

An Improvement on Constructions of t -EC/AUED Codes

Rajendra S. Katti and Mario Blaum

Abstract—A common method of constructing t -error correcting/all unidirectional error detecting (t -EC/AUED) codes is to choose a t -EC 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 t -EC code so that the tail needed will be shorter in some cases. The weight distribution span of the code is reduced by eliminating the all-zero codeword and replacing it with a codeword of weight greater than $2t + 1$.

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

1 INTRODUCTION

THE problem of constructing error correcting/all unidirectional error detecting codes has received wide attention in recent literature [1], [2], [3], [4], [5], [6], [7], [8]. Applications of these codes are in fault detection and correction in memory systems. In this paper we restrict ourselves to code construction only. Further motivation and background can be found in [2].

Let X and Y be two n -tuples over $GF(2)$. We denote the number of coordinates in which X is 1 and Y is 0 by $N(X, Y)$. For instance, if $X = 10100$ and $Y = 11001$, then $N(X, Y) = 1$ and $N(Y, X) = 2$. Notice that the Hamming distance between X and Y is $N(X, Y) + N(Y, X)$. The next theorem [4], [7] gives necessary and sufficient conditions for t -EC/AUED codes.

THEOREM 1.1. *A code C is a t -EC/AUED code if and only if, for all distinct $X, Y \in C$, $N(X, Y) \geq t + 1$ and $N(Y, X) \geq t + 1$.*

Next, we describe the most common method of constructing t -EC/AUED codes [3]. The first step is to choose a t -EC code. Given k information bits, this would result in an $[m, k, 2t + 1]$ code, where m is the length of the t -EC code and $2t + 1$ is its minimum distance. 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 present a method to decrease r in some cases by reducing the weight distribution span of the code. The method is particularly efficient for small values of m and relatively large values of t .

Next we define the tail matrix used to construct a t -EC/AUED code [3].

DEFINITION 1.1 [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 $0 \leq i < j \leq p - 1$,*

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

- R. Katti is with the Department of Electrical Engineering, North Dakota State University, Fargo, ND 58105.
- E-mail: katti@plains.nodak.edu.
- M. Blaum is with the IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose, CA 95120
- E-mail: blaum@almaden.ibm.com.

Manuscript received Sept. 1, 1994; revised Feb. 28, 1995.

For information on obtaining reprints of this article, please send e-mail to: transactions@computer.org, and reference IEEECS Log Number C95169.

We denote a $p \times r$ descending tail matrix of strength s by $T(p, r; s)$.

The next construction [3] provides a method for constructing t -EC/AUED codes.

CONSTRUCTION 1.1 [t -EC/AUED CODES]. *Let C' be a t -EC code of length m and let T be a $T(m + 1, r; t + 1)$ 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 a t -EC/AUED code.

In [1], two techniques to reduce r in some cases were given. In this paper we give a third technique to further reduce r . The method relies on the replacement of the all-0 codeword with another codeword, which reduces the weight distribution span of the code in $2t + 1$.

The next section describes the construction of the t -EC/AUED code.

2 CODE CONSTRUCTION

The construction described in this section depends on the replacement of the all-zero codeword by a codeword of weight $\lceil m/2 \rceil$ in code C' . The next algorithm gives the encoding procedure:

ALGORITHM 2.1 [ENCODING PROCEDURE]. *Let k be the number of information bits. Choose an $[m, k + 1, 2t + 1]$ systematic code C' containing the all-1 vector with m as small as possible and a $T(\lceil m/2 \rceil - 2t, r; t + 1)$ descending tail matrix T with rows t_i , $0 \leq i \leq \lceil m/2 \rceil - 2t - 1$, and r as small as possible. If \underline{u} is an information vector of length k , then \underline{u} is encoded as follows:*

- 1) If $\underline{u} \neq \underline{0}$, let $\underline{c} \in C'$ be the encoding of $(\underline{u}, 0)$ in C' . If $w(\underline{c}) \leq \lceil m/2 \rceil$ then encode \underline{u} as \underline{c} , otherwise encode it as $\underline{c} \leftarrow 1 \oplus \underline{c}$ (in words, as the complement of \underline{c}).
- 2) If $\underline{u} = \underline{0}$, then encode \underline{u} as a (fixed) $\underline{c} \in C'$ of weight $\lceil m/2 \rceil$ such that \underline{c} does not belong in the set of codewords obtained in step 1.
- 3) Append to the vector \underline{c} obtained in steps 1 or 2 the tail $t_{w(\underline{c}) - 2t - 1}$, completing the encoding.

Notice that C is t -EC/AUED since a subset of C' is also t -EC. The main difference between the first construction in [1] or the construction in [5] and the above construction is the replacement of the all-0 codeword from C' with another codeword. This results in a $T(\lceil m/2 \rceil - 2t, r; t + 1)$ matrix instead of a $T(\lceil m/2 \rceil + 1, r; t + 1)$ matrix. If m is even, since we are using only half of the codewords of C' in Algorithm 2.1, then, in particular, half of the codewords of weight $m/2$ are not assigned to information vectors and one of them can be chosen to encode the all-zero vector. If m is odd then no codeword of weight $\lceil m/2 \rceil$ is assigned to an information vector and any codeword with that weight can be chosen to represent the all-zero vector. The next example illustrates the construction method.

EXAMPLE 2.1. Assume that $k = 3$ and $t = 1$. Consider the $[7, 4, 3]$ Hamming code with systematic generator matrix

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

The codewords with weight at most 3 are

0000000
1000011
0100101
0010110
1001100
0101010
0011001
1110000

We replace the all zero codeword above by the codeword 0001111 which has weight 4. Using the descending tail matrix

$T(2,1;2) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, we obtain the code given by the following

codewords:

1000011 1
0100101 1
0010110 1
1001100 1
0101010 1
0011001 1
1110000 1
0001111 0

This code improves by one bit the code obtained in [5]. For instance, if we want to encode 011, it gives in a first step the vector 0110011. This vector has weight 4, so we take its complement, that is, 1001100, which has weight 3. We complete the encoding by adding the tail 1, so the final encoded vector is 10011001.

We observe in this example that the same result would have been obtained if an [8, 4, 4] extended Hamming code was used. Notice that the codewords above are codewords of weight 4 in the [8, 4, 4] extended Hamming code.

The decoding algorithm is essentially as in [5], with an extra step to recognize the all-0 information vector. The methods in [1], [3], [5] as well as our method, use different weight spans. No single method is the best in all cases. Which method gives the best result in terms of total redundancy (or, with the smallest overall value of n) depends on the particular value of k and on the descending tail matrices available. For instance, Table IX in [1] provides the parameters of some of the best descending tail matrices known. Our method often ties the best known values of n for a given k , and sometimes it even improves those values. Some values of n obtained by our method improving those in [1] are given in Table 1. The values of r were taken from Table IX in [1], except for $t = 2$ and $k = 238$ and for $t = 4$ and $k = 98$, in which the values of r were taken from Table VI in [5]. Notice that the values from Table IX in [1] have a quite limited range. Very likely, we can find other values of k for which our method improves or ties the existing ones once larger parameters for descending tail matrices are provided. The requirement that code C is systematic is not necessary, but it makes the encoding and decoding easier (in any case, all linear codes have a systematic representation). We use BCH codes (or shortened BCH codes) in all cases, except for $t = 3$ and $k = 11$, in which we use the Golay code. There is also an implicit assumption that codewords of weight $\lceil m/2 \rceil$ exist, but this is always the case with BCH codes and the Golay code [9].

TABLE 1
SOME VALUES OF T AND K
FOR WHICH OUR METHOD IMPROVES THE ONES IN [1]

| t | k | m | r | n | n from [1] |
|-----|-----|-----|-----|-----|--------------|
| 1 | 3 | 7 | 1 | 8 | 9 |
| 1 | 10 | 15 | 3 | 18 | 19 |
| 1 | 25 | 31 | 5 | 36 | 37 |
| 2 | 6 | 15 | 2 | 17 | 19 |
| 2 | 20 | 31 | 6 | 37 | 38 |
| 2 | 238 | 255 | 13 | 268 | 269 |
| 3 | 4 | 15 | 1 | 16 | 19 |
| 3 | 11 | 23 | 3 | 26 | 29 |
| 3 | 15 | 31 | 5 | 36 | 39 |
| 3 | 44 | 63 | 9 | 72 | 74 |
| 3 | 105 | 127 | 14 | 141 | 142 |
| 4 | 51 | 80 | 11 | 91 | 92 |
| 4 | 98 | 127 | 17 | 144 | 145 |

3 CONCLUSION

We conclude this paper by showing how saving even one bit in a t -EC/AUED code results in great savings in the cost of a memory system using such a code. Consider a memory system consisting of 2^{15} words. Let each word in memory be a codeword in the 3-EC/AUED code with $k = 15$ whose parameters are given in Table 1. In this case our code results in a codeword length of 36 bits whereas the construction in [1] gives a codeword length of 39 bits. Therefore, the memory system using our code would have 3×2^{15} bits less than the memory system using the code in [1] resulting in considerable savings in the cost of the memory system.

REFERENCES

- [1] S. Al-Bassam and B. Bose, "Aymmetric/Unidirectional Error Correcting and Detecting Codes," *IEEE Trans. Computers*, vol. 43, no. 5, pp. 590-597, May 1994.
- [2] M. Blaum, *Codes for Detecting and Correcting Unidirectional Errors* (reprint collection), IEEE Computer Society Press, 1993.
- [3] 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.
- [4] 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.
- [5] 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.
- [6] 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.
- [7] 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.
- [8] D.L. Tao, C.R.P. Hartmann, and P.K. Lala, "An Efficient Class of Unidirectional Correcting/Detecting Codes," *IEEE Trans. Computers*, vol. 37, no. 7, pp. 879-882, July 1988.
- [9] F.J. MacWilliams and N.J.A. Sloane, *The Theory of Error-Correcting Codes*, North-Holland, 1978.