國立交通大學

電子工程學系　電子研究所碩士班

碩士論文

一個使用互反雙重柵欄的渦輪解碼器設計

A Turbo Decoder Design Using Reciprocal Dual Trellis

學生：林振揚

指導教授：張錫嘉教授

中華民國九十八年七月

一個使用互反雙重柵欄的渦輪解碼器設計

A Turbo Decoder Design Using Reciprocal Dual Trellis

研究生：林振揚　　Student：Chen-Yang Lin

指導教授：張錫嘉教授　　Advisor：Hsie-Chia Chang

國立交通大學

電子工程學系 電子研究所 碩士班

碩士論文

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

Electronics Engineering
July 2009
Hsinchu, Taiwan, Republic of China

中華民國九十八年七月

一個使用互反雙重柵欄的渦輪解碼器設計

學生：林振揚　　　　　　　　　指導教授：張錫嘉 教授

國立交通大學

電子工程學系 電子研究所碩士班

摘　要

錯誤更正碼一般而言需要比較有彈性的選擇碼率來適應不同的通道環境。為了達到這個需求，不同碼率的解碼器必須要被提出來，而高碼率的錯誤更正碼是需要被採用來提高通道的使用效率及傳輸速度。一般的渦輪解碼器通常使用高基數(high radix)的柵欄結構來解高碼率的碼，因此當碼率升高時，柵欄的複雜度會呈指數函數現象提高。在本論文中，我們引用了互反雙重柵欄的結構來減低柵欄在高碼率解碼器的複雜度。此外，我們採用了 Sign-Magnitude 的數字表示方式來更進一步降低硬體複雜度。我們使用穿孔(puncture)技術在 WCDMA 的渦輪編碼器上來產生不同碼率的碼。在研究了四種碼率 1/3、1/2、2/3、4/5 的穿孔渦輪碼後，模擬結果顯示錯誤更正效能隨著碼率越低而提高。

在本論文最後，根據四種不同碼率的 SISO 解碼器合成解果顯示，當碼率提高時，邏輯閘只會有些許的增加。最後，我們提出了一個多重碼率渦輪解碼器的硬體架構，其傳輸速度會隨著操作碼率的提高而上升。根據在 90nm 製程下的實驗結果，所提出的解碼器包含 370k 的運算邏輯閘及 58kb 的儲存單元。在供應電壓 0.9 伏特下，操作在碼率 4/5 的功率消耗是 80mW，並且達到 101Mb/s 的傳輸速度。

A Turbo Decoder Design Using Reciprocal Dual Trellis

Student：Chen-Yang Lin　　Advisor：Dr. Hsie-Chia Chang

Department of Electronics Engineering

Institute of Electronics

National Chiao Tung University

Abstract

ECC codes generally require the selection of a flexible coding rate to meet different channel characteristics. In addition, the high code rate schemes are required to increase channel efficiency for high throughput systems. Since conventional turbo decoders in high code rate usually apply high radix trellis structure, the complexity of trellis increases exponentially as the code rate rises. In this thesis, we introduce the reciprocal dual trellis to reduce the trellis complexity for high code rate. The sign magnitude representation is introduced to lower the hardware complexity. We apply puncture methodology to turbo code of WCDMA to generate different code rates. After investigate four code rates 1/3, 1/2, 2/3, 4/5 of punctured turbo codes, the simulations results reveal that the performance can be improved by using lower code rate.

The synthesis results of the four code rate SISO decoders show that there is a moderate increase of logic gates as code rate rises. Finally, a multiple code-rate turbo decoder architecture using the reciprocal dual trellis is proposed. As the operating code rate rises the throughput also increases. Fabricated by UMC CMOS 90nm 1P9M process, the proposed decoder which contains 370K gates and 58kb storage elements can achieve 101Mb/s with 80mW under code rate 4/5.

誌謝

韶光荏苒、光陰匆匆、歲月如梭，兩年的碩士研究生涯，有如白駒過隙，終於，我也可以寫到這一頁…

首先要感謝張錫嘉老師，感謝老師不僅能夠在研究上給予我指導，更難能寶貴的是，我可以看到一位領導者應有的風範及態度，西方有句諺語:「要使水手去造船，不如教導他們嚮往大海。」老師一直提醒我們要做一隻會去找草來吃的羊，而不是等待著牧羊人的餵食，老師對於研究總是懷著赤子之心，總是在我困惑的時候，教我如何去思考，我想這是除了專業知識之外，是我從張老師身上找到最好的寶物之一，在此，再次感謝張錫嘉老師。

再來要感謝 Ocean 及 Oasis 的成員，實在是相當開心能夠在這裡學習。首先要感謝大頭學長，他總是扮演著老師的角色，對於我的研究成果的分享或是遇到瓶頸的時候，總是隨 call 隨到，並且很有遠見的幫我提一些建議。還有國光學長，除了感謝你常常幫我們處理工作站上的問題之外，也覺得你是個很可以分享生活瑣事的學長。還要感謝跟我一起度過碩士生涯的夥伴們，不論是一起做研究或是慶生，都覺得你們真的是相當窩心的好夥伴，謝謝你們曾經陪伴過我。

另外還要感謝新竹地區禪學社及領袖社的夥伴們，深深感受到能夠認識你們真的是我相當大的福氣，我們一起努力籌備著很多活動，讓我這個研究生老人還可以常常體會到年輕人的熱情、活力、還有一些的懵懂，哈哈!有了你們的陪伴，我的研究生涯才不會太枯燥，讓我還能常常保有一顆年輕的心，總是能夠很樂觀的去面對每個挑戰，謝謝你們。

最後我要感謝最親愛的家人以及女朋友，在我想退縮的時候拉我一把；在我失意的時候願意給我支持。最後將此論文獻給曾經幫助過我的人，我滿懷著感恩的心，謝謝你們。

Contents

1 Introduction **1**

1.1 Motivation 1

1.2 Thesis organization 2

2 Turbo Code **3**

2.1 Turbo principle 3

2.1.1 Encoder of turbo code 3

2.1.2 Turbo interleaver 5

2.1.3 Decoder of turbo code 5

2.1.4 Error floor effect 6

2.2 Decoding algorithm 7

2.2.1 The MAP algorithm 7

2.2.2 The Log-MAP and MAX-Log-MAP algorithm 11

2.3 Decoding with reciprocal dual trellis 13

2.3.1 Construct reciprocal dual trellis 15

2.3.2 Decoding based on reciprocal dual trellis structure 16

2.4 Sliding window method of turbo code 17

3 Reciprocal Dual Trellis Algorithm **20**

3.1 Log domain approach 20

3.1.1 Sign magnitude scheme 20

3.1.2 Proposed equation for calculating extrinsic value 21

3.2 Punctured convolutional codes 24

3.2.1 Apply rate compatible punctured turbo code to WCDMA 25

3.2.2 Decoding with SM and Dual-MAP algorithm 27

3.3 Performance analysis . . . 29

4 Dual-MAP Turbo Decoder Architecture **35**

4.1 Architecture overview . . . 35

4.2 Dual-MAP decoder . . . 37

4.2.1 SMM unit . . . 37

4.2.2 SMA unit . . . 38

4.2.3 Extrinsic unit . . . 39

4.3 Throughput evaluation . . . 39

5 Implementation Results **41**

5.1 Implementation of Dual-MAP turbo decoders . . . 41

5.1.1 Comparison of different high radix MAP decoders . . . 42

5.1.2 Layout specification . . . 43

5.2 Comparison of different turbo decoders . . . 44

6 Conclusion and Future Works **46**

6.1 Conclusion . . . 46

6.2 Future works . . . 46

List of Figures

1.1 Block diagram of digital communication system. 2

2.1 Turbo encoder with puncture. 4
2.2 Interleaver of turbo decoder. 5
2.3 Conventional Turbo decoder. 6
2.4 A rate 1/2 memory order 2 RSC encoder and its state transition diagram. 8
2.5 The trellis diagram of the (2,1,2) RSC encoder. 8
2.6 Origianl trellis and reciprocal dual trellis comparison. 14
2.7 The process diagram of sliding window algorithm. 18
2.8 Different sliding window length Comparison 19

3.1 Correction term of $\bar{\mathbf{E}}$ operation. 22
3.2 Puncture procedure. 24
3.3 Trellis transformation relationship 26
3.4 Performance of using Dual-MAP and MAP under punctured code 27
3.5 Turbo encoder for WCDMA. 28
3.6 Performance of different code rate with block length 2048. 29
3.7 Comparison of different sliding window size. 30
3.8 Comparison of different iteration. 31
3.9 Fixed point comparison. 32
3.10 Segment method. 33
3.11 Performance comparison of different segment method. 33
3.12 Performance of each code rate with design parameters. 34

4.1 Iterative decoding of turbo decoder. 36
4.2 Decoding schedule of Dual-MAP decoder. 37

4.3 Architecture of SMM unit. 37
4.4 Architecture of SMA unit. 38
4.5 Radix-2 and radix-4 Log MAP recursion units. 39
4.6 Architecture of Extrinsic unit. 40

5.1 Area distributin of modules in turbo decoder. 42
5.2 Layout photo of multiple code-rate turbo decoder. 43

List of Tables

3.1 Puncture tables 25
3.2 Summary of fixed representation in MAP decoder 31

5.1 Summary of synthesis result 41
5.2 Compare with radix-4×4 MAX-LOG-MAP circuits 42
5.3 Proposed turbo decoder chip summary 44
5.4 Comparison of different turbo decoders 45

Chapter 1

Introduction

1.1 Motivation

The fundamental block diagram of traditional digital communication system is illustrated in Fig. 1.1. The system transmit an information source to a destination through an unknown channel. Generally, the communication system is simplified to three component parts which consists of transmitter, receiver, and channel. The transmitter includes source encoder, channel encoder, and modulator, is used to transmit the information more effectively and more reliably over unknown channels. Furthermore, the receiver will reverse the signal received by demodulator, channel decoder, and source decoder. Since the channel impairments such as noise, interference and distortion may cause the error in the received signal, the channel encoder is used in the system in order to minimize the transmission errors by adding certain redundancy to the source codeword. These redundant bits can be used for error detecting and correcting. Thus, the channel coding eliminate the effects of noise disturbances compared with an uncoded communication system.

However, most channel code schemes are not flexible and efficient for modern data transmission. For videodata, some important parts must be well protected to ensure reconstructed quality. Therefore, we can use more check bits to protect the important part of videodata and use less check bits when transmit less important data. Further, this encode manner can also apply to meet different channel situations. For example, we can use more check bits when data will be transmit through a noisy channel and use less check bits when transmit through better channel.

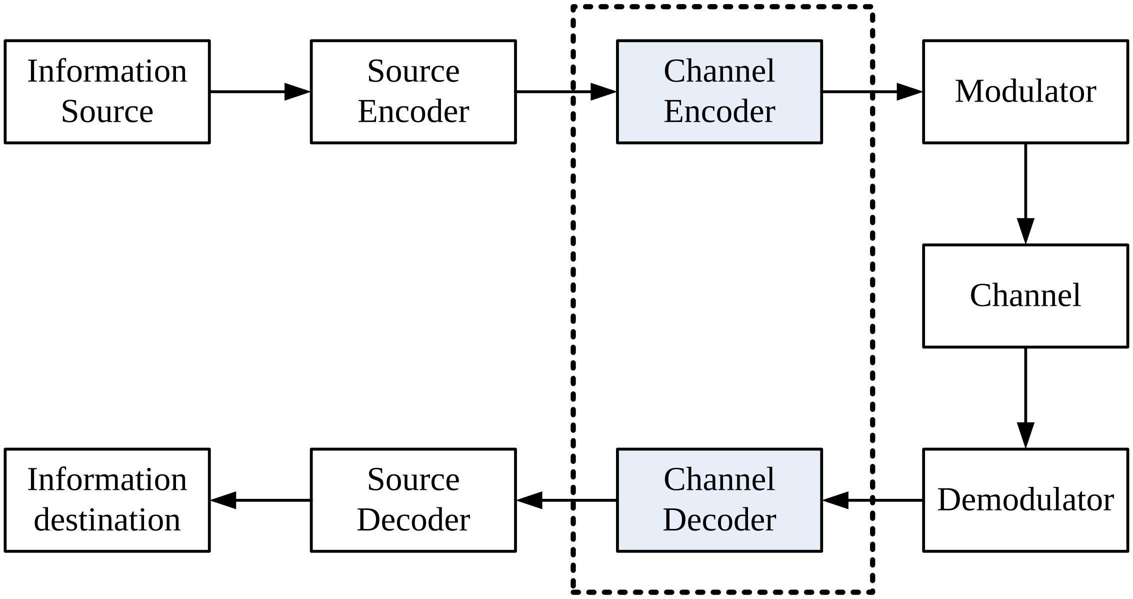

Figure 1.1: Block diagram of digital communication system.

From the view of the communication system, we integrate the features of data type and channel to design a flexible channel decoder. Furthermore, we focus on turbo decoders because it performs excellently on error correction ability. However, high code rate turbo decoder design is a real challenge since the complexity of its trellis structure. In this thesis, we will apply another decoding concept mentioned in Ref. [1] to slove this problem. Furthermore, we try to apply various code rates to protect different kind of data and to achieve unequal error protection. Finally, we want to design a multiple code rate and low hardware complexity turbo decoders.

1.2 Thesis organization

This thesis consists of 6 chapters. In chapter 2, the concepts of several iterative decoding algorithms of turbo codes will be introduced. In chapter 3, we will apply puncture tables to turbo codes of WCDMA system to achieve high code rate. Furthermore, the simulation analysis and hardware architecture parameters are also described. Chapter 4 introduces the design of reciprocal dual trellis turbo decoder, including the hardware architecture and characteristics of decoder. In chapter 5, the hardware implementation result will be shown. Finally, the conclusions and future works are given in chapter 6.

Chapter 2

Turbo Code

The parallel concatenated convolutional code, also named turbo code, was invented by C. Berrou, A. Glavieux, and P. Thitimajshima in 1993. It has been proved to have a excellent performance near Shannon limit. The common turbo encoder is composed of two recursive systematic convolutional code with parallel concatenated and separate by a pseudo random interleaver. Turbo code is adopted in 3GPP, 3GPP2, DVB-RCS and WiMAX standards due to its excellent error correction ability. In this chapter, turbo decoding with original trellis and reciprocal dual trellis will be introduced. The error floor effect in turbo decoding and some decoding techniques will also be interpreted.

2.1 Turbo principle

2.1.1 Encoder of turbo code

The turbo encoder is composed of two recursive systematic convolutional (RSC) encoders. Which are connected in parallel but separated by an interleaver. The block diagram of the turbo encoder is illustrated in Fig. 2.1. Puncture table is used to select the send bits or is an option to increase the data rate. In the first encoder, the information bits are encoded to the systematic part $c_0(D)$ and the parity part $c_1(D)$; thus, $c_0(D) = x(D)$. The second encoder encodes the bit stream $\tilde{x}(D)$, which are the information bits passing through the interleaver. However, the systematic part after interleaving $\tilde{x}(D)$ will be not be send during transmission. Code rate of an encoder is defined:

$$\text{R} = (\text{information bits in a codeword})/(\text{total codeword bits}).$$

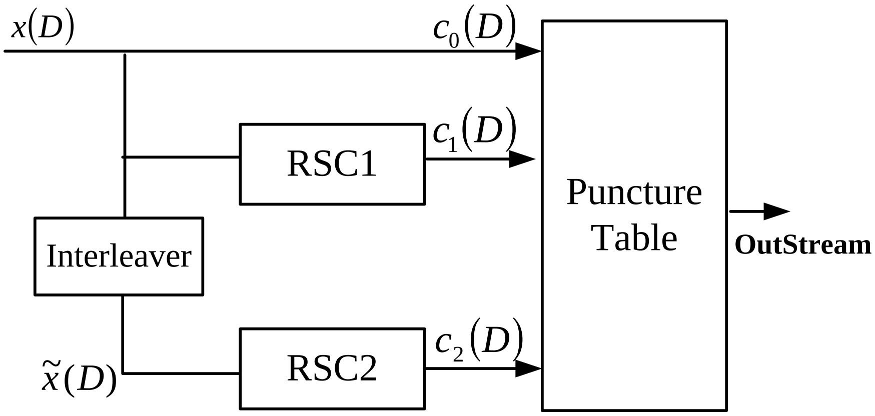

Figure 2.1: Turbo encoder with puncture.

The following derivations for R, we do not consider the puncture table. Hence, the *OutStream* in Fig. 2.1 is the codeword. Encoder 1 produces p_1 check bits and encoder 2 produced p_2 check bits, and the code rates are R_1 and R_2, respectively. If there are k information bits pass the turbo encoder and the overall turbo encoder code rate R can be derived as :

$$R = \frac{k}{k + p_1 + p_2}.$$

And we substitue p_1 and p_2 by applying code rate equations of encoder 1 and encoder 2.

$$R_1 = \frac{k}{k + p_1}, \quad R_2 = \frac{k}{k + p_2}$$

Finally, we can derive the code rate of the overall turbo encoder.

$$\frac{1}{R} = \frac{1}{R_1} + \frac{1}{R_2} - 1 \tag{2.1}$$

After encoding the information sequence, several terminating methods will be applied to stop the encoding process. We briefly describe three terminating methods. First, the simplest method is to truncate the information directly after encoded a block length. Therefore, this terminating approach results some performance loss. The second method is using dummy information bits at the end of the information sequence forcing registers in the encoder back to all zero states. This approach will maintain decoding performance, however, because of transmitting another dummy information bits, the code rate also decreases. The third is tail-biting method. This method encodes information bits twice. The first encoding procedure is aim to find the final register states in the encoder. The second encoding procedure is the actual encoding, and the encoder starts at the state which is the final state of first encoding procedure. Hence, tail-biting approach results the end of the register not necessary to all zero. This method ensures the same error protection as the second method mentioned above, but the code rate is not changed. If the ending state of the decoding trellis was known, we can set initial condition more precisely, and theoretically, the decoding performance can be improved.

2.1.2 Turbo interleaver

This is a process of rearranging the ordering of a data sequence in a one-to-one deterministic format. In turbo code, the interleaver such as in Fig. 2.2 is an essential component for bit-error-rate performance. A proper coding gain can be achieved with small memory encoders since the interleaver scrambles a long block input symbols. The interleaver de-correlates the input symbol between two encoders, therefore, an iterative decoding algorithm can be applied between two component decoders. The performance upper-bound corresponding to a uniform random interleaver has been evaluated in [2]. Theoretically, the block size (N) of interleave increase, the performance of bit-error-rate is expected to get better, and the factor 1/N is also called the interleaver gain.

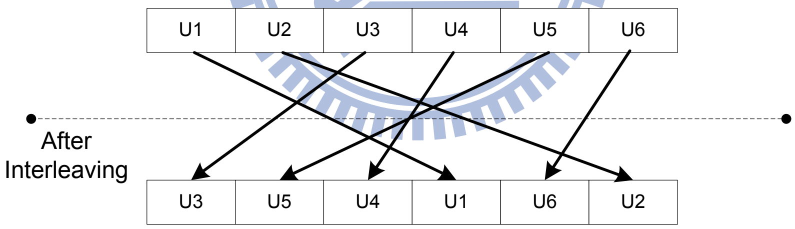

Figure 2.2: Interleaver of turbo decoder.

2.1.3 Decoder of turbo code

A common iterative turbo decoder is shown in Fig. 2.3. Where r_s is the received systematic information, r_{p_1} is the received parity generated by the first component RSC encoder, and r_{p2} is also the received parity generated by the second component RSC encoder. The iterative turbo decoding consists of two series constituent SISO (*soft in / soft out*) decoders, and are concatenated by the interleaver and de-interleaver. An interleaver is used to permute the systematic information and delivers the scrambled data into the second SISO decoder. During this iterative decoding procedure, each constituent SISO delivers the output extrinsic L_{ex} which is the *a priori* L_{in} for the next constituent SISO, therefore, $L_{in}^1 = L_{ex}^2$ and $L_{in}^2 = L_{ex}^1$ after the interleaver or de-interleaver process. Generally, the performance of bit-error-rate can be improved when the number of decoding iteration increases, however, there is no obvious improvement if a threshold of the iteration number has been reached.

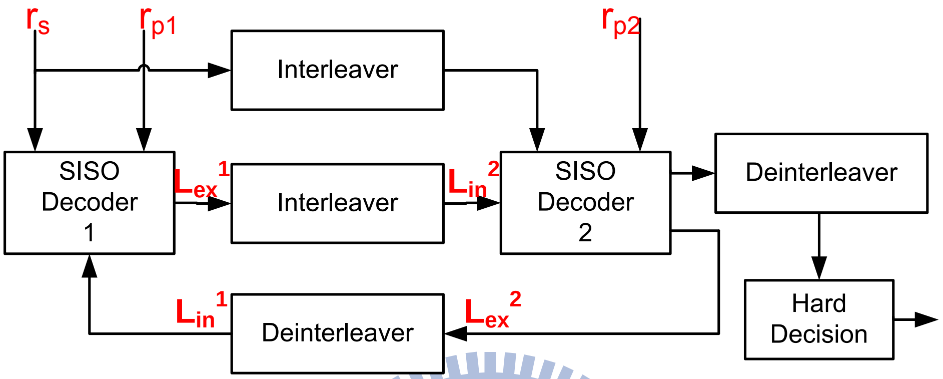

Figure 2.3: Conventional Turbo decoder.

2.1.4 Error floor effect

Although turbo coding provides an excellent performance, the bit-error-rate (BER) certainly decrease quite slowly and almost saturate at high signal-to-noise ratio (SNR). This phenomenon is due to relative small free distance of turbo codes and is called an *error floor* [3]. Consider the relation of minimum free distance and the bit error probability in turbo coding, which can be expressed by

$$P_b \propto Q\left(\sqrt{2d_{free}R\frac{E_b}{N_0}}\right), \tag{2.2}$$

where d_{free} is the minimum free distance of the codeword space, R is the code rate, and E_b/N_0 is the SNR.

2.2 Decoding algorithm

In turbo decoding algorithm, the maximum a posteriori probability (MAP) [4] algorithm and soft-output Viterbi algorithm (SOVA) [5] are commonly applied for the SISO decoders. Unlike the SOVA which uses maximum likelihood (ML) algorithm to minimize the word error probability, whereas, the MAP algorithm exploits the information of codeword to minimizes the symbol error probability. Therefoe, in this section, we will focus on MAP algorithm, because it has been proved that the MAP algorithm is the optimal decoding method for turbo codes compared with SOVA [6]. Moreover, some useful for hardware implementation algorithm such as the Log MAP and Max-Log MAP will also be introduced briefly. Finally, an effective decoding algorithm for high code rate will also be introduced [7].

2.2.1 The MAP algorithm

The MAP decoding algorithm (also called as *BCJR algorithm*), is introduced in 1974 by Bahl, Cocke, Jelinek, and Raviv [4]. For each transmitted information symbol u_t, the MAP algorithm estimates its *a posteriori probabilities* (APP) based on the whole received codeword sequence $\mathbf{r}$ over a discrete memoryless channel (DMC) and computes the log-likelihood ratio (LLR), which was defined as:

$$L(u_t) = L(u_t|\mathbf{r}) = \log \frac{P(u_t = +1|\mathbf{r})}{P(u_t = -1|\mathbf{r})}, \tag{2.3}$$

for $1 \leq t \leq N$, where N is the received codeword length, and compares this value to a zero threshold to determine the hard estimatimation of u_t :

$$u_t = \begin{cases} +1, & \text{if } L(\hat{u}_t) \geq 0 \\ -1, & \text{otherwise} \end{cases} \tag{2.4}$$

As an example, a rate 1/2 memory order 2 RSC encoder and its state transition are illustrated in Fig. 2.4. Note that the solid lines represent the state transitions corresponding to an information bit u_t of $+1$, while the dotted lines represent the state transitions corresponding to an information bit u_t of -1. Its decoding trellis diagram is shown in Fig. 2.5. In Fig. 2.5, the APP's in (2.3) can be computed by the summation of state transition

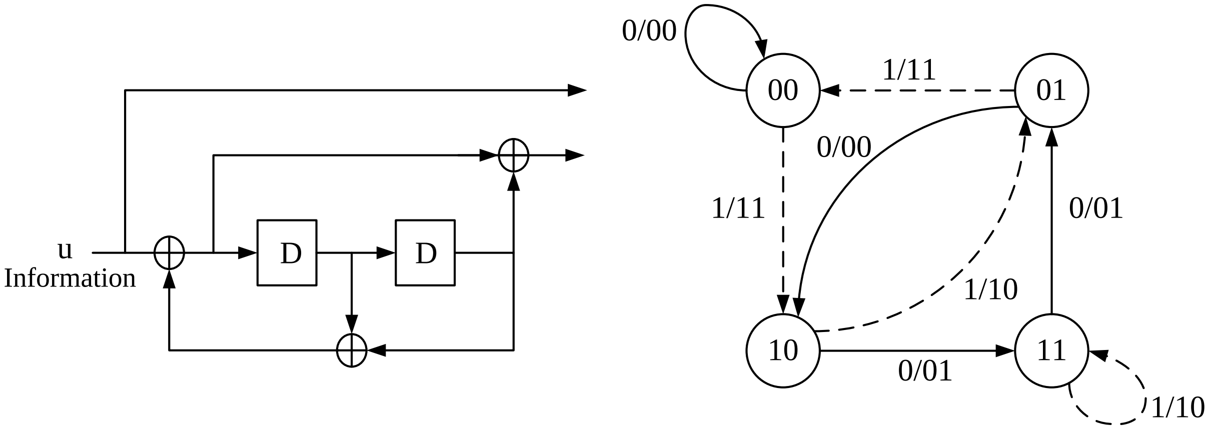

Figure 2.4: A rate 1/2 memory order 2 RSC encoder and its state transition diagram.

probabilities. Therefore, the equation can be further expressed as :

$$
\begin{aligned}
L(u_t) &= \log \frac{P(u_t = +1|\mathbf{r})}{P(u_t = -1|\mathbf{r})} \\
&= \log \frac{\sum_{(m',m)\in \mathbf{B}_t^{+1}} P(S_{t-1} = m', S_t = m|\mathbf{r})}{\sum_{(m',m)\in \mathbf{B}_t^{-1}} P(S_{t-1} = m', S_t = m|\mathbf{r})} \\
&= \log \frac{\sum_{(m',m)\in \mathbf{B}_t^{+1}} P(S_{t-1} = m', S_t = m, \mathbf{r})}{\sum_{(m',m)\in \mathbf{B}_t^{-1}} P(S_{t-1} = m', S_t = m, \mathbf{r})},
\end{aligned}
\tag{2.5}
$$

where $P(S_{t-1} = m', S_t = m, \mathbf{r})$ represents the joint probability of the existing transition from S_{t-1} at time t to S_t at time $t+1$. $\mathbf{B}_t^{+1}$ and $\mathbf{B}_t^{-1}$ is the sets of (m', m), denoted the state transitions which are due to input bit $u_t = +1$ and $u_t = -1$ respectively.

In order to compute the joint probability required for $L(u_t)$ in (2.5), we define the following metrics equations:

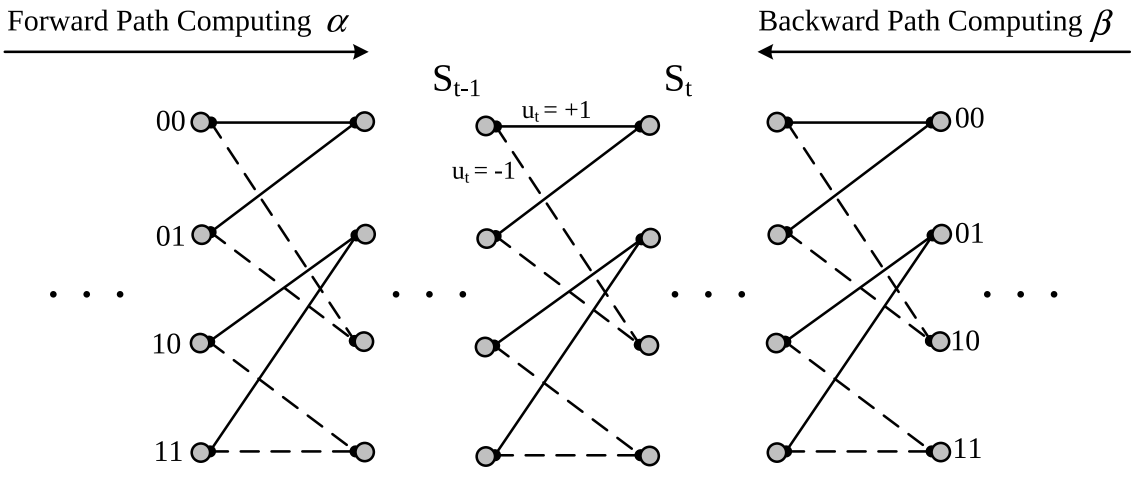

Figure 2.5: The trellis diagram of the (2,1,2) RSC encoder.

- The forward recursion metric α :

$$\alpha_t(m) = P\{S_t = m, \mathbf{r}_0^t\} \tag{2.6}$$

- The backward recursion metric β :

$$\beta_t(m) = P\{\mathbf{r}_{t+1}^{N-1} | S_t = m\} \tag{2.7}$$

- The branch metric γ :

$$\gamma_t(m', m) = P\{S_t = m, r_t | S_{t-1} = m'\} \tag{2.8}$$

- The joint probability λ :

$$\lambda_t(m', m) = P(S_{t-1} = m', S_t = m, \mathbf{r}) \tag{2.9}$$

Since we assume the codeword sequence after encoding is transmitted through discrete memoryless channel, the joint probability can be expressed as

$$\begin{aligned}\lambda_t(m', m) &= P(S_{t-1} = m', S_t = m, \mathbf{r}_0^{t-1}, r_t, \mathbf{r}_{t+1}^{N-1}) \\ &= P(\mathbf{r}_{t+1}^{N-1} | S_{t-1} = m', S_t = m, \mathbf{r}_0^{t-1}, r_t) \cdot P(S_t = m, r_t | S_{t-1} = m', \mathbf{r}_0^{t-1}) \cdot P(S_{t-1} = m', \mathbf{r}_0^{t-1}) \\ &= P(\mathbf{r}_{t+1}^{N-1} | S_t = m) \cdot P(S_t = m, r_t | S_{t-1} = m') \cdot P(S_{t-1} = m', \mathbf{r}_0^{t-1}).\end{aligned} \tag{2.10}$$

Here, $\mathbf{r}_0^{t-1}$ represents the received codecord sequence at time instance 0 to $t-1$, while $\mathbf{r}_{t+1}^{N-1}$ is at time instance $t+1$ to the end of sequence. The second equation of (2.10) results from Bayes' rule, and the third equation is due to the Markov process in the state transitions. Therefore, the joint probability defined in (2.9) can be expressed by terms of (3.1.2), (3.1.2) and (2.8), hence (2.9) can be written as :

$$\lambda_t(m', m) = \alpha_{t-1}(m') \cdot \gamma_t(m', m) \cdot \beta_t(m). \tag{2.11}$$

Now we will derive the equations (3.1.2), (3.1.2) and (2.8) as follow:

$$\begin{aligned}\alpha_t(m) &= P(S_t = m, \mathbf{r}_0^{t-1}) \\ &= \sum_{m' \in \mathbf{S}} P(S_{t-1} = m', S_t = m, \mathbf{r}_0^{t-1}) \\ &= \sum_{m' \in \mathbf{S}} P(S_t = m, r_{t-1} | S_{t-1} = m', \mathbf{r}_0^{t-2}) \cdot P(S_{t-1} = m', \mathbf{r}_0^{t-2}) \\ &= \sum_{m' \in \mathbf{S}} P(S_t = m, r_{t-1} | S_{t-1} = m') \cdot P(S_{t-1} = m', \mathbf{r}_0^{t-2}) \\ &= \sum_{m' \in \mathbf{S}} \alpha_{t-1}(m') \cdot \gamma_t(m', m).\end{aligned} \tag{2.12}$$

Since that the registers of the encoder are all zero in the beginning of encoding process, hence, the initial condition of α_t are :

$$\alpha_0(0) = 1, \quad \alpha_0(m) = 0 \quad \text{for } m \neq 0 \tag{2.13}$$

Similarly, we have

$$\begin{aligned}
\beta_t(m) &= P(\mathbf{r}_{t+1}^{N-1} | S_t = m) \\
&= \sum_{m' \in \mathbf{S}} P(S_{t+1} = m', \mathbf{r}_{t+1}^{N-1} | S_t = m) \\
&= \sum_{m' \in \mathbf{S}} P(S_{t+1} = m', r_{t+1}, \mathbf{r}_{t+2}^{N-1}, S_t = m) \ / \ P(S_t = m) \\
&= \sum_{m' \in \mathbf{S}} P(\mathbf{r}_{t+2}^{N-1} | S_{t+1} = m', r_{t+1}, S_t = m) \cdot P(S_{t+1} = m', r_{t+1} | S_t = m) \\
&= \sum_{m' \in \mathbf{S}} P(\mathbf{r}_{t+2}^{N-1} | S_{t+1} = m') \cdot P(S_{t+1} = m', r_{t+1} | S_t = m) \\
&= \sum_{m' \in \mathbf{S}} \gamma_{t+1}(m, m') \cdot \beta_{t+1}(m'),
\end{aligned} \tag{2.14}$$

where $\mathbf{S}$ represent the set of all states. If the trellis of encoding finally converges to zero state at $t = N - 1$, the following initial conditions of β_t are :

$$\beta_N(0) = 1, \quad \beta_N(m) = 0 \quad \text{for } m \neq 0 \tag{2.15}$$

Note that in Fig. 2.5, the forward metric α and the backward metric β are computed recursively in opposite direction, furthermore, the calculation of them requires the branch metric first. Hence, for any existing transitions from state m' to m in a trellis stage, the branch transition probability $\gamma_t(m', m)$ can be derived as :

$$\begin{aligned}
\gamma_t(m', m) &= P(S_t = m, r_t | S_{t-1} = m') \\
&= \frac{P(S_{t-1} = m', S_t = m, r_t)}{P(S_{t-1} = m')} \\
&= \frac{P(S_{t-1} = m', S_t = m)}{P(S_{t-1} = m')} \cdot \frac{P(S_{t-1} = m', S_t = m, r_t)}{P(S_{t-1} = m', S_t = m)} \\
&= P(S_t = m | S_{t-1} = m') \cdot P(r_t | S_{t-1} = m', S_t = m) \\
&= P(u_t) \cdot P(r_t | \mathbf{v_t}),
\end{aligned} \tag{2.16}$$

Note that $P(u_k)$ is the a-prior probability of u_k and $\mathbf{v_t}$ is the codeword associated with the transition $S_{t-1} = m'$ to $S_t = m$ corresponding to encoder input u_t.

As a summary of the MAP algorithm, with computation of $\gamma_t(m', m)$ in (2.16), we can derive α and β for each state at different time instances. As a result, the joint probability in (2.11) is also available for $t = 0, 1, \cdots, N-1$. The log-likelihood ratio $L(u_t)$ can be calculated by

$$L(u_t) = \log \frac{\sum_{(m',m)\in \mathbf{B}_t^{+1}} \alpha_{t-1}(m') \cdot \gamma_t(m', m) \cdot \beta_t(m)}{\sum_{(m',m)\in \mathbf{B}_t^{-1}} \alpha_{t-1}(m') \cdot \gamma_t(m', m) \cdot \beta_t(m)}. \tag{2.17}$$

2.2.2 The Log-MAP and MAX-Log-MAP algorithm

The MAP algorithm requires large memory and a large number of operations involving exponentiations and multiplications. The hardware realization of MAP decoder will be quite complex and difficult. Therefore, the Log-MAP algorithm is proposed to solve this problem. First, we transfer the branch metrics defined in the MAP algorithm to the logarithmic domain; that is

$$\bar{\gamma}_t(m', m) = \log \gamma_t(m', m). \tag{2.18}$$

Referring to (2.12) and (2.14), the forward path metric $\bar{\alpha}_t$ can be expressed as

$$\begin{aligned} \bar{\alpha}_t(m) &= \log \alpha_t(m) \\ &= \log \sum_{m'\in \mathbf{S}} e^{\bar{\alpha}_{t-1}(m') + \bar{\gamma}_t(m', m)}, \end{aligned} \tag{2.19}$$

and the backward path metric $\bar{\beta}_t$ can be expressed as

$$\begin{aligned} \bar{\beta}_t(m) &= \log \beta_t(m) \\ &= \log \sum_{m'\in \mathbf{S}} e^{\bar{\gamma}_{t+1}(m, m') + \bar{\beta}_{t+1}(m')}. \end{aligned} \tag{2.20}$$

Note that the initial conditions of path metrics also have changed, since all computations work with the logarithm domain.

$$\begin{aligned} &\bar{\alpha}_0(0) = 0, \quad \bar{\alpha}_0(m) = -\infty \quad \text{for } m \neq 0 \\ &\bar{\beta}_N(0) = 0, \quad \bar{\beta}_N(m) = -\infty \quad \text{for } m \neq 0 \end{aligned} \tag{2.21}$$

After substituting (2.18), (2.19) and (2.20), the APP information $L(\hat{u}_t)$ in (2.17) can be rewritten as

$$L(u_t) = \log \frac{\sum_{(m',m)\in \mathbf{B}_t^{+1}} e^{\bar{\alpha}_{t-1}(m') + \bar{\gamma}_t(m', m) + \bar{\beta}_t(m)}}{\sum_{(m',m)\in \mathbf{B}_t^{-1}} e^{\bar{\alpha}_{t-1}(m') + \bar{\gamma}_t(m', m) + \bar{\beta}_t(m)}}. \tag{2.22}$$

Considering the following Jocobian algorithm [8]

$$\begin{aligned}\log(e^{\delta_1}+e^{\delta_2}) &\equiv \max{}^*(\cdot)\\ &= \max(\delta_1,\delta_2)+\log(1+e^{-\left|e^{\delta_2}-e^{\delta_1}\right|}) \qquad (2.23)\\ &= \max(\delta_1,\delta_2)+f_c(|\delta_2-\delta_1|).\end{aligned}$$

where $f_c(\cdot)$ is a compensation function and thus the performance can be improved. By a recursive procedure of (2.23), the expression $\log(e^{\delta_1}+e^{\delta_2}+\cdots+e^{\delta_n})$ can be computed exactly, as follows

$$\begin{aligned}\log(e^{\delta_1}+e^{\delta_2}+\cdots+e^{\delta_n}) &= \log(\Delta+e^{\delta_n}), \quad \Delta=e^{\delta_1}+\cdots+e^{\delta_{n-1}}=e^{\delta}\\ &= \max(\log\Delta,\delta_n)+f_c(|\log\Delta-\delta_n|) \qquad (2.24)\\ &= \max(\delta,\delta_n)+f_c(|\delta-\delta_n|).\end{aligned}$$

Now we can use (2.23) to represent forward metrics in (2.19) and backward metrics in (2.20) as

$$\bar{\alpha}_t(m)=\max_{m'\in\mathbf{S}}{}^*\{\bar{\alpha}_{t-1}(m')+\bar{\gamma}_t(m',m)\}, \qquad (2.25)$$

and

$$\bar{\beta}_t(m)=\max_{m'\in\mathbf{S}}{}^*\{\bar{\gamma}_{t+1}(m,m')+\bar{\beta}_{t+1}(m')\}, \qquad (2.26)$$

Therefore, the (2.29) can be expressed as

$$\begin{aligned}L(\hat{u}_t) =& \max_{(m',m)\in\mathbf{B}_t^{+1}}{}^*\{\bar{\alpha}_{t-1}(m')+\bar{\gamma}_t(m',m)+\bar{\beta}_t(m)\}\\ &- \max_{(m',m)\in\mathbf{B}_t^{-1}}{}^*\{\bar{\alpha}_{t-1}(m')+\bar{\gamma}_t(m',m)+\bar{\beta}_t(m)\}.\end{aligned} \qquad (2.27)$$

The Log MAP algorithm, the (2.27), are considered to reduce the hardware complexity comparing with MAP algorithm. However, some difficulty for hardware implementation still exists since computing $f_c(\cdot)$ also involves exponentiations and multiplications. This problem can be solved by using a look up table, but this approach might result a little bit-error rate degration and increase the size of hardware.

In order to further simplify the complexity, consider the approximation derived in (2.28). As the approximation is used to reduce the complexity of the MAP algorithm, the performance of the Max-Log MAP algorithm is sub-optimal.

$$\log(e^{\delta_1}+e^{\delta_2}+\cdots+e^{\delta_n})\approx\max_{i\in\{1,2,\cdot,n\}}\delta_i. \qquad (2.28)$$

Note that the term $f_c(\cdot)$ is ignored in comparison with (2.24). Then we can simplify the equation (2.22) as follows:

$$\begin{aligned} L(u_t) = & \max_{(m',m)\in\mathbf{B}_t^{+1}} \{\bar{\alpha}_{t-1}(m') + \bar{\gamma}_t(m',m) + \bar{\beta}_t(m)\} \\ & - \max_{(m',m)\in\mathbf{B}_t^{-1}} \{\bar{\alpha}_{t-1}(m') + \bar{\gamma}_t(m',m) + \bar{\beta}_t(m)\}. \end{aligned} \tag{2.29}$$

Therefore, compared with the MAP algorithm, the Max-Log-MAP algorithm utilizes additions to replace the multiplications and avoids the complicated exponentiations. However, the performance would degrade because of the information loss in (2.28).

2.3 Decoding with reciprocal dual trellis

To meet the growing demand of high data rate at high bandwidth and power efficiencies, some researches have been focused on high code rate and their decoding algorithms that are powerful in the view of correction ability, yet reasonable complexity. However, for high rate k/n convolutional code $(n-k<k)$, the branch calculation in normal MAP algorithm applying on trellis constructed by encoder polynomial is highly complicated $\left(Radix-2^k\right)$. In such case, the MAP algorithm working on the corresponding reciprocal dual code's trellis is less complexity $\left(Radix-2^{n-k}\right)$ since the number of codeword in reciprocal dual code space are less than that of the original code.

Decoding trellis shown as Fig. 2.5 can be treated as a linear block code. All of the paths are possible codewords generated by one of the RSC encoder, and one can calculated by other codewords. For example, for an $(n=3, k=2)$ encoder with 2 registers and the decoding trellis using original and reciprocal dual trellis are illustrate in Fig. 2.6. Each state in reciprocal dual trellis is connected to 2 branches and that in original trellis is connected to 4 branches. Thus, if interleaver length is fixed and k gets larger, this decoding procedure seems to be difficult due to the complexity of original decoding trellis.

In [7], a new MAP decoding algorithm for high code rate convolutional codes using reciprocal dual convolutional code is presented. The advantage of this approach is a reduction of the computational complexity since the number of codewords to calculate is decreased for code rate higher than 1/2. According to [7], the log-likelihood ratio of a posterior probability $L(u_l)$ can be alternatively calculated by its reciprocal dual codewords

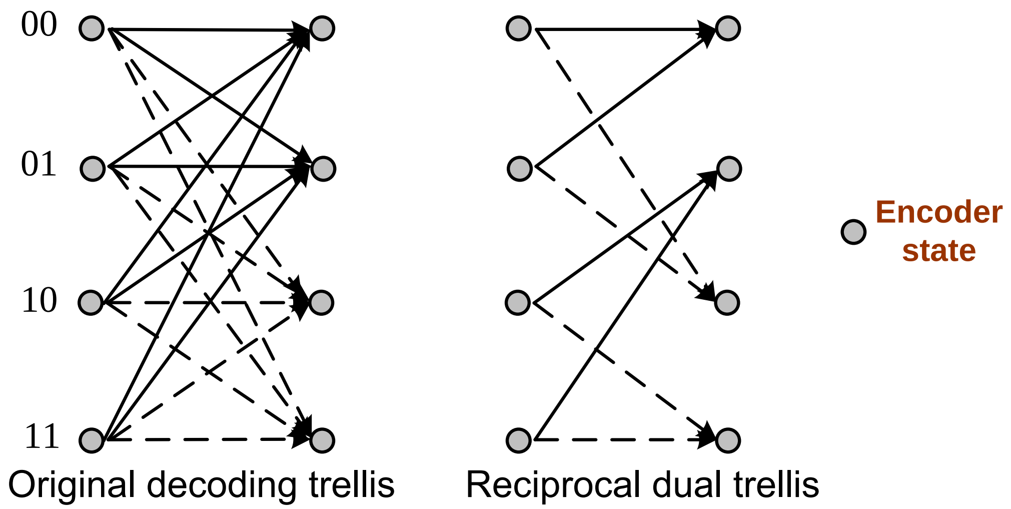

Figure 2.6: Origianl trellis and reciprocal dual trellis comparison.

$\tilde{\mathbf{c}}_i^\perp$, $1 \leqslant i \leqslant 2^{N-K}$, that is :

$$L(u_l) = L(c_l;y_l) + \log \frac{\sum_{\tilde{\mathbf{c}}_i^\perp \in \tilde{\mathbf{C}}^\perp} \prod_{j=0,j\neq l}^{N-1} tanh\left(L\left(c_j;y_j\right)/2\right)^{\tilde{\mathbf{c}}_{ij}^\perp}}{\sum_{\tilde{\mathbf{c}}_i^\perp \in \tilde{\mathbf{C}}^\perp} (-1)^{\tilde{\mathbf{c}}_{il}^\perp} \prod_{j=0,j\neq l}^{N-1} tanh\left(L\left(c_j;y_j\right)/2\right)^{\tilde{\mathbf{c}}_{ij}^\perp}} \tag{2.30}$$

Note that at the rightest side of (2.30) is the log-likelihood ratio of extrinsic value $L(\widehat{u}_l)$:

$$L(\widehat{u}_l) = \log \frac{\sum_{\tilde{\mathbf{c}}_i^\perp \in \tilde{\mathbf{C}}^\perp} \prod_{j=0,j\neq l}^{N-1} tanh\left(L\left(c_j;y_j\right)/2\right)^{\tilde{\mathbf{c}}_{ij}^\perp}}{\sum_{\tilde{\mathbf{c}}_i^\perp \in \tilde{\mathbf{C}}^\perp} (-1)^{\tilde{\mathbf{c}}_{il}^\perp} \prod_{j=0,j\neq l}^{N-1} tanh\left(L\left(c_j;y_j\right)/2\right)^{\tilde{\mathbf{c}}_{ij}^\perp}} \tag{2.31}$$

where $\mathbf{c} = (c_0, c_1, ..., c_{N-1})$ is a codeword of a systematic block code $\mathbf{C}$, $\tilde{\mathbf{c}}^\perp = \left(\tilde{\mathbf{c}}_0^\perp, \tilde{\mathbf{c}}_1^\perp, ..., \tilde{\mathbf{c}}_{N-1}^\perp\right)$ is a codeword of reciprocal dual code $\tilde{\mathbf{C}}$ of $\mathbf{C}$, and y_l refers to the matched filter output associated with c_l.

$$L\left(c_j;y_j\right) = \begin{cases} L\left(y_j|c_j\right) + L\left(c_j\right), & \text{if } c_j \text{ is an information bit} \\ L\left(y_t|c_j\right), & \text{if } c_j \text{ is an parity check bit} \end{cases} \tag{2.32}$$

Under an additive white Gaussian noise (AWGN) and has the varience $\sigma^2 = N_0/(2E_s)$, the term $L\left(y_j|c_j\right)$ can be written as :

$$L\left(y_j|c_j\right) = 4\frac{E_s}{N_0} \cdot y_j, \text{ where } N_0/(2E_s) \text{ is the signal-to-noise ratio} \tag{2.33}$$

And $L\left(c_j\right)$ ia the LLR of a prior probability, which is denoted :

$$L\left(c_j\right) = \log \frac{P(c_j = 0)}{P(c_j = 1)}. \tag{2.34}$$

2.3.1 Construct reciprocal dual trellis

In this section, we will introduce reciprocal dual trellis from some basic algebric properties of convolutional codes. A rate $R = k/n$ convolutional encoder under the field $F = GF(2)$ generates codeword $\mathbf{v}_t$ at time t

$$\mathbf{v}_t = (v_t^0, ..., v_t^{n-1}) \in F^n$$

and given $\mathbf{u}_t = (u_t^0, ..., u_t^{k-1})$ are information bits. Sequences of $\mathbf{u}_t$ and $\mathbf{v}_t$ can be written as

$$\mathbf{u}(D) = \sum_{t=0}^{\infty} \mathbf{u}_t D^t \quad , \quad \mathbf{v}(D) = \sum_{t=0}^{\infty} \mathbf{v}_t D^t$$

and the encoder can realize the mapping by the polynomial $\mathbf{G}(D)$ such that $\mathbf{v}(D) = \mathbf{u}(D)\mathbf{G}(D)$.

The dual convolutional code $C^{\perp}$ of a convolutional code C is a $(n-k)$-dimension which consists of all code seqence $\mathbf{v}^{\perp}(D)$ orthogonal to all $\mathbf{v}(D) \in C$. Hence, $C^{\perp}$ is a $(n, n-k)$ convolutional code generated by $\mathbf{H}$ with property $\mathbf{G}(D)\mathbf{H}^T(D) = \mathbf{0}$.

With a code C, a reciprocal convolutional code $\tilde{\mathbf{C}}$ can be obtained by substituting D^{-1} for D in $\mathbf{G}$ and by multiplying the $j-$th row with $D^{d(j)}$, where $1 \leqslant j \leqslant k$ and $d(j)$ is the degree of the $j-$th row of $\mathbf{G}(D)$. As a result, $\tilde{\mathbf{v}}(D) \in \tilde{C}$ is equal to the time-reversed sequence $\mathbf{v}(D^{-1})$.

We summarize the steps to construct reciprocal dual trellis for convolutional codes when its encoder is given by $\mathbf{G}(D)$:

1. Transfer $\mathbf{G}(D)$ to equivalent systematic encoder $\mathbf{G}_{sys}(D)$ if $\mathbf{G}(D)$ is not systematic.
2. Apply the property $\mathbf{G}(D)\mathbf{H}^T(D) = \mathbf{0}$ to find the corresponding parity check matrix $\mathbf{H}(D)$.
3. Calculate the reciprocal polynomial of $\mathbf{H}(D)$, and denote it as $\tilde{\mathbf{H}}(D)$.
4. The reciprocal dual trellis of $\mathbf{G}(D)$ can be constructed by using $\tilde{\mathbf{H}}(D)$.

Here, we show an example. A rate 2/3 convolutional code C is described by the nonsystematic polynomial generator matrix

$$\mathbf{G}\,(D) = \begin{pmatrix} 1+D & D & 1+D \\ D & 1 & 1 \end{pmatrix}$$

and is generated by the equivalent systematic matrix

$$\mathbf{G}_{sys}\left(D\right)=\begin{pmatrix}1 & 0 & \frac{1}{1+D+D^2}\\ 0 & 1 & \frac{1+D^2}{1+D+D^2}\end{pmatrix}$$

Then the rate 1/3 dual code $C^{\perp}$ is encoded by

$$\mathbf{H}\left(D\right)=\begin{pmatrix}1 & 1+D^2 & 1+D+D^2\end{pmatrix}$$

Note that $\mathbf{H}(D)$ is the parity check matrix of $\mathbf{G}(D)$ with the property $\mathbf{G}(D)\mathbf{H}^T(D)=\mathbf{0}$. Hence, the reciprocal dual code $\tilde{\mathbf{C}}^{\perp}$ is generated by $\tilde{\mathbf{H}}(D)$

$$\tilde{\mathbf{H}}(D)=D^2\cdot\mathbf{H}(D^{-1})=\begin{pmatrix}D^2 & 1+D^2 & 1+D+D^2\end{pmatrix}$$

2.3.2 Decoding based on reciprocal dual trellis structure

In [7], (2.31) can be represented as the relationship of encoder states transition. First of all, we define two sets $\mathbf{S}_A\left(\mathbf{s}\right)$ and $\mathbf{S}_B\left(\mathbf{s}\right)$ to describe the possible transitions from a state $\mathbf{s}$ to another state within one trellis stage. $\mathbf{S}_A\left(\mathbf{s}\right)$ contains the states $\mathbf{s}_i$ such that there exits the transition $\mathbf{s}_i\rightarrow\mathbf{s}$, and $\mathbf{S}_B\left(\mathbf{s}\right)$ is the set of destination states $\mathbf{s}_j$ from state $\mathbf{s}$ $\left(\mathbf{s}\rightarrow\mathbf{s}_j\right)$. Therefore, $\mathbf{S}_A$ and $\mathbf{S}_B$ are the same meaning as α and β. Here, we apply α, β, and γ parameters to represent forward and backward recursions.

Moreover, bits associated with transition $\mathbf{s}_1\rightarrow\mathbf{s}_2$ are combined in the n tuple, that are $\left(b_0\left(\mathbf{s}_1,\mathbf{s}_2\right),...,b_{n-1}\left(\mathbf{s}_1,\mathbf{s}_2\right)\right)$. Using the substitution $g_j=tanh\left(L\left(c_j;y_j\right)/2\right)$ at t trellis stage, and define the partial products :

$$\gamma_t\left(\mathbf{s}_1,\mathbf{s}_2\right)=\prod_{j=0}^{n-1}g_{t\times n+j}^{b_j\left(\mathbf{s}_1,\mathbf{s}_2\right)}\tag{2.35}$$

The forward recursion

$$\alpha_{t+1}\left(\mathbf{s}\right)=\sum_{\mathbf{s}'\in\mathbf{S}_A\left(\mathbf{s}\right)}\alpha_t\left(\mathbf{s}'\right)\cdot\gamma_t\left(\mathbf{s}',\mathbf{s}\right),\quad 0\leqslant t<N-1\tag{2.36}$$

The backward recursion

$$\beta_{t-1}\left(\mathbf{s}\right)=\sum_{\mathbf{s}'\in\mathbf{S}_B\left(\mathbf{s}\right)}\beta_t\left(\mathbf{s}'\right)\cdot\gamma_{t-1}\left(\mathbf{s},\mathbf{s}'\right),\quad 2\leqslant t\leqslant N\tag{2.37}$$

If we direct truncate the trellis at the end of receiving a codeword, the boundary conditions are $\alpha_0\left(\mathbf{s}\right)=1,\quad 0\leqslant\mathbf{s}<2^v$, $\beta_N\left(\mathbf{s}\right)=0,\quad 1\leqslant\mathbf{s}<2^v$ and $\beta_N\left(\mathbf{0}\right)=1$ where v is the number of register in one RSC and 2^v is the state number of the encoder. Thus (2.31) can be rewritten as (2.39), where the special products $\tilde{\gamma}_t(\mathbf{l}, \mathbf{s}, \mathbf{s}')$

$$\tilde{\gamma}_t(\mathbf{l}, \mathbf{s}, \mathbf{s}') = \prod_{j=0, j \neq l-t\cdot n}^{n-1} g_{t\cdot n_0+j}^{b_j(\mathbf{s}_1,\mathbf{s}_2)} \tag{2.38}$$

The time instant $t = \lfloor \mathbf{l}/n \rfloor$ depends on index $\mathbf{l}$.

$$L(\widehat{u}_l) = \log \frac{\sum_{\mathbf{s}_1=0}^{2^v-1} \sum_{\mathbf{s}_2 \in S_B(\mathbf{s}_1)} \alpha_t(\mathbf{s}_1) \cdot \tilde{\gamma}_t(\mathbf{l}, \mathbf{s}_1, \mathbf{s}_2) \cdot \beta_{t+1}(\mathbf{s}_2)}{\sum_{\mathbf{s}_1=0}^{2^v-1} \sum_{\mathbf{s}_2 \in S_B(\mathbf{s}_1)} \alpha_t(\mathbf{s}_1) \cdot (-1)^{b_{l-t\cdot n}(\mathbf{s}_1,\mathbf{s}_2)} \cdot \tilde{\gamma}_t(\mathbf{l}, \mathbf{s}_1, \mathbf{s}_2) \cdot \beta_{t+1}(\mathbf{s}_2)} \tag{2.39}$$

Finally, the LLR of a posterior probability (APP) can be represented by equations (2.33), (2.34), and (2.39)

$$L(u_t) = L(c_j) + 4\frac{E_s}{N_0} \cdot y_j + L(\widehat{u}_l). \tag{2.40}$$

2.4 Sliding window method of turbo code

In the traditional SISO decoding algorithm, the LLR of APP computation requires the path metric values generated by the forward and backward processes. Furthermore, since the backward recursive computation initials from the end of decoding trellis, as shown in Fig. 2.5, the decoding process can be started after the entire block message to be received. If the received sequence length is large, it will lead to long output latency and huge memory requirement for hardware implementation. For example, the maximum block length of 3GPP standard is 5114, which means 5114 LLR values and path metrics should be stored. It is the main disadvantage of turbo code for real applications.

The main problem is that long block length can not be divided into several short sub-block immediately, since the unknown initial condition of backward recursive metrics computations will damage the performance of turbo codes. Therefore, the sliding window approach was proposed [9] to overcome it. This algorithm utilizes the fact that the backward metrics can be highly reliable even without the initial condition if the backward recursion goes long enough. Fig. 2.7 shows the process of the sliding window algorithm and will be further illustrated as follows. First, the received codeword sequence is divided into several sub-blocks of length of W. And W is called the convergence length, which normally is set to be five times the constraint length of component encoder in turbo code to ensure the reliable initialization. In the sliding window approach, the end of sub-block

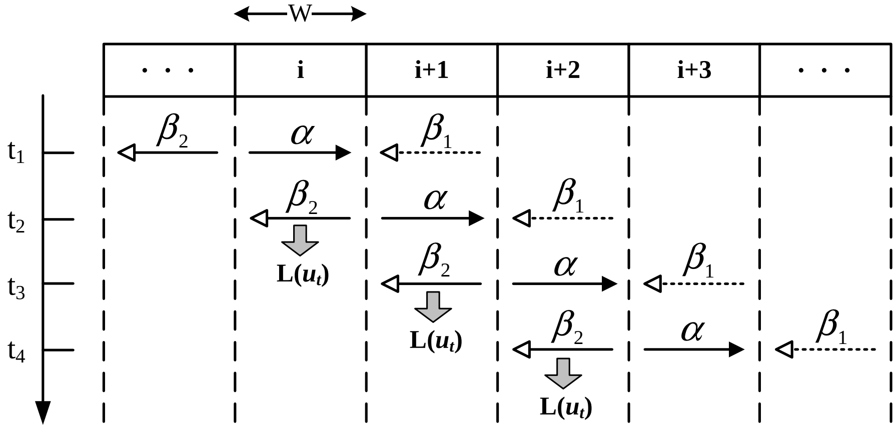

Figure 2.7: The process diagram of sliding window algorithm.

is the initial of next sub-block whether the forward or backward recursive operation. Thus, the initial metric values are inherited from the last metrics calculated in the previous sub-block. Note that the dummy backward recursion β_1 is employed to establish the initial condition for the true backward recursion β_2. Although the initial condition for the β_1 is unknown except the last sub-block, we utilize the equally likely condition for the β_1 values at time instance $(i+1)\cdot W$:

$$\beta_1(m) = \frac{1}{M}, \quad \text{for all } m \in \mathbf{S} \tag{2.41}$$

where **S** represents all possible states and M is equal to the total state number. During the forward recursion α proceeds in the i-th sub-block and stores these values into memory, the dummy backward recursion β_1 is performed in the $i+1$ sub-block concurrently. As soon as the β_1 computation is finished, the initial metrics in the i-th sub-block are available for the β_2 recursion. And $L(\hat{u}_t)$ can be calculated based on the α metrics in the memory, the β_2 metrics in computation, and the corresponding branches metrics in the i-th sub-block.

For example, we concatenate two truncated component codes defined by

$$G(D) = \begin{pmatrix} 1 & 0 & \frac{1}{1+D+D^2} \\ 0 & 1 & \frac{1+D^2}{1+D+D^2} \end{pmatrix}$$

and using a block interleaver with length= 400 to abtain a $(800, 400)$ code of rate= $1/2$.

Compared to different sliding window, the bit error rate(BER) after six iterations is shown in Fig.2.8(SW25 represents window length=25 trellis stage and SW200 represents no sliding window). The reciprocal dual trellis decoding with sliding window is significantly less complex than its optimum counterpart in terms of memory cost, however, it achieves for both codes virtually the same bit error performance.

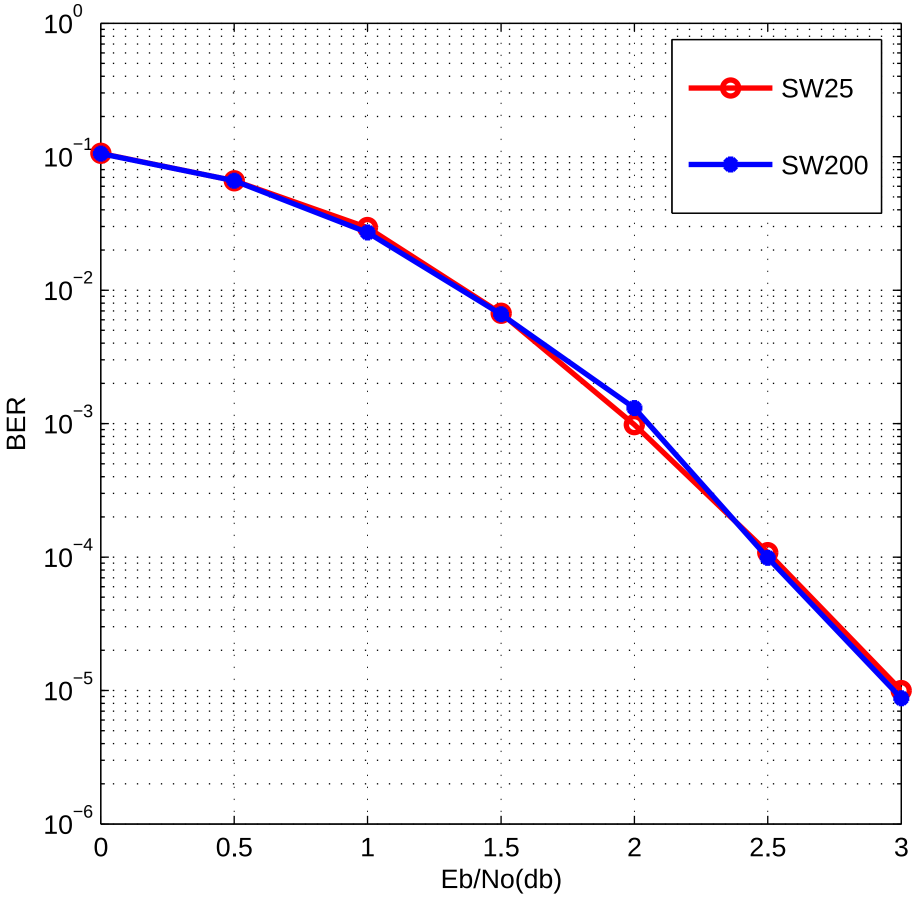

Figure 2.8: Different sliding window length Comparison

Chapter 3

Reciprocal Dual Trellis Algorithm

3G mobile multimedia communication systems based on WCDMA support a flexible transmission capability to provide packet data service as well as voice service. Mobile multimedia communication requires a coding scheme that can accommodate various code rate requirements. In this thesis, we apply puncture skill to get various code rates and adopt WCDMA turbo code as the mother code, and using reciprocal dual trellis as decoding trellis.

3.1 Log domain approach

For high code rate, MAP algorithm working on the reciprocal dual trellis is preferable. However, in (2.39) reqiures both adders and multipliers, it is considered to be much hardware complexity. Hence, we further reduce the hardward cost by taking LOG operation of all metrics which is similar to Log-MAP method. However, the challenge involved in log domain implementation of this algorithm when bit metric value $tanh\left(L\left(c_j;y_j\right)/2\right)$ is negative. To present these metrics, some numerical transformation is applied.

3.1.1 Sign magnitude scheme

In [10], the sign magnitude is a simple representation to represent the reciprocal dual trellis metrics. In this, a real number x is represented as $x = X_S e^{-X_M} \equiv [X_S, X_M]$, where $X_S = sign\left(x\right)$ and $X_M = -log\left(|x|\right)$. The arithmetic operations with this representation were defined here.

Negation :

$$-x \equiv [-X_S, X_M] \tag{3.1}$$

Addition :

$$x + y \equiv min^*(x, y) = [S(min(X_M, Y_M)), min(X_M, Y_M) - \log(1 + X_S Y_S e^{-|X_M - Y_M|})]$$
$$where \quad S(X_M) = X_S \quad or \quad S(Y_M) = Y_S \tag{3.2}$$

Multiplication :

$$x \times y \equiv Sum^*(x, y) = [X_S Y_S, X_M + Y_M] \tag{3.3}$$

Division :

$$x/y \equiv [X_S Y_S, X_M - Y_M] \tag{3.4}$$

In the normal log-MAP, additionis implemented by the equivalent **E** operation. For the addition of two non-negative real numbers a and b, represented in *log* domain a A and B, we have

$$a + b \equiv A\mathbf{E}B = min(A, B) - \log(1 + e^{-|A-B|}) \tag{3.5}$$

the second term $-\log(1+e^{-|d|})$, where d is the difference between the log domain values, is called the correction term. Another log domain operator that gives the absolute difference between two non-negative real numbers as :

$$|a - b| \equiv A\bar{\mathbf{E}}B = min(A, B) - \log(1 - e^{-|A-B|}) \tag{3.6}$$

this is also called the correction term. In sign-magnitude representation, the correction terms perform the algebraic addition of two real numbers, could involve either an **E** or $\bar{\mathbf{E}}$ operation depending on the relative signs of the two values. The correction term in the **E** operation which only takes values in the range $[-\log(2), 0]$, however, the $\bar{\mathbf{E}}$ operation $-\log(1 - e^{-|d|})$ shown in Fig. 3.1, can have any value in the range $[\infty, 0]$. This makes the fixed point hardware implementation more complex since the representation range should be chosen carefully, and warrants a large look up table (LUT) for the correction terms.

3.1.2 Proposed equation for calculating extrinsic value

In this section, we will discuss the computation of extrinsic value according to log domain scheme. According to (2.31), let us define the bit metric g_j

$$g_j = tanh\left(L\left(c_j; y_j\right)/2\right)$$

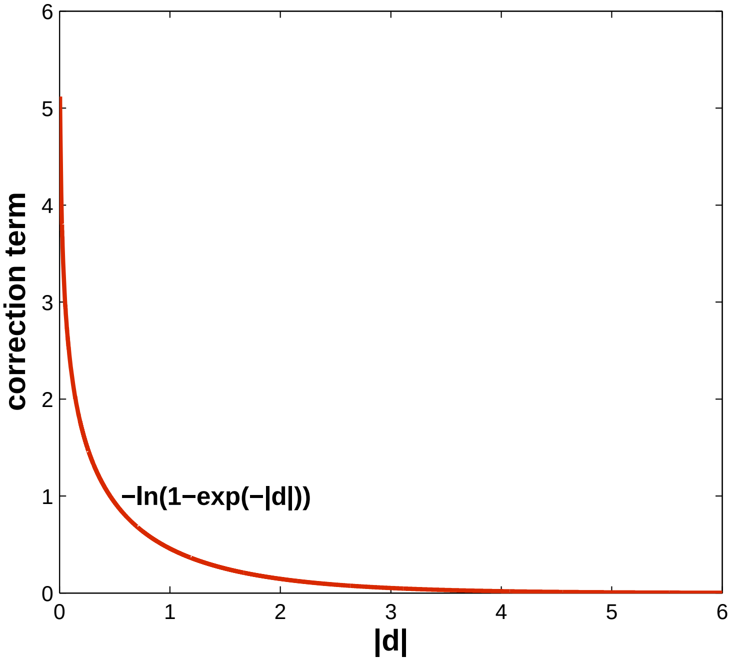

Figure 3.1: Correction term of $\bar{\mathbf{E}}$ operation.

and summation metric U_l^b.

$$U_l^b = \sum_{\tilde{\mathbf{c}}_i^\perp \in \tilde{\mathbf{C}}^\perp ; \tilde{\mathbf{c}}_{il}^\perp = b} \prod_{j=0}^{N-1} g_j^{\tilde{\mathbf{c}}_{ij}^\perp} \tag{3.7}$$

- The forward recursion metric α :

$$\alpha_t(\mathbf{s}_t) = \min{}^*(Sum^*_{\mathbf{S}}(\alpha_{t-1}(\mathbf{s}_{t-1}), \gamma_t(\mathbf{s}_{t-1}, \mathbf{s}_t)))$$

- The backward recursion metric β :

$$\beta_{t-1}(\mathbf{s}_{t-1}) = \min{}^*(Sum^*_{\mathbf{S}}(\gamma_t(\mathbf{s}_{t-1}, \mathbf{s}_t)), \beta_t(\mathbf{s}_t)))$$

- The branch metric γ :

$$\gamma_t(\mathbf{s}_{t-1}, \mathbf{s}_t) = Sum^*(g_j^{b_j(\mathbf{s}_{t-1}, \mathbf{s}_t)}, ..., g_{j+n-1}^{b_{j+n-1}(\mathbf{s}_{t-1}, \mathbf{s}_t)})$$

where $t = [j/n]$ is the time index ([x] denotes the integer part of x), b is the transition bit from $\mathbf{s}_{t-1}$ to $\mathbf{s}_t$ at decoding index l, and $\mathbf{S}$ is the set of possible transitions from state $\mathbf{s}_{t-1}$ to $\mathbf{s}_t$. The denomirator of (2.31) can be represented by the summation of two parts, it is :

$$\begin{aligned}
&\frac{1}{{g_l}^0}\cdot\sum_{\tilde{\mathbf{c}}_i^\perp\in\tilde{\mathbf{C}}^\perp;\tilde{\mathbf{c}}_{il}^\perp=0}\prod_{j=0}^{N-1}g_j^{\tilde{\mathbf{c}}_{ij}^\perp}+\frac{1}{{g_l}^1}\cdot\sum_{\tilde{\mathbf{c}}_i^\perp\in\tilde{\mathbf{C}}^\perp;\tilde{\mathbf{c}}_{il}^\perp=1}(-1)^1\prod_{j=0}^{N-1}g_j^{\tilde{\mathbf{c}}_{ij}^\perp}\\
&=U_l^0-(U_l^1/g_l)
\end{aligned}$$

The numerator of (2.31) can be written as :

$$\begin{aligned}
&\frac{1}{{g_l}^0}\cdot\sum_{\tilde{\mathbf{c}}_i^\perp\in\tilde{\mathbf{C}}^\perp;\tilde{\mathbf{c}}_{il}^\perp=0}\prod_{j=0}^{N-1}g_j^{\tilde{\mathbf{c}}_{ij}^\perp}+\frac{1}{{g_l}^1}\cdot\sum_{\tilde{\mathbf{c}}_i^\perp\in\tilde{\mathbf{C}}^\perp;\tilde{\mathbf{c}}_{il}^\perp=1}\prod_{j=0}^{N-1}g_j^{\tilde{\mathbf{c}}_{ij}^\perp}\\
&=U_l^0+(U_l^1/g_l)
\end{aligned}$$

Therefore, (2.31) can be simplified to :

$$L(\widehat{u}_l)=\log\frac{1+\frac{U_l^1}{U_l^0g_l}}{1-\frac{U_l^1}{U_l^0g_l}}\tag{3.8}$$

Equation (3.8) is also described in [10]. In fact, $\frac{U_l^1}{U_l^0g_l}$ can be represented as $[U_S^l,U_M^l]$ due to sign magnitude representation. Hence, (3.8) can be

$$L(\widehat{u}_l)=\log\frac{1+U_S^le^{-U_M^l}}{1-U_S^le^{-U_M^l}}\tag{3.9}$$

Using the relation $tanh(x/2)=(1-e^{-x})/(1+e^{-x})$, hence, (2.31) can also be written as

$$L(\widehat{u}_l)=\begin{cases}-\log(|tanh(U_M^l/2)|), & \text{if } U_S^l=1\\ \log(|tanh(U_M^l/2)|), & \text{if } U_S^l=-1\end{cases}\tag{3.10}$$

(3.10) is the proposed equation for calculating extrinsic values. Hence, the extrinsic equation (3.10) is much suitable to hardware design. According to (2.40), the LLR of a posteriori probability $L(u_l)$ is written as

$$L(u_l)=\log\frac{p_l(0)}{p_l(1)}+C\times r_s+L(\widehat{u}_l),\quad \text{where C is a constant and } r_s \text{ is a recieved symbol.}\tag{3.11}$$

Hence, the rule of decision is made by:

$$decision=\begin{cases}1, & \text{if } L(u_l)<0\\ 0, & \text{if } L(u_l)\geq 0\end{cases}\tag{3.12}$$

For convenient, we call the sign magnitude algorithm to SM and reciprocal dual trellis algorithm to Dual-MAP in the following sections.

1896

3.2 Punctured convolutional codes

In this section, we want to generate high rate encoder polynomials. A general way to generate high rate convolutional code is to use a low rate (1/2) encoder and delete some of its parity check bits, and this is called puncture method. Some codeword bits might be punctured according to a deterministic puncture pattern or puncture matrix. The bit error rate may be a little different because of using different puncture patterns. The puncture procedure is shown as Fig. 3.2, assume that a source sequence $(X_0, X_1, ..., X_8, ..., X_{K-1})$ is sent to an encoder of code rate (1/2), and is encoeded to codeword sequence $(X_0, B_0, X_1, B_1, ..., X_8, B_8,\ ..., X_{K-1}, B_{K-1})$. Before modulation, the codeword is stolen some bits according to a puncture pattern. In Fig. 3.2, the puncture pattern is defined

$$P = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

the column number of this matrix represents the puncture period, and element 1 means

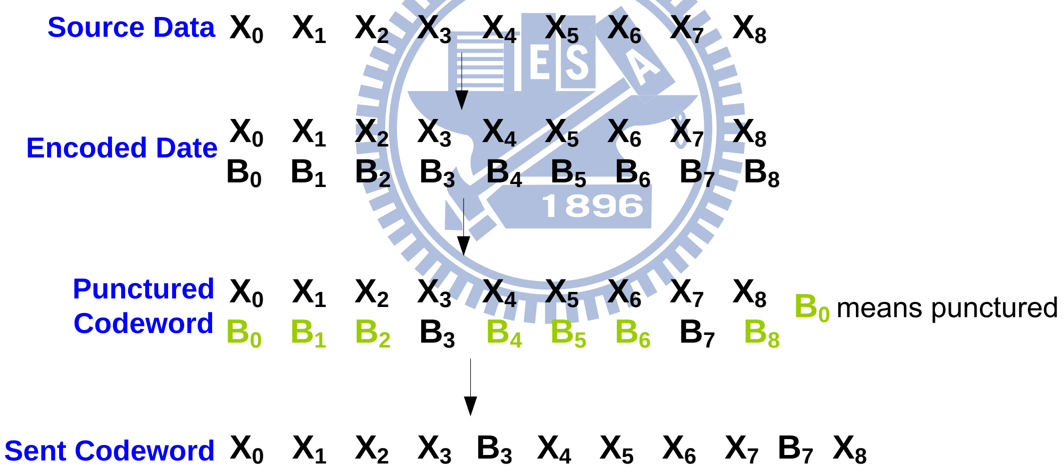
Source Data X_0 X_1 X_2 X_3 X_4 X_5 X_6 X_7 X_8
Encoded Date X_0 X_1 X_2 X_3 X_4 X_5 X_6 X_7 X_8
B_0 B_1 B_2 B_3 B_4 B_5 B_6 B_7 B_8
Punctured Codeword X_0 X_1 X_2 X_3 X_4 X_5 X_6 X_7 X_8
B_0 B_1 B_2 B_3 B_4 B_5 B_6 B_7 B_8
B_0 means punctured
Sent Codeword X_0 X_1 X_2 X_3 B_3 X_4 X_5 X_6 X_7 B_7 X_8

Figure 3.2: Puncture procedure.

that the corresponding codeword bit must be sent whereas element 0 means that corresponding the bit is punctured or stolen. Finally, this puncture procedure increases the code rate from (1/2) to (4/5) and produces anther codeword sequence $(X_0, X_1, X_2, X_3, B_3,$ $...X_8, ..., X_{K-3}, X_{K-2}, X_{K-1}, B_{K-1})$

We take an example, assume an original systematic encoder of code rate 1/2 is :

$$G_{sys}(D) = \begin{pmatrix} 1 & \frac{1+D^2}{1+D+D^2} \end{pmatrix}$$

and using the puncture pattern

$$P = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$$

an equivalent punctured encoder with rate = 2/3 is defined by :

$$G\prime(D) = \begin{pmatrix} 1+D & 1+D & 1 \\ D & 0 & 1+D \end{pmatrix}$$

Hence, the reciprocal dual code can be generated by the polynomial generator matrix:

$$\tilde{H}(D) = \begin{pmatrix} 1+D^2 & 1+D+D^2 & 1+D \end{pmatrix}$$

The relationship of trellis ($G_{sys}(D)$, $G\prime(D)$, $\tilde{H}(D)$) is shown in Fig. 3.3. $G\prime(D)$ trellis is a readix-4 for turbo decoders, however the reciprocal dual code $\tilde{H}(D)$ is a radix-2 architecture. In Fig. 3.4 shows the performance of decoding algorithm using $\tilde{H}(D)$ trellis (Dual-MAP) and $G\prime(D)$ (MAP) trellis. The interleaver length is 400 which results a $(600, 400)$ punctured block code. The result does very make sense due to both decoding algorithm reach the same performance. However, using original puncture trellis is not efficient compared to reciprocal dual trellis.

3.2.1 Apply rate compatible punctured turbo code to WCDMA

RCPC(Rate Compatible Punctured Convolutional) codes are one practical solution to adaptive coding mentioned in [11]. RCPC codes use a single rate-$(1/n)$ convolutional encoder/decoder pair and only to share a puncturing table. RCPT(Rate Compatible Punctured Turbo) code can be used in a similar way a RCPC codes. This section will focus on applying RCPC to turbo code in WCDMA. The used puncture tables are shown in Table 3.1. The "1" or "0" in these tables represents send or punctured. The systematic

Table 3.1: Puncture tables

Table	$PT1$	$PT2$	$PT3$	$PT4$
Systematic	1	11	1111	11111111
Parity 1	1	01	0001	00000001
Parity 2	1	01	0001	00000001
Code rate	1/3	1/2	2/3	4/5

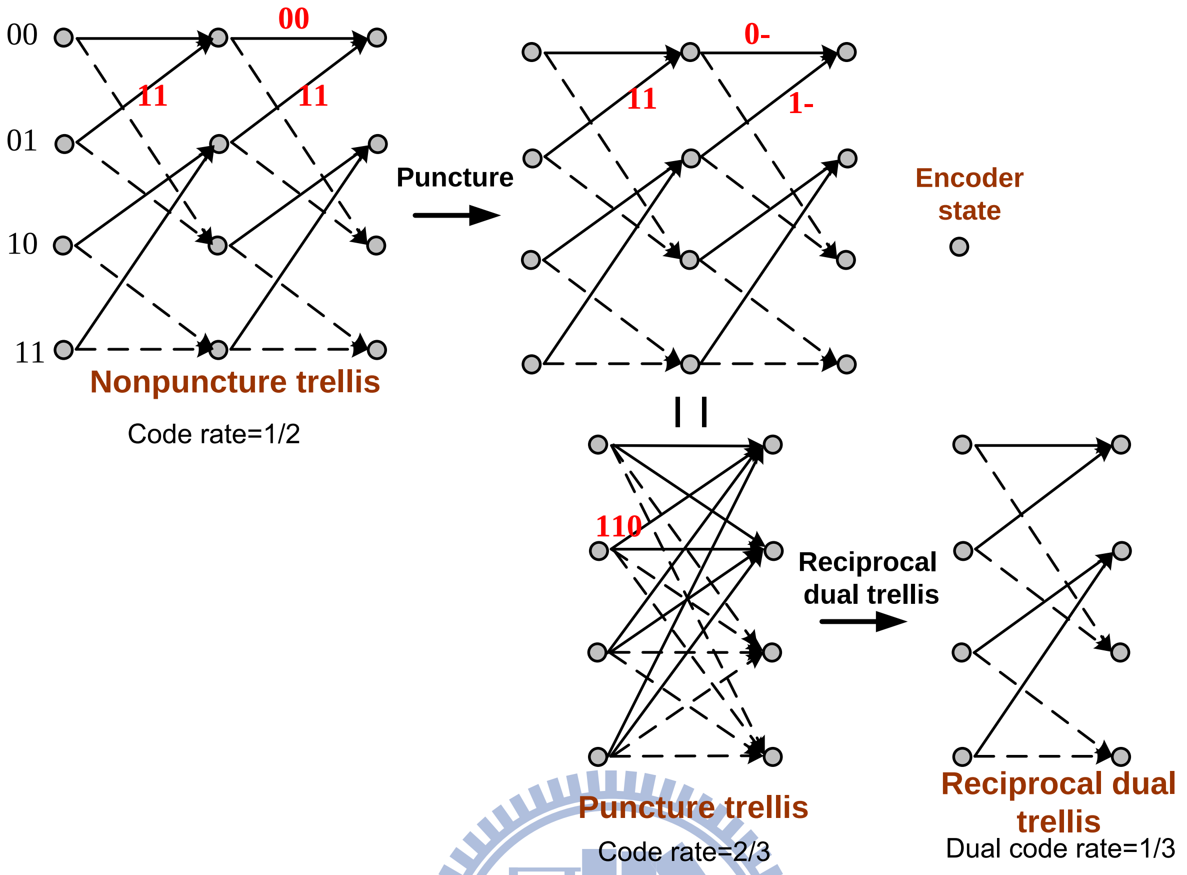

Figure 3.3: Trellis transformation relationship

bits are reserved for all code rates, however, the parity bits are reserved for only some period. The first row is systematic information, the second row is first parity bit, and the third row is the second parity bit. The characteristic of the tables is that codeword of lower code rate must contains codeword of higher rate.

In 3GPP(WCDMA) standard, the scheme of turbo encoder is a Parallel Connected Convolutional Code (PCCC) with 8-state constituent encoders and one interleaver. The nonpuncture code rate of turbo encoder is 1/3. The structure of turbo encoder is shown in Fig. 3.5. The transfer function of the 8-state constituent code for the PCCC is

$$G(D) = \begin{pmatrix} 1 & \frac{1+D+D^3}{1+D^2+D^3} \end{pmatrix}$$

The bit sequence input for a given code block to channel coding by c_0,c_1,c_2,c_3,...,c_{K-1}, where K is the number of bits to encode. After encoding, the bits are denoted by $d_0^{(i)}$, $d_1^{(i)}$, $d_2^{(i)}$, $d_3^{(i)}$,...,$d_{D-1}^{(i)}$, where D is the number of encoded bits per output stream and i indexes the encoder output stream.

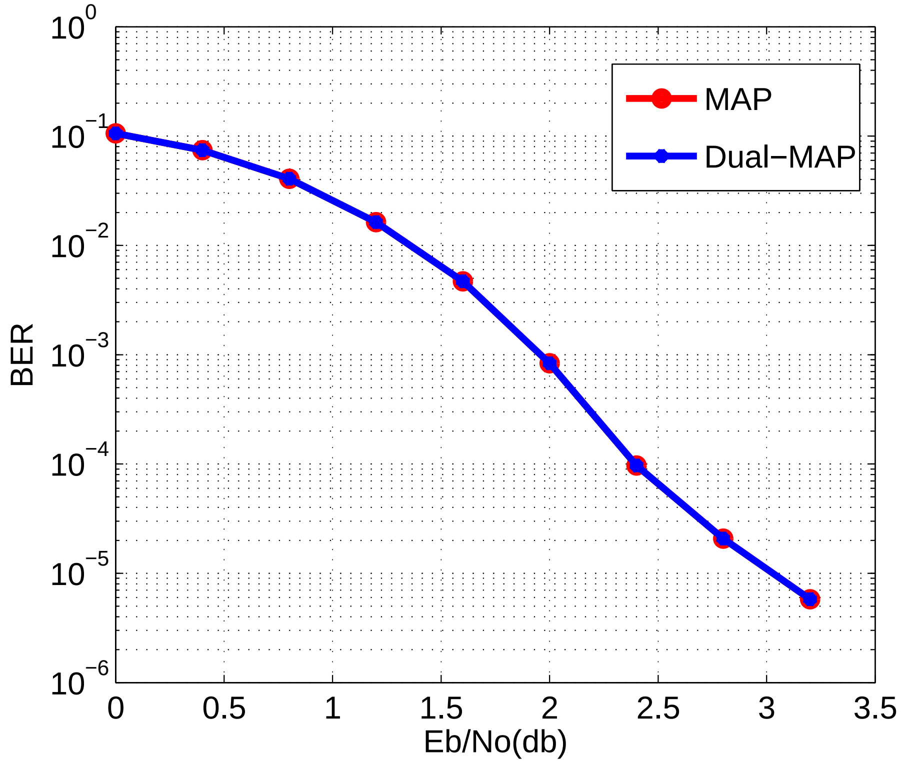

Figure 3.4: Performance of using Dual-MAP and MAP under punctured code

The initial value of the shift registers of the 8-state constituent encoders shall be all zero when starting to encode the input bits. The output from the turbo encoder is $d_k^{(0)} = x_k$,$d_k^{(1)} = z_k$,$d_k^{(2)} = z'_k$ for $k = 0, 1, 2, ..., K-1$. The bits input to the turbo encoder are denoted by c_0,c_1,c_2,c_3,...,c_{K-1}, and the bits output from the first and second encoder are denoted by z_0,z_1,z_2,z_3,...,z_{K-1} andz'_0,z'_1,z'_2,z'_3,...,z'_{K-1}

3.2.2 Decoding with SM and Dual-MAP algorithm

When puncture table $PT1$ is applied, that will produce a nonpunctured generator $G_1(D)$ which is the same as $G(D)$. We define $G_2(D)$, $G_3(D)$,and $G_4(D)$ are punctured generator when applying $PT2$, $PT3$, and $PT4$, respectively. The systematic punctured generator polynomials are :

$$G_1(D) = \begin{pmatrix} 1 & \frac{1+D+D^3}{1+D^2+D^3} \end{pmatrix}$$

$$G_2(D) = \begin{pmatrix} 1 & 0 & \frac{1+D+D^2}{1+D^2+D^3} \\ 0 & 1 & \frac{1+D+D^2+D^3}{1+D^2+D^3} \end{pmatrix}$$

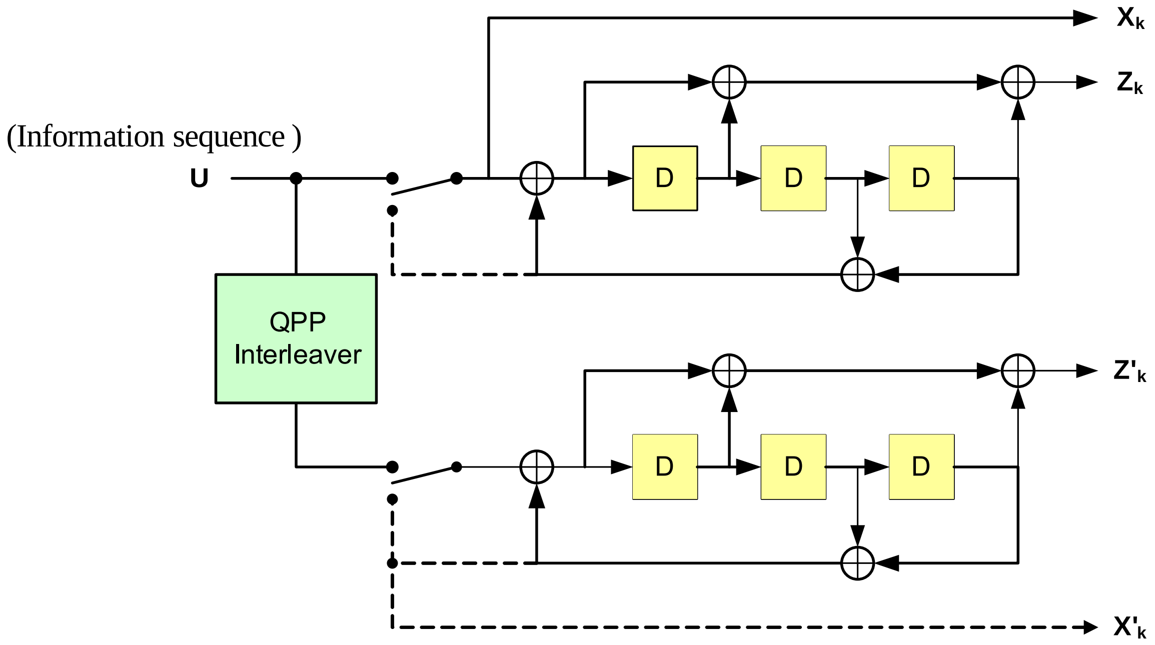

Figure 3.5: Turbo encoder for WCDMA.

$$G_3(D) = \begin{pmatrix} 1 & 0 & 0 & 0 & \frac{1+D^2}{1+D^2+D^3} \\ 0 & 1 & 0 & 0 & \frac{1+D}{1+D^2+D^3} \\ 0 & 0 & 1 & 0 & \frac{1}{1+D^2+D^3} \\ 0 & 0 & 0 & 1 & \frac{1+D^3}{1+D^2+D^3} \end{pmatrix}$$

$$G_4(D) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1+D+D^3}{D+D^2} \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1+D^2+D^3}{D+D^2} \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \frac{D^2+D^3}{D+D^2} \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & \frac{D}{D+D^2} \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \frac{D^2}{D+D^2} \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \frac{D^3}{D+D^2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \frac{D+D^3}{D+D^2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \frac{D+D^2+D^3}{D+D^2} \end{pmatrix}$$

And their corresponding reciprocal dual trellis are generated respectively by $\tilde{H}_1(D)$, $\tilde{H}_2(D)$, $\tilde{H}_3(D)$, and $\tilde{H}_4(D)$:

$$\tilde{H}_1(D) = \begin{pmatrix} 1+D+D^3 & 1+D^2+D^3 \end{pmatrix}$$

$$\tilde{H}_2(D) = \begin{pmatrix} D+D^2+D^3 & 1+D+D^2+D^3 & 1+D+D^3s \end{pmatrix}$$

$$\tilde{H}_3(D) = \begin{pmatrix} D+D^3 & D^2+D^3 & D^3 & 1+D^3 & 1+D+D^3 \end{pmatrix}$$

$$\tilde{H}_4(D) = \begin{pmatrix} D+D^2 & D^3 & D^2 & D & D+D^3 & D+D^2+D^3 & D^2+D^3 & 1+D^2+D^3 & 1+D+D^3 \end{pmatrix}$$

Notice that the row number of these matrix is only one even if their original code rate are higher than 1/2. Here, we use these reciprocal dual trellis as decoding trellis and the BER performance are shown in Fig. 3.6

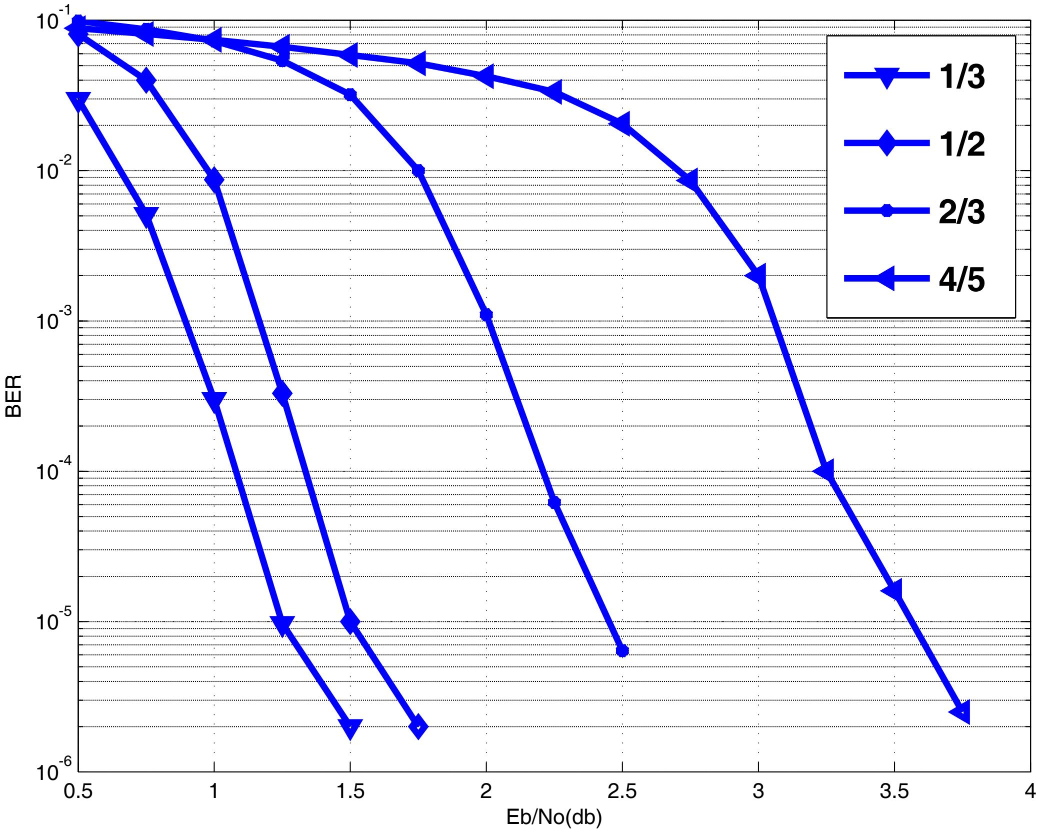

Figure 3.6: Performance of different code rate with block length 2048.

3.3 Performance analysis

In this section, we will present the simulation results and some parameter setting for hardware implementation. All the simulation results are signal-to-noise(SNR) versus BER under BPSK modulation and AWGN channel. In Fig 3.7, there is about 0.05dB loss between the sliding window size of 32 and 64 at the BER= 10^{-5} under the fixed 6 iterations. At code rates 1/3, 1/2, 2/3, and 4/5, the block length of each is 2048, and the performance of different iterations are presented in Fig 3.8. The BER performance of each code rate is almost saturate at 6 iterations. Thererfore, we choose iteration number 6 to our design.

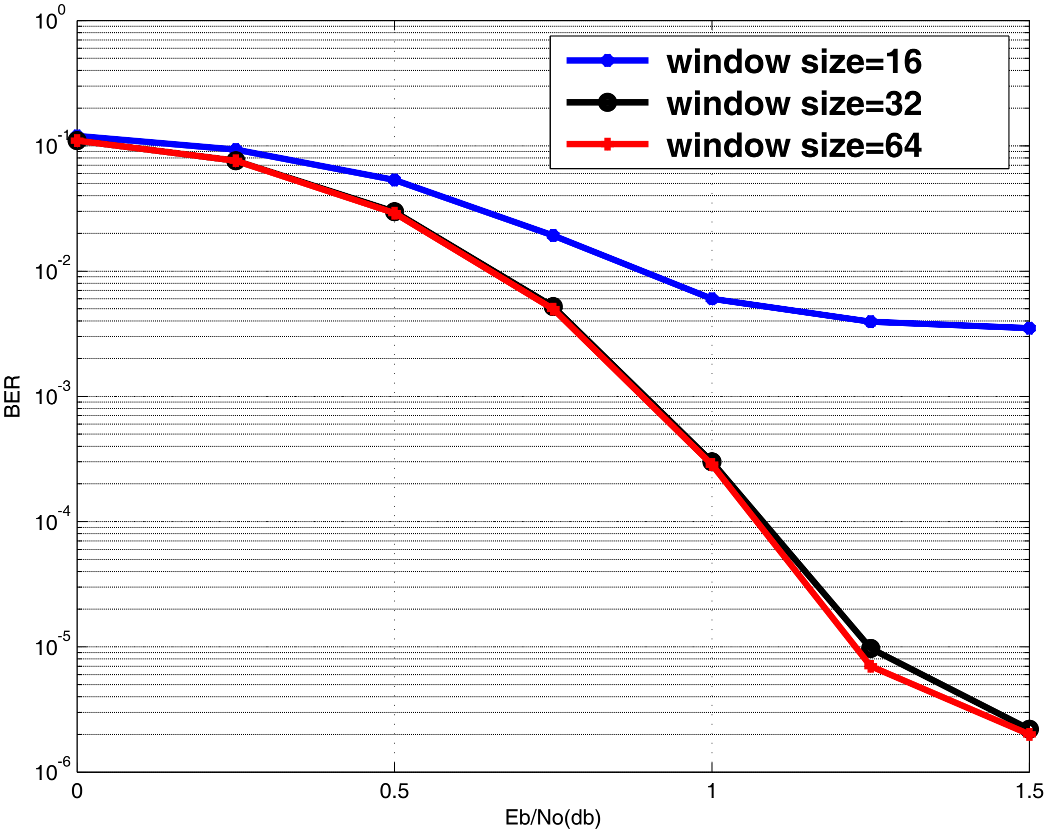

Figure 3.7: Comparison of different sliding window size.

The fixed point representation of the internal variable in the SISO decoder is determined from the received symbol quantization. Fig. 3.9 shows the simulation result with different input symbol quantization under block length 2048, code rate 1/2, window size 32, and 6 iterations. Note that a, b in the figure denotes the quantization scheme where a is the integer part, and b is the fractional part. We can observe that the performance loss of (3.3) is quite small compared with (3.4) format. And we decide that the quantized format 6 bits (3.3) is suitable scheme for our design. In addition, the width of extrinsic information, branch metric, and path metric can be derived and we summarize the fixed representations in Table 3.2.

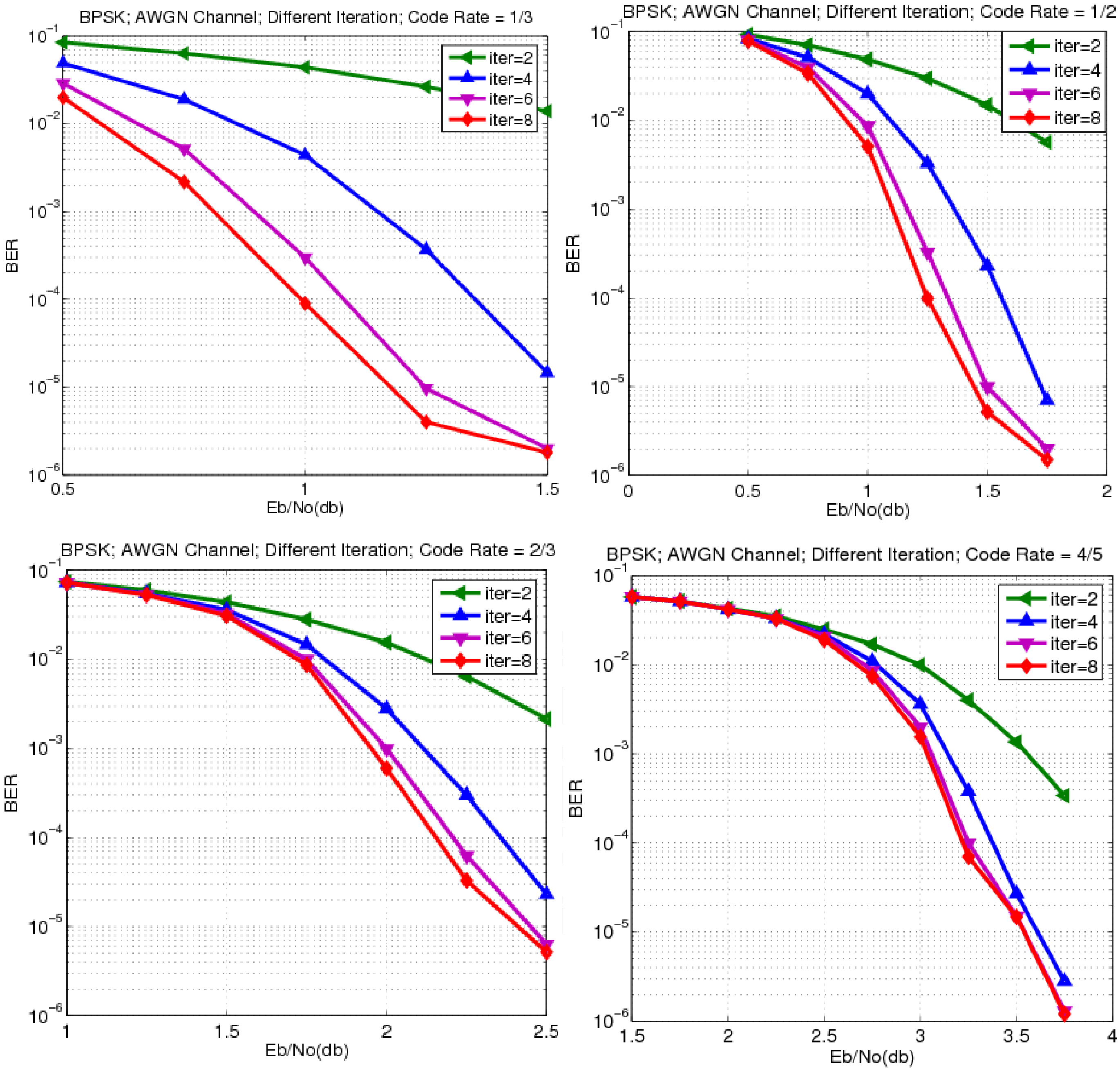

Figure 3.8: Comparison of different iteration.

Table 3.2: Summary of fixed representation in MAP decoder

quantities	Input symbols	Extrinsic information	Branch metrics	State metrics
width	6	10	12	12

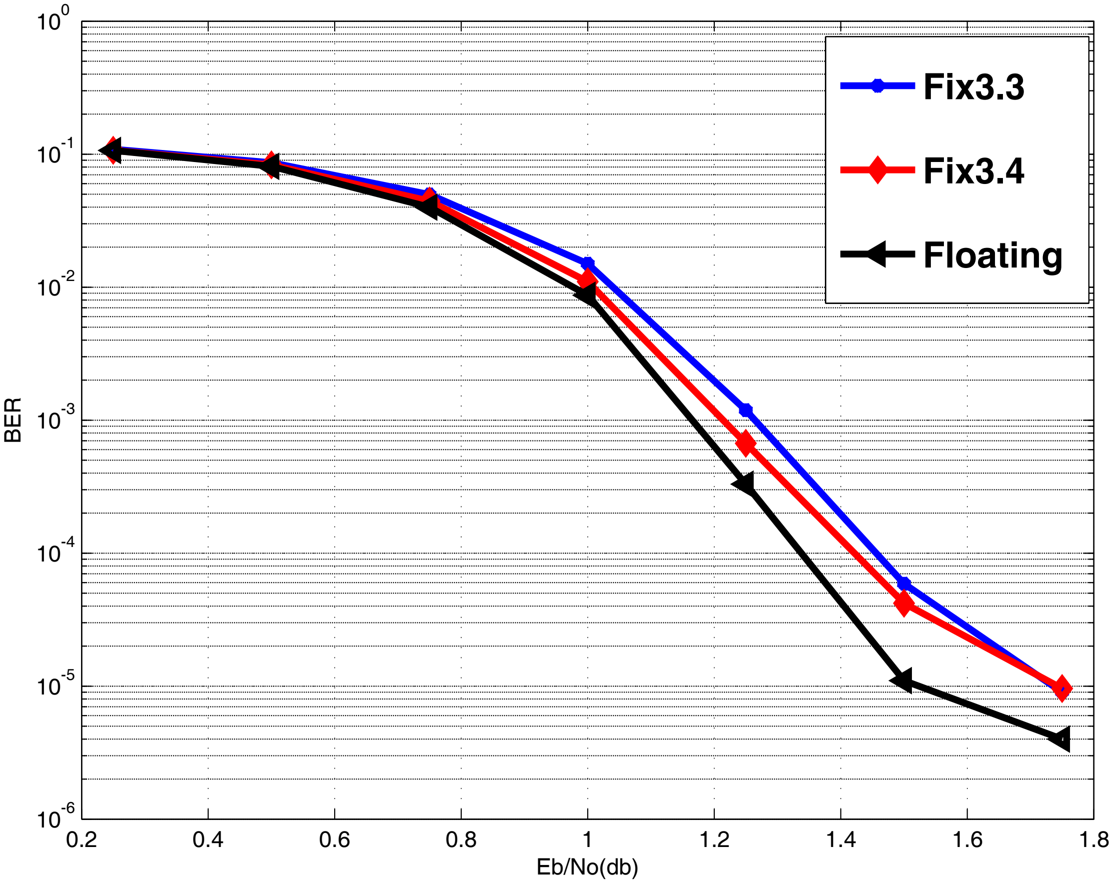

Figure 3.9: Fixed point comparison.

Finally, we talk about the look up table approximation shown in Fig. 3.1. Since the correction term $\log(1 - e^{-|d|})$ can has any value in the range $[\infty, 0]$ if d goes from 0 to ∞, this makes correction term changes severely when d closes to zero. Therefore, segment method of d is a very significant issue. Fig. 3.11 shows performance of two ways to segment the range of d. The simulations are under interleaver length 2048, code rate $= 4/5$, and sliding window size $= 32$ trellis stages. In Fig. 3.10, uniform segment is to divide the range to some equal intervals, and nonuniform segment is to divide the range according to the slope of correction term function. Here, according the performance of bit error rate, we shall choose nonuniform segment method to divide the range of d.

The design parameters and methodology are applied to our decoder. The performances of each code rate are shown in Fig. 3.12, and floating point simulations also are list. The fixed point performance loss is 0.3 to 0.5dB compared with floating point at BER $= 10^{-5}$.

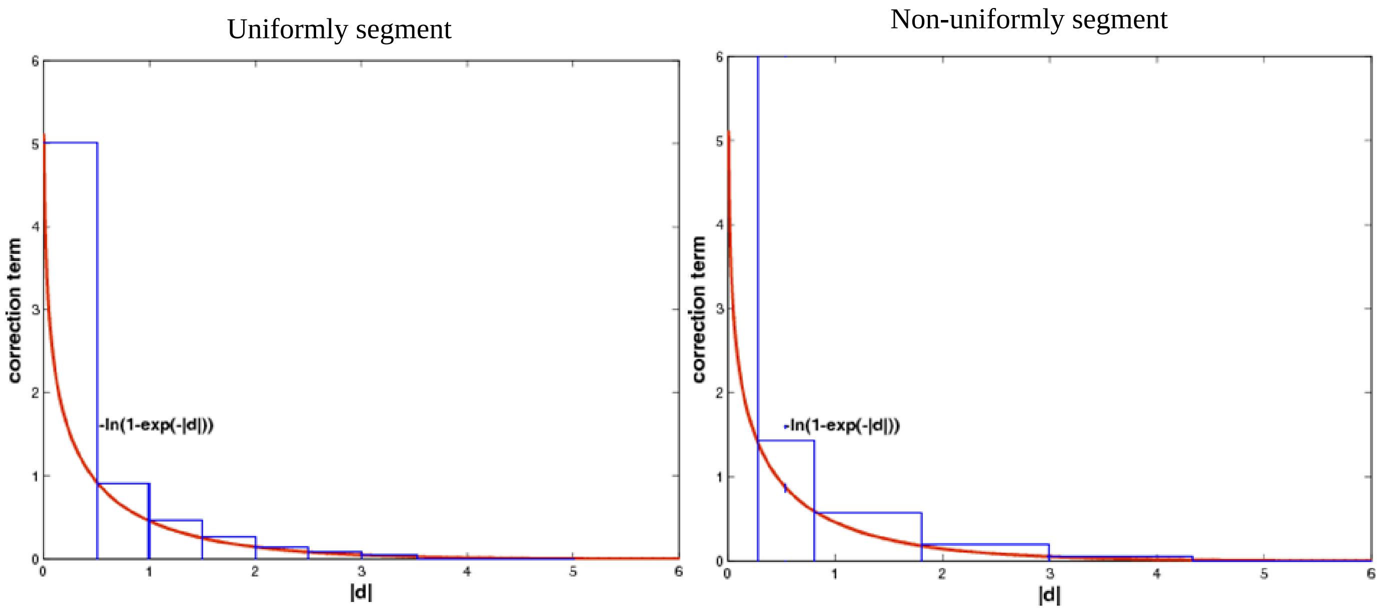

Figure 3.10: Segment method.

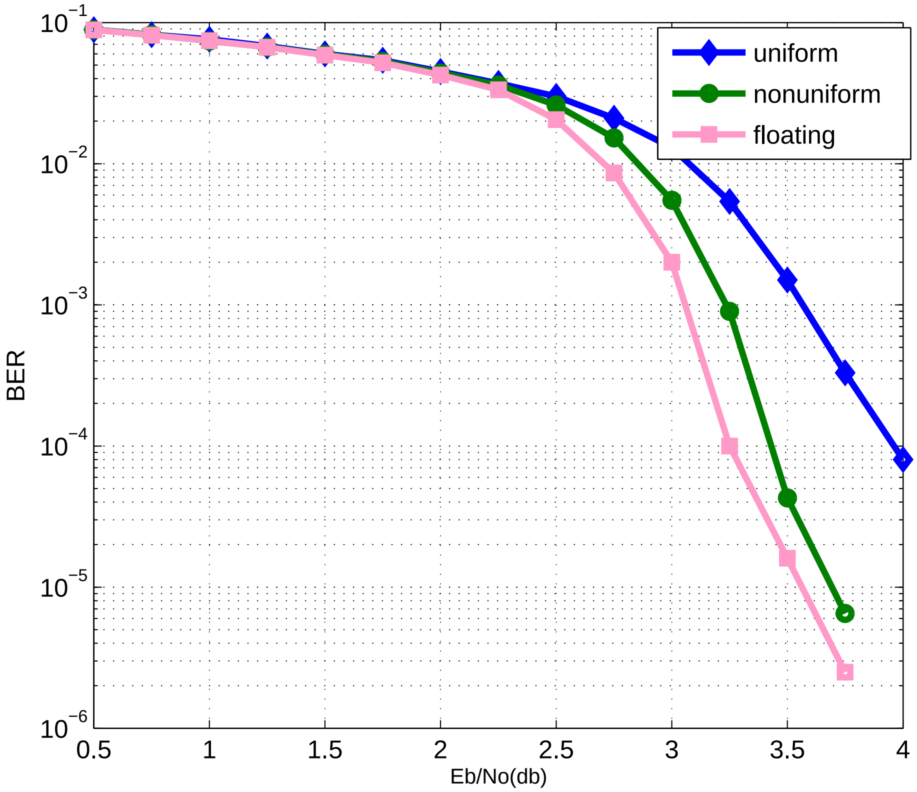

Figure 3.11: Performance comparison of different segment method.

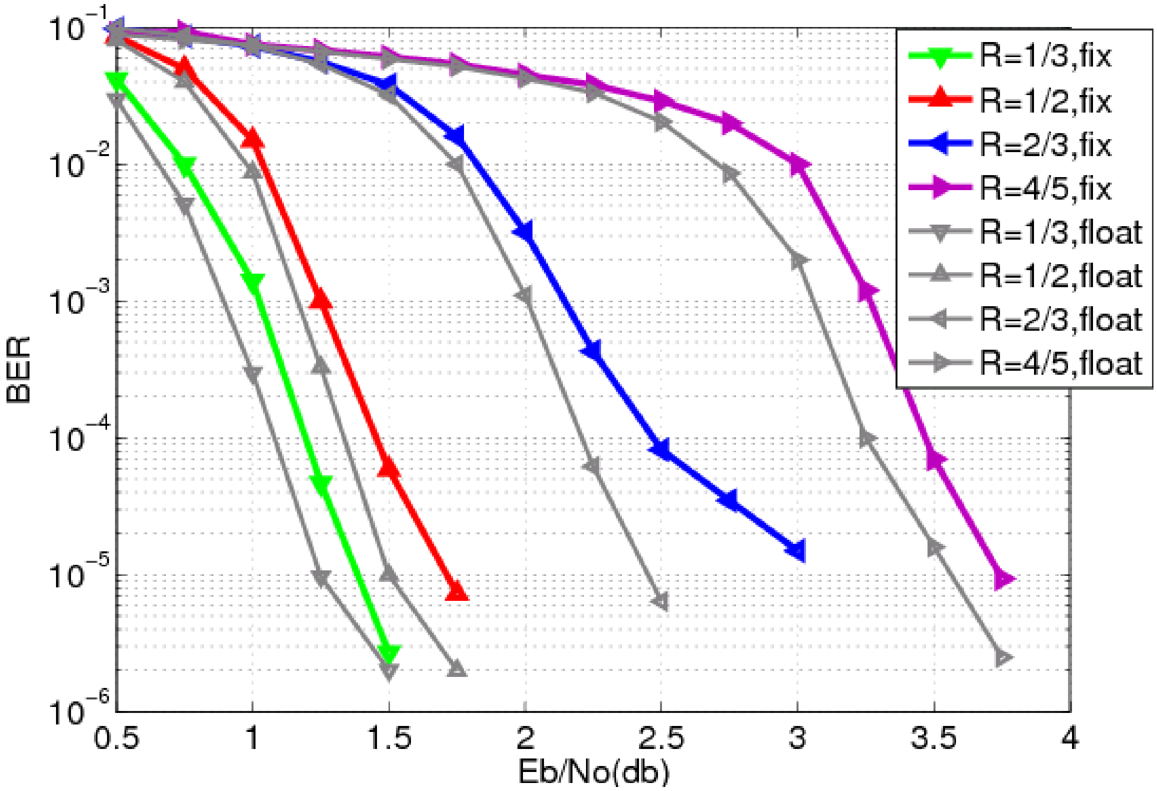

Figure 3.12: Performance of each code rate with design parameters.

Chapter 4

Dual-MAP Turbo Decoder Architecture

According to equation (2.39), there are 1, 2, 4, and 8 extrinsic values can be calculated during a trellis stage when we use $PT1$, $PT2$, $PT3$, and $PT4$ puncture tables respectively. In this chapter, we will disscus the architecture of reciprocal dual trellis turbo decoder.

4.1 Architecture overview

The architecture of proposed turbo decoder using reciprocal dual trellis is shown in Fig. 4.1. The SISO decoder performs Dual-MAP algorithm and outputs extrinsic values. The Input Buffers are used to store the received values from channel. The extrinsic memory stores the extrinsic values from SISO. Under each iteration, the SISO computes the extrinsic values which is a priori value estimation for the next iteration. There are two stages in a iteration. In normal stage, data is read from Input Buffer and extrinsic memory in normal order, and the extrinsic values are witten into the extrinsic memory in interleaved order. In interleaver stage, data is read from Input Buffer and extrinsic memory in interleaved order, and the extrinsic values are witten into the extrinsic memory in normal order. However, by using the reciprocal dual trellis, we can just apply radix-2 branch calculational circuit to achieve the turbo decoder design.

We apply sliding window approach mentioned in [9] to the SISO decoder. In Fig. 4.1, Window BUFs store the input soft values for evaluating α and β. The SMM is the

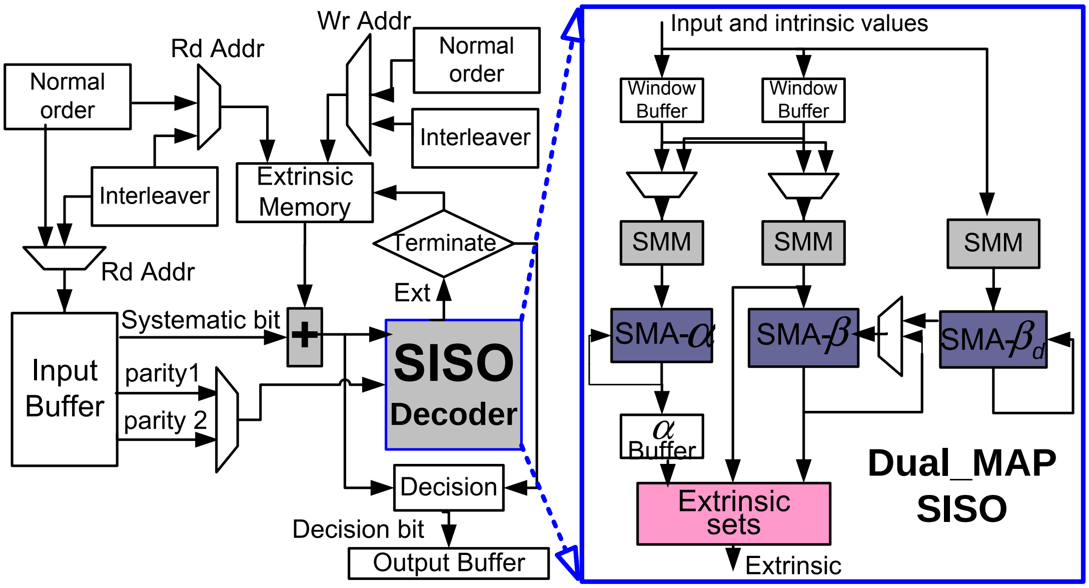

Figure 4.1: Iterative decoding of turbo decoder.

abbreviation of sign magnitude multiplication and calculates the branch metrics. Each SMA is the abbreviation of sign magnitude addition and calculates the path metrics for each recursion. To avoid waiting for the whole codeword for β evaluation, we use SMA-β_d from the end of the next sliding window to evaluate the initial values of β. And the α-Buffer performs the Last-In/First-Out (LIFO) for the reversing output of α.

Fig. 4.2 is the decoding schedule of the SISO decoder. At the first time interval T_0, the input values are written into a Window BUF1 in reversing order. At the second time interval T_1, the second input data(W_1) are written into the other Window BUF2 also in reversing order and SMA-β_d calculates β_d to evaluate the initial conditions of β. Simultaneously, the SMA-α utilizes the data in W_0 to compute α reversely and saves the results in the α-Buffer. Finally, the first window data will be read at T_2 interval for β calculation and the third window data(W_2) will be written into Window BUF1 simultaneously. When the SMA-β unit starts to calculate, the extrinsic values can also be calculated by Extrinsic sets. As a result, the latency of the SISO decoder is about two sliding window size.

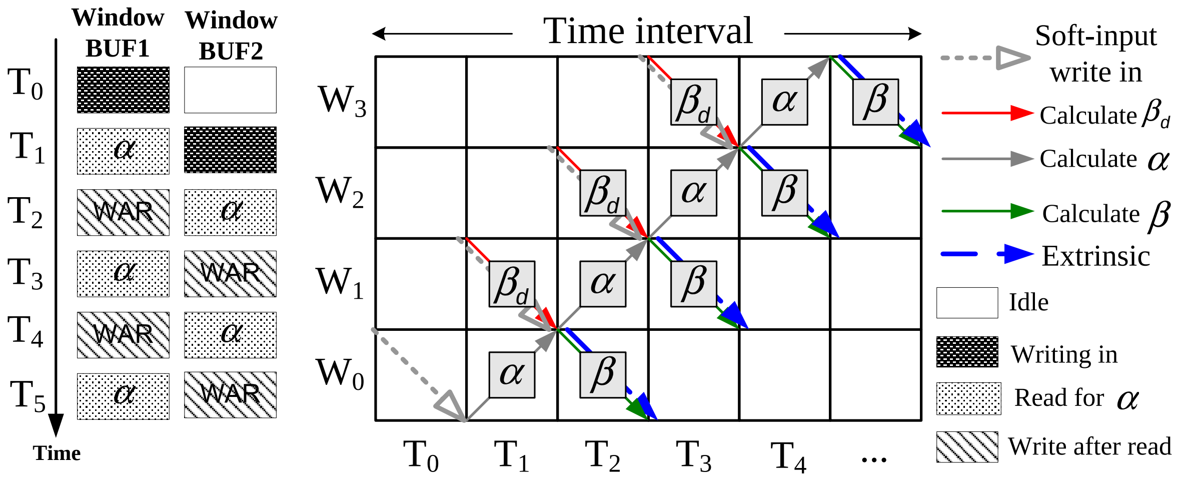

Figure 4.2: Decoding schedule of Dual-MAP decoder.

4.2 Dual-MAP decoder

4.2.1 SMM unit

The SMM in Fig.4.1 performs the multiplication of sign magnitude algorithm and the circuit is illustrated in Fig.4.3. The output of ”*Sign*” is calculated by the XOR logical operation of input signs and the ”*Mag*” is the summation of input magnitude parts.

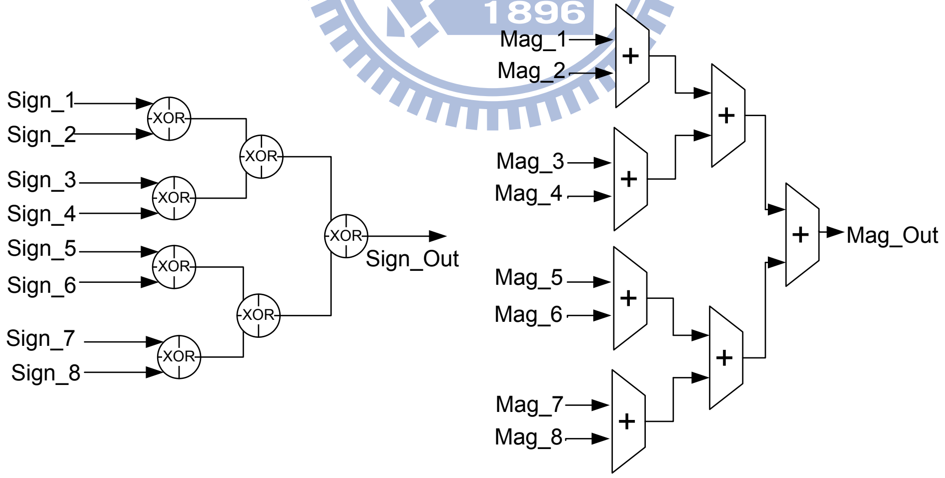

Figure 4.3: Architecture of SMM unit.

4.2.2 SMA unit

Each SMA unit used in recursion calculation is illustrated in Fig. 4.4, and is mainly used to perform sign magnitude addition. In Fig. 4.4, two sets of AFA(α) and Gamma(γ) represented in sign magnitude type are inputs of SMA unit. $Mag1$ is the magnitude part and $S1$ is the sign part of Sum*1 output. The Sign bit will select the minimum between $Mag1$ and $Mag2$, and the output sign is also selected by Sign bit. The difference(ABS) between $Mag1$ and $Mag2$ is sent to the look up tables. And the look up tables(LUT) are used to approximate functions $\log(1+e^{-|A-B|})$ and $\log(1-e^{-|A-B|})$ as described in 3.1.1. Finally, the adder calculates the output of magnitude. To avoid overflow in finite datawidth, we adopts the modulo normalization of path metric [12] in our design.

Now, we further analysis radix-4 Log MAP recursion unit. According to [13], radix-4 recursion unit has four candidates to select. As we mentioned in equations (2.23), the function of radix-4 Log MAP recursion can be written as (4.1), and directly implement this function into architecture Fig.4.5.

$$\begin{aligned}\log(e^{\delta_1}+e^{\delta_2}+e^{\delta_3}+e^{\delta_4}) &= \max{}^*(\delta_1,\delta_2,\delta_3,\delta_4)\\ &= \max{}^*(\max{}^*(\delta_1,\delta_2),\max{}^*(\delta_3,\delta_4))\end{aligned} \tag{4.1}$$

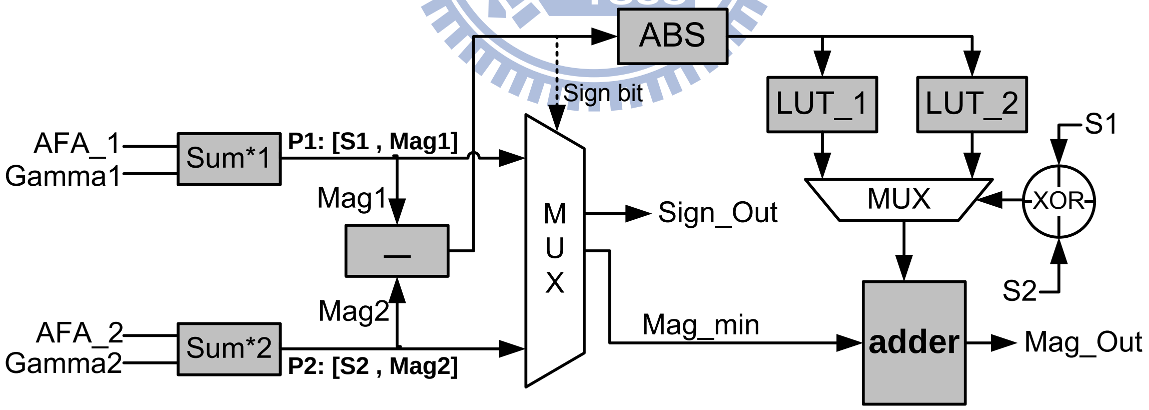

Figure 4.4: Architecture of SMA unit.

In Log-MAP algorithm, for a $(n, n-1)$ code rate RSC, there will be a radix-2 trellis at the decoder for $n = 2$. For higher code rate, assume $(n = 3)$, and there will be a radix-4 trellis at the decoder. Notice that, in radix-2 Log MAP architecture, the hardware

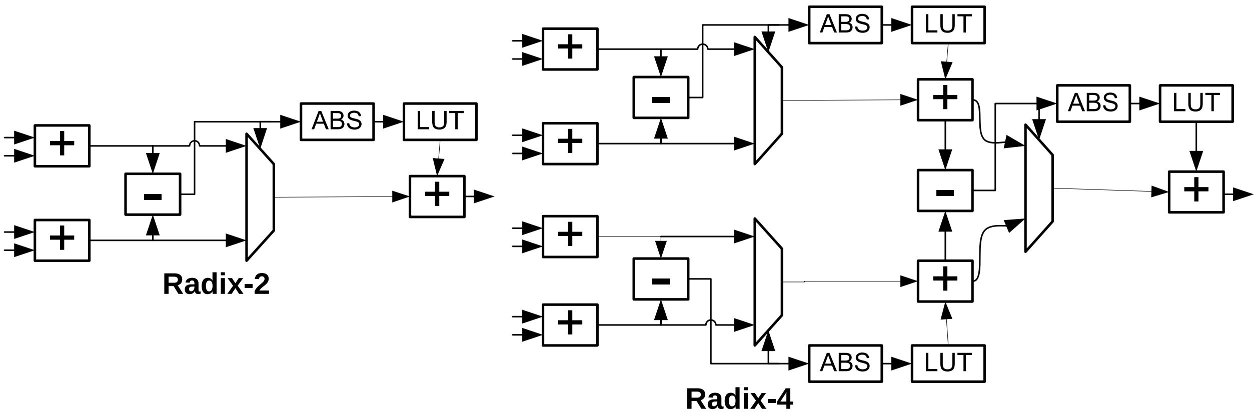

Figure 4.5: Radix-2 and radix-4 Log MAP recursion units.

complexity is lesse than SMA unit. However, in radix-4 situation, the number of the computational units are much more than SMA unit.

4.2.3 Extrinsic unit

An architecture of Extrinsic unit is proposed according to (3.1.2) and is illustrated in Fig. 4.6. The inputs are forward, backward, branch metrics, and the bit metric (g_j) which is at the position that we intend to decode. The PMN(Path Metric Network) unit is the connection between path metrics and two HMP(Hierarchical Min Processor) according to the transition bit at decoding index j. The connections of PMN various with decoding index j, and Fig. 4.6 shows an example. min^* in HMP performs the sign magnitude addition and both HMP perform the summation of (3.7) for $b = 0$ and $b = 1$. Finally, a look up table is required to perform (3.10) function. The output of look up table is the extrinsic value. Parallel Extrinsic units can be used to decode all extrinsic values of the trellis stage to achieve high throughput.

4.3 Throughput evaluation

In MAP decoding algorithm, the decoding process through pre-decoding round and post-decoding round is called one iteration. That is why the denominator of equation (4.2) are two times of iterations. Based on all factors mentioned above, the throughput in our design can be evaluated eventually. The throughput of our design will show in the

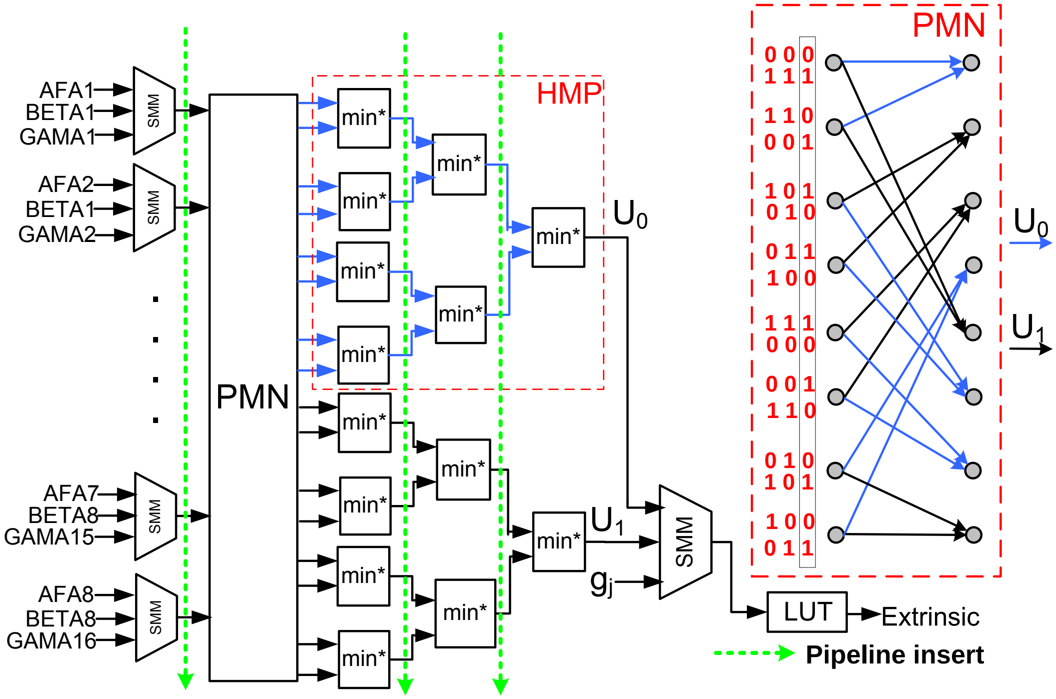

Figure 4.6: Architecture of Extrinsic unit.

next chapter in Table 5.1.

$$
\begin{aligned}
Throughput(^{bits}/_{s}) &= \frac{Block\ Length}{Needed\ cycles} \times clock\ rate \\
&= \frac{f \times Block\ Length}{2 \times iteration\#(Block\ Length/Parallel + 2 \times Sliding\ Window\ Size + C)}
\end{aligned}
\tag{4.2}
$$

- **f** : *Operation frequency*
- **Parallel** : *Parallel Extrinsic units numbers*
- **C** : *some latency in decoder*$(\lll Block\ Length)$
- **iteration** : *number of iterations*

Chapter 5

Implementation Results

In this chapter, the implementation results of each code rate decoders will be shown. Furthermore, we integrate the four different code rate decoders into one, and the implementation results are also shown in this chapter.

5.1 Implementation of Dual-MAP turbo decoders

The throughput of each SISO decoder is different because the parallel Extrinsic untis are applied. Implemented by UMC 90nm 1P9M CMOS technology, the operation frequency of each SISO decoder can reach 200MHz in synthesis. And the area of each SISO decoder including the parallel Extrinsic units are presented here with the unit of gate counts. Table 5.1. shows the results in all cases.

Table 5.1: Summary of synthesis result

Code Rate	Component Code Rate	SISO Area	EXTRINSIC (sets)	Throughput (MS/s)
$R = 1/3$	$R_c = 1/2$	$88K$	1	16
$R = 1/2$	$R_c = 2/3$	$107K$	2	31
$R = 2/3$	$R_c = 4/5$	$178K$	4	57
$R = 4/5$	$R_c = 8/9$	$270K$	8	101

Finally, we can mainly use the architecture of code rate 4/5 decoder to achieve the multiple code-rate decoder. The total gate counts of this decoder are about 850K. The area distribution of modules in the decoder are shown in pie chart 5.1. The logic gate counts of this decoder are about 370K while the storage units such as buffers used for input, extrinsic, and output occupy about 480K gate counts.

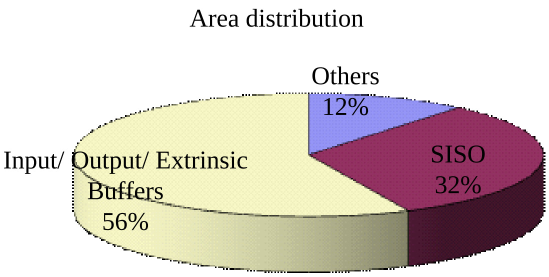

Figure 5.1: Area distributin of modules in turbo decoder.

5.1.1 Comparison of different high radix MAP decoders

In [14], a MAP decoder is implemented using radix-4×4 MAX-LOG-MAP algorithm. We compare the decoder in [14] with our design of code rate $R = 2/3$ because there are the same number of extrinsic value delivered from each SISO at the same time. The comparison is summarized in table 5.2 briefly. However, the total gate counts of this MAX-LOG-MAP decoder are still larger than our design.

Table 5.2: Compare with radix-4×4 MAX-LOG-MAP circuits

Component	Proposed	Ref. [14]
Technology	90nm	13um
Operation Frequency(MHz)	200	238(APR)
Sliding Window	32	20
Total SISO Gate Counts	**178K**	**220K**
Decoding Trellis	Reciprocal dual trellis	Original trellis

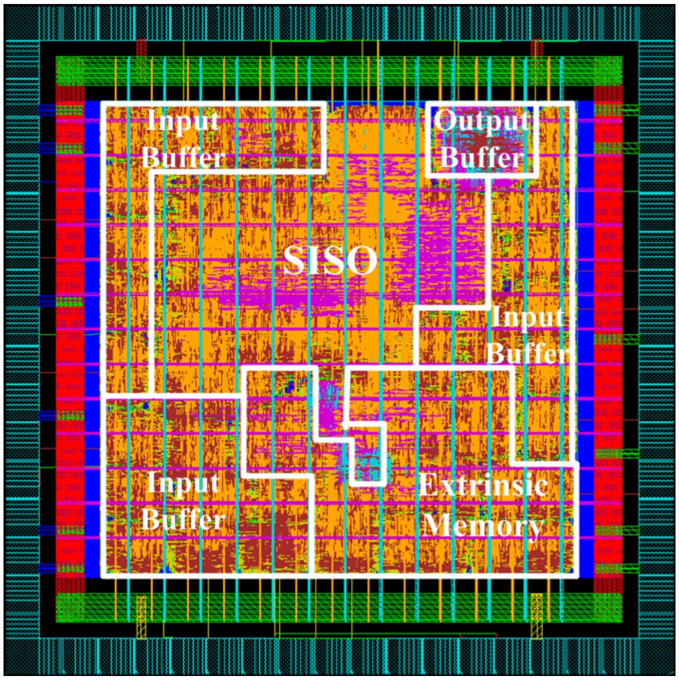

Figure 5.2: Layout photo of multiple code-rate turbo decoder.

5.1.2 Layout specification

The specification of the multiple code-rate turbo decoder is given in Table 5.3, which is implemented through the cell-based design flow. This chip has been implemented in a 0.9V, UMC 90nm 1P9M CMOS technology, and the layout view is shown in Fig. 5.2. The core size of this chip is 3.78mm^2 in 90nm process. The total gate counts are 850K while the logic gate counts are about 370K and the chip core density is about 66%. After static timing analysis and post layout simulation, the operating frequency can achieve up to 200MHz and the throughput are 16, 31, 57, and 101Mb/s at code rate 1/3, 1/2, 2/3, and 4/5, respectively. The power consumption operated at 200MHz is 80mW at code rate 4/5 and 82mW at code rate 1/3 with 0.9V supply voltage.

Besides, the power consumption estimated by the post-layout simulation and can be converted to energy efficiency. The energy efficiency is defined as the average energy consumed per bit within each decoding iteration (nJ/bit/iter). For our decoder, the energy efficiency at code rate 4/5 can be calculated as

$$\frac{80\text{mW}}{6 \times 101\text{Mb/s}} = 0.132\text{nJ/bit/iter}.$$

Table 5.3: Proposed turbo decoder chip summary

Technology	UMC 90-nm 1P9M CMOS
Block length	2048
Iteration	6
Quantization of input	6 bits (3.3)
SISO decoder algorithm	Sign Magnitude and Dual-MAP
Code polynomial	$[\ 1\ \frac{1+D+D^3}{1+D^2+D^3}\ ]$
code rate	1/3, 1/2, 2/3, 4/5
Core area/Gate count	3.85mm^2 / 850K
Throughput	16, 31, 57, and 101Mb/s
Supply voltage	0.9V
Clock rate	200MHz
Utilization	66%
Power consumption	80mW(@0.9V, 200MHz, and code rate=4/5)
Energy efficiency nJ/bit/iter	0.132

5.2 Comparison of different turbo decoders

Table 5.4 lists the comparison of the proposed design with other published papers. The main topic of our works is to investigate benefits from using reciprocal dual trellis. Therefore, we compare the area of SISO, control units, and some calculational circuit which not include storage unit such as SRAM.

Table 5.4: Comparison of different turbo decoders

	Proposed	Ref. [15]	Ref. [13]	Ref. [16]
Technology	90nm	0.25μm	0.18μm	0.18μm
Block length	2048	N/A	5114	5114
Code Rate	1/3, 1/2, 2/3, 4/5	1/3	1/3	1/3
SISO Algorithm	Sign Magnitude Dual-MAP	Radix-2 LOG-MAP	Radix-4 LOG-MAP	Radix-2 LOG-MAP
Iteration	6	8	6	10
Clock frequency(MHz)	200	95	145	89
Core area (mm^2)	3.78	N/A	14.5(3.63*)	9(2.25*)
Logic gate counts**	370K	24.8K (SISO)	410K	85K
SRAM(kb)	58	2.25	450	239***
Power consumption(mW)	82 (@1/3 rate)	N/A	1450(262*)	292(73*)
Energy efficiency (nJ/bit/iter)	0.854 (@1/3 rate)	1.231(0.16*)	10(2.5*)	14.6(3.65*)
Throughput(Mb/s)	16, 31, 57, 101	2(at least)	24	2
Status	APR	Synthesis	CHIP	CHIP

*: Scale to 90nm.

**: Gate counts of SISO, control units, and some calculational circuits.

***: Include Data Fetch circuits, input window buffer, and path metric memory.

Chapter 6

Conclusion and Future Works

6.1 Conclusion

In this thesis, we propose a multiple code-rate turbo decoder which uses the reciprocal dual trellis. The sign magnitude representation is used to reduce the hardware complexity and the performance loss is 0.3-0.5dB while compared with floating point simulation at BER $= 10^{-5}$. In SISO decoder design, the input sequence order is reversed to eliminate the β_d input buffer. The core of this turbo decoder is 3.78mm^2 in UMC 90-nm 1P9M CMOS which countains 370K logic gates and 58Kb storage units. The maximum code rate is 4/5 and the throughput can reach 101Mb/s with 200MHz operation frequency. The power consumption at 0.9V voltage is 80mW at rate 4/5 and 82mW at rate 1/3.

Table 5.1 reveals that there is a moderate increase of logic gates as code rate rises. The gate counts of code rate 4/5 design are only 270K. Hence, we consider that reciprocal dual trellis decoding algorithm is preferable to high code rate turbo decoder design rather than high radix trellis structure.

6.2 Future works

In our design, there are still several issues to be concerned. First, the lookup table (LUT) used in calculating forward, backward, β_d, and especially in extrinsic units which occupy considerable area. Though the reciprocal dual trellis reduces the number of branches, the LUT size may decrease the benefit of the reciprocal dual trellis. Second, the correction term $-\log(1-e^{-|d|})$ might have positive values, and this will result a large datawidth used in combinational circuits to maintain the performance. Third, the throughput of lower code rate might be improved by exploiting idle modules.

Bibliography

[1] C. R. P. Hartmann and L. D. Rudolph, “An optimum symbol by symbol decoding rule for linear codes,” *IEEE Trans. Inform. Theory*, vol. IT-22, pp. 514–517, Sept. 1974.

[2] J. Sun and O. Y. Takeshita, “Interleavers for turbo codes using permutation polynomials over interger rings,” *IEEE Trans. Inform. Theory*, vol. 51, no. 1, pp. 101–119, Jan. 2005.

[3] J. H. Andersen, “‘turbo’ coding for deep space applications,” in *IEEE International Symposium on Inform. Theory*, no. 17-22, Sep. 1995, p. 36.

[4] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, “Optimal decoding of linear codes for minimizing symbol,” *IEEE Trans. Inform. Theory*, vol. IT-20, pp. 284–287, Mar. 1974.

[5] J. Hagenauer and P. Hoeher, “A viterbi algorithm with soft-decision output and its applications,” in *IEEE CLOBE-COM*, Dallas, Nov. 1989, pp. 47.1.1–47.1.7.

[6] Robertson, E. Villebrun, and P. Hoeher, “A comparison of optimal and sub-optimal decoding algorithm,” in *Proc. IEEE Int. Conf. Commun.*, 1995, pp. 1009–1013.

[7] S. Riedel, “Map decoding of convolutional codes using reciprocal dual codes,” *IEEE Trans. Inform. Theory*, vol. 44, no. 3, pp. 1176–1187, MAY. 1998.

[8] J. A. Erfanian, S. Pasupathy, and G. Gulak, “Reduced complexity symbol detectors with parallel structures for ISI channels,” *IEEE Trans. Commun.*, vol. 42, no. 2/3/4, pp. 1261–1271, Feb./Mar./Apr. 1994.

[9] S. A. Barbulescu, "Sliding window and interleaver design," *IEEE Electronics letters*, vol. 37, no. 21, pp. 1299–1300, Oct. 2001.

[10] S. S and S. S. Pietrobon, "Log domain implementation of the dual-app algorithm," *Australian Commun. Theory Workshop*, pp. 100–104, Feb. 2005.

[11] J. Hagenauer, "Rate-compatible punctured convolutional codes (rcpc codes) and their applications," *IEEE Trans. Commun.*, vol. 36, no. 4, pp. 389–400, Apr. 1988.

[12] C. Shung, P. Siegel, G. Ungerboeck, and H. Thapar, "VLSI architertures for metric normalization in the Viterbi algorithm," in *Int. Conf. Communications*, vol. 4, Atlanta, CA, Apr. 1990, pp. 1723–1728.

[13] M. Bickerstaff, L. Davis, C. Thomas, D. Garrett, and C. Nicol, "A 24mb/s radix-4 log map turbo decoder for 3gpp-hsdpa mobile wireless," in *ISSCC Dig. Tech. Papers*, Feb. 2003, pp. 150–151.

[14] C. H. Tang, C. C. Wong, C. L. Chen, and C. C. Lin, "A 952Ms/s Max-Log MAP decoder chip using radix-4x4 ACS architecture," in *IEEE Asian Solid-State Circuits Conference (A-SSCC)*, Nov 2006, pp. 79–82.

[15] D. S. Lee and I. C. Park, "Low-power log-map turbo decoding based on reduced metric memory access," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 53, no. 6, pp. 1244–1253, June 2005.

[16] M. Bickerstaff, "A unified turbo/viterbi channel decoder for 3gpp mobile wireless in 18um cmos," in *ISSCC Dig. Tech. Papers*, Feb. 2002, pp. 124–125.

作者簡歷

姓名：林振揚

出生地：台灣 台中縣

出生日期：1983.10.22

學歷：1999.9~2002.6 省立台中第一高級中學
2003.9~2007.6 國立交通大學 電機與控制工程學系 學士
2007.9~2009.7 國立交通大學 電子研究所 系統組 碩士