

國立交通大學

電信工程學系

碩士論文



基於索引層級的疊代訊源通道解碼機制

**Index-Based Iterative Source-Channel Decoding**

研究生：潘彥璋

指導教授：張文輝 博士

中華民國九十七年六月

基於索引層級的疊代訊源通道解碼機制

## Index-Based Iterative Source-Channel Decoding

研究生：潘彥璋

Student：Yen-Chang Pan

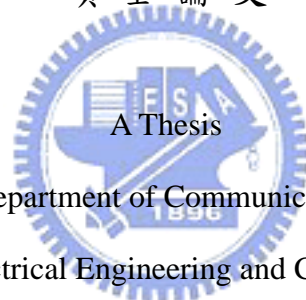
指導教授：張文輝

Advisor：Wen-Whei Chang

國立交通大學

電信工程學系

碩士論文



Submitted to Department of Communication Engineering

College of Electrical Engineering and Computer Science

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of

Master

in

Communication Engineering

June 2008

Hsinchu, Taiwan, Republic of China

中華民國九十七年六月

# 基於索引層級的疊代訊源通道解碼機制

學生：潘彥璋

指導教授：張文輝 博士

國立交通大學電信工程學系碩士班

## 中文摘要

在實際的數位通訊系統，訊源編碼器的設計受限於複雜度與延遲需求，其編碼輸出參數仍有殘餘冗息存在。為了加強系統的強健性，我們在合併訊源通道解碼平台下，引入渦輪原則並據以設計一索引層級的疊代演算法。核心元件包括軟性輸出通道解碼器、軟性位元訊源解碼器以及索引交錯器三個工作單元。其關鍵在於利用訊源殘餘冗息和通道編碼提供的保護位元，在不同解碼器之間逐次交換外源訊息以累積更多可靠的判定內容。通道解碼器根據分段柵狀碼圖而發展索引層級的 BCJR 演算法，讓外源訊息的交換得以提升至更具效率的索引層級。模擬結果顯示，索引層級疊代解碼演算法的強健效能明顯優於位元層級的疊代解碼演算法。

# **Index-Based Iterative Source-Channel Decoding**

Student: Yen-Chang Pan      Advisor: Dr. Wen-Whei Chang

Department of Communication Engineering

National Chiao Tung University

## **Abstract**

This thesis presents a new method of iterative source-channel decoding (ISCD) technique. The ISCD consists of a channel decoder followed by a source decoder, which are connected with a feedback loop. The main attraction of ISCD is that reliability gains resulting from both artificial and source residual redundancy can be exchanged iteratively in a turbo-like process. The extrinsic information produced by each decoding algorithm is used to strengthen the a priori information of another decoder. However, conventional ISCD system computes the extrinsic information in bit-level and therefore the correlation of source parameters cannot be fully utilized. A new index-based ISCD is presented which consists of a softbit source decoder (SBSD) and a channel decoder based on the modified BCJR algorithm using sectionalized code-trellis. The index-based ISCD does not produce bit-level information in its decoding process and therefore improves the way of utilizing the source correlation. Simulation results show that significant improvements of error robustness are achieved by ISCD compared to conventional concatenated decoding schemes. Furthermore, the index-based ISCD outperforms the bit-based ISCD under all channel-SNR conditions.

# Acknowledgements

I would like to acknowledge my advisor, professor Wen-Whei Chang, for his valuable guidance throughout my research. I would like to thank the colleagues in Speech Communication Lab. and all my friends in NCTU for their helpful discussions and suggestions.

I would also like to thank my family for their encouragement and financial support.



# Contents

<b>Abstract (in Chinese)</b> .....	<b>i</b>
<b>Abstract</b> .....	<b>ii</b>
<b>Acknowledgement</b> .....	<b>iii</b>
<b>Contents</b> .....	<b>iv</b>
<b>List of Tables</b> .....	<b>vi</b>
<b>List of Figures</b> .....	<b>vii</b>
<b>Chapter 1 Introduction</b> .....	<b>1</b>
<b>Chapter 2 Bit-Based Source-Channel Decoding</b> .....	<b>6</b>
2.1 System Implementation.....	6
2.2 BCJR Algorithm for Channel Decoding .....	9
2.2 Softbit Source Decoding .....	12
<b>Chapter 3 Index-Based Source-Channel Decoding</b> .....	<b>15</b>
3.1 System Implementation.....	15
3.2 Modified BCJR Algorithm.....	18
3.3 Softbit Source Decoding .....	21
<b>Chapter 4 Index-Based ISCD Algorithm in Log Domain</b> .....	<b>24</b>
4.1 Log-Likelihood Ratio .....	24
4.2 Modified BCJR Algorithm in Log Domain .....	25
4.3 Softbit Source Decoding in Log Domain .....	27
<b>Chapter 5 Simulation Results</b> .....	<b>30</b>
5.1 Experimental Setup .....	30

5.2	Performance of Bit-Based ISCD .....	31
5.3	Performance of Index-Based ISCD .....	32
5.4	Comparison of Index-Based and Bit-Based ISCD .....	35
5.5	Complexity Analysis of Index-Based ISCD .....	39
<b>Chapter 6</b>	<b>Conclusions.....</b>	<b>41</b>
<b>Appendix A:</b>	<b>Derivation of Formula (4.6).....</b>	<b>43</b>
<b>Appendix B:</b>	<b>Interpolated-SBSD .....</b>	<b>45</b>
<b>Bibliography</b>	<b>.....</b>	<b>49</b>



# List of Tables

5.1	Performance of bit-based ISCD .....	32
5.2	Performance of index-based ISCD with uniform initialization (UI) and non-uniform initialization (NI).....	33
5.3	Performance of index-based ISCD with non-uniform initialization (NI).....	34
5.4	Performance of index-based ISCD in probability-domain and log-domain .....	36
5.5	Comparison of bit-based ISCD, index-based ISCD, and concatenated decoding for source correlation factor $\rho = 0.8$ .....	37
5.6	Comparison of bit-based ISCD, index-based ISCD, and concatenated decoding for source correlation factor $\rho = 0.95$ .....	38
5.7	Summary of complexity for channel decoder with $L = 300$ , $\nu = 2$ . The values outside (inside) the parentheses represent computation (memory) .....	40
5.8	Summary of complexity for source decoder with $L = 300$ , $\nu = 2$ . The values outside (inside) the parentheses represent computation (memory) .....	40
B.1	Performances of various decoders.....	48



# List of Figures

2.1	Block diagram of bit-based transmitter.....	14
2.2	Block diagram of bit-based receiver.....	14
2.3	An example of code trellis.....	14
3.1	Block diagram of the index-based transmitter.....	22
3.2	Block diagram of the index-based receiver.....	22
3.3	Trellis diagrams used for (a) the encoder and (b) the decoder.....	23
4.1	Log-domain index-based ISCD.....	29
5.1	Performance of bit-based ISCD.....	32
5.2	Performance of index-based ISCD with non-uniform initialization (NI).....	34
5.3	Performance of index-based ISCD in probability and log-domain.....	35
5.4	Comparison of bit-based and index-based ISCD.....	36
5.5	Comparison of bit-based ISCD, index-based ISCD, and concatenated decoding for source correlation factor $\rho = 0.8$ .....	37
5.6	Comparison of bit-based ISCD, index-based ISCD, and concatenated decoding for source correlation factor $\rho = 0.95$ .....	38
A.1	Plot of $\log(1 + e^{- a-b })$ vs. $ a - b $ .....	44
B.1	Performances of various decoders.....	47

# Chapter 1

## Introduction

In digital multimedia communications, the characteristic parameters extracted from media sources are highly sensitive against transmission errors. Some bit errors in the received parameters can result in extremely annoying artifact. How to design a robust coding/decoding system involves a lot of technical challenges. In a conventional system design problem, the source codec and channel codec are designed separately and independently. This approach is justified by Shannon's separation theorem [1], which ensures the optimal performance of the overall communication system. However, separate design of the source and channel codec is based on the assumed optimality of each other and therefore is not feasible due to impractical constraints. In real-world communication systems, complexity and signal delay are highly limited so that the design of perfect source and channel coding techniques is impossible. Moreover, the separate design prevents one codec from taking advantage of the imperfection and characteristics of the other. It has been shown by many researchers that the system performance can be further improved by a joint design of the source and channel codecs. The name joint source-channel coding/decoding (JSCC/JSCD) has been used to refer to all the techniques that were developed to compensate for the drawbacks of the separation principle.

While joint design of source and channel codecs seems to be a better solution, the way how to realize is still not clear due to the lack of good models and necessary theories in this area. It is expected that a joint source-channel codec design is much more difficult than the separate design. To overcome the difficulties, a number of joint design approaches have been proposed in the past two decades. Depending on whether the implementation is for the transmitter or receiver, joint design methods can

be divided into JSCC and JSCD. The very early work in JSCC involves an integrated design of source and channel codecs. However, due to the difficulties in designing practical encoders and decoders, there are very few papers in this research area. To make JSCC realizable, integrated design of source codec and channel codec is simplified by fixing one codec and designing the other so that the end-to-end distortion is minimized. Related techniques can be roughly divided into three categories [2]: source coding robust to channel errors, channel coding adaptive to source properties, and combined source-channel coding.

In this thesis, we will focus the JSCD design problem at the receiver site. Due to practical constraints, the outputs of a source encoder are not independent identically distributed (i.i.d.) and exhibit some form of non-uniform distribution and correlation, which are referred to as source residual redundancy. A common approach is to take advantage of the source residual redundancy at the source or/and channel decoder to improve the channel robustness as well as signal quality. This is called redundancy-based JSCD and can be further divided into three categories: error concealment, source-controlled channel decoding (SCCD), and iterative source-channel decoding (ISCD).

In error concealment, the residual redundancy is exploited by the source decoder to reduce the subjective effects of the residual errors which are not eliminated by the channel decoder. In this way, the remaining channel errors are “concealed.” The general concept of this JSCD technique is proposed by Fingscheidt and Vary in [3] and is called the softbit source decoding (SBSD). The SBSBD technique can be used after channel decoding with soft-outputs or for the case where channel coding is not used at all. The estimation is carried out for the encoded parameters rather than for individual bits of the parameter indexes, since the dependencies of the indexes are stronger than the correlations of the index bits. Fingscheidt’s approach to SBSBD exploits source residual redundancy in terms of an unequal parameter distribution and inter-frame correlation. By additionally exploiting correlation between source codec parameters, the capabilities of SBSBD have been extended [4].

In contrast to SBSBD which incorporates the residual redundancy into source decoding, the SCCD was proposed by Hagenauer [5] with an attempt to exploit the residual redundancy in a channel decoding process. Unlike conventional channel decoding schemes, the inputs of the SCCD decoder also include source a priori

information. When the channel is less noisy, the channel decoder relies mostly on the received symbol values; otherwise, as during deep fade, decoding relies more on the a priori information of the symbol than the received symbol value. The most commonly used a priori information is the a priori knowledge of the source distribution, which can be obtained in advance using a training sequence. Conventional BCJR algorithm [6] for decoding convolutional codes has been devised based on a bit-level code trellis. In our recent work [7], we developed a modified BCJR algorithm for the SCCD and investigated its application to distributed speech recognition (DSR). The basic strategy of our modified BCJR algorithm is to sectionalize the bit-level trellis in a way that bit-level as well as symbol-level source correlations can be exploited. To proceed with this, we chose to decode the quantizer indexes in a frame as nonbinary symbols according to their index length.

In ISCD scheme, SBS and SCCD are combined together in an iterative process to achieve further improvement compared with using SBS or SCCD alone. The breakthrough of channel coding was made when Berrou *et al.* introduced the turbo codes [8][9]. At the transmitter site the encoding scheme consists of a parallel concatenation of two recursive systematic convolutional codes, wherein the inputs of the constituent encoders are separated by a large interleaver. At the receiver site an iterative decoding strategy is applied to exchange so-called extrinsic information between both constituent decoders in each iteration step. The extrinsic information from one decoder is used as the additional a priori information for another. This strategy resembles the belief propagation decoding algorithm for low density parity check (LDPC) codes originally introduced by Gallager [10]. The novel iterative decoding approach made it possible to reach Shannon's performance bound quite closely by 0.5dB with reasonable computational efforts and delay. It has been named turbo because the iterative exchange of information reminds them of the turbo-charger of combustion engines. The general turbo principle introduced by Hagenauer [11] shows that it can be further applied to most of the concatenation of two systematic soft-in-soft-out (SISO) decoders. The soft-input-soft-output (SISO) property of SBS makes it a well candidate as the constituent code in a turbo-like system. The ISCD derived by Adrat [12] has successfully adapted the turbo principle to the system with concatenation of a channel decoder and a softbit source decoder. The former utilizes the explicitly artificial redundancy and the latter utilizes the implicitly residual

redundancy. The ISCD takes the advantage of turbo principle and raises the estimated quality of source parameters to a level higher than conventional separated decoding.

However, since SBSB is essentially a symbol-based algorithm which estimates the a posteriori probabilities of each source parameter; it must be modified in order to be concatenated with a bit-based channel decoder. Adrat provided a solution for modifying the SBSB, which adds additional calculation for extrinsic information of each bit within an index. This not only makes SBSB more complex, but also splits the turbo decoder into two sub-blocks that operates with different segment-perspective. Since general form of ISCD is a sub-optimum solution derived from the optimum estimation of source parameters [13], there must exist an alternate ISCD system with consistent decoding algorithm. From now on we will refer to Adrat's solution as bit-based ISCD, and we would like to seek a solution starting with a modified channel decoder, namely index-based ISCD.

Symbol-based turbo code has been proposed by Bingeman [14], which parses the parallel data streams of the turbo encoder into  $n$ -bit symbols and maps each symbol to a point in a  $2^n$ -ary signal set. Furthermore, the interleaver is restricted to permute in terms of sub-blocks. With this restriction in place, the effective encoder operates on a symbol-by-symbol basis. Trade-offs between the BER performance, code rate, spectral efficiency, and decoder complexity can be made by the selection of different symbol sizes and modulation techniques. In the case of symbol-based turbo code with BPSK modulation, the BER performance can be improved while at the same time decreasing the decoder complexity as compared with the traditional turbo code.

In this thesis, we will develop the index-based ISCD based on the modification of BCJR decoder. Apparently the term "symbol" here corresponds to the index of the quantized parameters at the output of source encoder. The bit-based interleaver between the source and recursive systematic convolutional (RSC) encoders is replaced by an index-based interleaver to ensure the proper order of parameters at the input of the RSC encoder. The contents of the quantizer, the bit-mapper, and the RSC encoder remain the same as in bit-based ISCD. At the receiver site, the BCJR algorithm is modified to operate in a symbol-by-symbol manner similar as in [7]. In order to simplify the SBSB, it also receives and delivers the symbol-based extrinsic information similar as in [4]. The derivation of extrinsic information from the SBSB in [3] in the index-based perspective is given and therefore the turbo decoder only

exchanges index-based terms without the conversion between bit-level and index-level. Experimental results show that the parameter-estimate quality of the index-based ISCD outperforms the bit-based ISCD, especially for heavily noisy channels.

The bit-based ISCD will be reviewed in Chapter 2. It includes the detail of the conventional BCJR algorithm with adaptation to turbo decoding and the modified SBSB with the extrinsic information. In Chapter 3 we will derive the index-based ISCD by further modifying the modified BCJR algorithm and applying an index-based interleaver, and give the interpretation of index-based extrinsic information. Chapter 4 transforms the algorithms in Chapter 3 from probability-domain to log-domain in order to meet the requirements of real-time implementation. The simulation results of both systems will be shown and compared in Chapter 5. Finally, we will conclude the thesis and list some future research directions in Chapter 6.



## Chapter 2

# Bit-Based Iterative Source-Channel Decoding

Iterative source channel decoding (ISCD) [4] is a turbo code-like system for multimedia communication. Unlike conventional system, an interleaver is added between source (including quantizer and bit mapper) and channel encoder to generate uncorrelated bit sequences at their input. Similarly, a de-interleaver is also added between channel and source decoder to recover the order of original data. The reason for doing this is to make source and channel decoders to deal with mutually independent signals. Additionally, the decoder is modified with an extra feedback process that takes advantage of iterative information exchanging. This is so called “turbo principle” and is widely applied in systems with two channel encoders (decoders), known as turbo code. Here we have BCJR channel decoder and a softbit source decoder with distinct decoding algorithms. The crucial part is to determine extrinsic information exchanged from one decoder to another.

In this chapter, we will briefly introduce the detail of an ISCD system. Starting from the transmitter, all constituent blocks will be clearly defined. At the receiver side, we will discuss two decoders separately.

### §2.1 System Implementation

Consider the block diagram of the transmitter shown in Figure 2.1. At time instant  $t$ , we have a real valued source  $v_t$ , which is quantized by a scalar Max quantizer and then mapped to an M-bit sequence  $\mathbf{u}_t$  as follows:

$$\mathbf{u}_t = \{u_t(1), u_t(2), \dots, u_t(m), \dots, u_t(M)\}.$$

For consistency of encoding and decoding, we assume all bits are pre-modulated using Binary Phase Shift Keying (BPSK), e.g.  $u_t(m) \in \{+1, -1\}$  for all  $t$  and  $m$ . Define the set:

$$\mathbf{U}_1^T = \{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_t, \dots, \mathbf{u}_T\}$$

which contains a large amount of  $M$ -bit sequences, are passed through a bit interleaver  $\Phi$  to generate the independent bit pattern  $\mathbf{x}_1^L$ :

$$\mathbf{x}_1^L = \{x_1, x_2, \dots, x_l, \dots, x_L\},$$

where  $L = MT$  is the size of the interleaver. Each bit  $x_l$  corresponds to a specific re-ordered bit  $u_t(m)$ :

$$x_l = u_t(m), \quad l = 1, 2, \dots, L \quad (2.1)$$

The interleaved bit pattern  $\mathbf{x}_1^L$  is considered as the information bits for the channel encoder. We will focus on the  $(n, 1)$ –Recursive Systematic Convolutional (RSC) code with constraint length  $\nu$ . Thus, each input bit  $x_l$  will generate one systematic output bit  $y_l^s = x_l$  and  $n - 1$  parity check bits  $\mathbf{y}_l^p$ :

$$\begin{aligned} \mathbf{Y}_1^{\hat{L}} &= \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_l, \dots, \mathbf{y}_{\hat{L}}\}, \\ \mathbf{y}_l &= \{y_l^s, \mathbf{y}_l^p\} = \{x_l, \mathbf{y}_l^p\}, \quad l = 1, 2, \dots, \hat{L} \end{aligned} \quad (2.2)$$

where the length  $\hat{L}$  includes  $L$  information bits and the termination bits for the RSC encoder. The block output  $\mathbf{Y}_1^{\hat{L}}$  is then transmitted through an AWGN channel.

Let the noise-corrupted channel output be  $\tilde{\mathbf{Y}}_1^{\hat{L}} \in \mathbb{R}^{\hat{L}}$ , which provides channel-related information for the decoding algorithm. Since the channel is assumed memoryless, we can calculate the received block information  $P(\tilde{\mathbf{Y}}_1^{\hat{L}} | \mathbf{Y}_1^{\hat{L}})$  as the product of bit-based probabilities of systematic and parity outputs  $p(\tilde{y}_l^s | x_l)$  and  $p(\tilde{\mathbf{y}}_l^p | \mathbf{y}_l^p)$ . With proper channel estimation (channel signal-to-noise ratio  $E_s/N_0$  is known), we have the explicit terms of channel-related information in terms of channel transition probabilities:

$$p(\tilde{y}_l^s | x_l) = \frac{1}{\sqrt{2\pi}\sigma_{noise}} \cdot \exp\left[-\frac{E_s}{N_0}(\tilde{y}_l^s - x_l)^2\right] \quad (2.3)$$



$$p(\tilde{\mathbf{y}}_l^p | \mathbf{y}_l^p) = \left( \frac{1}{\sqrt{2\pi}\sigma_{noise}} \right)^{n-1} \cdot \exp \left[ -\frac{E_s}{N_0} \|\tilde{\mathbf{y}}_l^p - \mathbf{y}_l^p\|^2 \right] \quad (2.4)$$

where  $\sigma_{noise}^2 = N_0/2$  is the variance of noise and  $\|\cdot\|$  is the norm function.

The receiver will exploit both channel-related and source *a priori* information in the decoding algorithm to better estimate the output  $\tilde{v}_t$ . Its structure is illustrated in Figure 2.2.

Iterative source-channel decoder is in turbo-like structure. In order to achieve good performance, the constituent components must satisfy the turbo principle:

1. Concatenation of component blocks
2. Interleavers
3. Soft-In-Soft-Out (SISO) process
4. Exchange of extrinsic information

Since the channel information is passed to both decoders, the system is called a parallel concatenated scheme. During the iterative process, each decoder calculates the extrinsic information of every possibly transmitted bit; which is then combined with the source *a priori* information to generate the more reliable *a priori* information used for next decoding iteration. The bit-level *a posteriori* probabilities (APP) can be used to compute the index-level APP based on the independence assumption in each index bit pattern. In summary, the estimation of the parameters  $\tilde{v}_t$  consists of the following steps:

1. Set  $k = 0$ ,  $P_{SBS D}^{[ext]}(x_l) = 1$  for all  $l$ .
2. Pass the channel-related information  $\{p(\tilde{y}_l^s | x_l), p(\tilde{\mathbf{y}}_l^p | \mathbf{y}_l^p)\}$  and bit-level *a priori* information  $P(x_l) \cdot P_{SBS D}^{[ext]}(x_l)$  to channel decoder's input. The channel decoder uses BCJR algorithm to calculate bit-level extrinsic information  $P_{CD}^{[ext]}(x_l)$  and bit-level APP  $P(x_l | \tilde{\mathbf{Y}}_1^L)$ . De-interleave  $P_{CD}^{[ext]}(x_l)$  to obtain  $P_{CD}^{[ext]}(u_t(\lambda))$  for use in softbit source decoding.
3. De-interleave  $P(x_l | \tilde{\mathbf{Y}}_1^L)$  and use the MMSE algorithm to compute the conditioned mean for the parameter  $\tilde{v}_t$ :

$$\tilde{v}_t \text{ (after channel decoding)} = \sum_{i=0}^{2^M-1} v_t(i) \cdot \prod_{l \in (\text{interleaved positions of } u_t(\lambda))} P(x_l | \tilde{\mathbf{Y}}_1^L, \mathbf{u}_t = i) \quad (2.5)$$

where  $v_t(i)$  is the codebook value of index  $i$ . The resulting parameter-to-noise ratio is defined as performance at the  $(k^+)$ -th iteration. If the improvement is less than a preset threshold, stop process; otherwise, go to step 4.

4. Pass de-interleaved channel-related information  $p(\tilde{u}_t(\lambda)|u_t(\lambda))$  and index-level a priori information  $\left[ P(\mathbf{u}_t|\mathbf{u}_{t-1}) \cdot \prod_{m=1}^M P_{CD}^{[ext]}(u_t(m)) \right]$  to softbit source decoder's input. Perform the softbit source decoding algorithm to calculate bit-level extrinsic information  $P_{SBSD}^{[ext]}(u_t(\lambda))$  and index-level APP  $P(\mathbf{u}_t|\tilde{\mathbf{U}}_1^T)$ . Interleave  $P_{SBSD}^{[ext]}(u_t(\lambda))$  to obtain  $P_{SBSD}^{[ext]}(x_l)$  for use in channel decoding.
5. Use  $P(\mathbf{u}_t|\tilde{\mathbf{U}}_1^T)$  to compute the conditional mean for the parameter  $\tilde{v}_t$ :

$$\tilde{v}_t \text{ (after source decoding)} = \sum_{i=0}^{2^M-1} P(\mathbf{u}_t = i|\tilde{\mathbf{U}}_1^T) \cdot v_t(i) \quad (2.6)$$

The resulting parameter-to-noise ratio is defined as performance at the  $(k + 1)$ -th iteration. If the improvement is less than a preset threshold, stop process; otherwise,  $k \leftarrow k + 1$  and return to step 2.

## §2.2 BCJR Algorithm for Channel Decoding

BCJR algorithm [6][15] is widely used for soft-input soft-output channel decoding of convolutional codes. In order to increase the reliability of current information bit, it observes the entire received block  $\tilde{\mathbf{Y}}_1^L$  and utilizes the trellis structure of convolutional encoder to derive a recursive formula that drastically lower the computational complexity. Furthermore, the recursive reliability passing structure makes BCJR an adequate constituent decoder in iterative decoding. In this section we will briefly review the BCJR algorithm. Specifically, we show how to derive APP from the received signal and define extrinsic output for each information bit. For convenience, we only consider the case for  $x_l = +1$  among all formula in this

section; however, similar calculations are needed for  $x_l = -1$  to ensure the sum of  $P(x_l = +1|\tilde{\mathbf{Y}}_1^L)$  and  $P(x_l = -1|\tilde{\mathbf{Y}}_1^L)$  equals to 1.

The trellis diagram determines the current output bits by observing the current input bit and the latest state content updated by the previous input bit. In other words, the current input bit and the latest state content determine a unique state transition (or branch). We can also use this transition to update the state content before dealing with the next input bit. Therefore, we include the state information to a posteriori probability of  $x_l$ :

$$\begin{aligned} P(x_l = +1|\tilde{\mathbf{Y}}_1^L) &= \frac{P(x_l = +1, \tilde{\mathbf{Y}}_1^L)}{P(\tilde{\mathbf{Y}}_1^L)} = \frac{\sum_{(s_{l-1}, s_l) \in \Sigma_l^+} p(s_{l-1}, s_l, \tilde{\mathbf{Y}}_1^L)}{P(\tilde{\mathbf{Y}}_1^L)} \\ &= C \cdot \sum_{(s_{l-1}, s_l) \in \Sigma_l^+} p(s_{l-1}, s_l, \tilde{\mathbf{Y}}_1^L) \end{aligned} \quad (2.7)$$

where  $C$  is a constant independent of  $x_l$ ,  $s_{l-1}$  and  $s_l$  are the state before and after the current state transition, and  $\Sigma_l^+$  is the set of all state transitions  $s_{l-1} \rightarrow s_l$  resulting from the input bit  $x_l = +1$ . Figure 2.3 shows an example.

By assuming a memoryless channel and the independence of input bits  $x_l$ , the joint probability  $p(s_{l-1}, s_l, \tilde{\mathbf{Y}}_1^L)$  can be split into three terms:

$$\begin{aligned} p(s_{l-1}, s_l, \tilde{\mathbf{Y}}_1^L) &= p(s_{l-1}, \tilde{\mathbf{Y}}_1^{l-1}) \cdot p(s_l, \tilde{\mathbf{y}}_l | s_{l-1}) \cdot p(\tilde{\mathbf{Y}}_{l+1}^L | s_l) \\ &\triangleq \alpha_{l-1}(s_{l-1}) \cdot \gamma_l(s_{l-1}, s_l) \cdot \beta_l(s_l) \end{aligned} \quad (2.8)$$

The forward recursion for *forward metric*  $\alpha_l(s_l)$ :

$$\alpha_l(s_l) = \sum_{s_{l-1}} \gamma_l(s_{l-1}, s_l) \cdot \alpha_{l-1}(s_{l-1}) \quad (2.9)$$

The backward recursion for *backward metric*  $\beta_{l-1}(s_{l-1})$ :

$$\beta_{l-1}(s_{l-1}) = \sum_{s_l} \gamma_l(s_{l-1}, s_l) \cdot \beta_l(s_l) \quad (2.10)$$

The *branch metric*  $\gamma_l(s_{l-1}, s_l)$ :

$$\begin{aligned} \gamma_l(s_{l-1}, s_l) &= P(s_l | s_{l-1}) \cdot p(\tilde{\mathbf{y}}_l | s_{l-1}, s_l) \\ &= P(x_l = +1) \cdot p(\tilde{\mathbf{y}}_l | \mathbf{y}_l) \\ &= P(x_l = +1) \cdot p(\tilde{y}_l^s | x_l = +1) \cdot p(\tilde{\mathbf{y}}_l^p | \mathbf{y}_l^p) \end{aligned} \quad (2.11)$$

The 2<sup>nd</sup> equality comes from  $(s_{l-1}, s_l) \in \Sigma_l^+$  in (2.7), and the 3<sup>rd</sup> equality holds for a memoryless channel. It is common to start a convolutional code with an all-zero state, and force it back to all-zero state after encoding all information bits. Therefore, the

forward and the backward metrics can be initialized as follows:

$$\alpha_0(s_0) = \begin{cases} 1, & s_0 = \mathbf{0} \\ 0, & s_0 \neq \mathbf{0} \end{cases}$$

$$\beta_{\hat{L}}(s_{\hat{L}}) = \begin{cases} 1, & s_{\hat{L}} = \mathbf{0} \\ 0, & s_{\hat{L}} \neq \mathbf{0} \end{cases}$$

The next task is to define extrinsic information of the path ‘‘CD to SBSD’’, specifically, the term  $P_{CD}^{[ext]}(x_l = +1)$ . Note that extrinsic information may not be a legible probability (they are not summed to 1), we must calculate  $P_{CD}^{[ext]}(x_l = \pm 1)$  separately.

By observing the inputs of SBSD, we find that there are three distinct groups of information:

- Channel-related information:  $p(\tilde{u}_t(\lambda)|u_t(\lambda)) = p(\tilde{y}_l^s|x_l)$
- Index-level *a priori* information:  $P(\mathbf{u}_t|\mathbf{u}_{t-1})$
- Extrinsic information from channel decoder:  $P_{CD}^{[ext]}(u_t(\lambda))$

Since SBSD and CD use the same channel-related information and the bit-level *a priori* information can be derived from marginal probability of index-level *a priori* information, both terms must be separated from the extrinsic information  $P_{CD}^{[ext]}(x_l = +1)$ . We first consider the extrinsic part in the branch metric:

$$\begin{aligned} \gamma_l(s_{l-1}, s_l) &= P(x_l = +1) \cdot p(\tilde{y}_l^s|x_l = +1) \cdot p(\tilde{\mathbf{y}}_l^p|\mathbf{y}_l^p) \\ &\triangleq P(x_l = +1) \cdot p(\tilde{y}_l^s|x_l = +1) \cdot \gamma_l^{[ext]}(s_{l-1}, s_l) \end{aligned} \quad (2.12)$$

where  $\gamma_l^{[ext]}(s_{l-1}, s_l) = p(\tilde{\mathbf{y}}_l^p|\mathbf{y}_l^p)$ . With (2.12), (2.7) becomes:

$$\begin{aligned} P(x_l = +1|\tilde{\mathbf{Y}}_l^{\hat{L}}) &= C \cdot \sum_{(s_{l-1}, s_l) \in \Sigma_l^+} \alpha_{l-1}(s_{l-1}) \cdot \gamma_l(s_{l-1}, s_l) \cdot \beta_l(s_l) \\ &= C \cdot P(x_l = +1) \cdot p(\tilde{y}_l^s|x_l = +1) \cdot \sum_{(s_{l-1}, s_l) \in \Sigma_l^+} \alpha_{l-1}(s_{l-1}) \cdot \gamma_l^{[ext]}(s_{l-1}, s_l) \cdot \beta_l(s_l) \\ &\triangleq C \cdot P(x_l = +1) \cdot p(\tilde{y}_l^s|x_l = +1) \cdot P_{CD}^{[ext]}(x_l = +1) \end{aligned} \quad (2.13)$$

Where the extrinsic information derived from the channel decoding is

$$P_{CD}^{[ext]}(x_l = +1) = \sum_{(s_{l-1}, s_l) \in \Sigma_l^+} \alpha_{l-1}(s_{l-1}) \cdot \gamma_l^{[ext]}(s_{l-1}, s_l) \cdot \beta_l(s_l) \quad (2.14)$$

After the 1<sup>st</sup> decoding iteration, the *a priori* information is updated by the feedback of SBSD:

$$P(x_l = +1) \leftarrow \left[ P_{SBSD}^{[ext]}(x_l = +1) \cdot P(x_l = +1) \right] \quad (2.15)$$

which also updates the branch metric:

$$\gamma_l(s_{l-1}, s_l) = P(x_l = +1) \cdot P_{SBSD}^{[extr]}(x_l = +1) \cdot p(\tilde{y}_l^s | x_l = +1) \cdot \gamma_l^{[extr]}(s_{l-1}, s_l) \quad (2.16)$$

Finally, the APP in (2.13) becomes:

$$P(x_l = +1 | \tilde{\mathbf{Y}}_l^L) = C \cdot P(x_l = +1) \cdot P_{SBSD}^{[extr]}(x_l = +1) \cdot p(\tilde{y}_l^s | x_l = +1) \cdot P_{CD}^{[extr]}(x_l = +1) \quad (2.17)$$

## §2.3 Softbit Source Decoding

In this section, the softbit source decoding (SBSD) algorithm [3] is developed to exploit the 1<sup>st</sup>-order a priori knowledge (AK1) source information. To proceed with this, the index transition probabilities  $P(\mathbf{u}_t | \mathbf{u}_{t-1})$  are calculated in advance by a large amount of training data. The index-level APP is derived as follows:

$$\begin{aligned} P(\mathbf{u}_t = i | \tilde{\mathbf{U}}_1^T) &= P(\mathbf{u}_t = i | \tilde{\mathbf{U}}_1^t) \\ &= P(\mathbf{u}_t = i | \tilde{\mathbf{U}}_1^{t-1}, \tilde{\mathbf{u}}_t) \\ &= C \cdot p(\tilde{\mathbf{u}}_t | \mathbf{u}_t = i) \cdot \sum_{j=0}^{2^M-1} P(\mathbf{u}_t = i | \mathbf{u}_{t-1} = j) \cdot P(\mathbf{u}_{t-1} = j | \tilde{\mathbf{U}}_1^{t-1}) \end{aligned} \quad (2.18)$$

where

$$p(\tilde{\mathbf{u}}_t | \mathbf{u}_t = i) = \prod_{\substack{m=1 \\ \{\mathbf{u}_t=i\}}}^M p(\tilde{u}_t(m) | u_t(m)) \quad (2.19)$$

is derived from the product of bit-level channel-related information under a memoryless channel assumption. Since channel decoding is performed before SBSBD, we can combine its extrinsic information with *a priori* information as follows:

$$P(\mathbf{u}_t = i | \mathbf{u}_{t-1} = j) \leftarrow \left[ P(\mathbf{u}_t = i | \mathbf{u}_{t-1} = j) \cdot \prod_{\substack{m=1 \\ \{\mathbf{u}_t=i\}}}^M P_{CD}^{[extr]}(u_t(m)) \right] \quad (2.20)$$

Then (2.18) becomes:

$$P(\mathbf{u}_t = i | \tilde{\mathbf{U}}_1^t) = C \cdot \prod_{\substack{m=1 \\ \{\mathbf{u}_t=i\}}}^M p(\tilde{u}_t(m) | u_t(m)) \cdot P_{CD}^{[extr]}(u_t(m)) \cdot \sum_{j=0}^{2^M-1} P(\mathbf{u}_t = i | \mathbf{u}_{t-1} = j) \cdot P(\mathbf{u}_{t-1} = j | \tilde{\mathbf{U}}_1^{t-1}) \quad (2.21)$$

Similar as the case of channel decoding, our next task is to find the SBSBD's extrinsic information that can be fed back to channel decoder. At this point we notice that the SBSBD algorithm is index based, which differs from the bit-level channel

decoder. Therefore, we must compute the marginal APP of a specific bit  $u_t(\lambda)$  using the index-level APP in (2.21). Additionally, we classify the input of channel decoder into bit-level channel-related information  $p(\tilde{y}_l^s|x_l = +1)$ , which corresponds to de-interleaved  $p(\tilde{u}_t(\lambda)|u_t(\lambda) = +1)$ , and bit-level a priori information  $P(x_l = +1)$ , which corresponds to de-interleaved  $P(u_t(\lambda) = +1)$ . The next step is to express the bit-level APP in terms of bit-level extrinsic information of SBSBD and CD:

$$\begin{aligned}
P(u_t(\lambda) = +1|\tilde{\mathbf{U}}_1^t) &= \sum_{(\mathbf{u}, u_t(\lambda) = +1)} P(\mathbf{u}, \tilde{\mathbf{U}}_1^t) \\
&= C \cdot P(u_t(\lambda) = +1) \cdot p(\tilde{u}_t(\lambda)|u_t(\lambda) = +1) \cdot P_{CD}^{[ext]}(u_t(\lambda) = +1) \cdot \\
&\quad \cdot \left\{ \sum_{(\mathbf{u}, u_t(\lambda) = +1)} \prod_{\substack{m=1 \\ m \neq \lambda}}^M p(\tilde{u}_t(m)|u_t(m)) \cdot P_{CD}^{[ext]}(u_t(m)) \cdot \sum_{j=0}^{2^M-1} P(\mathbf{u}_t^{[ext]}(\lambda)|\mathbf{u}_{t-1} = j) \cdot P(\mathbf{u}_{t-1} = j|\tilde{\mathbf{U}}_1^{t-1}) \right\} \\
&\triangleq C \cdot P(u_t(\lambda) = +1) \cdot p(\tilde{u}_t(\lambda)|u_t(\lambda) = +1) \cdot P_{CD}^{[ext]}(u_t(\lambda) = +1) \cdot P_{SBSBD}^{[ext]}(u_t(\lambda) = +1)
\end{aligned} \tag{2.22}$$

In the above equation, we assume independency of bit  $u_t(\lambda)$  and other bits  $\mathbf{u}_t^{[ext]} = \{u_t(1), \dots, u_t(\lambda-1), u_t(\lambda+1), \dots, u_t(M)\}$ :

$$P(\mathbf{u}_t^{[ext]}(\lambda)|\mathbf{u}_{t-1} = j) \triangleq \frac{P(\mathbf{u}_t|\mathbf{u}_{t-1} = j)}{P(u_t(\lambda) = +1)} \tag{2.23}$$

where  $\mathbf{u}_t = \{u_t(\lambda), \mathbf{u}_t^{[ext]}(\lambda)\}$ . We also define the extrinsic information of SBSBD:

$$P_{SBSBD}^{[ext]}(u_t(\lambda) = +1) \triangleq \sum_{(\mathbf{u}, u_t(\lambda) = +1)} \prod_{\substack{m=1 \\ m \neq \lambda}}^M p(\tilde{u}_t(m)|u_t(m)) \cdot P_{CD}^{[ext]}(u_t(m)) \cdot \sum_{j=0}^{2^M-1} P(\mathbf{u}_t^{[ext]}(\lambda)|\mathbf{u}_{t-1} = j) \cdot P(\mathbf{u}_{t-1} = j|\tilde{\mathbf{U}}_1^{t-1}) \tag{2.24}$$

Then (2.24) is fed back to channel decoder input as the term  $P_{SBSBD}^{[ext]}(x_l = +1)$  in (2.15).

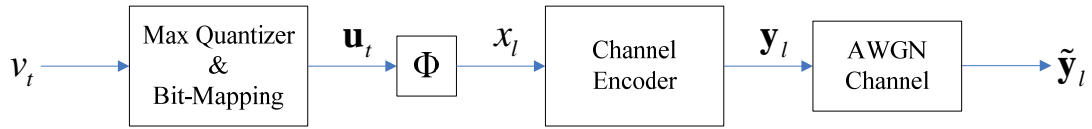


Figure 2.1: Block diagram of bit-based transmitter

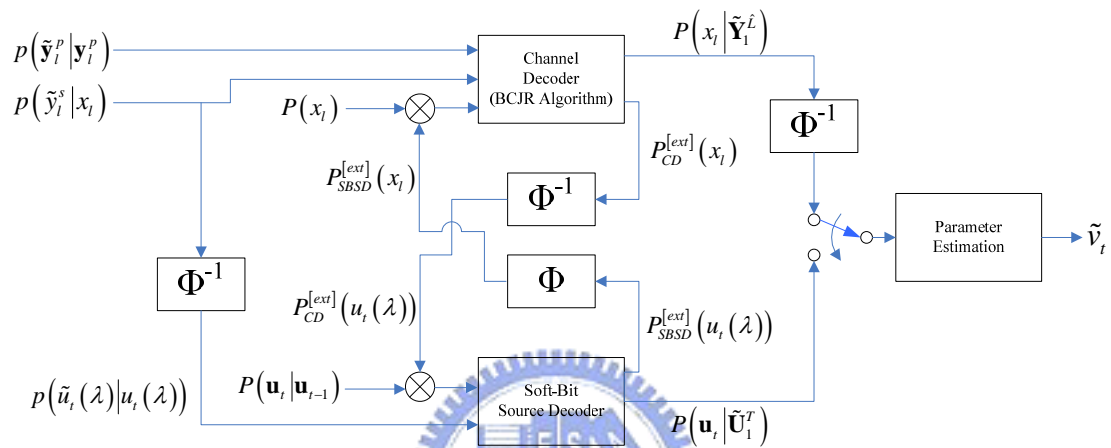


Figure 2.2: Block diagram of bit-based receiver

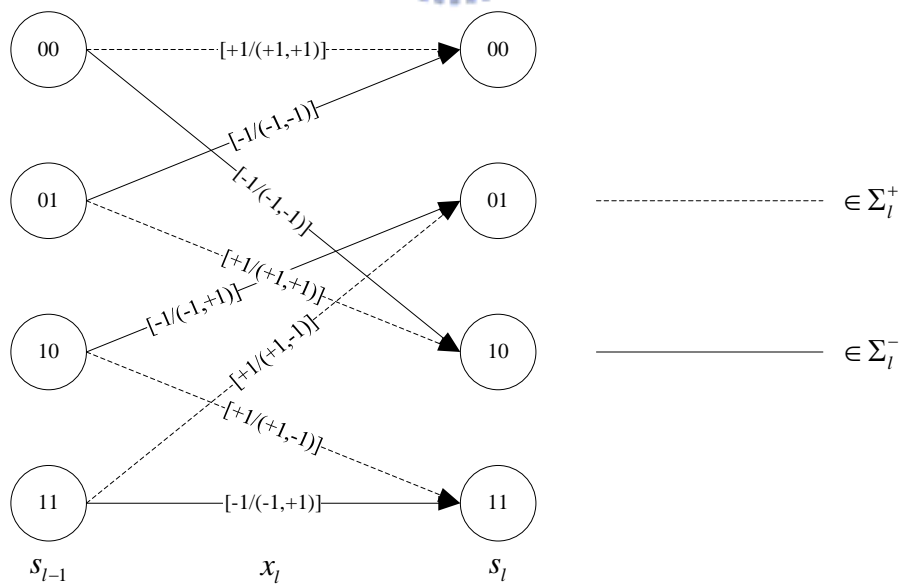


Figure 2.3: An example of code trellis

## Chapter 3

# Index-Based Iterative Source-Channel Decoding

When using the bit-level ISCD, there are two shortcomings in compatible conversion between bit-level and index-level information during decoding process. One is the multiplication of bit-level APP after channel decoding algorithm. Another is the independence assumption between a particular bit and other bits within the same index when deriving extrinsic information for SBSD. A better solution is to develop a consistent decoding process operates at the index level.

To accomplish this goal, we need to modify the BCJR algorithm in a way that only index-level information is involved. In our previous work [7], a modified BCJR algorithm has been proposed to estimate index APP for source-controlled channel decoding. The basic strategy is to sectionalize the channel code trellis and utilize source residual redundancy. We now wish to develop a new ISCD algorithm that exchanges extrinsic information at the index level.

Due to the existence of an interleaver, the bit sequences entering the channel decoder will no longer have correlation, and therefore we simplify the modified BCJR algorithm without the use of residual redundancy. Besides, we will use the index-based interleavers which do not scramble the bit order within an index.

### §3.1 System Implementation

Consider the index-based transmitter shown in Figure 3.1, where only the index-based interleaver is different from the bit-based interleaver in Figure 2.1. An index-based only permutes the order of different indexes, but never changes the order of bits within each index. Therefore, after interleaving, a sequence of indexes will



appear in a memoryless form. We denote such an interleaved and memoryless index sequence as:

$$\begin{aligned}\mathbf{X}_1^T &= (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_\tau, \dots, \mathbf{x}_T), \\ \mathbf{x}_\tau &= \mathbf{u}_\tau, \quad \tau = 1, 2, \dots, T\end{aligned}\quad (3.1)$$

The basic structure of the channel encoder remains unchanged. We only need to change the viewpoint of how to encode the input signals. Here we focus our discussion on the  $(n, 1)$  – RSC code. Let  $(n, 1) \rightarrow (Mn, M)$ , the code rate remains the same and therefore we define the index-level channel encoder output:

$$\begin{aligned}\mathbf{Y}_1^{\hat{T}} &= \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_\tau, \dots, \mathbf{y}_{\hat{T}}\}, \\ \mathbf{y}_\tau &= \{\mathbf{y}_\tau^s, \mathbf{y}_\tau^p\} = \{\mathbf{x}_\tau, \mathbf{y}_\tau^p\}, \quad \tau = 1, 2, \dots, \hat{T}\end{aligned}\quad (3.2)$$

where  $\mathbf{y}_\tau^s$  and  $\mathbf{y}_\tau^p$  correspond to systematic and parity output, and the length  $\hat{T}$  includes  $T$  indexes and termination redundancy. Specifically,  $\mathbf{y}_\tau^s$  contains  $M$  bits and  $\mathbf{y}_\tau^p$  contains  $(n - 1) \cdot M$  bits. Transmission over an AWGN channel is still at bit-level and the bits within each output  $\mathbf{y}_\tau$  are still in bipolar form ( $\in \{+1, -1\}$ ).

At the receiver side, we must first transform the bit-level channel information into its index-level version. Since an AWGN channel is memoryless, the channel-related information (2.3) and (2.4) of different bits are independent. The index-level channel-related information of systematic and parity parts of encoder output are expressed in terms of index-level channel transition probabilities:

$$p(\tilde{\mathbf{y}}_\tau^s | \mathbf{x}_\tau) = \left( \frac{1}{\sqrt{2\pi}\sigma_{noise}} \right)^M \cdot \exp \left[ -\frac{E_s}{N_0} \|\tilde{\mathbf{y}}_\tau^s - \mathbf{x}_\tau\|^2 \right] \quad (3.3)$$

$$p(\tilde{\mathbf{y}}_\tau^p | \mathbf{y}_\tau^p) = \left( \frac{1}{\sqrt{2\pi}\sigma_{noise}} \right)^{(n-1)M} \cdot \exp \left[ -\frac{E_s}{N_0} \|\tilde{\mathbf{y}}_\tau^p - \mathbf{y}_\tau^p\|^2 \right] \quad (3.4)$$

where  $\sigma_{noise}^2 = N_0/2$  is the variance of noise and  $\|\cdot\|$  is the norm function.

Figure 3.2 shows the block diagram of the receiver. While the basic structure of receiver is similar to the bit-based version, the information exchanges between constituent decoders are performed at the index-level, resulting in a more consistent iterative decoding process. Decoding process is modified as follows:

1. Set  $k = 0$ ,  $P_{SBS D}^{[ext]}(\mathbf{x}_\tau) = P(\mathbf{x}_\tau)$

2. Pass channel-related information  $\{p(\tilde{\mathbf{y}}_\tau^s | \mathbf{x}_\tau), p(\tilde{\mathbf{y}}_\tau^p | \mathbf{y}_\tau^p)\}$  and index-level *a priori* information  $P_{SBSD}^{[ext]}(\mathbf{x}_\tau)$  to channel decoder's input. Perform the modified BCJR algorithm based on a sectionalized trellis and calculate the index-level extrinsic information  $P_{CD}^{[ext]}(\mathbf{x}_\tau)$  and index-level APP  $P(\mathbf{x}_\tau | \tilde{\mathbf{Y}}_1^T)$ . De-interleave  $P_{CD}^{[ext]}(\mathbf{x}_\tau)$  to obtain  $P_{CD}^{[ext]}(\mathbf{u}_t)$  for use in softbit source decoding.
3. De-interleave  $P(\mathbf{x}_\tau | \tilde{\mathbf{Y}}_1^T)$  and use the MMSE algorithm to compute the conditional mean for the parameter  $\tilde{v}_t$ :

$$\tilde{v}_t \text{ (after channel decoding)} = \sum_{i=0}^{2^M-1} P(\mathbf{x}_\tau = i | \tilde{\mathbf{Y}}_1^T) \cdot v_t(i) \quad (3.5)$$

( $\tau$ =interleaved position of  $t$ )

where  $v_t(i)$  is the codebook value of index  $i$ . We then calculate the parameter-to-noise ratio, which is defined as the performance at the  $(k^+)$ -th iteration. If the improvement is less than a preset threshold, stop process; otherwise, go to step 4.

4. Pass de-interleaved channel-related information  $p(\tilde{\mathbf{u}}_t | \mathbf{u}_t)$  and index-level *a priori* information  $\left[ P(\mathbf{u}_t | \mathbf{u}_{t-1}) \cdot P_{CD}^{[ext]}(\mathbf{u}_t) \right]$  to softbit source decoder's input. Perform the softbit source decoding algorithm and calculate the index-level extrinsic information  $P_{SBSD}^{[ext]}(\mathbf{u}_t)$  and index-level APP  $P(\mathbf{u}_t | \tilde{\mathbf{U}}_1^T)$ . Interleave  $P_{SBSD}^{[ext]}(\mathbf{u}_t)$  to obtain  $P_{SBSD}^{[ext]}(\mathbf{x}_\tau)$  for use in channel decoding.
5. Use  $P(\mathbf{u}_t | \tilde{\mathbf{U}}_1^T)$  to compute the conditional mean for the parameter  $\tilde{v}_t$ :

$$\tilde{v}_t \text{ (after source decoding)} = \sum_{i=0}^{2^M-1} P(\mathbf{u}_t = i | \tilde{\mathbf{U}}_1^T) \cdot v_t(i) \quad (3.6)$$

The resulting parameter-to-noise ratio is defined as the performance at the  $(k + 1)$ -th iteration. If the improvement is less than a preset threshold, stop process; otherwise,  $k \leftarrow k + 1$  and return to step 2.

## §3.2 Modified BCJR Algorithm

It is necessary to modify channel decoder to make the decoding to perform on an index at each time. We proposed to merge the bit-level channel output by regarding  $M$  consecutive bits as an unit. Since a state transition occurs in accordance with each bit entering the channel encoder, we must build an  $M$ -stage merged trellis diagram of RSC code, which is called “*sectionalized trellis diagram.*” Figure 3.3 shows an example, where two bits of the trellis code diagram is merged. Note that all states had become four-in-four-out. If we merge more bits within an index, parallel transitions may occur. Specifically, the constraint length of RSC code is  $\nu$  and the goal is to merge  $M$  bits into an index, parallel transitions will occur while  $M > \nu$ .

The conventional BCJR algorithm must be modified to deal with the increased branch number and parallel transitions. From another perspective of viewpoint, we identify a transition branch according to the input index value. Although many parallel transition branches exist between fixed states, each branch still corresponds to an unique input index value  $\mathbf{x}_\tau = i$  and an unique output  $\mathbf{y}_\tau$ . Our new algorithm splits the forward, backward, and branch metrics by adding specific index value notation for different input  $\mathbf{x}_\tau$ . This makes decoder not to suffer from parallel transitions; however, it also needs to multiply metrics and increases the storage memory. Starting with the *a posteriori* probability (APP):

$$P(\mathbf{x}_\tau = i | \tilde{\mathbf{Y}}_1^{\hat{\tau}}) = \sum_{s_\tau} \frac{P(\mathbf{x}_\tau = i, s_\tau, \tilde{\mathbf{Y}}_1^{\hat{\tau}})}{P(\tilde{\mathbf{Y}}_1^{\hat{\tau}})} = C \cdot \sum_{s_\tau} P(\mathbf{x}_\tau = i, s_\tau, \tilde{\mathbf{Y}}_1^{\hat{\tau}}) \quad (3.7)$$

the value  $i$  takes from  $2^M$  different possibilities. Here the summation is performed through all states rather than the transition set  $(s_{\tau-1}, s_\tau) \in \Sigma_\tau^i$ , since the latter may contain repetitive elements. The joint probability term can be split into:

$$\begin{aligned} P(\mathbf{x}_\tau = i, s_\tau, \tilde{\mathbf{Y}}_1^{\hat{\tau}}) &= P(\mathbf{x}_\tau = i, s_\tau, \tilde{\mathbf{Y}}_1^\tau) \cdot P(\tilde{\mathbf{Y}}_{\tau+1}^{\hat{\tau}} | \mathbf{x}_\tau = i, s_\tau, \tilde{\mathbf{Y}}_1^\tau) \\ &\triangleq \alpha_\tau^i(s_\tau) \cdot \beta_\tau^i(s_\tau) \end{aligned} \quad (3.8)$$

where  $\alpha_\tau^i(s_\tau)$  and  $\beta_\tau^i(s_\tau)$  represent *forward* and *backward* metric, respectively. We can further derive the recursion formula:

$$\begin{aligned}
\alpha_\tau^i(s_\tau) &= P(\mathbf{x}_\tau = i, s_\tau, \tilde{\mathbf{Y}}_1^\tau) \\
&= \sum_{s_{\tau-1}} \sum_j P(\mathbf{x}_\tau = i, s_\tau, \mathbf{x}_{\tau-1} = j, s_{\tau-1}, \tilde{\mathbf{Y}}_1^{\tau-1}, \tilde{\mathbf{y}}_\tau) \\
&= \sum_{s_{\tau-1}} \sum_j P(\mathbf{x}_{\tau-1} = j, s_{\tau-1}, \tilde{\mathbf{Y}}_1^{\tau-1}) P(\mathbf{x}_\tau = i, s_\tau, \tilde{\mathbf{y}}_\tau | \mathbf{x}_{\tau-1} = j, s_{\tau-1}, \tilde{\mathbf{Y}}_1^{\tau-1}) \quad (3.9) \\
&\triangleq \sum_{s_{\tau-1}} \sum_j \alpha_{\tau-1}^j(s_{\tau-1}) \gamma_{i,j}(\tilde{\mathbf{y}}_\tau, s_\tau, s_{\tau-1})
\end{aligned}$$

and

$$\beta_{\tau-1}^i(s_{\tau-1}) = \sum_{s_\tau} \sum_j \beta_\tau^j(s_\tau) \cdot \gamma_{j,i}(\tilde{\mathbf{y}}_\tau, s_\tau, s_{\tau-1}) \quad (3.10)$$

Both recursions utilize a common term  $\gamma_{i,j}(\tilde{\mathbf{y}}_\tau, s_\tau, s_{\tau-1})$ , which is the *branch* metric at time instant  $\tau$ . Furthermore, we have:

$$\begin{aligned}
\gamma_{i,j}(\tilde{\mathbf{y}}_\tau, s_\tau, s_{\tau-1}) &= P(\mathbf{x}_\tau = i, s_\tau, \tilde{\mathbf{y}}_\tau | \mathbf{x}_{\tau-1} = j, s_{\tau-1}, \tilde{\mathbf{Y}}_1^{\tau-1}) \\
&= P(\mathbf{x}_\tau = i | \mathbf{x}_{\tau-1} = j, s_{\tau-1}, \tilde{\mathbf{Y}}_1^{\tau-1}) \cdot P(s_\tau | \mathbf{x}_\tau = i, \mathbf{x}_{\tau-1} = j, s_{\tau-1}, \tilde{\mathbf{Y}}_1^{\tau-1}) \\
&\quad \cdot P(\tilde{\mathbf{y}}_\tau | s_\tau, \mathbf{x}_\tau = i, \mathbf{x}_{\tau-1} = j, s_{\tau-1}, \tilde{\mathbf{Y}}_1^{\tau-1}) \quad (3.11) \\
&\approx P(\mathbf{x}_\tau = i) \cdot P(s_\tau | \mathbf{x}_\tau = i, s_{\tau-1}) \cdot P(\tilde{\mathbf{y}}_\tau | s_\tau, \mathbf{x}_\tau = i)
\end{aligned}$$

Implicit in this equation we assume that interleaved indexes are independent and channel is memoryless. The term  $P(\mathbf{x}_\tau = i)$  is the a priori probability of  $\mathbf{x}_\tau$ , which can be computed in advance by a training data. The term  $P(s_\tau | \mathbf{x}_\tau = i, s_{\tau-1})$  is a branch indicator function, which has a binary form as follows:

$$P(s_\tau | \mathbf{x}_\tau = i, s_{\tau-1}) = \begin{cases} 1, & s_\tau = S(\mathbf{x}_\tau = i, s_{\tau-1}) \\ 0, & \text{otherwise} \end{cases} \quad (3.12)$$

where  $S(\mathbf{x}_\tau = i, s_{\tau-1})$  determines a specific branch corresponding to index  $i$  and previous state  $s_{\tau-1}$ . In bit-based version of BCJR algorithm, we had ignored this kind of indication, since the bit-based trellis structure was relatively simple. However, such a branch indication becomes crucial for grouping the propagated branches. In order to avoid chaos as  $M$  grows large, this branch indicator function eliminates most of unnecessary branches when we focus on index  $i$ . The last term  $P(\tilde{\mathbf{y}}_\tau | s_\tau, \mathbf{x}_\tau = i)$  is the received channel-related information in (3.3) and (3.4). With the above representation of branch metric, recursion formula can be simplified as follows:

$$\alpha_\tau^i(s_\tau) \approx P(\mathbf{x}_\tau = i) \cdot P(\tilde{\mathbf{y}}_\tau | s_\tau, \mathbf{x}_\tau = i) \sum_{s_{\tau-1}} \sum_j \alpha_{\tau-1}^j(s_{\tau-1}) \cdot P(s_\tau | \mathbf{x}_\tau = i, s_{\tau-1}) \quad (3.13)$$

$$\beta_{\tau-1}^i(s_{\tau-1}) \approx \sum_{s_\tau} \sum_j \beta_\tau^j(s_\tau) \cdot P(\mathbf{x}_\tau = j) \cdot P(s_\tau | \mathbf{x}_\tau = j, s_{\tau-1}) \cdot P(\tilde{\mathbf{y}}_\tau | s_\tau, \mathbf{x}_\tau = j) \quad (3.14)$$

Same as Chapter 2, the next task is to determine an index-based version of extrinsic information. Similarly, we extract the intrinsic parts of branch metric:

$$\begin{aligned} \gamma_{i,j}(\tilde{\mathbf{y}}_\tau, s_\tau, s_{\tau-1}) &= P(\mathbf{x}_\tau = i) \cdot P(s_\tau | \mathbf{x}_\tau = i, s_{\tau-1}) \cdot P(\tilde{\mathbf{y}}_\tau | s_\tau, \mathbf{x}_\tau = i) \\ &\triangleq P(\mathbf{x}_\tau = i) \cdot P(\tilde{\mathbf{y}}_\tau^s | \mathbf{x}_\tau = i) \cdot \gamma_{i,j}^{[ext]}(\tilde{\mathbf{y}}_\tau, s_\tau, s_{\tau-1}) \end{aligned} \quad (3.15)$$

where

$$\gamma_{i,j}^{[ext]}(\tilde{\mathbf{y}}_\tau, s_\tau, s_{\tau-1}) = P(s_\tau | \mathbf{x}_\tau = i, s_{\tau-1}) \cdot P(\tilde{\mathbf{y}}_\tau^p | s_\tau, \mathbf{x}_\tau = i) \quad (3.16)$$

contains only the information of parity output corresponds to input  $i$  and an unique state transition. By substituting (3.15) into (3.9) and (3.8), we have the relation between APP and extrinsic information:

$$\begin{aligned} P(\mathbf{x}_\tau = i | \tilde{\mathbf{Y}}_1^\tau) &= C \cdot \sum_{s_\tau} \sum_{s_{\tau-1}} \sum_j \alpha_{\tau-1}^j(s_{\tau-1}) \gamma_{i,j}(\tilde{\mathbf{y}}_\tau, s_\tau, s_{\tau-1}) \beta_\tau^i(s_\tau) \\ &= C \cdot \sum_{s_\tau} \sum_{s_{\tau-1}} \sum_j \alpha_{\tau-1}^j(s_{\tau-1}) \cdot P(\mathbf{x}_\tau = i) \cdot P(\tilde{\mathbf{y}}_\tau^s | \mathbf{x}_\tau = i) \cdot \gamma_{i,j}^{[ext]}(\tilde{\mathbf{y}}_\tau, s_\tau, s_{\tau-1}) \cdot \beta_\tau^i(s_\tau) \quad (3.17) \\ &\triangleq C \cdot P(\mathbf{x}_\tau = i) \cdot P(\tilde{\mathbf{y}}_\tau^s | \mathbf{x}_\tau = i) \cdot P_{CD}^{[ext]}(\mathbf{x}_\tau = i) \end{aligned}$$

where the channel decoder extrinsic information corresponds to input  $i$  at time instant  $\tau$  is:

$$P_{CD}^{[ext]}(\mathbf{x}_\tau = i) = \sum_{s_\tau} \sum_{s_{\tau-1}} \sum_j \alpha_{\tau-1}^j(s_{\tau-1}) \cdot \gamma_{i,j}^{[ext]}(\tilde{\mathbf{y}}_\tau, s_\tau, s_{\tau-1}) \cdot \beta_\tau^i(s_\tau) \quad (3.18)$$

Following the structure of iterative decoding, the a priori information  $P(\mathbf{x}_\tau = i)$  is equal to the feedback of SBS D's extrinsic information. During decoding process, we will update  $P(\mathbf{x}_\tau = i)$ :

$$P(\mathbf{x}_\tau = i) \leftarrow P_{SBS D}^{[ext]}(\mathbf{x}_\tau = i)$$

This is different from the discussion in Chapter 2, where the bit-based algorithm multiplies the bit a priori information  $P(x_l = \pm 1)$  to the feedback extrinsic term  $P_{SBS D}^{[ext]}(x_l = \pm 1)$ . The main reason is that for the computation of  $P_{SBS D}^{[ext]}(x_l = \pm 1)$ , the term  $P(x_l = \pm 1)$  was considered as intrinsic information and was separated from extrinsic term. Here, however, extrinsic information of SBS D is obtained from source redundancy (which will be discussed later). There is a conflict between probabilistic meaning of  $P(\mathbf{x}_\tau = i)$  and  $P_{SBS D}^{[ext]}(\mathbf{x}_\tau = i)$ , and therefore utilizing both terms at the same time will lead to information reuse and serious performance degradation. Thus

we may set  $P(\mathbf{x}_\tau = i) = P_{SBS D}^{[ext]}(\mathbf{x}_\tau = i) = 1/2^M$  at the initialization step. Initiating  $P(\mathbf{x}_\tau = i) = 0^{\text{th}}$ -order source redundancy will result in a better performance at the  $0^{\text{th}}$ -iteration. With *a priori* information replaced by the feedback term, the updated branch metric and APP output of channel decoder can be modified as follows:

$$\gamma_{i,j}(\tilde{\mathbf{y}}_\tau, s_\tau, s_{\tau-1}) = P_{SBS D}^{[ext]}(\mathbf{x}_\tau = i) \cdot P(\tilde{\mathbf{y}}_\tau^s | \mathbf{x}_\tau = i) \cdot \gamma_{i,j}^{[ext]}(\tilde{\mathbf{y}}_\tau, s_\tau, s_{\tau-1}) \quad (3.19)$$

$$P(\mathbf{x}_\tau = i | \tilde{\mathbf{Y}}_1^t) = C \cdot P_{SBS D}^{[ext]}(\mathbf{x}_\tau = i) \cdot P(\tilde{\mathbf{y}}_\tau^s | \mathbf{x}_\tau = i) \cdot P_{CD}^{[ext]}(\mathbf{x}_\tau = i) \quad (3.20)$$

### §3.3 Softbit Source Decoding

In Chapter 2, SBS D algorithm was modified to fit the bit-based system and the derivation of extrinsic information was complicated. Furthermore, the independence assumption of bit  $u_t(\lambda)$  with other bits within the same index  $\mathbf{u}_t$  in (2.23) was not appropriate and may cause performance degradation.

In this chapter, inputs of both decoders are raised up to the index level, and therefore the compatible transformation between bit and index no longer exists. The new derivation of BCJR algorithm over sectionalized trellis allows the SBS D to retain its original form. The intrinsic-extrinsic separation can be directly applied to (2.18). Since the SBS D algorithm often operates “after” channel decoding, we first define the *a priori* information update:

$$P(\mathbf{u}_t = i | \mathbf{u}_{t-1} = j) \leftarrow [P(\mathbf{u}_t = i | \mathbf{u}_{t-1} = j) \cdot P_{CD}^{[ext]}(\mathbf{u}_t = i)] \quad (3.21)$$

Substitute (3.21) into (2.18), the relation between APP and extrinsic information can be revealed:

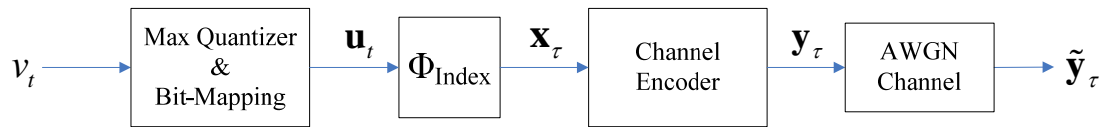
$$\begin{aligned} P(\mathbf{u}_t = i | \tilde{\mathbf{U}}_1^t) &= C \cdot p(\tilde{\mathbf{u}}_t | \mathbf{u}_t = i) \cdot \sum_{j=0}^{2^M-1} P(\mathbf{u}_t = i | \mathbf{u}_{t-1} = j) \cdot P_{CD}^{[ext]}(\mathbf{u}_t = i) \cdot P(\mathbf{u}_{t-1} = j | \tilde{\mathbf{U}}_1^{t-1}) \\ &= C \cdot p(\tilde{\mathbf{u}}_t | \mathbf{u}_t = i) \cdot P_{CD}^{[ext]}(\mathbf{u}_t = i) \cdot \sum_{j=0}^{2^M-1} P(\mathbf{u}_t = i | \mathbf{u}_{t-1} = j) \cdot P(\mathbf{u}_{t-1} = j | \tilde{\mathbf{U}}_1^{t-1}) \quad (3.22) \\ &\triangleq C \cdot p(\tilde{\mathbf{u}}_t | \mathbf{u}_t = i) \cdot P_{CD}^{[ext]}(\mathbf{u}_t = i) \cdot P_{SBS D}^{[ext]}(\mathbf{u}_t = i) \end{aligned}$$

Now the extrinsic information of SBS D becomes:

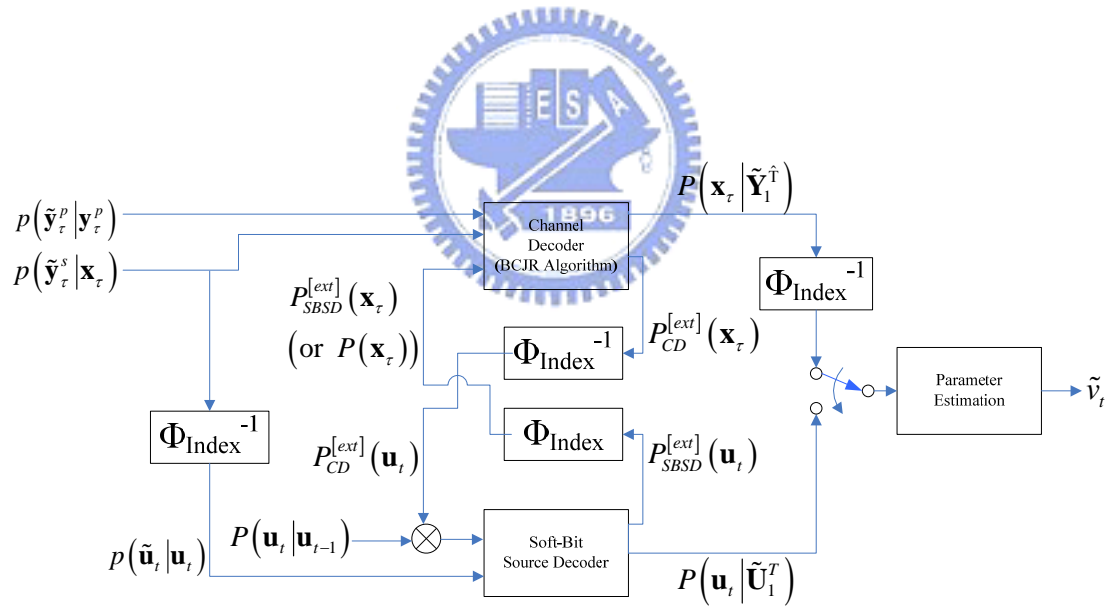
$$P_{SBS D}^{[ext]}(\mathbf{u}_t = i) = \sum_{j=0}^{2^M-1} P(\mathbf{u}_t = i | \mathbf{u}_{t-1} = j) \cdot P(\mathbf{u}_{t-1} = j | \tilde{\mathbf{U}}_1^{t-1}) \quad (3.23)$$

which utilizes 1<sup>st</sup>-order Markov redundancy of source codec. Besides, (3.23) is a

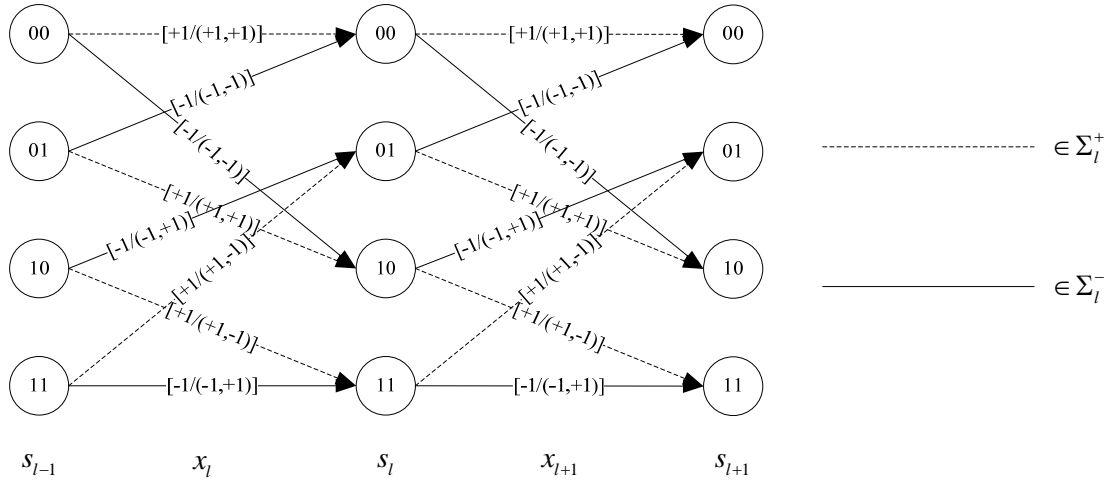
recursive formula, which is updated by APP of index at time  $t-1$ . This term is also fed back to the input of channel decoder as updated a priori information.



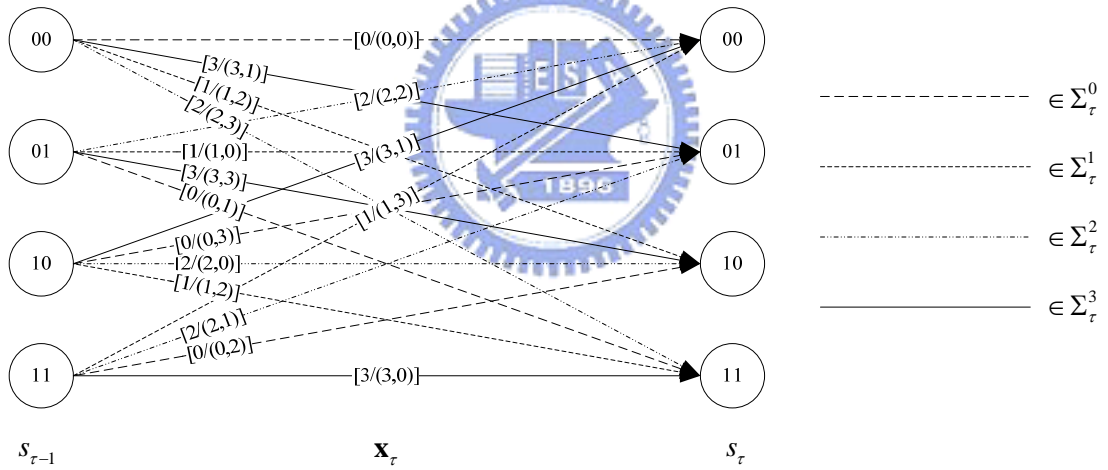
**Figure 3.1: Block diagram of the index-based transmitter**



**Figure 3.2: Block diagram of the index-based receiver**



(a) Bit-level trellis diagram



(b) Sectionalized trellis diagram

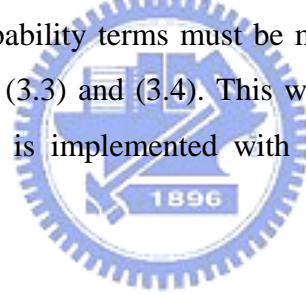
**Figure 3.3: Trellis diagrams used for (a) the encoder and (b) the decoder**



## Chapter 4

### Index-Based ISCD Algorithm in Log-Domain

Log-likelihood ratio (LLR) or log-value (L-value) implementation is widely used for simplifying the bit-based iterative source-channel decoding systems [13]. In this section we would like to transform the index-based iterative source-channel decoding from probability domain to log-domain. The main reason is that  $\mathbf{x}_\tau$  contains  $M$  bits and therefore, at least  $M$  probability terms must be multiplied together to obtain the index-based information, e.g. (3.3) and (3.4). This will result in numerical precision problem when the algorithm is implemented with finite word-length on a digital computer.



#### §4.1 Log-Likelihood Ratio

Consider a bit-based system involving a binary signal  $u_t(\lambda) \in \{+1, -1\}$ , then its L-value is defined as:

$$L(u_t(\lambda)) = \log \frac{P(u_t(\lambda) = +1)}{P(u_t(\lambda) = -1)} \quad (4.1)$$

where  $\log$  represents the natural logarithm. We can deduce that bit-based L-value merges two probabilities in to a single ratio without losing any information. Moreover, all common terms will be cancelled out after the division. Another advantage is that all multiplication has been changed to addition, which is a very good property for circuit realization. However, for index-based system discussed in Chapter 3, we focused on index  $\mathbf{u}_t \in \{0, 1, \dots, 2^M - 1\}$  with  $2^M$  realizations. There are more than two probabilities corresponding to each index. A single L-value is insufficient to preserve all information of these probabilistic terms. Therefore, an index-based

L-value has been proposed in [14] as follows:

$$L(\mathbf{u}_t = i) = \log \frac{P(\mathbf{u}_t = i)}{P(\mathbf{u}_t = 0)}, \quad i = 1, 2, \dots, 2^M - 1 \quad (4.2)$$

which reduces one probability term and enjoys similar advantages as in bit-based case. The final results of this Chapter are summarized in Figure 4.1. The information exchanges are performed in terms of their index-based LLR, which will be discussed in subsequent sections. Notice that channel-related L-value term  $L_{ch}(\mathbf{x}_\tau) = L_{ch}(\mathbf{u}_t)$  is defined by applying (4.2) to (3.3). Since  $\mathbf{x}_\tau \in \{+1, -1\}^M$ ,  $L_{ch}(\mathbf{x}_\tau)$  can be simplified as follows:

$$\begin{aligned} L_{ch}(\mathbf{x}_\tau = i) &= \log \frac{P(\tilde{\mathbf{y}}_\tau^s | \mathbf{x}_\tau = i)}{P(\tilde{\mathbf{y}}_\tau^s | \mathbf{x}_\tau = 0)} \\ &= \log \left\{ \exp \left[ -\frac{E_s}{N_0} \|\tilde{\mathbf{y}}_\tau^s - \mathbf{x}_\tau\|^2 \right]_{\mathbf{x}_\tau=i} \right\} - \log \left\{ \exp \left[ -\frac{E_s}{N_0} \|\tilde{\mathbf{y}}_\tau^s - \mathbf{x}_\tau\|^2 \right]_{\mathbf{x}_\tau=0} \right\} \\ &= \frac{L_c}{2} \cdot \left[ (\tilde{\mathbf{y}}_\tau^s \cdot \mathbf{x}_\tau)_{\mathbf{x}_\tau=i} - (\tilde{\mathbf{y}}_\tau^s \cdot \mathbf{x}_\tau)_{\mathbf{x}_\tau=0} \right], \quad i = 1, 2, \dots, 2^M - 1 \end{aligned} \quad (4.3)$$

where  $L_c = 4E_s/N_0$  and  $(\tilde{\mathbf{y}}_\tau^s \cdot \mathbf{x}_\tau)$  is the dot product of a real-valued vector  $\tilde{\mathbf{y}}_\tau^s$  and a binary vector  $\mathbf{x}_\tau$ . Since the computation of MMSE parameter estimate still needs the APP values at the end of decoding process, we define an inversion of L-value definition for channel decoder's output:

$$P(\mathbf{x}_\tau = i | \tilde{\mathbf{Y}}_1^\dagger) = \begin{cases} \frac{1}{1 + \sum_{j=1}^{2^M} \exp \left[ L(\mathbf{x}_\tau = j | \tilde{\mathbf{Y}}_1^\dagger) \right]}, & i = 0 \\ \frac{\exp \left[ L(\mathbf{x}_\tau = i | \tilde{\mathbf{Y}}_1^\dagger) \right]}{1 + \sum_{j=1}^{2^M} \exp \left[ L(\mathbf{x}_\tau = j | \tilde{\mathbf{Y}}_1^\dagger) \right]}, & i = 1, 2, \dots, 2^M - 1 \end{cases} \quad (4.4)$$

which contains a normalization and is performed at the end of both decoding algorithms.

## §4.2 Modified BCJR Algorithm in Log-Domain

In previous work [11][15], log-domain BCJR algorithm is derived directly from conventional bit-based BCJR algorithm. In this section we will derive the log-domain modified BCJR algorithm by defining the log-domain forward, backward, and branch

metric. We begin with applying (4.2) to index APP in (3.7) and (3.8):

$$\begin{aligned}
L(\mathbf{x}_\tau = i | \tilde{\mathbf{Y}}_1^{\hat{}}) &\triangleq \log \frac{P(\mathbf{x}_\tau = i | \tilde{\mathbf{Y}}_1^{\hat{}})}{P(\mathbf{x}_\tau = 0 | \tilde{\mathbf{Y}}_1^{\hat{}})} \\
&= \log \frac{\sum_{s_\tau} \alpha_\tau^i(s_\tau) \cdot \beta_\tau^i(s_\tau)}{\sum_{s_\tau} \alpha_\tau^0(s_\tau) \cdot \beta_\tau^0(s_\tau)}, \quad i = 1, 2, \dots, 2^M - 1.
\end{aligned} \tag{4.5}$$

Recognizing that summations cannot be performed directly in log-domain, we suggest to simplify the calculations by using the following equation (see Appendix A):

$$\max^*(a, b) \equiv \log(e^a + e^b) = \max(a, b) + \log(1 + e^{-|a-b|}) \tag{4.6}$$

where  $a, b$  are arbitrary real numbers. Since the summation includes  $2^u$  terms corresponding to all possible states of  $s_\tau$ , we can apply  $\max^*$  function defined in (4.6) to sums of more than two exponential terms by:

$$\max^*(a, b, c) \equiv \log(e^a + e^b + e^c) = \max^*[\max^*(a, b), c] \tag{4.7}$$

where  $c$  is an arbitrary real number. Since the summands in (4.5) must be in exponential form, we thus compute the natural logarithm of forward metric by:

$$\begin{aligned}
\bar{\alpha}_\tau^i(s_\tau) &\triangleq \log \alpha_\tau^i(s_\tau) \\
&= \log P(\mathbf{x}_\tau = i) + \log P(\tilde{\mathbf{y}}_\tau | \mathbf{x}_\tau = i, s_\tau) + \log \left\{ \sum_{s_{\tau-1}} P(s_\tau | \mathbf{x}_\tau = i, s_{\tau-1}) \sum_j \exp[\log \alpha_{\tau-1}^j(s_{\tau-1})] \right\} \\
&= \log P(\mathbf{x}_\tau = i) + \log P(\tilde{\mathbf{y}}_\tau | \mathbf{x}_\tau = i, s_\tau) + \max_{s_{\tau-1}}^* \left\{ \log P(s_\tau | \mathbf{x}_\tau = i, s_{\tau-1}) + \max_j^* [\bar{\alpha}_{\tau-1}^j(s_{\tau-1})] \right\}
\end{aligned} \tag{4.8}$$

where we apply (4.7) twice for 2<sup>nd</sup> equality and

$$\max_k^* [f(k)] \triangleq \max^* [f(1), f(2), \dots, f(K)], \quad k \in \{1, 2, \dots, K\}$$

Note that the L-value of branch indicator function is defined by

$$\log P(s_\tau | \mathbf{x}_\tau = i, s_{\tau-1}) = \begin{cases} 0, & s_\tau = S(\mathbf{x}_\tau = i, s_{\tau-1}) \\ -\infty, & \text{otherwise} \end{cases}$$

Similarly, we have the natural logarithm of backward metric:

$$\begin{aligned}
\bar{\beta}_{\tau-1}^j(s_{\tau-1}) &\triangleq \log \beta_{\tau-1}^j(s_{\tau-1}) \\
&= \log \sum_j P(\mathbf{x}_\tau = j) \sum_{s_\tau} \beta_\tau^j(s_\tau) \cdot P(s_\tau | \mathbf{x}_\tau = j, s_{\tau-1}) \cdot P(\tilde{\mathbf{y}}_\tau | s_\tau, \mathbf{x}_\tau = j) \\
&= \max_j^* \left\{ \log P(\mathbf{x}_\tau = j) + \max_{s_\tau}^* [\bar{\beta}_\tau^j(s_\tau) + \log P(s_\tau | \mathbf{x}_\tau = j, s_{\tau-1}) + \log P(\tilde{\mathbf{y}}_\tau | s_\tau, \mathbf{x}_\tau = j)] \right\}
\end{aligned} \tag{4.9}$$

By applying (4.7) to (4.5), we also obtain the APP in terms of (4.8) and (4.9) as

follows:

$$L(\mathbf{x}_\tau = i | \tilde{\mathbf{Y}}_1^\dagger) = \log \left\{ \sum_{s_\tau} \exp[\bar{\alpha}_\tau^i(s_\tau)] \cdot \exp[\bar{\beta}_\tau^i(s_\tau)] \right\} - \log \left\{ \sum_{s_\tau} \exp[\bar{\alpha}_\tau^0(s_\tau)] \cdot \exp[\bar{\beta}_\tau^0(s_\tau)] \right\} \quad (4.10)$$

$$= \max_{s_\tau}^* [\bar{\alpha}_\tau^i(s_\tau) + \bar{\beta}_\tau^i(s_\tau)] - \max_{s_\tau}^* [\bar{\alpha}_\tau^0(s_\tau) + \bar{\beta}_\tau^0(s_\tau)]$$

Since the APP is calculated explicitly, the extrinsic information can be obtained by subtracting the L-values of intrinsic information. By interpretation of (3.17), we split the L-value of APP into:

$$L(\mathbf{x}_\tau = i | \tilde{\mathbf{Y}}_1^\dagger) = \log \frac{P(\mathbf{x}_\tau = i)}{P(\mathbf{x}_\tau = 0)} + \log \frac{P(\tilde{\mathbf{y}}_\tau^s | \mathbf{x}_\tau = i)}{P(\tilde{\mathbf{y}}_\tau^s | \mathbf{x}_\tau = 0)} + \log \frac{P_{CD}^{[ext]}(\mathbf{x}_\tau = i)}{P_{CD}^{[ext]}(\mathbf{x}_\tau = 0)} \quad (4.11)$$

$$\triangleq L_a(\mathbf{x}_\tau = i) + L_{ch}(\mathbf{x}_\tau = i) + L_{CD}^{[ext]}(\mathbf{x}_\tau = i)$$

where the three terms represent the L-values of a priori, channel-related, and extrinsic information derived from channel decoding. The last term can also be derived explicitly

$$L_{CD}^{[ext]}(\mathbf{x}_\tau = i) = \max_{s_\tau}^* \left\{ \bar{\beta}_\tau^i(s_\tau) + \log P(\tilde{\mathbf{y}}_\tau^p | s_\tau, \mathbf{x}_\tau = i) + \max_{s_{\tau-1}} \left\{ \log P(s_\tau | \mathbf{x}_\tau = i, s_{\tau-1}) + \max_j^* [\bar{\alpha}_{\tau-1}^j(s_{\tau-1})] \right\} \right\} \quad (4.12)$$

$$- \max_{s_\tau}^* \left\{ \bar{\beta}_\tau^0(s_\tau) + \log P(\tilde{\mathbf{y}}_\tau^p | s_\tau, \mathbf{x}_\tau = 0) + \max_{s_{\tau-1}} \left\{ \log P(s_\tau | \mathbf{x}_\tau = 0, s_{\tau-1}) + \max_j^* [\bar{\alpha}_{\tau-1}^j(s_{\tau-1})] \right\} \right\}$$

Furthermore, if we define the L-value feedback from softbit source decoder:

$$L_{SBSD}^{[ext]}(\mathbf{x}_\tau = i) \triangleq \log \frac{P_{SBSD}^{[ext]}(\mathbf{x}_\tau = i)}{P_{SBSD}^{[ext]}(\mathbf{x}_\tau = 0)}, \quad i = 1, 2, \dots, 2^M - 1 \quad (4.13)$$

Following the description of how to obtain (3.19) in Chapter 3, we will modify the iterative form of recursion formula (4.8) and (4.9) as follows:

$$\bar{\alpha}_\tau^i(s_\tau) = L_{SBSD}^{[ext]}(\mathbf{x}_\tau = i) + \log P(\tilde{\mathbf{y}}_\tau | \mathbf{x}_\tau = i, s_\tau) + \max_{s_{\tau-1}}^* \left\{ \log P(s_\tau | \mathbf{x}_\tau = i, s_{\tau-1}) + \max_j^* [\bar{\alpha}_{\tau-1}^j(s_{\tau-1})] \right\} \quad (4.14)$$

$$\bar{\beta}_{\tau-1}^i(s_{\tau-1}) = \max_j^* \left\{ L_{SBSD}^{[ext]}(\mathbf{x}_\tau = j) + \max_{s_\tau}^* \left[ \bar{\beta}_\tau^j(s_\tau) + \log P(s_\tau | \mathbf{x}_\tau = j, s_{\tau-1}) + \log P(\tilde{\mathbf{y}}_\tau | s_\tau, \mathbf{x}_\tau = j) \right] \right\} \quad (4.15)$$

Finally, we have the iterative form of index APP in (4.11):

$$L(\mathbf{x}_\tau = i | \tilde{\mathbf{Y}}_1^\dagger) = L_{SBSD}^{[ext]}(\mathbf{x}_\tau = i) + L_{ch}(\mathbf{x}_\tau = i) + L_{CD}^{[ext]}(\mathbf{x}_\tau = i) \quad (4.16)$$

### §4.3 Softbit Source Decoding in Log-Domain

The bit-based L-value of SBSBD has been proposed by M. Adrat [4]. Here we follow the similar procedure in the previous section to derive the log-domain index-based

SBSD. We first apply the definition (4.2) to (2.18):

$$\begin{aligned}
L(\mathbf{u}_t = i | \tilde{\mathbf{U}}_1^t) &\triangleq \frac{P(\mathbf{u}_t = i | \tilde{\mathbf{U}}_1^t)}{P(\mathbf{u}_t = 0 | \tilde{\mathbf{U}}_1^t)} \\
&= \log \frac{p(\tilde{\mathbf{u}}_t | \mathbf{u}_t = i) \cdot \sum_{j=0}^{2^M-1} P(\mathbf{u}_t = i | \mathbf{u}_{t-1} = j) \cdot P(\mathbf{u}_{t-1} = j | \tilde{\mathbf{U}}_1^{t-1})}{p(\tilde{\mathbf{u}}_t | \mathbf{u}_t = 0) \cdot \sum_{j=0}^{2^M-1} P(\mathbf{u}_t = 0 | \mathbf{u}_{t-1} = j) \cdot P(\mathbf{u}_{t-1} = j | \tilde{\mathbf{U}}_1^{t-1})}, \quad i = 1, 2, \dots, 2^M-1
\end{aligned} \tag{4.17}$$

From the discussion in Section 3.3, we substitute (3.21) to (4.17):

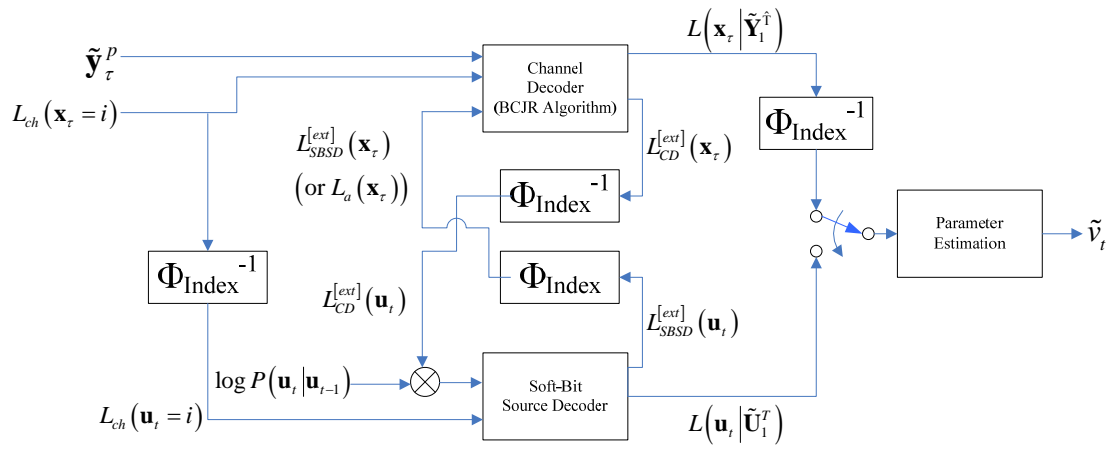
$$\begin{aligned}
L(\mathbf{u}_t = i | \tilde{\mathbf{U}}_1^t) &= \log \frac{p(\tilde{\mathbf{u}}_t | \mathbf{u}_t = i)}{p(\tilde{\mathbf{u}}_t | \mathbf{u}_t = 0)} + \log \frac{P_{CD}^{[extr]}(\mathbf{u}_t = i)}{P_{CD}^{[extr]}(\mathbf{u}_t = 0)} + \log \frac{\sum_{j=0}^{2^M-1} P(\mathbf{u}_t = i | \mathbf{u}_{t-1} = j) \cdot P(\mathbf{u}_{t-1} = j | \tilde{\mathbf{U}}_1^{t-1})}{\sum_{j=0}^{2^M-1} P(\mathbf{u}_t = 0 | \mathbf{u}_{t-1} = j) \cdot P(\mathbf{u}_{t-1} = j | \tilde{\mathbf{U}}_1^{t-1})} \\
&\triangleq L_{ch}(\mathbf{u}_t = i) + L_{CD}^{[extr]}(\mathbf{u}_t = i) + L_{SBSD}^{[extr]}(\mathbf{u}_t = i)
\end{aligned} \tag{4.18}$$

where the three terms represent the de-interleaved (4.3), the de-interleaved extrinsic information in (4.16), and the extrinsic information derived from source decoding:

$$L_{SBSD}^{[extr]}(\mathbf{u}_t = i) = \log \frac{\sum_{j=0}^{2^M-1} P(\mathbf{u}_t = i | \mathbf{u}_{t-1} = j) \cdot P(\mathbf{u}_{t-1} = j | \tilde{\mathbf{U}}_1^{t-1})}{\sum_{j=0}^{2^M-1} P(\mathbf{u}_t = 0 | \mathbf{u}_{t-1} = j) \cdot P(\mathbf{u}_{t-1} = j | \tilde{\mathbf{U}}_1^{t-1})} \tag{4.19}$$

By applying (4.7), we can further simplify the above calculation as follows:

$$\begin{aligned}
L_{SBSD}^{[extr]}(\mathbf{u}_t = i) &= \max_j^* \left[ \log P(\mathbf{u}_t = i | \mathbf{u}_{t-1} = j) + \log P(\mathbf{u}_{t-1} = j | \tilde{\mathbf{U}}_1^{t-1}) \right] \\
&\quad - \max_j^* \left[ \log P(\mathbf{u}_t = 0 | \mathbf{u}_{t-1} = j) + \log P(\mathbf{u}_{t-1} = j | \tilde{\mathbf{U}}_1^{t-1}) \right] \\
&= \max_j^* \left[ \log P(\mathbf{u}_t = i | \mathbf{u}_{t-1} = j) + L(\mathbf{u}_{t-1} = j | \tilde{\mathbf{U}}_1^{t-1}) \right] \\
&\quad - \max_j^* \left[ \log P(\mathbf{u}_t = 0 | \mathbf{u}_{t-1} = j) + L(\mathbf{u}_{t-1} = j | \tilde{\mathbf{U}}_1^{t-1}) \right]
\end{aligned} \tag{4.20}$$



**Figure 4.1: Log-domain index-based ISCD**



# Chapter 5

## Simulation Results

### §5.1 Experimental Setup

Consider a source parameter  $v_t$ , which is modeled by a first-order Gauss-Markov process with correlation factor  $\rho = 0.95$  and its variance is normalized to  $\sigma_v^2 = 1$ . The parameters  $v_t$  are individually quantized by an 8-level Max quantizer using  $M = 3$  bits, resulting in a bit-pattern  $\mathbf{u}_t$  in the form of natural binary bit-assignment.

A total of  $T=100$  bit-patterns  $\mathbf{U}_1^T$  are passed to a 10-by-30 binary block interleaver, which inputs bit information into a 10-by-30 matrix horizontally and outputs them vertically as the matrix is filled up. A bit-based ISCD system uses the bit-scrambled interleaver, whose output  $\mathbf{x}_1^L$  is considered as a block of uncorrelated bits. On the other hand, an index-based ISCD system uses the index-based interleaver so that the bit-order within each  $\mathbf{u}_t$  remains unchanged and the output  $\mathbf{X}_1^T$  is considered as a block of uncorrelated indexes. This results in a 10-by-10 decimal block interleaver.

Channel encoding is performed by (2,1)–RSC code with constraint length  $\nu = 2$  and generator matrix

$$\mathbf{G}(D) = \begin{bmatrix} 1 & \mathbf{g}^{(1)}(D) \\ & \mathbf{g}^{(0)}(D) \end{bmatrix} = \begin{bmatrix} 1 & (1+D^2) \\ & (1+D+D^2) \end{bmatrix}.$$

Additional redundant termination bits are preserved after encoding each block of interleaver's output, which makes the overall coding rate 300/604, slightly lower than 1/2. The RSC encoded output is then modulated using the BPSK and transmitted over an AWGN channel specified by a known SNR, denoted as  $E_s/N_0$ .

At the receiver side, decoding will be performed iteratively by following the steps presented in sections 2.1 and 3.1. Since the parameters  $v_t$  are modeled by a recursive

process, the SBSB results  $P(\mathbf{u}_T|\tilde{\mathbf{U}}_1^T)$  at the current block are treated as the initialization values of  $P(\mathbf{u}_0|\tilde{\mathbf{U}}_1^T)$  at the next block. In other words, softbit source decoder estimates a source with nearly infinite-length. Specifically, 30000 first-order Gauss-Markov parameters are generated at each test. We measure the performance of both ISCD systems by calculating the parameter SNR:

$$\text{Parameter SNR (dB)} = 10 \cdot \log_{10} \frac{E[v_t^2]}{E[(v_t - \hat{v}_t)^2]}. \quad (5.1)$$

## §5.2 Performance of Bit-Based ISCD

The performances of bit-based ISCD over different values of channel SNR are shown in Table 5.1 and plotted in Figure 5.1. The effect of information exchanging allows the parameter SNR to increase as the number of iteration increases. The results of 0<sup>th</sup>, 0+<sup>th</sup>, 1<sup>st</sup>, 1+<sup>th</sup>, and 2<sup>nd</sup> iteration are shown. At the 0<sup>th</sup> iteration, the calculation only involves channel-related information and bit a priori information

$$P(\mathbf{u}_t = i | \tilde{\mathbf{U}}_1^T) = C \cdot \prod_{\substack{\lambda=1 \\ \{\mathbf{u}_t=i\}}}^M P(u_t(\lambda)) \cdot P(\tilde{u}_t(\lambda) | u_t(\lambda)) \quad (5.2)$$

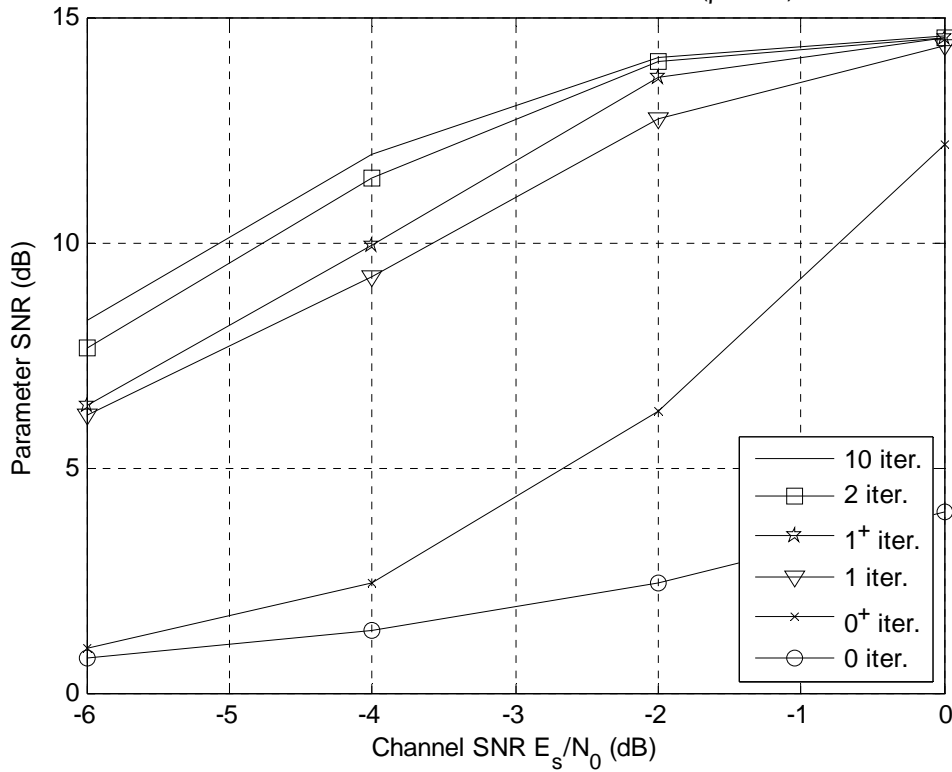
which utilizes neither parity bits nor source residual redundancy. Furthermore, we also found that the performance of 10<sup>th</sup> iteration is considered as the upper bound of the bit-based ISCD system. It is noticed that when channel SNR is low ( $\leq -2dB$ ), the largest performance gap occurs between 0+<sup>th</sup> and 1<sup>st</sup> iterations, allowing to increase the parameter SNR up to 6.8dB. This indicates that utilizing the strong correlation property of source can provide a great help to combat heavily noisy environment. When channel SNR is around 0dB, the performance of 0+<sup>th</sup> iteration raises up to 12dB, and the subsequent iterations quickly meets the saturation. The protection of channel coding works well for moderate  $E_s/N_0$ . When channel SNR is more than 3dB, channel decoding is sufficient for achieving the upper bound 14.62dB of Max quantizer.



Iteration	Channel SNR (dB)			
	-6	-4	-2	0
0	0.784	1.383	2.423	4.032
0+	1.005	2.441	6.263	12.178
1	6.162	9.227	12.757	14.371
1+	6.379	9.918	13.653	14.538
2	7.655	11.431	14.009	14.546
10	8.265	11.967	14.084	14.569

**Table 5.1: Performance of bit-based ISCD**

Performance of Bit-Based ISCD with  $10^*30$  Block Interleaver ( $\rho=0.95$ ), Natural Binary BA



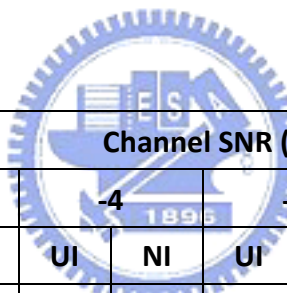
**Figure 5.1: Performance of bit-based ISCD**

### §5.3 Performance of Index-Based ISCD

As mentioned in section 5.1, the parameter setting and the  $E_s/N_0$  range for simulation are the same for bit-based and index-based ISCD. The difference is that two kinds of initialization settings are considered in this section. One is the uniform initialization (UI), which assumes the value of  $P_{SBS D}^{[ext]}(\mathbf{x}_\tau = i) = 1/2^M \forall i$  in (3.19)

and (3.20) at the  $0^{+}$  iteration. In other words, uniform initialization is used with no a priori knowledge (NAK). On the other hand, the non-uniform initialization (NI) sets  $P_{SBS D}^{[ext]}(\mathbf{x}_\tau = i) = P(\mathbf{x}_\tau = i)$ , where  $P(\mathbf{x}_\tau = i)$  is the  $0^{\text{th}}$ -order source redundancy. In other words, non-uniform initialization is used with  $0^{\text{th}}$ -order a priori knowledge (AK0).

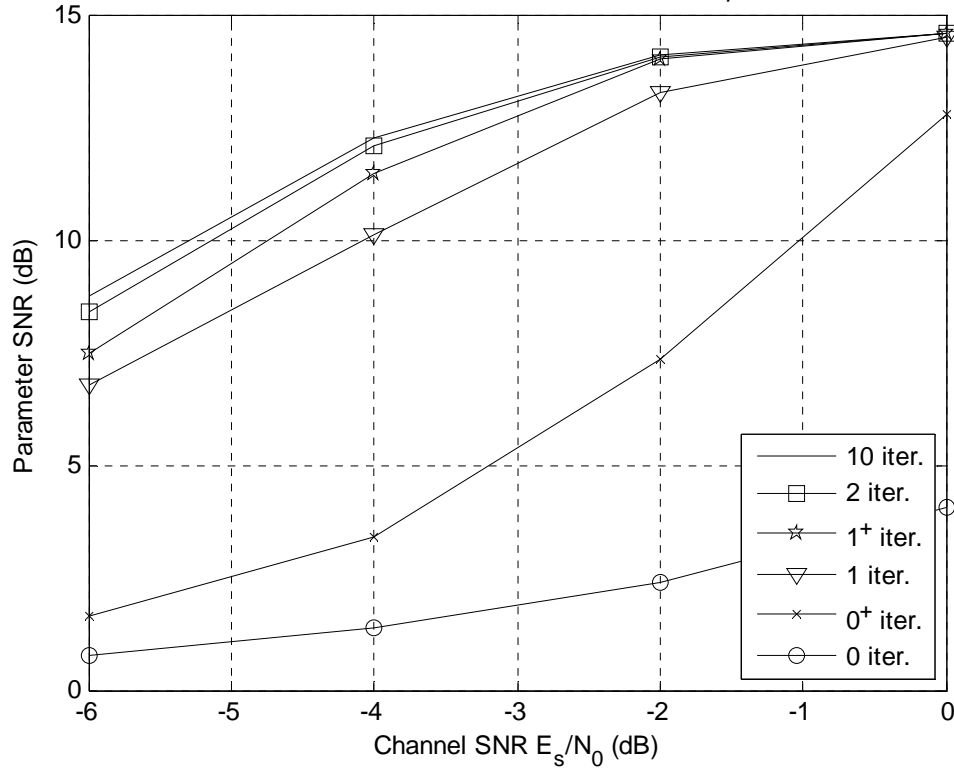
The results are shown in Table 5.2 and plotted in Figure 5.2. The trend of performance improvement is almost the same as in bit-based system. We observe that although NI is better than UI for several iterations at the beginning, they both converge to a similar result for  $1^{+}$  and subsequent iterations. We can thus consider UI as an alternate solution when AK0 training data cannot be acquired.



Iteration	Channel SNR (dB)							
	-6		-4		-2		0	
	UI	NI	UI	NI	UI	NI	UI	NI
<b>0+</b>	1.095	1.63	2.733	3.379	6.759	7.345	12.555	12.768
<b>1</b>	6.659	6.79	9.924	10.106	13.15	13.246	14.466	14.48
<b>1+</b>	7.402	7.463	11.417	11.473	14.016	14.02	14.57	14.57
<b>2</b>	8.421	8.409	12.054	12.075	14.082	14.074	14.571	14.571
<b>10</b>	8.766	8.763	12.256	12.256	14.085	14.085	14.572	14.572

**Table 5.2: Performance of index-based ISCD with uniform initialization (UI) and non-uniform initialization (NI)**

Performance of Index-Based ISCD with  $10 \times 30$  Block Interleaver ( $\rho=0.95$ ), Natural Binary BA



**Figure 5.2: Performance of index-based ISCD with non-uniform initialization (NI)**

Since the index-based ISCD with NI gives better performance, we will omit the UI results in subsequent comparisons. In Table 5.3 and Figure 5.3, we further present the performances of index-based ISCD operating in the log-domain and in probability domain. It can be seen that numerical results are very close, indicating that the transformation from probability to log-domain is a better choice for real-time implementation.

Iteration	Channel SNR (dB)							
	-6		-4		-2		0	
	Prob.	Log	Prob.	Log	Prob.	Log	Prob.	Log
<b>0+</b>	1.63	1.634	3.379	3.383	7.345	7.365	12.768	12.753
<b>1</b>	6.79	6.783	10.106	10.121	13.246	13.257	14.48	14.468
<b>1+</b>	7.463	7.457	11.473	11.496	14.02	14.025	14.57	14.558
<b>2</b>	8.409	8.404	12.075	12.083	14.074	14.082	14.571	14.559

**Figure 5.3: Performance of index-based ISCD in probability and log-domain**

Performance of Index-Based ISCD with  $10^*30$  Block Interleaver ( $\rho=0.95$ ), Natural Binary BA

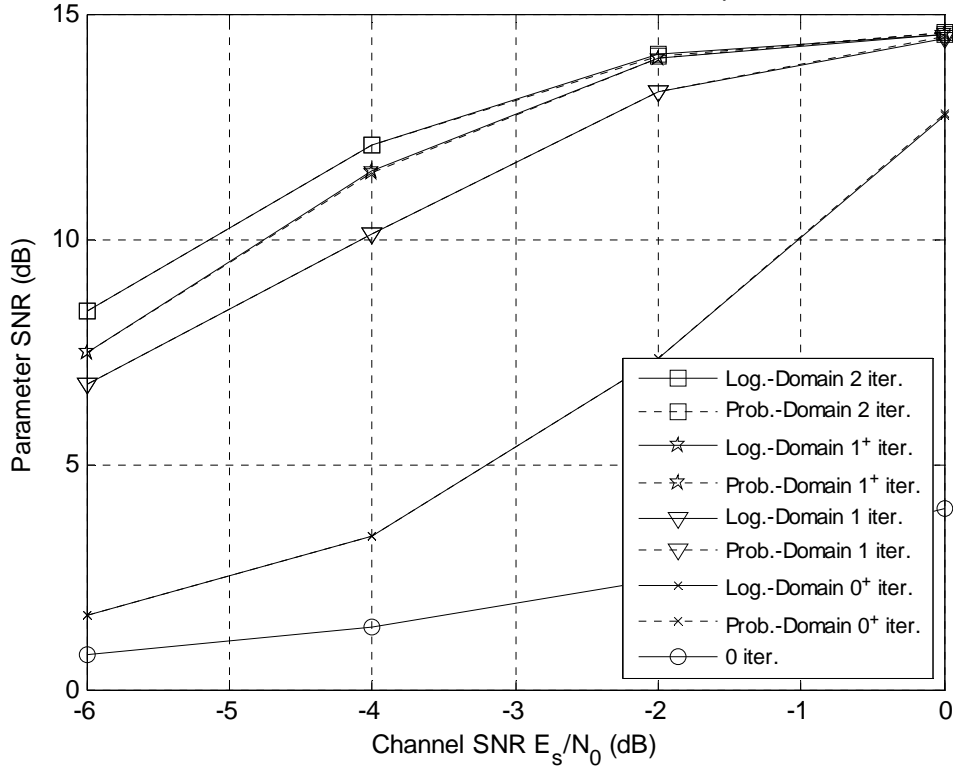


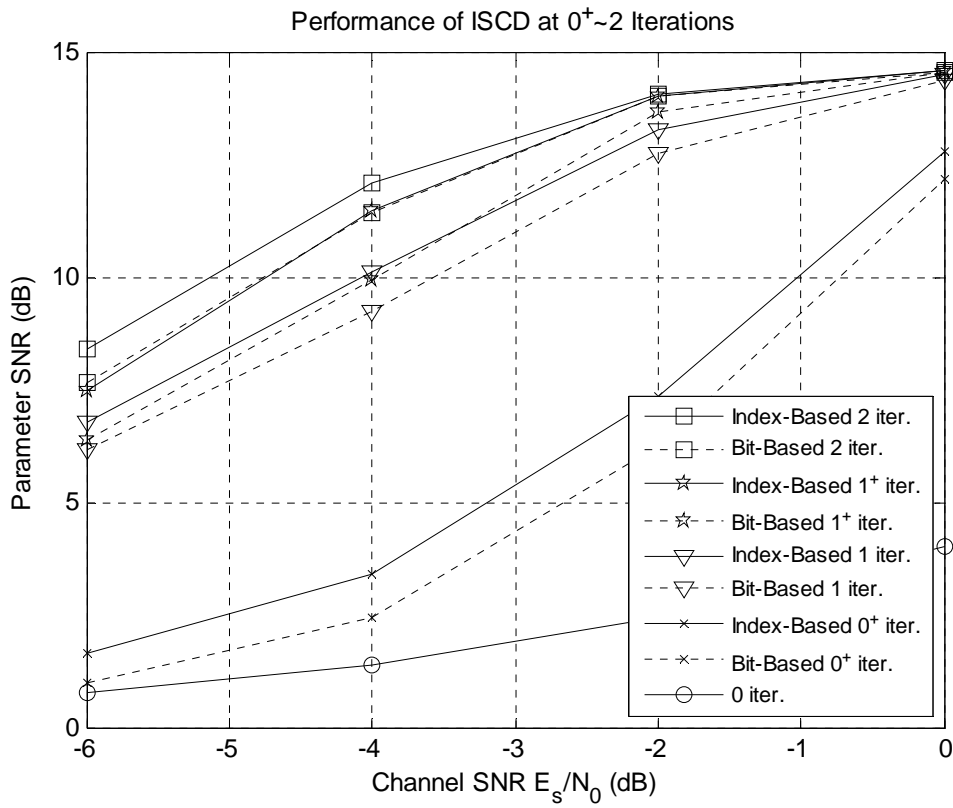
Figure 5.3: Performance of index-based ISCD in probability and log-domain

## §5.4 Comparison of Index-Based and Bit-Based ISCD

Table 5.4 and Figure 5.4 compare the performances of bit-based (BB) and index-based (IB) ISCD systems at the 0<sup>th</sup>, 1<sup>st</sup>, 1<sup>th</sup>, and 2<sup>nd</sup> iterations. The results of the latter system come from the NI case in section 5.2. At the beginning, the performances of 0<sup>th</sup> and 1<sup>st</sup> iterations present apparent differences. Index-based ISCD outperforms the bit-based ISCD by at most 1.1dB. The largest performance gap occurs at the 1<sup>th</sup> iteration, the performance gain can be up to 1.5dB. This shows that information exchange at the index level is more effective and the loss of using (2.5) in bit-based system has been compensated. For the 2<sup>nd</sup> iteration, both systems saturate as  $E_s/N_0 \geq -2dB$ . However, there still exist performance gaps when transmission over the AWGN channel with low  $E_s/N_0$ . In average, the bit-based system needs to execute two more steps, or one decoding algorithm in estimation process presented in Chapter 2, to keep up with the index-based system.

Iteration	Channel SNR (dB)							
	-6		-4		-2		0	
	BB	IB	BB	IB	BB	IB	BB	IB
0+	1.005	1.63	2.441	3.379	6.263	7.345	12.178	12.768
1	6.162	6.79	9.227	10.106	12.757	13.246	14.371	14.48
1+	6.379	7.463	9.918	11.473	13.653	14.02	14.538	14.57
2	7.655	8.409	11.431	12.075	14.009	14.074	14.546	14.571

**Table 5.4: Comparison of bit-based and index-based ISCD**



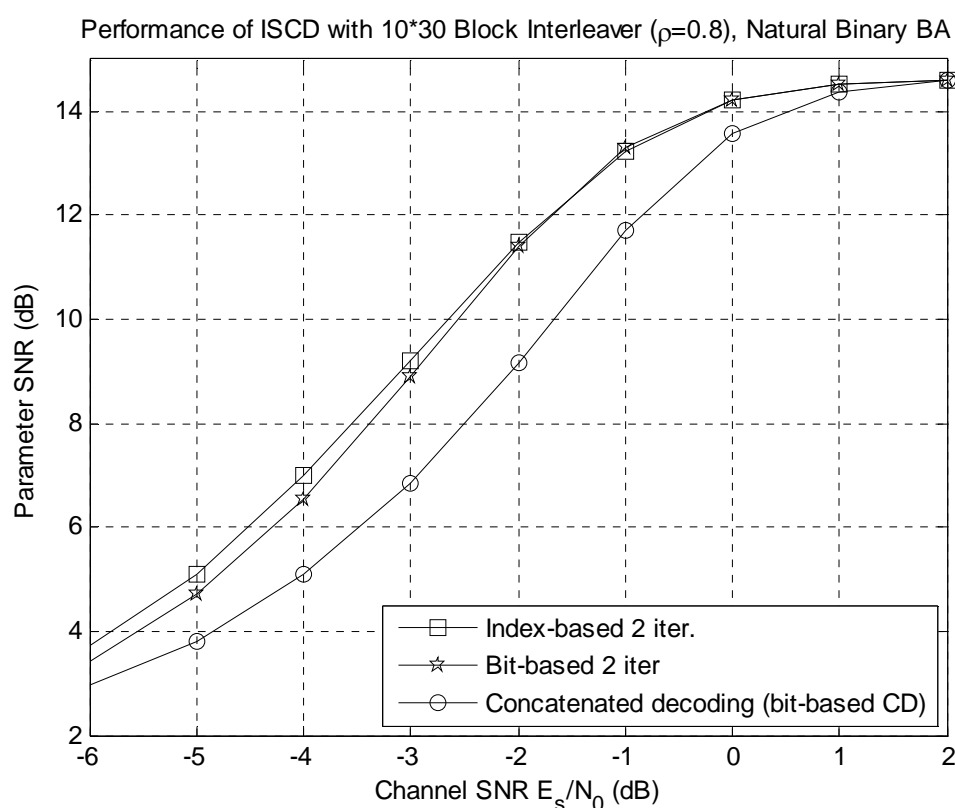
**Figure 5.4: Comparison of bit-based and index-based ISCD**

Figure 5.5 and Figure 5.6 show the index-based as well as the bit-based ISCD compared to the concatenated decoding with source correlation factor 0.8 and 0.95. Utilizing the same amount of redundancy, the concatenated decoding represents the optimal conventional separate decoding scheme. It concatenates the channel and source decoder in serial without interleaver. The a posteriori probabilities estimated by channel decoder are considered as the channel-related information at source

decoding, i.e. the channel decoder is merged with the AWGN channel to produce an equivalently more robust channel. The bit-based ISCD exhibits a peak  $2dB$  performance gain compared to concatenated decoding when correlation factor  $\rho = 0.8$  and about  $2.5dB$  when  $\rho = 0.95$ . This indicates that the ISCD, which is a type of JSCD design, can make both decoder benefit from each other and thus outperforms the conventional separate design. Further investigation shows that the performance gain between index-based and bit-based ISCD increases for more correlated source.

Decoder Type	Channel SNR (dB)								
	-6	-5	-4	-3	-2	-1	0	1	2
<b>Concatenated Decoding</b>	2.951	3.824	5.101	6.864	9.182	11.695	13.558	14.36	14.59
<b>Bit-Based ISCD (2 iter.)</b>	3.433	4.716	6.56	8.906	11.396	13.29	14.229	14.529	14.593
<b>Index-Based ISCD (2 iter.)</b>	3.712	5.092	6.982	9.22	11.503	13.251	14.203	14.531	14.599

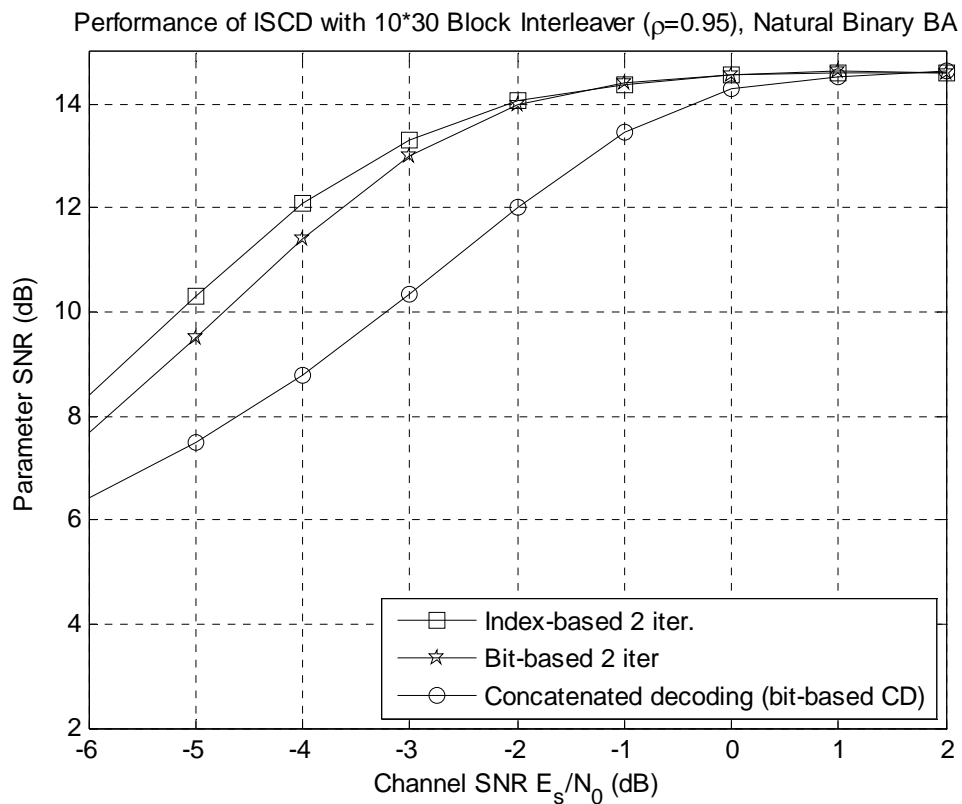
**Table 5.5: Comparison of bit-based ISCD, index-based ISCD, and concatenated decoding for source correlation factor  $\rho = 0.8$**



**Figure 5.5: Comparison of bit-based ISCD, index-based ISCD, and concatenated decoding for source correlation factor  $\rho = 0.8$**

Decoder Type	Channel SNR (dB)			
	-5	-4	-3	-2
Concatenated Decoding	7.488	8.773	10.353	12.035
Bit-Based ISCD (2 iter.)	9.507	11.415	13.016	14.001
Index-Based ISCD (2 iter.)	10.32	12.097	13.321	14.07
Decoder Type	Channel SNR (dB)			
	-1	0	1	2
Concatenated Decoding	13.472	14.278	14.535	14.624
Bit-Based ISCD (2 iter.)	14.423	14.574	14.621	14.606
Index-Based ISCD (2 iter.)	14.392	14.559	14.613	14.612

**Table 5.6: Comparison of bit-based ISCD, index-based ISCD, and concatenated decoding for source correlation factor  $\rho = 0.95$**



**Figure 5.6: Comparison of bit-based ISCD, index-based ISCD, and concatenated decoding for source correlation factor  $\rho = 0.95$**

## §5.5 Complexity Analysis of Index-Based ISCD

In this section, we will give an analysis of the computational complexity required for both ISCD systems. We assume  $L_a(\mathbf{x}_\tau)$  and  $L_{ch}(\mathbf{x}_\tau)$  are known. The complexity of BCJR algorithm can be measured by the number of branch metrics, forward and backward metrics, and the extrinsic information of channel decoding (EXT-CD). The complexity of SBSB can be measured by the number of the recursive terms and the extrinsic information of source decoding (EXT-SB). We first consider the memoryless and  $M$ -bit sectionalized BCJR algorithm for channel decoding, as discussed in section 3.2. The number of branch metrics is directly related to the total number of branches out of each state. Although the calculation of branch metrics only involves addition, all the metric-values must be stored and therefore consuming the memory. There are  $2^M$  branches leaving each state and  $2^\nu$  states at each time instant for an  $(n, 1, \nu)$ -RSC code, which results in a total of  $2^{\nu+M}$  branches per state transition. Since the size of interleaver is  $L$  bits, there are  $L/M$  state transitions per decoding block and therefore the total number of branch will approximately equal to  $2^{\nu+M} \cdot L/M$ . The number of forward/backward metrics is related to the total number of states. The calculation of forward/backward metric involves addition and a recursive form of  $\max^*(\cdot)$  function, all the metric-values must also be stored. There are  $2^\nu$  states at each time instant and  $L/M$  state transitions per decoding block, resulting in a total of  $2^\nu \cdot L/M$  states per decoding block. The number of EXT-CD equals to  $2^M$  at each time instant and  $2^M \cdot L/M$  per decoding block. The calculation of EXT-CD involves addition and a recursive form of  $\max^*(\cdot)$  function, all the values must also be stored. In the case of bit-based ISCD ( $M = 1, \nu = 2, L = 300$ ), the numbers of branch metrics, forward/backward metrics, and EXT-CD are 2400, 1200, and 600, respectively. In the case of index-based ISCD ( $M = 3, \nu = 2, L = 300$ ), the corresponding numbers are 3200, 400, and 800. The complexity of channel decoder is summarized in Table 5.5.

Next, we consider the SBSB algorithm based on  $M$ -bit index by assuming that the index log-APPs from previous time instant are known. We state the recursive term in log-domain as follows:

$$\max_j^* \left[ \log P(\mathbf{u}_t = i | \mathbf{u}_{t-1} = j) + L(\mathbf{u}_{t-1} = j | \tilde{\mathbf{U}}_1^{t-1}) \right], \quad i = 0, 1, \dots, 2^M - 1 \quad (5.3)$$

In order to obtain log-APPs, the recursive terms must be calculated over all possible



index-values and therefore a total of  $2^M$  recursive terms are calculated for decoding one parameter. The decoding-block size is  $L$  bits or equivalent  $L/M$  indexes. Since the computation of log-APPs from (5.2) only involves addition of (5.2),  $L_a(\mathbf{x}_\tau)$ , and  $L_{ch}(\mathbf{x}_\tau)$ ; we replace (5.2) by the index log-APPs after decoding one parameter. The term (5.2) also represents the extrinsic information EXT-SBSD of index-based ISCD. However, in the case of bit-based ISCD, additional  $2M$  calculations must be performed to obtain the bit-based EXT-SBSD in (2.22). The complexity of source decoder is summarized in Table 5.6. We notice that the bit-based ISCD requires one more decoding process to achieve the same performance as index-based ISCD. The evaluation of the system complexity needs to take account of the number of iterations required for the same parameter SNR.

	Branch metric	Forward/backward metrics	EXT-CD
General M	$0 \left( \frac{2^{\nu+M} \cdot L}{M} \right)$	$\frac{2^\nu \cdot L}{M} \left( \frac{2^\nu \cdot L}{M} \right)$	$\frac{2^M \cdot L}{M} \left( \frac{(2^M - 1) \cdot L}{M} \right)$
M=1	0 (2400)	1200 (1200)	600 (300)
M=3	0 (3200)	400 (400)	800 (700)

**Table 5.7: Summary of complexity for channel decoder with  $L = 300$ ,  $\nu = 2$ . The values outside (inside) the parentheses represent computation (memory)**

	Recursive term	EXT-SBSD
Bit-based	$\frac{2^M \cdot L}{M} (2^M)$	$2L (L)$
Index-based	$\frac{2^M \cdot L}{M} (2^M)$	$0 \left( \frac{(2^M - 1) \cdot L}{M} \right)$

**Table 5.8: Summary of complexity for source decoder with  $L = 300$ ,  $\nu = 2$ . The values outside (inside) the parentheses represent computation (memory)**

# Chapter 6

## Conclusions

In this thesis, we first introduced the bit-based ISCD system with BCJR channel decoding and modified SBSB algorithms under an AWGN channel. Then, we presented the detailed derivation of a novel index-based ISCD system with modified BCJR and SBSB algorithms. For real-time implementation, we also applied the concept of index-based L-value to simplify the ISCD computation. The parameter SNR performances of bit-based and index-based ISCDs were simulated and compared, together with an analysis of memory and computational complexity.

Simulation results show that the bit-based ISCD provides a remarkable improvement compared to conventional separate decoding. Further improvement is obtained by using the index-based ISCD, which exchanges extrinsic information of a source-controlled channel decoder and a softbit source decoder in the index-level. The index-based ISCD is more flexible than the bit-based ISCD, which requires bit-to-index conversion of the APPs after channel decoding. The index-based ISCD also shows a better performance than the bit-based version, and similar result occurs when we replace the SBSB by the interpolated-SBSB (see Appendix B). When applying the interpolated-SBSB, the ISCD shows further improvement compared to the original SBSB and therefore outperforms the SCCD utilizing the AK1 redundancy. This is due to the fact that the iterative decoding effectively takes the advantage of redundancy of both codecs and benefits from their extrinsic information exchange. However, the trade-offs between performance and complexity must be carefully evaluated as index-size  $M$  grows large.

Future research directions in the study of index-based ISCD are listed as follows:

- Examine the ISCD performance under a fading, memory, or more realistic

channel model.

- Examine the performance for different types of interleaver, including block and pseudo-random interleaver.
- The extrinsic information transfer (EXIT) chart [16] is a mathematical tool that is useful to evaluate the performance of turbo codes by separately analyzing each constituent codec. The EXIT chart has been adapted to the bit-based ISCD [17], which makes the maximum number of profitable iterations predictable and provides an index assignment optimization criterion in ISCD. However, the EXIT chart required for analyzing the index-based ISCD has yet been discovered. Further work is to develop an index-based EXIT chart and investigate its application to design an index-assignment-optimized ISCD for vector quantizers (VQ).



## Appendix A: Derivation of Formula (4.6)

Consider two real numbers  $a, b \in \mathbb{R}$ , we now show how to compute  $\log(e^a + e^b)$  without using  $e^a$  and  $e^b$  directly. We first assume  $a \geq b$ , then

$$\begin{aligned}\log(e^a + e^b) &= \log\left[e^a(1 + e^{b-a})\right] \\ &= \log e^a + \log(1 + e^{b-a}) \\ &= a + \log(1 + e^{b-a})\end{aligned}\tag{A.1}$$

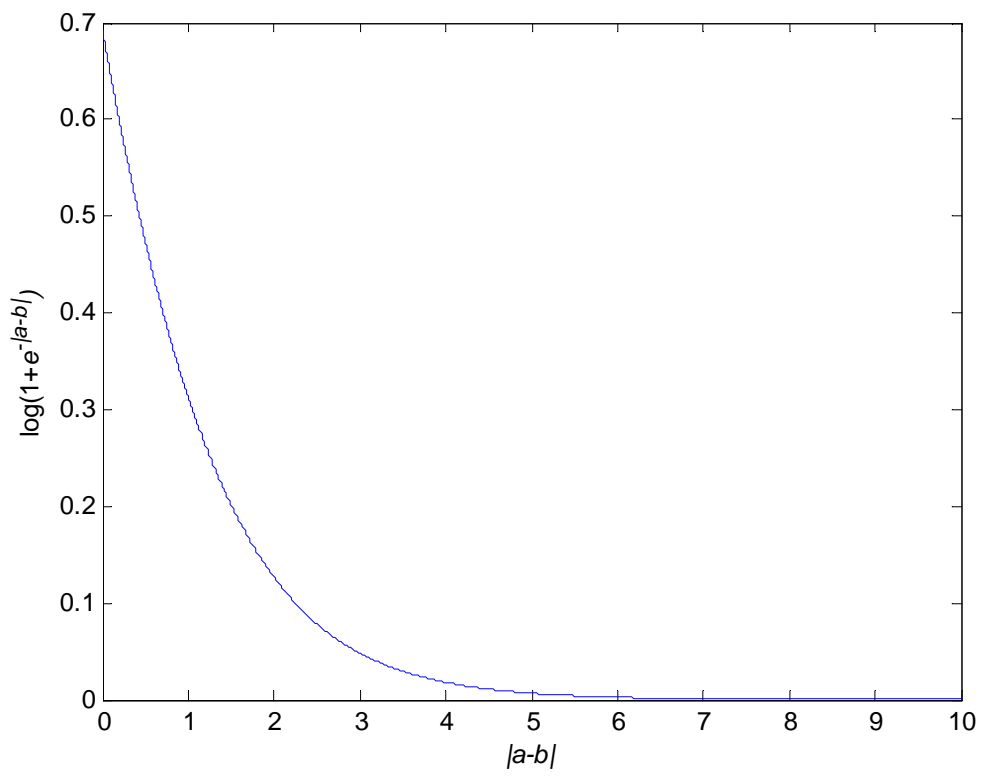
On the other hand, if  $a < b$ , we have

$$\begin{aligned}\log(e^a + e^b) &= \log\left[e^b(e^{a-b} + 1)\right] \\ &= \log e^b + \log(1 + e^{a-b}) \\ &= b + \log(1 + e^{a-b})\end{aligned}\tag{A.2}$$

By combining (A.1) and (A.2), we define a new function

$$\max^*(a, b) \triangleq \log(e^a + e^b) = \max(a, b) + \log(1 + e^{-|a-b|})\tag{A.3}$$

where  $\max()$  determines the larger value between  $a$  and  $b$ . Since we always have  $0 < e^{-|a-b|} \leq 1$ , the range of the logarithm term lies between 0 and  $\log 2$ , as shown in Figure A.1. A better way for its calculation is to build a look-up table. Proceeding in this way, we have removed the calculation of exponential terms  $e^a$  and  $e^b$ , which may cause the overflow problem.



**Figure A.1: Plot of  $\log(1+e^{-|a-b|})$  vs.  $|a-b|$**



## Appendix B: Interpolated-SBSD

In Chapter 2, we presented the modified SBSB and its extrinsic information for use in ISCD. The formula (2.18) is based on the principle of efficient decoding of source parameters and therefore only utilizes the correlation of previous decoded parameters. Since the turbo decoding is generally a block decoding algorithm, we can further utilize the correlation of the parameters in the whole decoding block. A new method called interpolated-SBSD (I-SBSD) was introduced in the bit-based ISCD [4] for data transmission over a memoryless channel. We now modified the I-SBSD for its use in index-based ISCD. We begin with the index-APP in (2.18)

$$\begin{aligned}
 P(\mathbf{u}_t = i | \tilde{\mathbf{U}}_1^T) &= P(\mathbf{u}_t = i | \tilde{\mathbf{U}}_1^t, \tilde{\mathbf{u}}_t, \tilde{\mathbf{U}}_{t+1}^T) \\
 &= C \cdot P(\tilde{\mathbf{U}}_{t+1}^T | \mathbf{u}_t = i, \tilde{\mathbf{U}}_1^t) \cdot P(\mathbf{u}_t = i, \tilde{\mathbf{U}}_1^{t-1}, \tilde{\mathbf{u}}_t) \\
 &\triangleq C \cdot \beta_t(\mathbf{u}_t = i) \cdot \alpha_t(\mathbf{u}_t = i)
 \end{aligned} \tag{B.1}$$

where

$$\begin{aligned}
 \alpha_t(\mathbf{u}_t = i) &= P(\mathbf{u}_t = i, \tilde{\mathbf{U}}_1^{t-1}, \tilde{\mathbf{u}}_t) \\
 \beta_t(\mathbf{u}_t = i) &= P(\tilde{\mathbf{U}}_{t+1}^T | \mathbf{u}_t = i, \tilde{\mathbf{U}}_1^t)
 \end{aligned} \tag{B.2}$$

represent the forward and backward metrics and can be recursively calculated as follows

$$\begin{aligned}
 \alpha_t(\mathbf{u}_t = i) &= p(\tilde{\mathbf{u}}_t | \mathbf{u}_t = i) \cdot \sum_{j=0}^{2^M-1} P(\mathbf{u}_t = i | \mathbf{u}_{t-1} = j) \cdot \alpha_{t-1}(\mathbf{u}_{t-1} = j) \\
 \beta_t(\mathbf{u}_t = i) &= \sum_{j=0}^{2^M-1} P(\mathbf{u}_{t+1} = j | \mathbf{u}_t = i) \cdot p(\tilde{\mathbf{u}}_{t+1} | \mathbf{u}_{t+1} = j) \cdot \beta_{t+1}(\mathbf{u}_{t+1} = j)
 \end{aligned} \tag{B.3}$$

With (B.3), the correlation of the whole decoding block can be utilized for estimating each parameter. The I-SBSD now acts like a BCJR decoding algorithm without parity bits. Following the discussion in section 2.3, we apply (2.19) and (2.20) to (B.3) to obtain the recursive formula with updated a priori information

$$\begin{aligned}
\alpha_t(\mathbf{u}_t = i) &= \prod_{\substack{m=1 \\ \{\mathbf{u}_t=i\}}}^M p(\tilde{u}_t(m)|u_t(m)) \cdot P_{CD}^{[extr]}(u_t(m)) \cdot \sum_{j=0}^{2^M-1} P(\mathbf{u}_t = i | \mathbf{u}_{t-1} = j) \cdot \alpha_{t-1}(\mathbf{u}_{t-1} = j) \\
\beta_t(\mathbf{u}_t = i) &= \sum_{j=0}^{2^M-1} P(\mathbf{u}_{t+1} = j | \mathbf{u}_t = i) \cdot \prod_{\substack{m=1 \\ \{\mathbf{u}_{t+1}=j\}}}^M P_{CD}^{[extr]}(u_{t+1}(m)) \cdot p(\tilde{u}_{t+1}(m)|u_{t+1}(m)) \cdot \beta_{t+1}(\mathbf{u}_{t+1} = j)
\end{aligned} \tag{B.4}$$

By substituting (B.4) to (B.1), the index-APP becomes

$$\begin{aligned}
P(\mathbf{u}_t = i | \tilde{\mathbf{U}}_1^T) &= C \cdot \beta_t(\mathbf{u}_t = i) \cdot \prod_{\substack{m=1 \\ \{\mathbf{u}_t=i\}}}^M p(\tilde{u}_t(m)|u_t(m)) \cdot P_{CD}^{[extr]}(u_t(m)) \cdot \\
&\quad \sum_{j=0}^{2^M-1} P(\mathbf{u}_t = i | \mathbf{u}_{t-1} = j) \cdot \alpha_{t-1}(\mathbf{u}_{t-1} = j)
\end{aligned} \tag{B.5}$$

Substituting (B.5) to (2.22) and utilizing (2.23), the bit-level APP for  $u_t(\lambda) = +1$  can be expressed in terms of bit-level extrinsic information of SBSBD and CD

$$\begin{aligned}
P(u_t(\lambda) = +1 | \tilde{\mathbf{U}}_1^T) &= \sum_{(\mathbf{u}, u_t(\lambda) = +1)} P(\mathbf{u} | \tilde{\mathbf{U}}_1^T) \\
&= C \cdot P(u_t(\lambda) = +1) \cdot p(\tilde{u}_t(\lambda) | u_t(\lambda) = +1) \cdot P_{CD}^{[extr]}(u_t(\lambda) = +1) \cdot \\
&\quad \cdot \left\{ \sum_{(\mathbf{u}, u_t(\lambda) = +1)} \beta_t(\mathbf{u}_t = i) \cdot \prod_{\substack{m=1 \\ m \neq \lambda}}^M p(\tilde{u}_t(m) | u_t(m)) \cdot P_{CD}^{[extr]}(u_t(m)) \cdot \sum_{j=0}^{2^M-1} P(\mathbf{u}_t^{[extr]}(\lambda) | \mathbf{u}_{t-1} = j) \cdot \alpha_{t-1}(\mathbf{u}_{t-1} = j) \right\} \\
&\triangleq C \cdot P(u_t(\lambda) = +1) \cdot p(\tilde{u}_t(\lambda) | u_t(\lambda) = +1) \cdot P_{CD}^{[extr]}(u_t(\lambda) = +1) \cdot P_{SBSBD}^{[extr]}(u_t(\lambda) = +1)
\end{aligned} \tag{B.6}$$

where the bit-based extrinsic information of I-SBBD can be expressed as

$$\begin{aligned}
P_{SBSBD}^{[extr]}(u_t(\lambda) = +1) &= \sum_{(\mathbf{u}, u_t(\lambda) = +1)} \beta_t(\mathbf{u}_t = i) \cdot \prod_{\substack{m=1 \\ m \neq \lambda}}^M p(\tilde{u}_t(m) | u_t(m)) \cdot P_{CD}^{[extr]}(u_t(m)) \cdot \\
&\quad \sum_{j=0}^{2^M-1} P(\mathbf{u}_t^{[extr]}(\lambda) | \mathbf{u}_{t-1} = j) \cdot \alpha_{t-1}(\mathbf{u}_{t-1} = j)
\end{aligned} \tag{B.7}$$

Similarly, following the discussion in section 3.3, we apply (3.21) to (B.3) to obtain the recursive formula with updated a priori information

$$\begin{aligned}
\alpha_t(\mathbf{u}_t = i) &= p(\tilde{\mathbf{u}}_t | \mathbf{u}_t = i) \cdot P_{CD}^{[extr]}(\mathbf{u}_t = i) \cdot \sum_{j=0}^{2^M-1} P(\mathbf{u}_t = i | \mathbf{u}_{t-1} = j) \cdot \alpha_{t-1}(\mathbf{u}_{t-1} = j) \\
\beta_t(\mathbf{u}_t = i) &= \sum_{j=0}^{2^M-1} P(\mathbf{u}_{t+1} = j | \mathbf{u}_t = i) \cdot P_{CD}^{[extr]}(\mathbf{u}_{t+1} = j) \cdot p(\tilde{\mathbf{u}}_{t+1} | \mathbf{u}_{t+1} = j) \cdot \beta_{t+1}(\mathbf{u}_{t+1} = j)
\end{aligned} \tag{B.8}$$

By substituting (B.8) to (B.1), the index-APP becomes

$$\begin{aligned}
P(\mathbf{u}_t = i | \tilde{\mathbf{U}}_1^T) &= C \cdot \beta_t(\mathbf{u}_t = i) \cdot p(\tilde{\mathbf{u}}_t | \mathbf{u}_t = i) \cdot P_{CD}^{[extr]}(\mathbf{u}_t = i) \cdot \sum_{j=0}^{2^M-1} P(\mathbf{u}_t = i | \mathbf{u}_{t-1} = j) \cdot \alpha_{t-1}(\mathbf{u}_{t-1} = j) \\
&= C \cdot p(\tilde{\mathbf{u}}_t | \mathbf{u}_t = i) \cdot P_{CD}^{[extr]}(\mathbf{u}_t = i) \cdot \beta_t(\mathbf{u}_t = i) \cdot \sum_{j=0}^{2^M-1} P(\mathbf{u}_t = i | \mathbf{u}_{t-1} = j) \cdot \alpha_{t-1}(\mathbf{u}_{t-1} = j) \\
&\triangleq C \cdot p(\tilde{\mathbf{u}}_t | \mathbf{u}_t = i) \cdot P_{CD}^{[extr]}(\mathbf{u}_t = i) \cdot P_{SBSBD}^{[extr]}(\mathbf{u}_t = i)
\end{aligned} \tag{B.9}$$

where the index-based extrinsic information of I-SBSD is expressed as

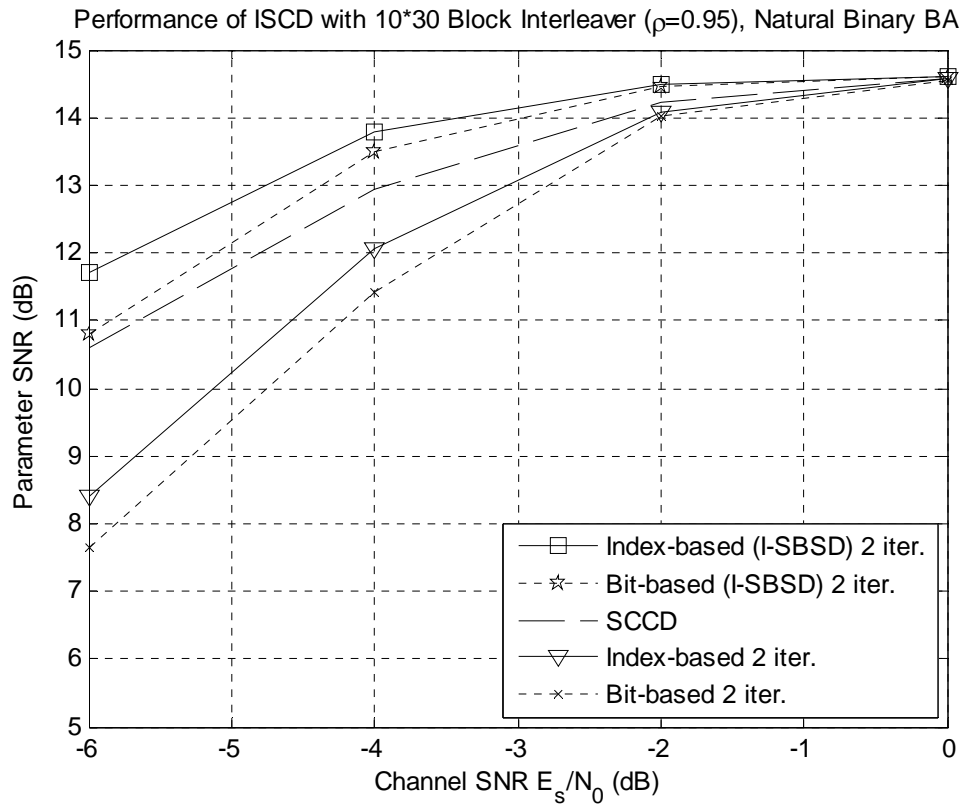
$$P_{SBSD}^{[ext]}(\mathbf{u}_t = i) = \beta_t(\mathbf{u}_t = i) \cdot \sum_{j=0}^{2^M-1} P(\mathbf{u}_t = i | \mathbf{u}_{t-1} = j) \cdot \alpha_{t-1}(\mathbf{u}_{t-1} = j) \quad (\text{B.10})$$

Simulations are carried out to examine its performance gain. The experimental settings and the channel decoder are the same as in Chapter 5. The parameter-estimated performances are summarized in Table B.1 and Figure B.1. In the table, the source-controlled channel decoding (SCCD) is the modified BCJR algorithm derived in [7], which combines the correlation of the parameters and the trellis structure of convolutional code in a single decoder. It is more complex than the memoryless modified BCJR algorithm derived in Chapter 3. The simulation result shows that the ISCD with I-SBSD significantly improves the performance when transmitting over a heavily noisy channel. The ISCD with I-SBSD systems outperforms the SCCD, both in bit-level and index-level.

Decoding Type	Channel SNR (dB)			
	-6	-4	-2	0
Bit-Based ISCD (2 iter.)	7.655	11.431	14.009	14.546
Index-Based ISCD (2 iter.)	8.409	12.075	14.074	14.571
SCCD	10.586	12.927	14.236	14.569
Bit-Based ISCD with I-SBSD (2 iter.)	10.809	13.485	14.451	14.611
Index-Based ISCD with I-SBSD (2 iter.)	11.706	13.785	14.492	14.61

**Table B.1: Performances of various decoders**





**Figure B.1: Performances of various decoders**



## Bibliography

- [1] C.E. Shannon, "A Mathematical Theory of Communications," *Bell Syst. Tech. J.*, vol. 27, pp. 379-423, pp. 623-656, 1948.
- [2] L. Xiaobei, "Joint Source-channel Decoding and its Application to MELP Encoded Speech," Ph.D. thesis, *Nanyang Technological University*, Singapore, 2004.
- [3] T. Fingscheidt and P. Vary, "Softbit Speech Decoding: A New Approach to Error Concealment," *IEEE Trans. Speech Audio Processing*, vol. 9, no. 3, pp. 240-251, 2001.
- [4] M. Adrat, "Iterative Source-Channel Decoding for Digital Mobile Communications," Ph.D. thesis, vol. 16 of ABDN, *Druck & Verlagshaus Mainz GMBH Aachen*, Aachen, Germany, 2003.
- [5] J. Hagenauer, "Source-Controlled Channel Decoding," *IEEE Trans. Commun.*, vol. 43, pp. 2449-2457, Sep. 1995.
- [6] L.R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate," *IEEE Trans. on Information Theory*, pp. 284-287, March 1974.
- [7] C. Lee and W. Chang, "MAP Symbol Decoding for Speech Recognition Over Wireless Networks," accepted by IEEE International Conference on Multimedia & Expo., Hannover, German, 2008.
- [8] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error Correcting Coding: Turbo Codes," in *Proc. IEEE International Conference on Communications (ICC)*, pp. 1064-1070, 1993.
- [9] C. Berrou and A. Glavieux, "Near Optimum Error Correcting Coding And Decoding: Turbo-Codes," *IEEE Trans. on Commun.*, vol. 44, no. 10, pp. 1261-1271, 1996.
- [10] R.G. Gallager, "Low-Density Parity-Check Codes," *IRE Trans. on Information*

*Theory*, pp. 21-28, January 1962.

- [11] J. Hagenauer, "Iterative Decoding of Binary Block and Convolutional Codes," *IEEE Trans. on Information Theory*, vol. 42, no. 2, March 1996.
- [12] M. Adrat, P. Vary, and J. Spittka, "Iterative Source-Channel Decoder Using Extrinsic Information from Softbit-Source Decoding," *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '01)*, vol. 4, pp. 2653-2656, Salt Lake City, Utah, USA, May 2001.
- [13] N. Görts, "A Generalized Framework for Iterative Source-Channel Decoding," *Annals of Telecommunications, Special Issue on Turbo Codes*, pp. 435-446, July/August 2001.
- [14] M. Bingeman, "Symbol-Based Turbo Codes for Wireless Communications," Master's thesis, *University of Waterloo*, Waterloo, Ontario, Canada, 2002.
- [15] S. Lin and D.J. Costello, "Error Control Coding," 2<sup>nd</sup> ed., *Pearson Prentice Hall*, ISBN 0-13-017973-6, 1983, 2004.
- [16] S.t. Brink, "Convergence Behavior of Iteratively Decoded Parallel Concatenated Codes," *IEEE Trans. on Communications*, vol. 49, no. 10, October 2001.
- [17] M. Adrat and P. Vary, "Iterative Source-Channel Decoding: Improved System Design Using EXIT Charts," *EURASIP Journal on Applied Signal Processing*, pp. 928-941, June 2005.