RECURSIVE CONSTRUCTION FOR ERROR CORRECTING CONSTANT WEIGHT CODES

Gowri Ramanathan
Department of Computer Science
Oregon State University
Corvallis, OR 97331

e-mail: ramanag@mist.cs.orst.edu

A research paper
submitted in partial fulfillment of
the requirements for the degree of
Master of Science

Major Professor:	Dr. Bella Bose
Minor Professor:	Dr. Curtis Cook
Committee Member:	Dr. Vikram Saletore

January 9, 1992

Abstract

In this project we have designed a new construction method for t error correcting constant weight codes. This method is an improvement over the existing codes in terms of information rate. The construction method is recursive as it is based on the observation that $2t$ error correcting code can be built by concatenating two t error correcting codes.This results in reduction of code word length for higher t values.

TABLE OF CONTENTS

	Page
Abstract	i
Table of Contents	ii
Notations	iii
Acknowledgements	iv
1. Introduction	1
2. Preliminaries and Previous Construction Methods	3
2.1. Preliminaries, Terms and their definitions	3
2.2. 1-EC/AUED Code Construction	5
2.3. t-error Correcting Codes for t=2,3, and 4	5
3. Lin's Constant Weight Code Construction	8
4. Recursive Code Construction	11
4.1. Construction Methodology	11
4.2. Encoding Procedure	13
4.3. Decoding Algorithm for Recursive Code	15
5. Comparison of Recursive Code with Lin's Code	18
6. Conclusion	19
Reference	20

Notations

k - length of the datapart

w - wight of the datapart

V - set of k-tuples of weight w

v - information symbol, $v \in V$, $v = (v_1, v_2, ..., v_k)$

n - length of the code word

t - number of errors corrected

l - weight of the code word

G - abelian group with elements $(g_1, g_2, ...)$, g_1 is the identity element

F - Galois field $GF(p^{\mu})$ where p is a prime.

α - primitive element of F

d(x,y) - hamming distance between the vectors x and y

U - set of all q out of p vectors.

$\Phi(X)$ - Φ is a one-to-one mapping which maps the elements of the set X into a set of constant weight vectors

Ψ - is any one-to-one mapping defined between $GF(2^{\mu})$ and U, where $|U| \geq 2^{\mu}-1$

Acknowledgements

In 1988,When I took the course *'Introduction to Coding Theory'* offered by Dr. Bella Bose, it has changed my professional life forever from being a pure mathematician to a computer scientist. I am very grateful to Dr. Bose for having helped me to make this decision with ease. Apart from positively influencing me in seeing the avenues for applications of Mathematics in computer science, Dr. Bose has been a constant source of encouragement as a supervisor to my Master's degree project, for which I owe my sincere thanks to him.

I wish to extend my thanks to Dr. Curtis Cook and Dr. Vikram Saletore for consenting to be in my master's degree project committee. I appreciate all the inputs given by my committee members during the development of this project.

The graduate courses offered by various faculty members of the computer science department had helped me immensely to take my first few steps as a toddler into the field of computer science. I am thankful to our faculty members for the pleasure of learning, they had given me in their classes.

I wish to express my heart-felt thanks to the computer science department chairman, Dr. Walter Rudd and the director of Educational Opportunity Program, Dr. Larry Griggs for giving me the financial support during my Master's program in computer science. My special thanks to Dr. Atta Akyeampong, Educational Opportunity Program, for being very understanding of my financial needs.

I wish to thank my husband, Ramesh Krishnamurthy, for supporting me in various aspects and phases of this project. Also I am deeply indebted to him for helping me to realize my moral obligations to serve the world as a scientist. He had definitely added a social dimension not only to this project but also to my future endeavors.

Finally, I must thank my parents, who have taught me multitude of basic things, all of which were helpful during this project.

1. Introduction:

In a constant weight code each code word contains a fixed number of ones. The constant weight codes, in particular the balanced codes where each code word contains equal number of ones and zeros, have many applications.

- Balanced codes are most suited for fiber optical communications [11, 12, 13, 15, 22, 27, 29, 30]. As the associated electronic circuitry in the fiber optical communication responds poorly when switching at high speed, the whole transmission process has to be AC-coupled. This demands a code which will eliminate the dc component and the balanced codes have this characteristic.
- Recently, the balanced codes are proposed for noise reduction in VLSI chips[26]. Because of the presence of parasitic inductance (L) of the power supply package lead and $V = L\frac{di}{dt}$, the current fluctuations cause a voltage drop between the chip and the circuit board. This can force the designer to design the chip to operate at a lower speed. Using balanced codes, the voltage drop due to $\frac{di}{dt}$ can be made zero.
- Balanced codes are perfect asymmetric/unidirectional error detecting codes [4, 7, 10, 14]. Also these codes are optimal in the sense that no perfect asymmetric/unidirectional code will have more code words than the balanced code for a given length.
- One other application of this code is in encoding the states of a sequential machines of fault-tolerant sequential circuits [24, 28].
- For data integrity purposes these codes are used in storing data in CD ROM [16, 19]. Any inadvertent $0 \rightarrow 1$ changes in the data stored in these write once memories can be detected by using balanced codes.
- They are used in the public key crypto system proposed by Ben-Zion [3].
- The error correcting balanced codes are used in fault masking in bus lines of VLSI systems [21]. The big advantage of fault masking using balanced codes is that it has no additional circuits except for bus terminal gates.
- Balanced codes are used in the design of t Error Correcting and All Unidirectional Error Detecting(t-EC/AUED) codes [8, 17, 18, 20, 23, 25].
- Recently, in [5], the authors have shown how these codes can be used in transferring data in asynchronous systems with no acknowledgement signals.

Designing a constant weight code with t error correcting capability has been open for several years. The earlier construction methods proposed for $t = 2,3$ and 4 are presented in the second chapter of this report. Wide spreading fiber optical transmission media demands

a constant weight code with t error correcting capability where t is an aribitrary finite integer. Until Lin's construction[20], designing$t$ error correcting constant weight codes was known.

The goal of this research project is to construct a systematic t-error correcting(t-EC) constant weight codes. The construction method proposed in this report is an improvement over the method suggested by Lin[20]. Though the design given in this report will build t-EC balanced codes it can very well be extended to construct t-EC constant weight codes. Futher, research work in analyzing the optimality of this code is pending.

In this paper we assume that the data part is a constant weight vector, in particular a balanced vector, as was in [8, 17, 18, 25]. (The conversion from information part to this balanced code can be done using some of the recently developed efficient schemes [1, 2, 16]). Appropriate checks are appended to this constant weight vectors to get error correcting constant weight code. In that sense these are called quasi-systematic codes. We present a construction method which has better information rate than the existing codes for higher t values (i.e. $t > 4$).

The paper is organized as follows. In section 2, the preliminaries and previous construction methods for t=1,2,3 &4 are briefly reviewed. We present our recursive construction in section 4 following a short introduction of Lin's codes in section 3. Finally in section 5 we present the comparison between Lin's code, and the recursive code introduced in this paper.

2. Preliminaries and Previous Construction Methods

2.1 Preliminaries, Terms and their definitions

The message vector that has to be communicated is called *information symbol*. We append some redundant bits to the information symbol called *check symbol* to obtain the required distance among the encoded words. The code in which the check symbol and information symbol are separable in each encoded word, is known as *systematic code*. Further, in the systematic codes, the information symbol of the code word is unmodified. If the information symbols are modified during the encoding procedure, those codes are known as *quazi-systematic* codes.

The number of bits in the code word is referred to as the *length of the code word*. For example, in u = (001100), the length of the code word is six. There are several metrics to measure the distance between two vectors, one of the one most commonly used measures in coding theory is *Hamming distance* between vectors. Hamming distance between two vectors u and v, denoted as d(u.v), is defined as the number of positions of u and v in which their bit values differ. For example u = (0101), v = (1100) then d(u,v) = 2.

The information symbol is encoded into some code word before it is sent on the communication channel. Because of the channel noise, the received word may be different from the word sent. The number of different coordinates in these two words is called the number of errors made during transmission. The code built with t errors correcting capability will enable the decoder to get the message sent from the received word even in the presence of up to t errors in the received word. Data sent over the channel may suffer different kinds of errors.

A widely used channel is the binary symmetric channel in which $1\rightarrow0$ and $0\rightarrow1$ errors occur with equal probability. In the asymmetric channel the probability of a $0\rightarrow1$ error is lower than the probability of a $1\rightarrow0$ error. In optical disk storage media, a 0 can be changed to 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 $1\rightarrow0$ error occurs with much higher probability than a $0\rightarrow1$ error. The errors that occur in the symmetric channel are known as *symmetric errors*, whereas the errors in asymmetric channel are called *asymmetric errors*. There is one other kind of errors called *unidirectional errors*. As the name suggests, in this kind either 1's become 0 or 0's become 1. But either 1 will become

0 or 0 will become 1 is not known apriori, hence the unidirectional error correcting/detecting code design differs from that of symmetric error correcting/detecting codes. In the VLSI chips, when the gates malfunction, either stuck-at-one or stuck-at-zero condition is created. In either situation either all 1's sent will be changed to 0's or all 0's sent will be changed to 1's, giving rise to unidirectional errors.

Hamming, Bell System scientist, had designed the single symmetric error correcting and detecting codes in 1950. *BCH*(Bose-Chaudhuri-Hocquenghem) codes are a generalization of the Hamming codes to correct multiple errors. BCH codes are cyclic, and thus the code words are generated by a generating polynomial. According to the BCH bound[31], to design t error correcting code the generating polynomial has to have $2t$ roots, which are consecutive powers of the primitive element of the underlying finite field. This characteristic of the BCH code is used in the construction method presented in this report.

Example 2.1: To construct a 3-EC BCH code over the binary alphabet, the generating polynomial g(X) has to have six roots which are consecutive powers of the primitive element α. Suppose, these roots are α, α^2, α^3, α^4, α^5, and α^6, then

$$g(X) = X^{10} + X^8 + X^5 + X^4 + X^2 + X + 1$$

g(X) is the greatest common divisor of the minimum polynomials of α, α^3, and α^5. This ensures g(X) has α, α^2, α^3, α^4, α^5, and α^6 as its roots. In this case, α is a primitive element of the galois field GF(2^4). The code word length is 15, information bits can be upto 5 and the check bits are 10.

The number of 1's in a code word stands for its weight. The code in which all code words have the same weight is a *constant weight code*. A *balanced code* is a constant weight code in which the weight of the code word is half of its length. We have seen the wide applications of balanced codes in the previous chapter.

The code which can correct up to t symmetric errors and detect all unidirectional errors is denoted as *t-EC/AUED*. The following theorem [8, 9] states the necessary and sufficient condition for a code to be t-EC/AUED.

Theorem 2.1:

A code C is capable of correcting t symmetric errors and of detecting all t+1 or more unidirectional errors iff the code C satisfies the following condition:

$$\text{for all } X, Y \in C,\ N(X,Y) \geq t+1.$$

Here N(X,Y) is the number of $1 \rightarrow 0$ crossover from X to Y.

When this result is applied to a constant weight code C:

C is capable of correcting t symmetric errors iff for all $X, Y \in C$, $d(X,Y) \geq 2t+2$.

In the remainder of this section we briefly explain the known methods of t-error correcting constant weight codes for t = 1,2,3 and 4. The concepts adopted in these code constructions are useful in designing codes for $t > 4$.

2.2. 1-EC/AUED Code Construction[8]:

Let G be an abelian group of size k and V be the set of words with constant weight w and length k. Define a mapping

$$T: V \rightarrow G$$

$$\text{where } T(v) = \sum_{i=1}^{k} v_i g_i$$

$v = (v_1, v_2, \ldots, v_k) \in V$ is any arbitrary vector and

$$v_i g_i = \begin{cases} g_1 \text{ (identity element of G) for } v_i = 0 \\ g_i \text{ for } v_i = 1 \end{cases}$$

Let U be the set of $\lfloor \frac{p}{2} \rfloor$-out-of-p vectors where p is the smallest integer such that $\binom{p}{\lfloor p/2 \rfloor} \geq |G|$. Thus the check symbol of v is given by $\Phi(T(v))$ Note that Φ is a one-to-one mapping from G to U.

Example 2.2: The design of 1-EC codes with 6 information bits is as follows:
The 2^6 information symbols can be represented by a 4-out-of-8 code. Note that a 4-out-of-8 code has 70 words but only 64 need to be used. Now choose $G = Z_8$ to build the check symbol, and choose p = 5 as $\binom{p}{\lfloor \frac{p}{2} \rfloor} \geq 8$. The length of the code word is 8+5 = 13.

2.3. *t*-error correcting codes [17,18,25] for t=2,3 and 4:

The following construction in the case of t = 2 was proposed independently by both Kundu and Reddy[18] and Saitoh et. al.[25].

Let F be a Galois field of order at least $k+1$ elements, and the elements of F be $\{f_0=0, f_1=1, f_2, \ldots, f_k\}$. Any code word of 2-EC/AUED under this construction has three parts:

$$\text{datapart } v, \text{ two check symbols } \Phi(T_1(v)) \text{ and } \Phi(T_2(v))$$

$$\text{where } T_1(v) = \sum_{i=1}^{k} f_i v_i \text{ and } T_2(v) = \prod_{i=1}^{k} f_i^{v_i}.$$

Here the summation and multiplication operations are field addition and field multiplication. Further we assume Φ is a one-to-one mapping from the Galois field to the set of $\lfloor \frac{p}{2} \rfloor$-out-of-p vectors U, where p is the smallest integer such that $\binom{p}{\lfloor \frac{p}{2} \rfloor} \geq |F|$.

Three error correcting codes were independently given by Kundu[17] and Saitoh et. al.[25]. Any code word in this construction has 4 parts: datapart v, three check symbols $\Phi(T_1(v))$, $\Phi(T_2(v))$ and $\Phi(T_3(v))$. $T_1(v)$ and $T_2(v)$ are defined as above whereas $T3(v) = \sum_{i=1}^{k} (\frac{v_i}{f_i})$. Here the field should be $GF(2^\mu)$ and $2^\mu > k$.

Example 2.3: The design of 2 and 3-EC codes with 6 information bits is as follows:
As discussed in example 1.1 we need to map the 2^6 information symbols into 4-out-of-8 code. The smallest field F with $|F| > 8$ is $GF(3^2)$. Here we have to choose p as 5 and so $\Phi(T_1(v))$ and $\Phi(T_2(v))$ are 5-tuples. The code word length is 8+5+5 = 18.

In the case of 3-EC, the smallest $GF(2^\mu)$ field with $2^\mu > 8$ is $GF(2^4)$. These 16 elements can be represented by a 3-out-of-6 code, i.e. p = 6, so the total code length will be 8+6+6+6 = 26.

Example 2.4: The design of 4-EC codes with 6 information bits is as follows:

As discussed in example 1.1 and 1.2 we need to map the 2^6 information symbols into 4-out-of-8 code and need GF(2^4) to generate the check symbols. The 16 field elements can be represented by a 3-out-of-6 code, i.e. p = 6, so the total code length will be 8+6+6+6+6+6 = 26.

3. Lin's constant weight code construction

In [20] Lin has described methods for designing quasi-systematic t-error correcting constant weight codes for any t. For $t = 2$ and 3, his codes use the same number of check bits as that of [17,18,25]. For $t = 4$ his method is better than [25]. Since our design methods are improvements over his, we first briefly explain his methods in this section.

First we describe his codes for $t = 5$ and then for any t. Let F be a Galois field of 2^μ elements, where $2^\mu - 1 \geq k$, and α be a primitive element in F. Any code word in this construction has the following form:

$$\psi(Q_{-1}(v)) \mid \Phi(P_{-3}(v)) \mid \Phi(P_{-1}(v)) \mid v \mid \Phi(P_1(v)) \mid \Phi(P_3(v)) \mid \psi(Q_1(v))$$

where $P_j = \sum_{i=1}^{k} v_i(\alpha^j)^i$ for j = -3,-1,1,3 and v is the data part.

Φ is a one-to-one mapping from the Galois field elements to set of $\lfloor \frac{p}{2} \rfloor$-out-of-p vectors U, where p is the smallest integer such that $\binom{p}{\lfloor \frac{p}{2} \rfloor} \geq |F|$.

Let $P_1(v)$ be $(p_1, p_2, \ldots, p_\mu)$ and $P_{-1}(v)$ be $(p_{-1}, p_{-2}, \ldots, p_{-\mu})$

Let $G = \{g_1 = 0, g_2, \ldots g_\mu\}$ be an abelian group of μ elements, then define

$$Q_1(v) = \sum_{i=1}^{\mu} p_i.g_i \text{ where } p_i.g_i = g_1 \text{ when } p_i = 0$$

$$p_i.g_i = g_i \text{ when } p_i = 1$$

$$Q_{-1}(v) = \sum_{i=1}^{\mu} p_{-i}.g_i \text{ where } \quad p_{-i}.g_i = g_1 \text{ when } p_{-i} = 0$$

$$p_{-i}.g_i = g_i \text{ when } p_{-i} = 1$$

ψ is a one-to-one mapping from the elements of group G to the set X of $\lfloor \frac{q}{2} \rfloor$-out-of-q vectors, where q is the smallest integer such that $\binom{q}{\lfloor \frac{q}{2} \rfloor} \geq |G|$.

In this way of designing we can notice that for any two different data vectors v and v′,

$$d(\psi(Q_{-1}(v))\Phi(P_{-1}(v)), \psi(Q_{-1}(v'))\Phi(P_{-1}(v'))) \geq 4 \text{ when } P_{-1}(v) \neq P_{-1}(v').$$

Similarly

$$d(\psi(Q_1(v))\Phi(P_1(v)),\psi(Q_1(v'))\Phi(P_1(v'))) \geq 4 \text{ when } P_1(v) \neq P_1(v').$$

Whereas when $P_{-1}(v) = P_{-1}(v')$ or $P_1(v) = P_1(v')$ we have $d(v,v') \geq 4$ by BCH bounds, and when $P_{-1}(v) = P_{-1}(v')$ and $P_1(v) = P_1(v')$ we get $d(v,v') \geq 10$ using BCH bounds.

In all the cases mentioned above, together with the balanced check symbols $\Phi(P_{-3}(v))$ and $\Phi(P_3(v))$, we get the minimum distance of 10 between any two encoded words. We can make the observation that the check symbols involving the lower powers of the primitive element have to be augmented with suitable vectors to have higher minimum distance among them. We can look at this construction as concatenation of two code words from 4-distant and 2-distant codes to the datapart. This observation gives a natural extension of this code construction to general *t* EC code.

In Lin's construction of distant 4t+2 code any code word has the form

$$c = w_1|w_2|w_3|...|w_t|v|v_t|...|v_3|v_2|v_1$$

Here v is the data vector of length *k* and the rest of the vectors in the code words are check symbols. By choosing a suitable Galois field F, define

$$X = P_1(v) \mid P_3(v) \mid .... P_{2t-1}(v) \text{ and } \Phi(X) = v_t$$

$$Y = P_{-1}(v) \mid P_{-3}(v) \mid .... P_{-(2t-1)}(v) \text{ and } \Phi(Y) = w_t$$

where Φ and $P_j(v)$'s are as defined in the above construction extending for j = -(2t-1),..., -3, -1, 1, 3,..., 2t-1.

To construct the check symbols $v_{t-1},v_{t-2},...v_1$ and $w_{t-1},w_{t-2},...w_1$, X and Y values are computed in a similar manner by taking the check symbol computed in the previous step in the place of data vector *v*. For the sake of simplicity, only positive powers of the primitive elements are considered for computing Y values while building the check symbols $w_{t-1},w_{t-2},...w_1$.

The general design principle in this construction can be explained as follows: Each code word has a data part and two check parts each of which is a code word from a constant weight code of minimum distance $2t$. The construction of check parts is done in $2t$ steps, in each step appending a constant weight vector and thus reducing the minimum distance requirement by 2.

4. Recursive Code Construction

4.1 Construction Methodology

Suppose we want to construct a quasi-systematic constant weight code C with the minimum distance 2t+2 (where $t \geq 1$). Each code word c in C will have the form L|v|R where v is the datapart and R and L are code words from constant weight codes with minimum distance of $2\lceil \frac{t}{2} \rceil$ and $2\lfloor \frac{t}{2} \rfloor$ respectively. By having well defined data parts for R and L we can recursively construct the check parts R and L. We can look at these check parts in another way as follows: we require the check part R to be a code word from a code of minimum distance $2m + 2$ where $m = \lceil \frac{t}{2} \rceil - 1$ and $m \geq 1$. Now we have m in the place of t in the original code construction. We can construct R as $R_L|v_R|R_R$ where R_R is a code word from a constant weight code with a minimum distance of $2\lceil \frac{m}{2} \rceil$ and R_L is a constant weight code with a minimum distance of $2\lfloor \frac{m}{2} \rfloor$ and v_R is the datapart of R.

Now we formally define the main idea for our construction method. Let V be the set of all constant weight vectors of length k. Let $s_1 = \lceil \frac{t}{2} \rceil$ and $s_2 = \lfloor \frac{t}{2} \rfloor$,
$v = (v_1, v_2, \ldots, v_k) \in V$. Let $F = \{0, \alpha^0, \alpha^1, \ldots, \alpha^{2^\mu - 1}\}$ be the Galois field with 2^μ elements, α is a primitive element in F. We assume that μ is the smallest integer such that $2^\mu > k$.

$P_R: V \rightarrow \{GF(2^\mu)\}^{s1}$ and $P_L: V \rightarrow \{GF(2^\mu)\}^{s2}$

$$P_R(v) = \begin{bmatrix} P_1(v) \\ P_3(v) \\ \vdots \\ P_{2s_1-1}(v) \end{bmatrix} \qquad P_L(v) = \begin{bmatrix} P_{-1}(v) \\ P_{-3}(v) \\ \vdots \\ P_{-(2s_2-1)}(v) \end{bmatrix}$$

Here $P_i(v) = \sum_{v_j=1} (\alpha^j)^i$ for $i = -(2s_2-1), \ldots -3, -1, 1, 3, \ldots, 2s_1-1$. Note that P_R and P_L map a vector $v \in V$ into s_1 and s_2 tuples over $GF(2^\mu)$. Let U_1 and U_2 be two constant weight codes with minimum distance $2s_1$ and $2s_2$ respectively. Also let $|U_1| \geq 2^{\mu s1}$ and $|U_2| \geq 2^{\mu s2}$. Define two one-to-one function Φ_1 and Φ_2 where

$$\Phi_1 : GF(2^\mu)^{s1} \rightarrow U_1 \text{ and } \Phi_2 : GF(2^\mu)^{s2} \rightarrow U_2.$$

Then $R = \Phi_1(P_R)$ and $L = \Phi_2(P_L)$ and the final encoded word will have the form L|v|R

Example 4.1: Let V be the set of 7-out-of-14 code. Suppose we want to design an 8 error correcting code i.e. distance of the code is 18. From the above design the appropriate Galois field is $GF(2^4)$. If we choose α as the root of $x^4 + x + 1 = 0$ then α generates all non-zero elements in $GF(2^4)$. Suppose $v = (11111110000000) \in V$. Then

$$P_1(v) = \alpha^0 + \alpha^1 + \alpha^2 + \alpha^3 + \alpha^4 + \alpha^5 + \alpha^6 = \alpha^5$$
$$P_3(v) = (\alpha^0)^3 + (\alpha^1)^3 + (\alpha^2)^3 + (\alpha^3)^3 + (\alpha^4)^3 + (\alpha^5)^3 + (\alpha^6)^3 = \alpha^{14}$$
$$P_{-1}(v) = (\alpha^0)^{-1} + (\alpha^1)^{-1} + (\alpha^2)^{-1} + (\alpha^3)^{-1} + (\alpha^4)^{-1} + (\alpha^5)^{-1} + (\alpha^6)^{-1} = \alpha^{14}$$
$$P_{-3}(v) = (\alpha^0)^{-3} + (\alpha^1)^{-3} + (\alpha^2)^{-3} + (\alpha^3)^{-3} + (\alpha^4)^{-3} + (\alpha^5)^{-3} + (\alpha^6)^{-3} = \alpha^{11}$$

Then we have

$$P_R(v) = \begin{bmatrix} \alpha^5 \\ \alpha^{14} \end{bmatrix} \quad \text{and} \quad P_L(v) = \begin{bmatrix} \alpha^{14} \\ \alpha^{11} \end{bmatrix}$$

Here $P_R(v)$ and $P_L(v)$ are 2-tuples over $GF(2^4)$, hence we need U_1 and U_2 to be constant weight codes of minimum distance 8 and have size greater than or equal to 2^8.

Lemma 4.1:

The above code construction gives a minimum distance $2t + 2$ among the code words.

Proof:

Let $v = (v_1, v_2, \ldots, v_k)$ and $v' = (v'_1, v'_2, \ldots, v'_\kappa)$ be two information symbols. They are encoded as

$$c = L \mid v \mid R \text{ and } c' = L' \mid v' \mid R'$$

respectively.

case1: $L \neq L'$ and $R \neq R'$

By construction R and R′ are code words from code with minimum distance $2\lceil \frac{t}{2} \rceil$ and L and L′ are code words from code with minimum distance $2\lfloor \frac{t}{2} \rfloor$. Hence $d(c_1, c_2) \geq 2t+2$.

case 2: $L \neq L'$ and $R = R'$

The datapart of R is $P_R(v)$ and the datapart of R' is $P_R(v')$. Since $R = R'$ and v, v' are constant weight vectors, we get

$1,\alpha,\alpha^3,..\alpha^{2\lceil\frac{t}{2}\rceil-1}$ as roots of the binary coefficient polynomial

$$\Sigma(v_i-v_i')X^i = 0$$

As α is chosen to be a primitive element of the Galois field, we have α^2, α^4, α^6, $...,\alpha^{2\lceil\frac{t}{2}\rceil}$ also as roots of the above polynomial. Using the BCH bound we can conclude that $d(v,v') \geq 2\lceil\frac{t}{2}\rceil + 2$, and since $d(L, L') \geq \lfloor\frac{t}{2}\rfloor$ thus we get $d(c, c') = 2t+2$.

case3: $L = L'$ and $R \neq R'$

Same as proof of case 2.

case4: $L_v = L'$ and $R_v = R'$

In this case we have

$\alpha^{2\lfloor\frac{t}{2}\rfloor},\alpha^{2\lfloor\frac{t}{2}\rfloor-1},...\alpha^{-1},1,\alpha,\alpha^2,..\alpha^{2\lceil\frac{t}{2}\rceil}$ as roots of the polynomial

$$\Sigma(v_i - v_i')X^i = 0$$

Once again using BCH bound, we get $d(c, c') = 2t+2$.

R and L are required to be code words from constant weight codes with minimum distance s_1 and s_2 respectively. Note that we can choose $P_R(v)$ and $P_L(v)$ as the dataparts of R and L respectively and can define recursively R and L using the procedure adopted in the case of the original datapart v.

In this code construction each code word consists of a data part and *t* check symbol vectors. Thus, encoding has *t* steps and adjoining a check symbol in each step reduces the minimum distance requirement by half, which is in contrast to Lin's construction of decreasing the minimum distance requirement by 2 on appending a check symbol in each step. This sharp reduction of minimum distance requirement at each step in the recursive code construction makes it a better code in terms of the information rate.

4.2. Encoding Procedure:

Encoding procedure for a constant weight code of minimum distance 2t+2 is listed in its various phases as follows:

1. Let M be the message space for the code. Choose integers k and w such that $\binom{k}{w}$ will be at least as big as |M|. Define a one-to-one mapping from the set M to the set V which consists of k-tuples of weight w. Let $v = (v_1, v_2, \ldots v_i, \ldots v_k)$ be the data vector obtained under this mapping.

2. if $t \geq 1$ proceed to step 3 else stop.

3. Let μ be an integer satisfying the condition $2^{\mu-1}-1 < k \leq 2^{\mu}-1$. Let α be a primitive element of $GF(2^{\mu})$. Let $s_1 = \lceil \frac{t}{2} \rceil$ and $s_2 = \lfloor \frac{t}{2} \rfloor$, $v = (v_1, v_2, \ldots, v_k) \in V$. Define one-to-one mapping P_R and P_L as follows:

P_R: V -> $\{GF(2^{\mu})\}^{s1}$ and P_L: V -> $\{GF(2^{\mu})\}^{s2}$

$$P_R(v) = \begin{bmatrix} P_1(v) \\ P_3(v) \\ \cdot \\ \cdot \\ \cdot \\ P_{2s_1-1}(v) \end{bmatrix} \qquad P_L(v) = \begin{bmatrix} P_{-1}(v) \\ P_{-3}(v) \\ \cdot \\ \cdot \\ \cdot \\ P_{-(2s_2-1)}(v) \end{bmatrix}$$

Here $P_i(v) = \sum_{v_j=1} (\alpha^j)^i$ for $i = -(2s_2-1), \ldots -3, -1, 1, 3, \ldots, 2s_1-1$. Note that P_R and P_L map a vector $v \in V$ into s_1 and s_2 tuples over $GF(2^{\mu})$.

Notice P(v) is a $\lceil \frac{t}{2} \rceil \mu$-tuple and Q(v) is a $\lfloor \frac{t}{2} \rfloor \mu$-tuple.

4. Let F_1 and F_2 be sets of $s_1\mu$-tuples and $s_2\mu$-tuples respectively. Choose integers q_R and q_L such that $\binom{q_R}{\frac{q_R}{2}} \geq |F_1|$ and $\binom{q_L}{\frac{q_L}{2}} \geq |F_2|$. Let U_R be the set of $\frac{q_R}{2}$-out-of-q_R and U_L be the set of $\frac{q_L}{2}$-out-of-q_L. Define a one-to-one mapping:

$$\Phi_R: F_1 \rightarrow U_R$$

Let v_R be $\Phi_R(P_R(v))$. Similarly define a one-to-one mapping:

$$\Phi_L: F_2 \rightarrow U_L$$

Let v_L be $\Phi_L(P_L(v))$. Append v_R on the right and v_L on the left to v as check symbols.

5. Repeat step 2 once with t as s_1-1, k as q_R and v as v_R and the second time with t as s_2-1, k as q_L and v as v_L.

We can notice that step 2 will be executed log(t+1) times in this code construction, and totally t check symbols are appended. Also by lemma 1 we can ensure that this recursive encoding procedure gives the minimum distance of 2t+2 between any two code words.

4.3 Decoding Algorithm for Recursive Code:

Let l be the weight of the code words. Suppose c = L|v|R is the code word sent and c′ = L′|v′|R′ is the code word received. Let v_R' be the first right hand side check symbol and v_L' be the first left hand side check symbol received. These are the dataparts of R′ and L′ respectively.

Step 1. if wt.(c′) > l+t or wt(c′) < l - t then we can detect that at least t+1 unidirectional errors have occurred. Stop.

Step 2 if v′ does not have the original weight or is not a valid vector go to step 4.

Step 3 Encode v′ to obtain c″. When d(c′,c″) ≤ t, v′ is the data part sent which has been received correctly, c″ is the correct code word. Stop. Otherwise go to step 4.

Step 4 If the first check symbol v_R' appended on the right of v has the incorrect weight go to step 6. Otherwise, let $P_R(v') = \Phi^{-1}(v_R')$. Using the BCH decoding algorithm, we can correct up to $\lceil \frac{t}{2} \rceil$ errors in v′P_R(v′). If v″ is the resultant of this decoding, encode v″ to obtain c‴. If d(c′,c‴) ≤ t then v″ is the data vector sent. Stop.

Step 5 If the first check symbol v_L' appended on the left of v′ has the incorrect weight go to step 8. Let $P_L(v') = \Phi^{-1}(v_L')$. Using the BCH decoding algorithm, we can correct up to $\lfloor \frac{t}{2} \rfloor$ errors in $v'P_L(v')$. If v″ is the resultant of this decoding, encode v″ to obtain c‴. If $d(c',c''') \leq t$ then v″ is the data vector sent. Stop.

Step 6 Decoding of c′ would have failed in the step (2) through (7) either when v and(or) the first check symbols on the left and right do not have the correct weights or there are more number of errors than the BCH decoding algorithm can effectively correct. If the decoding has failed in spite of the correct weights of the check symbols, we can once again apply t-error correcting BCH decoding algorithm to $P_L(v')v'P_R(v')$. If v″ is the vector obtained after decoding, encode v″ to obtain c‴. If $d(c',c''') \leq t$ then v″ is the data vector sent Stop.

Step 7 If the decoding has failed in all the steps (2) through (6) and if the weights of the first left and right check symbols are not correct we know there are errors in the received check symbols. By the virtue of the recursive construction of this code, we can correct the errors in the check symbols also using steps 2 through 6. When decoding fails in all the steps go to step 9.

Step 8 Repeat steps 4 through 6 with the check symbols obtained in step 7.

Step 9 Uncorrectable errors have occurred.

Theorem 4.1:

The above decoding algorithm is valid.

Proof:

Assume there are m errors in the received code word c′. Let $P_L(v') = \Phi^{-1}(v_L')$ and $P_R(v') = \Phi^{-1}(v_R')$.

Case 1: m = 0

No error occurred in c′. In step 3 we will obtain $d(c',c'') = 0 \leq t$, and hence it is decoded correctly.

Case 2. m > t and all the errors are unidirectional.

When all the errors are $0 \rightarrow 1$ then the weight of the received code word is greater than l+t or when all the errors are $1 \rightarrow 0$ then the weight of the $c' < l$-t. In step 1 we detect that there are more than t errors in c'.

Case 3: m < t; all m errors occurred in v', v'_R and v'_L

Suppose $P_L(v')v'P_R(v')$ has less than t errors.

In the presence of even a single error in the datapart, note we will obtain $d(c',c'') > t$. Hence decoding will fail in step 1.

When $m > 0$ and $m \leq \lceil \frac{t}{2} \rceil$, and all m errors are in $v'v'_R$, BCH decoding algorithm will correct up to $\lceil \frac{t}{2} \rceil$ in $v'P_R(v')$ in step 4. Thus we will obtain the correct datapart sent.

When $m > 0$ and $m \leq \lfloor \frac{t}{2} \rfloor$, and all m errors are in $v'_L v'$, BCH decoding algorithm will correct up to $\lfloor \frac{t}{2} \rfloor$ in $v'P_L(v')$ in step 6. Thus we will obtain the datapart sent.

When $\lceil \frac{t}{2} \rceil < m < t$, and all m errors are in $v'_L v' v'_R$ in step 8 we will obtain the correct datapart sent.

Case 4: $0 < m \leq t$, but the number of errors in $P_L(v')v'P_R(v')$ is greater than t.

In this case, there are some symmetric errors in v'_L and(or) v'_R, but the weight of these check symbols are not altered. This situation may lead to more errors in $\Phi^{-1}(v'_L)$ and $\Phi^{-1}(v'_R)$ and thus BCH decoding algorithm can no longer correct all the errors. Hence it is required to correct the errors in the check symbols before we apply the decoding to the datapart. Case 3 through 5 are applicable to the error patterns in v'_L and v'_R. Since either v'_R or v'_L has $< \lceil \frac{t}{2} \rceil$ errors, step 10 of decoding procedure decodes v'_R or v'_L or both correctly. When one or both of them is error free step 11 decodes the datapart correctly, since the number of errors in the datapart $< t$.

5. Comparison of the Recursive Code with Lin's Code

One can observe that the difference in the length of the code words in these two codes is largely dependent on the value of *t*. Also a greater percentage of the difference in code word lengths comes from the first few check symbols appended. The first two check symbols appended will have the same number of bits in both Recursive and Lin's code constructions. Thus, we can look at the lengths of the next four check symbols appended which will determine the lower bound on the difference in code word lengths. Assuming *t* is divisible by 4, the bit length of the four check symbols appended in the second recursive call of the Recursive code construction is $(t-2)\log((\frac{t}{2}).\log n)$, whereas in Lin's code construction they have

$$(t-2)\log((\tfrac{t}{2}).\log n) + (t-4)\log((\tfrac{t}{2}-1)\log((\tfrac{t}{2})\log n) \text{ bits.}$$

Hence the first four check symbols under Lin's construction method has

$$(t-4)\log((\tfrac{t}{2}-1)\log((\tfrac{t}{2})\log n)$$

more bits than the four check symbols of Recursive construction. Notice that the other check symbols appended in Lin's construction will also have more bits than the corresponding check symbols of Recursive construction. Thus the lower limit in the difference of the code word lengths is $\Omega(t\log t + \log\log\log n)$. Table 1 shows how the number of bits saved in the length of Recursive code word over Lin's code word grows as *t* increases for a fixed data vector size of 12.

k = 12

t	Length of the code word under Construction		Difference in the length of code word
	Recursive	Lin's Code	
2	18	18	0
3	35	35	0
4	46	46	0
5	56	61	5
6	66	76	10
7	84	100	16
8	102	124	22
9	118	151	33
10	134	178	44
11	154	210	56
12	168	242	74

Table 1.

6. Conclusion

In this paper we improved the code proposed by Lin [20] in terms of codeword length. We extended his recursive construction of the first two check symbols to all the check symbols which led to a reduction in the sizes of intermediate check symbols resulting in a overall reduction in the codeword size. We showed that the codeword in the new construction shrinks in length compared to that of Lin's code at least at the rate of $\Omega(t\log t + \log\log\log n)$.

References

[1] S. Al-Bassam and B. Bose, "Design of efficient balanced codes", Proceeding of the 19th Int. Symp.on Fault Tolerant Computing, pp. 229-236, June 1989.

[2] S. Al-Bassam, B. Bose, "On Balanced Codes", IEEE Trans. on Information Theory, Vol. IT-36, No. 2, pp. 406-408, March 1990.

[3] B. Z. Chor, *Two issues in Public Key Cryptography*. Cambridge, Ma.: MIT Press, 1986.

[4] J. M. Berger, "A note on Error Detection Codes for Asymmetric Channels", Information and Control, Vol. 4, pp. 68-73, March 1961.

[5] M. Blaum and J. Bruck, "Coding for Skew-Tolerant Parallel Asynchronous Communications", Research Report, IBM Research Division, July 1991.

[6] F. J. H. Bonick and H. C. A. Van Tilborg, "Constructions and Bounds for Systematic t-EC/AUED codes", IEEE Trans. on Information Theory, Vol. IT-36, No. 6, pp. 1381-1390, November 1990.

[7] J. Borden, "Optimal Asymmetric Error Detecting Codes", Information and Control, Vol. 53, No. 1/2, pp. 66-73, April/May 1982.

[8] B. Bose and T. R. N. Rao, "Theory of Unidirectional Error Correcting/Detecting Codes", IEEE Trans. on Computers, Vol. C-31, No. 6, pp. 521-530, June 1982.

[9] B. Bose and D. K. Pradhan, "Optimum Unidirectional Error Detecting/Correcting Codes", IEEE Trans. on Computers, Vol. C-31, No. 6, pp. 564-568, June 1982.

[10] B. Bose and D. Lin, "Systematic Unidirectional Error Detecting Codes", IEEE Trans. on Computers, Vol. C-34, No. 11, pp. 1026-1032, Nov. 1985.

[11] R. H. Deng and M. Herror, "DC-Free coset codes", IEEE Trans. on Inform. Theory, vol. IT-34, No. 4, pp. 786-792, July 1988.

[12] T. Etzion, "Constructions of error-correcting dc-free block codes", Abstracts of Papers of Int. Symp. on Information Theory, pp. 124, January 1990.

[13] 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) dmin = 4", IEEE Trans. on Magn., Vol. MAG-34, pp. 881-883, Sept. 1984.

[14] C. V. Freiman, "Optimal Error Detection Codes for Completely Asymmetric Binary Channels", Information and Control, Vol. 5, pp. 64-71, March 1962.

[15] S. Kawanishi, et. al., "DmB1M Code and its Performance in a Very High-Speed Optical Transmission System", IEEE Trans. Commun., Vol. COM-36, No. 8, pp. 951-956, Aug. 1988.

[16] D. E. Knuth, "Efficient Balanced Codes", IEEE Transactions on Information Theory, Vol IT-32, No. 1, pp. 51-53, January 1986.

[17] S. Kundu, "Design of non-systematic 3syEC/AUED codes of asymptotically optimal order", Proceeding of Int. Symp. on Information Theory, January 1990.

[18] S. Kundu and S. Reddy, "On Symmetric Error Correcting and all Unidirectional Error Detecting Codes", IEEE Trans. on Computers, Vol. 39, No. 6, pp. 752-761, June 1990.

[19] E. L. Leiss, "Data Integrity in Digital Optical Disks", IEEE Trans. on Computers, Vol. C-33, No. 9, pp. 818-827, Sept. 1984.

[20] M. C. Lin, "Constant Weight Codes for Correcting Symmetric Errors and Detecting Unidirectional Errors", presented in Int. Symp. on Information Theory, June 1991.

[21] K. Matsuzawa and E. Fujiwara, "Masking Asymmetric Line Faults using Semi-Distance Codes", Proc. 18th Int. Symp. on Fault Tolerant Computing, June 1988.

[22] D. Morris, *Pulse Code Formats for Fiber Optical Data Communication*. NY: Marcel Decker, 1983.

[23] D. Nikolos, N. Gaitanis and G. Philokyprou, "Systematic t-Error Correcting/All Unidirectional Error Detecting Codes",.IEEE Trans. on Computers, Vol. C-35, No. 5, pp. 394-402, May 1986.

[24] D. K. Pradhan, *Fault Tolerant Computing: Theory and Techniques*. Englewood cliffs, NJ: Prentice Hall, 1986.

[25] Y. Saitoh, et. al., "A method to construct constant-weight t-error correcting codes", in the 11th Sym. on Info. Theory and its Applications, Beppu, Japan, December 1988.

[26] J. F. Tabor, "Noise Reduction Using Low Weight and Constant Weight Coding Techniques", Technical Report 1232, MIT Artificial Intelligence Laboratory, May 1990.

[27] Y. Takasaki, et. al., "Optical Pulse Formats for Fiber Optic Digital Communications", IEEE Trans. on Communication, COM-24, pp. 404-413, April 1976.

[28] 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, pp. 1270-1275, Nov. 1971.

[29] A. X. Widmer and P. A. Franaszek, "A DC-Balanced, Partioned-Block, 8B/10B Transmission Code", IBM J. Res. Develop., Vol. 27, pp. 440-451, Sept. 1983.

[30] N. Yoshikai, et. al., "mB1C Code and its Performance in an Optical Communication System", IEEE Trans. Commun., Vol. COM-32, pp. 163-168, Aug. 1984.

[31] W. W. Peterson and E. J. Weldon Jr., *Error Correcting Codes, Second Editon*, The MIT Press, Cambridge, Massachusetts, 1971.