National Library of Canada
Acquisitions and Bibliographic Services Branch
395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale du Canada
Direction des acquisitions et des services bibliographiques
395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Canada

Capability of Permutation Decoding of Cyclic Codes

Ming Jia

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfilment of the Requirements

for the Degree of Doctor of Philosophy at

Concordia University

Montréal, Québec, Canada

September, 1995

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Mr. Ming Jia**

Entitled: **Capability of Permutation Decoding of Cyclic Codes**

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

complies with the regulations of this University and meets the accepted standards with respect to originality quality.

Signed by the final examining committee:

Dr. R.M.H. Cheng	Chair
Dr. T. Aaron Gulliver	External Examiner
Dr. Jaroslav Opatrny	External to Program
Dr. Jeremiah F. Hayes	Examiner
Dr. Mohammad R. Soleymani	Examiner
Dr. Tho Le-Ngoc	Supervisor
Dr. Anader Benyamin-Seeyar	Supervisor

Approved by _____

Dr. Otto Schwelb
Graduate Program Director
Department of Electrical and Computer Engineering

___ _____ 19____ _____

Dr. Donat Taddeo
Dean, Faculty of Engineering and Computer Science

ABSTRACT

Capability of Permutation Decoding of Cyclic Codes

Ming Jia, Ph.D.

Concordia University, 1995

Error-trapping decoding employs a very simple combinational logic circuit for error detection and correction. It is most effective for decoding single-error-correcting codes, some short double-error-correcting codes, and single burst-error-correcting codes. However, when the error-trapping is applied to long and high rate codes with large error-correcting capability, it becomes very ineffective and much of the error-correcting capability of the codes is sacrificed.

Permutation decoding is essentially an improved variation of error-trapping decoding, with extended capability and effectiveness. This thesis aims to analyze the performance of permutation decoding. The study determines whether a specific code is permutation decodable (PD), and if it is PD, how many permutation steps are needed to decode the code.

The *tight* lower bounds for 2-step and 3-step (T, U) permutation decodable cyclic codes (for odd-valued t) have been investigated in [33]. In the first part of this thesis, the *tight* lower bounds for 3-step (T, U) permutation decodable cyclic codes (for even-valued t) are derived to complete the investigation on the bounds of 3-step (T, U) permutation decodable codes. With these bounds, we can determine if a given cyclic code is 3-step (T, U) permutation decodable or *not*.

In the second part, this thesis investigates the performance of multiple-step (T, U) permutation decoding of cyclic codes. At first, the characteristics of the error-free gaps of an error pattern in different permutation domains are investigated. It is shown that the gap lengths in the error pattern after a U permutation can be expressed as linear combinations of the gap lengths in the error pattern before U permutation. The relationship of gap lengths between *any* two consecutive or non-consecutive permutation domains is

derived. An insight into how the gap lengths change and therefore the capability of (T, U) permutation decoding increases is also given.

The characteristics of the code rate are studied in this thesis. From the concept of the optimum permutation step, the way to estimate the number of U permutations needed to decode a permutation decodable code is shown. The code rate of (T, U) permutation decodable codes R' and the code rate of error-trapping decodable codes are compared to determine the effective region of the (T, U) permutation decoding technique.

Permutation decoding using primitive elements as multipliers is also examined. It is shown that by using primitive elements of a prime field as multipliers the capability of permutation decoding to decode cyclic codes of prime length can be increased. The implementation structures for both serial and parallel permutation decoding are also discussed. A scheme to combine both (T, U) and M_i permutation decoding strategies is introduced. In this scheme, after using M_i permutation to permute a code into an equivalent code, parallel (T, U) permutations can be used in different code domains. In this way, the error-pattern detection capability of permutation decoding can be dramatically increased while the decoding circuit remains simple.

ACKNOWLEDGEMENTS

I am deeply grateful to my supervisors Dr. Tho Le-Ngoc and Dr. Anader Benyamin-Seeyar for their invaluable advice, encouragement and patience throughout this research.

Grateful acknowledgment is also expressed to my former supervisors Dr. Jeremiah F. Hayes and Dr. Yu Wei for their guidance and encouragement during the early stage of my study; to the South-East University, Nanjing, China, for giving me the opportunity to come and study at Concordia University.

To my family
and friends.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

1.1 Introduction

Controlling errors for reliable reproduction of data becomes more and more important for efficient digital data transmission and storage systems. The possibility of achieving reliable digital transmission over a noisy channel was originally introduced by Shannon in 1948 [1, 2].

Since the appearance of Shannon's classic papers in 1948 and 1949, a great deal of research has been devoted to the problem of designing efficient schemes by which information can be coded for reliable transmission across channels which are corrupted by noise. From a practical standpoint, the essential limitation of all coding and decoding schemes proposed to date has not been Shannon's capacity but the complexity (and cost) of the decoder [4]. For this reason, efforts have been directed toward the design of coding and decoding schemes which could be easily implemented.

The major engineering problem of error control coding is to design and implement the channel encoder/decoder pair such that (1) information can be transmitted (or recorded) in a noisy environment as fast as possible, (2) reliable reproduction of the information can be obtained at the output of the channel decoder, and (3) the cost of implementing the encoder and decoder falls within acceptable limits [3]-[23].

The basic idea of error correcting and detecting codes is to increase the distance between

information-bearing signals by the addition of redundancy [5]. There are two approaches to adding this redundancy: block codes and convolutional codes. The encoder for a block code breaks the continuous sequence of information digits into k-symbol sections or blocks. It then operates on these blocks independently according to a particular rule employed by the code so that each possible information block is associated with an n-tuple of channel symbols (where $n > k$) which is called a *code word*. Since the n-symbol output code word depends only on the corresponding k-bit input message, the encoder is memoryless, and can be easily implemented with a combinational logic circuit. The encoder for a convolutional code operates on the information sequence without breaking it up into independent blocks. The encoder processes the information continuously and associates each long information sequence with a code sequence containing somewhat more digits. The encoder breaks its input sequence into k_0-symbol blocks, where k_0 is usually a small number. Then, on the basis of this k_0-tuple and the preceding information symbols, it emits an n_0-symbol section of the code sequence. This process can be easily implemented by passing the data sequence through a linear shift register circuit. The redundant data stream is derived by modulo-2 addition of the sample contents of the shift register. The redundancy lies in the fact that for every input bit there is more than one output bit from the encoder. For the purpose of this thesis, we restrict our attention to a subclass of all block codes, the *linear block codes* [4].

A desirable property for a linear block code to possess is a systematic structure as shown in Figure 1.1.

A linear systematic $(n,\ k)$ block code is completely specified by a $k \times n$ generator matrix

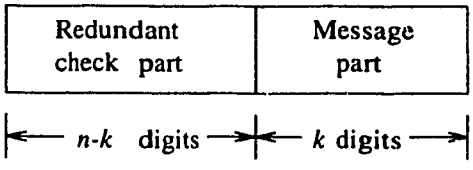


Figure 1.1: Systematic format of a code word.

$\mathbf{G}$ of the following form:

$$\mathbf{G} = \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \mathbf{g}_2 \\ \vdots \\ \mathbf{g}_{k-1} \end{bmatrix} = \begin{bmatrix} p_{00} & p_{01} & \cdots & p_{0,n-k-1} & 1 & 0 & 0 & \cdots & 0 \\ p_{10} & p_{11} & \cdots & p_{1,n-k-1} & 0 & 1 & 0 & \cdots & 0 \\ p_{20} & p_{21} & \cdots & p_{2,n-k-1} & 0 & 0 & 1 & \cdots & 0 \\ & & & & \ddots & & & & \\ p_{k-1,0} & p_{k-1,1} & \cdots & p_{k-1,n-k-1} & 0 & 0 & 0 & \cdots & 1 \end{bmatrix},$$

where p_{ij} is from a prime field $GF(p)$[1]. Let $\mathbf{u} = (u_0, u_1, \ldots, u_{k-1})$ be the message to be encoded. The corresponding code word is

$$\begin{aligned} \mathbf{v} &= (v_0, v_1, v_2, \ldots, v_{n-1}) \\ &= (u_0, u_1, \ldots, u_{k-1}) \cdot \mathbf{G}. \end{aligned} \tag{1.1}$$

So, the rightmost k digits of a code word $\mathbf{v}$ are identical to the information digits $u_0, u_1, \ldots, u_{k-1}$, and the leftmost $n-k$ redundant digits are linear sums of the information digits.

1.2 Description of Cyclic Codes

The most important class of linear block codes is the class of cyclic codes in which each codeword is a cyclic shift of another codeword, i.e., if $(v_0, v_1, v_2, \ldots, v_{n-1})$ is a codeword, then so is $(v_{n-1}, v_0, \ldots, v_{n-3}, v_{n-2})$. Cyclic codes are of particular interest because of their easy implementation [5]. Most linear codes used in practice are cyclic codes [3]. The structure and description of cyclic codes are based on polynomials whose coefficients are from a field with q elements. These polynomials constitute an extented field of a prime field $GF(q)$, which is denoted by $GF(q^m)$[2]. Each element in $GF(q^m)$ is a polynomial which represents a cyclic codeword.

[1] Finite fields are also called *Galois* fields, in honor of their discoverer. A Galois field of q element is denoted by $GF(q)$.

[2] For any positive integer m, it is possible to extend the prime field $GF(q)$ to a field of q^m elements which is called an extension field of $GF(q)$ and is denoted by $GF(q^m)$[3].

There are many articles in the literature describing cyclic codes in detail, so they will not be repeated here. The reader interested in the structure and properties of cyclic codes may find [3, 4, 5, 6, 12, 14, 15, 16, 17, 34, 44] useful.

On memoryless channels, the channel noise affects each transmitted symbol independently, hence transmission errors occur randomly in the received sequence. The codes devised for correcting random errors are called *random-error-correcting codes*. On channels with memory, the noise is not independent from transmission to transmission, therefore transmission errors occur in clusters or bursts. The codes devised for correcting burst errors are called *burst-error-correcting* codes.

In general, codes for correcting random errors are not efficient for correcting burst errors. Cyclic codes are very effective for both random- and burst-error detection and correction. Many effective cyclic codes for correcting burst errors have been discovered [3, 10, 11].

The error-correcting capability of a given code is largely determined by the minimum pair-wise Hamming distance d between any two codewords. In theory, a knowledge by the decoder of all the 2^k possible codewords enables it to compare the received word with all such codewords. In the BSC (Binary Symmetric Channel) case, the best decoding decision is to select the codeword having the smallest Hamming distance from the received word. On this basis, the decoder guarantees the correction of any error of weight $t = \lfloor \frac{d-1}{2} \rfloor$ or less, where $\lfloor\ \rfloor$ denotes the integer portion of the enclosed expression.

1.3 Decoding Techniques for Cyclic Codes

The Meggitt decoding technique can be considered as the most general decoder for cyclic codes, and it applies in principle to any cyclic code However, the complexity of its error-pattern detection circuit increases rather rapidly as the number of errors to be corrected grows. There are cases in which the error-pattern detection circuits are simple, but in general, refinements are necessary for practical implementation. For decoding Bose, Chaudhuri, and Hocquenghem (BCH) codes, the iterative algorithm devised by Peterson, Berlekamp, and Chien [40, 4, 41] is much simpler than the Meggitt decoder.

In general, algorithms devised for decoding BCH codes are composed of three parts, namely,

1. Syndrome computation;

2. Finding the error-location polynomial;

3. Computation of error-location numbers and error correction.

These algorithm require the implementation of multiplication and division of two variables over a Galois field.

There are some other decoding schemes which do not require the implementation of Galois field multiplication and division of two variables and they are much simpler than the algorithms based on algebraic calculation. However, these algorithms have their own limitations. They can only decode relatively short and low rate codes (e.g., error-trapping) and/or are limited to a small number of codes which are supposed to have the necessary structural characteristics (e.g., majority-logic decoding).

1.4 Majority-Logic Decoding

Majority-logic decoding is an effective scheme for decoding certain classes of block codes, especially for decoding certain classes of cyclic codes. The first majority-logic decoding algorithm was devised in 1954 by Reed [42] for a class of multiple-error-correcting codes discovered by Muller [43]. Reed's algorithm was later extended and generalized by many coding investigators. The first unified formulation of majority-logic decoding algorithms was due to Massey [8]. Most majority-logic decodable codes found so far are cyclic codes [3] or quasi-cyclic codes [19].

Consider an (n, k) cyclic code C with dual code C_d[3]. Suppose there exist J vectors in the dual code,

$$\mathbf{w}_1 = (w_{10}, w_{11}, \ldots, w_{1,n-1}),$$

[3]The dual code of C is the null space of the (n, k) linear code C generated by matrix G (i.e., for any $\mathbf{v} \in C$ and any $\mathbf{w} \in C_d$, $\mathbf{c} \cdot \mathbf{w} = 0$).

$$\begin{aligned} \mathbf{w}_2 &= (w_{20}, w_{21}, \ldots, w_{2,n-1}), \\ &\vdots \\ \mathbf{w}_J &= (w_{J0}, w_{J1}, \ldots, w_{J,n-1}), \end{aligned} \tag{1.2}$$

which have the following orthogonal properties:

- $w_{1,n-1} = w_{2,n-1} = \cdots = w_{J,n-1} = 1$,
- For $i \neq n-1$, there is at most one vector whose i-th component is a "1".

Then any error pattern of $\lfloor \frac{J}{2} \rfloor$ or fewer errors can be corrected.

Majority-logic decoding is a remarkably simple technique but it is limited to a small number of codes which have the necessary structural characteristics. When J is small compared to the minimum distance d_{min} of the code, one-step majority-logic decoding becomes very inefficient, and much of the error-correcting capability is sacrificed.

The concept of one-step majority-logic decoding can be generalized in such a way that many cyclic codes can be decoded by employing several levels of majority-logic gates. Multiple-step majority-logic decoding can increase the number of orthogonal vectors by loosening the restriction from orthogonal on a special digit to orthogonal on a set of error digits. Because any error pattern of $\lfloor \frac{J}{2} \rfloor$ or fewer errors can be corrected by majority-logic decoding, more cyclic codes can be decoded. A code is said to be *L-step majority-logic orthogonalizable* or *L-step majority-logic decodable* if L steps of orthogonalization are required to make J orthogonal vectors which are orthogonal on a single error digit. A code is said to be *completely L-step orthogonalizable*[4] if J is one less than the minimum distance of the code. Since majority-logic gates are used to estimate selected sums of error digits at each step of orthogonalization, a total of L levels of majority-logic gates are required for decoding. The number of gates required at each level depends on the structure of the code. For an (n, k) L-step majority-logic decodable code, no more than k majority-logic gates are ever required [8], but for a given L-step majority logic decodable cyclic code, *there*

[4]This process of getting orthogonal vectors which are orthogonal on a set of error digits until a set of J or more check sums orthogonal on only a single error digit obtained is called *orthogonalization* [8].

is no known systematic method for minimizing the number of majority-logic gates except trial-and-error. For almost all known classes of L-step majority-logic decodable codes, the number of majority-logic gates required is an exponential function of L [3]. For large L, the decoder is likely to be impractical.

It has been shown [8] that the $(2^m - 1, 2^m - m - 1)$ Hamming code is completely orthogonalizable in $m - 1$ steps. Thus, the Hamming codes can be decoded by majority-logic decoding. However, the error-trapping technique for Hamming codes which will be described in Section 1.6 can be more simply implemented than majority-logic decoding [3].

1.5 Information Set Decoding

The use of information sets as the practical basis of a decoding algorithm for group codes was first proposed by Prange in 1962 [26]. In an (n, k) linear block code an information set is defined to be any set of k positions in the code word that can be specified independently. The remaining $(n-k)$ positions are referred to as the parity set. Since the symbols contained in the information set can be specified independently, they uniquely define a code word. If there are no errors in these positions, then the remaining symbols in the transmitted code word can be reconstructed. This property provides the basis for all information set decoding algorithms. In accordance with this property, the information set decoding procedure may be described in three steps:

1. *step 1.* Select several different information sets according to *some rule*;
2. *step 2.* Construct a code word for each set assuming that the symbols in the information set are correct;
3. *step 3.* Compare each of the hypothesized code words with the actual received sequences and select the code word which is closest.

Given a code of error correcting capability t, if for every possible error pattern there always exist at least one information set which is error free, then the code is *information set decodable.*

There are mainly two ways to select information sets [12], namely:

- Predetermined sets;
- Random search.

Generally, there are no satisfactory solutions to determine the minimum number of distinct sets and how to find them. Notice that the required size of the collection of information sets is very sensitive to the number of errors that must be corrected, and it has been shown that the smallest number of information sets needed to decode a code N_{cov} is bounded by [13]:

$$N_{cov} \geq \lceil \frac{n}{n-k} \lceil \frac{n-1}{n-k-1} \cdots \lceil \frac{n-t+1}{n-k-t+1} \rceil \cdots \rceil \rceil \tag{1.3}$$

where the symbol $y = \lceil x \rceil$ denotes the smallest integer, y, such that $y \geq x$. From (1.3) it can be seen that for high rate codes (k is large) with large error-correcting capability, N_{cov} is a very large number. For example, for a (127, 106, 7) BCH code, the smallest number of information sets N_{cov} is bounded by[5]

$$N_{cov} \geq 343,$$

while for a (255, 123, 39) BCH code, N_{cov} is bounded by

$$N_{cov} \geq 20,155,400$$

In practical applications, it is very difficult to find an optimum information set with such a large size. Generally speaking, information set decoding has its difficulty in finding the information sets[6]. Because of the considerable inherent algebraic structure of cyclic codes, a practical information set decoding implementation of cyclic codes can be very simple [21, 24]. The limitation of this technique is that it may use only a small part of the information sets which are needed to decode a code to its full error-correcting capability.

[5]This code cannot be decoded by (T, U) permutation decoding. However, when the primitive element permutation decoding technique (Chapter 4) is applied, the BCH code of the same code length with error correcting capability $t = 2$, 3, 4, and 5 can be decoded.

[6]Even if the information sets needed to decode a certain code are known, the logic circuit needed to generate these information sets can be extremely complex.

1.6 Error-Trapping Decoding

A distinctive property of cyclic codes is that every successive k information bits form an information set. Based on this property Mitchell and Rudolph [21, 24] independently devised a decoding technique using a very simple combinational logic circuit for error detection and correction[7].

Error-trapping chooses information sets by simply cyclic shifting a code word one bit at a time and always taking the first k bits as a new information set. Since after the n-th cyclic shift the original received code word is obtained, the number of distinct information sets is fixed, and is equal to the code length, n. Obviously, for a code to be error-trapping decodable, there should exist at least one set of k consecutive bits which is error free in every possible error pattern (notice that the information sets are formed by k consecutive bits).

Error-trapping is most effective for decoding single-error-correcting codes, some short-double-error-correcting codes, and single burst-error-correcting codes. However, when the error-trapping is applied to long and high rate codes with large error-correcting capability, it becomes very ineffective and much error-correcting capability is sacrificed. This is because for long and high rate codes with large error-correcting capability, it is unlikely that the errors will be confined to $(n-k)$ consecutive positions (so that the chosen information set is error free). Again, let us take the (127, 106) BCH code as an example. An error-trapping decoder provides $n = 127$ information sets. But from the previous discussion, we know that for an information set decoder to decode a (127, 106) BCH code without sacrificing its error-correcting capability, the lower bound on the number of information sets required is 343. By comparing the number of information sets provided by error trapping (which is 127) with this lower bound, it is easily seen that the basic error-trapping technique is unable to provide a sufficient number of information sets to correct all the error patterns

[7] Error-trapping technique can also be viewed as a practical variation of the general decoding method of Meggitt [3], in which the error-pattern is detected when the error-pattern is identical to the syndrome of the shifted version of the received code word.

within the error-correcting capability of the code.

Since 1962, several improved error-trapping methods have been devised in an effort to extend the capability and effectiveness of the error-trapping decoder for multiple-error-correcting cyclic codes [24]-[27]. These methods have been developed to achieve the same purpose: to get k (information bits) consecutive error free positions in a code word. The main two methods, namely Kasami's error-trapping and permutation decoding, use two different ways to get these k error free consecutive positions. Kasami used covering polynomials to eliminate the effect of errors that can not be moved into the $(n-k)$ parity check positions, and permutation decoding uses permutation to permute the error positions to get k error free consecutive positions. Both methods increase the capability of the error-trapping decoder but they are still only applicable to relatively short and low rate code. However, their restrictions are different. For a Kasami decoder, when the code length n and error-correcting capability t become large, the number of threshold gates required in the error-detecting logic circuits becomes very large and impractical. For a permutation decoder, when the code length n and error-correcting capability t become large, the number of permutation steps required to decode a code increases so that the decoder becomes very slow (if parallel processing is used, the decoding time will not change, but the complexity will increase [38]).

1.7 Kasami's Decoder

In 1964, Kasami proposed a decoder which is a variant of the information set error-trapping technique [25]. Kasami's error-trapping technique uses covering polynomials to eliminate the effect of errors which cannot be moved into the $n-k$ parity check positions. It requires finding a set of polynomials $[Q_j(x)]_{j=1}^{N}$ of degree $k-1$ or less, so that for any correctable error pattern $e(x)$, there is one polynomial $Q_j(x)$ such that $x^{n-k}Q_j(x)$ matches the message section of $e(x)$ or the message section of a cyclic shift of $e(x)$. The polynomials $Q_j(x)$ are called the "covering polynomials".

Although these decoding techniques are rather simple in principle, they are still only

applicable to relatively short and low rate codes. When the code length n and error-correcting capability t become large, the number of distinct information sets or covering polynomials in $[Q_j(x)]_{j=1}^{N}$ become very large. Also, to select the set of covering polynomials $[Q_j(x)]_{j=1}^{N}$ for a specific code is not an easy problem [3].

1.8 Permutation Decoding

The interest of this research is in the decoding technique known as *permutation decoding*, which is essentially an error-trapping technique introduced by Prange [26]. A serial decoder based on this treatment was given by MacWilliams [27], who made use of code preserving permutation sets to obtain k error-free positions from which the rest of the code word could be reconstructed.

For every cyclic code C in the vector space F^n of dimension n, with symbols from the finite field $F = GF(q)$, where q is the size of the field, there are various code preserving permutations [33]. In this thesis we specifically use the following group (T, U) permutation which is applicable to cyclic codes:

Let C be a $(n,\ k,\ 2t+1)$ cyclic code over $GF(q)$. If $c(x)$ is a code polynomial, that is $c(x) \in C$, then

$$c(x) = \sum_{j=0}^{n-1} b_j x^j,$$

where b_j is a symbol from $GF(q)$. The group (T, U) permutation is defined as follows:

$$c'(x) = T^{\beta}c(x) = \sum_{j=0}^{n-1} b_j x^{j+\beta} \quad (mod\ \ x^n - 1),$$

and

$$c''(x) = U^i c(x) = \sum_{j=0}^{n-1} b_j x^{(p^i j)} \quad (mod\ \ x^n - 1).$$

where p is the characteristic of $GF(q)$, and the symbols of the cyclic code C are from $GF(q)$. If p is relatively prime to n, then the code is invariant under group (U) permutation [27] . So, when p is relatively prime to n, for $c(x) \in C$,

$$U^i T^{\beta}[c(x)] = (([c(x)]^{p^i})x^{\beta} \ \ mod\ \ (x^n - 1))$$

is also a code word.

Suppose $r(x) = c(x) + e(x)$ is the received codeword polynomial where $e(x)$, the error pattern polynomial, is of weight t or less. Then the syndrome of the permuted received word

$$s_{i\beta}(x) = (x^{\beta}[r(x)]^{p^i} \ mod \ (x^n - 1)) \ mod \ g(x) \tag{1.4}$$

is

$$s_{i\beta}(x) = (x^{\beta}[e(x)]^{p^i} \ mod \ (x^n - 1)) \ mod \ g(x) \tag{1.5}$$

We define $e(x)$ to be a permutation decodable (PD) pattern if values of i and β exist such that $e_{i\beta}(x) = ((x^{\beta}[e(x)]^{p^i}) \ mod \ (x^n - 1))$ has degree $(n - k - 1)$ or less. That is, all the errors in the permuted $e(x)$ are confined to the first $n - k$ parity-check positions and $s_{i\beta}(x) = e_{i\beta}(x)$. In this case the error pattern is $((x^{-\beta} \cdot s_{i\beta}(x))^{p^{n-i}} mod \ (x^n - 1))$ [6]. If these conditions on i and β hold for every $e_{i\beta}(x)$, such that the error $e(x)$ is PD with $i = 0, 1, \ldots, s < \gamma$, where γ is the least integer satisfies that $(p^{\gamma} = 1 \ mod \ n)$, then we say that the code C is $(s+1)$-step PD. It should be noticed that for a s-step permutation decodable codes, the number of information sets that we can use is $s \times n$.

Permutation decoding does not need to select the distinct information sets or the set of covering polynomials and it is best suited to codes which are invariant under a large group of permutations [27]. In essence, permutation decoding increases the number of selected information sets according to a predetermined rule so that more error patterns can be corrected.

Recently, tight lower bounds on the code length n for $(n, k, 2t+1)$ cyclic codes have been found by using group (T, U) permutations, for: 1) 2-step (T, U) permutation decodable binary cyclic codes with t being odd or even valued and 2) 3-step (T, U) permutation decodable binary cyclic codes with $t = 2$ and odd-valued t [33]. In the next chapter, we extend these results for the case of even-valued t $(t \geq 4)$. The results show that the improvement of 3-step PD with respect to 2-step PD is proportional to the value of k or t, and this implies that this variation of error-trapping decoding can be applied to higher rate and larger error correcting cyclic codes.

1.9 Advances in This Work

In this thesis, we first derive tight lower bounds on the block code length n of three-step permutation-decodable cyclic codes. That is, for a fixed number of permutation steps $s = 3$, we present the tight upper bounds on the code rate of permutation decodable cyclic codes.

The more important contribution of the thesis is the study of the performance of multiple-step $(T,\ U)$ permutation decoding of cyclic codes. We derive the mapping matrix, which gives a comprehensive relation between gap lengths in different domains. We prove that the gap lengths in an error pattern after U permutation is actually a linear combination of the gap lengths in the error pattern before U permutation. We characterize the relationship of the error pattern gaps between *any* two consecutive or non-consecutive domains. This gives insight into how the gap lengths change and therefore how the capability of $(T,\ U)$ permutation decoding increases. The characteristics of the code rate of permutation decodable codes have also been studied in this thesis. From the concept of the optimum permutation step, we show how to estimate the required number of U permutation steps to decode a permutation decodable code. We compare the code rate R' of the permutation decodable codes and the code rate of error-trapping decodable codes, which shows the effective region of the $(T,\ U)$ permutation technique. In this effective region, some powerful *BCH* codes (e.g. (127, 113, 5), (127, 106, 7), etc.) can also be decoded.

The last contribution concerns permutation decoding using primitive elements as multipliers (M_i permutation decoding). We show that by using primitive elements of a prime field as multipliers we can increase the capability of permutation decoding method in decoding cyclic codes of prime length. The implementation structures for both serial and parallel permutation decoding are also discussed. Finally, the idea of combining both $(T,\ U)$ and M_i permutation decoding methods is examined, i.e., after using M_i permutation to permute a code to an equivalent code, parallel $(T,\ U)$ permutation can also be used in different code domain. In this way, the error-pattern detection capability of permutation decoding can be increased dramatically while the decoding circuit complexity remains simple.

It may be noted that some of the results given in this thesis have been presented

published elsewhere [35]-[39].

1.10 Plan of the Thesis

This thesis is divided into five chapters. A brief description of each chapter follows.

Chapters 2 through 4 cover the main contributions of the thesis. In Chapter 2, we derive the tight lower bounds on the code length n, or equivalently upper bounds on the code rate $\frac{k}{n}$. A study on the performance of multiple-step $(T,\ U)$ permutation decoding of cyclic codes is presented in Chapter 3. It is shown that the relation between the gap-length vectors of the permuted error polynomials are related by a linear mapping. Based on this property, the capability of $(T,\ U)$ permutation decoding is studied and the relationship between the number of permutations s and code parameters n, k and t of a permutation decodable cyclic code is established. The idea of permutation decoding using primitive elements as multipliers is presented in Chapter 4. Some illustrative examples are also given to show the effectiveness of this method. Remarks and conclusions of the results of this work are given in Chapter 5, together with some suggestions for further study.

Chapter 2

TIGHT LOWER BOUNDS FOR 3-STEP PERMUTATION DECODABLE CYCLIC CODES

In Chapter 1, we introduced various cyclic codes decoders, including error-trapping and its variants. One of the improved forms of error-trapping is permutation decoding. In this chapter, we derive tight lower bounds on the code length n of $(n, k, 2t+1)$ cyclic codes which can be decoded by a permutation decoder.

First in Section 2.1, we present a summary of known results as the starting point of our work. Then, we introduce some preliminary notations and definitions in order to make this chapter self-contained. In Section 2.3, we derive the tight lower bounds for 3-step permutation decodable cyclic codes. Section 2.4 contains conclusions and some numerical results.

2.1 Summary of Known Results

Since MacWilliams proposed *(T, U) Permutation Decoding* in 1964, a lot of research has been done to determine the decodability of cyclic codes by using (T, U) permutations [29]-[33]. Here, we summarize the known lower bounds on n as follows:

1. A binary $(n, k, 2t+1)$ cyclic code C is 1-step PD if and only if:

$$n > kt\,.$$

Notice that 1-step permutation decoder is the original form of error-trapping technique and therefore 1-step PD codes can be decoded by cyclic shift permutations [27].

2. The bounds for 2-step PD $(n, k, 2t+1)$ cyclic codes are: [29]-[33]

   - When $t = 2$:

$$n > \frac{3k}{2}\,;$$

   - When $k = 2$:

$$n > \frac{3t}{2}\,.$$

   - When k and t are odd:

$$n > t \cdot k\,;$$

   - When k is even and t is odd:

$$n > t \cdot (k-1)\,;$$

   - When k is odd and t is even:

$$n > (t-1) \cdot k\,;$$

   - When k and t are even:

$$n > (t-1)(k-1) + 2\,;$$

3. The bounds for 3-step PD $(n,\ k,\ 2t+1)$ cyclic codes are: [33]

   - When k and t are odd:

$$n = t \cdot k - 2(2l+1), \quad if \quad k = 2l+3 \quad and \quad 0 \leq l < \frac{t-1}{2};$$

   - When k is even and t is odd:

$$n = t \cdot (k-1) - 2(2l+1), \quad if \quad k > 2l+4 \quad and \quad 0 \leq l < \frac{t-1}{2};$$

   - When $t = 2$:

$$n = k + \frac{k-1}{2} \quad for \quad k > 3;$$

   - When $k = 2$:

$$n \geq \frac{3t-1}{2}.$$

4. The following codes are not permutation decodable:

   - For any t, the $(n,\ k,\ 2t+1)$ cyclic codes with

$$n = \delta \cdot \tau,$$

     where $\delta \leq k$, and $\tau \leq t$, are not PD.

   - If the $(n,\ k,\ 2t+1)$ code is not s-step PD, then the $(n',\ k,\ 2t+1)$ codes with

$$n' = n - 2^{s-1} \cdot l, \quad for \quad l \geq 1,$$

     are not s-step PD.

   - For any t, the codes $(n,\ k,\ 2t+1)$ with

$$n = \frac{3}{4}\delta \cdot \tau,$$

     where $\delta \leq k$, $\tau \leq t$, and $\delta \cdot \tau \neq 0 \mod 8$, are not PD.

2.2 Notations and Definitions

Let an n-symbol error polynomial $E(x) = 1 + x^{e_1} + \ldots + x^{e_{t-1}}$ with weight t, over $GF(2)$, be represented by a vector $\hat{e}$, $\hat{e} = \{e_i,\ i = 0, 1, \ldots, t-1\}$ is called the Error Pattern (EP), where $e_0, e_1, e_2, \ldots, e_{t-1}$ are the error positions and $e_0 < e_1 < e_2 < \ldots < e_{t-1}$. For simplicity in analysis, it is assumed that one of the errors is fixed at the zero position ($e_0 = 0$). The gap length G in an EP is the number of consecutive error-free coordinate positions between any two consecutive error locations. The pattern associated with the result of permutation by U^i is referred to as the EP in the R_i-domain.

Suppose that the EP in the R_i-domain is not $(i+1)$-step PD, then the corresponding error locations in the R_i domain can be divided into two classes. Namely, Class 1 contains even-valued errors (presented as "o") and Class 2 contains odd-valued errors (presented as "x"). These "o" and "x" errors in the R_{i+1}-domain correspond to the error positions in the R_i-domain which are located before and after the $\lfloor \frac{n}{2} \rfloor$-th location in the pattern, respectively.

We then distinguish two types of patterns as follows:

1) γ-type pattern:

(o x o) or (x o x)

2) σ-type pattern:

(o x x o) or (x o o x)

The number of γ-type and σ-type patterns in an EP are denoted by N_γ and N_σ, respectively.

It is noted that there may be many other patterns in an EP. However, in our analysis we consider the worst case, and hence, we are only concerned with these two patterns. This is because it can be proven that the other types of patterns will result in smaller gap-lengths [33].

NOTE 1. All the variables with subscripts "e" or "o" are considered as even, or odd valued variables, respectively.

NOTE 2. When we say that an $(n, k, 2t+1)$ code is $(s+1)$-step PD, we mean that such a code is decodable with $S_{0\beta}(x)$, $S_{1\beta}(x)$, ..., $S_{s\beta}(x)$; we do not imply that such a code exists.

NOTE 3. The zero coordinate place, in the analysis of the EPs in all R_i-domains, $i \leq 1$, has a "double parity" feature. For n odd, the zero location is considered as an even-valued number for the first gap, and it is considered as an odd-valued number for the last gap in the pattern.

NOTE 4. In this thesis, whenever we refer to a code we mean a binary cyclic code.

2.3 Tight Lower Bounds for 3-Step Permutation Decodable Cyclic Codes

For 3-step PD codes, no bounds existed for the case of t an even number ($t \geq 4$). In this thesis, we have addressed this problem and in the form of Theorems we are going to extract tight lower bounds for the case even-valued t. First the results for the codes which are not 3-step PD are given in the form of Theorems as follows.

2.3.1 When k is odd

Theorem 2.1 *The* $(n, k_o, 2t_e+1)$ *codes with* $n = k_o(t_e-1) - 2(2l+1)$, *for*

$$l = \frac{t_e - 4}{2}$$

are not 3-step PD.

Proof of Theorem 2.1: It is sufficient to prove the theorem for an error vector of weight t_e that does not have any gap of length $G \leq k_o$ in the R_0-, R_1- and R_2-domains. Now, Let us consider an error pattern $\{e_i\}$ in R_2-domain:

$$\begin{cases} e_0 = 0, \\ e_i = i(k_o - 2) - k_o + 6, \quad for\ 1 \leq i \leq t_e - 1. \end{cases} \tag{2.1}$$

The gaps between any consecutive error position pairs in (2.1) are obtained as:

$$\begin{cases} G(e_1,\ e_0) = 3, \\ G(e_{i+1},\ e_i) = k_o - 3, \quad for\ 1 \le i \le t_e - 2, \\ G(n,\ e_{t_e-1}) = k_o - 3. \end{cases} \tag{2.2}$$

Thus, when $k_o \ge 5$, no gaps $G \ge k_o$ exist. Also, no gaps $G \ge k_o$ exist in the corresponding patterns in R_1- and R_2-domains. In the case $k_o = 3$, we consider the following error pattern in R_0-domain:

$$\begin{cases} e_i = i, \quad for\ 0 \le i \le t_i - 2, \\ e_{t_e-1} = t_e. \end{cases} \tag{2.3}$$

It can also be verified that none of the gap-lengths in the R_0-, R_1- and R_2-domains are greater than $k_o - 1$. This concludes the proof of Theorem 2.1.

Q. E. D.

Theorem 2.2 *The* $(n,\ k_o,\ 2t_e + 1)$ *codes with* $n = k_o(t_e - 1) - 2(2l + 1)$ *for*

$$l = \frac{k_o - 1}{2}$$

are not 3-step PD.

Proof of Theorem 2.2: Similar to the proof of Theorem 2.1, we give an error vector of weight t_e that does not have any gap of length $G \ge k_o$ in the R_o-, R_1- and R_2-domains. We also divide the proof into two cases: for $\frac{k_o-1}{2}$ being even and for $\frac{k_o-1}{2}$ being odd.

Case 1, for $\frac{k_o-1}{2}$ being even:

Consider the error pattern in the R_2-domain:

$$\begin{cases} e_0 = 0, \\ e_1 = 4, \\ e_2 = 8, \\ e_3 = k_o - 1, \\ e_i = (i-3)k_o, \quad for\ 4 \le i \le t_e - 1. \end{cases} \tag{2.4}$$

The gaps between any consecutive error position in Equation (2.4) are obtained as:

$$\begin{cases} G(e_{i+1},\ e_i) = 3, \quad for\ i = 0, 1, \\ G(e_3,\ e_2) = k_o - 10, \\ G(e_4,\ e_3) = 1, \\ G(e_{i+1},\ e_i) = k_o - 1, \quad for\ 4 \le i \le t_e - 2, \\ G(n,\ e_{t_e-1}) = k_o - 1. \end{cases} \tag{2.5}$$

Thus, from (2.5) we can conclude that the pattern is not one-step PD. It can also be verified that none of the gaps $G \ge k_o$ exist in the corresponding patterns in R_1- and R_0-domains.

Case 2, odd $\frac{k_o-1}{2}$:

Consider the error pattern in R_2-domain:

$$\begin{cases} e_0 = 0, \\ e_1 = 4, \\ e_2 = 8, \\ e_3 = k_o, \\ e_4 = k_o + 1, \\ e_i = (i-3)k_o, \quad for\ 5 \le i \le t_e - 1. \end{cases} \tag{2.6}$$

From (2.6), the gaps between any consecutive error position are obtained as:

$$\begin{cases} G(e_{i+1},\ e_i) = 3, \quad for\ i = 0, 1, \\ G(e_3,\ e_2) = k_o - 9, \\ G(e_4,\ e_3) = 0, \\ G(e_{i+1},\ e_i) = k_o - 1, \quad for\ 4 \le i \le t_e - 2, \\ G(n,\ e_{t_e-1}) = k_o - 1. \end{cases} \tag{2.7}$$

Similarly, we can verify that no gaps $G \ge k_o$ exist in the R_0- and R_1-domains. This completes the proof of Theorem 2.2.

Q. E. D.

Fact 2.1: A (53, 7, 21) code is not 3-step PD.

This is because there are several error patterns in this code which do not have gap of length $G \geq 7$.

So, from Theorems 2.1, 2.2 and Fact 2.1, we conclude the following result for the codes with k odd which are not 3-step PD:

Result I: The $(n, k_o, 2t_e + 1)$ codes with $n = (t_e - 1) \cdot k_o - 2(2l + 1)$, for

$$l \geq \frac{k_o - 1}{2} \qquad or \qquad l \geq \frac{t_e - 4}{2}$$

are not 3-step PD.

Now, we will give the results for the codes with k odd which are 3-step PD.

Theorem 2.3 *The* $(n, k_o, 2t_e + 1)$ *codes with* $n = (t_e - 1) \cdot k_o - 2(2l + 1)$, $k_o \geq 2l + 3$, $0 \leq l \leq \frac{t_e - 6}{2}$, *except the code (53, 7, 21), are 3-step* PD.

Proof: This theorem will be proven using the principle of contradiction. Suppose code C is not 3-step PD. Then, consider different types of patterns σ and γ in the R_1-domain, for any error pattern which is not one- or two- step PD in the worst case, are shown in Table 2.1. (Note that the other types of patterns will result in lower gap lengths). Thus, in the R_0-domain, by using Table 2.1, the following relationship can be obtained:

$$\begin{cases} 2N_\sigma + N_\gamma = t_e - 1, \\ t_e + N_\sigma g_\sigma + N_\gamma g_\gamma + g_c \geq k_o(t_e - 1) - 2(2l + 1). \end{cases} \tag{2.8}$$

From equation (2.8), we have

$$N_\sigma \leq \frac{2k_o + 4(2l+1)}{k_o + 1} \leq \frac{6k_o}{k_o + 1} \leq 6 \tag{2.9}$$

in the R_1-domain. Since t_e is even, so N_σ must be odd [33]. Now, we proceed to the cases of $N_\sigma = 1$, $N_\sigma = 3$ and $N_\sigma = 5$ by considering the EPs in the R_2-domain with respect to the EPs in the R_1- and R_0-domains. Let us denote the number of the σ-type patterns of EP in the R_2-domain as N'_σ. Then according to equation (2.9) we can conclude that $N'_\sigma \leq 5$. Thus we have the following three lemmas:

Table 2.1: Gap lengths associated with σ and γ- type patterns (with gap lengths $< k_o$) in R_i- and R_{i+1}-domains

Type of Patterns	Number of Patterns	The type of patterns and gap-lengths in R_{i+1}-domain	The corresponding gap-lengths in R_i-domain
σ	N_σ	$\leq k_o - 2$ o $\overbrace{\text{x} \quad \text{x}}$ o $\leq 2k_o - 1$ $\overbrace{\text{x} \quad \underbrace{o \quad o} \quad \text{x}}$ $\leq k_o - 2$	$g_\sigma \leq \frac{k_o - 3}{2} + k_o - 1$
γ	N_γ	$\leq k_o - 1 \quad \leq k_o - 1$ $\overbrace{o \quad \text{x} \quad o}$ $\leq 2k_o - 1$ $\underbrace{\text{x} \quad o \quad \text{x}}$ $\leq k_o - 1 \quad \leq k_o - 1$	$g_\gamma \leq k_o - 1$
	1	x o 1st odd-value last even-value	$g_c \leq k_o - 1$ (central-gap)

Note: The central gap is referred to as the gap in the R_i-domain, which corresponds to the first odd- and the last even- error positions in the R_{i+1}-domain.

Lemma 2.1 *Suppose that $N_\sigma = 1$. Then the corresponding patterns in the R_2-domain for $N'_\sigma = 1$, 3 or 5, except for the code (57, 7, 21), are 3-step PD.*

Lemma 2.2 *Suppose that $N_\sigma = 3$. Then the corresponding patterns in the R_2-domain for $N'_\sigma = 1$, 3 or 5 are 3-step PD.*

Lemma 2.3 *Suppose that $N_\sigma = 5$. Then the corresponding patterns in the R_2-domain for $N'_\sigma = 1$, 3 or 5, are 3-step PD.*

Proof of Lemma 2.1: We prove this lemma using the principle of contradiction. The proof of the lemma will be divided into the following three cases:

Case 2.1.1, for $N'_\sigma = 1$:

In this case, if for $n = k_o(t_e-1)-2(2l+1)$ the pattern is not decodable, then there should be a collection of gaps of lengths $(k_o - 1 - 2j)$ in R_1-domain, where $1 \leq j \leq l+2$. Now, in the worst case (i.e., when $j = 1$), the following relation should exist in the R_1-domain:

$$(2k_o - 1) + (k_o - 1) + x(k_o - 3) + (t_e - 4 - x)(k_o - 1) + (t_e - 2) = n \tag{2.10}$$

where x is the number of gaps of length $(k_o - 3)$. From (2.10) we have:

$$x = 2l + 1 \tag{2.11}$$

Therefore, in the worst case, the code length n, as shown in Figure 2.1, should satisfy:

$$(4k_o - 1) + (2l + 2)(k_o - 3) + (t_e - 5 - (2l + 2))(k_o - 1) + (t_e - 4) \geq n, \tag{2.12}$$

or

$$(t_e - 1) \cdot k_o - 2(2l + 2) \geq n. \tag{2.13}$$

in R_2-domain for $t_e \geq 10$. The condition in (2.13) does not satisfy the assumptions of Theorem 2.3. For the cases of $t_e \leq 8$, we consider the following options (all the EPs used here make the EPs have the largest spans):

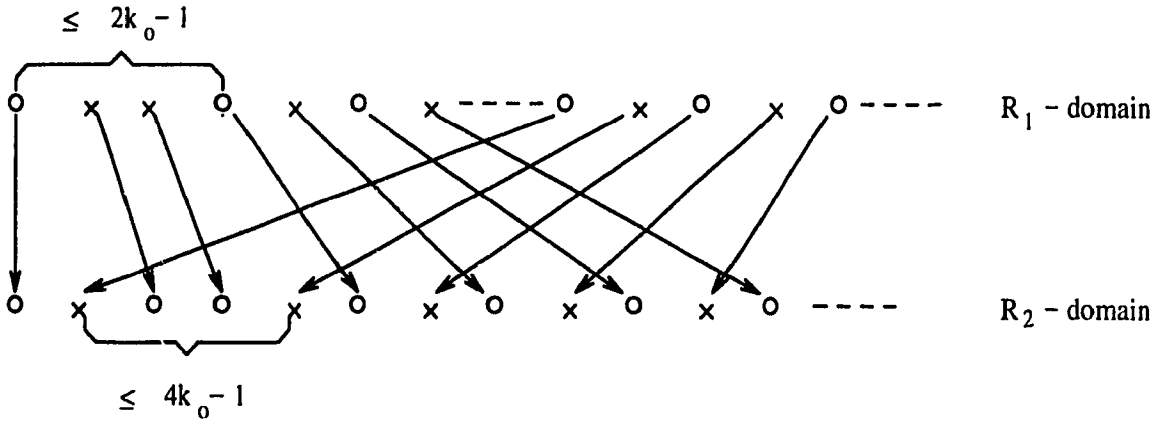


Figure 2.1: A sketch representing the case $N_\sigma = 1$ and $N'_\sigma = 1$ in the worst case.

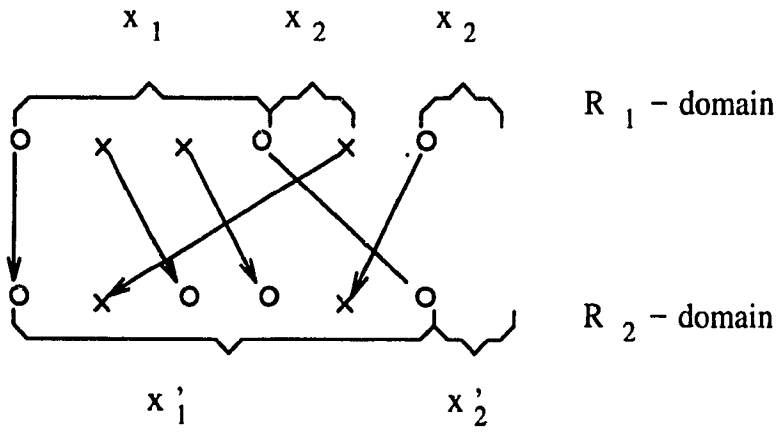


Figure 2.2: The case $t_e = 6$.

Option 2.1.1.1: The case $t_e = 6$

As shown in Figure 2.2, for $x'_1 + x'_2 + 2 = n$ in the R_2-domain, only the following two choices can have:

$$\begin{cases} x'_1 = 4k_o - 1, & x'_2 = k_o - 3; \quad or \\ x'_1 = 4k_o - 3, & x'_2 = k_o - 1. \end{cases} \tag{2.14}$$

In Choice 1, the correspondence of x_2 in R_1-domain, x'_2, will equal to $k_o - 2$, which is impossible since x_2 should be even. Similarly in Choice 2, x'_1 will equal to $2k_o - 2$, which is also impossible since x_1 should be odd.

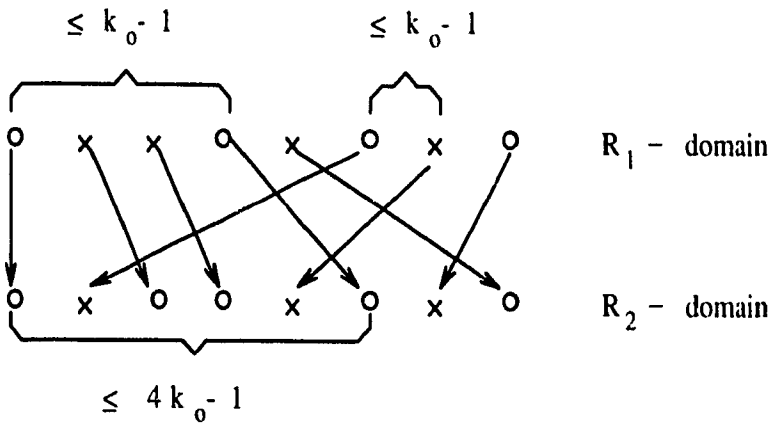


Figure 2.3: The case $t_e = 8$.

Option 2.1.1.2: The case $t_e = 8$

Similarly, we can get the number of gaps of length $(k_o - 3)$ in the worst case in the R_1-domain such that:

$$(2k_o - 1) + (k_o - 1) + x(k_o - 3) + (t_e - 4 - x)(k_o - 1) + (t_e - 2) = n \tag{2.15}$$

From (2.15) we have

$$x = 2l + 1 \tag{2.16}$$

So, in the worst case, the code length n, as shown in Figure 2.3, should satisfy:

$$(4k_o - 1) - 2(2l + 2) + (t_e - 5)(k_o - 1) \geq n, \tag{2.17}$$

or

$$7k_o - 2(2l + 4) \geq n. \tag{2.18}$$

which is a contradiction.

Case 2.1.2, for $N'_\sigma = 3$

We prove this case in two steps: $k_o = 4i - 1$ and $k_o = 4i + 1$.

Option A, for $k_o = 4i - 1$

Similar to the proof of Case 2.2.1, we can get the number of gaps of length $(k_o - 3)$ in the R_1-domain as follows:

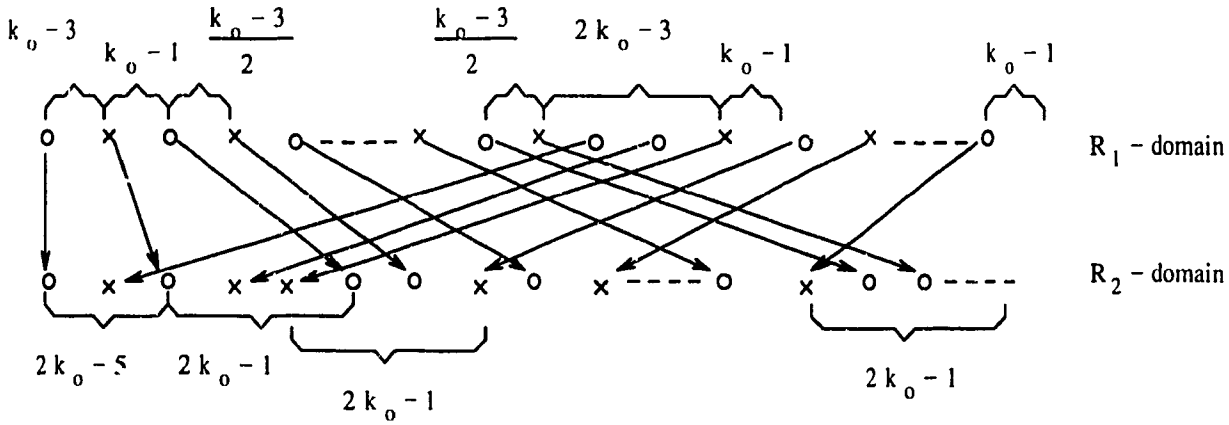


Figure 2.4: A sketch representing the case $N_\sigma = 1$ and $N'_\sigma = 3$ in the worst case $k_o = 4i - 1$.

$$2(k_o - 3) + 3(k_o - 1) + (2k_o - 3) + x(k_o - 3) + (t_e - x - 9)(k_o - 1) + (t_e - 2) = n, \quad (2.19)$$

or

$$x = \frac{4l - k_o - 3}{2}. \quad (2.20)$$

If $k_o \geq 4l - 1$, x is negative. This is impossible. When $k_o \leq 4l - 5$, then $x \geq 1$. Therefore, in the worst case, the code length n, as shown in Figure 2.4 should satisfy

$$3(2k_o - 1) + (2k_o - 5) + (t_e - 10 - (x + 1))(k_o - 1) + (x + 1)(k_o - 3) + (t_e - 7) \geq n, \quad (2.21)$$

or

$$k_o(t_e - 1) - 2(2l + 2) \geq n. \quad (2.22)$$

in R_2-domain for $t_e \geq 14$. The condition in Equation (2.22) does not satisfy the assumptions of the Theorem 2.3.

Option B, for $k_o = 4i + 1$

Similarly, we can get the number of gaps of length $(k_o - 3)$ in R_1-domain as follows:

$$x = \frac{4l - k_o - 9}{2}. \quad (2.23)$$

If $k_o \geq 4l - 7$, x becomes negative, which is impossible. When $k_o \leq 4l - 11$, we have $x \geq 1$, so in R_2-domain the code length n, as shown in Figure 2.5, should satisfy:

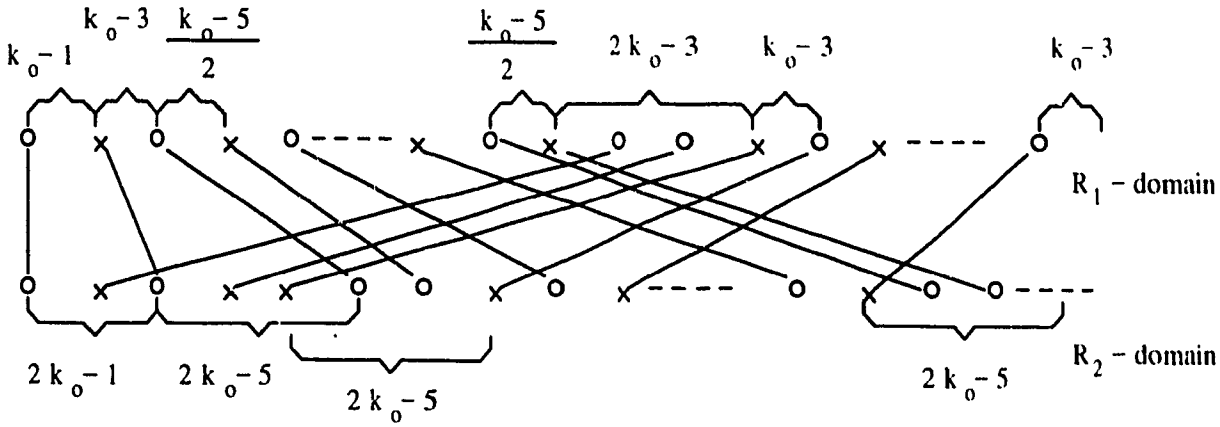


Figure 2.5: A sketch representing the case $N_\sigma = 1$ and $N'_\sigma = 3$ in the worst case $k_o = 4i + 1$.

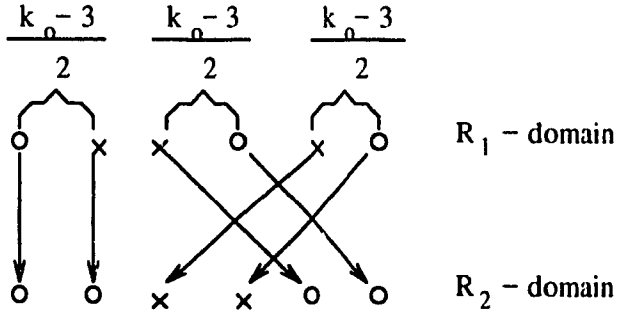


Figure 2.6: The case $t_e = 6$.

$$3(2k_o - 5) + (2k_o - 1) + (t_e - 10)(k_o - 1) - (4l - k_o - 7) + (t_e - 7) \geq n,$$

or

$$k_o(t_e - 1) - 2(2l + 3) \geq n.$$

for $t_e \geq 14$. Which again is a contradiction.

From Options A and B we conclude that Case 2.1.2 for $t_e \geq 14$ is 3-step PD. Next, we continue to derive the result for $t_e \leq 12$.

Option 2.1.2.1, The case $t_e = 6$

In the worst case, the code length n, as shown in Figure 2.6, should satisfy:

$$3(\frac{k_o - 3}{2}) + 2(k_o - 1) + (k_o - 2) + 6 \geq n$$

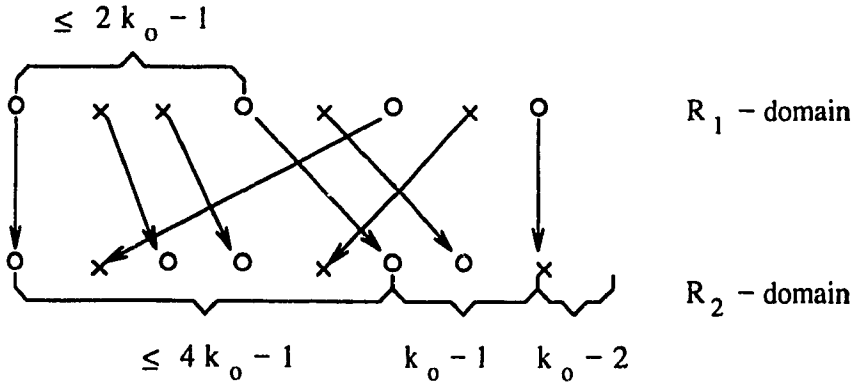


Figure 2.7: The case $t_e = 8$.

in R_1-domain. That is

$$k_o \leq -1$$

which is a contradiction.

Option 2.1.2.2, The case $t_e = 8$

In the worst case, the code length n, as shown in Figure 2.7, should satisfy:

$$(4k_o - 1) + (k_o - 1) + (k_o - 2) + (t_e - 5) \geq n$$

in R_2-domain. That is:

$$k_o \leq 4l + 1$$

As for $t_e = 8$, l only equals to 0 and 1. That is, $k_o \leq 1$ when $l = 0$ and $k_o \leq 5$ when $l = 1$, which is contradiction with respect to assumptions of Theorem 2.3.

Option 2.1.2.3, The case $t_e = 10$

Note that the gap of (o x) type pattern must be even, so when $k_o = 4i - 1$, the corresponding gap of (o o) in R_1-domain should be $\frac{k_o-3}{2}$; and when $k_o = 4i + 1$, should be $\frac{k_o-5}{2}$. When $k_o = 4i - 1$, in the worst case, the code length n, as shown in Figure 2.8, should satisfy:

$$(4k_o - 1) + (k_o - 2) + (2k_o - 1) + (k_o - 1) + (t_e - 6) \geq n,$$

that is:

$$k_o \leq 4l - 1. \tag{2.24}$$

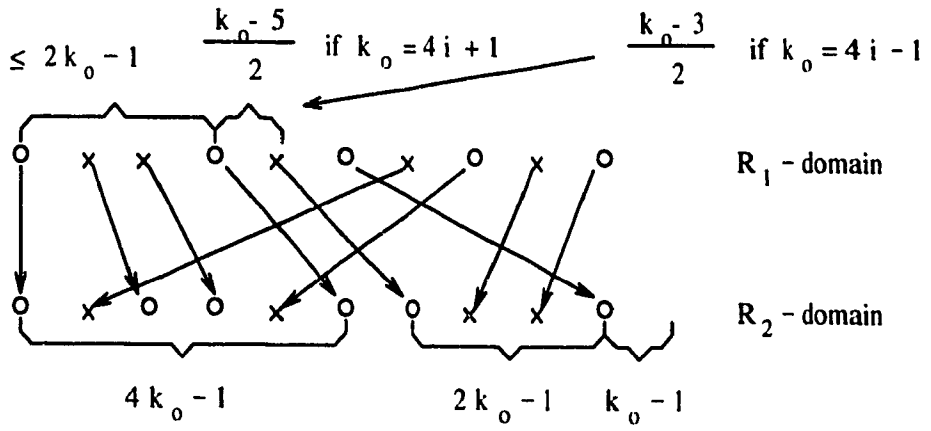


Figure 2.8: The case $t_e = 10$.

As for $t_e = 10$, and according to the assumption of Theorem 2.3, the limit on l is $l \leq 2$. So

$$\begin{cases} k_o \leq -1 & when\ l = 0, \\ k_o \leq 3 & when\ l = 1, \\ k_o \leq 7 & when\ l = 2. \end{cases} \tag{2.25}$$

Only the case $k_o = 7$, $l = 2$ does not contradict the assumption of Theorem 2.3, that is to say, except for the code (53, 7, 21), Option 2.1.2.3 makes a contradiction with $k_o = 4i - 1$.

When $k_o = 4i + 1$, in the worst case, the code length n, as shown in Figure 2.8, should satisfy

$$(4k_o - 1) + (k_o - 4) + (2k_o - 1) + (k_o - 1) + (t_e - 6) \geq n,$$

that is,

$$k_o \leq 4l - 3.$$

which contradicts the assumption of Theorem 2.3 when $l \leq 2$.

Case 2.1.3, for $N'_\sigma = 5$

Assume that there exists an error pattern such that $N_\sigma = 1$ and $N'_\sigma = 5$ which is not 3-step PD. Therefore, for such an EP, in the worst case, the code length n, as shown in Figure 2.9, should satisfy:

$$3(2k_o - 1) + 2(k_o - 2) + (t_e - 11)(k_o - 1) + (t_e - 6) \geq n,$$

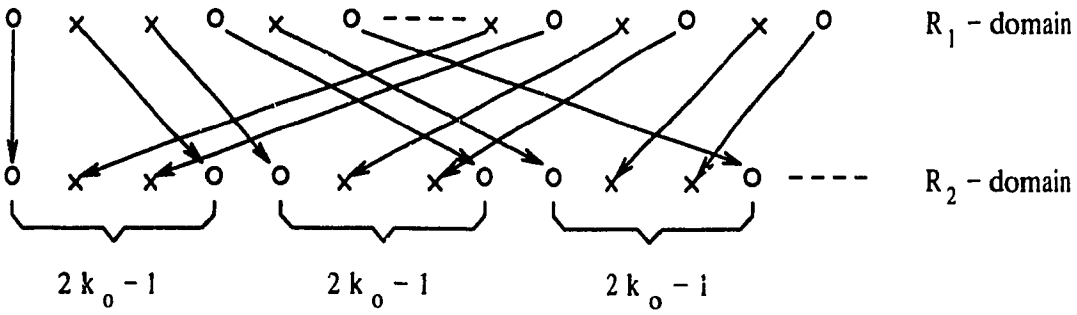


Figure 2.9: A sketch responding the case $N_\sigma = 1$ and $N'_\sigma = 5$ in the worst case.

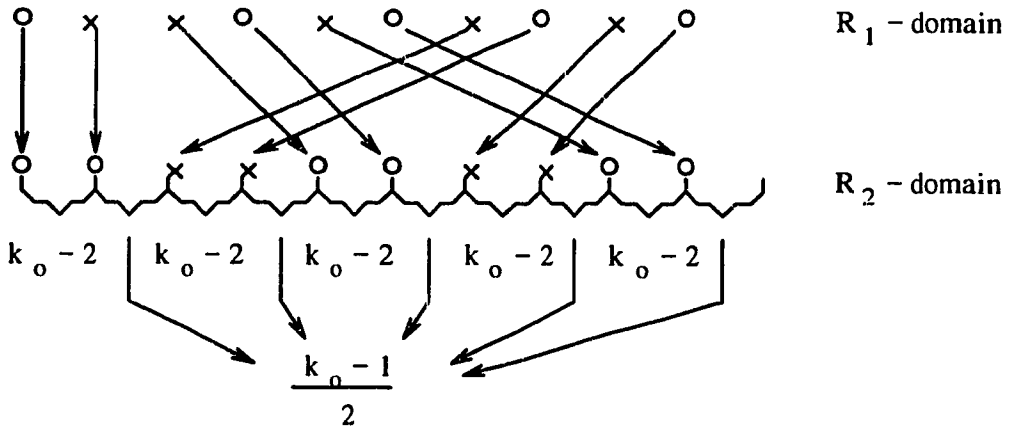


Figure 2.10: The case $t_e = 10$.

or

$$k_o \leq 2l - 1 \tag{2.26}$$

in R_2-domain for $t_e \geq 12$. The condition in (2.26) does not satisfy the assumptions of Theorem 2.3.

For $t_e = 10$ (as $t_e < 10$ is impossible in this case), in the worst case, the code length n, as shown in Figure 2.10, should satisfy:

$$5(k_o - 2) + 5(\frac{k_o - 1}{2}) + t_e \geq n,$$

or

$$k_o \leq \frac{8l - 1}{3}$$

in R_2-domain. Clearly, it contradicts the assumption of Theorem 2.3 when $l \leq 2$. This completes the proof of Lemma 2.1.

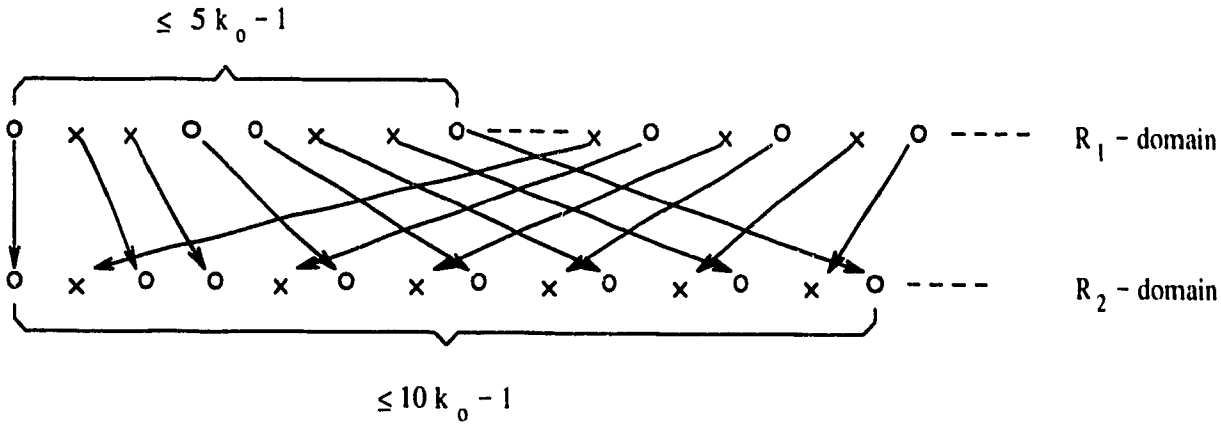


Figure 2.11: A sketch representing the case $N_\sigma = 3$ and $N'_\sigma = 1$ in the worse case.

Proof of Lemma 2.2:

This Lemma will be established through contradiction. Similar to the proof of Lemma 2.1, we consider the following three cases:

Case 2.2.1, for $N'_\sigma = 1$:

In this case, let us assume that there exists an EP such that $N_\delta = 3$ and $N'_\delta = 1$ which is not 3-step PD. Therefore, for such an EP, in the worst case, the code length n, as shown in Figure 2.11, should satisfy:

$$(10k_o - 1) + (t_e - 13)(k_o - 1) + (t_e - 12) \geq n$$

or

$$k_o \leq 2l + 1 \tag{2.27}$$

in R_2-domain for $t_e \geq 14$. The condition of Equation (2.27) contradicts the assumptions of Theorem 2.3. For the cases of $t_e \leq 12$, we consider the follows:

Option 2.2.1.1, The case $t_e = 6$:

In the worst case, the code length n, as shown in Figure 2.12, should satisfy:

$$4k_o - 1 \geq n$$

in R_2-domain. That is to say, $k_o \leq 1$, which is a contradiction.

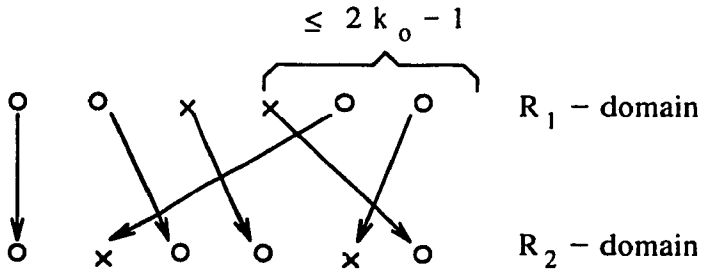


Figure 2.12: The case $t_e = 6$.

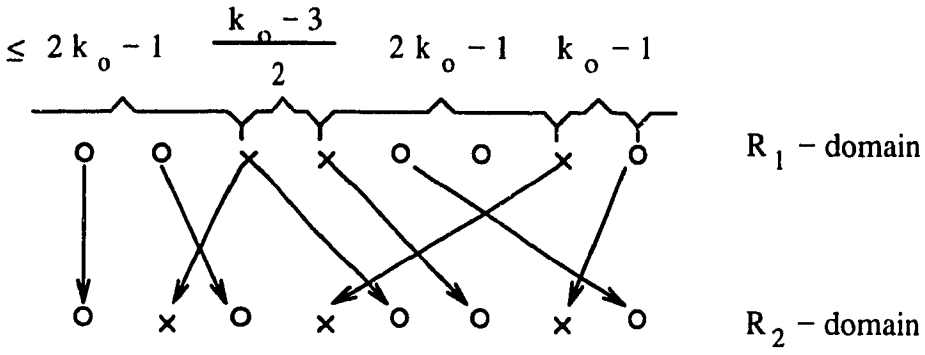


Figure 2.13: The case $t_e = 8$.

Option 2.2.1.2, The case $t_e = 8$:

In the worst case, the code length n, as shown in Figure 2.13, should satisfy:

$$2(k_o - 1) + (k_o - 3)/2 + (k_o - 1) \geq n$$

in R_2-domain. That is

$$k_o \leq \frac{8l - 5}{3}$$

As for $t_e = 8$, according to the assumption of Theorem 2.3, the limit for l is $l \leq 1$. When $l = 0$, then $k_o \leq -1$, which is impossible; when $l = 1$, then $k_o \leq 3$, again it contradicts the assumption that $k_o \geq 2l + 3$.

Option 2.2.1.3, The case $t_e = 10$:

In the worst case, the code length n, as shown in Figure 2.14, should satisfy:

$$\begin{cases} 9k_o - 2 - x \geq n \\ x + (4k_o - 1) + 2(k_o - 1) + (t_e - 6) \geq n \end{cases} \tag{2.28}$$

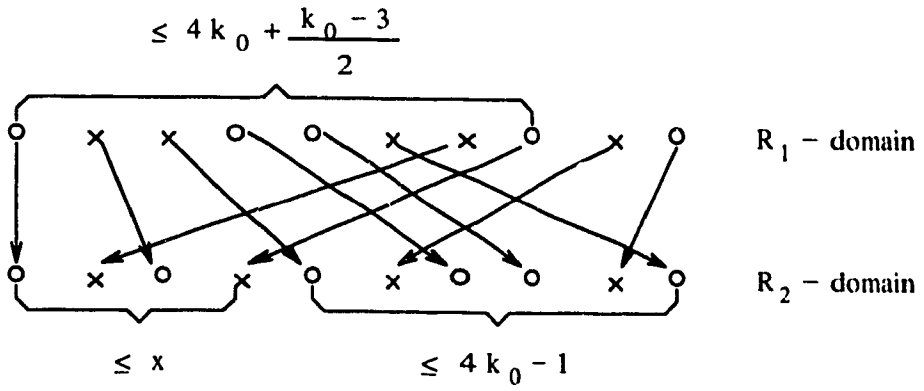


Figure 2.14: The case $t_e = 10$.

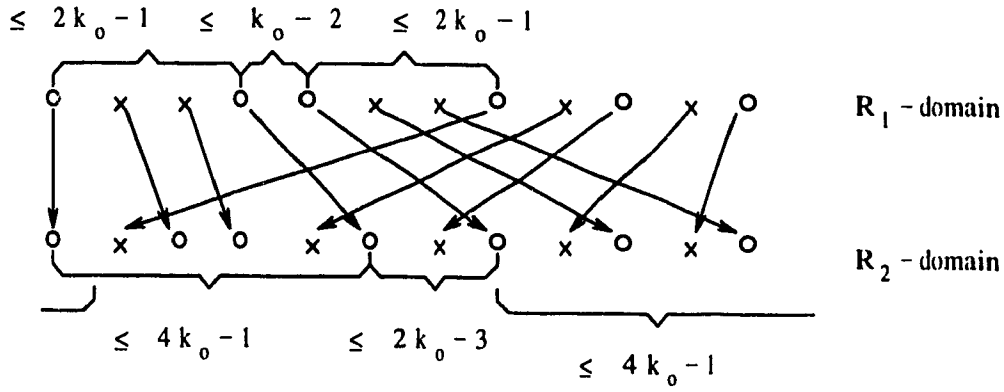


Figure 2.15: The case $t_e = 12$.

in R_2-domain. From (2.28), we have:

$$4l \geq x \geq 3k_o - 2(2l+1) - 1,$$

that is:

$$k_o \leq \frac{8l+3}{3}.$$

As $t_e = 10$, so according to the assumption, $l \leq 2$. When $l = 0$, $k_o \leq 3$; $l = 2$, $k_o \leq 5$. All are contradictions with respect to the assumption that $k_o \geq 2l + 3$.

Option 2.2.1.4, The case $t_e = 12$:

In the worst case, the code length n, as shown in Figure 2.15 should satisfy:

$$2(4k_o - 1) + (2k_o - 3) + 2 - (k_o - 1) \geq n$$

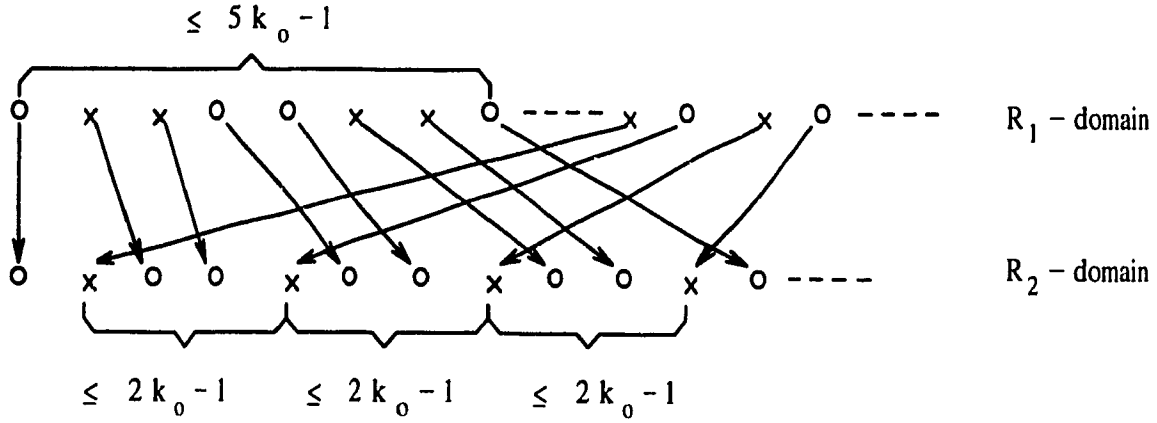


Figure 2.16: A sketch representing the case $N_\sigma = 3$ and $N'_\sigma = 3$ in the worst case.

in R_2-domain. That is:

$$k_o \leq 2l$$

which is a contradiction with the assumption that $k_o \geq 2l + 3$.

Case 2.2.2, for $N'_\sigma = 3$:

For an EP which is not 3-step PD, in the worst case, the code length n, as shown in Figure 2.16, should satisfy:

$$3(2k_o - 1) + (t_e - 9)(k_o - 1) + (t_e - 6) \geq n$$

or

$$k_o \leq 2l + 1$$

in R_2-domain, for $t_e \geq 12$. Which is a contradiction with respect to the assumptions of Theorem 2.3. For the cases of $t_e \leq 10$, we consider as follows:

Option 2.2.2.1, The case $t_e = 6$:

In the worst case, the code length n, as shown in Figure 2.17, should satisfy:

$$3\frac{k_o - 3}{2} + 3(k_o - 1) + 6 \geq n,$$

or

$$k_o \leq 8l + 1 = 1.$$

Which is a contradiction with respect to the assumption of $k_o \geq 2l + 3 = 3$.

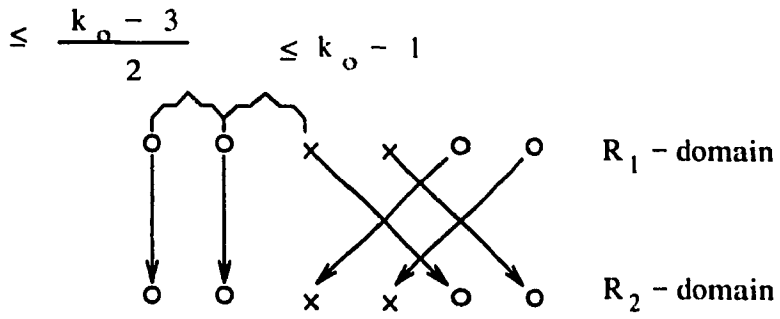


Figure 2.17: The case $t_e = 6$.

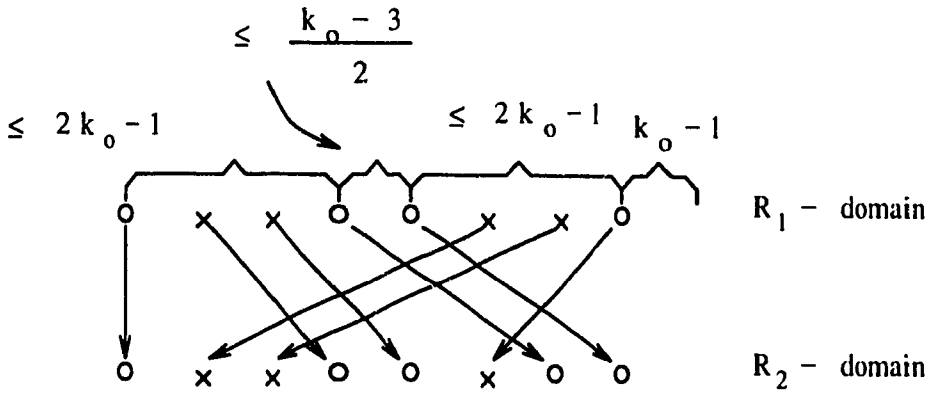


Figure 2.18: The case $t_e = 8$.

Option 2.2.2.2, The case $t_e = 8$:

In the worst case, the code length n, as shown in Figure 2.18, should satisfy:

$$(t_e - 4) + 2(2k_o - 1) + (k_o - 3)/2 + (k_o - 1) \geq n,$$

or

$$k_o \leq \frac{8l + 3}{3}.$$

As $t_e = 8$, so when $l = 0$, $k_o \leq 1$; when $l = 1$, $k_o \leq 3$; which contradict the assumption of Theorem 2.3.

Option 2.2.2.3 The case $t_e = 10$:

In the worst case the code length n, as shown in Figure 2.19 should satisfy:

$$3(2k_o - 1) + (k_o - 1) + (t_e - 6) \geq n$$

in R_2-domain. That is:

$$k_o \leq 2l + 1$$

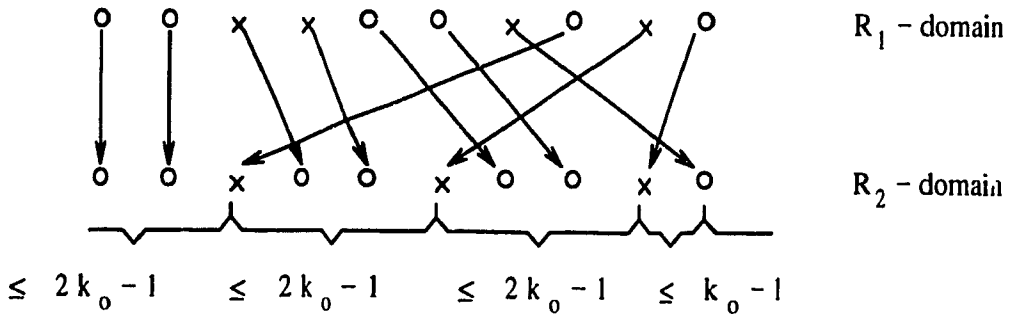


Figure 2.19: The case $t_e = 10$.

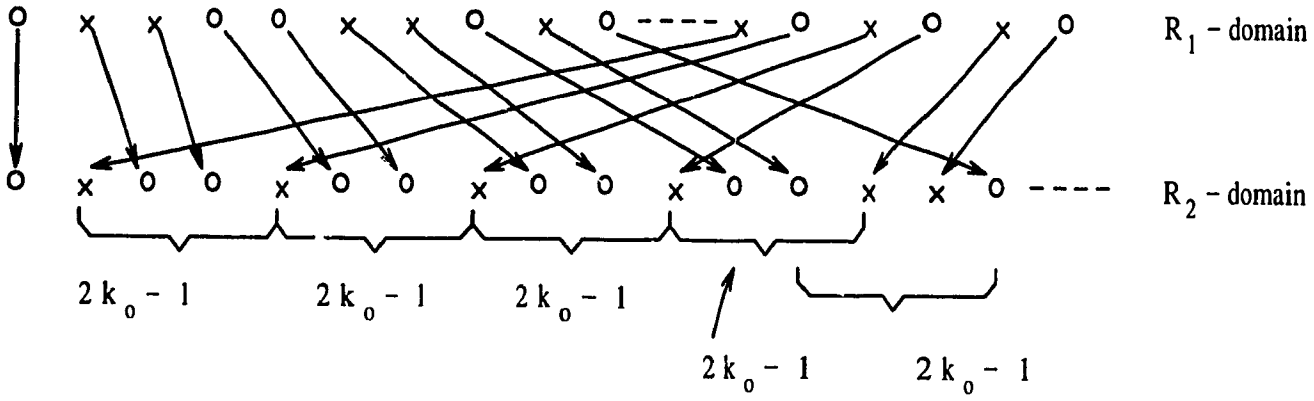


Figure 2.20: A sketch representing the case $N_\sigma = 3$ and $N'_\sigma = 5$ in the worst case.

which contradict the assumptions of Theorem 2.3.

Case 2.2.3, for $N'_\sigma = 5$:

For an EP which is not 3-step PD, in the worst case, the code length n, as shown in Figure 2.20, should satisfy:

$$5(2k_o - 1) + (t_e - 14)(k_o - 1) + (t_e - 10) \geq n,$$

or

$$k_o \leq \frac{4l + 1}{3}$$

in R_2-domain for $t_e \geq 16$. Which is a contradiction with respect to the assumption of Theorem 2.3. For the cases of $t_e \leq 14$, we consider as follows:

Option 2.2.3.1, The case $t_e = 10$:

In the worst case, the code length n, as shown in Figure 2.21, should satisfy:

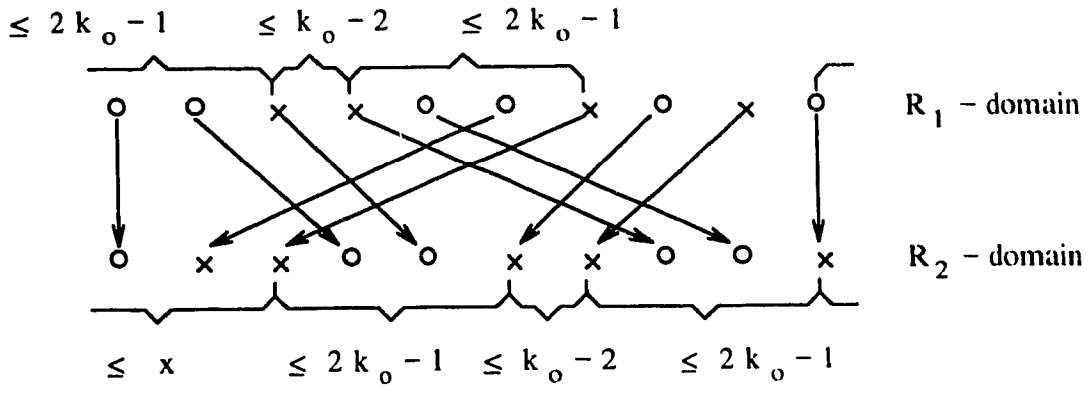


Figure 2.21: The case $t_e = 10$.

$$\begin{cases} 2(2((2k_o - 1) + (k_o - 2) + 2) + 1) - 1 - x \geq n, \\ x + 2(2k_o - 1) + (k_o - 2) + 4 \geq n. \end{cases} \tag{2.29}$$

From Equation (2.29), we have

$$4k_o - 2(2l + 1) \leq x \leq k_o + 2(2l + 1) - 3,$$

or

$$k_o \leq \frac{4(2l + 1) - 3}{3}.$$

Since $t_e = 10$, then $l \leq 2$. When $l = 2$, $k_o \leq 5$. Clearly, it contradicts the assumption that $k_o \geq 2l + 3$.

Option 2.2.3.2, The case $t_e = 12$:

In the worst case, the code length n, as shown in Figure 2.22, should satisfy:

$$3(2k_o - 1) + 2(k_o - 2) + (k_o - 1) + (t_e - 6) \geq n,$$

or

$$k_o \leq 2l - 1$$

in R_2-domain. It contradicts the assumption that $k_o \geq 2l + 3$.

Option 2.2.3.3, The case $t_e = 14$:

Similarly, the code length of n, as shown in Figure 2.23, should satisfy:

$$3(2k_o - 1) + 2(k_o - 2) + 3(k_o - 1) + (t_e - 6) \geq n,$$

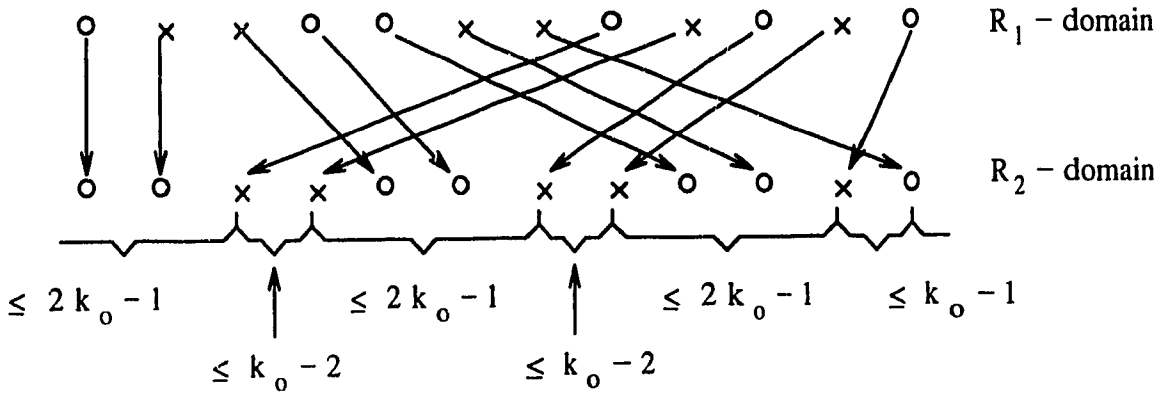


Figure 2.22: The case $t_e = 12$.

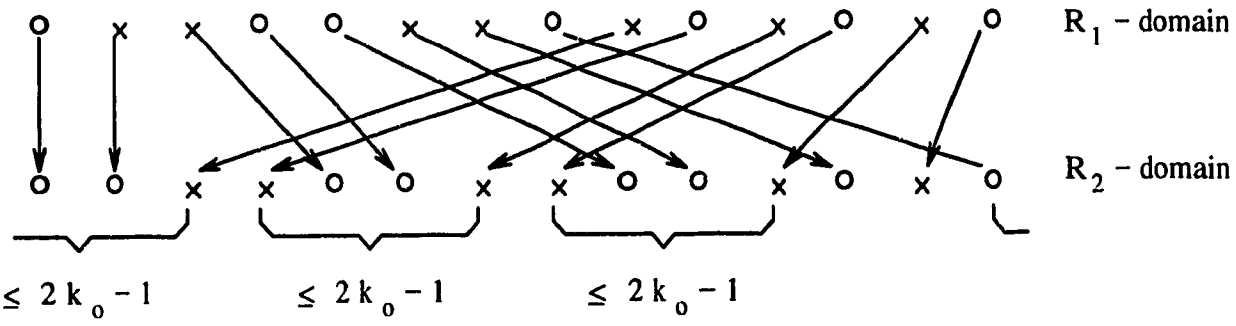


Figure 2.23: The case $t_e = 14$.

or

$$k_o \leq 2l$$

in R_2-domain. Again it is a contradiction with respect to the assumption that $k_o \geq 2l+3$. This completes the proof of Lemma 2.2.

In the same way, we get Lemma 2.3.

The proofs of Lemma 2.1, 2.2 and 2.3 complete the proof of Theorem 2.3.

Q. E. D.

Next, we present the result for the case when k is even.

2.3.2 When k is even

Theorems 2.1, 2.2, and 2.3 give tight lower bounds on the code length n for the codes with k odd. Now, by the following theorems we present the bounds for the codes with k being even.

Theorem 2.4 *The* $(n,\ k_e,\ 2t_e+1)$ *codes with* $n = (t_e-1)\cdot(k_e-1) - 2(2l+1)$, *for* $l = \frac{t_e-4}{2}$, *or* $n = (t_e-1)\cdot(k_e-3)+4$, *are not 3-step PD.*

Proof of Theorem 2.4: Let $k_e = k_o + 1$, then

$$\begin{aligned} n &= (t_e-1)(k_e-1) - 2(2l+1) \\ &= (t_e-1)k_o - 2(2l+1) \end{aligned}$$

According to Theorem 2.1, $(n,\ k_o,\ 2t_e+1)$ codes with $n = (t_e-1)k_o - 2(2l+1)$, for the case $l = \frac{t_e-4}{2}$, are not 3-step PD. Therefore, it becomes obvious that the $(n,\ k_e,\ 2t_e+1)$ codes cannot be 3-step PD either.

Q. E. D.

Theorem 2.5 *The* $(n, k_e, 2t_e+1)$ *codes with* $n = (t_e-1)(k_e-1)-2(2l+1)$, *for* $k_e \leq 2l+4$, *are not 3-step PD.*

Proof of Theorem 2.5: We will divide the proof into two parts: when $t_e/2$ is even and when $t_e/2$ is odd.

Case 1, $t_e/2$ even:

Based on the proof of Theorem 2.1, it is sufficient to prove the theorem by finding an error pattern of weight t_e that does not have any gaps of length $G \leq k_e$ in the R_0-, R_1- and R_2-domains.

Now consider the error pattern in R_2-domain:

$$\begin{cases} e_0 = 0, \\ e_1 = 4, \\ e_2 = 8, \\ e_i = i(k_e-1) - 2k_e + 6, \quad for\ 3 \leq i \leq t_e - 3, \\ e_{t_e-2} = \begin{cases} t_e(k_e-1) - 5k_e + 12, & for\ k_e/2\ being\ even, \\ t_e(k_e-1) - 5k_e + 14, & for\ k_e/2\ being\ odd, \end{cases} \\ e_{t_e-1} = t_e(k_e-1) - 4k_e + 12. \end{cases} \tag{2.30}$$

From (2.30), the gaps between any consecutive error position pairs in the R_2-domain are:

$$\begin{cases} G(e_{i+1}, e_i) = 3, \quad for\ i = 0, 1, \\ G(e_3, e_2) = k_e - 6, \\ G(e_{i+1}, e_i) = k_e - 2, \quad for\ 3 \leq i \leq t_e - 4, \\ G(e_{t_e-2}, e_{t_e-3}) = \begin{cases} 2 & for\ k_e/2\ being\ even, \\ 4 & for\ k_e/2\ being\ odd, \end{cases} \\ G(e_{t_e-1}, e_{t_e-2}) = \begin{cases} k_e - 1, & for\ k_e/2\ being\ even, \\ k_e - 3, & for\ k_e/2\ being\ odd, \end{cases} \\ G(n, e_{t_e-1}) = k_e - 6. \end{cases} \tag{2.31}$$

Thus, when $k_e \geq 6$ ($l \geq 1$), no gaps are greater then $k_e - 1$ in the R_2-domain. It can be verified that the corresponding patterns in R_1- and R_0-domains also have no gaps $G \geq k_e$

exist. For the case $k_e = 4$, we can also find EPs which have no gaps $G \geq k_e - 1$ in the R_2-, R_1- and R_0-domains.

Case 2, odd $t_e/2$:

Similarly, first consider an error pattern in the R_2-domain:

$$\begin{cases} e_0 = 0, \\ e_1 = 4, \\ e_2 = 8, \\ e_i = i(k_e - 1) - 2k_e + 6, \quad for\ 3 \leq i \leq t_e - 3, \\ e_{t_e-2} = t_e(k_e - 1) - 4k_e + 6, \\ e_{t_e-1} = t_e(k_e - 1) - 4k_e + 12. \end{cases} \tag{2.32}$$

From (2.32), the gaps are:

$$\begin{cases} G(e_{i+1},\ e_i) = 3, \quad for\ i = 0, 1, \\ G(e_3,\ e_2) = k_e - 6, \\ G(e_{i+1},\ e_i) = k_e - 2, \quad for\ 3 \leq i \leq t_e - 4, \\ G(e_{t_e-2},\ et_e - 3) = k_e - 4, \\ G(e_{t_e-1},\ et_e - 2) = 5, \\ G(n,\ e_{t_e-1}) = k_e - 6. \end{cases} \tag{2.33}$$

So, when $k_e \geq 6$, no gaps are greater then $k_e - 1$ in the R_2-domain. It can be verified that there are also no gaps $G \geq k_e - 1$ exist in the corresponding EPs in either the R_1- or the R_0-domains. In the case $k_e = 4$ ($l = 0$), it can also be proven that for some EPs there are no gaps $G \geq k_e - 1$ exist in the R_0-, R_1- and R_2-domains.

This completes the proof of Theorem 2.5.

Q. E. D.

Fact 2.2: The following $(n,\ k_e,\ t_e)$ codes are not 3-step PD:

1. $l = 0, \quad t_e = 6, \quad k_e = 6,\ 8,\ 10,\ 14,$

   $8 \leq t_e \leq 10, \quad k_e = 6;$

2. $l = 1, \quad 8 \leq t_e \leq 10, \quad 8 \leq k_e \leq 10;$

3. $l = 2, \quad t_e = 12, \quad k_e = 10;$

4. $l = 3, \quad t_e = 12, \quad k_e = 12.$

This is because there are several error patterns in these codes which do not have any gaps of length $G \geq k_o$ in either the R_o-, R_1-, and R_2-domains.

So, from Theorems 2.4 and 2.5, and Fact 2.2, for the codes with k even which are not 3-step PD we conclude the following:

Result II: The $(n, k_e, 2t_e+1)$ codes with $n = (t_e-1)(k_e-1)-2(2l+1)$, for $l \geq \frac{k_e-4}{2}$ or $l \geq \frac{t_e-4}{2}$, and the following codes are not 3-step PD:

1. $l = 0, \quad t_e = 6, \quad k_e = 6, 8, 10, 14,$
   $8 \leq t_e \leq 10, \quad k_e = 6;$

2. $l = 1, \quad 8 \leq t_e \leq 10, \quad 8 \leq k_e \leq 10;$

3. $l = 2, \quad t_e = 12, \quad k_e = 10;$

4. $l = 3, \quad t_e = 12, \quad k_e = 12.$

Next, we will present the result for codes with k even which are 3-step PD.

Theorem 2.6 *The* $(n, k_e, 2t_e+1)$ *codes with* $n = (t_e-1)(k_e-1)-2(2l+1)$, *for* $k_e \geq 2l+6$, $0 \leq l \leq \frac{t_e-6}{2}$, *are 3-step* PD, *except for the following cases:*

1. $l = 0, \quad t_e = 6, \quad k_e = 6, 8, 10, 14,$
   $8 \leq t_e \leq 10, \quad k_e = 6;$

2. $l = 1, \quad 8 \leq t_e \leq 10, \quad 8 \leq k_e \leq 10;$

3. $l = 2, \quad t_e = 12, \quad k_e = 10;$

4. $l = 3, \quad t_e = 12, \quad k_e = 12.$

Proof of Theorem 2.6: This theorem will be established by the principle of contradiction. Let us assume that the $(n, k_e, 2t_e + 1)$ codes with $n = (t_e - 1)(k_e - 1) - 2(2l + 1)$ and $k_e = 2l + 6$, are not 3-step PD. Moreover, let us assume $\{e_i\}$ is an EP in R_0-domain which is not 1-step, 2-step or 3-step PD. With reference to (2.8), for the different types of patterns γ, σ in the EPs given in Table 2.2, in the worst case the following relationships should hold:

$$\begin{cases} 2N_\sigma + N_\gamma = t_e - 1, \\ t_e + N_\sigma g_\sigma + N_\gamma g_\gamma + g_e \geq (t_e - 1)(k_e - 1) - 2(2l + 1). \end{cases} \tag{2.34}$$

where $g_\sigma \leq (k_e - 1) + \frac{k_e - 2}{2}$; $g_\gamma \leq k_e - 2$; $g_e \leq k_e - 1$.

From (2.34), we have

$$N_\sigma \leq \frac{4(2l + 1) + 2k_e}{k_e - 4} \leq \frac{6k_e - 20}{k_e - 4} \tag{2.35}$$

in R_1-domain. Since t_e is even, so N_σ must be odd [9]. Consequently, from (2.35), we get $N_\sigma \leq 7$. Thus we have the following four lemmas:

Lemma 2.4 *Suppose that $N_\sigma = 1$. Then the corresponding patterns in the R_2-domain for $N'_\sigma = 1, 3, 5$ or 7, except for the following codes:*

1. $l = 0, \quad t_e = 6, \quad 6 \leq k_e \leq 10,$

   $8 \leq t_e \leq 10, \quad k_e = 6;$

2. $l = 1, \quad t_e = 8, \quad 8 \leq k_e \leq 10,$

   $t_e = 10, \quad k_e = 8;$

3. $l = 2, \quad t_e = 10, \quad k_e = 10.$

are 3-step PD.

Lemma 2.5 *Suppose that $N_\sigma = 3$. Then the corresponding patterns in the R_2-domain for $N'_\sigma = 1, 3, 5$ or 7, except for the codes:*

1. $l = 0, \quad t_e = 6, \quad k_e = 6, 8, 10, 14,$

   $8 \leq t_e \leq 10, \; k_e = 6;$

Table 2.2: Gap lengths associated with σ and γ- type patterns (with gap lengths $< k_e$) in R_i- and R_{i+1}-domains

Type of Patterns	Number of Patterns	The type of patterns and gap-lengths in R_{i+1}-domain	The corresponding gap-lengths in R_i-domain
σ	N_σ	$\leq k_e - 1$ o x x o $\leq 2k_e - 1$ x o o x $\leq k_e - 1$	$g_\sigma \leq \frac{k_e-2}{2} + k_e - 1$
γ	N_γ	$\leq k_e - 2 \quad \leq k_e - 2$ o x o $\leq 2k_e - 3$ x o x $\leq k_e - 2 \quad \leq k_e - 2$	$g_\gamma \leq k_e - 2$
	1	x o 1st odd-value last even-value	$g_c \leq k_e - 2$ (central-gap)
Note: The central gap is referred to as the gap in the R_i-domain, which corresponds to the first odd- and the last even- error positions in the R_{i+1}-domain.			

2. $l = 1, \quad t_e = 8, \quad 8 \leq k_e \leq 10.$

$t_e = 10, \quad k_e = 8;$

3. $l = 2, \quad t_e = 10, \quad k_e = 10.$

are 3-step PD.

Lemma 2.6 *Suppose that $N_\sigma = 5$. Then the corresponding patterns in the R_2-domain for $N'_\sigma = 1, 3, 5$ or 7, except for the codes:*

1. $l = 0, \quad t_e = 10, \quad k_e = 6;$

2. $l = 1, \quad t_e = 10, \quad 8 \leq k_e \leq 10;$

3. $l = 2, \quad t_e = 12, \quad k_e = 10;$

4. $l = 3, \quad t_e = 12, \quad k_e = 12.$

are 3-step PD.

Lemma 2.7 *Suppose that $N_\sigma = 7$. Then the corresponding patterns in the R_2-domain for $N'_\sigma = 1, 3, 5$ or 7, are 3-step PD.*

Proof of Lemma 2.4: We prove this lemma using the principle of contradiction. The proof of the lemma will be divided into the following four cases:

Case 2.4.1, for $N'_\sigma = 1$:

In this case, there should be a number x, of gaps of lengths $(k_e - 2 - 2j)$ for $j \geq 1$. In the worst case (when $j = 1$), consider the EP in Figure 2.24 (All the EPs used here make the EPs have longest spans.), the relationship:

$$(2k_e - 3) + (k_e - 2) + x(k_e - 4) + (t_e - 4 - x)(k_e - 2) + (t_e - 2) = n \qquad (2.36)$$

should exist in the R_1-domain, where x is the number of gaps of length $(k_e - 4)$. From (2.36), we have:

$$x = 2l + 1$$

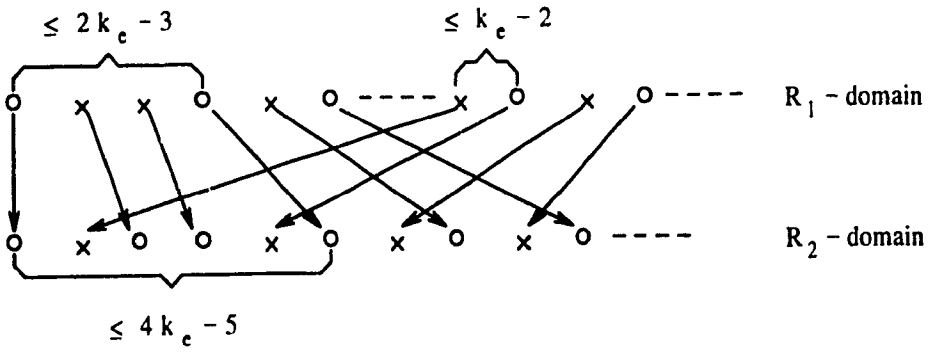


Figure 2.24: A sketch representing the case $N_\sigma = 1$ and $N'_\sigma = 1$ in the worst case.

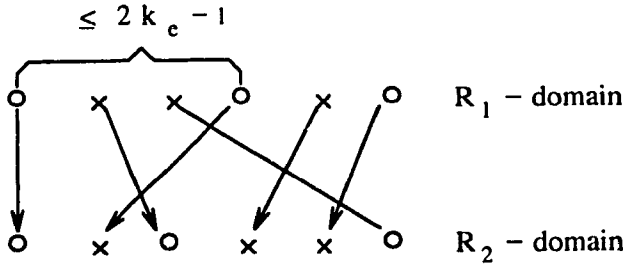


Figure 2.25: The case $t_e = 6$.

Therefore, in the worst case, the code length n, as shown in Figure 2.24, should satisfy:

$$(4k_e - 5) + 2\lceil\frac{x}{2}\rceil(k_e - 4) + (t_e - 5 - \lceil\frac{x}{2}\rceil)(k_e - 2) + (t_e - 4) \geq n \qquad (2.37)$$

in R_2-domain for $t_e \geq 10$. By substituting for $2\lceil\frac{x}{2}\rceil = 2l + 2$ in (2.36), we have

$$(t_e - 1)(k_e - 1) - 2(2l + 2) \geq n$$

which is a contradiction with respect to the assumptions of Theorem 2.6. For $t_e \leq 8$, consider the following cases:

Option 2.4.1.1, The case $t_e = 6$:

In the worst case, the code length n, as shown in Figure 2.25, should satisfy:

$$4k_e - 1 \geq n \qquad (2.38)$$

in the R_2-domain. Equation (2.38) contradict the assumptions of Theorem 2.6 except for the code (23, 6, 13).

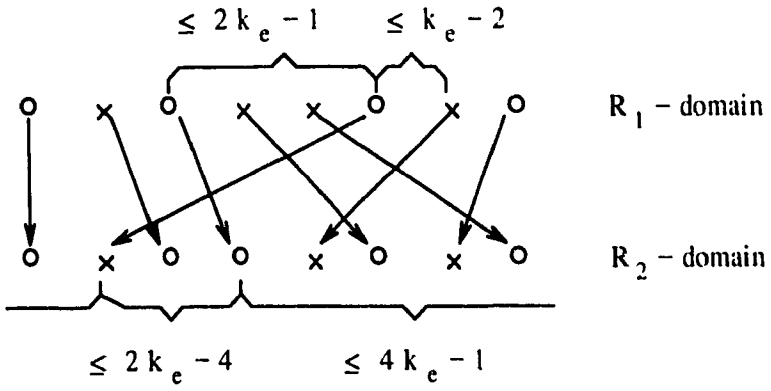


Figure 2.26: The case $t_e = 8$.

Option 2.4.1.2, The case $t_e = 8$:

In the worst case, the code length n, as shown in Figure 2.26, should satisfy:

$$(4k_e - 1) + (2k_e - 4) + (t_e - 6) \geq n \tag{2.39}$$

in the R_2-domain. Equation (2.39) again is a contradiction, except for the codes (33, 6, 17), (43, 8, 17), (57, 10, 17). So except the codes (23, 6, 13), (33, 6, 17), (43, 8, 17), (57, 10, 17), the Case 2.4.1 is three-step PD.

Case 2.4.2, for $N'_\sigma = 3$:

We divide this case into two parts: Case A, $k_e = 4i$; Case B, $k_e = 4i + 2$.

Case A:, For $k_e = 4i$:

Similar to the Case 2.4.1, in the worst case (when $j = 1$), and considering the EP in Figure 2.27, the following relationship

$$4(k_e - 2) + (k_e - 4) + (2k_3 - 3) + x(k_e - 4) + (t_e - x - 9)(k_e - 2) + (t_e - 2) = n \tag{2.40}$$

should exist in R_1-domain. From Equation (2.40), we have:

$$x = \frac{2(2l + 1) - k_e}{2}$$

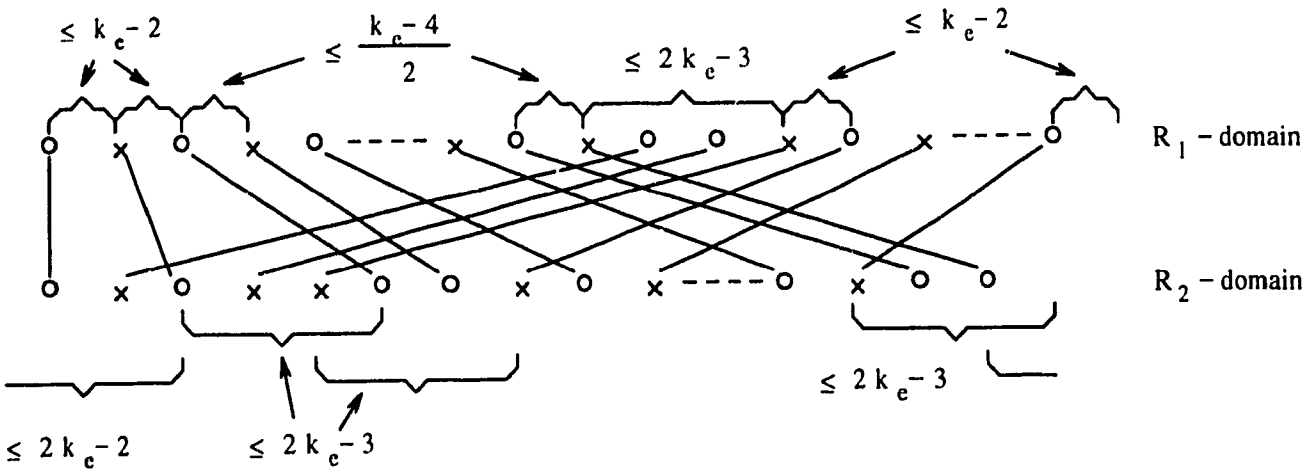


Figure 2.27: A sketch representing the case $N_\sigma = 1$ and $N'_\sigma = 3$ in the worst case.

If $k_e \geq 4l + 4$, x is negative. This is impossible. When $k_e \leq 4l$, $x \geq 1$ is an odd number. For the corresponding EP in the R_2-domain, the total code length n, should satisfy:

$$(2k_e - 2) + 3(2k_e - 3) + 2\lceil\frac{x}{2}\rceil(k_e - 4) + (t_e - 10 - 2\lceil\frac{x}{2}\rceil)(k_e - 2) + (t_e - 8) \geq n \quad (2.41)$$

By substituting

$$2\lceil\frac{x}{2}\rceil = \frac{4(l+1) - k_e}{2}$$

in (2.41), we get

$$(t_e - 1)(k_e - 1) - 2(2l + 2) \geq n$$

which is a contradiction.

Case B:, For $k_e = 4i + 2$:

Similar to the Case A, consider the EP in Figure 2.28, we get

$$x = \frac{2(2l+1) - k_e}{2} + 1$$

in the R_1-domain. If $k_e > 4l + 2$, x is negative. When $k_e \leq 4l + 2$, we have $x \geq 1$ is an odd number. So in the R_2-domain, we have the same relation as Equation (2.41). By substituting

$$2\lceil\frac{x}{2}\rceil = \frac{2(2l+1) - k_e}{2} + 2$$

in (2.41), we get:

$$(t_e - 1)(k_e - 1) - 2(2l + 3) \geq n$$

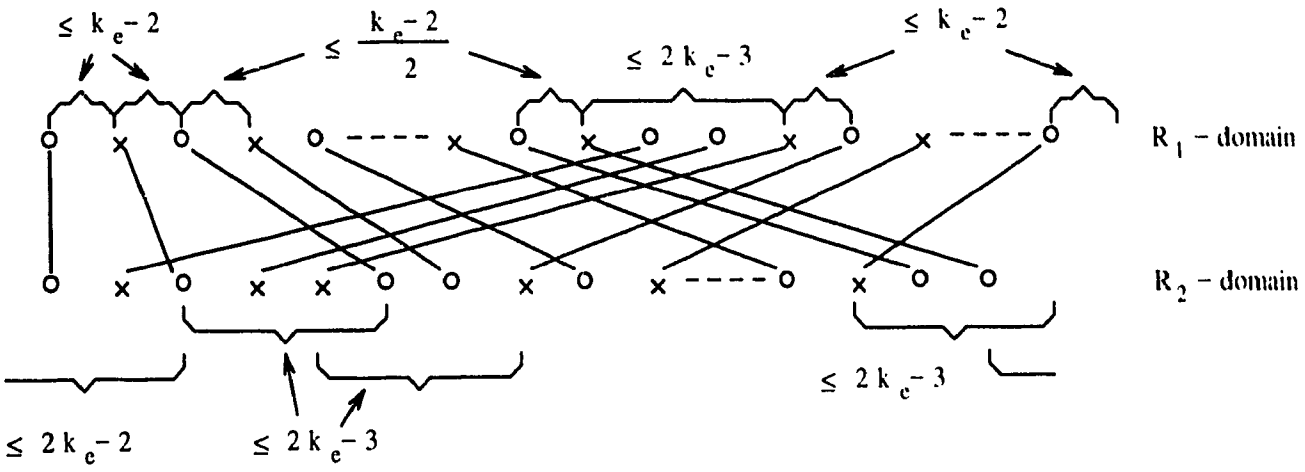


Figure 2.28: A sketch representing the case $N_\sigma = 1$ and $N'_\sigma = 3$ in the worst case.

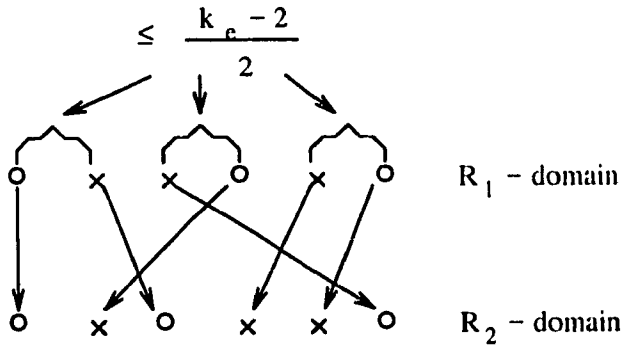


Figure 2.29: The case $t_e = 6$.

which again is a contradiction. Note that Figure 2.27 and 2.28 only exist for $t_e \geq 12$, so we should consider the cases $t_e = 6,\ 8,\ 10$, separately.

Option 2.4.2.1, The case $t_e = 6$:

In the worst case, the code length n, as shown in Figure 2.29, should satisfy:

$$3(\frac{k_e - 2}{2}) + (k_e - 1) + 2(k_e - 2) + 6 \geq n,$$

or

$$k_e \leq 10.$$

in the R_1-domain. Which is a contradiction when $k_e \geq 12$.

Option 2.4.2.2, The case $t_e = 8$:

In the worst case, the code length n, as shown in Figure 2.30, should satisfy:

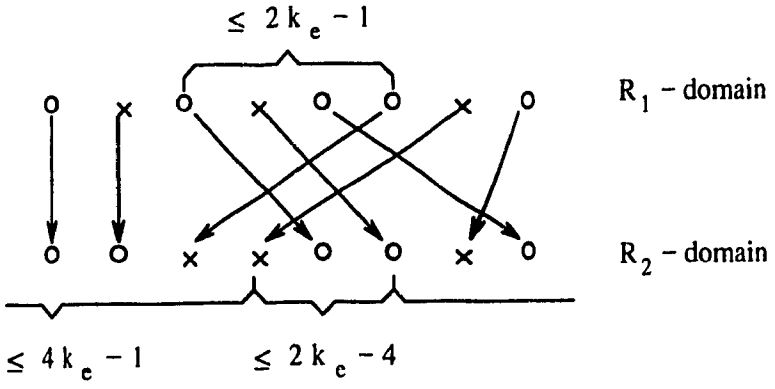


Figure 2.30: The case $t_e = 8$.

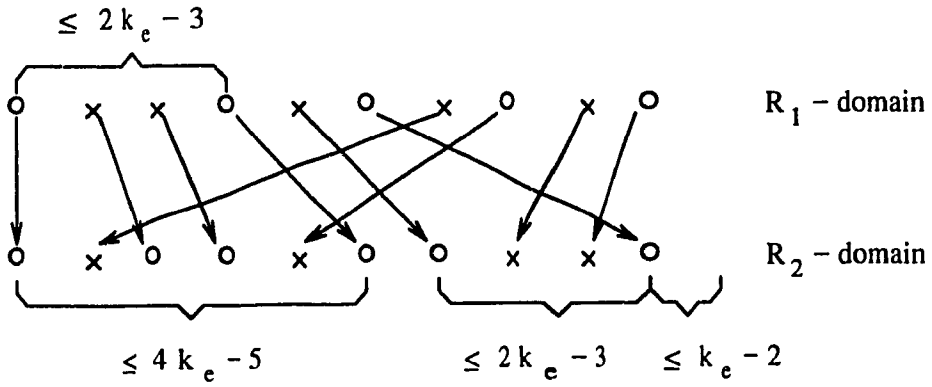


Figure 2.31: The case $t_e = 10$.

$$(4k_e - 1) + (2k_e - 4) + 2 \geq n$$

This is a contradiction for $k_e \geq 12$.

Option 2.4.2.3, The case $t_e = 10$:

In the worst case, the relation as shown in Figure 2.31 should exist. We consider this option with $k_e = 4i$ and $k_e = 4i + 2$ separately. Suppose $k_e = 4i$, then according to Figure 2.31, the code length n should satisfy:

$$(4k_e - 5) + (k_e - 3) + (2k_e - 3) + (k_e - 2) + (t_e - 6) \geq n$$

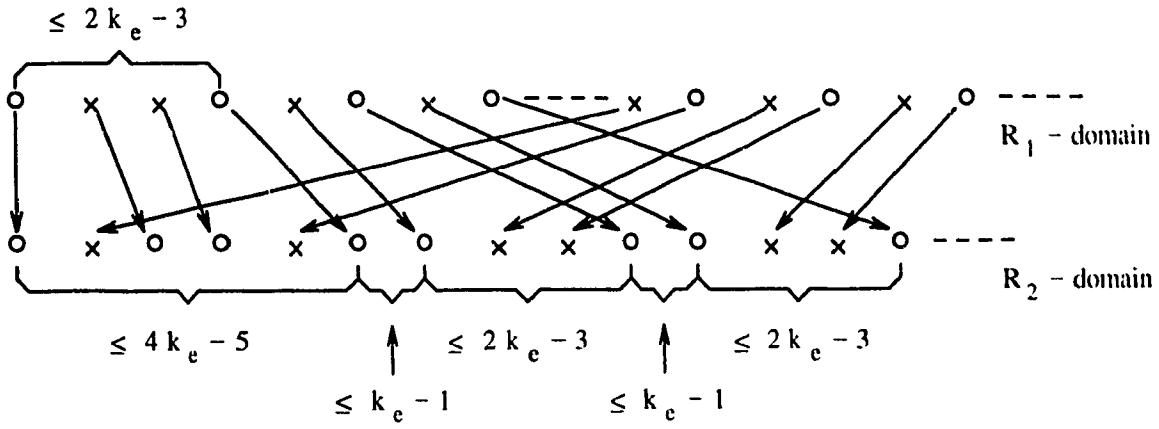


Figure 2.32: A sketch representing the case $N_\sigma = 1$ and $N'_\sigma = 5$ in the worst case.

in the R_2-domain. It is a contradiction. When $k_e = 4l + 2$, then according to Figure 2.31, the code lenght n should satisfy:

$$(4k_e - 5) + (k_e - 1) + (2k_e - 3) + (k_e - 2) + (t_e - 6) \geq n$$

in the R_2-domain. Again it is a contradiction.

Case 2.4.3, For $N'_\sigma = 5$:

In the worst case, the code length n, as shown in Figure 2.32, should satisfy:

$$(4k_e - 5) + 2(k_e - 1) + 2(2k_e - 3) + (t_e - 13)(k_e - 2) + (t_e - 8) \geq n, \tag{2.42}$$

or

$$k_e \leq 2l + 2 \tag{2.43}$$

in the R_2-domain, which is a contradiction with respect to the assumption $k_e \geq 2l + 6$. Note that (2.42) and (2.43) are satisfied only for $t_e \geq 14$, so we continue to consider the cases of $t_e \leq 12$.

Option 2.4.3.1, The case $t_e = 10$:

In the worst case, the code length n, as shown in Figure 2.33, should satisfy:

$$(4k_e - 1) + (2k_e - 3) + (k_e - 1) + \frac{k_e - 2}{2} + (t_e - 6) \geq n$$

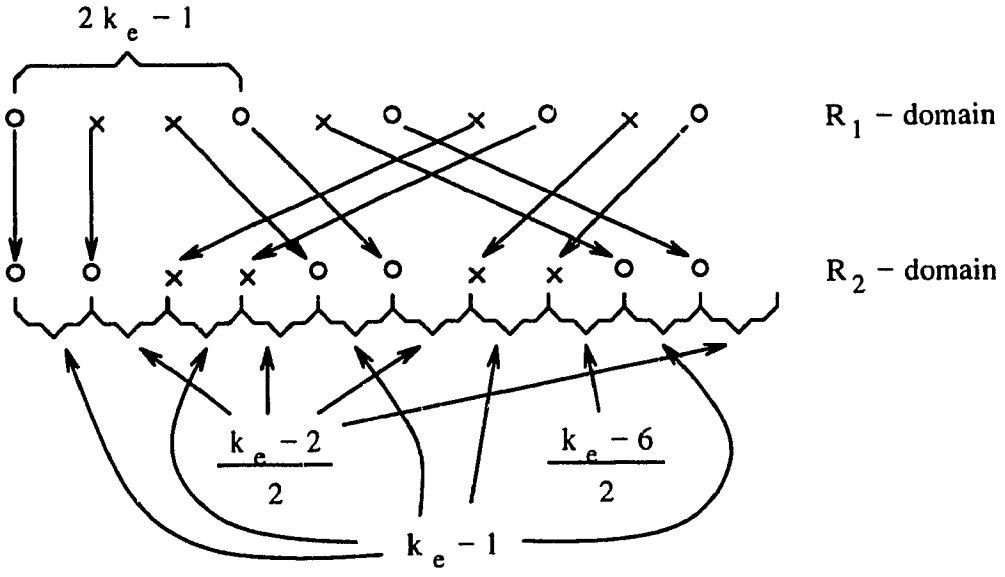


Figure 2.33: The case $t_e = 10$.

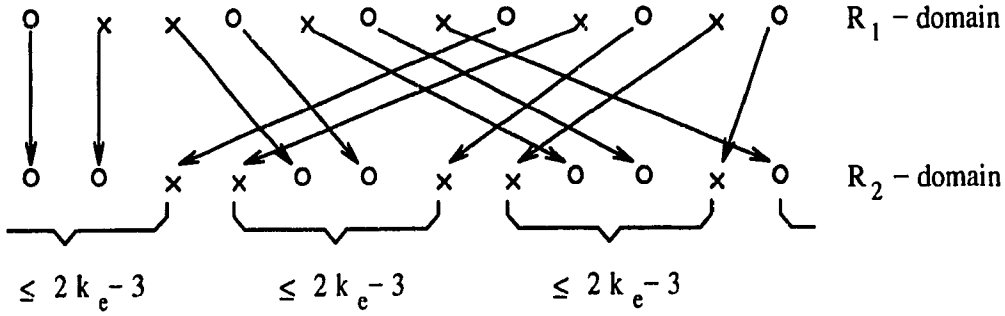


Figure 2.34: The case $t_e = 12$.

or

$$k_e \leq 6 + \frac{8l}{3} \tag{2.44}$$

in the R_2-domain. As $t_e = 10$, then $l \leq 2$. So except for the cases: $l = 0$, $k_e = 6$; $l = 1$, $k_e = 8$; $l = 2$, $k_e = 10$; the (2.44) is a contradiction with respect to the assumption that $k_e \geq 2l + 6$.

Option 2.4.3.2, The case $t_e = 12$:

In the worst case, the code length n, as shown in Figure 2.34, should satisfy:

$$3(2k_e - 3) + 2(k_e - 1) + k_e - 2 + (t_e - 6) \geq n$$

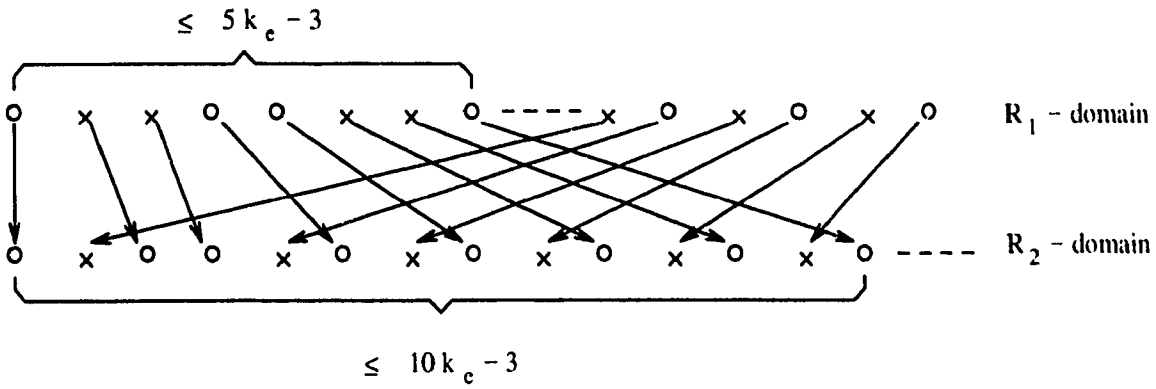


Figure 2.35: A sketch representing the case $N_\sigma = 3$ and $N'_\sigma = 1$ in the worst case.

or

$$k_e \leq 2l + 2$$

in the R_2-domain, which is a contradiction with the assumption that $k - e \geq 2l + 6$.

Case 2.4.4, for $N'_\sigma = 7$:

This part of the proof is very similar to those of Cases 2.4.1, 2.4.2 and 2.4.3, and will be omitted.

So these cases together complete the proof of Lemma 2.4.

Proof of Lemma 2.5: This Lemma will be established through contradiction. Similar to the proof of Lemma 2.4, we consider the following four cases.

Case 2.5.1, for $N'_\sigma = 1$:

In this case, let us assume that there exist an EP such that $N_\sigma = 3$ and $N'_\sigma = 1$ which is not 3-step PD. Therefore, for such an EP, in the worst case, the code length n, as shown in Figure 2.35, should satisfy:

$$(10k_e - 5) + (t_e - 13)(k_e - 2) + (t_e - 12) \geq n$$

or

$$k_e \leq 2l + 4 \tag{2.45}$$

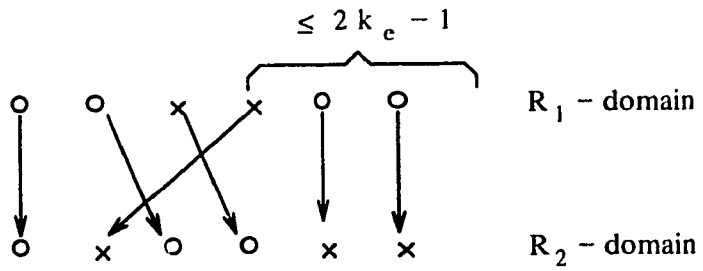


Figure 2.36: The case $t_e = 6$.

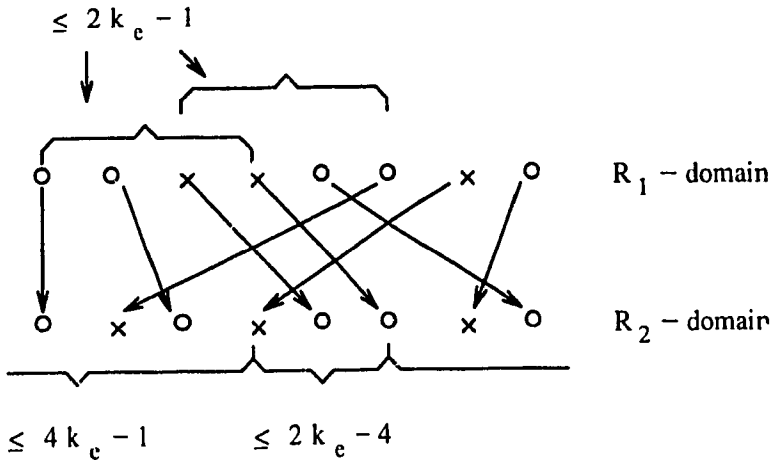


Figure 2.37: The case $t_e = 8$.

in the R_2-domain for $t_e \geq 14$. The condition of Equation (2.45) is a contradiction with respect to the assumptions of Theorem 2.6. For the cases of $t_e \leq 12$, we consider as follows:

Option 2.5.1.1 The case $t_e = 6$:

In the worst case, the code length n, as shown in Figure 2.36, should satisfy:

$$4k_e - 1 \geq n$$

or

$$k_e \leq 4l + 6 = 6$$

So, except for the code (23, 6, 13), it is a contradiction with respect to the assumption that $k_e \geq 2l + 6 = 6$.

Option 2.5.1.2, The case $t_e = 8$:

In the worst case, the code length n, as shown in Figure 2.37, should satisfy:

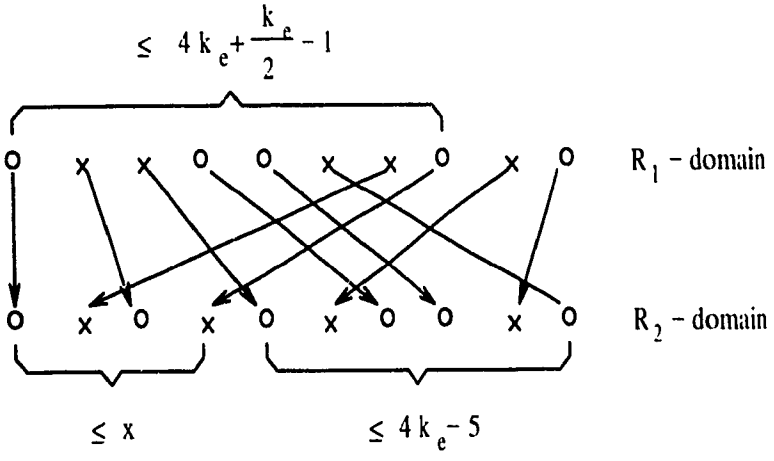


Figure 2.38: The case $t_e = 10$.

$$(2k_e - 4) + (4k_e - 1) + 2 \geq n \tag{2.46}$$

or

$$k_e \leq 4l + 6$$

in the R_2-domain. With $t_e = 8$, and according to the assumption of Theorem 2.6, $l = 1$. Except for the codes (33, 6, 17), (43, 8, 17) and (57, 10, 17), the condition of (2.46) is a contradiction with the assumption that $k_e \geq 2l + 6$.

Option 2.5.1.3, The case $t_e = 10$:

In the worst case, the code length n, as shown in Figure 2.38, should satisfy:

$$\begin{cases} 3k_e - 4 - 2(2l+1) \leq 8 + 2(2l+1), \\ k_e \leq 4 + \frac{4(2l+1)}{3}. \end{cases} \tag{2.47}$$

With $t_e = 10$, according to the assumption of Theorem 2.6, $l \leq 2$. Except the codes (57, 8, 21) and (71, 10, 21), the condition of Equation (2.47) is a contradiction with respect to the assumption of $k_e \geq 2l + 6$.

Option 2.5.1.4, The case $t_e = 12$:

In the worst case, the code length n, as shown in Figure 2.39, should satisfy:

$$(4k_e - 5) + (2k_e - 5) + (4k_e - 1) + 2 - (k_e - 2) \geq n,$$

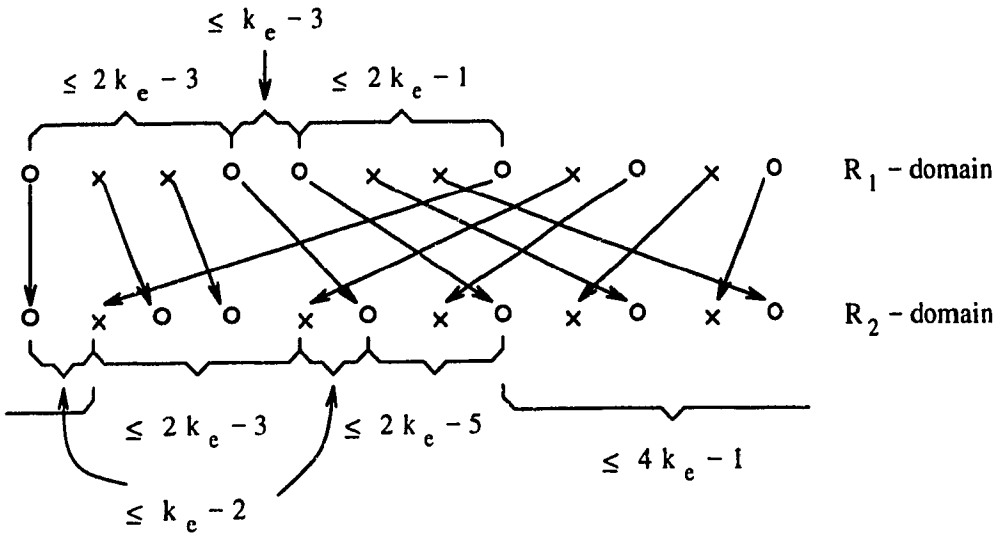


Figure 2.39: The case $t_e = 12$.

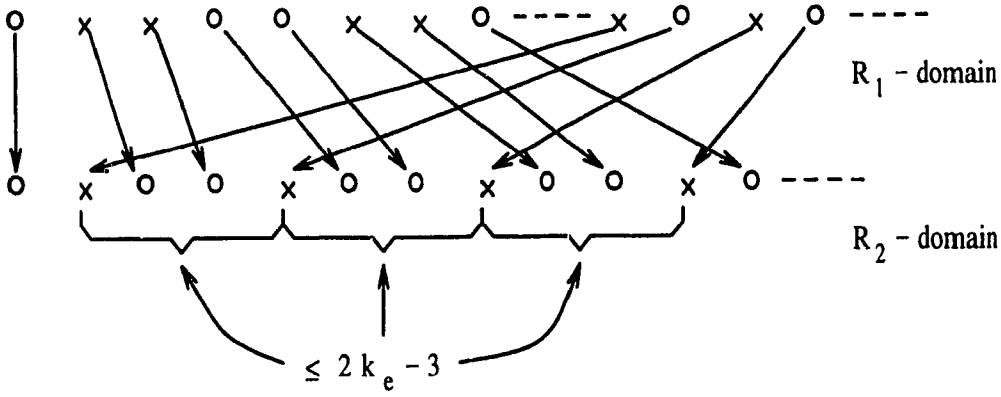


Figure 2.40: A sketch representing the case $N_\sigma = 3$ and $N'_\sigma = 3$ in the worst case.

or

$$k_e \leq 2l + 2$$

which is a contradiction.

Case 2.5.2, For $N'_\sigma = 3$:

In the worst case, the code length n, as shown in Figure 2.40, should satisfy:

$$3(2k_e - 3) + (t_e - 9)(k_e - 2) + (t_e - 6) \geq n, \tag{2.48}$$

or

$$k_e \leq 2l + 2$$

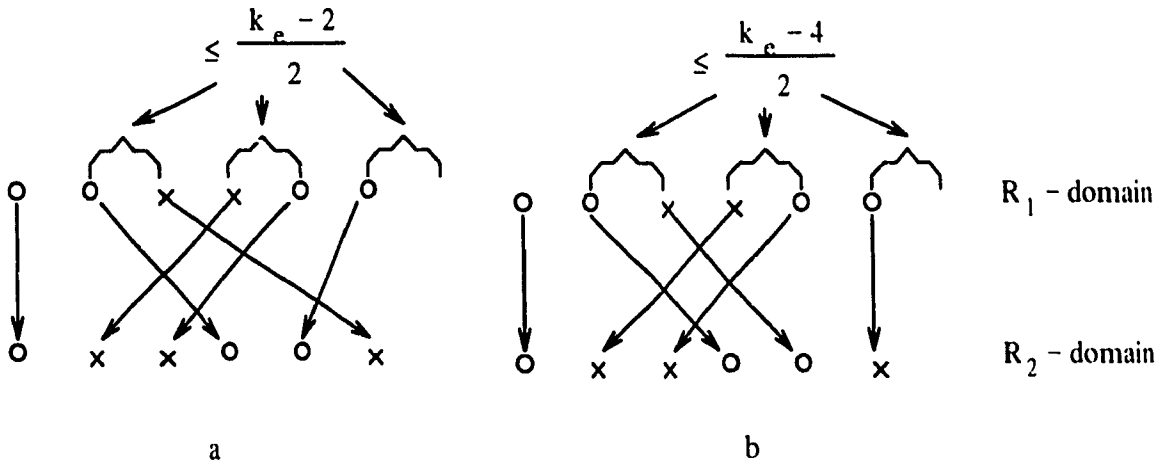


Figure 2.41: The case $t_e = 6$: a: $k_e = 4i$ b: $k_e = 4i + 2$.

for $t_e \geq 12$ in the R_2-domain. The condition of (2.48) is a contradiction with respect to the assumption of Theorem 2.6.

For the case $t_e \leq 10$, we consider the following options.

Option 2.5.2.1, The case $t_e = 6$:

We consider this option by two parts: $k_e = 4i$ and $k_e = 4i + 2$. When $k_e = 4i$, in the worst case, the code length of n, as shown in Figure 2.41, should satisfy:

$$3(k_e - 1) + 3(\frac{k_e - 4}{2}) + 6 \geq n,$$

or

$$k_e \leq 8$$

in R_1-domain. So, when $k_e \geq 12$, it is a contradiction.

When $k_e = 4i + 2$, in the worst case, the code length of n, as shown in Figure 2.41, should satisfy:

$$2(k_e - 1) + 3(\frac{k_e - 2}{2}) + 6 \geq n,$$

or

$$k_e \leq 10 + 4(2l + 1) \tag{2.49}$$

in R_1-domain. So, except for the codes (23, 6, 13), (43, 10, 13) and (63, 14, 13), condition of Equation (2.49) is a contradiction (Note that $l = 0$, as $t_e = 6$).

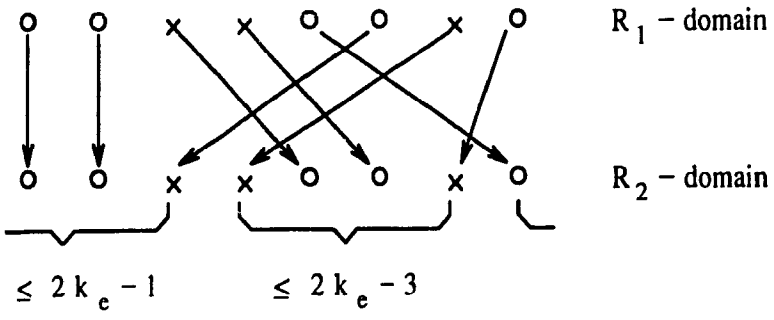


Figure 2.42: The case $t_e = 8$: a: $k_e = 4i$ b: $k_e = 4i + 2$.

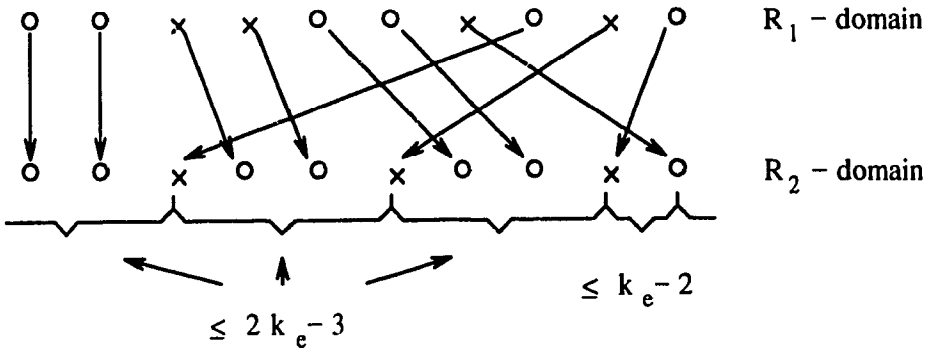


Figure 2.43: The case $t_e = 10$.

Option 2.5.2.2, The case $t_e = 8$:

In the worst case, the code length n, as shown in Figure 2.42, should satisfy:

$$(2k_e - 1) + (2k_e - 3) + (k_e - 1) + (k_e - 2) + (t_e - 4) \geq n,$$

or

$$k_e \leq 4 + 2(2l + 1) \tag{2.50}$$

in R_2-domain. As $t_e = 8$, so $l \leq 1$. Except for the codes (33, 6, 17), (43, 8, 17) and (57, 10, 17), the condition of Equation (2.50) is a contradiction with the assumption that $k_e \geq 2l + 6$.

Option 2.5.2.3, The case $t_e = 10$:

In the worst case, the code length n, as shown in Figure 2.43, should satisfy:

$$3(2k_e - 3) + (k_e - 2) + (t_e - 6) \geq n,$$

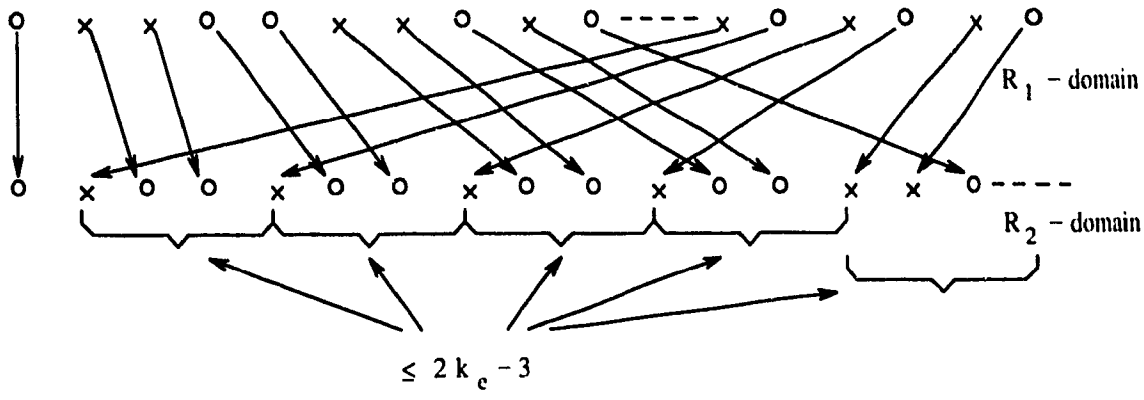


Figure 2.44: A sketch representing the case $N_\sigma = 3$ and $N'_\sigma = 5$ in the worst case.

or

$$k_e \leq 2l + 2$$

in R_2-domain, which is a contradiction with respect to the assumption $k_e \geq 2l + 6$.

Case 2.5.3, For $N'_\sigma = 5$:

In the worst case, the code length n, as shown in Figure 2.44, should satisfy:

$$5(2k_e - 3) + (t_e - 14)(k_e - 2) + (t_e - 10) \geq n,$$

or

$$k_3 \leq \frac{4(l+1)}{3}$$

for $t_e \geq 16$ in R_2-domain, which is a contradiction with respect to the assumption of $k_e \geq 2l + 6$.

For the cases of $t_e \leq 14$, we consider as follows:

Option 2.5.3.1, The case $t_e = 10$:

In the worst case, the code length n, as shown in Figure 2.45, should satisfy:

$$\begin{cases} 2((2(2k_e - 1) + (k_e - 1) + 2) + 1) - 1 - x \geq n, \\ x + 2(2k_e - 3) + (k_e - 1) + 4 \geq n \end{cases} \tag{2.51}$$

in R_2-domain. That is:

$$4k_e - 6 - 2(2l + 1) \leq x \leq k_e + 8 + 2(2l + 1), \tag{2.52}$$

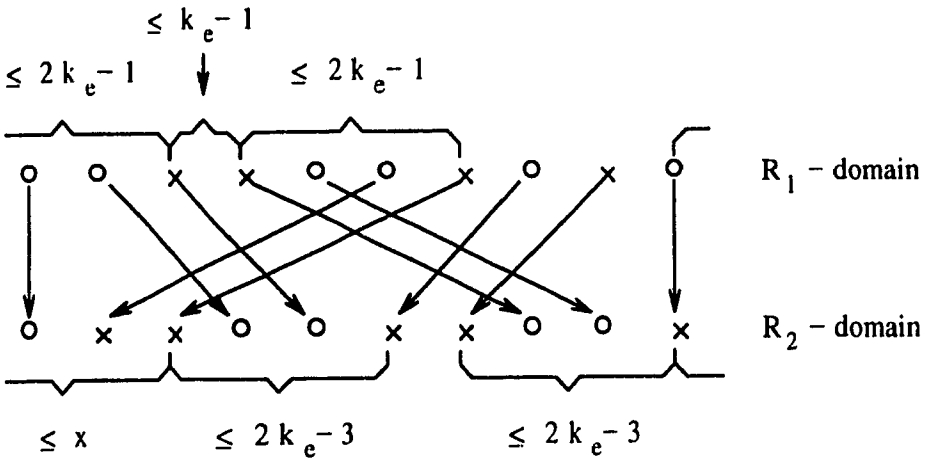


Figure 2.45: The case $t_e = 10$.

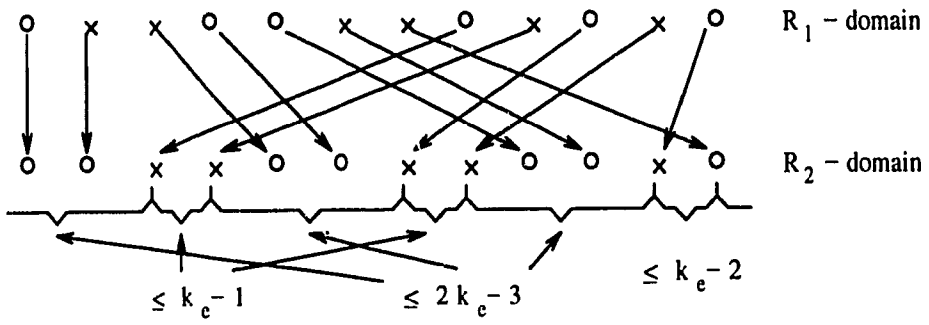


Figure 2.46: The case $t_e = 12$.

or

$$k_e \leq \frac{14 + 4(2l+1)}{3}.$$

As $t_e = 10$, so $l \leq 2$. Except for the codes (43, 6, 21), (57, 8, 21) and (71, 10, 21), Equation (2.52) is a contradiction with respect to the assumption of $k_e \geq 2l + 6$.

Option 2.5.3.2, The case $t_e = 12$:

In the worst case, the code length n, as shown in Figure 2.46, should satisfy:

$$3(2k_e - 3) + 2(k_e - 1) + (k_e - 2) + 6 \geq n,$$

or

$$k_e \leq 2l + 2$$

in R_2-domain. Which is a contradiction with the assumption of $k_e \geq 2l + 6$.

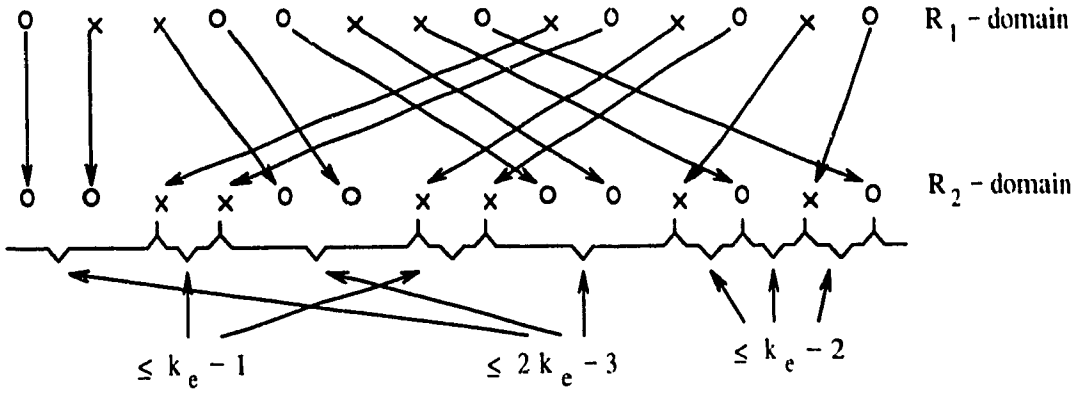


Figure 2.47: The case $t_e = 14$.

Option 2.5.3.3, The case $t_e = 14$:

In the worst case, the code length n, as shown in Figure 2.47, should satisfy:

$$3(2k_e - 3) + 2(k_e - 1) + 3(k_e - 2) + (t_e - 6) \geq n,$$

or

$$k_e \leq 2l + 2$$

in R_2-domain, which is a contradiction with respect to the assumption of $k_e \geq 2l + 6$.

Case 2.5.4, For $N'_\sigma = 7$:

This part of proof is very similar to those of cases 2.5.1, 2.5.2 and 2.5.3, and will be omitted.

Therefore, Cases 2.5.1, 2.5.2, 2.5.3 and 2.5.4 together complete the proof of Lemma 2.5.

Proof of Lemma 2.6: This Lemma will be established through contradiction. Similar to the proof of Lemma 2.5, we consider the following four cases:

Case 2.6.1, For $N'_\sigma = 1$:

In this case, let us assume that there exist an EP such that $N_\sigma = 5$ and $N'_\sigma = 1$ which is not 3-step PD. Therefore, for such an EP, in the worst case, the code length n, as shown in Figure 2.48, should satisfy:

$$(4k_e - 5) + 2(4k_e - 1) + 2(2k_e - 1) + (t_e - 21)(k_e - 2) + (t_e - 16) \geq n,$$

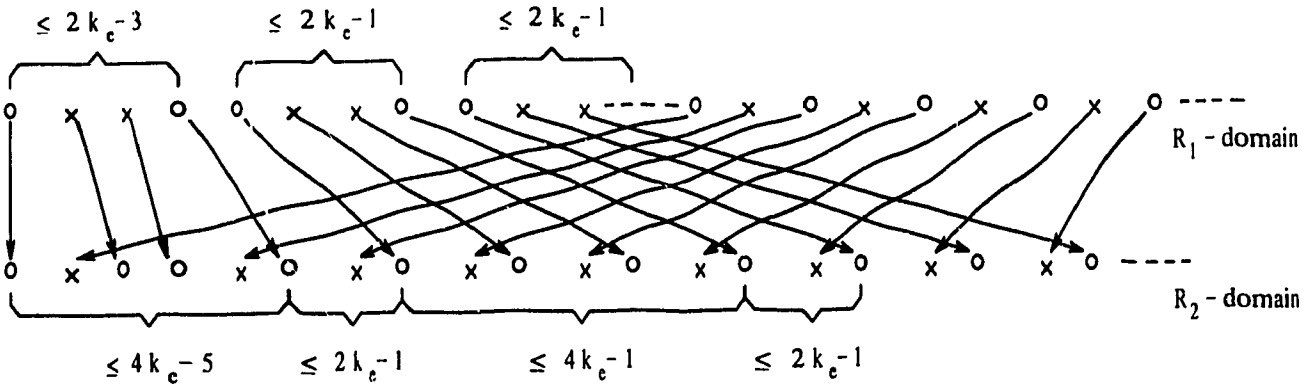


Figure 2.48: A sketch responding the case $N_\sigma = 5$ and $N'_\sigma = 1$ in the worst case.

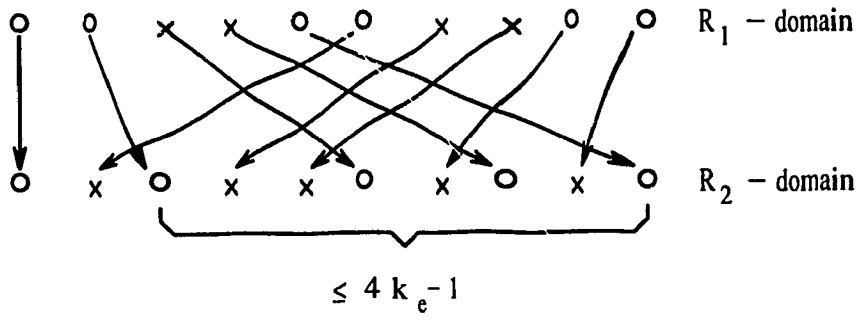


Figure 2.49: The case $t_e = 10$.

or

$$k_e \leq \frac{2+l}{2}$$

for $t_e \geq 22$ in R_2-domain.

For the cases of $t_e \leq 20$, we consider as follows:

Option 2.6.1.1, The case $t_e = 10$:

In the worst case, the code length n, as shown in Figure 2.49, should satisfy:

$$(4k_e - 1) + 3(k_e - 2) + (t_e - 6) \geq n,$$

or

$$k_e \leq 2l + 4$$

in R_2-domain. Which is a contradiction with the assumption that $k_e \geq 2l + 6$.

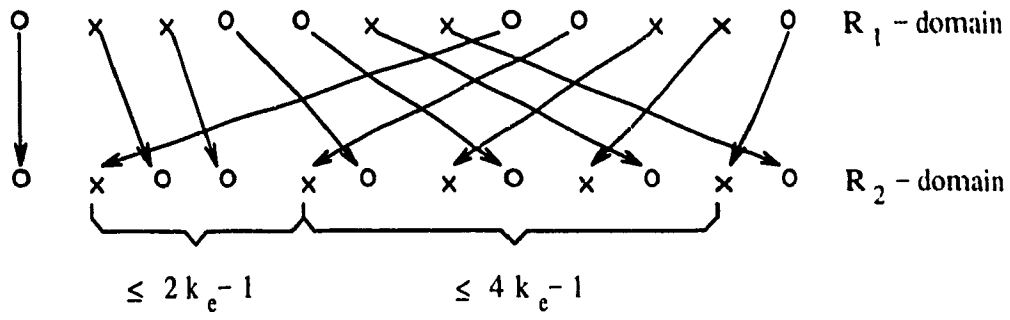


Figure 2.50: The case $t_e = 12$.

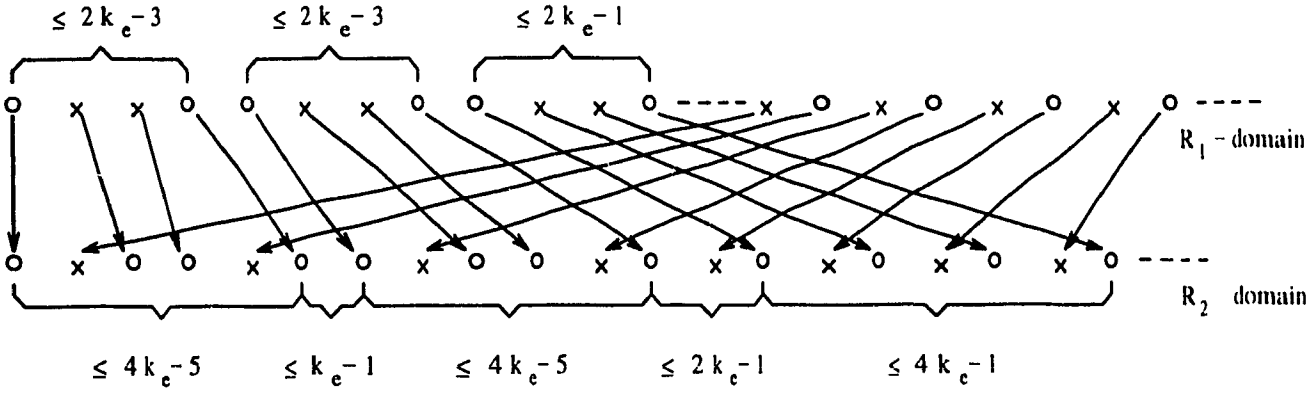


Figure 2.51: A sketch representing the case $N_\sigma = 5$ and $N'_\sigma = 3$ in the worst case.

Option 2.6.1.2, The case $t_e = 12$:

In the worst case, the code length of n, as shown in Figure 2.50, should satisfy:

$$(4k_e - 1) + (2k_e - 1) + 3(k_e - 2) + (t_e - 7) \geq n,$$

or

$$k_e \leq 2l + 4$$

which again is a contradiction with the assumption of $k_e \geq 2l + 6$. In the similar way, we can have the result that when $14 \leq t_e \leq 20$, the error pattern also make contradiction with respect to the assumption in the worst case.

Case 2.6.2, For $N'_\sigma = 3$:

In the worst case, the code length n, as shown in Figure 2.51, should satisfy:

$$2(4k_e - 5) + (4k_e - 1) + (2k_e - 1) + (k_e - 1) + (t_e - 19)(k_e - 2) + (t_e - 14) \geq n,$$

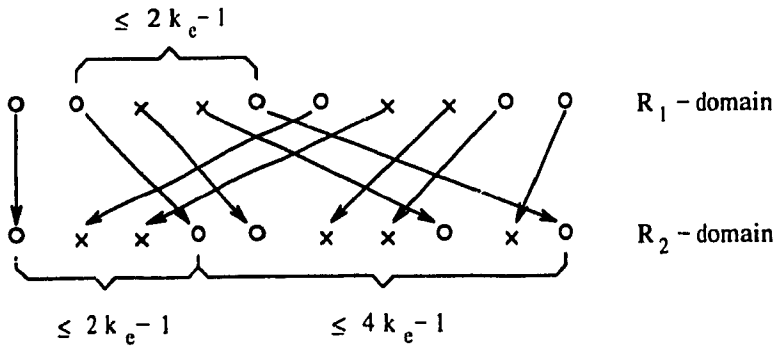


Figure 2.52: The case $t_e = 10$.

or

$$k_e \leq \frac{4l}{3} + 4$$

for $t_e \geq 20$ in R_2-domain, which is a contradiction with the assumption of $k_e \geq 2l + 4$.

For the cases of $t_e \leq 18$, we consider as follows:

Option 2.6.2.1 The case $t_e = 10$:

In the worst case, the code length of n, as shown in Figure 2.52, should satisfy:

$$(4k_e - 1) + (2k_e - 1) + (k_e - 2) + 3 \geq n,$$

or

$$k_e \leq 4 + 2l$$

in R_2-domain, which is a contradiction with respect to the assumption of $k_e \geq 2l + 6$.

Option 2.6.2.2, The case $t_e = 12$:

In the worst case, the code length of n, as shown in Figure 2.53, should satisfy:

$$2(4k_e - 1) - 1 + (k_e - 2) + (t_e - 10) \geq n,$$

or

$$k_e \leq 2l + 4$$

in R_2-domain, which is a contradiction.

In the same way, we can have the result that when $14 \leq t_e \leq 18$, the error pattern also make contradiction with the assumption in the worst case.

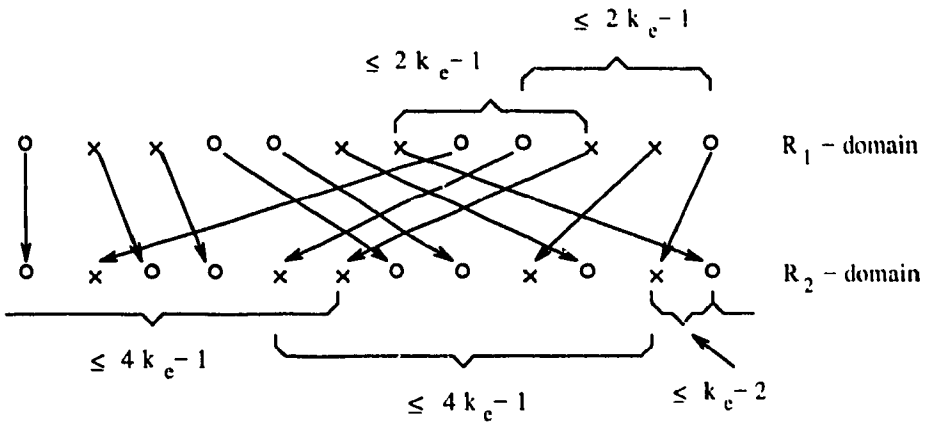


Figure 2.53: The case $t_e = 12$.

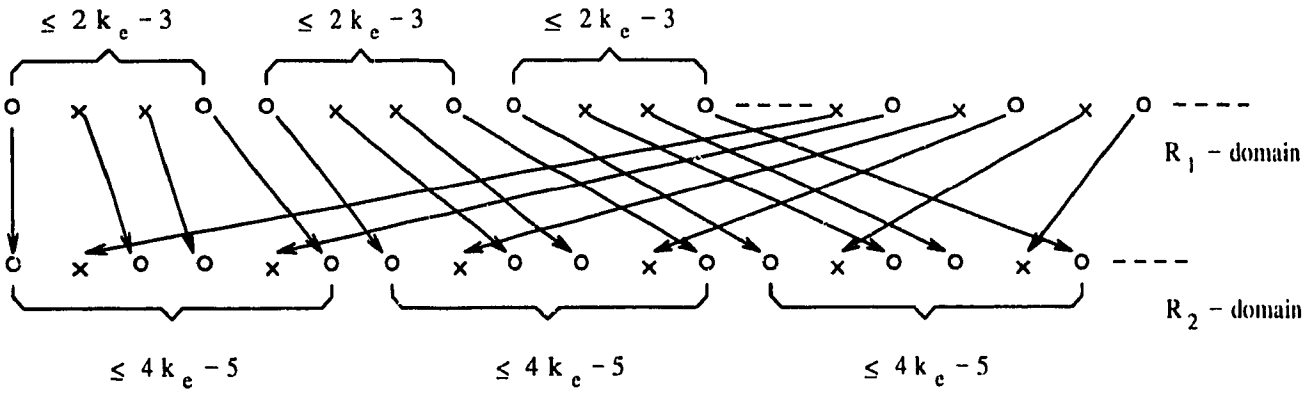


Figure 2.54: A sketch representing the case $N_\sigma = 5$ and $N'_\sigma = 5$ in the worst case.

Case 2.6.3, for $N'_\sigma = 5$:

In the worst case, the code length n, as shown in Figure 2.54, should satisfy:

$$3(4k_e - 5) + 2(k_e - 2) + (t_e - 17)(k_e - 2) + (t_e - 12) \geq n,$$

or

$$k_e \leq 2l + 2$$

for $t_e \leq 18$in R_2-domain, which is a contradiction with the assumption of $k_e \geq 2l + 6$. For the cases of $t_e \leq 16$, we consider as follows:

Option 2.6.3.1, The case $t_e = 10$:

In the worst case, the code length for n, as shown in Figure 2.55, should satisfy:

$$5(\frac{k_e - 2}{2}) + 5(k_e - 1) + 10 \geq n,$$

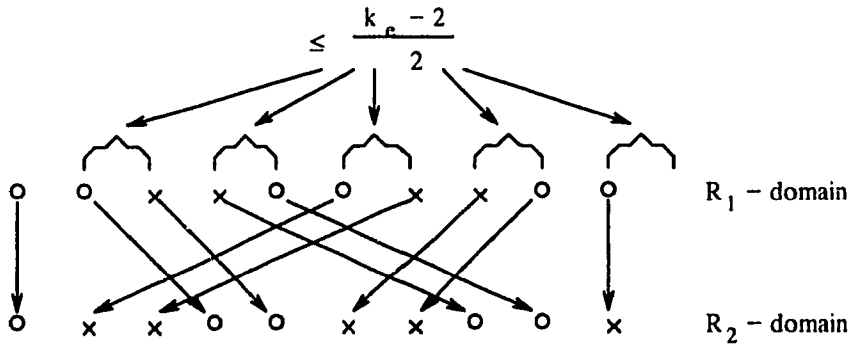


Figure 2.55: The case $t_e = 10$.

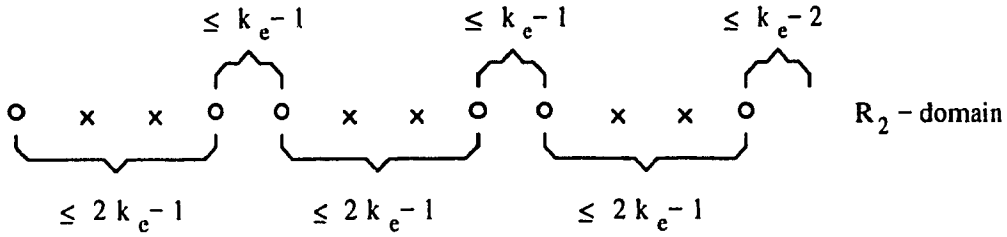


Figure 2.56: The case $t_e = 12$.

or

$$k_e \leq 6 + \frac{4(2l+1)}{3}. \tag{2.53}$$

As $t_e = 10$, so according to the assumption of Theorem 2.6, $l \leq 2$. Therefore, except for the codes (43, 6, 21), (75, 10, 21), Equation (2.53) is a contradiction with respect to the assumption of $k_e \geq 2l + 6$.

Option 2.6.3.2, The case $t_e = 12$:

In the worst case, the code length n, as shown in Figure 2.56, should satisfy:

$$3(2k_e - 1) + 2(k_e - 1) + (k_e - 2) + (t_e - 6) \geq n,$$

or

$$k_e \leq 6 + 2l \tag{2.54}$$

in the R_2-domain. For $t_e = 12$, according to the assumption of Theorem 2.6, $l \leq 3$. So, except for the codes (53, 6, 25), (71, 8, 25), (89, 10, 25) and (107, 12, 25), (2.54) is a contradiction with respect to the assumption that $k_e \geq 6 + 2l$. A computer was used to

search for the error pattern of the codes (53, 6, 25) and (71, 8, 25), with the result that no EPs exist in these two codes. So the codes (53, 6, 25) and (71, 8, 25) are also 3-step PD.

Case 2.6.4, for $N'_\sigma = 7$:

This part of the proof is very similar to those of cases 2.6.1, 2.6.2 and 2.6.3, and are omitted here.

Cases 2.6.1, 2.6.2, 2.6.3 and 2.6.4 together complete the proof of Lemma 2.6.

In a same way, Lemma 2.7 can be proven.

Lemmas 2.4, 2.5, 2.6 and 2.7 together completes the proof of Theorem 2.6.

Q. E. D.

Based on the results obtained above, we give tables 2.3 and 2.4, for the specific numbers of correctable errors $t_e = 6$ and $t_e = 10$, as illustrative examples. These tables show the tight lower bounds on code length n, for a given information length k, for 1-, 2- and 3-step *PD* codes.

2.4 Conclusions

In this chapter, we have developed tight lower bounds on the code length n, or equivalently the upper bounds on the code rate $\frac{k}{n}$. The results are for 3-step *PD* codes with even valued t_e. We present the main results as follows:

RESULT 1: The $(n, k_o, 2t_e + 1)$ codes with $n = (t_e - 1)k_o - 2(2l + 1)$, for $l = \frac{t_e-4}{2}$, or $l = \frac{k_o-1}{2}$, and the code (53, 7, 21), are not 3-step *PD*.

RESULT 2: The $(n, k_o, 2t_e+1)$ codes with $n = (t_e-1)k_o-2(2l+1)$, for $k_o \geq 2l+3$, $0 \leq l \leq \frac{t_e-6}{2}$, except for the code (53, 7, 21), are 3-step *PD*.

RESULT 3: The $(n, k_e, 2t_e+1)$ codes with $n = (t_e-1)(k_e-1)-2(2l+1)$, for $l = \frac{t_e-4}{2}$, or $l = \frac{k_e-4}{2}$, and the following codes:

1. $l = 0, \quad t_e = 6, \quad k_e = 6, 8, 10, 14,$

   $8 \leq t_e \leq 10, \quad k_e = 6;$

Table 2.3: Tight lower bounds on the code length n for $t_e = 6$

k	1-step PD codes	2-step PD codes	3-step PD codes
2	13	11	11
3	19	17	13
4	25	19	17
5	31	27	23
6	37	29	29
7	43	37	33
8	49	39	37
9	55	47	43
10	61	49	47
11	67	57	53
12	73	59	53
13	79	67	63
14	85	69	67
15	91	77	73
16	97	79	73
⋮	⋮	⋮	⋮

Table 2.4: Tight lower bounds on the code length n for $t_e = 10$

k	1-step PD codes	2-step PD codes	3-step PD codes
2	21	17	17
3	31	29	25
4	41	31	29
5	51	47	39
6	61	49	47
7	71	65	57
8	81	67	61
9	91	83	71
10	101	85	79
11	111	101	89
12	121	103	89
13	131	119	107
14	141	121	107
15	151	137	125
16	161	139	125
⋮	⋮	⋮	⋮

2. $l = 1, \quad 8 \leq t_e \leq 10, \quad 8 \leq k_e \leq 10;$

3. $l = 2, \quad t_e = 12, \quad k_e = 10;$

4. $l = 3, \quad t_e = 12, \quad k_e = 12.$

are not 3-step PD.

RESULT 4: The $(n, k_e, 2t_e + 1)$ codes with $n = (t_e - 1)(k_e - 1) - 2(2l + 1)$, for $k_e \geq 2l + 6$, $0 \leq \frac{t_e - 6}{2}$, are 3-step PD. Exceptions are:

1. $l = 0, \quad t_e = 6, \quad k_e = 6, 8, 10, 14,$

   $8 \leq t_e \leq 10, \quad k_e = 6;$

2. $l = 1, \quad 8 \leq t_e \leq 10, \quad 8 \leq k_e \leq 10;$

3. $l = 2, \quad t_e = 12, \quad k_e = 10;$

4. $l = 3, \quad t_e = 12, \quad k_e = 12.$

The above results provide the following two bound improvements:

BOUND 1: The lower bounds on n for the $(n, k_o, 2t_e + 1)$ codes have been improved by as much as $2(2l + 2)$, where $l = \min\{\frac{t_e - 6)}{2}; \frac{k_e - 3}{2}\}$, compare to 2-step permutation (Theorem 2 of [33]).

BOUND 2: The lower bounds on n for the $(n, k_e, 2t_e + 1)$ codes have been improved by as much as $2(2l + 3)$, where $l = \min\{\frac{t_e - 6}{2}; \frac{k_e - 6}{2}\}$, compare to the 2-step permutation (Theorem 3 of [33]).

Note that the bound improvements in both bounds: $2(2l+2) = \min\{2(t_e - 4); 2(k_o - 1)\}$ in Bound 1 and $2(2l + 3) = \min\{2(t_e - 3); 2(k_e - 3)\}$ in Bound 2 are proportional to k_o or k_e and t_e. From the above mentioned results, it can be concluded that when $t_e > k_e$ (or k_o), the codes $(n, k_o, 2t_e + 1)$ are improved more than the codes $(n, k_e, 2t_e + 1)$.

Further note that the improvement of 2-step PD with respect to 1-step PD is proportional to k or t [33], and this is also true in the improvement for 3-step PD with respect to 2-step PD. These results show that error-trapping decoding can be applied to higher rate cyclic codes.

Chapter 3

CAPABILITY OF MULTIPLE-STEP (T, U) PERMUTATION DECODING OF CYCLIC CODES

3.1 Introduction

The exact lower bounds on the performance of s-step permutation decodable cyclic codes with $s \leq 3$ were derived in [33, 35]. One can expect that a specific cyclic code is more likely to be permutation-decodable by increasing s. However, the general relationship between the required number of permutations s and the code parameters n, k and t of permutation decodable (PD) cyclic codes is unknown.

In this chapter, we examine a lower bound on the code length n for a given set of k, t and s, where t is odd. Then, the *turn-point* parameter k^* is introduced and a general relationship between s and the code parameters n, k and t of permutation-decodable cyclic codes is established. An optimum permutation step which provides the largest improvement in code rates of PD codes is also defined. Based on these results, we will show how the

code rate of a (T, U) permutation decodable cyclic code increases when the permutation step increases.

This chapter is organized as follows. In Section 3.2, the characteristics of error-free gaps are discussed. The capability of multiple-step (T, U) permutation decoding is given in Section 3.3. In Section 3.4, we present a bound on the code rate and a study on the relationship between the code parameters n, k, t and the number of permutation steps s required to decode the code. The characteristics of the code rate and some related examples are given in Section 3.5. Section 3.6 presents the conclusions on the obtained results.

3.2 Characteristics of Error-free Gaps

A permuted error polynomial, $U^i e(x)$, containing t errors can be written as

$$U^i e(x) = \sum_{j=1}^{t} e_{p_j(i)} x^{p_j(i)}$$

where $p_1(i) < p_2(i) < \cdots < p_t(i)$ represent the error positions. The difference

$$g_i^j = p_{j+1}(i) - p_j(i)$$

indicates the length of the error-free gap between two adjacent error positions $p_j(i)$ and $p_{j+1}(i)$. The objective of applying (T, U) permutation is obviously to rearrange the error positions in the permuted error polynomial in such a way that we can obtain at least one error-free gap length larger than or equal to k. Therefore, it is important to understand the characteristics of the error-free gaps. For this reason, we will study in this section the relationship between the error-free gaps in two permuted error polynomials $U^i e(x)$ and $U^l e(x)$, $i \neq l$.

Due to the properties of cyclic codes, we can assume that one error position is at $p_1(i) = 0$ without loss of generality. This even-valued error position remains unchanged when permutation is applied. In the following analysis, we consider only the γ pattern with alternate even-valued and odd-valued error positions as shown in Figure 3.1. It is obvious that there are other error patterns. However, it was shown in [33] that the γ error pattern

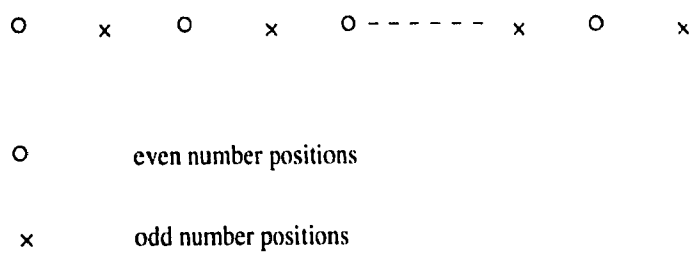


Figure 3.1: Error pattern with alternate even-value and odd-value error positions.

results in the worst case in terms of error-free gaps (more about this will be discussed in Section 3.3).

Let $\mathbf{g}_i$ be a $t \times 1$ column vector called the *gap-length vector* corresponding to the permuted error polynomial $U^i e(x)$, and defined as

$$\mathbf{g}_i = \left[g_i^1,\ g_i^2,\ \ldots,\ g_i^t\right]^T$$

We note that

$$\sum_{j=1}^{t} g_i^j = n - t.$$

The following theorems establish the relationship between two gap-length vectors $\mathbf{g}_i$ and $\mathbf{g}_l$.

Theorem 3.1 (RELATION BETWEEN $\mathbf{g}_i$ and $\mathbf{g}_{i+1}$) *If the error patterns of the permuted error polynomials $U_i e(x)$ and $U_{i+1} e(x)$ are of γ type, then their respective gap length vectors $\mathbf{g}_i$ and $\mathbf{g}_{i+1}$ are related by the following mapping:*

$$\mathbf{g}_{i+1} = \mathbf{M} \cdot \mathbf{g}_i$$

where the $t \times t$ matrix $\mathbf{M}$ is called the ***mapping matrix*** *and has the following structure:*

$$\mathbf{M} = \begin{bmatrix} 1 & 1 & \ldots & 1 & -1 & -1 & \ldots & -1 \\ 1 & -1 & \ldots & -1 & 1 & 1 & \ldots & 1 \\ & & & \ddots & & & & \\ -1 & -1 & \ldots & -1 & 1 & 1 & \ldots & 1 \end{bmatrix}. \tag{3.1}$$

That is, every row in $\mathbf{M}$ is the $\frac{t-1}{2}$th cyclic shift to the left of the previous row, and the first row can be viewed as the $\frac{t-1}{2}$th cyclic shift to the left of the last row.

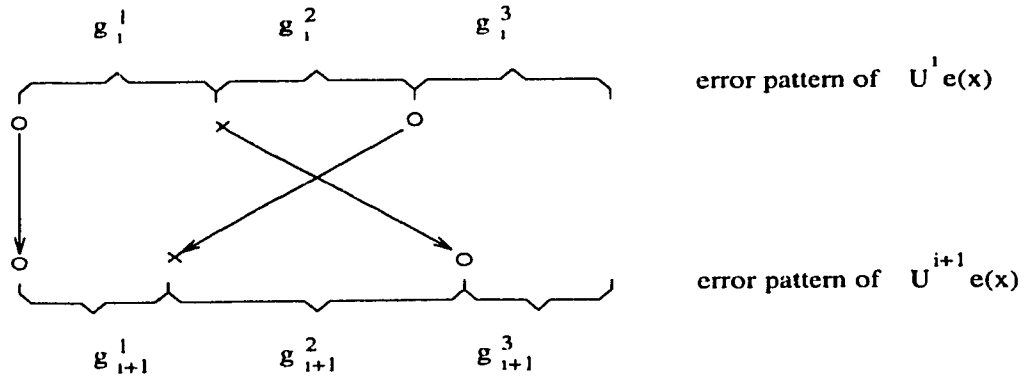


Figure 3.2: Gap relations between $U^i e(x)$ and $U^{i+1} e(x)$ for the case $t = 3$.

Proof of Theorem 3.1: We will prove Theorem 3.1 by induction.

First, consider the case for $t = 3$ over $GF(2)$.

By applying another permutation on $U^i e(x)$ to obtain $U^{i+1} e(x)$, the gap between two adjacent error positions in $U^i e(x)$ is doubled.

$$\begin{cases} g_{i+1}^1 + g_{i+1}^2 = 2g_i^1 \\ g_{i+1}^2 + g_{i+1}^3 = 2g_i^3 \\ g_{i+1}^1 + g_{i+1}^3 = 2g_i^2 \end{cases} \tag{3.2}$$

which can be written in its matrix form

$$\mathbf{g}_{i+1} = \begin{bmatrix} 1 & 1 & -1 \\ 1 & -1 & 1 \\ -1 & 1 & 1 \end{bmatrix} \mathbf{g}_i. \tag{3.3}$$

It follows that

$$\mathbf{M} = \begin{bmatrix} 1 & 1 & -1 \\ 1 & -1 & 1 \\ -1 & 1 & 1 \end{bmatrix},$$

which satisfies the properties mentioned in Theorem 3.1.

Now, suppose that the Theorem is true for $t = t_c$. We are going to prove that the Theorem is also true for $t = t_c + 2$.

Figure 3.3 illustrates the relationship of two consecutive error domains with γ type error

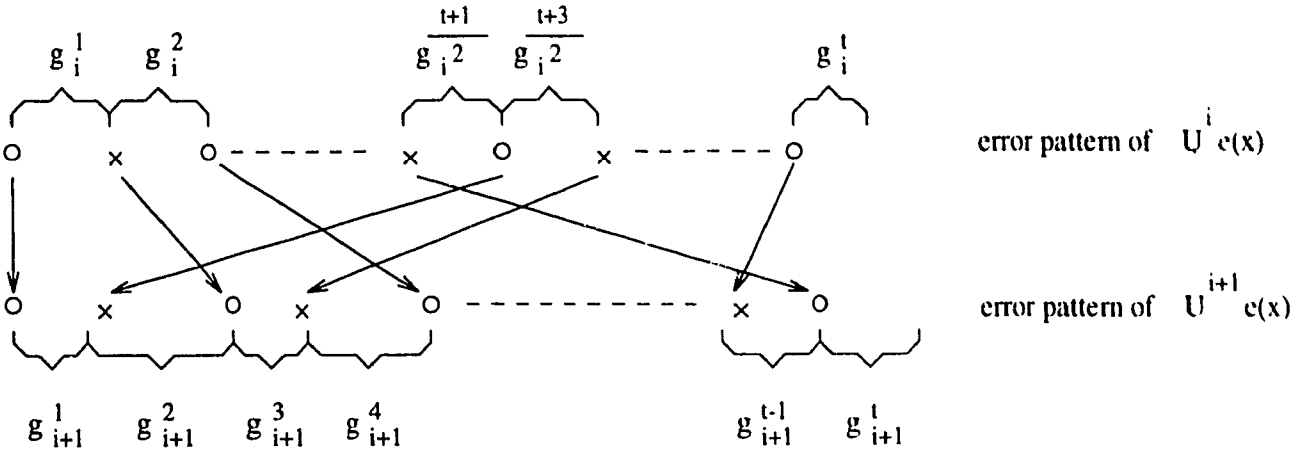


Figure 3.3: Gap relations between $U^i e(x)$ and $U^{i+1} e(x)$ for the case $t = t_c$.

patterns for $t = t_c$, i.e.,

$$\begin{cases} g_{i+1}^1 + g_{i+1}^2 &= 2g_i^1 \\ g_{i+1}^2 + g_{i+1}^3 &= 2g_i^{\frac{t_c+3}{2}} \\ g_{i+1}^3 + g_{i+1}^4 &= 2g_i^2 \\ & \vdots \\ g_{i+1}^{t_c-1} + g_{i+1}^{t_c} &= 2g_i^{t_c} \\ g_{i+1}^1 + g_{i+1}^{t_c} &= 2g_i^{\frac{t_c-1}{2}} \end{cases} \tag{3.4}$$

Equation (3.4) can also be represented in a matrix form:

$$\mathbf{g}_{i+1} = \mathbf{M} \cdot \mathbf{g}_i,$$

where

$$\mathbf{M} = \begin{bmatrix} 1 & 1 & \dots & 1 & -1 & -1 & \dots & -1 \\ 1 & -1 & \dots & -1 & 1 & 1 & \dots & 1 \\ & & & \ddots & & & & \\ -1 & -1 & \dots & -1 & 1 & 1 & \dots & 1 \end{bmatrix}_{t_c \times t_c}$$

Now, consider the case $t = t_c + 2$, where there are 2 additional errors inserted into the error pattern shown in Figure 3.3. Since the error patterns of $U^i e(x)$ and $U^{i+1} e(x)$ are of the γ-type, the two additional errors should be inserted into $U^i e(x)$ with one before $\frac{n}{2}$ and the other after $\frac{n}{2}$. These two errors will be mapped onto two adjacent positions in $U^{i+1} e(x)$.

Therefore, for the case $t = t_c + 2$, the gap relationship between two consecutive domains becomes:

$$\begin{cases} g_{i+1}^1 + g_{i+1}^2 = 2g_i^1 \\ g_{i+1}^2 + g_{i+1}^3 = 2g_i^{\frac{t_c+5}{2}} \\ g_{i+1}^3 + g_{i+1}^4 = 2g_i^2 \\ \quad\vdots \\ g_{i+1}^{t_c+1} + g_{i+1}^{t_c+2} = 2g_i^{t_c+2} \\ g_{i+1}^1 + g_{i+1}^{t_c+2} = 2g_i^{\frac{t_c+3}{2}} \end{cases} \tag{3.5}$$

For the first $(t_c + 1)$ rows of (3.5), if we add all the rows with odd sequential number then subtract all the rows with even sequential number, we get:

$$g_{i+1}^1 - g_{i+1}^{t_c+2} = 2(g_i^1 - g_i^{\frac{t_c+5}{2}} + g_i^2 - g_i^{\frac{t_c+7}{2}} + \cdots + g_i^{\frac{t_c+1}{2}} - g_i^{t_c+2}) \tag{3.6}$$

Now, we add (3.6) to the last row of (3.5) and after some manipulation, we get:

$$g_{i+1}^1 = g_i^1 + \cdots + g_i^{\frac{t_c+3}{2}} - (g_i^{\frac{t_c+5}{2}} + \cdots + g_i^{t_c+2}) \tag{3.7}$$

On the other hand, (3.5) can be converted into the following closed form relations:

$$\begin{cases} g_{i+1}^{j+1} = 2g_i^{\frac{j+1}{2}} - g_{i+1}^j & j = 1, 3, \cdots, t_c; \\ g_{i+1}^{j+1} = 2g_i^{\frac{t_c+3}{2}+\frac{j}{2}} - g_{i+1}^j & j = 2, 4, \cdots, t_c + 1; \\ g_{i+1}^{t_c+2} = 2g_i^{\frac{t_c+3}{2}} - g_{i+1}^1. \end{cases} \tag{3.8}$$

The above equation indicates that, for the case odd $j = 1, 3, \cdots, t_c$,

$$\begin{aligned} g_{i+1}^{j+1} &= 2g_i^{\frac{j+1}{2}} - g_{i+1}^j \\ &= 2g_i^{\frac{j+1}{2}} - \mathbf{m}_j\mathbf{g}_i \\ &= \mathbf{m}_{j+1}\mathbf{g}_i \end{aligned}$$

where $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_t$ represent the t rows of matrix $\mathbf{M}$. That is to say, if we first change the signs of the j-th row of $\mathbf{M}$, then change the sign of the $\frac{j+1}{2}$th element of it from -1 to $+1$, we get the $(j+1)$-th row of $\mathbf{M}$ (later we will show that the $\frac{j+1}{2}$th element of $\mathbf{m}_j$ is -1). Equation (3.7) is the case $j = 1$, i.e.,

$$g_{i+1}^1 = \mathbf{m}_1\mathbf{g}_i$$

where

$$\mathbf{m}_1 = [\underbrace{1 \ 1 \ \ldots \ 1}_{\frac{t_c+3}{2}} \ \underbrace{-1 \ -1 \ \ldots \ -1}_{\frac{t_c+1}{2}}]. \tag{3.9}$$

Now, for the case even $j = 2, 4, \cdots, t_c + 1$,

$$\begin{aligned} g_{i+1}^{j+1} &= 2g_i^{\frac{t_c+3}{2}+\frac{j}{2}} - g_{i+1}^{j} \\ &= 2g_i^{\frac{t_c+3}{2}+\frac{j}{2}} - \mathbf{m}_j \mathbf{g}_i \\ &= \mathbf{m}_{j+1}\mathbf{g}_i. \end{aligned} \tag{3.10}$$

Therefore, if we first change the signs of the j-th row of $\mathbf{M}$, then change the sign of the $(\frac{t_c+3}{2} + \frac{j}{2})$-th element of it from -1 to $+1$, we get the $(j+1)$-th row of $\mathbf{M}$ (later we will show that the $(\frac{t_c+3}{2} + \frac{j}{2})$-th element of $\mathbf{m}_j$ is -1). From (3.8), we have:

$$\begin{aligned} g_{i+1}^{2} &= 2g_i^{1} - g_{i+1}^{1} \\ &= 2g_i^{1} - \mathbf{m}_1 \mathbf{g}_i \\ &= [1 \ \underbrace{-1 \ -1 \ \ldots \ -1}_{\frac{t_c+1}{2}} \ \underbrace{1 \ 1 \ \ldots \ 1}_{\frac{t_c+1}{2}}]\mathbf{g}_i \\ &= \mathbf{m}_2 \mathbf{g}_i. \end{aligned} \tag{3.11}$$

From (3.9) and (3.11) we can see that $\mathbf{m}_2$ is the $\frac{t_c+1}{2}$th cyclic shift to the left of $\mathbf{m}_1$.

Similarly, from (3.10) and $j = 2$, we have:

$$\begin{aligned} g_{i+1}^{3} &= 2g_i^{\frac{t_c+5}{2}} - g_{i+1}^{2} \\ &= 2g_i^{\frac{t_c+5}{2}} - \mathbf{m}_2 \mathbf{g}_i \\ &= [-1 \ \underbrace{1 \ 1 \ \ldots \ 1}_{\frac{t_c+3}{2}} \ \underbrace{-1 \ -1 \ \ldots \ -1}_{\frac{t_c-1}{2}}] \\ &= \mathbf{m}_3 \mathbf{g}_i. \end{aligned} \tag{3.12}$$

Therefore, $\mathbf{m}_3$ is the $\frac{t_c+1}{2}$th cyclic shift to the left of $\mathbf{m}_2$. The difference between $\mathbf{m}_1$ and $\mathbf{m}_3$ is that $\mathbf{m}_3$ is the (t_c+1)-th cyclic shift of $\mathbf{m}_1$ to the left, which is also equivalent to cyclic shift of $\mathbf{m}_1$ once to the right. A similar relation also exists between $\mathbf{m}_2$ and $\mathbf{m}_4$.

In general, suppose the relation between $\mathbf{m}_{j-2}$ and $\mathbf{m}_j$ (when j is odd) is the same as

the relation between $\mathbf{m}_1$ and $\mathbf{m}_3$, then $\mathbf{m}_j$ takes the form:

$$[\underbrace{-1\ -1\ \ldots\ -1}_{\frac{j-1}{2}}\ \underbrace{1\ 1\ \ldots\ 1}_{\frac{t_c+3}{2}}\ \underbrace{-1\ -1\ \ldots\ -1}_{\frac{t_c-j+2}{2}}] \tag{3.13}$$

and

$$g_{i+1}^{j} = \mathbf{m}_j \mathbf{g}_i. \tag{3.14}$$

From (3.8), we have

$$\begin{aligned} g_{i+1}^{j+1} &= 2g_i^{\frac{j+1}{2}} - g_{i+1}^{j} \\ &= 2g_i^{\frac{j+1}{2}} - \mathbf{m}_j \mathbf{g}_i, \end{aligned} \tag{3.15}$$

and

$$\begin{aligned} g_{i+1}^{j+2} &= 2g_i^{\frac{t_c+3}{2}+\frac{j+1}{2}} - g_{i+1}^{j+1} \\ &= 2g_i^{\frac{t_c+3}{2}+\frac{j+1}{2}} - 2g_i^{\frac{j+1}{2}} + \mathbf{m}_j \mathbf{g}_i \\ &= \mathbf{m}' \mathbf{g}_i + \mathbf{m}'' \mathbf{g}_i + \mathbf{m}_j \mathbf{g}_i \\ &= (\mathbf{m}' + \mathbf{m}'' + \mathbf{m}_j) \mathbf{g}_i \end{aligned} \tag{3.16}$$

where $\mathbf{m}'$ is a vector with the $\left(\frac{t_c+3}{2} + \frac{j+1}{2}\right)$-th element being $+2$ and other elements being 0, $\mathbf{m}''$ is a vector with the $\frac{j+1}{2}$th element being -2 and other elements being 0. From the definition of the mapping matrix $\mathbf{M}$, we know that

$$g_{i+1}^{j+2} = \mathbf{m}_{j+2} \mathbf{g}_i.$$

Therefore, by combining (3.16) and the above equation, we obtain

$$\mathbf{m}_{j+2} = \mathbf{m}' + \mathbf{m}'' + \mathbf{m}_j.$$

Since in (3.13) the $\left(\frac{t_c+3}{2} + \frac{j+1}{2}\right)$-th element is -1 and the $\frac{j+1}{2}$th element is $+1$, $\mathbf{m}_{j+2}$ has the following form:

$$[\underbrace{-1\ -1\ \ldots\ -1}_{\frac{j+1}{2}}\ \underbrace{1\ 1\ \ldots\ 1}_{\frac{t_c+3}{2}}\ \underbrace{-1\ -1\ \ldots\ -1}_{\frac{t_c-j}{2}}] \tag{3.17}$$

which is again equivalent to cyclic shift of $\mathbf{m}_j$ once to the right. Note that the $\frac{j+1}{2}$th element in equation (3.17) is -1 while the $\frac{j+1}{2}$th element of $\mathbf{m}_j$ is $+1$.

Similarly, we can prove that when j is even, $\mathbf{m}_{j+2}$ is also equivalent to a cyclic shift of $\mathbf{m}_j$ once to the right.

From above, we see that when j increases by 2, then $\mathbf{m}_j$ is shifted to the left (t_c+1) times which is equivalent to saying that it is shifted to the right only once. Hence, $\mathbf{m}_{t_c}$ should be the $\frac{t_c-1}{2}$th cyclic shift to the right of $\mathbf{m}_1$, which is

$$\mathbf{m}_{t_c} = [\underbrace{-1\ -1\ \ldots\ -1}_{\frac{t_c-1}{2}}\ \underbrace{1\ 1\ \ldots\ 1}_{\frac{t_c+3}{2}}\ -1] \tag{3.18}$$

and $\mathbf{m}_{t_c+1}$ should be the $\frac{t_c-1}{2}$th cyclic shift to the right of $\mathbf{m}_2$, that is:

$$\mathbf{m}_{t_c+1} = [\underbrace{1\ 1\ \ldots\ 1}_{\frac{t_c+1}{2}}\ \underbrace{-1\ -1\ \ldots\ -1}_{\frac{t_c+1}{2}}\ 1]. \tag{3.19}$$

Therefore,

$$\begin{aligned} g_{i+1}^{t_c+1} &= \mathbf{m}_{t_c+1}\mathbf{g}_i \\ &= 2g_i^{\frac{t_c+1}{2}} - \mathbf{m}_{t_c}\mathbf{g}_i \\ &= 2g_i^{\frac{t_c+1}{2}} - g_{i+1}^{t_c} \end{aligned} \tag{3.20}$$

As for $g_{i+1}^{t_c+2}$, it can be shown that

$$\mathbf{m}_{t_c+2} = [\underbrace{-1\ -1\ \ldots\ -1}_{\frac{t_c+1}{2}}\ \underbrace{1\ 1\ldots\ 1}_{\frac{t_c+3}{2}}] \tag{3.21}$$

and

$$\begin{aligned} g_{i+1}^{t_c+2} &= \mathbf{m}_{t_c+2}\mathbf{g}_i \\ &= 2g_i^{t_c+2} - \mathbf{m}_{t_c+1}\mathbf{g}_i \\ &= 2g_i^{t_c+2} - g_{i+1}^{t_c+1} \end{aligned} \tag{3.22}$$

(3.20) and (3.22) are in agreement with (3.8). This completes the proof of Theorem 4.1.

Q. E. D.

Corollary 3.2.1 (STRUCTURE OF THE ROWS OF M) *The j-th row of* M *has the following form:*

$$\overbrace{-1\ -1\ \ldots\ -1}^{\frac{t-1}{2}}\ \overbrace{1\ 1\ \ldots\ 1}^{\frac{t+1}{2}}\ \overbrace{-1\ -1\ \ldots\ -1}^{\frac{t-1}{2}} \tag{3.23}$$

if j is odd, and

$$\overbrace{1\ 1\ \ldots\ 1}^{\frac{j}{2}}\ \overbrace{-1\ -1\ \ldots\ -1}^{\frac{t-1}{2}}\ \overbrace{1\ 1\ \ldots\ 1}^{\frac{t-j+1}{2}} \tag{3.24}$$

if j is even.

Corollary 3.2.2 (STRUCTURE OF THE COLUMNS OF M) *(a) The first column of* M *has the following structure:*

(i) *Its first element is 1;*

(ii) *The values of its remaining elements alternate between* 1 *and* -1.

(b) For $l = 2, 3, \ldots, t$, the l-th column is the second cyclic shift to the bottom of the $(l-1)$-th column. It follows that the first column is also the second cyclic shift to the bottom of the last column.

Proof of Corollary 3.2.2: Consider the first column of M. (3.23) and (3.24) indicate that the first element has value -1 when the number of the row is odd, except the first row; and the first element has the value 1 when the number of the row is even. This is because there are $\frac{t-1}{2}$ -1s in a row, and when $3 \leq j \leq t$, the -1 part at the beginning of the row always exists (since $1 \leq \frac{j-1}{2} \leq \frac{t-1}{2}$). Therefore, the first column of M is of the form:

$$[1\ \ 1\ -1\ \ 1\ -1\ \ 1 \ldots -1\ \ 1\ -1]^T \tag{3.25}$$

One important point to note is the relationship between j-th row and $(j+1)$-th row. When j is odd, all the elements of the $(j+1)$-th row are of the opposite sign of the j-th row, except the $(\frac{j+1}{2})$-th element (which is 1 in both rows). Since $j \leq t$, so $\frac{j+1}{2} \leq \frac{t+1}{2}$. That is to say, the $(\frac{j+1}{2})$-th element is located in the left part of M. It is easy to verify that a similar relation exists when j is even, but with the $(\frac{j+t+1}{2})$-th element having the

same sign + in both rows and this $(\frac{j+t+1}{2})$-th element can only be located in the right part of M.

Suppose r is the sequential number of a column. When

$$r < \frac{t+1}{2},$$

from the above discussion, we know that the column has the form:

$$[1\ -1\ 1\ \ldots\ 1\ -1\ \underbrace{1}_{2r-1}\ \underbrace{1}_{2r}\ -1\ 1\ \ldots\ -1\ 1\ -1]^T \tag{3.26}$$

This is because the $(\frac{j+1}{2})$-th element of the j-th and $(j+1)$-th rows are both $+1$ (where j is the odd sequential number of rows), and this $(\frac{t+1}{2})$-th element is located in the left part of M. From

$$\frac{j+1}{2} = r$$

we get:

$$j = 2r - 1.$$

That is, two adjacent $+1$ s appear at the positions of $(2r-1)$ and $(2r)$ in the r-th column. So the r-th column is the second cyclic shift to the bottom of the $(r-1)$-th column.

For the case $r = \frac{t+1}{2}$ and

$$\frac{j-1}{2} < \frac{t+1}{2},$$

according to (3.23), the $(\frac{t+1}{2})$-th elements of the odd rows always have the value $+1$. Similarly, because

$$\frac{j+1}{2} < \frac{t+1}{2}$$

for $j \leq (t-1)$, from (3.24), the $(\frac{t+1}{2})$-th elements of the even rows always have the value -1. Therefore, the $(\frac{t+1}{2})$-th column has the form

$$[1\ \ -1\ \ 1\ \ \ldots\ \ -1\ \ 1\ \ \ -1\ \ 1]^T$$

which is also the second cyclic shift to the bottom of the $(\frac{t-1}{2})$-th column.

Similarly, when

$$r > \frac{t+1}{2},$$

the column has the form:

$$[-1 \;\; 1 \;\; -1 \;\; \ldots \;\; 1 \;\; -1 \underbrace{1}_{2r-t-1} \overbrace{1}^{2r-t} -1 \;\; 1 \;\; \ldots \;\; -1 \;\; 1]^T. \tag{3.27}$$

This is because the $(\frac{j+t+1}{2})$-th element of the j-th rows are both $+1$ (where j is the even sequential number of rows), and this $(\frac{j+t+1}{2})$-th element is located in the right part of M. From

$$\frac{j+t+1}{2} = r$$

we have:

$$j = 2r - t - 1.$$

That is, two adjacent $+1$s appear at the positions of $(2r-t-1)$ and $(2r)$ in the r-th column. So the r-th column is the second cyclic shift to the bottom of the $(r-1)$-th column.

For the last column, since $r = t$, it has the form

$$[-1 \;\; 1 \;\; -1 \;\; \ldots \;\; 1 \;\; -1 \;\; 1 \;\; 1]^T.$$

From this form, we can see that the first column is the second cyclic shift to the bottom of the last column.

Q. E. D.

Theorem 3.2 (RELATION BETWEEN $\mathbf{g}_i$ and $\mathbf{g}_{i+u}$) *If the error patterns of $U^i e(x)$ and $U^{i+u}e(x)$ are of γ-type, then their corresponding gap length vectors $\mathbf{g}_i$ and $\mathbf{g}_{i+u}$ are related by the following mapping*

$$\mathbf{g}_{i+u} = \mathbf{M}^u \cdot \mathbf{g}_i$$

where the matrix $\mathbf{M}^u$ is defined as

$$\mathbf{M}^u = \underbrace{\mathbf{M} \times \mathbf{M} \times \ldots \times \mathbf{M}}_{u},$$

and has the following properties:

1. *The sum of all the elements in a row of $\mathbf{M}^u$ is 1;*

2. *All elements in* $\mathbf{M}^u$ *have only two integral values:* $a^{(u)}$ *and* $-b^{(u)}$, *where* $a^{(u)} > 0$, $b^{(u)} > 0$;

3. $a^{(u)} + b^{(u)} = 2^u$;

4. *Every row in* $\mathbf{M}^u$ *is the* $n_b^{(u)}$*-th cyclic shift to the left of the previous row, and the first row can be viewed as the* $n_b^{(u)}$*th cyclic shift to the left of the last row, where* $n_b^{(u)}$ *is the number of elements with value* $-b^{(u)}$ *in a row.*

Proof of Theorem 3.2: We prove this theorem by induction.

For $\mathbf{M}$, according to Theorem 3.1, it is shown that these properties are satisfied. In this case, we have

$$\begin{aligned} a^{(1)} &= -b^{(1)} \\ &= 1; \\ n_a^{(1)} &= \frac{t+1}{2}; \\ n_b^{(1)} &= \frac{t-1}{2}. \end{aligned}$$

Assume that for $\mathbf{M}^{u-1}$, these four properties are satisfied, now we prove that for $\mathbf{M}^u$, all four properties are also satisfied.

Suppose $\mathbf{M}^{u-1}$ has the form:

$$\mathbf{M}^{u-1} = \begin{bmatrix} a^{(u-1)} & a^{(u-1)} & \dots & a^{(u-1)} & -b^{(u-1)} & -b^{(u-1)} & \dots & -b^{(u-1)} \\ & & & \ddots & & & & \\ -b^{(u-1)} & -b^{(u-1)} & \dots & -b^{(u-1)} & a^{(u-1)} & a^{(u-1)} & \dots & a^{(u-1)} \end{bmatrix}_{t \times t}$$

and the numbers of elements with value $a^{(u-1)}$ and $b^{(u-1)}$ are $n_a^{(u-1)}$ and $b^{(u-1)}$ respectively. We partition the proof for two cases: even $n_a^{(u-1)}$ and odd $n_a^{(u-1)}$.

Case 1, even $n_a^{(u-1)}$:

Since

$$\mathbf{M}^u = \mathbf{M}^{u-1} \cdot \mathbf{M},$$

it follows that

$$\mathbf{M}^u = \begin{bmatrix} a^{(u-1)} & a^{(u-1)} & \dots & a^{(u-1)} & -b^{(u-1)} & -b^{(u-1)} & \dots & -b^{(u-1)} \\ & & & \ddots & & & & \\ -b^{(u-1)} & -b^{(u-1)} & \dots & -b^{(u-1)} & a^{(u-1)} & a^{(u-1)} & \dots & a^{(u-1)} \end{bmatrix} \cdot$$

$$\begin{bmatrix} 1 & 1 & \dots & 1 & -1 & -1 & \dots & -1 \\ & & & \ddots & & & & \\ -1 & -1 & \dots & -1 & 1 & 1 & \dots & 1 \end{bmatrix}_{t \times t} \tag{3.28}$$

Because the first column of $\mathbf{M}$ takes the form of $[1 \;\; 1 \;\; -1 \;\; \dots \;\; 1 \;\; -1]^T$, and each column of $\mathbf{M}$ is the second cyclic shift to the bottom of the previous column, it is easy to verify that when $n_a^{(u-1)}$ is even, the first row of $\mathbf{M}^u$ is:

$$[\underbrace{a^{(u)} \;\; a^{(u)} \;\; \dots \;\; a^{(u)}}_{n_a^{(u)}} \;\; \underbrace{-b^{(u)} \;\; -b^{(u)} \;\; \dots \;\; -b^{(u)}}_{n_b^{(u)}}] \tag{3.29}$$

with

$$\begin{aligned} a^{(u)} &= 2a^{(u-1)} + b^{(u-1)} \\ b^{(u)} &= b^{(u-1)} \\ n_a^{(u)} &= \tfrac{n_a^{(u-1)}}{2} \\ n_b^{(u)} &= n_b^{(u-1)} + \tfrac{n_a^{(u-1)}}{2}. \end{aligned}$$

By adding all the elements in the first row of $\mathbf{M}^u$, we obtain:

$$\begin{aligned} \sum_{j=1}^{j=t} m_{1j} &= n_a^{(u)} a^{(u)} + n_b^{(u)} b^{(u)} \\ &= \frac{n_a^{(u-1)}}{2}(2a^{(u-1)} + b^{(u-1)}) + (n_b^{(u-1)} + \frac{n_a^{(u-1)}}{2})(-b^{(u-1)}) \\ &= n_a^{(u-1)} a^{(u-1)} - n_b^{(u-1)} b^{(u-1)} \\ &= 1. \end{aligned} \tag{3.30}$$

where m_{1j} is the j-th element in the first row of $\mathbf{M}^u$. Notice that the two values in $\mathbf{M}^u$ are $a^{(u)}$ and $(-b^{(u)})$. Hence,

$$\begin{aligned} a^{(u)} + b^{(u)} &= (2a^{(u-1)} + b^{(u-1)}) + b^{(u-1)} \\ &= 2(a^{(u-1)} + b^{(u-1)}). \end{aligned} \tag{3.31}$$

Because we assumed that

$$a^{(u-1)} + b^{(u-1)} = 2^{u-1},$$

is true, it follows that

$$a^{(u)} + b^{(u)} = 2^u.$$

So far Properties 1, 2 and 3 have been proven. Next, we prove that property 4 is also satisfied.

Suppose $m_i^{(u)}$ is the i-th row in $\mathbf{M}^u$. Since $m_i^{(u-1)}$ is the $n_b^{(u-1)}$-th cyclic shift to the left of $m_{i-1}^{(u-1)}$, $m_i^{(u-1)}$ is also the $n_a^{(u-1)}$-th cyclic shift to the right of $m_{i-1}^{(u-1)}$.

From (3.28), we see that

$$m_i^{(u)} = m_i^{(u-1)}\mathbf{M}.$$

If we designate $(m_{i-1}^{(u-1)})|_{r \to n_a^{(u-1)}}$ as the $n_a^{(u-1)}$-th cyclic shift to the right of $m_{i-1}^{(u-1)}$ and $\mathbf{M}|_{t \to n_a^{(u-1)}}$ as the $n_a^{(u-1)}$-th cyclic shift of the rows to the top of $\mathbf{M}$, we can rewrite the above equation as:

$$\begin{aligned} m_i^{(u)} &= (m_{i-1}^{(u-1)})|_{r \to n_a^{(u-1)}}\mathbf{M} \\ &= m_{i-1}^{(u-1)}\mathbf{M}|_{t \to n_a^{(u-1)}}. \end{aligned} \tag{3.32}$$

Since each column of $\mathbf{M}$ is the second cyclic shift to the bottom of the previous column, the effect of cyclic shifting the rows of $\mathbf{M}$ to the top $n_a^{(u-1)}$ times is the same as cyclic shifting the columns of $\mathbf{M}$ to the right $\frac{n_a^{(u-1)}}{2}$ times. That is to say, $m_i^{(u)}$ is the $\frac{n_a^{(u-1)}}{2}$-th cyclic shift to the right of $m_{i-1}^{(u)}$, which is also the $(t - \frac{n_a^{(u-1)}}{2})$-th cyclic shift to the left of $m_{i-1}^{(u)}$. Notice that

$$\begin{aligned} n_b^{(u)} &= t - n_a^{(u)} \\ &= t - \frac{n_a^{(u-1)}}{2}, \end{aligned} \tag{3.33}$$

therefore $m_i^{(u)}$ is the $n_b^{(u)}$-th cyclic shift to the left of $m_{i-1}^{(u)}$. This satisfies the Property 4.

Case 2, odd $n_a^{(u-1)}$:

Similarly, since the first column of $\mathbf{M}$ takes the form of $[1 \;\; 1 \;\; -1 \;\; \ldots \;\; 1 \;\; -1]^T$, and each column of $\mathbf{M}$ is the second cyclic shift to the bottom of the previous column, it is easy to verify that when $n_a^{(u-1)}$ is odd, the first row of $\mathbf{M}^u$ is:

$$[\underbrace{a^{(u)} \;\; a^{(u)} \;\; \ldots \;\; a^{(u)}}_{n_a^{(u)}} \;\; \underbrace{-b^{(u)} \;\; -b^{(u)} \;\; \ldots \;\; -b^{(u)}}_{n_b^{(u)}}] \tag{3.34}$$

with

$$\begin{aligned} a^{(u)} &= a^{(u-1)} \\ b^{(u)} &= a^{(u-1)} + 2b^{(u-1)} \\ n_a^{(u)} &= \tfrac{t+n_a^{(u-1)}}{2} \\ n_b^{(u)} &= \tfrac{n_b^{(u-1)}}{2}. \end{aligned}$$

By adding all the elements in the first row of $\mathbf{M}^u$, we get the sum of it as:

$$\begin{aligned} \sum_{j=1}^{j=t} m_{1j} &= n_a^{(u)}a^{(u)} + n_b^{(u)}b^{(u)} \\ &= (\frac{t+n_a^{(u-1)}}{2})(a^{(u-1)}) + (\frac{n_b^{(u-1)}}{2})(-a^{(u-1)} - 2b^{(u-1)}) \\ &= n_a^{(u-1)}a^{(u-1)} - n_b^{(u-1)}b^{(u-1)} \\ &= 1. \end{aligned} \tag{3.35}$$

Notice that the two values in $\mathbf{M}^u$ are $a^{(u)}$ and $(-b^{(u)})$, so that

$$\begin{aligned} a^{(u)} + b^{(u)} &= a^{(u-1)} + (a^{(u-1)} + 2b^{(u-1)}) \\ &= 2(a^{(u-1)} + b^{(u-1)}), \end{aligned} \tag{3.36}$$

therefore,

$$a^{(u)} + b^{(u)} = 2^u.$$

So far, we have proven that the Properties 1, 2 and 3 are satisfied. Now we prove that Property 4 is also satisfied.

If we designate $(m_{i-1}^{(u-1)})|_{l \to n_b^{(u-1)}}$ as the $n_b^{(u-1)}$-th cyclic shift to the left of $m_{i-1}^{(u-1)}$ and $\mathbf{M}|_{b \to n_b^{(u-1)}}$ as the $n_b^{(u-1)}$-th cyclic shift to the bottom of $\mathbf{M}$, we can rewrite the equation

$$m_i^{(u)} = m_i^{(u-1)} \times \mathbf{M}$$

as

$$\begin{aligned} m_i^{(u)} &= (m_{i-1}^{(u-1)})|_{l \to n_b^{(u-1)}} \times \mathrm{M} \\ &= m_{i-1}^{(u-1)} \times \mathrm{M}|_{b \to n_b^{(u-1)}} \end{aligned} \tag{3.37}$$

Since each column of M is the second cyclic shift to the bottom of the previous column, the effect of cyclic shifting the rows of M to the bottom $n_b^{(u-1)}$ times is the same as cyclic shifting the columns of M to the left $\frac{n_b^{(u-1)}}{2}$ times. That is to say, $m_i^{(u)}$ is the $\frac{n_b^{(u-1)}}{2}$-th cyclic shift to the left of $m_{i-1}^{(u)}$. Notice that

$$\begin{aligned} n_b^{(u)} &= t - n_a^{(u)} \\ &= \frac{n_b^{(u-1)}}{2}, \end{aligned} \tag{3.38}$$

therefore, $m_i^{(u)}$ is the $n_b^{(u)}$-th cyclic shift to the left of $m_{i-1}^{(u)}$. This satisfies Property 4.

Q. E. D.

3.3 Capability of Multiple-step *(T, U)* Permutation Decoding of Cyclic Codes

Using the theorems developed in Section 3.2, the capability of multiple-step $(T,\ U)$ permutation decoding is determined. The results are given in the following theorems.

Theorem 3.3 *The* $(n,\ k_e,\ 2t+1)$ *codes with* $n = t(k_e - 2^{s-1}+1)+2^{s-1}$, $k_e \geq t(2^{s-2}-1)+1$ *for* $t = 5$ *and* $k_e \geq t(2^{s-2}-1) - 2^{s-2}+3$ *for* $t \geq 7$, *are s-step PD.*

Proof of Theorem 3.3: First, we prove the existence of M^u.

The mean gap size of the error patterns with t errors in a $(n,\ k_e,\ 2t+1)$ code is:

$$\begin{aligned} m &= \frac{n-t}{t} \\ &= k_e - 2^{s-1} + \frac{2^{s-1}}{t}. \end{aligned} \tag{3.39}$$

So that,

$$k_e - m = 2^{s-1} - \frac{2^{s-1}}{t}.$$

For $t = 5$,

$$k_e \geq t(2^{s-2} - 1) + 1,$$

therefore,

$$\begin{aligned} \frac{k_e}{2} - (k_e - m) &\geq 2^{s-3}(t - 4 + \frac{4}{t}) - \frac{t}{2} + \frac{1}{2} \\ &> 0. \end{aligned} \tag{3.40}$$

When $s \geq 3$ and $t = 5$, from the above equation, we have:

$$\frac{k_e}{2} > k_e - m. \tag{3.41}$$

Similarly, for $t \geq 7$, since

$$k_e \geq t(2^{s-2} - 1) - 2^{s-2} + 3,$$

therefore,

$$\begin{aligned} \frac{k_e}{2} - k_e - m &\geq 2^{s-3}(t - 5 + \frac{4}{t}) - \frac{t}{2} + \frac{3}{2} \\ &> 0. \end{aligned} \tag{3.42}$$

When $s \geq 3$ and $t \geq 7$, from the above equation, we have:

$$\frac{k_e}{2} > k_e - m. \tag{3.43}$$

Now, suppose that $\mathbf{g}_0$ is a gap-length vector of $e(x)$ and

$$\mathbf{g}_{s-1} = \mathbf{M}^{s-1} \cdot \mathbf{g}_0$$

is the corresponding gap-length vector of $U^{s-1}e(x)$. Theorem 3.2 shows that elements of $\mathbf{g}_{s-1}$ are linear combinations of the elements in $\mathbf{g}_0$. We can view it as there are two parts, one consisting $n_a^{(s-1)}$ consecutive positions and the other consisting $n_b^{(s-1)}$ consecutive positions. Therefore,

$$\begin{aligned} g_{s-1}^1 &= a^{(s-1)} \sum_{j=1}^{n_a^{(s-1)}} g_0^j - b^{(s-1)} \sum_{j=n_a^{(s-1)}}^{t} g_0^j \\ &= a^{(s-1)}(n_a^{(s-1)}m + \Delta^{(s-1)}) - b^{(s-1)}(n_b^{(s-1)}m - \Delta^{(s-1)}) \\ &= m + 2^{s-1}\Delta^{(s-1)}, \end{aligned} \tag{3.44}$$

where m is the mean value of the gaps and

$$\Delta^{(s-1)} = \sum_{j=1}^{n_a^{(s-1)}} g_0^j - n_a^{(s-1)} m.$$

Note that in (3.44) we used two relations:

$$n_a^{(s-1)} a^{(s-1)} - n_b^{(s-1)} b^{(s-1)} = 1,$$

and

$$a^{(s)} + b^{(s)} = 2^s.$$

Similarly, g_{s-1}^j can be obtained in the same way as (3.44) but with cyclic shifts of the sums to the left by $(j \cdot n_b^{(s-1)})$ times.

Now, we prove that there is at least one gap of length

$$\begin{aligned} g_{(s-1)}^j &= m + 2^{s-1}\Delta^{(s-1)} \\ &\geq k_e. \end{aligned} \tag{3.45}$$

When the condition $n = t(k_e - 2^{s-1} + 1) + 2^{s-1}$, $k_e \geq t(2^{s-2} - 1) + 1$ for $t = 5$ and $k_e \geq t(2^{s-2} - 1) - 2^{s-2} + 3$ for $t \geq 7$ are satisfied, from (3.41) or (3.43), we have

$$\frac{k_e}{2} > k_e - m.$$

In other words, there always exists a $\Delta^{(s-1)}$, which makes the code decodable but satisfies

$$m - 2^{s-1}\Delta^{(s-1)} > 0$$

(note that $m > \frac{k_e}{2}$). From the proof of Lemma 4.2 in [33], it is known that $\mathbf{M}^2$ exists. According to the previous discussion, the mapping form of $\mathbf{M}^u$ exists for $u \geq 3$. Therefore, $\mathbf{M}^u$ exists for all u.

Next, we prove that when the mapping form of $\mathbf{M}^s$ exists, if a $(n_{s+1}, k, 2t+1)$ code is not $(s+1)$-step permutation decodable, then the $(n'_s, k, 2t+1)$ code with

$$n'_s = n_{s+1} + 2^{s-1}$$

is not s-step permutation decodable.

Note that the error patterns take the mapping form of $\mathbf{M}^s$ from the R_0-domain to the R_s-domain is the most stable pattern. This is because property 1 of Theorem 3.2 can be applied. Since all the elements in $\mathbf{M}$ have the values $+1$ and -1, each element of $\mathbf{g}_s$ is a linear combination of the t elements of $\mathbf{g}_{s-1}$. Actually, it equals the sum of $(\frac{t+1}{2})$ elements minus the sum of other $(\frac{t-1}{2})$ elements of $\mathbf{g}_{s-1}$. Therefore, the elements of $\mathbf{g}_s$ intend to duplicate the elements of $\mathbf{g}_{s-1}$. The extreme case is that all the t-gaps in the R_{s-1}-domain equals to the same average value m, then after each $\mathbf{M}$ mapping, the gaps in the R_s-domain remains the same as gaps in the R_{s-1}-domain. So if $m < k$, the code with this kind of pattern is not permutation decodable, no matter how many U permutation steps are used. The error pattern that needs the maximum number of steps to decode is the error pattern which is not decodable in the R_0-domain and is most stable after each U permutation. That is to say, if a code is not decodable, and the error pattern takes the mapping form of $\mathbf{M}$ from R_{s-1}-domain to R_s-domain, then there must be at least one error pattern of this form which makes the code undecodable.

Now, we divide our proof into two parts:

1. When $n_a^{(s-1)}$ is even;

2. When $n_a^{(s-1)}$ is odd.

Case 1, even $n_a^{(s-1)}$:

The gap vector in the R_s-domain has the form:

$$\mathbf{g}_s = \mathbf{M}^s \cdot \mathbf{g}_0$$

where $\mathbf{g}_s$ and $\mathbf{g}_0$ are the gap vectors in the R_s and R_0 domains and $\mathbf{M}^s$ has the form

$$\mathbf{M}^s = \begin{bmatrix} a^{(s)} & a^{(s)} & \dots & a^{(s)} & -b^{(s)} & -b^{(s)} & \dots & -b^{(s)} \\ & & & \ddots & & & & \\ -b^{(s)} & -b^{(s)} & \dots & -b^{(s)} & a^{(s)} & a^{(s)} & \dots & a^{(s)} \end{bmatrix}_{t \times t} \tag{3.46}$$

Suppose that g_s^1 has the largest value (for other gaps in the R_s-domain, the analysis is the same), since it is not decodable in the R_s-domain, $g_s^1 < k_c$.

Now consider the code $(n'_s, k_e, 2t+1)$ with

$$n'_s = n_{s+1} + 2^{s-1}t.$$

Since n'_s increases $2^{s-1}t$ relative to n_{s+1}, the effect is that each gap in the $[g_{s-1}]$ increases by 2^{s-1} and the mean value becomes

$$m' = m + 2^{s-1}.$$

Because $n_a^{(s-1)}$ in $\mathbf{M}^{s-1}$ is even, $\mathbf{M}^{s-1}$ has the form:

$$\mathbf{M}^{s-1} = \begin{bmatrix} a^{(s-1)} & a^{(s-1)} & \dots & a^{(s-1)} & -b^{(s-1)} & -b^{(s-1)} & \dots & -b^{(s-1)} \\ & & & \ddots & & & & \\ -b^{(s-1)} & -b^{(s-1)} & \dots & -b^{(s-1)} & a^{(s-1)} & a^{(s-1)} & \dots & a^{(s-1)} \end{bmatrix}_{t \times t} \tag{3.47}$$

with $n_a^{(s-1)} = 2n_a^{(s)}$, and $n_b^{(s-1)} = n_b^{(s)} - n_a^{(s)}$.

Suppose the largest gap in $[g'_{s-1}]$ is $g'^{(1)}_{s-1}$, then $g'^{(1)}_{s-1}$ can be expressed as:

$$\begin{aligned} g'^{(1)}_{s-1} &= n_a^{(s-1)} m' + \Delta^{(s-1)} - (n_b^{(s-1)} m' - \Delta^{(s-1)}) \\ &= n' + 2^{s-1}\Delta^{(s-1)} \\ &= m + 2^{s-1} + 2^{s-1}\Delta^{(s-1)}. \end{aligned} \tag{3.48}$$

If

$$g'^{(1)}_{s-1} \leq g^1_s, \tag{3.49}$$

the codes $(n'_s, k_e, 2t+1)$ is not $(s-1)$-step decodable. (3.48) can be rewritten as:

$$g^1_s - g'^{(1)}_{s-1} = 2^{(s-1)}(2\Delta^{(s)} - \Delta^{(s-1)} - 1).$$

Therefore, if

$$2\Delta^{(s)} - \Delta^{(s-1)} - 1 \geq 0, \tag{3.50}$$

then the code $(n'_s, k_e, 2t+1)$ is not $(s-1)$-step permutation decodable.

Since $n_a^{(s-1)} = 2n_a^{(s)}$ and $n_b^{(s-1)} = n_b^{(s)} - n_a^{(s)}$,

$$\begin{aligned} g_{s-1}^{\prime(1)} &= a^{(s-1)}\left(2n_a^{(s)}m + \left(1 - \frac{n_a^{(s)}}{n_b^{(s)}}\right)\Delta^{(s)}\right) - b^{(s-1)}\left(\left(n_b^{(s)} - n_a^{(s)}\right)m - \left(1 - \frac{n_a^{(s)}}{n_b^{(s)}}\right)\Delta^{(s)}\right) \\ &= m + 2^{s-1}\left(1 - \frac{n_a^{(s)}}{n_b^{(s)}}\right)\Delta^{(s)}. \end{aligned} \tag{3.51}$$

Therefore,

$$\Delta^{(s-1)} = \left(1 - \frac{n_a^{(s)}}{n_b^{(s)}}\right) \cdot \Delta^{(s)} \tag{3.52}$$

and the (3.50) can be rewritten as

$$\left(1 + \frac{n_a^{(s)}}{n_b^{(s)}}\right)\Delta^{(s)} \geq 1 \tag{3.53}$$

Notice that we used the mean value of gaps covered by $n_a^{(s-1)}$ but not by $n_a^{(s)}$ as

$$m = n_a^{(s)}m - \frac{n_a^{(s)}}{n_b^{(s)}}\Delta^{(s)}$$

in the derivation of (3.51). Actually, we can look at both sides of $n_a^{(s)}$ positions which is covered by $n_b^{(s)}$ and choose the smaller one. But for simplicity, we use mean value instead.

Since

$$n_a^{(s)} + n_b^{(s)} = t$$

we rewrite (3.53) as

$$\Delta^{(s)} \cdot t \geq n_b^{(s)}. \tag{3.54}$$

For $\Delta^{(s)} \geq \frac{n_b^{(s)}}{t}$, the condition of (3.50) is always satisfied. Because

$$\Delta^{(s)} = \sum_{j=1}^{n_a^{(s)}} g_s^j - n_a^{(s)}m, \tag{3.55}$$

we get

$$\Delta^{(s)} - \frac{n_b^{(s)}}{t} = \sum_{j=1}^{n_a^{(s)}} g_s^j - n_a^{(s)}m - 1 + \frac{n_a^{(s)}}{t}. \tag{3.56}$$

Now let us look at the case when the above equation has the smallest value. Suppose that $n_a^{(s)} = 1$ and with the gap length $k_c - y$, where y is an integer of even value; and $n_b^{(s)} = t - 1$ with the gaps take the gap length $k_e - y$ and $k_e - y - 2$ alternatively. Note that other

patterns will result in consecutive gaps with gap length $k_e - y$ or $k_e - y - 2$, which makes $\Delta^{(s)}$ have larger value. (3.56) can be rewritten as:

$$\begin{aligned}\Delta^{(s)} - \frac{n_b^{(s)}}{t} &= (k - y + 2) - \frac{n - t}{t} - 1 + \frac{1}{t} \\ &= k - y + 2 - \frac{\frac{t-1}{2}(k - y) + \frac{t+1}{2}(k - y + 2)}{t} - 1 + \frac{1}{t} \\ &= 0 \end{aligned} \tag{3.57}$$

Therefore, (3.54) holds, so does (3.50).

Similarly, **Case 2** can proven. The only difference in the proof is that $2n_b^{(s)} = n_b^{(s-1)}$ is used instead of $2n_a^{(s)} = n_a^{(s-1)}$, as in **Case 1**.

According to Theorem 14 in reference [33], if $(n'_{s-1},\, k_o,\, 2t + 1)$ is not $(s - 1)$-step permutation decodable, then $(n_{s-1},\, k_o,\, 2t + 1)$ is not $(s - 1)$-step permutation decodable.

For $s = 3$, it has been proven that the result is true [33]. Now suppose for s-steps, the result is true, but for $(s + 1)$-steps, it is false.

Since the code with

$$n_{s+1} = t(k_e - 2^s + 1) + 2^s$$

are not $(s+1)$-step permutation decodable, according to the previous discussion, the codes with

$$\begin{aligned} n_s &= n_{s+1} + 2^{s-1}(t - 1) \\ &= t(k_e - 2^{s-1} + 1) + 2^{s-1} \end{aligned} \tag{3.58}$$

are not s-step permutation decodable. But we have supposed that the codes with n_s are s-step PD, so the codes with n_{s+1} are $(s + 1)$-step permutation decodable.

Q. E. D.

Theorem 3.4 *The* $(n,\, k_o,\, 2t+1)$ *codes with* $n = t(k_o - 2^{s-1} + 2) + 2^{s-1}$, $k_o \geq t(2^{s-2} - 1)$ *for* $t = 5$ *and* $k_o \geq t(2^{s-2} - 1) - 2^{s-2} + 2$ *for* $t \geq 7$, *are s-step PD.*

Proof of Theorem 3.4: Suppose $k_e = k_o + 1$, then all the conditions in Theorem 3.4 will be equivalent to that in Theorem 3.3. Because under these conditions the code $(n, k_e, 2t + 1)$ is s-step permutation), the code $(n, k_e - 1, 2t + 1)$ is also s-step PD. Noting that $k_o = k_e - 1$, therefore, Theorem 3.4 holds.

Q. E. D.

The following Theorem and its Corollary show that if the condition $k_o \geq t(2^{s-2} - 1) - 2^{s-2} + 2$ in Theorem 3.4 is not satisfied, than the code is not permutation decodable.

Theorem 3.5 *The* $(n, k_o, 2t + 1)$ *codes with* $n = t(k_o - 2^{s-1} + 2) + 2^{s-1}$, $k_o = t(2^{s-2} - 1) - 2^{s-2}$ *for* $t \geq 7$, *are not s-step PD.*

Proof of Theorem 3.5: When

$$k_o = t(2^{s-2} - 1) - 2^{s-2}$$

The code length n becomes

$$n = k_o(t - 2)\,.$$

With reference to the Theorem 13 in [33], the codes are not s-step PD.

Q. E. D.

Corollary 3.3.1 *The* $(n, k_e, 2t + 1)$ *codes with* $n = t(k_e - 2^{s-1} + 1) + 2^{s-1}$, $k_e = t(2^{s-2} - 1) - 2^{s-2} + 1$ *for* $t \geq 7$, *are not s-step PD.*

This is apparent because if we substitute $k_e = k_o + 1$ into Theorem 3.5, we know there exists no error free gap with gap-length larger or equal to $k_e - 1$.

3.4 Bounds on the Code Rate

Lower bounds have been found on the code length n of the s-step permutation decodable cyclic codes. For a better understanding of the general relationship between the number

of U permutations and the code parameters n, k and t of the corresponding permutation decodable cyclic codes, we present bounds of the code rate R_c for binary $(n,\ k,\ 2t+1)$ permutation decodable codes in the following two theorems:

Theorem 3.6 *The code rate R_c of s-step PD codes with $k > k^*$ and $t = 5$ is upper bounded by*

$$\frac{1}{t-2+\frac{2}{t}},$$

where $k_o^* = t \cdot (2^{s-2}-1)-2$ and $k_e^* = t \cdot (2^{s-2}-1)-1$.

Proof of Theorem 3.6: Here, we just give the proof for k_o, the proof for k_e is similar to k_o. Suppose the code length given in Theorem 3.3 and 3.4 are used, then

$$\begin{aligned} \frac{\partial R_c}{\partial k_o} &= \frac{t+2^{s-2}(1-t)}{n^2} \\ &< 0 \end{aligned} \tag{3.59}$$

That is to say, R_c decreases with k_o. On the other hand, at the point where $k_o = k^*$, the relation between R_c and s is given by the following equation:

$$\begin{aligned} \left.\frac{\partial R_c}{\partial s}\right|_{k_o=k_o^*} &= \frac{1}{t-2+\frac{2^{s-1}-4}{t(2^{s-2}-1)-2}} \\ &> 0 \end{aligned} \tag{3.60}$$

That is, R_c increases with s, and the limit is:

$$\lim_{s\to\infty} R_c\Big|_{k_o=k_o^*} = \frac{1}{t-2+\frac{2}{t}} \tag{3.61}$$

Therefore, for $k_o > k^*$ and $s \geq 4$, the inequality

$$R_c < \frac{1}{t-2+\frac{2}{t}}. \tag{3.62}$$

holds.

Q. E. D.

Table 3.1: A comparison between $\frac{1}{t}$ and R'

t	$\frac{1}{t}$	R'	$\frac{R'-\frac{1}{t}}{\frac{1}{t}}$
1	1	1	0%
2	0.5	0.8	60%
3	0.3333	0.5	50%
5	0.2	0.294	47%
7	0.1429	0.2	40%
9	0.1111	0.1429	29%
11	0.0909	0.1111	22%

Theorem 3.7 *The code rate R_c of s-step PD codes with $k > k^*$ and $t \geq 7$ is less than $\frac{1}{t-2}$, where $k_o^* = t \cdot (2^{s-2} - 1) - 2^{s-2}$ and $k_e^* = t \cdot (2^{s-2} + 1) - 2^{s-2} + 1$.*

Proof of Theorem 3.7:

Similarly, since

$$\frac{\partial R_c}{\partial k} < 0$$

and

$$R_c|_{k=k^*} < \frac{1}{t-2}$$

the inequality

$$R_c < \frac{1}{t-2}$$

holds.

Q. E. D.

It would be helpful to have an idea of how large R' is compared to the limit $\frac{1}{t}$ (which is the code rate bound of the codes decodable by a basic error trapping decoder), and this provides an estimate of the efficiency of a (T, U) permutation decoder. A comparison between $\frac{1}{t}$ and R' is given in Table 3.1 and plotted in Figure 3.4.

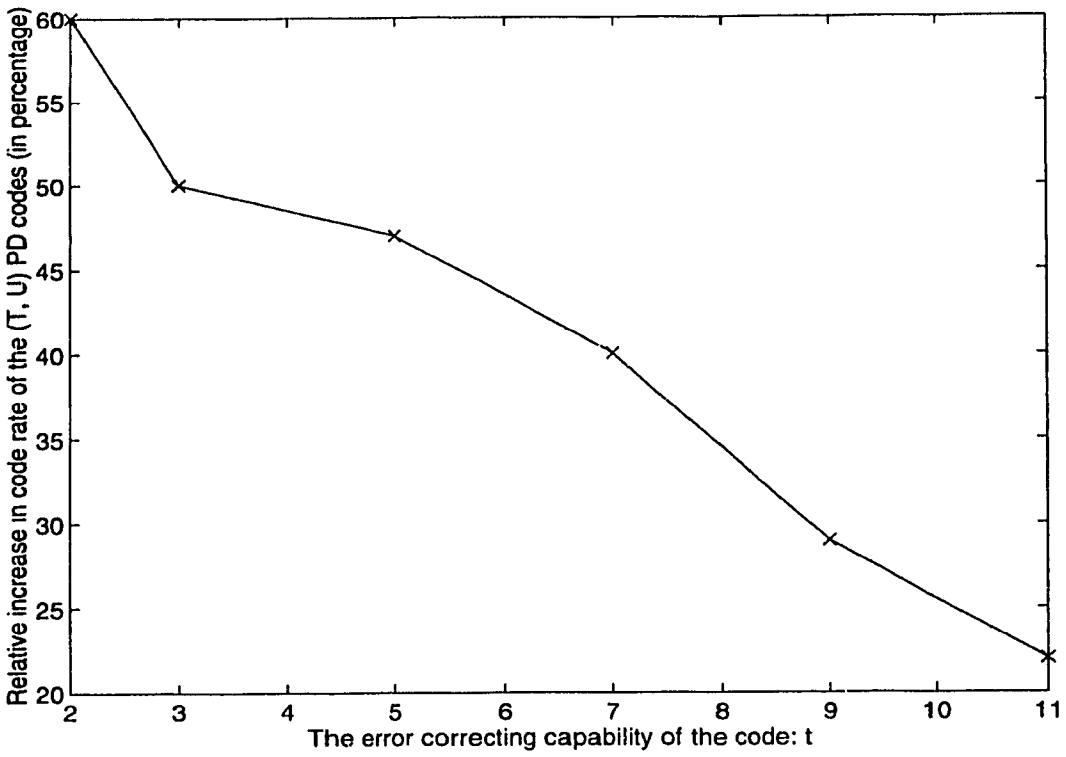


Figure 3.4: Plot of $(\frac{R'}{t-1} - 1)$ versus t.

From Figure 3.4 we can see that the (T, U) permutation decoder is most powerful for the lower error correcting capability of cyclic codes.

In the previous analysis, we have given two bounds for $t = 5$ and $t \geq 7$. They can be rewritten in the form of

$$R_c = \frac{1}{t - 2(1 - f(t))}$$

where $f()$ is a function of t, with

$$\begin{aligned} f(5) &= \frac{1}{5}, \\ f(7) &= 0. \end{aligned}$$

For $t > 7$, $f(t)$ should be a number smaller than 0 (zero). Therefore, for $t > 7$, the bound here is not a tight bound, and it is shown from the simulation results that R_c will decrease as t increases, but at a lower rate than $\frac{1}{t}$.

3.5 Characteristics of the Code Rate

One practical question in using permutation decoding is "How does R_c increases with s?".

First, we examine the rate of increase in R_c as s increases. Let R_{cs} and $R_{c(s+1)}$ denote the code rates of the s-step and $(s+1)$-step PD-codes respectively. The increase in the code rate at the s-step is

$$\Delta R_{cs} = R_{c(s+1)} - R_{cs}.$$

Since

$$\begin{aligned} \frac{\partial R_c}{\partial s} &= \frac{k_o 2^{s-2}(t-1)\ln 2}{n^2} \\ &> 0 \end{aligned} \tag{3.63}$$

and

$$\begin{aligned} \frac{\partial^2 R_c}{\partial s^2} &= \frac{k_o(k_o+1)t(t-1)2^{s-2}\ln^2 2}{n^3} \\ &> 0 \end{aligned} \tag{3.64}$$

when $k > k^*$, ΔR_{cs} increases with s. Consider the case $t = 7$ as an example. The cases of $t = 3$ and 5 are similar. For an s-step permutation,

$$k_s^* = t(2^{s-2} - 1) - 2^{s-2}. \tag{3.65}$$

Therefore

$$\begin{aligned} \Delta k_s^* &= k_{s+1}^* - k_s^* \\ &= k_s^* + t \end{aligned} \tag{3.66}$$

In other words, for each additional U permutation, k^* is increased by $(k^* + t)$, i.e., more than doubled.

On the other hand, when $k < k^*$, the code length n does not decrease at the rate $\Delta n = \Delta \cdot t$, so the increase in the code rate R_c is very slow. ΔR_{cs} increases with s, to the extent that k^* is large enough and $k \leq k^*$ is satisfied.

We call the step corresponding to the largest ΔR_{cs} the optimum permutation step which makes the largest increase in the code rate of PD codes. For a given $(n,\ k,\ 2t+1)$ code, the optimum step is the step which makes $k < k^*$ satisfied. Since k^* can be calculated from Theorems 3.6 and 3.7, which is a function of s, the concept of an optimum permutation step can be used to estimate the steps needed to decode a certain PD code.

The following example shows how ΔR_{cs} increases as the number of permutation steps s increases, and illustrates the concept of an optimum permutation step.

Example: For a given k and t of a $(T,\ U)$ permutation decodable cyclic code, find the optimum number of steps s, and the corresponding code length n.

Here, we consider two cases: Case (i): $t = 3$ and $k_o = 195$; Case (ii): $t = 3$ and $k_o = 21$. The optimum number of steps corresponding to these two cases are 8 for Case (i) and 5 for Case (ii).

The optimum value of s and corresponding code length n are also given in Tables 3.2 and 3.3 for $(n,\ 195,\ 7)$ and $(n,\ 21,\ 7)$ cyclic codes. In these tables, the third column indicates the maximum achievable code rate for s-step PD codes (R_s), the fourth column represents the improvement in the maximum achievable code rates from $(s-1)$ steps to s steps. Notice that the values in the fourth column corresponding to the optimum number of steps are the maximum values.

Results given in Tables 3.2 and 3.3 are plotted in Figures 3.5 and 3.6, respectively. They show how the code rate of permutation decodable codes increases with the number of U permutation steps.

The relations between R_c, k and s can also be illustrated in Figure 3.7, where

$$R' = \begin{cases} \frac{1}{t+2+\frac{2}{t}} & \text{for } t = 5 \\ \frac{1}{t-2} & \text{for } t \geq 7. \end{cases}$$

For the region of $k < k^*$, the code rate of PD codes is around R'. Some codes are restricted by their smaller d, while some other codes which have larger d may enjoy higher code rates when s continues to increase.

Another important feature which can be seen from Figure 3.7 is how k^* increases as

Table 3.2: Optimum value of s for PD $(n, 195, 7)$ cyclic codes

s	n	$R_s = \frac{k}{n}$	$\frac{R_s}{R_{s-1}} - 1$
1	585	0.33333	
2	585	0.33333	0%
3	585	0.33333	0%
4	575	0.33913	1.7%
5	559	0.34884	2.9%
6	527	0.37002	6.1%
7	463	0.42117	13.8%
8	**335**	**0.58209**	**38.2%**

Table 3.3: Optimum value of s for PD $(n, 21, 7)$ cyclic codes

s	n	$R_s = \frac{k}{n}$	$\frac{R_s}{R_{s-1}} - 1$
1	63	0.33333	
2	63	0.33333	0%
3	63	0.33333	0%
4	53	0.39623	18.9%
5	**37**	**0.56757**	**43%**
6	37	0.56757	0%

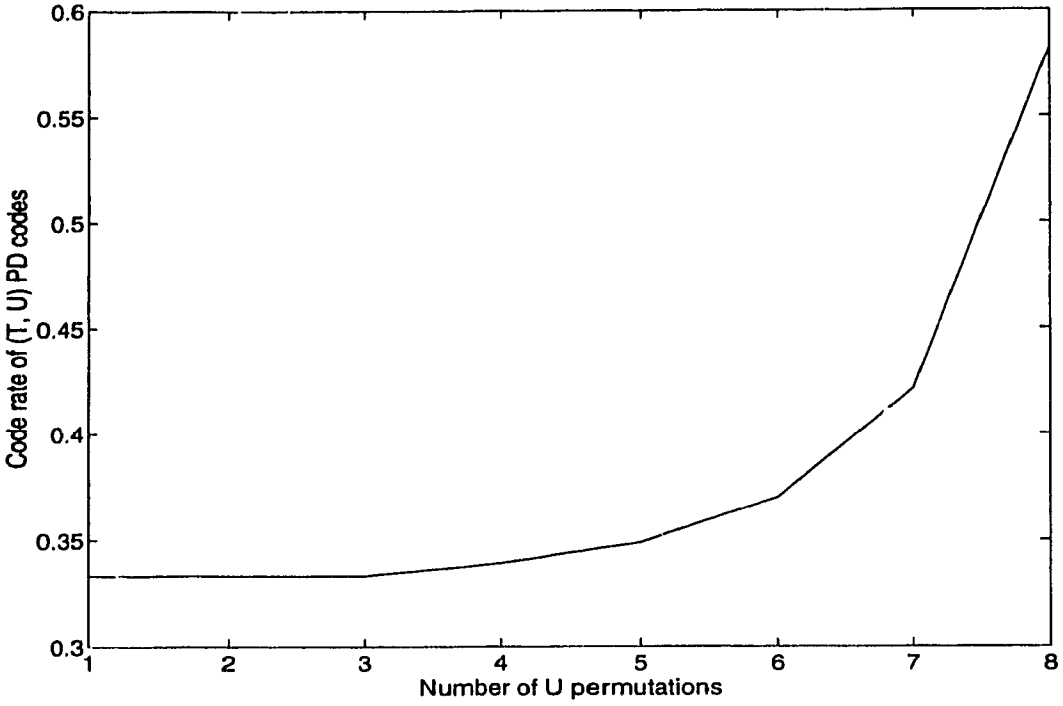


Figure 3.5: Illustration of the optimum value of s for the PD (n, 195, 7) cyclic code.

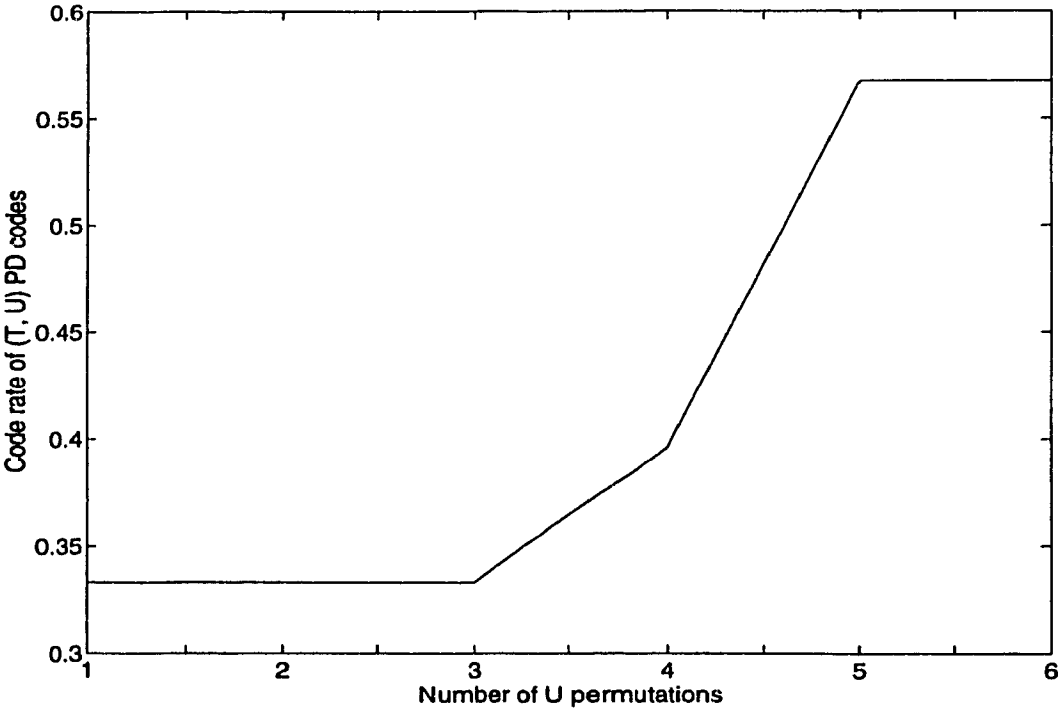


Figure 3.6: Illustration of the optimum value of s for the PD (n, 21, 7) cyclic code.

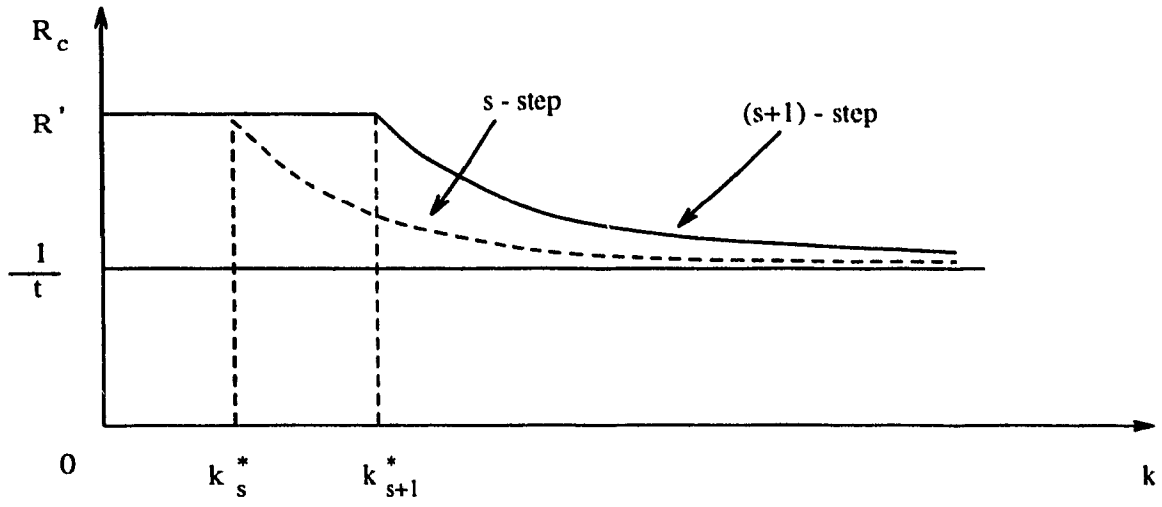


Figure 3.7: The relation between R_c, k, t and s.

each additional U permutation is applied, i.e., for each additional U permutation, k^* is more than doubled. This figure also shows how ΔR_{cs} increases with s, to the extent that k^* is large enough and $k \leq k^*$ is satisfied (which is the optimum permutation step). Finally, Figure 3.7 shows that when k increases, R_c approaches $\frac{1}{t}$, i.e., (T, U) permutation is more efficient for short cyclic codes.

3.6 Conclusions

This chapter presented a study on the performance of multiple-step (T, U) permutation decoding of cyclic codes. The characteristics of error-free gap lengths in the permuted error polynomials were examined. It was shown that the relation between the gap-length vectors of the permuted error polynomials are related by a linear mapping. Based on this property, the capability of (T, U) permutation decoding was studied and the relationship between the number of permutations s and code parameters n, k and t of a permutation decodable cyclic code was established. The following results can be summarized:

1. When $k \rightarrow \infty$, the rate of s-step PD codes decreases and approaches $\frac{1}{t}$.

2. When $k \to k^*$, R_c approaches to R', where

$$R' = \begin{cases} \frac{1}{t+2+\frac{2}{t}} & \text{for } t = 5 \\ \frac{1}{t-2} & \text{for } t \geq 7. \end{cases}$$

3. For each additional step, k^* is more than doubled. In the region of $k < k^*$, R_c is around the value R'.

4. $\Delta R_{cs} = R_{c(s+1)} - R_{cs}$ increases with the number of permutations s until $k < k^*$ is satisfied.

5. There exists an optimal value of s which makes the largest improvement in the code rate of PD codes.

Chapter 4

PERMUTATION DECODING USING PRIMITIVE ELEMENTS AS MULTIPLIERS

4.1 Introduction

In Chapter 3 we introduced the general relationship between the number of times that U permutation is used and the code parameters n, k and t of the corresponding permutation decodable cyclic codes. From these results, the performance of (T, U) permutation decoding method is bounded by the code rate R'. Actually, when $k < k^*$, the capability of the (T, U) permutation decoding depends on the code length n, and it is possible for some specific code lengths to decode much higher rate codes. One such case can occur when p is the primitive element of a prime field of order n. To have a more general idea of the limit of the permutation decoding technique, we will consider permutation decoding which uses primitive elements of a prime field as multipliers to increase the capability of permutation-decoding method in the region of $k \leq k^*$.

First, let us give an example. For BCH codes with code length $n = 127$, the maximum number of U permutation that can be applied is 6, and all the BCH codes of this code

length have a code rate higher than R' and cannot be decoded by (T, U) permutation decoding. But if we use 13 as a multiplier instead of 2, the BCH codes are decodable for $t = 2, 3, 4$, and 5. The code rate of these codes is compared with R' in the following table:

Table 4.1: A comparison between R' and the code rate of BCH codes with code length 127 which can be decoded by multiplier 13

t	R'	R_{BCH}
2	0.8	0.89
3	0.5	0.83
4		0.78
5	0.294	0.72

The corresponding figure is shown in Figure 4.1:

It should be emphasised that if we do not consider whether the codes under consideration exist or not, then the code rates of cyclic codes which are permutation decodable by multiplier 13 will be higher than the code rates listed in Table 4.1. From Figure 4.1 we can also see that as the error correcting capability t increases the code rates of permutation decodable (with multiplier 13) BCH codes decrease much slower than R' does.

4.2 Using Primitive Elements as Multipliers

It is also possible to decode cyclic codes by using permutations which may not preserve the code. However the decoder must have the ability to correct error patterns in the equivalent codes. In order to define the generator polynomial $g(x)$ by the equation

$$g(x) = \prod_{k \in K} (x - \alpha^k)$$

we must specify both α, a particular primitive nth root of unity, and K, the set of powers of α which are roots of $g(x)$. Although different codes arise from different choices of α, these different codes are equivalent in the sense that they are permutations of each other.

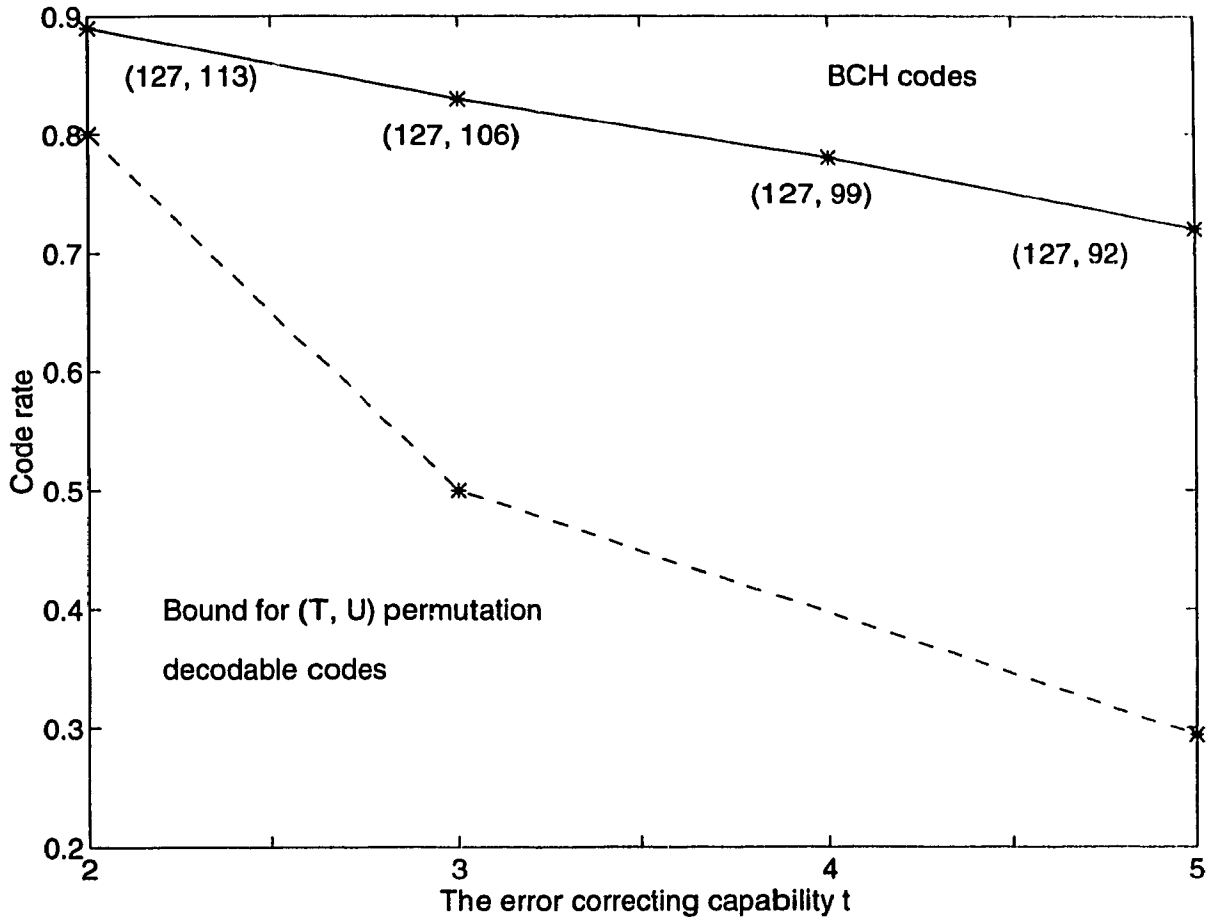


Figure 4.1: A comparison between R' and the code rate of the BCH codes with code length 127 which can be decoded with multiplier 13.

Clearly, the decoder can correct an error pattern in any code when all errors lie in or can be shifted into the "parity" portion of the codeword.

Let us consider a binary cyclic code with length n equal to a prime number p, then the set $S = \{0, 1, 2, \ldots, p-1\}$ is a field of order p under $modulo-p$ addition and multiplication. In fact, the set S can be partitioned into subsets[1] which are invariant under U permutation. For example, for $p = 31$, these subsets are (0), (1, 2, 4, 8, 16), (3, 6, 12, 24, 17), (5, 10, 20, 9, 18), (7, 14, 28, 25, 19), (11, 22, 13, 26, 21), (15, 30, 29, 27, 23). The union of any number of invariant subsets is also invariant under U; and this limits the improvement of the permutation decoding technique. These subsets exist because 2 is a non-primitive element of the field $\{0, 1, 2, \ldots, p-1\}$. Because $(2^5 \ mod \ p = 31) = 1$, so the maximum number of possible permutation steps is 5, and the subsets contain at most 5 elements each.

Now we consider the use of a primitive element of the field $GF(p)$ as a multiplier. We define M_i permutation as:

$$M_i^s c(x) \equiv \sum_{j=0}^{n-1} b_j x^{i^s \cdot j} \quad mod \ x^p - 1.$$

which is equivalent to

$$x^j \longmapsto x^{i^s \cdot j} \quad mod \ x^p - 1, \quad i, s \in S.$$

If M_i-permutation is used $(s-1)$ times to decode a certain code, then we say the code is s-step permutation-decodable.

Since $GF(p)$ is a prime field, there exist some primitive elements i with order $p-1$ which satisfies $i^{p-1} = 1$. So we can perform M_i permutation $(p-2)$ times. Because the field $GF(p)$ cannot be partitioned into subsets which are invariant under M_i permutation, the capability of the permutation method is expected to be increased. As an example, for the BCH (31, 11, 11) code, if we use 12 (which is a primitive element) as a multiplier, it is 6-step permutation-decodable (PD). However, it is not a PD code if we use U permutation, i.e., by using the primitive element as the multiplier, the code rate in the region of $k \leq k^*$ is increased.

[1]The subsets in the partition of S are called *cyclotomic cosets* [12]

Because p is a prime number, every multiplier i $(1 \leq i \leq p-1)$ is relatively prime to p (the block length of the word), and the mapping M_i is an automorphism. When i is not the characteristic of $GF(q)$, the permutation will not preserve the code, but it may map one cyclic code into another equivalent cyclic code.

The condition on which a cyclic code will be mapped to another equivalent cyclic code is given in the following theorem.

Theorem 4.1 *If the generator $g_1(x)$ of a linear cyclic code C_1 is mapped to a codeword of another equivalent cyclic code C_2, then C_1 is mapped to C_2.*

Proof of Theorem 4.1 [2]: Suppose $g_i(x)$ are the generator polynomials of $C_i (i = 1, 2)$. If $g_1(x)$ is mapped to a codeword of C_2, then $g_2(x) | M_j(g_1(x))$. For $x^h g_1(x) (1 \leq h \leq k)$, it is mapped to the codeword

$$\begin{aligned}
(x^h \cdot g_1(x))^j \mod (x^n - 1) &= x^{hj}(g_1(x))^j \mod (x^n - 1) \\
&= x^{hj}((g_1(x))^j \mod (x^n - 1)) \mod (x^n - 1) \\
&= x^{hj} \cdot M_j(g_1(x)) \mod (x^n - 1)
\end{aligned}$$

which is the hjth cyclic shift of $M_j(g_1(x))$ and it is also a codeword of C_2. Clearly this mapping preserves addition and multiplication. If $x^{uj} = x^{vj} \ (mod\ x^p)$, then $(u - v)j = 0 \ mod\ p$. Since p is a prime, j is relatively prime to p, this implies that $i - j = 0 \ mod\ p$ or $x^u = x^v$. Hence this mapping is an isomorphism, and the above condition holds.

Q. E. D.

4.3 Example

Consider the *BCH* (31, 16, 7) code. Suppose the code C_1 is generated by

$$g_1(x) = x^{15} + x^{11} + x^{10} + x^9 + x^8 + x^7 + x^5 + x^3 + x^2 + x + 1$$

[2] it has been proven that if C is a binary cyclic code of length n, then the permutation M_i maps C onto another binary cyclic code C' or onto C itself [44].

Then, with M_{12} permutation, g_1 is mapped into a code word

$$c(x) = x^{29} + x^{27} + x^{25} + x^{24} + x^{22} + x^{15} + x^{12} + x^8 + x^5 + x^3 + 1$$

which is divisible by

$$g_2(x) = x^{15} + x^{13} + x^{11} + x^8 + x^6 + x + 1.$$

Therefore, M_{12} maps the code C_1 into another BCH code C_2 which is generated by $g_2(x)$. Similarly, with M_{12} permutation, C_2 is mapped into other BCH codes which are generated by

$$g_3(x) = x^{15} + x^{14} + x^{12} + x^{11} + x^{10} + x^8 + x^6 + x^4 + x^3 + x^2 + 1$$

and

$$g_4(x) = x^{15} + x^{14} + x^{13} + x^{12} + x^{10} + x^8 + x^7 + x^6 + x^5 + x^4 + 1$$

successively. Therefore we can use the primitive element 12 as a multiplier to decode this code. This code is found to be 4 - step permutation decodable. The comparison of U and M_{12} permutation for decoding the BCH codes of length 31 is summarized in the following table.

4.4 Implementation of U and M_i Permutation Decoders

There are two ways to realize the permutation decoding of a certain cyclic code: parallel processing and serial processing. For s-step permutation decodable codes, s syndrome registers are needed in parallel processing. As shown in Figure 4.2, the received vector is shifted to the s syndrome registers at the same time, so the time needed to decode is almost the same as the basic error-trapping decoder (ETD). The block diagram of a permutation decoder using serial processing is given in Figure 4.4. In this case, only one syndrome register is need, but the decoding time is approximately equal to the number of steps needed to decode the code multiplied by the decoding time of a parallel processing permutation decoder.

Table 4.2: A comparison between U and M_{12} permutation of decoding BCH codes of length 31.

BCH Code	Multiplier	Steps (PD or NPD)
(31, 16, $d = 7$)	2	5 - step PD
	12	4 - step PD
(31, 11, $d = 9$)	2	4 - step PD
	12	3 - step PD
(31, 11, $d = 11$)	2	NPD
	12	6 - step PD
(31, 6, $d = 15$)	2	4 - step PD
	12	3 - step PD

If parallel processing is applied for fast decoding, the M_i permutation decoder is simpler than the U permutation decoder because the former requires a smaller number of syndrome registers than the latter.

If serial processing is used for decoding, then the M_i permutation decoder is slightly more complex than the U permutation decoder because the M_i permutation may not preserve the code, and, therefore, a logical circuit is needed to control the feedback of the syndrome register. However the decoding time of a serial M_i-permutation decoder is shorter than that of a serial U-permutation decoder. The syndrome register of the M_{12} decoder for the BCH (31, 16, $d = 7$) code is shown in Figure 4.4.

4.5 Remarks

1. In some cases, the capability of the permutation decoding method is expanded. For example, the BCH (31, 11, $d = 11$) code is not permutation decodable by (T, U)

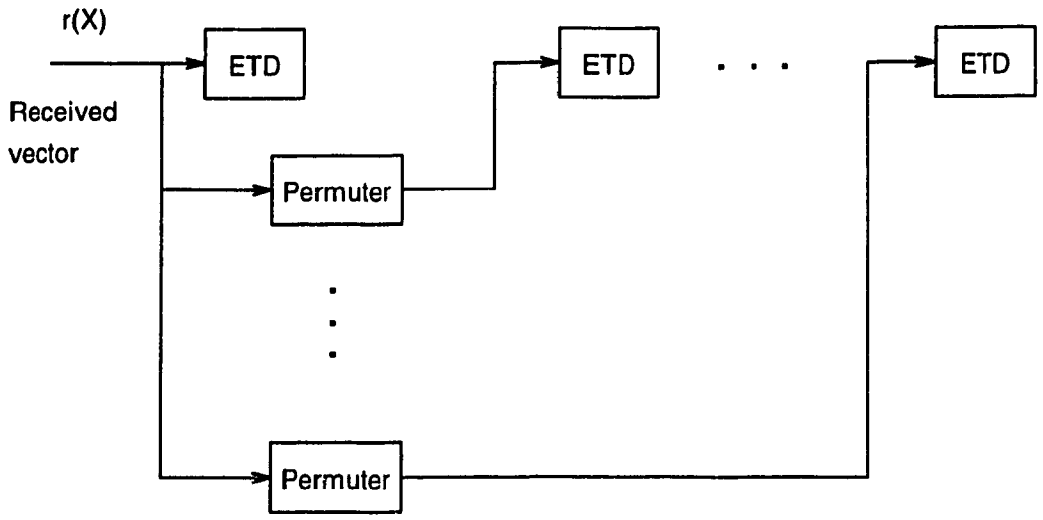


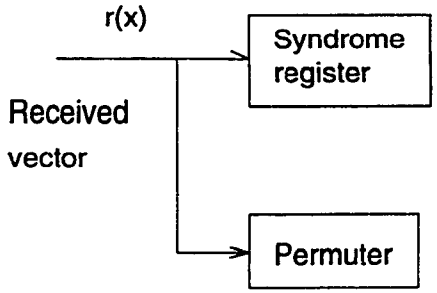
Figure 4.2: Permutation decoder using parallel processing.

permutation, but it is a 6-step permutation decodable code by (T, M_{12}) permutation.

2. M_i permutation decoder is simpler with parallel processing than U permutation decoder, it is also faster in both parallel and serial processing decoding. However, in serial processing, since M_i permutation may not preserve the code, an additional logical circuit is needed to control the feedback in the syndrome register of the M_i permutation decoder, this makes the M_i permutation decoder slightly more complex than the U permutation decoder.

3. When t increases, more permutation steps are required. The maximum number of required syndrome registers equals the number of existing equivalent codes. After the code is permuted so many times by the primitive multiplier, it will be mapped back to the original one.

4. The code rate of the permutation decodable codes decreases much more slower than the code rate of the (T, U) permutation decodable codes does, that is to say, this method is very efficient in increasing the code rate of the permutation decodable



An example of the Permuter of the U permutation decoder for the binary (31, k, d) code

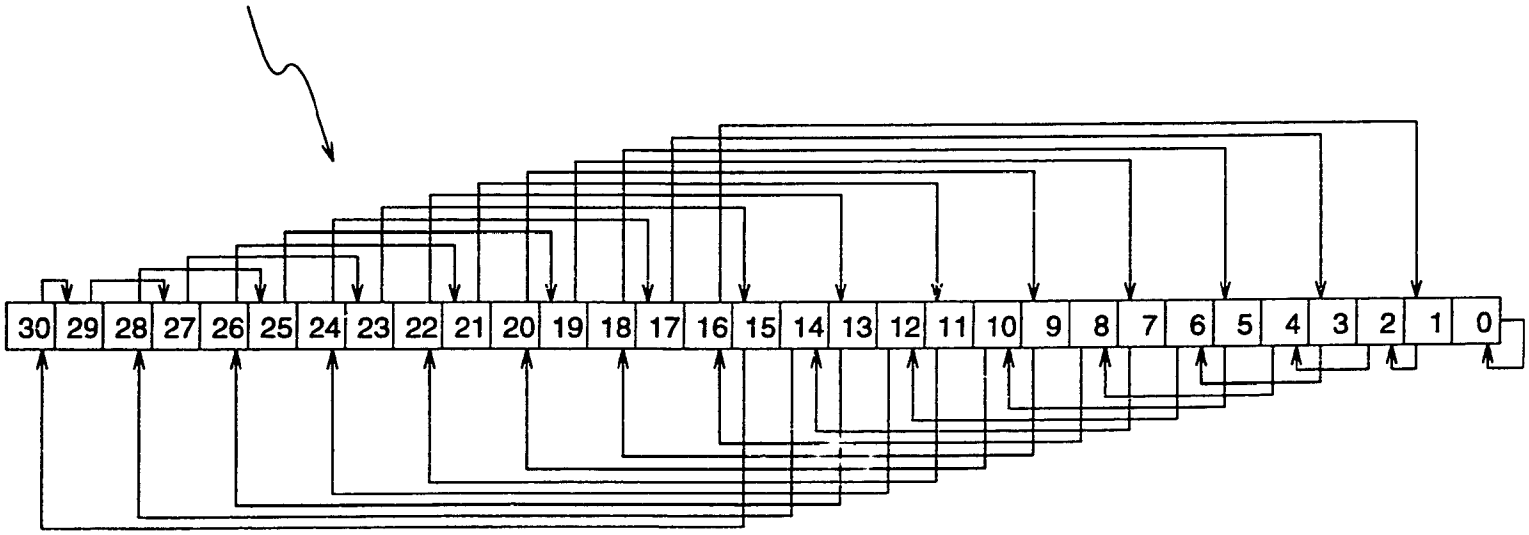
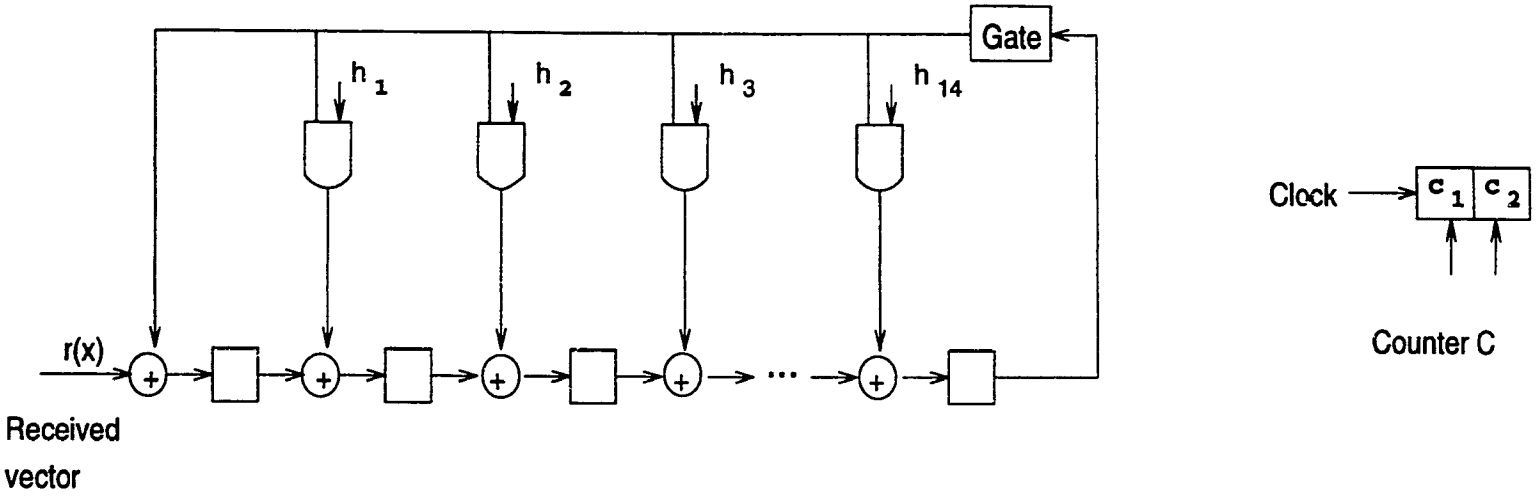


Figure 4.3: Permutation decoder using serial processing



$h_1 = \bar{c}_2$ $h_2 = \bar{c}_1$ $h_3 = \bar{c}_1$ $h_4 = c_2$ $h_5 = \bar{c}_1 \bar{c}_2 + c_1 c_2$ $h_6 = c_1 + c_2$

$h_7 = \bar{c}_1 \bar{c}_2 + c_1 c_2$ $h_8 = 1$ $h_9 = \bar{c}_1 \bar{c}$ $h_{10} = \bar{c}_1 + c_2$ $h_{11} = \bar{c}_1 + \bar{c}_2$

$h_{12} = c_2$ $h_{13} = c_1$ $h_{14} = c_2$

Figure 4.4: Syndrome register of the M_{12} decoder for the BCH (31, 16, d=7) code

codes with higher error correcting capability.

Please notice that permutation decoding does not solve the two basic problems in information set decoding, i.e.,

1. What is the minimum number of distinct sets?

2. How does one go about finding these sets?

What permutation decoding does is to increase the number of selected information sets according to a predetermined rule so that more error patterns can be corrected. It should be emphasized that permutation decoding by using a fixed multiplier may not be an optimum way of choosing an information set because:

1. The distinct sets chosen by fixed multiplier may not be the minimum number of distinct sets needed to decode a code;

2. The existence of invariant subsets for every fixed multiplier limits the capability of permutation decoding.

The way to solve this problem is to use different multipliers to decode a code. This can be done in a serial or parallel manner. The key point here is that we don't use a fixed multiplier, but several multipliers to decode a code, and each multiplier is not used to its limit (which is determined by the size of invariant subsets). In this way, the decoding circuitry can be much simpler than the one using primitive element as a fixed multiplier; the time used in decoding a code can be much shorter (suppose serial processing is used); and the decoding capability can be much higher (can decode all the existing code as long as we can find an appropriate multiplier set).

One point deserves to be noticed is that by using a primitive element as a fixed multiplier, the total information sets can be searched by a decoder is $n \times n$.

We can see that the bound improvements in both bounds: $2(2l+2) = \min\{2(t_e - 4);\ 2(k_o - 1)\}$ in Bound 1 and $2(2l+3) = \min\{2(t_e - 3);\ 2(k_e - 3)\}$ in Bound 2 are proportional to k_o or k_e and t_e. From the above mentioned results, we can also conclude that when $t_e > k_e$ (or k_o), the codes $(n,\ k_o,\ 2t_e + 1)$ get more improvement than codes $(n,\ k_e,\ 2t_e + 1)$.

In Chapter 3, we studied the performance of multiple-step $(T,\ U)$ permutation decoding of cyclic codes. We proposed the mapping matrix, which gives a comprehensive relation between gap lengths in different domains. We proved that the gap lengths in the error pattern after U permutation is actually a linear combination of the gap lengths in the error pattern before U permutation. We characterized the relationship of the error pattern gaps between *any* two consecutive or non-consecutive domains. From the proof of Theorem 3.3 and 3.4, we gave an insight into how the gap lengths changed and therefore the capability of $(T,\ U)$ permutation decoding increased.

Chapter 3 also presented the general relationship between the permutation step s and the code parameters n, k and t of a permutation decodable cyclic code. From this general relationship, we got the following conclusions:

1. When $k \to \infty$, the rate of s-step PD codes decreases and approaches $\frac{1}{t}$.

2. When $k \to k^*$, R_c approaches to R', which is

$$\frac{1}{t_o - 2 + \frac{2}{t_o}}$$

and

$$\frac{1}{t_o - 2}$$

for $t_o = 5$ and $t_o \geq 7$ respectively.

3. For each additional step, k^* more than doubles. In the region of $k < k^*$, R_c is around the value R'.

4. $\Delta R_{cs} = R_{c(s+1)} - R_{cs}$ increases as the permutations steps s increases until k^* is large enough and $k < k^*$ is satisfied.

5. There exists an optimal step which makes the most improvement in the code rate of PD codes.

The characteristics of the code rate have also been studied. From the concept of the optimum permutation step, we showed how to estimate the U permutation steps needed to decode a permutation decodable code. We compared the code rate R' with the code rate of error-trapping decodable codes, which shows the effective region of the (T, U) permutation technique. In this effective region, some powerful BCH codes (e.g., (127, 113, 5), (127, 106, 7), etc.) can be decoded.

Chapter 4 discussed permutation decoding using primitive elements as multipliers, from both invariant subsets point of view and information set decoding point of view. In Chapter 4, we presented the following main results:

1. In some cases, the capability of the permutation decoding method is expanded. For example, the BCH $(31, 11, d = 11)$ code is not permutation decodable by (T, U) permutation, but it is a 6-step permutation decodable code by (T, M_{12}) permutation.

2. M_i permutation decoder is simpler with parallel processing than U permutation decoder, it is also faster in both parallel and serial processing decoding. However, in serial processing, since M_i permutation may not preserve the code, an additional logical circuit is needed to control the feedback in the syndrome register of the M_i permutation decoder, this makes the M_i permutation decoder slightly more complex than the U permutation decoder.

3. When t increases, more permutation steps are required. The maximum number of required syndrome registers equals the number of existing equivalent codes. After the code is permuted so many times by the primitive multiplier, it will be mapped back to the original one.

4. The code rate of the permutation decodable codes decreases more slowly than the code rate of the (T, U) permutation decodable codes does, that is to say, this method

is very efficient in increasing the code rate of the permutation decodable codes with higher error correcting capability.

Furthermore, Chapter 4 suggested a new way of applying information set decoding, i.e., use the (T, U) permutation and (T, M_i) permutation at the same time. By mapping a code to different equivalent codes and using (T, U) permutation decoding parallelly in these code domains, the capability of permutation decoding techniques can be increased dramatically. In this way, we explored the full advantage of both the (T, U) and the (T, M_i) permutation decoding. This may provide the most powerful permutation decoding technique while keep the decoder relatively simple.

5.2 Further Work

The permutation decoding method for decoding cyclic codes can be studied further in the following ways:

1. Using primitive elements as multipliers is very efficient in increasing the code rate of the permutation decodable codes with higher error correcting capability (Chapter 4). This type of decoder should be investigated further.

2. Permutation decoding using fixed multipliers may not be an optimum way of choosing different information set (Chapter 4). To find a rule in choosing multiplier sets is an important way to apply permutation decoding to higher rate codes.

3. By combining both (T, U) and M_i permutation decoding, (i.e., after using M_i permutation to permute a code to an equivalent code, parallel (T, U) permutation can be used in a different code domain) the capability of permutation decoding can be increased while the complexity and the time needed for decoding remains the same as with the (T, M_i) permutation decoding alone. It is believed that the effective region of this combined technique is significantly larger than the region using the (T, U) permutation alone. This is because the number of information sets used in decoding is significantly increased in the combined case.

Bibliography

[1] C. E. Shannon, "A Mathematical Theory of Communications," *Bell Syst. Tech. J.*, vol. 27, pp. 379-423 (Part I), 623-656 (Part II), July 1948.

[2] C. E. Shannon, "Communication in the Presence of Noise," *Proc. IRE,* vol. 37, pp. 10-21, January 1949.

[3] S. Lin and D. J. Costello, *Error-Control Coding; Fundamentals and Applications,* Prentice-Hall, Englewood, New Jersey, 1983.

[4] E. R. Berlekamp, *Algebraic Coding Theory,* McGraw-Hill, New York, 1968.

[5] Richard D. Gitlin, Jeremiah F. Hayes and Stephen B. Weinstein, *Data Communications Principles,* Plenum Press, New York, 1992.

[6] W. W. Peterson and E. J. Weldon, Jr., *Error-Control Codes,* MIT Press, 1972.

[7] R. W. Lucky, J. Salz and E. J. Weldon, *Principles of Data Communication,* McGraw-Hill, New York, 1968.

[8] J. L. Massey, *Threshold Decoding,* MIT Press, Cambridge, 1963.

[9] R. G. Gallager, *Information Theory and Reliable Communication,* Wiley, New York, 1968.

[10] P. Fire, "A Class of Multiple-Error-Correcting Binary Codes for Non-independent Errors," Sylvania Report No. RSL-E-2, Sylvania Electronic Defense Laboratory, Reconnaissance Systems Division, Mountain View, Calif., March 1959.

[11] B. Elspas and R. A. Short, "A note on Optimum Burst-Error Correction Codes," *IRE Trans. Inform. Theory,* IT-8, pp. 39-42, January 1962.

[12] F. J. MacWillians and N. J. A. Sloane, *The Theory of Error-Correcting Codes,* North-Holland, Amsterdam, 1977.

[13] G. C. Clark and J. B. Cain, *Error-Correcting Coding for Digital Communications,* Plenum Press, New York, 1981.

[14] R. E. Blahut, *Theory and Practice of Error Control Codes,* Addison Wesley, 1983.

[15] R. J. McEliece, *The Theory of Information and Coding: a mathematical framework for communication,* Addison-Wesley, 1977.

[16] W. G. Chambers, *Basics of Communications and Coding,* Oxford University Press, 1985.

[17] A. M. Michelson, *Error-control techniques for digital communication,* Wiley, New York, 1985.

[18] J. E Meggitt, "Error Correcting Codes and Their Implementation," *IRE Trans. Inf. Theory,* IT-7, pp.232-244, October 1961.

[19] V. K. Bhargava, "Some Results on Quasi-Cyclic Codes", Ph. D. Thesis, Queen's University, Kingston, April 1974.

[20] M. E. Mitchell, "Coding and Decoding Operation Research," *G.E. Advanced Electronic Final Report on Contract AF 19(604)-6183,* Air Force Cambridge Research Labs., Cambridge, Mass., 1961.

[21] M. E. Mitchell, "Error-Trap Decoding of Cyclic Codes," *G.E. Report No. 62MCD3,* General Electric Military Communications Dept., Oklahoma City, Okla., Dec. 1962.

[22] H. O. Burton and D. D. Sullivan, "Errors and Error Control," *Proc. IEEE,* 60(11), pp. 1293-1310, November 1972.

[23] K. Brayer, "Error Control Techniques Using Binary Symbol Burst Codes," *IEEE Trans. Commun.*, COM-16, pp. 199-214, April 1968.

[24] L. Rudolph, "Easily Implemented Error-Correction Encoding-Decoding," *G. E. Report No. 62MCD2,* General Electric Corporation, Oklahoma City, Okla, Dec. 1962.

[25] T. Kasami, "A Decoding Procedure For Multiple-Error-Correction Cyclic Codes," *IEEE Trans. Inform. Theory,* IT-10, pp. 134-139, April 1964.

[26] E. Prange, "The use of Information Sets in Decoding Cyclic Codes," *IEEE Trans. Inform. Theory,* IT-8, pp. 85-89, Sept. 1962.

[27] F. J. MacWilliams, "Permutation Decoding of Systematic Codes," *Bell Syst. Tech. J.*, 43, Part I, pp. 485-505, Jan. 1964.

[28] E. J. Weldon, Jr., "A comparison of an Interleaved Golay Code and a Three- Dimensional Product Code," *Final Report, USNELC Contract N0095368M5345,* Aug. 1968.

[29] P. W. Yip, "Permutation decodable cyclic codes," M.S. thesis, Univ. of Ottawa, Ottawa, ON, Canada, May 1974.

[30] S. G. S. Shiva, K. C. Fung, and H. S. Y. Tan, "On permutation decoding of binary cyclic double-error-correcting codes of certain lengths," *IEEE Trans. Inform. Theory,* vol. IT-16, pp. 641-643, Sept. 1970.

[31] S. G. S. Shiva and K. C. Fung, "Permutation decoding of certain triple-error-correcting binary codes," *IEEE Trans. Inform. Theory,* vol. IT-18, pp. 444-446, May 1972.

[32] S. G. S. Shiva, A. Benyamin-Seeyar, and V. K. Bhargava, "On the permutation-decodability of triple-error-correcting codes," in *Proc. 21st Annu. Allerton Conf. on Communications, Control, and Computing,* Illinois Univ., Oct. 1983.

[33] A. Benyamin-Seeyar, S. S. Shiva and V. K. Bhargava "Capability of the Error-Trapping Technique in Decoding Cyclic Codes," *IEEE Trans. on Inform. Theory,* vol. IT-32, No. 2, pp. 166-180, March 1986.

[34] John G. Proakis, *Digital Communications,* McGraw-Hill Book Company, 1989.

[35] M. Jia, A. Benyamin-Seeyar, and T. Le-Ngoc, "Exact Lower Bounds on the Codelength of Three-Step Permutation-Decodable Cyclic Codes," *IEEE Trans. Inform. Theory,* IT-38, No. 6, pp. 1812-1817, November 1992.

[36] M. Jia, A. Benyamin-Seeyar and T. Le-Ngoc, "On the Capability of (T, U) Permutation Decoding Method," *IEEE Trans. Commun.*, vol. 42, No. 2/3/4, pp. 192-195, February/March/April 1994.

[37] M. Jia, A. Benyamin-Seeyar and T. Le-Ngoc, "(T, U) Mapping Matrices, Their Properties and Their Applications in Permutation Decoding," *1994 IEEE International Symposium on Information Theory,* 27 June - 1 July 1994, Trondheim, Norway.

[38] T. Le-Ngoc, M. Jia and A. Benyamin-Seeyar, "Permutation Decoding Using Primitive Elements as Multipliers," *Information theory and applications*: third Canadian workshop, Rockland, Ontario, Canada, May 30-June 2, 1993.

[39] M. Jia, T. Le-Ngoc and A. Benyamin-Seeyar, "Performance of Multiple Step (T, U) Permutation Decoding of Cyclic Codes," *Globecom'95,* November 13-17, 1995, Singapore.

[40] W. W. Peterson, "Encoding and Error-Correction Procedures for the Bose-Chaudhuri Codes," *IRE Trans. Inf. Theory,* IT-6, pp. 459-470, September 1960.

[41] R. T. Chien, "Cyclic Decoding Procedures for the Bose-Chaudhuri-Hocquenghem Codes," *IEEE Trans. Inf. Theory,* IT-10, pp. 357-363, October 1964.

[42] I. S. Reed, "A Class of Multiple-Error-Correcting Codes and the Decoding Scheme," *IRE Trans.*, IT-4, pp. 38-49, September 1954.

[43] D. E. Muller, "Applications of Boolean Algebra to Switching Circuit Design and to Error Detection," *IRE Trans.*, EC-3, pp. 6-12, September 1954.

[44] Vera Pless, *Introduction to the Theory of Error-Correcting Codes*, John Willey and Sons, New York, 1982.