國 立 交 通 大 學

電子工程學系 電子研究所碩士班

碩 士 論 文

區塊渦輪編解碼器設計與實現

Design and Implementation of
Block Turbo Code Codec

研 究 生：廖俊閔 Jimmy J.M. Liao

指導教授：張錫嘉 博士 Dr. Hsie-Chia Chang

中 華 民 國 九 十 六 年 十二 月

區塊渦輪編解碼器設計與實現

Design and Implementation of Block Turbo Code Codec

研 究 生：廖俊閔　　Student ： Jimmy J. M. Liao

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

國 立 交 通 大 學

電子工程學系 電子研究所碩士班

碩 士 論 文

A Thesis

Submitted to Department of Electronics Engineering & Institute of Electronics

College of Electrical & Computer Engineering

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of Master

in

Electronic Engineering

June 2007

中 華 民 國 九 十 六 年 十二 月

區塊渦輪編解碼器設計與實現

學生：廖俊閔　　指導教授：張錫嘉　博士

國立交通大學

電子工程學系　電子研究所碩士班

摘要

本論文為區塊渦輪碼(Block Turbo Code)編解碼器電路設計之研究，原區塊渦輪編碼演算法裡，每一個迴圈計算時皆需要實驗性參數來修正計算量，針對此本論文提出了一個類幾何演算法取代實驗性參數修正計算量的方式，所提出來的新演算法亦適合硬體電路實現。我們設計的範例是以 WiMAX 為本論文電路應用，並考量及提出電路平行化處理方法在有限的硬體資源限制下達到我們要的效能。 本篇論文電路設計流程平台不單只以 C 語言來實現我們演算法階層的創意，及用硬體描述語言 Verilog 來實現新的硬體架構，我們也同時使用了 SystemC 的驗証平台來減少我們電路架構驗証所需時間，且 SystemC 所建的模組亦提供未來電子系統層級電路設計使用。

Design and Implementation of Block Turbo Code Codec

Student：Jimmy J.M. Liao Advisor : Dr. Hsie-Chia Chang

Department of Electronics Engineering

Institute of Electronics

National Chiao-Tung University

Abstract

In this thesis, a block turbo code of 802.16e is proposed. Unlike the conventional decoding algorithm requiring empirically derived parameters, the proposed geometric-like algorithm uses hamming distance to compensate the information. Not only improving the error performance, the proposed algorithm also facilitates hardware implementation. Moreover, a design methodology for parallel architecture is presented to meet various throughputs. The memory accessing hazard in parallel architecture can be overcome by the proposed multi-bank-array algorithm. The proposed algorithm is a partition and scheduling technique without extra memory. By the proposed algorithm and parallel design methodology, the block turbo code encoder and decoder defined in WiMAX(802.16e) is implemented. Note that, a design

flow from algorithm level (in C language) to hardware level (in Verilog) is presented. A systemC model is also built to provide a more efficient verification strategy and allows electronic system level design.

誌謝

本論文得以完成，首先非常感謝 OCEAN 團隊的每位伙伴及感謝我的家人；因為有 OCEAN 研究團隊的研究環境，才能讓我在碩士生涯得到真正地成長紮實且受益良多，也因為家人的支持，才能讓我專心於研究並取得碩士學位。並感謝口試委員們給學生很多寶貴的建議及想法，讓學生能更廣泛思考。

感謝溫瓌岸教授(溫媽)及 TWT 實驗室曾給予我的栽培，給我充份的資源環境去思考研究方向與問題，感謝 TWT 博士班學長文安長期以來的照顧；謝謝 TWT 碩士班的學弟： 書旗、家岱、漢健、柏麟等學弟在我碩士生活中一起成長及分享; TWT 碩士班同學：振威、志德、彥宏、彥凱、懷仁、俊憲，曾一起生活過，特別感謝最夠義氣的振威一直支持我幫助我走下去，也誠心感謝 TWT 已畢業博士班學長莊源欣曾給我許多指教及建議。

在求學旅程中許許多多的恩師，因為有你們不斷地帶給我支持與希望，我才有機會從一個很普通的學生再到交大來深造；感謝專科導師陳文淵教授在專科時期的指導;感謝高工導師蔡孟烈老師，雖然只在老師班上短短不到一年，但因為有孟烈老師給了我求學慾及不放棄的新力量還有老師不求回報的付出，我才能脫胎換骨；謝謝高工老師簡文魁老師，文魁老師亦師亦友不斷地精神支持及鼓勵，讓我感受到老師如同朋友，而這個力量讓我多了一份自信；感謝高三導師陳天賜老師協助我學業上很多問題並提供我讀書方法及高二導師江進文老師的栽培，而其他高工老師傅慈貞、黃麗娜、張麗娜、張珠兒．．．．等等老師們，也謝謝你們教導過我。再來要感謝國中導師羅淑鳳老師未曾放棄任何一個在普通班頑皮的學生，讓我的未來不是夢不放棄求學的可能性，在叛逆的青春期中協助不夠成熟的小朋友把持住善良的本性;另外國中老師林淑靜老師、薛雲芳老師、童軍團的盧嬌老師及其他曾帶給我希望的國中老師，也謝謝他們帶給我學習的興趣與希望。

接著要感謝朋友團，我的十幾年的高工同學朋友們，志文、崇裕、俊臣、宏

文，因為十多年來有你們不管經濟上或是精神上的協助與打氣，我才能堅持到底；也感謝專科同學朋友們，清宏、銘益、傳盛、吉興、彥佑、佳仁、振麒、益祥，總是在我一些新知及技術上需要幫助時給我強大的後盾。還有感謝學弟義凱在我最後二年碩士生活共同成長互相學習照顧，讓我這後面二年仍能充滿戰鬥力；感謝交大博士邵雲龍學長、張彥中(nelson)學長及劉子明(明哥)學長在我碩士生涯中最無力時打氣，感謝圖書館蔡淑琴大姐協助我圖書諮詢及學習；還有所有曾協助過我的研究助理小姐們也謝謝你們。

特別要感謝 OCEAN 團隊博士班建青學長、彥欽學姐及其他博士群，因為你們對我的信任與支持讓我到 OCEAN 團隊成長，更教給我很多很多很多的知識與方法，我真的很感謝學長姐無私的分享與教導，才能成就我這段研究；還有要感謝張錫嘉教授，在我需要討論研究方向與問題時給了很多建議及協助；感謝胡樹鑫教授長期以來對我的信任及幫忙，我才能達成我的理念;感謝黃俊達教授給我精神上的支持還有學業上的啟發。

最後再次謝謝我母親及我的家人，及上述所有老師朋友同學們，僅以此論文回饋與分享給你們，謝謝你們，因為有你們才有我今天的成果。

廖俊閔

2007 年 12 月

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Communication is required in the world, and it is the most important to receive the exact message that is transmitted. Therefore, the protecting message in noise channel is the major topic. A general digital communication system is illustrated in Figure 1.1. The "binary source" is message, where I_i (i= 0, 1,.., n) represents different message symbol. "Binary encoder" adds redundant parities to protect message. e_i (i=1, 2,…,n) are codewords after encoding. Codeword is modulated into analog signal X(t) that is transmitted via channel, and the received analog signal Y(t) including noise. Demodulation deals with the recovery of Y(t) waveform and estimates the channel value to next stage, "Binary Decode". "Binary Decode" is a process of error correction for r_i (i=1, 2, …, n) symbol. Finally, the message I'_i (i= 0, 1,.., n) can be get in the binary sink of receiver. Of course we expect I'_i is the same as I_i, so the good method of error correction is one of the key factors when I'_i is not the same as I_i.

Channel coding deals with the problems of detecting and correcting the transmission errors that are introduced by the noise of channel. A good channel coding design is to construct encoders and decoders in such a way as to effect:

<1> Fast encoding of messages;

<2> Easy transmission of encoded codeword;

<3> Maximum transfer of information per unit time:

<4> Maximal detection or correction capability.

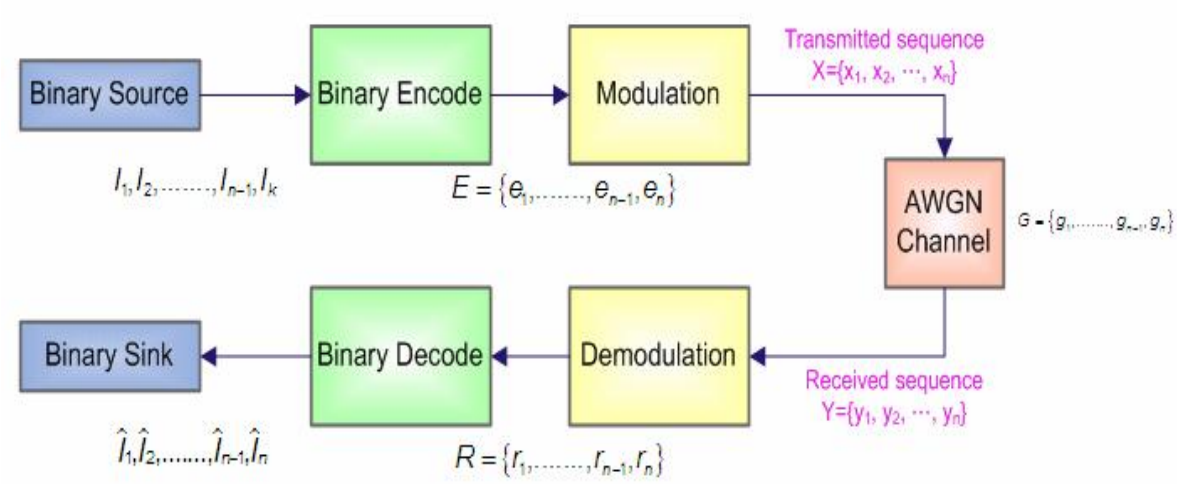


Figure 1.1: Communication system block diagram

There are two kinds of the error correlation code, which are block code [14] and convolutional code [15]. Block code is a code that a block of message is encoded at same time. There is no time dependency between each message. Convolutional code differs from block code in that the encoder contains memory and the encoder outputs at any given time unit depend not only on the inputs at that time unit but also on some number of previous inputs. All of these codes are unique-decoding. Conventional unique-decoding can correct up to $t = \frac{d-1}{2}$ errors, where d is hamming distance. In this thesis, block turbo code is our focus. Block turbo code is a block code, and block turbo code has been widely applied in many applications, such as VDSL2, IEEE 802.16, and so on.

Block turbo code can overcome bust error [12] and is used in applications requiring either high code rates or low complexity. Block turbo code is also an

application of list decoding, but it is not only an application of unique-decoding. List decoding was proposed by Elias and Wozencraft [16]. The list decoding algorithm outputs a candidate list of codewords as answers. We can use the list to find out one of the candidates that is the solution of decision codeword and the bound ($t = \frac{d-1}{2}$) of correcting ability is enlarged.

1.1 Motivation

There are two parameters (α, β), which are experimental values in conventional block turbo code iterative decoding algorithm. These two parameters can be tuned by try-and-error method with experience to a correct combination, but a systematic approach is required to eliminate these two disturbing parameters. A parallel architecture of hardware issue is our problem in implementation. The gate count has to be saved and the requirement of speed is also met so that we want to find out a parallel architecture. Our proposed algorithm and hardware are implemented according to WiMAX specification for example. The algorithm can be also employed in other systems.

1.2 Organization

The organization of this thesis is overviewed as following: Chapter 2 presents the conventional methodology of block turbo code. The innovated iterative decoding algorithm is proposed in Chapter 3. The hardware implementation with our new algorithm is demonstrated in Chapter 4. Chapter 5 illustrates our algorithm level performance and hardware level performance. Chapter 6 concludes with a summary

of the contribution and the future works.

Finally, The Figure 1.2 shows the flow chart of this research. In the Figure 1.2, the step FPGA emulation and the step Back End are our future work.

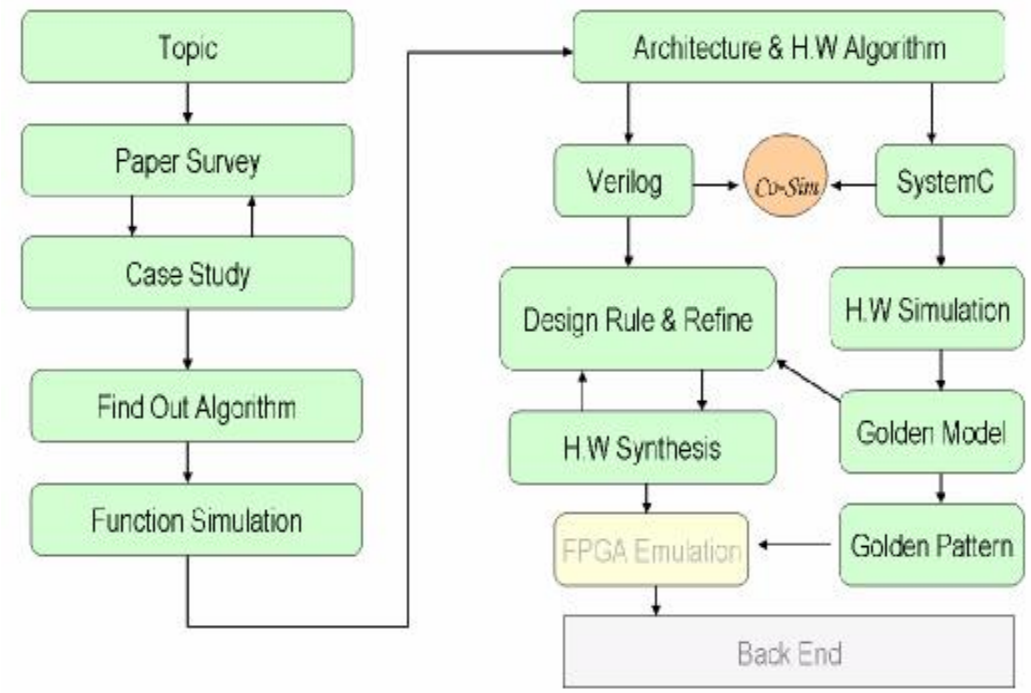


Figure 1.2: The flow chart of the research

Chapter 2

Block Turbo Code

Block turbo code (BTC) is a concatenated code and the performance can be achieved close to the Shannon's theoretical limit. Concatenated block coding was first introduced by P. Elias in 1954 [1]. In 1993, a hyper-dimension turbo code was presented [2] and the code is based on the extended Hamming code with iterative MAP decoding. As for BTC, it was first proposed by Pyndiah in 1994[3]. In this chapter, a BTC study is presented. The concepts of BTC will be described in section 2.1. All related decoding algorithms are shown in section 2.2. Based on this algorithm, (WiMAX) application [4] is introduced in section 2.3.

2.1 Encoding of BTC

BTC is composed of two or more block codes with respect to different dimensions of BTC. For example, two linear block codes as BTC component code, C_1 and C_2 are assumed as BTC component codes. Their parameters are (n_1, k_1, d_1) and (n_2, k_2, d_2), where n_i is codeword length, k_i is the message length, and d_i is the minimum Hamming distance, i=1, 2. $C_b = C_1 \otimes C_2$ is defined as the BTC encoding result. The encoding steps are illustrated in Figure 2.1. The row space is encoded, and the column

space is encoded sequentially. The rows of matrix C_b are the codewords of C_1, and the columns of matrix C_b are that of C_2.

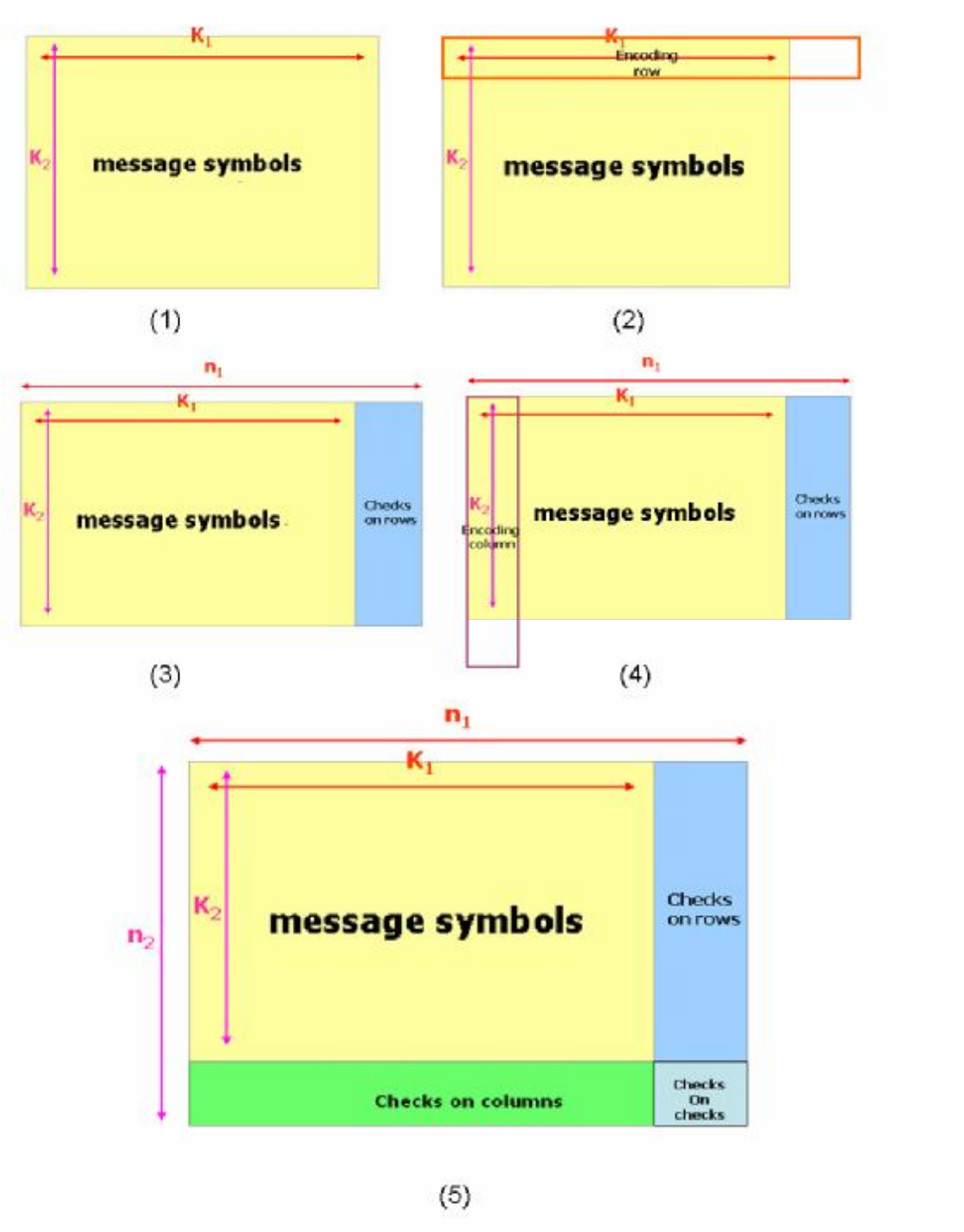


Figure 2.1: The flow diagram of BTC encoding

The parameters of BTC C_b are just the products of component code C_1 and C_2. The C_b parameters are defined as (n_b, k_b, d_b) which are $n_b = n_1 \times n_2$, $k_b = k_1 \times k_2$, and $d_b = d_1 \times d_2$ respectively. For this reason, BTC is also called turbo product code (TPC). The capability of error is improved by these properties. The message symbols of row space are protected by the parities of row (checks on rows); likewise, the message symbols of column space are protected by the parities of column (checks on columns). Finally, the checks on checks is encoded out to protect the parities of the row and column space. The processes are encoding and interleaving the message array in the encoding procedure. This property of encoding can overcome the burst noise interference and improve the performance.

2.2 Decoding of BTC

List decoding of error correcting codes is the method of enlarged traditional error correction. If the radius can be enlarged, more codewords can be included in candidate list so there are more opportunities to find out which candidate is the closest to channel value of the receiver with hamming distance or soft distance. However, these methods don't require more and more redundant parities for message protection.

In this section, we are going to introduce two list decodable codes algorithms: chase algorithm and sliding encoding window algorithm. These two algorithms are methods of creating the candidates list. This section also introduces the iterative decoding algorithm for soft in soft out (SISO) computation. All other list decodable algorithms not related to BTC are not discussed here.

2.2.1 Chase Algorithm

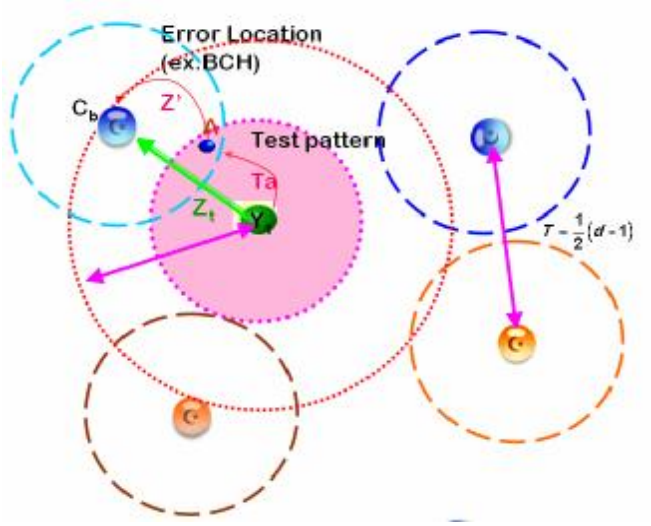


Figure 2.2: The test pattern permutation of Chase algorithm

The numbers of codewords increase exponentially with the number of encoding bits. Generalized minimum distance (GMD) [5] computation is also exponent of encoding bits. Chase proposed an algorithm with a low computation complexity and small performance degradation in 1972[6]. The Chase algorithm provides an approach for codeword permutation beyond the radius of hamming ball which is illustrated in Fig2.2

In the Fig2.2, Y_1 is the received codeword. The test pattern T_a is created to permute Y_1 to A. After the algebraic decoding method is used to decode A , an error location Z' is generated. The Z_t can be derived as the following equations:

$$\begin{aligned} A &= Y_1 \oplus T_a \\ C_b &= A \oplus Z \\ Z_t &= Y \oplus C_b = T_a \oplus A \oplus A \oplus Z = T_a \oplus Z_b \end{aligned} \tag{2.1}$$

Y_1 is decoded to C_b by Z_t. In the same way, we make more test patterns to generate

candidates into candidate list.

There are three different methods in Chase algorithm for making T_a :

Chase 1: The test error pattern set T is given through all error sequences of binary weight less than or equal to $\left\lfloor \frac{d_{min}}{2} \right\rfloor$, and the amount of test patterns can be indicated as $|T| = \binom{n}{\left\lfloor \frac{d_{min}}{2} \right\rfloor}$.

Chase 2: The test error pattern set T is calculated by using all binary vector combinations corresponding to the $t = \left\lfloor \frac{d_{min}}{2} \right\rfloor$ that are low reliable positions, i.e., $|T| = 2^{\left\lfloor \frac{d_{min}}{2} \right\rfloor}$.

Chase 3: The (d_{min} - 1) low reliable positions are extracted from the codeword. The vectors in T have "1" in the i low reliable positions and "0" else where i=0, 2... $d_{min} - 1$ and i= 0, 1, 3,…$d_{min} - 1$ for odd and even d_{min} respectively, i.e., $|T| = \left\lfloor \frac{d_{min}}{2} \right\rfloor + 1$.

2.2.2 Sliding Encoding Window (SEW) Algorithm

Another candidate searching approach is SEW algorithm. The algorithm uses the characters of cyclic code and block code encoding method to implement decoding. It was proposed in 2005 [7] [8].

A q-ary cyclic code is still a valid codeword after a valid codeword is shifted for S symbols so shifting S symbols produce more candidates which are in the neighborhood of the received sequences and valid codewords. The SEW algorithm consists of two steps to generate the candidates list:

(A) Sliding phase (SP) :

A systematic code (n, k) is given. One symbol is defined as w bits. The received codeword is cyclically shifted for S of Δ q-ary symbols and store all results in buffer for the next phase.

(B) Encoding phase (EP) :

The first k symbols of each SP result are extracted in order which is called a window. The window is like a new message part, and then t_n low reliable bits in the window are combed out and do bit-flipping for all combination, where bit-flipping means that the bit is inversed. Finally, the different combinational windows are encoded and the new code words is yielded.

The new codes in our candidates list are generated by these two steps, and the amount of test patterns $|T| = \frac{N}{\Delta} \times 2^{\left\lfloor \frac{d_{\min}}{2} \right\rfloor}$. The algorithm sketch is illustrated in Figure 2.3. C1 is the received sequence. C2 and C3 are the execution result of the SP. The window and the low reliable bits in Figure 2.3 are used to explain the EP.

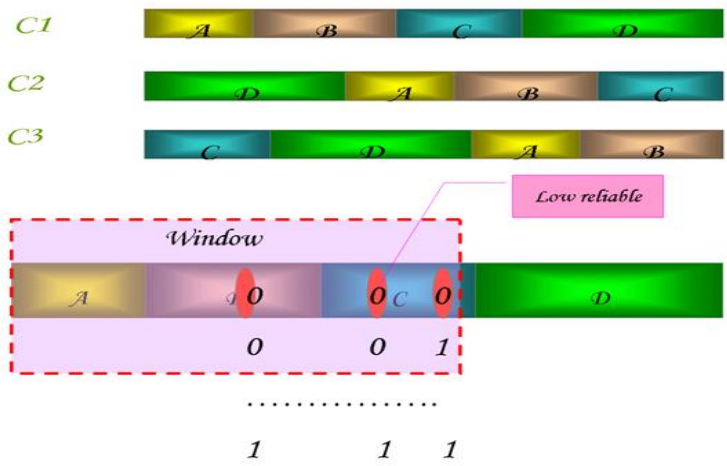


Figure 2.3: The execution phase of SEW

2.2.3 Iterative Extrinsic Information Algorithm

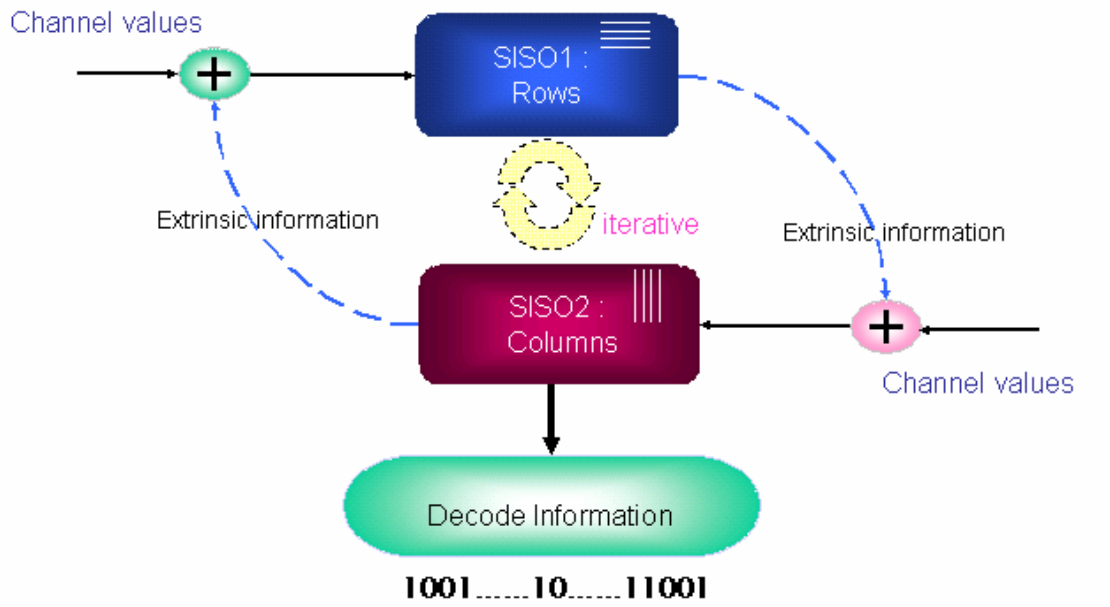


Figure 2.4: The flow diagram of iterative decoding algorithm

BTC is a soft-decision code. Hence, soft in soft out (SISO) iterative algorithm is used to compute the extrinsic information with intrinsic information. Every dimension of BTC decodes in rotation. An example of the two dimensional BTC is shown in Figure 2.4 diagram.

In the first run, the SISO1 decodes all rows of BTC when all channel values are received from de-modulation. Then, the SISO2 decodes all columns of BTC in the next run. When the second run has been done, it is called that one iteration is finished at this time. In other words, one iteration takes 2 runs. A hard-decision decoder decodes the final information of the codeword after several iterations. Then, the decoding information is yielded.

Let C (n, k, d) be a binary linear block code in which n, k, and d represent length, message length, and minimum distance, respectively. Binary phase shift keying (BPSK) is used to modulate a codeword which is transmitted over an additive white

Gaussian noise (AWGN). E= (e_1, e_2, ……e_n) and R= (r_1, r_2, ……r_n) are assumed as the transmitted codeword and receiver vector respectively, where $e_j \in \{+1, -1\}$, j=1, 2, …, n. R is the combination of E and G, and G=(g_1, g_2, ……g_n) are AWGN samples of normal distribution $N(0,\delta^2)$. C^i is defined as the i-th codeword of list. The creation of the list has been introduced in section 2.2.1 and 2.2.2. In this study, the chase 2 algorithm and SEW algorithm are discussed. Now, the candidates of the list are computed for the decision codeword D_c which is closest received channel value. D_c can be obtained as following equation:

If

$$\left|R-C^l\right|^2 \le \left|R-C^i\right|^2 \quad , l \ne i \text{ and } C^i, C^l \in \text{list} \tag{2.2}$$

then $D_c = C^l$

where $$\left|R-C^l\right| = \sum_{j=1}^{n}\left(r_j - C_j^l\right)^2 \tag{2.3}$$

The log likelihood ration (LLR) is used to measure the reliabilities of the Soft de-mapping. The LLR of each element y_j is given by the relation:

$\Lambda(y_j) = \ln\left(\frac{\Pr\{e_j=+1 \mid r_j\}}{\Pr\{e_j=-1 \mid r_j\}}\right)$ for BPSK in the AWGN channel with variance $\text{Var}(y_i)=2\sigma^2$. The LLR can be derived as the following equation:

$$\Lambda(y_j) = \left(\frac{2}{\sigma^2}\right) r_j \tag{2.4}$$

The reliability of the decision codeword is also major so the LLR of each element d_j of decision Dc is given by the relation:

$$\Lambda(d_j) = \ln\left(\frac{\Pr\{e_j=+1 \mid R\}}{\Pr\{e_j=-1 \mid R\}}\right) = \frac{\sum_{C^i \in list_j^{+1}} \Pr\{E=C^i \mid R\}}{\sum_{C^i \in list_j^{-1}} \Pr\{E=C^i \mid R\}} \tag{2.5}$$

$$\text{and} \quad P\left\{R \mid E = C^i\right\} = \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n \exp\left(-\frac{\left|R - C^i\right|}{2\sigma^2}\right) \tag{2.6}$$

In (2.5), the $list_j^{+1}$ is a set for all $c_j^i = +1$, and the $list_j^{-1}$ is a set for all $c_j^i = -1$.

Recall Bayes' rule:

$$\Pr\left\{E = C^i \mid R\right\} = \Pr\left\{R \mid E = C^i\right\} \frac{\Pr\left\{E = C^i\right\}}{\Pr\left\{R\right\}} \tag{2.7}$$

(2.6) and (2.7) can be substituted to (2.5). The LLR can be expressed as

$$\Lambda\left(d_j\right) = \frac{1}{2\sigma^2}\left(\left|R - C_j^{-1}\right|^2 - \left|R - C_j^{+1}\right|^2\right) + \ln\left\{\frac{\sum_{C^i \in list_j^{+1}} \exp\left(\frac{\left|R - C_j^{+1}\right|^2 - \left|R - C^i\right|^2}{2\sigma^2}\right)}{\sum_{C^i \in list_j^{-1}} \exp\left(\frac{\left|R - C_j^{-1}\right|^2 - \left|R - C^i\right|^2}{2\sigma^2}\right)}\right\}$$

$$= \frac{1}{2\sigma^2}\left(\left|R - C_j^{-1}\right|^2 - \left|R - C_j^{+1}\right|^2\right) + \ln\left\{\frac{\sum_{C^i \in list_j^{+1}} A_i}{\sum_{C^i \in list_j^{-1}} B_i}\right\} \tag{2.8}$$

In (2.8), the $\sum_{C^i \in list_j^{+1}} A_i \approx \sum_{C^i \in list_j^{-1}} B_i$, $\sigma \to 0$ *for high SNR* will make the term of nature log which can be ignored. We obtain an approximation of the LLR with respect to the decision d_j equal to

$$\Lambda'\left(d_j\right) = \frac{1}{2\sigma^2}\left(\left|R - C_j^{-1}\right|^2 - \left|R - C_j^{+1}\right|^2\right) \tag{2.9}$$

By utilizing equation (2.3) the relation can be expressed as

$$\Lambda'\left(d_j\right) = \frac{2}{\sigma^2}\left(r_j + \sum_{l=1, l\neq j}^{n} r_l c_{l(j)}^{+1} p_l\right),\ p_l = \begin{cases} 0, & for \ C_{l(j)}^{+1} = C_{l(j)}^{-1} \\ 1, & C_{l(j)}^{+1} \neq C_{l(j)}^{-1} \end{cases} \tag{2.10}$$

$\sigma^2 = 2$ is used to normalize intrinsic information (2.4) to get $\Lambda\left(y_j\right) = \mathrm{r_j}$, and r_j'

$w_j = \sum_{l=1, l\neq j}^{n} r_l c_{l(j)}^{+1} p_l$ and r'_j is defined as $\Lambda'\left(d_j\right)$

So the relation of equation (2.10) may be expressed as:

$$r_j' = r_j + w_j \tag{2.11}$$

The term w_j is a corrected term like extrinsic information, and the term r_j is a soft-input data like intrinsic information. The D_c is defined as the decision codeword of candidate list, and $C_p(j)$is the competing code word with respect to the D_c. The definition means that the *j*-th position of $C_p(j)$ is the inverse *j*-th position of D_c. (2.9) is also normalized by $\sigma^2 = 2$ and can be expressed as

$$r_j' = \left(\frac{\left|R - C_p(j)\right|^2 - \left|R - D_c\right|^2}{4} \right) d_j \tag{2.12}$$

where dj is the hard-decision of D_c mapping to -1 or +1. The extrinsic information can be obtained by

$$w_j = r_j' - r_j \tag{2.13}$$

Because BTC is a hyper-dimension array, (2.13) can be rewritten as

$$[W] = [R'] - [R] \tag{2.13}$$

If we can't find a competing code word C_p with respect to decision code word D_c, there must be a tradeoff approach to compute r_j' that is:

$$r_j' = \beta \times d_j \ \ with \ \ \beta \geq 0 \tag{2.14}$$

Now, a new parameter "m" is defined as a run number, and m is increased as m=m+1 after decoding one dimension of BTC array. In (2.14), the experimental value of β is gradually increasing to 1.0 in every runs. β is shown as flows [9]:

$$\beta(m) = [0.2, \quad 0.4, \quad 0.6, \quad 0.8, \quad 1.0, \quad 1.0, \quad 1.0, \quad 1.0] \tag{2.15}$$

Finally, iterative decoding algorithm of the BTC is shown as the following run equations :

$$[R(m)] = [R] + \alpha(\mathrm{m}) \times [W(\mathrm{m})] \tag{2.16}$$

$$[W(\mathrm{m+1})] = [R'(\mathrm{m})] - [R(\mathrm{m})] \tag{2.17}$$

$[R]$ is intrinsic information. $[W(\mathrm{m})]$ is extrinsic information in m-th run and is zero in first run, and $[R'(\mathrm{m})]$ is output soft value in m-th run. $[R(\mathrm{m})]$ is decoded by SISO, and where $\alpha(\mathrm{m})$ is a scaling factor for that, and the standard deviation of values is different between array $[R]$ and array $[W(\mathrm{m})]$. $\alpha(m)$ is also used to reduce the effect of the extrinsic information in the first run when the BER if relatively high [9]. The evolution of α with the decoding m-th run is

$$\alpha(m) = [0.0,\ \ 0.2,\ \ 0.3,\ \ 0.5,\ \ 0.7,\ \ 0.9,\ \ 1.0,\ \ 1.0] \tag{2.18}$$

The α is also a parameter of the experimental result. The completed BTC iterative decoding block diagram is illustrated in Figure 2.5. One iteration means that all dimension computation has been finished, for example, 2 dimension (row and column space) BTC. m is increased 2 after one iteration.

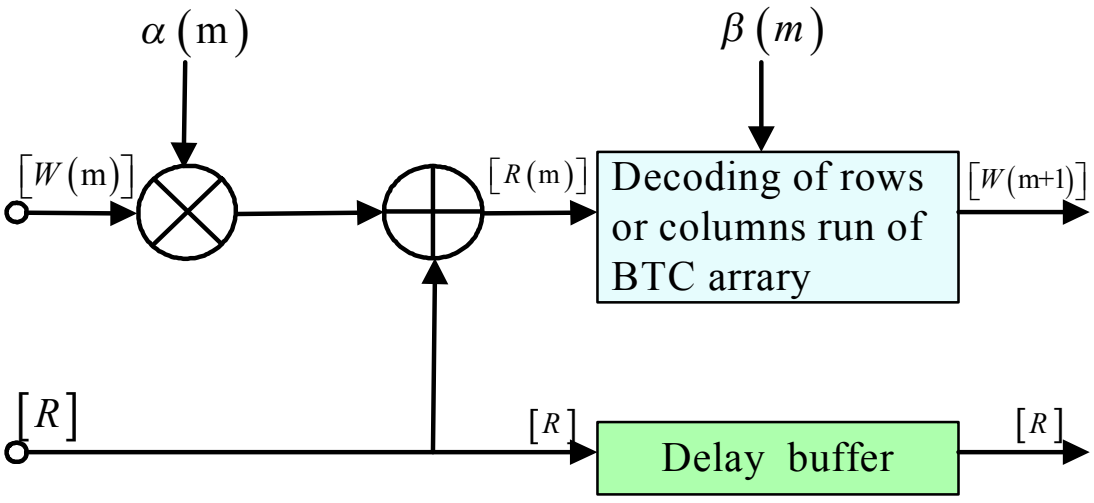


Figure 2.5: Structure of BTC

2.3 802.16e (WiMAX) BTC

Worldwide interoperability for microwave access (WiMAX) is a standards-based communication system (i.e. IEEE 802.16 [4][10]) and is intended for wireless "metropolitan area networks". WiMAX is expected to implement multimedia applications with wireless connections and can provide broadband wireless access up to 30 miles (50Km) for a fixed station, and 3- 10 miles (5-15km) for a mobile station..

In this section, WiMAX specification is going to be introduced. Because it's forward error correction (FEC) application includes BTC. This application as an example is implemented for our BTC study.

Some modifications are described for a linear code in this segment. A linear code, C (n-1, k) can be extended for increasing one parity bit, and C' (n, k) is produced. The extended bit is derived by the XOR of the k information bits and the (n-k-1) parity bits of the code C. This procedure is called extended code.

The "shortened" cyclic code is going to introduce. For example, A two-dimensional BTC, $C_b(n_b, k_b)$ is composed of component codes, $C_1(n_1, n_1)$ and $C_2(n_2, k_2)$, as shown on Figure 2.1. The S_1 rows and S_2 columns' bits of message symbols of the BTC are initialed zero, and the transmitted data is inserted in the remaining of the message symbols. Then, the message array is encoded. Finally, An BTC C_b is transmitted except those S_1 rows and S_2 columns. The chart is illustrated in Figure 2.6.

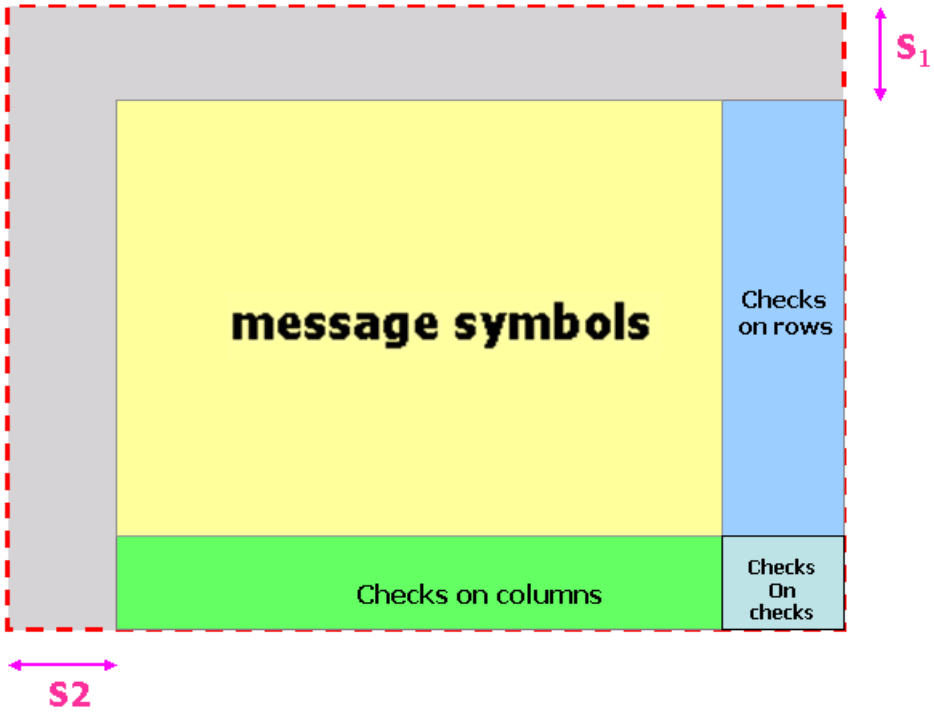


Figure 2.6: The sketch of shorten code

Now, we are going to introduce WiMAX BTC specification. The hamming code generator polynomials are listed in Table.2.1.

N	K	Generator polynomial
7	4	X3+X1+1
15	11	X4+X1+1
31	26	X5+X2+1
63	57	X6+X+1

Table 2.1: The generator polynomial of BTC

The BTC component codes are listed in Table.2.2.

Component codes (n, k)	Code type
(64, 57), (32, 26), (16, 11), (8, 4)	Extended hamming code
(64, 63), (32, 31), (16, 15), (8, 7)	Parity check code

Table 2.2: BTC component code of WiMAX

To match an arbitrary requires transmission packet size. BTC may be shortened by removing symbols from the BTC array until the appropriate size is reached. The following two steps are involved in the specification for shortening BTC:

Step1: I_1 rows and I_2 columns are removed from the two-dimensional BTC array. This is equivalent to shortening the BTC codes.

Step 2: B individual bits are removed from the first row of the BTC array starting with the LSB.

The derived block length of the shortened code is $(k_1 - I_1)(k_2 - I_2) - B$ as illustrated in Figure 2.7. The corresponding information length is given as $(n_1 - I_1)(n_2 - I_2) - B$. Consequently, the code rate is given by the following equation :

$$R = \frac{(k_1 - I_1)(k_2 - I_2) - B}{(n_1 - I_1)(n_2 - I_2) - B} \tag{2.19}$$

Our study is focused on extended hamming code (64, 57) to be our BTC component code, and we want the throughput about 30Mbps on clock rate 200MHz.

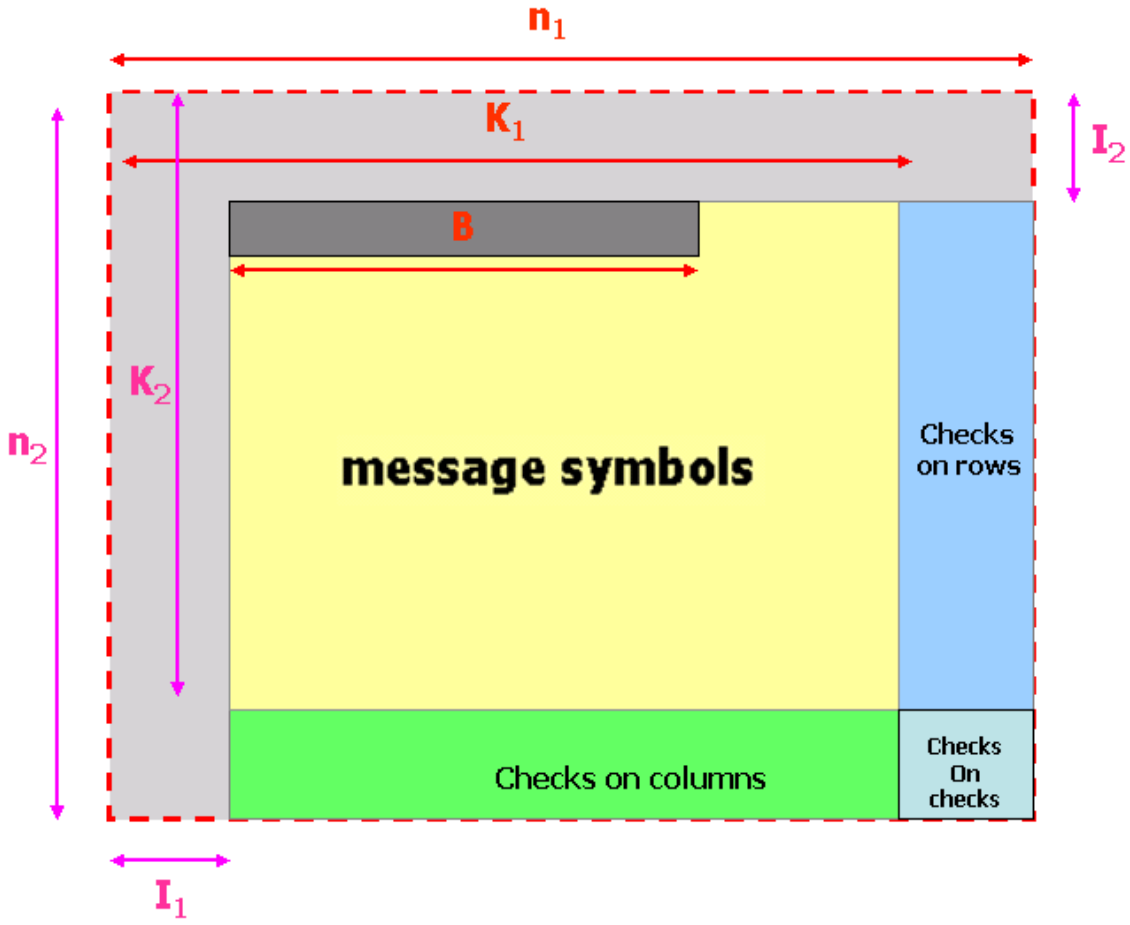


Figure 2.7: The BTC shorten code for WiMAX

Chapter 3

Proposed Geometry-like Algorithm

In this chapter, the proposed geometry-like algorithm is introduced. The algorithm is for the hardware implementation without experimental parameters. The algorithm level program is implemented by C language.

Some foundations of geometry are described with some detailed formulation before our induction. A BPSK is defined to map (-p, +p). The p is defined as p=1 in our study. The one dimension coordinate diagram is illustrated in Figure 3.1, and r is a received channel value at receiver. The variable e is defined as the value of transmission. The a posteriori probability is inducted as follows:

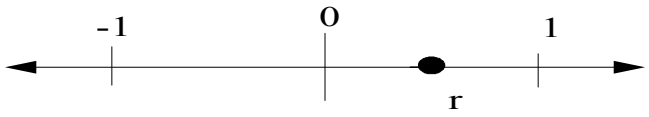


Figure 3.1: Analysis of the Probability for BPSK

$$\Pr\{e=\pm 1 \mid r\} = P\{r \mid e\} \cdot \frac{\Pr\{e=\pm 1\}}{\Pr\{r\}} = P\{r \mid e\} \cdot k_r$$

$$\text{where } P\{ r \mid e=\pm 1\} = \left[\frac{1}{\sqrt{2\pi}}\right]^n \exp\left[-\frac{|r \pm 1|^2}{2\sigma^2}\right]$$

$$\Lambda\left(y_j\right)=\ln\left[\frac{\Pr\{e=+1\mid r_j\}}{\Pr\{e=-1\mid r_j\}}\right]=\left(\frac{2}{\sigma^2}\right)r_j \tag{3.1}$$

In Figure 3.2, the v is perpendicular to the hemline of the triangular ΔABC. Pythagoras' Theorem is used to drive the equation (3.2):

$$\begin{aligned}
&S_1^{\,2}-S_3^{\,2}=V^2\\
&S_2^{\,2}-S_4^{\,2}=V^2\\
&S_1^{\,2}-S_2^{\,2}=(S_3^{\,2}+V^2)-(S_4^{\,2}+V^2)=S_3^{\,2}-S_4^{\,2}
\end{aligned} \tag{3.2}$$

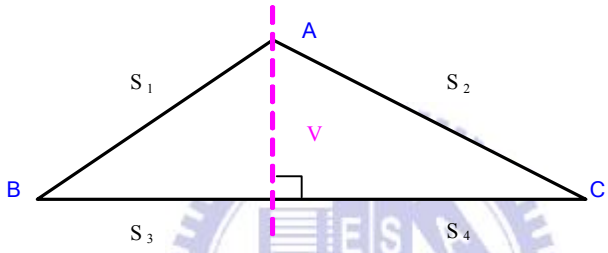


Figure 3.2: Geometric equation analysis of the equation (3.2)

The two axis coordinate diagram is illustrated in Figure 3.3. A triangular ΔABC is sketched on the diagram. The apex C is projected to r' of x-axis. $|B-A|$ is defined as the distance of $\overline{AB}$. a and b are symmetrical. Thus, r' can be found the relation as the equation (3.3):

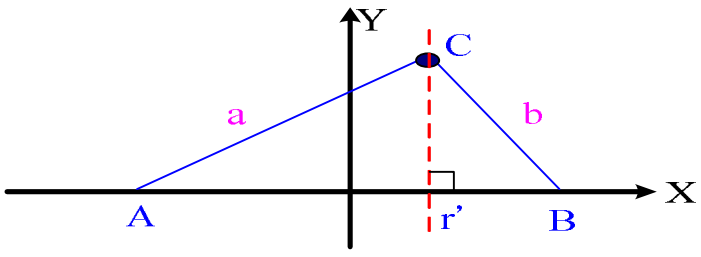


Figure 3.3: Geometric equation analysis of the equation (3.3)

$$a^2 - [\ \frac{1}{2}(|B-A|) + r'\ ]^2 = b^2 - [\frac{1}{2}(|B-A|) - r')\ ]^2$$

Let $t = \frac{1}{2}(|B-A|)$ rewrite above equation

$$a^2 - b^2 = [\ t+r'\ ]^2 - [t-r'\ ]^2 = 4\times t\times r'$$

$$a^2 - b^2 = 2(|b-a|)\, r'$$

$$r' = \frac{1}{2(|B-A|)}(\ a^2 - b^2\ ) \qquad (3.3)$$

A new algorithm is going to be inducted now. A two dimensional BTC is supposed, and there are two entries in every space (column and row) of the received data. For example, 2 coordinates are introduced to $R = \begin{bmatrix} 0.2 & 1.0 \\ 0.5 & -0.6 \end{bmatrix}$ in each space.

In row space:

$R_{row1} = (0.2,\ 1.0)$

$R_{row2} = (0.5,\ -0.6)$

In column space:

$R_{columm1} = (0.2,\ 0.5)$

$R_{columm2} = (1.0,\ -0.6)$

The row space $R_{row1} = (0.2, 1.0)$ is sketched to explain the algorithm on constellation diagram as an example. The diagram is illustrated in Figure 3.4. The constellation diagram expresses the channel values of every space. There are 4 coordinates (1, 1), (-1, 1), (1, -1), (-1, -1). These coordinates mean the former values at transmitter. The R_1 on the diagram is the result of the transmitted value adding noise and R1 is projected to R2.

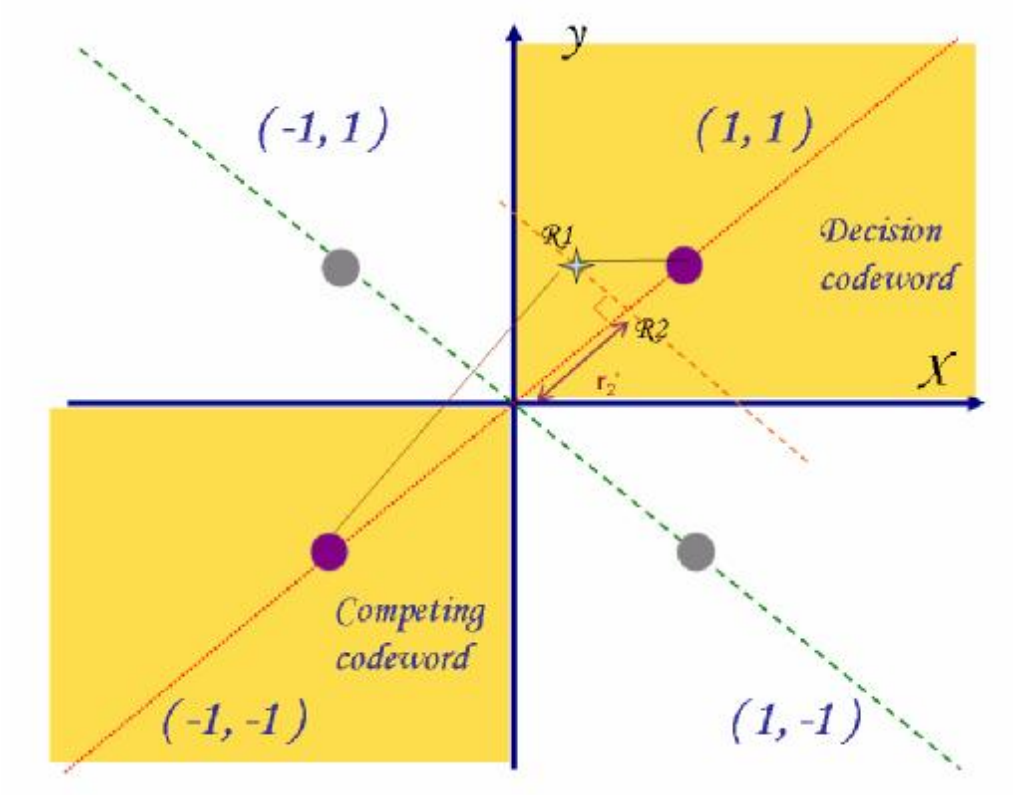


Figure 3.4: Equivalent analysis on constellation

Base on Figure 3.4, The equation (2.12) is recalled, we know:

$$r_1{}'_j = \left(\frac{\left|R_1 - C_p(j)\right|^2 - \left|R_1 - D_c\right|^2}{4} \right) d_{1j}$$

and use the concept of equation (3.2) , $r_1{}'_j$ is rewritten as the following equation:

$$\begin{aligned} r_1{}'_j &= \left(\frac{\left|R_1 - C_p(j)\right|^2 - \left|R_1 - D_c\right|^2}{4} \right) d_{1j} \\ &= \left(\frac{\left|R_2 - C_p(j)\right|^2 - \left|R_2 - D_c\right|^2}{4} \right) d_{2j} = r_2{}'_j \end{aligned} \tag{3.4}$$

The equation proves that R_1 and R_2 are equivalent, and we use the concept of equation (3.3) to get $r_2{}'_j$:

$$r_2{}'_j = \frac{1}{2\left|C_p(j) - D_c\right|}\left(\left|R_2 - C_p(j)\right|^2 - \left|R_2 - D_c\right|^2\right)$$
$$= \frac{1}{2\left|C_p(j) - D_c\right|}\left(\left|R_1 - C_p(j)\right|^2 - \left|R_1 - D_c\right|^2\right) \quad (3.5)$$

So the reliability of R_1 can be substituted by R_2. $\left|C_p(j) - D_c\right|$ is the value of the distance. The Hamming distance can be used to solve the value. The hard decision of competing codeword and decision codeword is defined as following:

$$HC_p = \left[HC_{p1} \quad HC_{p2} \quad \cdots \quad HC_{pn}\right]$$
$$HD_c = \left[HD_{c1} \quad HD_{c2} \quad \cdots \quad HD_{cn}\right]$$

Where HC_{p1} , $HD_{c1} \in 0,1$

Hamming distance is defined as $d(HC_p, HD_c) = \sum_{i=1}^{n}\left(HC_{pi} \oplus HD_{ci}\right)$ based on BPSK(-1, +1). Then,

$$\left|C_p(j) - D_c\right|^2 = \sum_{k=1}^{n}\left(C_p(j)_K - D_{c\,k}\right)^2 = \sum_{k=1}^{n}\left(2 \times \left(HC_{p_k} \oplus HD_{ck}\right)\right)^2$$

so we can express $\left|C_p(j) - D_c\right| = \sqrt{2^2 \times d(HC_p, HD_c)}$ and re-write (3.5) as:

$$r_2{}'j = \frac{1}{4 \times \sqrt{d(HC_p, HD_c)}}\left(\left|R_1 - C_p(j)\right|^2 - \left|R_1 - D_c\right|^2\right) \quad (3.6)$$

Now, the equation (2.13) is recalled and multiplied by 2, and then we can get the following equation:

$$2w_j = 2r_1{}'_j - 2r_{1j} = 2r_2{}'_j - 2r_{2j}$$

$\hat{w}_j$ is defined as the extrinsic information of our algorithm and the definition is shown as the following equation:

$$\hat{w}_j = 2w_j - r_m \quad (3.7)$$

There are two special situations in our algorithm. The situations are discussed as

follows:

<1> The no competing codeword situation has to be considered, and our approximation of the soft output is $r'_j = d_j \in -1, +1$. It means that the soft value of this bit is reliable. There are no noise so the value can be mapped to (+1, -1). Because we also want to gradually move the information to (+1, -1), the compensation of information is only half for the next run. Therefore, the following equations can be obtained:

$$\widehat{W} = \frac{1}{2}(d_j - R) .$$

<2> Another consideration is the parallel condition situation. This is a special case. If the competing codeword and the decision codeword are connected to a line parallel with the axis of constellation, the extrinsic information $\widehat{W}$ keeps old value. If the extrinsic information is changed on this condition, another value of axis would be wrong with respect to decoding axis.

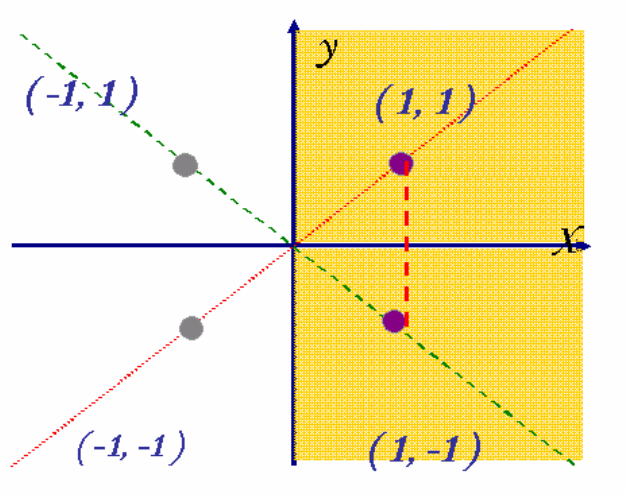


Figure 3.5: Parallel condition chart

A trick of making more candidates is used. An example is explained as follows:

$R_a = \langle a_1,\ a_2,\ a_3,\ a_4,\ a_5 \rangle = \langle -1.2,\ 0.7,\ 0.5,\ 0.8,\ -0.3 \rangle$ is assumed. Then, R_a soft value is decoded to $R_{a'} = \langle -1.2,\ 0.7,\ 0.5,\ 0.8,\ -0.4 \rangle$. This assumption treats a special problem for low reliable priority. We can see the a_5 entry whose reliability of $R_{a'}$ is still the lowest after decoding. The choice of low reliable bits is the same as that in the next decoding run. The other possible entry can not be tested so $R_{a'}$ is adapted as following equation:

$$R_{b'} = \frac{1}{2}\langle -1.2,\ 0.7,\ 0.5,\ 0.8,\ -0.4\times 2 \rangle = \langle -0.6,\ 0.35,\ 0.25,\ 0.4,\ -0.4 \rangle$$

Other bit priority is decreased except for entry a_5, and the new candidates can't be combed out for next decoding run. The trick is used in our algorithm.

Finally, our new algorithm is summarized as following pseudo code:

step 1:

$$[R(m)] = \frac{1}{2}\ ([R] + [\widehat{W}(m)])$$

step 2:

if(NO competing codeword)

$$[\widehat{W}(m+1)] = \frac{1}{2}(\ d_j\ - [R\ (m)]\ )$$

else

if(parallel condition is not tenable)

$$[\widehat{W}(m+1)] = 2[R'(m)] - [R\ (m)\ ]$$

else

$$[\widehat{W}(m+1)] = \widehat{W}(m)$$

where $R' = \frac{1}{4\times\sqrt{d(HC_p, HD_c)}}\left(\left|R\ - C_p(j)\right|^2 - \left|R\ - D_c\right|^2\right) \times d_j$

Chapter 4

Hardware Architecture Design

In this chapter, the hardware design and implementation are shown. Section 4.1 describes BTC encoder. The BTC decoder design is described in section 4.2. The parallelism issue of the circuit architecture is discussed in section 4.3.

4.1 Encoder Design

The functional block diagram is illustrated in Figure 4.1. Because the hardware design is considered as the trade between memory gate count and throughput, we use a 64*64 bit memory, one BCH encoder, and parity buffer is used to store parity after encoding. There are 64 component codewords in a BTC array to use one BCH encoder so a scheduler is designed to control total cycles within the limit. In our design, the throughput is about 0.326 bit/cycle.

The BCH encoder of our design is a linear feedback shift register (LFSR)[12]. After the BCH codeword is encoded, all the codeword bits of BCH are xor-ed together. The result of xor is the extension bit.

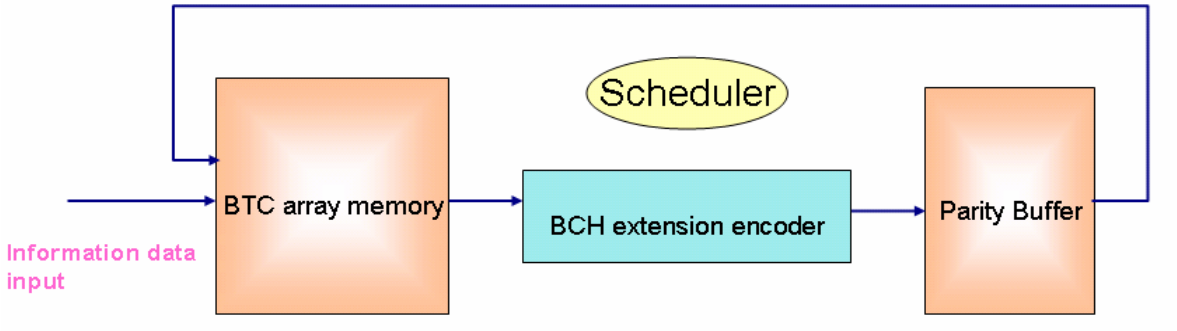


Figure 4.1: The block diagram of encoder

4.2 Decoder Design

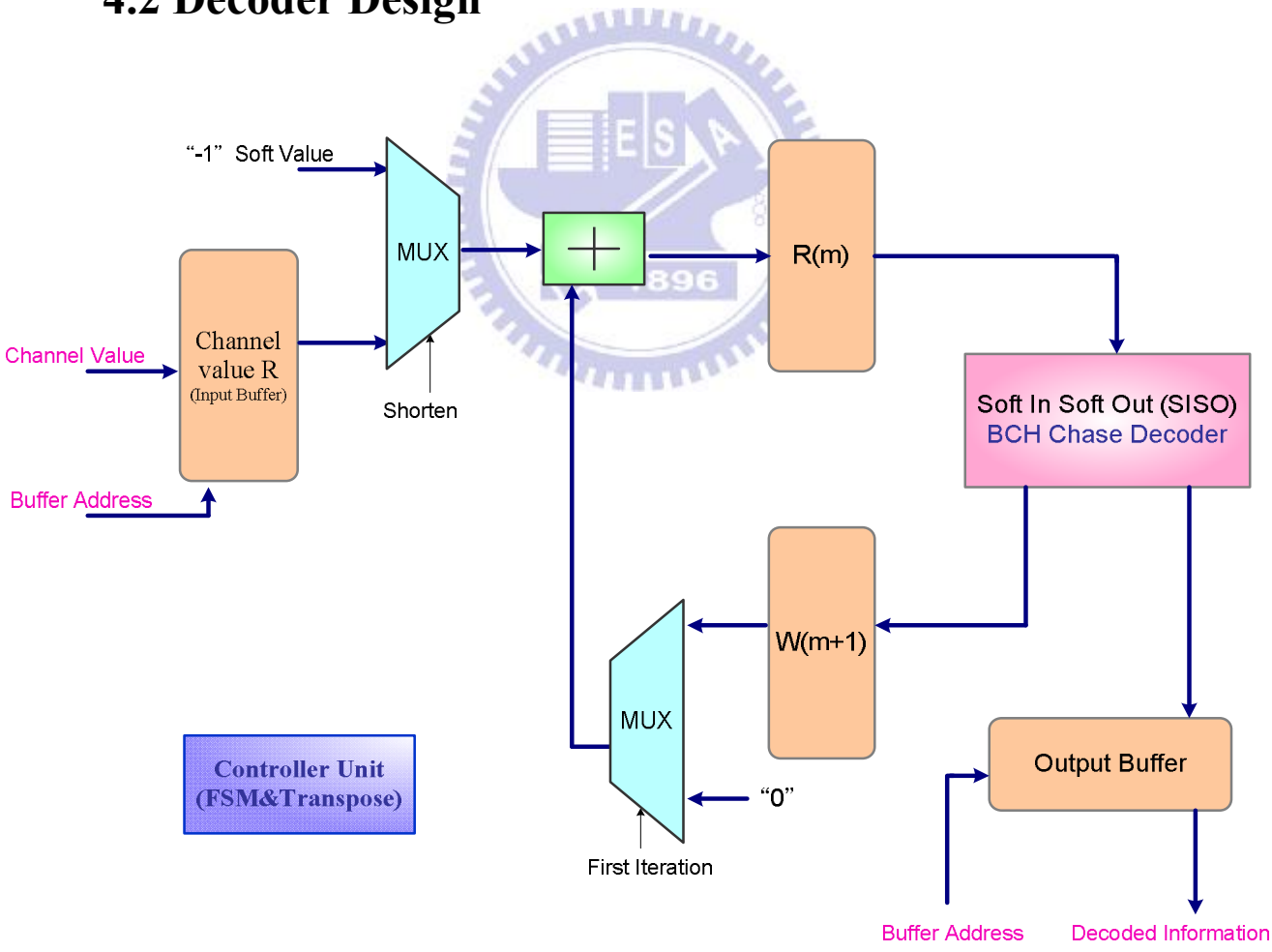


Figure 4.2: The block diagram of decoder

The blocks of our decoder design are shown in Figure 4.2. the channel value sequence of our decoder is got from the de-modulation, and is saved in input buffer R. R plus extrinsic information wm is stored in Rm. The Rm is used to compute the next extrinsic W_{m+1} by the SISO. The decoded result of hard decision is outputted to output the buffer. The buffer can be read by the next stage of system circuit. There are four memories about the gate count for the most part. They are channel value R, Rm, Wm, and the output buffer respectively. These memories are hard to reduce. The SISO block is our BTC decoder core. The main algorithm is used in this block. There are parallel circuit and gate count issue of the consideration in our design so SISO circuit size is our focus. A special transposing array controller circuit is also designed in controller unit except for the finite state machine controller.

4.2.1 Algebraic Decoding

Let a codeword be $v(x) = v_0 + v_1 x^1 + v_2 x^2 + \ldots\ldots\ldots + v_{n-1} x^{n-1}$. The AWGN channel is defined as $Noise(x)$. The received codeword is shown as the following equations:

$$R(x) = V(x) + Noise(x)$$

$R(x)$ can be rewritten as:

$$R(x) = q_i(x) \bullet \phi_i(x) + b_i(x)$$

Where, $\phi_i(x)$ is the minimal polynomial.

and then let

$$S_i = R(\alpha^i) = b_i(\alpha^i) \tag{4.1}$$

Our design specification is BCH(63, 57), and the code is a perfect code (t=1). One error of the received codeword can be corrected, and the codeword only need the

syndrome, $S_1 = b(\alpha)$ to be the error location [12]. The result of the perfect code decoding must be a valid codeword so there are no out of location situation and checking valid codeword, but it may not be our transmitted codeword at transmitter. In Summary, because the error location is S_1, we can design a simply algebraic decoder circuit without complexity algorithm, e.g. BM algorithm [13]. The error location can be decoded by LFSR [12].

4.2.2 The SISO Architecture

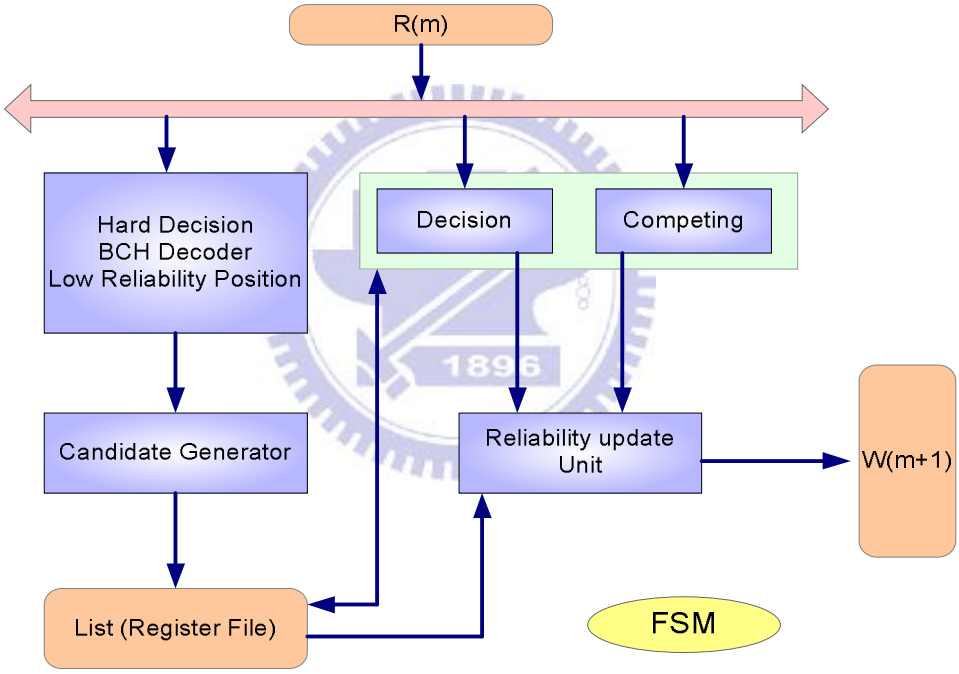


Figure 4.3: The block diagram of SISO

The SISO consists of Algebraic decoder (BCH decoder), a candidate generator, a list memory, a decision and competing circuit, and a algorithm reliability update unit. It is illustrated in Figure 4.3. The computation time of our design spends about 550 cycles for finishing one decoding run. Our design in every block is interpreted as the follows:

A. Hard decision, BCH decoder, insertion sorting for low reliability position

After Rm is ready, the hard decision is determined. The hard decision result is used to compute syndrome for error location by LFSR. The insertion sorting approach is proposed to sort low reliable bits of the received codeword. The truth table of sorting is listed on Table.4.1. The insertion sorting circuit is designed according to Table.4.1. The flags (X_1, X_2, X_3) are produced by the three comparer (Cmp). The multiplexers (MUX) are controlled by flags to select data to store the value in register (R1~R4). The insertion sorting circuit is shown in Figure 4.4.

Flag	X1	X2	x3
R4<R1<R2<R3	1	1	1
R1<R4<R2<R3	0	1	1
R1<R2<R4<R3	0	0	1
R1<R2<R3<R4	0	0	0

Table 4.1: The truth table of insertion sorting flag

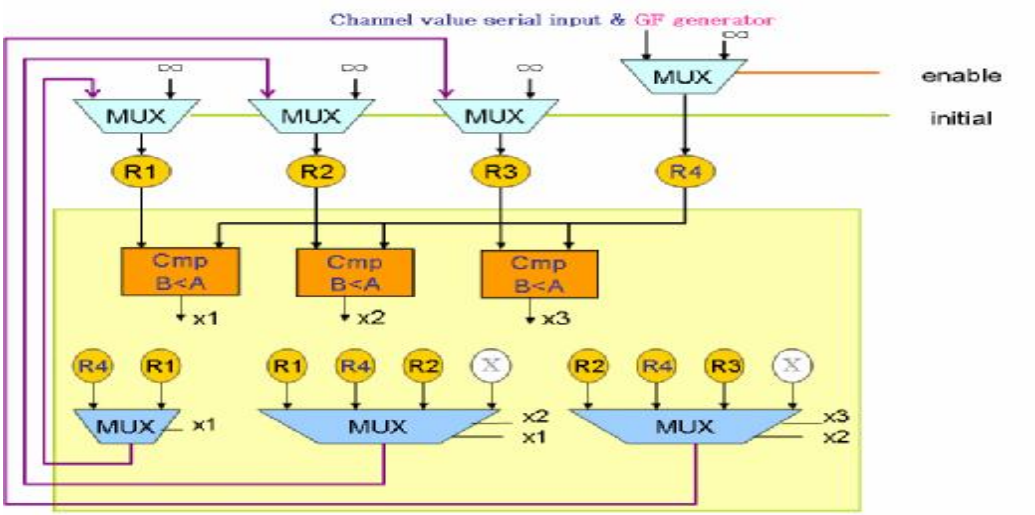


Figure 4.4: The circuit of insertion sorting

B. Candidate generator

When the syndrome and low reliable position have been computed from the last stage, the counter controls the multiplexers to generate all permutation test patterns. The syndrome xor all the combinational results of multiplexes to produce new syndrome. The location table is looked up the table by new syndrome and map which bit is corrected. Finally, after all of the output of location table is xor-ed , then candidates are generated.

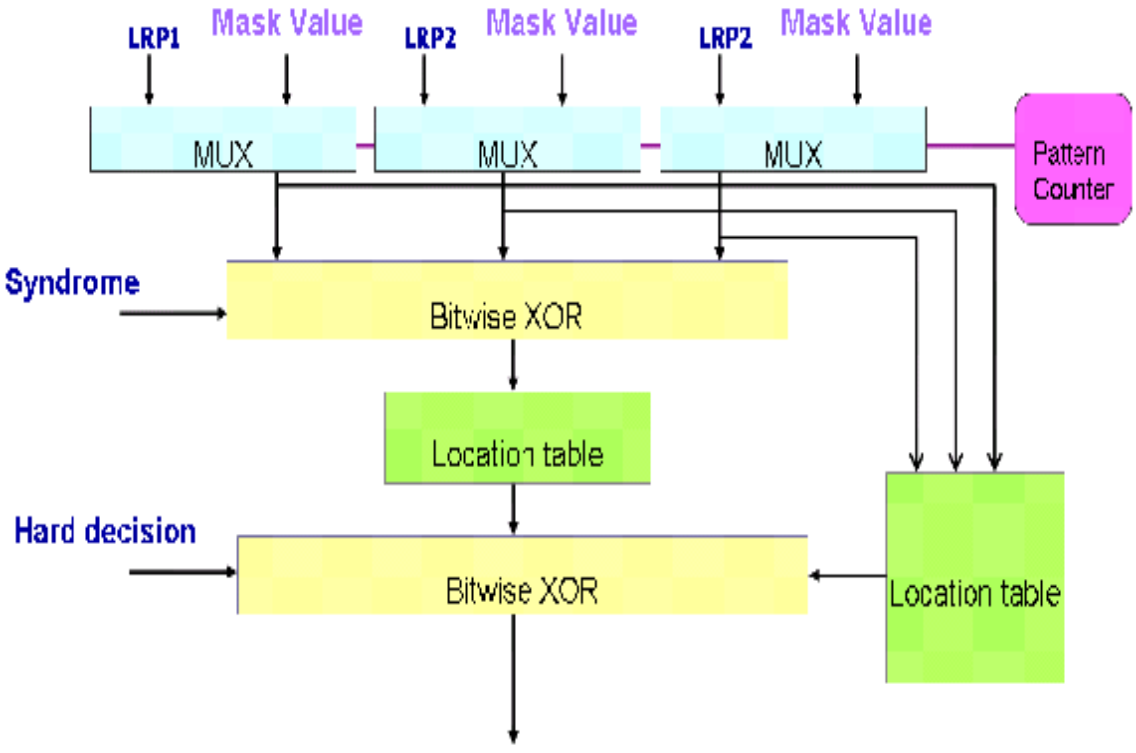


Figure 4.5: The circuit of candidate-generator

C. List (register file)

Because there are only 8 candidates in the list of our design, the list

memory is implemented by using a register file and is not memories. The register file is multi-port register file that provide parallel data outputting for the next stage.

D. Decision & competing unit

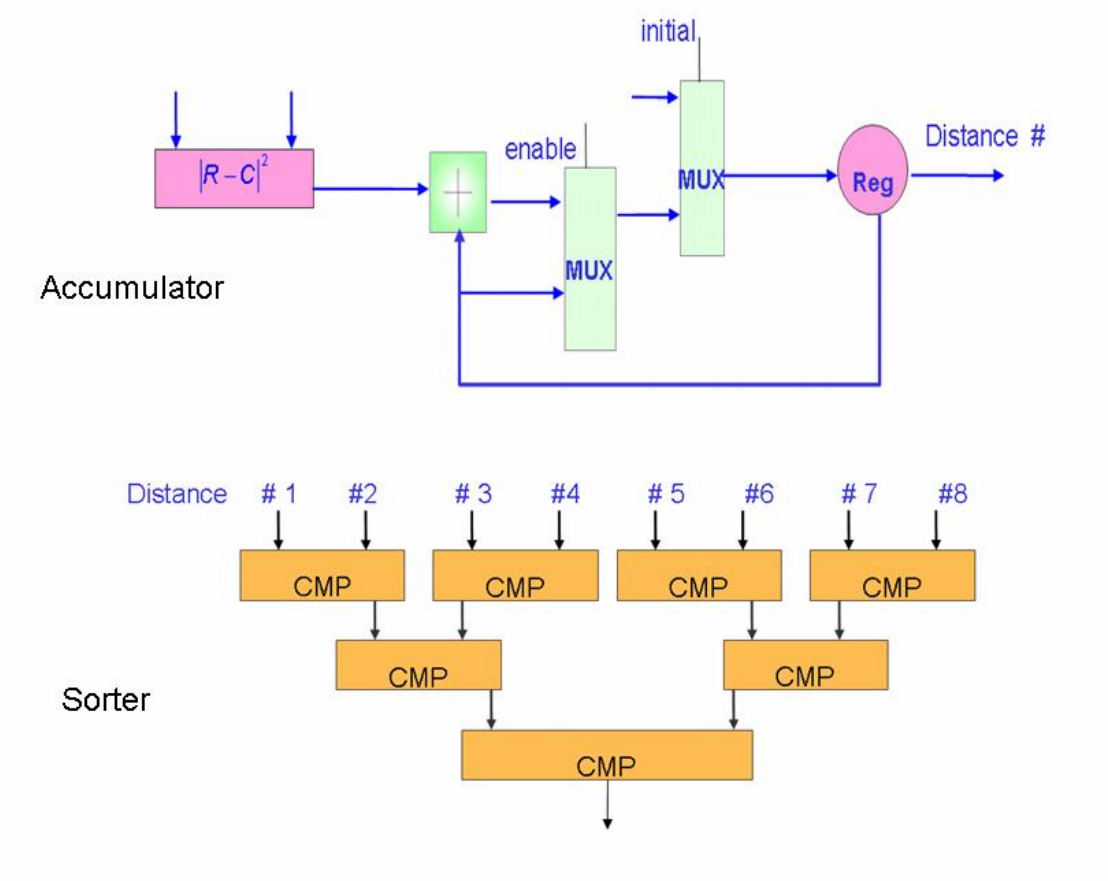


Figure 4.6: The circuit of decision&competing unit

There are 8 candidates that would be compared out the least minimal distance. The design of the accumulators is illustrated in Figure 4.6. The 8 accumulators are used to compute all distance between the 8 candidates and the channel value, and then the soft distance of the candidates are sorted by 7 comparers. After sorting, the minimal distance of the decision codeword and

the competing codeword can be outputted to next stage.

E. Reliability update unit

The reliability update unit of the circuit is shown in Figure 4.7. We think over the computation speed of the circuit so the 6 levels adder (Figure 4.8) are implemented to compute hamming distance. The look up table is used to find out the value of complexity square roots computation in our design. The rest of reliability update unit is also based on our decoding algorithm and is shown in Fig4.7.

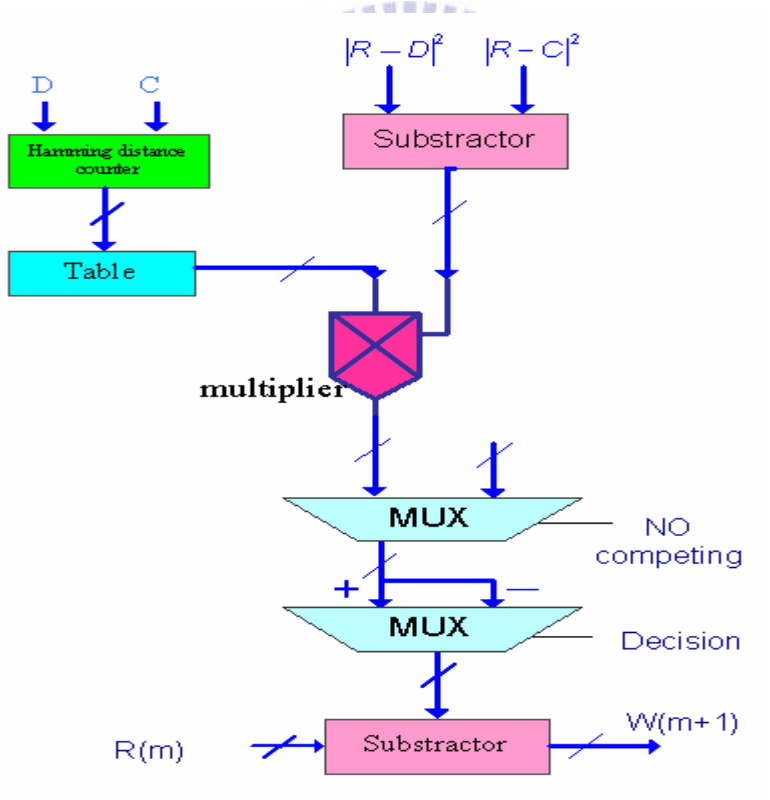


Figure 4.7: The circuit of the reliability update unit

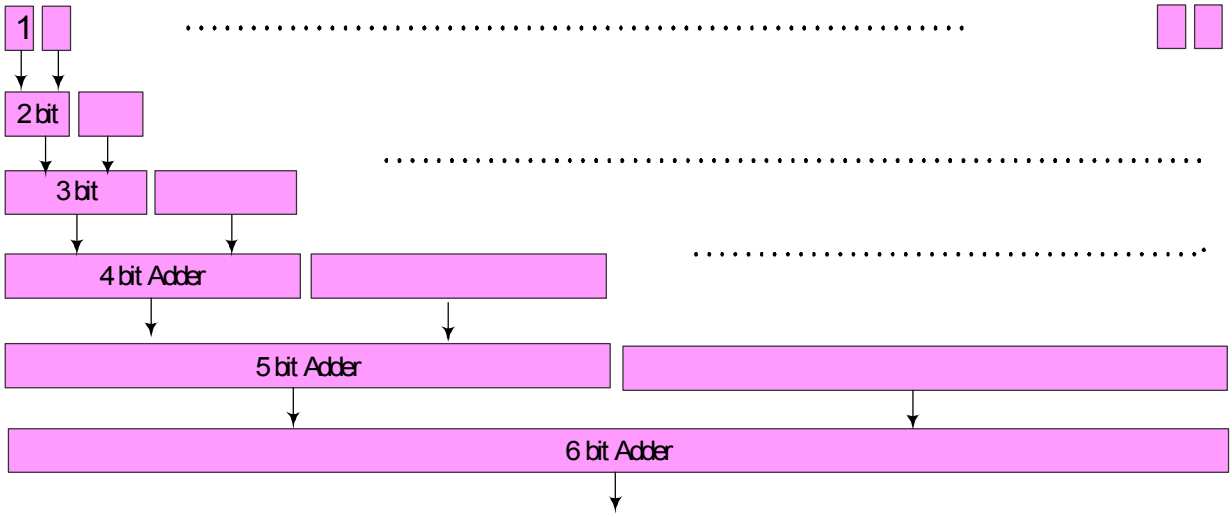


Figure 4.8: Six-level adder circuit

4.3 Parallel Architecture

4.3.1 Parallel Architecture Planning

Because of the requirement of WiMAX specification, the throughput is an important issue. Therefore, the throughput is excepted to arrive at least 30Mbit/sec. For this reason, the parallel architecture of the decoder circuit is projected for arriving throughput constraint. The BTC (64, 57) includes 64 sub-code words (component code) in every dimension so the all sub-code words can be independently processed by the different SISOs. The quantity of usability of SISO is from 1 to 64. If more SISOs are used in decoder, the BTC decoder would be faster, but the gate count is also increased. Therefore, we consequently consider the trade off. The memory is divided into parts when the number of SISO is decided by our analysis. Every memory partition is defined as a memory bank, and we generally called these

partitions multi bank array (MBA). Each MBA has 4096 entries for 64×64 soft data. Our analysis formulation is based on Figure 4.2. and illustrated as the following equation (4.2):

$$Total\ cycle\ = 8run \times (S \times \frac{64}{N}\ +\ \frac{4096}{N}) + input\ latency \qquad (4.2)$$

Where "S" (cycles) is the numbers of cycles for SISO computation, "N" (SISOs) is parallel numbers, "input latency" means the total cycle time of the input buffer to be fully written, and "8run" means every SISO excute 8 times for iteration decoding algorithm.

Now, the BTC(64, 57) is estimated as the circuit clock rate at about 200MHz so we calculate the following equation:

$$\because\ \text{clk rate} \times (\frac{\text{total bits}}{\text{total cycles}}) = \text{throughput}$$

$$\therefore\ \frac{\text{total bits}}{\text{total cycles}} = 0.15 bit/cycle$$

The BTC total information bits has been known, and that is $57^2 = 3249\ bits$.

The above substitute to (4.2):

$$\therefore \frac{3249}{0.15} = 8 \times (550 \times \frac{64}{N}\ +\ \frac{4096}{N}) + \frac{4096}{N}$$

$$\therefore\ N = \frac{(8 \times (39296) + 4096) \times 0.15}{3249}\ = 14.7 \cong 16$$

$N = 16 = 2^4$ is chosen for the parallel number estimation of formulation because power of "2" number is easily implemented in hardware. Now, the circuit parallel number is N=16 so the MBA has 16 banks, and every bank has 256 entries. The MBAs will be co-operating with proposed parallel architecture in the planning. The chart is shown as Figure 4.9.

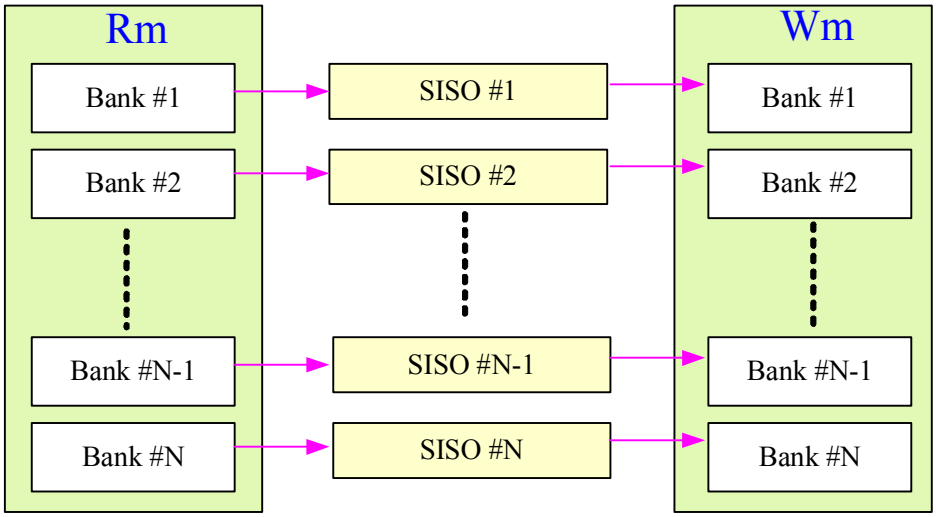


Figure 4.9: The block diagram of parallel architecture

4.3.2 Parallel Multi-Bank-Array Structure

There is a bottleneck of the MBA accessing in parallel architecture. The two ports MBAs are controlled together to move data with each other so the parallel architecture would have read/write hazard in the architecture of the MBAs. A buffer can be added to solve the hazard, but the memory gate count is also increased. Therefore, an algorithm of MBA accessing is proposed to overcome the hazard.

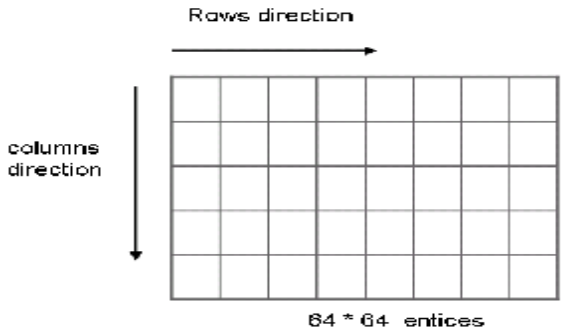


Figure 4.10: The overview of a MBA

The row and column space of the BTC decoding use the common SISOs in different runs. Our memory usability is only 3 (64*64 entries) MBAs, i.e. R-MBA, Wm-MBA, and Rm-MBA. A simple MBA chart is illustrated in Figure 4.10. and the memories accessing scheduling of the iterative decoding algorithm is listed as Table.4.2. “Transpose” means reading or writing the MBA and the addressing of the MBAs is not in order. “Normal” means reading or writing the MBA and the addressing of the MBAs is in order for each bank.

An assumption of simple example is illustrated in Figure 4.11. MBAs (Wm-MBA and Rm-MBA) are partition to 4 banks. Suppose that Wm is normal, and Rm-MBA is transpose. In first run, the bank#1 of the Wm is read out to Rm, but the four write ports of Rm-MBA have to wait bank#1 read port in order. other read ports of Wm-MBA are idle except the bank#1 of Wm-MBA. However, the resource of memory is wasted. Thus, we want all port working at the same time, and there is no accessing hazard without an additional buffer for the gate count issue.

Run number	R	W_m	$R_{m+1}=R+W_m$
0 (for row)	Normal	0	normal
1 (for colunm)	Normal	Normal	Transpose
2 (for row)	Normal	Transpose	Normal
3 (for colunm)	Normal	Normal	Transpose
4 (for row)	Normal	Transpose	Normal
5 (for colunm)	Normal	Normal	Transpose
6 (for row)	Normal	Transpose	Normal
7 (for colunm)	Normal	Normal	Transpose

Table 4.2: Algorithm flow table

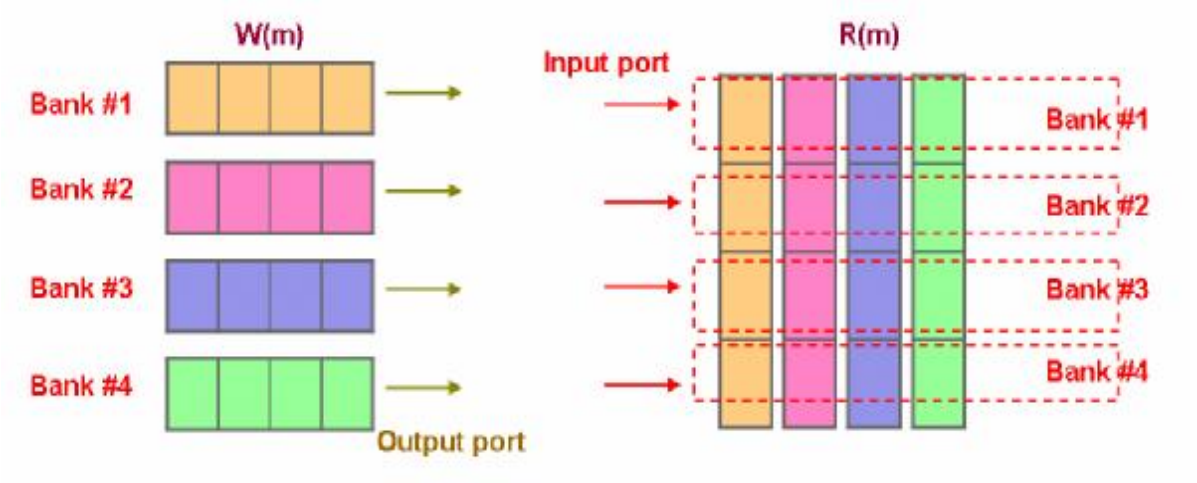


Figure 4.11: The MBA transposing

A bottleneck is very clearly shown in above example. If we want to transpose a memory, and there is no one idle in all ports of all MBAs, then some data will be delayed or over written. Due to this reason, a MBA algorithm is proposed to overcome read/write problem.

Our proposed algorithm is a special memory bank scheduling. MBA is partitioned into several slices. Every first address of MBA is started by different slices. It is shown in Figure 4.12. The number in each slot of Figure 4.12 stands for the corresponding time index. All ports can be worked at the same time and there are no data buffers. The feature of algorithm is that we reapportion the timing and the address of read/write for scheduling accessing. Based on the idea, our pseudo code of the algorithm is going to be introduced.

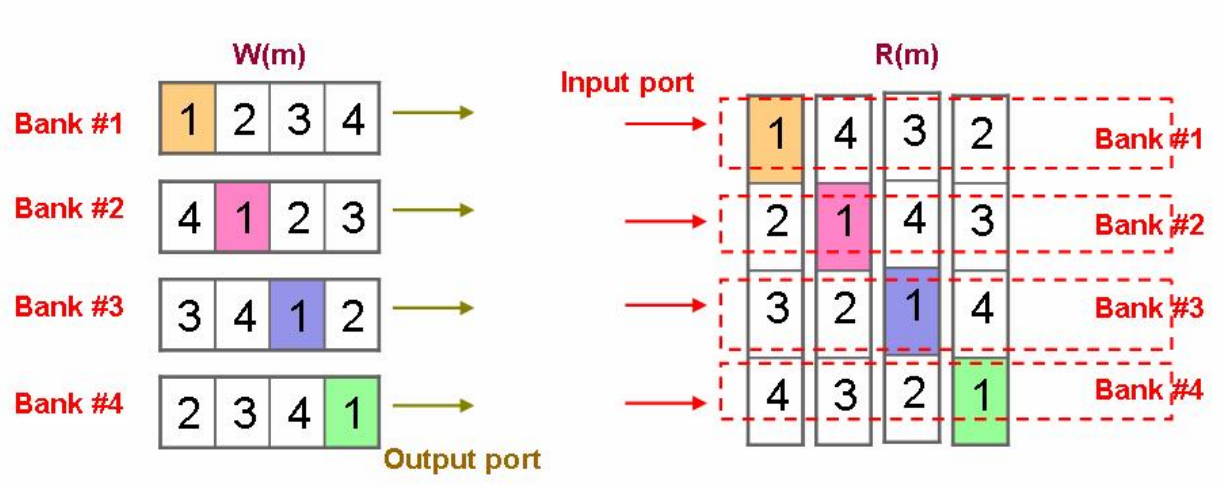


Figure 4.12: MBA scheduling

The Figure 4.13 is used to define variables for our pseudo code of proposed algorithm. The slice #s and bank #b are space variables for transposing blocks. The index i and index j are time variables for transposing elements. i and j indicate which address is read or written at every cycle for slice transposing. s and b control the addresses of the ports to avoid overlapping each other.

Our proposed MBA accessing algorithm can be used on the consideration of accessing hazard and resource issue, and the pseudo code is shown as follows:

for (s = 0 : 15) //slice

for(b = 0 : 15) //bank number

for(i = 0 : 3) // partition index

for(j= 0: 3)// partition index

if((run number mod 2) ==0)// 1 iteration=2 runs (row+colunm)

$$R(m)_b[i, j+4\times((s+b) \bmod 16)] = R_b[i, j+4\times((s+b) \bmod 16)] + W(m)_{(s+b)\bmod 16}[j, i+4b]$$

else

$$R(m)_b[i, j+4\times((s+b) \bmod 16)] = R_{(s+b)\bmod 16}[j, i+4b] + W(m)_{(s+b)\bmod 16}[j, i+4b]$$

Where “S” is slice number, “b” is bank number, i and j are indexes in the partition for every slices.

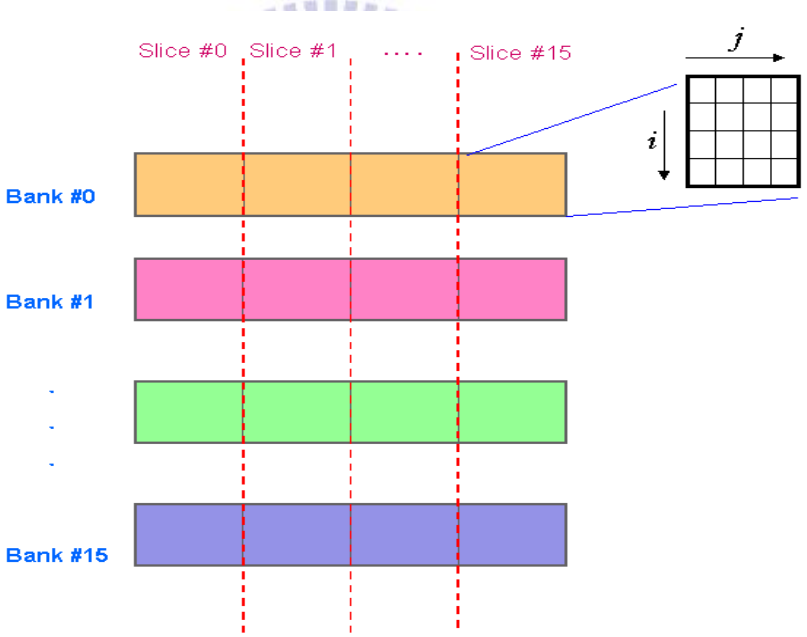


Figure 4.13: The chart of MBA algorithm

Chapter 5

Simulation Platform and Implementation Result

5.1 Algorithm Level Simulation

We take WiMAX specification to be our design example, BTC (64, 57) and Chase 2 algorithm is chosen as our candidate-choosing method. The proposed algorithm is implemented on C language (algorithm level). The different iteration performance is shown in Figure 5.1, i.e. 1 iteration= 2 runs. Compared to the previous studies [9, 10], our performance in this report is more effective. Any experimental parameters are not required to be used in our proposed algorithm.

SEW algorithm is also used to create candidate list in our algorithm. The SEW performance and the shortened BTC performance in Figure 5.2 indicates that our proposed algorithm suits for different candidate-choosing method.

The number of low reliability bit for generating the test pattern of the Chase 2 method is also analyzed because the different numbers of the test bits spend different run time, and there is different performance. There are 2, 3, 4, and 5 test bits in the simulation of performance Shown on Figure 5.3. The result exhibits performance convergence on 3 test bits. We decide to use 3 bits in our design and all of the bits is not too more plenty of bits

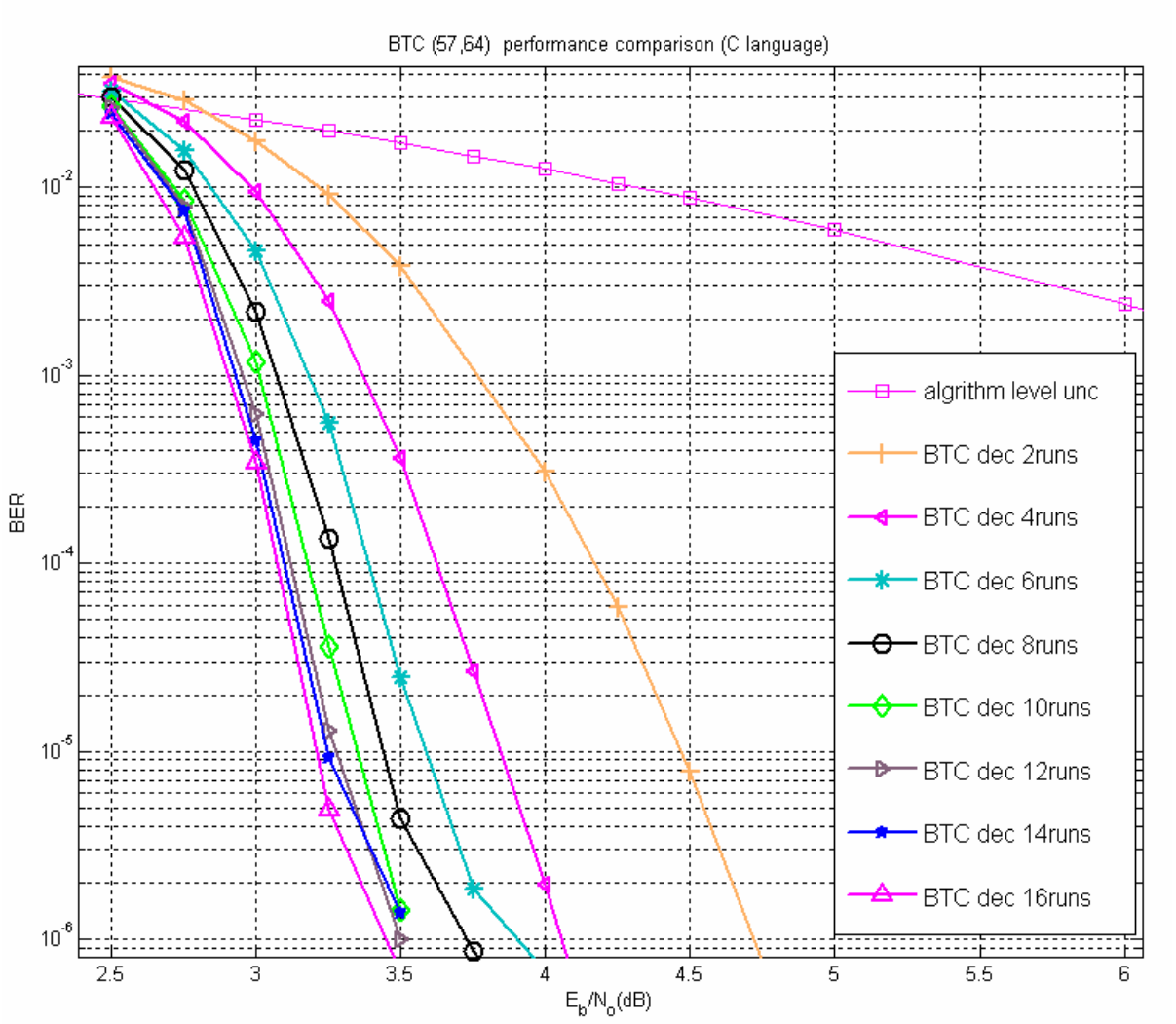


Figure 5.1: The performance of different runs simulation

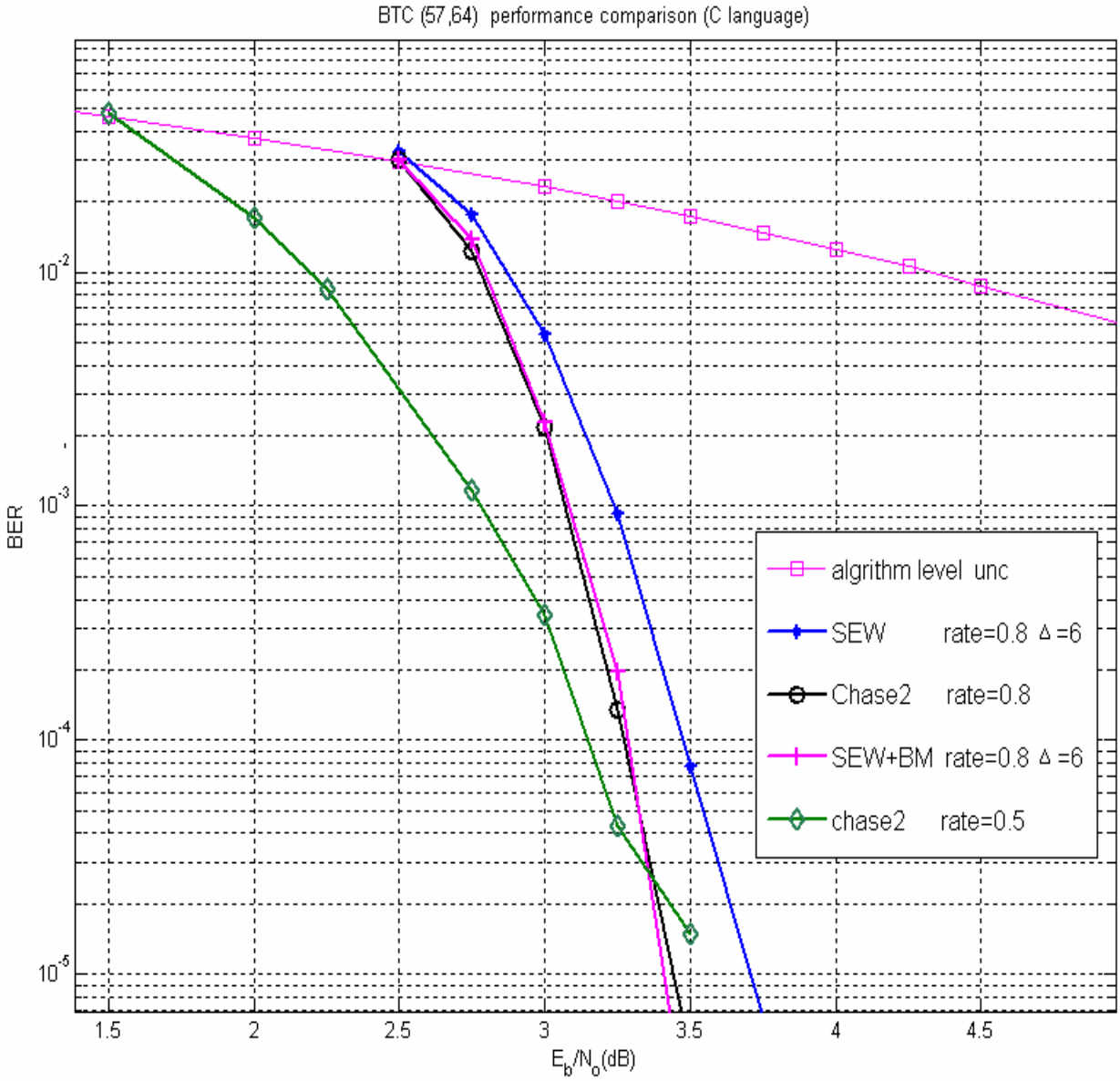


Figure 5.2: The performance of different list decoding algorithm

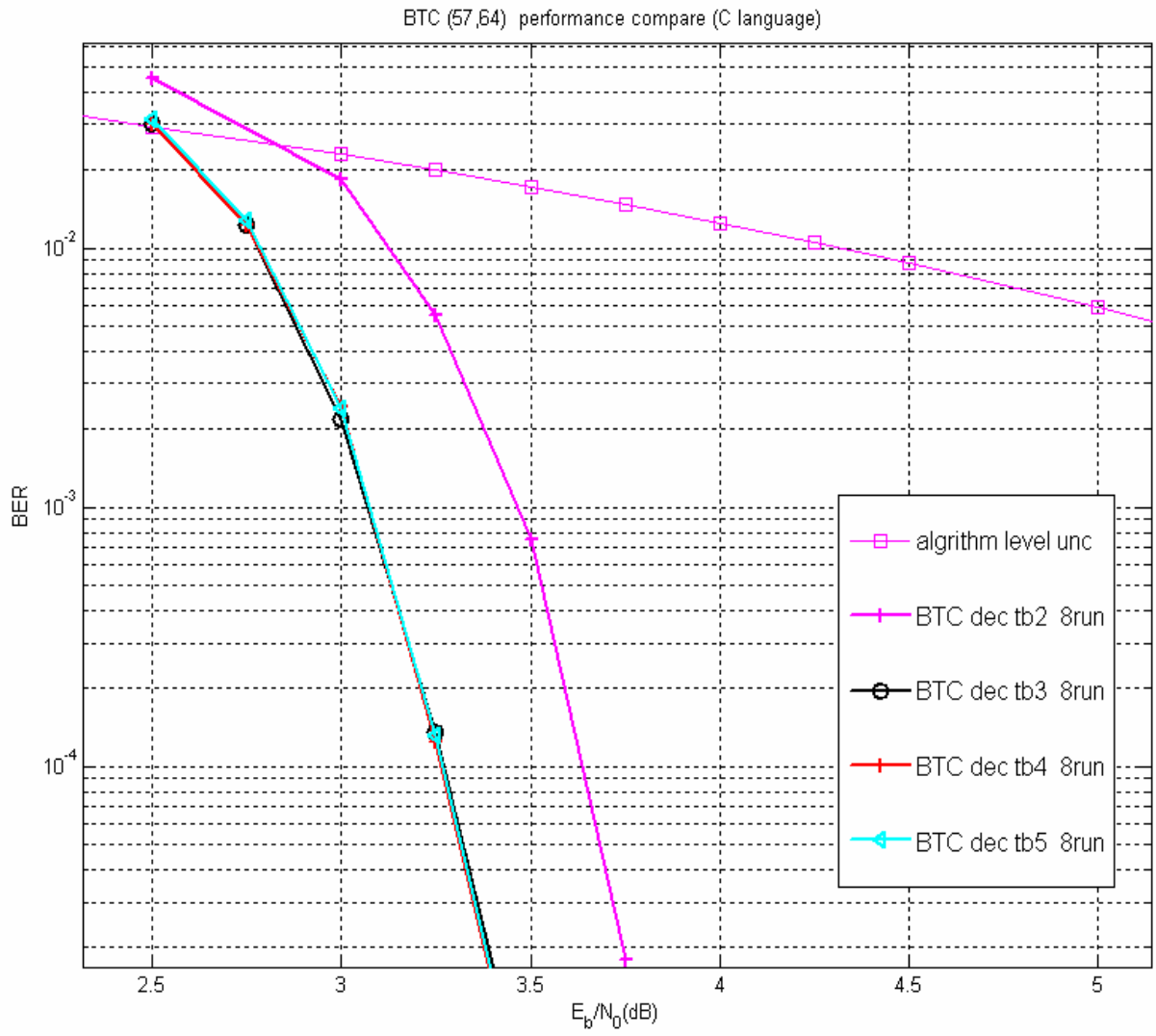


Figure 5.3: The performance of different test bits usage

5.2 SystemC Platform

SystemC is developed from the C++ programming language. SystemC extends the capabilities of C++ by enabling modeling of hardware descriptions and provides very important C++ library, such as concurrency, timed events and data types for hardware behavior model. These libraries are all legal C++ instruction. SystemC doesn't have new syntax instruction to the C++ programming language, and it provides a simulation kernel. SystemC model can be used to simulate the executable specification of the design or system that you write in SystemC. We can effectively describe a cycle-accurate model of our design. SystemC also provides a methodology for describing: system level design, software algorithms, and hardware architecture. More importantly, SystemC is an open source. This means that it is freely available to use under an open source license agreement.

Circuit design is getting bigger and bigger in gate count and faster in speed and more complex. There are some considerations: faster simulation, hardware/software co-simulation, and architectural exploration. Therefore, we expect a platform which has these features. Because SystemC is just for: System level design, describing hardware architectures, describing the software algorithms, verification, and IP exchange. For this reason, SystemC platform is used for our design, and the design can also provide electrical system level design(ESL) to be used in the future.

5.3 Hardware Level Simulation and Report

SystemC model is implemented and it can be co-simulated with every sub-module of RTL code. Their function is identical in cycle accumulation. A co-simulated verification platform is constructed and the platform is illustrated in Figure 5.4. The performances of the different precisions are shown in Figure 5.5, where P.X.Y means X bits for integer and Y bit for decimal. In the simulation result, the P3.5 precision is used in our final circuit and synthesized.

The synthesizer EDA tool which is design compiler of Synopsys Company's tool is used to synthesize our design and its result is shown in Table.5.1. The timing constrain is met at 200MHz and our circuit is implemented in 0.13um CMOS process.

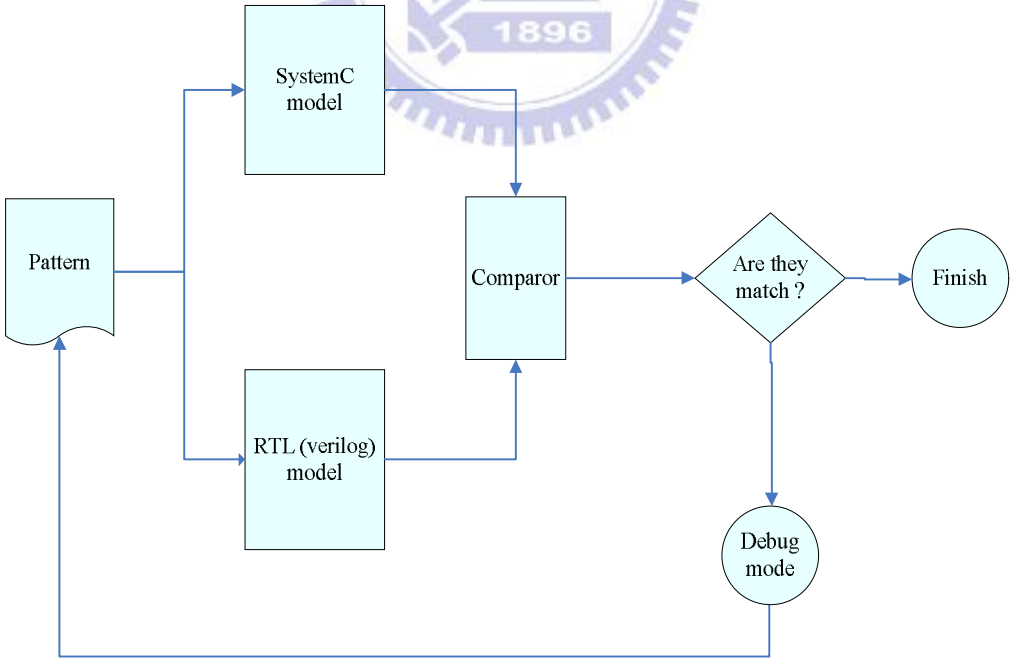


Figure 5.4: The verification flow diagram

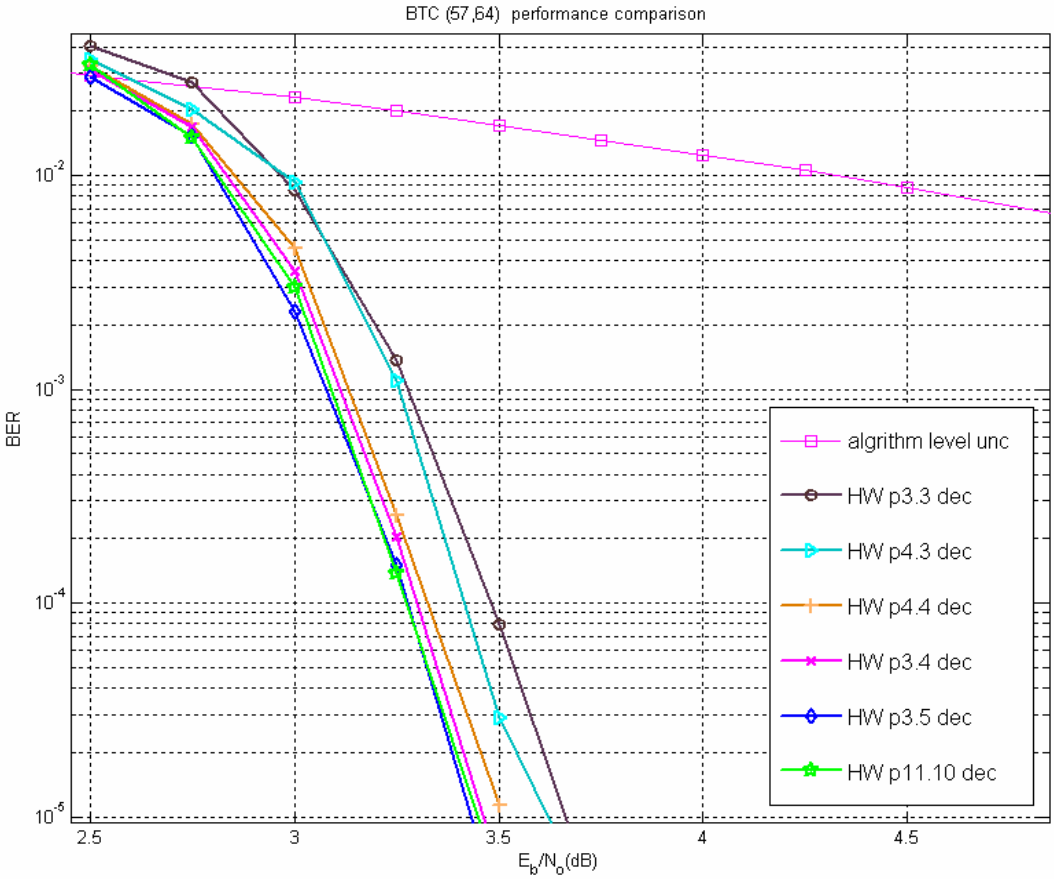


Figure 5.5: The performance of different precision analysis

Figure 4.13

The sub module of decoder	Gate count (0.13um)
Controller	1000
SISO	23916
MBA	9095
Output Buffer	15406

Top module (parallel architecture)	Gate count
Encoder	28,193
decoder	590,163

Table 5.1: Gate count report

Chapter 6

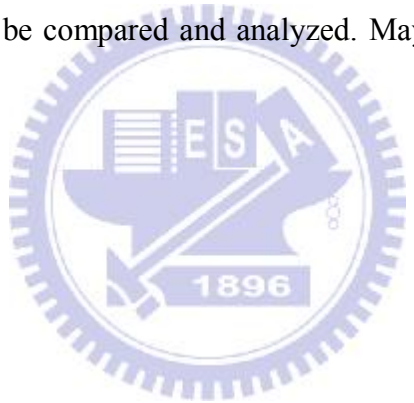
Conclusions

6.1 Summary

In this thesis, a BTC codec for WiMAX (IEEE 802.16e) standard is implemented. Our proposed geometric-like decoding provides the similar or better the previous error performance and there are not experimental parameters. We also consider parallelism for the throughput requirement of WiMAX (IEEE 802.16e) specification. Therefore, we not only design basis circuit, but also develop a multi-bank-array algorithm to deal the memory accessing hazard of matrix transposition. Moreover, the SystemC environment is built to simulate and verify our design, leading to reduction in hardware level simulation time. The environment also provide SystemC model for ESL in our study. The model can be use on ESL flow in the future.

6.2 Future Works

The WiMAX (IEEE 802.16e) system has many different specification of the BTC specification. The BTC (64, 57) is implemented in this thesis. Other specification may be also implemented by using our proposed algorithm and design flow. Beside, our proposed iterative decoding algorithm can be used in different candidate list algorithm, this is another study. The front-end design of our design has done, so we could finish back-end to tap out a IC, and IP-lize our BTC. Additionally, BCJR can be used in BTC that is also an important topic. The performance of conventional BTC and the BCJR algorithm BTC can be compared and analyzed. Maybe a new algorithm could be innovated.



Bibliography

[1] P. Elias, "Error-Free Decoding," *IEEE Trans. Inform. Theory*, vol.4, no.4, pp. 29 - 37, sep. 1954.

[2] J. Lodge, P. Hoeher, and J. Hagenauer, "Separable MAP "filters" for the Decoding of Product and Concatenated Codes," *in IEEE Int. Conf. Proc.*, vol. 3, pp. 1740-1745, May. 1993

[3] R. Pyndiah, A. Glavieux, A. Picart, S. Jacq., "Near optimum decoding of product codes," *in proc. IEEE GLOBGECOM'94*, vol.113 pp.339 - 343, Dec. 1994.

[4] WiMAX Forum, http://www.wimaxforum.ort/technology/.

[5] G. D. Forney, Jr., "Generalized Minimum Distance Decoding," *IEEE Trans. Inform. Theory*, vol.11, no. 2, pp.125-31, April. 1966.

[6] D. Chase, "Class of algorithms for decoding block codes with channel measurement information," *IEEE Trans. Inform. Theory,* vol.18, pp. 170-182, Jan. 1972.

[7] M. Lalam, K. Amis, D. Leroux, "On the use of Reed-Solomon codes in Space-Time Coding,". *IEEE International Symposium, PIMRC'05,* vol.1, pp.31 - 35, Sept. 2005..

[8] M. Lalam, K. Amis, D. Leroux, D. Feng, J. Yuan, "an improved iterative decoding algorithm for block turbo codes," *IEEE International Symposium, PIMRC'06,* pp. 2403 – 2407, July. 2006

[9] R.M. Pyndiah, "Near-optimum decoding of product codes: block turbo codes," *IEEE Tran., Communications*, vol. 46, pp. 1003-1010, Aug. 1998.

[10] C. Argon, S.W McLaughlin, "A parallel decoder for low latency decoding of turbo product codes," *IEEE Commun. Lett.*, Vol. 6, no. 2, pp. 70 - 72, Feb. 2002.

[11] F. Dongning, Y. Jinhong, K. Amis,"Rate-Compatible Shortened Turbo Product Codes," *IEEE VTC'06.*, vol. 5, pp. 2489 - 2493, spr. 2006

[12] S. Lin, Daniel J. Costello, *Error control codings*, second. Ed., 1942.

[13] E.R. Berlekamp, *Algebraic Coding Theory,* McGraw-Hill, New York, 1968.

[14] R.W. Hamming, "Error detecting and error correcting codes," *Bell Syst. Tech. J.*, vol. 29, pp. 147-160, Apr. 1950.

[15] P. Elias, "Coding for noisy channels," *IRE Cov*. Rec, vol. pt.4, pp. 37-47, 1955.

[16] P. Elias, “List decoding for noisy channels," *Institute of Radio Engineers*, pp.94-104, 1957.

Vita

姓名： 廖俊閔

性別： 男

出生地：台中縣

生日： 民國六十六年七月六日

地址： 台中市北屯區和平里和祥街 205 巷 68 號

學歷： 國立交通大學電子工程研究所碩士班 2004/09~2007/12

國立勤益工商專校 1995/09~1998/06

台中私立新民高級中學 1993/09~1995/06

經歷： 義務役財務士官 1998/10~2002/07

博達科技生產部組長 2000/07~2002/05

論文題目：Block Turbo Code Codec Design and Implementation

區塊渦輪編解碼器設計與實現