VITERBI ALGORITHM IN CONTINUOUS-PHASE FREQUENCY SHIFT KEYING

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology
In
VLSI Design and Embedded System

By

L. Mallesh

Roll No: 20507008

Department of Electronics and Communication Engineering
National Institute of Technology
Rourkela
2005-2007

VITERBI ALGORITHM IN CONTINUOUS-PHASE FREQUENCY SHIFT KEYING

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology
In
VLSI Design and Embedded System

By

L. Mallesh

Roll No: 20507008

Under the Guidance of

Prof. G. Panda

Department of Electronics and Communication Engineering

National Institute of Technology

Rourkela

2005-2007

National Institute of Technology
Rourkela

CERTIFICATE

This is to certify that the thesis entitled. "**Viterbi Algorithm in Continuous-Phase Frequency Shift Keying**" submitted by Sri **L.Mallesh** in partial fulfillment of the requirements for the award of Master of Technology Degree in Electronics and Communication Engineering with specialization in "**VLSI Design and Embedded System**" at the National Institute of Technology, Rourkela (Deemed University) is an authentic work carried out by his under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University/Institute for the award of any Degree or Diploma.

Date

Prof.G.Panda
Dept. of Electronics and Communication Engg.
National Institute of Technology
Rourkela-769008

ACKNOWLEDGEMENTS

This project is by far the most significant accomplishment in my life and it would be impossible without people who supported me and believed in me.

I would like to extend my gratitude and my sincere thanks to my honorable, esteemed supervisor **Prof. G. Panda,** Head, Department of Electronics and Communication Engineering. He is not only a great lecturer with deep vision but also and most importantly a kind person. I sincerely thank for his exemplary guidance and encouragement. His trust and support inspired me in the most important moments of making right decisions and I am glad to work with him.

I want to thank all my teachers **Prof. G.S. Rath**, **Prof. K. K. Mahapatra, Prof. S.K. Patra** and **Prof. S.K. Meher** for providing a solid background for my studies and research thereafter. They have been great sources of inspiration to me and I thank them from the bottom of my heart.

I would like to thank all my friends and especially my classmates for all the thoughtful and mind stimulating discussions we had, which prompted us to think beyond the obvious. I've enjoyed their companionship so much during my stay at NIT, Rourkela.

I would like to thank all those who made my stay in Rourkela an unforgettable and rewarding experience.

Last but not least I would like to thank my parents, who taught me the value of hard work by their own example. They rendered me enormous support during the whole tenure of my stay in NIT Rourkela.

L.Mallesh

CONTENTS

ABSTRACT

The Viterbi algorithm, an application of dynamic programming, is widely used for estimation and detection problems in digital communications and signal processing. It is used to detect signals in communication channels with memory, and to decode sequential error-control codes that are used to enhance the performance of digital communication systems. The Viterbi algorithm is also used in speech and character recognition tasks where the speech signals or characters are modeled by hidden Markov models. This project explains the basics of the Viterbi algorithm as applied to systems in digital communication systems, and speech and character recognition. It also focuses on the operations and the practical memory requirements to implement the Viterbi algorithm in real-time.

A forward error correction technique known as convolutional coding with Viterbi decoding was explored. In this project, the basic Viterbi decoder behavior model was built and simulated. The convolutional encoder, BPSK and AWGN channel were implemented in MATLAB code. The BER was tested to evaluate the decoding performance.

The theory of Viterbi Algorithm is introduced based on convolutional coding. The application of Viterbi Algorithm in the Continuous-Phase Frequency Shift Keying (CPFSK) is presented. Analysis for the performance is made and compared with the conventional coherent estimator.

The main issue of this thesis is to implement the RTL level model of Viterbi decoder. The RTL Viterbi decoder model includes the Branch Metric block, the Add-Compare-Select block, the trace-back block, the decoding block and next state block. With all done, we further understand about the Viterbi decoding algorithm.

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 INTRODUCTION:

Convolutional coding has been used in communication systems including deep space communications and wireless communications. It offers an alternative to block codes for transmission over a noisy channel. An advantage of convolutional coding is that it can be applied to a continuous data stream as well as to blocks of data. IS-95, a wireless digital cellular standard for CDMA (code division multiple access), employs convolutional coding. A third generation wireless cellular standard, under preparation, plans to adopt turbo coding, which stems from convolutional coding.

The Viterbi decoding algorithm is a decoding process for convolutional codes in memory-less noise. The algorithm can be applied to a host of problems encountered in the design of communication systems. The Viterbi decoding algorithm provides both a maximum-likelihood and a maximum a posteriori algorithm. A maximum a posteriori algorithm identifies a code word that maximizes the conditional probability of the decoded code word against the received code word; in contrast a maximum likelihood algorithm identifies a code word that maximizes the conditional probability of the received code word against the decoded code word. The two algorithms give the same results when the source information has a uniform distribution.

1.2 BACKGROUND LITERATURE SURVEY:

The Viterbi Algorithm (VA) was first proposed as a solution to the decoding of convolutional codes by Andrew J. Viterbi in 1967, [1], with the idea being further developed by the same author in [2]. It was quickly shown by Omura [3] that the VA could be interpreted as a dynamic programming algorithm. Both Omura and Forney [3, 4] showed that the VA is a maximum likelihood decoder. The VA is often looked upon as minimizing the error probability by comparing the likelihoods of a set of possible state transitions that can occur, and deciding which of these has the highest probability of occurrence. A similar algorithm, known as the Stack Sequential Decoding Algorithm (SSDA), was described by Forney in [5] as an alternative to the VA, requiring less hardware to implement than the VA. The SSDA has been proposed as an alternative to the VA in such applications as target tracking [6], and high rate convolutional decoding [5]. It can be shown though, that this algorithm is sub-optimum to the VA in that it discards some of the paths that are kept by the VA.

1.3 THESIS CONTRIBUTION:

This section outlines some of the contributions of the study presented in this thesis. In this project, the basic Viterbi decoder behavior model was built and simulated. The convolutional encoder, BPSK and AWGN channel were implemented in MATLAB code. The BER was tested to evaluate the decoding performance. The application of Viterbi Algorithm in the Continuous-Phase Frequency Shift Keying (CPFSK) is presented. Analysis for the performance is made and compared with the conventional coherent estimator and the complexity of the implementation of the Viterbi decoder in hardware device

The main issue of this thesis is to implement the RTL level model of Viterbi decoder. The RTL Viterbi decoder model includes the Branch Metric block, the Add-Compare-Select block, the trace-back block, the decoding block and next state block. With all done, we further understand about the Viterbi decoding algorithm.

1.4 THESIS OUTLINE:

Following the introduction, the remaining part of the thesis is organized as under; Chapter 2 discusses introduction to error control coding. Chapter 3 presents the fundamentals of convolutional code. This chapter discusses the encoder structure and its many representations. Also, it discusses the primary decoding algorithm for convolutional code, namely the Viterbi algorithm. Both hard- and soft-decision Viterbi algorithms are presented in Chapter 3. Furthermore, performance issues related to convolutional code are discussed. Chapter 4 discusses fundamentals of viterbi algorithm and Applications of viterbi decoder. Chapter 5 discusses introduction to Pass band Modulation. Chapter 6 discusses application of viterbi algorithm in continuous-phase frequency shift keying and its implementation.

2. ERROR CONTROL CODING

2.1 PRELIMINARIES:

In this section we define the terms that we will need to handle the later topics. Definitions are tailored to suit this paper and may differ from those presented in the literature.

Digital communications systems are often partitioned as shown in Fig.1.1. The following paragraphs describe the elements of Fig.1.1 and define other terms common to error-control coding.

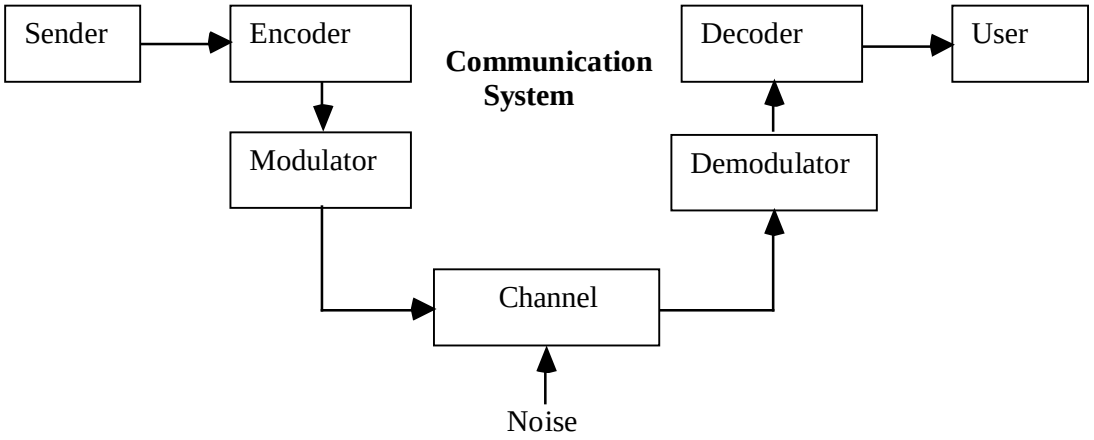


Fig.2.1 - The Digital Communications System

Encoder and Decoder - The encoder adds redundant bits to the sender's bit stream to create a codeword. The decoder uses the redundant bits to detect and/or correct as many bit errors as the particular error-control code will allow.

Modulator and Demodulator - The modulator transforms the output of the encoder, which is digital, into a format suitable for the channel, which is usually analog (e.g., a telephone channel). The demodulator attempts to recover the correct channel symbol in the presence of noise. When the wrong symbol is selected, the decoder tries to correct any errors that result. Communications Channel - The part of the communication system that introduces errors. The channel can be radio, twisted wire pair, coaxial cable, fiber optic cable, magnetic tape, Optical discs or any other noisy medium.

Error-Control Code – A set of codewords used with an encoder and decoder to detect errors, correct errors, or both detect and correct errors.

Bit-Error-Rate (BER) - The probability of bit error. This is often the figure of merit for a error-control code. We want to keep this number small, typically less than 10^{-4}. Bit-error- rate is a useful indicator of system performance on an independent error channel, but it has little meaning on bursty or dependent error channels.

Message-Error-Rate (MER) - The probability of message error. This may be a more appropriate figure of merit because the smart operator wants all of his messages error-free and could care less about the BER.

Undetected Message Error Rate (UMER) - The probability that the error detection decoder fails and an errored message (codeword) slips through undetected. This event happens when the error pattern introduced by the channel is such that the transmitted codeword is converted into another valid codeword. The decoder can't tell the difference and must conclude that the message is error-free. Practical error detection codes ensure that the UMER is very small, often less than 10^{-16}.

Random Errors - Errors that occur independently. This type of error occurs on channels that are impaired solely by thermal (Gaussian) noise. Independent-error channels are also called memoryless channels because knowledge of previous channel symbols adds nothing to our knowledge of the current channel symbol.

Burst Errors - Errors that are not independent. For example, channels with deep fades experience errors that occur in bursts. Because the fades make consecutive bits more likely to be in error, the errors are usually considered dependent rather than independent. In contrast to independent-error channels, burst-error channels have memory.

Energy per Bit- The amount of energy contained in one information bit. This is not a parameter that can be measured by a meter, but it can be derived from other known parameters. Energy per bit (E_b) is important because almost all channel impairments can be overcome by increasing the energy per bit. Energy per bit (in joules) is related to transmitter power P_t (in watts), and bit rate R (in bits per second), in the following way:

$$E_b = \frac{P_t}{R}$$

If transmit power is fixed, the energy per bit can be increased by lowering the bit rate. Thus, the reason why lower bit rates are considered more robust. The required energy per bit to maintain reliable communications can be decreased through error-control coding as we shall see in the next section.

Coding Gain - The difference (in dB) in the required signal-to-noise ratio to maintain reliable communications after coding is employed. Signal-to-noise ratio is usually given by E_b/N_0, where N_0 is the noise power spectral density measured in watts/Hertz (joules). For example, if a communications system requires a E_b/N_0 of 12 dB to maintain a BER of 10^{-5}, but after coding it requires only 9 dB to maintain the same BER, then the coding gain is 12 dB minus 9 dB = 3 dB. (Recall that dB (decibels) = 10 $\log_{10} X$, where X is a ratio of powers or energies.)

Code Rate - Consider an encoder that takes k information bits and adds r redundant bits (also called parity bits) for a total of $n = k + r$ bits per codeword. The code rate is the fraction k/n and the code is called a (n, k) error-control code. The added parity bits are a burden (i.e. overhead) to the communications system, so the system designer often chooses a code for its ability to achieve high coding gain with few parity bits.

2.2 ADVANTAGES OF CODING:

The traditional role for error-control coding was to make a troublesome channel acceptable by lowering the frequency of error events. The error events could be bit errors, message errors, or undetected errors. Coding's role has expanded tremendously and today coding can do the following:

Reduce the occurrence of undetected errors. This was one of the first uses of error-control coding. Today's error detection codes are so effective that the occurrence of undetected errors is, for all practical purposes, eliminated.

Reduce the cost of communications systems. Transmitter power is expensive, especially on satellite transponders. Coding can reduce the satellite's power needs because messages received at close to the thermal noise level can still be recovered correctly. Overcome Jamming. Error-control coding is one of the most effective techniques for reducing the effects of the enemy's jamming. In the presence of pulse jamming, for

example, coding can achieve coding gains of over 35 dB.

Eliminate Interference. As the electromagnetic spectrum becomes more crowded with man-made signals, error-control coding will mitigate the effects of unintentional interference.

For strictly power-limited (unlimited bandwidth) channels, Shannon's lower bound on E_b/N_0 is 0.69, or –1.6 dB. In other words, we must maintain an E_b/N_0 of at least -1.6 dB to ensure reliable communications, no matter how powerful an error-control code we use. For bandwidth-limited channels with Gaussian noise, Shannon's capacity formula can be written as the following.

Where r is the spectral bit rate in bits/s/Hz. For example, consider a bandwidth-limited channel operating with uncoded quadrature phase shift keying (a common modulation technique with 2 bits/symbol and a maximum spectral bit rate of $r = 2$ bit/s/Hz) and a required BER of 10^{-5}. We know that without coding, this communications system requires an E_b/N_0 of 9.6 dB [5]. Shannon's formula above says that to maintain reliable communications at an arbitrarily low BER, we must maintain (for $r = 2$ bits/s/Hz) an E_b/N_0 of at least 1.5 (1.8 dB). Therefore, if we need to lower the required E_b/N_0 by more than 7.8 dB, *coding can't do it.* We must resort to other measures, like increasing transmitter power. In practice, the situation is worse because we have no practical code that achieves Shannon's lower bound. A more realistic coding gain for this example is 3 dB rather than 7.8 dB.

Another limitation to the performance of error-control codes is the modulation technique of the communication system. Coding must go hand-in-hand with the choice of modulation technique for the channel. Even the most powerful codes cannot overcome the consequences of a poor modulation choice.

2.3 PRINCIPLE OF ERROR CODING:

A full understanding of the structure and performance of error-control codes requires a foundation in modern algebra and probability theory, which is beyond the scope of this paper. Instead, we appeal to the reader's intuition and common sense. Let's begin by showing how the encoder and decoder work for binary block codes.

The block encoder takes a block of k bits and replaces it with a n-bit codeword (n is bigger than k). For a binary code, there are 2^k possible codewords in the *codebook*. The channel introduces errors and the received word can be any one of 2^n n-bit words of which only 2^k are valid codewords. The job of the decoder is to find the codeword that is closest to the received n-bit word. How a practical decoder does this is beyond the scope of this paper, but our examples will use a brute force look-up table method.

2.3.1 Error Detection Only

The minimum distance of a code gives a measure of its error detection capability. An error control code can be used to detect all patterns of u errors in any codeword as long as $d_{min} = u + 1$. The code may also detect many error patterns with more than u errors, but it is guaranteed to detect *all* patterns of u errors or less. We'll assume that the error detection decoder comprises a look-up table with all 2^k valid codewords stored. When an n-bit word is received by the decoder, it checks the look-up table and if this word is one of the allowable codewords, it flags the n-bit word as error-free and sends the corresponding information bits to the user. We'll use Figure 2 to illustrate three cases: no errors, a detectable error pattern, and an undetectable error pattern.

Case 1: No errors. Let's assume that the encoder sends codeword C and the channel introduces no errors. Then codeword C will also be received, the decoder will find it in the look-up table, and decoding will be successful.

Case 2: Detectable error pattern. This time we send codeword C and the channel introduces errors such that the n-bit word Y is received. Because Y is not a valid codeword, the decoder will not find it in the table and will therefore flag the received n-bit word as an errored codeword. The decoder does not necessarily know the number or location of the errors, but that's acceptable because we only asked the decoder to *detect* errors. Since the decoder properly detected an errored codeword, decoding is successful.

Case 3: Undetectable error pattern. We send codeword C for the third time and this time the channel introduces the unlikely (but certainly possible) error pattern that converts codeword C into codeword D. The decoder can't know that codeword C was sent and must assume that codeword D was sent instead. Because codeword D is a valid codeword, the decoder declares the received n-bit word error-free and passes the corresponding information bits on to the user. This is an example of decoder failure.

Naturally, we want the decoder to fail rarely, so we choose codes that have a small probability of undetected error. One of the most popular error detection codes is the shortened Hamming code, also known as the cyclic redundancy check (CRC). Despite its widespread use since the 1960s, the precise performance of CRCs was not known until Fujiwara et al. [7] published their results in 1985.

2.3.2 Forward Error Correction (FEC)

Comparing the spheres surrounding codewords A and *B* in Figure 2, we see that the error correcting capability of a code is given by $d_{min} = 2t + 1$ (this is the minimum separation that prevents overlapping spheres). Or in other words, a code with $d_{min} = 3$ can correct all patterns of 1 error, one with $d_{min} = 5$ can correct all patterns of 2 errors, and so on. A code can be used to correct *t* errors and detect *v* additional errors as long as $d_{min} \geq 2t + v + 1$. Now refer to Figure 2 and consider three error decoding cases for the error correction decoder: correct decoding, decoding failure, and error detection without correction.

Case 1: Correct decoding. Assume that codeword *C* is sent and the *n*-bit word Y is received. Because Y is inside *C*'s sphere, the decoder will correct all errors and error correction decoding will be successful.

Case 2: Decoding failure. This time we send codeword *C* and the channel gives us *n*- bit word *Z*. The decoder has no way of knowing that codeword *C* was sent and must decode to *D* since *Z* is in *D*'s sphere. This is an example of error correction decoder failure.

Case 3: Error detection without correction. This case shows one way that an error correction code can be used to also detect errors. We send codeword *C* and receive *n*-bit word X. Since *X* is not inside any sphere, we won't try to correct it. We do, however, recognize that it is an errored codeword and report this information to the user.

In the last example, we could try to correct *n*-bit word *X* to the nearest valid codeword, even though X was not inside any codeword's sphere. A decoder that attempts to correct all received n-bit words whether they are in a decoding sphere or not is called a complete decoder. On the other hand, a decoder that attempts to correct only *n*-bit words that lie inside a decoding sphere is called an incomplete or *bounded distance* decoder. Bounded distance decoders are much more common than complete decoders. Now that we understand the basics of encoding and decoding, let's investigate a simple error correction code.

2.3.3 The Repetition Code

Consider a (5, 1) repetition code that repeats each bit four times. The encoder looks like this:

$0 \longrightarrow 0\,0\,0\,0\,0$

$1 \longrightarrow 1\,1\,1\,1\,1$

The decoder takes 5 bits at a time and counts the number of 1's. If there are three or more, the decoder selects 1 for the decoded bit. Otherwise, the decoder selects 0. The minimum distance of this code is 5, so it can correct all patterns of two errors. To compute the error performance of this code, consider a random error channel with probability of bit error, *p*. After decoding, the probability of bit error is simply the probability of three or more bit errors in a 5 bit codeword. This probability is computed for several values of *p* with results listed in Table 1.

Post-Decoding probability of bit error for (5, 1) Repetition Code	
Input BER	Output BER
10^{-2}	9.9×10^{-6}
10^{-3}	1.0×10^{-8}
10^{-4}	1.0×10^{-11}
10^{-5}	1.0×10^{-14}
10^{-6}	1.0×10^{-17}

Table 2.1 Post-Decoding probability of bit error for (5, 1) Repetition code

The values listed in Table 1.1 show that even this simple code offers dramatic improvements in error performance, but at the price of a 400% overhead burden.

2.4 POPULAR CODING TECHNIQUES:

In this section we highlight six of the most popular error-control coding techniques. We will discuss automatic repeat request (ARQ), forward error correction (FEC), hybrid ARQ, interleaving, erasure decoding, and concatenation.

2.4.1 Automatic Repeat Request (ARQ)

An error detection code by itself does not control errors, but it can be used to request repeated transmission of errored codewords until they are received error-free. This

technique is called automatic repeat request, or *ARQ*. In terms of error performance, ARQ outperforms forward error correction because codewords are always delivered error-free (provided the error detection code doesn't fail). This advantage does not come free – we pay for it with decreased throughput. The chief advantage of ARQ is that error detection requires much simpler decoding equipment than error correction. ARQ is also adaptive since it only re-transmits information when errors occur. On the other hand, ARQ schemes require a feedback path which may not be available. They are also prone to duping by the enemy. A pulse jammer can optimize its duty cycle to increase its chances of causing one or more errors in each codeword. Ideally (from the jammer's point of view), the jammer forces the communicator to retransmit the same codeword over and over, rendering the channel useless.

There are two types of ARQ: stop and wait ARQ and continuous ARQ.

Stop-and-wait ARQ: With stop-and-wait ARQ, the transmitter sends a single codeword and waits for a positive acknowledgement (ACK) or negative acknowledgement (NAK) before sending any more codewords. The advantage of stop-and-wait ARQ is that it only requires a half duplex channel. The main disadvantage is that it wastes time waiting for ACKs, resulting in low throughput.

Continuous ARQ: Continuous ARQ requires a full duplex channel because codewords are sent continuously until a NAK is received. A NAK is handled in one of two ways: With *go back-N* ARQ, the transmitter retransmits the errored codeword plus all codewords that followed until the NAK was received. The parameter *N* is determined from the round trip channel delay. For geosynchronous satellite channels, *N* can be very large because of the 540 millisecond round trip delay. The transmitter must store *N* codewords at a time and large values of *N* result in expensive memory requirements. With *selective-repeat* ARQ, only the errored codeword is retransmitted, thus increasing the throughput over go back-*N* ARQ. Both types of continuous ARQ offer greater throughput efficiency than stop-and-wait ARQ at the cost of greater memory requirements.

2.4.2. Forward Error Correction (FEC)

Forward error correction is appropriate for applications where the user must get the message right the first time. The one-way or broadcast channel is one example. Today's error correction codes fall into two categories: block codes and convolutional codes.

Block Codes: The operation of binary block codes was described in Section 4.0 of this paper. All we need to add here is that not all block codes are binary. In fact, one of the most popular block codes is the Reed-Solomon code which operates on *m*-bit symbols, not bits. Because Reed-Solomon codes correct symbol errors rather than bit errors, they are very effective at correcting burst errors. For example, a 2-symbol error correcting Reed-Solomon code with 8 bit-symbols can correct all bursts of length 16 bits or less. Reed Solomon Codes are used in JTIDS, a new deep space standard, and compact disc (CD) players.

Convolutional Codes: With convolutional codes, the incoming bit stream is applied to a *K*-bit long shift register. For each shift of the shift register, *b* new bits are inserted and *n* code bits are delivered, so the code rate is *b/n*. The power of a convolutional code is a function of its constraint length, *K*. Large constraint length codes tend to be more powerful. Unfortunately, with large constraint length comes greater decoder complexity. There are several effective decoding algorithms for convolutional codes, but the most popular is the Viterbi algorithm, discovered by Andrew Viterbi in 1967. Viterbi decoders are now available on single integrated circuits (VLSI) from several manufacturers. Viterbi decoders are impractical for long constraint length codes because decoding complexity increases rapidly with constraint length. For long constraint length codes ($K > 9$), a second decoding algorithm called sequential decoding is often used. A third decoding technique, *feedback decoding*, is effective on burst-error channels, but is inferior on random error channels. In general, convolutional codes provide higher coding gain than block codes for the same level of encoder/decoder complexity.

One drawback of the codes we have looked at so far is that they all require bandwidth expansion to accommodate the added parity bits if the user wishes to maintain the original unencoded information rate. In 1976, Gottfried Ungerboeck discovered a class of codes that integrates the encoding and modulation functions and does not require bandwidth expansion. These codes are called Ungerboeck codes or trellis coded modulation (TCM). Virtually every telephone line modem on the market today operating above 9.6 k bits/s uses TCM.

2.4.3. Hybrid ARQ

Hybrid ARQ schemes combine error detection and forward error correction to

make more efficient use of the channel. At the receiver, the decoder first attempts to correct any errors present in the received codeword. If it cannot correct all the errors, it requests retransmission using one of the three ARQ techniques described above. Type I hybrid ARQ sends all the necessary parity bits for error detection and error correction with each codeword. Type II hybrid ARQ, on the other hand, sends only the error detection parity bits and keeps the error correction parity bits in reserve. If the decoder detects errors, the receiver requests the error correction parity bits and attempts to correct the errors with these parity bits before requesting retransmission of the entire codeword. Type II ARQ is very efficient on a channel characterized by a "good" state that prevails most of the time and a "bad" state that occurs infrequently.

2.4.4. Interleaving

One of the most popular ways to correct burst errors is to take a code that works well on random errors and interleave the bursts to "spread out" the errors so that they appear random to the decoder. There are two types of interleavers commonly in use today, block interleavers and convolutional interleavers.

The block interleaver is loaded row by row with L codewords, each of length n bits. These L codewords are then transmitted column by column until the interleaver is emptied. Then the interleaver is loaded again and the cycle repeats. At the receiver, the codewords are deinterleaved before they are decoded. A burst of length L bits or less will cause no more than 1 bit error in any one codeword. The random error decoder is much more likely to correct this single error than the entire burst.

The parameter L is called the interleaver degree, or interleaver depth. The interleaver depth is chosen based on worst case channel conditions. It must be large enough so that the interleaved code can handle the longest error bursts expected on the channel. The main drawback of block interleavers is the delay introduced with each row-by-row fill of the interleaver. Convolutional interleavers eliminate the problem except for the delay associated with the initial fill. Convolutional interleavers also reduce memory requirements over block interleavers by about one-half. The big disadvantage of either type of interleaver is the interleaver delay introduced by this initial fill. The delay is a function of the interleaver depth and the data rate and for some channels it can be several seconds long. This long delay may be unacceptable for some applications. On voice circuits, for example, interleaver delays confuse the unfamiliar listener by introducing long pauses

between speaker transitions. Even short delays of less than one second are sufficient to disrupt normal conversation. Another disadvantage of interleavers is that a smart jammer can choose the appropriate time to jam to cause maximum damage. This problem is overcome by randomizing the order in which the interleaver is emptied.

In practice, interleaving is one of the best burst-error correcting techniques. In theory, it is the worst way to handle burst errors. Why? From a strict probabilistic sense, we are converting "Good" errors into "bad" errors. Burst errors have structure and that structure can be exploited. Interleavers "randomize" the errors and destroy the structure. Theory differs from reality, however. Interleaving may be the only technique available to handle burst errors successfully. For example, Viterbi shows that, for a channel impaired by a pulse jammer, exploiting the burst structure is not enough. Interleaving is still required. This does not mean that we should be careless about our choice of code and take up the slack with long interleavers. Codes designed to correct burst errors can achieve the same performance with much shorter interleavers. Until the coding theorists discover a better way, interleaving will be an essential error control coding technique for bursty channels.

2.4.5. Erasure Decoding

When the receiver detects the presence of jamming, fading, or some transient malfunction, it may choose to declare a bit or symbol *erased*. For example, the receiver may erase the symbols "A" and "L" from the message SIGNAL to get SIGN– –. This is not the same as deletion, which would give SIGN. Because the location of the erased bits is known, erasure decoding usually requires fewer parity bits than error correction decoding. A code with minimum distance d_{min} can correct e erasures if $d_{min} = e + 1$. An error correction code can be used to correct t errors and e erasures as long as $d_{min} \geq 2t + e + 1$. For example, an error-control code with minimum distance 7 can be used to correct 2 errors and 2 erasures.

3. CONVOLUTIONAL CODING

3.1 INTRODUCTION:

Over the years, there has been a tremendous growth in digital communications especially in the fields of cellular/PCS, satellite, and computer communication. In these communication systems, the information is represented as a sequence of binary bits. The binary bits are then mapped (modulated) to analog signal waveforms and transmitted over a communication channel. The communication channel introduces noise and interference to corrupt the transmitted signal. At the receiver, the channel corrupted transmitted signal is mapped back to binary bits. The received binary information is an estimate of the transmitted binary information. Bit errors may result due to the transmission and the number of bit errors depends on the amount of noise and interference in the communication channel.

Channel coding is often used in digital communication systems to protect the digital information from noise and interference and reduce the number of bit errors. Channel coding is mostly accomplished by selectively introducing redundant bits into the transmitted information stream. These additional bits will allow detection and correction of bit errors in the received data stream and provide more reliable information transmission. The cost of using channel coding to protect the information is a reduction in data rate or an expansion in bandwidth.

3.1.1 TYPES OF CHANNEL CODES:

There are two main types of channel codes, namely block codes and convolutional codes. There are many differences between block codes and convolutional codes. Block codes are based rigorously on finite field arithmetic and abstract algebra. They can be used to either detect or correct errors. Block codes accept a block of k information bits and produce a block of n coded bits. By predetermined rules, n-k redundant bits are added to the k information bits to form the n coded bits. Commonly, these codes are referred to as (n,k) block codes. Some of the commonly used block codes are Hamming codes, Golay codes, BCH codes, and Reed Solomon codes (uses non binary symbols). There are many ways to decode block codes and estimate the k information bits. These decoding techniques will not be discussed here but can be studied in courses on Coding Theory.

Convolutional codes are one of the most widely used channel codes in practical communication systems. These codes are developed with a separate strong mathematical structure and are primarily used for real time error correction. Convolutional codes convert the entire data stream into one single codeword. The encoded bits depend not only on the current k input bits but also on past input bits. The main decoding strategy for convolutional codes is based on the widely used Viterbi algorithm.

3.2 CONVOLUTIONAL CODES:

This chapter describes the encoder and decoder structures for convolutional codes. The encoder will be represented in many different but equivalent ways. Also, the main decoding strategy for convolutional codes, based on the Viterbi Algorithm, will be described. A firm understanding of convolutional codes is an important prerequisite to the understanding of turbo codes.

3.2.1 Encoder Structure

A convolutional code introduces redundant bits into the data stream through the use of linear shift registers as shown in Figure 2.1.

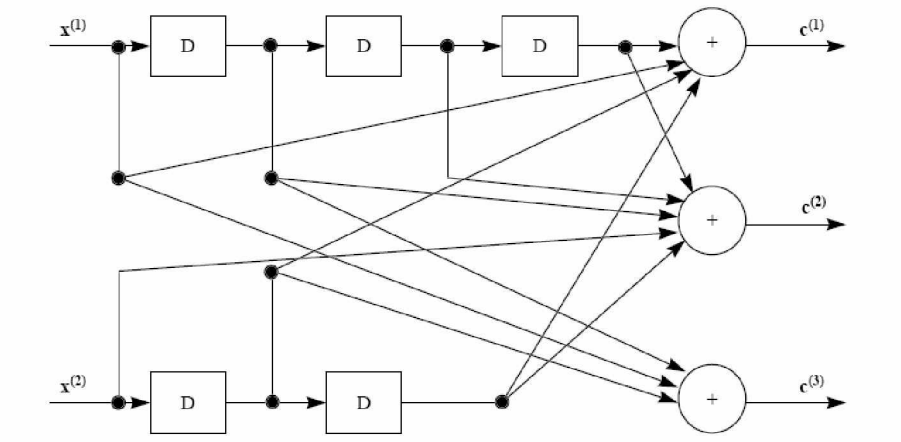


Fig.3.1 Example of Convolutional Encoder.

The information bits are input into shift registers and the output encoded bits are obtained by modulo-2 addition of the input information bits and the contents of the shift registers. The connections to the modulo-2 adders were developed heuristically with no algebraic or combinatorial foundation.

The code rate r for a convolutional code is defined as

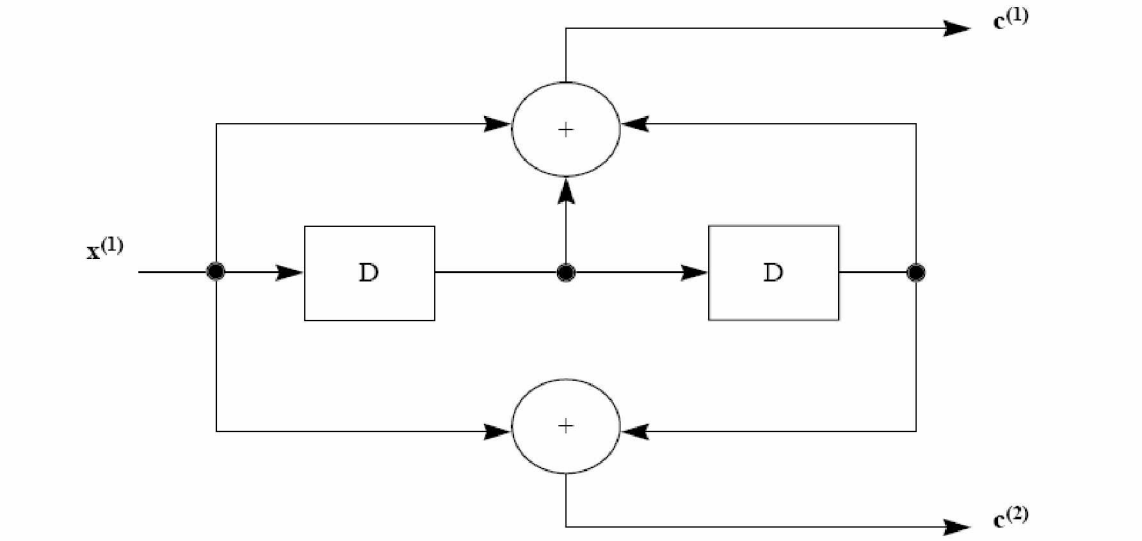
$$r = \frac{k}{n}$$

here k is the number of parallel input information bits and n is the number of parallel output encoded bits at one time interval. The constraint length K for a convolutional code is defined as

$$K = m + 1 \tag{2.2}$$

where m is the maximum number of stages (memory size) in any shift register. The shift registers store the state information of the convolutional encoder and the constraint length relates the number of bits upon which the output depends. For the convolutional encoder shown in Figure 2.1, the code rate r=2/3, the maximum memory size m=3, and the constraint length K=4.

A convolutional code can become very complicated with various code rates and constraint lengths. As a result, a simple convolutional code will be used to describe the code properties as shown in Figure 2.2.



the modulo-2 adders. A generator vector represents the position of the taps for an output. A "1" represents a connection and a "0" represents no connection. For example, the two generator vectors for the encoder in Figure 3.2 are g_1 = [111] and g_2 = [101] where the subscripts 1 and 2 denote the corresponding output terminals.

3.3.2 Tree Diagram Representation

The tree diagram representation shows all possible information and encoded sequences for the convolutional encoder. Figure 2.3 shows the tree diagram for the encoder in Figure 2.2 for four input bit intervals.

In the tree diagram, a solid line represents input information bit 0 and a dashed line represents input information bit 1. The corresponding output encoded bits are shown on the branches of the tree. An input information sequence defines a specific path through the tree diagram from left to right. For example, the input information sequence **x** = {1011} produces the output encoded sequence **c**={11, 10, 00, 01}. Each input information bit corresponds to branching either upward (for input information bit 0) or downward (for input information bit 1) at a tree node.

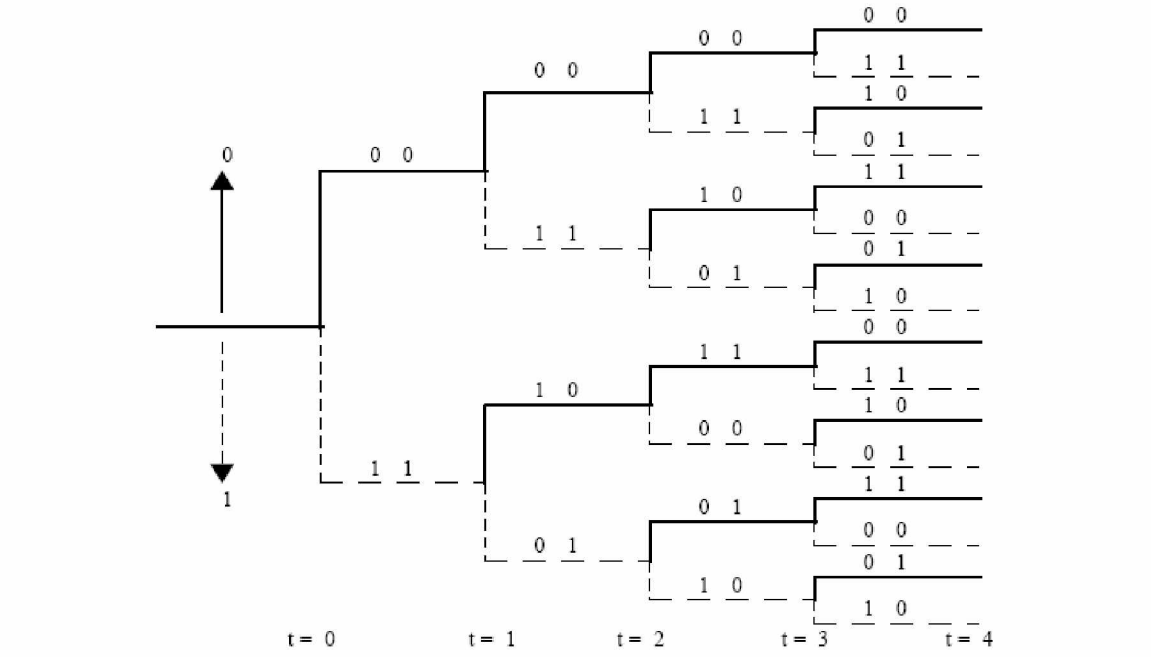


Fig3.3 Tree Diagram representation of encoder in Fig 3.2 for four input bit intervals

3.3.3 State Diagram Representation

The state diagram shows the state information of a convolutional encoder. The state information of a convolutional encoder is stored in the shift registers. Figure 3.4 shows the state diagram of the encoder in Figure 3.2.

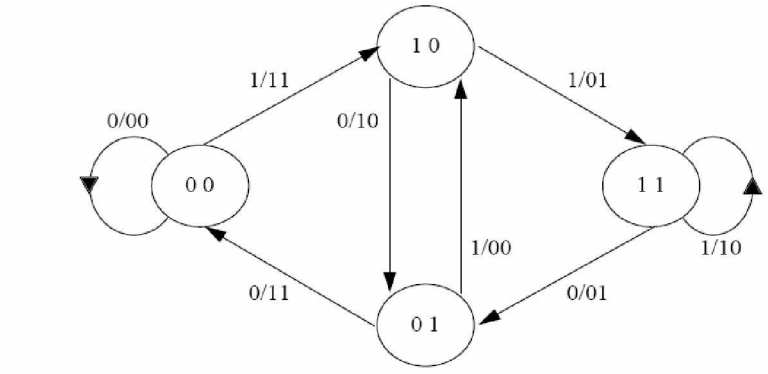


Fig 3.4 State diagram representation of encoder in Fig 3.2.

In the state diagram, the state information of the encoder is shown in the circles. Each new input information bit causes a transition from one state to another. The path information between the states, denoted as x/**c**.

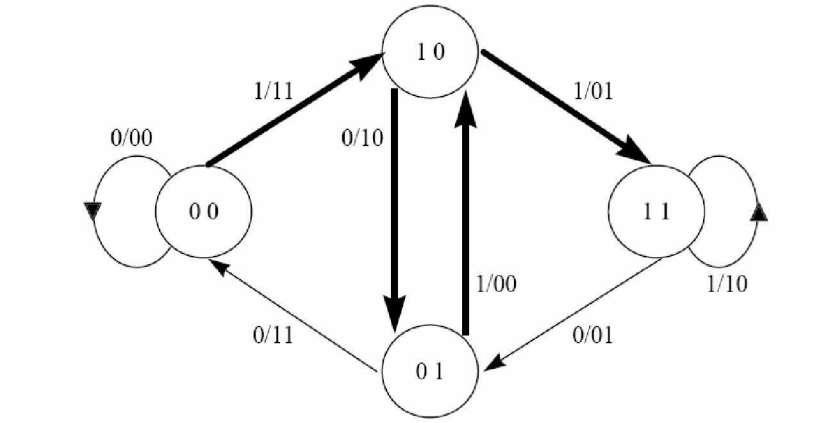


Fig 3.5 The state transitions (path) for input information sequence{1011}.

For example, the input information sequence **x**={1011} leads to the state transition sequence s={10, 01, 10, 11} and produces the output encoded sequence c={11,10,00,01}. Figure 3.5 shows the path taken through the state diagram for the given example

3.3.4 Trellis Diagram Representation

The trellis diagram is basically a redrawing of the state diagram. It shows all possible state transitions at each time step. Frequently, a legend accompanies the trellis diagram to show the state transitions and the corresponding input and output bit mappings (x/**c**). This compact representation is very helpful for decoding convolutional codes as discussed later.

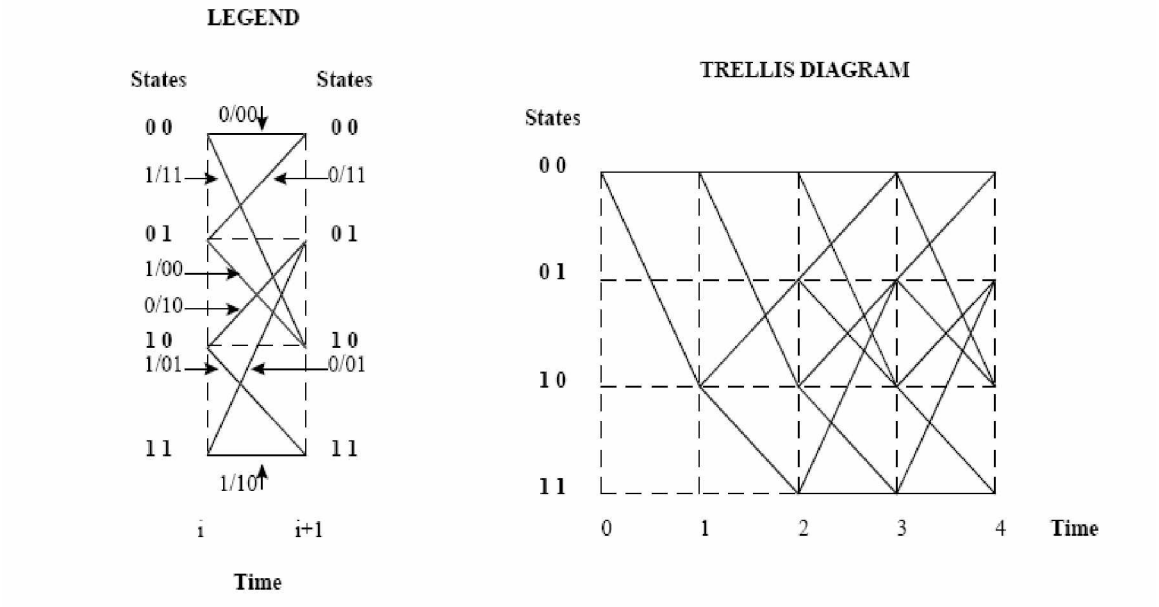


Fig 3.6 Trellis diagram representation of encoder in Fig 3.2 for four input bit intervals

3.4 HARD-DECISION AND SOFT-DECISION DECODING:

Hard-decision and soft-decision decoding refer to the type of quantization used on the received bits. Hard-decision decoding uses 1-bit quantization on the received channel values. Soft-decision decoding uses multi-bit quantization on the received channel values.

For the ideal soft-decision decoding (infinite-bit quantization), the received channel values are directly used in the channel decoder. Figure 3.7 shows hard- and soft- decision decoding.

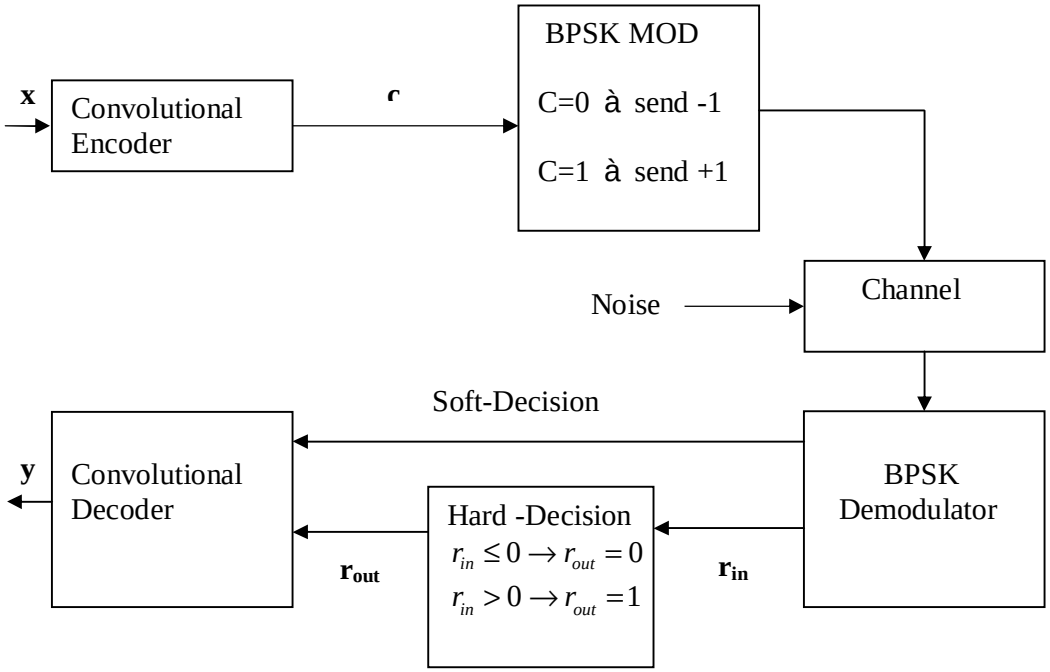


Fig 3.7 hard- and Soft-decision decoding

3.5 HARD-DECISION VITERBI ALGORITHM:

For a convolutional code, the input sequence **x** is "convoluted" to the encoded sequence **c.** Sequence **c** is transmitted across a noisy channel and the received sequence **r** is obtained. The Viterbi algorithm computes a maximum likelihood (ML) estimate on the estimated code sequence **y** from the received sequence **r** such that it maximizes the probability p(**r**|**y**) that sequence **r** is received conditioned on the estimated code sequence **y.** Sequence **y** must be one of the allowable code sequences and cannot be any arbitrary sequence. Figure 2.10 shows the described system structure.

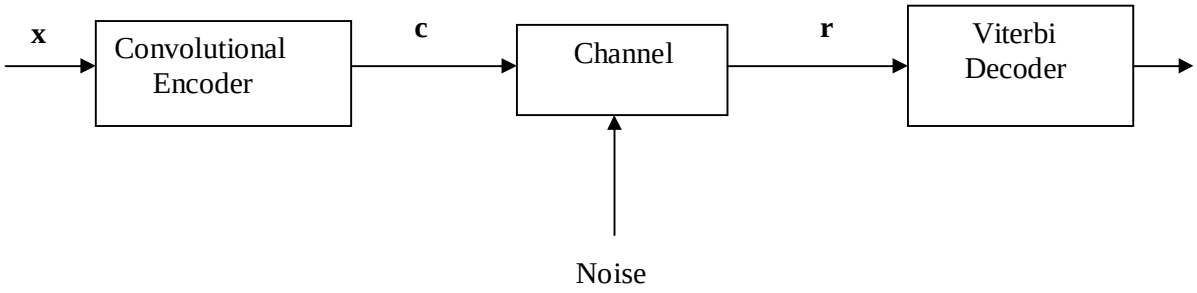


Fig 3.8 Convolutional code system.

For a rate r convolutional code, the encoder inputs k bits in parallel and outputs n bits in parallel at each time step. The input sequence is denoted as

$$\mathbf{x}=(x_0^{(1)}, x_0^{(2)}, ..., x_0^{(k)}, x_1^{(1)}, ..., x_1^{(k)}, x_{L+m-1}^{(1)}, ..., x_{L+m-1}^{(k)}) \quad (2.3)$$

and the coded sequence is denoted as

$$\mathbf{c}=(c_0^{(1)}, c_0^{(2)}, ..., c_0^{(n)}, c_1^{(1)}, ..., c_1^{(n)}, c_{L+m-1}^{(1)}, ..., c_{L+m-1}^{(n)}) \quad (2.4)$$

where L denotes the length of input information sequence and m denotes the maximum length of the shift registers.

Additional m zero bits are required at the tail of the information sequence to take the convolutional encoder back to the all-zero state. It is required that the encoder start and end at the all-zero state. The subscript denotes the time index while the superscript denotes the bit within a particular input k-bit or output n- bit block. The received and estimated sequences **r** and **y** can be described similarly as

$$\mathbf{r}=(r_0^{(1)}, r_0^{(2)}, ..., r_0^{(n)}, r_1^{(1)}, ..., r_1^{(n)}, r_{L+m-1}^{(1)}, ..., r_{L+m-1}^{(n)}) \quad (2.5)$$

and

$$\mathbf{y}=(y_0^{(1)}, y_0^{(2)}, ..., y_0^{(n)}, y_1^{(1)}, ..., y_1^{(n)}, y_{L+m-1}^{(1)}, ..., y_{L+m-1}^{(n)}). \quad (2.6)$$

For ML decoding, the Viterbi algorithm selects **y** to maximize $p(\mathbf{r}|\mathbf{y})$. The channel is assumed to be memoryless, and thus the noise process affecting a received bit is independent from the noise process affecting all of the other received bits.

The Viterbi algorithm utilizes the trellis diagram to compute the path metrics. Each state (node) in the trellis diagram is assigned a value, the partial path metric. The partial path metric is determined From state s = 0 at time t = 0 to a particular state s = k at time $t \geq 0$. At each state, the "best" partial path metric is chosen from the paths terminated at that state. The "best" partial path metric may be either the larger or smaller metric, depending whether *a* and *b* are chosen conventionally or alternatively.

The selected metric represents the survivor path and the remaining metrics represent the nonsurvivor paths. The survivor paths are stored while the nonsurvivor paths are discarded in the trellis diagram. The Viterbi algorithm selects the single survivor path left at the end of the process as the ML path. Trace-back of the ML path on the trellis diagram would then provide the ML decoded sequence. The hard-decision Viterbi algorithm (HDVA) can be implemented as follows :

$S_{k,t}$ is the state in the trellis diagram that corresponds to state S_k at time t. Every state in the trellis is assigned a value denoted $V(S_{k,t})$.

1. (a) Initialize time t = 0.

   (b) Initialize $V(S_{0,0}) = 0$ and all other $V(S_{k,t}) = +\infty$.

2. (a) Set time t = t+1.

   (b) Compute the partial path metrics for all paths going to state S_k at time t.

3. (a) Set $V(S_{k,t})$ to the "best" partial path metric going to state S_k at time t. Conventionally, the "best" partial path metric is the partial path metric with the smallest value.

   (b) If there is a tie for the "best" partial path metric, then any one of the tied partial path metric may be chosen.

4. Store the "best" partial path metric and its associated survivor bit and state paths.

5. if t < L+m-1, return to Step 2.

The result of the Viterbi algorithm is a unique trellis path that corresponds to the ML codeword.

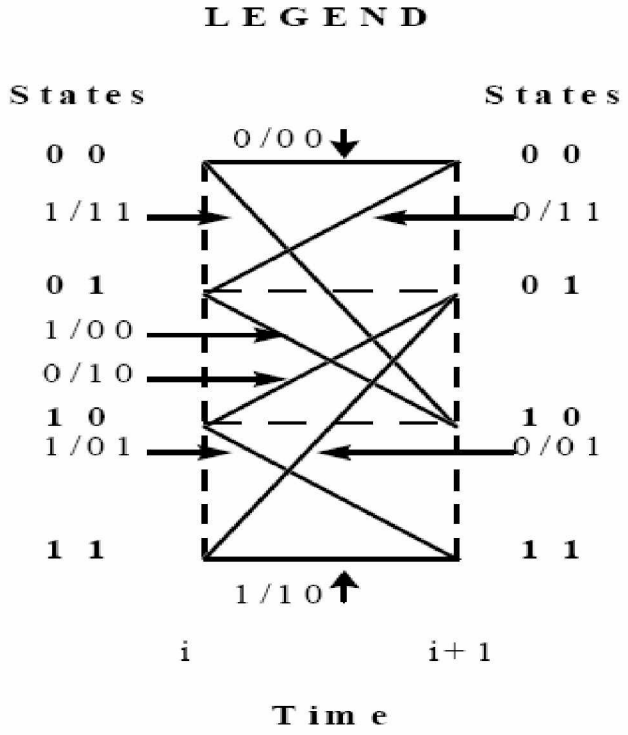


Fig 3.9 The state Transition diagram (trellis legend) of the example convolutional encoder.

A simple HDVA decoding example is shown below. The convolutional encoder used is shown in Figure 2.2. The input sequence is **x**={1010100}, where the last two bits are used to return the encoder to the all-zero state. The coded sequence is **c**={11, 10, 00, 10, 00, 10, 11}. However, the received sequence **r**={10, 10, 00, 10, 00, 10, 11} has a bit error (underlined). Figure 3.9 shows the state transition diagram (trellis legend) of the example convolutional encoder.

From the trellis diagram in Figure 2.9, the estimated code sequence is **y**={11, 10, 00, 10,00, 10, 11} which is the code sequence **c**. Utilizing the state transition diagram in Figure 2.12, the estimated information sequence is **x'**={1010100}.

3.6 SOFT-DECISION VITERBI ALGORITHM:

There are two general methods of implementing a soft-decision Viterbi algorithm. The first method (Method 1) uses Euclidean distance metric instead of Hamming distance metric. The received bits used in the Euclidean distance metric are processed by multi-bit quantization. The second method (Method 2) uses a correlation metric where its received bits used in this metric are also processed by multi-bit quantization.

The soft-decision Viterbi algorithm (SDVA1) can be implemented as follows: $S_{k,t}$ is the state in the trellis diagram that corresponds to state S_k at time t. Every state in the trellis is assigned a value denoted $V(S_{k,t})$.

1. (a) Initialize time t = 0.
   (b) Initialize $V(S_{0,0}) = 0$ and all other $V(S_{k,t}) = +\infty$.
2. (a) Set time t = t+1.
   (b) Compute the partial path metrics for all paths going to state S_k at time t.
3. (a) Set $V(S_{k,t})$ to the "best" partial path metric going to state S_k at time t.

Conventionally, the "best" partial path metric is the partial path metric with the smallest value.

   (b) If there is a tie for the "best" partial path metric, then any one of the tied partial path metric may be chosen.
4. Store the "best" partial path metric and its associated survivor bit and state paths.
5. If t < L+m-1, return to Step 2.

3.7 PERFORMANCE ANALYSIS OF CONVOLUTIONAL CODE:

The performance of convolutional codes can be quantified through analytical means or by computer simulation. The analytical approach is based on the transfer function of the convolutional code which is obtained from the state diagram. The process of obtaining the transfer function and other related performance measures are described below.

3.7.1 Transfer Function of Convolutional Code

The analysis of convolutional codes is generally difficult to perform because traditional algebraic and combinatorial techniques cannot be applied. These heuristically constructed codes can be analyzed through their transfer functions. By utilizing the state diagram, the transfer function can be obtained. With the transfer function, code properties such as distance properties and the error rate performance can be easily calculated. To obtain the transfer function, the following rules are applied:

1. Break the all-zero (initial) state of the state diagram into a start state and an end state. This will be called the modified state diagram.
2. For every branch of the modified state diagram, assign the symbol D with its exponent equal to the Hamming weight of the output bits.
3. For every branch of the modified state diagram, assign the symbol J.

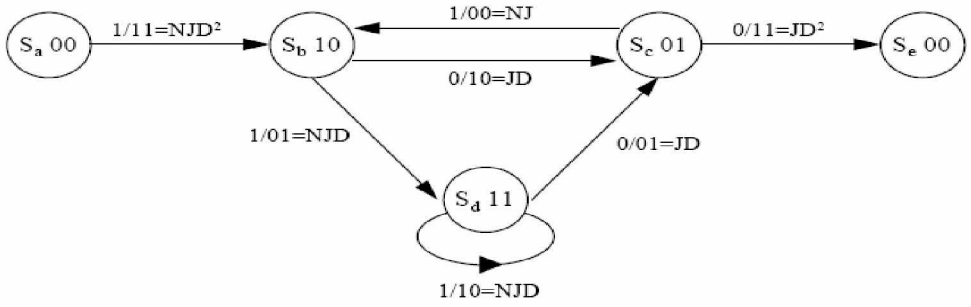


Fig 3.10 The modified state diagram of Fig 2.4

4. Assign the symbol N to the branch of the modified state diagram, if the branch transition is caused by an input bit 1.

For the state diagram in Figure 3.4, the modified state diagram is shown in Figure 3.10.

Nodal equations are obtained for all the states except for the start state in Figure 3.14. These results are

$$S_b = NJD^2 S_a + NJS_c$$
$$S_c = JDS_b + JDS_d$$
$$S_d = NJDS_b + NJDS_d$$
$$S_e = JD^2 S_c$$

The transfer function is defined to be

$$T(D,N,J) = \frac{S_e(D,N,J)}{S_s(D,N,J)}$$

By substituting and rearranging,

$$T(D,N,J) = \frac{NJ^3D^5}{1-(NJ+NJ^2)D}$$

$$= NJ^3D^5 + (N^2J^4 + N^2J^5)D^6 + (N^3J^5 + 2N^3J^6 + N^3J^7)D^7 + ...$$

(Expanded polynomial form).

3.7.2 Degree of Quantization

For soft-decision Viterbi decoding, the degree of the quantization on the received signal can affect the decoder performance. The performance of the Viterbi decoder improves with higher bit quantization. It has been found that an eight-level quantizer degrades the performance only slightly with respect to the infinite bit quantized case

3.7.3 Decoding Complexity for Convolutional Codes

For a general convolutional code, the input information sequence contains k*L bits where k is the number of parallel information bits at one time interval and L is the number of time intervals. This results in L+m stages in the trellis diagram. There are exactly 2_{k*L} distinct paths in the trellis diagram, and as a result, an exhaustive search for the ML sequence would have a computational complexity on the order of $O[2_{k*L}]$. The viterbi algorithm reduces this complexity by performing the ML search one stage at a time in the trellis. At each node (state) of the trellis, there are 2_k calculations. The number of nodes per stage in the trellis is 2_m. Therefore, the complexity of the Viterbi algorithm is on the order of $O[(2_k)(2_m)(L+m)]$. This significantly reduces the number of calculations required to implement the ML decoding because the number of time intervals L is now a linear factor and not an exponent factor in the complexity. However, there will be an exponential increase in complexity if either k or m increases.

4. VITERBI DECODING ALGORITHM

4.1 INTRODUCTION:

The Viterbi Algorithm (VA) was first proposed as a solution to the decoding of convolutional codes by Andrew J. Viterbi in 1967, with the idea being further developed by the same author in. It was quickly shown by Omura that the VA could be interpreted as a dynamic programming algorithm. Both Omura and Forney showed that the VA is a maximum likelihood decoder. The VA is often looked upon as minimizing the error probability by comparing the likelihoods of a set of possible state transitions that can occur, and deciding which of these has the highest probability of occurrence. A similar algorithm, known as the Stack Sequential Decoding Algorithm (SS-DA), was described by Forney in as an alternative to the VA, requiring less hardware to implement than the VA. The SSDA has been proposed as an alternative to the VA in such applications as target tracking, and high rate convolutional decoding. It can be shown though, that this algorithm is sub-optimum to the VA in that it discards some of the paths that are kept by the VA.

Since it's conception the VA has found a wide area of applications, where it has been found to be an optimum method usually out performing previous methods. The uses it has been applied to not just covers communications for which it was originally developed, but includes diverse areas such as handwritten word recognition, through to nonlinear dynamic system state estimation.

This report is in effect a review of the VA. It describes the VA and how it works, with an appropriate example of decoding corrupted convolutional codes. Extensions to the basic algorithm are also described. In section 3 some of the applications that the VA can be put to are described, including some uses in communications, recognition problems and target tracking. The area of dynamic signature verification is identified as an area requiring further research.

In this section the Viterbi Algorithm (VA) is defined, and with the help of an example, its use is examined. Some extensions to the basic algorithm are also looked at viterbi algorithm (VA) can be viewed as a solution of estimation for a finite sequence from Markov process through memoryless noise channel as illustrated in Figure 1:]

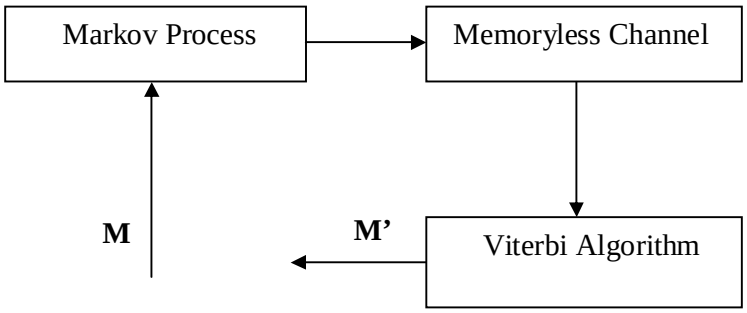


Figure 4.1: General Model for Viterbi Algorithm

Sequence detection with Viterbi decoding has been widely considered for the detection of signals with memory. It was originally invented to decode convolutional codes. So, the introduction of the Viterbi algorithm will mainly be based on the decoding process for the convolution coding.

There exist several statistical tools in VA estimation such as the Maximum A posteriori Probability (MAP) and Maximum Likelihood Sequence Estimation (MLSE).

4.2 MAP AND MLSE

MAP and MLSE can be both viewed as a derivation from the BAYES Estimation. In BAYES criterion, two notations are made:

The priori probabilities (denoted as P (H0) and P (H1))

The cost to each possible decision (denoted as Cij), i, j = 0, 1, as the cost associated with the decision D_i given that the true hypothesis is H_j. Hence, the decision rule resulting from the BAYES criterion is:

$$\frac{f_{Y/H1}(Y/H_1)}{f_{Y/H_0}(Y/H_0)} \underset{H_0}{\overset{H_1}{\gtrless}} \frac{P_0(C_{10}-C_{00})}{P_1(C_{01}-C_{11})}$$

In MAP, let the costs

Cii = 0,i= 0, 1

Cij = 1,i ≠j and i, j = 0, 1

Hence, minimizing the risk is equivalent to minimizing the probability of error. Then, the decision rule is reduced to

$$P(H_1/y) \underset{H_0}{\overset{H_1}{\gtrless}} P(H_0/y)$$

In MLSE, let

$$P_0(C_{10} - C_{00}) = P_1(C_{01} - C_{11})$$

It yields:

$$P(y/H_1) \underset{H_0}{\overset{H_1}{\gtrless}} P(y/H_0)$$

4.3 VITERBI ALGORITHM:

In this section the Viterbi Algorithm (VA) is defined, and with the help of an example, its use is examined. Some extensions to the basic algorithm are also looked at.

The VA can be simply described as an algorithm which finds the most likely path through a trellis, i.e. shortest path, given a set of observations. The trellis in this case represents a graph of a finite set of states from a Finite States Machine (FSM). Each node in this graph represents a state and each edge a possible transitions between two states at consecutive discrete time intervals. An example of a trellis is shown below in Figure 1a and the FSM that produced this trellis is shown in Figure 1b. The FSM referred to here is commonly used in digital electronics and is often refered to in the literature as a Markov Model (MM).

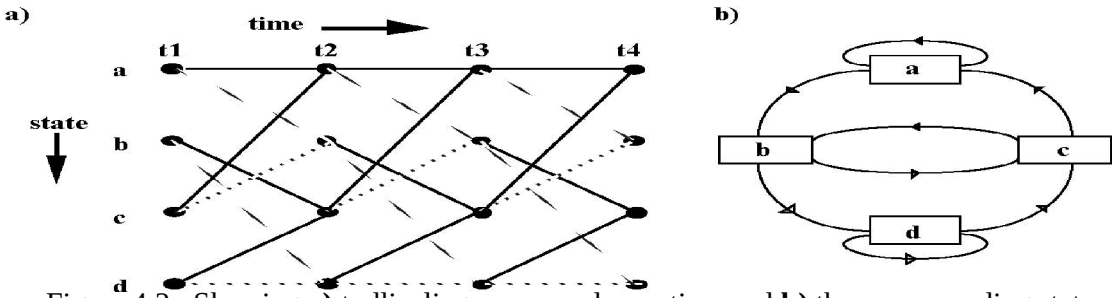


Figure 4.2: Showing **a)** trellis diagram spread over time and **b)** the corresponding state diagram of the FSM.

For each of the possible transitions within a given FSM there is a corresponding out-put symbol produced by the FSM. This data symbol does not have to be a binary digit it could instead represent a letter of the alphabet. The outputs of the FSM are viewed by the VA as a set of observation symbols with some of the original data symbols corrupted by some form of noise. This noise is usually inherent to the observation channel that the data symbols from the FSM have been transmitted along.

The trellis that the VA uses corresponds to the FSM exactly, i.e. the structure of the FSM is available, as is the case in it's use for convolutional code decoding. Another type of FSM is the Hidden Markov Model (HMM) . As the name suggests the actual FSM is hidden from the

VA and has to be viewed through the observations produced by the HMM. In this case the trellis's states and transitions are estimates of the under-lying HMM. This type of model is useful in such applications as target tracking and character recognition, where only estimates of the true state of the system can be produced. In either type of model, MM or HMM, the VA uses a set of metrics associated with the observation symbols and the transitions within the FSM. These metrics are used to cost the various paths through the trellis, and are used by the VA to decide which path is the most likely path to have been followed, given the set of observation symbols.

Before defining the VA the following set of symbols have to be defined:

t - The discrete time index.

N - Total number of states in the FSM.

x_n - The nth state of the FSM.

o_t - The observation symbol at time t, which can be one of M different symbols.

sp_{nt} - The survivor path which terminates at time t, in the nth state of the FSM.

It consists of an ordered list of x_n's visited by this path from time t = 0 to time t.

T - Truncation length of the VA, i.e. the time when a decision has to be made by the VA as to which sp_{nt} is the most likely.

π_n - Initial state metric for the nth state at t = 0. Defined as the probability that the n^{th} state is the most likely starting start, i.e. Prob(x_n at t = 0).

a_{nm} - The transition metric for the transition from state x_m at time t - 1 to the state x_n at time t. Defined as the probability that given that state x_m occurs at time t - 1, the state x_n will occur at time t, i.e. Prob(x_n at t | x_m at t - 1).

b_n - The observation metric at time t, for state x_n. Defined as the probability that the observation symbol o_t would occur at time t, given that we are in the state x_n at time t, i.e. Prob (o_t | x_n at t).

Γ_{nt} - The survivor path metric of sp_{nt}. This is defined as the Product of the metrics (π_n, a_{nm} and b_n) for each transition in the n^{th} survivor path, from time t = 0 to time t.

The equations for the model metrics, π_n, a_{nm} and b_n, can be derived mathematically where their properties result from a known application. If the metric properties are not known, re-estimation algorithms can be used, such as the Baum-Welch re-estimation algorithm, to obtain optimum probabilities for the model. It is also usual to take the natural logarithm of the metrics, so that arithmetic underflow is prevented in the VA during calculations.

The VA can now be defined:

In English the VA looks at each state at time t, and for all the transitions that lead into that state, it decides which of them was the most likely to occur, i.e. the transition with the greatest metric. If two or more transitions are found to be maximum, i.e. their metrics are the same, then one of the transitions is chosen randomly as the most likely transition. This greatest metric is then assigned to the state's survivor path metric, Γ_{nt}. The VA then discards the other transitions into that state, and appends this state to the survivor path of the state at t - 1, from where the transition originated. This then becomes the survivor path of the state being examined at time t. The same operation is carried out on all the states at time t, at which point the VA moves onto the states at t + 1 and carries out the same operations on the states there. When we reach time t = T (the truncation length), the VA determines the survivor paths as before and it also has to make a decision on which of these survivor paths is the most likely one. This is carried out by determining the survivor with the greatest metric, again if more than one survivor is the greatest, then the most likely path followed is chosen randomly. The VA then outputs this survivor path, sp_T, along with it's survivor metric, Γ_T.

4.4 EXAMPLE:

Now that the VA has been defined, the way in which it works can be looked at using an example communications application. The example chosen is that of the VA's use in convolutional code decoding, from a memoryless Binary Symetric Channel (BSC), as described in. A picture of the communications system that this example assumes is shown below in Figure 2. This consists of encoding the input sequence, transmitting the sequence over a transmission line (with possible noise) and optimal decoding the sequence by the use of the VA.

The input sequence, we shall call it I, is a sequence of binary digits which have to be transmitted along the communications channel. The convolutional encoder consists of a shift register, which shifts in a number of the bits from I at a time, and then produces a set of output bits based on logical operations carried out on parts of I in the register memory. This process is often referred to as convolutional encoding. The encoder introduces redundancy into the output code, producing more output bits than input bits shifted into it's memory. As a bit is shifted along the register it becomes part of other output symbols sent. Thus the present output bit that is observed by the VA has information about previous bits in I, so that if one of these symbols becomes corrupted then the VA can still decode the original bits in I by using information from the previous and subsequent observation symbols. A diagram of the convolutional encoder used in this ex-ample is shown in Figure 3. It is assumed here that the

shift register only shifts in one bit at a time and outputs two bits, though other combinations of input to output bits are possible.

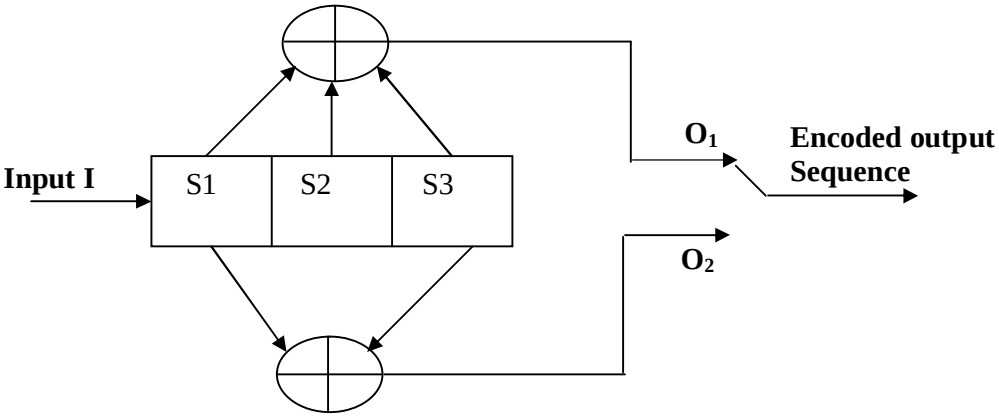
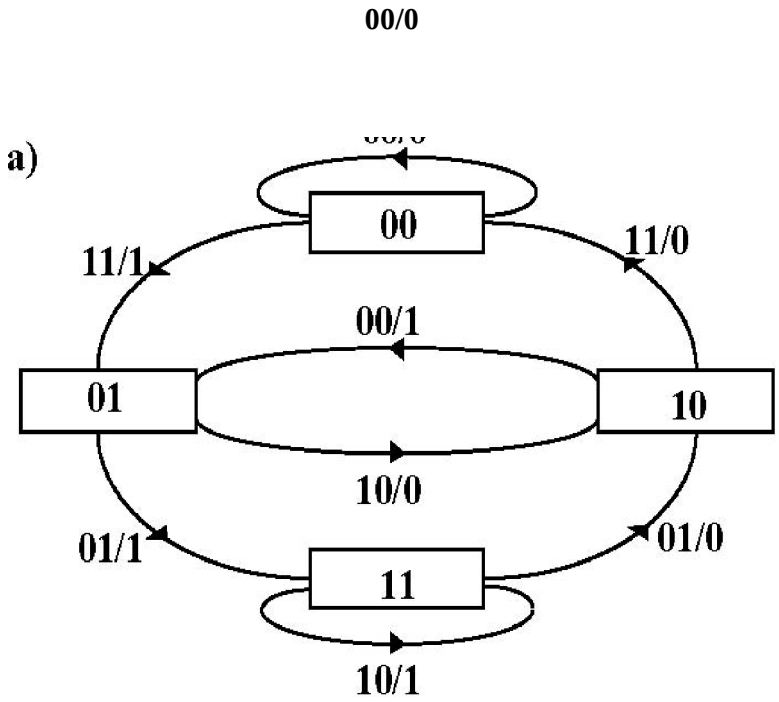


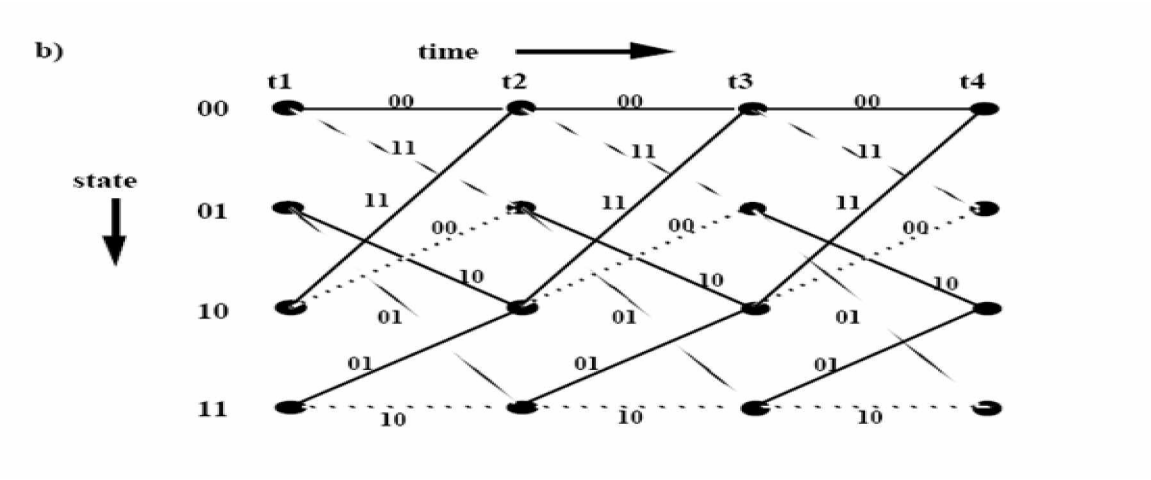
Figure 4.3. Example convolutional encoder.

This encoder can be represented by the FSM shown in Figure 4a. The boxes in this diagram represent the shift register and the contents are the state that the FSM is in. This state corresponds to the actual contents of the shift register locations S2 followed by S1, i.e. if we are in state 01, then the digit in S1 is 1 and the digit in S2 is 0. The lines with arrows represent the possible transitions between the states. These transitions are labeled as x/y, where x is a two digit binary number, which represents the output symbol sent to the communications channel for that particular transition and y represents the binary digit from I, that when shifted into the encoder causes that particular transition to occur in the state machine.The encoded sequence produced at the output of the encoder is transmitted along the channel where noise inherent to the channel can corrupt some of the bits so that what was transmitted as a 0 could be interpreted by the receiver as a 1, and vice versa.

These observed noisy symbols are then used along with a trellis diagram of the known FSM to reconstruct the original data sequence sent. In our example the trellis diagram used by the VA is shown in Figure 4b. This shows the states as the nodes which are fixed as time progresses. The possible transitions are shown as grey lines, if they were caused by a 1 entering the encoder, and the black lines, if they were caused by a 0 entering the encoder. The corresponding outputs that should of been produced by the encoder are shown, by the two bit binary digits next to the transition that caused them. As can be seen in Figure 4b the possible transitions and states remain fixed between differing time intervals. The trellis diagram of Figure 4b can be simplified to show the recursive nature of the trellis, as is shown in Figure 4c.

It was shown by Viterbi in that the log likelihood function used to determine survivor metrics can be reduced to a minimum distance measure, known as the Hamming Distance. The Hamming distance can be defined as the number of bits that are different between, between the symbol that the VA observes, and the symbol that the convolution encoder should have produced if it followed a particular input sequence. This measure defines the combined measure of a_{nm} and b_n for each transition in the trellis. The π_n's are usually set before decoding begins such that the normal start state of the encoder has a $\pi_n = 0$ and the other states in the trellis have a π_n whose value is as large as possible, preferably ∞. In this example the start state of the encoder is always assumed to be state 00, so $\pi_0 = 0$, and the other π_n's are set to 100.





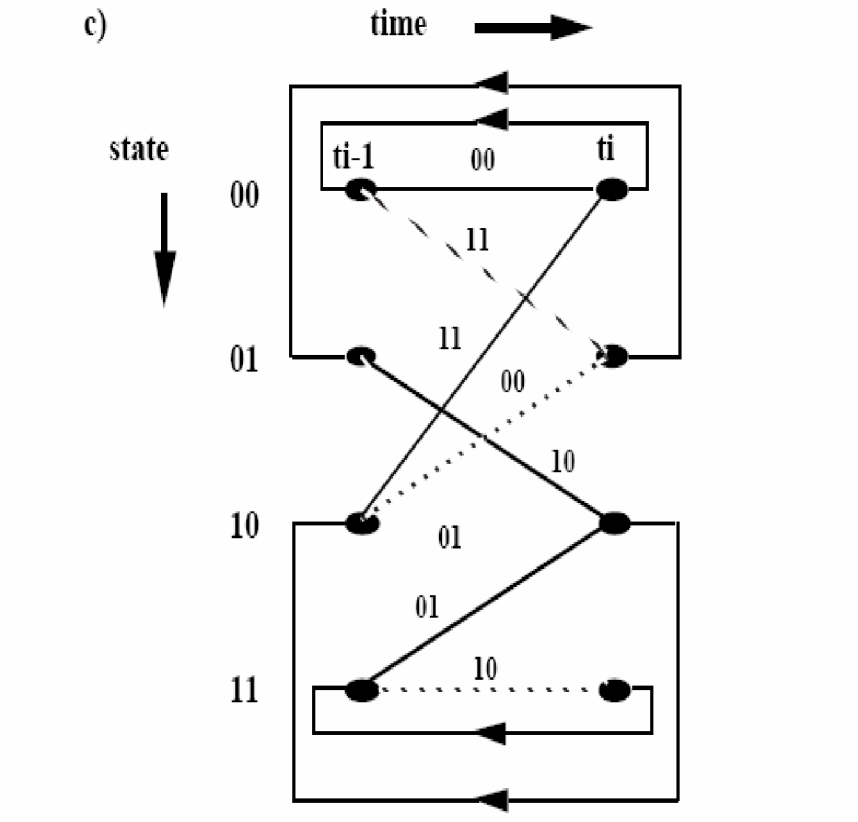


Figure 4.4. Showing the various aspects of the FSM in the example. a) shows the FSM diagram, b) the trellis diagram for this FSM spread over time and c) shows the recursive structure of this trellis diagram.

As an example if it is assumed that an input sequence I, of 0 1 1 0 0 0 is to be transmitted

across the BSC, using the convolutional encoder described above, then the out-put obtained from the encoder will be 00 11 01 01 11 00, as shown in Table 1. The output is termed as the Encoder Output Sequence (EOS). Table 1 also shows the corresponding contents of each memory element of the shift register, where each element is assumed to be initialized to zero's at the start of encoding. As the EOS is constructed by the encoder, the part of the EOS already formed is transmitted across the channel. At the receiving end of the channel the following noisy sequence of bits may be received, 01 11 01 00 11 00. As can be seen there are two bit errors in this sequence, the 00 at the beginning has changed to 01, and similarly the fourth symbol has changed to 00 from 01. It is the job of the Viterbi Algorithm to find the most likely set of states visited by the original FSM and thus determine the original input sequence.

Input	States			Outputs	
0	0	0	0	0	0
1	1	0	0	1	1
1	1	1	0	0	1
0	0	1	1	0	1

Table 4.1 Output of Convolutional Encoder.

For simplicity the four states of the encoder are assigned a letter such that a = 00, b = 01, c = 10 and d = 11. At the start of the decoding process, at t = 1, the π_n's are first of all assigned to each of the corresponding $\Gamma n0$, so $\Gamma_{a0} = 0$ and $\Gamma_{b0} = \Gamma_{c0} = \Gamma_{d0} = 100$. The hamming distance for each transition is then worked out, e.g. at t = 1 the symbol observed by the VA was 01, so the hamming distance for the transition from a at time 0 to a at time 1 is 1.

Once this is done the VA then finds the total metric for each transition, e.g. for state a at t =1 there are two possible transitions into it, one from c with a total metric of 101 and one from a with a total metric of 1. This is shown in Figure 5a for all the states at time t =1, showing the π_n's for each state after the state letter. The VA then selects the best transition into each state, in the case of a this would be the transition from a at t = 0, since it's metric is the minimum of the two transitions converging into a at t = 1. Then the VA sets the sp_{n1} and Γ_{n1} for each state, so $sp_{a1} = \{a,a\}$ and $\Gamma_{a1} = 1$. The survivor paths and the corresponding survivor path lengths are shown in Figure 5b at t = 1. Then the VA repeats the above process for each time step and

the decoding and survivor stages are shown in Figure 5c and 5d for t = 2, and the same stages for t = 6.

When the VA reaches time T = t, then it has to decide which of the survivor paths is the most likely one, i.e. the path with the smallest Hamming Distance. In this example T is assumed to be 6, so the path terminating at state a in Figure 5f has the minimum Hamming distance, 2, making this the most likely path taken. Next, the VA would start outputing the estimated sequence of binary digits that it thinks were part of the original input sequence. It is found that the estimated sequence is 0 1 1 0 0 0 which corresponds directly to the original input sequence. Thus the input sequence has been recovered dispite the error introduced during transmission.

4.5 ALGORITHM EXTENSIONS:

Now that the VA has been examined in some detail, various aspects of the algorithm are now looked at so that a viable research area can be established. In this section, some of the possible extensions to the algorithm are looked at though these are limited in scope.

In the example above the VA relies on inputs from a demodulator which makes hard decisions on the symbols it received from the channel. That is, whatever type of mod-ulation used, be it phase, frequency or amplitude modulation, then the demodulator has to make a firm decision whether a 0 or 1 was transmitted. One obvious extension of the VA is that of replacing this firm decision with a soft-decision. In this method the demodulator produces soft outputs, i.e. each symbol produced by the demodulator in-stead of consisting of a 0 or 1 consists of the symbol that the demodulator thinks was sent along with other bits which represent the confidence that the symbol was transmit-ted. This increases the information presented to the VA increasing it's performance. The VA can then be used to decode these soft decisions and output a hard decision as before.

The next step up from this is a Viterbi algorithm which produces soft output decisions of it's own- this is known as a Soft Output Viterbi Algorithm (SOVA). In this version of the VA a hard decision is given on the most likely survivor path, but information about how confident that each symbol in the path occured is also produced.

Another extension to the algorithm was suggested recently, by Bouloutas *et a*. Bouloutas's extension generalizes the VA so that it can correct insertions and deletions in the set of observations it receives, as well as symbol changes. This method combines the known FSM, that produced the symbols in the first place, such as in the example above, with an FSM of the observation sequence. A trellis diagram is produced, known as a product trellis diagram,

which compensates for insertions and deletions in the observation sequence. For each of the insertion, deletion and change operations a metric is assigned, which also depends upon the application the VA is being applied to. The VA produces the most likely path through the states of the FSM, estimates of the original data sequence, as well as the best sequence of operations performed on the data to obtain the incorrect observation sequence. An application of this VA, is in the use of correcting programming code whilst compiling programs, since many of the errors produced while writing code tend to be characters missed out, or inserted characters. An-other extension to the VA is that of a parallel version. Possible solutions to this have been suggested by Fettweis and Meyr, and Lin *et al*. The parallel versions of the VA have risen out of the needs for fast hardware implementations of the VA.

4.6 APPLICATIONS:

This section looks at the applications that the VA has been applied to. The application of the VA in communications is initially examined, since this is what the algorithm was initially developed for. The use of the VA in target tracking is also looked at and finally recognition problems. In all the applications, that the VA has been applied to since it's conception, the VA has been used to determine the most likely path through a trellis, as discussed in section 2. What makes the VA an interesting research area is that the metrics, (π_n, a_{nm} and b_n), used to determine the most likely sequence of states, are application specific. It should be noted that the VA is not limited to the areas of application mentioned in this section though only one other use is known to this author. This is the application of the VA in digital magnetic recording systems. Though this section sums up the uses of the VA there are probably a number of other application areas that could be identified.

3.6.1 Communications.

A number of the uses of the VA in communications have already been covered in sec-tion 2. These include the first proposed use of the VA in decoding convolutional codes. It was shown that the VA could be used to combat Intersymbol Interference (ISI), by Forney in. ISI usually occurs in modulation systems where consecutive signals disperse and run into each other causing the filter at the demodulator to either miss a signal, or wrongly detect a signal. ISI can also be introduced by imperfections in the filter used. Forney suggested that the VA could be used to estimate the most likely sequence of 0's and 1’s that entered the modulator at the transmitter end of the channel, given the sequence of ISI affected observation symbols. So if a convolutional code was used for error control then the output from this VA would be passed through another VA to obtain the original input sequence into the transmitter. The VA

used in this situation is more commonly referred to as a Viterbi Equalizer and has been used in conjunction with Trellis-Coded Modulation (TCM) and also on different channel types. The advantage of using a VA in the equalizer is that the SOVA, described in section 2, can be used to pass on more information into the decoder.

Another application of the VA in communications is that of decoding TCM codes. This method of encoding was presented by Ungerboeck, for applications where the redundancy introduced by convolutional encoding could not be tolerated because of the reduction in data transmission rate or limited bandwidth. This method combines the error correcting abilities of a trellis code with the redundancy which can be introduced into the modulation signal itself via multiple amplitude levels or multiple phases. Instead of transmitting the extra redundant bits produced by the convolutional encoder for error control, these bits are mapped onto different amplitudes or phases in the modulation set which ensures that the bandwidth of the signal does not have to be increased. The VA is then used to decode the TCM codes, and produce the most likely set of bits as in convolutional decoding. It should also be noted that the decisions coming from the demodulator are in fact soft decisions, and a soft-decision VA has to be used in the de-coding.Much of the research carried out in the use of the VA in communications has been directed into finding better performance TCM codes and in the application of the VA as an equalizer in different communication environments. It was therefore decided to look for another application that the VA could be applied too.

4.6.2 Target Tracking.

The use of the VA in the field of Target Tracking is investigated in this section. Work in this application area has been carried out to date using Kalman Filters for the tracking of targets. The basic aim is to take observations from radar, sonar or some other form of detector and to estimate the actual position of the target in relation to the detector. In an ideal world this would be a simple task since the detector should give us the true position of the target, but in reality various problems arise which affect the readings from the detector. These usually from noise, be it background, signal deteoriation or due to imperfections in the detector.

There is also the matter of manoeuvrering by the target which typically results in the modelling of a non-linear system. Another type of noise that can be introduced into the signals used by the detector is random interference.

Background Position

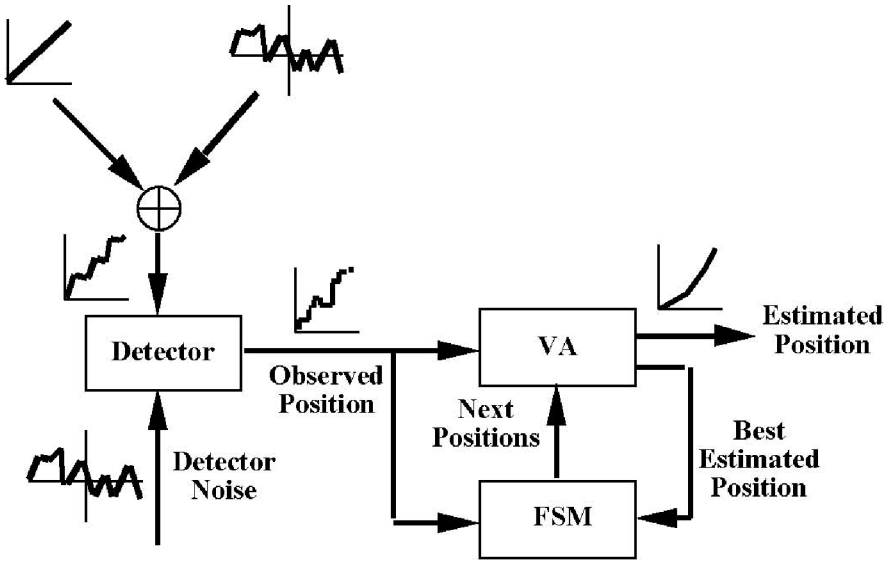


Fig 4.5 Model of Tracking system.

These problems were tackled by Demirbas in where the VA was used to produce estimates of a target's state, be it in terms of speed, position or acceleration. The method uses an FSM to construct the trellis diagram, both the next states and the transitions between these states, using some a motion model. The motion model used is a approximation of non-linear motion. This recursive model is used to produce the next set of possible states that the object could go into along with the transitions into these states. A model of the tracking system proposed by Demirbas is shown in Figure 6 above.

Each state of the trellis represents a n-dimensional vector, for example a specific range, bearing and elevation position of a plane. Unlike the Kalman filter, this method does not use a linear motion model, but a non-linear one. It is also noticed that the number of states produced at the next stage in the trellis can increase or decrease, unlike most other applications of the VA where the number of states is fixed throughout the trellis. The VA can only consider a small amount of possible states that a target can move into, since in theory we can have a very large or even an infinite amount of possible states. The a_{nm} and π_n metrics can be worked out whilst the trellis diagram is being constructed and the b_n metrics depend on whether we are tracking the target in the presence of interference or with normal background noise. The VA is then used to estimate the most likely path taken through this trellis using the observation sequence produced by the detector. It was found that this method is far superior to the Kalman filter at estimating non-linear motion and it is comparable in

performance to the Kalman filter when estimating linear motion. Also considered by Demirbas in is the use of the Stack Sequential Decoding Algorithm instead of the VA, to produce fast estimates. Though this method is sub-optimum to the one using the VA.

This model can be applied to any non-linear dynamic system such as population growths, or economics, but Demirbas has applied this tracking method to tracking manoeuvrable targets using a radar system. In these papers, Demirbas adapts the above system so that instead of a state in a trellis consisting of a position estimate in all dimensions, (range, bearing and elevation in this case), he splits these up into separate components and each of these components is estimated separately. So each of the dimensions has it's own trellis for estimation. The motion models for each of the trellises needed in this system are also presented in these papers and it is noted that the VA has to produce an estimate at each point so that the trellis diagrams can be extended.

The model can been taken one step further by accounting for missing observations,
e.g when a plane goes out of radar range momentarily. This was dealt with by Demirbas in where interpolating functions were used to determine the missing observations from the observations received. This set of observations was then given to the VA to estimate the true positions. Another adoption proposed in uses the VA to estimate the position of a plane when any single observation received depends upon a number of previous observations, as in ISI.

Another tracking method has been developed by Streit and Barrett where the VA is used to estimate the frequency of a signal, using the Fast Fourier Transform of the signal received. Unlike Demirbas, Streit uses a HMM tracker where the trellis is fixed and constructed from an observation model not a motion model.The states of the HMM represent a certain frequency. This method of tracking can be used to track changing frequency signals such as those used by some radars to detect where a target is.

All through Demirbas's work and Streit's work several improvements come to mind. One is the adaption of the VA so that it can deal with more than one target's position being estimated at the same time, i.e. multiple target tracking. Another improvement, particularly in Demirbas's work, is the use of an adaptive maneuvering model. Demirbas assumes in all his work that the manoeuvre is known to the tracker, but in reality this parameter would not be known at all, so it would have to be estimated as is done for the Kalman filter. This maneuvering parameter could be represented by a random variable, or it can be estimated by using a similar estimation scheme for the target's position, or estimated as suggested in. Another problem not considered by Demirbas is adapting the noise models along with the finite state model. Though the tracking model given in Streit's work could be trained.

The problem of multiple targets tracking using the VA has been solved for the tracking of time-varying frequency signals by Xie and Eval. They describe a multitrack VA which is used to track two crossing targets. This system is an extension to the frequency line tracking method presented in. Though this is not the same type of tracking as mentioned in Demirbas's work, it should be a simple matter of altering the parameters of the model to fit this type of tracking.

4.6.3 Recognition.

Another area where the VA could be applied is that of character and word recognition of printed and handwritten words. This has many applications such as post code and ad-dress recognition, document analysis, car licence plate recognition and even direct in-put into a computer using a pen. Indeed the idea of using the VA for optical character reading (OCR) was suggest by Forney in. Also, Kunda *et al* used the VA to select the most likely sequence of letters that form a handwritten English word. Another advantage with this model is that if a word produced by the VA is not in the system dictionary then the VA can produce the other less likely sequence of letters, along with their metrics, so that a higher syntactic/semantic model could determine the word produced. It can be easierly seen that a similar method would apply to the determination of a sequence of letters of a printed character as in OCR. In fact the VA can be used to recognize the individual characters or letters that make up a word. This is dealt with in for Chinese character recognition, though a similar method could be used to recognize English letters. The VA could be used at an even lower level of processing in the recognition phase than this, where it could be applied to character segmentation, i.e. determining which area of the paper is background, and which area contains a part of a letter. It was noted by the authors of that the use of the VA and HMM in character recognition has not been widely investigated thus making this area an interesting one for further research. The VA has also been used in other areas of pattern recognition, where it has been used to detect edges and carry out region segmentation of an image. Another recognition problem is that of speech recognition, which unlike character and word recognition, the VA along with HMM's have been used widely . The VA has also been used with neural networks to recognize continuous speech

4.7 VITERBI DECODER:

The convolution encoder is basically a finite-state machine, and the VA decoding is done on the optimal decoder based on MLSE. Figure 3.6 shows the block diagram in which the convolutional coding and Viterbi decoding are applied

The Viterbi decoding involves the search through the trellis for the most likely sequence. i.e., to select the path with the maximum probability of P (z|x), where z is the received sequence and x is the signal which is needed to estimate

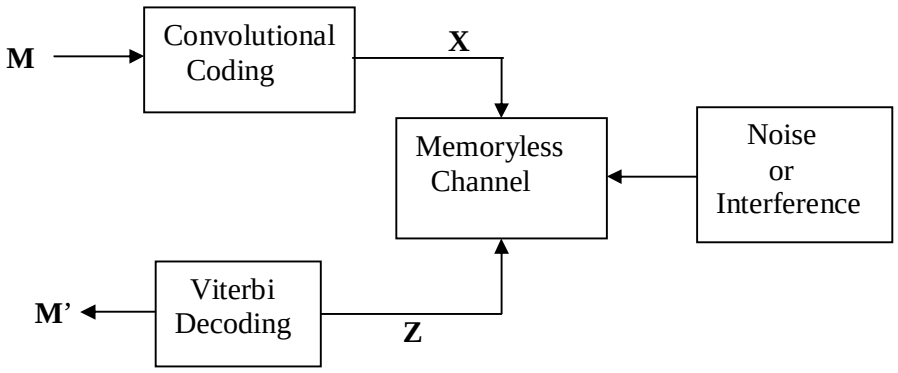


Fig.4.6 Viterbi Decoder for the convolutional coding

If the error probability in channel is P (1|0) =P (0|1) = p, L is the sequence length, and e is the number of error bits at the same time they are both positive constants.It means that finding the maximum likelihood path in trellis diagram is equivalent to finding the minimum hamming distance.

Figure3.3 (series) shows the bit sequences in the convolutional coding and decoding process. Figure 3-1 is the original signal with length of 20. Figure 3-2 is the sequence after the (2, 1, 3) convolutional coding. The sequence length is enlarged to 40 so that the distance of the signal is enlarged. Figure 3-3 is the received sequence through Additive White Gaussian Noise (AWGN) channel with Bit Error Rate (BER) of 0.1

4.7.1 Implementation of a Viterbi decoder

The major tasks in the Viterbi decoding process are as follows:

1 Quantization: Conversion of the analog inputs into digital.

2 Synchronization: Detection of the boundaries of frames and code symbols.

3 Branch metric computation.

4 State metric update: Update the state metric using the new branch metric.

5 Survivor path recording: Tag the surviving path at each node.

6 Output decision generation: Generation of the decoded output sequence based on the survivor path information. Figure 2.6 shows the flow of the Viterbi decoding Algorithm, which performs the above tasks in the specified order.

This section discusses the different parts of the Viterbi decoding process. Analog signals are quantized and converted into digital signals in the quantization block. The synchronization block detects the frame boundaries of code words and symbol boundaries. We assume that a

Viterbi decoder receives successive code symbols, in which the boundaries of the symbols and the frames have been identified.

The branch metric computation block compares the received code symbol with the expected code symbol and counts the number of differing bits

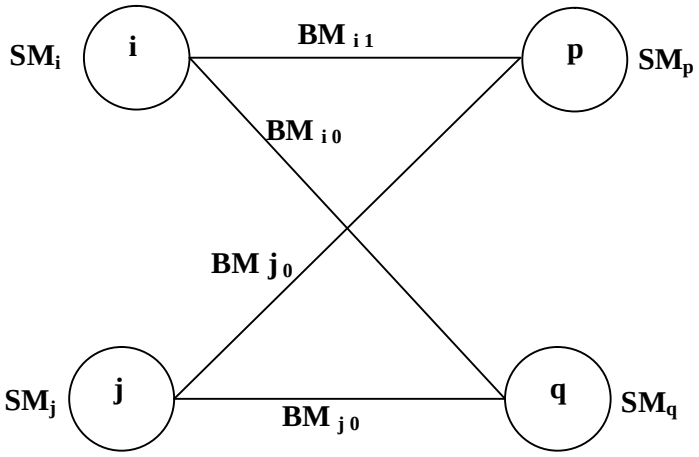


Figure 4.7 A butterfly structure for a convolutional encoder with rate 1/n

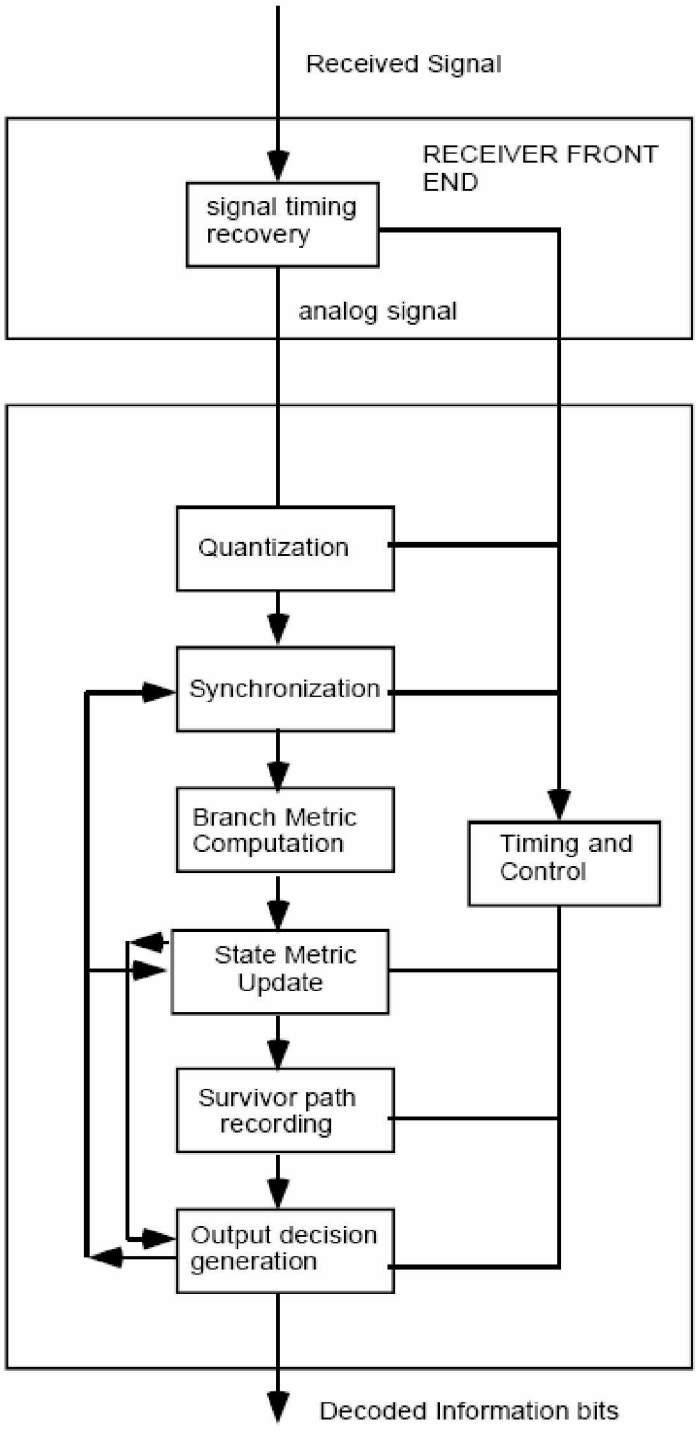


Figure 4.9 : The flow in General Viterbi Decoder.

The state metric update block selects the survivor path and updates the state metric. The trellis diagram for a rate 1/n convolutional encoder consists of butterfly structures. This structure contains a pair of origin and destination states, and four interconnecting branches.

In the Figure 4.9 the upper (lower) branch from each state I or j is taken , when

the corresponding source input bit is '1' ('0'). If the source input bit is '1' ('0'), the next state for both i or j is state p(q). the following relations in figure are established for a (n,1,m) convolutional encoder.

Notation

SM_X : state metric of a state x

BM_{XK} : branch metric from a state x under the source input k, where k Є {'0','1'}

For state I	For branch metric BM_{xx}	For state metric SM_x
$j = 2^{m-1} + i$	$BM_{i0} = n - BM_{i1}$	$SM_p = Min[(SM_i + BM_{i1}), (SM_j + BM_{j1})]$
$p = 2i + 1$	$BM_{j0} = BM_{i1}$	
$SM_q = Min[(SM_i + BM_{i0}), (SM_j + BM_{j0})]$		
q =2i	$BM_{j1} = BM_{i0}$	

Figure 4.10 The relationships of the states and branch metrics in a butterfly

It is important to note that state *p* is even and state *q* is odd. This implies that an odd (even) state is reached only if the source input bit is '0' ('1'). This property is utilized for the traceback, which is explained later. Another important point to be noted is that, it is possible to traceback from a state at a stage *t* to its previous state at the stage *t-1* provided the survivor branch of the state is the upper path or the lower path. If the survivor branch of an odd state *p* at stage *t* is the upper (lower) path, the previous state at stage *t-1* is state *i(j)*. Note that *i* is obtained as *(p-i)/2* and *j* is $2^{m-i} + i = 2^{m-i} + (p-i)/2$. Similar results can be applied to an even state. In summary, if we record whether the survivor path is the upper path or the lower path, we can traceback from the final state to the initial state.

Each butterfly wing is usually implemented by a module called Add-Compare-Select (ACS) module. An ACS module for state *p* in Figure 4.4 is shown in Figure 4.6. The two adders compute the partial path metric of each branch, the comparator compares the two partial metrics, and the selector selects an appropriate branch. The new partial path metric updates the state metric of state *p*, and the survivor path-recording block records the survivor path.

The number of necessary ACS module is equal to half the number of total states. Time sharing of some ACS modules is possible to save the hardware, but such sharing slows down the operation and dissipates more power. In this thesis we reckon replication of necessary ACS modules, which is more power efficient. Consider a trellis diagram shown in Figure 4.7. The register of state S_1 at *t=3* contains 101. Note that the trellis follows along the bold path,

and the decoded output sequence is 101. This approach eliminates the need to traceback, since the register of the final state contains the decoded output sequence. Hence, the approach may offer a high-speed operation, but it is not power efficient due to the need to copy all the registers in a stage to the next stage. We have investigated on the power efficiency of this approach

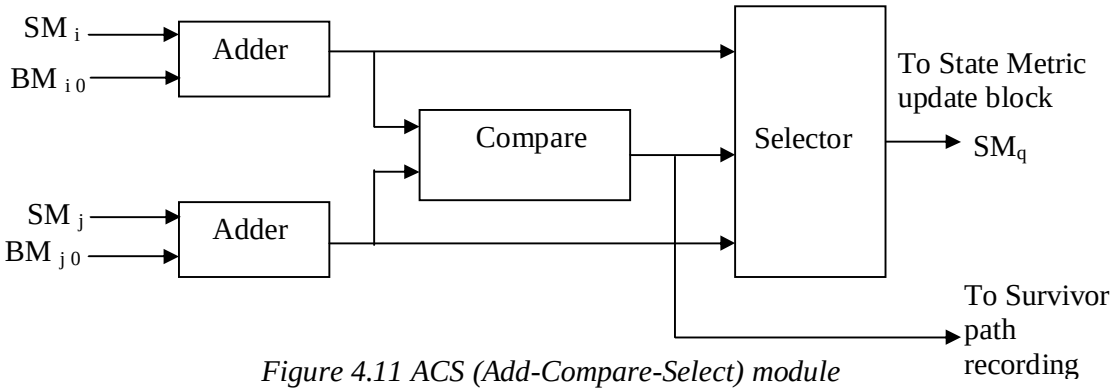


Figure 4.11 ACS (Add-Compare-Select) module

The other approach called trace back records the survivor branch of each state. As explained earlier, it is possible to trace back the survivor path provided the survivor branch of each state is known. While following the survivor path, the decoded output bit is '0' ('1') whenever it encounters an even (odd) state. A flip-flop is assigned to each state to store the survivor branch and the flip-flop records '1' ('0') if the survivor branch is the upper (lower) path. Concatenation of decoded output bits in reverse order of time forms the decoded output sequence.

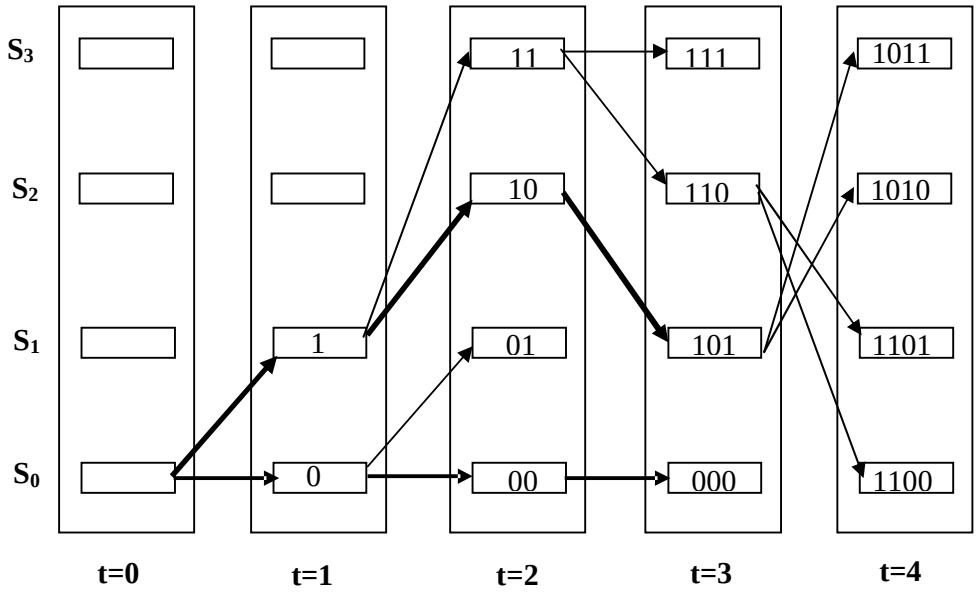
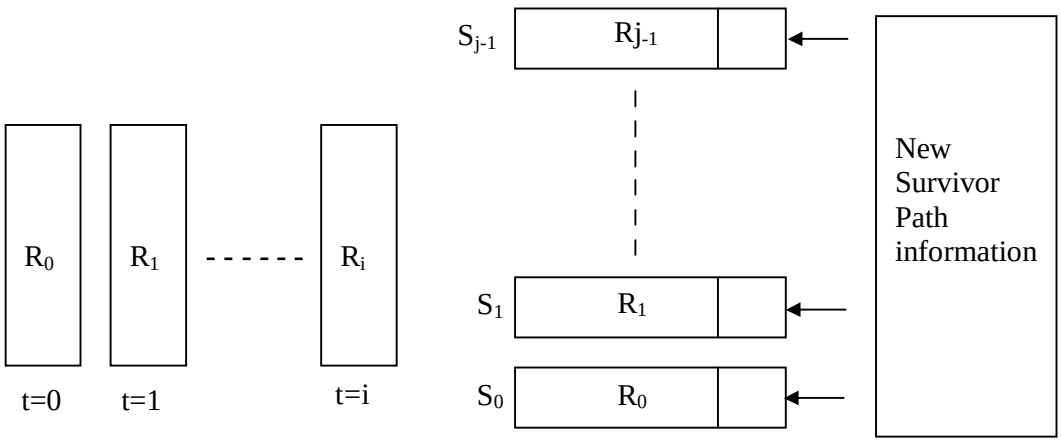


Figure 4.12 Register-exchange information generation method

It is possible to form registers by collecting the flip-flops in the vertical direction or in the horizontal direction as shown in Figure 2.12. When a register is formed in vertical direction, it is referred to as “selective update” in this thesis. When a register is formed in horizontal direction, it is referred to as “shift update”.

In selective update, the survivor path information is filled from the left register to the right register as the time progresses. In contrast, survivor path information is applied to the least significant bits of all the registers in “shift update”. Then all the registers perform a shift left operation. Hence, each register in the shift update method fills in survivor path information from the least significant bit toward the most significant bit. Figure 2.13 shows a selective update in the traceback approach.

The shift update is more complicated than the selective update. The shift update is described in, and the selective update is proposed by us to improve the shift update. In chapter 4, we show that the selective update is more efficient in power dissipation and requires less area than the shift update. Due to the need to traceback, the traceback approach is slower than the register-exchange approach.



(a) Selective update method *(b) Shift update method*

Figure 4.13:Two options for forming Registers

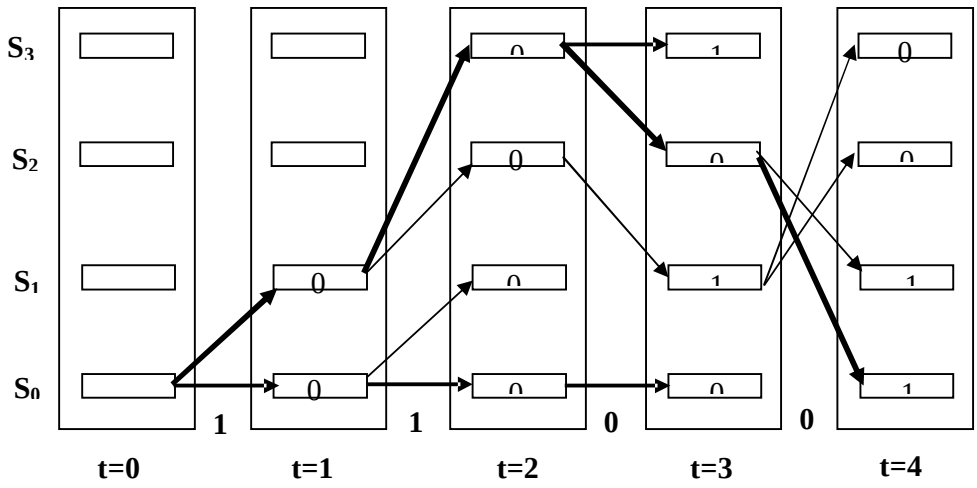


Figure 4.14 Selective update in the traceback approach

5. PASS BAND MODULATION

5.1 BASE-BAND AND PASS-BAND SIGNALLING:

For baseband signaling the waveform only contains frequencies $|f| \leq B$. In practice, most of the power is localised in this range and there is negligible power at higher frequencies. PAM is an example of baseband signaling. These signals are transmitted directly over a low-pass channel. Some pulse shaping is necessary to minimize Inter Symbol Interference (ISI).

For passband signaling the signal power is concentrated in a band centred on a carrier frequency i.e. $|f - f_c| \leq B$. Only negligible signal power exists outside this range. This is generally achieved by modulating a baseband signal with a carrier of frequency f_c, usually sinusoidal. Systems are generally designed to minimize the probability of symbol error in the presence of noise.

We will make the following assumptions about the transmission channel:

1. The channel is linear with bandwidth wide enough to accommodate the transmission of the modulated signal with almost zero distortion.
2. The transmitted signal is perturbed by an additive, zero-mean, stationary, white noise.
3. The receiver is time synchronised with the transmitter i.e. the receiver knows the times that the modulation changes state.

Sometimes the receiver is also phase-locked with the transmitter. This is called coherent detection and the receiver is a coherent receiver. If the phase of the incoming signal is not known i.e. non-coherent detection, the receiver is called a non-coherent detector.

The transmitted signal $S_i(t)$ is of finite energy:

$$E = \int_0^\tau S_i^2(t)dt$$

One such signal e.g. $S_1(t)$ or $S_2(t)$ etc, is transmitted every T seconds. The signal transmitted depends upon the message.

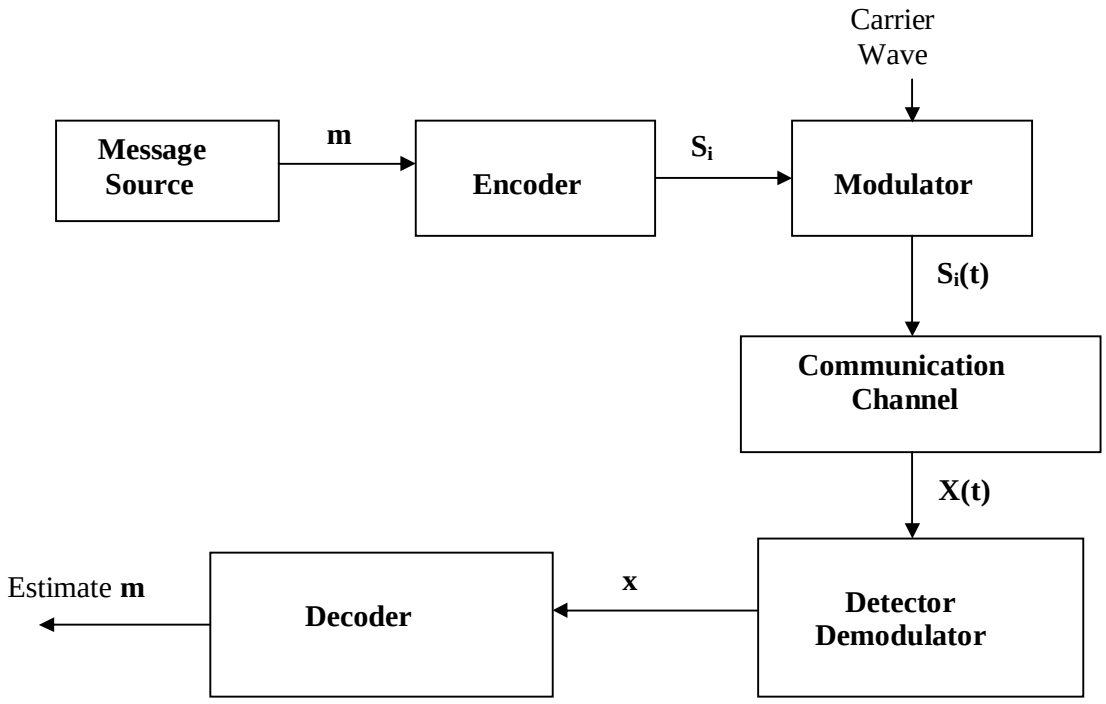


Figure5.1: A generalised digital pass band transmission system.

5.2 INTRODUCTION TO ASK, FSK and PSK:

Modulation involved switching (known as keying) between short bursts of different signals to transmit the encoded message. A general carrier wave may be written:

$$C(t) = A\sin(2\pi ft + \phi)$$

Three quantities may be varied to transmit the message: the amplitude *A*, the frequency *f* and the phase. Modulation methods based on varying these quantities to transmit digital data are known as Amplitude Shift Keying (ASK), Frequency Shift Keying (FSK) and Phase Shift Keying (PSK).

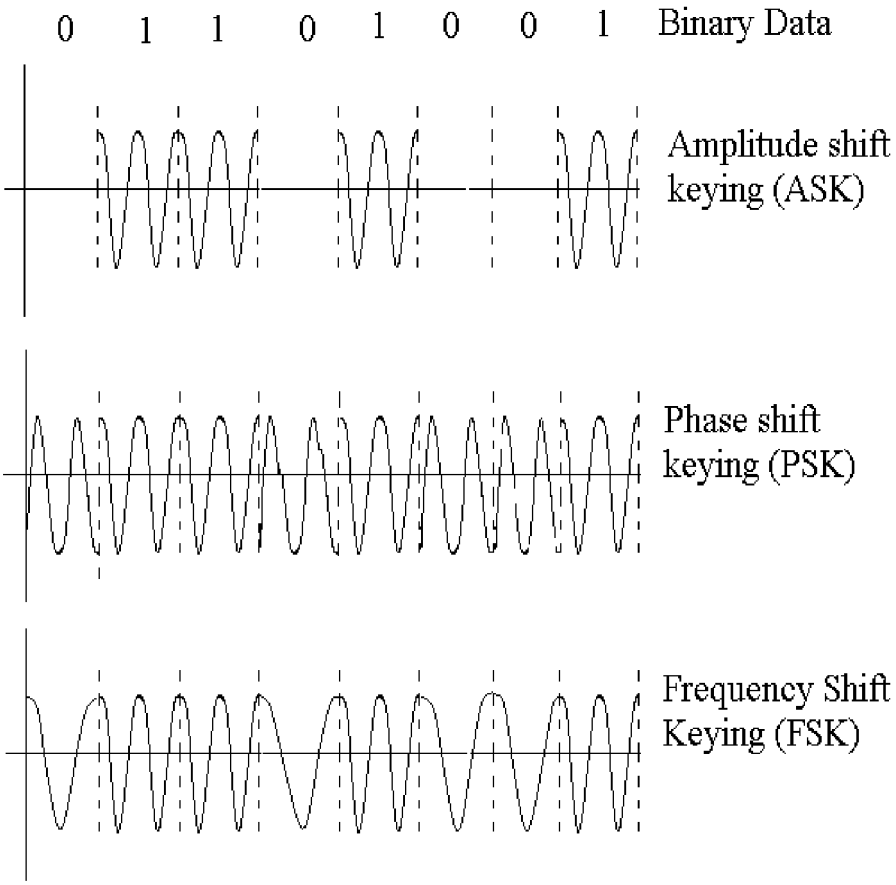


Figure 5.2: Basic digital modulation schemes.

5.2.2 Binary Frequency Shift Keying (FSK)

For frequency Shift Keying, the frequency of the signal is used to transmit the message i.e.

$$S_1(t) = A\cos(2\pi f_1 t) \qquad \text{or} \qquad S_1(t) = A\cos(2\pi(f_c - \Delta f)t)$$
$$S_2(t) = A\cos(2\pi f_2 t) \qquad\qquad S_2(t) = A\cos(2\pi(f_c + \Delta f)t)$$

for $-\frac{T}{2} \le t \le \frac{T}{2}$ and $f_1, f_2 << \frac{1}{T}$.

Δf is known as the frequency deviation and, for practical systems, varies in the range $\frac{r_b}{4} \le \Delta f \le \frac{r_b}{2}$ where r_b is the bit rate.

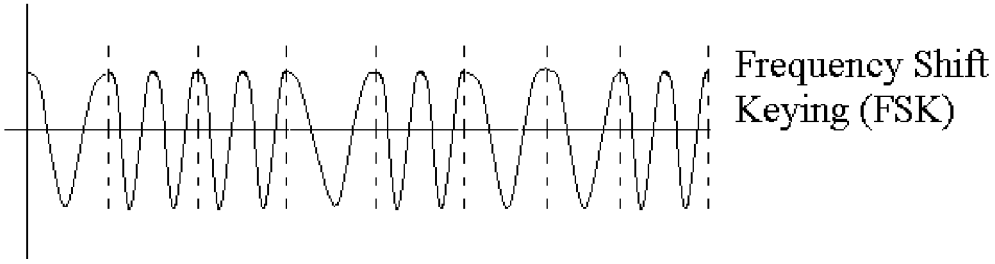


Figure 5.3: Example of continuous phase FSK.

For arbitrary Δf the FSK signal will have a step phase change at the transition between bits. However, Δf can be chosen so the FSK signal has continuous phase across bit boundaries. Continuous Phase Modulation (CPM) has the same bandwidth but the power falls of faster in adjacent channels so reducing inter-channel interference. The CPM FSK signal is easier to demodulate if the two coding signals S_1 and S_2 are orthogonal i.e. $\int_{-T/2}^{T/2} S_1(t)S_2(t)dt = 0$.

Continuous phase and orthogonality are imposed if $f_1 = m\frac{r_b}{2}$, $f_2 = n\frac{r_b}{2}$ and $\Delta f = (m-n)\frac{r_b}{2}$.

There are a number of ways of generating binary FSK signals:

1. Use two independent oscillators and switch between them
2. Use one oscillator and multiply up to the two frequencies switching between two modulating oscillators,
3. Use one voltage-controlled oscillator (VCO) in a phase-locked-loop (PLL) and switch between frequencies.

Methods 2 and 3 produce continuous phase signals while 1 has sudden phase shifts.

A large number of methods exist for the demodulation of FSK signals. Figures 5.4 and 5.5 illustrate coherent and non-coherent detectors. The coherent demodulator is essentially two parallel AM demodulators where the output of the upper or lower path are only non-zero when signal is present around the multiplier input signal frequency. The system relies upon the orthogonality of the trigonometric basis functions. The comparator produces a full-scale positive output when a signal is detected around f_1 and a full-scale negative output when a signal is detected around f_2.

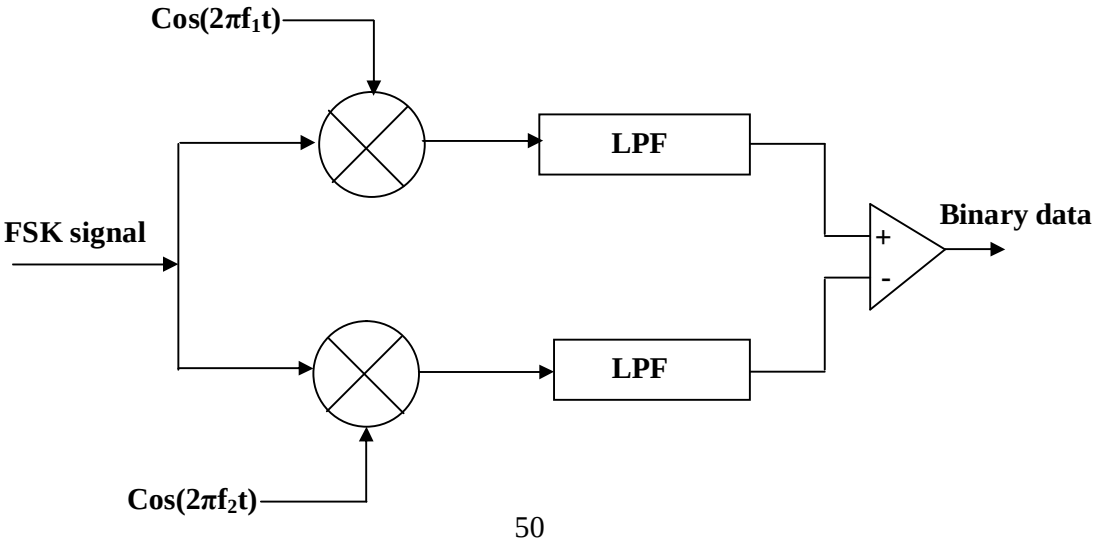


Figure5. 4: Coherent FSK detector and demodulator.

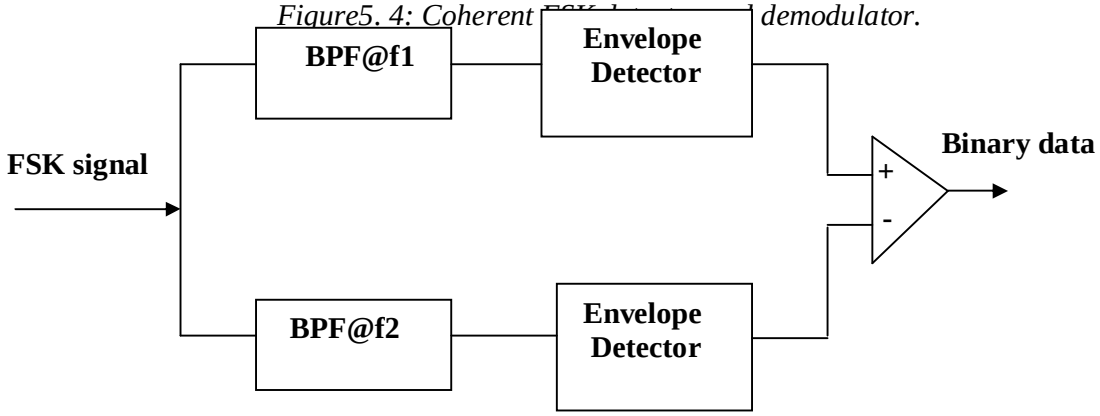


Figure 5.5: Non-coherent FSK detector and demodulator.

The non-coherent detector breaks the input FSK signal into two signals, each containing a frequency band corresponding to f_1 and f_2. This is achieved by two band-pass filters (BPF) with the pass bands centred on f_1 and f_2. The envelope detectors provide the non-coherent detection.

5.2.3 Frequency Spectra of FSK Signals

The FSK signal can be though of as the sum of two (orthogonal) ASK signals: the first ASK signal codes 1 as a burst of signal at f1 while the second signal codes a 0 as a burst of signal at f1. Each ASK signal can be thought of as a pulse convolved with a random train of delta functions and has an amplitude spectrum as in Figure 4. If the FSK signal was derived from the output of two independent (not phase locked) oscillators then the FSK power spectrum strongly resembles the sum of the two ASK spectra.

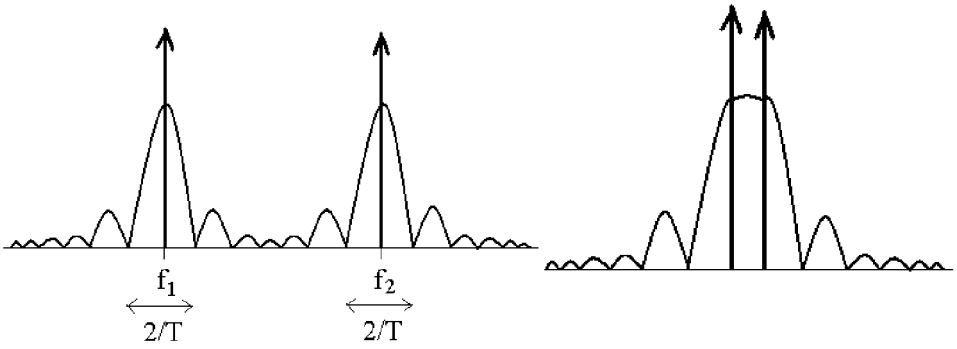


Fig. 11A Fig. 11B

Figure 5.6: Independent FSK amplitude spectrum for A) $\Delta f >> r_b$ and B) $\Delta f < r_b$

If the FSK signal is phase continuous then the two ASK signals are not independent. This leads to relations between the phases of the two spectra and so the power spectra cannot

be directly added. When the frequency deviation is small the FSK spectrum can be look quite unlike two ASK spectra.

For wide band modulation i.e. $\Delta f >> r_b$, the bandwidth is approximately $2\Delta f$. For narrow band modulation, i.e. $\Delta f < r_b$ the bandwidth is $2r_b$ (same as for OOK). For real systems the aim is to minimise the bandwidth used. The minimum possible frequency deviation for the two basis signals to be orthogonal and yield continuous phase is $\Delta f = r_b/4$. This is known as Orthogonal FSK or Minimum Shift Keying. Orthogonal FSK is commonly used in practice.

5.3 Binary Phase Shift Keying (PSK):

The general binary Phase Shift Keying (PSK) signal may be written:

$$S(t) = A_c \cos(2\pi f_c t + D_p m(t))$$
$$= A_c \cos(2\pi f_c t)\cos(D_p m(t)) - A_c \sin(2\pi f_c t)\sin(D_p m(t))$$

where $m(t) = \pm 1$ is a bi-polar, base-band, message signal made up of rectangular pulses. Since $m(t)$ is either +1 or -1 and sin and cos are odd and even respectively

$$S(t) = A_c \cos(2\pi f_c t)\cos(D_p) - A_c \sin(2\pi f_c t) m(t) \sin(D_p)$$

The first term is a scaled carrier wave where the scaling depends upon the phase deviation D_p. The second term contains the data (message) and is also scaled by D_p. In order to minimise the signalling efficiency i.e. minimise the probability of error, the data term needs to be maximised. The amplitude of the data term is maximised when $\sin(D_p) = 1$ i.e. $D_p = \frac{\pi}{2}$. In this case $\cos(D_p) = 0$ and the carrier term has zero amplitude. This is a special case of BPSK and is known as Phase Reversal Keying (PRK).

PRK generation is very similar to ASK (OOK) generation. The only difference is that the message signal is now bi-polar.

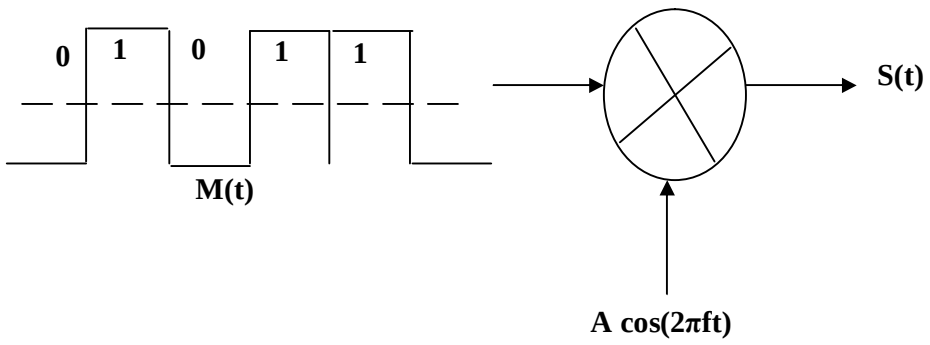


Figure 5.7: Illustrating of the modulation of a bi-polar message signal to yield a PRK

signal.

The spectrum of the PRK signal is also the same as for ASK (OOK) except that the discrete carrier delta function is no longer present (as the mean signal is now zero).

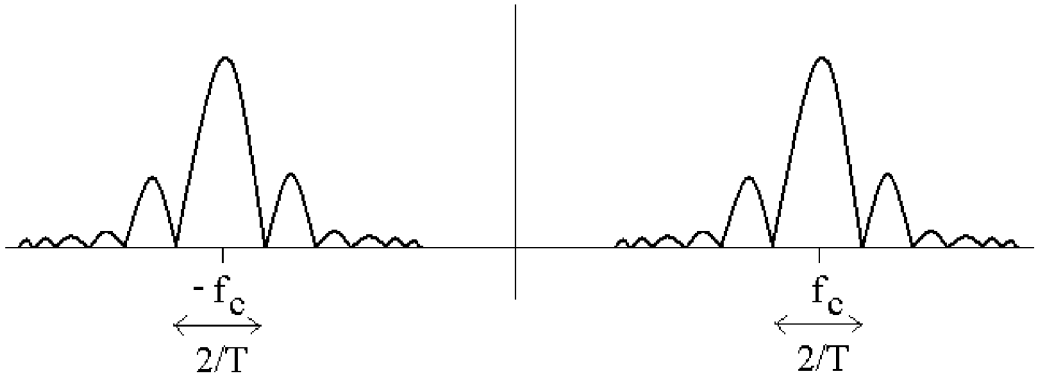


Figure 5.8: Amplitude spectrum of PSK.

Detection of PSK

PSK has a constant envelope so it cannot be detected by non-coherent methods e.g. envelope detectors. Coherent detection must be used.

Since there is no discrete carrier term in the PRK signal, a phase-locked loop (PLL) may not be used to extract the carrier reference. However, for general BPSK i.e.

$$|D_p| < \frac{\pi}{2}$$

a small amount of carrier is also broadcast and this can be used to aid demodulation.

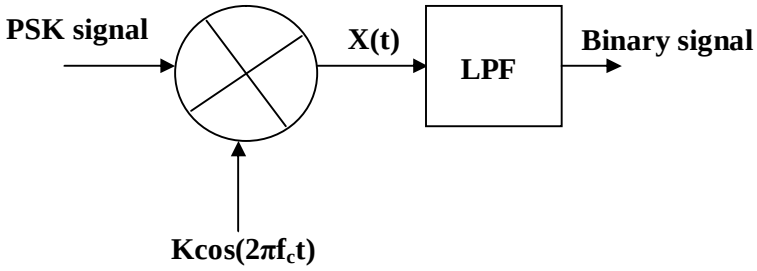


Figure 5.9: Coherent demodulation of PRK signal.

Several methods have been proposed for generating a reference carrier signal from a received PRK waveform. The extra effort in detecting PRK is worthwhile since, of all the binary modulation schemes, it offers the best noise performance.

5.4 DIFFERENTIAL PHASE SHIFT KEYING (DPSK):

This is a partially coherent technique to detect PRK. In DPSK the information is encoded in the difference between consecutive bits e.g.

PCM code input		1	0	1	1	0	1	0	0	1
Differential code	1	1	0	0	0	1	1	0	1	1
Phase	0	0	π	π	π	0	0	π	0	0

Table 5.1: example of Differential Phase Shift Keying (DPSK)

The DPSK sequence has one extra starting bit which is arbitrary. Succeeding bits in the differential encoding are determined by the rule that "there is no change in the output state if a "1" is present.

6. VITERBI ALGORITHM IN CP-FSK

6.1 INTRODUCTION:

The HART signal path from the processor in a sending device to the processor in a receiving device is shown in figure 1.5. Amplifiers, filters, etc. have been omitted for simplicity. At this level the diagram is the same, regardless of whether a Master or Slave is transmitting. Notice that, if the signal starts out as a current, the "Network" converts it to a voltage. But if it starts out a voltage it stays a voltage.

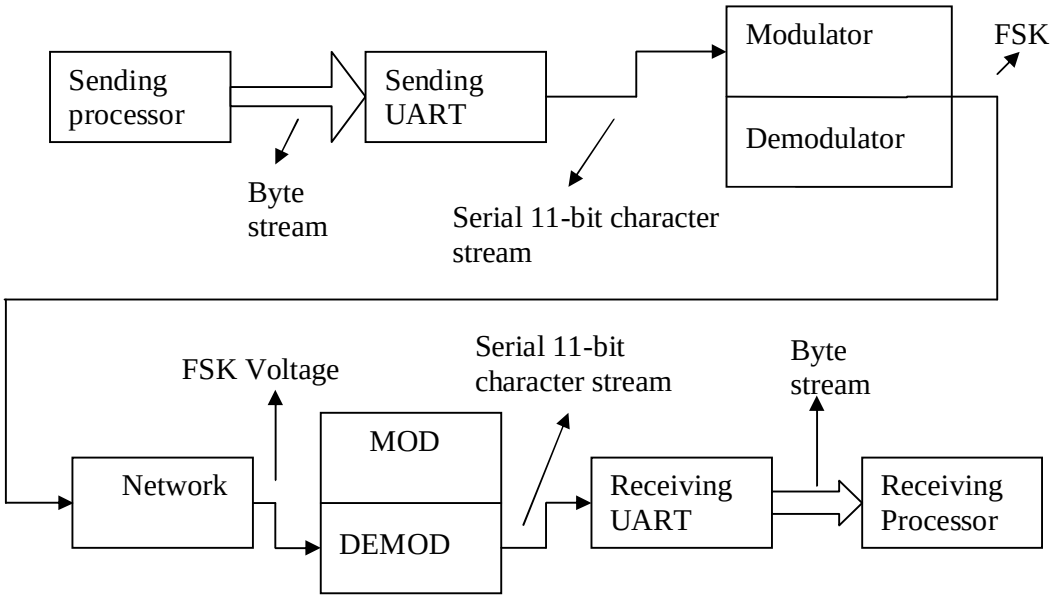


Figure 6.1 HART Signal Path

The transmitting device begins by turning ON its carrier and loading the first byte to be transmitted into its UART. It waits for the byte to be transmitted and then loads the next one. This is repeated until all the bytes of the message are exhausted. The transmitter then waits for the last byte to be serialized and finally turns off its carrier. With minor exceptions, the transmitting device does not allow a gap to occur in the serial stream.

The UART converts each transmitted byte into an 11 bit serial character, as in figure 1.6. The original byte becomes the part labeled "Data Byte (8 bits)". The start and stop bits are used for synchronization. The parity bit is part of the HART error detection. These 3 added bits contribute to "overhead" in HART communication.

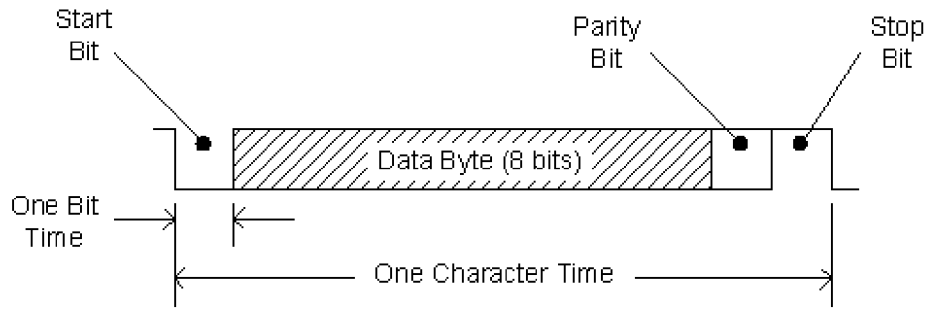


Figure 6.2 HART Character Structure

The serial character stream is applied to the Modulator of the sending modem. The Modulator operates such that a logic 1 applied to the input produces a 1200 Hz periodic signal at the Modulator output. Logic 0 produces 2200 Hz. The type of modulation used is called Continuous Phase Frequency Shift Keying (CPFSK). "Continuous Phase" means that there is no discontinuity in the Modulator output when the frequency changes. A magnified view of what happens is illustrated in figure 1.7 for the stop bit to start bit transition. When the UART output (modulator input) switches from logic 1 to logic 0, the frequency changes from 1200 Hz to 2200 Hz with just a change in slope of the transmitted waveform. A moment's thought reveals that the phase doesn't change through this transition. Given the chosen shift frequencies and the bit rate, a transition can occur at any phase.

At the receiving end, the demodulator section of a modem converts FSK back into a serial bit stream at 1200 bps. Each 11-bit character is converted back into an 8-bit byte and parity is checked. The receiving processor reads the incoming UART bytes and checks parity for each one until there are no more or until parsing of the data stream indicates that this is the last byte of the message. The receiving processor accepts the incoming message only if it's amplitude is high enough to cause carrier detect to be asserted. In some cases the receiving processor will have to test an I/O line to make this determination. In others the carrier detect signal gates the receive data so that nothing (no transitions) reaches the receiving UART unless carrier detect is asserted.

6.2 CONTINUOUS PHASE FSK:

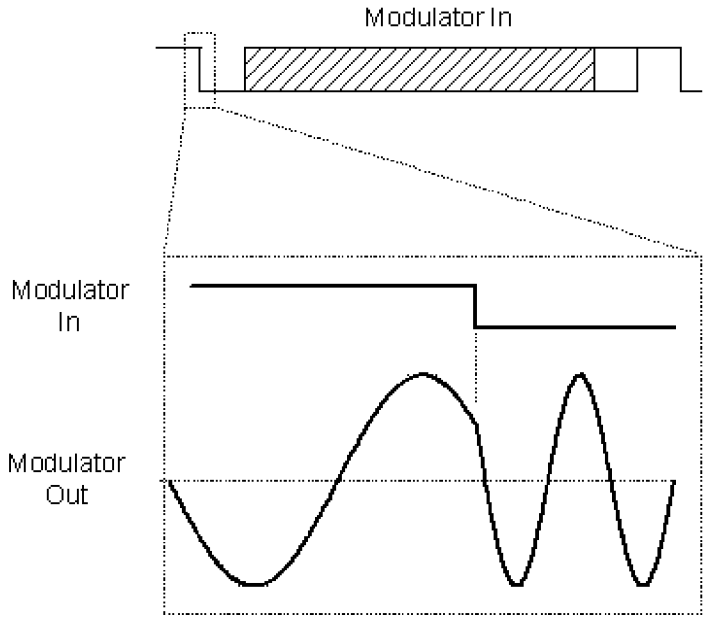


Figure 6.3 -- Illustration of Continuous Phase FSK

A mathematical description of continuous phase FSK is given below

Equation Describes CPFSK

The HART signal is described mathematically as

$$V = V_0 \sin[\theta_0 + \theta(t)]$$

Where

V = signal voltage,

t = time,

V_0=Amplitude

θ_0 =s an arbitrary starting phase,

$\theta(t)$ is given by

$$\theta(t) = \int_0^t \left[2\pi(1700Hz) + 2\pi(500Hz) \sum_n B_n(t - nT) \right] d\tau$$

Where $B_n(t)$ is a pulse that exists from $0 < t < T$ and has a value of 1 or -1, according to whether the nth bit is a 0 or 1. T is one bit time. If phase is plotted versus time it is a steadily increasing value that increases with two possible slopes. Based on the setting in above section, VB can be implemented in Binary CPFSK(BCPFSK). MAP will be used to select the shortest path in the trellis diagram.

It is known that the probability of bit errors (Pe) in this model is estimated

As $Pe = Q(\frac{\sqrt{2}}{2\sigma})$. Assume that the priori probability of the binary signal is the same (i.e.: P (1) = P (0) =0.5), it is easy to show that the probability to shift from one state to another is the same.

Two survival sequences in the k^{th} step are denoted as Γ_{k1} and Γ_{k2} . The Viterbi recursive equation can be written as the following three stages:

1. Initialize: k=0, $\Gamma_{k1} = \Gamma_{k2} = 0$, and assign the memory space for state sequence X (k) where X (k) is a 2 X K_e array. K_e is the length of the sequence.
2. For step k:

$if\ (y_{ok}, y_{1k}) = (1,0)$

$\Gamma_{k1} = \min\{[\Gamma_{k-1,1} - \ln(1-P) - \ln(1-P_e)], [\Gamma_{k-1,2} - \ln(Q) - \ln(Pe)]\}$

$\Gamma_{k2} = \min\{[\Gamma_{k-1,1} - \ln(P) - \ln(P_e)], [\Gamma_{k-1,2} - \ln(1-Q) - \ln(Pe)]\}$

$f\ (y_{ok}, y_{1k}) = (0,1)$

$\Gamma_{k1} = \min\{[\Gamma_{k-1,1} - \ln(1-P) - \ln(P_e)], [\Gamma_{k-1,2} - \ln(Q) - \ln(Pe)]\}$

$\Gamma_{k2} = \min\{[\Gamma_{k-1,1} - \ln(P) - \ln(1-P_e)], [\Gamma_{k-1,2} - \ln(1-Q) - \ln(Pe)]\}$

$f\ (y_{ok}, y_{1k}) = (0,-1)$

$\Gamma_{k1} = \min\{[\Gamma_{k-1,1} - \ln(1-P) - \ln(P_e)], [\Gamma_{k-1,2} - \ln(Q) - \ln(1-Pe)]\}$

$\Gamma_{k2} = \min\{[\Gamma_{k-1,1} - \ln(P) - \ln(P_e)], [\Gamma_{k-1,2} - \ln(1-Q) - \ln(Pe)]\}$

$f\ (y_{ok}, y_{1k}) = (-1,0)$

$\Gamma_{k1} = \min\{[\Gamma_{k-1,1} - \ln(1-P) - \ln(P_e)], [\Gamma_{k-1,2} - \ln(Q) - \ln(Pe)]\}$

$\Gamma_{k2} = \min\{[\Gamma_{k-1,1} - \ln(P) - \ln(P_e)], [\Gamma_{k-1,2} - \ln(1-Q) - \ln(1-Pe)]\}$

2. Sequence Γ_{k1} , Γ_{k2} and state sequence X (k) are stored, and stage 2 is executed for step k+1.With the finite state sequences X (k), the shortest complete path will be got and the corresponding signal sequence can be estimated.

6.3 PERFORMANCE ANALYSIS:

It is known that the probability of any error event starting at time k may be upper-bounded or lower bounded. In the model of CPFSK, the bound can be calculated as follows: two modulated signal are assumed to be:

$$s_0(t) = \sqrt{2E_b}\cos(w(0)t + \theta_1)$$
$$s_1(t) = \sqrt{2E_b}\cos(w(1)t + \theta_2)$$

From step k, the error events can happen at step k, k+1, k+2, k+3… These can be seen in Figure 7. (e.g.: the correct path should be 0-0-0-0, but due to the error events in the middle, the path could be 0-π- π-0)

The probability can be obtained from the Euclidean Distance between the incorrect sequence and the correct sequence. For the sequence of step k+1, the Euclidean Distance is $2\sqrt{E_b}$, the probability for this particular error event is $Q(\frac{\sqrt{E_b}}{\sigma})$. It is the lower bound of Pe2. The probability is accumulated for all the error events in subsequent steps. The range of Pe can be obtained as

$$Q(\frac{\sqrt{E_b}}{\sigma}) < Pe < Q(\frac{\sqrt{E_b}}{\sigma}) + Q(\frac{\sqrt{3E_b}}{\sigma}) + ...Q(\frac{\sqrt{(2k+1)E_b}}{\sigma})$$

The right part of this formula is bounded because $Q(x)$ decreases rapidly with x. So, the Pe can be estimated as $Q(\frac{\sqrt{E_b}}{\sigma})$. Since the one-side white noise power spectral density (denoted as n_0) $n_0 = \frac{1}{2}\sigma^2$, so the Pe can be simplified as: $Pe = Q(\frac{\sqrt{2E_b}}{2\sqrt{n_0}})$

However, the Coherent Detection of Binary FSK inis

$$Pe = Q(\frac{\sqrt{E_b}}{\sqrt{n_0}})$$

So Pe (probability of bit error) is reduced from $Q(\frac{\sqrt{E_b}}{\sqrt{n_0}})$ to $Q(\frac{\sqrt{E_b}}{\sqrt{2n_0}})$ when the VA is used in BCPFSK. In order to verify my deduction, the relevant simulation of BCFSK is processed in Matlab. Figure 8 depicts that when the scale of Bit Error Rate (BER) is

Under 0.01dB, the detection based on Viterbi Algorithm improves SNR by 3dB, which matches the mathematical analysis above. Actually, the VA algorithm gives an ideal performance, which gives a benchmark for the sub-optimum decoding methods.

6.4 MEMORY COST

It can be seen that due to the recursive feature, VA does not need much storage space in estimation process of MAP or MLSE. Most of the memory will be allocated for the storage of the state sequence X (k). The storage size of the sequence is $m \times L$ where m is the number of states and L is the truncated length of the sequence.

6.4.1 Computational Cost

In BCFSK, the decoder will do the job of plus-compare-select twice for each additional step. Similarly, the job of plus-compare-select will be done m times in m states trellis decoding process for each input sequence. Hence, the total cost for the sequence with length (L) and M states will be $O(ML)$. In some high-speed Digital Signal Processor such as TMS320C series and ADSP21000 series, with processing speed of thousands of MIPS (million instructions per second), it is not difficult to implement VA in perspective of hardware.

6.5 IMPLEMENTATION ISSUES

There are several issues that need to address in the future so that VA can be applied into more Digital Signal Processing (DSP) applications.

6.5.1 The sequence length

From this model, the decision of the estimation won't be made until the entire signal is received. Hence, it leads to the problem that VA will not work if the state sequence is infinite. It is necessary to truncate the survivors to some manageable length (L) under such circumstance. L should be chosen large enough, so that at time K = n*L, (n is positive integer). All the survivors are in the same state sequence. Hence, the length L is related to the power of the noise in the channel obviously. On the other hand, it should not be too long otherwise it will introduce the unnecessary delay and lower the efficiency of decoding. The trade-off has to be made between accuracy and efficiency.

6.5.2 Last step in BCPFSK

The state sequence X (k) is estimated only when the survivor length Γ_{k+1} is calculated because the longer the sequence is, the more memory will be hogged. On the other hand, the probability of decision error decreases accordingly. At the last step (denoted as K_{last}), there

are still m states left for the next step. So there are$\Gamma_{Klast+1}$, $\Gamma_{Klast+2}$ to make these sequences to converge at the same node at step K_{last}. If the decision is made only according to Γ_{klast} , the probability of the error in the last state will be much greater than that in previous states. Some dummy sequence will be used to drive the state sequence to go on at the end of the truncated sequences. If the length of the dummy sequence is Le, then Le should be chosen large enough so that at step K_{last} + Le, all the m sequences will converge at the same node at step K_{last} + Le. Under such circumstance, the decision can be made with state sequence stored in the X (k) because at step K_{last}+Le, all the state sequences from step 1 to step K_{last} + Le are the same. The length Le depends on the number of the states m and the power of the noise

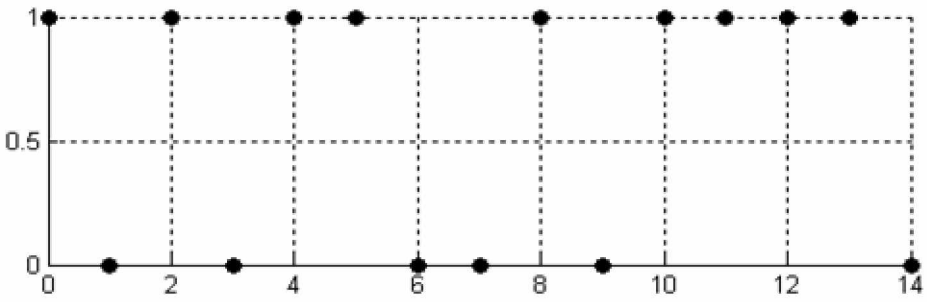
6.5.3 Initial state

The algorithm is required to start with knowing the initial state X (0), but it might not be satisfied in practice. If K_e is the state that all the possible sequence in CFSK converges into, then VA can take K_e as the initial state, which is proved to be finite in. Figure 9 depicts the problems in 5.2 and 5.3. It shows the output state sequence of the Viterbi decoding in convolutional coding. All the sequences converge together for the first 12 steps but diverge after step 12. So, the initial state can be decided as state 1. Actually, the states in first 12 steps can be precisely decided in this case. So to the problem in 5.3, the initial state can be the state in any step between step 0 and step 12. To the problem in section5.2, the sequences diverge from step 12 to step 16. The extra more steps are required after step 16 to make these sequences converge again so that all the decision on path selection can be made precisely.

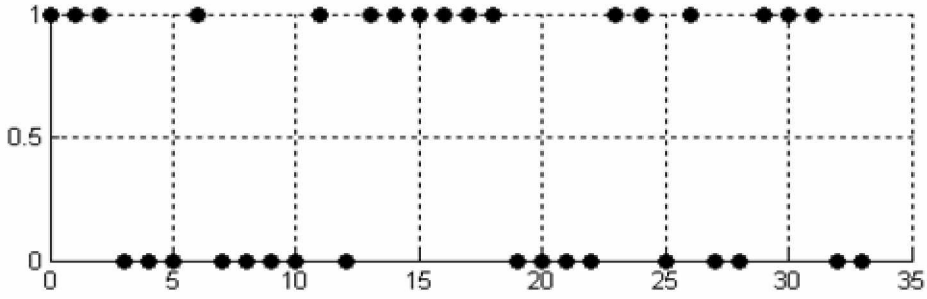
7.1 SIMULATION RESULTS

1. CONVOLUTIONAL ENCODING AND VITERBI DECODER

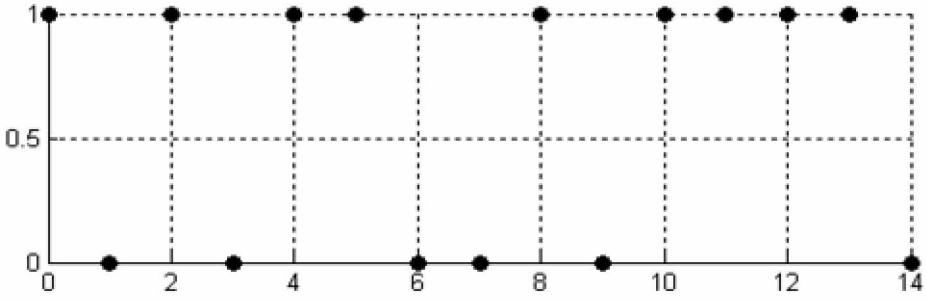
1.1 INPUT SEQUENCE



1.2 ENCODED SEQUENCE

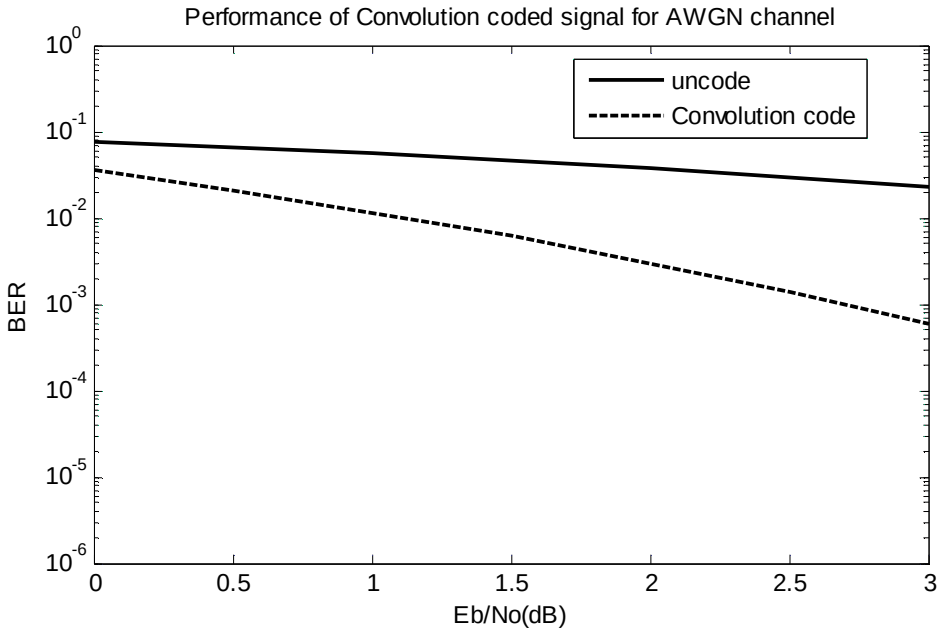


1.3 VITERBI DECODER SEQUENCE

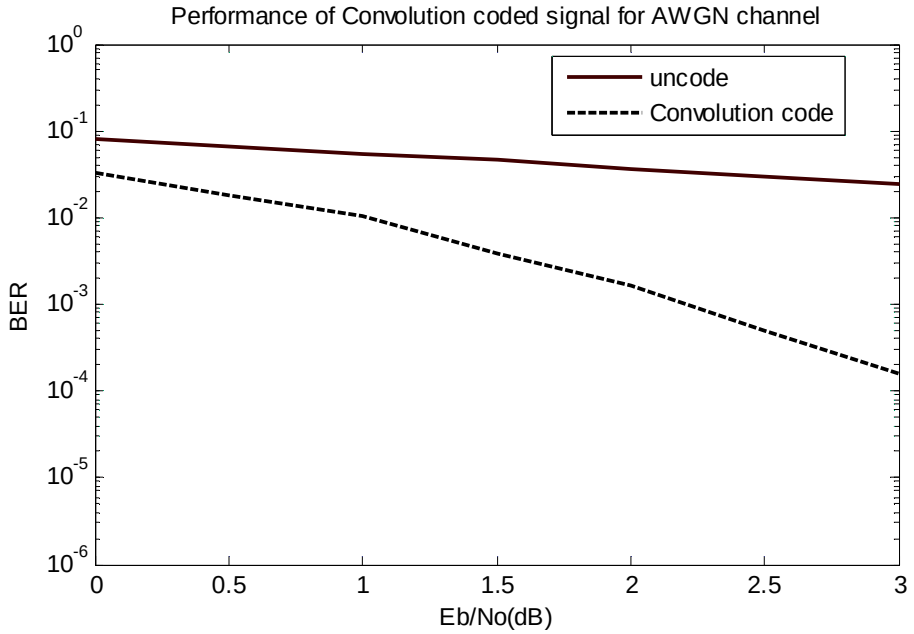


7.2 PERFORMANCE OF VITERBI DECODER FOR CONVOLUTIONAL CODES

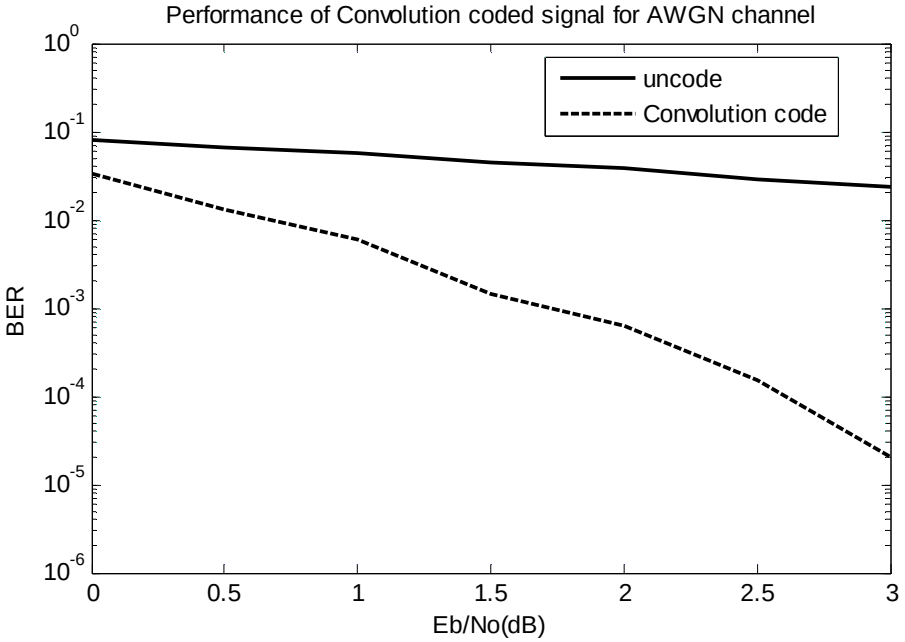
2.1 Convolution code Rate 1/2 K=3 generator polynomials G0= 7, G1= 5



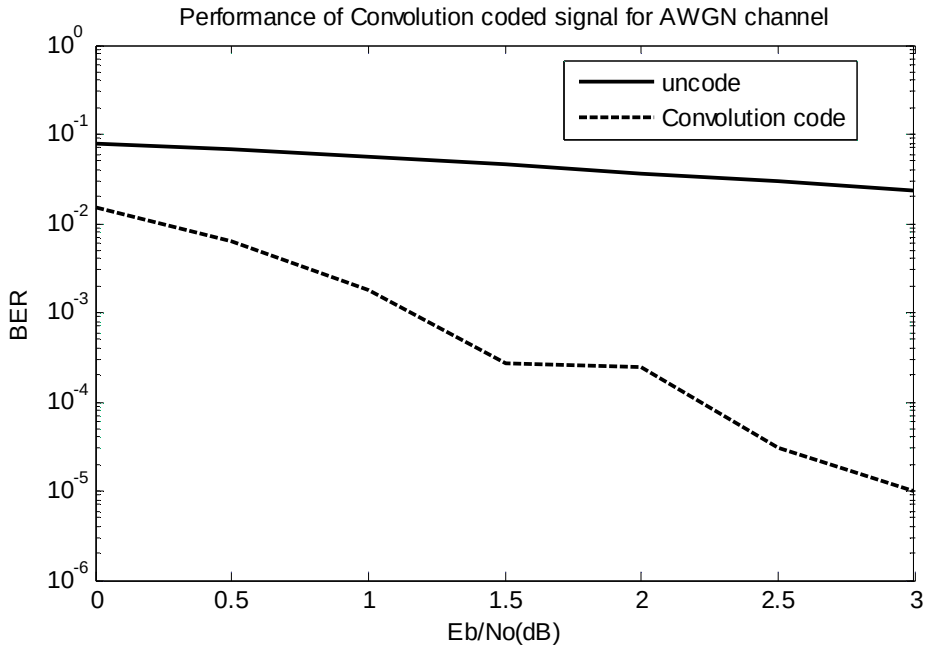
2.2 Convolution code Rate 1/2 K=5 generator polynomials G0= 35, G1=23



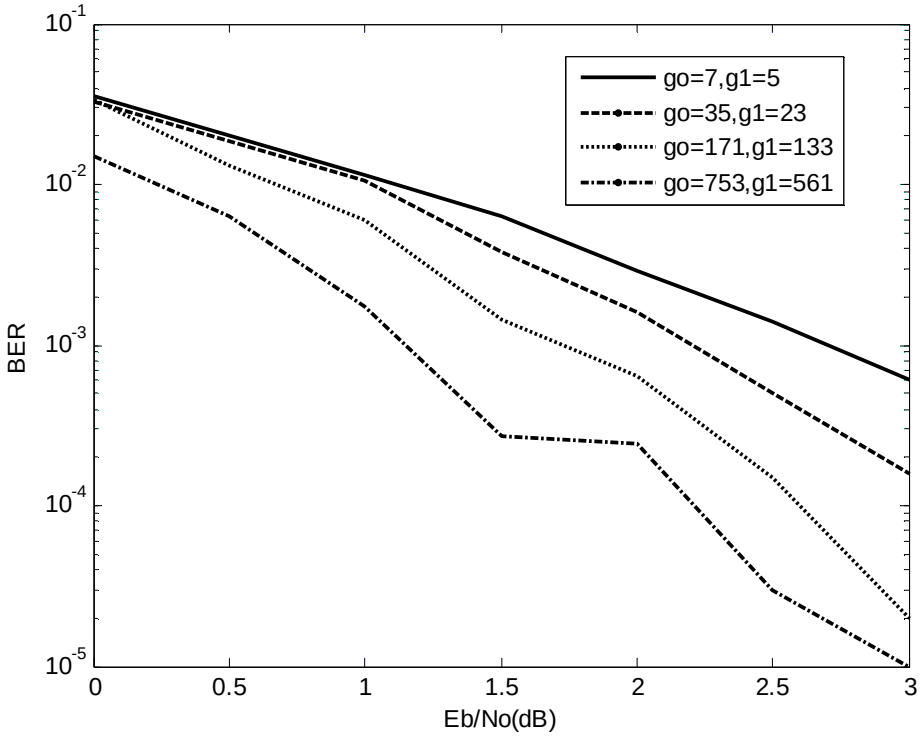
2.3 Convolution code Rate 1/2 K=7 generator polynomials G0=171, G1= 133



2.4 Convolution code Rate 1/2 K=9 generator polynomials G0= 753, G1= 561

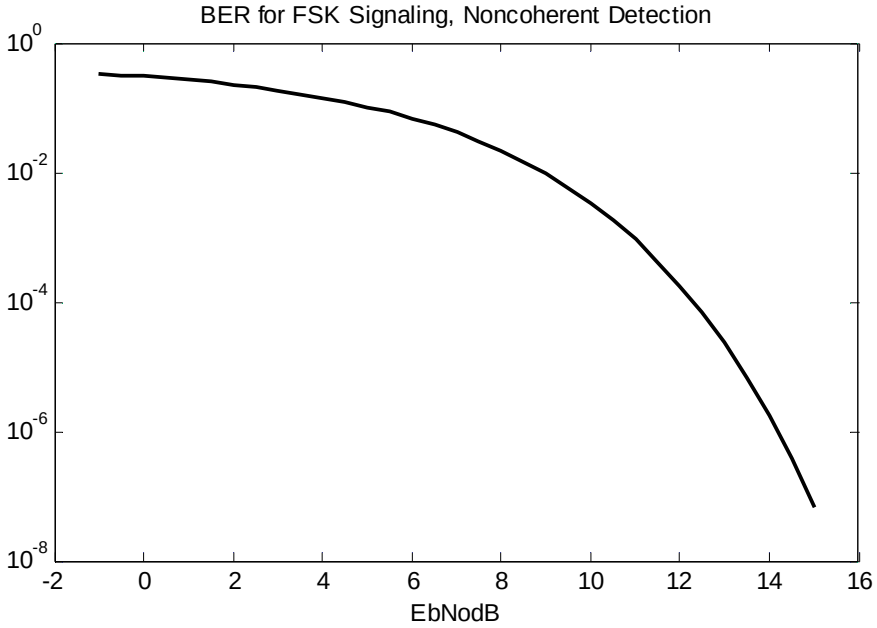


2.5 Comparison of convolutionally coded signal for different generator polynomials

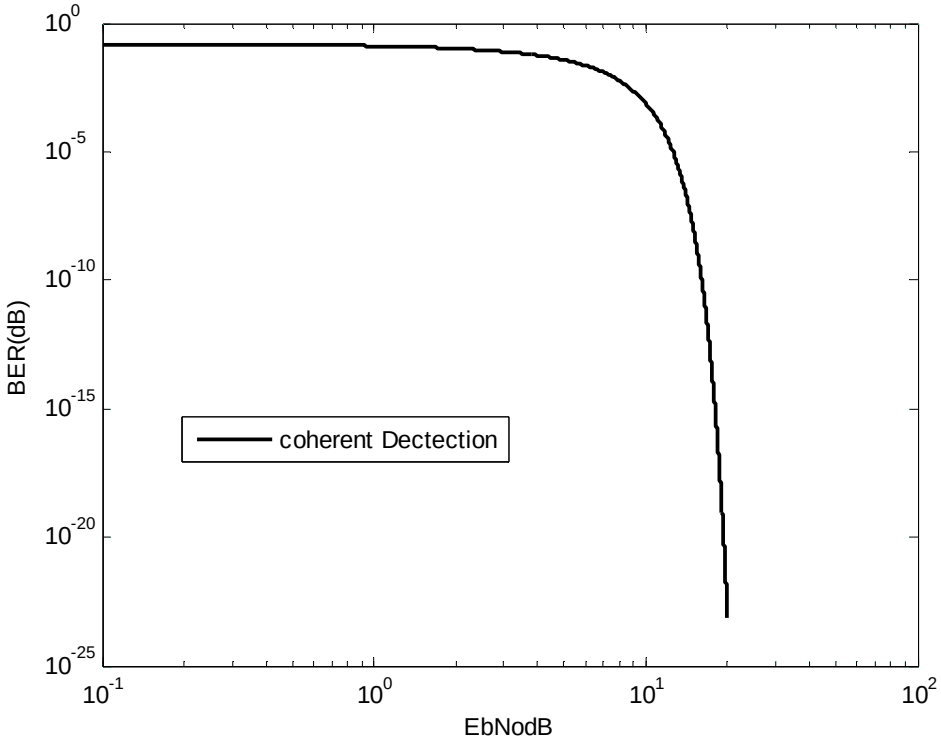


7.3 PERFORMANCE OF FREQUENCY SHIFT KEYING

3.1 Performance of Non-Coherent detected FSK Signal

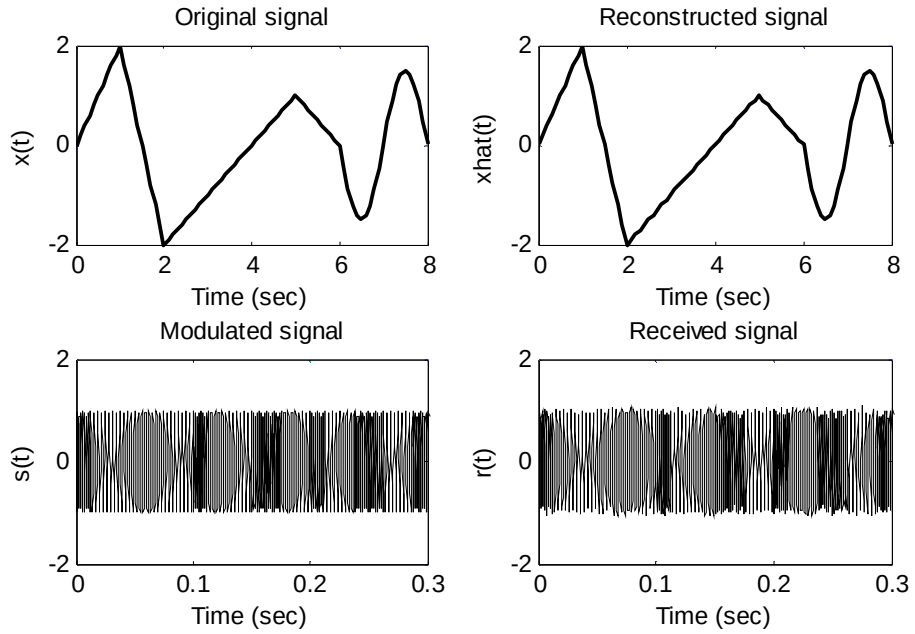


3.2 Performance of Coherent detected FSK Signal

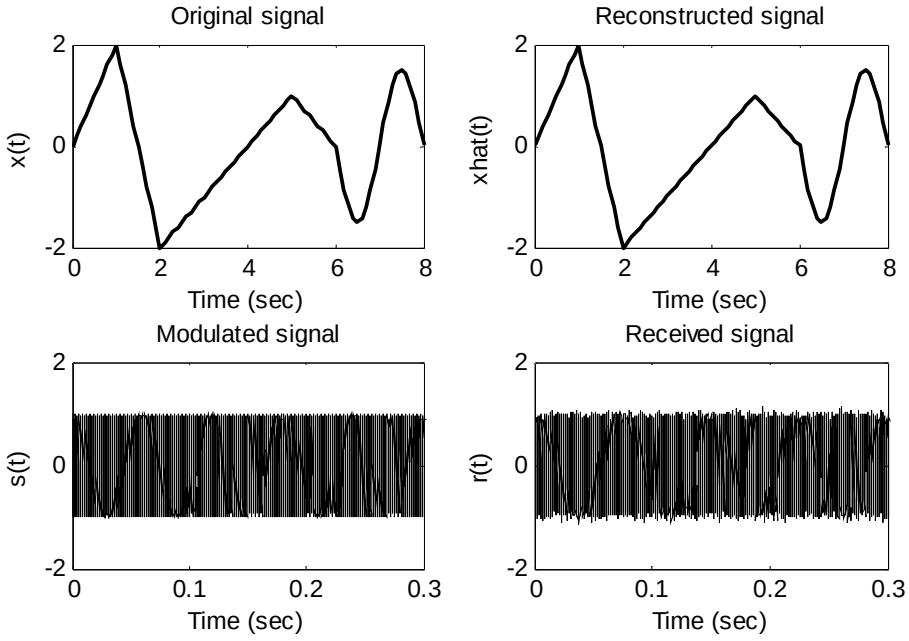


7.4 CONVOLUTIONALLY MODULATED AND DEMODULATED FSK SIGNAL

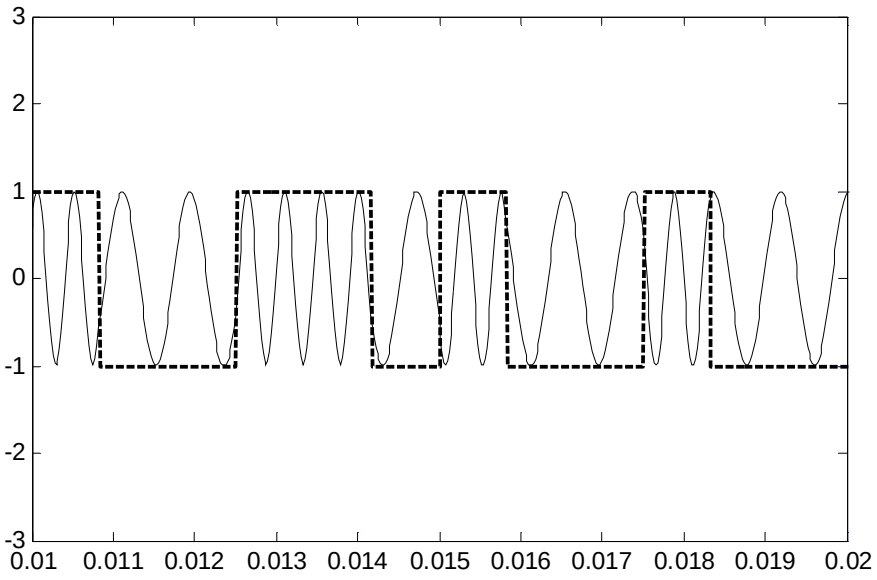
4.1 Bits for Sample = 8



4.2 Bits per Sample=16

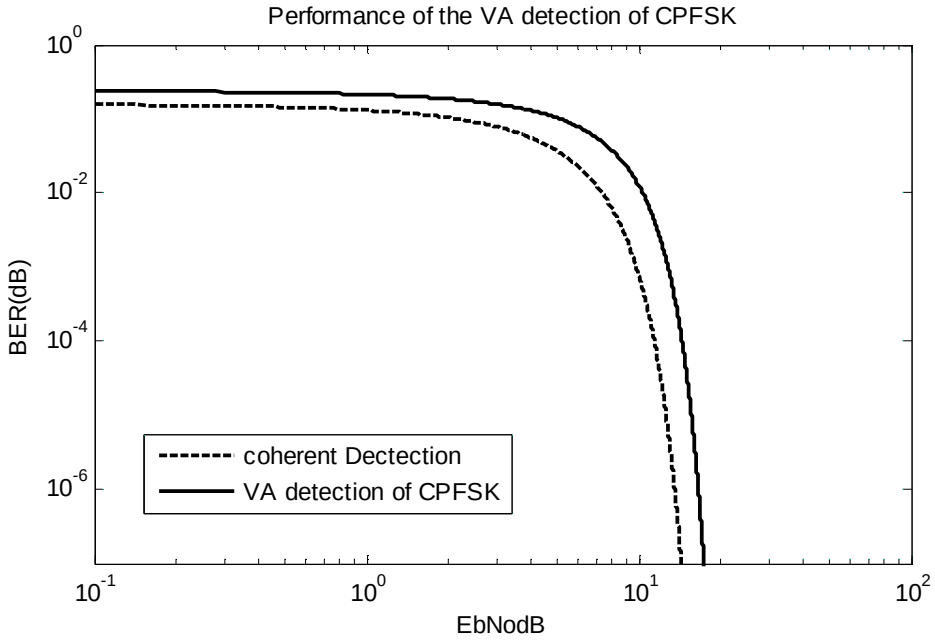


7.5 MODULATED SIGNAL OF CONTINUOUS-PHASE FREQUENCY SHIFT KEYING



7.6 PERFORMANCE OF VITERBI ALGORITHM IN CONTINUOUS-PHASE FREQUENCY SHIFT KEYING

6.1PERFORMANCE OF VA FOR COHERENT AND NON COHERENT DETECTION



8. CONCLUSION AND FUTURE WORK

Convolutional coding is a coding scheme often employed in deep space communications and recently in digital wireless communications. Viterbi decoders are used to decode convolutional codes. Viterbi decoders employed in digital wireless communications are complex in its implementation and dissipate large power. We investigated a Viterbi decoder design.

By building the convolutional encoder and Viterbi decoder in the behavior model, the MATLAB simulation results give us a light on its performance and how to implement it in a RTL system. From the results, we find the Viterbi decoding algorithm is mature error correct system, which will give us a BER at 8.6E-007 at 5db on an AWGN channel with BPSK modulation. By puncturing, for rate 2/3, we will pay around a 2db cost. For rate 3/4, we will pay for a 3 db cost during the transmission.

VA is also analyzed in the perspective of memory requirement and computational cost. The recursive characteristic of the algorithm makes it relatively easy to implement in some high-speed programmable hardware devices. The concept of VA is to adopt extra memory to enlarge the distance of the signal sequence or the states. For MLSE, convolutional coding encodes the signal sequence with the encoding ratio of ½ and the constraint length of 3.Hence, the maximum distance of the binary signal is enlarged from 1 to 2.For MAP, the application of phase and frequency of FSK modulation can be utilized as a ½ ratio encoding method for the binary sequence to enlarge the signal distance. VA fully depends on the constraint characteristics of the signal. The statistical methods such as MLSE and MAP are used to extract the original signal with the presence of noise. So, the memory feature and the statistical methods make the signal more robust to the noise.

With these characteristics, VA has a bright future in wireless communication. For example, Direct Sequence Code Division Multiple Access (DS-CDMA) is an active research area with the objective of efficient channel sharing and utilization strategies. A fundamental problem in cellular DS-CDMA systems is interference. While some research has already shown that the channel coding/decoding algorithms provide a significant improvement in the performance of Multi-user detectors. VA will be widely used to improve the performance of the detectors.

For the time issue, we do not implement a higher performance Viterbi decoder with such as pipelining or interleaving. So in the future, with Pipeline or interleave the ACS and the trace-back and output decode block, we can make it better. Another trend is to implement a Viterbi decoder in the Turbo code algorithm. With different require specification, there must be a modify in the original design.

REFERENCES

[1] Viterbi Algorithm in Continuous-Phase Frequency Shift Keying Hailun Tan; Digital Telecommunications, 2006. ICDT '06. International Conference on 2006 Page(s):5 - 5

[2] G. David Forney, J., Maximum-Likelihood Sequence Estimation of Digital Sequences in the Presence of Intersymbol Interference. IEEE, Trans, Inf. Theory, 1972.

[3] Viterbi, A.J., Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. IEEE Trans. Information Theory, 1967. 13(2): p. 260-269.

[4] G. David Forney, J., The Viterbi Algorithm. Proceedings of IEEE, 1973. 61(3).

[5] Proakis, J.G., Digital Communications: McGraw Hill.

[6] Zhigang Cao, Y.Q., Communication Theory: Tsinghua University Press.

[7] Piret, P., Convolutional Codes, an algebraic approach: MIT Press.

[8] Chari, M.J.R.a.S., Demodulation of Cochannel FSK Signals Using Joint Maximum likelihood Sequence Estimation. IEEE, Trans., 1993. 2: p. 1412 - 1416

[9] Narkat, M., Signal Detection and Estimation: Artech House.

[10] Dai, M., Information Theory: Tongji University Press.

[11] Jun Horikoshi, K.K.a.T.O. Error Performance Improvement of Bandlimited QTFSK Assisted by an Efficient Coded Modulation and Viterbi Sequence Estimation. in IEEE, Trans, Infomation. Theory. 1994.

[12] Rappaport, T.S., Wireless Communications, Principles and Practice: Prentice Hall.

[13] M. G. Pelchat, R. C. Davis, and M. B. Luntz, “Coherent demodulation of continuous-phase binary FSK signals,” iPnr oc. Inl. Telemetry Conf. (Washington, D. C., 1971).

[14] R. de Buda, “Coherent demodulation of frequency-shift keying with low deviation ratio,” IEEE Trans. Commun. Technol., vol. COM-20, pp. 429-435, June 1972.

[l5] 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.

[16] G. D. Forney, Jr., “Convolutional codes I: Algebraic structure,” IEEE Trans. Inform. Theory, vol. IT-16, pp. 720-738, Nov. 1970.

[17] G. C. Clark, R. C. Davis, J. C. Herndon, and D. D. McRae, “Interim report on convolution coding research,” Advanced System Operation, Radiation Inc., Melbourne, Fla., Memo Rep. 38, Sept. 1969.

[18] A. J. Viterbi and J. P. Odenwalder, “Further results on optimal decoding of convolutional codes,” IEEE Trans. Inform. Theory (Corresp.), vol. IT-15, pp. 732-734, Nov. 1969.

[19] A. J. Viterbi, “Convolutional codes and their performance in communication systems,” IEEE Trans. Commun. Technol.vol. COM-19, pp. 751-772, Oct. 1971.

[20] G. D. Forney, Jr., “Convolutional codes 11: Maximum likelihood decoding,’! Stanford Electronics Labs., Stanford, Calif., Tech. Rep. 7004-1, June 1972.

[21] J. A. Heller, “Short constraint length convolutional codes,” in Space Program Summary 37-54**,** vol. 111. Jet Propulsion Lab., Calif. Inst. Technol., pp. 171-177, 0ct.-Nov. 1968.

[22] “Improved performance of short constraint length convolutional codes,” in S pace Program Summary37-56, vol. 111. Jet Propulsion Lab., Calif. Inst. Technol., pp. 83-84, Feb.-Mar. 1969.

[23] J. P. Odenwalder, “Optimal decoding of convolutional codes,” Ph.D. dissertation, Dep. Syst. Sci., Sch. Eng. Appl. Sci., Univ. of California, Los Angeles, 1970.

[24] J. L. Ramsey, “Cascaded tree codes,” MIT Res. Lab. Electron., Univ. of California, Los Angeles, 1970. Cambridge, Mass., Tech. Rep. 478, Sept. 1970.

[25] G. W. Zeoli, “Coupled decoding of block-convolutional concatenated Los Angeles, 1971. codes,” Ph.D. dissertation, Dep. Elec. Eng., Univ. of California,

[26] J. M. Wozencraft, “Sequential decoding for reliable communication,” in 1957 IRE Nat. Conv. Rec., vol. 5, pt. 2, pp. 11-25.

[27] R. M. Fano, “A heuristic discussion of probabilistic decoding,” IEEE Trans. Inform. Theory, vol. IT-9, pp. 64-74, Apr. 1963.

[28] K. S. Zigangirov, “Some sequential decoding procedures,” Probl Pered. Inform., vol. 2, pp. 13-25, 1966.

[29] F. Jelinek, “Fast sequential decoding algorithm using a stack,” IBM J. Res. Develop**.,** vol. 13, pp. 675-685, Nov. 1969.

[30] L. R. Bahl, C. D. Cullum, W. D. Frazer, and F. Jelinek, “An efficient algorithm for computing free distance,” IEEE Trans. Inform. Theory (Corresp.), vol. IT-18, pp. 437-439, May 1972.

[31] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, “Optimal decoding of linear codes for minimizing symbol error rate” (Abstract), in 1972 In#. Symp. Information Theory (Pacific Grove, Calif., Jan. 1972),

[32] P. L. McAdam, L. R. Welch, and C. L. Weber, “M.A.P. bit decoding of convolutional codes,” in 1972 Inf. Symp. Informafion Theory(Pacific Grove, Calif., Jan. 1972), p. 91.