An Abstract of the Thesis of

Sulaiman A. Al-Bassam for the degree of Doctor of Philosophy in Computer Science presented on January 4, 1990.

Title: Balanced Codes

Abstract approved: _____

Bella Bose

Balanced codes, in which each codeword contains equally many 1's and 0's, are useful in such applications as in optical transmission and optical recording. When balanced codes are used, the same number of 1's and 0's pass through the channel after the transmission of every word, so the channel is in a dc-null state. Optical channels require this property because they employ AC-coupled devices. Line codes, in which codewords may not be balanced, are also used as dc-free codes in such channels.

In this thesis we present the research that leads to the following results:

1- Balanced codes -- These have higher information rate than existing codes yet maintain similar encoding and decoding complexities.

2- Error-correcting balanced codes -- In many cases, these give higher information rates *and* more efficient encoding and decoding algorithms than the best-known equivalent codes.

3- DC-Free *coset* codes -- A new technique to design dc-free *coset* codes was developed. These codes have better properties than existing ones.

4- Generalization of balanced codes -- Balanced codes are generalized in three ways among which the first is the most significant:

a) Balanced codes with low dc level -- These codes are designed based on the combined techniques used in (1) and (3) above. A lower dc-level and higher transitions density is achieved at the cost of one extra check bit. These codes are much more attractive, to optical transmission, than the bare-bone balanced codes.

b) Non-Binary Balanced Codes -- Balanced codes over a non-binary alphabet.

c) Semi-Balanced Codes -- Codes in which the number of 1's and 0's in every code word differs by at most a certain value.

5- t-EC/AUED coset codes -- These are t error correcting/all unidirectional error detecting codes. Again the technique in (3) above is used to design t-EC/AUED coset codes. These codes obtain higher information rate than the best-known equivalent codes and yet maintain the same encoding/decoding complexity.

Balanced Codes

by

Sulaiman Al-Bassam

A Thesis

submitted to

Oregon State University

in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

Completed January 4, 1990

Commencement June 1990

Approved:

_____

Professor of Computer Science in charge of major

_____

Head of Department of Computer Science

_____ _____

Dean of Graduate School

Date thesis is presented January 4, 1990

Typed by Sulaiman Al-Bassam for Sulaiman Al-Bassam

Acknowledgements

Most, if not all, of the ideas presented in this thesis were inspired in one way or another by my advisor Professor Bella Bose. I am truly grateful to him for that and for his guidance and support for many years. I would like to thank Dr. Andrew Klein, Dr. Ted Lewis, Dr. Toshi Minoura, and Dr. Walter Rudd for serving on my committee and for their instructive comments. I would like to thank Dr. Toshi Minoura for his special friendship.

I am also grateful to UPM (University of Petroleum and Minerals) for supporting my studies and OSU (Oregon State University) for providing me with excellent education.

Special thanks and appreciation goes to my friend (and officemate) Bob Rowley for many exciting and fruitful discussions during the years. Last, but not least, to my family for their unlimited support and encouragement. In particular, to my brothers Mohammed and Adnan for their continuous help.

Table of Contents

List of Figures

List of Tables

Balanced Codes

Chapter 1

Introduction

1.1 Overall Communication System

Error control coding is a powerful tool in obtaining efficient and reliable transmission of data over noisy channels. A simple model of a communication system in which error control coding is applied is shown in Figure 1.1. A source generates a message U that is to be transmitted to a user over a noisy binary channel.

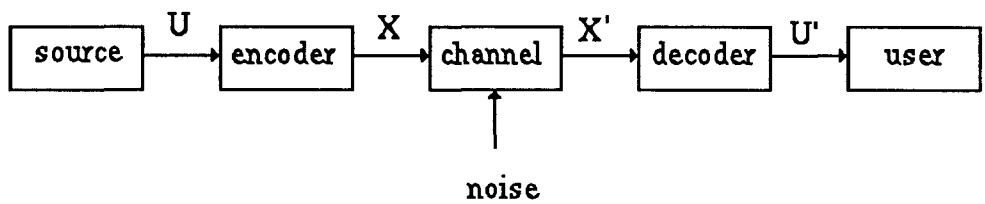


Figure 1.1 The overall communication system.

Each information word is encoded into some codeword X. This word X is sent through the channel. Because of channel noise, the received word X' may be different from X. The number of different coordinates in which X and X' differ is called the number of errors made during transmission. The decoder must decide from X' the correct message U (or equivalently X). The "channel" in Figure 1.1 can also be some storage device in which data is recorded for later retrieval. Data may suffer different kinds of errors in memory systems.

The efficiency and reliability of the communication and storage systems depend heavily on the behavior of the channel and the encoding/decoding procedures. A widely used channel is the binary symmetric channel in which 1→0 and a 0→1 errors occur with equal probability p (as shown in Figure 1.2). In the asymmetric channel the probability of a 0→1 error is ε and the probability of a 1→0 error is p with p >> ε (as shown in Figure 1.2). In the ideal asymmetric channel (also called a z-channel) ε = 0.

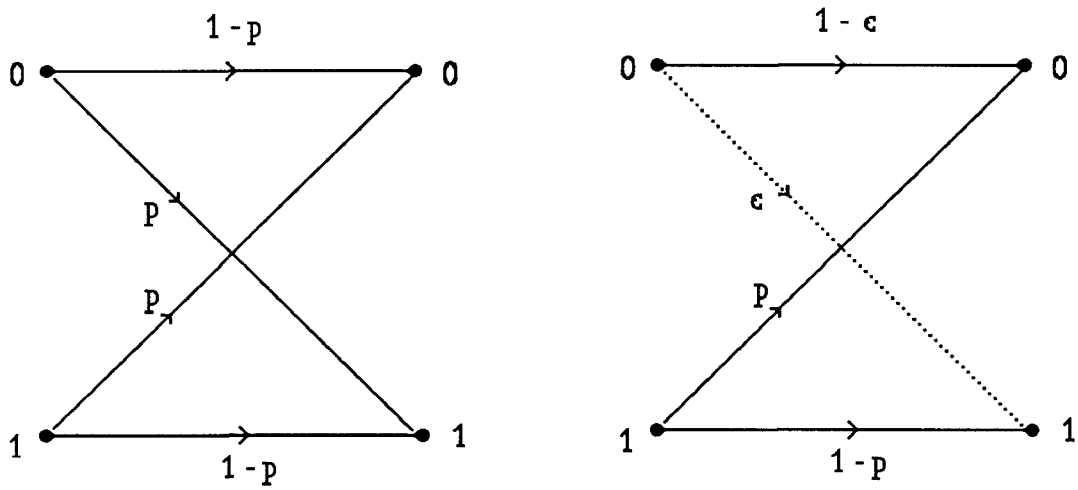


Figure 1.2 The Binary Symmetric and Asymmetric Channels.

Most conventional communications and storage channels suffer from symmetric errors. The bulk of coding theory deals with designing efficient codes for such channels. Many researchers developed efficient codes that correct and/or detect *symmetric* errors [Pet 72].

Some channels display only asymmetric errors such as in optical disk in which a 0 can be changed to a 1 but a written 1 can never be changed to a 0. The optical communication channels are considered to be asymmetric channels in which a 1→0 error occurs with much higher probability than a 0→1 error [Mor 83, Tak 76, Uye 88]. Also, the observed errors in other recently developed semiconductor and optical memory systems are of asymmetric type [Pra 80b]. Efficient *asymmetric* error correcting and *asymmetric* error detecting codes

that are suitable for these applications were rapidly developed [Ber 61, Bor 82, Bos 90, Con 79, Fre 62, Lin 88, Var 73, Web 88, Pra 80b].

1.2 Balanced Codes

Regardless of the behavior of the transmission or storage channel (i.e. symmetric or asymmetric), the channel may employ some other constraints. For example, in magnetic recording, it is required that there are not many 1 to 0 or 0 to 1 transitions in the encoded stream since magnetic devices are relatively slow in switching. Optical communication channels which behaves as an asymmetric channels require codes that are balanced, in which each codeword contains equally many 1's and 0's. Balanced codes are also useful in other applications such as:

1) Unidirectional and Asymmetric error detection [Ber 61, Fre 62, Pra 80a].

2) Used in the design of t-EC/AUED (t error correcting/ all unidirectional error detecting) codes [Bos 82b, Kun 88, Kun 90, Sai 88].

3) Fault masking in bus lines of VLSI systems [Mat 88].

4) State assignment in fault-tolerant sequential circuits [[Toh 71].

5) Maintaining data integrity in write-once memories [Lei 84, Knu 86].

6) Fiber optical communication [Den 88, Fer 84, Mor 83, Kaw 88, Tak 76, Wid 83, Yos 84].

7) Used in cryptographical systems [Ben 86].

In the following we discuss in more details the application regarding optical communication. The introduction of optical communication systems and optical disks

marked a significant advance in the field of information transfer and recording. Optical communication systems has provided the means of transferring vast quantities of data at very high speeds through their light-weight fibers which are electrically isolated and completely immune to inductive interference. As the employment of fiber optics expands, it is clear that there are some limiting factors, not in the actual transmission process, but in the required performance of the associated electronic circuitry.

A fiber optical communication system consists of three parts; (1) an optical transmitter, (2) a fiber optical communication channel, and (3) an optical receiver. Basically, the transmitter emits a light which is sent through the channel where it is sensed by the optical sensor on the receiver end.

The advantages of fiber optics of providing high-speed transmission can be costly. This is because the electronic circuitry has a poor response when switching at extremely high speeds, and accordingly requires the whole transmission process to be AC-coupled. This dictates the need for special codes to be used in fiber optical systems that provide no dc components in the coded stream. DC components occur by the accumulation of a similar signal (0 or 1). In a sense, if 0 and 1 are assigned negative and positive charges, a dc-free transmission occurs when the accumulated charge (disparity) at the end of the transmission is 0.

Because the optical channels generally act just as a transmission media, there are no coding schemes associated with them. It is up to the communications engineer to choose a code that is suitable for a specific application. The choice of a code can have an effect on the operation of the whole system. Clock-timing regeneration, data phase synchronization, and error-monitoring capabilities depend on the codes employed in the link. These codes should also have a short run length (the maximum number of consecutive 0's or 1's in the encoded stream) which creates many $1 \rightarrow 0$ and $0 \rightarrow 1$ transitions. Transitions are essential

for the receiver clock synchronization and detection processes. The optical communication codes must have encoders and decoders that are simple, as is essential for high-speed communication.

1.3 Previous Work and Research Results

Before reviewing the work on balanced codes, we list some standard notations that are widely used in the literature. We also introduce some new definitions.

Standard Notations:

k : number of information bits.

r : number of check bits.

n : the length of the code (n = k+r).

W(X) : the number of 1's in a binary vector X, also known as the Hamming weight .

N(X,Y) : the number of $1 \rightarrow 0$ crossovers from X to Y.

D(X,Y) : the Hamming distance between X and Y, i.e. .D(X,Y) = N(X,Y)+N(Y,X).

$X^{(j)}$: denotes X with its first j bits complemented, i.e. ($\overline{x}_1$, ..., $\overline{x}_j$, x_{j+1}, x_{j+2}, ...).

t-EC/B : a balanced code with distance 2t+2, i.e. a t-Error Correcting Balanced Code.

DC(n,k,2t+2) : a t-EC/B of length n and 2^k codewords.

DC(n,k) : short for DC(n,k,2) code, a balanced code of length n and 2^k codewords.

(The two notations t-EC/B and DC(n,k,2t+2) are used interchangably).

Definition 1.1 A code C of length n is balanced if for all $X \in C$, $W(X) = \lceil n/2 \rceil$ (or $\lfloor n/2 \rfloor$).

Example 1.1 The set of binary vectors { 1100, 1001, 1010, 1100, 0101, 0011} is a code of length 4. This code is balanced since each code word has weight 2. □

The information rate of a code is the ratio of the number of code words by all possible binary vectors. In the previous example the information rate is $\frac{6}{16}$. Before reviewing the work in balanced codes, we show how balanced codes are related to unordered codes.

Balanced Codes - A Class of Unordered Codes

A binary code C is called an unordered code when no codeword is "contained" in another; that is, the positions of the 1's in one codeword will never be a subset of the positions of the 1's in a different codeword. In other words, C is called unordered if and only if

$$N(X,Y) \geq 1 \quad \text{for all } X,Y \in C \quad \text{and} \quad X \neq Y$$

Balanced codes also have the property that no codeword is "contained" in another and so they are a class of unordered codes. To form the maximal unordered code of length n, Sperner's lemma [Spe 28] says that we can do no better than to construct the set of all balanced words of length n.

Balanced Codes - Systematic and Non-Systematic

A code is called systematic if the codewords start with the information bits unmodified. On the other hand in a non-systematic code, the information bits are modified in at least one codeword. A non-systematic balanced code of length n can be simply formed from the set of all binary vectors of length n and weight n/2 which has $\binom{n}{n/2}$ words. This code, say C, has maximal information rate but has no simple encoding and decoding methods (mostly table look ups). Using Stirling's approximation $x! \approx \sqrt{2\pi x}\ (x/e)^x$, we get $|C| = \binom{n}{n/2} \approx \sqrt{\frac{2}{\pi n}}\, 2^n$. The number of redundant bits (r) in C is $n - \log |C| \approx n - (n - 1/2 \log (2/\pi) - 1/2 \log (n))$, and $1/2 \log (2/\pi) \approx 0.326$, so

$$r \geq 1/2 \log n + 0.326 \tag{1-1}$$

On the other extreme, a systematic balanced code of k information bits needs at least k check bits. This is true since the information symbols in a systematic code are not altered and therefore the check symbol of the all zero information word must have k ones, thus

$$r \geq k \tag{1-2}$$

Such a code can be constructed by choosing the check symbol to be the complement of the information symbol. The information rate of this code (i.e. number of information bits by code length) is only 0.5 but the encoder and decoder will be very simple and fast.

Balanced Codes - Efficient Designs

The objective of coding theory is to find efficient codes that are suitable for different applications. The efficiency of a code is measured by its information rate and its encoding/decoding speed and complexity. There is always a trade-off in these two factors; it is most puzzling to find codes that are "good" in both properties. When codes are compared, one says code C_1 is better than C_2 if C_1 is better than C_2 in both properties or better in one and similar in the other.

Our primary interest is to design efficient balanced codes; i.e., balanced codes with high information rates with simple and fast encoder/decoder circuitry. In the search for efficient balanced codes, Knuth in 1986 gave two methods to construct such codes [Knu 86]; one method employs serial decoding while the other method uses parallel decoding. The main results given there are

1. Serial decoding: $k = 2^r$ information bits and r check bits (i.e. $r \leq \log k$).
2. Parallel decoding: $k = 2^r - r - 1$ information bits and r check bits (i.e. $r \leq \log n$).

In both methods, *serial* and *parallel* decoding, some appropriate number of information bits of the information word are complemented, starting from the first bit, and then a check is assigned to this modified information word to make the entire word balanced. In sequential decoding the check represents the weight of the original information word. The decoding of the received word is done by complementing one, two, three, etc., bits till the weight of the information word is equal to the value represented by the check. On the other hand, in the parallel decoding method the check directly indicates the number of information bits complemented and hence at the decoder side the original information word can be obtained by complementing these bits in one step. Clearly, the parallel decoding scheme is faster than the serial one. The serial scheme was extended from $k = 2^r$ to $k = 2^{r+1} - r - 2$ information bits (using r check bits) in [Bos 87].

In Chapter 2 we present construction of the serial and parallel balanced codes with the following parameters:

1. Serial decoding: $k \approx 2^{r+1} - 0.8\sqrt{r} - 2$.
2. Parallel decoding: $k = 2^r$.

Moreover, these codes are shown to be optimal up to the complementation method described originally in [Knu 86]. In a sense, when Knuth's method is used to construct balanced codes, one can do no better than the codes designed here.

Balanced Codes - with Error Correction

Single and double-error correcting balanced codes (1-EC/B and 2-EC/B) have been designed by [Bos 82a] and [Kun 88], respectively. Design of 3 and 4-error correcting balanced codes, 3-EC/B and 4-EC/B, were given independently in [Sai 88] and [Kun 90] and [Sai 88], respectively. Other t-EC/B codes with constraint on the run length and the

disparity† are designed in [Etz 90, Til 89]. All these codes are also non-systematic and, in general, have no easy encoding and/or decoding procedures.

In Chapter 3, new classes of t-EC/B codes are given, for $1 \leq t \leq 4$. These codes are based on the same complementation principle, i.e. a few bits of the information symbol are complemented and an appropriate check symbol is appended to form the final codeword. These codes are in many cases superior to the existing code in both the information rate *and* in encoding/decoding simplicity.

Non-Balanced DC-Free Codes

Deng and Herro in 1988, used coset codes to design error-correcting dc-free codes [Den 88]. The coset codes are derived by partitioning linear block codes and have about the same encoding and decoding complexity as linear block codes. Although these coset codes are not balanced, they still maintain low disparity and short run length. The idea is based on having two representations for every information symbol, one with high disparity (i.e. with weight $\geq n/2$) and another with low disparity (i.e. with weight $\leq n/2$). A disparity register at the encoder circuit decides wether the next information symbol should be encoded to a high or low disparity codeword in order to neutralize the running accumulated disparity.

In Chapter 4 we design new dc-free coset codes that have similar properties as the codes in [Den 88] but the new codes contain much higher transitions density and have smaller run length.

---

† The run length is the maximum number of consecutive 1's (or 0's) in any code word and the disparity is the difference between the number of 1's and 0's in a code word.

Extensions to Balanced Codes

In Chapter 5 we give three generalization of balanced codes. The first, and the most important, is the design of error-correcting balanced codes that have low dc level and many transitions. In an ordinary balanced codeword of length n, there will be at least one (1 to 0 or 0 to 1) transition and the maximum disparity and run length will both be at most n/2. We shall see that using an extra check bit, one can design balanced codes with n/2 transitions, maximum disparity of n/4, and run length of n/4. These codes are more attractive for optical communication than ordinary balanced codes. Secondly, non-binary balanced codes are designed; in these codes the alphabet is assumed to be {0, 1, ..., q-1}. When q = 2, these degenerate to the ordinary balanced codes. Finally, we design semi-balanced codes in the sense that the number of 1's and 0's in every codeword differ by at most m, say; i.e., the weight of any codeword will be between $\lceil (n-m)/2 \rceil$ and $\lceil (n+m)/2 \rceil$.

Balanced Codes - As t-EC/AUED Codes

It will be shown that every error-correcting balanced code can be also viewed as a t-EC/AUED (t error correcting / all unidirectional error detecting) code. The design of efficient t-EC/B codes will also yield efficient t-EC/AUED codes. The t-EC/AUED codes have wide applications in VLSI circuits and have been studied extensively in [Bos 82a, Bos 82b, Bru 89, Kun 88, Kun 90, Mon, Nik 86, Pra 80a, Sai 88, Tao 88].

In Chapter 6, the technique described in Chapter 4 is used to design efficient t-EC/AUED *coset* codes. These codes considerably improve the best known t-EC/AUED *coset* codes [Bru 89]. Also, in some cases, the new *systematic* t-AEC/AAED codes give higher information rate than the best-known *systematic* t-AEC/AAED codes.

In Chapter 6 we also develop the theory and design of codes that correct t asymmetric

errors and *simultaneously* detect d (d>t) asymmetric errors (t-AEC/d-AED codes). There seems to be no direct relation between these codes and balanced codes; the motivation of the design has come from realizing that the existing codes for the complete asymmetric channel can either correct or detect asymmetric errors but not both [Ber 61, Bor 82, Con 79, Fre 62, Var 73, Web 88]. The new codes have high information rate and an encoding/ decoding complexity comparable to the existing error correcting codes. Finally, in Chapter 6 we show that in some cases we can improve the t-EC/d-UED (t symmetric error correcting and d (d>t) unidirectional error detecting) codes given in [Lin 88].

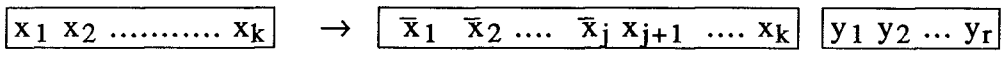
In the last chapter we give a brief summary and pose some unsolved questions for future research.

Chapter 2

Balanced Codes

2.1 Preliminaries and Design Principles

Recall that every codeword in a balanced code C of length n has equally many 1's and 0's, i.e. $W(X) = \lceil n/2 \rceil$ for all $X \in C$. Knuth in [Knu 86] has proposed a simple and efficient construction of balanced codes. These codes have high information rate as well as simple and fast encoding and decoding algorithms. In his construction, the encoding and decoding require only a complementation of some bits of the information word, starting from the beginning. Suppose that the binary information symbol X of length k is represented as $x_1\ x_2\ ....\ x_k$. The balanced codeword will consist of X with the first j bits complemented (j may be 0) and a check symbol of length r. The encoding is depicted in the following diagram.



information symbol balanced codeword

Serial and *parallel* decoding schemes were given in [Knu 86]. In serial decoding, the check symbol determines the original weight of the information word, so the decoder complements one bit at a time until it recovers the original information word. On the other hand, in parallel decoding the check symbol determines the number of the complemented bits, so the decoder simultaneously complements that many bits to recover the original information word. Clearly, parallel decoding schemes are faster than the serial ones.

Recall that DC(n,k) denotes a balanced code of length n and 2^k codewords. In the serial (and parallel) DC(k+r,k) codes designed in [Knu 86] the number of information bits were $k = 2^r$ (and $2^r - r - 1$, respectively). One of the questions left in [Knu 86] was whether there exist a better serial DC(k+r,k) and whether there exists a parallel DC(k+r,k) code with $k = 2^r$. The *serial* DC(k+r,k) codes have been subsequently improved to $k = 2^{r+1} - r - 2$ in [Bos 87].

In this chapter, we construct a *serial* DC(k+r,k) code with $k = 2^{r+1} - 0.8\sqrt{r} - 2$ and a *parallel* DC(k+r,k) code with $k = 2^r$ (or $2^r - 1$) for r even (or odd). These codes have the same encoding and decoding complexities as those of [Knu 86]. We also show that these are the best codes that can designed using the method of "complementation" described here and in [Knu 86].

Table 2.1 lists the new serial DC(k+r,k) codes compared to those in [Bos 87] and Table 2.2 compares the parallel DC(k+r,k) codes to those in [Knu 86], for some values of r.

r	k	
	[Bos 87]	proposed
3	11	12
4	26	28
5	57	60
6	120	124
7	247	251
8	502	507
9	1013	1019
10	2036	2043

Table 2.1 Some improved serial DC(k+r,k) codes.

r	k	
	[Knu 86]	proposed
4	11	16
5	26	31
6	57	64
7	120	127
8	247	256
9	502	511
10	1013	1024

Table 2.2 Some improved parallel DC(k+r,k) codes.

Recall that if X is a binary vector of length k, represented as $(x_1\ x_2\ , ...,\ x_k)$, then $X^{(j)}$ is defined to be X with its first j bits complemented, i.e. $X^{(j)} = (\ \overline{x}_1, ..\ ,\ \overline{x}_j, x_{j+1}, ..\ , x_k)$.

Let $\sigma_j(X) = W(X^{(j)}) = W(\ \overline{x}_1, ...\ ,\ \overline{x}_j, x_{j+1}, ...\ , x_k)$. For example, if X = 1100 0000 then $\sigma_5(00111000) = W(00111000) = 3$. Let W(X) = w; then the function $\sigma(X)$ satisfies the following properties:

$\sigma_0 = w$

$\sigma_k = k - w$

$\sigma_j = \sigma_{j-1} \pm 1$ for $0 < j \leq k$;

and in particular, for any i,j,

$\sigma_j \in [\ \sigma_i - |\ j - i\ |\ ,\ \sigma_i + |\ j - i\ |\ ]$.

where $x \in [a,b]$ means $a \leq x \leq b$ and $x \in (a,b)$ denotes $a < x < b$.

Since the function σ represents a "random walk" from $\sigma_0 = w$ to $\sigma_k = k - w$, one can obtain a word of any weight i in the range between w and k - w (inclusive) from X by complementing some first j bits of X for some $0 \leq j < k$. For example, let X = 01010000 and thus W(X) = 2. We can obtain words of weight 2, 3, 4, 5, and 6 by complementing the first 0, 1, 6, 7, and 8 bits respectively. From the above discussions it is easy to verify the following lemma.

Lemma 2.1 The random walk of any word X of length k and weight w will go through weights w to k - w, inclusive. In other words, given X of weight w then there exists a j such that $\sigma_j(X) = i$ for $\min(w,k-w) \le i \le \max(w,k-w)$. In particular, there always exists a j such that $\sigma_j(X) = \lfloor k/2 \rfloor$ (and $\lceil k/2 \rceil$).

2.2 Balanced Codes -- Serial Decoding

In this section we design serial DC(k+r,k) codes with $k = 2^{r+1} - 0.8\sqrt{r} - 2$, improving upon the codes given in [Bos 87]. In serial balanced codes, the check symbol determines the original weight of the information word. The serial scheme is introduced in the following example.

Example 2.1 Let k = 12 and r = 3 in an 8-out-of-15 code. Suppose the check 010 determines weight 4. Given an information word X, complement the first j bits of X until $\sigma_j(X) = W(X^{(j)}) = 7$. By Lemma 2.1, we see that it is always possible to get weight 7 starting from any word of weight 4. The final codeword will be $X^{(j)}$ 010 which is of weight 8. For example if X = 0000 1111 0000 then after complementing 3 bits we get $X^{(3)}$ = 1110 1111 0000 which is of weight 7; the final codeword will be 1110 1111 0000 010. In decoding, the check symbol will indicate the original weight of the information part. The original information symbol can be reconstructed by complementing $X^{(j)}$ until it reaches that specific weight. For instance, when 1110 1111 0000 010 is received, the check symbol 010 indicates weight 4 and so the information part "1110 1111 0000" is complemented until it reaches weight 4.

□

We will soon see that a check symbol can balance 1 or 2 weights (but not more). It is necessary to ensure that such encoding and decoding will always be correct for all the information words. We first develop the necessary and sufficient conditions for a these

maps to be one-to-one, i.e. to have correct encoding and decoding. Given these conditions, we derive the maximum number of information bits achievable in such code. The code can be designed simply by listing the different check symbols and the weights they balance. Of course, every weight between 0 and k must be balanced by at least one check symbol.

2.2.1 Conditions of Correct Coding

Suppose that the set $\{0,1\}^k$ of information symbols is partitioned into k+1 classes S_0, S_1, ..., S_k where $S_i = \{u \mid W(u) = i\}$. The check symbols are assigned on the basis of the weights of the information symbol, i.e. all information symbols with equal weights will have the same check. Moreover, as we shall see, a check symbol "ch" can be a check for at most two different classes S_i and S_j. Once the check ch is appended to an information symbol u, successively complement the first 0, 1, 2, ... bits of "u ch" until the entire word is balanced. The decoding process is also simple. When the codeword is received, the check ch will indicate the original weight(s) and so the information part is complemented until one of these weights is obtained.

The encoding and decoding process is captured in the notation:

$$H :: S_{v_1} \cup S_{v_2} \cup \cdots \cup S_{v_q} \rightarrow S_v \text{ where } H \in \{0,1\}^r \text{ is a check symbol.}$$

This notation means that every word of weight v_1, v_2, $\cdots$, or v_q, (i.e. every word in $S_{v_1} \cup S_{v_2} \cup \cdots \cup S_{v_q}$), is complemented one bit at a time until its weight becomes v, and then the check symbol H is appended to it to get the complete codeword. The code is balanced if $v = \lceil (k+r)/2 \rceil - W(H)$. The decoder, on the other hand, extracts the check symbol and complements the rest of the codeword one bit at a time until one of the weights v_1, v_2, $\cdots$, or v_q is obtained. The map must be one-to-one to get correct coding.

Example 2.2 Let k = 12 and r = 3. If we let 000 determine the weights 0 and 12, then both weights must be changed to 8 to get a balanced code, i.e. $000 :: S_0 \cup S_{12} \rightarrow S_8$. So the word 0000 0000 0000 will be encoded to 1111 1111 0000 000, and the word 1111 1111 1111 will encode to 0000 1111 1111 000. Both codewords will be decoded correctly because one bit is complemented at a time until weight 0 or 12 is reached. □

We would like to find the criteria that assure correct coding, i.e. that the map is one-to-one. The case when q = 1 the single map, say $H :: S_a \rightarrow S_v$, is obtained. The following Theorem states the necessary and sufficient conditions for the single map to be 1-1.

Theorem 2.2 [Knu 86]: The (single) map $H :: S_a \rightarrow S_v$ is 1-1 iff $\min(a,k-a) \leq v \leq \max(a,k-a)$. □

Notice that $\min(a,k-a) \leq v \leq \max(a,k-a)$ is equivalent to either $a \leq \min(v,k-v)$ or $a \geq \max(v,k-v)$.

Example 2.3 When k = 12 and r = 3, the map $110 :: S_4 \rightarrow S_6$ is 1-1 since $\min(4,12-4) \leq 6 \leq \max(4,12-4)$. □

In the case when q = 2, referred to as a double map, we have $H :: S_a \cup S_b \rightarrow S_v$. Assume, without loss of generality, that b > a. Observe that if the random walk† of some word V, of weight v, reaches weights a and b, then the map will result in ambiguous coding. The reason is that the words of weight a and b, say A and B, will both encode to V.

† The random walk (defined in [Knu 86]) is the set of points $(i,W(X^i))$ for $0 \leq i \leq k$ where X^i is X with the first i bits complemented

Example 2.4 Consider the map 010 :: $S_4 \cup S_9 \rightarrow S_7$, when k = 12 and r = 3. The random walk of V = 1110 0000 1111, of weight 7, reaches weights 4 and 9, producing the words A = $V^{(3)}$ = 0000 0000 1111 and B = $V^{(8)}$ = 0001 1111 1111. So both A and B will encode to the same word V in the map 010 :: $S_4 \cup S_9 \rightarrow S_7$.

□

Lemma 2.3 If H :: $S_a \cup S_b \rightarrow S_v$ (where b > a), then b ≥ max(v,k-v) and a ≤ min(v,k-v) are necessary conditions for correct encoding and decoding.

Proof: First we show that b > k/2 and a < k/2. If b > a ≥ k/2, then the random walk of $V = 0^{k-v}\,1^{v}$ will reach weights a and b. Similarly, when k/2 ≥ b > a, the random walk of $V = 1^{v}\,0^{k-v}$ will reach weights a and b. By Theorem 2.2, b > k/2 implies b ≥ max(v,k-v) and a < k/2 implies a ≤ min(v,k-v).

□

Let the path V → A → B → V' be the random walk of some word V of weight v changing to a word A of weight a, to a word B of weight b, and finally to a word V' of weight k-v. We will denote such a path by the quadruple (v,a,b,k-v), and similarly, we denote the path V → B → A → V' by (v,b,a,k-v). In example 2.4, V can have the path (7,4,9,5). Let mbits(v,a,b,k-v) be the minimum number of bits needed to construct such a path. At least |v-a| bits need to be complemented to get from weight v to weight a, |b-a| bits to get from weight a to b, and |b-(k-v)| bits from weight b to k-v. So mbits(v,a,b,k-v) ≥ |v-a| + |b-a| + |b-(k-v)|. By Lemma 2.3, a ≤ v and b ≥ k-v, so

mbits(v,a,b,k-v) ≥ 2(b-a) + 2v - k, and similarly

mbits(v,b,a,k-v) ≥ 2(b-a) + 2(k-v) - k.

The path (v,a,b,k-v) is not possible when mbits(v,a,b,k-v) > k. So the path (v,a,b,k-v) is possible only if b-a ≤ k-v. Similarly the path (v,b,a,k-v) is possible only if b-a ≤ v. This is stated in the following Lemma.

Lemma 2.4 The path (v,a,b,k-v) is possible iff b-a ≤ k-v, and the path (v,b,a,k-v) is possible iff b-a ≤ v. In particular, one of the paths (v,a,b,k-v) or (v,b,a,k-v) is possible iff (b-a) ≤ max(v,k-v).

Proof: We will first show that the path (v,a,b,k-v) is possible iff b-a ≤ k-v. Suppose the path (v,a,b,k-v) is possible. If b-a > k-v, then mbits(v,a,b,k-v) ≥ 2(b-a) + 2v - k > k, i.e. the path (v,a,b,k-v) is not possible, a contradiction, so b-a ≤ k-v.

On the other hand, suppose that b-a ≤ k-v. Then the random walk of $V = 1^{v-a}\,0^{k-v}\,1^{a}$ will construct the path (v,a,b,k-v) as follows:

$$1^{v-a}\,0^{k-v}\,1^{a} \rightarrow 0^{k-a}\,1^{a} \rightarrow 0^{v-a}\,1^{b-a}\,0^{(k-v)-(b-a)}\,1^{a} \rightarrow 0^{v-a}\,1^{k-v}\,0^{a}.$$

The proof for (v,b,a,k-v) is similar. Combining the two parts, it follows that the path (v,a,b,k-v) or (v,b,a,k-v) is possible iff (b-a) ≤ max(v,k-v). □

In example 2.4, b-a ≤ max(v,k-v), i.e. 9-4 ≤ max(7,12-7), and therefore the path (v,a,b,k-v), i.e. (7,4,9,5), is possible. Now we state the main Theorem.

Theorem 2.5 Let $H :: S_a \cup S_b \rightarrow S_v$ (where b > a). Then the map is 1-1 iff b-a > max(v,k-v).

Proof: Suppose that b-a > max(v,k-v). If the map is not 1-1, then there exist two distinct words of weight a and b, say A and B, that will encode to the same word V. Suppose $A^{(i)} = V$ and $B^{(j)} = V$ (Recall that $X^{(s)}$ is X with the first s bits complemented). Since $V^{(i)} = A$ and $V^{(j)} = B$, the path (v,a,b,k-v) (or (v,b,a,k-v)) is possible if i < j (or i > j), . This contradicts Lemma 2.3, since b-a > max(v,k-v). So the map must be 1-1.

Conversely, suppose the map is 1-1. If b-a ≤ max(v,k-v), then by Lemma 2.4,, we have a path (v,a,b,k-v) or (v,b,a,k-v); so both A and B encode to V, contradicting the

assumption that the map is 1-1. Therefore b-a > max(v,k-v).

□

In example 2.3, the map $000 :: S_0 \cup S_{12} \rightarrow S_8$ is 1-1 since 12-0 > max(8,12-8).

Theorem 2.5 also implies that $q \leq 2$, since $\max(v,k-v) \geq k/2$. For instance, when q = 3, say $H :: S_a \cup S_b \cup S_c \rightarrow S_v$, there exist no integers $0 \leq a < b < c \leq k$ such that $b-a \geq k/2$, $c-b \geq k/2$, and $c-a \geq k/2$.

The problem now reduces to finding the best combination of single maps satisfying Theorem 2.2 and double maps satisfying Theorem 2.5. There are 2^r different maps corresponding to the 2^r check symbols. When d check symbols are used for single maps and 2^r - d check symbols are used for double maps, the number of different weights that can be recognized is $2(2^r - d) + d = 2^{r+1} - d$, so $k = 2^{r+1} - d - 1$.

2.2.2 Code Optimality

To maximize $k = 2^{r+1} - d - 1$, d must be minimized. In the construction given in [Bos 87], d was r+1; in this section we show that $d = 0.8\sqrt{r} - 1$.

Suppose that at least d ($d \leq r+1$) single maps are needed to construct the optimal code. Let the single maps be

$H_1 :: S_{a_1} \rightarrow S_{v_1}$,

$H_2 :: S_{a_2} \rightarrow S_{v_2}$,

:

$H_d :: S_{a_d} \rightarrow S_{v_d}$

and the double maps be

$H_{d+1} :: S_{a_{d+1}} \cup S_{b_{d+1}} \rightarrow S_{v_{d+1}}$,

$H_{d+2} :: S_{a_{d+2}} \cup S_{b_{d+2}} \rightarrow S_{v_{d+2}}$,

:

$$H_{2r} :: S_{a_{2r}} \cup S_{b_{2r}} \rightarrow S_{v_{2r}}$$

where the set $\{a_1, \ldots, a_{2r}, b_{d+1}, \ldots, b_{2r}\} = \{0, \ldots, k\}$ and $\{H_1, \ldots, H_{2r}\} = \{0,1\}^r$.

By Lemma 2.3, $a_i < k/2$ and $b_i > k/2$ for $d+1 \le i \le 2^r$. The condition $\min(a_i,k-a_i) \le v_i \le \max(a_i,k-a_i)$ for $1 \le i \le d$ and the condition $b_i-a_i > \max(v_i,k-v_i)$ (or $b_i-a_i \ge \max(v_i,k-v_i) + 1$), for $d+1 \le i \le 2^r$ must be satisfied. There are $\binom{r}{i}$ maps with $v_i = \lceil (k+r)/2 \rceil - i$. Without loss of generality, we choose the d single maps to be $H_i :: S_{v_i} \rightarrow S_{v_i}$ for $1 \le i \le d$, where $v_i = (k-d-1)/2 + i = 2^r - d - 1 + i$, i.e. $W(H_i) = \lceil (r+d+1)/2 \rceil - i$. The double maps must satisfy

$$\sum_{i=d+1}^{2^r} b_i - a_i \ge \sum_{i=d+1}^{2^r} \max(v_i,k-v_i) + 1, \text{ or}$$

$$\sum_{i=d+1}^{2^r} b_i - a_i \ge \sum_{i=1}^{2^r} \max(v_i,k-v_i) + 1 - \sum_{i=1}^{d} \max(v_i,k-v_i) + 1 \qquad (2\text{-}1)$$

But,

$$\sum_{i=d+1}^{2^r} b_i - a_i = \sum_{i=d+1}^{2^r} b_i - \sum_{i=d+1}^{2^r} a_i = \{(k+d+1)/2 + \ldots + k\} - \{0 + \cdots + (k-d-1)/2\} = 2^r (2^r - d).$$

And, $\sum_{i=1}^{2^r} \max(v_i,k-v_i) + 1 = 2^r (k/2+1) + T(r) = 2^r (2^r - (d-1)/2) + T(r)$, where

$T(r) = r/2 \binom{r}{r/2} + \text{odd}(k)/2 \binom{r}{r/2}$ r even,

$T(r) = (r+1)/4 \binom{r+1}{(r+1)/2}$ r odd.

Finally, $\sum_{i=1}^{d} \max(v_i,k-v_i) + 1 = d\, 2^r - (d^2-2d+\text{odd}(d))/4$.

After substituting in (2-1) and simplifying, we get

$$(d^2-2d-\text{odd}(d))/4 - (d-1)\, 2^{r-1} + T(r) \le 0 \qquad (2\text{-}2)$$

We attempt to get an approximation to d. First, notice that the term $(d^2-2d-\text{odd}(d))/4$ is negligible compared to the exponential term $(d-1)\, 2^{r-1}$. Dropping this term we get:

$$d \ge 1 + T(r) / 2^{r-1} \qquad (2\text{-}3)$$

Furthermore, using Stirling's formula, $x! \cong \sqrt{2\pi x}\,(x/e)^x$, to approximate T(r), we get

$$T(r) = \sqrt{r/2\pi}\;\; 2^r + \text{odd}(k)\;\sqrt{1/\pi r}\;\; 2^r \qquad r \text{ even, and}$$

$$T(r) = \sqrt{(r+1)/2\pi}\;\; 2^r \qquad r \text{ odd.}$$

Approximating T(r) by $\sqrt{r/2\pi}\;\; 2^r$ in (2-3), we get

$$d \geq 1 + 0.8\sqrt{r} \tag{2-4}$$

The approximated values in (2-4) are the same as the real ones, for $r \leq 35$, except when $r = 14$ where the approximated value is smaller by 1. This is negligible compared to 2^{14+1} when used in approximating k, as in

$$k \cong 2^{r+1} - 0.8\sqrt{r} - 2 \tag{2-5}$$

The information rate of this DC(k+r,k) code which is $\frac{k}{k+r}$ rapidly approaches unity as r increases.

2.2.3 Code Design

In order to understand the construction we introduce another way of obtaining the minimum number of single maps (d). Let the number of different check symbols with weight i be $g_i = \binom{r}{i}$ for $0 \leq i \leq r$. For any integer $d \geq 1$, let $i' = \lfloor r/2 \rfloor - \lfloor (d\text{-odd}(r))/2 \rfloor$ and $i'' = \lfloor r/2 \rfloor + \lfloor (d\text{-even}(r))/2 \rfloor$. Also, let the vector **h** be defined as follows:

$$\begin{aligned} h_i &= i' - i \quad \text{for } 0 \leq i \leq i' + \lceil (d-2)/2 \rceil \\ &= i - i'' \quad \text{for } i'' - \lfloor (d-2)/2 \rfloor \leq i \leq r. \end{aligned}$$

The level h_i is a measure of how far the checks are from the center; the closer the check to the center, i.e. to k/2, the smaller is h_i and the smaller is $\max(v_i, k-v_i)$. The checks further away from the center will have a larger $\max(v_i, k-v_i)$ and will be more restrictive in

the map assignment. Finally, let the vector **x** be defined as $x_i = 1$ if $i' \le i \le i''$, and 0 otherwise. The minimum number of single maps needed is the least d such that

$$(\mathbf{g}-\mathbf{x}).\mathbf{h} \le 0 \qquad (2\text{-}6)$$

The least d satisfying (2-6) is equivalent to the least d satisfying (2-2). Actually, T(r) of equation (2-2) is $(\mathbf{g}-\mathbf{x}).\mathbf{h}$ when d = 1.

Example 2.5 Let r = 4 and d = 1 to construct a DC(34,30) balanced code. In Figure 2.1, we can see that $(\mathbf{g}-\mathbf{x}).\mathbf{h} = 12 > 0$, so it is not possible to construct a DC(34,30) code.

	i	g_i	x_i	h_i	$(g_i-x_i)\,h_i$
	0	1	0	2	2
	1	4	0	1	4
i' = i"	2	6	1	0	0
	3	4	0	1	4
	4	1	0	2	2

Figure 2.1 Serial DC(34,30) code parameters.

However, by (2-3), $d \ge 1 + 12 / 2^3$ (or by (2-4) $d \ge 1 + 0.8\sqrt{4}$), we get d = 3. When d = 3, then $(\mathbf{g}-\mathbf{x}).\mathbf{h} = -3 \le 0$, as in Figure 2.2. So d = 3 satisfies (2-6), and hence k can be up to 32-3-1 = 28, so a DC(32,28) code can be constructed. Notice that, if d = 2 then $(\mathbf{g}-\mathbf{x}).\mathbf{h} = 7 > 0$ which does not satisfy (2-6), hence 3 is the optimal value of d. Figure 2.3 lists the (single and double) maps for the DC(32,28) code.

	i	g_i	x_i	h_i	$(g_i-x_i)\,h_i$
	0	1	0	1	1
i'	1	4	1	0	0
	2	6	1	-1	-5
i"	3	4	1	0	0
	4	1	0	1	1

Figure 2.2 Serial DC(32,28) code parameters.

H	b	a	v	max(v,k-v)	b-a
0111	---	13	13	15	---
0011	---	14	14	14	---
0001	---	15	15	15	---
0101	16	1	14	14	15
0000	17	0	16	16	17
0110	18	3	14	14	15
1111	19	2	12	16	17
0010	20	4	15	15	16
0100	21	5	15	15	16
1000	22	6	15	15	16
1001	23	7	14	14	16
1010	24	8	14	14	16
1100	25	9	14	14	16
1011	26	10	13	15	16
1101	27	11	13	15	16
1110	28	12	13	15	16

Figure 2.3 Check symbols in the serial DC(32,28) code.

It can be readily verified that the single maps, corresponding to when x_i is 1 in Figure 2.2, satisfy Theorem 2.2, and the double maps satisfy Theorem 2.5. Notice that one check symbol from level -1 (i.e. of weight 2) was used in combination with one check symbol from level 1 (i.e. of weight 0 or 4), as in

0101	16	1	14	14	15
0000	17	0	16	16	17

and

0110	18	3	14	14	15
1111	19	2	12	16	17

So, all the check symbols at level 1 (i.e. needing b-a ≥ 17) are consumed. The remaining check symbols all have level 0 or -1, i.e. need b-a ≥ 15 or b-a ≥ 16, but b-a = 16 = 2^r can always be chosen from there on, i.e. 20-4, 21-5, ⋯ , 28-12. Notice that,

$$(\mathbf{g}\text{-}\mathbf{x}).\mathbf{h} = \sum_{i=d+1}^{2^r} \max(v_i, k-v_i) + 1 - \sum_{i=d+1}^{2^r} b_i - a_i = -3.$$

In general, j check symbols of level -i can be used in combination with i check symbols of level j. So, after assigning the single maps, the symbols from the highest level are matched with the symbols from the lowest level until the largest level symbols are all consumed. The process is repeated with the symbols on the next lower level, and so on. When level 0 is reached, the combination b-a = 2^r is used for the rest of the double maps. Note that if $(\mathbf{g}\text{-}\mathbf{x}).\mathbf{h} > 0$ then the above process is not possible.

2.3 Balanced Codes -- Parallel Decoding

In this section, parallel DC(k+r,k) codes with k = 2^r (or 2^r - 1) for r even (or odd) are designed. These codes improve the original construction with k = 2^r - r - 1 given in [Knu 86]. In parallel decoding, the check symbol directly indicates the number of complemented information bits. The original information word can be obtained by complementing that many bits in one step. Parallel decoding is faster than serial decoding, but is usually require one extra check bit.

These codes are constructed in sub-section 2.3.1. In sub-section 2.3.2 it is shown that these codes are indeed optimal.

2.3.1 Code Design

The parallel DC(k+r,k) codes are described in the following algorithm.

Design Algorithm: Let $A = \{0,1\}^r$ be the set of all 2^r check symbols and $m = \binom{r}{r/2}$. Partition A into subsets $D_1, D_2, ..., D_{m-1}$ as follows:

D_i = a maximal subset of A - $\{D_j \mid j < i\}$ such that if $X,Y \in D_i$ then $W(X) \neq W(Y)$ for $1 \leq i \leq m$.

So D_1 will contain r+1 elements, each with distinct weight $0 \leq i \leq r$, D_2 will contain elements one of each possible weight from A - D_1, and D_3 will contain elements one of each possible weight from A - $(D_1 \cup D_2)$, and so on. It can be easily verified that D_j has elements of weights in $[\lceil r/2 \rceil - \lfloor D_j/2 \rfloor, \lfloor r/2 \rfloor + \lfloor D_j/2 \rfloor]$.† The words in D_j are assigned an integer d_j where

$$d_1 = 0$$

$$d_{j+1} = d_j + \lfloor D_j/2 \rfloor + \lceil D_{j+1}/2 \rceil \text{ for } 1 \leq j < m.$$

Example 2.6 For r = 3 and r = 4 the possible subsets (D_i's) and their corresponding number assignments (d_i's) are shown below.

r = 3:

$D_1 = \{ 000 , 001 , 011 , 111 \}$ $\quad d_1 = 0$

$D_2 = \{ 010 , 101 \}$ $\quad d_2 = 3$

$D_3 = \{ 100 , 110 \}$ $\quad d_3 = 5$

r = 4:

$D_1 = \{ 0000 , 0001 , 0011 , 0111 , 1111 \}$ $\quad d_1 = 0$

$D_2 = \{ 0010 , 0101 , 1011 \}$ $\quad d_2 = 4$

$D_3 = \{ 0100 , 0110 , 1101 \}$ $\quad d_3 = 7$

$D_4 = \{1000 , 1001 , 1110 \}$ $\quad d_4 = 10$

$D_5 = \{1010 \}$ $\quad d_5 = 12$

---

† When it is not ambiguous, we use D_i to represent the size of D_i.

$D_6 = \{1100\}$ $\qquad\qquad$ $d_6 = 13$

□

To encode a given information word of length $k \leq 2^r$ (or $2^r - 1$) for r even (or odd), complement the first d_i bits of the information word and try to assign a suitable check word $Y \in D_i$, for $1 \leq i \leq m$, such that the weight of the codeword is (k+r)/2, i.e. $\sigma_{d_i}(X) + W(Y) = (k+r)/2$. Thus the encoding process is sequential. Note that (k+r)/2 is always an integer because k and r are both even or both odd.

The decoding algorithm is straight-forward. Suppose that the check symbol of the received word belongs to the subset D_i. Then complement the first d_i bits of the received word to get the original information word. The decoding can be done in parallel, i.e. the first d_i bits can all be complemented simultaneously. Note that a table of the check symbols and their d_i's can be used to directly find d_i for any check symbol. This completes the design algorithm.

Example 2.6 (continue) For r = 3 the max number of information bits $k = 2^r - 1 = 7$; we can construct a DC(10,7) code. Let 1000000 be the given information word. The modified words after complementing the first $d_1 = 0$, $d_2 = 3$, and $d_3 = 5$ bits will be 1000000, 0110000, and 0111100 respectively. Now the word obtained after complementing 5 bits can be assigned the check $100 \in D_3$. The codeword will be 0111100 100 and now it is balanced.

Suppose the decoder receives this word. Since the check $100 \in D_3$ the decoder will complement the first $d_3 = 5$ bits to get the original information word.

When r = 4, the number of information bits $k = 2^r = 16$; i.e. producing a DC(20,16) code. Suppose $X = 0000000000000011 = 0^{14}\,1^2$ is the information word. Note that, for i = 1 or 2, there exists no $Y \in D_i$ such that $\sigma_{d_i}(X) + W(Y) = (16+4)/2 = 10$. However, by complementing the first $d_3 = 7$ bits of X and assigning $0100 \in D_3$ as the check, X can be

balanced. Thus the complete codeword is 1111111000000011 0100. To get the original word, at the decoder side, the first $d_3 = 7$ bits are complemented since the check $0100 \in D_3$.

The check symbols assignment for $r = 4$ is depicted in Figure 2.4. The figure also shows how the random walk of the word $0^{14}\ 1^2$ starting from weight 2 and ending in weight 14. Notice how the random walk intersects the check symbol 0100.

□

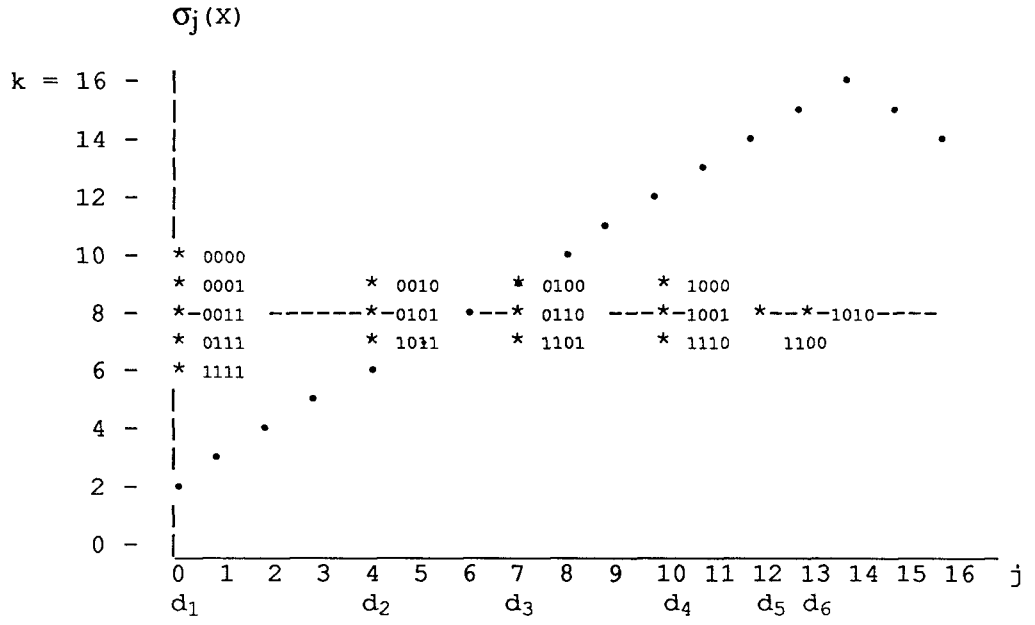


Figure 2.4 Check symbols in the parallel DC(20,16) code.

The correctness of the algorithm is stated in the following theorem.

Theorem 2.6 For $k = 2^r$ (or $2^r - 1$) for r even (or odd), the design algorithm given above gives a balanced code.

□

The proof will be given at the end of this section. In the proof, it will be shown that for any information symbol $X \in \{0,1\}^k$ there exist $Y \in D_i$ such that $\sigma_{d_i}(X) + W(Y) = (k+r)/2$. Assume that $k = 2^r$ (or $k = 2^r - 1$) for r even (or odd) in the rest of this section and consider

the following lemmas.

Lemma 2.7 If $X \in \{0,1\}^k$ and $(k-r)/2 \leq W(X) \leq (k+r)/2$ then there exists a $Y \in D_1$ such that $\sigma_{d1}(X) + W(Y) = (k+r)/2$.

Proof: This is clear because D_1 has check symbols with weights in [0,r]. So, if $W(X) = (k - r)/2 + i$ where $0 \leq i \leq r$, choose the check $Y \in D_1$ with weight $r - i$. Then $\sigma_0(X) + W(Y) = (k+r)/2$, i.e. the codeword XY is balanced.

□

Thus the set of information words with weights $(k-r)/2$, $(k-r)/2 + 1$, $(k-r)/2 + 2$, ... , $(k+r)/2$ need not be complemented at all.

Lemma 2.8 Let $X \in \{0,1\}^k$ such that $W(X) < (k-r)/2$. Then there exists an integer $s \leq d_m$ such that $\sigma_s(X) = \lceil k/2 \rceil$.

Proof: From lemma 2.1 it follows that there exists an $s \in [0,k]$ such that $\sigma_s(X) = \lceil k/2 \rceil$. Thus we need only to prove that $s \leq d_m$. From the code design it can be verified that $k - d_m = \lfloor r/2 \rfloor + 1$. If $s > d_m$ then we get a contradiction as follows. From the property of the function σ we have

$$\sigma_k(X) \in [\, \sigma_s(X) - (k-s) \,,\, \sigma_s(X) + (k-s) \,], \text{ so}$$

$$\sigma_k(X) \in (\lceil k/2 \rceil - (k-d_m) \,,\, \lceil k/2 \rceil + (k-d_m)), \text{ then}$$

$$\sigma_k(X) \in (\lceil k/2 \rceil - \lfloor r/2 \rfloor - 1 \,,\, \lceil k/2 \rceil + \lfloor r/2 \rfloor + 1).$$

So $\sigma_k(X) < \lceil k/2 \rceil + \lfloor r/2 \rfloor + 1$, i.e. $\sigma_k(X) \leq (k+r)/2$, but $\sigma_k(X) > (k+r)/2$ since $W(X) < (k-r)/2$, a contradiction. Therefore, $s \leq d_m$.

□

Lemma 2.9 Let $X \in \{0,1\}^k$ with weight $W(X) < (k-r)/2$. Let s be the smallest integer such that $\sigma_s(X) = \lceil k/2 \rceil$ and $d_j < s \leq d_{j+1}$ for some $1 \leq j < m$. Then there exists a $Y \in D_i$ with $\sigma_{d_i}(X) + W(Y) = (k+r)/2$ where $i = j$ or $j+1$.

Proof: The proof will be split into two cases. It will be shown that if $s \leq d_j + \lfloor D_j/2 \rfloor$ then $Y \in D_j$ and if $s > d_j + \lfloor D_j/2 \rfloor$ then $Y \in D_{j+1}$.

Case 1: $s \leq d_j + \lfloor D_j/2 \rfloor$

From the property of σ, $\sigma_{d_j}(X) \in [\sigma_s(X) - (s-d_j), \sigma_s(X) + (s-d_j)]$. But $\sigma_s(X) = \lceil k/2 \rceil$ and $\sigma_{d_j}(X) < \lceil k/2 \rceil$ by the minimality of s so $\sigma_{d_j}(X) \in [\lceil k/2 \rceil - \lfloor D_j/2 \rfloor, \lceil k/2 \rceil]$.

In order to balance X, D_j must have check symbols with weights in $(\lfloor r/2 \rfloor, \lfloor r/2 \rfloor + \lfloor D_j/2 \rfloor]$. We can balance X since D_j has elements of weights in $[\lceil r/2 \rceil - \lfloor D_j/2 \rfloor, \lfloor r/2 \rfloor + \lfloor D_j/2 \rfloor]$.

Case 2: $d_j + \lfloor D_j/2 \rfloor < s \leq d_{j+1}$

From the design algorithm we have $d_{j+1} = d_j + \lfloor D_j/2 \rfloor + \lceil D_{j+1}/2 \rceil$, so $s > d_j + \lfloor D_j/2 \rfloor$ implies $s > d_{j+1} - \lceil D_{j+1}/2 \rceil$. Recall $\sigma_{d(j+1)}(X) \in [\sigma_s(X) - (d_{j+1}-s), \sigma_s(X) + (d_{j+1}-s)]$ so $\sigma_{d(j+1)}(X) \in (\lceil k/2 \rceil - \lceil D_{j+1}/2 \rceil, \lceil k/2 \rceil + \lceil D_{j+1}/2 \rceil)$.

In order to balance X, D_{j+1} must have check symbols with weights in $(\lfloor r/2 \rfloor - \lceil D_{j+1}/2 \rceil, \lfloor r/2 \rfloor + \lceil D_{j+1}/2 \rceil)$. We can balance X since D_{j+1} has elements of weights in $[\lceil r/2 \rceil - \lfloor D_{j+1}/2 \rfloor, \lfloor r/2 \rfloor + \lfloor D_{j+1}/2 \rfloor]$, and when r is even (odd) then the size of D_{j+1} is odd (even).

□

Lemma 2.10 Let $X \in \{0,1\}^k$ such that $W(X) > (k+r)/2$. Then there exists an integer $s \leq d_m$ such that $\sigma_s(X) = \lfloor k/2 \rfloor$.

Proof: Similar to the proof of lemma 2.8.

Lemma 2.11 Let $X \in \{0,1\}^k$ with weight $W(X) > (k+r)/2$. Let s be the smallest integer such that $\sigma_s(X) = \lfloor k/2 \rfloor$ and $d_j < s \leq d_{j+1}$ for some $1 \leq j < m$. Then there exist $Y \in D_i$ with $\sigma_{d_i}(X) + W(Y) = (k+r)/2$ where $i = j$ or $j+1$.

Proof: Similar to the proof of lemma 2.9.

Proof (of Theorem 2.6): The proof now is straight-forward. Let $X \in \{0,1\}^k$ with weight W(X). If $(k-r)/2 \leq W(X) \leq (k+r)/2$ then X can be balanced by lemma 2.7. If $W(X) < (k-r)/2$ (or $W(X) > (k+r)/2$) then X can be balanced by lemma 2.9 (or lemma 2.11). □

2.3.2 Code Optimality

In this section, the optimality of the code (up to the described complementation method) is considered. The first theorem will state that $r \geq \lceil \log(k)/2 \rceil$ for any k. This implies that the parallel construction when r is even is optimal. The second theorem asserts that $r > \log k$ when $k = 2^r$ and r is odd; this makes the construction for odd r optimal, too. Thus, the balanced codes presented here achieve the highest information rate and yet maintain the same encoding and decoding complexity as [Knu 86].

Theorem 2.12 If k be the number of information bits, then at least k check symbols are needed to obtain any *constant weight* code using the parallel decoding scheme described.

Proof: Let $S = \{ 1^i 0^{k-i}, 0^i 1^{k-i} \mid 1 \leq i \leq k \}$. Claim: at most two random walks in S (or simply two words in S) intersect any particular check point; i.e. a check symbol can be assigned to at most two words in S. Once this claim is shown then it is clear that one needs at least $|S|/2 = 2k/2 = k$ check symbols, i.e. $r \geq \lceil \log(k)/2 \rceil$, when parallel decoding is used.

Let ck be a check symbol complementing s bits. If ck is assigned to two information words $X,Y \in S$ to get a constant weight code, then $\sigma_S(X) + W(ck) = \sigma_S(Y) + W(ck)$ and hence $\sigma_S(X) = \sigma_S(Y)$. So it is sufficient to prove that at most two words X and Y in S intersect any point (s,w) i.e. satisfying $\sigma_S(X) = \sigma_S(Y) = w$.

Observe that $\sigma_t(1^i 0^{k-i}) = |t-i|$ and $\sigma_t(0^i 1^{k-i}) = k - |t-i|$ for $1 \le i \le k$. So $\sigma_s(1^i 0^{k-i}) = w$ implies $s-i = w$ or $-s+i = w$, i.e. $i = s-w$ or $i = s+w$. Similarly, $\sigma_s(0^i 1^{k-i}) = w$ implies $j = (s-w) + k$ or $j = (s+w) - k$. Moreover, if $(s-w) \in [1,k]$ then $(s-w) + k > k$ and, conversely, if $(s-w) + k \in [1,k]$ then $(s-w) < 1$. So it is not possible that both $\sigma_t(1^{(s-w)} 0^{k-(s-w)})$ and $\sigma_t(0^{(s-w)+k} 1^{(s-w)})$ intersect (s,w). Similarly, it is not possible that both $\sigma_t(1^{(s+w)} 0^{k-(s+w)})$ and $\sigma_t(0^{(s+w)-k} 1^{2k-(s+w)})$ intersect (s,w). Therefore, at most two random walks will intersect any point (s,w).

□

Theorem 2.13 If $k = 2^r$ information bits and r is odd then at least k+1 check symbols are needed to obtain any constant weight code using the parallel decoding scheme described.

□

The proof of Theorem 2.13 is harder than that of Theorem 2.12 so before we give the proof we state one corollary to Theorem 2.12 and one lemma.

Corollary 2.14 Let k be the number of information bits with k check symbols forming a constant-weight code using the parallel decoding scheme described.

Then every word in $S = \{ 1^i 0^{k-i}, 0^i 1^{k-i} \mid 1 \le i \le k \}$ intersects *exactly* one check point.

Proof: Every word in S must intersect at least one check point since a constant-weight code is assumed. From the proof of Theorem 2.12 we know that a check symbol can balance at most two words out of the 2k words in S. So every word in S must intersect exactly one check symbol.

□

As an example, the random walks of every word in $S = \{ 1^i 0^{16-i}, 0^i 1^{16-i} \mid 1 \le i \le 16 \}$ intersects exactly one check point.

Before stating the lemma, some more insight in the check point allocation is needed. Suppose the desired constant-weight code is c-out-of-(k+r). If $c < k/2$ then it is not possible to encode the word $0^{k/2}\ 1^{k/2}$ since $\sigma_j(0^{k/2}\ 1^{k/2}) \geq k/2$ for all j. Similarly, if $c > k/2+r$ then it is not possible to encode the word $1^{k/2}\ 0^{k/2}$ since $\sigma_j(1^{k/2}\ 0^{k/2}) \leq k/2$ for all j. Therefore, $k/2 \leq c \leq k/2+r$.

Moreover, in a c-out-of-n code the check points must lie between the c-r and c lines inclusive. So in any constant-weight code the check points must lie in between k/2-r and k/2+r. All check points lie inside the *Space*, as defined below; and an information word intersects a check point only inside this *Space*.

Definitions:

Let $X \in \{0,1\}^k$ be an information symbol, $a = k/2 - r$, and $b = k/2 + r$, then

Space = $\{ (x,y) \mid 0 \leq x \leq k \text{ and } a \leq y \leq b \}$

$L(X) = \{ (j, \sigma_j(X)) \mid 0 \leq j \leq k , (j, \sigma_j(X)) \in$ *Space* $\}$

$ur(i) = \{ (x,y) \mid y \leq x+(b-i) , y \leq -x+(b+i) , (x,y) \in$ *Space* $\}$

$lr(i) = \{ (x,y) \mid y \geq x+(a-i) , y \geq -x+(a+i) , (x,y) \in$ *Space* $\}$

$ut(i) = ur(i) \cup lr(i+k) \cup lr(i-k)$

$lt(i) = lr(i) \cup ur(i+k) \cup ur(i-k)$

Let M be ut(i) , lt(i) , or L(X), then define

$| M | = | \{ (x,y) \mid (x,y) \in M , (x,y)$ is a check point $\} |$; i.e.,

the number of check symbols inside M.

Lemma 2.15 Let $k \geq 8$ be the number of information bits. Let k check symbols form a constant-weight code using the parallel decoding scheme described. Then $| ut(i) | = | lt(j) |$ for $0 \leq i,j \leq k$.†

† Actually, it can be shown that $| ut(i) | = | lt(i) | = 2r + 1$ for $0 \leq i \leq k$.

Proof: First, we will show that $|ut(i)| = |ut(j)|$ and $|lt(i)| = |lt(j)|$ for all i,j. Then $|ut(i)| = |ut(j)|$ will follow from the fact that $ut(\theta) = lt(k)$ (or $ut(k) = lt(0)$).

The proof for $|lt(i)| = |lt(j)|$ is similar to that of $|ut(i)| = |ut(j)|$, so we will only prove $|ut(i)| = |ut(j)|$. To prove that $|ut(i)| = |ut(j)|$ it is sufficient to show that $|ut(i)| = |ut(i+1)|$ for $0 \le i < k$. Consider the region $R = ut(i) \cup ut(i+1)$.

Then $R = ut(i) \cup L(w_1)$ where $w_1 = \begin{cases} 1^{i+b+1}\,0^{k-i-b-1} & \text{if } 0 \le i < a \\ 0^{i-a+1}\,1^{k-i+a-1} & \text{if } a \le i < k \end{cases}$

and $ut(i) \cap L(w_1) = \varnothing$ so $|R| = |ut(i)| + |L(w_1)|$. But $|L(w_1)| = 1$ since $w_1 \in S$ (Corollary 2.14). Therefore,

$$|R| = |ut(i)| + 1 \tag{2-7}$$

Similarly, $R = ut(i+1) \cup L(w_2)$ where $w_2 = \begin{cases} 0^{k-b+i}\,1^{b-i} & \text{if } 0 \le i < b \\ 1^{i-b}\,0^{k-i+b} & \text{if } b \le i < k \end{cases}$

and $ut(i+1) \cap L(w_2) = \varnothing$ so $|R| = |ut(i+1)| + |L(w_2)|$. Again $|L(w_2)| = 1$ since $w_2 \in S$, hence

$$|R| = |ut(i+1)| + 1 \tag{2-8}$$

From (2-7) and (2-8) it can be readily implied that $|ut(i)| = |ut(i+1)|$.

□

Proof: (of Theorem 2.13) When $r = 1$ and $k = 2^1$ it can be easily shown that at least $k + 1 = 3$ check symbols are needed. Assume that $k = 2^r$ where $r > 1$ and r is odd. We will prove by contradiction that $k (= 2^r)$ check symbols *can not* form a constant-weight code. Suppose it is possible, then by lemma 2.15 $|ut(i)| = |lt(i)|$; in particular,

$$\sum_{i=0}^{k-1} |ut(i)| = \sum_{i=0}^{k-1} |lt(i)|.$$

Only the term $\sum_{i=0}^{k-1} |\, ut(i) \,|$ is analyzed because $\sum_{i=0}^{k-1} |\, lt(i) \,|$ is similar. In $\sum_{i=0}^{k-1} |\, ut(i) \,|$ the number of check points inside each *ut*, excluding *ut(k)*, are summed. A check point can be counted more than once in the sum because it may be a member of more than one *ut*, e.g. if $ch \in ut(i+1) \cap ut(i+2) \cap \ldots \cap ut(i+j)$, then ch is counted j times in the sum.

We attempt to count the same sum differently, now over the set of check points rather than over the *ut*'s. First, define no_ut(ch) as follows

$$no\_ut(ch) = |\, \{\, ut(i) \mid ch \in ut(i),\ i \neq k \,\} \,|, \text{ and similarly}$$

$$no\_lt(ch) = |\, \{\, lt(i) \mid ch \in lt(i),\ i \neq k \,\} \,|$$

It can be seen that summing over the check points is the same as summing over the *ut*'s. Therefore,

$$\sum_{\forall\ ch} no\_ut(ch) = \sum_{i=0}^{k-1} |\, ut(i) \,|$$

To find no_ut(ch), consider a check point of the form ch = (x, k/2+i) then

$$ch \in ut(j) \quad \text{for } x - (r-i) \leq j \leq x + (r-i)$$

where addition (subtraction) is mod k†. So,

$$no\_ut((x, k/2+i)) = 2r + 1 - 2i \quad \text{and similarly}$$

$$no\_lt((x, k/2+i)) = 2r + 1 + 2i$$

Note that no_ut(ch) (and no_lt(ch)) does not depend on the first component, x of ch. Recall that in a c-out-of-n code the check points lie between c-r and c lines, inclusive. In fact, there will be $\binom{r}{i}$ check points of the form (x, c-i) for $0 \leq i \leq r$. And no_ut(x,c-i) = $(2r + 1) - 2(c - k/2 - i)$, therefore

† Note that ut(k) is not counted in no_ut(ch).

$$\sum_{\forall\ ch} no\_ut(ch) = \sum_{i=0}^{r} \binom{r}{i} no\_ut(x,c-i) = \sum_{i=0}^{r} \binom{r}{i} [\ (2r + 1) - 2\ (c - k/2 - i)\ ]$$

and similarly

$$\sum_{\forall\ ch} no\_lt(ch) = \sum_{i=0}^{r} \binom{r}{i} [\ (2r + 1) - 2\ (c - k/2 - i)\ ].$$

But $\sum_{\forall\ ch} no\_ut(ch) = \sum_{\forall\ ch} no\_lt(ch)$ since $\sum_{i=0}^{k-1} |\ ut(i)\ | = \sum_{i=0}^{k-1} |\ lt(i)\ |$. Therefore,

$$\sum_{i=0}^{r} \binom{r}{i} (c - k/2 - i) = 0 \qquad (2\text{-}9)$$

The only solution to Equation (2-9) is $c = (k+r)/2$. However, $c \neq (k+r)/2$ since k is even, r is odd, and c is integer. Therefore, when r is odd, we get a contradiction of the assumption that 2^r check symbols are sufficient to construct a constant-weight code of 2^r information bits. So, in this case, at least $2^r + 1 = k + 1$ check symbols are needed to construct any constant-weight code.

□

Chapter 3

Error-Correcting Balanced Codes

Balanced codes have an equal number of 1's and 0's in each codeword. Balanced codes with error correcting capability are highly desirable for optical communication and recording of data on magnetic and optical disks [Den 88, Etz 90, Fer 84, Lei 84, Tak 76, Til 89, Wid 83]. The t-error correcting balanced (t-EC/B) codes can be also viewed as t-EC/AUED (t-error correcting/All unidirectional error detecting) codes. The t-EC/AUED codes are studied in detail in Chapter 6.

This chapter presents t-error correcting balanced codes with efficient encoding and decoding algorithms, for $1 \leq t \leq 4$. These codes, in many cases, have information rates better than that of the equivalent codes given in the literature and, at the same time, have easier encoding and decoding algorithms. The new codes are designed based on the method described in Chapter 2 and on some algebraic structures, namely groups and fields. As in the case for the balanced codes (designed in Chapter 2) the new t-EC/B codes will also have easy encoding and decoding algorithms.

Two construction methods are given here , Construction I gives lower information rate but has slightly simpler encoding/decoding algorithms. In both constructions the first few bits of the information symbol are complemented and then an appropriate check symbol is appended to get the final codeword. In many cases the information symbol is not complemented at all, so these codes may be called "partially separable codes". In these codes the check bits are separated from the information bits; and the information bits can be

easily extracted from the information part by complementing 0 or more bits. In a *separable* (or *systematic*) code the information symbols are not modified at all.

In spite of partial separability and ease of encoding and decoding of the proposed codes, they give, in many cases, higher information rate than the equivalent non-systematic codes given in [Bos 82b, Kun 88, Kun 90, Sai 88]. In some cases the information rates of the new codes matches the ones in the literature; however the new codes are still considered to be better since they have easier and faster encoding and decoding algorithms.

As mentioned earlier, the terms t-EC/B and DC(n,k,2t+2) will be used interchangably. A summary of the parameters of some new DC(n,k,2t+2) codes (compared to ones given in the literature) are given in the following tables.

n	Maximum number of information bits (k)		
	Construction I	Construction II	[Bos 82b]
15	7	8	7
22	13	14	13
29	18	20	19
40	28	30	30
59	46	48	47
81	66	69	68
116	101	103	103
165	148	151	151
234	217	219	219

Table 3.1 Parameters of some DC(n,k,4) (i.e. 1-EC/B) codes.

n	Maximum number of information bits (k)		
	Construction I	Construction II	[Kun 88]
21	6	8	6
29	13	14	14
36	18	20	19
48	28	30	29
66	45	47	46
90	66	69	68
121	97	99	99
175	148	151	151
244	217	219	219

Table 3.2 Parameters of some DC(n,k,6) (i.e. 2-EC/B) codes.

n	Maximum number of information bits (k)		
	Construction I	Construction II	[Kun 90, Sai 88]
27	6	8	6
39	14	16	15
45	20	21	21
70	41	43	42
124	89	91	90
147	111	113	113
236	196	199	198

Table 3.3 Parameters of some DC(n,k,8) (i.e. 3-EC/B) codes.

n	Maximum number of information bits (k)		
	Construction I	Construction II	[Sai 88]
39	6	8	6
45	14	16	15
53	20	21	21
86	41	43	42
128	72	75	74
144	89	91	90
258	194	197	196

Table 3.4 Parameters of some DC(n,k,10) (i.e. 4-EC/B) codes.

3.1 Previous Work and Construction I

Single and double-error correcting balanced codes (1-EC/B and 2-EC/B) have been designed in [Bos 82b] and [Kun 88], respectively. Designs of 3 and 4-error correcting balanced codes (3 and 4-EC/B) were given independently by [Sai 88] and [Kun 90], and [Sai 88], respectively. All these codes assume that the information symbols are already balanced and hence they are non-systematic. Encoding and decoding of these codes may involve huge table ups. Others have designed t-EC/B codes with constraints on the run lengths (the maximum number of consecutive 0's or 1's) and the accumulated charge (the difference between the number of 1's and 0's) [Etz 90, Til 89]. These codes are also non-systematic and, in general, have no easy encoding and/or decoding procedures.

Before describing the construction methods given in [Bos 82b, Kun 88, Kun 90, Sai 88], some notations and definitions are given; also construction I is given at the end of this section.

Recall that k is the number of information bits, $U \in \{0,1\}^k$ is an information symbol, r is the number of check bits, and $ch \in \{0,1\}^r$ is a check symbol. Also recall that $X^{(j)}$ is a binary vector X with its first j bits complemented.

Notations and Definitions:

t-EC/B : t-Error Correcting Balanced Code, i.e. a balanced code with distance 2t+2.

DC(n,k,2t+2) : a t-EC/B code of length n and 2^k codewords .

G : an Abelian (i.e. commutative) group of N elements.

GF(q) : a Galois† Field of size q where $q = p^m$ for prime p.

s : the smallest integer such that $\binom{s}{s/2} \geq q$.

---

† It is interesting to note that Galois, a French mathematician, died in a sword quarrel in 1872 when he was only 23 years old.

U : be the set of all $\lfloor s/2 \rfloor$-out-of-s (or $\lceil s/2 \rceil$-out-of-s) vectors.

ψ : is any 1-1 function defined as $\psi : GF(q) \rightarrow U$, i.e. if $\psi(a) = \psi(b)$ then $a = b$.

Let X be a binary vector of length n' represented as $X = x_1x_2...x_{n'}$ and F be a field of size $q > n'$; then define

$$f_1(X) = \sum_{x_i=1} a_i, \qquad \text{for } 1 \le i \le n'$$

$$f_2(X) = \prod_{x_i=1} a_i \qquad \text{for } 1 \le i \le n'$$

$$f_3(X) = \sum_{x_i=1} \frac{1}{a_i} \qquad \text{for } 1 \le i \le n'$$

The summation in $f_1(X)$ and the product in $f_2(X)$ and $f_3(X)$ are performed over the field F. We choose $a_q = 0$ so that the 0 element will not appear in the product of $f_2(X)$ and $f_3(X)$. For convenience, define

$$F_i(X) = \psi(f_i(X)) \text{ for } 1 \le i \le 3.$$

So $F_i(X)$ will be a binary vector of length s and weight s/2.

The codes given in [Bos 82b, Kun 88, Kun 90, Sai 88] can now be described as follows:

Theorem 3.1 Let C' be an n'/2-out-of-n' code, then for $1 \le t \le 3$,
$C = \{ X' F_1(X') ... F_t(X') \mid X' \in C' \}$ is a t-EC/B code of length $n' + t\,s$ and size $|C'|$. □

When t = 1, any group G can be used instead of a field since $f_1(X)$ involves only addition [Bos 82b]. On the other extreme, when t = 3 a field of characteristic 2, i.e. $GF(2^m)$, must be used.

As mentioned earlier these codes are non-systematic since a constant-weight code C' is assumed and the information part (X') contains information as well as check bits.

Example 3.1 The design of 1, 2, and 3-EC/B codes with 6 information bits is as follows: The 2^6 information symbols can be represented by an n'/2-out-of-n' code where

n' = 8 which has 70 balanced words.

Now to get 1-EC/B code we choose the smallest group of order N where N > n'. Let $G = Z_8$ and let s = 5 since a 2-out-of-5 code has more than 8 codewords. So the 1-EC/B code will have length 8 + 5 = 13. For the 2-EC/B codes, the smallest field with q elements where q > n' = 8 is $GF(3^2)$ of 9 elements. These 9 elements can be represented by a 2-out-of-5 code, i.e. s = 5. The code length will be 8 + 5 + 5 = 18. For the 3-EC/B codes, the smallest $GF(2^m)$ field with $2^m > n' = 8$ is $GF(2^4)$ of 16 elements. These 16 elements can be represented by a 3-out-of-6 code, i.e. s = 6, so the total code length will be 8 + 6 + 6 + 6 = 20.

The t-EC/B Codes -- Construction I

Recall that in Chapter 2 we designed DC(k+r,k) codes with $k = 2^{r+1} - 0.8\sqrt{r} - 2$. Here in Construction I, a given information symbol of length k is first converted to a balanced code using the method described in Chapter 2. Then t check symbols are appended, as done earlier, to get a t-EC/B code. The encoding of construction I can be depicted in the following diagram:

$$\boxed{x_1 \,..\, x_k} \rightarrow \boxed{\bar{x}_1 \,..\, \bar{x}_j\, x_{j+1} \,..\, x_k}\ \boxed{y_1 \,..\, y_r} \rightarrow$$

$$\boxed{\bar{x}_1 \,..\, \bar{x}_j\, x_{j+1} \,..\, x_k}\ \boxed{y_1 \,..\, y_r}\ \boxed{ch_1 \,...\, ch_s}$$

information symbol — balanced word — balanced code word

More formally, let k be the number of information bits in the desired t-EC/B code. Design a balanced code C' of length k + r such that $k \le 2^{r+1} - 0.8\sqrt{r} - 2$, i.e. $r \approx (\log k) - 1$, and let

$$C = \{ X' F_1(X') \,...\, F_t(X') \mid X' \in C' \}$$

Now C is t-EC/B code of length k + r + ts with k information bits. In many cases, the t-EC/B codes designed using this method have slightly lower information rates than those given in [Bos 82b, Kun 88, Kun 90, Sai 88]. However, these codes are partially separable and have a much easier encoding and decoding algorithms.

Example 3.2 Suppose that a 1-EC/B code of 7 information bits is desired to be constructed. With 3 check bits we can design a DC(7+3,7) (balanced) code since $7 \leq 2^{3+1} - 0.8\sqrt{3} - 2$. We choose the group $G = Z_{10}$ (the additive cyclic group of size 10). The elements of this group can be represented by a 2-out-of-5 balanced code. So with extra 5 check bits, a 1-EC/B code of length 15 is obtained as shown in the first entry in Table 3.1.

□

3.2 The 1-EC/B codes - Construction II

Recall that the 1-EC/B codes given in [Bos 82b] or in Section 3.2 involve two steps: converting the information symbol to a balanced word and appending a constant-weight check symbol. In the new 1-EC/B codes (Construction II) these two steps are combined, yielding a higher information rate. The encoding in Construction II is as follows:

$$\boxed{x_1\ x_2\ \ldots\ldots\ x_k} \rightarrow \boxed{\overline{x}_1\ \overline{x}_2\ \ldots\ \overline{x}_j\ x_{j+1}\ \ldots\ x_k}\ \boxed{y_1\ y_2\ \ldots\ y_r}$$

information symbol balanced codeword

These 1-EC/B codes will be used in the next sections as inner codes for t-EC/B codes, for $2 \leq t \leq 4$. The design of 2-EC/B and 3-EC/B codes require 1-EC/B codes that are designed based on a *Field* and the 4-EC/B codes require a 1-EC/B inner code based on a *binary* Field. However, an arbitrary 1-EC/B code can be designed based on any Abelian *group*.

Notice that since the 1-EC/B codes are partially separable (in the sense that only some

bits of the information symbol are complemented) the new 2, 3, and 4-EC/B codes will also be partially separable since they contain this 1-EC/B as an inner code.

3.2.1 Arbitrary 1-EC/B Codes (over Abelian Groups)

Suppose we wish to design a 1-EC/B code of k information bits and r check bits. Let $G = \{g_1, g_2, ..., g_N\}$ be a group of N elements where $N \geq k + r$. Let $H \subset G$ of size r containing $g_{k+1}, ..., g_{k+r}$, i.e. $H = \{g_{k+1}, g_{k+2}, ..., g_{k+r}\}$. Let 0 denote the additive identity and -g denote the additive inverse of g. Let X = U ch where, in bit representation, $U = u_1u_2...u_k$, $ch = c_1c_2...c_r$, and $X = x_1x_2...x_n$, i.e. $x_i = u_i$ for $1 \leq i \leq k$ and $x_i = c_{i-k}$ for $k < i \leq n$. Now, define the following functions:

$$f'(U) = \sum_{u_i=1} g_i \tag{3-1}$$

$$f''(ch) = \sum_{c_i=1} g_{k+i} \tag{3-2}$$

Recall that $f_1(X) = \sum_{x_i=1} g_i$ for $1 \leq i \leq n$, and so $f_1(X) = f_1(U\ ch) = f'(U) + f''(ch)$.

Note that f' and f'' depend on the ordering of the elements in the group G. In particular, f''(ch) depends on the subset H of G. The function f''(ch), and hence H, will have a central importance in the design and we refer to G and the subset H as the pair [G,H].

Theorem 3.2 [Bos 82b]: If C is a w-out-of-n code and $f_1(X) = f_1(Y)$ for every X, Y $\in$ C, then C has a minimum Hamming distance of 4.

□

The 1-EC/B code designed here will satisfy $f_1(X) = 0$ for any X $\in$ C and will therefore be 1-EC/B code by the above theorem.

Definition: Given k, r, and [G,H] where $|G| = N \geq k + r$ and $|H| = r$, we say a set $CH = \{ch_1, ch_2, ..., ch_N\}$ of check symbols is a "compound check symbol" if $W(ch_i) = W(ch_j)$ and $f''(ch_i) \neq f''(ch_j)$, for $1 \leq i < j \leq N$. We also define the Hamming weight of a

compound check symbol CH to be $W(CH) = W(ch_1)$ since all the elements in CH have the same weight. Notice that for any $g \in G$ there exists a $ch \in CH$ with $f''(ch) = g$.

This CH is used to balance one (or two) weight(s) as done in the serial codes in Chapter 2; moreover, the balanced codeword X will also be constructed to satisfy $f_1(X) = 0$ as follows. After the information symbol U is converted into the desired weight by complementing j bits (i.e. getting $U^{(j)}$), we choose $ch \in CH$ with $f''(ch) = -f'(U^{(j)})$. The codeword $X = U^{(j)}\, ch$ will be balanced and will satisfy $f_1(X) = 0$ since $f_1(U^{(j)}\, ch) = f'(U^{(j)}) + f''(ch) = f'(U^{(j)}) + (-f'(U^{(j)})) = 0$. In decoding, when the weight of some codeword is off by one from the middle the erroneous bit can be easily corrected using the property that $f_1(X) = 0$.

Example 3.3 To construct a 1-EC/B code with $k = 4$ and $r = 6$, let $G = Z_{10}$ and $H = \{0, 1, 2, 3, 4, 7\}$. So the weights of the 4 information bits will be 5, 6, 8, and 9; and the weights of the check bits will be 0,1, 2, 3, 4, and 7. The following 4 compound check points can be constructed.

000101	011001	100101	011110
000011	010101	100011	110101
101000	001101	010011	101101
100100	001011	111000	011101
010100	000111	110100	011011
001100	101100	110010	010111
001010	011100	101010	001111
000110	011010	100110	111010
010001	010110	110001	110110
001001	001110	101001	101110
CH_1	CH_2	CH_3	CH_4

Notice that, in any $CH_s = \{ch_1^s,\ldots,ch_{10}^s\}$, $1 \le s \le 4$, we have $W(ch_i^s) = W(ch_j^s)$ and $f''(ch_i^s) = i-1$, for $1 \le i, j \le 10$. For instance, $ch_7^1 \in CH_1$ has weight 2 and $f''(ch_7^1) = f''(001010) = 2 + 4 = i - 1$. Also, for any $g \in G$ there exists $ch_i^s \in CH_s$ with $f''(ch_i^s) = g$.

These compound check symbols can be used to balance all weights between 0 and 4 because the checks CH_1, CH_2, CH_3, and CH_4 will balance information symbols of weights 3, 2, (0 and 4), and 1, respectively. The information words of weight 1, 2, and 3 will not be complemented at all and words of weight 0 and 4 will be complemented until they become weight 2. The encoding of all the 16 information symbols is illustrated in the Figure 3.1.

info. symbol (U)	i (in CH_i)	complemented info. symbol ($U^{(j)}$)	$f(U^{(j)})$	ch ∈ CH_i with $f''(ch) = -f(U^{(j)})$	final codeword
		(5 6 8 9)		(0 1 2 3 4 7)	
0 0 0 0	3	1 1 0 0	1	1 0 1 0 0 1	1 1 0 0 1 0 1 0 0 1
0 0 0 1	4	0 0 0 1	9	1 1 0 1 0 1	0 0 0 1 1 1 0 1 0 1
0 0 1 0	4	0 0 1 0	8	1 0 1 1 0 1	0 0 1 0 1 0 1 1 0 1
0 0 1 1	2	0 0 1 1	7	0 0 1 0 1 1	0 0 1 1 0 0 1 0 1 1
0 1 0 0	4	0 1 0 0	6	0 1 1 0 1 1	0 1 0 0 0 1 1 0 1 1
0 1 0 1	2	0 1 0 1	5	1 0 1 1 0 0	0 1 0 1 1 0 1 1 0 0
0 1 1 0	2	0 1 1 0	4	0 1 1 1 0 0	0 1 1 0 0 1 1 1 0 0
0 1 1 1	1	0 1 1 1	3	0 0 0 1 1 0	0 1 1 1 0 0 0 1 1 0
1 0 0 0	4	1 0 0 0	5	0 1 0 1 1 1	1 0 0 0 0 1 0 1 1 1
1 0 0 1	2	1 0 0 1	4	0 1 1 1 0 0	1 0 0 1 0 1 1 1 0 0
1 0 1 0	2	1 0 1 0	3	0 1 1 0 1 0	1 0 1 0 0 1 1 0 1 0
1 0 1 1	1	1 0 1 1	2	0 1 0 0 0 1	1 0 1 1 0 1 0 0 0 1
1 1 0 0	2	1 1 0 0	1	0 0 1 1 1 0	1 1 0 0 0 0 1 1 1 0
1 1 0 1	1	1 1 0 1	0	0 0 0 1 0 1	1 1 0 1 0 0 0 1 0 1
1 1 1 0	1	1 1 1 0	9	0 0 0 0 1 1	1 1 1 0 0 0 0 0 1 1
1 1 1 1	3	0 0 1 1	7	1 1 1 0 0 0	0 0 1 1 1 1 1 0 0 0

Figure 3.1 Encoding of a DC(10,4,4) code, i.e. a 1-EC/B code with k = 4 and r = 6.

Let CHS = {CH_1, CH_2, ...} be the set of all compound check symbols. In Example 3.3, CHS has 4 compound check symbols one of weight 2, two of weight 3, and one of weight 4.

The construction of CHS will be explained after describing the encoding and decoding procedures. In the following encoding procedure, it is assumed that CHS has enough compound check symbols to balance all weights between 0 and k and, at the same time, the

compound checks with the weight(s) they balance satisfy Theorem 2.5.

Encoding Procedure for 1-EC/B codes:

Let U be an information symbol and suppose that CH is used to balance words of weight W(U).

1- Complement the first j bits of U until $W(U^{(j)}) = \lceil \frac{k+r}{2} \rceil$ - W(CH).

2- Let ch $\in$ CH with f''(ch) = -f'($U^{(j)}$).

3- The encoded information symbol will be $X = U^{(j)}$ ch.

For any information symbol U we have $W(U^{(j)} \text{ ch}) = \lceil \frac{k+r}{2} \rceil$ and $f_1(U^{(j)} \text{ ch}) = 0$; so it follows from Theorem 3.2 that C is a 1-EC/B code. The decoding procedure is also straight forward.

Decoding Procedure for 1-EC/B codes:

Let X = U ch be the received codeword and ch $\in$ CH. Suppose that CH is used to balance weight(s) i (and j); now consider the following cases:

1- $W(X) = \lceil n/2 \rceil$: if $f_1(X) = 0$ then assume no errors and the information symbol will be $U^{(s)}$ where s is the least integer such that $W(U^{(s)}) = i$ (or j). If $f_1(X) \neq 0$ then detect multiple errors.

2- $W(X) = \lceil n/2 \rceil + 1$: correct a single 0→1 error at the i-th bit where $g_i = f_1(X)$.

3- $W(X) = \lceil n/2 \rceil - 1$: correct a single 1→0 error at the i-th bit where $g_i = -f_1(X)$.

4- $W(X) < \lceil n/2 \rceil - 1$ (or $W(X) > \lceil n/2 \rceil + 1$) : detect multiple errors.

We now turn our attention to the CHS set. Given k, r, and [G,H] where $|G| = N \geq k + r$ and $|H| = r$, we would like to maximize the size of CHS (i.e. the number of compound

check symbols). Let CHS_w be the set of compound check symbols of weight w and let

$$V_w^g = \{ ch \mid W(ch) = w \text{ and } f''(ch) = g \}$$

The set $\{ch_1, ch_2, ..., ch_N\}$ where $ch_i \in V_w^{gi}$ forms a compound check symbol of weight w. Every compound check symbol in CHS_w must have N unique elements from each set V_w^g for $g \in G$.

Thus, $| CHS_w | \leq \min | V_w^g |$ (over $g \in G$).

But $\sum_{g \in G} | V_w^g | = \binom{r}{w}$ and $V_w^g \cap V_w^h = \phi$ for $g \neq h$, so there exists at least one V_w^g such that $| V_w^g | \leq \lfloor \binom{r}{w} / N \rfloor$. Thus, $|CHS_w | \leq \lfloor \binom{r}{w} / N \rfloor$, and so $| CHS | \leq \sum_{w=0}^{r} \lfloor \binom{r}{w} / N \rfloor$. The maximum occurs when

$$| CHS | = \sum_{w=0}^{r} \lfloor \binom{r}{w} / N \rfloor \tag{3-3}$$

Any compound check symbol can balance at most 2 weights, thus

$k \leq 2 \sum_{w=0}^{r} \lfloor \binom{r}{w} / N \rfloor$. But $N \geq k + r$ and $\lfloor i \rfloor \approx i - 0.5$,

$k \leq 2 \sum_{w=0}^{r} [\binom{r}{w} / (k+r) - 0.5] = 2 [\frac{2^r}{k+r} - 0.5(r+1)]$, so

$k + r + 1 \leq \frac{2^{r+1}}{k+r}$, or $(k+r)^2 \leq 2^{r+1}$ from which the following two relations can be obtained.

$$r \geq 2 \log n - 1 \tag{3-4}$$

$$k \leq 2^{(r+1)/2} - r \tag{3-5}$$

Relation (3-5) gives an approximation of the maximum number of information bits that can be obtained using r check bits in this method. The true maximum (k^*), in which all compound check symbols satisfy Theorem 2.5, was found using a computer program.

The values of the true maximum k^* and the approximation of the maximum given in (3-5) are listed in Table 3.5 for some values of r. As seen from the table, $2^{(r+1)/2} - r$ is a close approximation of k^*.

r	obtained values (k)	true max k^*	approx. max $2^{(r+1)/2} - r$
6	4*	4	5
7	8*	8	9
8	14*	14	14
9	20	22	23
10	30	33	35
11	48	51	53
12	69	76	78
13	103	115	115
14	151	166	167
15	219	239	241
16	316	348	346

Table 3.5 The number of information bits (k) in the new 1-EC/B codes, the true maximum (k^*), and the approximation of the maximum.

Given r check bits, the maximum number of information bits (k^*) can be constructed only when condition (3-3), which depends on the choice of [G,H], is satisfied. Given a group G, the choice of H is crucial for condition (3-3) since it determines how the $\{0,1\}^r$ check symbols distribute among the V_w^g sets (for $0 \le w \le r$ and $g \in G$). Several groups were studied and the Z_n group, with an appropriate subset H, was found to give the maximum number of compound check symbols. Table 3.5 lists the obtained values of k in the new 1-EC/B codes. The pairs [G,H] used to get these values of k are given in Table 3.6. These pairs were obtained by trial-and-error and have no general rule.

For r = 6, 7, and 8, optimal ($k = k^*$) 1-EC/B codes with k = 4, 8, and 14 were found, as seen in Table 3.5. For $r \ge 9$, k is slightly smaller than k^*. For instance, $k^* = 22$ for r = 9 but only k = 20 was obtained using $[G,H] = [Z_{29}, \{1,2,3,4,9,13,14,17,19\}]$.

The codes may be further improved by a suitable selection of some other [G,H] combinations. It is conjectured here that, for most values of r, there exist a pair [G,H] that indeed achieve k^*.

r	k	[G,H]
6	4	[Z_{10}, {1,2,3,4,5,8}]
7	8	[Z_{15}, {1,2,3,4,5,6,11}]
8	14	[Z_{22}, {1,2,3,4,5,9,14,19}]
9	20	[Z_{29}, {1,2,3,4,9,13,14,17,19}]
10	30	[Z_{41}, {1,2,4,8,9,14,15,17,26,35}]
11	48	[Z_{59}, {1,2,3,5,17,32,33,40,47,52,58}]
12	69	[Z_{83}, {1,2,3,5,8,14,25,35,45,50,60,68}]
13	103	[Z_{116}, {1,2,3,5,8,14,25,35,45,49,64,73,101}]
14	151	[Z_{166}, {1,2,3,5,8,14,25,36,45,55,85,108,123,159}]
15	219	[Z_{235}, {1,2,3,5,8,14,25,35,45,53,54,69,85,132,168}]
16	316	[Z_{332}, {1,2,3,5,8,14,25,35,45,60,85,114,162,184,200,249}]

Table 3.6 List of [G,H] pairs used to find the value of k in Table 3.5.

It is worth noting that in any (systematic or non-systematic) 1-EC/B code C of length n = k + r, the condition $|C| \leq \frac{2}{n}\binom{n}{n/2 - 1}$ must hold [Pet 72], or

$$r \geq \frac{3}{2}\log n + 0.83 \tag{3-6}$$

or, $$k \leq n - \frac{3}{2}\log n - 0.83 \tag{3-7}$$

There may not be any (systematic or non-systematic) 1-EC/B code with information rate satisfying (3-7) with equality. The best known non-systematic 1-EC/B codes can be found in [Bro 90]. In Table 3.5, r and k satisfy $r \leq 2 \log (k+r)$ which compares favorably with the bound $r \geq 2 \log (k+r) - 1$, obtained in (3-4), and the non-systematic bound $r \geq \frac{3}{2}\log n + 0.83$, given in (3-6).

3.2.2 The 1-EC/B codes over GF(q), Z_{q-1}, and Z_{2^m-1}

As mentioned earlier, the t-EC/B codes, for $2 \leq t \leq 4$, use the 1-EC/B code as the inner code. In addition, the 2-EC/B code design require that the 1-EC/B code be designed based on some field or cyclic group Z_{q-1} where q is a power of prime. Similarly, 3 and 4-EC/B codes require that the 1-EC/B code be designed based on a binary field $GF(2^m)$ or a cyclic group Z_{2^m-1}. It was found that using Z_{2^m-1} is always better than $GF(2^m)$; so only Z_{2^m-1} is considered for the construction of the 3 and 4-EC/B codes.

To distinguish between the different 1-EC/B codes, we refer to them as a GF(q) 1-EC/B code, Z_{q-1} 1-EC/B code, etc. The different 1-EC/B codes constructed with the different algebraic structures are listed in Table 3.7. As shown in the table, some values obtained are similar to those obtained by the groups. As was the case with the groups, these values may be further improved by a better choice of group/subset or field/subset combinations.

r	arbitrary group	GF(q) or Z_{q-1}		Z_{2^m-1}	
6	4	4	Z_{10}	1	Z_7
7	8	8	Z_{15}	8	Z_{15}
8	14	14	Z_{22}	8	Z_{15}
9	20	20	Z_{30}	16	Z_{31}
10	30	30	GF(41)	21	Z_{31}
11	48	47	GF(59)	43	Z_{63}
12	69	69	GF(83)	51	Z_{63}
13	103	99	GF(113)	91	Z_{127}
14	151	151	Z_{166}	113	Z_{127}
15	219	219	GF(239)	199	Z_{255}
16	316	315	GF(331)	239	Z_{255}

Table 3.7 The maximum number of information bits in the 1-EC/B codes based on groups, GF(q) or Z_{q-1}, and Z_{2^m-1}, when r check bits are used.

3.3 The 2-EC/B codes

Recall that the 2-EC/B codes given in [Kun 88] are constructed by appending two check symbols of length s and weight s/2 to every information vector. The information symbols are assumed to be of constant-weight form. Here a method similar to that given in [Kun 88] is used but the information vectors are first converted to a 1-EC/B code and then a single constant-weight check symbol is appended to obtain the 2-EC/B code. This will also yield a partially separable code since the inner 1-EC/B code is partially separable. The choice of this extra check will depend on the field (rather than the group) used to obtain the 1-EC/B code.

Let GF(q) = $\{a_1, a_2, ..., a_q\}$ and let 0 and 1 denote the additive and multiplicative identities, respectively. Let the last element a_q be 0. For any $a \in GF(q)$, let -a and 1/a denote the additive and multiplicative inverses of a, respectively. Let $H \subset GF(q)\backslash 0$ of size r containing $a_{k+1}, ..., a_{k+r}$, i.e. $H = \{a_{k+1}, a_{k+2}, ..., a_{k+r}\}$, and let $X \in \{0,1\}^{n'}$ where $n' < q$. Recall that, $f_1(X) = \sum_{x_i=1} a_i$, $f_2(X) = \prod_{x_i=1} a_i$, s is the smallest integer such that $\binom{s}{s/2} \geq q$, U is the set of all s/2-out-of-s vectors, $\psi : GF(q) \rightarrow U$ is a 1-1 function, and $F_2(X) = \psi(f_2(X))$.

Notice that, since $q > n'$ and $a_q = 0$, the 0 element is excluded from the computations in f_2. Now the 2-EC/B codes can be constructed as follows.

Encoding Procedure for 2-EC/B codes:

1- Let C' be a GF(q) 1-EC/B code, i.e. a 1-EC/B code designed based on some field, of k information bits and length n'.

2- Let $C = \{ X' F_2(X') \mid X' \in C' \}$

Theorem 3.3 The above code C is a 2-EC/B code of k information bits and length $n' + s$.

Proof: Every codeword in C has weight $\lceil (n'+s)/2 \rceil$ since U can be chosen to contain words of weight $\lceil s/2 \rceil$ (or $\lfloor s/2 \rfloor$) if n' is even (or odd). To show that C is 2EC we must show that $D(C) \geq 2t + 2 = 6$. Let $X, Y \in C$ be two codewords of the form $X = X'ch_x$ and $X = Y'ch_y$ where $X', Y' \in C'$ and $ch_x, ch_y \in U$. Because C' is a 1-EC/B code it follows that $D(X',Y') \geq 4$; so if $D(X',Y') \geq 6$ then $D(X,Y) \geq 6$. When $D(X',Y') = 4$ it suffices to show that $ch_x \neq ch_y$ and hence $D(X,Y) \geq 6$. Suppose $D(X',Y') = 4$. Without loss of generality, assume that X' and Y' differ only in the first four bits (and agree in all other bits) as shown below:

	b_1	b_2	b_3	b_4	.....
X' =	1	1	0	0	.....
Y' =	0	0	1	1	.....

where $b_1, b_2, b_3, b_4 \in GF(q)\backslash 0$. Since $f_1(X') = f_1(Y') = 0$, we have

$$b_1 + b_2 = b_3 + b_4 = \sigma_1 \tag{3-8}$$

Now if $ch_x = ch_y$, i.e. $F_2(X') = F_2(Y')$, then $f_2(X') = f_2(Y')$, so

$$b_1 b_2 = b_3 b_4 = \sigma_2 \tag{3-9}$$

But, (3-8) and (3-9) imply that $(z - b_1)(z - b_2) = (z - b_3)(z - b_4)$, so the quadratic equation

$$z^2 - \sigma_1 z + \sigma_2 = 0$$

has four distinct roots b_1, b_2, b_3, and b_4 over the GF(q). This is impossible; so if $D(X',Y') = 4$ then $ch_x \neq ch_y$ and hence $D(X,Y) \geq 6$.

□

Decoding Procedure for 2-EC/B codes:

Let Y = X ch be the received codeword where X is a word in a 1-EC/B code C' of length n' and ch is a check symbol of length s and weight s/2. Let $S_1 = f_1(X)$ and $S_2 = f_2(X)/\beta$ where $\psi(\beta) = ch$, i.e. $\beta = \psi^{-1}(ch)$. S_1 and S_2 are called syndromes. Consider the following cases:

1- $W(X) = \lceil n'/2 \rceil$ and $S_1 = 0$: Assume no errors. Let X = U ch' and suppose that ch' $\in$ CH and CH is used to balance weight(s) w_1 (and w_2). The information symbol will be $U^{(j)}$ where s is the least integer with $W(U^{(j)}) = w_1$ (or w_2).

2- $W(X) = \lceil n'/2 \rceil$ and $S_1 \neq 0$: If $S_2 = 1$ or $W(ch) \neq s/2$ then detect multiple errors; otherwise assume a single $0 \rightarrow 1$ error and a single $1 \rightarrow 0$ error in positions i and j, respectively. Then,

$$S_1 = a_i - a_j \text{ and}$$

$$S_2 = a_i / a_j$$

Therefore, $a_j = S_1 / (S_2 - 1)$ and $a_i = S_1 + S_1 / (S_2 - 1)$, so the error positions are identified.

3- $W(X) = \lceil n'/2 \rceil + 1$ (or $W(X) = \lceil n'/2 \rceil - 1$) : If $S_1 = 0$ then detect multiple errors; otherwise assume a single $0 \rightarrow 1$ error (or $1 \rightarrow 0$ error, respectively). The error can be corrected by complementing the i^{th} bit where $a_i = f_1(X)$ (or $a_i = -f_1(X)$, respectively), as done in the 1-EC/B decoding procedure.

4- $W(X) = \lceil n'/2 \rceil + 2$ (or $W(X) = \lceil n'/2 \rceil - 2$) : If $W(ch) \neq s/2$ then detect multiple errors; otherwise assume two $0 \rightarrow 1$ (or two $1 \rightarrow 0$, respectively) errors in positions i and j. Then

$$S_1 = a_i + a_j \text{ and } S_2 = a_i a_j$$

(or $S_1 = - a_i - a_j$ and $S_2 = 1 / (a_i a_j)$, respectively).

The two error positions i and j can be found because a_i and a_j are the roots of

$$z^2 - S_1 z + S_2 = 0$$
$$(\text{or } z^2 + S_1 z + \frac{1}{S_2} = 0, \text{ respectively}).$$

5- $W(X) > \lceil n'/2 \rceil + 2$ (or $W(X) < \lceil n'/2 \rceil - 2$) : detect multiple errors.

The restriction of a field 1-EC/B code to design a 2-EC/B code may decrease the information rate of the 2-EC/B codes. The field 1-EC/B code may have smaller information rate than the group 1-EC/B code since any field 1-EC/B code is also a group 1-EC/B code but the converse is not necessarily true. A slight modification of 2-EC/B code design can improve the situation. Recall that the format of 2-EC/B code is:

$$C = \{ X' F_2(X') \mid X' \in C' \} \tag{3-10}$$

where C' is a field 1-EC/B code and any $X = X'ch \in C$ satisfies $f_1(X') = 0$ and $\frac{f_2(X')}{\beta} = 1$ where $\psi(\beta) = ch$, i.e. $\beta = \psi^{-1}(ch)$.

The following theorem which is similar to Theorem 3.2, will be useful for designing a slightly improved 2-EC/B codes.

Theorem 3.4 If C is a w-out-of-n code and $f_2(X) = 1$ for every $X \in C$ then C has a minimum Hamming distance of 4.

Proof: Similar to the proof of Theorem 3.2.

□

Suppose that C' is a 1-EC/B code with $f_2(X') = 1$ for every $X' \in C'$, then the following is a 2-EC/B code.

$$C = \{ X' F_1(X') \mid X' \in C' \} \tag{3-11}$$

Now, given a 1-EC/B code based on $[Z_{q-1}, H]$, as designed in Section 3.2, we can

design an equivalent 1-EC/B code that is based on [GF(q),H'] as follows. Recall that H contains the weights of the check bits. Suppose that the k information weights are $I \subset GF(q)\backslash\{H \cup 0\}$. Let α be a primitive element of GF(q), $H' = \{\alpha^i \mid i \in H\}$, and $I' = \{\alpha^i \mid i \in I\}$. Then the [GF(q),H'] 1-EC/B code with I as the information weights is equivalent to the $[Z_{q-1},H]$ 1-EC/B code with I' as the information weights. This is true because the product, in $f_2(X)$, of the GF(q) code is merely the summation of powers of α which corresponds to the original summation, of $f_1(X)$, in the Z_{q-1} code. Notice that the 0 element of the field does not belong to H' or I'.

The 2-EC/B codes can then be designed on either a GF(q) or a Z_{q-1} 1-EC/B code. To maximize the number of information bits we choose the 1-EC/B code that achieves higher information rate. If GF(q) (or Z_{q-1}) code was used, then the codes in (3-10) (codes in (3-11), respectively) are obtained.

The encoding and decoding of the 2-EC/B codes in (3-11) are similar to that of the original 2-EC/B codes in (3-10).

3.4 The 3 and 4-EC/B codes

A 3-EC/B code can be constructed using a $GF(2^m)$ 1-EC/B code. Two extra check symbols of length s and weight s/2 are appended to this $GF(2^m)$ 1-EC/B code to get 3-EC/B code.

We only give the encoding procedure and a theorem of its correctness. The proof of the theorem and the decoding procedure are similar to [Kun 90, Sai 88] and are therefore omitted.

3-EC/B Encoding Procedure:

1- Let C' be a 1-EC/B code of k information bits and length n' constructed using

$GF(2^m)$ as given in Section 3.2.

2- Then $C = \{ X' F_2(X') F_3(X') \mid X' \in C' \}$ is a 3-EC/B code of k information bits and length n' + 2s.

A 4-EC/B code can also be constructed based on a $GF(2^m)$ 1-EC/B code with 4 extra constant-weight check symbols of length s. This can be achieved by using the following functions [Sai 88]:

$$t_1(X) = \sum_{x_i=1} a_i^3 \text{ and}$$

$$t_2(X) = \sum_{x_i=1} \frac{1}{a_i^3}$$

Then $C = \{ X' F_2(X') F_3(X') T_1(X') T_2(X') \mid X' \in C' \}$ is a 4-EC/B code, where $T_1(X) = \psi(t_1(X))$ and $T_2(X) = \psi(t_2(X))$.

Similar to the 2-EC/B codes, a Z_{2^m-1} 1-EC/B code C' can be used to construct the 3 and 4-EC/B codes. Let C" be the equivalent $GF(2^m)$ 1-EC/B code (with $f_2(X) = 1$) obtained from Z_{2^m-1} 1-EC/B code (with $f_1(X) = 0$). Then

$C = \{ X'' F_1(X'') F_3(X'') \mid X'' \in C'' \}$ is a 3-EC/B code, and

$C = \{ X'' F_1(X'') F_3(X'') T_1(X'') T_2(X'') \mid X'' \in C'' \}$ is a 4-EC/B code.

3.5 t-EC/B Codes -- Parallel Decoding

So far all the designed codes were based on the serial balanced codes; that is because they have higher information rate than the parallel ones. A parallel t-EC/B code can be designed by simply choosing a parallel balanced code (0-EC/B) and then augmenting it by t check symbols as done in Construction I and II. For relatively small values of k it is possible to design parallel codes that are as good as the serial ones. In these cases, the parallel code is preferred since it has a faster decoding algorithm.

Let k_p be an upper bound on the number of information bits in any parallel 1-EC/B code of length k_p+r. To find k_p suppose that the maximum number of compound check symbols is |CHS|. Recall that Theorem 2.12 states that in parallel decoding at least k check symbols are needed to obtain any *constant weight* code. So $k_p \leq$ |CHS|, but |CHS| is bounded in (3-3), as follows

$$|\,CHS\,| \leq \sum_{w=0}^{r} \left\lfloor \binom{r}{w} / N \right\rfloor \text{ where } N \geq k + r, \text{ so } k_p \leq \sum_{w=0}^{r} \left\lfloor \binom{r}{w} / (k+r) \right\rfloor$$

Using a similar derivation as (3-4) and (3-5), we get

$$k_p \leq 2^{r/2} - r \tag{3-12}$$

Finding parallel 1-EC/B codes satisfying (3-12) (i.e. optimal) is equivalent to finding serial codes satisfying (3-5). They both depend on the choice of the group and its subset of size r. Let k_p^* be the maximum number of information bits that can be obtained in any parallel 1-EC/B code of length k_p^*+ r. The following table lists some achievable DC(k+r, k, 4) codes. Notice that for r = 6, 7, and 9, $k = k_p^*$.

r	obtained values (k)	maximum k_p^*	[G,H] pairs of the obtained values of k
6	4*	4	[Z_{10}, {0,1,2,3,4,7}]
7	5*	5	[Z_{25}, {0,1,2,4,7,8,10}]
8	9	10	[Z_{17}, {0,1,2,3,4,7,10,14}]
9	16*	16	[Z_{25}, {0,1,2,4,7,12,13,17,22}]
10	21	25	[Z_{31}, {0,1,2,3,4,7,13,19,20,24}]
11	33	37	[Z_{44}, {0,1,2,4,7,12,13,20,24,28,38}]
12	50	55	[Z_{64}, {0,1,2,4,7,13,16,24,32,42,48,63}]
13	75	81	[Z_{89}, {0,1,2,4,7,13,19,24,34,44,54,75,83}]
14	109	118	[Z_{123}, {0,1,2,4,7,13,19,24,34,44,61,69,87,96}]
15	158	170	[Z_{173}, {0,1,2,4,7,13,20,24,39,44,74,84,115,129,144}]

Table 3.8 The number of information bits (k) in the parallel 1-EC/B codes (and their [G,H] pairs) and the maximum (k_p^*).

Chapter 4

DC-Free Coset Codes

Linear block codes can be designed to have powerful error-correcting and error-detecting capabilities and can be encoded and decoded efficiently. However, they usually do not possess desirable dc properties. A code which is free of dc, or one that has constant dc component regardless of data patterns, provides many advantages for fiber optic and magnetic or optical recording. High-gain fiber optic transmitters can be improved if based on the average signal power, especially at high speed. It is also desirable to have short run length (the maximum number of 1's (or 0's) between two neighboring 0's (or 1's), and a high transition density in every codeword, where a transition occur when a 1 (or a 0) is followed by a 0 (or a 1). The high transition density reduces the run length and the intersymbol interference, and is useful for the receiver clock synchronization and detection processes. In many applications it is also desirable to have error correcting/detecting capabilities with these constraints. High-speed and low-complexity encoders/decoders are required for all applications, especially the ones operating on high-speed channels.

DC-Free line codes, having some (or all) the properties above, have been given in [Den 88, Fer 84, Mor 83, Kaw 88, Tak 76, Wid 83, Yos 84]. This chapter presents the design of error correcting dc-free coset codes with short run length and high transition density. The coset codes are derived by partitioning linear block codes. Thus, the new codes will have encoding and decoding complexity similar to that of linear block codes.

In the following two sections it will be shown how the "transition coset code" improves

on the DnB1M code given in [Sat 88] and how the "transition dc-free coset code" improves on the dc-free coset code given in [Den 88]. In the last section these codes are compared with balanced codes and some modified balanced codes of smaller disparity is given.

4.1 Basic Principles

This section introduces the basic foundations needed to construct these codes.

Definitions: Let X, Y be two binary vectors of length n. Then define

RL(X): the run length of X which is the maximum consecutive 1's (or 0's) occurring in X.

$DIS^j(X)$: the difference between the number of 1's and the number of 0's in X up to bit j, i.e. it is the disparity at bit j. Notice that $DIS^n(X) = 2\,W(X) - n$.

NT(X): the number of transitions in the word X. In a word $X = x_1\, x_2 \ldots x_n$, there is a transition between bits j and j+1 if $x_j \oplus x_{j+1} = 1$.

Notice that $0 \le NT(X) \le n-1$.

$\underline{10}$: denotes the vector 10101.., $\underline{01}$ denotes 01010..., and

$\underline{1}$: denotes the all 1 vector.

Example 4.1 Let X = 1001 1010 and Y = 0011 1011 then $RL(X) \le 2$, $NT(X) = 5$, $DIS^3(X) = -1$, and $DIS^5(X) = 1$.

□

Lemma 4.1 Let X be a binary vector of length n with NT(X) transitions, then

a) $\lceil NT(X)/2 \rceil \le W(X) \le n - \lceil NT(X)/2 \rceil$.

b) $W(X) - n + \lfloor NT(X)/2 \rfloor \le DIS^j(X) \le W(X) - \lfloor NT(X)/2 \rfloor$ for $1 \le j \le n$;
and in particular, $2\lceil NT(X)/2 \rceil - n \le DIS^n(X) \le n - 2\lceil NT(X)/2 \rceil$.

c) $RL(X) \le n - NT(X)$.

d) $NT(X \oplus \underline{10}) = NT(X \oplus \underline{01}) = (n-1) - NT(X)$.

Proof: The proof is divided into four parts.

a) A binary symbol 1 (or 0) can produce at most two transitions as in 010 (or 101). Since X has NT(X) transitions then there are at least $\lceil NT(X)/2 \rceil$ 1's (and $\lceil NT(X)/2 \rceil$ 0's). Therefore, $\lceil NT(X)/2 \rceil \leq W(X) \leq n - \lceil NT(X)/2 \rceil$.

b) Only 1's with no preceding 0's can increase the max disparity. In X, there are W(X) 1's in which at least $\lfloor NT(X)/2 \rfloor$ are preceded by a 0 (since there are NT(X) transitions). Therefore, the max disparity is $W(X) - \lfloor NT(X)/2 \rfloor$. Similarly, only 0's with no preceding 1's can decrease the minimum disparity. There are n - W(X) 0's in X in which at least $\lfloor NT(X)/2 \rfloor$ are preceded by a 1. So the minimum disparity is $-[n - W(X) - \lfloor NT(X)/2 \rfloor]$. Hence, $W(X) - n + \lfloor NT(X)/2 \rfloor \leq DIS^j(X) \leq W(X) - \lfloor NT(X)/2 \rfloor$ for $1 \leq j \leq n$. Since $DIS^n(X) = 2W(X) - n$ and $\lceil NT(X)/2 \rceil \leq W(X) \leq n - \lceil NT(X)/2 \rceil$ (from part a) so $2\lceil NT(X)/2 \rceil - n \leq DIS^n(X) \leq n - 2\lceil NT(X)/2 \rceil$.

c) To show that $RL(X) \leq n - NT(X)$, it is enough to show that $NT(X) \leq n - RL(X)$. The latter is clear since a word X with RL(X) consecutive 1's (or 0's) contains at most n - RL(X) transitions.

d) Before showing that $NT(X \oplus \underline{10}) = NT(X \oplus \underline{01}) = (n-1) - NT(X)$, notice that if $x_j \oplus x_{j+1} = 1$ (i.e. a transition) then $\bar{x}_j \oplus x_{j+1} = 0$ (and $x_j \oplus \bar{x}_{j+1} = 0$), i.e. no transition. Conversely, if $x_j \oplus x_{j+1} = 0$ then $\bar{x}_j \oplus x_{j+1} = 1$ (and $x_j \oplus \bar{x}_{j+1} = 1$). There are n-1 positions where a transition can occur. When X is XORed with $\underline{10}$ (or $\underline{01}$) the positions with transitions will have no transitions and conversely, the positions with no transitions will have a transition. Therefore, $NT(X \oplus \underline{10}) = NT(X \oplus \underline{01}) = (n-1) - NT(X)$.

□

4.2 Transition Code (Ct)

The "code vector space partitioning" method was used in [Den 88] to construct a dc-free coset codes with error-correcting capability. Similar techniques are used here to construct coset codes with high number of transitions, short run length, and error-correcting capability.

Let C' be an (n,k+1,d) code of length n and 2^{k+1} codewords with a k+1 $\times$ n generator matrix G'. Assume that C' contains a codeword of weight $\lceil n/2 \rceil$ (or $\lfloor n/2 \rfloor$). Then G' can be transformed to G such that $\underline{10}$ (or $\underline{01}$) is the first row of G. If the half-weight word is not of the form $\underline{10}$ (or $\underline{01}$) then the columns of G' can be permuted to obtain the $\underline{10}$ (or $\underline{01}$) vector. Without loss of generality, assume that the first row of G, say V_t, is $\underline{10}$. Rename rows 2 to k+1 in G to be g_1 to g_k. Then G can be written in the form

$$G = \begin{pmatrix} \underline{10} \\ g_1 \\ g_2 \\ \cdots \\ g_k \end{pmatrix}$$

The first row of G will be used as a transition control vector. The encoding and decoding processes described below are variations of the ones given in [Den 88]. Let U be an information vector of length k. Then <0,u>G or <1,u>G will have at least $\lfloor n/2 \rfloor$ transitions†, since NT(<0,u>G) = (n-1) - NT(<1,u>G) by Lemma 1.1(d). Then U is encoded as $<a_t,u>G$ such that $NT(<a_t,u>G) \geq \lfloor n/2 \rfloor$, where a_t is the transition control bit.

Let V' be a received codeword. Decode V' with respect to G (recall that G has a minimum Hamming distance d and k+1 information bits). If V' has more than t errors then declare uncorrectable errors, otherwise let the corrected word be V. Let $<a_t,U> = < a_t, u_1,$

† We use the notation <a,u> to represent a vector with "a" as its first component and u its second.

$u_2, u_3, ..., u_k>$ be the k+1 information bits extracted from V. The decoded information vector will be U.

The transition code, C_t, selects the 2^k high transitions codewords, i.e. the ones with at least $\lfloor n/2 \rfloor$ transitions, among the 2^{k+1} possible codewords. Alternatively, the transition code can select the low transitions codewords, i.e. ones with at most $\lfloor n/2 \rfloor$ transitions. The transition code is not systematic; however, the last k bits will either be the information bits or will be the information bits with the even bits complemented.

Example 4.2 Let C' be the (8,7,2) even parity code with the generator matrix G'. Since a word of weight 4 is present in C', G' can be rewritten as follows:

$$G = \begin{pmatrix} \underline{10101010} \\ 01000001 \\ 00100001 \\ 00010001 \\ 00001001 \\ 00000101 \\ 00000011 \end{pmatrix}$$

The information vector 000000 will be encode as <1,000000>G = 10101010, and the information vector 101000 will be encoded as <0,101000>G = 01010000. Suppose that V' = 10101111 is the received word. Since W(V') is even, no errors are assumed and V' is decoded to V = 1010111 = $<a_t,U>$. Since a_t = 1 and U = 010111, the decoded information vector will be 010111 $\oplus$ $\underline{01}$ = 000010.

□

The properties of the transition codes are stated in the following corollary.

Lemma 4.2 Let C_t be a transition code of length n. If X $\in$ C_t (i.e. NT(X) $\geq \lfloor n/2 \rfloor$) then

a- $\lfloor (n+2)/4 \rfloor \leq W(X) \leq n - \lfloor (n+2)/4 \rfloor$.

b- If $W(X) \geq \lfloor n/2 \rfloor$ then $-\lfloor (n+2)/4 \rfloor \leq DIS^j(X) \leq \lceil n/2 \rceil$ for $1 \leq j \leq n$ and

If $W(X) \leq \lfloor n/2 \rfloor$ then $-\lceil n/2 \rceil \leq DIS^j(X) \leq \lfloor (n+2)/4 \rfloor$ for $1 \leq j \leq n$;

and in particular, $2\lfloor(n+2)/4\rfloor - n \le DIS^n(X) \le n - 2\lfloor(n+2)/4\rfloor$.

c- $RL(X) \le \lceil n/2 \rceil$.

Proof: Direct substitution of $NT(X) \ge \lfloor n/2 \rfloor$ into Lemma 4.1 (a, b, and c) and using the identity $\lceil \lfloor n/2 \rfloor / 2 \rceil = \lfloor (n+2)/4 \rfloor$.

□

The simplest transition code, one with no error correcting/detecting capability, of rate $\frac{n-1}{n}$ will be compared with previous codes of equivalent information rate that are mainly used for optical transmission. First we give a brief background. In [Sat 88] a "DnB1M" (Differential n Binary 1 Mark insertion) line code is suggested for high-speed optical communication systems. This code is obtained by AMI (Alternate Mark Inversion) coding for the first n-1 bits and then the complement of the last bit is appended at the end (which will always produce a transition between the last two bits). More formally, DnB1M can be defined as follows. Let $U = u_1 .. u_{n-1}$ be the n-1 information vector and $V = v_1 .. v_n$ be the n bit codeword (v_0 is an initial state) then define

- $v_0 = 0$ (or 1),
- $v_i = v_{i-1} \oplus u_i$, for $1 \le i \le n-1$
- $v_n = v_{n-1} \oplus 1$.

This code has many advantages such as short run length, a transition in every codeword, and a good balance of 1's and 0's (marks and spaces) regardless of the 1 arrival probability.

Let C* = (n,n-1,1) be a transition code obtained from C' = (n,n,1) code (with $G' = I_{n\times n}$). C* achieves all the features of the DnB1M code plus two main advantages. The transition density in each codeword increases from 1 to $\lfloor n/2 \rfloor$, and the narrow weight distribution yields smaller DC variation and a rapid balance of 1's and 0's. There are other advantages of this code in jitter suppression and power spectrum. The next table

summarizes some of the improvement of the transition code.

	Code Rate	Max run length	min # of transitions	weight range
Codes in [Sat 88]	(n-1)/n	n	1	1..(n-1)
Proposed codes	(n-1)/n	n	$\lfloor n/2 \rfloor$	n/4..3n/4

Table 4.1 Comparison of the C_t codes and the codes given in [Sat 88].

4.3 Transition DC-Free Code ($C_{t\,w}$)

Instead of using the transition control vector $\underline{10}$, one can use a weight control vector $V_w = \underline{1}$, i.e. the all 1 vector. In this case, words of weights between 0 and $\lfloor n/2 \rfloor$ or between $\lceil n/2 \rceil$ and n can be selected. This code is referred to as weight code, C_w. This code is used to stabilize the dc level in a line code in [Den 88] and to design EC/AUED codes in [Alb 89b, Bru 89].

In combining the two control vectors, the advantages of both the transition codes and weight codes can be utilized. Such a code can be designed as follows. Let C' be an (n,k+2,d) linear code with a generator matrix G'. Let C' contain the $\underline{1}$ word† and a half-weight word. Then G' can be written as

$$G = \begin{pmatrix} \underline{10} \\ \underline{1} \\ g_1 \\ g_2 \\ \cdots \\ g_k \end{pmatrix}$$

The intersection of the codes C_t and C_w can be obtained. The last k bits are used as information and the first 2 are used for control to produce the transition and weight control

† The Golay, BCH, and some shortened BCH codes have the $\underline{1}$ vector.

code, $C_{t,w}$. The transition control selects the codewords with high transition density and then the weight control adjusts the disparity according to the current disparity, as depicted in Figure 4.1.

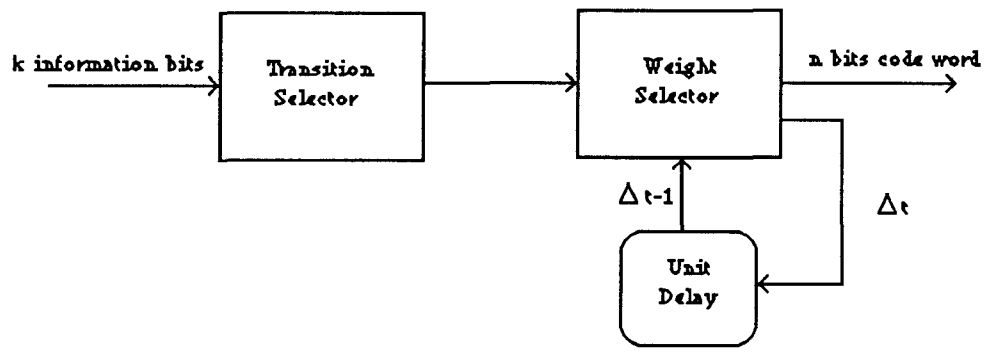


Figure 4.1 Conceptual encoding diagram of the $C_{t\,w}$ code.

Consider the following encoding algorithm:

1- Let U be the information vector to be sent at time t and Δ_t be the disparity at time t

2- Choose a_t such that $NT(<a_t,0,u>G) \geq \lfloor n/2 \rfloor$.

3- Choose a_w such that $W(<a_t,a_w,u>G) \geq \lfloor n/2 \rfloor$ if $\Delta_t \leq 0$, and $W(<a_t,a_w,u>G) \geq \lfloor n/2 \rfloor$ if $\Delta_t \geq 0$.

4- X is encoded as $<a_t,a_w,u>$ G.

In practice, a_t and a_w can be computed simultaneously from the values NT(<0,0,X>G), W(<0,0,X>G), and W(<1,0,X>G).

The decoding algorithm is as follows:

1- Let V' be the received codeword. Decode V' with respect to C' to V (recall that C has a minimum Hamming distance d). If errors are detected (and not corrected) declare the errors else let the corrected word be V.

2- Let $<a_t,a_w,u>$ be the k+2 information bits extracted from V. The decoded information vector is U.

Example 4.3 Let C' be the (8,7,2) even parity code with generator matrix G' as given in Example 4.2. Since words of weight 4 and 8 are present in C', G' can be written as follows:

$$G = \begin{pmatrix} \underline{10101010} \\ \underline{11111111} \\ 00100001 \\ 00010001 \\ 00001001 \\ 00000101 \\ 00000011 \end{pmatrix}$$

Assume that U = 100111 is to be sent at time t and the disparity is negative, i.e. $\Delta_t < 0$. First, since $<0,0,100111>G$ = 00100111 has only 3 transitions, a_t must be 1 so $NT(<1,0,100111>G) = NT(10001101) = 4$. And since $\Delta_t < 0$, the next word must have weight 4 or higher. But W(10001101) = 4 so $a_w = 0$. Hence the codeword will be $<1,0,100111>G$ = 10001101.

Let V' = 01000100 be the received codeword. Because the weight is even, no errors are assumed and V' is decoded to V = 0100010 = $<a_t,a_w,u>$. From (4-2) in the decoding algorithm, the decoded information vector will be U' = 00010 $\oplus$ 0 $\underline{10}$ $\oplus$ 1 $\underline{01}$ = 00010 $\oplus$ $\underline{01}$ = 01000.

□

The maximum disparity can be obtained as follows. By Lemma 4.2, $DIS^n(X) \le n - 2\lfloor(n+2)/4\rfloor$. Suppose that the disparity after transmitting the current word is maximum, i.e. $\Delta_t = n - 2\lfloor(n+2)/4\rfloor$. The next word, say Y, should have a negative disparity so $-\lceil n/2\rceil \le DIS^j(Y) \le \lfloor(n+2)/4\rfloor$. Therefore, the running disparity will not exceed $n - 2\lfloor(n+2)/4\rfloor + \lfloor(n+2)/4\rfloor = \lfloor(3n+1)/4\rfloor$. Similarly the minimum disparity will not be less than $-\lfloor(3n+1)/4\rfloor$. The run length of the line code will not be more than $2\lceil n/2\rceil$ since $RL(X) \le \lceil n/2\rceil$ for any

codeword in $C_{t,w}$. It is worth noting that the max (and min) disparity and max run length are only upper bounds and may never be achieved. The reason is that a specific concatenation of words achieve such bounds. These words may not be produced by $C_{t,w}$, so it is quite possible that the disparity and run length are significantly smaller than the upper bounds.

The proposed $C_{t,w}$ line code, obtained from C' = (n,k+2,d), has the following properties:

- The number of codewords is 2^k.
- The same old code distance, i.e. $D(C_{t,w}) = D(C') = d$.
- The running disparity will be between $-\lfloor(3n+1)/4\rfloor$ and $\lfloor(3n+1)/4\rfloor$.
- The max run length in the line code is not more that $2\lceil n/2\rceil$.
- There are at least $\lfloor n/2\rfloor$ transitions in every word.

The codes given in [Den 88] achieve the first three properties. However, these codes have more transitions and shorter run length as shown in the following Table.

	Max Disparity	run length	min # of transitions
Codes in [Den 88]	$\lfloor(3n+1)/4\rfloor$	$3n/2^{\dagger}$	0
Proposed Codes	$\lfloor(3n+1)/4\rfloor$	$2\lceil n/2\rceil$	$\lfloor n/2\rfloor$

Table 4.2 Comparison of the $C_{t\,w}$ codes and the codes given in [Den 88].

† In [Den 88], when n mod 4 = 0, the run length can be as much as 3n/4, e.g. when n = 8, the run length can be 12 as follows:

```
1111 1100 0000 0000 0011   ...    line output
0    4    4    0   -4    -4  ...    running disparity.
```

Extensions to the Ct,w codes

The transition and weight controlled codes can be further improved by using two or more, say p, transition control vectors and two or more, say q, weight control vectors. For example, let C' = (8,7,2), with G' as in Figure 4.2 (a). If p = 2 and q = 2 then G' can be rewritten as G in Figure 4.2 (b). In this case, when p = q = 2, the $C_{t,w}$ code will have 3 information bits, 4 control bits, and a parity bit. The number of transitions will be at least 4, 2 in the first 4 bits and 2 in the second 4 bits. In fact, the line code produced here is the same as using p = q = 1 with n' = $\lfloor n/2 \rfloor$. So the maximum disparity will be $\lfloor (3n'+1)/4 \rfloor$ and the maximum run length will be $2\lceil n'/2 \rceil$, where n' = $\lfloor n/2 \rfloor$. In general, if p = q and s = $\lceil n/(2p) \rceil$, then the disparity will be bounded by $\lfloor 3s/2 \rfloor$ and the run length is bounded by 2s. In [Den 88] the disparity is the also $\lfloor 3s/2 \rfloor$ but the run length can be up to 3s if s is even. Thus, these codes are still superior.

Figure 4.2 (c) shows the G matrix when p = 1 and q = 2. It turns out that no gain can be obtained by this combination, i.e. will have the same disparity and run length as in p = q = 1. The last example we consider is when p = 2 and q = 1 as shown in Figure 4.2 (d). This code will have at least $\lfloor n/2 \rfloor$ transitions and a maximum disparity of $\lfloor (3n+1)/4 \rfloor$, similar to the case p = q = 1. However, the run length will not exceed $2\lceil n/4 \rceil$.

$$\begin{pmatrix} 10000001 \\ 01000001 \\ 00100001 \\ 00010001 \\ 00001001 \\ 00000101 \\ 00000011 \end{pmatrix} \qquad \begin{pmatrix} \underline{10100000} \\ \underline{11110000} \\ \underline{00001010} \\ \underline{00001111} \\ 00100001 \\ 00010001 \\ 00000011 \end{pmatrix} \qquad \begin{pmatrix} \underline{10101010} \\ \underline{11110000} \\ \underline{00001111} \\ 00100001 \\ 00010001 \\ 00000101 \\ 00000011 \end{pmatrix} \qquad \begin{pmatrix} \underline{10100000} \\ \underline{00001010} \\ \underline{11111111} \\ 00100001 \\ 00010001 \\ 00000101 \\ 00000011 \end{pmatrix}$$

a) G' of (8,7,2) b) p = 2, q = 2 c) p = 1, q = 2 d) p = 2, q = 1

Figure 4.2 Different $C_{t\,w}$ dc-free coset codes.

We chose not to include a detailed analysis of these cases, however if the control

vectors exist, then the code can be easily designed. This leaves open the different ways of using such codes and suggests that the final design is dependant on the specific application and its requirement.

Chapter 5

Extensions of Balanced Codes

This chapter describes three ways of generalizing the balanced codes designed in the previous chapters. The first, and the most important, is the design of balanced codes with low dc level and high transitions. These codes are designed based on the combined techniques used Chapters 2 and 4. One extra check bit is used to construct balanced codes that have almost half the dc level of the original balanced code and have much higher transitions density. These codes are much more attractive for optical transmission than the bare-bone balanced codes. The second generalization is the design of balanced codes over non-binary alphabet. When the alphabet size is q, containing {0, 1, ..., q-1}, the balanced code will be referred to by $DC^q(n,k)$ code. When q is 2 , the ordinary balanced code is obtained. The last generalization is the design of semi-balanced codes, or DC_m codes, in which the codewords have weights between $\lceil (n-m)/2 \rceil$ and $\lceil (n+m)/2 \rceil$. When m is 0, the DC_m codes degenerates to an ordinary balanced code.

5.1 Balanced Codes with Low DC Levels

Let C be a DC(n,k,2t+t) code; i.e. a t error correcting balanced code of length n and 2^k codewords. Since every codeword X in C will have $\lceil n/2 \rceil$ 1's and $\lfloor n/2 \rfloor$ 0's in X, we get

$$
\begin{aligned}
&-\lfloor n/2 \rfloor \le DIS^j(X) \le \lceil n/2 \rceil \\
&NT(X) \ge 1 \\
&RL(X) \le \lceil n/2 \rceil
\end{aligned}
\qquad (5\text{–}1)
$$

Recall that $DIS^j(X)$ denotes the difference between the number of 1's and the number of 0's in X up to bit j, RL(X) is the run length of X which is the maximum consecutive 1's (or 0's) occurring in X, and NT(X) is the number of transitions in the word X.

The balanced codes designed in Chapters 2 and 3 satisfy the properties in (5-1) above. In this section we show how to decrease the maximum running disparity and the run length, and to increase the number of transitions in the balanced codes. The main idea is to increase the number of transitions in every balanced word as done in the transition code (C_t) in Chapter 4. This can be accomplished by using the identity $NT(X \oplus \underline{10}) = NT(X \oplus \underline{01}) = (n-1) - NT(X)$ in Lemma 4.1 (d). Recall that the design of the balanced codes involves complementation of the first j bits of the information symbol; and then an appropriate check symbol is appended to get the final codeword. Suppose that an information symbol X of length k is encoded to a balanced word Y where $NT(Y) < \lfloor k/2 \rfloor - 1$. If the modified information symbol $X' = X \oplus \underline{10}$ is encoded to a balanced code Y', then $NT(Y') \geq \lfloor k/2 \rfloor - 1$. This is stated in the following lemma.

Lemma 5.3 Let X be an information vector of length k and let $X' = X \oplus \underline{10}$. Let the balanced encoding of X (and X'), using the construction of balanced codes in Chapter 2, be Y (and Y' respectively). Then

$$\text{either } NT(Y) \geq \lfloor k/2 \rfloor - 1 \text{ or } NT(Y') \geq \lfloor k/2 \rfloor - 1$$

Proof: First notice that $NT(X^{(j)}) \geq NT(X) - 1$. This is true since all the transitions between bit 1 and j-1 and transitions between bit j+1 to k are preserved. The only transition loss (or gain) occurs between bit j and j+1. So complementing the first j bits of X can only decrease (or increase) the transitions by 1. So if $NT(X) \geq \lfloor k/2 \rfloor$, then $NT(Y) \geq \lfloor k/2 \rfloor - 1$. Suppose that $NT(X) < \lfloor k/2 \rfloor$. By Lemma 4.1 (d), $NT(X') = NT(X \oplus \underline{10}) = (k-1) - NT(X) \geq \lfloor k/2 \rfloor$; therefore, $NT(Y') \geq \lfloor k/2 \rfloor - 1$.

□

The code design now becomes clear. A DC(n,k,2t+2) can be changed to a DC(n,k-1,2t+2) having words with at least $\lfloor k/2 \rfloor - 1$ transitions. This can be accomplished by converting the information symbols of length k - 1 to a transition code C_t (of length k). Recall that all the words in the transition code have at least $\lfloor k/2 \rfloor$ transitions (see Lemma 4.2). This transition code is then balanced using the ordinary methods of Chapter 2 (or 3). As seen earlier, the complementation of few bits can only decrease the number of transitions by 1. So, the resulting code will be balanced and all its words have at least $\lfloor k/2 \rfloor - 1$ transitions. As a matter of fact, it can be shown that there will be $\lfloor k/2 \rfloor + t - 1$ transitions in every word in the new DC(n,k-1,2t+2) code. But in general, k is much larger than t, so we chose to neglect this term.

Given that all the words in this balanced code satisfy $W(X) = \lceil n/2 \rceil$ and $NT(X) \geq \lfloor k/2 \rfloor - 1$, then from Lemma 4.1 (b) and (c) the following code properties can be obtained:

$$
\begin{aligned}
&-\lceil k/4 \rceil - \lfloor r/2 \rfloor \leq DIS^j(X) \leq \lceil k/4 \rceil + \lceil r/2 \rceil \\
&NT(X) \geq \lfloor k/2 \rfloor - 1 \\
&RL(X) \leq \lceil k/4 \rceil + \lceil r/2 \rceil + 1
\end{aligned}
\qquad (5\text{–}2)
$$

The maximum disparity of this code is reduced by almost one half that of (5-1), since k is much larger than r. Also the number of transitions are increased dramatically and the maximum run length is decreased by about one half. Of course, the information rate of this balanced code is $\frac{k-1}{n}$ whereas it is $\frac{k}{n}$ for the codes in (5-1).

Balanced Codes vs. Line Codes

This section gives a brief comparison between these new balanced codes and the $C_{t,w}$ codes designed in Section 4.3. The $C_{t,w}$ codes have maximum disparity of $\lfloor (3n+1)/4 \rfloor$ with information rate of $\frac{k}{n}$. The number of information bits depends on the error correcting

capability of such codes. Here, we consider the case when t = 1 to give a flavor of the comparison.

Suppose that C_1 is a DC(n,k-1,4) code with low disparity as designed above. Recall that, using s check bits, $k \approx 2^{(s+1)/2} - s$ in the 1-EC/B codes (see 3.5). Let the disparity and information rate of C_1 be D_1 and I_1, respectively. Then

$$D_1 \leq \lceil k/4 \rceil + \lceil s/2 \rceil \approx \frac{2^{(s+1)/2}}{4} \tag{5-3}$$

$$I_1 \approx 1 - \frac{s + 1}{2^{(s+1)/2}} \tag{5-4}$$

Let the other code C_2 be a $C_{t,w}$ code that corrects one error. Recall that these codes are based on some linear codes. The *best* linear codes that correct one error are Hamming codes which have r check bits and length $2^r - 1$. So, their information rate is $1 - \frac{r}{2^r - 1}$. When 2 extra information bits are used to control the dc value the information rate will reduce slightly. Let the disparity and information rate of C_2 be D_2 and I_2, respectively. Then

$$D_2 \leq \lfloor (3n+1)/4 \rfloor \approx \frac{3}{4} 2^r \tag{5-5}$$

$$I_2 \approx 1 - \frac{r + 2}{2^r - 1} \tag{5-6}$$

The argument of comparing is as follows: Given a certain disparity level, which code have higher information rate? To get similar disparity in the two codes, i.e. equating D_1 to D_2 in (5-3) and (5-5) and taking log on both sides, we obtain

$$s = 2\,r + 2 \log 3 - 1 \tag{5-7}$$

When (5-7) is substituted in (5-4) to get the information rate we get

$I_1 = 1 - \frac{s + 1}{2^{(s+1)/2}} = 1 - \frac{2\,r + 2 \log 3}{2^{r+\log 3}} = 1 - \frac{2}{3}\left(\frac{r + \log 3}{2^r}\right)$. It can be readily seen

that this information rate is higher than I_2 in (5-6). It can be concluded that, for a certain level of disparity, these balanced codes have higher information rate than the coset dc-free coset codes.

Example 5.1 The DC(22,14,4) (in table 3.1) can be converted to a DC(22,13,4) code with codewords all having at least 6 transitions (actually 7 in this case). The maximum accumulated charge will be 8 and the information rate will be $\frac{13}{22}$. On the other hand, a single error correcting (shortened) Hamming code of length 11 and 2^7 codewords can be designed. With this (11,7) code, 2 bits are to be used to control the dc level. The dc level will not exceed $\lfloor (3n+1)/4 \rfloor = 8$. The information rate of this code is $\frac{7-2}{11}$ which is less than $\frac{13}{22}$ obtained above.

□

5.2 Balanced Codes over a Non-Binary Alphabet

It will be shown in this section how to extend the construction of balanced codes from a binary {0,1} alphabet to a q-ary alphabet {0, 1, ..., q-1}. We first define the weight of X $\in$ {0, 1, ..., q-1}^{k} to be the sum of its digits; i.e. $W(X) = \sum_{i=1}^{k} x_i$. Furthermore, assume that $\bar{x}_i = q - x_i$. The objective now is to encode every information of length k to a "balanced" word X of length n such that $W(X) = \lceil (q-1)n/2 \rceil$.

Let $DC^q(n,k)$ denote a code of length n and 2^k codewords over the alphabet {0, 1, .., q-1}, where each codeword $X \in DC^q(n,k)$ satisfies $W(X) = \lceil (q-1)n/2 \rceil$. Suppose we wish to design $DC^q(k+r,k)$ code. In this case there will be q^r check symbols. The number of check symbols of weight w, which can be found in [Cap 89], is

$$\sum_{i=0}^{\lfloor w/(q-1) \rfloor} (-1)^i \binom{n}{i} \binom{n-1+w-qi}{n-1}$$

The maximum number of balanced words (of weight n(q-1)/2) will be

$\sum_{i=0}^{\lfloor n/2 \rfloor} (-1)^i \binom{n}{i} \binom{n/2-1-qi}{n-1}$ which can be approximated by $\frac{q^n}{\sqrt{\pi n}} \frac{6}{q^2 - 1}$.

But, any balanced code should satisfy $q^k \leq \frac{q^n}{\sqrt{\pi n}} \frac{6}{q^2 - 1}$; therefore, the minimum number of check bits is

$$r \geq \frac{1}{2} \log_q k + 2 \tag{5-8}$$

The codes constructed here use methods similar to that of Chapter 2; the constructed codes have the following parameters.

1) Serial $DC^q(k+r,k)$ with $k \approx \frac{2}{(q-1)^2} q^r + \frac{q-3}{q-1} r - \frac{q-3}{(q-1)^2}$ (5-9)

2) Parallel $DC^q(k+r,k)$ with $k \approx \frac{q^r}{q-1}$ (5-10)

From (5-10) above, the following relation on r can be obtained: $r \approx \log_q k + 1$, as compared to $r \geq \frac{1}{2} \log_q k + 2$ in (5-8). The number of check digits is about twice as much as required by a non-systematic code. Recall that this was also the case for binary balanced codes. It is quite amazing that the parallel $DC^q(n,k)$ codes have higher information rate than the serial ones when $q > 3$ and almost the same when $q = 3$. This may suggests that there is a better construction of the serial codes.

Before going into the code design, we discuss the behavior of random walks in a non-binary case. A random walk of a binary vector changes by ± 1 in every bit complementation; whereas a random walk of a word in $\{0, 1, ..., q-1\}^k$ changes by at most q-1 (or -(q-1)) in each digit complementation; i.e. $W(X^{(j)}) - q < W(X^{(j+1)}) < W(X^{(j)}) + q$. For example, suppose that $q = 3$ and $k = 4$, then the word $X = 0210$ of weight 3 will have the following random walk.

$W(X^{(0)}) \rightarrow W(X^{(1)}) \rightarrow W(X^{(2)}) \rightarrow W(X^{(3)}) \rightarrow W(X^{(4)})$

$W(0210) \rightarrow W(2210) \rightarrow W(2010) \rightarrow W(2010) \rightarrow W(2012)$

$3 \rightarrow 5 \rightarrow 3 \rightarrow 3 \rightarrow 5$

The random walk function σ will satisfy the following properties:

$\sigma_0 = W(X)$

$\sigma_k = (q-1)\,k - W(X)$

$\sigma_j \in [\, \sigma_{j-1} - (q-1), \sigma_{j-1} + (q-1)\,]$ for $0 < j \leq k$

As in the binary case, the random walk starts from weight W(X) and ends at $(q-1)\,k - W(X)$. However, it may not go through all the weights in between since it can vary by q-1 at every digit complementation. In the serial case this is solved by using (q-1) check symbols of consecutive weights, say i, i+1, ..., i+(q-1). This group of check symbols now can balance information words of a certain weight, say w; or two weights w_1 and w_2. In the parallel case, the check symbols are placed in such a way that when a random walk hits the middle line $\lceil (q-1)k/2 \rceil$ it will indeed hit some check point even though it may increase (or decrease) by a rate of q-1. The serial and parallel constructions are explained in more details in the following sections.

5.2.1 Serial $DC^q(n,k)$ codes

The information words of weights between $\lceil (q-1)n/2 \rceil - (q-1)r$ and $\lceil (q-1)n/2 \rceil$ can be directly balanced by appending check symbols of weights from (q-1)r to 0, respectively. Now we have to balance weights 0 to $\lceil (q-1)n/2 \rceil - (q-1)r - 1$ and the weights $\lceil (q-1)n/2 \rceil + 1$ to (q-1)k. With the other check symbols, totaling $q^r - (q-1)r - 1$, every q-1 check symbols of consecutive weights can be used to balance two weights as in the construction given in Section 2.2. Let CH be a group of such check symbols; i.e. $CH = \{ ch_1, ch_2, ..., ch_{q-1} \}$ where $ch_i = j+i$ for some j and $1 \leq i < q$. It is relatively easily to show that CH can always

balance words of weight w, where $0 \le w \le \lceil (q-1)n/2 \rceil - (q-1)r - 1$ or $\lceil (q-1)n/2 \rceil + 1 \le w \le (q-1)k$. Moreover, CH can balance two weights, w_1 and w_2 where $| w_2 - w_1 | \ge \lceil (q-1)n/2 \rceil$.

Now the problem reduces to form as much CH's as possible from the remaining $q^r - (q-1)r - 1$ check symbols.

In the best case, these check symbols will from into $\lfloor \frac{q^r - (q-1)r - 1}{q-1} \rfloor$ CH's.

Therefore, the maximum number of different weights that can be balanced is

$$2 \lfloor \frac{q^r - (q-1)r - 1}{q-1} \rfloor + (q-1)r + 1 \le \frac{2}{q-1} q^r + (q-3)\, r - \frac{q-3}{q-1}.$$

Since the number of different weights will be $k(q-1) + 1$, we get relation (5-9) which is

$$k \le \frac{2}{(q-1)^2} q^r + \frac{q-3}{q-1} r - \frac{q-3}{(q-1)^2}$$

Notice that when $q = 2$ this construction reduces to the one given in [Bos 87]; i.e. $k = 2^r - r - 2$.

Example 5.2 When $q = 3$ and $r = 2$, from the above equation, or from (5-10), we can get $k = 4$. There will be 9 different weights from 0 to 8 of the information symbols. Since the code length is 6, the final balanced words should have weight $\lceil (q-1)n/2 \rceil = 6$. The 9 check symbols { 00, 01, 02, 10, 11, 12, 20, 21, 22} can be used to balance all possible weights (0 through 8) as follows. We use 22, 12, 02, 01, and 00 to balance weights 2, 3, 4, 5, and 6, respectively. The other 4 check symbols { 10, 11, 20, 21 } can form 2 compound checks $CH_1 = \{10, 11\}$ and $CH_2 = \{20, 21\}$ which can be used to balance weights 1 and 8, and weights 0 and 7, respectively. This is shown in the following figure.

weight	0	1	2	3	4	5	6	7	8
checks	{20,21}	{10,11}	22	12	02	01	00	20,21	10,11

Figure 5.1 Check symbols in the serial $DC^3(6,4)$ code.

5.2.2 Parallel $DC^q(n,k)$ codes

As in Section 2.3, the construction starts by partitioning all the (q^r) check symbols into m subsets D_i, for $1 \le i \le m$. The extra condition that the size of any D must be at least q-1 must be satisfied. These subsets are assigned an integer value as follows.

$$d_1 = 0$$

$$d_{j+1} = d_j + \left\lfloor \frac{\lfloor D_j/2 \rfloor + \lceil D_{j+1}/2 \rceil}{q-1} \right\rfloor \quad \text{for } 1 \le j < m$$

Now, it can be shown that if a random walk of some word reaches weight $\lceil (q-1)k/2 \rceil$ after the complementation of s digits where $d_j \le s < d_{j+1}$, then this word can be balanced using a check symbol from D_j or D_{j+1} (refer to Lemma 2-7 to 2-11).

Since $\sum_{i=1}^{m} D_i \approx q^r$, then $d_{j+1} \le q^r$. This means that at most q^r weights can be balanced which implies that the maximum number of information digits satisfies $k \le \frac{q^r-1}{q-1}$ as given in (5-10).

Example 5.3 To design a parallel $DC^3(6,4)$ code, consider the following. Since q = 3 and r = 2 we k = 4 by (5-10). As in example 5.1, there will be 9 different information weights (0 to 8) that will be converted into weight 6 codewords. The two subsets D's with their extend of complementation (d's) are given below.

$D_1 = \{ 00, 10, 02, 12, 22 \}$ $\quad d_1 = 0$

$D_2 = \{ 01, 11, 21 \}$ $\quad d_2 = 2$

If an information symbol X has weight $2 \le W(X) \le 6$, then it can be balanced with 0

(=d_1) complementation and appending an appropriate check symbol from D_1. Information vectors of weights 0, 1, 7, and 8 can always be balanced after complementing 2 (=d_2) digits and then appending an appropriate check symbol from D_2. Notice that, the check symbol 20 is not included in any D subset since it was not needed.

5.2.3 On the Number of Check Digits

In this section, we derive a lower bound on number of check digits required for the parallel $DC_m(k+r,k)$ code. This lower bound is that $r \geq \log_q k + 1$. This will indicate the the previous construction is close to optimal. In what follows we sketch the proof of the argument.

Let S_1 be the set of information vector of the form $(a_1, a_2, ..., a_k)$ where

$a_1 = a_2 = ... a_i = (q-1)$,

$0 \leq a_{i+1} < q-1$, and

$a_{i+2} = a_{i+3} = ... = a_k = 0$ where $0 \leq i \leq k$.

And let S_2 be the set of information vector of the form $(a_1, a_2, ..., a_k)$ where

$a_1 = a_2 = ... a_i = 0$,

$0 \leq a_{i+1} < q-1$, and

$a_{i+2} = a_{i+3} = ... = a_k = q-1$ where $0 \leq i \leq k$.

Notice that $| S1 | = | S2 | = (q-1) k + 1$. The only common words between S1 and S2 are the all zero and the all (q-1) words. Let $S = S1 \cup S2$; then $| S | = 2 (q-1) k$. We will show that a check symbol can balance at most 2 words from S, i.e. at most two random walks of S intersect any point in the space. This will imply that the number of check symbols must be at least $| S | / 2 = (q-1) k$; or $q^r \geq (q-1) k$.

To show this, partition S into $S_0, ..., S_{q-1}$ where if $X \in S_i$ then X has weight = i mod

q. It can then be shown that if X and Y are in different partitions then $W(X^{(j)}) \neq W(Y^{(j)})$. Moreover, it can be shown that at most two words in the same subset, say X and Y, will satisfy $W(X^{(j)}) = W(Y^{(j)})$ for any j. Therefore, $q^r \geq (q-1)\, k$; so $k \leq q^r/(q-1)$, or $r \geq \log_q k + \log_q (q-1)$ which implies that the parallel construction is almost optimal.

5.3 $DC_m(n,k)$ Codes

Let $DC_m(n,k)$ denote a code of length n with 2^k codewords and each codeword satisfy $\lceil (n-m)/2 \rceil \leq W(X) \leq \lceil (n+m)/2 \rceil$. A $DC_0(n,k)$ code is the ordinary balanced code. The $DC_m(n,k)$ codes have the property that the weight of a codeword can be any of the m+1 weights around the center (n/2). Equivalently, every codeword has a maximum accumulated charge (or a disparity) of no more than m. In this section, the construction methods of serial and parallel $DC_0(n,k)$ codes in Chapter 2 are generalized to construct the $DC_m(n,k)$ codes. With these methods, we construct the following:

- Serial $DC_m(k+r,k)$ codes with $k \approx (m+1)\, 2^{r+1} - \frac{m}{2} - \frac{0.8\sqrt{r}}{m+1} - 2$,

- Parallel $DC_m(k+r,k)$ code with $k = (m+1)\, 2^r$ (or $k = (m+1)\, 2^r - 1$ if r+m is odd).

These codes can be shown to be optimal when we use the method given in [Knu 86].

5.3.1 Serial $DC_m(n,k)$ Codes

The code construction will be similar to that of Section 2.2.3. Also, the notation of Section 2.2.3 will be followed closely here. Before giving the code design, we first go through one example.

Example 5.4 Suppose that k = 20, r = 2 and m = 2 forming a $DC_2(22,20)$ code, i.e.

a (10, 11, or 12) out-of-22 code. The check 00, for instance, can be appended to weights 10, 11, or 12. Also the check 00 can be used to balance† two different weights, say a and b. Again, these weights can be balanced to weights 10, 11, or 12. When they are balanced to weight 10, we must have b-a > max(20-10,10) = 10. Similarly, when they are balanced to weights 11 or 12 we must have b-a > (20-11,11) = 11 and b-a > (20-12,12), respectively. So the check symbol 00 participates in 3 different weights and acts as if it is 3 different check symbols on levels, say 0, 1, and 2. Similarly, the checks 01 and 10 will be on levels 1, 2, and 3, and 11 will be on levels 2, 3, and 4, as shown in Figure 5.2. □

Level	Weight	check symbols	g_i
0	12	00	1
1	11	00, 01, 10	3
2	10	00, 01, 10, 11	4
3	9	01, 10, 11	3
4	8	11	1

Figure 5.2 Check symbols in the serial $DC_2(22,20)$ code.

When the check symbol and the weight of the codeword are used to determine the original weight(s) of the information word, one can utilize $(m+1)\,2^r$ check symbols. If d check symbols are used for single maps and the rest, i.e. $(m+1)\,2^r - d$, are used for double maps, then $k = (m+1)\,2^{r+1} - d - 1$.

Define a vector **g** to have the number of check symbols on the different levels. This vector can be computed as follows:

for $0 \le i \le r+m$ do { $g_i = 0$ }
for $i = 0$ to m do
    for $j = 0$ to r do { $g_{i+j} = g_{i+j} + \binom{r}{j}$ }

† Balance is used here to mean encoding a word to some weight between $\lceil (n-m)/2 \rceil$ and $\lceil (n+m)/2 \rceil$.

The vectors **x** and **h** are defined exactly as in section 2.2, with the replacement of (r+m) instead of r. Again, the minimum number of single maps needed is the least d such that

$$(\mathbf{g}-\mathbf{x}).\mathbf{h} \leq 0 \tag{5-11}$$

As equation (2-2) and (2-3) are equivalent (in Chapter 2), so is equation (5-11) with

$$(d^2 - 2d - \text{odd}(d)) / 4 \; - \; (d-1)\,(m+1)\,2^{r-1} \; + \; T(r,m) \; \leq \; 0 \tag{5-12}$$

where $T(r,m) = (\mathbf{g}-\mathbf{x}).\mathbf{h}$ when $d = 1$. An approximation for d is

$$d \geq 1 + T(r,m) / ((m+1)\,2^{r-1}) \tag{5-13}$$

and T(r,m) can be bounded as follows:

$$T(r,m-2) + m\,2^r \; \leq \; T(r,m) \; \leq \; T(r,m-2) + m\,2^r + 0.8\,\sqrt{r}\,2^r,$$

where $T(r,0) = 0.8\,\sqrt{r}\,2^r$ and $T(r,1) = T(r+1,0) \approx 0.8\,\sqrt{r}\,2^r$.

Solving this recurrence relation, assuming that m is even for simplicity, we get

$$(\frac{m\,(m+2)}{2} + 0.8\,\sqrt{r}\,)\,2^{r-1} \; \leq \; T(r,m) \leq (\frac{m\,(m+2)}{2} \; + \; (m+2)\;\; 0.8\,\sqrt{r}\,)\;\; 2^{r-1}.$$

Approximating (m+2)/(m+1) by 1 and substituting in (5-13),

$$\frac{m}{2} + \frac{0.8\,\sqrt{r}}{m+1} + 1 \; \leq \; d \; \leq \; \frac{m}{2} + 0.8\,\sqrt{r} + 1$$

produces the following bound on k:

$$(m+1)\,2^{r+1} - \frac{m}{2} - 0.8\,\sqrt{r} - 2 \; \leq \; k \; \leq (m+1)\,2^{r+1} - \frac{m}{2} - \frac{0.8\,\sqrt{r}}{m+1} - 2$$

This last bound is derived from (5-13), which in turn is a linear approximation to the second degree equation (5-12). When the quadratic term is very small compared to the lin-

ear term, i.e. when r and m are large, this approximation becomes more accurate. In particular, when $r > 1$ and $m \geq r/2$ or $m = 0$, the lower bound is less by at most 1 from the real value, and is used to approximate k as in:

$$k \approx (m+1)\, 2^{r+1} - \frac{m}{2} - 0.8\sqrt{r} - 2 \qquad (5\text{-}14)$$

Example 5.4 (continued) When $r = 2$ and $m = 2$, from Table 5.1 or by (5-14), we get $d = 3$, forming a $DC_2(22,20)$ code. Indeed, $d = 3$ is the optimal value with $(\mathbf{g}-\mathbf{x}).\mathbf{h} = -1 \leq 0$, as shown in Figure 5.3.

	i	g_i	x_i	h_i	$(g_i-x_i)\, h_i$
	0	1	0	1	1
i'	1	3	1	0	0
	2	4	1	-1	-3
i"	3	3	1	0	0
	4	1	0	1	1

Figure 5.3 Serial $DC_2(22,20)$ code parameters.

H	b	a	v	max(v,k-v)	b-a
01	---	9	9	11	---
01	---	10	10	10	---
01	---	11	11	11	---
00	12	1	10	10	11
00	13	0	12	12	13
10	14	3	10	10	11
11	15	2	8	12	13
00	16	4	11	11	12
10	17	5	9	11	12
10	18	6	11	11	12
11	19	7	9	11	12
11	20	8	10	10	12

Figure 5.4 Check symbols in the serial $DC_2(22,20)$ code.

Figure 5.4 lists the maps achieving the $DC_2(22,20)$ code. It can be easily verified that the conditions of Theorems 2.2 and 2.5 are satisfied for every map. As stated earlier, both

the check symbol and the weight of the codeword are used to determine the original weight(s) of the information word. For instance, if a codeword of weight 12 is received with check symbol 11, then the original weight is either 8 or 20.

r \ m	0	1	2	3	4	5	6	7	8	9	10
1	2	3	3	4	4	5	6	6	7	7	8
2	3	3	3	4	4	5	5	6	6	7	7
3	3	3	4	4	4	5	5	6	6	7	7
4	3	3	4	4	5	5	5	6	6	7	7
5	3	3	4	4	5	5	5	6	6	7	7
6	3	4	4	4	5	5	5	6	6	7	7
7	4	4	4	4	5	5	6	6	6	7	7
8	4	4	4	4	5	5	6	6	7	7	7
9	4	4	4	5	5	5	6	6	7	7	7
10	4	4	5	5	5	5	6	6	7	7	7
11	4	4	5	5	5	5	6	6	7	7	8
12	4	4	5	5	5	5	6	6	7	7	8

Table 5.1 Minimum number of single maps (d) in some serial $DC_m(k+r,k)$ codes.

(Recall that $k = (m+1)\,2^{r+1} - d - 1$).

5.3.2 Parallel $DC_m(n,k)$ Codes

In a parallel balanced code, a check symbol can be associated with only one number, whereas, in parallel $DC_m(n,k)$ code, the check symbol can determine m+1 numbers, depending on the weight of the codeword. Again, one can utilize $(m+1)\,2^r$ check symbols. Extending the construction method used in Section 2.3, we can obtain the optimal information bits of

$$k \le (m+1)\,2^r - \mathrm{odd}(r+m) \qquad (5\text{-}15)$$

This is optimal since it was shown in Section 2.3 that at least i check symbols are needed to obtain a code with i information bits for a parallel $DC_0(n,k)$ code. In the parallel $DC_m(n,k)$ code, $k \leq (m+1)\, 2^r$ since there are $(m+1)\, 2^r$ check symbols. Moreover, it was shown in Section 2.3 that $k \leq 2^r$ - odd(r) for the $DC_0(n,k)$ code. This can be generalized to $k \leq (m+1)\, 2^r$ - odd(r+m) for $DC_m(n,k)$ codes. Before discussing the construction method, consider the following example.

Example 5.5 Consider a parallel $DC_2(k+r,k)$ code with r = 2. From (5-15) we get k = 12 information bits, yielding a $DC_2(14,12)$ code. The check symbol 00 can be appended to weights 6, 7, or 8; 01 and 10 can be appended to weights 5, 6, or 7; and 11 can be appended to weights 4, 5, or 6, as shown in Figure 5.5.

Level	Weight	check symbols	g_i
0	8	00	1
1	7	00, 01, 10	3
2	6	00, 01, 10, 11	4
3	5	01, 10, 11	3
4	4	11	1

Figure 5.5 Check symbols distribution in the parallel $DC_2(14,10)$ code.

Every check symbol, along with its level (or with the weight of the resulting codeword), will determine the number of complemented bits, as shown in Figure 5.6 or Table 5.2. For instance, the check 10, when appended to a weight 5 weight to obtain a final codeword of weight 6, will determine the complementation of 8 bits for decoding. As seen in Figure 5.6, 7 bits of the information word $1^2\, 0^{14}$ are complemented and then the check 10 is appended. The dotted line in the figure shows the random walk of $1^2\, 0^{14}$.

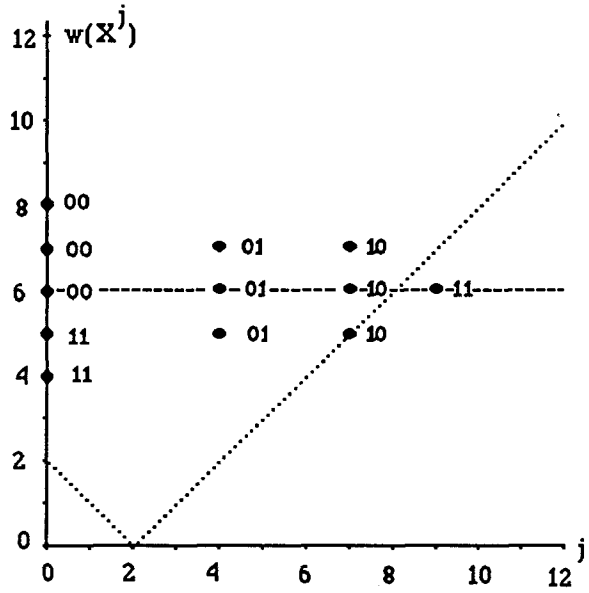


Figure 5.6 Check symbols in the parallel $DC_2(14,12)$ code.

H	Weight(x)	complemented bits (j)
00	6	0
00	7	0
00	8	0
01	5	4
01	6	4
01	7	4
10	5	7
10	6	7
10	7	7
11	4	0
11	5	0
11	6	9

Table 5.2 Check symbols in the parallel $DC_2(14,12)$ code.

The random walk of any information word must cross the k/2 line (depicted as a dashed line in Figure 5.6). The random walk of $1^2\,0^{10}$ intersects the point (7,5), so the information word is decoded into 0011 1110 0000 10, with final weight 6. The decoder, on the other hand, will complement 7 bits, since that is the number of bits determined by

the check symbol 10 and final weight 6. It can be verified that every random walk will intersect at least one check point in the space.

Let X = YH be the received codeword, where H is the check symbol. From H and W(X) the number of the complemented bits, say j, can be found from Table 5.2. The original information word will be Y with the first j bits complemented, i.e. $Y^{(j)}$.

There are g_i check symbols that can be appended to weight $\lceil (k+r+m)/2 \rceil - i$, for $0 \le i \le r+m$. Or, equivalently, there are g_i symbols with level i. In the previous example there were 1, 3, 4, 3, and 1 symbols with levels 0, 1, 2, 3, and 4, respectively. The construction algorithm given in Section 2.3.1 can be generalized is as follows. First, let A be the set of all (m+1) 2^r points. And, let D_i, for $i \ge 1$, be the maximal subset of $A - \cup_{0<j<i} D_j$ such that if X and $Y \in D_i$, then level(X) $\neq$ level(Y). The number of the different D_i's will be $g_{\lfloor (r+m)/2 \rfloor}$. All the points in D_i are assigned an integer d_i (not to be confused with d in serial decoding), where

$$d_1 = 0, \text{ and}$$
$$d_{j+1} = d_j + \frac{|D_j| + |D_{j+1}|}{2} \quad \text{for } 0 < j < g_{\lfloor (r+m)/2 \rfloor}$$

In Example 5.5, $D_1 = \{00, 00, 00, 11, 11\}$, $D_2 = \{01, 01, 01\}$, $D_3 = \{10, 10, 10\}$, and $D_4 = \{11\}$; and $d_1 = 0$, $d_2 = 4$, $d_3 = 7$, and $d_4 = 9$, as shown in Figure 5.6.

This construction method guarantees that every random walk will intersect at least one check point; therefore, every information word can be encoded. The proof is a simple generalization of of the one given in Section 2.3.2.

Chapter 6

Asymmetric/Unidirectional Error Correcting and Detecting Codes

The technique given in Chapter 4 has lead to an efficient design of t-EC/AUED, t error correcting and all *unidirectional* error detecting codes. The t-EC/AUED codes are equivalent to the t-AEC/AAED (t *asymmetric* error correcting and all *asymmetric* error detecting) codes. A balanced code that corrects t errors is also a t-EC/AUED (or a t-AEC/AAED). However, a t-EC/AUED code is not necessarily balanced.

This chapter also introduces the theory and design of codes that correct t asymmetric errors and simultaneously detect d (d>t) asymmetric errors (t-AEC/d-AED). Although these codes are not directly related to balanced codes, they generalize to t-EC/AUED codes. We chose to describe these codes before the t-EC/AUED codes. In the following some background of the previous work on different kinds of codes is given.

6.1 Previous Codes

Codes that correct t and detect d symmetric errors (t-EC/d-ED) are well-studied in the literature. These codes have a minimum Hamming distance of t+d+1. We have seen in many recent applications the errors are only of asymmetric or unidirectional type [Mor 83, Tak 76, Uye 88, Pra 80b, Mce 85]. Many error control codes that are suitable for these applications have been developed. However, the existing codes for the complete asymmetric channel can either correct or detect asymmetric errors, but not both [Ber 61,

Bor 82, Con 79, Fre 62, Var 73, Web 88]. This gave the motivation to introduce the theory and design of codes that correct t asymmetric errors and simultaneously detect d (d>t) asymmetric errors (t-AEC/d-AED). These codes have a much higher information rate than symmetric error correcting/detecting codes and yet maintain the same encoding and decoding complexity as existing error correcting codes. It is clear that when $d \geq n$, these codes become t-AEC/AAED codes.

The t-AEC/AAED (equivalently, t-EC/AUED) are studied extensively in the literature [Bos 82a, Bos 82b, Kun 88, Kun 90, Mon, Nik 86, Pra 80a, Sai 88, Tao 88]. And recently t-EC/AUED coset codes were given in [Bru 89]. We present systematic t-AEC/AAED codes that are better in some cases than the best-known t-EC/AUED codes. Also t-AEC/AAED coset codes that considerably improve the best-known coset codes [Bru 89], are presented here. These codes will then be compared with the t-error correcting balanced codes given in Chapter 3.

The last section gives a note on how to improve the design of thet symmetric error correcting and d (d>t) unidirectional error detecting (t-EC/d-UED) codes given in [Lin 88].

The construction of the above three classes of codes (t-AEC/d-AED, t-AEC/AAED, and t-EC/AUED) starts by choosing an inner t symmetric error correcting (t-EC) code, and then appending a tail check in such a way that the resulting code has the desired error detecting capability. In all three cases the tail check will be a function of the weight of the original codeword in the inner t-EC code. This will yield simple and fast encoding and decoding algorithms. From here on, we assume that the asymmetric errors are of $1 \rightarrow 0$ type. Before describing the codes we give the following lemma.

Lemma 6.1 [Nik 86]: If $D(X,Y) = d$ and $W(Y) = W(X) + j$ then

$$N(X,Y) \geq \lceil (d-j)/2 \rceil \text{ and } N(Y,X) \geq j + \lceil (d-j)/2 \rceil.$$

□

Where $D_a(X,Y)$ is the asymmetric distance between X and Y, i.e. $\max\{N(X,Y), N(Y,X)\}$.

6.2 The t-AEC/d-AED Codes

The necessary and sufficient conditions for a code to be t-AEC/d-AED (t asymmetric error correcting and d asymmetric error detecting) are stated in the following theorem.

Theorem 6.2 A code C is t-AEC/d-AED (where d>t) iff for distinct X and $Y \in C$, either

1- $N(X,Y) \geq t+1$ and $N(Y,X) \geq t+1$, or

2- $D_a(X,Y) \geq d+1$.

Proof: Assume the asymmetric errors are $1 \rightarrow 0$ type and let X and $Y \in C$. To show that condition 1 or 2 is necessary, assume that both of them are not satisfied, say $N(X,Y) \leq t$ and $D_a(X,Y) \leq d$; then $N(Y,X) \leq d$ since $N(X,Y) \leq t$. In this case, if $N(X,Y)$ ($\leq t$) errors occur in X yielding X' and $N(Y,X)$ ($\leq d$) errors occur in Y yielding Y', then it is possible that $X' = Y' = Z$; hence the decoder can not decide whether to correct or detect the errors in Z.

Before proving the sufficient condition, let us define the following. For any $X \in C$, let S_X^t be the set of words obtained from X with t or less asymmetric errors and S_X^{tt} be the set of words obtained from X with more than t but less than or equal to d asymmetric errors. Then, for any distinct X and $Y \in C$, we must show that

a) $S_X^t \cap S_X^{tt} = \emptyset$,

b) $S_X^t \cap S_Y^t = \emptyset$, and

c) $S_X^t \cap S_Y^{tt} = \emptyset$.

Condition (a), $S_X^t \cap S_X^{tt} = \emptyset$, is clear since the weight of any vector in S_X^t is less than

the weight of any vector in S^{ψ}_{X}. To prove (b) and (c), consider the two cases.

Case 1 If $N(X,Y) \geq t+1$ (and $N(Y,X) \geq t+1$), then for any $U \in S^{\downarrow}_{X}$, $V \in S^{\downarrow}_{Y}$, and $W \in S^{\psi}_{Y}$, $N(U,W) \geq 1$ and $N(U,W) \geq 1$, so $S^{\downarrow}_{X} \cap S^{\downarrow}_{Y} = \varnothing$ and $S^{\downarrow}_{X} \cap S^{\psi}_{Y} = \varnothing$.

Case 2 If $N(X,Y) \leq t$ then, by the second condition ($D_a(X,Y) \geq d+1$), we have $N(Y,X) \geq d+1$. Again, $S^{\downarrow}_{X} \cap S^{\downarrow}_{Y} = \varnothing$ and $S^{\downarrow}_{X} \cap S^{\psi}_{Y} = \varnothing$, because if $U \in S^{\downarrow}_{X}$, $V \in S^{\downarrow}_{Y}$, and $W \in S^{\psi}_{Y}$, then $N(V,U) \geq (d+1) - t \geq 1$ and $N(W,U) \geq (d+1) - d = 1$.

□

To construct a t-AEC/d-AED code, we start with an inner t-EC code (say a BCH code) and then append a tail check in such a way that the resulting code can also detect d asymmetric errors. The tail check will be a function of the weight of the original codeword in the inner ec code. This tail check is referred to as a D-tail sequence, or a D-sequence for short. The D-sequence of length r, with parameters t and d, is defined as follows:

Let $D[r,t,d] = \{s_0, s_1, \ldots s_{m-1}\}$ be a D-sequence where each s_i is of length r; then for $1 \leq i,j \leq m$, either

1) $N(s_i,s_{i+j}) \geq \min(\lceil j/2 \rceil, t+1)$ or

2) $N(s_{i+j},s_i) \geq (d+1) - \max(j, j+\lceil (2t+1-j)/2 \rceil)$

where $i+j$ in s_{i+j} is computed modulo m.

When a D[r,t,d] sequence is appended to a t-EC code, a t-AEC/d-AED code is produced.

6.2.1 Systematic t-AEC/d-AED Codes

These codes can be constructed as follows:

Construction 6.1

1- Let C' be any (n',k,2t+1) systematic code, and

2- Let D[r,t,d] = $\{s_0, s_1, ..., s_{m-1}\}$, where r is as small as possible; then

3- C = $\{X\, s_{W(X) \bmod m} \mid X \in C'\}$ is a t-AEC/d-AED code of length n = n' + r.

Encoding: An information vector U of length k will first be encoded by C' to X of length n', and the final codeword will be XY where $Y = s_{W(X) \bmod m}$.

Decoding: Let V = XY and V' = X'Y' be the transmitted and received codewords, respectively.

1- Let X" be the corrected word of X' using C'.

2- Let the corrected codeword be V" = X"Y" where $Y'' = s_{W(X'') \bmod m}$.

3- If N(V",V') > t or N(V',V") > 0, then declare uncorrectable errors; otherwise,

4- The k information bits extracted from X" is the decoded information vector.

Theorem 6.3 The above construction method yields a t-AEC/d-AED code. Assume that some codeword suffered e asymmetric† errors; then the decoding algorithm will correct the errors if $e \leq t$ and will detect them if $t < e \leq d$.

Proof: It is relatively easy to show that all codewords satisfy the conditions of Theorem 6.2. To show the validity of the decoding algorithm, let V, V', and V" be the send, received, and corrected codewords, respectively. Assume that V suffered e asymmetric errors resulting in V'. Let the corrected number of asymmetric errors in V' be e', resulting in V". If e' > t, then the errors are detected, so suppose that $e' \leq t$. If $e \leq t$, then V = V" (since C' is a t-EC code) and the errors are corrected. What remains to be

† In fact, if it is guaranteed that no *symmetric* errors occur in the tail check, then the previous code can correct any t or fewer *symmetric* errors and detect any d or fewer *asymmetric* errors.

shown is that when $t < e \leq d$, the errors are detected. But V and V" are codewords in the t-AEC/d-AED code, so either $N(V,V") \geq t+1$ and $N(V",V) \geq t+1$ or $D_a(V",V) \geq d+1$. Consider these two cases:

Case 1 $N(V,V") \geq t+1$ and $N(V",V) \geq t+1$. Here $N(V",V) \geq t+1$ implies $N(V",V') \geq 1$, and hence the errors are detected.

Case 2 $D_a(V",V) \geq d+1$. So either $N(V,V") \geq d+1$ or $N(V",V) \geq d+1$. The case $N(V",V) \geq d+1$ is handled the same as above. Now $N(V,V") \geq d+1$ implies $N(V',V") \geq 1$, and hence the errors will be detected.

□

The design of efficient t-EC codes and D-sequences results in efficient t-AEC/d-AED codes. In particular, when r is relatively small, the sequence $D[r,t,t+r] = \{1^{(r-1)-i}\,0^i\,1,\ 1^{(r-1)-i}\,0^i\,0 \mid 0 \leq i \leq r-1\}$ can be used to design efficient t-AEC/d-AED codes.

Theorem 6.4 Any t-EC code can be changed into a t-AEC/(t+r)-AED code by merely adding an extra r check bits.

Proof: Simply append the D-sequence, $D[r,t,t+r] = \{1^{(r-1)-i}\,0^i\,1,\ 1^{(r-1)-i}\,0^i\,0 \mid 0 \leq i \leq r-1\}$ to the t-EC code.

□

Example 6.1 Let C' be the (15,11,3) Hamming single error correcting code. With an extra 4 bits, a 1-AEC/5-AED code of length 19 with 11 information bits can be constructed by appending the sequence

$$D[4,1,5] = \begin{matrix} 1111 \\ 1110 \\ 1101 \\ 1100 \\ 1001 \\ 1000 \\ 0001 \\ 0000 \end{matrix}$$

There are 16 different weights in C'. The tail check of any word of weight i will be $s_{i \bmod 8}$, i.e. words of weight 0 and 8 will have the tail check $s_0 = 1111$, words of weight 1 and 9 will have the tail check $s_1 = 1110$, etc. Notice that a (26,11,7) BCH code of distance 7 that corrects 1 and detects 5 *symmetric* errors has 15 (rather than 8) check bits. In addition, the 1-EC/d-UED codes given in [Lin 88] can detect up to 4 unidirectional errors using an extra 4 check bits. Also note that the 1-EC/d-UED codes given in [Lin 88] are designed by appending a tail check to a distance 2t+2 code, whereas here we use distance 2t+1 inner codes.

□

When r increases, one can obtain a detection capability higher than t+r. For example, when r is 9, one can obtain a 1-AEC/27-AED code using a D[9,1,27] sequence. Table 6.1 lists some t-AEC/d-AED codes obtained by appending D-sequences, constructed in Section 6.5, to BCH (or shortened BCH) t error correcting codes. It should be noted that the 0-AEC/d-AED codes were originally obtained in [Bos 85].

The lower bound theorem, given in [Lin 88], on the number of check bits required for any systematic t-EC/d-UED code is also valid for t-AEC/d-AED codes, even though the latter codes are less restrictive than the former. The theorem given in [Lin 88] can be modified slightly to read as follows.

Theorem 6.5 [Lin 88] In any systematic t-AEC/d-AED code of length n with k information bits, the following condition must hold:

$$n\text{-}k \geq \log \left(\sum_{i=0}^{t+1} \binom{k}{i} - \binom{k\text{-}d+t}{t+1} \right)$$

Proof: Consider the following sets of k-tuple information words.

Let $B_0 = \{X = (x_0, x_1, \ldots x_{k-1}) \mid W(X) \leq t\}$, and

$B_i = \{X = (x_0, x_1, \ldots x_{k-1}) \mid W(X) \leq t+i \text{ and } x_j = 1 \text{ for } 0 \leq j \leq i-1\}$, for $1 \leq i \leq d-t$.

Let $B = \bigcup_{i=0}^{d-t} B_i$. If X and Y are in B, then $D_a(X,Y) < d+1$ since $W(X) < d+1$ and $W(Y) < d+1$. Moreover, from the way these words are selected, either $N(X,Y) < t+1$ or $N(Y,X) < t+1$. So every word in B requires a distinct check symbol; thus $n-k \geq \log |B|$ and

$$|B| = \sum_{i=0}^{t} \binom{k}{i} + \sum_{i=1}^{d-t} \binom{k-i}{t} = \sum_{i=0}^{t} \binom{k}{i} + \binom{k}{t+1} - \binom{k-d+t}{t+1}$$

□

r	t = 0	t = 1	t = 2	t = 3	t = 4
2	2	3	4	5	6
3	3	4	5	6	7
4	6	5	6	7	8
5	11	6	7	8	9
6	22	8	8	9	10
7	43	10	9	10	11
8	84	14	11	11	12
9	165	27	13	12	13
10	326	54	15	14	14
11	647	81	17	16	15
12	1288	142	22	18	17
13		263	43	20	19
14		526	86	22	21
15		789	129	24	23
16		1262	172	30	25
17			278	59	27
18			478	118	29
19			717	177	31
20					33
21					75
22					150

Table 6.1 Values of d of some t-AEC/d-AED codes for $0 \leq t \leq 4$.

Where r is the length of the D-sequence, i.e. the added redundancy to the t-EC code.

Berger has shown that a systematic code with k information bits that detects all

asymmetric errors must have at least $\lceil \log (k+1) \rceil$ check bits [Ber 61]. This bound becomes a special case of the previous theorem, i.e. when t = 0 and d≥k.

When t = 0 or t = 1 and d is relatively small, the number of redundant bits in Construction 6.1 matches the lower bound, obtained in Theorem 6.5, or differs by 1 or 2 bits. The gap between the actual redundancy of Construction 6.1 and the theoretical lower bound increases as t increases.

6.2.2 Non-Systematic t-AEC/d-AED Codes

A non-systematic t-AEC/d-AED code can be obtained by using a non-systematic t-EC inner code (C') in Construction 6.1.

Another method of constructing a non-systematic t-AEC/d-AED code of length n is as follows. Let S_i be the maximal set of codewords of length n and weight i where $N(X,Y) \geq t+1$ for any X and Y in S_i; i.e. S_i is the maximal t-AEC/AAED set of length n and weight i. Then the set $S = \bigcup S_i$ where $i = \lfloor n/2 \rfloor \bmod (d+1)$, forms a t-AEC/d-AED code. This construction is a generalization of the 0-AEC/d-AED code given in [Bor 82]. The non-systematic 0-AEC/d-AED code given in [Bor 82] was shown to be optimal. In the following we give an upper bound for the 1-AEC/AAED codes and then show that this construction obtains about half this upper bound.

Theorem 6.6 Let C be any 1-AEC/d-AED code of length n; then

$$|C| \leq \frac{n}{2} \sum_{i=0} \binom{n}{\lfloor n/2 \rfloor \pm i.d}.$$

Proof: Let C be a 1-AEC/d-AED code of length n which gives the maximum number of codewords. The total number of 1's (or 0's) in C will be at least $\frac{n}{2}|C|$. Without loss of generality, assume that the total number of 1's is more than the total number of 0's (if not, then the complement 1-AEC/d-AED code $\overline{C}$ has more 1's than 0's). For any $X \in C$, let

S_X be the set of words obtained by a single $1\rightarrow0$ error from X. Since C is capable of correcting all single errors, for all distinct X and $Y \in C$ we will have $S_X \cap S_Y = \varnothing$. Furthermore, for any $X_1 \in S_X$ and $Y_1 \in S_Y$, we will have either $D(X_1,Y_1) \geq d$ or $N(X_1,Y_1) \geq 1$ and $N(Y_1,X_1) \geq 1$. Also, for any $X_1, X_2 \in S_X$ we have $N(X_1,X_2) \geq 1$ and $N(X_2,X_1) \geq 1$. Let $S = \cup S_X$ for all $X \in C$; hence $|S| \geq \frac{n}{2}|C|$. The words in S form a 0-AEC/(d-1)-AED, i.e. a (d-1)-asymmetric error detecting code. From the bound in [Bor 82], we have $|S| \leq \sum_{i=0} \binom{n}{\lfloor n/2 \rfloor \pm i.d}$.

But $|S| \geq \frac{n}{2}|C|$, and therefore $|C| \leq \frac{2}{n} \sum_{i=0} \binom{n}{\lfloor n/2 \rfloor \pm i.d}$.

□

1-AEC/d-AED codes with the number of codewords close to the above bound can be constructed as follows. To form S_i in the above construction method, we use the group theoretic method given in [Bos 82a]. In their method, an Abelian group of order n is used to partition all the words of a certain weight, say i, into n partitions such that every partition is a 1-AEC/AAED code. Moreover, the size of at least one partition, say S_i, will be $|S_i| \geq \frac{1}{n}\binom{n}{i}$; therefore

$$|S| \geq \frac{1}{n} \sum_{i=0} \binom{n}{\lfloor n/2 \rfloor \pm i.(d+1)}.$$

6.3 The t-AEC/AAED Codes

The necessary and sufficient condition for a code to be t-AEC/AAED (t asymmetric error correcting and all asymmetric error detecting) is stated in the following theorem.

Theorem 6.7 [Bos 82b, Pra 77, Pra 80a] A code C is t-AEC/AAED iff $N(X,Y) \geq t+1$ and $N(Y,X) \geq t+1$ for any $X, Y \in C$.

□

It should be noted that the above condition also holds for t-EC/AUED codes, t

symmetric error correcting and all unidirectional error detecting, as given in [Bos 82b, Pra 77, Pra 80a]; so the t-AEC/AAED and t-EC/AUED codes are equivalent.

6.3.1 The t-AEC/AAED Systematic Codes

The construction of the t-AEC/AAED codes is similar to that of the t-AEC/d-AED codes. It starts by choosing a t-EC code and then appending a tail check, called an A-sequence, in such a way that the resulting code is able to detect all asymmetric errors. The tail check will be a function of the weight of the original codeword of the inner EC code.

An A-sequence of length r and strength t is defined as follows [Bla 89]:

$A[r,t] = \{s_0, s_1, ..., s_{m-1}\}$ where $N(s_i,s_{i+j}) \geq \min(\lceil j/2 \rceil, t+1)$ for $0 \leq j < m-i$. An even A-sequence can be defined as

$A_e[r,t] = \{s_0, s_1, ..., s_{m-1}\}$ where $N(s_i,s_{i+j}) \geq \min(j,t+1)$ for $0 \leq j < m-i$. Again, the s_i's are of length r.

Notice that an A-sequence is just a D-sequence with $d = \infty$, i.e. $A[r,t] = D[r,t,\infty]$. The codes can now be constructed as follows.

Construction 6.2

1- Let C' = (n',k,2t+1) systematic code with $a \leq W(X) \leq b$ for any $X \in C'$.

2- Let $A[r,t] = \{s_0, s_1, ..., s_{m-1}\}$, where $m \geq b-a+1$ (and r is as small as possible);

3- Then, $C = \{X\, s_{W(X)-a} \mid X \in C'\}$ is a t-AEC/AAED code of length $n = n' + r$.

Similarly, if C' = (n',k,2t+2) is an even weight code with $a \leq W(X) \leq b$ for any $X \in C'$ and $A_e[r,t] = \{s_0, ..., s_{m-1}\}$, where $m \geq (b-a)/2+1$, then $C = \{X\, s_{(W(X)-a)/2} \mid X \in C'\}$ is a t-AEC/AAED code of length $n = n' + r$.

Encoding: A vector U will be first encoded by C' to X, and the final codeword will be XY where $Y = s_{W(X)-a}$.

Decoding: Let V = XY and V = X'Y' be the transmitted and received codewords, respectively, then

1- Let X" be the corrected word of X' using C'.

2- Let the corrected codeword be V" = X"Y" where $Y'' = s_{W(X'')-a}$.

3- If D(V",V') > t , then declare uncorrectable errors and stop; otherwise

4- The k information bits extracted from X" are the decoded information vector. □

The redundancy of the code can be minimized when the check symbols in the A-sequence are increased and when the weight distribution of the inner code is reduced. In Section 6.5 we construct efficient A_e and A-sequences. In many cases the systematic codes obtained using these sequences are better than those previously known [Bla 89].

The following theorem gives a lower bound on the number of redundant bits in any systematic t-AEC/AAED code. This theorem is a slight extension to the one given in [Bos 82b].

Theorem 6.8 In any systematic t-AEC/AAED code with k information bits and r check bits, the following condition must hold:

$$2^r - 2\sum_{i=0}^{t}\binom{r}{i} \geq 2\sum_{i=0}^{t}\binom{k}{i} + \binom{k}{t+1} - \binom{2t+1}{t+1} - 2$$

Proof: Consider the following sets of k-tuple information words.

$B_0 = \{X = (x_0, x_1, \ldots x_{k-1}) \mid W(X) \leq t \text{ or } W(X) \geq k-t\}$, and

$B_i = \{X = (x_0, x_1, \ldots x_{k-1}) \mid W(X) \leq t+i \text{ and } x_j = 1 \text{ for } 0 \leq j \leq i-1\}$, for $1 \leq i \leq k-2t-1$.

Then define $B = U_{i=0}^{d-t} B_i$ - (00...000) - (11...111). For any words X and Y in B, either

$N(X,Y) < t+1$ or $N(Y,X) < t+1$. So every word in B needs a distinct check symbol. Moreover, if Y is the check symbol of a word X in B, then Y must have at least t+1 zeros and at least t+1 ones. This is because the check of (00...000) must cover Y by at least t+1 ones; similarly, Y must cover the check of (11...111) by at least t+1 ones. So only words of weights more than t+1 and less than r - (t+1) can be used as checks for the set B, i.e. only $2^r - 2\sum_{i=0}^{t}\binom{r}{i} \geq |B|$, so

$$|B| = 2\sum_{i=0}^{t}\binom{k}{i} + \binom{k}{t+1} - \binom{2t+1}{t+1} - 2.$$

□

From the above theorem, we see that the proposed codes are optimal or close to optimal for small values of t.

6.3.2 The t-AEC/AAED Coset Codes

A non-systematic t-AEC/AAED code can be obtained by using a non-systematic t-EC inner code (C') in Construction 6.2. Note that an inner code C' with a narrow weight distribution needs a smaller size A-sequence. When linear codes are used as t-AEC/AAED inner codes, the A-sequence must have size n+1 since linear codes always contain the all-zero and the all-one vectors. Coset codes, which are partitions of linear codes, may have narrow weight distribution and still have encoding and decoding complexity as linear codes.

In this section we will use a codes similar to the C_t and $C_{t,w}$ of Section 4.2 and 4.3 that have narrow weight distribution. The corresponding code for C_t is referred to as C_h which is obtained by any half-weight vector (as opposed to the $\underline{10}$ vector for the C_t code). The $C_{h,f}$ code is constructed from a C_h code and the full-weight control vector ($\underline{1}$). The C_h and $C_{h,f}$ codes are used as inner codes (C') to construct the t-AEC/AAED codes. The

following lemma is the basis of these codes. This lemma parallels Lemma 5.1.

Lemma 6.9 Let X, H, and F be binary vectors of length n where $W(H) = \lceil n/2 \rceil$ and $W(F) = n$; then

a) $\lceil n/4 \rceil \le W(Y) \le \lfloor (3n+1)/4 \rfloor$ for $Y = X$ or $X \oplus H$.

b) If $\lceil n/4 \rceil \le W(X) \le \lfloor (3n+1)/4 \rfloor$, then

$\lfloor (n+2)/4 \rfloor \le W(Y) \le \lfloor n/2 \rfloor$ for $Y = X$ or $X \oplus F$.

proof: Suppose that $\lceil n/4 \rceil \le W(X) \le \lfloor (3n+1)/4 \rfloor$ does not hold, and assume that $W(X) > \lfloor (3n+1)/4 \rfloor$. In this case, when X is XORed with H, the number of 1's that will be changed to 0's is at least $\lceil n/4 \rceil$ since $W(X) > \lfloor (3n+1)/4 \rfloor$ (and at most $\lceil n/2 \rceil$ since $W(H)=\lceil n/2 \rceil$). So, if $W(X) > \lfloor (3n+1)/4 \rfloor$, then $\lceil n/4 \rceil \le W(X \oplus H) \le \lfloor (3n+1)/4 \rfloor$. Also, if $W(X) < \lceil n/4 \rceil$, then $\lceil n/4 \rceil \le W(X \oplus H) \le \lfloor (3n+1)/4 \rfloor$. For the second part, assume that $\lfloor (n+2)/4 \rfloor \le W(Y) \le \lfloor n/2 \rfloor$ does not hold. Then $\lfloor (n+2)/4 \rfloor \le W(\overline{Y}) \le \lfloor n/2 \rfloor$ will hold since $\lfloor (n+2)/4 \rfloor \le \lceil n/4 \rceil$ and $\lfloor (n+2)/4 \rfloor = n - \lfloor (3n+1)/4 \rfloor$.

□

When $W(H) = \lfloor n/2 \rfloor$ in the above lemma, then $\lceil n/4 \rceil$ is changed to $\lceil (n-1)/4 \rceil$ as follows:

a- $\lceil (n-1)/4 \rceil \le W(Y) \le \lfloor (3n+1)/4 \rfloor$ for $Y = X$ or $X \oplus H$.

b- If $\lceil (n-1)/4 \rceil \le W(X) \le \lfloor (3n+1)/4 \rfloor$, then

$\lfloor (n+2)/4 \rfloor \le W(Y) \le \lfloor n/2 \rfloor$ for $Y = X$ or $X \oplus F$.

The C_h Codes

Let C' be an (n,k+1,d) linear code of length n, 2^{k+1} codewords, and minimum Hamming distance d. The encoding and decoding of these codes are almost the same as the C_h codes given in Section 4.2. The only difference is the here we use any half-weight vector say H, that belongs to C' instead of the alternating 1's and 0's vector (10). In this section, we adapt encoding and decoding methods of the C_h codes similar to that of [Bru

89] rather than the approach given for the C_t codes in Section 4.2 which uses method similar to [Den 88].

Encoding: Let C' = (n,k+1,d) be any linear code and let H ∈ C' where W(H) = $\lceil n/2 \rceil$ and the first bit of H is 1. Let U = u1 u2 ... uk be the information vector. Then U is encoded to X as follows:

1- Encode <0,U> to a word X in C'.

2- If W(X) < $\lceil n/4 \rceil$ or W(X) > $\lfloor (3n+1)/4 \rfloor$ then let X = X ⊕ H.

From Lemma 6.9 (a) it is guaranteed that either $\lceil n/4 \rceil \le W(X) \le \lfloor (3n+1)/4 \rfloor$ or $\lceil n/4 \rceil \le W(X \oplus H) \le \lfloor (3n+1)/4 \rfloor$.

Decoding: Let X' be the received word.

1- Decode X' to X using the code C'. If there t or more errors, then stop; otherwise

2- Let $<a_h,U> = a_h, u_1, u_2, ..., u_k$ be the k+1 information bits extracted from X; then

3- $U' = U \oplus a_h H$ will be the decoded information symbol.

□

This C_h = (n,k+1,d) code of length n and distance d, will have all its 2^k codewords with weights between $\lceil n/4 \rceil$ and $\lfloor (3n+1)/4 \rfloor$. The number of different weights is not more than $\lceil n/2 \rceil$+1. Actually, the number of weights will be $\lceil n/2 \rceil$+1 if n ≡ 0 (mod 4) and $\lceil n/2 \rceil$ otherwise.

If the $\underline{1}$ vector, F, is used instead of the half-weight vector, H, then words of weights between 0 and $\lfloor n/2 \rfloor$ (or between $\lceil n/2 \rceil$ and n) can be selected. This idea is used in [Bru 89] to obtain $\lfloor n/2 \rfloor$+1 different weights. The main drawback of their code is that the $\underline{1}$ vector must be a codeword in C'. Note that in C_h, a half-weight vector is required to be in C', but the chance of having a half-weight vector is much larger than of having the $\underline{1}$ vector.

The $C_{h,f}$ Codes

When the $\underline{1}$ vector (F) is also in C', both vectors, H and F, can be used to narrow down the weight distribution. In this case, two information bits are used to control the weight as done in the $C_{t,w}$ code design in Section 4.3. Again a different approach to the encoding and decoding of these codes, is given.

Encoding: Let C' = (n,k+1,d) be any linear code. Let the all 1 vector F be in C' and let $H \in C'$ where $W(H) = \lceil n/2 \rceil$ and the first two bits of H are 1 and 0. Let U = u1 u2 ... uk be the information vector. Then U is encoded to X as follows:

1- Encode <0,U> to a word X in C'.

2- If $W(X) < \lceil n/4 \rceil$ or $W(X) > \lfloor (3n+1)/4 \rfloor$ then let $X = X \oplus H$.

3- If $W(X) \geq \lceil n/4 \rceil$ then let $X = X \oplus F$.

By Lemma 6.9, we can see that X will satisfy $\lfloor (n+2)/4 \rfloor \leq W(X) \leq \lfloor n/2 \rfloor$.

Decoding: Let X' be the received word.

1- Decode X' to X using C'. If there are t or more errors, then stop; otherwise

2- Let $<a_h, a_f, U> = a_h, a_f, u_1, ..., u_k$ be the k+2 information bits extracted from X.

3- Then, $U' = U \oplus (a_h \oplus a_f) H \oplus a_f F = U \oplus a_h H \oplus a_f \overline{H}$

will be the decoded information symbol.

□

A $C_{h,f}$ = (n,k+2,d) code of length n and distance d will have all its 2^k codewords with weights between $\lfloor (n+2)/4 \rfloor$ and $\lfloor n/2 \rfloor$. There will be $\lfloor n/2 \rfloor - \lfloor (n+2)/4 \rfloor + 1 = \lfloor n/4 \rfloor + 1$ different weights in this code.

Design of the t-AEC/AAED Coset Codes

Now the t-AEC/AAED coset codes can be constructed by simply using a C_h or a $C_{h,f}$

inner code (C') in Construction 6.2. The encoding and decoding of these codes will have the extra step, over a non-coset code, of the possible XORing with H and/or F vectors. When one or two bits are used for weight control, these bits are transformed from information to redundant bits. However, the gain is paid off by a significantly shorter A-sequence needed for this narrower weight distribution. So, given the A-sequences, the overall minimum code length can be found by trying different inner (coset and non-coset) codes. Table 6.2 illustrates some improvements of the proposed coset codes over the previous (coset or systematic) codes.

t = 1		
k	new	old
9	17	19
12	22	24
15	25	27
20	31	33
21	32	34
24	35	37
43	56	58
67	82	84
70	85	87
73	88	90

t = 2		
k	new	old
5	17	20
9	24	27
14	29	34
19	35	39
22	42	45
23	43	46
24	43	47
26	46	49
29	49	52
34	54	57
39	60	63
44	65	68
49	70	73
56	79	82

t = 3		
k	new	old
7	27	32
14	35	41
22	47	54
29	56	61
31	59	64
32	60	65
33	61	66
36	63	69
43	71	76
55	87	92
62	94	99
69	101	106
76	108	113
83	115	120

t = 4		
k	new	old
9	35	42
19	51	59
28	61	69
32	68	74
33	69	75
34	70	76
35	71	77
36	72	78
37	71	79
43	83	89
52	92	99
61	102	108
70	111	117
79	120	127

Table 6.2 Length of some new and old t-AEC/AAED coset codes, for $1 \leq t \leq 4$.

Detailed Construction the t-AEC/AAED Coset Codes

Tables 6.3 to 6.6 give detailed construction parameters of the codes listed in Table 6.2. The new t-AEC/AAED codes of length n and k information bits are constructed using inner

(n',k+m,d) codes where m is the number of control vectors; i.e. when m = 0, 1, or 2 the inner code is a systematic, C_h, or $C_{h,f}$ code, respectively. The 'wt. dist.' column indicates the weight distribution of the inner code. The '# wts' column gives the number of different weights (or even weights when d is even). The 'r' column indicates the length of the smallest A-sequence (or A_e-sequence when d is even) with at least '# wts' words. The A_e-sequences and A-sequences developed in Section 6.5 and listed in Tables 6.8 and 6.9 are used for this purpose. The new code length (n) will be n' + r.

k	n	(n',k+m,d)	wt. dist.	# wts	r
9	17	(15,9+2,3)	(4, 7)	4	2
12	22	(19,12+2,3)	(5, 9)	5	3
15	25	(22,15+2,3)	(6,11)	6	3
20	31	(26,20+1,3)	(7,19)	13	5
21	32	(27,21+1,3)	(7,20)	14	5
24	35	(31,24+2,3)	(8,15)	8	4
43	56	(51,43+2,3)	(13,25)	13	5
67	82	(76,67+2,3)	(19,38)	20	6
70	85	(79,70+2,3)	(20,39)	20	6
73	88	(83,73+2,4)	(21,41)	10	5

Table 6.3 Construction parameters of some 1-AEC/AAED coset codes.

k	n	(n',k+m,d)	wt. dist.	# wts	r
5	17	(15, 5+2,5)	(4, 7)	4	2
9	24	(21,9+2,5)	(5,10)	6	3
14	29	(27,14+2,6)	(7,13)	3	2
19	35	(31,19+2,5)	(8,15)	8	4
22	42	(35,22+1,5)	(9,26)	18	7
23	43	(36,23+1,5)	(9,27)	19	7
24	43	(38,24+2,5)	(10,19)	10	5
26	46	(39,26+1,5)	(10,29)	20	7
29	49	(43,29+2,5)	(11,21)	11	6
34	54	(49,34+2,6)	(12,24)	6	5
39	60	(53,39+2,5)	(13,26)	14	7
76	100	(92,76+2,5)	(23,46)	24	8

Table 6.4 Construction parameters of some 2-AEC/AAED coset codes.

k	n	(n',k+m,d)	wt. dist.	# wts	r
7	27	(25, 7+2,8)	(7,12)	3	2
14	35	(31,14+2,7)	(8,15)	8	4
22	47	(43,22+2,8)	(11,21)	5	4
29	56	(49,29+2,7)	(12,24)	13	7
31	59	(50,31+1,7)	(13,37)	25	9
32	60	(51,32+1,7)	(13,38)	26	9
33	61	(53,33+1,8)	(14,39)	13	8
36	63	(57,36+2,8)	(15,28)	7	6
43	71	(63,43+2,7)	(16,31)	16	8
55	87	(78,55+2,7)	(20,39)	20	9
62	94	(85,62+2,7)	(21,42)	22	9
69	101	(92,69+2,7)	(23,46)	24	9
76	108	(99,76+2,7)	(25,49)	25	9
83	115	(107,83+2,8)	(27,53)	13	8

Table 6.5 Construction parameters of some 3-AEC/AAED coset codes.

k	n	(n',k+m,d)	wt. dist.	# wts	r
9	35	(31,9+2,9)	(8,15)	8	4
19	51	(45,19+2,9)	(11,22)	12	6
28	61	(54,28+2,9)	(14,27)	14	7
32	68	(57,32+1,9)	(14,42)	29	11
33	69	(58,33+1,9)	(15,43)	29	11
34	70	(59,34+1,9)	(15,44)	30	11
35	71	(60,35+1,9)	(15,45)	31	11
36	72	(61,36+1,9)	(15,45)	31	11
37	71	(63,37+2,9)	(16,31)	16	8
43	83	(73,43+2,9)	(18,36)	19	10
52	92	(83,52+2,10)	(21,41)	10	9
61	102	(91,61+2,9)	(23,45)	23	11
70	111	(100,70+2,9)	(25,50)	26	11
104	150	(138,104+2,9)	(35,69)	35	12

Table 6.6 Construction parameters of some 4-AEC/AAED coset codes.

Example 6.2 Consider the first code in Table 6.3, i.e. the 1-AEC/AAED code of length 17 and 9 information bits. First the (15,11,3) code is changed to a $C_{h,f}$ (15,9+2,3) code with 9 information bits and weight distribution between 4 and 7. This inner code has 4 different weights. From Table 6.9 the A-sequence of size 4 in the entry t = 1 and r = 2 can then be appended to the resulting weights to form a 1-AEC/AAED code of length 17 and 9 information bits.

6.4 The t-EC/d-UED Codes

The necessary and sufficient conditions for a code to be t-EC/d-UED (t error correcting and d unidirectional error detecting) are stated in the following theorem.

Theorem 6.10 [Lin 88] A code C is t-EC/d-UED iff for any X and Y $\in$ C, one or more of the following conditions holds.

1) $N(X,Y) \geq t+1$ and $N(Y,X) \geq t+1$, or

2) $D(X,Y) \geq t+d+1$.

□

The code design of the t-EC/d-UED codes is similar to that of t-AEC/d-AED codes, but U-sequences will be used instead of D-sequences. A U-sequence U[r,t,d] of length r is defined as follows:

Let $U[r,t,d] = \{s_0, s_1, \ldots s_{m-1}\}$ be a U-sequence; then for $1 \leq i,j \leq m$, either

1) $N(s_i,s_{i+j}) \geq \min(\lceil j/2 \rceil, t+1)$ or

2) $D(s_i,s_{i+j}) \geq \min(d-t-1+\text{odd}(j), (t+1+d)-j)$

where i+j in s_{i+j} is computed modulo m and where s_i's are of length r.

Some efficient U[r,t,d] sequences are constructed in Section 6.5. The new t-AEC/d-AED codes which are constructed using these sequences improve the ones given

in [Lin 88] as summarized in Table 6.7. These codes become more superior as r and t increase.

t = 1			t = 2			t = 3			t = 4		
r	new	old	r	new	old	r	new	old	r	new	old
8	13	12	12	20	19	15	18	17	20	34	33
9	26	26	13	41	41	16	27	26	21	71	71
10	53	34	14	84	49	17	56	56	22	146	71
11	80	70	15	127	81	18	115	64	23	221	87
12	141	130	16	170	137	19	174	92	24	296	103
13	262	262	17	276	237	20	233	172	25	371	155
14	525	314	18	476	339	21	348	348	26	446	229
15	788	630	19	715	541	22	699	668	27	521	353
16	1261	1162	20	1084	914	23	1340	1340	28	710	527

Table 6.7 Values of d of some new (and old [Lin 88]) t-EC/d-UED codes, for $1 \leq t \leq 4$. (Where r is the added redundancy to a t-EC code).

6.5 Design of Tail Sequences

The design of the tail sequences is given in the following order: the A-sequences, the U-sequences, and finally the D-sequences.

6.5.1 Design of A-sequences

These sequences are defined as follows:

$A[r,t] = \{s_0, s_1, ..., s_{m-1}\}$ where $N(s_i,s_{i+j}) \geq \min(\lceil j/2 \rceil, t+1)$ for $0 \leq j < m-i$, and

$A_e[r,t] = \{s_0, s_1, ..., s_{m-1}\}$ where $N(s_i,s_{i+j}) \geq \min(j,t+1)$ and $0 \leq j < m-i$,

where the s_i's are of length r.

If $A_e[r-1,t] = \{s_0, s_1, ..., s_{m-1}\}$ (of size m), then it can be easily shown that the sequence $\{<s_0,1>, <s_0,0>, <s_1,1>, <s_1,0>, ..., <s_{m-1},1>, <s_{m-1},0>\}$ is an A[r,t] sequence (of size 2m). Hence, the relation $| A[r,t] | \geq 2 | A_e[r-1,t] |$ is obtained. We will

only consider the design of $A_e[r,t]$ sequences. Consider the following two methods.

Method 1: Notice that if $A_e[r-1,t] = \{s_0, s_1, ..., s_{m-1}\}$ (of size m), then $\{<s_0,1>, <s_0,0>, <s_1,0>, <s_2,0>, ..., <s_{m-1},0>\}$ is an $A_e[r,t]$ sequence (of size m+1). Starting with the base case $A_e[1,t] = \{1,0\}$, the $A_e[r,t] = \{1^{r-i}\,0^i \mid 0 \le i \le r\}$ (of size r+1) can be produced.

The method can be improved by realizing that in some cases more than one word can be added to $A_e[r-1,t]$. For instance, when $| A_e[r-1,t] | \ge 2t+2$, i.e. $m \ge 2t+2$, the following is an $A_e[r,t]$ sequence of length m+2:

$\{<s_0,1>, <s_0,0>, <s_1,0>, .., <s_t,0>, ..., <s_{m-t},1>, .., <s_{m-2},1>, <s_{m-1},1>, <s_{m-1},0>\}$

So when $A_e[r-1,t] \ge 2t+2$, one can add at least two words, one at the top and another at the bottom, as done in the previous sequence. As the size of $A_e[r-1,t]$ increases, more words can be added, as follows. Let $B = \{s_{t+1}, s_1, ..., s_{m-t-2}\}$ and let $b_1, b_2, ..., b_{t+1}, ..., b_{2t+1}$ be any 2t+1 consecutive words of B. By "splitting" b_{t+1}, the following A_e-sequence is produced.

$\{<b_1,1>, <b_2,1>, .., <b_t,1>, <b_{t+1},1>, <b_{t+1},0>, <b_{t+2},0>, .., <b_{2t},0>, <b_{2t+1},0>\}$

In general, to construct $A_e[r,t]$ from $A_e[r-1,t]$, add one word at the top and another at the bottom, and then transform every consecutive 2t+1 words in $\{s_{t+1}, s_1, ..., s_{m-t-2}\}$ to 2t+2 words in $A_e[r,t]$. The following relation can be obtained.

$$A_e[r-1,t] \ge A_e[r-1,t] + \left\lfloor \frac{(A_e[r-1,t] + 2t)}{2t+1} \right\rfloor$$

Example 6.3 Let $A_e[4,1] = \{1111, 1110, 1100, 1001, 0011, 0010, 0000\}$, which has 7 elements. Then $A_e[5,1] \ge A_e[4,1] + (7+2)/3 = 10$, which can be obtained by adding a word at the top and a word at the bottom, and splitting 1001 in {1100, 1001, 0011} to get {11001, 10011, 10010, 00110} as follows.

$$A_e[4,1] = \begin{matrix} 1111 \\ 1110 \\ 1100 \\ 1001 \\ 0011 \\ 0010 \\ 0000 \end{matrix} \qquad A_e[5,1] = \begin{matrix} \mathbf{11111} \\ \mathbf{11110} \\ \mathbf{11100} \\ \mathbf{11001} \\ \mathbf{10011} \\ \mathbf{10010} \\ \mathbf{00110} \\ \mathbf{00101} \\ \mathbf{00001} \\ \mathbf{00000} \end{matrix}$$

□

Method 2: This method is a modification to the one given in [Blu 89]. Let Asy[r',t] = { a_1, a_2, ..., a_k } be an t-asymmetric error correcting code of length r', where $W(a_i) \geq W(a_j)$ for $i \leq j$. And let $A_e[r'',t]$ = { s_1, s_2, ..., s_m }. Then { $<a_1,s_1>$,..$<a_1,s_m>$, $<a_2,s_1>$, ...$<a_2,s_m>$, ..., $<a_k,s_1>$, ...$<a_k,s_m>$ } is an $A_e[r'+r'',t]$ code of (k×m) words. So in order to design $A_e[r,t]$ one can choose the maximum over r' of Asy[r',t] * $A_e[r-r',t]$. This can be improved by only appending $A_e[r-r',t]$ to the middle Asy[r',t] - 4 words. Instead of getting 2 $A_e[r-r',t]$ for the last two words we can obtain $A_e[t+1+r',t]$ words by fixing r'-t-1 bits to be zeros. And similarly, for the first two words, we can obtain $A_e[t+1+r',t]$ instead of 2 $A_e[r-r',t]$ words by fixing r'-t-1 bits to be ones. The size of $A_e[r,t]$ will be (Asy[r',t] - 4) * $A_e[r-r',t]$ + 2 $A_e[t-1+r',t]$. So,

$A_e[r,t]$ = the maximum over r' of (Asy[r',t]-4) * $A_e[r-r',t]$ + 2 $A_e[t+1+r',t]$.

Example 6.4: Let C be the asymmetric distance 3 code of length 9 and 12 codewords [Web 88]. An $A_e[10,2]$ of length 24 can be produced by appending a 1 and 0 to every codeword of C. However, we can produce 26 words as follows. Append 1 and a 0 to the middle 8 words in C yielding 16 words. Without loss of generality, let the first two words of C be 11111 1111 and 11111 1000; and the last two words be 0000 00111 and 0000 00000. Then,

Instead of 11111 11111, 11111 11110, 11111 10001, 11111 10000 we can get 11111 1**1111**, 11111 1**1110**, 11111 1**1100**, 11111 1**1000**, 11111 1**0000** and

instead of 00000 00001, 00000 01110, 00000 00001, 00000 00000 we can get 00000 1**1111**, 00000 0**1110**, 00000 0**1100**, 00000 0**1000**, 00000 0**0000**

Getting a total of 16 + 5 + 5 = 26 words. Notice that an A[11,2] of length 48 can be constructed by appending a 1 and a 0 to each word in this A_e[10,2].

□

The A_e[r,t] is always chosen to be the longer sequence obtained by the two methods. Table 6.8 lists some A_e-sequences with an indication of the method used to produce them.

r	$A_e[r,1]$	$A_e[r,2]$	$A_e[r,3]$	$A_e[r,4]$
1	2	2	2	2
2	3	3	3	3
3	4	4	4	4
4	7^3	5	5	5
5	10	6	6	6
6	16^*	10^3	7	7
7	24^2	12	8	8
8	38^*	16^*	13^3	9
9	72^2	20^2	15^1	10
10	124^2	26^2	18^1	16^3
11	216^2	38^2	21^1	18^1
12	348^2	62^2	26^3	20^1
13	632^2	110^2	30^1	23^1
14	:	198^2	40^2	26^1
15		374^2	64^2	32^2
16		558^2	92^2	36^2
17		798^2	136^2	40^2
18		:	248^2	58^2
19			472^2	82^2
20			904^2	112^2
21			:	146^2
22				186^2

Table 6.8 Sizes of the new $A_e[r,t]$ sequences..

(1: method 1, 2: method 2, 3: from [Mon], and *: given below.)

The $A_e[8,1]$ of size 38 can be obtained by adding 2 words to the 1 asymmetric error correcting code of size 36. One word is added between the first word and the second, and the other is added between the next to last and the last codewords. The $A_e[6,1]$ and $A_e[8,2]$ sequences of size 16 are

$A_e[6,1]$	$A_e[8,2]$
111111	11111111
111110	11111110
111100	11111100
111001	11011010
110011	10110101
100111	01101011
001111	11100001
101010	11100000
010101	01110000
110000	00111000
011000	00011100
001100	00001110
000110	00000111
000011	00000011
000001	00000001
000000	00000000

r	A[r,1]	A[r,2]	A[r,3]	A[r,4]
1	2	2	2	2
2	4	4	4	4
3	6	6	6	6
4	8	8	8	8
5	14	10	10	10
6	20	12	12	12
7	32	20	14	14
8	48	24	16	16
9	76	32	26	18
10	144	40	30	20
11	248	52	36	32
12	432	76	42	36
13	696	124	52	40
14	:	220	60	46
15		396	80	52
16		748	128	64
17		:	184	72
18			272	80
19			496	116
20			944	164

Table 6.9 Sizes of the new A[r,t] sequences.

obtained from Table 6.8 and the relation $A[r,t] \geq 2\, A_e[r-1,t]$.

6.5.2 Design of U-sequences

Recall that in the U-sequence $U[r,t,d] = \{s_0, s_1, ..., s_{m-1}\}$, where each s_i is of length r, either

1) $N(s_i,s_{i+j}) \geq \min(\lceil j/2 \rceil, t+1)$ or

2) $D(s_i,s_{i+j}) \geq \min(d-t-1+odd(j), (t+1+d)-j)$

where $1 \leq i,j \leq m$ and $i+j$ in s_{i+j} is computed modulo m.

Let $Z[r',t] = \{a_1, a_2, ..., a_k\}$ be a t-AEC/AAED (or t-EC/AUED) code of length r', i.e. $N(X,Y) \geq t+1$ for all $X, Y \in Z[r',t]$. And let $A[r-r',t] = \{s_1, s_2, ..., s_m\}$. Then $\{<a_1,s_1>,..<a_1,s_m>, <a_2,s_1>, ...<a_2,s_m>, ..., <a_k,s_1>, ...<a_k,s_m>\}$ is a U[r,t,d] sequence of (k×m) words. In this U[r,t,d] sequence, $d = (Z[r',t]-1) * A[r-r',t] + (r-r') - t$ since the first two words with $N(X,Y) \leq t$ will be $(Z[r',t]-1) * A[r-r',t] + 1$ apart, and they will have an extra distance of (r-r').

The maximum, over r', of $d = (Z[r',t] - 1) * A[r-r',t] + (r-r') - t$ can be used to construct a U[r,t,d] sequence. Table 6.7 summarizes the values of d of these sequences for some values of r and t.

Example 6.5 Suppose we wish to construct a U[7,1,8] sequence. Let r' = 4; then Z[r',t] = {1100, 0011} of size 2. Since r-r' = 3, we will use the A[3,1] = {111, 110, 101, 100, 001, 000} of 6 elements. Then the following sequence is a U[7,1,8] of 12 elements.

$$U[7,1,8] = \begin{matrix} 1100\ 111 \\ 1100\ 110 \\ 1100\ 101 \\ 1100\ 100 \\ 1100\ 001 \\ 1100\ 000 \\ 0011\ 111 \\ 0011\ 110 \\ 0011\ 101 \\ 0011\ 100 \\ 0011\ 001 \\ 0011\ 000 \end{matrix}$$

□

6.5.3 Design of D-sequences

Recall that in the D-sequences $D[r,t,d] = \{s_0, s_1, \ldots s_{m-1}\}$, where each s_i is of length r, either

1) $N(s_i,s_{i+j}) \geq \min(\lceil j/2 \rceil, t+1)$ or

2) $N(s_{i+j},s_i) \geq (d+1) - \max(j, j+\lceil (2t+1-j)/2 \rceil)$

where $1 \leq i,j \leq m$ and $i+j$ in s_{i+j} is computed modulo m.

These sequences can be obtained as follows:

1- The sequence $\{1^{(r-1)-i}\, 0^i\, 1,\ 1^{(r-1)-i}\, 0^i\, 0 \mid 0 \leq i \leq r-1\}$ is a D[r,t,t+r], and

2- Any U[r,t,d] sequence is at least a D[r,t,d] sequence. In particular, the U[r,t,d] sequences constructed in the last section are D[r,t,d+t] sequences.

Example 6.6 The U[7,1,8] of 12 elements in the previous example is a D[7,1,9] sequence.

□

Table 6.1 lists values of d of some D[r,t,d] sequences for different values of r and t.

Chapter 7

Conclusion

7.1 Summary

The main results of this work are the designs of balanced codes that can correct t errors for $0 \leq t \leq 4$. When $t = 0$ (i.e. no error correction) the designed codes have higher information rates than the equivalent codes and yet maintain the same encoding/decoding complexity. The t-EC/B, for $1 \leq t \leq 4$ (in many cases) give higher information rate than the existing codes; and (in all cases) have simpler and faster encoders and decoders.

DC-free line codes were presented in which codewords are not necessarily balanced. The new codes have the same information rate and encoding/decoding complexities as existing codes, but have other desirable properties such as high transition densities and shorter run lengths than the equivalent codes given in [Den 88]. The techniques used to design such codes has lead to an efficient design of t-EC/AUED coset codes. These codes considerably improve upon the codes given recently in [Bru 89].

We have seen that a balanced code that corrects t errors (t-EC/B) can also be viewed as a t-EC/AUED code. It is noticed that the t-EC/B codes designed in Chapter 3 give higher information rate than the (coset) t-EC/AUED codes. In spite of their dc-null properties, balanced codes obtain higher information rates for $1 \leq t \leq 3$. It should be noted that also, in some cases, the 4-EC/AUED coset codes may have higher information rates than the 4-EC/B codes.

The balanced codes were generalized in three ways: (1) Design of balanced codes with low dc level; these codes are designed based on the combined techniques of balanced codes and dc-free coset codes. They use one extra check bit but have a significantly lower dc-level and higher transitions density. These codes are much more attractive for optical transmission than the bare bone balanced codes. (2) Design of balanced codes over a non-binary alphabet.. And (3) Design of "semi-balanced" codes in which the number of 1's and 0's in every code word differs by at most a certain value, say m.

7.2 Future Research

It was shown that the balanced codes given in Chapter 2 are optimal up to the complementation method used. The designed codes have a redundancy of about log k, see (2-5), and the number of check bits in a completely non-systematic code is at least 0.5 log k, see (1-1). Thus the codes can be further improved but new designs seem to be hard to obtain. The other direction there is to find some other simple function (like complementation which has efficient encoders and decoders) that may further reduce the redundancy.

The problems become more interesting in the design of t-EC/B codes (for $1 \leq t \leq 4$) given in Chapter 3. We have seen that the t-EC/B codes for $2 \leq t \leq 4$ are based on the 1-EC/B codes. The 1-EC/B code becomes the central issue. In the design of such codes, algebraic structures, namely Abelian groups, fields, or binary fields, were used. After choosing a structure, the design becomes extremely sensitive to a choice of a subset H of r elements from this structure. The subsets (H) used in the codes constructed in Chapter 3 were obtained mostly by trail-and-error or by heuristic methods. The question is: Given an algebraic structure can we *algorithmically* find the best subset H that maximizes the information rate of the code? The other question is which Abelian group of size n should be chosen? It is only observed that Z_n (the additive cyclic group of order n) gives the best

results. (This selection problem does not occur with fields since there is a unique field structure of size p^m for a prime p.

The dc-free line codes given in Chapter 4 are based primarily on increasing the number of transitions in each codeword from 1 to n/2 using one extra check bit. When two or more check bits are used, it is still not clear how to increase the transitions more than n/2. Similarly, the balanced codes with k/2 - 1 transitions designed in Section 5.1 use an extra check bit in order to reduce the disparity to almost half the original disparity. Again, using 2 or more extra check bits, can we further reduce the maximum disparity (or increase the number of transitions)? For example, using 2 extra check bits, is it possible to reduce the maximum disparity to 1/4 (or even 1/3) the original disparity?

Similar problems arise in Chapter 6. In the design of the t-EC/AUED coset codes, the weight distribution of the underlying code of length n was the main factor. We have seen that using an extra check bit, the weight distribution of a linear code can be reduced to n/2; and using two check bits the weight distribution can be reduced to n/4. Of course, the question is: Can we reduce the weight distribution even further by using 3 or more extra check bits? For instance, can we reduce the weight distribution to n/8 (or even to n/6) using 3 or more check bits?

The solution to such questions will have a dramatic effect on the codes designed in Chapters 4 and 6. Most of these codes will have even higher information rate if such solutions exist.

Bibliography

[Alb 89a] S. Al-Bassam and B. Bose, "Design of efficient balanced codes," Proceeding of the 19th Int. Symp. on Fault Tolerant Computing, June 1989 (to appear in IEEE Trans. on Computers).

[Alb 89b] S. Al-Bassam and B. Bose, "Asymmetric/Unidirectional Error Correcting and Detecting Codes," in the 7th conference of Applied Algebra and Error Correcting Codes, France, June 1989 (to appear in IEEE Trans. on Computers).

[Alb 89c] S. Al-Bassam, B. Bose, and R. Rowley, "Transitions DC-Free Coset Codes," AAECC 7, Toulouse, France, June 1989 (submitted to the special issue of Applied Mathematics on Error Control Codes).

[Alb 90a] S. Al-Bassam and B. Bose, "On Balanced Codes," to appear in Trans. on Information Theory, vol. IT-36, March 1990.

[Alb 90b] S. Al-Bassam and B. Bose, "Design of Efficient Error-Correcting Balanced Codes," submitted to FTCS 20, June 1990.

[Alo 88] N. Alon, et. al, "Balancing Sets of Vectors," IEEE Trans. on Information Theory, vol. IT-34, Jan 1988.

[Ben 86] C. Ben-Zion, *Two issues in Public Key Cryptography*. Cambridge, Ma.: MIT Press, 1986.

[Ber 61] J. M. Berger, "A note on Error Detection Codes for Asymmetric Channels," Information and Control, vol. 4, March 1961.

[Bla 89] M. Blaum and H. van Tilborg, "On t-Error Correcting/All Unidirectional Error Detecting Codes," IEEE Trans. on Computers, vol. C-38, Nov. 1989.

[Bor 82] J. Borden, "Optimal Asymmetric Error Detecting Codes," Information and Control 53, April 1982.

[Bos 82a] B. Bose and D. Pradhan, "Optimal Unidirectional Error Correcting/Detecting Codes," IEEE Trans. on Computers, vol. C-31, June 1982.

[Bos 82b] B. Bose and T. R. N. Rao, "Theory of Unidirectional Error Detecting/ Correcting Codes," IEEE Trans. on Computers, vol. C-31, June 1982.

[Bos 85] B. Bose and D. Lin, "Systematic Unidirectional Error Detecting Codes," IEEE Trans. on Computers, vol. C-34, Nov. 1985.

[Bos 87] B. Bose, "On Unordered Codes," Proceeding of the 17^{th} Int. Symp. on Fault Tolerant Computing, July 1987.

[Bos 90] B. Bose, "Asymmetric Error Correcting Codes," ISIT, Jan. 1990.

[Bro 90] A. Brouwer, et. al., "A New Table of Constant-Weight Codes," to appear IEEE ISIT, Jan. 1990.

[Bru 89] J. Bruck and M. Blaum, "Some New EC/AUED Codes," FTCS-19, June 1989.

[Cal 89] A. Calderbank, M. Herro, and V. Telang, "A Multilevel Approach to the Design of DC-Free Line Codes," Trans. on Information Theory, vol. IT-35, May 1989.

[Cap 89] R. Capocelli, L. Gargano, and U. Vaccaro, "Efficient q-ary Immutable Codes," in the 7^{th} conference of Applied Algebra and Error Correcting Codes, France, June 1989.

[Con 79] S. Constantin and T. Rao, "On the Theory of Binary Asymmetric Error Correcting Codes," Information and Control, vol. 40, 1979.

[Den 88] R. H. Deng and M. A. Herro, "DC-Free Coset Codes," IEEE Trans. Information Theory, vol. IT-34, NO. 4, July 1988.

[Etz 90] T. Etzion, "Constructions of error-correcting dc-free block codes," ISIT, Jan. 1990.

[Fer 84] H. C. Ferreira, "Lower Bounds on the Minimum Hamming Distance Achievable with Run Length Constraint or DC Free Block Codes and the Synthesis of a (16,8) d_{min} = 4," IEEE Trans. on Magn., vol. MAG-34, Sept. 1984.

[Fre 62] C. V. Freiman, "Optimal Error Detection Codes for Completely Asymmetric Binary Channels," Information and Control, vol. 5, March 1962.

[Kaw 88] Satoki Kawanishi, et. al., "DmB1M Code and its Performance in a Very High-Speed Optical Transmission System," IEEE Trans. Commun., vol. COM-36, Aug. 1988.

[Knu 86] D. E. Knuth, "Efficient Balanced Codes," IEEE Trans. on Information Theory, vol. IT-32 Jan. 1986.

[Kun 88] S. Kundu and S. Reddy, "On Systematic t-Error Correcting/All Unidirectional Error Detecting Codes," IEEE ISIT, June 1988. (To appear in IEEE Trans. on Computers.)

[Kun 90] S. Kundu, "Design of Non-Systematic 3syEC/AUED Codes of Asymptotically Optimal Order," ISIT, Jan. 1990.

[Lei 84] E. L. Leiss, "Data Integrity in Digital Optical Disks," IEEE Trans. on Computers, vol. C-33, Sept. 1984.

[Lin 88] D. Lin and B. Bose, "Theory and Design of t-Error Correcting/ d-Unidirectional error Detecting Codes," IEEE Trans. on Computers, vol. C-37, April 1988.

[Mat 88] K. Matsuzawa and E. Fujiwara, "Masking Asymmetric Line Faults using Semi-Distance Codes," NTT Electrical Communication Laboratories, Tokyo, Japan, 1988.

[Mce 85] R. McEliece, "The Reliability of Computer Memories," Scientific America, vol. 252, Jan. 1985.

[Mon] B. Montgomery and B. Kumar, "Systematic Random Error Correcting and All Unidirectional Error Detecting Codes," to appear.

[Mor 83] D. Morris, *Pulse Code Formats for Fiber Optical Data Communication.* NY: Marcel Decker, 1983.

[Nik 86] D. Nikolos, et. al., "Systematic t-Error Correcting/All Unidirectional Error Detecting Codes," IEEE Trans. on Computers, vol. C-37, May 1986.

[Pet 72] W. Peterson and E. Weldon, *Error Correcting Codes.* Cambridge, Ma.: MIT Press, 1972.

[Pra 77] D. Pradhan and S. Reddy, "Fault Tolerant Fail-safe Logic Network," in Proc. IEEE COMPCON, March 1977.

[Pra 80a] D. K. Pradhan, "A New Class of Error Correcting/Detecting Codes for Fault Tolerant Computer Applications," IEEE Trans. on Computers, vol. C-29, June 1980.

[Pra 80b] D. Pradhan and J. Stiffler, "Error-Correcting Codes and Self-Checking Circuits," IEEE Computer, vol. 13, March 1980.

[Pra 86] D. K. Pradhan, *Fault Tolerant Computing: Theory and Techniques.* Englewood cliffs, NJ: Prentice Hall, 1986.

[Rio 68] J. Riordan, *Combinatorial Identities.* New York: Wiley, 1968.

[Sai 88] Y. Saitoh, et. al., "A Method to Construct Constant-Weight t-Error Correcting Codes," in the 11^{th} Sym. on Information Theory and its Applications, Beppu, Japan, Dec. 1988.

[Spe 28] E. Sperner, "Ein Satz über Untermengen einer endlichen Menge," Math. Zeitschrift, vol. 27, 1928.

[Tak 76] Y. Takasaki, et. al., "Optical Pulse Formats for Fiber Optic Digital Communications," IEEE Trans. on Communication, COM-24, April 1976.

[Tao 88] D. Tao, et. al., "An Efficient Class of Unidirectional Error Detecting/ Correcting Codes," IEEE Trans. on Computers, vol. C-39, July 1988.

[Til 89] H. van Tilborg and M. Blaum, "On Error-Correcting Balanced Codes," IEEE Trans. on Information Theory, vol. IT-35, Sept. 1989.

[Toh 71] Y. Tohma, R. Sakai, and R. Ohyama, "Realization of Fail-Safe Sequential Machines by Using k-out-of-n Code," IEEE Trans. on Computers, vol. C-20, Nov. 1971.

[Uye 88] T. Uyematsu, et. al., "Effect of Asymmetric Error Correcting Codes in Photon Communication Systems," IEEE ISIT, June 1988.

[Var 73] R. Varshamov, "A Class of Codes for Asymmetric Channels and a Problem from the Additive Theory of Numbers," IRE Trans. on Information Theory, vol. IT-19, Jan 1973.

[Web 88] J. Weber, et. al., "Bounds and Constructions for Binary Codes of Length less than 24 and Asymmetric Distance less than 6," IEEE Trans. on Information Theory, IT-34, Sept. 1988.

[Wid 83] A. X. Widmer and P. A. Franaszek, "A DC-Balanced, Partioned-Block, 8B/10B Transmission Code," IBM J. Res. Develop., vol. 27, Sept. 1983.

[Yos 84] N. Yoshikai, et. al., "mB1C Code and its Performance in an Optical Communication System," IEEE Trans. Commun., vol. COM-32, Aug. 1984.