INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

Design and implementation of non-binary convolutional turbo codes

Yingzi Gao

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Applied Science at

Concordia University

Montréal, Québec, Canada

December 2001

ABSTRACT

Design and implementation of non-binary convolutional turbo codes

Yingzi Gao, MASc.

This thesis is about non-binary convolutional turbo codes - codes constructed via parallel concatenation of two circular recursive systematic convolutional (CRSC) encoders linked by an interleaver. The focus of the work is on the understanding and design of non-binary convolutional turbo codes. This includes investigation of central components that influence non-binary convolutional turbo code performances, such as the component encoders and the interleaver, as well as the procedure of iterative decoding. The investigations are carried out for transmission on additive white Gaussian noise channels.

First, this thesis presents the theoretical background of channel coding and turbo coding. Next, a general and efficient maximum *a posteriori* (MAP) soft-input soft-output (SISO) decoding algorithm is presented. And then, the simplified Max-Log-MAP algorithm is derived for the double-binary convolutional turbo code, which follows the specifications of turbo coding/decoding in the DVB-RCS standard (Digital Video Broadcasting standard for Return Channel via Satellite), for twelve different block sizes and seven coding rates. The quantizer of turbo-decoder is designed for the goal of implementation. The effect of quantiztion on the performance of the decoder is analyzed and simulated. The correction coefficient of the simplified Max-Log-MAP algorithm is also discussed.

The DVB-RCS standard turbo code uses quaternary alphabet and QPSK modulation. In order to increase the bandwidth efficiency, we present an extended non-binary turbo-coding scheme consisting of 8-ary triple-binary codes combined with 8PSK modulation. A comprehensive study over AWGN channel is carried out to show the good performance of the concatenated codes, the influence of various parameters and the sysmbol-by-sysmbol Max-Log-MAP algorithm.

Dedicated to my father and mother .....

ACKNOWLEDGEMENTS

It is with the utmost sincerity and appreciation to my supervisor Dr. M. R. Soleymani for giving me the opportunity to work with him. Without his continuous support and encouragement, I would not have been successful with my research. I would also like to express my gratitude for his confidence in me and my work by giving me a large scientific freedom and having the door of his office open in case of doubts and setbacks all the time. I admire his kindness very much for creating a fruitful working environment within the wireless and Satellite communications Laboratory.

I am deeply indebted to Yan Zhang and Usa Vilaipornsawai, Yuying Dai and Yan Mei, Qing Zhang and Jianfeng Weng, as well as Dr.Majid Barazande-Pour. With their help I go through all the periods of frustration and exhaustment. I also want to thank all the member of the laboratory staff and the system administrators for their great assistance.

This research has been supported by the NSERC under grant OGPIN001 and the Canadian Space Agency/Canadian Institute for Telecommunications Rearch/NSI Communications Inc. (CSA/CITR/NSI) under grant entitled "Spectrum Efficient Transmission with Turbo Codes for Satellite Communication Systems".

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS AND SYMBOLS

ADSL	Asymmetric Digital Subscriber Line
APRI-SOVA	A PRIori - Soft Output Viterbi Algorithm
ARQ	Automatic Repeat reQuest
AWGN	Additive White Gaussian Noise
BCH	Bose-Chaudhure-Hocquenghem
BER	Bit Error Rate
BCJR	Bahl-Cocke-Jelinek-Raviv
BSC	Binary Symmetric Channel
CCSDS	Consultative Commitee for Space Date System
CITR	Canadian Institute for Telecommunications Rearch
CRSC	circular recursive systematic convolutional
CSA	Canadian Space Agency
DAMA	Demand-Assigned Multiple Access
DSP	Digital Signal Processing
DVB-RCS	Digital Video Broadcasting standard for Return Channel via Satellite
ETSI	European Telecommunications Standards Institute
FEC	Forward Error Correction
FER	Frame Error Rate
FPGA	Field Programmable Gate Array
GF	Galois Field
HCCC	Hybrid Concatenated Convoultional Code
LLR	Log-Likilihood Ratio
MAP	Maximum A Posteriori
MF-TDMA	Multi-Frequency Time-Division Multiple Access

MPEG	Moving Picture Experts Group
M-PSK	M-ary Phase-Shift Keying
MSB	Most Significant Bit
PCCC	Parallel Concatenated Convolutional Code
QPSK	Quadrature Phase-Shift Keying
RCST	Return Channel Satellite Terminal
RM	Reed-Muller
RS	Reed-Solomon
RSC	Recursiv Systematic Convolutional
SCCC	Serial Concatenated Convolutional Code
SISO	Soft-Input Soft-Output
SOVA	Soft-Output Viterbi Algorithm
TCM	Trellis Coded Modulation
TCT	Time-solt Composition Table
TTCM	Turbo Trellis Coded Modulation
UMTS	Universal Mobile Telecommunication Service
VA	Viterbi algorithm

Chapter 1

Introduction

In modern era, communication technology is the most important technology due to its fast growth and revolutionary changes. It plays a great role in business, education and other aspects of our daily life. Telecommunication technology provides subscribers a wide range of services such as voice, data, e-mail, fax, internet, teleconferencing, image processing, video on demand, etc. A digital communication system is a means of transporting information from one party to another. The transmission in digital form allows for the use of a number of powerful signal processing techniques that would otherwise be unavailable, including, of course, error-control coding [1]. Error-control coding, having its roots in Shannon's Information Theory, has developed from a mathematical curiosity into a fundamental element of almost any system that transmits or stores digital information over the past fifty years. Many early coding applications were developed for deep-space and satellite communication systems. With the emergence of digital cellular telephony, digital television, and high-density digital storage, coding technology promises to predominate not only in scientific and military applications, but also in numerous commercial applications. In this thesis, the focus is a channel coding technique for satellite communication systems.

1.1 Composition of a Digital Communication System

In order to understand the role of error control coding, a block diagram of a basic communication system is shown in Figure 1.1 [2]. The *information source* generates message signals that are to be transmitted. The message can be either analog or digital, depending on the type of the source. The analog signal is sampled and quantized before it is transmitted through a digital system.

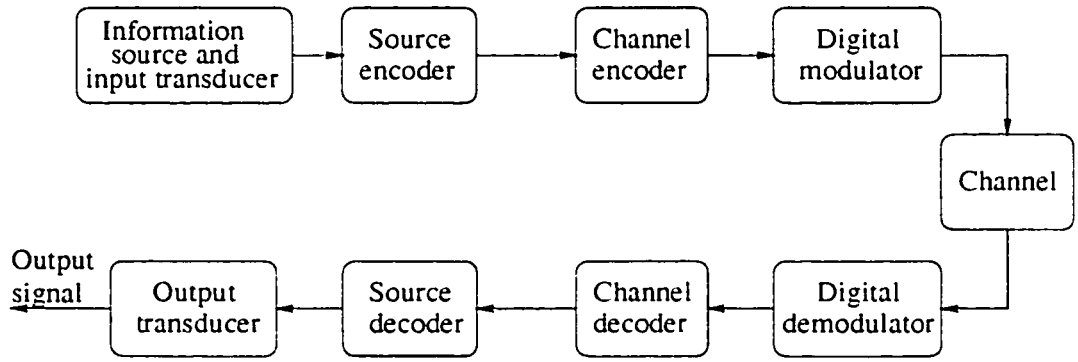


Figure 1.1: Basic elements of a digital communication system

The *source encoder* is designed to convert the source output sequence into a sequence of binary digits with minimum redundancy since the information source usually contains redundancy. For efficiency reasons, the information source is transformed by the source encoder into a sequence of bits called the *information sequence*. If the information source is digital, the source encoding does not require an analog to digital (A/D) conversion. However, the task to represent the message with as few bits as possible remains. This process is called *data compression*, a process during which the amount of redundancy present in the source messages is reduced or removed ideally.

After the source encoder, the *channel encoder* is purposely incorporated in the system to add redundancy into the information sequence. This redundancy is used to minimize transmission errors caused by channel impairment in the received

signal. Channel coding is a good way for achieving the necessary transmission fidelity with the available transmitter and receiver resources, such as power, bandwidth, and modulation technique. *Turbo codes* - the topic of this thesis - are this type of channel codes.

Since the output signal of the channel encoder is not normally suitable for transmission over a physical *channel*, the *digital modulator* maps the encoded digital sequences into a train of short analog waveforms suitable for propagation. The main goals of the modulation operation are to match the signal to the channel, enable simultaneous transmission of a number of signals over the same physical channel and to increase the speed of information transmission. This waveform is sent over physical channels, such as wireless channels, wire lines, fiber optic channels, satellite links, microwave radio links over free space, magnetic recording media, etc. Whatever the medium, the transmitted signal will be contaminated in a random manner by, e.g., thermal noise generated by electronic devices or cosmic noise picked up by the antenna. At the receiver end, the *digital demodulator* processes the corrupted waveform and produces an estimation of the transmitted data.

The output of the demodulator is passed to the *channel decoder* that makes estimates of the actually transmitted message. The decoder process is based on the encoding rule and the characteristics of the channel. The goal of the decoder is to minimize the effect of the channel noise. A measure of how well the demodulator and decoder perform is the frequency with which errors occur in the decoded sequence (*i.e.* The average probability of a bit-error at the decoder is a measure of the performance of the demodulation-decoder combination [2].).

Finally, the *source decoder* based on the source encoding rule transforms the sequence into an estimate of the source output sequence and delivers it to the user.

Among all the above functional blocks, the channel encoding and decoding pair is our concern. Different modulation schemes and many considerations of hardware implementation are also discussed in this thesis.

1.2 Development of Channel Coding Technique

The block diagram in Figure 1.1 presents a one-way system, where the transmission is strictly in the forward direction, from the transmitter to the receiver. In contrast to a two-way system that can use ARQ (Automatic Repeat Request) with error detection and retransmission, the error control strategy for a one-way system must be FEC (Forward Error Correction), which automatically corrects errors detected at the receiver. Most coded systems use some kind of FEC, even when the system is not strictly one-way [3]. FEC includes block codes, convolutional codes, as well as concatenated codes that built upon block and convolutional codes, and now, turbo codes are the new member of this group of FEC.

Block coding was the first coding technique developed. The encoder of an (n, k) block code accepts a message of k information symbols and maps them into a codeword of n symbols. Hamming codes are the first class of error correction linear block codes devised by Hamming [4]. These codes and their variations have been widely used for error control in digital communication and data storage systems. After that, the codes that are now called RM (Reed-Muller) codes were first described by Muller in 1954 and supplemented by Reed [5]. The resulting RM codes were an important step beyond the Hamming and Golay codes of 1949 and 1950 because of their flexibility in correcting varying numbers of errors per code word [1]. The discovery of BCH (Bose-Chaudhuri-Hocquenghem) codes and RS (Reed-Solomon) codes [6][7][8]made a great breakthrough in channel coding . Binary BCH codes include the Hamming and Golay codes. Among the non-binary BCH codes, the most important subclass is the class of RS codes that are optimal in the maximum separable distance sense. However, good RS codes require an alphabet size that grows with the block length. Forney broke the asymptotic difficulty by concatenating short random codes with long RS codes [9].

Reed first described a multiple-error-correcting decoding algorithm for RM codes based on sets of parity-check equations, which is called threshold decoding

[5]. The arithmetic Decoding algorithm and the algebraic Decoding algorithm are applied to binary Golay codes. Peterson's Direct-Solution Decoding algorithm and Berlekamp's algorithm are used for binary BCH codes; Peterson-Gorenstein-Zierler Decoding algorithm, Berlekamp-Massey algorithm and Euclid's algorithm are used for non-binary BCH and Reed-Solomon codes [1].

Convolutional codes were first introduced by Elias [10] in 1955 as an alternative to block code. Convolutional codes differ from block codes in that the encoder contains memory and the n encoder outputs at any given time unit depend not only on the k inputs at that time unit but also on m previous input blocks [3].

After Wozencraft first discovered the Sequential decoding [11] as a practical decoding algorithm for long randomly chosen convolutional codes, Massey [12] proposed a less efficient but simpler-to-implement decoding method called threshold decoding for both block and convolutional codes. An exceptional discovery was the Viterbi Algorithm (VA) [13], which works extremely well when the constraint length is small. Later, Omura [14] showed that Viterbi's algorithm was a solution to the problem of finding the minimal-weight path through a weighted, directed graph. In1973 and 1974, Forney showed that the Viterbi Algorithm provides both a maximum-likelihood and a maximum *a posteriori* (MAP) decoding algorithm for convolutional codes [15][16]. He went on to demonstrate that, in its most general form, the Viterbi algorithm is a solution to the problem of "MAP estimation of the state sequence of a finite-state discrete-time Markov process observed in memoryless noise" [15].

It has been found that with similar complexity, larger coding gains can be achieved by code concatenation. Forney first introduced concatenation in 1966 [9], where an inner code and an outer code were used in cascade. A common structure is a powerful non-binary RS outer code followed by a short constraint length inner convolutional code with soft-decision Viterbi decoding [17]. A symbol interleaver is used between the inner and outer code so that long bursty errors from the inner

decoder are broken into separate blocks for the outer decoder [18].

Turbo codes, were first presented to the coding community by Berrou *et al.* in 1993 [19], achieved a performance close to the Shannon limit [2] with the combination of two recursive convolutional codes, an interleaver, and a MAP iterative decoding algorithm. In this paper, they quoted a BER performance of 10^{-5} at an E_b/N_0 of 0.7dB using only a 1/2 rate code, generating tremendous interest in the field. The MAP algorithm, which is an algorithm for estimating random parameters with prior distributions, was first presented by Bahl, Cocke, Jelinik and Raviv in 1974 [20]. It is also called as the BCJR algorithm and is popular in the research community because of the introduction of turbo codes in recent years.

Prompted by turbo codes, which are also called parallel concatenated convolutional codes (PCCCs), serial concatenated convolutional codes (SCCCs) and hybrid concatenated convolutional codes (HCCCs) are constructed with the same components to provide similar coding gains [21]. SCCC and HCCC can perform better than PCCC at high signal-to-noise ratios, because of a superior distance profile. In addition to convolutional codes, block codes, such as Hamming codes, RM codes and RS codes, can also be used as the constituent code in the concatenation shceme.

Recently, the double-binary convolutional code was adopted in the DVB-RCS standard for its better performance than the classical concatenation of a convolutional code and a RS code. The transmission of data using various block sizes and coding rates make this coding scheme very flexible. Furthermore, two recent techniques in turbo coding are adopted in this coding scheme:

- Parallel concatenation of circular recursive systematic convolutional (CRSC) codes [22] makes convolutional turbo codes efficient for block coding,

- Double-binary elementary codes provide better error-correcting performance than binary codes for equivalent implementation complexity [23].

While the DVB-RCS standard has an excellent performance, its bandwidth efficiency is limited by the puncturing and QPSK modulation to less than 2bits/s/Hz.

In order to achieve bandwidth efficiencies above 2bits/s/Hz, one needs coding and modulation schemes with M-ary alphabet (M > 4). A new 8-ary triple-binary CRSC code is designed for 8PSK modulation in order to increase the bandwidth efficiency. This triple-binary code still has the features of the CRSC codes, which avoid the degradation of the spectral efficiency. Two circulation operation states have to be determined and the sequence has to be encoded four times. The different permutations (interleaving) can be achieved using generic equations with only a restricted number of parameters.

1.3 Outline of the thesis

This thesis outlines a program of study intended to bring the initial promise of turbo codes closer to practical fruition. The proposed thesis will address implementation issues for PCCC structure with non-binary convolutional turbo codes in the theoretical and algorithmic manner.

In Chapter 2, the fundamentals of turbo coding are explained. Starting with the principle of design of the binary convolutional codes and iterative decoding, the advantages of double-binary codes are introduced. Then, the motivation of the design of triple-binary codes is discussed.

In Chapter 3, the double-binary convolutional turbo code of DVB-RCS standard is illustrated. Simulation results show the performance of the standard coding scheme. In Chapter 4, turbo-decoder quantization is addressed and the simulation results show that the performance of an optimized 4-bit quantizer designed for double-binary convolutional turbo codes, is very close to the unquantized result, which uses a fixed-point model that corresponds to the given bit-width of the DSP.

In Chapter 5, the 8-ary triple-binary/8PSK codes are illustrated. Simulation results show the performance of the coding scheme chosen.

In Chapter 6, the conclusions and contributions are proposed.

Chapter 2

Turbo Coding

2.1 Introduction

The original turbo code [19] is the combination of two recursive systematic convolutional (RSC) codes, a pseudo-random interleaver, and an iterative MAP decoder. The turbo coding/decoding principle is illustrated in Figure 2.1. Π represents the interleaver between Encoder 1 and Encoder 2 and Π^{-1} represents deinterleaver between Decoder 2 and Decoder 1. Later, the interleaver and the PCCCs structure will be discussed in details. In this thesis, the PCCCs structure is considered in both binary and non-binary cases.

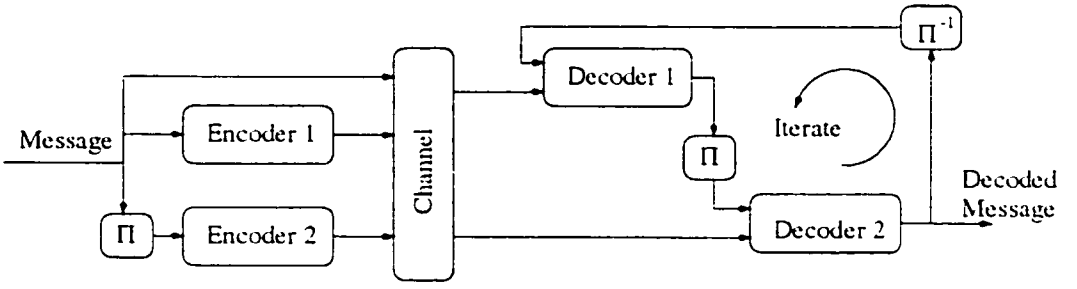


Figure 2.1: The turbo coding/decoding principle

According to the turbo coding principle, the turbo code design issues include

component code design, trellis termination method, interleaving strategy and implementation complexity based on the system design space for Turbo-codes [24]. Figure 2.2 depicts the system design space, which comprises a service-dependent and an implementation-dependent part [24]. The components of the Turbo-code encoder directly define the service-dependent part of the system design space: component codes, puncturer, interleaver and modulator. Though the required number of iterations is implementation-dependent, this number may also depend on the service to realize different qualities of service. In this thesis, we just consider the static iterations, for example, 8 iterations, which is enough to show the performance of the coding scheme chosen and implementation consideration.

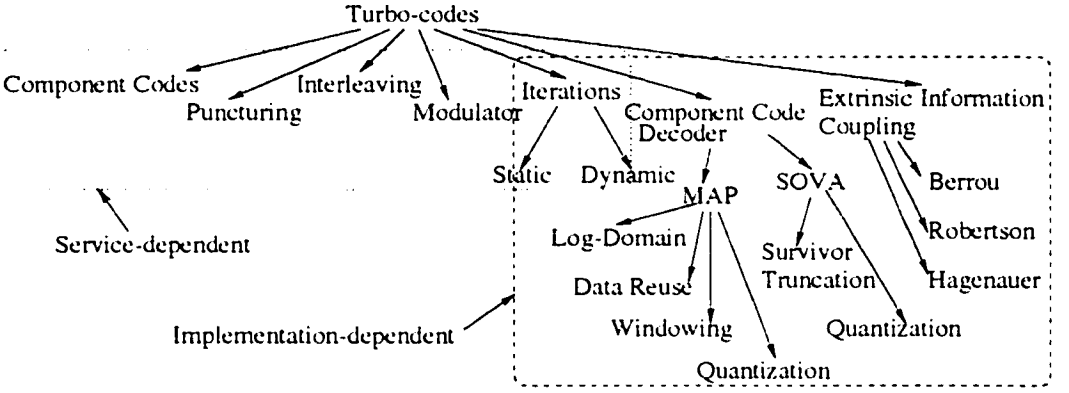


Figure 2.2: System design space

In order to reduce computational complexity, increase throughput, or reduce the power consumption, we just consider the sub-optimal Max-Log-MAP algorithm for non-binary convolutional turbo codes. Extrinsic information coupling (for the feedback) is performed according to Hagenauer [25]. The DVB-RCS standard turbo code uses a quaternary alphabet and QPSK modulation. For increasing bandwidth efficiency, we extend the double-binary turbo-coding scheme by designing 8-ary triple-binary codes to be combined with 8PSK modulation. Since Max-Log-MAP algorithm takes max-operation and omits the exponential part, the correct coefficient is discussed in chapter 4 as well as turbo-decoder quantization.

2.2 Design of binary convolutional turbo codes

Some of the approaches used in the past for the design of binary convolutional turbo codes are reviewed and discussed in this section. A general binary convolutional turbo encoder structure using two component encoders is illustrated in Figure 2.3. It consists of three basic building blocks: an interleaver, the component encoders, and a puncturing device with a multiplexing unit to compose the codeword. The interleaver is a device that re-orders the symbols in its input sequence.

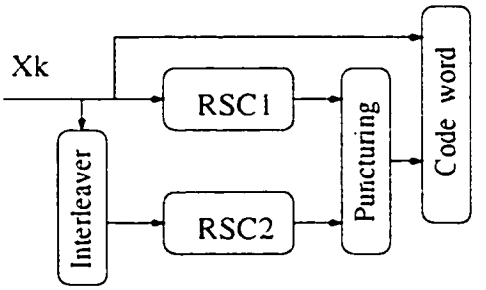


Figure 2.3: Encoder block diagram (Binary)

2.2.1 The Component Encoders

The Component encoders are recursive systematic convolutional (RSC) encoders, *i.e.*, systematic convolutional encoders with feedback. Such an encoder with two memory elements is depicted in Figure 2.4. For systematic codes, the information sequence is part of the codeword, which corresponds to the direct connection from the input to one of the outputs. For each input bit, the encoder generates two codeword bits: the systematic bit and the parity bit. Thus, the code rate is $1/2$ and the encoder input and output bits are denoted U_k and $(X_{k,1} = U_k, X_{k,2})$, respectively.

If the generator matrix of a non-recursive convolutional encoder with rate $1/n$ is

$$G(D) = (g_0(D)\ g_1(D) \ldots g_{n-1}(D)) \tag{2.1}$$

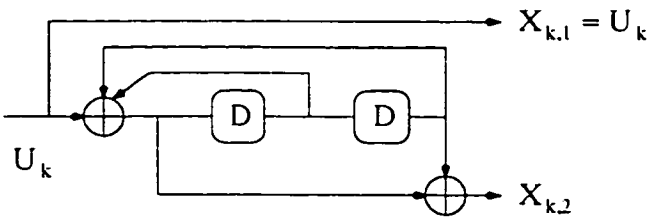


Figure 2.4: Recursive systematic convolutional encoder with feedback for rate 1/2 code with memory 2. The generator polynomials are $g_0(D) = 1 + D + D^2$ and $g_1(D) = 1 + D^2$.

the recursive encoder will be defined by,

$$G_{sys}(D) = (1 \ \frac{g_1(D)}{g_0(D)} \ldots \frac{g_{n-1}(D)}{g_0(D)}) \tag{2.2}$$

Since the performance of any binary code is dominated by its free distance d_{free} (the minimum Hamming distance between codewords, which coincides with the minimum Hamming weight of a nonzero codeword for linear codes) and its multiplicities [26], the optimal-recursive component encoders should maximize their effective free distance and minimize their multiplicities to achieve a good performance. Furthermore, to achieve a good performance, it is also important that the component codes are recursive.

In the design of convolutional codes, one advantage of systematic codes is that encoding is somewhat simpler than for non-systematic codes because less hardware is required, and another advantage is that no inverting circuit is needed for recovering the information sequence from the codeword[3]. On the other hand, a non-systematic code may be subject to catastrophic error propagation, which is called catastrophic code that a finite number of channel errors causes an infinite number of decoding errors. Systematic codes are always non-catastrophic.

2.2.2 Interleaving

Interleaving is a process of rearranging the ordering of an information sequence in a one-to-one deterministic format before the application of the second component

code in a turbo coding scheme. The inverse of this process is called deinterleaving which restores the received sequence to its original order. Interleaving is a practical technique to enhance the error correcting capability of coding systems [27]. It plays an important role in achieving good performance in turbo coding schemes.

Constructing a long block code from short memory convolutional codes by the interleaver results in the creation of codes with good distance properties, which can be efficiently decoded through iterative decoding [28]. The interleaver breaks low weight input sequences, and hence increases the code free Hamming distance or reduces the number of codewords with small distance in the code distance spectrum. On the other hand, the interleaver spreads out burst errors through providing "scrambled" information data to the second component encoder, and at the decoder, decorrelates the inputs to the two component decoders so that an iterative suboptimum decoding algorithm based on "uncorrelated" information exchange between the two component decoders can be applied. For example, after correction of some of the errors in the first component decoder, some of the remaining errors can be spread by the interleaver such that they become correctable in the other decoder. By increasing the number of iterations in the decoding process, the bit error probability approaches that of the maximum likelihood decoder. Typically, the performance of a turbo code is improved when the interleaver size is increased, which has a positive influence on both the code properties and iterative decoding performance.

A key component of turbo code is the interleaver whose design is essential for achieving high performance and is of interest to many turbo code researchers. Many interleaving strategies have been proposed, for example, Block interleavers including Odd-Even block interleavers and Block helical simile interleavers; Convolutional interleavers and Cyclic shift interleavers; Random interleavers including pseudo-random interleaver, Uniform and Non-uniform interleavers, S-random interleavers; Code matched interleavers, Relative prime interleavers; Golden interleavers, *etc.*[29] [30] [31] [32] [33] [34] [35] [36] [37] [38] [39] [40] [41].

2.2.3 Trellis Termination

As mentioned above, the performance of a code is highly dependent on its Hamming distance spectrum. For convolutional turbo codes, the Hamming distances between codewords are the result of taking different paths through the trellis. In principle, the larger the number of trellis transitions in which the two paths differ, the larger is the possible Hamming distance between the corresponding codewords. It is thus desirable that the shortest possible detour from a trellis path is as long as possible, to ensure a large Hamming distance between the two codewords that correspond to the two paths. However, in practice, convolutional turbo codes are truncated at some point in order to encode the information sequence block-by-block. If no precautions are taken before the truncation, each of the encoder states is a valid ending state and thus the shortest possible difference between the two trellis paths is made up of only one trellis transition. Naturally, this procedure may result in very poor distance properties, with accompanying poor error correcting performance. This question has been discussed in [42] [43] [44] [28] [45] [46] [47].

Since the component codes are recursive, it is not possible to terminate the trellis by transmitting ν zero tail bits. The tail bits are not always zero, which depend on the state of the component encoder after N information bits. Trellis termination force the encoder to the all-zero state at the end of each block to make sure that the initial state for the next block is the all-zero state. This way, the shortest possible trellis detour is the same as without truncation, and the distance spectrum is preserved.

Another approach to the problem of trellis truncation is tail-biting. With tail-biting, the encoder is initialized to the same state as it will end up in, by the end of the block. For feed-forward encoders tail-biting is readily obtained by inspection of the last ν bits in the input sequence, since these dictate the encoder ending state. The advantage of using tail-biting compared to trellis termination is that tail-biting does not require transmission of tail bits (the use of tail bits reduces the code rate

and increases the transmission bandwidth). For large blocks, the rate-reduction imposed by tail-bits is small, often negligible. For small blocks, however, it may be significant. References [48] [49] [50] [51] [52] addressed tail-biting.

2.2.4 Puncturing

Puncturing is the process of removing certain symbols/positions from the codeword, thereby reducing the codeword length and increasing the overall code rate. In the original Turbo code proposal, Berrou *et al.* punctured half the bits from each constituent encoders, except for the re-ordering caused by the interleaver. Thus, puncturing half the systematic bits from each constituent encoder corresponds to sending all the systematic bits once, if the puncturing is properly performed. The overall code rate is $R = 1/2$. Furthermore, puncturing may have different effect on different choices of interleavers, and on different constituent encoders.

When puncturing is considered, some output bits of v_0, v_1 and v_2 are deleted according to a chosen pattern defined by a puncturing matrix P. For instance, a rate 1/2 turbo code can be obtained by puncturing a rate 1/3 turbo code. The commonly used puncturing matrix is given by

$$\mathrm{P} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

where the puncturing period is 2. According to the puncturing matrix, the parity check digits from the two component encoders are alternately deleted. The punctured turbo code symbol at a given time consists of an information digit followed by a parity check digit which is alternately obtained from the first and the second component encoders. If the code rates of two component codes are denoted by R_1 and R_2, respectively, the overall turbo code rate R [27] will satisfy

$$\frac{1}{R} = \frac{1}{R_1} + \frac{1}{R_2} - 1 \tag{2.3}$$

2.3 Iterative Decoding

Iterative turbo decoding consists of two component decoders serially concatenated via an interleaver, identical to the one in the encoder. SISO algorithm (Soft input/Soft output algorithm) is very well suited for iterative decoding because it accepts *a priori* information at its input and produces *a posteriori* information at its output. In turbo decoding, trellis based decoding algorithms are recursive methods for estimation of the state sequence of a discrete-time finite-state Markov process observed in memoryless noise. With reference to decoding of noisy coded sequences, the MAP algorithm, which stands for "Maximum *a posteriori* Probability", is used to estimate the most likely information bit to have been transmitted in a coded sequence.

2.3.1 Tools for iterative decoding

2.3.1.1 Log-likelihood Algebra

The log-likelihood ratio of a binary random variable u_k, $L(u_k)$, is defined as

$$L(u_k) = \ln \frac{P(u_k = +1)}{P(u_k = -1)} \tag{2.3}$$

where u_k is in GF(2) with the elements {+1, -1} from time $k-1$ to time k, and +1 is the "null" element under the $\oplus$ addition.

Since

$$P(u_k = +1) = 1 - P(u_k = -1) \tag{2.4}$$

and

$$L(u_k) = \ln \frac{P(u_k = +1)}{1 - P(u_k = +1)} \tag{2.5}$$

then

$$e^{L(u_k)} = \frac{P(u_k = +1)}{1 - P(u_k = +1)} \tag{2.6}$$

Hence,

$$P(u_k = +1) = \frac{e^{L(u_k)}}{1 + e^{L(u_k)}} = \frac{1}{1 + e^{-L(u_k)}} = (\frac{e^{-L(u_k)/2}}{1 + e^{-L(u_k)}}) \cdot e^{L(u_k)/2} \tag{2.7}$$

$$P(u_k = -1) = 1 - \frac{1}{1 + e^{-L(u_k)}} = \frac{e^{-L(u_k)}}{1 + e^{-L(u_k)}} = (\frac{e^{-L(u_k)/2}}{1 + e^{-L(u_k)}}) \cdot e^{-L(u_k)/2} \tag{2.8}$$

can be represented as

$$P(u_k = \pm 1) = \frac{e^{\pm L(u_k)}}{1 + e^{\pm L(u_k)}} = (\frac{e^{-L(u_k)/2}}{1 + e^{-L(u_k)}}) \cdot e^{u_k \cdot L(u_k)/2} = A_k \cdot e^{u_k \cdot L(u_k)/2} \tag{2.9}$$

where $A_k = (\frac{e^{-L(u_k)/2}}{1+e^{-L(u_k)}})$ is the common item.

If the binary random variable u_k is conditioned on a different random variable or vector y_k, then we have a conditioned log-likelihood ratio $L(u_k|y_k)$ with

$$\begin{aligned} L(u_k|y_k) &= \ln \frac{P(u_k = +1|y_k)}{P(u_k = -1|y_k)} \\ &= \ln \frac{p(y_k|u_k = +1) \cdot P(u_k = +1)}{p(y_k|u_k = -1) \cdot P(u_k = -1)} \\ &= \ln \frac{p(y_k|u_k = +1)}{p(y_k|u_k = -1)} + \ln \frac{P(u_k = +1)}{P(u_k = -1)} \\ &= L(y_k|u_k) + L(u_k) \end{aligned} \tag{2.10}$$

2.3.1.2 Soft Channel Outputs

After transmission over a Gaussian fading channel,

$$\begin{aligned} L(u_k|y_k) &= \ln \frac{p(y_k|u_k = +1) \cdot P(u_k = +1)}{p(y_k|u_k = -1) \cdot P(u_k = -1)} \\ &= \ln \frac{exp(-\frac{E_s}{N_0}(y_k - a)^2)}{exp(-\frac{E_s}{N_0}(y_k + a)^2)} + \ln \frac{P(u_k = +1)}{P(u_k = -1)} \\ &= \ln\{exp[-\frac{E_s}{N_0}(y_k^2 - 2 \cdot a \cdot y_k + a^2 - y_k^2 - 2 \cdot a \cdot y_k - a^2)]\} + L(u_k) \\ &= 4 \cdot a \cdot \frac{E_s}{N_0} \cdot y_k + L(u_k) \\ &= L_c \cdot y_k + L(u_k) \end{aligned} \tag{2.11}$$

with $L_c = 4 \cdot a \cdot \frac{E_s}{N_0}$. For a fading channel, a denotes the fading amplitude whereas for a Gaussian channel, we set $a = 1$. For a Binary Symmetric Channel (BSC), we have the same relationship where L_c is the log-likelihood ratio of the crossover probabilities P_0, *i.e.*, $L_c = log((1 - P_0)/P_0)$. L_c is called the reliability value of the channel [25].

Since

$$P(y_k) = p(y_k|u_k = +1) \cdot P(u_k = +1) + p(y_k|u_k = -1) \cdot P(u_k = -1) \tag{2.12}$$

$$p(y_k|u_k = -1) = \frac{P(y_k)}{P(u_k = -1)} - \frac{p(y_k|u_k = +1) \cdot P(u_k = +1)}{P(u_k = -1)} \tag{2.13}$$

and

$$L(y_k|u_k) = \ln \frac{p(y_k|u_k = +1)}{\frac{P(y_k)}{P(u_k=-1)} - \frac{p(y_k|u_k=+1) \cdot P(u_k=+1)}{P(u_k=-1)}} = L_c \cdot y_k \tag{2.14}$$

then

$$e^{L_c \cdot y_k} = \frac{p(y_k|u_k = +1)}{\frac{P(y_k)}{P(u_k=-1)} - \frac{p(y_k|u_k=+1) \cdot P(u_k=+1)}{P(u_k=-1)}} \tag{2.15}$$

$$p(y_k|u_k = +1) = (\frac{P(y_k)}{P(u_k = -1)} - \frac{p(y_k|u_k = +1) \cdot P(u_k = +1)}{P(u_k = -1)}) \cdot e^{L_c \cdot y_k} \tag{2.16}$$

Hence

$$p(y_k|u_k = +1) = \frac{\frac{P(y_k)}{P(u_k=-1)} \cdot e^{L_c \cdot y_k}}{1 + \frac{P(u_k=+1)}{P(u_k=-1)} \cdot e^{L_c \cdot y_k}} \tag{2.17}$$

substitute $P(u = +1)$ and $P(u = -1)$ as in Equations (2.7) and (2.8):

$$\begin{aligned} p(y_k|u_k = +1) &= \frac{\frac{P(y_k) \cdot (1 + e^{L_c \cdot y_k})}{e^{-L(u_k)}} \cdot e^{L_c \cdot y_k}}{1 + e^{L(u_k)} \cdot e^{L_c \cdot y_k}} \\ &= \frac{\frac{P(y_k) \cdot (1 + e^{L_c \cdot y_k})}{e^{-L(u_k)}} \cdot e^{L_c \cdot y_k} \cdot e^{-(L(u_k) + L_c \cdot y_k)}}{e^{-(L(u_k) + L_c \cdot y_k)} + 1} \\ &= \frac{P(y_k) \cdot (1 + e^{-L(u_k)})}{1 + e^{-(L(u_k) + L_c \cdot y_k)}} \end{aligned} \tag{2.18}$$

similarly

$$p(y_k|u_k = +1) = \frac{P(y_k)}{P(u_k = +1)} - \frac{p(y_k|u_k = -1) \cdot P(u_k = -1)}{P(u_k = +1)} \tag{2.19}$$

$$\begin{aligned} L(y_k|u_k) &= \ln \frac{\frac{P(y_k)}{P(u_k=+1)} - \frac{p(y_k|u_k=-1)\cdot P(u_k=-1)}{P(u_k=+1)}}{p(y_k|u_k = -1)} \\ &= L_c \cdot y_k \end{aligned} \tag{2.20}$$

$$e^{-L_c \cdot y_k} = \frac{p(y_k|u_k = -1)}{\frac{P(y_k)}{P(u_k=+1)} - \frac{p(y_k|u_k=-1)\cdot P(u_k=-1)}{P(u_k=+1)}} \tag{2.21}$$

$$\begin{aligned} p(y_k|u_k = -1) &= \frac{\frac{P(y_k)}{P(u_k=+1)} \cdot e^{-L_c \cdot y_k}}{1 + \frac{P(u_k=-1)}{P(u_k=+1)} \cdot e^{-L_c \cdot y_k}} \\ &= \frac{P(y_k) \cdot (1 + e^{-L(u_k)}) \cdot e^{-L_c \cdot y_k}}{1 + e^{-(L(u_k)+L_c \cdot y_k)}} \end{aligned} \tag{2.22}$$

Hence

$$\begin{aligned} p(y_k|u_k = \pm 1) &= \left(\frac{P(y_k) \cdot (1 + e^{-L(u_k)}) \cdot e^{-L_c \cdot y_k/2}}{1 + e^{-(L(u_k)+L_c \cdot y_k)}}\right) \cdot e^{u_k \cdot L_c \cdot y_k/2} \\ &= B_k \cdot e^{u_k \cdot L_c \cdot y_k/2} \end{aligned} \tag{2.23}$$

where $B_k = \left(\frac{P(y_k)\cdot(1+e^{-L(u_k)})\cdot e^{-L_c \cdot y_k/2}}{1+e^{-(L(u_k)+L_c \cdot y_k)}}\right)$ is the common item.

2.3.1.3 Principle of the Iterative Decoding Algorithm

Assume that we have a "soft-in/soft-out" decoder available as shown in Figure 2.5 [25] for decoding the component codes.

The output of the "symbol-by-symbol" maximum *a posteriori* (MAP) decoder is defined as the *a posteriori* log-likelihood ratio for a transmitted "+1" and a transmitted "-1" in the information sequence

$$L(\hat{u}) = L(u|y) = ln \frac{P(u = +1|y)}{P(u = -1|y)} \tag{2.24}$$

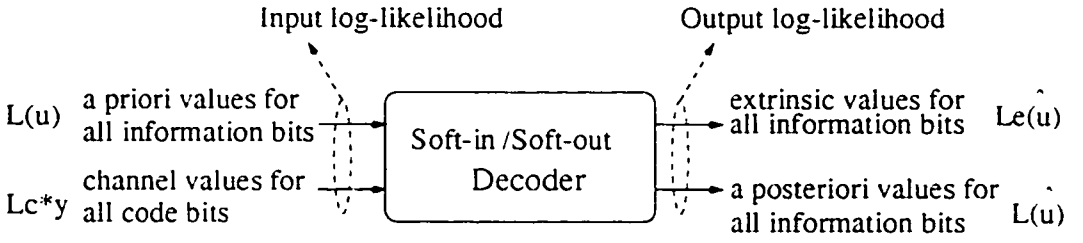


Figure 2.5: "Soft-in/Soft-out" decoder

Such a decoder uses *a priori* values $L(u)$ for all information bits u, if available, and the channel values $L_c \cdot y$ for all coded bits. It also delivers soft outputs $L(\hat{u})$ on all information bits and an extrinsic information $L_e(\hat{u})$ which contains the soft output information from all the other coded bits in the code sequence and is not influenced by the $L(u)$ and $L_c \cdot y$ values of the current bit. For systematic codes, the soft output for the information bit u will be represented in three additive terms

$$L(\hat{u}) = L_c \cdot y + L(u) + L_e(\hat{u}) \tag{2.25}$$

This means we have three independent estimates for the log-likelihood ratio of the information bits: the channel values $L_c \cdot y$, the *a priori* values $L(u)$ and the values $L_e(\hat{u})$ by a third independent estimator utilizing the code constraint. Assume equally likely information bits: then we do not have any *a priori* information available for the first iteration, thus we initialize $L(u) = 0$. The whole procedure is shown in figure 2.6.

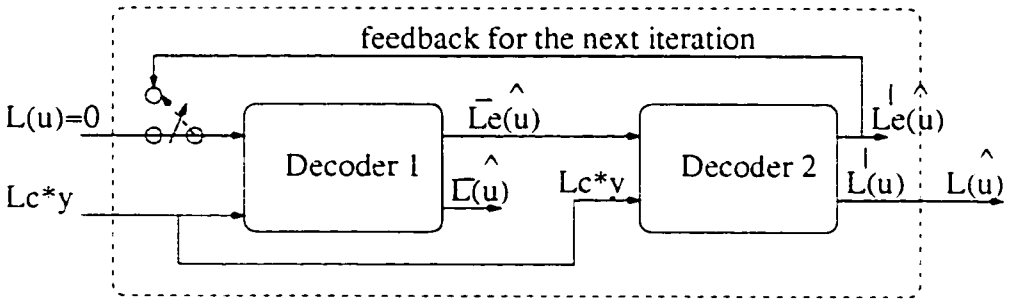


Figure 2.6: Iterative decoding procedure with two "soft-in/soft-out" decoders

2.3.2 Optimal and Suboptimal Algorithms

The Maximum Likelihood Algorithms (like the Viterbi Algorithm) find the most probable ***information sequence*** that was transmitted, while the maximum *a posteriori* (MAP) algorithm finds the most probable ***information bit*** to have been transmitted given the coded sequence. The information bits returned by the MAP algorithm need not form a connected path through the trellis.

For estimating the states or the outputs of a Markov process, the symbol-by-symbol maximum *a posteriori* (MAP) algorithm is optimal. Log-MAP algorithm avoids the approximations in the Max-Log-MAP algorithm and hence is equivalent to the true MAP but without its major disadvantages. MAP like algorithms, Soft-output viterbi algorithm (SOVA) and the Max-Log-MAP algorithm, are both suboptimal at low signal-to-noise ratios. The relationship between these algorithms is illustrated in Figure 2.7.

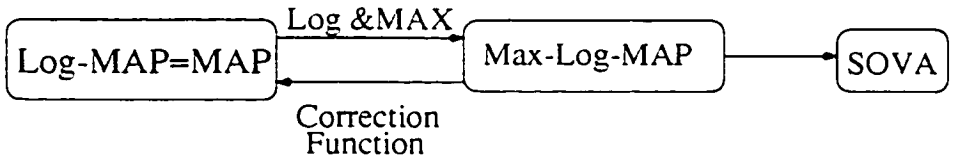


Figure 2.7: Relationship between MAP, Log-MAP, Max-Log-MAP and SOVA

2.3.2.1 Max-function

Define

$$E(x, y) = ln(e^x + e^y) \tag{2.26}$$

$$\begin{aligned} \ln(e^x + e^y) &= \ln e^x + \ln(e^x + e^y) - \ln e^x \\ &= x + \ln \frac{e^x + e^y}{e^x} \\ &= x + \ln(1 + e^{y-x}) \end{aligned} \tag{2.27}$$

Similar way

$$\begin{aligned}\ln(e^x + e^y) &= \ln e^y + \ln(e^x + e^y) - \ln e^y \\ &= y + \ln \frac{e^x + e^y}{e^y} \\ &= y + \ln(1 + e^{x-y}) \end{aligned} \tag{2.28}$$

Hence

$$\begin{aligned}E(x, y) &= \ln(e^x + e^y) \\ &= \max(x, y) + \ln(1 + e^{-|x-y|}) \end{aligned} \tag{2.29}$$

and

$$E(x, y) = \ln(e^x + e^y) \approx \max(x, y) \tag{2.30}$$

Similarly,

$$E(x, y, z, w) = \ln(e^x + e^y + e^z + e^w) \tag{2.31}$$

$$\begin{aligned}\ln(e^x + e^y + e^z + e^w) &= \ln e^x + \ln(e^x + e^y + e^z + e^w) - \ln e^x \\ &= x + \ln \frac{e^x + e^y + e^z + e^w}{e^x} \\ &= x + \ln(1 + e^{y-x} + e^{z-x} + e^{w-x}) \end{aligned} \tag{2.32}$$

or

$$\ln(e^x + e^y + e^z + e^w) = y + \ln(1 + e^{x-y} + e^{z-y} + e^{w-y}) \tag{2.33}$$

or

$$\ln(e^x + e^y + e^z + e^w) = z + \ln(1 + e^{x-z} + e^{y-z} + e^{w-z}) \tag{2.34}$$

or

$$\ln(e^x + e^y + e^z + e^w) = w + \ln(1 + e^{x-w} + e^{y-w} + e^{z-w}) \tag{2.35}$$

Hence

$$\begin{aligned} E(x,y,z,w) &= \max(x,y,z,w) \\ &\quad + \ln(e^{x-\max(x,y,z,w)} + e^{y-\max(x,y,z,w)} + e^{z-\max(x,y,z,w)} + e^{w-\max(x,y,z,w)}) \\ &\approx \max(x,y,z,w) \end{aligned} \tag{2.36}$$

Generally, using the same way, define

$$\begin{aligned} E_{i=1}^{k} x_i &= \ln \sum_{i=1}^{k} e^{x_i} \\ &= \ln(e^{x_1} + e^{x_2} + \cdots + e^{x_{k-2}} + e^{x_{k-1}} + e^{x_k}) \\ &= \ln(e^{x_1} + e^{x_2} + \cdots + e^{x_{k-2}} + e^{\ln(e^{x_{k-1}} + e^{x_k})}) \\ &= \ln(e^{x_1} + e^{x_2} + \cdots + e^{x_{k-2}} + e^{E(x_{k-1}, x_k)}) \\ &= \ln(e^{x_1} + e^{x_2} + \cdots + e^{\ln(e^{x_{k-2}} + e^{E(x_{k-1}, x_k)})}) \\ &= \ln(e^{x_1} + e^{x_2} + \cdots + e^{E(x_{k-2}, E(x_{k-1}, x_k))}) \\ &= E(x_1, E(x_2, \cdots, E(x_{k-2}, E(x_{k-1}, x_k)))) \end{aligned} \tag{2.37}$$

Hence

$$E_{i=1}^{k} x_i = \ln \sum_{i=1}^{k} e^{x_i} = \max(x_i) + \ln \sum_{i=1}^{k} e^{x_i - \max(x_i)} \tag{2.38}$$

2.3.2.2 MAP algorithm

The trellis of a binary feedback convolutional encoder has the structure shown in Figure 2.8.

From above, define the log-likelihood ratio as:

$$L(\hat{u}_k) = \ln \frac{P(u_k = +1|y)}{P(u_k = -1|y)} = \ln \frac{\sum_{u_k=+1}^{(s',s)} P(s', s, y)}{\sum_{u_k=-1}^{(s',s)} P(s', s, y)} \tag{2.39}$$

where

$$\begin{aligned} P(s', s, y) &= P(s', y_{j<k}) \cdot P(s, y_k|s') \cdot P(y_{j>k}|s) \\ &= \underbrace{P(s', y_{j<k})} \cdot \underbrace{P(s|s') \cdot P(y_k|s', s)} \cdot \underbrace{P(y_{j>k}|s)} \\ &= \alpha_{k-1}(s') \cdot \gamma_k(s', s) \cdot \beta_k(s) \end{aligned} \tag{2.40}$$

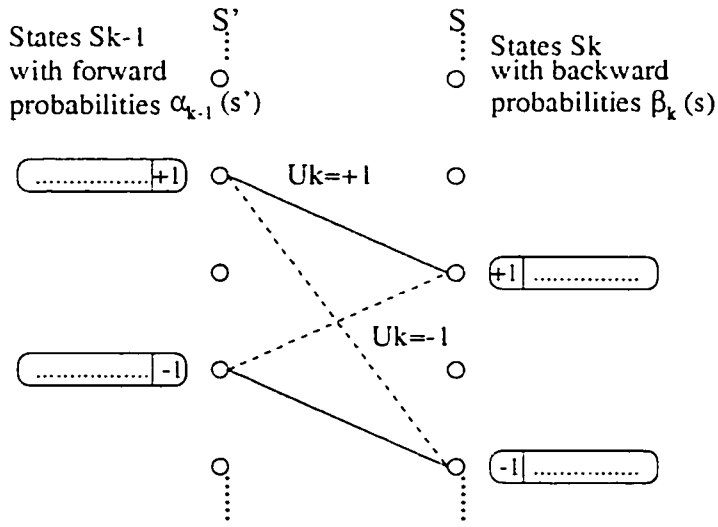


Figure 2.8: Trellis structure of systematic convolutional codes with feedback encoders

Here $y_{j<k}$ denotes the sequence of received symbols y_j from the beginning of the trellis up to time $k-1$ and $y_{j>k}$ is the corresponding sequence from time $k+1$ up to the end of the trellis. The forward recursion and backward recursion of the MAP algorithm yield

$$\alpha_k(s) = \sum_{(s',s)} \gamma_k(s',s) \cdot \alpha_{k-1}(s') \tag{2.41}$$

$$\beta_{k-1}(s') = \sum_{(s',s)} \gamma_k(s',s) \cdot \beta_k(s) \tag{2.42}$$

$$L(\hat{u}_k) = \ln \frac{\sum_{u_k=+1}^{(s',s)} \gamma_k(s',s) \cdot \alpha_{k-1}(s') \cdot \beta_k(s)}{\sum_{u_k=-1}^{(s',s)} \gamma_k(s',s) \cdot \alpha_{k-1}(s') \cdot \beta_k(s)} \tag{2.43}$$

The branch transition probability:

$$\begin{aligned} \gamma_k(s',s) &= P(s|s') \cdot p(y_k|s',s) \\ &= P(y_k|u_k) \cdot P(u_k) \end{aligned} \tag{2.44}$$

The index pair s' and s determines the information bit u_k and the coded bits $x_{k,v}$, for $v = 2, ..., n$.

where

$$\begin{aligned} P(y_k|u_k) &= P(y_{k,1}|u_k) \cdot (\prod_{v=2}^{n} P(y_{k,v}|u_k, s', s)) \\ &= P(y_{k,1}|u_k) \cdot (\prod_{v=2}^{n} P(y_{k,v}|x_{k,v}) \end{aligned} \tag{2.45}$$

is the independent joint probabilities of the received symbols and

$$P(u_k) = P(u_k = \pm 1) = A_k e^{u_k L(u_k)/2} \tag{2.46}$$

From above equation (2.23),

$$\begin{aligned} P(y_k|u_k) &= P(y_{k,1}|u_k) \cdot (\prod_{v=2}^{n} P(y_{k,v}|x_{k,v}) \\ &= B_k \cdot exp(\frac{1}{2} L_c \cdot y_{k,1} \cdot u_k) \cdot (\prod_{v=2}^{n} exp(\frac{1}{2} L_c \cdot y_{k,v} \cdot x_{k,v}) \\ &= B_k \cdot exp(\frac{1}{2} L_c \cdot y_{k,1} \cdot u_k + \frac{1}{2} \sum_{v=2}^{n} L_c \cdot y_{k,v} \cdot x_{k,v}) \end{aligned} \tag{2.47}$$

Hence,

$$\begin{aligned} \gamma_k(s', s) &= P(y_k|u_k) \cdot P(u_k) \\ &= B_k \cdot exp(\frac{1}{2} L_c \cdot y_{k,1} \cdot u_k + \frac{1}{2} \sum_{v=2}^{n} L_c \cdot y_{k,v} \cdot x_{k,v}) \cdot A_k \cdot exp(\frac{1}{2} u_k \cdot L(u_k)) \\ &= A_k \cdot B_k \cdot exp(\frac{1}{2} L_c \cdot y_{k,1} \cdot u_k + \frac{1}{2} \sum_{v=2}^{n} L_c \cdot y_{k,v} \cdot x_{k,v} + \frac{1}{2} u_k \cdot L(u_k)) \end{aligned} \tag{2.48}$$

The term A_k and B_k in (2.48) are equal for all transitions from level $k-1$ to level k and hence will cancel out in the ratio of (2.43). Therefore, the branch transition operation to be used in (2.41) and (2.42), is reduced to the expression

$$exp(\frac{1}{2} u_k (L_c \cdot y_{k,1} + L(u_k))) \cdot \gamma_k^{(e)}(s', s) \tag{2.49}$$

with

$$\gamma_k^{(e)} = exp(\frac{1}{2} \sum_{v=2}^{n} L_c \cdot y_{k,v} \cdot x_{k,v}) \tag{2.50}$$

Thus, the log-likelihood ratio becomes

$$L(\hat{u}_k) = L_c \cdot y_{k,1} + L(u_k) + \ln \frac{\sum_{u_k=+1}^{(s',s)} \gamma_k^{(e)}(s',s) \cdot \alpha_{k-1}(s') \cdot \beta_k(s)}{\sum_{u_k=-1}^{(s',s)} \gamma_k^{(e)}(s',s) \cdot \alpha_{k-1}(s') \cdot \beta_k(s)} \tag{2.51}$$

As discussed in Equation (2.25), for any random bit in the information sequence:

$$L(\hat{u}_k) = L_c \cdot y_{k,1} + L(u_k) + L_e(\hat{u}_k) \tag{2.52}$$

the extrinsic information can be calculated as:

$$L_e(\hat{u}_k) = L(\hat{u}_k) - L_c \cdot y_{k,1} - L(u_k) \tag{2.53}$$

or

$$L_e(\hat{u}_k) = \ln \frac{\sum_{u_k=+1}^{(s',s)} \gamma_k^{(e)}(s',s) \cdot \alpha_{k-1}(s') \cdot \beta_k(s)}{\sum_{u_k=-1}^{(s',s)} \gamma_k^{(e)}(s',s) \cdot \alpha_{k-1}(s') \cdot \beta_k(s)} \tag{2.54}$$

2.3.2.3 Log-MAP algorithm

The Log-MAP algorithm is a transformation of MAP, which has equivalent performance without its problems in practical implementation. It works in the logarithmic domain and multiplication is converted to addition. The followings are the calculations of branch transition probability and forward/backward recursion:

$$\gamma_k^{LM}(s',s) = \ln \gamma_k(s',s) = \frac{1}{2} L_c \cdot y_{k,1} \cdot u_k + \frac{1}{2} \sum_{v=2}^{n} L_c \cdot y_{k,v} \cdot x_{k,v} + \frac{1}{2} u_k \cdot L(u_k) \tag{2.55}$$

$$\alpha_k^{LM}(s) = \ln \alpha_k(s) = \ln(\sum_{s'} e^{\gamma_k^{LM}(s',s)} \cdot e^{\alpha_{k-1}^{LM}(s)}) = \ln(\sum_{s'} e^{\gamma_k^{LM}(s',s) + \alpha_{k-1}^{LM}(s)}) \tag{2.56}$$

$$\beta_{k-1}^{LM}(s') = \ln \beta_{k-1}(s') = \ln(\sum_{s} e^{\gamma_k^{LM}(s',s)} \cdot e^{\beta_k^{LM}(s')}) = \ln(\sum_{s} e^{\gamma_k^{LM}(s',s) + \beta_k^{LM}(s')}) \tag{2.57}$$

Therefore, the log-likelihood ratio is given by

$$
\begin{aligned}
L(\hat{u}_k) &= \ln \frac{\sum_{u_k=+1}^{(s',s)} e^{\gamma_k^{LM}(s',s)} \cdot e^{\alpha_{k-1}^{LM}(s)} \cdot e^{\beta_k^{LM}(s')}}{\sum u_{k=-1}^{(s',s)} e^{\gamma_k^{LM}(s',s)} \cdot e^{\alpha_{k-1}^{LM}(s)} \cdot e^{\beta_k^{LM}(s')}} \\
&= \ln \frac{\sum_{u_k=+1}^{(s',s)} e^{\gamma_k^{LM}(s',s)+\alpha_{k-1}^{LM}(s)+\beta_k^{LM}(s')}}{\sum_{u_k=-1}^{(s',s)} e^{\gamma_k^{LM}(s',s)+\alpha_{k-1}^{LM}(s)+\beta_k^{LM}(s')}} \\
&= \ln\Big(\sum_{u_k=+1}^{(s',s)} e^{\gamma_k^{LM}(s',s)+\alpha_{k-1}^{LM}(s)+\beta_k^{LM}(s')}\Big) \\
&\quad - \ln\Big(\sum_{u_k=-1}^{(s',s)} e^{\gamma_k^{LM}(s',s)+\alpha_{k-1}^{LM}(s)+\beta_k^{LM}(s')}\Big)
\end{aligned} \tag{2.58}
$$

2.3.2.4 Max-Log-MAP algorithm

With max(·) function, the Log-MAP algorithm becomes Max-Log-MAP algorithm resulting the degradation in the performance, but the complexity is reduced.

$$
\alpha_k^{MLM}(s) = \max([\gamma_{u_k=+1}^{LM}(s',s) + \alpha_{k-1}^{LM}(s')], [\gamma_{u_k=-1}^{LM}(s',s) + \alpha_{k-1}^{LM}(s')]) \tag{2.59}
$$

$$
\beta_{k-1}^{MLM}(s') = \max([\gamma_{u_k=+1}^{LM}(s',s) + \beta_k^{LM}(s)], [\gamma_{u_k=-1}^{LM}(s',s) + \beta_k^{LM}(s)]) \tag{2.60}
$$

$$
\begin{aligned}
L(\hat{u}_k) &= \max_{u_k=+1}^{(s',s)}([\gamma_k^{LM}(s',s) + \alpha_{k-1}^{MLM}(s) + \beta_k^{MLM}(s')], \\
&\quad [\gamma_k^{LM}(s',s) + \alpha_k^{MLM}(s) + \beta_k^{MLM}(s')]) \\
&\quad - \max_{u_k=-1}^{(s',s)}([\gamma_k^{LM}(s',s) + \alpha_{k-1}^{MLM}(s) + \beta_k^{MLM}(s')], \\
&\quad [\gamma_k^{LM}(s',s) + \alpha_k^{MLM}(s) + \beta_k^{MLM}(s')])
\end{aligned} \tag{2.61}
$$

2.4 Design of double-binary convolutional turbo codes

For efficient convolutional turbo coding, the number of memory is a key consideration since the component codes with small constraint lengths ensure convergence

at very low signal to noise ratios and the correlation effects are minimized [53]. Moreover, reasonable constraint lengths make hardware implementation on a single integrated circuit possible since the material complexity of the decoder grows exponentially with the code memory. The solution chosen uses memory $v = 3$ component codes. Double-binary elementary codes provide better error-correcting performance than binary codes for equivalent implementation complexity [23]. And also, parallel concatenation of circular recursive systematic convolutional codes (CRSC) [22] makes convolutional turbo codes efficient for coding of data cells in blocks. The encoder structure of double-binary component codes is depicted in Figure 2.9.

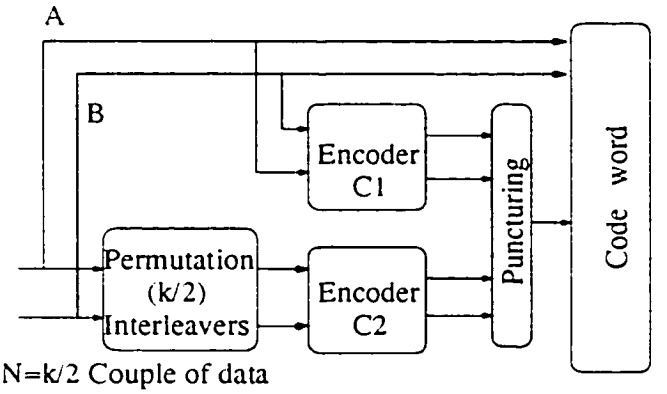


Figure 2.9: Encoder block diagram (Double-binary convolutional turbo code)

2.4.1 Circular Recursive Systematic Convolutional (CRSC) codes

For block-oriented encoding, the Convolutional turbo codes have to be truncated at some point, which result in degradation in performance without precaution. It is easy to know the initial state as encoder is generally forced into the "all zero" state at the beginning of encoding. However, the decoder has no special information available regarding the final state of the trellis. There are many approaches to this problem, for example, forcing the encoder state at the end of the encoding phase for one or all component encoders. Tail bits are used to "close" the trellises and are then

sent to the decoder. This method presents two major drawbacks. First, minimum weight w_{min} is no longer equal to the original w_{min} for all information data[1], since, at the end of each block, the second "1" bringing the encoder back to the "all zero" state may be a part of the tail bits. In this case, turbo decoding is handicapped if tail bits are not encoded another time. The second problem is that the spectral efficiency of the transmission is degraded and the degradation is more for shorter blocks.

With circular convolutional codes, the encoder retrieves the initial state at the end of the encoding operation. The decoding trellis can therefore be seen as a circle and decoding may be initialized everywhere on this circle. This technique, well known for non recursive codes (the so-called "tail-biting"), has been adapted to the specificity of the recursive codes [22]. Adopting circular coding avoids the degradation of the spectral efficiency of the transmission when forcing the value of the encoder state at the end of the encoding stage by the addition of tail bits [56].

2.4.2 Circular states (tail-biting) principle

Circular coding ensures that, at the end of the encoding operation, the encoder retrieves the initial state, so that data encoding may be represented by a circular trellis. The existence of such a state, called the circular state S_c, is ensured when the size of the encoded data block, N, is not a multiple of the period of the encoding recursive generator. The value of the circulation state depends on the contents of the sequence to encode and determining S_c requires a pre-encoding operation: first, the encoder is initialized in the "all zero" state. The data sequence is encoded once, leading to a final state S_N^0. Next, find S_c through the final state S_N^0 explained as follow.

[1]The weight w of a binary word is defined as the number of information bits equal to '1', that is the number of infromation bits differing from the "all zero" word, which is used as a reference for linear codes. For a recursive codes, used in DVB-RCS standard, when the final states are fixed by the encoder, the minimum value for w is 2. For more details see [54] [55], for example.

In practice, for example, in DVB standard [57], the relation between S_c and S_N^0 is provided by a small combinational operator with $v = 3$ input and output bits. To perform a complete encoding operation of the data sequence, two circulation states have to be determined, one for each component encoder, and the sequence has to be encoded four times instead of twice.

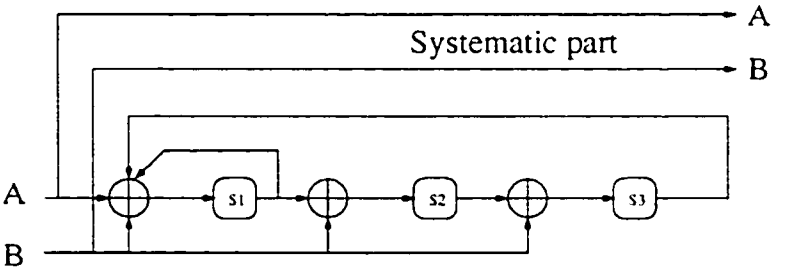


Figure 2.10: Recursive convolutional (double binary) encoder with memory $v = 3$. The output, which is not relevant to the operation of the register, has been omitted

Let us consider a recursive convolutional encoder, for instance, the encoder is depicted in Figure 2.10 [22]. At time k, register state S_k is a function of previous state S_{k-1} and input vector X_k. Let G be the generator matrix of the considered code. State S_k and S_{k-1} are linked by the following recursion relation:

$$S_k = G \cdot S_{k-1} + X_k \tag{2.62}$$

For Figure 2.10 encoder, vector S_k and X_k , and matrix G are given by:

$$S_k = \begin{bmatrix} S_{1,k} \\ S_{2,k} \\ S_{3,k} \end{bmatrix}; \qquad X_k = \begin{bmatrix} A_k + B_k \\ B_k \\ B_k \end{bmatrix}; \qquad G = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

From (2.62), we can infer:

$$S_k = G \cdot S_{k-1} + X_k \tag{2.63}$$

$$S_{k-1} = G \cdot S_{k-2} + X_{k-1} \tag{2.64}$$

$$\vdots \quad \vdots \quad \vdots$$

$$S_1 = G \cdot S_0 + X_1 \tag{2.65}$$

Hence, S_k may be expressed as a function of the initial state S_0 and of data feeding the encoder between times 1 and k:

$$S_1 = G^k \cdot S_0 + \sum_{p=1}^{k} G^{k-p} \cdot X_p \tag{2.66}$$

If N is the input sequence length, it is possible to find a circular state S_c such that $S_c = S_N = S_0$. Its value is derived from (2.66):

$$S_c = \left\langle I + G^N \right\rangle^{-1} \sum_{p=1}^{N} G^{N-p} \cdot X_p \tag{2.67}$$

where I is the identity matrix.

State S_c depends on the sequence of data and exists only if $I + G^N$ is invertible (Note that some G matrices are not suitable). In particular, N cannot be a multiple of the period L of the encoding recursive generator, defined as:

$$G^L = I \tag{2.68}$$

If the encoder starts from state S_c, it comes back to the same state when the encoding of the N data symbols (in Figure 2.10 encoder) is completed. Such an encoding process is called circular because the associated trellis may be viewed as a circle, without any discontinuity on transitions between states.

Determining S_c requires a pre-encoding operation. First, the encoder is initialized in the "all zero" state. Then, the data sequence of length N is encoded once, leading to final state S_N^0. Thus, from (2.66):

$$S_N^0 = \sum_{p=1}^{N} G^{N-p} \cdot X_p \tag{2.69}$$

Combining this result with (2.67), the value of circulation state S_c can be linked to S_N^0 as follows:

$$S_c = \left\langle I + G^N \right\rangle^{-1} S_N^0 \tag{2.70}$$

In a second operation, data are definitely encoded starting from state S_c. S_c value is then calculated from expression 2.70. The disadvantage of this method lies in having to encode the sequence twice: once from the "all zero" state and the second time from the state S_c. Nevertheless, in most cases, the double encoding operation can be performed at a frequency much higher than the data rate, so as to reduce the latency effects.

2.4.3 Two-level permutations (interleaving)

The performance of parallel concatenation of convolutional codes (PCCC) at low error rates, is essentially governed by the permutation that links the component codes. The simplest way to achieve interleaving in a block is to adopt uniform or regular interleaving: data are written row-wise and read column-wise in a rectangular matrix. This kind of permutation behaves very well towards error patterns with weight 2 or 3, but is very sensitive to square or rectangular error patterns, as explained in [54]. Classically, in order to increase distances given by rectangular error patterns, non-uniformity is introduced in the permutation relations. Many proposals have been made in this direction, especially for the UMTS application. The CCSDS turbo code standard may also be cited as an example of non-uniform permutation. However, the disorder that is introduced with non-uniformity can affect the scattering properties concerning weight 2 or 3 error patterns [56].

With double-binary codes, non-uniformity can be introduced without any repercussion on the good scattering properties of regular interleaving [53]. The principle involves introducing local disorder into the data couples (two bits), for example (A, B) becoming (B, A) - or (B, A+B), or others - periodically before the second encoding. This helps to avoid many error patterns that would appear without applying it. Therefore, this appears to be a significant gain in the search for large minimum distance.

2.4.4 Iterative decoding principle for circular recursive codes

Circular codes are well suited to the turbo decoding concept. In fact, the circular code principle may be applied in two slightly different ways, according to whether the code is self-concatenated or not.

Case 1: The code is self-concatenated, that is the second encoding step directly follows on from the first step without intermediate reinitializing of the register state. Circulation state S_c is calculated for the whole sequence of length $2N$. At reception, the decoder performs a decoding of the second sequence length N.

Case 2: The code is not self-concatenated, that is the encoder is initialized at the beginning of each encoding stage. Two circulation states S_{c_1} and S_{c_2} corresponding to both encoded sequences, are calculated. At reception, the two sequences of length N are decoded separately.

Depending on the case, data encoding is represented by the second circular trellises. Whatever elementary algorithm is used, iterative decoding requires repeated turns around the circular trellis(es), the extrinsic information table being continuously updated during data processing. Iterations naturally follow one after the other without any discontinuity between transitions from state to state.

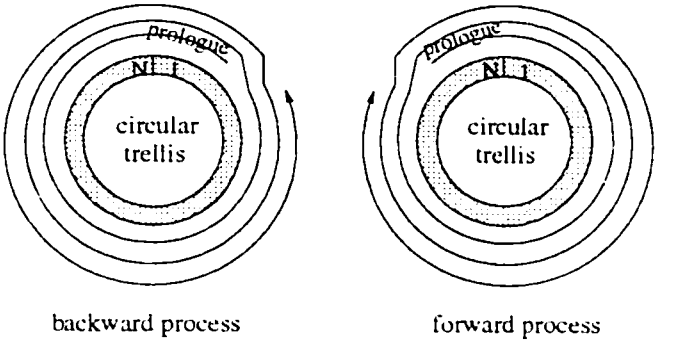


Figure 2.11: Processing a circular code by the backward-forward algorithm

In the case where the MAP (maximum *a posteriori*) algorithm or the simplified algorithm, Max-Log-MAP algorithm, is applied, decoding the sequence consists of

going round the circular trellis anti-clockwise for the backward process, and clockwise for the forward process (Figure 2.11)[22], during which data is decoded and extrinsic information is built. For both processes, probabilities computed at the end of a turn are used as the initial values for the next turn. The number of turns performed around the circular trellis is equal to the number of iterations required by the iterative process. In practice, the iterative process is preceded by a "prologue" decoding step, performed on a part of the circle for a few v. This is intended to guide the process towards an initial state which is a good estimate of the circulation state.

2.5 Design of triple-binary codes for 8PSK modulation

In [23], the author addressed that using double-binary codes as component codes represents a simple means to reduce the correlation effects since they have a direct incidence on the erroneous paths in the trellises, which leads to a lowered path error density and reduces correlation effects in the decoding process. This leads to a better performance so that an 8-state double-binary turbo codes perform better than 16-state binary turbo codes. On the other hand, the influence of puncturing and simplified version of MAP algorithm are less significant in the case of double-binary codes [56]. Moreover, from an implementation point of view, the bit rate at the decoder output is twice that of a binary decoder processing the same number of iterations, with the same circuit clock frequency and with an equivalent complexity per decoded bit. Thus, given the data block size, the latency of the decoder is divided by 2 compared with the binary case because the size of the permutation matrix is halved [56].

Even though double-binary convolutional turbo codes have so many advantages that they are adopted in DVB standard, they have the disadvantage of being

suitable only for combination with quadrature QPSK modulation, and also the iterative decoding is handicapped by the symbol values of the channel output according to 8PSK symbol mapping. The symbol-by-symbol MAP algorithm and the puncturing mapping for double-binary codes do not work for double-binary codes combined with 8PSK mapping.

Motivated by the above considerations, the triple-binary codes are designed for combining with 8PSK modulation. The encoder structure is also PCCC. The component codes are still CRSC codes which avoid the degradation of the spectral efficiency of the transmission when forcing the value of the encoder state at the end of the encoding stage by the addition of tail bits. Determining the circular state S_c follows the principle that was discussed in section 2.4.2, however, the memory $v = 4$ is chose in order to get good performance. The encoder is fed by blocks of k bits or N triplets ($k = 3 * N$ bits). N is a multiple of 8. See Figure 2.12.

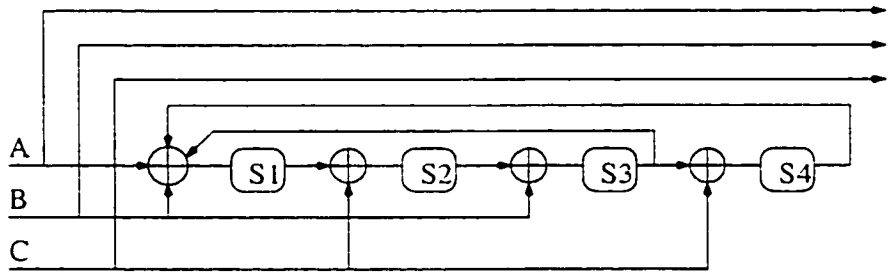


Figure 2.12: Circular recursive systematic convolutional (CRSC) with memory $v = 4$. The output, which is not relevant to the operation of the register, has been omitted

For the restriction of the 8PSK symbol mapping, the non-uniformity has not be introduced in the interleaver, *i.e.*, we do not have the first level interleaving, for example, (A, B, C) becoming (B, C, A) - or (B, C+A, B+A), or others. However, the permutation parameters still set as P_0 , P_1 , P_2 and P_3 with fomular $i = (P_0 * j + P + 1) mod. N$.

The decoding principle is the same as that for double-binary codes. The bit-by-bit simplified Max-Log-MAP algorithm applied to double-binary codes is modified to

symbol-by-symbol. Higher decoder speeds resulting from the high rate convolutional codes used in the code construction. The details are discussed in Chapter 5.

2.6 Summary

The fundamental principles behind binary convolutional turbo coding have been introduced, including the component codes, interleaving, trellis termination, puncturing and the principles of iterative decoding. The central components of a turbo code encoder are the recursive systematic convolutional (RSC) encoders and the interleaver that link them in parallel by re-ordering the bits in the information sequence before they enter the second constituent encoder. Both optimal algorithm–MAP algorithm and Log-MAP algorithm, suboptimal algorithm–Max-Log-MAP algorithm and SOVA depend on the tools, Log-likelihood Algebra, Soft channel output and the principle of iterative decoding algorithm.

Circular recursive systematic convolutional (CRSC) component codes, non-uniform permutation and different puncturing map make double-binary turbo codes efficient for block coding and provide better error-correcting performance than binary codes for equivalent implementation complexity.

Motivated by the advantages of double-binary convolutional turbo codes, 8-ary triple-binary codes are designed to adapt 8PSK modulation according to the principle of the design of double-binary codes.

Chapter 3

DVB-RCS standard and double-binary convolutional turbo codes

Since convolutional turbo codes are very flexible codes, easily adaptable to a large range of data block sizes and coding rates, they have been adopted in the DVB standard for Return Channel via Satellite (DVB-RCS). In this chapter, we follow the specifications of turbo coding/decoding in that standard, for twelve block sizes and seven coding rates, and the simulation results, in particular for the transmission of ATM cells in AWGN channel, show the performance of the coding scheme chosen. Moreover, the iterative decoding procedure and simplified iterative decoding algorithm for double-binary convolutional turbo code is presented.

3.1 DVB-RCS system considerations

The DVB Committee approved DVB-RCS standard for Return Channel via Satellite [56], EN 301 790. This is also the ETSI standard to provide two-way, full-IP, asymmetric communications via satellite in order to supplement the coverage of ADSL

(Asymmetric Digital Subscriber Line) and Cable modem. Once a single "hub" infrastructure is in place, satellite offers the service deployed quickly all over a large area, especially the service quality and the cost per subscriber is independent of the distance between the terminal and the access point. This standard specifies an air interface allowing a large number of small terminals to send "return" signals to a central gateway and at the same time receive IP data from that hub on the "forward" link in the usual DVB/MPEG2 (Digital Video Broadcasting/Moving Picture Expert Group-2) broadcast format, which places satellite in a favorable position. Figure 3.1 shows the reference model of the satellite interactive network and the details refer to the DVB standard[57].

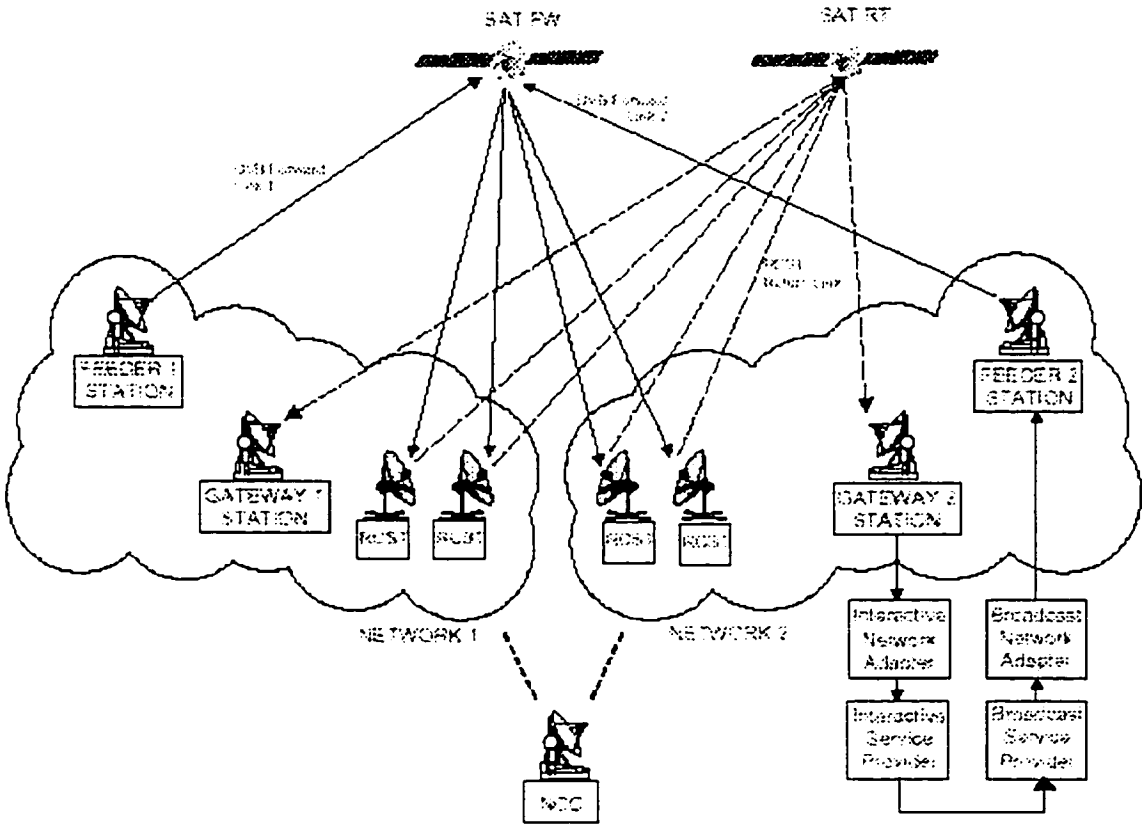


Figure 3.1: Refererce Model for the Satellite Interactive Network

In the DVB-RCS standard, the satellite resource on the return link, whose

speed (terminals-to-hub) can range from 144 Kbps to 2 Mbps, is shared among the terminals transmitting small packets and using MFTDMA (Multi-Frequency Time Division Multiple Access) / DAMA (Demand-Assigned Multiple Access) techniques. Once the session is established with the hub, the link is permanently open because this access is packet-based and on-demand, thus providing an "always-on" IP connection, with efficient use of satellite resources (provided the duty-cycle of the terminals is low on the return link).

The new standard includes such advanced features as a software-radio system (the terminal acquires the characteristics of the transmission parameters, it has to use from signals broadcast by the hub), network synchronization in the digital domain (all terminals are synchronized to the same precise clock transmitted digitally by the hub), sophisticated bandwidth-on-demand protocols that can mix constant-rate, dynamic-rate, volume-based and best-effort bandwidth allocation and advanced forward-error-correction turbo coding (as will be used in the new On-Board satellite processing system, Skyplex [58]). This thesis presents the rationale behind the selection of this FEC and the performance achieved on a software implementation of the decoder. The quantization of the input data at the decoder is discussed in Chapter 4.

3.2 DVB-RCS coding requirements

Since DVB-RCS applications involve the transmission of data using various block sizes and coding rates, the coding scheme has to be very flexible, with better performance than the classical concatenation of a convolutional code and a Reed-Soloman code. On the other hand, it has to be able to process data so as to allow the transmission of data bit rates up to 2Mbps. as indicated in Section 3.1.

As mentioned in Chapter 2, the use of double-binary circular recursive systematic convolutional (CRSC) component codes makes turbo codes very efficient

for block coding. The different permutations (interleavers) can be achieved using generic equations with only a restricted number of parameters. Moreover, a simple puncturing device is sufficient to adapt coding rate and the same decoding hardware can be used to manage every block size/coding rate combination. Therefore, the double-binary convolutional turbo codes that was proposed for DVB-RCS applications is powerful, very flexible and can be implemented with reasonable complexity.

3.3 System Model

Figure 3.2 shows the System Model of the double-binary convolutional turbo code. Coding for channel error protection is applied to traffic and control data, which are transmitted in the types of bursts. In this thesis, the AWGN channel, QPSK modulation and demodulation are used as in the DVB-RCS standard. The complete system model in detail is depicted in Figure 3.12 on page 58.

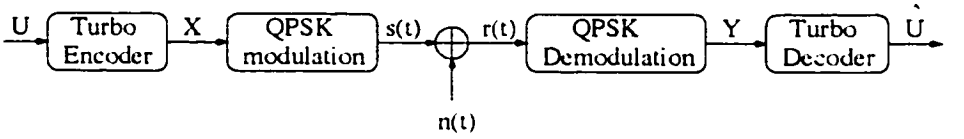


Figure 3.2: System Model of DVB-RCS standard

3.3.1 Encoder structure

The encoder structure is depicted in figure 3.3. The data sequence to be encoded, made up of k information bits, feeds the CRSC encoder twice: first, in the natural order of the data (switch in position 1), and next in an interleaved order, given by time permutation function Π (switch in position 2).

The encoder is fed by blocks of k bits or N couples ($k = 2 * N\ bits$). N is a multiple of 4 (k is a multiple of 8). It uses a double-binary Circular Recursive Systematic Convolutional (CRSC) code (see Figure 3.4). The MSB (Most Significant

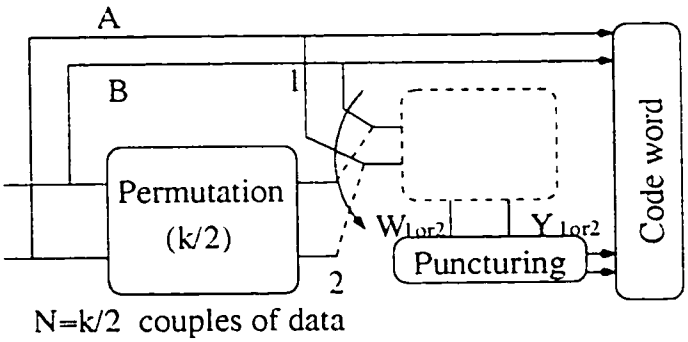


Figure 3.3: Encoder block diagram (Double-binary convolutional turbo code)

Bit) bit of the first byte after the burst preamble is assigned to A, the next bit to B and so on for the remainder of the burst content.

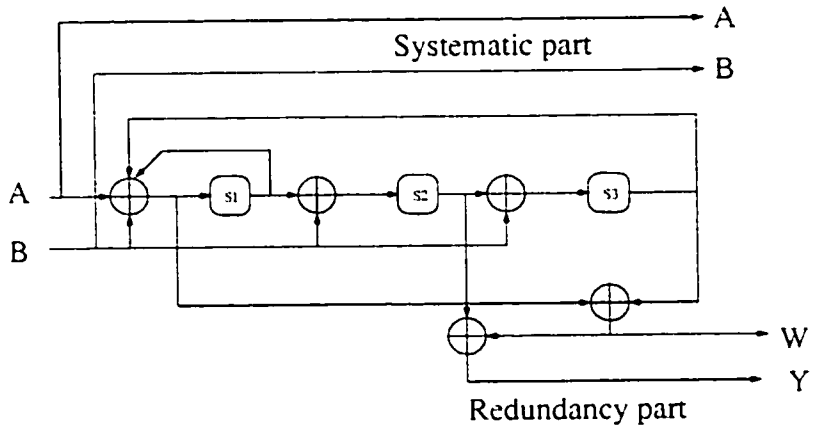


Figure 3.4: Double-binary Circular Recursive Systematic Convolutional encoder

The polynomials defining the connections are described in octal and symbolic notations as follows:

- for the feedback branch: 15(in octal), equivalently $1 + D + D^3$ (in symbolic notation);

- for the Y parity bits: 13, equivalently $1 + D^2 + D^3$;

- for the W parity bits: 11, equivalently $1 + D^3$.

The input A is connected to tap "1" of the shift register and the input B is connected to the taps "1", D and D^2. The state of the encoder is denoted $S(0 \leq S \leq 7)$ with $S = 4 \cdot s_1 + 2 \cdot s_2 + s_3$. Since the value of the circulation state depends on the

contents of the sequence to encode, determining the circulation state S_c requires a pre-encoding operation.

First, the encoder is initialized in the "all zero" state and fed by the sequence in the natural order with incremental address $i = 0, \ldots, N-1$. The data sequence is encoded once, leading to a final state S_N^0. S_c value is then calculated from expression $S_c = \left\langle I + G^N \right\rangle^{-1} \cdot S_N^0$. According to the length N of the sequence, use the following correspondence to find S_c. (See Table 3.1) And then, the encoder is fed by the same

$S_N^0 \rightarrow$ $N\ mod.7$	0	1	2	3	4	5	6	7
1	$S_c = 0$	$S_c = 6$	$S_c = 4$	$S_c = 2$	$S_c = 7$	$S_c = 1$	$S_c = 3$	$S_c = 5$
2	$S_c = 0$	$S_c = 3$	$S_c = 7$	$S_c = 4$	$S_c = 5$	$S_c = 6$	$S_c = 2$	$S_c = 1$
3	$S_c = 0$	$S_c = 5$	$S_c = 3$	$S_c = 6$	$S_c = 2$	$S_c = 7$	$S_c = 1$	$S_c = 4$
4	$S_c = 0$	$S_c = 4$	$S_c = 1$	$S_c = 5$	$S_c = 6$	$S_c = 2$	$S_c = 7$	$S_c = 3$
5	$S_c = 0$	$S_c = 2$	$S_c = 5$	$S_c = 7$	$S_c = 1$	$S_c = 3$	$S_c = 4$	$S_c = 6$
6	$S_c = 0$	$S_c = 7$	$S_c = 6$	$S_c = 1$	$S_c = 3$	$S_c = 4$	$S_c = 5$	$S_c = 2$

Table 3.1: Circulation state correspondence table

sequence in the natural order with the circulation state S_{c_1}. This first encoding is called C_1 encoding.

Secondly, the encoder (after initialization) is fed by the interleaved sequence with incremental address $j = 0, \ldots, N-1$ with the circulation state S_{c_2} after pre-encoding which is the same operation process as in C_1 encoding. This second encoding is called C_2 encoding. The permutation function $i = \prod(j)$ that gives the natural address i of the considered couple, when reading it at place j for the second encoding, is given in subsection 3.3.2.

Therefore, to perform a complete encoding operation of the data sequence, two circulation states have to be determined, one for each component encoder, and the sequence has to be encoded four times instead of twice. This is not a real problem, as the encoding operation can be performed at a frequency much higher than the data rate.

Figure 3.5 shows the trellis diagram of the above double-binary convolutional turbo encoder.

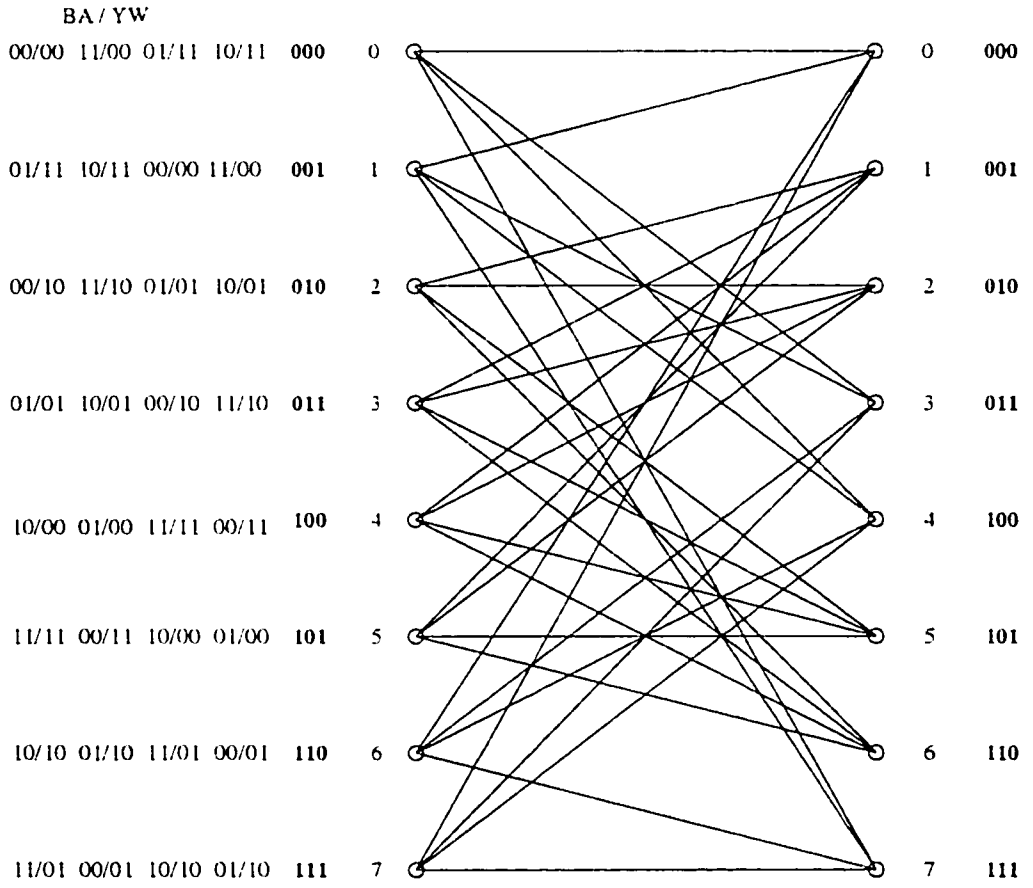


Figure 3.5: Trellis diagram of CRSC turbo code

There are 8 states in the trellis and the numbers shown on the left of each state represent the input to the encoder and their corresponding trellis outputs. The numbers from left to right correspond to state transitions from top to bottom exiting from each state.

3.3.2 Description of permutation

Let N be the number of data couples in each block at the encoder input (each block contains $2N$ data bits). The permutation is done on two levels, which is

non-uniform interleaving, the first one inside the couples (**level 1**), the second one between couples (**level 2**):

Set the permutation parameters P_0, P_1, P_2 and P_3. Table 3.2 provides the default parameters to be used for different block lengths. Those parameters can be updated by the Time-slot Composition Table (TCT) (See DVB-RCS standard sub-clause 8.5.5.4 [57]).

Frame size in couples	P_0	$\{P_1, P_2, P_3\}$
$N = 48$ (12 *bytes*)	11	$\{24, 0, 24\}$
$N = 64$ (16 *bytes*)	7	$\{34, 32, 2\}$
$N = 212$ (53 *bytes*)	13	$\{106, 108, 2\}$
$N = 220$ (55 *bytes*)	23	$\{112, 4, 116\}$
$N = 228$ (57 *bytes*)	17	$\{116, 72, 188\}$
$N = 424$ (106 *bytes*)	11	$\{6, 8, 2\}$
$N = 432$ (108 *bytes*)	13	$\{0, 4, 8\}$
$N = 440$ (110 *bytes*)	13	$\{10, 4, 2\}$
$N = 848$ (212 *bytes*)	19	$\{2, 16, 6\}$
$N = 856$ (214 *bytes*)	19	$\{428, 224, 652\}$
$N = 864$ (216 *bytes*)	19	$\{2, 16, 6\}$
$N = 752$ (188 *bytes*)	19	$\{376, 224, 600\}$

Table 3.2: Turbo code permutation parameters

level 1

If j *mod.* $2 = 0$ ($j = 0, \ldots, N - 1$), let $(A, B) = (B, A)$ (invert the couple)

level 2

- If j mod. $4 = 0$, then $P = 0$;
- If j mod. $4 = 1$, then $P = N/2 + P_1$;
- If j mod. $4 = 2$, then $P = P_2$;
- If j mod. $4 = 3$, then $P = N/2 + P_3$;

$$i = (P_0 * j + P + 1) \ mod. \ N \tag{3.1}$$

The interleaving relations satisfy the odd/even rule, *i.e.*, when j is even, i is odd and vice-versa). This enables the puncturing patterns to be identical for both encodings.

3.3.3 Rates and puncturing map

Seven code rates are defined for the Turbo mode:

$$R = 1/3,\ 2/5,\ 1/2,\ 2/3,\ 3/4,\ 4/5,\ 6/7$$

This is achieved through selectively deleting the parity bits (puncturing). The puncturing patterns of Table 3.3 are applied. These patterns are identical for both codes C_1 and C_2 (deletion is always done in couples). The puncturing rate is indicated to the Return Channel Satellite Terminals (RCSTs) via the Time-slot Composition Table (TCT) (See DVB-RCS standard sub-clause 8.5.5.4 [57]).

$$1/3\ \begin{matrix} Y' \\ W' \end{matrix}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad 2/5\ \begin{matrix} Y' \\ W' \end{matrix}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

$$1/2\ \begin{matrix} Y' \\ W' \end{matrix}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad 2/3\ \begin{matrix} Y' \\ W' \end{matrix}\begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$3/4\ \begin{matrix} Y' \\ W' \end{matrix}\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad 4/5\ \begin{matrix} Y' \\ W' \end{matrix}\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$6/7\ \begin{matrix} Y' \\ W' \end{matrix}\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Table 3.3: Puncturing patterns for double-binary convolutional turbo codes. "1" = keep

When code rates $R \geq 1/2$, all the second parity bits W' are deleted. Rates 1/3, 2/5, 1/2, 2/3 and 4/5 are exact ones, independently of the block size. Rates 3/4 and 6/7 are exact ones only if N is a multiple of 3. In other cases, the actual rates are very slightly lower than the nominal ones.

Depending on the code rate, the length of the encoded block is depicted in Table 3.4:

Encoded block /Code Rate	$2N+M$ *for* $R<1/2$	$N+M$ *for* $R \geq 1/2$
$R=1/3$	$M=N$	
$R=2/5$	$M=N/2$	
$R=1/2$		$M=N$
$R=2/3$		$M=N/2$
$R=3/4$		$M=N/3$ *if* N *mod.* $3=0$ $M=(N-4)/3+2$ *if* N *mod.* $3=1$ $M=(N-8)/3+3$ *if* N *mod.* $3=2$
$R=4/5$		$M=N/4$
$R=6/7$		$M=N/6$ *if* N *mod.* $3=0$ $M=(N-4)/6+1$ *if* N *mod.* $3=1$ $M=(N-8)/6+2$ *if* N *mod.* $3=2$

Table 3.4: The length of the encoded block

3.3.4 Order of transmission and mapping to QPSK constellation

Two orders of transmission are allowed:

- in the natural order, all couples (A, B) are transmitted first, followed by all couples (Y_1, Y_2) that remain after puncturing and then all couples (W_1, W_2) that remain after puncturing (see figure 3.6);

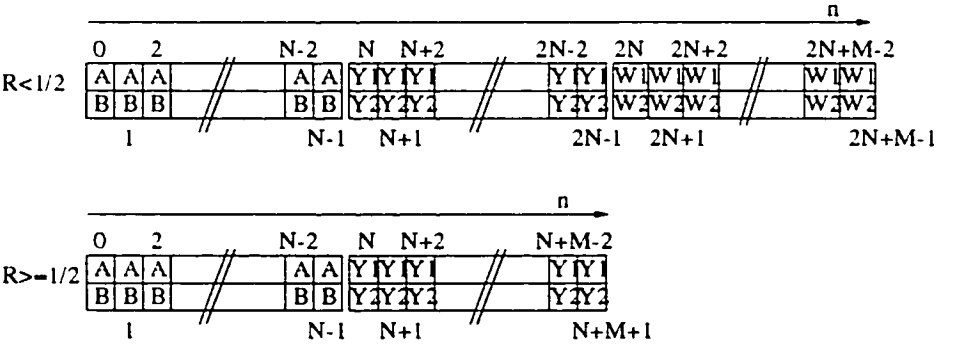


Figure 3.6: Encoded blocks (natural order)

- in the reverse order, the couples (Y_1, Y_2) are transmitted first, in their natural order, followed by the couples (W_1, W_2), if any, and then finally by the couples (A, B).

Each couple is mapped to one QPSK constellation point as shown in Figure 3.8. In figure 3.6, the row with the A symbols is mapped on the I channel (C_1 in figure 3.8).The signal shall be modulated using QPSK, with baseband shaping. Immediately after the preamble insertion, the outputs C_1 and C_2 of the encoder shall be sent without modification to the QPSK bit mapper (see Figure 3.7).

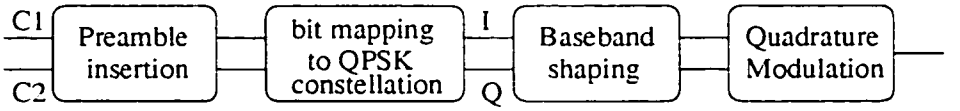


Figure 3.7: Processing after the encoder

Convolutional Gray-coded QPSK modulation with absolute mapping (no differential coding) shall be used. Bit mapping in the QPSK constellation shall follow Figure 3.8. If the normalization factor $1/\sqrt{2}$ is applied to the I and Q components, the corresponding average energy per symbol becomes equal to 1.

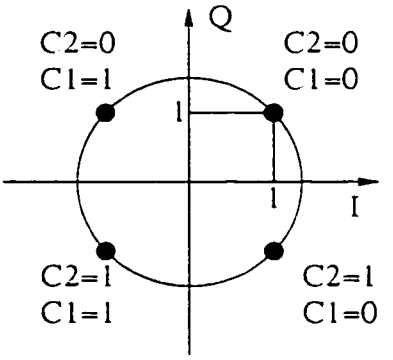


Figure 3.8: Bit mapping into QPSK constellation

The output C_1 of the channel coding shall be mapped to the I channel of the modulation. The output C_2 shall be mapped to the Q channel of the modulation.

3.3.5 Decoder structure

According to the principle of iterative decoding algorithm, the decoder of double-binary convolutional turbo code is designed as shown in Figure 3.9:

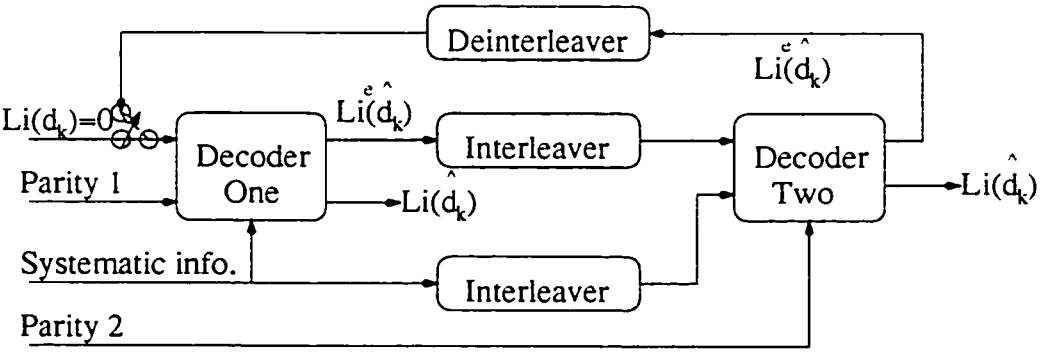


Figure 3.9: Decoder structure of Non-binary convolutional turbo code

The systematic information is the channel value of information bits $d_k = i$, for $i = 00, 01, 10, 11$. Parity 1 and Parity 2 are the channel value of the output of encoders parity bits. $L_i(\hat{d}_k)$ is the log-likelihood ratio and $L_i^e(\hat{d}_k)$ is the extrinsic information.

3.4 Decoding procedure of double-binary convolutional turbo codes

Even though the symbol-by-symbol maximum *a posteriori* (MAP) algorithm is optimal, from an implementation point of view, the component decoding algorithm applied is the Max-Log-MAP algorithm for the complexity/performance compromise. Good convergence, close to the theoretical limits [59] - from 1.0dB to 1.8dB, depending on the coding rate - can be observed, thanks to the double-binary component code.

3.4.1 Decoding rule for CRSC codes with a non-binary Trellis

The trellis of a double-binary feedback convolutional encoder has the structure shown in figure 3.10. Let S_k be the encoder state at time k. The bits d_k is associated with the transition from time $k-1$ to time k. The trellis states at level

$k-1$ and at level k are indexed by the integer S_{k-1} and S_k, respectively.

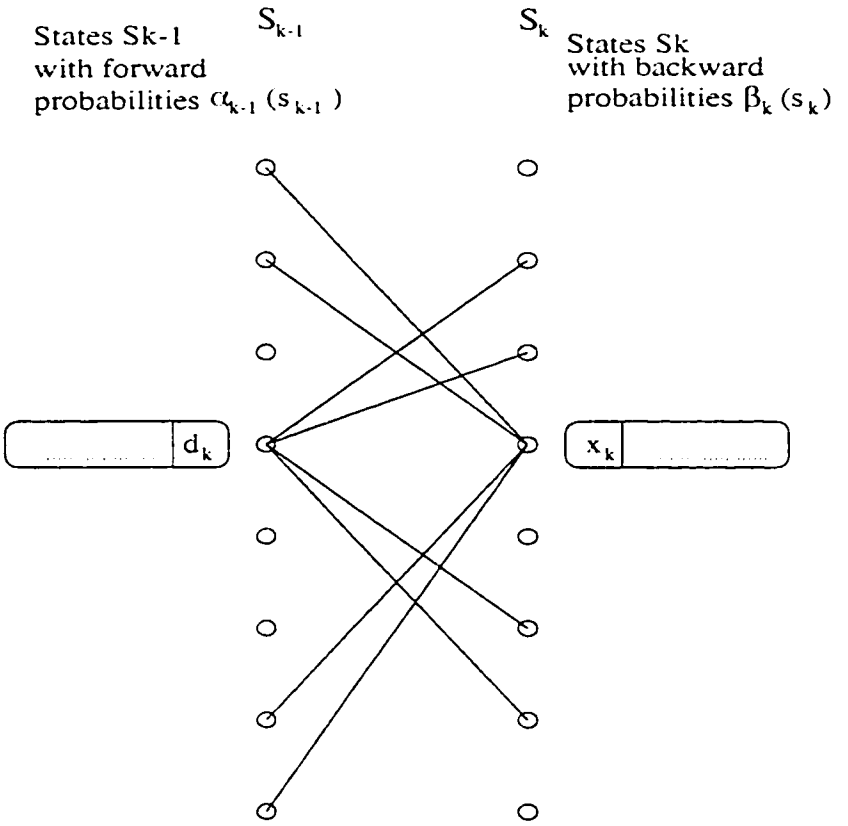


Figure 3.10: Trellis structure of double-binary convolutional codes with feedback encoder

The goal of the MAP algorithm is to provide us with

$$\begin{aligned} L_i(d_k) &= \ln \frac{P_r[d_k = i|Observation]}{P_r[d_k = 0|Observation]} \quad for\ i = 1, 2, 3 \\ &= \ln \frac{\sum_{d_k=i}^{(S_{k-1},S_k)} p(S_{k-1}, S_k, y_k)}{\sum_{d_k=0}^{(S_{k-1},S_k)} p(S_{k-1}, S_k, y_k)} \end{aligned} \tag{3.2}$$

The index pair S_{k-1} and S_k determines the information symbols (couple bits in a symbol) d_k and the coded symbols (couple bits in a symbol) x_k, where d_k is in GF(2^2) with elements {0,1,2,3} from time $k-1$ to time k. The sum of the joint probabilities $p(S_{k-1}, S_k, y_k)$ in the numerator or in the denominator of Equation (3.2) is taken over all existing transitions from state S_{k-1} to state S_k labeled with the information bits $d_k = 0, 1, 2, 3$ (that is, $d_k = 00, 01, 10, 11$. We use decimal notations instead of binary

for simplicity, as well as $x_k = 00, 01, 10, 11$). Assuming a memoryless transmission channel, the joint probability $p(S_{k-1}, S_k, y_k)$ can be written as the product of three independent probabilities

$$\begin{aligned} p(S_{k-1}, S_k, y_k) &= p(S_{k-1}, y_{j<k}) \cdot p(S_k, y_k | S_{k-1}) \cdot p(y_{j>k} | S_k) \\ &= \underbrace{p(S_{k-1}, y_{j<k})} \cdot \underbrace{P(S_k | S_{k-1}) \cdot p(y_k | S_{k-1}, S_k)} \cdot \underbrace{p(y_{j>k} | S_k)} \\ &= \alpha_{k-1}(S_{k-1}) \cdot \gamma_k^i(S_{k-1}, S_k) \cdot \beta_k(S_k) \end{aligned} \tag{3.3}$$

Here $y_{j<k}$ denotes the sequence of received symbols y_j from the beginning of the trellis up to time $k-1$ and $y_{j>k}$ is the corresponding sequence from time $k+1$ up to the end of the trellis. The forward recursion of the MAP algorithm yields

$$\alpha_k(S_k) = \sum_{S_{k-1}} \sum_{i=0}^{3} \gamma_k^i(S_{k-1}, S_k) \cdot \alpha_{k-1}(S_{k-1}) \tag{3.4}$$

The backward recursion yields

$$\beta_{k-1}(S_{k-1}) = \sum_{S_k} \sum_{i=0}^{3} \gamma_k^i(S_{k-1}, S_k) \cdot \beta_k(S_k) \tag{3.5}$$

Whenever a transition between S_{k-1} and S_k exist the branch transition probabilities are given by

$$\gamma_k^i(S_{k-1}, S_k) = P(S_k | S_{k-1}) \cdot p(y_k | S_{k-1}, S_k) = p(y_k | d_k) \cdot P(d_k) \quad for\ i = 0, 1, 2, 3 \tag{3.6}$$

Find the natural logarithm of the branch transition probability metrics as

$$\ln \gamma_k^i(S_{k-1}, S_k) = \overline{\gamma_k^i}(S_{k-1}, S_k) \tag{3.7}$$

and the natural logarithm of $\alpha_k(S_k)$ and $\beta_k(S_k)$ as

$$\begin{aligned} \ln \alpha_k(S_k) &= \overline{\alpha_k}(S_k) \\ &= \ln \sum_{S_{k-1}} \sum_{i=0}^{3} e^{\overline{\gamma_k^i}(S_{k-1}, S_k)} \cdot e^{\overline{\alpha_{k-1}}(S_{k-1})} \end{aligned} \tag{3.8}$$

$$\begin{aligned}\ln \beta_{k-1}(S_k) &= \overline{\beta_{k-1}}(S_k) \\ &= \ln \sum_{s_k} \sum_{i=0}^{3} e^{\overline{\gamma_k^i}(S_{k-1},S_k)} \cdot e^{\overline{\beta_k}(S_k)} \end{aligned} \tag{3.9}$$

Hence, the log-likelihood ratios are represented by

$$\begin{aligned}L_i(d_k) &= \ln \frac{\sum_{d_k=i}^{(S_{k-1},S_k)} \gamma_k^i(S_{k-1},S_k) \cdot \alpha_{k-1}(S_{k-1}) \cdot \beta_k(S_k)}{\sum_{d_k=0}^{(S_{k-1},S_k)} \gamma_k^0(S_{k-1},S_k) \cdot \alpha_{k-1}(S_{k-1}) \cdot \beta_k(S_k)} \quad for \;\; i = 1,2,3 \\ &\quad \ln \frac{\sum_{d_k=i}^{(S_{k-1},S_k)} e^{\overline{\gamma_k^i}(S_{k-1},S_k)} \cdot e^{\overline{\alpha_{k-1}}(S_{k-1})} \cdot e^{\overline{\beta_k}(S_k)}}{\sum_{d_k=0}^{(S_{k-1},S_k)} e^{\overline{\gamma_k^i}(S_{k-1},S_k)} \cdot e^{\overline{\alpha_{k-1}}(S_{k-1})} \cdot e^{\overline{\beta_k}(S_k)}} \end{aligned} \tag{3.10}$$

3.4.2 Simplified Max-Log-MAP algorithm for double-binary convolutional turbo code

First, according to the decoding rule and Equation (3.6) and (3.7), find the logarithm of the branch transition probability as:

$$\ln \gamma_k^i(S_{k-1},S_k) = \overline{\gamma_k^i}(S_{k-1},S_k) = \ln p(y_k|d_k) \cdot P(d_k) \tag{3.11}$$

The distribution of the received parity and systematic symbols are given by

$$\begin{aligned}p[y_k|d_k=i] &= p[y_k^s|x_k^s(i)] \cdot p[y_k^p|x_k^p(i,S_{k-1},S_k)] \\ &= \frac{1}{\pi \cdot N_0} exp\{-\frac{E_s}{N_0}[(y_k^{s,I} - x_k^{s,I}(i))^2 + (y_k^{s,Q} - x_k^{s,Q}(i))^2]\} \\ &\quad \cdot \frac{1}{\pi \cdot N_0} exp\{-\frac{E_s}{N_0}[(y_k^{p,I} - x_k^{p,I}(i,S_{k-1},S_k))^2 + (y_k^{p,I} - x_k^{p,Q}(i,S_{k-1},S_k))^2]\} \\ &= B_k \cdot exp\{\frac{1}{2}L_c \cdot [y_k^{s,I} \cdot x_k^{s,I}(i) + y_k^{s,Q} \cdot x_k^{s,Q}(i)]\} \\ &\quad \cdot exp\{\frac{1}{2}L_c \cdot [y_k^{p,I} \cdot x_k^{p,I}(i,S_{k-1},S_k) + y_k^{p,I} \cdot x_k^{p,Q}(i,S_{k-1},S_k)]\} \end{aligned} \tag{3.12}$$

where y_k^s, y_k^p represent the received systematic and parity symbols, and $y_k^{s,I}$, $y_k^{s,Q}$, $y_k^{p,I}$, $y_k^{p,Q}$ represent the received bit values which transmitted through the I and Q channel respectively; $x_k^s(i)$, $x_k^p(i,S_{k-1},S_k)$ represent the systematic and parity symbols for $i = 0,1,2,3$, and $x_k^{s,I}(i)$, $x_k^{s,Q}(i)$, $x_k^{p,I}(i,S_{k-1},S_k)$, $x_k^{p,Q}(i,S_{k-1},S_k)$ represent the bits of codeword mapped to QPSK constellation respectively.

Here,

$$B_k = (\frac{1}{\pi \cdot N_0})^2 exp\{-\frac{E_s}{N_0}[(y_k^{s,I})^2 + (x_k^{s,I}(i))^2 + (y_k^{s,Q})^2 + (x_k^{s,Q}(i))^2 + (y_k^{p,I})^2 + (x_k^{p,I}(i, S_{k-1}, S_k)^2 + (y_k^{p,I})^2 + (x_k^{p,Q}(i, S_{k-1}, S_k))^2]\} \quad (3.13)$$

Hence,

$$\begin{aligned} \ln \gamma_k^i(S_{k-1}, S_k) &= \overline{\gamma_k^i}(S_{k-1}, S_k) \\ &= \ln P(y_k|d_k) \cdot P(d_k) \\ &= \frac{1}{2}L_c \cdot [y_k^{s,I} \cdot x_k^{s,I}(i) + y_k^{s,Q} \cdot x_k^{s,Q}(i)] + \ln P(d_k) + K \\ &\quad + \frac{1}{2}L_c \cdot [y_k^{p,I} \cdot x_k^{p,I}(i, S_{k-1}, S_k) + y_k^{p,I} \cdot x_k^{p,Q}(i, S_{k-1}, S_k)] \end{aligned} \quad (3.14)$$

Where the constant K includes the constants and common terms that are cancelled in comparisons at the later stages.

Next, compute $\alpha_k(S_k)$ and $\beta_k(S_k)$ as

$$\begin{aligned} \alpha_k(s_k) &= \sum_{S_{k-1}} \sum_{i=0}^{3} \gamma_k^i(S_{k-1}, S_k) \cdot \alpha_{k-1}(S_{k-1}) \\ &= \gamma_k^0(S_{k-1}, S_k) \cdot \alpha_{k-1}(S_{k-1}) + \gamma_k^1(S_{k-1}, S_k) \cdot \alpha_{k-1}(S_{k-1}) \\ &\quad + \gamma_k^2(S_{k-1}, S_k) \cdot \alpha_{k-1}(S_{k-1}) + \gamma_k^3(S_{k-1}, S_k) \cdot \alpha_{k-1}(S_{k-1}) \\ &= e^{\overline{\gamma_k^0}(S_{k-1}, S_k)} \cdot e^{\overline{\alpha_{k-1}}(S_{k-1})} + e^{\overline{\gamma_k^1}(S_{k-1}, S_k)} \cdot e^{\overline{\alpha_{k-1}}(S_{k-1})} \\ &\quad + e^{\overline{\gamma_k^2}(S_{k-1}, S_k)} \cdot e^{\overline{\alpha_{k-1}}(S_{k-1})} + e^{\overline{\gamma_k^3}(S_{k-1}, S_k)} \cdot e^{\overline{\alpha_{k-1}}(S_{k-1})} \end{aligned} \quad (3.15)$$

and then take max(·) function,

$$\begin{aligned} \ln \alpha_k(S_k) &= \overline{\alpha_k}(S_k) \\ &= \ln[e^{\overline{\gamma_k^0}(S_{k-1}, S_k) + \overline{\alpha_{k-1}}(S_{k-1})} + e^{\overline{\gamma_k^1}(S_{k-1}, S_k) + \overline{\alpha_{k-1}}(S_{k-1})} \\ &\quad + e^{\overline{\gamma_k^2}(S_{k-1}, S_k) + \overline{\alpha_{k-1}}(S_{k-1})} + e^{\overline{\gamma_k^3}(S_{k-1}, S_k) + \overline{\alpha_{k-1}}(S_{k-1})}] \\ &\approx \max\{[\overline{\gamma_k^0}(S_{k-1}, S_k) + \overline{\alpha_{k-1}}(S_{k-1})], [\overline{\gamma_k^1}(S_{k-1}, S_k) + \overline{\alpha_{k-1}}(S_{k-1})], \\ &\quad [\overline{\gamma_k^2}(S_{k-1}, S_k) + \overline{\alpha_{k-1}}(S_{k-1})], [\overline{\gamma_k^3}(S_{k-1}, S_k) + \overline{\alpha_{k-1}}(S_{k-1})]\} \end{aligned} \quad (3.16)$$

Similarly

$$\begin{aligned}
\beta_{k-1}(S_{k-1}) &= \sum_{S_k}\sum_{i=0}^{3}\gamma_k^i(S_{k-1},S_k)\cdot\beta_k(S_k) \\
&= \gamma_k^0(S_{k-1},S_k)\cdot\beta_k(S_k)+\gamma_k^1(S_{k-1},S_k)\cdot\beta_k(S_k) \\
&\quad +\gamma_k^2(S_{k-1},s_k)\cdot\beta_k(S_k)+\gamma_k^3(S_{k-1},S_k)\cdot\beta_k(S_k) \\
&= e^{\overline{\gamma_k^0}(S_{k-1},S_k)}\cdot e^{\overline{\beta_k}(S_k)}+e^{\overline{\gamma_k^1}(S_{k-1},S_k)}\cdot e^{\overline{\beta_k}(S_k)} \\
&\quad +e^{\overline{\gamma_k^2}(S_{k-1},S_k)}\cdot e^{\overline{\beta_k}(S_k)}+e^{\overline{\gamma_k^3}(S_{k-1},S_k)}\cdot e^{\overline{\beta_k}(S_k)}
\end{aligned} \tag{3.17}$$

$$\begin{aligned}
\ln\beta_{k-1}(S_{k-1}) &= \overline{\beta_k}(S_k) \\
&= \ln[e^{\overline{\gamma_k^0}(S_{k-1},S_k)+\overline{\beta_k}(S_k)}+e^{\overline{\gamma_k^1}(S_{k-1},S_k)+\overline{\beta_k}(S_k)} \\
&\quad +e^{\overline{\gamma_k^2}(S_{k-1},S_k)+\overline{\beta_k}(S_k)}+e^{\overline{\gamma_k^3}(S_{k-1},S_k)+\overline{\beta_k}(S_k)}] \\
&\approx \max\{[\overline{\gamma_k^0}(S_{k-1},S_k)+\overline{\beta_k}(S_k)],[\overline{\gamma_k^1}(S_{k-1},S_k)+\overline{\beta_k}(S_k)], \\
&\quad [\overline{\gamma_k^2}(S_{k-1},S_k)+\overline{\beta_k}(S_k)],[\overline{\gamma_k^3}(S_{k-1},S_k)+\overline{\beta_k}(S_k)]\}
\end{aligned} \tag{3.18}$$

For iterative decoding of circular trellis, Tail-biting is

$$\begin{aligned}
\overline{\alpha_0}(S_0) &= \overline{\alpha_N}(S_N) \quad for\ \forall S_0 \\
\overline{\beta_N}(S_N) &= \overline{\beta_0}(S_0) \quad for\ \forall s_N
\end{aligned} \tag{3.19}$$

Therefore, computing the log-likelihood ratios follows the Equation (3.10) and takes max(·) function as

$$\begin{aligned}
L_i(\hat{d}_k) &\approx \max_{(s_{k-1},s_k)}[\overline{\gamma_k^i}(S_{k-1},S_k)+\overline{\alpha_{k-1}}(S_{k-1})+\overline{\beta_k}(S_k)] \\
&\quad -\max_{(s_{k-1},s_k)}[\overline{\gamma_k^0}(S_{k-1},S_k)+\overline{\alpha_{k-1}}(S_{k-1})+\overline{\beta_k}(S_k)]
\end{aligned} \tag{3.20}$$

Moreover, to separate the Log-likelihood ratios into intrinsic, systematic and extrinsic information, define:

$$\begin{aligned}
\gamma_k^{i(e)}(S_{k-1},S_k) &= exp\{\frac{1}{2}L_c\cdot[y_k^{p,I}\cdot x_k^{p,I}(i,S_{k-1},S_k) \\
&\quad +y_k^{p,I}\cdot x_k^{p,Q}(i,S_{k-1},S_k)]\}\cdot A_k
\end{aligned} \tag{3.21}$$

here, the constant

$$A_k = \frac{1}{\pi \cdot N_0} exp\{-\frac{E_s}{N_0}[(y_k^{p,I})^2 + (x_k^{p,I}(i, S_{k-1}, S_k))^2 + (y_k^{p,I})^2 + (x_k^{p,Q}(i, S_{k-1}, S_k))^2]\} \tag{3.22}$$

Hence, the logarithm of the branch transition operation reduce to the expression with

$$\begin{aligned} \ln \gamma_k^{i(e)}(S_{k-1}, S_k) &= \overline{\gamma_k^{i(e)}}(S_{k-1}, S_k) \\ &= \frac{1}{2} L_c \cdot [y_k^{p,I} \cdot x_k^{p,I}(i, S_{k-1}, S_k) + y_k^{p,I} \cdot x_k^{p,Q}(i, S_{k-1}, S_k)] + K' \end{aligned}$$

where the constant K' includes the constants and common terms that are cancelled in comparisons in the later stages.

In another way, find the Log-likelihood ratios as

$$\begin{aligned} L_i(\hat{d}_k) &= \ln \frac{\sum_{d_k=i}^{(S_{k-1},S_k)} \gamma_k^i(S_{k-1}, S_k) \cdot \alpha_{k-1}(S_{k-1}) \cdot \beta_k(S_k)}{\sum_{d_k=0}^{(S_{k-1},S_k)} \gamma_k^0(S_{k-1}, S_k) \cdot \alpha_{k-1}(S_{k-1}) \cdot \beta_k(S_k)} \quad for\ i = 1, 2, 3 \\ &= \ln \frac{exp\{\frac{1}{2} L_c \cdot [y_k^{s,I} \cdot x_k^{s,I}(i) + y_k^{s,Q} \cdot x_k^{s,Q}(i)]\} \cdot P[d_k = i]}{exp\{\frac{1}{2} L_c \cdot [y_k^{s,I} \cdot x_k^{s,I}(0) + y_k^{s,Q} \cdot x_k^{s,Q}(0)]\} \cdot P[d_k = 00]} \\ &\quad + \ln \frac{\sum_{d_k=i}^{(S_{k-1},S_k)} \gamma_k^{i(e)}(S_{k-1}, S_k) \cdot \alpha_{k-1}(S_{k-1}) \cdot \beta_k(S_k)}{\sum_{d_k=0}^{(S_{k-1},S_k)} \gamma_k^{0(e)}(S_{k-1}, S_k) \cdot \alpha_{k-1}(S_{k-1}) \cdot \beta_k(S_k)} \\ &= \{\frac{1}{2} L_c \cdot [y_k^{s,I} \cdot x_k^{s,I}(i) + y_k^{s,Q} \cdot x_k^{s,Q}(i)] - \frac{1}{2} L_c \cdot [y_k^{s,I} \cdot x_k^{s,I}(0) + y_k^{s,Q} \cdot x_k^{s,Q}(0)]\} \\ &\quad + \underbrace{\ln \frac{P[d_k = i]}{P[d_k = 0]}}_{L_i(d_k)} + \underbrace{\ln \frac{\sum_{d_k=i}^{(S_{k-1},S_k)} \gamma_k^{i(e)}(S_{k-1}, S_k) \cdot \alpha_{k-1}(S_{k-1}) \cdot \beta_k(S_k)}{\sum_{d_k=0}^{(S_{k-1},S_k)} \gamma_k^{0(e)}(S_{k-1}, S_k) \cdot \alpha_{k-1}(S_{k-1}) \cdot \beta_k(S_k)}}_{L_i^e(\hat{d}_k)} \end{aligned} \tag{3.23}$$

So the extrinsic information can be calculated as

$$L_i^e(\hat{d}_k) = \ln \frac{\sum_{d_k=i}^{(S_{k-1},S_k)} \gamma_k^{i(e)}(S_{k-1}, S_k) \cdot \alpha_{k-1}(S_{k-1}) \cdot \beta_k(S_k)}{\sum_{d_k=0}^{(S_{k-1},S_k)} \gamma_k^{0(e)}(S_{k-1}, S_k) \cdot \alpha_{k-1}(S_{k-1}) \cdot \beta_k(S_k)} \tag{3.24}$$

$$\begin{aligned} L_i^e(\hat{d}_k) &= L_i(\hat{d}_k) - \frac{1}{2} L_c \cdot [y_k^{s,I} \cdot x_k^{s,I}(i) + y_k^{s,Q} \cdot x_k^{s,Q}(i)] \\ &\quad + \frac{1}{2} L_c \cdot [y_k^{s,I} \cdot x_k^{s,I}(0) + y_k^{s,Q} \cdot x_k^{s,Q}(0)] - \ln \frac{P[d_k = i]}{P[d_k = 0]} \end{aligned} \tag{3.25}$$

Compute symbol probabilities for next Decoder:

$$L_i(d_k) = L_i^e(\hat{d}_k) = \ln \frac{P[d_k = i]}{P[d_k = 0]} \quad for\ i = 1, 2, 3 \quad from\ previouse\ Decoder \tag{3.26}$$

Since

$$\begin{cases} L_0^e(\hat{d}_k) &= \ln \frac{P[d_k=00]}{P[d_k=00]} = \ln 1 = 0 \\ L_1^e(\hat{d}_k) &= \ln \frac{P[d_k=01]}{P[d_k=00]} \\ L_2^e(\hat{d}_k) &= \ln \frac{P[d_k=10]}{P[d_k=00]} \\ L_3^e(\hat{d}_k) &= \ln \frac{P[d_k=11]}{P[d_k=00]} \end{cases} \tag{3.27}$$

Then

$$\begin{cases} P[d_k = 01] &= e^{L_1^e(\hat{d}_k)} \cdot P[d_k = 00] \\ P[d_k = 10] &= e^{L_2^e(\hat{d}_k)} \cdot P[d_k = 00] \\ P[d_k = 11] &= e^{L_3^e(\hat{d}_k)} \cdot P[d_k = 00] \end{cases} \tag{3.28}$$

and

$$P[d_k = 00] + P[d_k = 01] + P[d_k = 10] + P[d_k = 11] = 1 \tag{3.29}$$

Hence

$$\begin{cases} P[d_k = 00] &= \frac{1}{1+e^{L_1^e(\hat{d}_k)}+e^{L_2^e(\hat{d}_k)}+e^{L_3^e(\hat{d}_k)}} \\ P[d_k = 01] &= \frac{e^{L_1^e(\hat{d}_k)}}{1+e^{L_1^e(\hat{d}_k)}+e^{L_2^e(\hat{d}_k)}+e^{L_3^e(\hat{d}_k)}} \\ P[d_k = 10] &= \frac{e^{L_2^e(\hat{d}_k)}}{1+e^{L_1^e(\hat{d}_k)}+e^{L_2^e(\hat{d}_k)}+e^{L_3^e(\hat{d}_k)}} \\ P[d_k = 11] &= \frac{e^{L_3^e(\hat{d}_k)}}{1+e^{L_1^e(\hat{d}_k)}+e^{L_2^e(\hat{d}_k)}+e^{L_3^e(\hat{d}_k)}} \end{cases} \tag{3.30}$$

Using max(·) Function

$$\begin{cases} \ln P[d_k = 00] &= -\max[0, L_1^e(\hat{d}_k), L_2^e(\hat{d}_k), L_3^e(\hat{d}_k)] \\ \ln P[d_k = 01] &= L_1^e(\hat{d}_k) - \max[0, L_1^e(\hat{d}_k), L_2^e(\hat{d}_k), L_3^e(\hat{d}_k)] \\ \ln P[d_k = 10] &= L_2^e(\hat{d}_k) - \max[0, L_1^e(\hat{d}_k), L_2^e(\hat{d}_k), L_3^e(\hat{d}_k)] \\ \ln P[d_k = 11] &= L_3^e(\hat{d}_k) - \max[0, L_1^e(\hat{d}_k), L_2^e(\hat{d}_k), L_3^e(\hat{d}_k)] \end{cases} \tag{3.31}$$

3.4.3 Initialization and Decision of decoding procedure

Assume equally likely information symbols: we do not have any *a priori* information available for the first iteration, we initialize

$$\begin{cases} L_0(d_k = 00) &= 0 \\ L_1(d_k = 01) &= 0 \\ L_2(d_k = 10) &= 0 \\ L_3(d_k = 11) &= 0 \end{cases} \tag{3.32}$$

and according to equation (3.30), we have

$$P_0[d_k = 00] = \frac{1}{1 + e^{L_1(d_k)} + e^{L_2(d_k)} + e^{L_3(d_k)}} = \frac{1}{4}$$

$$\begin{aligned} P_0[d_k = 01] &= \frac{1}{4} \\ P_0[d_k = 10] &= \frac{1}{4} \\ P_0[d_k = 11] &= \frac{1}{4} \end{aligned}$$

Using max(·) Function:

$$\begin{aligned} \ln P_0[d_k = 00] &= -max(0, L_1, L_2, L_3) = 0 \\ \ln P_1[d_k = 01] &= L_1 - max(0, L_1, L_2, L_3) = 0 \\ \ln P_2[d_k = 10] &= L_2 - max(0, L_1, L_2, L_3) = 0 \\ \ln P_3[d_k = 11] &= L_3 - max(0, L_1, L_2, L_3) = 0 \end{aligned} \tag{3.33}$$

Similarly, equally-likelihood assumption for all symbols

$$\begin{aligned} \alpha_0(s_0) &= 1 \quad for\ \forall s_0 \\ \beta_N(s_N) &= 1 \quad for\ \forall s_N \end{aligned} \tag{3.34}$$

take logarithm

$$\begin{aligned} \overline{\alpha_0}(S_0) &= \ln \alpha_0(s_0) = 0 \quad for\ \forall s_0 \\ \overline{\beta_N}(S_N) &= \ln \beta_N(s_N) = 0 \quad for\ \forall s_N \end{aligned} \tag{3.35}$$

and initialize $\overline{\gamma_0^t} = 0$.

The reliability value of the channel

$$L_c = 4a \cdot \frac{E_s}{N_0} = 4a \cdot R_c \cdot \frac{E_b}{N_0} \tag{3.36}$$

where R_c is the code rate.

After several decoding iterations, the decisions are made according to:

$$\hat{d}_k = \begin{cases} 01 & if\ L(\hat{d}_k) = L_1(\hat{d}_k)\ and\ L_1(\hat{d}_k) > 0 \\ 10 & if\ L(\hat{d}_k) = L_2(\hat{d}_k)\ and\ L_2(\hat{d}_k) > 0 \\ 11 & if\ L(\hat{d}_k) = L_3(\hat{d}_k)\ and\ L_3(\hat{d}_k) > 0 \\ 00 & else \end{cases} \tag{3.37}$$

where

$$L(\hat{d}_k) = \max(L_1(\hat{d}_k), L_2(\hat{d}_k), L_3(\hat{d}_k)) \tag{3.38}$$

3.5 Simulation results

Table 3.5 gives some examples of the DVB-RCS turbo code performance observed over a Gaussian channel at Frame Error Rate ($FER = 10^{-4}$)[1], compared to the theoretical limits [59] and the simulation results reported in [56].

Code Rate	Theoretical limits	In paper [56]	In this thesis
1/2	1.3 dB	2.3 dB	2.3 dB
2/3	2.2 dB	3.3 dB	3.3 dB
3/4	2.6 dB	3.9 dB	4.05 dB
4/5	3.1 dB	4.6 dB	4.8 dB
6/7	3.8 dB	5.2 dB	5.6 dB

Table 3.5: $E_b/N_0(dB)$ *at* $FER = 10^{-4}$, 8-iteration, simulation over AWGN channel with Max-Log-MAP algorithm. ATM cells, 53 bytes.

The Figure 3.11 exhibits the performance of block size 53 bytes with the simplified Max-Log-MAP algorithm. So far, there is no any error floor happened. In

[1]Here, we have 100 bit-error events for all simulations in this thesis. See Appendix A.

[56], FER down to 10^{-8} (equivalent to $BER = 10^{-10}/10^{-11}$), the measurements show the absence of error floor.

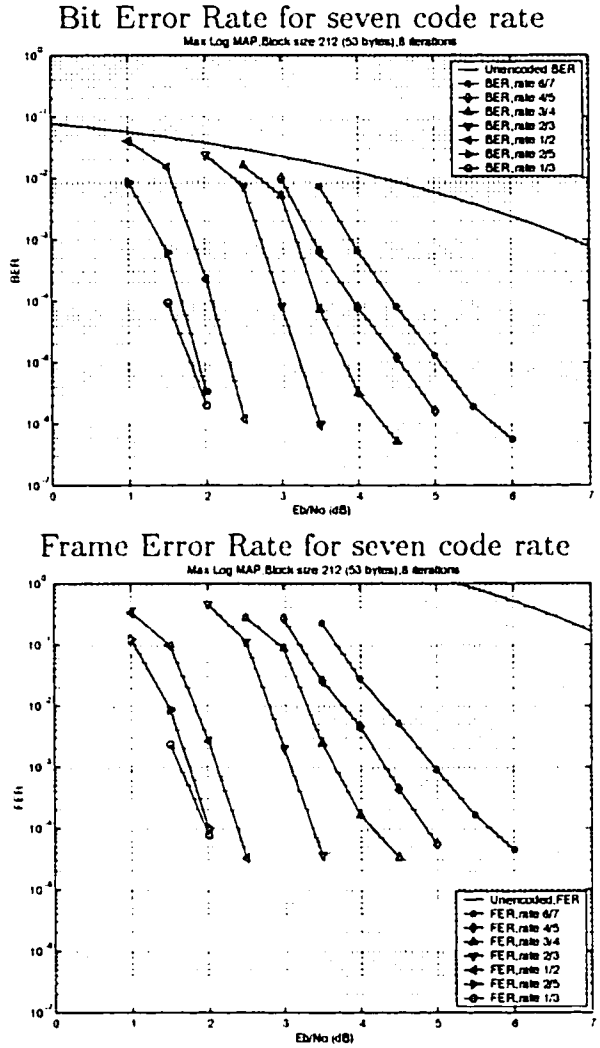


Figure 3.11: Bit Error Rate and Frame Error Rate for seven code rate.

3.6 Conclusion

Using double-binary convolutional turbo codes leads to many advantages [56]:

- Adopting circular coding avoid the degradation of the spectral efficiency of

the transmission when forcing the value of the encoder state at the end of the encoding stage by the addition of tail bits.

- The influence of puncturing is less significant than with binary codes (the natural rate of the double-binary turbo code being 1/2 instead of 1/3 for a binary turbo code).

- The use of double-binary codes also makes the introduction of two-level permutations possible.

- The performance degradation due to the application of a simplified version of the MAP algorithm is less significant in the case of double-binary codes (less than 0.1 dB) than in the case of binary codes (0.3 dB to 0.4 dB), and provide better error-correcting performance than binary codes for equivalent implementation complexity.

- From an implementation point of view, the bit rate at the decoder output is twice that of a binary decoder performing the same number of iterations, with the same circuit clock frequency and with an equivalent complexity per decoded bit (the material complexity of the decoder, for a given memory length, is about twice the complexity of a binary decoder, but two bits are decoded at the same time). Moreover, given the data block size, the latency of the decoder is divided by 2 compared with the binary case because the size of the permutation matrix is halved.

Therefore, double-binary CRSC code that was proposed for DVB-RCS applications is powerful, very flexible and can be implemented with reasonable complexity. Moreover, double-binary CRSC codes are compatible with other techniques applied to error floor optimization.

The system model for the whole encoding/decoding procedure is shown in Figure 3.12.

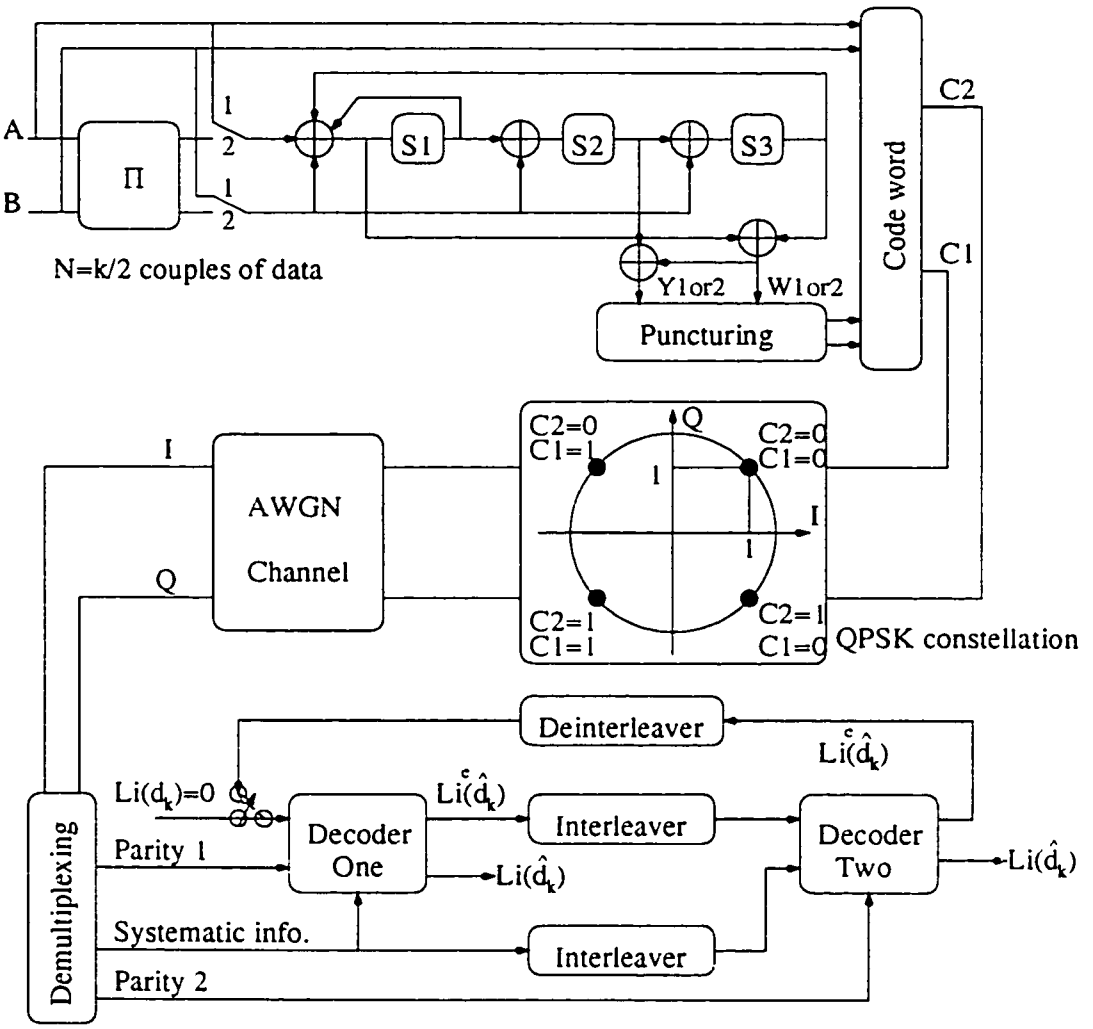


Figure 3.12: system model with double-binary CRSC code

Chapter 4

Fixed point implementation of Turbo-decoder

As discussed above, double-binary CRSC codes have an amazing error correcting capability and are, therefore, very attractive for many applications. The complexity of a turbo-decoder is much higher than the complexity of the encoder. Thus, we put emphasis on the decoder. In a real-time decoder, the decoding algorithms need to have low complexity and need to be implemented using fixed point arithmetic. The Max-Log-MAP algorithm discussed in this thesis is simple enough for the complexity/performance compromise. On the other hand, low cost and low energy consumption are extremely important issues for turbo-decoder implementation. Consequently, fixed point arithmetic and quantization, are unavoidable issues in the implementations of advanced channel decoding techniques. In this chapter, the input data quantization and the effect of correction coefficient are presented.

4.1 Introduction

Quantization is the process of representing the data with one or a few bits of precision. In channel coding, the channel symbols are corrupted by the channel noise

and interference in digital communication system. Applying the quantizer at the receiver makes the channel-decoder work with finite precision or with fixed-point arithmetic.

There are few strategies for turbo-decoder quantization depending on different decoding algorithms. Input data quantization is essential and assuming finite accuracy of the internal values is addressed in [60][61] upon MAP, Log-MAP, Max-Log-MAP and APRI-SOVA decoding algorithms. Another strategy is to presume an unified quantization of all signals [62][63] and a systematic approach towards an internal quantization scheme of a 4-state turbo-decoder with finite accuracy of the input data as discussed in [64]. The first investigation of combined bit-width optimization of input data and internal data for an 8-state turbo-decoder based on parameters relevant for UMTS is discussed in [65].

The resulting bit-true models of the turbo-decoder may differ for software, hardware and mixed hardware/software target architectures as digital signal processing algorithms. These algorithms, like the MAP algorithm used in a turbo-decoder, are usually specified in the floating-point domain. Fixed-point number representation is mandatory for most target architectures, thus transformation from floating-point to fixed-point is a necessary step towards an actual implementation [24]. Primary goal for a software implementation is to find a fixed-point model that corresponds to the given bit-width of the DSP. Further bit-width minimization can reduce the switching activity and has thus influence on the power consumption. Primary goal for a dedicated hardware implementation is to choose all bit-widths as small as possible, resulting in a reduction of area and power consumption. Hence, an optimized quantization has a large impact on the implementation cost. Both 3-bit and 4-bit quantization are discussed in this chapter.

The strategy of turbo-decoder quantization described in [65], is optimal for MAP or Log-MAP decoding algorithm with m-bit input samples, and it is impossible in the case of a Max-Log-MAP decoding algorithm implementation because of the

approximation

$$E_{i=1}^{k} x_i = \ln \sum_{i=1}^{k} e^{x_i} \approx \max(x_i). \tag{4.1}$$

In this chapter, a different scheme of input data quantization for 8-state double-binary CRSC code is designed for a wide range of coding rates. The system model for turbo-decoder quantization is depicted in Figure 4.1. Actually, the QPSK demodulation, quantizer and turbo decoder are combined together in the simulation, and also only the channel output, the input data of decoder is quantized.

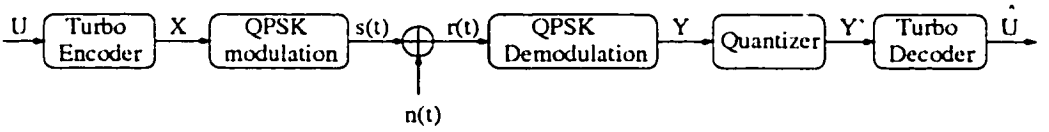


Figure 4.1: System model for quantization

4.2 Input Data Quantization

An ideal turbo-decoder would work with finite precision, or at least with fixed-point numbers. In practical systems, the received channel symbols should be quantized with one or a few bits of precision in order to reduce the complexity of the turbo decoder.

The usual quantization precision is three bits which is introduced in [66]. For the QPSK modulation and an AWGN channel, the received values are corrupted with a Gaussian distribution (see Figure 4.2) around the transmitted symbols {-1, 1}. More than 99% of the occurring values are covered by limiting the dynamic-range of the received channel values to [-4, 4] according to $\pm 1 \pm 3\sigma$[1]. This dynamic-range is reasonable and can be represented by 3 bits in a uniform quantization. Using more bits results in higher complexity, but less degradation in performance. Let's

[1]For Gaussian distribution, the distribution function is given by $F(x) = G(\frac{x-\eta}{\sigma})$ with the notation $N(\eta;\sigma)$ and RV x. The probability of the values of RV x between $(\eta - 3\sigma, \eta + 3\sigma)$ is $P\{|x - \eta| < 3\sigma\} = P\{\eta - 3\sigma < x < \eta + 3\sigma\} = 0.9974$.

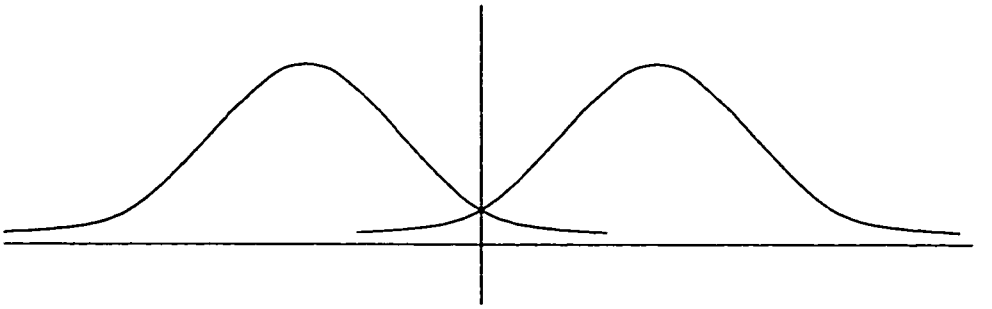


Figure 4.2: The distribution of the transmitted symbols

use a uniform, 3-bit quantizer having the input/output relationship shown in the Figure 4.3, where D is the size of the quantization step.

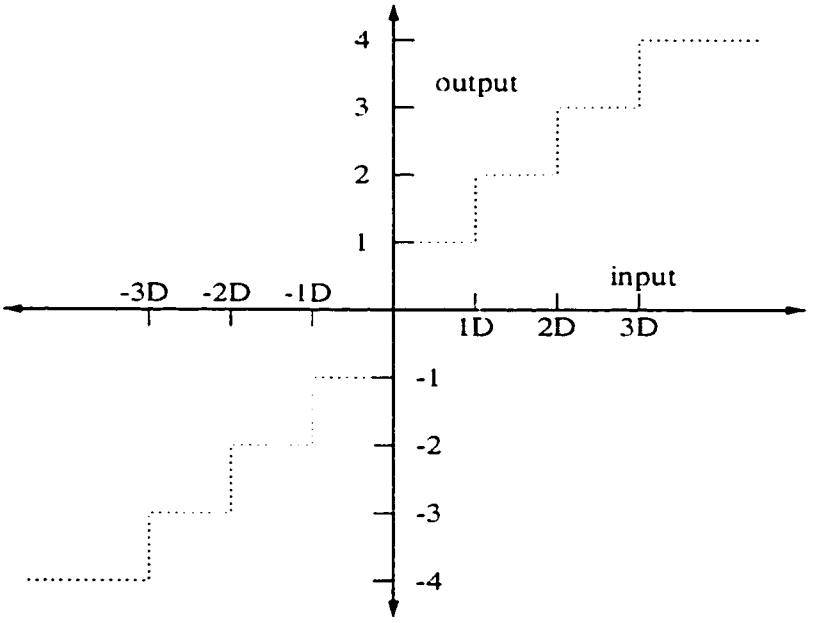


Figure 4.3: Quantizer model in 3-bit

The selection of the quantizing step size is an important consideration because it can have a significant effect on the performance. The step size, D, can be chosen as fixed value upon different coding rates, for example, $D = 11$, or calculated according to the formula $D = f \cdot \sigma = f \cdot \sqrt{1/(2 \cdot (E_s/N_0))}$, where E_s/N_0 is the energy per symbol to noise density ratio, and f is a factor depending on different coding rate.

Denote the channel symbols at the receiver as $y_k^{s,I}$, $y_k^{s,Q}$, $y_k^{p,I}$, $y_k^{p,Q}$ as in Chapter

3. Since the branch transition probability

$$\begin{aligned}\ln \gamma_k^i(S_{k-1}, S_k) &= \frac{1}{2}L_c \cdot [y_k^{s,I} \cdot x_k^{s,I}(i) + y_k^{s,Q} \cdot x_k^{s,Q}(i)] + \ln P(d_k) + K \\ &\quad + \frac{1}{2}L_c \cdot [y_k^{p,I} \cdot x_k^{p,I}(i, S_{k-1}, S_k) + y_k^{p,I} \cdot x_k^{p,Q}(i, S_{k-1}, S_k)] \quad (4.2)\end{aligned}$$

and the extrinsic information

$$\begin{aligned}L_i^e(\hat{d}_k) &= L_i(\hat{d}_k) - \frac{1}{2}L_c \cdot [y_k^{s,I} \cdot x_k^{s,I}(i) + y_k^{s,Q} \cdot x_k^{s,Q}(i)] \\ &\quad + \frac{1}{2}L_c \cdot [y_k^{s,I} \cdot x_k^{s,I}(0) + y_k^{s,Q} \cdot x_k^{s,Q}(0)] - \ln \frac{P[d_k = i]}{P[d_k = 0]} \qquad (4.3)\end{aligned}$$

relate to the received channel symbols, where L_c is the reliability value of the channel and $i = 1, 2, 3$, they are quantized after being multiplied by $\frac{1}{2} \cdot L_c$.

The input data of decoder is multiplied by 2^5 and truncated at -128 and 127 according to the limited dynamic-range of the received channel values to [-4, 4]. Since the input data from -128 to 127, an 8-bit look-up table with indices from 0 to 256 is enough to cover the occurring values of the decoder input data. Thus, all the calculations of decoder are in integer. This resulting bit-true model of the turbo-decoder not only corresponds to the given bit-width of the DSP, but also all bit-widths are as small as possible. Therefore, the energy consumption is low. Moreover, the step size is very flexible for coding rate changes.

Simulation results show that the performance of a decoder with 3-bit quantization is very sensitive to the step size chosen. In Figure 4.5 and Figure 4.6, the selection of step size depends on the code rate and the performance of adaptive step size is better than that of fixed step size. The parameters of fixed step size and adaptive step size are presented in Table 4.3.

To achieve better performance with quantization, 4-bit quantization is designed in the same way as showed in Figure 4.4. There are two kinds of step size, fixed and adaptive step sizes.

For high code rate, we have to modify the step size of the quantizer. For

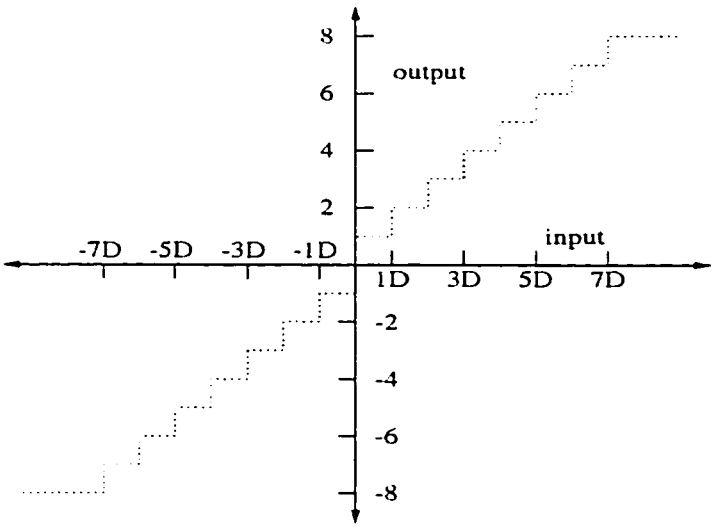


Figure 4.4: 4-bit quantization level

adaptive step size, D, can be calculated according to the formula

$$\begin{aligned} D &= f \cdot \sigma_s \\ &= f \cdot \sqrt{1+\sigma^2} \\ &= f \cdot \sqrt{1+1/(2 \cdot (E_s/N_0))} \end{aligned} \tag{4.4}$$

where E_s/N_0 is the energy per symbol to noise density ratio and f is a factor upon different coding rates.

Figure 4.7 and Figure 4.8 show that 4-bit quantization of the channel input data is a reasonable compromise between implementation complexity and degradation of decoding performance. The performance of adaptive step size is better than that of fixed step size and very close to the unquantized performance, which we can say there are no degradation in the decoding performance.

4.3 Effect of correction coefficient

Max-Log-MAP algorithm is derived from the Log-MAP algorithm by approximating $\ln \sum_{i=1}^{k} e^{x_i}$ with $\max(x_i)$. This approximation results in some degradation in the

performance compared to that of MAP algorithm. Here we discuss the correction coefficient required to apply so that the performance of the Max-Log-MAP algorithm approaches that of the MAP algorithm.

For a binary convolutional turbo code, the correct function in Equation 2.30 is $\ln(1+e^{-|x-y|})$. The look-up table can be chosen as in Table 4.1:

$\lvert x-y\rvert$	0.00	0.25	0.50	0.75	1.00	1.25	1.50	1.75	>2.00
$\ln(1+e^{-\lvert x-y\rvert})$	0.69	0.58	0.47	0.39	0.31	0.25	0.20	0.16	0

Table 4.1: Look-up table for correction term in binary convolutional turbo code

A 1-bit approximation is given by

$$\ln(1+e^{-|x-y|}) = \begin{cases} 3.0/8.0 & if\ x < 2.0 \\ 0 & otherwise \end{cases} \tag{4.5}$$

It has been observed that the performance degradation due to the application of a simplified version - Max-Log-MAP algorithm of the MAP algorithm is less significant in the case of double-binary codes (less than 0.1dB) than in the case of binary codes (0.3 to 0.4 dB) [56]. The following equation is approximated by a very simple function

$$\begin{aligned} \ln(e^x + e^y + e^z + e^w) &= \max(x,y,z,w) \\ &\quad + \ln(e^{x-\max(x,y,z,w)} + e^{y-\max(x,y,z,w)} \\ &\quad + e^{z-\max(x,y,z,w)} + e^{w-\max(x,y,z,w)}) \end{aligned} \tag{4.6}$$

where

$$0 < \ln(e^{x-\max(x,y,z,w)} + e^{y-\max(x,y,z,w)} + e^{z-\max(x,y,z,w)} + e^{w-\max(x,y,z,w)}) \leq \ln 4 \tag{4.7}$$

Define

$$\begin{aligned} &\ln(e^{x-\max(x,y,z,w)} + e^{y-\max(x,y,z,w)} + e^{z-\max(x,y,z,w)} + e^{w-\max(x,y,z,w)}) \\ =\ &\ln(1+e^{-|a|}+e^{-|b|}+e^{-|c|}) \end{aligned} \tag{4.8}$$

$\|a\|$	$\|b\|$	$\|c\|$	$\ln(1+e^{-\|a\|}+e^{-\|b\|}+e^{-\|c\|})$
0.00	0.00	0.00	1.386
0.25	0.25	0.25	1.204
0.50	0.50	0.50	1.037
0.75	0.75	0.75	0.883
1.00	1.00	1.00	0.744
1.25	1.25	1.25	0.620
1.50	1.50	1.50	0.512
1.75	1.75	1.75	0.420
2.00	2.00	2.00	0.341
2.50	2.50	2.50	0.220
3.00	3.00	3.00	0.139

Table 4.2: Look-up table for correction term

where $-|a|, -|b|, -|c|$ are the three values among $x-\max(x,y,z,w)$, $y-\max(x,y,z,w)$, $z-\max(x,y,z,w)$, or $w-\max(x,y,z,w)$.

Since internal data quantization is impossible in the case of a Max-Log-MAP decoding algorithm implementation and the degradation of performance is less significant in the case of double-binary codes, the correction coefficient is trivial. In this thesis, the approximation is given by

$$\ln(1+e^{-|a|}+e^{-|b|}+e^{-|c|}) = \begin{cases} 5.0/8.0 & if \quad \max(|a|,|b|,|c|) < 2.0 \\ 0 & otherwise \end{cases} \tag{4.9}$$

The simulation results show that a look-up table of two level doesn't affect the the performance of double-binary CRSC codes very much (see Figure 4.9). Obviously, the more look-up table levels chosen, the better performance of Max-Log-MAP decoding algorithm would be.

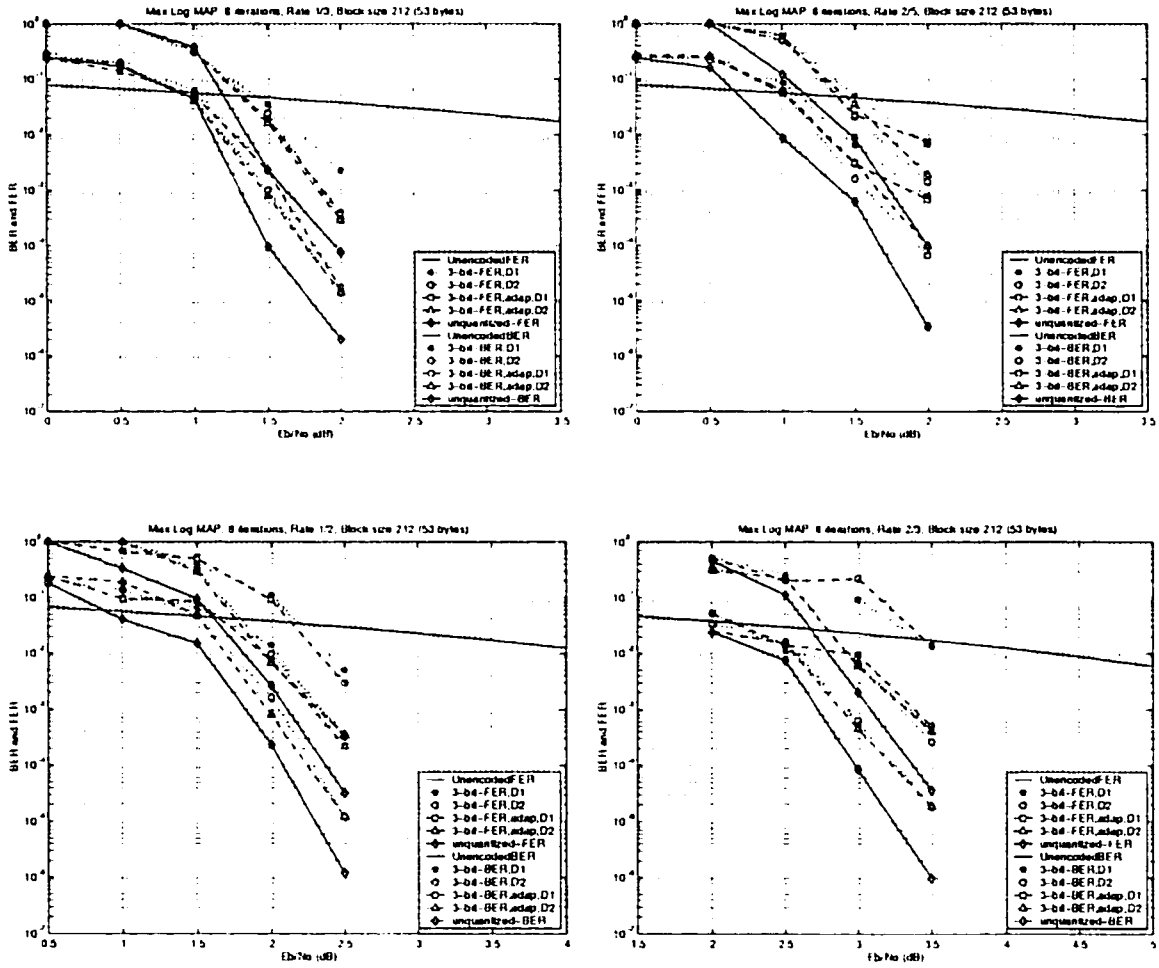


Figure 4.5: 3-bit quantization. code rate=1/3, 2/5, 1/2, 2/3. The parameters of decision level refer to Table 4.3.

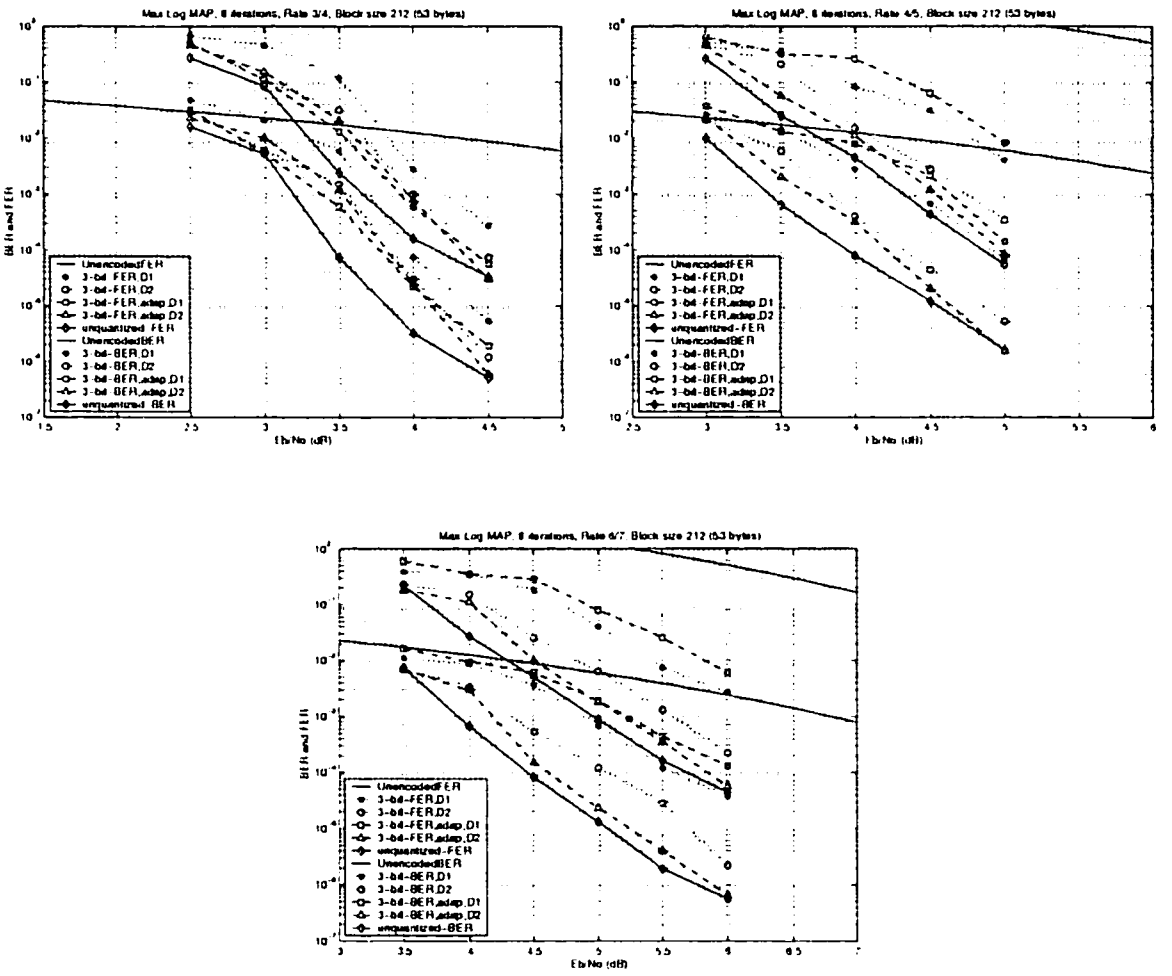


Figure 4.6: 3-bit quantization, code rate: 3/4, 4/5, 6/7. The parameters of step size refer to Table 4.3.

Code rate	Fixed step size	*Adaptive step size* $\sigma = \sqrt{1/2 * (E_s/N_0)}$
$R = 1/3$	$D1 = -128, -31, -21, -11, 0, 11, 21, 31, 127$ $D2 = -128, -56, -36, -18, 0, 18, 36, 56, 127$	$D1 = 0.51 * 32 * \sigma$ $D2 = 0.53 * 32 * \sigma$
$R = 2/5$	$D1 = -128, -31, -21, -11, 0, 11, 21, 31, 127$ $D2 = -128, -56, -36, -18, 0, 18, 36, 56, 127$	$D1 = 0.45 * 32 * \sigma$ $D2 = 0.55 * 32 * \sigma$
$R = 1/2$	$D1 = -128, -31, -21, -11, 0, 11, 21, 31, 127$ $D2 = -128, -63, -42, -21, 0, 21, 42, 63, 127$	$D1 = 0.5 * 32 * \sigma$ $D2 = 0.55 * 32 * \sigma$
$R = 2/3$	$D1 = -128, -31, -21, -11, 0, 11, 21, 31, 127$ $D2 = -128, -63, -42, -21, 0, 21, 42, 63, 127$	$D1 = 0.5 * 32 * \sigma$ $D2 = 0.55 * 32 * \sigma$
$R = 3/4$	$D1 = -128, -31, -21, -11, 0, 11, 21, 31, 127$ $D2 = -128, -65, -45, -21, 0, 21, 45, 65, 127$	$D1 = 0.725 * 32 * \sigma$ $D2 = 1.0 * 32 * \sigma$
$R = 4/5$	$D1 = -128, -31, -21, -11, 0, 11, 21, 31, 127$ $D2 = -128, -63, -42, -21, 0, 21, 42, 63, 127$	$D1 = 0.5 * 32 * \sigma$ $D2 = 1.0 * 32 * \sigma$
$R = 6/7$	$D1 = -128, -31, -21, -11, 0, 11, 21, 31, 127$ $D2 = -128, -63, -41, -20, 0, 20, 41, 63, 127$	$D1 = 0.5 * 32 * \sigma$ $D2 = 1.0 * 32 * \sigma$

Table 4.3: Parameters of fixed step size and adaptive step size (3-bit quantization)

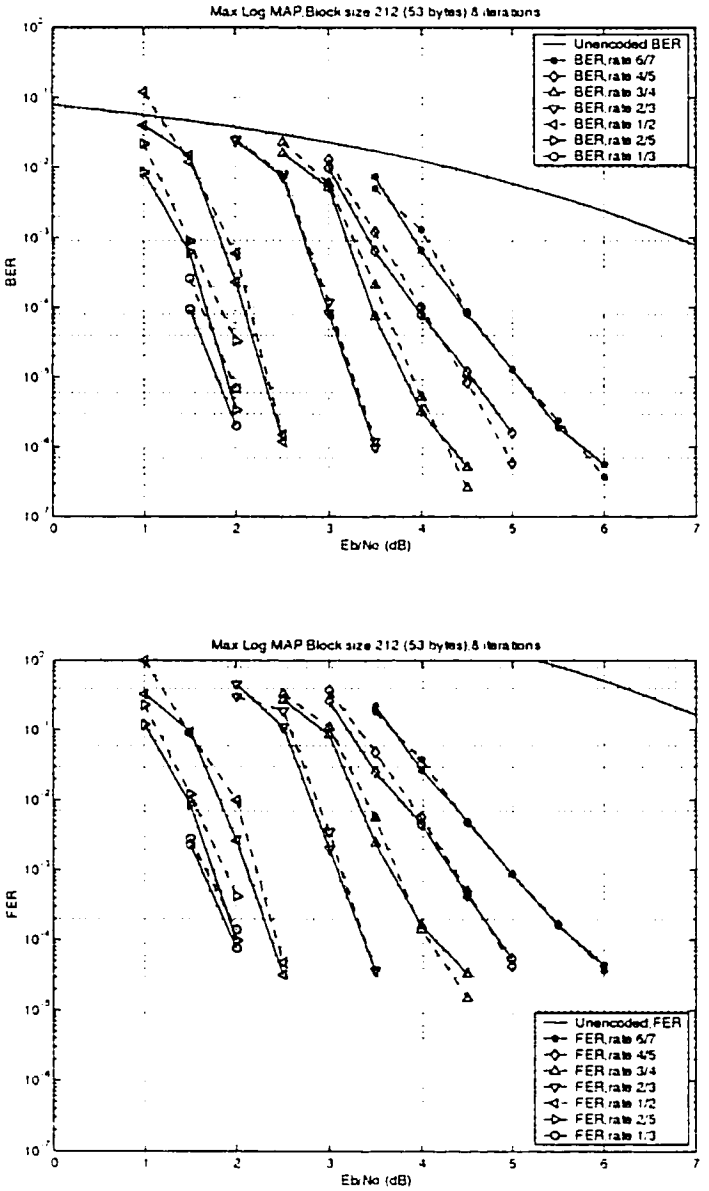


Figure 4.7: 4-bit quantization with adaptive decision level. The Solid lines are unquantized and the dashed lines are quantized with 4-bit.

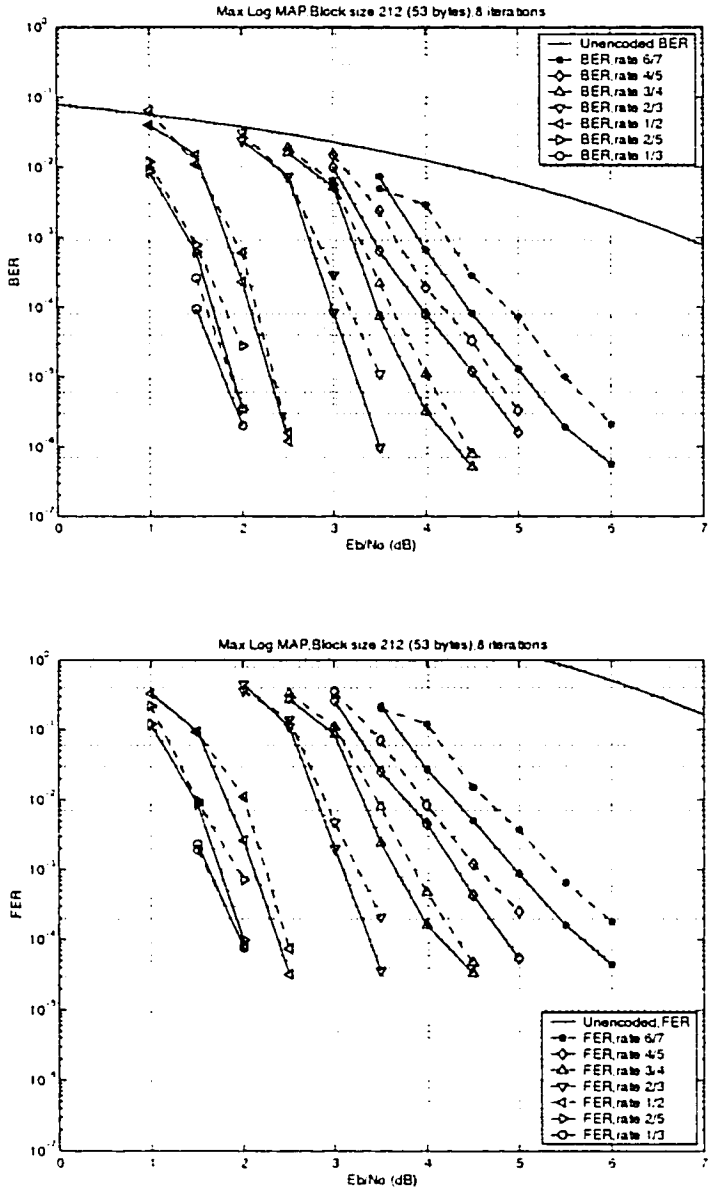


Figure 4.8: 4-bit quantization with fixed decision level. The Solid lines are unquantized and the dashed lines are quantized with 4-bit.

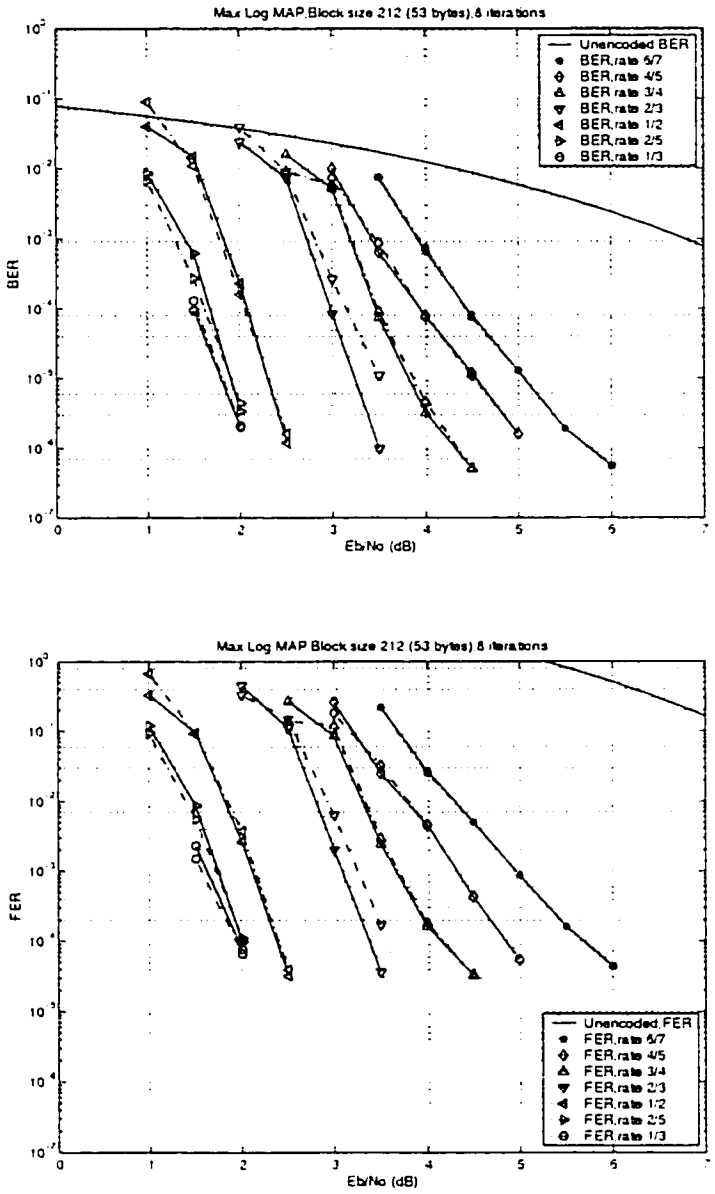


Figure 4.9: With correct coefficient: two level look-up table. The dashed lines are the performances with correction coefficient.

Chapter 5

Triple-binary codes and 8PSK modulation

Power and bandwidth are limited resources in modern communications systems. Efficient exploitation of these resources will invariably involve an increase in the complexity of a communication system. If the signal set dimensionality per information bit is unchanged, the spectral efficiency remains unchanged, *i.e.*, there is no bandwidth expansion. Even though double-binary Circular Recursive Systematic Convolutional (CRSC) codes have an excellent performance, they are limited by the QPSK modulation to a bandwidth efficiency of less than 2bits/s/Hz, as well as the puncturing, which there are less redundancy parity bits to be punctured to achieve higher bandwidth efficiency, for example, there are only 1/12 parity bits left in each encoder for code rate 6/7. In this chapter, the triple-binary Circular Recursive Systematic Convolutional (CRSC) code, 8-ary triple-binary turbo code, is designed for coding with high spectral efficiency using 8PSK modulation and Gray mapping, symbol puncturing and symbol interleaving. The turbo encoder design involves the component encoder design, the interleaver design and the puncturer design. Certain special conditions need to be met at the encoder and the iterative decoder need to be adapted to symbol-by-symbol decoding.

5.1 System Model

This triple-binary code still has the features of the CRSC codes, which avoids the degradation of the spectral efficiency. The system model is similar to DVB-RCS standard and exhibited in Figure 5.1. The data sequence to be encoded, made up of k information bits, feeds the CRSC encoder twice: first, in the natural order of the data (switch in position 1), and next in an interleaved order, given by time permutation function Π (switch in position 2).

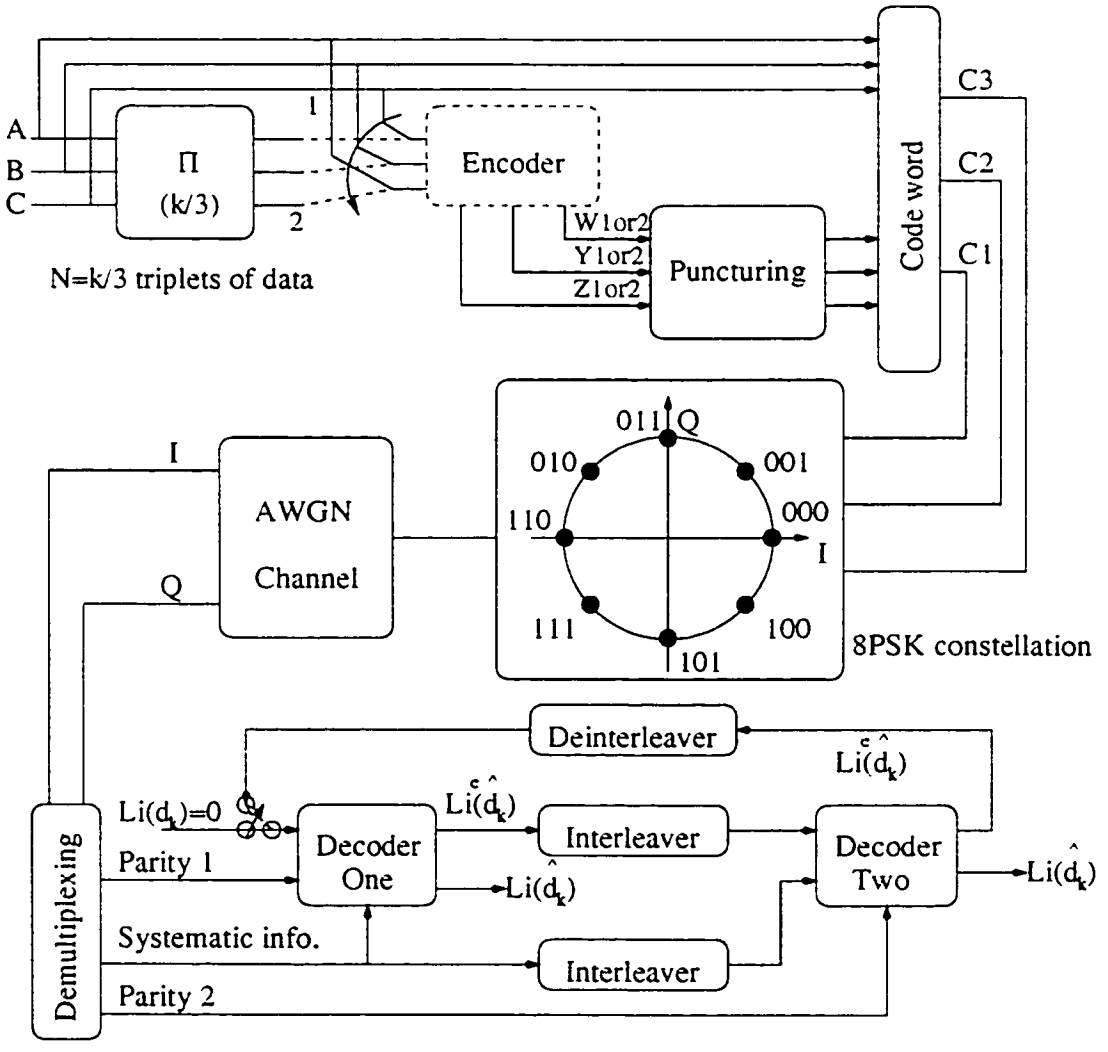


Figure 5.1: System model of triple-binary code combined 8PSK modulation

5.1.1 Constituent encoder

What is crucial to the practical suitability of turbo codes is the fact that they can be decoded iteratively with good performance. However, the resulting iterative decoder is restricted by the signal mapper of 8PSK constellation and certain special conditions need to be met at the encoder. The encoder is fed by blocks of k bits or N triplets ($k = 3 * N$ *bits*). The information length k is a multiple of 24 since 3 and 8 are mutually prime. It uses a 8-ary triple-binary Circular Recursive Systematic Convolutional (CRSC) code, whose three parallel input bits, three parallel systematic bits and three parallel parity bits make it convenient for the puncturer to increase coding rate, and for the 8PSK constellation mapping, therefore, the iterative decoding without any numerical problem. One triple-binary CRSC code designed is depicted in Figure 5.2.

Generator G1:

$g_0(D) = 1 + D + D^4$ 31, *for the feedback branch*
$g_1(D) = 1 + D^2 + D^3 + D^4$ 27, *for the Z parity bits*
$g_2(D) = 1 + D^3 + D^4$ 23, *for the Y parity bits*
$g_3(D) = 1 + D^4$ 21, *for the W parity bits*

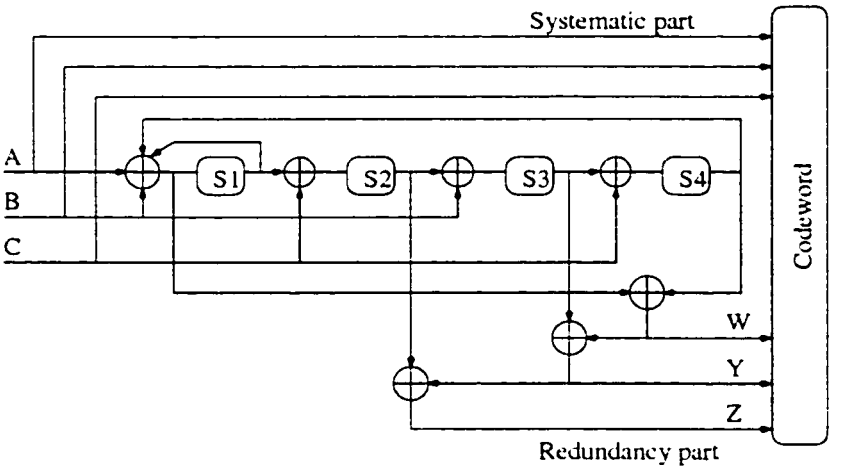


Figure 5.2: Encoder structure with generator G1

Since the constituent codes with small constraint lengths ensure convergence at very low signal to noise ratios and the correlation effects are minimized, the solution chosen uses memory $v = 4$ component codes for efficient convolutional turbo coding.

Furthermore, reasonable constraint lengths make hardware implementation on a single integrated circuit chip possible since the material complexity of the decoder grows exponentially with the code memory.

Not only the performance of any binary code but also that of non-binary code is dominated by its free distance d_{free} and its multiplicities. To achieve a good performance, the component encoders should have large effective free distance and small multiplicities. To analyze triple-binary code performance in this region, an analytical approach can be applied, based on the knowledge of the code-free distance d_{free}. For a triple-binary code with free distance d_{free}, we will denote by N_{free} its multiplicity (the number of codewords with weight d_{free}), and by w_{free} its information bit multiplicity (defined as the sum of the Hamming weights of N_{free} information frames generating the codewords with weight d_{free}).

Generator G2 :

$$g_0(D) = 1 + D^3 + D^4 \quad 23, \textit{ for the feedback branch}$$
$$g_1(D) = 1 + D + D^2 + D^4 \quad 35, \textit{ for the Z parity bits}$$
$$g_2(D) = 1 + D + D^4 \quad 31, \textit{ for the Y parity bits}$$
$$g_3(D) = 1 + D^4 \quad 21, \textit{ for the W parity bits}$$

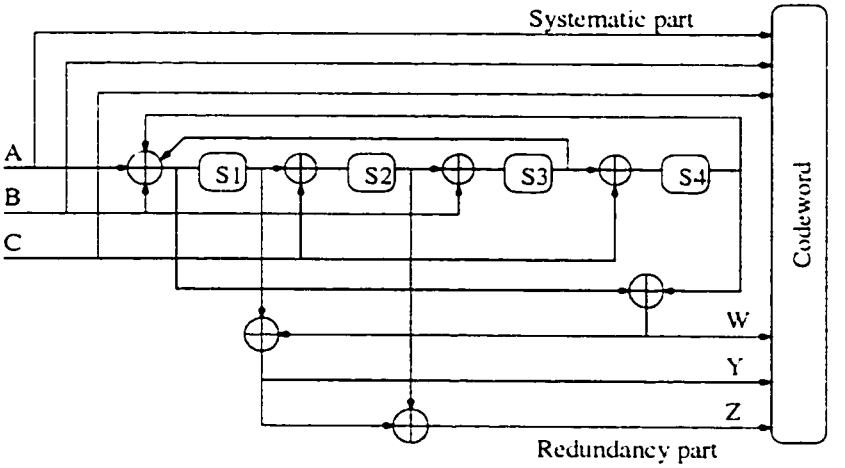


Figure 5.3: Encoder structure with generator G2

Encoder structure with generator G1 and G2 have the same free distance d_{free}, but their multiplicity N_{free} are different: the N_{free} of encoder structure with generator G1 is 5; the N_{free} of encoder structure with generator G2 is 3. The simulation

result (Figure 5.4) show that the different performance of these two structures.

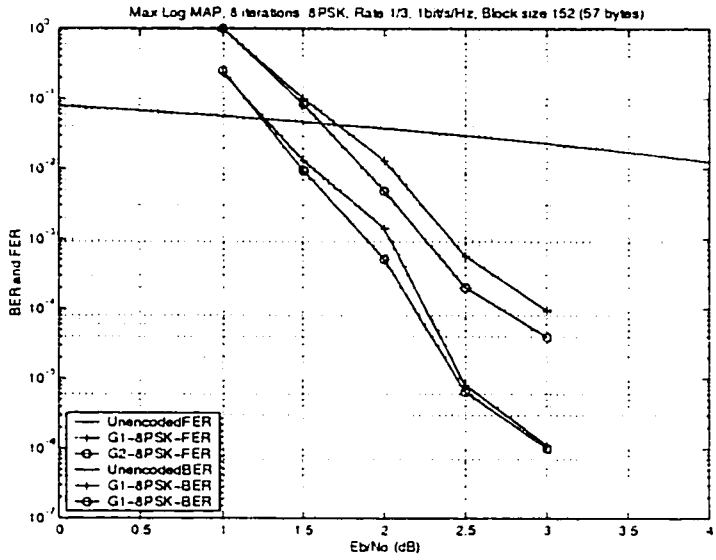


Figure 5.4: Performance for block size N =152 with different encoder structure.

5.1.2 Circular state

For Circular coding, the encoder retrieves the initial state at the end of the encoding operation so that data encoding may be represented by a circular trellis. The value of the circulation state depends on the contents of the sequence to encode and determining S_c requires a pre-encoding operation: first, the encoder is initialized in the "all zero" state. The data sequence is encoded once, leading to a final state S_N^0. Then, S_c value is calculated from expression $S_c = \left\langle I + G^N \right\rangle^{-1} \cdot S_N^0$ as discussed in chapter 2, and matrix G are given by:

$$G1 = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} : \qquad G2 = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

The state of the encoder is denoted $S\,(0 \leq S \leq 15)$ with $S = 8 \cdot s_1 + 4 \cdot s_2 + 2 \cdot s_3 + s_4$. According to the length N of the sequence, use Table 5.1 to find S_c.

In this thesis, we just investigate three different frame sizes, $N = 152$ (53 *bytes*), $N = 752$ (282 *bytes*) and $N = 224$ (84 *bytes*).

$N\ mod.15 \rightarrow$ S_N^0	(G_1) 2	(G_2) 2	(G_2) 14
0	0	0	0
1	11	11	15
2	7	13	1
3	12	6	14
4	15	10	3
5	4	1	12
6	8	7	2
7	3	12	13
8	5	5	7
9	14	14	8
10	2	8	6
11	9	3	9
12	10	15	4
13	1	4	11
14	13	2	5
15	6	9	10

Table 5.1: Circulation state correspondence table for triple-binary codes

To perform a complete encoding operation of the data sequence, two circulation states have to be determined, one for each component encoder, and the sequence has to be encoded four times instead of twice as described in Chapter 3.

5.1.3 Description of the turbo code permutation

For double-binary CRSC codes, non-uniform interleaving can be introduced that local disorder into the data couples, for example, (A, B) become (B, A), or (B, A+B), etc. However, for triple-binary CRSC codes, non-uniformity makes the iterative decoding very difficult and complex because of 8PSK constellation mapping, therefore, to achieve the different permutations that govern the performance of parallel concatenation of convolutional codes (PCCC) at low error rates, use generic equations

with only a restricted number of parameters.

Let N be the number of data triplets in each block at the encoder input (each block contains $3N$ data bits). Set the permutation parameters p_0, p_1, p_2 and p_3. $j = 0, \ldots, N-1$.

- If j mod. 4 = 0, then $p = 0$;
- If j mod. 4 = 1, then $p = N/2 + p_1$;
- If j mod. 4 = 2, then $p = p_2$;
- If j mod. 4 = 3, then $p = N/2 + p_3$;

$$i = (p_0 * j + p + 1) \ mod. \ N \tag{5.1}$$

The interleaving relations satisfy the odd/even rule (*i.e.*, when j is even, i is odd and vice-versa) that enables the puncturing patterns to be identical for both encoders.

Frame size in triples	$P_1 = \{p_0, p_1, p_2, p_3\}$	$P_2 = \{p_0, p_1, p_2, p_3\}$
$N = 152$ (57 *bytes*)	{11,78,4,2}	
$N = 224$ (84 *bytes*)	{23,114,8,118}	{19,108,4,116}
$N = 752$ (282 *bytes*)	{19,2,16,6}	{19,376,224,600}

Table 5.2: Triple-binary code permutation parameters

Simulation results (See Figure 5.5) show the effect of different permutation parameters. The performance of P_1 parameter is better than that of P_2.

5.1.4 Puncturing map, order of transmission and mapping to 8PSK constellation

Two code rates are defined for the triple-binary CRSC turbo mode: $R = 1/3$, *and* $2/3$. For rate 1/3, the systematic bits and all encoded bits are transmitted, and rate 2/3 is achieved through selectively deleting the parity bits (puncturing). The puncturing patterns of Table 5.3 are applied. This pattern is identical for both codes C_1 and C_2 (deletion is always done in triplets).

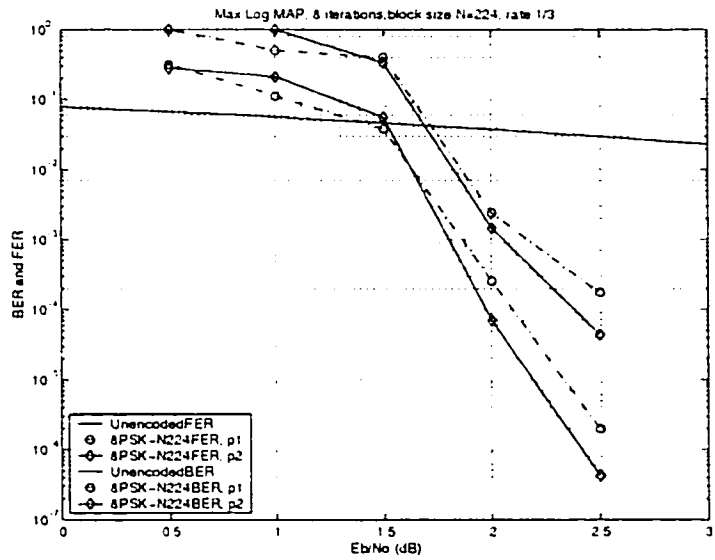


Figure 5.5: Performance of frame size N=224 (84 bytes) with different permutation parameters. Encoder structure with generator G2

$$\text{Code rate } 1/3 \quad \begin{matrix} Z \\ Y \\ W \end{matrix} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\text{Code rate } 2/3 \quad \begin{matrix} Z \\ Y \\ W \end{matrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Table 5.3: Puncturing patterns (compared with unpunctured patterns) for triple-binary CRSC codes. "1" = keep

The order of transmission is in the natural order: all triplets (A, B, C) are transmitted first, followed by all triplets (W_1, Y_1, Z_1) that remain after puncturing and then all triplets (W_2, Y_2, Z_2) that remain after puncturing (see Figure 5.6).

Each triplet is mapped into one 8PSK constellation point as shown in Figure 5.7. In Figure 5.6, the columns with the systematic symbols and the columns with parity symbols are each mapped into one 8PSK constellation point, *i.e.*, systematic symbols (C, B, A), or parity symbols (Z1, Y1, W1) and (Z2, Y2, W2) correspond to (C3, C2, C1). The signal shall be modulated using 8PSK, with baseband shaping. The output (C3, C2, C1) of the channel coding shall be mapped into the I channel

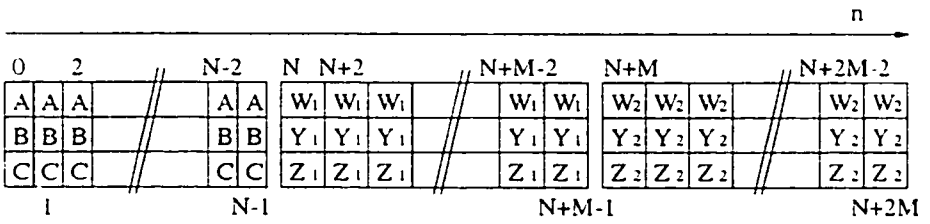
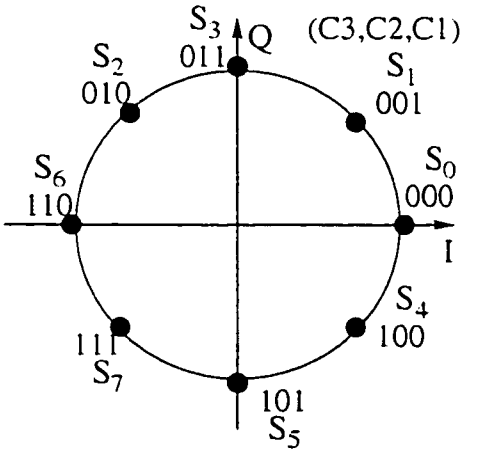


Figure 5.6: Encoded blocks (natural order). M=N, unpunctured; M=1/4N, punctured.

and the Q channel as shown on the bottom of Figure 5.7.



(C3, C2, C1) = ((C, Z1, or Z2), (B, Y1, or Y2), (A, W1, or W2)) = (I, Q)

$$
\begin{array}{llll}
S_0 = (000) \rightarrow (1,0) & S_4 = (100) \rightarrow (\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}) \\
S_1 = (001) \rightarrow (\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}) & S_5 = (101) \rightarrow (0,-1) \\
S_2 = (010) \rightarrow (-\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}) & S_6 = (110) \rightarrow (-1,0) \\
S_3 = (011) \rightarrow (0,1) & S_7 = (111) \rightarrow (-\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2})
\end{array}
$$

Figure 5.7: Gray mapping for 8PSK constellation

Furthermore, we can rotate the mapping to Figure 5.8 and still get the similar performance even though the encoder generator is different (See Figure 5.9).

5.2 Iterative decoding procedure

In DVB-RCS standard, each couple is mapped into one QPSK constellation point and every bit is mapped on the I channel and Q channel, respectively. Actually, the

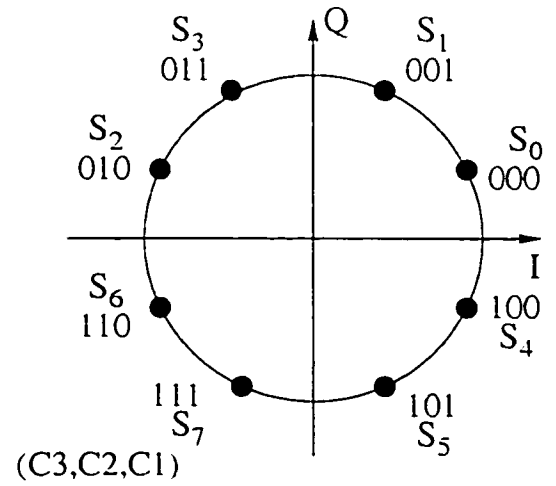


Figure 5.8: Rotation of gray mapping for 8PSK constellation

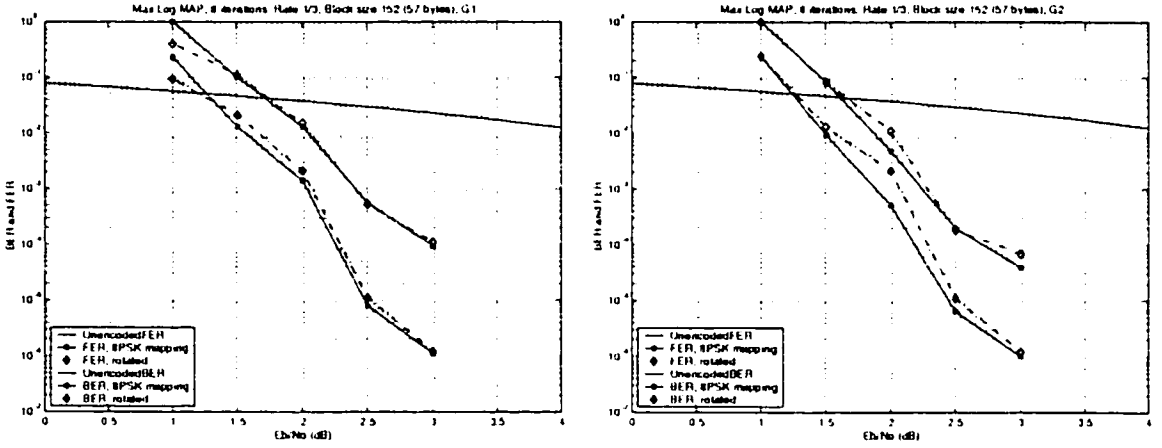


Figure 5.9: Different 8PSK constellation point with different encoder generator

signal shall be sent without modification to the QPSK bit mapper (see Figure 3.7) and the iterative decoding is bit-by-bit. For the restriction of the numerical problem of iterative decoding, the interleaving and puncturing are symbol-by-symbol according to 8PSK constellation mapping, therefore, the decoding algorithm is derived as symbol-by-symbol Max-Log-MAP algorithm.

The trellis of the triple-binary feedback convolutional encoder that we use, has 16 states and each node has 8 symbol inputs and 8 symbol outputs. Let S_k be the encoder state at time k. The symbol d_k is associated with the transition from time

$k-1$ to time k. The trellis states at stage $k-1$ and at stage k are indexed by the integers S_{k-1} and S_k, respectively.

$$\begin{aligned} L_i(d_k) &= \ln \frac{P[d_k = i|Observation]}{P[d_k = 0|Observation]} \quad for \quad i = 1,2,3,4,5,6,7 \\ &= \ln \frac{\sum_{d_k=i}^{(S_{k-1},S_k)} p(S_{k-1}, S_k, y_k)}{\sum_{d_k=0}^{(S_{k-1},S_k)} p(S_{k-1}, S_k, y_k)} \end{aligned} \tag{5.2}$$

The index pair S_{k-1} and S_k determines the information symbol d_k and the coded symbol x_k, where d_k is in GF(2^3) with elements {0,1,2,3,4,5,6,7} from time $k-1$ to time k. The sum of the joint probabilities $p(S_{k-1}, S_k, y_k)$ in the numerator or in the denominator of (5.2) is taken over all existing transitions state S_{k-1} to state S_k labeled with the information symbols $d_k = 0,1,2,3,4,5,6,7$ (that is, $d_k = 000, 001, 010, 011, 100, 101, 110, 111$. We use decimal notations instead of binary for simplicity, as well as $x_k = 000, 001, 010, 011, 100, 101, 110, 111$). Assuming a memoryless transmission channel, the joint probability $p(S_{k-1}, S_k, y_k)$ can be written as the product of three independent probabilities

$$\begin{aligned} p(S_{k-1}, S_k, y_k) &= p(S_{k-1}, y_{j<k}) \cdot p(S_k, y_k|S_{k-1}) \cdot p(y_{j>k}|S_k) \\ &= \underbrace{p(S_{k-1}, y_{j<k})} \cdot \underbrace{P(S_k|S_{k-1}) \cdot p(y_k|S_{k-1}, S_k)} \cdot \underbrace{p(y_{j>k}|S_k)} \\ &= \alpha_{k-1}(S_{k-1}) \cdot \gamma_k^t(S_{k-1}, S_k) \cdot \beta_k(S_k) \end{aligned} \tag{5.3}$$

Here $y_{j<k}$ denotes the sequence of received symbols y_j from the beginning of the trellis up to time $k-1$ and $y_{j>k}$ is the corresponding sequence from time $k+1$ up to the end of the trellis. The forward recursion of the MAP algorithm yields

$$\alpha_k(S_k) = \sum_{k-1} \sum_{t=0}^{7} \gamma_k^t(S_{k-1}, S_k) \cdot \alpha_{k-1}(S_{k-1}) \tag{5.4}$$

The backward recursion yields

$$\beta_{k-1}(S_{k-1}) = \sum_{S_k} \sum_{t=0}^{7} \gamma_k^t(S_{k-1}, S_k) \cdot \beta_k(S_k) \tag{5.5}$$

Whenever a transition between S_{k-1} and S_k exists the branch transition probabilities are given by

$$\gamma_k^i(S_{k-1}, S_k) = P(S_k|S_{k-1}) \cdot p(y_k|S_{k-1}, S_k) = p(y_k|d_k) \cdot P(d_k) \quad for\ i = 0, 1, 2, 3, 4, 5, 6, 7 \tag{5.6}$$

The distribution of the received parity and systematic symbols are given by

$$\begin{aligned} p[y_k|d_k = i] &= p[y_k^s|x_k^s(i)] \cdot p[y_k^p|x_k^p(i, S_{k-1}, S_k)] \\ &= \frac{1}{\pi \cdot N_0} exp\{-\frac{E_s}{N_0}[(y_k^{s,I} - x_k^{s,I}(i))^2 + (y_k^{s,Q} - x_k^{s,Q}(i))^2]\} \\ &\quad \cdot \frac{1}{\pi \cdot N_0} exp\{-\frac{E_s}{N_0}[(y_k^{p,I} - x_k^{p,I}(i, S_{k-1}, S_k))^2 + (y_k^{p,Q} - x_k^{p,Q}(i, S_{k-1}, S_k))^2]\} \\ &= B_k \cdot exp\{\frac{1}{2} L_c \cdot [y_k^{s,I} \cdot x_k^{s,I}(i) + y_k^{s,Q} \cdot x_k^{s,Q}(i)]\} \\ &\quad \cdot exp\{\frac{1}{2} L_c \cdot [y_k^{p,I} \cdot x_k^{p,I}(i, S_{k-1}, S_k) + y_k^{p,Q} \cdot x_k^{p,Q}(i, S_{k-1}, S_k)]\} \end{aligned} \tag{5.7}$$

where y_k^s, y_k^p represent the received systematic and parity symbols, and $y_k^{s,I}$, $y_k^{s,Q}$, $y_k^{p,I}$, $y_k^{p,Q}$ represent the received symbol values that are transmitted through the I and Q channels, respectively; $x_k^s(i)$, $x_k^p(i, S_{k-1}, S_k)$ represent the systematic and parity symbols for $i = 0, 1, 2, 3, 4, 5, 6, 7$, and $x_k^{s,I}(i)$, $x_k^{s,Q}(i)$, $x_k^{p,I}(i, S_{k-1}, S_k)$, $x_k^{p,Q}(i, S_{k-1}, S_k)$ represent the symbols of codeword mapped to 8PSK constellation, respectively.

Here,

$$\begin{aligned} B_k &= (\frac{1}{\pi \cdot N_0})^2 exp\{-\frac{E_s}{N_0}[(y_k^{s,I})^2 + (x_k^{s,I}(i))^2 + (y_k^{s,Q})^2 + (x_k^{s,Q}(i))^2 \\ &\quad + (y_k^{p,I})^2 + (x_k^{p,I}(i, S_{k-1}, S_k))^2 + (y_k^{p,Q})^2 + (x_k^{p,Q}(i, S_{k-1}, S_k))^2]\} \end{aligned} \tag{5.8}$$

5.2.1 Symbol-by-symbol Max-Log-MAP algorithm for 8-ary codes

Symbol-by-symbol Max-Log-MAP algorithm is derived for triple-binary codes with higher order modulation 8PSK using Gray mapping. First, find the logarithm of

the branch metrics as

$$\begin{aligned}
\ln \gamma_k^i(S_{k-1}, S_k) &= \overline{\gamma_k^i}(S_{k-1}, S_k) \\
&= \ln P(y_k|d_k) \cdot P(d_k) \\
&= \frac{1}{2} L_c \cdot [y_k^{s,I} \cdot x_k^{s,I}(i) + y_k^{s,Q} \cdot x_k^{s,Q}(i)] + \ln P(d_k) + K \\
&\quad + \frac{1}{2} L_c \cdot [y_k^{p,I} \cdot x_k^{p,I}(i, S_{k-1}, S_k) + y_k^{p,Q} \cdot x_k^{p,Q}(i, S_{k-1}, S_k)] \quad (5.9)
\end{aligned}$$

Where constant K include the constant and common terms that are cancelled in comparisons at the later stages.

Next, compute $\alpha_k(S_k)$ and $\beta_k(S_k)$ as

$$\begin{aligned}
\ln \alpha_k(S_k) &= \overline{\alpha_k}(S_k) \\
&= \ln \sum_{S_{k-1}} \sum_{i=0}^{\bar{\imath}} \gamma_k^i(S_{k-1}, S_k) \cdot \alpha_{k-1}(S_{k-1}) \\
&= \ln \sum_{S_{k-1}} \sum_{i=0}^{\bar{\imath}} e^{\overline{\gamma_k^i}(S_{k-1}, S_k)} \cdot e^{\overline{\alpha_{k-1}}(S_{k-1})} \\
&= \ln \{ \sum_{S_{k-1}} \sum_{i=0}^{\bar{\imath}} [e^{\overline{\gamma_k^i}(S_{k-1}, S_k) + \overline{\alpha_{k-1}}(S_{k-1})}] \} \\
&\approx \max_{S_{k-1}, S_k, i} [\overline{\gamma_k^i}(S_{k-1}, S_k) + \overline{\alpha_{k-1}}(S_{k-1})] \quad (5.10)
\end{aligned}$$

Similarly

$$\begin{aligned}
\ln \beta_{k-1}(S_k) &= \overline{\beta_{k-1}}(S_k) \\
&= \ln \sum_{S_k} \sum_{i=0}^{\bar{\imath}} \gamma_k^i(S_{k-1}, S_k) \cdot \beta_k(S_k) \\
&= \ln \sum_{S_k} \sum_{i=0}^{\bar{\imath}} e^{\overline{\gamma_k^i}(S_{k-1}, S_k)} \cdot e^{\overline{\beta_k}(S_k)} \\
&= \ln \{ \sum_{S_k} \sum_{i=0}^{\bar{\imath}} [e^{\overline{\gamma_k^i}(S_{k-1}, S_k) + \overline{\beta_k}(S_k)}] \} \\
&\approx \max_{S_k, S_{k-1}, i} [\overline{\gamma_k^i}(S_{k-1}, S_k) + \overline{\beta_k}(S_k)] \quad (5.11)
\end{aligned}$$

For iterative decoding of circular trellis, Tail-biting is

$$\begin{aligned} \overline{\alpha_0}(S_0) &= \overline{\alpha_N}(S_N) \quad for\ \forall S_0 \\ \overline{\beta_N}(S_N) &= \overline{\beta_0}(S_0) \quad for\ \forall s_N \end{aligned} \tag{5.12}$$

The log-likelihood ratios are represented by

$$\begin{aligned} L_i(d_k) &= \ln \frac{\sum_{d_k=i}^{(S_{k-1},S_k)} \gamma_k^i(S_{k-1},S_k)\cdot\alpha_{k-1}(S_{k-1})\cdot\beta_k(S_k)}{\sum_{d_k=0}^{(S_{k-1},S_k)} \gamma_k^0(S_{k-1},S_k)\cdot\alpha_{k-1}(S_{k-1})\cdot\beta_k(S_k)} \quad for\ \ i=1,2,3,4,5,6,7 \\ &= \ln \frac{\sum_{d_k=i}^{(S_{k-1},S_k)} e^{\overline{\gamma_k^i}(S_{k-1},S_k)}\cdot e^{\overline{\alpha_{k-1}}(S_{k-1})}\cdot e^{\overline{\beta_k}(S_k)}}{\sum_{d_k=0}^{(S_{k-1},S_k)} e^{\overline{\gamma_k^0}(S_{k-1},S_k)}\cdot e^{\overline{\alpha_{k-1}}(S_{k-1})}\cdot e^{\overline{\beta_k}(S_k)}} \end{aligned} \tag{5.13}$$

Therefore, to compute the log-likelihood ratios, we follow the equation (5.13) and take max(·) function as

$$\begin{aligned} L_i(\hat{d}_k) \approx & \max_{(s_{k-1},s_k)}[\overline{\gamma_k^i}(S_{k-1},S_k)+\overline{\alpha_{k-1}}(S_{k-1})+\overline{\beta_k}(S_k)] \quad for\ i=1,2,3,4,5,6,7 \\ & -\max_{(s_{k-1},s_k)}[\overline{\gamma_k^0}(S_{k-1},S_k)+\overline{\alpha_{k-1}}(S_{k-1})+\overline{\beta_k}(S_k)] \end{aligned} \tag{5.14}$$

Moreover, to separate the Log-likelihood ratios into intrinsic, systematic and extrinsic information, define:

$$\begin{aligned} \gamma_k^{i(e)}(S_{k-1},S_k) &= exp\{\frac{1}{2}L_c\cdot[y_k^{p,I}\cdot x_k^{p,I}(i,S_{k-1},S_k) \\ &\quad + y_k^{p,Q}\cdot x_k^{p,Q}(i,S_{k-1},S_k)]\}\cdot A_k \end{aligned} \tag{5.15}$$

Here, the constant

$$A_k = \frac{1}{\pi\cdot N_0} exp\{-\frac{E_s}{N_0}[(y_k^{p,I})^2+(x_k^{p,I}(i,S_{k-1},S_k))^2+(y_k^{p,Q})^2+(x_k^{p,Q}(i,S_{k-1},S_k))^2]\} \tag{5.16}$$

Hence, the logarithm of the branch transition operation reduces to the expression with

$$\begin{aligned} \ln\gamma_k^{i(e)}(S_{k-1},S_k) &= \overline{\gamma_k^{i(e)}}(S_{k-1},S_k) \\ &= \frac{1}{2}L_c\cdot[y_k^{p,I}\cdot x_k^{p,I}(i,S_{k-1},S_k)+y_k^{p,I}\cdot x_k^{p,Q}(i,S_{k-1},S_k)]+K' \end{aligned}$$

where constant K' includes the constant and common terms that are cancelled in comparisons at later stages.

In another way, find the Log-likelihood ratios as

$$\begin{aligned} L_i(\hat{d}_k) &= \ln \frac{\sum_{d_k=i}^{(S_{k-1},S_k)} \gamma_k^i(S_{k-1},S_k)\cdot\alpha_{k-1}(S_{k-1})\cdot\beta_k(S_k)}{\sum_{d_k=0}^{(S_{k-1},S_k)} \gamma_k^0(S_{k-1},S_k)\cdot\alpha_{k-1}(S_{k-1})\cdot\beta_k(S_k)} \quad for\ i=1,2,3,4,5,6,7 \\ &= \ln \frac{exp\{\frac{1}{2}L_c\cdot[y_k^{s,I}\cdot x_k^{s,I}(i)+y_k^{s,Q}\cdot x_k^{s,Q}(i)]\}\cdot P[d_k=i]}{exp\{\frac{1}{2}L_c\cdot[y_k^{s,I}\cdot x_k^{s,I}(0)+y_k^{s,Q}\cdot x_k^{s,Q}(0)]\}\cdot P[d_k=0]} \\ &\quad + \ln \frac{\sum_{d_k=i}^{(S_{k-1},S_k)} \gamma_k^{i(e)}(S_{k-1},S_k)\cdot\alpha_{k-1}(S_{k-1})\cdot\beta_k(S_k)}{\sum_{d_k=0}^{(S_{k-1},S_k)} \gamma_k^{0(e)}(S_{k-1},S_k)\cdot\alpha_{k-1}(S_{k-1})\cdot\beta_k(S_k)} \\ &= \{\frac{1}{2}L_c\cdot[y_k^{s,I}\cdot x_k^{s,I}(i)+y_k^{s,Q}\cdot x_k^{s,Q}(i)] - \frac{1}{2}L_c\cdot[y_k^{s,I}\cdot x_k^{s,I}(0)+y_k^{s,Q}\cdot x_k^{s,Q}(0)]\} \\ &\quad \underbrace{+\ln\frac{P[d_k=i]}{P[d_k=0]}}_{L_i(d_k)} + \underbrace{\ln \frac{\sum_{d_k=i}^{(S_{k-1},S_k)} \gamma_k^{i(e)}(S_{k-1},S_k)\cdot\alpha_{k-1}(S_{k-1})\cdot\beta_k(S_k)}{\sum_{d_k=0}^{(S_{k-1},S_k)} \gamma_k^{0(e)}(S_{k-1},S_k)\cdot\alpha_{k-1}(S_{k-1})\cdot\beta_k(S_k)}}_{L_i^e(\hat{d}_k)} \end{aligned} \tag{5.17}$$

So, the extrinsic information is

$$\begin{aligned} L_i^e(\hat{d}_k) &= L_i(\hat{d}_k) - \frac{1}{2}L_c\cdot[y_k^{s,I}\cdot x_k^{s,I}(i)+y_k^{s,Q}\cdot x_k^{s,Q}(i)] \\ &\quad + \frac{1}{2}L_c\cdot[y_k^{s,I}\cdot x_k^{s,I}(0)+y_k^{s,Q}\cdot x_k^{s,Q}(0)] - \ln\frac{P[d_k=i]}{P[d_k=0]} \end{aligned} \tag{5.18}$$

The computation of the symbol probabilities for the next Decoder is as follows,

$$L_i(d_k) = L_i^e(\hat{d}_k) = \ln\frac{P[d_k=i]}{P[d_k=0]} \quad for\ i=1,2,3,4,5,6,7 \quad from\ previouse\ Decoder \tag{5.19}$$

Since

$$\begin{cases} L_0^e(\hat{d}_k) &= \ln\frac{P[d_k=0]}{P[d_k=0]} = \ln 1 = 0 \\ L_i^e(\hat{d}_k) &= \ln\frac{P[d_k=i]}{P[d_k=0]} \end{cases} \tag{5.20}$$

then

$$P[d_k=i] = e^{L_i^e(\hat{d}_k)}\cdot P[d_k=0] \tag{5.21}$$

and

$$\sum_{i=0}^{7} P[d_k = i] = 1 \tag{5.22}$$

Hence,

$$\begin{cases} P[d_k = 0] &= \frac{1}{1+\sum_{i=1}^{7} e^{L_i^e(\hat{d}_k)}} \\ P[d_k = i] &= \frac{e^{L_i^e(\hat{d}_k)}}{1+\sum_{i=1}^{7} e^{L_i^e(\hat{d}_k)}} \end{cases} \tag{5.23}$$

Using max(·) Function

$$\begin{cases} \ln P[d_k = 0] &= -\max[0, L_i^e(\hat{d}_k)] \\ \ln P[d_k = i] &= L_i^e(\hat{d}_k) - \ln P[d_k = 0] \end{cases} \quad for\ i = 1, 2, 3, 4, 5, 6, 7 \tag{5.24}$$

5.2.2 Initialization and Decision of decoding procedure

Assume equally likely information symbols. Then, we do not have any *a priori* information available for the first iteration, and we initialize

$$L_i(d_k = i) = 0 \quad for\ i = 0, 1, 2, 3, 4, 5, 6, 7 \tag{5.25}$$

according to Equation (5.23), we have

$$\begin{cases} P[d_k = 0] &= \frac{1}{1+\sum_{i=1}^{7} e^{L_i^e(\hat{d}_k)}} = \frac{1}{8} \\ P[d_k = i] &= \frac{e^{L_i^e(\hat{d}_k)}}{1+\sum_{i=1}^{7} e^{L_i^e(\hat{d}_k)}} = \frac{1}{8} \end{cases} \quad for\ i = 1, 2, 3, 4, 5, 6, 7 \tag{5.26}$$

Using max(·) Function:

$$\begin{cases} \ln P[d_k = 0] &= -\max[0, L_i^e(\hat{d}_k)] = 0 \\ \ln P[d_k = i] &= L_i^e(\hat{d}_k) - \ln P[d_k = 0] = 0 \end{cases} \quad for\ i = 1, 2, 3, 4, 5, 6, 7 \tag{5.27}$$

Similarly, because of equal-likelihood assumption for all symbols, we have

$$\begin{aligned} \alpha_0(s_0) &= 1 \quad for\ \forall s_0 \\ \beta_N(s_N) &= 1 \quad for\ \forall s_N \end{aligned} \tag{5.28}$$

take logarithm

$$\begin{aligned} \overline{\alpha_0}(S_0) &= \ln \alpha_0(s_0) = 0 \quad for\ \forall s_0 \\ \overline{\beta_N}(S_N) &= \ln \beta_N(s_N) = 0 \quad for\ \forall s_N \end{aligned} \tag{5.29}$$

and initialize $\overline{\gamma_0^t} = 0$.

The reliability value of the channel

$$L_c = 4a \cdot \frac{E_s}{N_0} = 4a \cdot R_c \cdot \frac{E_b}{N_0} \tag{5.30}$$

where R_c is the code rate.

After several decoding iterations, the decisions are made according to:

$$\begin{cases} \hat{d}_k = i & if\ L(\hat{d}_k) = \max(L_i(\hat{d}_k))\ and\ L_i(\hat{d}_k) > 0 \\ & for\ i = 1, 2, 3, 4, 5, 6, 7 \\ \hat{d}_k = 0 & else \end{cases} \tag{5.31}$$

where

$$L(\hat{d}_k) = \max(L_i(\hat{d}_k)) \quad for\ i = 1, 2, 3, 4, 5, 6, 7 \tag{5.32}$$

5.3 Simulation results

In this thesis, three different frame sizes are investigated and Figure 5.10 shows the good performance. The curves on the left of Figure 5.10 correspond to a code rate of 1/3, *i.e.*, a bandwidth efficiency of 1bit/s/Hz. The curves on the right correspond to a code rate of 2/3, *i.e.*, a bandwidth efficiency of 2bits/s/Hz.

1. Compared with double-binary CRSC codes in DVB-RCS standard, which use QPSK modulation.

The curves with solid line on the left of Figure 5.11 correspond to the double-binary/QPSK codes, a code rate of 1/2, block size $N = 228$ (57 bytes) and the permutation (interleaving) is uniform, one level, as well non-uniform, two level. The dashed line is triple-binary/8PSK code, a code rate of 1/3 and one level uniform

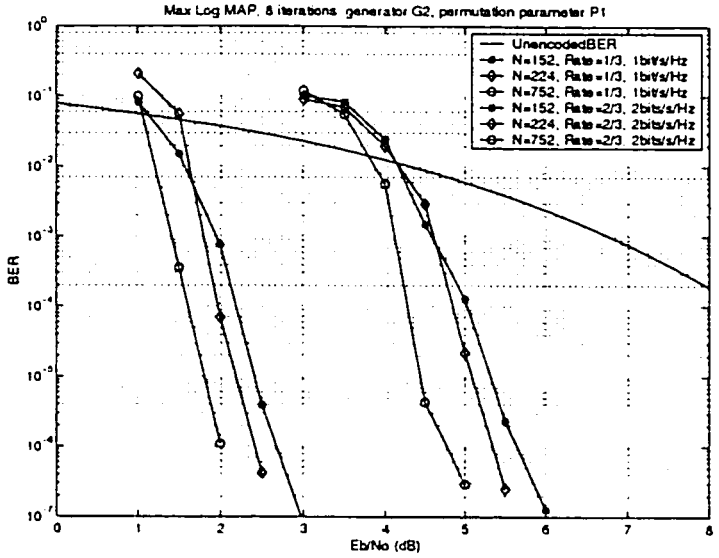


Figure 5.10: Performance of three different frame sizes with different bandwidth efficiency

interleaving. All of them are bandwidth efficiency of 1bit/s/Hz with close performance.

The curve with solid line on the right is the ATM cell (53 bytes) with a code rate of 6/7 and a bandwidth efficiency of 1.7 bits/s/Hz. The dashed line with star is the punctured triple-binary CRSC code with a code rate 2/3 and a bandwidth efficiency of 2bits/s/Hz, $N = 152$ (57 bytes). The performance of triple-binary/8PSK code is better than double-binary/QPSK code at higher signal-to-noise ratios with higher bandwidth efficiency.

2. Compared with TTCM. Calculate the channel capacity.

Trellis-coded modulation (TCM), introduced by Ungerboeck [67][68][69] is a very successful method of reducing power requirements without increase in the requirements on bandwidth. The innovative aspect of TCM is the concept that encoding and modulation should not be treated as separate entities, but rather, as a unique operation. TCM schemes have been applied to telephone, satellite and microwave digital radio channel, where coding gains of the order of 3-6dB are obtained with no loss of bandwidth or data rate.

Turbo trellis coded modulation (TTCM) proposed in [70] is the extension of the turbo codes where the component codes are replaced by Ungerboeck TCM codes in the recursive systematic forms to retain the advantages of both classical turbo codes and TCM codes. However, the triple-binary/8PSK code with frame size, $N = 752$ (282 *bytes*, 2256 *bits*), a code rate of 2/3, the memory $v = 4$, applied Max-Log-MAP algorithm, has a good performance compared with that of TTCM codes introduced in [71]. In their paper, two Ungerboeck-type codes [67] in combination with trellis-coded modulation (TCM) were employed in their recursive systematic form as component codes in an overall structure rather similar to binary turbo codes, thus, the memory is $v = 3$ for convolutional turbo code and $v = 2$ for Ungerboeck TCM. The MAP algorithm was applied and block size $N = 1024$ (256 *bytes*, 2048 *bits*) for code rate 1/3 in this TTCM scheme. The channel capacity is 2 bit/s/Hz at 5.9 dB (E_s/N_0). (See Figure 5.12)

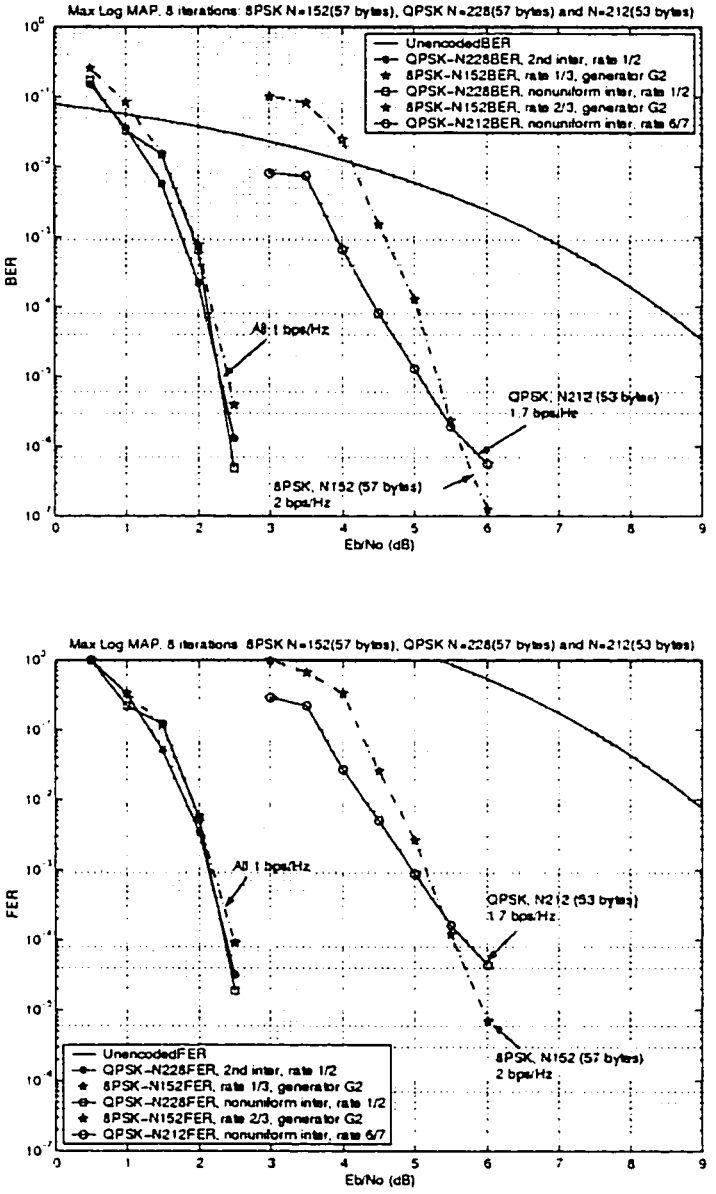


Figure 5.11: Performance compared with double-binary CRSC codes

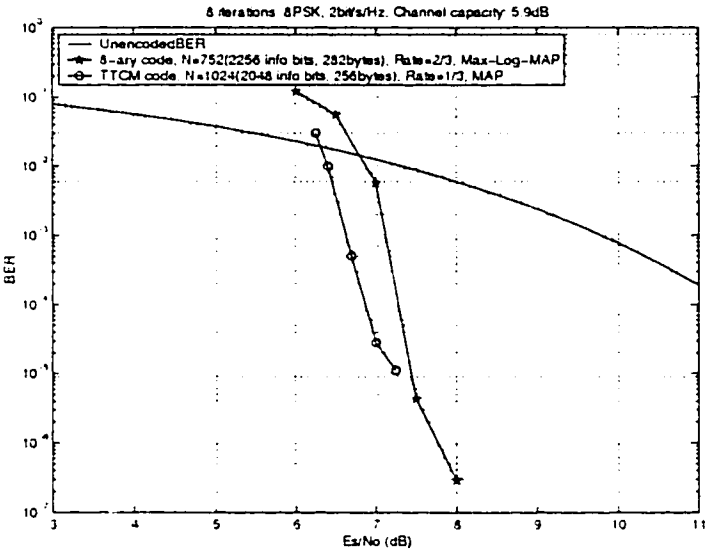


Figure 5.12: Triple-binary CRSC code compared with TTCM. Both for 8PSK modulation and Channel capacity: 2bps/Hz at 5.9 dB (E_s/N_0)

Chapter 6

Conclusions

The subject of this thesis is the design of turbo codes, that is, how to choose and/or design the components in a turbo encoder to get the best possible performance and higher transition speed. However, since turbo codes are decoded using an iterative and suboptimal decoding algorithm, new design criteria arise. In fact, the success of iterative decoding proves to be related to the choice of the components in the turbo code encoder and the modulation scheme used.

The codes investigated in this thesis are constructed via parallel concatenation of two binary recursive convolutional codes (RSC), double-binary circular recursive systematic convolutional (CRSC) codes, and 8-ary triple-binary circular recursive systematic convolutional (CRSC) codes. The parallel concatenation is implemented by interleaving, *i.e.*, re-ordering the information sequence before it is input to the second component encoder. The two most critical parts of a turbo code encoder are, thus the interleaver and the component encoders. Other essential aspects of a turbo encoder are trellis termination and puncturing. Trellis termination is an issue when dealing with data packets where truncation is necessary at some point of the trellis. Puncturing is the process of excluding bits from the outputs of the component encoders, so that the concatenated transmitted sequence is a decimated version of the encoder output.

One of the major breakthroughs related to the introduction of turbo codes is the process of iterative decoding. With iterative decoding, two or more constituent decoders take turns in attempting to decode the received message. In every new attempt, each decoder make use of the outcome from the other decoder's last decoding attempt. Properly formulated, and with sufficient signal-to-noise ratio, the process of iterative decoding is remarkably successful in refining the decoder outputs until the two decoders converge to a joint decision.

In DVB-RCS standard, the substitution of binary codes by double-binary codes has a direct incidence on the erroneous paths in the trellises, which leads to a lowered path error density and reduces correlation effects in the decoding process. This leads to a performance better than that of binary turbo codes for equivalent implementation complexity. Circular coding is a kind of "tail-biting" technique that avoids reducing the code rate and increasing the transmission bandwidth. Non-uniform interleaving is applied to avoid many error patterns due to adopting double-binary CRSC codes. The influence of puncturing and suboptimal decoding algorithm, Max-Log-MAP algorithm, are less significant with double-binary turbo codes than with binary turbo codes. Using double-binary codes, the latency of the decoder is halved. Therefore, double-binary CRSC code could be easily adopted for many applications, for various block sizes and code rates while retaining excellent coding gains.

In practice, the decoding algorithms need to be implemented using fixed point arithmetic, using a Field Programmable Gate Array (FPGA) or fixed point Digital Signal Processor (DSP) chip. In this thesis, we addressed the problem of quantized input to the turbo decoder and the associated fixed point arithmetic. The bit-true model is mandatory for hardware and software implementation with good trade-off between complexity and performance.

Triple-binary CRSC codes inherit most of the advantages of double-binary CRSC codes, however, they are more flexible and efficient for encoding blocks of data. Especially, the 8-ary alphabet of triple-binary turbo codes when combined

with 8PSK makes the puncture possible to achieve high code rate and more than 2bits/s/Hz bandwidth efficiency while double-binary CRSC code and QPSK modulation cannot exceed this restriction. The bitwise interleaver known from classical binary turbo codes is replaced by an interleaver operating on a group of bits and the equations of permutation parameters are chosen the same as the level 2 of non-uniform interleaver. These parameters depend on the block size. The structure of the iterative decoder and the symbol-by-symbol Max-Log-MAP algorithm are derived for non-binary trellises to avoid numerical problems and reduce the decoding complexity because of a set of constraints for the component code, interleaver and 8PSK symbol mapping. The primary study of 8-ary triple-binary CRSC codes show their potential as an alternative for more effective and efficient transmission of data via satellites without increasing the required bandwidth.

6.1 Contributions of this thesis

In summary, the major contributions of this thesis are the followings:

1. A thorough survey of convolutioanl turbo code design, including binary and non-binary codes, and other techniques such as interleaver design, the techniques of trellis truncation, puncturing, and M-PSK modulation concluded.

2. The optimal and suboptimal decoding algorithms, including MAP, Log-MAP, and simplified Max-Log-MAP that is exploited to speed up calculation for both binary and non-binary turbo codes have been studied thoroughly.

3. An quantizer of turbo-decoder is designed in fixed-point model corresponding to the given bit-width of the DSP in order to reduce the complexity and the power consumption of the decoder. The efficient C model codec includes every specification of the double-binary CRSC codes in DVB-RCS standard.

4. A new component encoder, consisting of triple-binary CRSC codes, is designed to achieve more bandwidth efficiency. The interleaver design, the puncturing map design, as well as symbol-by-symbol Max-Log-MAP algorithm for this turbo code was presented.

6.2 Suggestions of future research

Not only double-binary CRSC codes in DVB-RCS standard but also the 8-ary triple-binary CRSC codes in this thesis have achieved an amazing performance, therefore, there are many considerable potentials for the applications in some techniques such as turbo equalization, turbo trellis coded modulation (TTCM) and multiuser detection with turbo feedback.

First, since double-binary CRSC codes are already a mature technology for DVB-RCS standard, it is possible and easy to investigate how they work with other spectrally efficient multilevel modulation schemes and turbo trellis coded modulation. On the other hand, the study of double-binary CRSC codes over various types fading channels, is necessary and valuable.

Secondly, in the strategy of triple-binary codes/8PSK modulation combination, we can investigate ways to overcome the problem caused by the restriction of symbol-by-symbol iterative decoding so that the introduction of non-uniform interleaver is possible. In this thesis, three frame sizes were investigated and therefore, we can also apply this coding scheme to other frame sizes since there are sizable and flexible space designed for many different frame sizes without any difficulty in circular coding.

Finally, the theoretic foundation for non-binary CRSC codes should be investigated in the future. The performance bounds for non-binary CRSC codes should be evaluated and analyzed.

Bibliography

[1] S.B.Wicker, *Error Control Systems for Digital Communication and Storage.* Prentice-Hall Englewood Cliffs, 1995.

[2] J.G.Proakis, *Digital Communications.* New York: McGraw-Hill,Inc., third ed.,1995.

[3] S.Lin and D.J.Costello, *Error Control Coding:Fundamentals and Applications.* Englewood Cliffs, NJ: Prentice Hall, Inc., 1983.

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

[5] I.S.Reed, "A Class of Multiple-Error-Correcting codes and a Decoding Scheme," *IEEE Transactions on Information Theory*, vol. 4, pp. 38–49, September 1954.

[6] R.C.Bose and D.K.Ray-Chaudhure, "On a class of error correcting binary group codes," *Information and control*, vol. 3, pp. 67–79, Mar. 1960.

[7] A.Hocquenghem, "Codes correcteurs d'erreurs," *Chiffres*, vol. 2, pp. 147-156, 1959.

[8] I.S.Reed and G.Solomon, "Polynomial codes over cerain finite fields," *SIAM Journal on Applied Mathematics*, vol. 8, pp. 300-304, 1960.

[9] G.D.Forney, "Concatenated Codes," *Cambridge, MA: MIT Press*, 1966.

[10] P.Elias, "Coding for Noisy Channels," *IRE Conv. Rec., Part 4*, pp. 37–47, 1955.

[11] J.M.Wozencraft and B.Reiffen, "Sequential Decoding," *Cambridge, MA: MIT Press*, 1961.

[12] J.L.Massey, "Threshold Decoding," *Cambridge, MA: MIT Press*, 1963.

[13] A.J.Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inform. Theory*, vol. IT-13, pp. 260–269, Apr. 1967.

[14] J.K.Omura, "On the Viterbi Decoding Algorithm," *IEEE Transactions on Information Theory*, vol. IT-15, pp. 177–179, January 1969.

[15] J. G.D.Forney, "The Viterbi Algorithm," *Proc. IEEEE*, vol. 61, pp. 268–278, March 1973.

[16] J. G.D.Forney, "Convolutional Codes ii: Maximum likelihood decoding," *Inf. Control*, vol. 25, pp. 222–266, July 1974.

[17] C. C. for Space data Systems, "Recommendations for space data standard: Telemetry channel coding," *Blue Book Issue 2. CCSCS 101.0-B2*, Jan. 1987.

[18] J.Hagenauer, E.Offer, and L.Papke, "Matching Viterbi decoders and Reed-Solmon decoders in concatenated systems," in *Reed-Solomon Codes and their Applications (S.B.Wicker and V.K.Bhargava.eds.)*, pp. 242–271, Piscataway, NJ: IEEE press, 1994.

[19] C.Berrou, A.Glavieux, and P.Thitimasjshima, "Near Shannon limit error-correcting coding and decoding: turbo-codes(1)," in *Proc.. IEEE Int. Conf. on Commun.*, pp. 1064–1070, May 1993.

[20] L.Bahl, J.Cocke, F.Jelinek, and J.Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. IT-20, pp. 284–287, March 1974.

[21] D.Divsalar and F.Pollara, "Serial and hybrid concatenation codes with applications," in *Proc. Int Symp. on Turbo codes and Related Topics, Brest, France*, pp. 80–87, Sept. 2000.

[22] C.Berrou, C.Douillard, and M.Jézéquel, "Multiple parallel concatenation of circular recursive convolutional (crsc) codes," *Annals of Telecommunications*, vol. 54, No.3-4, pp. 166–172, March-April 1999.

[23] C.Berrou and M.J ezéquel, "Non binary convolutional codes for turbo coding," *Electronic Letters*, vol. 35, No.1, pp. 39–40, Jan. 1999.

[24] F.Berens, A.Worm, H.Michel, and N.Wehn, "Implementation Aspects of Turbo-Decoders of Future Radio Applications," in *Proc. VTC'99 Fall. Amsterdam, The Netherlands*, pp. 2601–2605, Sept. 1999.

[25] J.Hagenauer, Fellow, IEEE, E.offer, and L.Papke, "Iterative Decoding of Binary Block and Convolutional Codes," *IEEE Trans. Inform. Theory*, vol. IT-42, pp. 429–445, Mar. 1996.

[26] R. Garello, P. Pierleoni, and S. Benedetto, "Computing the Free Distance of Turbo Codes and Serially Concatenated Codes with Interleavers: Algorithms and Applications," *IEEE Journal on Selected Areas in Communications*, vol. 19, No.5, pp. 800–812, May 2001.

[27] B.Vucetic and J. Yuan, *Turbo Codes Principles and Applications*. Kluwer Academic Publishers, Boston/Dordrecht/London, 2000.

[28] J. Hokfelt, O. Edfors, and T. Maseng, "On the Theory and Performance of Trellis Termination Methods for Turbo Codes," *IEEE Journal on Selected Areas in Communications*, vol. 19, No.5, pp. 838-839, May 2001.

[29] F.Daneshgaran and M.Mondin, "Design of interleavers for turbo codes based on a cost function," in *International Symposium on Turbo Codes and Related Topics. Brest, France*, pp. 255-258, Sept. 1997.

[30] J.Andersen and V.Zyablov, "Interleaver design for turbo coding," in *International Symposium on Turbo Codes and Related Topics. Brest, France*, Sept. 1997.

[31] S.Crozier, J.Lodge, P.Guinan, and A.Hunt, "Performance of turbo codes with relative prime and golden interleaving strategies," in *Sixth International Mobile Satellite Conference. Ottawa, Canada*, pp. 268-275, June 1999.

[32] S.Dolinar and D.Divsalar, "Weight distributions for turbo codes using random and nonrandom permutations," in *TDA progress report, Jet propulsion Lab., Pasadena, CA*, pp. 42-122, August 1995.

[33] A.S.Barbulescu and S.S.Pietrobon, "Interleaver design for three dimensional turbo-codes," in *1995 IEEE International Symposium on Information Theory. Whistler, BC, Canada*, September 1995.

[34] M.Eroz and A.R.Hammons, "On the design of prunable interleavers for turbo codes," in *Vehicular Technology Conference. Houston, USA*, May 1999.

[35] M.Hattori, J.Murayama, and R.J.McEliece, "Psedo-random and self-terminating interleavers for turbo codes," in *Winter 1998 Information Theory Workshop*, pp. 9-10, February 1998.

[36] J.Hokfelt, O.Edfors, and T.Maseng, "Interleaver structures for turbo codes with reduced storage memory requirement," in *Vehicular Technology Conference. Amsterdam, Netherlands*, pp. 1585–1589, Sept. 1999.

[37] J.Hokfelt and T.Maseng, "Methodical interleaver design for turbo codes," in *International Symposium on Turbo Codes and Related Topics. Brest, France*, pp. 212–215, Sept. 1997.

[38] H.R.Sadjapour, M.Salehi, N.J.A.Sloane, and G.Nebe, "Interleaver design for short block length turbo codes," in *IEEE International Conference on Communications. New Orleans, USA*, June 2000.

[39] M.Öberg, "Turbo Coding and Decoding for Signal Transmission and Recording Systems," in *PhD thesis. University of California,San Diego, CA, USA*, 2000.

[40] A.Shibutani, H.Suda, and F.Adachi, "Multi-stage interleaver for turbo codes in ds-cdma mobile radio," in *Asia-Pasific Conference on Communications*, November 1998.

[41] O.Y.Takeshita and D.J.Costello, "New classes of algebraic interleavers for turbo-codes," in *IEEE international Symposium on Information Theory.Cambridge, MA, USA*, p. 419, August 1998.

[42] W.Blacker, E.Hall, and S.Wilson, "Turbo code termination and interleaver conditions," *Electron. Lett.*, vol. 31, No.24, pp. 2082–2084, Nov.23 1995.

[43] A.S.Barbulescu and S.S.Pietrobon, "Terminating the trellis of turbo codes in the same state," *Electron. Lett.*, vol. 31, No.1, pp. 22–23, Jan. 1995.

[44] M.C.Reed and S.S.Pietrobon, "Turbo code termination schemes and a novel alternative for short frames," in *7th IEEE Int. Symp. Personal, Indoor, Mobile Communications,Taipai, Taiwan*, vol. 2, pp. 354–358, Oct.15-18 1996.

[45] P.Guinand and J.Lodge, "Trellis termination for turbo encoders," in *17th Biennial Symp. On communications, Kingston, Canada*, pp. 389–392, May 1994.

[46] J.Hokfelt, O.Edfors, and T.Maseng, "A survey on trellis termination alternatives for turbo codes," in *IEEE Vehicular Technology Conference. Houston, Texas, USA*, pp. 2225–2229, May 1999.

[47] J.Hokfelt, C.F.Leanderson, O.Edfors, and T.Maseng, "Distance spectrum of turbo codes using different trellis termination methods," in *International Symposium on Turbo Codes and Related Topics. Brest, France*, pp. 463–466, Sept. 1997.

[48] J.B.Anderson and S.M.Hladik, "Tailbiting map decoders," *IEEE Journal on Selected Areas in Communications*, vol. 16, No.2, pp. 297–302, Feb. 1998.

[49] S.Crozier, P.Guinand, J.Lodge, and A.Hunt, "Construction and performance of new tail-biting turbo codes," in *6-th International Workshop on Digital Signal Processing Techniques for Space Applications (DSP'98). Noordwijk, The Netherlands*, Sept. 1998.

[50] N. Stralen, J.A.F.Ross, and J.B.Anderson, "Tailbiting and decoding recursive systematic codes," *Electron. Lett.*, vol. 35, No.17, pp. 1461–1462, Aug. 1999.

[51] Y.-P.Wand, R.Ramesh, A.Hassan, and H.Koorapaty, "On map decoding for tailbiting convolutional codes," in *IEEE international Symposium on Information Theory*, p. 225, June 1997.

[52] C.Weib, C.Bettstetter, S.Riedel, and D.J.Costello, "Turbo decoding with tailbiting trellises," in *1998 URSI International Symposium on Signals, Systems and Electronics*, pp. 343–348, September 1998.

[53] C.Berrou, C.Douillard, and M.Jézéquel, "Designing turbo codes for low rates," *Digest of IEE Colloq. on "Turbo codes in digital bordcasting-could it double capacity?"*, vol. 165, Nov. 1999.

[54] C.Berrou and A.Glavieux, "Near optimum error correcting coding and decoding: turbo-codes," *IEEE Trans. Com*, vol. 44, No.10, pp. 1261–1271, Oct. 1996.

[55] C.Berrou and A.Glavieux, "Turbo codes, general principles and applications," *Proc. of the 6th Int. Tirrenia Workshop on Digital Communications, Pisa, Italy*, pp. 215–226, Sept. 1993.

[56] C.Douillard, M.Jézéquel, C.Berrou, N.Brengarth, J.Tousch, and N.Pham, "The Turbo Code Standard for DVB-RCS," in *Proc. of the 2nd Int Symp. on Turbo codes, Brest, France*, pp. 551–554, Sept. 2000.

[57] "Digital Video Broadcasting (DVB): Interaction channel for satellite distribution systems," *http://www.etsi.org*, Dec. 2000.

[58] N.Brengarth, R.Novello, N.Pham, V.Piloni, and J.Tousch, "DVB-RCS turbo code on a commercial OPB statellite payload: Skyplex," in *Proc. of the 2nd Int Symp. on Turbo codes, Brest, France*, pp. 535–538, Sept. 2000.

[59] S.Dolinar, D.Divsalar, and F.Pollara, "Code performance as a function of block size," in *TMO progress report, JPL, NASA*, pp. 42–133.

[60] Y.Wu and B.D.Woerner, "The Influence of Quantization and Fixed Point Arithmetic upon the BER Performance of Turbo Codes," in *Proc. IEEE International Conference on Vehicular Tecnology (VTC Spring'99)*, vol. 2, pp. 1683–1687, May 1999.

[61] Y.Wu and B.D.Woerner, "Internal data width SISO decoding module with modular renormalization," in *Proc. IEEE Veh. Tech. Conf., (Tokyo, Japan)*, May 2000.

[62] D.E.Cress and W.J.Ebel, "Turbo Code Implementation Issues for Low Latency, Low Power Applications," in *Proceedings of 1998 Symp. on wireless Personal Communications, MPRG, Virginia Tech, VA, USA*, June 10-12 1998.

[63] Z. et al., "VLSI Implementation Issues of Turbo Decoder Design for Wireless Applications," in *Proc. 1999 IEEE Workshop on Signal Processing System (SiPS), Design and Implementation, Taipei, Taiwan*, October 20-22 1999.

[64] H.Michel, A.Worm, and N.Wehn, "Influence of Quantization on the Bit-Error Performance of Turbo-Decoders," in *Proc. VTC'00 Spring, Tokyo, Japan*, May 2000.

[65] H.Michel and N. Wehn, "Turbo-decoder Quantization for umts," in *IEEE communications letters*, vol. XX.NO.Y, MONTH 2000.

[66] "Simulation source code examples," *http://pw1.netcom.com/chip.f/viterbi.html*, 2001.

[67] G.Ungerboeck, "Channel coding with multilevel/phase signals," *IEEE Trans. Inf. Theory*, vol. IT-28, pp. 56–67, Jan. 1982.

[68] G.Ungerboeck, "Trellis-coded modulation with redundant signal sets. Part I: Introduction," *IEEE commun.Mag.*, vol. 25, No.2, pp. 5–11, 1987.

[69] G.Ungerboeck, "Trellis-coded modulation with redundant signal sets. Part Ii: State of the art," *IEEE commun.Mag.*, vol. 25, No.2, pp. 12–21, 1987.

[70] P. Robertson and T. Worz, "Coded modulation scheme employing turbo codes," *Electron. Lett.*, vol. 31, pp. 1546–1547, Aug. 1995.

[71] P. Robertson and T. Worz, "Bandwidth-Efficient Turbo Trellis-Coded Modulation Using Punctured Component Codes," *IEEE Journal on Selected Areas in Communications*, vol. 16, No.2, pp. 206–218, Feb. 1998.

Appendix A

Statistical methods for the measurement of BER and FER

In this Appendix, we develop a procedure for bit and frame error rate measurement through the application of statistical methods. Our analysis is based on an article on http://www.iol.unh.edu/testsuites/fe/tests/pmd/appendeixD.html.

Bit error rate and Frame error rate are the measures of how well the demodulator and decoder perform. They may be interpreted as the average number of errors that would occur in a long sequence of transmitted bits. To make a reliable measurement, the number of information bits should be transmitted large enough, or enough number of errors should be observed, so that a reasonable conclusion can be inferred from the test results.

In this appendix, two statistical methods, hypothesis testing and confidence intervals, are discussed. The confidence interval for some of the error probability is derived in this thesis.

Statistical Model

Assume that every bit received is an independent Bernoulli trial that is a test for which there are only two possible outcomes (like a coin toss). The property of

independence implies that the outcome of one Bernoulli trial has no effect on the outcomes of the other Bernoulli trials. Define

$$f(n) = \begin{cases} 1 & if\ error\ occur \\ 0 & if\ no\ error\ occur \end{cases} \tag{A-1}$$

The number of errors k in n independent Bernoulli trials is taken from a binomial distribution. The binomial distribution is defined as

$$b(k, n, p) = \binom{n}{k} p^k (1-p)^{n-k} \tag{A-2}$$

where p is the probability of error.

This model reflects the fact that for a given probability, p, a test in which n bits are sent could yield many possible outcomes. However, some outcomes are more likely than others and this likelihood principle allows us to make conclusions about the BER for a given test result.

Hypothesis Testing

By the statistical method of hypothesis testing, we hypothesize that the probability of a bit error, p, for the system is less than some target BER, P_0. For the hypothesis of $p \leq P_0$, count the number errors k in n bits which have been sent. If $k = 0$, we may conclude $p \leq P_0$; if $k > 0$, we may say $p > P_0$. However, the conclusion may be erroneous since the probability p may be less than P_0 even though $k > 0$. On the other hand, it may be equal or great than P_0 even though $k = 0$. Define two different types error of conclusion shown in Table A-1.

Type I Error	probability α	$k > 0$ even though $p \leq P_0$
Type II Error	probability β	$k = 0$ even though $p > P_0$

Table A-1: Definitions of type I and type II errors

Where

$$\begin{aligned} \alpha &= 1 - (1-p)^n \leq 1 - (1-P_0)^n \\ \beta &= (1-p)^n < (1-P_0)^n \end{aligned} \tag{A-3}$$

The upper bound on the probability of a type II error is a function of the target bit error rate, P_0, and the number of bits, n. In order to verify that p is less than a given P_0 for a given probability of type II error, we have to determine the minimum number of bits that need to be sent, *i.e.*,

$$n > \frac{\ln \beta}{\ln(1-P_0)} \tag{A-4}$$

In practice, if $P_0 = 10^{-m}$, take $n = 10^{m+2}$.

Equation A-3 shows that increasing n to make β small, the upper bound on α is raised simultaneously. This makes sense since the likelihood of observing a bit error increases with the number of bits that you send, no matter how small bit error rate is. Therefore, while the hypothesis test is very useful in determining a reasonable value for n, we must be very careful in interpreting the results. Specifically, if we send n bits and observe no errors, we are confident that p is less than our target bit error rate. However, if we do observe bit errors, we cannot be quick to assume that the system did not meet the BER target since the probability of a type I error is so large. In the case of $k > 0$, a confidence interval can be used to help us interpret k.

Confidence Interval

The statistical method of confidence intervals will be used to establish a lower bound on the bit error rate given that $k > 0$. A confidence interval is a range of values that is likely to contain the actual value of some parameter of interest. The interval is derived from the measured value of the parameter, referred to as the point estimate, and the confidence level, $(1 - \alpha)$, the probability that the parameter's actual value lies within the interval.

A confidence interval requires a statistical model of the parameter to be bounded. In this case, we use the statistical model for k given in Equation A-2. If we were to compute the area under the binomial curve for some interval, we would be computing the probability that k lies within that interval. To compute the area under the binomial curve, we need a value for the parameter p. To compute a confidence interval for k, we assume that k/n, the point estimate for p, is the actual value of p. The computation of the lower tolerance bound for k based on the confidence interval $[k_l, +\infty]$ is a special case. Actual value of k is greater than k_l with probability equal to the confidence level.

To determine the value of k_l, it is useful to assume that the binomial distribution can be approximated by a normal (Gaussian) distribution when n is large. The mean and variance of this equivalent distribution are the mean and variance of the corresponding binomial distribution (given in Equations A-5 and A-6).

$$\mu_k = np \tag{A-5}$$

$$\sigma_k^2 = np(1-p) \tag{A-6}$$

Now, let α be the probability that $Z \leq z$, where Z is a standard normal random variable. A standard random variable is one whose mean is zero and whose variance is one. The random variable K can be standardized as shown in Equation A-7.

$$Z = \frac{K - \mu_k}{\sigma_k} \tag{A-7}$$

This concept is shown in Figure A-1.

Note that Z is greater than z with probability $(1-\alpha)$, the confidence level. We apply this inequality to Equation A-7 and solve for K to get Equation A-8.

$$\begin{aligned} K &> \mu_k + z_\alpha \sigma_k \\ K &> np + z_\alpha \sqrt{np(1-p)} \end{aligned} \tag{A-8}$$

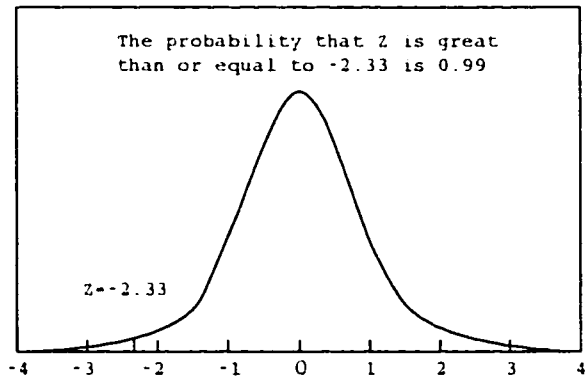


Figure A-1: Computing the probability that $Z \geq -2.33$
(standard normal distribution)

As assume $p = k/n$, we can generate an expression for k_1, the value that K will exceed with probability $(1 - \alpha)$. This expression is given in Equation A-9.

$$k_1 = k + z_\alpha n \sqrt{\frac{(k/n)(1 - k/n)}{n}} \tag{A-9}$$

Sample Test Construction

Conclude from the discussions above, two inequalities are derived from Equation A-4 and

$$P_0 \geq p_1 = \frac{k_1}{n} + z_\alpha \sqrt{\frac{(k/n)(1 - k/n)}{n}} \tag{A-10}$$

to construct a bit error rate test.

$$\begin{cases} n & > & \frac{-\ln \beta}{P_0} \\ -\ln \beta & \geq & k + z_\alpha \sqrt{k} \end{cases} \tag{A-11}$$

Assuming that $\ln(1 - P_0)$ is $-P_0$ (valid for P_0 much less than one) and $(1 - k/n)$ is very close to 1 and substituting $-\ln \beta / P_0$ for n.

The largest value of k that satisfies equation $-\ln \beta = k + z_\alpha \sqrt{k}$ is k_1. The benefit of these two equations is that a bit error rate test is uniquely defined by β and α and that the test scales with P_0. Table A-2 defines n and k_1 in terms of β and α.

b	$-\ln\beta$	n	α	z_α	k_1
0.10	2.30	$2.30/P_0$	0.10	-1.29	5
0.10	2.30	$2.30/P_0$	0.05	-1.65	6
0.05	3.00	$3.00/P_0$	0.05	-1.65	7
0.05	3.00	$3.00/P_0$	0.01	-2.33	10
0.01	4.60	$4.60/P_0$	0.05	-1.65	9
0.01	4.60	$4.60/P_0$	0.01	-2.33	13

Table A-2: n and k_1 as a function of β and α.

Where b is the chance of a type I error. As an example, let us construct a test to determine if a given system is operating at a bit error rate of 10^{-6} or better. Given that a 5% chance of a type I error is acceptable, the test would take the form of sending 3×10^6 bits and counting the number of errors. If no errors are counted, we are confident that the BER was 10^{-6} or better.

Given that a 5% chance of a type II error is acceptable, we find that k_1 is 7. If more than 7 errors are counted, we are confident that the bit error rate is greater than 10^{-6}. However, up to 100 errors are counted in this thesis.

Moreover, we can find $z_{\alpha/2}$ such that

$$P(-z_{\alpha/2} < Z < z_{\alpha/2}) = \frac{1}{\sqrt{2\pi}} \int_{-z_{\alpha/2}}^{z_{\alpha/2}} e^{-z^2/2} dz = 1 - \alpha$$

where Z is a standard distributed random variable with mean 0 and variance 1. Now let

$$Z = \frac{\overline{X} - \mu}{\sigma/\sqrt{n}} \tag{A-12}$$

Substituting, we find

$$P(-z_{\alpha/2} < \frac{\overline{X} - \mu}{\sigma/\sqrt{n}} < z_{\alpha/2}) = P(\overline{X} - z_{\alpha/2}\sigma/\sqrt{n} < \mu < \overline{X} + z_{\alpha/2}\sigma/\sqrt{n}) = 1 - \alpha$$

Thus, we see that, with probability $1 - \alpha$, the true mean of the distribution lies within the interval $(\overline{X} - z_{\alpha/2}\sigma/\sqrt{n}, \overline{X} + z_{\alpha/2}\sigma/\sqrt{n})$. Standard values for $z_{\alpha/2}$ are shown in Table A-3.

$1-\alpha$	$z_{\alpha/2}$
0.9	1.645
0.95	1.96
0.99	2.576
0.999	3.291

Table A-3: Tail probabilities for the Gaussian distribution.

For example, determine the confidence interval of frame size $N = 212$ in Section 3.5 and bit error rate 9.8×10^{-7} at 3.5dB with code rate 2/3, $n = 2 * 212 * 279805 = 1.19 \times 10^8$. Define bit error rate

$$P(e) = \frac{1}{n} \sum_{i=1}^{n} f(i)$$

where

$$\begin{aligned} P_r[f(i) = 1] &= P(e) \\ P_r[f(i) = 0] &= 1 - P(e) \end{aligned}$$

Find the expectation and variance of Equation A-1:

$$E[f(i)] = 1 \times P_r[f(i) = 1] + 0 \times P_r[f(i) = 0] = P(e)$$

$$\begin{aligned} \sigma^2 &= E[f^2(i)] + E^2[f(i)] \\ &= 1^2 \times P(e) + 0^2 \times [1 - P(e)] + P^2(e) \\ &= P(e)[1 - P(e)] \end{aligned}$$

Consider $1 - \alpha = 0.99$ at $z_{\alpha/2} = 2.576$, where $\overline{X} = P(e) = 9.8 \times 10^{-7}$ and $\sigma \approx \sqrt{9.8 \times 10^{-7}} = 9.9 \times 10^{-4}$, therefore, $z_{\alpha/2}\sigma/\sqrt{n} = 2.576 \times 9.9 \times 10^{-4}/\sqrt{1.19 \times 10^8} = 2.34 \times 10^{-7}$, then,

$$\overline{X} - z_{\alpha/2}\sigma/\sqrt{n} = 9.8 \times 10^{-7} - 2.34 \times 10^{-7} = 7.46 \times 10^{-7} \qquad \text{(A-13)}$$

$$\overline{X} + z_{\alpha/2}\sigma/\sqrt{n} = 9.8 \times 10^{-7} + 2.34 \times 10^{-7} = 1.2 \times 10^{-6} \qquad \text{(A-14)}$$

Hence,

$$\begin{aligned} & P(\overline{X} - z_{\alpha/2}\sigma/\sqrt{n} < \mu < \overline{X} + z_{\alpha/2}\sigma/\sqrt{n}) \\ = \; & P(7.46 \times 10^{-7} < \mu < 1.2 \times 10^{-6}) \\ = \; & 1 - \alpha \end{aligned} \tag{A-15}$$

i.e., the confidence interval of this bit error rate is within $(7.46 \times 10^{-7},\ 1.2 \times 10^{-6})$ with probability 0.99.

To calculate the confidence interval of 100 errors taken, apply Equation A-12, $K = 100$, $\sigma_k = \sqrt{nP(e)} = \sqrt{1.19 \times 10^8 \times 9.8 \times 10^{-7}} = 10.8$ and $z_\alpha = 2.33$, then

$$\begin{aligned} & P(K - z_\alpha\sigma_k < \mu < K + z_\alpha\sigma_k) \\ = \; & P(75 < \mu < 125) \\ = \; & 1 - \alpha \end{aligned} \tag{A-16}$$

or

$$K > np + z_\alpha\sigma_k = 1.19 \times 10^8 \times 9.8 \times 10^{-7} + (-2.33) \times 10.8 \approx 91$$

i.e., more than 91 errors should be observed and the confidence interval of taking 100 errors is within (75, 125) with probability 0.99.

Frame Error Rate Measurement

It is often easier to measure packet errors than it is to measure bit errors. In these cases, it is helpful to have some linkage between the packet error rate and the bit error rate. To make this linkage, we assume that the bit error rate is low enough and the packet size is small enough so that each erroneous packet contains exactly one erroneous bit.

For example, determine the confidence interval of frame size $N = 212$ and frame error rate 3.6×10^{-5} at 3.5dB with code rate 2/3, $n = 279805 = 2.8 \times 10^5$.

Consider $1 - \alpha = 0.99$ at $z_{\alpha/2} = 2.576$, where $\overline{X} = P_F(e) = 3.6 \times 10^{-5}$ and $\sigma \approx \sqrt{3.6 \times 10^{-5}} = 6 \times 10^{-3}$, therefore, $z_{\alpha/2}\sigma/\sqrt{n} = 2.576 \times 3.6 \times 10^{-3}/\sqrt{2.8 \times 10^5} =$

2.9×10^{-5}, then,

$$\overline{X} - z_{\alpha/2}\sigma/\sqrt{n} = 3.6 \times 10^{-5} - 2.9 \times 10^{-5} = 0.7 \times 10^{-5} \quad \text{(A-17)}$$

$$\overline{X} + z_{\alpha/2}\sigma/\sqrt{n} = 3.6 \times 10^{-5} + 2.9 \times 10^{-5} = 6.5 \times 10^{-5} \quad \text{(A-18)}$$

Hence,

$$\begin{aligned} & P(\overline{X} - z_{\alpha/2}\sigma/\sqrt{n} < \mu < \overline{X} + z_{\alpha/2}\sigma/\sqrt{n}) \\ = \; & P(0.7 \times 10^{-5} < \mu < 6.5 \times 10^{-5}) \\ = \; & 1 - \alpha \end{aligned} \quad \text{(A-19)}$$

i.e., the confidence interval of this bit error rate is within $(0.7 \times 10^{-5},\ 6.5 \times 10^{-5})$ with probability 0.99.