國立交通大學

電子工程學系　電子研究所碩士班

碩 士 論 文

高編碼率之 CP-PEG LDPC 解碼器設計與實做

Design and Implementation of High Code-Rate LDPC Decoder based on CP-PEG Code Construction

學生：林高守

指導教授：張錫嘉 方偉騏 教授

中華民國九十八年七月

高編碼率之 CP-PEG LDPC 解碼器設計與實做

Design and Implementation of High Code-Rate LDPC Decoder based on CP-PEG Code Construction

研 究 生：林高守　　Student：Kao-Shou Lin

指導教授：張錫嘉教授　　Advisor：Hsie-Chia Chang

方偉騏教授　　Wai-Chi Fang

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

July 2009

Hsinchu, Taiwan, Republic of China

中華民國九十八年七月

高編碼率之 CP-PEG LDPC 解碼器設計與實做

學生：林高守　　指導教授：張錫嘉 教授

方偉騏 教授

國立交通大學

電子工程學系 電子研究所碩士班

摘　要

本論文提出了高編碼率之 CP-PEG 低密度同位元碼解碼器。我們使用 CP-PEG 演算法建造了一個(2048,1920) 非規則低密度同位元碼，錯誤更正能力勝過其他 PEG-BASED 的演算法所建造的碼。 然而，高字碼 15/16 將會導致大的 Check node degree，也會成為硬體實做上的困難點。我們使用了 VSS 排程降低解碼圈數，並提出單級管線架構減少訊息的儲存量，同時我們又更進一步最佳化 CNU 減少所需的暫存器。比起傳統架構，總共 73%的訊息可以省去不用儲存。此解碼器在 90nm 製程下，當供應電壓為 1.4V，最高能達到 11.5Gbps 的解碼速度，晶片的面積是 3.78mm^2。當供應電壓為 0.8V, 能源效率為 0.033 nJ/bit 解碼速度為 5.77Gbps。根據實驗結果，此 CP-PEG 解碼器的解碼速度達到 IEEE 802.15.3c (1440,1344)碼的要求，並且 CP-PEG 解碼器擁有與(1440,1344)類似的編碼率。所以我們所提出的方法可以有效的用於設計出高編碼率低密度同位元檢查碼的實做上。

Design and Implementation of High Code-Rate LDPC Decoder based on CP-PEG Code Construction

Student：Kao-Shou Lin Advisor：Dr. Hsie-Chia Chang

Dr. Wai-Chi Fang

Department of Electronics Engineering

Institute of Electronics

National Chiao Tung University

Abstract

In this thesis, a LDPC decoder chip based on CP-PEG code construction is presented. The (2048, 1920) irregular LDPC code generated by CP-PEG algorithm has better performance than other PEG-based codes; however, the large check node degrees introduced by high code-rate 15/16 become the implementation bottleneck. To design such high code-rate LDPC decoder, our approach features *variable-node-centric* sequential scheduling to reduce iteration number, single pipelined decoder architecture to lessen the message storage memory size, as well as optimized check node unit to further compress the register number. Overall 73% message storage memory is saved compared to traditional architecture. Fabricated in 90nm CMOS technology, a test decoder chip could achieve maximum 11.5 Gbps throughput under 1.4V supply voltage with core area of 3.78 mm^2. While the throughput meets IEEE 802.15.3c (1440, 1344) LDPC code requirement. In addition, CP-PEG (2048, 1920) LDPC code own the similar code rate as the (1440, 1344) code. Thus our proposed methodology is proven to be effective in high code rate decoder design and implementation.

誌　謝

感謝交通大學，讓我在短短兩年的碩士生涯加上四年的大學生涯收穫許多，在許多師長的提攜以及同學朋友的鼓勵幫助下，讓我的研究路程順利且平穩，首先我要感謝的是我的共同指導教授方偉騏老師，謝謝他給我最大的空間做自己有興趣的研究，並且對我相當的包容與體諒。接著我要感謝我另一個指導教授張錫嘉老師，親切的老師不僅能和我們討論到研究內容，也能和我們聊一些八卦有趣的事，老師不只在研究上，或在待人處事上，都是讓我覺得自己遠遠不及並需要再學習的。

再來要感謝 Ocean group 的全部成員，很高興的能成為 Ocean group 的一分子。尤其要感謝帶我的陳志龍學長，無私的指導，讓我在專業領域中獲益良多。而在遇到瓶頸問題時，學長的耐心指導，也幫助了我很順利完成各項研究進度。當然也感謝 OASIS 實驗室的每位學長同學和學弟妹們，讓我度過了一個充實又開心的研究生涯。

另外要感謝碩士班生涯陪伴我的大學同學，時穎，政寬，品為，俊豪，狗狗，圈片，標哥，翰平，聖凱，威良，小獸，謝謝你們的陪伴與嘴砲，一起 Play Ball 等，讓我的研究生涯不至於太苦悶。

我要感謝一直在背後支持我的女友郁涵，當我遇到挫折或壓力大的時候，你總是能聽我的苦水，鼓勵我，給我力量。最後，僅將此論文獻給疼愛我的父母與哥哥，謝謝你們的愛與支持讓我能夠無憂無慮的學習。

To my family and Yu-Han for their love and support,

and my country Taiwan

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Low-density parity-check (LDPC) code is a famous error control code with near Shannon limit performance [1] and can be described by its parity-check matrix H. The rows and columns of H are mapped to check nodes and variable nodes of a bipartite graph, on which the belief-propagation (BP) algorithm exchange messages between nodes iteratively to decode LDPC codes [2]. The message exchanging order between nodes is called scheduling, which will influence the convergence speed of the decoding algorithm. In standard BP algorithm, simultaneous update of all check node messages or variable node messages is named as *flooding scheduling*. Alternatively, the layered BP algorithm [3] [4] preforming message update along different check node groups is a method of *check-node-centric sequential scheduling* (CSS). Researches have revealed that CSS could reduce maximum iteration to approximate half of the standard BP with similar performance.

Recently, LDPC codes adopted in high-throughput systems have high code-rate property to increase channel efficiency. However, the introduced large check node degree dc will cause implementation difficulties. For example, the largest check node degree of (2048, 1723) LDPC code adopted in IEEE 802.3an [5] equals 32, leading to routing difficulty and low chip density. Even though the CSS could reduce the iteration number, the throughput is still degraded due to long critical path of check node unit (CNU). The situation will become worse for the (1440, 1344) LDPC code of IEEE 802.15.3c [6] with $dc = 45$.

In this thesis, the proposed decoder aims at providing a high-throughput and hardware-

efficient solution to the high code-rate LDPC with large check node degrees. In order to reduce iteration number, the decoding scheduling is based on the *variable-node-centric sequential scheduling* (VSS; also known as shuffled decoding [7]), where the messages are updated along different variable node groups. Since the inputs of CNU operation are also divided into several subgroups, the complexity and critical path delay of CNU are reduced. Furthermore, single pipelined approach and modified CNU are proposed to minimize the message storage memory. Using a (2048,1920) LDPC code constructed by circulant permutation progressive edge-growth (CP-PEG) algorithm [8] as a design example, the overall decoder chip implemented in 90nm technology will show its advantages in terms of throughput, energy efficiency, and hardware efficiency.

1.2 Thesis Organization

The rest of this thesis is organized as follows. Chapter II introduces the code structure and the decoding algorithm with VSS. In Chapter III, we propose a modified scheduling algorithm and an improved decoder architecture. Performance simulation and implementation result are shown in Chapter IV. The conclusion is given in Chapter V. In order to investigate the error floor of the CP-PEG code, we build a AWGN core to speed up the simulation process, and the AWGN core is introduced in Appendix A.

Chapter 2

Decoding Algorithm and Code Structure

2.1 LDPC decoding algorithm

2.1.1 Standard 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 standard BP is carried out as followed.

1. **Initialization:** Set $i = 1$,maximum number of iterations to I_{Max}. For each m, n, set $z_{mn}^{(0)} = P_n$,

2. **Iterative Decoding:**

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

$$\tau_{mn}^{(i)} = \prod_{n' \in N(m) \backslash n} \tanh(\frac{z_{mn'}^{(i-1)}}{2}) \tag{2.1}$$

$$\varepsilon_{mn}^{(i)} = \log \frac{1 + \tau_{mn}^{(i)}}{1 - \tau_{mn}^{(i)}} \tag{2.2}$$

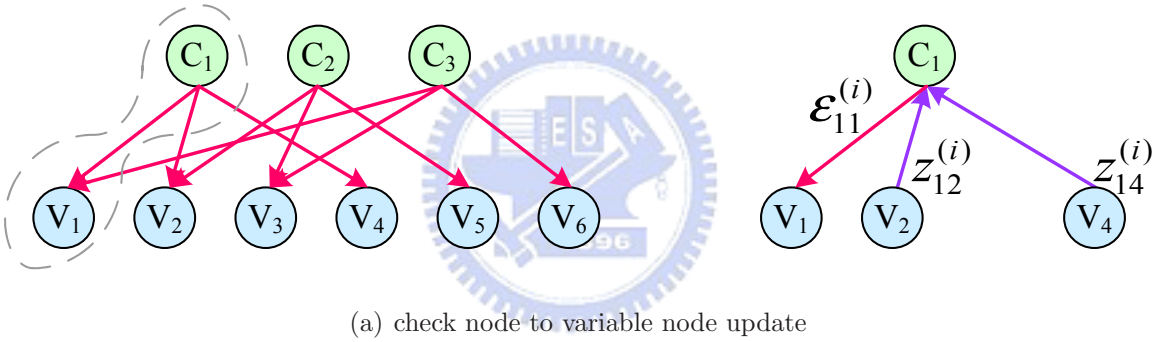
(b) variable node to check node update step, for $1 \leq n \leq N$ and each $m \in M(n)$, process

$$z_{mn}^{(i)} = P_n + \sum_{m' \in M(n) \backslash m} \varepsilon_{m'n}^{(i-1)} \tag{2.3}$$

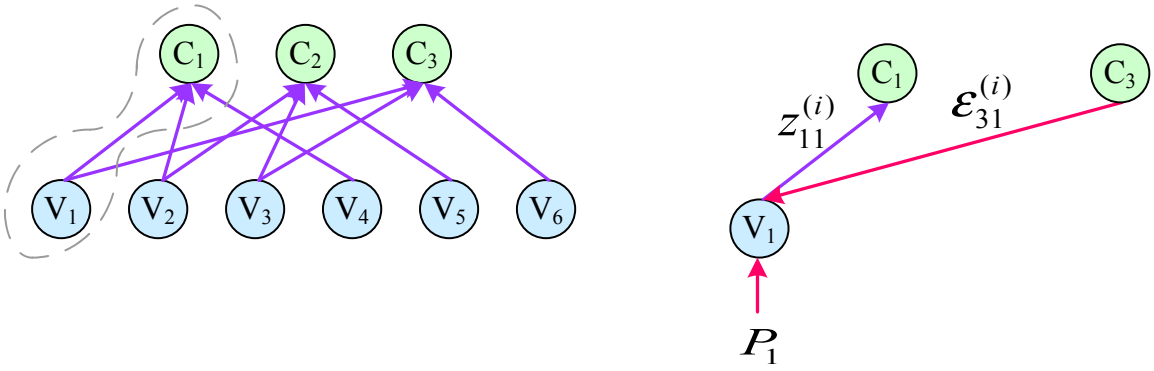
$$z_{n}^{(i)} = P_n + \sum_{m \in M(n)} \varepsilon_{mn}^{(i-1)} \tag{2.4}$$

3. **Hard Decision:** Let X_n be the n-th bit of decoded codeword. If $z_n^{(i)} \geq 0$, $X_n = 0$, else if $z_n^{(i)} < 0$, $X_n = 1$. If $H(x^{(i)})^t = 0$ or I_{Max} is reached, stop and output the code word. Otherwise,set $i = i + 1$ and go to **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 are shown in Fig. 2.1(a) and 2.1(b).



(a) check node to variable node update



(b) variable nod to check node update

Figure 2.1: Illustration of standard BP.

2.1.2 Variable-node-centric Sequential Scheduling

In VSS approach, the initialization, stopping criterion test, and output steps remain the same as the standard BP algorithm. The only difference between two algorithms lies in 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 one group of variable nodes and check nodes which are connected the group of variable nodes at a time. In addition, each group of messages is updated in order. Furthermore, it count one iteration when all groups have been updated. 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 our VSS approach with normalized factor $\beta = 0.75$. The updating procedure of NMS algorithm with VSS approach is carried out as follows.

1. **Initialization:** $z_{mn}^{(0)} = P_n$

2. **Iterative Decoding:** For $0 \leq g \leq G-1$, perform the following two steps.

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

$$\varepsilon_{mn}^{(i)} \approx \prod_{\substack{n' \in N(m) \backslash n \\ n' \leq g \cdot N_G - 1}} sign(z_{mn'}^{(i)}) \times \prod_{\substack{n' \in N(m) \backslash n \\ n' \geq g \cdot N_G}} sign(z_{mn'}^{(i-1)})$$
$$\times \min \left\{ \min_{\substack{n' \in N(m) \backslash n \\ n' \leq g \cdot N_G - 1}} \left\{ \left| z_{mn'}^{(i)} \right| \right\}, \min_{\substack{n' \in N(m) \backslash n \\ n' \geq g \cdot N_G}} \left\{ \left| z_{mn'}^{(i-1)} \right| \right\} \right\} \times \beta \tag{2.5}$$

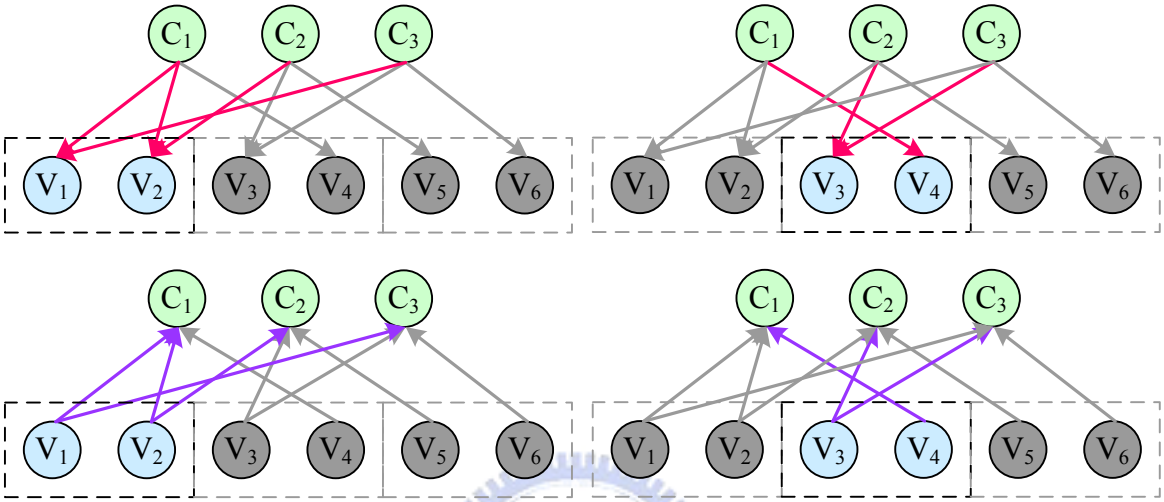
   (b) variable node to check node update step, for $g \cdot N_G \leq n \leq (g+1) \cdot N_G - 1$, process

$$z_{mn}^{(i)} = P_n + \sum_{m' \in M(n) \backslash m} \varepsilon_{m'n}^{(i-1)} \tag{2.6}$$

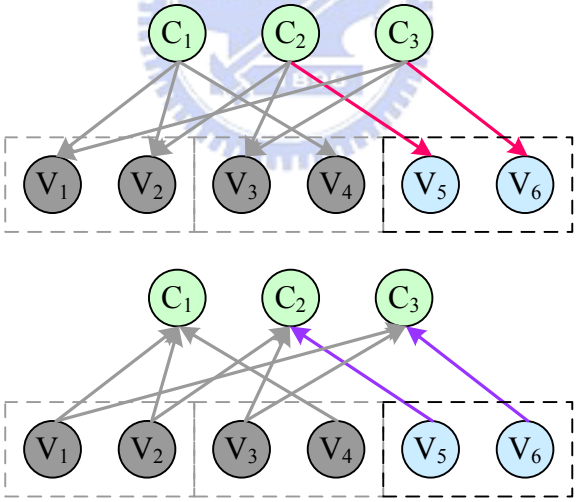
$$z_n^{(i)} = P_n + \sum_{m \in M(n)} \varepsilon_{mn}^{(i-1)} \tag{2.7}$$

3. **Hard Decision:** Let X_n be the n-th bit of decoded codeword. If $z_n^{(i)} \geq 0$, $X_n = 0$, else if $z_n^{(i)} < 0$, $X_n = 1$.

The decoding processes for one iteration of VSS is illustrated in Fig. 2.1.2 with $G = 3$ as example. The arrows with red color represent check node to variable node messages to be updated. The arrows with purple color represent variable node to check node messages to be updated. On the other hand, gray arrows represent that messages are not updated.



(a) Update first group (b) Update second group



(c) Update third group

Figure 2.2: Illustration of VSS

2.2 CP-PEG LDPC Code

2.2.1 CP-PEG LDPC Code Construction

The (2048, 1920) irregular LDPC code, rate-15/16, used in this paper was constructed by CP-PEG algorithm and shown in Fig. 2.3. The constructed parity-check matrix H consists of $p \times p$ circulant permutation (CP) and all-zero matrices. A CP matrix is a cyclic square matrix with constant row and column weight of one. The number of each CP matrix indicates the cyclic shift amount and -1 means all zero matrix. By setting $p = 32$, there are $4 \times p$ check nodes and $64 \times p$ variable nodes in bipartite graph, where each check node has uniform degree 46, and $16 \times p$, $24 \times p$, $24 \times p$ variable nodes have degrees of 4, 3, 2 respectively. The performance of this code was proven to have better performance than other PEG-based structure LDPC codes [8]; nevertheless, the high check node degree required suitable decoder architecture to overcome implementation difficulties.

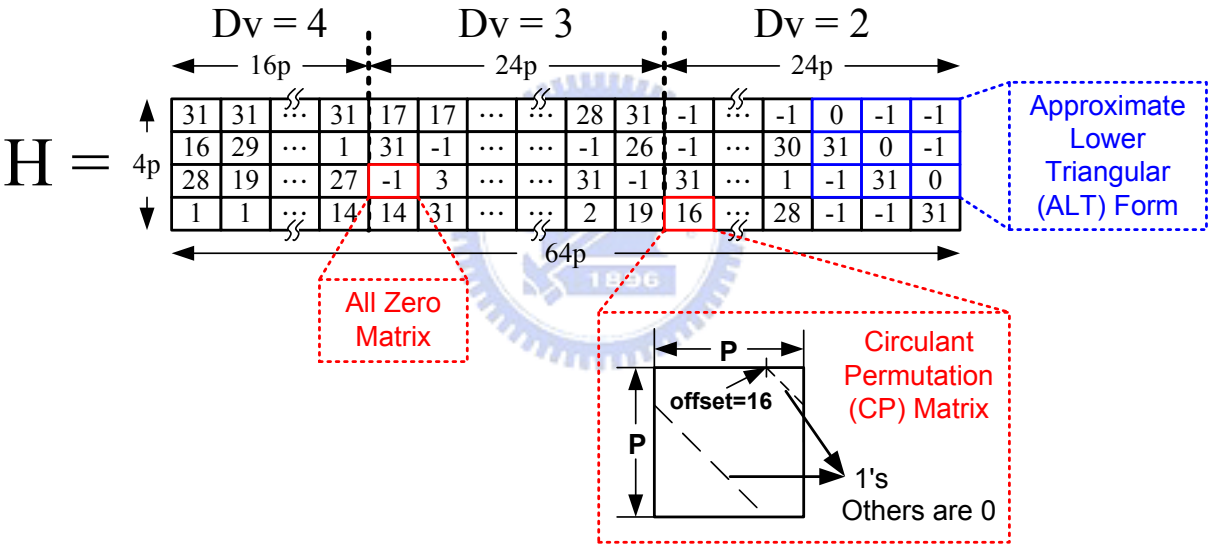


Figure 2.3: Parity-check matrix of (2048, 1920) LDPC code.

2.2.2 Parity Check Matrix Permutation and Division

We can observe equation (2.6) and (2.7), only one group of variable node units (VNUs) are required to update the messages from variables node to check nodes.

As shown in Fig. 2.4, the codeword is divided into 4 groups (i.e. $G = 4$) for VSS, therefore the parity-check matrix H is divided and permuted into 4 submatrices (H1 to

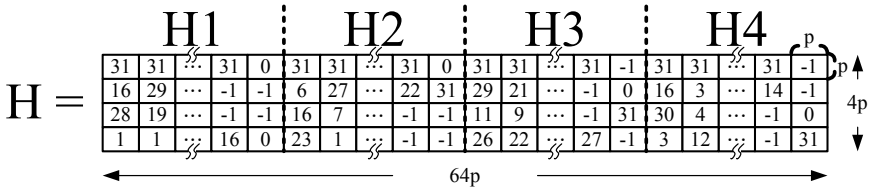
H4).

In order to fully reuse the same VNUs when updating different groups, each submatrix consists of equal number of variable nodes with the same degree to reduce the hardware cost of VNU and unnecessary control. Moreover, the submatrices with the same shift amounts (shaded blue CP matrices) are arranged in the same position which makes the same connections between CNUs and VNUs when updating different groups. By utilizing this method, number of mux could be reduced, and the routing and control could be further simplified. The reduced number of mux is shown in Table 2.1.

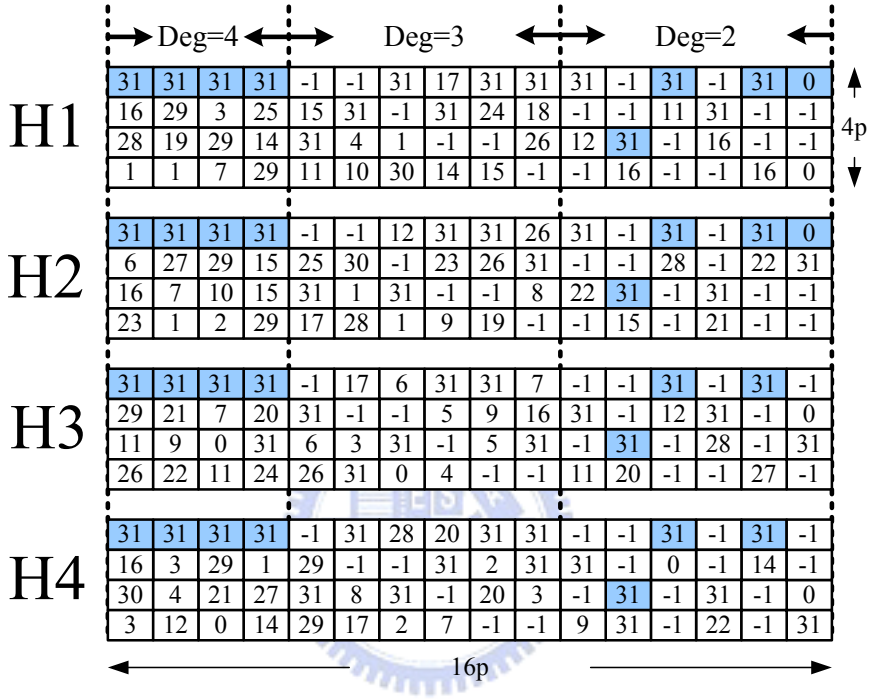
However, if the other parity check matrix does not contain many submatrices with same shift amount, there is still a technique to permute the submatrix to reduce the number of mux. We could permute each submatrix in the same position to make the submatrix with same shift amount. Take the 5_{th} column of each submatrix $H1, H2, H3, H4$ as example which is shown in Fig. 2.4(c). The offset of submatrix in 5_{th} column and 3_{rd} row of $H3$ is 6, and we could permute the column to let the offset become 31. In consequence, the submatrix in 5_{th} column and 3_{rd} row of $H1, H2, H3, H4$ become the same. Thus the mux could be reduced. In addition, the permutation will not cause performance degradation because the structure (local girths or global girths) of the tanner graph still remains the same.

Table 2.1: Mux reduction statistics

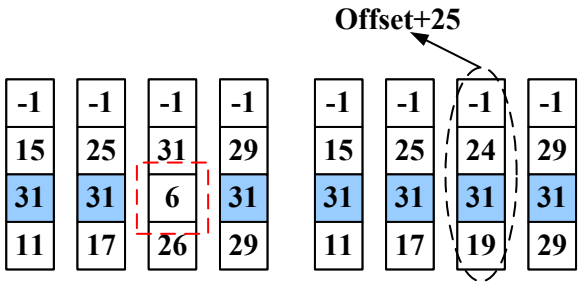
Direction	VNU to CNU	CNU to VNU
Original number of MUX2	128	N/A
Reduced number of MUX2	32	N/A
Original number of MUX4	1408	1472
Reduced number of MUX4	224	224



(a) overview of permuted matrix



(b) Permuted and divided into four groups



(c) Further MUX reduction technique

Figure 2.4: Permuted parity-check matrix of (2048, 1920) LDPC code.

Chapter 3

Proposed Decoder Architecture

In this chapter, a complete decoder architecture will be presented, including datapath, scheduling, and VLSI structure of CNU and optimized CNU.

3.1 Single Pipelined Architecture

The entire decoder depicted in Fig. 3.1(a) is composed of fully-parallel CNUs and partial-parallel VNUs, where the VNU2, VNU3, and VNU4 will handle variable node operations with degree 2, 3, and 4 respectively. Let $\alpha_g^{(i)}$ denotes the sorted messages sent from variable nodes in the g-th group to one specific check node at i-th iteration, which is:

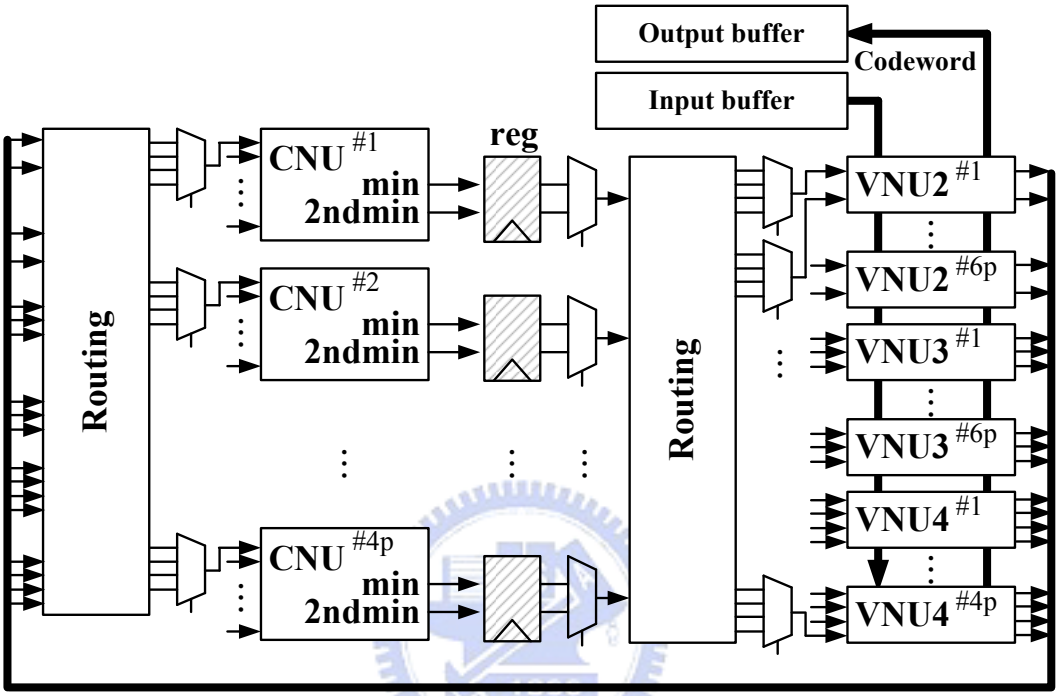
$$\alpha_g^{(i)} = \min_{\substack{n' \in N(m) \setminus n \\ g \cdot N_G \le n' \le (g+1) \cdot N_G - 1}} \left\{ \left| z_{mn'}^{(i)} \right| \right\} \tag{3.1}$$

Then the magnitude part of check node to variable node message in (2.5) could be computed by the following equation:

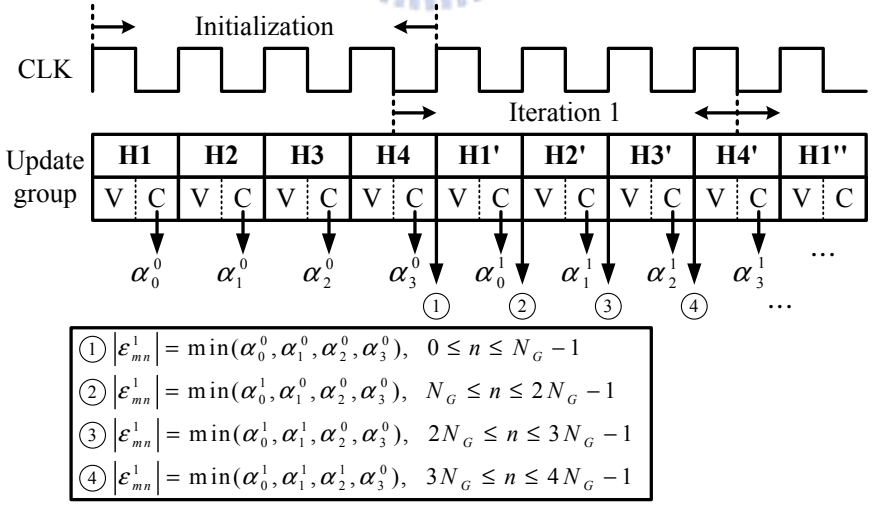
$$\left| \varepsilon_{mn}^{(i)} \right| = \min \left\{ \left\{ \alpha_j^{(i)} \right\}_{j<g}, \alpha_g^{(i)}, \left\{ \alpha_k^{(i-1)} \right\}_{k>g} \right\} \tag{3.2}$$

Fig. 3.1(b) demonstrates the timing diagram of proposed decoder. There are G initialization cycles required to calculate α_g^0 for $0 \le g \le G-1$. Since only one subgroup of the message $z_{mn}^{(i)}$ is updated in g-th cycle of one iteration, the main operation of CNU could be simplified to calculate $\alpha_g^{(i)}$ (local sorting) in each cycle and then perform global sorting like equation (3.2).

From the propose single pipelined architecture, only messages $\alpha_g^{(i)}$ and $\varepsilon_{mn}^{(i)}$ are stored. The sorted results could be represented by min value, second min value, and the index of min value in NMS algorithm. Therefore, the proposed decoder only latches two values, one index, and sign part of messages in each subgroup, while the variable node to check node message $z_{mn}^{(i)}$ is on-the-fly calculated. The single pipelined architecture is feasible because the CNU could be updated immediately after VNU's operations in VSS approach.



(a) Single pipelined LDPC decoder architecture



(b) variable-node-centric sequential scheduling

Figure 3.1: Proposed architecture and scheduling.

3.2 Check Node Unit

This section presents detail CNU architecture based on VSS scheduling. The CNU architectures are further optimized to reduce storage requirement and the number of sorter. Different CNU architectures will also influence different convergence speed which will be discussed in the following subsections.

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 CP-PEG LDPC codes with $dc = 46$, The VSS approach with $G = 4$ could divide 46 inputs of the CNU into four part. Because 46 could not be divisible by 4, the number of messages need to be computed in different groups will be 11 or 12. This could be handled by little extra circuit which determines the number of messages which should be computed by CNU when updating different groups.

The detail will be presented including accumulative sorter, Accumulative Sign Operation Unit, Reduced Memory Accumulative Sorter 1, Reduced Memory Accumulative Sorter 2.

3.2.1 Accumulative Sorter

Fig. 3.2 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 min and second min values in each subgroups, and global min and second min values of a check node will be found by a global sorter.

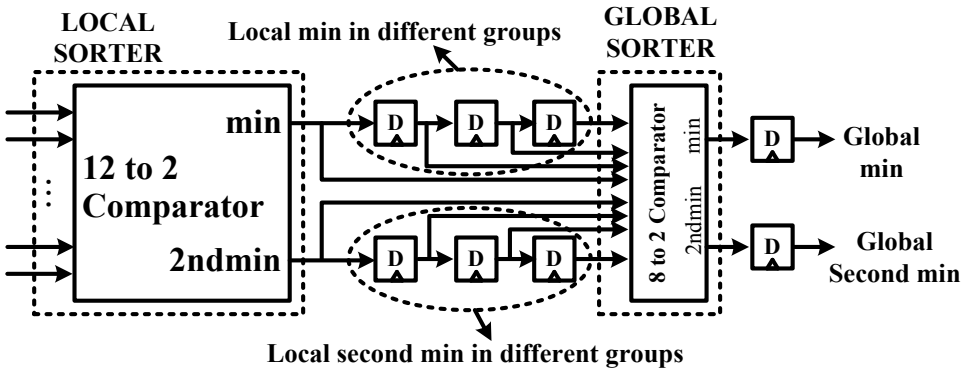


Figure 3.2: Accumulative sorter

3.2.2 Accumulative Sign Operation Unit

Fig. 3.3 shows the sign part of CNU. Similarly, the sign operation can be computed in an accumulative way like the accumulative sorter. The XOR12 is used to find the local sign in each subgroups, and global sign could be found by XOR4. Since only $d_c/4$ messages are computed by CNU in a cycle, $d_c/4$ messages's signs are queued in a cycle. After the global sign is available, the $d_c/4$ messages's signs are computed with global sign by xor and sent to VNUs.

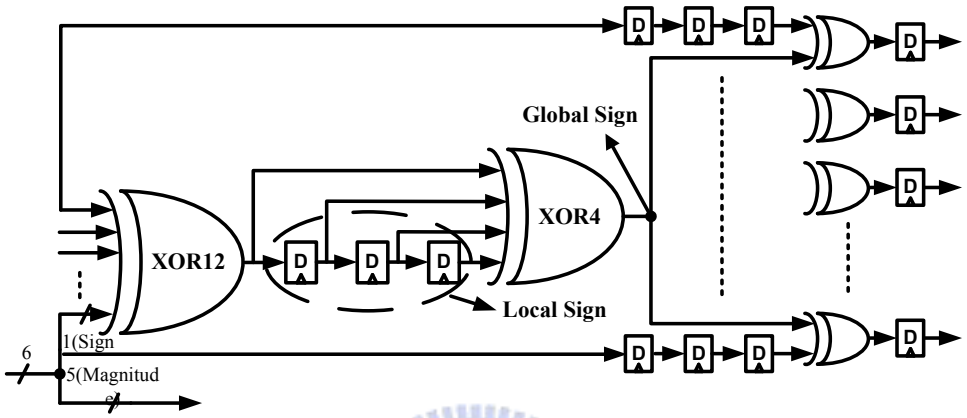


Figure 3.3: Sign operation unit

3.2.3 Reduced Memory Accumulative Sorter 1

In order to further reduce the storage overhead of each subgroup, we propose a reduced memory accumulative sorter 1 as shown in Fig. 3.4. The basic idea is to simplify the local min and local second min of $G-1$ subgroups into only one group. Some extra control circuits are needed to open or close the feedback loop in Fig. 3.4. This sorter architecture is beneficial since the complexity reduction of storage registers and global sorters is higher than the overhead of control circuits. The algorithm of Reduced Memory Accumulative Sorter 1 is listed below. The open loop operation can be implemented by selecting the value of feedback loop to maximum value(ex:5'b11111) which make the value invalid in comparison. The performance and hardware complexity will be discussed in following subsection.

Algorithm: Reduced Memory Accumulative Sorter 1

1: **Initialization**

2: inner feed back loop closed

3: outer feed back loop open

4: find global min and global second min

5: **Decoding**

6: inner feed back loop open

7: **if**(global min in current Group **or** current Group == 4)

8: global min outer feed back loop open

9: **else**

10: global min outer feed back loop closed

11: **if** (global second min in current Group **or** current Group== 4)

12: global second min outer feed back loop open

13: **else**

14: global second min outer feed back loop closed

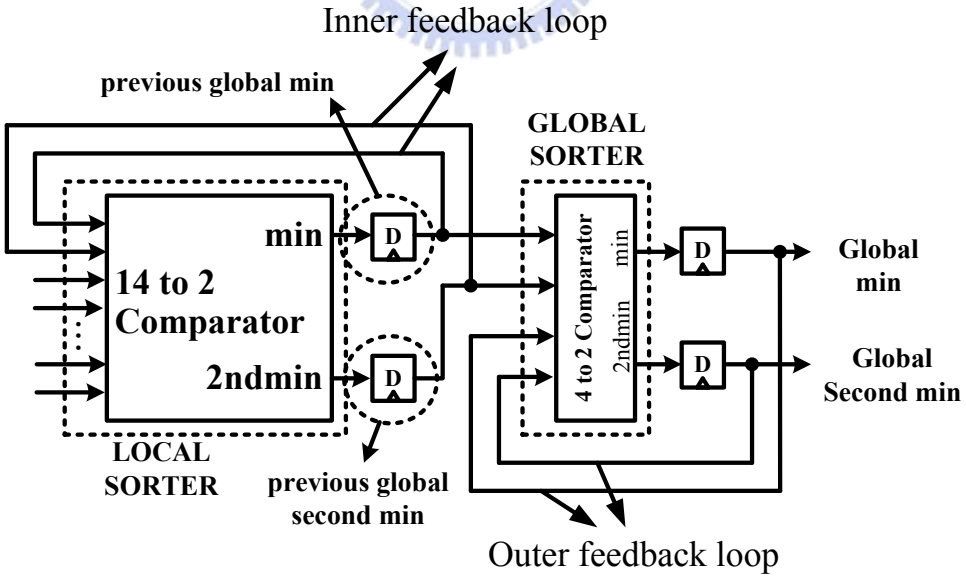


Figure 3.4: Reduced storeage accumulative sorter

3.2.4 Reduced Memory Accumulative Sorter 2

For the same purpose, we propose a reduced storage accumulative sorter 2 as shown in Fig. 3.5. The basic idea is to store only local min values in each group, and use 4-to-2 sorter to find global min and global second min values. Clearly, global min value will be correctly sorted in comparison with original accumulative sorter; nevertheless, the global second min could be sorted incorrectly if the global second min and global min are located in the same group[1].

Compared to reduced memory accumulative sorter 1, this sorter architecture is more beneficial in hardware complexity. The first reason is that the input number of local sorter is fewer and only min value need to be found by local sorter. The second reason is that the second min index does not need to be stored. The third reason is that control circuit to open or close the feedback loop is not required .

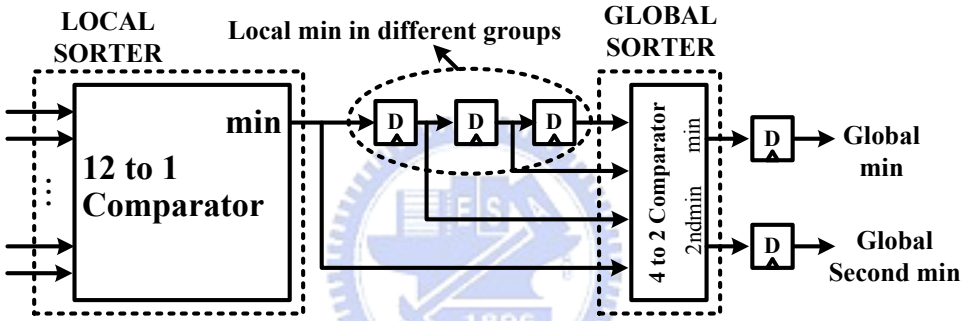


Figure 3.5: Reduced storeage accumulative sorter 2

3.3 Variable Node Unit

Fig. 3.6 shows the VNU4, where SM to TC represents sign-magnitude to two's-complement conversion, and TC to SM represent two's-complement to sign-magnitude conversion. Since the parity check matrix is permuted carefully, the $16 \times p$ VNU's contain $4 \times p$ VNU4s, $6 \times p$ VNU3s and $6 \times p$ VNU2s, and there is no any redundant VNUs appearing in variable node to check node update. In Fig. 3.2, the four registers correspond to

[1]Intuitively, the the expected value of second min of this architecture is larger than the accumulative sorter. Different scaling factor or other mechanic could be applied to make the performance close to the original accumulative sorter. However, it require further investigation.

four different channel values in the four different groups. For example, when the variable node to check node operation is performed in H3($g = 3$), the channel values in H3 will be shifted to be computed. Similarly, the decoded bits are stored in the four register and shifted to the corresponding group decoding bits.

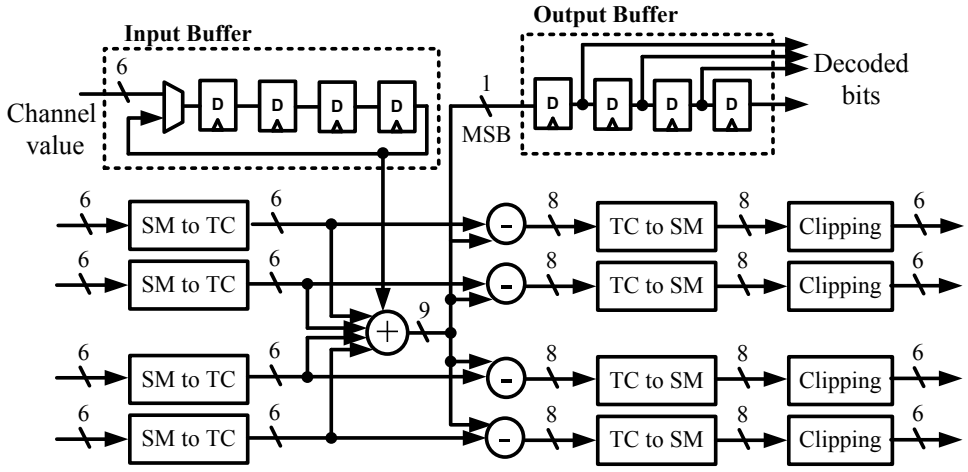


Figure 3.6: VNU architcture

3.4 Comparison and Discussion

3.4.1 Evaluation of Different Sorter Architecture

Three sorter architecture has been proposed. The convergence speed and hardware complexity of different architecture will be discussed in this sub section. The whole decoder with different sorter architecture has been implemented by standard-cell design flow and synthesized by utilizing 90-nm 1P9M CMOS technology. AS decoder denote whole decoder which CNU is implemented by original accumulative sorter. RMAS1 decoder denote whole decoder which CNU is implemented by Reduced Memory Accumulative Sorter 1. RMAS2 decoder denote whole decoder which CNU is implemented by Reduced Memory Accumulative Sorter 2. The sign part of CNU of three decoders are implemented in same architecture(accumulative sign operation unit).

The synthesis result (whole decoder) is shown in table 3.1; each decoder is synthesize by Synopsys design compiler with same tickle file. The convergence speed of different architecture is shown in Fig. 3.7. We found that the performance of RMAS1 decoder is

almost equivalent to AS decoder. The convergence speed of RMAS2 decoder is slightly lower than RMAS1 decoder and AS decoder. Of the three architecture, RMAS2 decoder provides lowest hardware cost;however, the performance of RMAS2 degrade. RMAS 1 decoder provides lower hardware cost and own almost equivalent performance in comparison with AS decoder.

Table 3.1: Comparison between different sorter architectures

Architecture[1]	AS Decoder	RMAS1 Decoder	RMAS2 Decoder
Gate Count[2]	727k	661k	570k

[1] Synthesis result

[2] The clock period is set to 9ns during synthesis

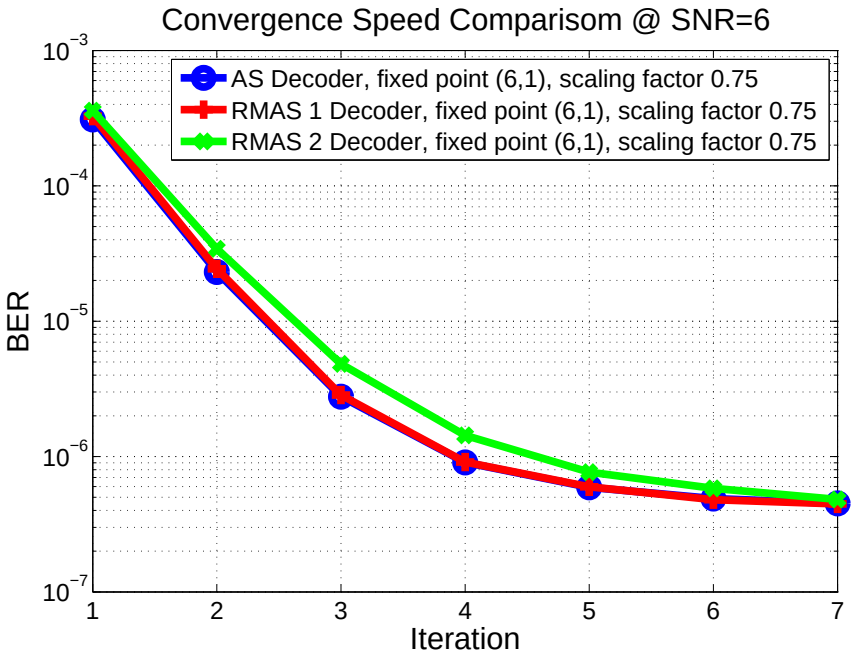
3.4.2 Comparison With Conventional Architecture

In single pipelined architecture, more subgroup number G will result in fewer inputs of local sorter but more inputs of global sorter; the storage for min, second min, and index values of each subgroup will increase. The mux (G inputs mux) number will be around $2 \times N/G$ between CNU and VNU. In addition, the critical path will be longer when G is larger. However, single pipelined architecture is suitable for high code rate design with proper chosen G.

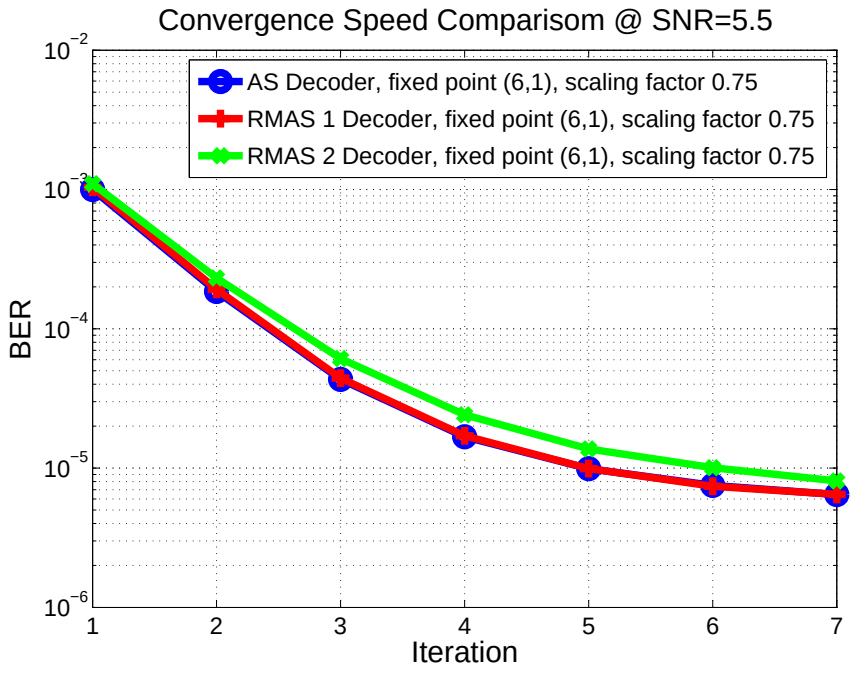
In traditional two-stage pipelined architecture, both $z_{mn}^{(i)}$ and $\varepsilon_{mn}^{(i)}$ messages are kept in registers or memory. Assume the bit-width w of messages is 6 and variable node degree is dv, then the required memory size (or registers) is as follows:

$$\begin{aligned} & MEM_{VNU} + MEM_{CNU} \\ = \; & (\sum dv) \cdot w + \\ & (N-K) \cdot (Min + 2ndMin + Index + Sign) \\ = \; & (\sum dv) \cdot w + (N-K) \cdot (2 \cdot (w-1) + log_2\lceil dc \rceil + dc) \\ = \; & 5888 \cdot 6 + 128 \cdot (2 \cdot 5 + log_2\lceil 46 \rceil + 46) = 43264 \end{aligned} \tag{3.3}$$

For the proposed single pipelined RMAS 1 decoder in Fig. 3.4, the memory size is



(a) comparison at SNR=6



(b) comparison at SNR=5.5

Figure 3.7: Convergence speed of different sorter architecture.

reduced to

$$\begin{aligned} & MEM_{CNU} \\ = \; & (N-K)\cdot \\ & (2\cdot(Min+2ndMin+Index+2ndIndex)+Sign) \\ = \; & (N-K)\cdot(4\cdot(w-1)+4\cdot log_2\lceil dc\rceil+dc) \\ = \; & 128\cdot(4\cdot 5+4\cdot log_2\lceil 46\rceil+46)=11520 \end{aligned} \tag{3.4}$$

Therefore the overall register reduction of proposed architecture is 73%, leading to the following advantages: fewer registers, higher utilization of fuctional units, and reduced complexity. Since high-rate LDPC codes usually have more VNUs than CNUs (in our case: 512 VNUs and 128 CNUs), the elimination of registers from VNU to CNU not only reduces hardware cost but also lowers power consumption of clock tree.

Chapter 4

Simulation and Implementation Results

4.1 Code Performance

Under AWGN channel with BPSK modulation, the performance curves are simulated to determine the required bit-width and maximum iteration number. The simulation parameters of proposed algorithm are 6-bit input quantization (5-bit integer and 1-bit decimal fraction), scaling factor 0.75 for NMS algorithm, and 4 iterations. In Fig. 4.1, the bit-error rate (BER) curves indicate that 4 iterations for the proposed algorithm are sufficient to achieve similar performance of standard BP algorithm with 7 iterations. Furthermore, in the aspect of almost the same code-rate and better error-correcting capability, our CP-PEG LDPC codes outperforms 1.2 dB better than the (255, 239) RS code at BER euqal to 10^{-5}, which reveals the potential of CP-PEG LDPC codes for storage applications and fiber optical communication systems. The overall SNR loss between this work and Shannon limit is only 1.6dB.

4.2 Implementation

The proposed RMAS1 decoder is implemented by standard-cell design flow and fabricated in 90-nm 1P9M CMOS technology. The core occupied 3.84 mm^2 area with 68% utilization. The die photo is shown in Fig. 4.2, and the distribution of CNUs and VNUs is

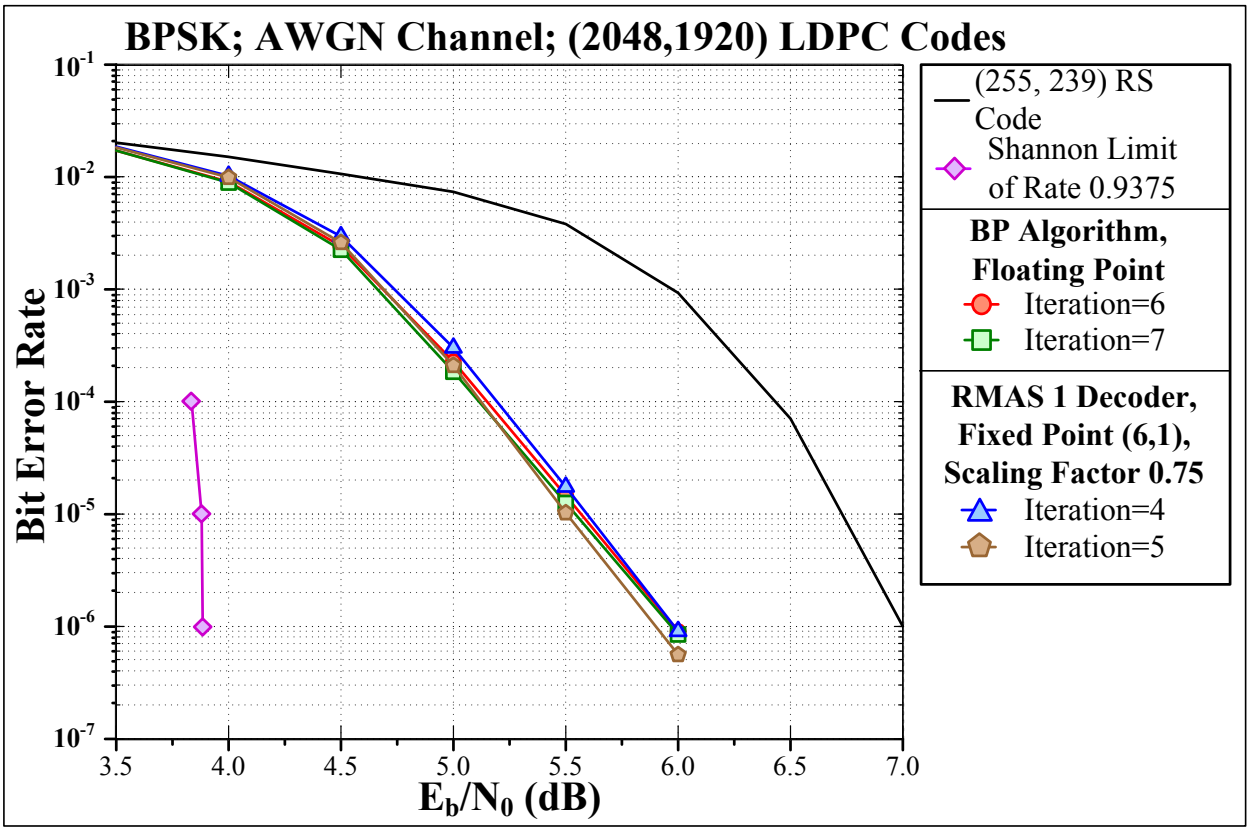


Figure 4.1: Performance.

auto-determined by APR tool. Since required decoding cycles of one LDPC codeword is 4 initialization cycles plussing 4 iterations, the throughput is (1920bit/20cycles)×frequency. Fig. 4.3(b) shows the measured maximum throughput and power dissipation under different SNR conditions and supply voltages. The measurement result indicates that the test chip can achieve 11.5 Gbps throughput under 1.4V supply voltage, where the average power dissipation for E_b/N_0 ranging from SNR 3 to 10 dB is 1037mW. The throughput could be scaled down to 5.77Gbps with 0.8V supply voltage to meet the throughput requirement of IEEE 802.15.3c standard and the energy efficiency will be 0.033 nJ/bit.

Compared with the state-of-the-art in literatures Table 4.1, the proposed LDPC decoder outperforms others in the aspects of throughput, hardware efficiency, and power efficiency. Since the LDPC code specification of these designs are different, the SNR loss between each work to their Shannon limit is also listed for reference.

Table 4.1: Comparison with state-of-the-art

	RMAS1 decoder	CICC'07 [9]	SOVC'07 [10]	SOVC'09 [11]
Process	90-nm	0.13-μm	0.13-μm	65-nm
Code Spec	(2048, 1920)	(660,480)	Wimax	(2048,1723)
Code Rate	0.9375	0.73	0.5	0.84
Core Area	3.84 mm^2	7.3 mm^2	4.45 mm^2	5.35 mm^2
Gate Count	708k	690k	420k	N.A
Iteration	4	15	2 - 8	2-8
Input Quantization	6 bits	4 bits	8 bits	4 bits
Clock Frequency	120 MHz [1] 60 MHz [2]	300 MHz	83.3 MHz	700 MHz [3] 100 MHz [4]
Max. Throughput	11.5 Gbps @ 1.4v [1] 5.77 Gbps @ 0.8v [2]	2.44 Gbps	222Mbps	47.7 Gbps [3,8] 6.67 Gbps [4,8]
Power	1037 mW [1] 191.2 mW [2]	1383 mW[5]	52 mW	2800 mW [3] 144 mW [4]
Energy Efficiency	0.09 nJ/bit [1] 0.033 nJ/bit [2]	0.566 nJ/bit [5]	0.23 nJ/bit	0.058 nJ/bit [3] 0.02 nJ/bit [4]
Hardware Efficiency [6]	12.1	3.5	0.53	N.A
SNR loss to Shannon limit [7]	1.6 dB	2.8 dB	2.9 dB	1.4 dB

[1] 1.4V supply voltage

[2] 0.8V supply voltage

[3] 1.2V supply voltage

[4] 0.7V supply voltage

[5] when E_b/N_0=5.5dB

[6] Throughput/Gate count (Mbps/K-gate)

[7] when BER=10^{-5}

[8] Early termination is applied. Average decoding iteration is used to calculate throughput.

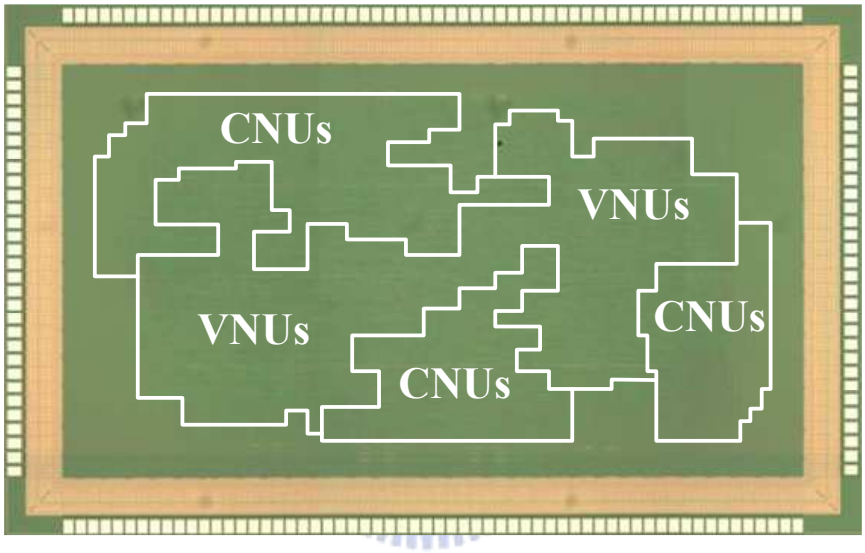
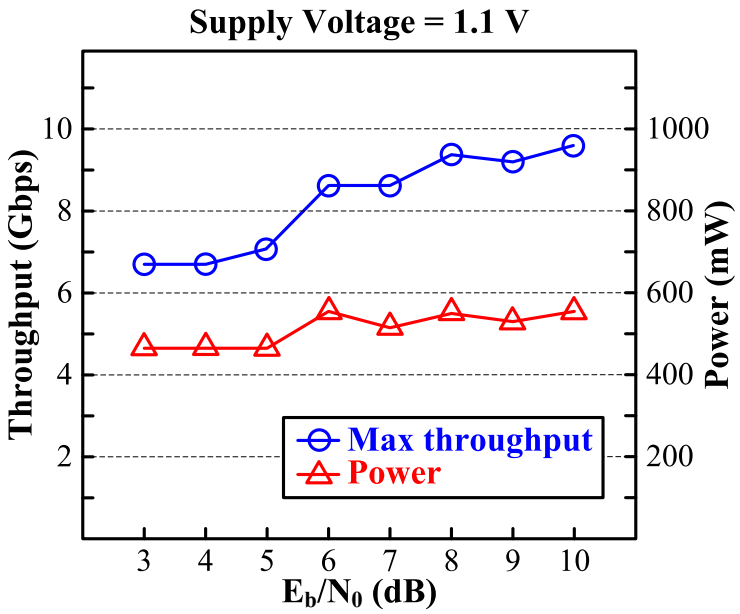
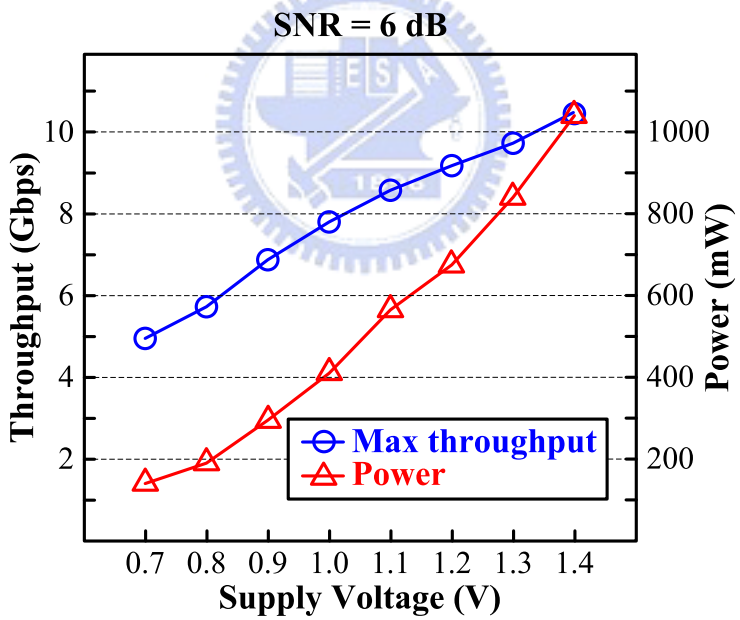


Figure 4.2: Chip micrograph.



(a) at different SNR conditions



(b) under different supply voltages

Figure 4.3: Measured maximum throughput and power consumption.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this thesis, a VSS scheduling high code-rate LDPC decoder is first proposed. The VSS approach results in higher convergence speed. Maximum number of decoding iteration can be reduced to increase the throughput. By utilizing the characteristic of the VSS scheduling, the single pipelined architecture is proposed to enable the messages from CNU to VNU and the messages from VNU to CNU to be computed in one cycle, thus the throughput is almost doubled. Three types of CNU architecture combined with the single pipelined architecture are proposed. In magnitude part, the computation in accumulative way reduces the sorter complexity. In sign part, the sign can be also computed in accumulative way which decrease the input number of XOR gate. Therefore, the computation of CNU in accumulative way results in an area-efficient design. In addition, single pipelined architecture combined with RMAS1 CNU can save 73% message storage memory compared to the conventional partial parallel decoder. The reduction of the registers lowers the power consumption of the clock tree and result in energy-efficient design. Mux reduction technique is proposed by dividing the variable nodes into each group carefully, which can make the connections between CNU and VNU remain the same when different group of VNUS are updated. Therefore, the input of the mux will be the same and the mux is redundant and removable. After implementation in 90nm technology, the test chip occupies 3.84 mm^2 area and supports maximum 11.5 Gbps data rate under 1.4V supply voltage with energy efficiency 0.090 nJ/bit.

CSS approach [4] which was applied to LDPC decoder implementation results in higher convergence speed, lower memory requirement and lower complexity of VNU. Compared to CSS, VSS approach has the same property in memory requirement and convergence speed, but CNU has lower complexity instead of VNU. Moreover, the architecture with CSS approach utilize parallel VNU and partial parallel CNU (one group of CNU). On the contrary, the architecture with VSS approach utilize partial parallel VNU(one group of VNU) and parallel CNU. After reviewing many LDPC codes in different specs or design cases, degree of variable nodes is still small(ex:3,4,5). If there is a LDPC code with large variable node degree, CSS approach will be the suitable candidate to alleviate the VNU complexity.

5.2 Future Work

This thesis has provided the methodology to optimize high code-rate LDPC decoder. However, the decoder architecture is only optimized to support a high code-rate LDPC code. A multi-mode LDPC decoder in future wireless devices may be required to support different code rate. For example, the code rate ranges from 15/16 to 1/2, and maybe the check node degree ranges from 6 to 46. How to design an area-efficient CNU architecture for each code rate will be essential. In addition, code construction methods could be investigated to develop a LDPC code which is more suitable for multi-mode VSS scheduling architecture.

Conventionally, the error correcting codes, such as RS code or BCH code, are required to be applied on FLASH memory to increase the reliability. High code rate LDPC code is also suitable for FLASH memory application because the error correcting ability of LDPC code is better than RS code or BCH code with the similar code rate. The channel in FLASH memory can be considered as higher SNR in AWGN channel, therefore the error floor of LDPC code needs to be investigated to insure that the error correcting ability in high SNR region is also better than RS code or BCH code. As a result, we build a AWGN platform, which will be introduced in Appendix, to investigate the error floor efficiently.

Appendix A

AWGN Core

A.1 Motivation

The CP-PEG LDPC code yield excellent error-correction which is demonstrated in Fig. 4.1; however, many LDPC codes exhibit an error floor, which corresponds to a decrease in the slope in the plot of bit error rate (BER) versus signal-to-noise ratio (SNR). Therefore, error floor of CP-PEG LDPC code should be investigated if the codes are applied for application such as wire line communication or FLASH memory.

Error floor of some LDPC codes appear at very high SNR which correspond to low BER such as 10^{-11}. In addition, some recent standards give maximum allowable BERs of only 10^{-12} for specified SNRs (e.g., IEEE 802.3 10 Gbit/s Ethernet [5]). At least 10^{14} bit of should be to simulated to estimate the BER appropriately. Generally, the performance(BER versus SNR) of LDPC code is obtained by simulation on computer. Nevertheless, 10^9 noise generated by a dual core Pentium processor running at 3.0 GHz with a 1-MB L2 cache takes about 2.5 hour. Therefore, to investigate the error floor efficiently, hardware AWGN generator should be implemented to speed up the simulation process.

Notice: The AWGN core is implemented based on the paper. [12], therefore, the following material in this chapter will be similar to the material in [12] .

A.2 Box-Muller Algorithm

Given two independent radon variable, U_1 and U_2, with standard uniform distribution over $[0, 1)$

$$\theta = 2\pi U_2 \tag{A.1}$$

$$R = \sqrt{-2\ln(U_1)} \tag{A.2}$$

$$X = R\cos(\theta) \tag{A.3}$$

$$Y = R\sin(\theta) \tag{A.4}$$

Then, X and Y are two independent Gaussian distribution $N(0, 1)$

A.3 Architecture

- Uniform Distribution Generation

  U_1 and U_2 can be implemented by utilizing Linear feedback shift register (LFSR) which depicted in Fig. A.1. Due to the finite length of LFSR, the precision of the standard uniform variates is limited to 2^{-WL}.

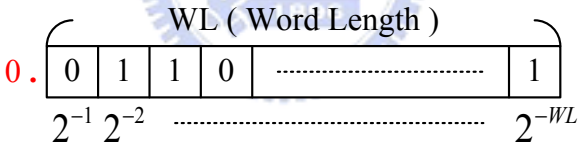


Figure A.1: Linear feedback shift register

- Tail Accuracy

  The normal distribution is an open-ended distribution $[-\infty, +\infty]$ in which values of increasing magnitude occur with increasingly small probabilities. We can observe equations$(A.3)$ and $(A.4)$, then we can find the maximum magnitude of variate x and y is decided by r in equation$(A.2)$. For the same reason, the precision of standard normal variate x, y is limited by r. Fig. A.2 shows that r become bigger when u_1 become smaller.Therefore,the tail accuracy can be decided by the word length (WL) of u_1.When $u_1 = 2^{-WL}$ which is minimum value can be represented in

u_1, the maximum value of r is $\sqrt{2WL\ln 2}$. In this work,WL is set to 32, and the tail accuracy is $\sqrt{64\ln 2} = 6.66$. In addition, the LFSR is implemented using the method [13] which guarantee the period is 2^{113}.

- Efficient Look-Up Table Implementation

  Once the WL of LFSR is decided, the tail accuracy is decided. As shown in Fig. A.2(a), the $R()$ is a non linear function, so the $R()$ need to be implemented by look up table. However, the look up table's output number is 2^{WL}, which result in high complexity and long critical path. Therefore, our goal is to approximate the look-up table with high accuracy and low complexity, and the detail methods will be introduced in the following paragraphs.

- Linear Polynomial Curve Fitting We adopted a linear polynomial curve fitting approach to approximate $R()$ function between $(0, 1)$. In other word, a period of curve $R(u_1)$ can be approximated by $p(u_1) = a \cdot u_1 + b$ over interval $[\alpha, \beta)$ with $p(\alpha) = R(\alpha)$ and $p(\beta) = R(\beta)$. Take $\alpha = 0.9$ and $\beta = 1$ as example which is shown in Fig. A.2(c).

- Hybrid (Logarithmic and Uniform) Segmentation

  As shown in Fig. A.2(a) and Fig. A.2(b), the function $R(u_1)$ has two regions with higher slope. One region is in the vicinity of $u_1 = 0$, and the other is in the in the vicinity of $u_1 = 1$. In the middle part of the curve where it is relatively linear. Therefore, it would be beneficial to utilize large segments for the linear region(middle), and small segments for non-linear region(both side). As shown in Fig. A.4, logarithmic segmentation is suitable for $R()$ implementation. Furthermore, we can compensate the approximation error by dividing logarithmic segmentation uniformly into $L = 2^l$ subsegments where each each subsegment is demoted by $s_{r,w,l}$. Each segment could be approximated by $p(u_1) = a_{r,w,l} \cdot u_1 + b_{r,w,l}$.

- Scaling Factor The coefficient $(a_{r,w,l}, b_{r,w,l})$ of each segment is stored in memory or look up table;When u_1 falls on corresponding segment, $p(u_1)$ is computed by the linear equation.The coefficient which can be represented in two's complement 16-bit fixed-point with 12-bit fraction , i.e.,in Q(16,12). Large amount of coefficient need to be stored;therefore, the appropriate bit width of coefficient to represented with

low approximation error is important. The scaling factor method could be applied to enhanced hardware efficiency.

When u_1 lies within r_0 and in segment $s_{r,w,l}$, a new scaled variable $\hat{u}_{1,w}$ is defined as $\hat{u}_{1,w} = 2^w u_1$. To compensate for scaling of u_1, the $a_{0,w,l}$ slope of segment $s_{0,w,l}$ is shifted to the right w bit positions as $\hat{a}_{0,w,l} = 2^{-w} a_{0,w,l}$. Thus, the large slope values which appear when u_1 is in the vicinity of 0 and 1(shown in Fig. A.2(a)) could be scaled down by 2^{-WL+2}. Similarly, when u_1 resides in segment s_1, w, l, the large value of a_1, w, l can be scaled by 2^{-w} and $b_{1,w,l}$ can be adjusted as $a_{1,w,l} + b_1, w, l+$. When u_1 resides in segment $s_{0,w,l}$,then $p(\hat{u}_1)$ will be computed as

$$p(\hat{u}_1) = \hat{a}_{0,w,l}\hat{u}_1 + b_{0,w,l} = 2^w \hat{a}_{0,w,l} u_1 + b_{0,w,l} \tag{A.5}$$

When u_1 resides in segment $s_{1,w,l}$,then $p(\hat{u}_1)$ will be computed as

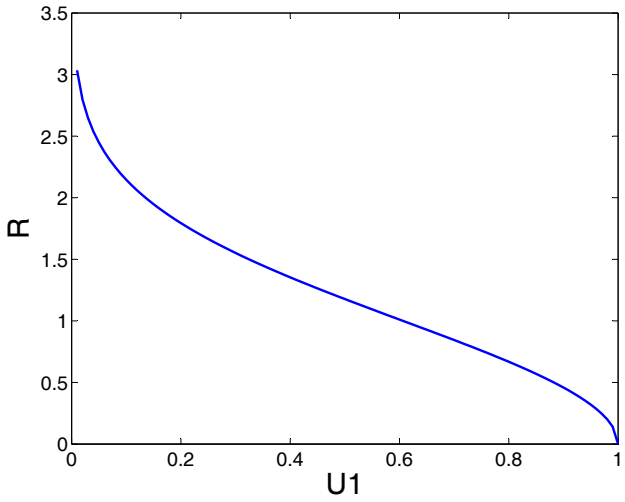
$$p(\hat{u}_1) = -2^w a_{1,w,l}\hat{u}_1 + (a_{1,w,l} + b_{1,w,l}) \tag{A.6}$$

- Address Generator(AG) and Cosine Sine Function

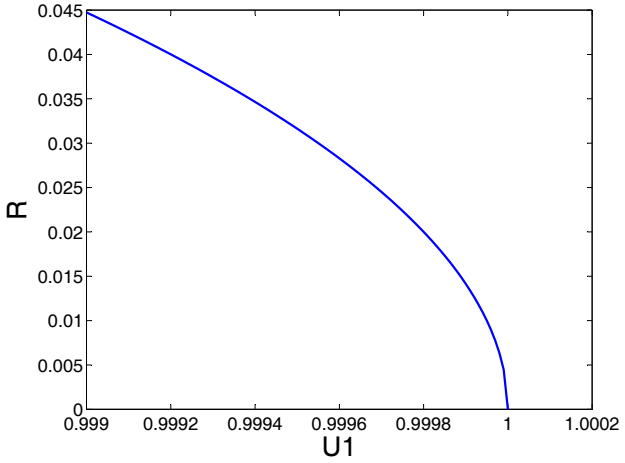
  AG is used to address the coefficient memory($a, b, \hat{u}_1$) when u_1 is located at correspond range. The cos and sin function is implemented by only look up table. The quarter cycle of the sin function is partitioned into 1024 segments;Using the symmetry of sin and cos function the cos and sin could be approximated compactly.

- Overall Architecture

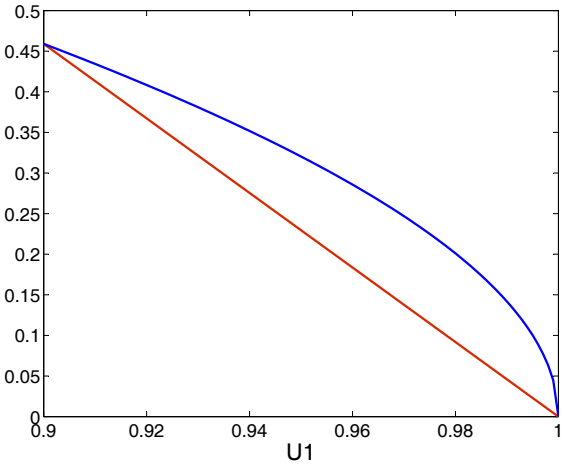
  Fig. A.3 shows the overall six-stage pipelined architecture of AWGN noise generator. The Box-Muller method generate $N(0, 1)$ from stage 1 to stage 5. The sixth stage is used to multiply variance(σ) to $N(0, 1)$ variate to generate $N(0, \sigma)$ variate. The σ with corresponding SNR of is configured in the beginning of noise generation. The bit-width of each part of AWGN generator is plotted in Fig. A.3.



(a) U1 range $(0, 1)$



(b) U1 range $(0.999, 1)$



(c) Curve fitting example; red line represent $p(u_1)$, and blue line represent $R(u_1)$

Figure A.2: R.() versus u1

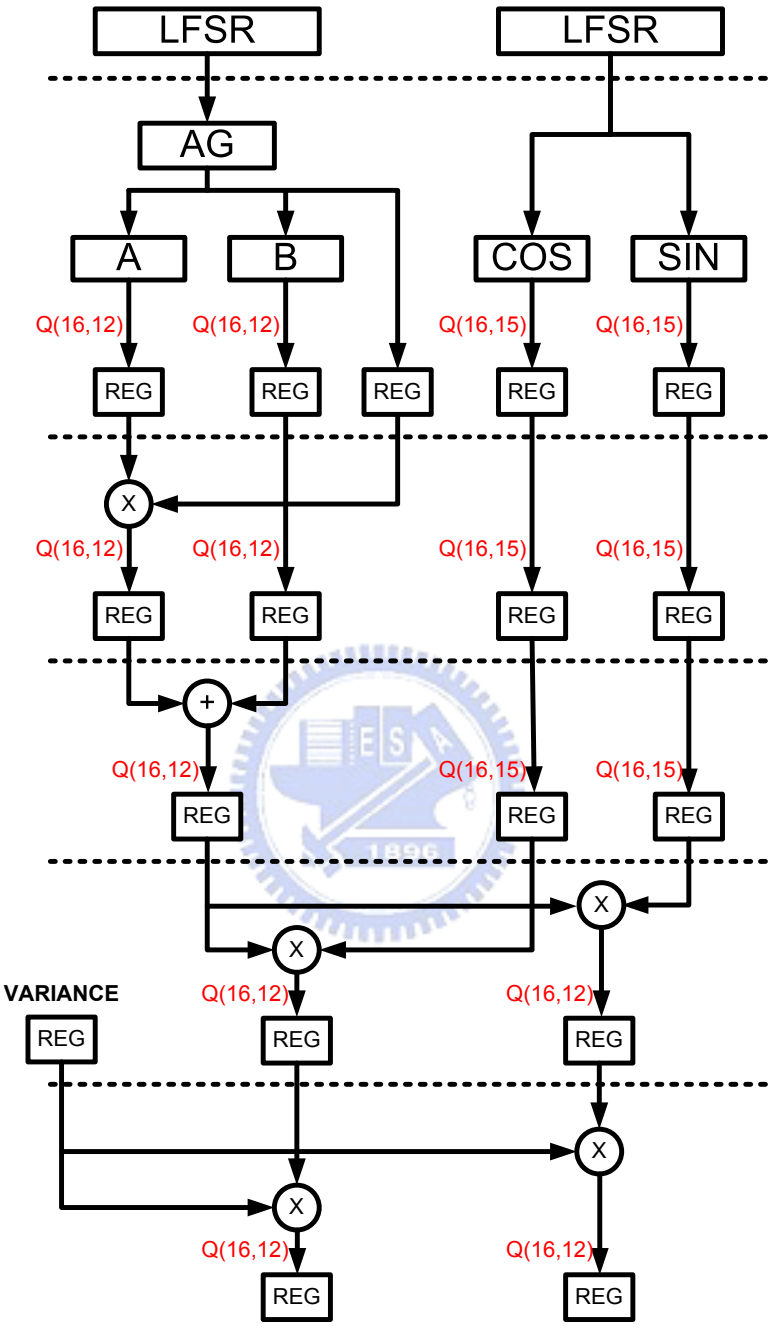


Figure A.3: box-muller architecture

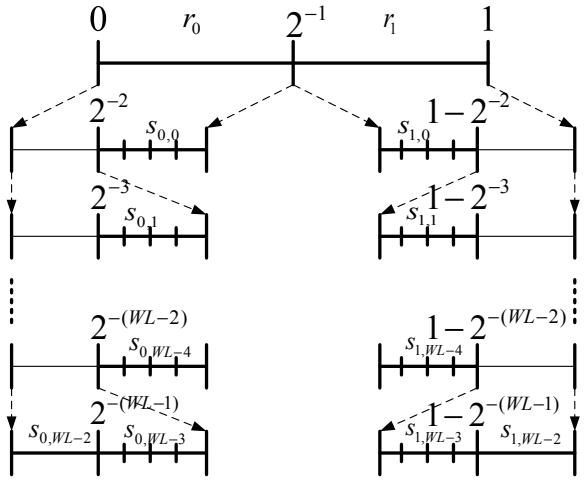


Figure A.4: Hybrid (logarithmic and uniform) segmentation of $u_1 \in (0, 1)$

A.4 FPGA emulation

The AWGN core is implemented and verified on workstation. Then, we prototype the AWGN core with BPSK and Viterbi Decoder on FPGA. As shown in Fig. reffpga, we could download our design via USB and Cinfigure FPGA via COM port. The FPGA platform contain software(UART[1]) which is used to communicate between FPGA and computer and which could support real time performance curve plot.

Through UART, we could configure FPGA and set emulation parameter conveniently and get the emulation result in real time. When the FPGA emulation is done, the error number will be sent from FPGA to UART, then the BER could be computed by computer.Fig.A.6 shows the UART interface.

The detail function is listed in table A.1. As shown in Fig.A.7, the performance of viterbi decoder is emulated with AWGN core and BPSK mapping. In table A.2, we could observe that the emulation time of FPGA is 60 times faster than computer simulation;thus, we could investigate the performance at very low BER region.

[1]Uart is mainly accomplished by Jia-Lung Lin, Kin-Chu Ho and Chih-Lung Chen

Table A.1: Function of UART

COM	Set com port number and Baud Rate between PC and FPGA
Deliver	Set the function to be execute
AWGN Simulation	Set the SNR of AWGN channel and number of transmitted data
Pattern	Set the binary data to configure GPIO

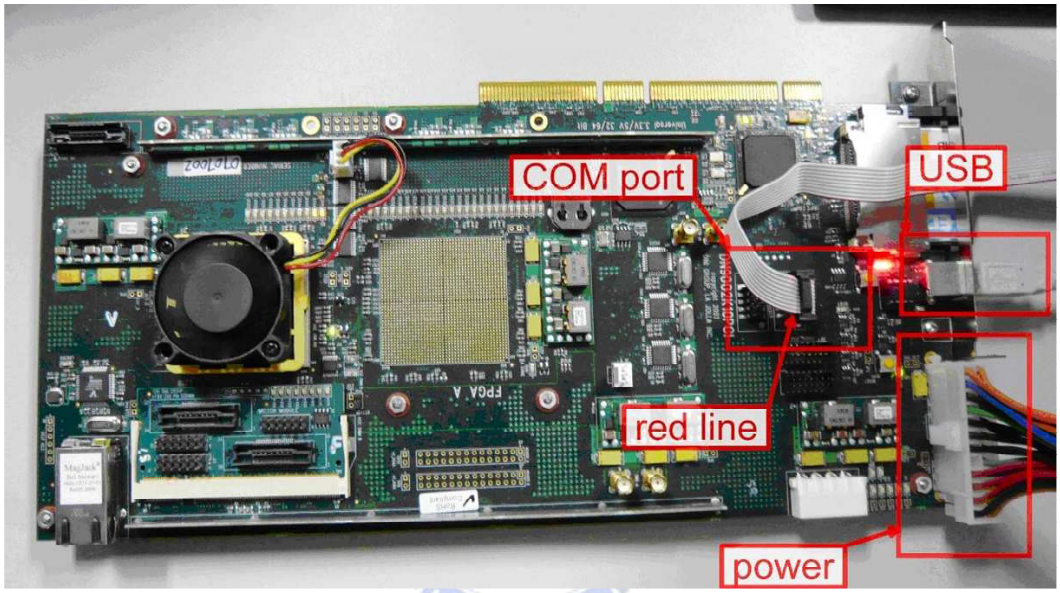


Figure A.5: BPSK emulation using FPGA:DN9200K10PCI

Table A.2: Comparison the simulation time

Platform	Intel Xeon(TM) CPU 3.20GHz ram: 2048 KB	Xilinx virtex5 clk frequency: 50Mhz[1]
decode 10^7 bit	12 second	0.2 second
decode 10^{11} bit	12000 sec = 33 hr	2000 sec = 0.5 hr

[1] Due to baud rate limit of COM port, actually, AWGN core can operate around 100 Mhz

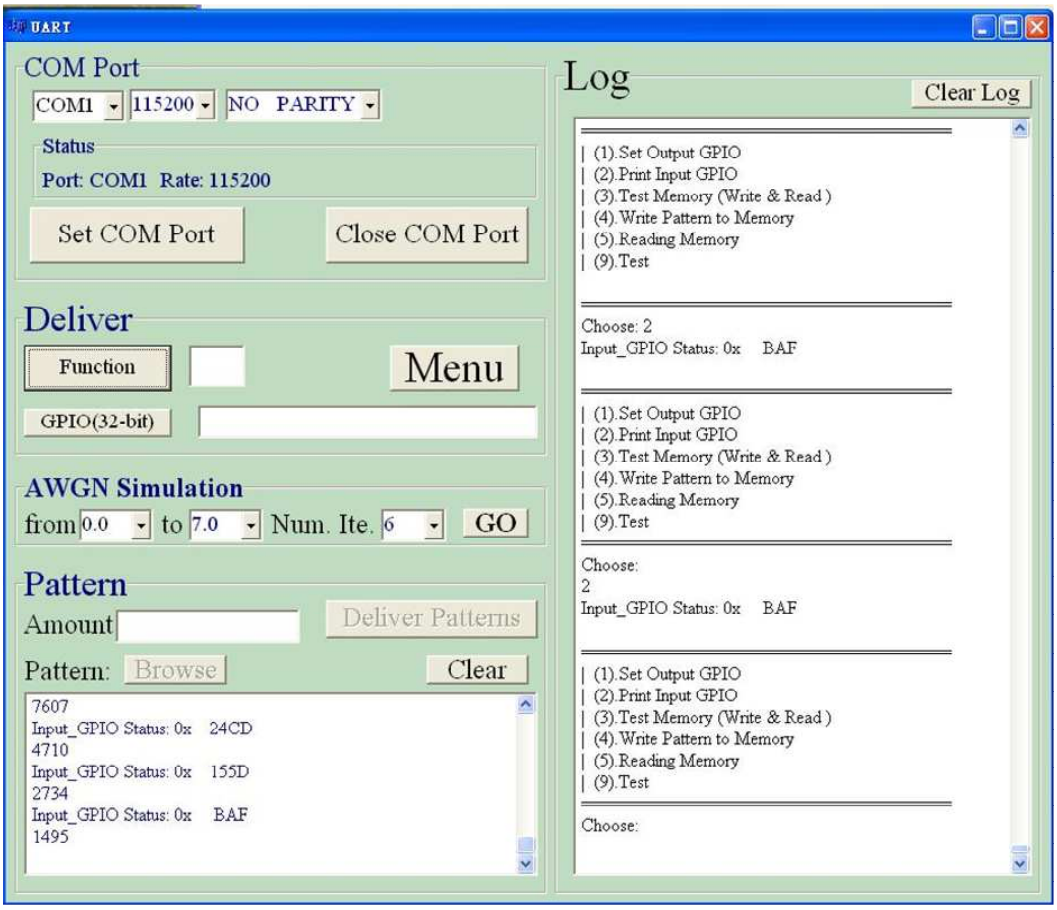


Figure A.6: The interface of the UART program

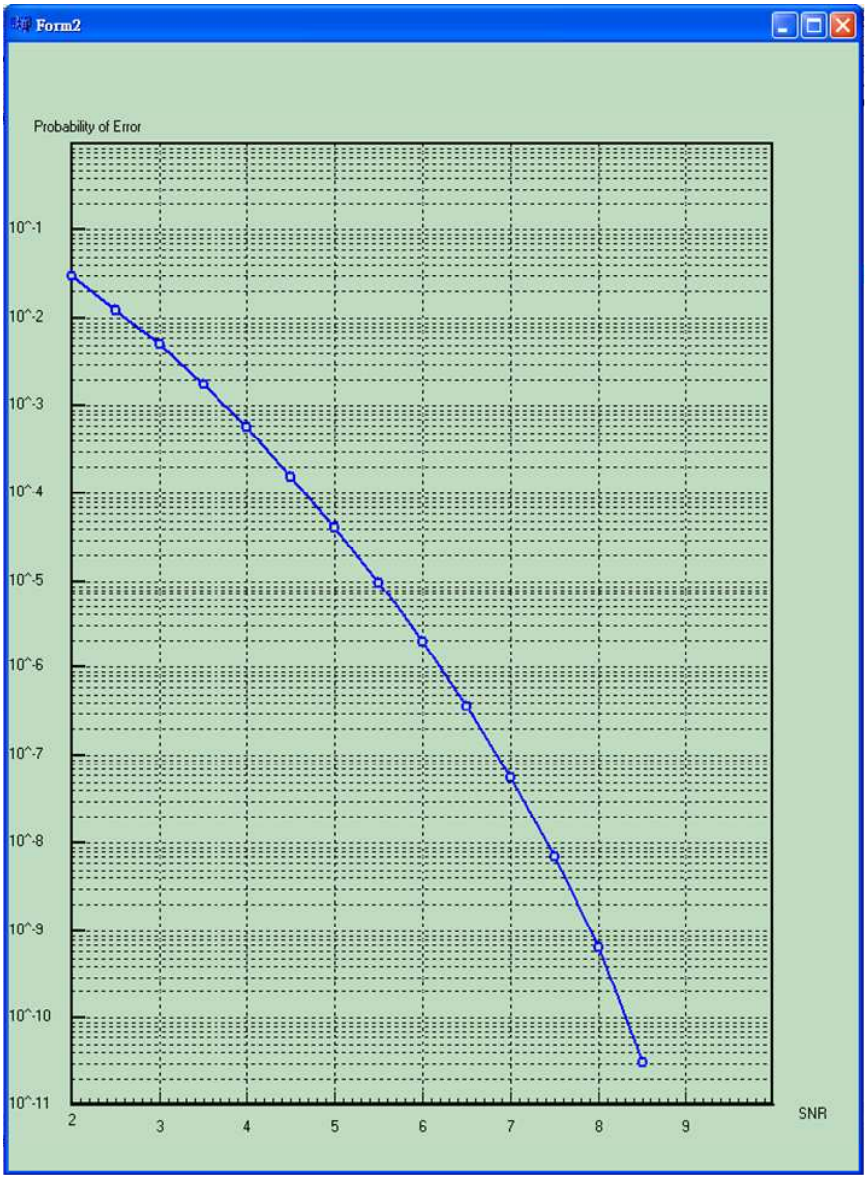


Figure A.7: Emulation result: BPSK with Viterbi decoder performance curve

Bibliography

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

[2] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," *Electron. Lett.*, vol. 33, no. 6, pp. 457–458, Mar. 1997.

[3] M. Mansour and N. Shanbhag, "High-throughput LDPC decoders," *IEEE Trans. on VLSI Systems*, vol. 11, no. 6, pp. 976–996, Dec. 2003.

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

[5] *Part 3: carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specificaions*, IEEE Std. P802.3an-2006, Sept. 2006.

[6] *Part 15.3: wireless medium access control (MAC) and physical layer (PHY) specifications for high rate wireless personal area networks (WPANs)*, IEEE Std. P802.15.3c-DF8, 2009.

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

[8] Y. K. Lin, C. L. Chen, Y. C. Liao, and H. C. Chang, "Structured LDPC codes with low error floor based on peg tanner graphs," in *IEEE Int. Sympo. Circuits and Systems (ISCAS'08)*, May 2008, pp. 1846–1849.

[9] A. Darabiha, A. C. Carusone, and F. R. Kschischang, "A 3.3-Gbps bit-serial block-interlaced min-sum LDPC decoder in 0.13-μm CMOS," in *Proc. IEEE CICC'07*, Sept. 2007, pp. 459–462.

[10] X. Y. Shih, C. Z. Zhan, C. H. Lin, and A. Y. Wu, "A 19-mode 8.29mm^2 52-mW LDPC decoder chip for IEEE 802.16e system," in *Proc. Int. Sympo. VLSI Circuits (SOVC'07)*, June 2007, pp. 16–17.

[11] Z. Zhang, V. Anantharam, M. Wainwright, and B. Nikolic, "A 47 gb/s LDPC decoder with improved error rate performance," in *Proc. Int. Sympo. VLSI Circuits (SOVC'09)*, June 2009.

[12] B. F. C. A. Alimohammad, S. F. Fard and C. Schlege, "A compact and accurate gaussian variate generator," *IEEE Trans. on VLSI Systems*, vol. 16, no. 5, pp. 517–527, May 2008.

[13] P. LEcuyer, "Tables of maximally equidistributed combined lfsr generators," *Math. Comput. Archive*, vol. 68, no. 225, pp. 261–269, 1999.