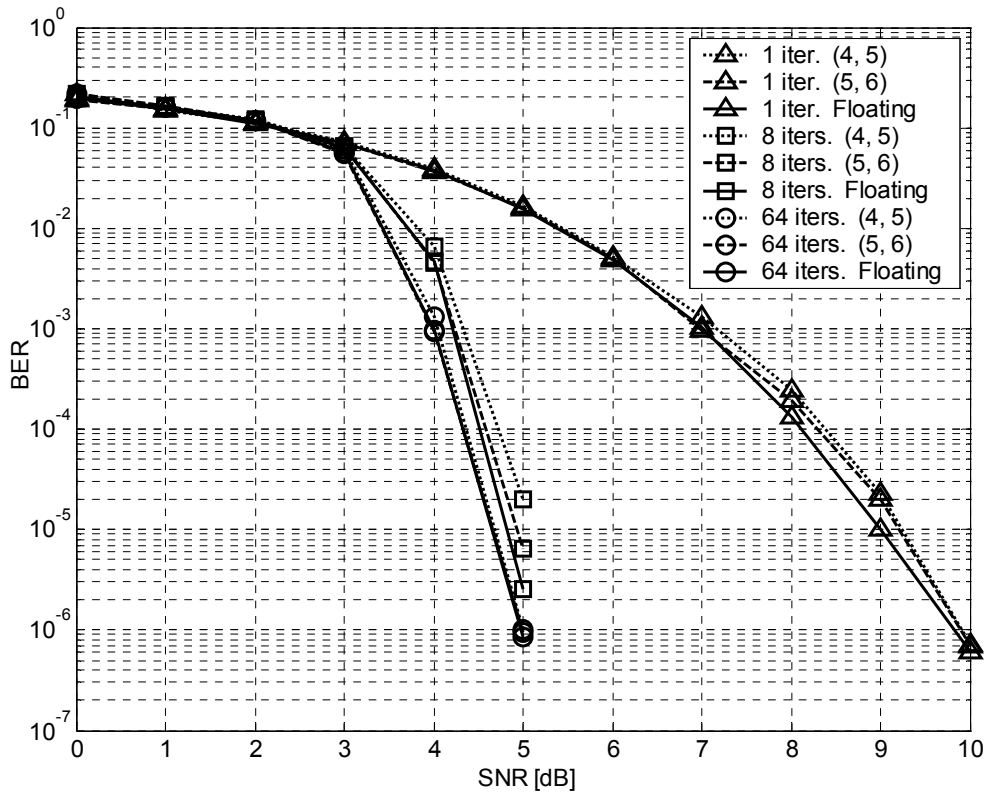


Fig. 3.7(a) Fixed point simulation of BER for the (1200, 720) LDPC code

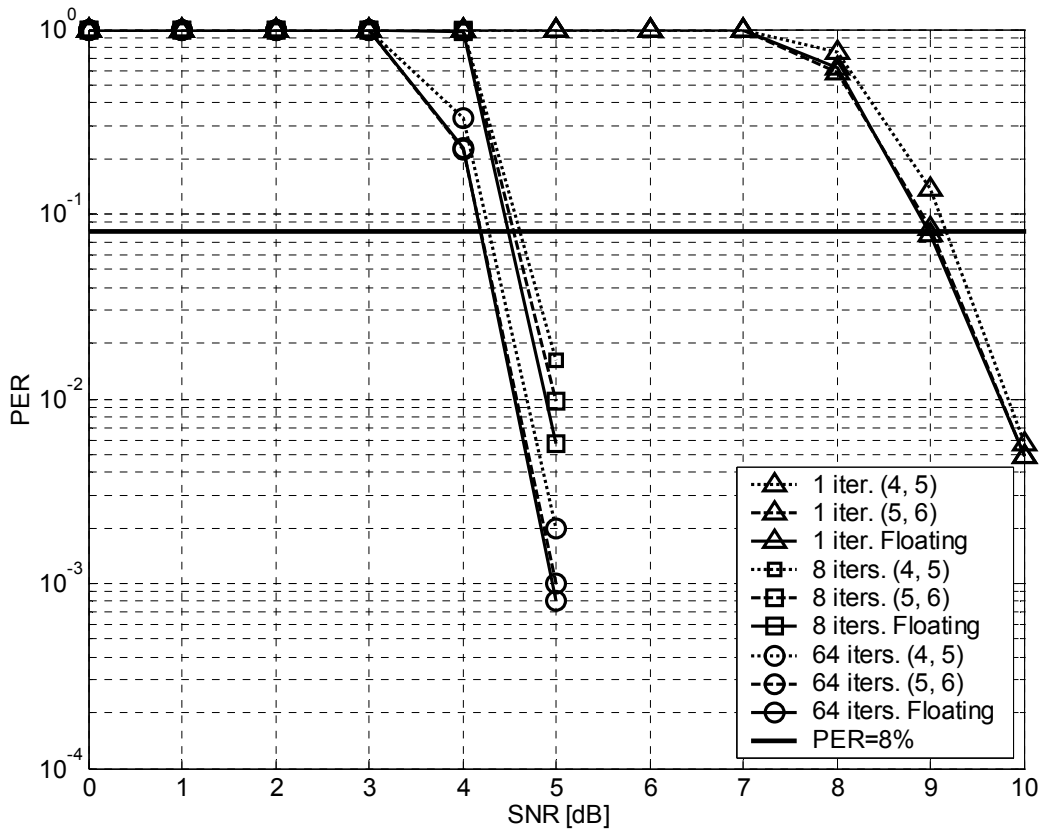
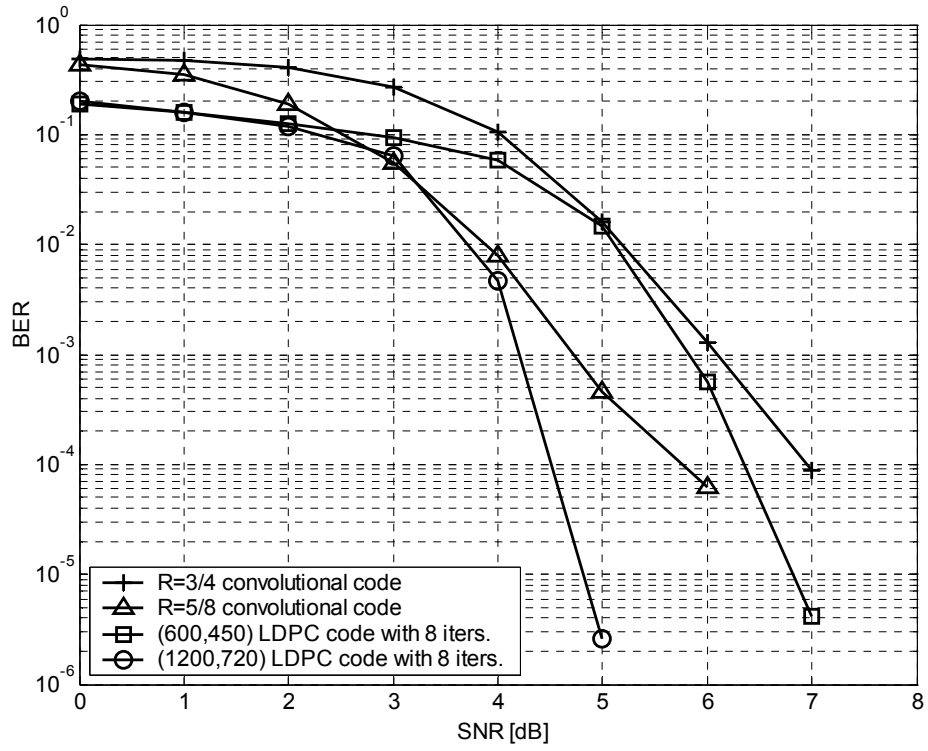


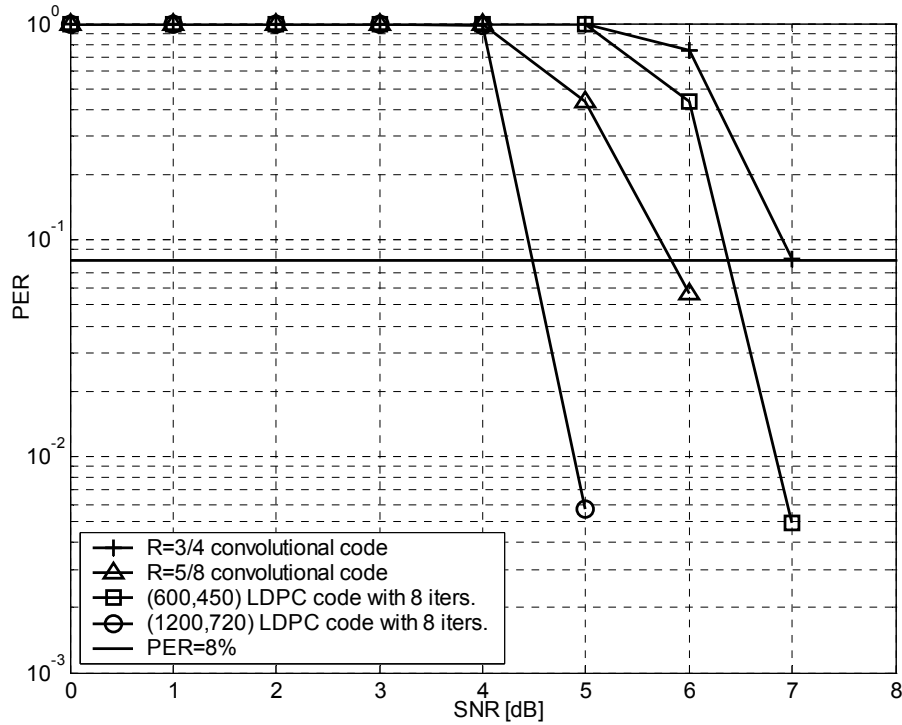
Fig. 3.7(b) Fixed point simulation of PER for the (1200, 720) LDPC code

3.2.3 Performance Comparison with Convolutional Codes

In Fig. 3.8, the performance of LDPC codes is compared to the 64-state convolutional coded system proposed in [23] where two different rates after puncturing the $R = 1/3$ convolutional code are selected as the references. It shows that both LDPC codes can outperform the convolutional codes after puncturing with only 8 iterations. The short block length and small decoding iterations will facilitate high speed implementation.



(a) BER



(b) PER

Fig. 3.8 Performance comparison for different codes

Chapter 4

Architectures of Proposed LDPC Code Decoders

The architectures of the proposed LDPC code decoders for two different LDPC codes, Code I and Code II, will be introduced in this chapter. Basic functional units, data flow rescheduling and memory arrangement methods will be discussed in detail. The measurement results of the proposed LDPC code decoder chips and a comparison with the state-of-the-art designs will also be listed. The specifications of Code I and Code II are summarized in Table 4.1.

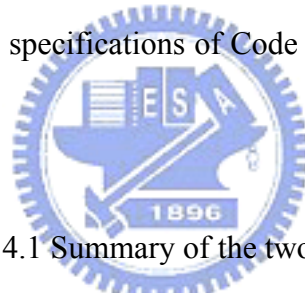


Table 4.1 Summary of the two LDPC codes

	Code I	Code II
Block length	600	1200
Information bits	450	720
Code rate	3/4	3/5
Code structure	Irregular	Irregular
Column weight	3	3
Row weight	11~14	7~9

4.1 Introduction to the Conventional Design

Based on the decoding algorithm, the block diagram of conventional LDPC code decoder is shown as Fig. 4.1. The bit node unit (BNU) is dedicated to the vertical step, while the check node unit (CNU) is used for the horizontal step. The BNU (or CNU) reads and processes the messages stored in the memory bank, and write them back into the memory bank after updating. It can be noticed that a large number of combinational feedback paths exist between the CNU (or BNU) and the memory unit, leading to the complex signal routing as well as degradation of the decoding speed in the VLSI implementation.

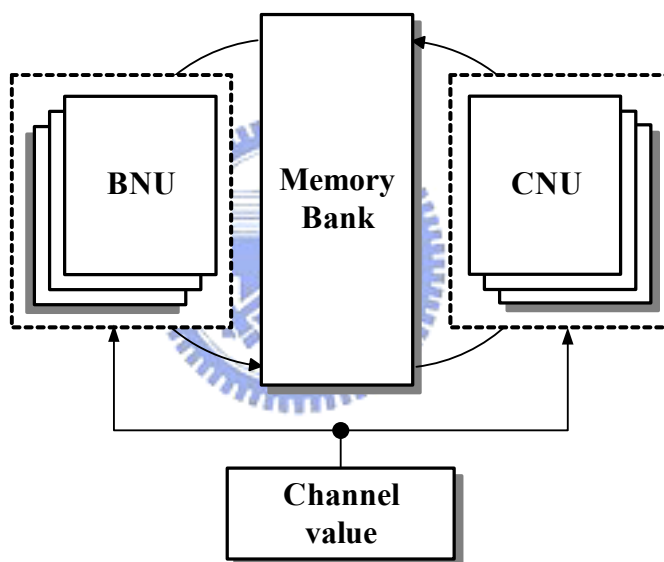
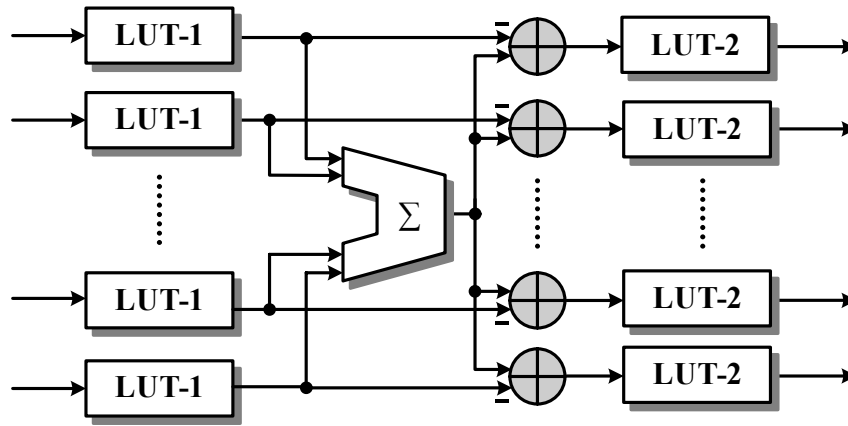


Fig. 4.1 Block diagram of conventional LDPC code decoder

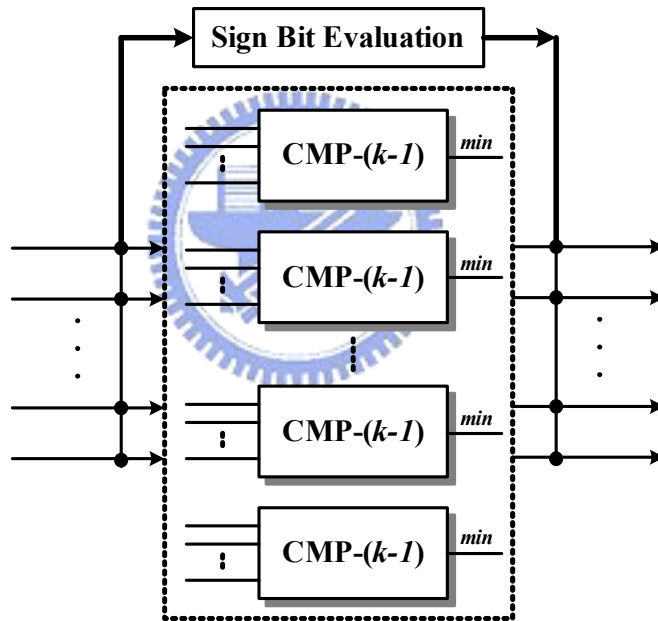
The conventional architecture of the CNU which is based on the LLR-SPA in (2.33) is shown in Fig. 4.2(a). The look-up tables (LUT) are used to implement the hyperbolic tangent (\tanh) and inverse hyperbolic tangent (\tanh^{-1}) functions.

The CNU can be implemented based on the min-sum algorithm as shown in Fig. 4.2(b) to reduce the hardware cost. As described in (2.38), the operations in the CNU can be divided into two parts: the sign evaluation and the minimum absolute value searching. The minimum

absolute values are searched by k comparators which consist of $k-1$ inputs (CMP- $(k-1)$), where k is the row weight of the parity check matrix.



(a)



(b)

Fig. 4.2 Architecture of conventional CNU based on: (a) LLR-SPA and (b) min-sum algorithm

The conventional BNU architecture with k inputs is shown in Fig. 4.3, where the SUM- $(k-1)$ is used to sum up $k-1$ values. Note that there is no difference on the BNU design between the LLR-SPA and the min-sum algorithm. Both LLR-SPA and min-sum algorithm have the same BNU design.

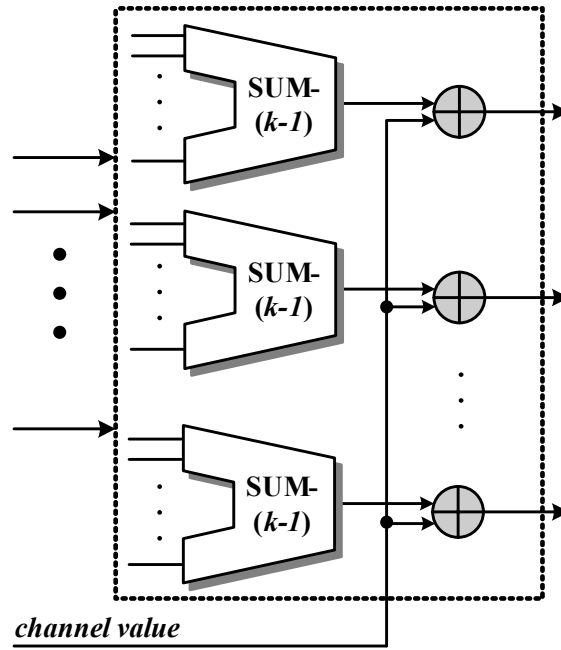


Fig. 4.3 Architecture of conventional BNU

4.2 Proposed LDPC Code 1 Decoder Design

The LDPC code decoders have inherently parallelism due to the non-dependency among check node updates or bit node updates; the throughput can be improved by linear increase of the hardware costs. However, the full-parallel implementation [9] is non-area-efficient for a system chip design. Therefore the partial-parallel architecture is employed in the proposed decoders to reduce circuit complexity according to the system requirements. In time-division multiplexing mode, the partial-parallel LDPC code decoders map a certain number of check nodes or bit nodes into a single processing unit. Extra decoding latencies are produced as compared with the full-parallel implementations. Thus a trade-off is made between the decoding speed and the hardware complexity. Besides, to simplify the hardware cost, the min-sum algorithm is chosen to implement the proposed design while keeping the system performance.

Fig. 4.4 presents the architecture of the proposed LDPC Code I decoder containing the distributor, memory unit, switch groups, CNU and BNU. Since the irregular parity check matrix H has a fixed number of column weight ($= 3$), the total number of weight in parity check matrix is $600 \times 3 = 1800$. To implement the decoder in a partial-parallel mode, the check nodes in the corresponding bipartite graph are partitioned into three parts, and the bit nodes are divided into four parts as shown in Fig. 4.5, where every three check nodes share a single CNU, and every four bit nodes share a single BNU. Therefore $150/3 = 50$ CNUs and $600/4 = 150$ BNUs are required in the proposed design. The switch groups in Fig. 4.4 are used to select which part of check nodes or bit nodes is under operation.

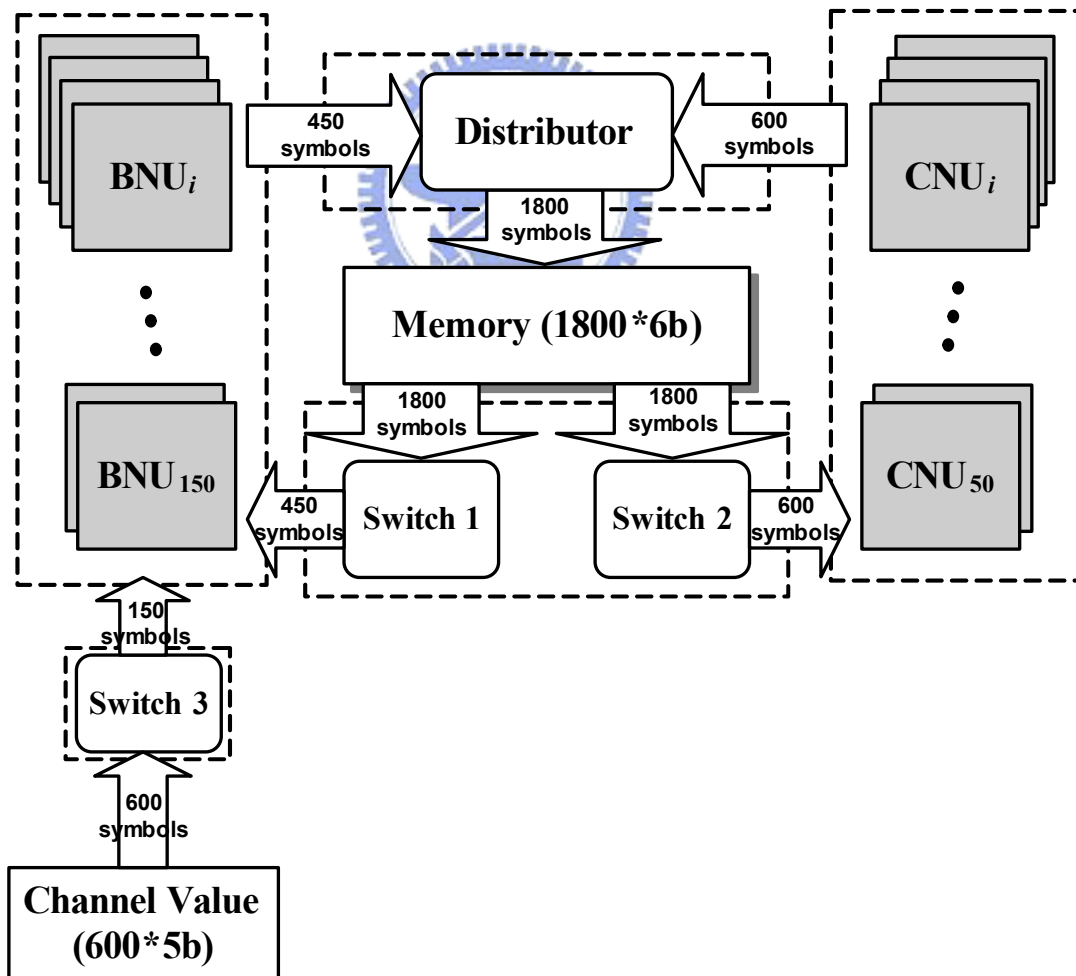


Fig. 4.4 The architecture of LDPC Code I decoder

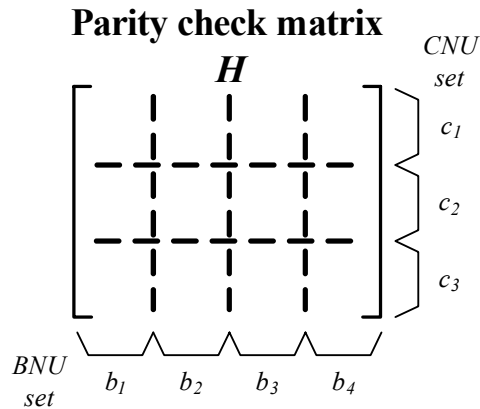


Fig. 4.5 The partition for parity check matrix H of Code I

Due to the random-like connections in the bipartite graph, the signal routing problem causes serious difficulties in the decoder implementation. As shown in Fig. 4.1, the combinational feedback paths leads to the degradation of the decoding speed and the routing area overhead in the VLSI implementation. In the proposed design, the pipeline registers are inserted in CNUs and BNUs to cut off those feedback paths as illustrated in Fig. 4.6. Thus, shorter critical path delay that reduces routing congestion can be achieved with little increases in the hardware costs.

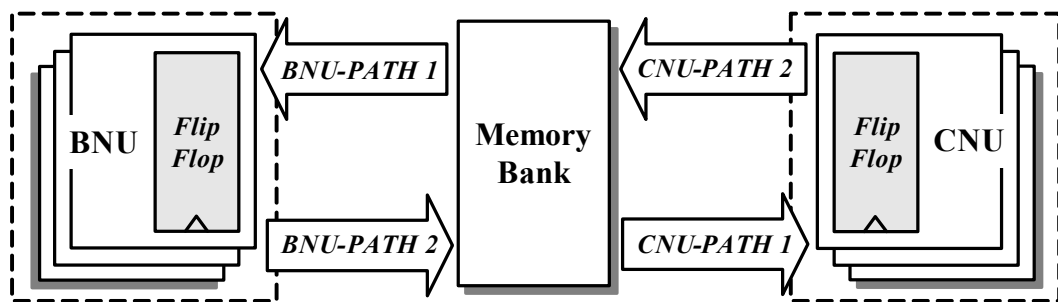


Fig. 4.6 Data path of proposed partial-parallel decoder

4.2.1 Channel Value Interconnection

For the conventional design in Fig. 4.1, both the CNUs and BNUs have to be connected to the channel values, which lead to large number of signal connections. Thus data

rescheduling is proposed to solve this problem in Fig. 4.7.

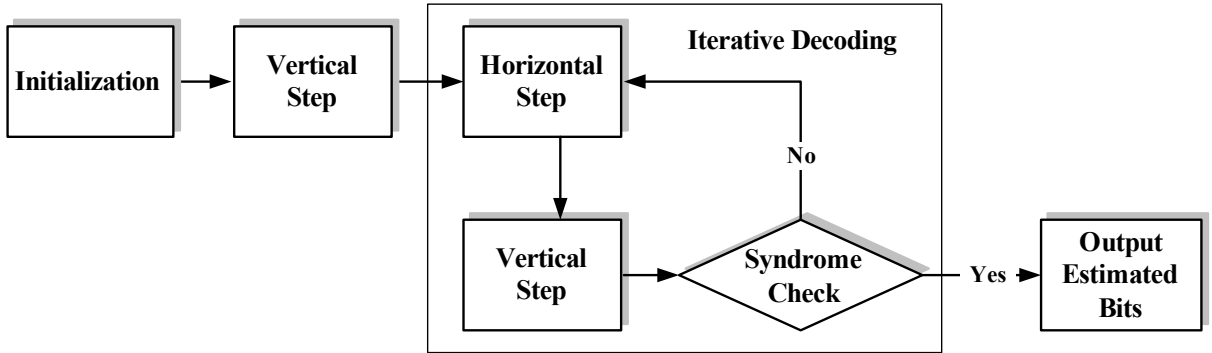


Fig. 4.7 Proposed LDPC decoding flow

As shown in Fig. 4.7, one extra vertical step is employed to replace the initialization through the CNUs. Recall equation (2.34)

$$L_{B_i \rightarrow C_j}(e_{ij}) = L(x_i) + \sum_{C_j \in M(i) \setminus C_j} L_{C_j \rightarrow B_i}(e_{ij'}), \quad (2.34)$$

only summations among the channel value $L(x_i)$ and the messages $L_{C \rightarrow B}(e_{ij})$ are performed in the BNUs. If the messages $L_{C \rightarrow B}(e_{ij})$ are set to zero during initialization, the channel values are thus loaded into the memory through the BNUs, and fed to the CNUs for the first horizontal step. In this scheme, only BNUs have to be connected to the channel values as illustrated in Fig. 4.4, leading to less signal routing costs with some increases in decoding latencies.

Fig. 4.8 gives the timing diagram of the proposed LDPC Code I decoder, where b_i and c_i correspond to the active BNU and CNU set in Fig. 4.5. The design takes nine cycles to complete a decoding iteration, including 4 cycles for horizontal steps with the CNUs and 5 cycles for vertical steps with BNUs. Additional five cycles are used to complete the channel value loading as described above. Thus total $9 \times 8 + 5 = 77$ cycles are required to finish the decoding process of a codeword with 8 iterations.

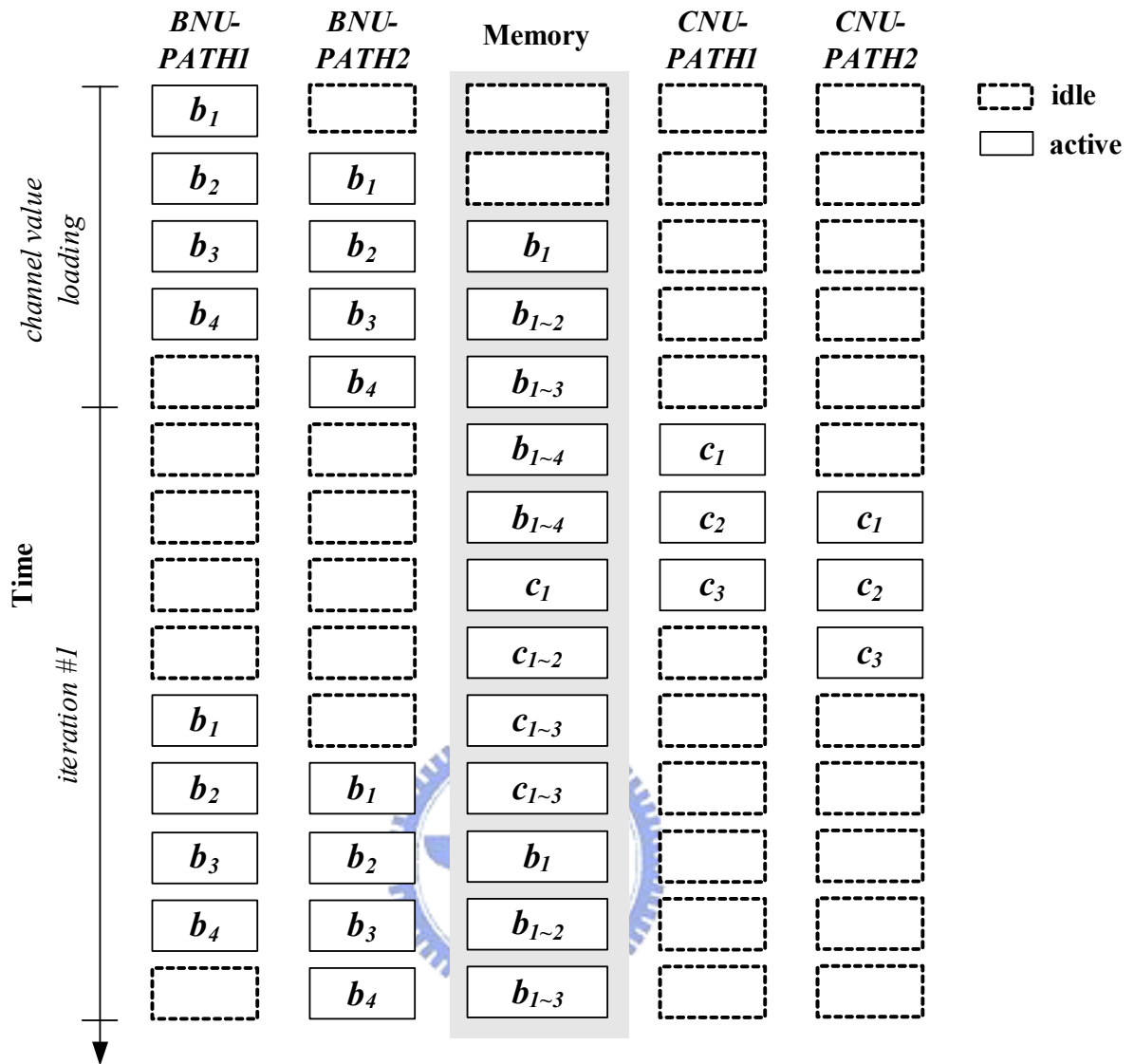


Fig. 4.8 Timing diagram of the proposed LDPC Code I decoder

4.2.2 Check Node Unit

As shown in Fig. 4.2(b), k comparators which search the minimal values among $k-1$ inputs are needed to implement the CNU based on the min-sum algorithm. As mentioned in [18], equation (2.38) can be modified as

$$\begin{aligned}
L_{C_j \rightarrow B_i}(e_{ij}) &\approx \left(\prod_{B_i \in L(j) \setminus B_i} \text{sign}(L_{B_i \rightarrow C_j}(e_{ij})) \right) \min_{B_i \in L(j) \setminus B_i} (|L_{B_i \rightarrow C_j}(e_{ij})|) \\
&= \left(\prod_{B_i \in L(j) \setminus B_i} \text{sign}(L_{B_i \rightarrow C_j}(e_{ij})) \right) \times \begin{cases} \min_{B_i \in L(j)} (|L_{B_i \rightarrow C_j}(e_{ij})|) & , \text{ if } |L_{B_i \rightarrow C_j}(e_{ij})| \neq \min_{B_i \in L(j)} (|L_{B_i \rightarrow C_j}(e_{ij})|) \\ 2^{\text{nd}} \min_{B_i \in L(j)} (|L_{B_i \rightarrow C_j}(e_{ij})|) & , \text{ otherwise} \end{cases} \quad (4.1)
\end{aligned}$$

where “2nd min” denotes the value which is smaller than all the other candidates except the minimal one. According to (4.1), the absolute value searching has to be performed only one time to find the minimum and the second minimum. Fig. 4.9 shows the block diagram of the compare-select unit (CS14) which searches for the minimal and the second minimal values from 14 inputs.

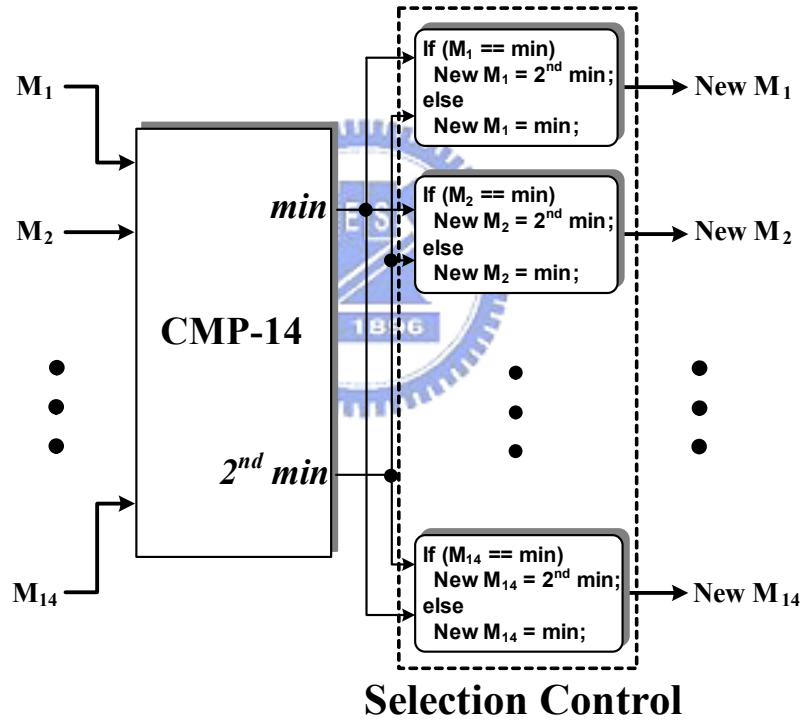


Fig. 4.9 Block diagram of CS14

Because the column weight of Code I is ranging from 11 to 14, the CNUs dealing with different number of inputs should be designed. In this section, only the 14-input CNUs are introduced and others are designed in the analogous approach. The detailed architecture of CMP-14 in Fig. 4.9 is illustrated as Fig. 4.10, which consists of the pipeline registers and two

kinds of comparators: CMP-2 and CMP-4. CMP-4 finds out the minimal and the second minimal values from the four inputs, a, b, c, and d. In addition, CMP-2 is a two input comparator which is much simpler than CMP-4.

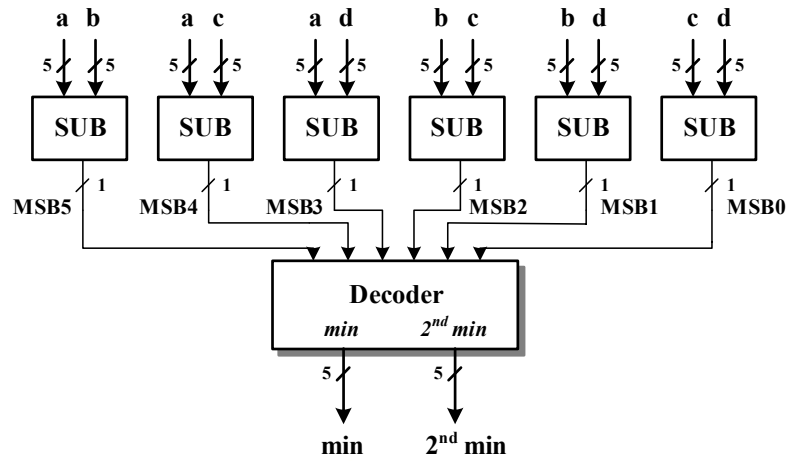


Fig. 4.10(a) Block diagram of proposed CMP-4

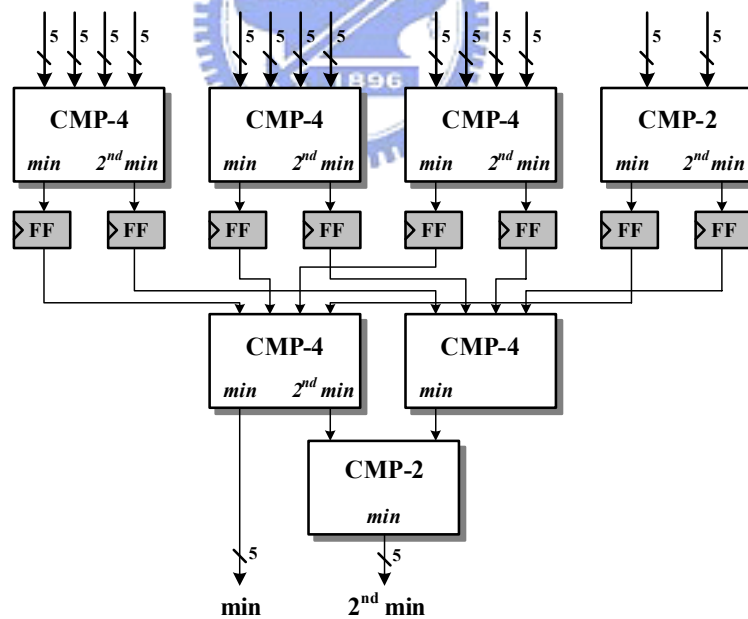


Fig. 4.10(b) Block diagram of proposed CMP-14

The proposed architecture of the 14-input CNU is shown in Fig. 4.11, where SM14 is sign-multiplication. To facilitate the operations on the sign and absolute value, all the 6-bit

values have been represented by the sign-magnitude notation with 2 integer bits and 4 fractional bits. The combinational path in the CNUs is cut off into CNU-PATH1 and CNU-PATH2 by the pipeline registers, leading to shorter critical path delay that reduces routing congestion.

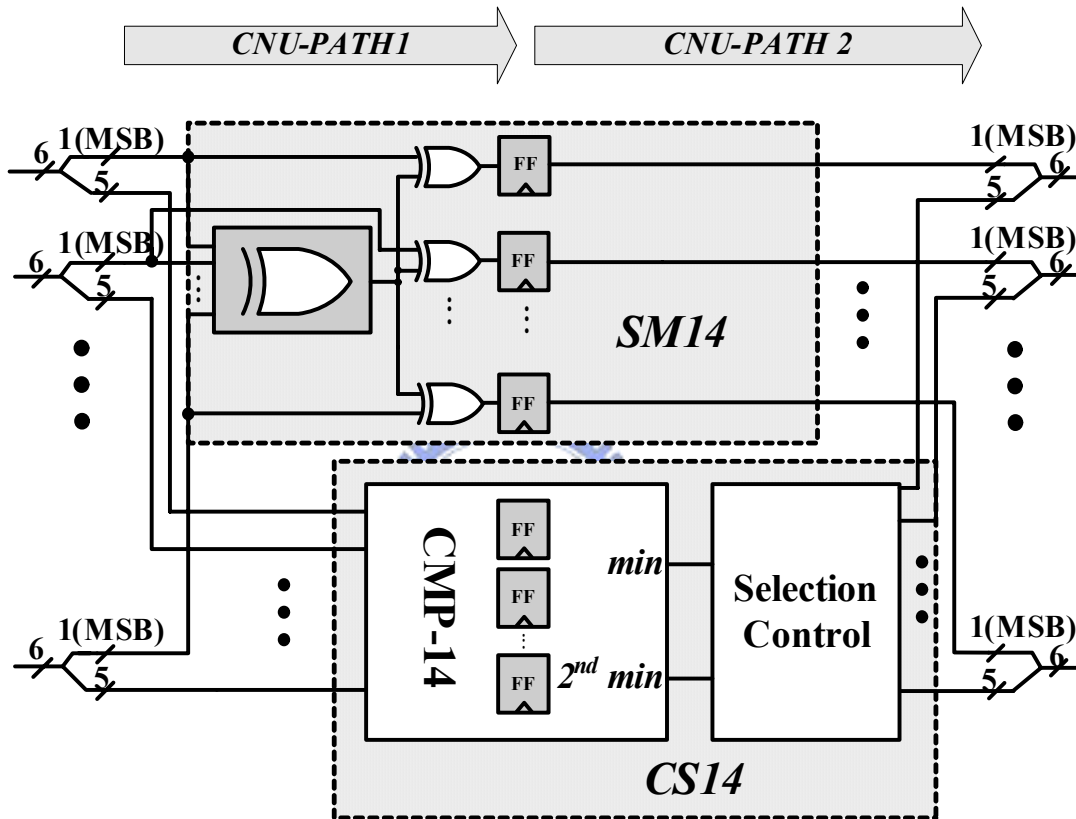


Fig. 4.11 The proposed 14-input CNU architecture

Table 4.2 lists the comparisons of three different CNU architectures. The LUT-1 and LUT-2 in Fig. 4.2(a) are implemented in 6-bit precision, including 2 integer bits and 4 fractional bits. The proposed CNU has the smallest size which is only about 22% of the others, whereas the maximum achievable operating speed is only a little smaller than conventional MS designs. Due to the fixed point implementation, some performance loss is produced. As a result, the decoder is implemented efficiently by using of the proposed CNU architecture.

Table 4.2 Comparison of different CNU architectures

	LUT <i>Fig. 4.2(a)</i>	Conv. MS <i>Fig. 4.2(b)</i>	Proposed <i>Fig. 4.11</i>
Max. speed	162 MHz	261 MHz	250 MHz
Gate count	7.16 K	6.86 K	1.6 K
Total gate count	358 K	343 K	80 K

4.2.3 Bit Node Unit

Fig 4.12 shows the block diagram of BNU. According to equation (2.34) and (2.35), the BNUs receive the channel value and the message values linked to the same bit node. All inputs with sign-magnitude (SM) notation are converted to be 2's complement (TC) representation, and summed to perform the updating calculation. The pipeline registers are inserted to break the critical paths into BNU-PATH1 and BNU-PATH2 as in the CNU. Finally, all the values are converted back to the SM notation and clipped to avoid overflow. And the most significant bit (MSB) of the summation of the three input messages and the channel value is used to decide the estimated codeword bit.

All the 6-bit values are quantized with 2 integer bits and 4 fractional bits, while the intermediate summations are represented with 4 integer bits and 4 fractional bits.

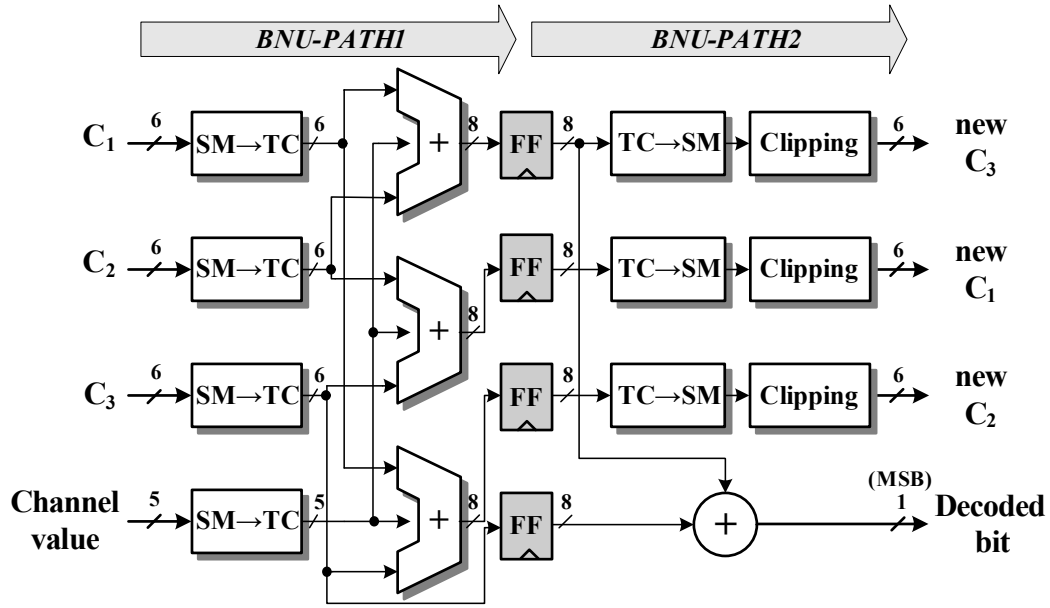
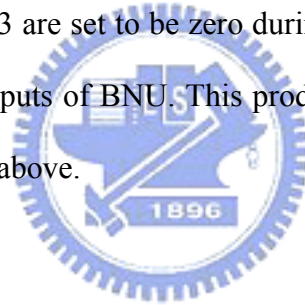


Fig. 4.12 The proposed BNU architecture

Note that if C_1 , C_2 and C_3 are set to be zero during initialization, the channel value will be directly bypassed to the outputs of BNU. This produces a path to load the channel values into the memory as mentioned above.



4.2.4 Chip Implementation

The proposed LDPC Code I decoder was implemented within an LDPC-COFDM UWB baseband transceiver chip [25] with the $0.18 \mu\text{m}$ 1P6M standard CMOS process. The chip micrograph of the entire UWB transceiver including the OFDM modem and the LDPC codec is given in Fig. 4.13. The encoder die size is 2.25 mm^2 , while the decoder die size is 16.5 mm^2 . The total gate count of the LDPC codec is 542 K, where 70K is for the encoder and 472K is for the decoder.

The chip has been tested to verify the functional correctness. The measured maximal data rate of the decoder is 480 Mb/s while working at 82.1 MHz, and consuming 232 mW. The detailed chip features are also summarized in Table 4.3.



Fig. 4.13 Die micrograph of the LDPC-COFDM UWB transceiver chip

Table 4.3 Summary of the LDPC Code I Chip

Technology		Standard 0.18- μm CMOS 1P6M
Package		CQFP-208
Supply voltage		1.8V core, 3.3 V I/O
Chip size	Encoder	1.5mm \times 1.5mm
	Decoder	5.0mm \times 3.5mm
Gate count	Encoder	70K
	Decoder	472K
Power dissipation		232mW @ 82.1MHz
Maximum data rate		480Mb/s

4.3 Proposed LDPC Code II Decoder Design

In Sec. 4.2, the proposed LDPC Code I decoder design is introduced and silicon proven to achieve 480Mb/s maximum data rate. The performance of LDPC code I decoder is acceptable for the MB-OFDM UWB system [23], but may be not for other high-speed communication systems mentioned in Chap. 3. As a result, the LDPC code II decoder is proposed to get better error-correcting ability and higher decoding throughput.

While considering circuit complexity, the 480×1200 parity check matrix \mathbf{H} of LDPC code II are divided into four 240×600 sub-matrixes to fit partial-parallel architecture, which is shown in Fig. 4.14. Since matrix \mathbf{H} of Code II has a fixed number of column weight ($= 3$), the total number of weight is $1200 \times 3 = 3600$. Based on this partition, the functional units in the decoder will process 1800 messages every cycle.

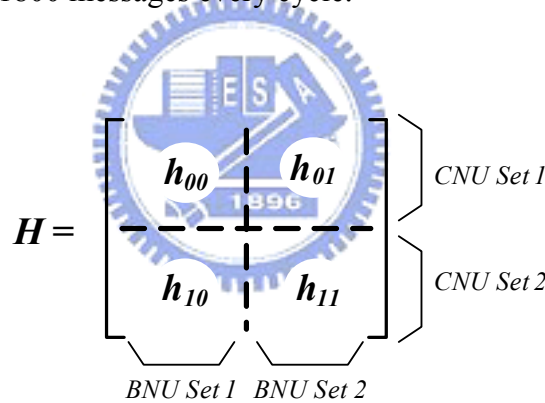


Fig. 4.14 The partition of parity check matrix \mathbf{H} of Code II

The proposed LDPC code II decoder architecture illustrated in Fig. 4.15 contains the input buffer, 240 CNUs, 600 BNUs and two dedicated message memory units (MMU). The set of data processed by CNUs are $\{h_{00}, h_{01}\}$ and $\{h_{10}, h_{11}\}$, whereas the data fed into BNUs should be $\{h_{00}, h_{10}\}$ and $\{h_{01}, h_{11}\}$. Note that two MMUs are employed to process two different codewords concurrently without stalls. Therefore, the LDPC decoder is not only area-efficient but the decoding speed is compatible with the fully parallel architecture. The detail ideas about the designs of MMUs will be introduced in the following.

The input buffer is a storage component that receives and keeps channel values for iterative decoding. Note that it only connects to the BNUs to get less routing congestion as discussed in Sec. 4.2.1.

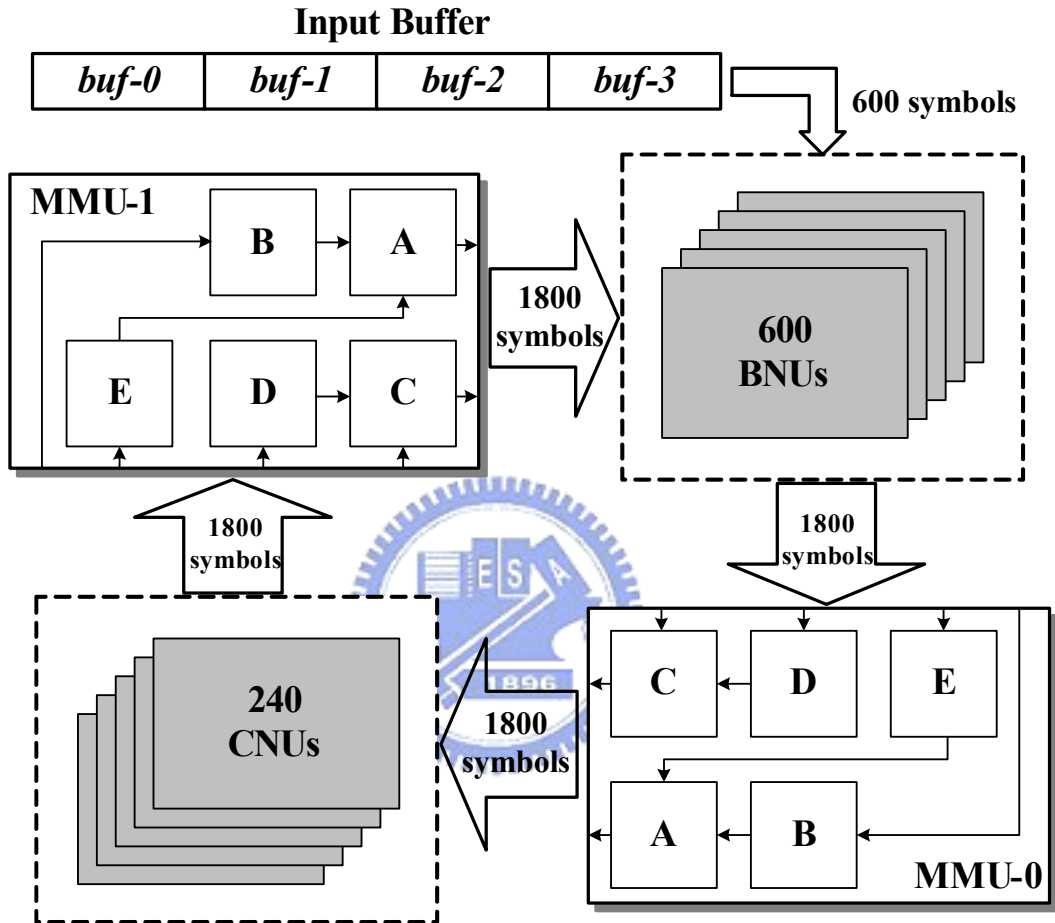


Fig. 4.15 The proposed LDPC code II decoder architecture

4.3.1 Input Buffer

Input buffer provides the channel values to the BNUs for iterative decoding. Because two different codewords are processed concurrently, total $1200 \times 2 = 2400$ symbols should be stored in the input buffer. According to the partition in Fig. 4.14, the buffer is divided into four sub-blocks, where each sub-block contains 600 channel values. The conventional design is illustrated in Fig. 4.16. Four sub-blocks, $buf-0 \sim buf-3$, are all connected to the channel

value inputs, and multiplexers are employed to switch appropriate values into the BNUs. Thus the signal routings are all “global”, meaning that all the connections are related to the inputs and outputs (I/O) of the buffer. The global connections and the multiplexers will lead to serious routing congestion.

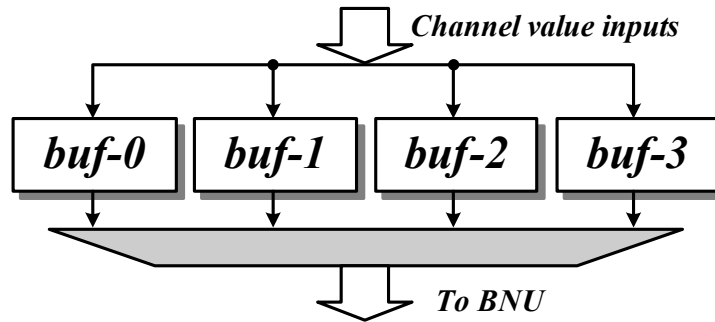


Fig. 4.16 The conventional architecture of input buffer

Fig. 4.17 shows the buffer structure based on register exchange (RE) approach and the operational timing diagram, where *buf-0* is designed as a shift register that serially receives the channel values from inputs and the other three sub-blocks exchange the data with *buf-0* sequentially. The notation E1, E2 and E3 represent the data exchange from *buf-0* to *buf-1*, *buf-2* and *buf-3*, respectively. During initialization, *buf-0* serially receives the channel values and passes them into other sub-blocks by executing the operations E1, E2 and E3 when *buf-0* is full-filled.

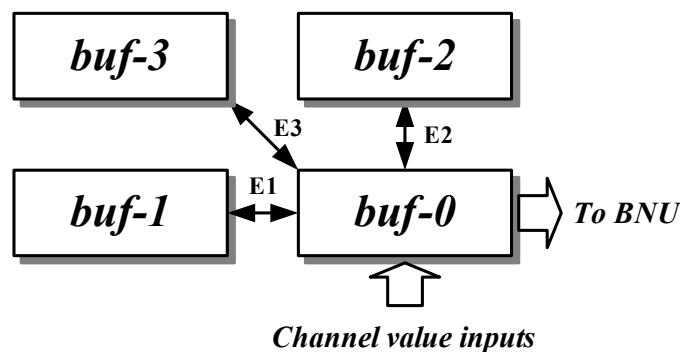


Fig. 4.17(a) The architecture of RE based input buffer

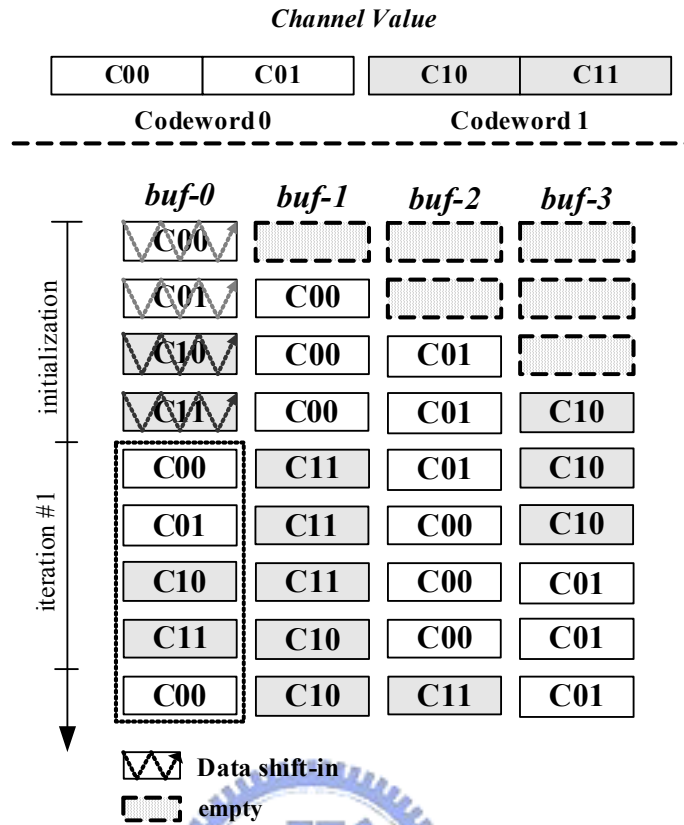
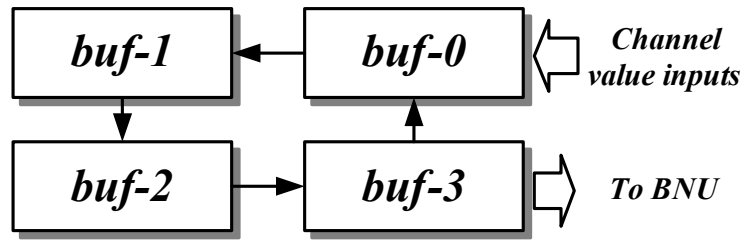


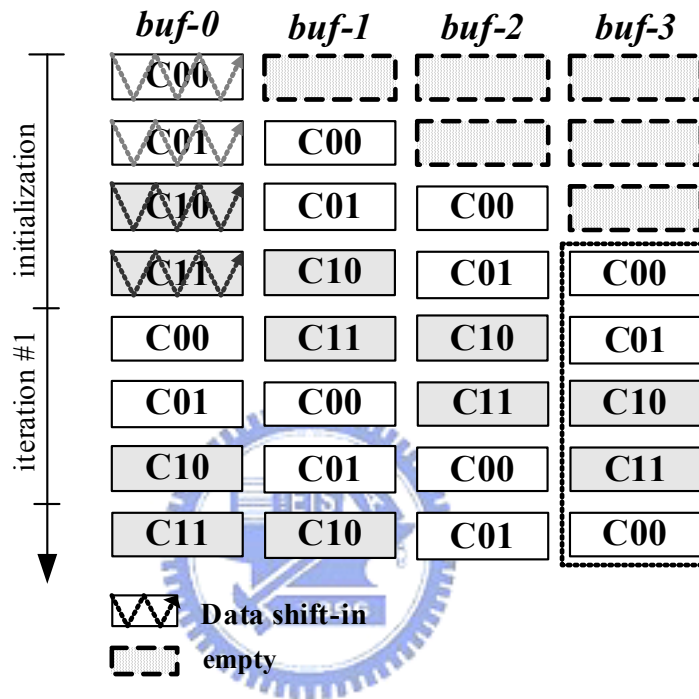
Fig. 4.17(b) The timing diagram of RE based input buffer

For this RE based buffer architecture, the global interconnections exist only in *buf-0*, and all the others are “local” among sub-blocks. However, the drawback is that a large number of multiplexers are required around *buf-0* to perform E1 ~ E3. Thus *buf-0* becomes a routing-critical block due to the multiplexers and the global interconnections.

To overcome this problem, an architecture based on register shifting (RS) is proposed as shown in Fig. 4.18(a), where four sub-blocks are arranged in a ring. The *buf-0* is a shift register that serially receives the channel values and *buf-3* transports the associated channel values to BNU. The timing diagram of the RS-based input buffer is presented in Fig. 4.18(b). Channel values of two different codewords are serially fed into *buf-0*, and shifted within the buffer ring when *buf-0* is full-filled. Therefore, the data flow is further simplified, and the multiplexers are eliminated, leading to simple signal transfer and routing interconnections.



(a)



(b)

Fig. 4.18 The architecture and timing diagram of RS-based input buffer

Fig. 4.19 gives the comparison of the three input buffer architecture. The RS-based input buffer can save about 20% gate count and 30% interconnection wires as compared with the conventional design.

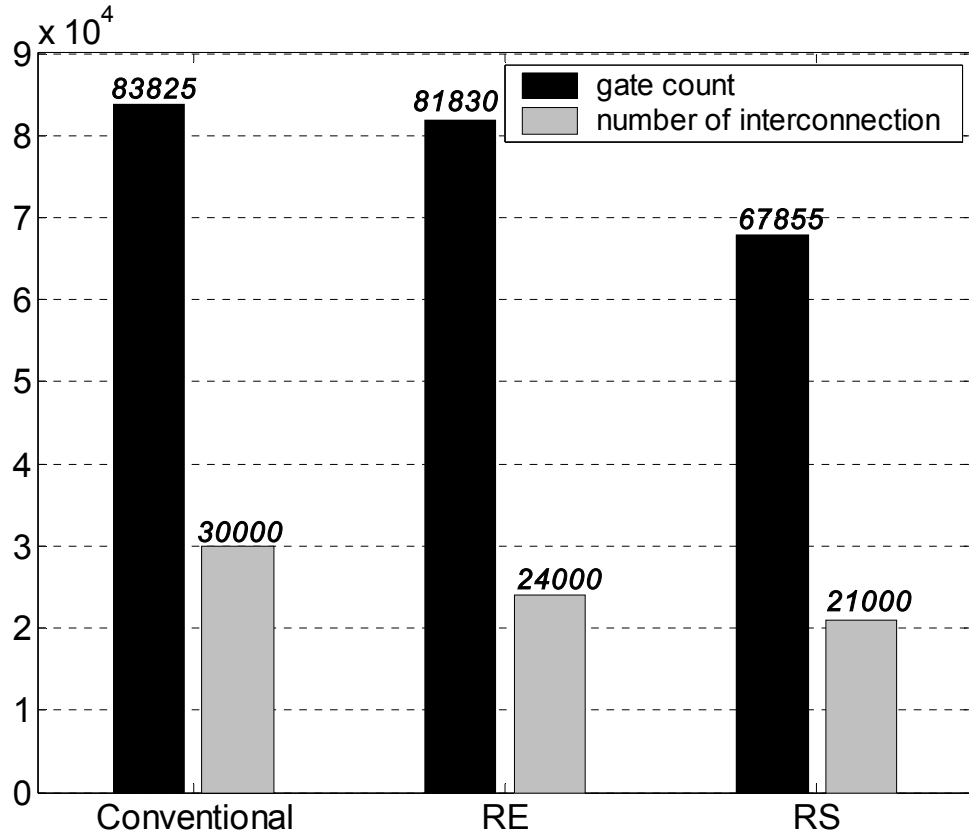


Fig. 4.19 The comparison of three input buffer designs



4.3.2 Check Node Unit and Bit Node Unit

Fig. 4.20 shows the CNU architecture for proposed LDPC code II decoder. The CNU can be divided into two parts: one is 1-bit sign-multiplication (SM) and the other is 5-bit compare-and-select unit (CS) that searches the minimal value and the second minimal value from the inputs. The new message for each bit node is a combination of the sign bit according to (4.1) and the new magnitude which is either “*min*” or “*2nd min*” of the CS unit. The detailed architecture of CMP-9 in Fig. 4.20 is designed as that shown in Fig. 4.9 and 4.10.

The BNU architecture is illustrated in Fig. 4.21. According to (2.34) and (2.35), BNU receives the channel value and the messages linked to the same bit node. All inputs with sign-magnitude (SM) notation are firstly converted to be 2’s complement (TC) representation, and then summed to perform the updated calculation. The summed values are also clipped to

avoid overflow. Finally, the MSB of the summation of all the inputs is used to decide the decoded bit.

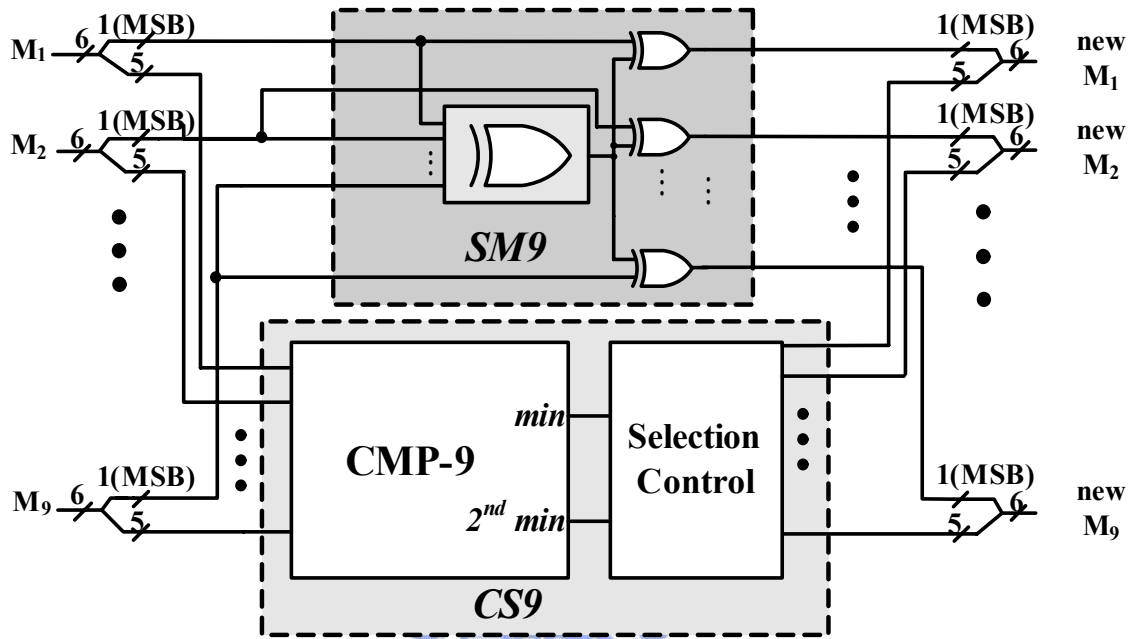


Fig. 4.20 CNU architecture of proposed LDPC Code II decoder

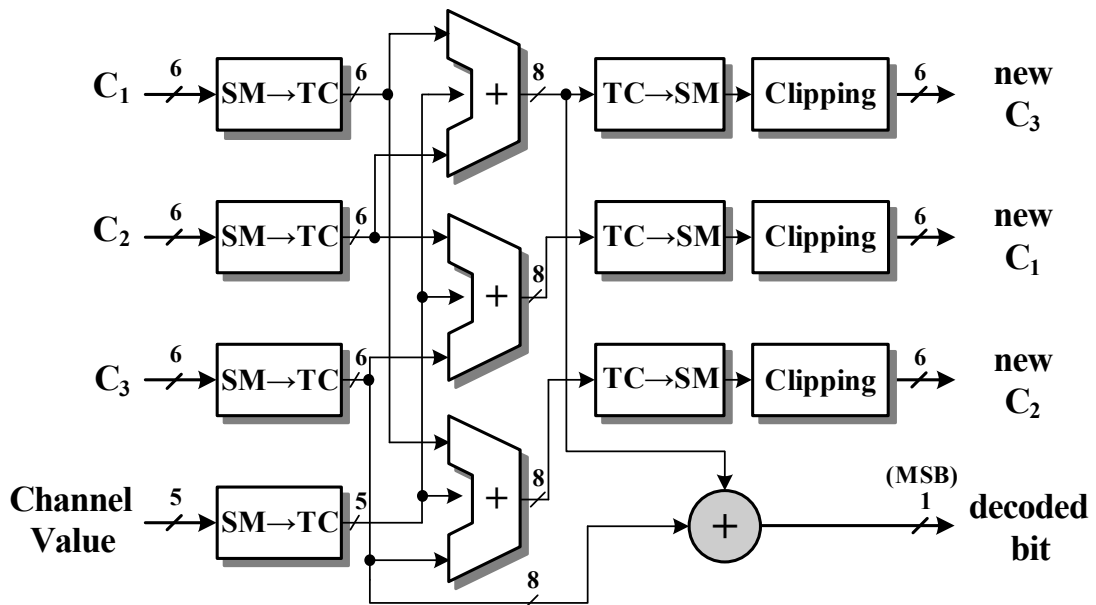


Fig. 4.21 BNU architecture of proposed LDPC Code II decoder

4.3.3 Message Memory Unit

Message memory units (MMU) are used to store the message values that are generated by CNUs or BNUs. The size of each MMU is 3600×6 bit due to the weight of the parity check matrix. To increase the decoding throughput, two MMUs are employed to concurrently process two different codewords in the decoder. The memory management strategies, described below, include multiplexers (MUX) or register exchange (RE), resulting in different level of routing complexity. The MUX based MMU architecture and the timing diagram are illustrated in Fig. 4.22.

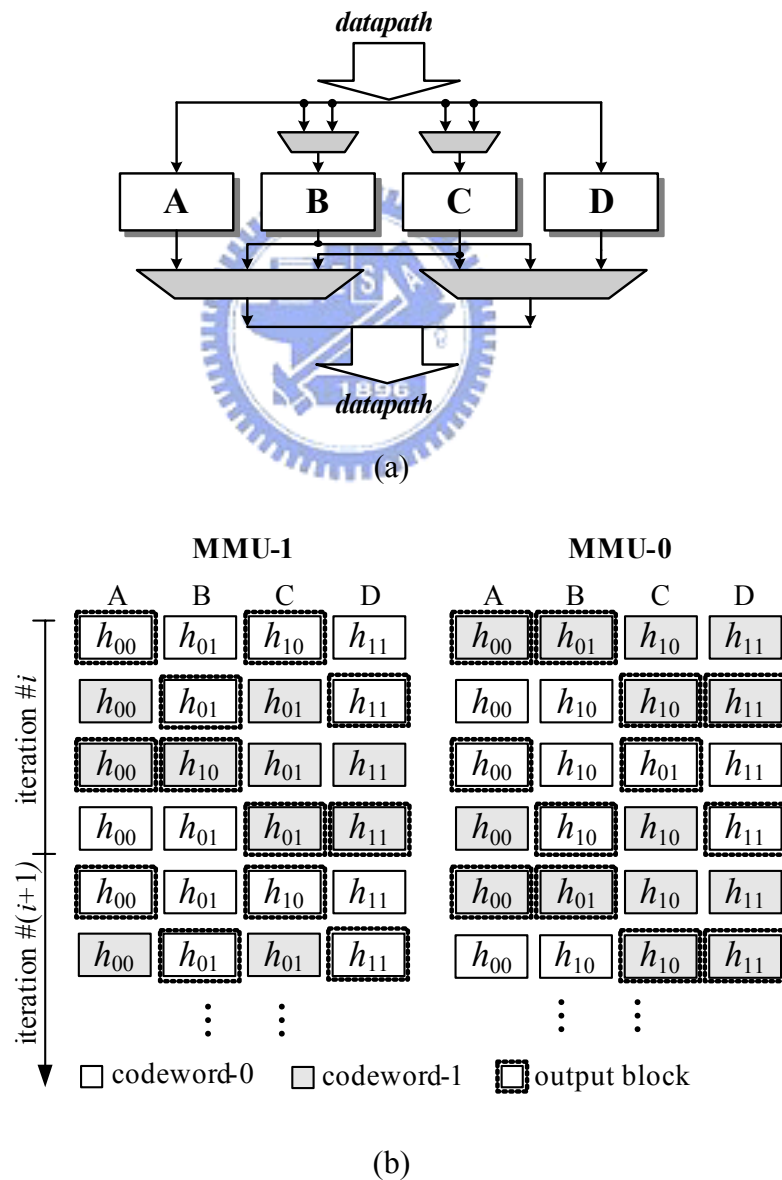


Fig. 4.22 The architecture and timing diagram of MUX-based MMU

According to the partition of the matrix H in Fig. 4.14, the MMU is divided into four sub-blocks: A, B, C and D. Many multiplexers are required for the inputs and outputs due to the partially parallel implementation and the concurrent process of two different codewords. Moreover, all the signal interconnections are related to the I/O, leading to global routings. As a result, the serious routing congestion occurs in the conventional MMU design.

To release the routing congestion problem, the architecture based on register exchange among four sub-blocks (RE-4B) is proposed as shown in Fig 4.23. In this design, only sub-blocks B, C and D capture data from data paths, and only sub-blocks A and C connect to the outputs. Thus most of global routings are transformed into local interconnections between sub-blocks, leading to a simple data flow. Moreover, the number of multiplexers is also reduced by the RE-4B based architecture.

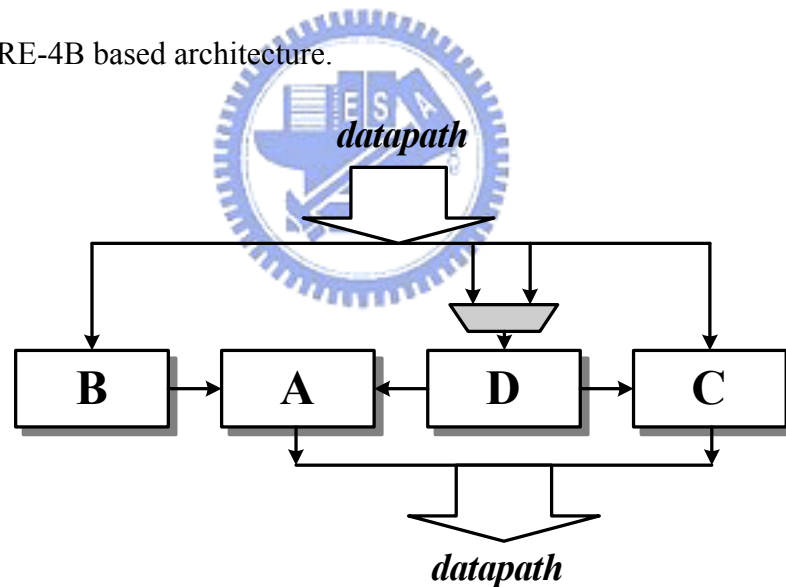


Fig. 4.23(a) The architecture of RE-4B based MMU

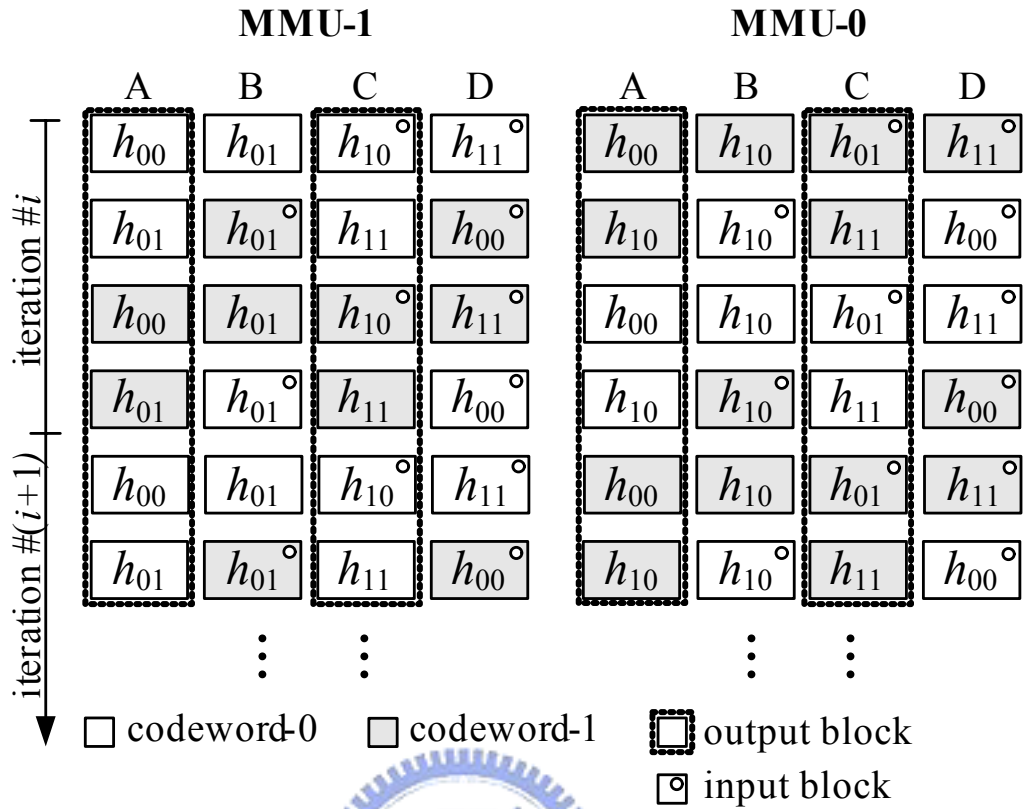


Fig. 4.23(b) The timing diagram of RE-4B based MMU

To further improve the MMU design, the register exchange scheme based on five sub-blocks (RE-5B) is proposed as shown in Fig. 4.24(a). One extra sub-block E is used as temporal memory for reducing the interconnection between other sub-blocks. In MMU-1, sub-blocks B, C, D and E capture the outputs from CNUs while sub-blocks A and C deliver the message data to BNUs. Fig. 4.24(b) shows the detailed timing diagram of reordering data sequence in MMU. The inputs of BNUs (CNUs) sequentially appear in sub-blocks A and C after reordering the data from CNUs (BNUs). Note that the solution to switch data sequence also enables the decoder to process two different codewords without stalls.

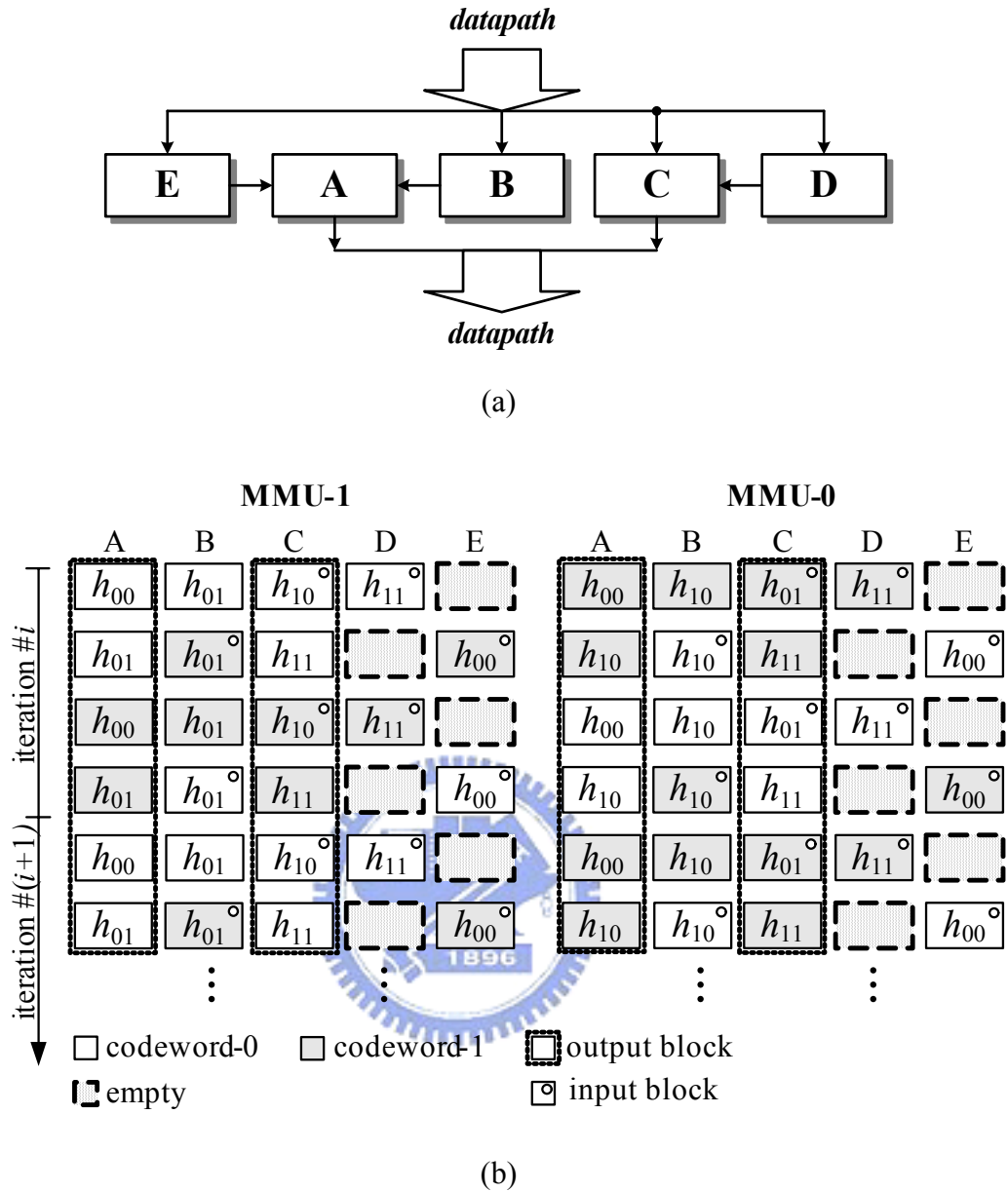


Fig. 4.24 The architecture and timing diagram of RE-5B based MMU

With the RE-5B based MMU architecture, the multiplexers between the MMUs and the data paths are eliminated. And most global interconnections are replaced by local routing between sub-blocks to reduce routing congestion. Fig. 4.25 shows a comparison among the three MMU schemes. The gate count and interconnection are measured only from MMU-0 and MMU-1, whereas the routing congestion overflow is investigated through implementing the decoder in a 25mm² 0.18- μ m chip with 6 metal layers. In RE-4B and RE-5B architectures,

there is a 15% ~ 23% decrease in gate count due to the removal of multiplexers. A significant drop in signal connections is also observed with RE approach; therefore, the routing congestion can be dramatically improved.

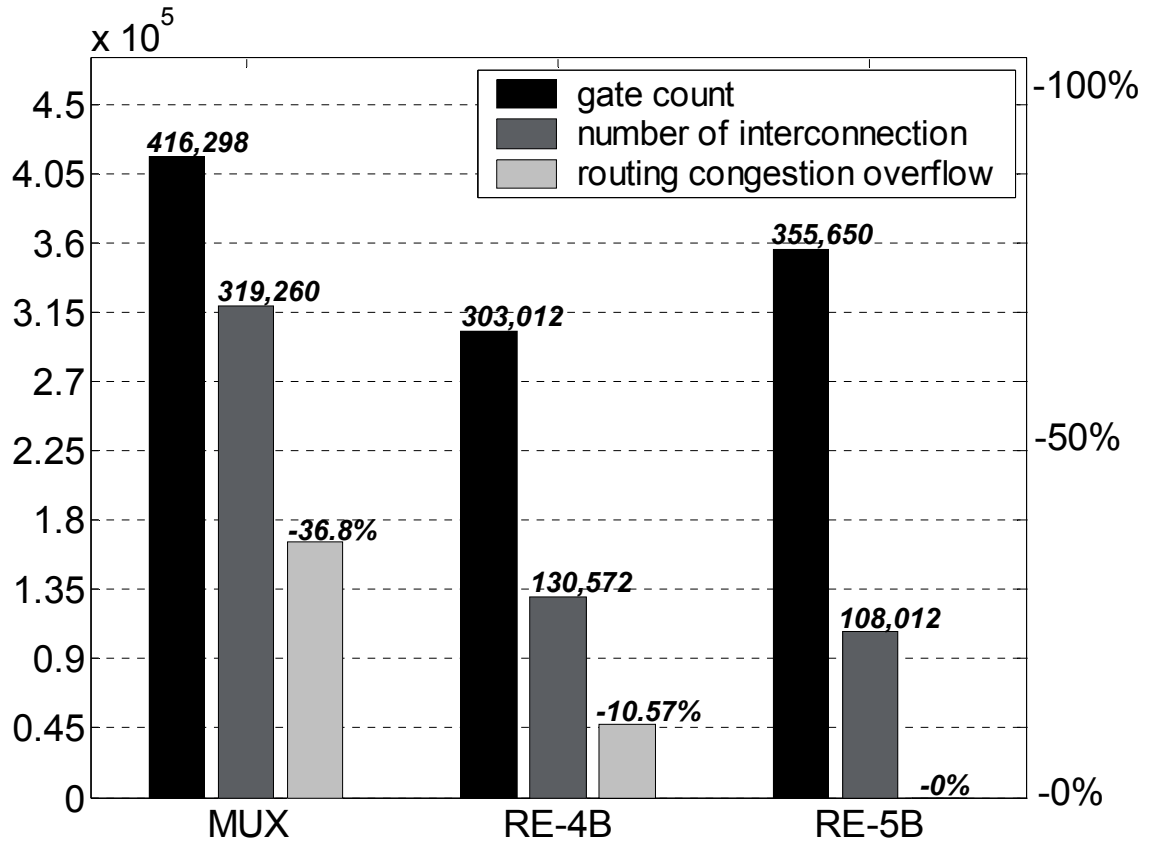


Fig. 4.25 Comparison of three MMU designs

4.3.4 Timing Schedule

The overall timing diagram of the decoder is shown in Fig. 4.26. As mentioned above, two different codewords are processed concurrently without any stalls. In our proposed design, the BNUs and CNUs have no idle time, leading to an efficient utilization of the functional units. The design takes four cycles to complete a decoding iteration for each codeword, including 2 cycles for horizontal steps in CNUs and 2 cycles for vertical steps in BNUs. For channel value loading, each codeword takes 2 extra cycles. Thus total $2 + 2 + 8 \times 4 = 36$ cycles

are required to finish the decoding of two different codewords with 8 decoding iterations.

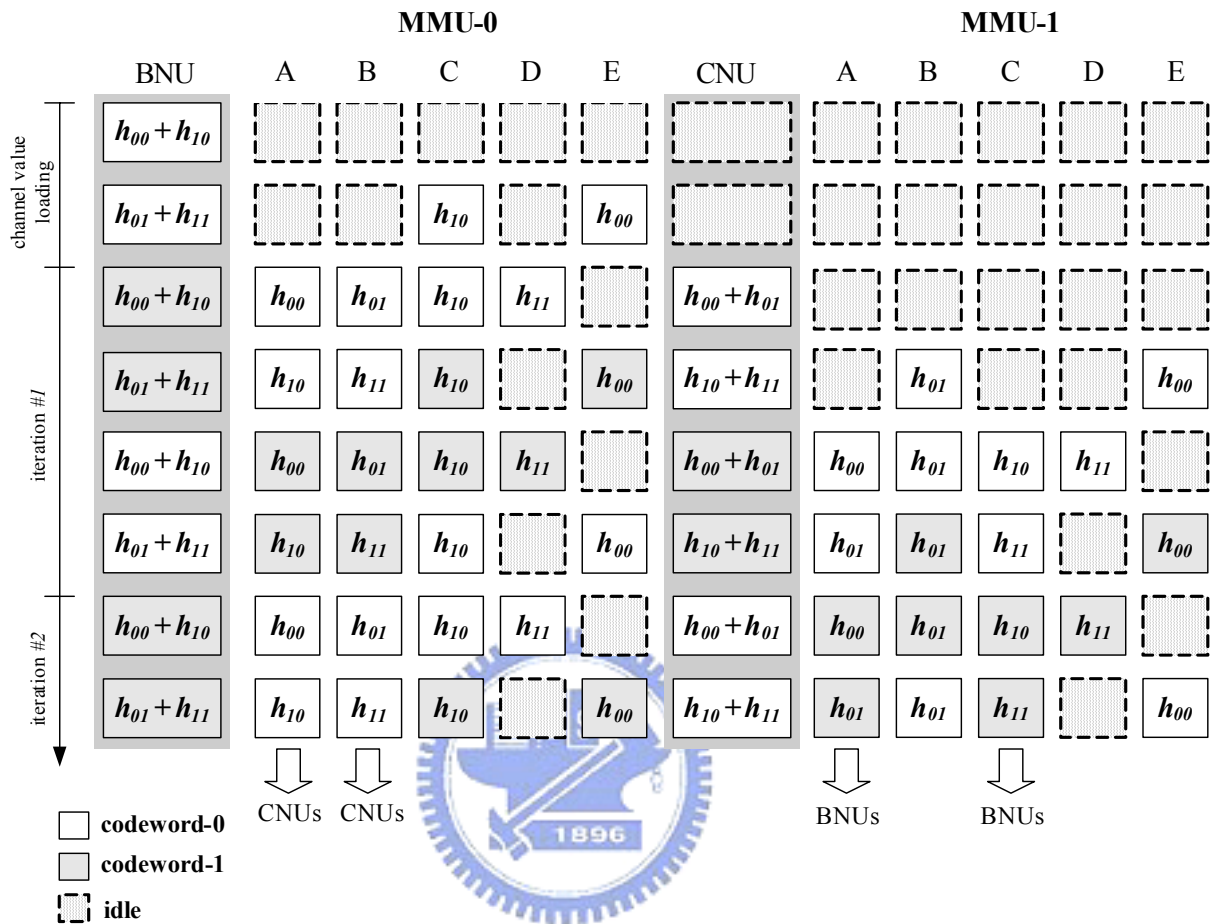


Fig. 4.26 Timing diagram of proposed LDPC code II decoder

4.3.5 Chip Implementation

A test chip has been fabricated in a 1.8V, 0.18 μ m 1P6M CMOS technology, and the die micrograph is shown in Fig. 4.27. The chip size is 25 mm² while the core occupies 21.23 mm². The total gate count is 1.15M including two MMUs while the chip core density is about 71.2%. By measurement, the decoder achieves 3.33Gb/s throughput with 8 decoding iterations under 1.62V power supply, and the power estimation is 644 mW.

A second test chip is implemented in a 1.2V, 0.13 μ m 1P8M CMOS technology, whose layout view is shown in Fig. 4.28. The chip size becomes 13.5 mm² where the core constitutes

10.24 mm². Moreover, the chip density grows to about 75.4% because of two extra metal layers. After static timing analysis (STA) and post-layout simulation, the maximum decoding speed has been improved to 5.92Gb/s with 8 decoding iterations under 1.02V supply and worst speed corner. The estimation also includes crosstalk analysis for signal wires that cause coupling noise. Table 4.4 gives the characteristic summary of two test chips.

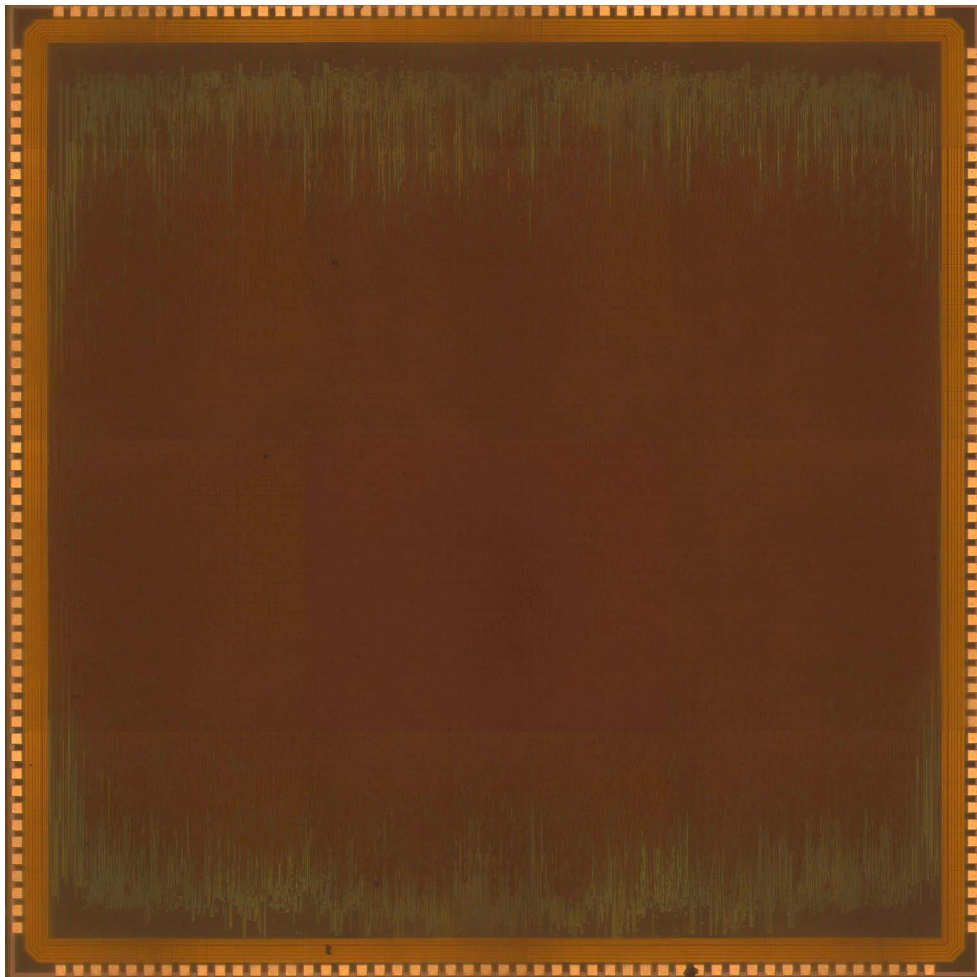


Fig. 4.27 Die micrograph of the 0.18 μ m LDPC code II decoder chip

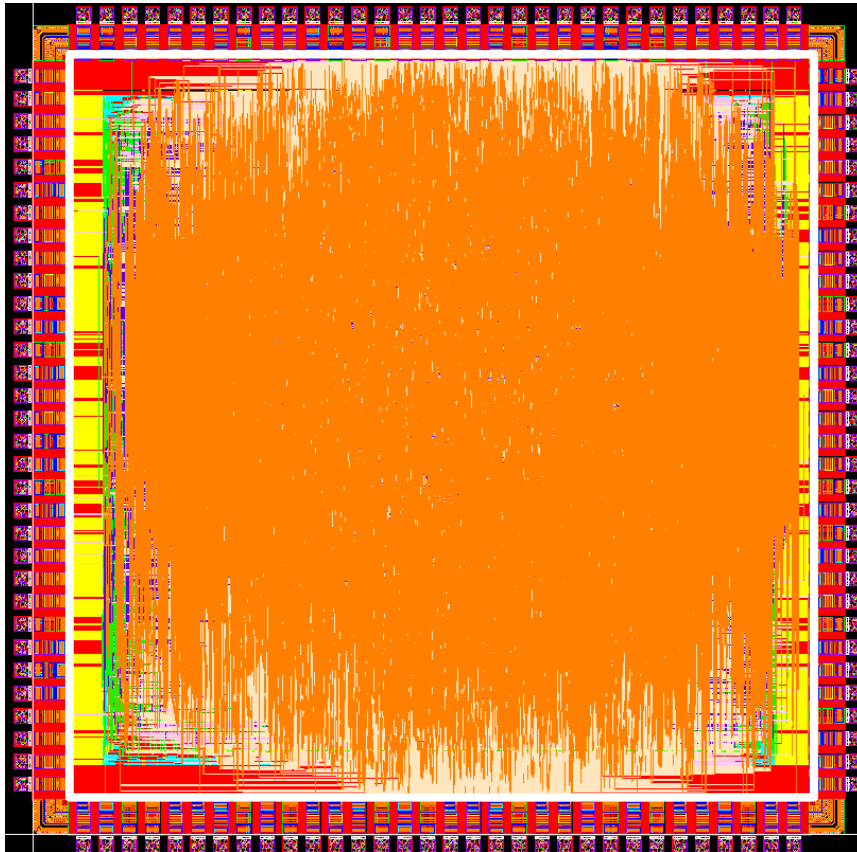


Fig. 4.28 Layout view of the 0.13μm LDPC code II decoder chip



Table 4.4 Summary of the LDPC Code II chip

Technology	0.18-μm CMOS 1P6M	0.13-μm CMOS 1P8M
Package	CQFP-208	N.A.
Supply voltage	1.8V core, 3.3 V I/O	1.2V core, 3.3V I/O
Chip size	5.0mm × 5.0mm	3.67mm × 3.67mm
Chip density	71.2%	75.4%
Gate count	1.15M	1.15M
Power dissipation	644mW @ 83MHz	299mW @ 145MHz
Maximum data rate	3.33Gb/s	5.8Gb/s

4.4 Summary and Comparison

The high speed LDPC code decoder designs are presented. The data rescheduling is employed to reduce the signal interconnections between the input buffer and the datapaths. The efficient functional unit designs make the decoder suitable for high speed applications. In addition, the message memories architecture permits parallel decoding of two codewords and diminishes the routing congestion issues. Consequently, the chip becomes smaller due to the increased chip density.

The comparisons of our proposed LDPC code decoders with state-of-the-arts are listed in Table 4.5. Except for [11], the decoders are implemented with non-structured LDPC codes to get a better performance and a general implementation solution.

Table 4.5 Comparison of LDPC chips

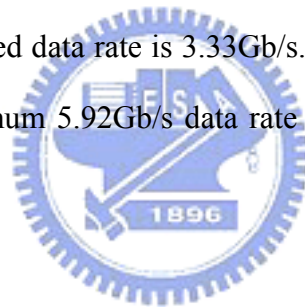
	Proposed I	Proposed II		[9]	[11]
Block length	600	1200		1024	2304
Code structure	irregular	irregular		irregular	structured
Code rate	3/4	3/5		1/2	2/3
Silicon proven	Yes	Yes	No	Yes	No
Technology	0.18- μm	0.18- μm	0.13- μm	0.16- μm	0.18- μm
Supply voltage	1.8V	1.8V	1.2V	1.5V	1.8V
Clock freq.	82.1MHz	83MHz	145MHz	64MHz	200MHz
Chip size	17.5mm ²	25mm ²	13.47mm ²	52.5mm ²	9.41mm ²
Gate count	472K	1.15M		1.75M	N.A.
Power dissipation	232mW	644mW	299mW	690mW	1,176mW
Data rate	480Mb/s	3.33Gb/s	5.8Gb/s	512Mb/s	128Mb/s
Decoding iteration	8	8		64	10

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this thesis, high-throughput and area-efficient LDPC code decoders are proposed for high-speed communication systems. A (600, 450) irregular LDPC code decoder is implemented in 0.18 μm technology and measured that it can achieve 480Mb/s data rate with 8 decoding iterations. Another (1200, 720) irregular LDPC code decoder is fabricated in 0.18 μm technology, whose measured data rate is 3.33Gb/s. Furthermore, the 0.13 μm (1200, 720) LDPC chip reaches the maximum 5.92Gb/s data rate with only 13.5 mm^2 area and 268mW power consumption.



5.2 Future Work

As mentioned in Section 3.1.1, DVB-S.2 system adopts LDPC codes with very large block lengths as the FEC kernel to get good error-correcting performance. However, the implementation complexity of LDPC code decoders goes larger as the block length grows. Besides, in DVB-S.2 system, there are a lot of different coding rate which are required for different application mode. In [27], a LDPC codec for DVB-S.2 is proposed, which can achieve 135Mb/s throughput rate. However, the chip size and the power consumption are both large. Our proposed designs may be applied to construct a low-power and area-efficient architecture for the DVB-S.2 LDPC code decoder. This will be an interesting topic for our future research works.

Bibliography

- [1] R. G. Gallager, “Low-Density Parity-Check Codes,” *IRE Trans. Inform. Theory*, vol. IT-8, pp. 21-28, Jan. 1962.
- [2] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA:MIT press, 1963.
- [3] R. M. Tanner, “A recursive approach to low complexity codes,” *IEEE Trans. Inform. Theory*, vol. IT-27, no. 5, pp. 399-431, Sep. 1981.
- [4] M. Sipser and D. A. Spielman, “Expander codes,” *IEEE Trans. Inform. Theory*, vol. 42, pp. 1710-1722, Nov. 1996.
- [5] D. J. C. Mackay, “Good error-correcting codes based on very sparse matrices,” *IEEE Trans. Inform. Theory*, vol. 45, pp. 399-431, Mar. 1999.
- [6] D. J. C. MacKay and R. M. Neal, “Near Shannon limit performance of low-density parity-check codes,” *Electron. Lett.*, vol. 32, pp.1645-1646, Aug. 1996.
- [7] S. Y. Chung, G. D. F. Jr., T. J. Richardson and R. Urbanke, “On the design of low-density parity-check codes within 0.0045dB of the Shannon limit,” *IEEE Commun. Lett.*, vol. 5, no. 2, pp. 58-60, Feb. 2001.
- [8] C. Berrou and A. Glavieux, “Near optimum error correcting coding and decoding: turbo –codes,” *IEEE Trans. Commun.*, vol. 44, no. 10, pp. 1261-1271, Oct. 1996.
- [9] A. J. Blanksby and C. J. Howland, “A 690-mW 1-Gb/s 1024-b, rate-1/2 low-density parity-check code decoder,” *IEEE Journal of Solid-State Circuits*, vol. 37, no. 3, pp. 404-412, Mar. 2002.
- [10] E. Yeo, P. Pakzad, B. Nikolic and V. Anantharam, “VLSI architectures for iterative decoders in magnetic recording channels,” *IEEE Trans. on Magnetics*, vol. 37, no. 2, pp. 748-755, March 2001.

- [11] M. M. Mansour and N. R. Shanbhag, "High-throughput LDPC decoders," *IEEE Trans. on VLSI Systems*, vol. 11, no. 6, pp. 976-996, Dec. 2003.
- [12] M. G. Luby, M. Mitzenmacher, M. A. Shokollahi and D. A. Spielman, "Improved low-density parity-check codes using irregular graphs," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 585-598, Feb. 2001.
- [13] T. J. Richardson and R. L. Urbanke, "Efficient encoding of Low-Density Parity-Check codes," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 638-656, Feb. 2001.
- [14] J. L. Fan, *Constrained Coding and Soft Iterative Decoding*, Kluwer Academic Publishers, 2001.
- [15] J. Hagenauer, E. Offer and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans. Inform. Theory*, vol. 42, no. 2, pp. 429-445, Mar. 1996.
- [16] N. Wiberg, "Codes and decoding on general graphs," Ph.D. dissertation, Univ. Linkoping, 1996.
- [17] X.-Y. Hu, E. Eleftheriou, D.-M. Arnold and A. Dholakia, "Efficient implementation of the sum-product algorithm for decoding LDPC codes," *IEEE Global Telecomm. Conf.*, vol. 2, pp. 25-29, Nov. 2001.
- [18] M. P. C. Fossorier, M. Mihaljevic and H. Imai, "Reduced complexity iterative decoding of low-density parity check codes based on belief propagation," *IEEE Trans. Comm.*, vol. 47, no. 5, May 1999.
- [19] ESTI, "Digital Video Broadcasting (DVB); Frame structure, channel coding and modulation for 11/12 GHz satellite services," *European Telecomm. Standard EN 300 421 V1.1.2*, Aug. 1997.
- [20] ESTI, "Digital Video Broadcasting (DVB); Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications," *European Telecomm. Standard EN 302 307 V1.1.1*, Mar. 2005.

- [21] A. Batra, J. Balakrishnan, G. R. A. J. R. Foerster and A. Dakbak, "Design of a multiband OFDM system for realistic UWB channel environments," *IEEE Trans. Microwave Theory Tech.*, vol. 52, no. 9, pp. 2123-2138, Sep. 2004.
- [22] L. Yang and G. Giannakis, "Ultra-wideband communications," *IEEE Signal Processing Mag.*, pp. 26-54, Nov. 2004.
- [23] A. Batra *et al.*, "Multi-band OFDM physical layer proposal for IEEE 802.15 task group 3a," submitted to IEEE P802.15 working group for WPANs, Sep. 2004.
- [24] D. Krishnaswamy and J. Vicente, "Scalable adaptive wireless networks for multimedia in the proactive enterprise," *Intel Technology Journal*, vol. 8, pp.291-302, 2004.
- [25] H. Y. Liu, C. C. Lin, Y. W. Lin, C. C. Chung, K. L. Lin, W. C. Chang, L. H. Chen, H.C. Chang and C. Y. Lee, "A 480Mb/s LDPC-COFDM-based UWB baseband transceiver," *IEEE Int. Solid-State Circuit Conf. Dig Tech. Papers*, pp. 444-445,609, 2005.
- [26] X. Y. Hu, E. Eleftheriou and D. M. Arnoldx, "Regular and irregular progressive edge-growth tanner graphs," *IEEE Trans. Inform. Theory*, vol. 51, no. 1, pp. 386-398, Jan. 2005.
- [27] P. Urard, E. Yeo, L. Paumier, P. Georgelin, T. Michel, V. Lebars, E. Lantreibecq and B. Gupta, "A 135Mb/s DVB-S2 compliant codec based on 64800b LDPC and BCH codes," *IEEE Int. Solid-State Circuit Conf. Dig Tech. Papers*, pp. 446-447,609, 2005.

作者簡歷

姓名：林凱立

出生地：台灣省台中市

出生日期：1982. 1. 4

學歷：1987. 9 ~ 1993. 6 台中市立中正國民小學

1993. 9 ~ 1996. 6 台中市立居仁國民中學

1996. 9 ~ 1999. 6 國立台中第一高級中學

1999. 9 ~ 2003. 6 國立清華大學 電機工程學系 學士

2003. 9 ~ 2005. 7 國立交通大學 電子研究所 系統組 碩士

得獎事績

九十二學年度 第二學期電子研究所書卷獎

九十二學年度 2004 全國系統晶片設計比賽光電通訊類 SOC 組特優獎

九十四學年度 斐陶斐榮譽學會新榮譽會員

發 表 論 文

- Hsuan-Yu Liu, Chien-Ching Lin, Yu-Wei Lin, Ching-Che Chung, Kai-Lin Lin, Wei-Che Chang, Lin-Hong Chen, Hsie-Chia Chang, Chen-Yi Lee, “A **480Mb/s LDPC-COFDM-based UWB Baseband Transceiver in 0.18 μ m CMOS Process**,” in *IEEE ISSCC*, Feb. 2005.
- Chien-Ching Lin, Kai-Li Lin, Hsie-Chia Chang and Chen-Yi Lee, “A **3.33Gb/s (1200, 720) Low-Density Parity Check Code Decoder**,” in *IEEE ESSCIRC*, Sep. 2005.

