ON DECODING OF MULTI-LEVEL MPSK MODULATION CODES

Technical Report

to

NASA

Goddard Space Flight Center

Greenbelt , Maryland 20771

Grant Number NAG 5-931

Report Number NASA 90-003

Shu Lin

Principal Investigator

Department of Electrical Engineering

University of Hawaii at Manoa

Honululu, Hawaii 96822

May 20 ,1990

ON DECODING OF MULTI-LEVEL MPSK MODULATION CODES

Alok Kumar Gupta
Shu Lin
Principal Investigator
Department of Electrical Engineering
University of Hawaii at Manoa
Honolulu, Hawaii 96822

ABSTRACT

In this report, the decoding problem of multi-level block modulation codes is investigated. It consists of two parts. In the first part, the hardware design of soft-decision Viterbi decoder for some short length 8-PSK block modulation codes is presented. An effective way to reduce the hardware complexity of the decoder by reducing the branch metric and path metric, using a non-uniform floating-point to integer mapping scheme, is proposed and discussed. The simulation results of the design are presented. In the second part, the multi-stage decoding (MSD) of multi-level modulation codes is investigated. The cases of soft-decision and hard-decision MSD are considered and their performance are evaluated for several codes of different lengths and different minimum squared Euclidean distances. It is shown that the soft-decision MSD reduces the decoding complexity drastically and it is suboptimum. The hard-decision MSD further simplifies the decoding while still mantains a reasonable coding gain over the uncoded system, if the component codes are chosen properly. Finally, some basic 3-level 8-PSK modulation codes using BCH codes as component codes are constructed and their coding gains are found for hard-decision multi-stage decoding.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Conventionally in digital communication, channel coding is designed and performed separately from modulation. In the cases of both block codes and convolutional codes, error control is achieved by replacing k-tuple message with a well-structured n-tuple codeword, where n > k. Transmission of these (n-k) redundant symbols requires either a bandwidth expansion or a reduction of data rate. Either case results in lowering the information rate per channel bandwidth, known as the bandwidth efficiency. This type of channel coding is suitable for power limited channels without bandwidth constraints, where bandwidth efficiency is traded for increased power efficiency and coding gain or reliability is achieved at the expense of bandwidth expansion or reduction of data rate. However, for channels with bandwidth constraint such as voiceband telephone terrestrial microwave channels and mobile and high speed satellite channels, bandwidth is a very precious resource and bandwidth expansion is either not desirable or possible. In the past, therefore, coding has never been popular for bandlimited channel.

Recently, however, there has been increased interest in some type of combined modulation and coding scheme known as coded-modulation or bandwidth efficient coding, that achieves coding gain

with little or no bandwidth expansion. At first, it may seem that this statement violates some basic power-bandwidth-error probability trade-off principle. However, there is still a trade-off at work; coded modulation achieves coding gain at the expense of increased decoder complexity. In 1982, Ungerbock showed that by combining coding and modulation properly, significant coding gain over uncoded modulation systems can be achieved without compromising bandwidth efficiency. Ever since a great deal of research effort has been expended in bandwidth efficient coded modulation for achieving reliable communication on bandlimited channels.

The basic idea behind the coded-modulation is as follows. The error performance of an uncoded non-orthogonal M-ary modulation (such as PSK or QAM) depends largely on the distance between the closest pair of signal points and is upper-bounded by

$$0.5\ Ne\ erfc(d_{min}/2\sqrt{No})$$

for a AWGN channel, where Ne is the average number of nearest neighbours per signal point, d_{min} is minimum distance between the signal points in the two dimensional Euclidean plane and No/2 is the power spectral density of noise. The minimum distance is determined by the average transmitter power and the number and position of the signal points. For a constant average power, the minimum distance between points decreases as the number of points increases. Therefore, assuming a constant channel symbol

rate and constant average power, the error performance degrades for systems that attempt to increase the transmission bit rate or bandwidth-efficiency by increasing the size of the symbol set.

The basic concept of coded modulation is to encode the information bits onto an expanded signal alphabet (relative to that needed for uncoded modulation). This signal set expansion provides the needed redundancy for error control without sacrificing bandwidth efficiency. The expanded signal set does result in a reduced distance between adjacent symbol points for a given average power. However, coding is used to introduce a certain dependency between successive signal points, such that only certain pattern (or sequences) of signal points are permitted. Thus the reduced distance between adjacent symbol points no longer determines the error performance. Instead, the minimum Euclidean distance between the members of the set of allowable symbol sequences (codeword in signal space) determines the error performance.

1.1 TYPES OF MODULATION CODES, PERFORMANCE MEASURE AND OPTIMUM DECODING STRATEGY

In a coded modulation error control system, information sequences are coded into signal sequences over a certain modulation signal set (e. g. an 8-PSK signal set). These signal

sequences form a modulation code. Based on the code structure, coded modulations are classified into two categories: trellis coded-modulations (TCM) and block coded-modulations (BCM). In a trellis coded modulation, an information sequence is encoded by a convolutional code and mapped onto a modulation signal set by a bit-to-symbol mapper. The resultant modulation code has a trellis structure and hence can be decoded by the Viterbi decoding algorithm. For this reason, we call this modulation code a trellis modulation code. In block modulation code, a message of k bits is encoded by block component codes of length n and mapped onto a modulation signal set by a bit-to-signal mapper. The resultant code is called a block modulation code. If the block component codes are chosen properly, the resultant modulation code has a trellis structure and hence can be decoded by the Viterbi decoding algorithm.

In order to achieve good error performance, modulation codes are generally decoded using soft-decision decoding based on an Euclidean distance measure. The coding gain of a modulation code is largely determined by its minimum squared Euclidean distance.

Let s be a point (X(s), Y(s)) in the two dimensional Euclidean space R. Let s and s' be two points in R. The squared Euclidean distance between s and s', denoted d(s,s'), is defined as follows:

$$d(s,s') = (X(s)-X(s'))^2 + (Y(s)-Y(s'))^2$$

Let $v = (s_1, s_2, \ldots\ldots, s_n)$ and $v' = (s_1', s_2', \ldots\ldots, s_n')$ be two n-tuples over R. The squared Euclidean distance between v and v', denoted $|v-v'|^2$, is defined as follows:

$$|v-v'|^2 = \sum_{i=1}^{n} (X(s_i) - X(s_i'))^2 + (Y(s_i) - Y(s_i'))^2$$

Let C be a modulation code of length n with signals from a two-dimensional modulation signal set S. The minimum squared Euclidean (MSE) distance of C, denoted D[C], is defined as follows:

$$D[C] = \min\{ | v-v' |^2 : v, v' \text{ in } C \text{ and } v \neq v'\}$$

The effective rate of C, denoted R[C], is given by

$$R[C] = 1/2n \log_2 |C|$$

which is the average number of information bits transmitted by C per dimension (of the signal set). The total number of information bits in one codeword of C is called the dimension of C.

Assume that the channel is an additive white Gaussian noise (AWGN) channel and all the code sequences of a modulation code are equally likely to be transmitted. Let $r = ((x_1,y_1), (x_2,y_2,),\ldots\ldots (x_n,y_n,))$ be the output sequences of the demodulator. Then the squared Euclidean distance between r and a code sequence $v = (s_1,s_2,\ldots\ldots s_n)$ in C is

$$|r-v|^2 = \sum_{i=1}^{n} (x_i - X(s_i))^2 + (y_i - Y(s_i))^2$$

For maximum likelihood decoding, the received sequence $\mathbf{r}$ is decoded into a code sequence $\mathbf{v}_i$ if

$$|\mathbf{r}-\mathbf{v}_i|^2 < |\mathbf{r}-\mathbf{v}_l|^2$$

for all $l \neq i$. To perform soft-decision maximum likelihood decoding, it is desirable for a modulation code to have a trellis structure so that the Viterbi decoding algorithm can be applied to reduce the number of computations and the decoder complexity.

Consider a coded modulation system using a modulation code C with minimum squared Euclidean distance D[C] and effective rate R[C]. For the purpose of performance comparison, we choose a proper reference system (coded or uncoded) using a modulation code C_S with MSE distance $D[C_S]$ and effective rate $R[C_S]$. Then, the asymptotic coding gain (ACG) of the coded system C over the reference system C_S (assuming same average power for both the system) is given by

$$10 \log_{10} \frac{R[C].D[C]}{R[C_s].D[C_s]}$$

The asymptotic coding gain is used as a simple measure of the performance of a coded modulation system.

This research project is set up to investigate the decoding problem of multi-level block modulation codes. The design of optimum decoder for some short length modulation codes with

simple trellis structure is presented and alternate and practical decoding schemes for more complex codes are investigated.

In Chapter 2, multi-level construction method for multi-level code is reviewed. In chapter 3, hardware design of Viterbi decoders for two short length 8-PSK block modulation codes is described. The multi-stage decoding of multi-level codes is investigated in chapter 4. Finally, chapter 5 is devoted to concluding remarks.

CHAPTER 2

MULTI-LEVEL CONSTRUCTION OF MPSK BLOCK MODULATION CODE

In this chapter, we describe a powerful technique for constructing M-ary PSK block modulation code with arbitrary large MSE distance, known as the multi-level construction.

2.1 CODE CONSTRUCTION

The construction steps are as follows:

1. Selection of a signal set: A set of 2^l signal points is chosen.

2. Signal labeling by set partitioning: Each signal point is labeled by a string of l bits. Such labeling is said to have l levels. These label strings must be designed to provide the resultant modulation code with the largest possible minimum squared Euclidean distance when bits-to-signal mapping is performed. Labeling is generally done by a set partitioning process.

3. Selection of component codes: The component codes may be binary or nonbinary, block or trellis.

4. Code construction and bits-to-signal mapping: Component codes are combined to form a label. Then, the label is mapped into a signal point. This mapping results in a multi-level modulation code.

If the number of component codes is equal to the number of labeling levels (l), the resultant modulation code is called a **basic** multi-level modulation code.

We use an example to demonstrate the construction.

1 Selection of a signal set: We consider 8-PSK signal set (or constellation) as shown in figure 2.1, where each signal point is labeled with an integer s from the set S = {0,1,2,3,4,5,6,7}. The squared Euclidean distance between two signal points s and s' in the 8-PSK signal set S is given by

$$d(s,s') = 4 \sin^2(\frac{(s-s')\pi}{8})$$

2. Signal labeling by set partitioning: Binary digits are assigned to each point in the 8-PSK signal space according to Ungerbock's set partitioning scheme.

Let X be the subset of S. The intra-set distance of X, denoted d[X], is given by

$$d[X] = \min \{d(x, x') : x, x' \text{ in } X\}$$

Let X and Y be two subsets of S. The set separation between X and Y, denoted d[X, Y], is defined as follows:

$$d[X, Y] = \min \{d(x, y) : x \text{ in } X \text{ and } y \text{ in } Y\}$$

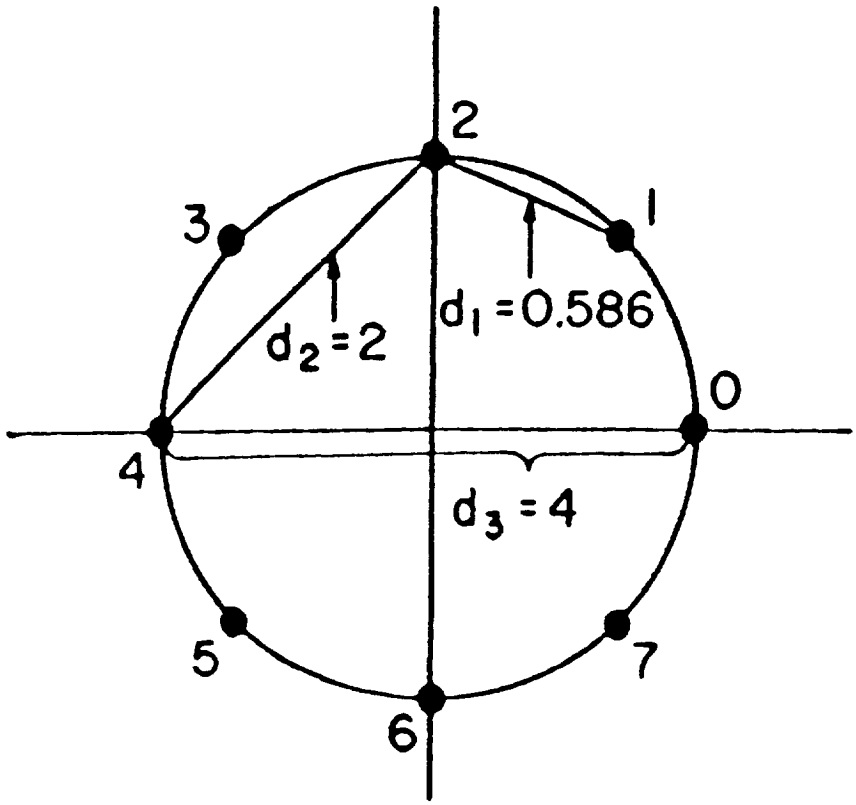


Fig. 2.1 8-PSK signal set

we form a chain of partitions of the signal set S with increasing number of subsets, increasing intra-set distances and set separations as shown in figure 2.2. The partition process results in 3 partitions of S with with increasing intra-set distances. At each partitioning stage, a subset of S is labeled by a binary sequence, and at the end of partitioning process, each signal point is labeled with a binary string of 3 digits.

The first partition consists of two disjoint subsets which are labeled by 0 and 1. The second partition consists of four disjoint subsets which are labeled by 00, 01, 10 and 11 respectively. The third partition consists of 8 disjoint subsets, each consisting of only one signal point, which are labeled by 8 unique 3-tuples. The partition is carried out in such a way, as the partition level increasing, the intra-set distance (the minimum squared Euclidean distance among signal points) of a set in a partition increases. For the 8-PSK case, the intra-set distances at 3 partition levels are:

$$0.586 \quad , \quad 2 \quad , \quad 4$$

The label strings formed from the above partitioning process have the following important properties:

(1) Two signal points with labels different at the first position are at a squared Euclidean distance at least $d_1 = 0.586$ apart.

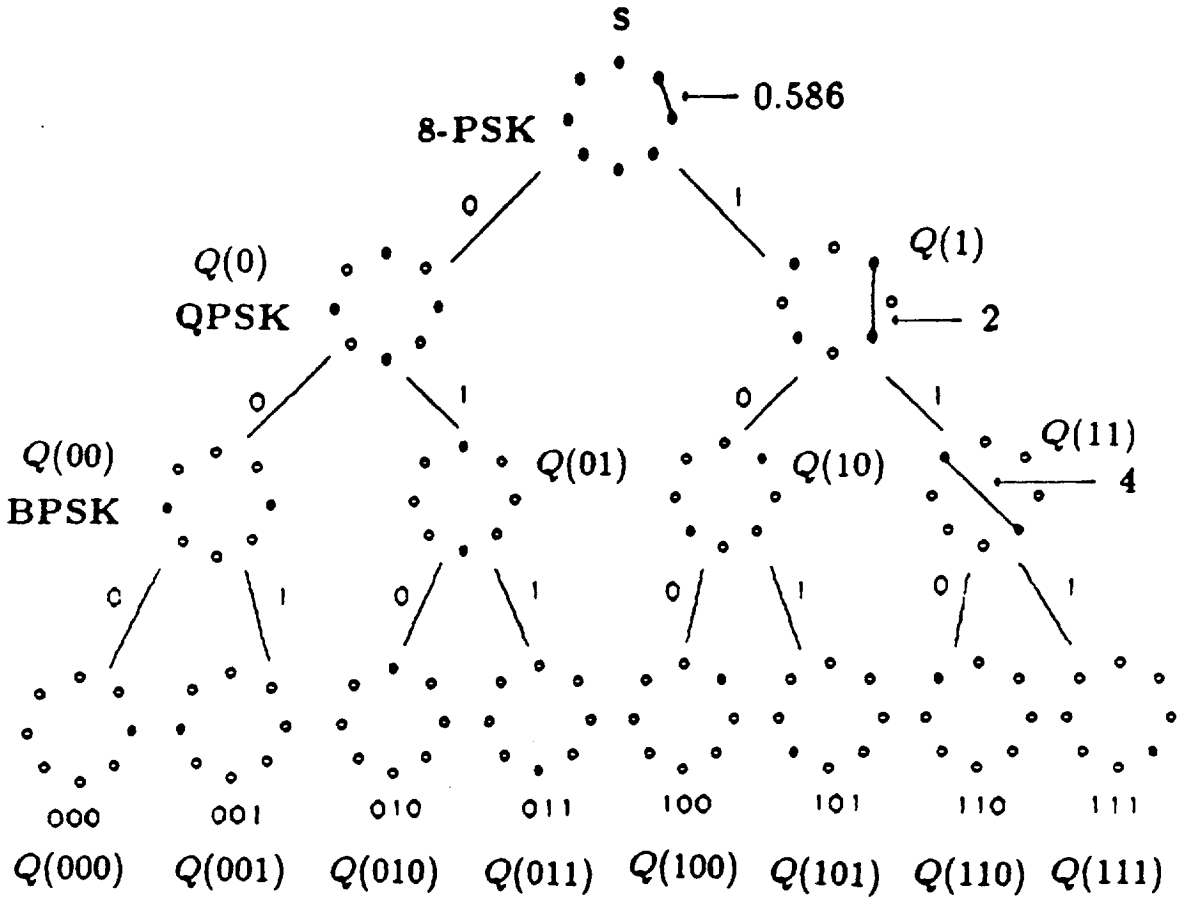


Fig. 2.2 Signal labeling and 8-PSK/QPSK/BPSK partitioning chain

(2) Two signal points with labels identical at the first position but different at the second position are separated by a squared Euclidean distance at least $d_2 = 2$ apart.

(3) Two signal points with labels identical at the first two positions but different at the last position are at a squared Euclidean distance at least $d_3 = 4$ apart.

The monotonically increasing property of the distances d_1, d_2 and d_3 is one of the keys to the construction of bandwidth efficient modulation codes. These distances are called the **distance parameters** of the signal label strings.

From figure 2.2, we see that each subset in the first partition is a QPSK signal set and each subset in the second partition is a BPSK signal set.

Each signal point s in S is labeled by a string of 3-bits. Let abc be the label for signal point s in S. Then a, b and c are 1st,2nd and 3rd levels respectively. Let μ be the mapping from binary labeling to a point in signal space, then

$$\mu(abc)=s$$

The 8-signal points and their corresponding labels are shown in figure 2.3. The label abc for s in S turns out to be the binary representation of integer s. Each integer s in S can be expressed in the following polynomial form

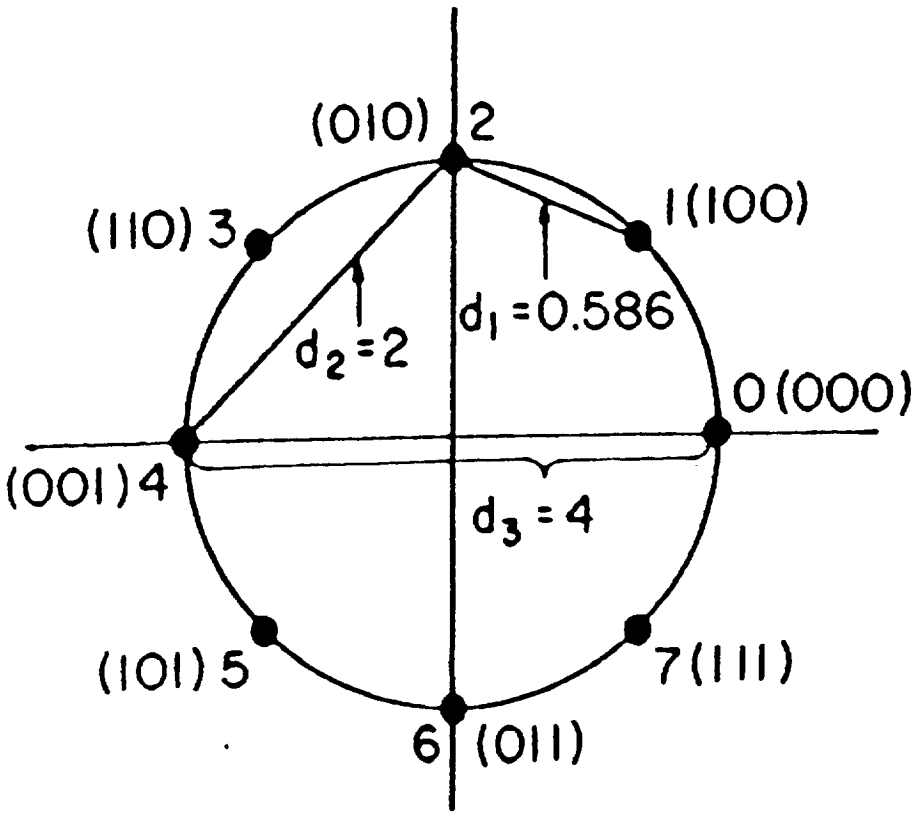


Fig. 2.3 8-PSK signal points and their labels

$$s = b_0 + b_1 \cdot 2 + b_2 \cdot 2^2$$

where $b_j = 0$ or 1 for $0 \leq i \leq 2$. The 3-tuples (b_0, b_1, b_2) is the binary representation of integer s.

Each prefix of a label represents a subset of signal points in S. For example, prefix a represents 4 signal points in a QPSK signal constellation, prefix ab represents 2 signal points in a BPSK signal constellation and prefix abc represents a single signal point in signal set S. Let Q(a) denote the set of signal points whose labels have a as the prefix. Then,

Q(a)={ s: s is a signal point in QPSK signal constellation}

Let Q(ab) denote the set of signal points whose labels have ab as the prefix. Then,

Q(ab)={s: s is a signal point in BPSK signal constellation}

Let d_1,d_2 and d_3 be the intra-set squared Euclidean distances of S, Q(a) and Q(ab) respectively. For the 8-PSK signal constellation d_1=0.586, d_2=2, d_3=4.

3. Selection of component codes: For the construction of a basic multi-level code, the number of component codes required is equal to number of levels in the labeling of signal points. For example, in 8-PSK signal set each signal point is labeled with 3 bits. Hence, 3 binary component codes are required to construct an 8-PSK block modulation code. All the component codes are chosen

to have equal length and proper minimum hamming distances. For $1 \le i \le 3$, let C_i be a binary (n , k_i) code with minimum hamming distances ∂_i.

4. Code construction and bits-to-signal mapping: Let

$$\mathbf{a} = (a_1 \; a_2 \; a_3 \; \ldots\ldots\ldots\ldots\ldots a_n)$$

$$\mathbf{b} = (b_1 \; b_2 \; b_3 \; \ldots\ldots\ldots\ldots\ldots b_n)$$

$$\mathbf{c} = (c_1 \; c_2 \; c_3 \; \ldots\ldots\ldots\ldots\ldots c_n)$$

be three codewords in C_1, C_2 and C_3 respectively. We form the following sequences,

$$\mathbf{a} * \mathbf{b} * \mathbf{c} = (a_1 \, b_1 \, c_1, \; a_2 \, b_2 \, c_2, \; \ldots\ldots\ldots\ldots a_n \, b_n \, c_n)$$

For $1 \le i \le n$, we take $a_i \, b_i \, c_i$ as the label of a signal point in the 8-PSK signal set. Then,

$$\mu(\mathbf{a} * \mathbf{b} * \mathbf{c}) = (\mu(a_1 \, b_1 \, c_1), \ldots\ldots\ldots\ldots, \mu(a_n \, b_n \, c_n))$$

is a sequences of 8-PSK signals. The set

$$C = \{ \mu(\mathbf{a} * \mathbf{b} * \mathbf{c}) : \mathbf{a} \text{ is in } C_1, \mathbf{b} \text{ is in } C_2 \text{ and } \mathbf{c} \text{ is in } C_3 \}$$

$$= C_1 * C_2 * C_3$$

is a 3-level block 8-PSK modulation code. The minimum squared Euclidean distance of a basic 3-level modulation code is [7]

$$D[C] = \min\{d_1 \partial_1, d_2 \partial_2, d_3 \partial_3\}$$

For a 3-level 8-PSK block modulation code,

$$D[C] = \min\{0.586\partial_1, 2\partial_2, 4\partial_3\}$$

2.2 ENCODING AND DECODING

Encoding of a 3-level basic 8-PSK modulation code C of length n constructed based on the above method can be done as follows. A message u of

$$k = \sum_{i=1}^{3} k_{bi}$$

bits (called a segment) is divided into 3 sub-blocks, the i-th sub-block consists of k_{bi} bits. For $1 \le i \le 3$, the i-th sub-block is encoded into a code vector v_i in the binary component code C_i of C. Then

$$\mu(v) = \mu(v_1 * v_2 * v_3)$$

is a codeword in C for the message segment u. The components of $\mu(v)$ are then mapped into points in a 2-dimensional 8-PSK signal set and transmitted. Hence, each message segment of k bits is encoded into a sequence of n 8-PSK signals.

A soft-decision decoding algorithm for the above 8-PSK codes can be devised as follows. For any element s in the group $S = \{0,1,\ldots 7\}$, let X(s) and Y(s) be defined as

$$X(s) = \cos(\pi s/4), \qquad Y(s) = \sin(\pi s/4)$$

For $1 \le j \le n$, let (x_j, y_j) be the normalized output of a coherent demodulator for the j-th symbol of a received frame. The received frame is then represented by the following 2n-tuple: $\mathbf{z} = ((x_1,y_1), (x_2,y_2,),. . . . . .(x_n,y_n))$. For the received frame $\mathbf{z}$ and a codeword $\mathbf{v} = (s_1, s_2, . . . ,s_n)$ in C, let $|\mathbf{z},\mathbf{v}|^2$ be defined as follows:

$$|\mathbf{z},\mathbf{v}|^2 = \sum_{i=1}^{n} (x_i - X(s_i))^2 + (y_i - Y(s_i))^2$$

Assume that the channel is an AWGN channel. Suppose every codeword of C is transmitted with the same probability. Then we have the following decoding rule: For a received frame $\mathbf{z}$, choose a codeword $\mathbf{v}$ in C with minimum $|\mathbf{z},\mathbf{v}|^2$. The segment $\mathbf{u}$ corresponding to $\mathbf{v}$ is then the decoded segment. This decoding rule achieves maximum likelihood decoding for C over an AWGN.

CHAPTER 3

DESIGN AND SIMULATION OF OPTIMUM DECODER FOR SOME BLOCK MODULATION CODES

In this chapter, the design of Viterbi decoder using parallel architecture for two short length block modulation codes is presented. These codes are suitable as inner code in high speed concatenated coded-modulation scheme. Since these codes have trellis diagram of reasonable complexity, high speed Viterbi decoder can be built on VLSI chip. The designs have been simulated for different quantization-levels and branch metric computation schemes and the actual coding gain over uncoded QPSK has been found. Also, the overview of a possible VLSI chip structure of the decoder for four-state code is given and various I/O pins are discussed.

3.1 DESCRIPTION OF THE CODES

Code 1: The three binary component codes C_1, C_2 and C_3 are as follows: (1) C_1 is the simple binary (8,1) repeatition code; (2) C_2 is the binary (8,7) code with all the even weight 8-tuples; (3) C_3 is the (8,8) binary code which consists of all binary 8-tuples. The minimum Hamming distances of C_1, C_2 and C_3 are $\partial_1=8$, $\partial_2=2$ and $\partial_3=1$ respectively. By using the multi-level construction method (described in chapter two), these three binary components codes

result in a linear multi-level code, $C = C_1 * C_2 * C_3$ over the 8-PSK signal set with minimum squared Euclidean distance D[C]=4, dimension K=16, length=8 and effective rate R[C]=1. This code provides 3 db asymptotic coding gain over the uncoded QPSK modulation without bandwidth expansion. It has a 4-state trellis diagram, as shown in figure 3.1, which consists of two identical parallel 2-state trellis sub-diagrams without cross connections between them. This structure suggests that the decoding of the code can be done with two 2-state Viterbi decoders to process the two trellis sub-diagrams in parallel. This implementation not only reduces the decoding complexity but also speeds up the decoding process. The code is 45^0 rotationally invariant allowing fast rysynchronization on non-coherent channel [7].

Code 2: The three binary component codes C_1, C_2 and C_3 are as follows: (1) C_1 is first-order Reed-Muller code of length 16 ((16,5) code) represented by $RM_{(4,1)}$; (2) C_2 is the binary (16,15) code with all the even weight 16-tuples represented by P_{16}; (3) C_3 is the (16,16) binary code which consists of all binary 16-tuples and is represented by V_{16}. The minimum Hamming distances of C_1, C_2 and C_3 are $\partial_1=8$, $\partial_2=2$ and $\partial_3=1$ respectively. Hence, the basic multi-level modulation code of length 16 constructed by these component codes is given by

$$C = RM_{(4,1)} * P_{16} * V_{16}$$

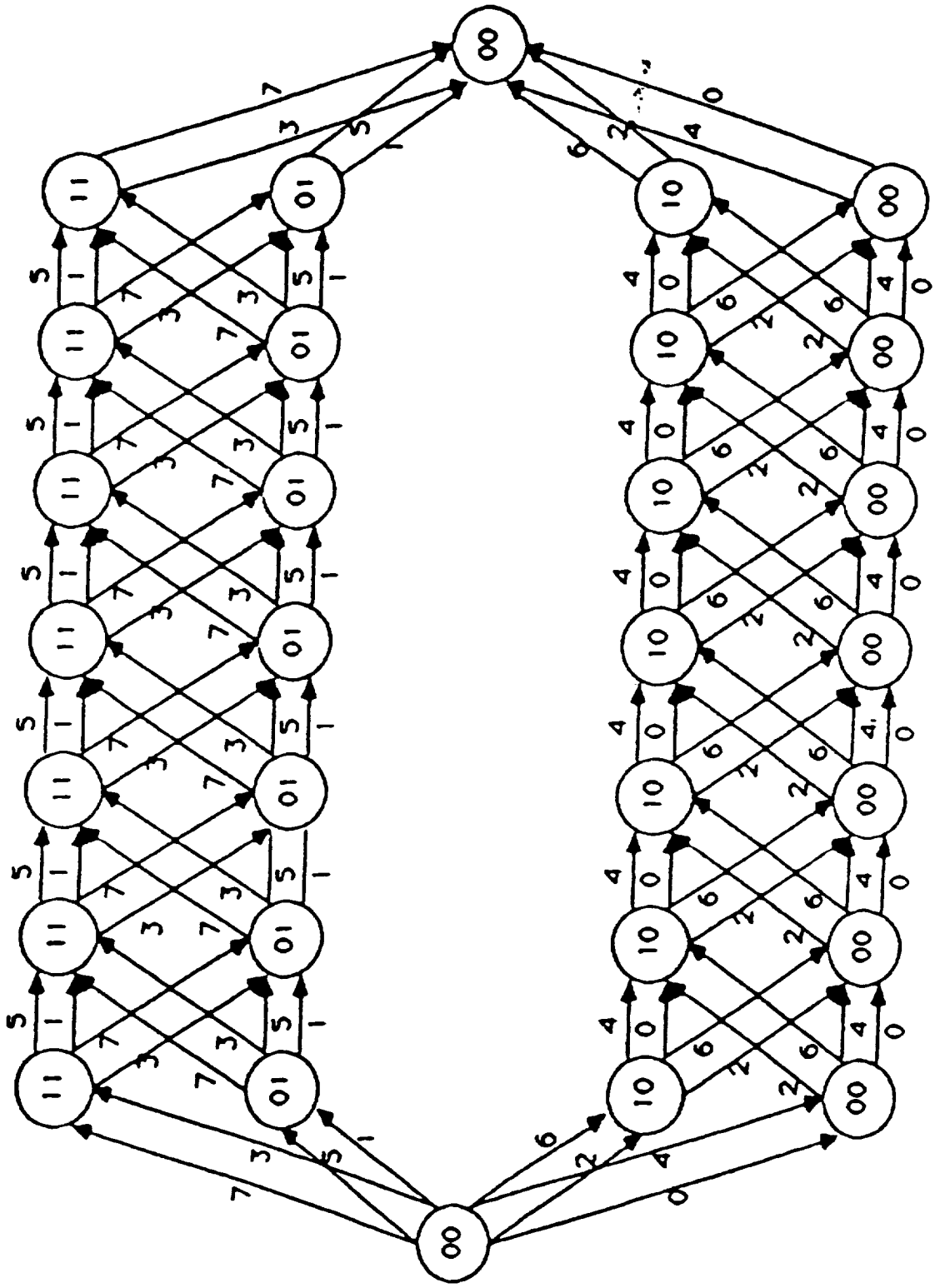


Fig. 3.1 Trellis diagram for 8-PSK modulation code1

This code has minimum squared Euclidean distance D[C]=4, dimension K=36 and effective rate R[C]= 9/8 (greater than one). This code provides 3.52 db asymptotic coding gain over the uncoded QPSK modulation with less bandwidth (a bandwidth reduction). However, It has 16-state trellis diagram, which makes the decoder more complicated.

As far as decoding is concerned, the advantages of BCM codes over TCM codes are as follows. (1) In case of BCM codes, the trellis terminates after certain number of sections. Hence there is no ambiguity about final survivor. In case of TCM codes, the decoder has to choose an appropriate survivor from certain number of survivors (one at each state) to make a decoding decision. This implies some additional overhead on the decoder. (2) Since in case of BCM codes, the entire codeword can be outputted in parallel, decoding delay is less for short length codes. In case of TCM codes, the decoding delay is 5 or more times the constraint length.

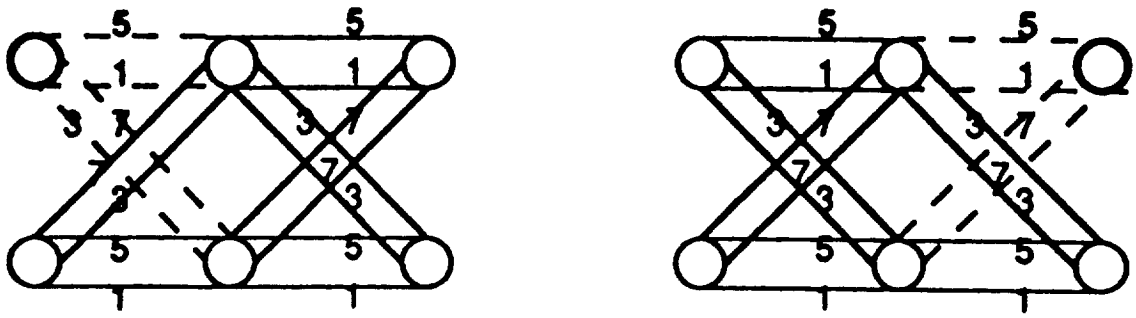
3.2 MODIFICATION OF TRELLIS DIAGRAM

For the simplicity of decoder (for high throughput) and the associated circuit, the decoder should be designed to perform identical operations in each and every section of the trellis. Hence, it is desirable to have a trellis with the same inter-connection structure in all the sections. Thus, the trellis diagram (figure 3.1)

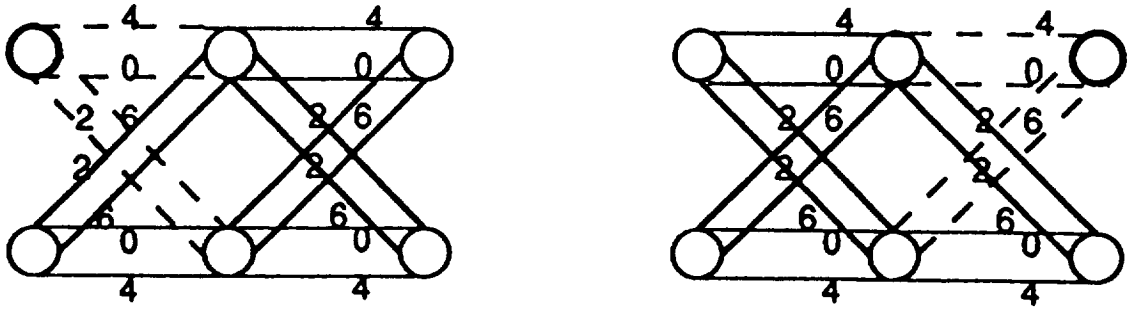
of the code needs to be modified. Note that the initial and final sections of the trellis are different from the rest of the repeatitive sections. Initial and final sections have some states and branches missing. The structure of the initial and the final sections of the trellis is modified in a way that it has the same form as th periodic sections of the trellis. The resulting trellis diagram (TD) will be referred to as the modified trellis diagram (MTD).

The MTD is obtained from the TD by introducing some additional states (or nodes) and branches in the initial and final sections of the trellis such that the inter-connection structure of the entire 8-sections trellis in the MTD has exactly the same periodic structure (figure 3.2). Note that since the initial and final states are always the zero states, the encoder can never be in the additional states introduced in the MTD. Therefore, these additional states are invalid states while the states present in the TD are valid states. Similarly, the additional branches introduced in the MTD are invalid branches while the branches present in the TD are valid branches. The decoder can never decode to a path which contains an invalid state or an invalid branch. There must be a mechanism built into the decoder to accomplish it.

We need to modify the initialization step of Viterbi algorithm. That is, initialize the metric value of the valid state at stage 0 to zero and the metric value of the invalid state at stage 0



Modified Upper Trellis



Modified Lower Trellis

Invalid branches have been shown by dotted lines and invalid states have been shown by bold circles.

Figure 3.2 Modified Trellis

to $b_{max} + e$, where b_{max} is the largest value of the branch metric and $e > 0$. The idea is to make sure that when a path originating from the invalid state and the one originating from the valid state are compared at the next stage i=1, the one originating from the valid state is always the survivor. This can be shown as follows:

Let us consider the metric values of various paths converging to state 1 at stage i=1. Let p1 be the metric value of the path originating from the valid state and converging to state 1 at stage i=1. Hence, p1 = initial metric value + branch metric value = 0 + bm1, where bm1 is the smaller of the branch metrics of two parallel branches converging to this state from the valid state. Let p2 be the metric value of the path originating from the invalid state and converging to state 1 at stage i=1. Hence, p2 = initial metric value + branch metric value = b_{max} + e + bm2, where bm2 is the smaller of the branch metrics of two parallel branches converging to this state from the invalid state. Now, p2 - p1= b_{max} + e + (bm2 - bm1). Since $(bm2 - bm1)_{min} = -b_{max}$ and $e > 0$, p2 - p1 > 0 or p1 < p2 for all possible values of bm2 and bm1. Hence, the path originating from the valid state is the survivor.

Thus for the initialization of the metric values of the various states as above, all invalid states and the branches introduced in the initial section of the trellis are eliminated at stage i=1.

At stage i=7, the last section of the trellis, we simply discard the invalid state, because we know that the final survivor

is at other state (valid state). Thus introduction of invalid state and branches in the final section of the trellis does not have any affect on the decoder or decoding decision.

The above modification in the trellis diagram eliminates the need of some additional hardware in decoder as well as some overhead in decoding operations, which is otherwise required to process the irregular sections (the initial and the final sections) of the trellis by proper initialization of state or path metric registers. Note that the above modification is significant for the simplicity of implementation of decoder for a code with more complex trellis.

3.3 DECODER DESIGN FOR THE FOUR-STATE CODE

As mentioned earlier, the entire decoder can be implemented as two identical 2-state decoders, one to process the upper trellis and other one to process the lower trellis. Hence, first the design of the decoder to process the upper trellis is presented and the design of the decoder to process the lower trellis will exactly be the same except '*symbol-mapping circuit* ' (discussed later).

The complete decoder design can be discussed in the following five parts:

1. Quantization and branch metric computation scheme

2. **Add-compare-select (ACS) circuit design**

3. **Information sequence updating and storage**

4. **Outputting the decoded word**

5. **Control circuit design**

3.3.1 QUANTIZATION & BRANCH METRIC COMPUTATION SCHEME

For a coded-modulation system, the appropriate metric is the squared Euclidean distance between the received symbol and the symbol in the trellis.

Let S = {0,1,2,3,4,5,6,7}. The elements in S represent the 8-signal points in an 8-PSK signal constellation. For any i in S, define

$$X(i) = \cos(2\pi i/8), \qquad Y(i) = \sin(2\pi i/8)$$

Let (X,Y) be the normalized output of a coherent 8-PSK demodulator for a received symbol, where X and Y are in-phase and quadrature components respectively. Then the ith branch metric is

$$M(i) = (X-X(i))^2 + (Y-Y(i))^2$$

for $0 \leq i \leq 7$.

Hence for each received symbol, 8 branch metrics are to be computed.

We assume that the output of the demodulator, that is X and Y components are given. Since the modulation is 8-PSK, quantization with more than three bits was considered. Simulation results (figure 3.10 (a)) showed that the four and five bits quantization degrades the performance by 0.40 and 0.15 db respectively. Hence, 5-bits quantization was chosen for the design. The price of using higher level quantization is a larger memory required to store the branch metric (in case ROM look-up table is used for branch metric computation). If the n-bits quantization is used, a ROM of capacity $8 \cdot 2^{2n}$ words (length of the word determined by maximum value of branch metric value) is required for branch metric computation.

X and Y components of the demodulator output are quantized using five-bit (32-levels) uniform quantizer. The range of quantization is determined by type of channel and signal-to-noise ratio (SNR) and found as follows:

Suppose the operating SNR per symbol of the system is 9 db and the channel is AWGN channel. Then,

$10 \log_{10}(S/N)=9$ db where, S=signal power$=(1/2)A^2$

N=noise power$= s^2$

Hence, $s=0.25\ A$

Since four times s covers more than 96% of the cases, hence we choose the range of quantization from -1.5A to 1.5A, where A is noise-free received signal amplitude.

By quantizing X and Y components of the received symbol into 32 levels each, we divide the two-dimensional Euclidean plane into 1024 squares (rectangular quantization). The received 8-PSK symbol will fall into one of these 1024 squares.

Note that the computation of one branch metric requires two multiplications and one addition. Eight such computations are required during each decoding cycle. Multiplication is time consuming. We can speed-up the metrics computation if we compute all the metrics in parallel using fast and parallel multipliers. But then the hardware complexity is going to increase tremendously. Hence, for higher speed of decoder as well as to reduce the hardware complexity, ROM look-up table is used for branch metric computation.

For each 8-PSK symbol, the 1024 squared Euclidean distances corresponding to all possible received signal points in two dimensional Euclidean plane have been computed. These distances are floating point numbers between 0 and 9.5. For high throughput and reasonable hardware complexity, it is necessary to design the Viterbi decoder which handles only integer and performs only integer arithmetic. Hence, we need to map these floating point

numbers into integer without allowing much degradation in performance.

One straight forward way to do this mapping is multiply each floating point number by some appropriate integer and then take the nearest integer value. Simulation showed that if we multiply each floating point number by 5 and then take the nearest integer value, this results in negligible degradation in performance. Essentially, by doing so we are able to distinguish between the distance metrics x and x + 0.2, where $0 \leq x < 9.5$.

The problem with the above uniform mapping scheme is that it leads to increase in the branch metric range and hence the number of bits required to represent the branch and path metric is increased. Consequently, more hardware is required for all the units (adders, comparators, multiplexers etc.) in the ACS part of the circuit. For example, for high speed decoder, carry look-ahead adders are used, hardware requirement of which increases more than in proportion for increase in each additional bit. Using above mentioned mapping scheme, the maximum value of branch metric came out to be 47, which requires 6 bits to represent a branch metric and 7 bits to represent a path metric.

Since reducing the branch and path metrics range even by one bit means a considerable savings in hardware, the following non-uniform mapping of floating point branch metric value to integer value is proposed:

The branch metrics with lower values, which are more likely to be the survivor are mapped with smaller range (essentially finer distinction) and those metrics with larger values are mapped with bigger range. The idea is, since the branch metrics with larger values are unlikely to be the survivor, we do not need to make fine distinction between two larger values of branch metrics.

We can divide the entire metric values (range 0-9.5) into two groups: the first group contains lower metric values in the range 0-4 and the second group consists of metrics with the values in the range 4-9.5. The floating point metric value in the first group is multiplied by 5 and then rounded off to the nearest integer value. The floating point metric value in the second group is assigned the next available integer with an interval of 0.5. Table 3.1 illustrates the non-uniform mapping.

The mathematical formulation for the above non-uniform mapping scheme can be given as follows:

Let F be the function that maps the floating point branch metric value to an integer value. Then F is given by,

$$F(x) = i, \qquad i*0.2 \le x \le (i+1)*0.2, \qquad 0 \le i \le 19$$

$$= 20 + i, \quad 4+i*0.5 \le x \le 4+(i+1)*0.5, \qquad 0 \le i \le 11$$

Table 3.1 Non-uniform mapping of branch metric values

Floating point value	Integer value
0.0-0.2	0
0.2-0.4	1
0.4-0.6	2
.....	..
.....	..
3.6-3.8	18
3.8-4.0	19
4.0-4.5	20
4.5-5.0	21
.....	..
.....	..
9.0-9.5	31

Thus, we limit the branch metric range to 31 and hence only 5 bits are required to represent the branch metric value. Similarly, path metric registers and all other related units for ACS operation now need 6 bits instead of 7. Two schemes were simulated. The non-uniform mapping scheme reduced the number of bits by one, thus reducing the hardware and gave negligible performance degradation over uniform mapping scheme.

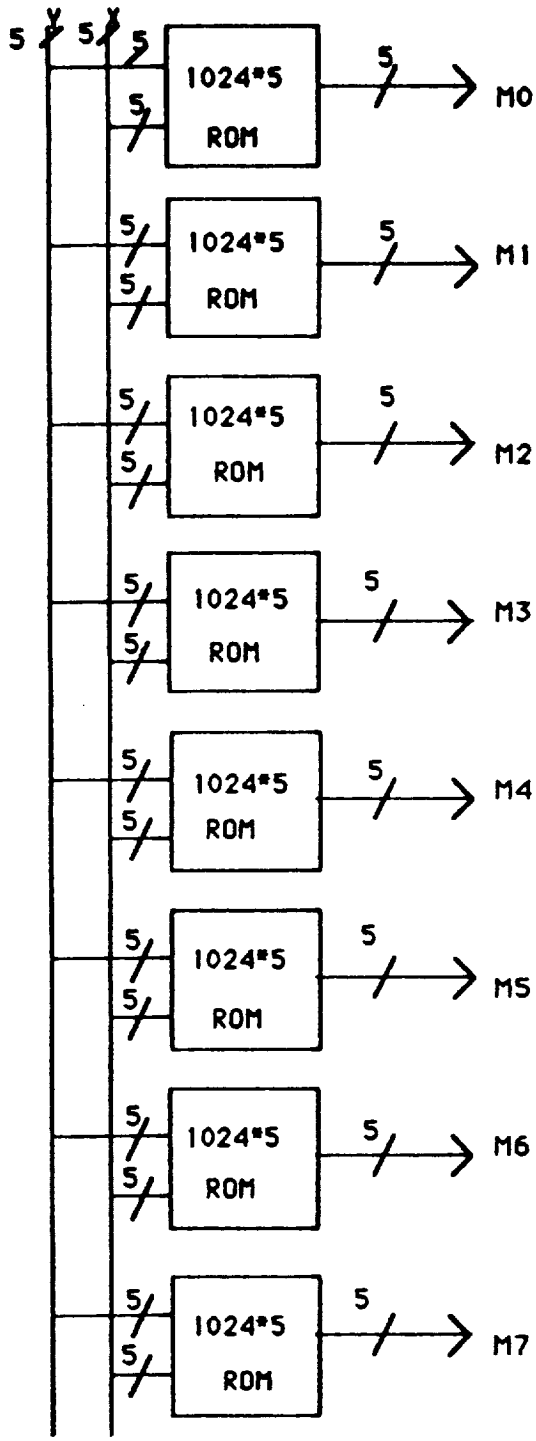


Fig. 3.3 Branch metric computation scheme

The 1024 squared Euclidean distances for a particular 8-PSK signal point is stored in a ROM whose storage capacity is 1024 words, each word consisting of 5 bits. We need 8 such ROMs for 8 signal points. Hence, total storage requirement is 8•1024•5 bits = 40 kbits of ROM.

The scheme for branch metric computation has been shown in figure 3.3. X and Y (5 bits each) are quantized value of in-phase and quadrature components of demodulator output. They act as address input to ROM's. A separate ROM is being used for each branch metric. The output of each ROM is a 5-bit word which is branch metric value and is passed to the ACS unit.

3.3.2 ACS CIRCUIT DESIGN

ACS circuit is the central unit of Viterbi decoder. It consists of shift registers with (parallel loading facility), adders, comparators (or substractors) and multiplexers.

From the trellis diagram (figure 3.1), note that at each time unit, there are four paths converging to each state requiring four adders (to generate the four path metrics in parallel) and three comparators (to compare these four metrics in parallel) for that state for parallel ACS operation. We show that if we divide the ACS operations in two steps, we can save in hardware.

In the first step, two branch metrics corresponding to pair of parallel branches which originates from the same previous state are compared and one with larger metric value is discarded. This results in generation of only two new paths at each state at each time unit. Hence, there remains only two path metrics to be compared at each state which is done in the second step of ACS operation. This scheme leads to reduction of one comparator and two adders per state which is considerable saving in hardware.

Because of addition, comparison and multiplexing operations, certain minimum time is required for ACS operations. Hence, proper number of clock cycles must be allowed for entire ACS operations and control circuit must take care of the delay required.

The detailed ACS circuit diagram has been shown in figure 3.4 (a) and 3.4 (b). R1 to R4 are 5 bit registers with parallel loading facility which contain four branch metrics for upper trellis. C1 and C2 are 5 bits comparators. MUX1 and MUX2 are 2 to 1 five bits multiplexers. The circuit in figure 3.4 (a) executes the first step of ACS operation. It compares the branch metrics of the parallel branches in the trellis and selects the one with the lower metric value.

Simulation results showed that if we limit the branch metric value to 5 bits (maximum 31), the maximum path metric value at BER 10^{-5} and 10^{-6} is 50 and does not exceed 52 at higher BER.

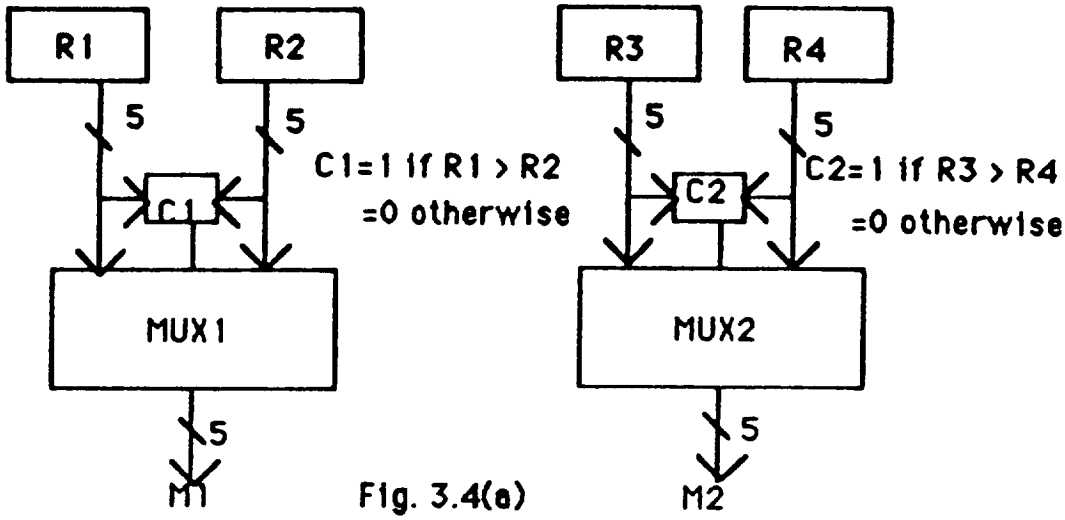


Fig. 3.4(a)

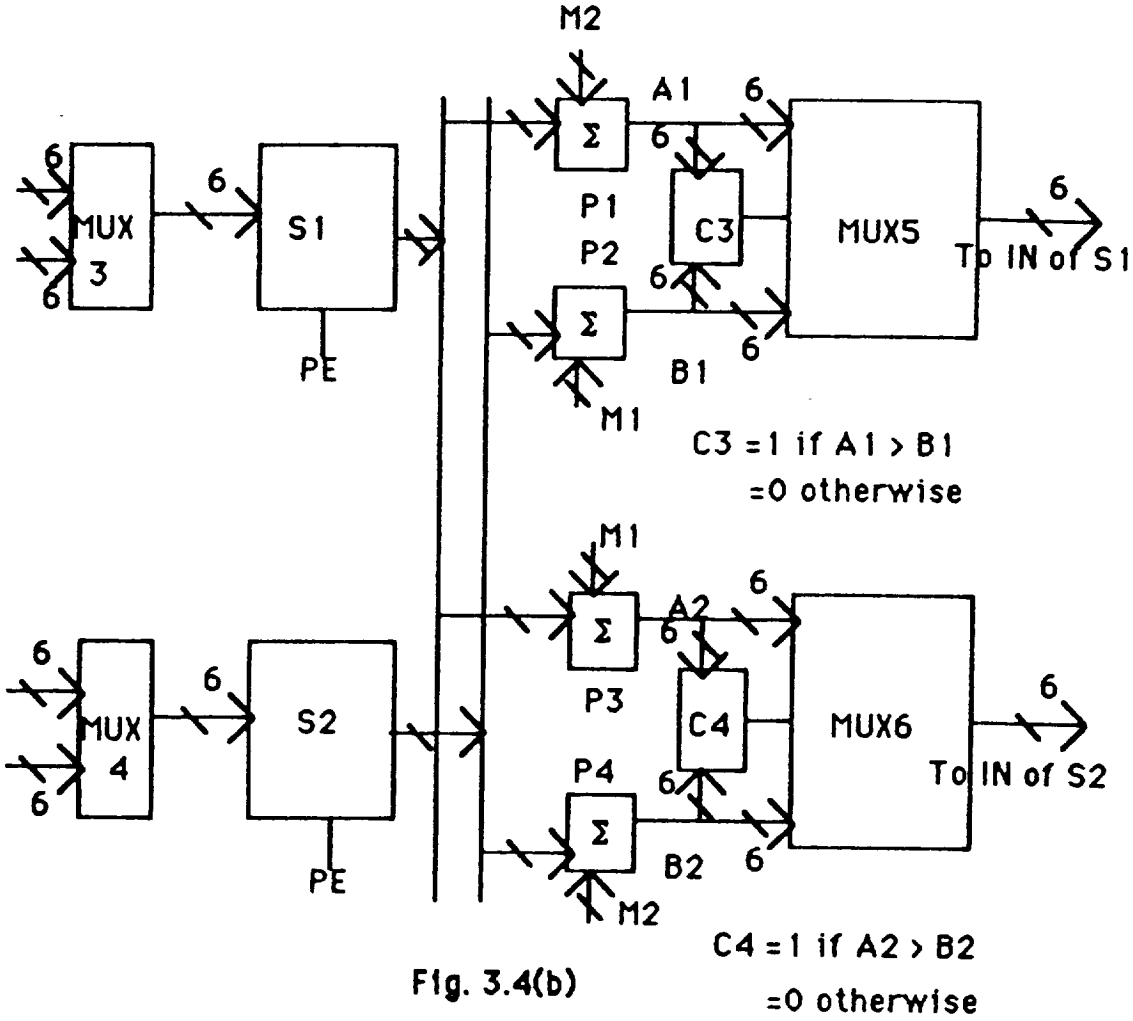


Fig. 3.4(b)

ACS CIRCUIT

Hence, path metric registers (S1,S2) should be 6 bits registers and associated adders, comparators and multiplexers all must be 6 bits units. Based on simulation results, we can safely assume that at acceptable BER, there will be no overflow in the ACS section of the decoder. Note that the metric normalization circuit (to avoid overflow) is not needed since the trellis terminates after eight sections.

The circuit in figure 3.4 (b) executes the second step of ACS operation, that is generation of new paths, comparison and selection. S1 and S2 are 6 bits state or path metric registers for state one and state two respectively. P1 to P4 are 6 bits adders. For high speed decoder, carry look-ahead adders should be used. C3 and C4 are 6 bits comparators. MUX3, MUX4, MUX5 and MUX6 are 6 bits 2 to 1 multiplexers. The output of MUX5 and MUX6 (which contain survivor path metrics after each ACS operation) are connected to the input of path metric registers through MUX3 and MUX4 respectively. The purpose of MUX3 and MUX4 is to select from the initial value (0 or 32) of path metrics (in the beginning of decoding of a new code word) or survivor path metric after each ACS operation (rest of the time). Note that the comparators can be replaced by fast substractors to further improve the speed of ACS operation.

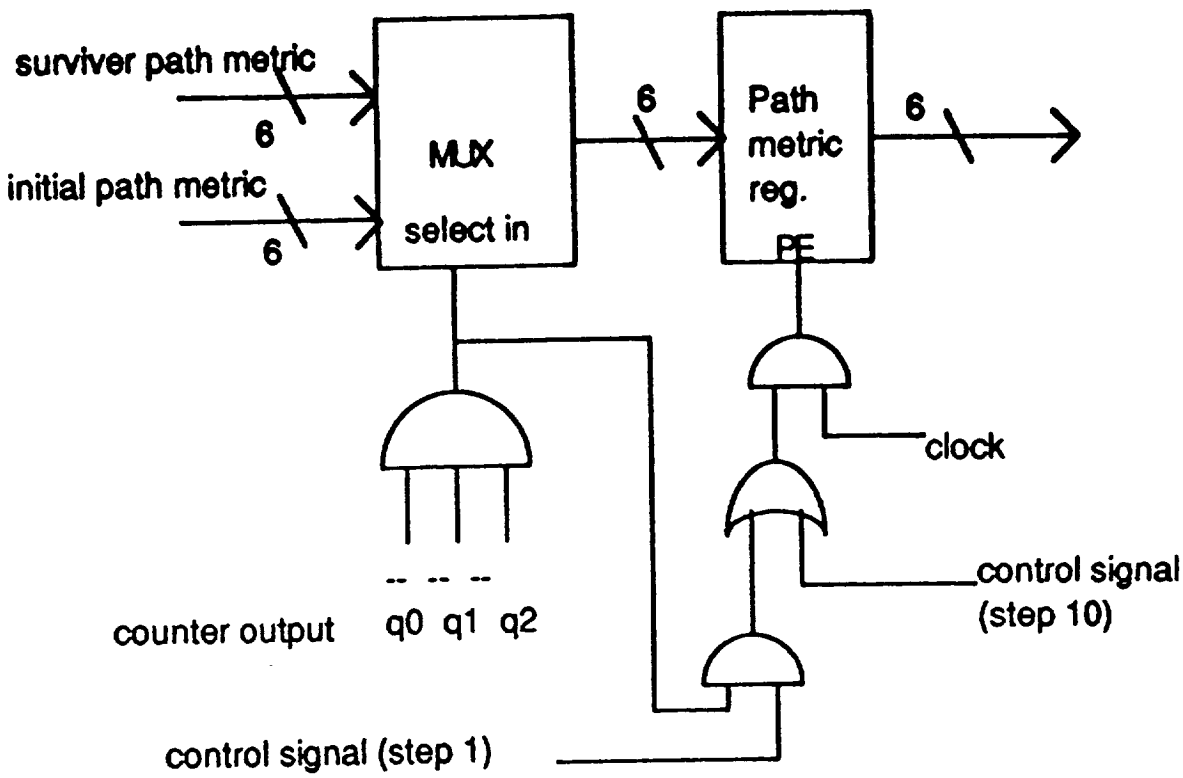


Fig. 3.5 Input connection of path metric register

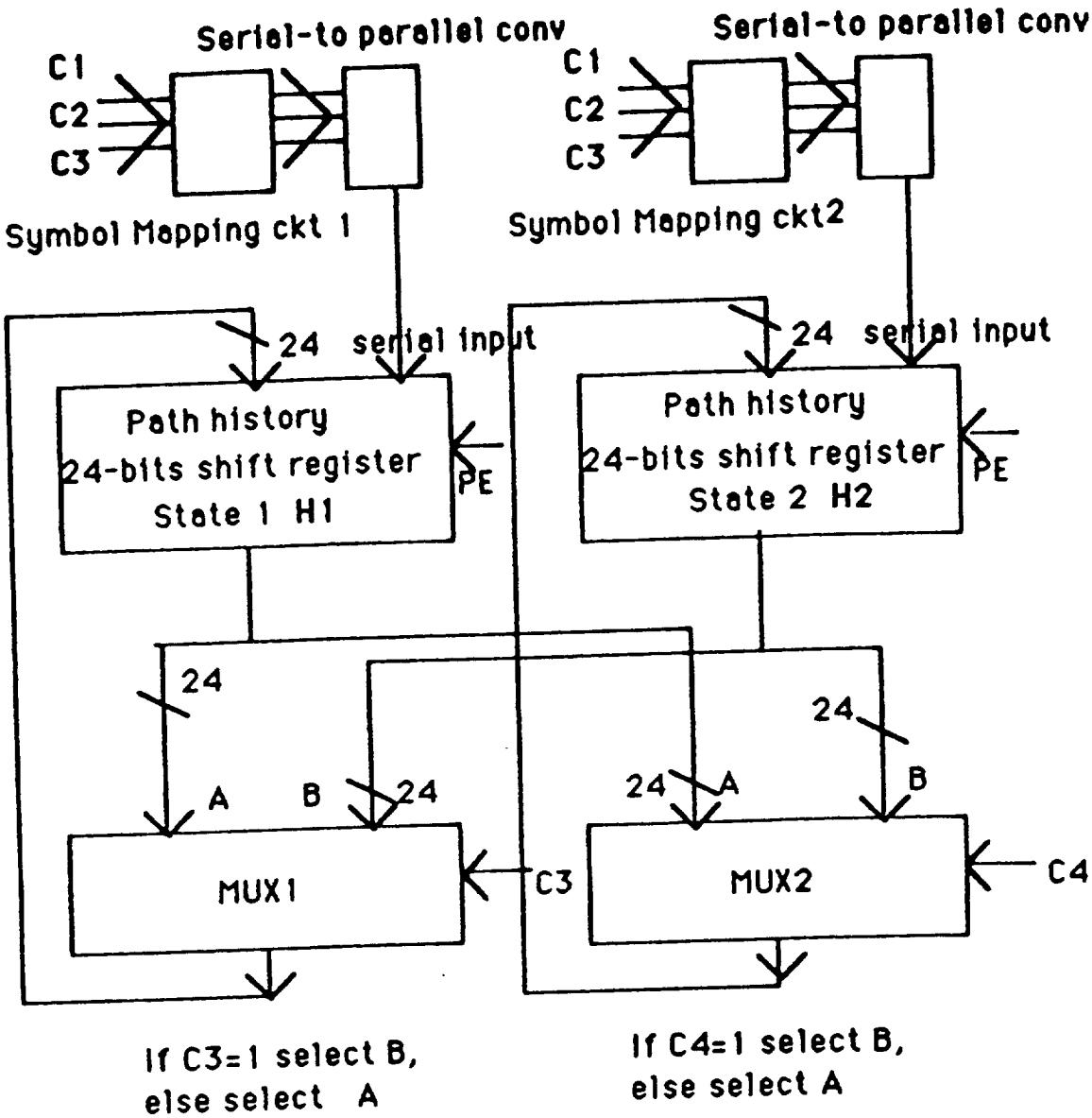
The complete input connection for the path metric register has been shown in figure 3.5. Whenever the count (which keeps track of the number of section being processed in the trellis) is 000, that is the start of the decoding of a new code word. Hence, multiplexer (MUX3 and MUX4) selects initial value of path metrics (0 and 32 corresponding to valid and invalid states respectively)

3.3.3 INFORMATION SEQUENCE UPDATING AND STORAGE

The register exchange method is used for information sequence updating and storage. The registers in which these sequences are stored are interconnected in precisely the same fashion as the ACS circuit. Each time a new branch is processed, the registers are interchanged corresponding to which sequence survives the comparison and a new symbol is added at one end of each registers.

The complete circuit-diagram for this part of the decoder has been shown in figure 3.6. H1 and H2 are two 24 bits path history registers (for storage of 8 symbols each 3 bits long) with parallel loading as well as serial shifting facility. MUX1 and MUX2 are 2 to 1 24 bits multiplexers. Based on the results of comparators C3 and C4 in the ACS section, information sequences are updated in registers H1 and H2.

Depending upon the results of comparators C1,C2,C3 and C4 (in ACS section), the new symbols corresponding to the survived branch are generated by *symbol-mapping circuits.* The complete truth-table and logic design of these circuits have been shown in figure 3.7. Newly generated symbols are parallely loaded to buffer registers (serial-to-parallel converters) and then serially shifted to path history registers H1 and H2.



C1,C2,C3 and C4 are comparators (in ACS section) outputs.

Fig. 3.6 Path history updating by Register Exchange Method

SYMBOL MAPPING CIRCUIT DESIGN FOR UPPER TRELLIS

STATE 1

COMPARATORS		OUTPUT	SURVIVED		SYMBOL
C1	C2	C3	Q2	Q1	Q0
0	X	1	1	1	1
1	X	1	0	1	1
X	1	0	0	0	1
X	0	0	1	0	1

Q0= HIGH
Q1=C3
Q2=NOT(C1) C3 + NOT(C2) NOT(C2)

STATE 2

COMPARATORS		OUTPUT	SURVIVED		SYMBOL
C1	C2	C4	Q2	Q1	Q0
X	1	1	0	0	1
X	0	1	1	0	1
1	X	0	0	1	1
0	X	0	1	1	1

Q0= HIGH
Q1= NOT C4
Q2=NOT(C2) C4 + NOT(C1) NOT(C2)

X = Do not care state
NOT = Inverse

Figure 3.7

3.3.4 OUTPUTTING DECODED WORD

After 8 branches have been processed (which is indicated by count 111), the path metrics for state one (valid state) of decoders for the upper and lower trellis are compared and appropriate path history register content is outputted. This is indicated to the user by making the output ready line HIGH. The diagram for this part of the decoder has been shown in figure 3.8. S1 and S2' are path metric registers of state 1 from decoder1 and decoder2 respectively.

Also, if the path metric value of the final survivor exceeds some predefined value (determined by acceptable BER), then it can be detected by a threshold circuit and an indication can be given to the user whether the decoded code word is unreliable or safely acceptable. This provision of error detection or reliability indicator for decoded code word can be included in the output circuit. Simulation studies showed that the maximum path metric values of the final survivor for BER 10^{-5} and 10^{-6} are 8 and 6 respectively.

3.3.5 CONTROL DESIGN

We have 8 clock (recovered from the received symbols) cycles to (1) initialize the path metric registers, (2) process 8 sections of

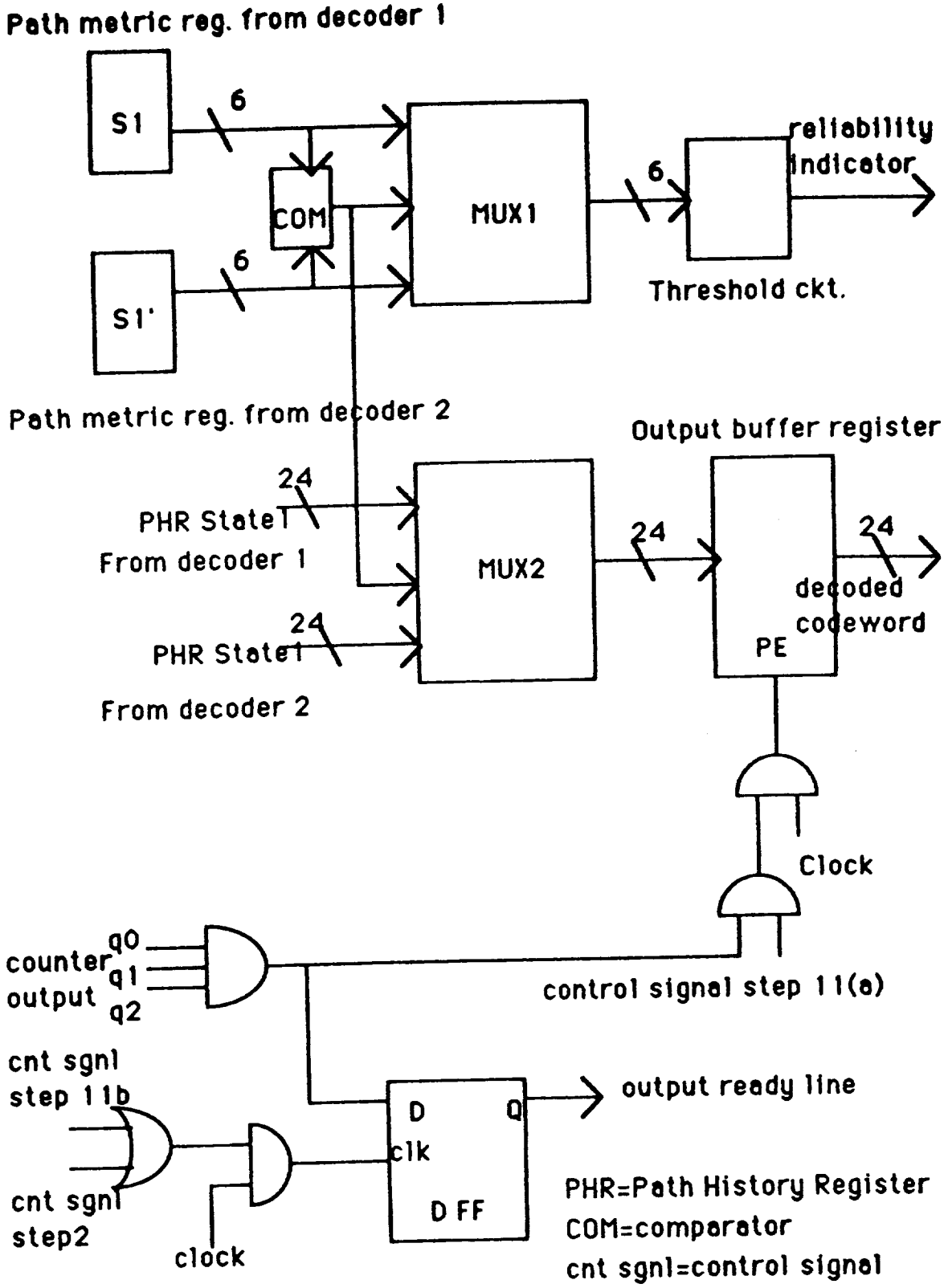


Fig. 3.8 Output Circuit

the trellis, and (3) output the decoded word and other required signals. A modulo-8 counter is used to keep track of the number of sections of the trellis processed. A separate input-line must be provided to reset the counter externally, whenever necessary.

During each clock cycle of the received symbol, all the following operations must be performed and corresponding control signals must be generated by the control-circuit.

step 1: The A/D converters makes available the quantized value of X and Y components of the 8-PSK demodulator output at the address input-line of ROMs and a start pulse initiates the control circuit.

step 2: If count is 000, initialize the state or path metric registers. (load 0 and 32 in two path metric registers by applying control signal to parallel enable input of these registers)

Also, make the 'output ready line' LOW and 'output error line' too, if HIGH, LOW.

step 3: ROMs read cycle. At the end of this cycle, branch metrics are available at the output lines of ROMs.

step 4 Load branch metric registers R1 to R4. Control signal is applied to parallel enable input of registers R1 to R4.

step 5: Delay cycle. (In order to allow sufficient time for entire ACS operations)

step 6: Register exchange cycle in path history section. (control signal is applied to parallel enable input of buffer registers for new survived symbol (generated by symbol mapping circuit) and parallel enable input of path history registers)

step 7: Serial shift of 1st bit of new symbol from buffer register to path history register.

step 8: Serial shift of 2nd bit of new symbol from buffer register to path history register.

step 9: Serial shift of 3rd bit of new symbol from buffer register to path history register.

step 10: Update the state metric registers. (Control signal is applied to parallel enable input of path metric registers)

step 11: If count is 111, (a) output the decoded code word. (control signal to parallel enable input of output buffer register) (b) Make the output ready line HIGH. Also, if final survivor path metric exceeds some limit (detected by threshold circuit), another output line should be made HIGH indicating that decoded word should not be accepted.

step 12: Increment the counter (by applying control signal to clock input of counter) and wait for the next start pulse to process the next section in the trellis.

DECODER FOR LOWER TRELLIS

SYMBOL MAPPING CIRCUIT DESIGN FOR LOWER TRELLIS

STATE 1

COMPARATORS		OUTPUT	SURVIVED		SYMBOL
C1	C2	C3	Q2	Q1	
Q0					
0	X	1	1	1	0
1	X	1	0	1	0
X	1	0	0	0	0
X	0	0	1	0	0

Q0= LOW

Q1=C3

Q2=NOT(C1) C3 + NOT(C2) NOT(C3)

STATE 2

COMPARATORS		OUTPUT	SURVIVED		SYMBOL
C1	C2	C4	Q2	Q1	
Q0					
X	1	1	0	0	0
X	0	1	1	0	0
1	X	0	0	1	0
0	X	0	1	1	
0					

Q0= LOW

Q1= NOT C4

Q2=NOT(C2) C4 + NOT(C1) NOT(C2)

Figure 3.9

The entire circuit for the decoder to process the lower trellis is exactly the same as the one to process the upper trellis discussed so far except for the symbol mapping circuit design which is shown in figure 3.9. These two decoders can share the same control circuit if built on a single chip.

3.4 SIMULATION RESULTS FOR 4-STATE CODE

The simulation results are shown in figure 3.10(a) and 3.10(b). Note that in case of ideal Viterbi decoder (no quantization and no floating point to integer conversion of branch metrics, hence decoder handles floating point arithmetics), the actual coding gain is 1.5 db over the uncoded QPSK (with grey code indexing) at BER of 10^{-5} without bandwidth expansion.

4-bit quantization results in 0.40 db loss whereas 5-bit quantization results in 0.15-0.20 db loss. Also, for floating point to integer conversion of branch metric values, the difference in performance between multiplication factors 4 and 5 (named as uniform mapping 2 and uniform mapping 1 respectively in the figure 3.10(b)) is 0.3 db. The multiplication factor 5 was found good enough. The factor 4 results in maximum integer branch metric equal to 37, whereas the factor 5 results in maximum integer branch metric equal to 47. Also, note that the difference in

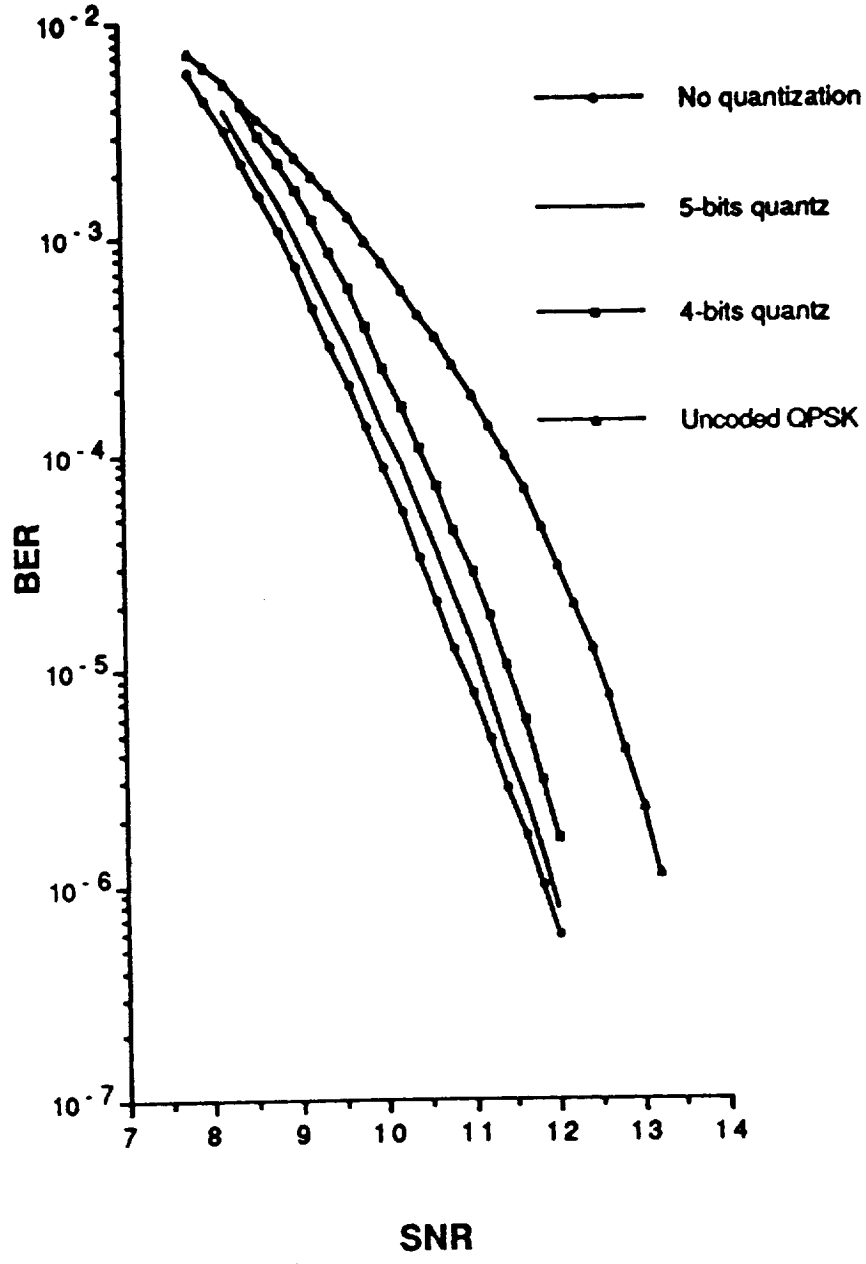


Fig. 3.10 (a) Error performance of 4-state code for different quantization-levels

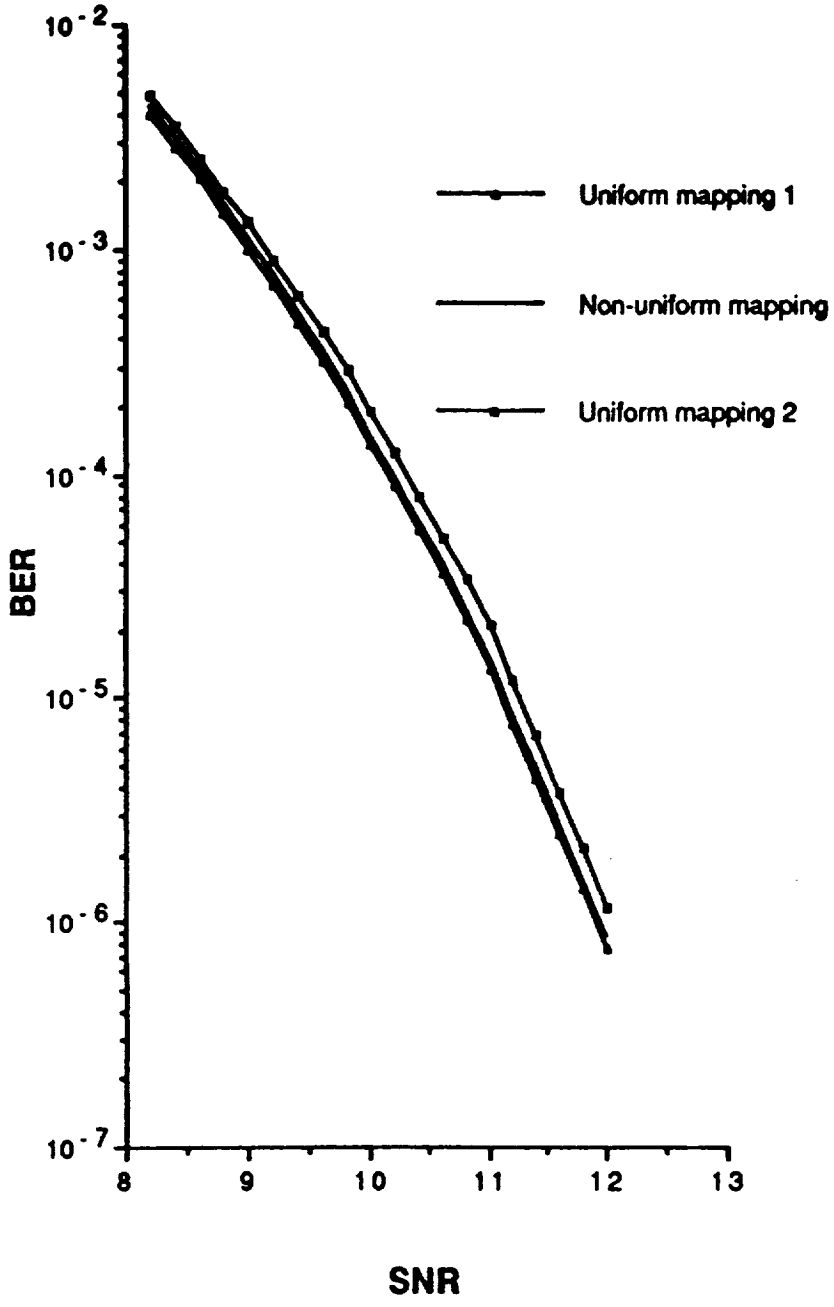


Fig. 3.10(b) Error performance for different branch metric computation schemes

performance between uniform and non-uniform floating point to integer mapping is negligibly small.

3.5 CHIP STRUCTURE

A possible chip layout structure for the Viterbi decoder for 4-state code has been shown in figure 3.11. The input and output lines description are as follows:

IN 1-10: Output from A/D converter (quantized value of normalized X and Y components of demodulator output), address input for ROMs.

IN 11: Input line to reset the counter externally whenever necessary and restart the decoding. Hence, this line provides external word synchronization.

IN 12: Input line to give a start pulse to the control circuit. Each time a new symbol is received and digitized by A/D converter, the latter gives a start pulse to the decoder through this line to process the next section in the trellis.

IN 13: Input line for high frequency clock pulse for control circuit.

IN 14-15: Power and ground lines.

OUT 16-39: Decoded code word is available on these output lines.

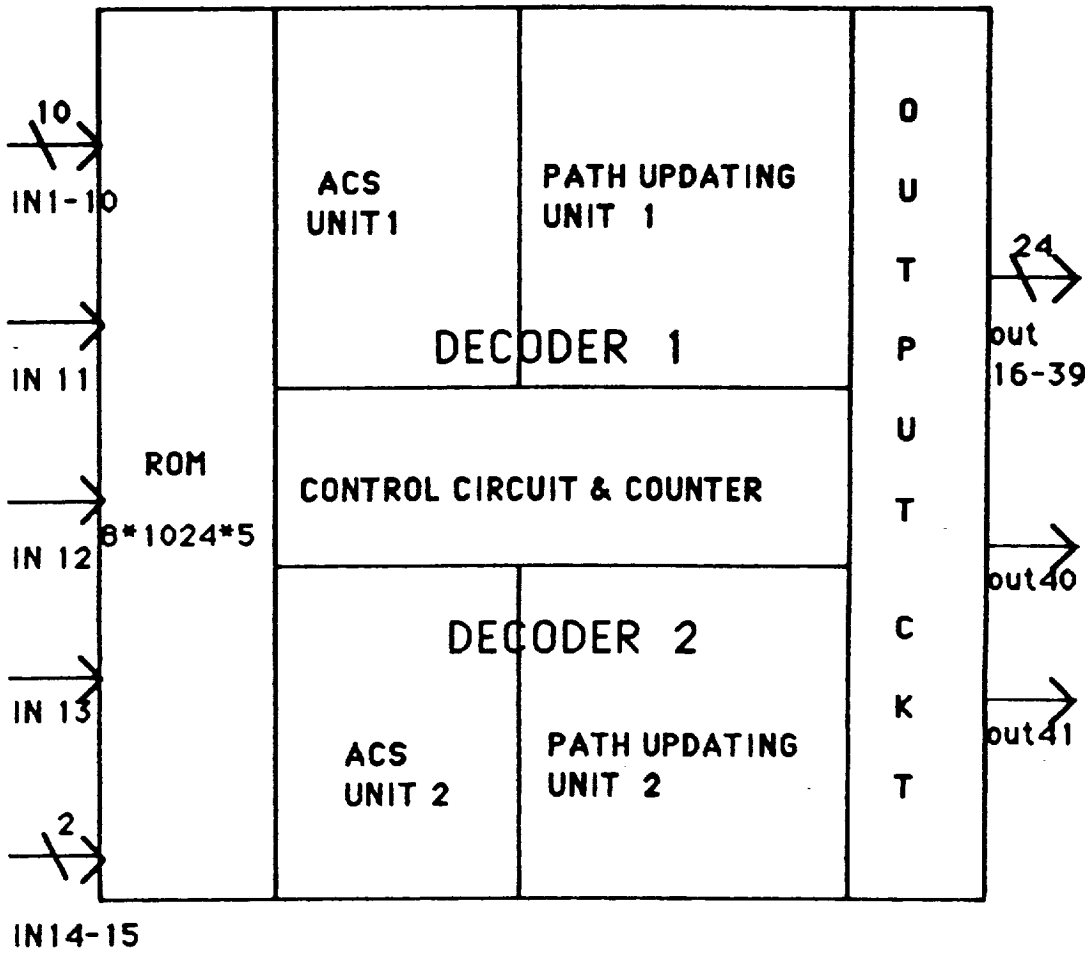


fig.3.11 Overview of chip structure of decoder for 4-state code

OUT 40: This output line goes HIGH whenever a new code word is available on the output line 12-35.

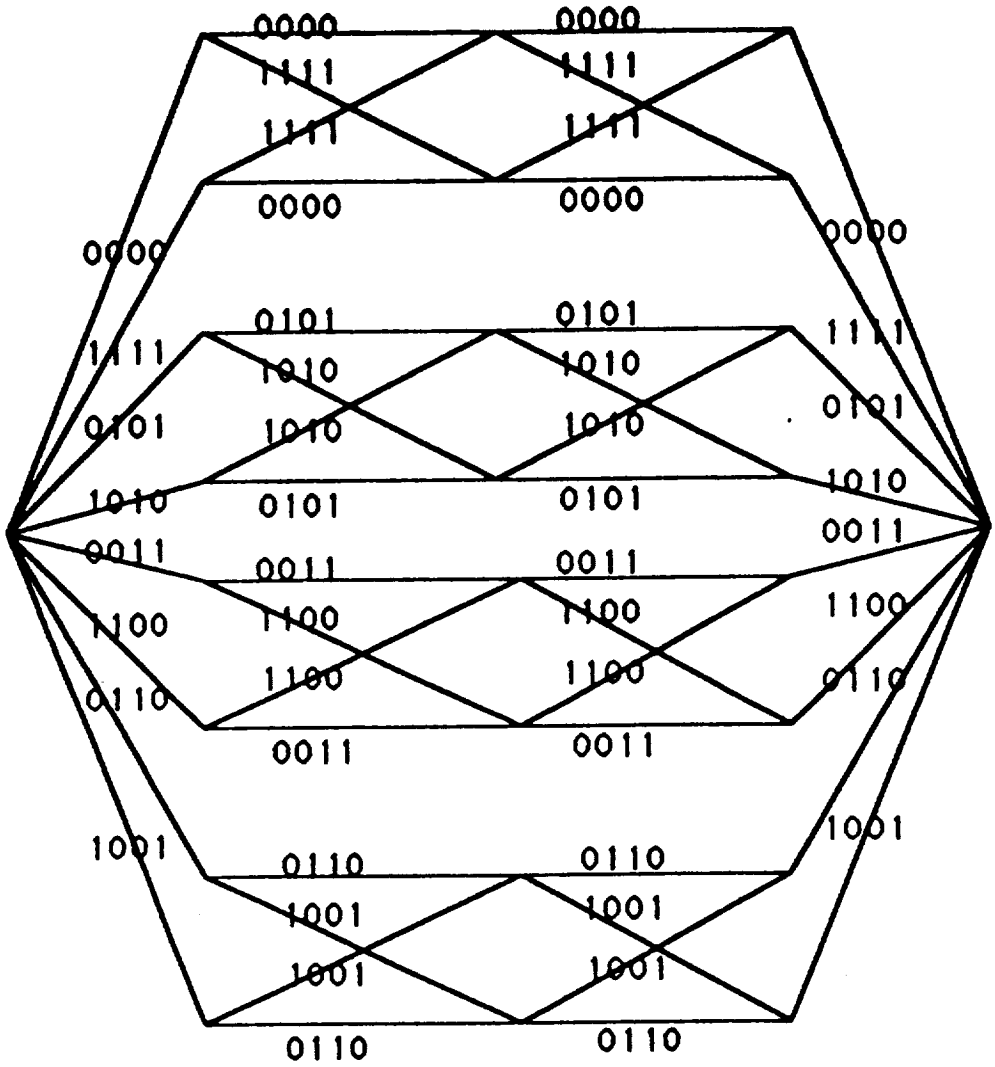
OUT 41: This line can act as a possible error indicator. It goes HIGH whenever the path metric value of final survivor exceeds some pre-defined value.

Complete decoder including ROM look-up table for branch metric can be built on a single VLSI chip but the chip size will be very big. Note that the number of pins in the chip can be reduced if the decoded code word is outputted serially (one symbol at a time). If the two sub-decoders are integrated on single chip, they share the same control circuit.

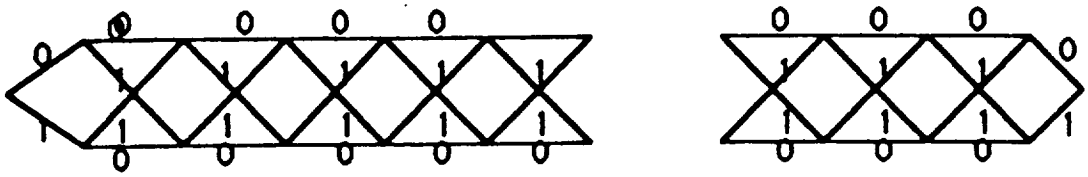
3.6 THE DECODER DESIGN FOR 16-STATE CODE

3.6.1 DESCRIPTION OF THE TRELLIS DIAGRAM

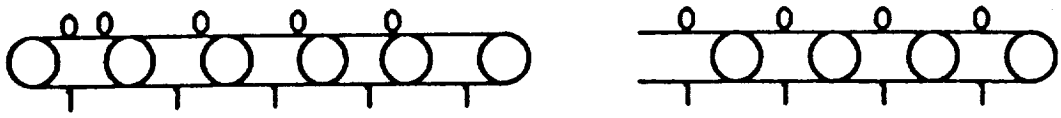
The trellis diagrams of the component codes for this modulation code has been shown in figure 3.12. The overall trellis diagram for the modulation code consists of four identical parallel 4-state special kind of trellis sub-diagrams without cross-connections among them. The trellis sub-diagrams (with modification for ease of implementation) are shown in figure 3.13. Each trellis sub-diagram is a special kind of four-section trellis such that each branch itself is a four-sections, 2-state small



4 sections trellis for (16,5) RM code



16 sections trellis diagram for (16,15) even weight code



16 sections trellis diagram for (16,16) code

Fig. 3.12 Trellis diagrams for the component codes

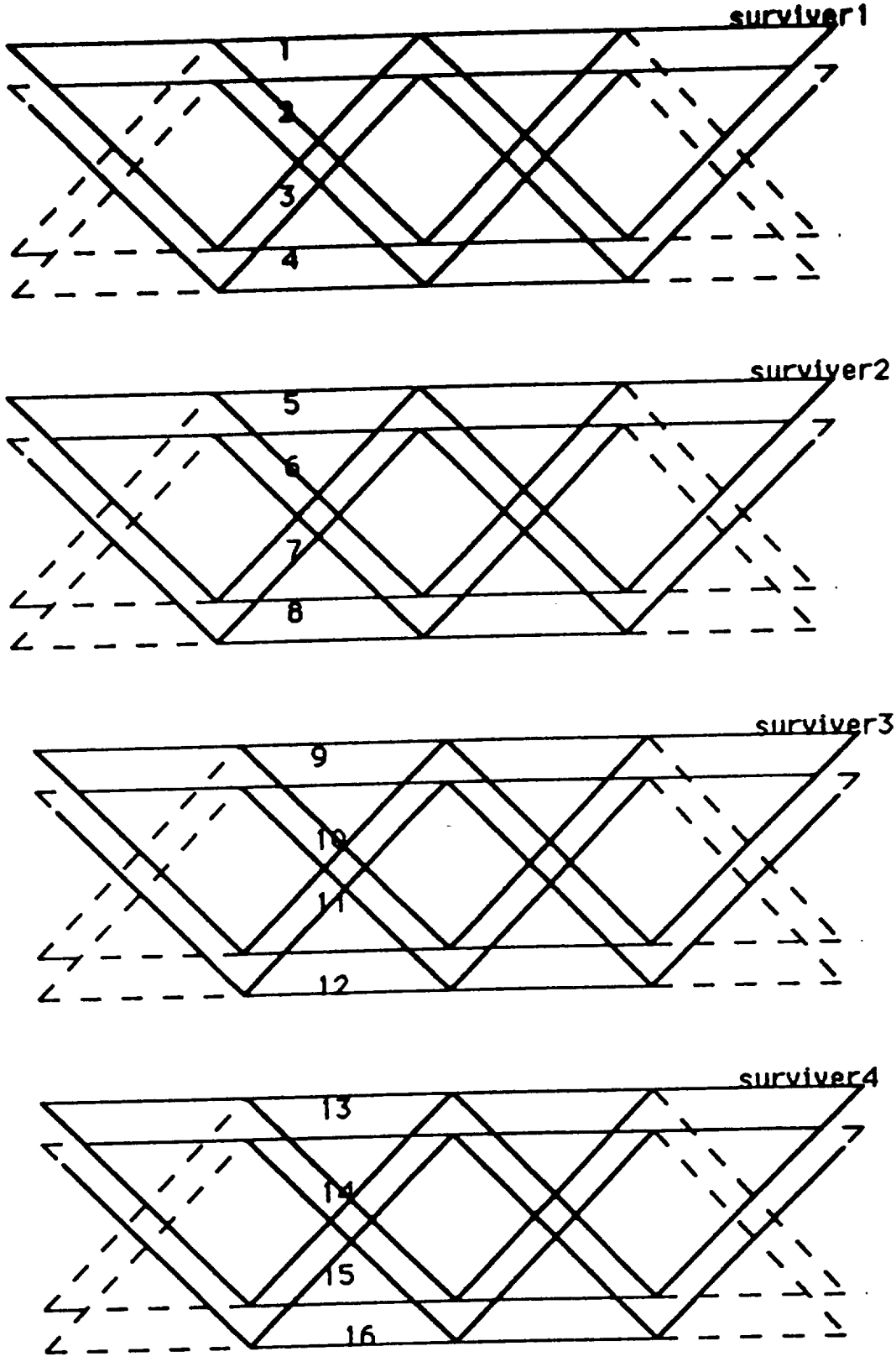


Fig.3.13 Trellis for the modulation code

trellis. The two parallel lines in the diagram represent a 2-state, 4-sections small trellis. Note that the dotted lines correspond to the invalid branches. At stage 1 and stage 16 of decoding process, there is only one valid state in each sub-diagram, that is the state1 of first small trellis in each trellis sub-diagram.

The diagram of one such small trellis, which constitutes the branch of the trellis sub-diagram, is shown in figure 3.14. These

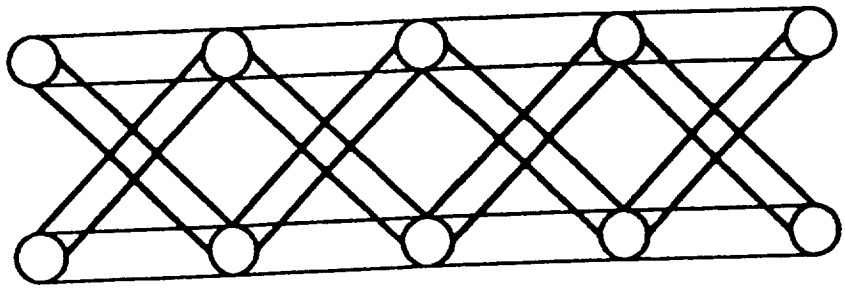


Figure 3.14 Small Trellis

small trelleses, in general, has different branch labels in different sections in contrast to a normal trellis, where each section of the trellis has the same branch label. There are four such small trelleses inherent in each one of the four trellis sub-diagrams of the modulation code.

Hence, for the entire decoder, we have to have total 16 2-state decoders one for each small trellis. The design of these decoders is the same as for 4-state code (discussed before) except some modifications (to be discussed).

It is obvious from the trellis sub-diagram that after eight and twelve sections of the trellis have been processed by the 2-state decoders, additional comparisons are to be made and corresponding path metrics and path history registers are to be updated. After 16 symbols have been processed (that is the completion of the processing of one code word), the survivor from each trellis sub-diagram is at state 1 of 2-state decoder for first small trellis. The final survivor is obtained by comparing these four survivors. This completes the decoding process.

The circuit design of the decoder to process the first trellis sub-diagram is discussed and the design of decoders for three other trellis sub-diagrams will exactly be the same.

3.6.2 ACS CIRCUIT

The ACS circuit for 2-state decoder for a single small trellis is shown in figure 3.15. M1 to M4 are smaller of the branch metrics of four pairs of parallel branches in the trellis.

The difference between this circuit and ACS circuit for the decoder for code1 is (1) additional 5 bits multiplexers MUX1 and MUX2 (figure 3.15) are required and (2) multiplexers connected to the inputs of path metric registers are 3 to 1 rather than 2 to 1. The need of additional multiplexers (MUX1 and MUX2) arises,

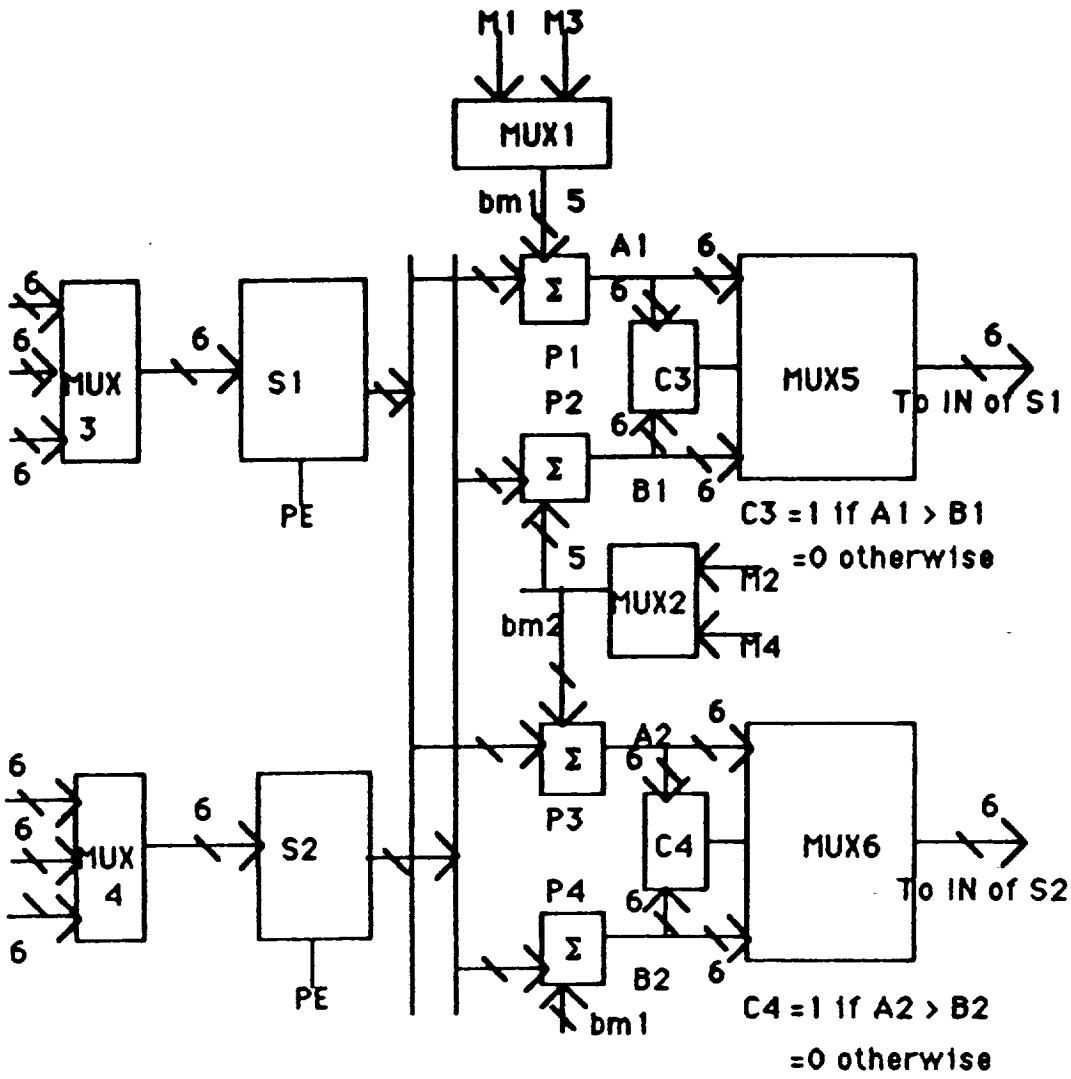


Figure 3.15 ACS Circuit

because the small trellis, in general, has different branch labels in different sections. The mod-16 counter output controls the selection of appropriate branch metric. Since additional 'comparison and select operation' needs to be performed after each time eight and twelve sections of the trellis have been processed, the multiplexers connected to the inputs of path metric registers have to be 3 to 1 multiplexers. The first input is for initialization of the path metrics (when count is 0), the second input is to store the result after additional comparison-select operation (when count is 7 and 12) and the third input is to update the path metric after normal ACS operation in each time unit.

Note that 16 such ACS circuits (one for each 2-state small decoder) are required for the entire decoder whereas in the case of 4-state decoder for code 1, we require only two such ACS circuits with less complexity.

In addition, four more 6 bits comparators and multiplexers per trellis sub-diagram are required for additional comparisons and updating of path metrics after each time eight and twelve sections of the trellis have been processed. Hence, note that the hardware requirement for the decoder for 16-state code is roughly 10 times larger than that of for the 4-state code.

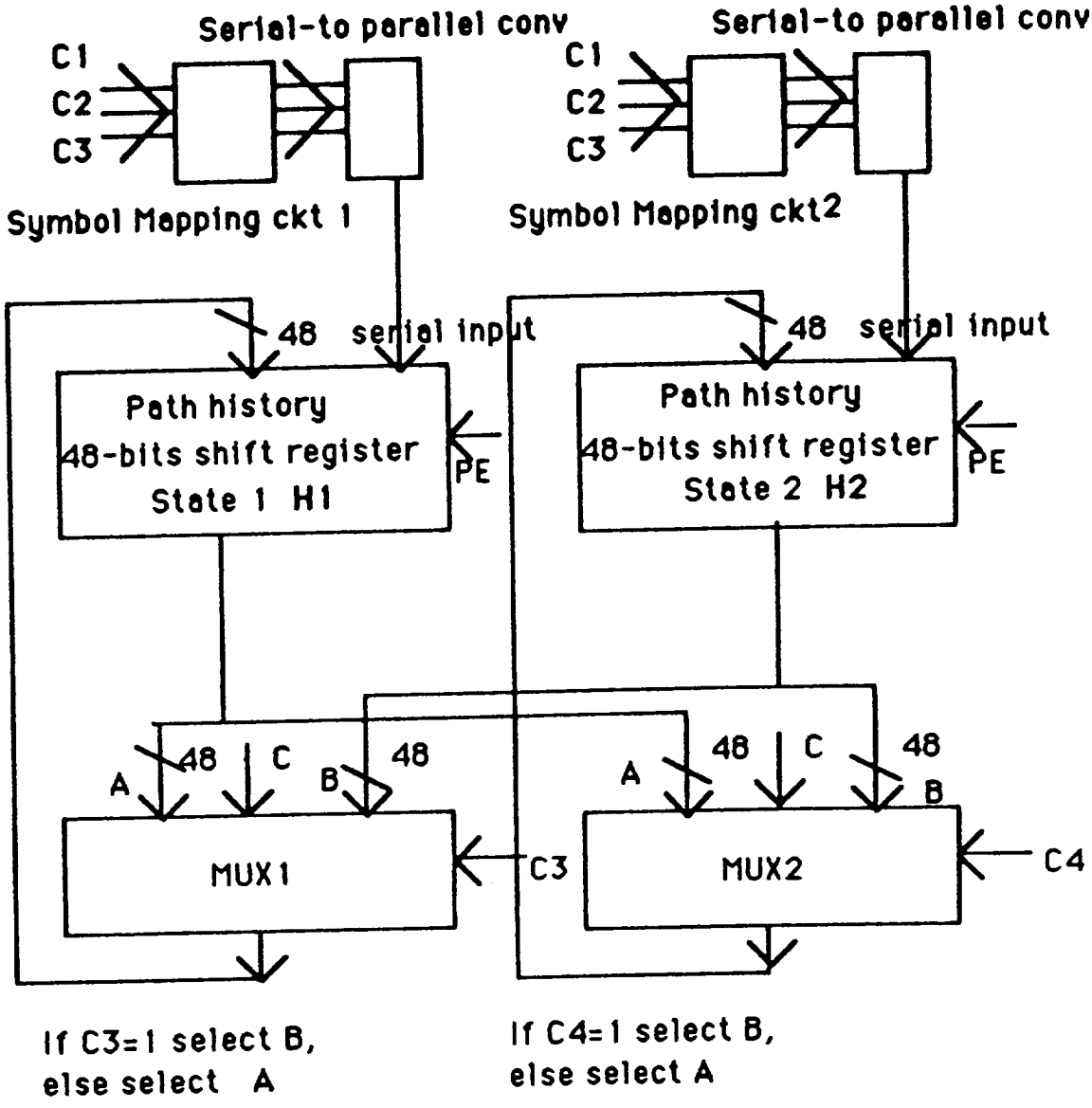
3.6.3 PATH HISTORY UPDATING CIRCUIT

The path history updating circuit using register exchange method for a single 2-state small decoder is shown in figure 3.16.

MUX1 and MUX2 are 3 to 1 48 bits multiplexers. H1 and H2 are 48 bits path history registers for storage of 16 symbols each 3 bits long. Additional path history updating is required after each time eight and twelve symbols have been processed. Hence, the survived sequence after additional comparison is connected to the third input of the multiplexer. Four additional 2 to 1 48 bits multiplexers per trellis sub-diagram are required to accomplish the additional path history updating after each time eight and twelve symbols have been processed.

3.6.4 OUTPUT CIRCUIT

After 16 symbols have been processed, the four survivor path metrics from the four valid states (one from each trellis sub-diagram) are compared, and the appropriate path history content is outputted. The diagram for this part of the decoder is shown in figure 3.17. The comparison of four path metrics is performed in two stages by three 6 bits comparators. The comparison results are used to control a 4 to 1 48 bits multiplexer, whose input lines are connected to the outputs of the path history register of state1



C1,C2,C3 and C4 are comparators (in ACS section) outputs.

Figure 3.16 Path History Updating

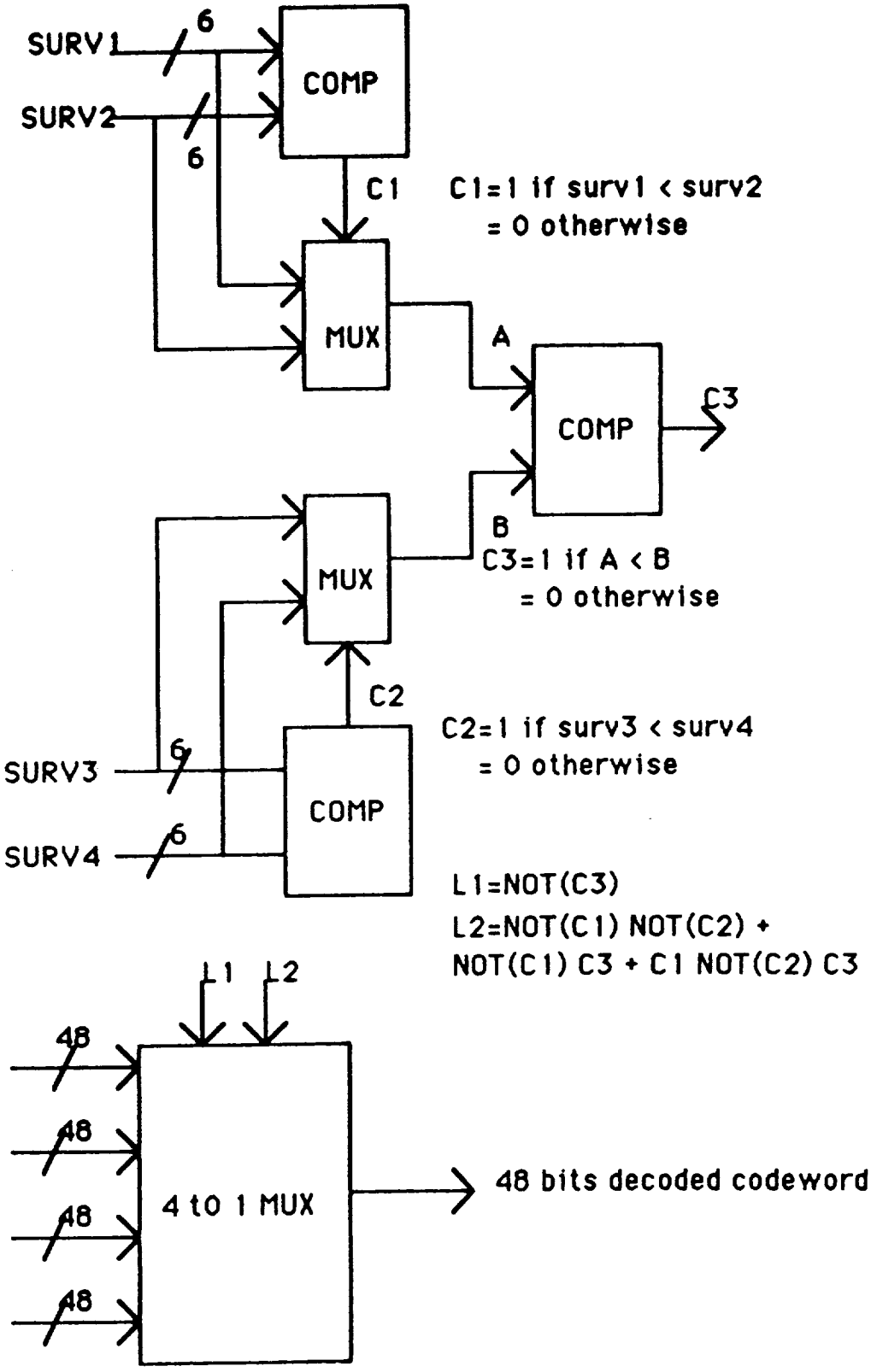


Figure 3.17 Output Circuit

of first small decoder for each trellis sub-diagram. The multiplexer outputs the final survived sequence.

3.6.5 CONTROL DESIGN

Step1 to step 10 remains the same as given in control design for code1.

Step 11: If count is 0111 or 1011:

(a) make additional comparisons of path metrics.

(b) Update the path history registers (by applying the control signal to parallel enable input of path history registers).

(c) Update the path metric registers (by applying the control signal to parallel enable input of path metric registers).

If count is 1111:

(a) output the decoded codeword (by applying the control signal to parallel enable input of output buffer register).

(b) Make the output ready line go HIGH.

Step 12: Increment the counter and wait for the next start pulse.

3.7 SIMULATION OF 16-STATE CODE AND SIMULATION RESULTS

Simulation results for 16-state code have been shown in figure 3.18. The actual coding gain (at BER 10^{-5}) for unquantized Viterbi decoder is 1.3 db (less than 4-state code) over uncoded QPSK. However, there is a reduction in bandwidth. 5 bits quantization results in 0.3 db loss. Once again, the non-uniform floating point to integer mapping scheme reduces the branch metric and path metric range without sacrificing coding gain.

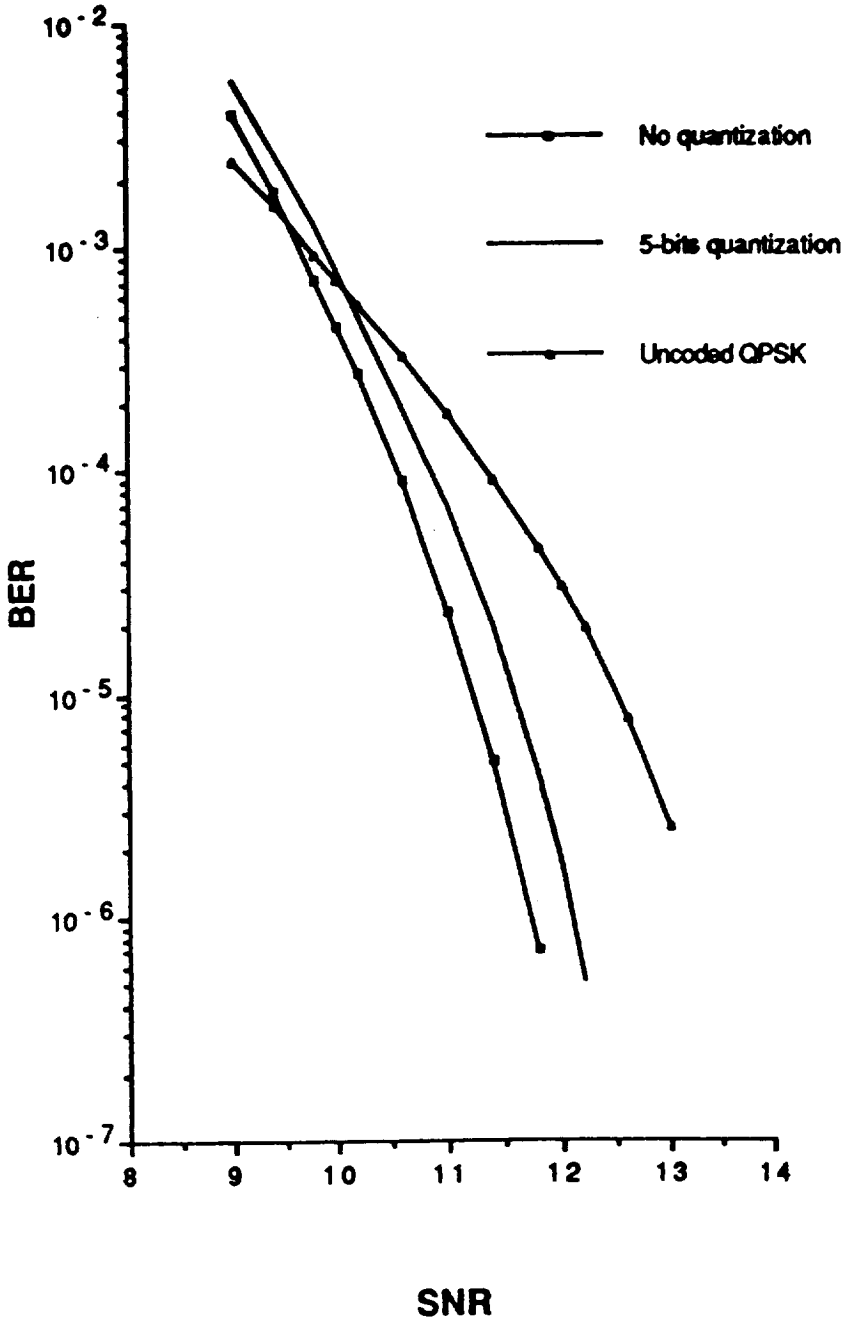


Fig. 3.18 Error performance of 16-state code

CHAPTER 4

MULTI-STAGE DECODING OF MULTI-LEVEL CODE AND PERFORMANCE EVALUATION

4.1 INTRODUCTION

Since the performance of a block modulation code is largely determined by its minimum squared Euclidean distance, the optimum decoding strategy for AWGN channel (assuming equal a priori probabilities) is soft decision maximum likelihood decoding based on the squared Euclidean distance. This requires computing the squared Euclidean distances of the received signal sequence from each of the $2^{\Sigma k_i}$ codewords in signal space and selecting the code word which has minimum squared Euclidean distance. If the value of Σk_i is large, this method becomes prohibitively complex.

If each of the component codes has trellis structure, the modulation code also has trellis structure. The trellis diagram of the modulation code is the direct product of the trellis diagrams of its component codes. Because of the multiplicative nature, the trellis diagram of the modulation code can be quite complicated with large number of states and large number of parallel branches even if the component codes have relatively simple trellis structure. Hence, even if a modulation code has trellis structure, it might not be practical to implement the Viterbi decoder for it.

This is true, in particular, for the modulation code with large minimum squared Euclidean distance and long length.

Hence, it is obvious that for practical purposes, it is necessary to devise a sub-optimum but practical decoding algorithm. The multi-stage decoding is very attractive solution to this problem.

The basic idea behind the multi-stage decoding of multi-level modulation code is as follows. The multi-stage decoding of multi-level block modulation code is based on their multi-level structure. Since the multi-level block modulation code consists of m component codes, we decode each of these component codes separately in m stages. The most powerful component code is decoded first and the least powerful component code is decoded last. The decoded information (or code word) in each stage is assumed to be correct and is stored in a buffer to be used by the later stages. The process continues stage by stage for m stages until all the information bits have been recovered.

At each stage of decoding, we can perform soft-decision maximum likelihood decoding or other sub-optimum decoding (e.g. algebraic decoding) depending on the component code being decoded. In other words, at each decoding level, we may take advantage of the structure of the component code. If each component code has trellis structure with reasonable number of states and parallel branches, we can decode each component code using a soft-decision

Viterbi decoder. If component codes have complex trelleses and soft-decision Viterbi decoding is not practical, then we may use hard-decision algebraic decoding.

4.2 MULTI-STAGE DECODING ALGORITHM FOR MULTI-LEVEL BLOCK MODULATION CODE

We illustrate the multi-stage decoding algorithm of basic multi-level modulation code by considering an example of 3-levels 8-PSK modulation code. It can be easily generalized for higher levels and other modulation codes. The basic idea remains the same.

Since basic 3-levels 8-PSK modulation code consists of 3 component codes, there are three stages of decoding. The schematic diagram for multi-stage decoding of 3-level 8-PSK block modulation code is shown in figure 4.1 where r is the received vector, and k_1 to k_3 are decoded message bits in the 1st, 2nd and 3rd stages respectively. Note that the decoded codeword in a stage is passed to all other later stages. The decoding process begins with the first level component and ends at the third level (last level) component code. One important observation that can be made from this diagram is that the multi-stage decoding creates pipelined parallelism in the decoding process. This is very desirable structure for high speed system.

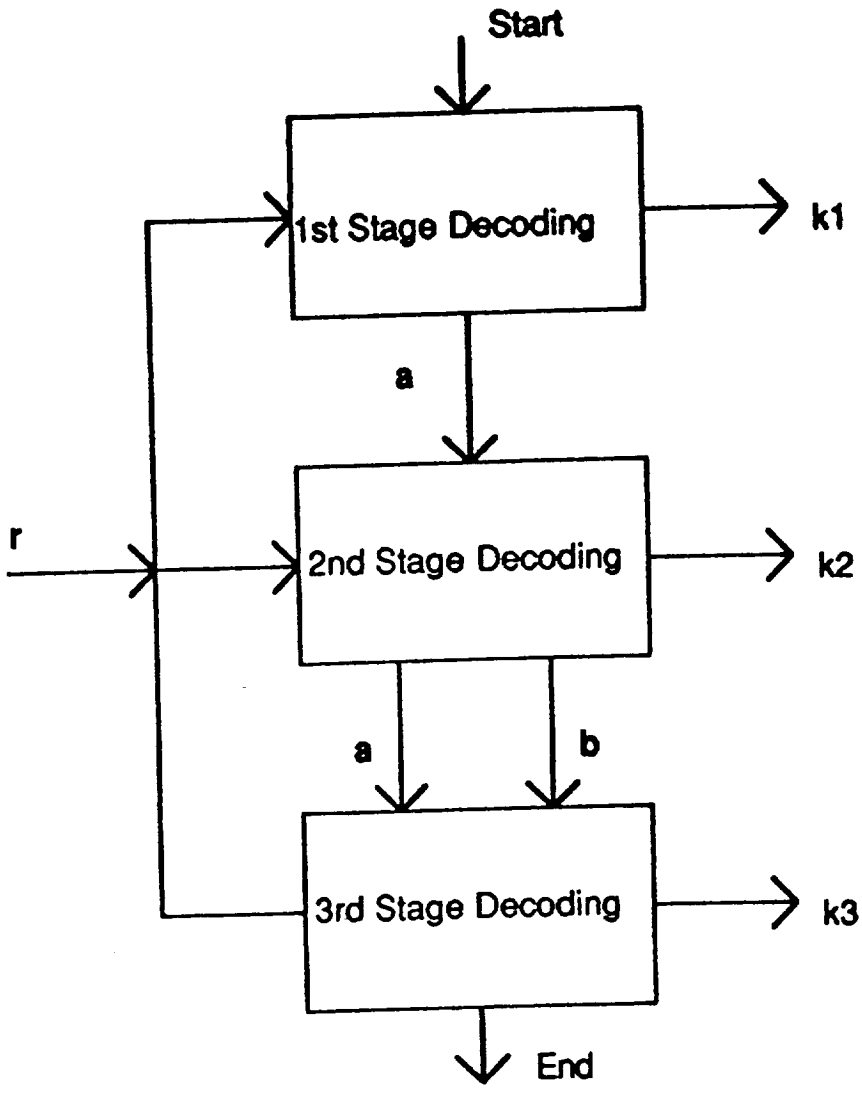


Fig. 4.1 multi-stage decoding of 3-level code

4.2.1 SOFT-DECISION MULTI-STAGE DECODING

Decoding in each stage is done by a soft-decision maximum likelihood decoder based on the squared Euclidean distance. Hence, decoding for each component code is optimum. Assume that the

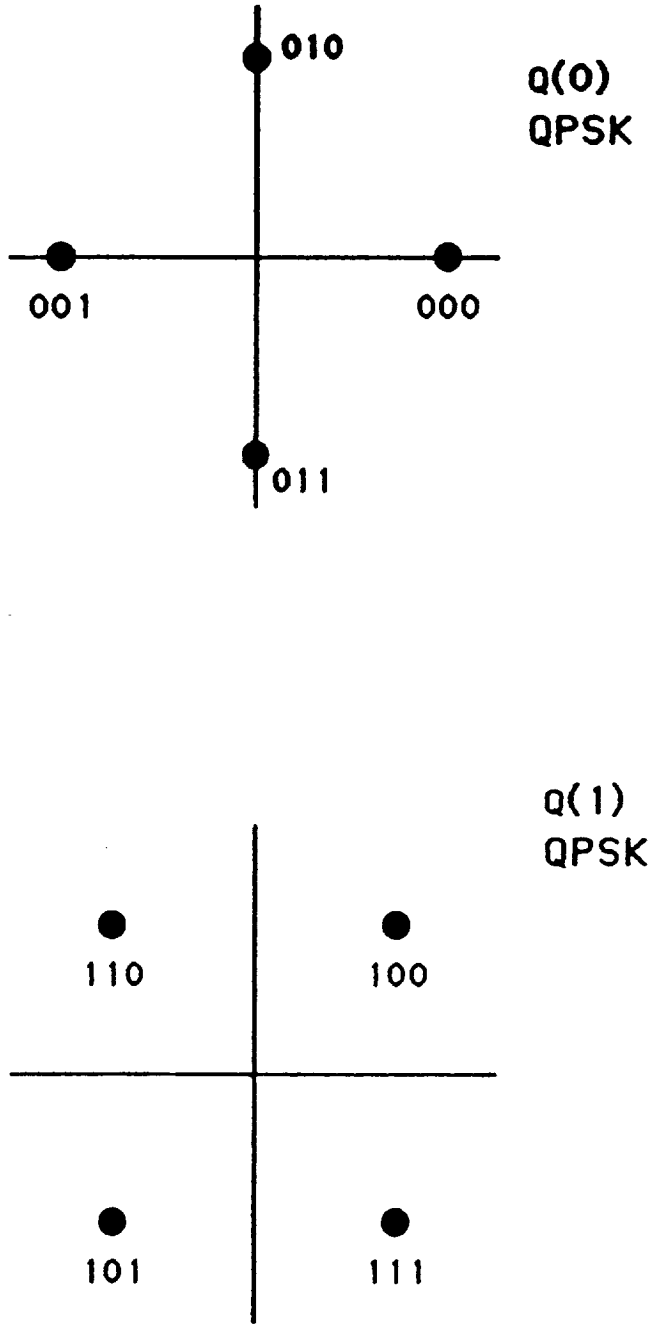


Fig. 4.2 QPSK signal constellations

channel is an AWGN channel. Let $\mathbf{r} = (r_1, r_2, \ldots\ldots\ldots, r_n)$ be the received sequence at the output of the 8-PSK demodulator, where $r_j = (x_j, y_j)$ is a point in the two dimensional Euclidean plane.

First decoding stage

Let $\mathbf{a} = (a_1, a_2, \ldots\ldots\ldots\ldots, a_n)$ be a binary codeword in the component code C_1.

Compute the squared Euclidean distances between r_j and the signal points in $Q(a_j)$. Recall that $Q(a_j)$ represents a QPSK signal set (shown in figure 4.2) which is obtained by partition of 8-PSK signal set with prefix a_j. Since there are four points in $Q(a_j)$, there are four such distances. Compare these four distances and find the minimum one. Let $d[r_j, Q(a_j)]$ be the minimum squared Euclidean distance between r_j and the points in $Q(a_j)$. For every codeword $\mathbf{a}$ in C_1, compute the distance,

$$d(\mathbf{r}, \mathbf{a}) = \sum_{l=1}^{n} d[r_l, Q(a_l)]$$

Note that the total number of such computations required is 2^{k1}. If C_1 has a simple trellis structure, this computation will be greatly reduced. Decode $\mathbf{r}$ into $\mathbf{a}^*$ for which $d(\mathbf{r}, \mathbf{a}^*)$ is the minimum. The decoded codeword $\mathbf{a}^*$ is stored in a buffer to be used in the second and third stages of decoding.

Second decoding stage

The decoded codeword $\mathbf{a}^*$ of the first stage is passed to the second stage. Let $\mathbf{b} = (b_1, b_2, \ldots\ldots\ldots\ldots, b_n)$ be a binary codeword in component code C_2.

Compute the squared Euclidean distances between r_i and the points in $Q(a_i^* b_i)$. Recall that $Q(a_i^* b_i)$ represents a BPSK signal set which is obtained by partition of 8-PSK signal set with prefix $a_i^* b_i$. Since there are two points in $Q(a_i^* b_i)$, there are two such distances. Compare these two distances and find the smaller one.

Let $d[r_i, Q(a_i^* b_i)]$ be the minimum squared Euclidean distance between r_i and the points in $Q(a_i^* b_i)$. For every codeword $\mathbf{b}$ in C_2, compute

$$d(\mathbf{r}, \mathbf{a}^*\mathbf{b}) = \sum_{i=1}^{n} d[r_i, Q(a_i^* b_i)]$$

Decode $\mathbf{r}$ into codeword $\mathbf{b}^*$ for which $d(\mathbf{r}, \mathbf{a}^*\mathbf{b}^*)$ is the minimum. The dedoded codeword is stored in a buffer to be used in the third stage.

Third decoding stage

The decoded codewords at the first and second stages, $\mathbf{a}^*$ and $\mathbf{b}^*$ are available to the third stage. Let $d[r_i, Q(a_i^* b_i^* c_i)]$ denotes the squared Euclidean distance between received symbol r_i and the

point in 8-PSK signal set with labeling $a_i{}^*b_i{}^*c_i$. For every codeword $c=(c_1,c_2,\ldots\ldots\ldots\ldots,c_n)$ in C_3, compute

$$d(r, a^*b^*c) = \sum_{i=1}^{n} d[r_i, Q(a_i{}^*b_i{}^*c_i)]$$

$$= \sum_{i=1}^{n} d[r_i, \mu(a_i{}^*b_i{}^*c_i)]$$

Decode r into codeword c^* for which $d(r, a^*b^*c^*)$ is the minimum. This completes the decoding. $\mu(a^* * b^* * c^*)$ forms the decoded 8-PSK signal sequence.

The soft-decision multi-stage decoding algorithm can be summarized as follows. The squared Euclidean distances (metrics) of received symbol from all 8-PSK signal points are computed and are made available to the soft-decision maximum likelihood decoders for each component codes. The decoding at each stage is done in three steps: (1) selection of proper signal set (with proper labeling) based on received symbol and the decoded informations in the previous stages, (2) finding the signal with minimum distance in the selected set, and (3) computation of appropriate metric for decision. If each component code has a trellis structure, then the Viterbi decoding algorithm can be applied to decode each component code.

The soft decision multi-stage decoding is not optimum even though the decoding of each component code is optimum. It is sub-optimum. The difference in performance between the optimum

decoding and the sub-optimum soft-decision multi-stage decoding has been found very small, a fraction of db in coding gain, for many multi-level modulation codes.

4.2.2 HARD-DECISION MULTI-STAGE DECODING

In case component codes do not have simple trellis structure, it is hard to perform soft-decision maximum likelihood decoding for each component code based on Euclidean distance. Also, in some cases soft-decision decoding can be too expensive. In general, the component codes chosen to construct the multi-level block modulation codes are well known binary block codes with well established decoding algorithm based on hard decision of the demodulator. Hence, the main idea behind the hard decision multi-stage decoding of multi-level block modulation code is to further simplify the decoding of each component code by employing known algebraic decoding algorithms for these codes. Also, the hard-decision multi-stage decoding provides more flexibility in the construction of multi-level modulation code in the sense that we can employ any class of binary block codes as component codes.

The decoding at each stage is performed in two steps. The first step is hard demodulation or bit decision of each symbol based on decoded codewords in previous stages. In the second step,

the hard demodulated binary sequence is given to a binary block code decoder.

Since at first stage of decoding, each 8-PSK symbol is hard demodulated independently without the help of any other information, the minimum squared Euclidean distance between the signal sequences, which is the major performance criterion of a modulation code, does not get a chance to play any role, a significant coding gain can be expected to be lost in comparison with the soft-decision multi-stage decoding. Note that while making bit-decision at first stage, no other information is available to the detector. Rather, we have a worse situation. Signal points are crowded and bit-decision is to be made by independent hard demodulation of each symbol without the help of any other information. On the other hand, since the decoded information in the first stage is used for decoding in later stages, we must make sure that decoding in first stage is correct. Hence, the decoding in the first stage is entirely based on Hamming distance of first level component code and this component code should be chosen to have as high Hamming distance as possible.

The decoding algorithm is as follows. Let $\mathbf{r} = (r_1, r_2, \ldots\ldots, r_n)$ be the received sequence at the output of the demodulator, where r_i is a point in two dimensional Euclidean plane denoted by R^2.

First decoding stage

Divide R^2 plane into two decision regions, R_0^2 and R_1^2 where R_0^2 contains the signal points whose labels have 0 as the prefix and R_1^2 contains the signal points whose labels have 1 as the prefix. For 3-level 8-PSK modulation code, the division of R^2 is shown in figure 4.3.

Hard decision: For $1 \leq i \leq n$, make decision on the first label bit a_i based on the received symbol r_i. If r_i is a point in R_0^2, set the output of the first stage detector $a_i=0$. If r_i is a point in R_1^2, set the output of the first stage detector $a_i=1$. Essentially, each r_i is demodulated independently into one of the 8 8-PSK symbols and a_i is set equal to the first label of the demodulated symbol.

Decoding C_1: After the first label bit decisions have been made for n received symbols, the binary vector $\mathbf{a} = (a_1, a_2, \ldots\ldots\ldots\ldots, a_n)$ at the output of the detector is passed to the decoder for component code C_1. The decoder C_1 operates on vector **a** and puts out the decoded codeword $\mathbf{a}^*$. The decoding may be maximum likelihood decoding or algebraic decoding. The schematic diagram of first-stage decoding is shown in figure 4.4.

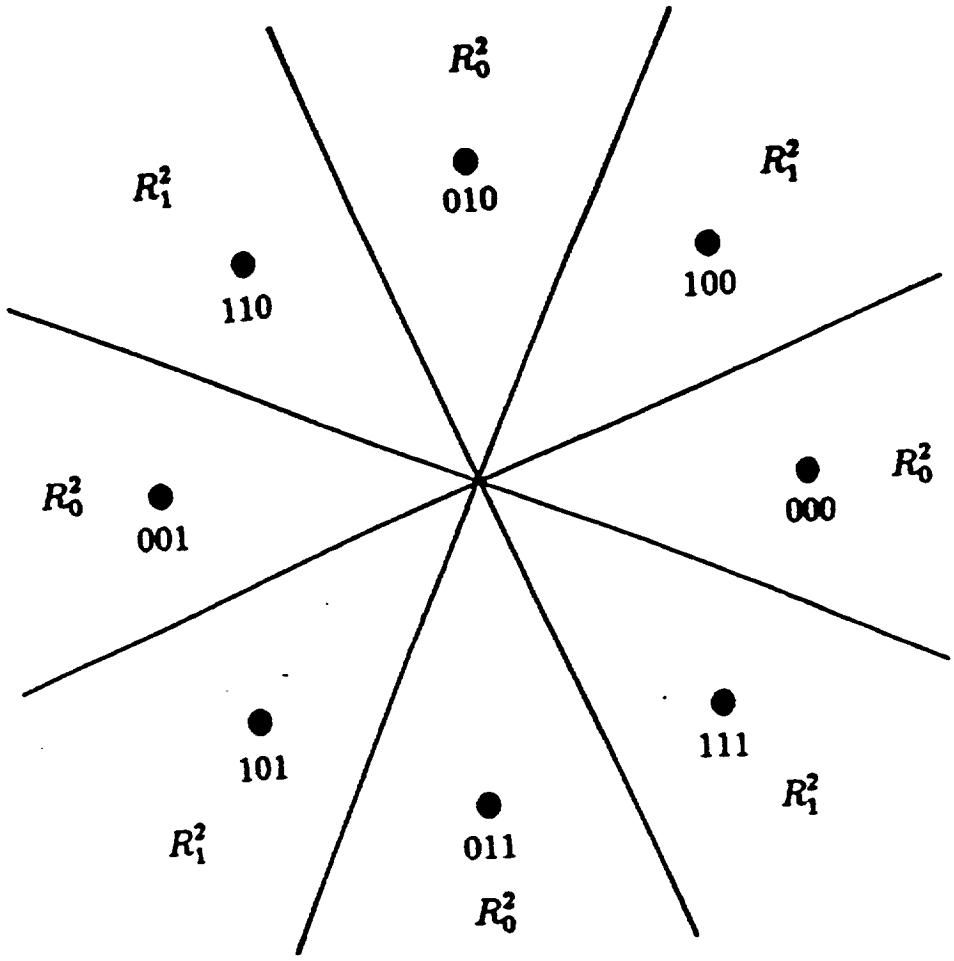


Fig. 4.3 Decision regions for the first label bit

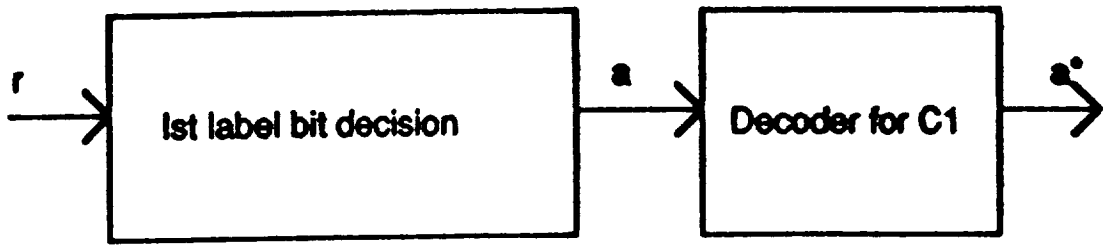


Figure 4.4 Schematic diagram of 1st stage hard decision decoding

Let

$$a^* = (a_1^*, a_2^*, \ldots\ldots\ldots\ldots, a_n^*)$$

be the decoded codeword in the first decoding stage.

Second decoding stage

The decoded codeword, a^*, from the first decoding stage is passed to the second stage. For $a_i^* = 0$, divide the R^2-plane into two decision regions, R_{00}^2 and R_{01}^2, where R_{00}^2 contains those signal points whose labels have 00 as the prefix and R_{01}^2 contains those signal points whose labels have 01 as the prefix. For $a_i^* = 1$, divide the R^2-plane into two decision regions, R_{10}^2 and R_{11}^2, where R_{10}^2 contains those signal points whose labels have 10 as the prefix and R_{11}^2 contains those signal points whose labels have 11 as the prefix.

For 3-level 8-PSK modulation code, the division of R^2 plane is shown in figure 4.5.

Hard decision: Depending on the value of a_i^*, the proper QPSK set either {0,2,4,6} or {1,3,5,7} is selected. Decision on the second

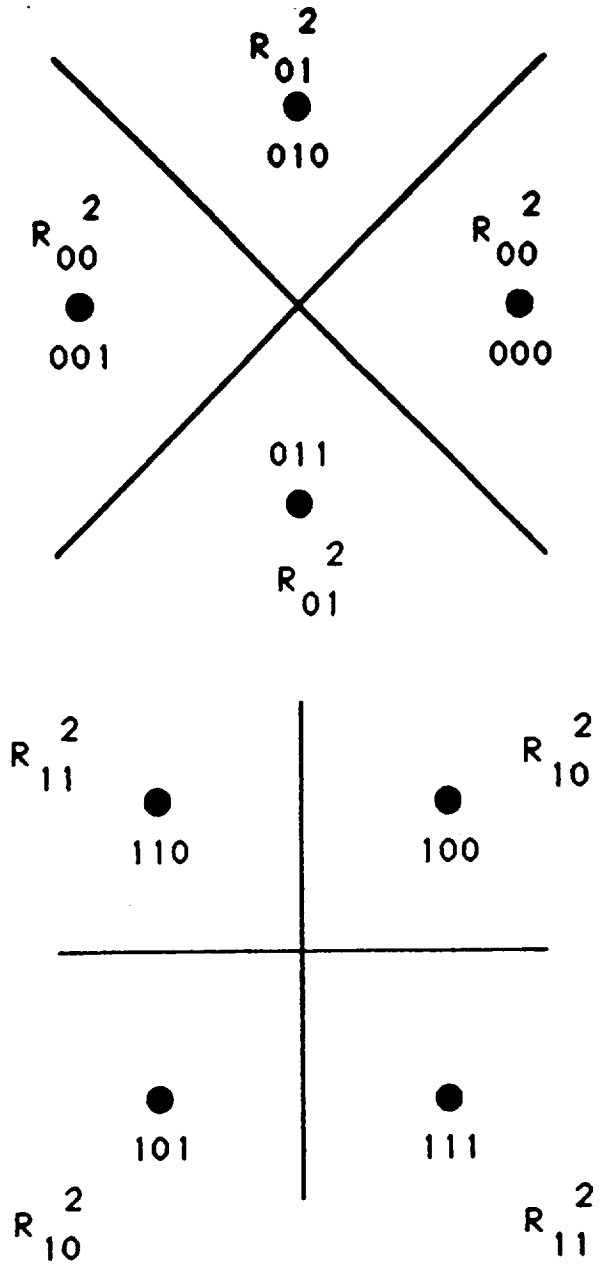


Fig. 4.5 Decision regions for the 2nd label bit

label bit b_i is made depending upon which one of the two possible decision regions for the selected QPSK signal set contains the received sequence r_i. For $1 \leq i \leq n$,

If $a^*_i=0$ and r_i is in the region $R_{00}{}^2$, then set the output of the second-stage detector,

$$b_i=0.$$

If $a^*_i=0$ and r_i is in the region $R_{01}{}^2$, then set the output of the second-stage detector,

$$b_i=1.$$

If $a^*_i=1$ and r_i is in the region $R_{10}{}^2$, then set the output of the second-stage detector,

$$b_i=0.$$

If $a^*_i=1$ and r_i is in the region $R_{11}{}^2$, then set the output of the second-stage detector,

$$b_i=1.$$

Decoding C_2: The sequence $\mathbf{b} = (b_1, b_2, \ldots\ldots\ldots\ldots, b_n)$ at the output of the detector is passed to the decoder for C_2, which operates on $\mathbf{b}$ and puts out the decoded codeword $\mathbf{b}^*$. The schematic diagram of the second stage decoder is shown in figure 4.6.

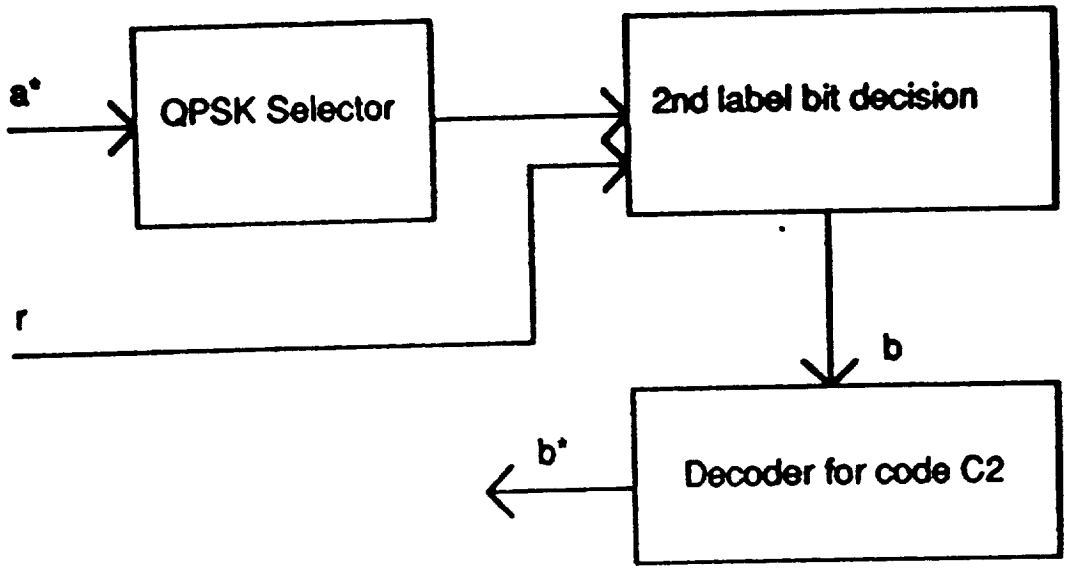


Fig. 4.6 Schematic diagram of the 2nd-stage hard-decision decoding

Let

$$b^* = (b_1^*, b_2^*, \ldots\ldots\ldots\ldots, b_n^*)$$

be the decoded codeword in the second stage of the decoding .

Third decoding stage

The decoded codewords, a^* and b^*, at the first and second decoding stages is passed to the third stage. Based on $a_i^* b_i^*$, the R^2 plane is divided into two decision regions, $R_{a_i^* b_i^* 0}^2$ and $R_{a_i^* b_i^* 1}^2$ where $R_{a_i^* b_i^* 0}^2$ contains the 8-PSK signal points with $a_i^* b_i^* 0$ as label and $R_{a_i^* b_i^* 1}^2$ contains the 8-PSK signal points with $a_i^* b_i^* 1$ as label. For 3-level 8-PSK modulation code, the division of R^2 plane is shown in figure 4.7.

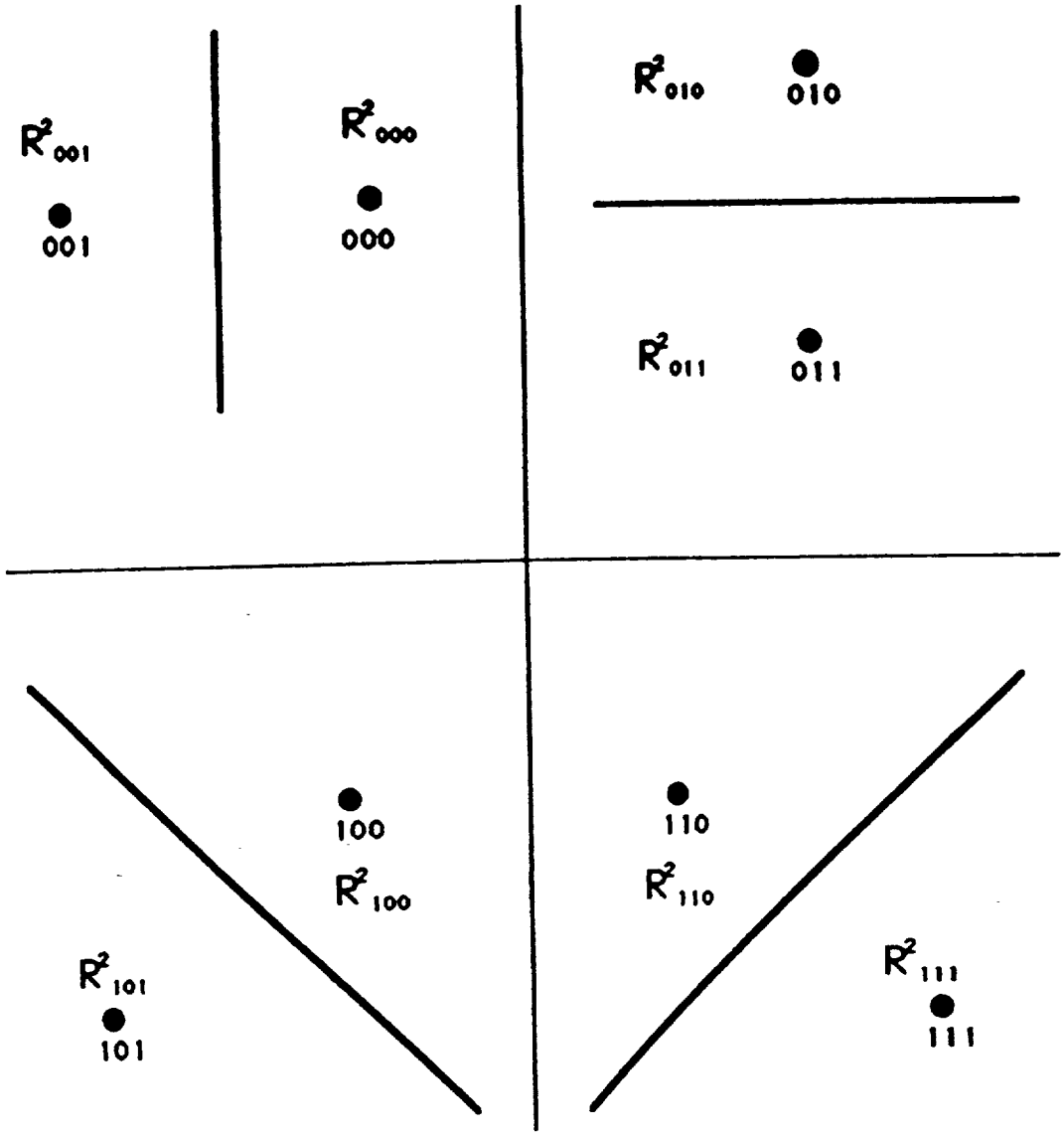


Figure 4.7 Decision regions for the third label bit

Hard decision: The dibit $a_i^* b_i^*$ is used to select the proper BPSK signal set. There are four such set. The decision on the third label bit c_i is made depending upon which one of the two possible decision regions of the selected BPSK set contains the received sequence r_i. The decision is made as follows. For given $a_i^* b_i^*$, if r_i is a point in $R_{a_i^* b_i^* 0}{}^2$, then set the output of the third stage detector,

$$c_i = 0$$

otherwise, set

$$c_i = 1.$$

Decoding C_3: The binary vector $\mathbf{c} = (c_1, c_2, \ldots\ldots\ldots\ldots, c_n)$ at the output of the detector is passed to the binary code decoder for code C_3 which puts out the decoded word $\mathbf{c}^*$. The schematic diagram of the third-stage decoder is shown in figure 4.8.

Let

$$\mathbf{c}^* = (c_1^*, c_2^*, \ldots\ldots\ldots\ldots, c_n^*)$$

be the decoded codeword in the third stage of the decoding. This completes the decoding process. $\mathbf{a}^*$, $\mathbf{b}^*$ and $\mathbf{c}^*$ forms the decoded set.

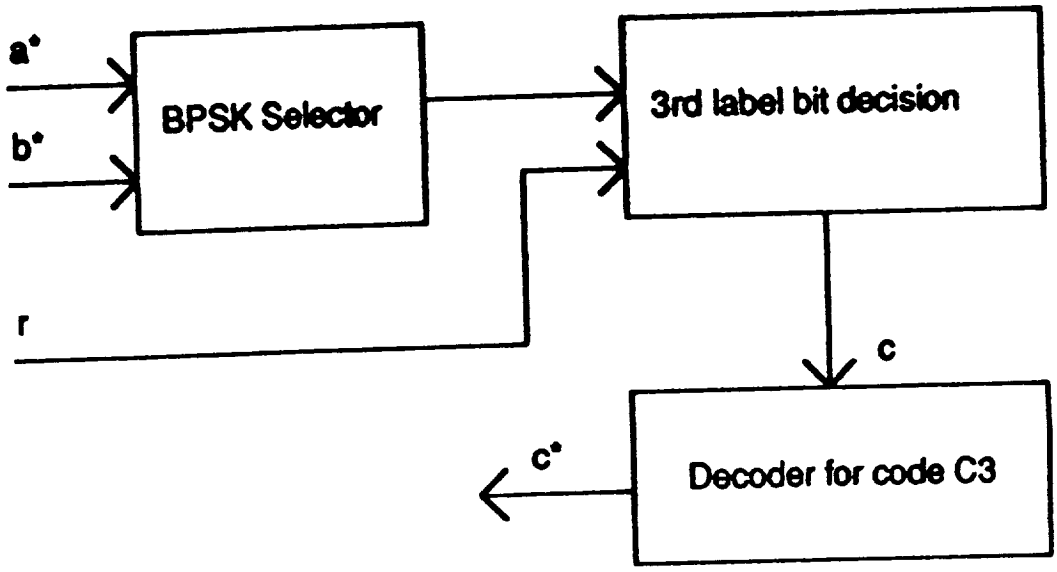


Fig. 4.8 schematic diagram of the 3rd-stage hard-decision decoder

4.3 PERFORMANCE EVALUATION BY COMPUTER SIMULATION

Multi-stage decoding does reduce the decoding complexity drastically, but it is sub-optimum. The probability of correct decoding of the code is given by

$$P_C = P_{C1} \cdot P_{C2|C1} \cdot P_{C3|C2,C1}$$

where P_{C1} is the probability of correctly decoding the first code, $P_{C2|C1}$ is the probability of correctly decoding the second code given that the first code was correctly decoded, and $P_{C3|C2,C1}$ is the probability of correctly decoding the third code given that the first two codes were correctly decoded.

One source of potential loss is easily seen. If the decoding at a certain stage is not correct, it may lead to decoding error in later stages. In other words, multi-stage decoding may suffer from error propagation.

The performance of the multi-stage decoding has been evaluated by computer simulation for codes with different minimum squared Euclidean distances and different code lengths for both soft and hard decision decoding. The following codes were chosen for simulation.

Code1

This is a 3-level 8-PSK modulation code of length =16, minimum squared Euclidean distance D[C] = 4. The code is given by

$$C = RM_{4,1} * P_{16} * V_{16}$$

The first component code C1 = $RM_{4,1}$ has a 4-section 8-state trellis, the second component code C2 = P_{16} has a 16-sections 2-state trellis and the third component code C3 = V_{16} has a 16-sections 1-state trellis. The modulation code C has a 16-state trellis with complex inter-connection structure (discussed in chapter 3).

The error-performance of the code with single-stage soft-decision maximum likelihood decoding (optimum decoding) and multi-stage soft-decision maximum likelihood decoding are shown

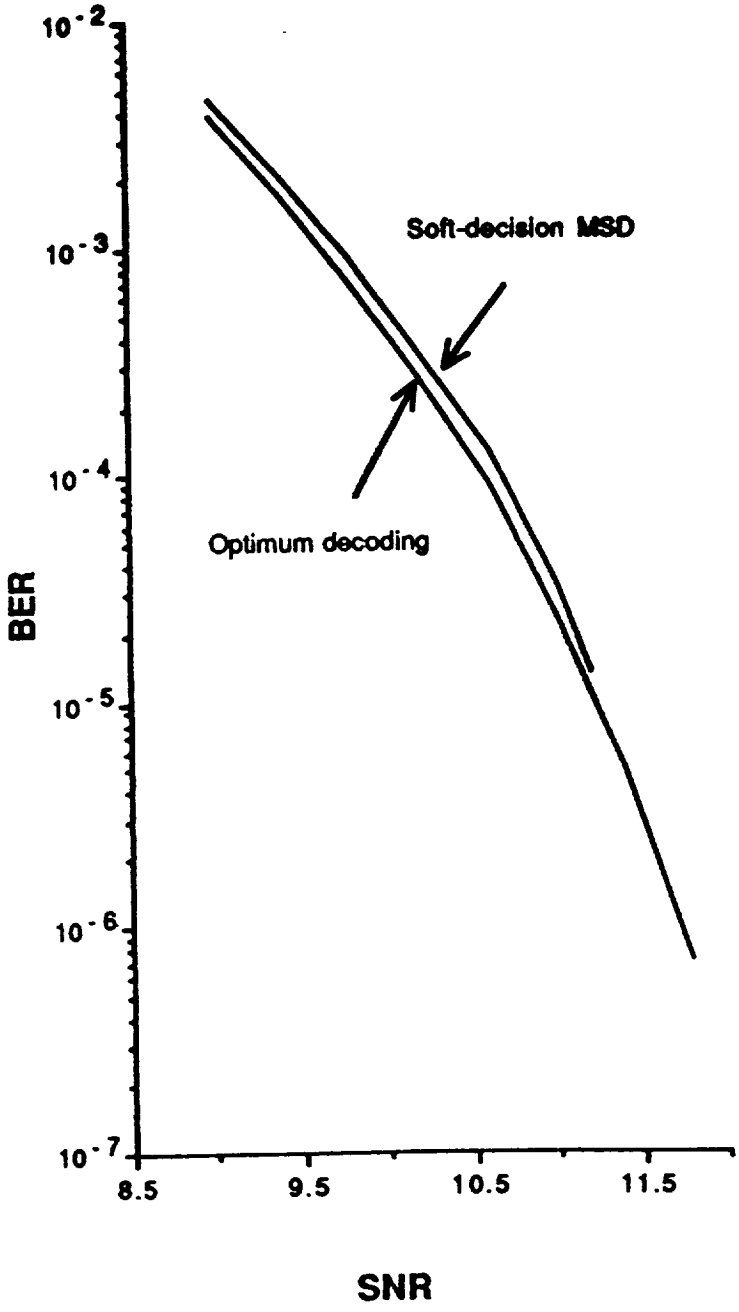


Fig. 4.9 Error performance of Code1 for optimum decoding and soft-decision MSD

in figure 4.9. Note that the difference in performance between optimum decoding and suboptimum multi-stage decoding is 0.15 db at BER 10^{-5}. Furthermore, this difference becomes even smaller at higher SNR. Hence, soft-decision multi-stage decoding for this code provides great complexity advantage without any significant loss in coding gain. For multi-stage decoding of this code, three *simple* Viterbi decoders are required, one with 8-states (or four 2-states decoders), second one with 2-state and third one with just 1-state. Recall that optimum Viterbi decoder for this modulation code requires sixteen 2-state decoders plus some additional circuits (designed discussed in chapter 3). On using multi-stage decoding, we need total five 2-state and single 1-state decoders.

Code2:

The three binary component codes C_1, C_2 and C_3 are as follows: (1) $C_1 = P_{16}'$ is (16,1) repeatition code which consists of all-one and all-zero 16-tuples; (2) $C_2 = RM_{4,2}$ is second order Reed-Muller code of length 16 ((16,11) code); (3) $C_3 = P_{16}$ is the binary (16,15) code with all the even weight 16 tuples. The basic 3-level 8-PSK modulation code C of length 16 constructed by these component codes is given by

$$C = P_{16}' * RM_{4,2} * P_{16}$$

The code has minimum squared Euclidean distance D[C]=8, dimension K=27 and effective rate R[C]=27/32. The first

component code $C_1 = P_{16}'$ has a 16-section 2-state trellis, the second component code $C_2 = RM_{4,2}$ has a 4-sections 8-state trellis and the third component code $C_3 = V_{16}$ has a 16-sections 2-state trellis. Hence, the modulation code C has a 32-states trellis.

The error-performance of the code with single-stage soft-decision maximum likelihood decoding (optimum decoding) and multi-stage soft-decision maximum likelihood decoding are shown in figure 4.10. Note that the difference in performance between optimum decoding and suboptimum multi-stage decoding is 0.30 db at BER 10^{-5}. Once again this difference becomes much smaller at higher SNR.

This code was also simulated for hard-decision multi-stage decoding with (1) maximum likelihood decoding for all the three component codes, (2) maximum likelihood decoding for C_1 and C_3 and majority logic decoding for C_2 ($RM_{4,2}$). The error performance is shown in figure 4.11. The hard decision multi-stage decoding (MSD) with all the code decoded with maximum likelihood decoder results in loss of 1.6 db over the optimum decoding at BER 10^{-5} (still provides 2.5 db coding gain over the uncoded QPSK) whereas the hard decision MSD with the second component code ($RM_{4,2}$ code) decoded with majority logic decoding results in loss of 2.4 db over the optimum decoding at BER 10^{-5} (still provides 1.7 db coding gain over the uncoded QPSK). The error-performance curve in figure 4.12 shows that this code, even with hard-decision MSD and

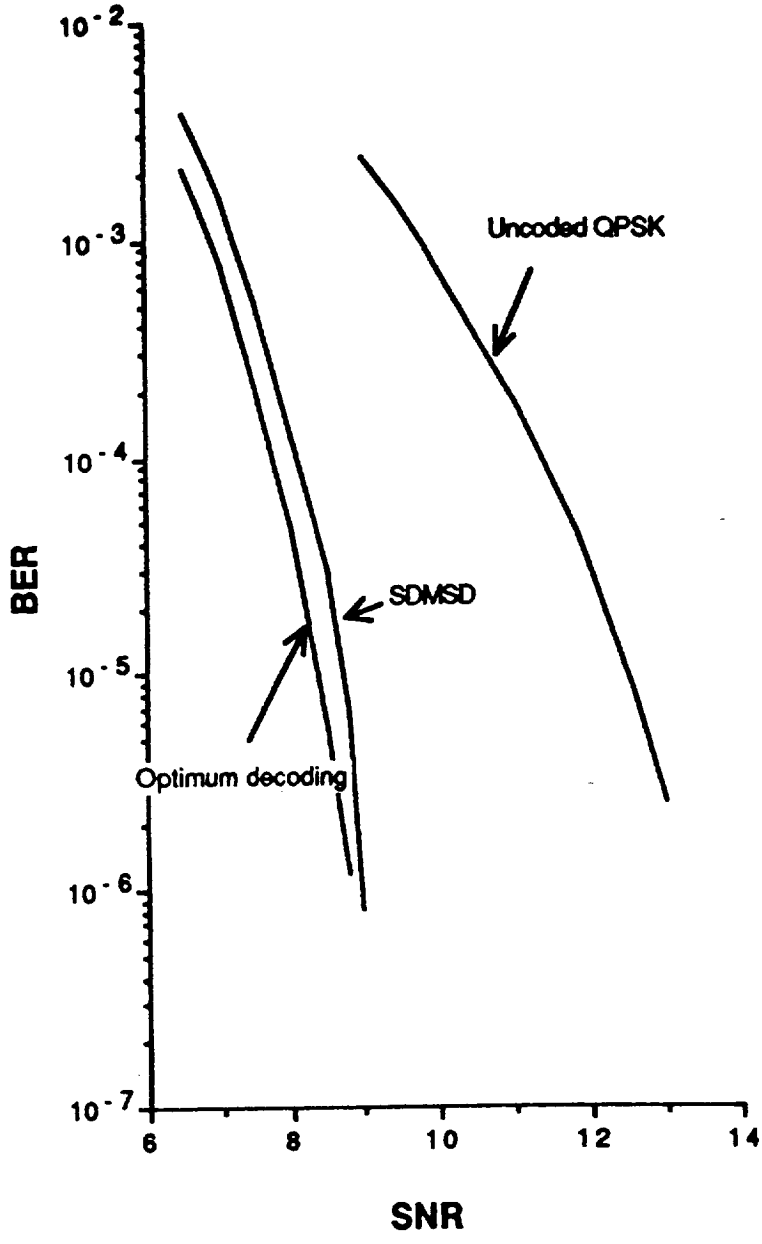


Fig.4.10 Error performance of Code2 for optimum decoding and soft-decision MSD

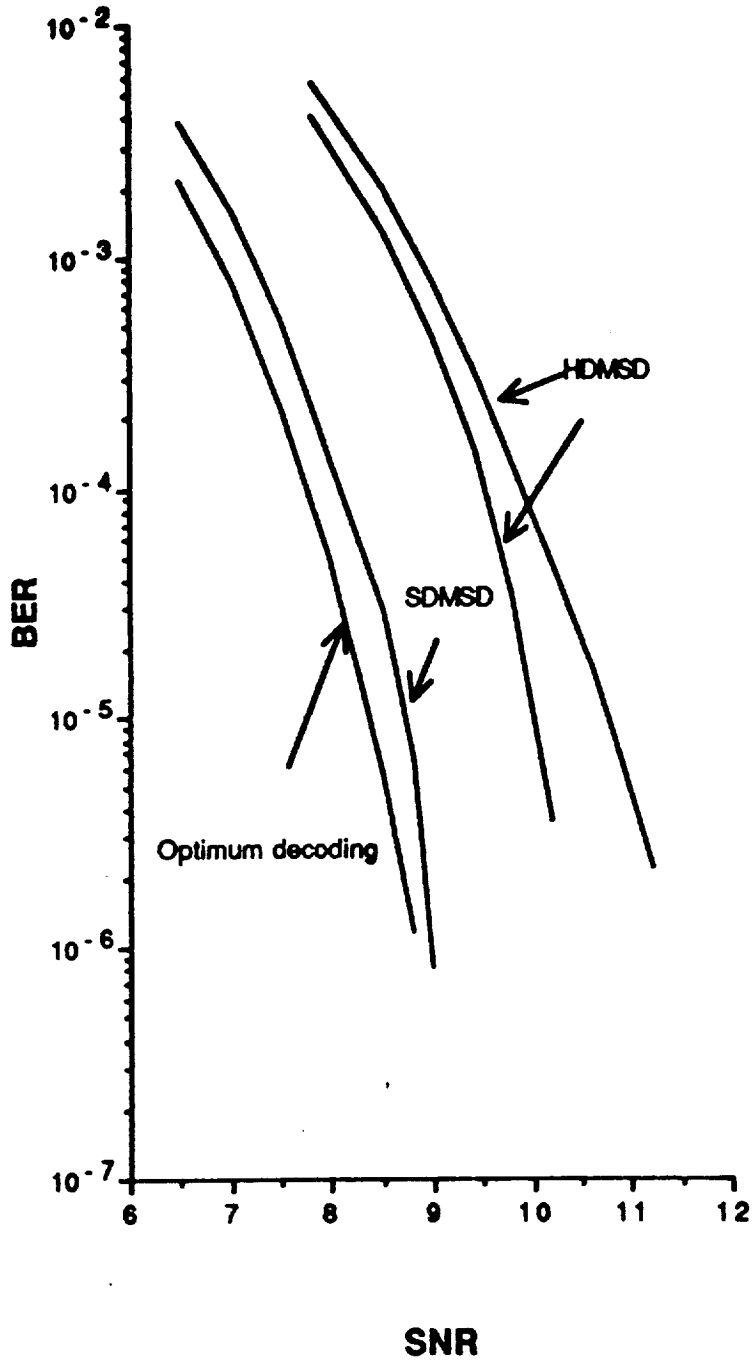


Fig. 4.11 Error performance of Code2 for soft and hard decision MSD

C-2

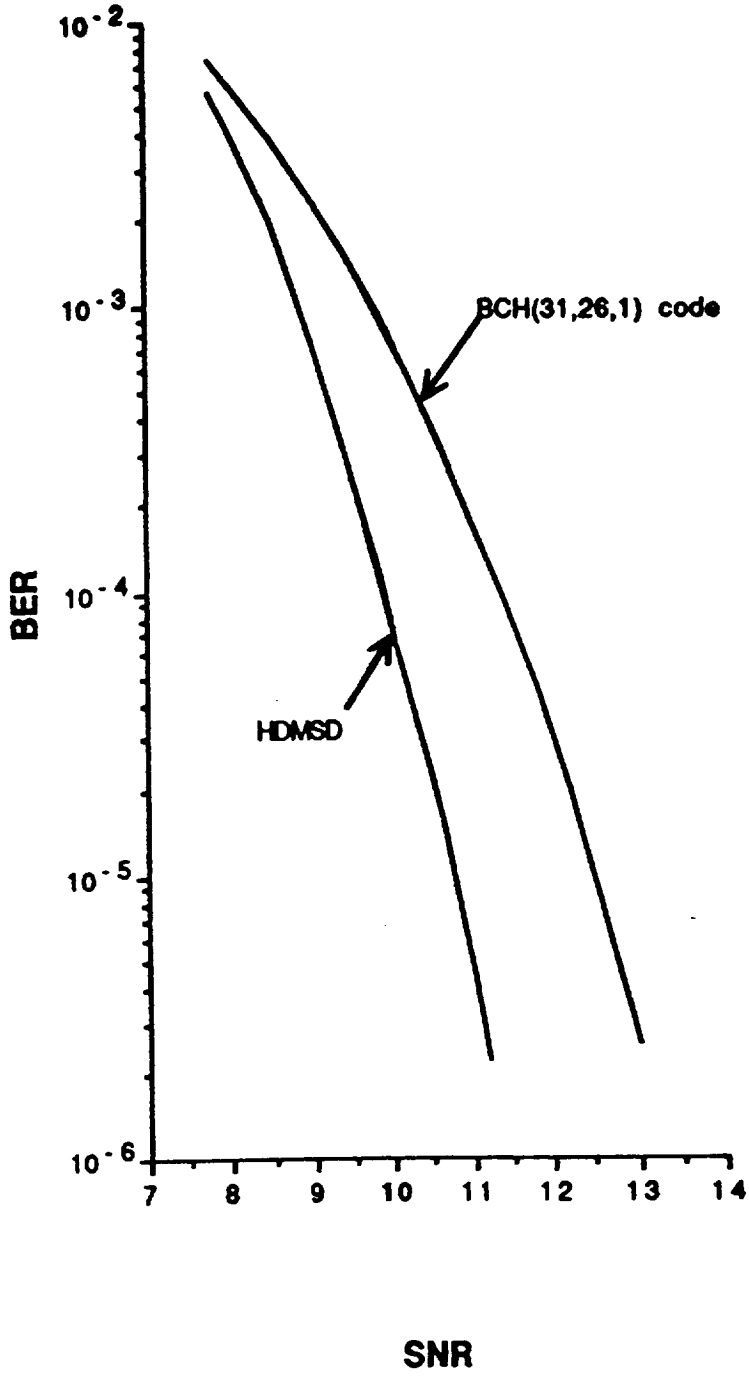


Fig4.12 Error performance of Code2 with hard-decision MSD and same rate BCH code

majority-logic decoding for the second code, has better error performance than (31,26,1) BCH code with the same rate.

code3:

This code was constructed using following components codes: (1) $C_1 = RM_{6,1}$ (2) $C_2 = RM_{6,3}$ (3) $C_3 = RM_{6,4}$, where $RM_{m,r}$ denotes the r-th order Reed-Muller code of length 2^m and minimum Hamming distance equal to 2^{m-r}. The basic multi-level modulation code C constructed by these component codes is given by

$$C = RM_{6,1} * RM_{6,3} * RM_{6,4}$$

The code has minimum squared Euclidean distance D[C]=16, length=64, dimension K=106 and effective rate R[C]=106/128. The first component code $C_1 = RM_{6,1}$ has a 4-section 32-state trellis, the second component code $C_2 = RM_{6,3}$ has a 4-sections 1024-state trellis and the third component code $C_3 = RM_{6,4}$ has a 4-sections 32-state trellis and each transition between two states consists of 2^{11} number of parallel branches !

Hence, it is obvious that maximum likelihood decoding of this code is not practical even with hard-decision multi-stage decoding. We use hard-decision multi-stage majority-logic decoding for each component code. Each component code is decoded by a majority-logic decoder. The error performance of this code is shown in figure 4.13(a). Note that the coding gain is 2.5 db over the uncoded QPSK at BER of 10^{-5}.

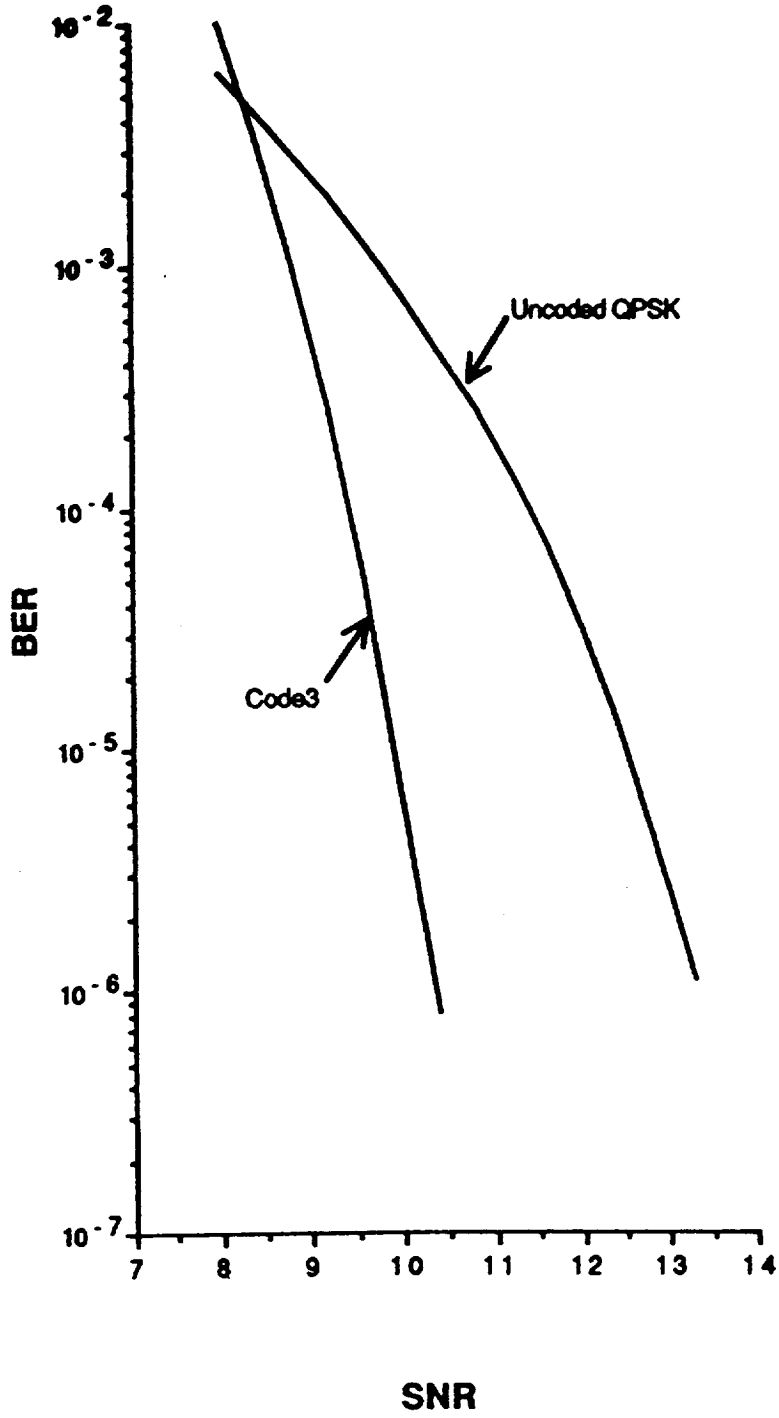


Fig.4.13(a) Error performance of Code3 with hard-decision MSD

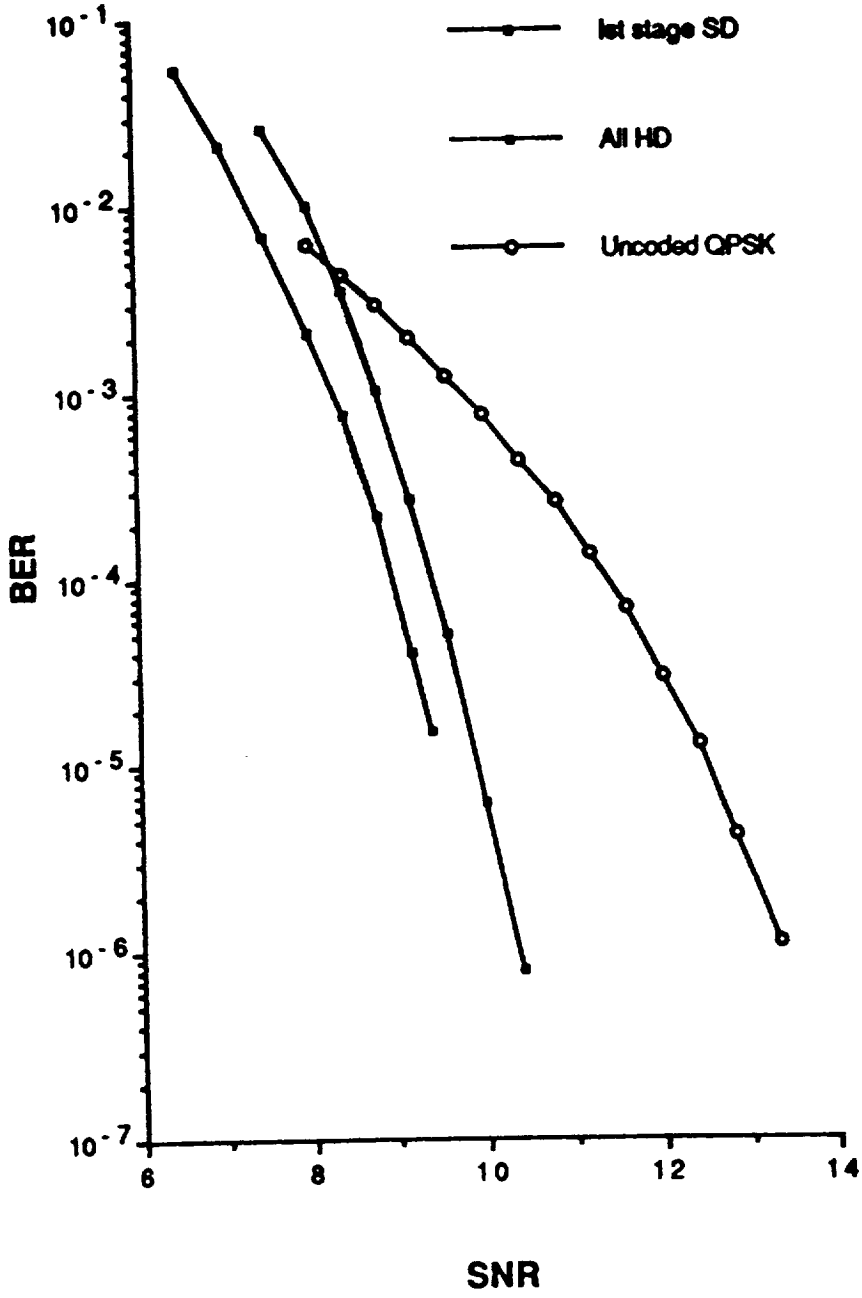


Fig. 4.13(b) Error performance of Code3 with Mixed MSD

This code was also simulated for soft-decision maximum likelihood decoding at first stage and hard decision majority logic decoding at second and third stages. The error performance with this decoding scheme is shown in figure 4.13(b). Note that using this scheme improves the performance by 0.5db at BER of 10^{-5}.

Thus, hard-decision multi-stage decoding further reduces the decoding complexity of the modulation codes while stills maintains reasonable coding gain over the uncoded system.

4.4 CONSTRUCTION OF MULTI-LEVEL CODES USING BCH CODES AS COMPONENT CODES AND HARD DECISION MSD

Some basic multi-level codes employing the multi-level construction method (discussed in chapter 1) were constructed using primitive BCH codes as component codes and their performance have been evaluated by computer simulation for hard-decision multi-stage decoding.

Let BCH(n,k,t) denotes a primitive BCH code of length n, where k= number of message bits and t= designed error correcting capability of the code. The designed minimum Hamming distance ∂ satisfies the following bound,

$$\partial \geq 2t+1$$

Each component code is decoded by classical Berlekamp decoding method for BCH codes.

The codes constructed are given in the table 4.1

Table 4.1 Multi-level codes using BCH codes

Code	Component code $C_1 * C_2 * C_3$	Dimension K	Length	D[C]	R[C]	Coding gain (BER 10^{-6})
Code1	BCH(127,29,21) BCH(127,106,3) BCH(127,120,1)	255	127	12	1.003	2.60 db
Code2	BCH(127,36,15) BCH(127,106,3) BCH(127,120,1)	262	127	12	1.030	1.85 db
Code3	BCH(127,36,15) BCH(127,106,3) BCH(127,113,2)	255	127	14	1.003	1.85 db
Code4	BCH(127,50,13) BCH(127,99,4) BCH(127,106,3)	255	127	15	1.003	1.35 db
Code5	BCH(255,45,43) BCH(255,223,4) BCH(255,247,1)	515	255	12	1.009	3.15db
Code6	BCH(255,45,43) BCH(255,231,4) BCH(255,247,1)	523	255	12	1.025	3.10db

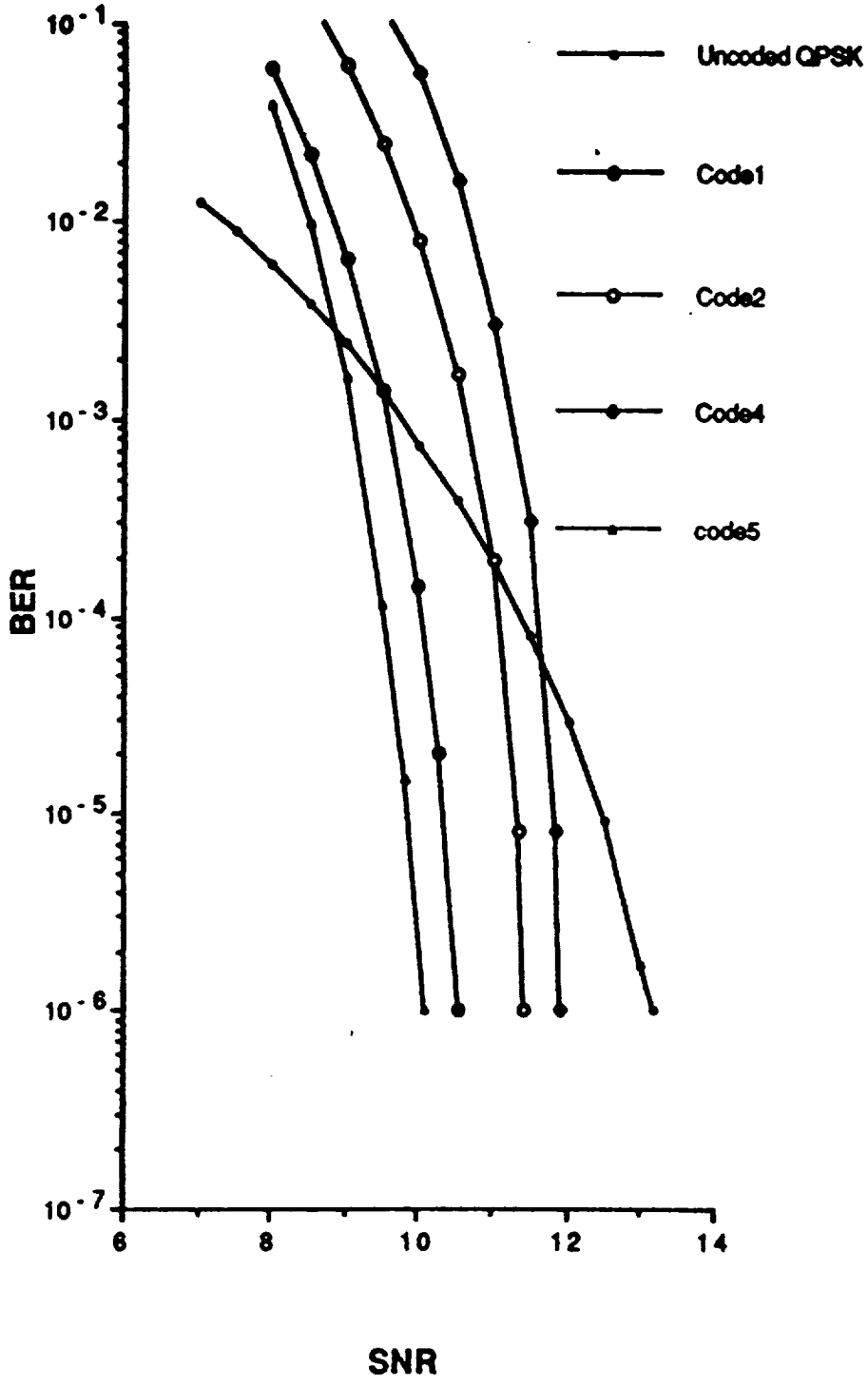


Fig. 4.14 Error performance of the new codes with hard decision MSD

CHAPTER 5

CONCLUSIONS

We have seen that non-uniform floating point to integer mapping of branch metric values reduces the branch and path metrics range for Viterbi decoder for modulation codes with negligible degradation in performance. Other non-uniform mapping can be designed to further reduce the metric range by dividing the entire floating point branch metric values into more than two groups. These schemes can be evaluated by computer simulation. Note that these schemes will work to reduce the hardware complexity of Viterbi decoder for TCM codes as well.

The throughput of Viterbi decoder for 4-state code can be improved by optimizing the path history circuit and control design. Attempts can be made to reduce the number of steps presented in the design.

The design of Viterbi decoder for 16-state code can be simplified if we use a 32-state 16-section trellis diagram instead of 8-state 4-section trellis diagram for the first level component code. In that case, total 32 2-state simple Viterbi decoders will be required to perform the decoding in parallel. Hardware requirement is increased, but better decoding throughput can be achieved. This scheme can be given the name 'multi-decoder decoding' (MDD). MDD can be used for any 3-level 8-PSK modulation

code for which (1) high decoding throughput is required, (2) the trellis diagrams for the second and third level component codes are relatively simple and (3) The number of codewords in the first level component code is small. The idea is to combine the trelleses of second and third component codes and then combine the resulting trellis with each codeword in the first component code. If there are n number of codewords in the first component code, then overall trellis diagram for the modulation code will consist of n number of identical trelleses of low complexity without cross connections among them. Each of these small trelleses can be processed in parallel by a separate Viterbi decoder of low complexity. Since these small decoders are identical, they can be built easily using VLSI technology. Also, note that for MDD, the first level component code need not have trellis structure.

Soft-decision multi-stage decoding reduces the hardware complexity drastically and it is suboptimum. The difference in performance between overall optimum decoding of modulation code and soft-decision MSD is very small, a fraction of db in coding gain. Furthermore, this difference becomes even smaller at higher SNR. Hence, the soft-decision MSD offers very good trade-off between performance and decoding complexity. Also, the multi-stage decoding creates pipelined parallelism in decoding process, which is desirable for high speed systems. For multi-stage decoding of multi-level code, it might be a good idea to construct the multi-level codes with

$$\partial_1 d_1 > \partial_i d_i,$$

where $2 \leq i \leq l$ and l is the number of levels.

The performance of multi-level modulation code for optimum and suboptimum decoding is determined mainly by the minimum squared Euclidean distance of the code. Also, if the two multi-level code has the same minimum squared Euclidean distance, the one with the shorter length is better than the other one as far as error performance for optimum and suboptimum decoding is concerned.

For hard-decision MSD, it is the minimum Hamming distance of the first level component code and not the minimum squared Euclidean distance of the modulation code, which determines the error performance. Hence, if the hard-decision MSD is to be performed, the first component code must be chosen to have as high Hamming distance as possible to achieve reasonable coding gain over uncoded system. Also, long and powerful codes should be used as component codes.

If the soft-decision maximum likelihood decoding is performed at the first-stage (this is possible, since the first stage has the most powerful code which has, in general, trellis diagram of reasonable complexity) and hard-decision decoding at other stages, then most of the loss (1.5-2.0 db) resulting from the hard-decision decoding at all the stages, can be recovered. This could be

another attractive and practical scheme for decoding of multi-level modulation code.

Finally, the advantages of multi-level codes with multi-stage decoding over trellis codes are as follows. Multi-level codes provide a flexible choice of the trade-off between coding gain, decoding complexity and decoding delay. The decoding complexity of trellis codes increase twice as the coding gain increases 0.4 db by optimum Viterbi decoding algorithm [22]. Multi-stage decoding algorithm for multi-level codes can achieve the designed distance with relatively small decoding complexity. Moreover, unlike trellis codes, good multi-level codes can be constructed by using previously known codes and proper mapping technique.

REFERENCES

[1] G. Ungerboeck, "Channel Coding with Multilevel/Phase Signals," IEEE Trans. on Information Theory, Vol. IT-28, pp. 55-67, January 1982.

[2] S. I. Sayegh, "A Class of Optimum Block Codes in Signal Space," IEEE Trans. on Communications, Vol. COM-34, No. 10, pp. 1043-1045, October 1986.

[3] R. M. Tanner, "Algebraic Construction of Large Euclidean Distance Combined Coding/Modulation Systems," presented at IEEE International Symposium on Information Theory, October 6-9, 1986, Ann Arber, Michigan, and also Technical Report UCSC-CRL-87-7, June 5, 1987.

[4] G. Ungerboeck, "Trellis-Coded Modulation with Redundant Signal Sets, Part I: Introduction," IEEE Communication Magazine, Vol. 25, No. 2, February 1987.

[5] G. Ungerboeck, "Trellis-Coded Modulation with Redundant Signal Sets, Part II: State of the Art," IEEE Communication Magazine, Vol. 25, No. 2, February 1987.

[6] T. Kasami and S. Lin, "Bandwidth Efficient Block Codes for M-ary PSK Modulation," Technical Report to NASA Goddard Space Flight Center, Greenbelt, MD. December 1987.

[7] T. Kasami, T. Takata, T. Fuziwara and S. Lin, "A Concatenated Coded Modulation Scheme for Error Control," Technical Report to NASA Goddard Space Flight Center, Greenbelt, MD. September 1988.

[8] A. J. Viterbi, "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm," IEEE Trans. on Information Theory, Vol. IT-13, pp. 260-269, April 1967.

[9] Jerrold A. Heller and Irwin Mark Jacobs, "Viterbi decoding for Satellite and Space Communication," IEEE Trans. on Communication Technology, Vol. COM-19, No. 5, pp. 835-848, October 1971.

[10] G. D. Forney, Jr., "The Viterbi algorithm," Proc. IEEE, Vol. 61, pp. 268-278, March 1973.

[11] S. Lin and D. J. Costello, Jr., Error Control Coding: Fundamental and Application, Prentice Hall: Englewood Cliffs, NJ, 1983.

[12] G. C. Clark and J. B. Cain, Error-Correction Coding for Digital Communications, Plenum Press, NY, 1982.

[13] Hari Krishna and Kou-Yu-Lin, "A Parallel SIMD Implementation of Viterbi Algorithm for Decoding Convolutional Codes," Research report, Department of Electrical and Computer Engineering, Syracuse University, Syracuse, NY.

[14] R. J. F. Fang, "A Coded 8-PSK System for 140Mb/s Information Rate Transmission Over 80 MHz Non-linear Transponder," COMSAT Laboratories, Clarksburg, Maryland. Paper presented at the 7th International Conference on Digital Satellite Communication (ICDSC-7) 12-16 May, 1986, Munchen, Federal Republic of Germany.

[15] Roksana Boreli, David Coggins, Branka Vucetic and Shu Lin, "VLSI Viterbi decoder for a BCM Code," Technical Report, University of Sydney and University of Hawaii.

[16] Hemant K. Thapar, "Real-Time Application of Trellis Coding to High Speed Voiceband Data Transmission," IEEE Journal on Selected Areas of Communications, Vol. SAC-2, No. 5, pp. 648-658, September 1984.

[17] Gerhard Fettweis and Heinrich Meyr, "Parallel Viterbi Algorithm Implementation: Breaking the ACS Bottleneck," IEEE Trans. on Communication, Vol. 37, No. 8, August 1989.

[18] Andries P. Hekstra, "An Alternative to Metric Rescaling in Viterbi Decoders," IEEE Trans. on Communications, Vol.37, No. 11, pp. 1220-1222, November 1989.

[19] S. G. Wilson, "Rate 5/6 Trellis-Coded 8-PSK," IEEE Trans. on Communications, Vol. COM-34, pp. 1045-1049, October 1986.

[20] A. R. Calderbank, "Multilevel Codes and Multistage Decoding," IEEE Trans. on Communications, Vol. 37, No. 3, pp. 222-229, March 1989.

[21] Shu Lin, "Multi-level Modulation Codes and Multi-Stage Decoding," Technical Report to NASA Goddard Space Flight Center, Greenbelt, MD, February 1990.

[22] Jiantian Wu, Daniel J. Costello, Jr., Shu Lin, "A Study of Multi-level Trellis Codes Part I - Construction," (Draft) February 1990.