

## A Note on SEC/AUED Codes

Rajendra Katti

**Abstract**—A common method of constructing single error correcting/all unidirectional error detecting (SEC/AUED) codes is to choose a SEC code and then to append a tail such that the new code can detect all unidirectional errors. The tail is a function of the weight of the codeword. We present a technique to reduce the weight distribution of the SEC code so that the tail needed is smaller. We also present a new method to construct the tail.

**Index Terms**—Decoding, descending tail matrix, encoding, error correcting codes, unidirectional errors, unidirectional error detecting codes.

### 1 INTRODUCTION

THE problem of construction of error correcting/all unidirectional error detecting codes has received wide attention in recent literature [1], [2], [3], [4], [5]. Such codes find applications in fault detection in memory systems. In this paper we restrict ourselves to code construction only.

Let  $X$  and  $Y$  be two  $n$ -tuples over  $GF(2)$ . We denote the number of  $1 \rightarrow 0$  crossovers from  $X$  to  $Y$  by  $N(X, Y)$  [2]. The Hamming distance between  $X$  and  $Y$  is  $N(X, Y) + N(Y, X)$ . The next theorem [2] gives necessary and sufficient conditions for SEC/AUED codes.

**THEOREM.** A code  $C$  is an SEC/AUED code if and only if, for all distinct  $X, Y \in C$ ,  $N(X, Y) \geq 2$  and  $N(Y, X) \geq 2$ .

Next, we show how such codes can be constructed. The first step is to construct an SEC code. Given  $k$  information bits, this would result in a,  $(m, k, 3)$  code, where  $m$  is the number of bits in each codeword and the code distance is 3. The second step is to append a tail of length  $r$  to each codeword such that the code can detect all unidirectional errors. The length of the code is then,  $n = m + r$ . The tail is a function of the weight of the codeword. In this paper we develop a technique to reduce  $r$  by reducing the weight distribution of the code.

We now define a tail matrix and how this matrix can be used to construct an SEC/AUED code.

**TAIL MATRIX.** A descending tail matrix of strength  $s$  is a  $p \times r$   $\{0, 1\}$ -matrix with rows  $t_i$ ,  $0 \leq i \leq (p-1)$ , such that for all  $0 \leq i \leq j \leq (p-1)$ ,

$$N(t_i, t_j) \geq \min \{s, \lceil (j-i)/2 \rceil\}.$$

A  $p \times r$  descending tail matrix of strength  $s$  is denoted  $T(p, r; s)$ .

**SEC/AUED CODE.** Let  $C'$  be an SEC code of length  $m$  and let  $T$  be a  $T(m+1, r; 2)$  descending tail matrix with rows  $t_0, t_1, \dots, t_m$ . Let  $C$  be the following code of length  $m+r$ :

$$C = \{(v, t_{w(v)}) : v \in C'\}$$

where  $w(v)$  denotes the Hamming weight of  $v$ . Then  $C$  is an SEC/AUED code.

In [6] two techniques to reduce  $r$  are given. In this paper we give a technique to further reduce  $r$ . The technique relies on the replacement of the all-0 codeword with another codeword, which results in weight distribution reduction of 3. The resulting codes

are nearly systematic. Finally we present a method to construct tail matrices that relies on a search procedure.

Section 2 describes the construction of the SEC/AUED code. Section 3 describes the encoding and decoding algorithms for the code constructed in Section 2. Section 4 describes tail matrix construction.

### 2 CODE CONSTRUCTION

The code is constructed in the following manner. Let  $k$  be the number of information bits. Then:

- 1) Choose an  $(m, k+1, 3)$  or an  $(m, k+2, 3)$  code  $C'$  containing the all-1 vector, such that  $m$  is even and as small as possible. In this paper we construct  $C'$  as Hamming codes.
- 2) Choose a  $T((m/2) - 2, r; 2)$  descending tail matrix  $T$  with rows  $t_i$ ,  $0 \leq i \leq (m/2) - 3$ , and  $r$  as small as possible.
- 3) Let  $C$  be the code,

$$C = \{(c, t_{w(c)}) : c \in C', w(c) \leq m/2\}$$

$C$  is SEC/AUED since a subset of  $C'$  is also SEC. The main difference between the construction in [6] and the above construction is the replacement of the all-0 codeword from  $C'$  with another codeword. This results in a  $T(\lfloor \frac{m}{2} \rfloor - 2, r; 2)$  matrix instead of a  $T(\lfloor \frac{m}{2} \rfloor + 1, r; 2)$  matrix. This may in turn reduce  $r$  by 0, 1, or 2 bits.

**EXAMPLE.** Assume  $k = 7$ . To construct an SEC/AUED code we first construct the  $(12, 8, 3)$  Hamming code whose generator matrix is,

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

The resulting code  $C'$  has codewords of weights 0, 3, 4, 5, 6, 7, 8, 9, and 12. To obtain the SEC/AUED code as in [6], codewords of weights 0, 3, 4, 5, 6 are chosen resulting in a need for a  $T(7, r; 2)$  matrix. The smallest  $r$  possible in this case is 4. However since  $m$  (12 in this case) is even, all weight 6 codewords are not used. It is therefore possible to select one of these codewords to represent the all-0 codeword. Therefore choosing codewords of weights 3, 4, 5, 6 is enough for constructing the SEC/AUED code. This results in a  $T(4, r; 2)$  matrix and the smallest  $r$  now is 2. The length of the overall SEC/AUED code is thus 14 instead of 16. The length of the overall SEC/AUED code obtained from [1] is also 16. When  $k$  is 22, the length of the overall SEC/AUED code obtained from [1] is 35. The overall SEC/AUED code length from [6] is 34 and our method results in a code length of 33. Our method therefore results in a saving of 2 bits as compared to the method in [1] and a saving of 1 bit as compared to the method in [6], for  $k = 22$ .

We shall now show how to choose the weight 6 codeword, that will represent the all-0 codeword, such that decoding is easy. Below we show all the weight 6 codewords in  $C'$ .

#### Part 1:

```
001100101110
001101100011
001110001011
001111000110
010100001111
010101101001
010110101010
```

• The author is with the Department of Electrical Engineering, North Dakota State University, Fargo, ND 58105.  
E-mail: katti@plains.nodak.edu.

Manuscript received July 6, 1993; revised Sept. 17, 1993.

For information on obtaining reprints of this article, please send e-mail to: transactions@computer.org, and reference IEEECS Log Number C95188.

010111001100  
 011001000111  
 011001101100  
 011011001001  
 011011100010  
 100101000111  
 100101101100  
 100111001001  
 100111100010  
 101000001111  
 101001101001  
 101010101010  
 101011001100  
 110000101110  
 110001100011  
 110010001011  
 110011000110

**Part 2:**

001100111001  
 001101110100  
 001110011100  
 001111010001  
 010100110011  
 010101010101  
 010110010110  
 010111110000  
 011000011101  
 011000110110  
 011010010011  
 011010111000  
 100100011101  
 100100110110  
 100110010011  
 100110111000  
 101000110011  
 101001010101  
 101010010110  
 101011110000  
 110000111001  
 110001110100  
 110010011100  
 110011010001

These codewords have been split into two parts based on whether they have a 1 or a 0 in the least significant bit position of the information part of the codeword. If we number the bits in a codeword such that the left most bit is numbered 1, then the codewords in one half of the above codewords have bit-8 equal to 0 and codewords in the other half have bit-8 equal to 1. To form the SEC/AUED code we choose the following codewords from  $C'$ ,

- 1) All codewords with weights 3, 4, 5.
- 2) All codewords with weight 6 with bit-8 = 0.
- 3) Any codeword with weight 6 with bit-8 = 1 to represent the all-0 codeword.

It should be noted that appending two zeros to the information bits may sometimes result in a reduction of one bit in the overall size of the codeword. This implies that it is better to append just one zero and let the all-0 codeword remain in the code.

**3 ENCODING AND DECODING**

In this section we explain how to encode and decode the data.

**The Encoding Algorithm:** Let  $u = (u_1, u_2, \dots, u_k)$  be a vector of in-

formation bits such that at least one  $u_i$  is not zero.

- 1) Encode  $(u_1, u_2, \dots, u_k, 0)$  or  $(u_1, u_2, \dots, u_k, 0, 0)$  into a vector  $c$  in  $C'$  such that  $c$  contains  $m$  bits where  $m$  is even. Notice that two zeros may have to be appended to the information in order to make  $m$  even.
- 2) if  $w(c) > m/2$  then  $c \leftarrow c \oplus 1$ , where  $\oplus$  denotes "exclusive-OR" and the 1 denotes the all-1 vector.
- 3) Encode  $(u_1, u_2, \dots, u_k, 0)$  or  $(u_1, u_2, \dots, u_k, 0, 0) = (0, 0, \dots, 0)$  into a vector  $c$  in  $C'$  such that  $w(c) = m/2$  and  $u_{k+1} = 1$  if  $(u_1, u_2, \dots, u_k, 0)$  was encoded or  $u_{k+1} = u_{k+2} = 1$  if  $(u_1, u_2, \dots, u_k, 0, 0)$  was encoded.
- 4) Let  $v = (c, t_{w(c)})$  be the output of the encoder, where  $t_i, 0 \leq i \leq (m/2) - 3$ , are the rows of  $T$ .

The resulting SEC/AUED code is practically systematic since the first  $k$  bits in a codeword  $v$  will either be the information bits or their complements or a set of bits such that  $w(v) = m/2$  with  $u_{k+1} = 1$  (or  $u_{k+1} = u_{k+2} = 1$ ) that represents the all-0 information vector. The decoding algorithm is essentially as in [6], with an extra step to recognize the all-0 information vector.

**The Decoding Algorithm:** Let  $C$  be the SEC/AUED code. Let  $\bar{v}$  be the received word and  $\bar{c}$  the first  $m$  bits of  $\bar{v}$ .

- 1) Decode  $\bar{c}$  with respect to  $C'$ . If more than one error, declare an uncorrectable error. Else let  $c$  be the corrected word. If  $c$  is an all-0 vector then declare an uncorrectable error.
- 2) Let  $v = (c, t_{w(c)})$ . If  $d(\bar{v}, v) > 1$  ( $d$  denotes Hamming distance), then declare uncorrectable error.
- 3) Else, let  $(u_1, u_2, \dots, u_k, u_{k+1})$  (or  $(u_1, u_2, \dots, u_k, u_{k+1}, u_{k+2})$ ) be the  $(k+1)$  (or  $(k+2)$ ) information bits from codeword  $c \in C'$ . If  $w(c) = m/2$  and  $u_{k+1} = 1$  (or  $u_{k+1} = u_{k+2} = 1$ ) then the output of the decoder is an all-0 vector of length  $k$ .
- 4) Else, the output of the decoder is given by the vector of length  $k$ ,

$$u = (u_1 \oplus u_{k+1}, u_2 \oplus u_{k+1}, \dots, u_k \oplus u_{k+1})$$

The reader can refer to examples in [6] for further clarification of the above algorithms.

**4 TAIL MATRIX CONSTRUCTION**

In this section we give a method to construct tail matrices using a search procedure. The resulting matrices are better than those obtained in [1] and [6]. Let us assume that we want to construct  $T(m, r; 2)$ ,  $r$  is the number of bits in each row of  $T$ . Our objective is to obtain as many rows as possible in  $T$ .  $T$  is constructed in the following manner,

- 1) Partition all  $r$ -bit binary numbers into  $(r+1)$  sets  $s_i, 0 \leq i \leq r$ , such that,  

$$s_i = \{\text{all } r\text{-bit binary numbers, } c : w(c) = i\}$$
- 2) Let the rows in  $T$  be  $t_i, 0 \leq i \leq (m-1)$ . Then  $t_0 = (1, 1, \dots, 1)$  (the all-1 vector of length  $r$ ), and  $t_1 = (1, 1, \dots, 1, 0)$ .  $t_1$  contains  $(r-1)$  1s and one 0.
- 3) Select  $t_j, 2 \leq j \leq (m-1)$ , by searching through  $s_i$  (where  $i$  is as high as possible) such that the vector selected has as many 1s in common with  $t_{j-1}$  and  $t_{j-2}$  as possible and  $N(t_{j-1}, t_j) = 1, N(t_{j-2}, t_j) = 1$ , and  $N(t_i, t_j) = 2$  for  $0 \leq k \leq (j-3)$ .

By searching through  $s_i$ , where  $i$  is as high as possible, we construct a  $T$  such that  $w(t_i) \leq w(t_j)$  for all  $i > j$ . This implies that the lower the position of a row in  $T$ , the lesser is its weight. This will maximize the number of rows in  $T$ . Selecting  $t_j$  such that it has as many 1s in common with  $t_{j-1}$  and  $t_{j-2}$  will result in maximizing the number of rows that have a given weight in  $T$ .

EXAMPLE 1. Let us construct  $T(m, 4; 2)$ . Then,

$$\begin{aligned}
 s_0 &= \{0000\} \\
 s_1 &= \{0001, 0010, 0100, 1000\} \\
 s_2 &= \{0011, 0101, 0110, 1001, 1010, 1100\} \\
 s_3 &= \{0111, 1011, 1101, 1110\} \\
 s_4 &= \{1111\}
 \end{aligned}$$

Therefore,  $t_0 = 1111$  and  $t_1 = 1110$ . We now look for  $t_2$  in  $s_3$  as all vectors in  $s_4$  have already been used. The vectors that we can choose from, are those in  $s_3$ , namely  $\{0111, 1011, 1101\}$ . Any one of these vectors can be chosen as  $t_2$  as each has two 1s in common with  $t_1$  and three 1s in common with  $t_0$ . Let us choose  $t_2$  as 0111. 1011 and 1101 cannot be considered for finding  $t_3$  as  $N(t_0, 1011, \text{ or } 1101) = 1$  and we want  $N(t_0, t_2) = 2$ . Therefore, we search for  $t_3$  in  $s_2$ .  $t_3 = 0110$  is a vector in  $s_2$  that satisfies the conditions in step 3 above as,  $N(t_2, t_3) = N(t_1, t_2) = 1$  and  $N(t_0, t_3) = 2$ . Continuing in a similar manner we obtain,

$$T = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

If we had chosen  $t_2$  as 1101 then we would have obtained T as,

$$T = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Notice that the T matrix constructed has nine rows whereas the T matrix obtained in [1] has eight rows.

EXAMPLE 2. Let us construct  $T(m, 6; 2)$ . Using the above procedure we get,

$$T = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The above T matrix has 19 rows whereas the T matrix in [6] has 18 rows.

## 5 CONCLUSIONS

We have presented a technique that reduces the weight distribution in SEC/AUED codes. This results in smaller lengths of rows in tail matrices. We have also presented a new technique to construct tail matrices. Both these techniques result in shorter SEC/AUED codes.

## REFERENCES

- [1] M. Blaum and H. van Tilborg, "On t-error correcting/all unidirectional error detecting codes," *IEEE Trans. Computers*, vol. 38, no. 11, pp. 1,493-1,501, Nov. 1989.
- [2] B. Bose and T.R.N. Rao, "On the theory of error correcting/detecting codes," *IEEE Trans. Computers*, vol. 31, no. 6, pp. 521-530, June 1982.
- [3] D. Nikolos, N. Gaitanis, and G. Philokyprou, "Systematic t-error correcting/all unidirectional error detecting codes," *IEEE Trans. Computers*, vol. 35, no. 5, pp. 394-402, May 1986.
- [4] D.K. Pradhan, "A new class of error correcting/detecting codes for fault-tolerant computer applications," *IEEE Trans. Computers*, vol. 29, no. 6, pp. 471-481, June 1980.
- [5] D.L. Tao, C.R.P. Hartmann, and P.K. Lala, "A efficient class of unidirectional correcting/detecting codes," *IEEE Trans. Computers*, vol. 37, no. 7, pp. 879-882, July 1988.
- [6] J. Bruck and M. Blaum, "New techniques for constructing EC/AUED codes," *IEEE Trans. Computers*, vol. 41, no. 10, pp. 1,318-1,324, Oct. 1992.